Documentation of Sky Thrill

SKYTHRILL

How to start the program

Unzip the folder shared with google drive and you will find a file name SkyThrill Application file and you can open it up. The goal of this game is to reach the goal platform and proceed to the next level. I have some feedbacks from people not used to first person game that this game is a bit too hard so I have written some sort of guide at the end of this documentation that you can read to beat the game.

Control

Movements – Standard WASD to move around

Sprinting – Hold shift to sprint

Jump – Spacebar

Crouch – press C to crouch

Slide – Hold onto shift and crouch to slide

Wall running – Jump and hold onto the direction (a/d) of the wall to wall jump for a short distance, you need to jump again after the short distance otherwise you would fall.

Sliding on the slope – Shift and Crouch but you have to break the slide with jump

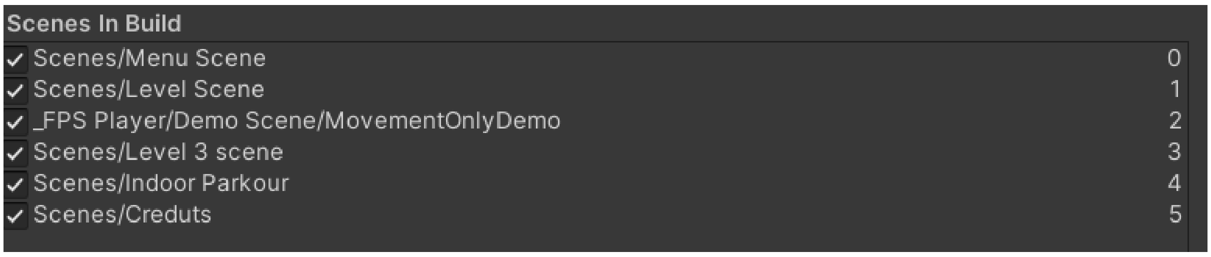
Climb up the ledge – You can press spacebar or W to climb up the ledge

Vaulting – Some walls allow you to jump over, you simply needs to press spacebar near it.

Game Architecture

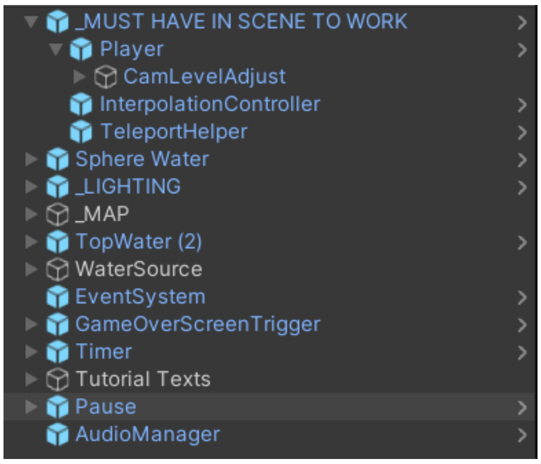
Scenes

The game consists of 6 different scenes in which 3 of them are levels and 3 of them are UI Panels. Scene transitions are managed by SceneManagements and adding it to the build setting, scene transitions between levels are done via triggers (by entering the goal platform and unlocking the next level).



Scenes In Build	
✓ Scenes/Menu Scene	0
✓ Scenes/Level Scene	1
✓ _FPS Player/Demo Scene/MovementOnlyDemo	2
✓ Scenes/Level 3 scene	3
✓ Scenes/Indoor Parkour	4
✓ Scenes/Creduts	5

Prefabs



```
_MUST HAVE IN SCENE TO WORK
    Player
        CamLevelAdjust
    InterpolationController
    TeleportHelper
Sphere Water
_LIGHTING
_MAP
TopWater (2)
WaterSource
EventSystem
GameOverScreenTrigger
Timer
Tutorial Texts
Pause
AudioManager
```

To reuse some of the components, I have made all of the objects needed in the Scene to be a prefab.

