



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING PULCHOWK CAMPUS

A project on

PING PONG

APPLICATION OF OBJECT ORIENTED PROGRAMMING USING C++

BY:

Samip Ghimire(078BCT075)

Saugat Adhikari(078BCT081)

Shashank Bhatta(078BCT084)

To : **DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING**
LALITPUR, NEPAL

August, 2023

ACKNOWLEDGEMENT

First of all, we would like to express our sincere thanks to our lecturer DayaSagar Baral for his insightful lectures, constant guidance and helpful encouragement.

We would like to thank the Department of Electronics and Computer Engineering at the Institute of Engineering's Pulchowk campus for providing an opportunity to collaborate, which helped us to implement the knowledge as a second-year project, and to develop a project of our own that leveraged our knowledge expand significantly and give us a new experience of teamwork.

We are also grateful to our distinguished seniors who have helped us. They stand by us with their knowledge, experience and suggestions. We would also like to thank all our friends who supported us directly and indirectly in the realization of this project. Last but not least, our deep appreciation goes to our family members, who have been a constant source of inspiration for us.

Any kind of suggestion or criticism is greatly appreciated and acknowledged.

Authors:

Samip Ghimire (078BCT075)

Saugat Adhikari (078BCT081)

Shashank Bhatta (078BCT084)

ABSTRACT

The main aim of this project was to develop a game program using an Object-Oriented Programming language, C++. For this project, we made a classic two player Ping Pong game using Simple DirectMedia Layer (SDL) for graphical interface. The goal of creating this game is also to learn about game development.

Retro Ping Pong: Nostalgia Match, takes players on a journey back in time with its pixelated graphics and classic arcade gameplay. Channel your inner paddle master as you engage in intense matches against the computer, relishing the simplicity of the past. With its vintage charm and addictive mechanics, this game serves up a smashing good time for players of all ages.

Keywords: Ping Pong, SDL, OOP, C++

Table of Contents

| | |
|--|-----------|
| 1. OBJECTIVES | 1 |
| 2. INTRODUCTION | 2 |
| 3. APPLICATION | 9 |
| 4. LITERATURE SURVEY | 11 |
| 5. EXISTING SYSTEM | 12 |
| 6. METHODOLOGY | 13 |
| 6.1. Initiation and Planning: | 13 |
| 6.2. Algorithm Design: | 13 |
| 6.3. Software Design: | 13 |
| 6.4. Testing and Debugging: | 13 |
| 7. IMPLEMENTATION | 14 |
| 7.1. System Block Diagram | 14 |
| 7.2. Gameplay Overview | 14 |
| 7.3. Game Interface | 15 |
| 7.4. Paddle and Ball Mechanics | 15 |
| 7.5. Gameplay Flow | 15 |
| 7.6. Winning Conditions | 15 |
| 8. RESULTS | 15 |
| 8.1. Game Start | 16 |
| 8.2. In-Game Play | 16 |
| 8.3. Score Update | 16 |
| 8.4. Victory Celebration | 17 |
| 8.5. Post-Match Options | 17 |
| 9. PROBLEMS FACED AND SOLUTIONS | 18 |
| 9.1. Sorting Out Code | 18 |
| 9.2. The Ghost Ball Mystery | 18 |
| 9.3. Flickering Screen | 18 |
| 9.4. Ball Speed Fluctuation | 18 |
| 10. LIMITATIONS AND FUTURE ENHANCEMENTS | 19 |
| 10.1. Limitations | 19 |
| 10.2. Future Enhancements | 19 |
| 11. CONCLUSION AND RECOMMENDATIONS | 21 |
| 12. REFERENCES | 22 |
| 12.1. Book References: | 22 |
| 12.2. Web References: | 22 |

1. OBJECTIVES

The main goal of this project is not to become expert game developers in a short time. Instead, we want to learn how to use a special way of making computer programs called "object-oriented programming." We'll do this by creating a simple game using a programming language called C++. This language is better for making games that work really fast and can run on many different types of computers.

