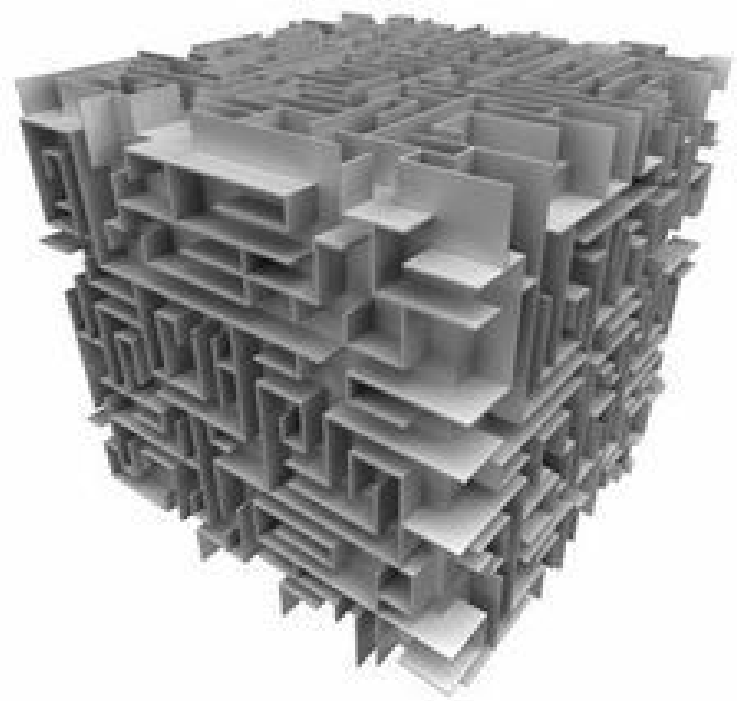
CS 30700

Software Engineering Design Document

PERSPECTIVE



TEAM 24

Peter Farmer
Vincent Jiang
Saurav Khanna
Akanksha Tripathy
Aneesh Vempaty

Table of Contents

1. Purpose 2
    1.1. Functional Requirements
    1.2 Non-Functional Requirements
2. Design Outline 5
    2.1. Monolithic Application
    2.2. Model-View Controller
    2.3. UML / State Diagram
3. Design Issues 9
    3.1. Functional Issues
    3.2. Non Functional Issues
4. Design Details 13
    4.1. Class Diagrams
    4.2. Description of Classes
    4.3. State / Sequence Diagrams
    4.4. UI Mockups

1. Purpose

Virtual reality is on the cutting edge of technology. However, we feel that there is a moderate lack of development and user experience. The virtual reality experience has not been exposed to the majority of people as it is a relatively new product. As a result, it's hard for people to truly grasp how unique virtual reality is and how vast the possibilities can be. In order to advertise and showcase how spectacular virtual reality can be, our product is to create a 3D immersive puzzle labyrinth that contains both aspects of reality and phenomena that's only accessible through virtual reality.

Surely there are a myriad of virtual reality games already available to download and play to this day such as Vox Machinae and 3D Minecraft. However, we feel that these games are just incorporating virtual reality as an additional feature instead of trying to explore what could be possible through virtual reality. It seems that with Minecraft and Vox Machinae, they are making similar games to what is available today and just slapping on a virtual reality headset to create a "new" experience. However, our goal is to make the user feel like they can actually change the world they are in by making the user able to manipulate the gravity and physics. By doing so, we will be actually creating an altered sense of reality.

In the game, the player will be placed in a testing simulation. The goal of the game is for the player to get to the center of the maze. However, the trick (the altered sense of reality) is that in order for players to find their way through, they will be able to explore by shifting gravity to walk on walls or ceilings. Through a series of tasks and challenges, the player will slowly get closer and closer to the center. By following through with this design, we believe that this game can represent the ideals of being close to reality with functions that are only possible through virtual reality.

1.1 Functional Requirements

<u>As a gamer</u>

1. I would like to have a progress bar reporting how I play.
2. I would like to have a conclusion screen when finishing the game.

3. I would like to see a functional navigation system while playing.
4. I would like to see a visible end to the game.
5. I would like to reopen my account and start where I left off from.
6. I would like to be able to restart the game.
7. I would like to see a functional website containing information about the game.
8. I would like to see great graphics.
9. I would like to hear sound effects while playing the game.
10. I would like to see multiplayer functionality. (if time allows)
11. I would like to have the game go up in difficulty as I play
12. I would like a functional title screen.
13. I would like to have a save/load game option.
14. I would like to make an account.
15. I would like to have the ability to create multiple accounts.
16. I would like to have a functional pause menu.
17. I would like to have functional interactions with the environment.
18. I would like functional player movement.
19. I would like to be able to restart the game.
20. I would like a timer to keep track of how I'm playing.
21. I would like to see puzzles that require connections between field of view and the environment.

<u>As a game developer</u>

1. I would like to implement stunning visuals
2. I would like to incorporate regular updates on the game to make it cleaner and faster.

<u>As a virtual reality developer</u>

1. I would like to see the environment be able to affect the user.

<u>As a virtual reality enthusiast</u>

1. I would like the display to be through a virtual reality headset.

<u>As a puzzle enthusiast</u>

1. I would like to see puzzles that require connections between field of view and the environment.

1.2 Non-Functional Requirements

1. Be able to download off Steam (as Mac/Windows/Linux computer game).
2. The interface must be user-friendly and easy to navigate.
3. Strong connection between front-end development and backend development.
4. Strong connection between the device running the game and the device hosting virtual reality.
5. Security in the sense that users cannot edit their high score.
6. Security in the sense that users cannot cheat their way through the game.

2. Design Outline

Our project is a desktop/VR game that we hope will provide a source of entertainment for gamers. We will incorporate a virtual reality headset in our game and hope to give our users a new experience. Our implementation will not use a client-server model but a monolithic application. This is because the game will not be implementing a server as it is only single player. Despite not using a server, we will still be implementing a Model-View-Controller [MVC] as a software architectural pattern to display the user-interface and game. We will be developing the game with the help of the Unity 5 game engine, an industry standard development environment to make the game run smoother. The main architecture will just be the desktop CPU to run the game but we will also incorporate a virtual reality platform for people to view the game.

