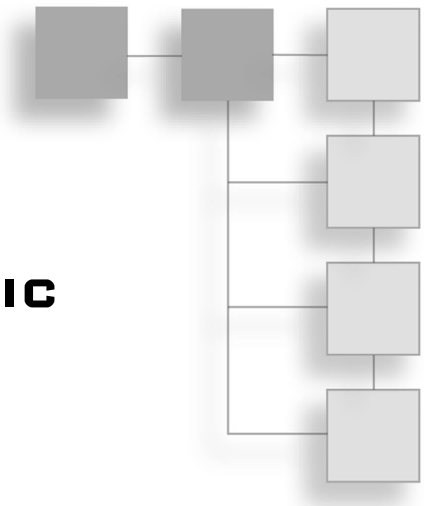
Chapter 4



Standard Geometric Primitives

The chapter titles are starting to sound more impressive, but don't worry. Like most computer terms, "geometric primitive" is just a big name for something so simple that it can be identified by almost any child.

Introduction

If you are like me, you may have, at some point in your past, pleasantly wasted hundreds of hours building with wooden blocks or plastic molded Legos (see Sidebar).

I remember building fantastic castles, pirate ships, space stations, dragons, rockets, and UFOs sitting on the floor of my den with my childhood friends and my extremely cool uncle Bruce. We thought we were just playing and having fun, but my mother somehow knew we were really mastering the fine art of modeling with geometric primitives. Of course, she would never have used those words for it.

So what exactly is a geometric primitive? Geometric primitives are the basic shapes we all know and recognize. They are simple three-dimensional shapes such as cubes, spheres, cylinders, and cones, and they work just like the blocks in a typical preschool building set.

In this section, you will learn how to model with these basic shapes, and you will discover once more how to turn primitive building blocks into the fantastic figments of your imagination. You will learn to use the gameSpace user interface to create and manipulate primitives, and then you will bring them into an actual game.

> Many years ago my good friend Dean Parks, who has helped to contribute some of the designs and models for this book, and I appeared briefly in an episode of the *P.M. Magazine* television program. We were building a space station in a Lego exhibit at the Los Angeles Children's Museum. Twenty years later, we are still building imaginary space stations.

Bring Out the Blocks

Everybody grab your preschool building set and dump all the blocks on the floor. In case you don't happen to have your preschool building set handy, take a look at Figure 4.1.

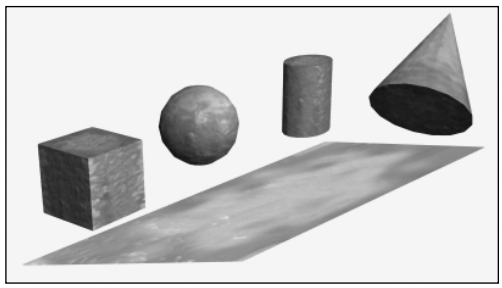


Figure 4.1
The basic primitive building blocks

Here you see a plane, a cube, a sphere, a cylinder, and a cone. In Figures 4.2 through 4.6, you can see these same five geometric primitives as they appear in typical orthographic views, which were discussed earlier in the book. Remember that orthographic views allow you to see the same object simultaneously from multiple directions.

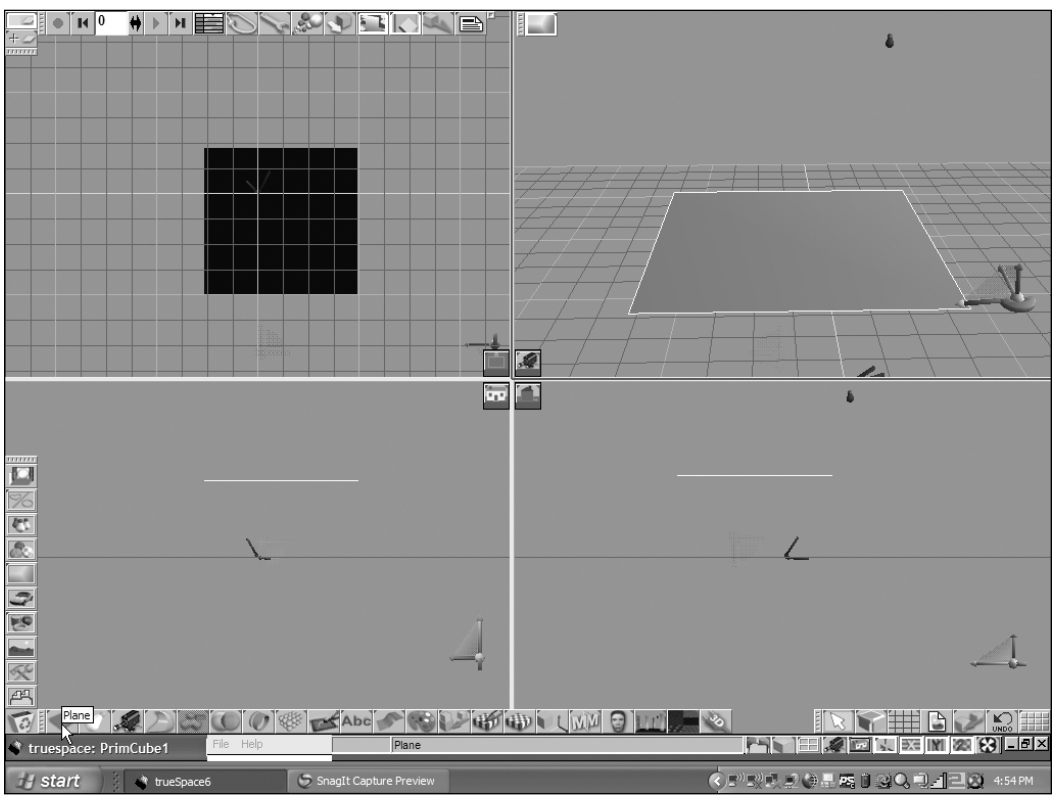


Figure 4.2
A plane is an extremely thin square. Picture a sheet of thin paper.

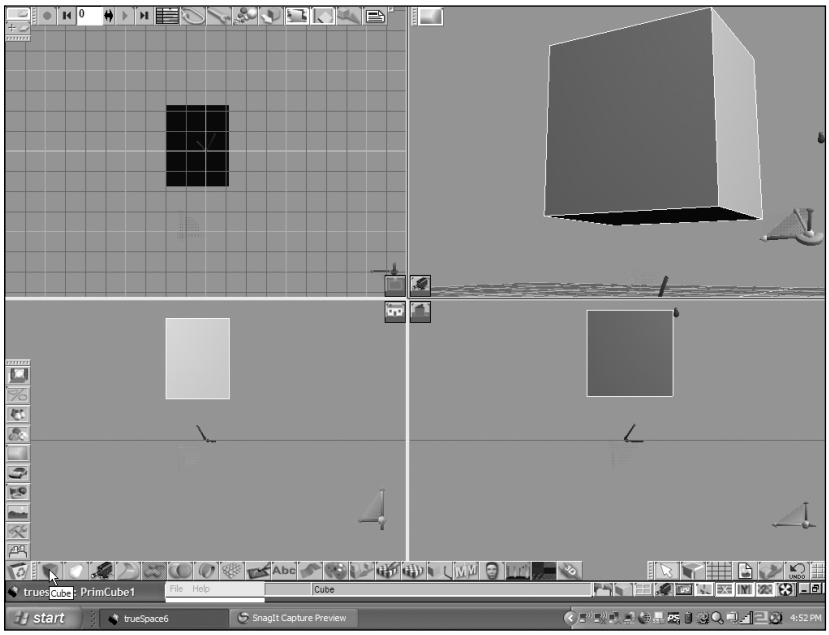


Figure 4.3
A cube is a six-sided square shape. Picture a square shipping carton.

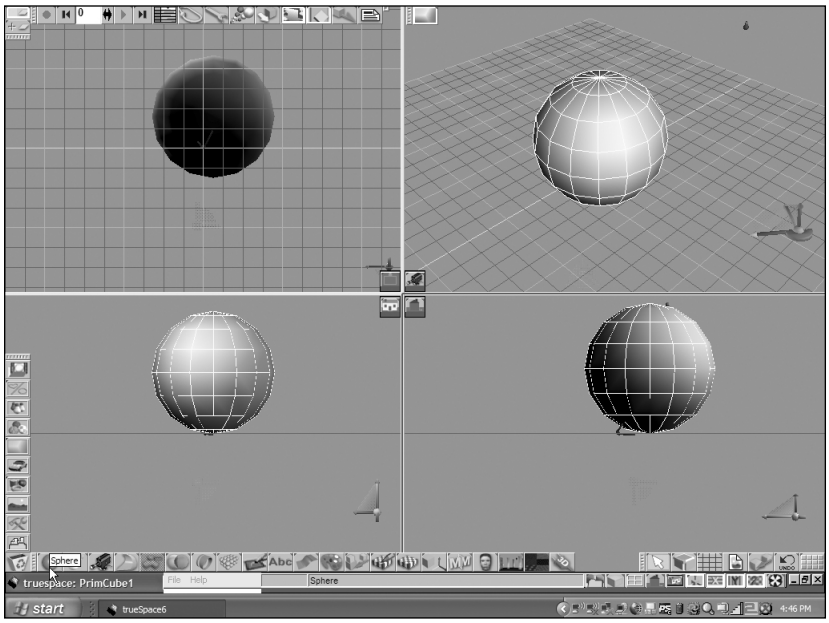


Figure 4.4
A sphere is a round ball. Picture a smooth beach ball.

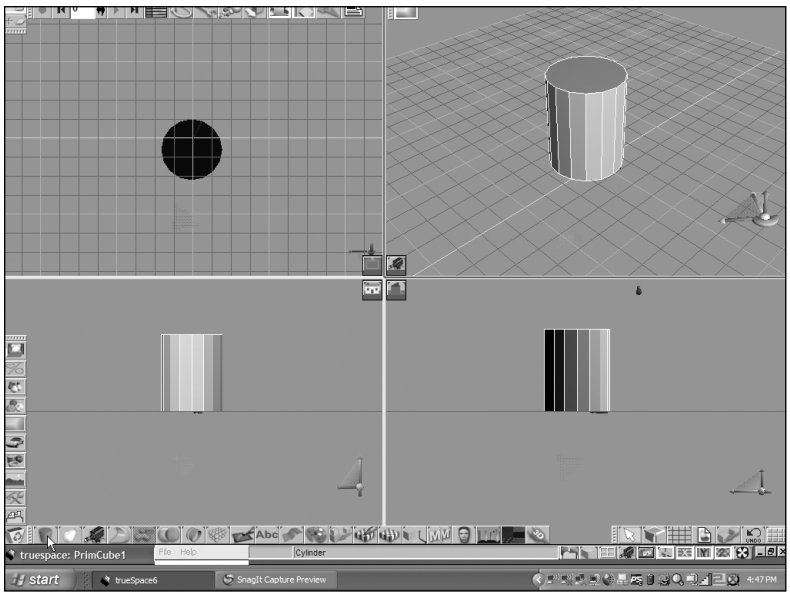


Figure 4.5
A cylinder appears round from one side and rectangular from the others. Picture a can of soda.

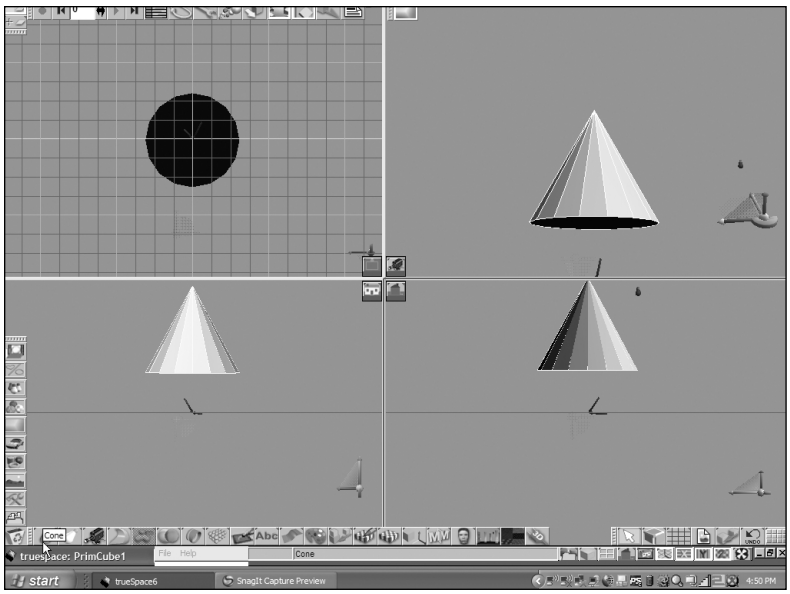


Figure 4.6
A cone appears round from one side and triangular from the others. Picture a funnel or a road safety cone.

Geometric primitives are the most basic three-dimensional shapes, such as planes, cubes, spheres, cylinders, and cones.

Primitives All Around Us

Children find it easy to look at a shoebox and see a house. They find it natural to pick up a basketball and pretend it is a planet. Yet often as children grow up, they find it harder to see the simple shapes in things that once seemed so obvious.

Geometric primitives are everywhere around us. The glass in your window is a plane. The wheels of your car are cylinders, and so is the shiny muffler pipe that sticks out the back. Perhaps the handle of your gear shift is a cylinder or a sphere, and I would bet the shaft of the gear shift is some kind of cylinder.

Look around the room that you are in right now. How many basic primitive shapes do you see around you? If you are one of those serious people who expect their cubes to be perfectly square or their spheres to be perfectly round, then you may not count too many. Artists, of course, tend to like their lines curved, their cubes squashed, and their spheres compressed. If you can accept that primitives can be stretched in one direction or another, then you are certain to find them everywhere you look.

I remember a delightful conversation I once had with Kathy Kirk, the chief show designer at Walt Disney Imagineering. I suggested that computers and computer-aided design software could help expedite the accurate drawing of straight lines and geometric shapes. She swept her hand grandly about and pointed to the fanciful storyboard drawings all around her office. "This is Disney," she said. "We don't use straight lines or geometric shapes here." She was right; Disney artists certainly use a lot of beautiful curves. But most Disney animators happily admit that they still depend on geometry. As Disney legend John Hench explained, they just like to squash and stretch their geometry as if it's somehow alive and moving.

