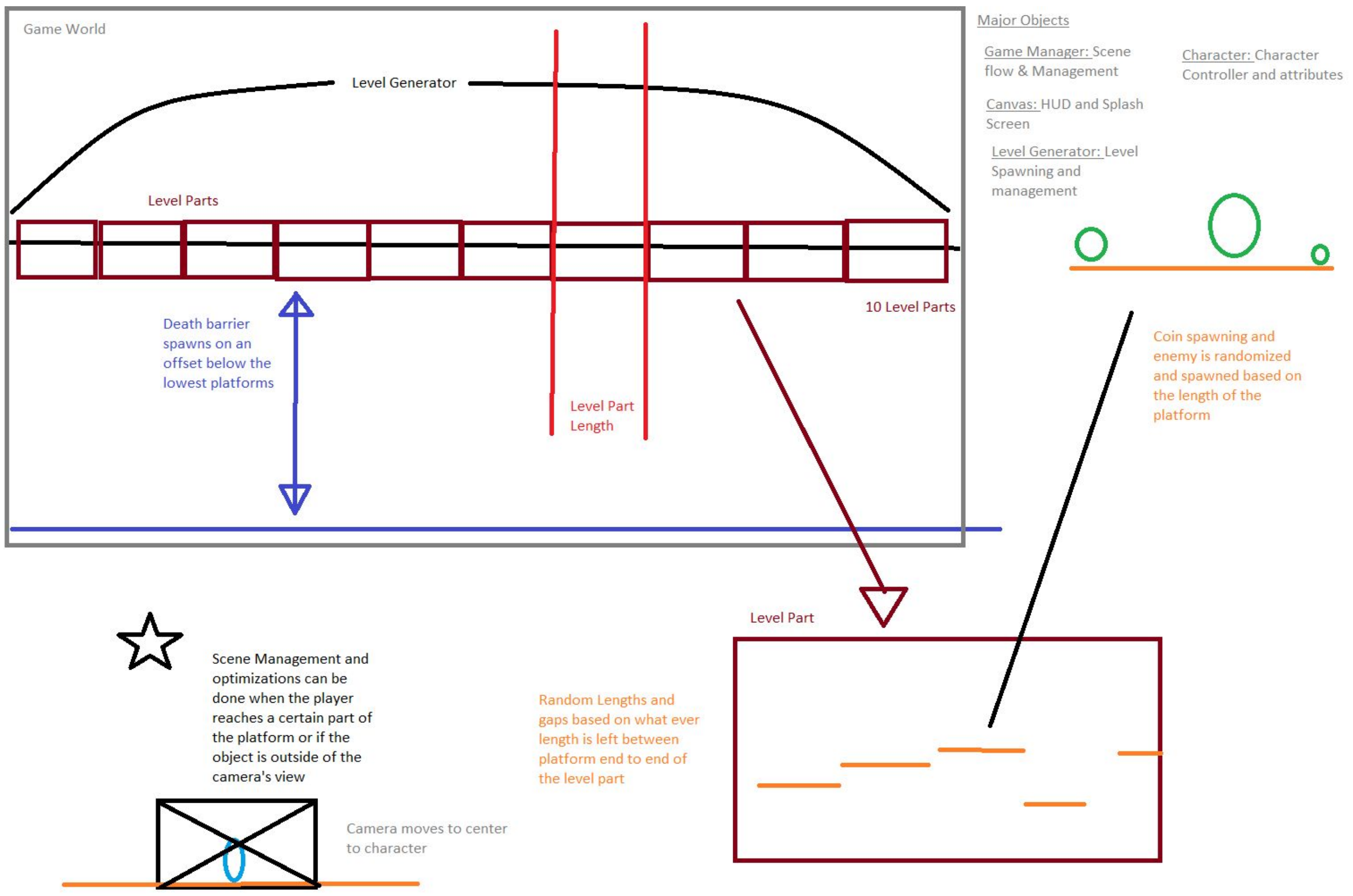
3770- Final Project

Date of Revision: 2020-12-14
Group Number: 14
Group Members: Cassandra Horne, Zachary Innes, Anh Nguyen, Vlad Roata, Wynne Wu

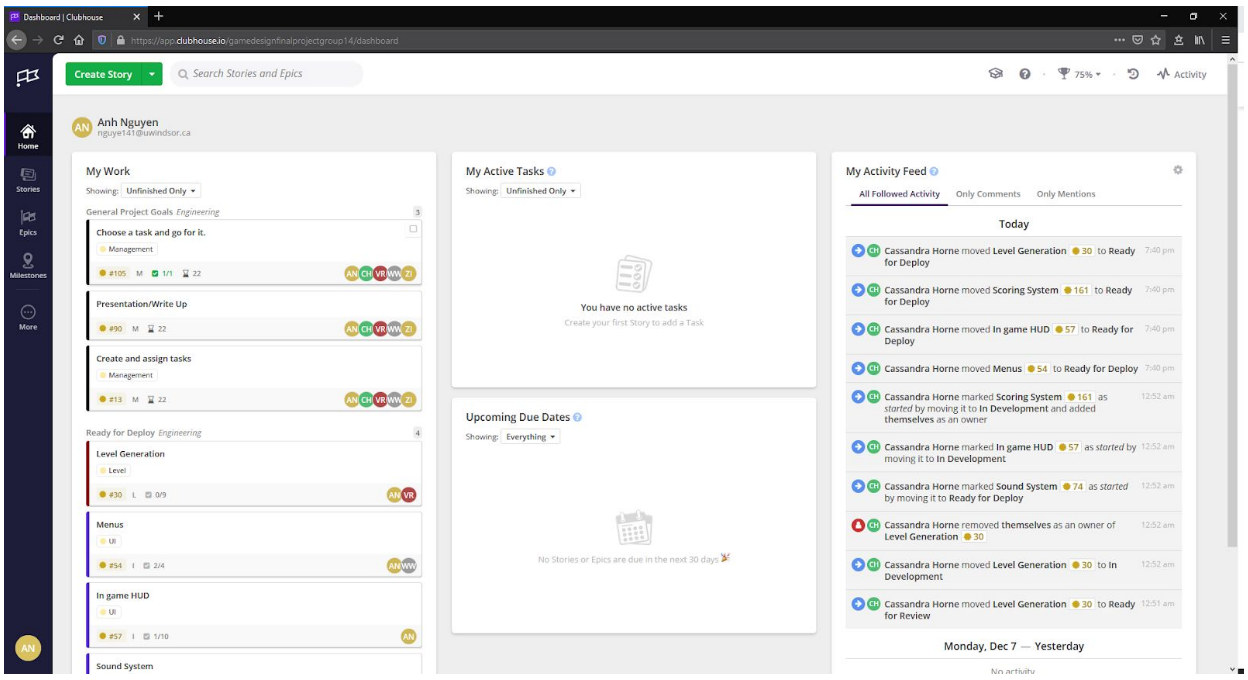
Key Concept
The story is simple, a plague doctor is trying to get home, but must destroy all kinds of viruses and bacteria along the way. The conflict in tones between the background and music reflect the mayhem and turmoil in the doctor's world. Things that don't seem so frightening can easily lead to a fiery downfall for our protagonist. The enemies our protagonist will meet are not the kind that are intimidating in an obvious way, but can bring a prompt end to the doctor's short life when confronting them. They are wild and unique, forcing the doctor to be resourceful and innovative in traversing the path home.

