Week 6 Presentation
Team 3

By Emily, Carson, & Mat

Game Recap

Tactical RPG

Linked List vs Array

- We had linked list implemented initially to hold the character objects / models as well as their attributes
- Our initial idea was to make it a circular linked list
  - Allow character swapping and cycling through enemies
- We then came up with a simpler solution that is easier to implement with two arrays
  - Character array and enemy array
  - Character swapping is only concerned with character array and enemy movement is only concerned with the enemy array

Linked List vs Array

- I ended up re-implementing our array of characters and added an array of enemies as well
- When a model is loaded, the name is checked and the corresponding Actor object from the Actor class is created
- An Actor contains attributes such as hit points, strengths, and weaknesses

```
for (const model of Object.values(characters)) {
    gltfLoader.load(model.url, (gltf) => {
        const root = gltf.scene;
        root.name = model.name;
        root.turns = 5; //determines the number of
        root.position.set(model.pos, 0.01, -3.5);
        root.scale.set(.34, .34, .34);
        //root.visible = false;
        ////////////linkedList.add(root); //add the
        if (root.name === "melee") {
            console.log("melee");
            let mike = new Melee("Mike");////////
            root.actor = mike;
        } else if (root.name === "ranged") {
            console.log("ranged");
            let rachel = new Ranged("Rachel");
            root.actor = rachel;
        } else if (root.name === "defender") {
            console.log("defender");
            let joe = new Defender("Joe");
            root.actor = joe;
        }
        charactersArray.push(root);
        scene.add(root);
    });
}//end for
```

Swapping Characters

- You can swap between characters using “r”
- The *characterCount* variable acts as the index within the character array
- By creating the handler variable, I can pass values to the event handler

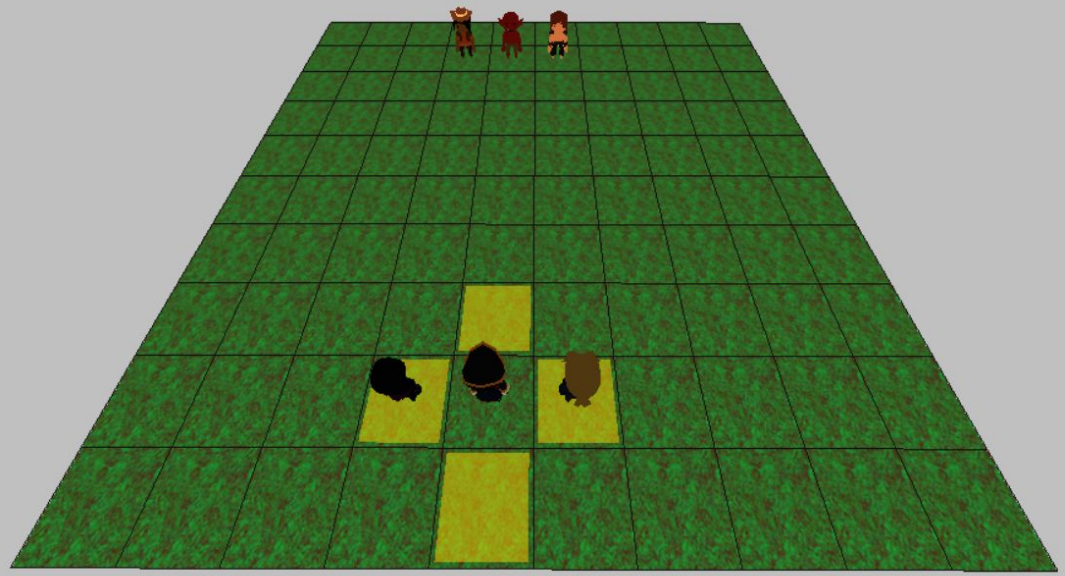
```
function changeCharacter() {
    //console.log(characterCount);
    if (characterCount < 2)
        characterCount++;
    else
        characterCount = 0;
    return;
}
```

```
let handler = function (charactersArray) {
    return function (event) {
        if (event.key === 'w' || event.key === 'a' || event.key === 's' || event.key === 'd' || event.key === 'c')
            movePlayer(event.key, charactersArray);
        else if (event.key === 'r')
            changeCharacter();
    };
};

window.addEventListener('keydown', handler(charactersArray), false);
```

Buttons!

