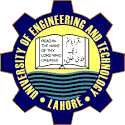
**Freedom Fodge**



**Session 2023 – 2027**

**Submitted by:**

ESA IRFAN 2023-CS-104

**Supervised by:**

Mam Maida Shahid

**Course:**

CSC-102 Programming Fundamental

**Department of Computer Science**

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

[**1.1 About Freedom Fodge** 3](#_Toc153960903)

[**1.2 Description about characters:** 3](#_Toc153960904)

[**1.2.1 Player (Jerry):** 3](#_Toc153960905)

[**1.2.2 Enemies (Helicopter):** 3](#_Toc153960906)

[**1.2.3 Fires:** 3](#_Toc153960907)

[**1.2.4 Maze and Obstacles:** 3](#_Toc153960908)

[**1.2.5 Teleportation:** 3](#_Toc153960909)

[**1.3 Objectives and Interactions:** 3](#_Toc153960910)

[**1.3.1 Collect Coins:** 3](#_Toc153960911)

[**1.3.2 Obstacles:** 3](#_Toc153960912)

[**1.3.3 Teleporters:** 3](#_Toc153960913)

[**1.4 Goal:** 4](#_Toc153960914)

[**1.5 Data structure** 4](#_Toc153960915)

[**1.6 Function Prototypes** 5](#_Toc153960916)

[**1.7 Wireframes:** 6](#_Toc153960917)

[**1.7.1 Main Page** 6](#_Toc153960918)

[**1.7.2 Game Menu** 6](#_Toc153960919)

[**1.7.2.1 Display 1** 6](#_Toc153960920)

[**1.7.2.2 Display 2** 7](#_Toc153960921)

[**1.7.2.3 Display 3** 7](#_Toc153960922)

[**1.7.2.4 Display 4** 8](#_Toc153960923)

[**1.7.3 Winning Menu** 8](#_Toc153960924)

[**1.8 Code:** 9](#_Toc153960925)

[Figure 1 Header 6](#_Toc153907559)

[Figure 2 1st side of Maze 6](#_Toc153907560)

[Figure 3 2nd side of Maze 7](#_Toc153907561)

[Figure 4 3rd side of Maze 7](#_Toc153907562)

[Figure 5 4th side of Maze 8](#_Toc153907563)

# **1.1 About Freedom Fodge**

"Dive into the shadows of confinement with **'Freedom Fodge'** an enthralling prison escape game that pushes the boundaries of your intellect and resourcefulness. Unjustly confined within the unforgiving walls of a mysterious maximum-security facility, you must unravel the enigma surrounding your imprisonment and orchestrate a daring escape.

# **1.2 Description about characters:**

## **1.2.1 Player (Jerry):**

Jerry can move in four directions: up, down, left, and right. The player's score increases when Jerry collects coins ('o') or crosses specific obstacles ('0').Jerry's health decreases when it collides with certain obstacles like '#' or '\*'.Jerry can teleport to different locations marked by 'T' on the maze.

## **1.2.2 Enemies (Helicopter):**

There are total 10 enemies in my game. The enemy moves either left or right based on Jerry's position. Random movement if Jerry is far. It act as a obstacle and useas a health drainage.

## **1.2.3 Fires:**

Fires move downward within the maze, simulating a projectile. Fires collide with maze obstacles and may affect Jerry's health.

## **1.2.4 Maze and Obstacles:**

The maze is represented by a 2D array of characters loaded from an external file Obstacles include walls ('#') and special obstacles ('0') that impact the player's score and health.

## **1.2.5 Teleportation:**

Jerry can teleoperate to specific location marked by T. Teleportation occurs every fourth time Jerry encounters a teleportation point.

# **1.3 Objectives and Interactions:**

## **1.3.1 Collect Coins:**

Move Jerry to collect 'o' (coins) to increase the player's score.

## **1.3.2 Obstacles:**

Avoid obstacles '#' to prevent health reduction.

## **1.3.3 Teleporters:**

Use teleporters ('T') strategically to move Jerry to different locations.

# **1.4 Goal:**

The goal of the game "Freedom Fodge" is to navigate Jerry through the maze, collecting coins to increase your score, collecting cheese to maintain or increase Jerry's health, and avoiding obstacles and enemies. The specific objectives may vary depending on the game's design, but generally, you aim to accomplish tasks within the maze, potentially reaching a designated goal or completing a series of levels.

