Bullet Hell

Final Report

Submitted to

The Faculty of Operation Catapult 99

Rose-Hulman Institute of Technology

Terre Haute Indiana

By

Group 34

Lewis Cook	Jakarta International School Jakarta, Indonesia
Jan Lin	American Heritage School Plantation, Florida
Daniel Olis	Ridgeview High School Orange Park, Florida
Wesley Wang	Cupertino High School Cupertino, California

June 30, 2016

Introduction

Today's video games share similar problems, high computer requirements, over-complexity which alienates new gamers, and an artificial sense of accomplishment. We sought to create a game that addresses these issues, and is a nostalgic trip back to the world of retro gaming. Through the python programming language and Pygame library we were able to create a game that fixed many of the problems of modern games, while also creating an enjoyable and fulfilling experience. The game is referred to as Catabullet as it was made for those visiting Catapult 99 Session 1.

Gameplay

Due to the obscurity and under appreciation of the "Bullet Hell" genre, we decided to try and create our own retro-inspired spin on the genre. The basic mechanics of the game involve shooting at enemy ships, and moving to dodge their shots. The user chooses a difficulty appropriate to their skill level and attempts to survive as long as they can, while shooting down enemies to increase their score.

The user is given three lives at the start and loses one each time they are hit with a bullet. However, to prevent confusing and instant game-overs, we give the user a brief period of invulnerability to reposition themselves after they are hit.

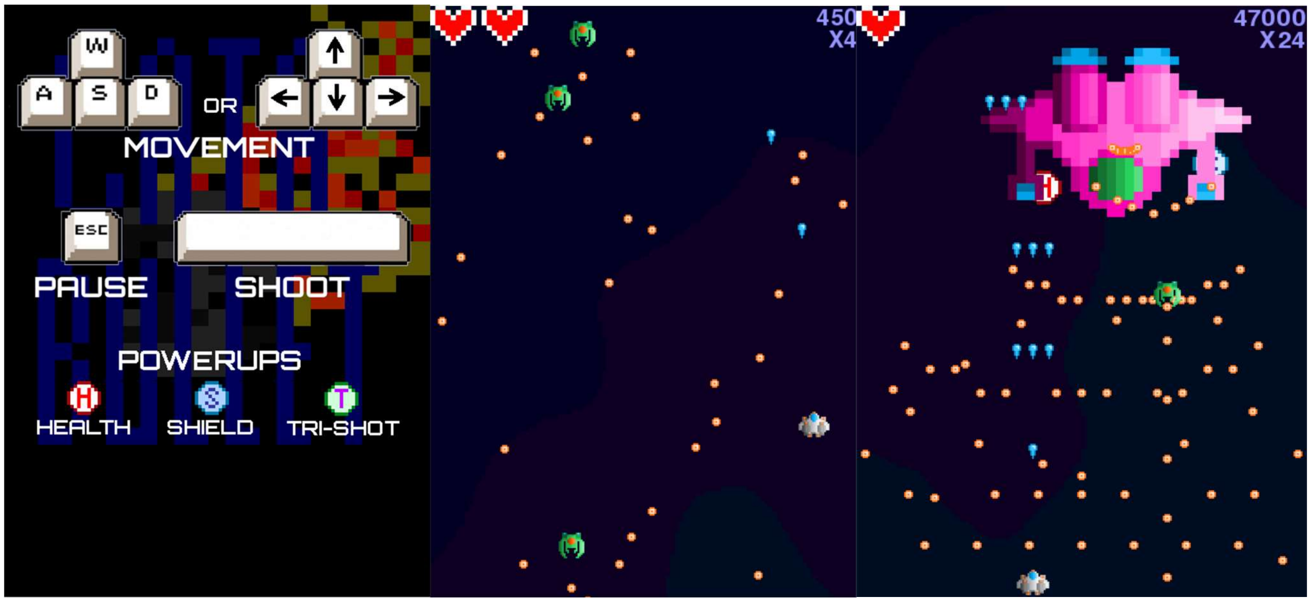


Figure 1- (Left)Instruction Screen, (Middle)Regular Gameplay, (Right)Boss Fight

To diversify gameplay, randomized enemy spawns, power-ups and "boss" enemies were added. Enemies will randomly spawn within the top portion of the screen. This allows each game to be unique. However, difficulty is kept consistent by controlling how often enemies spawn and how often they shoot. Power-ups spawn every twenty seconds and assist the user in some way whether it be a health refill, or an invincibility shield for a period of time. The power-ups add a random factor to the gameplay to stop the game from being the same experience with each play. The boss stops the user in their tracks as this difficult enemy must be defeated to continue playing.

