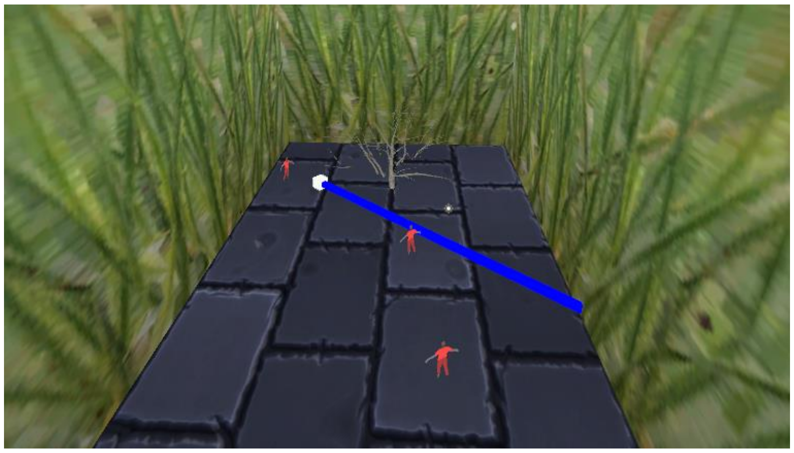


P3_DOC: ZOMBIE GAME

Team Awesome Sauce

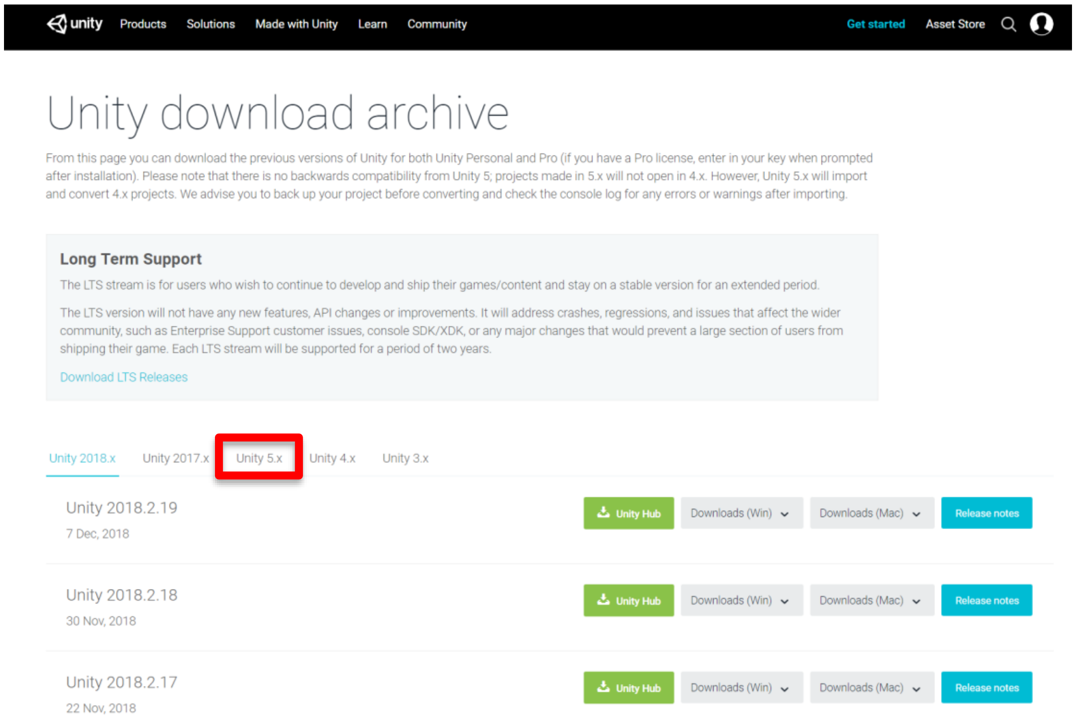
Instructions:

Our game is meant to be played with VR as shown in the presentation video, but some additions were made to make the game playable within Unity without VR. For this change, the game is played in a Top-Down view and the player controls a cube that can move, shoot, and build as zombies chase you. Teleporting was not accessible in this version due to Top-Down constraints, but teleporting is working in VR version as shown in the presentation video.

