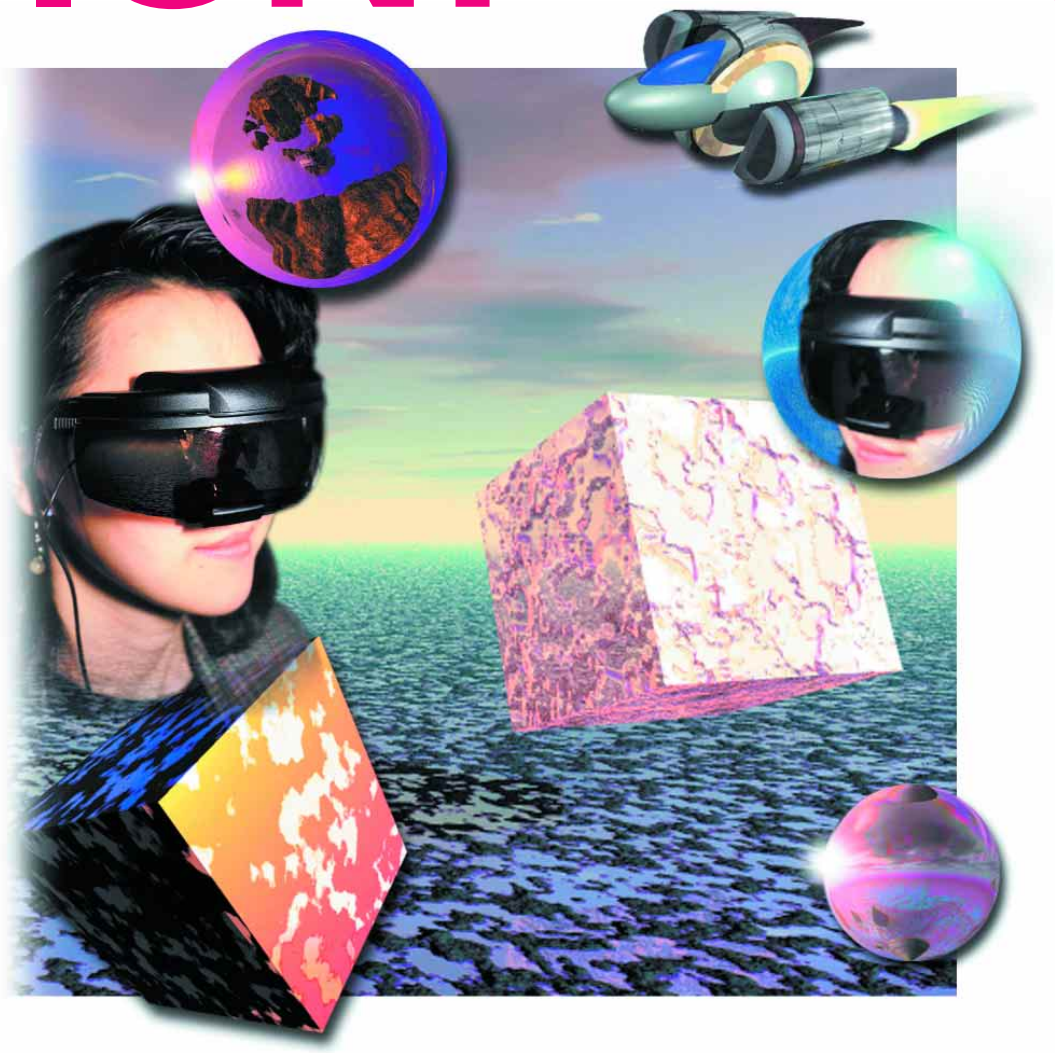
Creating and Rendering 3D Models

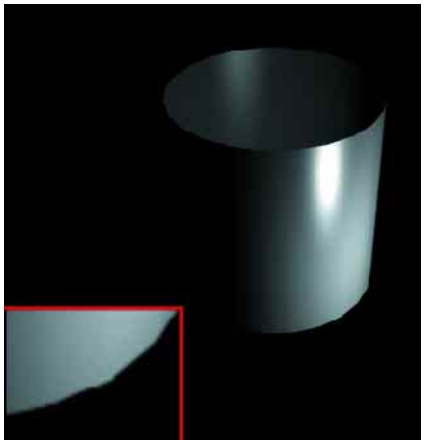
LIGHTS! CAMERA! ACTION!

