University of Dhaka

Department of Computer Science and Engineering

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

Project Name:
WOI-1971(War Of Independence-1971)

Team Members:

Abdullah Ibne Hanif Arean
Roll: FH-12
Registration No: 2019-917-795

Mehadi Hasan
Roll: SH-60
Registration No: 2019-517-843

Introduction

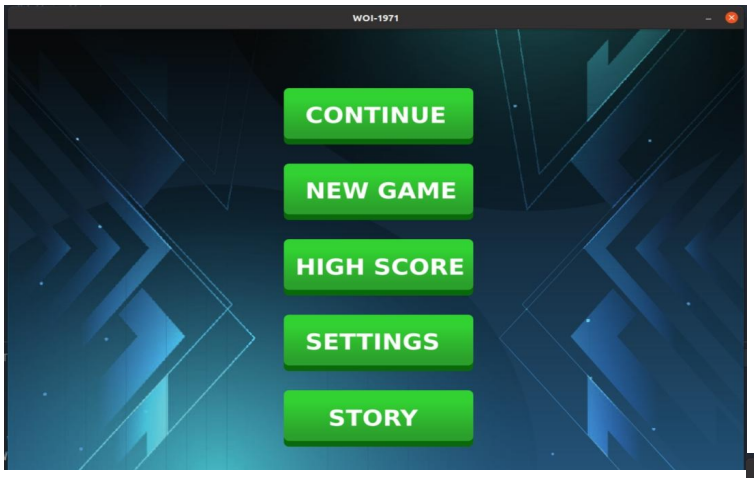
War Of Independence-1971 (WOI-1971) is an SDL-based action game written in C language. Simple, clear, and easily customizable code made this game distinctive. Based on the Bangladeshi people's war of independence against Pakistani military forces in 1971. The bravery of the Bangladeshi Freedom Fighters is honored with great reverence in this game.

Objectives

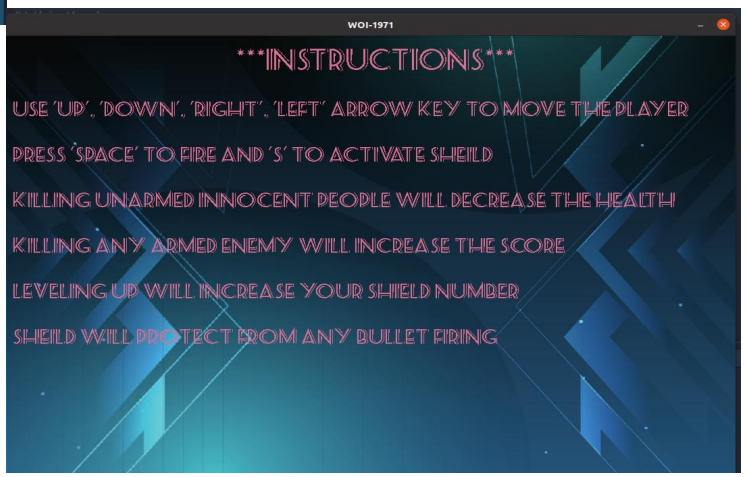
1. Applying the structured **C language** learning in Real-life project
2. Making a simple yet interesting action game on **Bangladesh War of Independence 1971**, tributing the freedom fighters
3. Attractive graphics and animation along with **interactive** sound
4. Well **organized and documented** code to easily understand and further development

