# Week 2 : Group 1

Merritt Hancock, Ryan Trull, Kenda Blair, Alan Bettis

## Week 2 Tasks

- Orbit Controls
- Enemy Implementation
- Refactoring
- Grid Overlay
- Cursor Implementation
- Movement Limitations
- 3d Map Model

## Camera

Switched to OrbitControl

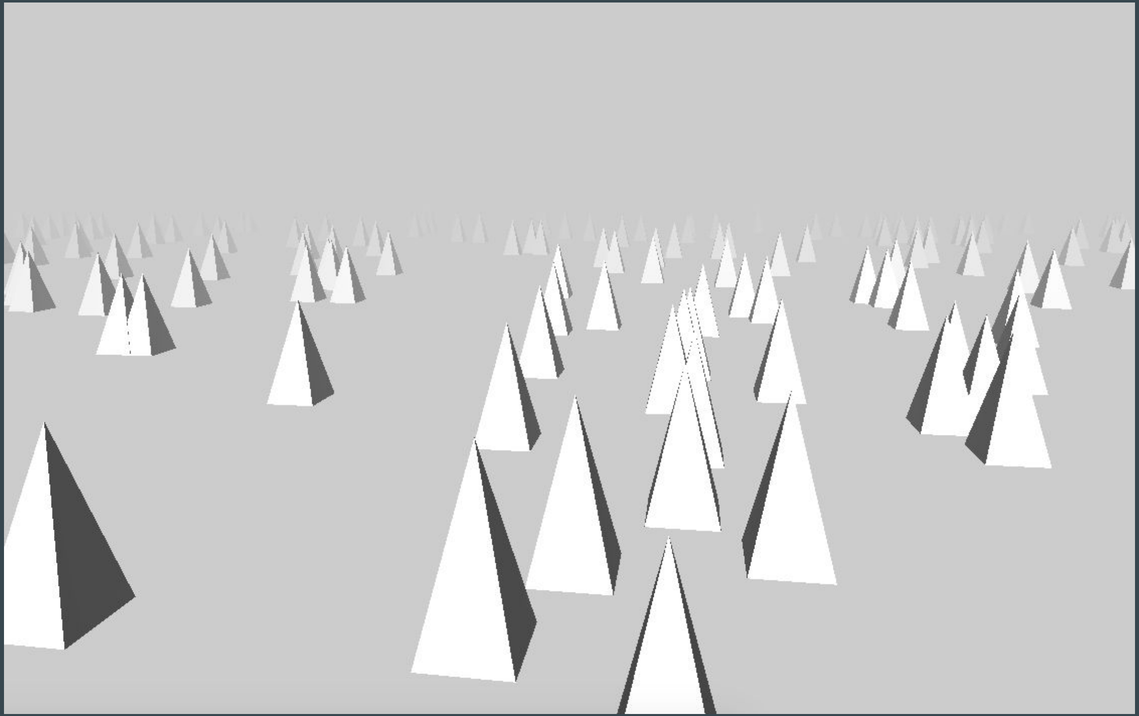
Rotate is Left click

Free move is Right click

Zoom is scroll wheel

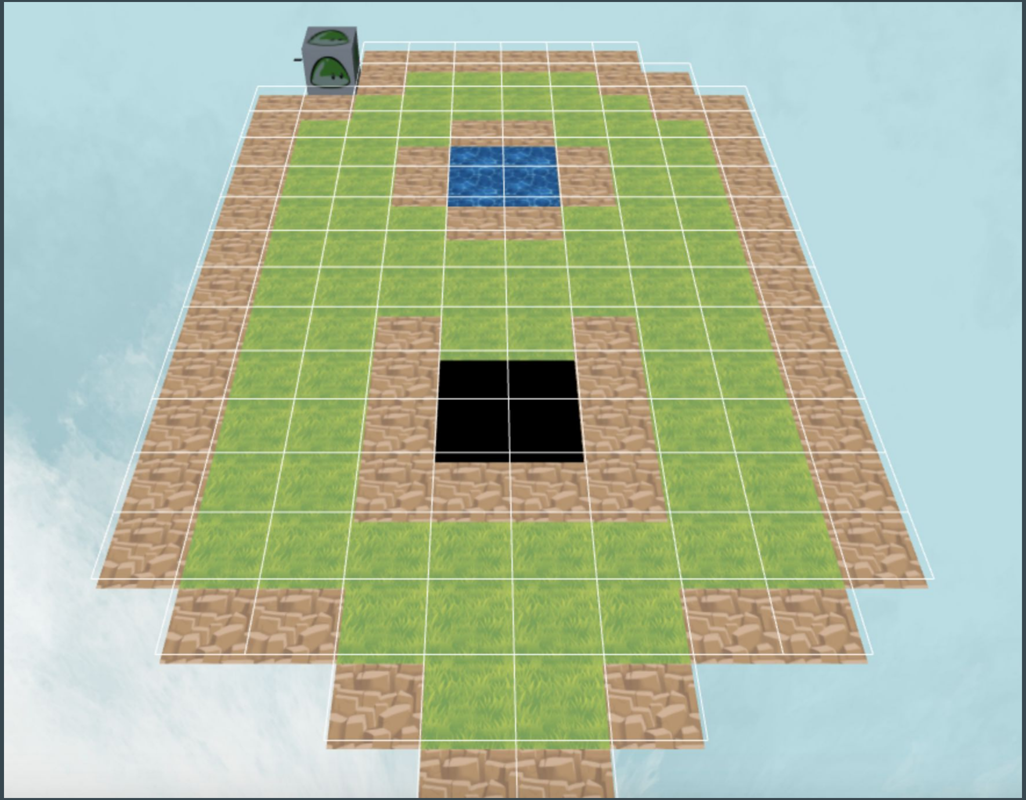
https://threejs.org/docs/index.html#examples/en/controls/OrbitControls

