Road Rumble

Session: 2023 – 2027

Submitted by:

Muhammad Talha 2023-CS-169

Supervised by:

Maida Mirza

Course:

CSC-102 Programming Fundamentals

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Short Description of Road Rumble:

- Road Rumble is a survival game where player navigate on a highway, dodging relentless enemies in multiple lanes. Player has the ability to shoot and switch lanes, the goal is to survive from moving enemies and make fire at them.

Game Characters:

1) Player:

Player is 4 x 5 Two Dimensional character who has the ability to move left, right, up and down.

```
char player[4][5] = {{'_','_','|','_','_'},
                     {'|',' ','|',' ','|'},
                    {'_',' ','P',' ','_'},
                    {'|',' ','|',' ','|'},
                     };
```

2) Enemy E1:

Enemy E1 is also a 4 x 5 Two Dimensional Character who plays enemy role in Road Rumble. It will be moving in the first lane of the highway.

```
char enemyE1[4][5] = {{'|','_','x','_','|'},
                      {' ',' ','|',' ',' '},
                      {'|','_','x','_','|'},
                       };
```

3) Enemy E2:

Enemy E1 is also a 4 x 5 Two Dimensional Character who plays enemy role in Road Rumble.

```
char enemyE3[4][5] ={{' ',' ','~',' ',' '},
                     {'(','-','!','-',')'},
                     {' ','(','_',')',' '},
                     };
```

4) Enemy E3:

Enemy E1 is also a 4 x 5 Two Dimensional Character who plays enemy role in Road Rumble.

```
char enemyE3[4][5] = {{'|','_','~','_','|'},
                     {'  ','  ','|','  ','  '},
```

{'|','_','^','_','|'},};

Rules & Interactions:

- This Game is based on survival. ○ There are two Levels in this Game, Level 1 is Easy and Level 2 is Hard ○ You need to protect your your player from being colliding with the enemy.
- There are 3 Lanes in Easy Level. Each Contain one Enemy moving up and down ○ There are 4 Lanes in Hard Level. Each Lane contains Moving Enemy with more speed ○ Use ARROW_KEYS to Move your Player UP, DOWN, LEFT and RIGHT.
- Press SPACE_KEY to generate fires upward and Press SHIFT_KEY to generate fires Downward.
- Press ESCAPE_Key to return from any level in game.
- You have three Hearts, after losing them game will be over
- You have to change lane before Lane Time becomes zero otherwise game will be over ○ You can change your game Level from Start Menu.,

Goal of the Game:

In this game, the main goal is to help the player survive on a busy highway by avoiding moving enemies and shooting them down. Use quick reflexes to switch lanes and shoot strategically. Aim to achieve the highest score by lasting as long as possible in this fast adventure.

This game will entertain the users as well as train them for quick decisions and actions and make them sharp.

Wireframes of the Game:

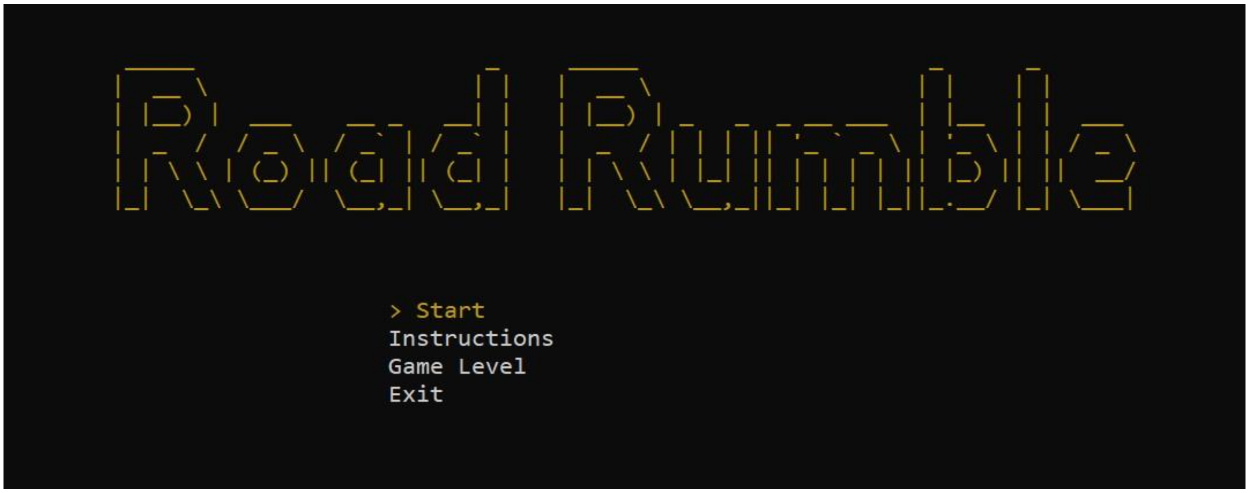
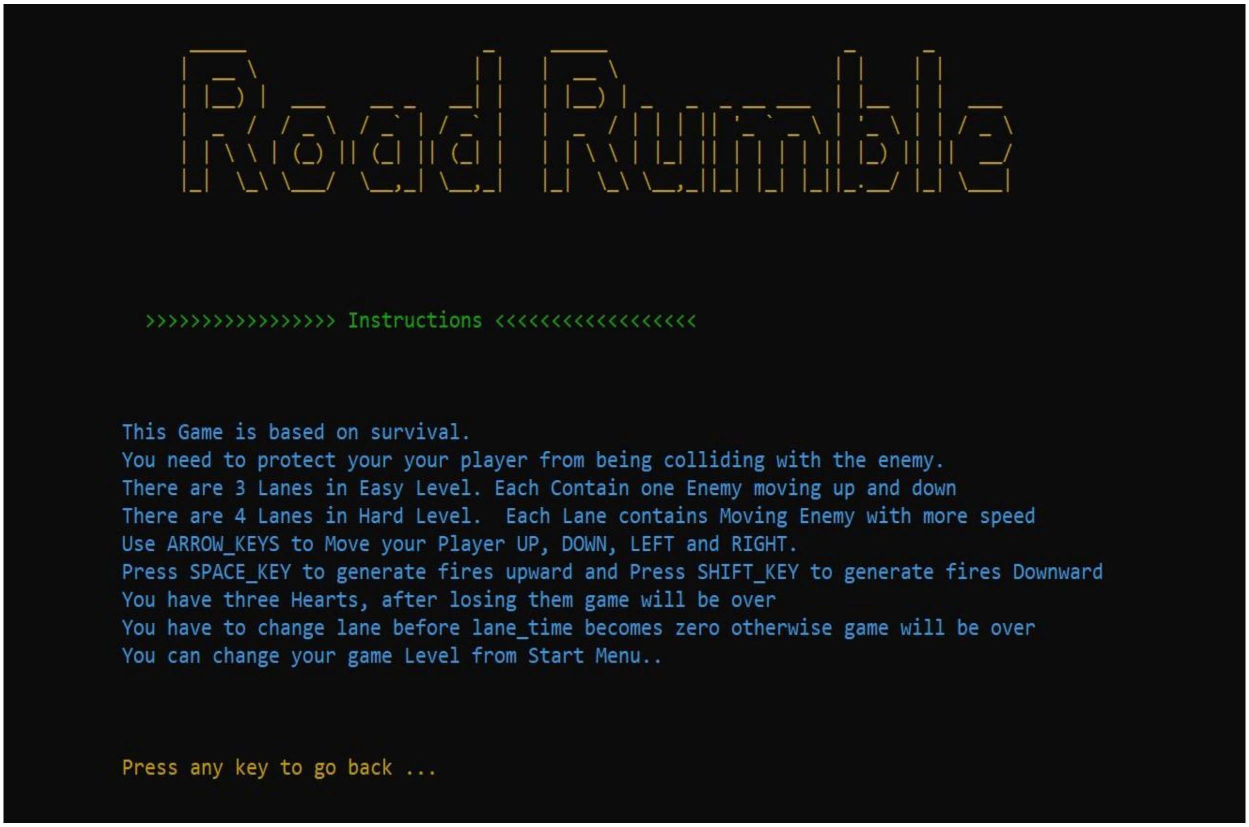