Here are the things we want to achieve with this project:

- **Learn How to Use Object-Oriented Programming:** We want to understand a special way of writing code that makes it easier to create programs. We'll use a programming language called C++, which is good for making things work well and fast on different computers.
- **Explore C++ Basics:** We'll learn about the basic things that make up the C++ language, helping us become familiar with its tools and capabilities.
- **Make Code Reusable:** We'll discover how to create our own pieces of code that we can use again and again, making our work more organized and efficient.
- **Use Software Library:** We'll use a library called SDL2 to help us with making our game. This library provides tools to work with graphics and sound in our programs.
- **Improve Coding Efficiency:** We'll find out how to write code that works well and doesn't waste time or space.
- **Understand Game Development:** We'll get a basic idea of how games are made and how they work.
- **Work as a Team:** This project will teach us how to cooperate with others, communicate effectively, and solve problems together.

So, by working on this project, we'll learn how to write code in a smart way, make a simple game, and work together as a team.

2. INTRODUCTION

The project revolves around the creation of a classic Ping Pong game, a two-player arcade-style game where players control paddles to hit a ball back and forth. The game aims to provide a nostalgic gaming experience while incorporating modern programming practices.

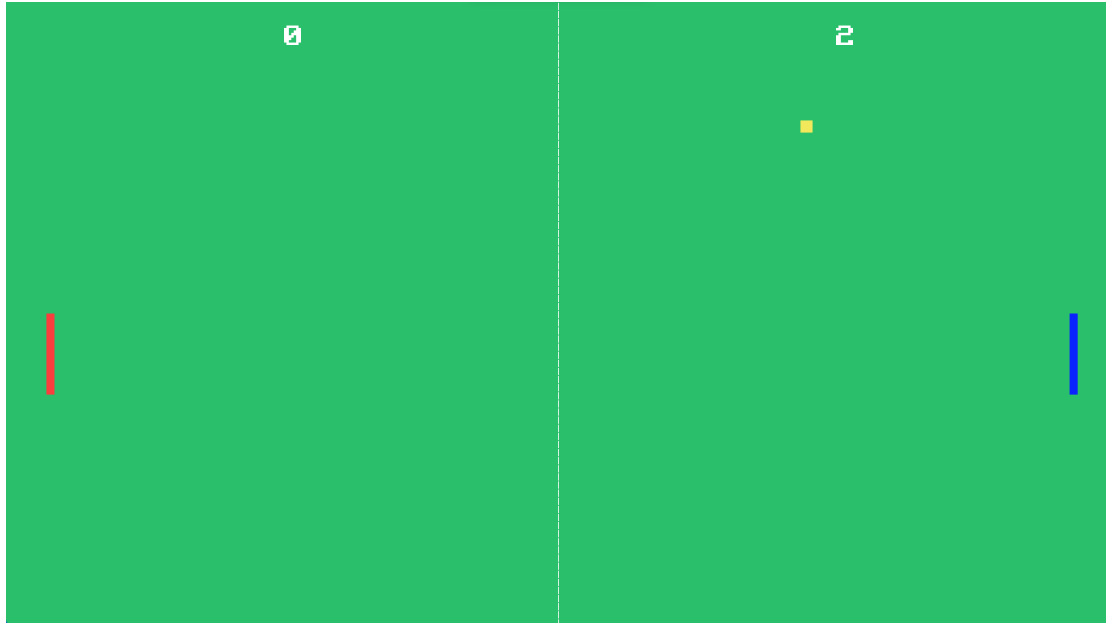


Fig 2.1: Ping Pong Game

In the Ping Pong game, players have the option to decide the score they need to reach in order to win.

Theoretical Background on Object Oriented Programming, C++ and SDL2

2.1 Object-Oriented Programming (OOP):

Object-Oriented Programming is a programming paradigm that focuses on organizing code into reusable units called "objects." Objects encapsulate both data (attributes) and behavior (methods), promoting modular and efficient code development. In OOP, the emphasis is on modeling

real-world entities as classes, which serve as blueprints for creating objects with shared properties and functionalities. Concepts like inheritance, encapsulation, and polymorphism are fundamental to OOP and contribute to code organization, reusability, and maintainability.

2.2 C++ Programming Language:

C++ is a versatile and powerful programming language that builds upon the foundation of the C programming language. It offers features like classes, objects, and other OOP principles, making it conducive to application and game development. C++ provides a balance between high-level abstractions and low-level memory control, enabling developers to create efficient and optimized code. Its wide range of libraries and extensive community support makes C++ a suitable choice for diverse programming projects.