- Added a new HUD.js file to store the heads up display functions
- It contains *onEndTurnClick()* as well as *onAttackClick()*
  - *onEndTurnClick()* - ends the user's turn and allows the enemies to move
    - Enemy movement is in progress
    - I created a *sleep* function that will make it appear that the enemies are moving slowly
  - *onAttackClick()* - attacks a chosen enemy
    - Attacking is in progress
    - In here, the Actor's *attack()* function will be utilized, however it needs to be reworked
- The buttons are added in *index.html* and are just .html buttons, so we will need to think about how to approach a title screen given this
  - Change visibility?

Attack

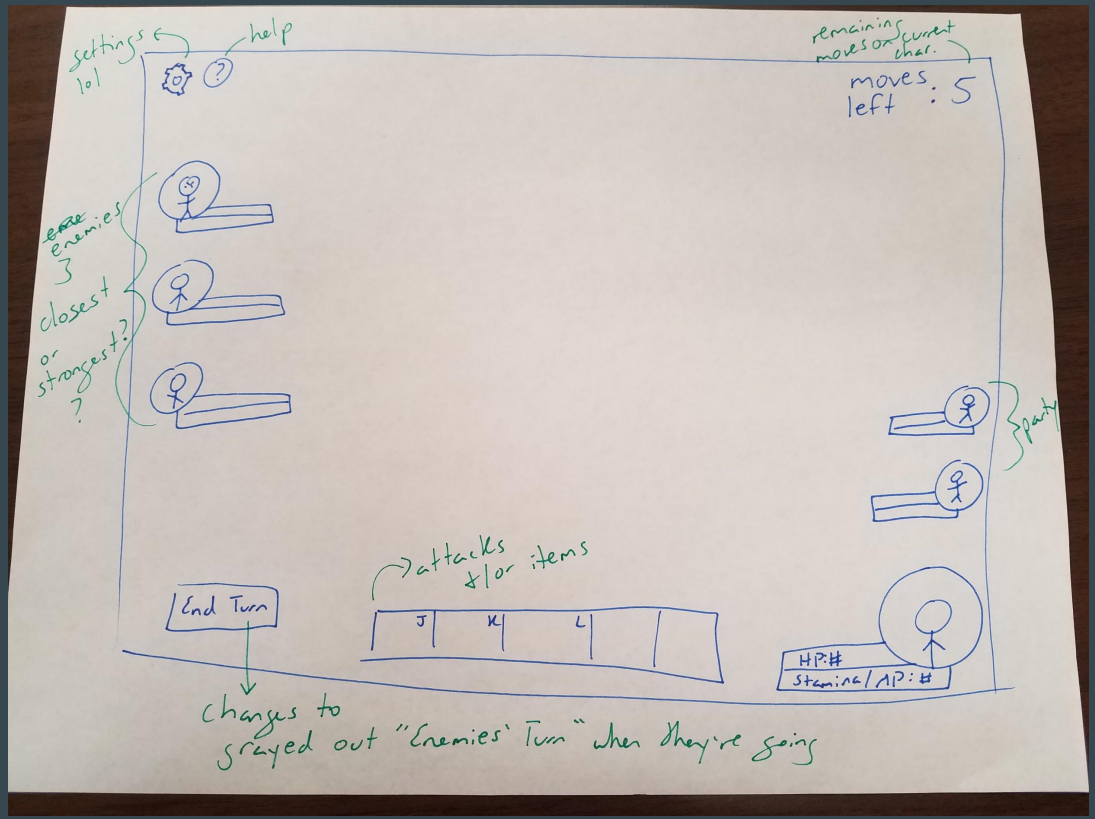


End Turn

Model Colors

- Changed the colors of enemies using blender
- Exported them as glTF files

HUD Development Idea



Terrain continued...

1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	0.75	0	0
1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	0.75	0	0
1	1	1	1	1	0.9	1	0	1
1	1	1	1	0.9	0.9	0.75	0	0
1	0.75	1	0.75	1	0.6	1	0	1
0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1

1	1	1	0.9	1	1	1	1	1
1	1	0.9	0.9	0.9	1	1	1	1
1	0.9	1	0.9	1	0.9	1	0.9	1
0.9	0.9	0.8	0.8	0.9	0.9	0.9	0.9	0.9
1	0.9	1	0.9	1	0.9	1	0.9	1
1	1	1	1	0.9	0.9	0.9	1	1
1	1	1	1	1	0.9	1	1	1
1	1	1	1	0.9	0.9	0.9	1	1
1	1	1	1	1	0.9	1	1	1

The fix...

