3D Graphics Demystified

Creating Worlds

Almost all computer systems shipped today – from the humblest sub-$500 PC to the most sophisticated workstation – have some form of 3D graphics. But most of the marketing and advertisements never really describe what 3D graphics are or why 3D graphics are useful. There's often a vague association with computer games, but little else.

The Purpose of 3D Graphics

3D graphics in general are used to create realistic looking environments that are visible through the 2D screen of a computer monitor. These environments are designed to be fully immersive – meaning the user can "see" or "move" in any direction, at will. The most sophisticated of these environments are highly interactive, allowing users to move or change the appearance of objects.

The Challenge of Interactivity

You may have seen the stunning visual images that exist in the computer game *Myst*. In fact, *Myst* is *not* interactive at all, but a series of pre-rendered pictures that are brought up each time you take a "step" in the world. You cannot turn in any direction, only in fixed increments. You can't take a virtual swim in the water, nor can you "climb" on top of the many buildings you see. While the authors used 3D graphics tools to create the images, they are simply a series of static images.

A movie is not interactive, either. Movies like *A Bug's Life* or *Jurassic Park* make heavy use of computer graphics, but those are all pre-generated – you can't manipulate the objects on the screen.

Creating a 3D world that's visible on your computer screen is a mathematically challenging application. Allowing a user to manipulate or interact with the virtual environment at will, in real time, is the goal. Even the fastest Pentium® III processor cannot do an adequate job of generating graphics which look realistic *and* allow fluid, realistic movement through that world.

The Applications Conundrum

The most prevalent 3D graphics applications for personal computers have been games. However, as CPUs and graphics processors become more powerful, other applications are emerging. Some are high-end, supercomputer-class applications migrating down to the PC, such as the geophysical visualization applications used by oil companies. But other types of applications are emerging as well. The business world, long mired in the static, 2D spreadsheet, is slowly discovering that 3D graphics can allow a faster grasp of complex business data. Virtual shopping or real estate browsing is now possible as well.

Enter 3D Graphics Hardware

You can think of the graphics chip on a 3D graphics card as a special CPU that can perform specific types of calculations and move certain types of data around very, very fast – much faster than the Pentium III or other processors. In fact, 3D graphics processors such as the GeForce 256™ from NVIDIA are referred to as GPUs – graphics processing units. The job of handling the various parts of an application is split up. Think of it as an assembly line, with the CPU as the foreman and other processors – like the graphics chips – as assembly line workers that can do specific jobs very well. In fact, even within the graphics chip, there's an assembly line of sorts. We'll talk about that in the next few pages.

When looking at reality, we don't pay attention to the things we can't see.

The Real World

Imagine you're standing at the kitchen window of a house. The sink is in front of you and you can see clearly out the window above the sink. Outside, you see a driveway with a car. A woman walks into view, gets in the car and drives away.

This is a pretty prosaic scene, but think about it for a moment. You can see out the window. But what's to the left and right of the window frame? Where did the woman come from? How did the car appear as it drove into the distance?

Now imagine that you can walk around while the above events are taking place. You turn around and notice what's behind you – not a façade (as on a movie set), but the cabinets, refrigerator and other typical items that exist in a kitchen. You turn around, open the window, and lean out, noticing a tree to the left of the window that was hidden by the wall.

Imagine looking out the "window" in a virtual world. What can you see?

Images courtesy of Viewpoint.

The Synthetic World

We normally take what we see for granted. But if we want to recreate the same scene so that it can be viewed on a computer, all the above issues come into play. This is especially true if the content has a high degree of interactivity, as in a game. The graphics programmer needs to worry about the things that can be seen, but also those items that aren't immediately visible, because as the user interacts with the computer, they may alter the view so that items that were hidden come into view.

Think of the computer screen as our kitchen window, above. Peering through our now-virtual window, we see the woman walk out to the car. A moment ago, she was hidden, but then moved into view. The car leaves the driveway, heads into the distance and disappears. If we can "lean" forward in our view, we may see the tree to the left of the window. The graphics programmer – and the graphics software – needs to keep track of things like the woman who moves into view, or the tree we can't immediately see. There's a tremendous amount of detail that must be considered. When the car drives away, how far will it be visible? How detailed is the computer model of the car? How realistic are the woman's movements and appearance?

The graphics hardware and software must consider the unseen as well as the seen.