Figure 1: Start Menu

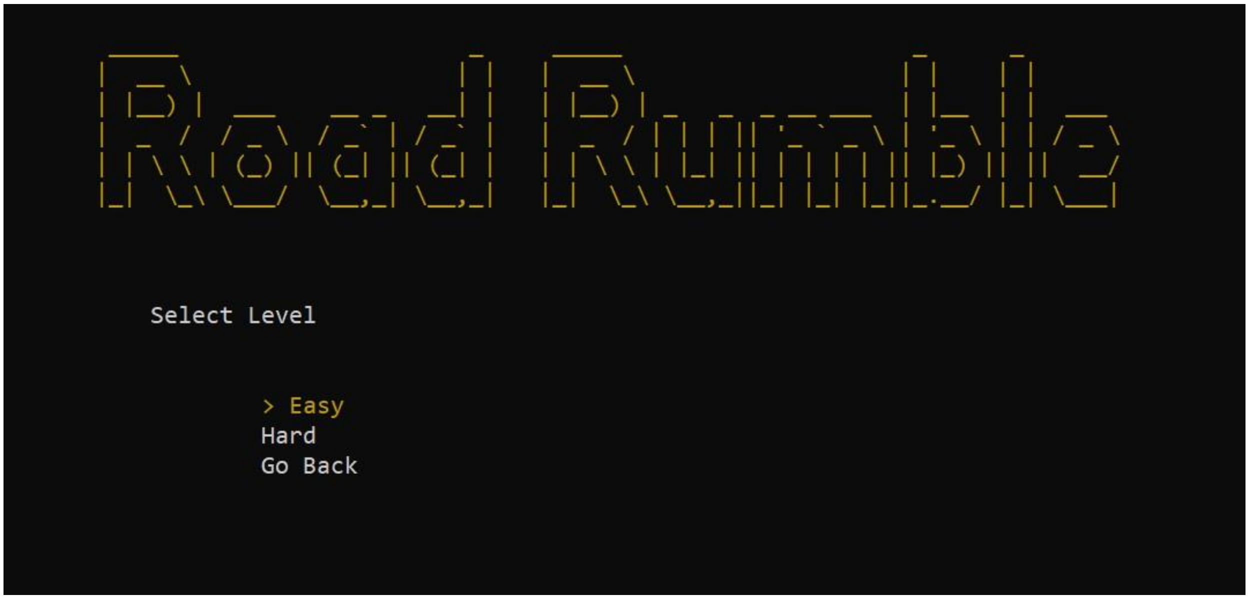


```
>>>>>>>>>>>>>>>>> Instructions <<<<<<<<<<<<<<<<<<


This Game is based on survival.
You need to protect your your player from being colliding with the enemy.
There are 3 Lanes in Easy Level. Each Contain one Enemy moving up and down
There are 4 Lanes in Hard Level.  Each Lane contains Moving Enemy with more speed
Use ARROW_KEYS to Move your Player UP, DOWN, LEFT and RIGHT.
Press SPACE_KEY to generate fires upward and Press SHIFT_KEY to generate fires Downward
You have three Hearts, after losing them game will be over
You have to change lane before lane_time becomes zero otherwise game will be over
You can change your game Level from Start Menu..


Press any key to go back ...
```

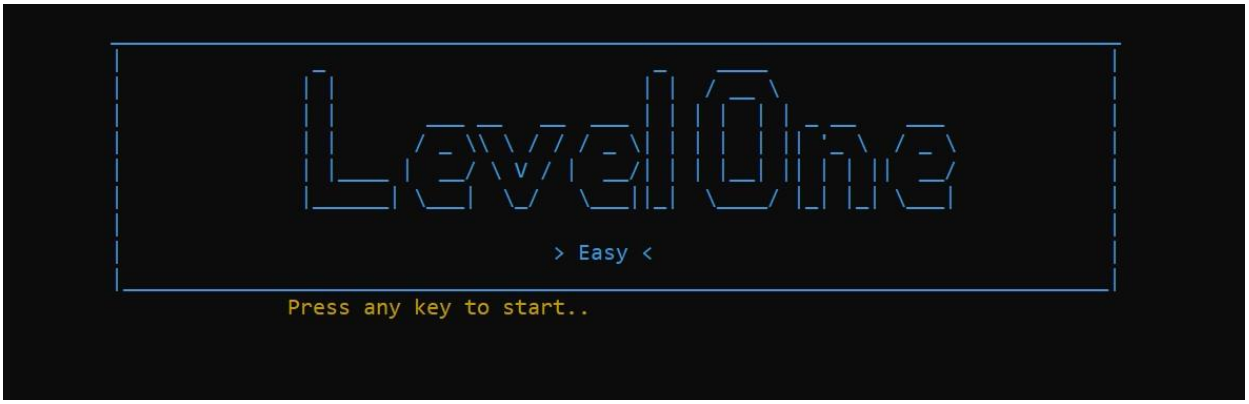
Figure 2: Instructions Menu



```
Select Level


        > Easy
        Hard
        Go Back
```

