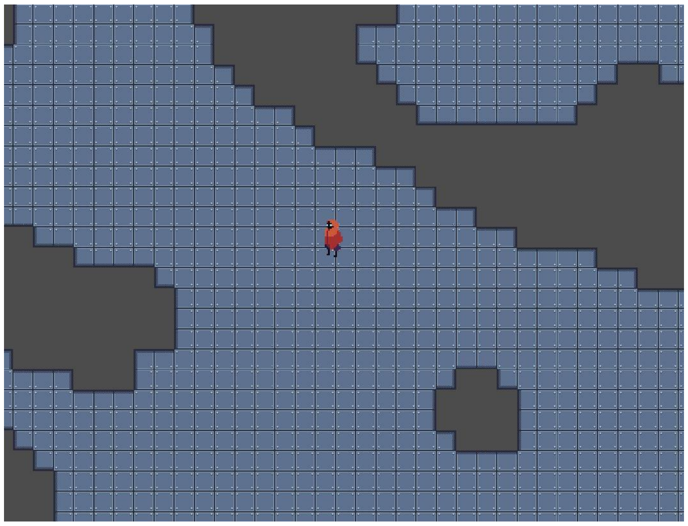
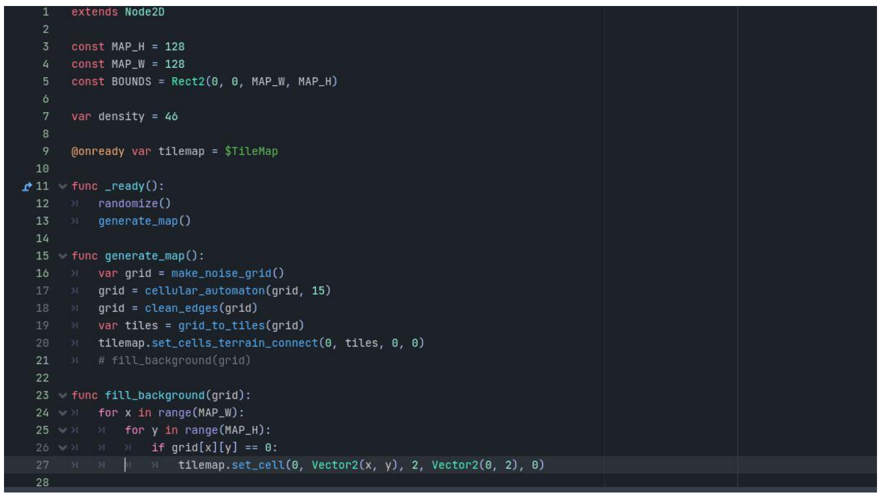
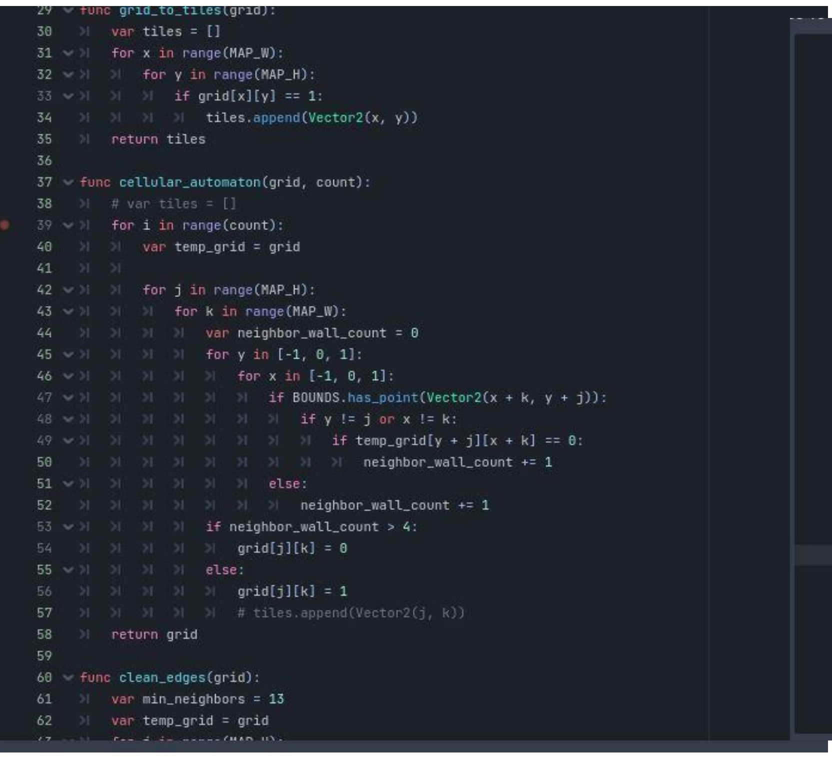
Task1: Procedural Generation
Task2: Camera Following

