University of Dhaka

Department of Computer Science and Engineering

Project Report:

Fundamentals of Programming Lab(CSE-1211)

Project Name:

Cannon Fight

Team Members

Md. Mahmudul Hasan, Roll-10

Afia Lubaina, Roll-14

Ahona Rahman, Roll-59

Introduction

“Cannon Fight” is a game that can be played in both single-player and multiplayer modes. Here, the player has to hit his opponent with a cannon by pressing the spacebar. The bomb is thrown in projectile motion, and the player has the opportunity to set the direction of the cannon and throw the bomb so that he can increase the probability of hitting his opponent, while the player also has the ability to move down the aisle by a ladder and walk to and fro so that he can protect himself during his own turn only. The player is given specific timing for fixing the angle of the cannon and throwing the bomb at his opponent. There is also a special gift box of which the player can choose during the game, the gift box provides any of three features: health increase, double damage, twice turns. The player who gets hit will lose life, and if it loses several turns and the life bar is empty, the player is defeated in the game. If the player manages to hit more and defeat his opponent, he wins.

Objective

Our major goal is to use the SDL (Simple DirectMedia Layer) library and C/C++ language to construct a simple graphical game project in a practical field. However, in order to create and use it, we wanted to increase our C/C++ understanding.

Project Features

The project, Cannon Fight’s features are divided into two categories. One is UI (User Interface) features, and the other is game-play features. User interface (UI) features are the series of screens, pages, and visual elements like buttons and icons that enable a person to interact with the game. On the other hand game-play features are those which can be used or achieved or visualized while playing the game. Many features are added in this game to

improve the user experience and make the game more flexible and interesting.
The features are:

User Interface Features:

1. Main Menu Features:

The main menu offers four options: “NEW GAME," “LOAD GAME," “GUIDE,"“, and "EXIT.” option.
Players can start a new game, load previous games if any are already loaded, select the guide section for instructions and control features, and exit the game by selecting the exit menu from the main menu page. The icons on the main menu page are described below:

Fig : Starting screen

1.1 New Game:

This option will take a player to the game. All the previous game data will be lost after that, and the player has to start again. To start a new game player has to select the “New game” icon. A player can choose both modes just as the followings:

Modes:

Players can choose between multiplayer and single player modes.

- ➢ In "2 players" mode, two players are allowed to play against each other.
- ➢ In “1 player” mode, a player is able to play against the computer. The features regarding the gameplay are the same, but in “1 player” mode, the computer plays against the player and automatically throws cannon balls within a specific time allotted.

1.2 Load Game

This option will take a player to the previously loaded game that someone left before. That means he gets the opportunity to start a game from the place where he left the game before. The player playing for the first time doesn’t get this feature.

1.3 Guide

This option is for game instructions. Here one can know how to play the game and which keys and mouse option one must use to play the game. There are in total 3 pages for the gameplay instructions.