Figure 3: Levels Menu

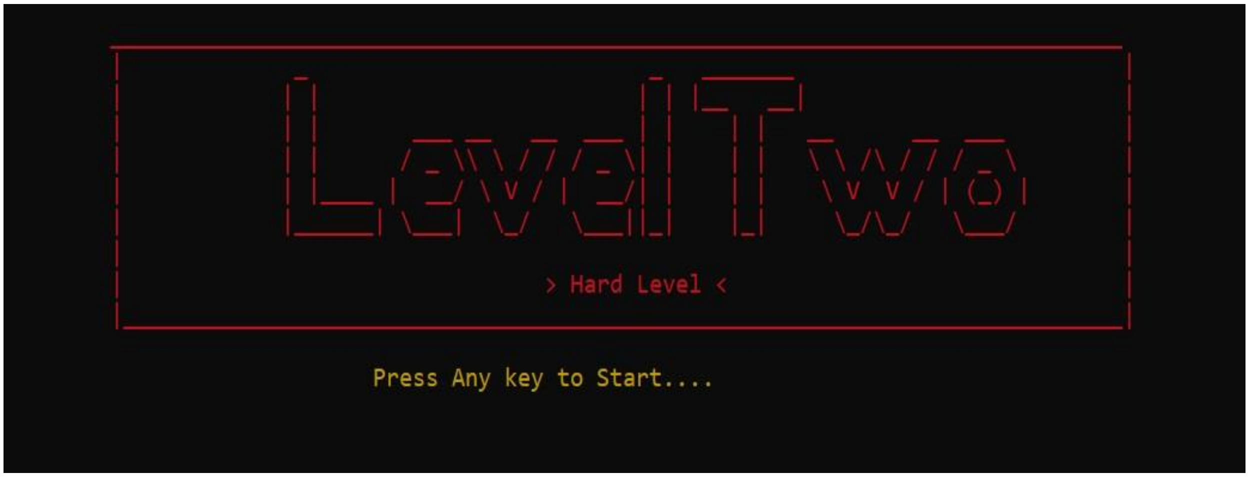


```
LevelOne

> Easy <

Press any key to start..
```

Figure 4: Level One Header

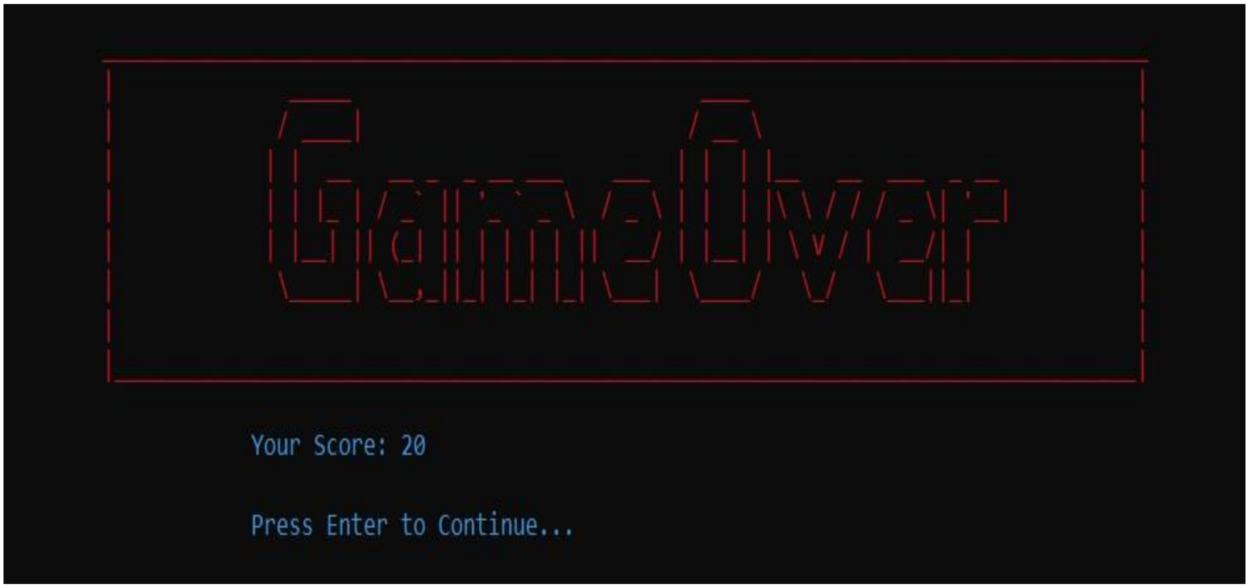


```
LevelTwo

> Hard Level <

Press Any key to Start....
```

Figure 5: Level Two Header

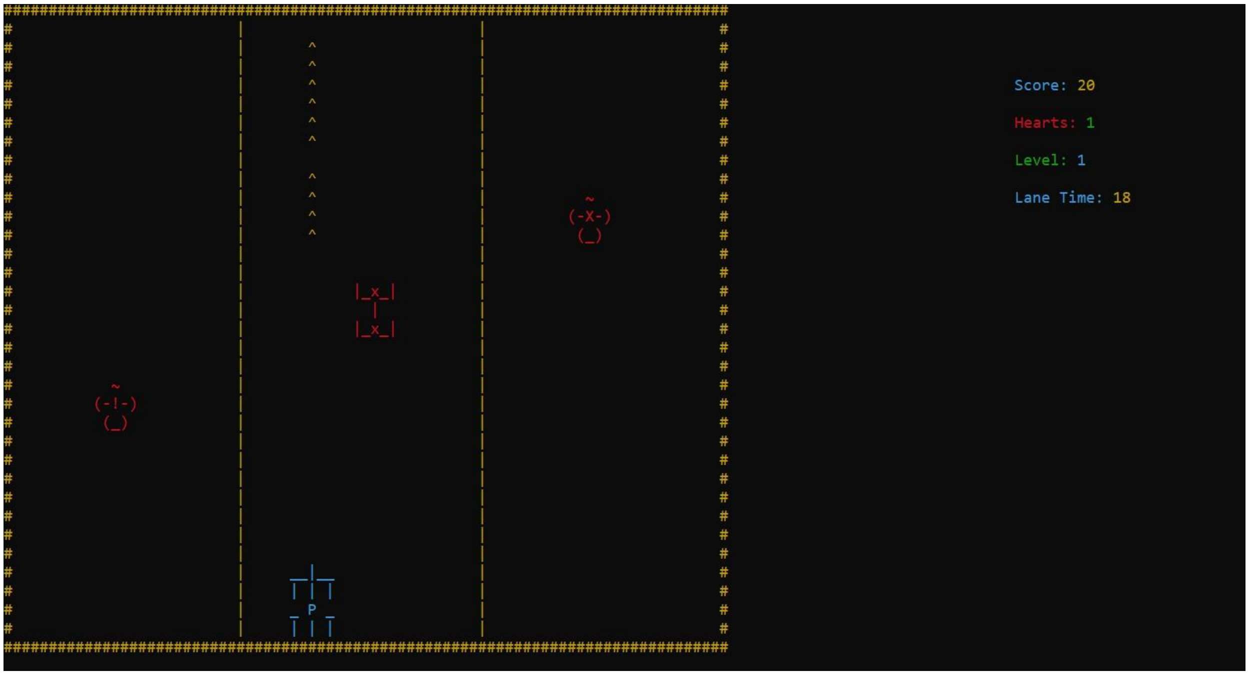


```
GameOver

Your Score: 20

Press Enter to Continue...
```

