Legend of the Boilermaker

CS 307, Fall 2019

Team 29

Design Document

Team Members:

- Christian Joseph Bortolotti
- Christopher Sean Connelly
- Haoran Wang
- Ji Soo Cha
- Seunghyum Lee
- Xingyu Wang

Project Coordinator (Manager):

- Siddarth Dhar

Index

Purpose

The video game industry is one of the world's largest. Video games as a medium offer a unique platform for creating immersive and relatable environments. It is also especially exciting to see objects or environments that we are familiar with appear in the game. Therefore, we aim to create an exciting and enjoyable gaming experience with story and background elements pertaining to Purdue. Our game will be appealing to students who attend Purdue University.

Functional Requirements:

1. As a player, the game will teach itself to me without the use of a 'game guide' or tutorial.
2. As a player, I would like for the game to be composed of levels from multiple genres to keep the game interesting.
3. As a player, I would like the different levels to all have a score that accumulate into a total high score.
4. As a player [if time allows], I would like the finished score to be uploaded to a served hosted with Heroku.
5. As a player [if time allows], I would like the scores to be verified before being uploaded.
6. As a player, I would like the different levels to share the same playable character.
7. As a player [if time allows], I would like for the playable character to be customizable.
8. As a player [if time allows], I would like for the playable character to be able to collect items that can then be shown on the character model.
9. As a player [if time allows], I would like for the items that the playable character collects to be visible from the server.
10. As a player, I would like the first level to be simple and introductory. Specifically, a platformer that can introduce me to the score system, powerups, and general controls.

11. As a player, the first level should be a 2D view of university hall, to be in a familiar Purdue setting
12. As a player, it should involve scripted background elements like changing traffic signals, and NPCs that follow world conditions
13. As a player, [If time allows], NPCs could interact with the player, like cops trying arrest a jaywalking player, to introduce challenge and interactivity
14. As a player, the second level should introduce a seperate game genre, in order to keep the game interesting
15. As a player, this second level should also expand upon and use concepts from the first level. The player character should be the same, and the points system should remain.
16. As a player, the last level should wrap up the game's progression and be the hardest level of the three, to finish the game strong.
17. As a player, the points system should be shown at the end of the game, with a breakdown of what level points were earned in.

Non-Functional Requirements:

- **Platform Compatibility:**
  Compatible to PC, Mac and Linux operating system.
  In this project we are using Unity as the game engine. Unity provides the capability that developers only need to build the game once, Unity will deploy it to many different platforms. Those platforms include PC, mobiles, VR and consoles. Our project will run on PC. If time allows, we may deploy the project to other platforms.

- **Smooth Gaming Experience:**
  Our project will not have demanding graphics and effect. Our project is aiming to have basic physical behaviour, we will use Unity's built-in physics engine which allows us to create object that behave in a realistic way. By controlling the physics from C# scripts we will make objects interactions more natural and smoothly. We want to make sure it can be run at decent frame rate on

older machines. Also, the in game effect will not cause player to have any motion sickness.

- **Game Storage and Version Control:**
  Our executable game file will be constrained to under 2GB.
  This project will use Unity's built-in Cloud Storage. We will delete redundant assets and models to keep the project not too large. Also, we will store our C# script in Github. Unity provides a version control systems also. Collaborate continuously monitors changes that are made by each team member. Team member can view the changes and revert changes or publish changes.

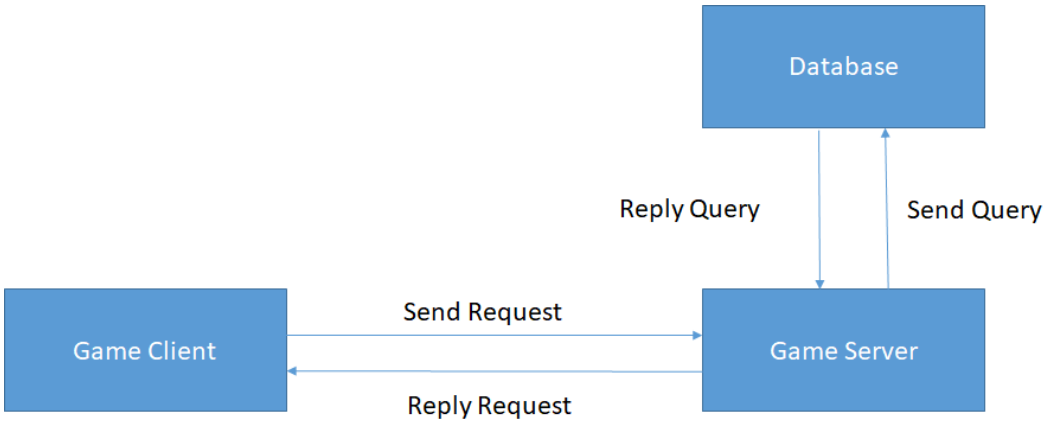
- **Security:**
  If time allows, we will implement a multiplayer version
  This project is implemented on Unity. Games that created by Unity is Sandboxed. That means game should not pose a threat to player's system. Also the game we are currently working on is an offline game. We may create a multiplayer game if time allows. In that case, we will use Unity's Webplayer to protect users.