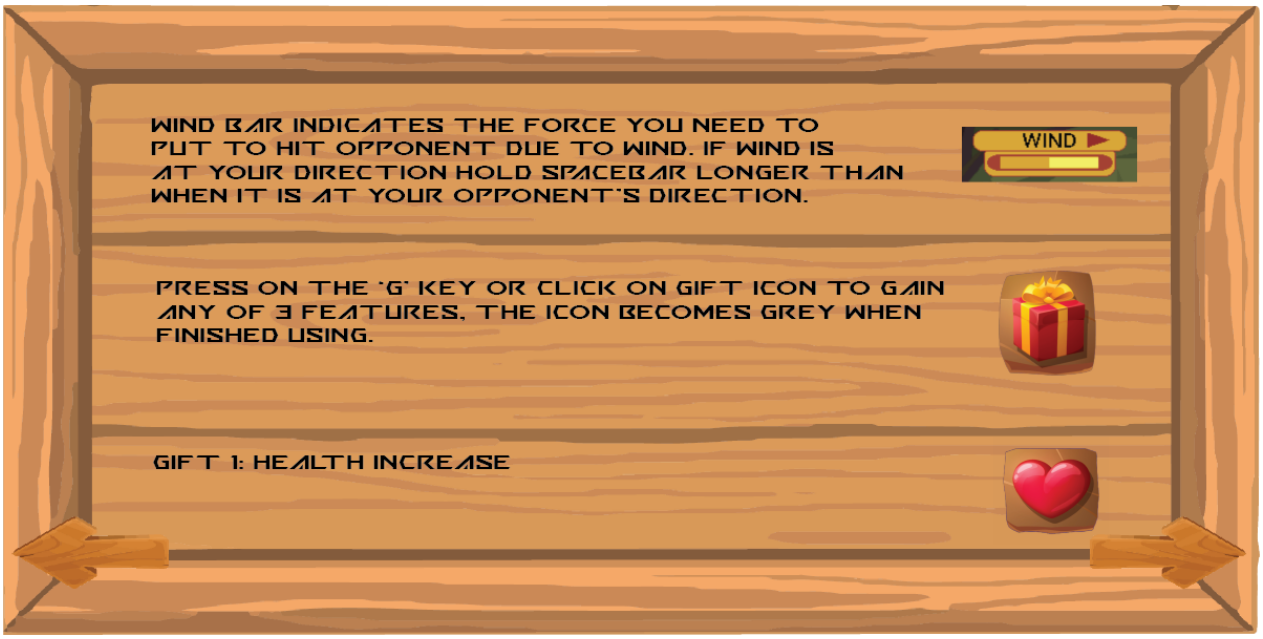

Fig: Guide menu

1.4 Exit

This option allows a player to exit out of the game.

Fig: Exit Menu

2. Pause option

This option allows one to go back to the main menu and choose the main menu options. Here the game is saved automatically and one can continue playing if he clicks on the load game option in the main menu.

Fig.:Pause button

3. Game over features

This appears as the game ends. If the player wins against the computer, a screen with " You win" writing appears. If the player loses, it shows "You lose". In 2 player mode ‘’left player wins” or “right player wins” appears on the screen.

Fig. : Game Over Screen for single player (if win)