I took Dr. Mihail's advice

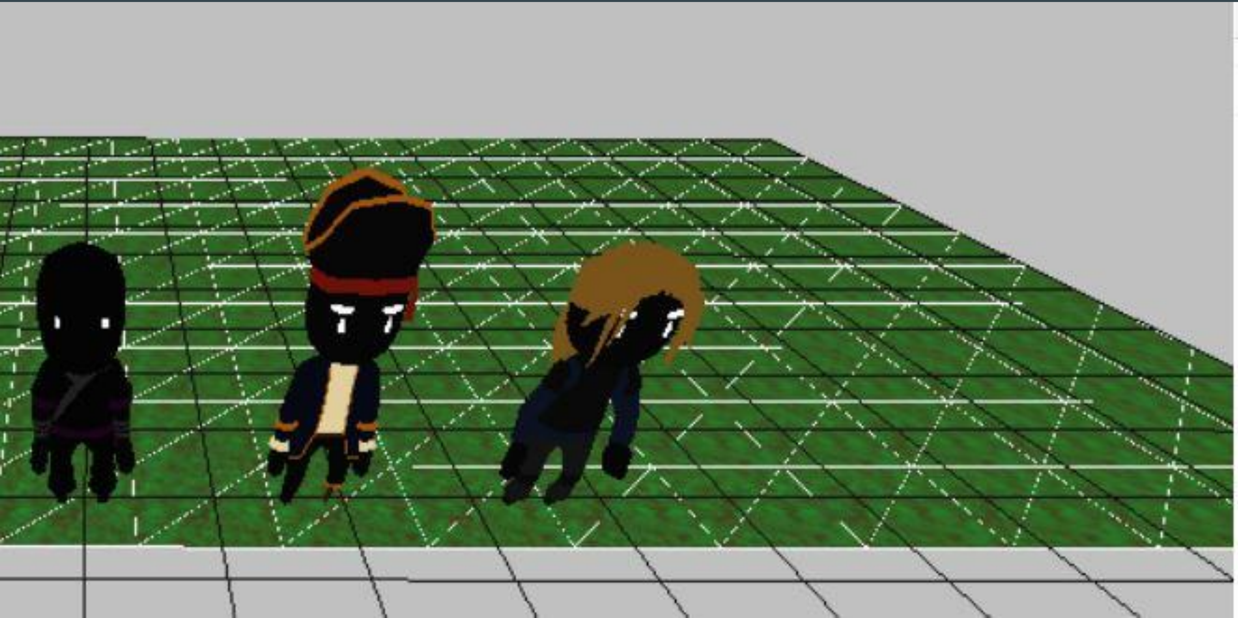
- I stopped checking neighboring squares for values
  - It worked like magic

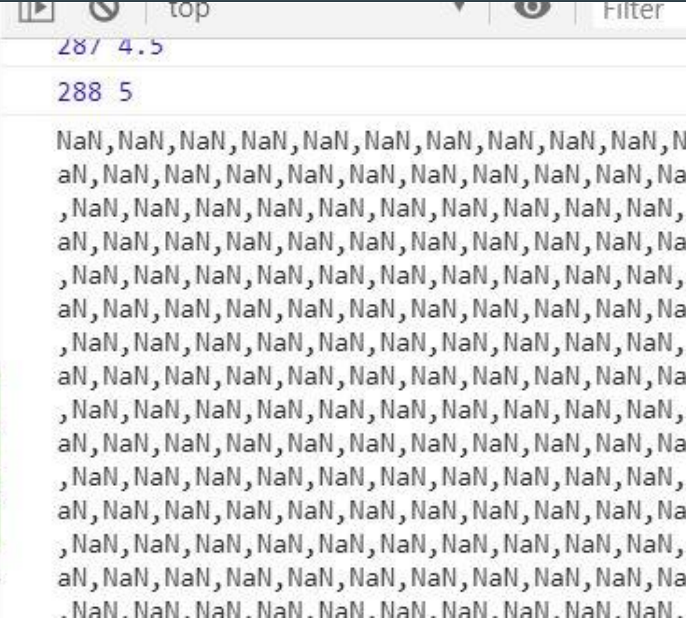
```
//Function that finds the outside center of each subarray
function diamondStep(heightMap, x, y, size){
  let mid = size/2;
  let topLeft = heightMap[x][y];
  let bottomLeft = heightMap[x][y+size];
  let topRight = heightMap[x+size][y];
  let bottomRight = heightMap[x+size][y+size];
  let center = heightMap[x+mid][y+mid];

  heightMap[x][y+mid] = customRound((topLeft+bottomLeft+center)/3); //Center left
  heightMap[x+mid][y] = customRound((topLeft+topRight+center)/3); //Top center
  heightMap[x+mid][y+size] = customRound((bottomLeft+bottomRight+center)/3); //Bottom center
  heightMap[x+size][y+mid] = customRound((topRight+bottomRight+center)/3);  //Center right
}
```

```
console.log js/heightMap.js:17
  [
    [
      1, 1, 1, 1, 1,
      1, 1, 1, 1
    ],
    [
      1, 1, 1, 1, 1,
      1, 1, 1, 1
    ],
    [
      1, 1, 1, 1, 1,
      1, 1, 1, 1
    ],
    [
      1, 1, 1, 1, 1,
      1, 1, 1, 1
    ],
    [
      1, 1, 1, 1, 1,
      1, 1, 1, 1
    ],
    [
      1, 1, 1, 1, 1,
      1, 1, 1, 1
    ],
    [
      1, 1, 1, 1, 1,
      1, 1, 1, 1
    ],
    [
      1, 1, 1, 1, 1,
      1, 1, 1, 1
    ],
    [
      1, 1, 1, 1, 1,
      1, 1, 1, 1
    ]
  ]
```