Game Description
This is a platforming game, with procedurally generated levels and many enemies and powerups. The objective is to get the plague doctor from the left end of the level to the right end of the level. Below is a design of the initial Level Generation and Game Scene overview. Notice how these objects retain a generic scheme to represent the modular and customizable design of our LevelGenerator.


Game World
Level Generator
Level Parts
10 Level Parts
Death barrier spawns on an offset below the lowest platforms
Level Part Length
Major Objects
Game Manager: Scene flow & Management
Character: Character Controller and attributes
Canvas: HUD and Splash Screen
Level Generator: Level Spawning and management
Coin spawning and enemy is randomized and spawned based on the length of the platform
Level Part
Random Lengths and gaps based on what ever length is left between platform end to end of the level part
Scene Management and optimizations can be done when the player reaches a certain part of the platform or if the object is outside of the camera's view
Camera moves to center to character

Planning and Workflow

For workflow, we decided to use a project management website called Clubhouse to manage tasks and outline stories and epics (milestones). Our project is source controlled by Unity Collab, instead of trying to use GitHub. To communicate we used Discord and held meetings once a week in the beginning and started holding daily meetings toward the end. This proved very effective for synchronizing work that was done independently by all team members. Using Discord also proved as an effective communication tool with the functionality to create an organized space to share files, post pictures, describe designs and allow group chatting to compromise for not being able to physically meet up. Below is an overview image of Clubhouse.



Anh Nguyen
nguye141@uwindsor.ca

My Work
Showing: Unfinished Only

General Project Goals *Engineering* 3

Choose a task and go for it.
Management
#105 M 1/1 22 AN CH VR WW ZI

Presentation/Write Up
#90 M 22 AN CH VR WW ZI

Create and assign tasks
Management
#13 M 22 AN CH VR WW ZI

Ready for Deploy *Engineering* 4

Level Generation
Level
#30 L 0/9 AN VR

Menus
UI
#54 I 2/4 AN WW

In game HUD
UI
#57 I 1/10 AN

Sound System

My Active Tasks
Showing: Unfinished Only
You have no active tasks
Create your first Story to add a Task

Upcoming Due Dates
Showing: Everything
No Stories or Epics are due in the next 30 days

My Activity Feed
All Followed Activity | Only Comments | Only Mentions

Today

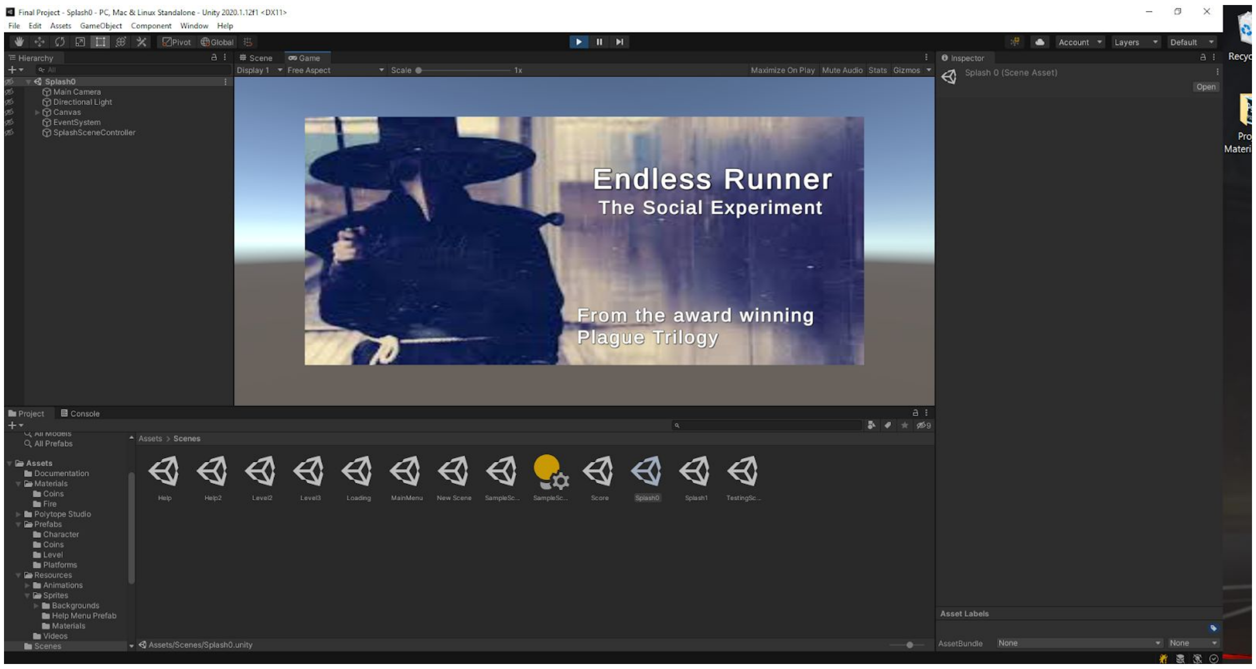
Cassandra Horne moved Level Generation 30 to Ready for Deploy 7:40 pm
Cassandra Horne moved Scoring System 161 to Ready for Deploy 7:40 pm
Cassandra Horne moved In game HUD 57 to Ready for Deploy 7:40 pm
Cassandra Horne moved Menus 54 to Ready for Deploy 7:40 pm
Cassandra Horne marked Scoring System 161 as *started* by moving it to In Development and added themselves as an owner 12:52 am
Cassandra Horne marked In game HUD 57 as *started* by moving it to In Development 12:52 am
Cassandra Horne marked Sound System 74 as *started* by moving it to Ready for Deploy 12:52 am
Cassandra Horne removed themselves as an owner of Level Generation 30 12:52 am
Cassandra Horne moved Level Generation 30 to In Development 12:52 am
Cassandra Horne moved Level Generation 30 to Ready for Review 12:51 am

