Renderer

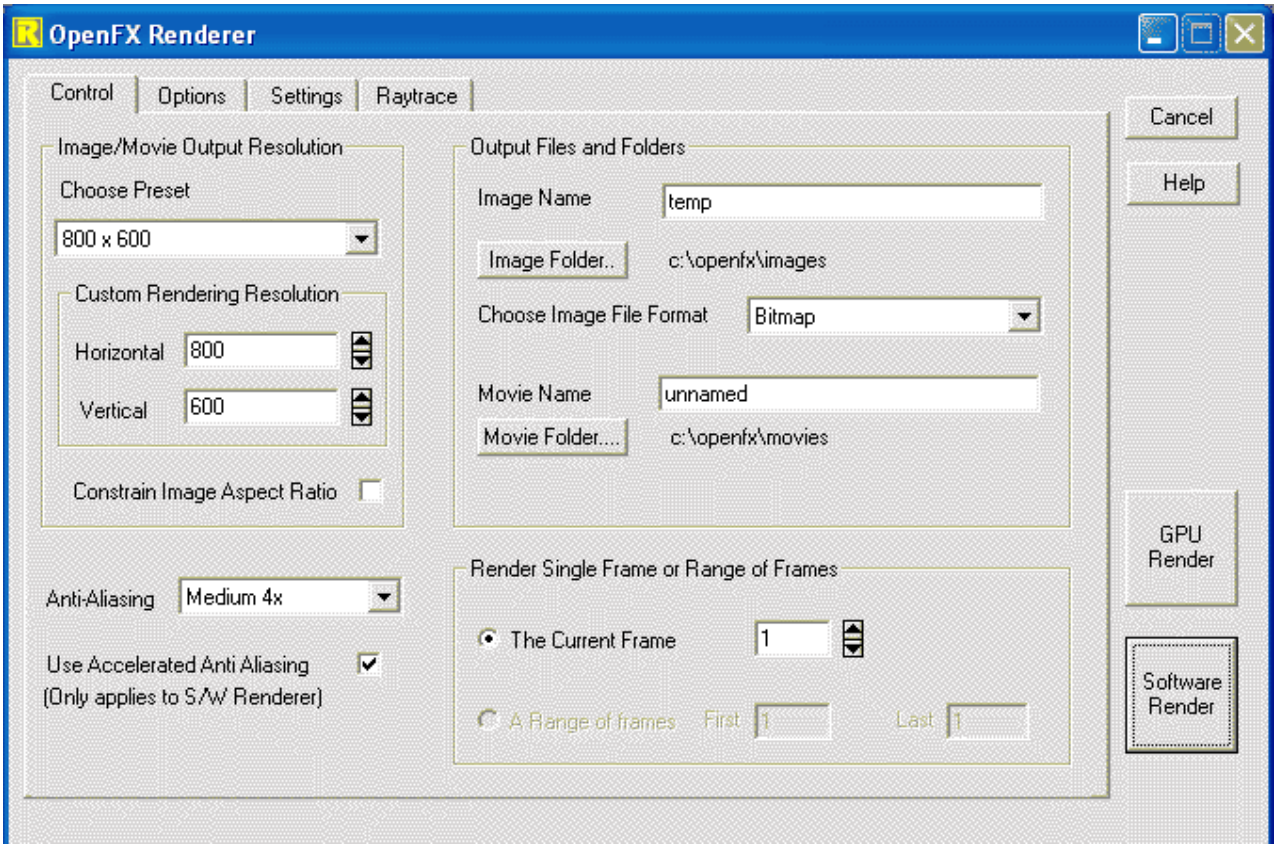
The Renderer is the module that creates the photographic images of models built in the Designer and the animated sequence of images as directed in the Animator.

Rendering is quite a time consuming process and the time it takes is very dependent on the size image being produced and the complexity of the Actor's costumes.

The Renderer module can be used in a mode (called the Script Renderer) that does not need either the Animator or Designer modules to be loaded. The Script Renderer may also be run on several others PCs at the same time.

Renderer Control Tab Page

The Control page of the Renderer offers the most significant control parameters. The most important buttons are on the right and control whether the scanline software renderer is used or the harware rendering engine(GPU rendering) is used. The hardware rendering engine will be much faster for most images and movies (up to 30 times faster) but it will not work on older hardware and some graphics cards may not be compatible. (You must have a graphics adapter that supports OpenGL 2.0) All nVidia graphics adapters produced in the last 5 years should be compatible. Most ATI graphics cards should also work.) The GPU renderer has been successfully tested on a wide range of ATI and nVidia hardware.



