Luc Nikiema
Computer Graphics
May 5, 2020

Final Design Document

Introduction

For this project, we were tasked with creating a 3D scene with at least 3 models (with animations and textures), lighting and a virtual camera with keyboard controls that allowed the user to navigate through the scene. I achieved this using WebGL, JSON 3D models, and .jpg images.

Code Structure

- **Text.js**
    - The HTML file calls initShaders() which fetches the vertex shader code and fragment shader code. That information populates an array called resources and is passed to myMain(resources) in CGFinal.js.
- **CGFinal.js**
    - Global Variables: Camera, objectArray (final 3D object)
    - myMain(resources) initializes the canvas, gl, and shaderProgram and then passes the resources array to shaderInit(gl, shaderProgram, resources) to compile the shaderProgram. myMain(resources) then loads all of the images to be used on the 3D models and passes them (and their respective .json files) to loadExternalJson(.json, htmlImage, gl) which is in objects.js. This function returns an array filled with meshes represent the final model.
        - **objects.js**
            - When loadExternalJson(.json, htmlImage, gl) is called, it fetches that .json file and populates two arrays with materials and attributes. That information gets passed to makeModel(modelObj, img, objMaterials, objAttribute, gl), where it iterates through both arrays and calls makeMesh(model, objMaterials[i], objAttribute[i], img, gl) which then makes a mesh object, creates and binds the appropriate buffers (vertices, normals, textures, index) in

bindAndBuffer(mesh, gl), and applies the image to the mesh in modelTexture(model, img, gl). The mesh object is then pushed onto the final object array.

- After this process is complete, initWebGL(gl, shaderProgram) is called which then calls drawLoop(gl, shaderProgram). This function calls draw(angle, gl, shaderProgram) infinitely, and passes in an incrementing value used to control the animation of certain models in the scene.
- draw(angle, gl, shaderProgram) calls matrixSetUp(gl, shaderProgram) which sets up the projection and camera matrices, as well as the lighting uniforms. It then initializes the world matrix and the model matrix location. The next steps occur in a for loop, one per object array:
  - A call to bindTexture(gl, shaderProgram, objArray[i]) which does the following:
    - Calls shadersandBuffersMesh(gl, shaderProgram, objArray[i])
      - Binds and enables attributes
    - Activates and binds texture for this mesh
    - Initialize the model matrix, apply transformations to matrix, gets model matrix location, binds index buffer, and then draws.

- **Camera.js**
  - The camera class (built upon from code example), is a consists of 3 vectors that represent the X, Y, and Z axis in a 3D space, the receiving camera matrix, and delta value. The functions control camera movement based on keyboard keys (using an event listener) as stated in the rubric from the midterm. The “J” key pans the camera to the left and the “L” key pans the camera to the right.