# **1.5 Data structure**

* const int rows = 58;
* const int cols = 220;
* char maze[rows][cols];
* int pX = 45, pY = 15;
* int obsatclesCounter = 0;
* int playerScore = 0;
* char playerInput = ' ';
* int coinsCounter = 0;
* int cheeseHealth = 12;
* int playersHealth = 100;
* int teleporateCounter = 0;
* const int maxFire = 100000; // Maximum number of fires
* int fireX[maxFire] = {0};
* int fireY[maxFire] = {0};
* int fireCounter = 0;
* int eX3 = 90, eY3 = 47;
* int eX2 = 110, eY2 = 20;
* int eX5 = 80, eY5 = 25;
* int eX1 = 40, eY1 = 40;
* int eX4 = 135, eY4 = 40;
* int eX6 = 150, eY6 = 40;
* int eX7 = 175, eY7 = 60;
* int eX8 = 185, eY8 = 25;
* int eX9 = 210, eY9 = 50;
* int eX10 = 210, eY10 = 13;
* string enemydirection = "left";
* string body[3][5] = {{"-", "-", "^", "-", "-"}, {">", "=", "[", "]", ")"}, {"-", "-", "^", "-", "-"}};
* string body[3][5] = {{"-", "-", "^", "-", "-"}, {"(", "[", "]", "=", "<"}, {"-", "-", "^", "-", "-"}};

# **1.6 Function Prototypes**

* void gotoxy(int x, int y);
* void printMaze(int startX, int startY);
* void setConsoleColor(int color);
* void hideCursor();
* void printJerry(int &Px, int &Py);
* void eraseJerry(int &pX, int &pY);
* void moveLeftJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY);
* void moveUpJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY);
* void moveDownJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY);
* void moveRightJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY);
* char getCharAtxy(short int x, short int y);
* void movementOfJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY);
* void generateFire(int &fireCounter, int fireX[], int fireY[], int maxFire, int pX, int pY, int eX, int eY);
* void randomCoinFormation(int &coinsCounter);
* void playerScoreFormation(int &playerScore);
* void cheeseGraph(int &cheeseHealth);
* void healthGraph(int &playerHealth, int &pX, int &pY);
* void telePoratingPlayer(int &teleporateCounter, int &pX, int &pY);
* void randomObstaclesFormation(int &obsatclesCounter);
* void displayEnemyRight(int x, int y);
* void displayEnemyLeft(int x, int y);
* void eraseEnemy(int x, int y);
* void enemyDirection(int &pX, int &pY, int &eX, int &eY, string &enemydirection);
* void moveFire(int &fireCounter, int fireX[], int fireY[], int maxFire);
* void enemyMovement(int &eX, int &eY, string &enemydirection);
* void print5Characters(string arr[][5], int size, int color, int x, int y);
* void readMaze(const char \*filename, char maze[rows][cols]);