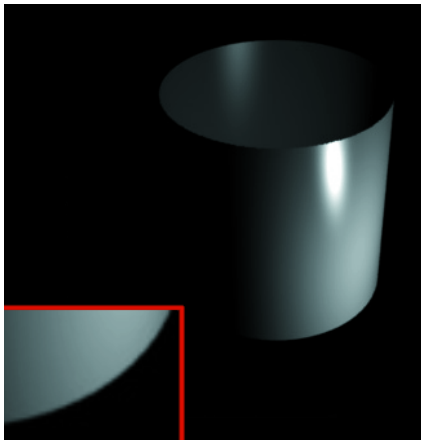


Many people find creating their own three-dimensional worlds a fascinating idea. It's no wonder, then, that there are numerous high-performance tools for modelling and rendering that run under Linux and require no professional experience. Manuel Lautenschlager sheds some light

Anyone who talks about 3D modelling usually thinks of powerful workstations from SGI or clusters of a few hundred computers sharing the load. And Linux has seen a great deal of use running these kind of systems. But someone who wants to open up virtual worlds on their home PC can use the free operating system too, allowing at least a taster of virtual reality with no worries and without having to pay a penny.

3D programs perform two tasks: they have to create three-dimensional scenes and models, and they have to render these scenes (computing the 'coloured in' three-dimensional image from the models). The two tasks are fundamentally different and also impose different demands on the hardware. This is why professionals use separate programs. A modeller program on a graphics workstation is used to construct scenes and a software





[left] Comparison of render quality: a cylinder, first processed with an integrated renderer (Maya Version 2.3) ...

[right] ... and again with a dedicated program (Blue Moon Rendering Tools Version 2.5). The difference in quality is clear, especially on the edges.

package called a renderer, specially developed for the purpose, builds up an image or an animation from the descriptive data.

Rendering is an enormously calculation-intensive process and takes a lot of time on low-powered machines. For demanding tasks, such as the work required for the film Titanic, entire clusters of machines were put together, forming so-called render farms. The virtual ship in Titanic came about with the aid of a specially written modelling program and was rendered on a cluster of 160 Alpha computers, 55 of them using Windows NT and 105 of them running Linux.

Because artists usually like to get an idea, while still drawing, of roughly how the finished scene will look, all current modeller programs include a render engine. The results of these come nowhere near those of dedicated renderers but are adequate for home use.

Layers and Filters

In the case of large projects, on the other hand, studios push dedicated program development even further. It's fairly common to have each part of an area, such as the water or the landscape, calculated by specialised programs. They're all then edited into a single image at the end. In order to be able to perform this task, data also has to be stored with an extra characteristic which isn't normally contained in a normal, two-dimensional image – depth data in the form of the Z-Buffer. This indicates how far the point shown is from the viewer. Each pixel of the image holds one of these values, which enables different scenes to be matched-up on the 3D stage.

Instead of depth data, "alpha layers" are often used, which can make an image transparent wherever you want. So for example a program which specialises in water areas and landscapes such as arete Digital Nature (http://www.areteis.com/) can render the ocean, but mask the Titanic ship with an alpha layer so that it can be produced by a different program.

Advantages of such layers include specific effects, which can also be masked by an alpha layer. Then a normal image-processing program like Gimp can add the effect again after rendering. This process is applied at the point when computer-generated images are to be combined with real filmed scenes, since it allows the light conditions to be taken into account as well.

The basic principles are the same in all modelling programs. The main differences are in user-friendliness and in the range of functions. Almost anyone can render their own movie and get up and running using free tools.

