Bangladesh University of Engineering & Technology

Department of Computer Science & Engineering

Space Invaders

CSE 102 Term Project

Submitted By:
A. H. M. FUAD
ID: 2205102

Supervisor:
FATIMA NAWMI
Adjunct Lecturer
CSE, BUET

March 6, 2024

Contents

Abstract

This report assembles the journey of my assigned term project 'Space Invaders'. Features and their implementation ideas, bugs and fixes, and key learnings will be discussed here.

1 Introduction

Space Invaders is a 1978 Japanese video game. It is a fixed shooter game. There are simply three characters: Shooter, Enemy, and Boss UFO. There is only one level with a fixed number of enemies. After a certain time, the Boss enemy comes and shoots bullets. Shooter has two lives. Enemies and Boss UFO shoot bullets to destroy the earth. Shooter has to safeguard himself and kill all the enemies. Thus, the shooter will gain points.

The task is to recreate it as closely as possible.

2 Features and Implementations

Game features are discussed here briefly with implementation ideas:

2.1 Main Menu

A main menu screen loads at the very beginning. It is an image of the size of the game screen. Dimension of the game screen is 1024×720. It has five sub-options.

- Start Game
- Shooter Avatar Choose
- Leaderboard
- Settings
- Exit

Implementation: This is an image of size 1024×720. When the user presses on a certain coordinate range, the state changes and the user enters the respective option.

2.2 Settings

Here, the user can choose if the sound will be on or off and reset high scores. After clicking on the Settings button a new window will appear. There are two different sub-features,

2.2.1 Sound

There are two types of sound available. One is constant sound for the main menu and game pause. Another is shooting and enemy destruction sound. Players can choose the sound will be on or off.

2.2.2 Reset Scores

Scores are initially saved in a .txt file. When a user clicks on this button all the saved scores are cleared.
Implementation: A simple line of code does the work:

```
fclose(fopen("path", "w"));
```

2.3 Leaderboard

When a user clicks on the leaderboard, scores are fetched from the text file, and the highest five numbers are shown on the screen. If not enough data then "N/A" is shown as a placeholder.

2.4 Avatar

Users can choose between two types of default avatars from the main menu.

Implementation: By clicking on the image of avatars, the shooter image avatar path changes. Both shooter image paths are stored in a 2D character array. If avatar 1 is selected, 1st index of the array is used as the path, and the other one is likewise.

2.5 Game Play

This is the most important part of this project. Features implemented in the gameplay are, Shooter position, Shooter life, Shooter's bullet firing and destruction of enemies, Enemy movement, Attack of the Enemies, Enemies descending after a certain time, Boss UFO appearance after a certain time, Boss UFO attack, Scoring, Game Pause, Game End.

2.5.1 Shooter

Movement: Shooter moves right or left by 10 pixels per move. Users can use the Right arrow and Left arrow keys to move the shooter.

Firing & Enemy Kill: Shooter will fire bullets (one at a time) after the user presses the SPACE button. There will be a firing sound in the background. Enemy kill-checking algorithm:

```
IF ENEMY_X >= BULLET_X && ENEMY_X + ENEMY_SIZE <= BULLET_X:
    IF BULLET_Y>= ENEMY_Y && BULLET_Y <= ENEMY_Y + ENEMY_HEIGHT:
        // killed
```

Life: Shooter has two lives. When an enemy bullet hits the shooter, one life is used. The game will end after both of the lives are used.

2.5.2 Enemy

Movement: There are 24 enemies, usually placed as a 4×6 grid. They will move horizontally together. When they touch a border they ascend a row down and move the other way.

Attack: After a random time interval enemy side shoots lightning. If this lightning hits the shooter, the shooter loses a life.

2.5.3 Boss UFO

Movement: This type of enemy has linear movement. It appears from the rightmost upper corner and moves horizontally leftwards.

Implementation: The Whole game screen is considered as a 12×12 grid. First, we need to assign a direction. Then each 10 milliseconds we iterate through every cell, and check if there is an enemy in this cell, if yes then we shift this enemy to the next cell according to direction. If the enemy is in a border cell then we need to shift it one row down and change the direction of movement.

Attack: If the shooter tries to come underneath and attack it, it will shoot firebombs which will destroy the shooter spaceship.

Health: This type of enemy has twice the health of usual enemies. Two bullets are needed to destroy one UFO.

2.5.4 Scoring

One bullet hit kills general enemies, and two hits kill Boss UFO. Each enemy kill is worth 10 points and each Boss kill is worth 50 points. The score is shown in the left bottom corner of the screen. It updates after every kill.

2.5.5 Pause & END

Pause: The game pauses when the user presses ‘P’ and resumes when the user presses ‘R’. But in the background, it’s a lot more. All the timers running for movement and checking are paused also. Shooter movement is also turned off when the game is paused.

END: The game ends in three conditions:

1. Both lives of the shooter are used
2. Any enemy touches the shooter
3. All of the enemies are killed

For 1 and 2, the ‘GAME OVER’ message appears on the screen after. For 3, the ‘Winner’ message appears on the screen.

When the game ends because of any three of these conditions, the final score is saved in a text file.

2.6 Exit

Pressing the 'ESC' or Exit button on the main menu kills the entire process and closes the game window.

3 Challenges and Fixes

- **Random Crash:** This was my biggest struggle in the whole process. The game used to crash in random places. I checked every algorithm involved, they were okay. Then I tried changing every photo I used. Even I reinstalled the compiler. It was better than before but not fixed yet. Finally, I went through IGraphics, especially iShowBmp() implementation. There were two implementations, one of which was commented out. I uncommented it and commented on the usual one. Finally, the crashing issue was solved.

- **Sound:** Usually using *Playsound()* leads to error. It needs to be compiled with -lwinmm argument in vscode and for code blocks winmm library needs to be linked from compiler linkers. Also, I was planning to play multiple sounds for the game background and bullet shooting. But later I came to know it is not possible to do so.

- **Debugging Coordinates:** In this project for option or screen switch, it was a necessity to know the mouse coordinates. Printing them one by one and checking them was an easy yet cumbersome task.

- **Code flow:** Code base is not that small so it was a little hard to recall and reorganize all the functions. And after a break, it was hard to regain the previous workflow. It was fun though!

4 Additional Works & Links

- Project Demonstration Video Link: Click Here

- A GitHub repository was maintained throughout the whole development process. HERE is the link to the repository.

- LaTeXwas used to write this report.