Spaceship Battle Game - Assembly Language Implementation

Muhammad Nouman Hafeez (21I-0416) + Muhammad Abdullah Omer (21I-2569)

August 9, 2025

Abstract

This report documents the design and implementation of a Spaceship Battle game developed entirely in x86 Assembly Language using the Irvine32 library. The game features multiple levels with increasing difficulty, enemy AI patterns, collision detection, scoring system, and persistent high score tracking. The report covers the game architecture, key algorithms, user interface components, and technical challenges overcome during development.

Contents

1 Introduction

The Spaceship Battle game is a classic arcade-style shooter implemented in low-level x86 Assembly language. Developed using Microsoft Visual Studio with the Irvine32 library, this game demonstrates advanced assembly programming techniques including:

- Complex game state management
- Real-time user input handling
- Collision detection algorithms
- File I/O for score persistence
- Sound effects integration
- Multi-level progressive difficulty

The game was developed by Muhammad Nouman Hafeez (21I-0416) and Muhammad Abdullah Omer (21I-2569) as an advanced programming project showcasing mastery of assembly language concepts.

2 Game Architecture

The game follows a structured procedural architecture with clear separation of concerns:

2.1 Main Game Loop

The core game loop handles:

- Input processing
- Game state updates
- Rendering
- Timing control

```
GameMainLoop PROC
MainLoop:
    call CheckPortal
    call CheckLives
    call UpdateIteration
    call ProcessInput
    call UpdateGame
    call DisplayInfo
    call CheckBorderRedraw
    call GameDelay
    jmp MainLoop
EndLoop:
    ret
GameMainLoop ENDP
```

Listing 1: Main Game Loop Structure

2.2 Memory Organization

The game uses several data structures:

- **Bullet Struct**: Tracks player projectiles
- **Enemy Struct**: Manages enemy ships
- **EnemyBullet Struct**: Handles enemy projectiles
- **SuperFood Struct**: Manages power-ups

```
; Bullet data structure
MAX_BULLETS = 10
Bullet STRUCT
    xPos byte ?
    yPos byte ?
    active byte ?
Bullet ENDS

; Enemy bullet data structure
MAX_ENEMY_BULLETS = 100
EnemyBullet STRUCT
    xPos byte ?
    yPos byte ?
    active byte ?
EnemyBullet ENDS

; Super Food data structure
MAX_SUPER_FOODS = 20
SuperFood STRUCT
    xPos byte ?
    yPos byte ?
    active byte ?
    lifeCounter byte ?
SuperFood ENDS

; Enemy data structure
MAX_ENEMIES = 15
Enemy STRUCT
    xPos byte ?
    yPos byte ?
    active byte ?
    direction byte ?
    fireCounter byte ?
Enemy ENDS
```

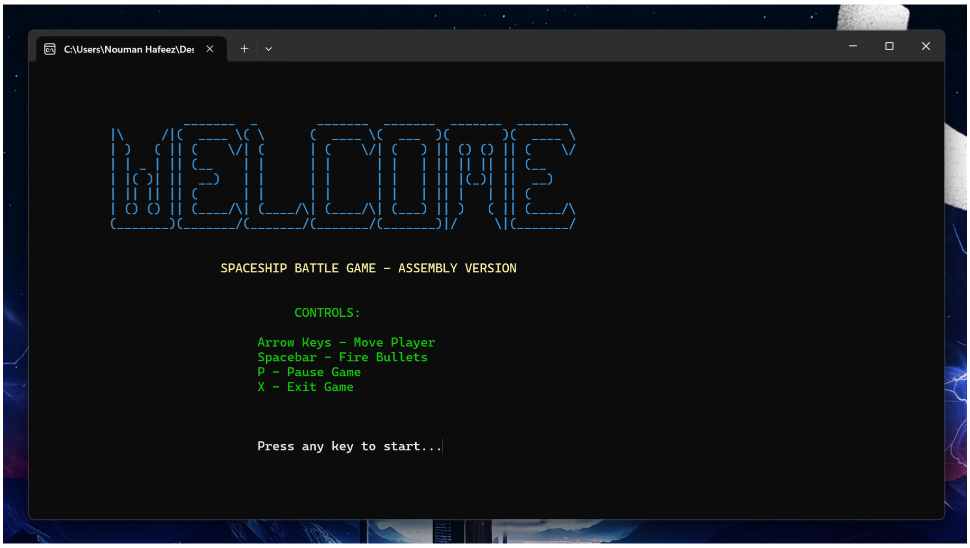
Listing 2: Game Data Structures

3 User Interface

The game features a comprehensive UI system with multiple screens:

3.1 Welcome Screen

The welcoming interface introduces players to the game (Figure 1).



```
SPACESHIP BATTLE GAME - ASSEMBLY VERSION

CONTROLS:

Arrow Keys - Move Player
Spacebar - Fire Bullets
P - Pause Game
X - Exit Game


Press any key to start...
```

