**Overview**

Patients who suffer from severe burns are often limited in mobility. In addition to the inextricable pain from the burns themselves, patients are often discouraged from their loss of mobility and feel a desire to explore. Studies show that video games in particular can both reduce the pain experienced by burn patients and increase overall mood by placing the patients in an immersive environment with elements contrary to the things that may have experienced during their injury (<https://www.hitl.washington.edu/projects/vrpain/>).

ReliefFinallyis a virtual reality-enabled adventure-based video game which aims to de-stress burn patients by placing them in calm environments where they are enabled to explore, unlock new environments, and interact with various elements (e.g. snow, water, wind, ice, etc.) that stand contrary to the reason for their injury.

**Setting up the project for development**

**Requirements:**

Must be running windows.

Must have a GearVR-compatible phone if you wish to test the game on android.

Follow this link: <https://cgcookie.com/2015/12/16/quick-start-guide-gear-vr-unity/>, or follow the summary below.

Download latest version of Unity (<https://unity3d.com/unity> -> Personal -> Learn More -> Try Personal -> download installer) and install.

Install Android Studio [https://developer.Android.com/Studio/index.html](https://developer.android.com/studio/index.html)

Find the file path of the Android SDK and save it somewhere.

Install most recent JDK (google java jdk and download the Java Se Development kit), find the file path of the JDK, should be in C:\ProgramFiles\Java and appear as jdk-somenumbers or jdksomenumbers) and save the file path somewhere.

Download project from github: <https://github.com/takoda1/ReliefFinally>.

Open Unity and open the root folder of ReliefFinally to open the project.

In the editor window, go to edit->preferences.

Click External Tools.

Under Android, it should say SDK and JDK, put the saved file paths of the SDK and JDK into these boxes.

Next, in the editor, go to File -> Build Settings, click Android, then click Switch Platform.

In the same window, click on player settings, then the Android logo in the top bar.

Click Other Settings and under Identification, change the Minimum API level to 19 or above.

If you are using Visual Studio to edit code, make sure to also set the… (Experimental version 4.6 leads to crashes when run on the device)

In the same player settings window, click XR Settings and enable Virtual Reality Supported.

Now follow the directions in the **Building for the Gear VR section** of <https://cgcookie.com/2015/12/16/quick-start-guide-gear-vr-unity/>.

**Using Visual Studio to edit code**

<https://docs.unity3d.com/Manual/VisualStudioIntegration.html>

In Edit->Project Settings->Player, under configuration, make sure the Scripting Runtime Version is set to Stable(.NET 3.5 Equivalent). Builds to an android device will fail if the runtime version is 4.6 experimental as of 2/16/2018.

**Debugging the game on the Android device.**

Must have Android Studio or at least developer tools to debug.

It is easiest to have the Android device connected by USB for debugging. If you wish, you can also wirelessly connect your device for debugging by following the instructions under **Connect to a device over Wi-Fi** here: <https://developer.android.com/studio/command-line/adb.html>.

First you have to enable USB debugging on the Android device. In settings, find the Build Number setting, usually under About Phone, and click it at least seven times. This should enable Developer options. Next click on Developer options in the Settings screen. Scroll down in the Developer options screen and enable the USB debugging setting. Now, whenever the phone is connected to a machine it should prompt you to allow or prevent USB debugging from the said machine. Allow debugging from your machine when you connect the Android device.

There are two methods to see Logcat logs from the Android device. Non-gui Adb Logcat and Gui monitor. Both can be found deep within your file system and run from there, but it is easier to set them as system PATH variables and run them with short commands from the Command prompt (Instead of running C:\Users\username\AppData\Local\Android\sdk\platform-tools adb logcat, you can just type adb logcat and execute).

To set system PATH variables in windows 10, type View Advanced System Settings in the search bar, click Environment variables in the window that pops up, then under System variables, click Path to highlight it then click Edit. To add new environment variables, click New, then either type in the path or browse for it. The path for logcat and monitor.bat should be something like C:\Users\username\AppData\Local\Android\sdk\platform-tools and C:\Users\username\AppData\Local\Android\sdk\tools.

Now, open a command prompt (type cmd in search bar -> press enter) and type either adb logcat or monitor.bat and press enter to execute.

**Adb Logcat summary**

Keep in mind that I’m not very familiar with Logcat, this is a very general overview.

