Physical jerks

Stretch it, warp it, wobble it or bounce it. Benjamin Woolley explains how to add solid physical characteristics to objects by using HyperMatter, a simple plug-in to 3DStudio MAX.

Physicists like to boast that the laws of nature apply across the entire universe. If we could fly to the furthest galaxy, we would find gravity exercising the same pull as it does on earth.

But such laws do not extend to the virtual universe. In the realm of 3D graphics there is no gravity, no mass, no momentum and no friction. Scenes comprise nothing but geometry, the models that inhabit them being nothing more substantial than surfaces which are knitted together in such a way as to describe a recognisable shape.

That shape may be given the appearance of solidity by being clothed in a texture. It may even be a texture (a "procedural" one) that in some way mimics a physical material. For example, most 3D graphics packages now come with procedural textures for mimicking wood and marble, which reproduce the sort of grain or veins that would be seen in an object sculpted out of those materials.

Zero gravity

But even the most realistically-rendered marble column or wooden table has only the visual features of the object it is supposed to represent: it will have none of the physical qualities. If your column and table stood on a sloping surface, the column would not topple nor the table slide. The force of gravity means nothing.

Usually, if you are rendering still images, you can either simulate or ignore such forces. However, if you are animating objects, the lack of physics can create real difficulties. Even the most simple of behaviours are difficult to reproduce. Objects unexpectedly float, pass through other objects, drift off to infinity and so on. The scene quickly transforms itself into a maelstrom of mad flying objects.

Furthermore, trying to mimic even the simplest dynamics such as an appropriate bounce for a ball textured to look like it is made of rubber, can verge on the impossible. You not only have to get the bounce right in terms of changes of velocity and trajectory, but you also have to deform the ball as it hits the ground.

Tweaking, squashing and morphing

You can tweak the velocity and the trajectory by editing the path and moving the position of key frames. You can deform the ball by creating two identical versions, squashing one up and then morphing between the two. But with each change you must re-render the animation to see if you are getting closer to the desired result, which makes the process very laborious. And unless you are prepared to create a key for every frame of the animation, the dynamics will inevitably look artificial.

Things are about to change. Physics has entered the 3D graphics world. The leading mid-market packages, Truespace and Ray Dream Studio, now come equipped with tools for adding physical properties to objects. I have also been trying out a plug-in for 3D Studio MAX called HyperMatter, which offers one of the most sophisticated so-called "physics engines" for reproducing physical phenomena.

You can get a form of physics in cheaper software. VRML 2.0, the standard modelling language of the internet, includes facilities for reproducing that most fundamental physical property of solid objects, impenetrability. By using the collision detection parameters, you can at least prevent objects and avatars from walking through walls and dropping through floors — a common occurrence in VRML 1.0 worlds.

Truespace and Ray Dream Studio offer a lot more than collision detection. They boast the ability to simulate gravity, elasticity, density and, in Truespace's case, "torque" for any selected object in any given scene. I haven't yet managed to spend enough time with either of these packages to be able to assess how well the physics work, but I have spent some time with HyperMatter, an intriguing plug-in for 3D Studio MAX from Second Nature Industries.

Both DOS and Windows versions of 3D Studio have been disappointingly lacking on the physics front, so HyperMatter is a welcome enhancement, if rather expensive at £485 (which you have to pay over and above the cost of 3D Studio MAX). What you get is a set of tools which plug straight into MAX's rather cluttered interface. The basic principle to using HyperMatter is simple: you select an object and press a button which "solidifies" it. This creates a new object, identical in size and shape, but with physical attributes.

Falling over

Physics, as it turns out from using tools like HyperMatter, is not all that simple, as you soon discover when you begin to play with solidified objects. To start with, they do not do what you instruct them to do. But then, this is inevitable, because forces like gravity will have an influence over how the object moves. So when you solidify an object which you are animating, the first thing you find when you come to play the animation is that it falls — and unless you have created

Benjamin's book review — *3D Graphics & Animation*

I cut my 3D teeth on a book called *Inside 3D Studio*, and I approached its UK distributor, Prentice Hall, for review copies of some of its other titles. A few phone calls later, what seemed like a skipload of breezeblocks tipped into my office: an extensive library of hefty volumes about various aspects of 2D and 3D graphics. The image alongside resulted from working through the only book not to focus on one particular graphics package: *3D Graphics & Animation* by Mark Giambruno.

It is a useful, intelligent, introduction to general principles, and includes tips (unfortunately, US-orientated but nevertheless useful) on building a portfolio and getting a job. It's expensive at £42, but that includes a CD-ROM and some detailed tutorials. It would be particularly useful to a novice with ambitions to become a pro, especially one who has managed to blag an old copy of 3D Studio.

another object to act as the ground, it keeps on falling. This is because the default "substance" from which a solidified object is made, is one with weight.