Two Primitives Are Better Than One

Although there are many game objects that can be built from simple primitives, think of the objects that you could build if you were to stick your primitives together. Wasn't this the idea behind the popular Legos building blocks? See Figure 4.7.

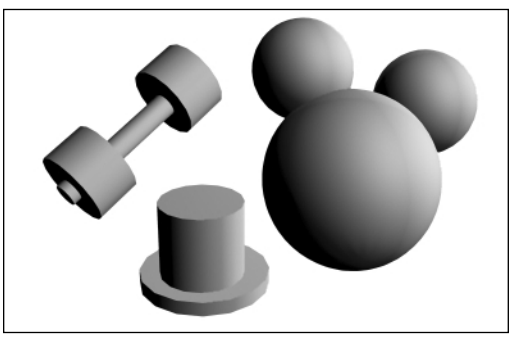


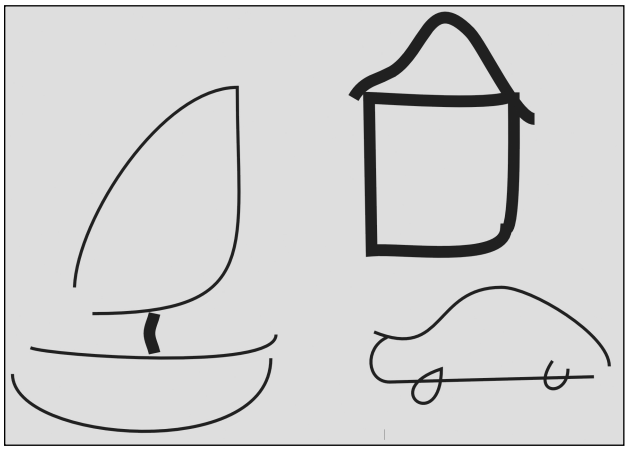
Figure 4.7
Combining geometric primitives

Two cylinders make a top hat. Three cylinders can make a weight lifter's barbell, and three simple spheres might appear to millions of children around the world like a very famous cartoon mouse.

With a little imagination, you can probably think of hundreds of everyday objects that you could easily make from a few combined primitives. With a modeling tool that can create, combine, and distort geometric primitives, you can easily produce an endless array of game objects and models.

The Power of Primitives

The real power of primitives is not in the software. It's the portable supercomputer you call your brain that really fills in the gaps between geometry and illusion. The human brain has a unique ability to extract real everyday objects from even the simplest lines and shapes. Take a look at the rough collection of lines in Figure 4.8. What do you see?



Have you ever gazed at a cloud in the sky and saw a dolphin, a shark, a fish, a whale, or a really large frosted jelly donut?

Even thousands of years ago, our ancestors saw such things in the clouds and in the tiny little dots that fill our nighttime skies. All right, I'll admit they probably never saw a large frosted jelly donut, but you get the idea.

Figure 4.8
What do you see?

A fundamental skill of good game artists is the ability to communicate and bring to their viewers' minds the most complicated of objects, using only the simplest forms, shapes, and patterns. Primitives are typically the simplest forms around.

It should be pretty easy to see why simple geometric primitives have become some of the most popular tools of modeling for 3-D games. Nearly every 3-D modeling package on the market supports some form of 3-D primitive modeling.

What makes modeling with primitives so popular?

- Geometric primitives are easy enough for any child to understand.
- Most artists can easily visualize geometric primitives.
- Programmers can implement and manipulate geometric primitives easily.
- Computer graphics hardware can draw geometric primitives extremely quickly.
- Geometric primitives can be combined and manipulated in a huge number of ways.

Game artists use geometric primitives to create a vast number of the common and not-so-common objects you may need in your games.

An important side benefit of modeling with primitives is that 3-D models created from primitives tend to be built efficiently. A game programmer would say "They have a very low polygon count." Although there are certainly exceptions to this rule, breaking your objects into large simple blocks tends to result in game models with less wasted detail and fewer complex parts.

gameSpace offers a powerful selection of tools to create, modify, combine, and manipulate geometric primitives.

Building with Blocks

In this section, you will learn to create and manipulate geometric primitives within gameSpace. You will build some simple game objects, and you will use these objects to enhance and improve a real computer game.

3, 2, 1, Launch

It's time to launch gameSpace. If you haven't installed gameSpace on your computer, go back to Chapter 3 and install it now. Also be sure to install the Gamescapers' Adventure Explorer software which was discussed in Chapter 3. See Figure 4.9.

Figure 4.9
Click this icon to launch gameSpace from your desktop.

Once you have installed gameSpace, you can launch the program in a couple of ways:

- Click on the gameSpace icon, which the installer program usually places on your desktop. See Figure 4.9.
- Click on the Windows Start button, then choose All Programs, and finally click on Caligari gameSpace or Caligari gameSpace Light.

Everything in Its Place

When you launch gameSpace, it usually remembers the last scene that you edited, so you may see some leftover objects from a previous tutorial or from your own projects and experimentation. You can save these items, if you wish, by choosing File > Save > Scene from the File Menu at the bottom of the gameSpace screen. See Figure 4.10.

tip

By default, gameSpace's menu item File/Preferences has an option/setting called "Auto Load." If this option is checked on the Preferences panel, gameSpace will reload the last scene you were working on. If you prefer to have a fresh, blank scene every time you start gameSpace, simply uncheck the option.

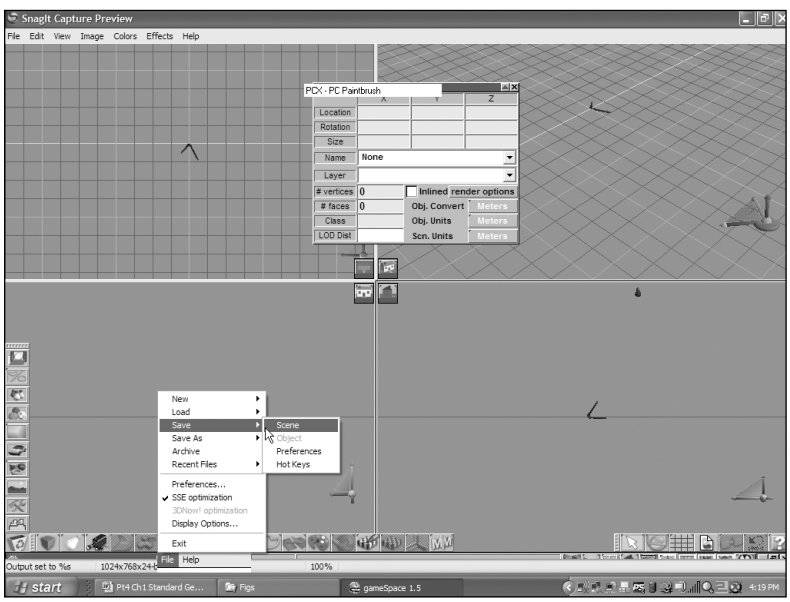


Figure 4.10
Saving a gameSpace scene

Now choose File > New > Scene from the File Menu to clear out the workspace, and you are ready to begin a new scene. See Figure 4.11.

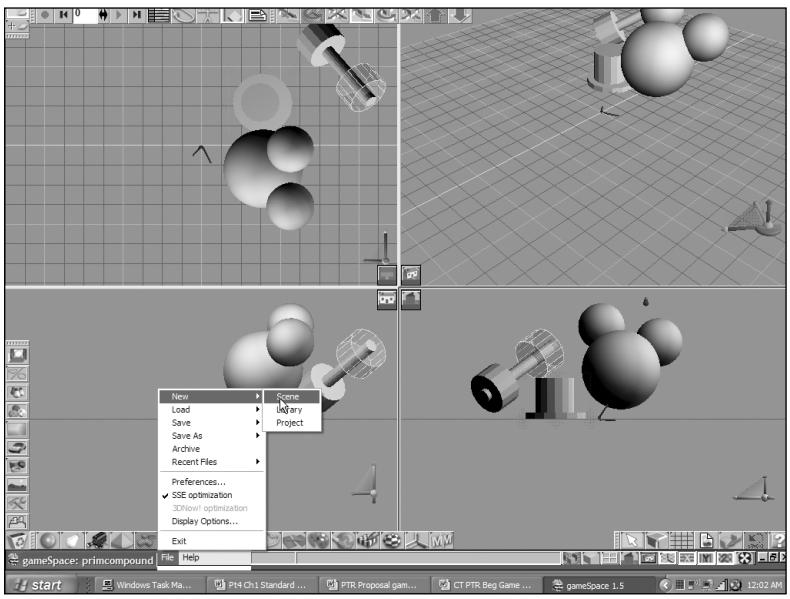


Figure 4.11
A brand new scene: imagine the possibilities!

Surprisingly, gameSpace has perhaps the smallest and simplest menu bar in the business. The menus, which are found at the bottom of the screen, are used only for file handling, user preferences, and easy access to the useful help files. If you are using the free version of gameSpace, you will unfortunately be missing the bulk of these help files.

If this is your first time working with gameSpace, you might, at first glance, feel a bit overwhelmed. Don't worry. It's actually much simpler than it appears.

Many Windows programs hide their commands in sprawling menus that you pull down from the top of the screen. Most typical graphics programs complement busy menus with a maze of elaborate panels, floating windows, and adjustable sidebars.

The creators of gameSpace have taken floating graphic toolbars to a whole new level. They obviously don't care much for menus. They created a visual interface where any possible command or feature is only one or two mouse clicks away.

They accomplish this by organizing hundreds of picture buttons into neat stacks. Imagine that the screen is a giant board game, and that each little picture box on the bottom of the board is actually a pile of game cards. To play a card, you choose the card you wish to play from the appropriate stack and place it on the top of that stack. That is exactly how the picture buttons, also known as icons, in gameSpace are arranged.

Icons are small buttons containing pictures that visually depict the actions they are meant to perform.

You might think that all of these stacks of little buttons would get in the way, but the buttons are organized in a simple and logical way. Once you understand the purpose of each set of icons on the gameSpace toolbars, you can easily find any tool you may need.

As you start your first model, you will quickly see how this works.

Toolbars are clusters of icons or groups of icons arranged in a row. gameSpace has a default set of toolbars that appear on the bottom, top, and sides of the screen, but you can easily modify these toolbars or even create new toolbars of your own.

Your First Model

For your first primitive model, you will create a simple box. A box may not sound so exciting, but in the magical world of video games (and the not-so-magical world of moving and shipping), they can be very useful. I can assure you that the models you make will get more interesting as we go.

This may be your first opportunity to experience the unique gameSpace interface. Remember the stacks of picture buttons (or icons) discussed above. The first slot on the toolbar, right next to the recycle bin at the bottom left of the screen, is what gameSpace calls the primitive library. I would describe it as a pile of icons for creating primitives. See Figure 4.12.

Few nongamers recognize the incredible power and versatility of boxes. Boxes can block your escape. They can shield you from gunfire. They can contain deadly, precious, or life-changing cargo. You can slide them across the floor. You can climb them, break them, shoot them, blast them to pieces, and if you are really careful, you can randomly jump from one box to another. A few very fortunate boxes have even made it to stardom as the lead characters in successful, if somewhat primitive, video games. Someone has to build them. So why not you?

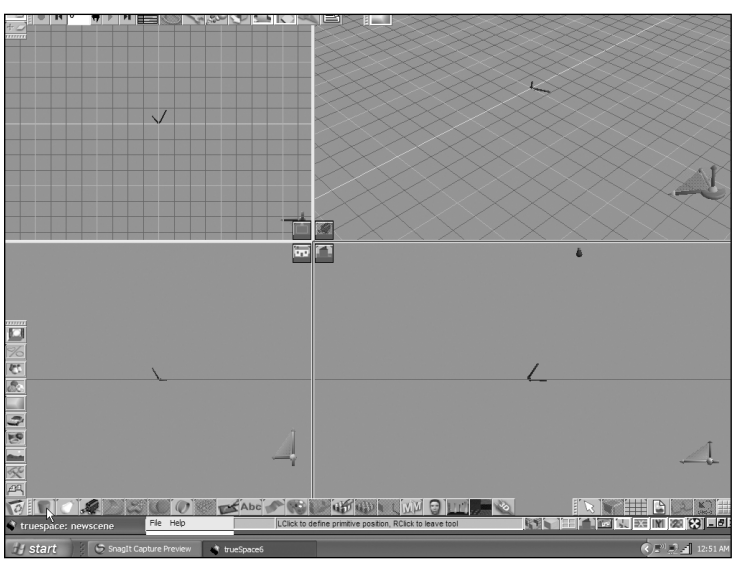


Figure 4.12
The primitive library or the pile of primitives

You might see an icon with a picture of a cube here. If you do, the proverbial "cube card" is at the top of the stack, and you can simply click on it. See Figure 4.13.

Figure 4.13
The cube icon

If, as in Figure 4.12, you see a picture of a cylinder, cone, plane, sphere, or anything other than a cube, then you must pull the "cube card" from the proverbial pile and bring it to the top of the deck.

To do this, simply click on and drag whatever icon is currently visible in the first slot beside the recycle bin. If your screen looked like Figure 4.12, you would click on and drag the picture of the cylinder. This reveals the full set of icons that can appear in this slot. See Figure 4.14.

