

**TRIBHUWAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**

**PULCHOWK CAMPUS**

**A PROJECT REPORT ON**

**CARROM BOARD GAME**

**USING C++**

**TEAM MEMBERS:**

**Prabesh Bashyal** [078BCT057]

**Pratik Adhikari** [078BCT062]

**Rishikesh Paudel** [078BCT064]

This report is submitted to the

**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING**

23rd August, 2023

**ACKNOWLEDGEMENT**

First and foremost, we would like to express our deepest gratitude and appreciation to our honorable lecturer, Daya Sagar Baral for his invaluable guidance, and encouragement for the development of this project. His insightful lectures, suggestions, and constructive feedback have helped us shape our ideas and refine our approach.

We would also like to extend our sincere thanks to the Department of Electronics and Computer Engineering, Pulchowk Campus (IOE) for providing us with necessary resources, facilities and interactive environment to learn and implement the knowledge through this project.

Finally, we would like to acknowledge our friends for the help and support they provided during this project, may it be tackling a bug or brainstorming new ideas. I would be remiss in not mentioning our family members who have unwavering faith in our abilities and were a great source of motivation for us.

Although it is not possible to mention everyone by name, we are deeply grateful to all the individuals who have contributed directly or indirectly to this project. We remain grateful for your involvement, support and contribution for implementation and completion of this OOP project. All these efforts played a vital role in project’s overall success.

Sincerely,

Prabesh Bashyal [078BCT057]

Pratik Adhikari [078BCT062]

Rishikesh Paudel [078BCT064]

**ABSTRACT**

Object Oriented Programming (OOP) is a paradigm of programming which provides easier techniques and facilities to create a robust and efficient system that addresses a real-world problem. This project focuses on effective utilization of OOP principles through C++ to create a Carrom Board game that binds the players into the game through its lively and joyful game environment.

The project centers around OOP concepts and uses C++ programming language. It also includes external C++ library Simple DirectMedia Layer (SDL) for developing the graphical user interface (GUI), handing user input and managing audio. Carrom board is a multiplayer game that has various rules that differes from place to place. In this project, we have tried to include most of the standard rules and avoid the local rules, along with customizable settings to ensure immersive gameplay.

[about socket programming and multiplayer options ……. ]

Through this Carrom Board Game we strive to create a visually appealing and user-friendly game that captures the essence of Carrom.

**TABLE OF CONTENTS**

**OBJECTIVES1**

**INTRODUCTION2**

## **Object Oriented Programming (OOP) in C++2**

## **Simple DirectMedia Layer (SDL)2**

## **OOP Features in Game2**

## **Socket Programming2**

## **Objects and Classes2**

### **Abstraction2**

### **E****ncapsulation2**

### **Inheritance2**

### **Polymorphism2**

**APPLICATION3**

**LITERATURE SURVERY4**

**EXISTING SYSTEM5**

**IMPLEMENTATION6**

**BLOCK DIAGRAM7**

**RESULTS8**

**PROBLEMS FACED AND SOLUTIONS9**

**LIMITATIONS AND FUTURE ENHANCEMENTS10**

**CONCLUSION AND RECOMMENDATIONS11**

**REFERENCES12**

**1. OBJECTIVES**

Here are the key points regarding the goal and objectives of the project:

* **Understanding Object-Oriented Programming (OOP):** The project aimed to gain a fundamental understanding of key concepts in OOP by developing a game. Games inherently involve various real-life objects, making them suitable for implementing object-oriented principles in C++.
* **Utilizing Objects in C++:** The project provided an opportunity to explore the implementation of objects in C++ and gain insights into their behaviour and methods. By working with objects, the project aimed to enhance understanding of encapsulation, inheritance, and polymorphism.
* **Code Reusability:** Emphasis was placed on writing reusable code rather than creating long and complex code blocks. The project aimed to break down the code into smaller sub-programs, promoting modularization and facilitating code reuse across different parts of the game.
* **Header Importing:** The concept of "header importing" in OOP was employed to improve code organization and readability. By using header files, the project aimed to encapsulate related functionality, making it easier to manage and comprehend the codebase.
* **Enhancing Programming Skills:** Developing a game allowed for hands-on practice and application of programming concepts. Through this project, participants aimed to improve their programming skills, problem-solving abilities, and gain practical experience in developing software applications.
* **Learning Software Development Processes:** The project provided an opportunity to understand the software development lifecycle, including requirements gathering, design, implementation, testing, and deployment. Participants aimed to apply these processes while developing the carrom board game.
* **Collaboration and Teamwork:** The project fostered collaboration and teamwork as participants worked together to design, develop, and test the game. Through effective communication and coordination, the team aimed to create a cohesive and functional final product.
* **Exploring Game Mechanics:** The development of the carrom board game allowed participants to dive into the mechanics of the game itself. They aimed to understand and implement the rules and gameplay elements specific to carrom, such as striking the striker, pocketing coins, and scoring.
* **Experiencing Visual Programming:** By utilizing the graphics library and incorporating visual elements, the project aimed to provide hands-on experience in visual programming. This involved creating graphical user interfaces, rendering game elements, and displaying dynamic graphics to enhance the gaming experience.

Overall, the project focused on utilizing OOP concepts, code reusability, graphics libraries, and teamwork to develop a carrom board game, facilitating a better understanding of OOP principles and enhancing programming skills.

**2.** **INTRODUCTION**

Carrom Board is a popular tabletop game played among 2-4 players, originated in South Asia. The word “Carrom” means to strike and rebound. This game is played on a square wooden board (29 x 29 inches) with pockets(holes) in each corner. The objective of the game is to pocket the playing pieces (discs) using a striker by various techniques and strategies. Different standards and rules exist in different areas.

****

The game is played using discs also known as carrom men which slide when struck by a Striker. Carrom board is like “strike and pocket” games, like pool. A carrom disc set contains 19 pieces in three distinct color and a striker. Black, white are the colors used for players and the red color is used for the queen. The striker is slightly heavier and bigger than the other discs and is used to hit the other pieces across the board.

The game is played by arranging all the carrom pieces in the center circle and hitting it with the striker from the boarder of the inner square box. Crossing the diagonal lines on the board by coming in touch with it and pocketing the striker is a foul. A successful pot entitles the player to shoot again. Any player pocketing the queen is required to cover it immediately by pocketing one of their carrom men on the consequent shot. If the player fails to do so, then the queen is returned to the center of the table. If a player pockets his striker, he has to pay a penalty. There are different variations in which the game can be played. It can be played by designating only one color to one player and pocketing only that kind which is called “game” or pocketing any color (black and white) which is called “*jogi*” locally.

While playing “*jogi*” the queen is given 50 points, black is given 10 points and white is given 20 points. As mentioned, point and rules differ from place to place.

**1.1 Object Oriented Programming (OOP) in C++**

Object oriented programming (OOP) is a way of structuring your program/code so that you create objects that performs tasks rather than creating functions like in procedural programming language.

As the name stands objects are the main part of OOP as we are focused on creating objects. This can be a difficult concept to grasp as a beginner, but it is a refined concept and is essential for bigger projects for maintaining and structuring the code.

I would like to give an example that I found on the internet to explain what objects are, objects are not that complex to understand we can take them as living beings like you and me. Similar like we perform tasks objects also perform tasks. If you order me to wash some clothes and went on with your work. Whether I wash the clothes myself or take a cab to a laundry pay them to wash and bring back the clothes, you wouldn't know how I washed. Objects are similar, you order them with tasks, and they provide you with results so that you wouldn’t have to go through the complex process completing the tasks. It encapsulates the complexity of the task providing user with very high level of abstraction.