Monday, Dec 7 — Yesterday
No activity

Game Elements and Techniques

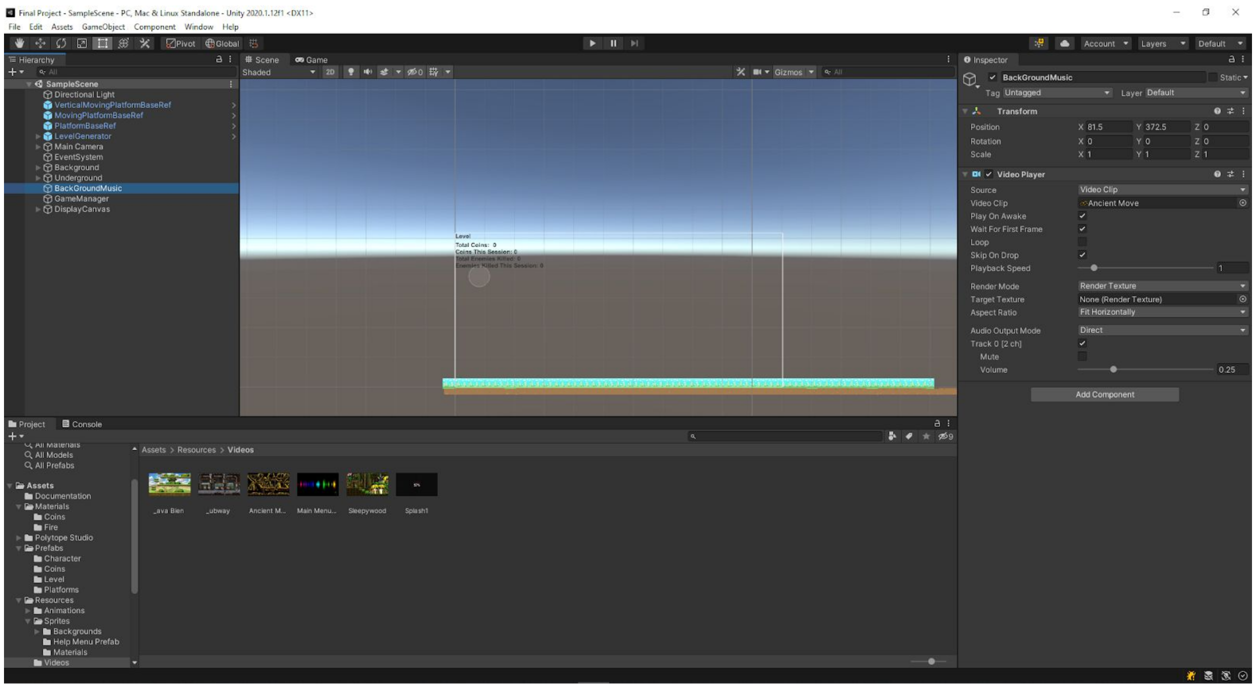
Splash Scene

We created two splash scenes for which the first splash scene displays the name of the game and the second splash scene displays a video with music. The video was made with the window app Paint and Photo. The Video was attached through the video player function in GameObject. In order to play the game, the first, second and main menu scene navigation is handled by a script called SplashController which loads the first splash scene when the scene number is 0 and displays the second splash scene when the scene number is 1. Both of them are handled by a function which delays each scene transition by 3 seconds to mimic loading. Lastly, after both splash scenes are displayed, the scene number will turn to 2 and will direct the game flow to the main menu. The scene number is built through the build settings in the File tab on the top left of the Unity project. Below is the splash screen example with SplashController



Background Music

There is a selection of music attached to the project. The music was attached through the Video Player function component in a GameObject. Once it's attached, we adjust the music's volume to our liking. Below is the Background music object in a Level Scene.



