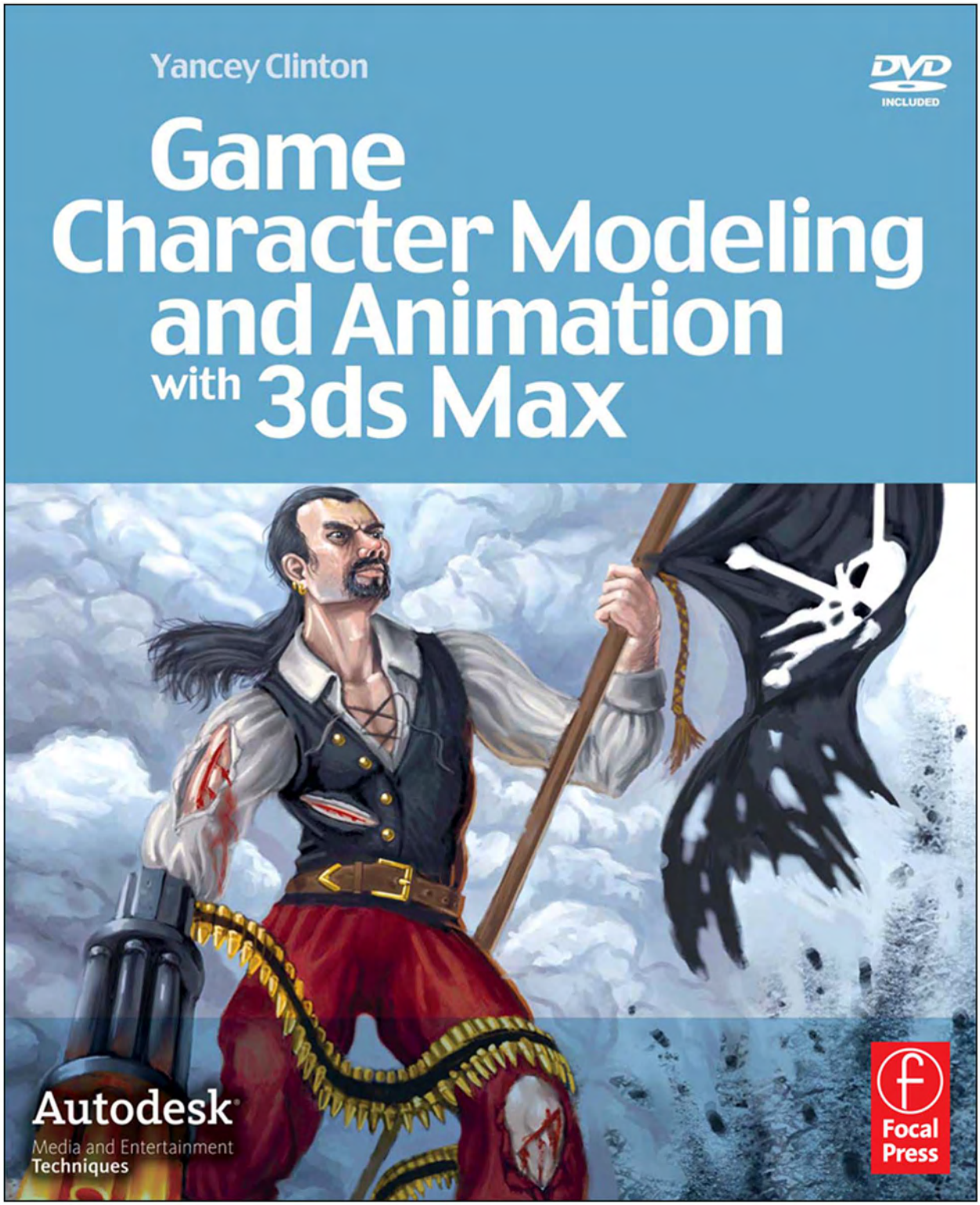
Focal Press

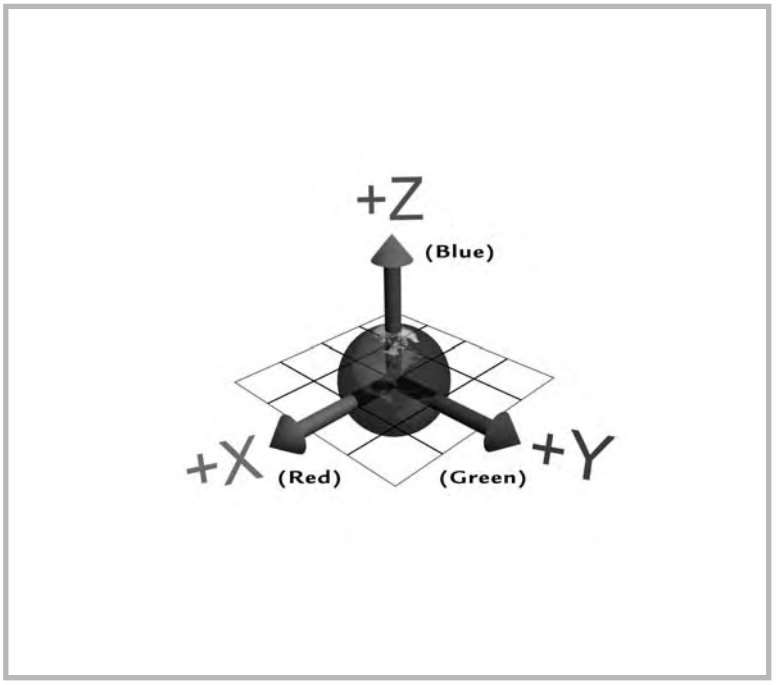
Like what you see?
Get the whole book at the
Focal Press Bookstore

Click Here!

Focal Press

http://focalbookstore.com/?isbn=9780240809786





1

What It Takes to Make a 3D Character

Hardware Resources

There are two ways to look at the computers necessary to use 3ds Max and other programs: the minimum and the maximum configurations. The nice thing about creating real-time assets is that it doesn't take a supercomputer to do it. It can even be done on a sub-$1,000 laptop or desktop. 3ds Max 6 can be run on an Intel Pentium 3 running Windows 2000 or NT with a minimum of 512MB of RAM; to use 3ds Max 7 you will need a little more power. An Intel Pentium 4 running Windows XP SP 1a or better and at least 1GIG of RAM should work fine. The 3ds Max 8 and 9 work just fine on my Centrino Duo laptop with 1GIG of RAM; I suggest that you try 3ds Max 9 first.

The other hardware you will need is a current video card. In order to run the Unreal Tournament game you will need a video card that is DirectX 9+ compatible. While it is possible to use a video card that is DirectX 7 or DirectX 8 compatible, unfortunately, the game will hardly run and in some cases not at all. This, however, is not a deterrent to creating the assets, just to implementing them and seeing them in a game.

One other piece of equipment that is not absolutely necessary for using 3ds Max, but is extremely useful is a three-button mouse with a scroll wheel.

There is no maximum. 3ds Max and Unreal Tournament will use as much computer power as you can throw at them.

Software Resources

You're in luck; most of the software necessary for this course is free or can be used in demo mode. At this time a demo version of 3ds Max 9 that can be used for 30 days can be downloaded from the Autodesk website. The companion DVD with this book contains a demo version of the Unreal Tournament game but, unfortunately, not the Unreal editor (you have to buy the game to get that). So if you want to, you can do this project super cheap, just as long as you finish within 30 days. Fortunately, there are other options for owning 3ds Max. The cheapest is a 1-year student/educator license, which at this time is only $180. The second option is for an unlimited-time student/educator license for