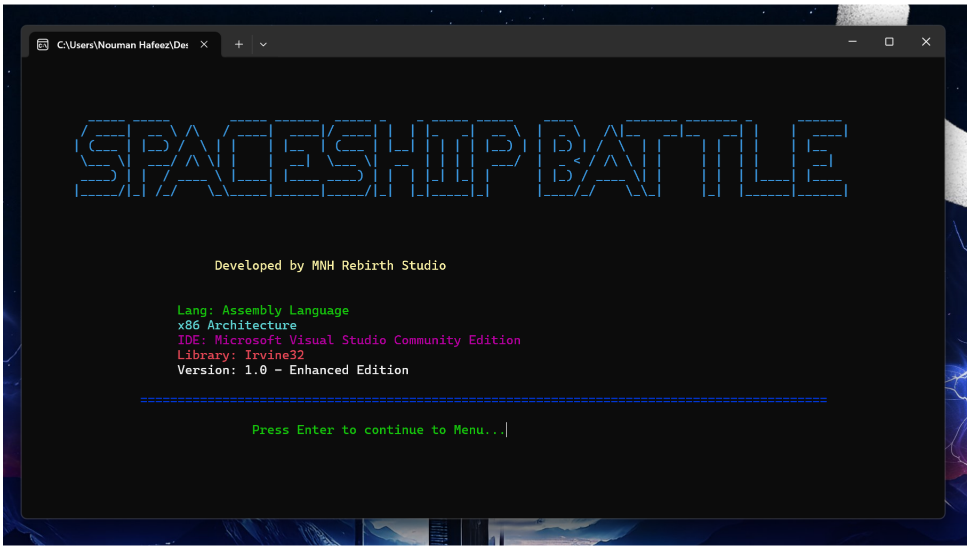
Figure 1: Welcome Screen

Key features:

- ASCII art title
- Game instructions
- Color-coded controls display

3.2 Spaceship Battle Screen

Technical information screen (Figure 2).



```
  _____ _____        _____ ______  _____ _    _ _____ _____    ____       _______ _______ _      ______ 
 / ____|  __ \ /\   / ____|  ____|/ ____| |  | |_   _|  __ \  |  _ \   /\|__   __|__   __| |    |  ____|
| (___ | |__) /  \ | |    | |__  | (___ | |__| | | | | |__) | | |_) | /  \  | |     | |  | |    | |__   
 \___ \|  ___/ /\ \| |    |  __|  \___ \|  __  | | | |  ___/  |  _ < / /\ \ | |     | |  | |    |  __|  
 ____) | |  / ____ \ |____| |____ ____) | |  | |_| |_| |      | |_) / ____ \| |     | |  | |____| |____ 
|_____/|_| /_/    \_\_____|______|_____/|_|  |_|_____|_|      |____/_/    \_\_|     |_|  |______|______|


                 Developed by MNH Rebirth Studio


              Lang: Assembly Language
              x86 Architecture
              IDE: Microsoft Visual Studio Community Edition
              Library: Irvine32
              Version: 1.0 - Enhanced Edition

         ============================================================================================

                        Press Enter to continue to Menu...
```

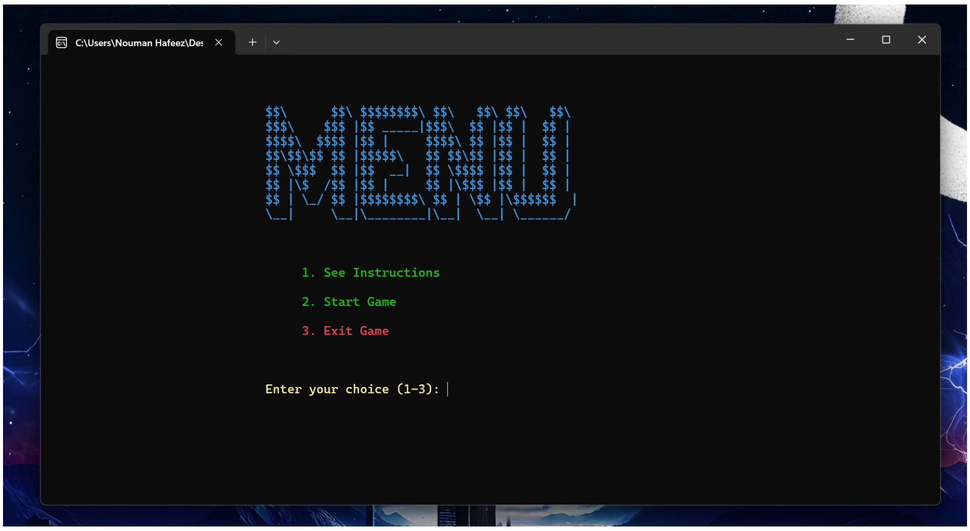
Figure 2: Spaceship Battle Information Screen

Displays:

- Development studio information
- Technical specifications
- Assembly language details

