Software Requirements Specification

for

Flappy Bird

Version 1.0 approved

Prepared by Brandon, Jeng, Nick, Travis

Class 22871 Team Dev

Created: 4/6/2020

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 References 1

2. Overall Description 1

2.1 User Classes and Characteristics 1

2.2 Operating Environment 1

2.3 Design and Implementation Constraints 1

2.4 Assumptions and Dependencies 1

3. Functional Requirements 2

3.1 User Interfaces 2

3.2 Hardware Interfaces 2

3.3 Software Interfaces 2

3.4 Communications Interfaces 2

4. System Use Cases 3

4.1 Starting the Game for the First Time (U1) 3

4.2 Deposit money into ATM (U2) 3

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 4

5.3 Security Requirements 4

5.4 Software Quality Attributes 4

6. Other Requirements 4

7. System Requirements Chart 4

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Brandon | 4/6/20 | Basic creation of classes and the game in general | 0.5 |
| Jeng | 4/12/20 | Timer added | 0.7 |
| Travis | 4/18/20 | Changed the movement system a bit, made windows same size and center to screen | 0.7.1 |
| Jeng | 4/19/20 | Timer updated and fully functional | 0.7.2 |
| Brandon | 4/27/20 | added floor as another method of defeat, changed the way the walls spawn. Updated movement system | 0.8 |
| Nick | 5/4/20 | Added sky background, changed player from a cube to the bird from Flappy Bird. Changed the enemies/walls into pipes. Ground and sky are animated | 0.9 |
| Travis | 5/5/20 | UI updates to better fit the screen, fixed issue with the enemies/walls not showing up. | 0.9.1 |
| Jeng | 5/6/20 | Added music to the game | 0.9.5 |
| Brandon | 5/6/20 | Fixed music, updated the enemy spawner so both types of pipes are able to spawn. | 0.9.6 |
| Brandon | 5/8/20 | Final version of the enemy spawn system, top and bottom pipes spawn in correct position. Added option to choose time limit of the game. Fixed many bugs | 1.0 |

# Introduction

## Purpose

The purpose of this product is to provide the user with a fun and interactive little game in which they play as a bird that must attempt to survive by dodging obstacles for a certain amount of time. The game is the only thing included in this project, so it is the whole package.

## References

The game Flappy Bird was a reference when creating the game, we attempted to recreate that game. Otherwise, there are no other references used in this project.

# Overall Description

## User Classes and Characteristics

Any type of user class will be using this program. It is a free game made for everyone so as long as someone is able to download the game then they will be able to play

## Operating Environment

We created this game using Visual Studio which is a Windows-based product and we only tested it on Windows-based system. Currently we only support the game for Windows operating systems. The hardware requirements are very minimal and almost any Windows-based computer could play it.

## Design and Implementation Constraints

None

## Assumptions and Dependencies

We assume that the user is playing on a relatively modern **Windows-based system** as we currently do not have support for other operating systems so we cannot be sure that this program will work on other operating systems.

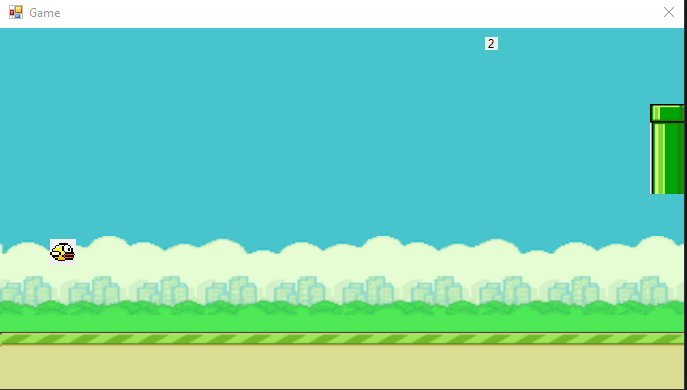
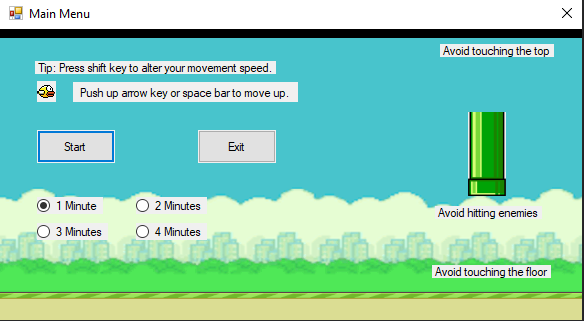
# Functional Requirements