around $600 to $700, depending on options. The last licensing option is a commercial one that will run you around $3,600. This option is necessary only if you plan to sell or get paid for work generated using 3ds Max. I also suggest that you buy the Unreal Tournament 2004 DVD version, which at this time sells for around $30. The last piece of software that is not necessary but useful is Photoshop. At this time there is a downloadable demo version of Photoshop CS3 from Adobe's website, but again it is not necessary to complete the project. There are also a number of freeware and shareware image editors with similar functionality.

Time Resources

I am not going to kid you on this one. There is a steep learning curve, and it takes time to learn a 3D animation package like 3ds Max. Fortunately, one way to speed up the process is to play any real-time game. Yup, you heard me—for homework I want you to play games. I have found that my students who avidly played 3D real-time games had a greater advantage over the ones who had not. It turns out that the game will train your brain to think in the virtual world. The coordination involved in navigating a virtual three-dimensional world space mimics the navigation used in 3ds Max. A prime example is that it helps to use both hands when working with 3ds Max. One hand is for keyboard shortcuts; the

other is for the mouse. This is the same for all video games, PC- or console-based; you have both hands on the controller, each button doing a different thing.

Once you have played enough, there is a point at which you no longer have to think about the controller and the things you want to happen. This will eventually happen with 3ds Max also. So here are the time frames I am talking about. For a professional modeler, it should take less than a week to create the model we will create, including the image used for the texture. I will introduce you to a technique that I use for creating textures, but to get into the details would require another book.

Because each student is starting this project with different experiences and backgrounds, it's difficult to pin down an exact time frame it will take to finish the project. What I can tell you is that the DMA summer class was a super-intensive 1-week class, where we worked 6–7 hours per day for 5 days straight to complete the project.

