Understanding Imagine 2.0

A *complete* Imagine reference

by STEVEN WORLEY

Understanding Imagine 2.0

Steven P. Worley

March 11, 1992

This manual was formatted in AmigaTEX (published by Radical Eye Software) on an Amiga 3000 computer. Many of the figures are screen grabs that have been included into the document directly as IFF images. Other figures are PostScript illustrations which have been automatically rasterized by AmigaTEX's PostScript interpreter, Post. The text was all inputted using GnuEmacs.

Contents

Preface

The evolution of the Impulse ray tracers (Silver—Turbo Silver—Imagine—Imagine 2.0) has been a long and laborious task. The latest incarnation, Imagine 2.0, a fourth generation modeling, animation, and ray tracing program, offers power and features not even available on high end platforms. This power comes with a small price, a steep learning curve to master what is a vast array of features. The Impulse ray tracers have brought more attention to the Amiga, and have had more articles/pictures published than any other renderer, and these features have much to do with that. The Imagine manual states many times that it takes time, time, and more time to master all that is offered. All too true, except for the efforts of a few masters of the program who can guide you through the process, uncovering esoteric aspects and giving valuable insight into the use of these features, saving you the *hundreds* of hours they have spent to master them. One such master is Steve Worley, known widely for running the Imagine Mailing List on Usenet. The first work I saw of Steve's was from the network and was a tutorial on one of Imagine's editors. I was very impressed and immediately called Mike Halvorson over at Impulse to get this guy on his team.

In coming to know Steve and his mastery of Imagine I can easily say he is one of the few people I would turn to for help with the program. In reading his book I have already learned tricks I hadn't thought to use... or have been inspired to try new tricks. In reading this book you will endow yourself with a greater understanding of how to get Imagine to do what *you* wish it to do. Most of us have predetermined ideas on what we'd like a particular trace to look like, what objects will appear, how they will look, etc. This *is* the hard part, but many of the answers lie here. I wish to congratulate Steve for his efforts in the community and the publishing of *Understanding Imagine 2.0*, and may it make an expert of you as well. Imagine is vast, but its capabilities are even wider. The information in the Imagine 2.0 manual and this book can make the sublime easy. Stick with it, you'll find that under all of those things to learn is ease of use, and enough special effects to keep the more ardent ray tracing enthusiast busy for a looonng time.

Louis Markoya
January 2, 1992.

Chapter 1

Introduction

You have just bought a complete guide to the software program Imagine 2.0, by Impulse Inc. This book covers all of the functions of Imagine 2.0 and provides a detailed guide to using the software to create your own pictures and animations. It also has a set of detailed appendices containing some interesting ideas and suggestions for your own projects.

1.1 Warning to New Users

Learning any piece of new software can be difficult, and the complexity of a three dimensional modeling program compounds this difficulty by orders of magnitude. Imagine in particular can be very tricky to learn. Imagine was the first piece of 3D software I had ever used, though I was very interested in learning about rendering. After two weeks of playing with the software, I was completely frustrated and confused, a state no other piece of software had ever put me into. Slowly I learned how Imagine functioned; suddenly I was past the "hump" and I was actually producing real pictures. Some programs have a steep learning curve, but Imagine's curve is so steep at the beginning it looks like a wall. *Don't give up!*

Now that I understand the software, I see why it is organized the way it is. It does not have an interface designed for simple understanding, instead it is optimized for maximum user control. Now that I understand the software well and use it constantly, there are few changes to the general interface that I could suggest to Impulse that would make Imagine easier for new users without removing some of the power and control that experienced users enjoy. The point is that once you understand the software, you'll be very happy with its configuration. By using this optimistic, long-term strategy, Impulse has made the software as powerful as possible, though at the expense of increased difficulty when you are learning.

1.2 About This Book

This book is a detailed guide to Imagine 2.0. It is a complete reference of all of Imagine's functions, and also provides supplementary material to help you learn how to *use* all of those functions. There are no follow-along tutorials included in this book; the amount of text already included is pushing the limits of easy publication.

The information that *is* included is as complete as I can make it. I have described every function in Imagine and its operation. I could have made a list of each menu option and describe those options with a paragraph apiece, but by talking about each command *in context* I feel the usability of the manual increases dramatically. For quick queries, I've tried to help your search for particular information as best I could. Every command is listed in the index in the back of the book, and the Reference Appendix gives a one or two line description of the command with a page reference for a detailed in-context description.

Since so much information is included, I've had a difficult time keeping the size of this book down. Beyond a certain number of pages, publishing gets difficult and costly, and the book itself becomes unwieldy and tougher to page through. Several times during the writing process I've had to shrink font sizes and reduce margins, and even so this book is well over 200 pages. Figures were perhaps the largest loss; I would have liked to use twice or three times as many to graphically illustrate different points, but since they take about a third of a page each I've been somewhat sparse with them.

Because so many commands work in combination, describing one function without referring to a related command can be very difficult. I've tried to cross-reference the text with marginal notes so that related information in a different part of the book can be found easily. For example, when discussing how to make an object act as a light source (a function of the Detail editor,) I cross reference the description with the section in the Action editor which describes all of the lightsource controls for intensity, color, shadowing and such. By having a note with a page number to point to closely related material, I hope to help you find the information you are looking for. Some chapters have many more cross-references than others; the Forms editor is very self-contained and has few margin notes, but the Stage and Action editors are so intertwined that I kept running out of margin with all of the references.

There is one chapter on each editor in Imagine, and an additional introductory chapter which describes Imagine in a broader scope. The "Basics" chapter explains the structure of Imagine (what *is* an editor, anyway?) and the basic methodology of how renderings are produced. It also describes specific tools that are common to *all* (or most) of the editors, such as the basic commands to move your viewpoint around in the world. While most of the chapters can be read in any order, new users should really start by reading the Basics chapter to get a sense of how the software works and how to use the common commands that all of the editors share.

The largest chapter is on the Detail Editor. It also includes a lot of background information that is useful in the later chapters of the book, so it is a good idea to read this chapter (or at least the first half) after you read the "Basics" chapter and before the others. The remaining chapters are self-contained enough that you can read them in almost any order.

This book is written for *all* users of Imagine. In each chapter, I try to start with few assumptions about your knowledge and present information starting from the basic premises and working to the complex functions. I've tried to write each chapter so that you can read it straight through and get a consistent, logical idea of how the editor works and what it is capable of. Even if you know Imagine well, try reading the chapters from start to finish, since you might pick up a lot of information that you didn't know about already.

The appendices have a different flavor than the reference text. The largest appendix is somewhat of a personal essay on general techniques and strategies to *use* the tools that the main chapters document. Just knowing every command's function won't give you the ability to make animations that look like the amazing effects in the movie *Terminator II.*[1] This appendix on strategy will suggest approaches to effectively using Imagine. While the main chapters are factual and can be trusted, this appendix is based on my personal experience. It is a set of hints, not an absolute set of guidelines on how things should be done.

Other appendices contain a variety of information. One contains a list of the most commonly asked questions about Imagine and quick answers to those questions. There is a chart of useful attributes for surfaces like chrome and paint. A short appendix lists pieces of hardware and software that can help you use Imagine and complement functions it performs. The command reference appendix gives a very quick summary of each command in each editor and points you to the section where it is described.

Physically, the book pages are large 8.5" by 11" because it is easier to incorporate clear figures with the larger paper size and there is more information per page (less page flipping!) There is a few pages of blank sheets at the back of the book for your own notes. I find that I am *always* writing down quick ideas, color combinations or whatever on scrap paper. If you use these spare pages, you'll always keep the information close at hand and at least locatable in the future.

1.3 About the Author

I am actually a relative newcomer to 3D software on the Amiga. I purchased Imagine 1.0 in late December of 1990, and after a rough couple of weeks learning the software, I was completely hooked. At the time, I was completing my final year at MIT, and I found that rendering was a serious competitor for my time. In January of 1991 I started an electronic mailing list over the Internet to discuss Imagine, which has grown in size to hundreds of subscribers who have posted over 2000 messages of Imagine questions, comments, and tips.

Since then, I've dedicated myself to learning the limits of the software and seeing how far I can push it. The quality of my pictures have increased considerably as I learned. Nearly all of my knowledge of Imagine is actually self-taught; Mike Halvorson is an advocate of experimentation as a learning tool, and I agree with him. You can discover a lot about a function and its abilities by just sitting down and playing with it. Some features, like "hinge," were baffling at first, until I experimented with it many times and thought about exactly what effects Imagine was producing. My knowledge of mathematics and reading general books on computer graphics also helped considerably. One or two of the tips I present are not original (like a section in the appendix on how to make teardrop shapes) but well over 95% of the information contained within this book came from using the program, Rick Rodriguez's original

[1]Well, to be honest, the appendices won't help you do that, either.

Imagine 1.0 manuals, and a *lot* of experience using the program.

1.4 Thanks

Several people helped me out with support, advice, criticism, taunts, jeers, and pizza. I'd like to thank Ed Chadez, Glenn Lewis, Jeff Peterson, Lou Markoya, and John Grieggs for their direct help, and Mike Halvorson and Steve Gillmor at Impulse for their support as well.

Also thanks to Dave Duberman for the idea of using invoice pockets for holding diskettes in the book.

Additionally, Thomas Rokicki of Radical Eye Software was especially helpful with the physical process of formatting this text. AmigaTEX is perhaps the finest typesetting language/system I know of. I'd hate to have to make all of the cross references (over 500!) in this book without it.

My greatest gratitude goes to my friend Evan Reidell. He introduced me to the Amiga, but much more importantly, he showed me how art, science, and especially creativity could work together in producing beauty in images, music, and art. Thanks, Evan.

1.5 Feedback

This book is as complete as I can make it. I plan to keep the text current by updating it every print run (expensive, but worth it), and the best way I know to add to it is to ask you, the people who already bought it, for your reactions. After you have read this book and used it, I would *greatly* appreciate any feedback, good and especially bad, which will help me build a newer version. Drop a postcard, letter or wad of $20 bills into the mail addressed to:

Steven Worley
Apex Software Publishing
405 El Camino Real Suite 121
Menlo Park, CA 94025

I don't hold giveaway lotteries for the people who respond, or promise you a free book or vacation cruise when you write. I will promise to respond to anything you have to say, and to thank you for any suggestions you have. I'm proud of the work I've done with this text, and by sharing it with you I hope to get a lot more people involved with this amazing program, Imagine.

If you truly enjoy this book and think that it is a useful reference, the best favor you can do for me is to tell your friends about it. Bring it to your user's group meeting, talk about it for a minute or two, and invite people to leaf through it. The more people who use Imagine and understand it, the more pictures, objects, and utilities will be out there for all of us to use.

Chapter 2

Imagine Basics

Imagine is in a special class of software that lets you create pictures and animations on your Amiga that have a special "real world" feeling to them. You are given the power to make a unique view into a universe of your own design. You define a 3D world, and Imagine can "render" any view of that world as if you had taken a picture with a camera at that position.

Imagine can be a difficult program to start using. It is not designed with the goal of being easy to learn; its goal was to provide the user with the most possible power in designing a scene or animation. This makes learning Imagine difficult, but this manual is designed to help the new users "learn the ropes" as well as provide experienced users with an organized reference manual.

2.1 Imagine's Organization

Imagine is organized by modules called *editors*. Each of these editors allow you to view and manipulate objects and your scene in different ways. Some editors let you place objects in scenes, or define how the objects change with time.

The Detail editor is where 3D objects are usually created and modified. It allows low-level editing of objects; you can add points and faces by hand, move them, delete old ones and in general be as picky as you like in adjusting every point.

The Forms editor also can create objects. It uses a special object creation method that can make the creation of some types of objects easy. Usually the objects that are created in Forms are loaded back into Detail for further editing.

The Cycle editor defines internal object motion. The objects can be "taught" how to move so you can make movement like a walking human figure or a clock that automatically swings its pendulum and turns its hands with time.

The Stage editor allows you to lay out scenes and prepare your animations. It loads the objects created in the three previous editors and defines the actual image to be rendered.

The Action editor is a companion to the Stage editor. It allows you to add some special effects as well as change some of the information about your scene, especially for animations.

The Project editor is where projects are initially created and where they are finally rendered. You start Imagine in this editor, and usually end there.

The Preferences editor is a small subprogram that allows you to define many parameters ranging from rendering controls to the default size of the background grid in the other editors. It is more of a utility than a constantly used function.

The process of making a rendered image usually involves using the Project editor to initialize a new world, using the Detail editor to build objects, the Stage and Action editors to lay the objects out, and the Project editor again to render the world. Everything really revolves around the manipulation of 3D objects, so it is important to understand exactly what they are and how they are formed.

2.2 Imagine Objects

When a computer program wants to draw a 3D object, it must have some way of internally representing it. Some modelers store each object as a bunch of 2 dimensional polygons and form a 3D object from a whole bunch of these polygons pasted together. A cube might be defined as six 2D squares arranged together, forming a hollow box like a cardboard carton. Since our final picture just has to *look* like it is solid, defining the outer surface of an object is usually all we need to do to make it seem as if the object *is* solid.

Any object can be defined as a bunch of flat polygons. Curved surfaces on an object like a sphere can use a lot of polygons in order to approximate the surface closely; certain computer tricks (including a very important one called Phong shading) can smooth out the surface even more. Most of the 3D objects (sometimes called models) that you've ever seen in any 3D computer graphic were defined as polygons. Sometimes advanced programs define surfaces with a mathematical equation or by a certain type of curved surface, and sometimes a computer model will have certain objects it "knows" how they should look (like a mathematically defined sphere or cone,) but most use polygons for defining objects, Imagine included.

All objects in Imagine are defined as a bunch of flat triangles. Nothing more. It is particularly easy for a computer to decide what a triangle would look like when viewed as a 3D image. Any complex polygon (like a square or octagon) can always be broken down into a bunch of triangles pretty easily. Having only one "shape" to deal with when building an object is actually a convenience for us, as we don't have to worry about questions of what type of polygons a certain object is made of, or how to convert one type of polygon into another. The computer likes dealing only with only triangles because it can optimize its renderer (the part of the program that actually draws the pictures) to expect and deal with just one simple shape instead of 246 different ones.

Although an object is made of only triangles (called *faces*) it has points and edges which define where these faces go. If you think of a simple triangle, it has 3 defining points at the corners, three edges connecting these points, and one face which actually makes up the body of the triangle. Imagine can better deal with the objects by defining these sub-parts (points, edges and faces,) and it allows us to manipulate the objects much more easily.

Every object has a number of defined *points*. Imagine understands an *edge* to be a line segment that connects any two of these points. A face is defined by naming the three edges that make it up. Instead of storing nine numbers for each face (the X,Y,Z location of each corner of the triangle) it just names the edges, which in turn identify the three points. This reduces the size of a description of an object considerably. It also helps in editing objects since if you move a point, each face that contains that point will adjust itself to the new location of the point. The other alternative would be to have each face manually manipulated individually, which is obviously a big pain.

As an example, think of a simple flat square. Imagine would store the square as two triangles that share one edge (the diagonal of the square.) The square would actually contain *five* edges (the four sides and the diagonal) and *four* points (one at each of the corners.) It would have two faces. A cube is stored as twelve faces, formed by eighteen edges, which are in turn defined by eight points.

This definition of objects actually gives us some extra leeway in how we define our 3D model. Imagine doesn't require your object to be connected at all; that is, your object could be two completely separate surfaces that never touch. You might want to make an object like a logo with 3D letters spelling out a name. The letters don't actually touch to form one solid connected object; they are independent from each other. Imagine doesn't care; you can call any collection of points, edges, and faces an object. Imagine also gives you tools for splitting off part of an object (like removing a letter from the logo) or joining two parts together.

Since this is a computer model and not a physical one, we can violate physics and have objects intersect each other. You might overlap two spheres half-way and join them together to form one object. You'll only see the outer surface when you render the new double-sphere object. Being able to intersect objects like this is very handy, since you can hide ragged edges of an object *inside* another, and you can make joints with no gaps.

There actually are two objects that Imagine does not define as a group of points, edges, and faces: a perfect sphere and an infinite plane. These are the only exceptions to the normal definition of objects in Imagine. Well, all right, there's another. An axis containing *no* points can still be manipulated as an object. It won't show up when rendered, but sometimes it's nice to use a lone axis as an invisible object in certain cases. You can also use the axis as the start of a brand new object.

Defining objects point-by-point is obviously not very suited for creating complex objects, sometimes with *thousands* of points. There are more powerful commands that let you modify your object in more global ways. You can add pre-made "primitive" objects like a cylinder or a torus (doughnut shape.) These primitive objects have the points, edges, and faces that form it already defined. There are certain tools that let you draw an outline, say the profile of a chess pawn, which is converted to a three-dimensional 'spun' object, as if it was chiseled out on a lathe. Other tools let you slice off parts of your object using knives that you can build yourself. In general, object creation is done with these powerful tools, and picky touch-ups are the only time you grab and move individual points. A sculptor does not glue sand grains together!

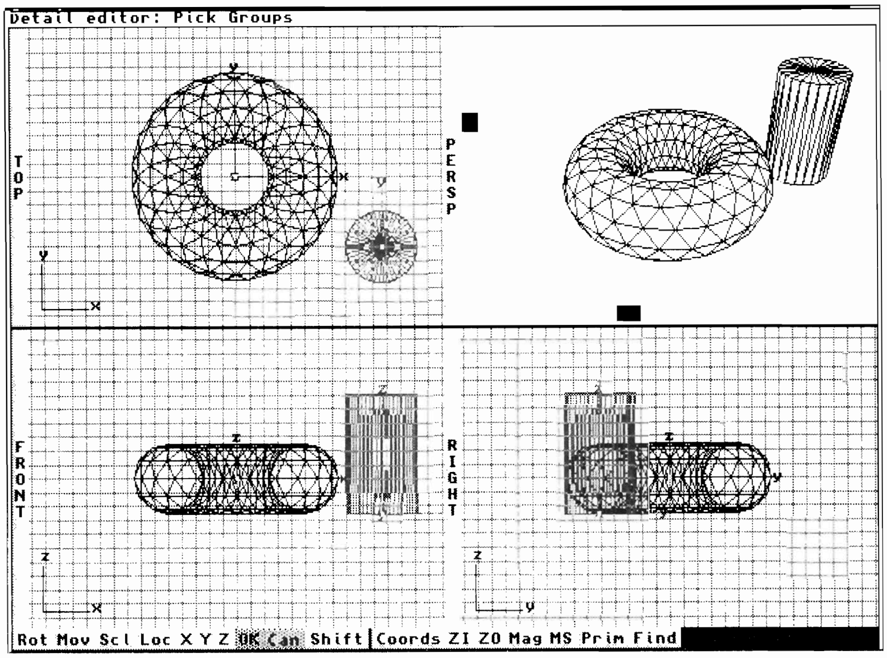

Detail editor: Pick Groups
TOP
PERSP
FRONT
RIGHT
Rot Mov Scl Loc X Y Z OK Can Shift Coords ZI ZO Mag MS Prim Find

Figure 2.1: A standard view from Imagine

2.3 Common Editor Tools

The creation of objects in Imagine is certainly not as intuitive as the process you used in kindergarten to squeeze clay into vague animal shapes. Instead of a real life view of the object, you get a complex quadview display. Instead of using your grubby fingers, you have a very 2D mouse. Most importantly, if you wanted to change the shape of a clay model, you just pushed and pulled on it. In Imagine (and any other computer modeler) you have to use a very specific set of tools to manipulate your objects. No matter how powerful these tools are, they are still going to limit the ways you can manipulate your creation.

2.4 Your View of the World

In each of the main editors where you manipulate objects (Detail, Stage, Forms, and Cycle,) you'll see what is known as a "Quad-View." A typical view is shown in Figure 2.1. There are four windows labeled "Top", "Front", "Right", and "Perspective", which are different ways of viewing the object you are manipulating. It is difficult to manipulate 3D objects with a 2D mouse and a 2D screen, and the quad-view is a compromise that makes the best of these unfortunate 2D restrictions.

The top, right, and front views show you the wire-frame skeleton of the object you're editing. A wire-frame is a view of your object with each edge shown as a line segment. Faces are *not* shown, so the object looks like it's built from pieces of wire that join at the outside edges of the object, hence the name wireframe. Wireframes have two advantages; they are much faster to draw than "solid" models, and since you can see *into* the object, you can manipulate points and edges on the interior of the object that you wouldn't normally see.

The title bar of the editor reminds you what editor you are in and occasionally displays useful information, such as what frame of an animation you are viewing. The gadgets at the bottom of the screen are user definable controls that are a substitute for menu commands. Imagine unfortunately does not support AREXX, but users can at least add their favorite commands to a button on the bottom of the screen to at least partially customize the program.

The majority of the screen is taken up by the four views of the world. The Top, Right and Front views are just that; a wireframe view of your object shown from the three orthogonal (right angle) directions. There is also a small axis at the bottom left corner of each view that shows the world's X, Y, Z coordinate system. In Imagine, the X, Y, Z is defined just like it is in mathematics; X is left to right, Y is in to out, and Z is down to up.

Some 3D programs define Z to be in-and-out, so note Imagine's difference.

There is an absolute *world* coordinate system defined by these axes. You can select the menu command "Coordinates" from the Display menu, which will continually display the coordinates of the mouse pointer in the world's X, Y, and Z system. The units that it measures in are arbitrary, but it is often convenient to call them "Imagine Units." Objects tend to be on the order of 10 to 100 Imagine Units in size, since this is a comfortable scale to deal with when we design scenes to be rendered.

There is often a grid overlaid on the three main windows. This grid is used to give you a sense of scale, and can be turned on or off in the Display menu. The spacing between the lines can be set by choosing "Grid Size", also from the Display menu. The default is 20, which is a reasonable starting size. Some commands let you use the grid to snap objects to precise locations, which is the main reason you might want to change the grid size. You can turn the grid on and off by using the "Grid On/Off" command from the Display menu.

The fourth window (which never displays a grid) is called the "perspective" window. It allows you to view your object from *any* direction. You can also change modes to view your object as a wireframe or as a "solid" model, where the faces become opaque so that you cannot see through your object. In this window, you *cannot* manipulate your objects; it is a view only.

Each of the four windows can be quickly zoomed to take up the full screen very easily by merely clicking on the tall narrow box to the left of each view that contains the name of the window. The window will expand to take up the entire screen, allowing you to have a better view of whatever you're working on. To zoom back to the quad-view, just click on the name to the left again. To go immediately from a full screen display of one view to a full screen display of another, you just click the name of the new view to the right. Being able to see all four views at once is often an advantage, but so is seeing a larger, more detailed view. This method allows you to quickly and easily change how you look at your model.

Perspective, the remaining view, also shows a wireframe view of the your world. You can change the view by manipulating the two white sliding boxes on the bottom and left of the window. The bottom white slider lets you view from different directions *around* the object. If the slider is in the middle, you're looking at the front. If it's 3/4 of the way to the right, you're looking at the right hand side, and if it's all the way in either direction, you're looking at the back. The vertical slider on the left controls the *angle* you're looking at the object from. Centered is a level perspective, all the way up gives you a straight-down view, and all the way down gives you a straight-up view. By combining these two sliders you can look at your object from any direction.

You can change the appearance of the Perspective view by selecting "Wireframe" or "Solid" from the Display pull-down menu. Solid takes longer to show your object, but removes the points that are hidden, getting rid of the X-ray wireframe view. A final way of changing the perspective view is by selecting "Shaded" from the Mode pull-down menu and zooming the perspective view to the full screen. This shades the object in false black and white colors which sometimes lets you see the shape of the object more clearly.

There are a few commands that let you change your absolute vantage of your object. You can zoom your view (on all windows) in and out by using "Zoom In" and "Zoom Out" from the View menu. This lets you see more of your object at once, or just a certain portion. Each zoom in or out will double or halve the scale respectively. You can also select a numerical zoom by selecting "Set Zoom" in the View menu, which allows more precise magnification levels by simply typing in a number. Zoom in and zoom out are often used, so knowing the keyboard equivalents of right-Amiga-i and right-Amiga-o can save a lot of time.

To scroll the views around, you can click in one of the three main views, then use the arrow keys to move the view in whatever direction you like. You'll notice that if you change one view, the others will change as well; all of the views are linked so they show the same volume of space. You can also scroll the view by telling Imagine where you want the view centered. You select "Re-center" from the View menu and click on where you want the new center of your view to be. Usually you click right in the middle of the object or area you're interested in. The keyboard equivalent of right-Amiga-. (period) is very convenient.

One important display option is found in the Display menu; it is called "Interlace." Interlace will change the screen resolution which the display uses. An interlaced screen is 400 pixels high, whereas a non-interlaced screen is only 200. Unfortunately, the interlaced display will flicker on many Amigas. An Amiga 3000 or a "flicker-fixer" equipped Amiga will be able to use interlaced mode without the flicker. The interlaced mode allows much more detail and more precise location of points, so it is by far the preferred mode to work in. Even if you do have a flickering display, it is probably worth the annoyance to have the extra resolution.

There are a couple ways to reduce interlace flicker. You can muck with the monitor's contrast and brightness, or you can change the screen colors to something less contrasting using the Preferences editor. My favorite solution is wearing sunglasses; believe it or not, it works very well, and you look cool while using your computer.

2.4.1 Moving Objects

Knowing how to move your views around is important, as when you're manipulating an object you'll find yourself changing your viewpoints around constantly. There is a whole new set of commands for moving the *objects* in the editor around.

When you are viewing an object in your world, you'll notice that you can see each point and edge in the wireframe. In addition, you'll see an *axis*, usually near the center of the object. In Imagine, *every object has its own independent axis.* An object's axis helps Imagine determine which way an object is facing, how it is scaled, and even what its position is. Imagine doesn't understand what the objects *are;* it doesn't realize that a complex object like an airplane should orient itself with wheels down instead of balanced sideways on a wingtip. The axis actually defines the object's position; if you ask Imagine to move an object, Imagine really just moves the axis, and the object's points, edges, and faces are dragged along with it. When you rotate an object, the rotation occurs around the object's axis, as opposed to the world's absolute reference system. Scalings, where you change the size of the object, also use the object axis as a basis.

When you want to manipulate a certain object, you have to tell Imagine which one (or ones!) that you're interested in, since you might have a dozen different object loaded at once. The way of choosing an object so you can manipulate it is just by clicking on its axis. The object will turn a pretty blue color (or sometimes purple; more later!) which indicates that the object is chosen; any manipulation commands will be done on this one object. The object is said to be "picked," and Imagine knows that you want to apply commands to this object as opposed to another.

Once you've picked an object, the most common manipulations are to move it around, rotate it, or scale it. These basic commands are often used, so Impulse has made it pretty easy to do. When you have a selected an object, you type the letter 'm' for "Move," or click on the small gadget at the bottom of the screen labeled "Mov." The object will disappear (!) and be replaced by a big yellow "bounding box" which encloses the volume where your object was. This bounding box represents the size, shape, position, and orientation of your object. Since the box is so simple to draw, Imagine can update it in realtime as you manipulate it, allowing you to position it quickly and easily.

After selecting the object and pressing 'm', Imagine knows you want to move the object. Putting the cursor in any of the three main views, pressing the left mouse button and then dragging the mouse will drag your object in the direction you move. You do not have to click on the yellow box; anywhere in the view is fine. You can keep moving the object as long as you like; you can let go of the mouse button, move the pointer to another position in any of the three views, and continue moving the object. You are also welcome to zoom in and out, make one view full-screen, or re-center your views at any time. When you are finally done moving your object, pressing the space bar or clicking the "OK" gadget will accept the change and your object will be displayed as a wireframe in its new location. If you've made a mistake, you can press the ESC key or the "Cancel" gadget instead. This also exits the move mode, but the object's position is unchanged from where it was before you started to move it. This is obviously useful for fixing mistakes or changing your mind.

Two other commands work much like move: rotate and scale. If you select your object and press 'r' or click on the "Rot" gadget at the bottom of the screen, you will be able to rotate your object, and you'll see the yellow bounding box spin as you drag the mouse with the button down. You can also change spin axes (to pitch or bank the object, as opposed to yawing it) by pressing 'x', 'y', or 'z' or clicking on the small "X," "Y," and "Z" gadgets at the bottom of the screen to define which axis you want to rotate around. All rotation is done around the *object's* axis.

Scaling is done by pressing 's' or clicking the "Scl" gadget at the bottom of the screen and dragging the mouse. Again, scaling is done relative to the *object's* axis. If the axis is in the center of the object, the object will grow in all directions. If it is at the bottom, the object will grow up and out, but not down.

Each of these three commands (move, scale, and rotate) can be called either when you've picked an object or during any other move, scale, or rotate command. For example, you might pick an object, press 'm' to move the object, position it in a new place, press 'r' to spin it, then 's' to scale it. You do not have to press the space bar after every change; only after you are finally satisfied with the new location, size, and orientation of your object do you want to press the space bar to accept the changes you've made. Aborting by using the ESC key or the "Can" gadget (for Cancel) will remove all of the changes (movements, rotations, and scalings) that you've made.

These manipulation commands are easy to use, and they have other controls that make certain manipulations even easier. The 'x', 'y', and 'z' commands that allow you to change rotation axes also work in moving and scaling. They act in these two modes as toggles; when you start a move, you are free to move it in all three directions, X, Y, and Z. You might want to restrict a direction of motion, though, if for example you are moving a table along a floor and you didn't want to accidentally lift the table into the air as you moved it left and right. Pressing the "x", "y", and "z" keys will toggle the allowable directions on and off, so pressing "z" will anchor the table's height, and pressing "z" again will allow you to lift it up if you change your mind. This also works in the scaling mode; if you want to make an object narrower without changing its height, you might toggle "z" and scale the object down. With the "z" toggle off, the object will maintain its Z height, but will shrink in the X and Y directions. At any time, the gadgets at the bottom of the screen highlight the directions that are "active."

A related shortcut is using the capital letters 'X', 'Y', and 'Z', which set the toggles to allow movement and scaling in one direction only. If you wanted to lift a table straight up, you just type 'Z' and the table will be free to move up and down, but not in the X or Y directions. This method of setting the toggles overrides whatever position they were set in before, but you can use the individual toggles afterwards to set whatever freedoms you like.

Imagine gives you even more flexibility if you want to use it. Whenever you move, rotate, and scale an object, it is based on a certain coordinate system. The default is to use a fixed coordinate system, defined by the set of axes that is fixed in place and shown at the bottom left of the three main views. This is called the "world" coordinate system. However, each object has its own "local" coordinate system, defined by its own axis. Imagine allows you to use a local coordinate system instead of the world system if you like.

For example, if you have an object in the shape of a plane, the local coordinate system probably has the Y axis (going front to back) in line with the main fuselage of the plane. Using "r" to rotate the plane, you can easily position it so that it is angled up like it is climbing into the sky. If you then wanted to move it in a straight line along its "flight path", the direction it's pointing, you could select move, and try to judge by eye the new position in the direction in front of the plane. If, instead, you select local mode (by using "l") and restrict motion along the Y direction by typing "Y", the plane will move smoothly along the line it's pointed along. In the world coordinate system, it's moving in both the Y and Z directions, but in its local coordinate system, it's moving only in its Y direction.

To switch between coordinate systems, you just type 'l' and 'w' whenever you want to change, or select the "Local" gadget. The current coordinate system will highlight the "Local" gadget at the bottom of the display display when you are using local coordinates. Many times the local and world coordinate systems will be the same (they will be aligned in the same direction), so one is equivalent to the other.

One final option when you're manipulating objects allows you to manipulate the axis of the object independently. If you want to move, scale, or rotate an object's axis (without simultaneously affecting the object!) you can use 'M', 'R', and 'S', the capital letter versions of the object manipulation commands, to affect only the axis. You can also use the "Shift" gadget instead of the keyboard equivalent. There are some occasions you might want to do this for fancy tricks, but most of the time, you just want to move the axis around just so that it lies near the center of your object.

The standard commands to move, rotate, and scale objects have been streamlined for ease of use since they are performed so often. Sometimes, however, they are somewhat lacking, especially when you need precise control over how your object is to be manipulated. For the precise control of object manipulation, Imagine has a special command called "Transformation" which allows you to numerically control your object as opposed to judging by eye.

The transformation command works much like the standard interactive commands in that you first select your object (by clicking on its axis) and then tell Imagine what to do to it. To enter the transform command, you click on the object (it becomes blue or purple) and pull down the menu item "Transformation" from the Object menu. A small requester will appear. You have six options you can choose from: translate, rotate, scale, position, alignment, and size. You also enter X, Y, and Z arguments.

Translate takes the X, Y and Z arguments and moves (translates) the object that distance.

Rotate will rotate the object around the axis you specify by an amount (in degrees) you specify in X, Y and Z. Performing more than one rotation at once is legal, but it is easy to make mistakes in final orientation. If you rotate around more than one axis at once, the Z rotation is performed, then the X rotation, then the Y rotation.

Scale will scale your object by a certain factor. To double the size, just enter 2 in each of the X, Y, and Z boxes. A negative number is completely legal, and if one or three of the scalings is negative, you'll actually get a scaled mirror image of your original object.

Position is like Translate in that it moves your object. Instead of moving a certain distance, however, it moves the object to the absolute world coordinates you specify.

Alignment is also absolute; it will rotate your object in whatever way necessary to align in the direction you specify, regardless of the original orientation. Setting X, Y, Z all to zero will make the object line up exactly with the world axes.

Size is again absolute. It uses the axis size as a benchmark, and will scale the object (and its axis) to an absolute size. The "default" size that all axes start out at is 32 Imagine Units, so entering an XYZ size of 32 32 32 will bring most objects back to their virgin sizes.

To use any of these sub-commands, just click on the box next to its name and type in the appropriate X, Y, and Z arguments in the boxes to the right. Selecting "OK" will perform the manipulations, "Cancel" will abort without affecting your object.

You have the option to use world or local coordinates, just as in the interactive commands; just click on either box to decide. The default is the world system. You can also manipulate only the axis (like the capital letter commands in interactive manipulation) by selecting "Transform axis only."

Most manipulations use the interactive controls, and the transformation requester is used only for accurate, measured changes.

One problem that you may run into after an interactive or a transformed manipulation is a "dirty" screen. Imagine erases the old object from before your move or scale or rotate, and draws it in the new position. However, to save time, it will not redraw any other wireframe object that happen to be in view. This means that the areas where the old object intersected any other object in the view will be blank; part of the other object will be erased. If you want to check to see if this is the case, you can select "Redraw" from the Display menu, which will redraw all of the objects, eliminating the problem. One case where this is almost necessary is when you have multiple copies of an object at the same location. If you move one copy, the other isn't redrawn after the first copy is moved away. Since it was in the exact same location as the old, erased, object, it looks like it has disappeared! This is easy to fix with redraw. It is another oft-used command, so knowing the keyboard equivalent of right-Amiga-r is handy.

A problem you'll run into when manipulating complex objects is the sheer time it takes to redraw the wireframe model (in three views). Imagine has a way to speed the display of these object; it shows the bounding box of the object (like the one shown in interactive manipulation) instead of the wireframe. You *lose* the detailed view of your object, but you can still see the position, size, and orientation of the objects. To make an object "Quickdraw" in this mode, you can use three commands in the Functions menu. "Quickdraw All" will make all of the objects display in quickdraw mode. "Quickdraw None" will make all objects display the normal wireframe. "Quickdraw Pick" will make your picked (blue or purple highlighted) object display in quickdraw mode. These quickdraw boxes are very handy, and since they can be toggled at any time in the Detail Editor, it makes sense to use them when screen updates start to get too slow.

2.4.2 Pick and Select

Since you can have many objects loaded at once, there has to be a way for you to tell Imagine what object or objects you want to perform your commands on. You've done this already by clicking on an object's axis and watching it turn color. This shows that the picked object is ready to be manipulated on.

What if we want to manipulate more than one object at a time? A standard way to "multi-pick" things (like icons in AmigaDOS, or objects in Imagine) is to use the shift key. By holding the shift key as you click on objects, Imagine knows you want *a bunch* of objects picked, not just the latest one. In fact, if you press the shift key, the display line at the top of the screen will change to show how many objects are picked. Commands will usually affect *all* of the picked objects, not just one. In the case of moving, scaling, and rotating more than one object, the *first* picked object's axis defines the basis of all the manipulations, as well as the local coordinate system for manipulating all of the objects.

There are easier way to pick many objects than by repeatedly clicking on each object's axis. Imagine allows you to change how objects are picked by the "Pick Method" submenu in the Modes menu. The default is "Click," which means that when you click directly on an object's axis, it will become picked. Other methods of picking can be chosen from the pick method submenu. If you use "Drag Box," instead of clicking on the object axes, you should press and hold the mouse button while dragging the mouse. A large box will follow your mouse, and when you release the button, an object within the box will become picked. If you press and hold the shift key when you release the mouse button, *all* of the objects within the box will become picked.

Lasso is similar, but more versatile. You press and hold the button while drawing a large circle or oval or squiggly shape. When you release the button, an object within the region you've drawn will become picked. Again, you can hold the shift key to pick *all* of the objects within.

A final option in the pick method submenu is called "Lock" . Lock isn't a method of picking; it really has more to do with when moving picked objects. Lock is a flag; you can toggle it on and off by selecting it from its submenu. When Lock is on, any moved object will snap to the nearest grid location when released. This is automatic and is easier than using the one-time "Snap to Grid" (described later, I promise!) again and again when you're trying to get precise placement.

Actually, this isn't true. Only points will snap to the grid; a bug in the current version of Imagine doesn't snap objects to grid intersections.

Two other utility commands can be found in the Pick/Select menu. "Pick All" will pick *all* of the objects in your workspace. "Unpick Last" will allow you to remove the last object you picked from your set of picked objects. This is handy when you pick one too many objects and you want to unpick the last one you chose.

It is easy to pick objects or sets of objects using the different pick methods. There is actually another powerful way to change what object or objects are picked; it is called "select." There is a very, very important difference between a "picked" object and a "selected" object; you've been using pick to highlight objects and manipulate them. Select is sort of a pick-wanna-be.

There is a solution when picking (or unpicking) objects becomes awkward (or impossible!). *Select* allows you to control what objects are picked by allowing you to add and remove objects from your set of picked objects one at a time.

One common problem that can occur is when two object axes are directly on top of each other. If you click on the common axis location, one of the objects will become picked. (The first one that was created or loaded into the Editor). If you click again, the same object will remain picked and the second object will just sit there. If you hold the shift key and click on the common axis again, the second object *will* be picked, but now *both* objects are picked. If you want to pick just the second object and not the first, you can either *move* one object just to uncover the other axis, or you can use select.

Think of buying lunch at a cafeteria, and you pick which food you want to eat. One way of "picking" food to add to your tray is by having the lunch worker point to each of the cafeteria's food bins, and saying "No, the next one, the next one, the next one; yes! That one!" as the worker points to the foods in turn. As the worker selects item after item, you can choose to *pick* the item he's pointing to at any time. The analogy extends; What if your arms are full of cafeteria food and you want to put some back? Your arms are busy holding all the food; you can't easily grab an item and put it down. You can, however, ask a friend to "unpick" the item for you. If your friend has trouble with big words (like "soup,") he can just point at each food in your arms in turn until he points to the granola yogurt you want to put down. You then say "Yes, yes! *Please* get rid of that!"

This is exactly what select allows you to do. Your arms are full with picked objects. You can't just click on an object to "Unpick" it because Imagine thinks you're just making sure you have it picked. You also might have problems indicating the right object to pick, as in the case of two objects on top of one another. The major difference between the cafeteria and Imagine is that your mentally challenged friend is also the cafeteria worker, and will point to both types of objects for you.

Select works by allowing you to highlight different objects in a controlled way. These highlighted objects will show in different colors, allowing you to tell the state of an object. A "selected" object might be picked or not. In the default color scheme, a normal object is black, a selected object is orange, a picked object is blue, and a picked *and* selected object is purple.

These colors are the defaults; you can change them in the Preferences editor.

Only one object can be selected at once, which is helpful in reducing confusion. The commands for selecting objects are completely different from those of *pick*ing objects; the whole point of select is that sometimes the methods used to pick objects are awkward, and select gives you an alternative way to pick them.

The easiest and most common method of selecting an object is by using two commands, "Select Next" and "Select Previous," both found in the Pick/Select menu. Using "Select Next" repeatedly will cycle through all of the objects in the order that they were created or loaded. This command does *not* care whether the object is picked or not; it will select all objects one at a time. "Select Next" is often a command you want to repeat, so knowing the keyboard shortcut of right-Amiga-n is almost necessary. By repeatedly using "Select Next," *any* object can be selected because Select next will eventually reach it. "Select Previous", right-Amiga-b (for back), will select objects in the opposite order, in case you overshoot with "Select Next." One convenience is that when an object becomes selected, your view will jump to center the object on the screen, always allowing you to see what you just selected.

When an object is selected, there are certain commands that will cause it to become picked or un-picked. The most common command is called "Pick Select," which can be found in the Pick/Select menu. When you use this menu option, the selected object will become picked. If the selected object is picked and you want to un-pick it, you can use "Unpick Select" from the pick/select menu to unpick it.

Pick Select is used a whole *lot. The F1 key defaults to performing this command.*

"Select Next" is sometimes painfully slow (you have to step though every object!), especially if you know exactly what object you want to select. One quick command that is sometimes useful is "Home," which selects the very first object you created or loaded into the Editor.

Two other useful commands to quickly select specific objects are "Find by Name" and "Find Requester", both found in the Functions menu. "Find by Name" allows you to type in an object's name (assigned in the Attributes requester, more later) and the object will become selected and your view will shift to center on the named object. In addition, the object becomes selected, allowing you to pick-select or unpick-select it. The "Find Requester" does the same thing except it displays the names of all of the currently loaded objects, and you just click on the name you want to select. This requester is also useful because it tells you the size (# of points, edges, and faces) of each object, which is an excellent judge of object complexity. It's also fun to say "Cool! My tomato has 1,821 points!"

A final command that is handy to use in all of the editors doesn't actually move or manipulate objects. It is a way of getting a higher quality display of the view you see in the Perspective view. The command "Quickrender" in the Project menu will actually render the image you see using whatever rendering method you wish. Imagine will ask if you want to add a lightsource, which is probably a good idea if your object or scene has no lights already. The type of rendering that Quickdraw performs is defined by the Preference editor. The image that is rendered is saved to RAM: by default, and you are given the option to delete the picture after you view it. Page 132 discusses the different Quickrender options.

When Imagine displays the Quickrender picture, you can press a button or press the mouse button to go back to the editor. Unfortunately, Imagine seems to buffer your keystrokes and mouse clicks, so sometimes when the render finally finished, you might see your Quickrender flash onscreen for an instant, then it's gone. This is especially common if you move to a second screen to use another program while Imagine is rendering (Amiga multitasking is great!). There is no easy way to get Imagine to redisplay the picture, but you can load the file from RAM: and display it from ADPro or using Iview. Another option would be to rename the Quickrender file into *Pic.0001* and placing it into a subproject's directory, then using the Project command "Import" to make it a viewable frame. None of these solutions are really satisfactory, but it's just about the only workaround that gets around this problem.

Import is discussed on page 128.

Chapter 3

The Detail Editor

The Detail Editor allows you to manipulate and modify objects in Imagine. Like the other editors (and any Amiga program, for that matter) Imagine gets input and directions from you by either moving the mouse and clicking its buttons or by typing on the keyboard. Most advanced options use pull-down menus to select the function you want to perform. An important trick, especially when you start using Imagine a lot, is keyboard equivalents. This lets you select menu items via the keyboard by pressing the right Amiga key along with another letter or number. All of the keyboard equivalents can be selected via pull-down menus, although not all menu items have keyboard equivalents. You'll find that learning the most used commands' keyboard equivalents can save a *lot* of time. Its quick and easy to punch right-Amiga-o to zoom your view out; pulling the menu down repeatedly is a pain. A few other commands (especially moving, rotating, and scaling objects) use the keyboard to indicate what you want to do (move, rotate, or scale) while simultaneously using the mouse to control the extent of the transformation.

A neat trick for learning keyboard equivalents is discussed on page 145.

You can get into the Detail editor from any point in Imagine by selecting the menu item "Detail Editor" from the Project pull-down menu. The screen should then split into the quad-view display discussed in the previous chapter.

You should definitely read the Basics chapter before attempting to use the Detail editor. The basic commands for manipulating your viewpoint are basic to using any of the editors, and a knowledge of basic object structure is also assumed.

3.1 Groups and Hierarchies

With complex models, sometimes you don't want to make one huge, gigantic object to represent the entire model. You might want to make a forest object that has 20 trees in it, and it seems silly to carve the whole thing out of one block. Or, you might be building an object that is logically a bunch of separate parts, like a clock with a face, a pendulum, two hands, and a frame.

Another important ability you might want is to be able to give different parts of a complex object different attributes, or colors. Imagine lets you color and define the look of your objects in different ways, and you can even tell it to make different parts of the same object look different. But when you're building something like a window, the glass panes are considerably different than the wood frames; it is easier to define each as a separate object then somehow group them together.

There is a function that lets you do exactly this: group objects together. When you have a model that you want to make (and keep!) in separate sections, Imagine allows you to establish a group of objects which will stay together. It allows you to treat the group as an entire ensemble (if you want to move everything, or apply a command to the whole set), or you can pick out one particular object and deal with it independently.

Grouping is very easy to do. If you want to group two objects together, you click on one object, then press the shift key and click on the other. (Remember that this is just the method of picking more than one object at once.) When you have multi-picked the objects, you select "Group" from the Object menu. A purple line will appear joining the axes of the objects. The first object that was selected becomes the "parent" of the group. If you group more than two objects, the purple "group" lines all run from each "child" object to the parent object. This lets you see what objects are grouped to one another, and helps identify the parent of the group. Sometimes it is nice to assign a lone axis as the parent of a group, especially when no part of a group lends itself to being a parent.

Splitting a group back into its component parts is also easy; just pick the group by clicking on the parent. The entire group will become picked, and selecting "Ungroup" from the Object menu will split the group. The purple joining lines will disappear, and each child will be independent again.

Once a group is made, it can be treated almost identically to an ungrouped object. You can pick it (by clicking on the parent) and the entire group will become highlighted. You can then move, scale, or rotate the entire group as a whole. If you click on a *child* object, the child will be picked, but not the group. You can then move, scale, or rotate it independently of the group, assign it individual attributes, or perform a command on it independently of the rest of the group. Even when you move the child object around, it will *stay* grouped; you must use "Ungroup" to ungroup objects. There are modes where you can pick parents separate from their children; this is described in the next section.

You can even make groups of groups. Or groups of groups of groups. This is done in exactly the same manner as before; you can pick one group, multi-pick a second, and group them. Having these multi-layer groups is sometimes very useful, for example, in modeling a human figure. You might make a finger group that contains all of the knuckles, a hand group including a palm, four finger groups, and a thumb group, an arm group consisting of a hand group, a wrist, a forearm, and an elbow, and a body group consisting of a head group, a torso, two leg groups, and two arm groups. This kind of nested grouping is called a "hierarchy," something like an ordering of your grouped objects like a family tree, where the body is the great-granddaddy of a knuckle. One great advantage of setting up a hierarchy is obvious when you want to move an arm. You pick the arm, and rotate it around the shoulder. All of the arm's children follow it, so the arm moves as a whole. You do *not* have to move 15 knuckles, a palm, a wrist, a forearm, and so on. If you want to adjust a finger, you can manipulate it and the knuckles will move together, but the arm will be unaffected. If you move the main parent body group, everything follows along as if the body were just one solid object, as opposed to dozens of parts. Hierarchies are obviously well suited for complex models.

Groups are useful when you have sub-parts of an object you want to keep together. Sometimes grouping simple objects is still useful even if there is no hierarchy to follow, since the individual objects are free to move apart from the parent, and can easily be assigned different attributes.

For example, if you're designing a human face, you might cause the eyeballs in the head to be an additional grouped object as opposed to just molded into the main face. Later, if you wanted to change the eye (make it a different color, or replace it with a different type of eye you can easily select the eye and change or replace it. This advantage compounds the other advantages of grouping; you can later animate the eyes looking in different directions, and you can easily change the attributes or texture of the eye while leaving the face undisturbed.

Chrome eyeballs! Cool!

3.2 Loading, Saving, and Other Object Manipulations

There are many useful commands that act on picked objects other than just simply moving, rotating, and scaling them. Two of the most practical are "Load" and "Save." "Load" will load a new object in from disk; it will display a file requester which you can choose the filename of the object to load. The most common place to put objects are in your *objects* subdirectory in your project directory.

See the Project Editor description of Imagine file structure on page 129.

"Save" will take the picked object or group and save it as a file onto disk. Note that *groups will save all of the objects in the group in the same file*, as long as you have the whole group picked by clicking on the parent. If you pick a child of a group and save it, you save *only* that object (and its children), and *not* the entire group it belongs to. You can give the saved object or group any name you want, but I suggest using a filename extension of *.iob* to help identify it as an Imagine object.

3.2.1 Add

You can also have Imagine create several simple types of objects for you. Instead of loading a object from a file on disk, Imagine can algorithmically create several simple shapes such as tubes and spheres. The command "Add" in the Functions menu has many options for building new "primitive" objects.

"Add Axis" will add a simple axis to your world, which can form as the basis of a new object. "Add Sphere" will add a perfect sphere object to your world. "Add Ground" adds an infinite plane object. Each one of these options will add the object into the world at the X, Y, Z location 0, 0, 0.

Imagine comes with several simple pre-built objects called primitives that are very convenient to use as starting points for creating your own objects. To make a primitive object, select "Add" in the Edit menu, and "Primitive" in the sub-menu. There are six simple shapes that Imagine will automagically create for you: a sphere, a cylinder, a cone, a disk, a plane, and a torus. When you select one, Imagine will ask how many points the object should have.

With primitives like a sphere, the more points that define it, the smoother its appearance is going to be when rendered. Remember that even curved surfaces are made from triangles, and the surface becomes better defined with more points. However, an object with more points than are necessary can become a burden; drawing the object in the editor takes more time, and although the final rendered picture with be higher quality with extra points, it will also take longer and use more RAM to render. Thus, when you add new primitive objects, Imagine asks what level of detail you would like.

For example, the sphere primitive asks how many circle sections and how many vertical sections will make it up. The default is a reasonable number of defining points. If you were looking for a higher quality sphere because you were going to zoom in very closely to it, you might use extra points. If the object is going to sit in the background and not be examined closely, you might select fewer points. Most of the time, the defaults serve as a nice compromise, but you are much more likely to simplify the object as opposed to increase the default level of detail. The plane primitive in particular lends itself to simplification; most of the time you can bear with defining the simplest plane possible (2 triangles) as opposed to the overburdened default of a grid containing 200 triangles.

Each primitive lets you define the numbers of points that define it; the parameters that you can vary are all pretty self-explanatory. For example, the cylinder lets you define how many points are to form the circle around the rim, and also how many sections the body of the tube should be defined as. Other options (available for some primitives) are simple flags that define whether to close the ends of the cylinder (to create a hollow tube versus a log) or to 'stagger points' in some models. Staggering points increases the smoothness of curves; you should almost always leave it on. Note that the disk and the plane are actually flat objects; the others all have depth. All objects also let you define their size, though of course they can be scaled interactively or numerically after they have been made.

3.2.2 Utility Commands

Another command you can apply to picked objects is "Snap to Grid" from the Functions menu. It operates on all picked objects, moving each of them so that their axis lies on top of the nearest grid intersection. This is very useful in trying to line up objects or for precise positioning. This is much like a one-time "Lock." It will also make individual points snap to the nearest grid locations when they are manipulated. (Point manipulation will be discussed soon!)

Lock is described on page 15.

There are a few other utility object commands. "Cut", "Copy", and "Paste" are found in the Object menu. "Cut" will remove your picked object from the Imagine world and store it in memory. When you select "Paste," the object will be re-inserted into the world at the same place it was prior to the cut. In fact, the object is *still* retained in memory, so you can move the restored object around and use "Paste" again to get a second copy to manipulate. You can repeat "Paste" as many times as you like to produce multiple copies of objects. "Copy" is like "Cut," except the object is not removed from the world after being copied to memory. You can again use "Paste" to add multiple copies to the world.

Since the pasted objects are all put in the same location, often you'll have to move one copy to get to the next. Judicious use of "Redraw" can help in showing exactly what copies are still floating around. The newly pasted objects are automatically selected, so using "Pick Select" to immediately pick it is awfully convenient.

Pick Select is discussed on page 16.

An useful command for making complex objects is called "Join," which can be found in the Functions menu. If you pick two or more objects, "Join" will assemble them into one single object. The new conglomerate object will have use the axis of the first object that was picked, and will contain all of the points, edges, and faces of *both* (or all) of the joined objects. Joined objects are difficult to unjoin later, so only use it when you *want* a solid object. Join is a fairly common command; you might build a car with the body sides, and join on a side mirror, then join the roof on, then join the floor. Remember the advantages of groups though; you probably *don't* want to join the tires to the car; if you instead group them you can rotate them later, as well as define the chrome hubcap separately from the car's paint and the rubber tire.

"Merge" is also found in the Functions menu. It is more of a utility command. It will remove any duplicate faces, edges or points in your object. Especially after you "Join" objects, you might have a lot of points lying on top of one another. Merge removes these extra, unneeded points, speeding rendering and even display in the editors. "Merge" also helps Phong shading work properly.

Phong shading is described on page 32.

"Delete" is a pretty obvious command, and is also found in the Functions menu. When you use "Delete," every picked object will be removed from the world. This command is used a lot to get rid of cruft and deadwood, so knowing the keyboard shortcut of right-Amiga-d is useful.

As with all of the editors, Imagine has one level of "Undo," which can be selected from the Project menu. When using dangerous commands like Delete, being able to recover from the command is important. "Undo" will work with almost any command. You can also "Undo" an "Undo," reinstating a command you decided you wanted anyway.

3.3 Modes

The basic commands to pick and move objects and view the world and everything in it are very important, and are used constantly. The actual work you perform in building objects depends on the user changing the view and manipulating the objects almost without thought.

No matter how good you are at manipulating objects and changing the view, using these commands will never *build* an object. To do this, Imagine has different *modes* that it performs different actions in. The most common modes allow you to manipulate objects and groups. Other modes let you pick and manipulate not objects, but the *points* of an object, or the edges, or the faces. Still other modes let us drag points around in different ways. Some let us add *new* points, edges, and faces.

Aha! So that's how we can build our own new objects!

These modes are easy to change; you can just pull down the Mode menu and select which mode you would like to be in. The current mode is always displayed in the status line at the top of the screen; this is often handy when you get confused about what you're doing. The keyboard equivalents for changing the current mode all use the right-Amiga key and a digit; this makes the keypad become a "mode selector" if you don't want to use the pull-down menus and have the stuff it takes to remember which digit is which mode. Personally, I don't have the stuff, so I usually lumber along with the pulldown menu rather than strain my poor brain.

3.3.1 Pick Modes

The default mode is "Pick Groups", which means that whenever you click on a group, it will be picked. (Simple!) If you want to pick individual objects, *even if they are the parent of a group,* there is another mode called "Pick Objects." Just select it from the mode menu, and when you click on any object (in a group or not, child or parent) it will be picked. You can obviously multi-pick single object by using the shift key. When you are dealing with ungrouped objects, "Pick Groups" and "Pick Objects" work identically.

Different modes let you deal with the different parts of an object. "Pick Groups" and "Pick Objects" always deal with entire objects at a time. You can rotate, scale, and move them, add them, group them, and delete them, though you can't affect their basic structure. The remaining modes all work on *parts* of objects, not objects themselves. Using these new modes, you can add extra parts to an object, or change the shape and configuration.

One important note is that to even enter these other modes, you must have selected at least one object (or group) for the new modes to act upon.

You'll also find that I consistently lied to you in part of the Basics chapter. I always referred to picking objects as opposed to picking anything else, like faces. *All* of the pick and select commands except "Find" work equally well in picking faces, edges, or points as opposed to just objects or groups. Most other commands like "Delete" will work on the parts of an object as well.

One new mode is "Pick Points." If you pick an object or group and enter the pick point mode, the object will turn black (the object is *not* picked anymore!) and its points will all become visible (they will show up as small squares.) Now you are in a different mode; you are no longer picking and selecting *objects*, you are dealing exclusively with points. You can then click on the points which will turn orange as you pick them. You can use the shift key to multi-pick, or the lasso and drag box to grab many points at once. You can also select points, and use all of the selection tools to help you get any subset of points you want. Selected points are green, picked points are orange, and picked and selected points are yellow.

All of these colors are definable with the Preferences editor.

