Quickfire The Game - Developer Diary

Ultan Kearns

December 11, 2019

Contents

1 Introduction **3**
- 1.1 What is this game? . . . 3
  - 1.1.1 What is a platformer? . . . 3
  - 1.1.2 Not a typical platformer . . . 4
  - 1.1.3 Aim of game . . . 4
- 1.2 About this game . . . 4
  - 1.2.1 Key points to know about this game 4
- 1.3 Experience with designer . . . 5

2 Planning **6**
- 2.1 Reading the design document . . . 6
- 2.2 Brainstorming . . . 6
- 2.3 Beginning the project . . . 7

3 Implementation **8**
- 3.1 Beginning implementation . . . 8
  - 3.1.1 Starting to code . . . 8
  - 3.1.2 Issues that Arised . . . 9
  - 3.1.3 Difficulty . . . 9
- 3.2 Achieving full implementation . . . 10

4 Methodology **12**
- 4.1 Debating Methodologies . . . 12
  - 4.1.1 Agile . . . 12

4.1.2 Waterfall 12
4.2 Why Agile? 13

5 Design **14**
5.1 Background 14
5.2 Music 14
5.2.1 Introduction 14
5.2.2 Music I chose an why 15
5.2.3 Menu Music 15
5.3 Menus 15
5.4 Game 16

6 Testing **17**
6.1 Beginning Testing 17
6.2 Testing Difficulty 17
6.3 Testing Gameplay 18

7 Conclusion **19**
7.1 What I learned 19
7.2 Effect Agile had on the project 19
7.3 Test Plan 20

Chapter 1

Introduction

1.1 What is this game?

The game is a 2D platformer called Quickfire. The object of the game is to survive as many rounds as possible while bullets come flying at the player, the bullets will get faster with each wave. At the beginning of the game the player will have a set number of lives and the object of the game is to avoid dying and to get a high score. The player will start at wave 1 and as they progress to higher waves the difficulty will increase which will be achieved by adjusting the speed of platforms and bullets accordingly.

1.1.1 What is a platformer?

A platformer is a game in which the objective of the game is to jump on platforms and reach the end of the level[1]. There are usually many obstacles which the player must face in order to advance through the level usually these are gaps, spikes or any other form of object which will either kill the player or damage his health.

1.1.2 Not a typical platformer

This game deviates from the standard platformer[1] in a very original way as it has bullets which the player must avoid in order to progress through the level. If the player is hit by a bullet or the player falls off the platform the game ends and the player's score is saved as the high-score if they have a score greater than the previous high-score or if there are no previous high-scores.

1.1.3 Aim of game

The aim of the game is to survive wave after wave of bullets and accumulate as many points as possible while also not falling off the platforms. The game also will get progressively more difficult as the player advances in order to give more of a challenge to skilled players.

1.2 About this game

This game was developed using the unity game engine and was programmed in C#. The game engine helped me a lot while developing this game and made programming the physics of the game so much easier. I started out with an empty scene and quickly I found this game took on a life of it's own. I hope you enjoyed playing it as much as I did developing it. My aim when making this game was to make an intuitive and challenging 2D platformer and to implement all the requirements as specified in the design document for this game. I found the design document to be very helpful and it was really comprehensive and included all the information I needed to make a great 2D platformer.

1.2.1 Key points to know about this game

- It's a 2D platformer

- The object of the game is to accumulate as many points as possible by surviving for a prolonged period of time and to achieve the high score
- The player must stay not fall off the platforms or this will result in the players death
- It's wave based and the difficulty increases as the wave number increases
- It was made using C# scripts and unity elements
- This game was developed using the unity engine
- This game is designed as a single player experience
- This games enemies are flying bullets which spawn randomly
- Originally the game was meant to be 20 waves but I thought that was a bit much and spoke to designer, we then agreed on 10 waves.

1.3 Experience with designer

The designer of the game was very helpful with all of my questions which were limited due to a very comprehensive design document and test plan. He was also very quick to design the game assets and send them to me promptly. I found working with the designer a great experience which will add to my skills as a developer, as dealing with clients is one of the key skills a developer can have in the modern world of software development.

Chapter 2

Planning

2.1 Reading the design document

Before I started coding I read the design document extensively and made a note of all the requirements which the game must meet. I met with the designer to discuss the key requirements of the game and discussed how they would like me to implement them.