The second confusion is that physical forces are features of the object, not the environment. You do not switch gravity "on" and watch everything start to sink. Unsolidified objects remain as gravity-free as before. They also remain as penetrable, with even solidified objects passing through them as if they were not there.

The reason for allowing you to create a world in which physics is both present and absent is twofold. Firstly, you may want to break the laws of physics once in a while (that, after all, is one of the freedoms computers allow). Secondly, the "physics engine", which has the job of calculating the interactions and dynamics of each solidified object, is a guzzler: it takes up huge quantities of processor cycles. You can easily double the time it takes to create even a preview of your animation, so you need to add only the physics you need.

Mind you, once you start adding physics to a scene, it is hard to stop — this tool is enormous fun. Dead objects come to life as you start to make them wobble, warp, sag, ripple and flop. HyperMatter includes a library of preset substances (with wonderful names like "Water Bomb") but all the parameters are editable, so you can create just about any substance you can imagine. These parameters include:

- elasticity;
- damping, the degree to which a solid object resists changes to its shape;
- compressibility, the degree to which an object loses volume as it is compressed; and
- friction and density, which determines the object's effect when it hits another: the denser the object, the greater its influence over the collision's outcome.

You can also set constraints on objects, the most important being the collision restraint (for collision detection).

Size matters

There are subtleties involved in adding physics to a scene, which you come to appreciate only when you begin to grapple with the technology. One such subtlety is the importance of size, and what it really means. In the real world, two objects of identical shape but different sizes will behave differently. For instance, while a small ball made of soft rubber will keep its shape, a large one will tend to sag under its own weight. HyperMatter simulates such differences by allowing you to adjust the degree and speed at which certain forces are applied.

It soon becomes apparent that adding physics does not necessarily add realism to a scene; sometimes it has the opposite effect, allowing you to make teapots of rubber and mice of jelly and see what happens when, using a cannon, you shoot them at a wall. However, even if an animation becomes less realistic, it also becomes less artificial. The dynamics are no longer so rigid and uniform, the shapes so static and flat. For these reasons alone, the introduction of solidity to 3D graphics is welcome, and I hope it will not be long before physics is as ubiquitous in the virtual universe as it is in the physical.

PCW Contacts

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

HyperMatter from Second Nature Industries
www.2n.com
Prentice Hall 01442 881900

Star quality

Don't neglect post-production processes if you want to get professional-looking results with your animations, warns Benjamin Woolley. Even budget 3D packages offer basic facilities.

In past columns I have provided an outline of the principles of animation. So let's take a look at one aspect of creating both still and animated images, that is somewhat neglected yet nevertheless crucial to achieving professional results. It is called "post-production" (or "post-processing") and embraces all the things you can do to an image after you have rendered it.

Most of us assume that once the renderer has done its job and the picture is on-screen, our work is finished. But if you look at any professionally produced computer-generated imagery, be it a game or a special effects-laden Hollywood blockbuster, you will see that most have been exposed to the post-production process. In other words, the computer-generated image has been combined with other images to produce the final effect.

An obvious example is a scene in a film like Jurassic Park, where the computer-generated dinosaur has been placed inside a live-action sequence shot in a studio. A less obvious example can be found in the game, Myst, where in the opening frames you see a distant gull soaring over the computer-generated sea.

Compositing

This, then, is what post-production or post-processing is mostly about: combining or "compositing" several images to form one (although there are other, allied effects that can be added at this stage, such as motion blur and fades). Most workstation-class 3D graphics packages include a range of tools to deal with this process but you are unlikely to find them in current budget 3D packages. However, even these now offer the basic facilities needed to achieve basic compositing.

Compositing is essentially about layering 2D images in a way that takes into account their contents. For example, if you want to combine a picture of a city with an animation of a dinosaur stomping through it, you need to create a mask of the buildings in the foreground of the picture so that the dinosaur passes behind them as it walks by. If you want to show a computer-generated glass object in front of a photographic background, you will want the background to show through the object according to the transparency of the glass.

"G" force

The jargon you need to understand to achieve this sort of effect, and the technology that underlies most computer-based post-production work, is the "G-buffer" and the "alpha channel".

As most of us know, a picture stored on a computer comprises "pixels"; the number depending on the picture's resolution. A 640 x 480 image has 480 rows, each containing 640 pixels. Each pixel is of a particular colour and that colour is determined by a number. In an 8-bit colour image, the number, comprising eight binary digits, determines which of 256 colours in a palette or "colour look-up table" is to be applied to that pixel. A "true colour" image uses 24 bits per pixel: eight for each of the primary colours (red, green and blue; the three colour "channels") which, when mixed together, can recreate any colour in the visible spectrum.