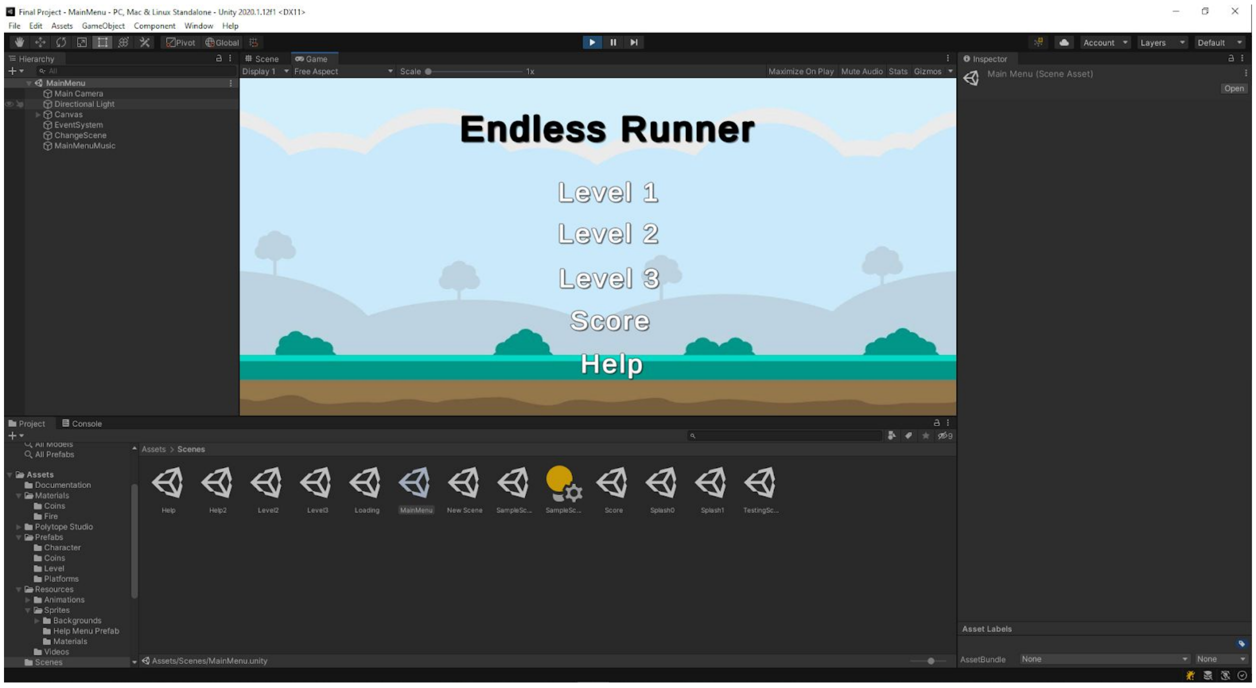
BackGroundMusic
Tag Untagged Layer Default

Transform
Position X 81.5 Y 372.5 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Video Player
Source: Video Clip
Video Clip: Ancient Move
Play On Awake: ✓
Wait For First Frame: ✓
Loop:
Skip On Drop: ✓
Playback Speed: 1
Render Mode: Render Texture
Target Texture: None (Render Texture)
Aspect Ratio: Fit Horizontally
Audio Output Mode: Direct
Track 0 [2 ch]: ✓
Mute:
Volume: 0.25

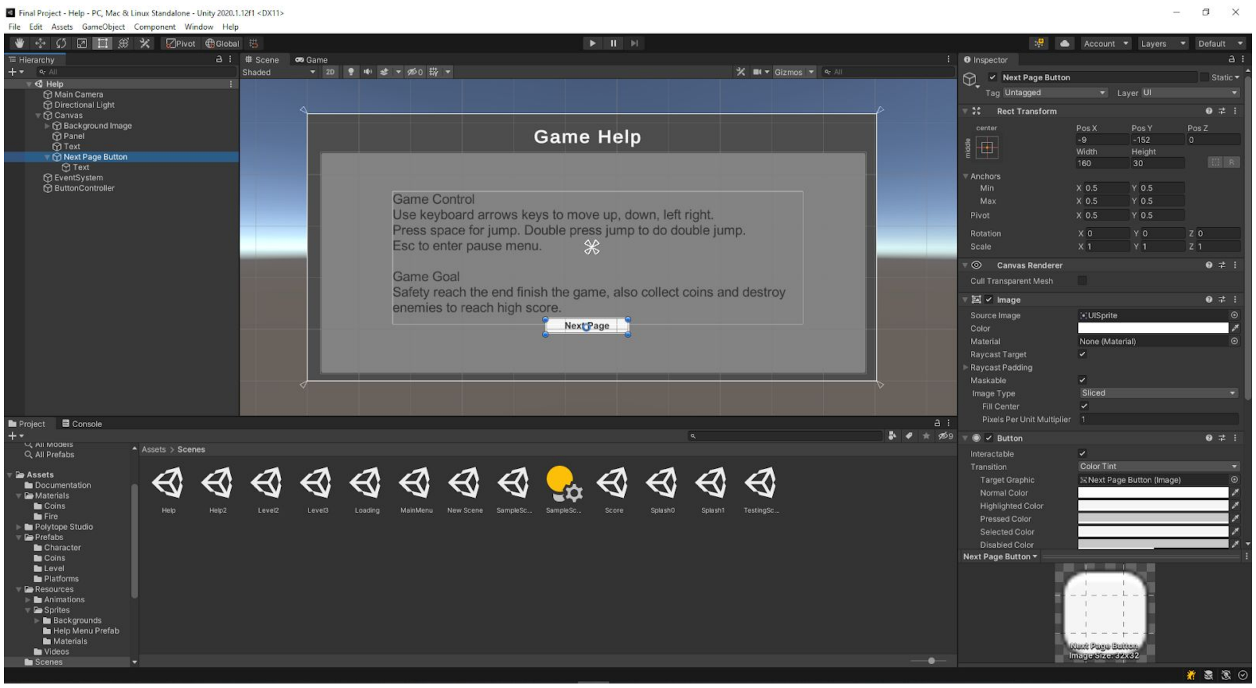
Main Menu

We first created Canvas then created UI (image) in the Canvas. We attached the background image into that UI (image) to display the background image we want. We also created buttons that include various levels, help menu, and score menu on individual panels of a canvas such as a Win screen. In order to direct to the various buttons we leveraged the onClick event on the button and passed in gameObjects with attached scripts to allow transitions between (Level 1, Level 2, Level 3, Score, Help). Example MainMenu scene.



Help Menu

A Canvas was first created then we created a Text object inside the Canvas to describe the game. A Button controller script was attached in order to direct the next help page or let the player go back to the menu.



Game Help

Game Control
Use keyboard arrows keys to move up, down, left right.
Press space for jump. Double press jump to do double jump.
Esc to enter pause menu.

