# Capsule-Man 2

By: Alex Fratepietro

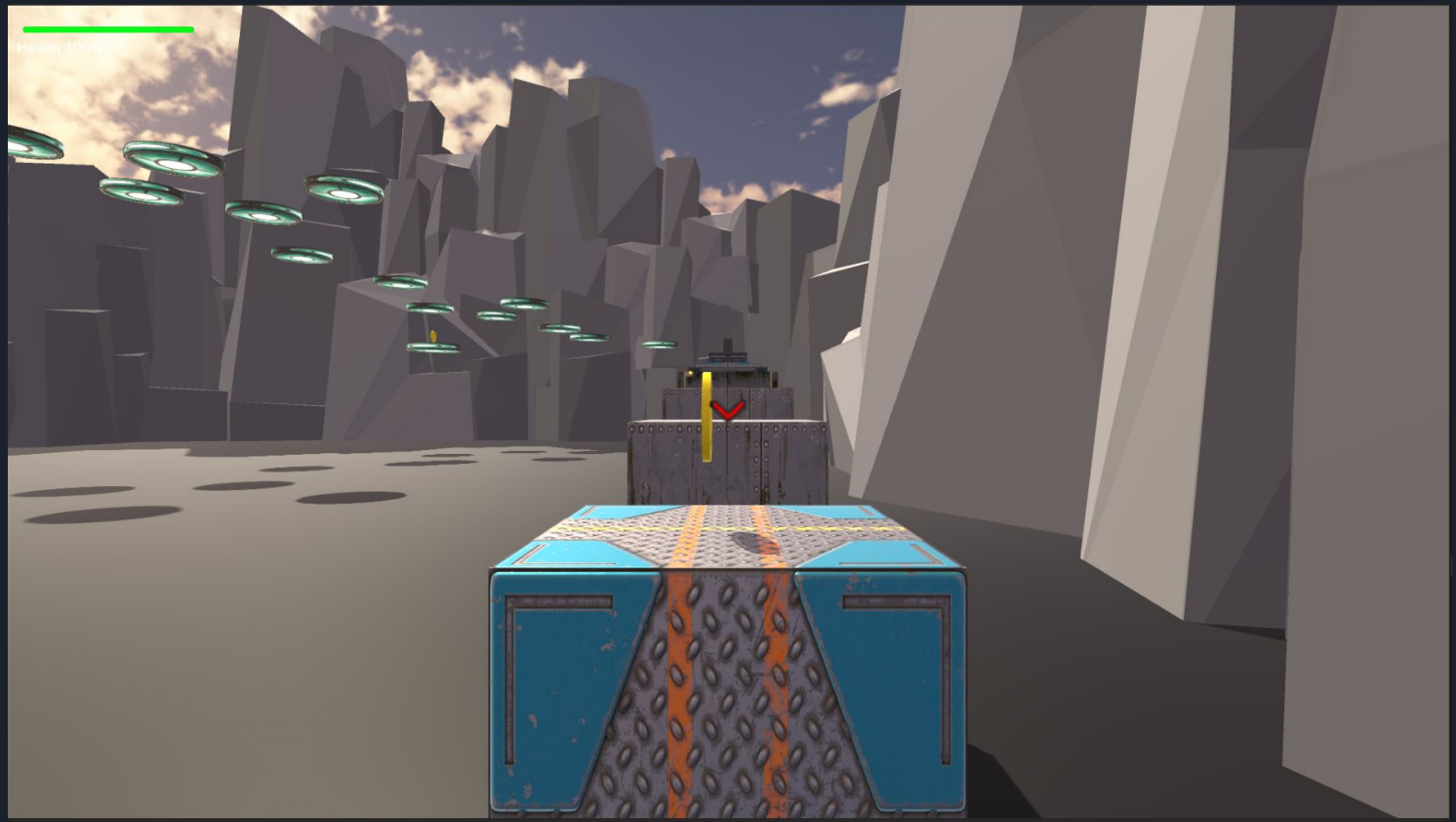
# Introduction

- Why we decided to make a game
- Playthrough
- The creation process
- Reflection

# Learning Unity and C#

- The internet
- Youtube
- Unity API documentation
- Stack overflow
- Expert from expert night