Image files do not stop there. Certain file formats, notably the TGA and TIFF file formats that you often find in the world of 3D graphics, allow 32 bits per pixel. The extra eight bits make up the image's so-called "alpha channel". This means each pixel can not only be one of 16,777,216 different colours, but it can also be one of 256 possible degrees of transparency, from 0 (fully transparent and therefore invisible) to 255 (opaque).

But some image files can go further still. They can have an entire G-buffer (G for graphics) comprising a series of additional channels. These might store data about the distance of each point in the scene from the camera viewpoint, or an object index which relates each pixel to the particular point on the object, in the original scene it depicts. Unfortunately, there is no standard for determining which type of information is put into which channel in a G-buffer: it is up to you and the image-editing or post-production software you are using.

Using a four-channel file

So let us concentrate on the basic four-channel file: R, G, B and alpha. How can this be used? If you are using a package like Ray Dream or TrueSpace, there are no tools included which are specifically designed to help you exploit the alpha channel. However, if you output the rendered image to a 32-bit format such as TIFF or TGA (and, more recently, the GIF substitute PNG) the software will generate an alpha channel automatically, which can then be used with any image-manipulation program that supports alpha channels — including shareware wonders like PaintShop Pro, as well as top-drawer products like Adobe Photoshop.

Once you have imported your 32-bit file, you can use the alpha channel to create a mask, or stencil, to "composit" with another file or add various filter or paint effects. For example, you could paint over the image, with the paint being excluded from the area masked (or with the paint only appearing in the masked area); or you could use the mask to mix two images together.

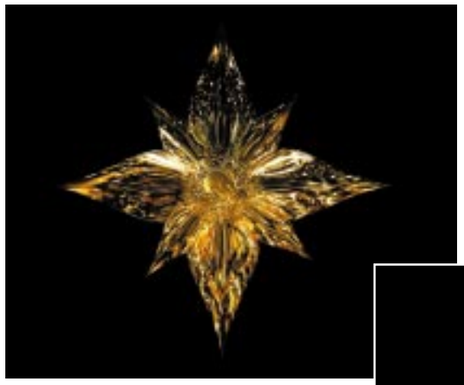


Fig 1 (left) **The element to be overlaid on the background: a translucent gold star**

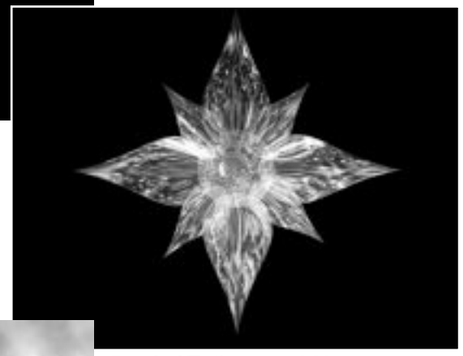


Fig 2 (right) **The same image as Fig 1 but with the colour channels switched off, showing only the remaining alpha channel**

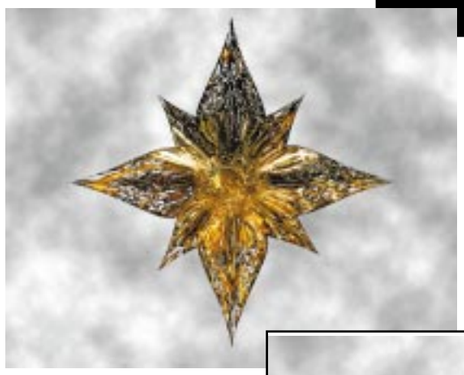


Fig 3 (left) **The image overlaid on a background without using the alpha channel; note the sharp edges**

Fig 4 (right) **The same image overlaid on the same background using the alpha channel**

Unlike a physical stencil, the masking is not total: its strength varies according to the alpha setting (zero to 255) for each pixel. This offers two advantages. Firstly, if you composit, say, a translucent gold-flecked star over a grey cloudscape (odd thing to do, but that's the example I have created!) the background shows through the foreground according to the transparency settings of the foreground object's material (Fig 1). The second advantage is "anti-aliasing": the edges of the foreground object are softened by making the pixels opaque, so they blend seamlessly with the background. You should be able to see this by comparing Figs 3 & 4.

At the moment, the alpha channel is just about the only post-processing facility available to most 3D enthusiasts without access to professional facilities. Unless you have the patience to individually edit hundreds of huge TGA or TIFF files, you are confined to working with stills rather than animations — that is, until the post-processing facilities already available in upscale packages begin to migrate downmarket, which shouldn't take too long.