2.3 SDL2 Library:

The Simple DirectMedia Layer (SDL2) is a cross-platform software development library designed to facilitate multimedia programming, including graphics, audio, and user input. SDL2 abstracts complex platform-specific functionalities, providing a consistent and user-friendly API for developers. This library is particularly valuable for game development, as it simplifies tasks such as rendering graphics, handling user inputs, and managing sound effects. Its platform independence ensures that applications created using SDL2 can run seamlessly on various operating systems.

Incorporating a solid grasp of these theoretical underpinnings will serve as a springboard for the practical application of OOP principles, C++ programming, and SDL2 integration in the subsequent stages of the Ping Pong game development.

2.4 OBJECT ORIENTED FEATURES

The primary objective of the project was to learn object-oriented programming concepts by practically implementing them. So, we have tried

our best to implement the features of object-oriented programming in our project.

2.4.1 Objects and Classes

As a basic building block of OOP, it is common to include these concepts in our program. Since game elements can be treated as objects, these are the classes used in Program:

- a. Vec2 Class:
 - This class represents a 2D vector with x and y components.
 - It's used to manage positions, velocities, and other 2D quantities in the game.
 - Objects of this class are created to define positions and velocities for paddles and the ball.
- b. Paddle Class:
 - This class represents a player's paddle in the game.
 - It encapsulates the paddle's position, velocity, and graphical representation.
 - It has methods to update the paddle's position, check for movement limits, and draw the paddle on the screen.
 - Objects of this class are created for both player one and player two paddles.
- c. Ball Class:
 - This class represents the game ball.
 - Similar to the Paddle class, it holds information about the ball's position, velocity, and graphical representation.
 - It includes methods to update the ball's position and draw the ball on the screen.
 - It also contains methods to handle collisions with paddles and walls and update the ball's behavior accordingly.
- d. PlayerScore Class:
 - This class manages the score display for each player.
 - It encapsulates the rendering of the player's score using SDL rendering functions.
 - The SetScore method is used to update the displayed score.
 - Objects of this class are created for both player one and player two scores.

- e. CheckPaddleCollision Function:
 - This function checks if a collision occurred between the ball and a paddle.
 - It takes a Ball object and a Paddle object as parameters and returns a Contact object indicating the collision type and penetration.
- f. CheckWallCollision Function:
 - Similar to CheckPaddleCollision, this function checks for collisions with the walls of the game window.
 - It takes a Ball object as a parameter and returns a Contact object indicating the collision type and penetration.

2.4.2 Abstraction

Abstraction is a fundamental concept in object-oriented programming that involves simplifying complex reality by modeling classes and objects that represent essential characteristics and behaviors while hiding unnecessary details. In our Ping Pong game code, abstraction is used to create higher-level structures that manage various game entities and their interactions while concealing the intricate implementation details. Let's explore how abstraction is used in our code:

- a. Classes as Abstractions:
 - Each class we've defined represents an abstraction. For example, the Paddle class abstracts the concept of a player's paddle in the game. It encapsulates properties like position, velocity, and drawing, hiding the underlying calculations and operations required to achieve those behaviors.
- b. Vec2 Class:
 - The Vec2 class is a simple abstraction of a 2D vector. It encapsulates the x and y components, allowing us to perform vector operations like addition and multiplication without exposing the raw arithmetic.
- c. Paddle Class:
 - The Paddle class abstracts the concept of a paddle. It provides methods to update the paddle's position, handle collisions, and draw the paddle. Users of the Paddle class interact with these methods without needing to know the specifics of how the paddle's movement or collisions are implemented.
- d. Ball Class:
 - Similarly, the Ball class abstracts the ball's behavior. It hides the collision calculations and the interactions between the ball and the

paddles or walls. Users of the Ball class only need to interact with its methods like Update and Draw, without needing to understand the underlying collision logic.

e. **PlayerScore Class:**

- The PlayerScore class abstracts the display of player scores. It encapsulates the rendering of score text and manages updates to the score display. This abstraction shields the user from the complexity of SDL rendering functions and text manipulation.

f. **Functions for Collision Detection:**

- The CheckPaddleCollision and CheckWallCollision functions provide abstractions for collision detection. These functions encapsulate the collision checks, returning a Contact object that abstracts the collision type and penetration depth. Users can determine the collision type without needing to understand the detailed checks.

2.4.3 Encapsulation and Data Hiding

In our Ping Pong game project, we have employed the principles of encapsulation and data hiding to enhance the organization, security, and maintainability of our codebase. These concepts allow us to create well-structured classes that encapsulate both data and the methods that operate on that data. By doing so, we ensure that our game's internal workings are shielded from direct external access, promoting a controlled and reliable interaction with our game components.

