# Firing and shooting game

A logo of a university of engineering and technology

Description automatically generated

# Session 2023 – 2027

# Submitted by:

Salman Naseem 2023-CS-78

# Supervised by:

Dr. Muhammad Awais Hassan

# Course:

CSC-102 Programming Fundamentals

Department of Computer Science

# University of Engineering and Technology

# Lahore Pakistan

**Story of game:**

This is a shooting game in which player can fire in three directions . Player can fire up, left and right. Enemies generate randomly .The direction of movement of enemies is vertical. In case of killing one enemy, there would be 5 score. There will generate energy packets after some time randomly. When player got it ,its power boost up and health increases .The energy packet will erase after some time and generated again and again. If enemy collides with player, the health of player decreases and game over when health decreases to zero.

**Game Characters Description:**

**Player (Spacecraft):**

* Controlled by the player using directional keys.
* Equipped with the ability to shoot projectiles by pressing the space button , less than button and greater than symbol button
* Limited health of player

**Enemies:**

* Enemies generate randomly.
* Maximum three enemies can generate at one time
* Direction of enemies is vertically.

**Game Objects Description:**

**Rules & Interactions:**

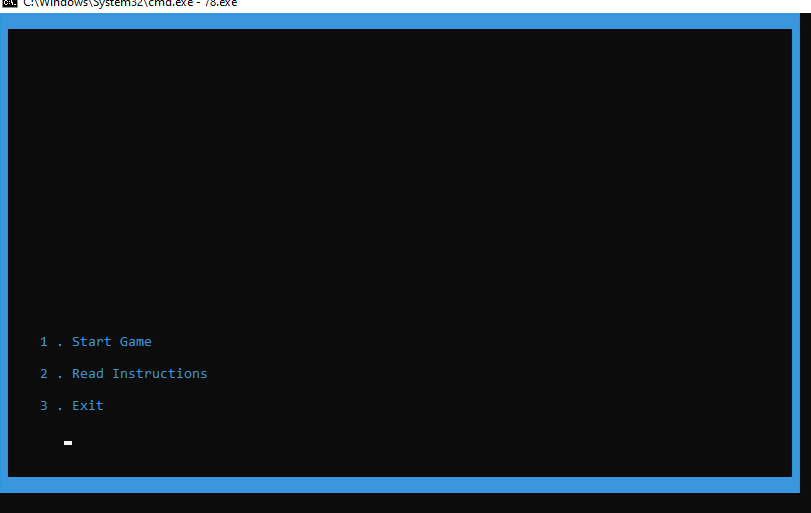
* The player should avoiding collisions with both the enemies.
* Colliding with an enemy sends the player's position back to the starting position.
* The player’s health decreases when collide with enemy.
* Game over when health become zero.

**Goal of the Game:**

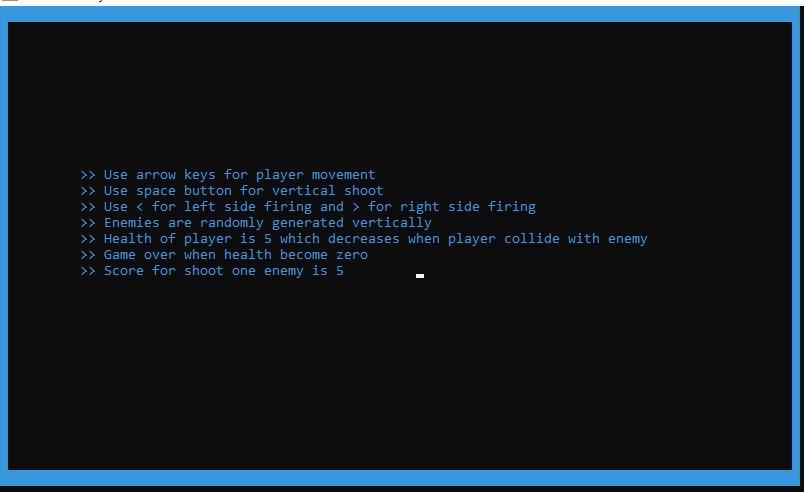
The Simple goal of the game is to score more and more by killing enemies.

