# 3ly-Zium (Elysium)
# for UTM CSCI 352

Colin Weatherly, Blade Johnson

**Abstract**

**This project is a video game in the 2D platformer genre. The player will run through a level, and upon each successful completion, the level is changed to increase the difficulty. The player will play through the stages in an attempt to reach high difficulty stages. The intended target is people that play video games, primarily those that stream games, as live reactions mesh well with high difficulties.**

## 1. Introduction

This project is a video game in the genre of 2D platformer. The goal of the game is to reach the exit of the level that the player character is placed in. The level's layout will progressively get more complex with each clear, along with adding new hazards that increase the difficulty further. The penalty for failing a level is to be sent back to the beginning of said level, allowing the player to practice the level with every attempt.

The targeted audience for this project is people in the 18-34 age group who play video games on a regular basis. Specifically, those in the age group that livestream games. Due to the increasingly difficult challenge that the game provides, having live reactions of attempts to beat the game will provide content for the targeted audience.

### 1.1. Background

A 2D platformer is a game genre set on a 2D plane where the player controlled character must move and jump to avoid obstacles and reach the end.

We decided to make a platforming game because of our love for games in general and our shared interest in platformers. Because of this, we feel we can properly create one.

### 1.2. Impacts

This project is not meant to impact the world in some way. The point of a game is to provide entertainment, which will only impact a portion of the population if it were to gain traction within the community. At the current time, there is no plan for a meaningful story in the game, so the impact will be purely driven by the quality of the gameplay.

### 1.3. Challenges

The main challenges were coding the physics of the player character, developing new mechanics instead of only having the ability to move left and right and jump, and keeping the game fun despite the difficulty. Making content for streamers and making the levels increasingly challenging were also challenges.

## 2. Scope

This platforming game is considered done once three levels of the game are playable from start to finish with minimal bugs or exploits. The player should be able to get through these stages without too much unnecessary frustration with either the difficulty of these stages or the control of the player character. The player character should be fully implemented with all moves intact (moving, jumping) and all hazards(such as spikes) should fail the level upon contact with the player character. As a stretch goal, we want to include more levels with new hazards. An additional stretch goal is the addition of extra moves(i.e walljumping, sliding) the player character requires to progress in more difficult levels. An additional stretch goal is scaling the window and the game assets to fit more than one resolution.

### 2.1. Requirements

The requirements were gathered by considering the needs of the user when playing though and practicing the levels in the game. We determined what would fit best for playability and user convenience, especially in the case of a 2D platforming game. We then used these to put together specific necessities of the program.

| Use Case ID | Use Case Name | Primary Actor | Complexity | Priority |
|---|---|---|---|---|
| 1 | Start New Game | User | Low | 2 |
| 2 | Level Select | User | Low | 3 |
| 3 | Character Jump | User | Medium | 1 |
| 4 | Character Dash | User | Medium | 1 |

TABLE 1. USE CASE TABLE

#### 2.1.1. Functional.

- User inputs via designated keys are read – the application should register the inputs and move the player character accordingly
- Selecting Start places the user in the first level – Unlike the level select which allows the user to choose a level, pressing the start button will immediately bring them to the first level
- Level fails upon contact with a hazard – should the player character run into a hazard(i.e pits), level is failed and character is returned to the starting point of the same level

#### 2.1.2. Non-Functional.

- The game correctly displays in a 1280x720 resolution window
- The game runs at a smooth framerate, preferably 30 frames per second
- User's inputs have little to no delay, preferably under half a second

### 2.2. Use Cases

These are cases in which the user will interact with the program's user interface. The use cases can be seen in Table 1.

Use Case Number: 1
Use Case Name: Start New Game
Description: The player decides to begin the game from the beginning (Level 1). They will click on the "Start" button (see Figure 2). This will load the first level of the game (see Figure 1).
Process flow:

1) Player starts the program, which begins with the main menu loaded.
2) The player left-clicks the "Start" option on the menu.
3) The game state is updated to gameplay, and the first level is loaded.

Termination Outcome: Gameplay has begun in level 1.

Use Case Number: 2
Use Case Name: Level Select
Description: The player should be taken to a level select when selecting the option on the main menu, where they can select any level to play (see Figure 3).
Process flow:

1) Player starts the program, which begins with the main menu loaded.
2) The player left-clicks the "Level Select" option on the main menu.
3) The level select screen is loaded, showing each level vertically in numerically descending order.
4) The player left-clicks the option corresponding to their desired level.
5) The game state is updated to gameplay, and the corresponding level loads.