Design Outline

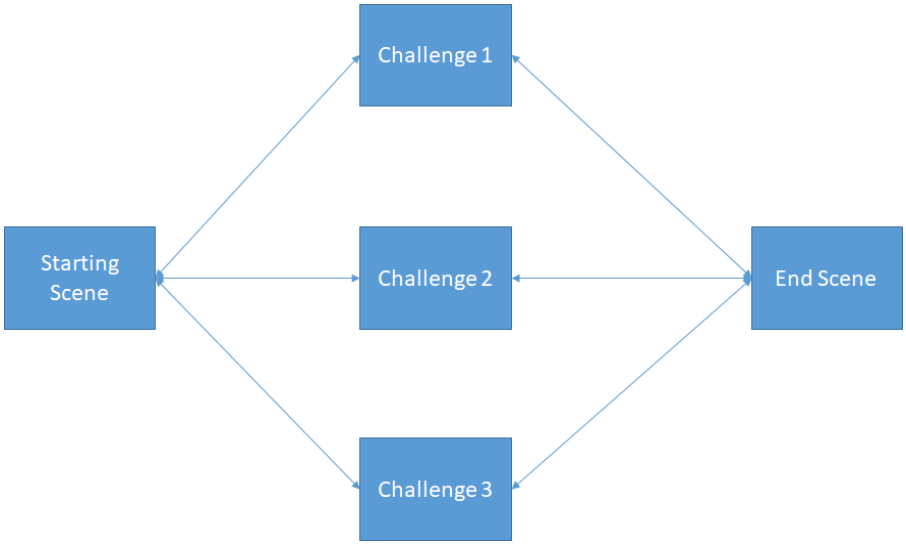
Components:

1. **Client:**
   Contains the executable game.
2. **Server:**
   Process requests send from client and reply back to client.
3. **Database:**
   Store and manages relative game data, such as user settings, character information as well as game state.

High Level Overview:



Scenes Overview:



Player will have the freedom to switch between all 5 scenes through the UI Menu. E.g. If player are done playing with challenge 1 and want to quit the game, they can switch directly from challenge 1 to end scene.

Storytelling:

Oh, no! Purdue has been invaded by characters from Fortnite. These characters captured the BoilerKey, the key to Mitch Daniels' machine to freeze the tuition. As a intelligent and fearless BoilerMaker, you will concur different challenges and retrieve back the BoilerKey. The BoilerKey has been broken into three parts by the Fortnite characters. Once, you completed all three challenges and retrive the whole BoilerKey, the Fortnite characters will disappear and the campus will return to peace; more importantly, tuition will be frozen again!

Scene Planning:

a) **Starting Scene:**
   1. **Background:** Alert! Purdue has been invaded by Fortnite and we need to get back to work. Fight off all the slackers and get the BoilerKey.
   2. **Game Environment:**
      - Overall view of Purdue campus
      - Visual details such as street light lights up, car movement, foggy sky, etc.
      - Audio details such as background music, audio effect when player select a level, etc.
   3. **Game Mechanics:**
      - player will be able to choose different levels through left or right arrow keys, the camera will follow the switching action.
      - Player can customize their in-game name, character physical traits, information about the character.
   4. **UI:**
      - a main menu will display different option such as, 'quit', 'save' and 'resume' as well as level indicators and so on.
      - Menu for player to customize the in-game character.

b) Game Challenge 1

1. **Background:** Fortnite characters have invaded Lawson building and you have to go on a adventure to get part 1 of the BoilerKey.
2. **Game Environment:**
    - Inside of the Lawson building
    - Visual details: muzzle flash when player crashes Fortnite characters, animation of Fortnite characters, etc.
    - Audio details: exciting background music with 3D effect, sound effect when player step on Fortnite characters.
3. **Game Mechanics:**
    - player will be able to control the character through arrow keys to move left/right, jump and squad.
    - Player will earn points either by jumping and crashing Fortnite characters or collecting coins.
4. **UI:** display time left, display score, access to main menu

c) Game Challenge 2

1. **Background:** It's snow time! That means it is also sledding time! You will need to sled down Slayter hill while successfully dodging objects. Once you reach down the bottom of the hill successfully in a fixed amount of time, you will acquire part 2 of the BoilerKey.
2. **Game Environment:**
    - A terrain will be modelled to mimic Slayter hill.
    - Visual details: snow effect, trail of the sled, animated crowds.
    - Audio details: background music, wind noise, noise from the crowds with 3D effect.
3. **Game Mechanics:**
    - Player can control the speed of the sled by up and down arrow keys, as well as control the directions by left/right arrow keys.
    - Player can obtain extra points by collecting coins along the way sledding downhill.
    - Player has to reach the destination in a fixed amount of time.
4. **UI:** access to main menu, display score, display time left

d) Game Challenge 3

1. **Background:** Congrats, BoilerMaker! You have made to the final challenge. The third part of the BoilerKey is hidden inside a forbidden place on campus and guarded by many Fortnite characters. You have to fight them and find the final part of the BoilerKey.
2. **Game Environment:**
    - A mysterious place on campus that will surprise the player.
    - Visual details: automated environment elements, different lighting effect, textures will have different appearance according to the angle of light.
    - Audio details: exciting background music, majority of game objects will have 3D audio effect, sound effect when Fortnite characters are defeated.
3. **Game Mechanics:**
    - Player will have full control of the characters, full three directional movement and casting spells.
    - Player will have to defeat the AI enemies to survive.
    - Player will use the given clue to find the final part of the BoilerKey.
4. **UI:** access to main menu, conversation boxes to give the player clues, display HP of the AI enemy, display the HP as well as strength of the player.

