Mat:

I began by adding the first map type, Vanilla Random to the heightmap generation as well as a randomization factor called getNoise. This method has a chance to add values between -1 and 1 to the Square Step vertices. Next I implemented the use of raycasters for setting the height of models upon being loaded. Next i added some new terrain object models to be used when generating the map. I then found a way to make the models load in the center of the intended square by creating a map “unit”. The unit is the number of vertices in the heightmap divided by that number minus one. Lastly, I implemented random placement into the game board generation. It needs tweaking, but it can be easily done.

Carson:

I started this week by helping my group finalize our project proposal. From there, I began to work on creating the title screen for our game. I was planning on using the same approach other groups were using and that was to create text in the scene with text geometry, and using a raycaster to select it. However, with this approach I kept having the same issue. That issue being that my text geometry would not show up on my screen. After spending several hours over multiple days, I was still having the same problem. To remedy this, I was going to speak with another group about their approach to their title screen and see how they might’ve done it. If all of this fails, I will move to implementing a title screen with just html and css. The last thing I have done this week is I reached out to group 1 and discussed yuka.js implementation in our games since they are so similar.

Emily:

For this week, one main task I worked on was the Project Proposal. My short time enrolled in the Game Design program at Wiregrass Georgia Technical College has made me aware of how detailed a Project Proposal for a game can be. I did my best to remember some important aspects of our game to add to the proposal. Things such as game rating, targeted gender, and targeted age are only a few parts of a game proposal that not many people consider. I also worked on implementing attacking. To do this, I extended onto our existing *attack* function within the Actor class. As of right now, when the “Attack” button is pressed, the first element of the character array “attacks” the first element of the enemy array. The corresponding weaknesses and resistances are taken into account, and the hitpoints of the enemy is decreased accordingly.