YUSUF ÖMER TURSUN

29/03/2024

SPRING 2024

CEN326

Project 1

TITLE: SPACE SHOOTER GAME

INTRODUCTION

The game I designed and coded for has the following features.

1 - In the Console mode and MkeyKernel Mode section, there are multiple enemies that come to my game from random parts of the screen. And the spaceship has multiple fire features to hit these enemies. To give the impression of space to my game, I visualized my game with a star effect.

2 - In the game I designed in BaseKernel, OpenGl and WINBGIM environments, the multi-enemy and multiple bullet features are supported in the same way,

while on the other hand, I increased the movement speed of the enemies by 1 for every 5 targets hit.

3 - In all 5 environments, there is an End of Game screen and score information written on this screen.

BASIC REQUIREMENTS

1 - On mkeykernel OS (Linux Kernel in console) https://github.com/arjun024/mkeykernel

- For the Mkeykernel environment, I first had to set up a virtual machine and install Ubuntu on this virtual machine. Then, I uploaded the mkeykernel kernel codes from github and wrote my own source codes into the kernel.c file. Finally, I completed the mekykernel environment by writing the appropriate compilation and run commands on the terminal screen.

2 - On basekernel OS (Linux Kernel in graphics mode) https://github.com/dthain/basekernel

-For the basekernel environment, I used the Ubuntu installed on my virtual machine again. I integrated my own source codes into the main.c file by uploading the basekernel kernel codes on Github. Then, I completed the basekernel environment by typing the appropriate compilation and run commands on the terminal screen.

3 - On Windows at Code Blocks in console mode(with C++)

-I installed CodeBlocks for console mode and then selected the appropriate c / c++ compiler. Then, I created a project and wrote my source codes in my project. Finally, I compiled and ran my source codes in CodeBlocks.

4 - On Windows at Code Blocks in graphics mode/WinBGI (with C++)

-For WinBGI, I had to use CodeBlocks again. But in addition, I had to install the graphics.h file in CodeBlocks. For this, I had to edit a few places in the Compiler settings of CodeBlocks. I added the graphics.h and winbgi.h files to the include folder of my project. Then, I compiled and ran my source codes in CodeBlocks.

5 - On OpenGL environment in graphics mode (with C++)

-I used Visiual Studio for the OpenGl environment and here I had to download freeglut from the Nuget package manager. After providing the appropriate installations, I compiled and ran my codes in Visiual Studio.

PSEUDOCODE

```
BEGIN
    BOARD_WIDTH <- 50
    BOARD_HEIGHT <- 20
    PLAYER_SYMBOL <- '>'
    ENEMY_SYMBOL <- '*'
    BULLET_SYMBOL <- '|'
    MAX_ENEMIES <- 5
    MAX_BULLETS <- 10
    playerX, playerY <- 0
    enemyX[MAX_ENEMIES], enemyY[MAX_ENEMIES]
    bulletX[MAX_BULLETS], bulletY[MAX_BULLETS]
    numEnemies <- 0
    numBullets <- 0

START
    setupGameBoard()

    FOREVER
        drawGameBoard()
        updateInput()
        updateGameState()
        wait(100)
```

```
setupGameBoard()
    playerX <- BOARD_WIDTH / 2
    playerY <- BOARD_HEIGHT - 1
    numEnemies <- 0
    numBullets <- 0

drawGameBoard()
    clearGameBoard()
    FOR EACH Y ROW
        FOR EACH X COLUMN
            IF X = playerX AND Y = playerY THEN
                print PLAYER_SYMBOL
            ELSE
                isEnemy <- false
                FOR EACH ENEMY
                    IF X = enemyX[i] AND Y = enemyY[i] THEN
                        print ENEMY_SYMBOL
                        isEnemy <- true
                        BREAK
                IF NOT isEnemy THEN
                    isBullet <- false
                    FOR EACH BULLET
                        IF X = bulletX[i] AND Y = bulletY[i] THEN
                            print BULLET_SYMBOL
                            isBullet <- true
                            BREAK
                    IF NOT isBullet THEN
                        print " "
        MOVE TO NEXT LINE
```

```
clearGameBoard()
    CLEAR SCREEN

updateInput()
    IF KEY PRESSED THEN
        key <- GET KEY
        IF key = 'a' AND playerX > 0 THEN
            playerX--
        ELSE IF key = 'd' AND playerX < BOARD_WIDTH - 1 THEN
            playerX++
        ELSE IF key = ' ' THEN
            fireBullet()

updateGameState()
    moveEnemies()
    moveBullets()
    checkCollisions()

moveEnemies()
    FOR EACH ENEMY
        enemyY[i]++
        IF enemyY[i] >= BOARD_HEIGHT THEN
            enemyX[i] <- RANDOM() % BOARD_WIDTH
            enemyY[i] <- 0

moveBullets()
    FOR EACH BULLET
        IF bulletY[i] >= 0 THEN
            bulletY[i]--
```

```
fireBullet()
    IF numBullets < MAX_BULLETS THEN
        bulletX[numBullets] <- playerX
        bulletY[numBullets] <- playerY - 1
        numBullets++

checkCollisions()
    FOR EACH BULLET
        FOR EACH ENEMY
            IF bulletX[i] = enemyX[j] AND bulletY[i] = enemyY[j] THEN
                removeEnemy(j)
                addNewEnemy()
                removeBullet(i)

removeEnemy(index)
    FOR k FROM index TO numEnemies - 1
        enemyX[k] <- enemyX[k + 1]
        enemyY[k] <- enemyY[k + 1]
    numEnemies--

addNewEnemy()
    IF numEnemies < MAX_ENEMIES THEN
        enemyX[numEnemies] <- RANDOM() % BOARD_WIDTH
        enemyY[numEnemies] <- 0
        numEnemies++

removeBullet(index)
    FOR k FROM index TO numBullets - 1
        bulletX[k] <- bulletX[k + 1]
        bulletY[k] <- bulletY[k + 1]
    numBullets--
```

```
main()
    setup()
    WHILE TRUE
        drawBoard()
        updateInput()
        updateGame()
        sleep(100) // Adjust game speed
```