e) End Scene:

1. **Background:**
   You have now completed all the challenges and acquired the BoilerKey, the key to success.
2. **Game Environment:**
    - A short video will be played to tell player the ending of the story.
3. **Game Mechanics:**
    - Player will have the option to exit the game, restart the game or check the credit.
4. **UI:** access to main menu, credit screen.

Design Issues

Functional Issues:

1. How should the scores and achievements be displayed?
    a. Option 1: Integrated in user interface at all time
    b. Option 2: Separate score screen that can be accessed in menu
    c. Option 3: Displayed when user enters each game level

    Decision: Option 1 and 3; the total score of the user will be displayed in the overworld at top right side of the screen, which then we can decide whether user can see the score break down for each level of the game. Once the user enters the level of the game, they will see the score of that level only instead of their total score. This is to reduce ambiguity of the score while the user is playing in a certain game level.

2. How will users be informed of controls and game instructions after the first tutorial?
    a. Option 1: Game manual is accessible in menu, which then leads to different screen
    b. Option 2: Each level will allow users for manual every time user enters
    c. Option 3: No manual will be available after the first tutorial

    Decision: Option 2; user will be given an option to read the manual once they enter the game via NPC (non-playable character). The user may choose to ignore the NPC if they already know the controls and instruction, but they may also read what NPC says to remind themselves of the instruction. This allows user to progress hastily if they know what they are doing while providing them an opportunity to learn the game if they forgot.

3. How will the user know where to travel and what to do in the map of Purdue campus?
    a. Option 1: Have general guideline be available on the screen

b. Option 2: User will travel anywhere they would like and eventually encounter the features
c. Option 3: User will travel seemingly open world but in one direction

Decision: Option 3; User will travel in the overworld that looks like a graph where each vertex will represent the location user can travel. The user will only travel via edges between the vertices, and each vertex is where the user will enter a game level to play the game. This reduces the ambiguity and is simple for the users to follow along without being lost.

4. How should we implement the overworld?
    a. Option 1: 3D approximate map of purdue with beacons at level locations
    b. Option 2: Stylized map of purdue with only level objects displayed

    Decision: Option 1; the overworld will display the map of purdue with certain points of the map being the beacon for a game level. This reduces the amount of necessary modeling. It also gives users easier understanding of the map and their progress as the game and story moves on.

5. What should be the story?
    a. Mitch Daniels needs your help! Purdue has been invaded by fortnite and we need to get back to work. Fight off all the slackers or he'll unfreeze tuition (fortnite models from sketchfab.com)
    b. The player needs to get to class, going through the three levels to get to a purdue building (any building)
    c. The player is participating in BGR and is walking around campus trying to find their group (the group appears on the map and is always at the next level, aka princess peach)

    Decision: Option 1; we decided that having a theme of defeating the enemy allows us to design each level more easily. Also, involving the idea of fantasy (Fortnite) enables us to bring in whole different genre and

assets to the game. With defeating enemy theme, we can provide users an overarching goal to defeat the final boss in the last level while the first two levels will be the stepstones for it.

Non-Functional Issues:

1. Should we try to include mobile users?
    a. Option 1: Design the game with PC/MAC/LINUX support at the forefront and only branch to mobile if time allows
    b. Option 2: The game is fully compatible with mobile devices
    c. Option 3: The game has exclusive support for PC/MAC/LINUX

    Decision: Option 1; there is no reason to entirely rule out the option of mobile porting. While there are certainly advantages to expanding the potential audience and device compatibility, its lack of essentiality means that it should only pursued if time allows.

2. How do we limit the amount of modeling we have to do?
    a. Option 1: Model a limited portion of campus at peak accuracy, surrounding it with preexisting, pre-modeled assets
    b. Option 2: Model all of campus with low accuracy, combining with preexisting, pre-modeled assets
    c. Option 3: Model a select few buildings, integrating them into a low accuracy model with preexisting, pre-modeled assets

    Decision: Option 3; Well-designed models of the buildings that host the most frequent player interaction is essential to creating a convincing and lively game environment. In the interest of time it is also essential to not be required to model every structure in every scene. Therefore, the most logical option would be to model a select few structures while including both pre-modeled assets and rougher, less detailed models of lesser used buildings.