**Wireframes of the Game:**

**MAIN PAGE**



**INSTRUCTIONS PAGE**



**Game View:**

****

**Code:**

**#include <iostream>**

**#include <sstream>**

**#include <conio.h>**

**#include <windows.h>**

**#include <cmath>**

**#include <cstdlib>**

**#include <ctime>**

**using namespace std;**

**void gotoxy(int x, int y);**

**char getCharAtxy(short int x, short int y);**

**void red();**

**void blue();**

**void green();**

**void yellow();**

**string startMenu();**

**void startgame();**

**void instructions(); // instructios before starting game**

**void printPlayer(int x, int y);**

**void erasePlayer();**

**void moveLeft();**

**void moveRight();**

**void moveUp();**

**void moveDown();**

**void shoot();**

**void shootLeft();**

**void shootRight();**

**void moveBullet();**

**void printBullet(int x, int y);**

**void eraseBullet(int x, int y);**

**void BulletInActive(int index);**

**void BulletRightInActive(int index);**

**void BulletLeftInActive(int index);**

**void generateRandomEnemies();**

**void drawEnemy();**

**void eraseEnemy();**

**void moveEnemy();**

**void collisionWithEnemy();**

**void handlePlayerHit();**

**void upBulletCollision(); // bullet generated by space button**

**void LeftBulletCollision();**

**void rightBulletCollision();**

**void Maze();**

**void playerHealth();**

**void energyPacket(int energyX, int energyY); // energy to increase player health**

**void printPacket(int energyX, int energyY);**

**void erasePacket(int energyX, int energyY);**

**void chkCollisionwithPacket(int energyX, int energyY);**

**bool isCollionWithPacket = false;**

**bool increaseHealth = false;**

**void printscore();**

**string option;**

**int playerX = 50;**

**int playerY = 25;**

**int Ex = 30, Ey = 2;**

**int maxEnemies = 3; // number of enemies as generated at one time**

**int enemyX[3]; // X coordinates of enemies**

**int enemyY[3]; // Y coordinates of enemies**

**int maxHealth = 3;**

**int palletTimer = 0; // generate energy packet timer**

**int palletEraseTimer = 0; // timer for erase pallet**

**bool isPrintpallet = true; // chk is it a time to print pallet**

**int energyX = 40, energyY = 10; // coordinates of pallet**

**int score = 0;**

**int bulletcount = 0; // Number of bullets shoted by Player by sapce key**

**int bulletRightcount = 0; // by right key**

**int bulletLeftcount = 0; // by left key**

**int bulletx[1000]; // X axis of Player bullet shooted by space button**

**int bullety[1000]; // Y axis of player bullet shooted by space button**

**int bulletRx[2000];**

**int bulletLx[2000];**

**int bulletLy[2000];**

**int bulletRy[2000];**

**bool isBulletActive[1000]; // chk whether bullet is shooted by player**

**bool isBulletLeftActive[2000];**

**bool isBulletRightActive[2000];**

**main()**

**{**

**while (true)**

**{**

**string startOption = startMenu();**

**if (startOption == "1")**

**{**

**startgame();**

**break;**

**}**

**else if (startOption == "2")**

**{**

**instructions();**

**}**

**else if (startOption == "3")**

**{**

**return 0;**

**}**

**}**

**}**

**void gotoxy(int x, int y)**

**{**

**COORD coordinates;**

**coordinates.X = x;**

**coordinates.Y = y;**

**SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coordinates);**

**}**

**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 : ' ';**

**}**

**void red()**

**{**

**SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 4);**

**}**

**void green()**

**{**

**SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 10);**

**}**

**void yellow()**

**{**

**SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 6);**

**}**

**void blue()**

**{**

**SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 3);**

**}**

**string startMenu()**

**{**

**while (option != "1" || option != "2" || option != "3")**

**{**

**system("cls");**

**Maze();**

**gotoxy(5, 20);**

**cout << "1 . Start Game";**

**gotoxy(5, 22);**

**cout << "2 . Read Instructions";**

**gotoxy(5, 24);**

**cout << "3 . Exit";**

**gotoxy(8, 26);**

**getline(cin, option);**

**return option;**

**}**

**}**

**void Maze()**

**{**

**string maze[30][100];**

**for (int row = 0; row < 30; row++)**

**{**

**for (int col = 0; col < 100; col++)**

**{**

**if (row == 0 || row == 29 || col == 0 || col == 99)**

**{**

**maze[row][col] = char(219);**

**}**

**else**

**{**

**maze[row][col] = ' ';**

**}**

**blue();**

**cout << maze[row][col];**

**}**

**cout << endl;**

**}**

**}**

**void instructions()**

**{**

**while (true)**

**{**

**system("cls");**

**Maze();**

**gotoxy(10, 10);**

**cout << ">> Use arrow keys for player movement";**

**gotoxy(10, 11);**

**cout << ">> Use space button for vertical shoot";**

**gotoxy(10, 12);**

**cout << ">> Use < for left side firing and > for right side firing";**

**gotoxy(10, 13);**

**cout << ">> Enemies are randomly generated vertically ";**

**gotoxy(10, 14);**

**cout << ">> Health of player is 5 which decreases when player collide with enemy";**

**gotoxy(10, 15);**

**cout << ">> Game over when health become zero";**

**gotoxy(10, 16);**

**cout << ">> Score for shoot one enemy is 5 ";**

**getch();**

**break;**

**}**

**}**

**void startgame()**

**{**

**system("cls");**

**bool isGameover = false;**

**Maze();**

**printPlayer(playerX, playerY);**

**generateRandomEnemies();**

**while (!isGameover)**

**{**

**srand(time(nullptr));**

**drawEnemy();**

**playerHealth(); // print player health outside the maze**

**if (\_kbhit())**

**{**

**if (GetAsyncKeyState(VK\_LEFT))**

**{**

**moveLeft();**

**}**

**if (GetAsyncKeyState(VK\_RIGHT))**

**{**

**moveRight();**

**}**

**if (GetAsyncKeyState(VK\_UP))**

**{**

**moveUp();**

**}**

**if (GetAsyncKeyState(VK\_DOWN))**

**{**

**moveDown();**

**}**

**if (GetAsyncKeyState(VK\_SPACE))**

**{**

**shoot();**

**}**

**else if (GetAsyncKeyState(VK\_OEM\_COMMA)) // less than symbol for left shoot**

**{**

**shootLeft();**

**}**

**else if (GetAsyncKeyState(VK\_OEM\_PERIOD)) // greater than symbol for right shoot**

**{**

**shootRight();**

**}**

**else if (GetAsyncKeyState(VK\_ESCAPE))**

**{**

**break;**

**}**

**}**

**moveEnemy();**

**Sleep(20);**

**upBulletCollision();**

**rightBulletCollision();**

**LeftBulletCollision();**

**moveBullet();**

**collisionWithEnemy();**

**printscore();**

**if (maxHealth <= 0)**

**{**

**isGameover = true;**

**}**

**energyPacket(energyX, energyY); // generate energy packet to boost player health**

**chkCollisionwithPacket(energyX, energyY);**

**erasePacket(energyX, energyY);**

**palletTimer++;**

**palletEraseTimer++;**

**}**

**system("cls");**

**Maze();**

**gotoxy(20, 10);**

**cout << "Game Over ";**

**gotoxy(20, 12);**

**cout << "Your Score is : " << score;**

**gotoxy(5, 33);**

**getch();**

**}**

**void playerHealth()**

**{**

**blue();**

**gotoxy(112, 8);**

**cout << "FISH HEALTH :" << endl;**

**gotoxy(112, 10);**

**cout << " ";**

**yellow();**

**gotoxy(112, 10);**

**cout << maxHealth;**

**}**

**void printscore() // print score on right side of maze**

**{**

**blue();**

**gotoxy(112, 15);**

**cout << "Score ";**

**gotoxy(112, 17);**

**cout << score;**

**}**

**void printPlayer(int x, int y)**

**{**

**green();**

**char fish[4][4] = {{' ', '/', '\\', ' '},**

**{'<', '@', ')', '>'},**

**{'\\', '\_', '/', ' '},**

**{'/', ' ', '\\', ' '}};**

**int yAxis = playerY;**

**for (int row = 0; row < 4; row++)**

**{**

**gotoxy(playerX, yAxis);**

**for (int col = 0; col < 4; col++)**

**{**

**cout << fish[row][col];**

**}**

**yAxis++;**

**}**

**}**

**void erasePlayer()**

**{**

**gotoxy(playerX, playerY);**

**for (int index = 0; index < 4; index++)**

**{**

**cout << " ";**

**}**

**gotoxy(playerX, playerY + 1);**

**for (int i = 0; i < 4; i++)**

**{**

**cout << " ";**

**}**

**gotoxy(playerX, playerY + 2);**

**for (int index = 0; index < 4; index++)**

**{**

**cout << " ";**

**}**

**gotoxy(playerX, playerY + 3);**

**for (int index = 0; index < 4; index++)**

**{**

**cout << " ";**

**}**

**}**

**void moveLeft()**

**{**

**char nextLocation1 = getCharAtxy(playerX - 1, playerY);**

**char nextLocation2 = getCharAtxy(playerX - 1, playerY + 1);**

**char nextLocation3 = getCharAtxy(playerX - 1, playerY + 2);**

**char nextLocation4 = getCharAtxy(playerX - 1, playerY + 3);**

**if (nextLocation1 == ' ' && nextLocation2 == ' ' && nextLocation3 == ' ' && nextLocation4 == ' ')**

**{**

**erasePlayer();**

**playerX = playerX - 1;**

**printPlayer(playerX, playerY);**

**}**

**}**

**void moveRight()**

**{**

**char nextLocation1 = getCharAtxy(playerX + 4, playerY);**

**char nextLocation2 = getCharAtxy(playerX + 4, playerY + 1);**

**char nextLocation3 = getCharAtxy(playerX + 4, playerY + 2);**

**char nextLocation4 = getCharAtxy(playerX + 4, playerY + 3);**

**if (nextLocation1 == ' ' && nextLocation2 == ' ' && nextLocation3 == ' ' && nextLocation4 == ' ')**

**{**

**erasePlayer();**

**playerX = playerX + 1;**

**printPlayer(playerX, playerY);**

**}**

**}**

**void moveUp()**

**{**

**char nextLocation1 = getCharAtxy(playerX, playerY - 1);**

**char nextLocation2 = getCharAtxy(playerX + 1, playerY - 1);**

**char nextLocation3 = getCharAtxy(playerX + 2, playerY - 1);**

**char nextLocation4 = getCharAtxy(playerX + 3, playerY - 1);**

**if (nextLocation1 == ' ' && nextLocation2 == ' ' && nextLocation3 == ' ' && nextLocation4 == ' ')**

**{**

**erasePlayer();**

**playerY = playerY - 1;**

**printPlayer(playerX, playerY);**

**}**

**}**

**void moveDown()**

**{**

**char nextLocation1 = getCharAtxy(playerX, playerY + 4);**

**char nextLocation2 = getCharAtxy(playerX + 1, playerY + 4);**

**char nextLocation3 = getCharAtxy(playerX + 2, playerY + 4);**

**char nextLocation4 = getCharAtxy(playerX + 3, playerY + 4);**

**if (nextLocation1 == ' ' && nextLocation2 == ' ' && nextLocation3 == ' ' && nextLocation4 == ' ')**

**{**

**erasePlayer();**

**playerY = playerY + 1;**

**printPlayer(playerX, playerY);**

**}**

**}**

**void shoot()**

**{**

**char nextLocation1 = getCharAtxy(playerX, playerY - 1);**

**if (nextLocation1 == ' ')**

**{**

**bulletx[bulletcount] = playerX + 2;**

**bullety[bulletcount] = playerY - 1;**

**isBulletActive[bulletcount] = true;**

**isBulletLeftActive[bulletcount] = false;**

**isBulletRightActive[bulletcount] = false;**

**gotoxy(bulletx[bulletcount], bullety[bulletcount]);**

**blue();**

**cout << "o";**

**bulletcount++;**

**}**

**}**

**void shootLeft()**

**{**

**int shootPx = playerX - 5, shootPy = playerY + 2;**

**char nextLocation1 = getCharAtxy(shootPx, shootPy);**

**if (nextLocation1 == ' ')**

**{**

**bulletLx[bulletLeftcount] = shootPx + 2;**

**bulletLy[bulletLeftcount] = shootPy - 1;**

**isBulletLeftActive[bulletLeftcount] = true;**

**isBulletActive[bulletcount] = false;**

**isBulletRightActive[bulletRightcount] = false;**

**gotoxy(bulletLx[bulletLeftcount], bulletLy[bulletLeftcount]);**

**blue();**

**cout << "o";**

**bulletLeftcount++;**

**}**

**}**

**void shootRight()**

**{**

**int shootPRx = playerX + 3, shootPRy = playerY + 2;**

**char nextLocation1 = getCharAtxy(shootPRx, shootPRy);**

**if (nextLocation1 == ' ')**

**{**

**bulletRx[bulletRightcount] = shootPRx + 2;**

**bulletRy[bulletRightcount] = shootPRy - 1;**

**isBulletRightActive[bulletRightcount] = true;**

**isBulletActive[bulletcount] = false;**

**isBulletLeftActive[bulletLeftcount] = false;**

**gotoxy(bulletRx[bulletRightcount], bulletRy[bulletRightcount]);**

**blue();**

**cout << "o";**

**bulletRightcount++;**

**}**

**}**

**void moveBullet()**

**{**

**for (int i = 0; i < bulletcount; i++)**

**{**

**if (isBulletActive[i] == true)**

**{**

**char next = getCharAtxy(bulletx[i], bullety[i] - 1);**

**if (next != ' ')**

**{**

**eraseBullet(bulletx[i], bullety[i]);**

**BulletInActive(i);**

**}**

**else**

**{**

**eraseBullet(bulletx[i], bullety[i]);**

**bullety[i] = bullety[i] - 1;**

**printBullet(bulletx[i], bullety[i]);**

**}**

**}**

**}**

**for (int i = 0; i < bulletLeftcount; i++)**

**{**

**if (isBulletLeftActive[i] == true)**

**{**

**char next = getCharAtxy(bulletLx[i] - 1, bulletLy[i]);**

**if (next != ' ')**

**{**

**eraseBullet(bulletLx[i], bulletLy[i]);**

**BulletLeftInActive(i);**

**}**

**else**

**{**

**eraseBullet(bulletLx[i], bulletLy[i]);**

**bulletLx[i] = bulletLx[i] - 1;**

**printBullet(bulletLx[i], bulletLy[i]);**

**}**

**}**

**}**

**for (int i = 0; i < bulletRightcount; i++)**

**{**

**if (isBulletRightActive[i] == true)**

**{**

**char next = getCharAtxy(bulletRx[i] + 1, bulletRy[i]);**

**if (next != ' ')**

**{**

**eraseBullet(bulletRx[i], bulletRy[i]);**

**BulletRightInActive(i);**

**}**

**else**

**{**

**eraseBullet(bulletRx[i], bulletRy[i]);**

**bulletRx[i] = bulletRx[i] + 1;**

**printBullet(bulletRx[i], bulletRy[i]);**

**}**

**}**

**}**

**}**

**void printBullet(int x, int y)**

**{**

**blue();**

**gotoxy(x, y);**

**cout << "o";**

**}**

**void eraseBullet(int x, int y)**

**{**

**gotoxy(x, y);**

**cout << " ";**

**}**

**void BulletInActive(int index)**

**{**

**isBulletActive[index] = false;**

**}**

**void BulletRightInActive(int index)**

**{**

**isBulletRightActive[index] = false;**

**}**

**void BulletLeftInActive(int index)**

**{**

**isBulletLeftActive[index] = false;**

**}**

**void upBulletCollision()**

**{**

**for (int i = 0; i < bulletcount; i++)**

**{**

**if (isBulletActive[i] == true)**

**{**

**for (int j = 0; j < maxEnemies; ++j)**

**{**

**if ((bulletx[i] >= enemyX[j] && bulletx[i] <= enemyX[j] + 6) &&**

**(bullety[i] == enemyY[j] + 2 || bullety[i] == enemyY[j] + 1))**

**{**

**score += 5;**

**eraseEnemy();**

**enemyY[j] = 2;**

**generateRandomEnemies();**

**eraseBullet(bulletx[i], bullety[i]);**

**BulletInActive(i);**

**}**

**}**

**}**

**}**

**}**

**void LeftBulletCollision()**

**{**

**for (int i = 0; i < bulletLeftcount; i++)**

**{**

**if (isBulletLeftActive[i] == true)**

**{**

**for (int j = 0; j < maxEnemies; ++j)**

**{**

**if ((bulletLx[i] == enemyX[i] + 6) && (bulletLy[i] == enemyY[i]) || (bulletLx[i] + 1 == enemyX[i] && bulletLy[i] == enemyY[i] + 1) || (bulletLx[i] == enemyX[i] + 6) && (bulletLy[i] == enemyY[i] + 2))**