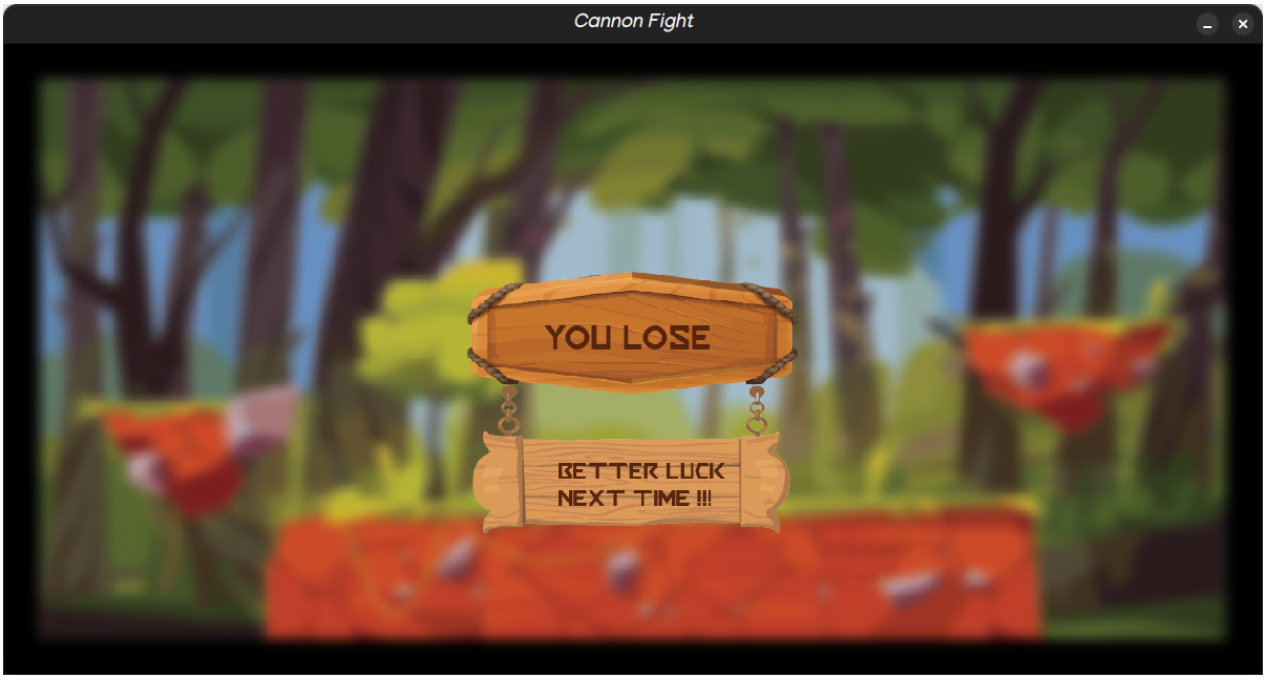

Fig. : Game Over Screen for single player (if lose)

Fig :Game Over Screen for double player

Game-Play Features:

These characteristics include the game's options and dimensions. These are the following:

Fig: Game interface

1.Life feature:

The player loses life as it gets hit by the cannon. The life bar represents how much life each player possesses after getting hit by the cannon. The red bar decreases if a player gets hit by the opponent in any part of its sprite’s body.

Fig: Life Bar

2.Time countdown

Each player has a set amount of time to position himself and throw the bomb at his opponent in a specific direction, and the sprite can also move down the ladder and get from the aisle to the land in that time. The time bar denotes the amount of time left for each player's turn, as the blue line decreases as per the amount of time spent.

Fig: Time bar

3. Cannon angle

The cannon can be adjusted to its desired angle using the up and down arrows of the keyboard. Players can move the cannon's shooting angle to their desired position using the up and down keys.

Fig. : Cannon angles

4.Sprite Moving and climbing ladder

The sprite can be moved down the ladder and moved forward or backward using the keys. The W and S keys are used to move the sprite up and down the ladder, and the A and D keys are used to move the sprite right and left. This way, the player can protect itself during the time of the opponent's shoot and also move during the time of shooting. But the player is not allowed to move during the opponent's turn.

Fig. : Sprite movement

5.Wind bar

While shooting, players must keep an eye on the amount of wind blowing in their direction as well as the amount of thrust they apply. If the wind is blowing strongly in his direction, the player must apply more thrust to the bomb. When there is less wind, there is less thrust. If the wind is flowing in the opposite direction, then the player has to throw with less thrust.

Fig: Wind bar

6.Thrust bar

The thrust bar displays the amount of thrust put on the bomb to be thrown. The bar increases as the space bar is held and stops at the release of the space bar.

Fig: Thrust bar

7.Sound effects

Different sound effects are used in this game. Besides the main background music, sounds are included in the bomb hits as the player loses life and hits the opponent successfully. Effects are added as the player gains life and uses the double throw option. Moreover, there are winning and losing sounds. Some extra sound effects include click sounds and sounds while the mouse pointer hovers over options.

8.Explosion

The bomb explodes after a successful hit on its opponent. A successful hit is when the bomb hits any part of the sprite's body.

Fig: Explosion

9.Gift box

This small gift box icon allows a player to get any of the three gifts, which include health increases, double damage, and two turns at a time. In single-player mode, the computer automatically choses gift icon after losing half life. In multiplayer mode , both players can choose a gift box any time while playing by pressing 'G' key or clicking on the gift icon.

Fig: Gift Box & gift items

Project Modules:

The project is divided into 11 header files. The descriptions of header files and it's functions are given below:

1.#include "Game.hpp" :

That's the heart of the game. The game is controlled and played here.

1.1- Game():

This constructor is used to initialize and load all the assets of the game in order to use them later on.

1.2- init():

Before starting a new game, this is called to give all the variables the necessary initial value.

1.3- resume():

To load a previously played game, it is called.

1.4- playMusic():

Starts the music of the main game.

1.5- stopMusic():

Stop the game music.

