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

We are aiming to create a "Flappy Bird" game. It should have similar functionality to the original Flappy Bird. If the project proves to be simple, we plan on adding bullet hell mechanics to make the game more interesting and difficult.

We think not only will it be a fun project to create, but will be an enjoyable experience for our peers. 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 bullet hell mechanics. Secondly, we could implement a number of power-ups and checkpoints.

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 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.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Start Window	User	Simple	1
2	Choose Difficulty	User	Simple	1
3	Playing *Leaping Llamas*	User	Hard	1
4	Scoreboard after **Game Over**	System	Med	1
5	High Score Screen	System	Med	1

TABLE 1. SAMPLE USE CASE TABLE

2.1.2. Non-Functional.

- 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.

2.2. Use Cases

This subsection is arguably part of how you define your project scope (why it is in the Scope section...). In a traditional Waterfall approach, as part of your requirements gathering phase (what does the product actually *need* to do?), you will typically sit down with a user to develop use cases.

You should have a table listing all use cases discussed in the document, the ID is just the order it is listed in, the name should be indicative of what should happen, the primary actor is typically most important in an application where you may have different levels of users (think admin vs normal user), complexity is a best-guess on your part as to how hard it should be. A lower number in priority indicates that it needs to happen sooner rather than later. A sample table, or Use Case Index can be seen in Table 1.

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, or exit the program.

1) The user shall press ”Start” button.
2) The program go to the difficulty screen.

Termination Outcome: The user will go to Case: 2.

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: 6.

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.

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure **??**). NOTE: this is not a use case diagram, but a kitten.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number: 2
Use Case Name: Choose difficulty
Description: The user on the program chooses what difficulty for the game. The user will be shown the controls for the game.

1) The user shall press ”Easy” button.
2) The screen shall show the controls for the proceeding game.

Termination Outcome: The user has chosen the difficulty and will be sent to the game.

Alternative: User chooses Medium difficulty

1) The user shall press ”Medium” button.
2) The screen shall show the controls for the proceeding game.

Termination Outcome: The user has chosen the difficulty and will be sent to the game.

Alternative: User chooses Hard difficulty

1) The user shall press ”Hard” button.
2) The screen shall show the controls for the proceeding game.

Termination Outcome: The user has chosen the difficulty and will be sent to the game.

Alternative: User chooses ”Back” button

1) The user shall press ”Back” button.
2) The program shall return to the Starting screen.

Termination Outcome: The user has chosen to go back to the Case: 1 screen.

Use Case Number: 3

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: 4

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: 5

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.

2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

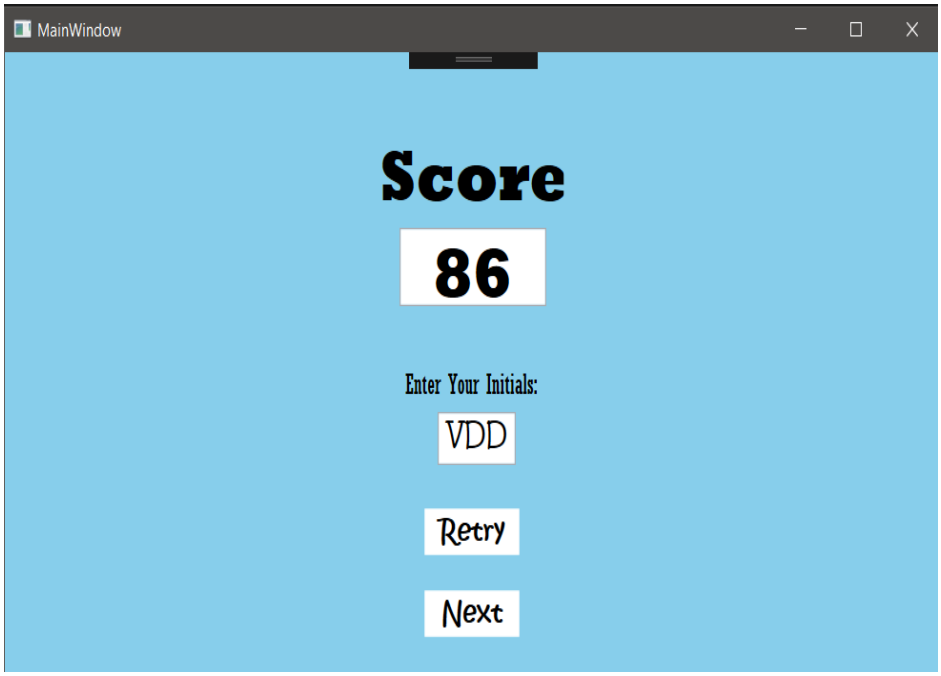

Figure 1. First picture, This is the 5_{th} Mock-Interface.

Figure 2. Second picture, This is the last Mock-Interface.

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure **??**.

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

References

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