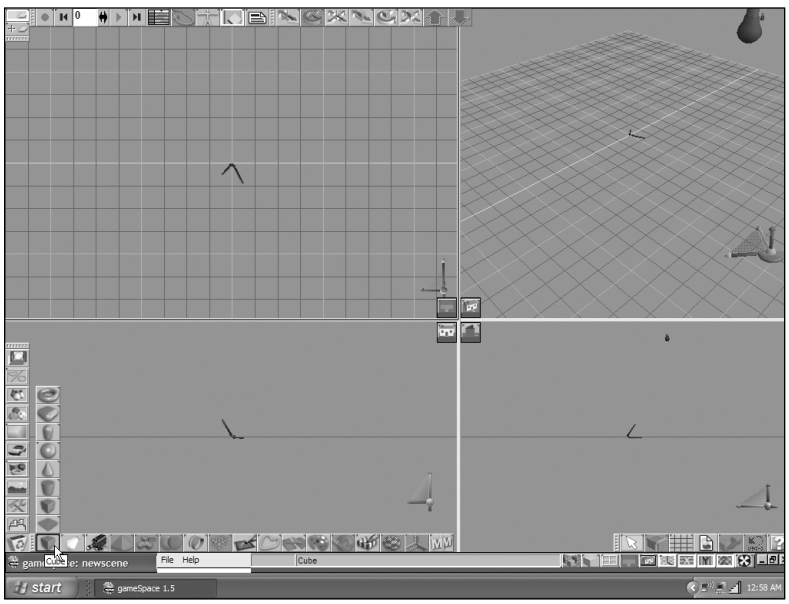


Figure 4.14
Click and drag to see every available card in the "pile of primitives."

Drag toward the top of the screen until the moving border appears around the cube icon shown in Figure 4.13, and release the mouse button.

The screen should now appear as in Figure 4.15, with the cube icon sitting next to the recycle bin. Notice the amber glow that appears around the cube icon. This tells you that gameSpace is now in Primitive Creation Mode. You may also notice that the mouse pointer on the screen will now appear with a little picture of a cube beside it.

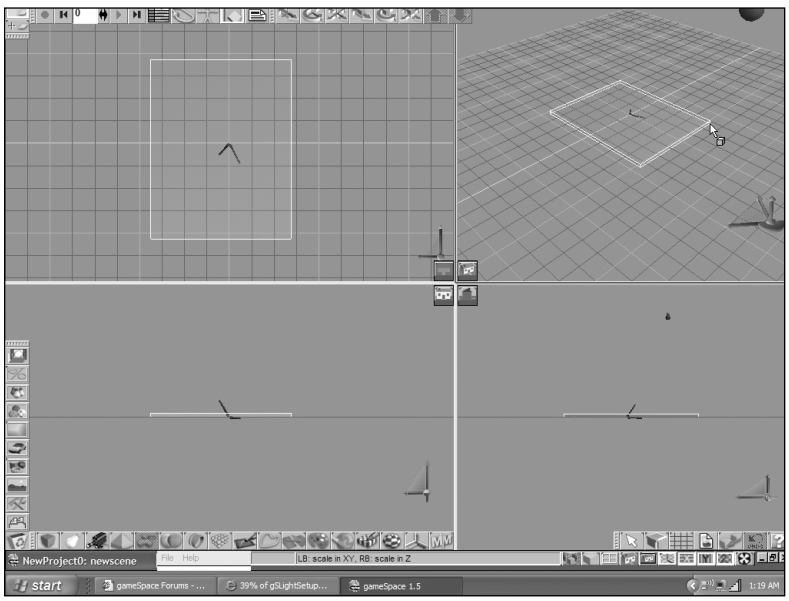


Figure 4.15
Ready to draw a cube.

Four Views Are Better Than One

Your gameSpace screen may still look quite different from the one in these screenshots, however. The screenshots indicate four separate viewports or windows on the screen, but you may only have one viewport on yours.

gameSpace can work in a variety of ways. There are times when it is easier to work in one single viewport, such as when you need to examine extremely small and intricate details. You should remember from Chapter 2, however, that four views allow you to see an object's details in all three dimensions.

Viewports are the windows through which you view the three-dimensional virtual world within gameSpace. gameSpace lets you change the number of visible viewports and the specific viewing angle from which each viewport is drawn.

To change the layout of your gameSpace screen, work the toolbars exactly as you did when selecting the Cube tool. This time, however, you will choose from the viewport configuration library, which appears on the lower toolbar at the far right-hand side of the screen. See Figure 4.16.

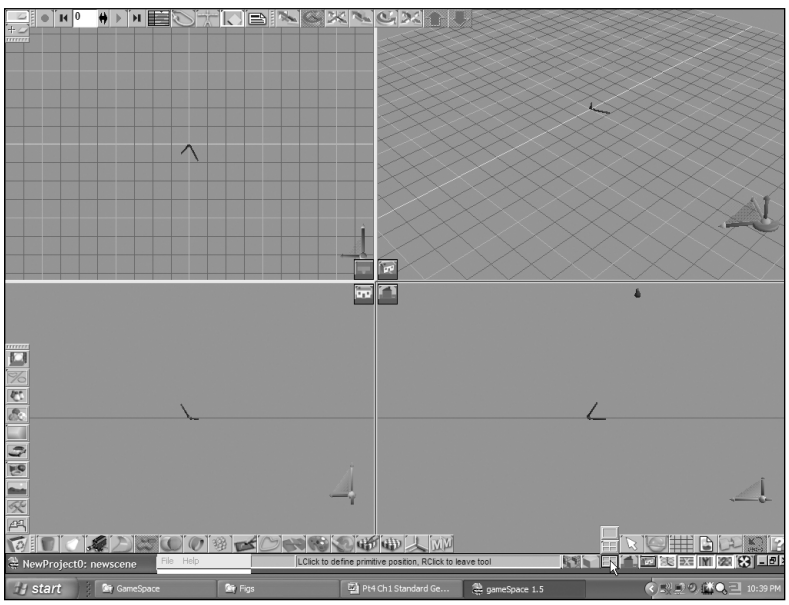


Figure 4.16
Choose a screen layout.

The screen layout icons, shown in more detail in Figure 4.17, allow you to select either one viewport, which fills the whole screen, or four smaller viewports, as seen in Figure 4.15.

Figure 4.17
Use these icons to choose between single viewport and multiple viewport modes.

Choose the single viewport mode by clicking on and dragging the currently visible screen layout tool until the icon showing only one viewport is selected. Now your screen should look exactly like Figure 4.18.

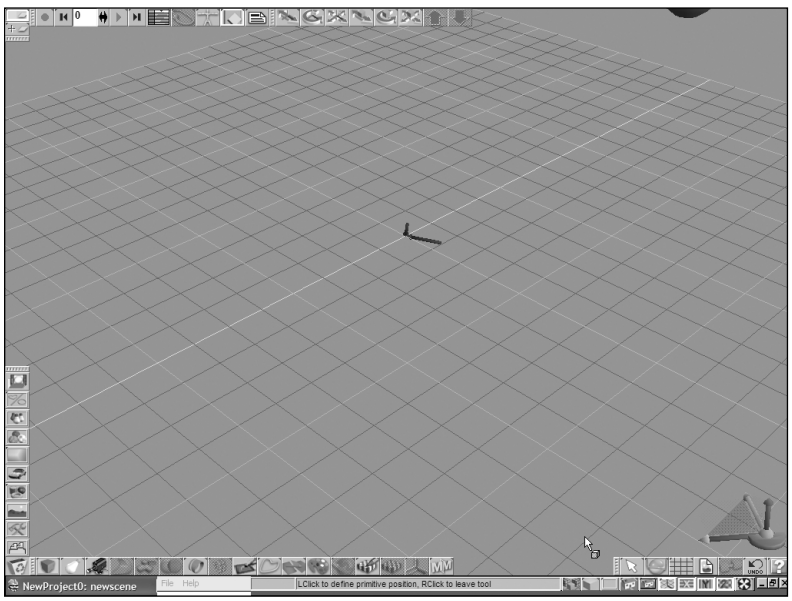


Figure 4.18
Single viewport mode

You are almost ready to make your first model.

Navigating in Three Dimensions

Before you can create a cube, you must understand how to move around and draw in three-dimensional space. It is one thing to view objects in three dimensions, but it is quite another to actually use a mouse to move them in three dimensions.

It is easy enough to use the computer's mouse to draw on a flat, two-dimensional surface. You simply move the mouse up, down, left, and right, and the mouse pointer moves exactly where you want it. When you are working in top, front, and side orthographic viewports the mouse can function in the usual way. These are two-dimensional windows into a three-dimensional scene. But what about a perspective viewport? At some point, you still need to draw in three dimensions.

tip

It is often convenient to use the two-dimensional orthogonal views to manipulate and position objects precisely. In these views, your mouse is restricted to two-dimensional movement up, down, left, and right within the active viewport.

It is obviously a little harder when you have to move the mouse in six possible directions, but the ability to do so can significantly enhance your modeling capabilities. Ideally, you should be able to draw three-dimensional forms that flow up and down, left and right, and forward and back all at the same time.

This problem has been solved in many different ways by almost every 3-D program on the market, but here is where gameSpace really shines. The geniuses at Caligari did a fantastic job of inventing one of the most simple, intuitive, and elegant 3-D navigation interfaces in the industry.

> There have also been some very creative and innovative hardware solutions to this problem. From swinging armatures and floating position trackers to control gloves and three-dimensional mouse-like devices, hardware manufacturers are racing to find a good three-dimensional control device that will achieve mass market acceptance.
>
> I have recently been playing with a new product from 3D connexion called the SpaceBall 5000. It is like a trackball with 12 buttons, but it can sense very fine hand movements in more than six directions, including up and down, left and right, forward and back, and several directions of rotation. Unfortunately the manufacturer does not yet provide direct support for the gameSpace interface, but this may change in the very near future.

The basic concept is quite simple. First, drag the mouse to move left and right or forward and back, then right-click-drag the mouse to move up and down. Mathematicians would say left-click-drag to move on the X and Y axes, then right-click to move on the Z axis, but if you are not a mathematician, an example may be in order.

A Cube Is Born

It is much easier to show this visually than to adequately describe it in words, so check out Figures 4.19 to 4.23 and then try it for yourself.

