Faculty of Engineering in Foreign Languages

Game and Interactive Simulation Systems

Project - PlatPool

Author:
José Gomes

February 24, 2022

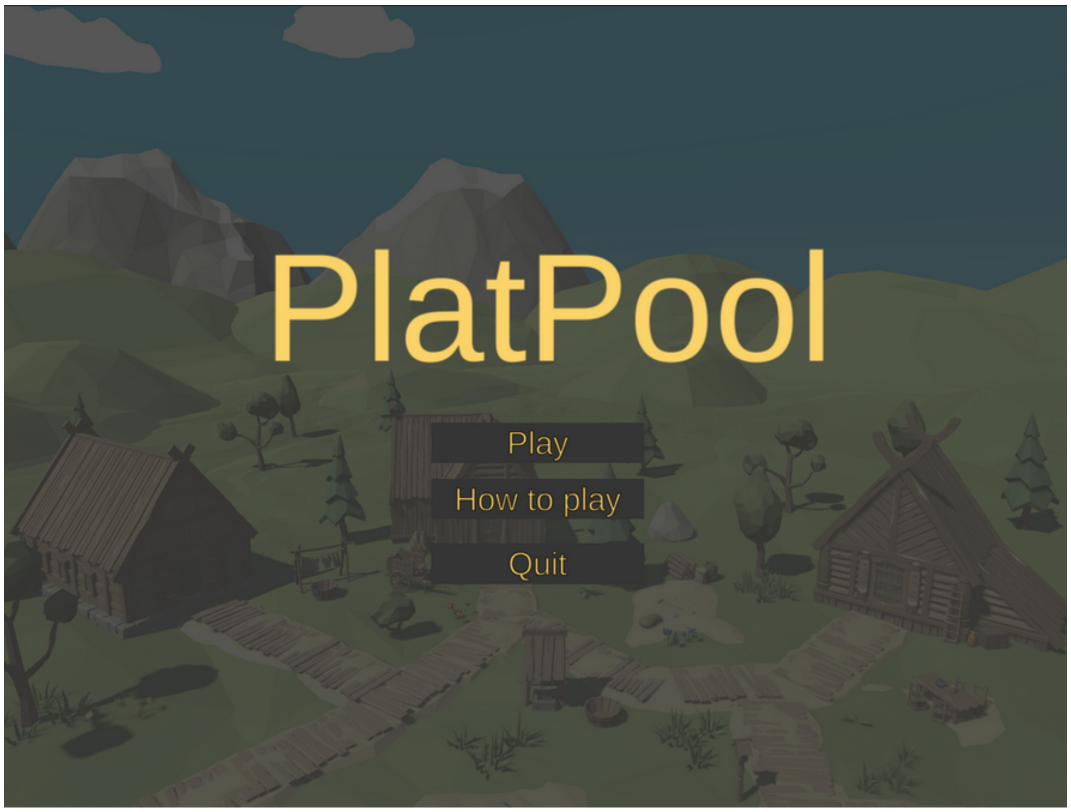
Content

1 Introduction

This project consists of a 3D Game made with Unity. The project aimed to develop a relatively simple game that explored various aspects of game development and the workflow associated with game engines.

1.1 PlatPool

The developed game is called "PlatPool" and is a puzzle-precision game where the player needs to be fast and precise with the movements as well as think about how to act. The core gameplay is based on the game "Press any button"(available at its Steam Page: https://store.steampowered.com/app/1448030/Press_Any_Button/).



Picture1: Main Menu

2 How to play

The player has three and only three platforms that it can use to prevent falling obstacles from touching the floor until they explode. The game has three main aspects that provide rich gameplay.

2.1 Limited Platforms

Because the player has limited platforms, every fourth platform that the player tries to place in the game area removes the oldest platform that has been placed. This way, only three obstacles can be resting on a platform at any time.

2.2 Locking Platforms

The player has the option to lock a platform in place by double-clicking it (turning it red - simple visual effect). This locking action has the side effect of removing said platform from the pool of platforms available, which means that instead of re-positioning the oldest platform, the second oldest platform is re-positioned - if only one platform is locked. The player can lock all platforms.