A. **Encapsulation:**

Encapsulation serves as a cornerstone of our code design, empowering us to create classes that encapsulate the essential attributes and behaviors of various game elements. Each class acts as a self-contained unit, bundling together related functionalities. For instance, in our **Paddle** class, we encapsulate the paddle's position, velocity, and methods for updating and rendering. This encapsulation not only makes our code more modular but also simplifies how we interact with different aspects of the game.

B. **Data Hiding:**

Data hiding is a vital aspect of encapsulation that enables us to protect our class's internal data from unauthorized access and modification. By designating certain properties as private, we prevent direct manipulation from outside the class. Instead, we provide controlled access through methods, ensuring that any data modifications adhere to specific rules and validations. In our **PlayerScore** class, we hide the rendering components

like surface, texture, and rect, exposing only the SetScore method to update the score. This approach maintains the integrity of our data and avoids unintended interference.

2.4.4 Operator Overloading

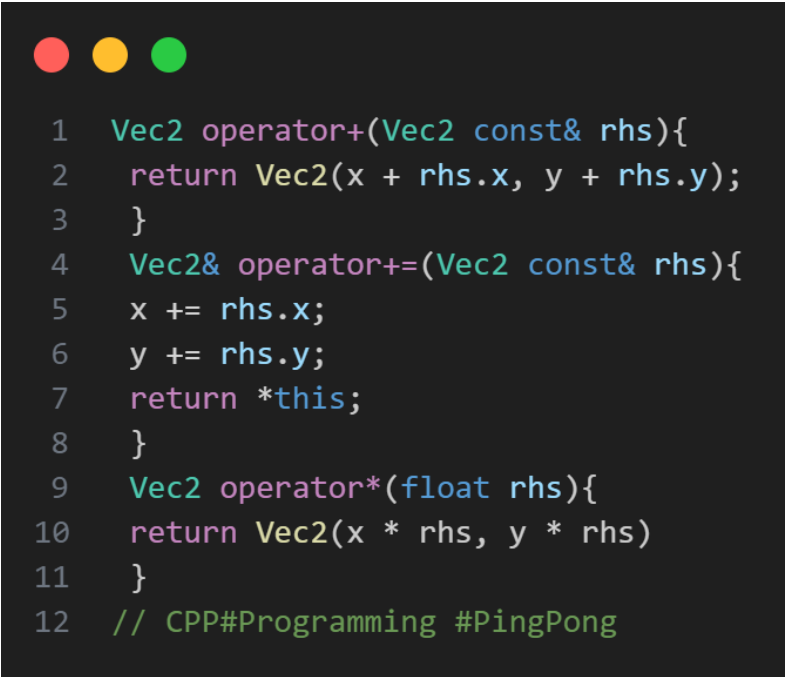
Operator overloading is a powerful feature in C++ that enables custom behavior for operators when applied to user-defined classes. In our Ping Pong game project, we have utilized operator overloading to enhance the expressiveness and readability of our code, specifically within the Vec2 class.

Operators Overloaded:

+ Operator: We have overloaded the + operator to perform vector addition for instances of the Vec2 class. This allows us to add two vectors together in a more intuitive and concise manner.

- Operator: Similar to the + operator, we have also overloaded the - operator to enable vector subtraction between Vec2 instances.

*** Operator (Scalar Multiplication):** The * operator has been overloaded to facilitate scalar multiplication for Vec2 objects. This allows us to scale vectors by a scalar value effortlessly.



```
1  Vec2 operator+(Vec2 const& rhs){
2      return Vec2(x + rhs.x, y + rhs.y);
3  }
4  Vec2& operator+=(Vec2 const& rhs){
5      x += rhs.x;
6      y += rhs.y;
7      return *this;
8  }
9  Vec2 operator*(float rhs){
10     return Vec2(x * rhs, y * rhs)
11 }
12 // CPP#Programming #PingPong
```

2.4.5 Polymorphism

Polymorphism is another important feature of OOP. It allows different objects to respond to same operation in different ways. The different ways of using same function or operator depending on what they are operating on is called polymorphism. In C++, polymorphism is mainly divided into two types:

1. Compile time polymorphism
2. Run time polymorphism

Pure Virtual Functions: Pure virtual function is a virtual function that only has a declaration but doesn't have a definition. Since they have no definition, these functions cannot be called and object consisting of pure virtual function cannot be created. Its usefulness comes from the fact that any class that derives from a base class consisting of a pure virtual function must implement the function for the derived class.