All motion picture and television is really an illusion. A series of pictures – "frames" are shown on the screen at a relatively high speed – 24 frames each second for film, 30 frames each second for television. It's like the flipbooks we played with as a kid, but all done in living color. But those images are really a series of static images being displayed quickly – they are not generated on the fly, and are not interactive.

When a computer creates a realistic animation, it is simply displaying frames on your computer screen at a high rate of speed – sometimes in excess of 60 frames each second. This requires a great deal of mathematical computation. It happens to be the kind of mathematical computation that CPUs do moderately well. On the other hand, it's possible to design dedicated silicon chips (GPUs) that can perform the types of math needed for three-dimensional graphics and animation very, very well. This creates the possibility of virtual environments that are visually stunning and fully interactive.

Breaking It Down

The problem of creating a virtual world with three-dimensional attributes on your computer monitor has been studied for many years. Like most problems, it can be broken down into component parts. Think of the process of generating 3D graphics as an assembly line. Like the factory assembly line, there are specialists that can handle specific jobs well:

- What the application itself – the computer program – does. For example, if you have an animation of someone shooting a basketball into a basket, the application takes care of how the player moves, the moment the basketball hits the backboard, the movement of the net when the ball falls through it, and the behavior of the ball when it bounces on the ground.
- The overall management of what the user actually sees is known as the *scene*. In our virtual basketball game, each individual frame of animation represents a scene. The scene contains information about what's visible to the user. As a computer basketball player fires off a shot, a number of questions have to be answered. Can you see the opposing goal (the one on the other end of the court)? Is the ball reflected in the glossy surface of the floor? Can you read the number on the player's jersey?

At this point, the problem becomes purely a graphics challenge. Our assembly line for 3D graphics kicks in. In the jargon of the industry, this assembly line for graphics is called the *3D graphics pipeline*. There are four main steps – transform, lighting, setup and clipping, and rendering.

Transforms: The Art of Change

Transformation is the task of handling how a scene changes from one frame to the next. Each frame changes just a little bit. Imagine a series of pictures of a horse running. Each picture captures slightly different positions of the legs moving and the head bobbing. Play back the pictures fast enough, and you "see" the horse moving. However, when creating the animation using 3D graphics technology, the changes that take place from one frame to the next must actually be mathematically calculated. This calculation process is known as *transforms*.

Some transforms are obvious. When our car in the previous example moves along the road, it's a really obvious example of change. But there's more than one type of transform. In our virtual kitchen, we may stick our virtual "head" out the window. In this case, the viewpoint is changing – the only object that's moving is the one we can't see – but we can see the change that occurs.

Simply moving an object is one type of transform...

Movement is known as *translation*.

Another way to look at it is to think of the zoom lens on a camcorder. When you zoom into a scene – kids playing in a sandbox – then the picture in your viewfinder changes. When you zoom in, the child's face suddenly seems to grow huge.

Moving our point of view is another type of transform.

Here, our virtual "camera" zooms in on part of the car.

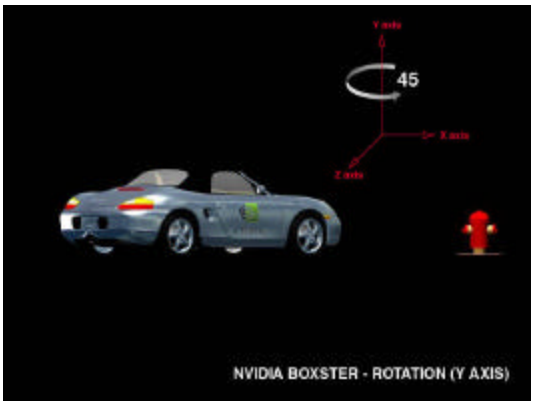
If you were trying to simulate this on a computer, each frame would show more and more of the objects being zoomed into, but surrounding objects would leave the scene. So each frame is different from the one before it. On top of that, you want this virtual object to look bigger, just like the child's head in the viewfinder. (When an object size is changed, it's called *scaling*.)

3D graphics transforms allow us to do things that can't be done in real life...

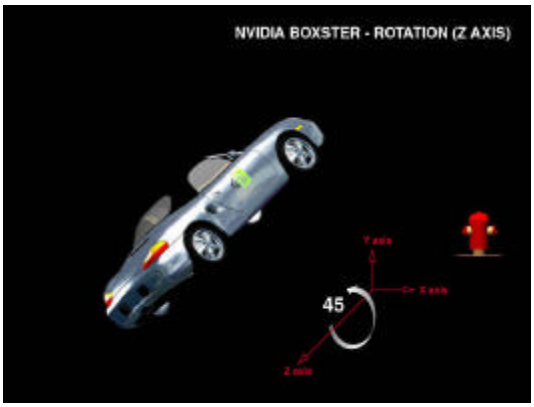
Such as make a car much larger...

