Forest Scene Manual

Paul Guerrero

September 9, 2006

1 The Interface

Figure 1: Screenshot of the interface.

The interface is split up into five panels. In the lower left corner is the performance panel showing fps statistics and triangle count. The panel at the bottom displays the movement keys and the keys used to control the

animation of the sun in the scene. The two panels at the top right of the screen show information about the Depth of Field effect and the Lens Flare Effect.

The large tabbed panel at the top is divided into three sections: general information, information about the static geometry tree and information about proxies in the scene.

All panels except the performance panel are explained in more detail in the next sections.

1.1 Movement and Animation Panel

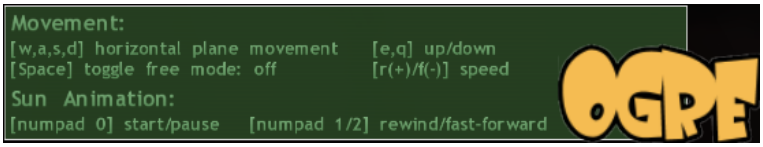

Figure 2: Screenshot of the Movement and Animation Panel.

This panels shows movement options and sun animation controls.

Navigation in the scene works as follows: Use the keys **w**,**a**,**s**,**d** to move forward, backward, left and right. The keys **e** and **q** move the camera up and down.

By default, the camera follows the terrain. Use **space** to change between follow and free mode.

The keys **r** and **f** change movement speed.

The sun in the scene and some colours (e.g. sky colour, ambient colour, etc.) are animated to simulate a day/night cycle. **numpad 0** starts and paused the animation. **numpad 1** and **numpad 2** can be used to advance or rewind the day cycle by a fixed amount.

1.2 Depth of Field Panel

This panel shows information about the Depth of Field effect. Use **numpad 7** to turn the effect on and off. **numpad 8** and **numpad 9** control the focal distance. The focal distance is the distance at which objects are in focus. **numpad 5** and **numpad 6** control the f-stop. The f-stop controls the strength of the depth of field effect (small f-stop values give a strong

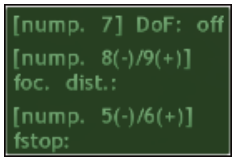

Figure 3: Screenshot of the Depth of Field Panel.

depth of field effect). With a small f-stop value, objects get blurred quickly when moving away from the focal distance.

1.3 Lens Flare Panel

Use **l** to toggle the Lens Flare effect.

1.4 General Options Panel

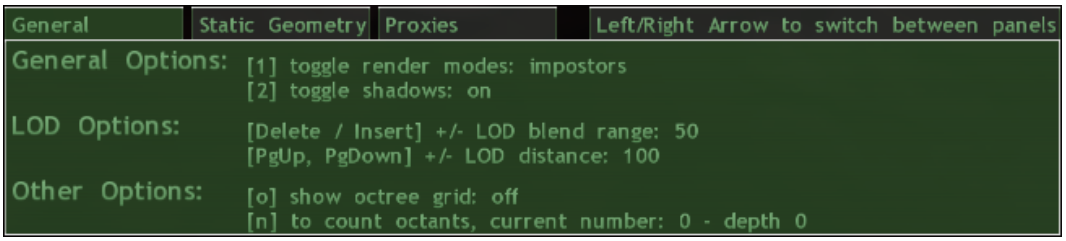

Figure 4: Screenshot of the General Options Panel.

This panel shows information about the render mode and Level-of-Detail settings.

The key **1** switches between render modes. The default render mode is static geometry rendering. **2** toggles shadows on and off. For performance reasons, shadows are only displayed for trees close to the camera.

The LOD Options have two parameters, the LOD blend range and the LOD distance (see Figure 5). The LOD blend range controls the distance range over which two LODs are blended. The larger this parameter, the smoother the blending. The LOD distance controls the depth at which the high detail LOD is no longer visible (only the low detail LOD). Blending starts at *LOD distance - LOD blend range* and ends at *LOD distance*. For

