Department of Computer Science and Engineering
University of Dhaka
24 December, 2023

Project Report:
Fundamentals of Programming Lab(CSE-1211)

Project Name:
Quidditch-Arcade

Team Members:
Nafiul Alim Adeeb (Roll: 23)
Sharmila Surovi (Roll: 39)

1. Introduction:

Quidditch Arcade is an enthralling 2D game developed using **C** language and the **SDL2** library, offering a magical adventure inspired by the renowned Quidditch sport from the beloved Harry Potter series by JK Rowling. Unlike the traditional Quidditch gameplay, this single-player experience places the player in the role of a Chaser, tasked with scoring by skillfully navigating and throwing the Quaffle through bewitched floating hoop. The game revolves around time-sensitive challenges where successful scoring not only accrues points but also extends the player's gameplay duration. However, navigating the sky isn't without peril—Bludgers pose a threat, causing time loss upon collision. The appearance of the elusive Golden Snitch after significant scoring adds an exciting dimension, rewarding players with a substantial point boost.'Quidditch Arcade offers varying difficulty levels, allowing players to choose between two different skill settings. With intricate movements requiring finesse, 'Quidditch Arcade' offers a thrilling and engaging experience tailored for Harry Potter enthusiasts seeking an innovative and immersive twist on the beloved wizarding sport.

2. Objective

The objective of our project was to use our knowledge of C programming and leverage the SDL library to produce an interesting and exciting game experience while retaining the originality of game creation. This attempt aims to practically apply many functions and features inherent in the C language, developing a greater understanding through practical application. In addition, the project aimed to create an engaging and enjoyable game while also providing insights on code modularization for future enhancements. The ultimate goal was not only to build an engaging game but also to provide the basis for potential future improvements, allowing for continuous growth and expansion for improved usability and functionality.

3. Project Features

We attempted to add many features to our project that utilize the major purpose of our application. The main focus was on the Gameplay features, as it will help the players experience smoother gameplay, but that too not without facing challenges. Many features have been added to this game to improve the user experience and make the game more customizable and entertaining. The features are:

Gameplay:

Game Interface

1. Rotation

The player controls the main character through the arrow keys. However, the movement is not limited to left/right and up/down as it is possible to rotate the character clockwise and counter-clockwise.

2. Bludgers and collision

All through the game, some bludgers float around the screen. The number of bludgers depends on the difficulty of the game. These bludgers are generated through the implementation of **linked lists**. The player has to avoid these floating bludgers and fly along the screen because if they collide there will be a time penalty for the player. The collision between the main character object and the bludgers has been implemented through the '**Bounding Box Collision**' method which is used widely in game development. When the player collides with a bludger, their position is reset to the top-left of the screen.

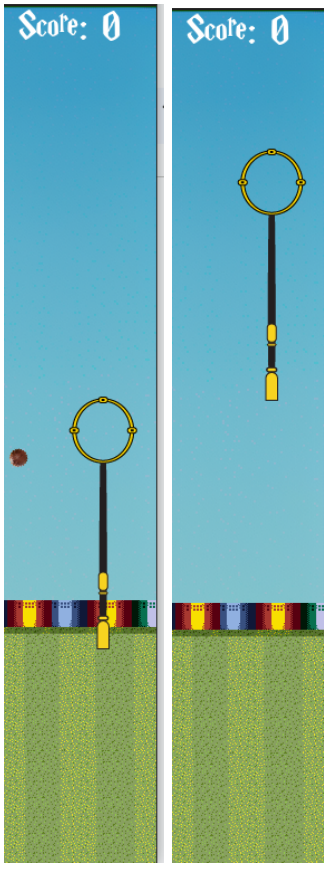
Player position reset after collision with bludger

3. Quaffles

The player can shoot Quaffles with a mouse click or keypress. These quaffles are also generated through the implementation of linked lists. Moreover, when a quaffle goes off-screen or it collides with a bludger, it is deleted from memory through the list. These quaffles are used to score points for the player.

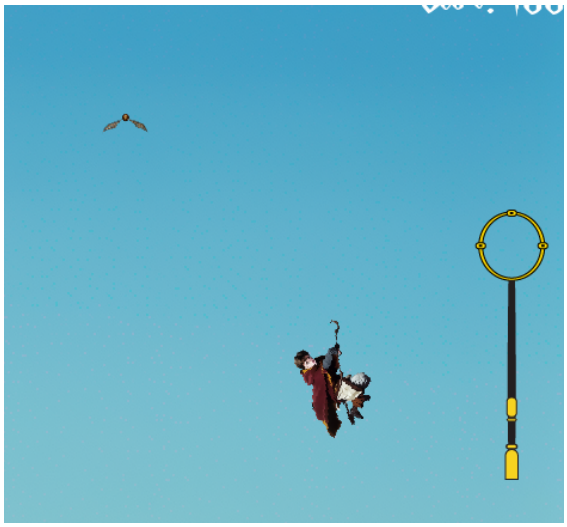
4. Vertically Moving Hoops and Score

The game features a vertically moving hoop at the right-hand side of the screen, as it results in a more challenging gameplay. When a quaffle is shot through the open space of the hoop, the player receives 10 points and the clock is incremented by 3 seconds. The score detection of the moving hoop has also been implemented by the box collision method. The speed of the hoop increases with difficulty level.



