Coding with Octocat

Jennifer Huestis and Trever Hall

Abstract

The project is a side scroller that draws inspiration from Google Chromes' dinosaur game easter egg. It is going to be computer science themed with Octocat running and trying to avoid "bugs and broken windows" as obstacles. The form of currency in the game is caffeinated products, which can be used to buy in game upgrades. The main goal is to survive as long as possible, and you are told the amount of time you survived (given by a running timer).

1. Introduction

This is a project that we are doing for a class. We don't expect to publish it or for it to have a large user-base, but we think it will be a fun and challenging game that is simple enough for anyone to play. It will follow the generic platformer design with the player auto running and incrementing the speed the longer you survive. The user will have to jump to try and avoid the obstacles. The player only has one life, so once they hit an obstacle they die.

Challenges we foresee are getting the screen to scroll and making sure everything is timed correctly so that the pixel art actually looks like it is moving. If it is not as challenging as we foresee, then we also have some stretch goals that we think would be nice to add to the game. Those would be things like adding more items into the game and having long-term high scores.

1.1. Background

If not familiar with the concept of a platformer, it is a two dimensional game with a player that can move within that plane. Examples of games like that include the original Mario games, Flappy Birds, Terraria, Donkey Kong and many more! These games of our childhood have inspired us to create one of our very own.

1.2. Challenges

There are many challenges to cresting a video game in WPF, but ones that stand out to us are: scrolling, timing, and pixel art.

1.2.1. Scrolling. The screen needs to scroll to the right. That is going to require auto generation of the screen from the right and deletion from the left to give the user the illusion of movement.

1.2.2. Timing. There are many elements that rely on a system for keeping track of time. The obvious one is the user needs to know how long they survived. Other than that though, enemy and player velocity depend on a the amount of time passed (in this instance pixels per second).

1.2.3. Pixel Art. All graphics are going to be complicated since its on WPF, and we must implement a system that consistently updates the window so that you can see the graphics move. Not only do we need to time movement, but we also need to change the art with every update so that it looks like characters are moving (walking, running, flying, etc).

2. Scope

The game begins with Octocat auto running toward the right after a user hits "start". When Octocat is running randomized obstacles (broken windows and bugs) will generate from the right of the window; the player will have to avoid them by jumping or ducking. The longer you survive (time is displayed on the screen and counting as you play) the faster the Octocat will run. When you die it will display the time that you survived as well as items collected on that round. The user can use caffeinated items they pick up to buy upgrades (minimally one upgrade). The requirements seem simple at first, but are difficult to fulfill.

2.1. Stretch Goals

There are a lot of additions we would like to add to the game, but two that we believe would add the most to the user experience are long-term high scores and adding more power ups and other useful items into the game.

2.1.1. Stored High Scores. We want to keep a record of the top ten scores and allow the user to enter a name to associate with it (just three letters like the retro games in arcades). This should be accessible from a button on the menu. The goal is to make it keep track even if you close the game and re-open it.

2.1.2. More Items. We want to add more items to buy. Things that we would like to include are an invincibility power-up where Octocat will start riding a flying rainbow cat avoiding all obstacles, a companion (it just follows the player around), and a variety of colors for Octocat.

2.2. Requirements

To make the game functional and provide each user with a good experience, it will have to follow the requirements we gathered below:

2.2.1. Functional.

- User is able to toggle music volume on and off.
- Users is able to toggle sound effects on and off.
- User is able to purchase items.
- User is able to record three character name with high scores.
- User is able to quit the application.
- User is able to jump and duck to avoid obstacles.
- User is able to use inventory items during the game.
- User's character detects collision with obstacles and items and reacts accordingly.

2.2.2. Non-Functional.

- Platform – user can only run the application on capable windows machines.
- Usability – There is no native support for those with disabilities that may hinder the users experience.
- Reliability – The program will not stop running until the user closes it.
- Data Integrity – The high scores will be stored locally and maintained by the game.
- Maintainability – Once finished the game should not need any maintenance.