To view a specific log, click on the desired log to stop new ones from scrolling the logs up.

To make the logs continue scrolling after stopping the logs, press shift+enter.

To exit adb logcat press ctrl+c.

If you just type adb logcat, you get every single log from the android device, which is undesireable if you just want logs from the game. Instead, use the logcat options to filter for what you want: <https://developer.android.com/studio/command-line/logcat.html>.

**Monitor summary**

In the bottom half of the monitor, it should be displaying all logs from Logcat. In the search bar at the top, you can type in any string that you are looking for in the logs and it will automatically filter those logs for you to view.

**Project structure**

**Controllers**

Project compatible with either the GearVR controller or StratusXL controller.

GearVR Controller has no direct mappings, instead, inputs must be handled by code. Look at OVRGearVRControllerTest (Assets->OVR->Scripts->Util) to see which code maps to which button.

I’ve already synced the StratusXL Controller with this project, but read below if you want to change button functionality.

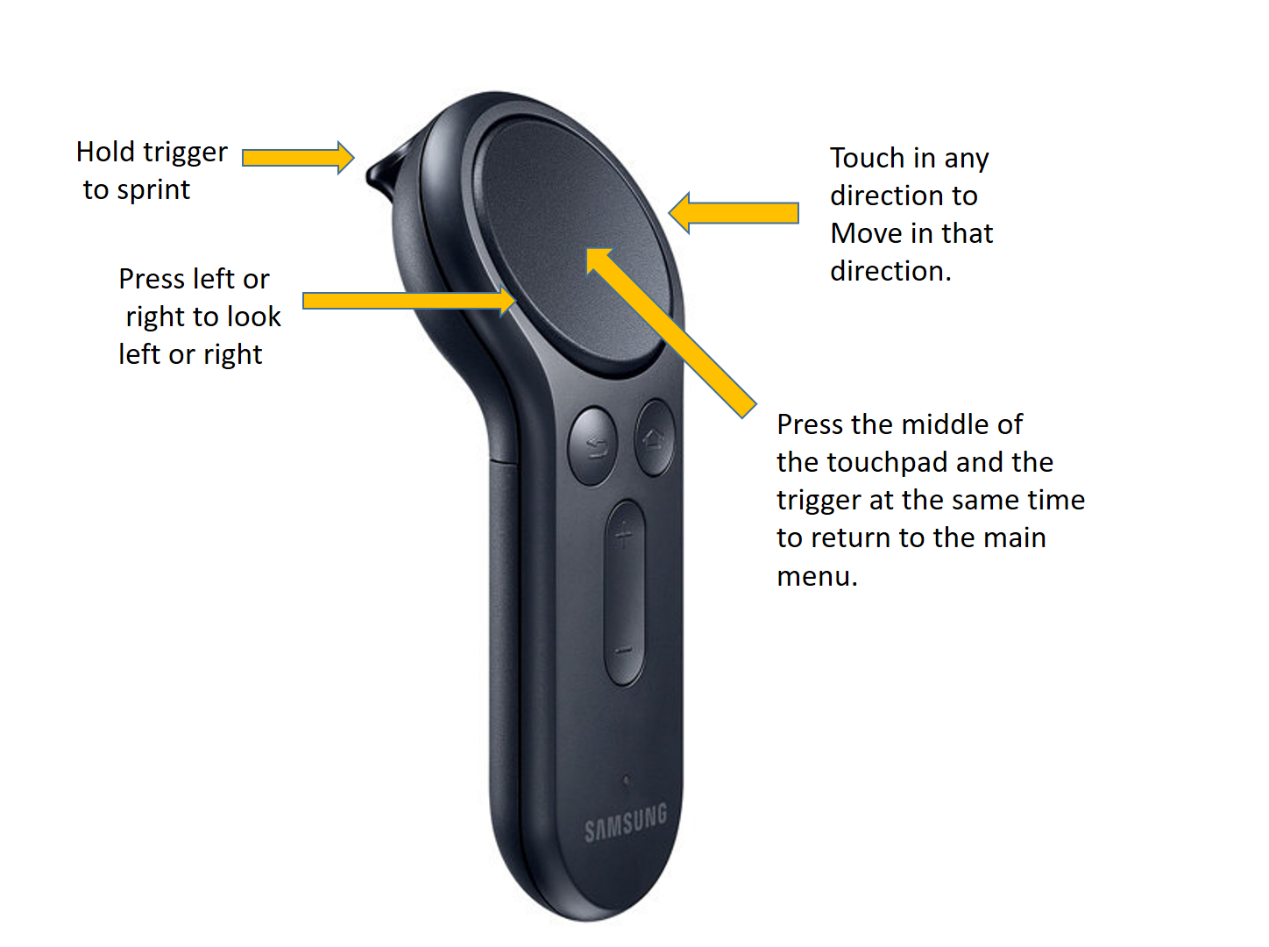
Syncing the StratusXL Controller with Unity:

<http://primeyoullc.net/steel-series-stratus-xl-game-controller-mapping-unity-3d/>

Button and axis mappings for the StratusXL for convenience.





These are the controls while playing in game.

**Scene Structure**

Scenes

MainMenuScene

InfoScene

TutorialScene

GrassyPlainsScene

BarnacleWatersScene

SnowyMountainScene

GrassyPlainsSceneControlled

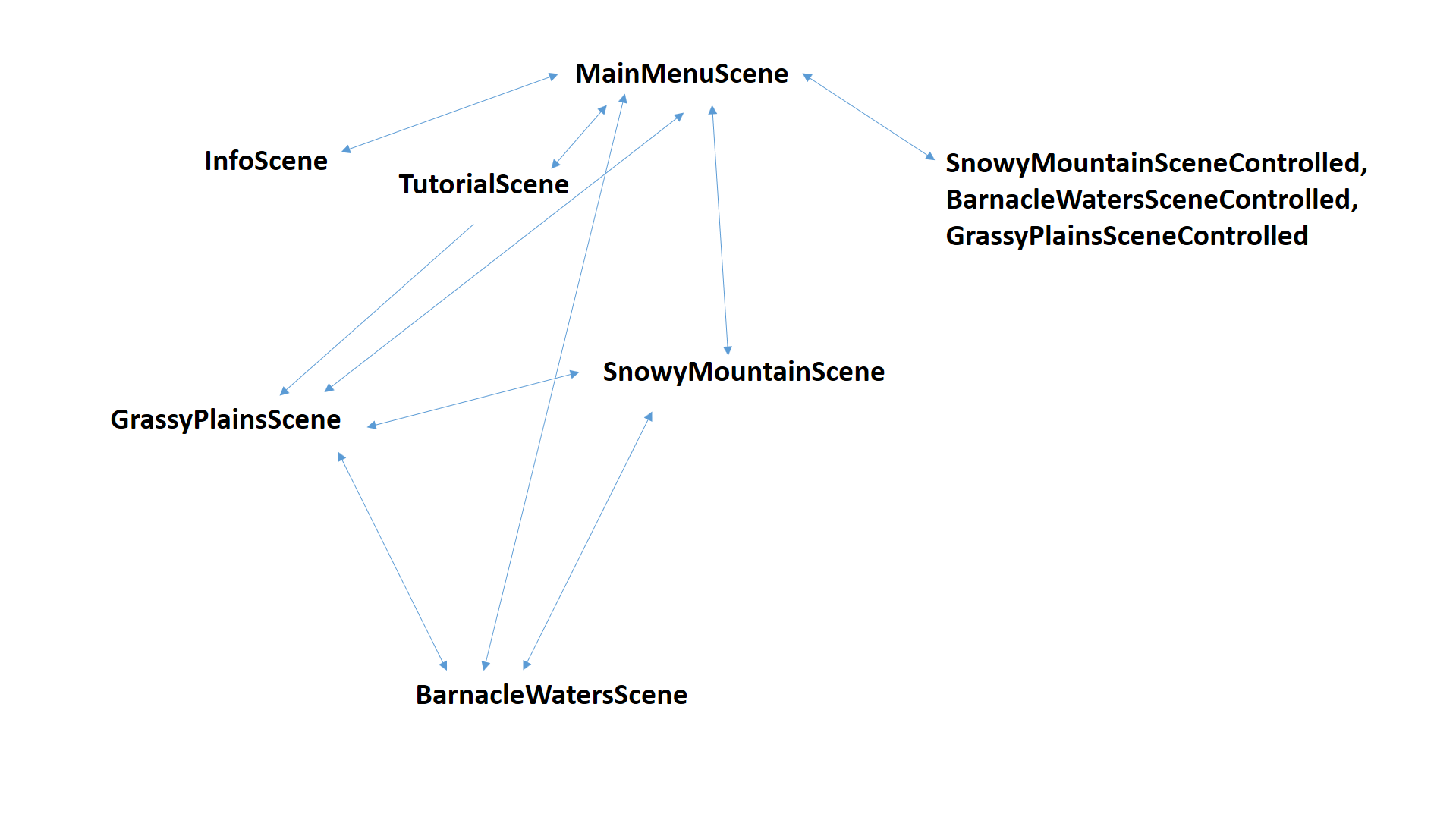
BarnacleWatersSceneControlled

SnowyMountainSceneControlled

Controls for the scenes

|  |  |  |
| --- | --- | --- |
| Scene | Look around with head | Controller movements |
| MainMenuScene | Y | N |
| InfoScene | Y | N |
| TutorialScene | Y | Y |
| GrassyPlainsScene | Y | Y |
| BarnacleWatersScene | Y | Y |
| SnowyMountainScene | Y | Y |
| GrassyPlainsSceneControlled | Y | N |
| BarnacleWatersSceneControlled | Y | N |
| SnowyMountainSceneControlled | Y | N |

Scene Interaction



Players can reach any scene from the MainMenuScene. Conversely, players can reach the MainMenuScene from any scene simply by pressing the trigger and the middle of the touchpad on the GearVR Controller or Y on the StratusXL. The end of the TutorialScene leads into the GrassyPlainsScene. The GrassyPlainsScene can be accessed from both SnowyMountainScene and BarnacleWatersScene without collecting any treasures. Treasures needed to access SnowyMountainScene are found in GrassyPlainsScene, treasures needed to access BarnacleWatersScene are found in SnowyMountainScene, and once all treasures needed to access a scene are found, the player can access that scene freely from either of the other two scenes excluding itself of course among GrassyPlainsScene, SnowyMountainScene, and BarnacleWatersScene. SnowyMountainScene and BarnacleWatersScene can be accessed directly from the MainMenuScene, but once you leave them for another scene among the three listed above, you won’t be able to return until you have collected the required treasures.

**Individual Scene Explanations**

**MainMenuScene**

Overview

The player looks around from a center pedestal at rectangular menu items that each represent a specific scene and can enter the desired scene by looking at it, which pulls the menu item forward to indicate that it is being hovered over, and pressing the trigger on the GearVR controller or A on the StratusXL.