3.3 Menu Screen

Main navigation interface (Figure 3).



```
$$\      $$\ $$$$$$$$\ $$\   $$\ $$\   $$\ 
$$$\    $$$ |$$  _____|$$$\  $$ |$$ |  $$ |
$$$$\  $$$$ |$$ |      $$$$\ $$ |$$ |  $$ |
$$\$$\$$ $$ |$$$$$\    $$ $$\$$ |$$ |  $$ |
$$ \$$$  $$ |$$  __|   $$ \$$$$ |$$ |  $$ |
$$ |\$  /$$ |$$ |      $$ |\$$$ |$$ |  $$ |
$$ | \_/ $$ |$$$$$$$$\ $$ | \$$ |\$$$$$$  |
\__|     \__|\________|\__|  \__| \______/ 



     1. See Instructions

     2. Start Game

     3. Exit Game



Enter your choice (1-3):
```

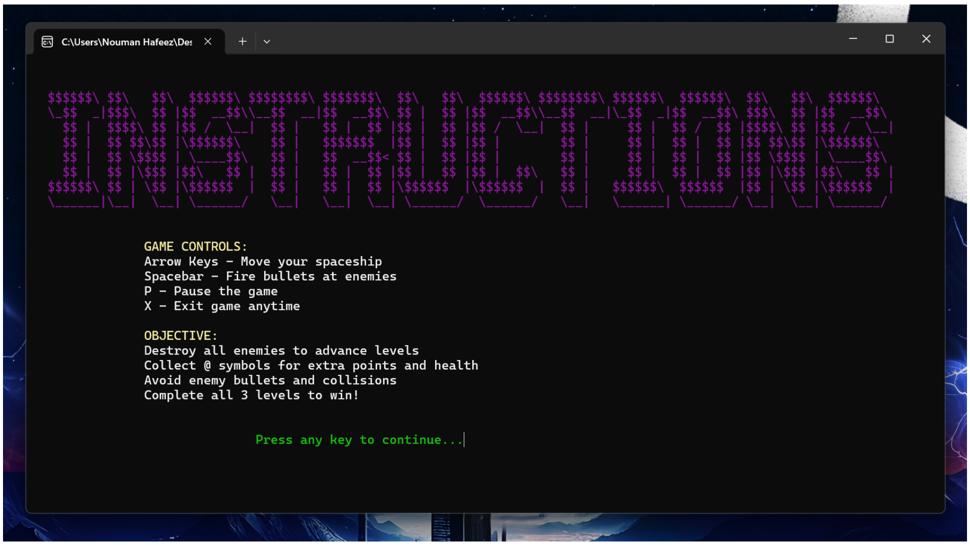
Figure 3: Game Menu Screen

Options:

- View instructions
- Start game
- Exit

3.4 Instructions Screen

Detailed game guidance (Figure 4).



```
$$$$$$\ $$\   $$\  $$$$$$\ $$$$$$$$\ $$$$$$$\  $$\   $$\  $$$$$$\ $$$$$$$$\ $$$$$$\  $$$$$$\  $$\   $$\  $$$$$$\
\_$$  _|$$$\  $$ |$$  __$$\\__$$  __|$$  __$$\ $$ |  $$ |$$  __$$\\__$$  __|\_$$  _|$$  __$$\ $$$\  $$ |$$  __$$\
  $$ |  $$$$\ $$ |$$ /  \__|  $$ |   $$ |  $$ |$$ |  $$ |$$ /  \__|  $$ |     $$ |  $$ /  $$ |$$$$\ $$ |$$ /  \__|
  $$ |  $$ $$\$$ |\$$$$$$\    $$ |   $$$$$$$  |$$ |  $$ |$$ |        $$ |     $$ |  $$ |  $$ |$$ $$\$$ |\$$$$$$\
  $$ |  $$ \$$$$ | \____$$\   $$ |   $$  __$$< $$ |  $$ |$$ |        $$ |     $$ |  $$ |  $$ |$$ \$$$$ | \____$$\
  $$ |  $$ |\$$$ |$$\   $$ |  $$ |   $$ |  $$ |$$ |  $$ |$$ |  $$\   $$ |     $$ |  $$ |  $$ |$$ |\$$$ |$$\   $$ |
$$$$$$\ $$ | \$$ |\$$$$$$  |  $$ |   $$ |  $$ |\$$$$$$  |\$$$$$$  |  $$ |   $$$$$$\  $$$$$$  |$$ | \$$ |\$$$$$$  |
\______|\__|  \__| \______/    \__|   \__|  \__| \______/  \______/    \__|   \______| \______/ \__|  \__| \______/


             GAME CONTROLS:
             Arrow Keys - Move your spaceship
             Spacebar - Fire bullets at enemies
             P - Pause the game
             X - Exit game anytime

             OBJECTIVE:
             Destroy all enemies to advance levels
             Collect @ symbols for extra points and health
             Avoid enemy bullets and collisions
             Complete all 3 levels to win!


                            Press any key to continue...
```