When you're picking points, edges, or faces, Imagine will work *only* with the points, edges, or faces in the object that was picked before you chose the "Pick Points" (or Edges or Faces) mode. This prevents you from confusing one object's points with another's. When you scroll around your view or redraw the screen, the other objects won't even be updated, so don't get scared if they seem to disappear. When you re-enter "Pick Objects" or "Pick Groups" mode, all of the objects will re-appear.

Picked points are easy to manipulate. If you use the standard "Move," "Rotate," and "Scale" commands you can interactively adjust sets of points independently of the rest of the object. When you scale and rotate points, they do it around their own center and not the object's main axis. For example, if you pick a bunch of points on the rim of a circle, when you scale the points the radius of the circle will increase. When you rotate the points, they rotate around the center of the region containing the points, and *not* the axis of the object.

You can also use the "Transformation" command for exact control, where you have the option of using the axis of the object or the center of the points to base scaling and rotation.

Just because you can pick something doesn't mean you can perform every command on it. In the case of points, you can delete your picked points or manipulate their position. You cannot do things to selected points that make no sense (like grouping them, or saving them to a file); that's just weird and Imagine will probably just ignore the command.

The Transformation requester is described on page 13.

You can perform some other commands that aren't applicable to objects as a whole, however. For example, a very useful command is called "Split." It takes the picked points, removes them from the original object, and gives them their own axis. In effect, the original object is split into two parts defined by the points you picked. Any connecting faces or edges are deleted (two objects do *not* share!). This might be very useful, for example, if you have a logo and want to pull one letter out of the object to do something special with it.

When you delete a point, you delete any edges and faces that the point is a part of.

One command that is unique to pick points mode is "Taut," which is found in the Functions menu. If you select three or more points and select "Taut", the middle points will jump to the line segment defined by the first and last points. This command might be useful to line up a bunch of points in a straight line quickly. "Taut" does *not* work with anything other than picked points.

Picking edges is similar to picking points, except you obviously use "Pick Edges" mode. To pick an edge you just click on the two points that make it up, or lasso or drag box the entire edge. Just like points, you can't perform every command on them, but you can delete them and split them. When you delete an edge, you delete any faces the edge was a part of, but you do *not* delete the two points that formed the edge. You *cannot* translate or move edges or use taut on them.

A new command you cannot perform on points but can use on faces is called "Fracture." This command is in the Functions pull down menu, and is often very useful. The fracture command will take and break each edge into two edges, with an additional point added to the midpoint of each picked edge. This command is very useful when you need to increase the detail level at a certain area of an object; the extra edges that appear allow you to manipulate them to add finer details and structures.

"Pick Faces" is again pretty straightforward. You must click on *all three* of the points that make up the face to pick it. Fracture works very well on faces; it splits each face (one triangle) into four triangles defined by the midpoints of the three edges. The new faces can then be manipulated independently.

Deleting faces removes the faces, but not the edges or points that it was made up of.

Picked faces allow you to characterize an object's appearance in local areas. The attribute requester normally allows you to give the object overall color, reflection, and transparency values. You can actually set these for every single face, if you like. You can pick one or more faces, select "Attributes" from the Object menu, and use the sliders to set the color, transparency, and filter values for the face or faces.

Object appearance and coloring faces is discussed in detail on page 35.

You will *not* see any change in the appearance of your object when you do this, but when you render the object, the faces you selected will all override the default object color with the attributes you selected. A danger is that face attributes are somewhat fragile. If you join or merge objects or start deleting or adding points to it, all face coloring is often lost. To keep this from happening, color individual faces *last*, just before saving your object.

One final command is occasionally useful when you want to reorder a set of points, edges, or faces in an object. If you want to change the order that Imagine stores the items (points, edges, or faces), you can pick them in the order you want them, then use the "Sort" command in the Pick/Select menu. This is used to determine what order items are selected in, and occasionally it is useful to change it, especially when using the "Pick Range" command.

Pick Range is explained on page 35.

3.3.2 Add Modes

Three additional modes are "Add Points," "Add Edges," and "Add Faces." "Add Points" will add an additional point to your object in the location you click on. "Add Edges" lets you click on *two* existing points and will add a new edge joining them. "Add Faces" mode will let you add a new face to an object by clicking on the *three* points that make it up.

"Add Lines" mode is a convenient combination of "Add Points" and "Add Edges." As you click, a new point is added in the location you point to, and further clicks will add additional points along with an edge joining the latest point to the one that was immediately preceding it. A few clicks around the border of a rough circle will make a set of points with the edges following the outline of that circle. Carefully clicking on the location of an existing point will cause the new line to connect to that point, so making closed shapes is possible.

3.3.3 Drag Points

"Drag Points" mode allows you to interactively drag individual points in your object around. If you select this mode, you can click on any point and drag it to a new location interactively by holding the mouse button down. Any edges or faces that this point is connected to will follow the point to its new location. This is similar to picking the points and using the interactive move command, but in this mode you don't have to keep hitting "m." Still, it's somewhat of a redundant mode.

The only real advantage of Drag Points is Magnetism. See page 59.

Dragging multiple points is also easy; just use the shift key, multi-pick the points by clicking on each in turn, and when you want to start dragging them, just release the shift key.

An important technique: What if you want to select a point or points in one view, and drag them in an orthogonal direction? For example, you have a plane defined by a horizontal 10 by 10 grid, and you want to pick a bunch of points from the middle and pull them up. If you click on the points from the top view, you can easily pick any of the points you're interested in, but you can only drag them left and right, forward and back. You want to be able to drag them *up*.

Here's the method for doing this: it is invaluable, so remember it. Whether you want to move one point or a hundred, press the shift key to multi-pick the points. Click on the points you want to move in *any* view, keeping the shift key depressed. To move all of these points, *keep the shift key depressed* and move the mouse to the view where you want to move the points in. Press and hold the left button, then *release* the shift key. The picked points will move with your mouse for as long as you keep the button held down. Releasing the button will anchor the points.

In the example with the 10 by 10 horizontal grid, you would press shift, click on the points you want in the top view, move to the front (or right) view, press the left mouse button, release the shift key, move the points around, and finally release the mouse button. That's it!

Magnetism, a more complex way of dragging points, is discussed on page 59.

One problem with manipulating points, edges, and faces is picking the *right* point. When the object is complex, the wireframe displays can get very cluttered. There is a convenient way of simplifying a view to get points out of your view; it is a mode called "Hide Points." In "Hide Points" mode, any points you select (with click, drag box, or lasso) will disappear from view; they won't be drawn. They still exist, they just aren't displayed and can't be picked or manipulated. You can "Hide" whatever points that get in the way of your work area, then change modes (usually to "Pick Points,") and manipulate the non-hidden parts of your object. Selecting "Pick Objects" or "Pick Groups" will make the hidden points re-appear. You have to start out in "Pick Points" mode or you cannot enter "Hide Points."

For example, if you're working on a helicopter model and you want to work on the rotor alone, you might pick the object and enter "Pick Points." Then you would select "Hide Points" mode, and use the lasso to indicate the main helicopter body. The rotor is left alone, and after changing into something like "Drag Points" or "Pick Faces" mode, it is easy to indicate what portion of the rotor you want to deal with without accidentally modifying the helicopter body. Selecting "Pick Objects" mode makes the entire helicopter, with the rotor changes, reappear.

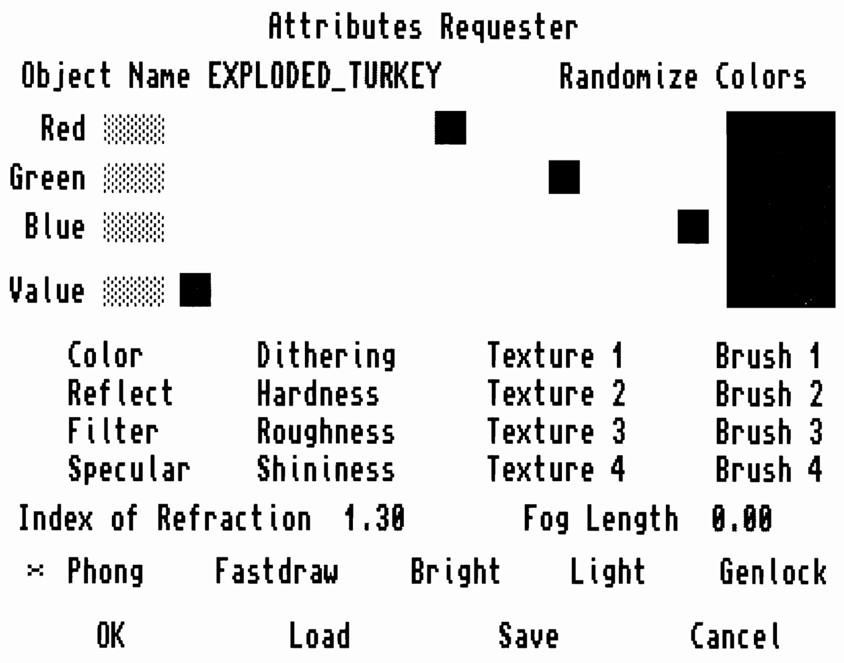
3.4 Object Appearance

The low-level commands to create and manipulate objects are sufficient to create any model you can think of. An additional level of control you have is the ability to define the surface color and attributes of your object. A flat plane might be made of two triangles, but depending on how you set the attributes of the plane, it might render as a pane of glass, a reflective mirror, a wood tabletop, a piece of graph paper, or a picture of your grandmother. Defining the surface attributes of objects gives them their character. Luckily, Imagine gives you excellent control of the appearance of objects though the use of attributes, textures, and brushmaps.

3.4.1 Attributes

Every object has a set of attributes that can be modified. These attributes tell Imagine what appearance the object's surface has; their color, the way light reflects on them, and the transparency. In a group, every object can have different attributes from the parent; when you pick a group and change (for example) its color, you only modify the *parent's* attributes. To change any object's attributes, just pick the object and select "Attributes" from the Object menu. A requester will appear, and you can select the different object properties to change. In addition, you can place brush maps and textures on the object, as well as add or change the object's name.

You can even change these attributes smoothly in time. See page 103.



```
Attributes Requester
Object Name EXPLODED_TURKEY        Randomize Colors
  Red
Green
 Blue

Value

   Color      Dithering    Texture 1    Brush 1
   Reflect    Hardness     Texture 2    Brush 2
   Filter     Roughness    Texture 3    Brush 3
   Specular   Shininess    Texture 4    Brush 4
Index of Refraction  1.30      Fog Length  0.00
 × Phong    Fastdraw    Bright    Light    Genlock
     OK          Load          Save          Cancel
```

Figure 3.1: The requester for setting object attributes

Choosing and setting attributes is immensely important to make your objects look realistic. Using textures and brushmaps give you near-infinite control on what your object's surface looks like. Figure 3.1 shows the requester that allows you to define all of these attribute values.

Surface Attributes

Attributes are defined by either typing a number (or name) into a requester or by sliding a gadget to select a value such as "Red Intensity." When a value is selected by a slider, you can also just type in a number to the gadget to the left of the slider to set it exactly. When you are setting colors, you will see a dithered approximation to the color you are setting; a great help.

There are a great variety of values you can set. Below is a list of what each parameter controls. The best way to find a good value for your object is to make an educated guess, then use "Quickrender" to see the effect it has on your object. Through experimentation, you can get a good feel how to use these parameters.

Quickrender is discussed on page 17.

Object Name Allows you to give a name to your object. This is very useful, especially in the Action editor where actors are not seen directly and you have to rely on their names to distinguish them. Also used to identify spline paths for commands like "Extrude."

Color The most straightforward control. You can use the three sliders to set the primary color of your object. The default object color is all white, the red, green, and blue values all at their maximum of 255.

Reflect Objects that are reflective can be characterized by *how* reflective they are and what colors tend to be reflected more. The Reflect sliders will change the mirror-like reflectivity of your object. A value of 0 means the object is not reflective at all, and 255 is completely mirror reflective. If you make the red, green, and blue components of reflection unequal, the reflected image will be biased towards those colors. For example, reflections in steel have a blue tint, so you might use an RGB reflect value of (60,60,100) for a very shiny piece of metal. *Caution:* high values of reflectivity will make mirror-like objects. Sometimes these objects seem almost to disappear! This is because they are so reflective they just show reflected images and don't show their own surface. To make the object more distinct, lower the reflect value. Reflect is also used in determining "Shininess" described later.

Filter Objects are often partially transparent. A piece of wood is opaque, but water is easy to see through. Objects might even have a tinted transparency, like stained glass. The Filter values of an object define how transparent an object is. When the values are high, the object becomes more transparent. At the limit, Filter RGB= (255, 255, 255), the object will be completely invisible. You can create a tinted transparency by using Filter RGB values forming the tint the object transmits. A piece of red glass might have filter values of (200,100,100). The best way to understand Filter is to think of what color light gets transmitted through: (255, 255, 255) means all light gets through the object, (0,0,0) means no light, and (200,100,100) means some light, tinted red. Filter is also used in Shininess described below, and *if Shininess is used Filter doesn't control transparency, and objects are opaque!*

Well, there are some effects an "invisible" object has due to Imagine's rendering problems. See page 153.

Specular When a light reflects directly off an object, you can see a special type of reflection called a "specular" reflection, like the highlight on a shiny apple. These highlights can be of different colors and intensities. The Specular control allows you to set the color and intensity of the object's specular reflection. Most objects have some specular reflection, with smooth, clean surfaces having more than rough dirty ones. The specular color is usually similar to the object's surface color, but there are exceptions. Objects which have a surface wax on them (like furniture or an apple) reflect the wax's sheen, and have white highlights. Plastics also have white specular highlights, since the light shines on the transparent binder of the plastic and not the suspended pigment particles. Water and glass have very intense white highlights. To set the specular intensity and color, just use the RGB sliders. Also see Hardness which affects specular highlight appearances as well.

Dithering This control is only applicable when you are rendering 12 bit files, something that really isn't recommended. However, if you do use these 12 bit files, you can set the amount of dithering used to display each object. Sometimes smooth objects like metals or glass look better with less dithering, though for most objects you want to keep dithering at 255. This control has no effect when you are rendering 24-bit files, so most people can safely ignore it.

You can read about different rendering file formats on page 126.

Hardness Hardness is another control over the appearance of specular highlights.

Instead of controlling color or intensity of these highlights, Hardness controls how tight the highlights are, or how much they physically spread over the object. A glass crystal ball would have a high hardness (very tight, intense highlights) whereas a beach ball would have large soft highlights and a lower Hardness value.

Roughness Most real world objects aren't made of perfectly smooth materials. Instead, they have small surface features which tend to make a rough appearance. You can make objects look more irregular by adding a non-zero value of Roughness; usually a small number like 10 is all that is needed. *Do not use Roughness in animations!* Because of the quick-and-dirty algorithm Imagine uses to add roughness, a rough object does not keep its appearance constant over subsequent frames; the surface looks like it is seething with insects or static. Try to use Roughness as little as possible.

Maybe you want *an insect covered object, though...*

Shininess Shininess is a lot like Reflect in that it makes the object reflect its environment. Instead of reflecting images perfectly, however, Shininess tends to have a more diffuse appearance. Metals in particular don't have a mirror sheen on them; instead they show a blurry coloration from their environment. By using Shininess, you can make you objects have this diffuse reflective appearance. A value of 0 in the Shininess control will have no shine added to your object, whereas a value of 255 will make your object very shiny. When you use shininess, several other attribute controls are stolen! That is, they don't control what they normally do, but instead provide additional control over the shininess. The values in the Filter control give the shininess its tint; to make a shiny gold foil appearance, you would set Shininess to perhaps 200 and Filter to (200, 200, 100). The "Index of Refraction" requester also enhances the shininess effect. A value of 1.0 makes the shininess very diffuse, but as you raise the number the blurry reflections become more focused and distinct. A value of 3.0 won't give the exact same appearance as Reflect, but the soft blur of shiny color on the object becomes much more focused.

Index of Refraction The index of refraction helps control shininess, but its main use is to tell Imagine how much to bend light when it passes though a transparent object. This bending is the effect you see in lenses or even glasses of water. A higher index of refraction causes light to bend more as it passes though the object, while a lower value makes less of an effect. A value of 1.0 (*not* 0.0) is the lowest index of refraction and light passes straight through these objects without distortion. Table 3.1 shows the indexes of many common materials.

This is only effective in raytracing.

In addition to controlling the direct appearance of your object, there are many small flags you can activate to help define information about your object.

Fastdraw (Quickdraw)

The "Fastdraw" button is really misnamed; it should be "Quickdraw" to be consistent with other Imagine commands. Quickdraw is an attribute that only affects objects when they are displayed in the wireframe views in Imagine. Instead of showing your

Material	Index
Air	1.0
Cold Air (0 Degrees C)	1.00001
Water	1.33
Alcohol	1.36
Quartz	1.46
Salt	1.5
Glass	1.52
Amber	1.55
Flint Glass	1.66
Diamond	2.4

Table 3.1: Index of refraction for common materials

object in complete wireframe, when you select "Fastdraw" in an object's attributes requester, the object will be displayed as a large bounding box in the three main display views. This box is very similar to the bounding box used to represent objects that are being interactively moved, scaled, and rotated; it has the same dimensions as your object, and you can quickly tell the scale, position, and orientation of your object. Because the box is faster to draw than a complex object with perhaps *hundreds* of edges, Quickdraw is a very useful option when you start making very complex models.

You can actually make objects use the Quickdraw display method without even going into the Attributes requester. If you pick an object and select "Quickdraw Pick" option from the Functions menu, your object will be shown in the Quickdraw method. You can even make *all* of the objects in a scene become Quickdrawed by selecting the "Quickdraw All" option. Similarly, "Quickdraw None" will turn off Quickdraw for all objects.

Quickdrawed? Is that a real verb?

Fog

Fog is a unique and powerful attribute. More than any other attribute option, it can completely change the character of objects. Fog will tell Imagine to render your object not as a solid (or even transparent) set of faces, but as a *volume* of opaque gas that absorbs light that passes through. If you make a solid sphere and render it with its fog attribute turned on, it will not have distinct edges, instead it will appear more like a blur of color hanging in the air.

Global fog is closely related to object fog; see page 114.

The fog appearance is produced by absorbing light from rays that pass through the object. If a ray only passes through a very small volume, it isn't affected very much, but if the ray passes through a larger volume, it has a longer travel length *in* the fog and is absorbed more. This relation between travel length and absorption is the difference between fog and simple transparency; transparent objects have the same translucency no matter how "thick" they are.

The algorithm that Imagine uses to compute the amount light is absorbed is completely described in Section 7.8.3 on page 114. The basic algorithm mixes a simple color into the color a light ray "should" be according to how long the ray was inside the fog volume. If the ray wasn't in the fog for a long distance, the ray's color isn't affected much. If it stayed in the volume for a long time, it is completely replaced with the fog color.

There are two controls over object fog; the first is the gadget labeled "Fog Length." This controls the density of the fog, with higher numbers making the fog *less* dense. This number is actually a distance: it can be thought of as the characteristic distance where the fog absorption becomes noticeable. You might want to set this value to roughly the length of the object. After a test render, to make the fog less dense, increase the fog length. To make the fog denser, decrease the number. If the "Fog Length" is set to zero (the default) the object is *not* made of fog and is rendered as usual.

The color of the fog is determined by the standard object color attribute. This is the shade that will be mixed into the light rays that pass through the object's volume. For a white smoke, you would use a white or grey color. The other attributes (like Reflect) are not really applicable to fog, and have little if any affect on the fog's appearance. However, textures and brushmaps that change the object's color *do* affect the fog; you can map a grinning devil's face onto a plume of smoke and animate it billowing for a spooky effect.

One nice thing about fog is that it is easy to make objects for it. Since the fog is very tenuous, small details in the object's shape aren't visible. A big blob produced in the Forms editor will make a *beautiful* cloud when fog is applied. This takes wonderfully little skill, just make a blob in Forms and pull at it until you get something resembling, well, a smooth blob.

Actually, Fog is a powerful tool, and isn't limited to smoke or clouds. The visible light beams that you might associate with lighthouses or headlights on stormy nights on the highway are easily made with fog volumes. Just make a cone or cylinder with end caps, and set the object's color to a nice white-yellow and turn on fog. It's easy!

Using fog to make visible light beams is discussed on page 154.

Phong and Object Shading

When an object is rendered, sometimes the fact that it is made of a bunch of triangles is painfully obvious. In order to try to render surfaces as smooth curves instead of faceted triangles, Imagine allows you to use a technique called "Phong shading," a common graphics trick. Phong shading tells Imagine's renderer to reflect light off a surface as if the surface *was* a curve. It does this by assuming the surface normal, the direction the surface points, is a smoothly changing value, when in fact it stays constant then suddenly changes at the triangle boundaries.

This is very much related to altitude mapping.

Although Phong shading can help considerably in making surfaces look smooth, sometimes Phong shading has problems. A profile of your object (or a shadow) will show the faceted faces around the edge of the object. Also, sometimes faces should *not* be smoothed, like a 90 degree corner which is *supposed* to be sharp.

You can turn Phong shading on for an object by clicking on the "Phong" box in the attribute requester.

You can actually tell Imagine to make certain parts of your object smoothly Phong shaded, and other parts "sharp," or faceted. You might have a object in the shape of the letter 'S.' You want the curvy sides of the letter to be Phong shaded, so they will appear smooth and continuous, but you *don't* want to Phong shade the caps of the curve that makes up the S; you want to keep the sharp right angle bend that appears there.

In Imagine 2.0, you can tell Imagine which edges should be "soft" or Phong shaded, and which edges should be "hard" or faceted. In the case of the 'S,' you would want all of the edges to be soft except for the four edges at the right angle bends of the end of the S.

You can tell Imagine to make sets of edges soft or sharp by using the "Make Soft" and "Make Sharp" commands in the Functions menu. When you are in "Pick Edges" mode, you can pick a set of edges and use the appropriate command to tell Imagine that the picked edges should be Phong shaded or not.

When you want to find out what edges have already been Phong shaded or not, the command "Pick Sharp" will automatically pick all of the non-Phonged edges when you are in "Pick Edges" mode. From here, you can see which edges are still unsmoothed, which is useful to check to see you've specified your Phong shaded edges properly.

There are a couple of other commands that also affect the way your object is shaded. The "Bright" button will cause your object to be shaded with a flat color instead of the normal light shading algorithms. Normally an object's exact surface color depends on the intensity of the light that falls on it, whether the surface is pointed towards or away from the light, and how far away lights are from the object. When you use "Bright," the object's surface is colored with the exact shade specified by the color attribute, with no shadows or light shading at all.

You can use "Bright" to make cartoon like scenes. See page 146.

This can be useful when you are using a brushmap mapped onto a flat plane to make a mini-backdrop such as a tree. (You just use a 2D image of a tree!) You don't want shadows to fall on the image, so you use "Bright." Also, for something like the windows in an office building, you can use "Bright" to make them solid colors even when your scene is darkly lit.

One last control also allows you to change your object's color completely. The "Genlock" button will render your object completely in the genlock color specified in Preferences (usually 0,0,0). This means you can cut a "hole" in your image for live video to show through that is the shape of a rendered object. This is really equivalent to rendering the object using "Bright" and a surface color of (0,0,0) except that in trace mode other objects will reflect the true object colors instead of the black silhouette.

Built in Lights

Objects can emit light. This is very handy when the object is something like a lamp or streetlight and the lightsource is really a renderable object. Also, the lights will move with the object, so a car's headlights will stay with a car as it travels down a road.

When the "Light" option is chosen in the Attribute requester, the object will emit a light from the center of the object's axis. *The surface of the object does* not *glow or produce light!* The light that is produced is just like the standard lights in the Action editor, a pointlike emitter.

Independent lights in Action is discussed on page 105.

When you select the option to make the object emit light, you are presented with a requester that is *identical* to the one for a standard light in the Action editor. You have options to change the color and intensity of the light, whether the light diminishes its intensity with distance, whether it casts shadows, and if the light is a spotlight or spherical source. All of these options are discussed starting on page 106.

When an object emits light as a cylindrical or conical source, the direction of the light emission is along the object's Y axis. This is somewhat annoying if you don't like your axis to be oriented that way, so I often make light-emitting objects by grouping a lone axis to the "real" object and having the axis emit the light. For example, there is a desk lamp I built (which is on the disk with this book!) where I have the lightsource grouped to the bulb that is supposedly emitting the light.

When you have a cylindrical or conical light source, the size of the axis is very important; it controls the distance that the light travels as well as the width of the beam. The Y axis size controls the distance the light travels, and the X size controls the spread. This is shown in Figure 7.6 on page 111. This means that an axis is often absurdly large; the lamp I mentioned is an example. Since you often want the light it emits to extend a long distance (like the whole scene!) the axis has to be the size of the whole world. This is annoying, but it does work.

Sorry for making you keep flipping to Action...

Faces and Subgroups

Sometimes you might want to add attributes to just part of an object. This is often done with textures or brushmaps, but sometimes you just want to color a few individual faces of your object.

If you enter "Pick Faces" mode and pick one or more faces, you can select the "Attributes" command and set the Color, Reflect, and Filter values for just those faces. You can't change the other attributes like index of refraction or shininess, but the main three attributes are usually enough to add some details to your object. If you need more precise control (like making part of the object a different index of refraction) you can just add another object (with the different attributes) grouped to the original object. "Split" and "Slice" are often useful for this.

A new option in Imagine 2.0 allows you to randomly color the object's faces. The "Randomize Colors" button near the top of the attributes requester will give each face a random color. This isn't of that much practical use unless you were trying to make a psychodelic teapot or something.

One problem with coloring individual faces is that it is often difficult to go back and modify the exact same set faces that you colored previously. The difficulty lies in just *finding* the faces, since there is no indication what color a face is unless you render the object. A new option in Imagine 2.0 addresses this problem by allowing you to make something called a "subgroup." A subgroup is a subset of the faces in an object. A face can be in as many subgroups as you like, and an object can have as many subgroups as you can think of. You might have a helicopter object and assign all of the faces in the rotor to the "Rotor" subgroup, and the faces in the jet turbine to the "Engine" subgroup. You might also have a third subgroup called "Top" that includes faces in the rotor, the engine, the rotor housing, and the exhaust.

A subgroup is useful because you can call them up by name. When you are in "Pick Faces" mode, you can tell Imagine to pick all of the faces in the "Rotor" subgroup. Once they are picked, you can assign the faces a particular attribute. The advantage is that later, when you want to change the rotor color again, you can just pick the same "Rotor" faces, and you'll be able to change the attributes of the *same group of faces*. When you want to use individual face coloring on an object, subgroups are invaluable.

Subgroups can also be useful when applying textures and brushmaps. You can restrict the regions that textures and brushmaps are applied to by specifying a certain subgroup (by name) to keep the effect limited to. You could therefore apply the Camo texture to just the rotor of the helicopter if you wanted to keep the rest of the body unchanged.

Making subgroups is easy. In "Pick Faces" mode, just pick the faces you want to define the subgroup and select the command "Make Subgroup" in the Functions menu. Imagine will ask you for a subgroup name. Just type in the name (like "Rotor") and that set of faces will be named as a subgroup.

To pick a subgroup, just enter "Pick Faces" mode and use the command "Pick Subgroup" in the Functions menu. Imagine will ask you for the subgroup name (like "Rotor") and will then pick all of the faces in the subgroup. You can then set the attributes of the faces or do whatever you like. You can also unpick faces, which might be useful if you want to pick all of the faces in an object *except* those in a subgroup. The command "Unpick Subgroup" in the Functions menu will do this for you.

If you want to get rid of a subgroup (not delete the faces, just tell Imagine to forget about their association) the command "Unmake Subgroup" will remove the subgroup from an object.

Another way of picking faces is with the "Pick Range" command. This is a handy way of choosing a range of faces by specifying them by the internal number of each face. This command will ask you for the beginning face number, the end face number, and the step between faces. Most of the time you don't know what faces are assigned to what numbers, but for something like a primitive sphere, the faces are numbered in an orderly way. You might use this command to pick every fifth face in the sphere, then delete them. This would remove one fifth of faces on the sphere, but the faces will be removed in an orderly pattern so you end up with stellated holes like a whiffleball. You can also use the command "Unpick Range" to unpick faces in a similar manner. The command "Sort" is sometimes useful with these two commands, since you can specify your own face ordering.

Sort is discussed on page 26.

Attribute Files

Luckily attributes don't have to be manually set for every object. You can copy the attributes of another object by using the "Load" gadget at the bottom of the attribute requester. When you specify the filename of another object, the attributes of that object are read in and applied to the current object. This will even load in textures and their settings and brushmaps.

You can also save attribute settings by using the "Save" gadget at the bottom of the attribute requester. This will actually save an Imagine object, a simple lone, empty axis, that has the same attributes, textures, and brushmaps as your current object. You can then later "Load" these axes to copy the attributes associated with them to the current object.

It is a good idea to make a little library of these saved attributes. The files are tiny (they are just a simple axis) and it is very handy to be able to simply load a file called *Chrome* and use that as a basis for your object's attributes.

3.4.2 Textures

No matter how good you are at creating objects, their appearance will never be detailed enough to look perfect. In the real world, most objects are rich with small details that you are used to seeing, like dirt on a wall or small bumps on an orange. Modeling these subtle effects with extra faces would be difficult or impossible, but Imagine lets you define surface appearance in a very complex way with a feature called a "texture."

Textures are algorithms that Imagine can use to help color an object. These textures do *not* color individual faces, but actually color the individual pixels that form an object during the render. If you have a "wood" texture on a flat plane made of just two faces, you will still see the rich detail of the woodgrain, and not just two differently colored triangles. Textures don't take much RAM at all to render, though they do slow rendering time down. Overall, textures are an excellent way to add detail to your picture; a woodgrained picture frame has a lot more character than a flat brown one.

Textures can also change the reflectivity and transparency (filter) values of your object. For most textures, when I say "color" I mean surface color, reflection, and transparency. Most textures allow you to set all three! Some textures, like disturb, will affect surface light reflection like altitude brush maps do. Textures are very versatile, and can even be layered on top of each other for more complex effects.

Altitude mapping is described on page 39.

Textures are just algorithms that Imagine uses to decide how to color a particular point on the surface of an object. When a ray hits an object, Imagine uses the location of that surface to determine how to color it. For example, a very primitive texture might color the object white if the location where the ray hit is above a certain line, and blue if it is below the line. This would make half of an object white, and half blue, divided at that transition line. A texture is just an algorithm, the set of rules (like the white-above/blue-below example) for determining the color of an object at every point in space.

Textures are added to individual objects by clicking on one of the four "texture" boxes in an object's attribute requester. If a texture is already being applied, one or more of these boxes will have an "X" in them which means you can click on that box to edit the texture's parameters. You can apply up to four textures to an object at once, though most of the time you just use none or one.

To apply a new texture, you click on an empty texture box. Imagine will have you choose a filename of a texture (probably found in the *Textures* drawer in your Imagine directory) that you want to apply to that object. After you select a texture, a large requester will appear which allows you to set the texture's parameters. Each texture has its own particular inputs. Most of the time, you need to enter a color, transparency, and reflection value for whatever effect the texture is applying to your object. These are raw numbers; no sliders. When you are setting RGB triples, the approximate color that the three numbers represent is shown in small box in the requester.

There are often a few extra parameters to set dealing with the way the texture is applied. This might be check spacing, wood grain thickness, or brick size. What these values control vary from texture to texture, so read the individual descriptions in the Appendix.

All of the different textures and their individual parameters are listed in the "Textures" appendix on page 171.

Some texture parameters control certain lengths or distances that are measured in absolute Imagine Units. For example you can set the size of a check in the Checks texture by simply entering the width of the check. If you enter a check size of 25 and your object is 100 units across, you'll get four checks across the width of your object. What happens if you scale your object, though, and make it larger? In Imagine 0.9–1.1 the dimensions of the checks remains constant, so if you scale your object up to 200 units, you would get 8 checks across. *In Imagine 2.0 the texture parameters that control sizes measured in actual Imagine Units will scale at the same time you scale your object.* In the example where the plane is scaled to a size of 200, the Checks texture would automagically change the check size parameter in its Check texture to 50 to keep the same four checks across the width of the object.

This parameter scaling is performed in order to keep the appearance of objects constant when you scale an object. In the early versions of Imagine everything was tied to absolute units, and scaling an object (especially in the Stage editor when laying out a scene) meant having to readjust the texture parameters. This new feature will keep texture appearances constant. It will adjust only those parameters which control a width measured in units. If you scale just the axis of the object, the texture parameters are *not* scaled, only a global object scaling will affect your texture parameters.

Be careful if you need an absolute size in a texture, such as if you want to use the Grid texture to make lines exactly 100 units apart when the scene is rendered. You have to be careful not to scale the object after you've placed the texture, or the absolute texture measurements will be scaled as well. However, this is pretty easy to handle since it just means you have to be aware that you shouldn't scale the object.

The only other parameter that controls textures is the texture axis, which can be manually edited. The texture axis is pretty important, since for most textures, you need a "base" location and orientation to give the details the texture applies a reference location to work from. For example, the Linear texture needs to know where the "fade" starts and what direction to fade in. What you do is just place the texture axis where you want the fade, and point the Z axis in the direction you want it to go. The wood texture at its simplest is a bunch of concentric cylinders of color. The texture axis tells Imagine where the centerline should be and which direction the cylinder points. Some textures really don't use the axis (like Camo) but when a position or orientation is defined it is done with the texture axis.

To move the texture axis, just click on the "Edit Texture Axis" gadget at the bottom of the requester and use the standard move, scale, and rotate commands to change the axis. You can numerically manipulate the texture axis with the "Transform axis" gadget.

Page 10 describes the standard manipulation commands and 13 describes Transform.

An important point; if your texture axis is *right* on a face, you might get some funky effects, since for a texture like checks, the surface of the object is *exactly* where the checks change. The algorithm does not know what color to return, so you get what Impulse calls a "digital bounce." This is very common when you're texture mapping a flat plane, since the texture axis is often positioned right on the plane. You can fix this problem very easily by moving the texture axis just a tiny bit so that it is off of the object's surface. Imagine 2.0 had default texture axis placement that tries to minimize this, but you have to be careful if you are adjusting the axis manually.

One last option in the large texture requester allows you to remove the texture from the object completely. Just click on the "Drop" gadget and the texture will be deleted.

Many, if not all, textures only affect some parts of an object. The camouflage texture is an excellent example. You set the default color of the object from the attributes requester. The Camo texture then layers its spots *on top* of the plain default color of the object. If there is no spot on a particular location, the default color will show through. This is true with most textures. Wood only adds the "grain" and lets the object's default color become the normal woody non-grain parts. Linear gradually fades from the default to another color. Checks adds color on its checks and lets the default attributes stay in the opposing checks.

Why is this important? Well, this can be used to great advantage. *You can add up to four textures simultaneously.* They are added in order from 1 to 4. What can you do with this? Well, you can take a desk, and with Texture 1 add a wood texture. Then you can add a Camo texture as #2, and the spots will cover up the wood, but you'll see grained wood where there are no spots and solid color spots where there's a Camo spot. Each texture layers a new color onto an object, leaving some of the original surface showing though. Since you can add up to four textures, you can have four layers.

Brushmaps layer as well, and go on after any textures. Brushmaps are described on page 39.

For example, an island object might be given a sandy beach and green vegetation by using textures in combination. First, a Radial texture which varies the base color from different shades of sandy-brown. Second, the wood texture with a very fine grain and a dark brown color makes a sandy beach with enough variation to keep it from being too boring. (Camo with very small spots might work, too.) Then, a Linear texture could fade the beach into a nice vegetation green color once you get past a certain distance from the beach. These textures would give the island a nice, detailed character that you would never be able to match by picking and coloring individual polygons.

You have some extra control over the areas of an object a texture is restricted to. You can specify a subgroup of faces that the texture will affect, leaving the rest of the object untouched. If you don't specify a subgroup (the default), the entire object has the texture applied to it. You can also apply the texture to the children of an object, to give an entire group a single texture. This is done by selecting the "Apply to Child Objects" button.

Subgroups are discussed on page 35.

An interesting ability of textures is that they can morph their parameter values as well as the position and orientation of the texture axes. This means that you can have an object's texture move and spin across the object in time, or make the checks applied to it grow or the color change in time. Morphing textures is described in the

Action chapter on page 103.

3.4.3 Brushmaps

Textures are not the only way to add detail to an object. A more direct, less elegant, but more versatile method is to use brush maps. Brush maps are ways of taking standard Amiga pictures (sometimes called IFFs, though pictures are a subset of the Interchange File Format) that can be placed "by hand" on your object. In its easiest incarnation, you could brush map a picture of your face onto a flat plane, put a frame (using wood texture!) around it, and you have a virtual art piece. In its most complex incarnation, you could take a set of 40 256-level intensity maps saved as IFF24s and tile them endlessly on a plane during a 40 frame anim, and have the map pixel intensity create reflections and highlights from the flat plane just like it was really an animated, wind-wave covered ocean.

Whew, that is indeed a complex example! It just shows that brushmaps are very versatile!

Brush maps can give objects the same four characteristics that textures can: surface color, reflection, transparency, and surface altitude. Going back to the "my face in a frame" example, a color map is straightforward. The brushmap image is simply used to determine the color of the object at the location the brush is applied. A reflection map will make the object reflect the color and intensity of light corresponding to the map; in other words, a black map would make the picture in the frame reflect no light, a white map would make the picture a mirror, and a yellow map would make only yellow light reflect.[1] Transparency (really Filter) is similar. Black is opaque, pure white is the clearest crystal, and yellow would let yellow light though. An example given by Rick Rodriguez in his manual is that a filter map of a color image of your face applied to a plane would object appear like a stained-glass window. You could have fun with that!

The last brush type is altitude, which is a bit more interesting. An altitude map just tells Imagine that light hitting the object's surface should be reflected, refracted, and specularated (!) as if it hit a surface that had a certain shape to it; this shape is defined by a "height" measured by the brushmap's intensity. If you mapped a picture with lots of small fuzzy grey dots onto a sphere, you would get reflections and light highlights as if the sphere had tiny pits in it like an orange. *The altitude map does* not *change the real surface height of your object at all.* It just makes the light bounce off of the surface as *if* it had these bumps on it. This is the difference between a displacement map and an altitude map.

It is very important that altitude maps have soft color changes. A sudden color change implies a sudden change in the altitude the brush represents, and the altitude mapping algorithm can't do much with too narrow of a transition. Use many shades of grey, and certainly avoid a direct transition from pure black to pure white. You can soften images by using "Smooth" in Deluxe Paint, or "Blur" in ADPro. If you output an 8 or 16 color image from ADPro for use as an altitude map, do not use any dithering, since the alternating pixels will add a rough appearance to what should be a smooth gradient.

[1]The yellow example is not strictly true... the yellow-mirror would reflect green and red light, which combine to make yellow.

One option you should use if you're using a transparency map is the "Full Scale Value" in the brush requester. This allows you to tell Imagine how transparent a pure white image should be. Normally a white intensity value of 255 is perfectly transparent. *When you use a standard 1–4 bitplane image, Imagine interprets full white as only 240.* This is due to the fact that the Amiga's standard palette has only 16 values for each color, so the maximum intensity is 15. When it is "extended" to a range of 0–255, Imagine multiplies by 16 to get 240. This is very annoying, since it implies that you cannot make perfect transparency with these brushmaps. Luckily, you can set the "Full Scale Value" to make this value of 240 the maximum range value for the brushmap. In other words, 240 now becomes perfectly transparent, which is the effect you're probably looking for. As a side benefit, you can also use this to brighten images (it works for colormaps!). If you lower the "Full Scale Value" low enough, you can get some psychodelic effects.

Any standard Amiga IFF can be used as a brushmap. This includes all pictures saved from Deluxe Paint, the most common picture editing program on the Amiga. A format of high color resolutions known as IFF24 saves three bytes of color information per pixel and produces beautiful color reproduction. These IFF24s can be produced by DCTV, ToasterPaint, Digi-view RGB saves (only 21 bits, but it is saved as 24 for compatibility), The Art Department, and many other graphics programs.

The 8 bitplane black and white IFF images that can be saved from a program like ADPro are just about the only exception.

Whatever type of brush you use, remember that Imagine can't do magic to your pictures. A 16 color picture of yourself is going to look positively cheesy compared to a 24-bit picture. For some applications, though, you don't need much more information. Applying a 4-color logo to an object won't benefit by using a 16 million color brush! When the image only has 4 colors, using a 2 bitplane brush is the most logical format. Imagine can handle any size image, including ones that are larger than the screen. Higher resolution is obviously higher quality, though it's pointless to use a 1000 by 1000 picture of a logo; even in a close-up shot, it would be indistinguishable from a 640 by 400 picture. However, using too low a resolution will be painfully obvious when you render since you will be able to see the individual pixels that form the brushmap image. Depending on how close of a view of the brushmap you show when you render the image, you usually don't need more than 320 by 200. This is very dependent on how close your camera view is to the object with the brushmap on it! I've used 100 by 100 brushes with great results.

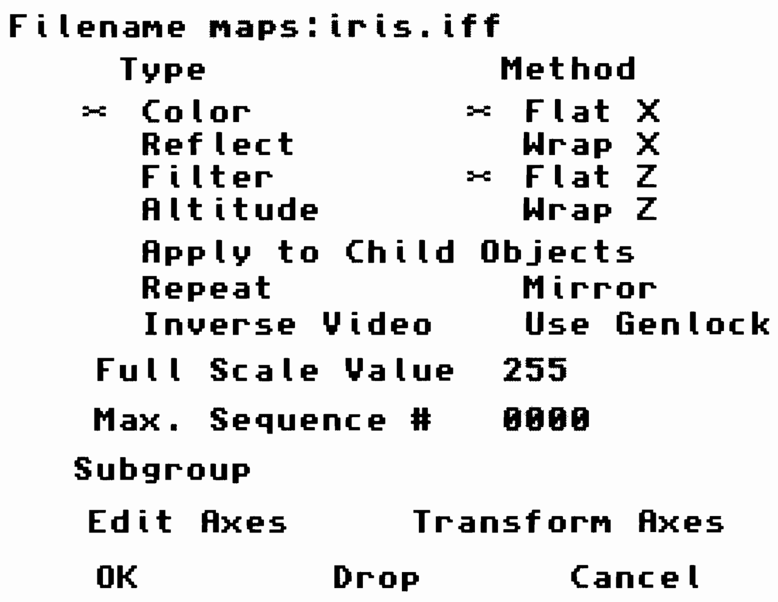
The Art Department is discussed on page 189.

A fun trick: Imagine can easily output *IFF24s, so you can use a previous rendering as a brush map.*

Once you have made or found your brushmap you have to place it on your object. The placement determines the size and orientation of the image, as well as what region of the object is affected, and in the case of altitude maps, how much the surface is apparently distorted.

Brushmaps are placed by manipulating a separate brushmap axis, much like the texture axis is positioned to provide a reference point for textures applied to an object. The brushmap axis can be moved, rotated, and scaled to position brushmaps on your object at any location you want. The exact positioning depends on the type of brush mapping that you are applying; this is described in the next subsections.

Imagine 2.0 has a wonderful new feature that automatically positions brushmap axes in a logical position so that the map will be applied to the entire surface of your object. This is a great help, since remembering all of the brush map positioning rules can be confusing some times. Even if you have to modify the starting position, the



Filename maps:iris.iff

Type	Method
[✓ Color]	[✓ Flat X]
Reflect	Wrap X
Filter	[✓ Flat Z]
Altitude	Wrap Z

Apply to Child Objects

Repeat Mirror

Inverse Video Use Genlock

Full Scale Value 255

Max. Sequence # 0000

Subgroup

Edit Axes Transform Axes

OK Drop Cancel

Figure 3.2: The requester for applying brushmaps

brush is at least set up in a reasonable orientation, size and position to begin with, so modifying it is easier than setting the axis configuration from scratch.

Applying Brushmaps

Brushmaps are applied from the Attributes requester. Clicking on one of the boxes labeled "Brushmap" will bring up a file requester asking for the filename of the brushmap you want to apply to your object. After you have selected a filename, a requester like the one shown in Figure 3.2 will appear. This requester allows you to control the way the brushmap is applied to your object.

At the top of the requester, the filename of the picture is given, allowing you to change or edit it later. The four types of brushmap effects (Color, Reflect, Filter, and Altitude) are listed on the left side of the requester. The default choice is "Color" but you can just click on the proper gadget to change it.

The most important option is the type of mapping that is used to apply the image to your object. The different types are controlled by the gadgets on the right labeled "Flat X," "Wrap X," "Flat Z," and "Wrap Z."

These description of the spherical and cylindrical mapping are simple analogies, and are almost painfully vague. Appendix F on page 185 has the lowdown on the real *way brushmaps are applied to objects.*

There are three basic types of brush wrapping: a "flat" wrap (Flat X Flat Z), a "spherical" wrap (Wrap X Wrap Z), and a "cylinder" wrap (Flat X, Wrap Z and Wrap X, Flat Z). These different options control how Imagine associates a position on the object with a certain pixel on the brushmap. Each type of mapping is described in the following subsections, but a brief description: Flat X Flat Z mapping will ignore any surface bumps and features and just apply itself directly, much like a slide projector would project onto a bumpy screen. A spherical wrap tries to encase the object in the brush, then shrinkwrap the map onto all of the surface features of the

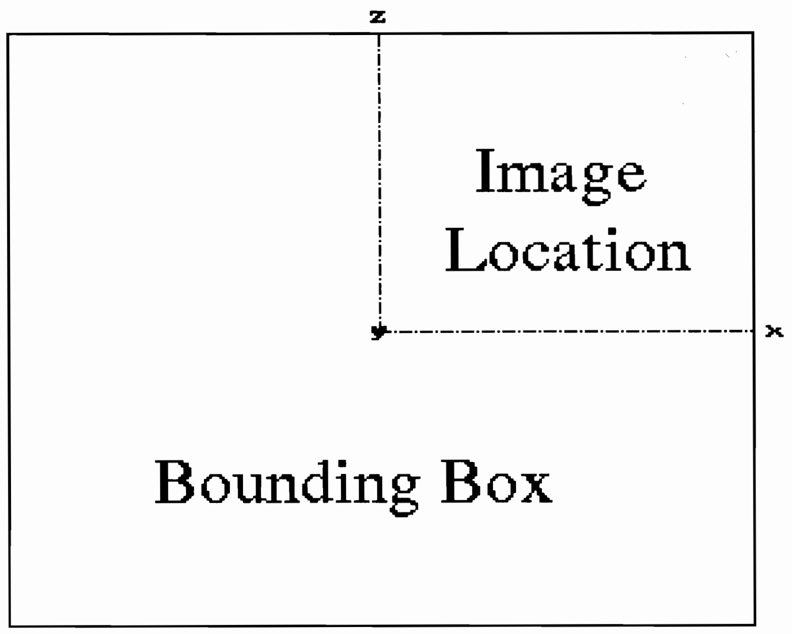


Figure 3.3: Region of the brushmap axis the image is mapped to

object. The cylinder wrap tries to follow contours in one direction, but ignore them in another. Think of taking a piece of gift wrap, and bending it around so its a hollow cylinder. Then place the object in the center of this vertical gift wrap cylinder and push *in* (but not up or down!) to follow the object contours.

Flat X Flat Z Wraps

Flat wraps are the most common and certainly the most controllable of the brush map application methods. Think of having a decal or poster that you want to stick onto a wall. Flat wrapping will perform this simple "flat" projection beautifully. A good example is trying to put a logo onto the side of a truck; an excellent example of where brush maps shine.

You should obviously have your logo picture and truck designed before you try to apply the brush. To place the brush onto an object, you should probably use the "Edit Axes" mode. This lets you move the axes with the same mouse and keyboard commands that you normally use for objects: 'm' for move, 's' for scale, 'r' for rotate, and 'x,' 'y,' and 'z' to toggle restricted directions. Of course, you can also use the gadgets at the bottom of the screen to control the axis manipulation.

The axis you are editing has a big yellow bounding box that is very deceiving. *The area where the brush is actually mapped is the upper right quadrant of this box!* The brush image is placed with its lower left corner right at the center of the axes, and its upper right corner at a point defined by the X and Z axes of the brush map axis. Figure 3.3 shows the way the brushmap axis determines the brush location.

You can also use the "Transformation" gadget to perform numerical manipulation of the brushmap axis.

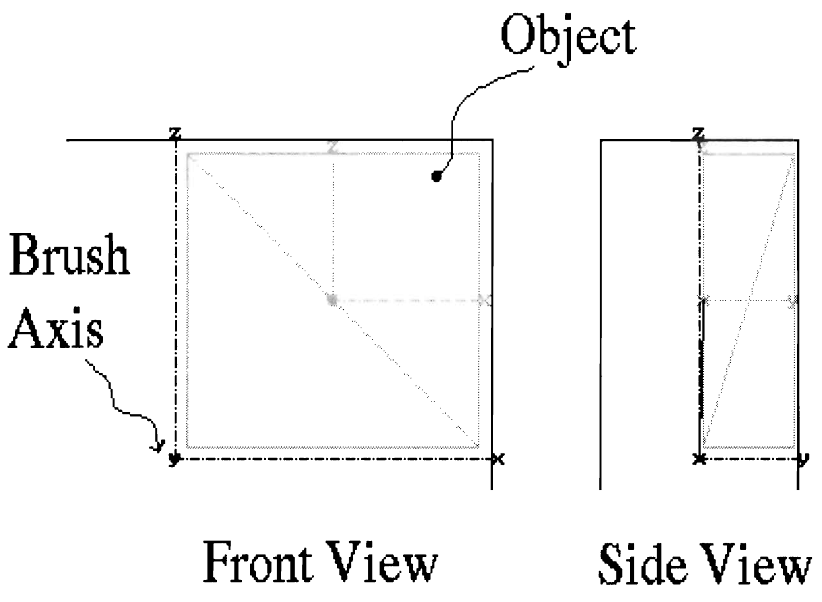


Figure 3.4: Position of the brushmap axes for a Flat Flat map

You want to position the brushmap axis so that the upper-right quadrant lies exactly where you want your brush to be positioned. If you want your brush to cover the entire side of the truck, you'd probably want to make the brush a few extra pixels high and wide so that you don't accidentally get a border around the edge of your logo. You can scale the X and Z sizes of the brush to stretch and squash the brush to whatever shape and size you want.

The Y axis of the brush map is pretty important as well. It tells Imagine how *deep* to apply your brush. Any part of the object that falls between the axis origin and the tip of the Y axis will be colored (or reflected, altituded, or whatever). If the surface is not within the region between the axis center and the end of the Y axis, the surface will *not* have the brush applied to it. For the truck, you'd want to move and scale the brush axis in the Y direction so that the Y axis line *intersects* one side of the truck but *not* the other. The intersected side of the truck would be within the influence of the brush map, whereas the other side of the truck would be left alone. If you scaled the Y axis to include both truck sides, the other side of the truck would get the brush map applied to it as well.[2] Figure 3.4 shows a Flat Flat map being applied to both sides of a rectangular solid. Every part of the object is covered by the brushmap.

Altituded? What a great word!

Be careful when you are applying a Flat Flat brushmap to an infinite ground object. You want to have the Y axis of the brushmap pointed *down* through the ground.

Flat X Flat Z brush mapping is very accurate since it is easy to position the axis

[2]In fact, you'd see a mirror image of the brush on the other side, since you'd be looking at it in the other direction.

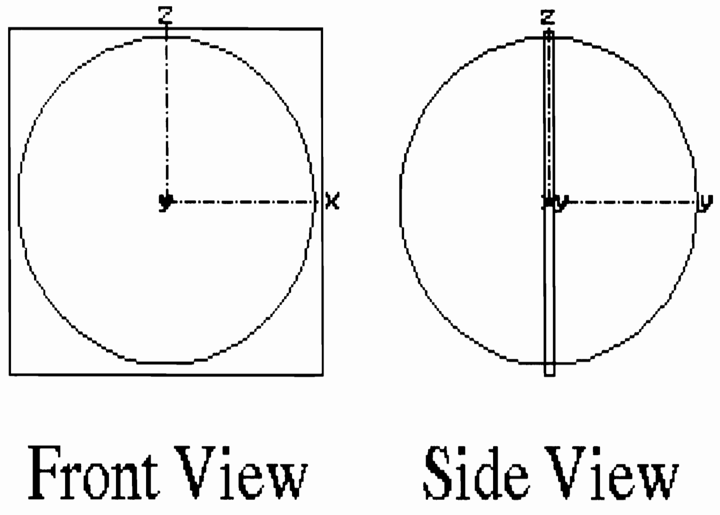


Figure 3.5: Brushmap axis positioning for Wrap X Wrap Z map

and the effect you get is very predictable. It is the "Wrap" maps that cause all of the problems.

Wrap X Wrap Z (Spherical) Maps

This is an accurate mapping, so you can make planets from maps. You want a map with equally spaced coordinates, not *a Mercator Projection map which stretches the top and bottom.*

These are the most complex wraps. I usually think of it as a globe projection; If you have a flat, square brushmap of a regular grid, when mapped onto an object with Wrap X Wrap Z, the lines of the grid will become the latitude and longitude lines on a globe.

The brushmap axis position determines the "source" of the projection that maps the image onto the object's surface. It should be positioned at the center of your object. If you position it towards one side, the image mapped to the surface will be compressed at the region closer to the axis and expanded at the areas farther away. The left and right sides of your image will circle around the axis and meet, but the top and bottom will not. (Again, like a world map.)

The Y axis should be small (Impulse recommends setting it to 1.0), and the X and Z axes are usually scaled to be slightly larger than the object you are applying the map to.

Figure 3.5 shows two views of the positioning of the brushmap axis on a spherical object. Figure F.1 on page 187 shows a picture of a basketball made with a Wrap X Wrap Z brushmap.

Wrap Flat (Cylindrical) Maps

Cylindrical maps are somewhat of a cross between the other two types of mapping. With a Flat X Wrap Z mapping, the Z height of the axis controls how high the image

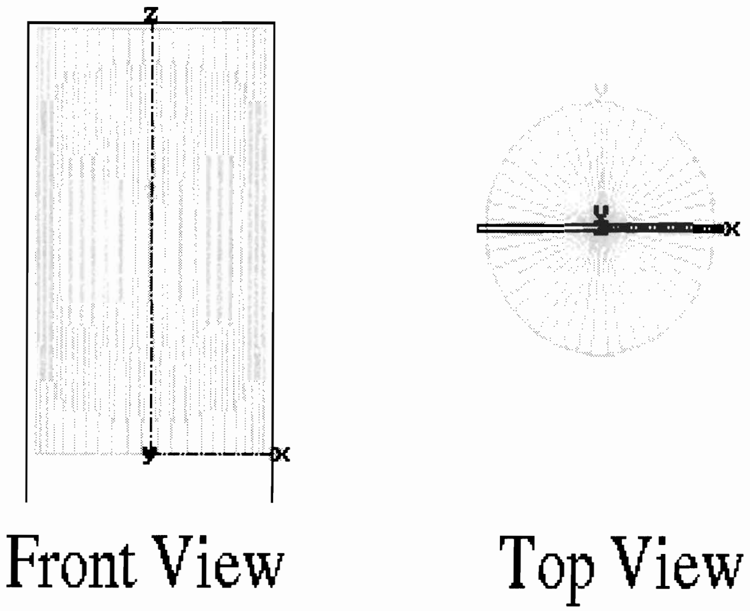


Figure 3.6: Brushmap axis positioning for Flat X Wrap Z map

is and where it is positioned. The image is then mapped onto your object like a tube around this Z axis. Your image will circle around and join together the left and right sides. A Wrap X Flat Z mapping will use the X axis to rotate around and your image will join the top and bottom edges of your image together.

Figure 3.6 shows two views of the axis positioning for a Flat X Wrap Z mapping. The Wrap X Flat Z positioning is the same except the positions of the X and Z are reversed.

Altitude Mapping

When you are applying an altitude map (as opposed to Color, Reflect or Filter maps) the size of the Y axis controls the apparent height of the displacement. The Y axis depth is used to measure how much indentation a full scale range of intensity (0–255) should simulate. For cylinder and sphere wraps, just scale the Y axis. For orange pits, the axis might be 1% of the sphere's size. For a deeply eroded planet, you might use 10%. More than this would probably make really stupid looking reflections.

For a cylindrical or spherical map, using the Y axis to control this parameter is no problem, since the Y axis isn't used for anything else.

A Flat Flat map uses the Y axis for determining the "depth" of the brush. If you are applying the brush to a flat plane, there usually isn't a problem. If you have a complex object that requires you to increase the Y axis to encompass more of the object, you *do* have a problem! What if you don't want to make the apparent altitude as high as the Y axis length? You can't shrink the Y axis size without changing the region that the brush is mapped to.