2.3. Use Cases

A user is able to follow the steps of the use cases provided below and get the specified outcome every time.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Adding a New High Score	Normal User	Med	1
2	Purchase an Item	Normal User	Med	1
3	View High Scores	Normal User	Low	2
4	Accessing Information	Normal User	Low	2
5	Start the Game	Normal User	Med	1
6	Change Octocat's Skin	Normal User	Med	3
7	Toggle Music on/off	Normal User	Low	3
8	Toggle Sound Effects on/off	Normal User	Low	3
9	Quit the Game	Normal User	Low	2
10	Avoid Obstacles	Normal User	High	1
11	Hitting Obstacles	Normal User	High	1

TABLE 1. USE CASE TABLE

Use Case Number: 1
Use Case Name: Adding a new high score
Description: The user has survived longer than an existing high score, or there is an empty slot in the high scores. Thus, the user is prompted to add there user three letter name to be displayed with the score.
Steps to add name to go the score:

1) User qualifies and is prompted by the game to add a name.
2) User types in 3 letters.
3) User clicks save out beside the text field

Termination Outcome: High score table is updated and displayed.

Use Case Number: 2
Use Case Name: Purchasing an Item
Description: The user wants to purchase and item.
Invariants: The user has sufficient funds to purchase the item selected.
Steps to buy an item from the store:
1) User navigates to the ”store” page.
2) User locates item.
3) User clicks the ”Buy” button bellow the item wanted.
Termination Outcome: The price is subtracted from the users currency, and one instance if the item is added to the users inventory.

Use Case Number: 3
Use Case Name: Viewing High Scores
Description: The user wants to view the high scores.
Steps to view the high scores:
1) User navigates to the main menu.
2) User left clicks the ”High Scores” button.
Termination Outcome: Screen displays high scores, and user can click the back button to return to the menu.

Use Case Number: 4
Use Case Name: Accessing Information
Description: The user wants to access background information and controls.
Steps to access information:
1) User navigates to the main menu.
2) User left clicks the information icon.
Termination Outcome: Moves to screen displaying the information, and user can click the back button to return to the menu.

Use Case Number: 5
Use Case Name: Start the Game.
Description: The user wants to start the game.
Precondition: User is at the start menu.
Steps to start the game:
1) User left clicks the ”Start” button.
Termination Outcome: The game will begin and Octocat starts auto-running.

Use Case Number: 6
Use Case Name: Change Octocat’s Skin
Description: The user wants to change the color of Octocat.
Precondition: User is at the start menu.
Steps to change Octocat’s skin:
1) User left clicks the Octocat icon.
2) User selects the desired color.
Termination Outcome: The color of Octocat icon and player is updated.

Use Case Number: 7
Use Case Name: Toggle Music on/off
Description: The user want to turn the music on or off.
Precondition: User is at the start menu.
Steps to toggle the music:
1) User left-clicks the music icon.
Termination Outcome: The music toggles on or off.

Use Case Number: 8
Use Case Name: Toggle Sound Effects on/off
Description: The user want to turn the sound effects on or off.

Precondition: User is at the start menu.
Steps to toggle the sound effects:
1) User left-clicks the sound effects icon.
Termination Outcome: The sound effects toggles on or off.

Use Case Number: 9
Use Case Name: Quit the Game
Description: The user wants to quit the game.
Precondition: User is at the start menu.
Steps to view quit the game:
1) User left clicks the quit icon.
Termination Outcome: The program will stop running.

Use Case Number: 10
Use Case Name: Avoid Obstacles
Description: The user wants to avoid obstacles.
Precondition: User has started the game, by clicking the start button.
Steps to avoid obstacles:
1) User clicks the up(jump) or down(duck) arrow to avoid an approaching obstacle.
Termination Outcome: The user does not hit the obstacle and the game continues.

Use Case Number: 11
Use Case Name: Hitting Obstacles
Description: The user fails to avoid an obstacle.
Precondition: User has started the game, by clicking the start button.
Steps to hit an obstacle:
1) The user does not use the arrow keys to jump(up) or duck(down) at the right time.
Termination Outcome: The user hits the obstacle and dies (unless saved by an item).