PSEUDOCODE TEXT VERSION

```
BEGIN
    BOARD_WIDTH <- 50
    BOARD_HEIGHT <- 20
    PLAYER_SYMBOL <- '>'
    ENEMY_SYMBOL <- '*'
    BULLET_SYMBOL <- '|'
    MAX_ENEMIES <- 5
    MAX_BULLETS <- 10
    playerX, playerY <- 0
    enemyX[MAX_ENEMIES], enemyY[MAX_ENEMIES]
    bulletX[MAX_BULLETS], bulletY[MAX_BULLETS]
    numEnemies <- 0
    numBullets <- 0

START
    setupGameBoard()

    FOREVER
        drawGameBoard()
        updateInput()
        updateGameState()
        wait(100)

setupGameBoard()
    playerX <- BOARD_WIDTH / 2
```

```
    playerY <- BOARD_HEIGHT - 1
    numEnemies <- 0
    numBullets <- 0

drawGameBoard()
    clearGameBoard()
    FOR EACH Y ROW
        FOR EACH X COLUMN
            IF X = playerX AND Y = playerY THEN
                print PLAYER_SYMBOL
            ELSE
                isEnemy <- false
                FOR EACH ENEMY
                    IF X = enemyX[i] AND Y = enemyY[i] THEN
                        print ENEMY_SYMBOL
                        isEnemy <- true
                        BREAK
                IF NOT isEnemy THEN
                    isBullet  <- false
                    FOR EACH BULLET
                        IF X = bulletX[i] AND Y = bulletY[i] THEN
                            print BULLET_SYMBOL
                            isBullet <- true
                            BREAK
                    IF NOT isBullet THEN
                        print " "
        MOVE TO NEXT LINE
```

```
clearGameBoard()
    CLEAR SCREEN

updateInput()
    IF KEY PRESSED THEN
        key <- GET KEY
        IF key = 'a' AND playerX > 0 THEN
            playerX--
        ELSE IF key = 'd' AND playerX < BOARD_WIDTH - 1 THEN
            playerX++
        ELSE IF key = ' ' THEN
            fireBullet()

updateGameState()
    moveEnemies()
    moveBullets()
    checkCollisions()

moveEnemies()
    FOR EACH ENEMY
        enemyY[i]++
        IF enemyY[i] >= BOARD_HEIGHT THEN
            enemyX[i] <- RANDOM() % BOARD_WIDTH
            enemyY[i] <- 0

moveBullets()
```

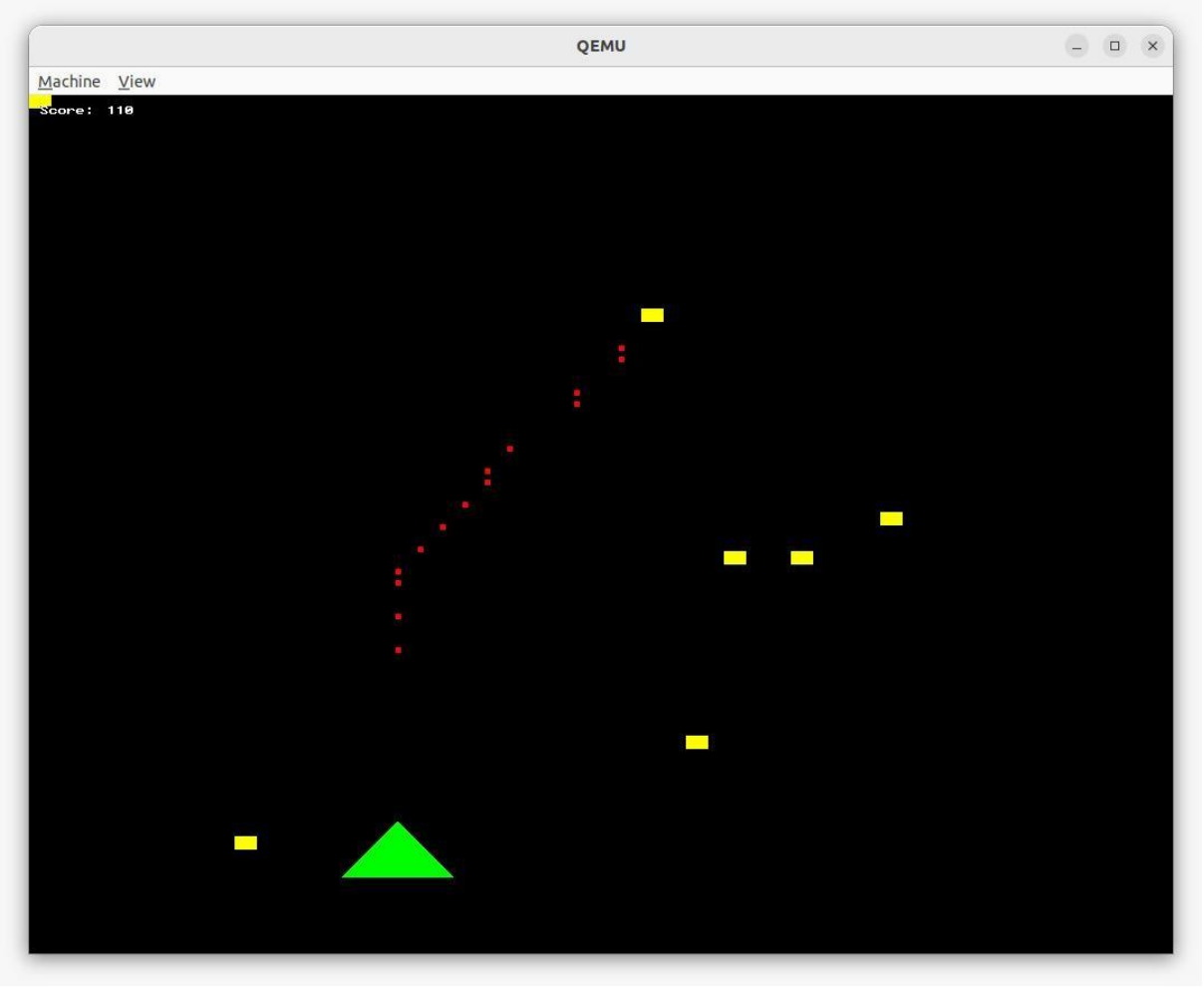
```
    FOR EACH BULLET
        IF bulletY[i] >= 0 THEN
            bulletY[i]--

fireBullet()
    IF numBullets < MAX_BULLETS THEN
        bulletX[numBullets] <- playerX
        bulletY[numBullets] <- playerY - 1
        numBullets++

checkCollisions()
    FOR EACH BULLET
        FOR EACH ENEMY
            IF bulletX[i] = enemyX[j] AND bulletY[i] = enemyY[j] THEN
                removeEnemy(j)
                addNewEnemy()
                removeBullet(i)

removeEnemy(index)
    FOR k FROM index TO numEnemies - 1
        enemyX[k] <- enemyX[k + 1]
        enemyY[k] <- enemyY[k + 1]
    numEnemies--

addNewEnemy()
    IF numEnemies < MAX_ENEMIES THEN
        enemyX[numEnemies] <- RANDOM() % BOARD_WIDTH
```

```
        enemyY[numEnemies] <- 0
        numEnemies++

removeBullet(index)
    FOR k FROM index TO numBullets - 1
        bulletX[k] <- bulletX[k + 1]
        bulletY[k] <- bulletY[k + 1]
    numBullets—

main()
    setup()
    WHILE TRUE
        drawBoard()
        updateInput()
        updateGame()
        sleep(100) // Adjust game speed
```