Game Goal
Safety reach the end finish the game, also collect coins and destroy enemies to reach high score.

Next Page

Inspector
Next Page Button
Tag Untagged Layer UI

Rect Transform
Pos X -9 Pos Y -152 Pos Z 0
Width 160 Height 30
Anchors
Min X 0.5 Y 0.5
Max X 0.5 Y 0.5
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Canvas Renderer
Cull Transparent Mesh

Image
Source Image UISprite
Color
Material None (Material)
Raycast Target ✓
Raycast Padding
Maskable ✓
Image Type Sliced
Fill Center ✓
Pixels Per Unit Multiplier 1

Button
Interactable ✓
Transition Color Tint
Target Graphic Next Page Button (Image)
Normal Color
Highlighted Color
Pressed Color
Selected Color
Disabled Color

Level Generation

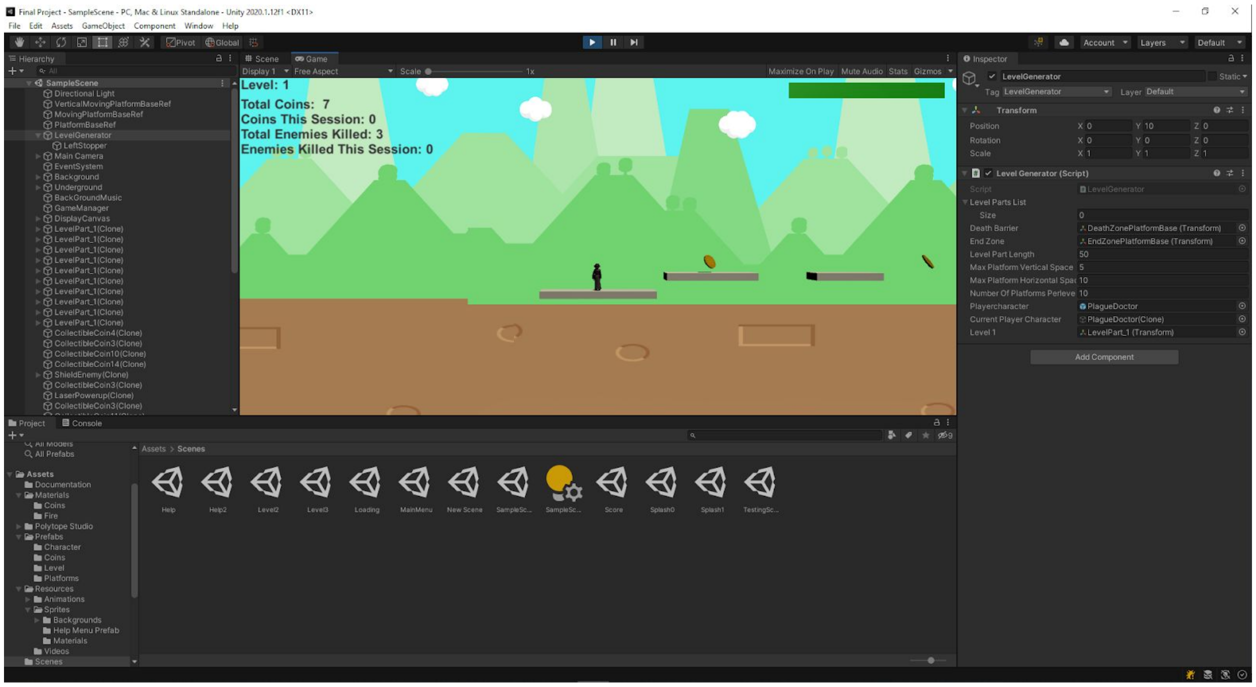
Level generation is handled by one master parent object called LevelGenerator. When the game initializes, the LevelGenerator prefab would instantiate and then repeatedly instantiate the specific amount of LevelPart prefabs contained in it's LevelPartsList property. In combination with a specified LevelPartLength attribute, LevelGenerator will instantiate Level Part prefabs based on the length of each level part to build a Level. The Level Generation prefab controls and maintains the child elements LevelParts.

As a LevelPart is being instantiated, each LevelPart prefab will then randomly generate platforms based on a specified set of unique platforms prefabs (Vertically moving, Horizontally moving, and Stationary blocks), specified by gameObject properties. In turn, the Level Part manages each instantiated Platform with a private list.

Lastly, each platform has some basic properties such as an endpoint object on the end of the platform to let the parent or any foreign script to find out the position and length of the platform.

Each platform's scaling and position (Vertical or Horizontal) and gaps between randomly generated within the given Part Length Specification. The generation also takes into account the given Maximum Player height or horizontal movement to be able to achieve completable levels.

Below shows an example generation of the level shown in the Scene gameObject Library.



Inspector

LevelGenerator

Tag LevelGenerator Layer Default

Transform

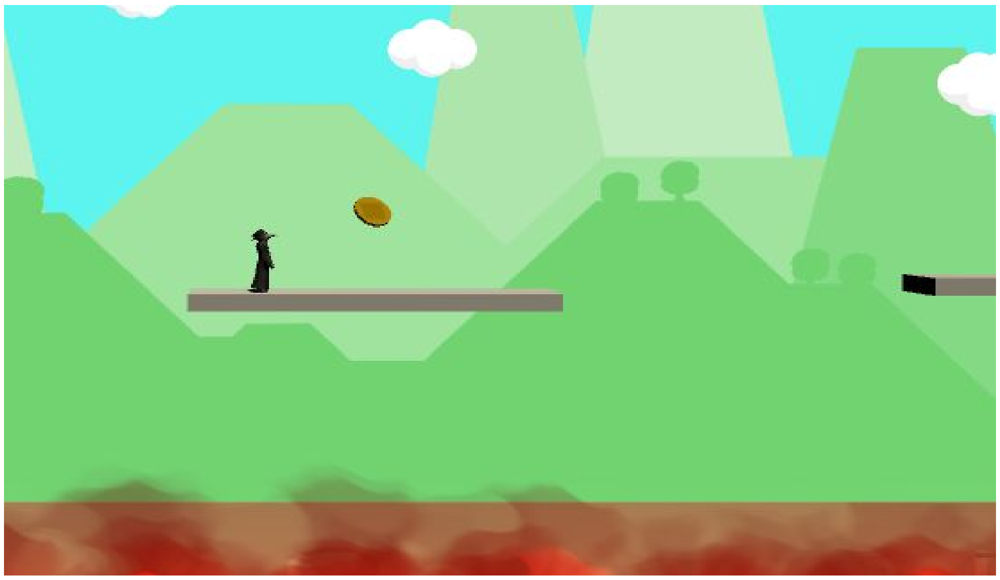
Position	X 0	Y 10	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Level Generator (Script)

