Using PagedGeometry
Tutorial 2: Trees and Bushes

Introduction

As you've seen in Lesson 1, once you know how to set up PagedGeometry, adding trees to your world is as easy as calling TreeLoader3D::addTree(). This tutorial expands on what you already know, and explains the fastest way to add trees and bushes to your scene.

Tutorial Requirements

- ✔ Tutorial 1
- ✔ A basic understanding of Ogre
- ✔ Some experience using C++

See also

- Example 3 - “Trees and Bushes”
- Example 2 - “TreeLoader2D”

Level of Detail

If you want, you can add trees, bushes, rocks, any pretty much anything you like to your scene using the PagedGeometry object from Lesson 1. But at a certain point you may find that even with all of PagedGeometry's batching and impostoring optimizations, the frame rate might not be as high as you like.

Bushes, rocks, and other small ground details aren't nearly as important as trees in the distance. If you reduce the viewing range of this “ground clutter”, you'll get a considerable performance boost without much noticeable loss of visual quality. For example, if your trees are visible up to a half of a mile, your bushes / rocks / etc. may only need to be rendered up to one or two hundred yards. This way, you can include a high level of detail (bushes, rocks, etc.) in your scene without wasting too much time rendering them in the far distance.

Multiple View Ranges

At this point you may be wondering how you're going to assign multiple view ranges to different types of entities, etc. Fortunately, there is a very easy way to achieve this: Create two instances of PagedGeometry - one is set up to render trees how you want, and the other is set up to render bushes how you want (with the closer view range).

This is the tree setup code:

```
PagedGeometry *trees = new PagedGeometry(camera, 50);
trees->addDetailLevel<BatchPage>(150, 30);
trees->addDetailLevel<ImpostorPage>(400, 50);
```

This is basically the same as the code seen in Lesson 1, with a few changes: The camera and pageSize data is specified in the constructor, rather than being set through functions. Also, trees->setInfinite() was removed since infinite mode is already enabled by default.

As you can see, the code sets up batches up to 150 units and impostors up to 400 units. While this should work fine for trees, the bushes won't need to be rendered this far, so another PagedGeometry object will be created for the sole purpose of rendering bushes/rocks/etc.:

```
PagedGeometry *bushes = new PagedGeometry(camera, 40);
bushes->addDetailLevel<BatchPage>(80, 20);
bushes->addDetailLevel<ImpostorPage>(160, 40);
```

Note that since this new PagedGeometry object (“bushes”) is independent from “trees”, you can set up the LODs completely differently if you like. In this case, BatchPage and ImpostorPage are still used, but the

ranges are changed to roughly half of what the trees will be using.

Now it's time to assign PageLoader's to the PagedGeometry objects. Obviously, each PagedGeometry object will need it's own PageLoader object so you can add your trees and bushes:

```
TreeLoader3D *treeLoader = new TreeLoader3D(trees, TBounds(0, 0, 1500, 1500));
trees->setPageLoader(treeLoader);

TreeLoader3D *bushLoader = new TreeLoader3D(bushes, TBounds(0, 0, 1500, 1500));
bushes->setPageLoader(bushLoader);
```