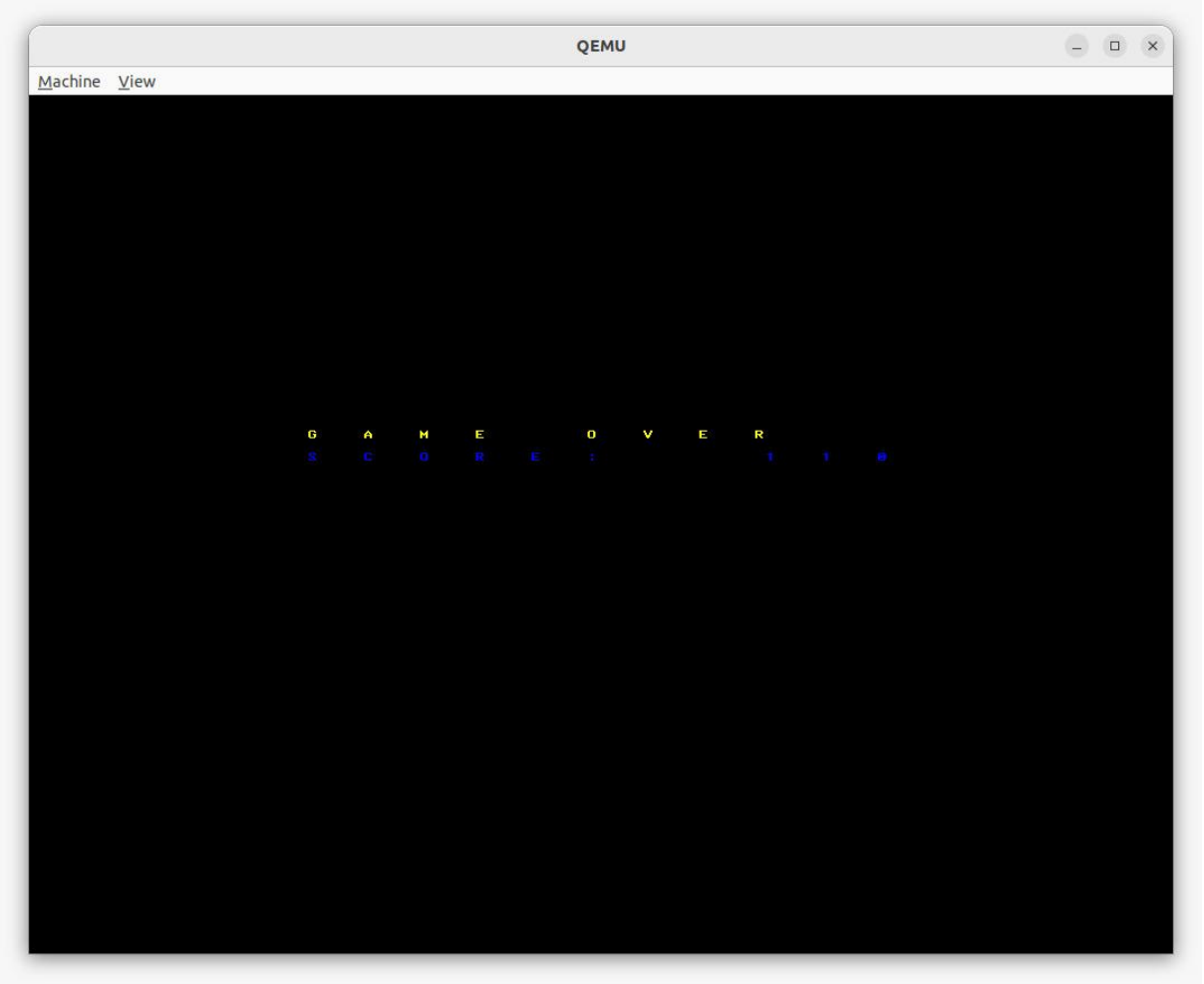
DEMONSTRATION

1 – BASE KERNEL

1.1

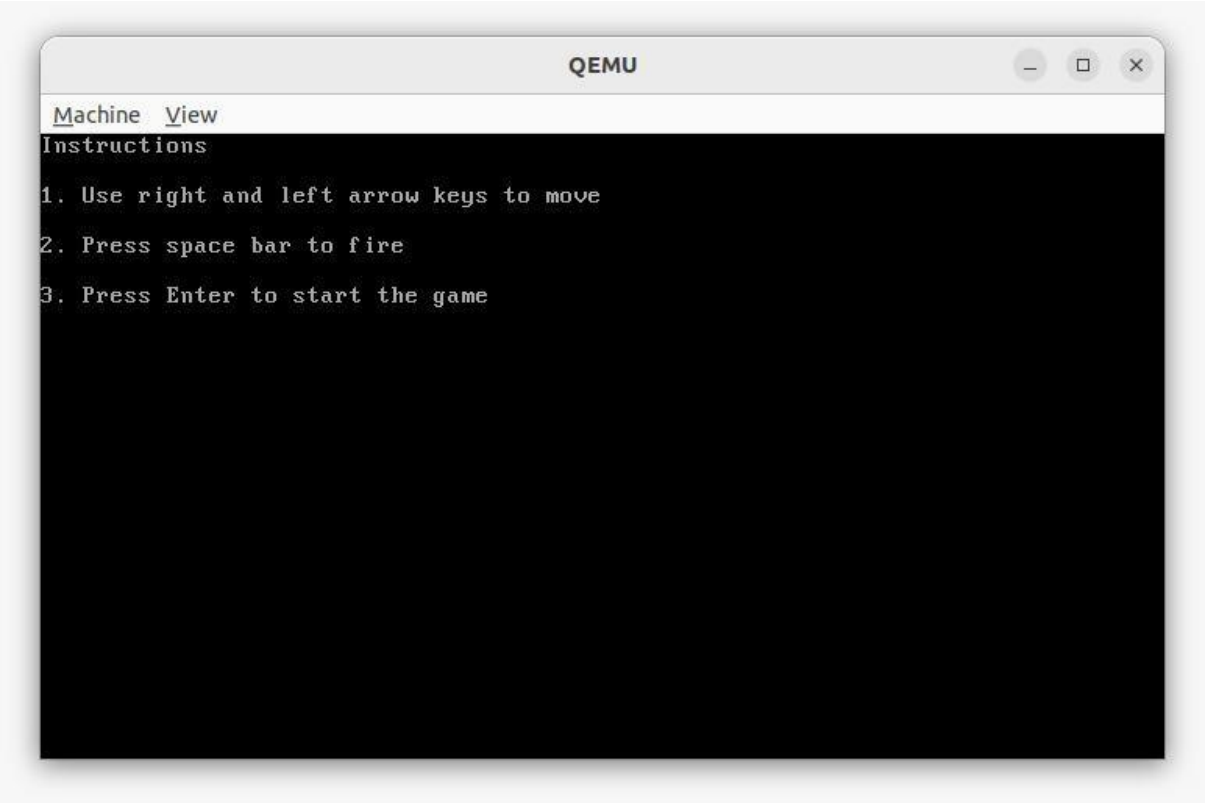


