1 Introduction

The "**Sudoku Archery Quest: A Fusion of Focus and Precision**" project introduces an ingenious fusion of two distinct yet captivating activities: the thrill of archery gaming and the mental agility of Sudoku puzzle-solving, all seamlessly integrated into an immersive computer graphics environment. This extraordinary amalgamation creates a novel and captivating experience that caters to a wide spectrum of players, harmonizing the excitement of real-time action with the cognitive stimulation of strategic problem-solving. Leveraging the power of OpenGL and GLUT libraries, this project establishes an interactive and dynamic platform, inviting users to indulge in the dual pleasures of conquering archery challenges and mastering Sudoku puzzles.

In this project we have used OpenGL source code in C++, and merged two real life game. It allow users With OpenGL, we can create stunning scenes that immerse in the archery challenge. The targets come alive with realistic, making every arrow shot feel like a true test of precision. The Sudoku grids are crisp and clear, ensuring that one can focus on solving the puzzle without any distractions. OpenGL allows us to add cool effects like smooth transitions between games and dazzling particle effects when arrows hit their targets.

The OpenGL and GLUT libraries serve as the technological backbone of the project, providing the tools to construct a visually captivating and user-friendly interface. This interface acts as a portal into the world of "Precision Fusion," allowing players to immerse themselves in challenging archery scenarios that demand focus, accuracy, and agility. Moreover, it provides a space where Sudoku enthusiasts can strategically fill grids and unravel number-placement mysteries.

In summary, the "Sudoku Archery Quest: A Fusion of Focus and Precision" project pushes the boundaries of traditional gaming experiences by merging two seemingly incongruent activities into a harmonious and captivating blend. By harnessing the capabilities of OpenGL and GLUT, the project delivers an interactive platform where users can relish the excitement of archery while nurturing their logical faculties through engaging Sudoku gameplay. Get ready to embark on a journey that celebrates the synergy of action and intellect, all encapsulated within a computer graphics marvel.

2 Objectives

- ❖ To apply OpenGL in a real life game project
- ❖ To develop an enjoyable Sudoku Game experience
- ❖ To create an immersive 2D Archery Environment
- ❖ To switch smoothly from Sudoku to Archery on success
- ❖ To Produce Clear User Guides and Effective Presentations

3 Application

Some potential applications of the "Sudoku Archery Quest: A Fusion of Focus and Precision" project are:

- **Gaming and Entertainment:** The primary application of the project lies in providing gamers with a novel and engaging gaming experience. By combining archery challenges and Sudoku puzzles, the project offers a unique form of entertainment.
- **Skill Enhancement:** The project can serve as a platform for players to enhance their hand-eye coordination, precision, and strategic thinking skills.
- **Cognitive Training:** The Sudoku puzzles integrated into the project can have cognitive training applications. Users can benefit from improved problem-solving skills, critical thinking, concentration, and memory retention.
- **Stress Relief:** As a recreational activity, the project can provide an avenue for stress relief. Players can immerse themselves in the challenges, diverting their minds from daily stressors while engaging in entertaining gameplay.
- **Educational Tool:** Schools or educational institutions can use it as a supplementary tool to teach students about logical deduction through Sudoku and physics principles through archery simulations.
- **Brain Exercise:** The combination of archery and Sudoku offers a well-rounded brain exercise. Players can switch between the fast-paced action of archery and the more contemplative Sudoku-solving mode, providing a comprehensive mental workout.

- **Time Management:** Players can set time limits for both archery challenges and Sudoku puzzles. This aspect can be particularly useful in helping individuals improve their time management skills and decision-making under pressure.
- **Casual Gameplay:** The project's diverse gameplay mechanics make it suitable for casual gamers as well. Players who enjoy occasional gaming sessions can find value in the mix of archery fun and Sudoku-solving relaxation.
- **Exploration of Hybrid Genres:** The project's unique blend of archery and Sudoku opens up avenues for further exploration of hybrid gaming genres, potentially inspiring other developers to combine seemingly unrelated gameplay elements in creative ways.

In essence, the "Sudoku Archery Quest: A Fusion of Focus and Precision" project can have a wide range of applications, ranging from entertainment and skill-building to education and therapy, all while providing an engaging and innovative gaming experience.

4 Required Tools

The development of the "Sudoku Archery Quest: A Fusion of Focus and Precision" project necessitates the use of several essential tools and technologies. The following tools are integral to the successful realization of this project:

- **IDE:** Code::Blocks
- **Programming Language:** C++
- **Graphics Rendering API:** OpenGL
- **Library:** GLUT (OpenGL Utility Toolkit)

5 Methodology

5.1 Sudoku Solver

5.1.1 Initialization of Sudoku Grid:

Initialize() function initializes the OpenGL environment. It sets up the display mode and window size, as well as the initial position of the Sudoku grid within the window. This function is crucial for configuring the graphical environment before any drawing or interaction takes place.

5.1.2 Drawing Individual cell:

drawSquare() function is responsible for drawing an individual cell of the Sudoku grid using OpenGL. It calculates the coordinates of the cell's vertices and uses OpenGL commands to draw the lines that form the cell's boundaries. The function also handles drawing the numbers that the user inputs into the grid.

5.1.3 Handling Keyboard Input:

keyboard() function handles keyboard input from the user. It is likely used to facilitate the process of filling in the Sudoku grid. When the user presses a number key while a cell is selected, this function likely registers that number in the corresponding cell.