This concept of writing and structuring the code is widely used and C++ provides a programming platform for writing code in that manner. This flexibility enables the development of robust, extensible, and maintainable software systems in C++.

**1.2 Simple DirectMedia Layer (SDL)**

Simple DirectMedia Layer (SDL) is a cross-platform software development library designed to provide a hardware abstraction layer for computer multimedia hardware components. It is a library used to build games or other multimedia applications that can run on many operating systems such as Android, iOS, Linux, macOS and Windows. It is a free open-source library available for C++ and also for many other programming languages. However, SDL is not a game engine but is used by game engines built on top of it. SDL provides support for 2D pixel operations, sound, file access, event handling, timing. With SDL, developers can write code that is portable across different platforms, making it a popular choice for building cross-platform games and multimedia applications efficiently and effectively.

**1.3 Socket Programming**

Socket programming is a fundamental concept in computer networking that enables communication between different devices over a network. It involves the use of software interfaces, known as sockets, to establish connections and exchange data between a client and a server. Sockets facilitate the seamless flow of information by providing a standardized way for applications to send and receive data, regardless of the underlying hardware or operating systems. Whether it's for building web applications, online gaming, file sharing, or any network-based interaction, socket programming forms the backbone of real-time data exchange on the internet, allowing diverse systems to communicate effectively.

**1.4 OOP Features in Game**

The main objective of this project is to strengthen our Object Oriented Programming concepts through the practical implementation. This project is made inclusive of all the main features/characteristics of OOP. Some of the features used in the making of the game are given here:

**1.4.1 Objects and Classes**

Objects are the building blocks in OOP. Every problem is solved through the creation of objects. These objects perform tasks that are to be done by the user maintaining the simplicity and hiding the complexity which helps in attaining the highest level of abstraction. As objects are the instance of class, we created various classes that contains appropriate methods to get our tasks done. Classes carromPieces were created to get the basic information about all the pieces and further classes like Black, White and Queen were created for specific pieces information. Many other classes were created and their objects were instantiated for other minor and major tasks.

**1.4.2 Abstraction**

Abstraction is an important feature of OOP that helps the programmer hide the internal details and complexity of the program and provides a simpler and easier interface to the user for their service. Classes are an example of Abstract Data Type (ADT) as it helps in achieving abstraction by hiding the implementation details. The objects go through various calculations and processes but the method we call only returns us the useful result and hides the complex work done by the objects this can only be achieved by abstraction. In the project, we created various classes and the methods consisted of various functions to define a texture and surface, load the image as surface, convert the surface to texture, render the texture, free the surface and clear the render to return a beautiful image to the user on their screen. Thus, we got a image as demanded by the user but we hid the complex process behind the rendering of the image through the application of this feature in OOP.

**1.4.3 Encapsulation**

Encapsulation is another important feature in OOP. This feature helps us bind the data together with function into a single unit. It can be said that encapsulation is a protective box that prevents the data from being accessed by other functions or classes defined outside the box. Encapsulation can be achieved through classes as each class helps us bind the data and methods together. The data and functions within a class can be accessed in various ways as specified by the access specifier.

There are three access specifiers:

1. **Public**: Data and functions defined under this specifier can be accessed from anywhere outside the class. This doesn’t provide data hiding functionality and data is prone to alteration.
2. **Private**: Data and functions defined under this specifier can be accessed only from within the class and can’t be altered by any code written outside the class. This provides protection to the data and prevents from accidental alteration
3. **Protected**: Data and functions under this specifier can be accessed from within the class itself and the class that has been derived from the class. This can be useful during inheritance.

This insulation of data from direct access by the program is called as data hiding. Data hiding prevents the data from accidental alteration and helps manage the data safe and protected. This also helps to maintain privacy of our data.

**1.3.4 Inheritance**

Inheritance, as its name suggests is a feature that is used to inherit the features of base class to a derived classes providing us with the abilities such as code reuse and elimination of redundant code. It maintains hierarchy between the base and derived classes.