Project Features

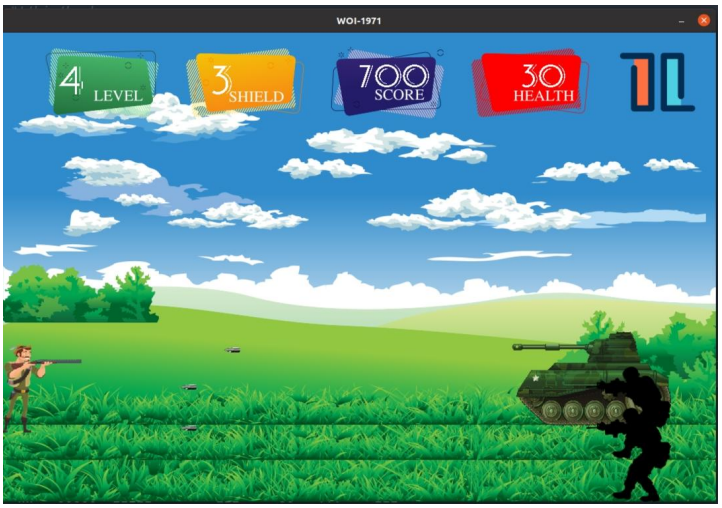
1. **Simple, clear and easily customizable code** in a structured and modularized manner with proper commenting.
2. Easy installation in **Debian-Based** Computers and one-click installation process **Direct Install and Play** (bash script will download everything required)
3. Memory efficient design and **Same performance** in every device (if possible to play)
4. One Command Game **update & upgrade** to the latest release feature with "./update.h command"
5. A Loading page with the **game logo (Custom Created)**



6. Attractive and **dynamic** menu options to control the game

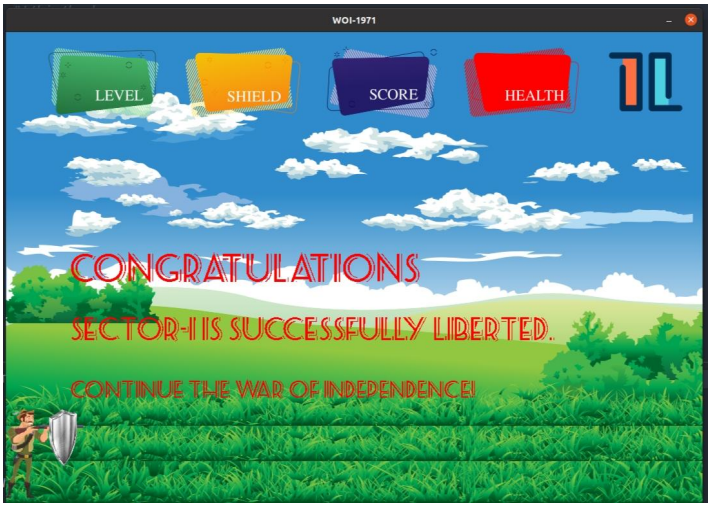


7. Showing **Instructions** on the created window

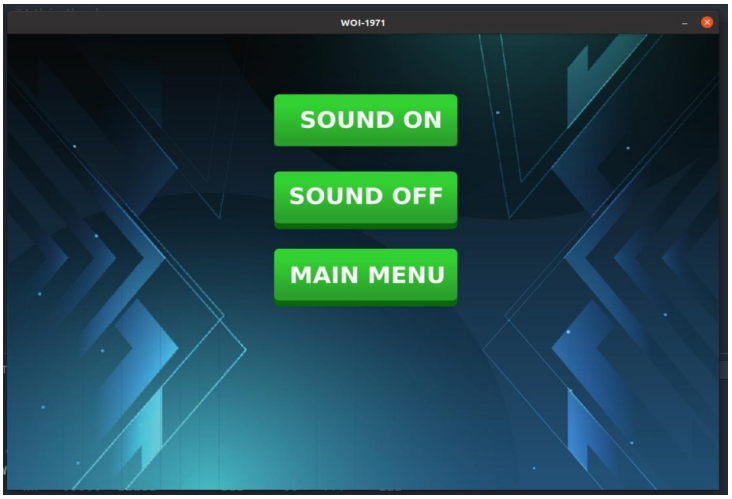


8. **Challenging Levels** with Multiple Enemy Types

9. Here are **Innocent People** too. If the hero kills them, he will get a penalty. And There is also an interesting feature, that players have **shields**, which will protect it from firing damage.



