UNIVERSITY OF GLOUCESTERSHIRE

at Cheltenham and Gloucester

University of Gloucestershire

CT4030: Object-Oriented Programming & Software Design

Assignment One
Brawl Pong: A Pong Video Game

Completed Date: 1st May 2024

Table of Contents

Introduction

This report covers the design, implementation and testing phases of the self-developed video game: Brawl PONG. The simplistic pong video game involved using the computer science programming model of Object-Oriented Programming (OOP) within the IntelliJ IDEA integrated development environment. Java was the backbone behind the development of Brawl Pong along with additional packages and libraries for the graphical user interface, inputs, assets, database, etc.

Within a pong video game, commonly the main objective involves players using a paddle to deflect a ball to one another where each player tries to reach a certain number of points before the other. The creation of Brawl Pong was to understand and test and implement knowledge of OOP and how well done it would be to successfully implement it's principles within the video game. Additionally, during the design and testing phase, appropriate data representation diagrams such as Unified Modelling Language User Case, Class and Sequence diagrams will be needed along with AGILE (SCRUM) as the core Software Development Lifecycle Framework.

Using the Agile (SCRUM) Framework

Srivastava et al (2017) stated that in 2013, 82% of respondents among 5000 participants which were chosen were found to be using the Agile SCRUM framework which makes it one of the most used SDLC frameworks. The framework involves multiple iterations of building and testing throughout the entirety of the SDLC to improve the quality of the software on functionality and other features through the "combination of the Iterative Model and the Incremental Model", making it the best suited framework for efficiency and effectiveness. During the entire software development stage of Brawl Pong, it was built within a fixed schedule.

A schedule was produced as a result to successfully plan and efficiently develop the video game within an appropriate time:

15th January 2024 – 21st January 2024 ->

Learning the basics + object-oriented programming using Java Programming Language

23rd January 2024 – 18th February 2024 ->

This includes:

- Designing the game mechanics and the aim of the game
- Designing game characters + level design
- Planning gameplay + classes involved within video game
- Produce UML User case, Classes & Sequence Diagrams

20th February 2024 – 27th February 2024 ->

- Creating and setting up game development environment using Java
- Producing game assets

1st March 2024 – 15th April 2024 ->

- Coding and repetitive unit/integration testing of the video game
- Additional advanced features added

15th April 2024 – 3rd May 2024 ->

- Improve code + fix any major or minor bugs to game
- Full test of game

Designing the Video Game: Brawl Pong

To understand the pong game and its interactions in a simpler way, it was described and represented using UML. A data representation method that shows the video game's behavioural and structural pathways using classes and other object-oriented principles such as abstraction and decomposition. Figure 1 & 2 displays the UML user case and class diagrams of the pong game. These diagrams provide the simplification of the basic interactions of the entities (the player and artificial intelligence) with the user case that represents the major and some minor functionalities of Brawl Pong.