1.2



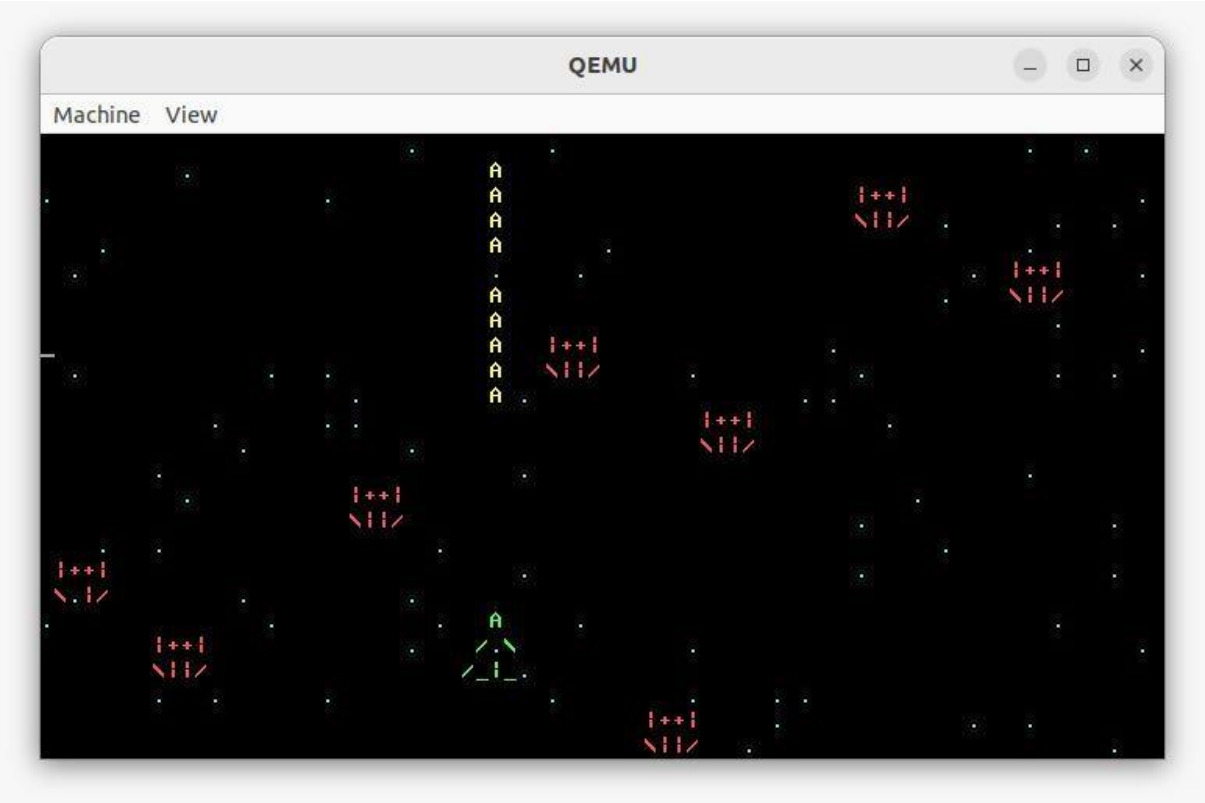
2 – MKEY KERNEL

2.1

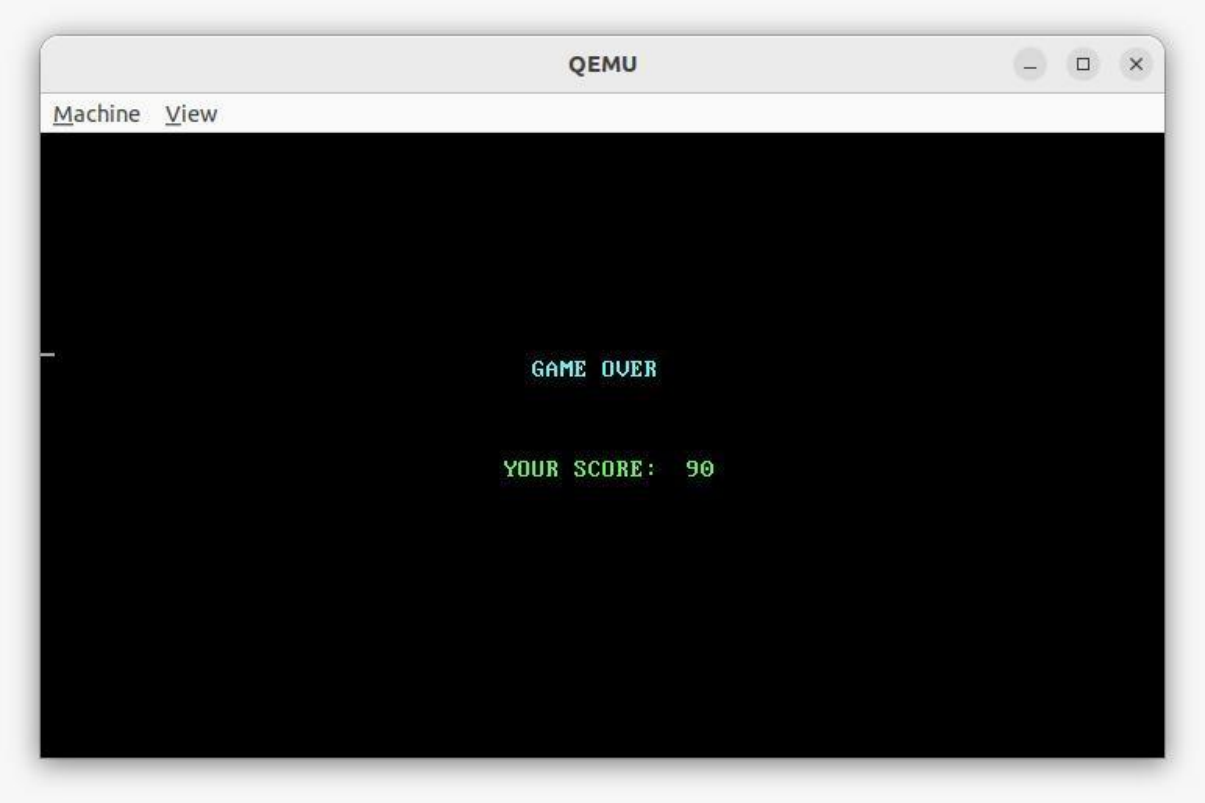


2.2