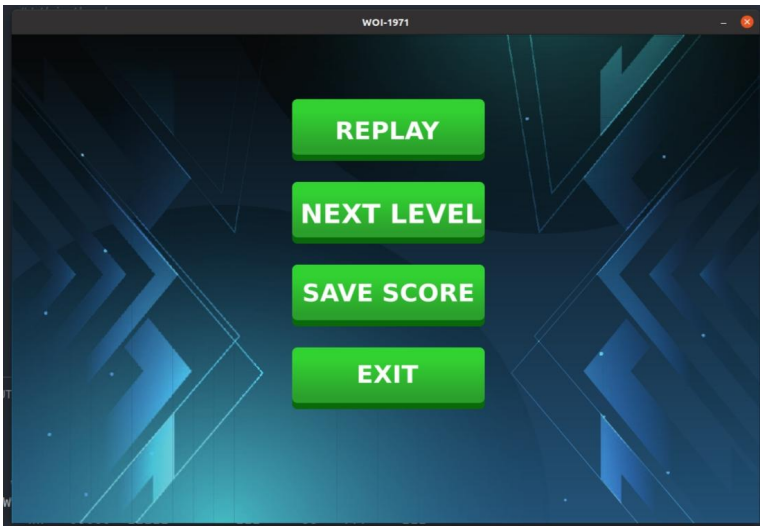
10. **Crispy Sound System** Specifications and sound on/off feature

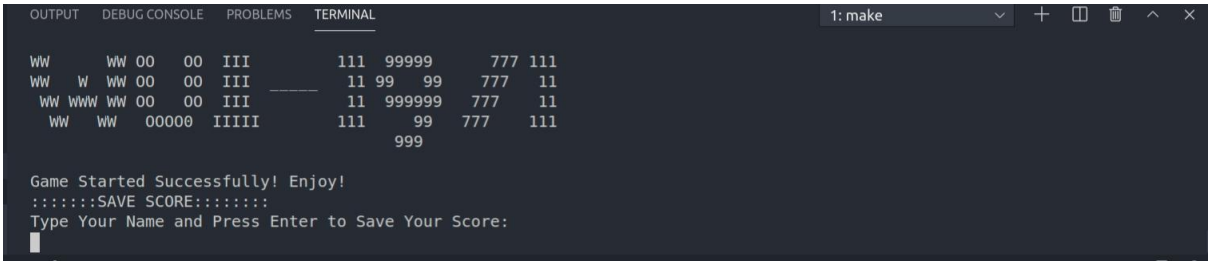


11. There is also a **Story** page where we can see and learn about some stories related to our independence war.

12. Simple yet interesting **game logic** and 100% control
13. Powerful hero and its **simultaneous** health and game-score update and display
14. Displaying **High Score** on the created window

15. A last page is designed to get Usernames(i.e. who is playing) by clicking the “Save Score” option





16. Online presence of code with powerful **GitHub Repository** with informative Readme.md. Clean GitHub **Workflow** with best practices of git.

Project Modules

1. main.cpp

Calling a struct Game variable and using its functions init(), running(), handleEvents(), update(), and render() in the main() function for initializing, running loop, handling the events, updating the game and game objects and

rendering the objects respectively. Moreover, we are quitting the game by calling the destroy_window() function from the ‘Game’ structure.

2. bullet.h

A structure for bullets of the player with two functions “Bulletfire” and “update”. Bullet first time rendered in Bulletfire function. In the update function, bullet position changes, and all the enemies, tanks and innocent is checked if hit by the bullet or not. After that update of player health, score, enemy health executed accordingly.

3. ebullet.h

A structure for bullets of the enemy (enemy & tank) with two functions “Bulletfire” and “update”. Bullet first time rendered in Bulletfire function. In the update function, bullet position changes, and the player is checked if hit by the bulet or not. After that update of player health, the score executed accordingly.

4. clean.h

Destroying window and renderer by using the function Clean() and quitting all the subsystems such as video, audio, image, font.

5. constants.h

All the constants and SDL-related directory is linked there, and building c preprocessors. For this reason, these include does not need to do in the rest of the files, we used constants.h .

6. game.h

This file contains game structure with four major functions, initialization, handle events, update, render where many header files are included and many functions are called. checker function if game is running is to easily access the game state.

7. gameplay.h

Here “gameplay.h” is the main header file with multiple functions, that truly makes the gameplay. All the necessary files are included here and called here.

Player, enemy, tank, innocent is called here according to game logic. Level-wise distribution of work will be found here.

8. player.h

Player structure denoting the main character of the game, render, draw and update the player. There is also shield maintenance. If the shield is on it is displayed. Player movement can be done along all the direction. Flexible player position input and render made the game more interesting. There is also a function to return the position of the Player.

