Leaping Llamas
for UTM CSCI 352 Spring 2017

Garrett Hay, Anderson Taylor, and Christina Hinton

Abstract

We will be creating a similar game in the vein of Flappy Bird, which we are calling "Leaping Llamas." If the initial challenge proves to be trivial, we intend to add combat elements to increase the difficulty. Players looking for a casual yet challenging game should enjoy Leaping Llamas.

1. Introduction

The goal of this project is to create a simple yet fun game that can be played by anyone on a windows computer using only a mouse and keyboard. For the type and content of the game we decided to go with a similar style to the popular game "Flappy Bird". At minimum we would like to include all of the basic functionality of the original game, so things like the auto generated level that is continuous and a high score system will definitely be incorporated into the app.

Once all of the basic functionality described above is implemented we have plans to add additional functionality including variable difficulty settings and possibly even a "bullet hell" type game mode. We're confident this will not only be an enjoyable experience for us the creators, but we also believe it will be fun for our peers to play post launch. By taking the "Flappy Bird" formula and making a more challenging experience, we think we can make a game that is relatively simple, but ultimately hard to master.

1.1. Background

"Flappy Bird" is a game where the player controls a small bird, using inputs to lift the bird in order to navigate through pipes. Gravity pushes the bird down, but inputs can easily send the bird too high.

If we reach our initial goals, we will convert Leaping Llamas into a bullet hell. A bullet hell is a game in which players must navigate through hordes of enemy projectiles without being hit. We believe it will make the game hilariously difficult and add an interesting twist to the Flappy Bird formula.

We decided to tackle this project because it's a simple, universally-recognized game. It's known for being fun and addictive. We want to make a game that's cute, funny, and addicting, and a game like Flappy Bird is a fantastic inspiration to us.

1.2. Challenges

We expect that insuring the player character's movement is smooth and responsive will be difficult. Hit boxes and fail states will be initially difficult. If we decide to have randomly generated pipes, we believe this will prove to be extremely tricky.

2. Scope

Our main goal is to make Leaping Llamas, a simple Flappy Bird-like game. If we find ourselves finished well before the due date, we will add additional mechanics to further set it apart from Flappy Bird. Firstly, we would like to add extra llama images to be used or 'skins'. Eventually bullet hell mechanics. Bullet hell mechanics are a usually extreme or hell-like amount of obstacles, usually bullet-like, that while challenging should be dodgable from somewhere on-screen. Secondly, we would like to add onto the existing game in the form of animating objects, in order of: the background, the llama, any buttons, and any obstacles like 'pipes' and/or bullet-likes.

2.1. Requirements

As part of fleshing out the scope of your requirements, you'll also need to keep in mind both your functional and non-functional requirements. These should be listed, and explained in detail as necessary. Use this area to explain how you gathered these requirements.

2.1.1. Functional.

- The game must be controlled with “Spacebar” key.
- The game must be implemented with C# WPF.
- The game must feature music and sound effects.
- The game must feature obstacle generation with several different difficulty options.
- The game must be pausable with the “P” key.
- The game must end with the “ESC” key

2.1.2. Non-Functional.

- Performance – Speed of obstacle generation.
- Scalability – The obstacle generation will be potentially unlimited obstacle generation.