Player Mechanics

All of the movements and camera controls are adopted from the youtube channel https://www.youtube.com/channel/UCYVyyyttdLFyZemFsEyPLCw, in the PlayerController class, it has a status variable to check which state the player is in, in this game it is idle, moving ,vaulting, crouching, sliding, grabbing and climbing ledge, swimming, climbing ladder, wall running, each of these status has a unique movement and specific check to see which state the player is in. For example, for grabbing ledge, you need to be up against a ledge and falling (with negative y move direction), and to climb up the ledge it checks that you are grabbing the ledge and input w (have y positive move direction). The checks are done in Update() and FixedUpdate(), crouching is checked in Update() since you still have controls of the player while the other movements are being checked in FixedUpdate() so that it will be in sync with the physics engine. To check which surfaces the player is on/interacting with, LayerMasks are used and all the different surfaces are added to the layers of the games.

Teleportation Helper

This is a trigger box that checks if the player has fallen out of the map (when the player has triggered this area it will teleport the user), in which it will make the public static Boolean dead to be true, and if it is true it will teleport the user back to the starting location.

Pause

Pause UI panel is normally not visible unless Esc key is pressed, it is being checked in an Update() function and when the game is paused the public static boolean isGamePaused would be true in which it will make the Timer, Player and Camera control to stop using `Time.timeScale = 0f;`

Timer

Timer is started with Start() function and updated in the Update() function if and only 1) if game is not completed, 2) game is not paused. If the player has died (public static Boolean dead being set to true), the timer would reset, otherwise it will keep counting down until it reaches 0 in which the player would restart the game automatically.

GameOverScreen Trigger

This is an object with onTrigger enable in which if it has detected the player passing through the place it will make the end screen visible in which the player can select to quit or proceed to the next level.

AudioManager

An instance of AudioManager is present in all of the scenes, it is done by having a check in its awake button to see if there is an instance of AudioManager already and if it has it won't instantiate a new instance of it, AudioManager is responsible for playing the in game theme and the sound effects (which have not been added to the game). It works by having a list of sounds and checking which sound effects are supposed to be player using triggers/layer masks.

Testing (Mainly use of Debug.Log())

Before implementing a function, I would just use Debug.Log to see if it is indeed interactable and is doing what I want it to do when certain inputs are pressed, for example before implementing Pause I would use Debug.Log while checking for the Esc Input to see if the debuggers have written out words in the console.

Other testing is done for this game via extensively playing through it while ensuring that the game is indeed winnable. One bug was found in which after the user has reached the goal platform and clicked the menu button, it will not save his progress, this is easily solved by changing the save action to be done when user has reached the goal platform rather than when user has clicked the next button.

Saving and Unlocking Level

Saving of progress is done with PlayerPrefs, in which the user is initially allocated a levelAt variable of 2, and if the player has reached the goal platform, it will increment the levelAt variable by one and unlock the save level. It is initialised as below on the level scene.

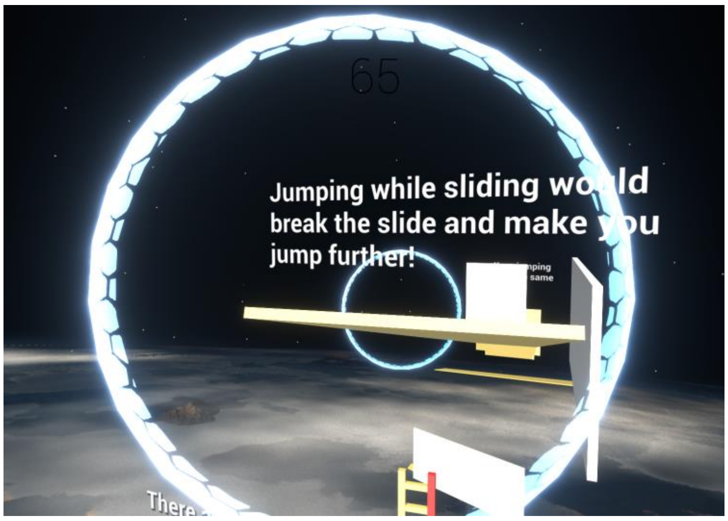
```
public Button[] lvlButtons;

    // Start is called before the first frame update
```

```
void Start()
{
    int levelAt = PlayerPrefs.GetInt("levelAt", 2);

    for (int i = 0; i < lvlButtons.Length; i++)
    {
        if (i + 2 > levelAt)
            lvlButtons[i].interactable = false;
    }
}
```