5. Snitch

When the player reaches a score that is a multiple of 50 e.g. 50, 100, 150, etc. the golden snitch appears on the screen at a random point and stays on the screen while flapping its wings for a total of 5 seconds. If the player can reach it within that time, then he/she receives a score of 60 points, along with an increment in the clock. The snitch has also been implemented using a linked list.



6. Countdown and Scoring Mechanism

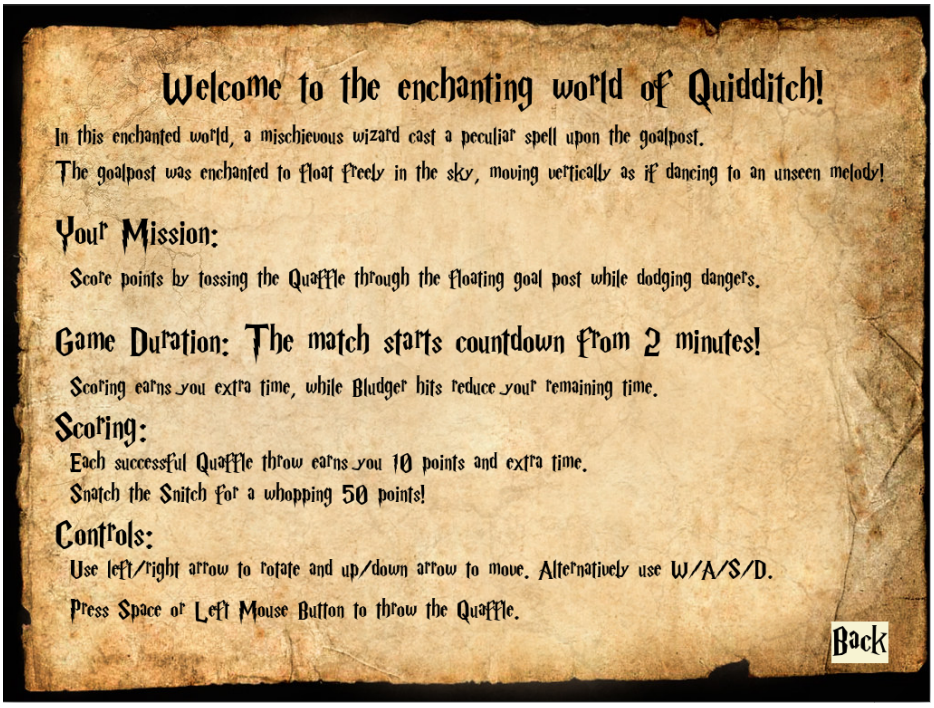
The game always begins with a countdown of 2 minutes. But, actually the player has more time than that as it increases with each score and also when the snitch is caught. The player can play the game and score points until the countdown is over. There is also a time bar at the top of the screen to provide a visual for the remaining time. The bar changes color to indicate the amount of time remaining left.

Main Menu:

The main menu offers five options: “Play”, “Instructions”, “Best Score”, “Settings” and “Exit”. Players can start playing the game, see the instructions and high score, choose the difficulty level from settings, and exit the game by clicking on the necessary icon. The icons are described below:

Instructions Page:

Here the necessary guideline for playing the game is given.



Best Scores:

The best score feature has been implemented with the help of '**File Handling**' in C. A text file stores the score after a game is over, and it is compared with previous scores to determine the best score.

Settings:

In the settings option, a player can easily turn on/off the sound and music. Moreover, the difficulty level can also be chosen from here. The default difficulty level is “Easy”.

End Game Page:

After the game's time runs out, an impactful 'Game Over' screen appears, prominently displaying the player's score. This end-game screen provides options for the player to either return to the main menu or exit the game entirely.