Because the game ends when the user runs out of lives, there is always room for player improvement and thus gives incentive for users to continue playing for long periods of time as they try and best their previous scores. The scoring system rewards skilled gameplay with a multiplier that increases when the user shoots down enemies, and reverts to the original value when the user gets hit.

Python History

Python is a programming language first implemented in 1989 by Guido van Rossum at Centrum Wiskunde & Informatica(CWI). It is famously named after the TV show *Monty Python's Flying Circus* integrating many of the show's jokes into the Python examples and tutorials. Python is a general purpose object-oriented programming language, meaning programmers must define methods which affect data structures(objects). It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python is an open source, making it freely and easily distributable. Early in its development Python became part of the Computer Programming for Everyone(CP4E) initiative which. Because of its involvement with CP4E, Python evolved into an easy to learn language extensible to other languages such as C and C++. Thus, many companies utilize Python because of its flexibility. Python is a powerful open source language, many module libraries such as Pygame have been created to accompany Python and make it easier for users to accomplish a certain task.

Process/Plan

Immediately after the group was formed, it was a unanimous decision to create a video game. After deliberating the game type, it was decided that a Bullet Hell game would be created for reasons aforementioned. User stories and storyboards (see Figure Figure.2) were created to draft what the game would look like and determine how the user would interact with game

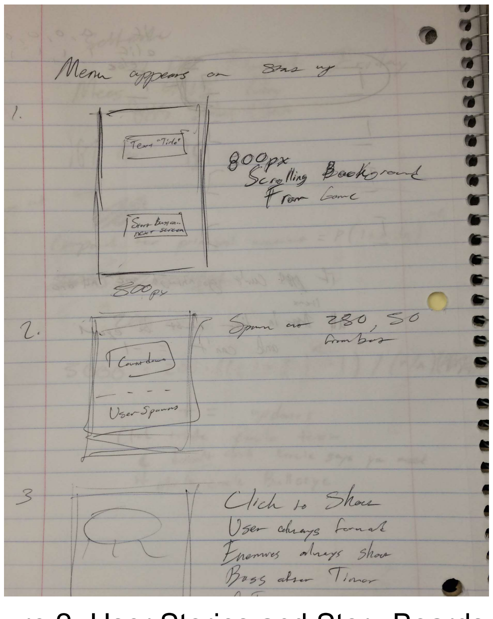


Figure 2- User Stories and Story Boards

elements.

We started planning our game with the basic mechanics and controls, such as WASD keys to move. Other movement features, such as aiming and shooting with the mouse, were removed later because of peer feedback and personal opinion. Other original ideas were changed or scrapped, such as the screen size and the game ending when you beat the boss. However most of the game and mechanics remained the same throughout the entire process and we were able to accomplish what we set out to create aside from reach goals such as networking for a multiplayer mode.

We also had elaborate plans on how we would balance the game to make it fair and fun, but not too easy. We tested many different ideas on restricting player movement and enemy spawns. However, we found out the most balanced version of our game was the simplest where the player movement was not restricted and the enemies had random spawns instead of fixated ones.

After creating the functional basics, we decided to add more features to make the game more fun. This is how the power up feature of the game came about. It easily fit into our code, which is mostly comprised of classes, and added another way to survive more easily. Ultimately, after tweaking the game and making minor changes we resulted upon the directions of gameplay shown below in Figure 3.