https://threejs.org/examples/?q=orbit#misc_controls_orbit



## Grid Overlay

- We made it so with each tile of terrain generated, a grid is also generated.
- This generates a much cleaner look to our game board since the grid does not overflow into the void.



## Untraversability

- Rocky Terrain has been replaced with walls, and is untraversable.
- Now, however, we are debating whether to get rid of untraversable ‘types’ of terrains and making certain ‘heights’ of terrain untraversable.
- This would mean that, as long as a terrain tile is too high, it would be untraversable regardless of whether it is a rocky, water, or grass type.
- Example: If our base height is 0, slime would not be able to traverse a grassy space at height 2 without a step of height 1.

## New Terrain Type: Gap

- Originally, we specified that Void spaces were empty spaces that would cause the player to fall.
- Going forward, we realized that these functionalities needed to be seperate.
- Void spaces are now only used for shaping a level.
- Gap spaces, which is solid black terrain, will now cause the player to fall.

## Cursor

Movement:

- Original implementation of our cursor allowed for single space movements where pressing Enter moved the player anywhere on the board.
- Future implementation will involve a cursor model as opposed to a flat ring.

Limitation:

- Movement of the cursor is now restricted based on a set range. Default range is 1 which means the player can move 1 space in any direction (including diagonals).
- Our cursor changes color as well. Green means Valid move within the range, Yellow means Invalid move.