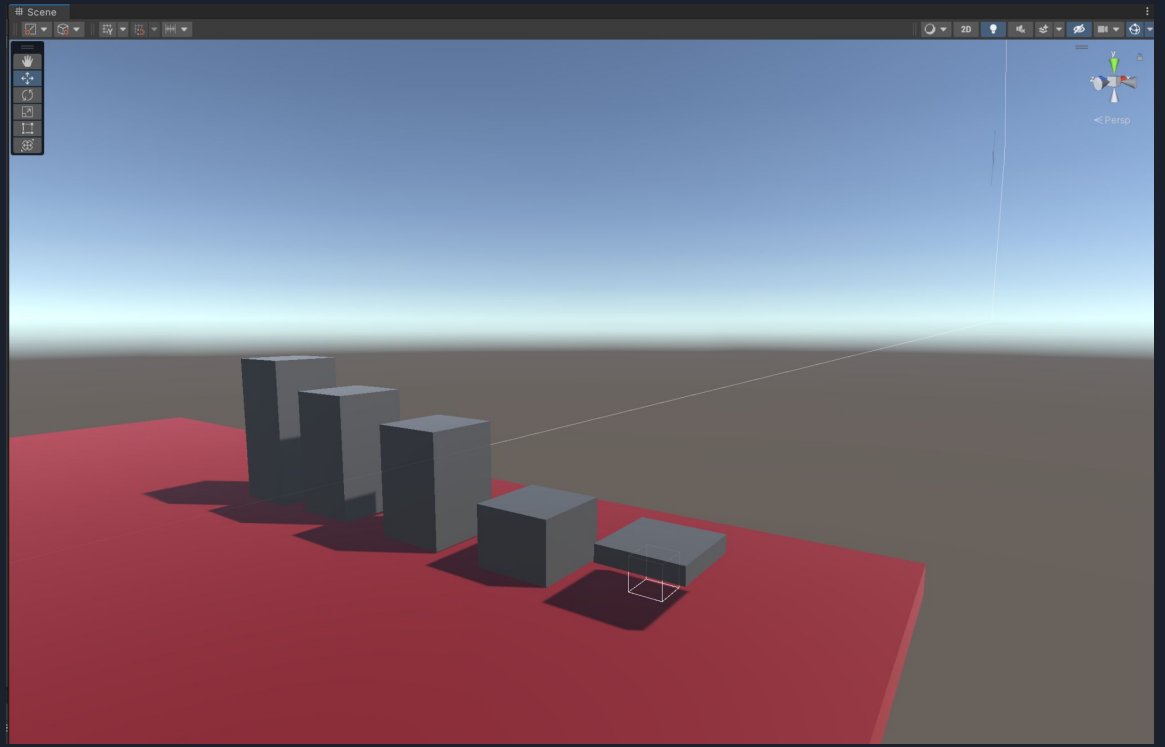
# Level 1 Design Process



# Start Simple

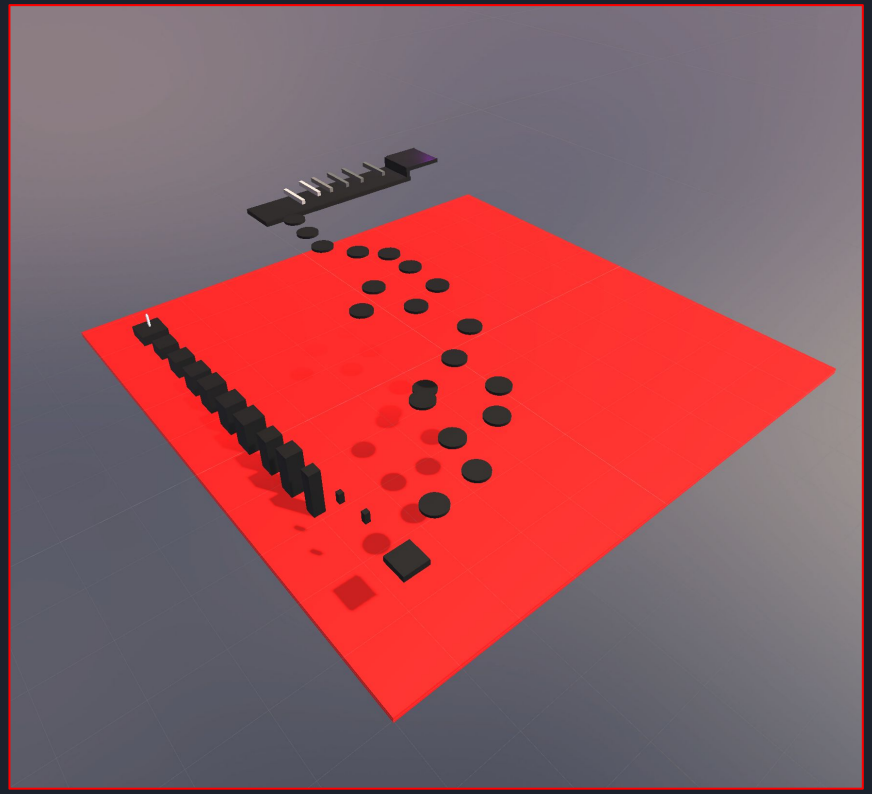
- No textures or models
- Just necessities
- Make sure everything is working

```
private void HandlePlayerMove() {
    // Move direction directly from axes
    float deltaX = Input.GetAxis("Horizontal") * moveSpeed;
    float deltaZ = Input.GetAxis("Vertical") * moveSpeed;
    _moveDirection = new Vector3(deltaX, _moveDirection.y, deltaZ);
    // Accept jump input if grounded
    if (_charCont.isGrounded) {
        if (Input.GetButton("Jump")) {
            _moveDirection.y = jumpSpeed;
        } else {
            _moveDirection.y = 0f;
        }

    }
    ApplyMovement();

}
```