Hijinks ensued

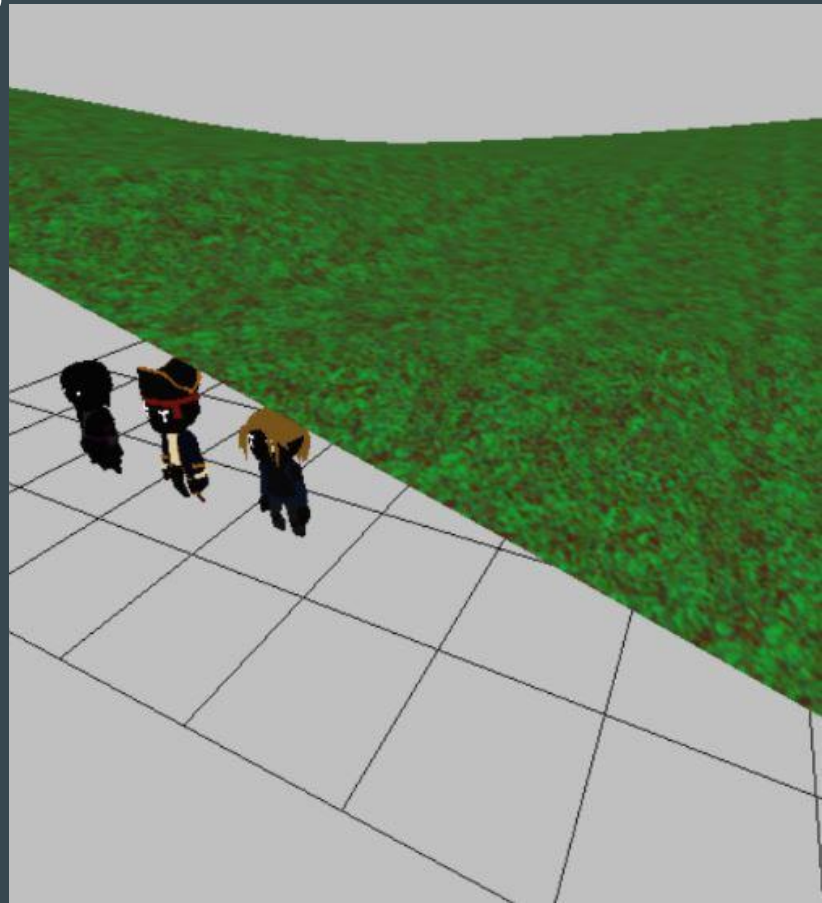
- When applying the height map to the terrain, I got some interesting results
    - My 2d height map array had one too many Ds



