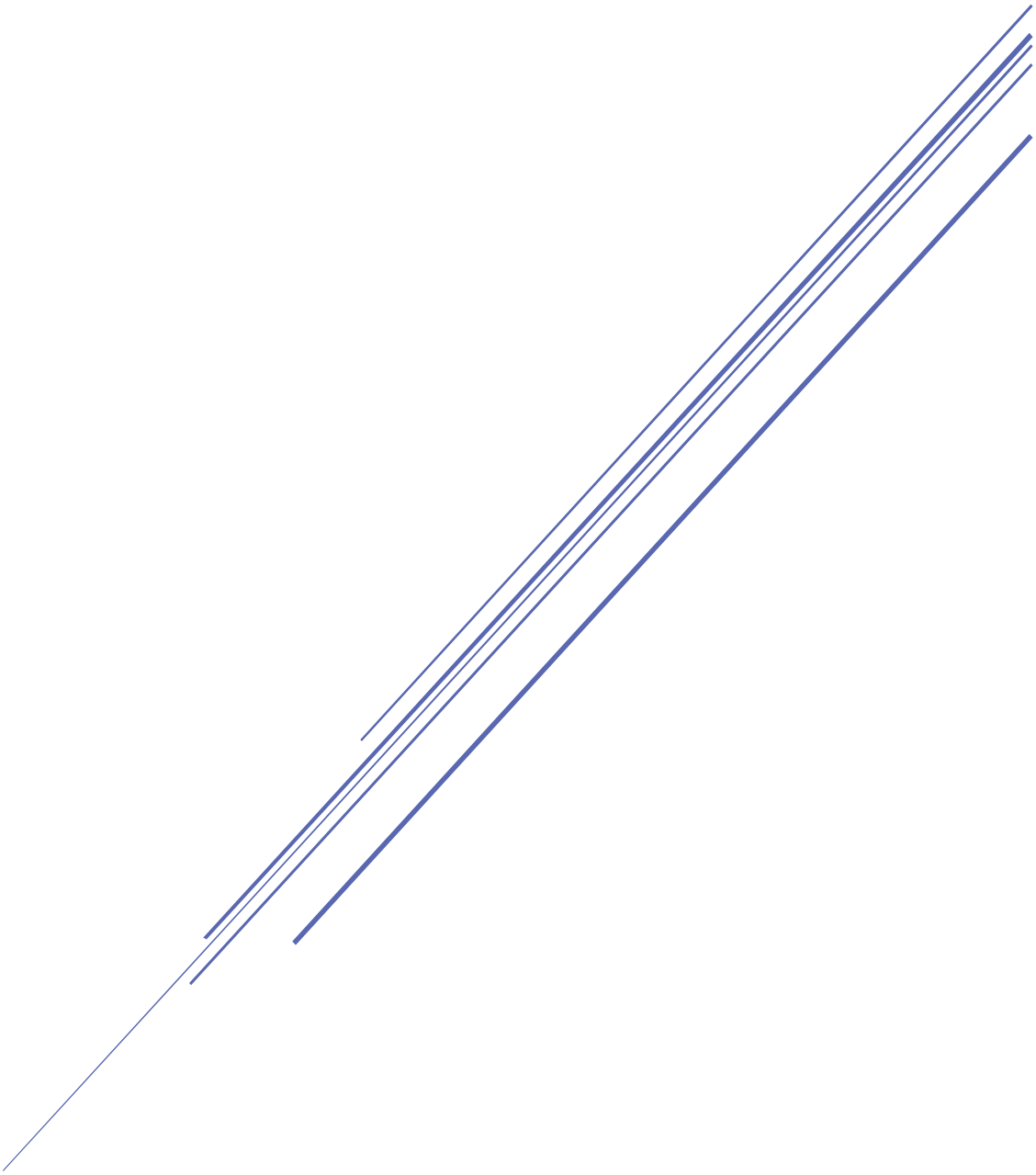
PROCEDURAL DUNGEON GENERATOR



Created by: Miles Ogilvie

Project Type: Unity 2D – Procedural Generation Prototype

Contents

1. Project Overview

Project Title: Procedural Dungeon Generator
Tools & Environment:

- Unity Version: 6000.2.7f2
- Language: C#
- IDE: Visual Studio / Rider
- Version Control: Git & GitHub Desktop
- Platform Target: Windows / WebGL (if applicable)

Project Description:
This project demonstrates procedural content generation using Unity. It automatically creates a randomized dungeon layout consisting of rooms, corridors, walls, and floor tiles. A controllable player character is spawned within the generated map, and the camera dynamically follows the player as they move through the environment.