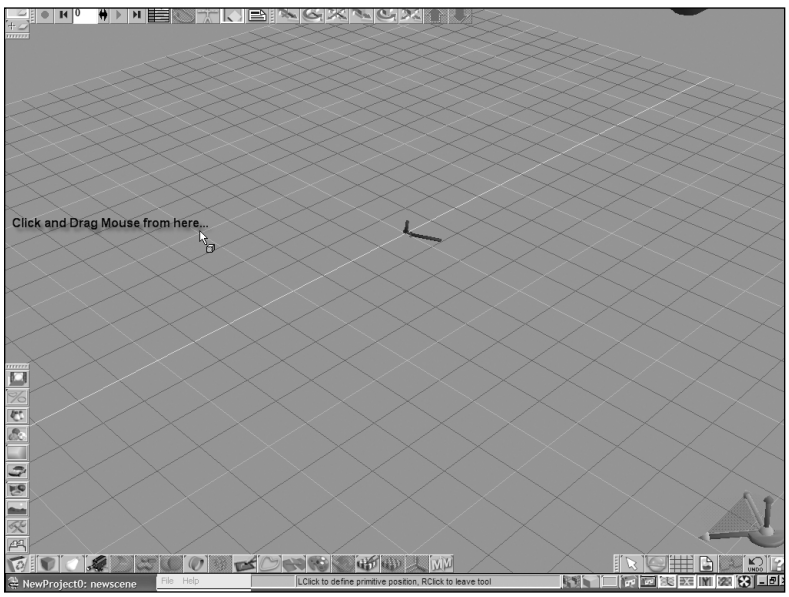


Figure 4.19
Making a cube, step 1

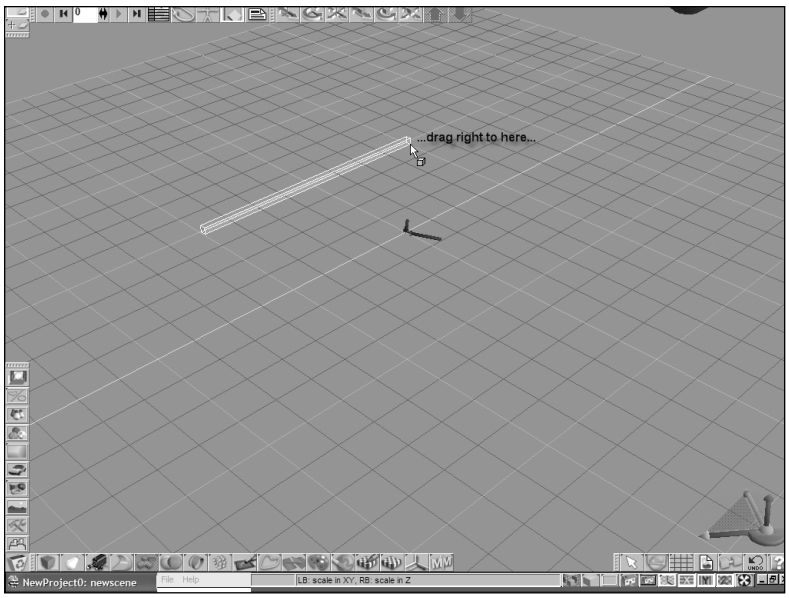


Figure 4.20
Making a cube, step 2

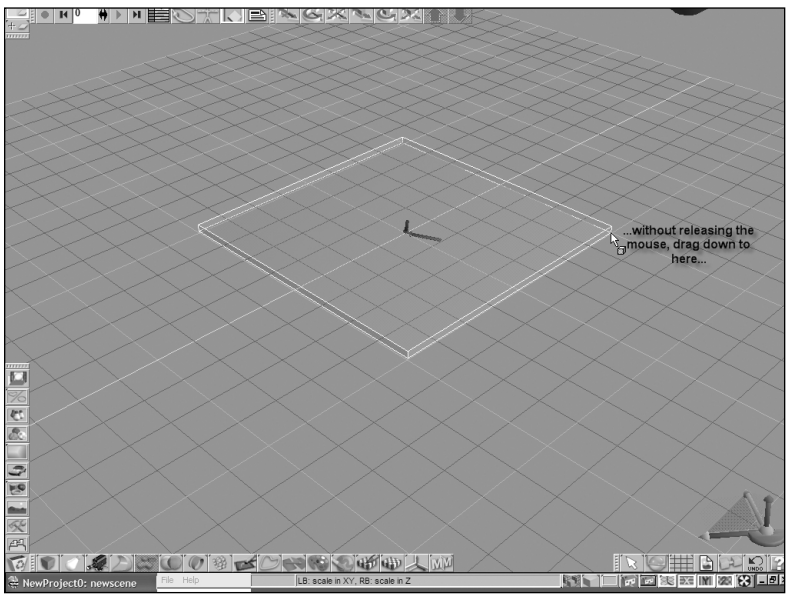


Figure 4.21
Making a cube, step 3

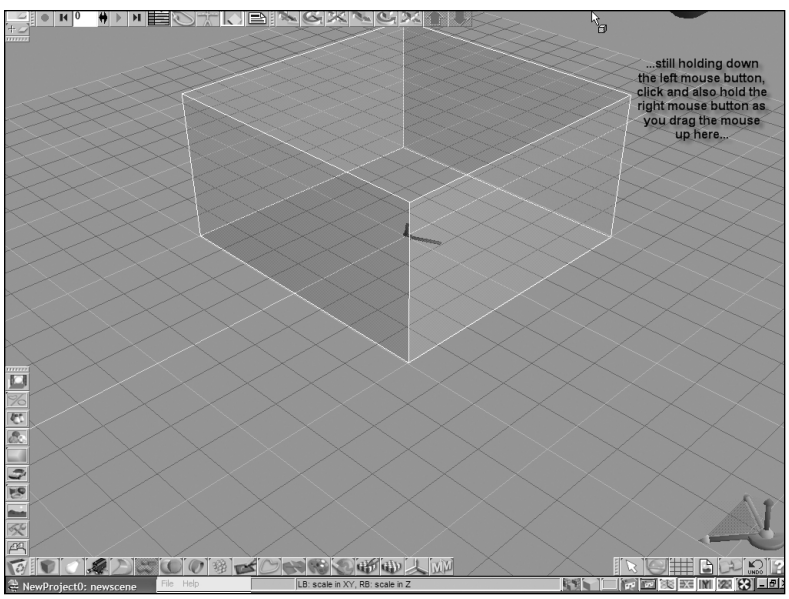


Figure 4.22
Making a cube, step 4

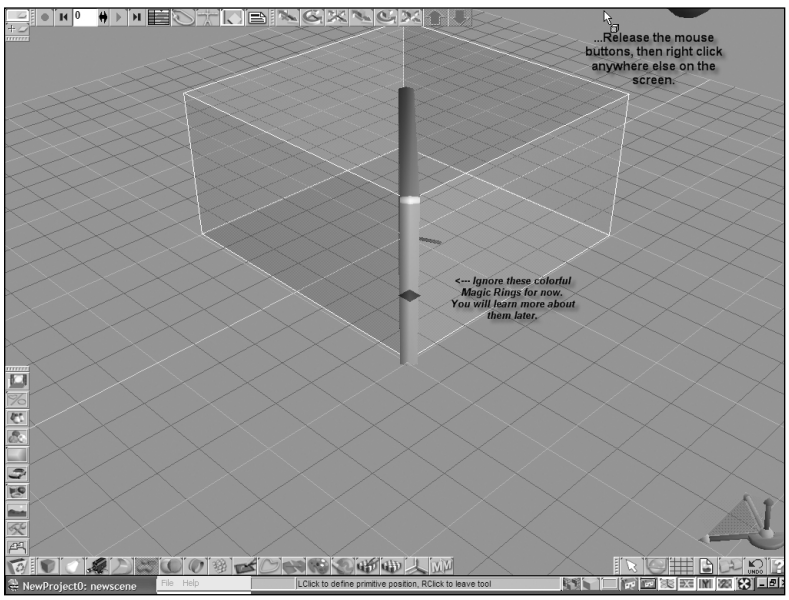


Figure 4.23
Making a cube, step 5

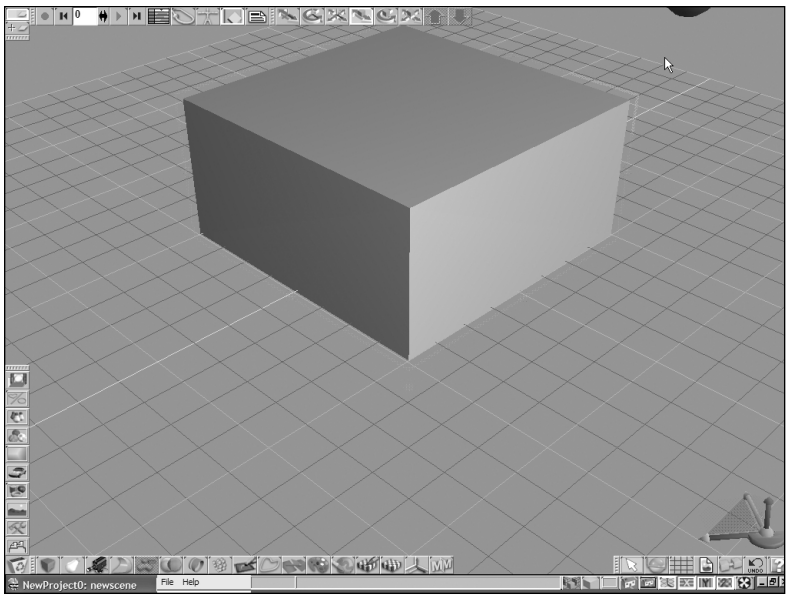


Figure 4.24
The completed cube

Hit the Delete key to erase the cube, and repeat the above process until you feel comfortable working in three dimensions. Try building a variety of cubes, from thin tall cubes to wide short cubes.

Adjusting Your Cube

Creating your cube is just the beginning. Sometimes it is just not possible to get exactly the shape you need. Most of your time in modeling is spent tweaking and finely adjusting the objects once they are created, and gameSpace makes this pretty easy.

Your cube is a pretty simple shape, but you may still want to adjust it a little. Select the object scale tool from the toolbar at the top of the gameSpace screen. You can also activate this tool by pressing the C key. See Figure 4.25.

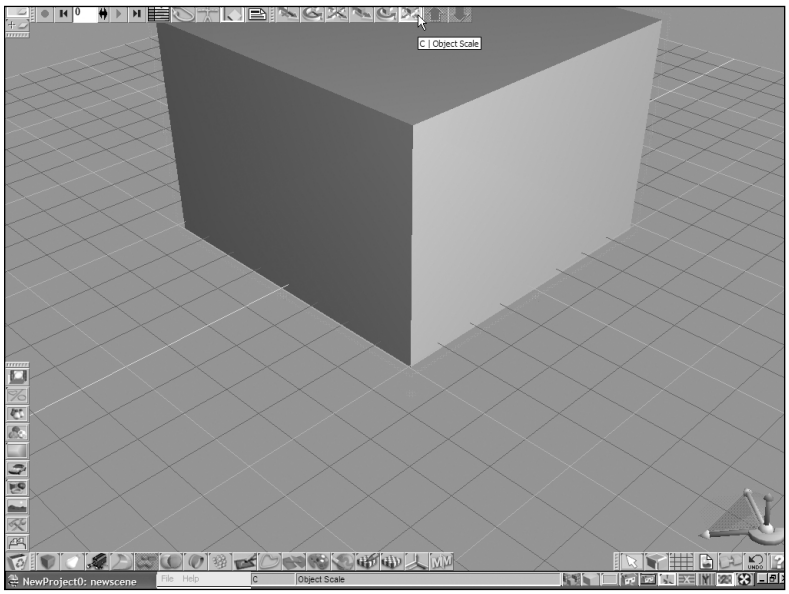


Figure 4.25
Select the object scale tool.

Now you can drag your mouse to scale, shrink, or stretch the cube in any direction. If you want to make the cube taller or shorter, remember to hold down the right mouse button as you drag. The right mouse button will scale the cube in the up and down direction.

With only one viewport, it may be difficult to determine when the cube is precisely square. So choose the multiple viewport icon from the screen layout tools at the bottom of the screen. See Figure 4.26.

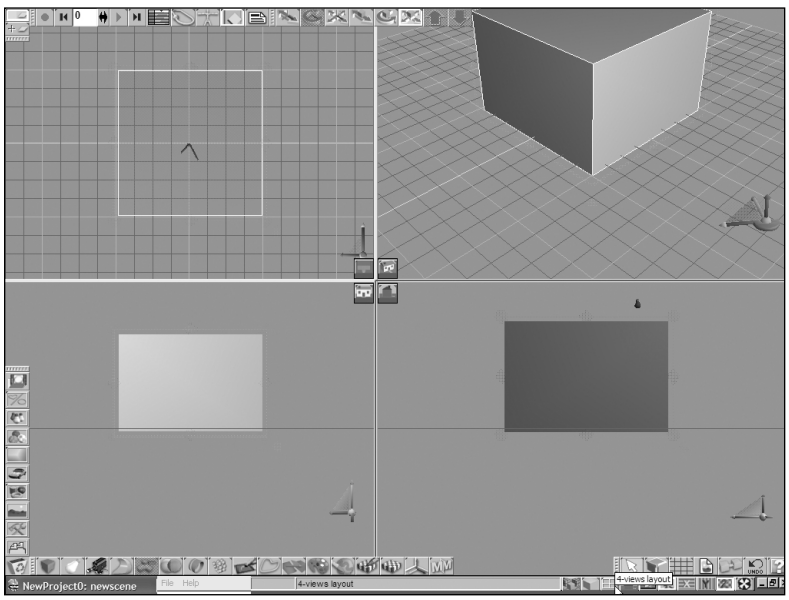


Figure 4.26
Change to a multiple viewport configuration.

Now you can use the three orthogonal viewports (top, front, and side) to see exactly how long or short the cube appears from each direction.

The buttons in the slots just to the left of the object scale tool allow you to move or rotate the selected object. Try using these tools to reposition the object and turn it around. Note how the use of left-click-drag and right-click-drag affects each of these tools.

If you need to zoom in or out for a better look in any viewport, click in that viewport, and then use your mouse scroll wheel or center button to move the viewport's viewing position toward or away from your object.

tip

Use the Z, X, and C keys to quickly activate the object move, object rotate, and object scale tools, respectively.

You might think these are odd letter choices for these functions, but notice that they are the first three keys on the bottom row of letters in any standard QWERTY keyboard. This makes it quite easy for you to quickly move between them, as long as you can remember their order. Object move is the first key, object rotate is the second, and object scale is the third, just as they appear on the toolbar.

Taking More Control

Modeling things visually using gameSpace's simple graphical interface can be easy, but often you need a little more control. Maybe it is enough that your cube model is almost a perfect cube, but sometimes the game requires a cube that is perfectly square. What you need is a little more precision.

Perhaps the level designer has planned for the cube to slip precisely into a specific hole in the level, or perhaps the programmers have programmed the game physics and collision code to some exact specification. It could just be that your producer has had too little sleep lately and thinks numerical precision can make up for an inadequate budget and a overzealous schedule. It really can't, you know.

Whatever the reason, the visual interface is just not enough. It is time for you to take control. Find the icon of the white arrow on the higher of the two toolbars on the bottom right of the screen. This is the object selection icon. See Figure 4.27.

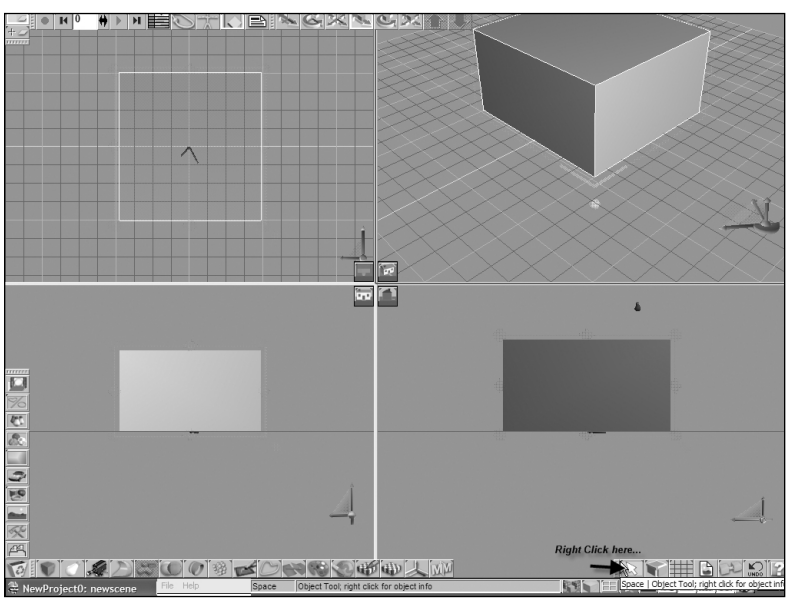


Figure 4.27
Right-click on the object selection tool.

Right-click the arrow icon to open the Object Info Pane. See Figure 4.28.

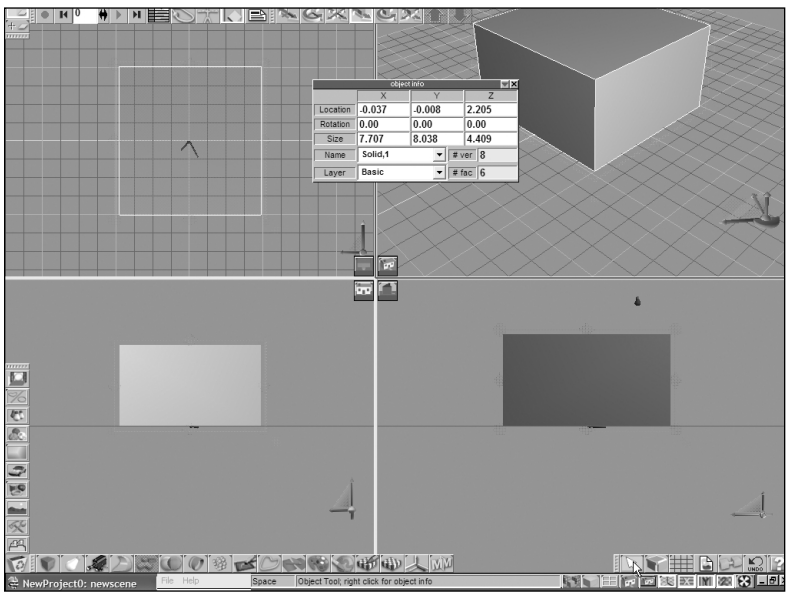


Figure 4.28
The Object Info Pane

The Object Info Pane tells you the precise location, rotation, and size of the selected object. In this case, the only object in the scene is the cube. The location, rotation, and size are each specified with three numbers, which describe the object in units along the X, Y, and Z axes.