2.3

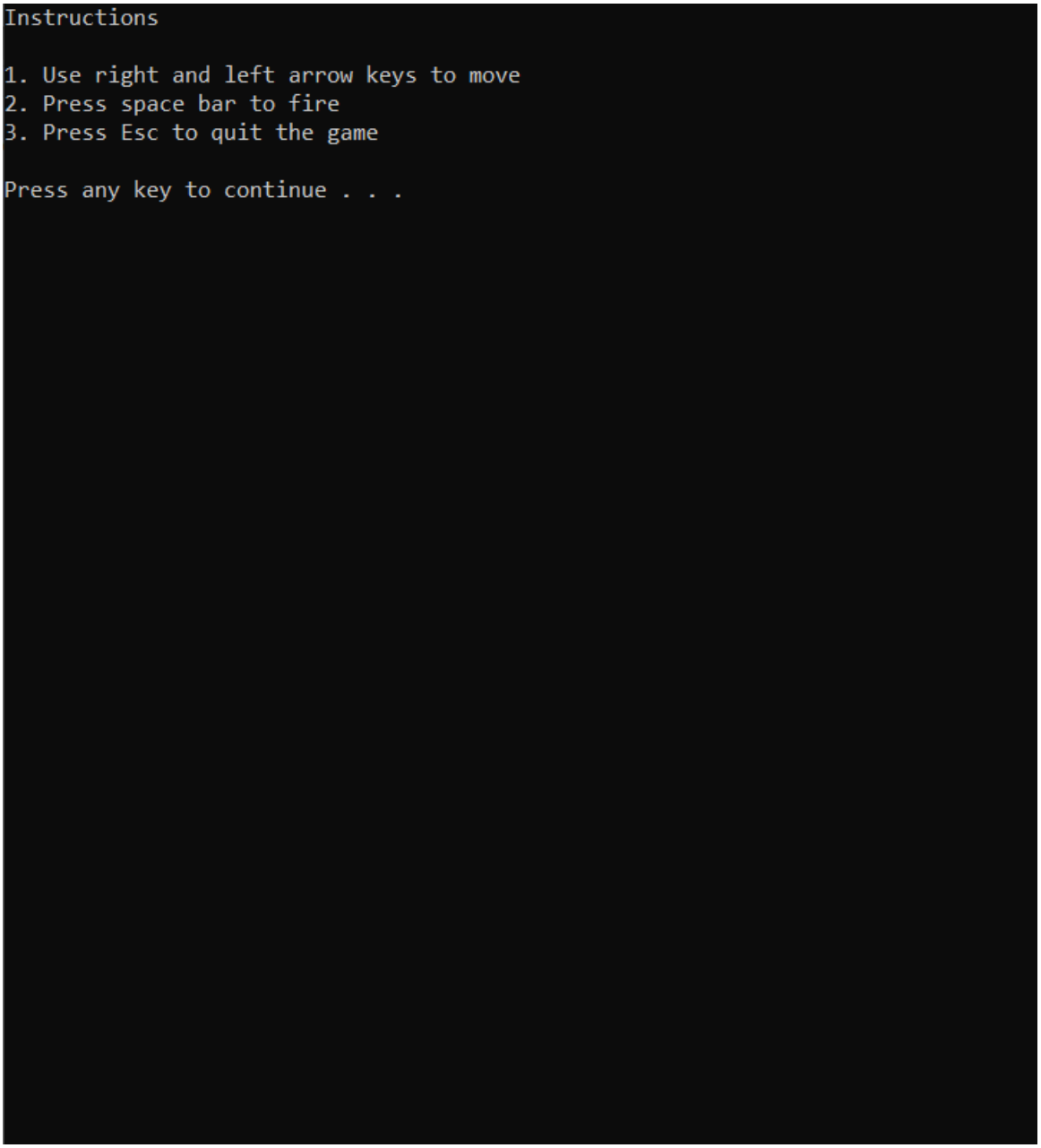


2.4



3 - CONSOLE MODE

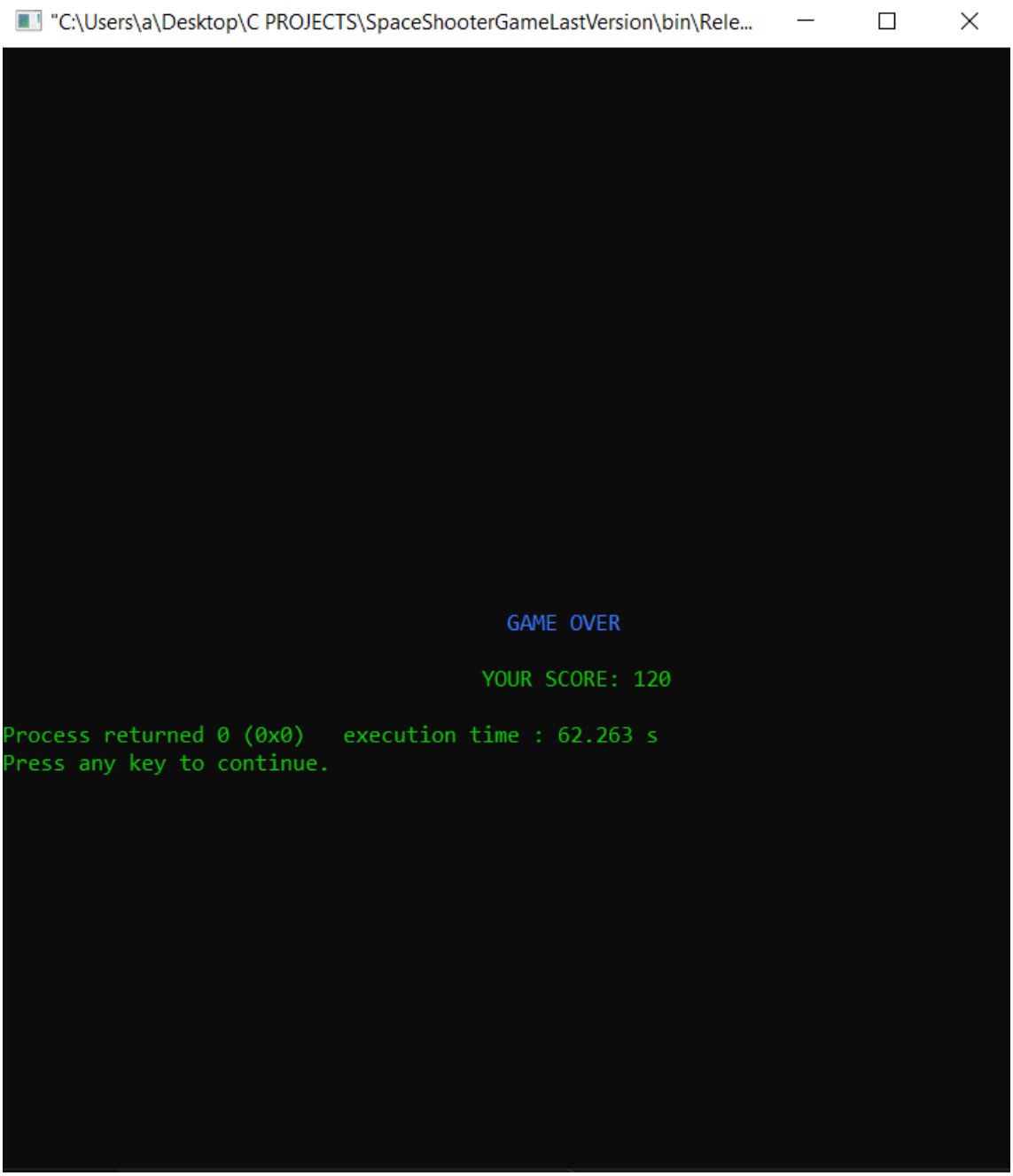
3.1