Let's get physical

This process of workstation-class tools migrating to the PC is turning into an exodus. Recently announced revisions of the mid-range packages, Truespace and Ray Dream Studio, are festooned with sophisticated facilities. Truespace 3, for instance, now boasts "metaballs", a particle system that allows you to move the particles around beneath an elastic skin to create organic shapes. Ray Dream Studio 5 offers ThinkFish's LiveStyles rendering system (*see PCW, August*). Truespace also boasts 3D painting (claiming to allow you to paint 3D objects in a scene in real time, rather than indirectly via texturing) and this is not even standard on 3D Studio MAX.

However, the producers of heavyweight professional packages are not standing still. Kinetix has unveiled version 2 of 3D Studio MAX. The new version includes important additions, notably NURBS, which allows you to model smooth, curved surfaces; support for the industry-standard Open GL graphics API, and for DirectX; a ray-tracing renderer which can be applied selectively to objects in a scene; a redesigned materials editor which allows drag-and-drop; and a bigger, more adaptable variety of sample swatches.

One of the themes common to all these upgrades is the introduction of a great deal more "physics"; in other words, objects can be made to behave (both as you build them and as you animate them) as though they have physical properties such as mass or elasticity. This is a significant development, and I eagerly await the chance to put it to the test.

PCW Contact

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

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.

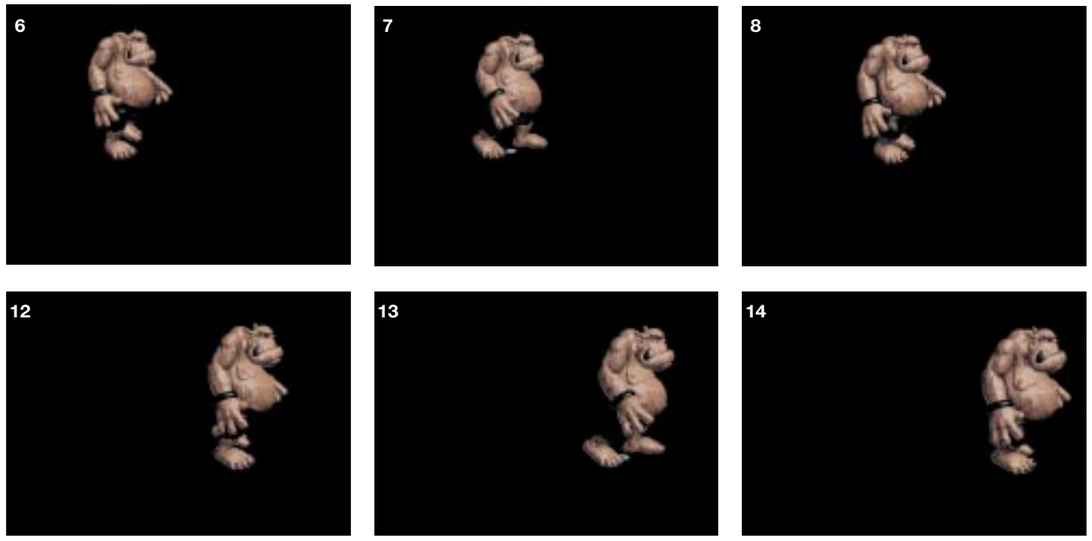
Character animation is difficult; perhaps the most challenging aspect of computer graphics. A spaceship zooming through the star-spangled firmament, or a car driving down a canyon of towerblocks: these are, in animation terms, easy to create, as the objects are moving at even speeds along straight lines 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

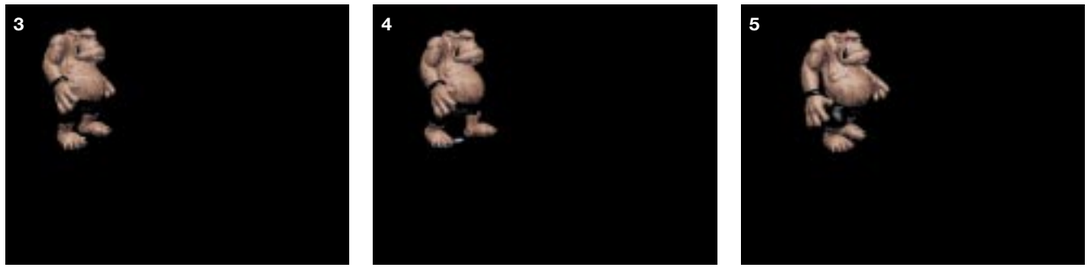
Once you have an idea of what your character will look like and how it will move, you can proceed with the task of modelling and animating it. Before you begin, though, you have one more important decision to make: how much of the animation you want to achieve through the model itself, and how much through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