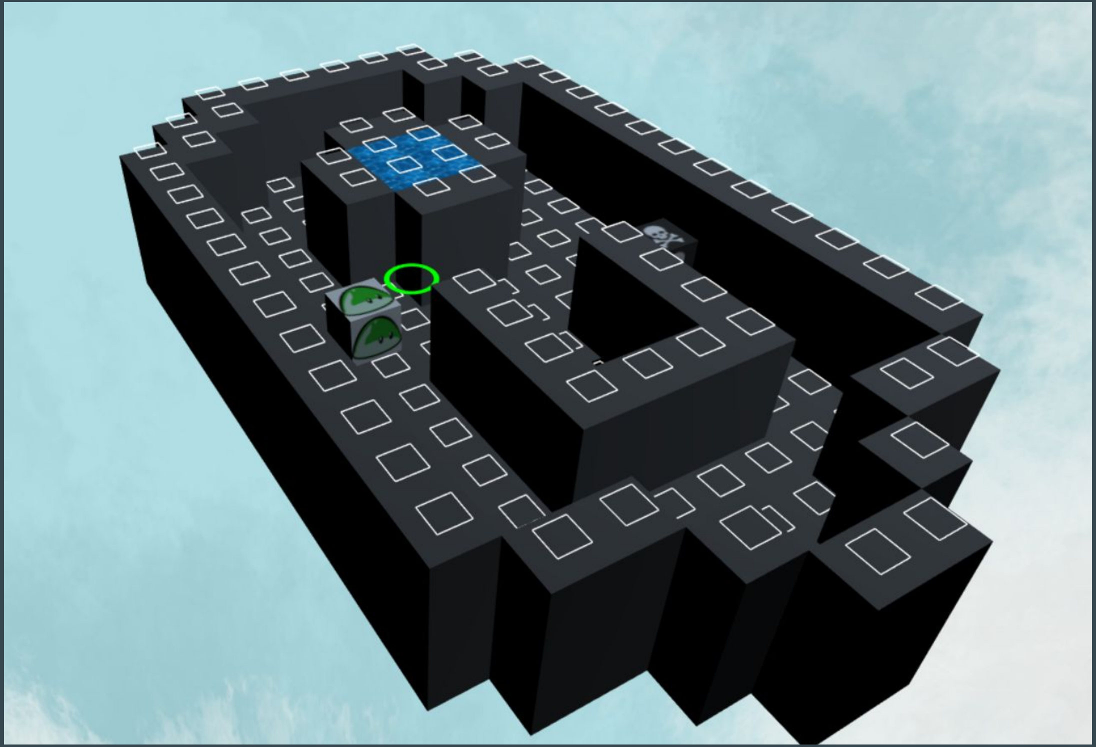
## Enemy Implementation

Challenges

- Spaghetti code
- Multiple enemy implementation

Looking forward, we want to implement AI. Since our game is puzzle-based, we want certain enemies to have a complete lack of randomness, eliminating RNG altogether. We want our puzzles to have at least one guaranteed solution.

## Our Current Grid

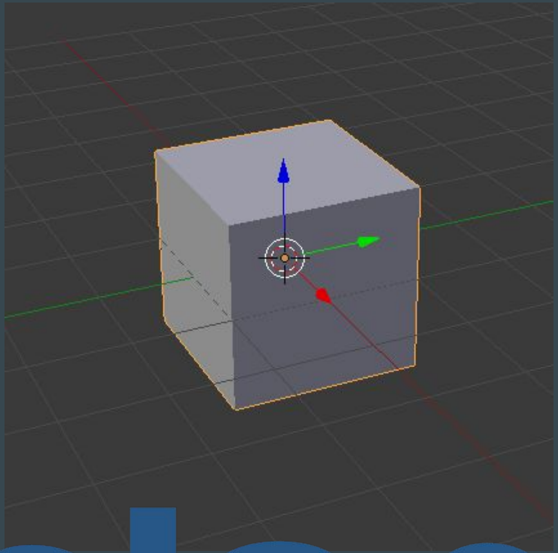


## Height Map

Implemented as an an array filled with values to show how many units high a terrain panel is, relative to the ground level..

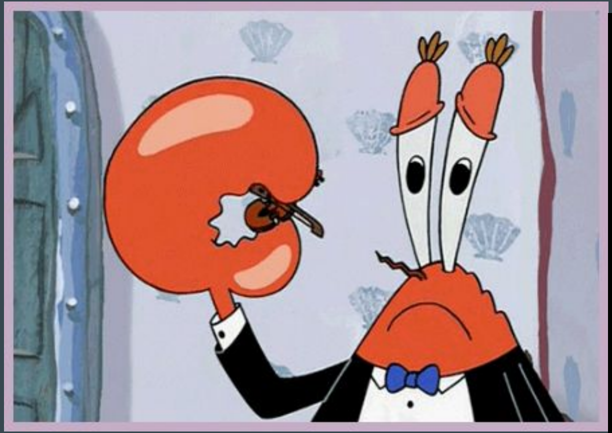
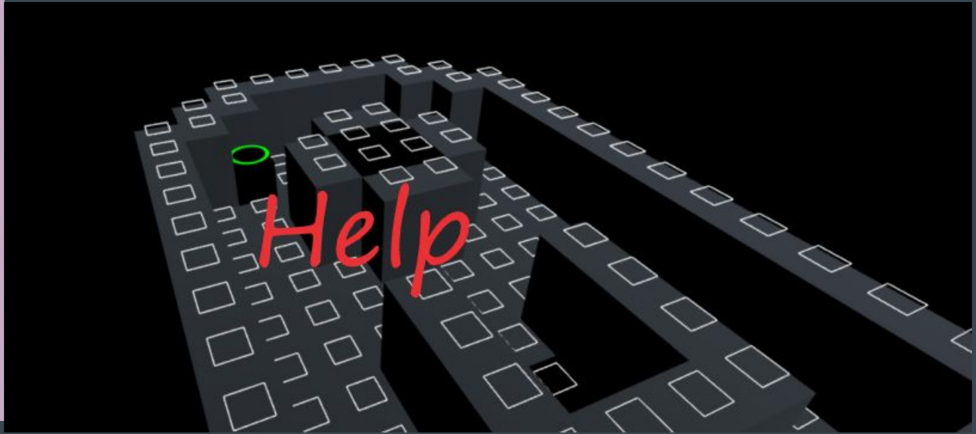
This is a temporary measure for until we implement a comprehensive “tile” object, but it can be accessed by other segments of code to position elements appropriately.

## To Blender!



## Challenges

- Enemy Movement
- Three.js version
- Movement Control

## Future Development

Tool tips

3D models

Enemy movement system

Turn-based

Textures

REFACTORING