CG WebGL Summative

pvxf29

CG WebGL Summative

Software Methodologies CG WebGL summative assignment.

We were tasked with creating a 3D representation of a lecture theatre using WebGL embedded in a web page.

A live demo of the site can be found here.

Note: Chrome + OS X compatible. Not tested across all platforms/ devices.

Report

Objects

All objects have been generated though the creation of cubes and Drawing of shapes utilising for loops to minimise code and increase efficiency

My scene contains a variety of static objects, from the walls of the room itself to the skybox visable through the room's windows. Also included are a variety of dynamic objects: + Blinds - Open and close, adjusting ambient lighting accordingly. Animated; transform shape. + A door - Open and close. Animated; rotate shape and translate to offset rotation. + Lights - Toggle on off (grouped by rows); change colour; play sound effects;

Texture mapping is applied on specific locations (skybox, board, and note). I previously applied textures to all surfaces but found that the overall appearance was not enhanced by this overuse of textures, so for the sake of visual appeal stuck primarily to colour mapping instead.

Light Sources

The scene includes 20 point lights and an ambient source. The 20 point lights are toggled in rows using keys 1-5. 2 of the point lights include flickering on irregular time delays. As point lights turn on they play sounds. Pressing 'm' changes the colour of the point lights randomly.

Interactions

The user interacts with the scene through two methods - keyboard input and mouse input. Initially I handled user input through the onkeydown method, which created jerky movement; onkeydown is only called ~3 times a second. By mapping keyboard inputs to a dictionary holding boolean states for the relevant keys, input and movement is tied to the scene draw tick and therefore is far smoother. This input is used to control the dynamic objects and lighting previously mentioned, as well as control the camera.

The camera represents the user, and is moved through WASD/mouse movement. The mouse controls use pointerLock to lock onto the user's mouse and adjust the camera's orientation (through tracking and accounting for x/y movement).

Because the camera's position is tracked, interactions based on proximity have been possible. There is a note on the lecturer's podium which, if moved near, brings up a screen overlay showing the note's contents.

Shaders

As the code was built in an object oriented manner, altering the shaders to include support for textures (as well as colours) was a simple case of including if statements checking for the object type and altering rendering accordingly.

Robustness

The code includes some cross platform handling and user input is heavily controlled and checked using if statements. As a result the program is fairly robust. However, some browsers may not support it (particularly the mouse control), and mobile touch interactions are not handled.

Extra Features

- A clean interface styled in CSS
- Screen overlays (for the note), through Javascript + CSS
- Audio in Javascript - background track on loop; lighting sound effects;
- Some crossplatform support included, though not thoroughly tested

Screenshots

Screenshot 1 shows the scene during it's brief introduction. The room begins in darkness (ambient light source intensity reduced to 0) and one by one the rows of lights turn on. The chalkboards are textured. Along the right hand wall a door can just be made out, which can be opened.

Screenshot 2 shows the dynamic movement of the blinds as they close (controlled by the user). As the blinds raise and lower the ambient lighting of the scene is raised and lowered accordingly. Outside of the window is a skybox. The lights can be toggled on/off, grouped by rows; only rows 2 and 4 are currently active. On the desk is a note object (see screenshot 3).

Screenshot 3 shows the user in close proximty to the aforementioned note object. An overlay view appears containing the note's contents appears, disappearing once the user moves away. By pressing/holding 'm', the colour of the lighting in the room is randomly varied, here appearing red.

Limitations

- The shaders do not including capablity for dealing with shadows so although the lighting appears somewhat convincing, it is not infact realistic.
- The current skybox is also poor and could be improved upon.
- Including 3D objects (e.g. .OBJ files) would be great, but having little experience with 3D modelling/ texturing packages and not wanting to take 3D assets from online, this seemed impractical. This would have made the code more efficient and increased rendering speeds, though the scene is not complex enough to suffer notable lag issues.

Resources

- WebGL Textbook
- WebGL Textbook examples
- FPS Tracking
- Pointer Lock API