Using PagedGeometry
Tutorial 3: Grass and Shadows

Introduction

At this point you should be fairly familiar with the basic use of PagedGeometry to add trees and bushes to your world. Now it's time to add some grass. This tutorial explains the use of PagedGeometry's GrassLoader module to add animated grass to your scene, and how to apply shadows (lightmaps) to your grass and trees to greatly improve the graphical realism.

Tutorial Requirements

- ✔ Tutorial 1 (#2 also recommended)
- ✔ A basic understanding of Ogre
- ✔ Some experience using C++

See also

- ◆ Example 4 - “GrassLoader”
- ◆ Example 7 - “Lightmaps”

GrassLoader

GrassLoader, like TreeLoader2D and TreeLoader3D, is a module that provides the core engine with the geometry you want rendered. Using one of the TreeLoader classes to add grass would be extremely tedious and inefficient, since each blade or clump of grass would need to be added individually. For this reason, GrassLoader exists to provide you an easy to use, efficient grass renderer.

It is important to note that GrassLoader differs in two significant ways from the other PageLoader's:

1. You can't use ImpostorPage to display grass. Since grass is displayed using many small non camera-aligned billboards, it's already as optimized as possible, geometrically. Attempting to use impostors for grass will result in a lot of impostor rendering, a lot of wasted memory, and a lot of lag when loading new pages. You should use either BatchPage, or preferably “GrassPage” to display grass, the latter being the most optimized.

2. Fade transitions are always used on grass, and do not cost any performance. When configuring the PagedGeometry object used to render your grass, do *not* enable transitions yourself. Doing this will only reduce performance unnecessarily.

Creating Grass

To add grass to your world, you start by creating a PagedGeometry object, as always:

```
PagedGeometry *grass = new PagedGeometry(camera, 50);
```

But this time, instead of adding the usual BatchPage and ImpostorPage combination that works well for trees, GrassPage will be used:

```
grass->addDetailLevel<GrassPage>(100);     //Draw grass up to 100 units away
```

GrassPage is the most effecient LOD to use when displaying grass, and should be used exclusively for this purpose (in other words, it's *not* effecient for displaying trees).

Next, setup a new GrassLoader instance to be used with this PagedGeometry object:

```
GrassLoader *grassLoader = new GrassLoader(grass);
grass->setPageLoader(grassLoader);
```

```
grassLoader->setHeightFunction(&getTerrainHeight);
```

In addition to creating and assigning a GrassLoader instance to the PagedGeometry object, you can see this code calls setHeightFunction(). Like TreeLoader2D, GrassLoader calculates the Y positions of grasses during runtime with this function. This is usually set to a function that returns the height of your terrain at the given point, so the grass/trees/etc appear directly on the surface of the terrain.

At this point PagedGeometry is fully configured to display grass! All you need to do is add some:

```
GrassLayer *layer = grassLoader->addLayer("GrassMaterial");
```

If you ran the application at this point, you would see your entire terrain covered in grass (provided that grass->update() is called every frame, of course). addLayer() basically adds a "layer" of grass to your scene which will appear everywhere, by default. In this case, grass is added and textured with "GrassMaterial".

The GrassLayer object returned allows you to configure the grass any way you want. You can adjust the size, animation, density, etc through this object. For example:

```
layer->setMinimumSize(2.0f, 2.0f);
layer->setMaximumSize(2.5f, 2.5f);
layer->setAnimationEnabled(true);
layer->setSwayDistribution(10.0f);
layer->setSwayLength(0.5f);
layer->setSwaySpeed(0.5f);
layer->setDensity(1.5f);
layer->setFadeTechnique(FADETECH_GROW);
```

As you can see, this sets the grass size, enables animations, configures the sway animation style, sets the density, and customizes the fade technique. All these options are fully documented in the PagedGeometry API documentation, so please refer to it for more detailed information about configuring your GrassLayer.

If you want, you can add more and more layers until you have all the grass variety you want. This tutorial will just use the single grass layer created above, for simplicity.

In most cases, you won't want the grass to grow *everywhere*. You might only want grass to appear in fields, or on lawn areas, for example. GrassLoader gives you complete control over the density levels of grass anywhere on your terrain through *density maps*. A density map is simply a greyscale 2D image you provide, where a white pixel indicates full grass density, and a black pixel indicates no grass at all (of course, any shade in between can be used too). You can literally "draw" grass into your world with any paint program by simply drawing a pattern image and using that image as a density map. For example, if your density map has a big round white circle in the middle, there will be a big round white circle of grass in the middle of your terrain.

But first, you need to specify the boundaries of your terrain, so the grass engine will know where your density map is to be applied:

```
layer->setMapBounds(TBounds(0, 0, 1500, 1500));
```

The line above configures the grass layer so the density map will take effect in the square region from (0x, 0z) to (1500x, 1500z). Usually you should supply the boundaries of your terrain here, since that's often the desired behavior.

Now, use setDensityMap() to supply your density map image:

```
layer->setDensityMap("GrassMap.png");
```

Note that the density map parameter is the filename of an image, not an Ogre Material. A material wouldn't make sense here, since it's not actually going to be applied as a texture to anything.

Try running your application. The patterns of your grassmap should show up in your game world as different densities of grass. Now you have full control over where and how your grass grows.

You can get a full working example of what you learned about grass by opening the "Example 3 – Trees and Bushes" project found in the examples folder.

Adding Shadows

At this point you should be able to add trees and grass wherever you want in your world, but there's one very important aspect that's missing: shadows. Without any shadows being applied to the trees and grass, they will often look out of place against the dark, shadowed sides of your mountains. This can be fixed by applying a lightmap to your trees and grass that shades them according to the terrain.

To apply a lightmap to grass, use the setColorMap() function, like this:

```
layer->setColorMap(“LightMap.png”);
```

And that's all there is to it, basically. Just make sure setMapBounds() is called at some point, since both color maps and density maps can't be applied without boundary information.

You may notice that "set**Color**Map" allows you to not only configure the shade of grass, but the coloration as well. You can use this feature to not only shade grass, but also color it to the terrain's features. For example, if you make your grass texture white (instead of green or whatever), you can apply your terrain's texture as a color map, and the grass will match up to the terrain perfectly. This often produces a very nice, smooth look to the grass, and blends in very well to the terrain. See Example 4 for an example of this.

Applying a lightmap to your trees is just as easy as with GrassLoader, since TreeLoader2D and TreeLoader3D both include a setColorMap() function. For example, you can do this:

```
treeLoader->setColorMap(“LightMap.png”);
```

Where "treeLoader" is a pointer to your TreeLoader2D or TreeLoader3D object. Unlike GrassLayer, there's no setMapBounds() function; the color map always takes effect in the boundary you specify in the TreeLoader2D/TreeLoader3D constructor.

For a full working example of tree and grass shadows, take a look at the "Lightmaps" example (Example 7) in the examples folder.

Updating PagedGeometry

As always, you need to call PagedGeometry::update() in order for the grass, etc to be displayed

```
[each frame]
{
      grass->update();
}
```

Conclusion

As you can see, adding grass and applying shadows to your scene isn't very difficult – it's simply a matter of specifying the desired properties (grass texture, size, density, lightmap, etc). For a full working example of what you learned in this tutorial, see "Example 7 – Lightmaps", found in the examples folder.