# **1.7 Wireframes:**

## **1.7.1 Main Page**

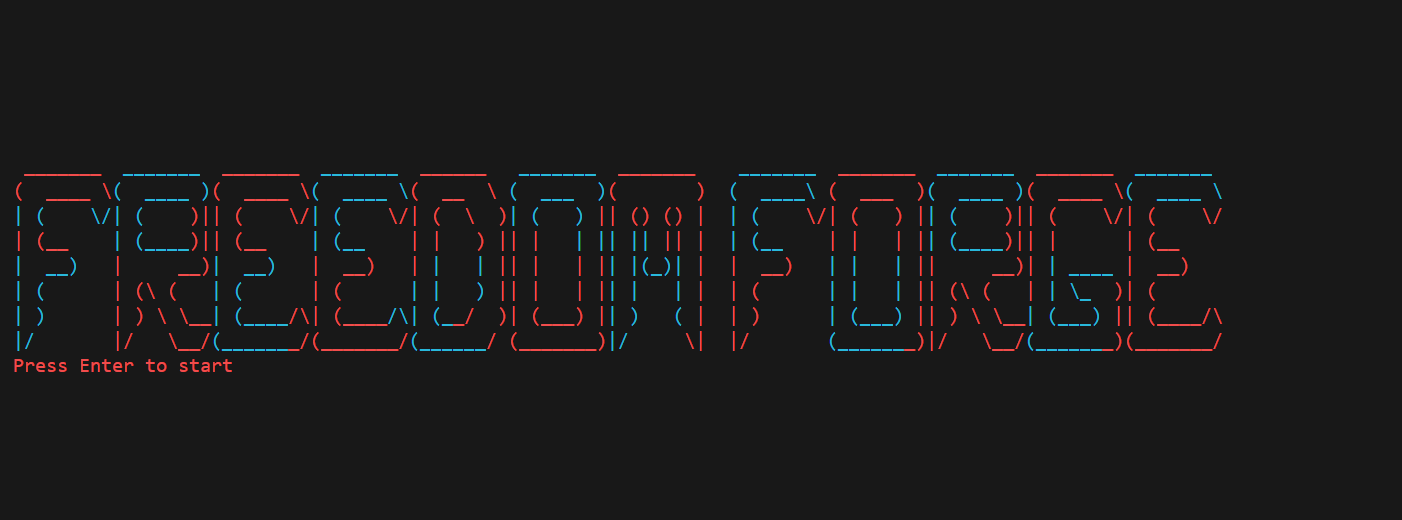


Figure 1 Header

## **1.7.2 Game Menu**

### **1.7.2.1 Display 1**



Figure 2 1st side of Maze

### **1.7.2.2 Display 2**

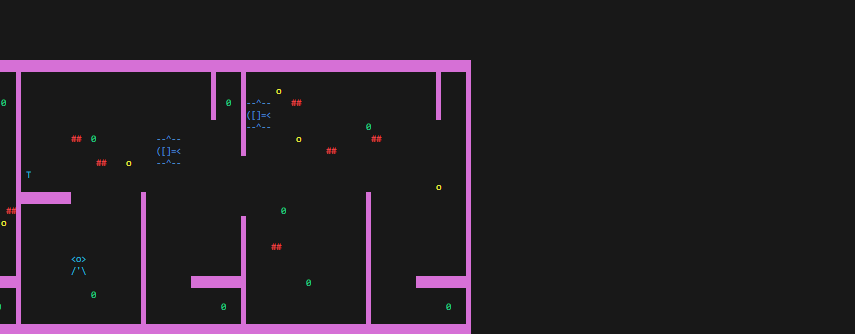


Figure 3 2nd side of Maze

### **1.7.2.3 Display 3**



Figure 4 3rd side of Maze

### **1.7.2.4 Display 4**

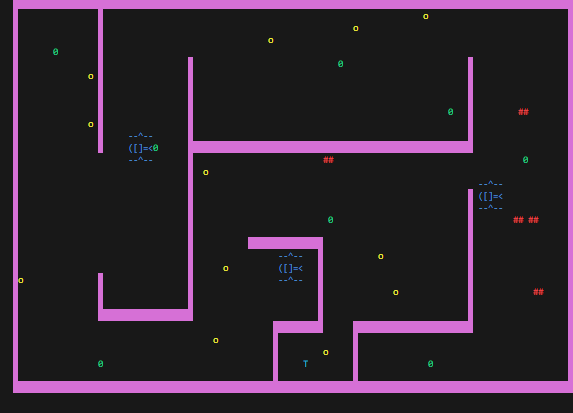


Figure 5 4th side of Maze

## **1.7.3 Winning Menu**

# **1.8 Code:**

|  |
| --- |
| #include <iostream>  #include <windows.h>  #include <conio.h>  #include <unistd.h>  #include <fstream>  using namespace std;  HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);  void gotoxy(int x, int y);  void printMaze(int startX, int startY);  void setConsoleColor(int color);  void hideCursor();  void printJerry(int &Px, int &Py);  void eraseJerry(int &pX, int &pY);  void moveLeftJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY);  void moveUpJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY);  void moveDownJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY);  void moveRightJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY);  char getCharAtxy(short int x, short int y);  void movementOfJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY);  void generateFire(int &fireCounter, int fireX[], int fireY[], int maxFire, int pX, int pY, int eX, int eY);  void randomCoinFormation(int &coinsCounter);  void playerScoreFormation(int &playerScore);  void cheeseGraph(int &cheeseHealth);  void healthGraph(int &playerHealth, int &pX, int &pY);  void telePoratingPlayer(int &teleporateCounter, int &pX, int &pY);  void randomObstaclesFormation(int &obsatclesCounter);  void displayEnemyRight(int x, int y);  void displayEnemyLeft(int x, int y);  void eraseEnemy(int x, int y);  void enemyDirection(int &pX, int &pY, int &eX, int &eY, string &enemydirection);  void moveFire(int &fireCounter, int fireX[], int fireY[], int maxFire);  void enemyMovement(int &eX, int &eY, string &enemydirection);  void print5Characters(string arr[][5], int size, int color, int x, int y);  void playerLivesSystem(int &playerLives, int &playersHealth, int &pX, int &pY);  void overHeader();  void gameHeader();  void loadingFunction();  const int rows = 58;  const int cols = 220;  char maze[rows][cols];  void readMaze(const char \*filename, char maze[rows][cols]);  int main()  {  readMaze("Level1Maze.txt", maze);  hideCursor();  int pX = 45, pY = 15;  int obsatclesCounter = 0;  int playerScore = 0;  char playerInput = ' ';  int coinsCounter = 0;  int cheeseHealth = 30;  int playersHealth = 100;  int teleporateCounter = 0;  int playerLives = 5;  const int maxFire = 100000; // Maximum number of fires  int fireX[maxFire] = {0};  int fireY[maxFire] = {0};  int fireCounter = 0;  int eX3 = 90, eY3 = 47;  int eX2 = 110, eY2 = 20;  int eX5 = 80, eY5 = 25;  int eX1 = 40, eY1 = 40;  int eX4 = 135, eY4 = 40;  int eX6 = 150, eY6 = 40;  int eX7 = 175, eY7 = 60;  int eX8 = 185, eY8 = 25;  int eX9 = 210, eY9 = 50;  int eX10 = 210, eY10 = 13;  string enemydirection = "left";  system("cls");  gameHeader();  loadingFunction();  system("cls");  printMaze(35, 10);  bool valid = false;  while (!valid)  {  movementOfJerry(playerScore, coinsCounter, cheeseHealth, pX, pY);  randomCoinFormation(coinsCounter);  randomObstaclesFormation(obsatclesCounter);  cheeseGraph(cheeseHealth);  healthGraph(playersHealth, pX, pY);  telePoratingPlayer(teleporateCounter, pX, pY);  playerScoreFormation(playerScore);  playerLivesSystem(playerLives, playersHealth, pX, pY);  enemyDirection(pX, pY, eX3, eY3, enemydirection);  enemyDirection(pX, pY, eX2, eY2, enemydirection);  enemyDirection(pX, pY, eX1, eY1, enemydirection);  enemyDirection(pX, pY, eX4, eY4, enemydirection);  enemyDirection(pX, pY, eX5, eY5, enemydirection);  enemyDirection(pX, pY, eX6, eY6, enemydirection);  enemyDirection(pX, pY, eX7, eY7, enemydirection);  enemyDirection(pX, pY, eX8, eY8, enemydirection);  enemyDirection(pX, pY, eX9, eY9, enemydirection);  enemyDirection(pX, pY, eX10, eY10, enemydirection);  generateFire(fireCounter, fireX, fireY, maxFire, pX, pY, eX1, eY1);  generateFire(fireCounter, fireX, fireY, maxFire, pX, pY, eX2, eY2);  generateFire(fireCounter, fireX, fireY, maxFire, pX, pY, eX3, eY3);  generateFire(fireCounter, fireX, fireY, maxFire, pX, pY, eX4, eY4);  generateFire(fireCounter, fireX, fireY, maxFire, pX, pY, eX5, eY5);  generateFire(fireCounter, fireX, fireY, maxFire, pX, pY, eX6, eY6);  generateFire(fireCounter, fireX, fireY, maxFire, pX, pY, eX7, eY7);  generateFire(fireCounter, fireX, fireY, maxFire, pX, pY, eX8, eY8);  generateFire(fireCounter, fireX, fireY, maxFire, pX, pY, eX9, eY9);  generateFire(fireCounter, fireX, fireY, maxFire, pX, pY, eX10, eY10);  moveFire(fireCounter, fireX, fireY, maxFire);  if (playerLives<=0)  {  system("cls");  overHeader();  valid = true;  }    }  }  void gotoxy(int x, int y)  {  COORD coordinates;  coordinates.X = x;  coordinates.Y = y;  SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coordinates);  }  void printMaze(int startX, int startY)  {  char box = 219;  for (int i = 0; i < rows; ++i)  {  gotoxy(startX, startY + i);  for (int j = 0; j < cols; ++j)  {  if (maze[i][j] != '0' && maze[i][j] != 'T')  {  setConsoleColor(7);  setConsoleColor(13);  }  else if (maze[i][j] == 'T')  {  setConsoleColor(7);  setConsoleColor(11);  }  else if (maze[i][j] == '|')  {  setConsoleColor(7);  setConsoleColor(11);  }  else  {  setConsoleColor(7);  setConsoleColor(10);  }  if (maze[i][j] == '@')  {  cout << box;  }  else  {  cout << maze[i][j];  }  }  }  setConsoleColor(7);  }  void setConsoleColor(int color)  {  HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);  SetConsoleTextAttribute(hConsole, color);  }  void printJerry(int &pX, int &pY)  {  setConsoleColor(11);  gotoxy(pX, pY);  cout << "<o>";  gotoxy(pX, pY + 1);  cout << "/'\\";  setConsoleColor(7);  }  void eraseJerry(int &pX, int &pY)  {  gotoxy(pX, pY);  cout << " ";  gotoxy(pX, pY + 1);  cout << " ";  }  void moveLeftJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY)  {  if (getCharAtxy(pX - 1, pY) == ' ' && getCharAtxy(pX - 1, pY + 1) == ' ')  {  eraseJerry(pX, pY);  pX = pX - 1;  printJerry(pX, pY);  }  if (getCharAtxy(pX - 1, pY) == 'o' || getCharAtxy(pX - 1, pY + 1) == 'o')  {  eraseJerry(pX, pY);  pX = pX - 1;  printJerry(pX, pY);  playerScore++;  coinsCounter--;  }  if (getCharAtxy(pX - 1, pY) == '0' || getCharAtxy(pX - 1, pY + 1) == '0')  {  eraseJerry(pX, pY);  pX = pX - 1;  printJerry(pX, pY);  playerScore = playerScore + 5;  cheeseHealth--;  }  }  void moveRightJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY)  {  if (getCharAtxy(pX + 3, pY) == ' ' && getCharAtxy(pX + 3, pY + 1) == ' ')  {  eraseJerry(pX, pY);  pX = pX + 1;  printJerry(pX, pY);  }  if (getCharAtxy(pX + 3, pY) == 'o' || getCharAtxy(pX + 3, pY + 1) == 'o')  {  eraseJerry(pX, pY);  pX = pX + 1;  printJerry(pX, pY);  playerScore++;  coinsCounter--;  }  if (getCharAtxy(pX + 3, pY) == '0' || getCharAtxy(pX + 3, pY + 1) == '0')  {  eraseJerry(pX, pY);  pX = pX + 1;  printJerry(pX, pY);  playerScore = playerScore + 5;  cheeseHealth--;  }  }  void moveUpJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY)  {  if (getCharAtxy(pX, pY - 1) == ' ' && getCharAtxy(pX + 1, pY - 1) == ' ' && getCharAtxy(pX + 2, pY - 1) == ' ')  {  eraseJerry(pX, pY);  pY = pY - 1;  printJerry(pX, pY);  }  if (getCharAtxy(pX, pY - 1) == 'o' || getCharAtxy(pX + 1, pY - 1) == 'o' || getCharAtxy(pX + 2, pY - 1) == 'o')  {  eraseJerry(pX, pY);  pY = pY - 1;  printJerry(pX, pY);  playerScore++;  coinsCounter--;  }  if (getCharAtxy(pX, pY - 1) == '0' || getCharAtxy(pX + 1, pY - 1) == '0' || getCharAtxy(pX + 2, pY - 1) == '0')  {  eraseJerry(pX, pY);  pY = pY - 1;  printJerry(pX, pY);  playerScore = playerScore + 5;  cheeseHealth--;  }  }  void moveDownJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY)  {  if (getCharAtxy(pX, pY + 2) == ' ' && getCharAtxy(pX + 1, pY + 2) == ' ' && getCharAtxy(pX + 2, pY + 2) == ' ')  {  eraseJerry(pX, pY);  pY = pY + 1;  printJerry(pX, pY);  }  if (getCharAtxy(pX, pY + 2) == 'o' || getCharAtxy(pX + 1, pY + 2) == 'o' || getCharAtxy(pX + 2, pY + 2) == 'o')  {  eraseJerry(pX, pY);  pY = pY + 1;  printJerry(pX, pY);  playerScore++;  coinsCounter--;  }  if (getCharAtxy(pX, pY + 2) == '0' || getCharAtxy(pX + 1, pY + 2) == '0' || getCharAtxy(pX + 2, pY + 2) == '0')  {  eraseJerry(pX, pY);  pY = pY + 1;  printJerry(pX, pY);  playerScore = playerScore + 5;  cheeseHealth--;  }  }  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 movementOfJerry(int &playerScore, int &coinsCounter, int &cheeseHealth, int &pX, int &pY)  {  if (GetAsyncKeyState(VK\_LEFT))  {  moveLeftJerry(playerScore, coinsCounter, cheeseHealth, pX, pY);  }  if (GetAsyncKeyState(VK\_RIGHT))  {  moveRightJerry(playerScore, coinsCounter, cheeseHealth, pX, pY);  }  if (GetAsyncKeyState(VK\_DOWN))  {  moveDownJerry(playerScore, coinsCounter, cheeseHealth, pX, pY);  }  if (GetAsyncKeyState(VK\_UP))  {  moveUpJerry(playerScore, coinsCounter, cheeseHealth, pX, pY);  }  }  void randomCoinFormation(int &coinsCounter)  {  if (coinsCounter < 20)  {  int randX, randY;  do  {  randX = rand() % (cols - 2) + 1;  randY = rand() % (rows - 2) + 1;  } while (maze[randY][randX] != ' ');  maze[randY][randX] = 'o';  gotoxy(randX + 35, randY + 10);  setConsoleColor(14);  cout << "o";  setConsoleColor(7);  coinsCounter++;  }  }  void randomObstaclesFormation(int &obsatclesCounter)  {  if (obsatclesCounter < 20)  {  int randX, randY;  do  {  randX = rand() % (cols - 2) + 1;  randY = rand() % (rows - 