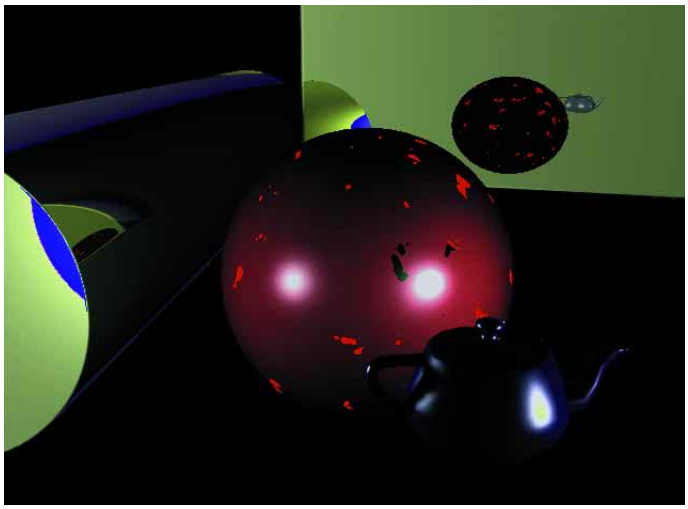
Blender

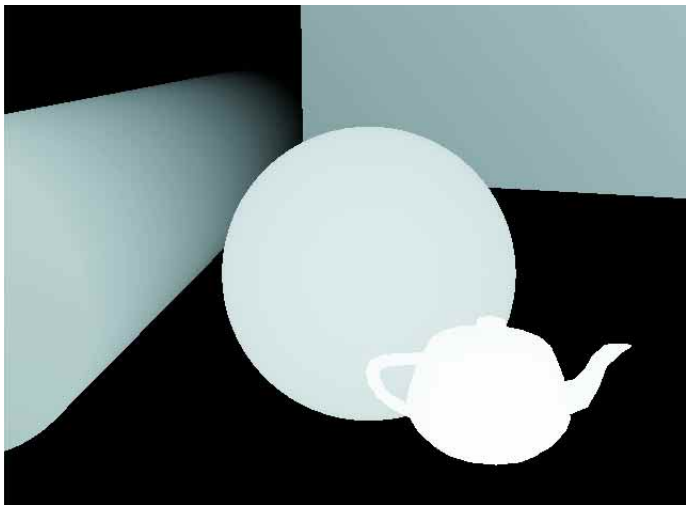
Blender is perhaps the most popular modelling program under Linux. It is a commercial program. There is a free useable version with limited functionality. This can be turned into the full version on payment of a fee of £57. The program is fully scriptable with Python, controlling radiosity, environment mapping and other advanced functions.

There are numerous plug-ins, some good tutorials and a lively user group, which recently met at the offices of NotANumber (the company behind the program) for a one-week conference. The program originally stems from the Commodore Amiga, believe it or not, but has since been ported onto a number of platforms.

[left] Here is a normal two-dimensional image...

[right] ... and the depth data on it.





Interface

The interface shows signs of its Amiga history. At first glance it looks a bit crowded and jumbled. There are no menus. Instead, you must press certain buttons to display tools and alternative work screens. It isn't immediately apparent which buttons do what.

In the full version the whole interface can be customised using a Python script. This effectively lets the program be restyled and altered with a few commands in order to adapt it to the current task. In allowing this the program is following a popular trend because many high-end programs now have a similar option for workflow optimisation. Since the interface can use OpenGL, it is even possible to have 3D elements in it.

As mentioned, Blender doesn't actually provide any standard menu structures. However, the interface can be made more interactive than in many more expensive programs. Anyone who can get the hang of adapting the workspace will be able to work surprisingly quickly with this program.

Use

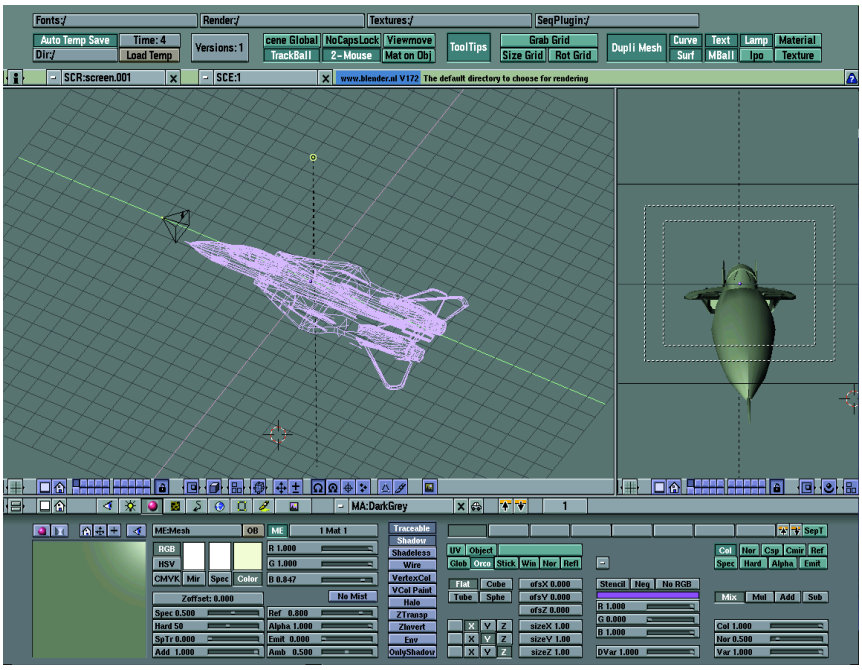
The numeric keypad and the space bar are important controls. The viewer can use the number keys to change the view of the scene, such as "walking" or "flying" around it. When the space bar is pressed a menu, containing important functions such as creating objects and storing scenarios, pops up under the mouse cursor. The buttons in the control window then allow fine adjustments. This space bar menu is also found in Maya.

By pressing [Tab] you can toggle back and forth between Edit and Move mode (or rather Transformation mode). No matter what mode you are in you can always move the object. The crucial difference is that the centre point of the object in Move mode is shifted along with it. This doesn't happen in Edit mode.

A scene from the Film: "The Fifth Element": The surface of the water was rendered with arete and the ship with Maya.