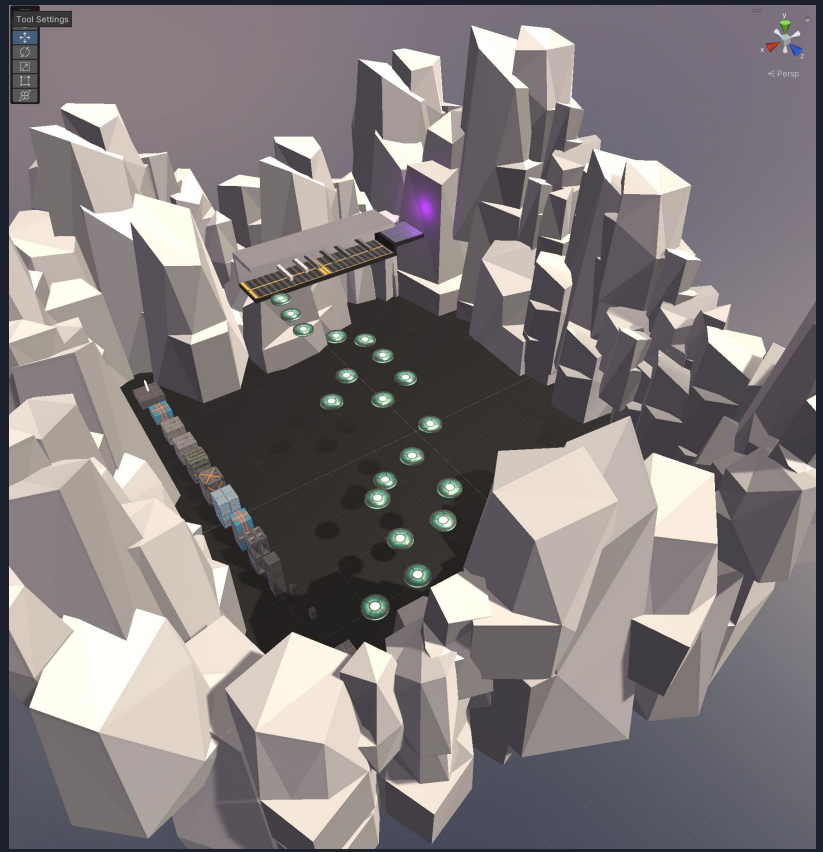
# Finish Map

- Layout of full level map
- Still no textures or models
- Completion of parkour



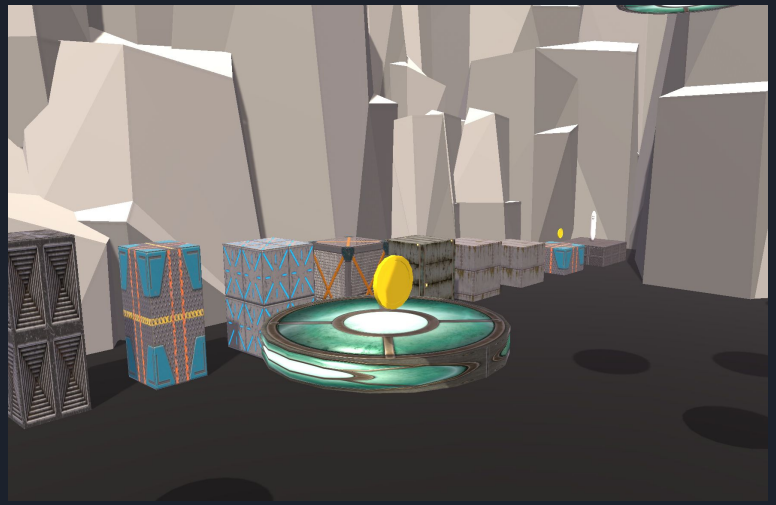
# Add Textures and Map Design

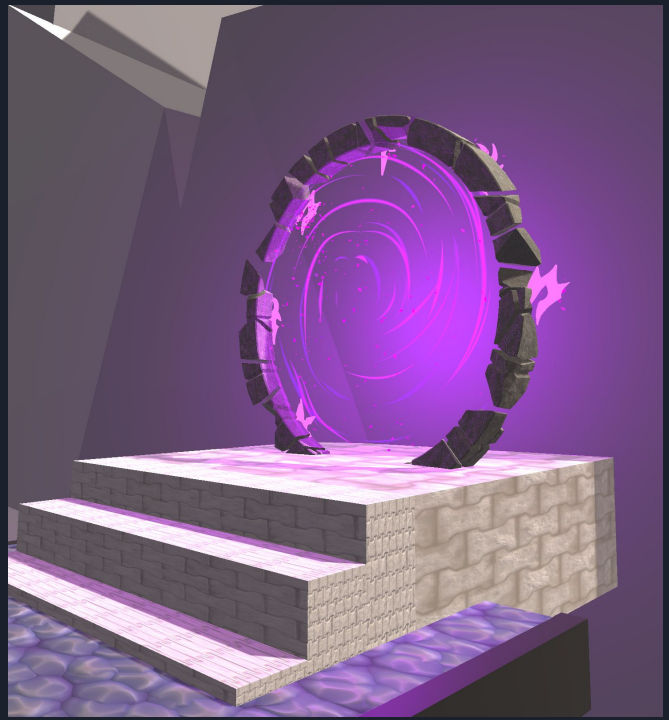
- Add textures and models
- Rocks and jump textures



# Add Details, Extras and UI

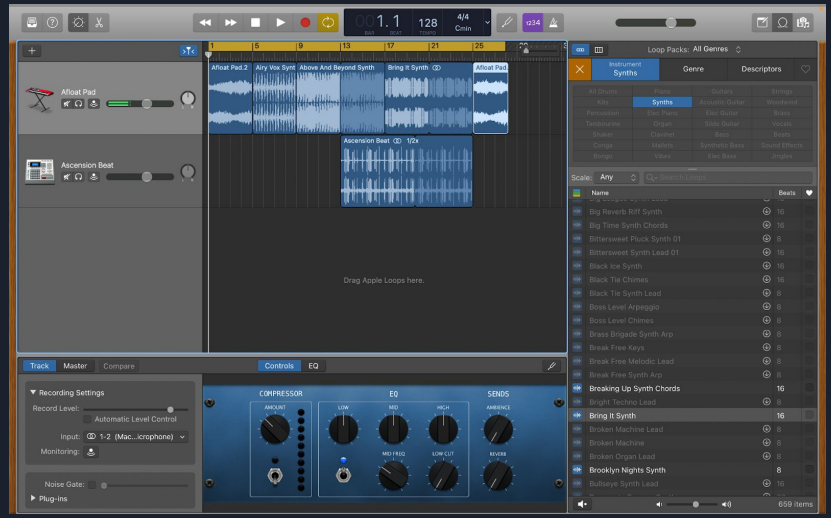
- Link to other levels
- Crosshair User Interface (UI)
- Coins