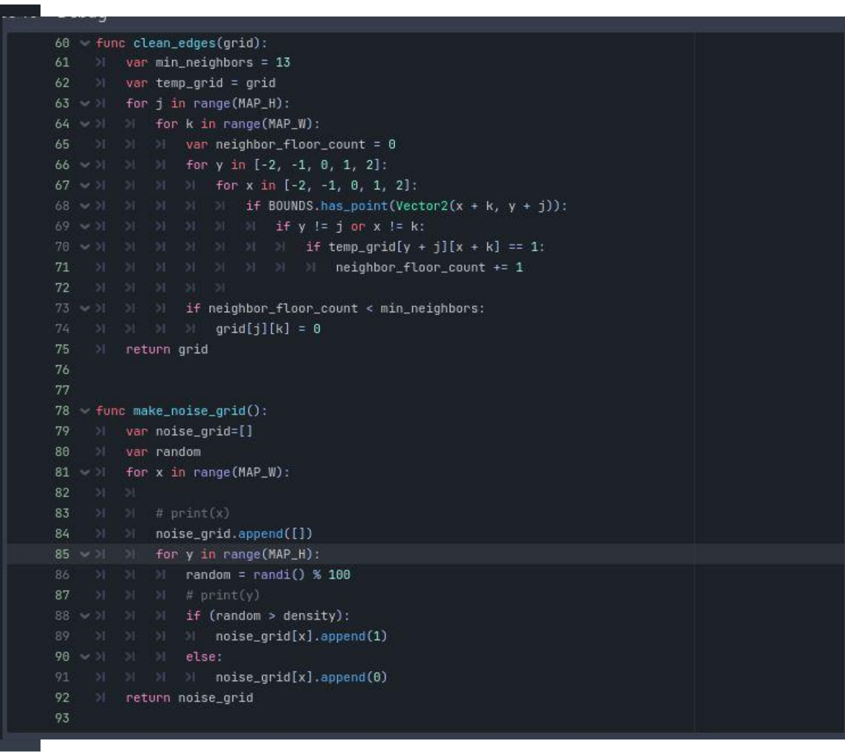




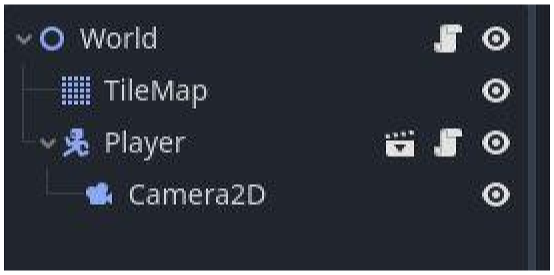
```
extends Node2D

const MAP_H = 128
const MAP_W = 128
const BOUNDS = Rect2(0, 0, MAP_W, MAP_H)

var density = 46

@onready var tilemap = $TileMap

func _ready():
    randomize()
    generate_map()

func generate_map():
    var grid = make_noise_grid()
    grid = cellular_automaton(grid, 15)
    grid = clean_edges(grid)
    var tiles = grid_to_tiles(grid)
    tilemap.set_cells_terrain_connect(0, tiles, 0, 0)
    # fill_background(grid)

func fill_background(grid):
    for x in range(MAP_W):
        for y in range(MAP_H):
            if grid[x][y] == 0:
                tilemap.set_cell(0, Vector2(x, y), 2, Vector2(0, 2), 0)

```



```
func grid_to_tiles(grid):
    var tiles = []
    for x in range(MAP_W):
        for y in range(MAP_H):
            if grid[x][y] == 1:
                tiles.append(Vector2(x, y))
    return tiles

func cellular_automaton(grid, count):
    # var tiles = []
    for i in range(count):
        var temp_grid = grid
        
        for j in range(MAP_H):
            for k in range(MAP_W):
                var neighbor_wall_count = 0
                for y in [-1, 0, 1]:
                    for x in [-1, 0, 1]:
                        if BOUNDS.has_point(Vector2(x + k, y + j)):
                            if y != j or x != k:
                                if temp_grid[y + j][x + k] == 0:
                                    neighbor_wall_count += 1
                        else:
                            neighbor_wall_count += 1
                if neighbor_wall_count > 4:
                    grid[j][k] = 0
                else:
                    grid[j][k] = 1
                # tiles.append(Vector2(j, k))
    return grid

func clean_edges(grid):
    var min_neighbors = 13
    var temp_grid = grid
```



```
func clean_edges(grid):
    var min_neighbors = 13
    var temp_grid = grid
    for j in range(MAP_H):
        for k in range(MAP_W):
            var neighbor_floor_count = 0
            for y in [-2, -1, 0, 1, 2]:
                for x in [-2, -1, 0, 1, 2]:
                    if BOUNDS.has_point(Vector2(x + k, y + j)):
                        if y != j or x != k:
                            if temp_grid[y + j][x + k] == 1:
                                neighbor_floor_count += 1
                        
            if neighbor_floor_count < min_neighbors:
                grid[j][k] = 0
    return grid


func make_noise_grid():
    var noise_grid=[]
    var random
    for x in range(MAP_W):
        
        # print(x)
        noise_grid.append([])
        for y in range(MAP_H):
            random = randi() % 100
            # print(y)
            if (random > density):
                noise_grid[x].append(1)
            else:
                noise_grid[x].append(0)
    return noise_grid

```


World
TileMap
Player
Camera2D