Script	LevelGenerator
Level Parts List	
Size	0
Death Barrier	DeathZonePlatformBase (Transform)
End Zone	EndZonePlatformBase (Transform)
Level Part Length	50
Max Platform Vertical Space	5
Max Platform Horizontal Spac	10
Number Of Platforms Perleve	10
Playercharacter	PlagueDoctor
Current Player Character	PlagueDoctor(Clone)
Level 1	LevelPart_1 (Transform)

Level Item Population

Items (collectibles, powerups, and enemies) are generated above platforms at different heights, depending on the nature of the object through the attachment of the PopulatePlatformRandomly class. This class includes level controls which can enable or disable different collectibles, powerups, and enemies, depending on the level currently active. This class also controls the odds of an enemy, collectible, or power-up being created. The number of each of these items is left flexible in the class through the use of arrays. The image below shows a coin that is randomly generated.

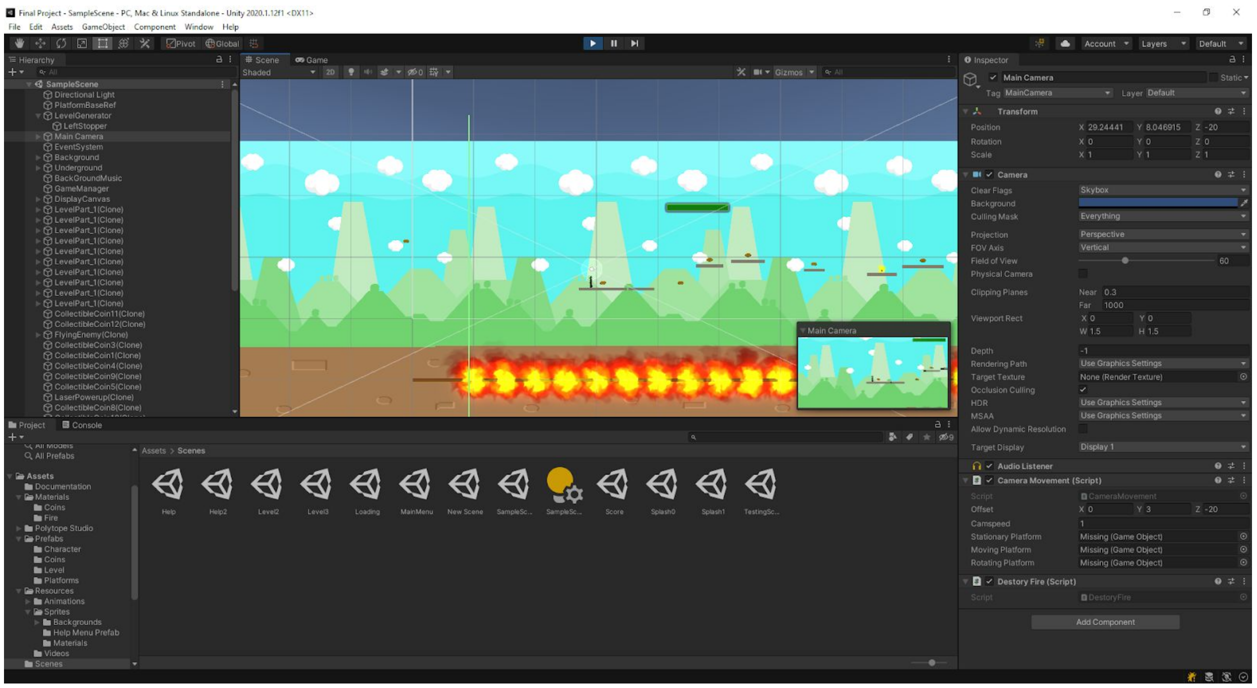


Camera and Character

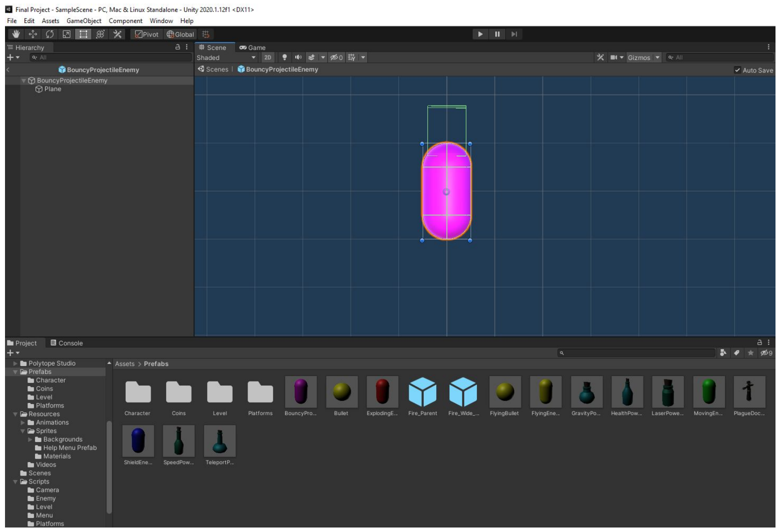
The player character can move up, down, left, jump, double jump, switch direction in mid air using keyboard arrow keys, and run by holding ctrl. We created a script called PlayerControllerAnimation in order to perform above action. The character can jump on an enemy to destroy them and collision with the side of the enemy would cause damage to the player. The script called EnemyHead allows the player to destroy the enemy and increase the kill enemy count by one every time the player jumps on top of the enemy's head. The enemy kill count is displayed on the top left display board and the count will be saved to high score.