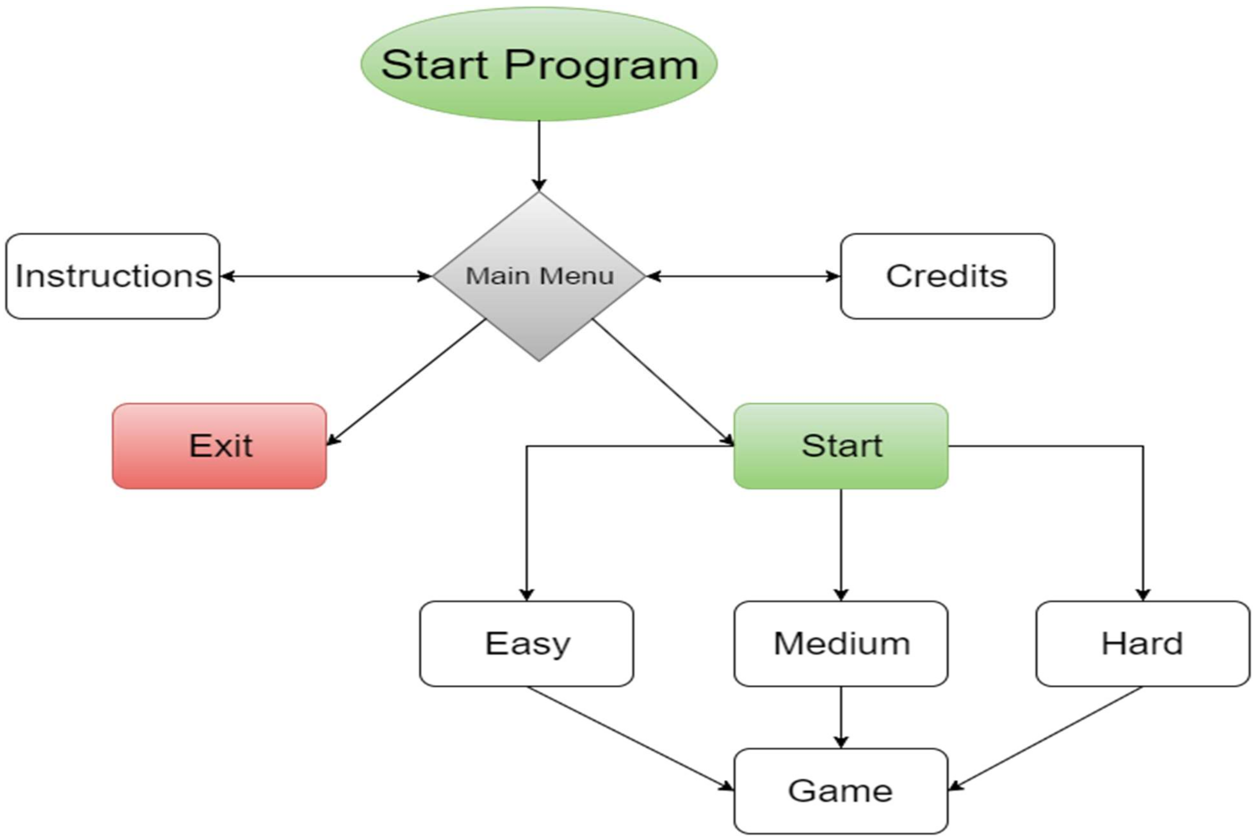


Figure 3- Flow Chart of Game Progression

Code

Our main game is based off of different timers to repeat the same event in a specified time frame. Do to their versatility, and the reduction of in game lag the timers were the perfect solution to helping our game run exactly as we want it to while making customizing the specific times is easy by just changing a specific difficulty (See Figure 4). All of the intervals are set except for the enemy spawn and enemy shooting intervals. These are set by the 3 different difficulties easy, medium, and hard.