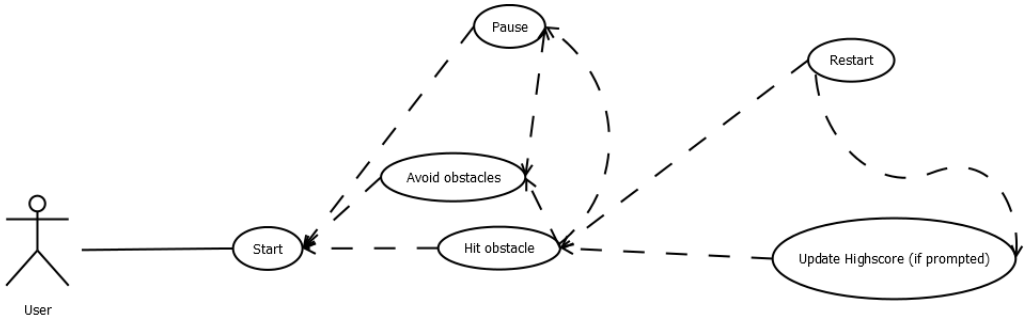

Figure 1. Use case to start the game and all of the use cases that it interacts with.

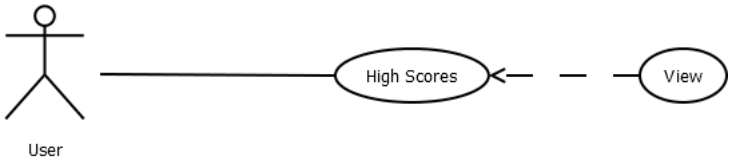

Figure 2. Use case for viewing high scores.

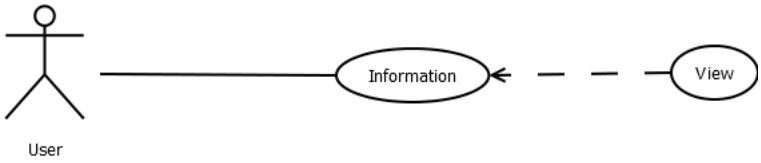

Figure 3. Use case to view information.

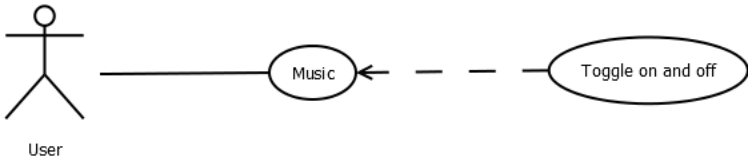

Figure 4. Use case to toggle music.

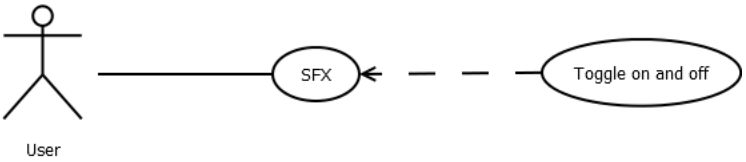

Figure 5. Use case to toggle sound effects.

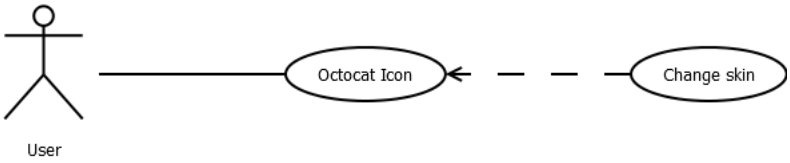

Figure 6. Use case to change Octocat's skin.

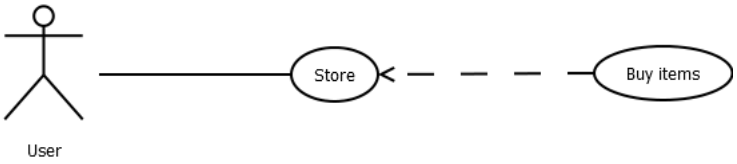

Figure 7. Use case to buy items from the store.

