CS 657x Project Description
Chad Austin
2004-12-16

Grass and Trees

The first stage of this project consisted of implementing a wind effect in grass and trees. Most of the work here is implemented in shaders/wind.vs. Each vertex has an offset applied to it based on the contribution of four different wind waves that move through the geometry. The effect of the wind is modulated by the vertex weight and a sloped Gaussian-ish distribution based on the distance from the wind apex to the vertex. The weight specifies how much the wind should affect the vertex; for example, points farther out on a tree or blade of grass should be blown around more.

The grass geometry is a tessellated triangle where the vertex weights range from zero, at the base, to one, at the tip. The tree is a basic fractal tree with three branches per iteration. The weights grow from zero to one exponentially from the trunk to the farthest branches.

Glow Effect

The glow around bright objects is a quite simple post-process effect. Conceptually, the algorithm works by rendering the screen into a texture, blurring it, multiplying it by itself a few times to reduce the effect of any area that isn't that bright, then additively blending it back on top of the screen. Every bright object in the scene then has a slight blurry bloom around it.

I implement this algorithm by rendering the screen to a texture, then blurring by rendering that texture to a smaller texture, and using multiple texture stages to perform the addition and multiplication steps. Most of the algorithm could be implemented in a fragment shader. One problem is that doing the blurring by shrinking the texture results in some aliasing artifacts. Another is that using multitexturing to perform the addition and multiplication steps can require a large number of texture units, depending on the exact equation you want.

Particle System

This particle system is a very primitive implementation of the design ideas in ParticleLib. The ParticleGroup and Particle classes are almost exactly the same as they were in ParticleLib. ParticleSystem is an entity Appearance implementation so it fits into the existing Entity framework. The position, appearance, and behavior of entities are specified in a map metadata file. Because all entities are transferred over the network, this means the server specifies to the clients where the particles show up in the scene.

The particles in this system can collide with world geometry. Since there is a potentially large number of segments in the scene, we optimize collision by splitting the collision map into regions.

Project Organization

There are six source files you will be most interested in: GrassState.cpp, TreeState.cpp, Application.cpp, ParticleSystem.cpp, ParticleGroup.cpp, and Particle.cpp. These are all contained in the Client module. The directory 'vc7' contains the Visual Studio .NET 2003 build system. Both the debug and the release build directories contain copies of the resource files. This is because all filenames are resolved relative to the executable file. So if you modify empyrean/resources/shaders/wind.vs, you won't see any changes. :)

To run the grass demo, open Application.cpp and change InitialState to be GrassState. To run the tree demo, do the same, substituting TreeState for GrassState. For anything else, change InitialState to IntroState. While running the game, F10 enables the glow effect. To see the particles, you must run the server and the client, then connect to the server and create a new game. The smoke effect will be to the left and the spark effect to the right a ways.