3.2

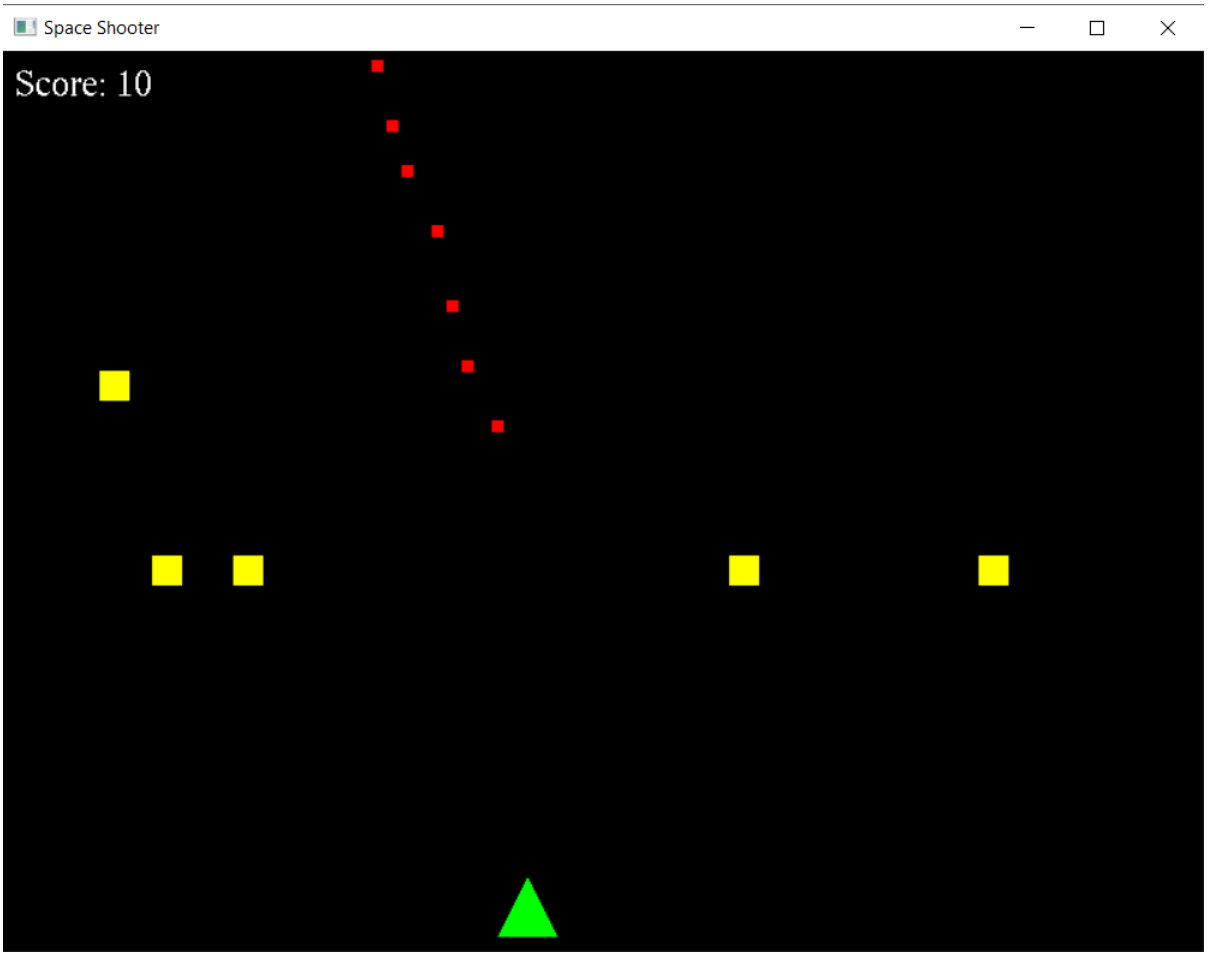

"C:\Users\a\Desktop\C PROJECTS\SpaceShooterGameLastVersion\bin\Rele...