Abstract Base Class: A class that has a pure virtual function is an Abstract Base Class. These classes cannot be used to instantiate an object but serve the following function.

3. APPLICATION

The Ping Pong Game is more than just a recreational activity; it holds educational value and offers an engaging experience for the general public. This section explores how the Ping Pong Game application can be enjoyed and appreciated by people of all ages.

3.1 Fun Escape:

Ping Pong Game offers pure entertainment for people looking to take a break. Its addictive gameplay lets players unwind and have fun, whether alone or competing with friends.

3.2 Easy Start:

With simple controls and mechanics, Ping Pong Game is suitable for everyone. Even if you're new to gaming, its user-friendly design lets you quickly grasp the rules and dive into the gaming world.

3.3 Learning through Play:

Beyond fun, Ping Pong Game teaches physics concepts like angles and collision. As you strategize to bounce the ball and react to its movements, you're actually engaging with motion principles, making learning enjoyable.

3.4 Quick Reflexes:

Playing Ping Pong Game sharpens coordination and reflexes. Precise timing between hand and eye movements is crucial, improving cognitive skills that apply in real-life situations.

3.5 Friendly Connections:

Multiplayer mode encourages bonding. Friends and family can have friendly matches, fostering shared experiences and laughter while building relationships.

3.6 Relaxation Break:

Playing Ping Pong Game offers stress relief. The game distracts from daily worries, giving a mindful escape. The rhythmic ball motion induces relaxation and mental rejuvenation.

3.7 Tailored Experience:

Personalize the game by setting your winning score. Empowerment to customize the game adds a sense of ownership, making it uniquely yours.

In summary, Ping Pong Game offers diverse benefits: from enjoyment and learning to coordination enhancement and social bonding. It's a valuable addition to digital entertainment, enriching lives and bringing joy to players of all ages.

4. LITERATURE SURVEY

Our literature survey shapes our game's design, technical prowess, and player engagement by drawing on existing wisdom in game development, ensuring a well-crafted and captivating experience.

Game Development Insights:

We explored resources on game mechanics, collision detection, and user interfaces, learning from established practices to ensure our Ping Pong Game meets industry standards and captivates players.

4.1 OOP and SDL2 Mastery:

By studying how others use object-oriented programming and SDL2 graphics, we gained insights into coding structures, entity management, and graphical interfaces, enhancing our project's design and functionality.

4.2 Algorithm Excellence:

Our research into collision, ball-paddle interaction, and game physics algorithms empowered us to optimize performance and accuracy, ensuring smooth gameplay.

4.3 User-Friendly Interfaces:

Through user interface design principles, we honed intuitive controls, clear feedback, and user-friendly menus, prioritizing a seamless and enjoyable player experience.

4.4 Seamless Multiplayer:

If we incorporate multiplayer, our study of networking and multiplayer architecture will guide us in creating smooth online interactions and addressing potential challenges.

4.5 Playful Learning:

Exploring educational game design and learning-through-play strategies inspired us to infuse educational elements into our game, aligning with our aim to educate while entertaining.

4.6 Player-Centric Customization:

Our examination of game customization and user preferences informed features like customizable winning scores, enriching player engagement and satisfaction.

5. EXISTING SYSTEM

The concept of a Ping Pong game is far from new. It has existed long before the advent of modern computers and technology. Therefore, it's not unexpected that the game has been translated into computer game formats multiple times, even on commercial scales.

Several analogous applications have already been developed across various platforms like Windows, Android, and iOS. With a focus on learning, our approach was to create simplified versions of these existing applications while incorporating our own modifications to enhance the learning experience.

6. METHODOLOGY

To accomplish our project goals, we adopted the following systematic approach:

6.1. Initiation and Planning:

We began by planning and distributing tasks among our team of three members. We familiarized ourselves with the required SDL2 library and revisited the fundamental game logic of Ping Pong. This ensured we had a solid grasp of the rules necessary for the game's implementation.

6.2. Algorithm Design:

Once armed with the necessary rules and information, we proceeded to design the algorithm and create a flowchart for the game project. We formulated a basic working model algorithm that served as a foundation for subsequent testing, validation, and continuous development.

