

Biofeedback Visualisation

Programming for Entertainment Systems: Coursework 2 Documentation



General notes:

Because this application is meant to be the starting of a biofeedback visualisation engine, direct interaction isn't the objective. Instead imagine that the values represented by analogue signals are something like temperature and galvanic skin response. Similarly there is a heartbeat function but currently this shouldn't be considered too literal. In the absence of actual hardware being plugged in, the same values can be manipulated using sliders and a button on the interface.

Also, the camera is fully mobile with WASD or Arrow movement keys along with up and down motion and mouse tilt.

A number of elements respond to the board information in different ways:

Two of the three lights respond directly to the information from the two analogue signals through their speed, strength and erratic choice of direction.

The third light is indirectly affected by these signals since its behaviour is to alternate between chasing the other lights, the moment it has caught up with one it chases the other. It isn't as autonomous as this however, since it still chooses its own directions to travel in.

Similarly and finally, the fish's motion is affected by the behaviour of all three lights, continually chasing after the centre point of the group.

The terrain is generated by loading a .raw image file and using the pixel data to represent the Y position of a grid of vertices.

Starting Point:

I have used Microsoft Visual Studio 2008 (version 9) and DirectX 9 to produce this work. All the models and graphics were produced by me. I have started with nothing but the Simple Sample from Microsoft's June 2005 SDK and a header file for C++ for connecting to the circuit board's DLL which needed changing to function correctly.

Performance:

A pretty clear performance issue was the mixing of the slower USB interface with a graphics system being run at full throttle. So the two functions have been logically separated with the USB board updating global variables at a timed interval while DirectX renders at its own pace.

Lighting can also cause a considerable drain on GPU performance, as well as the fact that there are a finite number that can be switched on at any one time. For this reason I have used a maximum of four since it is pretty standard for hardware to support 8. I have also avoided using a spotlight since the calculations involved for this light type are categorically the worst hit on lighting performance.

Evaluation:

Given more time the intention was to make an automated camera which followed a set course until the user overrides this with mouse or keyboard input (which would then restart after a period of user inactivity).

Secondly there is an empty waterPool class which would have created a similar vertex and indices buffer to the terrain except that it would simulate a water surface which I could have directly manipulated the Y values of vertices in order to also respond to biofeedback input.

I should have also used the built-in checks for the creation of lights that determine the hardware's support for them but for the purpose of demonstrating the lighting effects they are generated irrespective of the hardware.

The Program:

The main program handles the DirectX render loop and updates from the interface or USB circuit board. It is also responsible for initialising, updating, rendering and releasing the classes used in for the rest of the project.

mobileLight class

This class creates the three mobile point lights that fly around and determine their behaviour based on updates from the main application (USB or interface). It is capable of different types of behaviour, but all the lights involve random motion and properties determined by the analogue inputs.

floorTerrain class

This class will load a raw image file to be used as a heightmap, and then generate a terrain from vertex and indices buffers using quadrangles split into pairs of triangles to link the vertices. It also applies a texture and material.

fish class

The fish class loads the meshes from my first project (fish body parts) and animates them in a similar method (flipping fins etc). It also handles the fish's overall movement behaviour. Since it follows after the centre of the lights, its movement is determined by their random motion.