Main Features:

- Procedurally generated dungeon using room and corridor placement.
- Configurable map parameters (size, room count, width, etc.).
- Adjustable corridor width for improved navigation.
- Automatic wall placement around floor tiles.
- Player spawning system that ensures valid floor placement.
- Player movement system using Unity's 2D physics.
- Camera follow functionality with smooth interpolation.
- Main menu with start and quit options.

2. System Architecture

Code Scripts:

SCRIPT NAME	PURPOSE
DUNGEONGENERATOR.CS	Generates rooms, corridors, floors, and walls procedurally. Spawns player and sets up camera target.
PLAYERMOVEMENT.CS	Handles player input and Rigidbody-based movement (WASD controls).
CAMERAFOLLOW.CS	Smoothly follows the player using interpolation in FixedUpdate().

MAINMENU.CS	Controls Start and Quit functionality for the Start Screen Scene.

Supporting Assets:

- Wall & Floor Tile Sprites (PNG, 32x32 px)
- Player Sprite (32x32 px)
- Start Screen Scene with buttons and background video/GIF
- Pixel Perfect Camera setup for crisp rendering

3. Unity Scene Hierarchy

Procedural Dungeon Scene:

Main Camera

├— CameraFollow (Script)

DungeonManager

├— DungeonGenerator (Script)

TilesParent (Empty)

Player (Prefab)

Canvas (Optional for UI overlays)

EventSystem

Start Screen Scene:

Canvas

├— BackgroundVideo (Raw Image + VideoPlayer)

├— StartButton

├— QuitButton

MenuManager (MainMenu Script)

EventSystem

4. Dungeon Generator Implementation

Script: DungeonGenerator.cs

Purpose:
Handles all aspects of dungeon creation — initializing grid arrays, generating non-overlapping rooms, connecting rooms via corridors, and populating floor/wall tiles.

Key Variables:

VARIABLE	DESCRIPTION
MAPWIDTH, MAPHEIGHT	Defines overall grid size
MAXROOMS, MINROOMSIZE, MAXROOMSIZE	Controls room count and size
CORRIDORWIDTH	Controls corridor thickness
FLOORPREFAB, WALLPREFAB	Prefabs used for rendering tiles
PLAYERPREFAB	Player GameObject spawned after generation
TILESPARENT	Parent transform for spawned tiles
MAPGRID	Internal 2D array (0=empty, 1=floor, 2=wall)

Main Methods:

- **GenerateDungeon()**
- **CarveHorizontalTunnel()**
- **CarveVerticalTunnel()**
- **AddWallsAroundFloors()**
- **DrawMap()**
- **SpawnPlayer()**

5. Player Implementation

Script: PlayerMovement.cs

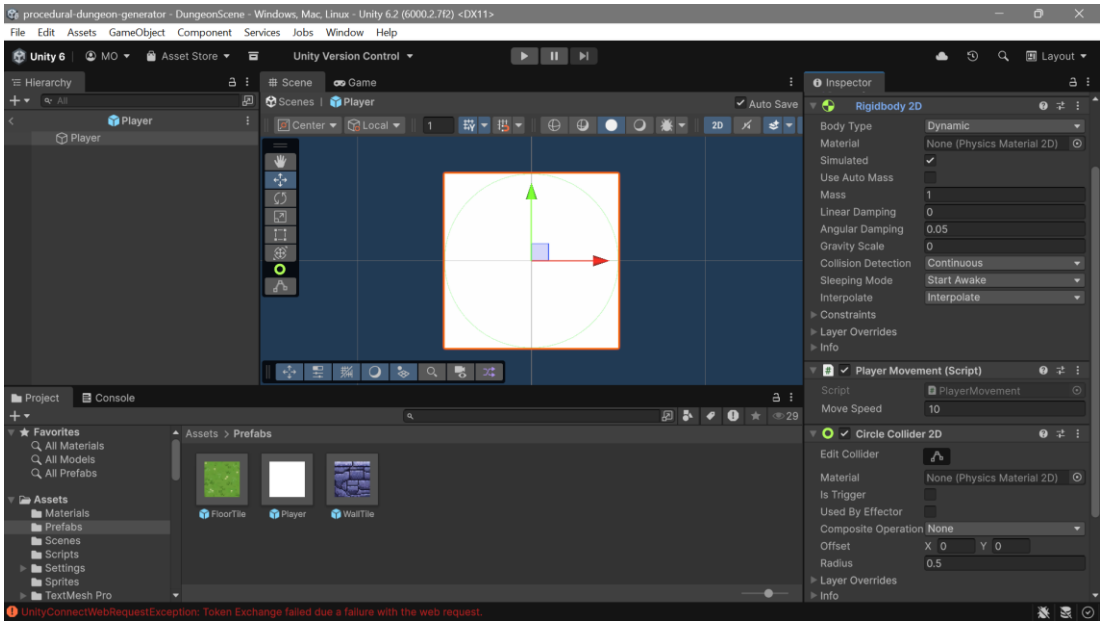
Purpose:
Handles keyboard input and movement through Unity's 2D physics engine.