**{**

**score += 5;**

**eraseEnemy();**

**enemyY[j] = 2;**

**generateRandomEnemies();**

**eraseBullet(bulletLx[i], bulletLy[i]);**

**BulletLeftInActive(i);**

**}**

**}**

**}**

**}**

**}**

**void rightBulletCollision()**

**{**

**for (int i = 0; i < bulletRightcount; i++)**

**{**

**if (isBulletRightActive[i] == true)**

**{**

**for (int j = 0; j < maxEnemies; ++j)**

**{**

**if ((bulletRx[i] >= enemyX[j] && bulletRx[i] <= enemyX[j] + 6) &&**

**(bulletRy[i] == enemyY[j] + 2 || bulletRy[i] == enemyY[j] + 1))**

**{**

**score += 5;**

**eraseEnemy();**

**enemyY[j] = 2;**

**generateRandomEnemies();**

**eraseBullet(bulletRx[i], bulletRy[i]);**

**BulletRightInActive(i);**

**}**

**}**

**}**

**}**

**}**

**void generateRandomEnemies()**

**{**

**for (int i = 0; i < maxEnemies; ++i)**

**{**

**if (rand() % 100 < 92 && rand() % 100 > 8)**

**{**

**enemyX[i] = abs((rand() % 100) - 8);**

**enemyY[i] = 2;**

**}**

**}**

**}**

**void drawEnemy()**

**{**

**char drawEnemy[2][7] = {**

**{'\_', '\\', '(', '\_', ')', '/', '\_'},**

**{' ', '/', '(', 'O', ')', '\\', ' '}};**

**yellow();**

**for (int i = 0; i < maxEnemies; ++i)**

**{**

**gotoxy(enemyX[i], enemyY[i]);**

**for (int j = 0; j < 2; ++j)**

**{**

**for (int k = 0; k < 7; ++k)**

**{**

**cout << drawEnemy[j][k];**

**}**

**gotoxy(enemyX[i], enemyY[i] + 1);**

**}**

**}**

**}**

**void eraseEnemy()**

**{**

**for (int i = 0; i < maxEnemies; ++i)**

**{**

**gotoxy(enemyX[i], enemyY[i]);**

**cout << " ";**

**gotoxy(enemyX[i], enemyY[i] + 1);**

**cout << " ";**

**}**

**}**

**void moveEnemy()**

**{**

**for (int i = 0; i < maxEnemies; ++i)**

**{**

**char nextLocation1 = getCharAtxy(enemyX[i], enemyY[i] + 2);**

**char nextLocation2 = getCharAtxy(enemyX[i] + 1, enemyY[i] + 2);**

**char nextLocation3 = getCharAtxy(enemyX[i] + 2, enemyY[i] + 2);**

**char nextLocation4 = getCharAtxy(enemyX[i] + 3, enemyY[i] + 2);**

**char nextLocation5 = getCharAtxy(enemyX[i] + 4, enemyY[i] + 2);**

**char nextLocation6 = getCharAtxy(enemyX[i] + 5, enemyY[i] + 2);**

**char nextLocation7 = getCharAtxy(enemyX[i] + 6, enemyY[i] + 2);**

**if (nextLocation1 == ' ' && nextLocation2 == ' ' && nextLocation3 == ' ' && nextLocation4 == ' ' && nextLocation5 == ' ' && nextLocation6 == ' ' && nextLocation7 == ' ')**

**{**

**eraseEnemy();**

**enemyY[i] = enemyY[i] + 1;**

**}**

**else if (nextLocation1 != ' ' || nextLocation2 != ' ' || nextLocation3 != ' ' || nextLocation4 != ' ' || nextLocation5 != ' ' || nextLocation6 != ' ' || nextLocation7 != ' ')**