The answer: basically you're out of luck. Using the Y axis to control several functions is a major Imagine design flaw. Actually, there is something you can do; if you want to reduce the apparent altitude, you can keep the Y axis as large as it has to be, but soften your brushmap. Instead of using a full range of colors (0–255), you can just use part of the range, like (0–128) which will halve the effect of the large Y axis.

Extra Brushmap Options

You have some extra options to help control the way the brushmaps are applied to an object. The "Inverse Video" button will use a reversed (color inverted) version of your brushmap. You can use the "Apply to Child Objects" control to make the brushmap affect not only the parent of a group, but its children as well.

To remove a brushmap that you've defined, just use the "Drop" button.

You can also restrict the brushmap to apply itself only on part of your object. This is done by naming a subgroup of faces to restrict the brush image to in the "Subgroup" requester.

Subgroups are discussed on page 35.

Once you know how to place individual brushes, you can start with the fancy tricks. Brushes overlay each other just like textures. You can put up to 4 brushes on an object, and they are applied in order. Many maps don't interfere, though; you could have a color map and a reflection map on the same object in the same place and both will work just fine.

To place two maps on an object in exactly the same position, use the Transformation requester and write down the position, size, and orientation of the first brush. Then copy this information to the requester in the second brush.

Repeating Brush Maps

Repeating brushmaps are a joy. They "tile" an object with an endless succession of images both side to side and top to bottom. The brush will repeat all the way out the end of the object. If you tile a ground, the brushes will go to infinity. The size of each tile is set by the brush axes, just like a non-tiled map. The brushes are placed next to each other with no space between them. You could draw a *very* detailed picture of a bathroom tile, and map it onto a wall. when rendered, the wall would be (surprise) tiled! I've used this to *great* effect for making very detailed sidewalks, rose trellises, brick walls, roof shingles, and golf greens. All of these are flat wraps.

Repeating brush maps will work with cylindrical maps (you'll get a vertical "stack" ascending up when you use Flat X Wrap Z). They have no effect on spherical brushmaps.

Actually, a bug in the current version of Imagine will not *allow you to repeat cylindrical maps.*

There isn't anything special you have to do with the brushmap axis; the size of the image is determined like it would be for a non-tiled mapping.

One additional option that is very useful is the "mirror" option. This makes every tile be a mirror reflection of its neighbor. The great advantage of this is that the edge colors *always* match, just like if your finger touches a mirror, your twin in the mirror will reach out and touch your finger at the exact same place. This might hide discontinuities in your brushmap if you want to hide the seams, or it might be a special effect you're looking for.

Repeating brushmaps aren't just for covering an infinite plain with your face. They can be an extremely powerful way to get very complex textures on an object. You can imagine drawing one very high resolution, high quality brick with scratches, pits, chips and tiny detail, then tiling it onto a wall. Presto! You have a brick wall with a lot of character, unlike the Brick texture which is too plain to fool anyone up close. This is probably the most useful aspect of infinite tilings. Making infinitely long or large objects like a sidewalk, golf green, brick wall, or rose trellis is quite easy with repeating brushmaps.

Animated Brush Maps

The last brush map ability is very useful. You can actually have *animated* brush maps which change every frame. Note that these pictures are not "Animbrushes" that Deluxe Paint will save. These are individual pictures, which means you can have a 24 bit "Animbrush." To use this feature, you should save the sequence of pictures you wish to show with names like

Mypic.0001
Mypic.0002
Mypic.0003
Mypic.0004
Mypic.0005
Mypic.0006

and so on up to however many pictures you have. Make sure to have *four* numbers in the extension. A file name of *Mypic.01* will *not* work. To use this sequence of pictures as an animation, you should use *Mypic* (with no period or number extension) as the brush file name, then set the "Max sequence #" to the number of pictures you have.

When the object is rendered over a series of frames, the brushmaps will be shown in order, just like you would expect. If you render more frames than the "Max Sequence Number," the brushes will repeat themselves starting from the beginning again.

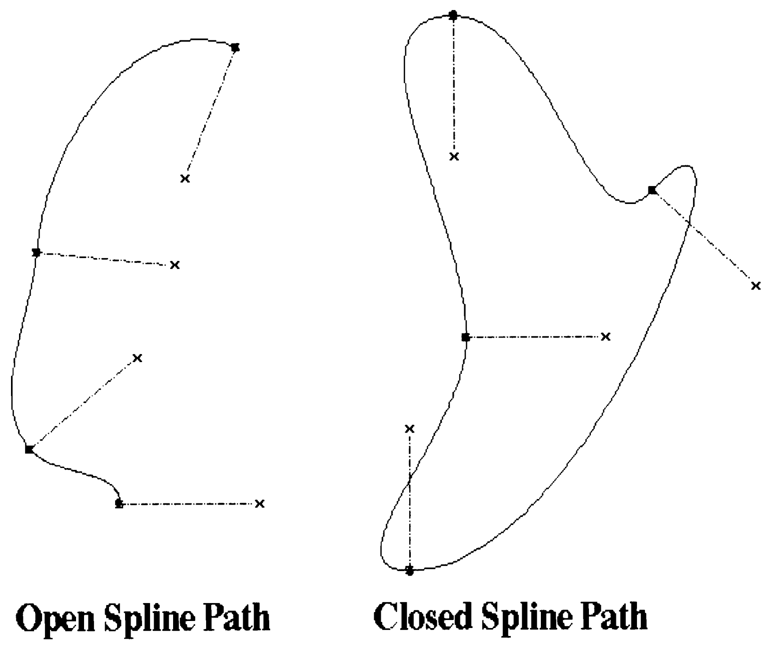


Figure 3.7: Two types of spline paths and their control points

3.5 Spline Paths

Wow, finally out of the Attributes section. That seemed to go on forever!

All objects (well, except for the perfect sphere) are modeled by a set of points, edges and lines. Imagine has another type of object that is used to model *paths* instead of surfaces; these paths can be used for a variety of functions from defining complex bent tubes to the path an airplane takes when animated. These paths can be built and edited from either the Detail or the Stage editors; since the paths are used routinely in both editors, it is very convenient to be able to manipulate them in both locations.

A spline path is a very specific type of object distinctly different than the objects you build to be rendered. Spline paths are *never* rendered; they are only a reference and path definition tool. Like "normal" objects, they can be positioned, scaled, and rotated, and have an axis that defines the "main" position of the path. Unlike a normal object, the spline path has no points, edges and faces. It instead consists of a curve that smoothly moves from one position in space to another. This curve is truly smooth, so even if you zoom into the path it will be a continuous curve without any bumpy straight line segments.

Spline paths are defined by "control points" that are individually editable. The path a spline takes is completely defined by these control points; the spline must pass through each control point, and the direction the spline is traveling *at* the control point controlled by rotating the control point. A spline path can have two or more of these control points, and the path can even be forced to make back to loop back upon itself. These closed paths are called "Closed Paths" while a path with two distinct ends is an "Open Path."

Figure 3.7 show the two types of spline paths, each with just four control points. The control points are easily visible, as they are shown by small axes with the path passing though the points in the direction of the point's Y coordinate. Notice how both the position and orientation of the control points affects the direction of the path. It is easy to make complex paths that smoothly follow any route you want by defining just a few points.

3.5.1 Creating Paths

There are two ways to create a path. The easiest is to use the "Add" command in the Functions menu and select "Open Path" or "Closed Path" which will add a simple path with two control points into your world. You can edit, move, and scale this basic path to make a spline curve of any complexity.

An alternative method of path creation uses axes. If you add a set of two or more empty object axes to your world, you can multi-pick the axes in order and tell Imagine to use the position and orientation of those axes to define the control points of a new spline path. The command "Make Open Path" or "Make Closed Path" will delete the axes and replace them with a path with control points in the same positions and orientations. This method of path creation is a little more complex, but sometimes it is easier to lay out the control points and manipulate them before they are included in a path.

3.5.2 Editing Paths

When you have an existing path you want to change, you can pick the path (it has its own clickable axis) and use the mode command "Edit Path" to enter into a path editing mode. This will make the control points of the path become visible. To exit from this editing mode, just select another mode like "Pick Groups."

In "Edit Path" mode you can click on a control point, and use the interactive manipulation commands to move and rotate them. You will see the spline path update in real time as you manipulate the control point. The control point can also be controlled by using the "Transformation" requester which allows precise numerical control over the point. You can also pick more than one control point at once and perform either interactive or numerical manipulations on it.

You can add new control points by splitting a path section between two points in half, which inserts a new control point at the split. If you pick one (or more) points, then use the "Fracture" command, extra control points will be added to the path. You are then free to manipulate these extra points any way you wish. Note that the last point in an open point cannot be split, since there is no path after it to fracture.

To delete a control point, just pick it and use the standard "Delete" command. A path must have at least two control points in it, so Imagine won't let you delete too many.

3.6 Molding Commands

In theory, you can create any object by adding an axis, then adding points, edges and faces. In practice, these are very low level commands; you generally use the more powerful commands like "Mold" and "Slice" found in the Object editor. The low level pick and add modes are built to give you the low level control that you sometimes need; however, they are more for defining basic outlines that are then used in the more powerful Object commands, or for touching up small details on nearly complete objects.

The "Mold" command controls a set of very powerful object transformations that can do more than just add and move points in your model. The different options of the Mold requester let you use your current object the basis of a variety of complex transformations. All of these commands are selected by picking your object, then choosing "Mold" from the Object menu. The "Mold Requester" will appear, which allows you to select from a variety of suboptions, briefly described here.

Extrude Copy your object many times as it is being moved, connecting each copy to the next with faces. Usually used with flat outlines.

Replicate Copy your object many times as it is being moved.

Spin and Sweep Copy your object many times as it is rotated, connecting each copy to the next with faces.

Conform Bend your object into a new geometry.

A complete description of each of the Mold options follows in the next sections.

3.6.1 Extrude

Extrusion is the most common Mold command. It allows you to take an object and "stretch" it in a third dimension by making many copies of the object and connecting each copy with faces. For example, you might make a flat outline of a circle. If you extruded the outline, you would produce a cylindrical tube, sort of like the way a macaroni machine extrudes dough by squashing it through strangely shaped holes. You have complete control over how far the object is extruded, how many copies of the object are used to form the extrusion, and what path the extrusion follows. You can even scale and rotate the object as it is being rotated.

Most objects you extrude are just outlines; they have no faces, just points and edges. If you are trying to produce a tube with a certain cross section, Extrude will do the right thing and add faces on the outside of your model like just like you want. You are certainly allowed to extrude faced models, especially if you want the beginning and end of your extrusion to be solid caps. This is very appropriate when you have a flat faced object (like a logo) and want to make a three dimensional version of it.

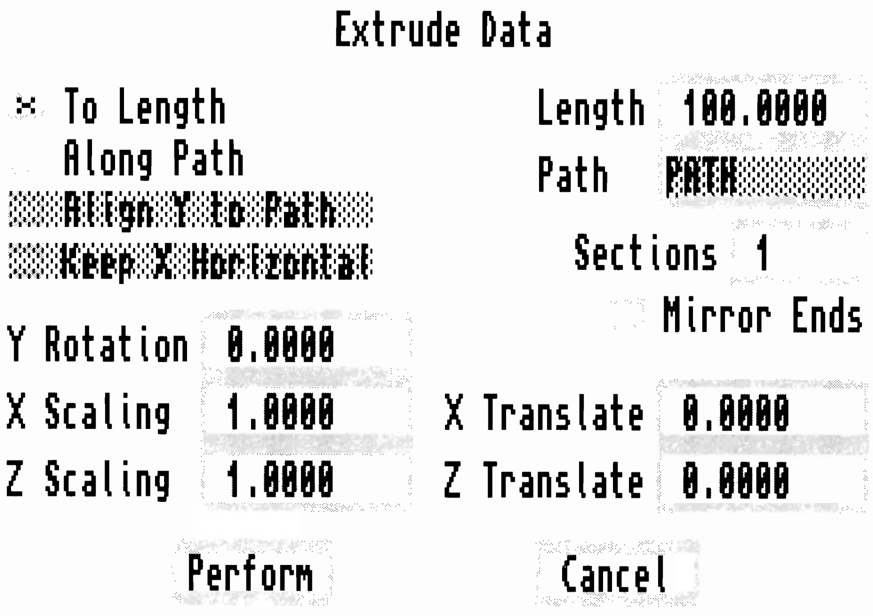
Making flat faced objects like logos is described in Section 3.7.1 on page 56.

When you pick an object, and use the "Extrude" option, you are presented with a requester like the one shown in figure 3.8.

There are two major types of extrusion, defined by what path the extrusion follows. A straight line path is the default, and is called "To Length" just to confuse you. The other option is to define a custom (probably much more complex) path by using a spline path, which is selected by the "Along Path" option.

Spline paths are discussed on page 48.

If you use the simple "To Length" option, the extrusion will occur in the direction of the object's Y axis. The length of the extrusion (how far back it goes) is selected by the "Length" gadget.



Extrude Data

[✓ To Length]
Along Path
Align Y to Path
Keep X Horizontal

Length 100.0000
Path PATH
Sections 1
Mirror Ends

Y Rotation 0.0000
X Scaling 1.0000
Z Scaling 1.0000
X Translate 0.0000
Z Translate 0.0000

Perform
Cancel

Figure 3.8: Information requester for extrusions

With the "Along Path" choice, you have several more options. The length and direction of the extrusion is already set by the path itself. You can elect to have the extruded object turn with the path (like a car would turn if it were going around a corner, always facing the direction it is moving) by selecting the "Align Y to Path" gadget. The "Keep X Horizontal" will keep your object from pitching up and down as it is being extruded. For most path extrusion you want to keep "Align Y to Path" on and "Keep X Horizontal" off. You must name the path you want to use by typing the name in the "Path" gadget. If you can't remember the name, use "Cancel" to get out of the requester, then use the "Find Requester" or just pick the path and use the Attributes Requester to look at the name directly.

The "Mirror Ends" option will flip the last copy of your object to that it is facing the other direction. This way the two "caps" of the extrusion will by symmetrical.

With either type of extrusion, you can decide on the complexity and detail of the path by telling Imagine how many sections the path should consist of. If you select just one section, Imagine will make the path by using two copies of your object and adding a "skin" between them. If you have 20 sections, Imagine will use 21 copies of your object, and will "skin" between each subsequent section. *When using a spline path, the higher this number is, the smoother your path will be.* If you use too few sections, the resulting object will look jerky and very amateur. If you use just one, you'll get a single straight tube extending from the start of your path to the end; probably not the effect you want, especially if the path was your name spelled out in cursive.

You can perform several manipulations with your object *as* it is being extruded. The "Y Rotation" requester will tell Imagine how much to rotate the object (in degrees) as it is being extruded. Luckily, you *can* enter a number like 720 to tell

Imagine to rotate the object more than once over the extrusion. The "X Scaling" and "Z Scaling" options allow you to scale the object as it is extruded, and the "X Translate" and "Z Translate" let you shift your object to one side or another as it is extruded. These commands all represent the *total* rotation, translation, or scaling that the object will undergo from start to finish. To determine how much each individual segment is changed from the previous one, divide the total offset, scale, or rotation by the number of segments.

These additional controls are pretty straightforward; Imagine will offset, scale, or rotate the object as it moves along the path. You can use this for several effects. To create a corkscrew, you can make a disk then move the disk's axis off to one side (actually off of the disk.) If you extrude this object in a straight line, you'll still get a circular tube. But if you rotate it while it is extruding, it will spin around its offset axis, leaving behind a spiral path. Similarly, you can make an object shrink or grow as it is extruded by using the scaling features. If you set both X and Z scaling to 0, your object will start out at full size and get smaller as it proceeds, eventually becoming a single point at the end. This is a great way to make Egyptian pyramids in the shape of your initials. For all of these advanced controls, keeping the number of segments high usually is a good idea.

Extrude is used a lot to form columns or tubes with arbitrary cross sections. It also is a necessary step in auto-facing outlines (which is described on page 57.)

3.6.2 Replicate

Replicate is almost identical to Extrude except for the fact that it does *not* leave a "skin" between the copies of the object. As seen in Figure 3.9, the requester is almost identical to Extrude's except "Number of Segments" has been changed to "Number of Copies."

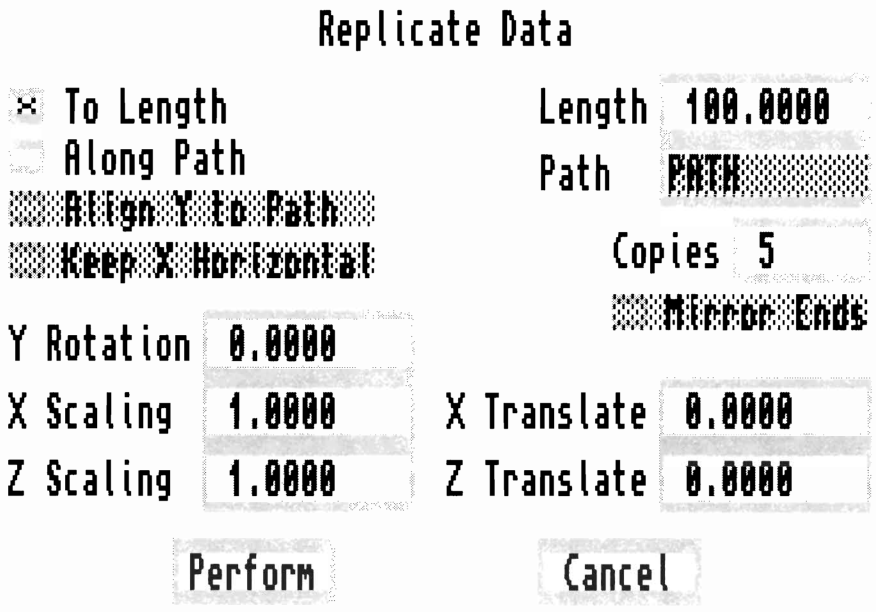
Replicate is useful when you want to make a set of copies of a shape, especially if you want to scale each copy so you have a full sized version, a slightly smaller one, an even smaller one, and so on.

The object that "Replicate" outputs are *not* separate objects; it is actually just a single object with a single axis.

One application of "Replicate" is to create nice tilings like lacework. If you have a flat, faced pattern that you want to copy into a larger piece, you can use "Replicate" to make copies. Take a nice flat "lace square" object with its Y axis pointing out of the flat square's plane. If you replicate the square with an extrusion depth of 0.0001 (so all of the copies are really the same "depth") and use "X Translate" and "Z Translate" to offset each copy, you can make a string of these squares in one direction. If you repeat the command with this new object, you can create a two dimensional tiling as large as you want. If you are careful with your X and Z Translation values, you can create checkerboards, double and triple gaps, and soft overlaps between all of the squares.

3.6.3 Spin and Sweep

Perhaps one of the classic object manipulation commands, "Spin" and "Sweep" allows the creation of radially symmetric objects like wine glasses or bowling pins. You can


Replicate Data

[✓ To Length]
Along Path
Align Y to Path
Keep X Horizontal

Length 100.0000
Path PATH
Copies 5
Mirror Ends

Y Rotation 0.0000
X Scaling 1.0000
Z Scaling 1.0000
X Translate 0.0000
Z Translate 0.0000

Figure 3.9: Information requester for replication

make a radially symmetric shape just by specifying an outline contour and telling Imagine to "Spin" or "Sweep" it. The shape will be rotated around the Z axis of the object, leaving a skin behind it (much like Extrude, except here the object is being rotated, not translated.) You can use these commands to build objects that could be milled on a lathe; anything that has radial symmetry.

The requester for these operations is particularly simple. There are two options to chose. "Spin Angle" will let you spin the outline all of the way around in a circle, or just part of the way. If you are making a wineglass, you probably want a full 360 degree sweep. "Number of Sections" determines the number of wedges in the spun object; 12 is a nice number for crystal glasses, but you might want to increase the number to 18 or 24 or so if you plan to get close to your object.

The only difference between "Spin" and "Sweep" is the way the top an bottom points in your outline are handled. In "Sweep," they are rotated around the Z axis of the object just like all of the other points in the object. However, if you "Spin" an object, these two points are *not* spun. Instead, they stay fixed in place, and faces are added from that fixed point to all of the copies of the *second* point in the spun outline. This makes something like a cone on the top and bottom of your outline. The advantage of "Spin" is that your outline is sealed at either end, making the swept object define a solid volume. For an open ended wine glass, "Sweep" is definitely the proper choice, as shown in Figure 3.10.

Nearly all objects that can be made with "Sweep" and "Spin" are easier to do in the Forms editor, which allows interactive control over the object shape and even lets you make sweeps with non-circular cross sections. See page 71 for an explanation.

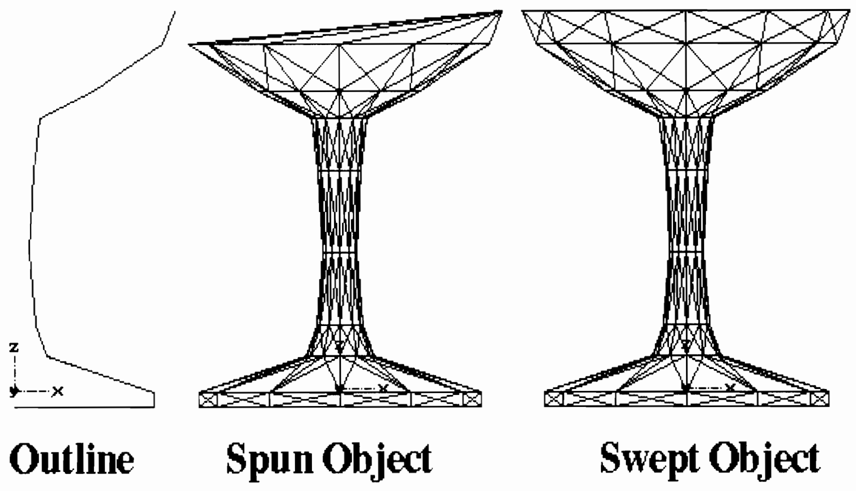


Figure 3.10: A "Spin" and "Sweep" of a wineglass outline

3.6.4 Conform

The Mold Conform commands allow you to bend your object in different ways. You can think of making a physical 3D model of your object out of stiff pipe cleaners, and then pressing that model onto a smooth hard shape like a basketball. The model will retain most of its original shape, but it will bend slightly to conform to the basketball's contours.

This is exactly what all three "Conform" commands allow you to do. You can take objects and add slight (or extreme!) amounts of bend to them by using an imaginary sphere or cylinder as a guide, or you can even use a custom path to guide the new object's shape.

Each of the different Conform types are similar in that they warp your object's shape, though they don't add or delete points, edges, or faces to your object.

Conform to Sphere

"Conform to Sphere" is exactly the case where you are pressing your pipecleaner model against a basketball shape. The "bend" depends on the position of your model's axis; Imagine starts to bend your object starting from this reference point. The bend is around the Y direction; that is, the object's Y axis points towards the center of this imaginary sphere the object is being warped around.

A neat trick of "Conform to Sphere" can make very nice teardrop shapes. Page 148 discusses this.

You can control the amount of bending by the two parameters that the "Conform to Sphere" requester gives you. "Sphere Radius" sets the radius of the imaginary sphere that you are bending your model around. "Object Radius" tells Imagine how big it should act as if your model is. The sphere radius roughly controls the angle and shape of the bend, and the object radius sets the amount or length of the bend.

These parameters take a *lot* of experimentation to set correctly. The best results seem to come when the object radius is 1–3 times as large as the sphere radius and roughly equal to the width of your true object. If there were a diagram I could include to make selecting object and sphere radius clear, I'd include it, but I have yet to discover a good description of Imagine's conform algorithm. Instead, I use trial and error, a poor substitute.

Conform to Cylinder

"Conform to Cylinder" works almost exactly the same way as "Conform to Sphere" does. Instead of using an imaginary sphere to shape your object, it instead uses a cylindrical shape.

Except it doesn't make neat teardrops.

Conform to Path

"Conform to Path" is a new option in Imagine 2.0. It allows you to use a spline path to guide your object. I think of it as a way to stretch an object along an arbitrary path. It is somewhat difficult to control, and like the Sphere and Cylinder Conforms, it takes trial and error to get the effects you are looking for.

When you pick an object and select "Conform to Path," the only information you have to provide is the path name.

"Conform to Path" seems to ignore your object's axis and use your object's world coordinates to orient your object. The object is stretched in the Y direction and then "bent" to follow a path that you specify. Ideally, if the object were a long thin tube stretching in the Y direction, if you use "Conform to Path" the tube would be stretched in such a way so that it follows the path much like an "Extrude" along the path would perform. You might build a model of a snake that was perfectly straight, then use "Conform to Path" to bend the object into a sinuous curve.

Still, it takes a *number* of trials to get useful results sometimes. I admit that this is one of the few options in Imagine where I'm uncertain exactly what operations Imagine is performing.

3.7 Outlines and Skin

A common type of object to build in the Detail editor is sometimes called an "outline." These are faceless, flat objects that consists of points connected by edges, usually in a closed loop. You can create these outlines manually, by adding a new axis to the world and using "Add Lines" mode to add the points and edges of the outline.

Add Lines is described on page 26.

These objects are very useful because they are perfect for extruding; you can use the outline as they are to define a cross section for an extruded tube. You can also add faces to outlines to make a solid flat object; this process is talked about in Section 3.8.1 on page 57.

3.7.1 Font Objects

A simple way to automatically create outlines of letters is to use Imagine's built in font converter. The command "Add" in the Function menu has an option called "Font

Object." A requester will appear, allowing you to choose a font (one of the ones in your *FONTS:* directory) and a line of text.

Imagine does not *use outline fonts to make text; it only uses bitmap fonts. Unfortunately it often chokes on* very *large fonts (more than 100pt).*

Imagine will automatically form an outline of the words you entered in the font you choose. It will ask if you want to add faces to the object. If you select this option, the object Imagine creates will be a solid (though flat) object, otherwise the object Imagine creates will be just the outline of the letters.

Imagine really seems to have trouble making smooth outlines, so they often require a lot of manual touchup to make a usable logo. Using a very large font size helps considerably.

3.7.2 Convert IFF/ILBM

The command "Convert IFF/ILBM" is similar to the "Add Font Object" command, except instead of adding an outline of letters, you can specify an arbitrary shape by making an IFF image in a paint program. When you select this option, Imagine will ask for a filename of an IFF image, and whether you want to autoface the outline.

The IFFs that you convert obviously determine the shape of the outline Imagine produces. The best images are simple two-color pictures without fine details like holes made of one or two pixels. Unfortunately, Imagine's IFF tracing algorithm is very blocky; a lot of detail is often lost so a smooth curve is sometimes converted into a jerky series of line segments. You can try to reduce this limitation by using larger IFF pictures.

This command is very useful for inputting shapes into Imagine. Even if you don't use the outline directly, it can help considerably in designing objects. For example, you might digitize the profile of a car, then use "Convert IFF/ILBM" to make a flat outline of that profile. Though the outline isn't really useful directly (if you extrude it, you get a *very* boring car,) you can keep it as a reference for the car object you *do* assemble.

3.7.3 Skin

Skin allows you to produce an object by using a series of outlines as a framework to stretch a "skin" of faces between. For example, you might make a series of ribs of an airplane wing and arrange them in the positions that they would be located in a real wing. The ribs might be different sizes and shapes, but you want to make a smooth surface that stretches over all of the ribs.

The only advantage of "Skin" over the Forms editor is the fact that Forms requires a closed, tube-like outline.

If you pick each of the ribs in turn and select the "Skin" function in the Object menu, Imagine will do exactly what you would expect. You can produce very complex objects with this command by defining the object's cross sections then making a solid model with "Skin."

Each outline used in Skin must be faceless, and most importantly, *each outline must have the same number of points in it.* Imagine just matches the outlines point-for-point, so if different outlines have different numbers of defining points the program won't be able to handle it.

Unless you are just making a quick object, "Skin" is really outclassed by the functions in Imagine's Forms editor. The Forms editor allows you to perform all of the functions of "Skin" and much much more, and has the added advantage of letting you interactively edit the outlines and their positions and letting you see the effect on your final model.

3.8 Slice

Slice is a powerful feature unique to Imagine. Unfortunately, it is still very buggy, even in Imagine 2.0. Nonetheless, it is a useful object manipulation command that can be used very effectively.

Slice allows you to "cut" objects into pieces by using *other* objects as a cutting tool. You can build a cookie cutter shape and use it to punch shapes out of a plane or any other objects.

To use Slice, you need to build or find an object to perform your cutting with. For the simplest cuts where you want to make a clean straight knife cut, you probably want to use a flat plane. Pick the object you want to cut *with* then hold the shift key and multi-pick the object(s) you wish to perform the cut *on*. Actually, *all* of the objects will be cut, but if you multi-pick more than two objects, the last objects will not cut each other; they will only cut and be cut by the first picked object.

When you have picked the objects, you can use the "Slice" option from the Object menu to perform the slice. Ideally Imagine should work for a moment, and you should have many smaller objects representing the parts left over after the slice. The parts will all be grouped together to an axis. You often want to pick and delete the parts of the slice you don't want any more, and sometimes using "Select" is very useful to sort out the different remnants. The sliced parts will all have names like *PART.18* so you might want to rename the objects.

"Ideally" because Slice often doesn't work: the next subsection on Slice bugs talks about this.

"Select" is discussed on page 16.

3.8.1 Adding Faces to an Outline

The most common use of "Slice" is to add faces to an outline, especially in earlier versions of Imagine which would not automatically face converted outlines.

The process of facing objects is pretty straightforward. The idea is to use the outline to punch a hole of the proper shape out of a flat plane that already has faces. This is just like the process of making cookies; you use a cookie cutter in the shape of a star, gingerbread man, vacuum cleaner (or whatever) to cut the dough in the same shape. When you remove the dough from outside the cut, you are left with a flat, solid object in the shape defined by your outline.

The analogy is apt, because the tools to do this in Imagine are perfect for this task. You can use "Extrude" to make a tube with the cross section of your outline. Just extrude with the default parameters which make flat extrusion of 100 units. Next, add a flat plane with the "Add Primitives" command in the Functions menu. It usually best to add a plane with just 1 by 1 sections, which is just a square formed by two large triangles.

Extrude is discussed on page 51.

If you scale the plane up so that the outline fits completely inside one triangle, you can set up the "Slice" that will form your object. Position the tube so that it is completely within one of the plane's faces and that the tube itself penetrates the

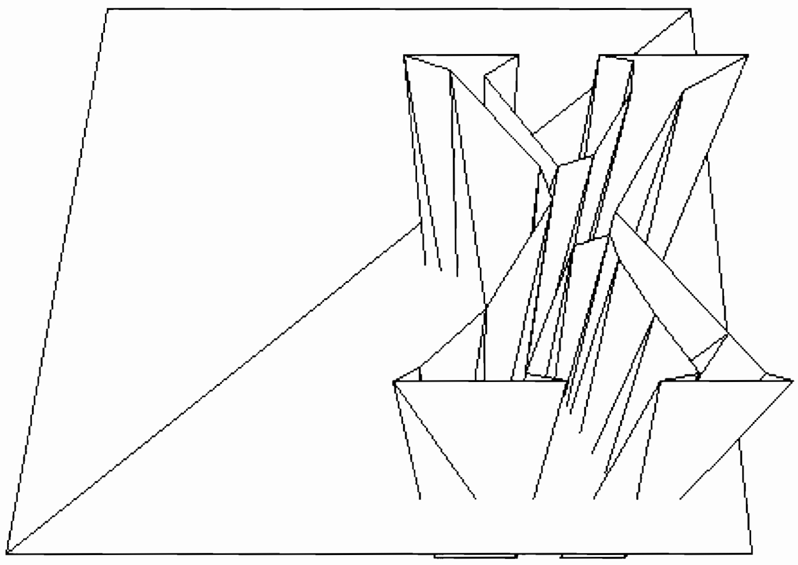


Figure 3.11: Perspective view of object positioning for a "Slice."

face, as shown in Figure 3.11 where a tube in the shape of the letter 'X' intersects a triangle in a simple plane.

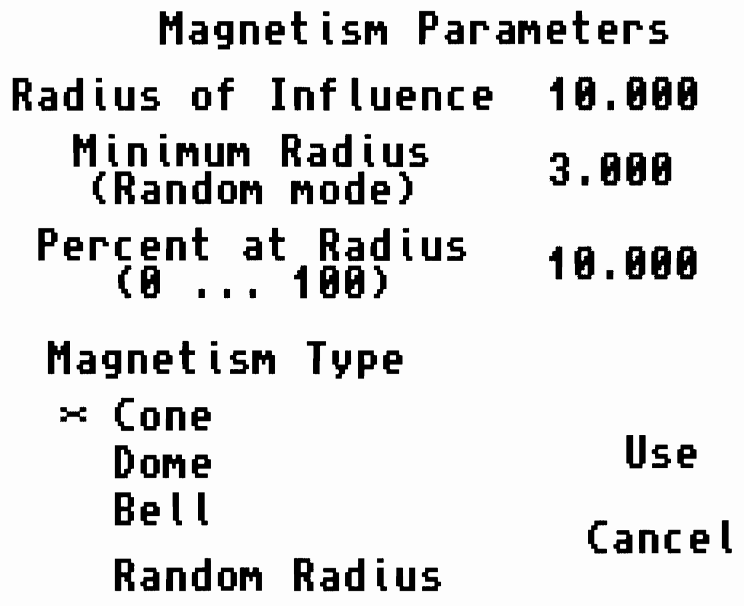
When you pick both of these objects and select "Slice," you will be left with at least four sliced pieces, including the tube which was also sliced in half. Pick the "cruft" like the tube remnants and the outside parts of the plane and delete them. You might also have to delete small sections such as the areas formed by holes in your outline (like the center of the letter 'O'.) You should be left with a faced version of your outline.

3.8.2 Bugs and the Evil "Error2"

Unfortunately, "Slice" isn't overly robust. It never makes a mistake in slicing; its output is correct. The problem with "Slice" is that many times the algorithm Imagine uses isn't capable of performing the slice and just gives up. Worse yet, sometimes "Slice," even in Imagine 2.0, will hang your machine forcing you to reboot.

When Imagine can't perform a slice, it usually makes one of two error messages appear. One says "An edge is too close to an edge" and Impulse suggests moving the objects into s slightly different position an trying again. Indeed, sometimes if you jiggle the object positions a bit, slice will be successful. However, with many complex objects this error seems to appear no matter how many times you readjust your objects.

The other error Imagine commonly displays is "Error2 splitting faces," which is annoyingly indistinct. There isn't any explanation for this error, so the only thing to do is to jiggle the objects and try again. Often an object just won't slice; you try and try but keep getting errors. In these cases there is nothing you can do except think


Magnetism Parameters

Radius of Influence 10.000

Minimum Radius (Random mode) 3.000

Percent at Radius (0 ... 100) 10.000

Magnetism Type

[✓ Cone]
Dome
Bell

Random Radius

Use

Cancel

Figure 3.12: "Magnetism Setup" information requester

of another way to manipulate your object, perhaps by manually deleting points.

Perhaps the worst error that Slice causes is a complete freeze of the program. If your objects are too complex the algorithm will completely lock up (your machine will still be running, it's not a system error) but you will be unable to escape from the slice, and the only way to recover is to reboot your machine. These lockups are particularly annoying since you can't tell when they have happened. On an A3000, if a slice ever takes more than about 15 minutes you've probably run into this bug.

When you are autofacing outlines using Slice, I find that Imagine can safely face the equivalent of a couple of letters (maybe 30-40 points and edges) safely, but four or more letters will consistently kill Imagine. Be careful whenever you use Slice and save your work before you use it.

3.9 Magnetism

Magnetism is an option that really only applies when you are in "Drag Points" mode. Instead of picking a set of points and moving them as a group, magnetism allows you to drag a set of points different amounts based on how far the point is from where you are pulling the object. If your object has many points in it, you can change your object's shape much like you were pulling and pushing on clay as opposed to analytically translating points.

Drag Points is described on page 26.

To use magnetism, you just have to enter "Drag Points" mode and select the "Magnetism On/Off" command. You can set the way magnetism acts by using the "Magnetism Setup" option to specify different options that change the character of the way magnetism moves multiple points.

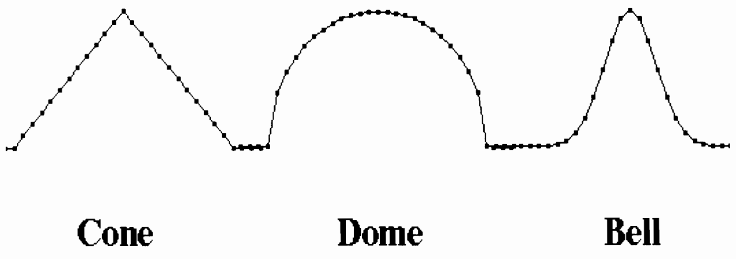


Figure 3.13: Different magnetism point attraction shapes

When this sub-mode is activated, you can click and drag points as before. However, when magnetism is activated an entire region (all of the points within a certain distance) will be affected instead of just one single point. When you drag the pointer with the mouse, nearby points will follow the mouse an amount related to how far away they are from the area you are grabbing. Figure 3.13 shows three different profiles formed by dragging a region of a straight horizontal line straight up. Points far away from the area that is dragged are unchanged, and the actual point that is dragged moves the amount you specify with the mouse. However, points around the dragged point are affected by different amounts, forming different shapes.

You can set the different shapes by using the "Magnetism Setup" requester. The default magnetism method, "Cone," drags points into a conical shape, with a sharp transition both at the point where you are dragging (the tip of the cone) and the base where unaffected (unmoved) points meet points that were too far away to move.

"Dome" makes a rounded half-sphere of the points that you drag. "Bell" makes a very smooth bell curve shape, which is smooth both at the point where you grab the points but also at the region where moved points merge with points too far away from the center of attraction.

The Imagine 2.0 manual has an inaccurate description of the "Percent at Radius" parameter, so be careful.

You can set the size of the region that Magnetism affects by setting the "Radius of Influence" value in the setup requester. Points beyond this radius will be unaffected by magnetism. You can also set how powerful the magnetism effect will be at that radius; you might want to have points *at* the radius of influence to be unaffected so they blend smoothly with the points outside the radius of influence, so you can set the "Percent at Radius" value to 0. If you set it to 100, all points in the region of influence will be moved the same amount no matter what shape you've specified. A value of 50% will make the points at the radius of influence move half as much as the original point you are dragging. Most of the time a value of 0 is best since it makes smoother transitions between points that have been dragged by Magnetism and those that were too far away.

Another option allows you to randomize Magnetism to give your object variety. If you are using Magnetism to build a landscape by pulling up mountains from a flat plane, the landscape might look unrealistic if all of the mountains were cones of the same width but different heights. If you set the "Random Radius" option in the magnetism requester, each time you drag a region, a new radius of influence is randomly chosen. You can set the minimum acceptable radius in the "Minimum Radius" gadget; you probably don't want to make mountains with a too narrow width.

Chapter 4

The Forms Editor

One of Imagine's most powerful object creation tools is the Forms Editor. Unlike the Detail Editor, Forms is not a general purpose object editor. Instead, Forms uses a specific method of object definition. The models it can make range from boat hulls and airplane wings to asteroids and soda cans. The editor is *very* powerful, but not necessary easy to describe.

The best way to think about the Forms editor is to view it as a way to define objects by their cross sections. Think of an object like an airplane wing. If you slice the wing with many parallel cuts from the front of the wing to the back, you'll see the airfoil cross section of the wing in each one of those slices. The wing gets thicker where it joins the plane, so those cross sections will be taller. Also, the cross sections aren't necessarily in line; for a swept-back wing, the cross sections are displaced backwards as you get to the end of the wing. The important thing to notice is that if you know the shape of each one of those cross sections as well as its position, you can pretty much predict what the overall solid shape of the wing is. You can think of each cross section as being a "rib" of the wing, and the wing itself formed by stretching canvas or sheet metal across those ribs to form the actual wing surface.

This is exactly the way the Forms editor works. You define a set of cross sections, and Imagine will use them as a skeleton framework to form the surface of your object. You don't have to worry about making the faces that define the surface; Imagine does it all for you. You have control over the shape of each cross section, and you can position and scale that cross section any way you like. Imagine will even let you stretch, bend and twist the cross sections. The best part is that all of this manipulation is interactive and particularly responsive.

One great advantage to the Forms Editor is the fact that the objects it makes are continuous; they are one smooth piece. This means that making smoothly rounded corners is easy. Organic forms in particular do *not* have sharp right angles and flat planes in them and are particularly well suited to creation in the Forms Editor. Any object created in the Detail Editor tends to be made of extruded or primitive objects joined together; there is rarely any smooth transition between the joined segments.

Another advantage of Forms is its precise control over your object. Forms is definitely the editor to use to build a complex continuous shape like a boat hull. In the Strategies Appendix (section C.2 on page 161), there is a very long essay on how to use Forms to accurately turn a complex shape into a usable 3D model.

Morphing is completely described on page 103.

A final advantage of Forms is in making animations where models *morph,* or change shape over time. Since the geometry[1] of your object is fixed, it is easy for you to build different objects for Imagine to morph to one another.

4.1 Creating New Forms

Unlike the Detail editor, where you just start adding points, edges, and faces to your object, you have to tell Imagine a little bit of information about your object before you can start building it in the Forms editor. This information tells Imagine how complex to make your object, whether it is symmetric, whether you want to be able to offset, tilt, and scale the cross sections, and if your object is hollow (like a tube) or a solid.

When you start the Forms editor, you will be presented with the standard quad-view display. There are actually some slight differences in the way the three main views work, but this is described later.

When you start the Forms editor, you can either load a previously constructed Forms object by using the "Load" command from the object menu, or start a new form from scratch. *You cannot load objects created in the Detail editor into Forms!* The way Forms builds its object requires a very specific model geometry that your Detail-created object probably doesn't have.

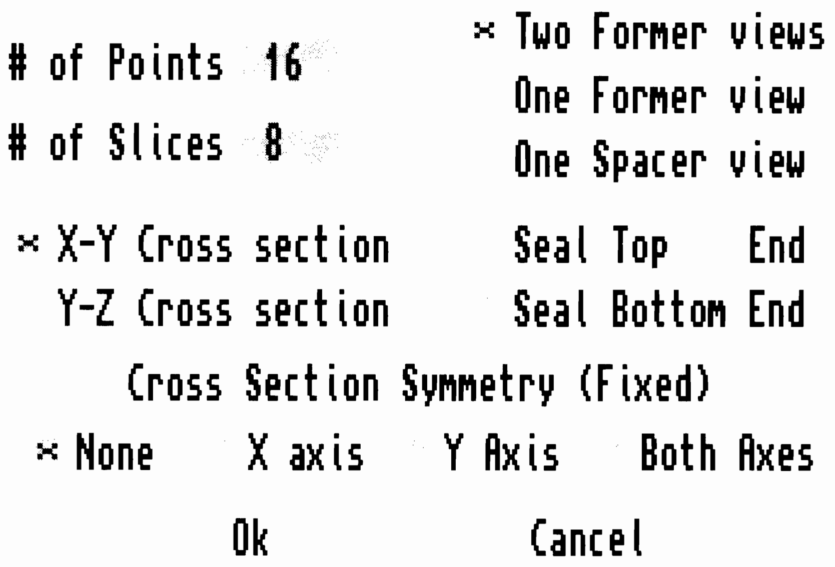
This specific geometry comes from the way Forms are defined: by their cross sections. Imagine creates a set of "slices" of your object. Each one of these slices has the same number of points. You can change the shape of these slices (or cross sections) by dragging the slice's points around. Imagine simultaneously lets you position, scale, tilt and bend these slices. To build your final object, Imagine adds faces between each of the slices to form the sides of your object. Since each slice has to be connected to the next, Imagine requires that each slice have the *same number of points.* This way there is an exact correspondence between all of the slices and Imagine doesn't have to make a possibly incorrect decision about what part of a slice matches with what part of the next slice.

This really isn't a terrible limitation, because you can always just use many points to define each cross section if you want a detailed object. You can have an arbitrary number of slices as well. You just can't have one slice with 10 points and the next with 23. Luckily, the Forms editor will allow you to add or delete points in a slice even after you've started to build your object, and you can also add or delete extra slices. Note that adding a new point to a slice will add a new point to *every* slice. See why?

When you use the "New" command from the Object menu to start a new object, you will be shown the requester in Figure 4.1. This allows you to set up some of the initial options that define your object and the way it can be manipulated.

Your first choice is how complex your object should be. Each slice has a certain number of points that define it. The more points you select, the larger your model will be, and the more control you will have over the shape. For an object like a lighthouse, you need just enough points to make a nice circle (maybe 16 points) but

[1]Actually, topology, if you want to quibble.



# of Points 16

# of Slices 8

[✓ Two Former views]
One Former view
One Spacer view

[✓ X-Y Cross section]
Y-Z Cross section

Seal Top End
Seal Bottom End

Cross Section Symmetry (Fixed)

[✓ None] X axis Y Axis Both Axes

Ok Cancel

Figure 4.1: Information requester for new Forms objects

if you were making a complex boat hull, you might use 25 so you have enough points to form a detailed shape. You can always add or delete cross section points, but it is usually easier to start with approximately the number you expect to need or even less. It is easier to add points than delete them, since the points you delete might be more important that you once thought! Also, if you have too many points, the cross sections sometimes get a little ragged just because it is difficult to position so many points accurately at once.

You can set the number of points in each slice by the "# of Points" gadget. For most objects, the default of 16 is a nice number to start with. The "# of Slices" gadget will (logically) set the number of slices in your object. Again, there is a certain decision on how many slices your object needs. A lighthouse has a very regular and smooth profile that might be characterized by only two slices (one at the top and bottom) but a boat hull changes character many times along its length, so you might use 20. Like the points in each slice, you can add and delete extra slices, so if you don't guess perfectly to begin with you can still correct yourself.

You have a choice of your object's orientation. If you are building an object like a lighthouse, selecting "X-Y Cross Section" will form your object out of horizontal slices, proceeding up and down. If you are building something like a boat hull, you might want the cross sections running vertically from left to right (stern to bow) so you can use "Y-Z Cross Section." The two options function nearly identically, except your model is oriented in a different direction to make it easier to view and visualize. A lighthouse on its side just looks weird.

Yeah, these same two examples get repetitive. Oh well.

You might often want each cross section to be symmetric. For example, a boat hull is bisymmetric; the left and right halves of each cross section are just mirror images of each other. You can tell Imagine to keep the cross sections symmetric by using the "Cross Section Symmetry" buttons at the bottom of the gadget. The default, "None," will give you complete independent control over every cross section point. If you select "X axis" or "Y axis," there will be one axis of symmetry. When you move a point on one side of the symmetry axis, its corresponding point on the other side of the axis will move with it. "Both Axes" will let you set symmetry for both axes, so every point will have *three* corresponding points that move with it. To have a bisymmetric cross section, you must have an even number of cross section points; for symmetry along both axes you need to have a multiple of 4 cross section points.

Actually, if you use a Y-Z cross section instead, your symmetry choices change to Y and Z.

The options "Seal Top End" and "Seal Bottom End" allow you to close each end of your object. You can think of the defining cross sections as making a long tube formed by ribs with a skin stretched over them. The ends are not closed, and your object will show this unless you select this option. If you close the ends, the last cross section (the one at the end) will be shrunk down to a single point, forcing the object to seal the end.

Right and Left if you are using a Y-Z oriented cross section.

The last options you have define the type of view you have of your object. The "Two Former," "One Former," and "One Spacer" options affect both your display and the control you have over your object. The "Former" views are more powerful, but often unnecessary and confusing. Most objects I build are with the Spacer view. The effects each of these options has on your view is described briefly in the next section, and a detailed description of the three different modes is found in Sections 4.3 and 4.5.

4.1.1 Modified Quad-view

The Form Editor does *not* display objects in its quadview in the same manner the Stage and Detail editors do. Instead it uses the Top view to display a single cross section of your model, and the Front and Right views to show the position, scale, and bend of each of those cross sections. In the Spacer view, there is *nothing* in the Right view and only a line of connected points in the front view. In the Former views, you see two lines of connected points in one or both of the Front and Right views.

It uses the Right view for the cross sections if you are using a Y-Z oriented model.

The perspective view of your object is accurate (it shows your object's true shape,) and you can select wireframe, solid, or shaded mode by using the View menu just like you would in the other editors. I prefer solid because many forms get very complex very fast and it is difficult to see the basic structure of objects in the tangled wireframe mode.

4.2 Cross Section Definition

There are actually three modes to the Form Editor. The default is "Edit" which allows you to click on points in the cross section shown in the Top view and just drag them into a new shape. The perspective view will update, showing the new configuration. Just like in the Detail editor, you can press and hold the shift key to multi-pick many points at once, so you can move many points simultaneously. If you've turned on cross section symmetry from the "New" object requester, the symmetric points will move along with the point(s) you are moving.

Right view with Y-Z cross section!

All of the standard view commands that you are used to still function the same way in Forms, including zooming and moving your viewpoint in the world and the point pick method.

If you aren't using "Lock" mode as a pick method, you can still pick one or more points and use "Snap to Grid" to move each point to the nearest grid location.

Even though "Edit" mode is the most common mode, it is really quite straightforward. Defining a cross section is performed by just dragging the points into the right shape.

You can add new points to a cross section by selecting the "Add" command from the Mode menu and clicking on *current* points. A new point will be added on the line connecting the point you clicked on and one of its neighbors. You can also position this new point by keeping the mouse button down and dragging the point to its new location. You can edit these new points in Edit mode at any later time. Remember that adding a new point to a cross section will add a new point to *every* cross section.

It is so simple to add new points that defining a new object with a lot of starting points to begin with is usually not a good idea. It is *very* difficult to control that many points at once, especially if they aren't in a good initial configuration. I find that it is easier to start off with a few points (like 8) and arrange these in a coarse approximation of the object to get general shape and proportions, then add more points for details. Trying to make large changes to a cross section with a lot of points can be a pain.

If you wish to delete a cross section point, the "Delete" mode will allow you to remove points that you click on. Like "Add," every cross section will have that same point deleted, since each cross section has to have the same number of points.

4.3 Spacer View

To understand how to make more than one cross section simultaneously, first you have to understand how the Spacer (or Former) views work. In the Spacer view method, while the Top view shows the shape of your object's sliced cross sections, the Front view shows the position of each of those cross sections. All of the slices are aligned along a "spine," sort of a centerline of the object. Each cross section is represented by a single point on this "spine," which indicates its position. You can click on one of these points and drag it up or down to change its position, but you cannot move it to the side. (The Former views will let you do this, at the cost of a *much* more complicated display).

Right in Y-Z oriented Forms: I think you get the drift, so I'll stop reminding you.

Each slice is "attached" to one or two others by a vertical line. These tell you which slices Imagine will skin between; if the slice is connected to only one other slice, it is one of the two end slices. Whenever you pick a slice and move it, Imagine will also highlight the lines that connect the slice to its neighbors for reference. Although you cannot move a slice to one side or another (and keep it there), while you are dragging the slice, you can move it sideways just so you can more clearly see where the slice is connected.

I like to think of the Spacer view as showing the cross sections as a big stack. The spacing between each point on the line in the Spacer view is just the separation between the slices. You can edit the stack spacing by just entering "Edit" mode and dragging the points.

Much like the editing cross sections, you can use "Add" mode to add new slices, "Delete" to remove them, and "Edit" to move them around.

4.4 Multiple Cross Sections

Editing a cross section is pretty straightforward, but the whole point of the Forms editor is to define your object by a *series* of these cross sections. If you are able to see how the Spacer view shows each sliced cross section, you can use the Spacer view to tell Imagine what cross section you want to edit.

You object may have as many cross sections as you want. In fact, Imagine will let you define just a couple of *key* cross sections, and it will interpolate the inbetween ones. For example, if you wanted to build a tower that had a square base and gradually turned into a tapered cylinder as it ascended, you could define just two cross sections (a square and a smaller circle) and let Imagine interpolate the inbetween cross sections. If you wanted the tower to change from a square cross section to a circular cross section midway up, then back to a square cross section, you could define just three cross sections. If there is a region where you want the cross section to be constant (no interpolations) it is easy to tell Imagine to keep a range of slices constant.

If you look at the Spacer view, each of the points represents a cross section. By default, Imagine assumes each cross section is the same shape, so a defining a single shape will set the entire object's cross section. This default cross section is defined to be the very top cross section. If you wanted to add a second cross section, you would use the "Make Key" command from the CrossSection menu. This tells Imagine that you want to add a second key cross section to your model. When you select this option, Imagine will ask you which cross section to define as a key. In the Front view, where the different cross sections and their spacings are shown, the top point (representing the top cross section) will be highlighted; this shows that this cross section is *already* a key. The unhighlighted points all represent interpolated cross sections. You've told Imagine that you want to make a new key, which means that you want to turn one of the interpolated cross sections into a defining cross section. By clicking on one of the *non*-highlighted cross sections, you'll create a new key cross section.

This new cross section is now the one displayed in the Top view. Initially, it will look exactly as before, but you can start editing it *without affecting the shape of the other keys.* The title bar of the screen will also display something like "Edit, Cross Section #15" which will tell you which slice you currently are manipulating.

To edit different keys after they've been defined, you use the "Select" function from the CrossSection menu. Each key cross section in the Front view will be shown in a highlighted color, and you can click on any of them to make that cross section appear in the Top view. From there, you can edit the cross section or do what you like with it.

To change a key cross section back to a normal interpolated cross section, the command "UnMake Key" will again highlight the current keys and allow you to choose the key you want to trash.

If you have more than one key defined, sometimes it is useful to copy the cross section from one key into the current one, especially if you are making something like a totem pole which might have cross sections that repeat often. The command "Copy From" in the CrossSection menu will highlight the current cross sections and allow you to choose which shape you want to copy. If you want to abort the copy, you can use the "Cancel" command also in the CrossSection menu.

If you want to have a range of cross sections be constant (you don't want them to be interpolated) you just have to set up an extra key. Make a key cross section (with "Make Key") at the beginning and end of the range of cross sections you want to keep constant. Edit one of them to be the shape you want the entire region to be, then when you're done, select the second key. Use "Copy From" to copy the shape from the key you've defined. Since both keys are identical, the cross sections inbetween are constant.

4.5 Two Former Views

The Spacer view is pretty straightforward, but an advanced display called the Former view gives you complete control over the positioning of your cross sections. Instead of just defining the separation between each cross section, you can move the cross sections from side to side, scale them, tilt them, and even bend them.

All of these advanced manipulations occur in the "Former Views." This method of viewing the placement of the cross sections is similar to the Spacer view in that the cross sections are represented by points connected by lines. But instead of having a single point for each cross section, there are actually *four*. These four points that represent the position on the cross section can be thought of as being boundaries of the cross section at the extreme four compass directions; 0, 90, 180, and 270 degrees.

Actually, when you realize that each cross section is controlled by its four points, the Former views become a lot more comprehensible. You see the four points, two in each view. If you move all four points up the same amount, you will move the cross section up along the object the same amount, *just like the Spacer view.* But since you can control more than one point, you can perform much more complex manipulations.

The best way to understand how the four points define the size, position, and tilt of the cross section is to think of the four points as being hooks that the shape defined by the cross section is stretched between. If all of the hooks are level, the cross section is level. If you move the hooks apart, the cross section will be stretched and get larger. If you move two adjacent hooks higher than the other hooks, the cross section will become tilted. You can even move two opposite hooks up, and the other hooks down, which stretches the cross section into a shape like a Pringles[2] potato chip, or a horse saddle.

For example: If you move the two points in the Front view to the right, the entire cross section will be displaced to the right by that same distance. If you move one of the front points up and the other down, the cross section will be tilted. If you move all the points apart, the cross section will be scaled.

In each manipulation, the shape of the cross section stays the same; it is just

[2]Pringles is a trademark of Procter and Gamble Inc. For questions or comments about Pringles, call (toll free!) 1-800-543-7276.

moved, scaled, tilted, or bent. The great part about the Former view is that you can make some incredibly complex objects. If you can make a totem pole defined by its cross sections using the Spacer view, you can use the Former view to make a totem pole that is larger at the base, corkscrews as it ascends, and is tied into a knot at the top. True, this would take some work, but it is possible with the control you have.

Editing the Former control points is exactly as it is in the Spacer view. When you drag a point, Imagine will draw "ghost" lines to the positions of the other three control points just so you can see their relative positions. This can be very handy in a confusing object. When you add or delete cross sections, remember that you will see *four* points appear or disappear, since each cross section is controlled by four points simultaneously.

The starting position of the Former points places them in a rough spherical shape: the cross sections at the top and bottom get scaled down to close in to seal off the "tube." I usually first drag the points top form an open tube, since most of the time I am designing objects with a straighter geometry.

There is nothing you can in Spacer view that you cannot do in Former view. The only disadvantage with Former view is the display complexity. Your final model will have the same number of points, edges, and faces no matter what method you use.