Taking reference from our project, Black, White and Queen are the derived classes derived form a base class CarromPieces. Here, the classes Black, White and Queen were inherited the features of CarromPieces. These derived classes can also access the methods and data of the base class defined under public and protected. The class CarromPieces also provides a template for the other three classes.

CarromPieces

(Base Class)

Black

White

Queen

(Derived classes)

**Hierarchical Inheritance**

**1.3.5 Polymorphism**

Polymorphism in OOP allows objects to respond to same operation in different ways. We can use a same function or a same operator in different ways depending on what they are operation, this is called polymorphism. The ability to exist in many forms and this can be achieved through OOP. Polymorphism is used for overloading a function, operator and for creating virtual functions. In C++, polymorphism is divided in to two types:

1. **Compile time polymorphism**

Compile time polymorphism also known as ‘static’ or ‘early binding’ can be achieved by operator and function overloading.

Here, the compiler knows which function to execute before the program is complied.

1. **Run time polymorphism**

Run time polymorphism also called ‘dynamic’ or ‘late binding’ occurs when the functions are resolved at runtime rather than compile time. This is achieved by the use of function overriding and virtual functions.

**3. APPLICATION**

Carrom Board is a popular game that is widely played in the South Asian countries. Nepalese people are also found to be very fond of the game and most of us have played carrom board at least once in our lives. Carrom board serves as a perfect entertainment for leisure and spending quality time with friends and family. In the digital world, we can see people on their phones/laptops most of the time so a digital replica of the real world carrom with no compromise quality with immersive gameplay can be easily accessible to all the people.

Our project mainly covers following areas of application:

1. It can be played between multiple users in real time through LAN.
2. It helps us spend quality time with members of family and friends.
3. It can be a great source of entertainment during leisure.

LITERATURE SURVERY

**EXISTING SYSTEM**

Carrom board game is not entirely new game, various applications have already been made regarding this game even for commercial uses. Therefore, what have created is not something entirely new but we tried to present it in a new way. Most of the carrom board games were not either properly documented or didn’t had the understandable resources to guide us. So, our system overcomes that difficulty by providing a easier and smoother gameplay for recreation and leisure.

Various ios and Android application for carrom board can be seen on App store and play store. People use their devices’ touchscreens to replicate the flicking motion of a striker, aiming to pocket coins into the corners while avoiding the opponent’s pieces. Some of the well known carrom board games for mobile and PC are; Carrom Pool, Carrom 3D, Carrom Disc Pool, Carrom Club.

METHODOLOGY

IMPLEMENTATION

BLOCK DIAGRAM

RESULTS

PROBLEMS FACED AND SOLUTION

**LIMITATIONS AND FUTURE ENHANCEMENTS**

The current version of our carrom board game does not include an online multiplayer option due to technical limitations and development constraints. We strive to offer the best possible experience with local multiplayer, multiplayer on same PC and practice sessions. We would also like to improve our leader board and make it more appealing to our players. As the digitalization is expanding, limiting our game to local multiplayer option will not release its full potential so we want to implement a real time online multiplayer game mode so that not only the friends that are side by side but you friends across the globe can connect to you with our game. We recognize the growing demand for online gameplay but implementing real-time multiplayer functionality involves complex networking architecture and synchronization mechanisms to ensure a seamless and fair gaming experience across the internet.

However, we are dedicated to enhancing the game in the future by actively working on incorporating a robust online multiplayer feature. By leveraging internet connectivity and advanced matchmaking algorithms, we aim to create an engaging and competitive environment for players to showcase their skills globally. We would also like to add various achievements in our game so that the player feels motivated when they complete a quest of carrom. We appreciate your patience and support as we work toward this exciting upgrade, which will undoubtedly elevate the game's appeal and provide a more immersive carrom experience for our players.

CONCLUSION AND RECOMMENDATIONS

REFERENCES