3.3

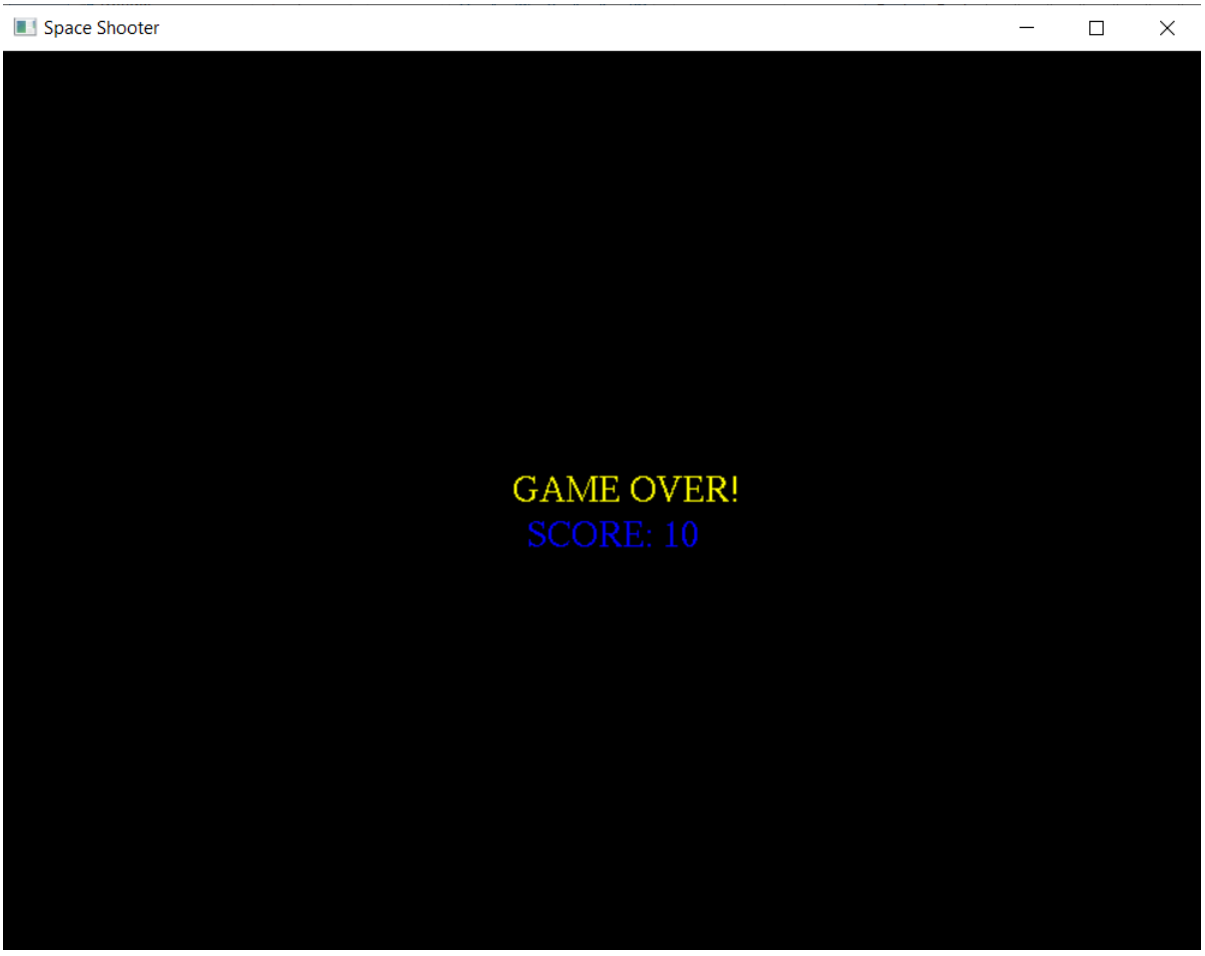

"C:\Users\a\Desktop\C PROJECTS\SpaceShooterGameLastVersion\bin\Rele...
GAME OVER
YOUR SCORE: 120