Resolution - Choose the resolution to which the image is rendered. Choose one of the presets or enter the number of horizontal and/or vertical pixels in the edit controls. The number of pixels in the image is horizontal resolution multiplied by the vertical resolution. The higher the resolution the longer it takes and the more memory it needs to render the image.

Anti-Aliasing - Choose the level of anti-aliasing. In low resolution modes the appearance of an image can be improved if it is anti-aliased. Basically, anti-aliasing smoothes out some of the 'jaggies' that are visible at the edges of colored regions in the image. Anti-Aliasing is very time consuming: it will take a little under 4 times as long to render a Good anti-aliased image than an image with no anti-aliasing. Using Best level of anti-aliasing, the image will take approximately 9 times longer to render than a non anti-aliased one.

Accelerated Anti-Aliasing - Accelerate the anti-aliasing process, with small quality reduction. Normally anti-aliasing an image increase the rendering time quite significantly, up to 9 or 10 times for a setting of Best. The main effect of anti-aliasing is seen near the edges of models or where faces with different colors join. Pixels in the interior of faces benefit very little from anti-aliasing. When this option is selected, anti-aliasing is applied near the edges of faces and not in the center. For most models this is as good as full anti-aliasing and it can be up to 3 times faster. Images with models that have image maps or shaders, where there is quite a bit of detail in the interior of faces, will not benefit as much from having their anti-aliasing accelerated.

Image Format - Select the file format into which the image or images will be stored. This automatically determines the color resolution in the image. For an single image or multiple still images the GIF format uses an 8 bit palette, while the TGA and TIF formats are either High-Color or True-Color (16 bit or 24 bit).

Image Name - Name the image and the directory to which it is to be rendered. The default image directory is "Render".

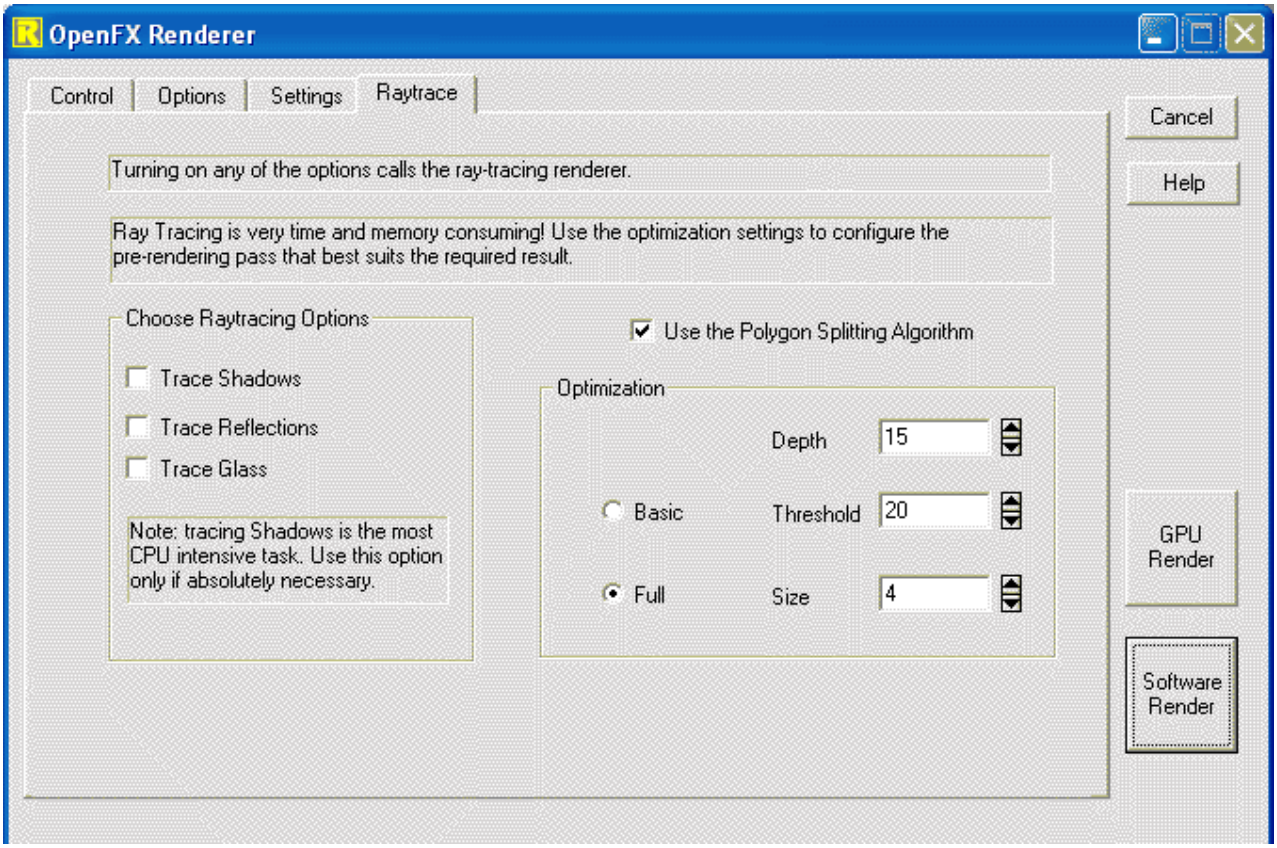
Movie Name - Name the animation and the directory to which it is to be rendered and compressed. The default animation directory is "Movies". Movies are written in the AVI fiel format and can be written using any installed codec or in an uncompressed format for high quality postprocessing.