5.1.4 Capturing Mouse Click:

The purpose of **mouseClick()** function is to capture mouse clicks on the Sudoku grid. When the user clicks on a cell, this function identifies the clicked cell and might update its status or allow the user to input a number. It interacts with other functions to update the grid based on the user's input.

5.1.5 Initial Graphical User Interface:

preDraw() function sets up the initial display state before entering the main drawing loop. It might display messages like "Sudoku Solver" and create buttons like "Solve" and "Clear" that users can interact with during the solving process. It prepares the graphical user interface elements.

5.1.6 Creating Sudoku Rules:

The role of **test()** function is to determine whether the current state of the Sudoku grid is a valid solution. It checks if the grid adheres to the Sudoku rules: each row, column, and 3x3 subgrid must contain all numbers from 1 to 9 without repetition. This function is likely used to verify whether the puzzle has been successfully solved.

5.1.7 Resetting Sudoku Grid:

clearBoard() function resets the Sudoku grid to its initial state. It clears all the filled cells, allowing the user to start solving the puzzle anew. It's a way to provide users with a clean slate to work with.

5.1.8 Entire User Interface:

The purpose of **display()** function is to render the entire Sudoku grid and user interface components using OpenGL. It draws the grid lines, numbers filled by the user, and any other interface elements created for the solver. It's responsible for updating the display whenever changes are made.

5.1.9 Creating predefined puzzle:

inputBoard() function initializes the Sudoku grid with a predefined puzzle. It sets up the initial state of the grid with some cells already filled, creating the puzzle that users will attempt to solve.

5.1.10 Sudoku Solving Algorithm:

The **solveBoard()** function is the heart of the Sudoku-solving algorithm. It uses a backtracking approach to recursively attempt to solve the puzzle. The function tries placing numbers in empty cells and checks if the puzzle remains valid. If not, it backtracks and explores different number placements until a valid solution is found.

5.1.11 Checking Sudoku Rules:

checkBoard() function validates whether a given number can be placed in a specific cell without violating the Sudoku rules. It checks if the number is already present in the cell's row, column, and 3x3 subgrid. This validation is crucial to ensure that the user's inputs and the solving algorithm maintain the puzzle's integrity.

5.2 Archery Game

5.2.1 Preview page of Archery Game:

display1() function introduces players to the archery game with an informative preview page. It employs varying font sizes to emphasize key details. It explains that the right mouse button is used for the menu, 'r' key releases arrows, and 'q' key exits. The page encourages starting the game with the space bar. The background color is a pleasing teal shade, and the layout is clear. This page familiarizes players with the game's mechanics before they dive in.

The display_string function, **display_string**(double x, double y, char string, int font) shows text on-screen in different fonts and positions. It takes parameters for horizontal (x) and vertical (y) positions, the actual text (string), and a font value (font). The function calculates the text length and displays each character using the selected font style. This function effectively conveys information in various font sizes, aiding clarity and player readiness.

5.2.2 Drawing an Arrow

The **disp( )** function takes care of displaying main arrow in the archery game scene. Depending on the view, the function either initializes the game and displays the introductory content using the init and display1 functions, or proceeds to showcase the actual archery gameplay.

The arrow is drawn if view is not zero. It's comprised of lines, triangles, and quads, all collectively forming the arrow's shape. Colored lines represent the arrow's body, and a red triangle at the tip signifies the point. Moreover, a blue quadrilateral outlines the tail end. These elements come together to create an arrow representation.

5.2.3 Drawing a Target

The **draw_target** function is responsible for rendering the archery game's target on the screen. It uses a loop to draw multiple target elements. It displays a magenta central point using GL_POINTS and surrounds it with a square-shaped target using GL_LINE_LOOP. Here, the GL_LINE_LOOP is used to create the square-shaped outline of the target around the central point.

If the target_state is not 0, it displays only white points using GL_POINTS, creating a visual distinction between active and hit targets. The four vertices provide for the loop form the corners of the square. OpenGL connects these vertices with lines, resulting in a closed square shape.

5.2.4 Initializing Target

The **init_targets** function initializes properties for 10 targets in the game. Through a loop, it assigns random positions within specified ranges to target_x and target_y. The target_state is set to 0, indicating not hit, while target_direction is randomly assigned as 0 or 1, determining initial movement. This function ensures diverse starting positions and movement directions for each target, contributing to the game's dynamic and challenging gameplay.

5.2.5 Updating Target

The **update_targets( )** function dynamically updates the positions of 10 targets in the game. It checks each target's state, and if not hit, it adjusts their vertical positions based on the target_direction. If target_direction is 1, the target moves upwards, and if it reaches the upper boundary, the direction is reversed. Similarly, if target_direction is 0, the target moves downwards until the lower boundary, then changes direction. The function continuously refreshes the display to show the updated positions.

5.2.6 Moving Arrow Upwards

The **id( )** function controls the upward motion of the arrow in the archery game. When the game window is active (view equals 1), the arrow's vertical position (y) is incrementally increased by a defined speed (VERTICAL_SPEED_OF_ARROWS). Once the arrow surpasses the upper screen boundary (MAX_Y), its position is reset to the bottom, and a count is incremented to track fired arrows. This function ensures continuous arrow movement and manages arrow resetting, contributing to the game's dynamic arrow-shooting experience.