1.6- handleEvents():

Here all kinds of interactions are done with humans and computers. All the keyboard and mouse inputs are taken care of in this function. Most importantly the "Artificial Intelligence" to implement the "one player" feature is coded there. It handles all the inputs of computer in the "one player version"

1.7- update():

While playing the game, all the changes are executed here. Every update in the ongoing game is the reflection of this function.

1.8- render():

Renders the game images to the window.

1.9- checkFinishGame():

Checks if the game is over or not.

1.10- clean() :

Clean the initialized assets and SDL

2.#include "TextureManager.hpp" :

2.1- loadTexture() :

Loads every image into the target texture

3.#include "Projectile.hpp" :

The main attraction of the game is the projectile motion of the bomb that is thrown from the cannon at a particular angle, and this function controls them.

3.1- Projectile():

Initializes variables that will be used later.

3.2- update():

This function updates the position of the bomb in a projectile motion line. There we used a formula from physics to implement the idea. The formula is given below:

$$v_x = u \times cos\alpha + v_w$$
$$v_y = u \times sin\alpha$$
$$x = v_x \times t$$
$$y = v_y \times t - \frac{1}{2} \times g \times t^2$$

3.3- render():

Render the changes of the bomb on the screen.

4. #include "Cannon.hpp":

It controls the cannon of the game.

4.1- cannon() :

It initializes the position and angle of the cannons

4.2- update() :

Updates the angle of cannon.

4.3- render() :

Renders the changes of the cannon to the screen.

5. #include "Sprite.hpp":

The whole animation and motion of the sprites is looked after here.

5.1- Sprite() :

Initializes the animation and position of the right and left sprites.

5.2- update() :

When the sprite moves, it handles it's new position and animation of that motion.

5.3- render() :

Renders every change of the sprites

6. #include "TTF.hpp" :

Shows and renders true type fonts on the screen.

7. #include "Collision.hpp":

The collision of the bomb and sprite is controlled in this header file.

7.1- Collision() :

Creates the collision boxes which will be used to detect the collisions

7.2- checkCollision() :

The logic behind detecting collision is coded in that function which tells if there is any collision or not.

8. #include "ProgressBar.hpp":

All the bars including speed bar, health bar, wind bar, time bar.

8.1- ProgressBar() :

Creates all the bars at their target position of the screen.

8.2- init() :

Initializes the values of the bars.

8.3- update() :

Updates the bar's values as the game progress.

8.4- render() :

Renders the changes of the bars to the screen.

9. #include "GameEngine.hpp":

The oneplayer that is against computer mode is implemented here. The moves of the computer are decided here.

9.1- GameEngine() :

Initializes the AI to control the computer moves.

9.2- getInfo() :

The velocity and angle of projection for the computer to hit the opponent is decided here. We derived a physics formula to calculate this. The formula is given below:

$$v_x = u \times cos\alpha + v_w$$

$$v_y = u \times sin\alpha$$

$$y = \frac{x \times v_y}{v_x} - \frac{g \times x^2}{2v_x^2}$$

We added a little error here so that computer makes some mistake to hit the opponent to make the game fair.

9.3- getPosition() :

Here the position of the sprite for the computer is computed.

10. #include "GiftBox.hpp":

Controls the gift box surprise features

10.1- GiftBox() :

Set and load the images at correct position,

10.2- init() :

Initializes the some variable

10.3- checkMouseClick():

Check whether a mouse is clicked on the box or not.

10.4- renderFeature():

Renders the feature got by the player

10.5- render() :

Renders the gift box

11. #include "Menu.hpp":

The GUI is implemented here and merged with the main game loop.

11.1- Menu():

Initializes the SDL the other assets of the game.

11.2- PlayMusic() & StopMusic():

Handles music of the menu

11.3- handleEvents() :

Controls the user input

11. 4- render ():

Renders the changes of the menu on the screen

11.5- isRunning() :

Confirms if program is running or not.