2.2 Brainstorming

I brainstormed some ideas with the designer before I began implementing them into the game. I suggested that the background and music could change when the player reached a certain wave so that the player was informed of the increasing difficulty. I found the designer was very receptive to my ideas and welcomed them into the game. I also started brainstorming what functionality I could add to the game and which way I could implement the designer's functionality in a way that felt fluid and intuitive to the player. An example of my brainstorming is where

I decided that the player should only have a single jump as opposed to a double or triple jump to add difficulty to the game, I also added an overlap circle to detect if the player is on a platform and stipulated that they can only jump when on a platform. I found that brainstorming ideas really helped me to implement and improve the games functionality.

2.3 Beginning the project

I started this project off with brainstorming ideas of how to implement the various features that the designer had specified in the design document. I thought of how best to implement them so that the movement and game-play would seem fluid and the controls intuitive. I also brainstormed what type of features could be improved upon and discussed these with the designer. I read the design document a few times to get the general feel for the game and to ensure I understood all the requirements for the game. I began by making a simple prototype and slowly improving upon it gradually. I discussed any changes I wished to make with the designer and ensured that they agreed with the changes before I began implementing them into the game.

Chapter 3

Implementation

3.1 Beginning implementation

3.1.1 Starting to code

When I started to code the project I had many issues such as boxcolliders not working and various other problems luckily these weren't hard to solve as the unity documentation site here:Link to Unity documentation is fairly extensive. I was able to solve a vast majority of these problems by analyzing code from the labs and was able to find solutions to my specific problems within the game I'm developing. I found myself running into many problems with Unity but through determination and using their documentation as well as some Youtube tutorials I found myself becoming more and more comfortable with using Unity and I actually found that I enjoyed making these games. I used

3.1.2 Issues that Arised

When I was beginning the project I had many issues with the platforms, I had issues with getting them to move in a way that was fluid, I had issues with spawning them and I had issues where the platform would just fall. Slowly as I learned more about unity I realized that I could fix these issues simply. I started off by getting them to move by using a **rigidbody2d**[2] and a velocity to it to push it to the left. I also used the lab code to help with spawning platforms and was able to solve the falling issue by making the platforms kinematic. I also had some issues with the music restarting all the time, I solved it by adding a new script called audio manager to stop the music from repeating.
I also was finding it hard to implement screen boundaries I used lab code as a template to help me find a solution which fit the game. I added a boundary in the x axis and the y axis to ensure that the player cannot move infinitely in either axis.
I also had issues with getting the player to jump I used a tutorial to help me with this which I will cite here:[3], this added a collider to the feet of the player to check if they are on a platform, if they are they are allowede to jump.
After these initial issues were fixed it was pretty smooth sailing and I was able to fix a majority of the problems I had with ease by checking out various tutorials although I relied heavily on the Unity. Docs[4]

3.1.3 Difficulty

The difficulty of the game represented a challenge as I did not want it to be too hard or too easy, while implementing the game I tested each feature implemented to ensure that the game was difficult enough to be entertaining but not too difficult as to turn away players. I began by imple-

menting the basics of the game then refined the difficulty bit by bit and then tested it by playing the game and determining if it was too hard. The difficulty increases with each wave, the bullet speed and platform speeds both increase and for each wave the bullets increase by 10 * the number of the wave eg. wave 2 will be 2 * 10 = 20 bullets. To alert the user to the change in difficulty every 2 waves the song will change, it will progress as follows: Ode to Joy - Beethoven[5] to Ride of The Valkyries - Wagner[6] to Storm - Vivaldi[7] to Tocatta & fugue in D - Bach[8] to Flight of The Bumblebee - Korsokov[9]. I chose these classical pieces as they represent the challenge the user is faced with during the waves they are played on, for example flight of the bumblebee is known to be a fairly intense and fast classical piece with a very high BPM(beats per minute). I also chose to change the platforms based on the difficulty level so that the user gets an extra clue about the progression of difficulty. When designing the difficulty I also had to choose what was an appropriate number of lives to award the player based on the four difficulties outlined in the design document Easy,medium,hard and IRL. I found the number of lives outlined in the design document to be more than sufficient: 3 lives for easy, 2 lives for medium, 1 life for hard, and 0 for IRL. I also had to determine the appropriate spawn delay for the bullets and platforms, after much testing I decided upon delaying each by 1.3 seconds using a coroutine. I also increased the speed by a certain float for each wave passed after vigorous testing to determine an appropriate speed of course.