5.2.7 Horizontal Movement of the Arrow

The **id1( )** function manages the horizontal movement of the arrow in the archery game. It checks for collision with targets by examining the arrow's position in relation to each target's coordinates. If a collision is detected, the target's state changes to "hit," the arrow is reset, and a counter increases. Additionally, the arrow's vertical motion is handed over to the id function. If the arrow reaches the right boundary of the screen, it is reset, and the counter increments again. This function ensures arrow movement, target collision detection, and boundary handling, contributing to the engaging gameplay dynamics.

5.2.8 Performing Desired Operation

The **keys( )** function acts as a keyboard input handler in the archery game. When the space bar is pressed, it displays the game graphics, activates the game window (view = 1), and initializes target positions. Pressing 'r' triggers the arrow release, invoking the id1 function for horizontal arrow movement and target collision checks. Pressing 'n' starts a new game by resetting variables, initializing targets, and changing the view. 'q' quits the game using exit(0). This function efficiently manages game actions based on keyboard input, facilitating smooth gameplay progression and interactions.

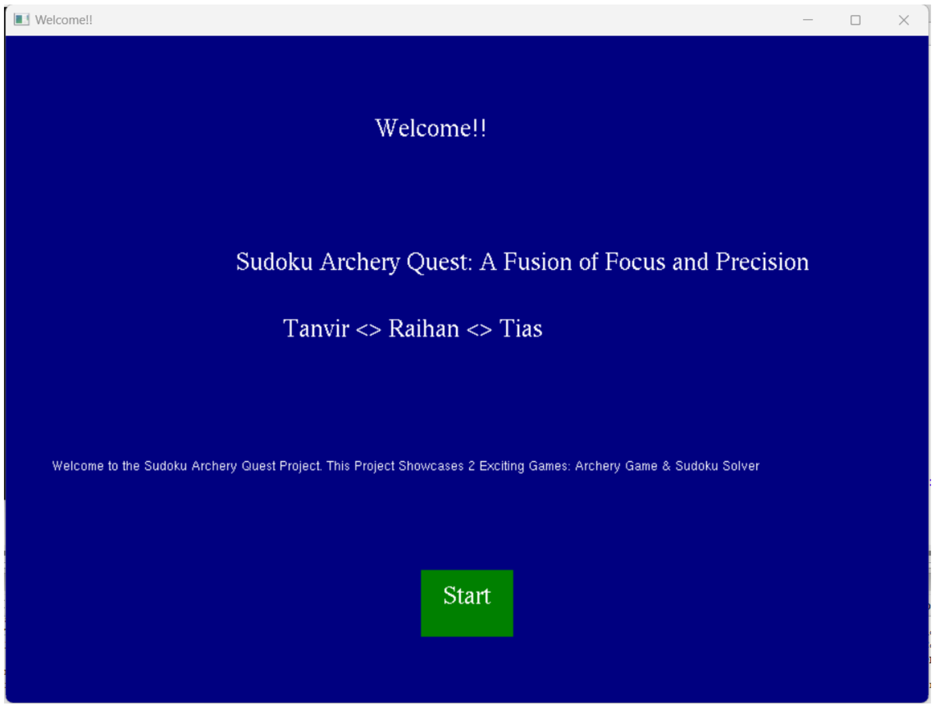
5.2.9 Result of Archery Game

The **counting** function manages the display and outcome conditions in the archery game. It generates a string indicating the number of arrows shot (count) and displays it. It checks if any targets are still unhit, if so, the win flag remains false. If all targets are hit (target_state[i] is 1 for all), the player wins, and a congratulatory message is displayed, stopping the idle function. If the player shoots more than or equal to 30 arrows without winning, a game-over message appears along with instructions to play again or exit. The counting function effectively handles game state changes and provides informative messages to the player.

6 Features

6.1 Welcome Page

In the Welcome Page, project name, project member names, and project description are displayed using the chosen font and positions.



Welcome!!

Sudoku Archery Quest: A Fusion of Focus and Precision

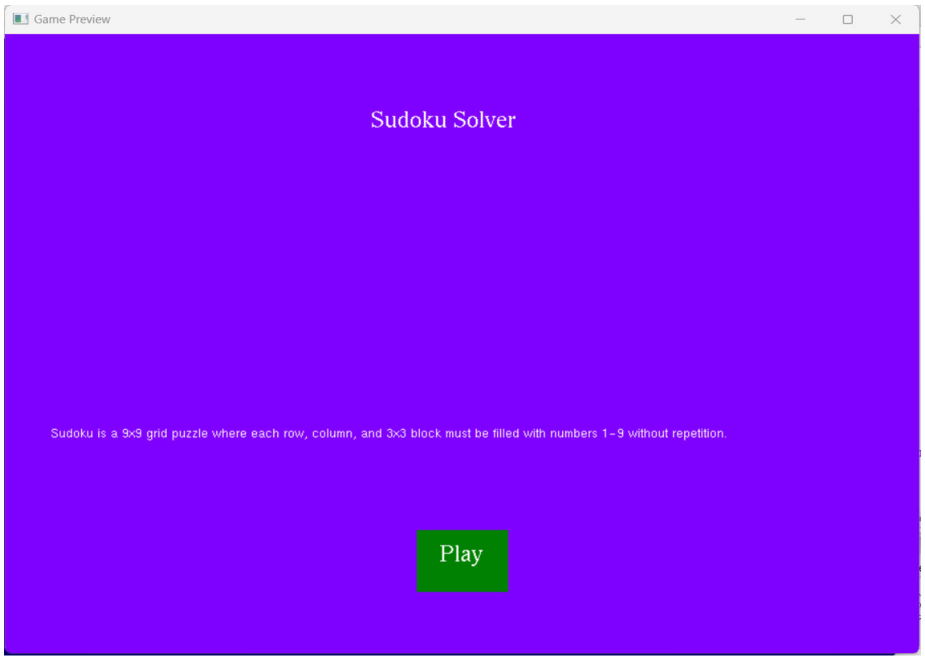
Tanvir <> Raihan <> Tias

Welcome to the Sudoku Archery Quest Project. This Project Showcases 2 Exciting Games: Archery Game & Sudoku Solver

Figure 01: Welcome Page

6.2 Sudoku Game Preview:

It shows the description and preview page for the Sudoku Solver.