In gameSpace, the X axis runs left and right. The Y axis runs in and out of the screen, and the Z axis is up and down. The grid that you see in the perspective viewport typically shows lines running in the X and Y directions.

Looking at the Object Info Pane in Figure 4.28, you can see that the cube in the screenshot is located near the center of the world, offset from the center by exactly 2.205 units in the up and down (Z) direction. If you wanted to precisely center this cube, you could replace all three location numbers with the number 0.0.

To do this, click on the first location number, beneath the X column title, and type in the new number. Use 0.0 if you wish to center the object. Repeat this process for the location numbers beneath the Y and Z column title. See Figure 4.29.

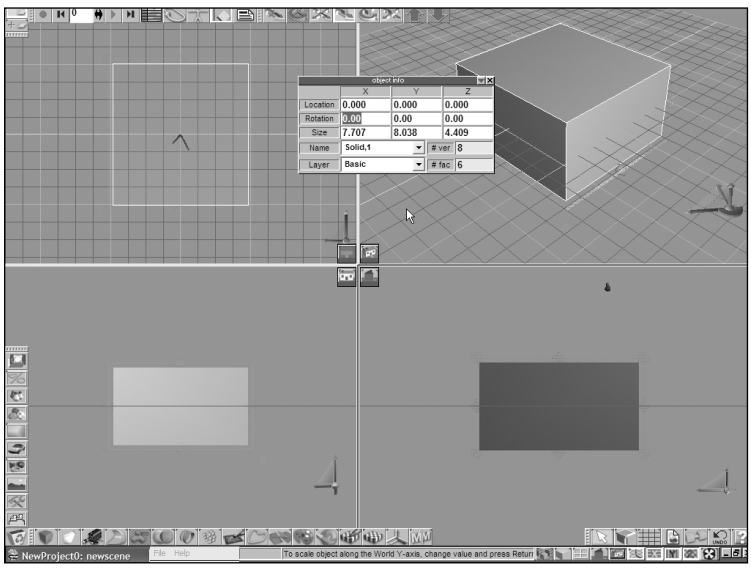


Figure 4.29
Here the cube is perfectly centered, but not perfectly square.

You can also see that the cube is far from perfectly square. Its size in the X and Y directions is nearly twice its size in the Z direction. To make it perfectly square in all directions, make sure the three size numbers are set to the same value. In Figure 4.30, they have been each set to exactly 7.0.

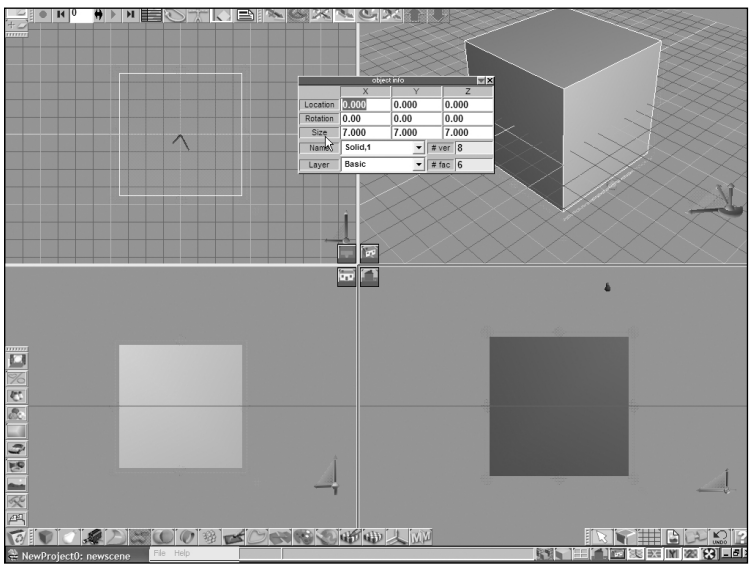


Figure 4.30
The cube is now perfectly squared.

If you are not happy with the location of the Object Info Pane, you can drag it anywhere around the screen by clicking and dragging on the blue title bar, where it says the words "object info." This can be very useful because it allows you to see objects that might otherwise be hidden by the Object Info Pane. gameSpace remembers where you last placed the Object Info Pane and tries to reposition it there the next time you open it.

Calibrate Your Rulers

You have seen how to change the numbers in the Object Info Pane, but do you really know what they mean? When I say the cube is 7 units wide, does that mean it is 7 inches or 7 miles? This is particularly important when you are making objects to import into a video game. Imagine the confusion if one game artist creates a cool player character for a game, while another makes the nifty supercharged atomic minivan that the player will ride in. If the player is modeled at 7 meters and the vehicle is modeled at 7 feet, the players are going to have a little problem.

To prevent this situation from arising, and ironically, to enable it as well, gameSpace lets you choose what the coordinates should mean for any given object. To see this information, you must open up more of the Object Info Pane. Notice the little red triangle (technically, it's an arrow tip) at the top right corner of the Object Info Pane. This downward-pointing triangle tells you that there is more of this Object Info Pane that is not currently visible. See Figure 4.31.

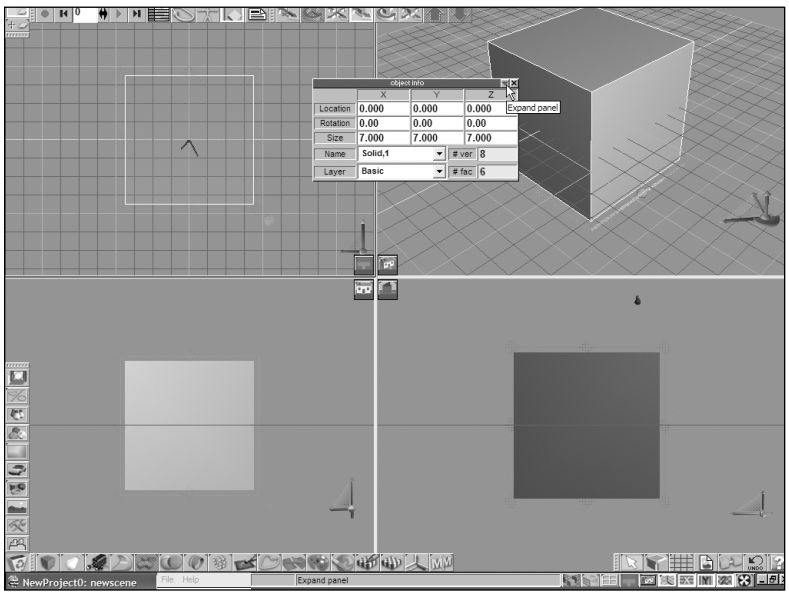


Figure 4.31
The downward-pointing triangle tells you there is more to view.

Click on the red triangle to expand the Object Info Pane. You will see a lot more details and options.

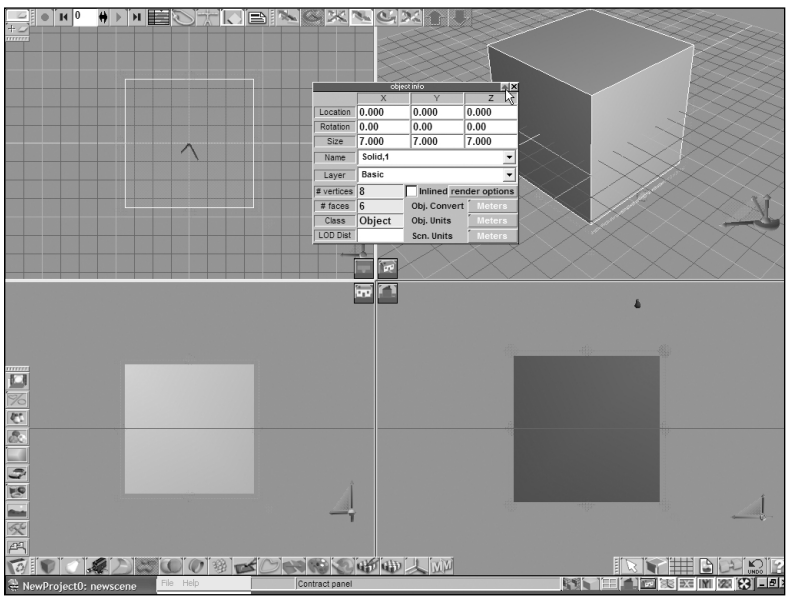


Figure 4.32
This is the full Object Info Pane.

Notice that the red triangle now points upward. Clicking on the arrow once again would collapse the Object Info Pane to once more hide the extra information.

The expanded Object Info Pane answers the questions about the meaning of units. Figure 4.32 tells us that the units are being measured in meters. Meters are good for engineers and surveyors, but most video-game programmers think in points and pixels. I recommend that you set all of your unit measurements to points. I will be using points throughout the examples in this book. Figure 4.33 shows the same object with its units set to points.

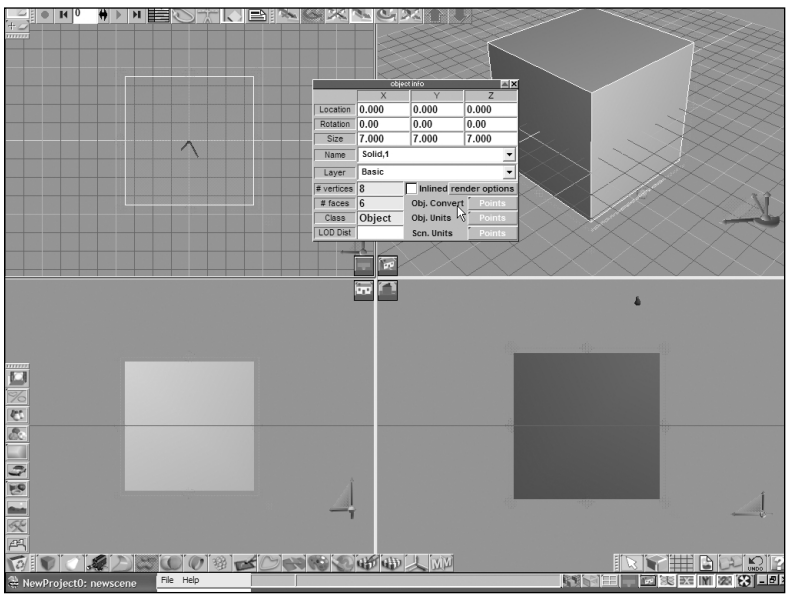


Figure 4.33
Now the units and screen units are both set to points.

The Price of Detail

Another important detail disclosed in the Object Info Pane is the number of vertices and faces used in the selected object. Remember that all objects in a 3-D game are made up of vertices (points), edges (run between vertices), and faces (polygons). In the case of this cube, there are eight vertices. These vertices are the corners of the cube. There are also exactly six faces, which are the cube's six sides.

Game artists must think about face counts far more critically than film artists, animators, architects, or illustrators. The more detail you add to your game objects, the more faces and vertices they require. As a general rule, the more faces you use in your game objects, the slower your game executes. This can and will impact game play significantly. In extreme cases, it can even force a video game to misbehave and crash in a variety of ways.

In less extreme cases, high "poly count" almost always increases the hardware requirements for the consumers who want to play your game. This translates into fewer players, fewer purchasers, fewer fans, less job security, and much less money in your yearly bonus.

As a game artist, your duty is to carefully balance the look of your objects against the efficient and economical use of faces. Game objects are typically low-poly objects. They must have a very low face count.

If you are using gameSpace Light, which comes with this book, then don't worry. gameSpace Light has a strict limit to the number of faces you can use in your objects, which can be a bit constraining, but may make you a better game modeler. It certainly forces you to model more economically.

Saving the Scene

When you have a good square cube, save it by choosing File > Save As > Scene, from the File menu. See Figure 4.34.

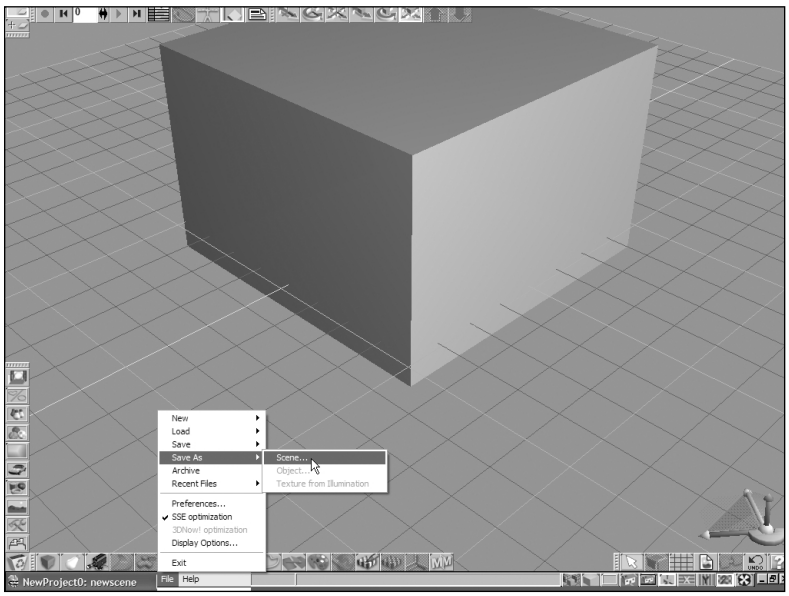


Figure 4.34
Saving your first model

gameSpace opens a window asking you to choose a name and a location to save your scene. It also gives you a choice of file types in which to save it. For now, choose the default Caligari .scn file type, and let gameSpace save the file in the suggested directory. See Figure 4.35. Once we start working with other file formats, you will want to save each object's files in its own separate directory.