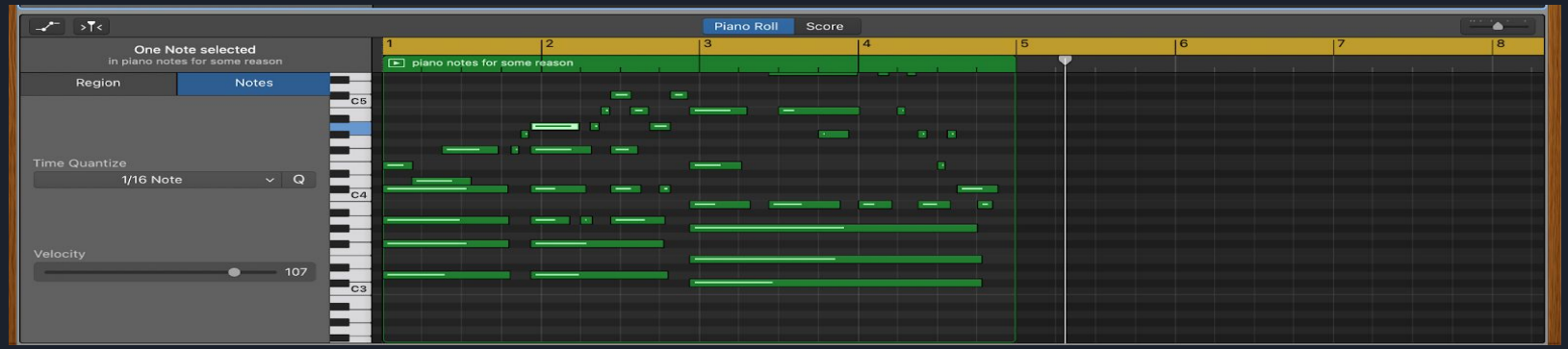




# Music

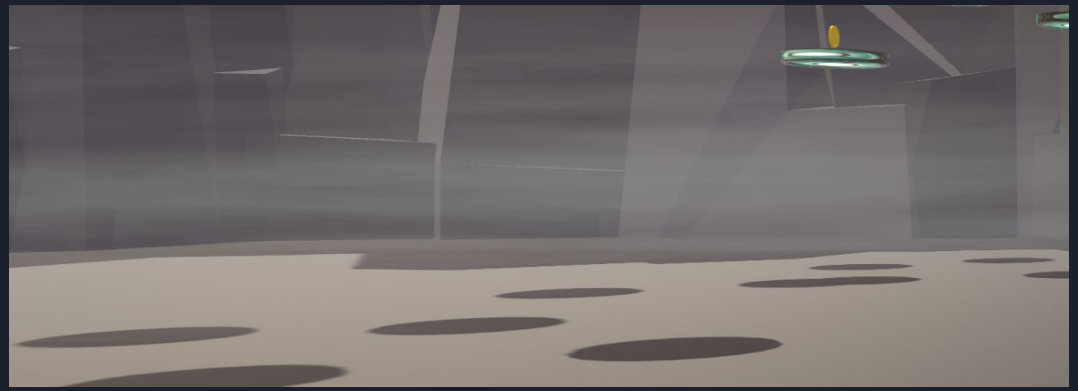
- DTC taught us Garageband
- Different genres for each level
- Menu theme



# Particle Packages

- Add interest to visual effects
  - Fog
  - Snow

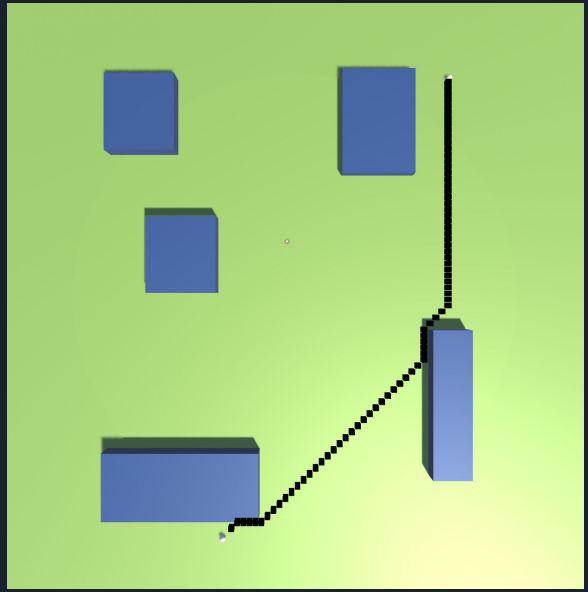


# Enemy Pathfinding Process

-Three numbers:

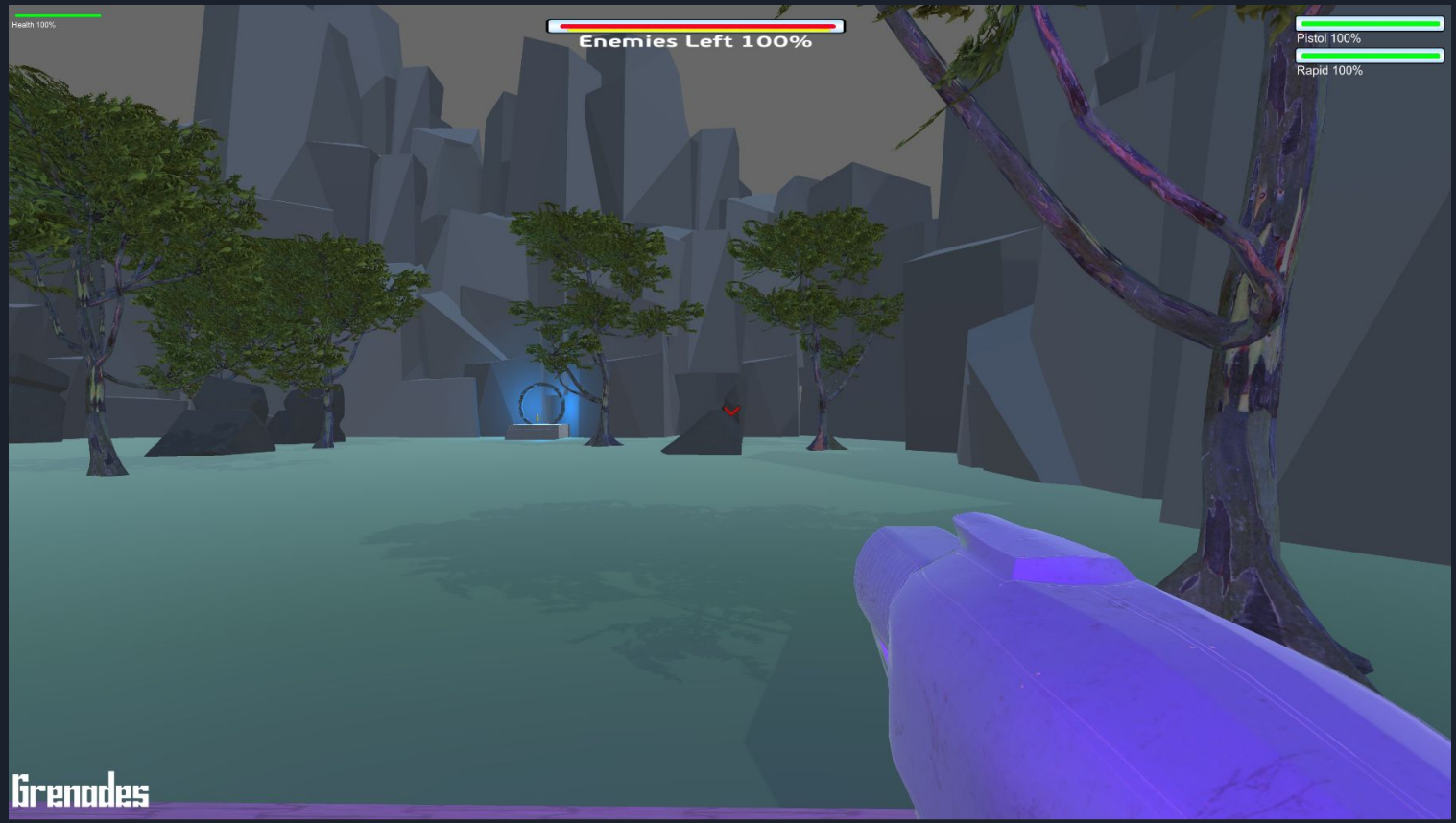
- G cost
- H cost
- F cost

-Nav-Mesh



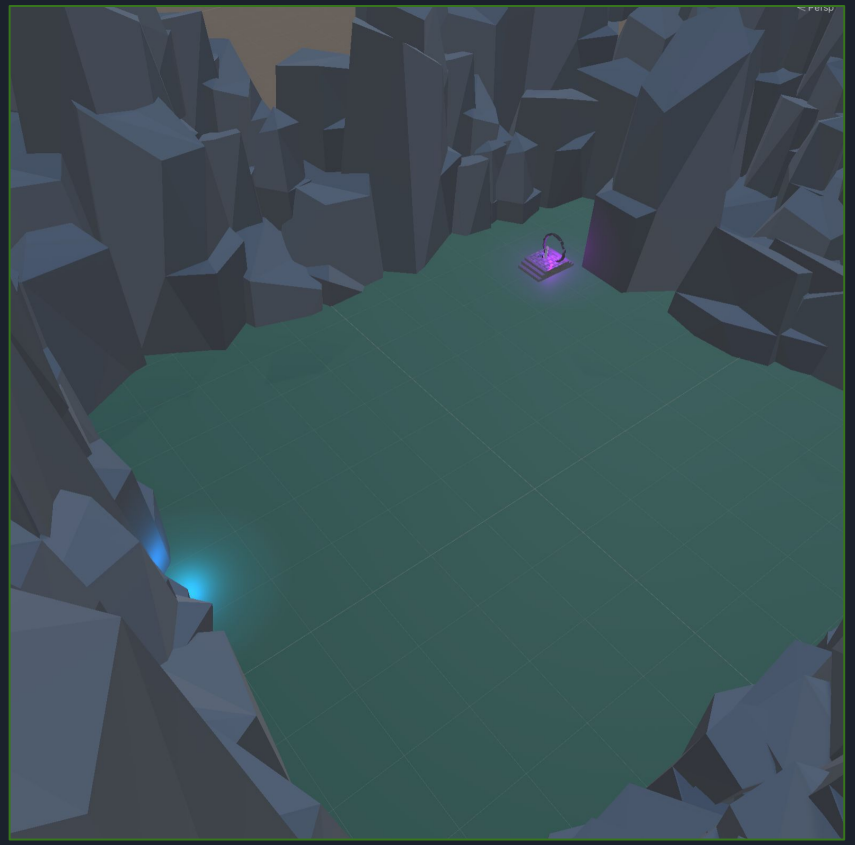
```
            foreach (Node neighbour in grid.GetNeighbours(currentNode)){

                if (!neighbour.walkable || closedSet.Contains(neighbour)){
                    continue;
                }
                int newMovementCostToNeighbour = currentNode.gCost + GetDistance(currentNode, neighbour)
                if (newMovementCostToNeighbour < neighbour.gCost || !openSet.Contains(neighbour)){
                    neighbour.gCost = newMovementCostToNeighbour;
                    neighbour.hCost = GetDistance(neighbour, targetNode);
                    neighbour.Parent = currentNode;

                    if (!openSet.Contains(neighbour))
                        openSet.Add(neighbour);
                }
            }
        }
```

# Level 2 Design Process



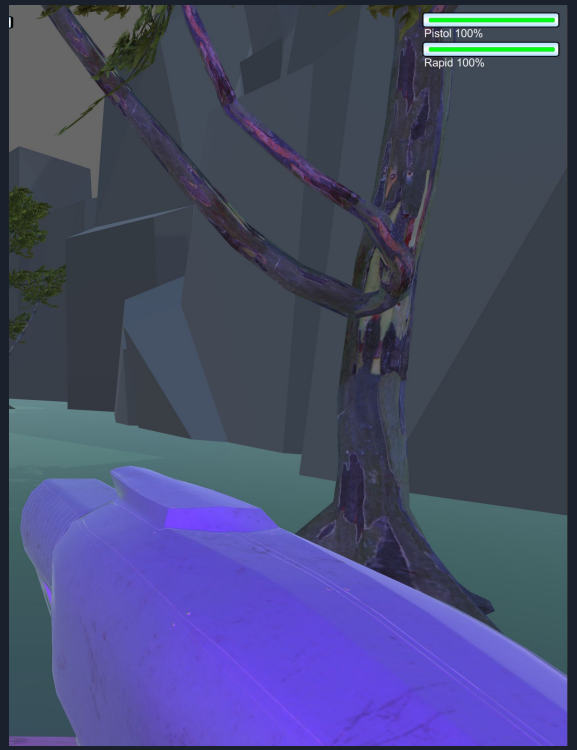
# Use Same Starting Steps

- Same movement
- Similar map designs
- Change in lighting and scenery



# Add Weapon and Firing Capabilities

- Required learning about instantiation
- Display ammo in quicktime
- Check for collision with enemies