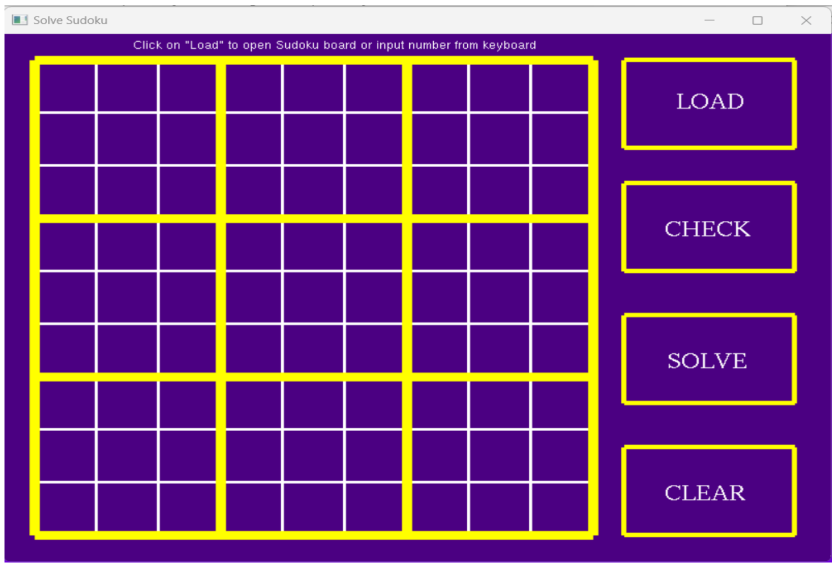


Sudoku is a 9x9 grid puzzle where each row, column, and 3x3 block must be filled with numbers 1-9 without repetition.

Figure 02: Sudoku Game Preview

6.3 Sudoku Initial Interface:

A distinct title 'Solve Sudoku' is prominently displayed at the top center of the interface.



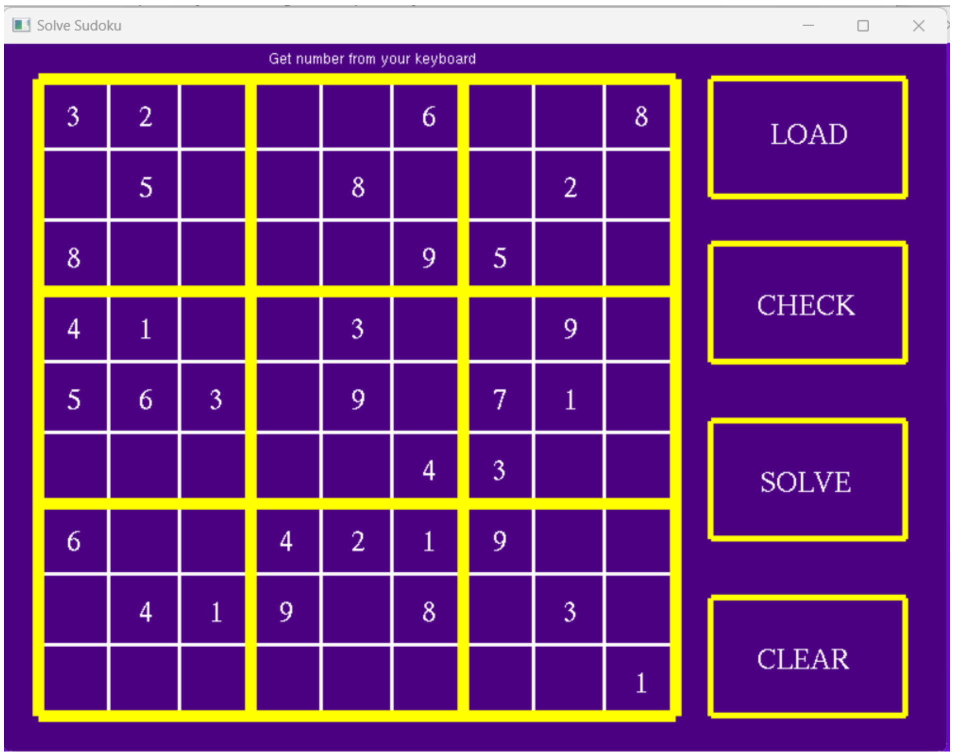
Click on "Load" to open Sudoku board or input number from keyboard

Figure 03: Sudoku Initial Interface

The gameplay area consists of a 9x9 grid, subdivided into nine 3x3 regions. The grid is initially empty, awaiting player to load Sudoku board or input number from keyboard.