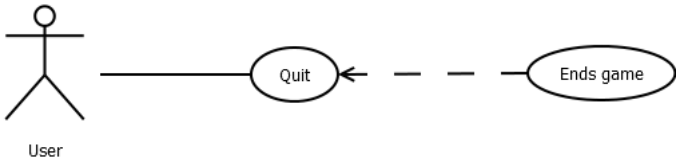

Figure 8. Use case to quit the game.

2.4. Interface Mockups

The mockups below are a rough draft for the design of the games interfaces. It is not exactly what the final project will look like, but does introduce what screens should be available and how the user will interact with them.

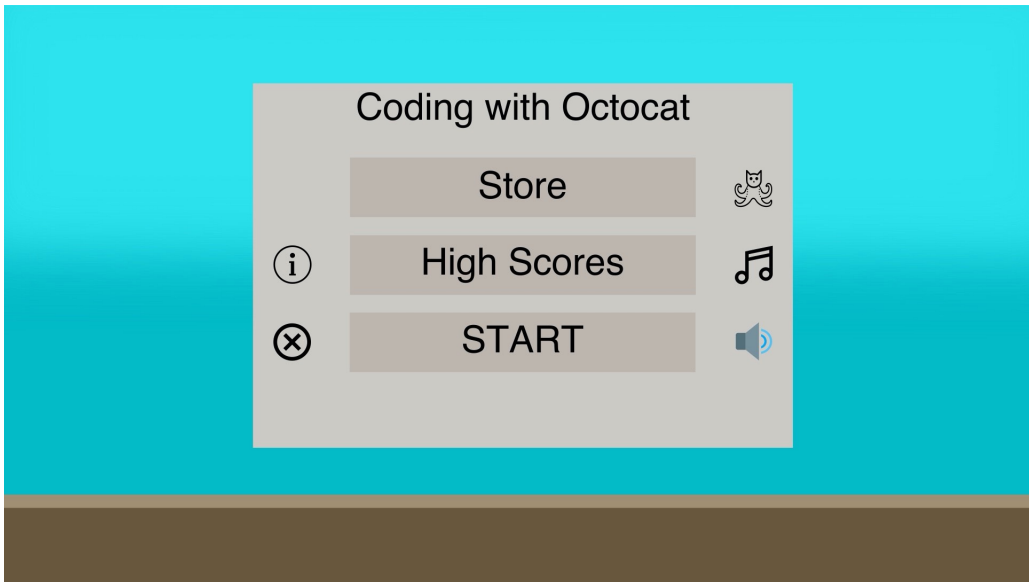

Figure 9. The start screen and menu where you can interact with the following use cases: start, view high scores, store, change skin, toggle music, toggle sound effects, view information, and quit.

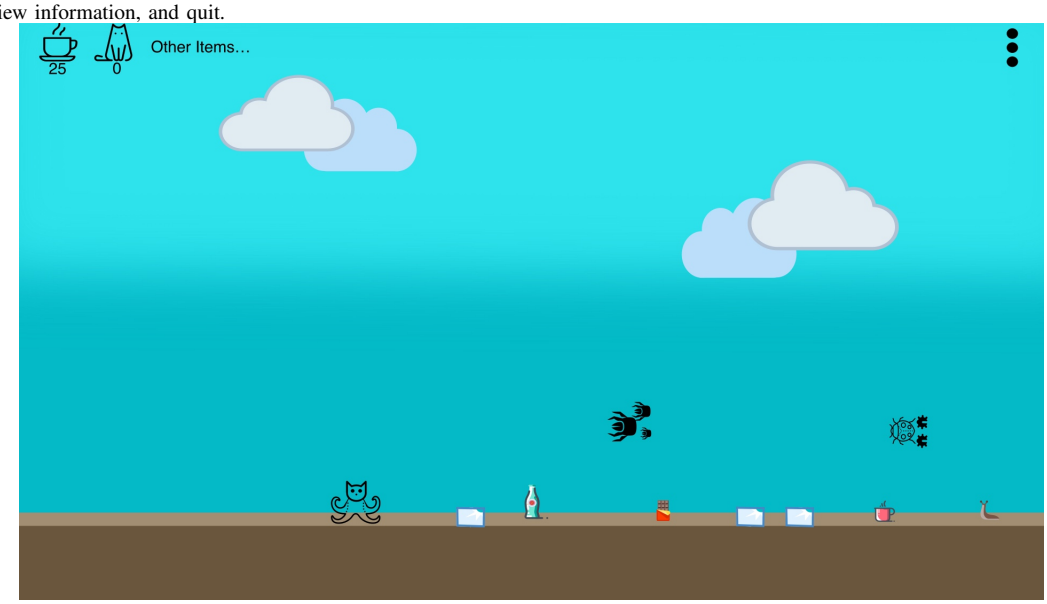

Figure 10. The screen where the user is in the game trying to avoid obstacles.