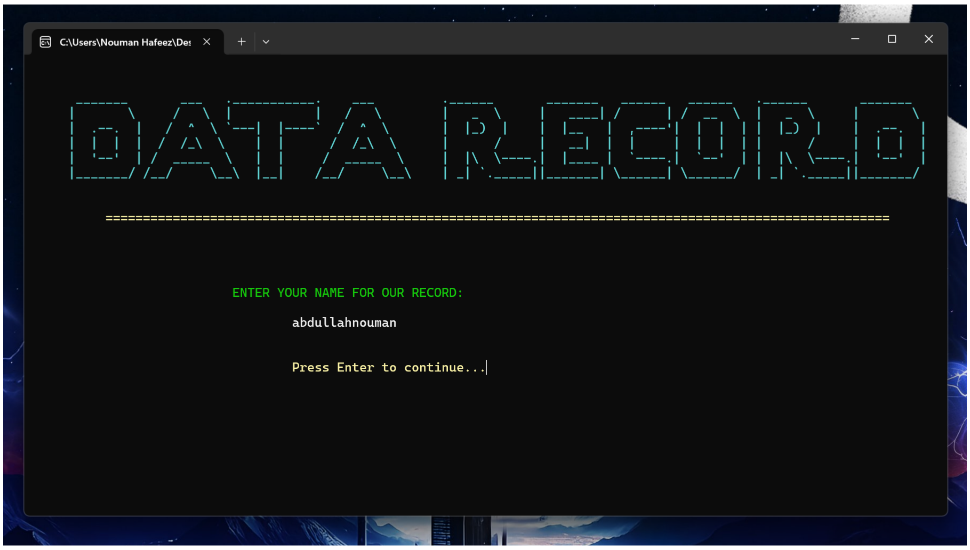
Figure 4: Game Instructions Screen

Covers:

- Control schemes
- Game objectives
- Power-up explanations

3.5 Name Input Screen

Player identification (Figure 5).



```
==========================================================================================

                    ENTER YOUR NAME FOR OUR RECORD:
                           abdullahnouman

                           Press Enter to continue...
```

Figure 5: Player Name Input Screen

Features:

- Data record header
- Name prompt
- Input validation

4 Gameplay

The core game features three progressive difficulty levels.

4.1 Level 1 Design

Concept sketch for Level 1 (Figure 6).

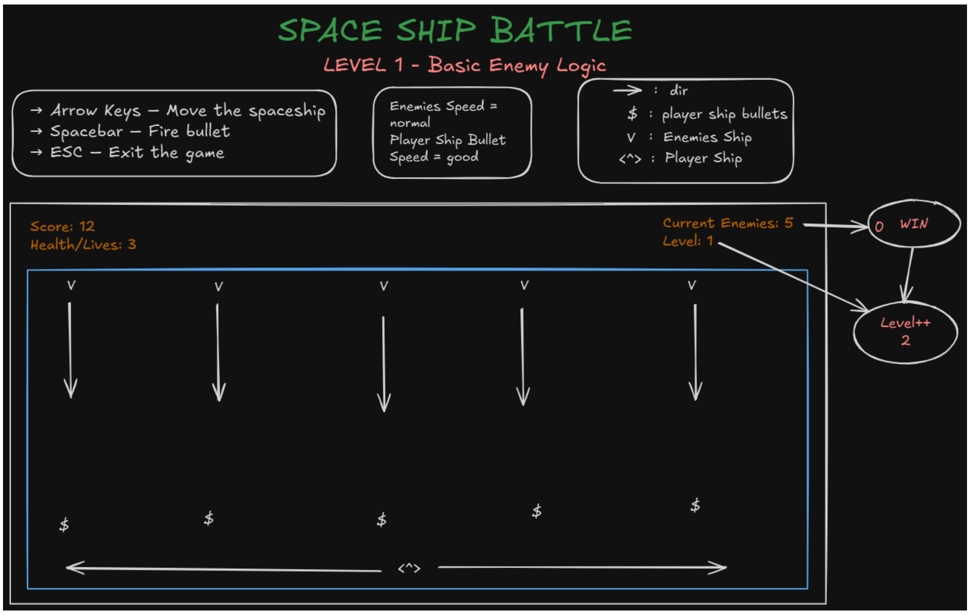


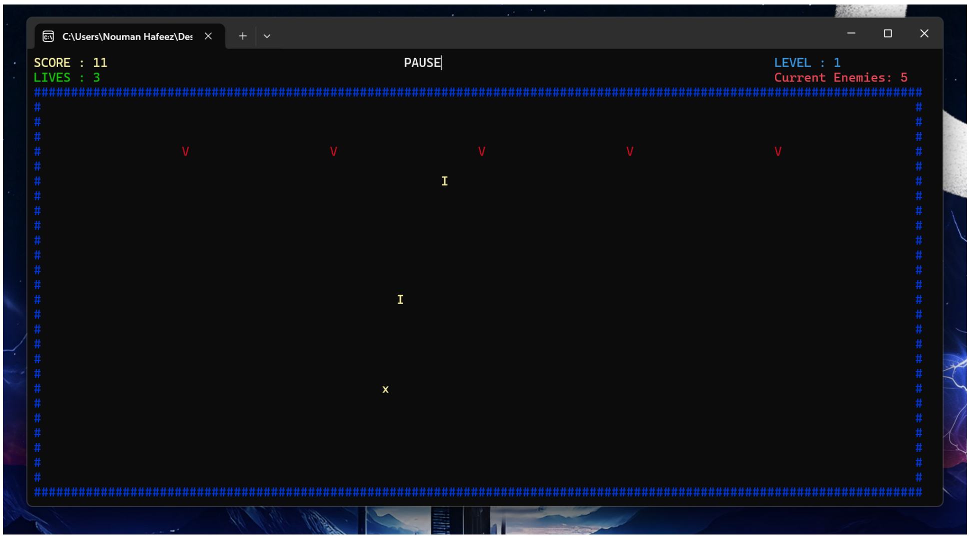
Figure 6: Level 1 Concept Sketch

Design elements:

- Basic enemy formations
- Simple movement patterns
- Clear player area
- Minimal obstacles

4.2 Level 1 Implementation

Basic enemy patterns (Figure 7).



```
SCORE : 11                                 PAUSE                                   LEVEL : 1
LIVES : 3                                                                          Current Enemies: 5
```

Figure 7: Level 1 Gameplay

Characteristics:

- 5 enemy ships
- Simple downward movement
- Slow firing rate
- Basic collision detection

4.3 Level 2 Design

Concept sketch for Level 2 (Figure 8).

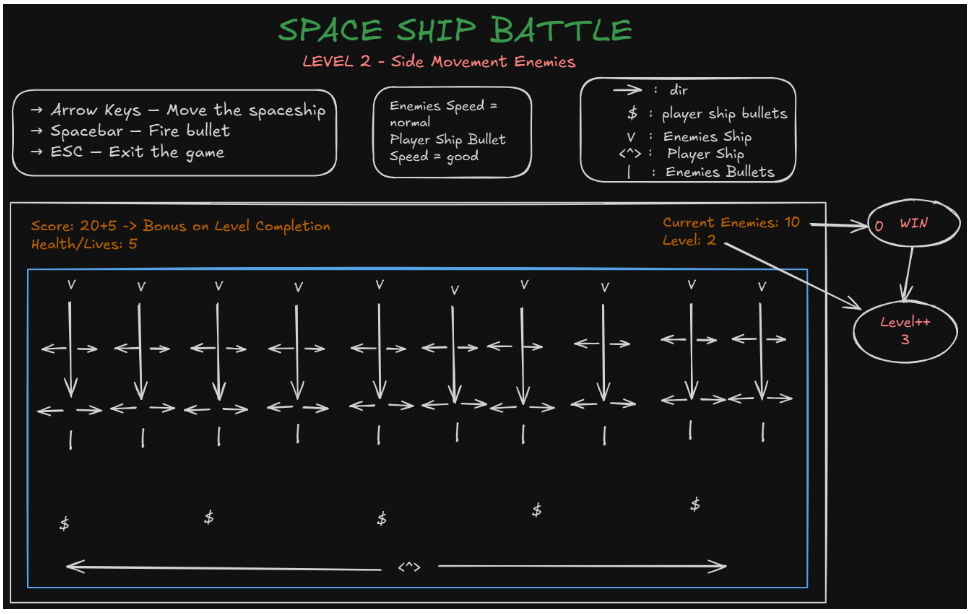


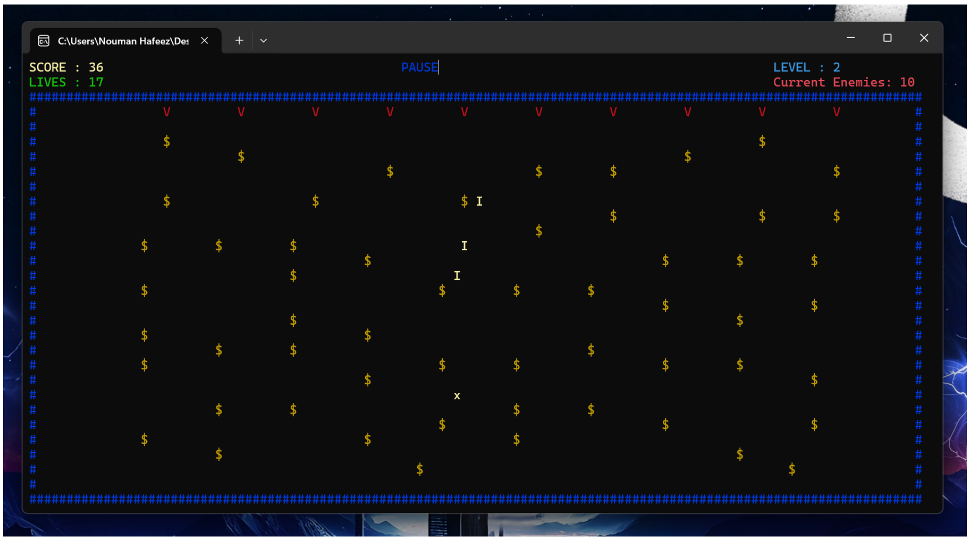
Figure 8: Level 2 Concept Sketch

Design elements:

- More complex enemy arrangements
- Horizontal movement patterns
- Increased enemy count
- Intermediate difficulty indicators

4.4 Level 2 Implementation

Enhanced difficulty (Figure 9).



```
SCORE : 36                                  PAUSE                                  LEVEL : 2
LIVES : 17                                                                         Current Enemies: 10
```

Figure 9: Level 2 Gameplay

Enhancements:

- 10 enemy ships
- Complex movement patterns

- Faster projectiles
- Enemy respawning

4.5 Level 3 Design

Concept sketch for Level 3 (Figure 10).