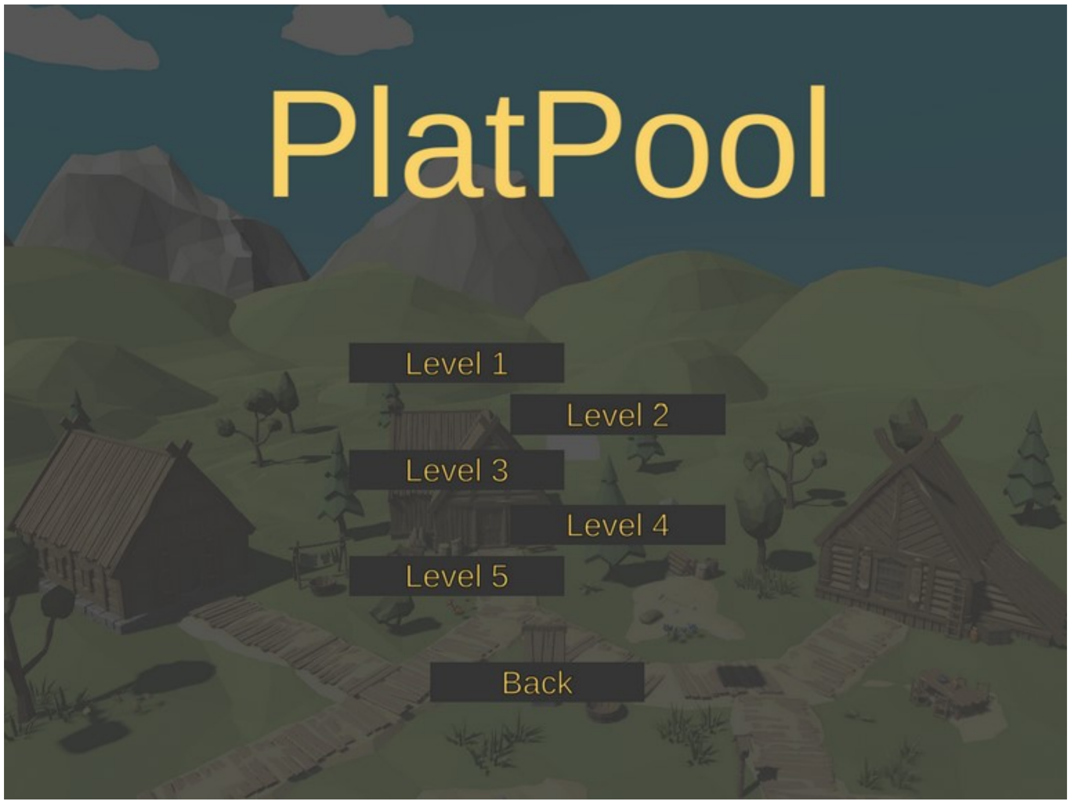
2.3 Different obstacles

The game has three types of obstacles that have different properties, explained as follows:

- Cube - The Cube has a slow falling speed and a short time before exploding
- Long - The Long has the same falling speed as the Cube but takes twice as long to explode.
- Cylinder - The Cylinder instantly explodes as it touches a platform, but it has a very high falling speed

2.4 Mechanics Integration

When placing the platforms, the player needs to think about the order in which to place them (to remove them in an appropriate order later), which platforms to lock and when to unlock them. This, combined with the various blocks, makes for very interesting gameplay that can be pretty challenging. The game also has the possibility to be expanded with different levels requiring different strategies or even different types of obstacles.



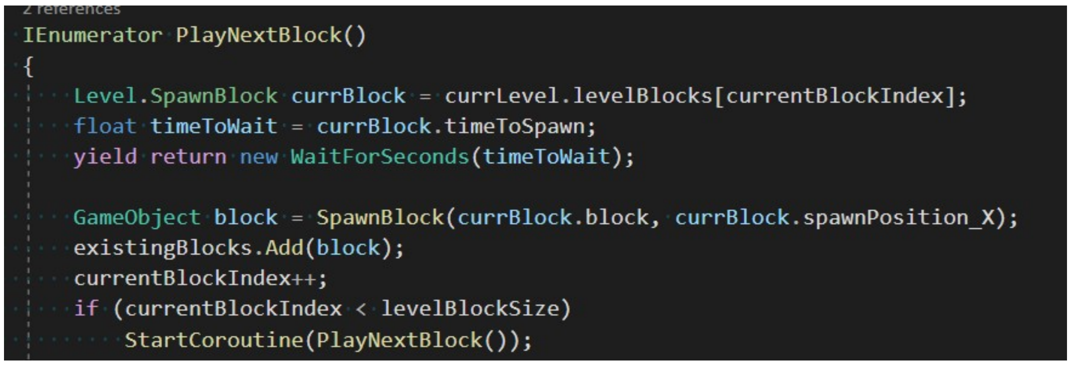
Picture2: Level selection screen

3 Development

3.1 Code Organization

Because the game is very simple, it was made in a single scene. The LevelController script loads the background scene into the main game on loading the game (this allows changing the background quickly). The game logic was mostly implemented with Unity Coroutines. Since the falling blocks are programmed to fall at certain times, Coroutines allow for reasonable time control and easy to maintain code. This project allowed me to improve my knowledge about Coroutines.