When the player runs out of the HP or when the player falls off the platform a menu will pop out that displays a level failed message and buttons that allow the player to restart the level or go back to the main menu.

However, if the player successfully finishes the current level then the player will be moved to the next level. Once the player reaches the end of the last level then a menu will pop out and congratulate the player for finishing the last level. The Camera follows the player throughout the game and the main camera never moves left. Lastly, the camera makes sure the player does not move out of the screen.



```
Inspector
Main Camera                          Static
Tag MainCamera        Layer Default
Transform
Position      X 29.24441   Y 8.046915   Z -20
Rotation      X 0          Y 0          Z 0
Scale         X 1          Y 1          Z 1
Camera
Clear Flags             Skybox
Background
Culling Mask            Everything
Projection              Perspective
FOV Axis                Vertical
Field of View           60
Physical Camera
Clipping Planes         Near 0.3
                        Far  1000
Viewport Rect           X 0     Y 0
                        W 1.5   H 1.5
Depth                   -1
Rendering Path          Use Graphics Settings
Target Texture          None (Render Texture)
Occlusion Culling       ✓
HDR                     Use Graphics Settings
MSAA                    Use Graphics Settings
Allow Dynamic Resolution
Target Display          Display 1
Audio Listener
Camera Movement (Script)
Script                  CameraMovement
Offset                  X 0   Y 3   Z -20
Campspeed               1
Stationary Platform     Missing (Game Object)
Moving Platform         Missing (Game Object)
Rotating Platform       Missing (Game Object)
Destory Fire (Script)
Script                  DestoryFire
Add Component
```



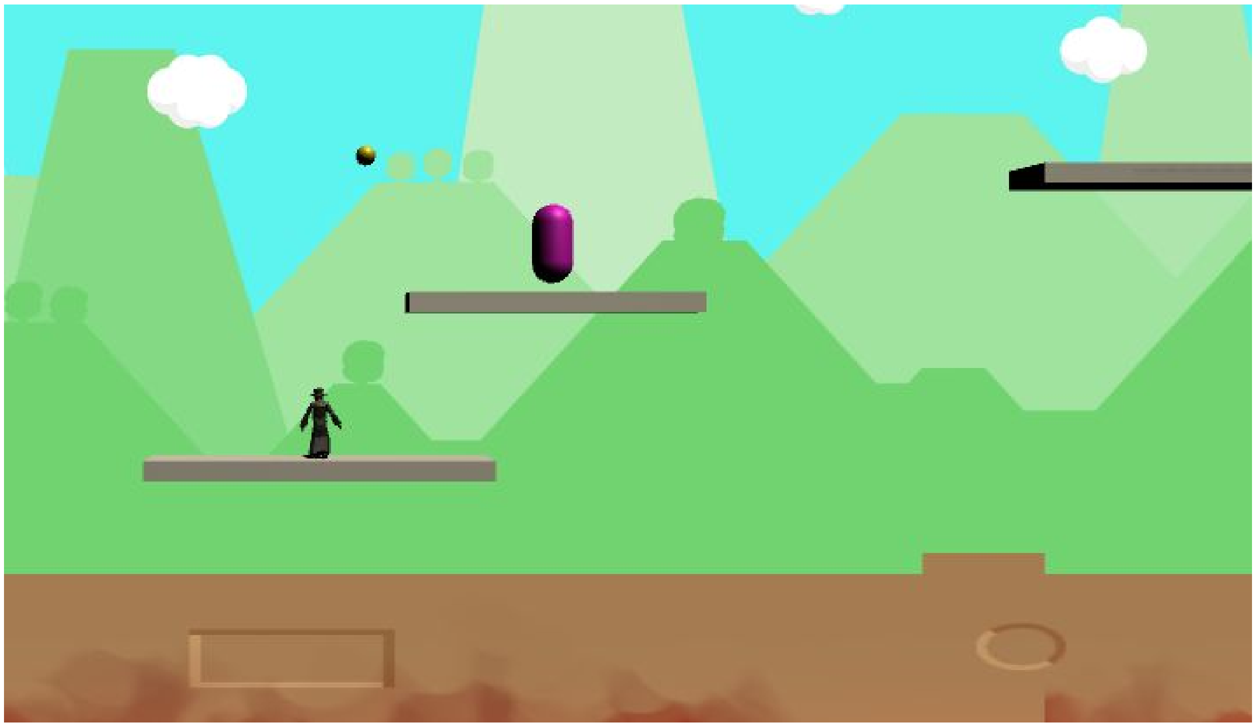
Example rigged Plague Doctor asset as well as a virus enemy shown above.

Enemies and Health

Our game has 5 enemy types:

1. An enemy that teleports on top of the player and explodes, dealing damage and pushing the player back
2. An enemy that shoots a projectile towards the player which bounces
3. An enemy that flies towards the player
4. An enemy with a shield which takes two head bounces to kill
5. An enemy that moves toward the player and shoots two beams out beside it which damage the player

Each of these enemies deals damage to the player, which is displayed by the health bar at the top right of the screen. When the health bar reaches zero, the level ends and an option to restart or return to the main menu appears. Falling into the fire below also instantly sets the player’s health to zero. Below is an appearance of a virus shooting a lethal projectile at the character!