Or exercise our whimsy by making the fire hydrant bigger. Changing an object's size is called *scaling*.

Another type of transform is rotation. If you take a car and rotate it 90 degrees, that's a transform. It's obvious, then, that in a computer animation, many changes are occurring from one frame to the next, from objects moving, to our viewpoint changing, to the size of objects changing as they move toward or away from our viewpoint.

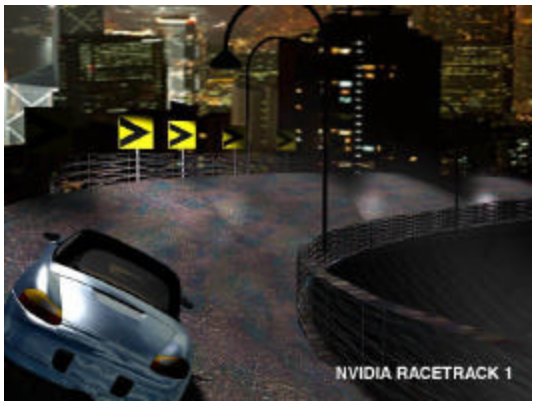


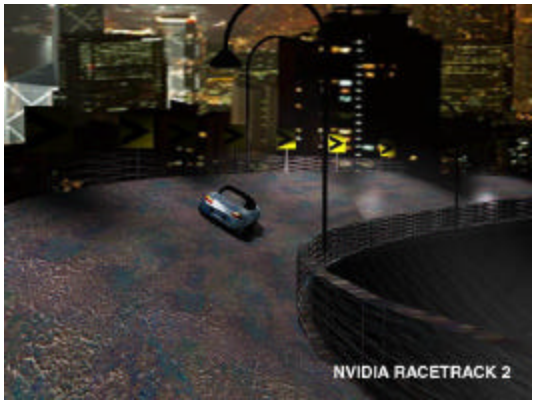
Rotation is another type of transform.

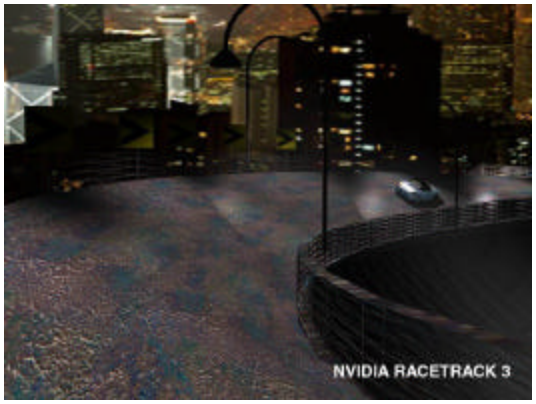


We can rotate through several different axes; these are calculated independently.

It turns out that the mathematics needed to handle transforms is relatively straightforward – but there's a *lot* of this type of calculation that must happen to create a scene. Remember, we're trying to simulate a realistic environment on our computer screen. In the real world, you can move left, right, forward, backwards and, in some cases, up and down. When a ballet dancer pirouettes, she rotates, and parts of her body are moving forward, backward, left and right simultaneously. If we were trying to simulate a ballet dancer on the computer, we'd have to keep track of all those directions (called *axes*) simultaneously.