Processing

The modelling process, like the interface, can take some getting used to. Functions like "move", "rotate" and "scale" can be activated not only using keystrokes but via drawing certain symbols in the viewing window. So drawing a rotated L switches into rotate mode and a pointed V switches into scaling mode. Mouse users might not think much of this but it will delight users with a graphics tablet. However, mouse users aren't left in the cold – by pressing the mouse button simultaneously with "Ctrl" and/or "Shift", the view can be turned, moved and zoomed.



The interface can be customised in almost any way but unfortunately it's not an intuitive procedure

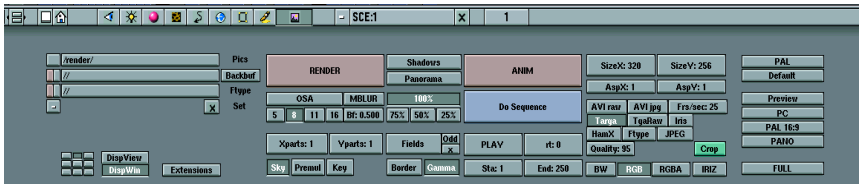
Starting Blender properly

Anyone using a window manager with several virtual image screens (not to be confused with "multiple desktops" which are found for example in KDE) will come across a problem with Blender when switching to another screen. Blender simply crashes after a few minutes of messing around this way because it is not allowed to draw directly on the visible screen. For this reason, it is best to run Blender on an X-Server of its own - the whole thing can be automated as follows:

```
#!/bin/bash

set -x MAXDSP=$(ls /tmp/.X*-lock|tail ➥
-1|cut -c 8)
NEWDSP="$(( $MAXDSP + 1 ))"
DISPLAY=":$NEWDSP"
X $DISPLAY &
while [ ! -f /tmp/.X"$NEWDSP"-lock ] ; ➥
do sleep 1 ; done
exec /usr/local/bin/blender
```

This script, however, only functions correctly if the server is running as "root", so either the suid bit has to be set or an Xwrapper mechanism must be installed.



Some user interfaces in Blender appear truly chaotic; these are the rendering options.

The main functions can once again be quickly called up using the space bar menu. Additionally there are aids such as the 3D cursor, which can define the creation position. This can prove itself highly useful. For any view it is possible to define five levels of detail at which the processing view should be rendered or displayed respectively.

Glossary

Modelling	
Primitive	The most basic elements of a model. All other, more complex, objects are assembled from these.
Nurbs	A Nurbs object is not described by polygons but by Bezier curves. This makes the surface appear more rounded.
Trims	Methods by which the holes in Nurbs areas can be cut.
Polygon	An object made out of triangles.
QuadFace /Quad-Poly	Two triangular polygons which are combined into a quadrangle. The objects resulting from this are required for clean subdivision surfaces.
Metaballs	Objects made up from balls that merge into each other. The properties of Metaballs can be weighted.
Pivot	The point of origin within the object matrix to which changes of translation, rotation and scaling relate.
Subdivision Surfaces	A new type of object for modelling in which a rudimentary control object such as a cube controls the actual object. Lines and points of this control object can also be weighted and thereby refine the appearance of the target object.
Construction History	The working steps by which an object was created can be reprocessed with this at a later stage; alterations to an object then have an effect on objects built using it.
Materials	
Shader	A generic term for render options. Shaders define and calculate the properties of materials.
Volumetric materials	Procedurally created material, comparable with mist from the point of view of properties. This enables the inside of an object (such as a cloud) to be realistic.
Weighted materials	Refers to the attraction or the effect of a point on an object.
Texture	A two-dimensional image which when imposed on a 3D-body via
Mapping	Term that describes wrapping a texture around an object. There are various types, the most important being a normal texture, when an object is to be coloured in or painted. Another type is called
Procedural textures	Textures produced with algorithms, such as shadings, wood and marble textures and textures produced with a noise filter.
Filters	Texture filters, similar to those in an image processing program; these are used to improve textures.
Animation	
Skeleton	Basic frame on which objects can be fixed. If the skeleton moves, the added objects also move accordingly.
Skinning	Geometry which is distorted with the aid of a basic skeleton.
Inverse kinematics	The end points of the skeleton can be freely moved and animated; the position of the joints is calculated by the program.
Forward cinematic	Each joint can be moved individually.
Path animation	A curve or line along which objects move.
Morph	Creates a smooth transition between objects; the control points of the objects are interpolated between various positions.
Motion capture	Capture of motion data using cameras and sensors placed over the clothing of actors. In this way, their movements can be digitised and a natural-looking movement of an animation is generated from this.
Keys	Key points which define the animation process. Keys can relate to all parameters.
Lights	
Point Light	Point-shaped source of light
Spotlight	Cone of light
Area Light	An certain area defined as a light source, such as a neon tube. Enables creation of soft ray-trace shadows
Shadow map	The shadow is calculated from the position of the light source. Its precision depends on the size of the Shadow map. This process never produces a 100% accurate result.
Raytrace Shadows	Each ray of light is traced back. In this way, the result is absolutely precise, but resultant shadows appear very hard.
Caustic Light	Extended light computation: the light can be concentrated as with a magnifying glass.
Rendering	
Diffuse to Diffuse Light transfer	A computing method in which coloured areas reflect coloured light.
Specular to Diffuse Light transfer	In general, this refers to the reflected rays of light such as the light reflected back from a mirror.
Radiosity	With this method of computing each object is assigned an energy value which is a function of the energy radiated by the surrounding object. This process, unlike Raytracing, produces very soft light conditions.
Raytracing	All rays of light are traced back to their source. This makes exact reflections possible with reflective surfaces and refractions in the case of transparent objects.
Oversampling	Refers to the number of render passes, where the quality of the rendered image improves with each additional pass.
Render farm / Distributed Rendering	Many computers, networked with each other, undertake a single computing task - this shortens the processing time accordingly