6.4 After loading Sudoku Board

After loading some of the grids, players can input numbers into the grid by clicking on individual cells. Clicking a cell activates the keyboard, allowing players to type numbers using their physical keyboard. Input numbers can be used to solve the Sudoku puzzle. The solver checks for errors in real-time as players input numbers.



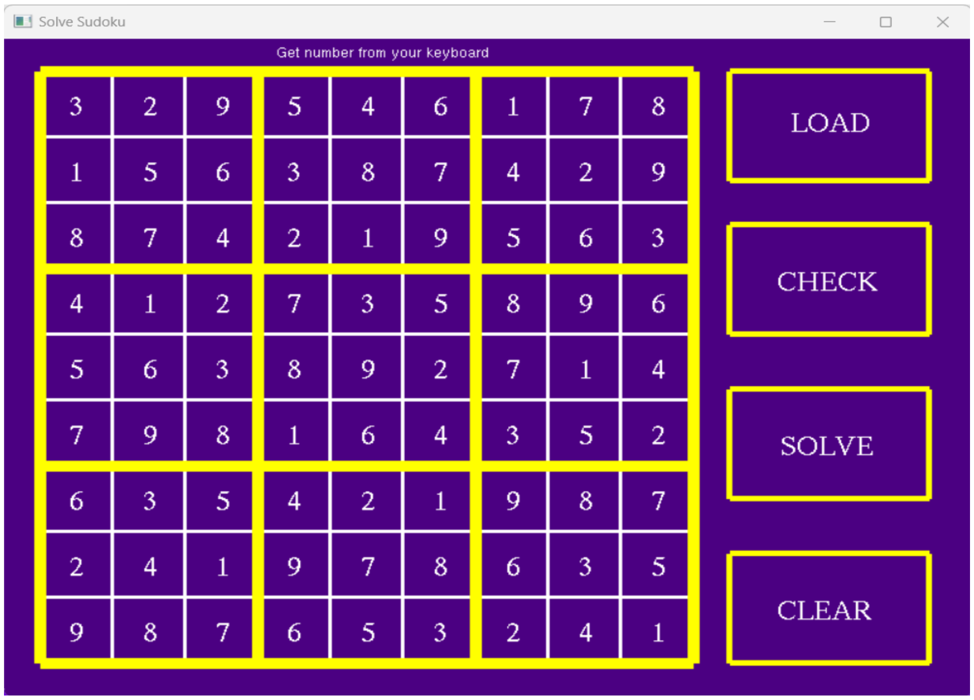
Get number from your keyboard

3	2				6			8
	5			8			2	
8					9	5		
4	1			3			9	
5	6	3		9		7	1	
					4	3		
6			4	2	1	9		
	4	1	9		8		3	
								1

Figure 04: After Loading

6.5 After Solving Sudoku Board

Players can click the "Check" button to verify whether the current state of the grid is a valid solution. The "Solve" button triggers an algorithm to attempt solving the Sudoku puzzle and displays the solution. The "Clear Board" button resets the grid, allowing players to start fresh.



Solve Sudoku

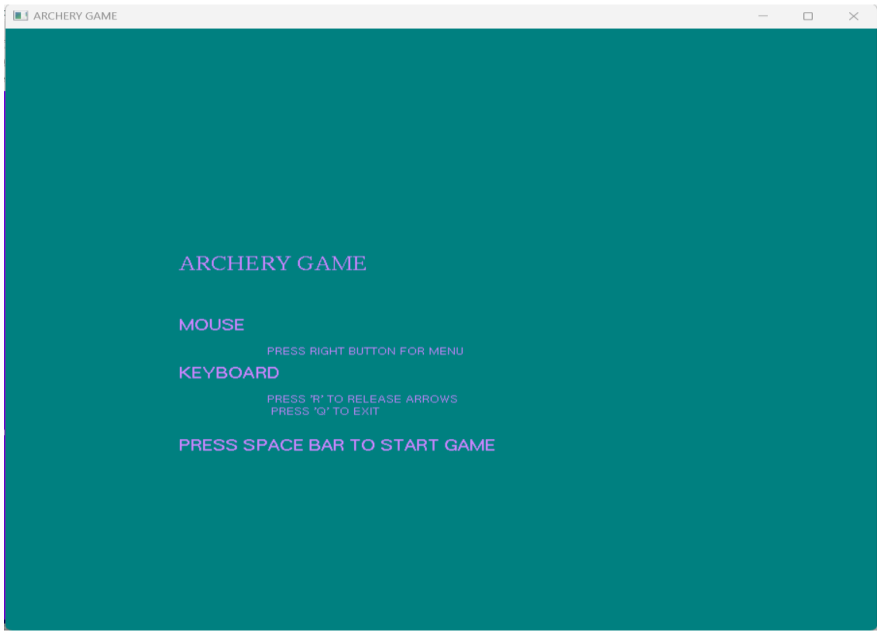
Get number from your keyboard

3	2	9	5	4	6	1	7	8
1	5	6	3	8	7	4	2	9
8	7	4	2	1	9	5	6	3
4	1	2	7	3	5	8	9	6
5	6	3	8	9	2	7	1	4
7	9	8	1	6	4	3	5	2
6	3	5	4	2	1	9	8	7
2	4	1	9	7	8	6	3	5
9	8	7	6	5	3	2	4	1

Figure 05: After Solving

6.6 Archery Game Preview:

It shows the game preview of Archery game and make the player know about the instructions using mouse and keyboard.