2) + 1;  } while (maze[randY][randX] != ' ');  maze[randY][randX] = '#';  setConsoleColor(12);  gotoxy(randX + 35, randY + 11);  cout << "##";  setConsoleColor(7);  obsatclesCounter++;  }  }  void playerScoreFormation(int &playerScore)  {  setConsoleColor(12);  gotoxy(10, 20);  cout << "Player Score: ";  setConsoleColor(7);  setConsoleColor(11);  gotoxy(26, 20);  cout << playerScore;  setConsoleColor(7);  }  void cheeseGraph(int &cheeseHealth)  {  gotoxy(10, 24);  cout << " ";  setConsoleColor(14);  gotoxy(10, 24);  cout << "Cheese Remaining " << cheeseHealth;  setConsoleColor(7);  }  void hideCursor()  {  HANDLE consoleHandle = GetStdHandle(STD\_OUTPUT\_HANDLE);  CONSOLE\_CURSOR\_INFO info;  info.dwSize = 100;  info.bVisible = FALSE;  SetConsoleCursorInfo(consoleHandle, &info);  }  void displayEnemyRight(int x, int y)  {  string body[3][5] = {{"-", "-", "^", "-", "-"}, {">", "=", "[", "]", ")"}, {"-", "-", "^", "-", "-"}};  print5Characters(body, 3, 1, x, y);  }  void displayEnemyLeft(int x, int y)  {  string body[3][5] = {{"-", "-", "^", "-", "-"}, {"(", "[", "]", "=", "<"}, {"-", "-", "^", "-", "-"}};  print5Characters(body, 3, 1, x, y);  }  void eraseEnemy(int x, int y)  {  string body[3][5] = {{" ", " ", " ", " ", " "}, {" ", " ", " ", " ", " "}, {" ", " ", " ", " ", " "}};  print5Characters(body, 3, 1, x, y);  }  void enemyDirection(int &pX, int &pY, int &eX, int &eY, string &enemydirection)  {  int horizontalDistance = abs(pX - eX);  int verticalDistance = abs(pY - eY);  if (horizontalDistance <= 5 && verticalDistance <= 5)  {  if (pX > eX)  {  enemydirection = "right";  }  else if (pX < eX)  {  enemydirection = "left";  }  if (pY < eY)  {  enemydirection = "up";  }  else if (pY > eY)  {  enemydirection = "down";  }  }  else  {  if (horizontalDistance > 5)  {  if (pX > eX)  {  enemydirection = "right";  }  else if (pX < eX)  {  enemydirection = "left";  }  }  if (verticalDistance > 5)  {  if (pY < eY)  {  enemydirection = "down";  }  else if (pY > eY)  {  enemydirection = "up";  }  }  int randomValue = rand() % 4;  if (randomValue == 0)  {  enemydirection = "left";  }  else if (randomValue == 1)  {  enemydirection = "right";  }  else if (randomValue == 2)  {  enemydirection = "up";  }  else if (randomValue == 3)  {  enemydirection = "down";  }  }  enemyMovement(eX, eY, enemydirection);  }  void enemyMovement(int &eX, int &eY, string &enemydirection)  {  if (enemydirection == "right")  {  if (getCharAtxy(eX + 5, eY) == ' ' && getCharAtxy(eX + 5, eY + 1) == ' ' && getCharAtxy(eX + 5, eY + 2) == ' ')  {  eraseEnemy(eX, eY);  eX++;  displayEnemyRight(eX, eY);  }  else  {  enemydirection = "left";  }  }  else if (enemydirection == "left")  {  if (getCharAtxy(eX - 1, eY) == ' ' && getCharAtxy(eX - 1, eY + 1) == ' ' && getCharAtxy(eX - 1, eY + 2) == ' ')  {  eraseEnemy(eX, eY);  eX--;  displayEnemyLeft(eX, eY);  }  else  {  enemydirection = "right";  }  }  else if (enemydirection == "up")  {  if (getCharAtxy(eX, eY - 1) == ' ' && getCharAtxy(eX + 1, eY - 1) == ' ' && getCharAtxy(eX + 2, eY - 1) == ' ' && getCharAtxy(eX + 3, eY - 1) == ' ' && getCharAtxy(eX + 4, eY - 1) == ' ')  {  eraseEnemy(eX, eY);  eY--;  displayEnemyLeft(eX, eY);  }  else  {  