To play and test the game, you will first need to download Unity. YOU MUST DOWNLOAD VERSION 5.6.2, OR IT WILL NOT WORK. You can find it here:

https://unity3d.com/get-unity/download/archive

Choose the Unity 5.x tab



Unity download archive

From this page you can download the previous versions of Unity for both Unity Personal and Pro (if you have a Pro license, enter in your key when prompted after installation). Please note that there is no backwards compatibility from Unity 5; projects made in 5.x will not open in 4.x. However, Unity 5.x will import and convert 4.x projects. We advise you to back up your project before converting and check the console log for any errors or warnings after importing.

Long Term Support

The LTS stream is for users who wish to continue to develop and ship their games/content and stay on a stable version for an extended period.

The LTS version will not have any new features, API changes or improvements. It will address crashes, regressions, and issues that affect the wider community, such as Enterprise Support customer issues, console SDK/XDK, or any major changes that would prevent a large section of users from shipping their game. Each LTS stream will be supported for a period of two years.

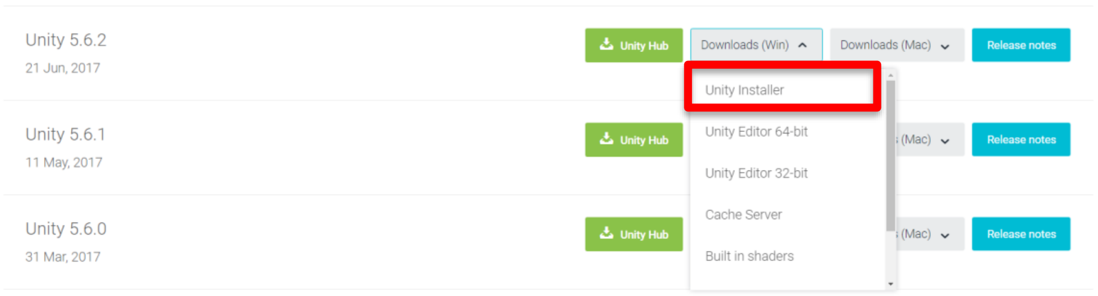
Download LTS Releases

Unity 2018.2.19
7 Dec, 2018

Unity 2018.2.18
30 Nov, 2018

Unity 2018.2.17
22 Nov, 2018

Then scroll down to Unity 5.6.2, press “Downloads (Win)” and choose the “Unity Installer”



Unity 5.6.2
21 Jun, 2017

Unity 5.6.1
11 May, 2017

Unity 5.6.0
31 Mar, 2017

After you are finished installing and setting up Unity, you may need to make an account within Unity. Next step is to get the game.

The zipped project is 444Mb which exceeds GitHub's 25Mb limit. So instead we have provided a Google Drive link to download it here:

https://drive.google.com/file/d/1Ged1eQeC8kd1_aHn_-hxVFeBCZXGgrV0/view?usp=sharing