4.5.1 Former Symmetry

Most of the time you don't need to move each of the four control points independently. The Former view has a very useful symmetry feature to fix this problem. If you turn on symmetry, whenever you move a point, its corresponding point(s) will follow and put themselves in a symmetric position.

Symmetry is controlled from the aptly named Symmetry menu. There are five options for symmetry:

Off Every control point is completely independent. The default.

Front The 0 degree points will follow the 180 degree points and vice versa. The 90 and 270 degree points are completely independent.

Right The 90 degree points will follow the 270 degree points and vice versa. The 0 and 180 degree points are completely independent.

Both The 90 degree points will follow the 270 degree points and vice versa. The 0 degree points will follow the 180 degree points and vice versa. Think of it as "oval" symmetry.

90 Degree Every point will follow the corresponding point in every degree view. The cross section will be forced to be undistorted and level.

These symmetry options are very easy to use. Note that turning on symmetry does not immediately make the cross sections symmetric; only points you touch and move will change. Also, unlike the cross section symmetry, you can turn it on and off at will.

Most graceful objects have at least one axis of symmetry, many have two, and some have 90 degree symmetry.

4.5.2 One Former View

The One Former view is a compromise between the Spacer view's simple display and the Two Former view's versatility. It allows you to use two control points to position, scale, and tilt each cross section instead of four. You can't bend the cross sections, you can only tilt them in two directions, and you can only scale them in one dimension, but it might be all you need to make your object.

All of the controls from the Two Former view work identically, except that there is only one type of available symmetry. It is labeled "Front Side" but it acts more like "90 Degree," so your cross sections will become undistorted and level.

4.5.3 Swept Objects

If you are making radially symmetric objects like wine glasses or bowling pins, you can use the Detail Editor commands "Sweep" and "Spin," but it turns out that the Forms editor can do the same job much more easily. If you make a Forms object with a single cross section in the shape of a circle, you can use either of the Former views to make a radially symmetric object. Make sure to turn on 90 degree symmetry, then just move the points into the wine glass or wedding cake shape you want.

Which are described on page 52.

The advantage of using Forms for defining these radially symmetric objects is that your updates are interactive: you can see how a certain change will affect the appearance of your object. You can also easily add and delete points to increase or decease your model's complexity.

4.6 Saving Forms

You can easily save the current object by using the "Save" command in the Object menu. A saved Form object can be loaded back into Forms for future editing, or you can load it into Detail. In the Detail editor, you will again see a traditional view of your object, and you can edit your object any way you like, including adding attributes and brushmaps. *After you manipulate your object in Detail, you cannot load the modified object back into Forms!* The specific geometry of Forms objects is not necessarily preserved by the Detail editor. To avoid getting yourself in trouble, you might want to use a convention like mine, where I label Forms objects with a file extension *.ifm* and Detail objects with *.iob*.

4.7 New Forms Objects

Didn't we talk about this a long time ago? Well, almost. If you select the "New" option to make a new Forms object while you already have an object loaded, you can "convert" it into another Forms object. Imagine will ask if you want to keep the same cross sections or the same Form points. If you say "Yes," Imagine will start the new object with the points arranged in the same cross section or Former shapes. This is very convenient when you want to automatically reduce the complexity of your object.

Even more importantly, this is the *only* way to turn on and off the cross section symmetry or change from a Spacer view to a Former view or vice-versa. *Changing from a Former view to a Spacer view will lose all of the extra Former information!* Be careful!

Chapter 5

The Cycle Editor

The Cycle editor is one method of animating the objects you have built in the Detail and Forms editors, especially when that animation is a complex, integrated motion like a walking man or flying bird as opposed to simple motion like a ball flying though the air. The Cycle editor gives you control over the motion of an entire set of objects relative to each other, especially when the objects are supposed to be connected together but free to move in complex ways, like an arm connected to a shoulder. Trying to make a human figure wave its hand by using the Stage Editor's path commands would be insanely difficult; Cycle provides a way of easily defining the group's internal motion, which then can be "acted out" on the Stage.

Most users ignore Cycle; like Forms, it isn't a necessary step in creating complete renderings. However, to create complex motions, Cycle is an invaluable tool in the animation process. New users might very well want to skip this chapter and learn how to use Detail and Stage first. But if you want to make flying birds, tightrope walking men, or helicopter objects that automatically spin their rotors, learning Cycle is well worth the time it takes to understand it.

Cycle produces objects that can be loaded into the Stage Editor much like those saved from the Detail Editor. Unlike the Detail editor, Cycle doesn't define the appearance of the objects, it defines the *motion* of those objects, and to a lesser extent, the object's organization. Cycle "objects" are really a combination of a set of normal objects and a description of how those objects should move relative to each other over time. The Cycle editor is where the positioning of these objects and their internal motion is defined.

It is important to understand what sort of motions should be made in the Cycle Editor and what types are made in the Stage. Cycle is designed for motion "internal" to an object. The swinging of a clock's pendulum, the motion of a baseball player's arms as he swings, and the rotation of a helicopter's rotors are "internal." The idea is that as a whole, the object isn't moving, just changing its pose or configuration. The clock stays still on the wall, and the baseball player isn't running across the ground as he swings. Making entire objects follow paths is exactly what is best done in the Stage editor, like defining the path of the baseball after it is hit by the bat, or the path a car takes as it turns a corner on a road. The beauty is that these two types of motion can easily be combined! The helicopter, with the Cycle-animated spinning rotors, can simultaneously follow a banking path through the air created in the Stage editor. A cycle of a man jogging in place combined with a straight path in Stage for him to follow makes an instant running man. Learning which editor is the proper one to create the type of motion you are looking for is pretty easy when you ask yourself if the object is moving as a whole, changing shape or pose, or both.

5.1 Cycle Mechanics

The Cycle Editor has the customary quadview display, showing top, front, right, and perspective views. Unlike the other main editors, you don't actually see your object in the three main views; instead you see and manipulate an abstract "skeleton" of your object with only the perspective view representing what your object looks like when actually rendered. This skeleton view can be very convenient, since parts of your object may be manipulated quickly and easily. The display of surface detail is usually irrelevant when you are just trying to define orientation and position of a part of your model, so the sparser skeleton display can reduce the complexity of the computer display and make it easier to see the organization of your model as opposed to the irrelevant object surface. There *are* occasions when it is important to manipulate your object when you can clearly see these details; the later section *Pose Commands in Detail* on page 83 talks about this complementary method of cycle design.

The basic organization of a cycle object consists of a set of "links" which are connected in a hierarchy, where a link is connected to its "parent" and can also have "children" links of its own attached to it in turn. The very top level of the hierarchy is the axis of the object, which is used in Stage to define the position, orientation, and size of the cycle object as a whole. *Every* link is below this main axis in the cycle hierarchy.

Links are displayed in the Cycle Editor as long, thin, flat diamonds, with the Z axis of the link along the long centerline of the diamond, and the X axis along the short direction. A small line points in the direction of the positive X axis so you can tell if the diamond has been twisted around to face the opposite direction. The main (Z) axis of a link is defined as being at the base of the diamond with the end of the Z axis pointing straight through the diamond and *ending directly on top of the axis of the link's parent.* The links are always directly attached to their parent in this manner, and understanding this is critical to understanding how cycle links are set up.

These links can be positioned, rotated, and scaled in any way, though Imagine will always keep the top end of the link connected to its parent. The basic mechanics of Cycle allow you to position the links in one pose, then position them again in a new pose in a later frame. When animated, Imagine will smoothly move each link from its initial pose to its final pose. There can be as many of these poses as you like, spaced as frequently or as sparsely in time as you desire. When animated, your cycle will move to each pose in turn, eventually moving from the last pose to the very first one again to begin the cycle over again (hence the term "cycle object.") You have a lot of control of cycles when you render; you don't have to loop them, you can start the cycle in the middle of its loop, and you can even make the cycle run in reverse. The

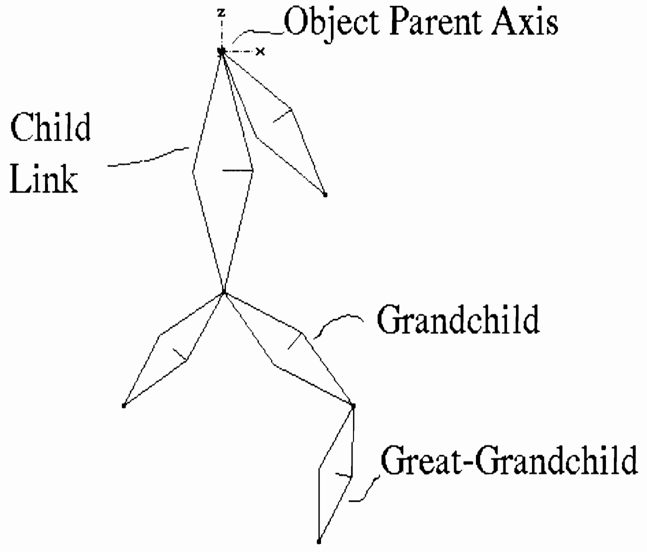


Figure 5.1: A Cycle link hierarchy

repeating nature is just a convenient way to define the sequence of poses the object follows.

The Cycle editor gives you tools for building and manipulating the links that define these poses. It also associates the links with the real objects they represent. Every link has the ability to define a real object's position in the cycle; the position, size, and orientation of the link directly determines the position, size, and orientation of the object the link is associated with. One major constraint of links is that they always are connected to their parent, which "keeps you honest" and prevents arms from detaching themselves from the shoulder, or rotorblades spinning off into the air. If you *do* want detachable objects, it is possible to add a link with no real object associated to it. This kind of link, called a "null link," is used as a spacer to physically separate a child from the main body of the cycle object. Since null links are not rendered, there is no visible connection between the child and main object in your final picture or animation.

Null links are discussed on page 78

A final point concerns exactly how an object associated with a link is rendered. **The object will be scaled, rotated, and positioned so that its Z axis exactly corresponds to the size, orientation, and position of the link that represents the object.** What this means is that the axis placement of the object when you save it in the Detail Editor is *crucial*. For an object representing an upper arm attached to a torso, you probably want to position the axis of the upper arm so that the center of the axis is at the base (the elbow) and the Z axis points up the arm and ends right at the end of the arm where the arm would join to the shoulder. When you associate this "upper arm object" with a link, the arm will be accurately placed so that it attaches to the shoulder in the proper place, and has the correct size and orientation. The link is *always* connected to its parent, so the top of the arm will always be connected to the torso at the shoulder. The axis of the arm is in the same location as the end of the link, so a child object (probably the forearm, or maybe a prosthetic chainsaw) would attach to the correct place on the upper arm.

Axis positioning of objects to be assigned to links is discussed in detail on page 84.

The skeleton of links and the assignment of objects to those links defines your cycle object's appearance. There is a function to assign each link a real object (created in Forms or Detail) that will be rendered at the location the link defines. As the links change position from frame to frame, the assigned objects will follow the links exactly. This exact relationship leads to one of the biggest dangers of Cycle; that *when you move a link, your objects might change size when you don't want them to.* Why is this? Since every link is *always* connected to its parent, if you move the link any closer or farther away from its parent, the link *has* to grow or shrink, and the object associated with the link will grow or shrink as well. This might be the effect you are looking for, but most non-cartoon objects keep their sizes constant. This limitation is really not a restriction, but a necessity to keep children connected to their parent.

5.2 Creating and Positioning Links

It is important to understand how cycles work before you try to build any, since the exact way that objects are associated with links can get confusing. You could get easily be lost when Imagine seems to put objects in weird places, or especially when a figure's head starts swelling when all you're trying to do is make him nod. Read and understand the previous section if things get weird and you don't know why!

Creating a cycle from scratch is very easy. The Cycle Editor displays the standard quadview of the world where you manipulate your figure. Remember that the Cycle display only shows the position and orientation of *links* in the main three views, and the "actual" object is only shown only in the perspective view. When you first start the Cycle editor, you are shown a single axis in the quadview, which can be the basis of a new cycle object if you want to start building one from scratch.

Just as in Detail and Forms, there are certain modes that the editor can be in. The current mode is always displayed in the title bar along with information about which frame you are editing and whether that frame is a keyframe or not.

The keyframe and animation commands are talked about in the next section, starting on page 79.

When you first start the Cycle editor, the default mode is "Add," which means that Imagine is ready to add new links to your cycle object. By clicking and holding the left button when you are pointing at the cycle object's axis in any view, you can drag out a link of whatever length and orientation you desire. Remember that the length of the link will directly determine the size of any objects they represent! You can also make children links by clicking and holding the left mouse button on the end of a current link and dragging. Usually when you initially build the skeleton of links, you just want to roughly place the links in the right positions and sizes and get the right number and structure of links set up, since other modes are more useful for actual positioning and fine-tuning of your cycle object. This "Add" mode is very straightforward, and can be entered at any time by selecting "Add" from the Mode menu.

A complementary mode, "Delete," simply deletes a link when you click on its end. Any children of the deleted link will also be removed.

Other modes allow easy manipulation of the link locations. One mode called "Move" is easy to understand. By clicking on the end of a link and holding the mouse button, you can interactively move the link around in any of the three main views. You can easily position the end of the link to any position you care to, but remember it will always stay connected to its parent, so if you move it closer or further away from the parent *the length of the link will change!* If you want to keep the size constant and just change the orientation of the link, two other modes are more appropriate and will not affect the size of your object.

The "Twist" mode allows you to rotate a link around its long Z axis, in effect spinning it around in place. You might want to build an airplane with a propeller as a separate link. In this case, Twist will let you spin the propeller around without actually moving the link's position. The propeller of your model will spin, but stay in one place. If you click on the end of a link, moving the mouse with the button held down will spin the link around.

"Pivot" is used to rotate an object around the point where it joins to its parent. If you had a human figure set up as a cycle object, to move the upper arm around the shoulder you would use Pivot to rotate the link to its new orientation. Just like twist and move, you just hold the mouse button down as you drag the mouse in any window.

You can pivot, twist, and move any link, but most of the time "Move" is used to set link sizes, Twist is used to spin objects, and Pivot is used to position objects without changing the object's size. In some motions, using both pivot and twist is appropriate, such as modeling a wrist. A wrist can bend (pivot), but it can also rotate (twist), and it can even do both at the same time. The main danger to look out for is in using Move, since it can easily change the lengths of links. If you are looking for realistic motion, don't use Move unless you are setting up or adjusting the link sizes, or when you *want* your objects to change size for a cartoony feeling.

When you twist, pivot, or move a link, the parent of the link is completely unaffected. But what happens to the children of the modified link? Since they must stay connected to their parent (the link you are manipulating) they have to move as well. It turns out that there are two different ways which the children can react, depending on a user-selectable mode. The default reaction is to have the children follow as a group whatever changes are applied to their parent. The children will not change size, but they will rotate and move to "stay with" their parent. If you twist a link, its children *as a group* will rotate around with the parent. A twisted wrist will make the hand flip over, with the thumb still staying attached to the palm in the proper place. A pivot will keep all of a link's children together, not left behind to float in space. This is almost always exactly the response you want since it allows you to raise an arm with a one single pivot instead of moving the upper arm, then forearm, then the hand into the proper position.

This behavior is controlled in the Pick pull-down menu. The default selection from this menu is called "Group," which tells Imagine to keep all of the links together as a group as described before. The alternative, "Object," allows you to manipulate a link independently from its children. If you wanted to spin an object around but keep its children still, you could use the object pick method to twist the link, leaving the children undisturbed. But what about Move and Pivot? Since the direct children of a link *have* to be connected to their parent, they have to respond when their parent's link is moved to another location. What they do in this case is keep their ends anchored, and stretch to stay connected to wherever their parent moves to. *Note that this means the children will change size!* Grandchildren links will be unaffected.

A special note regards the main axis of the object. It, like the links, can be moved, twisted, and pivoted. *These changes are not a real change in your final cycle object.* They merely change your viewpoint of that object in the Cycle editor. Since a cycle object is moved and positioned in the Stage editor by positioning and orienting its axis, the orientation that Stage assigns the object will make your entire cycle object change orientation. If you pivot the main axis in Cycle so that it (and the rest of the object) is upside down, the Stage editor won't care and will just reorient the object it so that it is rightside up. This in no way limits you, however! You can just rotate the object as a whole in the Stage editor, giving you the exact same effect.

Pivot, Twist, and Move give you complete control over the positioning of links, and with Add and Delete you can extend or prune the link skeleton. To actually tell Imagine what objects should be associated with these links you manipulate, you use the "Assign" mode. When in assign mode, you just click on the end of a link, and a requester will appear asking for an object's filename. There are two main restrictions in which type of object you can assign to a link. You cannot use a perfect sphere, so you'll have to use a primitive sphere made up of triangles instead. Most importantly, the object assigned to the link *must be a single object and not a grouped set of objects.* Apparently Imagine doesn't want to have a complex hierarchy of grouped objects assigned to yet another hierarchy, this time of links.

One further restriction of assigned objects: you *can* assign a lone axis to a link. (You might do this to use it as a lamp) but a bug in Imagine 2.0 will not animate this link properly. A workaround to this bug involves adding a few points (but no faces) to the axis before you assign it. With no faces, the axis will not be visibly rendered, but the Cycle editor won't choke the axis' motion.

Remember the way an object associates itself with a link; the object will be placed so that its axis will be at the end of the link, with the objects Z axis pointing towards (and ending on) the parent of the link. The object will be scaled so that the object's axis will be the same length as the link. If the link changes size, so does the object that is associated with it. (Yes, this has been said a few times, but it is crucial to understand this!)

Even the axis of the cycle object can be associated with a real object. This usually isn't a good idea, because you can't change the size, position, or orientation of the axis without affecting the entire cycle object. If you want a single object to be the parent of the object, just have one link be the direct child of the axis, and assign your object to this link. Any other links should be connected to this link or their children. Since the top link is the only one connected to the axis, it lies above all other links. The advantage over just assigning an object to the axis is that the top link can be moved, pivoted and twisted and the changes you make will actually register when you animate the object.

Links do not have to be associated with a real object. By just leaving them unassigned, they can be used as spacers sometimes called "null links." These null links are especially useful if you want to move a link so that it is not directly connected to its parent. One example might be a helicopter where the parent link is the main body and two children are the main rotor and tail rotor. If the axis of the body is directly below the main rotor, the main rotor's link will be pointing straight up. This makes animating the main rotor link easy, since you can just twist it to make the rotor spin. But what about the tail rotor? If you have a link from the center of the body to the tail, a twist will make the tail rotor rotate *sideways*, with the rotor moving from the right side of the tail, to the top, to the left, then underneath. If we use pivot, the rotor will turn in the right direction, but around the wrong axis! It will actually rotate around the top of its link where it joins its parent at the center of the helicopter. If animated, the tail rotor would fly into the air, *over* the main rotor, then proceed to move in front of the body, then underneath! Again, this is the wrong solution.

This is a problem where null links are a perfect solution. We want to represent the tail rotor as a short link that lies flat, perpendicular to the helicopter body, so that twisting it makes a rotation around the small rotor's axle. By adding a null link from the body straight back to a position at the tail, we can use this new link as a parent for the small rotor. Now it is trivial to animate the tail rotor properly since "twist" will make the link rotate the way we want. The null link is completely invisible when rendered, so our object's visible geometry won't change at all.

A final Cycle mode is "Deassign" which allows you to remove the association of a link with an object. This is mostly used to convert links to null links, though sometimes it is useful to reduce the clutter in the perspective view. After you've manipulated the links, you can re-assign the links with the proper object by using the "Assign" mode again.

These tools are enough to create and position the links of a cycle object of any complexity. However, these commands aren't the only way to set up cycle objects. Certain tools in the Detail editor can be used in conjunction with the Cycle manipulation tools, or as a complete replacement. These alternatives are discussed in length in the *Pose Commands in Detail* section on page 83.

Some fairly obvious commands in the Object menu let you load, save, and begin new cycle objects. A unique method of saving objects is also given by the "Snapshot" command from the Cell menu. Snapshot allows you to save a normal grouped object (not a cycle object) of the object that the current link structure and object assignment that is represented by the pose as shown in the current frame. (Eeek! Think of it as saving the object that is shown in the perspective view.) This command is most useful to make special effects, since you can manipulate this new object in the Detail editor. These Snapshot objects generally are not directly re-loadable into your cycle again after being manipulated.

5.3 Animating Cycles

Setting up a hierarchy of links to represent an object is a lot of work, but there are terrific benefits. The most important is that you can now quickly and easily reposition links to form new configurations and poses of your object. If you specify two or more of these poses, Imagine is intelligent enough to be able to figure out how to smoothly change the object displayed in one pose into that of another. This is the basis of how Imagine animates cycle objects.

Imagine has a very straightforward method of letting you specify the way an object should change over time. You can specify "key" poses at different times, which explicitly state how the model should be positioned at those specific times. You can have as many of these keyframe poses as you like. Between these keyframes, Imagine interpolates the position and orientation of the links to smoothly change one pose into another. These keyframes form "snapshots in time" which are enough for Imagine to figure what sort of motion you are looking for.

One classic example of this would be a walking figure. The first keyframe would show the start of a stride, with the left leg straight down, the right with the knee slightly bent and foot off of the ground. The next keyframe would show the right leg straightened out, with the foot firmly on the ground in front of the torso, the left leg on the ground behind the torso with the knee starting to bend to lift the foot off of the ground. The third keyframe would be the same as the first with the left and right roles reversed; the fourth keyframe would show the legs opposite to the second keyframe, and the final keyframe would be identical to the first, to bring the walker back to its original state. These four (or five) different poses are enough information to figure out what configuration they take in the inbetween frames. If Imagine does a bad job with a particular transition, you can always insert a new keyframe midway between the other keyframes and explicitly define the "midway" point to show more subtle detail.

Notice how the status line at the top of the Cycle editor's screen contains the a title like "Frame: 0*" This is important information since the status line tells you which frame you are looking at (0 in this case, the very first frame) and whether it is a keyframe or an interpolated frame (the * indicates that this frame is a keyframe.) Cycles can be as many frames long as you wish. You can choose to specify any frame as a keyframe, with the unspecified frames being interpolated from the keys.

Page 102 talks about the considerable control Stage gives you over the way cycles are played back.

In fact, the total number of frames you choose is almost unimportant as the Stage Editor can automatically use the keyframes to build a cycle moving smoothly at whatever frame rate you wish. If you have a four frame walking cycle with every frame being a key, you could still load the cycle into the Stage editor and animate it using four *hundred* frames to take those two steps and every frame would be different. Conversely, if you made a 400 frame cycle with 20 keyframes, you could render the walk as taking only 10 frames to complete, and Imagine would figure out the 10 best frames that represent the 400 frame cycle.

Frame 0 is somewhat special since it is the first frame in your cycle, and is always a keyframe. New links can only be added and deleted from this first frame, and it is also the only frame where objects can be assigned or deassigned to links. This is to prevent links from appearing or disappearing in the middle of a cycle animation. Non-zero frames still give you complete control over the position and orientation of all of the links. For some reason Imagine starts labeling frames with Frame #0 in Cycle, but with Frame #1 in Stage, so be careful or you might get confused.

One of the most common motions to animate in the Cycle is a simple rotation. You might want to have a spinning propeller, or make the hands of a clock rotate. The obvious way to do this is to build one keyframe with the hands of the clock (or propeller) pointing at 12 o'clock, and a second keyframe with the hands at 6. If the hands move from 12 to 6, then from 6 to 12, you should get a nice smooth rotation. If you actually try this, you might run into a problem that sometimes occurs with this 360 degree rotation. The problem is that Imagine has no idea that these are clock hands and are supposed to rotate in a very specific direction. From the beginning and end positions that you've defined, it knows that the hands *should* rotate, but which way? Imagine does its best, and uses some magic algorithm in these cases. Unfortunately, half of the time the magic trick fails, and links rotate the wrong way. There is a simple solution to this problem, however. If you use *three* keyframes (instead of two) with the hands at 12, 4, and 8 o'clock, the direction of rotation is plain. You could also use four keyframes, or even twelve, but three is sufficient to tell Imagine the which way to rotate the links.

The definition of keyframes is fairly easy. Most of the controls are menu items from the Cell menu. If you use "Goto," you'll be asked for a frame number to visit. If this frame is higher than the number of frames you've already defined, the total number of frames is extended and the new frame is made a keyframe. Both the first and last frames are *always* keys, so Imagine knows what configuration to start and finally end with.

Using the "Goto" command, you can visit any frame you wish and manipulate the links to make new object poses. *If you make any changes to a frame which is not a keyframe (no * in the status line), those changes won't be saved.* If you want to set up a pose in a frame that is not a keyframe, you can use the Cell menu command "Make Key" to make that frame a key. Conversely, you can turn a keyframe back into a normal interpolated frame by using "Unmake Key."

In each of these keyframes, you can use the standard "Twist," "Pivot," and "Move" modes to edit the pose of your cycle. *To avoid confusion, never move or manipulate the main axis of the object if you can help it.* Remember that you can twist and rotate the entire object at once by manipulating the main axis of the object, but these actions don't affect the actual cycle when it is animated in Stage. If you flip the entire object upside down in a keyframe, you can make what appears to be a cycle showing your object doing a somersault. But since the axis position and orientation doesn't actually get saved with the cycle object, when you try to animate your object in the Stage editor, no somersault will happen. As long as you don't manipulate the main axis, you'll avoid confusing yourself. There *are* advanced occasions where manipulation is useful, mainly when you want to want to see how the motion provided by a path in the Stage editor and the internal animation in cycle will combine.

There are quite a few other Cell menu commands that make it easy to travel from keyframe to keyframe. "First" brings you to frame 0, and "Last" brings you to the last frame of the cycle. "Next" moves you to the next frame (keyframe or not), and "Prev" brings you to the previous frame. "Next Key" will advance you to the next keyframe, and "Prev Key" to the previous keyframe. All six of these commands are just alternatives to using "Goto" and explicitly stating the frame you want to manipulate.

A simple use of "Goto" with a high frame number can extend the number of frames in your cycle. To reduce the number of frames, you can go to the first or last frame and use the option "Remove" from the Cell menu, which will remove that frame from your cycle. In these cases, the second or second to last frame will become the first or last frame respectively.

When building new keyframes, a useful tool is the "Copy From" command. This allows you to copy the pose of your object from any other keyframe. This is especially useful for the last cell in your animation. If you want to make a true looping cycle object, you want the object to return to the same position it started out from so when the loop brings the object back to the first frame there isn't a sudden jump in object configuration.

There is actually one subtle problem with this idea, however. If you set the last frame to be the same as the first, during the looping animation there will be two frames in a row that are identical. If the rest of the frames show a smooth movement (like the hands of a clock) there will actually be a an apparent pause in the motion as the duplicate frame is played. What we really want is for our cycle to end one frame earlier, such that the last frame is the position the hands (or whatever) would obtain just before reaching the position that the clock started out on. If we have a 4 frame cycle of a clock's hands, we want the hands to move from 12 to 3 to 6 to 9, not 12 to 4 to 8 to 12.

There are several ways around this looping problem. You could use the Stage Editor to only play part of the cycle, and leave off the last frame. A better alternative is the following: make your cycle one frame longer than you really want. Use the "Copy From" command to copy the first frame's pose to the last frame. Now, move to the next-to-last frame, which now has an interpolated pose with the object in the "almost back to where it started from" pose. Make this second-to-last frame a keyframe. Now go to the last frame and use "Remove." This cycle will now repeat smoothly. There is a danger, however! If you have a 12 frame cycle of a clock, use this trick to make it smoothly repeat, then stretch the playback of the cycle in the Stage editor to repeat twice over 144 frames, there will be a *huge* jump at frame 73 as the clock's hands move from 11 to 12, instead of 11:59 to 12:00 as it should. If you use this trick, try to keep the number of cycle frames close to the number of true frames. This doesn't mean you need 72 keyframes, just use a *lot* of interpolated frames.

In order to help view your animated figure, there is a method of previewing your cycle in motion. The Animate menu gives you the ability to make a small, quick animation of your cycle. The "Make" option will render every frame in your cycle as shown in the perspective view (which might take a while, depending on the number of frames you have.) Once made, you can use the "Play Once" or "Play Loop" menu items to view the animation. A mini-control panel will appear which allows you to set the playback speed, step through the frames one by one, reverse the motion, and start from the first frame again. "Play Once" will stop the playback at the end of each animation, whereas the "Play Loop" will automatically wrap from the last frame back to the first. "Play Big" will use a fullscreen window for the animation playback, though the frame rate might slow down. A final option, "Free RAM," will drop the animation from memory when you're done with it.

Morphing is described on page 103.

One last advanced idea. Imagine supports morphing, which can change one object's colors and shape smoothly into another's over a number of animated frames. This morphing also works with cycles, as long as the cycles have the same number of links in the same parent-child relationship, and the same number of keyframes in the same positions. With this morphing ability, you can have your walking man slowly change colors or whatever as he walks. Since the individual objects morph, you could even gradually change your walking man into a gorilla, a robot, or a snowcone.

Even more exciting, when you morph two cycles, you can also smoothly morph the way the cycle behaves. You could have a cycle of a man running and another cycle of the same man walking. Running and walking have considerably different strides, and therefore their cycle animations would look noticeably different. If you want to animate a walking man who then starts running, you can use the morphing features to change the walk cycle gradually to a running cycle over the period of three or four strides. This transition would be beautifully smooth and seamless. During this transition, the individual objects could even morph their shape. When you think about a running man smoothly morphing into a wing flapping bird, you truly see the power of Imagine.

Morphing animal walks is illustrated on page 150.

5.4 Pose Commands in Detail

Both the initial setup of a complex object and the later positioning of keyframes can be very difficult, especially since you cannot see the objects that each link represents except in the perspective view. Also, setting up the links to begin with can be arduous, especially keeping the relative scales of objects correct. For this reason there are several alternative methods of creating cycles. These alternatives can work as a replacement of the "pivot, twist, move" controls of the Cycle editor, and even better, can work in combination with them. The biggest advantage of using the Detail editor over the Cycle editor to set up cycles is that you can see the actual points and faces of the objects the links represent. The abstraction of the links goes away (this is both a good and bad thing,) and more accurate control can result.

There are two ways that the Detail editor can help Cycle produce its cycle objects. The first is in initially setting up the cycle's link structure and object assignments. If you have a grouped object in the Detail editor, the structure is already defined; Imagine knows what the parts are, where they should be placed, and the objects' parent/child relationships. There are two steps involved in using a group to produce the cycle structure. The determination of the links is very easy; the grouping hierarchy directly defines the way the links are set up. The line that groups a child object to its parent is exactly the right length and position to specify the link for that child object. Imagine has no problem creating a link for every object in a hierarchy.

The second step is the tricky one. *Every object must have its axis realigned and scaled.* Remember that an object assigned to a link always has its Z axis aligned with the link. If we want to make a grouped object into a cycle structure, we have to rotate each object's axis so that it is pointing at its parent, and scale that axis (without scaling the object) so that the Z axis ends directly at the position of the axis of the parent of the original object. If the links are defined by the grouping lines, then this axis twiddling will make the object appear at the right size and orientation when assigned to its link.

Imagine can perform this entire process automatically. The option "Cycle Setup" in the Detail editor's Functions menu performs all of these operations. The structure of your object after using this feature will be unchanged, but the object axes will be modified so the objects will be in the proper orientation in the Cycle editor.

How do you get this reconfigured object into Cycle? Very easily! You can simply use "Load" in the Cycle editor, and enter the filename of a grouped object instead of a cycle object. The Cycle object will ask you if you want to convert the object into a cycle object. In fact, even if you haven't used "Cycle Setup" already, the process will be done for you anyway during the loading process. This simple conversion takes a lot of the drudgery out of the creation of the basic cycle structure.

Be careful using the conversion, though! The cycle structures that are produced are always perfect copies of the original object as it was shown in the Detail editor, but *these structures are not always defined properly to animate easily!* Through bad placement of axes, you might have arms that don't bend at the elbow, but pivot around the center of the forearm. Through bad planning, you might have grouped the fingers as children of the thumb, and not the hand, so every time you move the thumb around, the fingers fly into the air and float like levitating pink sausages. Your helicopter's tail rotor isn't going to rotate well unless you use the null link trick, and if it was simply grouped to its parent, the main body, you're not going to get it to spin the way you want.

The commands that manipulate object axes independently are discussed on page 13

It takes planning to set up every object with its axis placed where you want its children to move around and to make sure that objects are grouped to the appropriate parent. Luckily it is easy to move the axes around in the Detail editor to set up the proper positioning (by using shift-M to move, shift-R to Rotate, and shift-S to scale,) and fairly easy to change groupings to build a proper hierarchy. Use a simple, plain axis in a group if you want to make the converted cycle structure to have a null link. By putting an axis at the tail of the helicopter and grouping it to the main body, then grouping the tail rotor to this new axis, you'll get the right null link structure for animation. The pivot centering problem with the arm can be solved by moving the axis of the upper arm to where the elbow should be, and not at the center of the arm. Linking the fingers to the palm instead if the thumb is done just by ungrouping the fingers, then regrouping them to the palm. With a little experience, it is easy to get your groups set up the way you need them.

If the Cycle editor's "Load" command can convert the grouped objects, then why is there a "Cycle Setup" command? The answer has to do with the ability for the Cycle editor to load poses of objects that were made from the Detail editor as keyframes (this is that second Detail function that I mentioned a few pages ago and left dangling.) You can manipulate that grouped object, save several poses of it, then load those poses into Cycle as keyframes. You don't have to use those twist and pivot commands! However, this ability has some restrictions since you can perform many operations in Detail that Cycle can't duplicate. The "Cycle Setup," "Cycle Shuffle," and "Cycle Transforms" commands allow you insure that your manipulations are "legal."

There are a few steps to follow if you want to use Detail to define your poses. First use "'Cycle Setup" to reorient the axes of the objects; this will insure they'll appear undistorted when you load them into Cycle. Once this preliminary step is done, you might want to save your object (using the normal "Save" command in the Object menu) if you want to use the object (as is) as one of the poses.

A "pose" is a configuration of your object that you want to use in a keyframe. To build a new pose, the idea is to use the Detail editor's object manipulation commands to put your grouped object into its new shape. The big danger is a lot of the manipulations can easily violate some of the restrictions placed on the way cycles can change their configuration. *Picking an object and simply moving it to a new position just isn't allowed!* Why? Because all objects must have the end of their Z axis always at the axis of their parent. If you move an object to a new position, you're also moving the end of that axis, which is not allowed in cycle objects. Even if you simply rotate the object, the Z axis will rotate off of the parent since the object rotates around the center of the axis. Here is where the command "Cycle Shuffle" becomes useful. If you rotate and move the objects around, this command will keep all of the scaling and orientation changes you've made, but it will move the objects back to a position so their Z axis ends on their parent like they are supposed to. If you pivot an upper arm (by using the normal rotate commands) so that it sticks straight out, it will actually pivot around the elbow (that's where the axis is) and the arm will actually detach itself from the body since it's rotating around its axis which *isn't* at the shoulder. When you use "Cycle Shuffle," the objects will move so they are reattached to their parents the way they are supposed to. Only after you use "Cycle Shuffle" are you assured that your object is in a "legal" pose that can be loaded into the Cycle editor as a keyframe. If you save a pose without using this cleanup command you won't get any errors, but you could get weird results when Cycle "fixes" your mistake for you in a way you didn't expect.

The big problem is that these rotations are hard to judge since the individual object axes seem to be in the worst possible places for manipulations. Every rotation moves the Z axis off of its parent, and judging what effect a certain rotation will have after it's been fixed with "Cycle Shuffle" is not pleasant. For this reason, there is an option called "Cycle Transform," also in the Functions menu. This option acts almost like a mode, as it can be turned on and off at will. A check mark will appear next to the menu option so you know that it is on or off.

What this function allows you to do is to rotate an object not around the center of its axis, but *around the end of its Z axis.* When you have this option turned on, rotating a forearm will make it pivot around the end connected to the shoulder, the way it is "supposed" to in Cycle. If you are setting up objects for poses, you'll probably want this option turned on most of the time. "Cycle Transform" only changes the center for interactive rotations; the numerical "Transformation" requester still uses the center of the axis.

When you've placed your object into a new pose, you should probably use "Cycle Shuffle" to make sure that your pose is legal, then you can just use the normal "Save" command from the Object menu. Give your poses different names so they don't overwrite each other! You might want to use number extensions to the name, or mini-descriptions of the pose, like *clocknoon.iob* or *clock3pm.iob.*

To load one of these poses as a keyframe in Cycle, first make sure you've already used the Cycle "Load" command to convert a version of the grouped object into a cycle structure. Goto (or make) to a keyframe, then use the simple command "Load Pose" from the Cell menu. The pose you defined in Detail will now be used to move the cycle object's links to the right configuration.

The possibilities of Cycle are incredible! This editor is far underutilized by beginning Imagine users, mostly because it is not a necessary step to build and render objects. But if you are an animator at heart, this editor will give you the ability to make your objects truly perform the way you want them to. Defining the appearance of an object is only half of the work; making it come alive is the true animator's challenge.

Chapter 6

The Stage Editor

The Stage editor allows you to load previously created objects and arrange them in a scene to be rendered. Just like a movie director, you have complete control over the actors (your objects,) the camera, and lighting of the scene. If you are making an animation, you control the way that all of these change in time.

The Stage editor is very closely linked to the Action editor; in Imagine 0.9–1.1 the Action editor was even called up from Stage as a subeditor. The Stage Editor allows you to move and place objects, lights, and the camera, whereas the Action editor allows refinement of those placements, as well as defining complex extra features like morphing and special effects. Because of the close association between the editors there are a lot of cross references between these two chapters. If you are confused about something, you might try skimming through *both* chapters to look for an explanation of whatever you need.

Every project you make or open from the Project editor has a "staging" file which tells Imagine how objects, lights, and your camera are arranged in your scene or animation. You can think of both the Stage and Action editors as being ways of editing this staging file. When you enter the Stage, this layout file is loaded and you can edit your scene and save the changes. For a new project, a very simple world consisting of a lone camera floating in empty space (no lights or objects) is your starting point.

After you've used the Stage editor to make any changes to your scene, *you must save your work* or Imagine won't think you want to keep any changes you might have made. This is done by using the "Save Changes" option from the Project menu in the Stage editor before you quit. Luckily Imagine will also put up a requester to make sure you don't quit by accident and lose any changes you might have made.

6.1 Stage Layout

The basic operation of the Stage editor is very simple; you have a world in which you can place and manipulate objects. You can load as many of these objects as you like (well, until you run out of RAM) and manipulate them, placing them into the world at any position, size and orientation.

Loading an object into the world is performed by simply using the "Load" command from the Object menu. The object that you load can be one saved from Detail, Forms, or even Cycle. The object will load and will appear in the Stage at the same location and orientation it was saved in from Detail. Once in your world, you can manipulate the object(s) by picking them and using the standard move, rotate, and scale commands that you use in the Detail editor. You can also use the "Transformation" requester from the Object menu to manipulate your object with exact measurements instead of the interactive, but imprecise, mouse movement commands.

The movement and transformation commands are discussed on page 10.

The command "Delete" will remove objects from your world.

By using the customary manipulation commands, you can easily set up the objects in your world. A simple scene might have you load in a flat plane you built, positioning it horizontally in your world to represent the floor of a room. A table could be loaded and moved to a position where its legs are just touching the floor plane. A flowerpot can be loaded and moved to rest on the table. The basic layout, especially of a non-animated scene, is really quite straightforward. Load the objects you've built and position them in the world so they are in the configuration you want. You can move around your world, add background gridlines, zoom in and out: all of the standard world viewing commands. Your preview window will show you what a perspective view of your scene looks like, and you can change the viewing angle and position by using the sliders in the window just as in the other editors.

These standard viewing commands are discussed on page 8.

One convenience is that you can load more than one copy of a single object into the scene and manipulate the copies independently. You might have a single tree object, but if you load 15 copies of it and place them in a disorganized array, you can build a forest out of just copies of this one object. You can even rotate and scale the copies so that it isn't obvious that each tree is identical. When you load more than one copy of an object which is called *tree,* Imagine will name the objects *tree, tree.1, tree.2,* and so on. Unfortunately Imagine uses 10 times as much RAM to represent 10 copies of one object. It doesn't realize that it's using identical copies when rendering.

If you want to rename multiple copies of an object so you can quickly identify them, you can use the "Rename" command in the Object menu to give each copy its own unique name. This allows you to use the "Find by Name" and "Find Requester" more easily. In the Action editor, each object is clearly labeled with its name so identifying copies is considerably easier if you use names for duplicate objects that describe them like *lawntree, bigmaple,* and *distanttree* than just *tree.33.*

The two "find" commands act exactly as they do in the Detail editor, as described on page 16.

When you are manipulating one or more objects, sometimes the redraw rates of your three main views can become painful. Luckily, Imagine allows you to automatically "Quickdraw" any or all of the objects in the world. "Quickdraw All," "Quickdraw Pick," and "Quickdraw None" act exactly as their counterparts in the Detail editor, allowing for much faster updates of complex objects at the expense of losing the exact wireframe view of those objects.

Quickdraw of objects is discussed on page 14.

6.2 Camera

Setting up the object positions is only half of the work of scene design. You want to define exactly what view of this world will be rendered: do you want a picture of your car driving down a road as seen from the front of the car? The back? From inside? The way to define the viewpoint from which your world is rendered is by manipulating the camera.

The camera is an object in your world, and it can be picked and rotated just like normal objects. The camera always exists; you cannot delete it or add a second one. In the three main views, the camera is shown as two concentric circles with a short line sticking out of the center. The center of the circles represents the camera position, and the line points in the direction the camera is facing. When you pick the camera and rotate it, the line will rotate to reflect the new orientation of the camera. A "crossbar" is shown as a diameter passing through the center to indicate the tilt of the camera, so you can tell if you are holding the camera at an angle or on its side. *The camera will never show up in the perspective view.* It is a "virtual" object with no real existence. It tells Imagine what viewpoint to use to render your world from; it is not an object to be rendered itself.

A camera sees only objects that are in front of it in the direction it is pointed. Just like a real camera, you want to position the camera to get a good view of your scene. An essential tool in placing your camera is the "Camera View" option from the View menu. This option changes your perspective window from a manually defined, arbitrary view of your world to a true view of your world *as seen by the camera.* The camera position and orientation completely define the view, so the sliders to control the viewing angle are disabled in this mode, and zooms and pans in your main three views also leave the perspective (camera) view unchanged.

Setting up your camera position and orientation is pretty easy since you have direct feedback from the preview window of what the camera sees. While the camera view will update every time there is a change in object or camera configurations, it unfortunately does not update *during* the movement or rotation, so you can't use the display to judge exactly how far to move your object or camera.

The camera view will give a very accurate preview of what your final rendering will show if you saved your scene in its current state. Using the "Quickrender" option from the Editor menu is also very convenient to get a quick idea of what the render will look like with all of the objects properly shaded and colored.

Quickrender is discussed in the Common Editor Tools section on page 17.

When you are manipulating the camera, use of the camera's local coordinates can be very convenient. By setting "local" mode with the keyboard command "l" or the "Local" gadget on the screen, it is easy to move the camera straight forward and back to get closer or further from the scene you are looking at by moving in the Y direction. (You use shift-Y to restrict the camera to move straight in or out, just like you would restrict normal objects to move along their Y axes.)

Local coordinates are described on page 12.

Some other camera manipulations are also easier in local mode, since rotating the camera in world coordinates can twist the camera in ways you didn't expect. If you want to pan the camera up or down, use local X rotation, and you'll be able to raise and lower the angle the camera is pointing at without twisting it. Local Z rotation will let you pan left or right, and global Z rotation will rotate the camera without changing the up/down angle that the camera is pointing.

Only rarely do you want to twist the camera at an angle, since everything will obviously look tilted. If you twist the camera like this (by accident, or by using world X or Y rotation), it is usually easiest to use the "Transformation" command in the Object menu and set the Y alignment (in *local* coordinates) to 0, which will level your camera.

If these camera rotations seem confusing, you can always get acceptable results by always rotating the camera in the local X or Z directions, and never using world coordinates or local Y rotation.

Actually, many times you never have to manually point the camera; there are ways to get the camera to "track," making it automatically rotate to point at an object. The command to do this is called "Camera (Re)track" which is found in the Object menu. When you execute this command, a requester will pop up asking for an object name; you just type in the name of the object you want the camera to look at. (This is the name you might have already set, like *bigmaple* in the previous example.) The camera will look directly at the *axis* of the object you tell it to track. You can use "Find Requester" to get a list of the names of your current objects. This tracking is a one-time aiming of your camera; the angle will not update if you move the camera or the object. In this case, you can use "Camera (Re)track" again to tell the camera to point at it (or another) object again.

You can make the camera permanently point at an object from the Action Editor; this is discussed on page 110. When this is done, the "Camera (Re)track" command will not ask for an object to point at but will immediately "refresh" its direction and point at the object you've assigned the camera to track each time you select the command.

Often you don't want the camera to point at an object, but just at a point in space. Imagine allows you to do this by adding an axis to the world that can be renamed and moved just like a normal object, but does not show up when rendered. If you use "Add Axis" from the Object menu, an axis with the default name of *track* will be added to the world. You can move this track axis anywhere you like, and you can use the "Camera (Re)track" command to point the camera to the location indicated by this track axis by telling the camera to track the object named *track*.

6.3 Lighting

There is a critical element that has to be added to every scene before it is rendered: lights! When Imagine renders a scene, it *has* to know what sort of lights are in the scene and where they are positioned. A scene with no lights would render as all black. Some lights can be built into objects, but usually the lights are added in the Stage editor.

Directly incorporating lights into objects is discussed on page 33.

Lights are shown in the three main views, and are represented by a circle with a dot in the middle. You can add as many or as few lights to a scene as you care to, but almost always you want at least one. To add a new light, select "Add" from the Object menu, and "Light" from the submenu. The light will appear at the origin of the world; you can pick it and use the standard movement commands to place it wherever you like.

Like the camera, the lights will not show up in the perspective view. Lights are invisible; you only see the *effects* of the light source and not the light itself. Even if the light is in the view of the camera, the actual render will *not* show a spot where the light source is. If you want to be able to see where the light is coming from, build a lightbulb object, a transparent sphere, or some other physical object and place it on or near where you put your light source. Even better, build an object with the light incorporated into it.

You have a lot of control over lightsources; you can change their color and intensity, whether they cast shadows, if they dim with distance, and whether the light is "spherical" or casts light only in one direction like a spotlight. These options are all controlled from the Action editor. Usually you place a light in the scene where you want it, and later diddle with its attributes in the Action editor.

Definition of these light parameters is described on page 106.

If the light is a directional spotlight (again, defined from Action) the light will have a small line pointing out of it that indicates the direction the light is facing. You can use the rotation commands to point the light in whatever direction you wish, and you can even track a light to an object just like you can with a camera.

Light design is a difficult and important problem. There is a short discussion of lighting setup in the Appendix on page 142.

6.4 Animation

Setting up a basic scene really is pretty simple; load your objects, arrange them, point the camera, add a couple of lights, save the scene, then render. But there is a special magic to animation, where your scene becomes alive and the viewer gets to experience your virtual world in motion instead of in just a simple still frame.

Animation has its own set of special set of controls and functions. To build a one minute animation of 1800 frames, you luckily do *not* have to visit each frame and place the objects and camera where you want them do be for that specific frame (though you can if you want to!)

Animation requires the use of the Action editor at least once to define the length of the animation is defined. Imagine defaults to a one frame animation, which is just a single still picture. After you define a multiple frame animation, several commands in the Stage become useful in defining the way your scene changes in time. Perhaps the most apparent change after you tell Imagine that you are making an animation is that when you start the Stage editor it will now immediately ask which frame you would like to go to. You can view any frame of an animation from the Stage and manipulate the objects in the world at those times. (Much more on this later!)

The definition of animation length is described on page 97.

Imagine has several tools that help define motion. You can have objects follow set paths in time, or you can "tween" positions where you define a starting frame and position, and the ending frame and final position, and Imagine will automatically generate the inbetween positions. Similar tools can define scaling and rotations as well. Cycle objects can create complex model motion.

Cycle objects are really controlled from the Action Editor. See page 102 for a complete description.

When you have a multi-frame animation, you can move to specific frames by using the commands from the Frame menu. These commands are fairly straightforward. "First" will bring you to the first frame, "Last" will bring you to the last frame. "Next" will advance to the next frame, and "Prev" will back up to the previous frame. The command "Goto" will let you enter a specific frame number to visit.

6.4.1 Keys

When a scene is animated, the actors (objects, camera, even the lights) can change size, position, orientation, and even identity. Most of the time there is a set of objects that never move in your world, like trees, and another set of objects that do move, like cars traveling down a road (perhaps aimed at the tree?) The camera can also move and rotate during the course of the animation.

The most common change objects undergo in time is to move from one position to the next. It would be an evil task to have to manually position every object in every frame, but luckily Imagine allows you to quickly define simple straight line motion. One method of defining this motion (and object orientation and scaling) uses a key technique much like how the keyframes in Cycle told Imagine how the cycle object changed configuration in time. By defining the starting position of an object at one frame and the ending position of the object in a later frame, Imagine can easily figure out where the object should be placed in the inbetween frames. This can also be done to define the size and orientation of any object in time.

Instead of using keyframes like the Cycle editor, Imagine lets you set up keys in a more versatile way. Each object can have its own keyframes that start and stop in any frame; other objects do not necessarily have to have the same key structure. For example, you could tell Imagine to position a car at one location in frame one, and another location in frame 100. Between those keyframes, Imagine will interpolate the car's position so it smoothly moves to its end position. You could then add a second car that appears at frame 30, passes the first car, and by frame 80 it is out of view. The objects don't have to have their keys in the same frames; each object has independent keys.

In fact, the position of an object can be keyed completely separately from that object's other size or alignment keys. For example, if you want an airplane to move from one position in one frame to another position in a second frame, you could set up the position keys just like you did for the car. The size and alignment of the airplane never change, since you haven't added keys for them. If you want the plane to do a barrel-roll over a couple of frames in the middle of the airplane's flight path, you could add new alignment keys for the frames in the middle, perhaps saying "The airplane is rightsideup in frame 10," "The airplane is upsidedown in frame 15," and "The airplane is rightsideup in frame 20." These alignment keys would be in addition to the position keys that say "The airplane is at this location in frame 1," and "The airplane is in this second position in frame 40." The position keys are totally separate from the alignment keys, so these five keys together will produce the effect you want. With standard keyframes, you'd have to set the position, orientation, and scaling in *all five* keyframes, which is much more work. Imagine's method is much more versatile, though it is more confusing.

If you like the idea of strict keyframes like cycle objects use, you can just make a position, alignment, and scaling key *every* time you want to define a key for any of the three. This will give you the same results as the standard keyframe technique.

Keys are talked about in the Action Editor; the best way to understand them is to visualize them in the timeline display the Action editor uses. If you understand the way the keys work after reading about them in Action, you can use certain tools in the Stage to define them quickly and easily.

Keys can be defined from the Stage editor by a simple menu command. Go to any frame, and manipulate the object you want to build a key for. To make the object's current position as a key in the current frame, use the command "Position Bar" from the Object menu. Similarly, you can make alignment and scaling keys by using "Alignment Bar" and "Size Bar." These commands will add a new bar to the object's position, alignment, or size timeline in the Action editor. By executing two or more of the commands, you can key (for example) both position and alignment or even the complete definition of position, alignment, and scale.

Check out page 100 for information on how keys are viewed and edited in Action!

The exact effect that the "Bar" commands have on your keyframes is described on page 119.

For example, say we want to make the airplane that travels in a straight line with a barrel roll for a few frames in the middle. You would go to frame 1, load the plane, and position the plane where you wanted it. You would then select "Position Bar" to set the start position. Then you could use the "Goto" command to go to frame 40, position the plane in its new location, and use "Position Bar" again. If you previewed the animation, you would see the plane smoothly move from its starting position to the ending position in frame 40. To add the barrel roll, you could go to frame 10 and select "Alignment Bar" to tell Imagine that the plane is still rightsideup at that frame. After going to frame 15, you turn the plane upsidedown and hit "Alignment Bar" again, then finally go to frame 20, orient the plane level again, then hit "Alignment Bar" for a final time. These three key alignments will tell Imagine "From frame 10, start with the plane level, then smoothly turn it upsidedown by frame 15, then turn it back level by frame 20." This will make your barrel roll. *Note that the barrel roll won't interfere with the smooth movement of the plane moving from its start and stop positions!* Different types of keys do not interfere with the other keys on the object, so the position keys are completely independent from the alignment keys, which are completely independent from the size keys.

Changing an existing key is simple: you just move to the frame where the key is defined and manipulate the object to its new position, orientation, or size. If you change the object's position on a frame defined as a key position, the old key position will be replaced with the new one. Similarly, alignment and size keys can be changed by simple manipulation of the object in the frame where the alignment or size key is defined.

This process is quick and easy except for one thing; it is difficult to identify the frames where you have already defined keys. There are no visual cues in the Stage editor to tell you which objects have been keyed in the current frame, or to show what frames an object has keys in. Unless you remember that you started your barrel roll in frame 10, changing the alignment key might be difficult, since the only way to update that key from the Stage editor is to visit the proper frame. *The only way to really see what keys an object has defining its motion is to go to the Action editor!* Usually when you want to make changes to the keys of an object it is easiest just use the Action editor, which is *built* for viewing and changing the keys.

Another important point: if you have, for example, an airplane moving between two key positions, you can't arbitrarily manipulate the airplane's position in one of the inbetween frames. If you go to one of the not-a-keyframe interpolated frames, you can easily move the object, *but even if you use "Save Changes" your movement will not be saved!* Imagine says "Oh, you have keys at the start and end, I know where the object *should* be." You are allowed to manipulate objects because you might be setting up a new key, but unless you actually make that key, *the changes don't mean anything and are not saved when you go to another frame or leave Stage.* A *very* common and confusing problem results when new users can't figure out why their objects keep resetting their positions.

Pay attention! This is one of the top problems new Imagine users have!

True control of keys is easiest to set up in the Action editor. You can set up keys using the quick "Bar" commands from Stage to get the positions, orientations, and sizes right, then go to the Action editor to better redefine and manipulate the keys.

When you are manipulating objects for keys, sometimes it is very useful to move objects so they are in the same relative position or orientation to an object as they were previously. If you wanted to keep two objects the same distance apart from each other in two different frames, you might use the command "Reset Relative" from the Object menu. This command will let you copy the relative position or orientation of an object from a previous (or future) frame to the current frame.

For example, if you have an animation of a landing Space Shuttle, you might have a couple of "chase planes" that follow the shuttle down. If you initially position a chase plane a little above and behind the shuttle, you might want to keep the plane in that same relative position for most of the animation. You could do this in many ways, but if you are using straight-line keys for the Shuttle's position, you probably want to use keys for the chase plane as well. All you have to do is move the chase plane to the proper position in the first frame, and use the "bar" commands to make that position a key. Then, go to a later frame (probably when the Shuttle has a new key). The chase plane will still be in its old position. If you pick the shuttle, then the plane, then choose the "Reset Relative" command, Imagine will ask you what frame to use. When you enter the frame number where the chase plane's position was initially defined, Imagine will find the relative position of the chase plane to the shuttle in that frame (a little above and behind) then move the chase plane in the *current* frame to be in the same relative position. This also works for orientation, so is you have the shuttle banking or turning, you could have the chase plane bank and turn the same amount.

Although using keys for defining the position of an object is awfully useful, they are somewhat stodgy since objects only move in a straight line between keys. I once made an animation of what I thought would be a bee buzzing around some flowers, but when animated it looked more like a pinball game with this flying ball banging into brightly colored bumpers. The straight line motion looked blatantly artificial. There *are* solutions to this problem, however! In the Action editor, there is an object attribute called "Hinge" that allows you to make curve your paths while still using the keys. A more direct method of making smooth and graceful motion is to use a spline path.

Hinges are talked about on page 106.

6.4.2 Spline Paths

Keys aren't the only way to define the way an object's position or orientation; sometimes you want an object to follow a complex path instead of bouncing from key position to key position along straight lines. Imagine lets you define spline paths for objects to follow.

These paths are exactly the same as the ones used in the Detail Editor. Imagine lets you build and edit the paths in either editor, which is a great boon. There are some subtle differences between the way the two editors edit paths, mostly in the way the path manipulation commands are named. You can add a new open or closed path by using the "Add" command in the Object menu. You can edit a path by picking it and selecting "Edit Path" mode from the Mode menu. In the Edit Path mode, the Path menu becomes active. You can pick a control point, and delete it by selecting "Delete Point" from the Path menu instead of using the simple "Delete" command as in Detail. To add a new control point to the path, you do not use "Fracture" as in Detail, but instead use the "Split Segment" command from the Path menu. To save a path, you use the "Save Path" command from the Path menu instead of Detail's generic "Save." All of the manipulations of spline paths are the same as they are in Detail, with the exception that you *cannot* define paths by positioning axes like you can in Detail with the "Make Path" command. The command "Pick Groups" will exit the "Edit Path" mode, and you can manipulate objects normally again.