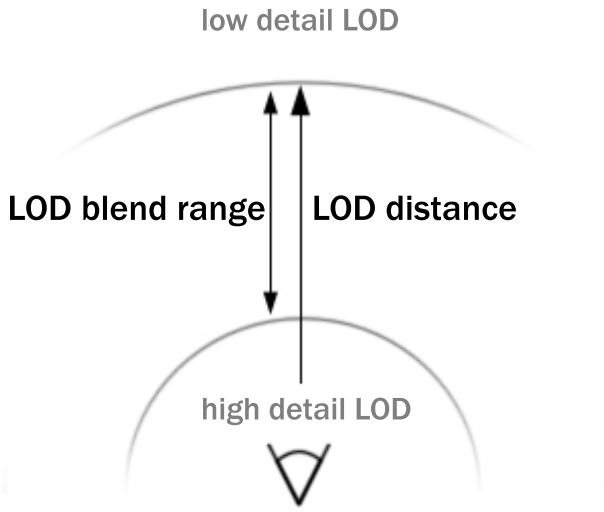

Figure 5: LOD blending.

example, if you set the the LOD blend range equal to the LOD distance, the two LODs will be blended from distance 0 to LOD distance.

Show octree grid (**o**) displays wireframe boxes for each octant in the octree. By pressing **n**, you get the current octant count and depth of the octree.

1.5 Static Geometry Panel

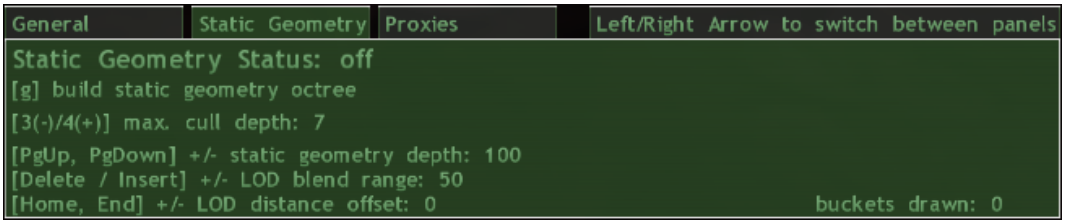

Figure 6: Screenshot of the Static Geometry Panel.

This panel shows information about the static geometry octree. Static geometry is used to put as much tree geometry as possible into one render call

while still being able to send only those parts of the scene to the graphics card that are visible and at enough distance to have a simpler LOD level (it would take up too much memory to have high detail trees in the static geometry).

The default status of the static geometry is off. To change the status, you have to build the static geometry octree (**g**) and then change the render mode to static geometry (**1**). You cannot use proxies and static geometry at the same time, only one of the two can be built.

The maximum culling depth specifies the octree depth at which octants are no longer split up into their child octants for culling if they are only partially visible (if you set this value to zero, no visiblity culling is done). Not rendering those parts of an octant that are not visible speeds up rendering if much geometry can be discarded, but more render calls are needed. Culling should only be done for large octants. The default value of 7 is a good choice for this specific scene.

The next three values control LOD settings. The first two options are the same as the LOD options in the General Options Panel (see section 1.4 and Figure 5), except that the LOD distance also controls the distance after which static geometry is used for the trees. It is sensible to use the same distance: when using a smaller static geometry distance, part of the LOD blending would be cut off, when using a larger distance, trees that could already be static geometry would still be drawn one at a time, wasting render calls. You can however control the offset of the LOD distance from the static geometry distance with the third LOD option. If it is positive, the LOD distance will be greater than the static geometry distance.

The last line on the right shows the number of static geometry 'buckets' drawn. Each bucket needs one render call.

1.6 Proxy Panel

This panel shows information about the proxies in the scene. Proxies are simplified representations of the geometry in an octant (octants in each octree depth get proxies). The octant is rendered from six sides and one colour is stored for each side. The idea is to show only this colour instead of the geometry in the octant's volume if the octant's area on screen is small. Two types of proxies have been implemented, billboard proxies and cube proxies.

Billboard proxies use billboards to display the stored octant's colour. Billboards always face the camera and the displayed colour is interpolated

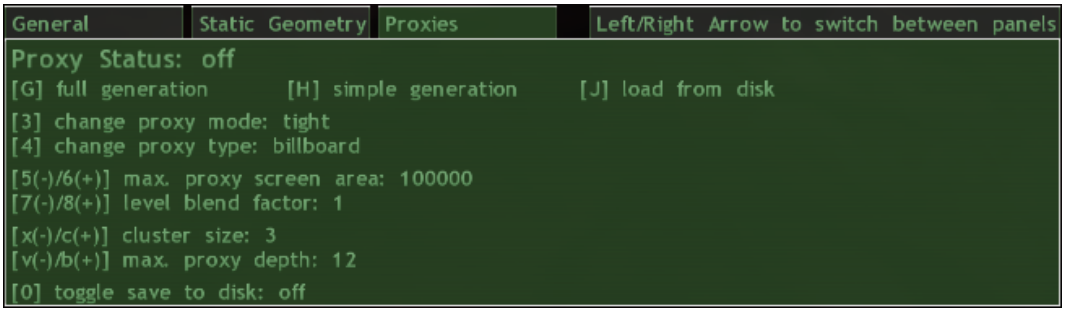

Figure 7: Screenshot of the Proxy Panel.

from three of the six stored colours depending on the viewing angle.