2.1 Monolithic application

Virtual Reality Viewer

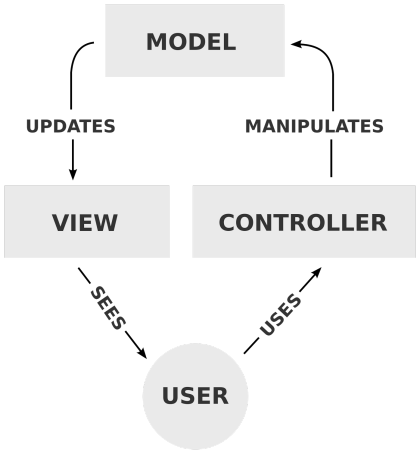
- The main view that we will want our users to experience will be through a virtual reality headset such as the industry standard Oculus Rift or HTC Vive.
- This main viewing platform will be the main interface of our system.
- The virtual reality headset will be connected to the game environment in order for the controller to seamlessly receive viewing changes.
- The data from the model will also have to be consistently rendered to have the optimal viewing experience for the gamer.

The Desktop Module

- The Desktop Module will have to be able to respond to certain requests from the user.

- This will be the main processor to handle all the information requests with the controller and update the model smoothly.

2.2 Model-View-Controller



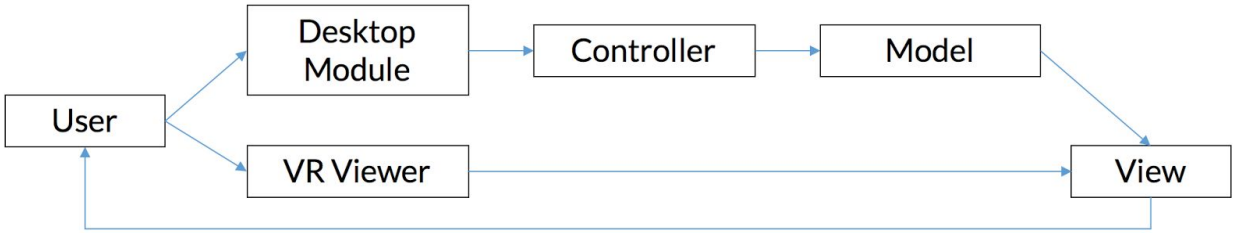
The figure above demonstrates our high-level overview of the system with four main parts:

1. USER - The User requests information to the controller through mouse clicks, movements, viewing angles, or action buttons.
2. CONTROLLER - The controller processes all this information by converting and appropriating the events into its suitable actions. For example, if a player were to press a button , an “open-door” action may be appropriate, or maybe if a user walks into a wall, a “blocking” action would be applied so that the user does not go through the wall.
3. MODEL - The model gets updated by the controller. In the previous examples, the walls/obstacles modifiers would get manipulated.
4. VIEW - This comes to the actual rendering and what the player sees as the effect of his actions. The game would be updated to it’s most current state.

Below is an image of the overarching high-level model for both the model-view-controller and the monolithic application. The user interacts with the

desktop module which then accesses the controller. The controller manipulates the model which is updated the view that the user can see. The user sees the view via the VR viewer in which the user can decide whether to do another action. The user can use both the desktop module and the VR Viewer to change the view.

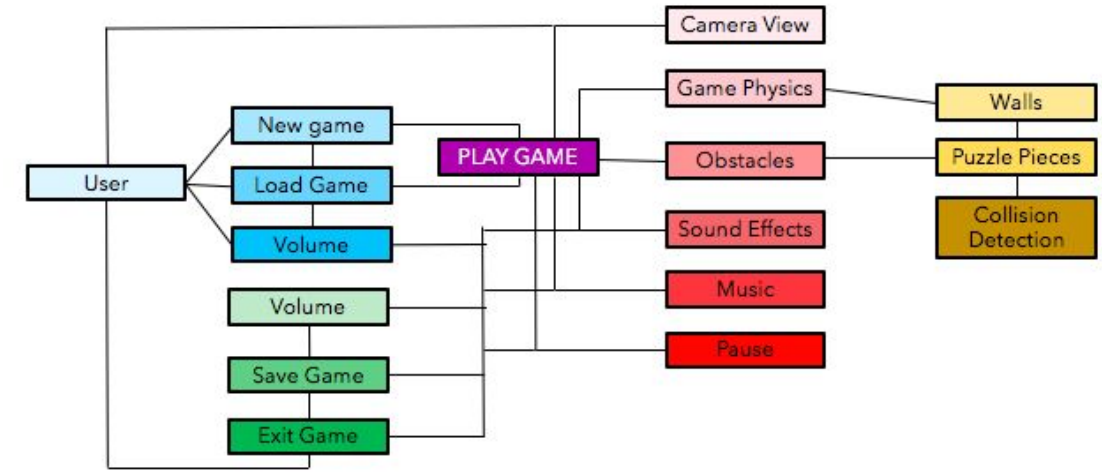
High-Level Overview



2.3 UML / State Diagram

Since our implementation uses a monolithic application, our UML more closely discusses the certain parts and interactions that the user will be able to utilize or experience while playing our game. As all the experiences and interactions are mostly in one level, the state diagram is not as uniform/formulaic as client-server diagrams. Below, the UML Diagram is explained in more detail.

UML Diagram



Blue Panels

The blue panels represent the game's main menu display selections. This is the primary navigation page and gives the user the option to create a new game, load a previous game or simply adjust the sound volume settings in the game through another sub-menu.

Purple Panel

The main place where the user will be immersed in the environment. This is where the majority of actions will be taken place and the model manipulated.