See page 48, where spline paths are described in Detail (in detail, too!)

Paths are mostly used in the Stage editor to define the paths for objects to follow. You can pick, scale, and rotate the paths like real objects, including keying the path's position or even having them follow paths of their own. Assigning objects to follow the paths is all described in the Action editor, but the ways you can use paths are virtually unlimited. You could make a circular path for a planet-colored sphere to follow. If you put a bright yellow light emitting sphere in the center, you have a miniature solar system. But you could also have a *second, smaller circular path which follows the first path just like the planet!* You could assign a second, smaller sphere to this second path. Boom! A moon! You could even add a *third* path which follows the second to make a space station which orbits the moon!

Path following is described on page 108.

You have a considerable amount of control over how an object follows a path, all of which is defined in the Action editor. One utility function which is useful is the "Show Path Length" command from the Object menu. This computes the total length of the object in Imagine Units, which can be useful when determining speeds and accelerations of moving objects.

An Imagine Unit is that mysterious undefined length that all coordinates are measured in.

6.4.3 Tracks

One last useful function of the Stage editor is the ability to add plain axes into the scene. Though they aren't rendered, they can be manipulated just like real objects, keyframed, and can even follow paths. The usefulness of these axes are basically as targets. If you want the camera to point at a particular point in space, you could place a plain axis at that point and then tell the camera to point at it with the "Camera (Re)track" command. In fact, when you add an axis to the scene with the "Add" function in the Object menu, the default name of the axis is *track*.

Tracks are also discussed in the object tracking section of the Action editor, on page 110.

6.5 Animation Preview

In order to help view your animated scene, there is a method of previewing your animation right from the Stage editor. The Animate menu gives you the ability to make a small, quick animation of your scene. The "Make" option will let you choose a set of frames to animate. Imagine will ask you for three numbers: the starting frame, the ending frame, and the "step," or how many frames to skip (for quick previews, sometimes seeing every frame just wastes time.) For example, entering "12,22,3" at the requester will make a (very) small animation of frames 12, 15, 18, and 21. The frames rendered will look just like the view you see in the perspective window; it will be rendered with the same viewpoint (camera view or not) and display mode (wireframe or solid.) Once the animation is made, you can use the "Play Once" or "Play Loop" menu items to view the animation. A mini-control panel will appear, which allows you to set the playback speed, step through the frames one by one, reverse the motion, and start from the first frame again. "Play Once" will stop the playback at the end of each animation, whereas the "Play Loop" will automatically wrap from the last frame back to the first. "Play Big" will use a fullscreen window for the animation playback, though the frame rate might slow down. A final option, "Free RAM" will drop the animation from memory when you're done with it.

A final command is found in the Object menu. "Snapshot" works much like the Cycle command of the same name. It allows you to pick an object in the world, then save it so you can manipulate it later in any of the editors. Why would we want to do this? Aren't objects loaded from a file to start with? So why bother saving it? Well, that's almost true. There are some manipulations you can perform with Stage and Action that will modify your object, and you might want to save the modified version. These manipulations are both controlled from the Action editor. Morphing objects and special F/X can change both the appearance and geometry of your object, and you might want to save your object in this changed configuration. For example, you might have an object morphing from one shape to another, and want to have a copy of the object in its mid-morph configuration to manipulate. To use Snapshot, just pick the object you want to save (in the frame you want) and select the "Snapshot" command. Imagine will ask for the filename to save the object as. The object can then be loaded in and manipulated in the Detail or Stage editors; it's just like any other object.

Morphing is discussed on page 103 and F/X are discussed on page 112.

Stephen Menzies, an excellent Amiga artist, used this function to produce excellent results in one of his renderings. In his picture *Exploded Garden*, he took made an ellipse and used the Explode F/X to expand it to about twice its volume. The individual triangles of the ellipse became visible with large gaps between them, giving the volume a delicate flower appearance. He then used the "Snapshot" command to save the half-exploded volume as a new object. By adding a green stem to the "flower" in the Detail editor, he was able to create a very effective plant model.

Chapter 7

The Action Editor

The Action editor is a close partner of the Stage editor. In fact, there is nothing that can be done in Stage that can't be done from Action other than editing paths. What Action allows you to do is define where your actors (lights, camera, and objects) are positioned in every frame of your animation. The Stage is very convenient since it allows interactive manipulation of your world, but complete control is provided from Action.

The Action editor has a display unlike any other of Imagine's editors. Instead of showing a graphic representation of your world as in Stage, it instead shows a *timeline* of your world, identifying the actors and their position, alignment, scale, and F/X applied to those actors, as well as the global world parameters in each frame. These timelines completely define what your scene will look like. You can think of the Stage as being an interactive way to manipulate some of the information in the timelines.

7.1 Time Line Display

The timeline display is very straightforward. The camera and every object and light in the scene each have a horizontal row containing seven subrows. The vertical columns represent time, as measured by frame numbers. The total number of frames in your animation is shown in the small gadget in the upper left of the screen. The default number of frames is just one: a single still frame. To make an animation, just enter a new number in the "# of frames" gadget. The most number of frames you can have is 9,999, which is probably a hundred times longer than most animations you will ever make.

If there are many actors, they might continue on downwards off of the screen. You can use the large scrollbar on the right hand side of the screen to view any section of the list of actors you like. You can similarly scroll through frames that extend off of the screen by using the horizontal scrollbar on the bottom of the screen.

The subrows represent different parameters of the object, such as its location and orientation in the world and what the filename of the object is. A complete explanation of each subrow is found in next sections, but here is a quick summary of what the subrows represent.

Actor Specifies what the actor represents, either by naming the object's filename or

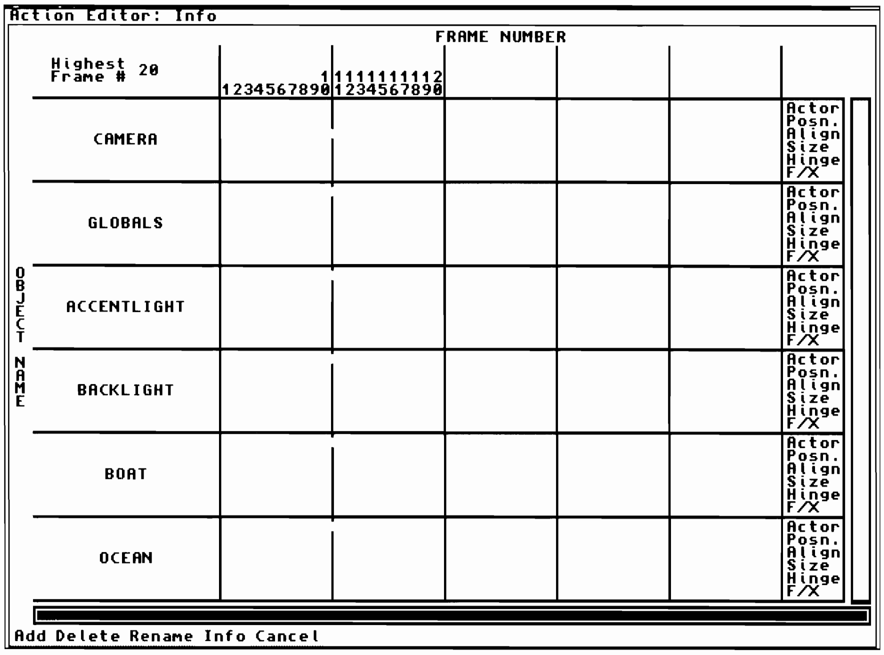


Figure 7.1: A view of the Action display for an example animation

light source information. Also controls object morphing and Cycle objects. The camera has an Actor subrow, but you cannot add a timeline to it. (Page 101)

Position Specifies the actor's position in time, either by key positions or paths. (Page 106)

Alignment Specifies the actor's orientation, either by keys, paths, or tracking to other objects. (Page 109)

Size Specifies the actor's size by keys. (Page 111)

Hinge Defines another object which is used to modify the path the actor takes when it moves between key positions. (Page 106)

F/X Adds a special function to an object, like explosions or rippling. There are two subrows so you can add more than one F/X simultaneously. (Page 112)

A typical view of an Action screen is shown in Figure 7.1. The vertical columns represent frames. Each of the subrows can extend across one or more of these frames, defining one of the actor's attributes for that range of frames. An animation of a car moving down a road for 20 frames might have an object cleverly called *car*. If the car entered the animation in frame 1 and left the animation in frame 20, there would be a bar in the "Actor" subrow in the car's row, stretching from the column representing frame 1 to the column representing frame 20. The bar tells Imagine that there is indeed an object that should be in the world from frame 1 to frame 20, and also tells Imagine what the filename of the car object is. If you were rendering a single still frame, the subrows would all just have stubby, one column wide bars.

These bars graphically represent how and when information such as the position or orientation of actors are defined. The specific information that these bars represent (such as what the actual filename of the car object is) can be modified very easily. The Action editor has certain modes (all accessible from the Functions menu) which will edit, add, delete, and even rename actors and their timelines. The default mode, "Info," will pop up a requester for any timeline that is clicked on. The requester lets you inspect and modify all of the parameters that the timeline defines, including the start and stop frame numbers of the timeline itself.

Different varieties of timelines are shown in different colors to reflect the different types of information they represent. Even in the same subrow there are sometimes different color timelines to represent different methods of determining an object's configuration. For example, a timeline that determines the object's position by keys is colored green, whereas a yellow bar represents the position when it is specified by a path. Don't worry too much about remembering what color stands for what; you can always click on a bar in "Info" mode to see what the bar stands for. If all of the bars were the same color, it would be hard to distinguish the individual bars from this big spaghetti pile of monochrome boxes. This way we still have a spaghetti pile, but at least we can see the individual noodles quickly and easily.

What a weird description.

The detailed information that each of these different color bars contain is described in the next sections, but this is a quick guide to the different bar types:

Red Actor, either an object filename or lightsource parameters

Orange Actor, representing a morphing object

Neon Green Position, defined by keys

Yellow Position, defined by a spline path

Blue Alignment, defined by keys

Aqua Alignment, defined by a path the object is following

Light Blue Alignment, defined by a "track"

Purple Size, defined by keys

Pink Hinges

Blue F/X

The basic commands from the Functions menu all act on all of these bars the same way. You can create a new bar by entering "Add" mode. Position the cursor in the subrow and frame number where you want the bar to start, and click the left mouse button. Imagine will print something like "Start Frame is 4, choose ending frame." In the *same subrow* click on the column representing the ending frame of the bar. If the subrow you are in allows different types of bars (like the option of using keys or paths for determining position) a requester will pop up asking you to choose between the different types of bars. For subrows with only one type of bar in them (like the actor bar,)no requester will appear and the default type will be used. In either case, another requester will appear, representing the detailed information that bar represents, such as the actor information specifying the object's filename. This is the same requester that "Info" displays; Imagine assumes that you want to enter the particular information the new bar represents immediately after you create it. You can always change the information later with the "Info" command again.

If you start adding a new bar but want to cancel the addition (because you screwed up, or lost track of what you are adding) you can use the menu option "Cancel Add" to stop the addition.

You can also add new actors to the world by adding a new "Actor" bar in the row with an object named *(new)*, which is always found at the end of the object list.

It is simple to delete timelines. Enter "Delete" mode and click on the bar you don't want anymore. Boom, the timeline disappears. You can also delete entire objects by clicking on their name at the far left, which erases all six of the subrows and removes the object completely from the world.

The "Rename" mode acts just like the "Rename" option from the Stage Editor. The name of an object does not necessarily have to be the filename of the object, it can be set to whatever name is most convenient or descriptive. This makes naming multiple copies of an object with more descriptive names easier. Two or more objects with the same name will automatically numbered, so you will get names like *CAR* and *CAR.2*. To rename an object, enter "Rename" mode and click on the name of the object on the far left.

When you have a lot of actors, sometimes the display can get a little confusing. You can use the "Find" command to find a specific actor if you know its name. "Sort" will sort the actors in alphabetical order. This is more useful then you might think. When you are positioning objects in the Stage editor, sometimes you want to reorder the actors so that the objects you are manipulating are rendered in the preview window first; you don't want to wait for the other objects to render. If you rename your objects from something like *CAR* to *AAACAR* then use "Sort" you'll place the object first in the list of actors, and it will render in the preview window before anything else. When you have many complex objects, this can be a great timesaver when you are making frequent adjustments to your scene.

7.2 Keys

The three types timelines representing position, alignment, and scaling share one convenient way of defining the object's orientation. They all can be defined by keys. By defining the starting position of an object in one frame and the ending position in another frame, Imagine can smoothly move the object from one point to the other in the inbetween frames.

Keys are also discussed in the Stage editor on page 92.

Keys show up clearly in the Action editor. For example, position keys are shown by green bars which extend over the frames they are keying.*The keyframe the bar represents is the* last *frame of the bar. The rest of the bar shows which frames the position or whatever is being interpolated over.* If there is no position bar for a particular frame, the position of the object is determined by the position in the last frame which the object had a position defined. For example, if an airplane has a key for one position in frame 1 and a key for another position in frame 40, you will see two bars in the position subrow. The first key will be only a small red block, beginning and ending on frame 1. This bar says "On frame 1, the object ends up in this position." The second bar would extend from frame 2 to frame 40, and would contain the final position of the airplane. This frame says "At frame 40, the object ends up at this position. From frames 2–39, the object's position should be smoothly interpolated."

Key frames are determined by the last *frame of a position, alignment, or size bar!*

By the way, keyframes are the last frame of a position, alignment, or size bar. Sure, I've said this 4 times now, but it is crucial!

To make an object like a tree stay in one place over time, you can just add a one-frame bar in frame 1. In future frames, there is no bar to tell Imagine where to place the object, so it goes back to the last known position (defined in frame 1) and uses that. Thus, the object stays in place for the duration of the animation.

To "teleport" objects (make them change position by suddenly jumping from one place to another), just add one-frame-wide bars. For example, a one frame bar in frame 1 defining one position and a second one frame wide bar in frame 20 defining another position will create an animation of an object that stays in one place for frames 1–19, and suddenly jumps to a second position and stays there for frames 20–40.

If there is no position bar in frame 1, Imagine sets the position to 0,0,0. Similarly, the default alignment is straight and size is 32.0 (the default axis size in Detail.) You almost always want to define a starting position, alignment, and size, or you can easily confuse yourself (especially when you can't figure out why your object is stuck at the origin.)

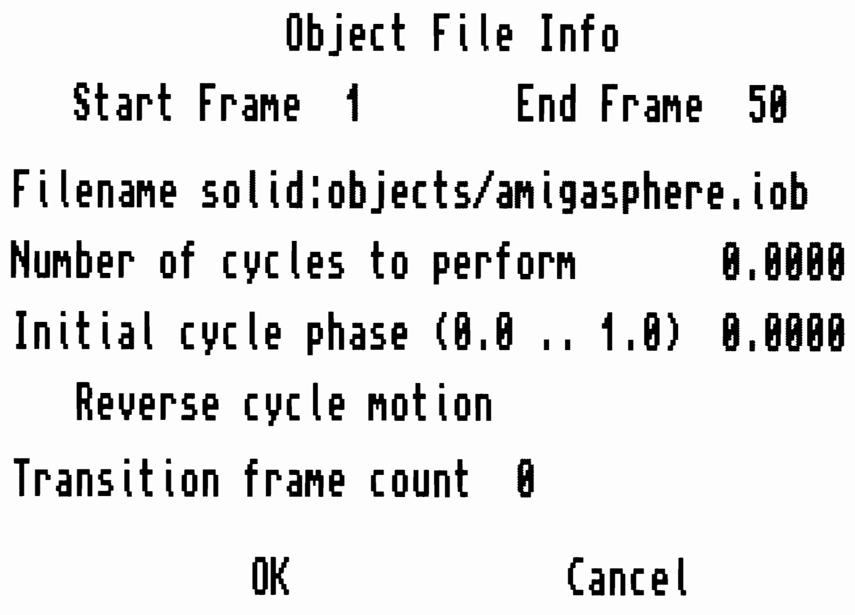
When you use the Stage editor's "Position Bar," "Alignment Bar," or "Size Bar" commands, the timelines are modified in one of three ways. If a key already exists for that frame (a bar ends on that frame,) the key position, alignment, or size for that key is replaced with the object's new position, alignment, or size. If a bar doesn't exist, a new bar will be made with the end of the bar in the current frame and the beginning of the bar extending back to the frame after the last frame containing a bar. If a bar passes through the current frame, that bar's beginning is moved to the frame after the current frame, and a new bar is added ending on the current frame and extending back to the frame after the last frame containing a bar.

This sounds a little confusing, but read it through slowly...

Keys are important, and using the "Position Bar," "Alignment Bar," and "Size Bar" commands from Stage combined with manual editing from the Action editor you can produce any key motion you like.

7.3 Actors

A very straightforward subrow for every object defines exactly what object (the specific filename) is used to represent the actor when the scene is rendered. The object is usually an object you saved from the Detail editor, but it could also be a Cycle object. If there is no timeline for any particular frame, *that Actor will not be rendered for that frame!* If the subrow is describes a normal object (as opposed to a light or the Global actor) when you use "Info" to display the specific information in the


Object File Info

Start Frame 1 End Frame 50

Filename solid:objects/amigasphere.iob

Number of cycles to perform 0.0000

Initial cycle phase (0.0 .. 1.0) 0.0000

Reverse cycle motion

Transition frame count 0

OK Cancel

Figure 7.2: The standard actor information requester

"Actor" subrow, you are shown a requester like the one in Figure 7.2. The gadgets in the requester allow you to inspect or even change the information contained in the subrow. To change a value, just click on the gadget, edit the number or text, then hit return.

The three gadgets on the top of the requester are the most commonly used. The "Start Frame" allows you to change the starting frame of the Actor timeline, and "End Frame" sets the ending frame. By setting these gadgets you can move timelines and stretch and shrink their durations.

The most important gadget is labeled "Filename" and determines exactly what object the actor represents. The name includes the absolute path of the file, which is actually something of a pain; if you move your objects into a new directory, you have to change the path in this variable; sharing projects with other people can be difficult if you don't have the same directory structure.

The way I set up my object directories is described on page 151.

When you load an object from the Stage editor, the filename you select for the object is placed in this timeline. The object can also be a spline path object, which can be used to define the position of other objects (See page 108), though the path itself is not rendered.

7.3.1 Using Cycle Objects

Cycle objects are described for an entire chapter, starting on page 73.

Cycle objects can also be used as actors. When their filename is specified in the "Filename" gadget, cycle objects are added to the scene just like normal objects. Unlike normal objects, they will change their shape with time, which is the whole point of using them.

You can control the way your cycle behaves from the Action editor. The "Number of cycles to perform" gadget controls how many times the cycle will repeat during the frames specified (the width of the bar.) If you set this number to 1.0, the cycle will repeat exactly once during the range of frames. If set to 0.0 (the default) the cycle object will not animate at all and just show its initial configuration for all frames. A value of 2.0 will repeat the cycle twice during the frames specified by the bar, and a value of 0.5 will make the cycle perform the first *half* of its cycle. The number of frames you specified the cycle to have in the Cycle editor is completely independent from the number of frames you specify the object to animate over.

The starting phase of the cycle can be set with the second gadget. This allows you to specify where in the "loop" you want the cycle animation to start. Normally you want to start with the beginning of the cycle, so this phase would be 0.0. If you wanted to start halfway through the cycle, you would set this value to 0.5.

The independence of "frames" for making Cycle objects is discussed on page 80.

A final gadget is a simple on/off button. "Reverse cycle motion" will make the animated cycle animate *backwards* in time, so you could make your clock run in reverse or whatever.

7.3.2 Morphing

One of the advanced abilities of Imagine is called "morph." This is an incredibly powerful function that allows you to smoothly change one object into another over a period of time. You might have two versions of an object, like a bird with its wings up and another copy of the bird with its wings down. Morph could smoothly change one bird into the other and back, creating a single bird that flaps its wings! This is a different effect from the animation cycle objects provide, since the wings are integral parts of the object, not separate objects that are just rotating to make the flap.

Not only does the physical shape of the object change over time, but the attributes, textures and brushmaps can also change. If you take an object with a green-paint attribute, you could morph that object into another copy of the same object with a chrome attribute. Over a period of frames, the green paint would change into chrome, slowly and smoothly! This is especially effective with textures, since all of the parameters of the textures can change from one value to another. You could set up an object with a Checks texture with small checks. By morphing that object into a copy of the same object with big checks, you could have the checks "expand" in time. You can only morph the a texture to another version of itself, so you can play with the check size, but you can't make Checks gradually become Wood.

Brushmaps can change their positions and sizes in a similar manner, so you can have an image shrink, move, or spin in time. If you use two different brushmaps on the two objects, the images will *not* morph or crossfade. The first image will be shown through the entire morph, and will be suddenly replaced with the second brushmap at the end of the morph. Luckily, you *can* use an animated brush. If you want the actual brushmap image to morph as well, you might want to look into using DPaint IV's picture morphing.

Animated brushmaps are discussed on page 47.

Morphing can produce some great effects, but unfortunately it won't work with every pair of objects. The algorithm Imagine uses to change the shape of the morphed object in time is very straightforward; it takes each point in the object and linearly interpolates that point's position in time, using the starting and ending points from

DPaint IV is discussed on page 189.

the two objects that make up the starting and ending object configurations. This means that not only do the objects have to have the same number of points in them, but they must also have the same structure; they must have the same number of edges and faces in the same configuration. There are really only two practical ways to get objects in this special configuration.

Point manipulation is discussed on page 24, and magnetism is discussed on page 59.

The easiest way to make two objects that you can morph is to build the first object normally. To make the second object, drag the points in the first object to the new shape by picking points and manipulating them, or using magnetism and dragging the points around. As long as you do not add or delete points, edges, or faces, your two objects will have the correct configuration for morphing from one to the other.

Another way of making two objects with the correct configuration for morphing is to use the Forms editor. If you make an object in Forms, you are only dragging the points that define the shape around and not actually adding or deleting new points, edges, or faces. Make any two objects *using the same number of "Points" and "Slices" when using "New" to initially make the objects.* As long as you don't add or delete points and slices, any two objects created with these same parameters will morph-compatible.

Two sets of grouped objects will not morph the way you expect; the positions of the objects do *not* change over time, until suddenly the configuration will change to the final position for the last frame only. If you want to move objects relative to each other like this (regardless of whether you want the surfaces or shapes of the individual objects to change as well) use the Cycle editor to make a cycle; this is *exactly* what Cycle is designed for.

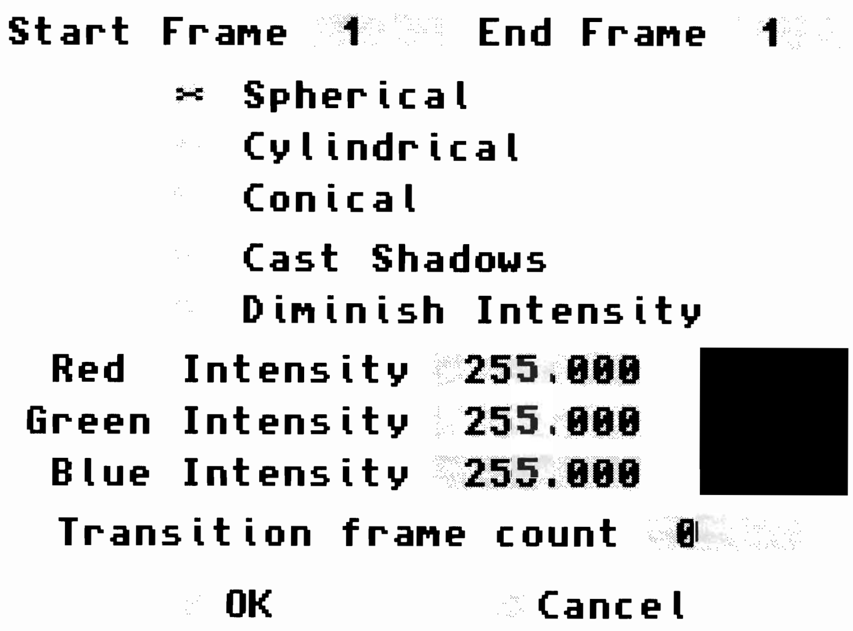
Cycle objects will morph, which allows an incredible amount of control over your animations, especially character animation. Not only will the individual objects that make up the cycle object morph their shape, but the actual cycle motion itself will change from one type to another.

Cycle morphing is discussed on page 82.

Morphing is somewhat tricky, but well worth experimenting with. The toughest part is making two objects which have the proper relationship to let Imagine morph their shapes. Using the two methods described previously (point dragging and Forms,) you can usually create the transition you are looking for.

It is pretty easy to actually set up a morph in Action. Add your first object as a timeline spanning one or more frames. Add the second object starting in the frame immediately after the first, extending for two or more frames (depending on how long you want the morph to take.) Enter the number of frames you want the morph to take in the "Transition frame count" gadget (this is usually equal to the width of the second bar minus one.) The frames where the object is being morphed will show up as orange on the timeline instead of the standard red.

For example, if you want an animation of an object morphing from one object in frame one, and ending up as a second object in frame 10, you would add the first object as a one-frame-wide bar at frame one, and the second object as a nine-frame-wide bar from frame 2 to frame 10. You would enter "8" as the "Transition frame count" since the morph takes place over the eight frames 2–9.



Start Frame 1 End Frame 1

[✓ Spherical]

Cylindrical

Conical

Cast Shadows

Diminish Intensity

Red Intensity 255.000

Green Intensity 255.000

Blue Intensity 255.000

Transition frame count 0

OK Cancel

Figure 7.3: The information requester for a light

7.3.3 Lights

Lights aren't specified like an object that was created in the Detail, Cycle, or Forms editors. Instead of the timeline specifying a filename, its requester defines the attributes of the light instead. The standard "Info" requester for a lightsource is shown in Figure 7.3. Like the normal Actor requester, you can change the starting and ending frames for the light in the "Start Frame" and "End Frame" gadgets.

Lights give you a lot of control over their effect on the scene. There are three different ways to define the direction the light shines. The most common type of light is "Spherical" which means that the light emits light in all directions equally, like a lamp without a shade. "Cylindrical" is like a spotlight; it casts a beam in one direction over a narrow spot. "Conical" is like a flashlight; it is also directional, but the beam spreads out as it travels, so the beam gets wider far away. You can click on the appropriate gadget to define any of the three types of lights. The Size subrow of a light defines the angle and distance that a cylindrical or conical light will emit light, and the Alignment subrow controls the direction the light is pointed.

See page 111 for size control of lights.

Another option is "Cast shadows." You have the option of making a light cast shadows or not. It might seem silly at first to add a light that doesn't cast a shadow, but they can be very useful when you are just trying to get light onto objects so they can be seen as opposed to trying to model a realistic lighting setup. *Shadows are only rendered in trace mode!* Shadows also take a lot of time to compute, and rendering times can increase dramatically when you have one or more shadow-casting lights.

One last option lets you make the light diminish in intensity with distance. A lamp in the real world will be less bright the further you get away from it; the light intensity drops off the further you get away. By selecting the "Diminish Intensity" flag, you can make your lightsources act this way. If you are modeling something like sunlight, you probably do not want to use this option. Real-life spotlights also do not really diminish their intensity with distance.

You can change the intensity and color of the light that is emitted by using the three intensity gadgets. By entering a value into each of the red, green, and blue gadgets, you can change the color and intensity of the lightsource. The default of RGB= (255,255,255) works very well for most purposes. To make a yellow light, you might change the values to (255,255,100). *It is often difficult to predict the results of using colored lights in a scene.* Imagine handles colored lights accurately and with no difficulty, but us mere humans have much less color intuition than you might think. Quick: what would you see if you shined a blue-green light on a yellow object?[1]

If you experiment with colored lights, you'll notice that you *can* mix them with other lights; a red, a green, and a blue light shined on a white object will return a white color, though the slight differences in angles will cause interesting color fringes.

You can also change intensity just by scaling the values in the Intensity gadgets. An RGB value of (128,128,128) would be half as intense as (255,255,255). You can also make your light *brighter* than the default of 255, which produces very bright results like the noontime sun (shadows become very deep, too.) If you crank the values high enough (like 3000, 2000, 500), you can really wash out your scene as if it were lit with nuclear weapons as opposed to simple lamps.

7.3.4 Axes (Tracks)

You can also add lone axes to your world, mostly for making "tracks." Axes have a very simple "Info" requester that lets you define the starting and ending frames of the track and nothing else.

Tracks are talked about in Stage on page 90, and later in this chapter on page 110.

7.4 Position and Hinge

The next subrow defines the position of the object in the world. The standard method of defining positions is by the use of keys, but there are a couple of alternative ways to define your object's position.

See page 100 for information about keys.

One alternative still uses keys to define the starting and ending positions of your object on two different frames. The linear interpolation that Imagine uses for keys sometimes just isn't "liquid" enough to make the motion look smooth. Sudden accelerations and sharp turns at each key just makes an artificial feel as the object bips around in straight lines like a ball in an out of control pinball game.

One method for "softening" the keys is to use the "Hinge" subrow. Hinge makes your object follow the keys you set up, but instead of traveling in a straight line, the object takes a curved course. This curve is defined by a "Hinge" object, which acts like a pivot that the object moves around. You simply add a "Hinge" subrow for the frames which you want to curve the object's motion (usually the same frames you are keyframing the position over) and enter the name of the hinge object in the requester.

[1]The object would be a pale green. Just accept it.

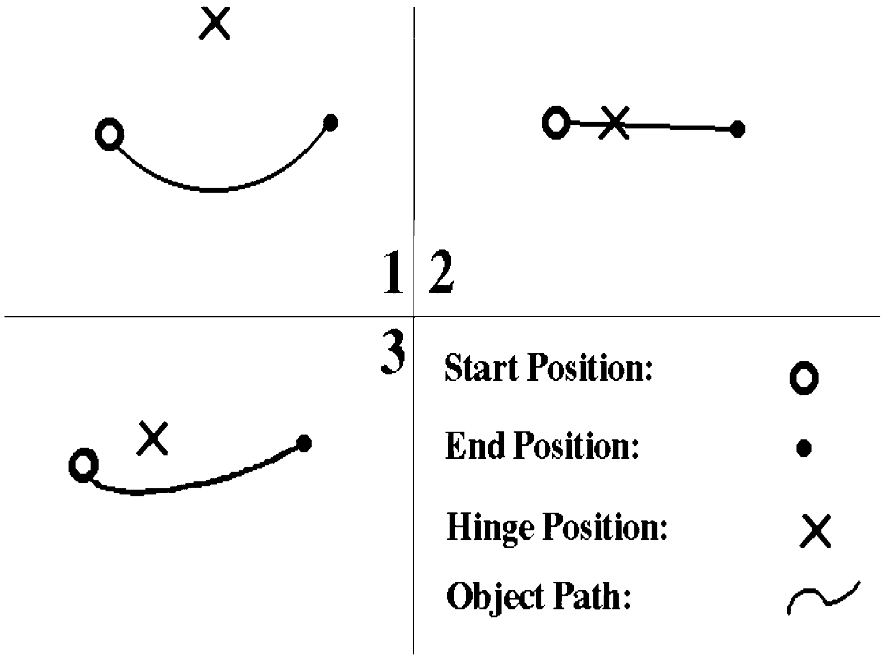


Figure 7.4: Paths an object follows with different hinges

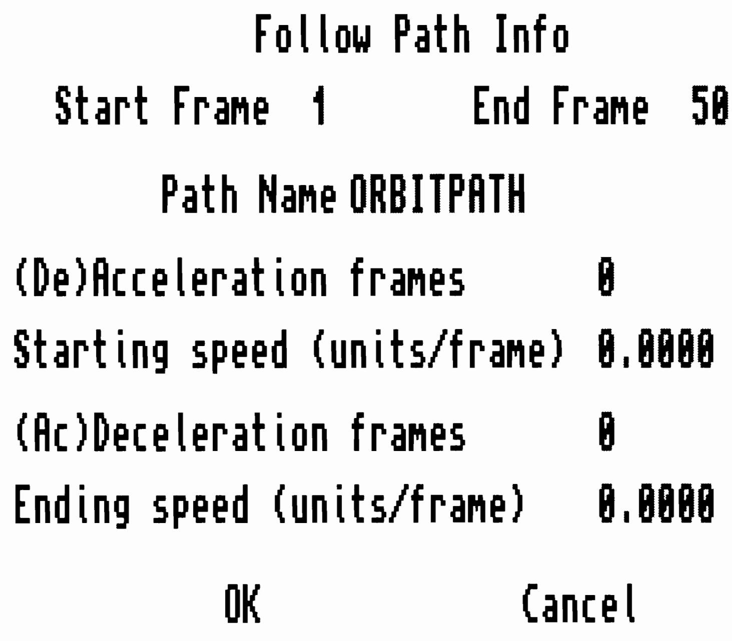
You can think of the hinge object as a center of rotation for your keyframed object to move around. Imagine knows where the object starts and where it has to end up, and creates a length of "rope" that it ties to the object. To get from one key to the other, the "rope" is taken in or let out as the object attached to the rope swings around the hinge. What actually happens is Imagine has the object follow the path an ellipse as defined by the starting and ending points, using the hinge object position as a focus of the ellipse.

Three examples are shown in Figure 7.4. In the first one, the hinge is equidistant from the starting and ending positions, and the object will follow a circular arc using the hinge as the center of the circle. The second example shows the hinge directly between the starting and ending positions. The object will still follow a straight line, since the ellipse defined by the hinge is flattened out into invisibility. The third example shows an asymmetrical hinge. The ellipse defined by the hinge is much curvier at the beginning of the path since it is closer to the hinge. A good rule of thumb is that the further the hinge is away from the points, the less effect it will have on the path. The exception is when the hinge is directly between the two points, when the object travels an ellipse so stretched out it is a straight line.

Hinges are a bit strange, but trivial to use once you understand them. A more common way to get curved motion is by using spline paths to define the position of the object.

7.4.1 Paths

You can use spline paths to define the position of your objects in time; in fact, this is their main use. You can add your spline path to the world as a separate actor than



Follow Path Info

Start Frame 1 End Frame 50

Path Name ORBITPATH

```
(De)Acceleration frames      0
Starting speed (units/frame) 0.0000
(Ac)Deceleration frames      0
Ending speed (units/frame)   0.0000
```

OK Cancel

Figure 7.5: The path information requester

have other actors follow the path.

Telling an object to follow a path is quite simple. Add a subrow to the actor's position for the frames where you want the path to be followed. The "Info" requester shown in Figure 7.5 will appear. Just like other timelines, you can change the starting and ending times of the timeline with the two "Start Frame" and "End Frame" gadgets.

Creation of spline paths is discussed on page 48 and on page 94.

To have your object follow the path, simply enter the path's name in the "Path Name" requester. This name is the one assigned to the actor by from either Stage or Action; it is *not* the filename of the path, it is the name shown on the far left of one of the other actor rows.

When an object is assigned to follow a path, it begins at the start of the path in the first frame, and ends at the ending frame at the opposite end of the path (if it is an open path) or back at the start of the path (if it is a closed path). The object travels smoothly between the starting and ending positions during the inbetween frames. Imagine will happily let you have several objects follow the same path at the same time, and each object is independently controllable.

Sometimes the closed path can be annoying since the starting and ending positions are exactly the same. If you want the object to loop around more than once (or you are looping an animation) the object will appear to "stutter" as it will be in the same place for two subsequent frames. Also, getting an object to travel only part of a path or start in the middle of the path is difficult. You can solve all of these problems by adding another actor to your scene. Add the path subrow to this actor for the proper frames, but *only have an actor subrow for the frames in the path you want to use.* You can use several actor rows for one object, just don't have an actor bar in more than one of the rows during the same frame. You can make the object follow a closed path for all but one frame, then use a second bar to make it follow the same path a second time, eliminating the stuttering. To use just part of a path, just "abort" early and use the second actor. This is a semi-advanced technique, and is just a workaround for some of the limitations in using paths.

Since an object assigned to a path travels the length of the path exactly once during the time you've defined, you have to add multiple copies of the path if you want to make several revolutions of a closed path.

The velocity of the object along the path is constant. (Well, not really, that's what the next section is about, but this is just a transition sentence that assumes you don't know that yet, right?) You can compute the velocity of the object by using the "Show Path Length" command from the Stage editor, and dividing by the number of frames it takes for the object to move along the path.

Path length is discussed on page 95.

7.4.2 Acceleration and Deceleration

Making an object follow a path gives it a smooth trail to follow, but if the speed of the object is constant the motion can look artificial. An F16 in a dogfight follows a complex twisting trail that can be easily defined by spline paths, but the F16 certainly doesn't travel at a constant speed. A car turning into a driveway will slow as it turns. (Or maybe not, depending on how good you are at modeling a collision with a garage.)

Imagine allows you to accelerate and decelerate your object as it travels along a path. Specifically, you can specify the starting speed of an object (in the first frame), the number of frames the object can accelerate over, the final velocity of the object in the last frame, and the number of frames the object can decelerate over.[2] Imagine will set the speed of the object based on the path length, the number of frames the path is defined over, and those four acceleration parameters.

Imagine's only real decision is to pick the "cruising" speed of the object for the frames between the acceleration and deceleration frames. It chooses this speed based on the requirement that the object has to start and end at the specified positions and times. Sometimes the result isn't apparently obvious; if you have a 10 unit path to travel over 10 frames, specify an initial velocity of 2 units/frame, a final velocity of 2 units/frame, 2 frames of acceleration and 2 frames of deceleration, what happens? The object will start off fast, actually slow down for its "cruise" in the middle of the path, then speed up again.

7.5 Alignment

The alignment subrow control the orientation of your object. There are three different methods for defining alignment of objects.

[2] I know, I know, there is no such thing as deceleration, only acceleration in the other direction. Believe it or not I have a degree in physics from MIT, which is probably why I'm making this long footnote so people will realize that I know something about Newtonian motion and I'm not just a bonehead who doesn't know the difference between velocity and a potato.

7.5.1 Keys

Alignment keys are well explained in Section 7.2 on page 100. When the alignment of an object is defined by a key, the color of the bar is blue.

The one problem you can run into when using keys for alignment is the same problem that is described with cycle objects in Section 5.3 on page 80. If you want to rotate an object a full 360 degrees, you just make a key with the object at 0 degrees, a second key with the object at 180 degrees, and a third key with the object at 0 degrees, right? Well, we run into a problem: which way does Imagine choose to rotate the object? It might not rotate the object clockwise from 0 degrees to 180, but counterclockwise from 180 to 0. Imagine always takes the "shortest" rotation it can, but when both directions are equally short, Imagine has to decide which direction to rotate. It turns out that Imagine usually does the right thing when you use three keys like this, but you could still have problems if you accidentally make the second key 179.9 degrees instead of 180 degrees. In this case, Imagine *will* rotate the object one way then backwards.

The solution to this problem (if it turns up) is to simply use four keys and not three. To rotate an object 360 degrees over 12 frames, *don't* make a 0 degree key at frame 1, a 180 degree frame at frame 7, and another 0 degree key at frame 13. Instead, make a 0 degree key at frame 1, a 120 degree key at frame 5, a 240 degree key at frame 9, and a 0 degree key at frame 13. By using the extra key when you are in doubt, you can never go wrong.

7.5.2 Tracking

Another method of defining the orientation of an actor is to have the actor "track" another object. In each frame, Imagine will rotate the actor so that it its Y axis points directly at the track object. You could use this to keep the camera continually pointed at an object, have a "follow" spotlight, or have an antiaircraft gun track a 767 as comes in for a landing. Tracks are very quick and easy to use. The most common use of "tracking" is to align the camera, since it is automatic and requires no thought on your part to keep the scene centered on the object you want. When the camera has been defined to track an object, you can update the camera's alignment when you move it in stage by using the "Camera (Re)track" command.

Camera (Re)Track is discussed on page 90.

One further control lets you rotate the object (or camera) as it points at the tracking object. (The camera is pointed at the track, but is it rightsideup, on its side, or what?) The two gadgets in the "Info" requester for tracking alignment let you set the initial and final Y rotation angle. The angle is interpolated from the start of the track timeline to the end frame, so a value of 360.0 will smoothly spin the object or camera exactly one revolution the way around during the frames the alignment track is specified for.

7.5.3 Path Alignment

A final method of setting object orientation is only an option when you are having the object follow a path. (See page 108.) Most times when an object like an airplane is traveling along a path, you want the object to stay pointed along its direction of

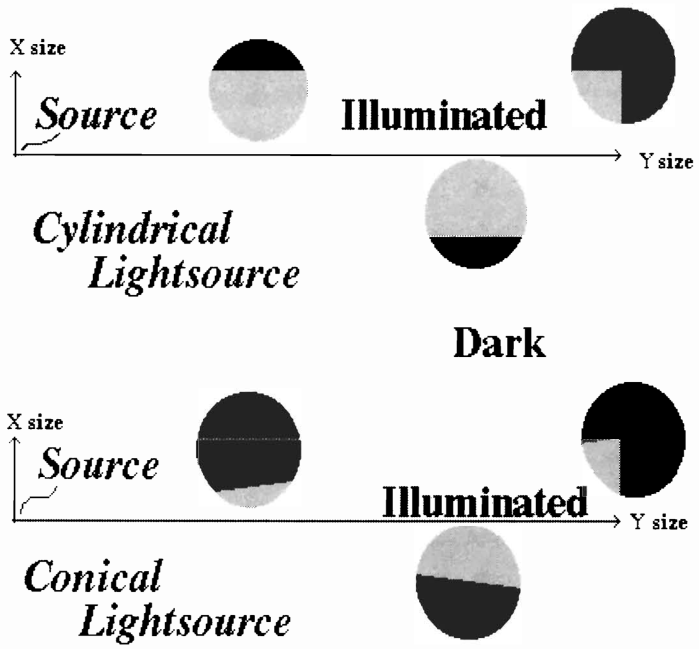


Figure 7.6: Size definition of conical and cylindrical lights

motion. (If an airplane heading North starts moving East, you can be certain it's turned in that direction and moving forward, as opposed to still facing North and moving sideways.) The "Align to Path" timeline lets you make your object use the path it is following to determine its orientation; the Y axis of your object will always be pointed in the direction of motion. Most of the time when you use paths to define motion, you want to use path alignment.

One option that you can set in the "Info" requester is the "Keep Y Horizontal" gadget. This allows you to keep your object level, as opposed to pitching up and down if your path climbs or drops. A helicopter climbing into the air does not tilt back; it stays level, though it still points in the horizontal direction of motion.

7.6 Size

The definition of the size of an actor is extremely straightforward, since it can *only* be defined by keys. A description of how the keys are defined is in Section 7.2 on page 100. When you add a size bar, it will be colored purple.

Lights (defined from the Actor subrow) can be affected by the size subrow. Cylindrical and conical lights use the size subrow to set the width of the lightbeam and the distance the beam travels.

Figure 7.6 shows the two light types that are affected by the size subrow. For a cylindrical light (a spotlight), the Y size of the light determines the distance the light travels; beyond this distance the light will not show up. Often, I just this number to a very large value to get light that travels for an "infinite" distance. The X size sets the radius of the spot itself, so you can use a small X size to get a laser-beam thin

For light emitting objects, the size of the object's axis acts just like the size subrow.

spot of light. You can use an enormous X size to illuminate everything with parallel light (coming from the same direction) like sunlight in space.

Conical lights also use the Y size to determine the distance the light travels. The X axis determines the size of the spot *at the very end of the distance the light travels.* The cone angle is really determined by the ratio of X/Y.

7.7 Special F/X

F/X are a very special actor attribute. F/X will actually perform a complex operation on your actor over time, like explode it into its component triangles or automatically "squash" it like a basketball.

F/X are added for a period of frames, during which the effect will appear on your object. You can add more than one F/X simultaneously by using the second F/X channel during the same frames as the first one.

The different F/X are described in Appendix E on page 179. They are very powerful effects, and are a lot of fun to play with. One caution: try not to overuse F/X; they are tools that should be used where appropriate. Exploding logos get very boring very fast! A better approach is to use standard object creation and layout to make your scenes exciting, using F/X as a supplement, not a replacement, to your creativity.

It is occasionally useful to manipulate an object in the Detail editor in the same shape as it is in the process of being exploded, squashed, or whatever. The Stage command "Snapshot" described on page 96 is designed for exactly this purpose.

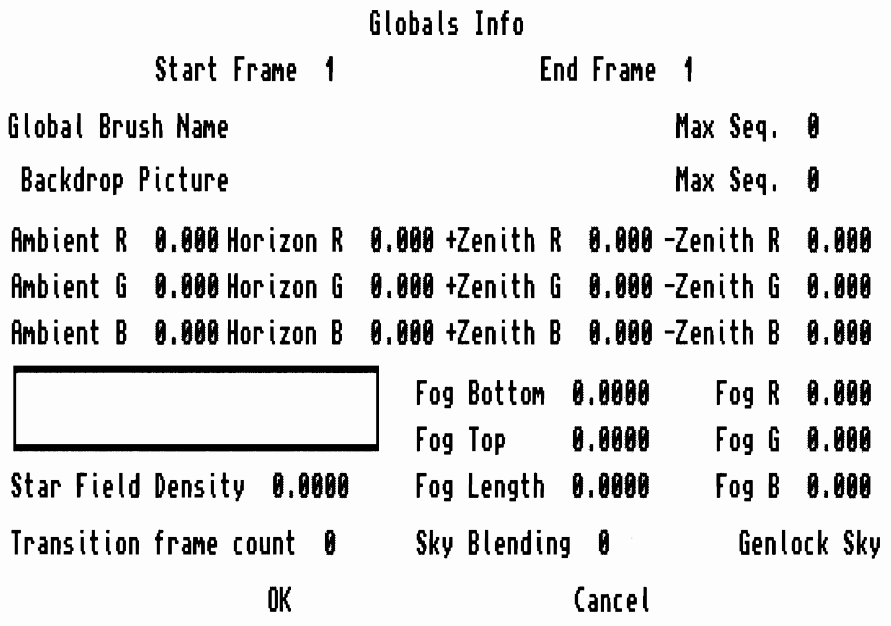
7.8 Global

There are many parameters that can affect the way your scene is rendered. An area where there is no object or ground shows the "sky," which can be set to be different colors. You might have a brushmap you want all of the objects in a scene to reflect. You can even add fog to your world. All of these parameters are controlled from the Globals actor.

Globals is obviously a special actor in Action. It has its own row, but adding a position, alignment, hinge, or F/X subrow to Globals is meaningless. The real parameters are all represented by the Global actor timeline. By entering "Info" mode and clicking on the Global actor subrow, you can edit all of the global parameters.

The reason that Global has a timeline instead of just a single requester in the Project editor is that you can actually have the global parameters change in time. If you want to fade the sky from violet to blue over the course of several frames, you can easily make the global parameters morph! Just like normal objects, the global parameters can be morphed in time. Just use "Add" mode and add a new globals actor for the frames you want to morph over. The "Info" requester has a gadget for a "Transition Frame Count" just like normal morphed objects. The interpolated frames will have all of the global parameters smoothly moving from the beginning to the end values.

Morphing is described on page 103.



Globals Info

Start Frame 1 End Frame 1

Global Brush Name Max Seq. 0

Backdrop Picture Max Seq. 0

Ambient R 0.000 Horizon R 0.000 +Zenith R 0.000 -Zenith R 0.000

Ambient G 0.000 Horizon G 0.000 +Zenith G 0.000 -Zenith G 0.000

Ambient B 0.000 Horizon B 0.000 +Zenith B 0.000 -Zenith B 0.000

Fog Bottom 0.0000 Fog R 0.000

Fog Top 0.0000 Fog G 0.000

Star Field Density 0.0000 Fog Length 0.0000 Fog B 0.000

Transition frame count 0 Sky Blending 0 Genlock Sky

OK Cancel

Figure 7.7: The Global Requester in Action

7.8.1 Ambient Light

Lighting is an important part of scene design, but no matter how many lights you use, it is difficult to really reproduce "real world" lighting. Shadows in the real world are never black; there is always extra diffuse light that allows you to see *some* detail in the shadows. This "ambient" light is difficult to model[3] but Imagine can approximate this diffuse lighting by adding an "ambient light" term. This lights every surface of the scene (usually by a small amount) to keep shadows from being pure black.

Ambient light is set in the Globals requester with the three "ambient light" intensity gadgets. You can choose the intensity, but I suggest using a small value like 10,10,10. If you use too much ambient light, your scene will lack color definition, like a vacation picture you get ruined by a cheesy one-hour-photo place. Values as high as 50 are sometimes safe, but only rarely do I have to raise the value that much. Colored ambient light is also usually not a good idea, though you might use it in something like an underwater scene where the ambient light really is tinted a certain color.

You can actually use ambient light to make a cartoony feel to your renderings. See page 146.

7.8.2 Sky and Stars

You can define the "sky" color of your world very easily. This color is shown in the background of your scene, and reflective and transparent objects will also show the sky properly reflected or transmitted through an object.

You can set the sky colors very easily by changing the values in the Global re-

[3]Though it is possible; it is done with a method called *radiosity* that is found in very high end renderers. It, unfortunately, is so slow it makes raytracing feel like real-time animation.

quester. There are three separate sky colors you can change. The sky is separated into three different bands, based on the how high (what angle) in the sky you are looking at. The "Horizon" is the wedge of sky that lies (logically) at the horizon (straight ahead) and extends up to an angle of 30 degrees. It also extends 30 degrees *below* the horizon (remember you might be able to see through a transparent ground, or you might not even have a ground at all.) The wedge of sky extending from 30 degrees to 90 degrees (straight up) is called "+Zenith." The wedge from -30 degrees to -90 degrees (straight down) is called "-Zenith." You can assign each wedge a different color; most of the time you obviously want a blue color for normal sky, or black for a nighttime sky or space scene.

Metals are especially sensitive to the sky colors. See page 166 for a discussion.

When the different wedges have different colors, there obviously has to be a transition between the colors. Imagine will fade from one sky color to the next over a narrow band, but you can make it make a more gradual change if you like. The "Sky Blending" gadget allows you to make the transition fairly sudden (using a small value like 0) or very gradual (with a large value like 255.) I usually use value of 50, which is gradual enough to look natural, but makes enough of a contrast to notice.

Another gadget is called "Genlock Sky." This on/off gadget allows you to make the sky black (so you can composite or genlock other images in the background), but *it still renders reflections using the sky colors you specify.*

A final option allows you to add stars to the background sky. These stars are placed randomly, and look fairly good. Unfortunately, they do not animate well (they don't move with the camera in the same way real stars would) and they don't show up in reflections or transmissions through objects. The stars are turned on by setting the "Star Field Density" to a small value like 0.03. You can increase or decrease the number of stars by changing the number.

Page 154 describes some alternatives to star backgrounds.

7.8.3 Fog

When rendered, Imagine can add the appearance of fog into your world automatically. This global fog is similar to object fog; light passing through the world is attenuated as it passes though the "atmosphere." Light coming from an object that passes through the air on the way to the camera will show the object's colors clearly if the distance it travels is fairly short. A ray traveling from a distant object will travel through more atmosphere and therefore be attenuated more.

The object fog attribute is discussed on page 31.

To make the appearance of fog, Imagine measures the distance that a ray travels through the "fog" and then mixes in a certain color with the ray depending on the distance the ray traveled. A ray that starts at a yellow object, travels through the fog, and enters the camera will start off yellow. If you set the "fog" color to a shade of grey, Imagine will add the grey color to this yellow ray depending on the distance the ray traveled. If the distance is very long, the yellow will be completely replaced with the grey.

This color that is added to the fog can be any shade, and it might not be obvious at first what color it should be. Is a nighttime fog black, white, or grey? I find that a slightly greyer version of the sky color works best, so a very dark grey would probably work best for a night scene and a blue-white looks good for daytime scenes.

The amount of fog color added to the ray is dependent on the user-defined "Fog

Length" as well as the distance the ray travels through the fog. The longer the fog length, the less the fog affects the ray. The fog length is defined as the distance it takes for the ray to be reduced to 37% of its original intensity. Mathematically, the final color of the ray is defined by

$$C_f = C \cdot e^{-\frac{L}{\lambda}} + F \cdot (1 - e^{-\frac{L}{\lambda}})$$

Forgive the equation, I just couldn't help myself!

where C_f is the final rendered color, C is the "true" color that the ray starts off with (what it would return if there is no fog), L is the distance the ray traveled through the fog, λ is the fog length, and F is the color of the fog.

Some people will appreciate the above formula, others will be shaking their heads saying "Jeez, Steve, just tell me how to use fog!" Don't worry, if you don't understand the funky math, you aren't missing much.

To use fog, you want to set the fog length to the distance at which you want things to start visibly disappearing. If you have a street scene with city blocks, and you want a light fog in the scene, just ask "How many blocks to I really want to see clearly?" If you want the fog to really obscure details of objects two blocks away, set the fog length to that two-block distance, as measured in Imagine Units in the Stage. Objects at that distance will really start fading into the fog; they'll be at 37% of their original color. Objects two fog lengths away will only appear at 14% of their original color. Objects three fog lengths away will appear at 5% of their original color and so on. An object that is one half of a fog length away will be about 61% of their original color, and objects one tenth of a fog length away will be at 90% of their original color. Gee, I should have made a table.

You can also restrict fog to appear at certain altitudes. This can make building a cloud layer quick and easy! The fog will start and stop at the altitudes you specify. You can have an airplane fly through the cloud layer and break out into the sunshine above. Fun!

To make a global fog, just enter the global requester and enter in the RGB colors for the fog and the fog length into the appropriate gadgets. A fog length of zero will turn off fog. To make different altitude bands, enter in values for the "Fog Top" and "Fog Bottom" in standard Imagine Units. If you set both the "Fog Top" and "Fog Bottom" to zero, the fog will extend to all of the world at every altitude.

Fog is very versatile. Even a light amount of fog will *really* enhance the apparent depth of your scene. You can really abuse fog in fun ways, too. If you make a dark blue-green color fog with a moderately short fog length, you can make some *terrific* underwater scenes.

Fog parameters also morph, so you can have *great* fun with changing fog parameters. You could have the fog length change over time, which will slowly make your world become Hey, murky or clear. By morphing the fog altitudes, you can have a fog layer "drop" onto the scene. A rising sun can slowly change the sky colors and start burning off the fog in your scene. Don't be afraid to use fog, it is a powerful effect that is actually pretty easy to use.

Though fog doesn't take any RAM to render, it does slow rendering down. Any non-zero fog length will make Imagine take about 5-10% longer to render scenes (on my machine at least.) The effects are usually worth it, but sometimes I design my scene first, test render it, and add fog only for the final renderings.

7.8.4 Background and Reflection Maps

Creation of realistic scenes is an involved process. One of the more common tricks in rendering is to use flat brushmaps to represent the background of your world (like a play might use a backdrop of a flat canvas with a landscape painted on it.) Your scene is placed over this backdrop. For example, you could use a picture of something like a cloudy sky as a backdrop to a world containing an airplane model. When rendered, the airplane would be superimposed over the sky picture. This option is very straightforward and easy to use. You can even animate the backdrop by using the "Max Seq. Number" gadget to specify multiple frames. This uses the same format as animated brushmaps, which is described on page 47.

You can also use (maybe abuse?) the backdrop feature in a few interesting ways that are described in the appendix on page 146.

The backdrop picture must be the same size (resolution) as the resolution you are rendering with! This bug (uh, sorry, "absent feature") makes using the backdrop picture *very* annoying when you are rendering different subprojects at different resolutions. There's really no way around this. Also, the backdrop picture is *not* seen through transparent objects, a serious limitation.

Imagine allows you to use a brushmap to help give your world character in a second way. A "Global Brush Map" can be thought of as representing a view of your world *behind* the camera. This brushmap is not directly shown by the camera, but can really help give your world character and realism. It does this by providing reflective objects something to reflect back to the camera. If you were rendering a desk in a room, you might have a global brush map that showed something as simple as a wall with a window in it. When rendered, the chrome desk chair will reflect that wall-with-a-window view, which makes the entire scene appear more accurate. As with the background map, this world reflection map can be animated by setting the "Max Seq." gadget to the number of frames in your sequence.

Global reflection maps are very important for reflective objects like metals. This is discussed in the appendix on page 166.