9. enemy.h

A structure is made for rendering and loading the enemy as well as keep track of its position and velocity.

10. tank.h

In this section, the tank is being rendered and loaded. Moreover, Kept the tracking of the tank's current position along with its velocity.

11. innocent.h

These three structures are the same type of function with slight differences, quite similar to player.h, they have their own function to init, render, handle events and update themselves. it can return its position and update its health accordingly. This function enables us to access its private variable can be accessed from any file easily

12. texturemanager.h

Loading textures and rendering them in this section made easy by this header file. One line code now can do a lot of works.

13. sound.h

Some sound-related variables and some built-in functions related to the sound/audio system have been used to control the sound system.

14. mainmenu.h

There are Continue, New Game, Highest Score, Settings and Story options in the main menu which are rendered and updated there.

15. last page.h

The header is used to render and update the Replay, Next Level, Save Score and Exit buttons in the last page.

16. highscore.h

The code for showing the top 5 scores has been implemented in this section by calling high() function.

17. pause.h

Rendering and updating the Continue, Exit and Main Menu options.

18. settings.h

Controlling the sound system by using logic and can go back to the main menu.

19. instruction.h

This header has been implemented to show the instructions and congratulations messages on the created window.

20. font.h

A FontManager structure is created to load the font and display them on the created window by using two different functions.

21. initialization.h

Initialized all the subsystems used in the whole game(i.e. audio, image, font etc).

22. install.sh

Simple bash scripting file to install of the game along with the dependencies . SDL2 Library and Github will be installed if not in the Linux device and Will Clone the remote github repository, then build the project making an executable “WOI-1971” file.

23. update.sh

Simple bash scripting file to install the latest update of the game project. The github repository is the main source now. the file will discard all the changes user made to the code and will pullall the latest changes developer made in the main github repository.

Team Member Responsibilities

Abdullah Ibne Hanif Arean, FH-12(2019-917-795), Team Leader

- Graphics Designing (Inkscape, Photoshop, Blender)
- Version Control (Git/GitHub)
- Logic design and implementation
- Structured Game Source Code Writing in C
- Page Showing Code, Dynamic Menu, Character Coding
- Game Deployment

Mehadi Hasan, SH-60(2019-517-843), Team Member

- Game Logic Design
- Structured Game Source Code Writing in C
- Font, File Handling, Sound, Instruction Coding
- Code Testing and Bug Fixing

Platform, Library & Tools

- C/C++ - Basic coding of the game done in c!
- VS Code - Free. Built on open-source yet powerful IDE!
- SDL2 - cross-platform development library designed to provide low-level access!
- Inkscape - professional quality vector graphics software and open source!
- Git/GitHub - software development and version control using Git!
- Bash- Unix shell and command language
- Markdown - a lightweight markup language for creating formatted text!

Limitations

- Movement of the characters do not have attractive animation, graphics are yet to be developed
- Written in C, in a structured manner, no object-oriented code written or used.
- As it is written in C only, no upgraded feature of c++ is available
- Replay of the game still to develop

Conclusions

Apart from learning new languages and technology, this game project taught us collaboration, pressure handling, peer communication and many other important qualities a software engineer should have. We hope we can implement these learnings in our life in the future.

Future Plan

The War of Independence is a great event in the national history of Bangladesh. There are a few games developed on this event though 50 years have passed since our independence. It is high time we made a stunning game on this event and made the world aware of the heroism of the Bangladeshi people in 1971. The game will be developed in a 2D SDL2/C++ initially for the project, but we have

aspirations to make it across all platforms(Android, IOS, Windows) 3D game using the power of Unreal Engine, Unity, Autodesk, Blender etc. later on.

Repositories & YouTube Video :

GitHub Repository: https://github.com/AbdullahArean/WOI-1971

Youtube Video: https://youtu.be/1e38v-MTL8w

References

1. SDL Learning Sources:

   https://lazyfoo.net/tutorials/SDL/

   https://www.packtpub.com/product/sdl-game-development/9781849696821

   https://www.parallelrealities.co.uk/

2. Free Image Source:

   https://www.freepik.com/