11. 6- clean():

Clean SDL

Team Members Responsibilities:

Team member 1:

Mahmudul Yeamim (Roll: 10)

1.Sprite movement, sprite animation. All the works related to sprite movements , i.e sprite climbing up and down the ladder.
2.Game engine - handling computer moves.
3.Creating AI for playing against the computer in one player mode
4.Collision related works.
5. Explosion related works
6. Handling user inputs. This includes , the handling events , the options that a player choses or key that a player presses while playing the game.
7. Saving previous games in the load game section.
8. Implementing game mode - single & double players

Team member 2:

Afia Lubaina (Roll: 14)

1.Texture management: loading textures and image related works.
2. Projectile motion of bomb. This includes the motion of the bomb according to its speed applied and its position.
3. Cannon handling and cannon controlling features. Helping the cannon to move up and down features.
4. Gift box button and the random features of gift box. This includes the random gifts that a player gets when clicked on the button. And their functions.
5. Game background, customized options like different menu bars and option bars designing.

Team member 3:

Ahona Rahman (Roll: 59)

1. Merging main menu with core game loop

Work related to the main menu page's new game, load game, help, exit button, and rendering visuals on the screen. A button's selection will direct the player to the associated page.

2. Progress bar

Produces all of the on-screen bars, including the time, health, wind, and speed bars. Renders the bar and initializes the values. As the game goes on, the bar's values are updated.

3. TTF

True type fonts are displayed and rendered on the screen.

4. Music

In this game, there are two types of sounds. They are UI music and game sounds. Those sound effects will appear at different times. If any buttons click, there will be a clicking sound. For cannon movement, there will be sound. If the ball collides with an opponent, an explosion sound will occur. Finally, there will be two distinct sounds for wins and losses, which correspond to the game over screen.

● ***Platform, Library & Tools:***

1. *Platform*: Linux
2. *Language*: C/C++
3. *Library*: SDL (Simple DirectMedia Layer) library
4. *Tools* : Adobe Photoshop, Pics Art Photo Editor, Mp3 Cutter, Online Music Converter, Online Image Resizer.

- ***Limitations:***

There are some limitations that a player may face while playing the game.

I. Players cannot add their names.
II. There will no longer be the existence of previous game records after a new starting.
III. In each mode, there is only one level.

- ***Conclusions:***

We gained a lot of knowledge from working on this project, but we also encountered several challenges. For us, creating a game is a completely new and beneficial experience. As a result, we had to start from scratch. Through this work, we have given the SDL library a brief introduction. We are now familiar with the creation and animation processes for models. Our capacity for thought and creativity has increased. We have improved our communication abilities through group collaboration. Actually, for us, it was a completely new experience.
The internet, learning materials, and video tutorials have all helped us learn new things. It can be claimed that creating games is a very sensible activity to engage in. It was challenging to deal with each and every aspect of the model to make it user-friendly.
We were required to complete every task on our own. Everything has to be put into action by us. Everything, from the cannon and ball movement to the blast and life damage, had to be done by us. So, from the UI to the gameplay, we had a lot of work to do. Therefore, the insight or experience went much beyond what we had anticipated. Overall, it was a wonderful and novel experience, and we are pleased to have incorporated both our own suggestions and those of our teacher into our own codes.

- ***Future Plan:***

  ➔ Game level extension.
  ➔ Introduce and implement new game features.
  ➔ Take user feedback and improve features according to their opinions.
  ➔ Introduce new environments and scenes.
  ➔ Execute load game feature even if the game is closed totally.
  ➔ Add more options like , bonus, point recording , player name including.

Repositories:

Github Repositories: https://github.com/mahmudulyeamim/Cannon-Fight.git

Youtube video: https://youtu.be/vELsUzH9Quc

References:

SDL Wiki: https://wiki.libsdl.org/SDL2/Tutorials
Lazy Foo: https://lazyfoo.net/tutorials/SDL/index.php
Youtube Channels:
https://youtube.com/playlist?list=PLHJE4y54mpC5_eEz9gCqIkNpU-n_2eyNt
https://youtube.com/playlist?list=PLhfAbcv9cehhkG7ZQK0nfIGJC_C-wSLrx
Freepik: https://www.freepik.com/vectors
Mixkit: https://mixkit.co/free-sound-effects/
GeeksforGeeks: https://www.geeksforgeeks.org/header-files-in-c-cpp-and-its-uses/