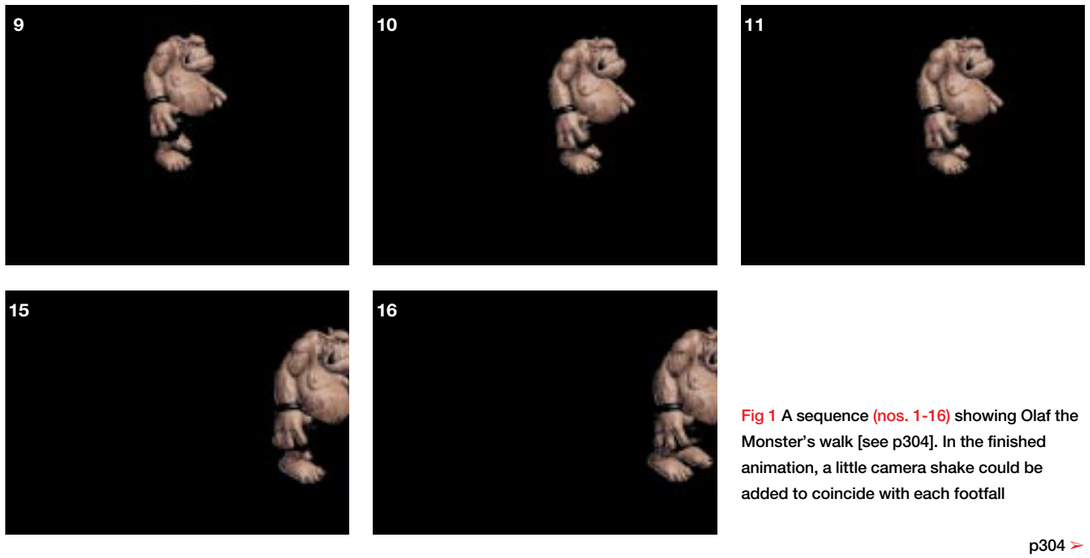


Fig 1 A sequence (nos. 1-16) showing Olaf the Monster's walk [see p304]. In the finished animation, a little camera shake could be added to coincide with each footfall

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?

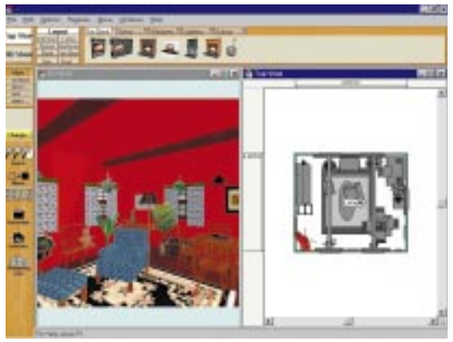
Having seen the funky ads on the television and read all the hype — and having talked to all the users who got new Pentium systems for Christmas and felt cheated when the MMX systems came out at the beginning of the year — I decided to have a go at finding out what real difference MMX would make to 3D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

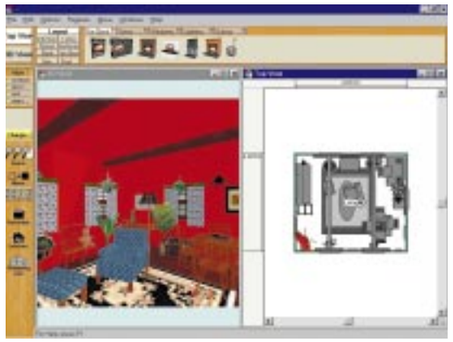
Having seen the funky ads on the television and read all the hype — and having talked to all the users who got new Pentium systems for Christmas and felt cheated when the MMX systems came out at the beginning of the year — I decided to have a go at finding out what real difference MMX would make to 3D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

