Walk this way

Benjamin Woolley paves the way for character animation. Making your figure take its first few steps can be tricky, but with the aid of 3D graphics tools you shouldn't trip up too much.

This month I want to take a nervous step into the minefield of character animation, which brings together some of the concepts covered in my two previous columns.

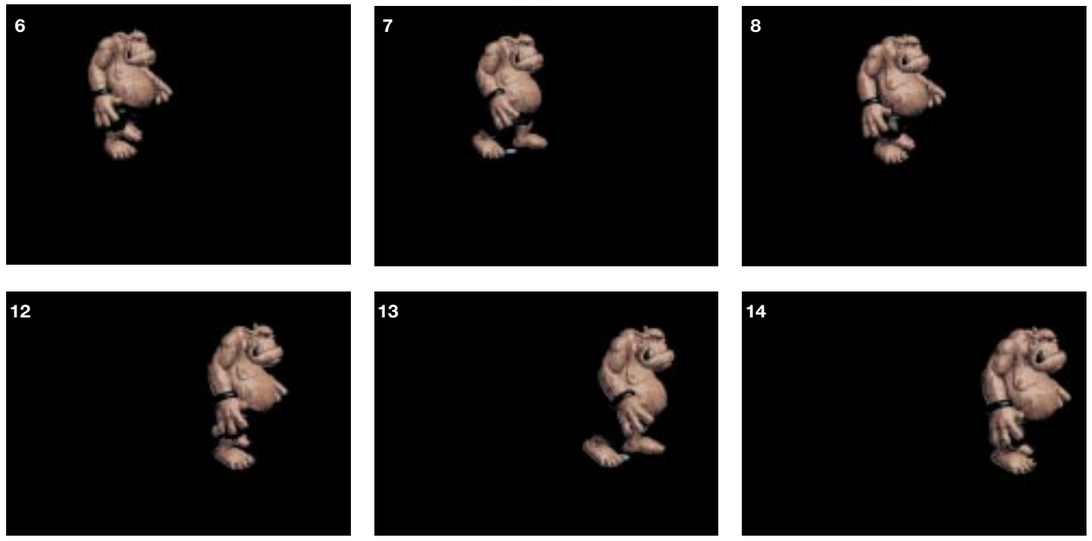
Character animation is difficult; perhaps the most challenging aspect of computer graphics. A spaceship zooming through the star-spangled firmament, or a car driving down a canyon of towerblocks: these are, in animation terms, easy to create, as the objects are moving at even speeds along straight lines and around curves. Compared with this, getting a character to walk is an impossibly difficult task.

Very few people are capable of *really* being able to animate, to bring things to life, and nearly all of them are employed by Disney. The computer has, at least until recently, afforded the amateur little help. Even with the most sophisticated system to aid you, there is no substitute for artistry. Nevertheless, with a little ingenuity and a lot of effort, decent results can be achieved.

Master plan

First, the good news. Animation is all about simplicity and barely at all about realism. You want proof? Consider the Master, John Lasseter, who created Toy Story. His first animation, Luxo Jnr, which was released in 1986 and received an Oscar nomination, was one of the first computer animations to attract widespread attention (you can find out more about Luxo and Lasseter at the Pixar web site at www.pixar.com).

Luxo Jnr starred a pair of anglepoise lamps (Luxo is the American trade name for



this familiar article of office furniture) playing with a ball. Despite the fact that these inanimate "characters" did not even have faces, Lasseter gave them the power to express emotions that would have stretched the talents of a member of the Royal Shakespeare Company.

So keep it simple. Observe how nearly all popular animated characters are outrageously simplified: Mickey Mouse has only three fingers and no knees; neither Tom nor Jerry have anything like fur.

Some games are now emerging, featuring human characters that succeed in moving in realistic ways. But remember, these characters were animated using motion tracking and you will not be able to reproduce the same sorts of dynamics unless you have access to this type of technology.

Motion tracking relies on real humans performing various movements with sensors (usually magnetic or luminescent) attached to key parts of their bodies (the joints, hands, feet and so on). A motion capture system monitors the movement of these sensors in real time and logs the position of each as a series of co-ordinates that can then be read into an animation program. To get a better idea of how this works, look at the web site of Polhemus <www.polhemus.com>, one of the leading motion capture hardware manufacturers.

Without the benefit of motion capture you have to use your own imagination to envisage how a character you want to create will move. The only sure way of doing this successfully is to first sketch out ("storyboard") your character on paper.

Textures

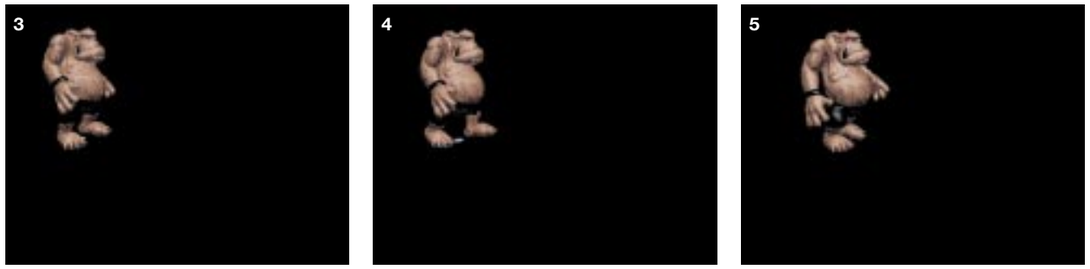
Once you have an idea of what your character will look like and how it will move, you can proceed with the task of modelling and animating it. Before you begin, though, you have one more important decision to make: how much of the animation you want to achieve through the model itself, and how much through the textures you use to clothe it.

You can achieve a great deal just by creating clever 2D textures. For example, you can create a texture map that represents the character's entire exterior: its facial features, skin, wrinkles, fingernails and clothes (if it is wearing any). This map will not be "tiled" repeatedly across the surface; it will be (in the jargon) a "decal" map, exactly tailored to fit over the model's geometry, almost like the character's hide (the reverse of what you would get if, for instance, you skinned Mickey Mouse).

If you get this 2D texture map right (it could be more than one map, perhaps with a separate one for the face) you may be able to reproduce some of the animation effects you want without having to resort to complex modelling procedures. For instance, you could create the character's facial expressions by turning the texture for the face into a 2D animation that shows the eyes blinking and the mouth smiling.

Do the bump

You can also use bump mapping to give the texture a three-dimensional feel; one that itself is animated. For example, to simulate wrinkles in a creature's skin you could create a greyscale version of the texture with stripes of black and white placed at strategic points. When you apply this bump



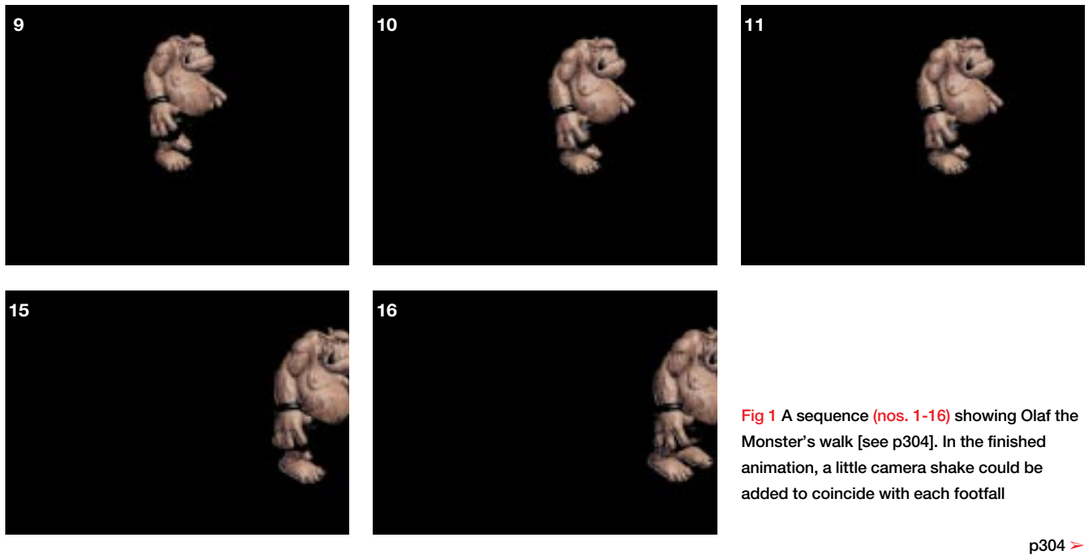


Fig 1 A sequence (nos. 1-16) showing Olaf the Monster's walk [see p304]. In the finished animation, a little camera shake could be added to coincide with each footfall

p304 ➢

Fig 2 **Olaf the Monster, a Character Studio biped, goes through his paces**

map to your creature and animate the model, you should find, if your map is an accurate one, that folds will appear exactly where they should. This will happen because the map will bunch up or stretch out as it adapts to the new shapes of the animated object to which it is applied.

Manipulation and morphing

Texture maps can get you so far, but at some point you will certainly have to manipulate the object you are intending to animate. There are two main ways of doing this. One is to create different versions of your object, representing the key states it will be in, and then morph between them. The other way is to manipulate the object into a new configuration for each key frame in the animation. The former method is the one to use when the software you have (for example, early releases of 3D Studio) does not allow you to animate an object's shape. Thankfully, most modern packages now provide the facilities for animating any parameter that can be edited, which makes morphing unnecessary.

Morphing works by changing one shape into another over time. You can only morph objects with the same number of vertices, which means creating a number of copies of the same basic model and manipulating each into a new shape or position.

Direct manipulation of the model means moving to each key frame in the animation module of the program (or with the animation button set to "on", in the new generation of modeless packages) and using standard modelling tools to make the appropriate changes to the geometry.

To use the latter method successfully, you will almost certainly need to split your character up into a number of sub-objects which you must then link together in a clearly organised hierarchy (see last month's column). You will also have to make sure all the pivot points are in the correct place for each sub-object. All this must be carefully worked out, in advance, because once you start animating, it is virtually impossible to change such parameters. If your software supports inverse kinematics, then this is when you will learn its advantages.

Our two-legged friends

For the first time, products are beginning to appear that allow the computer to take some of the legwork out of animation. Two products in particular promise to ease the job considerably, one being the new version of Poser from Fractal (or rather, MetaCreations, as the company is now called after its merger with Metatools), the other Character Studio from Kinetix. I have not yet tried Poser 2, but have been spending some time with Character Studio, a pricey but extremely powerful plug-in for 3D Studio MAX. The features it offers will, without doubt, become available on cheaper products in the not-too-distant future.

Character Studio comprises two modules, Biped and Physique. The latter is for creating muscle-like contours and I will deal with it in more detail in a future column. Biped is for creating and animating two-legged characters (it does not yet include features for our four-legged friends). These characters have certain simulated physical characteristics (such as gravitational dynamics, meaning they can move as though feeling the effect of gravity) and bipedal mechanics built into them (limbs and joints can't be moved beyond what would be physically possible), so as you manipulate them, they behave as real characters would.

One of the cleverest features of Character Studio is that it enables you to move a character around a scene simply by placing footsteps in the required positions (Fig 2); Character Studio then works out all the necessary animation.

Furthermore, by adjusting the spacing between the steps and changing the pose of the body (for example, by making the hip bone rock from side to side with each step), you can achieve just about any gait you want, from a march to a mince. You can save these as special motion files which can then be applied to other bipedal objects.

Character Studio (and similar products) will not transform rank amateurs into John Lasseters overnight, but it does demonstrate how, with a bit of added intelligence, 3D graphics tools can help even the most inexperienced enthusiast manage those first few steps into the exciting world of character animation.

PCW Contact

Benjamin Woolley, writer and broadcaster, can be contacted at 3d@pcw.co.uk.

Hey, get hip!

Two essential parts of animation are hierarchy and inverse kinetics. Benjamin Woolley uses an old song about hip bones and thigh bones and things, to give the low-down on linking.

That well-known song about the hip bone being connected to the thigh bone, and the thigh bone being connected to the knee bone, may be anatomically dodgy, but it provides an easy way of thinking about two, sometimes difficult but essential areas of animation: hierarchy and inverse kinematics.

Hierarchy is about linking objects in a scene and determining how they interact. It may seem like a relatively minor aspect of 3D graphics, but it is the key to successful animation. Without it, all you can really hope to do is work with simple objects changing in simple ways.

Dem bones, dem bones…

The hip bone/thigh bone analogy is a good example of a hierarchy of linked objects. If you were trying to animate part of a human skeleton, you might have a series of separate objects representing each major bone. If you wanted to make this skeleton walk, you could do it by linking these bones together. For example, you might link the thigh bone (or rather, bones, one for each leg) to the hip bone.

Such a link is not one of equals. One object will always be the parent, the other the child. Which is which is the decision of the animator and in this case you would normally want to make the hip the parent and the thigh bone the child. Similarly, you would want to make the shin bone the child of the thigh bone — thus creating the first two branches of a family tree with the hip as the root. It is because of this genealogical approach that a series of linked objects are generally known as a hierarchy.

The difference between a parent and a child is simple (and revealing of the 3D graphics application designer's view of parenthood): the child has to do whatever the parent does, but the parent does not have to do whatever its children do. Also, a parent can have many children, but a child cannot have more than one parent. So, taking the example of the leg, if you change the hip in any way (for example, move it) all the other bones follow. If you change the thigh, all its children (the shin, the foot and toes) follow, but the hip does not.

A parent/child relationship is a one-way conduit for information which you can think of as joining the two objects' "pivots"; the point around which each revolves. A pivot need not necessarily be at the object's centre: for example, with a thigh bone it might be at the top, where it meets the hip. The information sent is generally about the change or "transform" applied to the parent; typically, a change to its size, position or orientation.

However, this does not necessarily mean that if the parent remains untransformed, its child is otherwise free to behave in any way it wishes. You can limit a child's "degrees of freedom" by applying what are called "constraints": for example, constraining a thigh bone to rotate no more than about 120 degrees forwards or backwards and, say, 90 degrees, side to side.

Take these chains

This parent/child linking scheme passes on transform information by means of what is known as "forward kinematics" (FK). This works well in many circumstances but not particularly when you are trying to animate jointed structures like skeletons, where it is often the object at the bottom rather than the top of the hierarchy that you want to position first (say, the hand when wanting to animate it picking something up). You want to move the hand, and know that the arm will follow accordingly.

To achieve this, you use another linking scheme called inverse kinematics (IK). Until recently, IK was only available on the most expensive animation products. Thankfully, it has now migrated down to even the cheapest 3D applications.