Figure 6: Instructions Menu



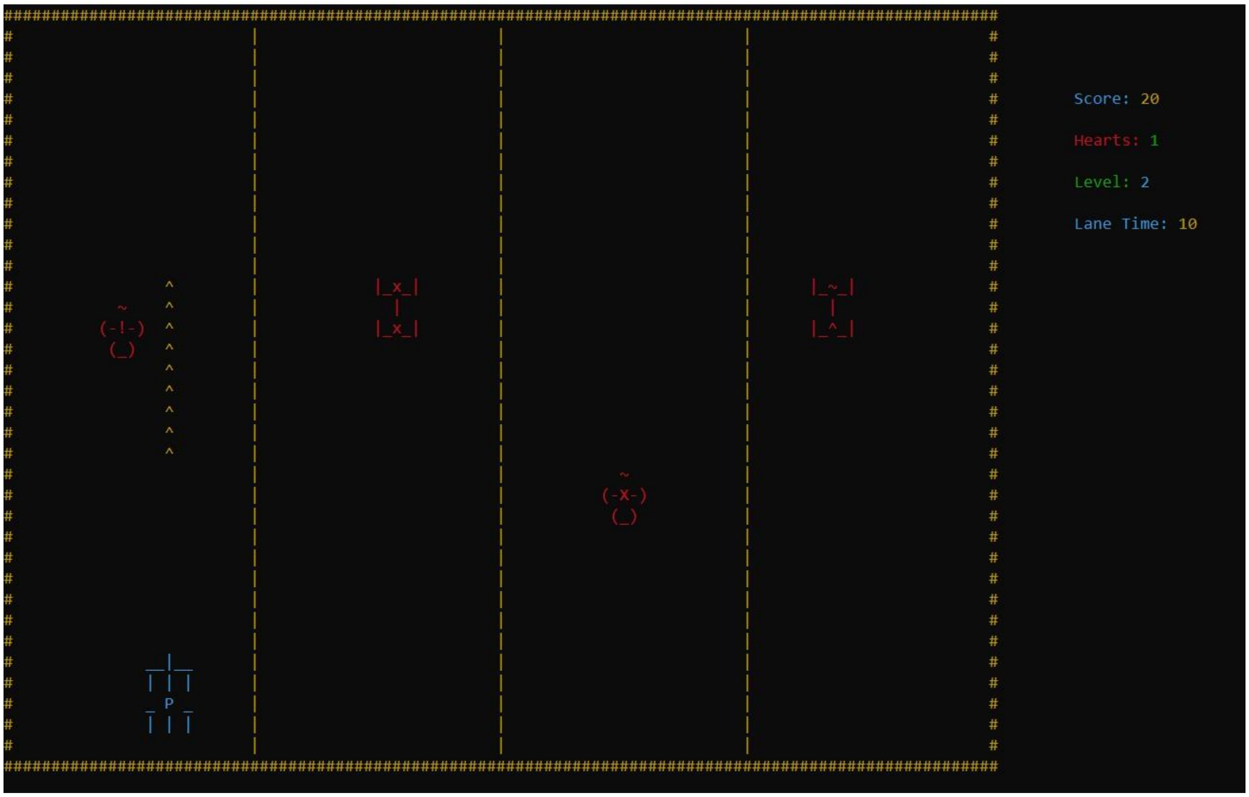
```
Score: 20

Hearts: 1

Level: 1

Lane Time: 18
```

Figure 7: Level One Interface



```
Score: 20

Hearts: 1

Level: 2

Lane Time: 10
```

Figure 8: Level Two Interface

Data Structures (2D Arrays):

```
char player[4][5] = {{'_','_','|','_','_'},
                   {'|',' ','|',' ','|'},
                    {'_',' ','P',' ','_'},
                    {'|',' ','|',' ','|'},
                     };


char enemyE1[4][5] = {{'|','_','x','_','|'},
                      {' ',' ','|',' ',' '},
                       {'|','_','x','_','|'},
                       };


char enemyE2[4][5] ={{' ',' ','~',' ',' '},
                      {'(','-','X','-',')'},
                     {' ','(','_',')',' '},
                      };
char enemyE3[4][5] ={{' ',' ','~',' ',' '},
                      {'(','-','!','-',')'},
                     {' ','(','_',')',' '},
                       };
char enemyE4[4][5] = {{'|','_','~','_','|'},
                      {' ',' ','|',' ',' '},
                       {'|','_','^','_','|'},
                       };
```

Function Prototypes:

```
void printheader();

void start();
void levels_menu();
void Instructions();
void Level_1_Header();
void Level_2_Header();
void printRoadLvl1();
void printroadLvl2();
void Game_Over();
void time_in_lane();

// Hide cursor Function
void hideAndVisibleCursor(bool);
// Side BAR SCORE,LEVEL,HEARTS and Lane Time
void show_score();
void show_health();
void check_health();
void show_lane_time();
void show_level();

void gotoxy(int x, int y);
char getCharAtxy(short int x, short int y);
void Color(int color);

// Player
void printPlayer();
void erasePlayer();
void movePlayerLeft();
void movePlayerRight();
void movePlayerUp();
void movePlayerUp();
void movePlayerDown();

// Enemeies

void printEnemy1();
void printEnemy2();
void printEnemy3();
void printEnemy4();
void eraseE2();
void eraseE3();
void eraseE4();
void eraseE1();
void moveE1();
void moveE2();
void moveE3();
void moveE4();

// Fire System
void fire_by_player();
void print_fire(int, int);
void remove_fires_from_arr(int fX, int fY);
void erase_fire(int frX, int frY);
void remove_fires_from_arr(int fX, int fY);
void move_fire(int frX, int &frY);
void check_fires(int frX, int frY);
```

```
bool check_detection();
```