Modelling and animating

There are a large number of functions such as "Nurbs modelling", "Meshsmooth" (a rounded off and weighted Quad-Poly-Object) and Metaballs, which can create nicely rounded objects. The processing of Nurbs objects functions smoothly, but the successful creation of the Nurbs curves necessary for this does depend somewhat on the circumstances.

The important functions for moving images, like animation paths, keyframes, inverse kinematics and object hierarchies are integrated fully. The keys can be processed with the built-in graph editor. All other characteristics of the objects can be entered and edited at this point as Bezier curves.

Materials, colours and lights

There are only two shaders directly built into the program: a standard shader and the halo shader. Others are available as plug-ins, however, it isn't possible to nest the characteristics of entire materials. Instead, you get an option for combining various textures. To apply texture to a frame there are several mapping types including bump maps (which can to a certain extent "dent" the object, in order to give it some structure.) Other mapping options affect the transparency of an element (the "Opacity" button) or its luminosity ("Self Illum".)

The program provides the most important procedural textures such as wood, clouds, noise and colour shading right from the start. There aren't any volume maps at this stage (these can, for example, make clouds look more real from the inside.) In order to refine the textures, filters loaded as bitmaps can be used on the images.

So-called vertex colours can be used in Blender to paint directly onto the object with a brush. This works in a similar way to Artisan in Maya, but isn't quite as intuitive to use. All textures, including the procedural ones, are then also displayed in the modelling view.

There are two types of light sources: point-shaped lights and spotlights. The standard options such as shadows and colours of lamps can be set separately. Effects like halos, rings and so on can be set in the material editor. The light conditions can also be previewed in the OpenGL-view.

Rendering

Rendering is done very quickly. The renderer is capable of 16 times oversampling to improve quality. One interesting feature is that it is possible to render directly in the processing view.

Integrated into the Render menu are effects like particle systems. This means that smaller objects such as particles of smoke are assigned reciprocal physical interactions, with the result that they act as a unit. These options are not found in any other programs since particles are really allocated to

geometry and the effects attributed to the particles are in any case amalgamated two-dimensionally in the renderer.

The chief developer of Blender, Ton Roosendaal, has announced version 2 of the software and released a beta version (1.8) which is said to have all the features of the full version.

RenderMan, the modern classic

Like Pixar, the manufacturing company behind it, RenderMan has an interesting history. It evolved from a collection of 3D tools which were used in movies such as "Toy Story" and "Bugs". Since then, RenderMan has become a commercial product and costs £5,655. The program is nevertheless being briefly discussed here because the RenderMan file format (rib) is slowly turning into a quasi-standard. RenderMan is available as an RPM package for Red Hat Linux.

A rib-file contains all the scene data in an ASCII text file. This includes stored objects, references to shading lists, lights and scene parameters, if necessary for each individual frame of the animation. A light definition, for example, looks like this:

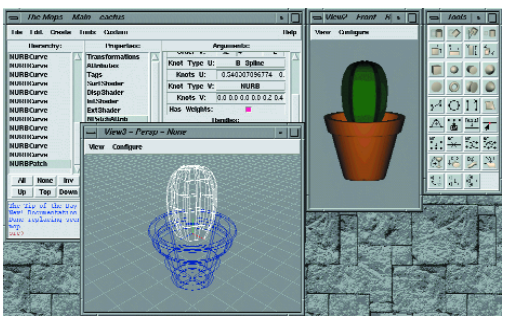
```
LightSource "pointlight" 1 "intensity" [9.5]
] "lightcolor" [1 1 1] \
          "from" [-1.5198 7.76837 -3.877]
```

Shading lists – ASCII files which describe nested and programmable materials – must first be compiled. They can then be installed in the rib file like a programming library in a C program in the following way:

```
Surface "Nice_new_surface"
```

There are graphical front-ends for both the creation of the rib file and also the shading list files. They can also be exported from many programs. However, exporting requires reprocessing, which is often time-consuming. Pixar uses, among others, the program MTOR to create objects.