6.3. Software Design:

With our algorithm framework in place, we transitioned to coding in the Object-Oriented paradigm. Our implementation primarily utilized C++ as the core language and SDL2 for graphics, as it provided an intuitive learning curve and ease of use. We chose Visual Studio and Visual Studio Code as our Integrated Development Environments (IDEs) along with Ming-W g++ as the compiler for the Windows operating system. In Linux, we employed the GCC (GNU Compiler Collection) C++ Compiler (g++).

6.4. Testing and Debugging:

We initiated our development by creating a Minimum Viable Product (MVP) version of the project, which underwent rigorous testing and debugging. Additional rounds of testing and debugging followed, allowing us to progressively integrate features, enhance the code, and refine the project based on our requirements and capabilities.

7. IMPLEMENTATION

The implementation phase of our project revolved around transforming the timeless concept of Ping Pong into a digital realm. Just as many of us have fond memories of playing this game in its physical form, we embarked on recreating that joyous experience through code.

Our project initially set out to apply Object-Oriented principles to create a simple version of the Ping Pong game. While our objective wasn't to create a fully-fledged game for the global applications market, our approach allowed us to delve into game development using a structured approach. The complete version of the program is available on GitHub at:

<https://github.com/GlitchyStar717/PingPongDraft1>.

7.1. System Block Diagram

The system block diagram presents an overview of the Ping Pong game's structural elements and their interaction

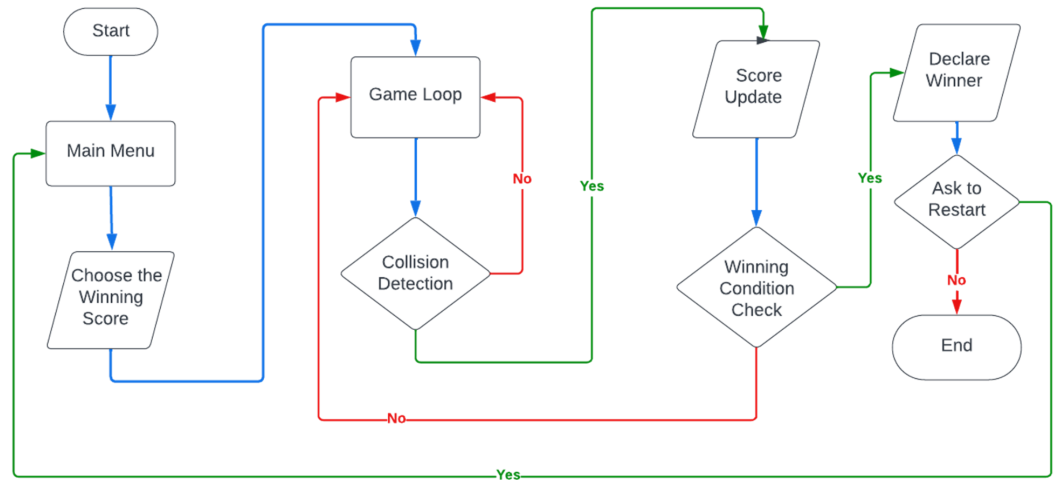


Fig 7.1 : System Block Diagram

7.2. Gameplay Overview

Upon launching the game, players are presented with the option to select the winning score required to secure victory. This feature allows players to decide how long they want the match to last and how challenging they want the gameplay to be.

7.3. Game Interface

The game interface showcases the classic Ping Pong table layout with paddles and a ball. The screen is divided into two halves, each representing a player's side of the table.

7.4. Paddle and Ball Mechanics

The game revolves around two key components: the paddles and the ball. Players control the paddles to strike the ball back and forth across the table. Paddle movements are responsive to player input, creating an interactive gameplay experience.

7.5. Gameplay Flow

The gameplay follows a straightforward sequence:

- Players control their respective paddles using designated keys or controls.
- The ball starts from the center and moves towards one player's side.
- Players maneuver their paddles to hit the ball and send it towards their opponent's side.
- The game continues as players volley the ball, aiming to prevent it from reaching their side's back wall.

7.6. Winning Conditions

The game determines a winner based on a set score limit chosen by players before starting the match. The player who reaches the specified score first is declared the winner. This score-based winning condition adds a competitive edge to the game.

This overview elucidates the essential gameplay elements of the Ping Pong game, highlighting its simplicity and focus on multiplayer enjoyment. Players can expect an engaging experience that captures the essence of the traditional Ping Pong sport in a digital format.