7.8.5 World Size

The global actor can have a size subrow added to it. This size subrow controls a *very complex and important* parameter that Imagine uses in its raytracing algorithm. When raytracing, Imagine assumes the world lies within a "box" in the Stage editor that ranges from ±1024 in X, Y, and Z coordinates. *If any of your objects are wholly or partly outside of this volume they will not be rendered, or will only be rendered partway when raytraced.* Imagine's raytracing algorithm makes the assumption that you've placed all of your objects within this imaginary box.

World size has no effect on scanline, flat, or wireframe renderings, only on traces.

If you design your world too large, you can change Imagine's default world size to keep everything inside of the world. If you add a size subrow to the Global actor, the X, Y, and Z sizes in the size timeline will change the size of the "renderable" world.

You can make your objects too big to fit in your world, but sometimes you can make your objects *too small.* If you have very small objects in your world, raytraces will work properly, but they may be *very, very* slow. A simple crystal ball on a table takes about 30 minutes to render on my A3000 when the table is about 1500 units across. But if I shrink the table and ball down to 15 units, and move the camera to get the *exact* same view, the raytrace takes *5 hours.* The output is identical, but Imagine's raytracing algorithm becomes very inefficient when the objects in your scene are very small.

World size is therefore a balance between keeping scenes small enough to render accurately, but large enough to render in a reasonable time. Unfortunately, you can waste a *lot* of time playing with world size getting things to work. I've found that even when the world size is set manually with the size channel, the trace times can be unpredictable and extremely sensitive. The method I use is to plan my scenes to fill the default world size (-1024–1024 on each axis) as fully as I can, and not to override the world size. Once you start adding a size to the world, it can take a lot of tweaking to get it to work out. Unfortunately, each time you test the size (by rendering) you have to perform a raytrace, a slow process even at a tiny resolution. The best way to avoid having this problem is to try to plan for not having it in the first place.

World size doesn't seem to morph well. But then again, you're doing some awfully weird things if you're trying to change its size linearly with time

If you set the world size to 0,0,0, Imagine will try to automatically determine the best world size. This is obviously very convenient, and most of the time you'll want to use this option. Manually setting world size is therefore more of an advanced technique since you can just use the automatic sizing. I have no idea why this isn't the default. It's only disadvantage is that it takes Imagine some amount of time (usually a few seconds) to determine the best world size.

A final tip: if you want to resize all of the objects in the world, just pick them *all*, including camera and lights, and scale them. The scene will render unchanged, since the camera and lights will have the same view of the objects. Textures will scale their size parameters as well, so the object appearances should be the same.

7.9 Camera Control

As discussed before, the camera is just another actor that can be positioned and aligned like another object. Unlike normal objects, however, you have several advanced control options that will have a special effect on the camera. This advanced control is really for very specialized effects; *beginners can safely ignore this section!*

When you own a real-life camera, there are many different settings for determining aperture, focus, and focal length. Imagine doesn't have an aperture or focus equivalent (its camera model is a "perfect" pinhole camera) but it does allow for moving the focal plane of the camera to zoom the view or get more of a perspective view.

One other option is to "change the film size." On TV, the images you see are a fixed width to height ratio (1.33 to 1). But in the movies, you typically see a widescreen image of 1.6 to 1, and big budget action movies often are shot in a "scope" aspect ratio (like Panavision^{TM}) which are actually at 2.0 to 1 or more. Though a bit tricky, you can get Imagine to output different aspect ratios to make "widescreen" movies.

7.9.1 Focal Length

A camera works by focusing light on a "focal plane," a position which the lens is designed to focus the image. Certain cameras can change their focal length, which has the effect of "zooming" the image. *This is* not *the same as positioning the camera closer to whatever you are pointing it at.* A zoomed image has less perspective to it; if you are far away, you are getting almost a flat perspective, whereas if you are right next to an object you get a severe perspective. This is used all of the time in movies; a camera will use a zoom to give the feeling of action at a distance, but without losing detail. Zooms are also useful for transitions, since you can use a zoom out to give the viewer a feel for the environment.

Focal length is set in Imagine by the camera size. The ratio of the X size to the Y camera size defines the focal length of the camera. If you decrease X or increase Y, you will zoom *in.* If you increase X or decrease Y, you will zoom *out.* If you try to make the ratio of X/Y too large, Imagine will warn you of a very wide angle view, but will let you continue.

Focal length control defiantly falls into the realm of the director; it is a creative decision that can add impact to the way you present images to the viewer. I've found focal length changes to work well for two effects. Transitions, introductions, and conclusions in animations can use a changing focal length over time to zoom in or out, and give a very camera-like feel to the scene. (You can key the sizes, the camera will obediently zoom smoothly in or out.)

The other use for focal length is setting up "widescreen" animations, with wide angle aspect ratios. This is described in the "Tips and Tricks" appendix on page 146.

Again, focal length is not a common setting to change, it is mentioned here for completeness. You can spend a lot of time adjusting the camera to get a minor effect.

7.10 Stage Manipulation of Action Timelines

Many options in the Stage editor affect the timelines in Action; in fact, I view the Stage as being a sub-editor of Action that allows you to interactively edit some of the timelines! Stage allows you to edit paths, render previews, and interactively determine positions, orientations, and sizes, but there really isn't anything it can do that can't be done from Action instead. Of course, Stage is still essential, since you want to *see* where you put your objects when you lay them out instead of just entering the coordinates; that's the real point of Stage.

Many of the commands in Stage affect the timelines in Action. Most of the effects have been already documented and some are obvious, but here is a list of what commands will change the timelines in Action. Remember, these are *Stage editor* commands.

Load Adds a new actor row, with an actor subrow (with the object filename) starting in the current frame and continuing to the last frame of the animation. It also adds a one-frame-key for position, alignment, and size for the actor in the current frame.

Add Adds a new actor row, with an actor subrow (with the light parameters, the path filename, or axis info) starting in the current frame and continuing to the last frame of the animation. It also adds a one-frame-key for position, alignment, and size for the actor in the current frame.

Rename Changes the main actor's name (*not* the filename of the object representing the actor.)

Delete Removes the actor and all of the subrows connected with that actor.

Position, Alignment and Size Bar If a key already exists for that frame (a bar ends on that frame), the key position, alignment, or size for that key is replaced with the new position, alignment, or size. If a bar doesn't exist, a new bar will be made with the end of the bar in the current frame and the beginning of the bar extending back to the frame after the last frame containing a bar. If a bar passes through the current frame, that bar's beginning is moved to the frame after the current frame, and a new bar is added ending on the current frame and extending back to the frame after the last frame containing a bar.

Move, Rotate, Scale, Transform, and Camera (Re)track If a key exists for the object (a bar ends on that frame) the key position, orientation, or size of the object is updated. If a key does not exist for the current frame, *no change will occur in any of the action bars.*

Chapter 8

The Project Editor

The Project editor is the first and last editor of Imagine. It is where you start new projects, making Imagine set up the directories for all of the files to go into. It is also where you end your projects, when you actually render your scene and assemble animations.

Projects are well named; they are a name assigned to a particular scene or animation you are working on. You can have several projects, but you only work on one at a time. You don't want to mix the frames from your flower animation with your Galactic Pirate battle scenes. A Project is a set of file directories that Imagine sets up to place all of the files associated with the scene or animation you're working on. The actual rendered frames and the "staging" file that defines your scene are put into your project's subdirectories, and you might store objects and brushmaps there as well.

A subproject is a way to define many ways of viewing and rendering a project. You might want to render quick test frames of your scene before you render a high resolution ray traced version. You can define different subprojects that keep rendered pictures filed apart from one another based on what resolution and options the picture was rendered with. This way you don't accidentally mix different types of pictures, plus you can quickly select from many different sets of pre-saved resolutions or rendering configurations, instead of having to re-define them each time.

When you are in the Project editor (you start there after you view the introductory picture, and you can use the Editor menu to get there from the other editors) you will see a screen similar to the one in Figure 8.1. It allows you to chose projects via a pull down menu, manipulate subprojects by the four gadgets on the top, render and control frames in a subproject by the group of controls in the center of the screen, and make and play animations by the six gadgets along the bottom.

8.1 Projects

Only one project can be open at once. The open project determines what scene you'll be editing in the Stage and Action editors, as well as what scene will be rendered. The current project is named in the title bar of the editor. The Project menu contains three options that allow you to make and select projects.

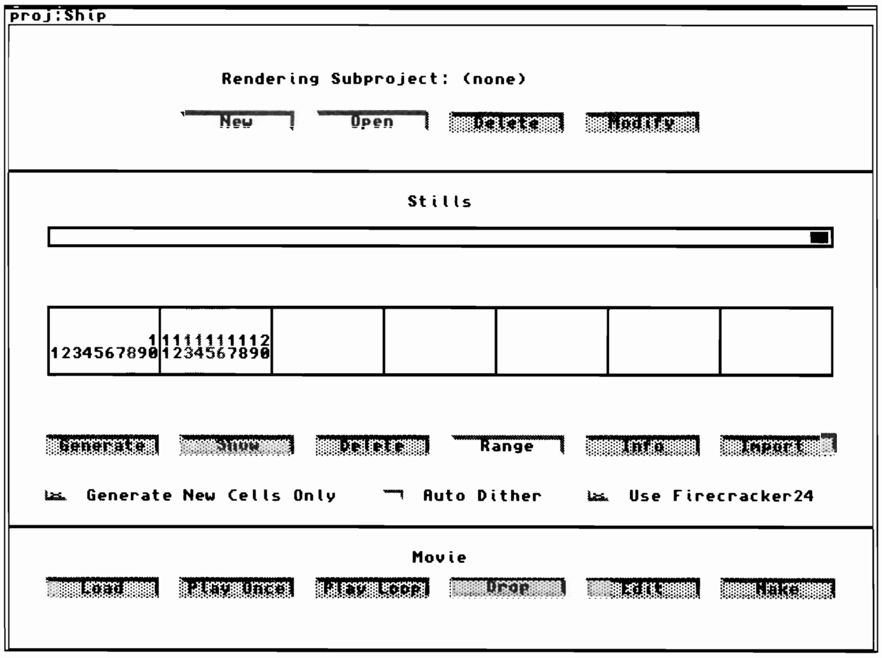


Figure 8.1: The Project editor

I **strongly** *suggest reading the tips on page 151 on making directories for your projects, or you will have trouble moving them or copying them later.*

The "New" command will create a new project. Imagine will put up a file requester and allow you to select a new project name and a directory for it to live in. Imagine will create a set of subdirectories in the location you choose, and define an initial empty world for you to edit in the Stage and Action editors.

The "Open" command will allow you to activate an old project, so you can edit it or render or view frames from it. Although the "Close" command will close a project, I have absolutely no idea why this option would ever be useful since you can just "Open" a new project without closing the current one if you want to switch projects.

These directories are described in the next subsection.

When you have an open project, many of the gadgets on the screen become unghosted, as you can now open and manipulate subprojects, and render and view frames.

8.1.1 Copying Projects

Actually, it is pretty easy to copy a project. If you use the AmigaDOS "Copy" command from the Workbench to make a copy of the project's subdirectory (the one with the *.imp* extension), you will successfully have copied your project. You can rename the copy to whatever new name you like, as long as it has that same *.imp* extension. Imagine will now recognize this new project *as* a valid project, and you can enter Stage and edit it without affecting your original project.

The trick on page 151 will help, but not cure, this problem.

Of course, if you have objects in the *objects* drawer of the original project, they will be copied as well. If you use these objects in the project, you'll see that they appear just fine in the new project as well. *But they are actually still the objects found in the* old *project!* Imagine has serious problems with the portability of the objects, textures, F/X, and brushmaps used in a project, since it stores all of the filenames using an absolute filename. If you want your copied project to use the objects in the *objects* directory of the *new* project, you'll have to enter the Action editor, and manually change the file name of each Actor.

8.2 Subprojects

As described in the introduction of this chapter, subprojects allow you to define sets of rendering parameters so you can have several different resolutions and rendering options for your pictures without mixing them up. You can have as many subprojects in a project as you like. For some complex, involved animations I've had as many as 10 subprojects to contain all of the different rendering styles I used.

To open an existing subproject, just click on the "Open" gadget. Imagine will open a file requester and allow you to select an existing subproject. The available subprojects are the ones that you've already created and now exist in your project's subdirectory. When a subproject is selected, any rendered frames are shown in the frame display bar in the middle of the screen and you are free to view and render those frames, as well as create and view animations.

"Delete" will erase a subproject and any rendered frames in it.

"New" will create a new subproject, including the subdirectory in the current project's directory. It will use a requester to ask for a new name, then a large requester will appear allowing you to set the subproject's parameters. This process is described in the next section.

"Modify" will let you change the parameters of the current subproject. A large requester will appear; the options in that requester are described in the next section. *When you modify the parameters of a subproject, all of the rendered frames in the subproject are deleted.* Often it is more appropriate to start a new subproject than it is to modify an old one, since you can have as many subprojects as you like.

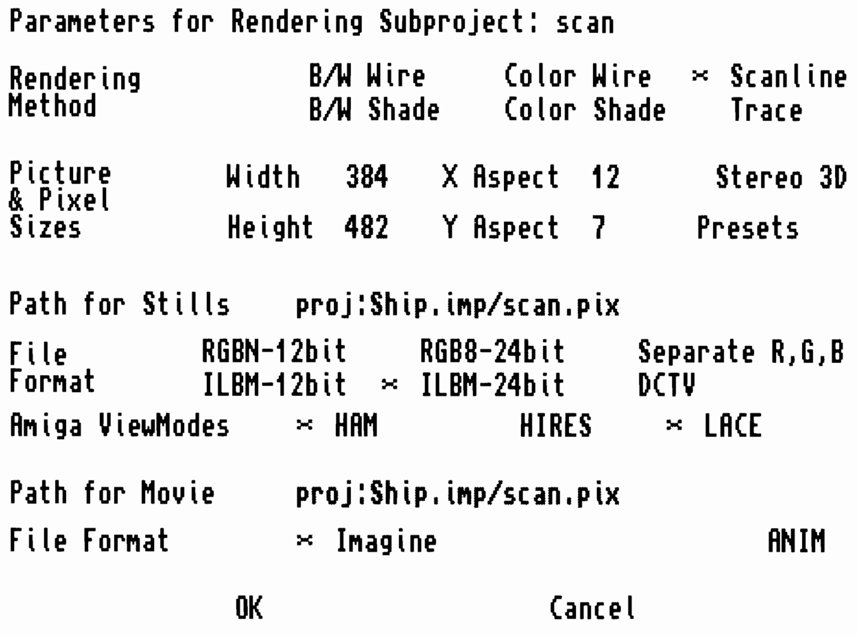
8.2.1 Rendering Parameter Options

Figure 8.2 shows the requester for defining the options of a subproject. These options allow you to change the size and aspect ratio of the frames, what rendering method will be used, the file format for the pictures, and the type of animation that is created.

The top of the requester is labeled "Rendering Method" and allows you to choose the algorithm Imagine uses for rendering your pictures. This is an important choice! Only one rendering method can be used at a time. The most useful two rendering methods are Scanline and Trace (raytrace.) The other four methods are mostly useful for very slow machines or for building animation previews.

Remember that all of the rendering methods render the same scene; the one that is defined (and previewed) in the Stage editor. The difference lies in the exact algorithm Imagine uses, which can *dramatically* affect the appearance of the rendered image, even though the same objects are shown in the same positions no matter what method is used.

There are also a few undocumented limitations that Imagine's rendering algorithms exhibit. There is a discussion of these problems in section B.13 on page 153.



Parameters for Rendering Subproject: scan

Rendering Method	B/W Wire	Color Wire	[✓ Scanline]
	B/W Shade	Color Shade	Trace
Picture & Pixel Sizes	Width 384	X Aspect 12	Stereo 3D
	Height 482	Y Aspect 7	Presets

Path for Stills proj:Ship.imp/scan.pix

File Format	RGBN-12bit	RGB8-24bit	Separate R,G,B
	ILBM-12bit	[✓ ILBM-24bit]	DCTV
Amiga ViewModes	[✓ HAM]	HIRES	[✓ LACE]

Path for Movie proj:Ship.imp/scan.pix

File Format [✓ Imagine] ANIM

OK Cancel

Figure 8.2: Defining a subproject's rendering options

Scanline

Scanline is the most often used rendering method. It can produce photo-quality images using a common computer graphics rendering algorithm. Your scene will be shown with all of the objects completely colored with textures and brushmaps, and faceted objects with Phong shading applied to them will appear smooth. Lights will correctly shade the objects, and the sky and stars will be shown properly. Transparent objects will be transparent.

Phong shading is described on page 32.

Setting sky gradients is described on page 113.

There are only a few features that Scanline will not provide. *Shadows are* not *rendered in Scanline renderings*, so the lighting can look a bit strange. Objects are still shaded, so if they face away from a light, they are darker, but you won't see an actual shadow cast by an obstruction between the light source and an object.

Though objects can be transparent, they will not refract light or images. For example, a crystal ball won't affect the image seen through it, and a diamond won't have the complex appearance you might be looking for.

The appearance of metals is talked about on page 166.

Though reflection isn't used in scanline renders, there actually is some performed. Shiny and reflective objects will reflect the global sky colors as well as infinite planes. This is a *great* advantage when rendering objects like metals which depend on their environment. Interestingly enough, when a reflected ground plane is shown, the textures and brush maps on the plane are properly shown in the reflection. This helps those checkerboard and chrome sphere renderings! Reflective objects will also reflect the images shown in the global world brush map.

Trace

Scanline is the most used rendering option since you get a realistic view of your world. However, raytracing provides a few extra benefits that can make it produce even more realistic pictures at the expense of extra rendering time.

Raytracing is much like Scanline in that it shows all of the objects in your scene with all of their textures and brushmaps, and they are smoothly shaded. Raytracing uses a completely different algorithm than Scanline, but for most objects their rendered appearance is nearly identical.

However, raytracing provides for true reflection and refraction. You can see the images of other objects in the side of a reflective object. A crystal ball will refract light like a lens and produce a distorted image of whatever is viewed through it.

Also, shadows are completely rendered, so you can have much more realistic lighting. The shadows that are cast are particularly "sharp" (they have distinct outlines) but they are accurate and look quite good.

Raytracing's biggest problem is speed. Because the algorithm that produces accurate reflections and refractions takes a lot of extra computation, it is particularly slow. A scanline rendering of a the exact same scene might take anywhere from 10% to 90% of the time a raytraced version would take.

B/W and Color Wire and Shade

The Wire rendering methods produce a similar output to the "Wireframe" preview window shown in all of the editors.

The Shade rendering methods are much more useful, since your scene will be will be shown as opaque solid objects like it should. The objects will also be lit by the lightsources that are placed in your world, though this lighting is a very simple algorithm that makes each triangle very visible. Shadows and Phong shading are *not* used, and although your objects are colored correctly, no brushmaps or textures are applied.

There are really only one reason you would use Shade (or Wire.) They are both very simple rendering methods so they are *very* fast. You can render them in just a few minutes on a 68000 based machine like an A1000 or A500, and a few *seconds* on a 68030 based machine. This speed makes them bearable to use on slow machines. For faster machines, a shaded version of an animation renders in a reasonable time (perhaps a half an hour for a complex animation) so you can make full-screen preview animations quickly and easily.

8.2.2 Picture Sizes

The next part of the subproject requester allows you to set the resolution of the rendered images: exactly how many pixels wide and high your image is. Unlike many other renderers for the Amiga, you can set the rendering resolution at any level you want; 100 by 100 for really quick previews, or 7694 by 6809 for your masterpiece work that will be displayed on a one-of-a-kind video monitor. Actually, the max size is 8192 by 8192.

Resolution	Interlaced	Overscan	X Size	Y Size	X Aspect	Y Aspect
Low	No	No	320	200	6	7
Low	Yes	No	320	400	12	7
Low	No	No	384	241	6	7
Low	Yes	No	384	482	12	7
High	No	No	640	200	3	7
High	Yes	No	640	400	6	7
High	No	No	768	241	3	7
High	Yes	No	768	482	6	7

Table 8.1: Aspect ratios of common Amiga display modes

Most images you render will be at standard Amiga resolutions, like 320 by 400 for low resolution interlaced, or 704 by 440 for a hires interlaced image with some overscan.

You can just enter the width and height of your image (in pixels) into the gadgets labeled "Width" and "Height."

The next two gadgets let you define the shape of the pixels that are rendered. On the Amiga, the pixels that are displayed on the screen are not square! In fact, they can change size depending on what resolution the screen is. A pixel in a low resolution interlaced image is almost twice as wide as it is high (a 12 to 7 ratio). A high resolution interlaced image has pixels that are half as wide, so the pixels are in a 6 to 7 ratio. If you are rendering an image to display on a Macintosh, you probably want to use square pixels (a 1 to 1 ratio.) The "X Aspect" and "Y Aspect" gadgets let you define the pixel shape. *Only the ratio matters!* X and Y aspect values of 6 and 7 are exactly the same as values of 12 and 14. There is much more discussion of aspect ratios in Section B.4 on page 146.

Figure 8.1 shows the aspect ratios of the standard Amiga graphics modes. Note that overscan displays have the same aspect ratio. Overscan doesn't change the shape of the pixels, it just shows more at the edges of the screen. To determine the aspect ratio of a new display (say of another computer) find an image that it displays and form the ratio

$$\frac{\#\ Y \text{ pixels}}{\#\ X \text{ pixels}} \cdot \frac{\text{Physical}X\text{Size}}{\text{Physical}Y\text{Size}}.$$

This is the X/Y pixel size ratio. For example, an Amiga lowres interlaced image is 320 by 400, and displays on a monitor with physical dimensions of 48 by 35 (close to NTSC 4/3.) $400/320 * 48/35 = 12/7$, the aspect ratio listed for lowres interlaced images.

Stereo 3D

The "Stereo 3D" button will make a very special picture with every other line of the image being rendered from one of two viewpoints separated by a short horizontal distance. If you view a picture in this format with a pair of LCD shutter glasses (like Haitex's X-Specs) you can see your image in 3D.

Presets

You can define presets, saved settings of resolution and aspect ratio, that you can quickly load by selecting this button. By clicking on one of the presets, you can load in the resolution and aspect settings associated with a preset.

You can make your own saved presets by using the Preferences editor.

The presets made by Impulse list 90% of the modes you would ever want to render in. Their overscan resolutions are only "partial" overscan, however; they don't extend as far as the Amiga is capable of handling. Table 8.1 lists the full overscan resolutions of the Amiga.

8.2.3 Frame Options

The default location for rendered frames is in a subdirectory in the project's directory. You can specify an alternate directory by entering a new directory name in the "Path for Stills" requester. Usually there is no need to change this option.

You can choose what file format pictures are saved as. Impulse has its proprietary "RGBN" and "RGB8" formats, but you can also save in standard IFF format or as a set of raw RGB files. Although the default is to use "RGBN," I strongly recommend that you use "ILBM-24 bit." This is the standard in the Amiga community for 24 bit files, and you might have trouble using files in the Impulse format in other Amiga programs.

The "RGBN" and "ILBM-12 bit" options will save a 12 bitplane (4096 color) version of your pictures. Although they take roughly half the size of 24 bit pictures, I still recommend using "ILBM-24 bit" since the 12 bitplane options are a lower quality file format.

The raw file format might be useful if you need the raw bytes that make up your image. Imagine will save three files containing the red, green, and blue bytes of information in your picture.

If you select "DCTV" as a file format, images will be shown in the special format that can be displayed on a DCTV. This is a 4-bitplane overscan image; there isn't any way to tell Imagine to use any other type of DCTV format. (Grrr...)

The DCTV is a neat piece of hardware. It is talked about on page 190.

You can tell Imagine what Amiga screen modes to use when images are displayed. You can use the three gadgets at the bottom of the screen to toggle on and off high resolution, interlaced screens, and HAM display.

8.2.4 Animation Options

The default location for animation files is in a subdirectory in the project's directory. You can specify an alternate directory by entering a new directory name in the "Path for Movie" requester. Usually there is no need to change this option.

When Imagine makes an animation, it makes it in one of two formats. The "Imagine" format is default, and allows you to view the animation from Imagine and edit a script to play the frames in arbitrary order.

The "Anim" format is the standard Amiga animation format, which can be shown by most Amiga animation programs, like my favorite, "Playanim." Imagine *cannot* directly play this format of animation from within the program.

8.3 Rendering and Manipulating Frames

When you have an open subproject, any frames that you have rendered will be shown in the long middle display in the center of the screen. The different frames are labeled and increase in frame number from left to right. If there are more than 70 frames, you can scroll through all of the frames by using the large horizontal scroll bar. Rendered frames show up as an asterisk ('*') under the frame number. You can pick frames by simply clicking on the frame number with the mouse. By holding the shift key, you can pick multiple frames.

You can pick (or unpick) a large number of frames by using the gadget labeled "Range." Imagine will ask for a starting frame, an ending frame, and a step (how many frames to skip between each picked frame) which will define a range of frames to pick. "1,1,10" will pick the first 10 frames, "5,2,13" will pick frames 5, 7, 9, 11, and 13.

When you've picked one or more frames, you can choose to perform a variety of operations on those frames. The "Generate" button will render the frame(s). If you have the "Generate New Cells Only" button pressed, a frame will *not* be rendered if a picture already exists for that frame. To render a replacement frame (if you've updated your scene, for example) either turn off the option or delete the pictures first.

Sort of a short description for the command that culminates the hours of work that went into creating and planning your scene or animation.

"Show" will display a previously rendered frame. It will be shown in the Amiga viewmode that you have specified in the subproject definition. If you have the "Auto Dither" gadget selected, Imagine will use dithering when the image is displayed. Dithering tries to alternate different colored pixels to better represent the true image color, and almost always results in a better looking image. If you have the "Use Firecracker 24" button set (and you have a Firecracker!) the image will be shown on the Firecracker in full 24 bit color.

"Delete" will delete the selected frames. If you delete frames using AmigaDOS (by deleting the actual picture files manually,) Imagine gets a little confused but will straighten itself out after it complains with a pop-up requester.

The Firecracker is a 24 bit display board, discussed on page 190.

"Info" will display useful information about the rendered frames, including the picture's file size, the date and time it was rendered, and how long it took to render.

"Import" will allow you to import a picture as a frame in your animation instead of rendering a frame. All this option really does is mark the frames as already being rendered; you should use AmigaDOS to put a picture file into the subproject's subdirectory. This option is most useful when you want to archive rendered pictures then restore them later.

8.4 Animations

You can build and view animations of your project right from Imagine. Each subproject can have an animation defined from the frames that have been rendered. You can build an animation using all or part of the existing frames. Animations (or "Movies") are controlled using the gadgets at the bottom of the project screen.

To make an animation, just pick the frames you want (usually all of them) then click on the "Make" option. Imagine will ask you several questions, allowing you want to use the picked frames to make the animation, whether to delete the frames it uses to make the animation, whether you want a looping animation, and whether to lock the palette.

A locked palette will mean that all of the frames of the animation will use the same colors. This is an advantage in some programs that cannot handle different palettes for each frame, though overall each frame won't be as well represented since they have to share the same colors.

A looping animation just means that it repeats when finished. Though you can loop any animation, there will be a skip at the end of an animation that you haven't told Imagine that it should repeat.

You can opt to have Imagine delete frames after they've been added to the animation. This really saves disk space, but the frames will be gone. I don't do this unless I'm rendering quick test animations.

After you've told Imagine to build the animation, it will work for a while; it will display what frame it is working on as it displays them. You can also use the screen gadgets to move the Imagine screen to the back to so you can watch Imagine show the frames; this at least gives you something to do as Imagine works.

When the animation is done building, it is saved as a disk file(s). To view it (any time after it is rendered) you have to load it into RAM by using the "Load" gadget. *Anims take a lot of RAM!* You can easily build an animation that is too big for you to load and view!

After the animation is loaded, you can view it once by using the "Play Once" and see it loop indefinitely by using the "Play Loop." You animation will be displayed in all of its glory. Press the mouse button or the ESC key to exit from a looping movie.

The "Drop" command will remove an animation from memory, freeing the RAM that it was using.

8.5 Project Directory Structure

This is almost an appendix to the Project editor; but it is sometimes useful to know where all of the files Imagine creates go. You actually don't need to know the details unless you are moving projects around or trying to free space on your hard drive.

A project creates a directory with an extension of *.imp*. This directory contains a single file named *staging*, which contains all of the information about your scene, like the object locations and filenames. There is also a subdirectory named *objects* which begins empty, though you are free to put objects into it. Imagine never uses this directory directly.

Each subproject has its own subdirectory with an extension of *.pix*, so a subproject with a name like "scan" would have a subdirectory named *scan.pix*. In this subdirectory is a file called *specs* that contains the subproject information, like the rendering resolution and file format. Rendered frames are saved in this directory as files named *pic.0001, pic.0002, pic.0003* and so on. The name "pic" is hardcoded, it never changes. The number that forms the extension is the frame number of the picture; obviously frames can't be larger than 9999.

A subdirectory in the subproject (yes, another level!) contains animations. This subdirectory is named *anim* and contains a file named *script* which contains information on the animation. Animations frames are saved as files called *anim.0001*, *anim.0002*, and so on.

Imagine really doesn't create any other files anywhere except objects, where you choose the location, and Quickrender pictures, which default to being stored temporarily in RAM:.

8.5.1 File Naming Conventions

An important suggestion; use descriptive names and extensions for any files you use in Imagine. The file *obj1* is going to mean nothing to you an hour from now. *tablecloth.iob* tells you that this is an Imagine object of a tablecloth; a useful description. I use the extension *.iob* to stand for Imagine objects, and *.pth* for spline paths. I label brushmaps and pictures with an extension depending on how many bitplanes the file has. *sunset.iff24* is a 24 bit IFF, and *exploding-slurpee.iff2* is a 2 bit plane (4 color) picture. I also label HAM pictures with the extension *.ham.* If you don't use these extensions, you won't get into trouble, but I find that if you stay consistent you can save yourself a lot of work in the long run.

Chapter 9

The Preferences Editor

Imagine has many default parameters that you can change. These parameters let you set a wide variety of options ranging from the screen colors shown in each of the editors to the number of multiple reflections that are rendered in Trace mode. You can also add custom new user gadgets at the bottom of each editor's screen. The rendering presets that are used for defining subproject screen sizes are also definable.

Imagine 2.0 has added a new editor just for modifying all of these user-changeable parameters. You can change global variables, set up function keys and user gadgets, and build default rendering presets.

The Preferences editor can be called up from the Editor menu in any other editor. The gadgets at the top of the screen let you select which type of information you want to view or edit; just click on the appropriate gadget and you will go to that display. Each of the screens is discussed in their own section in this chapter.

You have several options when you're done making your changes. The gadgets at the bottom of the screen allow you to tell Imagine what to do with any changes you've made.

If you want to save the current configuration, you can click on the "Save" gadget which will save all of the parameters and make them the default configuration every time you use Imagine in the future. (This default configuration is contained in a file called *imagine.config* if you ever want to edit it manually or copy it.)

The "Use" gadget will not save the current configuration, but will instead let you use the program with the temporary settings. You will be exited from the Preferences editor, and the changes you made will take effect, but *they are not saved* so the next time you run Imagine you'll go back to the old default setup. This is useful when you want to test an option, or just want to change a variable for a single rendering or something. Remember it's just a temporary change!

The "Cancel" gadget will abort any changes you've made, and exit you from Preferences. Your configuration will be unchanged from the configuration you had when you entered Preferences. When you screw up and don't know how to fix it, "Cancel" will usually save you.

"Reset" will change all of the parameters back to their original, Impulse-set values. If you *really* screw up and save the bad configuration, "Reset" will get you back to a reasonable setup.

"Last Saved" will load the last configuration you've saved. Any changes you've made will be overwritten by the new values.

The last two commands let you have special configurations you can name and call up at need. You can save a configuration with the "Save As" gadget with a special name. The default configuration will remain the default, but you can read the alternative configuration with the "Load From" gadget. For example, I have a configuration file called "Manual" that changes all of the editor colors to greyscales to make including screengrabs in this book easier. This is a configuration that I don't really want when I'm rendering, but I don't want to have to define the greyscale colors every time I need them.

The actual parameters you can change are all described in the following sections. The Preferences editor is rather straightforward and easy to use since it's really just a quick utility for number editing.

9.1 Global Parameters ("Misc Stuff")

The different parameters are all described individually, but they are all viewed and changed in a similar manner. You can click on a horizontal row to highlight it and indicate your want to edit it. The information in that row is copied to the gadgets at the bottom of the screen. You can just type in the gadgets to change the values of the presets.

Colors are specified by three numbers which indicate the R, G, and B components of the color. There is a set of sliders on the bottom right of the display that you can use to choose colors. *These sliders do not set the parameter's color.* They are a tool to tell you what numbers to type in. Sure, it's annoying, but better than guesswork.

Be careful with your mouseclicks; it is easy to click on an empty part of the screen which deselects the row you are editing.

The following are the parameters you can set. Each one has a four letter identifier.

PPTH Picture Path. The pathname to the *Imagine.pic* file. Imagine won't run without this picture file. Most people will have the file in the same directory as Imagine, so the PPTH variable is "", but floppy users might want to move the picture to another floppy to save space.

EDIT Editor. The text editor used to edit the script for "Make Movie" in the Project editor. You can use any editor you wish, like *memacs* which comes with AmigaDOS.

QUIK Quickrender style. This is the screen format that Imagine uses for displaying quickrenders.

QURM Quickrender Render Method. You can set the type of Quickrender image that is produced by naming one of the rendering presets like "HAM" or "Firecracker 384" or any other preset that you have already defined.

QUFF Quickrender File Format. You can choose what filetype is used for the image, like "RGB8-12bit" or my preference "ILBM-24bit".

QPTH Path for Quickrender picture. Where the Quickrender picture will be saved. It defaults to RAM: which is good for most systems, but if you have little RAM, you might direct the file to a disk. These files are usually deleted quickly.

LOAD Load all modules? For machines with little RAM, you might not to load all of the editors of Imagine at once. Instead, they will be loaded as needed, saving RAM, but causing pauses when you move from editor to editor.

LACE Interlace editors? You can choose to use an interlaced display or not; you can override this in most editors by the "Interlace" command.

GRON Grid lines on? Sets the default grid display state. Overridden by the "Grid on/off" command.

BWLN Make lines in "B&W Shade"?

COLN Make lines in "Color Shade"?

WARN Warn about unsaved changes?

BGRD Background color.

FGRD Foreground color.

REQC Requester and gadget color. Also used for bounding box and drag points.

BVLB Bright bevel color.

BVLD Dark bevel color.

GRID Grid color.

PICK Picked edge color.

PPNT Picked point color.

SPCK Selected picked edge.

SPPT Selected picked point color.

BWLC Line color in "B&W Shade."

COLC Line color in "Color Shade."

STAR Star color for stars added in the Action editor's Global requester.

COL0 Color 0 value. Used to determine what to use for "genlock" colors. Usually set to 0,0,0.

QSKY Quickrender background color.

GENC Genlock Color. Very important, since backgrounds will be rendered in this color, a fact crucial not only for genlocking but for composition of pictures in a program like ADPro.

NUMS Number of screens for "Make Movie." Set it to 2 or maybe 3. No big advantages.

EDLE Antialiasing level. **Perhaps the most important parameter in Preferences.** I wish this were an option in each subproject. Antialiasing reduces the jagged look of diagonal lines by making multiple samples of each pixel. The *lower* the number, the better the quality (less jaggies) in your image. A value of 0 is the best quality (most oversampling.) A higher number increases jaggies, and also increases rendering speed. For quick test passes (especially in raytrace,) you can increase thisΥÆ 7this number to 200 and your rendering speed will increase dramatically. A value of 30–50 works well as a "general purpose" setting, and a value of 0 should be used for your masterpiece.

RSDP Resolve depth for multiple reflections or refractions. When you are raytracing a complex scene, you might have reflections of one object off of another. If both objects are reflective, you might have multiple reflection where light is reflected from object to object to object to object. You can set the maximum number of multiple bounces by changing this value. This really only has an effect on complex scenes with lots of reflective or transparent objects.

OCTD Number of Octree levels. The Octree is an algorithm Imagine uses to speed rendering by sorting objects into bins to help prune the number of objects that it has to check to see if a ray hits it. The larger the octree, the faster rendering will be, though the setup time will increase and more RAM will be used to store the sorted objects. If you have a simple scene the setup time might be very long and you might want to decrease the number of levels.

GNDN # of divisions in ground. In the preview windows, you see grounds as grids. You can change the density of this grid with this parameter. I like the default of 16.

GNDS Ground Size. Grounds are infinite, but you can limit how much you see of them by setting this number. The default of 1024 is good.

SCRL Scroll percentage. How much the four arrow keys move your view in the world. 0.33 works best for me.

GSIZ Initial grid size. The default grid size, can be overridden in the editors by the "Grid Size" command.

EYES 3D Stereo Eye separation. This controls the camera displacement that Imagine uses to form the left and right views for stereo pictures. Values from 0 to 1 are decent choices, with different numbers causing different offsets. I use a number of .4.

OTRL Max Octree RAM. The octree partitioning can take a lot of RAM. You can set a maximum amount it uses using this parameter. See OCTD.

OTFL Min Free RAM after Octree. Instead of placing a top bound on the octree size, you can tell it to be as large as it likes as long as it leaves at least this much RAM. See OCTD.

9.2 User Gadgets and Function Keys

As described in the Basics chapter, most of the menu commands have keyboard equivalents. For example, right-Amiga-i will execute the command "Zoom In" giving you a magnified view of a smaller area of your world. Not all menu items have keyboard equivalents, and sometimes the default equivalents are poorly chosen or hard to remember.

To help you customize your control of Imagine, you can define your own set of keyboard commands with the top row of function keys on the keyboard. Additionally, the gadgets on the bottom of the screen in most of the editors allow you to set up a "mouse" shortcut for selecting a menu item without using a menu.

In addition, both the function keys and the custom user gadgets can be different for each editor. The F1 key could be defined to mean "Pick Select" in the Detail editor and "Go to First Frame" in the Cycle editor.

Both the function keys and the onscreen user gadgets are defined the same way in the Preferences editor. To edit the function keys, select the "Function Keys" gadget on the top row of the Preferences editor, or select the "User Gadgets" gadget to edit the custom gadgets. Choose which editor you want to define the shortcuts for from the second row, either "Detail," "Forms," "Cycle," "Stage," or "Action."

You will see a list of the currently defined keys or gadgets appear. You can click on a row to edit a key or gadget's meaning. The row will highlight, and you can use the gadgets at the bottom of the screen to change the current settings.

Function keys are numbered 1–20. The numbers 9–20 represent the *shifted* value of the function key, that is, 16 is shift-F6. The function that a key stands for is specified in the "Menu #" column. This shows a somewhat arcane way of representing a menu item by using the digits to represent which menu, which selection, and which sub-selection the command you are defining is located. Luckily, setting the value is much easier. If you start to use pull down menus, you'll see that they have been transformed! Instead of a pull down menu for the Preferences editor, you'll see a copy of the menu bar for the editor you are specifying shortcuts for. To select an item for the function key to represent, just choose the proper pull down menu selection just as if you were trying to execute the command.

You can also enter a comment in the last field, which is handy when you can't remember what "6a0" means in the Detail editor. Impulse has already set up some reasonable defaults (with the oft-used "Pick Select" on the best key, F1) and commented them so you can see what they are.

Custom gadgets are chosen similarly. You highlight their row, and define the functions by using a mock pull-down menu. The one additional control is the gadget's text. The custom gadgets can have whatever text you like written on them, from one character to 50. However, you are limited to the total number of characters that *all* of your gadgets have (there is only so much room on that thin bar!) so be brief. Three or four letter commands seem to work well for me. You can obviously edit the Impulse defaults and shorten (or lengthen) their names.

To add a new custom gadget, you just select the last row (which is labeled <*new*> and edit it like a current gadget. To delete a gadget, enter "000" into a row's "Menu #" gadget.

9.3 Rendering Presets

The use of presets and an explanation of the rendering options is discussed at length on page 123 in the Project chapter.

If you do render magazine covers, watch the aspect ratios! See page 146.

The render presets that are used in quickly defining subprojects can be edited and defined from the Preference editor. By selecting the "Rendering Presets" from the top row of gadgets, you will see a display of all of the current presets. Impulse has a nice selection of presets that cover most resolutions you might want to render at often, but, for example, you might often output magazine covers at a resolution of 2048 by 3480 and want to call up subprojects with this resolution quickly.

You can edit old presets by clicking on a row then entering or changing the appropriate information in the gadgets at the bottom of the screen. To make a new preset, select the row appropriately labeled <*new*> and edit the values to make a new preset. To delete an existing preset, just enter a value of "0" for the width and height.

Appendix A

Common Problems

There are certain problems and questions that continually crop up with all Imagine users. All of these questions have been addressed in the text, but this quick answer section will give you a quick answer and point you towards a more detailed reference. Some problems and bugs from Imagine 0.9–1.1 are discussed as well as whether they have been fixed in Imagine 2.0.

- How can I speed up my rendering? It's taking *way* too long.

  Beyond the obvious solution of getting a faster machine, there are a lot of parameters you can tweak. Using the Preferences editor, you can edit the EDLE parameter (Edge Level) to a higher number. This number controls antialiasing, how each pixel is sampled many times to get a good estimate of its true value. A low number like 0 is very high quality, but is very slow. 35 is the default. A value of 100 can speed up renders by 50%, but the quality of your pictures will go down.

  If you are always running out of memory, you'll find that adding more RAM will also speed up rendering. This is because in tight RAM situations your machine will start using slow Chip RAM instead of the faster Fast RAM. On a 2 Meg A3000, this effect is dramatic, since it has 512 K of Fast and one meg of Chip, Adding just one extra meg of RAM will easily *double* rendering speed of a 2 Meg A3000.

  In trace mode, rendering speed is linked to your object size; very very small objects take longer to render if the world is much larger. You want to scale your objects to fit into your world, but take up a good fraction of the world size. If the world size is 2048 wide, rendering a 1 unit high object can take *ten times* longer than a 500 unit high version, despite the fact that it's the same view of the same object.

- Is it all right leave my computer on overnight to render? How about a few days?

  This is occasionally a lively debate. It certainly is all right to leave your computer on even for long periods of time (even *months.*) Many people even think that leaving a computer on actually *decreases* wear since powerups are very stressful on the computer hardware. Hard drives in particular are probably good to leave on, especially Seagate drives which often fail to start spinning when they are initially turned on.

If you do leave your computer on (I do, all of the time), you probably should turn off the monitor when you aren't using it. There is a slight fire risk if a monitor fails, and unattended the damage might be severe. Also, if the power service in your area is erratic, you might have trouble with the computer freezing during one of the power glitches.

- Why do objects get chopped in half or not appear when I raytrace?

  Imagine has a "world size" that objects must stay within when they are being raytraced. This "world" is the volume in the Stage Editor that extends from X= -1024 to 1024, Y=-1024 to 1024, and Z= -1024 to 1024. If you have any objects outside of this volume they will *not* show up when rendered in Trace mode. You can either move and scale your objects so that they are all within the world volume, or you can increase the world size manually. This is done by going into the Action Editor in Stage, and using "Add" to add a *size* channel to the global actor's timeline. For a size, just type in the size you want the world to be; default is 1024. Note that lights and the camera *can* be outside of the world volume, only objects must stay inside. Also, the world size does *not* affect scanline rendering, just trace.

- How can I move projects onto another disk, or give them to a friend? Imagine can never find the objects when I move them anywhere.

  Imagine uses hardcoded paths to assign object and brushmap names. There really isn't a solution, but I talk about one way to reduce these problems by using path assigns on page 151.

- Imagine doesn't give me an "out of RAM" error, but I still get funny results like missing objects when I'm pushing my RAM limits.

  Imagine sometimes runs out of RAM but keeps merrily chugging along. Most often, an object will be left out or only partially rendered. If you have missing objects like this, watch your memory meter as the render sets itself up and then begins rendering. If it gets low (less than 250K of Fast RAM) you might be running into RAM problems. Try using smaller brushmaps or some of the other RAM saving techniques listed in this appendix.

- How can I conserve memory? I'm running out.

  First and foremost, try to keep from using lots of big brushmaps. Many times a smaller version of a picture will produce just as good an image, and will save a *considerable* amount of memory. Textures in Imagine don't use much RAM; you might try to add details with a texture rather than a custom brushmap. Another way of saving memory is never to have the animation builder automatically generate frames; RAM is wasted by keeping previous frames in memory as the new one is rendered. First, generate all of your frames, *then* build the anim. This can easily save 300K of memory over doing it the other way.

If you are scraping for memory, you can get about 80K by not running workbench. Make a floppy disk that has a modified Startup-Sequence that does *not* include the command *LoadWB* but instead runs Imagine with a command like *work:Imagine/ImagineFP* if Imagine lives on your work: drive. However, if you do a lot of rendering, you'll find that you can never have too much memory; it's always useful and these days it's pretty cheap. Artist Brad Schenck once said "If you aren't always running out of RAM, you're not trying hard enough."

- How do I get out of viewing a picture in the Project Editor?

  There is a bug in Imagine 0.9–1.1 which will freeze the displayed picture on the screen if you press the mouse button. This bug has been fixed in 2.0, but if you are using an earlier version, to get back to the Project editor after looking at a picture or watching an animation, press the ESC key. *if you accidentally press the mouse button first, the ESC key will* not *work!* If you do this by mistake anyway, you can pull down the picture by grabbing the top of the screen, exposing the Project Editor screen beneath. If you click on the exposed Project screen, *then* press ESC, the picture will then disappear.

- I move my objects around in Stage, and the scene looks great! But when I change to a new frame or when I render my animation, no matter how often I use "Save Changes" the objects are in the wrong place!

  When you move an object in the Stage editor, many times your change will *not* be recorded because you have actually told Imagine not to allow changes to that object. How? *The only time manipulation in Stage will "take" is during the last frame of an object's position, alignment, or scale timeline bar.* If you want to override the position bars, add a new bar using the Position Bar command from the Object menu.

- Why won't the brush mapping axis stay where I position it?

  There is a bug in Imagine 0.9–1.1 that does not keep the changes you make to the size, orientation, and placement of the brushmap axis when you interactively edit it. (You will manipulate) the brush, but if you go back to it, you'll find that the brush hasn't moved. The easiest way to get around this bug is to use *local* mode when manipulating the brush. Press "l" before you move, rotate, or scale the brush axes, and the changes you make should be remembered. This bug does not affect the Transform requester.

- How can I increase the quality of my Imagine HAM pictures?

  If you are using HAM to display your pictures, make sure that you select "auto-dither" before viewing the picture. Dithering will dramatically increase the color resolution of most pictures, though the HAM picture will be slower to display and the file size of the HAM picture will be larger. For the best HAM display, the Art Department by ASDG can load 24-bit pictures generated by Imagine and display them with noticeably better dithering than Imagine.

- How do I join the Imagine Mailing list?

  The Imagine mailing list is a group of Imagine users on the nationwide computer network called the Internet. If you have access to this network, you can get on the list. The list consists of postings from the list members talking about all aspects of 3D in general and Imagine in particular. If you are able to send and receive electronic mail on the Internet, mail *spworley@athena.mit.edu* and ask to join the list! That is all there is to it. Even if you do not have Internet mail access, many BBS's carry the list as well, though you might not be able to post. Portal on-line services has a complete feed of the mailing list. If you don't know what the Internet is, you probably don't have access to it.

- Sometimes I type a number in a gadget and hit "OK" and Imagine doesn't use that number! The gadgets just don't work sometimes!

  Some gadgets in Imagine will not register a number you input into them unless you hit the Return key in the gadget before clicking on the "OK" button. It works in some gadgets and not in others, so if you are getting strange results, try carefully using the Return key to see if the numbers are truly "getting though."

- What is the difference between line segment paths and spline paths?

  This question is moot in Imagine 2.0, since *only* spline paths are used. But in Imagine 0.9–1.1 Imagine uses two different types of paths, line segments and splines. Line segment paths are just an object that consists of a set of points linked by edges. The order of the points defines the direction of the path. To make this type of path, it is easiest to add a new axis in the Detail Editor, then enter "add lines" mode. Whenever you click, a new point and edge will be added where you're pointing, and you can trace your path in just a few seconds. This type of path is used for extruding along paths as well as the "Grow" F/X.

  The other path type is splines. In Imagine 0.9-1.1, these paths *cannot* be used in the Detail Editor, they are just for defining routes for objects to move along in the Stage Editor. Both open and closed paths are available; open paths are lines that define a route from a starting point to an ending point; closed paths always start and end at the same place, which is useful for cyclic motion or looping animation. You can create a new spline path by selecting "add path." The paths are interactively editable in the Stage Editor.

- How do I extrude to a path? I get an error, or a straight line.

  First, it is important that you use the right type of path. Imagine 2.0 uses spline paths for extrusion, but in versions 0.9–1.1 "Extrude to path" uses a series of connected line segments to define the direction the extruded object follows. For these early versions of Imagine, Extrude does *not* use the spline paths that are used in the Stage Editor. If you try to extrude and get an error message, you are using the wrong type of path- use a line segment path.

  When you've created a path to extrude along, use the attributes requester to see the name of the path; you might even want to call it *PATH* as opposed to *AXIS.5* or whatever it happens to be called already. Pick the object or outline you want to extrude, and get to the extrude requester by selecting "Mold." Toggle the extrude to path button, which will unghost the path name box. Type in the name of the path that you noted before. Also, make sure you change the "# of segments" box to a number greater than one! Your extrusion will be of better quality the higher you set this, though your object size and complexity will also increase. If you just get a tube that doesn't follow your path, set the # of segments to something greater than one.

- Why doesn't an altitude brush make a dent in my object?

  An altitude map just tells Imagine that light hitting the object's surface should be reflected, refracted, and specularated (!) as if it hit a surface that had a certain shape to it; the shape described by the brushmap's intensity. If you mapped a picture with lots of small fuzzy grey dots onto a sphere, you would get reflections and light highlights as if the sphere had tiny pits in it like an orange.*The altitude map does not change the real surface height of your object at all.* This is the difference between a displacement map and an altitude map. Nonetheless, this is a powerful effect, as making objects like golf balls and oranges, as well as complex surfaces like a turbulent ocean and a mottled brick wall are all easily possible.

  The altitude mapping in Imagine 0.9-1.1 has a major bug which reduces the apparent depth of the map, but it works gloriously in Imagine 2.0.

- What size should I make my objects? 1000 units? .1 units?

  The size of objects is not critical, as they can always be re-scaled in the Stage Editor to whatever size you wish. However, it is best to try to keep your objects on the order of a hundred units wide. If you start making a detailed object that is .1 unit wide, you might start hitting the limit of Imagine's spatial resolution (how accurately it can place a point) which is 1/65536 of a unit. Also, if you make the objects too large, you might run into the absolute size limit (65536 units) and you might have problems in raytracing if the object is outside of the world size.

- I can't make transparent glass. Help!

  Objects with any shininess set in the attributes requester will not be transparent. You must set the shininess of any glass to *zero*, and the filter values moderately high. This is actually not a bug, since the filter controls are really "appropriated" by Shininess to control the color and appearance of the "shine."

- What is the difference between flat-flat, flat-wrap, and wrap-wrap wrapping?

  There are three basic types of wrap- a "flat" wrap (Flat X Flat Z), a "sphere" wrap (Wrap X Wrap Z), and a "cylinder" wrap (Flat X, Wrap Z and Wrap X, Flat Z). Flat will ignore any surface bumps and features and just apply itself directly, much like a slide projector would project onto a bumpy screen. A sphere wrap tries to encase the object in the brush, then shrinkwrap the map onto all of the surface features of the object. The cylinder wrap tries to follow contours in one direction, but ignore them in another. Think of taking a piece of gift wrap, and bending it around so its a hollow cylinder. Then place the object in the center of this vertical gift wrap cylinder and push *in* (but not up or down!) to follow the object contours.

- I can't make objects that look like they're metal. Why not?

  There is a long discussion of metals on page 166.

  Metals are tricky to define, as their most distinguishing aspect is their reflection of the environment. If there is nothing for the metal object to reflect, your eye will not interpret the object as being shiny and therefore not metal. Thus, make sure to put the metal objects in an environment that allows the ground and sky to be reflected. A global brush map (in the Global requester in the Action Editor) allows you to specify a world for the metals to reflect. If you have a simple landscape (even a two color picture of mountains and sky made in ten seconds with a paint program) you'll see that there is a dramatic increase in the realism of the appearance of metals.

- Why do looping anims stutter when I play them?

  If you have a cyclic path that an object is following, the cycle starts and ends at the same point. Thus, an object is in the same place for the first frame and the last frame. If you are making a looping anim, the object will not move for that one frame, causing the anim to "stutter." If you were animating a compass needle spinning 360 degrees, a 6 frame animation would *not* show the angles (0, 60, 120, 180, 240, 300) degrees in each successive frame, it would show (0, 72, 144, 206, 278, 0) degrees. When animated, each angle is shown once, except zero, which is shown *twice.* So, for one frame, the object appears to suddenly stop moving, causing a *very* noticeable stutter in its motion.

  One way to get around this bug is to define one extra frame that you do not even have to render— make a 101 frame anim, and just don't render or use the last frame. This will make a smooth cyclic motion both for objects moving on paths and for cycle objects.

- What's the best default lighting setup? How about ambient light?

  Using just one light tends to make very sharp shadows and a very odd looking picture. More than about four lights starts making everything well illuminated, and specular reflections stand out everywhere. Ambient light can help show dark crevices of objects and soften shadows, but too much really washes out the world. I usually use small values in the range of 5–25.

  There is an art to light placement; depending on how you set them up you can get drastically different results in your final image. A standard light placement that works fairly reliably is to place one light very close to your camera, which insures that everything your camera sees has some light falling on it. A second, "fill" light can be used to help fill the shadows and enhance the smoothness of the scene lighting. This fill light is generally placed about 30–45 degrees to one side of the camera. This is a plain and simple setup, but it will get you a viewable image. Sometimes scenes call for complex lighting design, but I find the two light configuration a good place to start.

- How can I get the best quality for my final render?

  First, crank antialiasing down to its best level, 0. This will slow down rendering. *Always* render in 24 bit mode; the 12 bit mode will save file size, but at the expense of poorer color resolution. You might even consider rendering at a larger size than you want your final output, then use a utility like The Art Department to shrink the large picture down. Shrinking the rendered picture will then make each pixel a more accurate representation of the colors in the image at that location- this is a way of increasing antialiasing *past* Imagine's built in limit. Your 24 bit output will be of the best output quality; now the only question is output. An RGB 24 bit board like the Firecracker is probably optimum. NTSC display from a Video Toaster is also good, but can't compare to a sharp RGB display. Next best are display hardware like DCTV and HAM-E, which aren't 24 bit quality, but are still better than HAM. Finally, if you want to display directly on your Amiga without extra add-ons, you'll have to use HAM or a Hires picture. HAM is usually best, since it can display a wide variety of colors, but if you have a detailed picture without many colors, sometimes a high resolution 16 color display can be better. Even if you have to use HAM, stunning pictures are still possible. If you do use hires or HAM as a final output, you can load the 24 bit Imagine file into ADPro and render the hires or HAM picture from there; it has much better dithering and color selection algorithms.

- How can I make mirror images of objects?

  To flip an object so that it is mirror imaged, pick it in the Detail Editor, then use the Transform command and *scale* it in world coordinates by $X = -1.00$, $Y = 1.00$, $Z = 1.00$. You'll get a mirror image object. This is very useful when you are making an object like an airplane where you can just copy one object (like a wing) and mirror image it for the other side.

- How can I convert Imagine objects to and from other formats, like Sculpt-Animate 4D or Lightwave?

  There is a commercial program called Pixel 3D by Axiom Software that performs object conversions, but it has a large liability of not being able to translate grouped objects. This program will convert the geometry and rough coloring of the objects, but there really is no software that will translate an object perfectly.

  If you want to use another another object format and are proficient in programming, there is an excellent shareware utility called TTDDD (Textual Three Dimensional Data Description) written by Glenn Lewis. If you can program (in C, AREXX, even BASIC) you can use this utility to make your own conversions. TTDDD is incredibly useful for programmers who want to play with scripted or algorithmic object creation as well as translation. TTDDD can be found on may BBSes and FTP sites, though if you use it, you should send Glenn a few dollars; it's well worth it.

  TTDDD is described on page 192.

- Imagine is acting screwy in some other way that's not listed here.

  This appendix covers the common questions that most people have asked about Imagine. If you are getting strange effects you don't understand which aren't listed here (quite possible!), there are a few steps that might help you identify what is going on. First, make sure you understand what *should* happen. If you try a new tool and it doesn't do what you expect it to, odds are you just don't understand it well enough. Look in the index of this book and read about whatever you are trying to use. If the scene lighting is weird, read about lighting. If a texture isn't performing correctly, read about the use of textures in the Detail editor and about the specific texture you are using in the Texture appendix.