The advantages of RenderMan lie in the quality, flexibility and high processing rate. Another great advantage is the sub-division of surfaces wherein it is possible to create complicated organic objects without having to work at higher resolutions. The objects created are converted optimally by RenderMan into polygons. The main advantage of this method compared with Nurbs is that surfaces can



The surface of mops enters, in the guise of Tcl/Tk

be refined and weighted better. This is shown to great effect in the eye parts of the figures in "Toys-tory II", for example.

In addition there is a Shader language which makes it possible to create complex materials. Unfortunately RenderMan has no radiosity and no proper ray-tracing. Nor is there an area light or caustic light. To compensate for these drawbacks, the rather remarkable "Blue Moon Rendering Toolkit" (BMRT) was created.

Blue Moon Rendering Toolkit

BMRT costs considerably less then RenderMan. To be precise, it costs nothing at all. It produces better results but is slower. However, it has additional controls such as radiosity, extended light source methods and many other effects. Apart from that the two are compatible. It is even possible to combine the fast processing performance of RM with the quality features of BMRT. RM calculates the geometry whilst BMRT takes care of reflection, refraction and radiosity. BMRT can be found as a binary for all normal Linux distributions at the following website: http://www.bmrt.org .

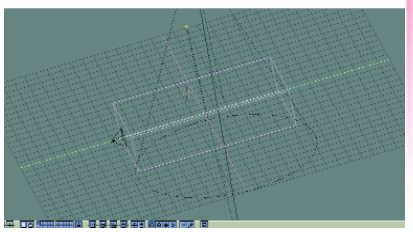
Mops

Mops is a modeller for RenderMan-compatible renderers and was written in Tcl. The program works solely with Nurbs objects but offers an adequate number of functions. For example, you can draw a line or Bezier curve and then use the revolve function to produce a rotating object. With the trim function parts of a Nurbs object can be cut out and a skin object stretched over it.

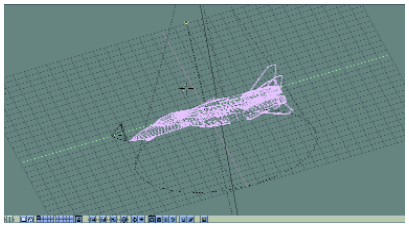
The program is operated by a combination of mouse and keyboard, though the program is designed primarily for keyboard use. All key combinations can easily be found in the comprehensive documentation.

In one part of the window the program offers an overview of all the objects present in the scene and their characteristics, which correspond to those of the rib specification. In addition there is another console in the window, providing access to the Tcl interface. This means that the program can to some extent be scripted, although it is a long way from being as flexible as Blender.

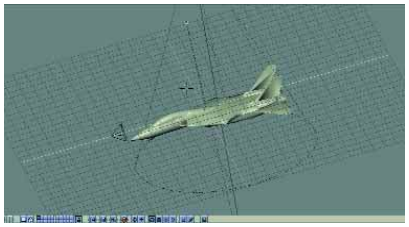
An under water panorama, produced and rendered in Blender.



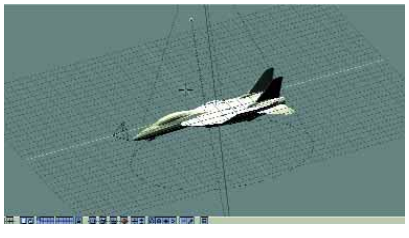
The simplest view - the bounding box defines only the scale of the object



Wireframe view - here it is also possible to see that the object is made up of quad-polygons



This OpenGL view shows the object with a surface but without any other effects

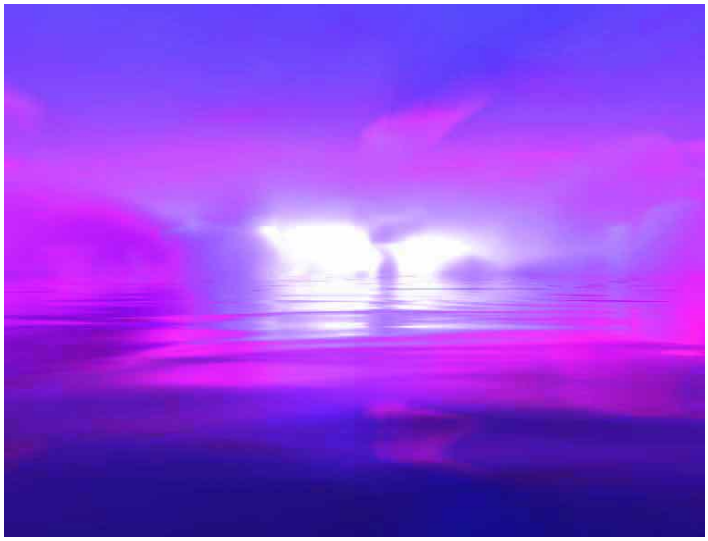


In this view the light conditions can now also be seen – the spotlight over the aircraft produces realistic shadows.