Red Panels

This is the primary interactions and actions an user can do. The camera view which is primarily controlled by the user, will respond to the virtual reality headset movement. The movement data is collected from the gyroscope, accelerometer, and the WASD movement controls. The game physics will be changed as users can manipulate their gravity so they will be able to walk on the walls and ceilings of hallways. This will create a new layer of complexity and excitement to the game. Obstacles can range from anything that the user can touch or stand on to the objects and puzzle pieces that the user will have to interact with to complete the level. Sound effects and music will create a needed abstraction to the game to provide suspense and excitement in otherwise dull moments. The sound effects will help the game to be completely immersive.

Yellow Panels

This is a subsection within obstacles to indicate more fundamental aspects of the different obstacles that the player will have to interact with. An example of this is the wall in which the player will be able to manipulate gravity to walk on.

Green Panels

Secondary in-game menu in which the player can adjust the sound, save his/her current game, and exit the game to the title menu. This menu can be accessed at anytime within the game.

3. Design Issues

In this section we'll be discussing the functional and non-functional issues that we've discussed during the planning process of the game. Most of these issues were decision trees that we ran into because it wasn't ideal to implement all options. These issues ultimately made a difference in the type of game that we are producing.

3.1 Functional Issues:

1. Do users need to login to use our game?
   Option A: Create a username and password that's only unique to our service
   Option B: Let users login through their Facebook, Google, Steam, etc
   Option C: No login required

   Decision: **Option B** seemed to fit best because even though the game doesn't have any multiplayer functionality as of right now, users can still share and post their high score via Steam to compete with other friends and users. The concept of logging will be even more essential if there is time to incorporate multiplayer functionality.

2. How hard should the game be to complete?
   Option A: For initial stages, make all walls accessible to the user, then as time passes, make it harder to recognize the path by taking away accesses
   Option B: Make it hard from the start and stick with the idea of making it a hard VR puzzle game from the get-go
   Option C: Make all walls accessible with slight tricks and secrets in various places so that it's more of a game just for the experience of VR puzzle gaming

   Decision: **Option A** we felt was the best choice to implement because it would utilize the concept of appealing to both puzzle enthusiasts and VR enthusiasts. We would have the ability to show hard a puzzle game can be and show the fun experiences of VR puzzle gaming by making it gradually harder for the user so that as they adapt more to the game, they can experience hard challenges as well that would be give more incentive to play and learn more. Most games follow the concept of becoming progressively harder so it makes sense for our game to follow that idea as well.

3. Should we incorporate multiple levels?
   Option A: Incorporating multiple levels would give users more challenges to complete and give the game more depth, but due to lack of time, it would be hard to come up with unique puzzles and might take away from productivity in other areas.
   Option B: Creating one giant level would give time for more development in other areas like environment and setting, but there would only be one level to play, which is not enough gameplay for common users. However, this level will be a very long level as it will contain many sub-rooms that gamers will have to unlock.

   Decision: **Option B** was more optimal in this decision because we feel that the showcase one long level with many different environments would be a better than having multiple tiny levels that would not be as well done or up to par. Having one long level would also allow us time to work on other elements of the game as well as the environment and visuals.

4. How should we portray changes in gravity?
   Option A: Add an animation when the user wants to change gravity so that they get some time to readjust themselves to the new setting.
   Option B: Let the user change gravity themselves to simulate themselves walking up onto the wall. *I.e.* The more they moved forward, the more so they would be positioned on that specific wall.

   Decision: **Option A** since it was more ideal in this situation. The difference between Option A and Option B is pretty miniscule because the difference is letting users step up onto the wall versus interacting with the wall and triggering an animation which would automatically step up the wall/ceiling that the user chose. Option A made more sense since it gives users an easier time to adjust to their new environment. Option B which could cause major confusion and lack of balance if users aren't adjusted to the game at first.

5. Should we allow users to shift gravity whenever they want?
   Option A: Yes, so users have control over whether or not they want to change the game physics.
   Option B: No, meaning that the user has to interact with certain environmental tiles in order to have a shift in gravity. *I.e* Having a player step on a upside down gravity tile to make the player be able to stand on the ceiling.

   Decision: **Option A** was the better choice because it allows users have more freedom within the game, which would make for a more enjoyable experience.

Having the user be able to change physics mechanics within the puzzle would give the users an experience only possible within virtual reality and allow for a new experience.

Decision: **Option A** was the better choice because letting users control whether they would like to shift gravity or not would allow for us to focus on other aspects of the game. For example, if we were to have stuck with Option B, it would mean that when gravity shifts, the entire environment would shift as well, including objects and other items that are within the surrounding. This option would be too hard to incorporate

3.2 Non-Functional Issues:

1. Which platform will we use to process the game?
   Option A: Mobile, such as samsung gear VR
   Option B: Desktop Platform

   Decision: **Option B** was the better choice because not only does the desktop have a better processing unit but also it would be able to reach a much wider range of audiences. The desktop would be able to calculate the big changes in the game environment and physics where as a mobile phone would probably not be able to handle all the manipulations to the model.

2. Which game engine should be used to implement the backend?
   Option A: CryEngine
   Option B: Unreal Engine 4
   Option C: Unity 5

