Goals and main task(s): Set up abstract data structures which can store data related to

rendering graphics.

Tasks: Learned how to use the GLM library

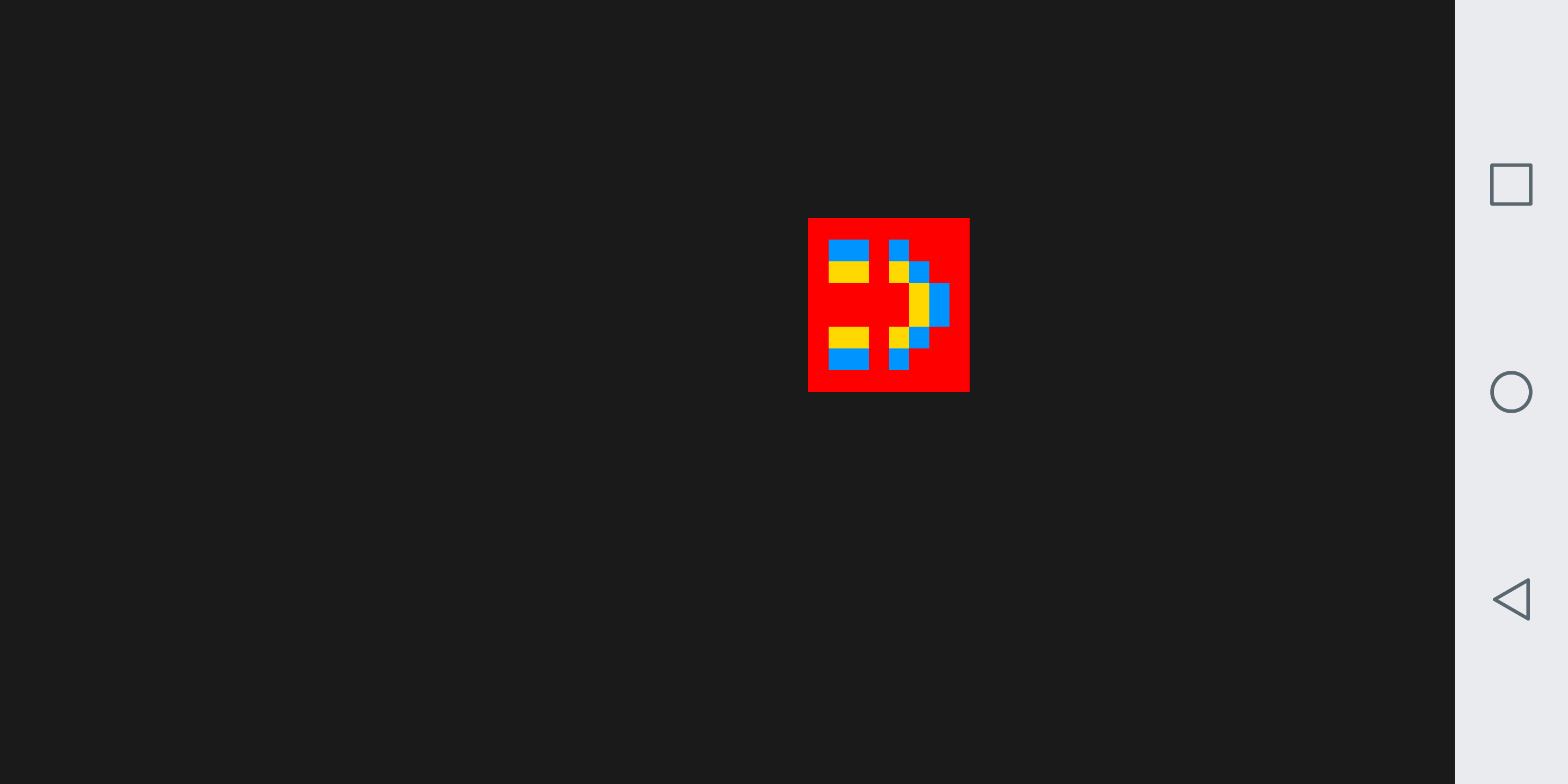
Reflections: Because data used in graphics (positions, rotations, scalars, matrices, etc.) need to be stored in computer memory, I used the GLM math library to do this for me. I first learned how to create those mathematical objects and then do computations with them. For example, I created a matrix (a table of values that is 4x4 wide and tall) to store projections (this is called a projection matrix). I then multiplied vectors that stored three-dimensional coordinates with the projection matrix to get a final two-dimensional coordinate that would be displayed on the screen. This process is known as projecting or converting three-dimensional points into coordinates on the plane of the two-dimensional screen.

Tasks: Created asset-loader and engine input/output stream

Reflections: I also finished the engine’s asset loader (an asset refers to textures, models, materials, etc.). The asset loader uses the engine’s custom input/output stream to access the platform’s file system and load the binary or text of the asset.

Tasks: Created data structures to use in rendering pipelines

Reflections: Lastly, I created data structures that hold information about graphical objects. For example, I created a data structure that holds information about light (its position, its attenuation, its intensity, etc.). In addition, I created the texture and shader data structures. A texture is just an image, and a shader is a program that runs on the graphics processing unit (GPU) to render data given from the central processing unit (CPU). I also created the following data structures: Vertex, Line, Ray, Material, and Mesh. A material defines the texture of a mesh (a three-dimensional object defined by vertices).

Below: A simple texture being rendered with a shader.