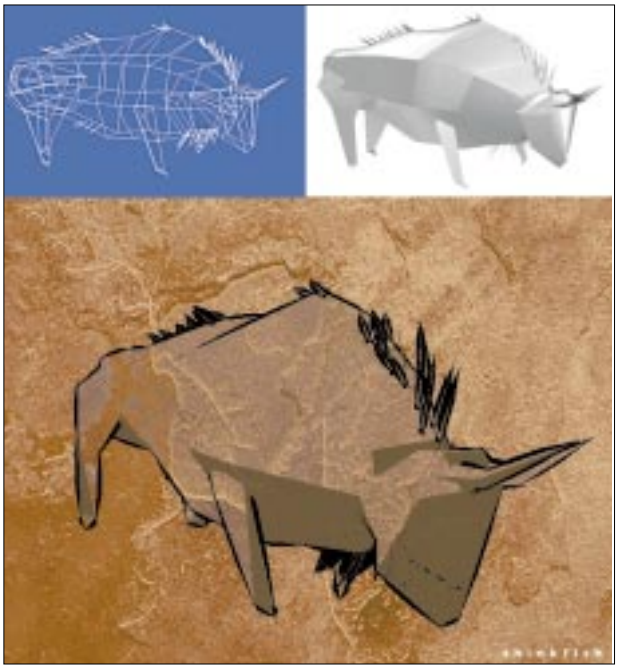


Fig 1 The ThinkFish "intelligent" renderer at work, picking out the key lines from a simple mesh

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 ➢

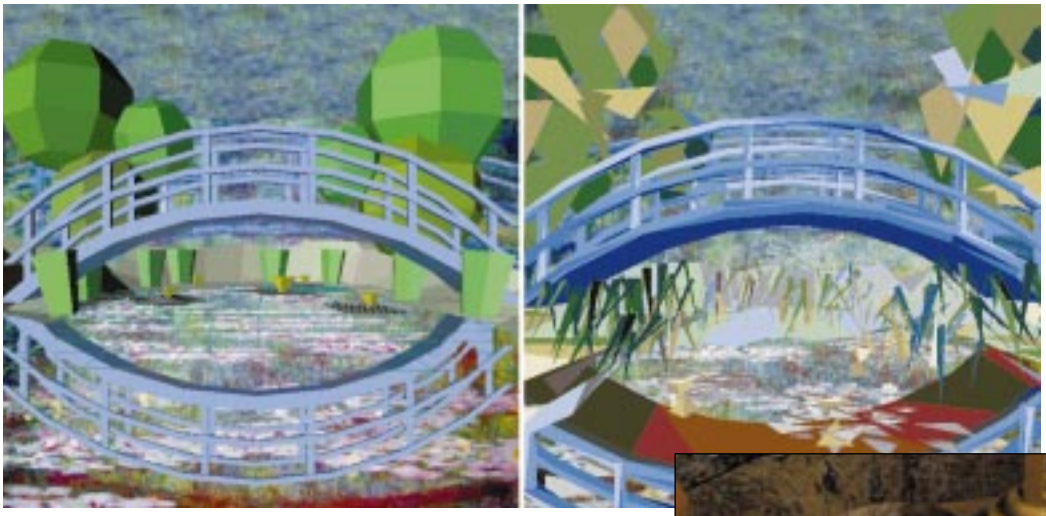
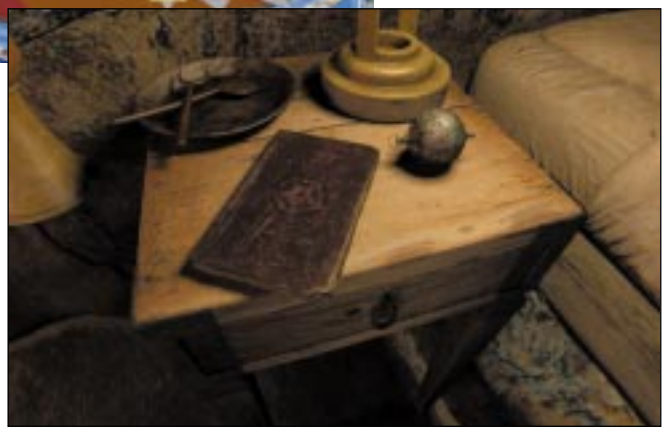


Fig 2 (left) Examples of the same scene in two different "LiveStyles"

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.