ARCHERY GAME

ARCHERY GAME

MOUSE

PRESS RIGHT BUTTON FOR MENU

KEYBOARD

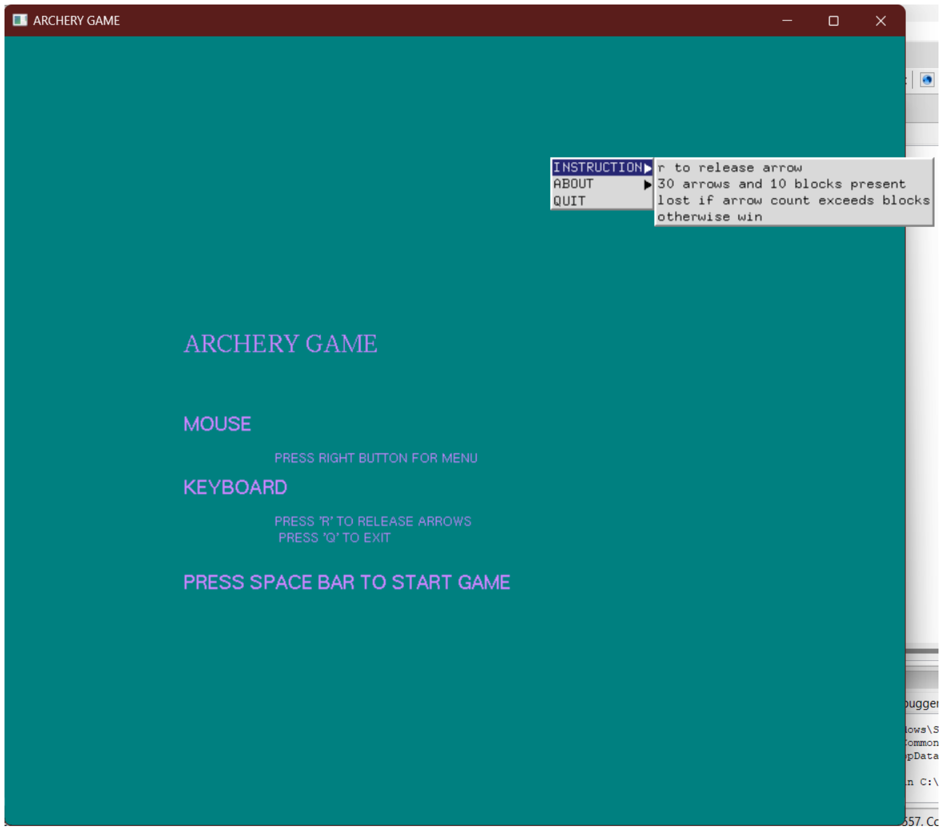
PRESS 'R' TO RELEASE ARROWS
PRESS 'Q' TO EXIT

PRESS SPACE BAR TO START GAME

Figure 06: Archery Game Preview

It explains that the right mouse button is used for the menu, 'r' key releases arrows, and 'q' key exits. The page encourages starting the game with the space bar. This page familiarizes players with the game's mechanics before they dive in.

After the right clicking of mouse, the instruction, about and quit will be shown.



```
r to release arrow
30 arrows and 10 blocks present
lost if arrow count exceeds blocks
otherwise win
```

ARCHERY GAME

MOUSE

PRESS RIGHT BUTTON FOR MENU

KEYBOARD

PRESS 'R' TO RELEASE ARROWS
PRESS 'Q' TO EXIT

PRESS SPACE BAR TO START GAME

Figure 07: Instruction

6.7 Initial Interface of Archery Game:

After pressing the space bar the Archery Game will start. Initially some target with magenta color square shown on the interface. The target object are moving udward and downwand. An arrow will also be displayed which is movable like square target. Number of arrows is zero when the game starts. As the game starts, both target square and arrow start moving vertically so that it creates some difficulty for the player to precisely reach to the target.