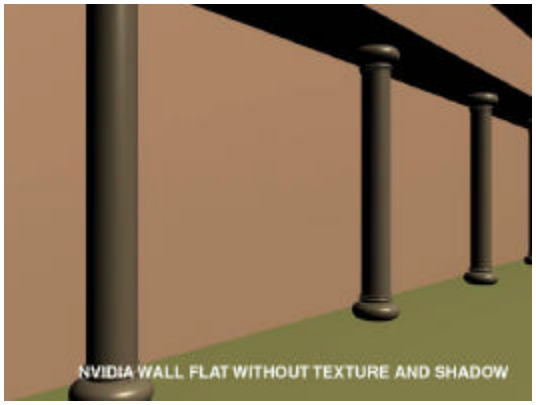




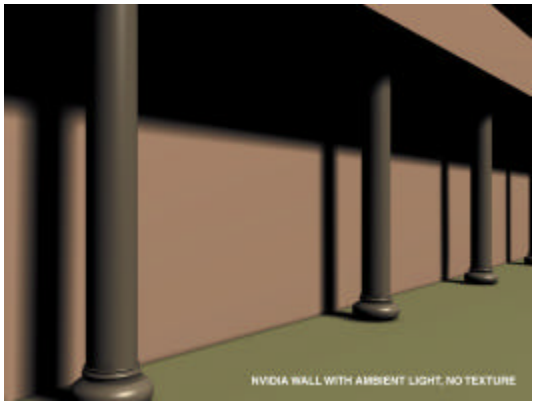
We can combine all the different transforms – rotation, scaling, and translation – into one fluid scene.

Let There Be Light

The task of creating the illusion of lights, whether from the sun, indoor lighting, spotlights or other lights is the next step in our assembly line. After all, you need light to be able to see what happens. Creating synthetic lights to brighten up a computer image is really a complex task involving an understanding of how light is emitted from a light source – say a light bulb – and spreads out over the area, illuminating various objects. Not surprisingly, this step is known as *lighting*.



Light is such a part of our existence, that the absence of light fools the eye. The supporting columns are not really part of the wall.

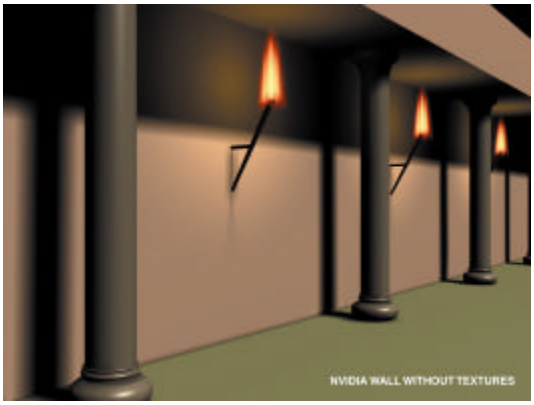


Adding a single light source allows us to see the columns as discrete objects.

If you look around you, you're likely to see a number of light sources. The sun streams in through a window. There's a lamp on your desk. Your computer display glows with its own light. There are several fluorescent light tubes glowing.

On top of that, light is reflected from various surfaces in the room. Shiny surfaces, such as mirrors or brass doorknobs are obviously reflecting light, but even dark or dull colored objects reflect some light.

Accurately simulating lights in a 3D graphics program also takes a huge amount of math "under the hood." Because light bounces willy-nilly off hundreds of surfaces in a typical room, trying to keep track of the path of a light beam as it bounces, refracts into multiple beams and eventually becomes absorbed takes a massive number of complex calculations. If the graphics program wants to create multiple light sources, the challenge becomes greater. For example, if you shine a flashlight on a brass doorknob, how much of the light you see is reflected from the flashlight, and how much from the fluorescent lights on the ceiling?



Adding a few realistic light sources makes our wall look much more realistic.

Many computer games “cheat” when it comes to lighting. Instead of actually tracing the path of light beams, games often use “light maps.” These are simply bitmaps – pictures stored on the computer – that fake the eye into thinking there are lights present.

Look at it this way. If you stare at a photograph of a brightly lit room, there are no real lights shining – it's a picture. Brightly lit areas are simply lighter colored. In a similar vein, light maps simply change the color of the object, making it appear as though a light is shining on it. Graphics programmers would like their scenes to do a better job of using realistic lighting. But that takes some serious computer horsepower – it takes a CPU 1-2 hours to calculate one frame or a realistically animated movie scene.

Render Unto Caesar

It's a Setup

The next step is taking the information created and converting it to something the computer display can begin to understand. It begins with breaking each object into component parts. Most 3D graphics programs create objects out of many small triangles. If you take our basketball example, a basketball can be created with dozens or hundreds of small triangles. Each triangle consists of edges and corners. The graphics card is sent the information on these edges and corners (the corner of a triangle – one point – is called a *vertex*). This data must be set up into pixels because the computer monitor doesn't understand triangles. The process for going from vertex information to pixel information is called *setup and clipping*. The part of the 3D graphics chip that handles this is the *setup engine*. (You may also see this called *triangle setup and clipping*.)

Each individual triangle is examined by the setup engine and broken up into individual components suitable for the next step.

At this point, we've completed the task of setting up the geometry of the 3D environment to information suitable for the rendering engine.