3. What game engine should be utilized?
    a. Unity
    b. Unreal Engine
    c. Godot

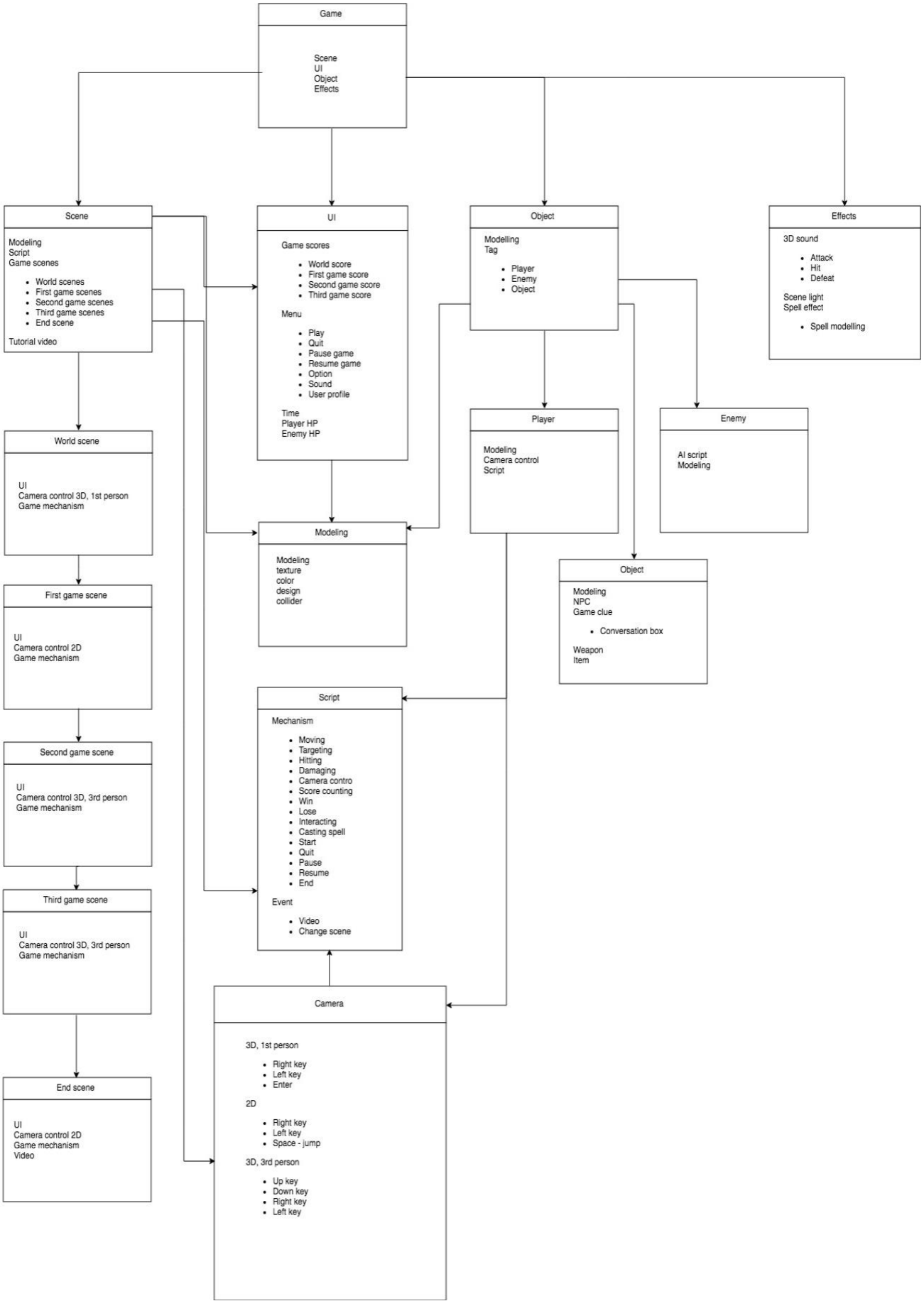
    Decision: Option 1; Unity has ample support for collaboration and provides resources and extensive documentation compared to the other options that should prove helpful in the creation process.

4. What scripting language should be utilized?
    a. Option 1: JavaScript
    b. Option 2: C#
    c. Option 3: Boo

    Decision: Option 2; C# most strongly balances familiarity and features compared to other options. JavaScript only has pseudo-support and additional mechanics of the language would need to be learned. Of the other options we have the most familiarity with C#.

Design Details

Class Diagram



Descriptions of Classes and Interaction between Classes

Our class design is based on an understanding of each scene. The game has a large map which connects each scene. Each scene may have one or two games. The last scene is boss fight.

- Game:
  It contains several aspects of the project
  Created at the beginning of the game
  This part connect the game from other major part of this game

- Scene:
  Abstract class that has similarities from other scenes
  Created at the beginning of the game
  Contains a tutorial video which may play at the beginning of the game.

- UI:
  Used to control current game, includes pause, resume, quit.
  Since it's general UI, it will be used across all the different little games.
  UI will be in the corner of screen so that player can access that at any time.