Cube Proxies are colour cubes of the size of the octant they represent. Each side of the cube has one of the stored colours.

It is recommended to use static geometry instead of proxies for the forest scene. Proxies would only be useful for trees that are extremely far away, such that one screen pixel (or very few pixels) would contain many trees.

The default status of the proxies is off. To change the status, you have to build proxies (keys **g**, **h** or **j**) and then change the render mode to proxies (key **1**). You cannot use proxies and static geometry at the same time, only one of the two can be built.

Proxies can be built either by rendering each side of octants in every level of the octree (full generation, key **g**), or by only rendering the sides of the leaf octants and then calculating the colours of the octants in higher levels from their child octants, which is faster (simple generation, key **h**).

If you have more than one LOD level, you should use the least detail level when generating proxies (e.g. by setting the LOD distance to 0 before generating the proxies). Using high detail geometry would be wasteful and take longer, since one tree is rendered to only a few colours at most.

The generated colours of the proxies can also be stored on disk by turning on the save to disk option (key **0**) before generating proxies. Next time, these colours can be loaded from disk (key **j**) and no rendering has to be done to generate the proxies. The settings used when storing the proxy colours get stored, too and are restored when loading the proxy colours. Loading proxy colours is only possible if the octree structure of the scene is the same as when the proxy colours were saved.

By default, proxies are created for a tight octree (as opposed to a loose

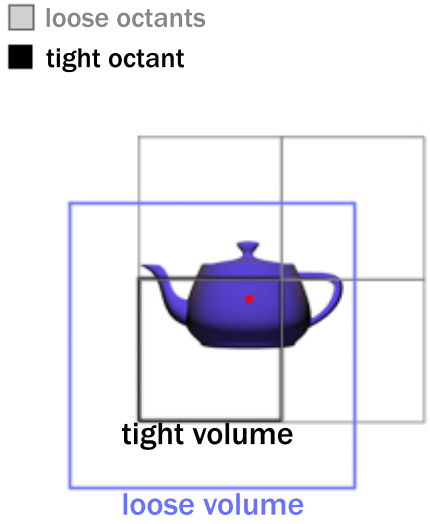

Figure 8: The teapot is completely included in its loose volume which overlaps the neighboring octants' loose (and tight) volumes.

octree where octant volumes overlap). In a tight octree, proxies don't overlap (see Figure 8). The drawback is that more octants and more proxies are necessary since one object may be included in the volumes of several octants and each octant needs a proxy. In a loose octree, each object is assigned to only one octant and is completely included in this octant's volume, but octant volumes overlap. By changing the proxy mode to 'loose mode' (key **3**) before generating proxies, fewer proxies are generated since fewer octants are needed, but the proxies overlap. It is recommended leave this option in 'tight mode' to avoid visible artifacts in the overlapping parts of the proxies.

Use the key **4** to choose the proxy type before generating proxies.

The keys **5** and **6** control the maximum area proxies may have on screen (in pixels).

When using billboards, blending may be used when switching between the proxy representation of octants at different depths of the octree (like LOD blending for proxies). This values specifies the blend range. A value of 1 means no blending, 0.5 is the maximum blending value. Only values between 1 and 0.5 are valid.

Proxies can be clustered, meaning that one octant gets more than one

proxy. The cluster size (keys **x** and **c**) controls the number of proxies. Each octant gets $cluster\_size^3$ proxies.

The maximum proxy depth is the octree depth after which octants no longer get proxies. Together with proxy clustering, it is possible to group proxies into clusters, which is much more efficient than drawing each proxy by itself, but has the drawback that only the whole cluster can be shown or hidden, not parts of it.

For example, to cluster all leaf proxies that have the same parent octant, choose a cluster size of 2 and a max. proxy depth of $octree\_depth - 1$.