The Rendering Step

Rendering is the process of displaying the image with the right visual characteristics (colors and shading) is known as *rendering*.

Computer screens are flat, 2D surfaces. They display images by using thousands of tiny dots (*pixels*) to create an image. Take a look at a photograph from a newspaper. From a distance, it looks like a picture. If you hold it up to your nose, you'll see that the picture really consists of many small dots. If you get close enough to squint at a color newspaper picture, you find out that it's made up of many small, colored dots. In fact, that's exactly what a computer image is: many small dots that, from a distance, make up a picture. The greater the density of dots, the higher in quality the resulting image will be.

Converting the geometry of a true 3D world to an image that can be displayed on our pixilated computer screen is yet another serious computational challenge.

At this point in our 3D pipeline, we have a lot of information that has now been converted to scan line data for the computer monitor, along with some information as to the brightness of the pixels. Now we need to take care of the actual colors in the scene. It's at this step that the correct colors are created for the screen and the direction and intensity of the lighting is converted to various intensities of color.

In addition, modern 3D graphics makes heavy use of *texture maps*, which are simply pictures applied to the object to make it look more realistic. A picture stored in a computer is typically stored as a bitmap. A bitmap is simply a computer version of our newspaper picture. If you've ever scanned a picture into a computer, you may realize that once in the system the picture is now a bunch of pixels that can be changed at will.

If we return to our basketball, a picture of a real basketball may be used to make our rough object look more realistic. Another common example is a road surface. A rectangle has a bitmap of gray asphalt with two yellow lines applied to it.

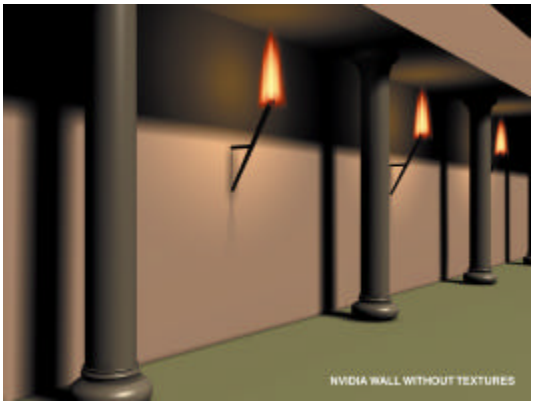
But it's not quite that simple. Remember, we're trying to represent a real, 3D world on a flat, 2D screen.

If you look out your window again and see a road, you'll see that the road tapers off in the distance. The two yellow lines begin to blur together. You're witnessing the effect of perspective. If our computer road was just a rectangle, it wouldn't look at all realistic. So the computer tapers the rectangle, adding perspective to the image, to make it look more real. Now we can apply our texture map (picture) of the road surface. But wait – the two yellow lines seem to go on forever. We need to add perspective to the texture map as well, so the road looks correct. This is called, unsurprisingly, *perspective correction*.

Another step that occurs is *filtering*. The problem with texture maps is that they are fixed sizes, and consist of a fixed number of pixels. Let's say we apply a picture of a stone wall to a large rectangle. Now we have something that looks like a wall. Now let's move our virtual head so that it's six inches from the wall. What you see are huge, square blobs. Each of these blobs represents one pixel of the texture. But each square blob is actually made up of many

screen pixels. If you move rapidly away from the wall, you may see the edge of the wall or parts of the texture "swim" in a distracting manner. This is called *aliasing*.

A filter is simply a bit of math which, when applied to a bitmap, blends together colors of adjacent pixels on the screen such a way as to remove the sharp edges created as we move closer to the wall or other surface. Other types of filtering take care of making the image look good as we mover further away. There are a host of filtering techniques that come into play.



Here's our wall again. Even with some lighting, it looks, well, flat.

Adding some texture mapping and several more light sources makes it look better.

As we pull back, we can see the effect of more texture mapping, as well as perspective correction and filtering.

3D for the Real World

Yesterday's Hardware

Despite the rapid changes in 3D graphics hardware, today's traditional processors are still limited. To understand why, let's return to our 3D graphics assembly line. If you recall, the four stages are transform (what changes from frame to frame), lighting, setup and clipping, and rendering. The problem is that today's traditional 3D processors only handle those last two steps in the 3D pipeline: setup and rendering.