4. Project Modules

1. **globals.h :** The globals.h file defines constants, variables, and structures that are globally accessible across the project.
2. **Init.c :** Here we have declared the functions to start up the game. Functions are:
    - initVid() : Initialize SDL video and image.
    - initAudio() : Initialize SDL audio.
    - destroyWindow() : Destroy window, quit SDL and clean memory.
3. **Assets.c:** This library is used for loading necessary assets for running the game. The function here is:
    - loadComponents() : Load all the necessary images, sprites and audio for the game.
4. **Physics.c :** This library contains crucial functions and structures that manage various game functionalities. Here are they:
    - double sinD() : Converts sin degree to radian.
    - double cosD() : Converts cos degree to radian.
    - int collision() : Checks collision between two objects.
    - void movePlayerUD() : Up down movement of player.
    - void movePlayerLR() : Left-right movement of player.
    - void rotateBy() : Rotate an object by a specified amount of degree.
    - void harryMovements() : Constraint the main character to stay in the window .
    - void Launchprojectile() : Add new quaffle to the linked list.
    - void moveQuaffles() : Quaffles movement logic .
    - void ShootQuaffle() : Throw quaffle by creating a new element in the quaffle linked list.
    - void moveBludgers() : Bludger movement logic.
    - void hoopMovements() : Hoop movement logic.
    - void scoring() : Update the score when the conditions are met.
    - void generateSnitch() : Generate a snitch at a random x and y position.
    - void toggleSnitchTexture(): Loop through sprite images of snitch.
    - void snitchBehavior() : Call the necessary functions to generate and delete a snitch.
5. **Engine.c:** This module appears to contain functions responsible for managing different aspects of the game. The functions are:
    - void instructions() : Opens the instructions page when clicked.
    - void storeScore() : Stores the current game score to a file.
    - int sortScores() : Sorts the saved scores to find the highest.

- void leaderboard() : Opens the highest score page. Relies on storeScore() and sortScores.
- void settings() : Opens the setting page from the Main Menu.
- void MainMenu() : The function to display the Main Menu.
- void addBludger() : Adds some bludgers to a linked list.
- void StartGame() : Initializes initial conditions to start the game.
- void GameOver() : Displays the Game Over screen, once the countdown is over.

6. **drawing.c:** This module is responsible for rendering graphical elements and texts within the game interface, providing the visual backbone for user interaction and game display. The functions here are:
    - void DrawText() : Draws text to the renderer.
    - void Draw() : Draws background images to fill the whole window.
    - void DrawObject() : Draws Object structs.
    - void DrawDynamicObject() : Draws dynamic objects in linked list i.e. bludgers.
    - void DrawLife() : Draws the timer bar
    - void DrawScreen() : Draws the main screen and responsible for onscreen objects.
7. **events.c :** This one manages key dates, responds to user inputs and handles SDL events to control interactions within the game environment. Description for each functions are:
    - int Key() : This function retrieves the state of a specific key represented by the input value, returning ‘TRUE’ if the key is pressed and ‘FALSE’ otherwise.
    - void HandleKey() : This function manages and updates the state of keys based on the provided symbol and whether it’s pressed or released, ensuring accurate handling of key events.
    - void HandleEvents() : Responsible for handling various SDL events.

8. **gameloop.c :** The gameloop.c file orchestrates the game flow, updating game elements, handling user input, and managing the game loop cycle. Functions here are :
    - void UpdateGame() : Handles game state updates, including time tracking, player movements, interactions, and scoring mechanisms.
    - void gameLoop() : Manages the game's loop cycle, ensuring consistent updates, handling events, and refreshing the screen to maintain gameplay continuity.

9. **linkedList.c :** The linkedlist.c module manages a linked list data structure, offering various functions to manipulate and handle linked list operations efficiently. Here the functions are:
    - Node createNode(data):* Creates a new node in the linked list with the provided data value, initializing the next pointer to NULL.
    - Node insertBeginning(head, data):* Inserts a new node containing the provided data at the beginning of the linked list.
    - Node insertEnd(head, data):* Adds a new node with the given data at the end of the linked list.
    - Node insertAtPosition(head, data, position):* Inserts a node with the provided data at a specified position in the linked list.
    - Node deleteBeginning(head):* Removes the first node from the linked list.
    - Node deleteEnd(head):* Deletes the last node from the linked list.
    - Node deleteAtPosition(head, position):* Removes a node from the specified position in the linked list.
    - void display(head): Displays the elements present in the linked list.
    - int countNodes(head): Counts and returns the total number of nodes in the linked list.
    - Node searchNode(head, data):* Searches for a node containing the provided data in the linked list and returns it if found, otherwise returns NULL.