2 Modifying the scene

2.1 Rearranging Objects

The scene is defined in the file called 'Objects.txt'. To add a new object, write its mesh name in one line followed by all instances of the mesh in the lines below. Each instance is specified in one line, starting with the unique name of the instance. A line in the Objects.txt file representing one instance of an object has the following format:

```
<UniqueName> <pos_x> <y> <z> <rot_x> <y> <z> <scale_x> <y> <z>
```

There are three object categories in the scene that are treated slightly different. Instances of meshes starting with `'Terrain'` are treated as terrain, the camera will try to follow the terrain if not in free mode. Instances of meshes starting with `'Building'` aren't added to static geometry or proxies and the camera may collide with them. All other meshes are treated as trees. Only trees are added to static geometry or proxies.

To add a LOD level to a mesh, add a line with the lower detail mesh name below the line of the high detail mesh, followed by the distance after which this LOD level should be displayed (this distance is usually overridden by the global LOD distance setting in the General Panel or the Static Geometry Panel - see sections 1.4 and 1.5). For example, to use two LOD levels on a mesh you could write the following:

```
myFullGeom.mesh
myLowDetail.mesh 50
myInstance 0 10.0 0 0 90 0 0.5 0.5 0.5
```

For more information on rearranging the scene and for instructions how to generate trees automatically, see the included TreeGeneration.pdf or TreeGeneration.doc document.

2.2 Adding new Tree Models

All trees currently in the scene use vertex and fragment programs for LOD blending and tree shading. When adding new trees a few things have to be done for these two features to work properly.
The easiest way is to copy an existing tree material
(e.g. `'media\Trees\Cammelia\cammelia1.material'` and
`'media\Trees\Cammelia\cammelia1Static.material'`)
and modify it as needed.

Otherwise, you have to modify your material to use the tree fragment and vertex shaders. Here is a list of the shaders needed (in Ogre material script syntax):

```
fragment_program_ref tree_fp_hlsl
{
  param_named_auto ambientColour ambient_light_colour
  param_named_auto lightColour light_diffuse_colour 0
}

vertex_program_ref tree_highDetail_vp_hlsl
{
  param_named_auto mWorldViewProj worldviewproj_matrix
  param_named_auto objSpaceLight light_position_object_space 0
  param_named_auto objSpaceCamera camera_position_object_space
  param_named_auto opacity custom 55
}
```

```
vertex_program_ref tree_lowDetail_vp_hlsl
{
  param_named_auto mWorldViewProj worldviewproj_matrix
  param_named_auto objSpaceLight light_position_object_space 0
  param_named_auto objSpaceCamera camera_position_object_space
  param_named_auto opacity custom 55
}

vertex_program_ref tree_static_vp_hlsl
{
  param_named_auto mWorldViewProj worldviewproj_matrix
  param_named_auto objSpaceLight light_position_object_space 0
  param_named_auto objSpaceCamera camera_position_object_space
  param_named_auto fogParams fog_params
}

fragment_program_ref tree_static_fp_hlsl
{
  param_named_auto ambientColour ambient_light_colour
  param_named_auto lightColour light_diffuse_colour 0
  param_named_auto fogColour fog_colour
}
```

The first shader should be used by both high and low detail models of the tree. The next two shaders by the high and low detail model, respectively.

Tree blending uses a pre-generated noise texture for blending without transparency. There are various sizes of these textures in the `'media'` directory. A noise texture has to be used in the second texture unit of a material for the blending in the shaders to work.

Trees use a different material for static geometry. When building static geometry, the program searches for a material with the `'_static'` suffix (e.g. `'myMaterial_static'` for a tree with material name `'myMaterial'`). If no such material is found, the existing material is cloned and simplified. The static material should use the last two shaders.

Also, all tree materials have to use alpha rejection (e.g. `'alpha_rejection greater 64'` in material script syntax).

Of course, you can copy and modify the shaders as needed (they are

located in the `'media'` directory).

Note: The Ogre export utility may make a few errors when exporting a tree model from 3ds max or Maya. When building static geometry, it is important for the bounding boxes of the trees to be correct, but unfortunately the Ogre export utility does not export them correctly in some cases when exporting from Maya. I recommend converting the exported `.mesh` file into an `.xml` and then converting it back to a `.mesh` file using the Ogre XML converter. By doing this, the bounding box information is discarded and Ogre is forced to calculate a new bounding box based on the vertex data of the model.