Key Features:

- Uses Input.GetKey() for WASD input.
- Movement logic in Update().
- Rigidbody physics in FixedUpdate().
- Collision detection with walls (BoxCollider2D / CircleCollider2D).

Player Rigidbody2D Settings:

SETTING	VALUE
BODY TYPE	Dynamic
GRAVITY SCALE	0
COLLISION DETECTION	Continuous
INTERPOLATE	Interpolate
FREEZE ROTATION Z	Checked ✓



6. Camera Implementation

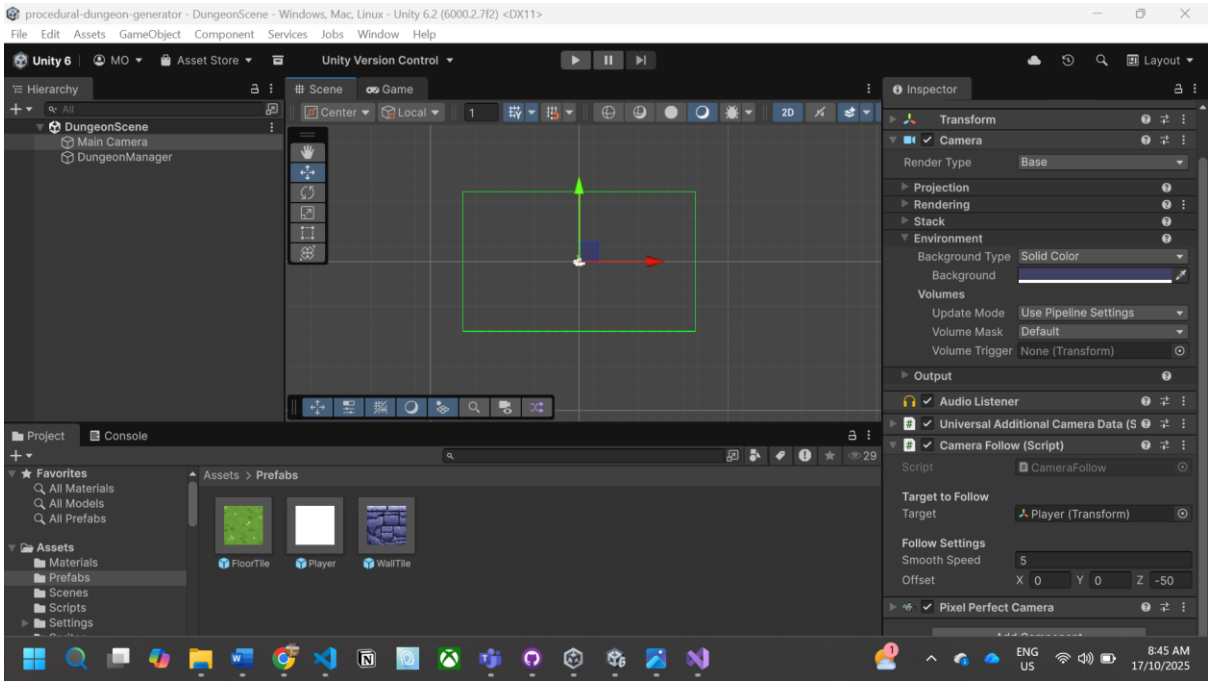
Script: CameraFollow.cs

Purpose:
Smoothly follows the player while maintaining proper synchronization with physics (uses FixedUpdate()).