UML, User Case, Class and Sequence Diagrams

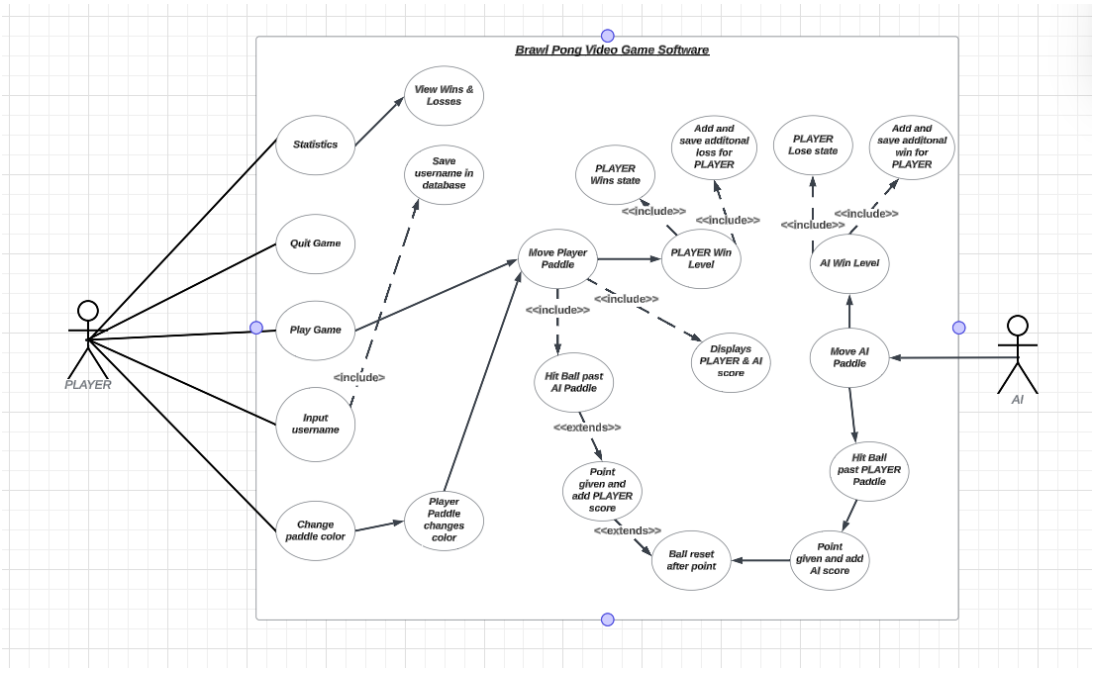


Figure 1 – UML User Case for Brawl Pong Video Game [LucidChart]

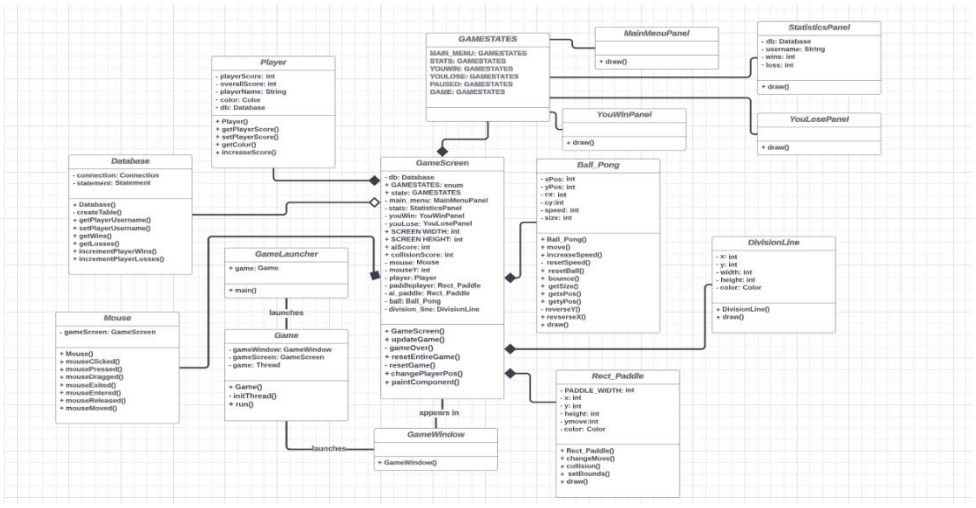
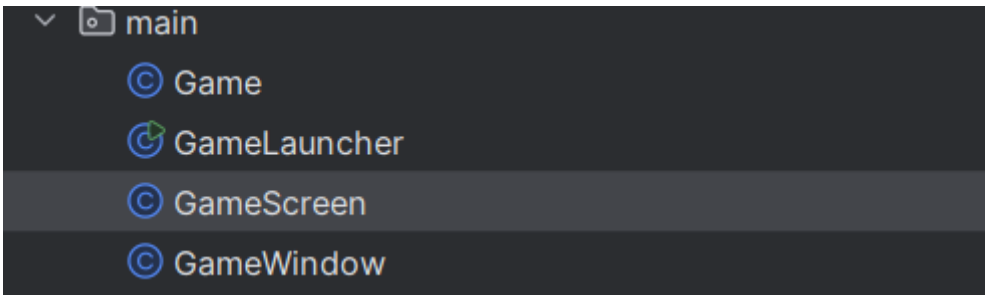
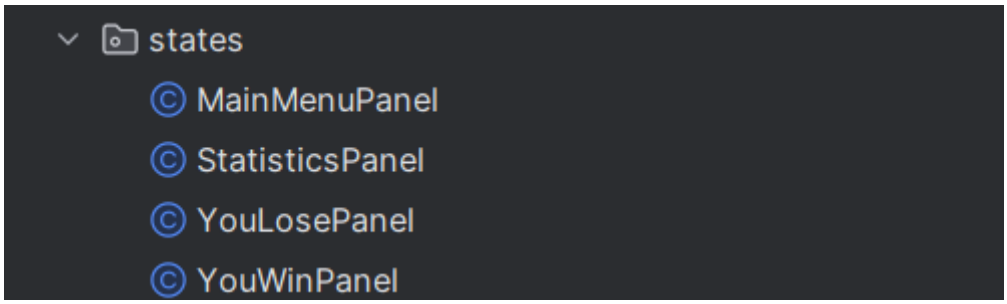