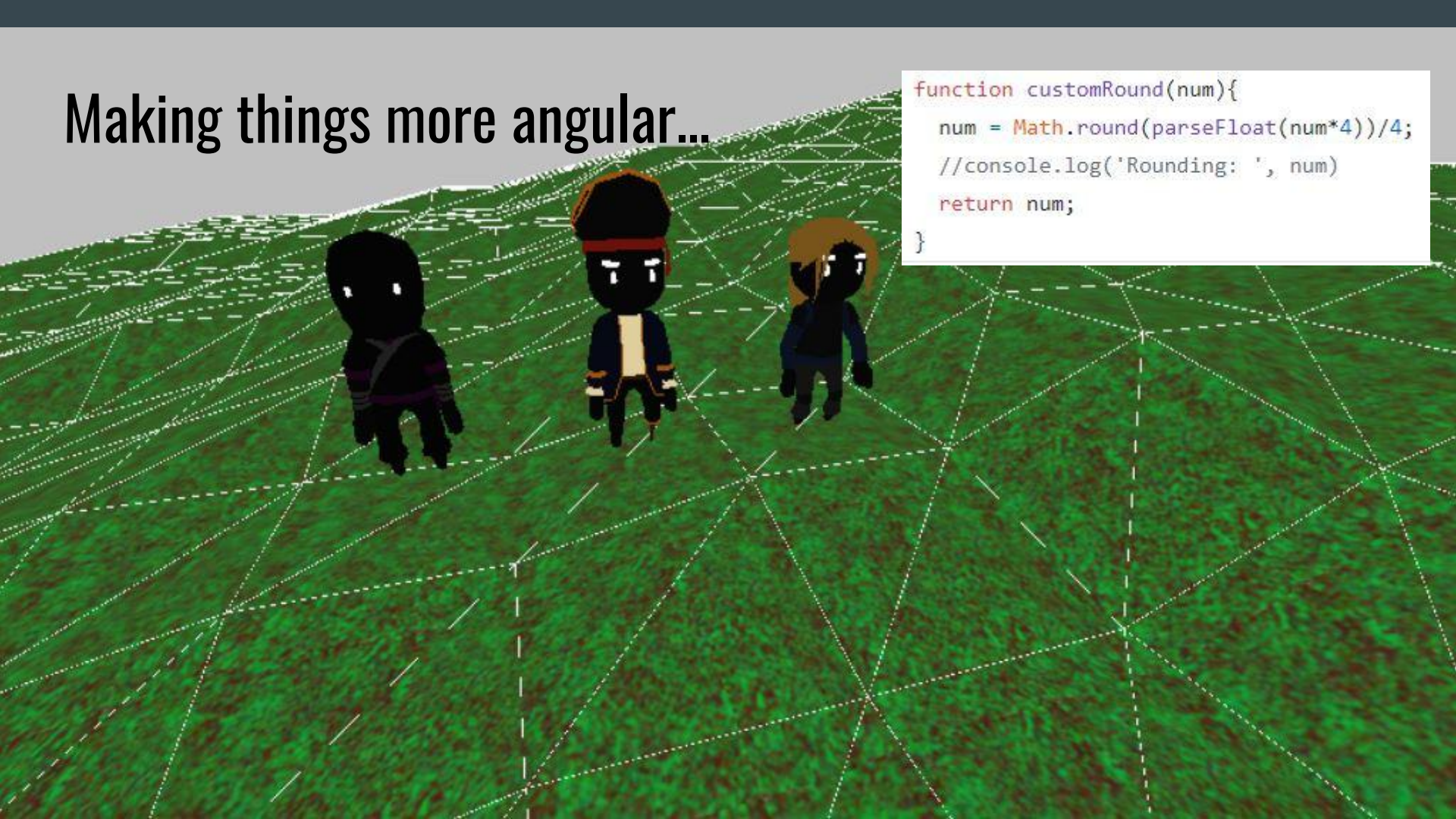


Properly applying the height values

```
//retrieve the position array from PlaneBufferGeometry
var positions = floorGeom.getAttribute('position').array;

//convert the heightmap to a 1d array
var hM = [];
for(var i = 0; i < heightMap.length; i++){
hM = hM.concat(heightMap[i]);

//apply the new array to every third entry in positions
for(let i = 0; i<(mapVerts*mapVerts); i++){
    positions[(i*3)+2] = hM[i];
}
```



Making things more angular...



```
function customRound(num){
  num = Math.round(parseFloat(num*4))/4;
  //console.log('Rounding: ', num)
  return num;
}
```

More considerations

- Dynamically placing objects at the correct height
- Drawing a grid on the terrain
  - Three's gridhelper won't help
- The next layer
  - Trees, rocks, etc...
- Randomness

Next steps

- Mat
  - Continue improving terrain
- Emily
  - Implement attacking
  - Implement enemy movement
- Carson
  - Work on the title / loading screen