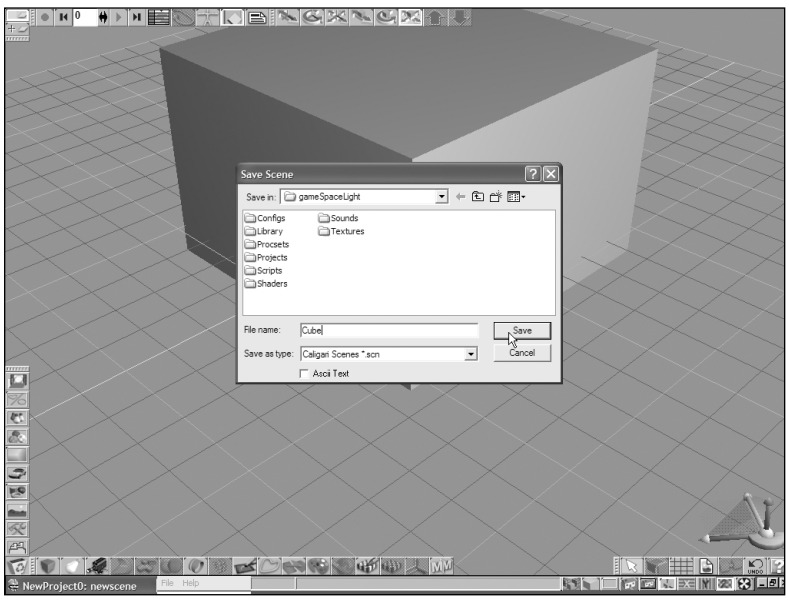


Figure 4.35
Saving Cube .scn

Kicking Around a Ball

The very first electronic video game, Pong, contained the very first video graphic ball. It is only fitting that your first video game should contain a ball or two as well. Pong's ball was only a dot that bounced between two little lines (paddles), but your first ball will be a real regulation soccer ball, or at least a close virtual simulation. You can probably guess what's coming next.

1. Choose File > New > Scene from the File menu.
2. Drag on the cube icon until you see the whole primitive library.
3. From the primitive library, select the sphere icon, which shows a picture of a smooth round ball, and release the mouse button.

Notice that the sphere icon has a close twin, the geosphereicon. This sort of subtoolbar works just like the regular toolbars you have seen before, only in this case, it is hidden deep inside another toolbar. See Figure 4.36.

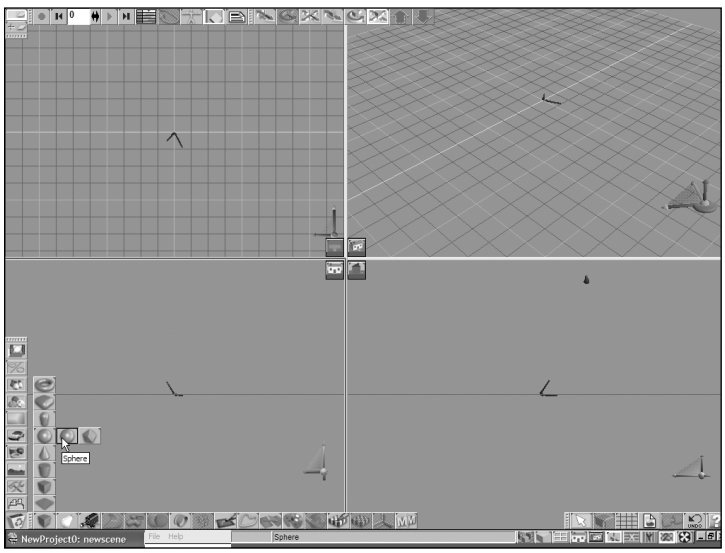


Figure 4.36
Choose the smooth sphere primitive icon.

Click in the center of either the top or the perspective viewport on the screen and drag the mouse to create a round ball. See Figure 4.37. It doesn't really matter how large you make the ball, as long as you can see it on the screen. You will use the Object Info Pane to scale and position it appropriately.

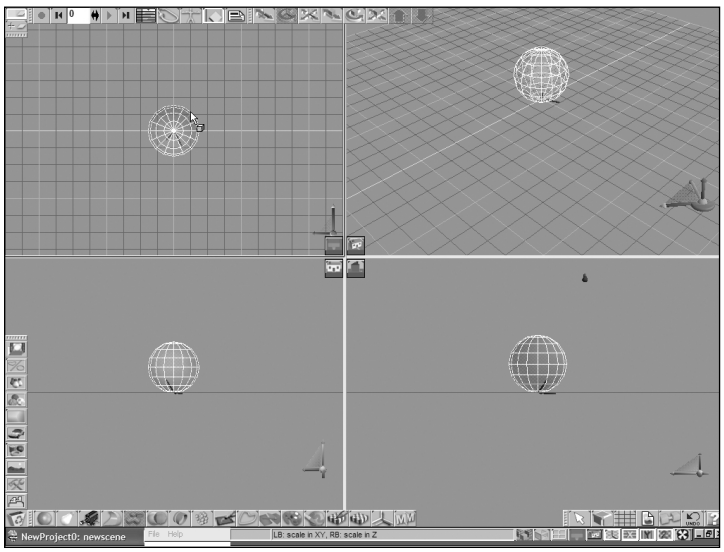


Figure 4.37
Click and drag to create a sphere.

Right-click anywhere else on the screen to get rid of the colorful Magic Rings.

You are going to use this object in a game, so it is important that you position and scale it precisely. That means that it is time to open the Object Info Pane. If necessary, open the Object Info Pane by right-clicking on the arrow icon, as shown earlier in Figure 4.27.

Set the object parameters exactly as shown in Table 4.1 and in Figure 4.38.

Table 4.1 Ball Object Parameters

	X	Y	Z
Location	0.146	0.2	0.926
Rotation	0	0	0
Scale	1.764	1.764	1.764

All units and screen units are set in Points.

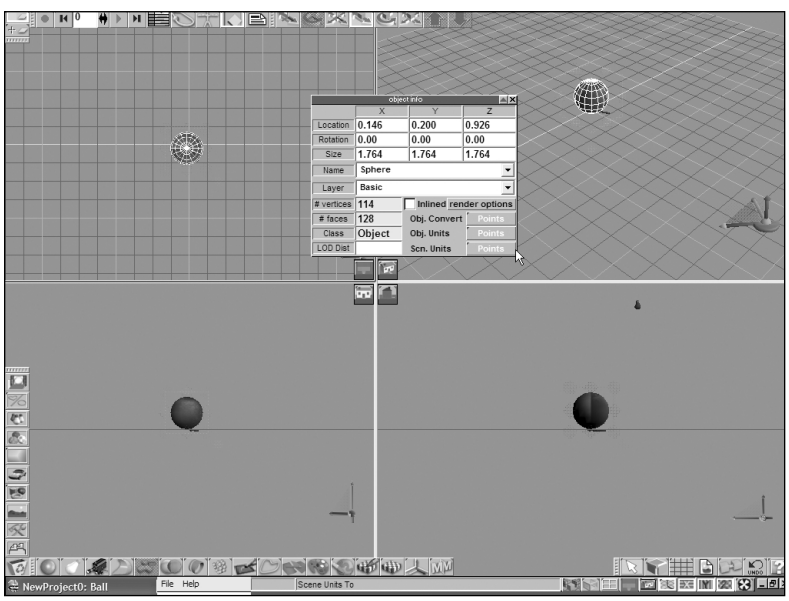


Figure 4.38
Set the ball location, rotation, scale, and units as seen here.

You have a ball, but it is not clearly a soccer ball. It could be a pretty round rock. To make it a soccer ball, you need to see the little black and white facets that are known and loved throughout the world, at least, that is, throughout Europe, Australia, and South America.

Painting on the Detail

You could create a true faceted sphere, which in this case might even reduce the face count of your object. But often, adding such details increases the face count and certainly increases the complexity and difficulty of building the model. You can accomplish the goal easily without spending hours meticulously modeling a soccer ball and then hand painting its various sides.

Textures are the minute details, colors, and visual artifacts that help the brain to sense and recognize the surface structure and internal content of any object.

Great textures are the secret ingredient of great game models, and gameSpace has a powerful and versatile Material Editor that makes texturing fun. gameSpace also comes with a Material Library full of useful textures and a sophisticated Library Manager to help you find, modify, and build your own libraries of models, textures, shaders, and bitmaps.

Materials are gameSpace elements that represent combinations of procedural (computer-generated) images and user-created images that combine and layer together to create the illusion of real-world textures.

The Material Library is a storehouse for prepackaged or user-customized gameSpace materials.

Shaders are programs and numeric procedures that instruct gameSpace how to generate various textures and individual components of textures based upon specific user-controlled parameters. Think of a shader as an automated texture factory, where you can control the texture's appearance by setting and adjusting its dials and controls.

The Material Editor is a pop-up panel where you can layer, edit, and modify any number of shaders and bitmaps to build highly complex and realistic textures for your objects.

Bitmaps are computer files that contain numeric representations of visual images. These can be digital photographs, retouched photos, scanned drawings, animated videos, or visual content from any other source. In other words, a bitmap is a picture.

Open the Material Library by clicking on its icon on the vertical toolbar to the left of the gameSpace screen. Its picture looks like a bowl full of painted balls. See Figure 4.39.

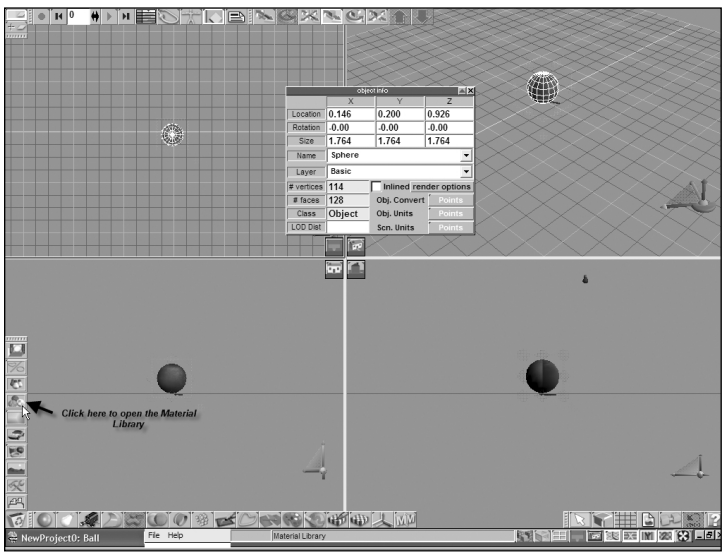


Figure 4.39
Click here to open the Material Library.

In the Material Library, you will see a wide variety of materials. These are only a small sample of the many thousands of rich textures that gameSpace can generate. Click on any material in the Material Library to open it for inspection in the Material Editor. See Figure 4.40.

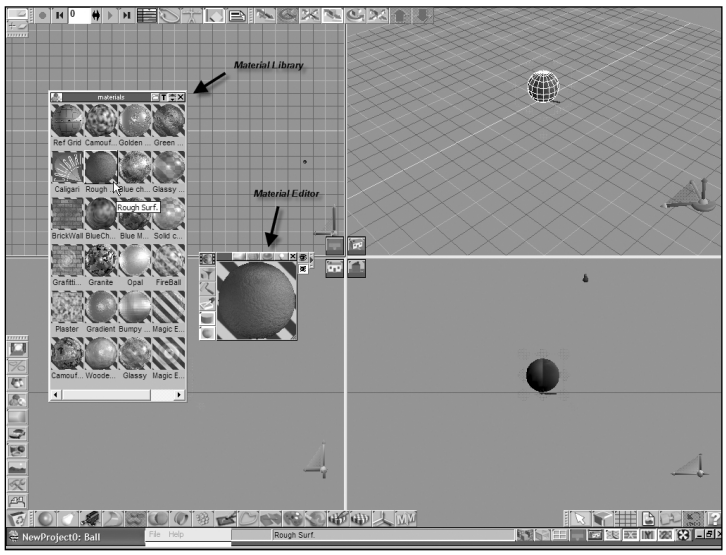


Figure 4.40
The Material Library and the Material Editor

To apply a material to your object, simply drag the material from the preview window in the Material Editor to the object in gameSpace. This is demonstrated in Figure 4.41.

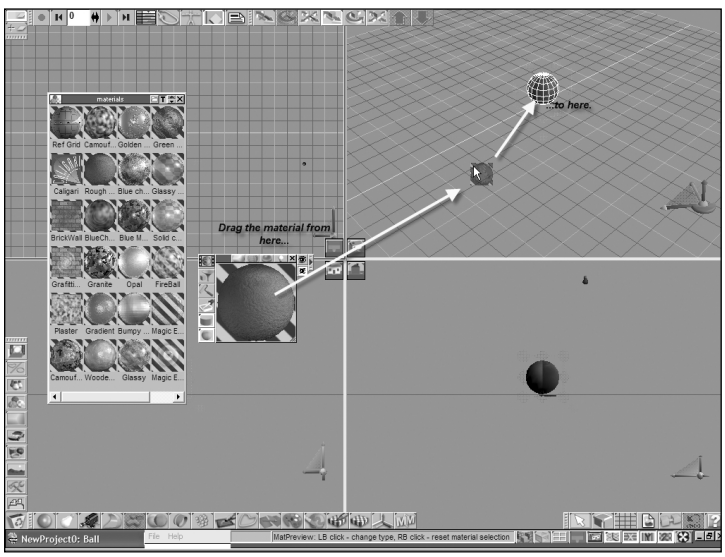


Figure 4.41
Dragging a material onto an object

The Material Editor offers lots of great materials, but none of them are right for your soccer ball. So you need to create your own.

To start with, you should choose a material that is somewhat like the material you need. None seem too close, so choose a simple one like the Ref Grid at the top of the Material Library. See Figure 4.42.