- Player:
  Data of the player, may include health and other status
  Created at the beginning of the game.
  Used to interact with other objects in the game

- Enemy:
  Contains all the data of enemy, include enemy actions, how enemy attacks.
  Boss does not belong to this category, Enemy only appears in the first few scenes.
  Created when the player enter each individual scene.

- World Scene:
  Beginning scene for the player
  Index of the whole game, player can choose which little game to play first
  Contains a score of player
  Player can also change basic settings in this scene

- First Game Scene:
  Contains its own UI and data for this game
  Contains structures for game mechanism
  Created at the beginning of first scene

- Second Game Scene:
  Contains its own UI and data for this game
  Contains structures for game mechanism
  Created at the beginning of second scene

- Third Game Scene:
  Contains its own UI and data for this game
  Contains structures for game mechanism
  Created at the beginning of third scene

- End Scene:
  This scene can be selected from the large world
  This is the last scene of the whole game(last sprint)
  It is for the little game of final boss fight
  Should be a maze running game, it contains structure of the maze

- Modelling
  Not a class or script, since it’s a game, we will model some buildings or characters in blender.
  Each model may consists with several sub models (arm, leg...) so that it will interact with the other object.

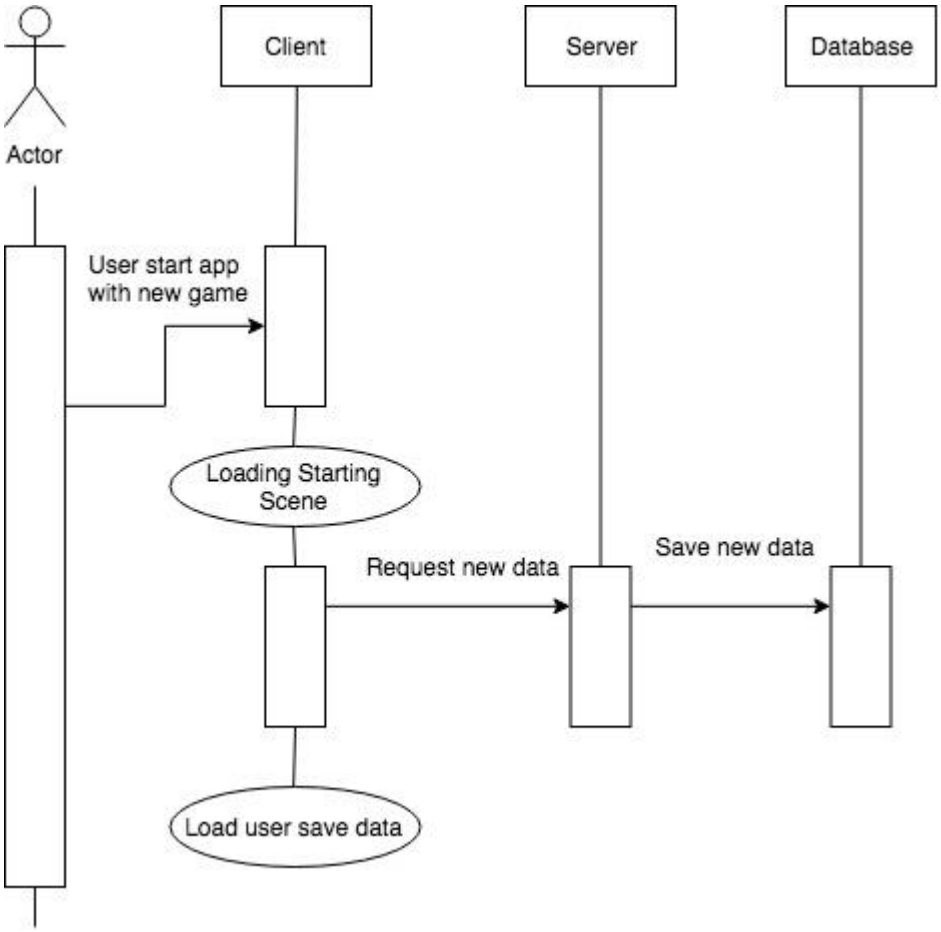
- Script
  Physical script will add physical features to the game
  May use to change Gravity, density
  Contains functions to define control systems.

- Camera
  Camera will keep moving since this project has several scenes.
  Contains three different axes.
  Each scene will reset the position of camera at the beginning of the scene.