Game developers – and other content creators – often think in terms of CPU budgets. In the era before today's GPU, up to 80% of the computer's horsepower would be used for rendering. So naturally, rendering was one of the first tasks that was offloaded from the CPU to the graphics card. Traditional 3D graphics processors simply calculate the math necessary for setup and rendering.

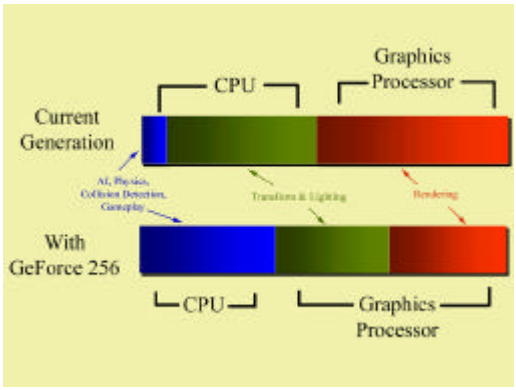
As time went by, games became more demanding. Users wanted higher resolutions, more colors, more detail and better performance. So the 3D chips got faster and added more rendering features. But today, the rendering engines are so fast that they are often waiting for the CPU to complete its tasks.

As users demanded more realism and better animation, the next logical step was to build hardware that could accelerate the transform and lighting – the tasks the CPU was struggling with – to create a complete GPU.

Today's Hardware

It was inevitable that the next generation of graphics accelerators – GPUs – would take on the chore of transform and lighting. The mathematics is straightforward – the only cost is in designing the chip to take on greater functionality. NVIDIA's GeForce 256 GPU has in excess of 22 million transistors. If you contrast that with today's Pentium III processor, which has only 9 million transistors, you can see that it is possible to create extremely powerful processors capable of completely handling the graphics processing with extreme speed.

Since the GeForce 256 GPU handles the transform and lighting, the CPU can concentrate on other tasks.



The GeForce 256 offloads transform and lighting from the host CPU, allowing the CPU to process more game play elements.

In games, there's a lot left for the CPU to do. There's an insatiable demand for more realistic physics in games. Flight simulators now calculate how the air flows over the wing, rather than using a discrete table of information on how a plane should behave. The number of objects in scenes is increasing as well. For example, *Falcon 4.0* is a simulation of the Air Force's F-16 jet fighter. In the game, an entire ground war is simulated in real time. But if you look closely at the ground war, you'll see tanks passing through each other and very little infantry. The task of deciding when a virtual object would bump into another one is called "collision detection." But with *Falcon 4.0*, which has some of the best graphics in a flight simulator, there just wasn't enough CPU horsepower to perform collision detection on hundreds of vehicles, worry about the air war *and* handle transform and lighting.

Another issue is artificial intelligence (AI). Despite the popularity of Internet gaming, most games are still played by one person, with the opponent chores handled by the PC. Programming a computer to be a competent opponent is a tough challenge, but with more CPU time available, better artificial opponents can be created.

With GeForce 256, the visual appearance of real-time, 3D computer graphics reaches an unprecedented level

Offloading graphics from the CPU to the GPU leaves more CPU cycles for physics, collision detection and AI, which makes for a richer gaming experience.

Today's Applications

In the past three years, the hardware in your PC that handles the graphics chores went through a revolution. Three years ago, finding a graphics adapter that could accelerate any part of our four-step 3D pipeline wasn't easy. Today, you cannot buy a computer that doesn't have some flavor of 3D graphics hardware already installed.

It was inevitable that game designers would adopt the new generations of 3D processors. Simulators such as flying and driving games, and action games like *Quake*, all try to create a virtual world that allows the user to move around in it at will in any direction – unlike older-generation games.

However, other types of applications are taking advantage of 3D.

Imagine, for example, real estate shopping on the web. You'll be able to "walk through" houses that are still being built in real time yet look complete on your screen. You could look out the window and check out the view. You could even get measurements of rooms to figure out furniture arrangement.

Another possible application is business visualization. One of the hot trends in business *data mining* which involves huge repositories of data. The traditional 2D spreadsheet – or even a relational database – can't present the data in an easy-to-digest form. But the data can be converted to a 3D format in order to visually represent potential relationships.

Future user interfaces may take advantage of 3D horsepower. Xerox and Silicon Graphics pioneered 3D user interfaces for exotic workstations in the 1980's, but that capability can now reside on any user's desktop. Even the World Wide Web can be presented as a "galaxy" of interrelated links. All of this is made possible by the advances in 3D graphics hardware.