Chain gang

With IK, the hierarchy becomes a kinematic chain (Figs 1 to 3). When you move a child at the end of a hierarchy (sometimes known, in a confusing mixed metaphor, as a "leaf", because it is at the end of the outer twig of the family tree) the IK engine calculates the movement that its parents, grandparents and great-grandparents and so on, must make to keep the chain together. If you move an object that is in the middle of the hierarchy, then any object further down the hierarchy (that object's children, grandchildren etc.) will be subject to forward kinematics, while any objects further up (its parents, grandparents and so on) will move according to inverse kinematics.

Fig 1 (left) **A chain of linked objects, with the rod as the root, the first link being its child, the second its grandchild, and so on down to the end of the chain**

Fig 2 (below) **The result of moving the unconstrained "leaf" object in the hierarchy**

Fig 3 (below right) **The same move, but with inverse kinematics enabled**

The key to making IK work is constraining the links or joints so that they behave properly. Most packages have a variety of preset link types that make this easier. For example, Ray Dream Animator, one of the cheapest but better-specified packages, has options for ball joints, constrained rotating joints, joints that can slide in and out, and a shaft joint (which allows an object to rotate freely around as well as slide up and down an axis, but not move away or towards the axis; think of a fire-fighter sliding down a pole).

At the other end of the PC price scale, 3D Studio MAX offers just three types of joint: rotational, sliding and path. There is, however, a welter of parameters that can be adjusted for each. The "path" joint is particularly useful: it allows an object to rotate or move around an animation path, as a key, for instance, might move or rotate around its ring.

When creating an animation, you have to spend a lot of time setting up your hierarchies and constraints. It may be laborious, but it will be time well spent. It will mean that your objects do not end up in a hopeless tangle, and that they behave more or less as you would expect them to throughout the entire animation.

PCW Contact

Benjamin Woolley, writer and broadcaster, can be contacted at 3d@pcw.co.uk.

MMX — the fast track?

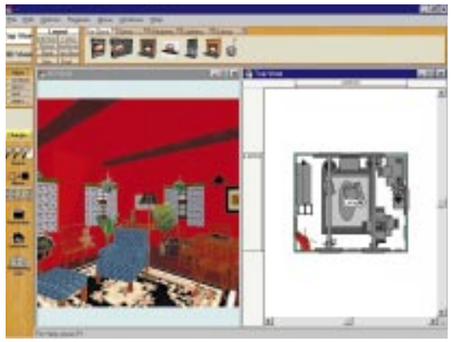
Having seen the funky ads on the television and read all the hype — and having talked to all the users who got new Pentium systems for Christmas and felt cheated when the MMX systems came out at the beginning of the year — I decided to have a go at finding out what real difference MMX would make to 3D graphics.

MMX should offer some advantage to any software because it has double the on-chip cache (from 16K to 32K) and, so we are told, a more efficient "branch-prediction" technology originally developed for the Pentium Pro.

However, there is also a claimed premium that comes from MMX-enabled software which uses the 57 new instructions that Intel has added to the basic set, which themselves exploit the chip's floating point register. It was this premium I wanted to test.

Two items arrived in my office that enabled me to achieve some sort of benchmark: a fast 200MHz MMX system in the form of the Aurora PowerMedia MMX from Northwood, and one of the first MMX-enabled 3D applications to reach the market — Visual Home Deluxe from FastTrak (*pictured, above right*).

Visual Home is an interesting example of that slowly emerging breed of 3D software aimed at the domestic market; literally so in this case, since it is concerned with DIY and house design. You can use it to plan the layout of rooms, floors (even whole houses) and to cost the building work. You can also use it as an interior design aid, for moving furniture around, changing the wallpaper or trying out pictures. After you have done all this, you can "walk through" the vision of your future home and see what it would be like.



Visual Home performed as well with MMX as without it

For the purposes of my experiment, I exploited the fact that Visual Home comes with executable files in both MMX and non-MMX versions on the distribution CD. I could install either simply by swapping the appropriate file in the installation folder; nothing else changed.

I did not have the tools to take scientific measurements of the rendering speed and other operations, but I did time the basic manoeuvres such as the time taken to edit geometry, add textures, and walk around a scene.

And the result? Absolutely no detectable difference. Visual Home (which uses a new version of the RenderWare real-time render engine, according to FastTrak) ran respectably fast — just as fast in MMX mode as in non-MMX mode. Why? Perhaps it is something to do with the way the floating point register is used in MMX mode. Whatever the reason, at least those of us who have yet to leap aboard the MMX bandwagon need not feel so left out.

Hey, get hip!

Two essential parts of animation are hierarchy and inverse kinetics. Benjamin Woolley uses an old song about hip bones and thigh bones and things, to give the low-down on linking.

That well-known song about the hip bone being connected to the thigh bone, and the thigh bone being connected to the knee bone, may be anatomically dodgy, but it provides an easy way of thinking about two, sometimes difficult but essential areas of animation: hierarchy and inverse kinematics.

Hierarchy is about linking objects in a scene and determining how they interact. It may seem like a relatively minor aspect of 3D graphics, but it is the key to successful animation. Without it, all you can really hope to do is work with simple objects changing in simple ways.

Dem bones, dem bones…

The hip bone/thigh bone analogy is a good example of a hierarchy of linked objects. If you were trying to animate part of a human skeleton, you might have a series of separate objects representing each major bone. If you wanted to make this skeleton walk, you could do it by linking these bones together. For example, you might link the thigh bone (or rather, bones, one for each leg) to the hip bone.

Such a link is not one of equals. One object will always be the parent, the other the child. Which is which is the decision of the animator and in this case you would normally want to make the hip the parent and the thigh bone the child. Similarly, you would want to make the shin bone the child of the thigh bone — thus creating the first two branches of a family tree with the hip as the root. It is because of this genealogical approach that a series of linked objects are generally known as a hierarchy.

The difference between a parent and a child is simple (and revealing of the 3D graphics application designer's view of parenthood): the child has to do whatever the parent does, but the parent does not have to do whatever its children do. Also, a parent can have many children, but a child cannot have more than one parent. So, taking the example of the leg, if you change the hip in any way (for example, move it) all the other bones follow. If you change the thigh, all its children (the shin, the foot and toes) follow, but the hip does not.

A parent/child relationship is a one-way conduit for information which you can think of as joining the two objects' "pivots"; the point around which each revolves. A pivot need not necessarily be at the object's centre: for example, with a thigh bone it might be at the top, where it meets the hip. The information sent is generally about the change or "transform" applied to the parent; typically, a change to its size, position or orientation.

However, this does not necessarily mean that if the parent remains untransformed, its child is otherwise free to behave in any way it wishes. You can limit a child's "degrees of freedom" by applying what are called "constraints": for example, constraining a thigh bone to rotate no more than about 120 degrees forwards or backwards and, say, 90 degrees, side to side.

Take these chains

This parent/child linking scheme passes on transform information by means of what is known as "forward kinematics" (FK). This works well in many circumstances but not particularly when you are trying to animate jointed structures like skeletons, where it is often the object at the bottom rather than the top of the hierarchy that you want to position first (say, the hand when wanting to animate it picking something up). You want to move the hand, and know that the arm will follow accordingly.

To achieve this, you use another linking scheme called inverse kinematics (IK). Until recently, IK was only available on the most expensive animation products. Thankfully, it has now migrated down to even the cheapest 3D applications.

Chain gang

With IK, the hierarchy becomes a kinematic chain (Figs 1 to 3). When you move a child at the end of a hierarchy (sometimes known, in a confusing mixed metaphor, as a "leaf", because it is at the end of the outer twig of the family tree) the IK engine calculates the movement that its parents, grandparents and great-grandparents and so on, must make to keep the chain together. If you move an object that is in the middle of the hierarchy, then any object further down the hierarchy (that object's children, grandchildren etc.) will be subject to forward kinematics, while any objects further up (its parents, grandparents and so on) will move according to inverse kinematics.

Fig 1 (left) **A chain of linked objects, with the rod as the root, the first link being its child, the second its grandchild, and so on down to the end of the chain**

Fig 2 (below) **The result of moving the unconstrained "leaf" object in the hierarchy**

Fig 3 (below right) **The same move, but with inverse kinematics enabled**

The key to making IK work is constraining the links or joints so that they behave properly. Most packages have a variety of preset link types that make this easier. For example, Ray Dream Animator, one of the cheapest but better-specified packages, has options for ball joints, constrained rotating joints, joints that can slide in and out, and a shaft joint (which allows an object to rotate freely around as well as slide up and down an axis, but not move away or towards the axis; think of a fire-fighter sliding down a pole).

At the other end of the PC price scale, 3D Studio MAX offers just three types of joint: rotational, sliding and path. There is, however, a welter of parameters that can be adjusted for each. The "path" joint is particularly useful: it allows an object to rotate or move around an animation path, as a key, for instance, might move or rotate around its ring.

When creating an animation, you have to spend a lot of time setting up your hierarchies and constraints. It may be laborious, but it will be time well spent. It will mean that your objects do not end up in a hopeless tangle, and that they behave more or less as you would expect them to throughout the entire animation.

PCW Contact

Benjamin Woolley, writer and broadcaster, can be contacted at 3d@pcw.co.uk.

MMX — the fast track?

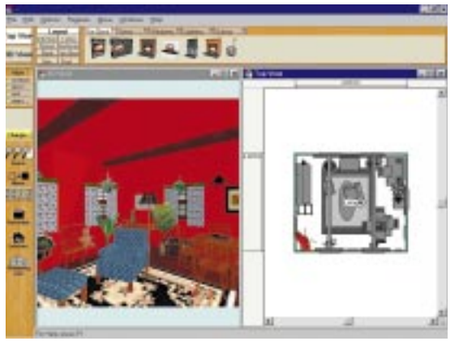
Having seen the funky ads on the television and read all the hype — and having talked to all the users who got new Pentium systems for Christmas and felt cheated when the MMX systems came out at the beginning of the year — I decided to have a go at finding out what real difference MMX would make to 3D graphics.

MMX should offer some advantage to any software because it has double the on-chip cache (from 16K to 32K) and, so we are told, a more efficient "branch-prediction" technology originally developed for the Pentium Pro.

However, there is also a claimed premium that comes from MMX-enabled software which uses the 57 new instructions that Intel has added to the basic set, which themselves exploit the chip's floating point register. It was this premium I wanted to test.

Two items arrived in my office that enabled me to achieve some sort of benchmark: a fast 200MHz MMX system in the form of the Aurora PowerMedia MMX from Northwood, and one of the first MMX-enabled 3D applications to reach the market — Visual Home Deluxe from FastTrak (*pictured, above right*).

Visual Home is an interesting example of that slowly emerging breed of 3D software aimed at the domestic market; literally so in this case, since it is concerned with DIY and house design. You can use it to plan the layout of rooms, floors (even whole houses) and to cost the building work. You can also use it as an interior design aid, for moving furniture around, changing the wallpaper or trying out pictures. After you have done all this, you can "walk through" the vision of your future home and see what it would be like.



Visual Home performed as well with MMX as without it

For the purposes of my experiment, I exploited the fact that Visual Home comes with executable files in both MMX and non-MMX versions on the distribution CD. I could install either simply by swapping the appropriate file in the installation folder; nothing else changed.

I did not have the tools to take scientific measurements of the rendering speed and other operations, but I did time the basic manoeuvres such as the time taken to edit geometry, add textures, and walk around a scene.

And the result? Absolutely no detectable difference. Visual Home (which uses a new version of the RenderWare real-time render engine, according to FastTrak) ran respectably fast — just as fast in MMX mode as in non-MMX mode. Why? Perhaps it is something to do with the way the floating point register is used in MMX mode. Whatever the reason, at least those of us who have yet to leap aboard the MMX bandwagon need not feel so left out.

All the **right moves**

Benjamin Woolley gets animated about moving images and makes a start by explaining the basics of modern animation concepts. More next month. And, enter The Cave of Madness.

So far, animation has not featured prominently in this column. I could claim the reason is that there have been more important things to write about. But to be honest, it is because I find animation difficult. It often draws on skills which are very different to those needed to create and texture objects and scenes.

Nevertheless, with software design improvements and facilities becoming more sophisticated, animation can no longer be ignored. So, in this and next month's column, I will grapple with some of the basic concepts of animation in the hope that it may encourage more of us to venture into this most enticing area of computer graphics.

Tween-age idols

In conventional animation, a cartoon is created by the master animator drawing up a series of frames which mark the most significant moments in the action. These are known as the "key" frames. An army of less exalted "hack painters" then have the job of creating the frames that fall in between the key frames. These are usually known as "tween" frames (as in be*tween*).

For example, the key frames may show a character starting to walk on one side of the picture and coming to a halt at the other. The tween frames comprise all the legwork (literally) required to join these two keys: the number of tweens determining the duration of the sequence and the smoothness of the action — the more frames, the longer the duration and the smoother the action.

The Cave of Madness

I cannot honestly write that I have found exploring VRML worlds all that much fun. They take an age to download and moving through them is like trying to swim through glue.

However, courtesy of software house IDS, we now have The Cave of Madness <www.ids-net.com>, an on-line adventure game created by Matt Costell, author of The 7th Guest.

To enter the Cave of Madness you need Netscape Navigator 3.01 and version 1 beta release 3a of Silicon Graphics' Cosmo player. You also need quite current hardware. My old Compaq Deskpro simply wasn't up to this sort of job so I borrowed a rather sleek 200MHz MMX multimedia Pentium box from Northwood, a new player on the PC market. I have to say the result (*illustrated*) shows that VRML 2.0 has, as they say in Hollywood, legs. The animation of the view-point, as I navigated through the world, and of the objects furnishing that world, was extremely smooth. There was no detectable download lag ("latency"), even though I was using a 28.8Kbit connection.

Using the same Northwood/Netscape/Cosmo Player combo, I also tried out a 3D cartoon on the Silicon Graphics VRML website <vrml.sgi.com>. It worked a treat, particularly the sound, which really appeared to shift position as I navigated around the scene. I would heartily recommend anyone who has lost touch with VRML and who has got the kit to give it another go by trying out these sites. It really is becoming another world.

The Cave of Madness, integrating a VRML window (*top right*) with HTML and Java

p262 ➢

With a 3D computer graphics package you are the master animator, and the computer, the army of hack painters. However, there is no magic to the computer's tweening abilities. It can basically only work out the tweens when the keys involve a change in a mathematical value (for example, an object's position, size or orientation).

Unlike a human tween artist, standard animation software is unable to bestow behaviour patterns on objects. For instance, it does not know that if you move a bipedal character from one point to another, it usually walks.