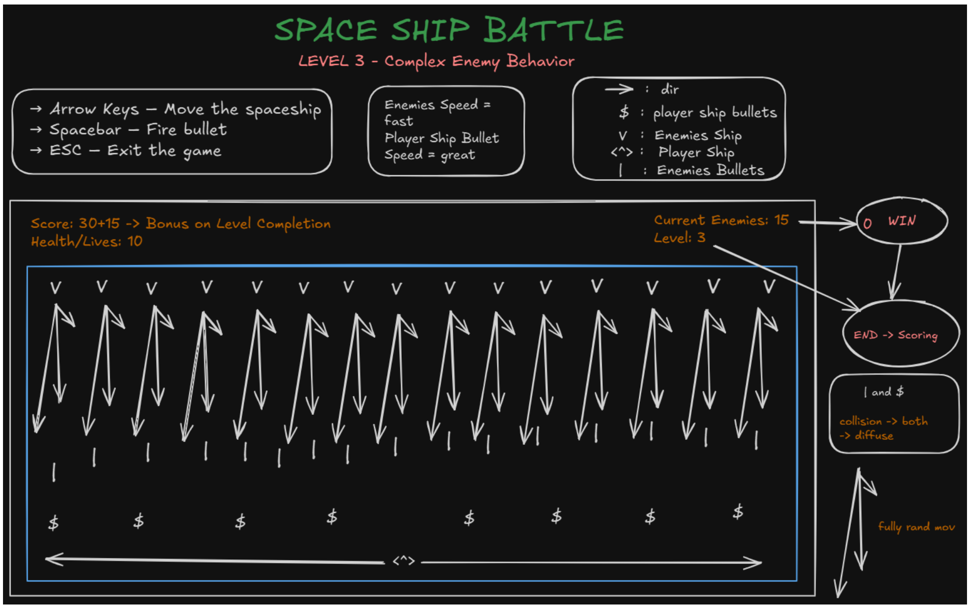


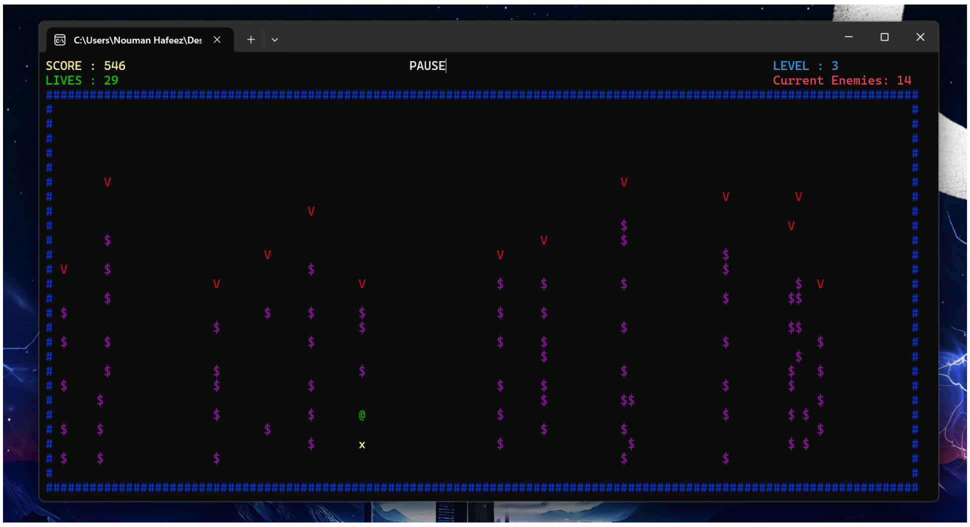
Figure 10: Level 3 Concept Sketch

Design elements:

- Chaotic enemy placement
- Random movement indicators
- Power-up locations
- Advanced gameplay elements

4.6 Level 3 Implementation

Advanced challenge (Figure 11).



```
SCORE : 546                                   PAUSE                                  LEVEL : 3
LIVES : 29                                                                           Current Enemies: 14
```

Figure 11: Level 3 Gameplay

Features:

- 15 enemy ships
- Random movement algorithms
- Rapid firing
- Bullet diffusion mechanics
- Super food power-ups

5 Game Systems

5.1 Collision Detection

The game implements several collision detection algorithms:

```
CheckCollisions PROC
    call CheckBulletEnemyHit
    call CheckPlayerEnemyHit
    call CheckEnemyBulletPlayerHit
    call CheckPlayerSuperFoodHit
    call CheckPlayerBulletEnemyBulletCollision
    ret
CheckCollisions ENDP

CheckBulletEnemyHit PROC
    mov edi, OFFSET bullets
    mov ecx, MAX_BULLETS
    ; ... collision detection logic ...
    ret
CheckBulletEnemyHit ENDP
```