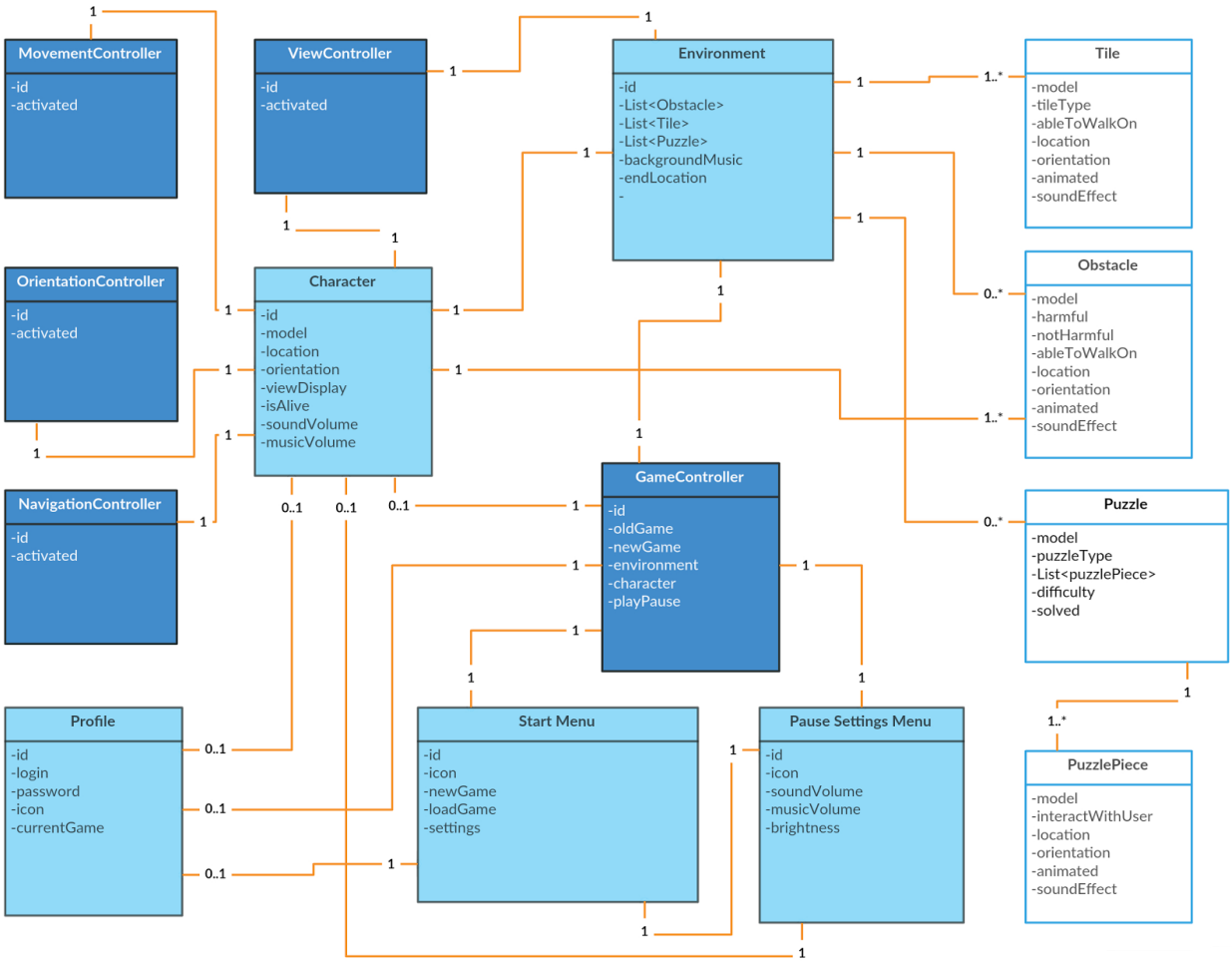
   Decision: **Option C** was the better choice because of our team members familiarity and experience with Unity 5. Unity 5 also has a lot of packages that support virtual reality games so it would come in handy for creating our game. Also picking something that our teammates have heard of and are familiar with saves time in learn what Unity 5 is.

3. Which virtual reality headset are we going to use?
   Option A: HTC Vive
   Option B: Oculus Rift
   Option C: Razer OSVR HDK 2

Decision: **Option B** because some of our team members have some familiarity using the Oculus Rift and the product is more widely recognized. The Oculus Rift has been available to consumers far before the HTC Vive and Razer OSVR HDK 2, hence it has been used more for virtual reality games and could be considered an industry standard for virtual reality. Plus, it can easily be paired with Steam games and PC games, making it a viable headset to use if we're going to post or our game on Steam.

4.Design Details

4.1 Class Diagram:



4.2 Description of Classes:

1. Users

- Profile
    - Represents each individual user who has an account for the game.
    - It contains a user's unique ID, login and password pair, personal avatar or icon, and a save file with the information about their most recent in-game save point.
    - Allows different users to log in and keep a record of their games.

- Character
    - A 3D model used by the user to interact with the game elements.
    - Contains the current orientation of user so that the game is able to tell whether the user is rightside-up, sideways, or upside-down and can render accordingly.
    - Has a unique <x, y, z> location to track the movement of the user.
    - Tracks when the character is hurt by obstacles such as fire so the game knows when to end and restart.

2. Menus
- Start Menu
    - Passes the proper argument to Game Controller which determines which game to instance based on whether the user chose New Game or Load Game.
    - There is an option to quit the game when they are finished playing.
    - The user can access the Pause Settings Menu. This is where the user can adjust different settings such as background music volume and sound effect volume.

- Pause Settings Menu
    - Pauses the game so the user can adjust volume settings, take a break, or quit the game entirely.
    - Can be accessed at any time when the player is in the environment.

3. Controllers
- Game Controller
    - Will create the game based on input from the Start Menu. The files for a saved game will be retrieved from the current user's profile.
    - Controls all aspects related to the flow of the game and will manage interactions between the character and the environment.
    - Allows the user the access to the Pause Settings Menu whenever they want.

- Movement Controller
    - Has a permanent ID that can be accessed by the Game Controller through the Character.
    - Takes keystroke input from the Desktop device running the game to update the Character location with movement animations.
    - Can be deactivated in the case of a pause or return to main menu call to prevent unnecessary or unintentional movement.

- Orientation Controller
    - Has a permanent ID that can be accessed by the Game Controller through the Character.
    - Takes keystroke input from the Desktop device running the game to update the Character orientation by shifting the camera to the respective orientation.
    - This will be the main way for the user to shift gravity by reorienting the Character model so the user experiences a “new” up and down direction.
    - Can be deactivated in the case of a pause or return to main menu call to prevent unnecessary or unintentional movement.