First steps

With most programs now being modeless (that is, having just one interface for all aspects of the design process), animation has become an extension of scene editing. For example, suppose you have created a scene with two globes, one smaller than the other. Suppose you then decide to move the small globe from one side of the larger globe, to the other side. To do this you would usually point at the smaller globe and, holding down the mouse key, drag it across the scene.

Now, suppose that you wanted to animate this move. You would set a length for the animation (usually expressed in frames; let us say 100), shift a slider to the final frame, set an animation button to "on" and then perform exactly the same move operation.

This would automatically create two new keys: the first representing the scene in its original state, the second representing the smaller globe in its new position. The computer would then divide the amount of movement by the number of frames, so, in this case, each frame would show the smaller globe moved by 1/100th of the total distance. Since, by convention, there are 25 or 30 frames per second (the former being the European standard for video, the latter the US standard), the total animation would be four seconds long.

The clever thing about computer animation is that if you edit any object or parameter (with a few exceptions) in any frame other than the first, you create a new key and the software will automatically work out the tweens.

To return to the above example, you might go to frame 100 and rotate the smaller globe 360 degrees around an axis placed in the centre of the larger globe. The result will be that the smaller globe completes an orbit of the larger one. You might go to frame 50 and distort the larger globe in some way (perhaps flatten it), then go to frame 100 and return it to its former shape — with some packages you can do this simply by copying the parameters for the object from frame 0 to frame 100. The result in this case will be that the larger globe slowly squashes and unsquashes.

Moving in rhythm, moving in style

So far, so easy. However, we now enter a new level of complication which demands some migraine-inducing brainwork but which is nevertheless essential to realistic animation.

Only occasionally do you want to apply changes evenly across an animation: usually when that animation is to be looped so it appears as a continuous, smooth movement. More usually, in a 100-frame animation, you do not want the effect to apply in exact 1/100th increments from start to finish. You may want smaller increments with which to begin and end, representing the sort of acceleration and deceleration of an effect that you generally find in real life, such as movement.

As is so often the case in 3D graphics, there is no standard term used to describe either this process of manipulating the rate and "strength" with which an animating effect is applied, nor how it is done. In Truespace, for instance, you use an "Animation Path Tool". In Extreme 3D you have "Ease" controls. In Ray Dream Animator you use "Tweeners". In 3D Studio MAX you have "function curves" and "TCB Controllers".

Ray Dream Tweener

Ray Dream's Tweener is perhaps the most elegantly designed tool. It represents the change being made between two key frames as a line on a graph, with the horizontal axis representing time and the vertical axis representing value, such as an object's position being animated. The standard shape of this line is a straight diagonal, which leads from zero to maximum. This indicates that the animated value is increased by the same increment in each frame.

By editing this line (using standard graphics tools) you can change the increments. For example, you can achieve the effect of acceleration at the beginning of the animation and deceleration towards the end, by making the Tweener a flattened-out "S" shape, which represents small increments to start with and which gradually grow in size until the mid-point of the animation, when they begin to reduce.

TCB

In other packages (Caligari's Truespace 2 and 3D Studio are two examples) a more complex system is used, particularly when it comes to animating motion. This is based on adjusting three parameters called tension, continuity and bias (known as TCB). I loathe TCB because even though I have grappled with it countless number of times, I continue to find it complicated and unintuitive.

The terminology comes from a technique for editing curved lines known as "splines". With TCB editing, the spline with which you are dealing is the "motion path" or "trajectory" (again, there is no standard term); the line that the moving object takes through the scene.

Splines (of which Bezier curves are the type normally used) are curves with at least three control points. When the curve represents a motion path, these control points are the key frames of the animation.

A control point determines the shape of the line that connects it to the next and previous point and the distribution of the tween frames along the path. You edit it by manipulating "handles"; lines which emanate from the control point at a tangent to the curve.

When working with TCB, one way to think of it is to imagine driving a car around a corner. "High tension" means you enter the corner fast, brake hard and turn sharply, "low tension" means you take the corner gradually and smoothly. "High continuity" means you enter the corner too fast and over-shoot, while "low continuity" means you tootle in at an even pace, turn, and tootle out again. "High bias" means you swerve out before the corner, and with "low bias" you swerve out after.

I hope this metaphor helps. Next month I will see if I can come up with another, to explain a yet more complex animation concept: inverse kinematics and the allied field of object hierarchy.

PCW Contact

Benjamin Woolley, writer and broadcaster, can be contacted at 3d@pcw.co.uk.

Art attack

Benjamin Woolley contrasts photorealism techniques in rendering 3D graphics, and previews Riven, a new game by the Miller brothers, which ventures into the realms of art.

I attended a lecture given by a 3D graphics pioneer at the SIGGRAPH graphics conference a few years ago, on a new form of rendering called "radiosity". He showed an image displayed on a PC sitting on a desk in an ordinary office. Not only the image on the screen was synthetic, he told his audience with pride; so was the screen itself, the desk, the office and everything. To me, it was obvious that the scene was synthetic: the human eye has the ability to spot even the most subtle clues that give the game away; it is something about the light, the composition or the perspective. The image may look photorealistic but it does not look *real*.

Radiosity is very much an engineering solution to the problem of "photorealism". In fact, it has its origins in thermal engineering and works on the principle of calculating the transfer of radiation between surfaces. In graphics terms, this means calculating the way light radiation bounces off one face and onto another. The result is "photorealistic" because radiosity can more accurately reproduce the diffusion of light present in physical environments: the phenomenon of "colour bleeding", for example, where the colour of one object bleeds over to those that surround it (picking up its reflected light).

Radiosity is beginning to reach the mainstream 3D market. Lightwork Design <www.lightwork.com> has a radiosity renderer which Kinetix will be shipping as a plug-in for 3D Studio MAX.

Intelligent fish

The fetish for photorealism represents one view of 3D graphics but an alternative, and more interesting, approach goes in the opposite direction. The surrealistically-named company, ThinkFish, was founded by MIT Media Lab's Rolf Rando with the aim of introducing a little more art to the science of 3D graphics. The means of doing this is a technology dubbed "LiveStyles", a rendering system which goes in the opposite direction to radiosity. ThinkFish calls it a Non-Photorealistic Renderer (NPR); which is, the company claims, "intelligent".

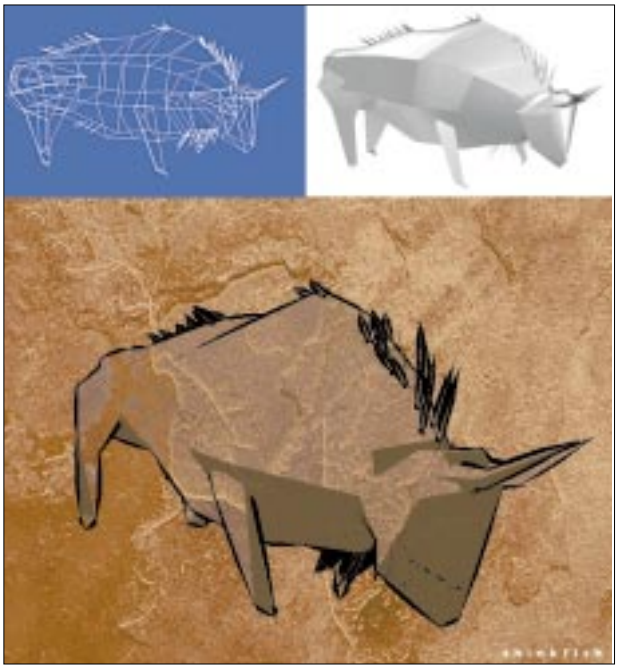


Fig 1 The ThinkFish "intelligent" renderer at work, picking out the key lines from a simple mesh

It examines a model to establish its "key lines" (Fig 1) and uses these to create the rendered image. It does this in the style of a cartoon, a pencil sketch or a paintbrush, or in any of an infinite number of different styles (the LiveStyles) which the user can select to produce the desired look.

The result is more like a 2D drawing than a 3D scene. There is no pretence at realism nor any attempt to create a feeling of depth or space. Rather, the user is encouraged to use their imagination to come up with something unique, something with the advantage of being a 3D scene which you can explore or animate.

ThinkFish's commercial strategy is to licence the technology to different graphics applications developers to include in their products and sell packages of different LiveStyles (Fig 2), "from Picasso to the Simpsons", to end-users for $30-plus.

At the time of writing, it had gained the support of Apple, which has announced a LiveStyles plug-in for applications which use QuickDraw3D (check with the excellent QuickDraw3D website at quickdraw3D.apple.com for news of availability). Fractal Design (to be renamed MetaCreations if its merger with Metatools goes through) also intends to include LiveStyles in the next release of its 3D products including Ray Dream Studio and Poser. A company called Vertigo has shipped a selection of LiveStyles with its Adobe Photoshop plug-in, Dizzy 3D (at the time of writing, only available for the PowerMac).

One of the benefits promised by the LiveStyles technology is that it will sidestep one of the great problems with 3D graphics: the hunger for hardware resources. The more photorealistic you try to be, the more

p278 ➢

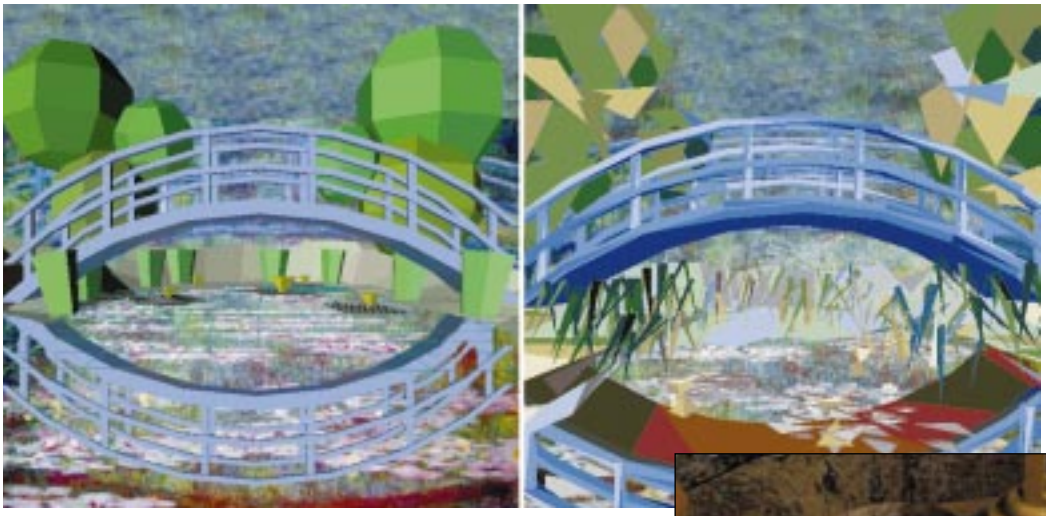
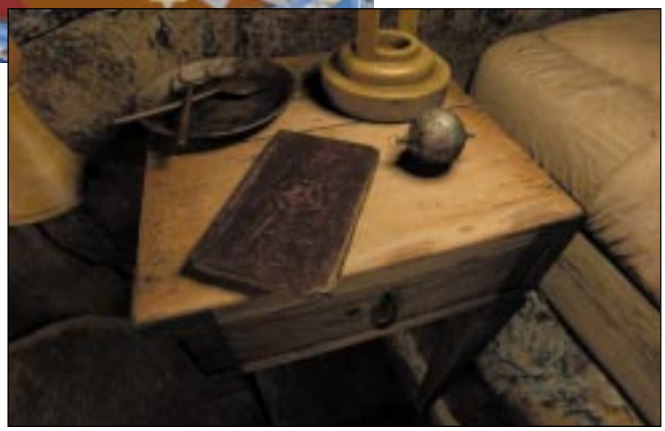


Fig 2 (left) Examples of the same scene in two different "LiveStyles"

Fig 3 (below) A preview of Cyan's Riven, taking to a new level the beautiful detailing that was a hallmark of its predecessor, Myst



ravenous this hunger becomes. Radiosity, for instance, depends not just on calculating how much light each shape will pick up from the light sources in the scene but how much of that light will be radiated back into the scene and picked up by other shapes. LiveStyles, in contrast, works by simplifying the scene and drawing it in the broadest brush strokes. As a result, Thinkfish claims that LiveStyles will allow fast, smooth, real-time rendering of the most complex scenes on standard PC hardware.

The Millers' tale

One scene that could never be rendered in real time on even the most powerful mega-specified supergizmo workstation is the island of Myst, the setting for the now legendary game from those grand wizards of game design, Rand and Robyn Miller.

I remember my first encounter with Myst: that luminous sky, those craggy rocks with huge iron cogs erupting from them, those elegant buildings and soaring conifers and the sound of water lapping and seagulls calling. Over three million copies of Myst have been sold since its launch in 1993 and the game has created a look that no other has equalled, except for Riven (the sequel to Myst). Thanks to CNET's Gamecenter <www.gamecenter.com> and other sources, like the unofficial Riven site <members.aol.com/mystsequel/index.html>, we have peeked at Riven, and the graphics look even better.

The secret of Myst's success, in my view, was the Miller brothers' decision not to go for the real-time rendering you get in Doom-style arcade games. Real-time rendering means the player has total freedom to explore and interact with the world the game inhabits. The downside is that the whole look of the thing has to be sufficiently simple to be created, frame by frame, 25 or more times per second on standard PC hardware, which results in the sort of gaudy graphics you get in Quake.

The Millers took a different approach. They saw rendering as part of the production process — part of their art, as it were. This means each possible view of the world being explored has to be pre-rendered, which limits flexibility. It also means each of those views can be exquisitely detailed and carefully composed. A typical, and for me, beautiful, example can be seen in Fig 3.

Myst was very much a Mac-based product, created mainly using Stratavision 3D. Riven was generated using SoftImage running on Silicon Graphics Indigo workstations — in other words, Hollywood-grade hardware. Also, according to CNET's report on Riven, Cyan (the Miller brothers' company) has produced its own shaders to extend the level of surface detail achieved in Myst. Despite having access to all that extra grunt, the best thing about Riven looks like being the revival of Myst's vision, the sort of 3D graphics that go beyond realism and into the realms of (dare I suggest it) art.

Models get personal

Occasionally I receive plaintive requests from readers wanting to know where they can get hold of clip 3D models for their projects. The good news is that there are huge libraries of such models available on the net. The bad news is that you have to pay for many of them. Thankfully, Avalon <avalon.viewpoint.com> is still free, and has a search engine which you can personalise to your own requirements. Avalon has a library of objects in most file formats and a selection of handy utilities. There are categories ranging from aircraft to dinosaurs, and models ranging from apes to the Venus de Milo, all free, but of varying quality.