2.2. Use Cases

Use Case Number: 1
Use Case Name: Start Window
Description: The user on the program starts the program. The user will choose whether to start the game, see the high scores, change the difficulty, or exit the program.
1) The user shall press ”Start” button.
Termination Outcome: The user will go to Case: 2.
Alternative: User chooses the ”Difficulty” button
1) The user shall press ”Difficulty” button.
2) The program cycles through different difficulties.
Termination Outcome: The user will go to Case: 1.
Alternative: User chooses the ”High Score” button
1) The user shall press ”High Score” button.
2) The program go to the High Score screen.
Termination Outcome: The user will go to Case: 4.
Alternative: User chooses the ”Exit” button
1) The user shall press ”Exit” button.
2) The program will end.
Termination Outcome: The user has kill the program.
Use Case Number: 2
Use Case Name: Playing *Leaping Llamas*
Description: The user on the program starts the game. The user will make the llama leap over and between the obstacles until the llama hits a obstacle. This will send the user to the scoreboard.
1) The user shall press ”Spacebar” button to ”Leap” over obstacles.
2) The game will count up by 1 for each obstacle on its counter.
3) The game will end once the user’s Llama hits an obstacle
Termination Outcome: The user has played the game portion of the program.
Use Case Number: 3
Use Case Name: Scoreboard after ’Game Over’
Description: A shopper on our site has finished shopping. They will click on a “Checkout” button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.
1) The program shall open a window with the user’s score and a ”Retry” and ”Next” buttons.
2) The user shall select ”Retry” button and proceed back to Use Case: 1 until selecting the ”Next” button.
Termination Outcome: The user will either replay the game with new score or proceed to the High Score screen.
Alternative: The user scored a High Score
1) The program shall open a window with a text box and shall prompt the user for a username and a ”Retry” and ”Next” buttons.
2) The username shall be stored along with the user’s score to a file with other High scores.
3) The user shall select ”Retry” button and proceed back to Use Case: 1 until selecting the ”Next” button.
Termination Outcome: The user will either replay the game with new score or proceed to the High Score screen.
Use Case Number: 4
Use Case Name: High Score screen

Description: A user has finished playing. The user will be shown the top 10 High Scores on the game. They will choose either to play again or return to the title screen.

1) The window shall display the High scores in nonincreasing order from the top of the window.
2) The user shall choose whether to click the ”Retry” button.
3) The program will return to the game on the same difficulty.

Termination Outcome: The user will either replay the game with new score.

Alternative: The user chooses the ”Title” button

1) The window shall display the High scores in nonincreasing order from the top of the window.
2) The user shall choose whether to click the ”Title” button.
3) The program will return to the title screen.

Termination Outcome: The user will be at the Title screen at case: 1.

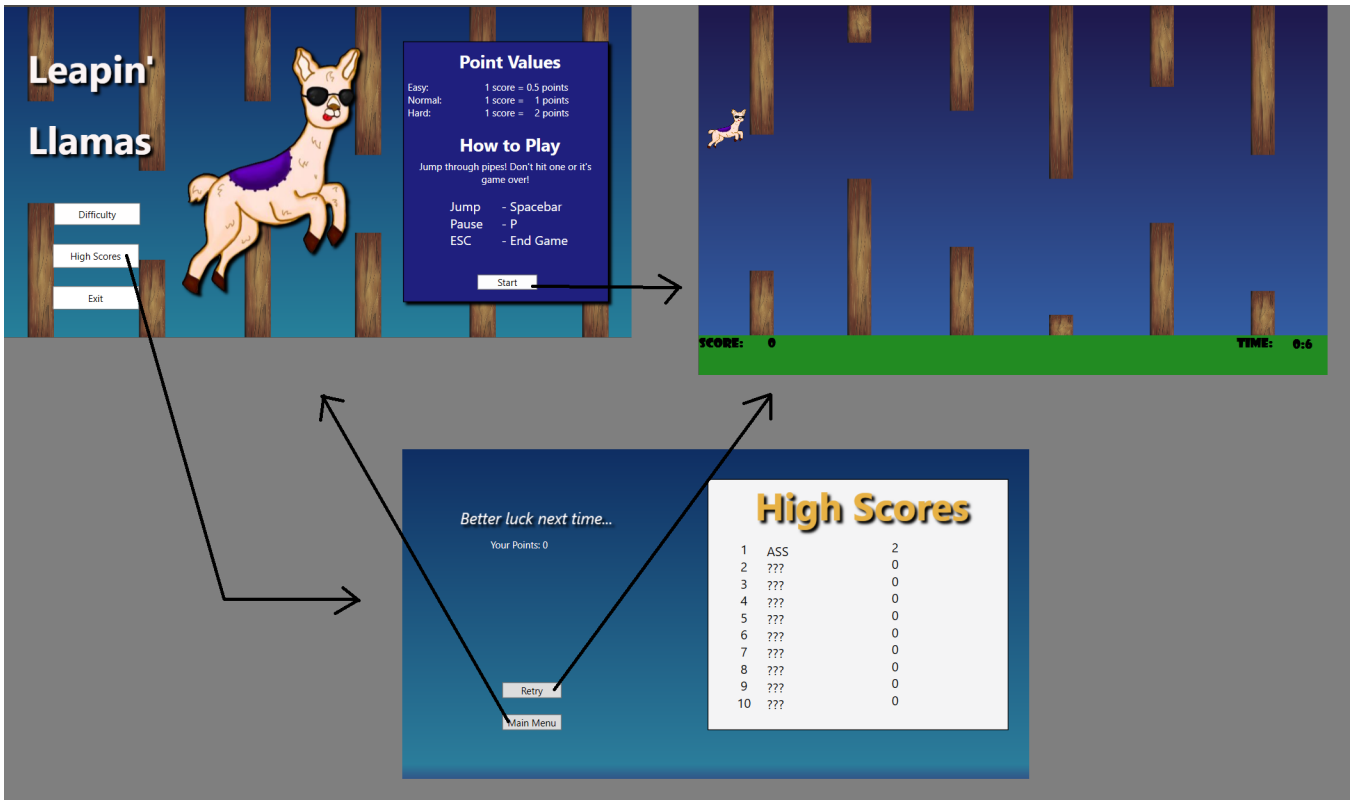

Figure 1. First picture, This is full number of Mock-Interfaces made.

2.3. Interface Mockups

For all below see Figure 1 First in order, the user will begin on the ‘Title’ screen, optionally choose from the changing difficulty button or leave it to default to ‘normal’ choose one of the button to move to the corresponding screen. Next the ‘Game’ screen will always lead to the ‘Score’ screen at the end of the game. The ‘Score’ screen has at least two choices. First, to retry and return to the ‘Game’ screen or second to exit to the ‘Title’ screen. Another choice may appear if the player has earned enough point to join the Highscore board and can enter a nickname up to nine characters in length before choosing one of the previous choices.

3. Project Timeline

See Figure 2 below.

Requirements

Begin: August 2018 End: September 2018

- The game must be controlled with "Spacebar" key or "left-click" mouse button.
- The game must be implemented with C# WPF.
- The game must feature music and sound effects.
- The game must feature obstacle generation with several different difficulty options.
- The game must be pausable with the"P" key.
- Security – user credentials must be encrypted on disk, users should be able to reset their passwords if forgotten.
- Performance – Speed of obstacle generation.
- Scalability – The obstacle generation will be potentially unlimited obstacle generation.

Design

Begin: September 2018 End: Mid-November 2018

For Leaping Llamas, we have decided to go with a cartoon art style, with a single simple easy to use menu. To make the game feel more streamlined and immersive we will attempt to use user controls instead of new windows for each different screen. This will keep everything inside of a single window, as is typical with most games.

For the gameplay design we will closely follow the "Flappy Bird Design", meaning we will have the user click to propel the llama upwards while the pipes scroll past. We plan to implement random pipe generation so that the user cannot learn any patterns in the pipes.

Finally, the code itself will use two different behavioral patterns. The first pattern we plan to use is the Strategy Pattern. We want to use the Strategy Pattern to select which difficulty setting the game uses when generating the pipes. The second behavioral pattern we intend to use is the Observer Pattern. We want to use the Observer Pattern to keep the high score list in sync with the values being shown on the screen.

Implementation

Begin: Mid-November 2018 End: Early-December 2018

Write code.

Verification

Begin: Early-December 2018 End: Mid-December 2018

Distribute to class for testing and feedback.

Maintenance

Begin: Mid-December 2018 End: Late-December 2018

Create new texture packs based on public demand.

Figure 2. Proposed timeline, This is a rough estimate of the project timeline.

4. Project Structure

We choose to make the difficulty button change to the current difficulty upon clicking it. We also choose to make the new highscore option invisible unless the amount of points earned is reached. As for the game portion we made the ‘esc’ and ‘p’ keys shortcuts to ending the game instantly and pausing the game respectfully. We also made the portion of the screen above the main screen inaccessible.

4.1. UML Outline

See Figure 3.

4.2. Design Patterns Used

Our Program will be using 2 design patterns. The first is a Decorator Pattern for selecting different skins of llama. The second is a Strategy Pattern on the different difficulties we have different fence classes for.