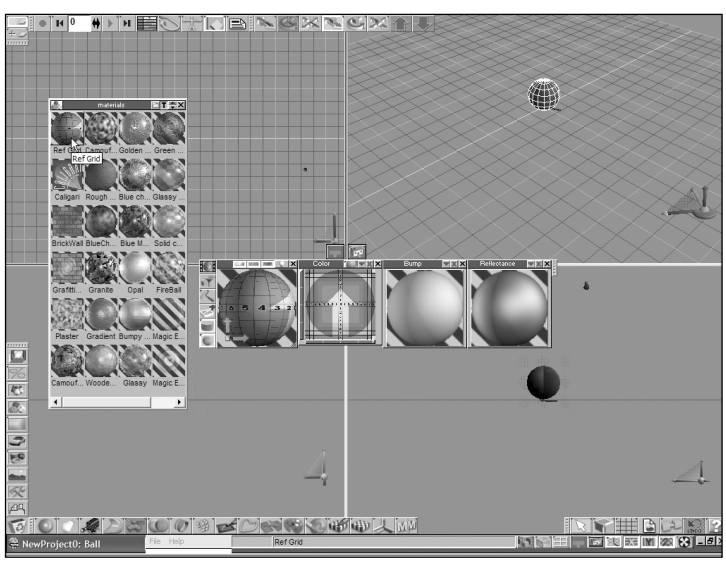


Figure 4.42
Click on Ref Grid to bring it into the Material Editor.

The Ref Grid uses a very simple shader that copies a bitmap image, a picture of a grid, directly onto the object. To see the details of this shader, click on the arrow pointing toward the right on the far right side of the Material Editor. This exposes the various shaders that make up the selected material.

You want to replace this shader with another more appropriate for a soccer ball, so open the Color Shader Library, which is the rainbowlike gradient just beneath the Material Library icon.

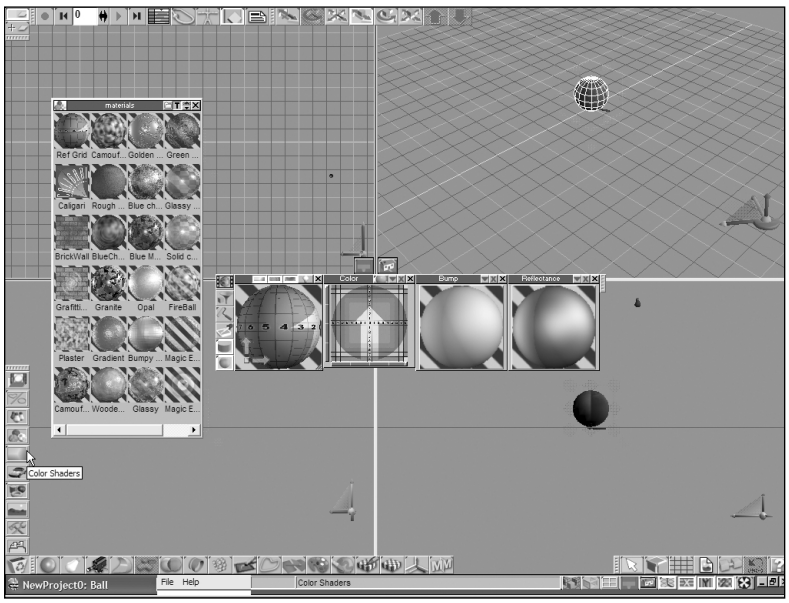


Figure 4.43
Opening the Color Shader Library

When you click the Color Shader icon, as shown in Figure 4.43, the Color Shader Library opens, and the Material Library vanishes. The Material Editor stays open, however.

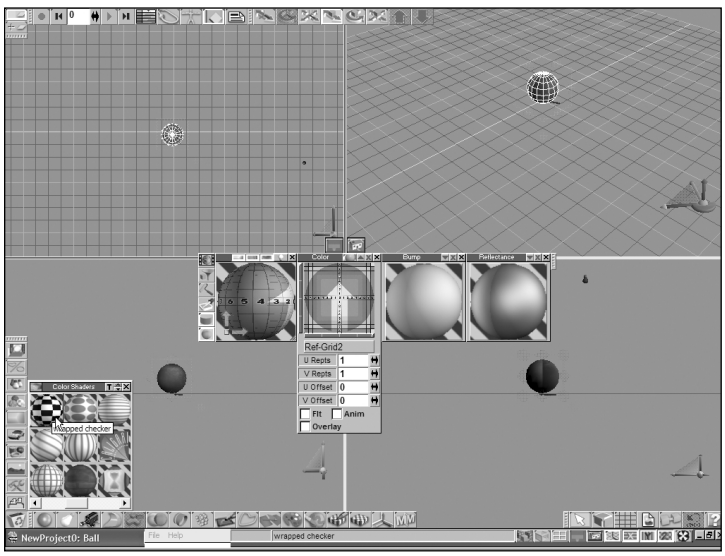


Figure 4.44
Choose the Wrapped Checker shader.

Drag the scroll bar at the bottom of the Color Shader Library until you see the Wrapped Checker shader. See Figure 4.44. Don't confuse this with the Cube shader, which looks quite similar. Click on the Wrapped Checker shader and see how it changes the material in the Material Editor. See Figure 4.45.

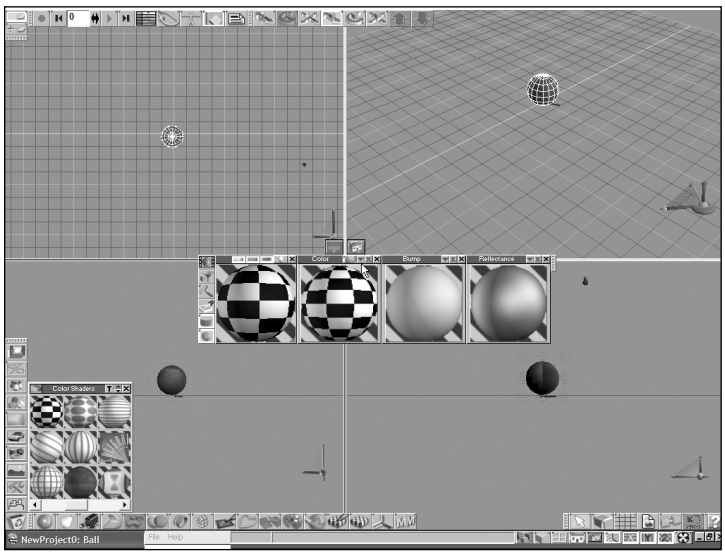


Figure 4.45
The Wrapped Checker shader at work

The Wrapped Checker shader is a procedural program that automatically generates its texture from a set of inputs that you can control. To see these inputs, click on the red triangle in the top right corner of the Color panel in the Material Editor. It works exactly like the red triangle in the Object Info Pane. See Figure 4.46.

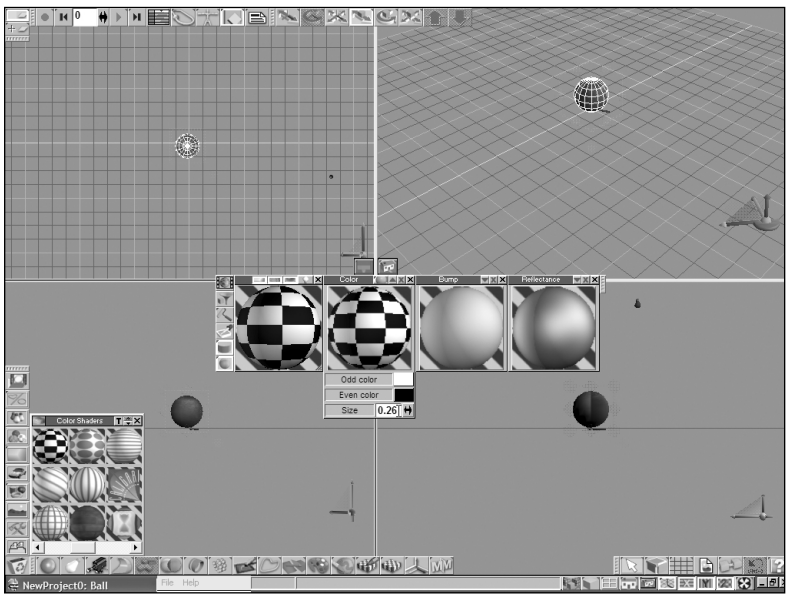


Figure 4.46
Adjust the parameters of the Wrapped Checker shader.

You can see that the Wrapped Checker shader allows you to choose the color for each set of checkers, as well as the size and density of the checker pattern. Try changing the number in the Size parameter in various steps between 0.0 and 1.0. You can also drag on the little spinner (looks like a bidirectional arrow) to raise or lower this value. Watch the Material Editor's main preview window to see the impact of the changes.

I found that a size setting of around 0.19 makes a pretty good soccer ball, but feel free to use any setting that you please here.

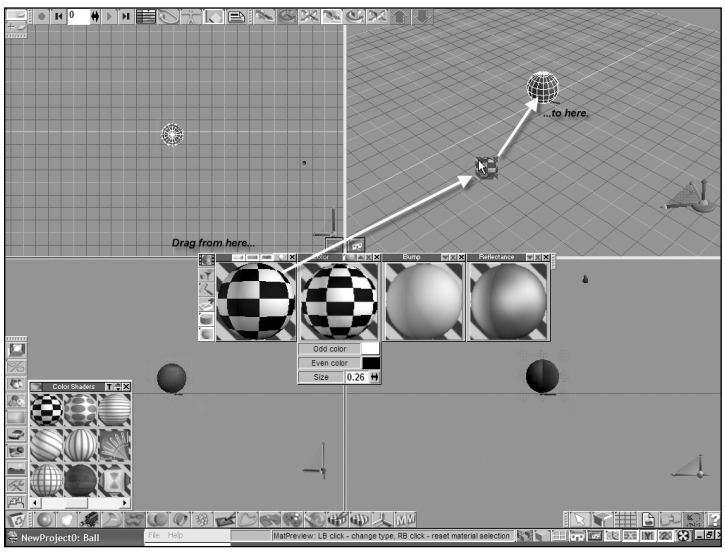


Figure 4.47
Drag the material onto the ball.

Finish the soccer ball by dragging the new material onto the ball. See Figure 4.47.

Saving a Game Object

It's time to save the ball. First, you should save the scene exactly as you did before. This is the file you will reopen if you ever want to change the ball in the future. I have a pretty strong feeling you will want to change it, so definitely save the file. If you have forgotten how to save the scene, see Figure 4.48.

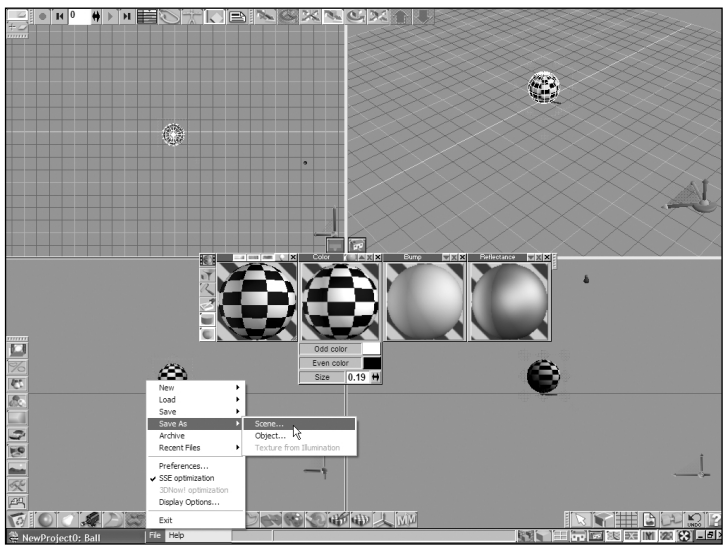


Figure 4.48
Save the scene for the ball.

Unlike your last model, you are actually going to bring this one into a game. That means you need to save it in a format

that a game can understand. The Caligari .scn file format is a great way to save your work, but it is not recognized by any game engine I have used. Depending upon your specific target game, you may need to save your object files in a variety of formats.

For the sample games I have included with this book, the format to use is a DirectX .x file. I made this choice in programming the demo game, but it won't always be the same for any other game you produce.

Another significant design decision I made in programming the sample games was that every object's files should be housed in that object's own private directory. This also is not always the case for every other game, but it has definite advantages when working with a lot of .x files.

> DirectX .x files store the model and the model's textures in separate files. Sometimes they are stored in a great many separate files. By placing each object's files in its own directory, it is easy to keep all of these files together, and it is much harder for them to become lost or separated.

For the remainder of this book, you should always save your object files as DirectX .x files and place them in their own directories. It's easy, it's a good way to keep things organized, but most importantly to remember for the projects in this book, it's not optional. Here's how it's done.

1. Select File > Save As > Object from the File menu. See Figure 4.49.

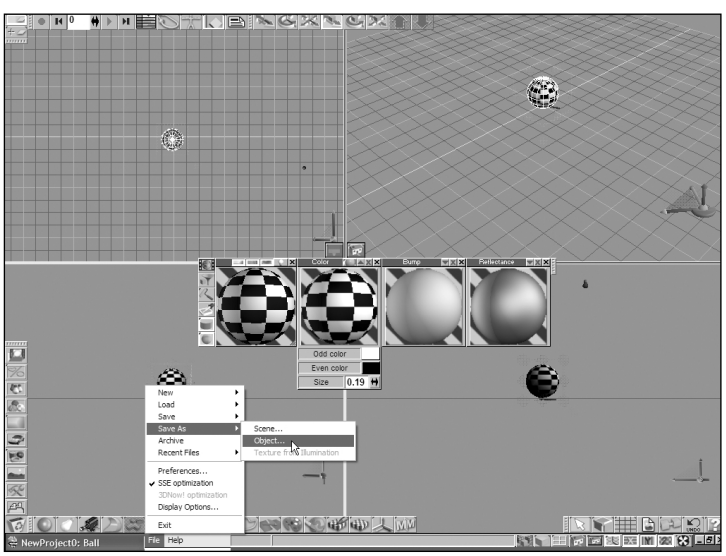


Figure 4.49
Choose File > Save As > Object.

