ComS 657x Final Project Report

Tyler Streeter

12/13/04

1 Overview

For this project I started with the idea of combining fire with simulated physics. Fire in most interactive applications today does not spread or affect things physically, so I wanted to try combining these two elements in my project. Specifically, I wanted to create a physically-simulated cabin that could be burned down. The three main components of this project were: 1) a description of the scene (including visual and physical properties) stored in an XML file, 2) fire that could spread from one object to another, and 3) flammable boards that could break off the main structure after burning for a while. This report discusses how the different aspects of this application were implemented and mention future work. The project website, including pictures and executables, is: www.vrac.iastate.edu/~streeter/cabin/cabin.html.

2 Methods

This section describes the technical methods used. It covers the use of simulated physics, the steps needed to create visual and collision meshes, the OGRE graphics library, and the interaction method used to enable users to carry and position objects.

2.1 Simulated Physics

The available options for simulated physics were Novodex [1] and Open Dynamics Engine (ODE) [2]. Novodex is an excellent library and is free for non-commercial use on Windows. I chose ODE because it is Open Source, free to use on all platforms, and I have used it before successfully.

Two limitations of using ODE were that it's API is not terribly simple and that it lacks some features I needed (e.g. switching easily between static geometry and dynamics objects, an intuitive combination of rigid bodies and collision meshes, loading of scene descriptions from files, etc.) I worked with Andres Reinot and Alan Fischer to develop an abstraction layer called Open Physics Abstraction Layer (OPAL) [3]. This Open Source library provides an abstract API for simulated physics that can be extended to use other physics simulation libraries (currently it only supports ODE). One of the features I implemented for this project was an XML file importer. This loads XML files which describe scenes of physical objects, including solid bodies and joints, making it possible to store "blueprints" for vehicles, ragdolls, buildings, etc. Each solid body node in the XML tree stored data on the object's dimensions, transforms, material parameters, and any extra user-defined parameters (e.g. which visual mesh to use for a particular solid body).

2.2 Fire Implementation

The fire was created by positioning a bounding sphere around each fire source. At a random time between three and ten seconds, the fire would check which

flammable objects were intersecting its sphere. The relatively long times between these extra collision checks kept the simulation fast and added to the fire's unpredictability. New fires would start at the intersection points. A fraction of the fire sources had light sources attached; these lights flickered at times and grew dim before the fires disappeared. Care was taken to keep the total number of fires manageable since each one caused a significant computational burden. At first, each board on the cabin was a static object. After burning for some time, however, the boards became dynamic and fell off the cabin. This enabled the whole cabin to collapse after burning for a long time.

2.3 Content Creation

I needed to have an effective system for creating XML files for OPAL. Two alternatives I considered were writing an exporter for 3D Studio Max [4] or for Blender [5], an Open Source modeling tool. 3D Studio Max has its own scripting language for writing things like exporters. Blender uses Python for its scripting system. I chose to write a Python exporter for Blender for three reasons: 1) I wanted to learn Blender since it is free and Open Source (I can Blender skills with me anywhere without having to buy an expensive modeling tool), 2) I wanted to learn Python anyway, and 3) since it uses Python for scripting, it has a lot of libraries available from the Python community.

Blender's Python API exposes all the necessary data for each object in a scene. I used Python's built-in XML library to generate a DOM tree of all the objects in a Blender scene. It was fairly simple to create an exporter that allows users to select objects to export and save the XML tree to a file for OPAL. Additionally, I used Blender to generate the visual meshes for my objects. I used an existing OGRE (see below) exporter for Blender to save these meshes in OGRE's mesh format.

2.4 OGRE Graphics Engine

I used OGRE (Object-oriented Graphics Rendering Engine) [6] to render the graphics. OGRE has a simple API and contains several useful features I needed for my project. Besides learning to use its core functionality, some of the features I used were its material and particle scripting systems and shadow generation. Material scripts in OGRE consist of lists of named materials. Each material contains techniques, passes, and texture units. Different techniques for a single material describe distinct ways to render a particular material; if the primary technique isn't support on a given graphics card, the next technique is used. Also, techniques can be used as different levels of detail. Every technique contains one or more passes to be executed during rendering. Each pass contains lighting information and zero or more texture units. A pass can also refer to external vertex and fragment programs.

OGRE's particle scripts contain one or more particle systems, each with its own emitter and affectors. Every particle system refers to a material in the material script files which tells the rendering system how to draw the particles. In this project I used separate particle systems (and materials) for the smoke and the fire.

The shadows in OGRE can be setup in various ways, but every method has benefits and drawbacks. The

possibilities include additive stencil, modulative stencil, and modulative texture shadows. I had a problem when I compiled my application on a machine using an ATI video card and ran the application using an nVidia card. I think this problem is due to OGRE's automatic vertex program generation for stencil shadow calculations. The vertex program generated on the ATI card machine was not compatible with the nVidia card.

See the OGRE manual for a quick overview [6] of other features.

2.5 Grasping Interaction Methods

To enable users to start fires in arbitrary locations, the application needed a simple way for them to pick up and position objects. My solution was to use a spring system that positions and orients an object in front of the camera whenever the user wants to pick up something. (Clicking a mouse button casts a ray into the scene and finds objects within grasping distance.) This spring system was added as a feature to OPAL. The benefit of this system is that the grasped object moves realistically; it doesn't just pop into place in front of the camera. Users can even throw objects by moving the mouse to one side quickly before releasing the mouse button.

3 Future Work

This section discusses possible features that could be added. First, the scene needs more art content. Trees surrounding the cabin, a detailed ground texture with dirt paths, and more burnable buildings would add to its immersiveness. OPAL could also use a better system for content generation that integrates visual and collision mesh generation in a single task. For example, a Blender script could take a visual mesh and automatically generate a collision mesh to fit its bounds. This could also be implemented in the OPAL library itself. The physics simulation package Novodex does this pretty well; it fits collision boxes (using an octree) or convex hulls to arbitrary visual geometry.

Sound effects should also be added. Ambient noises from the woods at night (e.g. owls and crickets) and fire pops and hisses would make the environment more believable. Each fire source could emit noise, making a loud roar after a while. Finally, I would like to make this application work in VR Juggler. I think it would make a great demo for a cave display with a tracked glove used for interaction.

References

1. Novodex, www.novodex.com

2. Open Dynamics Engine, www.ode.org

3. Open Physics Abstraction Layer, opal.sourceforge.net

4. 3D Studio Max, www4.discreet.com/3dsmax

5. Blender, www.blender.org

6. OGRE, www.ogre3d.org