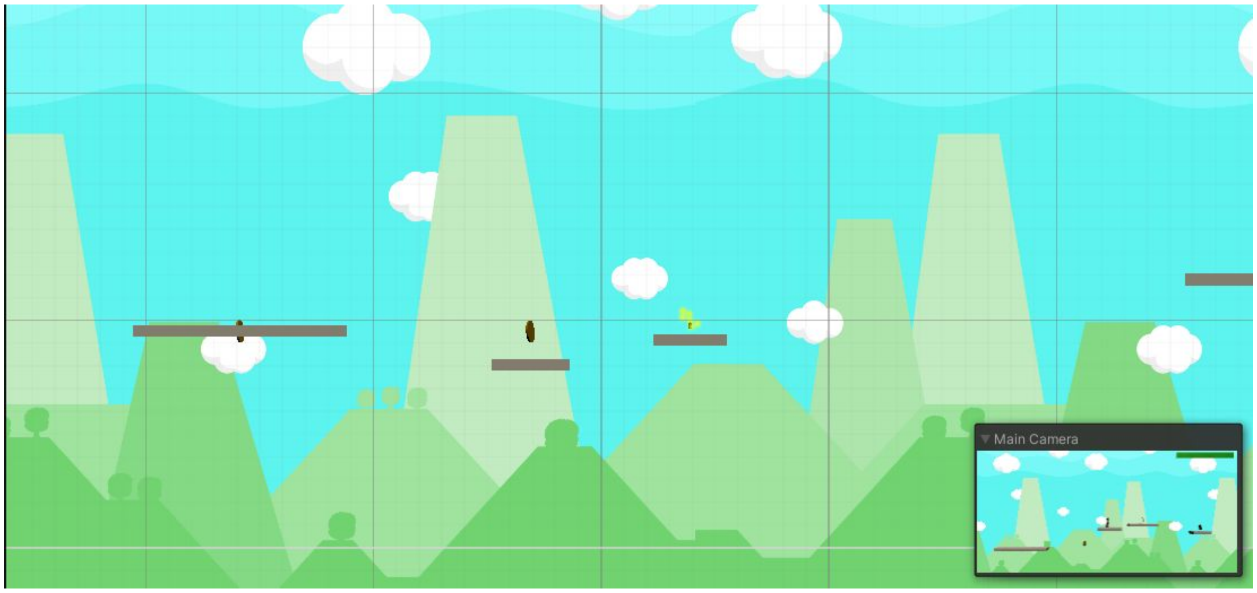


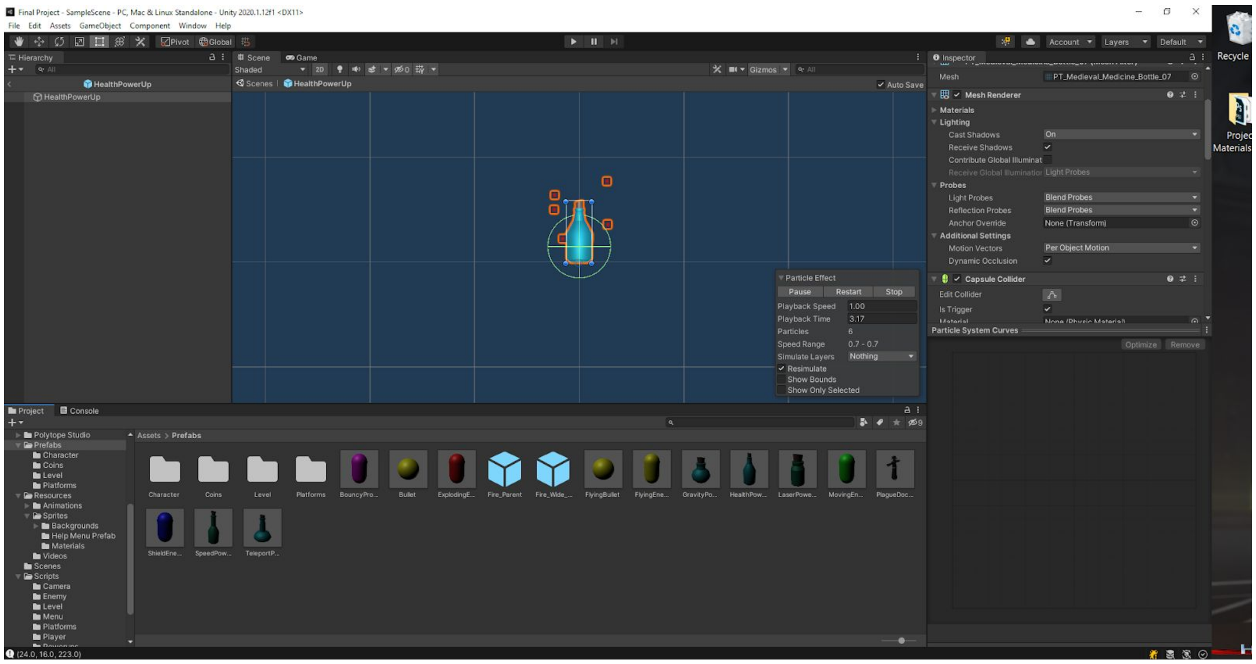
Powerups and Collectibles

Our game has 5 powerups:

1. A powerup to increase the player's speed
2. A powerup that gives the player a laser attack by left clicking
3. A powerup that heals the player
4. A powerup that allows the player to teleport
5. A powerup that decreases gravity, allowing the player to jump higher

Each of these powerups can spawn randomly on the platforms and slightly changes the player's colour as a visual indicator. The powerups are all controlled by the PlayerController script, which handles their activation and usage. Below are example images that showcase a randomly generated powerup in a randomly generated level as well as the powerup prefab image.





Mesh PT_Medieval_Medicine_Bottle_07

Mesh Renderer

Materials

Lighting

Cast Shadows On

Receive Shadows ✓

Contribute Global Illuminat

Receive Global Illuminatior Light Probes

Probes

Light Probes Blend Probes

Reflection Probes Blend Probes

Anchor Override None (Transform)

Additional Settings

Motion Vectors Per Object Motion

Dynamic Occlusion ✓

Capsule Collider

Edit Collider

Is Trigger ✓

Material None (Physic Material)

Particle System Curves

Optimize Remove

Particle Effect

Pause Restart Stop

Playback Speed 1.00

Playback Time 3.17

Particles 6

Speed Range 0.7 - 0.7

Simulate Layers Nothing

✓ Resimulate

Show Bounds

Show Only Selected