**{**

**eraseEnemy();**

**enemyY[i] = 2;**

**generateRandomEnemies();**

**}**

**}**

**drawEnemy();**

**}**

**void collisionWithEnemy() // player collision with enemy**

**{**

**if (maxHealth > 0)**

**{**

**for (int i = 0; i < maxEnemies; ++i)**

**{**

**// Bottom collision**

**if (playerY + 3 >= enemyY[i] && playerY <= enemyY[i] && playerX + 3 >= enemyX[i] && playerX <= enemyX[i] + 6)**

**{**

**handlePlayerHit();**

**}**

**if ((enemyX[i] == playerX && enemyY[i] + 2 == playerY) || (enemyX[i] + 1 == playerX && enemyY[i] + 2 == playerY) || (enemyX[i] + 2 == playerX && enemyY[i] + 2 == playerY) || (enemyX[i] + 3 == playerX && enemyY[i] + 2 == playerY) || (enemyX[i] + 4 == playerX && enemyY[i] + 2 == playerY) || (enemyX[i] + 5 == playerX && enemyY[i] + 2 == playerY) || (enemyX[i] + 6 == playerX && enemyY[i] + 2 == playerY) || (enemyX[i] + 7 == playerX && enemyY[i] + 2 == playerY)) // bottom**

**{**

**handlePlayerHit();**

**}**

**// Right side collision**

**if (playerX + 4 >= enemyX[i] && playerX <= enemyX[i] + 5 && playerY <= enemyY[i] + 1 && playerY >= enemyY[i] - 4)**

**{**

**handlePlayerHit();**

**}**

**// Left side collision**

**if ((playerX <= enemyX[i] + 6 && playerX >= enemyX[i] + 1 && playerY <= enemyY[i] + 1 && playerY >= enemyY[i] - 4) ||**

**(playerX + 3 == enemyX[i] - 1 && playerY + 1 == enemyY[i] + 2))**

**{**

**handlePlayerHit();**

**}**

**}**

**}**

**}**

**void handlePlayerHit()**

**{**

**maxHealth--;**

**int prevPlayerX = playerX;**

**int prevPlayerY = playerY;**

**erasePlayer();**

**playerX = 8;**

**playerY = 25;**

**printPlayer(playerX, playerY);**

**gotoxy(prevPlayerX, prevPlayerY);**

**erasePlayer();**

**printPlayer(playerX, playerY);**

**}**

**void energyPacket(int energyX, int energyY)**

**{**

**if (palletTimer >= 200 && isPrintpallet)**

**{**

**printPacket(energyX, energyY);**

**isCollionWithPacket = true;**

**}**

**}**

**void printPacket(int energyX, int energyY)**

**{**

**blue();**

**gotoxy(energyX, energyY);**

**cout << "E";**

**palletTimer = 0;**

**palletEraseTimer = 0;**

**isPrintpallet = false;**

**}**

**void erasePacket(int energyX, int energyY)**

**{**

**if (palletEraseTimer >= 50 && !isPrintpallet)**

**{**

**gotoxy(energyX, energyY);**

**cout << " ";**

**palletEraseTimer = 0;**

**palletTimer = 0;**

**isPrintpallet = true;**

**isCollionWithPacket = false;**

**}**

**}**

**void chkCollisionwithPacket(int energyX, int energyY)**

**{**

**if (isCollionWithPacket)**

**{**

**if ((playerX >= energyX && playerX <= energyX + 1) && (playerY + 1 == energyY || playerY == energyY) || (playerX == energyX - 1 && playerY == energyY + 1) || (playerX + 1 == energyX - 1 && playerY == energyY + 1) || (playerX + 2 == energyX - 1 && playerY == energyY + 1) || (playerX + 3 == energyX - 1 && playerY == energyY + 1) || (playerX + 4 == energyX - 1 && playerY == energyY + 1))**

**{**

**increaseHealth = true;**

**gotoxy(energyX, energyY);**

**cout << " ";**

**palletEraseTimer = 0;**

**palletTimer = 0;**

**isCollionWithPacket = false;**

**}**

**}**

**if (increaseHealth)**

**{**

**maxHealth++;**

**increaseHealth = false;**

**}**

**}**