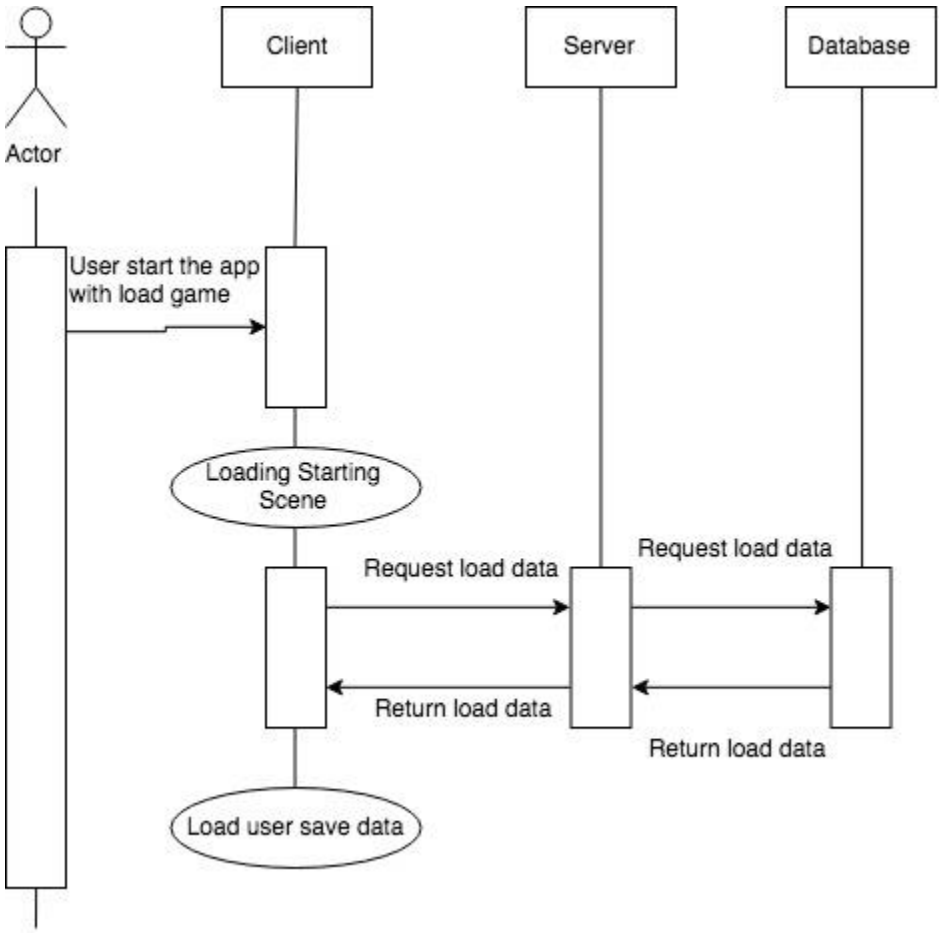
- Effects
  Collision effects that when player attach enemy.
  Contains several items and their data since those items can be used in different scenes.

Sequence Diagram

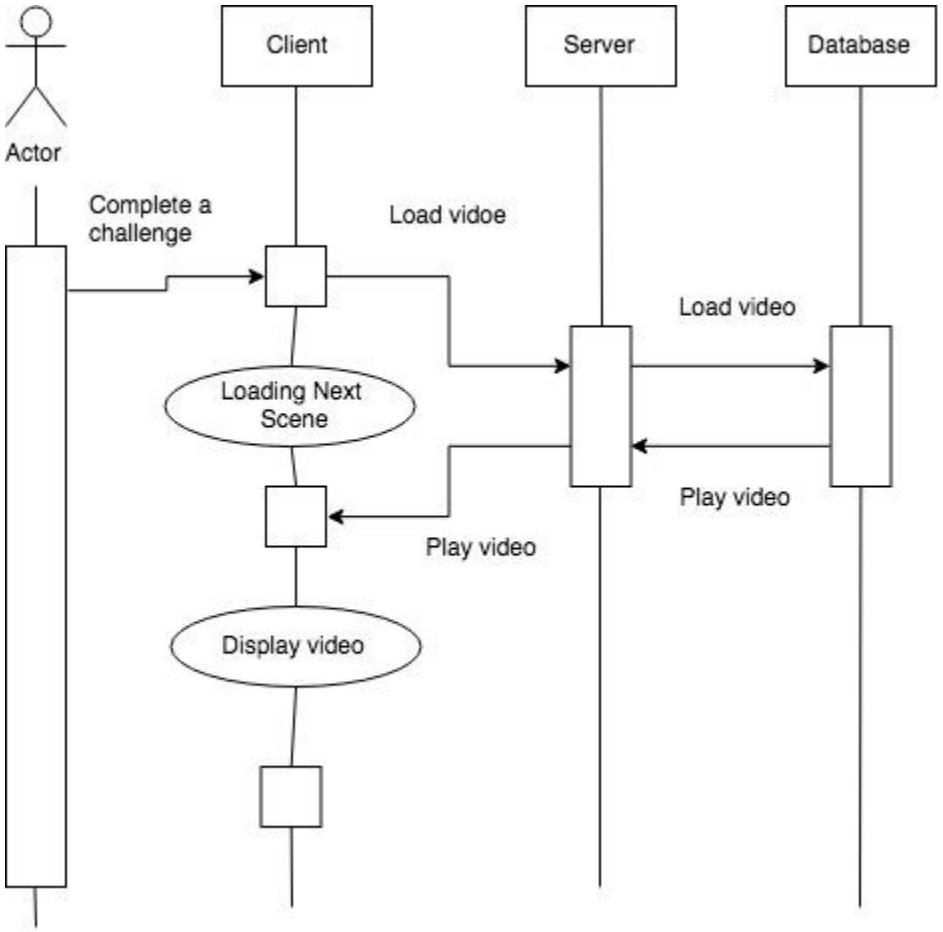
Sequence of events when the user first start the game.



Sequence of events when the user load the previous saved game

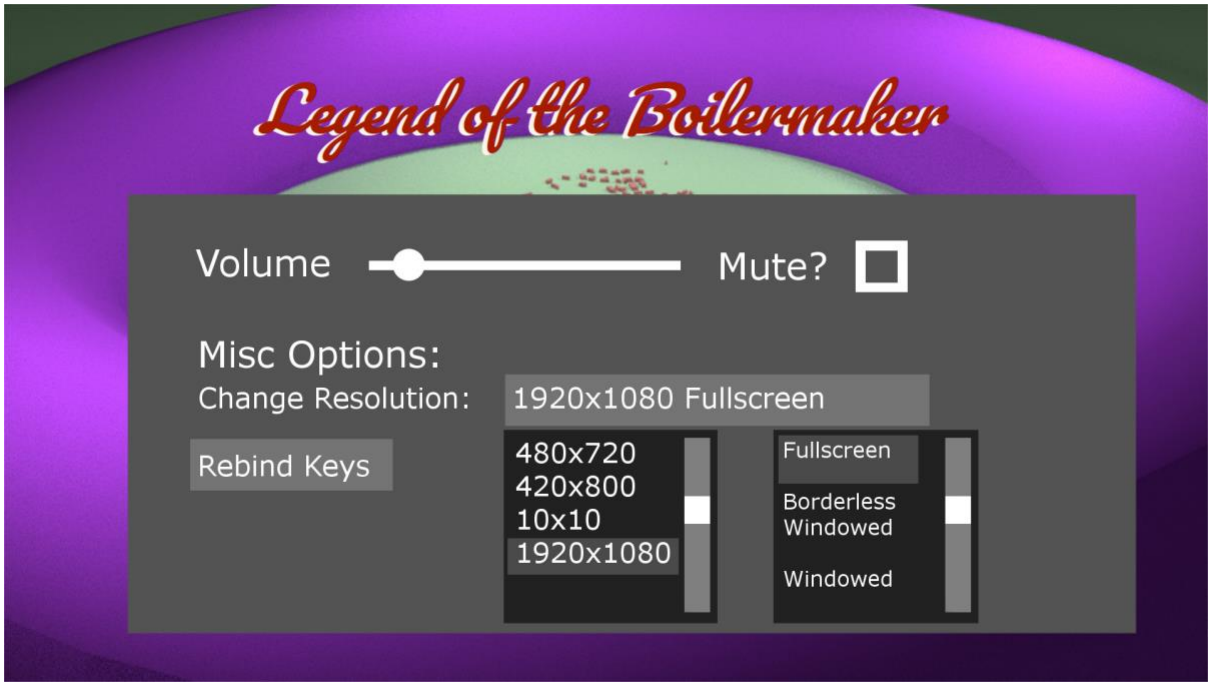


Sequence of event when the user completes a certain challenge to move on to the next scene where the next scene contains video to play