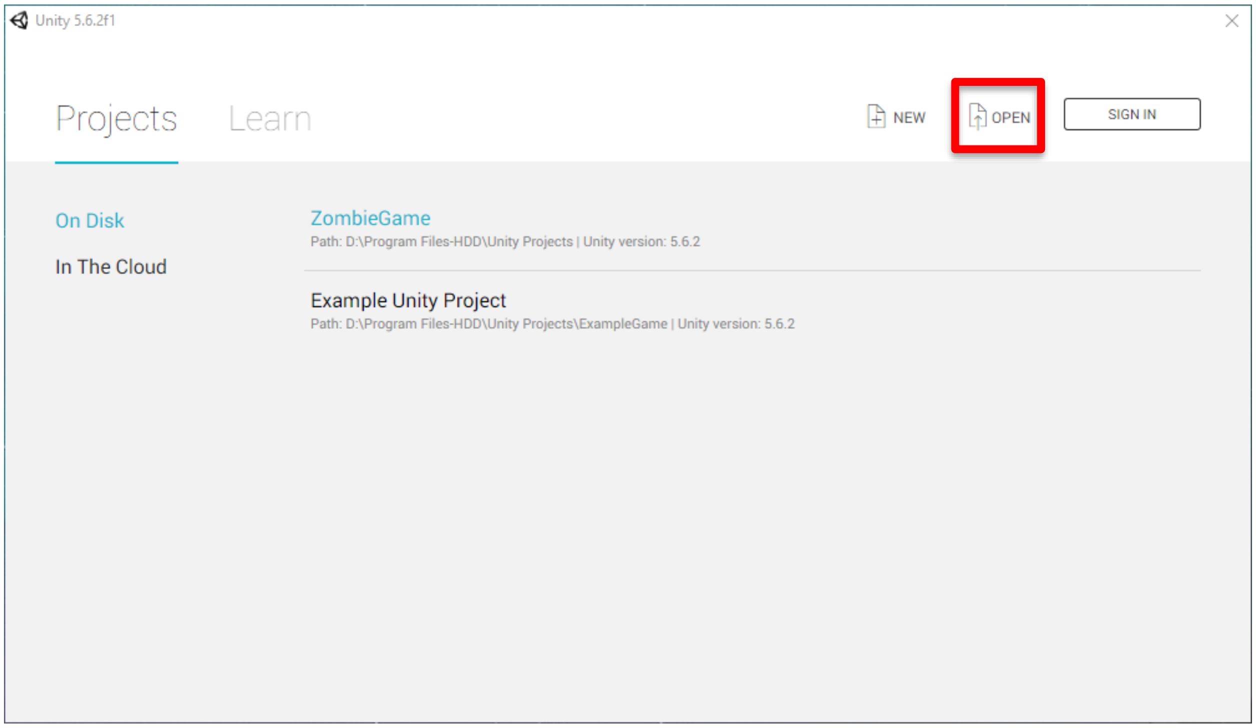
---

This should be the VR version, this is here more so teammates have access to it:

https://drive.google.com/file/d/1-a7rP4i_mBKIwM3tF1414l_QrL6HbqOG/view?usp=sharing

---

Unzip the project, and then open it with Unity.