- Navigation Controller
    - Has a permanent ID that can be accessed by the Game Controller through the Character.
    - Based on the Character location, the navigation system will update to indicate the correct direction for the user to move to reach the end location of the Environment.
    - Since this game will incorporate a 3D Maze, many users will get lost without a guide. This will provide an “Ariadne’s Thread” for all users.

- View Controller
    - Has a permanent ID that can be accessed by the Game Controller through the Character.
    - Takes movement input from the VR device the user is using to play the game and moves the player camera to view different aspects of the Environment.
    - Can be deactivated in the case of a pause or return to main menu call to prevent unnecessary or unintentional movement.

4. Game Models
- Environment
    - Contains the list of neutral tiles, obstacles, and puzzles that are within the game.
    - Will play the background music to provide another abstraction for the user to enjoy.

- Tile
    - All the neutral tiles that will be within the game. There will be many different tile models that will be paired with a tileType to help organize our assets.
    - Some tiles will be able to be walked on while others will be purely aesthetic. An example will be a water tile cannot be walked on while a wooden block tile can.
    - Each tile will have a location and orientation for instancing and re-instancing the map.
    - Some tiles will be animated to provide a less standstill game experience.

- Obstacle
    - All the obstacles that will be within the game. Each obstacle will have a different name and model to prevent the environment from being perpetual and stagnant.
    - Each obstacle will have a location and orientation for instancing and re-instancing the map.
    - Obstacles can hurt the user if they interact which will force the user to find a different path from what they originally intended. This will provide a longer gaming experience for the user.
    - Some obstacles will be animated to provide a less standstill game experience. Others will have sound effects to provide a more realistic experience. For example, a fire obstacle will have sound effects as well as animations to deter the user from interacting with it.

- Puzzle
    - There will be a variety of puzzles within the game to challenge the user's wits and abilities.
    - Each puzzle will have a list of Puzzle Pieces which are the individual components of the puzzle. The player will have to interact with each Puzzle Piece in order to solve the entire puzzle.
    - Every puzzle will be labelled with a difficulty level to help us maintain the game's simplicity vs complexity balance.

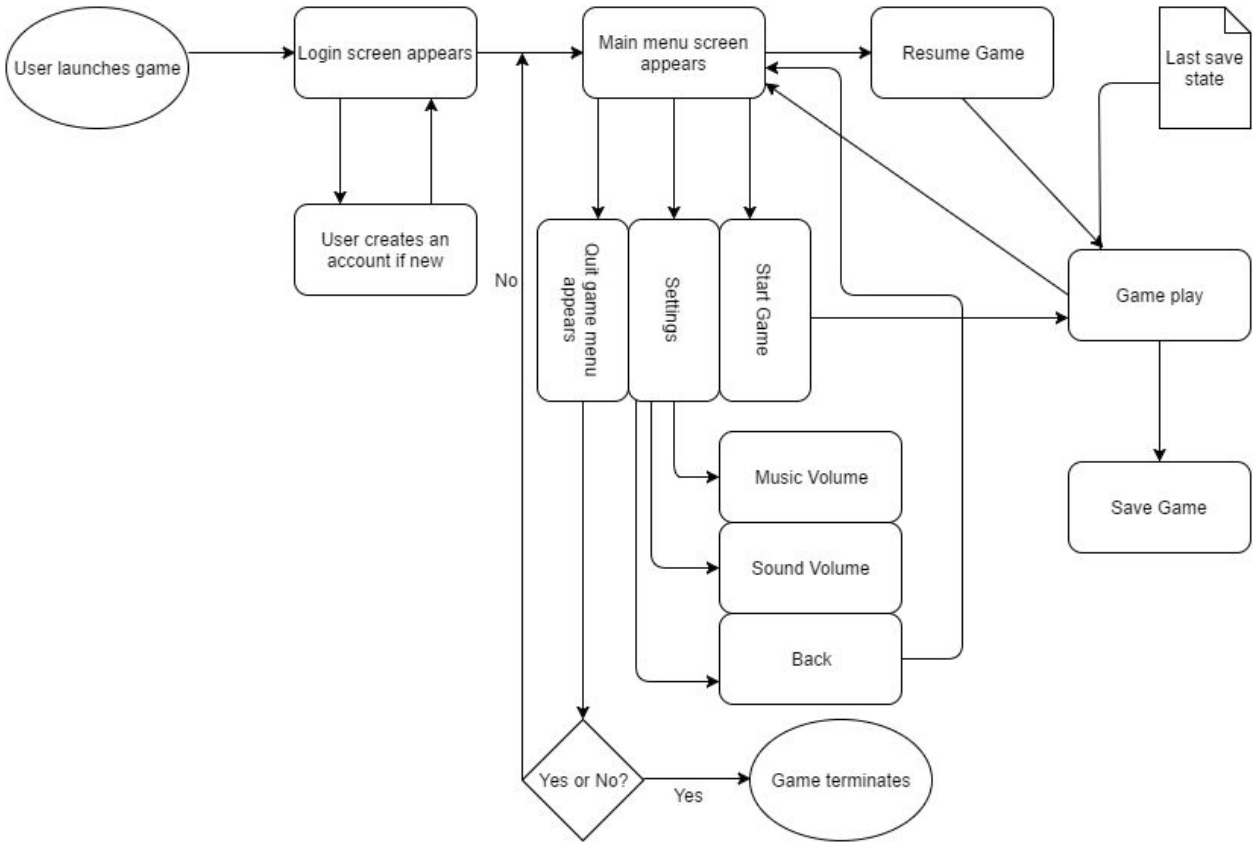
- Puzzle Pieces
    - Puzzles will be comprised of several Puzzle Pieces.
    - Each piece will have a different model to clarify what the puzzle is asking for.
    - The user will be able to interact with some of the Puzzle Pieces to solve the puzzle. For example, we might have a sliding picture puzzle which will be

made up of many sliding Puzzle Pieces, which can be reorientated to recreate the picture and solve the puzzle.