3.2 Achieving full implementation

In this section I will discuss how I achieved full implementation. After I had the basics of the game finished(Jumping, platform/bullet spawning etc) I began to work on the main

menu which consists of a serious of buttons the Play button loads the game scene, the options button activates the options menu which contains an option to mute the audio and adjust difficulty and a back button to navigate back to the main menu. Once I had these menus completed and tested I worked on setting the difficulty which I achieved by using the dropdown menu for difficulty and adding an event listener to change the difficulty and then retrieving the difficulty from the movement class to set the player lives. The audio was difficult to get working I made a script called AudioManager which controls which song plays on which wave as well as detects if the mute button on the options screen is checked. I vigorously tested both of these by playing through the game multiple times and checked each wave. After I got these finished I worked on bullet collisions, I used one of Unity's OnTriggerEnter to detect that the bullet collided with the player, I also chose to decrement the score that the user would've gotten had the bullet exited the screen. I also had to destroy objects(platforms and bullets) once they passed the screen boundaries, I chose to let them pass a little further than the screen boundaries before destroying them to make the game look and feel better. I destroyed them using Unity's Destroy function, they were created using a coroutine. I used playerprefs to save options such as mute and also for high scores.

Chapter 4

Methodology

4.1 Debating Methodologies

4.1.1 Agile

Agile came out on top when I was choosing methodologies for this project as it is very robust and suited to designing a game as each stage can be revisited and it's easy to write functions I needed to implement in the game and create sprints.

4.1.2 Waterfall

I debated using the waterfall methodology for this project but decided against it as each stage must be completed before moving onto the next stage and because once you finish a stage you can no longer revisit it. Needless to say this methodology did not make the cut for this project.

4.2 Why Agile?

For this project I used the Agile[10] programming methodology, I had a series of sprints[11] in which I would program functionality in the game and get a certain part of the game working. I found the Agile methodology very suited to this project as there was a lot to get done in a relatively short time-frame. I found the sprints particularly useful in terms of time management and it helped keep my other projects organised while I coded the game. The Agile methodology is also extremely popular in modern software development and is ubiquitous in industry. I thought I could use this project to gain valuable experience using the Agile methodology and to witness it's effects on productivity. Agile helped me during the course of this project by delivering an iterative approach in which I got something done in each sprint. It really motivated me to work more as I could see some of the fruits of my labour come to life at the end of each sprint. I also took advantage of Agile's ability to change during the course of development as I sometimes found myself wanting to redesign certain aspects of the game and improve upon them.

Chapter 5

Design

5.1 Background

I decided on using a single background which will continually follow the player. I wanted to convey to the player that their was a sense of urgency that must not be ignored upon reaching higher waves so I added a speed increase to the platforms and changed the platforms sprites as well as increasing the speed of the bullets as well as their number per wave.

5.2 Music

5.2.1 Introduction

Let me begin by saying that I did not write the music for this game instead I converted the audio from videos of 8 bit versions of classical pieces for this game.

5.2.2 Music I chose an why

Music is a very important aspect in many games as it can convey the mood to the player. I chose soft cheerful music for the start then as the waves increased I changed to much harsher and less comforting music so that the player would realize that they are progressing into the game and that the game will get harder the more they progress. I chose Beethoven's Ode to Joy[5]to start the game off as it is a calm and relaxing piece best known as the song the EU stole for their national anthem cause why not? For the next song to show an increase in difficulty I decided to use Wagner's Ride of the Valkyries[6] then to really get the player into the game I chose Vivaldi's Storm[7] followed Bach's Tocatta and Fugue in D[8] due to the fact these are fairly intense pieces, finally for the last wave I chose Flight of The Bumblebee as it gives off a sense of urgency due to it's high BPM(Beats per minute).

5.2.3 Menu Music

For the game over screen I used Funeral Marche by Chopin[12] to signify the loss of the player and for the main menu I chose Spring by Vivaldi[13] as it sounds nice and draws the user into the game.

5.3 Menus

The menus are fairly basic and offer all the functionality that the designer outlined in the specification. All the options are fairly straight forward as to not confuse the player before they have even started the game. I wanted to also make the menus as intuitive as possible so that the player did not have to spend ages figuring out which options were under which section. I used linear gradients for all backgrounds in the menu screen.