Termination Outcome: Gameplay has begun in the player's selected level.

Use Case Number: 3
Use Case Name: Character Jump
Description: The player character should jump into the air upon the player pressing the 'Z' key.
Process flow:

1) Player is in a stage of gameplay.
2) The player presses the 'Z' key.
3) The player character then leaps off the ground into the air.
4) After a few seconds of gradually rising upward, the player character then gradually falls until hitting the ground.

Termination Outcome: The player character has completed a full jump.

Use Case Number: 4
Use Case Name: Character Dash

Description: The character should dash forward in the direction being faced when pressing the ‘X‘ key.

Process flow:

1) Player is in a stage of gameplay.
2) The player presses the ’X’ key.
3) The player’s movement speed is doubled in the direction of movement(left, right) for a second.

Termination Outcome: The player has completed a full dash.

### 2.3. Interface Mockups

In Figure 1, this shows the player character (in the top left corner of the mockup) standing in the initial state when Level 1 begins. The character can traverse the platforms (the graphical rectangles that stick out from the background) to reach the right of the screen (which is where the goal would be). Falling in the pit would send you back to that position. Figure 2 shows the title screen that you see upon starting the program. The three options under the graphic (from left to right) start you into Level 1, take you to the level select screen, and close the program respectively. Figure 3 shows the level select which you can reach from the title screen’s middle option. Selecting “Level 1,” “Level 2,” or “Level 3” will start the player in the corresponding level of the game.


Figure 1. Mockup of a player loading into level 1

## 3. Project Timeline

Refer to Figure 4.

February 5th is where we propose the idea of a 2D platformer game.

March 1st sees us writing up use cases and setting requirements. Concepts are put into place with mockups being created.

March 28th is when the project is updated with designs that are more fleshed out and close to final. Initial structure ideas are outlined in a preliminary UML.

April 8th has movement being more or less completed, with wall and ground collision now being worked on.

April 15th has collision being finished or nearly finished. Level loading has begun implementation or is approaching implementation.

April 22nd has level loading finished or nearing completion. Menus have begun implementation. Bug testing has begun.

Lastly, April 28th has the project being completed, having met all goals and running with few bugs.


Figure 2. Mockup of the main menu screen

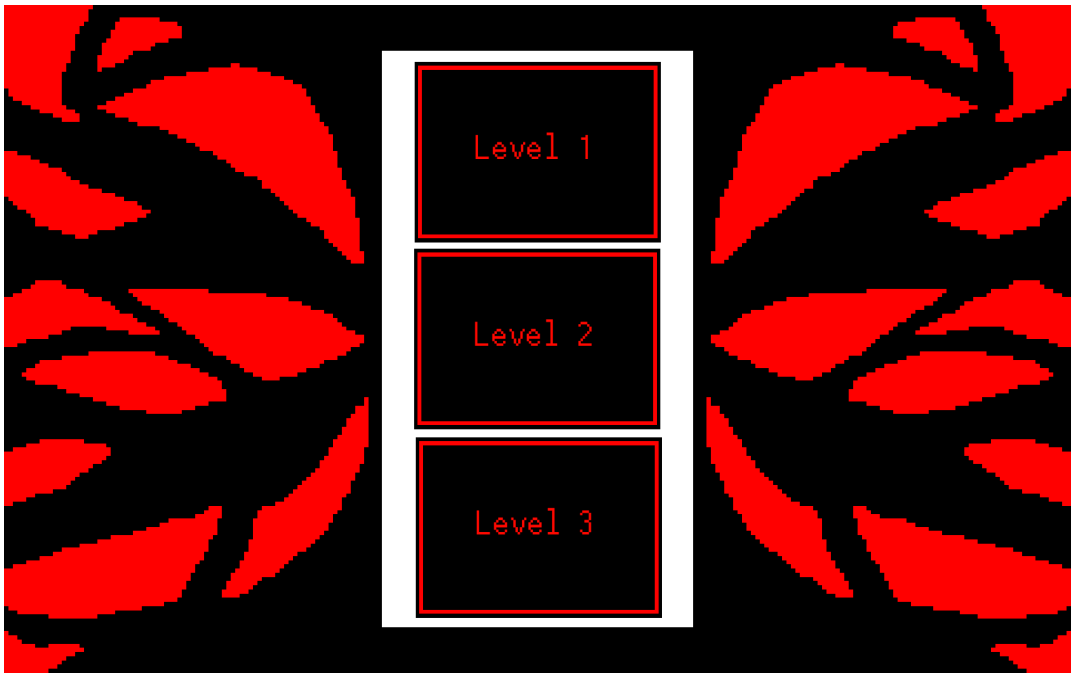


Figure 3. Mockup of the level select screen

## 4. Project Structure

Our overall approach was to make a game that featured simple menus that can head straight into gameplay or start gameplay at any point in the game.

From main, the menus will be loaded. From here, the Window1 class (Level Select) will contain methods for all of the choices, which will start the game, access the Window2 class (Level Select), or close the program. The Window1 and Window2 classes can both load into the gameplay. Window3, Window4, and Window5 classes (Levels 1, 2, and 3) run

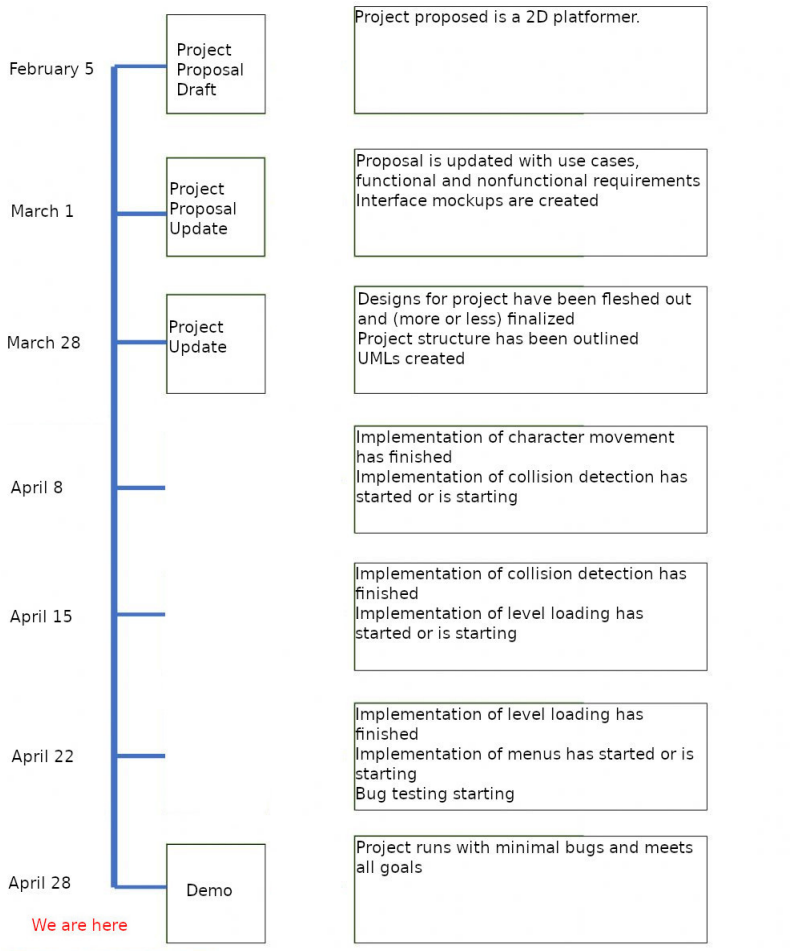


Figure 4. Timeline

the gameplay. These three classes contain the information for the character as well as level data and collision data. The MusicClass also plays music for the game in the background in each screen.

- MusicClass: We decided to make MusicClass a separate class so that it could be easily instantiated in any class and play the required music.

- QuitDecorator: We decided to add this as a small detail when you exit the game. We wanted to have a ”Thanks for playing!” message to let the user know we appreciate the time they took to play it.

- Graphical Direction: We decided on a sci-fi style art directions with a neon city in the background and a robotic-looking main character. This mostly came about from graphical experiments, but it ended up sticking.

## 4.1. UML Outline

Refer to Figure 5.

Window1 (title screen) is what the user loads into upon start up, as MainOverride creates an instance of Window1. Button_Click_1 creates an instance of LevelSelect, which allows users to start from any level in the game, and Button_Click_2 loads an instance of Window3 (Level 1). Button_Click_3 simply closes the program.

Window is where the user can select any of the 3 levels. Button_Click_1 creates an instance of Window3, starting you in the first level. Button_Click_2 creates an instance of Window4, starting you in the second level. Lastly, Button_Click_3 creates an instance of Window5, starting you in the third level.

The MusicClass creates a music player in each of the windows it is instantiated in (Window1, Window3, Window4, Window5). This allows these screens/sections of the game to have music played in the background.