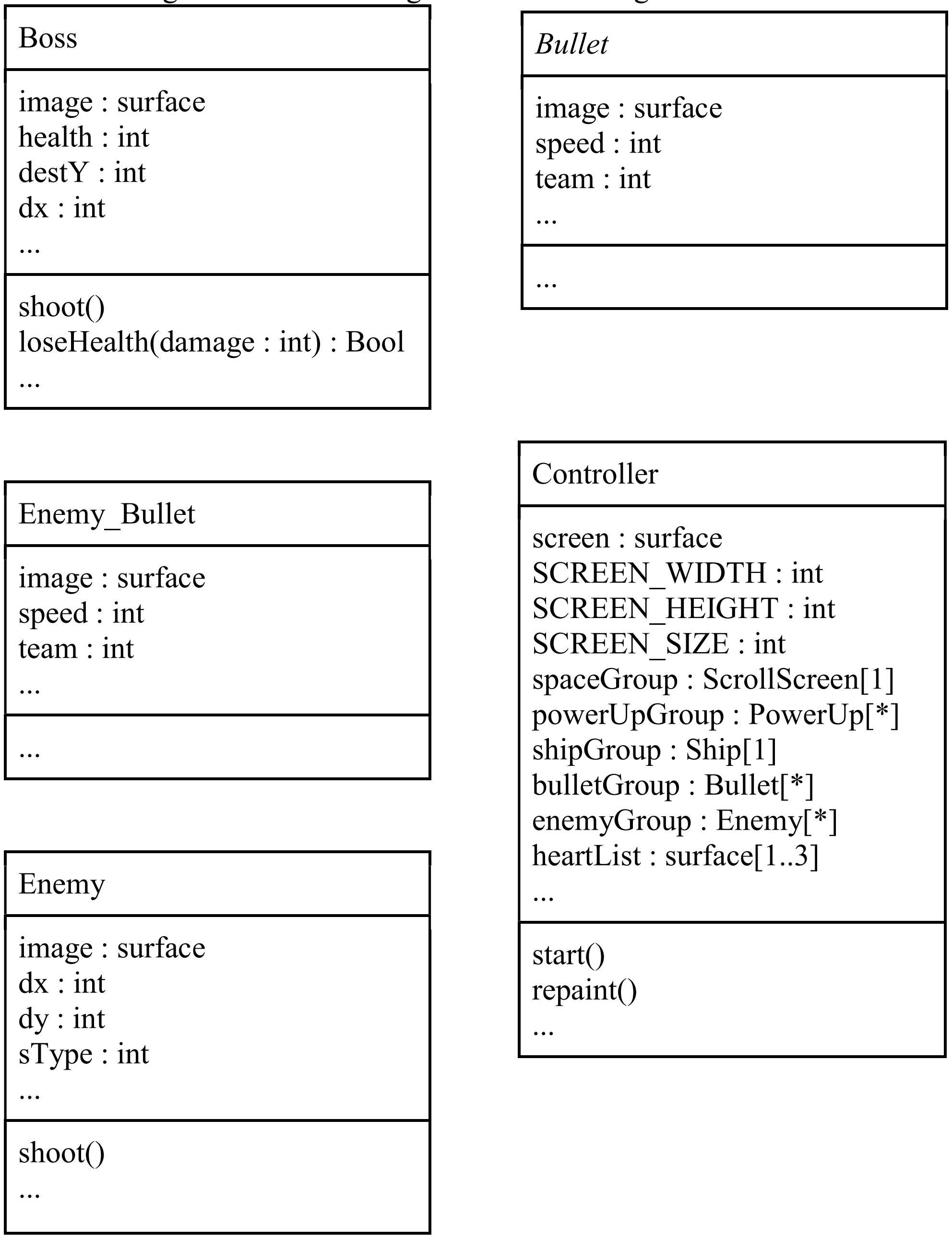
```
pygame.time.set_timer(pygame.USEREVENT+1, 50)            #Timer for bullet
pygame.time.set_timer(pygame.USEREVENT+2, 100)           #Timer for background
pygame.time.set_timer(pygame.USEREVENT+3, 300)           #Timer for shooting
pygame.time.set_timer(pygame.USEREVENT+4, 10)            #Timer for moving
pygame.time.set_timer(pygame.USEREVENT+5, self.diff)     #Timer for enemy shooting
pygame.time.set_timer(pygame.USEREVENT+6, self.diff)     #Timer for enemy spawn
pygame.time.set_timer(pygame.USEREVENT+7, 20000)         #Timer for Power Up spawn
pygame.time.set_timer(pygame.USEREVENT, 60000)           #Timer for Boss Spawn
```

Figure 4- Code for Timers

The spaceships, bullets, and power-ups are all written very similarly. They all contain similar methods and variables such as a sprite (the images its displays in the game), and update method which constantly refreshes itself on the intervals provided by the timers as stated above. We chose to use sprites instead of other shape-creating methods that Pygame provides because of the already-implemented collision methods sprites contain making it very simple to detect when a bullet hits an enemy or an enemy bullet hits the player.

UML Class Diagrams

Below in Figure 5 are UML diagrams documenting the classes contained within the program.