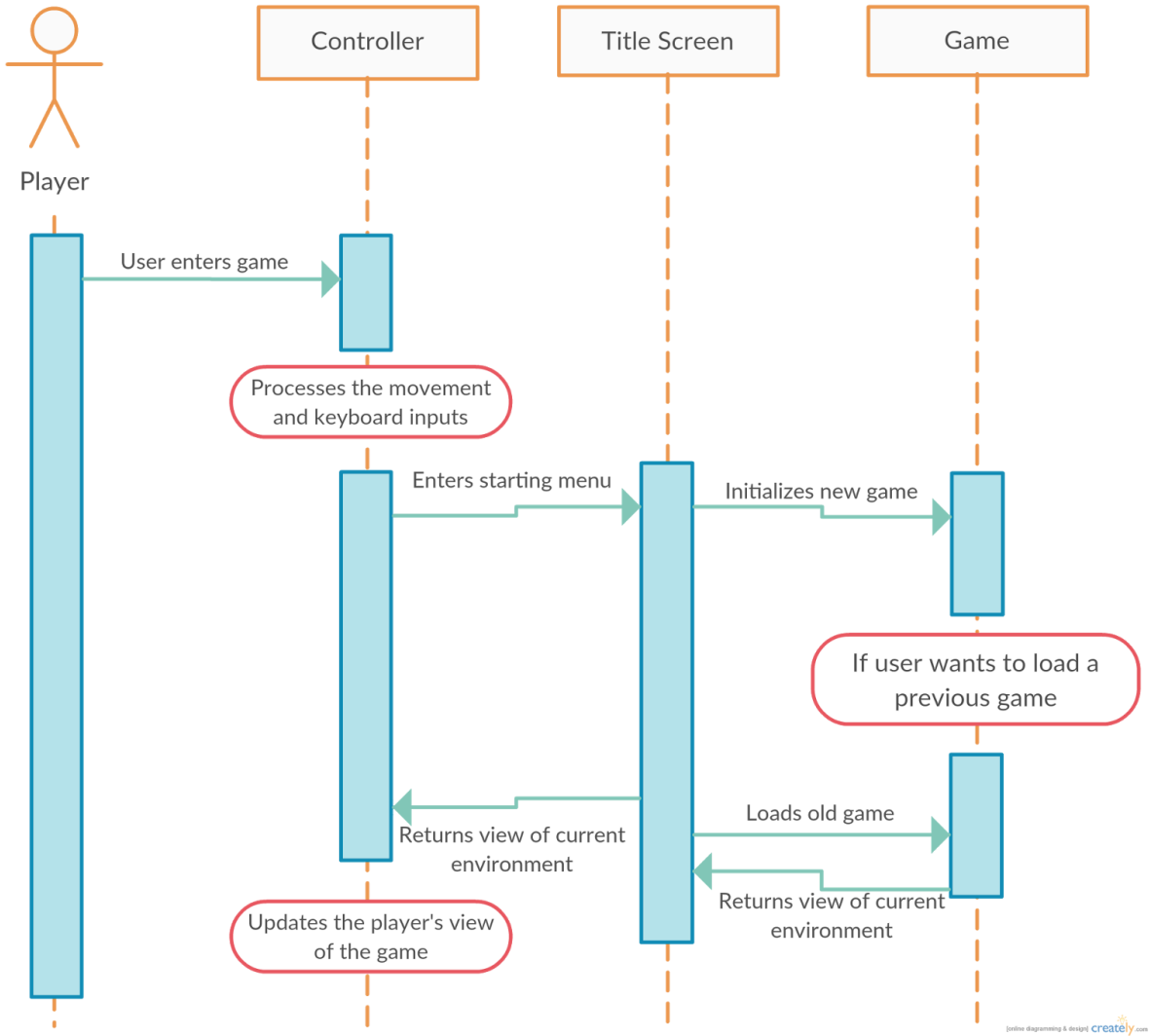
- Each obstacle will have a location and orientation for instancing and reinstancing the map.
- Some pieces will also be animated if it will help the user solve the puzzle.