Depending on your experience, it may take you longer. That seems like a lot of time, but remember that you're still learning. The next time that you want to make a model it will take you about half as much time as it did for the first. The tutorial is a linear step-by-step process, so it can easily be broken up into sessions that are comfortable for you.

Of MODs, Maps, and Machinima

In the world of game modification, there are three different levels of application. A mod is a modification of an existing game. For example, adding a new character, weapon, or vehicle would be considered a mod. That's what we are going to do. There are tons of different mods you can download for most real-time 3D games. There are a couple of web sites that can help you start moding your games. The first is *www.gamespy.com,* which has everything you need to mod the most popular games. Another is *www.gamespot.com,* which has tons of good stuff for and about moding.

A map is the addition of a new level to an existing game. To create a map is to make a virtual world space and all that entails. The sky, the land, the water, and the physics are all things that can be created and modified in the games editor. Then, using 3ds Max, anyone can create objects to be placed in that environment. Maps can include characters, but it's usually a lot of extra work to add them. There are tons of maps available for popular 3D games.

Another kind of mod is called a Total Conversion mod (TCMod). This is a new game that uses an engine designed for a completely different type of game. One extreme example of a TCMod is something like the game Battlefield 1942. This is a WWII first-person shooter with planes and tanks from that era. Currently, there is a downloadable TCMod available for it called Galactic Conquest. It basically turns a WWII game into a Star Wars-style game, completely replacing the game's original WWII-era design. Most of the time, maps are not that extreme. They usually have the same kind of play as the original game. A prime example is Unreal Tournament 2003 and 2004. There are literally hundreds of different mods and maps that you can download and play, but they all basically have the goal to kill the monsters in their game play.

The level of difficulty in creating mods and maps for any particular game varies extremely. The support for most game editors is limited to the user base. This is one of the reasons I chose Unreal Tournament 2004. It has lots of documentation and a large user base for support. Even though it was easy for me to create a player mod, it was not very easy to create my own map. As it turns out, the game engines are very particular in the way they want you to make things.

There are literally hundreds of games that can be modified in one way or another as long as you use the same tools that produced the game in the first place. Those tools consist of the 3D modeling program and the real-time game engine's editor. As there are only a few real-time 3D game engines, most of the real-time 3D games are made with the same three or four engines. For the Unreal Tournament 2004 game, the Unreal Engine 2 is used to run the game and the Unreal Editor 3 is used to edit the game.

The Unreal Engine 2 is also used in 20 or so other games of the same style, like Splinter Cell and America's Army. For more details on the Unreal Engine, check out *www.unrealtechnology.com,* and while you're there check out the Unreal Engine 3 documentation.

This goes the same for id Software's engine. It's used for DOOM 2 and QUAKE; the most current game using their engine is DOOM 3. This is one of the top five technologically and visually stunning games on the market today.

The Unreal Engine 3 is the current top of the line real-time engine, that is changing the way games look for the better. It was late 2006 when the first games created using the Unreal Engine 3 were released. The look and level of detail are unparalleled; I can't wait until Unreal Tournament is released.

So, using 3ds Max to create your own assets, you can just replace or add to the existing game architecture. The big trick is how to do

this. There are plenty of examples of games that were not intended to be modified that were hacked and the assets exchanged. This would be the hardest type of game to modify. The easiest would be ones that included an editor for the specific game. There are only a handful that do, and most of them I have just told you about. There are some third-party game editors, but they are not the easiest programs to learn and or use.

An interesting side effect of game moding is machinima. The word is a conjunction of the words "machine" and "cinema." Machinima is a new form of filmmaking. One little known aspect of the more common 3D games, like Unreal Tournament 2004, Half-Life 2, or even QUAKE, is the ability to animate and record free-floating cameras. This allows you to shoot a film in the real-time game. The tools you will learn in this book will allow you to make your own characters for your film. One of my favorite machinima serials is called Red VS Blue. It was made in Halo and does a whole lot with the simple environment of the Blood Gulch Map. For further information, check out *www.machinima.com*; it's a good place to further your education on all things machinima.

An Introduction to the 3D Environment

What is a 3D environment? Well, here's a hint: you live in one. While there are lots of similarities between the virtual 3D environment in the computer and the real world, there are several really big differences. The real world is limited by physics, and the virtual world is blissfully free of most rules of physics. Not only that, we are free to change any of them. Things like time, space, and size matter little in the virtual world. We have unlimited time both forward and backward, and we can work from the galactic scale all the way down to the microscopic without changing projects.