5. Results

Leaping Llamas started to really take shape by deliverable 5. Previously, the project was mostly conceptual in nature, however between deliverables 4 and 5 the program really took shape and is a playable, interesting user experience. As the game stands now there are options for 3 different difficulties, a ”infinitely” generated level that keeps high scores, and hit detection if the llama hits a barrier.

5.1. Future Work

Before the final submission we will implement different skins for the llama that the user can choose from. Aside from this the only thing we want to get fixed is the hitbox detection near the rear of the character. As for the far future plans for the project each member of our team will add the final version of the game to their Github.com accounts to showcase to potential employers.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

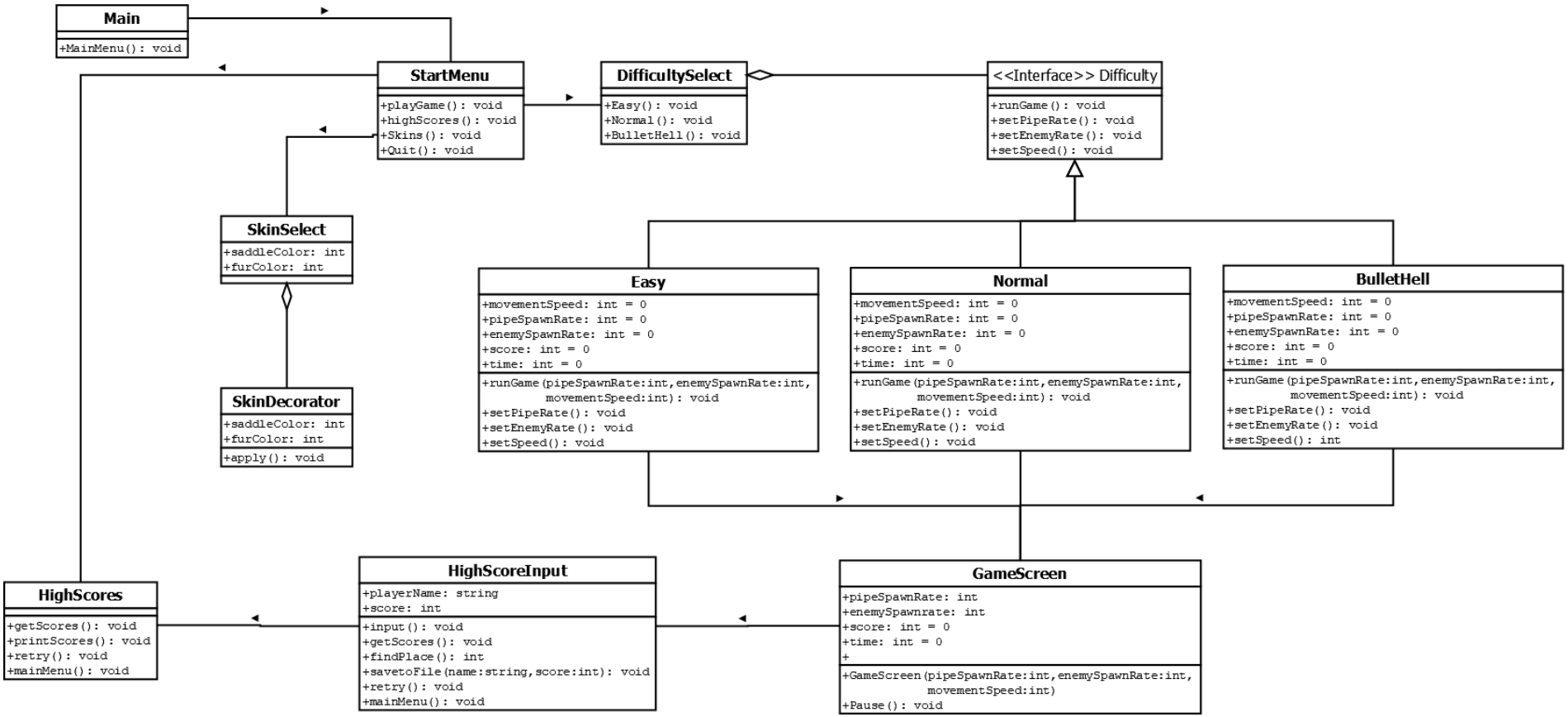

Figure 3. diagram of project using the Decorator Pattern and Strategy Pattern.