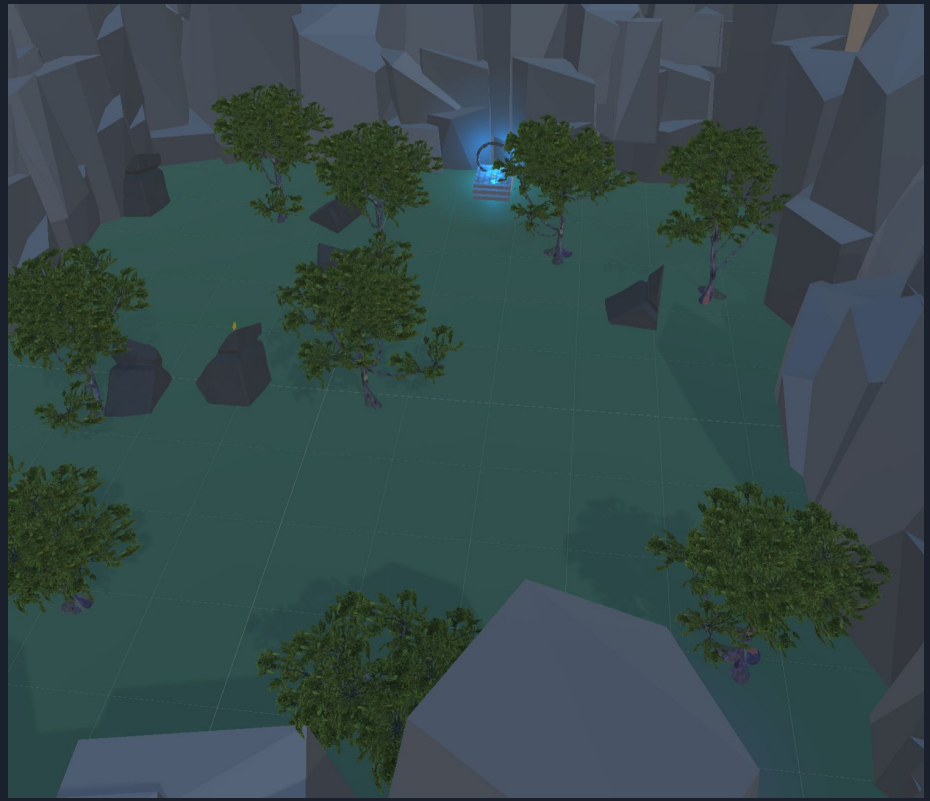


# Enemies

- Designed in Blender
- Tracking stats for each enemy
- Pathfinding
- Attack / damage to player

# Add Additional Details

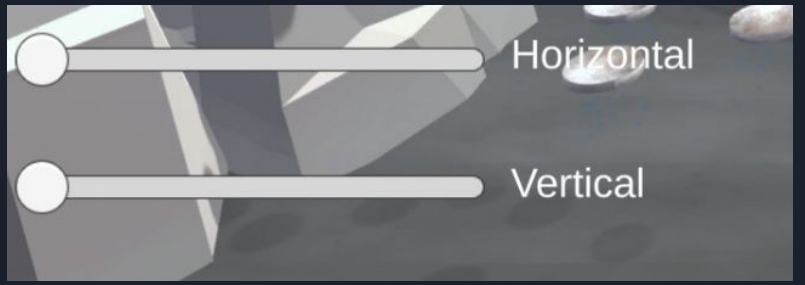
- Grenade option
- Extra obstacles (trees and rocks)
- Sky
- Coins
- Music



# Menu Design Process

# Receive Inputs

- Sensitivity inputs
- Button returns




```
public void ChangeH (float val) {
    PlayerStat.sensH = val;

}
public void ChangeV (float val) {
    PlayerStat.sensV = val;

}
```

# Tips and Stats

- Counters for different stats
- Tips on how to survive

```
private void Update() {
    txt.text = "DEATHS: " + PlayerStat.deaths.ToString();
    stat.text = "KILLS: " + PlayerStat.kills.ToString();
    coins.text = "COINS: " + PlayerStat.coinCollect.ToString();
}
```