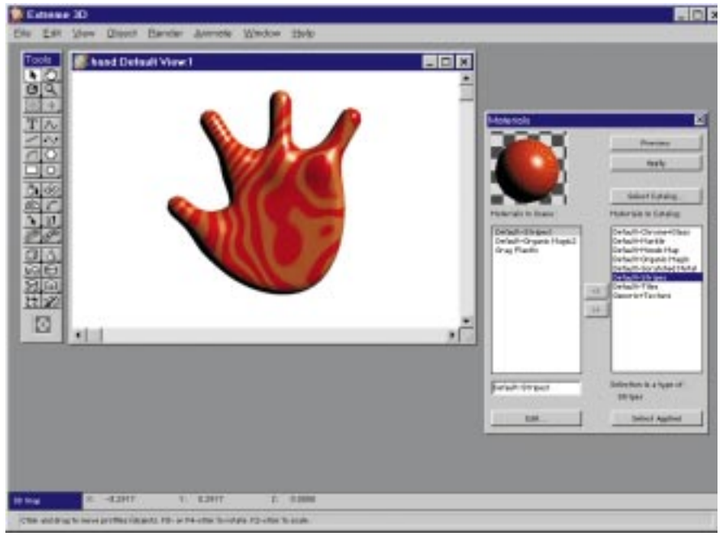


Fig 1 **Extreme 3D's metaform used to create a balloon hand**

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

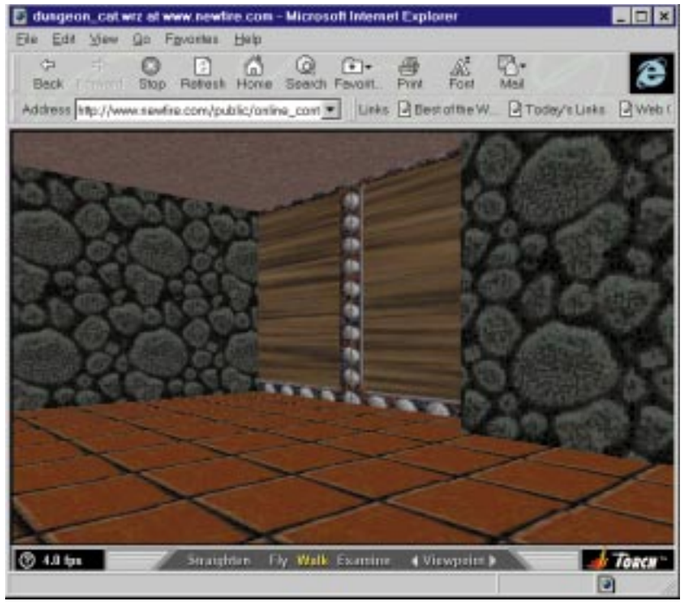
Extreme 3D from Macromedia is an important product. At around £500, it is priced substantially below full-blown professional packages and its specification now almost equals many of them. The recently-shipped version 2.0 now comes with a particle system and an implementation of a modelling method generally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

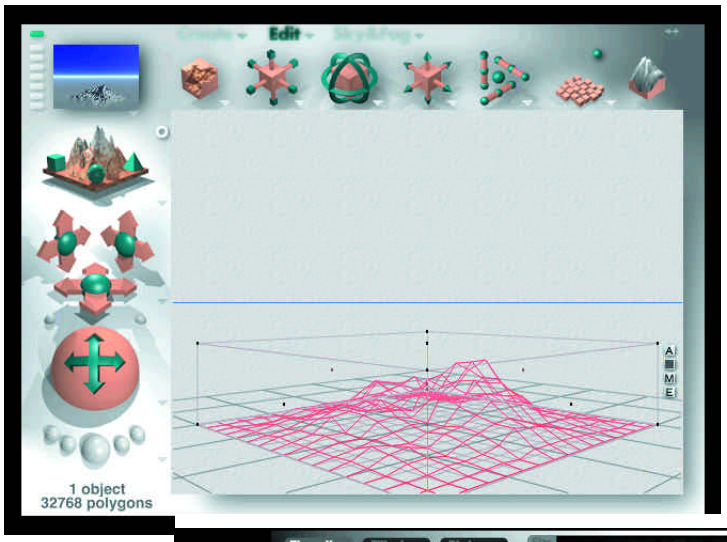


Fig 1 (left) Bryce 2's artistic interface

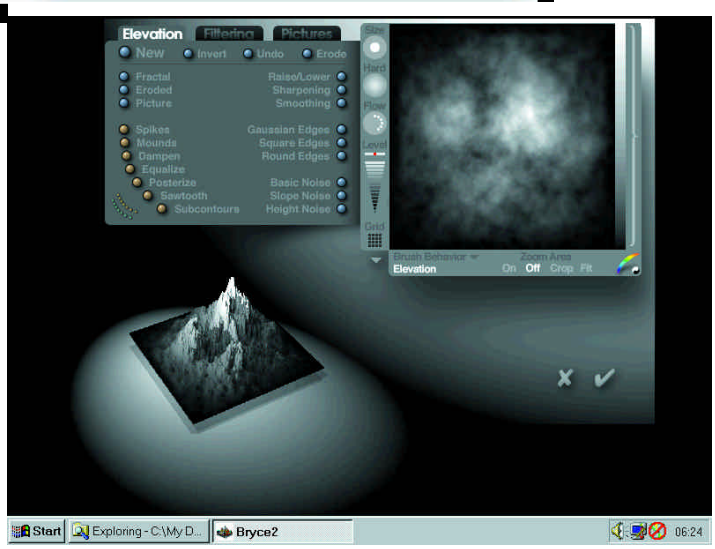


Fig 2 (below) Bryce 2's "terrain editor"