## User Interfaces

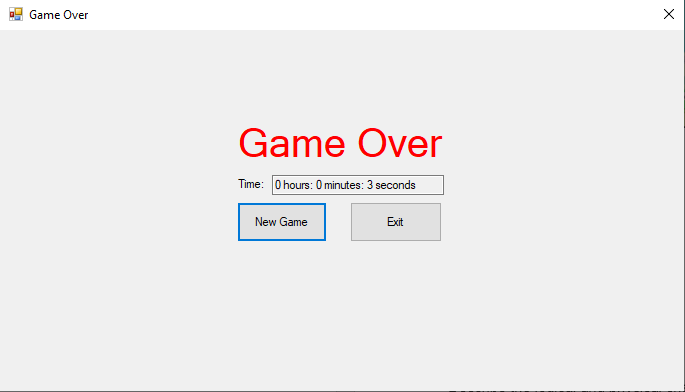
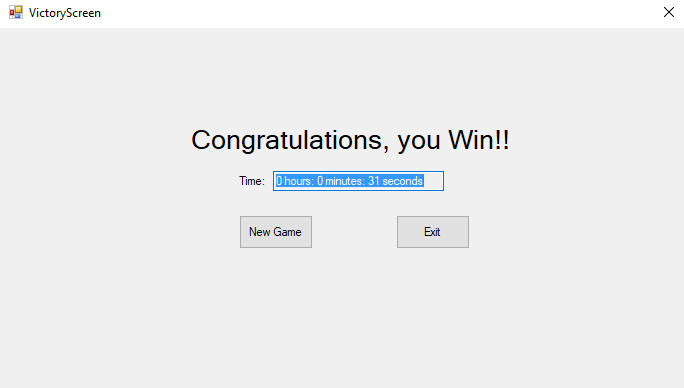
All text is in Microsoft Sans Serif font

All buttons are gray

Main Menu: Gameplay Screen:



Game Over Screen: Victory Screen:



## Hardware Interfaces

The hardware interfaces utilized with this program are the keyboard, specifically the arrow keys and spacebar, the mouse is needed to start and exit the game, and lastly the user will need a monitor in order to see the game.

## Software Interfaces

There are no outside connections to other software when using this program.

## Communications Interfaces

The program doesn’t require outside communications to occur for it to function.

# System Use Cases

## Starting the Game for the First Time (U1)

1. **U1**
2. **Objective** – The user starts up the application and wants to start playing the game
3. **Priority** – High
4. **Source** –
5. **Actors** – User, Application
6. **Flow of Events** 
   1. **Basic Flow**
      1. User opens the application to get to the main menu.
      2. User can select time options. 1, 2, 3, or 4 minute time limit.
      3. User presses the Start button to begin playing the game
      4. The game starts and the user gets to control the bird.
      5. User can dodge all obstacles and win the game after time is over.
      6. User is then able to either start another game or exit the application.
   2. **Alternative Flow 1** – At step 6.1.4 the user can’t dodge an obstacle and is defeated.
      1. User can see their time survived and can either start another game or exit app.
   3. **Exception Flow 1** – User closes the game forcefully by pressing the close button on the window.
7. **Preconditions** – User has the game installed onto their computer.
8. **Post conditions** – User was able to successfully experience the game and all of the options available.
9. **Notes/Issues** - None

# Other Nonfunctional Requirements

## Performance Requirements

See section 2.2 and 2.4 for more details.

## Safety Requirements

If you have any issue looking at computer screens have problems with slow flashing images, then consult someone before playing this game.

## Security Requirements

None

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

**In this section, just say “See section 7 requirements 55-62”. And I’ll assume those requirements are Software Quality related.**

# Other Requirements

None

# System Requirements Chart

*< Include a* ***table*** *in this section with the following columns:*

***ID*** *– Unique requirement ID*

***Priority*** *– Priority of this requirement*

***Type*** *– Functional(F) or Non-functional(NF)*

***Source*** *– Who is most interested in this requirement (John Smith – Customer). For this project you can make it up, in reality you’ll want to capture this as you capture the requirements.*

***Contained in Use Case(s****) – Which use cases reference this requirement or which use cases when executed will perform this requirement. There may be a few functional requirements without a use-case and the non-functional requirements generally will NOT be part of a use-case (so put N/A).*

***Description*** *– The description of the requirement. “The system shall …. “*

*>*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Priority** | **Type** | **Source** | **Use Case(s)** | **Description** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |