Chess 3D

Kyle Price
201707320

Dec 6, 2021

Introduction:

For this project I wanted to build a 3D chess game because games are very important to me and the purpose of wanting to enroll in this course was to learn about 3D graphics to be able to create 3D games. I chose chess because of my familiarity with it, and since it is one of my favourites.

Implementation:

This project consists of a full working chess application made using Javascript and WebGL. No libraries were used in this project. The bulk of the graphics implementation is handled in the game.js file, which is where most of the setup occurs and objects are created. The rest of the rendering and graphics calculations are done in the InteractiveObjects.js which is used to build a lightweight scene graph to hold and render all the objects and hierarchies in the application. For the chess game there is only a single parent object which is the textured tabletop and the rest of the objects (checkerboard, highlights, pieces) are all children of it. The models used in this project for the chess pieces as well as the textures were found online and a link to the sources can be found below.

Objects consist of two main parts: the transform and the mesh. The transform holds information about the scale, rotation, and position plus a helper function to get the resulting model-view matrix for the specific transform. The transform also consists of an extra property which is called default which is a secondary model-view matrix that will be applied to the model before the resulting matrix. This property was helpful since the models were not made for this project and had arbitrary dimensions. So by using the default property, I could set its default size, rotation, and most importantly its center/pivot location. The mesh is a container for the GPU buffers with a convenient draw function that will render the mesh for an object by passing in the model-view matrix. Before that you have to set up a model by calling the mesh.setModel(model) function which takes a javascript object with an array of points, array of normals, and the number of

triangles in the model and it will setup the vertex buffers for the given attributes. Then a secondary mesh.setMaterial(material) for setting up material properties as well. There is also an optional mesh.setTexture(texture, texCoords) which allows the mesh to have textures as well by similarly creating an attribute array for the texture coordinates and holding a reference to the texture so it can pass them in to the GPU before drawing.

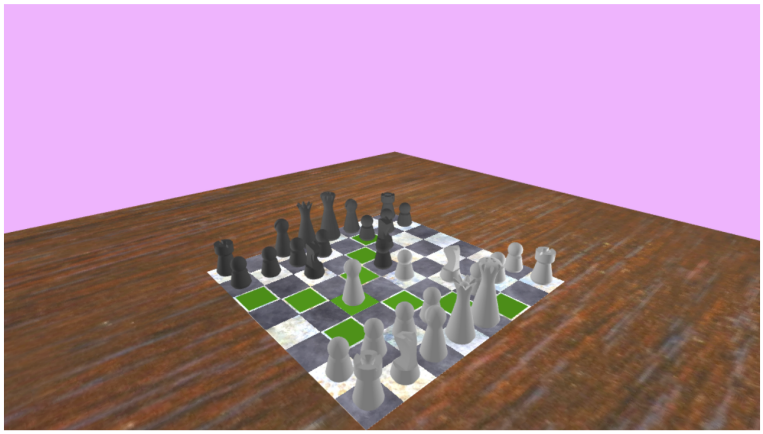
Input in the game uses keyboard input for rotating the board and miscellaneous things with lights and cameras. The colors for different objects were done encoding the positions of the board into the RGBA values. The red value corresponds to the files (columns) of the board, green values represent the rank (rows), and blue shows if it was a piece and also what player the piece belongs to. You can toggle showing the picking by colour rendering by pressing the P key.

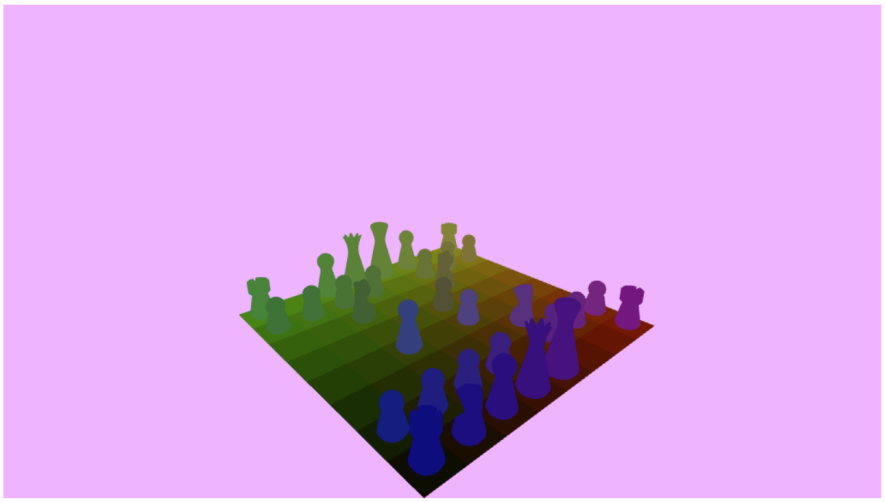
For a novel component I added a new type of shading, Gooch Shading, which is unique to Phong shading since it is a non-realistic type of shading that highlights geometric features.

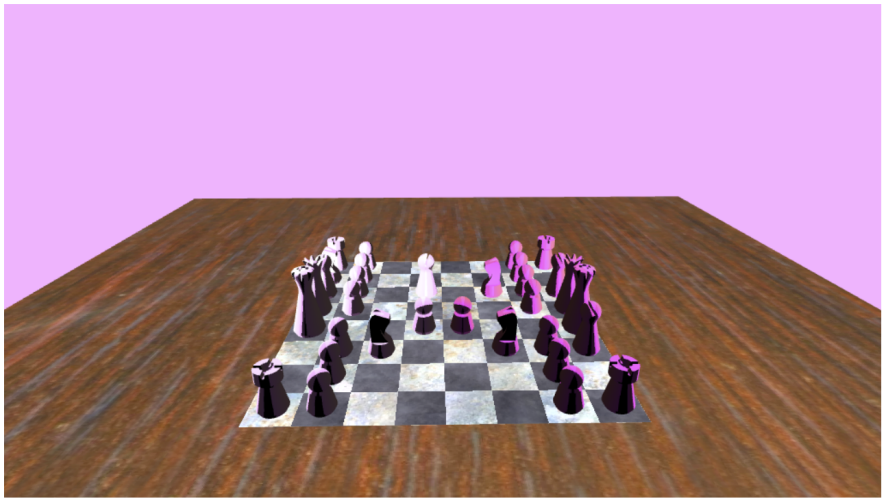
User Manual:

Click pieces to play, White goes first
P toggle picking view
T toggle overhead view
G Toggle Gooch Shading
L toggle light settings
Undo Moves with X
A/D Rotate the game board
W/S Moves the camera in and out of the board

Screenshots:







Running Application:

To run the application it needs to be hosted on a server.
Example with python in Terminal with current directory at the project folder:

~/Project $ python3 -m http.server 8000

Then it should be playable in a chrome browser at http://localhost:8000

Video Component Link:

https://youtu.be/g27wUJ36M68

External Sources:

Models: https://www.thingiverse.com/thing:2552392
Textures: https://opengameart.org/content/chess-board
Gooch Shading Example:
https://www.cl.cam.ac.uk/teaching/1718/FGraphics/6.%20Advanced%20Shader%20Techniques.pdf
Picking Example: https://webglfundamentals.org/webgl/lessons/webgl-picking.html