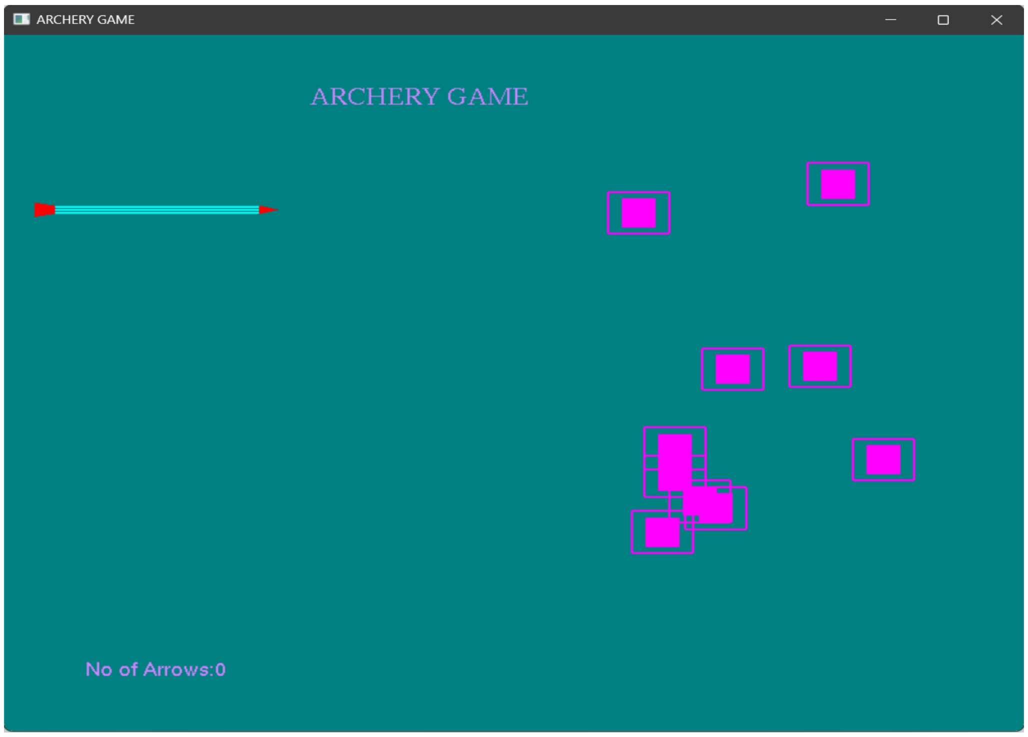


Figure 08: Initial Interface of Archery Game

6.8 After hitting some target:

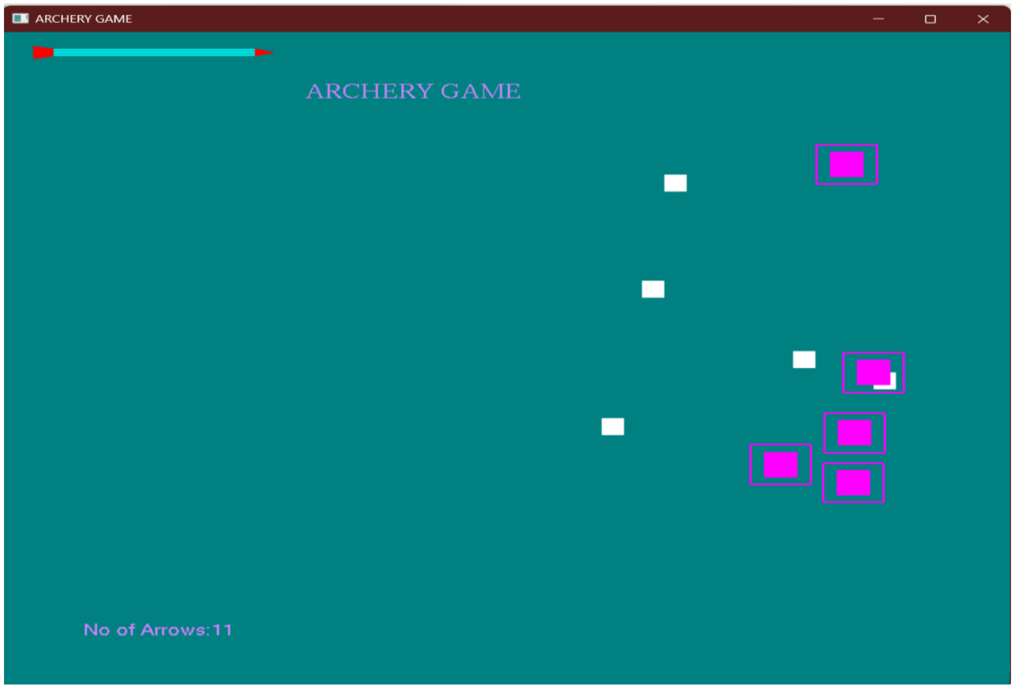
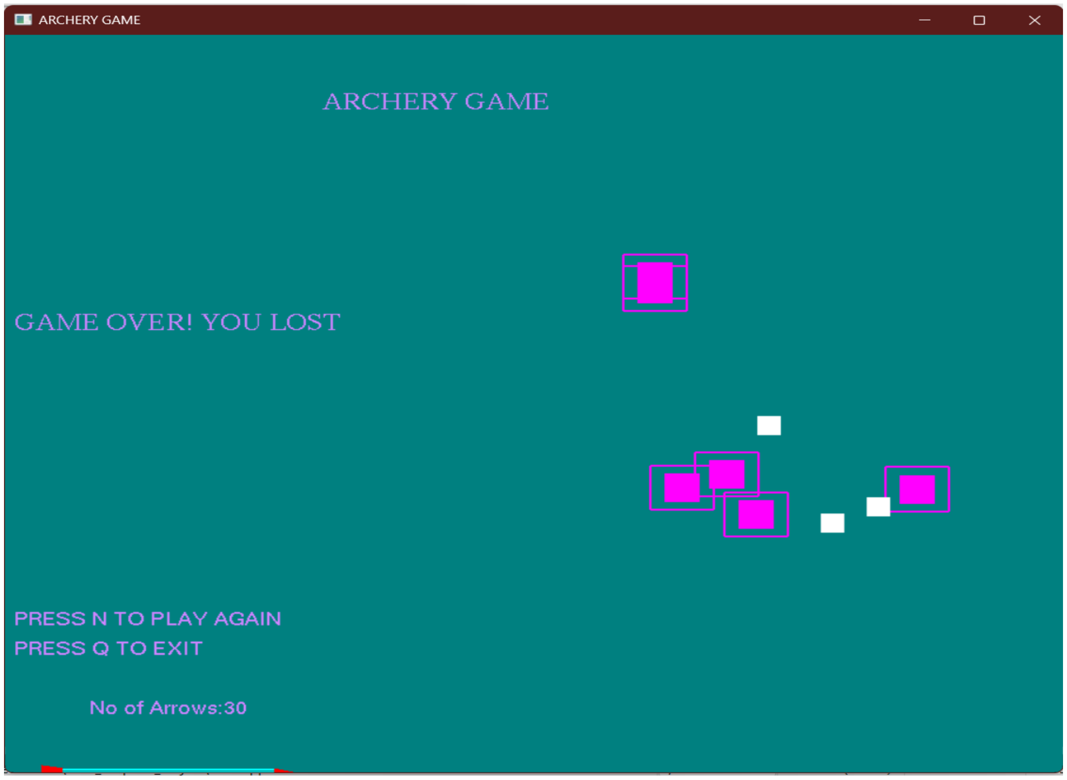