# Putting the Game Online

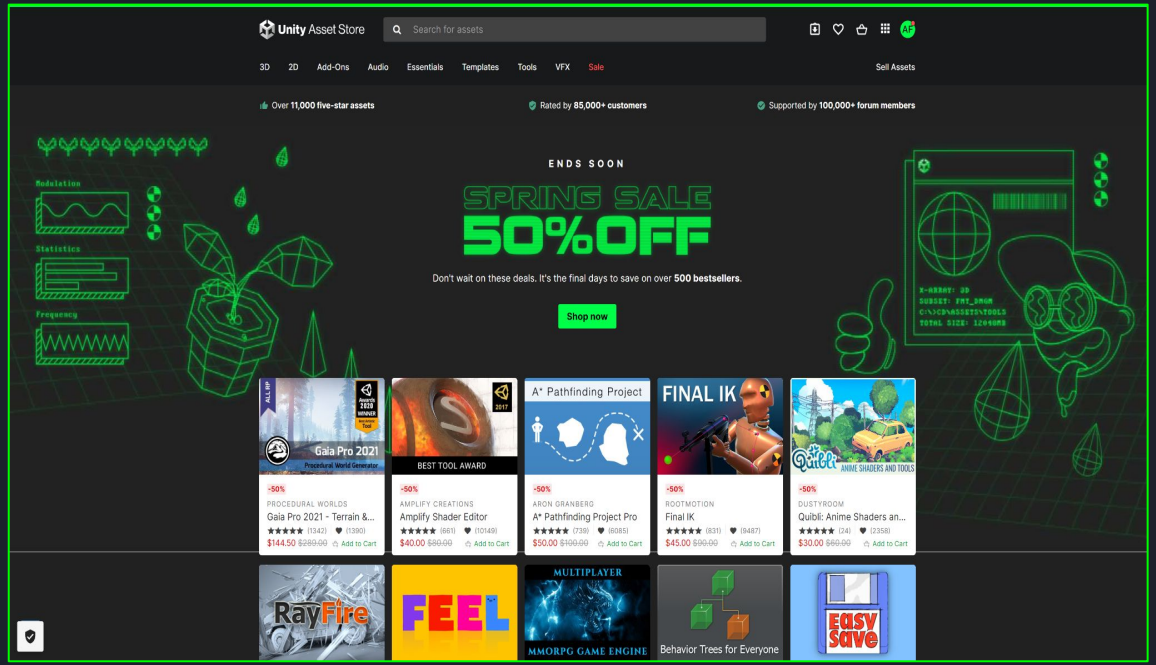
- Using Simmer.io
- Game address:

  https://simmer.io/@_Frate_/capsule

# Software Used

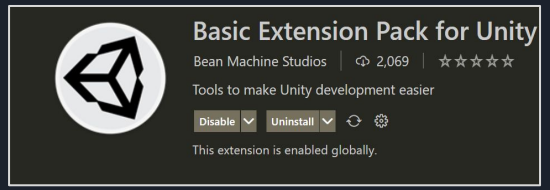
# Unity Asset Store

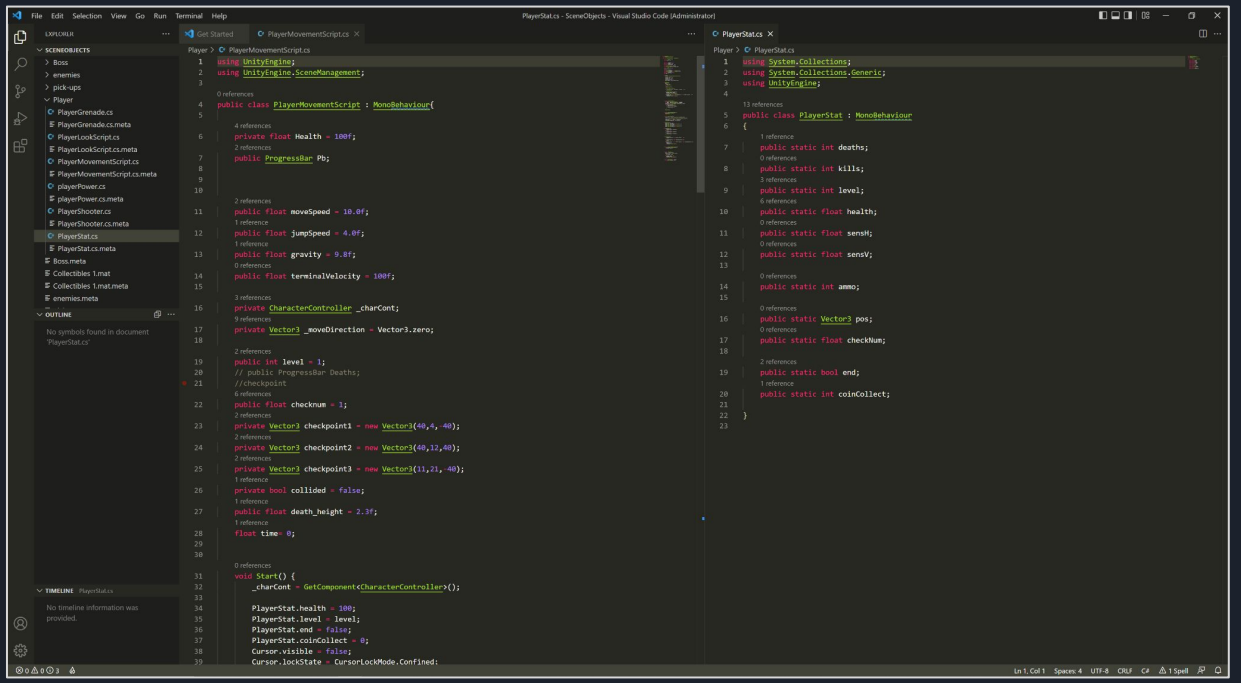
Place to download useful textures and models