Current Frame - Specify which frame of an animation is to be rendered.

Range - Specify a range of frames to be rendered.

Render Tracing Tab Page

The Tracing page of the Renderer enables Ray Trace rendering to be used. Turning on any of the Tracing options instructs the Renderer to use ray tracing where appropriate.



Trace Shadows - All shadows will automatically be ray traced.

Trace Reflections - All reflective surfaces will automatically be ray traced reflections.

Trace Glass - All glass surfaces will automatically be ray traced reflections with refraction if specified.

Use Polygon Splitting Algorithm - If this box is checked a more elaborate Ray tracing optimisation algorithm is used. For scenes with a large number of polygons it may be more efficient to use this procedure. (e.g. > 10000). For smaller scenes the original optimisation procedure probably still works better. The best option is to try both on a single frame in a low resolution and see which performs faster.

The Ray Tracing Algorithm

Ray tracing may be a slow procedure but it does give superb photographic quality images. There are many books that describe the theory of ray tracing in detail. A full description of a complete package is given in [1] and there are also many available freeware and shareware ray tracing programs. One of the most comprehensive is POV-Ray which is available from a large number of WWW locations.

The idea on which ray tracing is based seems almost too simple to be true but - here it is:

> *Follow the paths taken by particles of light (photons) on their journey from light source through the scene (as they are scattered back and forth between objects) and on until they are either captured by the camera or head off to infinity.*

In practice this is a hopeless task because of the huge number of photons emitted by a light source and the fact that all but a minute fraction of them will ever be scattered into the camera or eye. So ray tracing does it in reverse. It sends feelers or rays along the path of the photos from the camera out into the scene. If the feeler rays find anything they work back towards the sources of illumination and give us a path for the photons to follow from source to photographic plate. Sometimes these feeler rays may encounter reflective surfaces and when that happens they follow a new path and continue their journey. There are lots of other things that can happen to
feeler rays. Sometimes they may divide with each sub-ray following separate paths. The way in which the ray tracing algorithm handles such situations and the sophistication of the mathematical models it uses for light/surface interaction governs the quality of the images produced.

The standard algorithm for ray tracing can be summarised in five steps:

- Load the data describing objects in the scene to memory.
- Move each model into its appointed position.
- Apply the viewing transformation.
- Calculate the direction of a feeler ray from the viewpoint so that it passes through the point in the projection plane equivalent to a pixel in the display raster.
- The feeler ray is followed (traced) out into the scene until it intersects a polygon. Once the point of intersection is located the surface properties can be determined and the pixel set to the appropriate colour value.

If when a ray first hits something in the scene it isn't totally absorbed we can continue to trace its path until it reaches a final destination or is so attenuated that there is nothing to see.

So that's the basic algorithm, but if you have ever experimented with ray tracing software you probably found that at most you could use about 100 primitive shapes. Your main observation probably was the hours it took the computer to trace even an image of moderate resolution. What is worse is that the rendering time increases as the number of models in the scene increases thus if you need to render a scene with models made up from thousands of polygons a basic ray tracing renderer is virtually useless.

So the main question is: Why is ray tracing so much slower? The answer is in the algorithm, in particular the step: *For each pixel test every polygon and find the one closest to the origin of the ray.* This is potentially an enormous number of operations. For example to synthesize an image of size 640 by 480 using a 2 by 2 anti-aliasing supersample with a scene of 20,000 polygons requires 24 billion tests.