4.3 State / Sequence Diagram

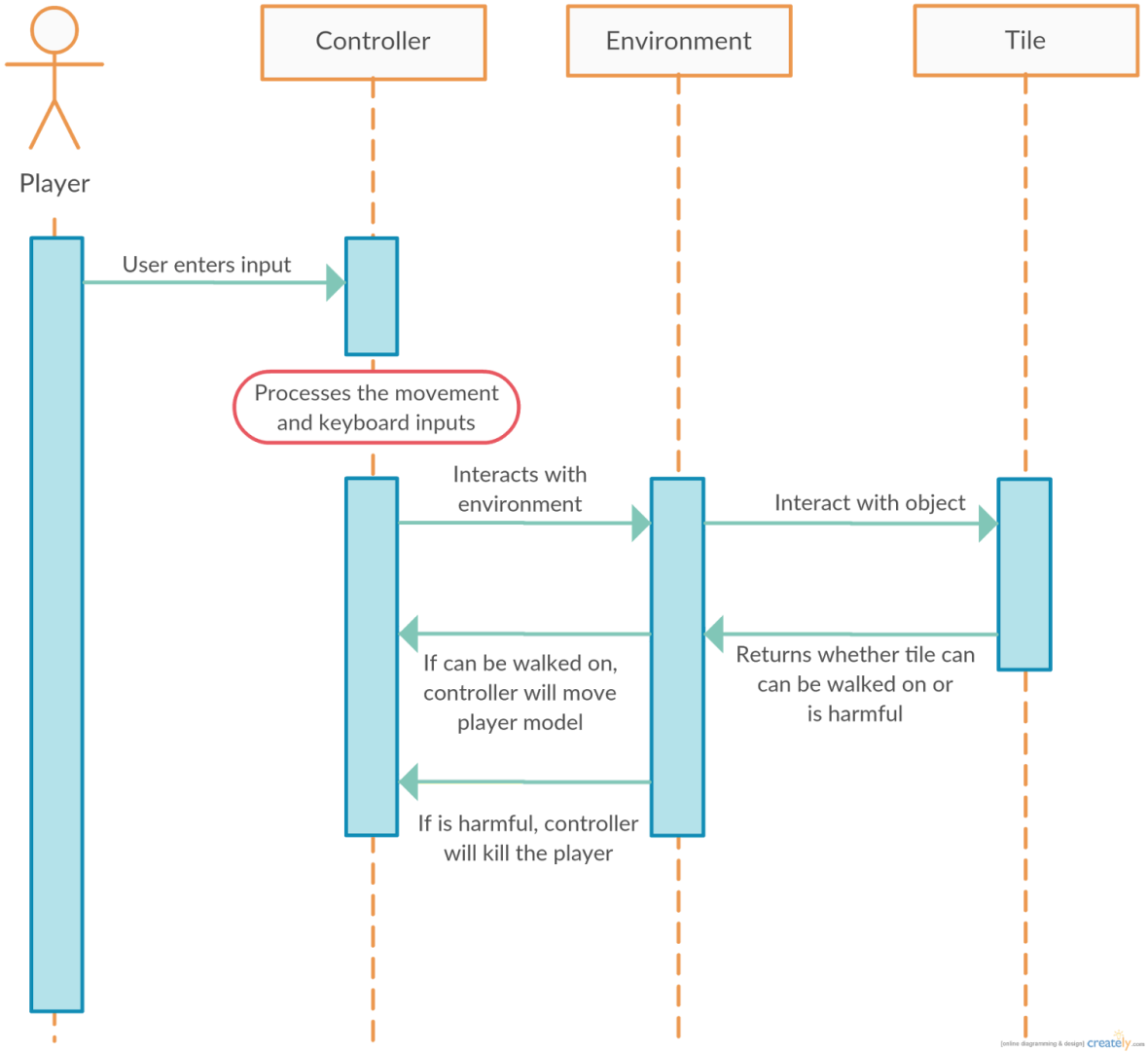
State Diagram



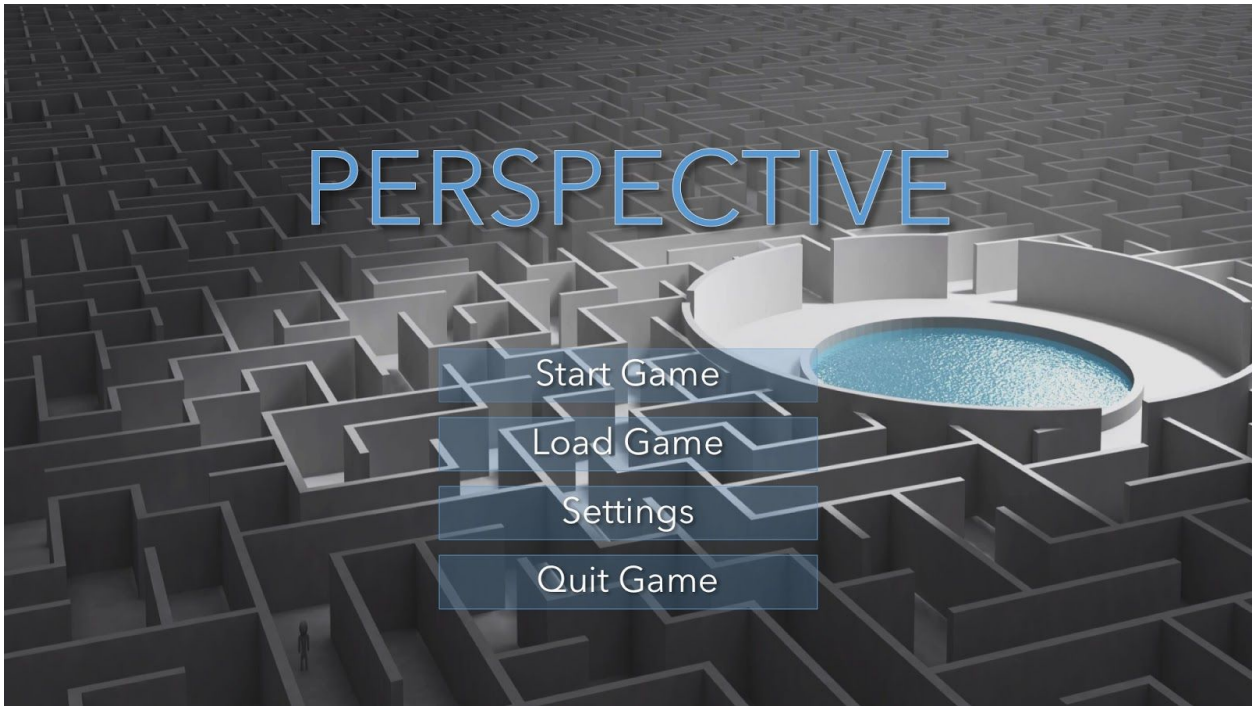
Sequence of Events When Starting Game



Sequence of Events Upon Interacting with Environment

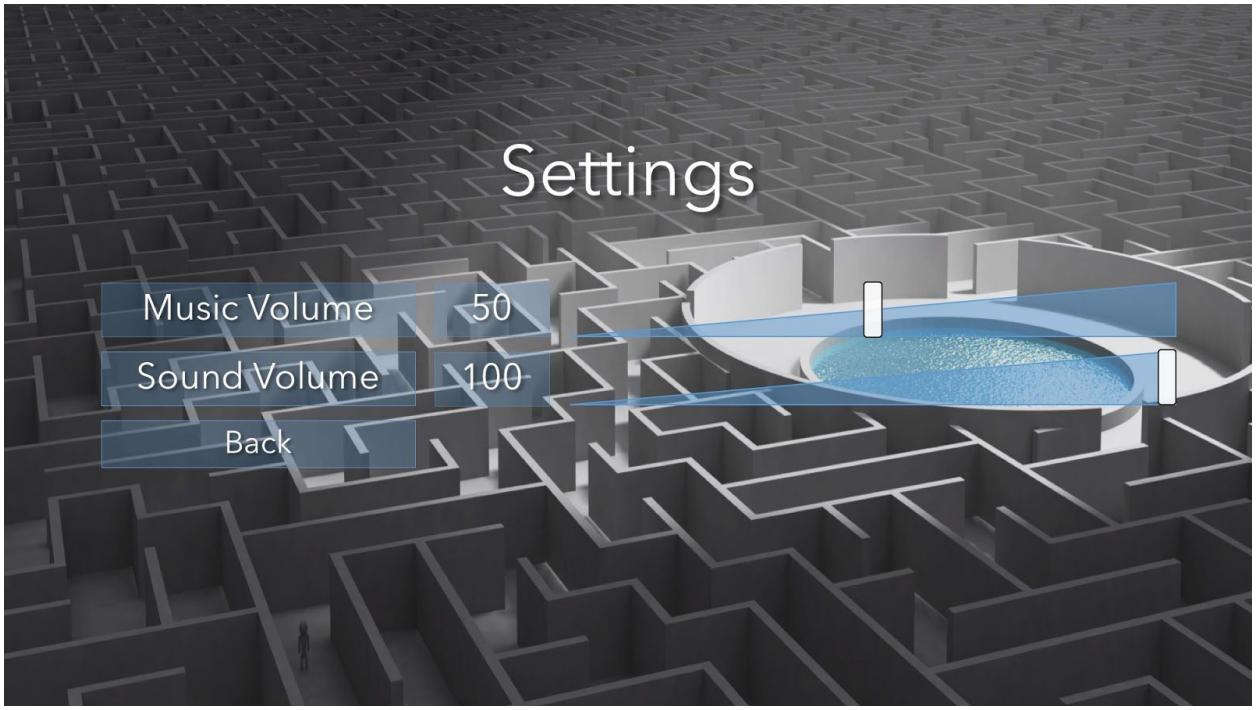


4.4 UI Mock-Ups



Starting Menu Mockup

This is an example of our start menu. Here the user has a variety of options to choose from. He/she could start a new game, load from a previous saved game, adjust settings, or quit the game. This will be the first page that the user will come into contact with and is the main source of menu navigation.

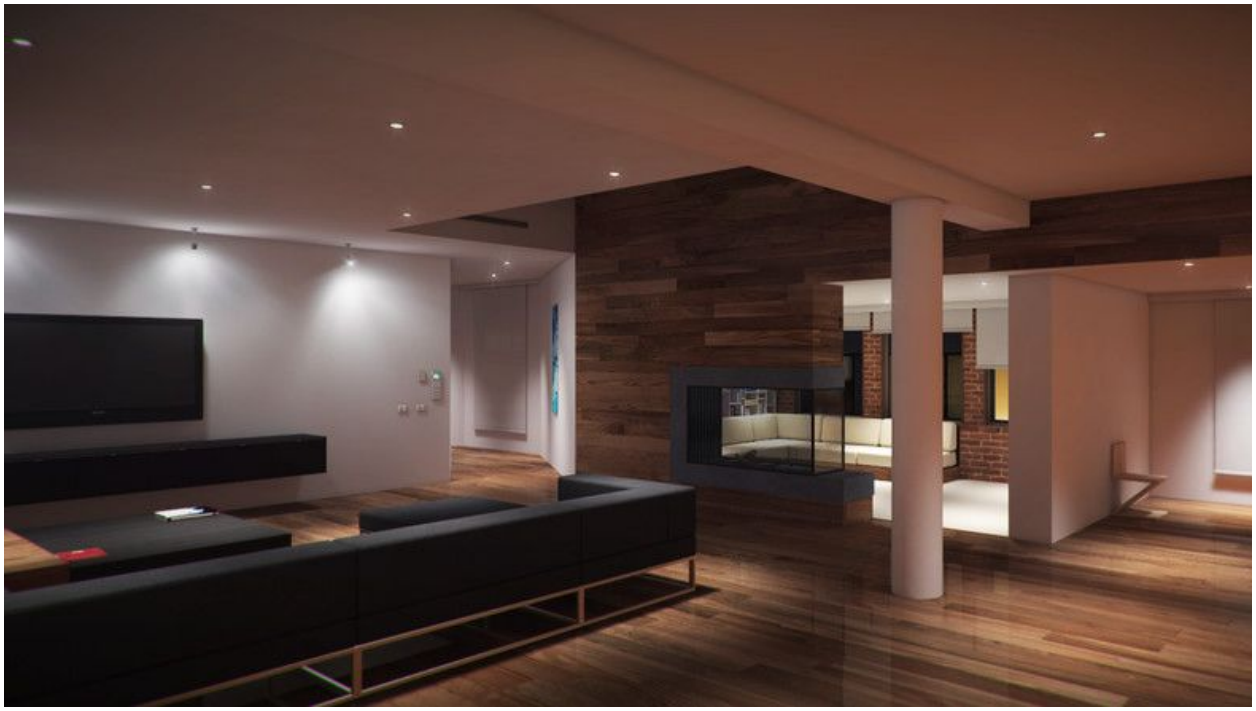


Settings Menu Mockup

This is an example of the settings submenu. This simple menu will allow the user to adjust music and sound effect volume or go back to the main menu.

Picture From ArchVizPRO Interior Assets
Unity Asset Store (www.assetstore.unity3d.com/en/)

This could be an example of the assets that we could get from the Unity asset store. There are many different assets on the website that we can pick and choose from so that we can make a realistic level design. We would like to make the game look as realistic as possible so the player will be completely immersed into his environment. The combination of amazing graphics and great gameplay will truly make this game a completely immersive experience. Below is another image from the Unity asset store.



Unity Asset Store (https://www.assetstore.unity3d.com/en/#!/content/38407)

Navigation System Mockup

This is a sample of a navigation tool that the player will be able to use in the game to help navigate through the level. This will be a useful tool for the player to know his current orientation and help the player make smarter decisions.