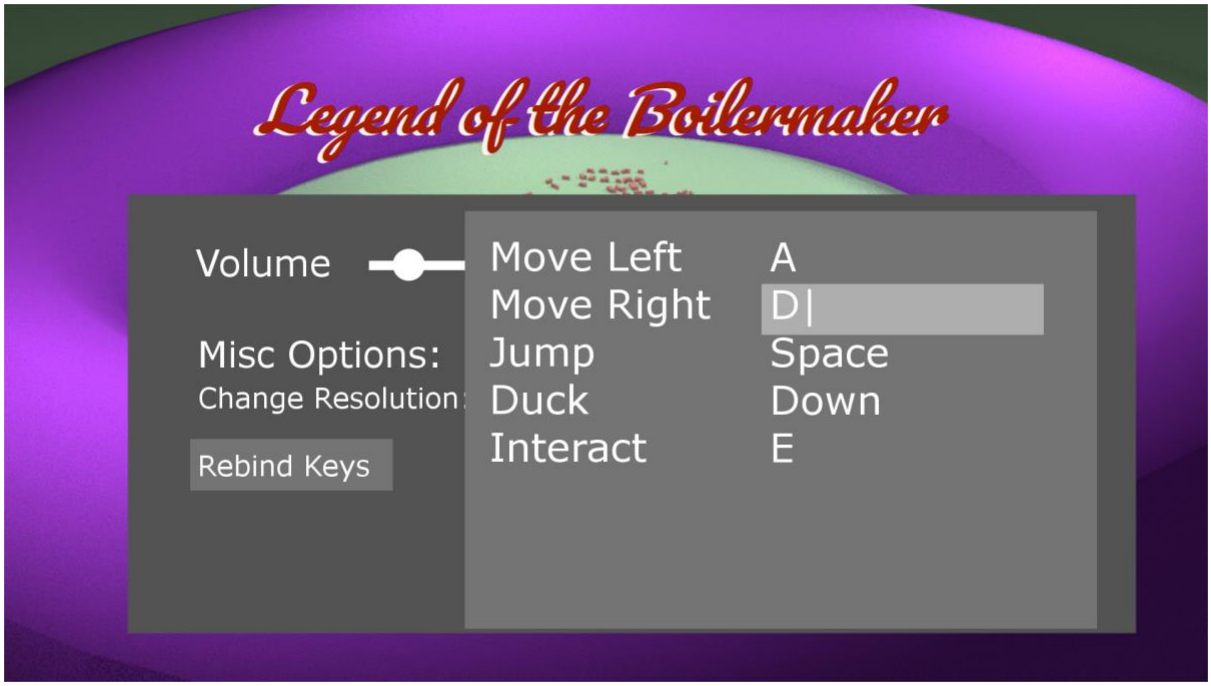


UI Mock-Up

- Start screen of the game
- Play (user will be prompted of New Game or Load game)
- Settings (user will be able to change settings for control, display, etc.)
- Quit (user will exit the game to desktop)



This menu appears when you press the “Settings” button from the previous UI. The resolutions and display options can be selected from the dropdown box, and will be highlighted when selected. Clicking mute disables audio, and clicking and dragging the volume slider changes the volume between 0-100%. A check appears on the mute box when toggled.

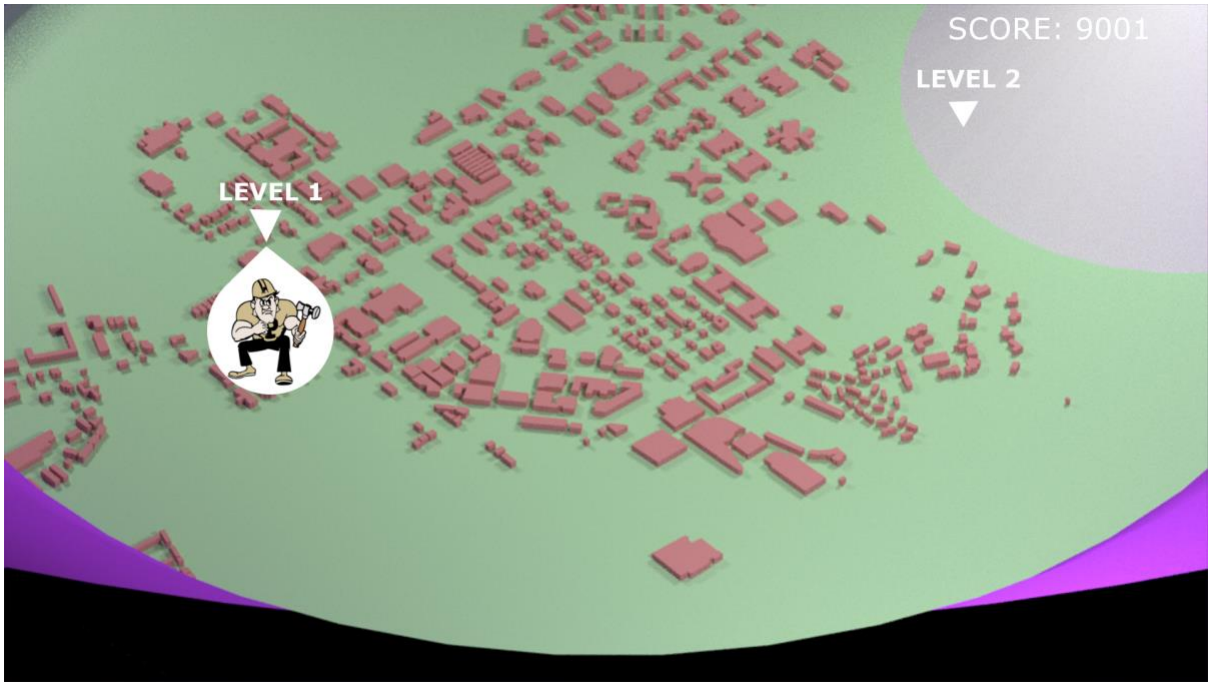


This screen appears when you click the “Rebind Keys” button. Clicking on the left row of entries will allow you to type the key you want to rebind that action to. If the player pressed the space key, the text ‘space’ will be displayed in that place. Invalid entries will not change the value when the key is pressed. (eg: typing in space for duck after it is already bound will not change the text or control after the keypress)

- Screen after pressing Play
- New Game will start a new game for the user
- Load Game will prompt another screen where the user can select from saved files

- Loading a game
    - User will load a game from the saved states
    - Each saved state will display basic information of the progress user had made.
    - Pressing Load next to each save entry will start the game from the saved state of the game

This screen is accessed by pressing the Load Game button. There are three total slots to choose from, and they can be saved during the gameplay from the pause menu. Loading a save will always put a player in the overworld, but will remember progress the score.



This is the screen where the player chooses what level to go into, and where the player is placed when they create a new game or if they load a save. This will be rendered in real time by the unity engine, and will be more detailed than the mockup. The player's position is indicated by the Purdue Pete icon, and by pressing the forward and backward movement keys they can navigate between levels they have unlocked. Pressing the interact button puts the player in the level that they have selected

The clothing store is selected as a level from the menu