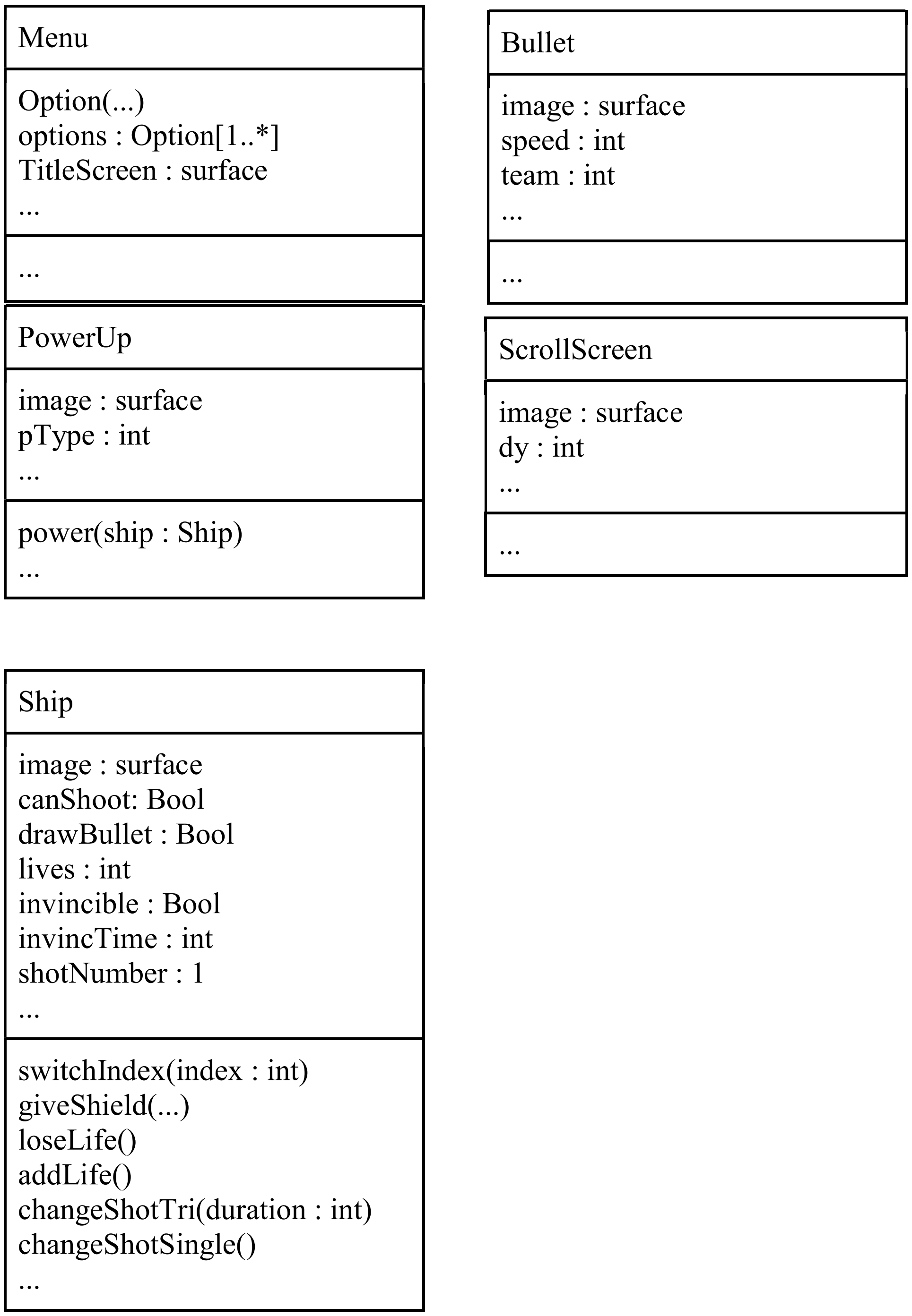


Figure 5- UML Class Diagrams

Problems

Like any large group project, we encountered several challenges along the way. Since this was the first time anyone from our group learned Python, it was troublesome for us to figure out what is possible in Python. Additionally, there were often issues with the code not interacting properly with other parts of the code. This was exacerbated because multiple people were working on the code at the same time. Although we utilized GitHub for easy code sharing, we still experienced conflicts and problems because it was the first time many of us experienced GitHub. One of the biggest issues we faced occurred after we had a working skeleton of our game. The problem was that our game lagged tremendously, causing numerous problems with the movement and animation. We overcame this hurdle by using the timers as mentioned above. The timers allowed for us to handle major events such as movement and shooting in a simple and precise way. In addition, we also faced large organizational problems throughout development. Although we did spend time working on the overview and the general outline, we did not discuss specifically how to implement our concepts and ideas. This lead to confusing code and made it very difficult to compartmentalize into individual, smaller classes.

Conclusion

After a few days of learning Python and two weeks of game design and creation, Catabullet was created. Based off of classic retro games and fitting into the modern genre of Bullet Hell games, Catabullet is a fun experience for everyone. Catering to all audiences with varying difficulties, entertaining users with a novel experience of a Bullet Hell, Catabullet is a game to be proud of.

Sources

https://www.audioblocks.com/ - Game sound effects taken from this site

https://en.wikipedia.org/wiki/History_of_Python - Used to research the history of Python

https://soundcloud.com/famusicfree/lost-in-space - Game background music taken from this site

http://www.pygame.org/docs/ - Use to access Pygame documentation

https://www.python.org/ - Used to access Python documentation