CSP2307 Introduction to 3D Game Programming

Workshop 4: Vertex buffer objects and timing (a.k.a. Bobcat Rampage!)

Experiment with timing and Vertex Buffer Objects.

Value: 5 Marks (assessed during the next workshop)

Preparation:

Download the code for this week's workshop from blackboard. Compile and run. Bobcat Rampage is played with the arrow keys – destroy the houses and run from the pursuing police cars. Police cars home in towards you, but try to avoid houses (and each other), use this strategically to avoid them.

The game could use several improvements (your job over several workshops), look through the code to see how it is constructed. The game shows examples of:

- Hiding functionality in a class to avoid code duplication
- Using the vector container type, along with an iterator to step through its elements.
- A basic model loader for the .ASE model type (currently loads only geometric information – vertices and triangles).
- Bounding sphere collision detection with a method to determine the bounding sphere radius of a model.
- Hierarchical transformations – each geometry object can have 'children' whose transform is relative to the parent.
- A simple 'flocking' style algorithm for the police cars.
- Inertia implemented for the excavator (bobcat) control.

Procedure:

Task 0: Working with VBO's (0 Marks)

Even though this is not worth any marks, it's important to do and understand what is going on. For some hints, download: workshop4_VBOhints.zip from blackboard.

Take your code from workshop 3 (the one where you have a method for drawing a cube). Change it so that instead of a cube a single triangle is drawn using a vertex buffer object with an index buffer.

To do this, you will have to use the GLEW libraries – the easiest way is to download the code for this week, copy the GLEW directory and the glew .lib and .dll files to your project, put in the #include and #pragma calls (see the workshop code) – **and don't forget to call glewinit() before calling any of the VBO functions.**

Once you have the triangle going, extend the program to draw one face of the cube (2 triangles), and then to draw the whole cube.

Task 1: Add 3rd person mode (1 Mark)

Starting with the code provided for this workshop - change the program so that the TAB key switches between 1st and 3rd person mode. To implement this, pass a flag into the render method (need to change the definition of the method) and set the viewing transform accordingly. (Hint: Use the position and orientation of the excavator.)

Task 2: Add an FPS counter (1 Mark)

Calculate frames per second and display it in the title bar of the window. A class (gameTimerHighPerformance.h) is included in the project – feel free to modify it to suit your needs (or use timeGetTime()) , then use it to figure out FPS.

Task 3: Draw using Vertex Buffer Objects (3 Mark)

Currently, Geometry objects are drawn using the method '`drawOpenGLImmediate()`'. Add functionality to the class to draw using Vertex Buffer Objects (do not change the drawOpenGLImmediate method – add a new method to the class).

- You will also need a method to create the VBO – do this on program start-up (when each object is created) or if the VBO for an object is no longer valid.

NOTE: You may notice that this code runs slowly – this is for several reasons. A major reason is the use of the 'for each' construct when iterating across objects. This construct makes a copy of the object in question. To make things faster, use an std::iterator object (look up on how to use them).