Figure 09: After hitting some target

If an arrow hit a target then it will be displayed as white colored, which indicates that the target was hitted. It explains that the right mouse button is used for the menu, 'r' key releases arrows, and 'q' key exits. The page encourages starting the game with the space bar. This page familiarizes players with the game's mechanics before they dive in.

6.9 Resulting interface after lost:

If the target remains on the window after throwing thirty arrows then a player will be lost. And displayed “GAME OVER! YOU LOST”. It also instructs player to play again by clicking “N” and for exiting press “Q”.



GAME OVER! YOU LOST

PRESS N TO PLAY AGAIN
PRESS Q TO EXIT

No of Arrows:30

Figure 10: Resulting interface after lost

6.10 Resulting interface after win:

When all target hit before passing all thirty arrows, a player will be win. All the target look like a white square point. The window shows the number of arrow required to complete the game and displayed a message “CONGRATULATION YOU WIN”.

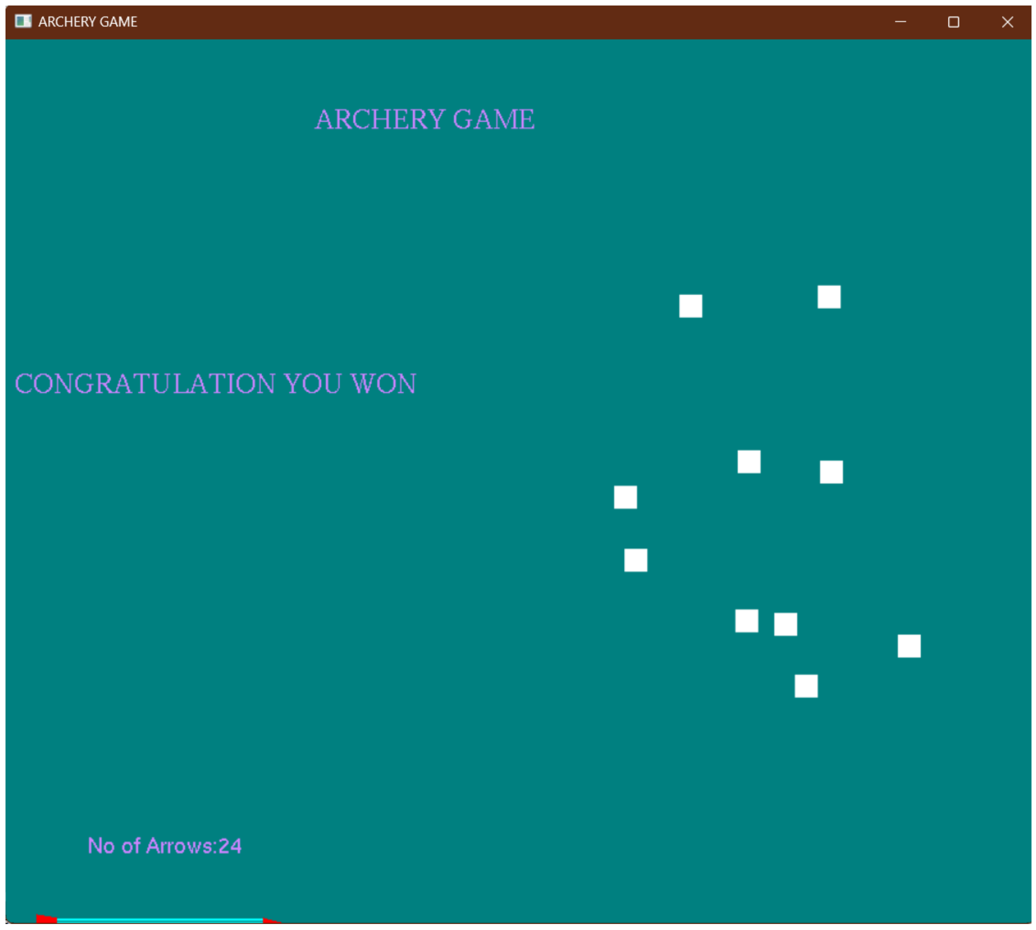


Figure 11: Resulting interface after win

7 Conclusion

In conclusion, "Sudoku Archery Quest: A Fusion of Focus and Precision" is a captivating fusion of Sudoku and archery challenges using OpenGL and C++. The project showcases technical prowess and creativity, offering players a unique experience where logical thinking through Sudoku transitions seamlessly into testing aiming precision in archery. From the initial preview to the arrow mechanics and target interactions, the implementation demonstrates design and coding, resulting a gameplay flow. By combining visuals, intuitive controls, and strategic gameplay, the project ensures an engaging experience that keeps players entertained and challenged. It highlights the potential of merging diverse concepts in gaming, exemplifying the versatility of computer graphics and development.