Complete Code of Road Rumble:

```
#include <iostream>
#include <windows.h>
#include <conio.h>
using namespace std;
// Headers
void printheader();
void start();
void levels_menu();
void Instructions();
void Level_1_Header();
void Level_2_Header();
void printRoadLvl1();
void printroadLvl2();
void Game_Over();
void time_in_lane();

// Hide cursor Function
void hideAndVisibleCursor(bool);
// Side BAR SCORE,LEVEL,HEARTS and Lane Time
void show_score();
void show_health();
void check_health();
void show_lane_time();
void show_level();
void gotoxy(int x, int y);
```

```
    char getCharAtxy(short int x, short int y);
    void Color(int color);

    // Player
    void printPlayer();
    void erasePlayer();
    void movePlayerLeft();
    void movePlayerRight();
    void movePlayerUp();
    void movePlayerUp();
    void movePlayerDown();

    // Enemeies


    void printEnemy1();
    void printEnemy2();
    void printEnemy3();
    void printEnemy4();
    void eraseE2();
    void eraseE3();
    void eraseE4();
    void eraseE1();
    void moveE1();
    void moveE2();
    void moveE3();
    void moveE4();
```

```
// Fire System
void fire_by_player();
void print_fire(int, int);
void remove_fires_from_arr(int fX, int fY);
void erase_fire(int frX, int frY);
void remove_fires_from_arr(int fX, int fY);
void move_fire(int frX, int &frY);
void check_fires(int frX, int frY);
bool check_detection();

// Coordinates
int px = 40, py = 20;
int pxE1 = 39, pyE1 = 18;
int pxE2 = 63, pyE2 = 23;
int pxE3 = 10, pyE3 = 2;
int pxE4 = 85, pyE4 = 26;
// Hearts , score, lanetime and level
int score = 0;
int hearts = 10;
int lane_time = 50;
// Fires
char fires[100];
int frX[100];
int frY[100];
int index_fr = 0;

// Enemies Direction
```

```
    int direction = 0, direction2 = 0, direction3 = 0, direction4 = 0;


    // Level
    int Level = 1;


    // ALL OBJECTS
    char player[4][5] = {                          //Player
        {'_', '_', '|', '_', '_'},
        {'|', ' ', '|', ' ', '|'},
        {'_', ' ', 'P', ' ', '_'},
        {'|', ' ', '|', ' ', '|'},
    };
    // Enemies
    char enemyE1[4][5] = {
        {'|', '_', 'x', '_', '|'},
        {' ', ' ', '|', ' ', ' '},
        {'|', '_', 'x', '_', '|'},
    };


    char enemyE2[4][5] = {
        {' ', ' ', '~', ' ', ' '},
        {'(', '-', 'X', '-', ')'},
        {' ', '(', '_', ')', ' '},
    };
    char enemyE3[4][5] = {
        {' ', ' ', '~', ' ', ' '},
        {'(', '-', '!', '-', ')'},
        {' ', '(', '_', ')', ' '},
    };
    char enemyE4[4][5] = {
        {'|', '_', '~', '_', '|'},
        {' ', ' ', '|', ' ', ' '},
        {'|', '_', '^', '_', '|'},
    };

    // Main
    main()
    {
        system("cls");
        system("color 06");
        hideAndVisibleCursor(false);
        printheader();
        start();
    }
    // Level 1
    void Level_Easy_1()
    {
        system("cls");
        Level_1_Header();
        cout << "\t\t\t\t\t\t"
             << "Press any key to start..";
        getch();
        system("cls");
        printRoadLvl1();
    bool gameover = false;
    while (true)
    {
        hideAndVisibleCursor(false);

        if (GetAsyncKeyState(VK_LEFT))
        {
            movePlayerLeft();
            lane_time = 50;
        }
        if (GetAsyncKeyState(VK_RIGHT))
        {
            movePlayerRight();
            lane_time = 50;
        }
        if (GetAsyncKeyState(VK_UP))
        {
            movePlayerUp();
        }
        if (GetAsyncKeyState(VK_DOWN))
        {
            movePlayerDown();
        }
        if (GetAsyncKeyState(VK_SPACE))
        {
            fire_by_player();
        }
```

```
            if (GetAsyncKeyState(VK_ESCAPE))
            {
                break;
            }
            for (int i = 0; i < index_fr; i++)
            {
                move_fire(frX[i], frY[i]);
                check_fires(frX[i], frY[i]);
            }
            moveE1();
            moveE2();
            moveE3();
            Sleep(35);
            show_score();
            show_health();
            show_level();
            time_in_lane();
            show_lane_time();
            check_detection();
            if (lane_time <= 0 || hearts == 0)
            {
                gameover = true;
                lane_time = 50;
                break;
            }
        }
        if (gameover)
        {
            system("cls");
            Game_Over();
            Sleep(1800);
        }
    }

    // Level 2 HARD
    void Level_2_Hard()
    {
        system("cls");
        Level_2_Header();
        cout << "\t\t\t\t\t\t\t"
             << "Press Any key to Start....\n\n";
        getch();
        system("cls");
        printroadLvl2();
        bool gameover = false;

        while (true)
        {
            hideAndVisibleCursor(false);

            if (GetAsyncKeyState(VK_LEFT))
            {
                movePlayerLeft();
                lane_time = 50;
```

```
}
if (GetAsyncKeyState(VK_RIGHT))
{
    movePlayerRight();
    lane_time = 50;
}
if (GetAsyncKeyState(VK_UP))
{
    movePlayerUp();
}
if (GetAsyncKeyState(VK_DOWN))
{
    movePlayerDown();
}
if (GetAsyncKeyState(VK_SPACE))
{
    fire_by_player();
}
if (GetAsyncKeyState(VK_ESCAPE))
{
    break;
}
for (int i = 0; i < index_fr; i++)
{
    move_fire(frX[i], frY[i]);
    check_fires(frX[i], frY[i]);
}
```

```
            moveE1();
            moveE2();
            moveE3();
            moveE4();
            Sleep(17);
            show_score();
            show_health();
            show_level();
            time_in_lane();
            show_lane_time();
            check_detection();
            if (lane_time <= 0 || hearts == 0)
            {
                gameover = true;
                lane_time = 50;
                break;
            }
        }
        if (gameover)
        {
            system("cls");
            Game_Over();
            Sleep(1500);
        }
    }
    // Start Menu of Road Rumble
    void start()
```

```
{
    char key;
    string menus[4] = {"Start", "Instructions", "Game Level", "Exit"};
    int current_opt = 0;

    while (true)
    {
        system("cls");
        printheader();
        for (int i = 0; i < 4; i++)
        {
            if (i == current_opt)
            {
                Color(06);
                cout << "\t\t\t\t\t\t\t\t"
                     << "> " << menus[i] << endl;
                Color(07);
            }
            else
            {
                cout << "\t\t\t\t\t\t\t\t" << menus[i] << endl;
            }
        }

        key = 0;
        while (!(key == 80 || key == 72 || key == 13))
        {
```

```
    key = _getch();
} // User can only use these three keys,   1.Enter     2.Up Key        3. Down key


if (key == 80) // Down Key
{
    if (current_opt < 3)
    {
        current_opt++;
    }
}
else if (key == 72) // Up Key
{
    if (current_opt > 0)
    {
        current_opt--;
    }
}
else if (key == 13)
{
    if (current_opt == 0)
    {
        if (Level == 1)
        {
            Level_Easy_1();
        }
        else if (Level == 2)
        {
```

```
                    Level_2_Hard();
                 }
              }
              else if (current_opt == 1)
              {
                 Instructions();
              }
              else if (current_opt == 2)
              {
                 levels_menu();
              }