If you are prepared to pay for your clip models, you will find a wider variety available. Viewpoint (which manages Avalon) has an enormous variety that can be browsed online (though not, when last I looked, purchased). Two other sites I managed to find are the REM 3D Bank <infografica.com/3dbank/s2.html> and the Marketplace <www.3dsite.com/marketplace/>. The latter allows online ordering via a secure website.

Benchmarks

If you are thinking of buying a 3D graphics accelerator card, have a look at Fourth Wave's benchmark site at www.fourthwave.com/3d-perf. It provides technical details on different benchmarks for measuring 3D performance, and a list of test results.

PCW Contact

Benjamin Woolley, writer and broadcaster, can be contacted at **3d@pcw.co.uk**.

AGP angst

Advanced Graphics Port — you can either take it seriously as an important graphics standard of the future, or you can ignore it. Benjamin Woolley wonders which is best.

Sometimes it is hard not to believe that, when it comes to marketing, the PC industry is about as principled as an Albanian pyramid seller. How else can you explain the announcement of MMX just after Christmas, or motherboards being shipped without USB support?

If you are interested in 3D graphics, there is a strong possibility you may be caught in a similar trap, and the reason is another little triplet of letters courtesy of Intel: AGP stands for Advanced Graphics Port, and we are now beginning to see a host of new graphics products being announced bearing those initials.

For instance, leading graphics hardware companies like 3DLabs, ATI and S3 have announced "AGP-compliant" chips. The first will be offering the Glint Gamma processor, ATI the 3D RAGE PRO and S3 the Virge/MX and GX2.

So what *is* AGP? And, when it arrives later this year, is it going to render non-AGP PCs as obsolete as non-MMX ones will no doubt prove to be? To address the latter question first, the obsolescence risk factor could be higher since AGP, like PCI, is effectively part of the PC's architecture. If your PC motherboard does not support it, you will be scuppered.

AGP was announced by Intel a year ago and marks an important step in the development of the PC as a 3D graphics platform. It effectively (although not literally) creates a Unified Memory Architecture (UMA). When Silicon Graphics launched its O2 workstation, UMA was one of its key features and one of the reasons it was so cheap (by Unix workstation standards).

In conventional systems, 3D graphics boards have to use limited on-board specialist graphics RAM to store the depth, colour and transparency information (the z-buffer, texture buffer and alpha-buffer, in technical terms) that goes to making up a 3D scene. Such RAM is expensive and there is never enough of it.

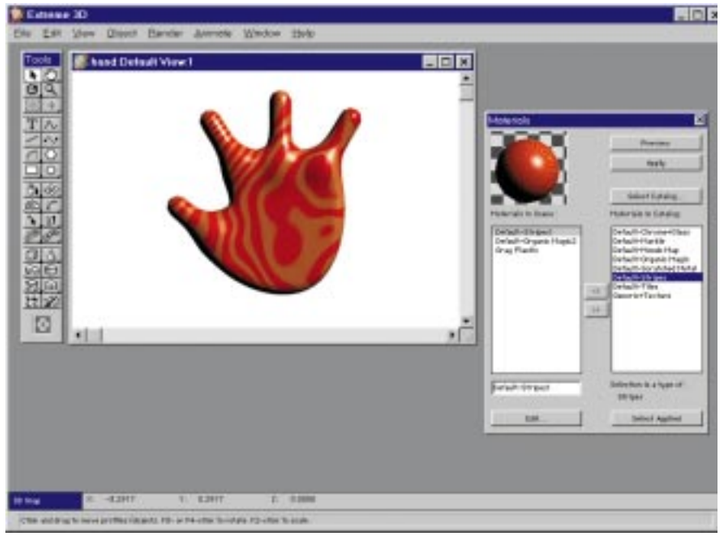


Fig 1 **Extreme 3D's metaform used to create a balloon hand**

AGP aims to solve this problem. It is a fast bus offering the 3D graphics accelerator chip direct access to the host system's RAM. 3D graphics applications can thus use any or all available memory as a unified block to build up each scene, enabling much larger scenes and much quicker rendering.

There seems little doubt that AGP will become an important graphics standard as 3D becomes increasingly pervasive on mainstream PCs. In a few years' time it may even be essential to run the latest 3D applications. Unfortunately, it is almost impossible to tell how many years. Like MMX, USB and like Windows 9 (?), the manufacturers have given no clear launch dates nor cost information.

Intel originally suggested it would begin to make its mark early this year but currently there is still no sign of AGP motherboards or graphics controllers. Most AGP chip vendors now talk in terms of shipments beginning in the second half of 1997. It could take a lot longer. It could suddenly pop up after Christmas, once many people have invested in non-AGP systems. So what is an upgrader to do?

Then again…

One possibility is to ignore all future developments and take advantage of the plummeting cost of systems that rely on older technology. It is generally agreed that the benchmark 3D platform at the moment is a 200MHz Pentium Pro with 64Mb of RAM, a graphics accelerator based on the 3DLabs Glint chip and a fast and wide SCSI interface driving a monster hard drive.

These systems are coming down in price, so now is a good time to get one. I spent a month using just such a system, Compaq's new Workstation 5000 installed with Windows NT 4.0 and I can report that it is perfect for modelling work, using applications like 3D Studio MAX.

However, if you don't have the budget for such a radical upgrade, don't despair. You can do a great deal simply by swapping your video card. I've been trying a Diamond Fire GL 1000 slotted into my old Compaq Deskpro. It was a nightmare to install, requiring a new BIOS for the board, a new set of display drivers and, eventually, a new operating system (I could not get all the Win95 drivers to work, so I had to swap over to NT 4.0). Once I had it up and running it meant that, even with an ancient Pentium, the system could display properly-shaded and lit models in a preview window that were rendered in more or less real time.

Boards like the Fire GL with 4Mb of video RAM (£285 ex VAT) use accelerators, like the 3DLabs Permedia NT chipset, which are aimed at the intermediary graphics market. So it may be worth upgrading the graphics controller only, before you dump the whole system.

Extremely frustrating

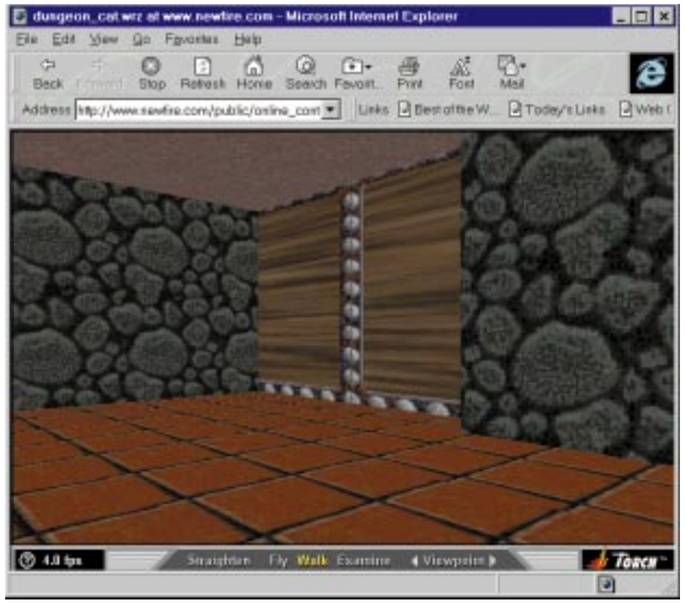
Extreme 3D from Macromedia is an important product. At around £500, it is priced substantially below full-blown professional packages and its specification now almost equals many of them. The recently-shipped version 2.0 now comes with a particle system and an implementation of a modelling method generally known as "metaballs".

Particle systems (see *PCW,* November 96) allow you to create clouds of particles (snowfalls, dust clouds, hailstorms) that are animated along a specified trajectory, with a specified degree of turbulence over time. The one supplied with Extreme 3D 2.0 is quite sophisticated and relatively easy to use. There is no library of presets, however, which is a pity because it takes a lot of time and practice to get the dynamics of a particle system to work (because you usually need to see the fully-rendered animation to establish whether the required effect has been achieved: low-res partially rendered previews are rarely adequate).

The metaballs modelling tool, called "Metaforms", was, for me, a more interesting addition, because 3D Studio MAX, which costs around five times more than Extreme 3D, does not include such a tool in its standard set. Metaforms (Fig 1) allows you to create simple shapes and fill them with a sort of virtual putty that you can then shape to create rounded, organic forms. Unfortunately, when it came to using the thing I suffered some sort of imagination crash; I could *not* think what to do with it. I tried dinosaurs, dolphins, cartoon characters and in every case ended up with something that looked like the sort of elongated balloons entertainers twist into ingenious shapes at kiddies' parties.

Extreme 3D has improved with this new release. It offers welcome and particularly strong support for VRML (including version 2.0). And the network rendering, which can be used to good effect even over small LANs (even those with a mix of PCs and Macs), is a boon. But, in my judgement, it still suffers from an awkward interface. For example, you can only swivel a view along one axis at once, which is very frustrating when you are trying to get a feel for an object's geometry: I find other budget packages, such as Truespace and Ray Dream, much easier in this respect.

Furthermore, Truespace version 3, which should be shipping by now, threatens to outclass the opposition with the promise of collision detection, its own implementation of metaballs ("Live Skin"), a way of moulding models called "Plastiform" and materials that give real physics (weight or elasticity) to the objects to which they are applied. It sounds exciting and likely to give Extreme 3D a run for its money.



The Heat is On

A number of companies are trying to find ways of pepping up VRML 2.0. Many, like Black Sun, have concentrated on the idea of developing it to help foster virtual communities. A new name, Newfire, is more interested in straightforward gaming. It has just announced Torch, a player which it claims can turn VRML worlds into Quake-style games (the example shown above is from a "Dungeon" demonstration game). Torch is designed to work with Direct3D, so it should make use of any on-board 3D acceleration with DirectX drivers. The company claims that as a result of this and a 3D engine that "carefully eliminates unseen polygons", Torch is four to eight times faster than other 3D internet players. Judge for yourself at www.newfire.com.

PCW Contact

Benjamin Woolley, writer and broadcaster, can be contacted at 3d@pcw.co.uk.

On top of the world

Benjamin Woolley and Bryce build a world in seven days: see how they set about their mountainous task. Ben's design style could be hampered, however, by the lack of file formats

It was an ambitious project. For the last episode of BBC2's *The Net*, I decided to have a go at building the world in seven days using nothing but 3D Studio Release 3 running on my Compaq Deskpro XL (which had what then seemed like a warp-speed 66MHz Pentium and a vast 16Mb of RAM).

I built up the world around me: a desk, a room, a fireplace, and a window overlooking a forest and snow-capped mountains. For the finale of this spectacle, the (virtual) camera zoomed out of the window, up through the soaring trees, up through the plumes of magnificent fireworks, and then turned to peer down as we pulled away into space, watching the mountainous terrain recede until we could see only continents and, finally, a globe like our own Earth floating in the speckled firmament. All this was done to the sound of the incomparable Sachmo singing It's A Wonderful World.

It wasn't a wonderful experience. Day in, night out, I had to re-render each sequence, then re-render the re-renders. Nothing went right, nothing. That is, except the bit I expected to be most difficult: building my virtual world's mountainous terrain.

One of the plug-ins then just released for 3D Studio was called Displace. You started off with a flat plane split up (tessellated, like a mosaic) into a fine grid. Each intersection in the grid represented a vertex, a point in the geometry. Over the top of this plane you mapped a two-dimensional image. This image was created by a fractal generator, which produced what looked like a black-and-white satellite image of a mountain range: peaks of white fading away to valleys of black.

The displacement plug-in used this fractal image to calculate how much to displace — or elevate — each vertex in the flat plane. The vertices mapped to the bright pixels were elevated the most; the vertices mapped to the darkest ones were elevated the least. The result was surprisingly natural-looking geology, produced in an instant. By exporting the fractal image to a paint program and adding colour to it, I could also create an accurate texture map to drape over the newly generated range, knowing that the rocky screes, grassy plains and snowy peaks painted into the picture would settle exactly onto the correct bits of the geometry.

I did not have time to get the terrain as richly textured as I hoped, but it did make me appreciate 3D software's potential to produce breathtaking natural vistas without demanding breathtaking skills and resources. Enter Bryce from software house,

p278 ➢

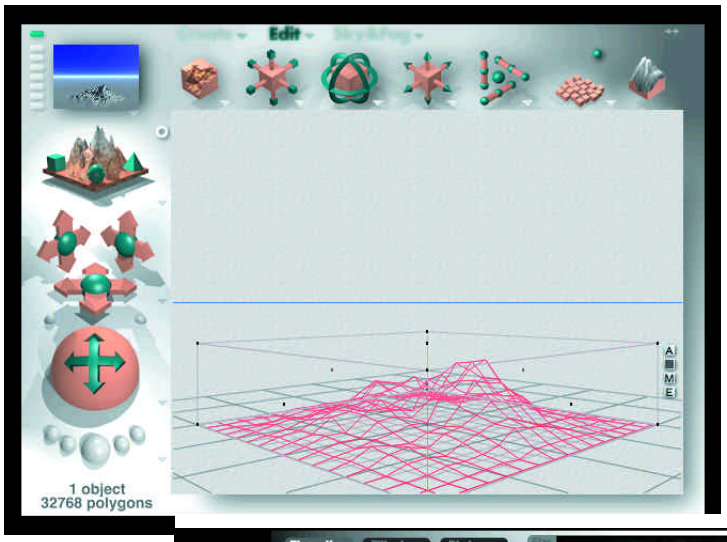


Fig 1 (left) Bryce 2's artistic interface

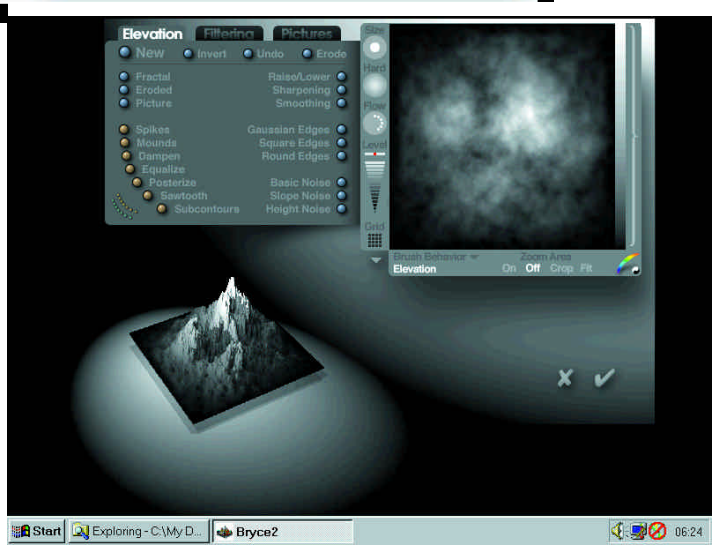


Fig 2 (below) Bryce 2's "terrain editor"