2. Create a new directory to store your ball object. See Figure 4.50.

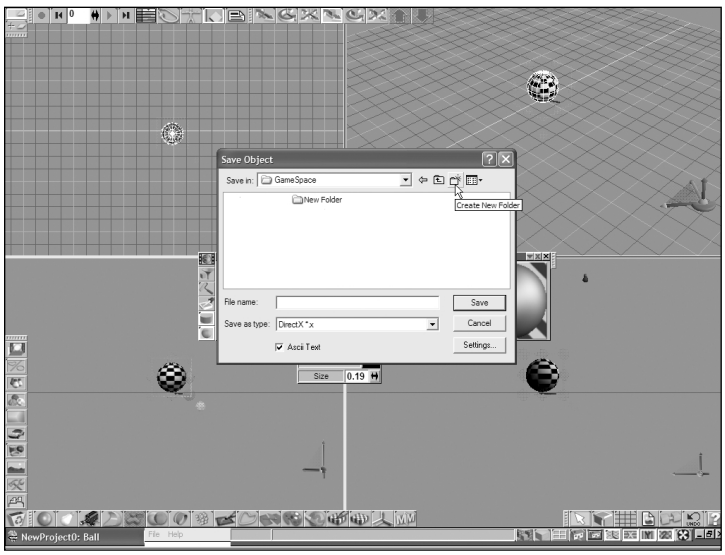


Figure 4.50
Create a new directory by clicking on the Create New Folder button above the directory view.

3. Click on the name of the new folder, and change the name to Ball. See Figure 4.51.
4. Be sure to choose DirectX .x file format from the file type list.
5. Double-click on the new Ball folder to choose it as the destination for your file.
6. Type the name Ball.x into the file name field at the bottom of the Save As window.
7. Click Save.

Your object files should now be saved appropriately.

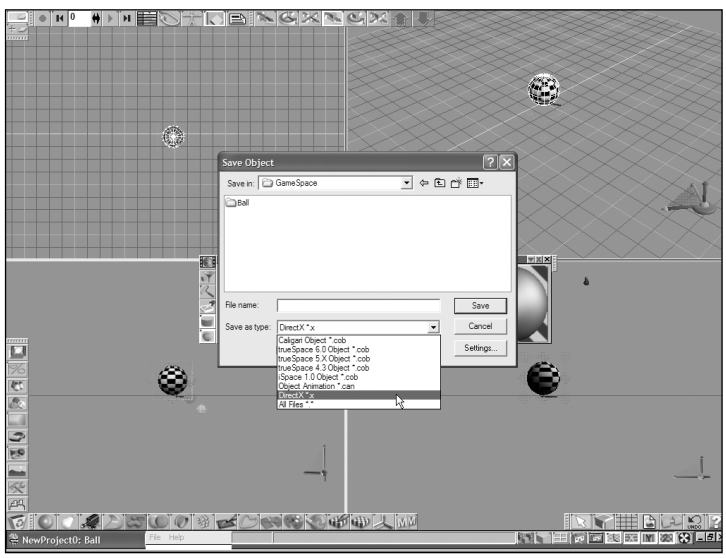


Figure 4.51
Rename the new folder and select the DirectX .x file format.

Close gameSpace by clicking on the little X at the bottom right-hand corner of the screen. Get out some quarters. It's time to play a video game!

Gaming with Objects

Congratulations. You have created your first game model. You have given it a texture. You have even saved it as a real game object file. Now all you need to do is to make a game. Fortunately, I just happen to have a game that's seriously in need of new game objects. If you combine your objects and my game, just think of the fun we can have.

Adventure Explorer is technically a third-person adventure game. Unfortunately, in its current state, it is not very exciting. Actually, without your new game models, it is nearly impossible to play, but it still looks kind of neat. If you haven't taken a look at it yet, this is a good time to give it a try.

Launch Adventure Explorer and take a moment to explore.

Playing in the Water

Adventure Explorer drops you into an underground cavern. See Figure 4.52.

Like most adventure games, your goal is to collect treasure and to find your way out. You will have to create many of the objects necessary to complete the game.

Figure 4.52
Explore the banks of a shimmering river.

Use the Left and Right Arrow keys to turn around. Use the Up and Down Arrows to move forward and back, and hit the spacebar to jump. See Figure 4.53.

Figure 4.53
Try to jump the stream, but be careful. Don't fall in!

If you happen to get too close to the water, you may find yourself stuck beneath the river. You can always press the R key to restart the game.

It shouldn't take you long to realize that the stream is a bit too wide to cross. I'll admit this is not a whole lot of fun yet, but you can see there is a lot of room to grow. What this game really needs is something to do with the water. Perhaps a half dozen soccer balls will do the trick.

When you get tired of watching the water or diving into it, you can press Esc to quit. If more than thirty minutes have passed and you are still trying to cross the water, congratulations, you obviously have the heart of a master gamer. Now stop playing in the water and get back to work.

Adding Models to a Game

The process of bringing a new model file into a game is traditionally called importing. The game engine with which I programmed Adventure Explorer understands the very same .x files that gameSpace saved.

The only trick to using your own models in Adventure Explorer is in the process of moving, copying, and renaming the appropriate files so that they can be easily located by Adventure Explorer.

To make this process as easy as possible, I created another program called ObjectImporter. See Figure 4.54. It has a simple dialog box with a bunch of clearly marked buttons. It is not very pretty, and it's far from impressive, but it couldn't get much easier to use.

You will find ObjectImporter in the same directory as Adventure Explorer. It only functions in this directory, so please don't try to move it.

Launch ObjectImporter and click on the button labeled Choose Object 1. Do not try any of the other buttons yet. You will get to them soon enough.

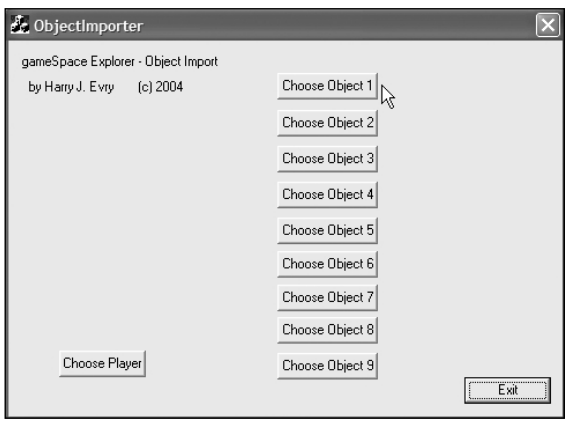


Figure 4.54
ObjectImporter takes the pain out of importing your models.

Another window opens, asking you to "Select a Model File to replace Game Object 1." Navigate to the new directory where you saved the object files for your ball. When the name of your .x file appears in the file selection window, click it. Then click on the button labeled Open. See Figure 4.55.

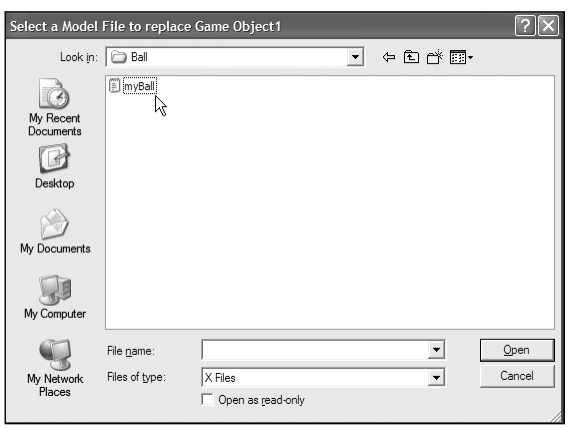


Figure 4.55
Find your object's .x file in the new directory you created for it.

Finally, ObjectImporter asks you to confirm your decision. See Figure 4.56. This is your last chance to keep your old model files as they were.

ObjectImporter does not delete any of the original files that you place into your own new object directories; it simply copies these files to its own internal directories. That means you can safely go back and forth between different object models for the same game objects, as long as you originally saved them each in their own separate directories.

On the other hand, if you later change your object's .x files and want to see the changes in the game, you must run ObjectImporter again to reselect the updated model file. ObjectImporter takes a virtual snapshot of the current state of your selected object file, but never updates that snapshot unless you tell ObjectImporter to select the object file again.

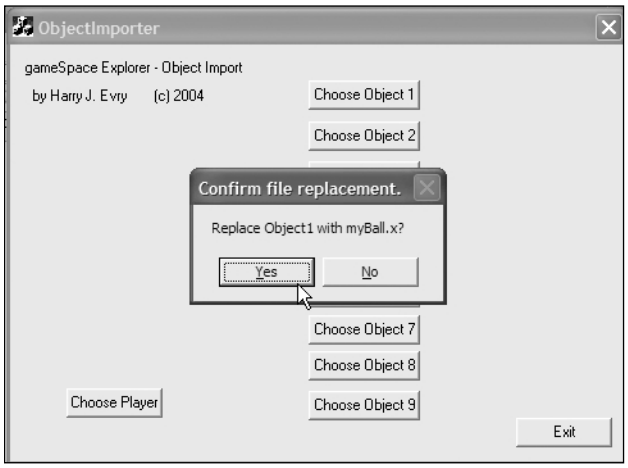


Figure 4.56
Click the button labeled Yes to import your new object files.

Once you have clicked on Yes, close ObjectImporter and launch Adventure Explorer. You should immediately notice a change above the water. I am not sure that floating soccer balls are the answer to all the game's problems, but if your screen looks anything like Figure 4.57, you are well on your way to becoming a game artist.

Figure 4.57
Your new objects have been added to the game.

Summary

In this chapter, you should have learned the following concepts:

- Geometric primitives are the most basic three-dimensional objects.
- Many game objects can be modeled using simple geometric primitives.
- Geometric primitives can be scaled, deformed, rotated, and combined.
- gameSpace uses icons and toolbars to organize its commands and features.
- gameSpace uses the right mouse button to drag things up and down in space.
- gameSpace lets you choose between single or multiple viewport configurations.
- Orthogonal viewports restrict mouse movement to two dimensions, while perspective viewports allow mouse movement in three dimensions.
- The Object Info Pane gives you precise numeric control of your objects' parameters.
- The Object Info Pane reports the number of faces and vertices contained in any object.
- Game artists must carefully balance the detail of their models against the number of faces and vertices they require.
- Placing an object at location 0,0,0 centers the object in the gameSpace world.
- Always use consistent units to describe your object models.
- Textures can be used to enhance the illusion of reality and detail in game models.
- gameSpace materials are used to assign visual textures to an object.
- gameSpace materials are made from combinations of shaders and bitmaps.
- Shaders are like automated numeric texture factories that can be modified and controlled by user-supplied parameters.
- Game object files should always be saved in their own separate directories.
- gameSpace .x files can be directly imported and used in Adventure Explorer.
- ObjectImporter copies new object models into Adventure Explorer.

Questions and Answers

Q: What are common geometric primitives?

A: Geometric primitives are the most basic three-dimensional objects, such as cubes, cones, and spheres.

Q: What numeric location is considered the center of the gameSpace world?

A: The X,Y,Z coordinates 0,0,0 represent the center of the gameSpace virtual environment.

Q: In terms of the X, Y, and Z axes, what directions do the lines on the visible grid in the gameSpace perspective view typically point?

A: In the gameSpace perspective viewport, the grid lines run in the same direction as the X and Y axes.

Q: What mouse maneuver would you use to move an object up in the air when working in the gameSpace perspective viewport?

A: Right-click and drag upward to move a gameSpace object up in the air when working in the perspective viewport.

Q: What mouse maneuver would you use to move an object up in the air when working in a side viewport?

A: Side viewports, like any orthogonal viewport, are two-dimensional projections of a scene. Therefore, you merely left-click and drag to move an object up and down in a side viewport.

Q: If the cone icon was visible in the first slot to the right of the recycle bin, how would you set up gameSpace to build a cylinder?

A: To select the Cylinder tool, click and drag upward or downward on the visible cone icon until the cylinder icon is visible and highlighted. Then you can release the mouse button and click and drag to draw a cylinder.

Q: What key would you press to quickly select the Object Move tool?

A: The Z key is the keyboard shortcut for the Object Move tool.

Q: How can you find the number of vertices and faces in an object?

A: The Object Info Pane contains data fields that display the vertex count and face count for any selected object.

Q: How do you expand the Object Info Pane or the Color Shader Panel?

A: Click on the red triangle or red downward-pointing arrow tip in the upper right corner of either panel.

Q: Where can you find gameSpace's predefined materials? How do you get there?

A: The Material Library contains a variety of predefined sample materials. Click the icon showing a bowl of painted balls to open the Material Editor.

Q: Why is it a good idea to save different .x files in their own separate directories?

A: DirectX .x files can save their meshes and textures in separate files. It is not uncommon for a single object model to require a number of supporting files. Placing these files in their own directories makes it easy to track which of these files must work together.

Discussion Questions

1. Pick any object in the room and discuss how you would model it using geometric primitives.
2. What are the key differences in concerns and approaches between a 3-D artist building models for video games and the same artist building models for movies, architecture, or illustration?
3. What are some advantages of building models from geometric primitives?
4. What geometric primitive typically requires the most faces? Why?
5. What are the principal goals of texturing in a game model?
6. What are the advantages and disadvantages of working in a multiple viewport configuration?
7. What are some important differences between .x files and .scn files, and when would you use each?
8. In what situations would you need to use the Object Info Pane?
9. If you were going to replace your soccer ball model with something more appropriate or useful to players in the Adventure Explorer, what might it be?

Exercises

1. Design a game vehicle using only standard primitives.
2. Use gameSpace to model a cylinder that is perfectly circular at its base and is exactly twice as high as it is wide.
3. Use gameSpace's primitive tools to create a witch's hat and save it as a .x file.
4. Change the texture of your soccer ball to make it look like a beach ball.
5. Use ObjectImporter to replace your soccer ball in Adventure Explorer with the new beach ball from exercise 4 as Object 1.
6. Use ObjectImporter to load the witch's hat from exercise 3 into Adventure Explorer as Object 1. You should first scale the hat model so that it is similar in size to your ball model.