              else if (current_opt == 3)
              {
                 break;
              }
           }
        }
     }
                                                          // Set Levels Menu
     void levels_menu()
     {

        string levels[3] = {"Easy", "Hard", "Go Back"};
        char key;
        int current_opt = 0;

        while (true)
```

```
    {
        system("cls");
        printheader();
        cout << "\t\t\t\t\t\t"
             << "Select Level \n\n\n";
        // Keys Colors
        for (int i = 0; i < 3; i++)
        {
            if (i == current_opt)
            {
                Color(06);
                cout << "\t\t\t\t\t\t\t"
                     << "> " << levels[i] << endl;
                Color(07);
            }
            else
            {
                cout << "\t\t\t\t\t\t\t" << levels[i] << endl;
            }
        }
        // Up Down Keys
        key = 0;
        while (!(key == 80 || key == 72 || key == 13))
        {
            key = _getch();
        }
        if (key == 80) // Down Key
```

```
{
    if (current_opt < 2)
    {
        current_opt++;
    }
}
else if (key == 72) // Up Key
{
    if (current_opt > 0)
    {
        current_opt--;
    }
}
else if (key == 13)
{
    if (current_opt == 0)
    {
        Level = 1;
        Color(03);
        cout << "\n\n\t\t\t\t\tGame Level Changed to Easy.... \n";
        Color(07);
        Sleep(900);
        break;
    }
    else if (current_opt == 1)
    {
        Level = 2;
        Color(04);
        cout << "\n\n\t\t\t\t\t\t\tGame Level Changed to Hard.... \n";
        Color(07);
        Sleep(900);
        break;
      }
      else if (current_opt == 2)
      {
        break;
      }
    }
  }
}
                                        // Instructions Menu
void Instructions()
{
  system("cls");
  printheader();
  Color(02);
  cout << endl;
  cout << "\t\t\t\t\t >>>>>>>>>>>>>>>>> Instructions <<<<<<<<<<<<<<<<<<\n";
  Color(03);
  cout << endl
     << endl
     << endl;
  cout << "\t\t\t\t     This Game is based on survival. \n";
  cout << "\t\t\t\t     You need to protect your your player from being colliding with the
enemy. \n";
```

```
    cout << "\t\t\t\t       There are 3 Lanes in Easy Level. Each Contain one Enemy moving up
and down \n";
    cout << "\t\t\t\t       There are 4 Lanes in Hard Level.  Each Lane contains Moving Enemy
with more speed\n";
    cout << "\t\t\t\t       Use ARROW_KEYS to Move your Player UP, DOWN, LEFT and
RIGHT.\n";
    cout << "\t\t\t\t       Press SPACE_KEY to generate fires upward and Press SHIFT_KEY
to generate fires Downward\n";
    cout << "\t\t\t\t       You have 10 Hearts, after losing them game will be over\n";
    cout << "\t\t\t\t       You have to change lane before lane_time becomes zero otherwise
game will be over\n";
    cout << "\t\t\t\t       You can change your game Level from Start Menu..\n\n\n\n";
    Color(06);
    cout << "\t\t\t\t       Press any key to go back ... ";
    Color(07);
    getch();
}
                                        // Header Level One
void Level_1_Header()
{
    Color(03);
    cout << R"(

_______________________________________________________________________________
______
                      |          _                  _   ____                   |
                      |         | |                | | / __ \                 |
                      |         | |     ___ __   __ ___| | | | | |_ __   ___          |
                      |         | |    / _ \\ \ / / _ \| | | | | |' _ \ / _ \        |
                      |         | |____|  __/ \ V / |  __/| | | |__| || | | ||  __/         |
________________________________________________________________________________
                    |        |_______| \____|  \_/   \____||_|  \_____/ |_| |_| \____|         |
                    |                                                    |
                    |                    > Easy Level <                          |

   |________________________________________________________________________________
   _____|
     )";
     Color(06);
   }
                                   // Header Level Two
   void Level_2_Header()
   {
     Color(04);
     cout << R"(

   ________________________________________________________________________________
   ___
                     |        _                  _   ________                |
                     |       | |                 | | |__   __|              |
                     |       | |     ____ __   __  ____ | |   | |  __     __  ____      |
                     |       | |    / _ \\ \ / / / _ \| |   | |  \ \ /\ / / / _ \      |
                     |       | |_____ |  __/ \ V / |  __/| |   | |   \ V  V / | (_) |     |
                     |       |_______| \____|  \_/   \____||_|   |_|    \_/\_/   \____/      |
                     |                                                    |
                     |                    > Hard Level <                          |

   |________________________________________________________________________________
   __|
     )";
     Color(06);
   }
                                           // Game Over
   void Game_Over()
   {
     system("cls");
     Color(04);
     cout << R"(

   ____________________________________________________________________________
   __________
                     |          _____                 ____                  |
                     |         / ____|               / __ \                 |
                     |        | |  __   __ _  _ __ ___   ___  | |  | |__   __  ___  _ __        |
                     |        | | |_ | / _` || '_ ` _ \  / _ \ | |  | |\ \ / / / _ \| '__|        |
                     |        | |__| || (_| || | | | | ||  __/ | |__| | \ V / |  __/| |          |
                     |         \_____| \__,_||_| |_| |_| \___|  \____/   \_/   \___||_|          |
                     |                                                        |

   |____________________________________________________________________________
   ________|
     )";
     Color(03);
     cout << endl;
     cout << "\t\t\t\t\t\t"
         << "Your Score: " << score;
     hearts = 10;
     cout << "\n\n\t\t\t\t\t\tPress Enter to Continue... ";
     getch();
   }
                                           // Main Header


   void printheader()
   {
     Color(06);
     cout << R"(


                              _____                 _   _____                 _     _
                             |  __ \               | | |  __ \               | |   | |
                             | |__) | ___   __ _  __| |  | |__) |_   _ _ __ ___  | |__  | | ___
                             |  _  /  / _ \ / _` |/ _` |  |  _  / | | | || '_ ` _ \ | '_ \ | | / _ \
                             | | \ \ | (_) || (_| || (_| |  | | \ \ | |_| || | | | | || |_) || ||  __/
                             |_|  \_\ \___/  \__,_| \__,_|  |_|  \_\ \__,_||_| |_| |_||_.__/ |_| \___|



     )";
     cout << endl;
     Color(07);
   }
                                           // Maze level One
   void printRoadLvl1()
   {
     Color(06);
     cout <<
   "#################################################################################" << endl;
```

```
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << "#                    |                    |                    #" << endl;