Listing 1: interface.c

```
01 #include < Lightflow.h >
02
03 int main (int argc, char* argv[])
04 {
05     LfLocalSceneProxy* scene =
06                     new LfLocalSceneProxy();
07     LfArgList list;
08
09     list.Reset();
10     list <<"position" <<LfPoint(5.0, -5.0,  4.0);
11     list <<"color"  <<LfColor(1.0, 1.0, 1.0)*3e2;
12     scene->LightOn(scene->NewLight("point", list));
13
14
15     list.Reset();
16     list <<"ka" <<LfColor( 0.0, 0.0, 0.5 );
17     list <<"kc" <<LfColor( 1.0, 0.5, 0.5 );
18     list <<"kd" <<0.5;
19     list <<"km" <<0.1;
20     LfInt plastic =
21             scene->NewMaterial("standard", list);
22
23
24     scene->MaterialBegin( plastic );
25
26     list.Reset();
27     list <<"radius" <<1.0;
28     scene->AddObject
29             (scene->NewObject("sphere", list));
30
31     scene->MaterialEnd();
32
33
34     list.Reset();
35     list <<"file" <<"ball1.txt";
36     LfInt saver =
37             scene->NewImager("saver", list);
38
39
40     scene->ImagerBegin( saver );
41
42     list.Reset();
43     list <<"eye" <<LfPoint(0.0, -4.0,  0.0);
44     list <<"aim" <<LfPoint(0.0,  0.0,  0.0);
45     LfInt camera =
46             scene->NewCamera("pinhole", list);
47
48     scene->ImagerEnd();
49
50
51     scene->Render( camera, 400, 300 );
52
53     delete scene;
54 }
```

Depending on the pattern used the same model can produce either an ocean with sunset or ...

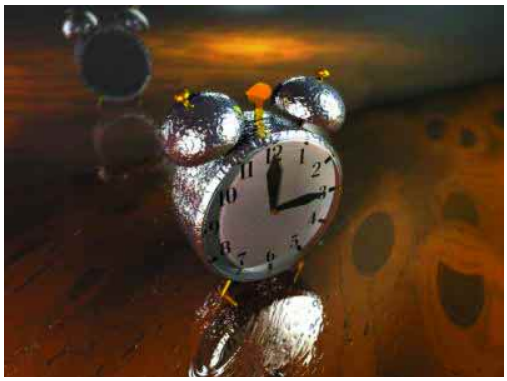


... a cloudscape with thundery atmosphere.

[left]
When performing scientific visualisation, realistic depiction of surfaces and light effects matters.

[right]
Clock with striker: Lightflow makes this kind thing possible, too ...

All the options correspond to the settings which are available in the RenderMan format. User-defined parameters can also be added. This means that it is possible to achieve very high quality in combination with RenderMan or BMRT. The light source functions are a real weakness, however. It would also be nice to be able to import files in the RenderMan format.

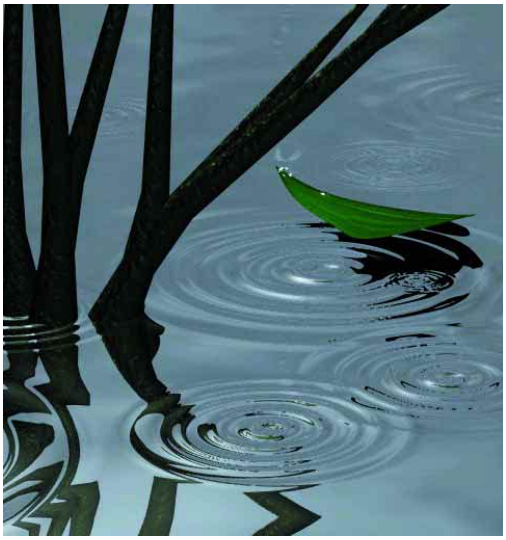


Lightflow Rendering Tools Version 2.0

The author of Lightflow, Jacopo Pantaleoni, is just 21 years old. He has been involved in programming 3D effects since he was 12. Lightflow is more an object-oriented C++- and Python library than a program in its own right, and is designed to be expanded. The example in Listing 2 shows how simple scenes can be programmed and visualised. The strength of Lightflow lies in the description of shadings of light in three-dimensional space. This is also its chief advantage: it can undertake procedural definition of surfaces, volumetric patterns and materials, lighting systems or camera positions.