If you are sure that things are truly broken and you're just not misunderstanding how to use a function, you get to experiment. Try varying different things to see what exactly causes the problem. If you have a problem with a brushmap wrap on a sphere, try using a different brushmap. Try using the same brushmap on a different sphere. Try moving the brushmap axis to other positions, even if you think they're wrong. Try rendering the object alone without other objects in the scene. Try deleting the brushmap from your object, then re-adding it back. Eventually you will find that the problem goes away or changes, and you might realize what caused the problem. If it was a RAM shortage, maybe when you rendered the object alone it finally worked because there were no other objects to use up your machine's RAM. Imagine *does* have bugs, which is to be expected in such a large program. The important part is to specifically identify these bugs and to learn how to get around them.

If you try everything and still are lost, there are still a couple of options. You can post a message on a local BBS or ask a question at your user's group meeting. Ask your dealer, or a friend who also uses Imagine. Sometimes another person will have experienced your problem before and will be able to help, or will just see something you missed.

Your last option is to call Impulse's help line. The fact that you've spent some time trying to diagnose the problem and still not found a solution probably means that you have indeed found a bug, and Impulse probably would like to know about it to correct it in future versions. *Do not* call Impulse with questions you haven't spent time trying solve by yourself; you'll waste both your time and theirs when the "problem" was caused by the fact you didn't understand a feature and not that there was a true error.

If you do come across problems, a great technique, or other tricks, I'd love to hear about it! Send a postcard or letter to me care of Apex Software Publishing. I might be able to help with the most evil problems, and I'd also like to share any hints or tips I can with other users.

Appendix B

Fun Tricks, Tips, and Ideas

This is a very disorganized chapter of (as the title says) fun tricks, tips, and ideas. Some of these hints can be the basis of a project or animation, and others are general tips that you can use in every project you build. Some of the hints in this chapter can produce some spectacular results!

B.1 Learning Keyboard Equivalents

Keyboard equivalents exist in almost every Amiga program. Instead of using the pull-down menu, you use the right-Amiga key (the outline of a capital A) with another key to perform the action. You can use these keyboard equivalents *fast* which makes using any program that much easier, and certain actions become a liquid flow of commands instead of a stuttering series of menu selections.

But learning these keyboard commands sucks! I've never had any luck learning them unless I was forced to, since that would require actual thought on my part. One day, in a flash of (rare) brilliance, I finally came up with a way to learn keyboard equivalents painlessly. This method will work with *any* computer program, but it is especially useful with Imagine and its vast array of commands.

The trick is very simple. Whenever you need to use a command and don't know the keyboard equivalent already, go ahead and pull down menus to find the command you want to use. *But do not select the command!* Instead, look at the keyboard equivalent printed next to the selection in the menu. Let go of the mouse and then *use* that equivalent by typing it on the keyboard. Sure, you're wasting that mouse movement, but by actually *using* the keyboard you imprint that physical action into your memory.[1] Allow yourself to use the menu to find the equivalent as often as you like; don't strain yourself to remember the commands. You'll find that the common commands become second nature *much* sooner than you expect.

Some commands don't have a built in keyboard equivalent, but you can make one using the programmable function keys. The most used command that has no keyboard equivalent is probably "Pick Select" which is *extremely* useful. The default

See page 135 to see how to define the F keys.

[1] "Motor memory" allows you to move your body in accustomed ways accurately without thought, which is why dancers can perform complex sequences without errors, or why an experienced typist can type while daydreaming about cheerleaders in a vat full of Jello.

Impulse function keys define "Pick Select" as F1, which is good, since F1 and F10 are the best function keys to use since they are easy to find.

B.2 Cartoons

You can make some interesting effects when you turn off realistic shading in Imagine. In particular, when you make all objects Bright, each object looks very monotone, just like a cartoon. The appearance is different from a simple color flat shading, since there are no brightness gradients across the object. I find that it is very easy to take the output of say, a livingroom, when rendered with all Bright objects, and touch it up in a paint program to look like a drawn version of the same living room. The accurate perspective makes animation of these "cartoons" smooth, but they are still easy to do since Imagine is doing all of the work. You can keep specular highlights on of off, though I find it is easier to leave them off.

Bright is an object attribute, discussed on page 33.

You can also make a similar effect by using a large amount of ambient light; this allows you to have *some* shading.

B.3 Fun with Backdrop Abuse

The fact that you can use backdrops makes some interesting possibilities. Perhaps the most useful ability is being able to render a scene, then use the picture Imagine generates as a backdrop itself, even in the same animation! As long as the camera doesn't move, the image will still look like the same scene to the viewer. You can animate objects in the foreground in front of this image, and as long as they don't move too far away from the camera, the scene will appear to be "normal." The rendering speed, however, can increase by orders of magnitude since only a few foreground objects are really being rendered. You have to be careful, though, since the objects you are rendering won't be occluded by anything in the backdrop, and they won't cast any shadows on objects in the static image. But other than that, this is a *great* way to make animations practical, since rendering 100 frames of just (in effect) one object is easy. You can also use a raytraced background while animating objects in scanline in the foreground.

B.4 Aspect Ratio and Widescreen Movies

As explained in both the Action and Project chapters, it is possible to get Imagine to output different aspect ratios to make a true widescreen perspective. The question of making a widescreen display might be confusing; after all, don't all computer monitors and TV screens show all images in the same 1.33 ratio? Well, yes, but you can use "letterboxing" to get wide-screen views. Letterboxing just means that you don't use the full height of your display, and usually the image is centered vertically on the screen with black bars on the top and bottom. This is used a lot on laserdisks of movies, since true movie buffs want to see a movie in its original widescreen aspect. You can do the same with animations on a computer screen. As a major serendipitous benefit, the fact that you are animating less pixels means rendering times are faster, and most importantly, you can play back the animations much faster. An animation with a 2 to 1 aspect ratio will play at roughly 150% of the speed of a animation with a 1.33 to 1 aspect ratio.

Aspect ratio is *not* controlled in the Stage and Action editors. Imagine really isn't designed to use widescreen views, but it can be forced to anyway through some sneaky but straightforward manipulation. The method for doing this is by playing with the pixel sizes when you define new subprojects for rendering. Section 8.1 on page 126 talks about setting these aspect ratios.

The final *picture* aspect ratio is determined by both the number and size of the pixels. The picture will become physically wider the more pixels wide the image is. It will also become wider the wider each pixel is. The *picture* aspect ratio is given by the formula

$$\frac{\#\ X \text{ pixels}}{\#\ Y \text{ pixels}} \cdot \frac{\text{Pixel}X\text{Aspect}}{\text{Pixel}Y\text{Aspect}}.$$

Normally this number is close to 1.33, which is the standard 4:3 NTSC display aspect ratio. The Amiga seems to use a value of 48:35 for its displays, which is very close.

If we want to make a widescreen image, we can look at the above formula and see that we have to either increase X pixel widths, decrease Y pixel widths, increase the number of X pixels, or decrease the number of Y pixels to get the aspect ratio we want. If you are displaying on an Amiga, the pixel aspect ratio is set. We also want to use the full width of the screen, but we can't increase the number of X pixels past what the Amiga is capable of displaying. Therefore we have to decrease the number of Y pixels. This will leave an empty area at the bottom when shown on a normal Amiga screen, but that's a price we expected to pay. If we are clever, we can center the image vertically and put black at the top and bottom of the screen, creating a letterbox effect, *exactly* the solution that is used to put widescreen movies on video. They can't change the size or shape of the television either, so they give up some vertical space.

The number of Y pixels to use depends on what aspect ratio you want. An aspect value of 1.66 will give you a widescreen movie, but 2.0 will give you the look and feel of *Indiana Jones and the Mirrored Spheres*. Just plug the ratio you want into the formula above (and use the pixel aspect ratios for the screen you are using from Table 8.1) and you can determine the number of Y pixels you need. For an overscan interlaced image (low or high resolution,) you want to use 330 pixels for an aspect ratio of 2.0 or 398 pixels for an aspect ratio of 1.66.

So, to make a widescreen picture or animation, you just decrease the number of Y pixels? Well, yes, but there's a little more to it than that. What does the preview window of the editor show? Does it letterbox? No, it doesn't. In fact, Imagine's editors don't know or care what resolution or aspect ratio you render in. So when a funny aspect ratio *is* rendered, what part of the world is shown? Something has to be lost, since the preview window shows either too little of what's happening to the sides or too much of what's happening on the top and bottom.

The answer is that Imagine uses the full width of the image shown in the preview window, and when rendered with a wide aspect ratio, it discards portions of the image on the top and bottom. That means when you render, your image will be a *subset* of the preview image! This is actually preferable to the alternative, which would be that you'd render *more* on either side of what you saw in the preview window. This would be horrible since you'd constantly be showing cruft that you didn't want shown, like the abrupt end of a sidewalk or the fact that your roadway is floating in an infinite void. Even so, you have to be careful since objects on the top and bottom of the display get cropped off, and it is difficult to tell when something is in the area that is rendered or not. I found that placing thin masking tape on my monitor to mark the top and bottom of the rendered subsection of the preview window worked well. (I used Quickrender and some experimentation to place the tape.)

When you do use a widescreen image, you'll want to set the focal length of the camera to show a less zoomed (more broad) view of the world. Section 7.9.1 on page 117 talks about this in detail. I usually multiply the default X/Y focal length ratio by a factor of about 1.5.

When Imagine displays the final widescreen rendered image, it will be shown in the right aspect ratio, and can even be animated that way automatically. However, if you want to center the image vertically, you have to do some work. It probably isn't easy to do in Imagine. You can composite it onto the center a full-screen black background, which is probably the best solution for a still. For an animation, compositing will work, but you want to keep filesizes down and not include that extra black space in the image itself. I found that if you make an all black screen and have an animation playback program play the anim on *top* of that screen, I could keep the anim centered on the screen with black on the top and bottom, exactly my goal. If you are using DCTV, you'll probably have to composite in black at least at the top stripe of the screen, since the DCTV hardware uses the top left corner of the image, and it *must* be located at the physical top left corner of the screen.

DCTV is discussed on page 190.

Widescreen movies are somewhat unique in computer animation, but they are certainly possible with Imagine; it's not that hard to cajole the right output from the program. The side benefit of faster rendering and especially faster animation playback are sweet bonuses as well.

B.5 Gravity

Acceleration along a path is described on page 109.

When you have an object following a spline path, you can set the acceleration of the object along that path. Acceleration is more useful than just making cars realistically accelerate from rest. You can simulate gravity fairly well with acceleration. Make a path pointed straight down, ending on the "ground." Set the initial speed to 0, and the acceleration to take all of the frames. The object will drop realistically, accelerating as it goes.

B.6 Teardrops

A quick way to make a teardrop shape is to make a primitive (faceted) sphere in Detail, then use the "Conform to Sphere" option in the Mold requester from the Object menu. If you have a sphere with a radius of 50, try conforming the sphere with the "Sphere Radius" set at 50 and the "Object Radius" set at 160. A higher "object radius" will stretch the teardrop out.

What is happening is that the sphere is actually being turned *inside out* as it is "wrapped" around this imaginary sphere. The sphere "runs out" of surface area as it is being pushed around this new shape, so the top starts tapering. A very vague description, but all you need to know is that it works.

B.7 Linear Texture Fun

The most useful textures in Imagine are probably wood and linear. Wood can do a lot of powerful effects, and linear is useful everywhere. The other textures are useful, too, of course, but I use linear and wood the most. There are a lot of impressive things you can do by abusing textures Here's a fun one:

- Create an object. A long logo works great. Color it and texture it any way you want.
- Add a linear texture, set the Z transition width to about 20% of the object length. Put the texture axis way over to one end, oriented towards the center of the logo or whatever. Make the color of the texture be black, no reflection, and 255 255 255 filter. (Yes, completely transparent.) Make sure the linear texture is the last one if you already have some other textures on the object.
- Render. You should have basically an invisible object, since the linear texture is completely transparent and covers the whole logo. Fix the axis if it is pointed the wrong way.
- Copy the object. In the copy, move the texture axis way to the other side, oriented the same way. Save it with a *different* filename. Test render. It should look just like your normal object without a funky linear texture. It should certainly *not* be transparent.
- The fun part. In the Action editor, morph object one into object two. The only change is the texture axis, so Imagine will interpolate its location from one end of the logo to the other for each frame. Make the animation at least 10 frames, preferably 20. You can render in scanline mode; it'll work just fine.

What happens is the linear transition band "flies" across the logo, fading the logo in as it moves from one side to the other. It's an impressive way to introduce an object into a scene! It is also pretty easy to do; 10 minutes tops.

B.8 Animal Walks in Cycle

If you are interested in character animation, the Cycle editor is very useful in building complex figures that can change their configurations in time. One project I made experimented with walking animals. Bipeds like humans have only two legs, so there are only a couple of different strides that they can perform. The quadrupeds I animated actually had a very complex rhythm to their walk, since the four legs can work together in many complex ways.

Gait Name	Left Front	Right Front	Left Rear	Right Rear
Amble	0	.5	.75	.25
Trot	0	.5	.5	0
Pace	0	.5	0	.5
Canter	0	.3	.7	0
Transverse Gallop	0	.1	.5	.6
Rotary Gallop	0	.1	.6	.5
Bound	0	0	.5	.5

Table B.1: Leg phases for different quadruped gaits

The order in which an animal moves its legs greatly depends on what speed it is moving. Grazing cattle drift along, lifting one leg at a time and placing it before moving the next leg. A galloping horse moves its rear legs almost as a group, with the front legs also staying together to take up the impact from the small jump the powerful rear legs provide. A horse in a mid-speed walk called a "trot" moves diagonally adjacent legs together.

I compiled a list of quadruped gaits, shown in Table B.1. This table shows the "phase" of each leg, or what point in the cyclical run or walk it lifts off of the ground. To figure what frame a leg should be lifted, take the length of the cycle (perhaps 20 frames) and multiply it by the leg phase. For example, the front right leg of a horse in a canter should be lifted off of the ground starting in frame 6 of a 20 frame cycle. The gaits in the table are listed in rough speed order, with a gallop being faster (both in animal speed and repetition of the cycle) than an amble.

This chart is especially exciting when you start morphing gaits. Cycles can morph to one another as long as they have the same keyframes, so if you are careful about setting up two cycles, you can use morphing to smoothly slow a galloping horse into a canter, then an amble, then stopped. This seamless gait change is very smooth and professional! The exciting possibilities of morphing cycles is discussed in Section 5.3 on page 82.

B.9 Motion Blurring

Motion blurring is an advanced feature found in high-end renderers. When you are making an animation or even a still image, a single frame can show objects that are moving fast as being blurred. This is due to the fact that a real camera keeps its shutter open for a finite amount of time, and objects that are moving quickly are photographed as a short streak as opposed to a still image.

This motion blurring can convey a real sense of realism, and one of my more popular pictures, "Strike" uses this. One way to create motion blurring is to make an animation which everything is moving *very* slowly, perhaps 10 times slower than a normal animation. If you render 10 frames in a row, these frames represent the positions of the objects over a short time. The objects that aren't moving will stay in the same position in each frame, but objects in motion will each have slightly different positions in each frame.

If you can somehow average these frames together into a single image, you can represent motion blurring very accurately. One method of doing this is with the program *ADPro* which allows you to average pictures with the composition controls. DCTV can also average images.

ADPro is discussed on page 189 and DCTV is discussed on page 190.

To come up with the average frame, load the first image of the series. Now, load the second image and merge it with the first with a weighting of 50% so each image has an equal contribution. If you add the third image with a contribution of 33%, the fourth with 25%, the fifth with 20%, and so on, you will end up with a composite average of all ten frames. There is nothing else to do! The blurring is automatically made by the displacements of the objects in the mini-anim. Fast moving objects are blurred into longer streaks, just as they should be.

If you don't use enough frames or your animation proceeds too fast, sometimes the blur shows the individual frames that make it up. In this case, you have slow down the animation (not necessarily easy.) I find it easier to initially build my animation *very* slow then if I don't need it that slow just using every other frame.

You also can change the *EDLE* antialiasing parameter way up to increase rendering speed. The averaging of many frames tends to perform extra antialiasing so you can get away with less samples in each frame.

B.10 Pathnames

Making projects is easy, but there can be *major* problems when moving projects around or sharing them with others. Since all of the file names in Imagine are hard-wired (they have the full path in them) you can't change the location of a project and expect Imagine to still be able to load the object files and brushmaps.

To help reduce this problem, I've started a convention that has saved me many times. I have made three virtual disk partitions called *Projects:, Objects:, and Maps:.* I put all of my projects in the *Projects:* disk, the objects in *Objects:* and brushmaps in *Maps:.* These aren't necessarily disk partitions, they are really virtual partitions made by executing the AmigaDOS *Assign* command.

Now that I have these assignments, moving files is much easier. When I ran out of disk space for my brushmaps, I just copied all of them onto a disk which had some room left, and changed the *Assign* statement. Also, I can put a project onto a floppy disk and assign *Projects:, Maps:, and Objects:* to it after copying the object and brushmap files I need onto it. Then I can render off of this floppy disk, or more likely, be able to give it to another Imagine user and let them play with the project. If the pathnames hadn't been assigns, I could copy the project, but the other Imagine user would either have to have the exact same directory structure as I (unlikely), or would have to manually load each object to change the brushmap names, then use the Action editor to change each object filename. Not pleasant.

B.11 JPEG

When you are rendering high resolution images with 24 bitplanes of color, you can quickly run out of disk space, since a hires picture can take a megabyte of filespace! One solution to this problem is an image file format called JPEG. This is a method of storing 24 bit files in very compressed format; a one megabyte hires image might be stored in a file only 75K large.

There are two disadvantages to JPEG. First, it takes a considerable amount of computation to compress and decompress the file. This decompression might take 20-30 seconds for a hires image on an Amiga 3000, or several minutes on an unaccelerated machine. Also, JPEG is a *lossy* storage format. Artifacts can be introduced into your image which can add some high frequency noise to the image.

Most people panic when they hear that JPEG is a lossy algorithm, but it really is not that bad. You can set the JPEG compression to trade off file size for image quality; a smaller file size obviously has higher loss. However, the loss that does occur is often so trivial it is not visible even on a 24 bit display! When you can get a 10 to 1 compression ratio without any visible loss, JPEG is well worth using.

ADPro is discussed on page 189.

The JPEG format is becoming quickly accepted on the Amiga community. ASDG's ADPro can read and write JPEG format, but there are also public domain implementations which are completely compatible and of equal quality. A copy of these PD JPEG algorithms is included on the diskette that came with this book.

B.12 Multitasking with Imagine

It is very useful to run other programs at the same time as Imagine, especially since Imagine might take hours or even days to render a picture. Luckily, the Amiga is a multitasking machine and starting a second (or tenth) program is easy. Just start Imagine running and use the screen gadgets (or right-Amiga-m) to bring Workbench to the front and launch your other programs.

The only limit in running many programs at once is RAM. If you have enough free, you can run nearly anything else except programs like games that take over the entire machine. You can even run a second copy of Imagine! You could be modeling objects with one copy while rendering a picture with another. Some people even run two copies of Imagine as a matter of course, so they can leave one copy running in the Stage editor, and another copy running in the Detail editor. When an object is saved from Detail, you can flip to the Stage version of Imagine, and use the "Goto" command to visit the *current* frame. This reloads all of the objects, and changes you made with the other copy of Imagine will update the models on the Stage. This technique is doubly useful when designing complex cycle objects, since you can see how a cycle animates in the Stage, then adjust the cycle using the other copy of Imagine. (Again saving the cycle, and using "Goto" in Stage to reload it.) There is a considerable convenience in this method since you never have to reload the objects in the Detail or Cycle editors.

Especially when render in the background, sometimes you want to lower Imagine's task priority. This means that Imagine won't use *any* of the Amiga's CPU unless there is some idle CPU cycles that would be wasted anyway. Instead of running your paint program at half-speed (since it splits the CPU with Imagine), you can run it at *full* speed as if Imagine didn't exist. The great part is that for most programs like a paint program, they don't use much CPU so the Imagine rendering time really doesn't slow that much. If you are using a modem or a text editor, the slowdown in rendering times is almost imperceptible. However, when you do ask for a complex action, the program you are using will respond at full speed as if there were no other programs running. Multitasking is great!

To change task priorities, you can use several public domain utilities which list running programs and let you change their priority numbers. Setting Imagine to run at a negative priority will accomplish what you want; not to use CPU unless there is some free. I use a public domain program called *Xoper*, but there are many other PD programs that will do this. Another way of setting priority involves starting a CLI, then executing the AmigaDOS command *taskpri -1* then running Imagine *from the CLI* by typing its path and filename.

Many of these programs can be found on the Fred Fish collection of PD disks.

B.13 Rendering Limitations

While this isn't a trick, it is very useful to know what undocumented limitations Imagine has, especially in its black-box rendering algorithms. There are quite a few places where Imagine falls short, especially when handling transparency.

An important note is that although images are reflected and refracted properly in trace mode, it is not a complete simulation. Light coming from lightsources is *not* affected by reflection and refraction, so you *cannot* build a magnifying lens to focus a light on a region, or use a mirror to reflect light to illuminate the back of an object. You can't add filters in front of a light to color it.

Also, transparent objects, even when they are completely transparent, can have some effect. Stars and backdrop maps cannot be seen through transparent objects. When you are in either scanline or trace, transparent objects filter out an objects "view" of the sky, which plays a critical role in coloring metals or reflective objects. John Grieggs, a respected Amiga artist, had a beautiful emerald that became a flat lump of rock when a transparent bell jar was placed over the jewel as a case.

Also, views through transparent objects sometimes aren't perfect. A view of an object with a texture applied to it sometimes does not show the texture, a very important failing.

Other problems with Imagine's renderer aren't so serious. The "perfect" sphere is actually rendered as a faceted sphere in scanline mode, which is blatantly obvious when you look at the profile of the sphere. If you are using scanline, a primitive sphere with many triangles will provide better results than the perfect sphere.

B.14 RAM Use by Rendering methods

During rendering, Scanline takes a good deal of RAM, partially because it has to keep brushmaps in memory to render on your objects. An interesting observation is that during the initialization phase when you start a render (before you see the percentage gauge start counting) the memory use of Imagine peaks. You can see this by watching the free memory display in Workbench. After the picture starts rendering, some RAM (perhaps 5–10%) is freed back up. This information might be useful if you want to multitask with Imagine. If RAM is tight, you might be better off to start Imagine rendering first, then start other programs *after* the actual picture starts rendering and you see the percentage gauge.

Raytracing also uses RAM for brushmaps. However, trace mode actually uses *less* RAM than scanline. Depending on the scene, it can be about 15% less. Like scanline, the peak RAM use is during the initialization phase of the render, with some RAM freed when the picture actually starts rendering.

B.15 Visible Light Beams with Fog

Actually, making visible light beams is pretty easy. Remember that fogged objects are really just light absorbing volumes in space. (Actually, it's more like light *tinting* volumes.) You want any ray that passes through a certain region to be become tinted with the color of the light. This region is a cone or tube is defined by the area the light is passing through.

We can make this volume tint rays that pass through it, which is a decent approximation to the effect we're looking for (which is caused by the bright light scattering off of air, dust, or water droplets). Just make a solid tube or cone of the size and shape you want the visible light volume to be. Just scaling a normal primitive tube is usually fine. Color it white or yellow (whatever shade the light is) and position it in your scene starting where the lightsource is and pointing in the right direction.

You can change how visible the light beam is by setting the "fog length" parameter, which controls absorption in the volume you've defined. The longer the "fog length," the less dense the fog (or absorption, or tinting, depending on how you look at it.) As a good starting place, make the fog length equal to twice the width of the "beam" as measured in Stage editor Imagine Units. Quickrender... you should see a noticeable, but still transparent light beam. If it's too dense, increase the fog length. If it's too tenuous, decrease the fog length.

There are two problems with visible light volumes. First, there is no shadowing. Since it's just a solid object, something that intersects the beam won't "stop" the light after it passes through. Also, the fog density is constant, so the intensity of the "visible light" does not decrease with distance. You might want to add a linear texture to make a nice color gradient to partially emulate the intensity dropoff. Not perfect, but not bad.

B.16 Star Backgrounds

You can add stars to your background sky by using the global "Starfield Density" control described on page 114. However, these stars don't animate realistically. If you make a space scene and pan your camera, the stars *don't* move like they should, really shattering the illusion. Also, the stars are all the same intensity (just one pixel of the given color.) When rendering at high resolutions (like for a magazine cover) the stars in effect decrease in size since they are only one pixel wide.

There are a couple of ways around this limitation, and both ways require almost building the stars manually. If we position a real object *way* off in the background, you can simulate an infinitely-distant object like a star. As long as your camera doesn't move around the world a distance that is significant compared to the distance the stars are away, the parallax should be minimal. For example, if the star objects are 1024 units away, you can probably get away with moving your camera any way you like in a box about 50 units wide. If you move it 500 units, you'll see the stars change relative position due to the large camera displacement actually getting another view.

You can build stars by using perfect spheres or simple solid shapes. I suggest a tetrahedron, a four sided solid. This is the simplest 3D figure that has a viewable cross section no matter which direction you view it from. (A disk might be viewed edge on, and a cube is a more complex shape.) A tetrahedron also seems to be rendered faster. If you create a tetrahedron, assign it a bright white color with a tint of blue, and set the "Bright" attribute, the object will be clearly visible no matter what angle it is viewed from. Use "Cut" and "Paste" about 4000 times to create a large number of these (you'll need more than you think!) Remember you can "Cut" 10 copies at once, then "Paste" those 10 copies, then "Cut" the 20 objects, and "Paste" 20, and so on, so making 4000 objects is pretty easy.

Bright is discussed on page 33.

When you've created a bunch of these objects, you should scatter them way off in the distance. I added a perfect sphere to the world and used it as a guide to help place the stars. Make sure to place them *everywhere* including the top and bottom of the sphere, and try to keep them unclumped. This is a somewhat boring task, but you only have to do it once. When you're done, "Join" all of the stars together (you'll never have to move them again) and save the single object as your "starfield."

If you can program, TTDDD is a perfect solution for this task of star placement. You can write a program that asks "How many stars?" and then automatically scatters them on the sphere's surface. You could even use a database of real star positions and position them in accurate locations.

TTDDD is discussed on page 192.

When you use this "Star" object, your scene usually has to be small (to keep the stars far away). This can effect raytracing times, so be careful! See page 116 for a discussion.

Another method of making stars is easier. You can make a brushmap of scattered points, and map it onto a sphere (using Wrap X Wrap Z). If you put all of your objects and lights and the camera *inside* the sphere, you'll see the brushmap background. This method has a couple of drawbacks. First, the sky colors will be blocked by the sphere. Even if you make the sphere transparent (using the star brushmap to control the transparency of the sphere, with opaque stars showing a white color of the sphere) the limitations of Imagine's renderer won't provide accurate sky coloring. (See page 153.) Second, making the stars appear randomly is tough; they will concentrate at the poles due to the mapping Wrap Wrap uses. (See the brushmap appendix.)

Either way you do it, you *can* make decent starfields, it is just a lot of work. If you raytrace, the extra rendering time may not be worth it. Luckily you only have to set it up once, then you can use it for *all* of your projects.

Appendix C

Strategies For Success

Even if you have mastered every tool in Imagine, there is still a lot information that can help you use those tools to develop your models. An entire second book could be written about this topic, but included in this appendix is a set of useful ideas that can help you design and build objects, scenes, and animation.

C.1 Object Creation Techniques

Some custom objects are very easy to build. A company name can be digitized, and Imagine's IFF converter can convert the brushmap into a 3D object. But what if you have a client who wants a model of their restaurant? Or you need to render a specific brand of car? It is the complex models that take considerable thought into how you can input their shape into your modeler.

When you decide to build an object, odds are you'll sit down and boot up Imagine. This is often the worst thing you can do! Once I was producing an ocean scene, and I had a vision of a dolphin leaping out of the waves. I built a stretched and bent teardrop shape, added a blunt nose, some flat, triangular fins, and colored the final object with a nice "dolphin grey." I was careful, and the final dolphin shape was very clean and detailed. But I was still dissatisfied with the results. After hours of tweaking, I still wasn't able to get a reasonable form that seemed right.

This is when I learned perhaps the most important rule of object design. Never, if you can possibly help it, just sit down, boot your Amiga, and try to build the object you need. First, sit down at a table and *think* about your object. Why? Well, it turns out there were several problems with my dolphin, which I finally realized when I saw a T-shirt with a picture of a dolphin on it. My model was based on my own skewed mental idea of what a dolphin's shape was, and had a good many discrepancies with the shape of a *real* dolphin. The beak was much larger than my mental model, the coloring actually was much more complex than I knew, and I was even missing the top fin. I learned that I could never trust myself to accurately remember the form and coloring of such a complex object; my memory just wasn't accurate enough. It wasn't a lack of skill with the modeler, it was just the fact that my model was *not* the same shape as a dolphin, which is why it never looked "right" to me.

So what do I do now? Well, there are several strategies I follow, depending on the type of object I need to build. They all revolve around the fact that I have a reference to follow; a guide that helps me visualize what the object *should* look like. If I were building a dolphin today, I would first go to a library and find a picture of a dolphin. It could be a blurry photocopy, but having that image helps me in several ways. First, I can actually study the image in detail. Gross mistakes, like missing fins, would quickly reveal themselves. Also, since I now have looked at the image with the idea of modelling it, I notice new features I would have never noted before. When I see a dolphin show at an aquarium, I don't notice the fact that the body width is about one fifth of the body length, I just say "Oooo! Cool dolphin!" When I try to build the "cool dolphin" from memory, I might (and did!) build the dolphin stretched out to twice the length it was supposed to be. Studying the image with the idea of modelling it will help revise your mental image of exactly what you want to produce.

What if I want to build a UFO or a 7–11 slurpee machine, something which I can't find a good picture of? I use perhaps the most powerful tool available on any computer platform; a pencil and paper. I am a truly horrible artist when dealing with drawing figures by hand, but sketching a basic shape can help my final object quality enormously! Changing a detail with a pencil is considerably easier than having to scrap a 3D model and start over. The key is that I have a firm idea of what the final object should look like when I actually sit down to build it.

C.1.1 Planning

Once you have an idea in mind, and you're sure of the shape you want to produce, what is the best strategy for inputting it to your modeler? No matter what program you're using, there are a few useful methods for building your object from scratch.

Perhaps the most important idea to keep in mind is that any object, no matter how complex, can (and should!) be broken down into simpler parts. Building a model of a car is difficult, but how about a spoke in the wheel? The hubcap? A headlight? These small parts are easier to build, and many of them can be formed out of "primitive" objects like tubes and spheres. An object is made of its parts, and if you build each part separately then finally assemble them together, you'll save a lot of work for yourself.

Imagine encourages you to build complex objects from simpler parts, then grouping them together into a complex shape. I've found that the best way to build models is by starting with the fundamental, base piece, and then attaching the little doodads and ornaments. As you build each little part, you can immediately place it on your "final" object, and see how it fits in with the other parts of the model. If you wanted to build a bicycle, build the frame first. As each new piece is added, you'll see immediately if something is mis-sized or broken in some other way. You'll also have a convenient way of storing your object in progress, since all of the parts are logically grouped together. How do you save just a headlight, a rear bumper, and a steering wheel together in one file?

One last advantage of modular modelling is the ability to replace the parts easily. If you are building a car, you might want build a quick and dirty set of wheels so you use some toruses. But what if you want to improve the model later? You can just pick the old subobjects and selectively replace them with new, more detailed versions. Conversely, sometimes to save space or to speed up rendering you might want to simplify a model, and again, you can selectively replace a subobject to change the quality.

One last point: Be careful how detailed you make every part. A classic mistake is to say "Oh, I want a super high quality beer bottle" so you design your object with a bottlecap with 5,000 triangles. Sure, your cap is very smooth, but you can easily increase rendering times if you don't try to keep detail levels constant. Perhaps you would want that much detail if you were making an animation which zoomed in on a logo on the cap, but use common sense to determine the object's complexity when you can. Oversampling like this is a habit I constantly fall into, where I specify much more detail than will ever show up when rendered. It not only wastes RAM and rendering speed, but also time I spend in modelling.

Once you get used to building small, simple objects, you'll be surprised how fast and easy it is to assemble them into large complete models!

C.1.2 Model Sources

When you want to build a complex model like a 767 jet in Imagine, a picture or sketch of the jet's shape is a great place to start. But for some objects like the 767, there is an incredibly rich source of information that can guide you in making very detailed and completely accurate models. That source is your local hobby shop, where they sell plastic model kits! If you think about holding a miniature airplane wing that you can study and even measure, you can see how ideal they are for producing an accurate computer model of the same shape. As a bonus, the model kits come with decals, which can be digitized and placed on your computer model as brushmaps!

Not every object in the world can be found at your hobby shop, but there are a surprising variety. Vehicles are by far the most popular; nearly any specific type of airplane or car can be found. An excellent variety of ships can be found as well; however they tend to be less detailed since the real world versions are so large the scale of the model can shrink to scales of 1/500 or even smaller. There are also models of many other objects, from the human body to Japanese science-fiction robots; if you look around you can often find truly unique kits.

If you have a choice of models to buy (many, many models come in different scales and from different manufacturers), remember that the larger the model is, the easier it is to measure, and the more detailed it is likely to be. If cost is a major factor, you'll find that the smaller models are considerably cheaper and not necessarily that much harder to deal with. The best quality models seem to be from Japanese model companies; the US models tend to be very simplistic and have very sparse decals.

A final note is that when you're done with the models, you can build them yourself, or give them to a friend or one of your kids. If you want to make a permanent friend, you can just dump your plastic parts into a plastic bag and bring them back to the hobby shop you bought them from. There is an entire group of "scratch builders" who build models from random parts, and will be *very* happy to have any donations of miscellaneous parts, even if they have pen marks on them and are from twelve different kits.

C.1.3 Real-world Measurement

In order to accurately copy most of these plastic parts in your modeler, you'll first need a few simple tools. A ruler, though useful, is actually not the best measurement device. By far, the best tool to use is a "caliper gauge." This tool consists of a long piece of metal or plastic with rules marked on it, and a piece that slides along the rule. The object to be measured is placed between the "fingers" of the gauge, and a mark on the sliding piece indicates the width the fingers are spread apart. Measurement is quick and accurate; I use a cheap $8 gauge that has an accuracy of one twentieth of a millimeter, far more than I need. Most of these gauges have two sets of fingers, one for measuring the distance of an object between the fingers, and another for measuring interior dimensions like the inside diameter of a pipe. The problem with a ruler is that it is difficult to measure small distances accurately and measuring something like the thickness of an airplane wing at a certain location is impossible.

One important convention when measuring real objects is to *always* keep your units consistent. There has to be a conversion between the size you measure on your model and the units you use in your modeler. In Imagine, I arbitrarily set one millimeter measured from a plastic model to be one unit in the Detail editor. As long as I don't suddenly start using one centimeter to one Imagine unit, every part I digitize will stay in scale and I'll have no problem. Model kits generally have a scale rating to them, such as "1/24" which tells you exactly what conversion between your model size will give you the proper final scale of the real-world object.

Mixing inches and centimeters will cause a headache. I recommend trying to get all metric measuring tools; it's a lot easier to make the conversions, and Imagine doesn't like it when you try to input 13/32 of an inch as a unit. You don't *really* want to use a calculator every time you enter a number!

A surprising number of the plastic model parts can be input to your modelling program very quickly and easily just by measuring a couple of dimensions and using a built-in primitive from your modeler. A mainmast of a sailboat is just a tube in the proper dimensions. Different types of tires can be made from a torus or a cylinder in the proper shapes. As long as you can just measure the simple object parts accurately, assembling most of the primitive shapes is easy. There is one common problem that can occur is when dealing with cylinders; remember that you are usually using the calipers to measure the cylinder's width, which is along a diameter of the object. If you use this number as the *radius* of the computer version of the object, you'll get a cylinder that is twice as big as you expected.

The second most useful tool in digitizing plastic models is simple graph paper. Flat shapes can be placed directly on the paper and traced, so the location of key boundary points can be quickly determined. For an airplane wing, I traced the outline of the wing on graph paper. I drew about 20 parallel "ribs" from the base of the wing to the tip, and used the graph paper to note the length of each rib (the wing tapers near the tip) as well as the position of that rib. I used the calipers to measure the maximum thickness of the wing along each of those ribs. Armed with this list of rib sizes, I created a 2D outline of an airfoil shape. I copied this single outline for each of the ribs, then scaled the outline at each rib to be the same height and length I measured on the model. Imagine 2.0's cross section editor allows you to enter in these cross sections, place and scale them, and interactively correct small mistakes in the shape and placement of the ribs. Try to find scientific graph paper, which is usually made of fine millimeter grids, with medium lines every five millimeters and heavy lines every 10 millimeters.

Another trick you can use that works extremely well for complex outlines calls for some hardware. If you have a genlock, place your object on a piece of white or black paper, whichever gives better contrast. Focus a video camera on the object so that it is looking straight down at the object and you can clearly see the outline of the part. If you truly want an accurate outline, you want to avoid getting a perspective view of the object; keep the camera centered right over the object. Also, do *not* use your macro lens. You're looking for the true projected outline, and if you get too close the camera perspective will start to distort surfaces at different depths. The best setup is to get your camera as far as possible *away* from the object and use a zoom lens to blow up the image to take up most of the screen. This involves a tall tripod if you're digitizing something on the ground, or supporting the object on its side and using the camera horizontally. Genlock the camera output to your computer, and enter your modeler. Unless you have strange background colors, you have a transparent background that shows the video image of your outline. You can now use this to decide where to place points and edges to best represent the image. Obviously you cannot zoom or pan in your modeler, but you can still get quite high quality measurements very quickly. Make sure to scale the final outline so that it matches the scale of the actual part.

I write the number of each plastic model part on each piece with a pen. When parts get mixed together, re-identifying them can be difficult. Most pens will do a horrible job, since ink doesn't like sticking to plastic, but I've found a $1 black "Sharpie" extra fine point permanent marker (by Sanford) works beautifully. You can also use the pen to make tick marks, identify "ribs" on wings, and make other useful measurement marks on the part.

You can digitize 90% of all of the model parts you come across without much thought just by measuring the part and assembling or extruding a computer version. The small parts in model kits tend to be simple, since complex shapes are difficult to form in the plastic extrusion method the model companies use.

C.2 Cross Section modelling

Complex objects that can't be broken down into parts tend to be organic and liquid shapes, or smooth shells or hulls of vehicles. In either case, the main distinguishing feature of the object is a smooth continuous surface, since any abrupt changes or sharp features can just be modeled by splitting the object into parts along the discontinuity.

Since these shapes are smooth, a lot of the standard tools we use to create objects aren't as useful. Primitive shapes glued together are discontinuous, and flat extrusions aren't detailed enough. The best tools for dealing with smooth shapes are the ones that deal with manipulating many points simultaneously. Most 3D modelers allow dragging many points at once. Imagine 2.0 will allow you to pick a group a points, and move, scale, or rotate them interactively. You can also use "magnetism" mode that to move surfaces like they were made of rubber or clay.

Unfortunately, these multiple point manipulation commands are very difficult to control exactly. They are terrific for editing your model to fix small errors or sharp edges, but if you have a blueprint or you are holding a model it is very tough to control these manipulation commands enough to make an accurate representation of your object. This is not saying it is not possible, just very difficult. When you are trying to copy a model, accuracy is obviously very important, so modes that aren't precisely controllable are tougher to use for model digitizing. This doesn't mean these modes aren't useful, they just don't help as much. Louis Markoya, a well known Amiga artist, is very adept at using Imagine's magnetism modes for making beautiful objects (like some terrific avian shapes), but not from blueprints!

C.2.1 Cross Sections

There is one other common modelling ability that allows for making complex one-piece objects. This is the ability to "skin" a succession of outlines to form in effect a "tube" with an arbitrary cross section along the tube. "Skinning" is a very powerful modelling technique. An object, no matter how complex, can always be modeled as a series of sliced cross sections as long as the object is smooth and continuous. Since smooth and continuous objects are exactly the type of object we are having trouble with, this makes this method of modelling very appropriate.

For example, I once had to make a boat hull for an ocean scene. The hull has a streamlined shape that has to be cast in one piece, so the standard modelling methods (quick extrusions and joining of primitives) just couldn't represent it. However, the hull was ideally suited for modelling by cross sections, since the shape of the hull changes smoothly over its length. Figure C.1 shows the hull's outline (a side view) as well as several of the cross sections of the object. With enough of these cross sections (I used a total of 13) I was able to characterize the hull very accurately.

Imagine requires each cross section to have the same number of points in order to make the computer's job of matching the corresponding cross sections easier. In the Forms editor, you can interactive define, edit, and adjust these cross sections. The control that you have over each cross section is phenomenal, since you can define and add new cross sections in the middle of an object, make interpolated cross sections that aren't "key" cross sections, and use different modes to keep the object symmetrical over different axes. Individual cross sections can even be stretched, warped, and tilted. Forms does its job very well, though it is not easy to learn to use at first. The editor's interactive control over cross sections more than offsets the limitation having to use an equal number of points in each cross section.

C.2.2 Measuring Cross Sections

Understanding how to use Forms to define objects by their cross section requires practice, but it really isn't difficult. The tough part is actually determining what those outlines should look like! Dragging points into a shape is easy once you know what that shape is supposed to be, but when you have a real-life model, how do you measure these cross sections in the first place?

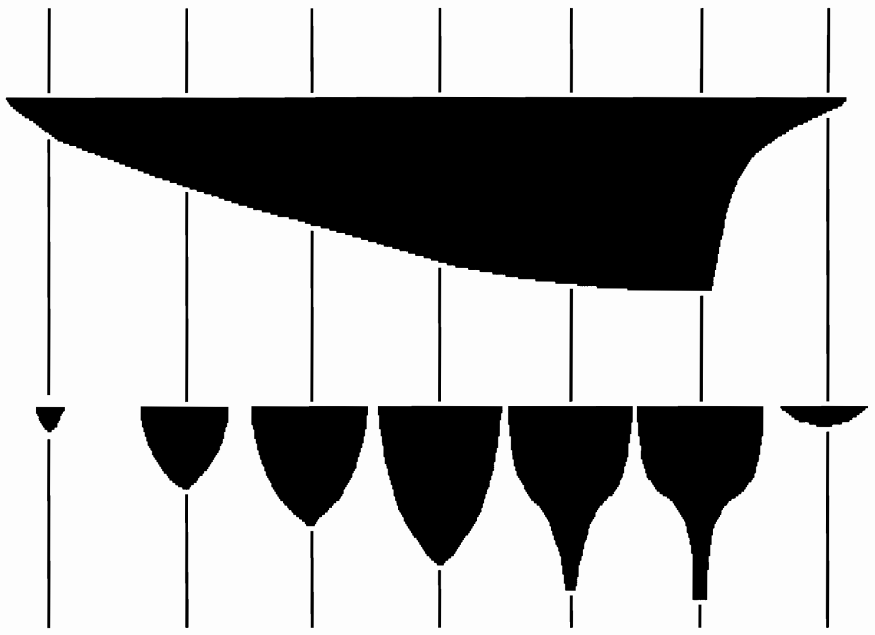


Figure C.1: A hull and its defining cross sections

There are many techniques for this. (Many of them will involve tools that look very strange sitting next to a computer, like hacksaws!) The goal is to take a physical object and determine its cross sections accurately. A ruler or a set of calipers will be of little use, since you want to locate the absolute locations of edges and points on the cross section and not just the lengths of the edges. You need an absolute reference in two coordinates simultaneously. Ideally, being able to trace the cross section outline on graph paper would allow you to accurately see the shape of the cross section and enter it into your modeler. Our goal is to eventually get the cross section shapes onto graph paper, since the model will quickly follow.

Transcribing cross sections onto paper is still very difficult! There is almost never a way to directly place your object on the paper to just trace the cross section outlines, so we have to use indirect methods. Perhaps the most elegant method involves a tool called a "contour gauge" found at most hardware stores. This tool measures odd profiles of corners and moldings so floor tiling can be cut in the exact opposite shape to snug up closely to the wall. The gauge consists of a large number of stiff metal wires aligned side by side in a row, held in place by the friction from two plates that sandwich the wires between them. The wires are relatively free to move in and out, so you can press the gauge against a shape and the wires move in the appropriate amount. You just press the gauge against your object, then place the gauge on your graph paper and copy the shape.

This seems like the ideal tool for cross section measurement, and it works quite well for many shapes. Unfortunately when you have a sharp contour, the wires bend out and around the edge instead of being pressed in. Also, the relatively coarse size of the wires makes very accurate measurements difficult. This tool looks like the ideal solution for cross section measurement, but in practice it is best for large surfaces without sharp features.

Mike Halvorson (president of Impulse) told me once that he had good luck modelling cars by cross section. He went to a hobby store and bought a clear plastic "shell" of a car. These shells are often used as the base for large radio controlled cars. Mike used thin, opaque tape along cross sections of the shell and eyeballed the now-visible cross section shapes. Though it isn't necessarily very accurate, this struck me as a particularly elegant way of getting the feel for the model's cross sections since the clear body makes seeing the shape defined by the tape easy.

C.2.3 Casting Molds

One possible way of measuring your object's cross sections might be to just take a hacksaw and slice your object into slices thin enough that they can be placed onto graph paper. Medical laboratories often build models this way, with a razor blade taking off sheets of tissue that are digitized with a camera, then assembled in a computer. In our case, this might very well be a practical solution, but unfortunately it tends to destroy your model as you measure it. Even if you don't want to keep your model, there is a significant danger that if you make a mistake you'll never be able to start over. Also, many models aren't sturdy enough to be able to slice accurately; a thin piece of plastic would rather break than be sawed through.

It is much better to measure the cross sections indirectly which still leaves your model intact. The best solution I've found for measuring cross sections is by making a cast of the model- a mold. Artists are used to taking casts of objects to copy them in plaster or other materials. If we can make a *physical* copy of our model without damaging it, we are free to mutilate the *copy* with a hacksaw. An additional benefit is that we can make our copy out of a sturdy material that is still easy to cut through.

After a lot of experimentation I've come up with a very reliable, cheap, and easy way to copy objects. Artists generally use a "molding compound" or "casting media" which is a paste that can be brushed directly onto an object. Alternatively, an object can be immersed in a bowl of the goop. This compound changes from a paste to a flexible, springy surface in about half an hour. Very surprisingly, the compound does not seem to stick to anything: plastic parts pop right out, metal is even easier. A large glob that accidentally fell onto carpeting (!) actually came off easily, leaving the imprint of the shag! Additionally, the compound is water soluble, non-toxic, and cheap. I found a product called "Instamold" at a local artist supply store that came in a powder form (mix with water to use) that cost about $6 for enough compound to make a gallon.

The casts that this material makes could be sliced and measured directly, but it is a little too pliable to hold its shape when cut and measured, and after about 6 hours it does become hard, but it shrinks and cracks. It is really more suitable to use as a mold for a *second* casting material. Artists use plaster of paris quite often, but plaster is a little too hard and messy to be slicing into the cross sections that we need. I found a second material that worked beautifully: styrofoam. You can buy pressure cans of a "non-toxic foaming insulation compound" at your hardware store for about $5, which works great. The foam comes out of the can in a mushy mess that expands as it cures (in about 2 hours) taking the shape of the mold. When it's dry, you can take the styrofoam model and start hacking away. The styrofoam cuts well and leaves sharp boundary lines that are easy to trace on graph paper.

Another way of casting your model is to use the styrofoam directly on your object just like the casting media. The styrofoam could be sliced after it sets, and you could measure the cross section shapes from the interior "hole" your object left. Unfortunately the styrofoam sticks to everything (including skin!) so you have to protect your model from the styrofoam first. I painted the plastic hull with a thin layer of liquid latex (also found at an artist's supply store.) Liquid latex is fun stuff, it is basically liquid rubber which dries rapidly. You can even dip your hands into it and make custom rubber gloves... It peels off without any fuss from plastic, but I wouldn't try it on carpeting.

I built a long, rectangular box out of thin cardboard that the boat hull completely fit in, similar to a shoebox. I placed the hull inside, then filled the remainder of the box with styrofoam. When the foam had hardened, I popped the model out (the thin latex layer stayed behind), and was left with a rectangular solid "anti-hull." I placed it into a mitre box (a tool for holding materials when being cut) and used a hacksaw to lop off wedges about two centimeters wide. Make sure to label the wedges before you start cutting to identify each cross section! It is also easier if you mark the cardboard with equally-spaced parallel lines to identify where to cut.

This second method actually is more convenient, since it is a one-step process. Additionally, since the "anti-model" was in a nice rectangular box, I was able to identify the orientation and position of each cross section, since the rectangular outside of the mold stays constant for each slice. This absolute reference is valuable for positioning the cross sections relative to each other.

C.2.4 Touch Up

Usually there is some extra work to be done after the skinning. The boat hull was smooth and was ready to use, but what about a car body? I modeled a Volkswagen Golf body, and successfully used cross sections to make a wonderfully detailed version. Unfortunately, it still needed a lot of work to finish it since the windshield and windows were cast as an integral part of the body. There are a couple of techniques for finishing off these small details.

Very often, you want to split part of the smooth shape off as a new object. With the Volkswagen, I wanted to remove the windows from the body, but still keep them to form the glass with. There are a couple of techniques for splitting objects like this. The most straightforward method just involves deleting the points and faces you don't need in the body (like the windshield), then loading a copy of the original shape, and deleting the points that you don't need in the windshield. With two passes of deleting, you get the two separate objects. You can "cut out" parts of your object this way pretty quickly and easily.

If you know you are going to be cutting out sections of your model, you might want to add extra cross-section slices in that region so that you have smaller faces there. That will allow you to better select the exact region you want to cut out.

Of course, the most elegant way to cut out sections of your model is to use "Slice." You can build a "cookie cutter" object that will cut arbitrary shapes out of your model. This is ideal for pulling things like windshields off of your model, especially since the region where you are cutting doesn't have to have a high number of faces: the program will add them as necessary.

C.3 Object Appearance and Attributes

When an object has been built in the Detail editor, often its appearance is lacking when you try to render it. The surface attributes of an object are obviously very important to the appearance of the object! In fact, you can argue that the attributes, brushmaps, and textures on an object contribute more to an object than the geometry of the points, edges, and faces. This is especially true with the complex control things like brushmap altitude mapping gives you over the object's appearance. Even a flat plane can become a beautiful landscape when you have the proper attributes assigned to it.

The best way to set the basic attributes of an object is by experience and experimentation. You can make many copies of an object and assign them slightly different attributes. If you lay these objects out in a grid then use "Quickrender" to get a view of all of the objects, you can quickly see what appearances look best.

In this book, Appendix H on page 195 gives a list of useful base object attributes that I have built up. The best way to use these attributes is to apply them to your object, then perturb them. Vary the parameters to create the effect you are looking for. You are welcome to use the attributes as-is, but the best results come from the tweaking and adjustments you make to each individual object.

C.3.1 Metal and Reflection

Perhaps one of the most important appearances that you want to try to make is metallic objects. Metal is tricky to build; you can't just give the object a shiny attribute and expect the object to suddenly appear like it is made of metal. True, shininess *is* important. The shiny appearance makes the object reflect its environment in a realistic fashion. The Chrome attribute listed in Appendix H has a high shininess value.

But a shiny object is *not* enough for good metal appearances. The smooth metal look comes not from the object, but from the colors and light that it reflects from the world around it. If you ever look at a chrome hubcap, you'll see that the top half of the hubcap has a blue-grey tint from the sky, and a brown-black tint on the bottom half where it is showing the ground. Even if the images the hubcap reflects are soft and indistinct, the colors of sky and ground are being shown.

Our eyes have become used to seeing this reflection of the world in shiny metals. If we don't see the soft colors reflected in metal, the object starts losing its metallic appearance.

But how do you get metal objects to reflect their environment? Do you have to raytrace to get good metal appearances? No, actually you don't. Impulse recognized the importance of global reflection in object appearance, and included several tools to help produce higher quality appearances.

Even in scanline, three items are reflected in objects, and these items can do wonders for making your scene look more realistic. First, the global sky parameters (including the color gradients) are used, so reflective and shiny objects will show the sky colors in their surfaces. Also, ground planes will be shown, and even the textures and brushmaps on the ground will be accurately shown. A chrome ball over a checkerboard ground will show reflected checks even when rendered in scanline. Finally, you can set up a "world reflection map," in the Action editor. This brushmap will be shown in the reflective objects of your scene. If you get a nice landscape picture, you can produce *marvelous* appearances when the items show reflections from this world brush. Even if the global brush is fairly simple, you can enhance the metallic effect enormously.

Global brushmaps are described on page 116.

Even in scenes where there *is* to world to reflect, our eyes expect to see a reflected horizon. If you are building a flying logo animation with metallic letters flying through black space, the addition of the sky and world reflection map is important. Otherwise the objects will reflect pure black, and they will not appear to be metallic at all. But what if you want that black background, and not a sky? Well, there's a slight trick. Add a global brushmap and a nice gradient sky color. Then select the "Genlock Sky" option in the Globals requester. This will make the sky render as black (the default genlock color) but your objects will still reflect the blue gradient sky like you want them to. This trick is very important for the flying chrome-letter-logo animations.

C.3.2 Transparency

When you start playing with raytracing and transparent objects, you might not seem to have as much control as you thought. You cannot just assign a nice blue-white transparency to an object like a couch, set the index of refraction to 1.52, and expect to have the couch suddenly appear as if it were made of glass!

To get nice refraction effects, you often have to experiment, which is unfortunate since raytracing can take a long time. When you want to make a refractive object, sometimes the table of indexes of refraction really isn't a good guide. Unless you are trying to accurately model a lens, often playing with the index can produce more pleasing results. Since most objects have complex geometries, they refract light severely and in complex ways and produce a chaotic image and not just a slightly bent and distorted view. Using artificially low indexes of refraction can often produce more pleasing images. A complex glass vase I built looked best when the index was lowered all of the way down to 1.1. At this value, the image that the vase transmitted was definitely distorted, but the image was still somewhat coherent. Using a "proper" glass index of 1.52 produced an unnatural blur of color. This isn't a fault of Imagine at all! It is just due to the fact that most glass objects in the real world don't have a geometry as complex as most 3D models. (When was the last time you saw a crystal couch?)

Like metals, transparent objects really show their environment. If there is no scene or object *behind* a transparent object, you won't see any interesting refraction effects. By setting up situations where transparent objects are clearly changing the light that passes through them, you can enhance the illusion of a virtual world in your images.

C.3.3 Geometry and Surface Complexity

Surface appearance can dramatically affect the rendered image of a model. For many models, the geometry (the points, edges, and faces that make up the mode) is surprisingly uncorrelated with the complexity of its rendered image. A space station made up of a torus with radial spokes might be able to be built in five minutes or less in the Detail editor by joining some primitives together. With the addition of some high quality brushmaps, it can be turned into a breathtakingly realistic model. It does take skill and practice to learn how to add brushmaps and textures to an object, but these skills are just as important as the knowledge of basic modelling and manipulation commands.

Although many models can be greatly enhanced (or defined!) by additional attribute assignments, many models really do require extra polygons to define small details. A very high quality model of a sailing ship might have ropes and pulleys that can't be faked with a brushmap or texture. Sometimes this calls for your own judgment. Should you build an airlock with an extruded outline, or can you make an altitude brushmap that will fake the appearance of a complex airlock?

This judgment is often made by thinking about how you will be using your model. If you are going to make an animation of a spacesuited man entering an airlock, it is pretty obvious that you'll want to build the airlock with polygons so you get a true perspective view when your camera comes in close to it. If you want the space station as a backdrop to a space battle, it might be a lot easier to just make a brushmap of the outside of the airlock. Since it is almost always easier to make a brushmap of something like a doorway than it is to model it with polygons, you can really benefit by using them as often as you can to save a lot of design work.

In fact, some appearances can't be made by any other method except brushmaps and textures. No matter how many polygons you added, you would never be able to get a realistic appearance of woodgrain onto an object. With a brushmap or texture, it's easy.

C.3.4 Brushmap Control

The toughest part of making a brushmap is in lining up small details so they will appear in the right place on the final model. Though you can almost always scale and rotate the brushmap when you are applying it, sometimes keeping the individual elements in a single brushmap aligned the way you want is difficult. For example, you might have a large space station brushmap that includes the image of an airlock door, some portholes, and some access panels. Though lining up the airlock on your model might be easy, sometimes the other parts of the image (like the portholes) might map into empty space, or at a location you didn't expect. If you scale your brushmap to move parts of the image closer or further apart, the size of your airlock is going to start to change size and shape, something you don't want.

