Computer Graphics

Assignment 5 – 3D game or animation

Handing in the Assignment

This assignment may be done as an individual project or in groups of no more than two people. This will not affect grading in any way.

All source code and an out-of-the-box executable program should be handed in as a single .zip, .rar or .7z file on Canvas.

In most cases you should not use the incremental motion operations of the ***Camera*** (or *ViewMatrix*) class. An exception would be a first person shooter type game. Otherwise just use ***look*** each frame.

Requirements:

Students must find ways to use and implement the following:

- Lighting (colors of lights and materials, position and motion of lights).
- Textures.
- Meshes with lists of vertices, normals and polygons (and texture cordinates?).

Not everything needs to be textured. An interesting application of textures is a "skybox", a huge cube surrounding the whole 3D area, with textures on the inside, serving as everything that can be seen at a distance.

Objects can be made using model or mesh loading classes that have been discussed in lab class or by loading objects from 3D design studios. It is fine to find such code online and utilize in students' projects. Something along these lines should be done for at least one object. It is not necessary to display all objects in this way and a whole lot can be done using basic forms like spheres and cubes, scaled and assembled in different ways. That's often more involved and fun too.

Students should consider implementing some of the following:

- Motion along calculated smooth curves (requirement in 3D animation).
- Models or surfaces calculated from functions (Bezier patches, sheres, blobs, etc.). These can then also be texture mapped.
- Levels of Detail (LOD). Let distance decide with how much detail an object is drawn each frame.
- Blending and transparency.
- Billboarding (sprites). Putting textured 2D polygons into a 3D environment to represent more complex objects.
- Height maps for landscapes.

Billboarding, transparency and all material related to OpenGL is well documented online and in books such as the "OpenGL Programming Guide" which has seen several versions, some of which are available at the library. The structure from assignment 3 and 4 can be used as a base but in many cases it's better to start somewhat fresh.

Students can implement anything that falls under the description but following are a few suggestions. You can also mix these suggestions together or make different versions of them. For instance a single player FPS would mean you don't need to do networking but you would need to work on the behavior/motion of other objects/characters or game elements instead.

3D racing game
Here we need some track which the player cannot go too far off. For instance you can implement gates which the player has to go through in a certain order. Cars (or Spaceships or whatever) would then drive the course from start to end, or a number of laps and whoever finishes first wins.

This can be implemented as a two player game on one computer with a split screen. One player would have a set of keys to control and would see his car and the whole scene from his point of view on the top half of the screen but the other player onthe bottom half. This requires going through the display routine twice (apart from Clearing and swapping the buffers), only changing the Viewport and the camera position/orientation between passes. Otherwise the whole scene is being drawn for both players. If the view is from inside the car then your car is probably not drawn but the opponents car must be drawn in each viewport.

A racing game could also have computer controlled cars or a two (or more) player networked setup.

3D puzzles and games (mouse or touch possibilities)
3D tetris, 3D breakout, 3D puzzles. Here you could make the gameplay 3D, or do a 2D game with 3D graphics as long as some of the animations happen in three dimensions. For example, breakout with cubes that have some 3D motion when they break or are thrown off the screen. Gameplay can be 2D but the view and graphics need 3D elements. Sky's the limit for what can be implemented here. In many cases it would be important to be able to move the camera in a sensible way to view the game from different angles.

Collision flyer (accelerometer game but possible on desktop)
Like paper plane. See the back of the plane and move mouse or use arrow keys to steer. Make sure not to hit obstacles.

3D animation (no input required)
This is a 3D realtime program that runs without any input from the user. It should run exactly the same every time it runs and has to make sure the time management can draw any frame at any given time (in seconds, not number of frame) exactly as it should be, regardless of what has already happened or how fast the computer is drawing it (how many frames per second).

This program has to include some cuts (changes of the camera's position) and smooth motion of the camera and some objects, at least once (preferable more) a motion by a Bezier Curve with at least 4 control points.

This project gives the most space for graphics (rather than collision, physics, gameplay) so students should consider the forms they're using, the motion of the camera, the lighting and changes in lighting, and so on. Also changes in the perspective angle (zoom) can be interesting.

First person shooter
Here you can start with your maze program and use that as the level. Also use the Camera class to keep track of your own positionn and set the Camera. This program needs a network connection to be made, and the player's status to be sent to the other participating computer(s). Each program then has to have a camera set at their own position but draw some model at the position sent from the other participant(s). There has to be a way to shoot, calculate collisions between shots and players (and walls) and let all participants know what the status is.

The graphics here need to be more involved than in the second assignment but a lot of the effort here will be on collision detection, networking and game status.