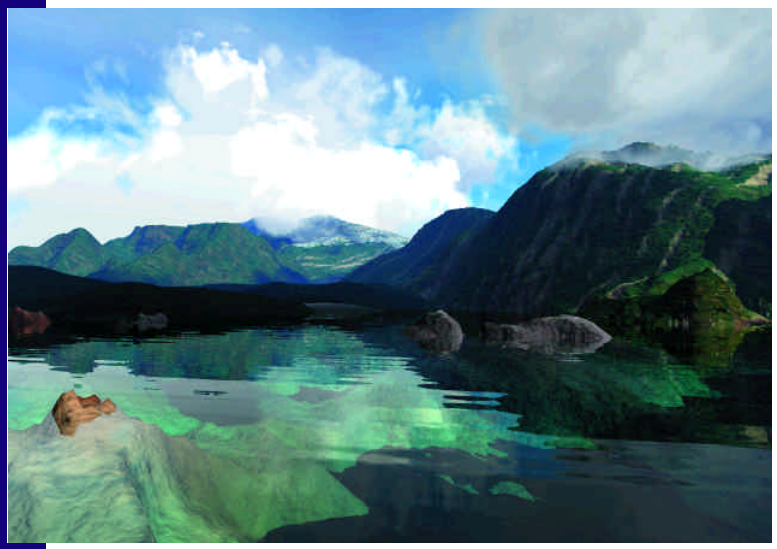


Fig 3 One of the samples supplied with Bryce 2. It's called Scotland, and is the work of Kai Krause, Metatools' resident guru

Metatools, the second version of which is one of the most enjoyable 3D tools around. Bryce falls into a new category of software product that is becoming increasingly common in the graphics market: plug-ins that have gone solo. Fractal (which, at the time of writing, was planning to merge with Metatools) did this with Detailer and Poser. Metatools did it with Goo and Bryce. Goo is for stretching and distorting bitmaps, and is firmly aimed at the recreation market. Bryce (named after a canyon in Utah) is for generating landscapes. It does it using the same basic principle as the 3D Studio Displace plug-in, but with some clever embellishments and the prettiest interface you've ever seen.

Firstly, the interface (Fig 1). It breaks all the conventions of the Mac (the platform for which most of Metatools' products were originally developed) and Windows. This isn't a dull desktop you're working on. Nor is it the sort of engineering studio-cum-nuclear-power station control room you get with products like 3D Studio MAX. It is, flatteringly for those of us who aspire to being artists, a studio. The icons pulse seductively when the pointer strokes them, giving a teasing hint as to what will happen if you touch them. The menu items are in soft focus and glow when you select them.

Behind the interface lie three main components. In conventional 3D parlance, you would call them a scene builder, a modeller and a materials editor. The scene builder allows you to create (from a wide selection of primitives) and manipulate objects — in particular, planes. A landscape is, when you think about it, a set of objects arranged between two infinite planes: a ground plane and a sky plane joined in the middle by the horizon. Each Bryce scene has these by default. You can also have infinite water and cloud planes that sit in between. You can then add a number of finite planes, perhaps one big one in the background that acts as a mountain range, and a smaller one in the foreground that represents the foothills. Where the peaks of the mountains poke through the cloud plane they are swathed in mists, and where the valleys dip beneath the water plane they become submerged beneath lakes.

To edit these planes you use the terrain editor (Fig 2), Bryce's main modelling tool. This is basically a bitmap editor for manipulating greyscale displacement maps. The window in the top right of the screen shows the map. To the left is a panel of tools for changing it, including ones that will add "erosion" (lots of little black cracks that creep in from the edges), raise or lower the elevation (increase or decrease the brightness), add noise, and so on. You can, of course, import bitmaps (created using a paint program like Photoshop) and even mix two together. The 3D black-and-white mountain range in the bottom left of the editor shows what the resulting terrain will look like, updated in real time. This sample terrain can be rotated using the mouse, so you can see it from all angles.

You texture these terrains using a type of material unique to Bryce 2. It is called a 3D texture, and the explanation in the manual is so paltry I didn't understand it. Suffice to write that the way the texture is applied to an object changes depending on the object's height and the angle of its sides. If the object is in the shape of a mountain, one texture can be used to put a white snowcap on its peak, a brown rock face on the slopes, and grassy cover on the plateaux.

The Materials Editor is not nearly as easy to use as Bryce's other components. For one thing, the terminology in the manual is non-standard. For another, trying to figure out how a 3D texture will be applied is about as intuitive as quantum mechanics. Thankfully, there is a generously stacked library of ready-made textures supplied with the CD, and, at extra cost, there is an Accessory Kit with more samples of both textures and terrains.

File formats

My dream is that products like Bryce will become the norm in the 3D world, replacing monstrous applications like 3D Studio MAX and Lightwave. Plug-in architecture is all very well, but it is expensive and cramps developers' design style. Instead, it would be much better to have separate applets: a selection of renderers, texturers, scene builders and modellers, each one with particular strengths for particular jobs.

There is one large obstacle standing in the way of this vision: file formats. Currently, there is no single standard for interchanging 3D data sets between graphics tools. This is partly because of proprietorial protectiveness of the software houses, but the problem goes deeper than that. With programs like Bryce having rendering novelties like 3D textures, it can be technically difficult to translate the resulting file into another format without losing important information. The most common interchange format in the PC world, DXF, is not up to the job, as it was developed centuries ago by Autodesk for CAD files and is really only suited to swapping untextured objects and meshes.

I do not know if it is possible to create a standard format that is capable of embracing all the novelties that products like Bryce 2 and Poser are bringing to the market. VRML 2, being extendible, may be up to the job. Apple's 3DMF, the format developed for its QuickDraw3D API is popular with companies like Fractal and Bryce (which have their roots in the Mac world), and it is flexible, so that may be one to consider.

Whatever happens, until a powerful interchange format emerges, the benefits of products like Bryce 2 will remain locked in their own little worlds.

PCW Contacts

Benjamin Woolley, writer and broadcaster, can be contacted at 3d@pcw.co.uk.

World in motion

It's a funny old world — or at least, it looks very different in 3D than the picture-book 2D views we're familiar with. Benjamin Woolley sets his sights on a more accurate projection.

The picture we have of the world is one that is fundamentally distorted because it is a two-dimensional version of a three-dimensional surface. If you look, for example, at the standard map of the world, the so-called "Mercator Projection", China appears to be roughly the same size as Greenland when in fact it is four times larger. This distortion occurs because the land nearer the poles is stretched out to the width of the equator (to form the rectangular shape of the map), so countries on the equator appear narrower than they should when compared to those closer to the poles. You can see how this happens in Figs 1 & 2. Fig 1 shows a map of the world. Note how huge Greenland is compared to China. Fig 2 shows the same map wrapped round a sphere, with Greenland now assuming its proper proportions. (I created the globe using Fractal Design's new Detailer package, of which more later.)

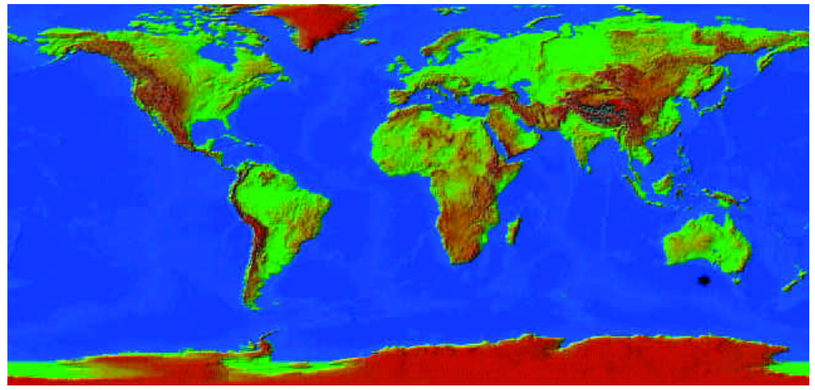


Fig 1 A texture map of the world. Note Greenland's size relative to China

There have been various attempts to produce more accurate projections (one of the best is said to be the Peters Projection, which makes Africa and other equatorial landmasses look huge, and more polar places, like our sceptred isle, teeny — you can have a look for yourself by browsing www.webcom.com/~bright/table.html), but none of them can be perfect. In the transition from 3D to 2D, something has to go, and in this case it is the true size and shape of each country.

As I have discovered from my email inbox, such problems are not confined to geography. A number of people have described the problems they have encountered trying get their texture maps to work, so I thought this month I would concentrate on this most perplexing area of 3D artistry, and at one tool that claims to make it easier.

Generally speaking, when you are trying to create a 3D scene, the sort of project you are dealing with is the reverse of Mercator's: you are trying to turn a 2D image into a 3D one, to take your flat map and wrap it round a sphere or, more usually, an irregular, complex shape. If you take another look at Fig 2, you can see quite clearly one of the first problems you encounter when trying to do this. Greenland's shoreline is slightly fuzzy, and there are two reasons for this. The first has to do with the size of the map: it has fewer pixels in it than there are on the surface of the object as seen from this perspective and at this size. You encounter this problem regularly, most obviously when the 2D bitmap, the texture, is placed on a wall or floor receding into the distance. As you can see in Fig 3, the bitmap is blurry at the point where the wall comes closest to the point of view. The solution to this problem is to match the texture's resolution to the wall's at the point closest to the camera. This means actually working out how many pixels there are down the edge of the wall, and making the appropriate edge of the bitmap the same number of pixels in size (in this case the bitmap is tiled, so I can divide the number of pixels in the rendered scene by the number of repetitions of the texture across the height of the wall).

The second reason for Greenland's blurriness is that where the map is approaching the poles, it is getting progressively scrunched up. There is no way of completely overcoming this problem unless you somehow manage to create a bitmap with progressively lower resolution towards the top and the bottom of the image. As far as I know, no image file format supports such variable resolution.

How, then, can you keep such distractions — "artefacts", as they are called in the business — to a minimum? By getting a grip on the way your 3D package projects or "maps" the texture onto the object. In all 3D packages there are basically three ways of mapping, usually known as spherical, planar and cylindrical. Spherical mapping is the sort demonstrated with the map of the world. Planar projects the texture onto the object as a film image is projected onto a screen. Cylindrical winds the image around an object like a label round a tin of beans. You can generally use these methods to texture simple objects: a vase, for example, can be textured using cylindrical mapping, especially if you use a paint program to stretch and contract the image to correspond with the vase's curves. However, some objects are just too complex to be textured using projected mapping, which means having to resort to a fourth method, surface mapping. A surface map is generated when the object is actually constructed, and if you think of the object as having a skin, the shape of the map is the shape of that skin carefully peeled off and laid flat.

If you are having problems getting a surface map to work, a weirdly distributed surface map could well be the cause. One way of solving it is to create a texture covered with a grid, using a gradation of colours so you can distinguish the position of the lines. Apply this grid as a surface-mapped texture to the object and see if that throws any light on how the map is arranged. Another easier solution is, of course, being able to paint and stick textures directly onto the surface of objects without bothering about technicalities like mapping co-ordinates. Which brings me on to Fractal Design's Detailer.

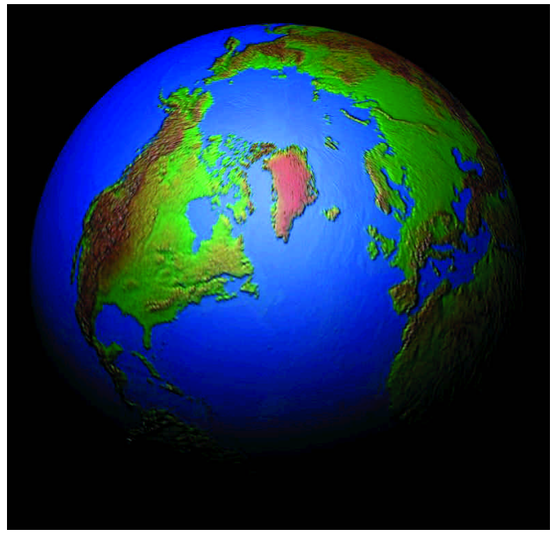


Fig 2 The texture map in Fig 1 wrapped round a sphere. Greenland assumes its proper proportions

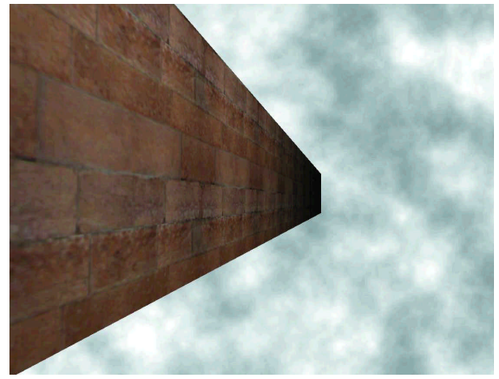


Fig 3 The purpose of this rather surreal image is to show a texture map being stretched beyond its resolution. Note the blurring where the wall is closest to our point of view

Detailer

When I first read the blurb about Detailer, I could barely believe it. "Amazing 3D Paint Program" proclaimed the press release. "A stunning new graphics application that allows users to paint on the surface of 3D models in real time." This could be the answer to all my prayers, I thought; 3D painting on the PC platform.

After spending a few weeks with Detailer, I have to say that it only partially lives up to its promise. It *can* work in real time, but most PCs will be stretched to the limit to keep up. And the design is fussy, introducing a whole new set of terms and concepts to a field already overburdened with both. However, I should point out that even if it is not quite 3D painting in the full-blown sense, it does offer one crucial new capability: it brings 2D and 3D together.

Generally, when I am working with textures, I have a paint package like Photoshop and a 3D package open on the system simultaneously. I edit the image, save it, load it into the 3D package's texture editor, apply it and then render the object to see what has happened. When, as is inevitably the case, I find the texture is too big, too small, too bright, too dark, too whatever, I have to start again. With Detailer, these two functions are combined. You have one window showing the 3D model being textured, another showing the 2D texture. When you change the texture, you see the result immediately in the model window. And there is another facility that helps deal with the surface mapping problem: being able to overlay a "mesh" that shows in 2D the surface ("implicit" in Detailer parlance) map of the object being worked upon — the skin, if you will. You can then paint over the mesh, building up a texture that maps directly onto the surface of the object.