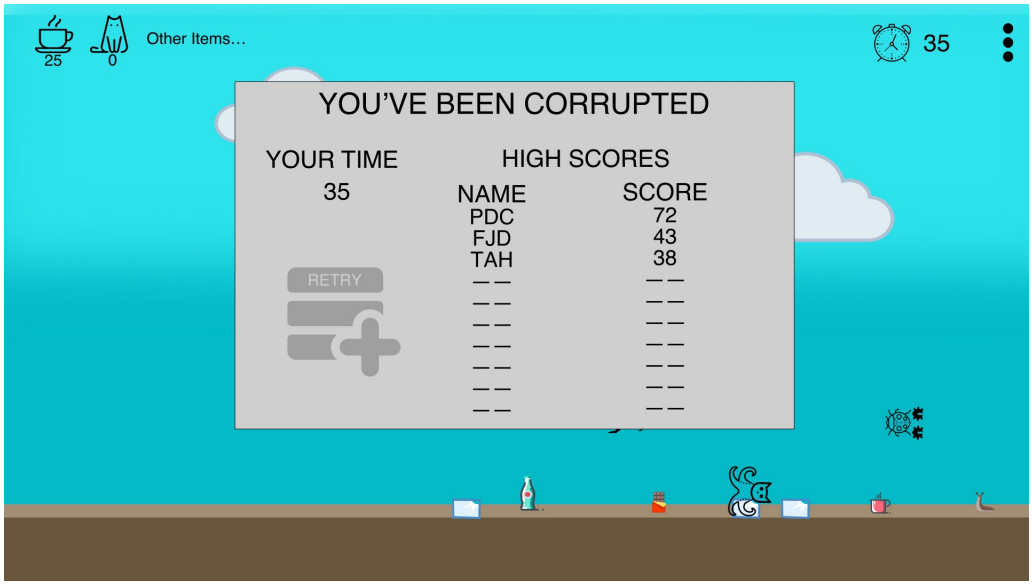

Figure 11. The screen where the user sees stats after they hit an obstacle and can restart.

Figure 12. The screen where the user can buy items and upgrades.