# Visual Studio Code

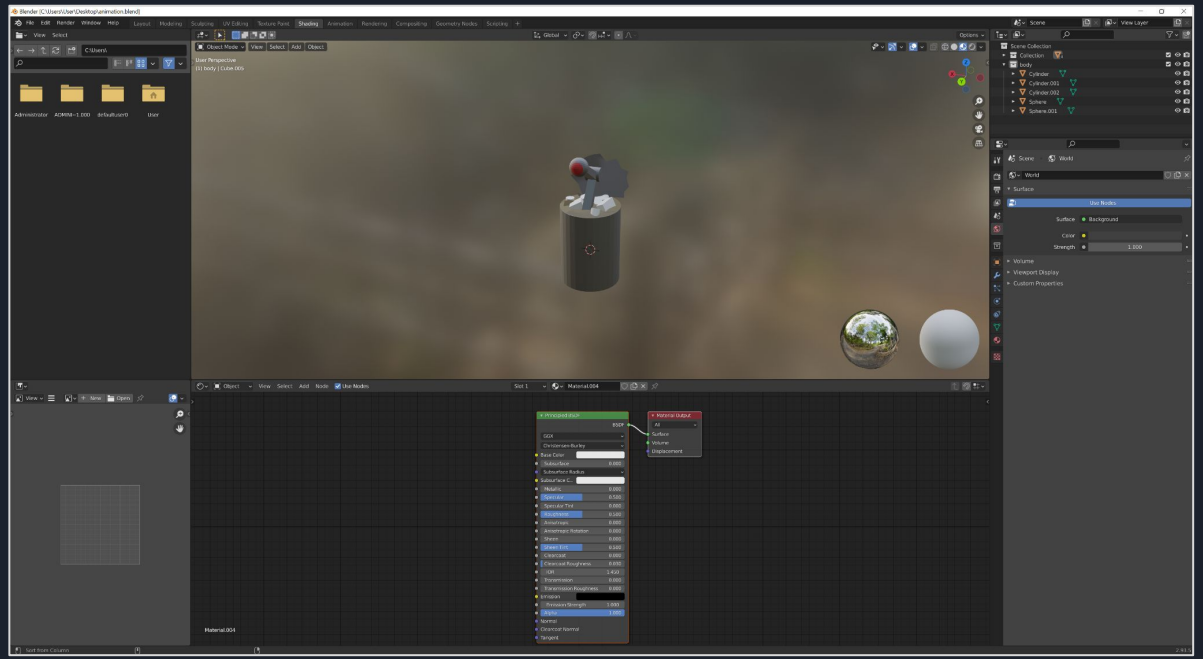
How the game was coded



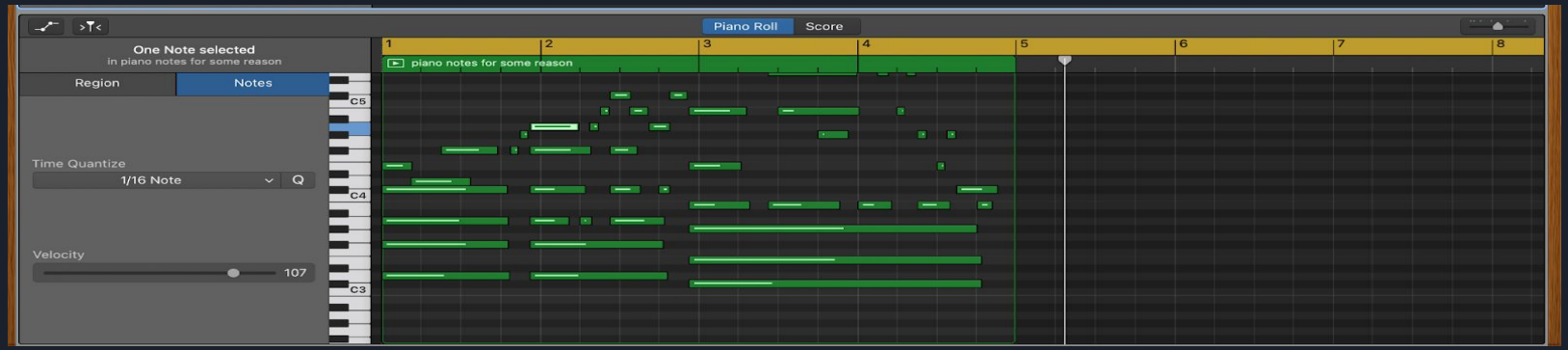
# Blender

How models were created, edited and textured



# Garage Band

- Easy to use
- Knew the basics

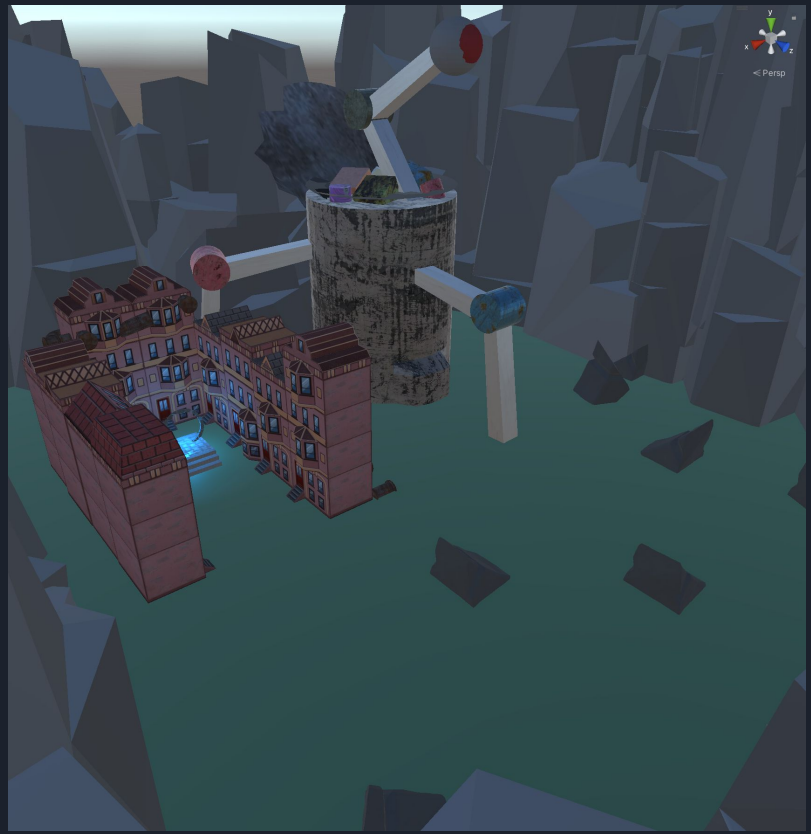


# Working As a Team

- More and different Ideas
- Work Amount
  - Being able to do more
  - More detail
- Designating specialties

# With More Time...

- Final Boss Level
- Rendering settings / graphics
- Database stats such as High Score



# Future

- 2D Game
  - More thematic
  - More time
  - More ambition
  - Publishing plans →

# Thank You