My trick for dealing with flat maps (especially large ones) really works well. You want to build your brushmap with a template that shows the true geometry of your object, and use that guide for determining where to place the different parts of the image. For the side of the space station, all you need is an outline that indicates what areas are "safe" (will map onto your object) and what areas aren't (the map will go into space, or a part you don't want.)

I once made a 767 model and wanted to make a large brushmap to add the windows and doors (as well as the logo) to the side of the airplane's body. Even if I measured the length and height of the model, I would still have problems keeping the individual elements in the proper sizes and relative locations if I tried to guess the positions of the brushmap features.

The solution is really quite straightforward. In the Detail editor, zoom in on a flat (not perspective) view of your model. In my case, I used a side view of the tubular 767 body. Now use a screen grabber (like ADPro's, or the freely distributable ScreenX by Steve Tibbetts) to grab the Imagine screen as an image. In a program like DPaint, you can erase the extra details like the modelling gadgets and leave just the outline of your object. You can now use this outline to decide how to color your object. With a side view of the proper shape and scale, adding the windshield, windows, and logo to the 767 was easy. Since the flat perspective of the view is a straight projection, you can map the painted image right back onto your model with no distortion. I just used a Flat X Flat Z mapping and scaled the brushmap so it just barely covered all of the parts of the fuselage. This insured that all of the details that I added in DPaint would appear in their proper place.

Many people wonder why they can't just take a paintbrush and paint your object. With this method, you can!

This method is particularly effective when you are hand digitizing a real model, as described in the previous sections. Most plastic model kits come with sets of color decals ready to transfer onto the final plastic model. These are *perfect* for digitizing and using as brushmaps! They are all in scale with each other, and are flat so they are easy to scan or point a camera at. Some models come with side/top/front views of your model, and you can digitize these figures and use them as brushmaps just like the outline grabbed from the modeler. There's no such thing as cheating in 3D modelling, so go ahead and use the brushmaps that have already been made for you. Of course, be careful of copyrights.

C.3.5 Making Brushmaps

Prepackaged brushmap packages are very convenient, but they will never be a replacement for custom brushes you make specifically for a certain object. Depending on your object, you might be mapping line drawings, a scanned image, or even another raytrace onto your object.

The tool of choice for making these brushmaps is of course a paint program. Deluxe Paint IV is certainly adequate for almost any manipulation you want to perform, and it allows you to create and edit most types of images. One feature of Deluxe Paint that most people never use is the ability to edit images larger than the screen. When you want to add fine details to your object, you want to use as large an image as possible. Deluxe Paint will allow you to pan around your image and edit a screens-worth at a time. You are limited only by RAM; I once edited a 3500 by 1200 brushmap!

DPaint is discussed on page 189.

Another paint program I use complements the features of Deluxe Paint. When I am dealing with many-colored images (like scanned pictures) I find that DCTV Paint is a terrific tool. Its images allow much smoother gradients with more colors, and the tools it provides for manipulating the images are terrific. It doesn't handle fine detail like line drawings very well, but that is where Deluxe Paint shines. Even ToasterPaint and Light 24 (the paint program for the Firecracker 24) can't match the ease of use of DCTV Paint. The largest drawback of DCTV Paint is that you cannot easily edit images larger than one screen.

DCTV is discussed on page 190.

If you don't have DCTV paint, there is a trick I used for editing images with many colors using plain DPaint. The problem with DPaint is it is limited to a fixed number of colors (or to just HAM) whereas 24 bit or DCTV have an enormously larger color palette. The trick I used is simply to use ADPro to scale the image up to double or triple its original size. Each pixel in the original image becomes represented by 4 or 9 new pixels. ADPro's dithering will represent the color of the original pixel with high accuracy since it can use all 4 or 9 of the new pixels to represent the exact shade. After you edit the image, you can load it back into ADPro and shrink it to its original dimensions. An added benefit is that antialiasing is automatically performed by the scaling back down, so diagonal lines are smoothed and free of jaggies.

This trick requires a lot of RAM (ADPro is memory-greedy) but it allows you to edit 24 bit files with DPaint! If you don't have DCTV or a 24 bit paint program, this might be your only choice with some images.

Appendix D

Textures in Imagine 2.0

Imagine 2.0 comes with many built in algorithmic textures, which allow you to define your object's surface appearance by a set of parameters fed into a procedure which then determines what color each part of your object should be. Using textures is described in detail on page 36 in the Detail chapter. This appendix describes the textures that come with Imagine and how they are used.

Most of the textures apply a layer of color on top of your objects surface and leave part of the original surface showing through. For example, if you use the Checks texture to make black checks on your object, your original surface will show through in the areas where the black checks *aren't* applied. This is extremely useful, since you can use this property to "layer" your textures on top of one another.

Sometimes the word "color" is used in these descriptions to mean "The Color, Reflection, and Filter attributes that are defined by the texture." It is usually obvious by context what this assignment is, but it is easier to say "You can define the color of the woodgrain." than "You can define the red, green, and blue components of the color, reflectivity, and filter values in the woodgrain's attributes."

D.1 Angular

The Angular texture adds five different colors to your object depending where a point is on the object. These different colors blend smoothly into one another, creating a very soft shading. You can think of the different areas that the colors are mapped to as corresponding to the different faces of a cube. You can define the colors of five of the cube's faces (the sixth is the default color of the object), and the areas near the joints of the cube faces are smoothly shaded, blending into each other. The faces are identified by direction the face lies in. All of the directions are of course relative to the texture axis.

Pos. X RGB The color mapped to the region in the positive X direction.

Neg. Y RGB The color mapped to the region in the negative Y direction.

Pos. Y RGB The color mapped to the region in the positive Y direction.

Neg. Z RGB The color mapped to the region in the negative Z direction.

Pos. Z RGB The color mapped to the region in the positive Z direction.

D.2 Bricks

Bricks adds (as you might expect) color in a brick pattern. Rectangles are added to your object in a three dimensional grid, with the size of each rectangle and the width of the gap of between the "bricks" definable. Actually, Bricks doesn't add bricks, it adds the *mortar*, the cement between the bricks. The original object surface shows through as the bricks, so (for example) you could apply the wood texture to your object, then bricks, and your object would look like it was made of wooden bricks. Remember that this is a three dimensional texture, so the bricks go *in* as well as up and across. Adding bricks to a curved surface will work, but it won't necessarily look realistic since the bricks will not curve with the object.

X,Y,Z Size The size of the bricks. Standard building bricks usually have sides roughly in a 6:3:2 size ratio.

Mortar Size The width of the colored mortar. On buildings I've looked at, this is usually about a fifth of the Z height of the bricks.

X,Z shift with Z,Y When bricks are stacked, you can place them directly on top of each other, or offset each one from the previous layer. Builders usually add a new layer of bricks offset from the previous layer by half of the length of the brick, so you might set the "X shift with Z" parameter to half the X width of the brick. Similarly, "Z shift with Y" will change the offset height of adjacent rows of bricks.

Mortar Color, Reflect, Filter RGB The attributes of the mortar.

D.3 Camo

Camo adds random spots of color onto your object. These spots are opaque blobs of color that look much like camouflage (hence the clever name.) You can vary the spacing of these blobs of color, and apply up to four layers of different colored spots. Your original object color is shown in the areas where no spots are placed.

Spot Spacing Changes the rough scale of the spacing between the spots. For something like an airplane, a number about 1/10 of the length of the object works well.

Random Seed A number to randomize the algorithm. Enter your birthday!

RGB Color 1-4 Allows you to determine the RGB colors of up to four sets of spots. The spots are added in order and might overlap, so you might have spots of color # 3 on top of color #1.

# of colors The number of spot colors to use. If you select "1" only one color will be applied. "4" will apply all four colors.

Spot area Controls the density of spots. If you lower this number to a small value like 0.05, you'll see very few spots. At the extreme of 1.0, strange effects take place and you get random crisscross areas. A value like 0.5 works well for most applications.

You can make some funky art-deco patterns by setting the "Spot Area" to a high value like 0.95 or 1.0.

D.4 Checks

Checks are perhaps the classic algorithmic texture. It applies a three dimensional grid of solid checks to your object. You can think of a stack of colored cubes arranged in a checkerboard grid that alternates cubes with holes. This alternation extends in *all three* directions.

Checks can be applied to any object, but sometimes you might get results you don't expect. A sphere with the Checks texture on it looks quite strange, since the curved surface makes the checks seem to appear at odd places. If you are looking to build the classic red and white checkered Amiga sphere, you are better off making a red and white checkerboard in a paint program and using a brushmap to apply it to the sphere.

If you are applying Checks to a plane, you probably want to use the Checks2 texture.

Brushmaps are discussed on page 39.

Check Size The size of one check.

Color, Reflect, Filter RGB The attributes of the checks.

D.5 Checks2

Checks2 is very much like Checks except it is a 2D texture; the texture only changes along two directions. Instead of a checkerboard pattern of cubes, the texture is like a checkerboard pattern of solid columns extending up and down infinitely along the texture's Z axis.

Check Size The size of one check.

Color, Reflect, Filter RGB The attributes of the checks.

D.6 Disturbed

Disturbed adds a rippled appearance onto your object as if it were made of water. This texture doesn't affect the color of your object, it instead makes the light reflect off of the object's surface as if the surface was physically rippled, much like the way altitude maps work. The disturbed appearance can be thought of as the interference between a couple of sources that are making ripples simultaneously. Disturb isn't easy to animate; instead you probably want to use the "Ripple" F/X described on page 182.

Amount < 1 The amplitude of the effect. A value of 0 will have no effect on your object, a number like 2 will muddy it into confusion. 0.5 works well.

Wavelength The rough scale of the ripples that are made. A number about 1/10 of the length of your object works well.

X Separation The distance between the ripple sources that are causing the disturbed pattern. A value roughly equal to the wavelength works well. A value of 0 will create no interference between the sources, and the disturbed effect becomes very regular.

Small < 1 A perturbation that adds a little character to the disturbed pattern. A value much greater than 1 will make a confusing looking texture. A value of 0.5 works well.

D.7 Dots

Dots will add a regular gridwork of spots (actually spheres in a 3D grid) to your object. This makes a regular pattern of spots on your object. The spots will appear at different sizes depending on how close your object's surface passes to the center of a sphere.

Dot Spacing How far apart the spots are located.

Dot Radius The radius of each spot. Remember you might see sections that are smaller if your object cuts through a "sphere" of color but doesn't pass through the center, a common occurrence.

Color, Reflect, Filter RGB The attributes of the spots.

D.8 Grid

The Grid texture is very straightforward. Grid adds a cross-hatch pattern of lines to your object, arranged in a smooth lattice shape. You can think of Grid as being the Brick texture with cubical bricks and no offsets of adjacent rows. Like Bricks, Grid colors the mortar, or boundary between the cubes that it stacks. You can change the size of the cubes as well as the width of the gap between them.

Grid Size The spacing between adjacent lines. (The size of the cubes.)

Line Size The width of the lines.

Color, Reflect, Filter RGB The attributes of the gridlines.

D.9 Linear

Linear is perhaps the most useful texture of all, just because it is so simple and versatile. Linear allows you to add color to one half of your object, with a gentle or abrupt transition between the two halves. This is done by specifying (with the texture axis) a point where the color of the object starts changing. Any parts of your object in the direction of the the positive Z texture axis will be colored with the attributes specified by the Linear texture. A "transition width" can be set which defines how sharp the transition between the original color and the linear color is. Any surface within this transition is colored with a gradient that smoothly changes from the original object color to the color being applied by the texture.

Linear is very versatile since it is so simple and elegant. You can position the texture axis so that your *entire* object is in the region that is colored by the texture (The axis would be completely on one side of the object, with its Z axis pointing towards the object.) Then, if you set the "Z transition width" to a length equal to the length of the object, your object will be completely within the transition between the two colors, so it will be colored with a smooth gradient. You can even apply a texture like Wood first, then use Linear to "fade" the woodgrain in over a distance. You can also add several copies of linear to add multiple colors to your object, so you can have up to 5 regions (the original color, plus the four areas defined by the Linear textures.)

Transition Z Width Defines how sharp the transition between the default object colors and the colors added by Linear is. This is the actual distance it takes to "fade in" the texture. A value of 0 will make the transition abrupt and complete.

Color, Reflect, Filter RGB The attributes added to the object in the region in the direction of the positive Z texture axis.

D.10 Pastella

Pastella adds random spots of color to your object. These spots are very "soft" in that they are fairly blurry and they don't have sharp colors; they are all pastels, colors which are not highly saturated neon hues.

Detail Size The rough size of the spots of color on your object.

Random Seed A number to randomize the algorithm. Enter your mother's birthday.

Color, Reflect, Filter RGB The attributes of the spots. Colors in particular will not be used exactly, but will rather be "whitened" into a more pastel shade.

D.11 Radial

Radial is very similar to the Linear texture. It allows you to add a color transition or gradient to your object. Instead of splitting your object into halves along a plane like Linear, Radial splits your object based on a radial distance from the texture axis. Everything outside of a definable radius will be colored with the attributes defined by the texture. You can also create a soft transition between the original color and the surface color by setting up a "transition width" which makes a gradient between your beginning and ending colors.

Start Radius What radius from the center of the texture axis the color transition begins at.

Transition Width Defines how sharp the transition between the default object colors and the colors added by Linear is. A value of 0 will make the transition abrupt and complete.

Color, Reflect, Filter RGB The attributes of added texture.

D.12 Spots

Spots is somewhat like Camo, in that spots of color are randomly added to your object. The spots in Spots have a different character, however. They look more like rings or lines twisted into irregular shapes, and filled with a second color. You can set both the line (outline) color and the spot's interior color.

Neither Spots nor Camo really adds circular areas of color like Dots.

Spot Spacing The rough distance between spots.

Random Seed A number to randomize the algorithm. Enter your cousin's birthday.

Color #1 RGB The color of the outlines around the spots.

Color #2 RGB The color of the spot interiors.

D.13 Waves

Like Disturbed, Waves does not change the color of your object, but makes it look like it is made of liquid. Waves is a much better behaved texture than Disturbed, and is really designed to be animated. Waves doesn't actually change your object's shape; that effect can be performed by the "Ripple" F/X described on page 182.

Waves adds a regular, repeating bump to your object, like a series of waves coming into a beach. You can change the height of the waves and the separation between adjacent waves. You can also animate them by using the "Distance Traveled" parameter. The waves move along in the direction of the X texture axis.

Wavelength The spacing (length) between subsequent waves.

Amplitude The apparent height of each wave.

Distance Traveled The distance the waves have moved. If you increase this number with time, you can make the waves move smoothly. Use texture morphing to automatically increase this parameter. (Texture morphing is talked about on page 103.)

D.14 Wood

Wood is another classic texture algorithm in computer graphics. It adds a woodgrain pattern to your object that can be quite realistic. It makes grain by modeling wood as a set of concentric cylinders of grain (a good model, that's what a tree is!) You can perturb the grain pattern to make it look more random and realistic, you can also change the thickness and attributes of the grain.

The direction of these "grain cylinders" is along the texture's Y axis. Wood looks best when this axis is nearly, though not quite, parallel to the longest object dimension. This gets you nice grain cross sections, and looks more realistic. Sometimes very small rotations of the wood's texture axis can make a dramatic change of the appearance of the woodgrain on your object, so experimentation is worthwhile.

Color RGB The color of the wood grain. Grains are usually darker than the wood itself and tend to be redder, but you can choose neon blue grain if you like.

Ring Spacing The normal distance between different rings in the wood. A smaller number will increase the number of rings but make each ring thinner.

Exponent The thickness of the rings. When this value equals 1.0, the grain rings are about the same thickness as the alternating bands of wood. A value of 20 will make the rings very thin. A number from 5–10 works well.

Variation How regular or irregular the wood grain is. A number like 0.1 will produce a very regular (very boring) pattern, while 3.0 will make the grain look more like twisted burl. I like the looks of woods with values in the range of 1.0–2.5.

Random Seed A number to randomize the algorithm. Enter my birthday.

Appendix E

F/X in Imagine

F/X are special effects you can apply to objects in the Action editor. They are similar to textures in the fact that they operate on your objects and produce an advanced effect from a procedural algorithm. Instead of changing surface color, however, F/X allow complex manipulation that let the geometry of objects to change, usually in an animation. The use of F/X is described on page 112. The F/X included with Imagine 2.0 are described in this section.

E.1 Boing

Boing will make an object squeeze itself along one axis. You could use this effect to make a bouncing basketball that deforms when it hits the ground, or to have a starship stretch and distort when it goes into warp drive. This F/X really doesn't do anything you can't do with keys to control the object's scale, but it is convenient especially for multiple contractions.

The version of "Boing" that Impulse distributed on many early copies of Imagine 2.0 is broken. *It will do nothing to your object at all. There is a replacement on the disk with this book.*

When applied to an object, the actor will be scaled down to a certain fraction of its size in one direction, then back up to its original size. You can also scale the object up by specifying a "Shrink" to a value greater than 1.0. If you tell the F/X to perform the Boing a number of times, the scaling cycles will occur consecutively. The number of frames that the F/X is applied to controls the amount of time it takes for the object to contract and re-expand.

XYZ Axis Which direction to contract the object along. A bouncing basketball would crunch in the Z direction, but a starship would probably stretch along its Y axis.

Squash To The direction of the squash. A center squash is a simple scaling, a squash in the + or − XYZ will make the squash "flow" in that direction.

Shrink Factor How much compression or expansion your object is scaled by. A value of 0.1 will contract the actor to 10% of its original length at the peak of the Boing. A value of 2.0 will stretch the object to 200% of its original length.

Number of Times The number of times the contraction/expansion cycle is performed during the frames the F/X is applied.

E.2 Explode

Explode is perhaps the most overused F/X, but it sure is fun. Explode will take and literally break your object up into its component triangles, and send those triangles spinning away from the center of your object. Although the spinning triangles do break your object into parts that fly away from each other, it is more of an abstract, art-like, explosion as opposed to the nice shrapnel-studded plumes of fire and smoke you see in science fiction movies. Nonetheless, it is still occasionally useful to use.

Impulse did not include Explode with many (or all) of its Imagine 2.0 disks. You can use the Imagine 1.1 version of the F/X (if you have it) as a temporary substitute.

You can control the type of explosion, which is characterized by the direction the object's component triangles fly. A spherical explosion makes the triangles fly directly away from the object's axis, the most common use. You can also make radial explosions, which is more like an exploding tube, where the triangles fly away from a central line, not point, or a linear explosion, which makes the triangles move in one single direction.

You can have the triangles rotate as they move, creating the effect of spinning shrapnel. You can also scale the triangles to make them disappear as they move away.

Another Explode option lets you make the triangles return to their starting positions, useful when you want to form an object from flying shrapnel. You can also reverse an explosion: by making it run in reverse you can "unexplode" an object.

Spherical/Radial/Linear Determines the direction the triangles of your object fly away (described above). Spherical is the most common option.

XYZ axis With a radial or linear explosion, you have to specify which axis the explosion occurs along. This doesn't apply for spherical.

Explosion Distance How far the triangles should move over the span of the F/X.

Expansion Angle The width of the angle that "shrapnel" is ejected from.

Triangle Scaling The final size of the triangles. A value of 0.0 will make the triangles disappear completely by the end of their motion. 1.0 will keep their sizes constant.

Minimum # of rotations When you want triangles to spin as they move, you can make sure all of them rotate at least this many times.

Max # of rotations Sets the maximum number of spins for a triangle.

Random Seed A number to randomize the texture. Use your telephone number.

Return to original positions Flag to return the triangles to their original positions in the object.

Reverse explosion timing Flag that makes the explosion run in reverse, making shrapnel unexplode into a solid object.

E.3 Fireworks

Fireworks is a simple extension of the Explode F/X. Fireworks works exactly the same except you can have the triangles drop in the Z direction as if gravity were pulling them down. You also have the option of making the faces sparkle, which makes the faces easy to see as opposed to the normal triangle shading.

Distance to Fall by End Determines the distance the flying particles should drop to the "ground" during the explosion.

Make Faces Sparkle Flag to make the individual faces sparkle with color.

E.4 Flash

Flash allows you to make a blinking object. The blink is not caused by a light or by changing the color of the object. Instead, the object is toggled between its normal appearance and an unshaded Bright appearance. This unshaded appearance means that no shadows will be applied to your object and no shading will be performed; you'll just see the flat color of your object. This is actually somewhat of a sterile and boring appearance, but adequate for something like a blinking control panel. For an object like a stoplight where you want an actual light to appear, you are better off just adding a new light at the proper frame using the Action editor.

The Bright attribute is described on page 33.

It is very simple to control Flash. An object with the Flash F/X will be shaded normally for a certain number of frames, then become Bright for another number of frames. You can start the object in either state.

On Frames The number of frames the object should be shaded with Bright in each Flash repetition.

Off Frames The number of frames the object should be shaded normally in each Flash repetition.

Start On/Off A toggle to start the object in either state when the F/X is first applied.

E.5 Grow

Grow allows you to extrude a tube just the Extrude command in the Detail editor, except you can animate the extrusion as it occurs. If you make a spline path in the shape of a logo written in cursive, you can have the letters "written" out in time in a gleaming tube of chrome. (Extrude is discussed on page 50.)

Grow is actually pretty easy to use. You have to make the spline path and the outline that you want to have extruded along the path. You can use the Detail command "Extrude" to test the path and outline and make sure the final extruded object appears the way you want it to. Group the original unextruded object to the spline path (the path being the parent!), then save this group as a file. Load this group in Stage or Action as an actor, and apply the Grow F/X to it. When animated, the outline will be extruded in time just like you would expect.

You have several parameters that can control the extrusion. All of them except one are the same options that a normal extrude in the Detail editor gives you.

Y Rotation An angle that the outline should be rotated along as it is extruded. 720.0 is two full revolutions.

X, Z Scaling Scales the outline in its X or Z direction as it is extruded. For example, 0.5 will halve the size, 1.0 will keep it constant, 2.0 will double it.

X, Z Translate Moves the outline to one side or another as it is being extruded.

Align Y to path Keeps the outline faced so that it is always facing the direction it is being extruded; most of the time you want to keep this option selected.

Keep X in path's plane Keeps the X axis of the outline level with the original path axis; the outline won't pitch up and down.

Mirror ends Make the outline flipped at each end so the endcaps are identical.

Time Reversed A new option: you can ungrow something and have the extrusion be "undone" in time.

E.6 Ripple

The Ripple F/X will make the surface of your object move as if water ripples were passing through it. You can make a radial ripple, like a stone being thrown into some water, or a linear ripple, like a flag waving in the wind. Unlike altitude maps and the Disturb and Wave textures, your object surface is actually displaced, creating true 3D ripples that can cast shadows and be seen in profile. Since the surface is moved by moving the object's points, the finer the mesh on the surface of the object the better the ripple effect will be.

Radial around Z Allows you to select a ripple like a rock dropped into a pool. The rings will expand out from the Z axis of your object.

Linear along X Allows you to make a flag-waving like ripple, or an ocean wave. The wave travels along the object's X direction.

Wavelength The distance between subsequent ripples.

Z Amplitude How much the points are displaced; the height of the ripple.

Travel Distance How far the ripples travel before dying out.

Ripple Count How many ripples should be made. For a waving flag, you probably want many ripples, but a thrown stone in a pond only makes two or three.

E.7 Rotate2.0

The Rotate F/X is an alternative method of rotating an object over time. This F/X does not so anything that could not be done with alignment keys from the Action or the Stage editor, but sometimes it is convenient when you want to spin an object for a long time.

The Rotate F/X will turn your object a certain number of degrees over the course of the F/X. You can specify a large number of degrees (like 720.0) to make the object rotate all of the way around.

Impulse says this F/X is a little touchy, and I agree with them. If you understand and can use key alignments, use them for rotations of 360 degrees or less. I only use Rotate when I need to spin an object like a top many times over a large number of frames, where the keys would take a long time to set up.

Key alignments are discussed in depth starting on page 100.

X,Y,Z Axis Which axis to rotate around.

Degrees The total angle to rotate over the course of the F/X.

E.8 Tumble

Tumble is similar to Rotate in that it rotates your object over time. Unlike Rotate, Tumble selects a random rotation axis and spins the object a random number of times. This randomness might be useful when you have many objects that are all chaotically spinning and you don't want to have to make a random rotation for each object.

Random/Z Axis Allows you to choose either a random axis or the Z axis to perform the rotations around.

Minimum # of Rotations The minimum number of rotations the object will undergo over the course of the F/X.

Maximum # of Rotations The maximum number of rotations the object will undergo over the course of the F/X.

Random number seed A number to help Imagine choose the random rotation. If you have many tumbling objects, give them all *different* numbers.

Appendix F

Brushmap Mathematics

Brushmaps are described in the Detail chapter in section 3.4.3 on page 39. The descriptions of the brushmapping options there are accurate, but somewhat vague. This appendix describes in detail how brushmap axes map an image onto an object. It uses some mathematics, but if you remember your high school trigonometry you should be able to follow it.

This technical discussion is valuable since is describes exactly how brushmaps are applied, and you can use this information to help plan your brushmaps and avoid surprises, especially when you use "Wrap Flat" (Cylindrical) and "Wrap Wrap" (Spherical) brushmaps.

This appendix is not by any means necessary reading to use brush maps. It is a technical explanation to help advanced users understand the brushmap coordinate mapping.

F.1 Brushmap Coordinate Mapping

The renderer's task of applying a brushmap to an object is one of coordinate transformation. Given an X, Y, Z location on the object, what transformation will map that location onto the X, Y coordinates of the brushmap? The brushmap coordinates are often called U and V to avoid confusion with the object's coordinates.

With all types of brushmaps, the object's coordinate system isn't even used. Instead, the brushmap axis is used to determine the coordinate system for mapping. This means that a translation in the brushmap axis will cause a similar translation in the image projected onto the 3D object surface. When I refer to the X, Y, Z coordinates of the object, I really mean the coordinates after the transformation of the brushmap axis is already applied.

A convention in this mapping problem is that the brushmaps U, V coordinates extend from 0 to 1, and the brushmap axis size also defines a coordinate interval from 0 to 1. Referring to a point with a coordinate value less than 0 or greater than 1 on the brushmap implies the mapping is off of the image of the brushmap and no assignment should be made.

When the "Repeat" gadget is used to make an infinite tiling of images, the final "applied" U and V coordinates are just the modulus remainder with 1.0: that is, the fractional part of the coordinate. 1.1 and 10474.1 will both map to 0.1. This will create the tiling since the coordinates of the brushmap will make a transition from 0.9999 (the right end of the brushmap in U) to 1.00000 which is mapped to 0.00000, the left end.

F.2 Cartesian Coordinates (Flat Mapping)

Flat mapping is very straightforward. There is a direct mapping from $X \mapsto U$ and $Z \mapsto V$. This causes the image to be shown directly as a flat projection into the object with no distortion. The "depth" control of the Y axis restricts the mapping to occur only if the Y coordinate is between 0 and 1. When the Y coordinate is larger or smaller, the object location is "outside" the region that the brushmap is applied to.

F.3 Cylindrical Coordinates (Flat Wrap Mapping)

This is where the coordinate system description becomes useful. The Wrap/Flat and Flat/Wrap brushmapping techniques are simply a change of the X, Y, Z Cartesian coordinate system into a cylindrical coordinate system of R, θ and Z, where $R = \sqrt{X^2 + Y^2}$ and $\theta = \arctan(Y/X)$. (This is for Wrap Z, Flat X.) If θ is remapped to the range 0–1, the mapping to brush coordinates is simply $\theta \mapsto U$ and $Z \mapsto V$. The angle of the point using the brushmap axis as an origin determines where on the map the image is sampled from. The linear mapping of Z is straightforward.

This mapping isn't actually accurate, since Imagine seems to ignore the true Y axis scaling and uses a scale the identical to the X size; this means that the mapping is truly cylindrical rather than an oval shape, which is probably a good idea. Notice that this means the X size of the brushmap axis really doesn't have any effect! This is actually the case; we've all been strung along for a long time thinking it mattered!

F.4 Spherical Coordinates (Wrap Wrap Mapping)

The transformation for Wrap Wrap mapping is just spherical coordinates. The X ,Y, Z values are transformed into two angles and a radius by the relations $R = \sqrt{X^2 + Y^2 + Z^2}$, $\theta = \arctan(Y/X)$ and the azimuth $\phi = \arctan(Z/R)$. The 0–2π values of θ are mapped to 0–1 and the $-\pi/2$ to $\pi/2$ value of ϕ is mapped to 0–1. The transformation to brushmap coordinates is now just $\theta \mapsto U$ and $\phi \mapsto V$. Again, the Y axis scale seems to be fixed with the X scale to keep the coordinate system square; the Y axis really doesn't make any difference in this transformation.

The knowledge of the mapping transformations allowed me to produce a very successful brushmap for applying onto a sphere. I wanted to build a basketball with the curved lines and the small irregular pits on the surface. It was crucial to my application that the basketball be seamless and undistorted. I wrote a program in C that characterized a basketball; given a point on a sphere's surface (measured with θ and ϕ,) it would return a value indicating whether the location was on one of the ball's stripes, on one of the raised pips, or just plain rubber. I then looped over an

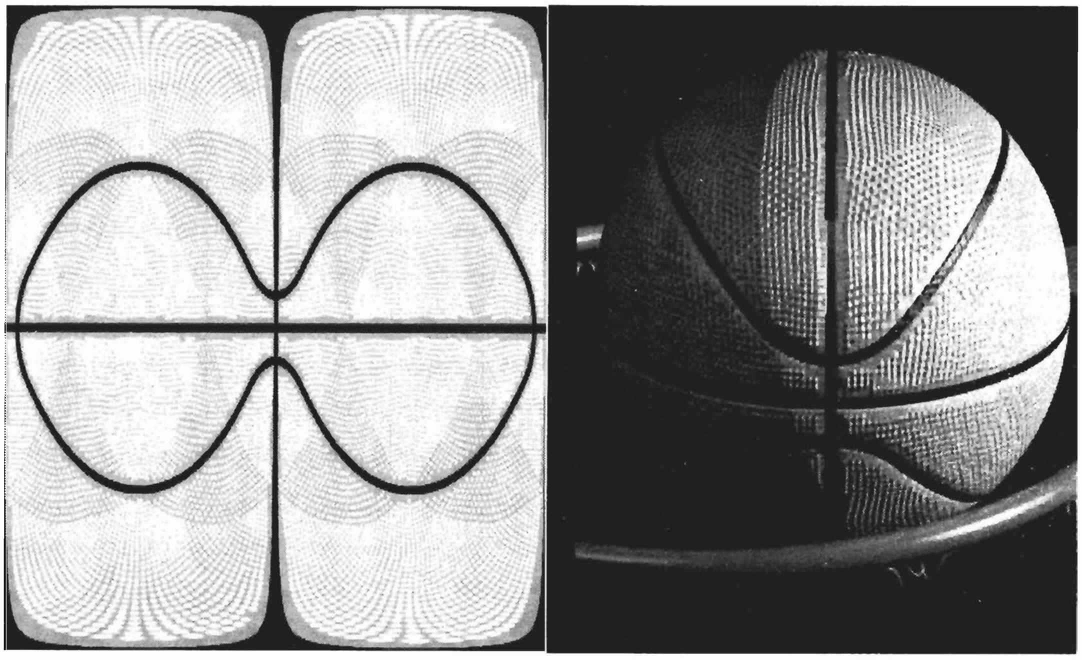


Figure F.1: A basketball wrap-wrap brushmap and the rendered output.

array of θ from 0–2π and ϕ from $-\pi/2$ to $\pi/2$. I sampled the basketball at those coordinates, and stored the result in a rectangular grid. I wrote the grid out as an IFF24 and used it as an altitude map, with a companion map without the pips for the color map. Figure F.1 shows the altitude map of a basketball in spherical coordinates, as well as a rendered version of the ball itself. Notice the broadening of the stripes at the poles, and also notice the seamless appearance of the rendered basketball.

Knowing how brushmaps are transformed to object coordinates is very useful indeed when you are trying to get accurate results. I hope that this technical digression will help some people who are trying to understand how to control object appearances.

Appendix G

Other Programs and Hardware

Imagine is all you need to start building objects, rendering pictures, and viewing animations on your Amiga. Several other programs or pieces of hardware complement and support Imagine and might give you the ability to build and output even higher quality pictures and animations.

I do not work for any of the companies listed here except for Apex (obviously!) and none of these companies asked for me to mention them. These opinions are very subjective, but I've found them all useful when dealing with 3D and the images it produces. I have to make a special mention of both ADPro and Deluxe Paint as being nearly essential companions in producing most high quality renderings.

G.1 Deluxe Paint

Deluxe Paint is the most popular paint program on the Amiga. It is a very functional bitmap editor, with all of the standard drawing tools you would expect in a paint program. It allows for fast and easy brush manipulation for overlay compositing of images, and can handle almost all picture editing tasks. It also is a complete animation editor which allows complex animation production and assembly.

This program is indispensable for rendering, mostly for making quick brushmaps. For example, if you want to make a roadway sign in Imagine, the best way to do it would be to use Deluxe Paint to make a brushmap of white letters on a green background, perhaps with a nice border around it, then mapping this image onto a plane. Trying to produce a simple effect like that without using a paint program would be a horrible waste of time.

There *are* many other paint programs that will perform the same capabilities of Deluxe Paint, but I have not found another that is as consistently useful.

G.2 The Art Department and ADPro

A very useful utility for dealing with IFF pictures on the Amiga is *The Art Department (TAD)* from ASDG Inc. This utility can read and write images in many different formats, including 24-bit IFF and Impulse RGBN format. In addition, *TAD* has many image processing capabilities that can help you adjust the color balance of your images. You can also scale images and render the pictures in almost any Amiga display mode or resolution.

TAD's big brother, The Art Department Professional (ADPro) is much more useful. It allows for saving images in a wide variety of formats, including formats used on other computers like MS-DOS and Macs. There are also a wide variety of operations that can be performed on images, such as adding text, cropping, applying filters, and scaling. Images can be loaded in a variety of formats and can be composited with each other to produce overlays on backgrounds, or averaged pictures. ADPro also loads and saves the JPEG compressed image format, also discussed in the Tricks appendix. ADPro also is completely AREXX drivable, which can help with processing entire batches of images or performing repetitive or complex tasks.

The Tricks appendix talks about motion blurring using ADPro.

Like Deluxe Paint, I find ADPro indispensable. Unlike Deluxe Paint, there really is no competition for the high-end image processing market. Luckily ASDG seems committed to continually improving ADPro.

G.3 Firecracker 24

Impulse builds a 24 bit display board for the Amiga. This allows you to view your pictures in all of their 16.7 million colors. It is hard to describe the difference between an image shown with Amiga HAM and the same image in full 24 bits; it is truly amazing.

The Firecracker is a plug in board for Amiga 2000's or 3000's. It outputs interlaced RGB that can be shown on a standard 1084 or 1950 monitor. The board does *not* use the video slot, which means you can put one in a machine with a genlock or Toaster already in the video slot.

The Firecracker 24 is completely supported by Imagine, which is of considerable benefit. Images can be directly displayed on the board from the Project menu. Quick-render output can also be shown on the board, which is even more useful.

The Firecracker comes with a serviceable real-time paint program. This paint program is unique in that you can *load Imagine 3D objects and render them,* producing a 2D brush that can be composited into an image.

G.4 DCTV

DCTV is a hardware add on that uses a standard Amiga image to encode a high-color image. The hardware "sees" the special image being displayed and decodes it into an NTSC composite image. The quality of this image can be stunning if you are showing a digitized image or colorful rendering. The DCTV trades off sharp images for color, but for most pictures, this trade is well worth it!

DCTV can also digitize color images using a slow-scan method (it takes several seconds to digitize a frame.) If you have a video camera, this is an excellent way of getting images into your computer since you can just point a video camera at a scene to make an instant brushmap. DCTV has a paint program that is just stellar, though it is more suited to editing many color images than single bitplane line drawings. This makes it a nice complement to Deluxe Paint.

DCTV can also animate high quality images at about 15 frames per second or lower quality images at a full 30 fps. (The same rate the Amiga can animate 3 and 4 bitplane hires images.) If you don't have a single frame recorder, this is the next best thing when you get tired of HAM.

G.5 Portal

Although Portal is not a program, it is still a useful resource for anyone interested in Amiga graphics. Portal is a service you call up on the telephone with a modem. Once online, you can issue commands to view messages about many topics (including Imagine!) download files (including megabytes of Imagine objects!) and talk to other people in a live chat screen (about things like Imagine!) The fees for using Portal are quite modest for an online service, and are fixed rate (you pay a fee once a month for all you want to call!) I am often logged into Portal for its nightly chats at 7:00 PST, and Wednesday nights are particularly good times to log in since the topic is Amiga graphics. You can get your Imagine questions answered in real time by an expert!

Portal Communications
10385 Cherry Tree Lane
Cupertino CA 95014
408–973–9111 (voice)
408–725–0561 (modem)

G.6 Studio Amiga

Studio Amiga is a modem bulletin board that specializes in 3D discussion and files. Run by sysop Jody Reimers, the system is free to use other than a long distance telephone call. There are many fewer files and messages on this BBS than on Portal, but *all* of the discussion revolves around 3D so it is well worth logging in. It might seem daunting to call long distance to a bulletin board, but the information on this board is usually well worth it. I highly recommend calling this board at least once.

Studio Amiga
817–557–2111

G.7 Pixel 3D 2.0

Pixel 3D by Axiom Software is a multi-purpose 3D object utility program. It is primarily a brushmap conversion program, automatically tracing brushmap outlines and making faced 3D objects from color bitmaps. Though Imagine has its built in brushmap (and font object) converter, Pixel 3D does have one neat feature that adds bevels to extruded images.

Pixel 3D also can load and save objects in a variety of formats, which makes it useful for object conversion. Unfortunately, the program cannot read Imagine grouped objects, a serious failing. You can get around this by joining your object into one large single-axis object, but this will lose all of the group information that makes object manipulation and attribute assignment easier. Still, if you want to convert single, ungrouped objects to or from Imagine format, Pixel 3D will convert to and from Lightwave, Videoscape 3D, 3DPro, Sculpt 3D, and Turbo Silver formats.

G.8 Vista Pro

Vista Pro is a program from Virtual Reality Laboratories for making realistic landscape scenes. The program can use real-world height data to produce views of real landscapes, or you can generate random fractal terrains. The program has controls to change the colors of the terrain, rivers, sea level, and lighting. It can actually render scenes with a built in renderer, which can be used as nice backdrops in Imagine scenes.

Or reflection maps!

The program can also output a 3D model of the terrain it displays. This model is in Turbo Silver object format, which Imagine can read, so you can have it build actual landscapes for your own scenes. The landscapes are formed by a mesh of colored triangles, and can be positioned, edited, and manipulated just like a normal object. These landscape objects tend to be very large, but you can tell Vista what complexity to output them at so you don't overburden your scene.

G.9 The Video Toaster

NewTek's Video Toaster provides many useful capabilities that can indirectly be used by Imagine. It can be used to grab frames of video that can be used as brushmaps. It has an NTSC (television resolution) 24-bit display, which, although as not as high resolution as the Firecracker 24, can still display excellent renditions of pictures rendered by Imagine. The Toaster comes with its own 3D rendering software, Lightwave 3D, and objects from Imagine can be converted to and from the Lightwave format using a program like Interchange or Pixel 3D.

G.10 TTDDD

Imagine's object format is called TDDD, an acronym for "Three Dimensional Data Description." TDDD is a compact binary description of the shape and color of the object or group it represents, as well as the textures and brushmaps that are applied to the objects.

Although Impulse currently does not offer any support for reading and writing these objects with anything but Imagine, there is an independent shareware program called TTDDD (*Textual* Three Dimension Data Description) which converts the binary object format into an ASCII text file and vice versa.

If you can program in an Amiga language (like C, BASIC, even AREXX), it is easy to write a program to input and output these text files and convert them to usable Imagine objects. With this ability, you can generate objects algorithmically or write your own conversion software from another 3D object format. Some of the applications this utility has been used for include making an animated waterfall, building a geodesic dome object, and building molecular "ball and stick" models from chemical formulas. Other handy utilities are included, such as one the creates a PostScript diagram of the top, front, right, and isometric views of any Imagine object.

This utility is not for most users, but if you are a programmer and want to play with generating these types of objects, TTDDD might be a valuable tool. The program (with some documentation and examples on disk) is available directly from the author, Glenn Lewis. He suggests a $10 shareware donation for the package.

Glenn M. Lewis
8341 Olive Hill Court
Fair Oaks, CA 95628
(Internet E-mail address: glewis@pcocd2.intel.com)

G.11 Surface Master and Map Master

Two programs are distributed through a company known as Computer Imagery. These programs, *Surface Master* and *Map Master*, are designed by artist Lou Markoya, and interactively demonstrate different Imagine abilities. Surface Master uses pictures of a rendered sphere to show different attribute combinations and effects. It also demonstrates the use of some of the textures and shows how they can be combined. Map Master shows the different type of brushmaps and includes many high resolution pictures suitable for altitude mapping and several other effects. These two programs are *very* useful if you can't visualize what the different attribute parameters do, or how different brushmaps work. *Surface Master* also has a small set of very useful predefined surfaces like chrome and brick that you can use on your own objects.

Computer Imagery
49 Walnut Ave.
Shelton CT 06484

G.12 Apex Software Publishing

Apex is a new company specializing in Amiga 3D and high-end graphics. This book is its first product, but expect to see more coming in the future! Apex also offers professional custom object, scene, and animation creation, and phone and personal visit consulting services. Apex consists of a very small but knowledgable staff (well, it's just me, I'm pretty knowledgable, and I'm 5' 8".) Feel free to write for a flyer or to heckle me about my overuse of semicolons as punctuation.

Apex Software Publishing
405 El Camino Real Suite 121
Menlo Park CA 94025

Appendix H

Useful Attributes

Name	Color			Reflect			Filter			Specular		
	R	G	B	R	G	B	R	G	B	R	G	B
Aluminum	114	114	143	138	138	138	0	0	0	255	255	255
Black Skin	13	94	82	0	0	0	0	0	0	137	96	111
Chrome	114	114	143	210	210	210	0	0	0	255	255	255
Glass	38	38	83	53	53	60	114	113	167	255	255	255
Painted Metal	240	240	240	25	25	25	0	0	0	140	140	140
Plaster	219	205	186	0	0	0	0	0	0	210	220	220
Red Paint	183	73	43	20	10	10	0	0	0	191	153	149
Sandstone	152	94	70	0	0	0	0	0	0	147	96	74
Ship Grey	99	99	99	10	10	10	0	0	0	135	145	160
Solar Panel	59	59	59	54	54	54	255	255	255	255	255	255
Water	130	130	140	100	100	100	240	240	250	255	255	255
White Skin	215	143	122	0	0	0	0	0	0	159	133	134
Wood	158	85	27	0	0	0	0	0	0	180	160	150

Name	Dither	Hard	Rough	Shine	Index
Aluminum	0	150	0	0	1.00
Black Skin	255	23	35	0	1.00
Chrome	0	190	0	0	1.00
Glass	40	192	0	0	1.66
Painted Metal	0	140	0	0	1.00
Plaster	200	35	0	68	1.00
Red Paint	235	200	0	0	1.00
Sandstone	255	23	130	0	1.00
Ship Grey	255	162	0	20	1.00
Solar Panel	0	110	0	0	1.00
Water	120	150	0	0	1.33
White Skin	255	23	35	0	1.00
Wood	255	100	0	0	1.00

Appendix I

Command Reference

This section briefly describes each of the menu commands available in Imagine, as well as the pages where that command is described in context with its function.

I.1 The Cycle Editor

Object Menu

Page	Command	Description
79	New	Abandon any current object and start a new cycle
84	Load	Load a previously created cycle object, or convert a grouped object from Detail into a cycle
79	Save	Save the current cycle object

Pick Menu

Page	Command	Description
77	Group	When a link is manipulated, makes a link's children move with it
77	Object	When a link is manipulated, makes a link's children stay fixed

Mode Menu

Page	Command	Description
77	Pivot	Rotate a link around the point it is joined to its parent
77	Twist	Rotate a link around its long Z axis
77	Move	Move the end of a link, possibly scaling the link as well
76	Add	Add a new link to the hierarchy
76	Delete	Remove a link from the hierarchy
78	Assign	Define a real object to appear in the position of a link
79	Deassign	Remove the real object associated with a link

Cell Menu

Page	Command	Description
81	First	Go to the first frame
81	Last	Go to the last frame
81	Next Key	Go to the next keyframe
81	Prev Key	Go to the previous keyframe
81	Goto	Go to a specific frame number
81	Next	Go to the next frame
81	Prev	Go to the previous frame
81	Make Key	Make the current frame a keyframe
81	Unmake Key	Make the current frame a normal interpolated frame
82	Copy From	Copy the position of the object from another keyframe
81	Remove	Delete the first or last keyframe
79	Snapshot	Make a static object of the current configuration
85	Load Pose	Use a pose saved from the Detail editor as a keyframe

Animate Menu

Page	Command	Description
82	Make	Build an animated preview
82	Free RAM	Remove the animation from memory
82	Play Once	Play the animation, pausing at the end
82	Play Loop	Play the animation, looping back to the start
82	Play Big	Play the animation using the full screen

I.2 The Project Editor

Project Menu

Page	Command	Description
122	New	Make a new project from scratch
122	Open	Edit an old project
122	Close	Close the current project

SubProject Menu

Page	Command	Description
123	New	Make a new picture size/configuration subdirectory
123	Open	Use an existing size/configuration for rendering
123	Delete	Remove a directory of renders
123	Modify	Change size/configuration for rendering

Still Menu

Page	Command	Description
128	Generate	Render selected frames
128	Show	Show selected frames
128	Delete	Delete selected frames
128	Range	Select or deselect a range of frames
128	Info	View size, creation date, and rendering time
128	Import	Mark a frame as being already rendered
128	Generate New Cells Only	Render only frames that have not been rendered before
128	Auto Dither	Use dithering in displayed pictures
128	Use FireCracker 24	Use the Firecracker to display stills

Movie Menu

Page	Command	Description
129	Load Movie	Load an animation
129	Play Once	Play a loaded animation once
129	Play Loop	Play a loaded animation repetitively
129	Drop	remove an animation from memory
128	Make	Build an animation from the selected frames

I.3 The Detail Editor

Object Menu

Page	Command	Description
21	Load	Load a Detail or Forms object or group
21	Save	Save the picked object or group
56	Convert IFF/ILBM	Change an IFF picture into an outline
20	Group	Join the picked objects into a group. The first object picked will be the parent, remaining objects will be children.
20	Ungroup	Ungroup the picked objects
22	Cut	Remove the picked object(s) from the world and temporarily store them in the Clipboard
22	Copy	Copy the current object(s) to the Clipboard
22	Paste	Put a copy of the object(s) in the Clipboard into the world
27	Attributes	Define attributes, textures, and brushmaps for the picked object(s)
50	Mold Extrude	Use an object or outline to make a "tube" along a line or path
52	Mold Replicate	Copy an object many times along a line or path
52	Mold Spin	Make an object formed by spinning an outline around a circle, with first and last points fixed
52	Mold Sweep	Make an object by spinning an outline around a circle
54	Mold Conform to Sphere	Bend the picked object around the outline of a sphere
55	Mold Conform to Cylinder	Bend the picked object around the outline of a cylinder
55	Mold Conform to Path	Bend the picked object to follow the contours of a path
56	Skin	Use a succession of outlines to make a surface
49	Make Path	Use two or more picked axes to make a spline path
49	Make Closed Path	Use two or more axes to make a closed spline path
57	Slice	Use one picked object to slice another into pieces like a cookie cutter; also used for autofacing outlines

Mode Menu

Page	Command	Description
24	Pick Groups	Pick groups of objects
24	Pick Objects	Pick individual objects
25	Pick Faces	Pick the faces of the picked object
25	Pick Edges	Pick the edges of the picked object
24	Pick Points	Pick the points of the picked object
26	Add Faces	Add new faces to the picked object
26	Add Edges	Add new edges to the picked object
26	Add Points	Add new points to the picked object
26	Add Lines	Add new points and edges to the picked object
26	Drag Points	Move points in the picked object (allows magnetism)
27	Hide Points	Temporarily hide points from the picked object
49	Edit Path	Edit the control points in a spline path
59	Magnetism	Turn on magnetism in Drag Points mode
59	Magnetism Setup	Define magnetism parameters

Functions Menu

Page	Command	Description
23	Delete	Delete picked items, or remove spline path control point
23	Join	Join two objects as one
23	Merge	Remove duplicate points, edges and faces
25	Fracture	Divide edges or faces, or add new control points to a spline path
25	Split	Split picked items off as their own object
22	Snap to Grid	Make selected points or objects move to closest grid intersection
25	Taut	Stretch a set of selected points out in a straight line
21	Add Axis	Add an object axis (with no points) to the world
21	Add Sphere	Add a perfect sphere object to the world
21	Add Ground	Add a new infinite ground to the world
21	Add Primitive	Add a new object to the world; a sphere, plane, torus, cone, or tube of arbitrary complexity
49	Add Open Path	Add a new spline path to the world
49	Add Closed Path	Add a new closed spline path to the world
56	Add Font Object	Add a flat 3D object of text in a selected font
31	Quickdraw All	Quickdraw all objects
31	Quickdraw Pick	Quickdraw all picked objects
31	Quickdraw None	Turn off Quickdraw for all objects
83	Cycle Setup	Scale and rotate object axes for use as a cycle
85	Cycle Shuffle	readjust axis locations to make legal cycle poses
85	Cycle Transforms	Rotate and scale objects around the end of the Z axis, not the center
33	Make Soft	Phong shade the picked edges
33	Make Sharp	Facet the picked edges
35	Make Subgroup	Name the picked faces as a subgroup
35	Unmake Subgroup	Remove a subgroup definition

Pick/Select Menu

Page	Command	Description
33	Pick Sharp	Pick all Phong shaded edges
35	Pick Subgroup	Pick the faces in a named subgroup
35	Unpick Subgroup	Unpick the faces in a named subgroup

I.4 The Forms Editor

Object Menu

Page	Command	Description
64	New	Make a new Forms object
64	Load	Load a previously saved Forms object
71	Save	Save the current forms object
67	Snap to Grid	Make the picked points move to the nearest grid intersection

Mode Menu

Page	Command	Description
66	Edit	Move the points of the Forms object
67	Add	Add new points to the Forms object
67	Delete	Remove points from the Forms object

Symmetry Menu

Page	Command	Description
70	Off	Turn off symmetry
70	Front View	Keep the two "former" definitions in the Front view symmetric
70	Right View	Keep the two "former" definitions in the Right view symmetric
70	Both	Keep the Front "former" definitions symmetric and the Right "former" definitions symmetric
70	90 Degree	Keep all "former" definitions symmetric with each other

Select Menu

Page	Command	Description
67	Click	Pick points by clicking on them
67	Lasso	Pick points by drawing a line around them
67	Drag Box	Pick points by dragging a box around them
67	Lock	Keep all moved points on grid intersections

CrossSection Menu

Page	Command	Description
68	Select	Choose a cross section to manipulate
68	Make Key	Make a new key cross section
68	Unmake Key	Turn a key cross section into a normal one
69	Copy From	Copy the shape of one cross section into the current key
69	Cancel	Abort a "Copy From"

I.5 The Stage Editor

Object Menu

Page	Command	Description
88	Load	Load a Forms, Cycle, or Detail object into the world
90	Add	Add a track axis, light, or path to the world
88	Rename	Give the picked object a new name
88	Delete	Delete the picked object from the world
95	Show Path Length	Compute the length of a spline path in units
88	Transformation	Numerically move, scale, or rotate and object
96	Snapshot	Make a new object from the current setup of an object
93	Position Bar	Define a key for an object's position in the current frame
93	Alignment Bar	Define a key for an object's alignment in the current frame
93	Size Bar	Define a key for an object's size in the current frame
90	Camera (Re)Track	Point the camera at a named object, or update its direction if it is tracking
94	Reset Relative	Move or align an object into the same relative orientation or position to another object as it is in a different frame.
88	Quickdraw All	Quickdraw all objects in the scene
88	Quickdraw Pick	Quickdraw all of the picked objects
88	Quickdraw None	Quickdraw none of the objects in the scene

Mode Menu

Page	Command	Description
95	Pick Groups	Pick and manipulate objects, lights, and the camera
95	Edit Path	Edit a picked spline path

Path Menu

Page	Command	Description
95	Save Path	Save a spline path
95	Split Segment	Add a new control point to a path
95	Delete Point	remove a control point from a path

Frame Menu

Page	Command	Description
91	First	Go to the first frame
91	Last	Go to the last frame
91	Next	Go to the next frame
91	Prev	Go to the previous frame
91	Goto	Go to a specific frame

Animate Menu

Page	Command	Description
95	Make	Build a preview animation
96	Free RAM	Remove a preview animation from RAM
96	Play Once	Play the preview animation, pausing at the end
96	Play Loop	Play the preview animation, looping back to the start when done
96	Play Big	Play the preview animation using the full screen

I.6 The Action Editor

Functions Menu

Page	Command	Description
99	Add	Add a new actor or timeline to an existing actor
100	Delete	Remove an actor or a timeline
100	Rename	Rename an actor
99	Info	View and change timeline information
100	Find	Find a specific actor
100	Sort	Sort the actors alphabetically
100	Cancel Add	Cancel the addition of a new timeline

I.7 Common Ground

Miscellaneous

Page	Command	Description
15	Pick Method	Define pick method: click, lasso, drag box
13	Transformation	Numerically move, scale, or rotate an object or its axis
17	Quick Render	Render the view from the Perspective view

Manipulation Gadgets

Page	Command	Description
11	Rotate	Rotate the picked object(s) or point(s)
11	Move	Move the picked object(s) or point(s)
12	Scale	Scale the picked object(s) or point(s)
13	Local	Use local coordinates for manipulation
13	Shift	Makes a manipulation affect the axis only, or allow only one movement axis
12	XYZ	Restrict manipulation to only certain world or local axes

Display Menu

Page	Command	Description
9	Coordinates	Continually display the mouse X, Y, and Z coordinates
10	Interlace	Use an interlaced display for editing
9	Grid On/Off	Turn the grid display on and off
9	Grid Size	Change the size of the background grid
14	Redraw	Redraw the screen to get rid of garbage or update it
10	Zoom In	Zoom in a factor of 2
10	Zoom Out	Zoom out a factor of 2
10	Set Zoom	Set the amount of zoom
10	Re-Center	Center your view of the world onto a new point
10	Wireframe	Use a wireframe view for the perspective window
10	Solid	Use a hidden line "solid" view for the perspective window
10	Shaded	Use a hidden line "solid" view for the perspective view, false shaded in full screen

Pick/Select

Page	Command	Description
15	Pick All	Pick all objects
16	Home	Move view to center on XYZ=(0,0,0)
16	Select Next	Select the next item in the world
16	Select Previous	Select the previous item in the world
16	Pick Select	Pick the selected item
16	Unpick Select	Unpick the selected item
15	Unpick Last	Unpick the last picked item
26	Sort	Sort items (useful for correlating points)
16	Find by Name	Select an object by its name
16	Find Requester	Select an object from a menu of names (and display object statistics)

Index

NOTES

NOTES

NOTES

NOTES

NOTES

Understanding Imagine 2.0

By Steven Worley

★★★★★

Informative, well organized, and thoroughly indexed.... Highly recommended! An alternative manual for the beginner and very useful for experienced users too. Five star rating!

Brad Schenck, **.info** Magazine

...the ultimate reference manual to the latest version of Impulse's famous 3D software, and should be in the possession of every user of Imagine 2.0... A must if you own the program.

David Duberman, **AVID** Magazine

Rendering three dimensional images on your Amiga can be rewarding both professionally and personally. Impulse Inc.'s latest version of Imagine (2.0) is the most powerful 3D object modeler and renderer for the Amiga computer. Unfortunately, most users of Imagine never use anywhere near all of the features this program offers. The sheer complexity of the software can make new users abandon the program in frustration, and even experienced users ignore many features whose purpose is a mystery.

Understanding Imagine 2.0 is a brand new work that addresses this problem. Written by prize winning Amiga artist Steven Worley, this book describes every function in Imagine and gives tips and design strategies to help you learn how to use those functions in your own work. Appendicies include a list of common problems and their solution, and even a complete guide of all of Imagine's textures and F/X. This book is not a tutorial or collection of disorganized hints, it is a complete guide to the Imagine software.

The book also comes with a bonus diskette of files, programs, brushmaps, and objects for Imagine, including several large objects made by Steve which have not been released publicly.

Apex

Apex Software Publishing
405 El Camino Real Suite 121
Menlo Park, CA 94025