In Picture3 a fragment of the coroutine can be examined in detail. The **IEnumerator** starts by getting the obstacle that is supposed to spawn next and waits for the right time by yielding the **WaitForSeconds** method. When the correct time has passed, the coroutine spawns the obstacle and starts the next coroutine with the **StartCourotine** method.



```
2 references
IEnumerator PlayNextBlock()
{
    Level.SpawnBlock currBlock = currLevel.levelBlocks[currentBlockIndex];
    float timeToWait = currBlock.timeToSpawn;
    yield return new WaitForSeconds(timeToWait);

    GameObject block = SpawnBlock(currBlock.block, currBlock.spawnPosition_X);
    existingBlocks.Add(block);
    currentBlockIndex++;
    if (currentBlockIndex < levelBlockSize)
        StartCoroutine(PlayNextBlock());
```

Picture3: Coroutine code sample

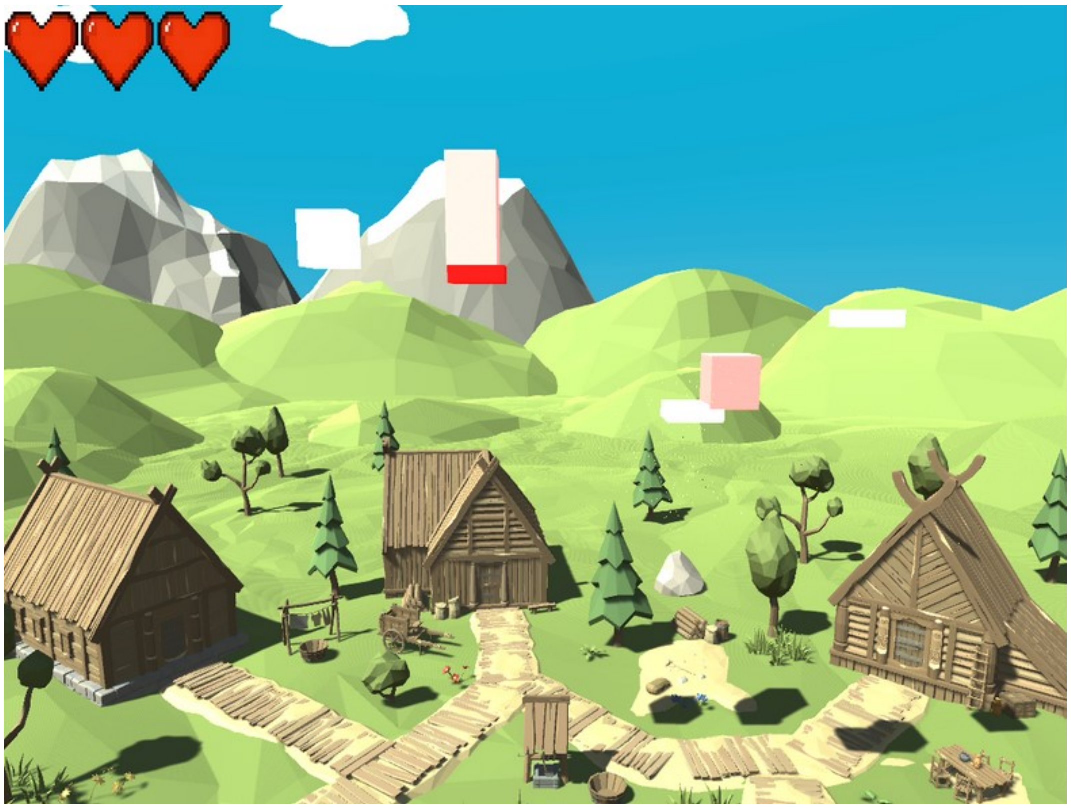
The PlatformController script consists of an ObjectPool of platforms (hence the name PlatPool) that controls the spawning and locking of the platforms.

3.2 Sound

All used sounds are from mixkit.co. The sound design was relatively simple. The SoundManager class is a Singleton (the actual Singleton pattern code is not my implementation) that controls all sound present in the game.

3.3 Assets

All visual assets (background scene and particle effects) are from the Unity Asset Store.



Picture4: Gameplay Screenshot

4 Conclusion

Overall, the development of a game like PlatPool does not require much expertise in Game Development or even Programming. However, this development took into account good programming practices and tried to keep the game's components as clean and organized as possible. Keeping an organized project and code-base is vital to have a reliable and easy to expand on the project.

5 Links

Source Code and executable: https://github.com/JoeMGomes/GISS/releases/tag/Release