Fractal Design is an interesting and increasingly influential company in the graphics field. Painter 4, Ray Dream Designer, Poseur, and now Expression (my favourite: a program that allows you to use drawing tools to paint) make up a more than adequate toolkit for the budding computer graphics artist. Detailer will be a perfect complement to this developing suite once certain shortcomings are dealt with: when there is some sort of mechanism for importing surface/implicit mappings or, even better, deriving them from the geometry; when the interface and jargon is simplified; when you can export the flattened-out meshes of objects with implicit mapping so you can use more sophisticated 2D packages to paint over them. I hope this is not unreasonable. I only suggest it because Detailer so tantalisingly holds out the prospect of making texturing a simple, even intuitive process.

PCW Contact

Benjamin Woolley, writer and broadcaster, can be contacted at 3d@pcw.vnu.co.uk

All in the past

Or is it? Virtual heritage promises realistic experiences of Gettysburg, the Colosseum, Stonehenge, and Hitler's vision of a post-war Berlin. Benjamin Woolley steps back in time.

As I write, an event is already underway in the centre of London, carrying the intriguing title "Virtual Heritage 96". Virtual heritage? What could that possibly be?

To the snobbish, all "heritage" is virtual — a fake recreation of the past that panders to the public's poor knowledge of history. It's all about grand country houses opening up shops to sell tacky knick-knacks, sales executives dressing up as Roundheads, and theme-park rides through reconstructed peasant villages saturated with synthetic sewage smells. What could be less real, more virtual?

Stonehenge, for one. To demonstrate the power of its new generation of processors, Intel got together with English Heritage and, under the direction of Professor Robert Stone, the VR pioneer who now runs VR Solutions, created a VR version of Stonehenge that could be accessed over the internet using Superscape's Viscape browser, a proprietary client for viewing scenes generated using the company's VRT authoring software.

Thanks to the involvement of English Heritage (the quango that manages Stonehenge), the consortium was given access to the site and built up a precise database of its geometry. This database was then used to generate models that showed what the 'Henge would have looked like through the ages, from 10,500 years ago to the start of the next millennium. The result was a good demonstration of how VR (in the sense of real-time 3D graphics) can be used to recreate a lost past.

Although words like "photorealistic" were bandied about to describe the quality of the model, nobody could possibly be fooled into thinking that the images of the Virtual Stonehenge you saw in the Viscape window were photographs. Nevertheless, it did give you the vaguest notion of what it might be like to be there without the distractions of coachloads of tourists and carloads of screaming children. It let you get inside the ring of stones, something we have not been allowed to do in actuality for years.

That, then, is an example of "virtual heritage". To quote Dr William Mitchell of Manchester Metropolitan University, a speaker at Virtual Heritage 96, it "…gives users the freedom to explore monuments that may no longer exist or may have been damaged or spoilt by the effects of tourism. Exploring virtual reconstructions leaves no footprints and can potentially allow a user to examine details that are just not possible to see physically."

Dr Mitchell has himself contributed to our virtual heritage with a project entitled "The Tomb of Menna", which formed the basis of his contribution to the Virtual Heritage 96 conference. Menna was an Egyptian scribe of the 18th Dynasty (whenever that was) and his tomb was discovered earlier this century in Thebes, the ancient city across the Nile from modern-day Luxor. Its recreation has been achieved using VRML, and pretty impressive it is too (Fig 1). The geometry is simple. The detail lies in the textures, which are highly compressed JPEG images of the friezes on the tomb's walls.

This is just one of an expanding array of projects that make up the virtual heritage movement. There's virtual Gettysburg, a virtual Chinese Terracotta Army and the virtual Colosseum. I myself was involved in realising a virtual Germania. Hitler planned to rename Berlin as Germania after he had defeated the allies. He even got his architect, Albert Speer, to draw up detailed plans, which featured on a TV programme about Berlin's history and future as a united Germany's capital. With the help of our friends at the modelling company, Modelbox, we used the plans to render up a series of animations. It was, I remember, an exciting process, since it allowed us to experience the impressive and oppressive scale of Speer's grandiose vision in a way Hitler himself never could (Fig 2).

It is possible that virtual heritage is a passing fad. The point of preserving Stonehenge is to provide a means of keeping in contact with something authentic in an increasingly artificial world. So to that extent it seems to be a contradiction in terms. It might also provide an excuse for

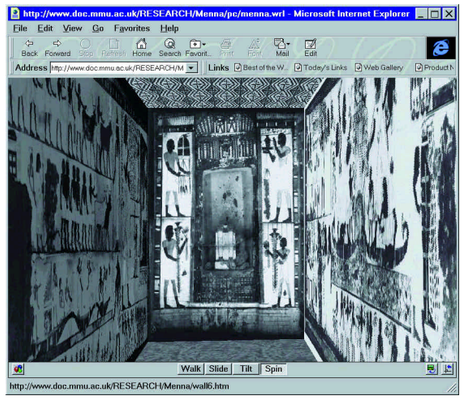


Fig 1 The Tomb of Menna by Dr William Mitchell

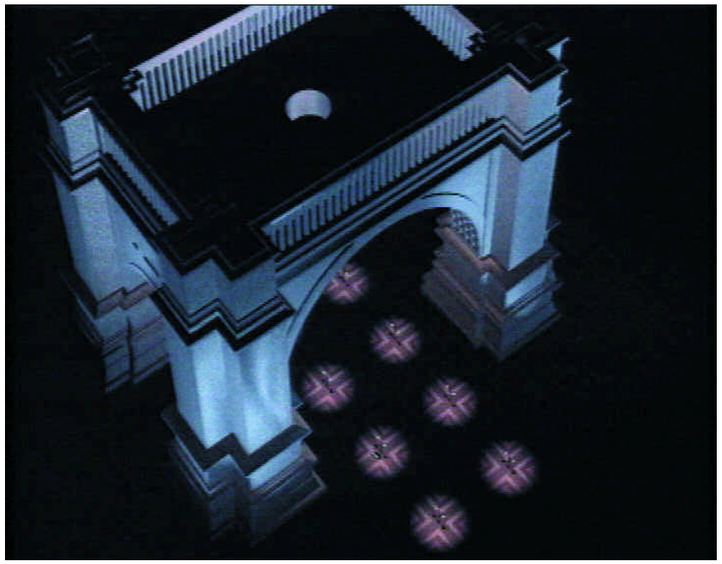


Fig 2 The Great Triumphal Arch of Speer's Berlin at night. This is a video grab, courtesy of Modelbox, hence the slightly fuzzy quality. It comes from the film we made recreating Speer's vision for Hitler's dream city. To give a sense of the arch's size, we inserted footage of a real car driving beneath it. At this scale it is a mere speck, barely visible at all, caught in one of the pools of street light illuminating the ground

authorities such as English Heritage to deny access to monuments that are currently open to the public (tourist-free sites are, after all, a lot easier and cheaper to manage).

But as IBM, for instance, demonstrated in its reconstruction of Dresden's magnificent Frauenkirche, which was demolished by the Allied bombing raids in World War II, virtual heritage provides us all with a valuable way of recovering what we can no longer experience.

The virtual universe's Big Bang

The technologies being developed for building shared spaces or multi-user virtual worlds or whatever you choose to call them are now emerging thick and fast. I am pleased to report that everyone is being extremely co-operative in this enterprise, even now that we have a new contender on the scene: Open Community, from the Mitsubishi Electric Research Laboratory (MERL) in Cambridge, Massachusetts (formerly known as Universal Worlds).

Open Community (www.merl.com/opencom/opencom.htm) does not come from the VRML community (although it will support worlds built using VRML models). Rather, it has its origins in a technology developed internally by MERL called SPLINE (Scalable Platform for Large Interactive Networked Environments). SPLINE has been under development for more than three years. Using it, a virtual world has already been built: Diamond Park, a place "where avatars could travel around a large park, talk to others using proximity-based voice chat, ride bicycles in a velodrome, create new world views, and play multi-user games".

To create Open Community, SPLINE has been combined with the Universal Avatar initiative (www.chaco.com/community/avatar.html) which aims to provide a standard for avatars so a virtual identity you create for one shared space on the internet could be used in another. The result is a sophisticated-looking application program interface (yes, yet another API) based on Java (yes, yet more Java) that embraces both the network and content sides of social spaces.

It is the fact that Open Community deals with the network side of the social spaces issue which, in my opinion, makes it particularly important because it is the network that makes social spaces unique, and presents the biggest challenge to making them work. The main problem is "latency". As we all know, you don't always get what you want from the internet when you want it. Data floods down the line one minute and dribbles down it the next. The reason for this is that the TCP/IP protocols, on which the internet depends, were not designed to deliver data in real time. They were designed to route things like email, files and scientific data which, generally speaking, one can afford to receive a minute or two later than expected.

For real-time applications, though, latency is a killer and shared spaces are, by their nature, real time. So Open Community promises to provide a set of tools which will manage this problem. A variety of techniques are suggested, ranging from the obvious (supplying the bulk of the data for a world on CD-ROM) to the ingenious. An example of the latter, given by the authors, is a simulated baseball game. When the batsman hits the ball, and a fielder runs to catch it, the batsman's "client" (the program running on the computer owned by the person controlling the batsman) anticipates where the ball will land, and passes on that information to the fielder's client before the ball has actually been hit. So the fielder's client can show the ball's initial direction even if the information about its actual trajectory is delayed by the network.

As we continue through 1997, I think the collaborative spirit in which Open Community and other initiatives are being discussed means there is a good chance of the industry doing justice to this most significant and exciting of 3D graphics/virtual reality applications. It is nice to start the year on such a positive note.

Render unto Criterion…

In the December issue column, I wrote about the Direct3D and QuickDraw 3D APIs. Who, I and many others were asking, will lead: Microsoft? Apple? Well, as I should have mentioned, for the time being neither will because the real leader is probably Criterion, the British company responsible for the RenderWare API. Criterion, now owned by Canon, claims RenderWare is the market leader. It is certainly popular, and is used in many games and VRML browsers such as Netscape's Live3D and SGI's Cosmo Player. It is fast, too (unlike Direct3D version 2, according to recent reports); you can see for yourself by trying out World Inc's AlphaWorld (www.worlds.net).

PCW Contact

Benjamin Woolley, writer and broadcaster, can be contacted at 3d@pcw.vnu.co.uk

Out of this world

The virtual world is huge, and getting better all the time. Benjamin Woolley dons his avatar and goes on tour to produce a rough guide to strange lands.

I know the web is supposed to be a revolutionary new medium, different from all its predecessors, being interactive, using multimedia and all that. But when you think about it, most of the information you get is not so radically different to what you glean from print and TV media: flat pages of illustrated text that look like magazine pages, combinations of sounds, text and video that could pass for designer news bulletins. There is, however, one "media type" the internet can deliver which is really novel: the shared virtual world. By this, I mean a computer-generated space that a number of people can access simultaneously across a network and inhabit via a virtual stand-in or "avatar".

Experimental versions of such worlds already exist: notably the WorldsAway game which you can access through CompuServe, and AlphaWorld from Worlds Inc., which is on the net at www.worlds.net.

WorldsAway is not really a shared "space", since the environment is generated not out of proper 3D models but 2D backdrops upon which avatars and objects are superimposed. AlphaWorlds, by contrast, is more like the authentic article, and one that has been quietly developing a substantial 3D presence since its public launch in October 1995. It was created by Worlds Inc., to showcase the company's interactive 3D technology which it has dubbed, picking up on Microsoft's flavour of the month, Active Worlds.

Last October, the company announced that it would begin shipping an Active Worlds Development Kit (to run on Sun, SGI and Windows NT platforms) so that third parties can create and publish shared spaces of their own.

AlphaWorld is impressive. You access it by downloading Worlds Inc's own client or browser program and "teleporting" to the AlphaWorld co-ordinates. The first time you enter, you are confronted with a void. Slowly, the world takes shape before your eyes, object by object, texture by texture, efficiently "streamed" down the line so you (or rather, your avatar) can begin to wander around (using the mouse or cursor keys) before all the data has been downloaded.

The world is huge and getting better: the full data set for all the models and textures probably runs into tens of megabytes. Thankfully, data is cached to your local hard drive so the more you access the world, the faster it appears on your screen.

Some of the first objects to appear are avatars, represented by virtual mannequins of various sizes, shapes, sexes, species and demeanours. Each one you see will be driven by another person who is sharing the space. They can see you, just as you can see them, and you can interact with them in much the same manner as a text-based MUD, through gestures or "speech" (typed text, displayed as a speech bubble above your head).

When you apply for "immigration" to AlphaWorld, you are given a standard avatar, but you can select another from a whole library of character types, each identified by a suggestive name. For instance: Butch, Helmut, or Shred (the surfer) which is a particularly popular choice, as you can tell from Fig 1; two Shreds are walking past me as I stand in the middle of AlphaWorld.

Another, perhaps more interesting, form of interaction possible in AlphaWorld is being able to shape the environment itself. You can build on any unused section of property by duplicating objects you find elsewhere in the world and dragging them

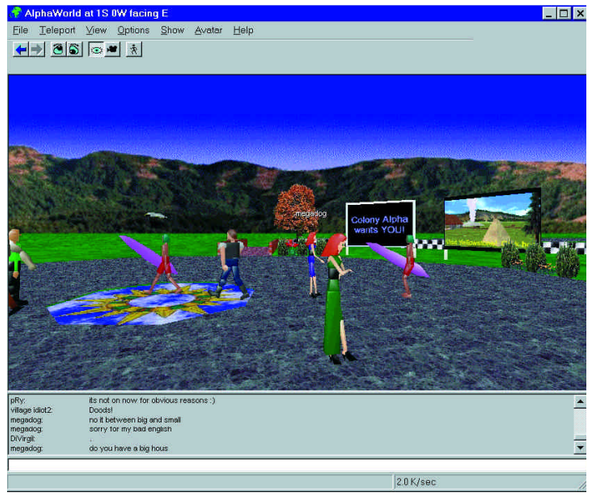


Fig 1
AlphaWorld's crowded central plaza

to your patch. You can alter some features of these objects (although not the basic geometry and look) and even give them behavioural characteristics. For example, your object could play a tune when someone bumps into it.