There will be two different 3D environments that I will introduce to you during this project. The first is that of 3ds Max. The other two are the Unreal Editor and the Unreal game itself. If you understand 3ds Max's environment, then the others should be easy for you to pick up. The best way to think about this is that you have an infinite, empty void. At an arbitrary point in this space is what we call the origin or center of the world. This point is our reference point, and the values are 0,0,0 on the X-, Y-, and Z-axes. To find any point in space, you need three values that reference the origin. The three values are labeled X, Y, and Z, and they define the offset

for the origin. The letters have their origins in math and are used to plot three-dimensional points in space.

The part of coordinate systems that isn't usually talked about is orientation, or which way is up. In 3ds Max, Z is the up coordinate, but this is not true of all 3D applications. The other two axes, X and Y, create an infinite plane that goes on forever, as shown in Figure 1.01. This is what we call the world coordinate system.

In contrast, in the Unreal Editor, the Z coordinate is not up and down, but in and out. The reason is that the video game engines use a camera view for orientation reference. That is, the screen is the X–Y plane, and moving the camera forward and back is the Z coordinate, as shown in Figure 1.02. This is one of the reasons we have to use the Unreal Editor to interface with the game engine, to convert things like the coordinates.

In 3ds Max's virtual world there is no inherent gravity, and the objects in this world are not solid. Those effects have to be simulated using 3ds Max's integrated physics simulator, called Reactor. It does this in a way similar to that of the game engine when you're playing. In fact, the company that made the Reactor plug-in for 3ds Max, Havoc, is now being integrated into the next generation of game engines. This brings 3ds Max even closer to being a direct game-authoring tool.

Figure 1.01

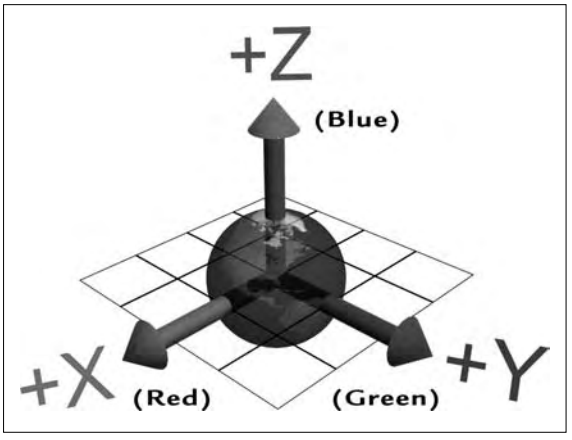
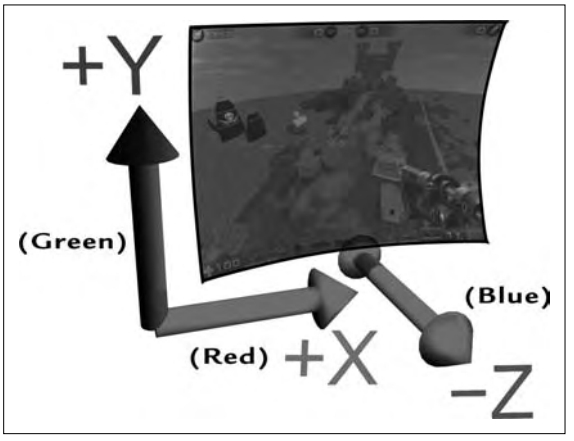


Figure 1.02



What Makes Up A 3d Character?

Mesh

The primary thing a 3D character is made of would have to be geometry. A broad definition of 3D geometry would be an object that can be edited and rendered. 3ds Max includes a few basic 3D geometric objects called primitives. One of the primary primitive objects we use is a box. As a primitive, the ways in which it can be edited are limited to things like the length, height, width, and the resolution of each dimension. If this were all we could change, it would be extremely difficult to create anything more complicated than box-shaped objects. Instead, we usually adjust the parameters of the primitive box and then convert it into an editable mesh, a much more editable form.

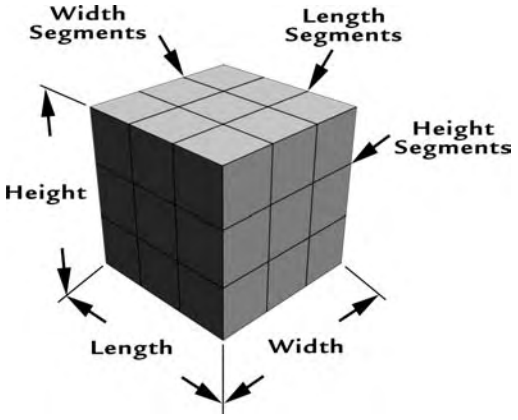
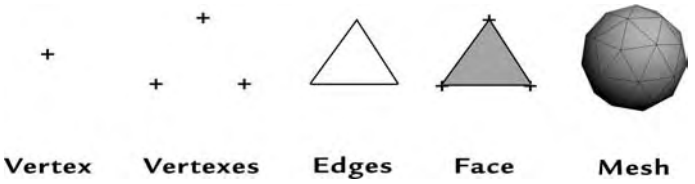