Camera Settings:

SETTING	VALUE

PROJECTION	Orthographic
SIZE	~15
POSITION Z	-10
PIXEL PERFECT CAMERA	Enabled ✓



7. Start Screen Implementation

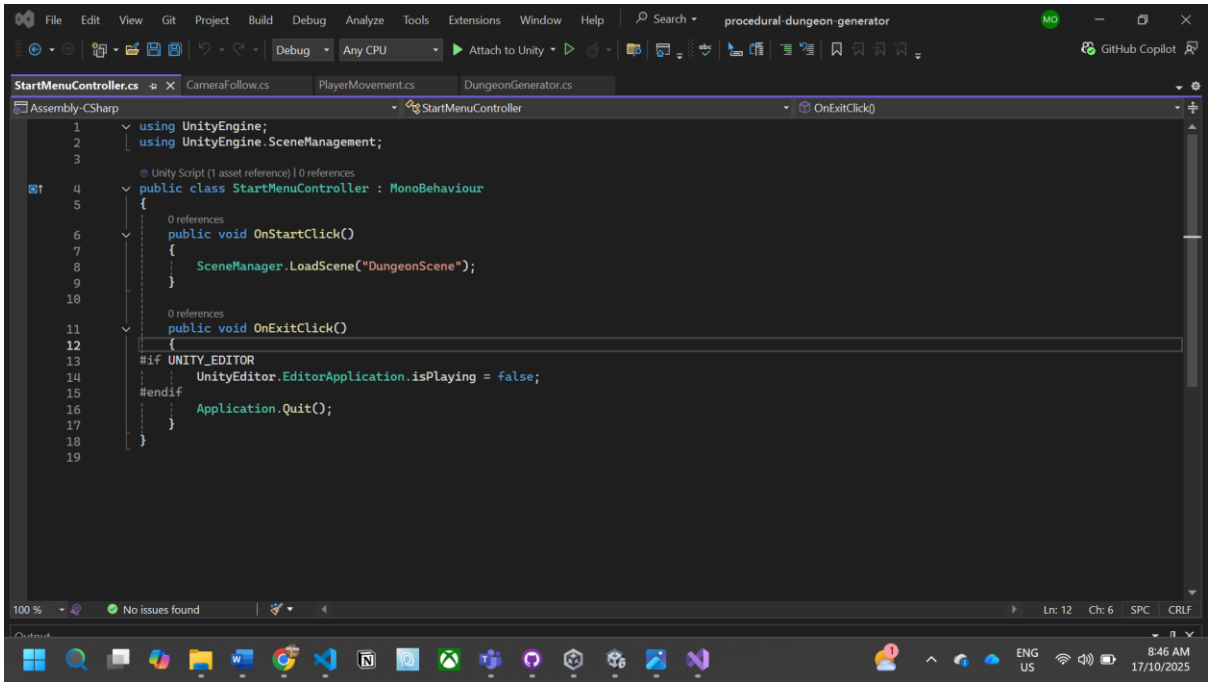
Scene: StartScreenScene

Scripts:

- MainMenu.cs

Components:

OBJECT	COMPONENT	DESCRIPTION
BACKGROUNDVIDEO	Raw Image + VideoPlayer	Displays looping MP4 as animated background
STARTBUTTON	Button	Loads dungeon scene
QUITBUTTON	Button	Exits application



8. Title Settings and Visual Quality

Fixes Applied:

- **Filter Mode:** Point (no filter)
- **Compression:** None
- **PPU:** 32
- **Camera:** Pixel Perfect Camera added
- **Lighting:** 2D Unlit rendering for clean visuals

9. Reflection and Future Improvement

Learnings:

- Gained understanding of procedural generation logic (random placement & overlap detection).
- Learned how to sync player physics with camera movement.
- Improved knowledge of Unity 2D systems (colliders, pixel-perfect rendering, physics).

Future Enhancements:

- Add torch or light sources for ambiance.
- Implement collectibles or enemies.
- Add dungeon exit and restart system.

- Introduce UI minimap of generated layout.
- Randomize tile textures for visual variety.

REMINDER:

This task is a WORK IN PROGRESS with additional content being added in future versions. Some aspects of the code have been generated from AI and tweaked by Me.