At the time of writing, a wide assortment of blue chip companies and other organisations were experimenting with Active Worlds technology and building their own spaces for people to explore. These include Visa which is designing a 3D online bank, Yellowstone National Park, and the Nokia phone company which is aiming to bring a little of the Scandinavian spirit to your screens.

One world which I considered to be particularly good was the Cyborg Nation (Fig 2). It was still under construction when I visited, and sparsely populated but given that what you see is being rendered in real time, I think it looks lovely. The sky and background are beautifully realised, and it is a delight to wander aimlessly around, awaiting some new object to spring up before you. I encountered the facade of a terraced house, a hovering metallic doughnut, a room with golden walls and a wireframe dome — it was rather like being in a Dadaist painting.

Although Active Worlds uses standard data file formats (such as RenderWare's RWX), which means third-party tools can be used to develop content, the system is proprietary. You will need the Development Kit to assemble worlds and publish them. This strategy has resulted in the steady evolution of an extremely effective product, but one that cannot rely for its future development on the same level of collaboration and competition as a technology relying on open standards. For that to happen, another approach is called for — one that is embodied in the new Living Worlds proposal.

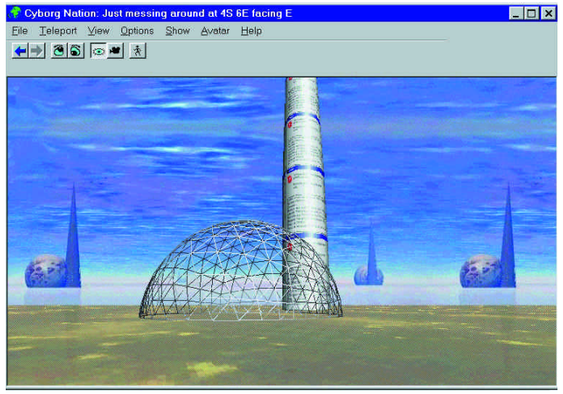


Fig 2
Cyborg Nation's virtual surrealism

Living Worlds

The idea behind Living Worlds is to use VRML 2.0 (see *Hands On 3D Graphics*, *PCW*, Dec'96) as the basis of a standard that allows the creation of the same type of shared virtual spaces which the Active Worlds technology already provides, but can be built, published and accessed using VRML-compliant tools and browsers.

Like HTML, VRML is totally public. Anyone can use it to create 3D objects and scenes that can be distributed across the web. Unfortunately, although it does allow the building of avatars and interaction with virtual spaces, these mechanisms are not standardised in a way that ensures true "interoperability", to use the term adopted by the Living Worlds team (a consortium of representatives from Sony, Paragraph, Worlds Inc., and others).

To illustrate the problem, the team dreamt up a series of scenarios: suppose someone called Art is at "home", suggest the Living Worlds team (in other words their avatar, or virtual presence, is in a 3D model of a living room realised using VRML); Art has recently "redecorated" his room, and there is new artwork on the walls that is automatically updated each month from some sort of interior design server.

This scenario shows how even the simplest of virtual spaces can quickly blur the distinction between authoring and using, and can come to rely on a variety of different sources and developers which update it, dynamically.

The Living Worlds team then imagine Art has some virtual visitors called Betty and Chuck (very American). They knock on the door. He opens it, sees them and greets them. This is the first point when some of the key interoperability questions are raised. How do Betty and Chuck "find" Art and how do they interact with him. Remember, there is no standard mechanism under VRML for words or gestures. Can they speak to each other, gesture, touch, sniff, hit… *mate*? — none of these points are unambiguously dealt with by VRML 2.0.

There are other, more subtle, issues the Living Worlds authors consider. What if you were able to exchange or buy virtual objects with behaviour characteristics? Suppose such objects could be delivered to you as complimentary gifts. What if the object were able to do some damage to your scene (perhaps a virtual puppy that bounces around Art's room, ruining the furniture and staining the carpet)? As the authors put it: "If this is beginning to sound like a virus, we've made our point. Multi-user apps in VRML, like those in any other language, will need some reliable way to protect themselves from inappropriate access."

Living Worlds is already coming up with answers to these questions, and in particular to the issue of avatars. There has already been an attempt by one team to create a "Universal Avatar" standard (you can find their discussion paper at www.chaco.com/avatar/avatar.html), and Living Worlds takes this a step further by refining the definition of an avatar and distinguishing it from other types of objects that would be expected to populate a shared space.

Avatars are usually defined as "transient and arbitrarily mobile" objects because they come and go, and are driven by humans. In contrast, other objects are "persistent and predictable" because they are driven by programs. However, most expect shared spaces to be populated by "bots" which are, essentially, program-driven objects designed to behave as if they were avatars, so any future standard will have to embrace their behaviour, too.

These are early days for shared spaces and the technologies that will create them. It remains to be seen whether it will be the proprietary approach (via Active Worlds and any emerging competitors) or the open standards approach (via Living Worlds) that will set the agenda and deliver the goods. Either way, it must surely be the area where 3D and the internet can create something truly unique.

•PCW Contacts

Benjamin Woolley can be contacted at **3d@pcw.vnu.co.uk** He presents The Net, which will be broadcast on BBC2 from mid-January.

Active Worlds www.worlds.net
Living Worlds www.livingworlds.com

Direct action

Benjamin Woolley assesses Direct3D, Microsoft's promising API for adding 3D functions to applications. He also finds himself in the thrall of power mania: which hardware is big enough?

The internet is not the only area of the information revolution that Microsoft once neglected and is now determined to dominate. 3D graphics are also now firmly in the company's laser-guided sights. Its strategy has been to buy up existing technologies and Microsoften them up for global exploitation. One of these is Reality Lab, a set of programming tools originally developed by the British company, RenderMorphics, for rendering simple textured shapes in real time. Microsoft has renamed it Direct3D and developed it as the 3D component of its burgeoning multimedia application programming interface, DirectX, version 3 of which had just been launched at the time of writing this. The DirectX "evangelists" (a troop of which are bound for Europe, I am told) are promising that their technology will enable PCs to equal the current performance of consoles and arcades once hardware developments like Intel's MMX and Microsoft's own Talisman (see the November column) are commonly available.

Direct3D is an API, which means it acts as a sort of programming language (used in combination with an existing one, such as C++) for adding 3D functions to applications. Those applications may be games, they may be programs for authoring games and other 3D content, they may be molecular modelling packages, even databases or spreadsheets.

Microsoft has put a lot of work into Direct3D, and the demonstrations I have seen on the developer CD-ROM are promising. A simple textured sphere or teapot (the standard artefact for graphics demos), for example, will render smoothly in real time in a 320x240 window on a standard

An image entitled "Screw the Mold" by Sandford Bemi Faisonat, which features in the Apple QuickDraw 3D Gallery *(quickdraw3d.apple.com)*. The image was rendered using QuickDraw 3D, although obviously not in real time

Pentium system. Direct3D (and, indeed, DirectX as a whole) also has the important feature of being able to take advantage of whatever hardware resources are available. If it finds a 3D graphics accelerator, it will be used, so long as there is a driver, which is likely, as most of the major 3D graphics chips are designed to support Direct3D. But equally important, if no acceleration is available, Direct3D objects, and any sounds or 2D images with which they are combined, will still be displayed, generated by a "Hardware Emulation Layer" that reproduces in software any functions that are unavailable in hardware.

Direct3D is not the only 3D API on the market. There is OpenGL too, which is aimed at the higher-end market and is already well-established. More significantly from a PC point of view, there is QuickDraw 3D from Microsoft's old rival, Apple. In at least one head-to-head comparison (published in the American magazine, *Byte*) QuickDraw 3D came out ahead of Direct3D for offering a greater range of object primitives and for its support of both the Mac and Windows platforms. Some of the Microsoft literature claims that DirectX, too, will be cross-platform. There are some doubts about this. According to at least one source within the company, the main purpose of the technology is to give 32-bit Windows operating environments a competitive edge over rivals, which obviously include Macintosh.

So which API will prevail, and does it matter? It certainly matters, because either Direct3D or QuickDraw 3D are likely to provide the basis for 3D becoming a standard part of the PC environment, as commonplace as sound and 2D graphics are now. You will need to consider this when choosing both software tools and hardware, trying wherever possible to keep your options open by getting support for both (which most third-party developers are, so far, promising to provide).

The question as to which API is likely to prevail is a trickier proposition. We all know who has the marketing muscle. We all know who controls the operating system (or at least, the one used by the vast majority). But 3D is a relatively new field in PC terms, QuickDraw is already well supported, and you only have to visit Apple's QuickDraw server (quickdraw3d.apple.com) to see that the company means business.

Infobyte

An Italian company called Infobyte specialises in creating VR tours of historical sites that are truly spectacular. They include the stunning Giotto frescoes in the Basilica of St Francis in Assisi (pictured here), St Peter's Basilica, the Coliseum and, most recently, the restored tomb of the Egyptian queen Nefertari, a VRML version of which can be explored by pointing your browser (running on an extremely powerful workstation) at the company's excellent web site, www.infobyte.it. Once this sort of thing runs in real time on an ordinary PC over a standard internet link, I think 3D's day will truly have arrived.

Power mad

Last month, my Compaq Deskpro's hard disk drive decided to experience a strange, slow-motion crash, deteriorating from full working order to complete cabbage-like coma in the space of an hour. I packed it off to my supplier, where it gathered dust for three weeks awaiting Compaq's delivery of a replacement.

During its absence I had to resort to my backup system, an old 486 Viglen Genie, which, unlike the Compaq, has chuntered away reliably in the background without a squeak of protest since I bought it some time in the last century. Being modestly specified in all departments except RAM (it has 16Mb), the Viglen, I thought, would prove to be unusable. In fact, I found it capable of doing just about the same amount of work. For 3D, I returned to Autodesk's 3D Studio running under DOS; for writing, Microsoft Word running under Windows 3.11; for the internet, good old Pegasus and WinFTP (I decided to forgo the delights of the web for a while). It was not tidy, it was not integrated, but it did work.

Those of us who are working with 3D graphics are currently in the thrall of power mania. We are constantly told that more means more: more processing power, more RAM, and yet more sophisticated software means more creativity, more spectacular effects and yet more moolah.

Dear Santa...

In its opening months, 1996 seemed it might turn out to be the moment when 3D finally fulfilled its promise. Creative Labs was selling the 3DBlaster board, VRML was becoming better known. However, the 3DBlaster did not turn out to be the graphics equivalent of the SoundBlaster because only individual programs (games) could take advantage of it. VRML, too, was a bit of a damp squib; few had the hardware to do anything with it, fewer the desire to spend their online hours wandering terrains that look like they were designed by the Early Learning Centre.

Now, as the New Year arrives, one gets the distinct impression that things are starting to move. With the Millennium board and now the Mystique, Matrox has started to establish 3D acceleration as a standard part of the PC graphics subsystem. With the plummeting price of memory, systems are coming equipped with the 16Mb of RAM that is the absolute minimum for handling textured 3D data. With the emergence of mainstream APIs (see main story), we at last have a mechanism for bringing the benefits of the third dimension not just to games, but to a whole welter of applications.

But I do not expect 1997 to be year zero: we have some way to go yet. Santa keeps forgetting to pack his sleigh with such essentials as modular, easy-to-use 3D authoring tools (the current crop are overweight and monolithic), a standard for plug-ins, and the imagination booster all of us involved in the graphics business need if we are to start to come up with content that is both wonderful and practical .

For me, what 1996 lacked most was a Myst, some game or virtual artefact that aroused one's excitement in the possibilities of 3D. So, Santa, please could you give us another of those in 1997? Not Myst 2, but something that demonstrates what wonderful, colourful, inspirational landscapes that even a humble PC can help create.

So when I sat down in front of what to many must still represent the pinnacle of desktop computing power, a Silicon Graphics workstation, it was in a mood of extreme scepticism. The machine in question was SGI's new "personal" workstation, the O2*. SGI's definition of personal is somewhat different to, say, Viglen's. The cheapest O2 costs just over £5,000, for which you get a 32Mb system armed with a MIPS R5000 RISC processor running at 180MHz. It offers blistering graphics performance through a "unified memory architecture" (i.e. no special-texture RAM) combined with built-in hardware acceleration and a system bus that can shove data around at a rate of 2.1 gigabytes per second.

I spent about an hour on the O2, and found it (temporarily at least) restored my confidence in technology. It was the first time I had used VRML that was both nice to look at and explore, smoother than anything I have so far experienced on an NT box or, for that matter, a Unix one. That, of course, was partly because it used the latest version of SGI's Webspace VRML 2.0 browser. But it also seemed to indicate that SGI might still retain the edge when it comes to optimising hardware for graphics.

However, do 3D artists have to start contemplating spending more than £5,000 in order to do decent work? Do we really need all that extra power? And if we do, should we pay the premium that is inevitable if you leave the general-purpose PC architecture behind and choose something from SGI? Or should we start thinking about going back to basics: stepping off the technology roller coaster, settling back with the old products that we know and like, and leaving it at that?

For me, for the moment, not even the allure of an O2 can completely discredit the latter strategy. But then, my Deskpro is now back and apparently working well, I have started to use the Workstation edition of Windows NT 4.0, and I have been eyeing a rather nice accelerator board. It can only be a matter of time before power madness once again prevails.

** See PCW December 96 for a full review of the Silicon Graphics O2.*

PCW Contacts

Benjamin Woolley, writer and broadcaster, can be contacted at **woolley@illumin.co.uk**. His home page is **www.illumin.co.uk/woolley**

World vision

Benjamin Woolley takes a fresh look at VRML 2.0 — using Moving Worlds technology, it lets you use 3D models and simulations on the net.

When I last wrote about VRML 2.0 (the second version of the standard for using 3D models and simulations over the internet), I got into a spot of bother. I described the decision by SGI and Netscape to declare the Moving Worlds technology as the best basis for the new standard as "pre-emptive".

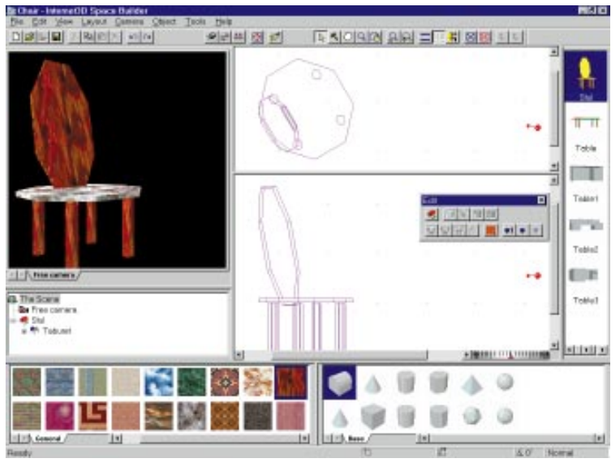


Fig 1 **Paragraph's Internet 3D Space Builder**

The two companies had announced the support of just about the entire 3D industry (excluding Apple, who is now on-side, and Microsoft) before there had been much discussion about the alternatives. My remarks were posted to the VRML newsgroup, where they provoked some sharp criticism (and a little support which was, for some reason, offered anonymously). It seems that a few members of the group did not like the suggestion that the VRML community was being, or could be, manipulated. Each proposal for VRML 2.0 would be assessed on its technical merits alone.