Figure 2– UML Class Diagram for Brawl Pong Video Game [LucidChart]

Within the class diagram, we have 15 classes which contribute to Brawl Pong that provides major or minor functionality. For example, the GameLauncher class acts as a launcher to open the game when running. It contains the main function to run the game which is separated into 3 other classes: Game, GameScreen and GameWindow.



The Game class is responsible for running the instances of the other 2 main video game class: GameScreen and GameWindow combined with the task to direct input to the screen, initial game thread and handling the frame rate per second of the pong game. The GameScreen class handles the entire game logic and the graphics on the screen. The GameWindow class produces the window and holds all the content of the instance of the GameScreen class. This helps simplify the several different tasks required for a video game to run while providing great code organisation and security for attributes and functions within a single class.




```
public void draw(Graphics g)  1 usage
```

Apart from these “main” classes, additional classes were implemented such as the “states” classes which are the classes responsible for the different game states within Brawl Pong. The instances of all these classes were all initiated and assigned within the GameScreen class but were linked by the Enum GAMESTATES containing several states to represent the instances of the “states” classes. As an addition, the usage of polymorphism is implemented heavily when drawing out graphics on the screen. Each “states” class each contain a “draw” function which draws different shapes and text however when called within the GameScreen class using their instances, calling their “draw” functions. This enables efficient code reusability, using the called similar functions of instances of a class while achieving different results.

The majority of the attributes in different classes are private while the majority of the functions are public. This is because several classes will use the instances of other classes and call out their functions to accomplish specific tasks within their class such as the GameWindow using the GameScreen class updateGame() function.

Video Game Requirements

Functional & Optional Requirements

Gilligan et al (2006) considers potential design requirements necessary to develop a playable pong game with a single switch device. In this instance, the mouse would solely act as the switch device required to move the paddle on the screen along with additional inputs such as navigation throughout the pong game.

1. *Game must have input to navigate throughout game for player & move player paddle [Functional] [Must]*
2. *The ball must move and bounce off the player and ai paddles, the top and bottom edges of the screen [Functional] [Must]*
3. *Game must have a suitable game logic for the player to win or lose during gameplay [Functional] [Must]*
4. *Game must provide points to player or AI during gameplay [Functional] [Must]*
5. *Game should display and change player and AI scores during gameplay [Functional] [Should]*
6. *AI paddle should be able to predict the position of the ball and move [Functional] [Should]*
7. *Game should change states such as restarting, pausing and exiting game [Optional] [Should]*
8. *Game should allow player username to be saved in database & seen in-game [Optional] [Should]*
9. *Game should allow player wins and losses to be saved in database & seen in-game [Optional] [Should]*
10. *Game could allow players to see their wins and losses [Optional] [Could]*

Other video game software requirements could be considered however the listed requirements are significant when developing Brawl Pong as they are the basic requirements behind a simple pong game.

Testing the video game: Brawl Pong

Test Case One: Brawl Pong [19th March]

Test Number	***Description of Test***	***Inputs required***	***Expected Results***	***Actual Results***	***Status***
1.	The player can play the game from the main menu when they click "Play Game"	Mouse clicked	When "Play Game" is clicked then the pong game begins	The screen was black for 5 seconds then the pong game began	FAIL. It was due to a timer related issue
2.	The player can use the mouse to move the paddle	Mouse clicked	During the gameplay, the paddle moves to the y-position of the mouse	The paddle successfully followed the y-position of the mouse	PASS. The movement of the paddle could be smoother.
3.	The pong ball can move and bounce of both paddles, the top and bottom of screen	None	During gameplay, the ball moves randomly and deflects of the paddles and the top and bottom of screen.	The ball moves randomly quickly and deflects the paddles and the edges of the screen.	PASS. The y and x velocity of the ball should be reduced and should