Figure 1.03

This opens up a whole other array of things we can play with. As an editable mesh we now have access to the sub-object parts of the box. These sub-object parts are the bits and pieces that give the box its shape and allow the program to render or draw the surface. The smallest bit is called a vertex. A vertex is a point in space denoted by a plus symbol (+). Then, between the vertices of an object is what we call an edge. It looks simply like a straight line between two vertices, kind of like a three-dimensional connect-the-dots. Then, if we connect three vertices by three edges, we can create a renderable face, the third object. If you put a bunch of faces together, you get a larger shape that we call a mesh.

Figure 1.04



There are several names for meshes; a mesh can be called a model or geometry. The density of the model or the number of faces in a model defines the smoothness of the rendered surface. For example, the digital dinosaurs used in *Jurassic Park* were made of several million triangles. That was in 1993. Now, the next generation of game engines will be able to handle characters of 5,000 triangles and will look like a model of a million or more. This will give us the ability to create film-quality models for real-time games.

Texture

The second most important part of a model is the texture. This is the image that will define what the rendered surface will look like. That's easy enough to say, but just a little harder to do. There are two big steps in creating a texture. The first part is called unwrapping. The problem is that an image is a two-dimensional thing, and we need to place it on a 3D object. In order to do that we need to separate the model into logical parts that can be flattened easily in order to place the image on it without distortion. Because the image is flat, it is much easier to place the map if the model is flat, too. Stretching is the bane of the texture artist.

There are two components that come into play when you're creating a texture map for a 3D model. The first component is the mesh. The second is the image. They both have limits on how much they can be modified in order to accommodate each other. Change one too much, and the whole process will fail. So let me break down the components a little more.

When you create a digital image it is made up of little squares called pixels. No matter how many pixels your image has, they remain square. The geometry is made up of triangles and unequal ones at that. The distortion in the shape of the geometry (triangles) causes the image placed on it to distort, and the pixels appear stretched out. This is because the image is trying to cover a nonsquare distorted surface. An easier way to imagine a texture is as a slide in a projector. If you project the slide on a flat screen, everything looks great. If you project it onto an uneven surface like a ball or a sheep or a ruffled curtain, the curves of the object distort

the image. Texture maps work the same way. Fortunately, you can directly edit the way the map (projector) sees the surface of the object (screen) by editing the mapping coordinates. This is one of the trickier parts of the texturing process. I will show you the way.

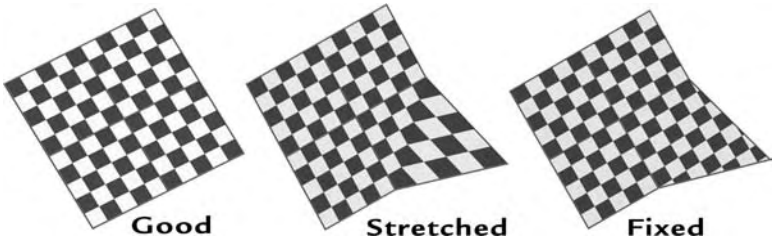


Figure 1.05

The second step is to create the image. Before we can do that we need to modify the mapping coordinates until all the geometry is laid out flat. We can take a picture of the flattened geometry and use Photoshop to paint directly on that image. When we apply the image to the geometry, everything should go where we painted it. What to paint and how to do it is a complicated process. There are many different styles and methods for creating images. For the painters out there, you can just paint what you want; for the rest of us, we can use pictures to cut and paste our way to a map.

Figure 1.06

Skeleton

The third ingredient is a skeleton. I am sure you're asking yourself, "What in the heck do you need a skeleton for?" Well, in the old days if you wanted your model to move around and bend like a human, you needed what was called an armature. When animation was done with Play-doh characters, there was a wire that ran inside

Figure 1.07

the "doh" to give it a substructure and support. We will revisit the subject of wire a little bit later. For now, think about your own skeleton. It's the same principle. We have bones to support the muscles that allow us to stand up and move around. They also constrain our movements. Basically, the skeleton acts like your bones, and the mesh simulates the flesh.

I wish I could tell you that this process is completely automated, but that would be a lie. The process of adding a skeleton to your model is called rigging, and it has two steps. The first step is to create a skeleton. This is a mostly automated procedure; for example, you just have to click and drag to create a biped humanoid. For creatures other than humanoids, there are easy ways of distorting the bones' shapes, even adding limbs or a tail. Once the skeleton is fitted inside the model we have to connect the two. This is the second step. What we have to do is turn the skeleton into a deformer of the mesh and tell the program what bones will control which vertices. This is also mostly an automated process. Unfortunately, we have to check the computer's work on a bone-by-bone basis. After all the obvious problems are fixed, we have to do animation tests to make sure the rig looks good when the model moves around. This leads us to the second-to-last procedure.