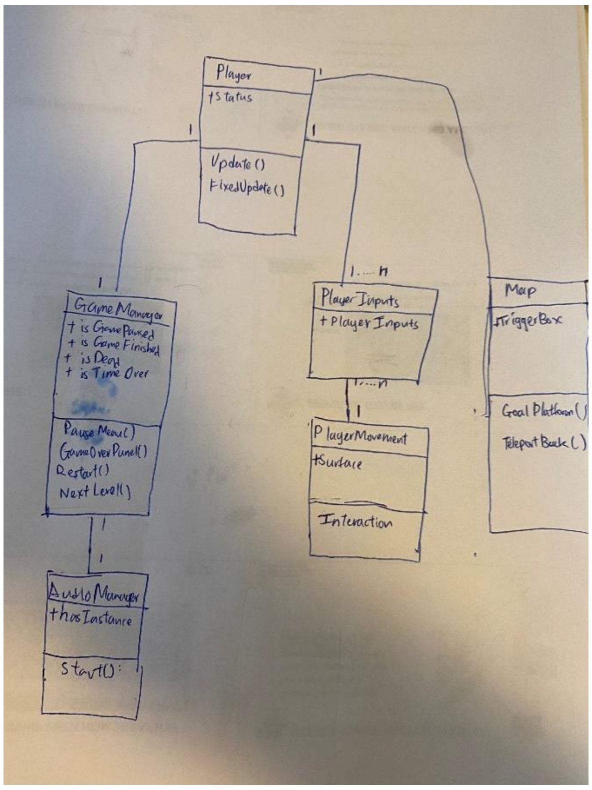
Aesthetic Aspect of the game


65
Jumping while sliding would break the slide and make you jump further!

Some post processing and shader graphs effects were used in this game, this is the most annoying bit of the project since Unity has updated their shader graphs and the naming and interfaces are slightly different from the tutorials shown on Youtube, it takes a while to read around and know what each thing does and what Unity has changed the naming convention to.

To improve the maps, I downloaded some free HDR skybox that make improve the overall aesthetics of the map.

UML Class Diagram



This is roughly what the classes do, the map is responsible for having a trigger box that would teleport the user backs if the user has fallen out of the map, GoalPlatform() that would be triggered if the user has reached it. Player class has a status to indicate the movement of the player such as sliding,crouching climbing over the ledge etc as mentioned above. It is achieved by the PlayerInput Class and PlayerMovement class which check for the player's inputs and what surface it is on. The game manager is responsible for restarting the game when timer runs out, pauses the game, moves the user to the next level and shows the GameOverPanel. The audio manager on the other hand would be instantiated if there is none and would play the theme song, the audio manager can implement the sound effects too to each surface but it would take another few hours of works presumably.

How to beat the game

Tutorial

Climb up the ladder, you will need to sprint and jump towards the ledge which is considerably far away, so you need to back up a bit from the ladder and hold shift and jump to jump towards the ledge, you would not fall since you can hold onto the ledge.

To speed run, you can try sprinting and jumping towards the white wall right away and wall run to the red goal platform.

Space

This is where most people would fail, you need to jump onto the ladder, climb up to the top of the ladder, and do this series of movement (a/d + jump + move your mouse to face the ladder behind), this level should be easy after a few tries.

After that, you should have no problem until you reach the slope, you would slide on the slope and jump to break the sliding. Then wall run against the white wall to reach the next slope, you would have to slide and break the jump again and reach the second white wall to get to the final slope, in which you can easily slide onto the goal platform

To speed run, you can ignore the second white wall in which you can jump directly from the second slope to the third slope.

Indoor Parkour

This level is pretty straight forward in which you would climb up and pass through all the obstacles! Many had no problems clearing this level and this was the most fun level to make.

How long it took me

The movements were adopted but to understand and use it, it took me about 15 hours. I tried to make a music for ~ 10 hours with Bosca Ceoil but ended up scraping it and found a commercial free music, all the aesthetics improvements and map building etc took the longest which is about 20 hours to learn and implement, the hardest thing would be working with shader graph and post processing volumes, since they never do what I intend them to do since there have been major updates with their shader graphs and most of the tutorials are obsolete.