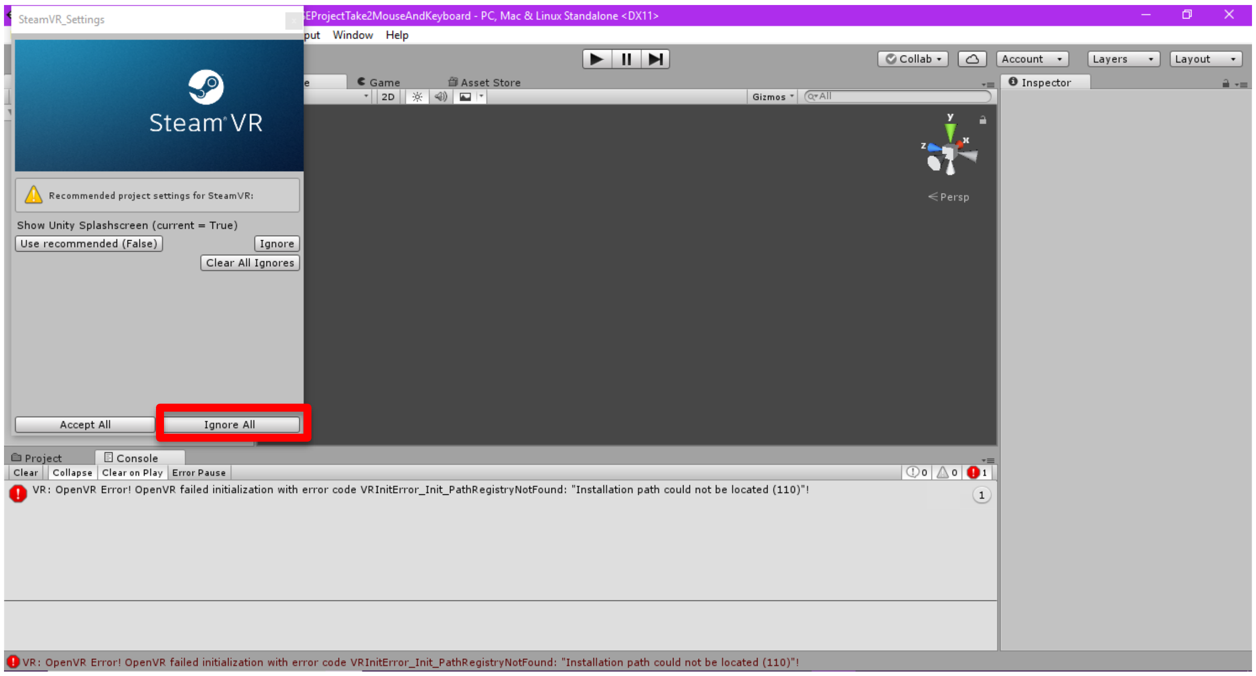


ZombieGame
Path: D:\Program Files-HDD\Unity Projects | Unity version: 5.6.2

Example Unity Project
Path: D:\Program Files-HDD\Unity Projects\ExampleGame | Unity version: 5.6.2

It will take some time to build, but once it opens you will likely see some errors in the console and a pop-up on screen, these should be related to not having a VR headset, but the game will still play in the Top-Down Mode.

Clicking “Ignore All” will ensure the pop-up doesn’t reoccur every time the game is played.

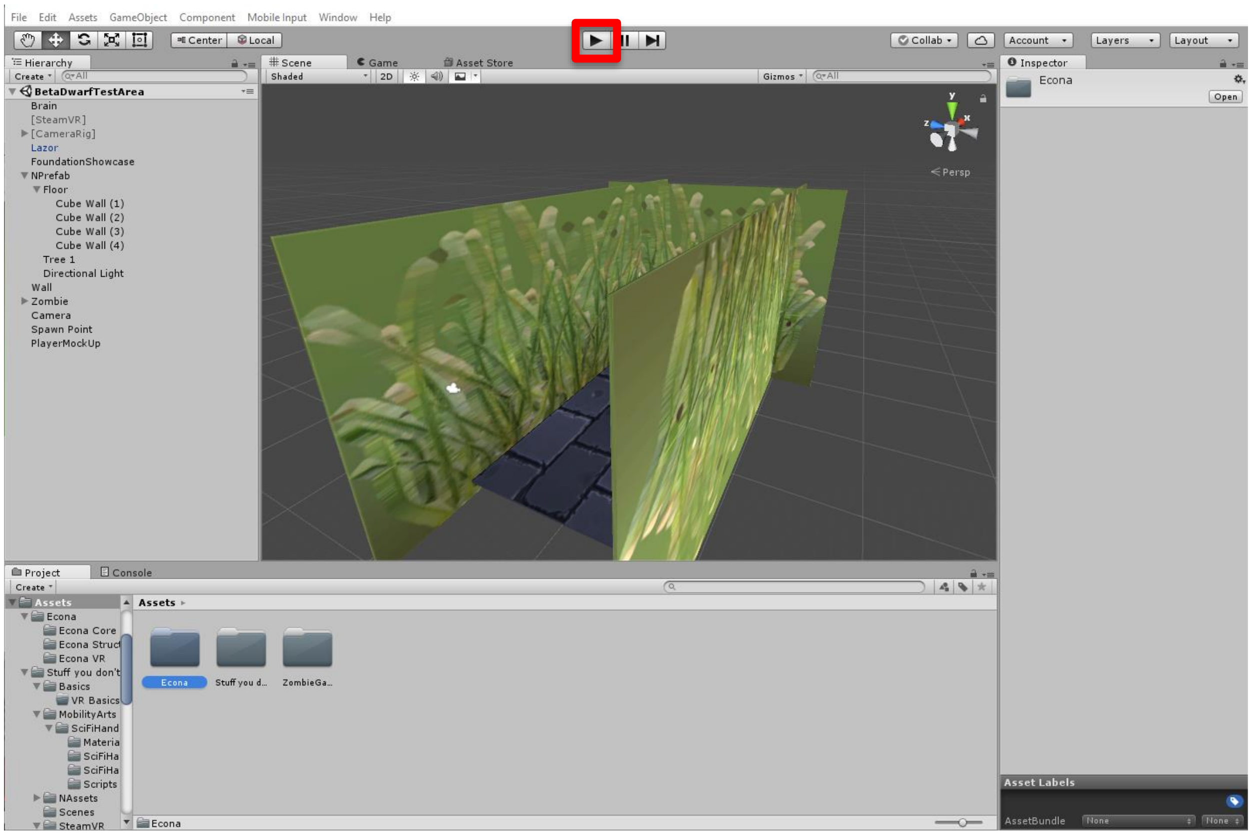


Recommended project settings for SteamVR:

Show Unity Splashscreen (current = True)

```
VR: OpenVR Error! OpenVR failed initialization with error code VRInitError_Init_PathRegistryNotFound: "Installation path could not be located (110)"!
```

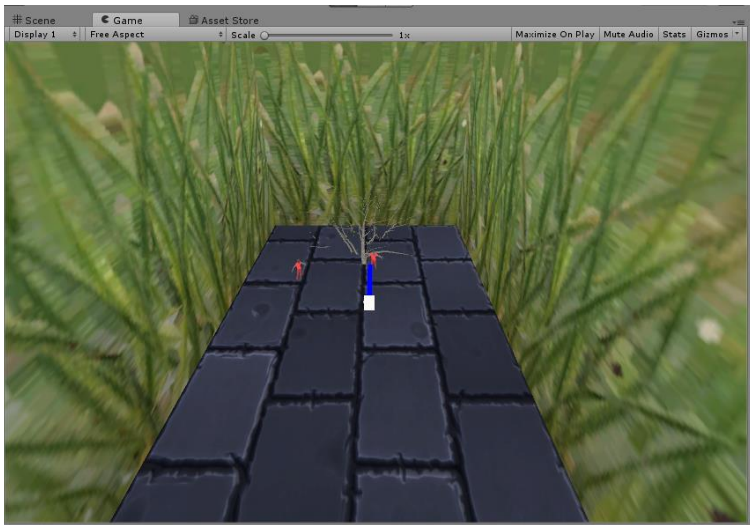
Pressing the play button will start the game and pressing it again will stop it.



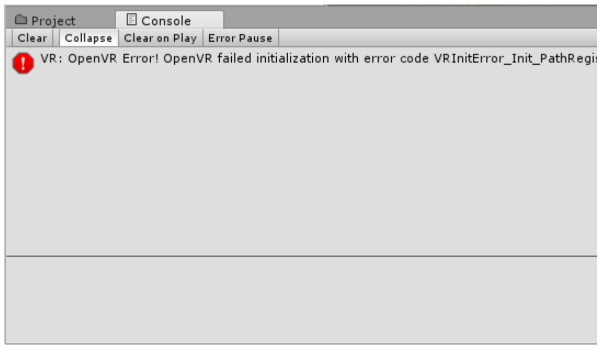
The player will be shown as a white cube on screen and have a blue laser used to guide your aim for shooting and building.

Keyboard controls:

- W – move forward
- S – move backward
- A – turn left
- D – turn right
- E – build wall
- SPACE – shoot