Adding the Trees and Bushes

At this point you should add all your trees to treeLoader, all your bushes to bushLoader, and your scene should be ready to go. The following code randomely places trees and bushes, but is of course for demonstration purposes (normally you'd use a level editor of some sort to place them):

```
Entity *myTree = sceneMgr->createEntity("MyTree", "tree.mesh");
Entity *myBush = sceneMgr->createEntity("MyBush", "bush.mesh");

//Add trees
float x = 0, y = 0, z = 0, yaw, scale;
for (int i = 0; i < 10000; i++){
	yaw = Math::RangeRandom(0, 360);

	if (Math::RangeRandom(0, 1) <= 0.8f){
		//Clump trees together occasionally
		x += Math::RangeRandom(-10.0f, 10.0f);
		z += Math::RangeRandom(-10.0f, 10.0f);
		if (x < 0) x = 0; else if (x > 1500) x = 1500;
		if (z < 0) z = 0; else if (z > 1500) z = 1500;
	} else {
		x = Math::RangeRandom(0, 1500);
		z = Math::RangeRandom(0, 1500);
	}

	y = getTerrainHeight(x, z);
	scale = Math::RangeRandom(0.9f, 1.1f);

	treeLoader->addTree(myTree, Vector3(x, y, z), Degree(yaw), scale);
}

//Add bushes
float x = 0, y = 0, z = 0, yaw, scale;
for (int i = 0; i < 20000; i++){
	yaw = Math::RangeRandom(0, 360);

	if (Math::RangeRandom(0, 1) <= 0.3f){
		//Clump bushes together occasionally
		x += Math::RangeRandom(-10.0f, 10.0f);
		z += Math::RangeRandom(-10.0f, 10.0f);
		if (x < 0) x = 0; else if (x > 1500) x = 1500;
		if (z < 0) z = 0; else if (z > 1500) z = 1500;
	} else {
		x = Math::RangeRandom(0, 1500);
		z = Math::RangeRandom(0, 1500);
	}
```

```
	y = getTerrainHeight(x, z);
	scale = Math::RangeRandom(0.9f, 1.1f);

	bushLoader->addTree(myBush, Vector3(x, y, z), Degree(yaw), scale);
}
```

Updating PagedGeometry

And to finish, remember to call both trees->update() *and* bushes->update() every frame:

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

Optimizing

Now you should have trees and bushes rendering in a fairly optimized setup. At this point it usually helps to play around with the page sizes and batching/impostoring view ranges. Try to find values that give the best possible performance without losing rendering quality.

Optimizing Tips

- Larger page sizes render more efficiently what is on the screen, but also causes more off-screen entities to be rendered. Depending on the volume and view range of your trees/bushes, decreasing or increasing the page size may give better performance.
- Generally, you should adjust the batching range as low as possible without making the flatness of impostors obvious. However, excessively low batching ranges can actually run slower, since when impostors are rendered near the camera there can be more overdraw.
- Once you have near-optimal FPS, try moving the camera around in your game. If high speed camera movement results in stuttering frame rates, then your page size may be too large. In this case, reduce the page size until the stuttering is no longer a problem. It may also help to enable bounded mode (PagedGeometry::setBounds())

TreeLoader2D (Optional)

You may notice in the example above in the "procedural" tree / bush placement code that GetTerrainHeight() is called to get the height of the terrain at any given x/z coordinate. Obviously, the height needs to be calculated this way so trees are placed on the terrain, not under it.

If all your trees use "Y = getTerrainHeight(X, Z)" to calculate the height, you can save a good amount of memory by switching to TreeLoader2D (rather than TreeLoader3D). This version of TreeLoader3D only accepts X and Z coordiantes, and calculates the Y value using a function you provide. For example:

```
TreeLoader2D *treeLoader = new TreeLoader2D(trees, TBounds(0, 0, 1500, 1500));
trees->setPageLoader(treeLoader);

treeLoader->setHeightFunction(&getTerrainHeight);

treeLoader->addTree(myEntity, Vector2(x, z), yaw, scale);
```

TreeLoader2D is basically identical to TreeLoader3D, excapt setHeightFunction() is used to supply

TreeLoader2D with your function which will be used to calculate the Y coordinates of each tree (from the X and Z coordinates), and addTree() accepts a Vector2 (x/z) instead of a Vector3 (x/y/z) position coordinate.

Conclusion

Adding trees, bushes, rockes, and many other details to your game world is very easy, but acheiving good frame rates often takes a little tweaking. Fortunately, PagedGeometry makes it easy to configure LODs and view ranges in almost any way, allowing you to effeciently render vast forests and jungles with dense brush and vegetation.

In the examples folder, you can get a full working example of what you learned in this tutorial by opening the "Example 3 – Trees and Bushes" project.