To try and reduce the time it takes to render a scene with a ray tracer some attempt must be made to try and reduce this enormous number of calculations. A technique we call *optimisation*.

Optimisation

It is the job of an optimisation routine to divide the scene up in some way so that every ray does not have to be tested against every polygon. Optimisation usually requires two modification to the ray tracing algorithms:

- A scene evaluation step that takes place before the scene is renderer.
- As each ray is traced the scene evaluation data is used to reduce the number of polygons that must be tested against it as the ray passes from point to point.

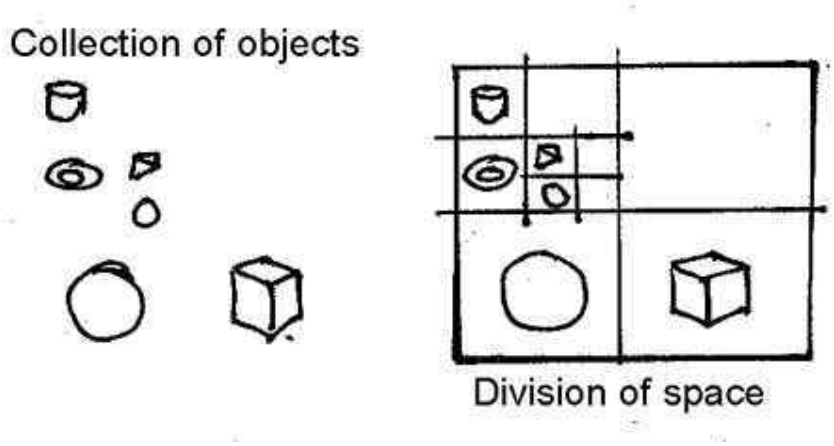
There are two approaches that have received much investigation:

- Bounding volumes
- Spatial subdivision

The renderer uses the spatial subdivision method and we will look at that:

The idea here is to break up space into a number of small boxes so that only a few polygons actually lie in each little box. Then as the rays are traced through the scene we follow them from box to box and check to see if a ray hits any of the polygons lying inside the boxes. *If there are only a few polygons in each box we will have considerably reduced the number of calculations and hence the time it takes to render a scene!* Ideally putting only one or two polygons in each box (in technical terms the "boxes" are know as "voxels") should be the goal of the first part of the optimisation scheme.

A two dimensional illustration of this principle is given below, it shows a collection of objects and a series of rectangles that divide up the region in which the objects lie so that so that only one object is in each little rectangle:



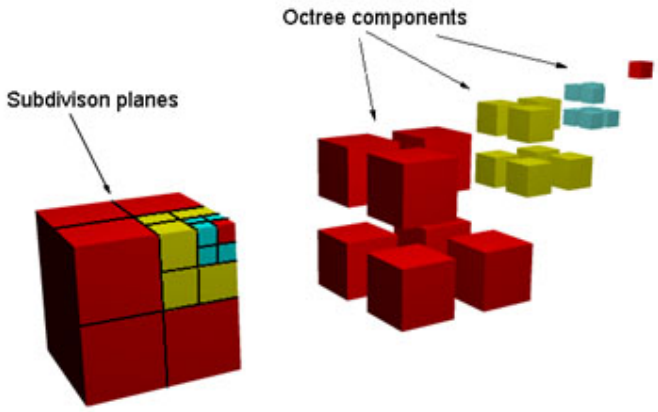
It is the job of the first optimisation step (scene evaluation) to divide up the scene into the boxes to be used in the later tracing step. There are conflicting goals here, each voxel should have as few polygons (faces) inside them as possible. This implies that they should be small - but if there are too many boxes then the tracing step will not be very fast because as the ray is followed it will have to travel through a very large number of boxes. This *is a "balancing act*" - technical term "*Optimisation*". It requires quite a bit of processing to find the best arrangement of the voxels but since many hours of processor time might be saved during rendering it is usually well worth the effort.

One comment worth making: The time it takes to optimise a scene is dependent on the number of polygons in the scene ***not*** on the number of pixels in the output picture. So it takes just as long to optimise a scene when rendering a 60 by 40 image as it does for a 1024 by 768 image. However it takes much much longer to trace the image for the larger raster. Thus here again is another very profitable reason to do the best possible job at optimising the scene prior to tracing any rays.