5. Team Member Responsibilities

Team Member 1: Nafiul Alim Adeeb (Roll: 23)

1. Contributed to the game's concept and suggested the arcade game style.
2. Structured Game code in C language
3. Version Control (Git/GitHub)
4. Illustrated game assets using design tools (Adobe Illustrator)
5. Organized modular code structure and file handling
6. Managed file systems and data
7. Implemented complex physics calculations
8. Developed the game engine and implemented the game loop
9. Collision Related Works
10. Worked on fonts and text display in the game
11. Managed texture and projectile of Quaffle and bludgers.
12. Modules: **physics.c, engine.c, linkedList.c, gameloop.c**

Team Member 2: Sharmila Surovi (Roll: 39)

1. Conceptualized and proposed the game idea.
2. Generated necessary game assets.
3. Resized assets and background images for the game.
4. Created sound effects and audio elements.
5. Developed structured game code in C language.
6. Designed the main menu and end menu pages.
7. Merged the main menu with the core game loop.
8. Conducted extensive code testing and debugging.
9. Produced progress bar.
10. Handled mathematical calculations and algorithms.
11. Handled high score calculation.
12. Modules: **globals.h , init.c, assets.c, drawing.c, events.c**

6. Platform, Library & Tools

- **Linux** - The operating system on which the project was developed, offering a versatile environment for software creation.
- **Visual Studio Code** - A robust, cross-platform code editor known for its versatility and extensive plugin ecosystem, facilitating smooth coding workflows.
- **C Language** - The fundamental programming language used in the project, offering flexibility and efficiency in coding structures.
- **SDL2 Library** - Simple DirectMedia Layer providing low-level access to audio, keyboard, mouse, and graphics hardware, crucial for game development.
- **Git/GitHub** - Git, an open-source version-control tool, collaborated with GitHub, a platform that enabled easy code hosting and collaboration among team members.
- **Adobe Illustrator** - A powerful vector graphics editor from Adobe, utilized for crafting intricate and detailed visual elements for the game's design.
- **MP3 Cutter** - Tool used for editing audio files, enabling precise adjustments to the game's sound effects and music.
- **Online Image Resizer** - Web-based tool for resizing and optimizing images to fit specific game requirements, ensuring smooth integration.

- **Online Music Converter** - Utilized to convert and adapt music formats to suit the game's needs, enhancing the auditory experience.
- **Scenario AI** - AI-driven tool employed to simulate scenarios and test game functionalities, ensuring a smooth and immersive gaming experience.
- **PNGWing** - An online repository providing a wide array of PNG images, used for sourcing various game assets and visual elements.

8. Limitations

Primarily, due to time constraints, we were unable to build the multiplayer feature, which was a target we had set for ourselves. Furthermore, our limited knowledge with graphic design tools limited the scope of our game's design. The absence of a higher-level engine hampered development and may have had an impact on overall design quality. However, in the future, we intend to optimize our codebase to improve game performance, with the goal of increasing speed and further development. This optimization procedure will also include the much-desired multiplayer capability, which will enhance the game experience for our audience.

9. Conclusions

This project has been an incredible learning experience for us in C programming and game development with SDL library." It felt like beginning from scratch and delving deeply into generating models, animating them, and building an entire game. A huge highlight was working together to solve issues and stimulate innovation. While online resources were helpful, working with an older library like SDL meant stepping into unfamiliar territory, spending significant effort polishing every component, from movement dynamics to game fairness.
Aside from coding, this project was a masterclass in teamwork and perseverance in the face of adversity. It has made us feel considerably more capable, and ready to take on new technological landscapes and complicated challenges. This practical training has prepared us for the ever-changing world of software development. Overall, it's been an exhilarating trip full of obstacles, exciting discoveries, and vital lessons in teamwork and problem-solving.

10. Future Plans

- ➔ **Multi-platform Compatibility:** Expand the game's reach by developing versions compatible with various platforms like Android, iOS, and Windows, widening accessibility to a broader audience.

- ➔ **Custom Graphic Design and Music:** Plan to craft fully customized graphics and music to enhance the game's unique feel and visual/audio experience.

- ➔ **Expanding Game Levels and Features:** Extend the game's levels and intricacies, aiming to incorporate the much-desired multiplayer feature and enhance interactivity within the gameplay.

- ➔ **User Engagement and Feedback:** Encourage user interaction by soliciting feedback to identify areas for improvement, subsequently implementing features based on user suggestions to enrich the gaming experience.

These plans aim to elevate the game's quality, accessibility, and engagement, laying the foundation for an even more immersive and enjoyable gaming experience.

Repositories

GitHub Repository: https://github.com/jon0iko/Quidditch-0.1
Youtube Video: https://youtu.be/6UL4YcV2fnY

References

[1] The SDL Library: https://wiki.libsdl.org/SDL2/FrontPage

[2] The SDL Library: https://wiki.libsdl.org/SDL_image/FrontPage

[3] Lazy Foo SDL Tutorials: https://lazyfoo.net/tutorials/SDL/

[4] PNGWing: https://www.pngwing.com/

[5] Sound effect: https://mixkit.co/

[6] Book: Kernighan and Pike's "The Practice of Programming". (https://en.wikipedia.org/wiki/The_Practice_of_Programming)

[7] Photoroom: https://www.photoroom.com/tools/remove-object-from-photo

[8] Picture Resizer: https://iloveimg.com/

[9] Scenario: https://www.scenario.com/

[10] MP3 Cutter: https://mp3cut.net/