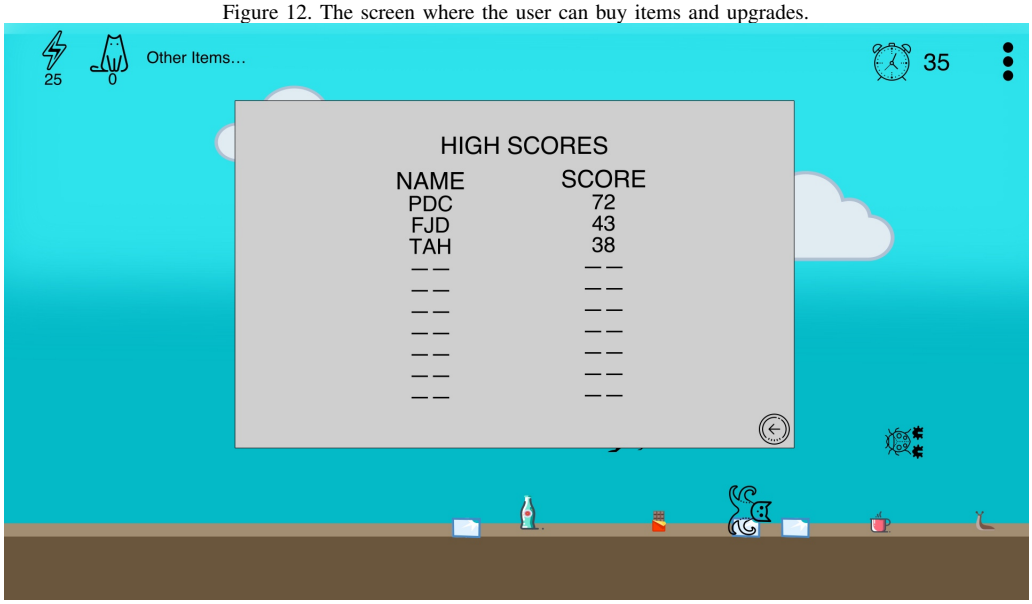

Figure 13. The screen where the user can view the high scores.

3. Project Timeline

To make sure we can keep track of what needs to be done when for the project, we have generated a timeline shown in figure 14 that outlines important tasks and when to accomplish them within 30 days. Some tasks are optional, so those have been placed at the end—even though we may not get to them.

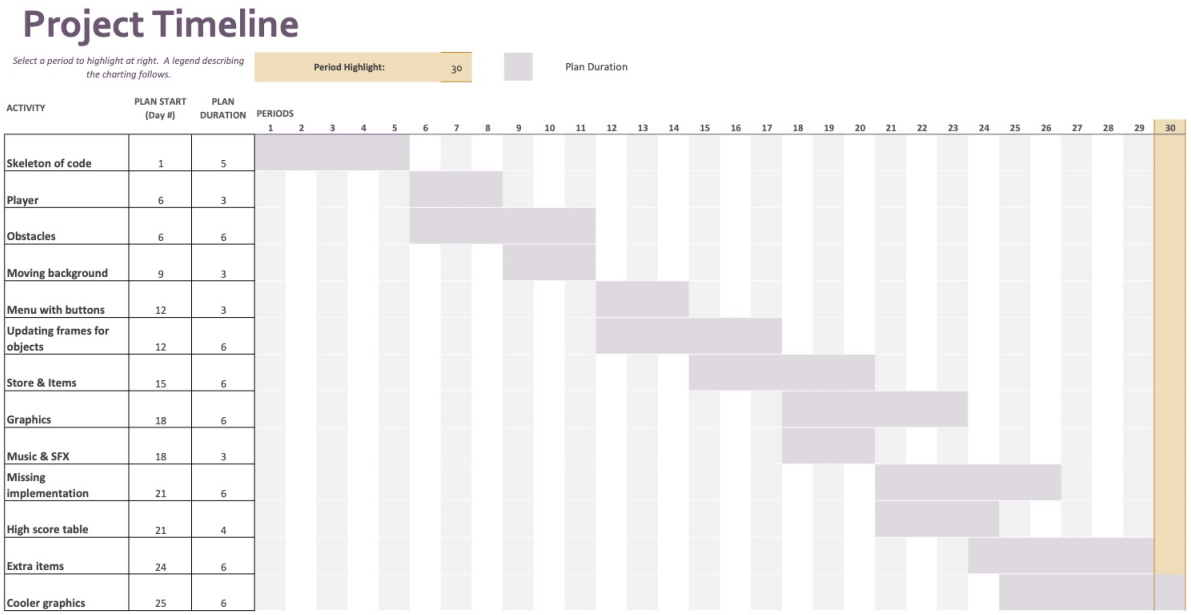

Figure 14. Timeline for the last 30 days of the project.

4. Project Structure

Our project is outlined in a UML that breaks Coding with Octocat down into classes, and it includes three design patterns that group classes together for certain tasks.

4.1. UML Outline

The UML outline shown below in figure 15 shows the structure of our program separated into classes that work together to make Coding with Octocat run. Some classes are grouped into design patterns; while, other classes do not need that functionality.

4.2. Design Patterns Used

To make Coding with Octocat functional, we are using three design patterns discussed in CSCI 352, factory method, strategy, and observer. Factory method is for generating obstacles, and strategy pattern adjusts the item appearance rate

based on the distance the player has traversed. Then, there is the observer pattern that is used to keep track of the frames, and tells the player, all obstacles, and all world objects to update position.