5.4 Game

Designing the games setting was by far the most challenging aspect of developing the game. I had to first of all implement a number of features then test if the game was engaging enough to keep a players attention. I started by making a basic version of the game with no score or waves, just a cube to represent the player and circles to represent bullets. Once I had the base game working I then decided to add in the sprites to make the game look good, after adding the sprites I then started to implement waves and scores while working on the platforms speed and bullets speed to test it while I worked on the game. In the end I wound up with a pretty good looking retro game which is engaging and, if I may say so myself, quite fun.

Chapter 6

Testing

6.1 Beginning Testing

I first started by reviewing the test plan and seeing which functionality I needed to implement. I then decided to implement the requirements as specified in the design document, once I finished the basic design of the game then I got into testing the functionality developed.

6.2 Testing Difficulty

Once I had the basic functionality for the game I began testing the difficulty. I had to ensure that the player could actually reach the platforms that spawned and that they would not spawn too far away from the player. I also had to account for the bullets and ensure that they moved at the right speed for each wave and ensure that they did not move or spawn in a way that the player had no reasonable chance to avoid them. I tested the game by giving out copies to my friends and having them report any issues back to me, I also played the game rigorously myself and

asked myself is this game difficult enough while not being too difficult? I managed to get the game to a difficulty which I felt was challenging while not deterring people from playing the game. I achieved this by messing around with platform and bullet speeds and delays for the coroutine, I kept trying until I found a level where I felt the game was challenging but not overly difficult.

6.3 Testing Gameplay

When I had accounted for the difficulty I began testing the game-play at first I played the game myself and then I enlisted the help of my friends to ask them if they thought it was too hard. I found their constructive criticism very useful and I redesigned the game based on their critiques. I wanted to ensure that the game prevented a challenge, that while not being so easy as to lose the interest of a player, would also not be too hard as to deter them from playing entirely. I decided that the gameplay was engaging and offered the user a challenging adventure through a land filled with bullets and platforms..... and one pigeon.

Chapter 7

Conclusion

7.1 What I learned

I learned a lot about game development from this project. I also learned a lot about the Unity game engine. At first I needed to learn a lot about the game engine and one of the sources that really helped me was the unity docs [4]. I also learned a lot about game and level design, as well as designing players. I learned a lot about the hidden work that goes into making a game, difficulty testing, art, music, gameplay, all of these I knew were important to a game but I did not know that they would be so difficult to test and implement. I found myself gaining a new found respect for game studios and those who work for them.

7.2 Effect Agile had on the project

I found that breaking the design into sprints was very helpful, unfortunately I was using these on post it notes on my laptop of things I had to get done so the sprints are mixed up with other projects although you can view the issues I

fixed in the game on my Github issues page for this project.

7.3 Test Plan

I have a link to the test plan here Link to Unity Project.

Bibliography

[1] Wikipedia, "Platform." https://en.wikipedia.org/wiki/Platform_game.

[2] Unity, "Rigidbody." https://docs.unity3d.com/ScriptReference/Rigidbody2D.html.

[3] BlackThornProductions, "2d double / triple jump platformer controller - easy unity tutorial." https://www.youtube.com/watch?v=QGDeafTx5ug.

[4] Unity, "Unity documentation." https://docs.unity3d.com/Manual/index.html.

[5] A. Lachniet, "Ode to joy 8 bit." https://youtu.be/H9vH85T2dQk.

[6] C. . Bit, "Ride of the valkyries 8 bit." https://youtu.be/bbTN8dqs54g.

[7] Jigoku, "Storm 8 bit." https://youtu.be/XFIRcXi--jM.

[8] DHYC, "Tocatta & fugue in d." https://www.youtube.com/watch?v=pv_XmCC6psU.

[9] 777MCore, "Flight of bumblebee 8 bit." https://www.youtube.com/watch?v=WYQWUY0zMSs.

[10] Wikipedia, "Agile software development." https://en.wikipedia.org/wiki/Agile_software_development.

[11] Yodiz, "What is a sprint." https://yodiz.com/help/what-is-sprint/.

[12] 2e8u, "Funeral marche 8 bit." https://www.youtube.com/watch?v=CODmvzK25dI.

[13] TonyDaveYayo, "Spring 8 bit." https://www.youtube.com/watch?v=Nvg-07RlpYg.