Animation

Animation for video games is a little different than you might imagine. Cyclical animations or actions are strung together to create the illusion of continuous movement from a limited library of motion. So when you press the Jump button, the game runs the jump animation, and when that's done, the game starts the run or stand animation. For every action you see in a game, there has to be an animation that can fit seamlessly with all the others.

Luckily for us, all the individual animations for the Unreal Tournament 2004 characters are included with the game. So we don't have to make all the animations from scratch. We will just cut and paste to add all the animations to your new character, unless you want to make a completely differently shaped character. For example, there is a raptor pack that you can download and play as a raptor. As you can imagine, the animations for a raptor would be quite different from those for a human. We will get into a little bit of that process in Chapter 7, but not into actually creating animation specifically. There are some great books on the topic of character animation; the one I suggest for my animation students is *The Animator's Survival Kit* by Richard Williams.

Compiling

The last step in the process is to compile all the different parts of your character into files that the game can use. This is another two-part process; one part will be done in 3ds Max and the other in the Unreal Editor.

After we have built our model, textured it, rigged it, and tested it with animation, it's ready to go. We will use a plug-in called Actor-X to export your models' information in a form that can be understood by the Unreal Editor. Then, from the Unreal Editor, there will be one more compilation into files that the game engine can understand. It sounds complicated, but this is one of the easiest parts of the whole process and should take about an hour. Then we can play our new character in the game.

The Process

Now that I have given you a brief overview of the entire process, let's do a short recap. First, we are going to make the model in two parts: the body and then the head. Next, we will unwrap the mapping coordinates of our models and apply the map. Then, we will add the skeleton and attach it to our models. Finally, we will export and compile our models in preparation for game play. By now you should have learned some new terms and, as we tour the interface, you will learn a whole lot more. In the next chapter we will take a look at the interface of 3ds Max before we start the project. For those of you who have experience with 3ds Max, I suggest that you scan over the next chapter, paying particular attention to the spline sections.

The Toolbox

Throughout most of this tutorial we will be using 3ds max. A simple way to think about it is that 3ds Max is a tool set, a whole big box of tools all arranged and laid out for you to use in the most marvelous erector set imaginable. You're limited only by your imagination and time. We are going to start by learning how to make geometry.

The 3ds Max Interface

The current version of the 3ds Max has an incredibly easy-to-use and easy-to-learn interface. Believe me, there are much harder 3D

programs out there. Even 3ds Max has come a long way from its humble beginnings as a DOS program. The first class I taught was in the last DOS version of 3ds Max, Release 4, and 3ds Max 8 is infinitely better. It's this evolution that makes it possible to create such a complex project as this one in such a short time frame.

Primitive Creation

There are lots of different things that can be created with 3ds Max, but we are going to focus on just the parts we will be using for the project. When 3ds Max opens, on the right-hand side of the screen you will see a panel with six tabs on it. This area is called the command panel, where we will be creating and modifying things. The tabs open different panels. The first tab displays the Create panel. The buttons underneath the Create tab determine what kinds of things you can create. For the most part, we will be using the first two panels. The rows of seven buttons at the top of the Create panel are the object category buttons. The active object category on the Create panel is the yellow button; by default it's the geometry category. Under the object category buttons is a drop-down menu that contains many different kinds of geometrical objects that can be created. We will not be getting into all that 3ds Max can do, but I suggest that you play around with it as much possible.

Click the GeoSphere button in the center of the panel. In the window labeled "Top," click with the left mouse button and drag out a sphere.

You can use the left mouse button to change the active viewport, but it's a bad habit to get into. Instead, you should practice using the right mouse button to change the active viewport so that you will not deselect the currently selected object.

Take some time and try creating all the standard primitives. Some of the primitives take more than one click or drag to create. Don't fret, because we are working in a virtual space. You can't break anything, so it's okay to experiment. While there are other things we can create, like lights and cameras, we should stick with primitives for the time being. This will help you get used to the interface and become more proficient with the way that 3ds Max uses the mouse.

One thing I want you to take care to notice is what viewport or window I ask you to make an object or modify an object in. Most of the time I will ask you to make things in the Top or Front viewports. It is difficult to make objects in the Perspective viewport. We generally use the Perspective viewport to check the changes we make in the other ones.

Viewport Navigation

Now that you have some objects in your scene, you will want to take a look around to see them from all sides. If you take a look in the bottom right-hand corner of the interface, you will find the viewport control tools. Some of them you might recognize, like the Zoom tool. If you select it, you can go to any viewport and zoom in and out by clicking the left mouse button and dragging the mouse up and down. Go ahead and give it a try.