Oculus already has prefabs that handle all the details of a camera for each eye and how they are tracked by head motion, so I placed the OVRCameraRig (Assets->OVR->Prefabs), which allows the player to look around in a scene, on top of the center pedestal.

To make the rectangular menu items react while being looked at, you need a script attached to the camera that sends out rays and see if the rays have hit a desired object, that being a menu item. If a ray has hit a menu item, call the appropriate methods on the menu item. Therefore, the menu item should also have a script with methods that allow it to perform behaviors.

Scripts

For the menu item, I made a general script that reacts to clicks and gaze and that will be subclassed, InteractiveItem, with methods Over, Click, Out, and Down. MenuItemInteractiveItem then subclasses InteractiveItem and implements the behavior of a menu item. The script is then attached to any object that is to be a menu item. If you want to specifically see its behavior look at the code and comments in the script (Assets->Scripts->Menu Scripts).

The MenuEyeRaycaster (Assets->Scripts->Menu Scripts) script allows the user’s gaze and appropriate trigger presses to interact with IteractiveItem (and by extension MenuItemInteractiveItem due to subclassing). This script casts a ray in the direction of the forward direction of the object that it is attached to (the Center Camera in OVRCameraRig in our case) and checks if the “gaze” has just passed over or off an InteractiveItem, and calls the appropriate methods on the InteractiveItem, then updates variables that indicate what is being looked at. If the trigger or A is pressed while an InteractiveItem is being looked at, it calls the Click method on the respective InteractiveItem.

**InfoScene**

Overview

The player looks around from a center pedestal at graphics that indicate controls, as well as text describing controls. The same OVRCameraRig that is used in the MainMenuScene is used here as well. There is no interaction in this scene other than pressing the trigger and the middle of the touchpad on the GearVR Controller or Y on the StratusXL to return to the MainMenuScene.

**TutorialScene**

Overview