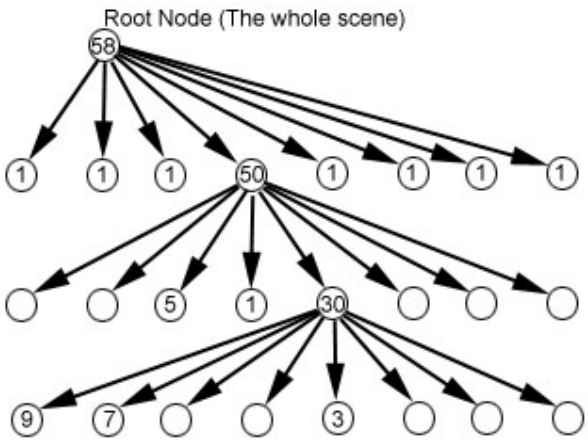
Octrees

The optimisation step starts by calculating the volume of space occupied by all the faces in the scene to create a bounding box to enclose all the faces in the scene (the so called root node in the octree). If this box contains more than the minimum number of faces it is divided into 8 smaller volumes. These child volumes are checked and any faces assigned to the parent and also fall inside them are put into to a list for each child. If any child box contains more than the minimum number of faces it is further subdivided into 8 sub-volumes, just as its parent's box was. This process is recursive and gives us a tree like structure called an *octree*, because at each branch there are 8 subdivisions.

This subdivision process for a cubic region of space is illustrated below:



The big cube has been divided into 8, one child node is further divided at each stage. In hierarchical terms the octree resembles:



The numbers in the circles represent an example of the number of faces inside the box associated with that node in the octree. You can see that as the subdivision is refined the number of faces in each box decreases. In this particular example the subdivision process is stopped when the number of faces in each box falls below 10.

Once the octree is fully built ray tracing can proceed. Thus when the ray-tracing part of the algorithm follows a ray through the scene it moves from box to box and only a maximum of 10

faces have to tested in each box. A big saving in computation time. The tree structure of the octree makes the task of following a ray through the scene quite quick.

There is a bit of work involved in calculating the movement of the ray and in getting a fairly quick test to see in which box the current position of the traced ray lies but we don't need to worry about this to understand the setting of the ptimisation parameters.

However before actually looking at the parameters there is one complicating issue we must mention:

All this looks great in theory until we come to consider a model made from a connected network of polygonal facets, then we have a bit of a dilemma! A network of connected facets cannot be separated into individual faces. No matter how hard we try, (how may times we subdivide), there is **no** way that we can separate each polygon so that a rectangle in the subdivision contains only one polygon. The fact that the polygons are connection is responsible for that. The very best we could probably do is associate some polygons with more than one voxel simultaneously. We are going to have to accept that in practice a one polygon to one subdivision goal is unattainable. So we must set a more realistic limit on the number of polygons we are prepared to accept in each voxel.

How close the renderer gets to the 1 to 1 ideal can be determined from the reported information. The (F) value is the number of faces in the scene, the (Fc) value reports how many faces have been placed in voxels. Thus if Fc=F we have a perfect 1 to 1 set-up and the ray tracing should be very fast. The (F/V) value is the maximum number of faces assigned to any one polygon, ideally this should be equal to the desired value (which is set by the Threshold parameter). In most cases we rarely have either Fc=F or F/V = Threshold , any scene in which Fc < 5 F should be regarded as acceptable. In other scenes it is probably worth experimenting with the parameter settings.

The parameters

- First the option to do a Full or Basic optimisation. There is quite a lot of calculation necessary to decide if a face lies inside a particular box. There are a large number of tests involved. The Basic optimisation only does a few of these tests and therefore it over-predicts the number of faces that lie inside a particular voxel. This means that while the optimisation phase will take less time the tracing phase will take longer. So use the Basic setting when the size of the picture is small and Full optimisation when rendering a high res. image with best anti-aliasing.

- The Depth parameter allows the user to tell the renderer what is the maximum number of layers in the octree it may use. Thus SFX won't go on subdividing even if the other settings would allow or require it. This parameter is needed because in some scenes too much memory would be needed and the optimisation step would take forever to complete.

- The Threshold parameter is related to the number of faces that are assigned to each voxel. It is the desired number of faces we would like to see assigned to each voxel. (All this is discussed above.) Ideally one or two but for larger models 9 or 10 or even 100 to 150 are possible values. It just depends on the number of polygons in the scene and only experimentation will tell if you are heading in the right direction. Again the (F/V) and (Fc) values reported should help you refine your settings.

- Size controls the minimum acceptable voxel size. What you choose depends on the dimensions of the model in internal units. The default setting is ideal for modes constructed inside the default WindowBox volume. For models that have grown big (in size) or have been imported a larger value of Size will help in speeding up the rendering process. The reported size value should help you refine this particular setting.