Proposer

Robert R. Lewis
Associate Professor of Computer Science
Washington State University, Tri-Cities
2710 Crimson Way
Richland, WA 99354

(509) 372-7178 (office)
(509) 372-7219 (fax)
bobl@tricity.wsu.edu
http://www.tricity.wsu.edu/~bobl

Title

coaster: Teaching Computer Graphics Incrementally

Abstract

coaster is a project that teachs a semester-long introductory computer graphics class by means of ten programming assignments. The assignments are incremental -- each one building on the previous ones -- and ultimately require implementation of most of the course content in the final one: a first-person rollercoaster simulation. Briefly described, the assignments (and their course contents) are: circles ("warmup", 2D graphics, applying trigonometry), wire track (3D graphics, parametric curves), wire car (meshes), "hedgehog" car (face and vertex normals), shaded car (lighting models and vertex shaders), shaded track (extrusion, model transforms), surfaces (Bezier surfaces, height maps), first person (viewing transforms, animation, splines), dynamics (physics-based modeling), and textures (textures, pixel shaders). There is also an eleventh project of the student's own (approved) design.

Students are provided with template code for the first ten programming assignments. The languages used are C++ on the CPU and GLSL on the GPU. Students are presumed to have access to OpenGL/GLSL 3.3/3.30 and the GLUT and GLEW libraries.

The class is taken by both undergraduate and graduate students. and has been presented twice at Washington State University, both times with about half of the students on a remote campus receiving it as a live telecourse. Student response has been very positive.

The goal of this lightning talk is to elicit interest from the computer graphics teaching community in making *coaster* systematically available to other universities by providing source code and training to instructors.

Description

For each topic in a typical computer graphics class, there are often one or more demonstration programs that the student can run and, perhaps, modify in some way. In general, each of these programs is "stand-alone". This prevents students from building larger, more impressive (and more motivating) programs.

In addition, many topics are covered only in lecture or are expected to be acquired from reading the textbook. If performance on the final exam is any indication, it's been my experience that neither of these approaches are entirely successful.

coaster was created with the understanding that implementation reinforces theory. The things that excite me about it (and will, I believe, excite the SIGCSE audience) are:

- Students were asking questions that showed better understanding of the concepts than I've seen my the previous 11 years of teaching this class.
- Students ended up with a program they can show off to friends, family, and perhaps future employers.
- The incremental approach used for coaster could be adapted to other courses (e.g. a compiler or AI project).
- A completed set of *coaster* projects (my own) makes a great presentation for recruiting and outreach as a way to emphasize how important learning math and physics are for computer science.