In the event, Moving Worlds was voted in as the new VRML standard by VAG, the presiding VRML Architecture Group. Microsoft's ActiveVRML, Moving Worlds' main contender in a field of six proposals, attracted a large negative vote. Obviously, VAG members felt that Microsoft's hold over the internet would develop very nicely without help from them.

Since the final draft of the standard was formally adopted on 4th August, VRML 2.0 has had a chance to get a toehold on the web, helped by a substantial presence at this year's Siggraph show in New Orleans. So this month I thought I would put all past disagreements behind me and have another, more thorough look at VRML 2, its capabilities and its future.

A better world

The standard is ambitious, promising to provide "a richer, more exciting, more interactive user experience than is possible within the static boundaries of VRML 1.0." There are five major areas of improvement: "enhanced" static worlds, interaction, animation, scripting and prototyping.

The enhancements to static worlds include the ability to put in backdrops, fog and bumpy terrain. Rendering scenes with fog may turn out to be too processor-intensive for all but the most powerful systems so, for a while, I don't expect to see it used much. But the backdrop facility which places a bitmap, such as a landscape, into the scene's background could prove to be a useful and simple way of adding a little more colour and character to a world.

The key to the interaction improvement is the concept of the "sensor". There are various "geometric" sensors that are triggered by events in space, and a sensor triggered by events in time. When a sensor is triggered, it can invoke some other node to be executed (a node is the VRML term for a programming command — see the *PCW* May edition of this column). An example of a geometric sensor is the ProximitySensor node. For this, you define a box-shaped region in space. If the user enters this space while navigating through the world, an event is triggered (for instance, an object in the vicinity becomes animated).

A particularly important enhancement to the VRML sensory environment is the introduction of sophisticated collision detection. A collision node prevents the user, or more precisely their "avatar", entering either specified geometry or all geometry in the scene. In particular, you can ensure that the user does not plough into uneven terrain instead of walking over the top of it.

In the field of animation, the third area of VRML 2.0 improvements, there is a whole new class of nodes called "interpolators", which can be used to alter an object's colour, position, orientation and size as well as other features.

Objects can also be controlled and given behaviour using scripting, the fourth main improvement. Interestingly, the VRML 2.0 specification does not specify which programming language should be used for scripting. The standard specifies that the language is one supported by the browser being used to view the world. This, of course, currently means Java but perhaps one day a developer will come up with an alternative that is tailored to 3D animation and simulation?

The final area of improvement is "prototyping", also known as "extensibility". VRML 2.0 allows new nodes to be created out of existing groups. You imagine this will typically be used to create nodes for complex objects. Since it is possible to pass parameters and event information to and from these prototype nodes, they can be controlled just like any other.

These and other enhancements have turned VRML from a basic 3D scene description language into a sophisticated animation and simulation programming tool. This should mean that, as promised, it can provide "a richer, more exciting, more interactive user experience". But in addition, it could also mean that creating these user experiences will be much more of a complex business. The VRML specification contains concepts and jargon that all but the most competent programmer will find daunting.

None of this is the fault of the standard's designers. They have tried very hard to make the full specification accessible and understandable. You can download it from the VRML home page (vag.vrml.org). It is a 1.5Mb file that has been compressed using the Unix "tar" format and you will need a utility like WinZip version 6.1 to decompress it. You will find it very well laid-out in HTML format, with convenient links for jumping between the various sections. There are a few tutorials, one based on a Siggraph '96 session (at www.sdsc.edu/siggraph96vrml/) and a couple available through SGI's VRML server (vrml.sgi.com/experts/vrml2tutorials.html). Of course, by the time you read this, there may be more.

For those with neither the time nor inclination to tackle such complexities, it is worth trying out some authoring tools that are beginning to emerge. You should discover a list of those online at the starting point for all VRML work, the VRML Repository at /www.sdsc.edu/vrml/. At the time of writing, only two tools were listed that supported VRML 2.0: Internet 3D Space Builder and Virtual Home Space Builder, both from Paragraph.

I tried Internet 3D Space Builder (Fig 1) and can report that it is one of the neatest, nicest 3D apps I have yet downloaded from the internet (from Paragraph's web site at www.paragraph.com). I was using a beta version that had no documentation and did not support all of VRML 2.0's features like animation but in its basic design it worked like a dream, allowing me to build worlds out of primitives and a small collection of more complex objects (mostly office furniture) simply by dragging them into the scene. A preview window showed what the result would look like in a browser and even allowed me to drag textures straight onto the surfaces of objects.

Fig 2 **SGI's Jupiter demo: the lower panel in the window provides a 3D tour of Jupiter and its moons, while the top panel delivers the facts**

It's a small world

Before having a go at building your own world, you might want to see what others have achieved so far. At the time of writing, there was little to see. Unlike Java or Shockwave, you sense a reluctance among content providers to use VRML and you can understand why.

The hardware is not yet in place to make realtime 3D a credible form of communication. Many people still have 486s, most have 8Mb of RAM or less and hardly any have 3D acceleration, or connections faster than 28.8bps. This means that the simplest VRML world or object behaves as if its batteries had run down. VRML tends to look ugly, as well, because the detailed textures are too heavy on resources.

Nevertheless, there are a few demonstration worlds around (see the Jupiter example in Fig 2), hinting at the riches to come. I viewed them using the beta 2 version of SGI's CosmoPlayer, which was, at the time, one of only three browsers listed by the VRML Repository as supporting VRML 2.0 (Netscape's first version of Live3D is a subset of VRML 2.0). I also tried a couple of scenes created using a Doom-to-VRML 2.0 converter. The results were incredibly slow to load and run, but suggested one possible source of material that would look good once 3D accelerators become more commonplace.

There remains some debate about whether VRML is the way to go with 3D on the internet. Various companies are touting alternatives. According to the graphics industry newsletter *Wave*, there is growing interest in using the so-called "DIS-Lite" standard as an alternative. DIS (Distributed Interactive Simulation) is a protocol developed by the American Department of Defense for networked simulations of battlefield operations. The companies that are working in this field, like Mak (www.mak.com), see a lite version of DIS as being the most effective way of building up a new generation of internet simulations and games.

The VRML Architecture Group may have voted on its vision of the 3D future but I sense that when it comes to the wider industry, the jury is still out. The time when VRML enjoys the same sort of global acceptance as HTML, or the same level of commercial support as Java, will be like a complex 3D world downloading onto a 486 via a v34 modem — slow in coming.

PCW Contacts

Benjamin Woolley, writer and broadcaster, can be contacted at **woolley@illumin.co.uk**. His home page is **www.illumin.co.uk/woolley/**

Why did the **chicken** cross the road?

To see Talisman, the new 3D hardware architecture from Microsoft. Benjamin Woolley looks at its application in real-time graphics, and gets in a twist about special FX.

I first visited SIGGRAPH, America's annual computer graphics megafest, in 1989 when it was held in Boston. I still have the mousemat to prove it, which features a large red lobster (Boston's unlikely choice of mascot). Even in those days, SIGGRAPH was huge, attracting upwards of 20,000 delegates from all four corners of the globe and the computing industry. It was there that I remember Al Gore, then a humble senator, now vice-president of the USA, opening the event with a live-by-satellite speech in which he talked of information "exploding in leaps and bounds". A wonderfully Moulinexed metaphor it may have been, but it accurately captured SIGGRAPH's transformation into one of the computing world's key events.

It was at SIGGRAPH 89 that people started talking excitedly about this newfangled virtual reality idea, and gazed with amazement and amusement at Jaron Lanier, the then fledgling VR industry's chief guru, doing his strange sort of jam session thing during what was otherwise supposed to be a serious technical conference.

These were the first tinglings of excitement that now seem to electrify SIGGRAPH every year as it becomes ever more firmly established as the venue for unveiling the most exciting ideas and developments in visual entertainment. At this year's conference, held in New Orleans, they came in their tens of thousands to get a peek at next year's movie effects, web content and games. Where Boston was full of bearded programmers and conceptual artists, New Orleans attracted the likes of Jeffrey Katzenberg, co-founder (with Stephen Spielberg and David Geffen) of the new computer-literate Hollywood studio, Dreamworks. It is also where you find a welter of new animation, including a strange little cartoon called "Chicken Crossing".

Finger-lickin' good...

Chicken Crossing was neither produced by DreamWorks nor any other studio. It came from Microsoft, a company that did not even attend Boston yet was out in full force in New Orleans. Although amusing enough as a work of entertainment, Microsoft's first attempt at a cartoon had the primary purpose of showing off "Talisman", a new technology being developed by the company's research division. This is, Microsoft states, "a new 3D graphics and multimedia hardware architecture" and if Chicken Crossing is anything to go by, it's the first sign that decent real-time 3D graphics may at last find their way onto the home PC.

First, let us consider what we mean by real-time graphics. In a 3D game like, say, Myst, or a movie with 3D graphic effects such as Twister (*see later*), the computer-generated images you see take hours, sometimes even days, to produce. So, obviously, they have to be done in advance. As a result, a game like Myst cannot strictly be 3D. Rather, it is a slideshow of 2D images with various puzzles determining the order in which they are seen.

A game like Doom is very different, because as you wander around those interminable tunnels (I am not a fan) the images are more or less generated from scratch as you go. This is necessary if the game is to allow you to roam freely through the artificial world it is trying to recreate, because to pre-render and store each possible scene as witnessed from every possible point of view would require impractical quantities of rendering time and storage capacity. Games like Doom deploy

Fig 1 **A particle system in action**

Fig 2 **Three stills from Chicken Crossing**

a special set of graphics tools (known as APIs) which use a variety of nifty shortcuts and compromises to generate each image as and when it is needed.

Several APIs have been developed for this task, one of the best known of which, Reality Lab, was created by a British company called RenderMorphics. Like so many innovative British high-tech companies, RenderMorphics was snapped up by Microsoft which used Reality Lab as the basis for Direct3D, which itself is a subset of a whole library of APIs designed for multimedia content, called DirectX.

DirectX provides the software layer for the Talisman architecture, and Chicken Crossing was supposed to demonstrate what the two could achieve, in combination. According to Microsoft, a Pentium PC with Talisman hardware (which should only cost two or three hundred dollars) could render and display each of the frames you see in Fig 2 and the 6,997 that made up the rest of the Chicken Crossing animation, in the time it takes for the screen to refresh (in other words, around one 75th of a second). This is an astonishing claim, given the richness of the textures and the number of objects: way beyond anything currently achievable on a Pentium system, even one with hardware acceleration.

In an extremely technical paper presented to SIGGRAPH, Microsoft explained how this impressive trick could be pulled off. Talisman, like any graphics technology, works by making compromises, the most important of which is layering. Most 3D scenes are rendered as true three-dimensional spaces, with the shading of each element of the scene calculated according to its position and orientation with respect to the rendered point of view. Talisman instead associates particular objects in a scene with particular layers, and then decides how much work needs to go into rendering each layer. So, for example, a layer comprising an object disappearing into the distance needs very little render time at all. Indeed, the effect can be reproduced in a 2D rather than 3D scene by scaling down the 2D image of the object as it recedes.

On the face of it, this is a clever solution, although how smart Talisman-based software will be when it comes to deciding how to handle layers, remains to be seen.

Another compromise is one that sounds rather obvious, even low-tech. It is graphics compression. For various technical reasons to do with the way a scene is calculated, compression is difficult to achieve with conventional renderers. With Talisman, the scene is rendered in blocks 32 x 32 pixels in size (the process carries the unglamorous name of chunking), which can be compressed using the same sorts of techniques used by the JPEG graphics format.

Microsoft says it will not be making Talisman boards, but will license the detailed "reference" design to hardware manufacturers. The company claims that because the design of the silicon is relatively simple and because many of the main components will be standard parts, boards should retail for less than $300. If this is the case, then that really should set the cat among the crossing chickens.

Particles

Summer is about blockbusters, and nowadays blockbusters are about showing off the latest computer graphics effects. Some of the most impressive were to be found in Twister, a movie about tornadoes. In my opinion, the computer-generated tornadoes were the most realistic feature of the whole movie (far more realistic than the characters) and I began to wonder about how they might have been produced. With the help, it turned out, of the resources of Industrial Light and Magic, AliasWavefront, several very pricey plug-ins and about 20,000 lines of customised code.

Having returned home, I tried to brew up a tornado for myself. Naturally I failed (it looked like an upturned tree trunk) but I did manage a smoking chimney (Fig 1).

The key to such effects is a set of 3D tools called particle systems. These are not yet to be found in cheaper 3D packages but they should trickle down into future releases. There are a number you can buy as plug-ins for mid-range programs: for Lightwave, for instance, you can buy products like Particle Storm for about £300.

I used a 3D Studio Release 4 plug-in called "Vapor" to produce the smoking chimney. It is an unexceptional effect but, believe me, it was not easy to create. All particle systems make enormous demands on the processor, not least because being effects that develop over time, they have to be calculated for each frame of an animation. This means that until you render the animation, which can take ages, you cannot really judge whether you have correctly captured the dynamics of your smoke trail or twisting tornado.

The key to all particle effects is a special class of objects called "emitters". These emit a series of smaller objects (the particles) that are generated at a particular rate and disperse in a particular direction, in a particular formation, at a particular speed.

The Vapor plug-in comes with a series of presets for producing different types of smoke, from a cigarette trail to a steam locomotive's billowing clouds. The latter was not particularly convincing, so I had to fiddle around with the parameters to achieve the effect seen in Fig 1 (which, I hope you will appreciate, looks a lot better when animated). Each change to the size and intensity of such parameters (the "whorl" and "turbulence") produced rather unpredictable results, so it took a good few goes, and hours of rendering time, to tune the effect. It just goes to show that there is no smoke, and no tornado, without toil.

PCW Contacts

Benjamin Woolley, writer and broadcaster, can be contacted at **woolley@illumin.co.uk**. His home page is **www.illumin.co.uk/woolley/**