    cout << 
"#########################################################################
########" << endl;
    Color(07);
}
                                    // Maze Level 2
void printroadLvl2()
{
    Color(06);
    cout << 
"#########################################################################
################################ " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout << "#                    |                    |                    |                    # " << endl;
    cout <<
  "#################################################################################
  #################################### " << endl;
    Color(07);
    }
                                        // Print and Erase Player
    void printPlayer()
    {
        int y = py;
        Color(03);
        for (int i = 0; i < 4; i++)
        {
            gotoxy(px, y);
            for (int j = 0; j < 5; j++)
            {
                cout << player[i][j];
            }
            y++;
        }
        Color(07);
    }
    void erasePlayer()
    {
        gotoxy(px, py);
        cout << "     " << endl;
        gotoxy(px, py + 1);
        cout << "     " << endl;
        gotoxy(px, py + 2);
        cout << "     " << endl;
        gotoxy(px, py + 3);
        cout << "     " << endl;
```

```
}
                                            // Print and Erase Enemies
void printEnemy1()
{
    int y = pyE1;
    Color(04);
    for (int i = 0; i < 4; i++)
    {
        gotoxy(pxE1, y);
        for (int j = 0; j < 5; j++)
        {
            cout << enemyE1[i][j];
        }
        y++;
    }
    Color(07);
}
void printEnemy2()
{
    int y = pyE2;
    Color(04);
    for (int i = 0; i < 4; i++)
    {
        gotoxy(pxE2, y);
        for (int j = 0; j < 5; j++)
        {
            cout << enemyE2[i][j];
        }
        y++;
    }
    Color(07);
}
void printEnemy3()
{
    int y = pyE3;
    Color(04);
    for (int i = 0; i < 4; i++)
    {
        gotoxy(pxE3, y);
        for (int j = 0; j < 5; j++)
        {
            cout << enemyE3[i][j];
        }
        y++;
    }
    Color(07);
}
void printEnemy4()
{
    int y = pyE4;
    Color(04);
    for (int i = 0; i < 4; i++)
    {
        gotoxy(pxE4, y);
        for (int j = 0; j < 5; j++)
        {
            cout << enemyE4[i][j];
        }
        y++;
    }
    Color(07);
}
void eraseE1()
{
    for (int i = 0; i < 4; i++)
    {
        gotoxy(pxE1, pyE1 + i);
        for (int j = 0; j < 5; j++)
        {
            cout << ' ';
        }
    }
}
void eraseE2()
{
    for (int i = 0; i < 4; i++)
    {
        gotoxy(pxE2, pyE2 + i);
        for (int j = 0; j < 5; j++)
        {
            cout << ' ';
        }
    }
}
void eraseE3()
{
    for (int i = 0; i < 4; i++)
    {
        gotoxy(pxE3, pyE3 + i);
        for (int j = 0; j < 5; j++)
        {
            cout << ' ';
        }
    }
}
void eraseE4()
{
    for (int i = 0; i < 4; i++)
    {
        gotoxy(pxE4, pyE4 + i);
        for (int j = 0; j < 5; j++)
        {
            cout << ' ';
        }
    }
}

                            // Move Functions for Player
```

```
void movePlayerLeft()
{
    if (getCharAtxy(px - 3, py) != '#')
    {
        erasePlayer();
        px--;
        printPlayer();
    }
    if (getCharAtxy(px - 4, py) == '|' || getCharAtxy(px - 5, py) == '|')
    {
        erasePlayer();
        px -= 18;
        printPlayer();
    }
}
void movePlayerRight()
{
    if (getCharAtxy(px + 7, py) != '#')
    {
        erasePlayer();
        px++;
        printPlayer();
    }
    if (getCharAtxy(px + 7, py) == '|' || getCharAtxy(px + 8, py) == '|')
    {
        erasePlayer();
        px += 18;
```

```
            printPlayer();
        }
    }
    void movePlayerUp()
    {
        if (getCharAtxy(px, py - 2) != '#')
        {
            erasePlayer();
            py--;
            printPlayer();
        }
    }
    void movePlayerDown()
    {
        if (getCharAtxy(px, py + 4) != '#')
        {
            erasePlayer();
            py++;
            printPlayer();
        }
    }
                                // Move Enemies
    void moveE1()
    {
        if (direction == 0)
        {
            if (getCharAtxy(pxE1, pyE1 - 2) != '#')
            {
                eraseE1();
                pyE1--;
                printEnemy1();
            }
            else
            {
                direction = 1;
            }
        }
        else if (direction == 1)
        {
            if (getCharAtxy(pxE1, pyE1 + 5) != '#')
            {
                eraseE1();
                pyE1++;
                printEnemy1();
            }
            else
            {
                direction = 0;
            }
        }
    }
    void moveE2()
    {
        if (direction2 == 0)
        {
            if (getCharAtxy(pxE2, pyE2 - 2) != '#')
            {
                eraseE2();
                pyE2--;
                printEnemy2();
            }
            else
            {
                direction2 = 1;
            }
        }
        else if (direction2 == 1)
        {
            if (getCharAtxy(pxE2, pyE2 + 5) != '#')
            {
                eraseE2();
                pyE2++;
                printEnemy2();
            }
            else
            {
                direction2 = 0;
            }
        }
    }
    void moveE3()
```

```
{
    if (direction3 == 0)
    {
        if (getCharAtxy(pxE3, pyE3 - 2) != '#')
        {
            eraseE3();
            pyE3--;
            printEnemy3();
        }
        else
        {
            direction3 = 1;
        }
    }
    else if (direction3 == 1)
    {
        if (getCharAtxy(pxE3, pyE3 + 5) != '#')
        {
            eraseE3();
            pyE3++;
            printEnemy3();
        }
        else
        {
            direction3 = 0;
        }
    }
```

```
}
void moveE4()
{
    if (direction4 == 0)
    {
        if (getCharAtxy(pxE4, pyE4 - 2) != '#')
        {
            eraseE4();
            pyE4--;
            printEnemy4();
        }
        else
        {
            direction4 = 1;
        }
    }
    else if (direction4 == 1)
    {
        if (getCharAtxy(pxE4, pyE4 + 5) != '#')
        {
            eraseE4();
            pyE4++;
            printEnemy4();
        }
        else
        {
            direction4 = 0;
        }
    }
}
                                    // Firing Functions
void fire_by_player()
{
    fires[index_fr] = '^';
    frX[index_fr] = px + 2;
    frY[index_fr] = py - 1;
    print_fire(frX[index_fr], frY[index_fr]);
    index_fr++;
}
void print_fire(int frX, int frY)
{
    if (getCharAtxy(frX - 2, frY - 1) != '#')
    {
        gotoxy(frX, frY);
        cout << "^";
    }
    else
    {
        remove_fires_from_arr(frX, frY);
    }
}
void remove_fires_from_arr(int fX, int fY)
{
    int index;
    for (int i = 0; i < index_fr; i++)
    {
        if (frY[i] == fY && frX[i] == fX)
        {
            index = i;
            break;
        }
    }
    for (int j = index; j < index_fr - 1; j++)
    {
        fires[j] = fires[j + 1];
        frX[j] = frX[j + 1];
        frY[j] = frY[j + 1];
    }
    index_fr--;
}
void erase_fire(int frX, int frY)
{
    gotoxy(frX, frY);
    cout << " ";
}
void move_fire(int frX, int &frY)
{
    erase_fire(frX, frY);
    frY--;
    print_fire(frX, frY);
}
```