QuitDecorator is a class that decorates the Button used for quitting out of the program on the title screen. It allows the button to display an extra message when sent, which is ”Thanks for playing!”

Each of the 3 level Windows (Window3, Window4, Window5) contain all of the data for that level as well as how the player collides with the level. Upon completing each level, the next is loaded.

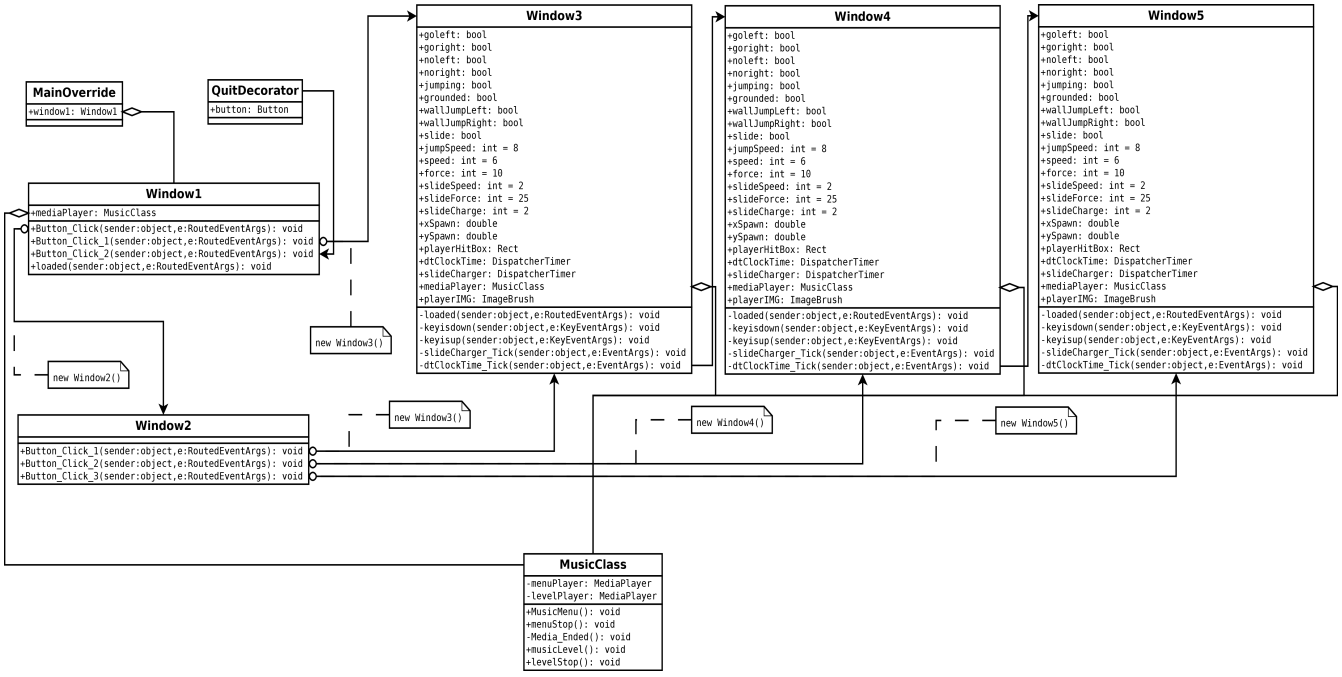


Figure 5. UML Outline

## 4.2. Design Patterns Used

The 2 design patterns used are the Factory pattern and the Decorator pattern.

Factory: Your selection from Window2 (Level select window) will create an instance of one of the three levels depending on the selection.

Decorator: A QuitDecorator decorates one of the buttons on Window1 (title screen) to add extra functionality. In this case, it adds a “Thanks for playing!” message whenever clicking it.

# 5. Results

While the project was difficult in some places to create, things came together fairly well. We were able to present the project in full on April 28th, and it seemed to have gotten some positive looks from people. In the end, we ended up satisfying all requirements we need and finished the project. The game is fully-functional and can be completed from beginning to end. While we were not able to act on many stretch goals before the presentation on April 28th, we had a good amount to show regardless. A refactor was in the works for certain parts of the project, but was not able to be completed, as work for the main branch was being updated simultaneously, and we weren’t able to keep the refactor up-to-date. Due to this, some code in the main branch still remains the slightest bit messy.

## 5.1. Future Work

From here, we plan to potentially clean up what we have even further. After all is said and done with this project, we don’t have any particular plans to do anything more with this project when it comes to distribution or anything else. We figure this will just remain as a past project, and will be included in our portfolios.