If you have a wheel mouse, I want you never to use that button again. The middle scrolling mouse button will act as an incremental zoom. If your mouse does not have one, I recommend that you get a mouse that does. It makes navigating in 3ds Max much easier.

So go ahead, click on any viewport and scroll away for a little while. If you hadn't noticed, you can also use the middle mouse button to change the active viewport. Doing so doesn't deselect the currently selected object.

The next button you might recognize is the Pan View tool. It's down and to the right, with the flat hand icon on it. Select it, and in any viewport click the left mouse button. You will be able to pan your view around. This is what we call a screen pan; no matter what viewport you're in, it will pan in the same way. Now that you know what this button does, I want you never to use it either. We actually use all four mouse buttons. Try depressing the scrolling mouse wheel. You should feel and hear a click, which is the fourth button. If you click

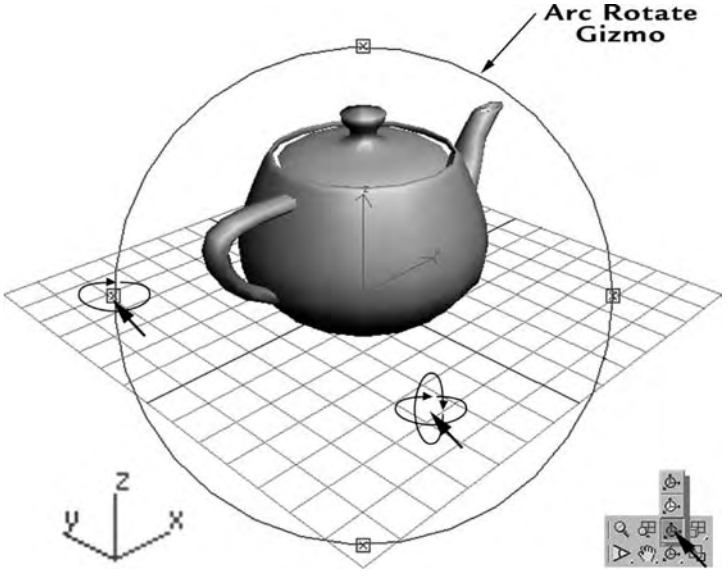


Figure 1.08

the scroll wheel when you are in a viewport, it will change to the Pan View tool. This takes a little practice to get used to, so try it out.

The next tool we are going to look at is the one that allows you to rotate your view of the project. The use of this tool does not rotate the objects in your scene, just your view of those objects.

It's the Arc Rotate tool. It has a circle with three arrows pointing out of it. If you click and hold down the left mouse button, you will see a pop-up menu giving you three options. Select the icon with the yellow circle. This is the option for sub-object rotation. The other two options are for rotation around the center of the world and the selected object. Once you activate the tool, a yellow gizmo will appear overlaid on top of your active viewport. If you change the viewport, the gizmo will follow. Now put your cursor in the center of the gizmo, click the left mouse button, and move your mouse around. It's kind of fun to give a spin, but don't worry if you can't control it well. Try clicking and dragging one of the boxes on the gizmo; this way you can limit the axis of rotation. This is one of the hardest tools to become comfortable using. It is also one of the most important tools you have for working in the 3D world space.

It takes a fair amount of time truly to understand the 3D world space. That's why playing 3D games can help you train your brain to understand a virtual world. The problem is that we have a 2D interface into a 3D world. It would be like trying to view the real world out of a small window. It would be almost impossible to navigate the real world with such a small view. So we end up rotating our view to make sure we are working correctly. Of course, there is a mouse shortcut for the Arc Rotate tool. It's the middle scroll wheel button and the Alt button on the keyboard. So give it a try in any viewport. Hold down the Alt button, put your cursor in the middle of the viewport, and, with the middle mouse depressed, move the mouse around.

Having all the viewport navigation tools oriented around the cursor dramatically increases your working speed. Here's just a small plug for one of my favorite new toys, the 3D mouse. You can check them out online at *www.3dconnexion.com*. This device replaces the Pan View, Zoom, and Arc Rotate tools altogether and gives you the ability to do all three at the same time. This greatly improves the quality of your artwork while at the same time decreasing the time it takes to complete any kind of project. They have extremely good student pricing, as well as a 30-day trial program.

For now, I want you to practice all the things you just learned. The more time you spend on using the viewport tools, the faster, and easier the whole learning process will be.

Figure 1.09

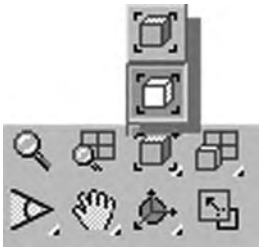
There are a couple of viewport navigation buttons that are not self-explanatory. The first is in the very bottom right-hand corner of the interface. This button will make the active viewport expand and contract. The default keyboard shortcut for the same action is Alt-W.