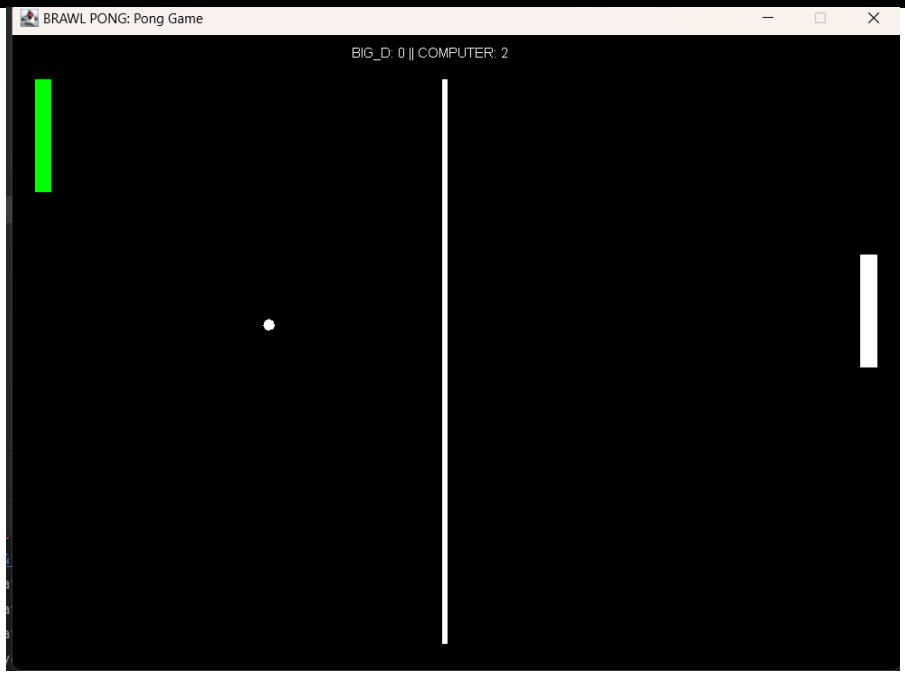
					increase graduall y
4.	The game should provide points to the player and ai & display them	None	During gameplay, if the player and the AI should get points and points should be changed and displayed	Nothing showed up on screen.	FAIL. It was not drawn in.

Test Case Two: Brawl Pong [25th April]

Test Number	***Description of Test***	***Inputs required***	***Expected Results***	***Actual Results***	***Status***
1.	The player can play the game from the main menu when they click “Play Game”	Mouse clicked	When “Play Game” is clicked then the pong game begins	The game instantly starts	PASS. The colour of paddle that player changes show along with username from database
2.	The player can use the mouse to move the paddle	Mouse clicked	During the gameplay, the paddle moves to the y-position of the mouse	The paddle successfull y followed the y-position of the mouse with smoother movemen t	PASS. Fix paddle movement when mouse fixed at a single y-position (idle)

3.	The pong ball can move and bounce of both paddles, the top and bottom of screen	None	During gameplay, the ball moves randomly and deflects of the paddles and the top and bottom of screen.	The ball moves randomly and deflects the paddles and the edges of the screen.	PASS. Slight issues when ball goes to the bottom + hits edges of the paddle
5.	The game should provide points to the player and ai & display them	None	During gameplay, if the player and the AI should get points and points should be changed and displayed	The scores show and incremen t.	PASS. Could try and save these scores

```
[SETTINGS] Connected to Database
[SETTINGS] Connected to Database
[SETTINGS] Connected to Database
[IN-GAME] Which color do you want to use? Red? Blue? White? Yellow? Green? Green
[SETTINGS] Frames Per Second: 119 FPS
```



Conclusion

Overall, many of the requirements were fulfilled and successfully performed in-game. This simplistic pong video game has the potential to become more efficient in terms of performing tasks with better programming and more visually appealing to players who play it however it could be due to the lack of knowledge to implement additional efficient and effective programming techniques, external libraries into the game.

This can be dealt through practice and experience, eventually leading the pong game to be updated and achieving its peak. Another issue which arose was the mishandling of connecting to the SQLite database constantly.

At the end, the game logic was successfully implemented, and the full functionalities of the pong game achieved. This means that it is playable as if it was any other video game.

Reference List

- Mac Namee, Brian & Gilligan, John. (2006). Interface Design Requirements For Playing Pong With A Single Switch Device.

- Srivastava, Apoorva & Bhardwaj, Sukriti & Saraswat, Shipra. (2017). SCRUM model for agile methodology. 864-869. 10.1109/CCAA.2017.8229928.