8. RESULTS

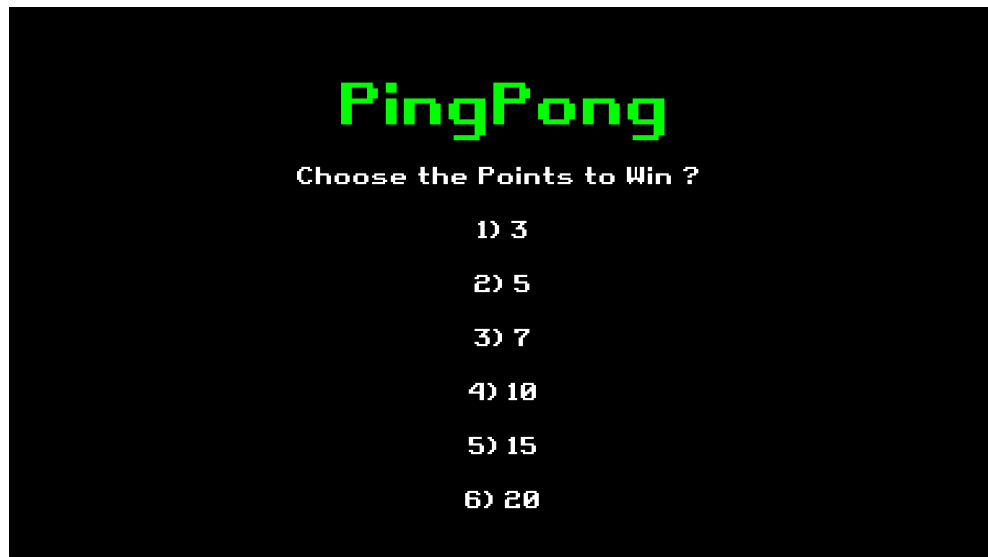
Upon the completion of our Ping Pong game project, we successfully accomplished the primary objective of applying Object-Oriented Programming (OOP) principles using C++ and SDL2. While the game lacks certain advanced features, it effectively demonstrates the utilization of OOP concepts for game development.

Throughout the development process, our team fostered collaboration among members, gaining proficiency in fundamental game programming and SDL2 modules. We improved our skills in version control using Git and GitHub, document preparation with MS-Word, and various other essential aspects.

Screenshots taken at different stages of the game provide visual insights into the final outcome of our project:

8.1. Game Start

The main menu serves as the entry point to the game, offering options for the number of points required to win the game.

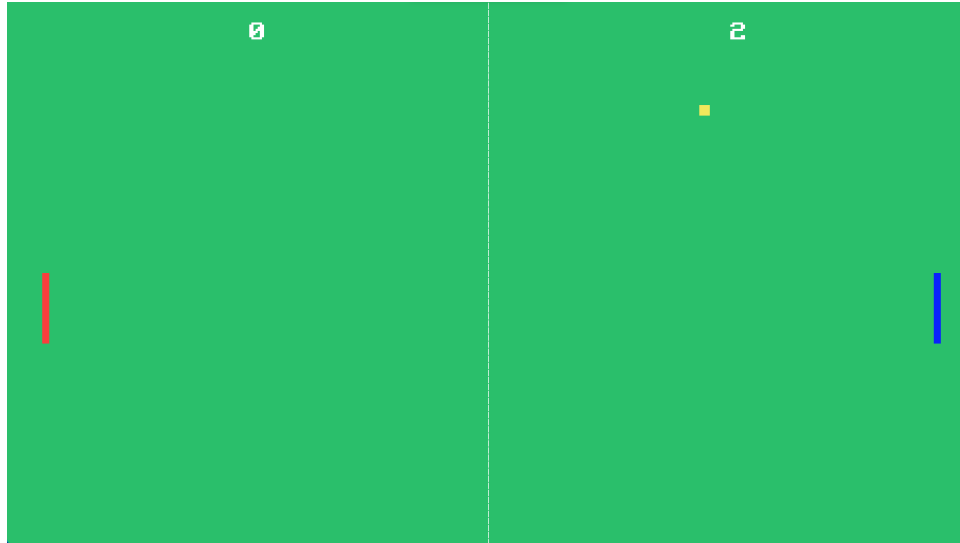


8.2. In-Game Play

The gameplay screen showcases the ping pong table and players' paddles, with the ball in motion between them.