// Check Fires collision

```
void check_fires(int frX, int frY)
{
    if (frX < pxE1 + 6 && frX > pxE1 && frY <= pyE1 + 5 && frY > pyE1)
    {
        eraseE1();
        pxE1 = 39;
        pyE1 = 4;
        score += 5;
        erase_fire(frX, frY);
        remove_fires_from_arr(frX, frY);
        moveE1();
    }
    if (frX < pxE2 + 6 && frX > pxE2 && frY <= pyE2 + 5 && frY > pyE2)
    {
        eraseE2();
        pxE2 = 63;
        pyE2 = 4;
        score += 5;
        erase_fire(frX, frY);
        remove_fires_from_arr(frX, frY);
        moveE2();
    }
    if (frX < pxE3 + 6 && frX > pxE3 && frY <= pyE3 + 4 && frY > pyE3)
    {
        eraseE3();
        pxE3 = 10;
```

```
            pyE3 = 4;
            score += 5;
            erase_fire(frX, frY);
            remove_fires_from_arr(frX, frY);
            moveE3();
        }
        if (frX < pxE4 + 6 && frX > pxE4 && frY <= pyE4 + 4 && frY > pyE4)
        {
            eraseE4();
            pxE4 = 85;
            pyE4 = 4;
            score += 5;
            erase_fire(frX, frY);
            remove_fires_from_arr(frX, frY);
            moveE4();
        }
    }
                                        // Check Player Colliosion with enemy
    bool check_detection()
    {
        if ((px <= pxE1 + 5 && px + 5 >= pxE1) && (py <= pyE1 + 4 && py + 4 >= pyE1))
        {
            if (hearts > 0)
            {
                hearts--;
                eraseE1();
                pyE1 = 3;
                printEnemy1();
            }
        }
        if ((px <= pxE2 + 5 && px + 5 >= pxE2) && (py <= pyE2 + 4 && py + 4 >= pyE2))
        {
            if (hearts > 0)
            {
                hearts--;
                eraseE2();
                pyE2 = 3;
                printEnemy2();
            }
        }
        if ((px <= pxE3 + 5 && px + 5 >= pxE3) && (py <= pyE3 + 4 && py + 4 >= pyE3))
        {
            if (hearts > 0)
            {
                hearts--;
                eraseE3();
                pyE3 = 3;
                printEnemy3();
            }
        }
        if ((px <= pxE4 + 5 && px + 5 >= pxE4) && (py <= pyE4 + 4 && py + 4 >= pyE4))
        {
            if (hearts > 0)
            {
                hearts--;
                eraseE4();
                pyE4 = 3;
                printEnemy4();
            }
        }
    }
                                        // Color Function
    void Color(int color)
    {
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), color);
    }
    void gotoxy(int x, int y)
    {
        COORD c;
        c.X = x;
        c.Y = y;

        SetConsoleCursorPosition(
            GetStdHandle(STD_OUTPUT_HANDLE), c);
        return;
    }
    char getCharAtxy(short int x, short int y)
    {
        CHAR_INFO ci;
        COORD xy = {0, 0};
        SMALL_RECT rect = {x, y, x, y};
        COORD coordBufSize;
        coordBufSize.X = 1;
        coordBufSize.Y = 1;
        return ReadConsoleOutput(GetStdHandle(STD_OUTPUT_HANDLE), &ci,
    coordBufSize, xy, &rect) ? ci.Char.AsciiChar : ' ';
    }
                                    // Show Score
    void show_score()
    {
        gotoxy(113, 4);
        Color(03);
        cout << "Score: ";
        Color(06);
        cout << score;
    }
                                    // Show Score
    void show_health()
    {
        gotoxy(113, 6);
        Color(04);
        cout << "Hearts: ";
        Color(02);
        cout << hearts;
    }
                                    // Show Level
    void show_level()
    {
        gotoxy(113, 8);
```

```
        Color(02);

        cout << "Level: ";

        Color(03);

        cout << Level;

    }

                                    // Lane Time

    void time_in_lane()

    {

        if (px < 23 && px > 2)

        {

            lane_time--;

        }

        else if (px > 26 && px < 47)

        {

            lane_time--;

        }

        else if (px > 54 && px < 77)

        {

            lane_time--;

        }

        else if (px > 80 && px < 105)

        {

            lane_time--;

        }

    }

                                    // Show Lane time

    void show_lane_time()
    {
        gotoxy(113, 10);
        Color(03);
        cout << "Lane Time: ";
        Color(06);
        cout << lane_time;
    }
    //                                      // Check Health
    void check_health()
    {
        if (hearts <= 0)
        {
            Game_Over();
        }
    }
                                            // Hide cursor Function
    void hideAndVisibleCursor(bool isShow)
    {
        HANDLE consoleHandle = GetStdHandle(STD_OUTPUT_HANDLE);
        CONSOLE_CURSOR_INFO info;
        info.dwSize = 100;
        info.bVisible = isShow;
        SetConsoleCursorInfo(consoleHandle, &info);

}
```