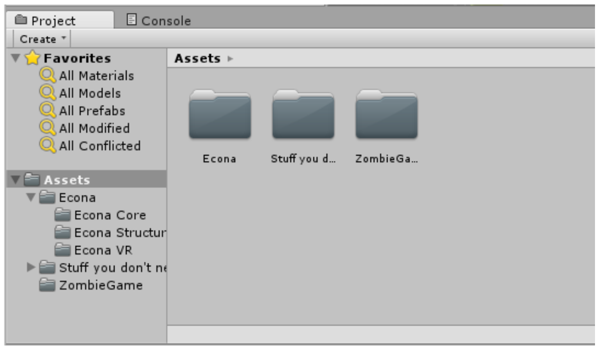


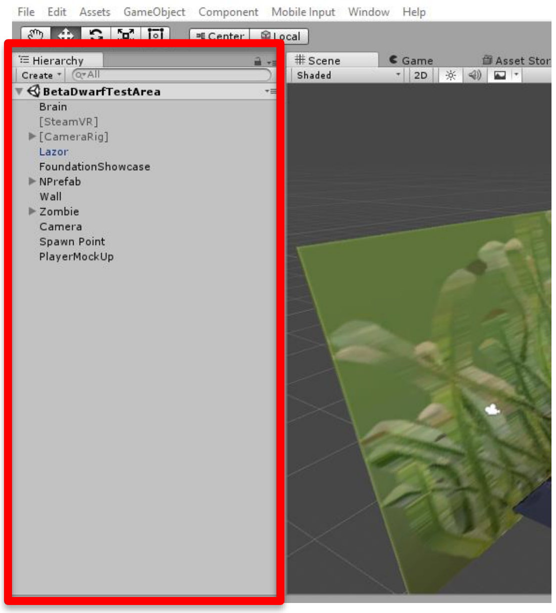
From there you are free move about the project. If you click the “Project” tab,



```
VR: OpenVR Error! OpenVR failed initialization with error code VRInitError_Init_PathRegi
```

You will be taken to a panel showing three folders





The Econa folder holds files for the Econa system. The middle folder holds files that are important to making things work, but not important for grading. This is things like art assets, textures, and SteamVR compatibility. The main portion of the project is found in the Zombie Game folder, and in the game objects listed in the hierarchy to the left. The Zombie Game folder holds the C# scripts written for the game.

Design Pattern:

On close inspection, the Econa system is similar to an advanced Visitor design pattern. While it does not use the word "visitor", the core ideas behind how it is structured mirror visitor. The Econa system is designed to create a separation between the structure of a program and the algorithms implemented by that program. In this case, it's creating the structure for how Events, Conditions, and Actions interact with each other. Rather than using the key word "visit", it uses "trigger", "checkCondition", and "runAction" for Events, Conditions, and Actions respectively.

There are detailed comments for how the Econa system works as well as a unit test for it in the Unity project, but as mentioned in the presentation, it is very conceptually difficult to understand, well past what I initially thought it would be. It is a solution very targeted at Unit tests, built to solve problems that don't exist in the same way for in other languages. While the unit test does stress the system rather well, and the cases all pass, it's not easy to read and understand the output of the test cases. This is why it is not used in the implementation of the actual game. We understand the difficulty this makes in grading but incorporating a new design pattern wouldn't be feasible at this point.

For a much more detailed look at what Econa is and how it works, follow the path in Unity

Econa -> Econa Core -> Econa.cs

Testing:

The testing done for this project primarily revolved around game testing. So rather than using a Java-style unit test class structure that wouldn't be able to properly interact with the in-game objects, we each provided a number of test cases related to our code that went through various scenarios. These scenarios covered interactions between the player, zombie, and environment, as well as the shooting, building, and teleporting aspects of the game. Our concern is that the tests seem simplistic, but these scenarios cover all aspects of the game in its current state. As new additions to the project get more complex, so will the tests that go along with them.

Mark's test cases focused on the Econa system and were separate from the other members' test cases. The rest of the team members created 10 test cases each, 5 related to their code, and 5 related to the project as a whole to test specific interactions between objects. Because of this process, there are some overlaps in scenarios and outcomes, but this ensured each member's tests would cover both simple and complex interactions. Each member's test cases can be found in the root directory of Team-Awesome-Sauce's GitHub under the files:

- MarksTestCase(s)
- Nicole's Testing Consideations
- Sihan's testing consideration
- Julian's Testing Considerations

Bugs and improvement:

Every game has bugs and room for improvement. Wall Walker and Energia, my (Mark) published games, are held together with duct tape at times because the robust path is often also a much slower path, and things just need to work. If something works 99% of the time, and the fail case is not game breaking, then it's often not a priority to fix. This is just a fact of game development, and it's a mental shift from a lot of other software development. This is not just Mark Sowders being a poor game developer, nearly every single game ever released has skeletons in its closet, there is just too much that should be done and not enough time to do it.

For example, the restartGame() function was supposed to restart the game to the starting state, but when it is called, nothing happens. But restartGame() is not normally called and does not impact the player experience, so it's not a priority to fix.

There was a tradeoff to be made with the structure of the project, and in order to make it more accessible for the team as a whole, the quality and organization suffered slightly. But in the end the game works and a player can interact in the play space as intended.