Both programming interfaces are freely available for non-commercial use. Plus, there is a comprehensive set of tools for volumetric rendering or radiosity, which allow the conversion of simple scenes into truly impressive images. This can be

This close-up of a face rendered with Realsoft 4D still looks rather plastic.



But Realsoft*4D has done a really good job with a water surface and waves.

seen to great effect in the sample images. Both ocean scenes are based on the same model, a surface and a sky, but using different multifractal patterns – in one case waves are created and in the other it is clouds.

In a similar way, motifs such as a glass object with near-perfect light conditions or an alarm clock with hammer striker effect are created

Realsoft 4D

Unfortunately there is very little information on Realsoft 4D [7] to be found on the Web. It's the successor to Real 3D and has been available for nearly ten years. It is primarily a commercial product from the company Realsoft, Inc. for the Windows NT platform, but it is now also available in a beta for Linux.

There are a few tutorials on the website under "Links", along with the obligatory galleries. But there are only vague references to the technical innards which would be of interest to specialists. The results appear primitive compared to other packages too. It is Lightflow that sets the standards in the quality of lighting conditions on surfaces.

Not just for hobbyists

Linux is the perfect platform for learning how to handle 3D graphics – the available tools are just what you need and won't plunder your bank account. Through Renderman-compatible software you even have a path into professional modelling. For truly large-scale professional projects, however, a pure Linux-based solution is not yet practicable. The modellers available for Linux are not yet setting any standards, even if the actual render program has long been among those at the very top. Nevertheless, for the pro or the semi-professional user modelling under Linux is a serious alternative, so long as the size of the project stays within limits. ■

Info

Blender links: *http://www.flyingsnail.com/blenderseven.html*
Mops homepage: *http://www.informatik.uni-rostock.de/~rschultz/mops/*
RenderMan specification: *http://www.pixar.com/products/renderman/toolkit/RISpec/*
Product info on Pixar's RenderMan: *http://www.pixar.com/products/renderman/*
Lightflow Tools: *http://www.lightflowtech.com*
Realsoft 4D: *http://www.realsoft.fi*

■

Overview: Renderers and Modellers

Modeller	Blender	Mops
Polygon editing	yes	no
Nurbs	yes	yes
subdivisions	yes	no
Metaballs	yes	no
Skripting	yes	yes
Materials	built in	RenderMan/Shadereditor
Import	Inventor, Blender	3DMF
Export	RenderMan, VRML1, VRML2, dxf, VideoScape	3DMF, RenderMan
Renderer	built in	RenderMan
Homepage	http://www.blender.nl	http://www.informatik.uni-rostock.de/~rschultz/mops/
Modeller	**ac3d**	**Realsoft4D**
Polygon editing	yes	yes
Nurbs	no	yes
subdivisions	no	yes
Metaballs	no	yes
Skripting	no	Visual Shading Language
Materials	very few	Viele
Import	3ds, dxf, lightwave, triangles, vectors, vrml1, triangles	rpl
Export	vrml1, vrml2, pov, r3enderman, dive, massive, dvs, Triangle	rpl, dxf
Renderer	RenderMan, povray	built in, Raytracing only
Homepage	http://www.comp.lancs.ac.uk/computing/users/andy/ac3d.html	http://www.realsoft.fi
Renderer	**Renderman**	**Blue Moon Rendering Tools (BMRT)**
global illumination	no	yes
Volume Materials	yes	yes
Nurbs	yes	yes
displacementmaps	yes	yes
subdivisions	yes	no
Field of Vision	yes	yes
Motion blur	yes	yes
Binarys available for	Red Hat	i386 (libc/glibc)
speciality	rather fast, subdevision surfaces	
Freeware	no	yes
Compatibility	Maya	RenderMan, maya
Homepage	http://www.pixar.com/products/renderman/	http://www.bmrt.org
Renderer	**Mental Ray**	**Lightflow**
global illumination	yes	yes
Volume Materials	yes	yes
Nurbs	yes	yes
displacementmaps	yes	yes
subdivisions	yes	no
Field of Vision	yes	yes
Motion blur	yes	no
Binarys available for	Intel Linux 2.2, SuSE 6.1, Red Hat 6.0	Red Hat 6.1, Debian 2.1
speciality	subdivision surfaces, global illumination	additional engines for lighting (i.e. Arealights)
	Integrated in Modeller »Softimage«	Multithreaded, C++ API with Python-Wrapper
Freeware	no	yes
Compatibility	Softimage, maya, RenderMan	3dsmax
Homepage	http://www.mental-ray.com	http://www.lightflowtech.com