The other buttons are right above it. The Zoom Extents All tool (the button with a small grid in the background) will zoom all the viewports to the extent of your project.

Just to the left of the Zoom Extents All tool is the Zoom Extents tool. This will zoom only the active viewport to the extent of your project, so if you ever get lost, click this button to reset your viewport.

Now you might ask why we have two buttons that do almost the same thing. Well, notice that in the bottom right-hand side of each button, the corner is knocked out. This means that there are other options for these buttons. Try holding the left mouse button down on one of them, and you will get a pop-up menu with two options. Select the one with the white box on it. This version of the button will zoom to the extent of the selected object.

Figure 1.10



Making Selections

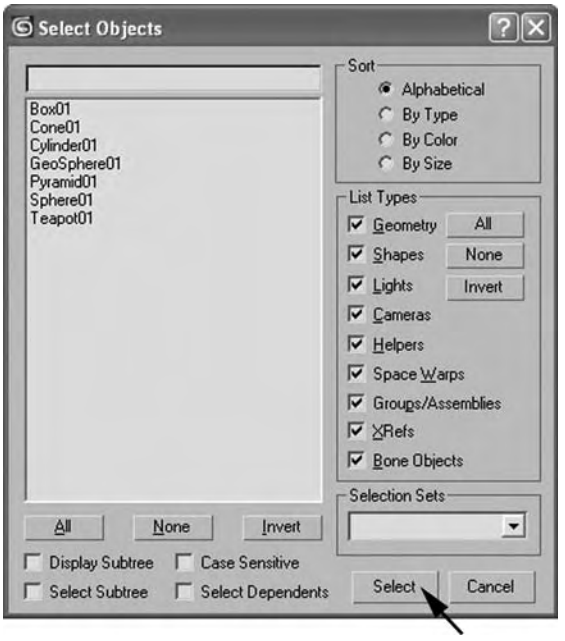
Most of the general working methods in 3ds Max are similar to the ones you are used to in Microsoft Windows. There are lots of drag-and-drop, click-and-drag, and right-click options. Let us not forget the ability to click and drag a rectangle to select multiple objects. As you might guess, if you click and drag out a rectangle in 3ds Max, the objects that are crossed or enclosed by that rectangle will be selected. It takes a little getting used to, so give it a try. Select individual objects by left-clicking on them. Try dragging out a rectangle and selecting multiple objects. Now that you have practiced that a little, let's add in another twist.

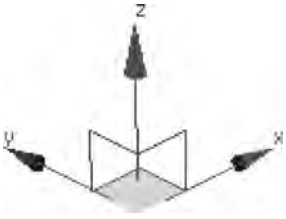
There are two keys on the keyboard that we use to augment our mouse clicks: the Ctrl key and the Alt key. The Ctrl key changes the selection tool to an additive selection tool. To the left edge of your cursor, a little plus symbol (+) appears. This also works with the drag window. If you hold down the Ctrl key before you drag out the window, it will be an additive window. The Alt key is for subtraction, so if you hold down the Alt key and select a selected object, it will be deselected. You should also be able to see a little minus symbol (–) next to your cursor.

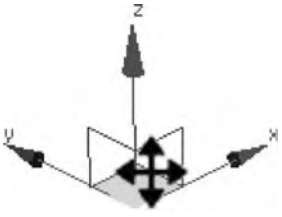
The last method of selecting objects is by a list. In the main toolbar there is a button called Select by Name. If you click this button, a pop-up window will give you a list of all the objects in your

project. The keyboard shortcut for the Select by Name window is "H." You can use the normal Windows selection keys, such as Ctrl and Alt, for adding and subtracting objects from the list. Then you can confirm your selection by clicking Select.

Figure 1.11







Primitive Editing

Transforms

The most primitive editing we can do is to change the position, rotation, and scale of the object, so that's where we will start. First, let's try moving things about. At the top of the interface, right in the center of the Main toolbar are three buttons. The Select and Move tool is the one that looks like a "+" with arrows on all four ends. If you click that button, it will turn yellow. Now select an object.

On top of that object will appear the Move gizmo. In the Front (f), Left (l), and Top (t) viewports you will only see two axes, X and Y. In the Perspective viewport you will see all three axes. The three axes are color coded red for X, green for Y, and blue for Z (remember, RGB = XYZ). This color coding is consistent throughout the program, so if you learn it now, when we get to more complicated things you will understand them more easily.

Let's start in the Top viewport. If you want to move the object around on the two axes, put your cursor on the red and green lines