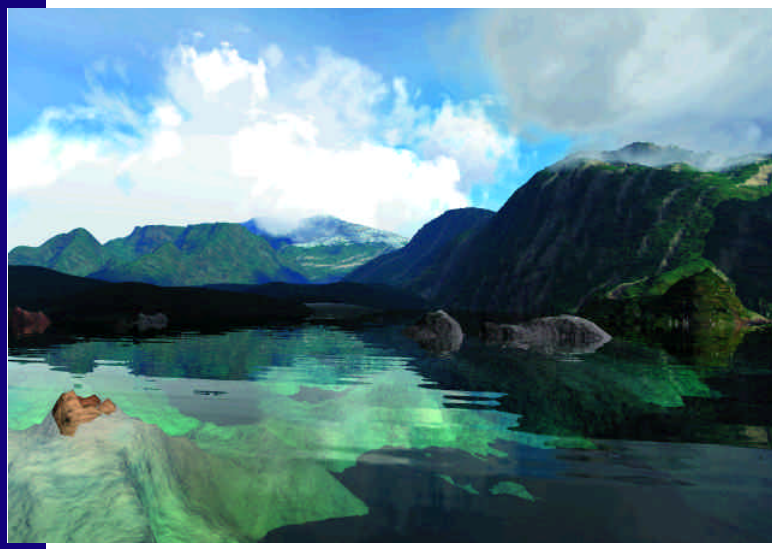


Fig 3 One of the samples supplied with Bryce 2. It's called Scotland, and is the work of Kai Krause, Metatools' resident guru

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.)

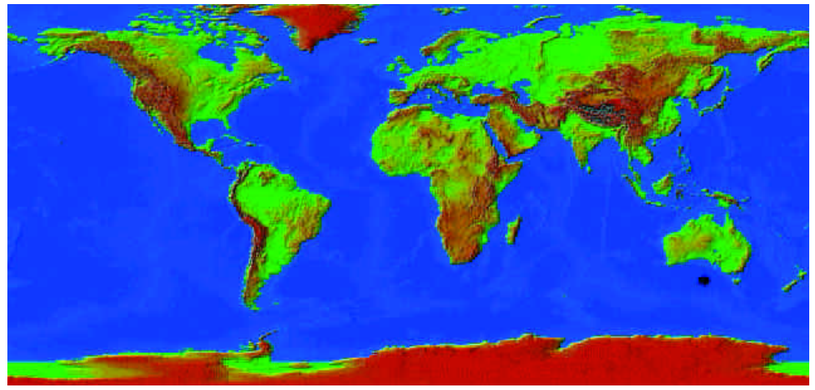


Fig 1 A texture map of the world. Note Greenland's size relative to China

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.

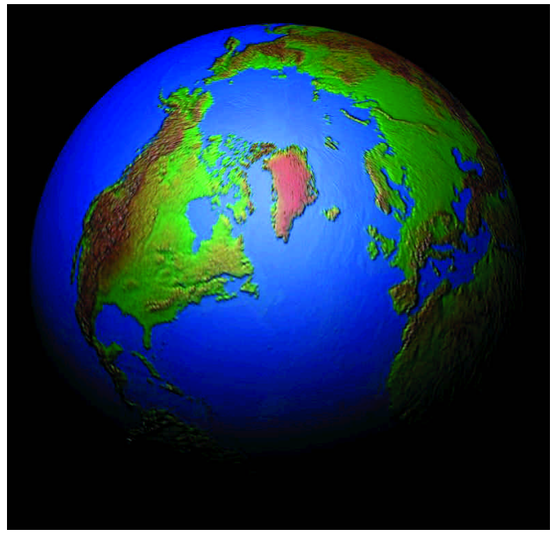


Fig 2 The texture map in Fig 1 wrapped round a sphere. Greenland assumes its proper proportions

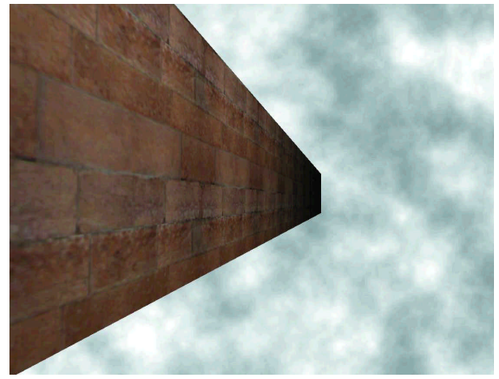


Fig 3 The purpose of this rather surreal image is to show a texture map being stretched beyond its resolution. Note the blurring where the wall is closest to our point of view

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