Listing 3: Collision Detection

5.2 Scoring System

The scoring system tracks:

- Enemy defeats
- Power-up collection
- Level completion bonuses

```
DisplayInfo PROC
    ; Display SCORE
    mov eax, yellow + (black SHL 4)
    call SetTextColor
    mov dl, 0
    mov dh, 0
    call Gotoxy
    mov edx, OFFSET strScore
    call WriteString

    ; Display current score value
    mov dl, 8
    mov dh, 0
    call Gotoxy
    movzx eax, score
    call writeDEC
    ; ... additional display code ...
    ret
DisplayInfo ENDP
```

Listing 4: Score Display

5.3 File I/O System

The game implements a robust file system for high score persistence:

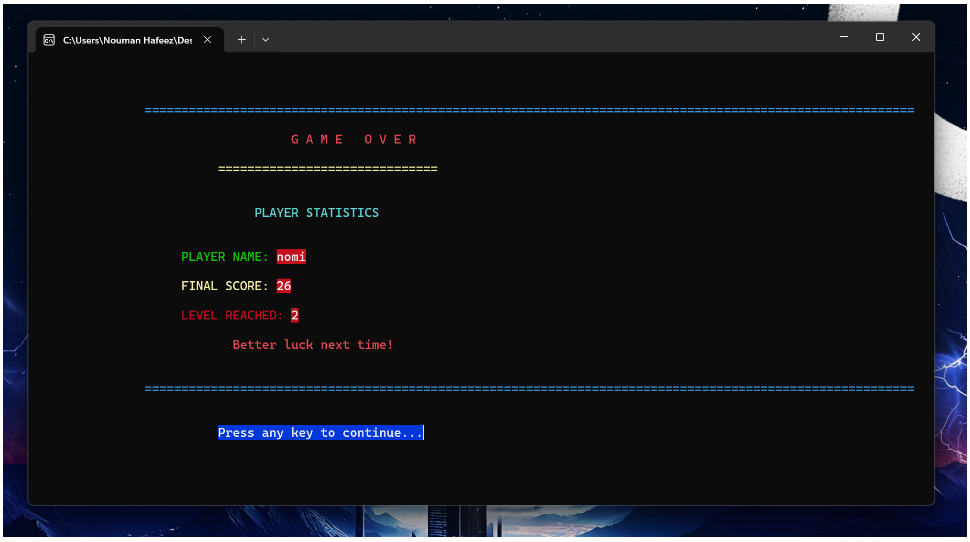
```
SaveRecord PROC
    ; File handling setup
    mov edx, OFFSET filename
    call OpenInputFile

    ; Write player data
    mov eax, RecordFileHandle
    mov edx, OFFSET Write1  ; "Name : "
    mov ecx, 7
    call WriteToFile

    ; Write score
    mov eax, RecordFileHandle
    mov edx, OFFSET strResult
    call WriteToFile

    ; Close file
    mov eax, RecordFileHandle
    call CloseFile
    ret
SaveRecord ENDP
```

Listing 5: Score Saving

6 Game Completion

6.1 Player Statistics

Post-game performance review (Figure 12).



```
        ==========================================================================================================
                          G A M E   O V E R
                  ==============================

                       PLAYER STATISTICS

             PLAYER NAME: nomi
             FINAL SCORE: 26
             LEVEL REACHED: 2
                    Better luck next time!

        ==========================================================================================================

                  Press any key to continue...
```

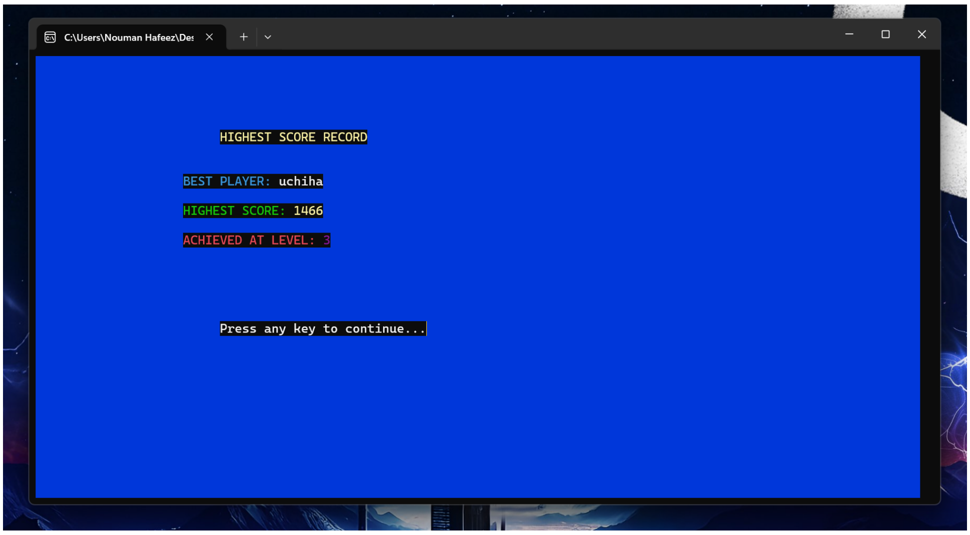
Figure 12: Player Statistics Screen

Displays:

- Final score
- Level reached
- Performance evaluation

6.2 Highest Score Screen

Leaderboard display (Figure 13).



```
                  HIGHEST SCORE RECORD

             BEST PLAYER: uchiha
             HIGHEST SCORE: 1466
             ACHIEVED AT LEVEL: 3



                  Press any key to continue...
```

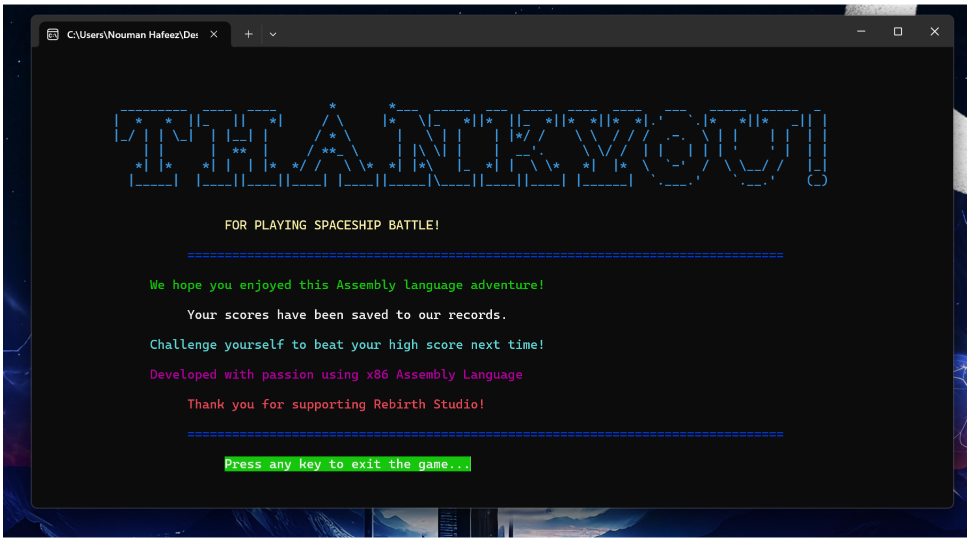
Figure 13: Highest Score Screen

Features:

- Top player name
- Record score
- Achievement level

6.3 Thank You Screen

Closing appreciation screen (Figure 14).



```
        FOR PLAYING SPACESHIP BATTLE!

    ================================================================================

  We hope you enjoyed this Assembly language adventure!

      Your scores have been saved to our records.

  Challenge yourself to beat your high score next time!

  Developed with passion using x86 Assembly Language

      Thank you for supporting Rebirth Studio!

    ================================================================================

        Press any key to exit the game...
```

Figure 14: Thank You Screen

Includes:

- Developer credits
- Appreciation message
- Final remarks

7 Technical Challenges

Key technical challenges overcome:

7.1 Real-time Input Handling

The game implements responsive keyboard input:

```
ProcessInput PROC
    call ReadKey
    jz PI_End

    cmp al, 0
    jne PI_Standard
    call HandleArrows
    jmp PI_End

PI_Standard:
    mov inputChar, al
    cmp inputChar, "x"
    je PI_Exit
    cmp inputChar, "p"
    je PI_Pause
    ; ... other key checks ...
ProcessInput ENDP
```

Listing 6: Input Handling

7.2 Enemy AI Patterns

Advanced enemy behaviors:

```
UpdateLevel3EnemiesRandomMovement PROC
    inc level3RandomCounter
    mov al, level3RandomCounter
    cmp al, level3RandomDelay
    jb UL3RM_End

    mov level3RandomCounter, 0
    call MoveLevel3EnemiesRandomPattern

UL3RM_End:
    ret
UpdateLevel3EnemiesRandomMovement ENDP
```

Listing 7: Enemy AI

7.3 Sound Effects

Sound integration through WinMM:

```
INCLUDE Irvine32.inc
includelib winmm.lib

PlaySoundA PROTO,
    pszSound:PTR BYTE,
    hmod:DWORD,
    fdwSound:DWORD

; Sound files
goBOOM byte "main-player-sound.wav",0
goDISHOOM byte "goDISHOOM.wav",0

; Play sound example
INVOKE PlaySoundA, OFFSET goBOOM, NULL, 20001H
```

Listing 8: Sound System

8 Conclusion

The Spaceship Battle game demonstrates advanced x86 Assembly programming techniques in a complete game implementation. Key achievements include:

- Complex game mechanics in low-level code
- Efficient memory management
- Robust file I/O operations
- Engaging user interface
- Progressive difficulty system

The project showcases how assembly language can be used for complex application development beyond simple demonstrations.

A Complete Source Code

The complete game source code is available in the accompanying ASM file.