Process returned 0 (0x0)   execution time : 62.263 s
Press any key to continue.

4 – OPENGL

4.1

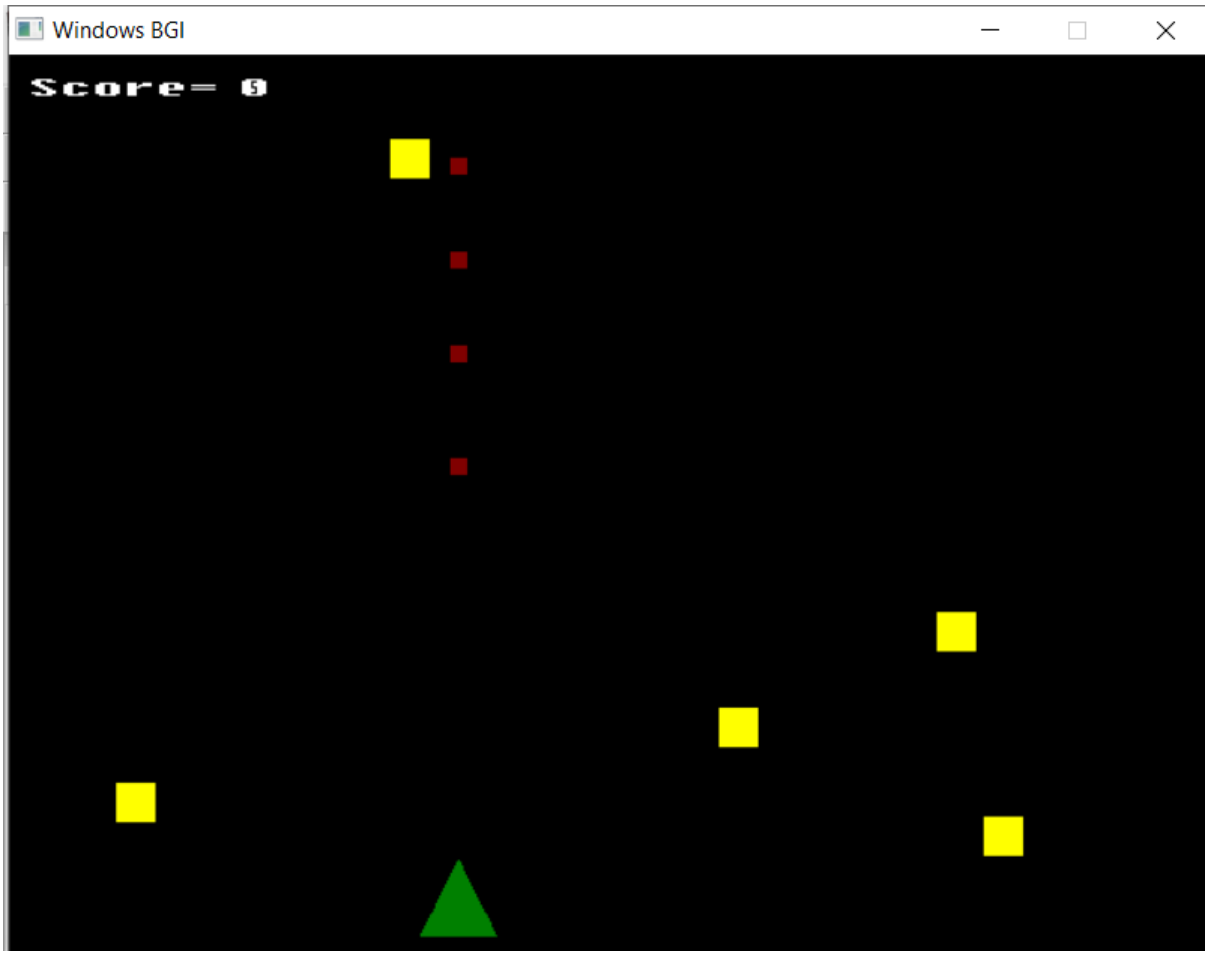


4.2



5 – WINBGI

5.1



5.2