8.3. Score Update

As the game progresses, the score is updated in real-time, indicating the progress towards the chosen winning score.



8.4. Victory Celebration

Upon achieving the set winning score, a victory message is displayed, celebrating the winning player's success.

8.5. Post-Match Options

After a match concludes, players have the option to restart the game or exit to the main menu.



9. PROBLEMS FACED AND SOLUTIONS

While making our Ping Pong game, we ran into some tricky situations, but we found clever ways to fix them:

9.1. Sorting Out Code

As our game got bigger, finding specific parts of the code became a bit tough. So, we decided to write comments that explained what each part of the code did. We also gave meaningful names to variables and functions, so anyone reading the code could understand it easily.

9.2. The Ghost Ball Mystery

Sometimes, the ball in our game would act invisible and mess up the game. We figured out that the ball's position wasn't being updated correctly. To solve this, we added extra checks in the ball's movement to make sure it moved properly all the time.

9.3. Flickering Screen

When the ball hit the paddles really fast, the screen sometimes flickered. This was because the screen was being updated too quickly. We slowed down the updates a bit, and the flickering stopped, making the game smoother to play.

9.4. Ball Speed Fluctuation

We observed that the ball's speed wasn't consistent, causing unexpected variations in its movement across the screen. This inconsistency disrupted the game's flow and fairness.

10. LIMITATIONS AND FUTURE ENHANCEMENTS

Our Ping Pong game project, although a valuable learning experience, has certain limitations and potential avenues for future improvements:

10.1. Limitations

- **Lack of Advanced Features:** Due to time constraints and academic commitments, we couldn't implement certain advanced features that would enhance the game's complexity and user experience.
- **Limited Multiplayer Options:** Our current version only supports local multiplayer. We weren't able to integrate online multiplayer functionality via networking, which could have extended the game's reach to players beyond the same physical location.
- **Single Style Board:** The game features a single style of game board design. A broader variety of board styles could have added visual interest and diversity to the gameplay.
- **Graphical Interface:** While functional, the game's graphical interface could have been improved to provide a more visually appealing and immersive experience.
- **Fixed Screen Size:** The game's screen size isn't dynamic, which means it may not adapt seamlessly to various screen sizes and resolutions.

10.2. Future Enhancements

- **Online Multiplayer:** Implementing an online multiplayer mode would allow players to compete with opponents from different locations, adding a new level of challenge and excitement.
- **Move Tracking and Undo Feature:** Incorporating a move tracking system and an undo feature could enhance the strategic aspect of the game by allowing players to review and adjust their moves.
- **Advanced AI:** Developing a more sophisticated AI opponent using algorithms like Alpha-Beta pruning and Minimax could provide a challenging single-player experience.
- **Enhanced Board Representation:** Utilizing advanced techniques like bitboards to represent the game board could optimize the game's performance and memory usage.
- **Varied Board Designs:** Introducing a range of board designs and styles would add visual variety and aesthetic appeal to the game, catering to different player preferences.

In conclusion, while our Ping Pong game project has its current limitations, it also offers exciting possibilities for future improvements. These enhancements could elevate the game's features, accessibility, and overall enjoyment for players of all levels.

11. CONCLUSION AND RECOMMENDATIONS

To sum it up, our Ping Pong game project was a great learning experience. We got to use Object-Oriented Programming (OOP) with C++ to build the game. Through this project, we learned how to plan, work together, and solve problems as a team.

We also discovered useful tools like Github for managing our code, MS-Word for making documents look nice, Canva for creating presentations, and Lucidcharts for making diagrams. Working together taught us how important it is to communicate well and help each other.

While coding was important, we realized that planning and thinking about what we want to do are even more crucial. Fixing issues, testing, and making the program better were important steps too.

In the end, we learned that making software involves different stages like planning, creating, testing, and fixing. This project showed us the power of Object-Oriented Programming and gave us a strong foundation for future projects.

12. REFERENCES

12.1. Book References:

“The Secrets of Object oriented Programming”, Daya Sagar Baral,
Diwakar Baral
“The C++ Programming Language”, Bjarne Stroustrup
“C++ How To Program” , Paul Deitel, Harvey Deitel
“Object Oriented Programming with C++” , Robert Lafore
“SFML Game Development By Example”, Raimondas Pupius

12.2. Web References:

<https://www.libsdl.org/>
<https://devdocs.io/cpp/>
<https://docs.microsoft.com/en-us/cpp/>