Assignment 3: collision detection, textures, blending

Due: March 21st, 2011; 11:59pm

Updated 3/9/11 – see collision detection I have reduced the requirements and added EC

ASK QUESTIONS EARLY – I WILL BE OUT OF THE COUNTRY WHEN THIS IS DUE AND MY E-MAIL REPLIES MAY BE SLOWER

You can use the same 3D models as before, create/obtain new ones, or use ones from your final project.

(template – also on blackboard) –This includes a new importer that will import and render many different types of models and texture formats (thanks Alex!). It is better than GLM. You may use it if you like or you may keep using GLM. Unfortunately, it only works in windows right now. Also code sample 1 will be really helpful for parts 1, 2, and 4.

Objective: Create an interactive experience that includes 3D OpenGL and the following topics:

1) (10 pts) Camera movement, lighting and shading, models. Feel free to build off of the previous assignment. You only need one type of camera, but it needs to be user controlled as in the last assignment.

2) Textures (20 pts): create a skybox – This is essentially a cube that surrounds your viewing area and has textures mapped to each face that look like a sky. Do not try to import a textured model for this. Use the importer code to load images and draw using glVertex and OpenGL commands for managing textures. Take a look at the `LoadGLTextures()` function in loader.cpp for an example of how to use the included DevIL library to load images. You will have to adapt this to load the 6 skybox images specified by paths and names. Feel free to obtain the images themselves from the web. With proper images, the skybox should appear seamless in OpenGL.

Grads: Heightmap: create and import an image that represents heights of some terrain. Render the 3D terrain in Opengl and add a texture to the terrain (e.g., some grass). There is a nice heightmap terrain tutorial at lighthouse3d

3) Collision Detection (50 pts): Shoot a projectile (e.g. paintballs, rain, cream pies) that uses a bounding sphere. The projectile should collide with and bounce off **(at least one)** of the objects in your scene. Do axis-aligned bounding boxes (50pts), oriented bounding boxes (50pts +EC) , or per-triangle calculation *(if you are brave* 50pts *+ more EC)*. *In all cases when using a bounding box, allow the user to toggle the drawing of the box by pressing 'b' (you can use GL_LINES for drawing it – remember to disable GL_TEXTURE_2D and GL_LIGHTING when drawing the box).*

Tips for success

1) test collisions and response between spheres and untransformed BBs before trying it with applied transforms. In fact, applying transforms to anything collidable besides the sphere projectile is NOT REQUIRED, but is worth some EC.

2) use spheres for the projectiles (you can basically treat these as points) and then you do not need to implement separating axes with OBB. There aren't other collisions that are required, so no need for OBB-OBB collisions.

3) draw your bounding boxes so you can see them (this is actually required)

4) graduates : Don't worry, I don't expect terrain collisions

4) Blending (20 pts): Implement transparency on at least one object. This should look as if the object was made of a semi-transparent material, like window tint on a car. Also use additive blending (GL_ONE,GL_ONE) on at least one object.

Turn in using one of the following options AT YOUR OWN RISK:
Blackboard
Email to TA (Sohel Hafiz sohelicpc@yahoo.com) – or use yousendit.com if you have large files
Post on web and email the URL to the TA – time stamps should be consistent with due date.

NOTE: Do not bother including exe files since some email filters will not allow this.