enemydirection = "down";  }  }  else if (enemydirection == "down")  {  if (getCharAtxy(eX, eY + 3) == ' ' && getCharAtxy(eX + 1, eY + 3) == ' ' && getCharAtxy(eX + 2, eY + 3) == ' ' && getCharAtxy(eX + 3, eY + 3) == ' ' && getCharAtxy(eX + 4, eY + 3) == ' ')  {  eraseEnemy(eX, eY);  eY++;  displayEnemyLeft(eX, eY);  }  else  {  enemydirection = "up";  }  }  }  void print5Characters(string arr[][5], int size, int color, int x, int y)  {  int bx;  for (int i = 0; i < size; i++)  {  bx = x;  setConsoleColor(color);  for (int j = 0; j < 5; j++)  {  gotoxy(bx, y);  cout << arr[i][j];  bx++;  }  setConsoleColor(7);  y++;  }  }  void healthGraph(int &playerHealth, int &pX, int &pY)  {  if (getCharAtxy(pX, pY + 2) == '#' || getCharAtxy(pX + 1, pY + 2) == '#' || getCharAtxy(pX + 2, pY + 2) == '#' || getCharAtxy(pX, pY - 1) == '#' || getCharAtxy(pX + 1, pY - 1) == '#' || getCharAtxy(pX + 2, pY - 1) == '#' || getCharAtxy(pX - 1, pY) == '#' || getCharAtxy(pX - 1, pY + 1) == '#' || getCharAtxy(pX + 3, pY) == '#' || getCharAtxy(pX + 3, pY + 1) == '#')  {  playerHealth--;  }  if (getCharAtxy(pX, pY + 2) == '\*' || getCharAtxy(pX + 1, pY + 2) == '\*' || getCharAtxy(pX + 2, pY + 2) == '\*' || getCharAtxy(pX, pY - 1) == '\*' || getCharAtxy(pX + 1, pY - 1) == '\*' || getCharAtxy(pX + 2, pY - 1) == '\*' || getCharAtxy(pX - 1, pY) == '\*' || getCharAtxy(pX - 1, pY + 1) == '\*' || getCharAtxy(pX + 3, pY) == '\*' || getCharAtxy(pX + 3, pY + 1) == '\*')  {  playerHealth = playerHealth - 5;  }  if (getCharAtxy(pX, pY + 2) == '-' || getCharAtxy(pX + 1, pY + 2) == '-' || getCharAtxy(pX + 2, pY + 2) == '-' || getCharAtxy(pX, pY + 2) == '^' || getCharAtxy(pX + 1, pY + 2) == '^' || getCharAtxy(pX + 2, pY + 2) == '^' ||  getCharAtxy(pX, pY - 1) == '-' || getCharAtxy(pX + 1, pY - 1) == '-' || getCharAtxy(pX + 2, pY - 1) == '-' || getCharAtxy(pX, pY - 1) == '^' || getCharAtxy(pX + 1, pY - 1) == '^' || getCharAtxy(pX + 2, pY - 1) == '^' ||  getCharAtxy(pX - 1, pY) == '<' || getCharAtxy(pX - 1, pY + 1) == '<' || getCharAtxy(pX - 1, pY) == '-' || getCharAtxy(pX - 1, pY + 1) == '-' || getCharAtxy(pX - 1, pY) == '(' || getCharAtxy(pX - 1, pY + 1) == '(' ||  getCharAtxy(pX + 3, pY) == '-' || getCharAtxy(pX + 3, pY + 1) == '-' || getCharAtxy(pX + 3, pY) == ')' || getCharAtxy(pX + 3, pY + 1) == ')' || getCharAtxy(pX + 3, pY) == '-' || getCharAtxy(pX + 3, pY + 1) == '-')  {  playerHealth = playerHealth - 10;  }  gotoxy(10, 26);  cout << " ";  setConsoleColor(14);  gotoxy(10, 26);  cout << "Health Remaining " << playerHealth;  setConsoleColor(7);  }  void telePoratingPlayer(int &teleporateCounter, int &pX, int &pY)  {  if (getCharAtxy(pX - 1, pY) == 'T' || getCharAtxy(pX - 1, pY + 1) == 'T' || getCharAtxy(pX + 3, pY) == 'T' || getCharAtxy(pX + 3, pY + 1) == 'T' || getCharAtxy(pX, pY + 2) == 'T' || getCharAtxy(pX + 1, pY + 2) == 'T' || getCharAtxy(pX + 2, pY + 2) == 'T' || getCharAtxy(pX, pY - 1) == 'T' || getCharAtxy(pX + 1, pY - 1) == 'T' || getCharAtxy(pX + 2, pY - 1) == 'T')  {  teleporateCounter++;  if (teleporateCounter % 4 == 0)  {  eraseJerry(pX, pY);  pX = 90, pY = 18;  printJerry(pX, pY);  }  else if (teleporateCounter % 4 == 1)  {  eraseJerry(pX, pY);  pX = 200, pY = 40;  printJerry(pX, pY);  }  else if (teleporateCounter % 4 == 2)  {  eraseJerry(pX, pY);  pX = 120, pY = 50;  printJerry(pX, pY);  }  else if (teleporateCounter % 4 == 3)  {  eraseJerry(pX, pY);  pX = 149, pY = 25;  printJerry(pX, pY);  }  }  }  void readMaze(const char \*filename, char maze[rows][cols])  {  ifstream file(filename);  if (!file)  {  cerr << "Error opening file: " << filename << endl;  return;  }  for (int i = 0; i < rows; ++i)  {  file.getline(maze[i], cols + 1); // +1 to account for the null terminator  }  file.close();  }  void generateFire(int &fireCounter, int fireX[], int fireY[], int maxFire, int pX, int pY, int eX, int eY)  {  int a = abs(pX - eX);  int b = abs(pY - eY);  if (a <= 10 && b <= 10)  {  fireCounter++;  if (fireCounter % 5 == 0 && fireCounter < maxFire && getCharAtxy(eX, eY + 3) == ' ')  {  fireX[fireCounter] = eX + 3;  fireY[fireCounter] = eY + 3;  if (getCharAtxy(fireX[fireCounter], fireY[fireCounter]) == ' ')  {  gotoxy(fireX[fireCounter], fireY[fireCounter]);  cout << "\*";  }  }  }  }  void moveFire(int &fireCounter, int fireX[], int fireY[], int maxFire)  {  for (int i = 1; i <= fireCounter; i++)  {  if (fireX[i] != 0 && fireY[i] != 0)  {  gotoxy(fireX[i], fireY[i]);  cout << " ";  if (fireY[i] < rows - 1) // Check if the fire is within the maze boundaries  {  fireY[i]++;  if (getCharAtxy(fireX[i], fireY[i]) == ' ')  {  gotoxy(fireX[i], fireY[i]);  setConsoleColor(12);  cout << "\*";  setConsoleColor(7);  }  else  {  // If the space in the direction of fire is not empty, consider handling collision or remove the fire  // For example, you can set fireX[i] and fireY[i] to 0 to indicate that the fire has hit an obstacle.  fireX[i] = 0;  fireY[i] = 0;  }  }  else  {  // The fire has reached the bottom boundary, remove or handle accordingly  fireX[i] = 0;  fireY[i] = 0;  }  }  }  }  void gameHeader()  {  cout << "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n";  SetConsoleTextAttribute(h, 12);  cout << "\t\t\t\t \_\_\_\_\_\_\_ ";  SetConsoleTextAttribute(h, 11);  cout << "\_\_\_\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << " \_\_\_\_\_\_\_ ";  SetConsoleTextAttribute(h, 11);  cout << " \_\_\_\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << " \_\_\_\_\_\_ ";  SetConsoleTextAttribute(h, 11);  cout << "\_\_\_\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << " \_\_\_\_\_\_\_ ";  SetConsoleTextAttribute(h, 11);  cout << "\_\_\_\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << " \_\_\_\_\_\_\_ ";  SetConsoleTextAttribute(h, 11);  cout << "\_\_\_\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << " \_\_\_\_\_\_\_ ";  SetConsoleTextAttribute(h, 11);  cout << "\_\_\_\_\_\_\_" << endl;  SetConsoleTextAttribute(h, 12);  cout << "\t\t\t\t( \_\_\_\_ \\";  SetConsoleTextAttribute(h, 11);  cout << "( \_\_\_\_ )";  SetConsoleTextAttribute(h, 12);  cout << "( \_\_\_\_ \\";  SetConsoleTextAttribute(h, 11);  cout << "( \_\_\_\_ \\";  SetConsoleTextAttribute(h, 12);  cout << "( \_\_ \\ ";  SetConsoleTextAttribute(h, 11);  cout << "( \_\_\_ )";  SetConsoleTextAttribute(h, 12);  cout << "( ) ";  SetConsoleTextAttribute(h, 11);  cout << "( \_\_\_\_\\";  SetConsoleTextAttribute(h, 12);  cout << " ( \_\_\_ )";  SetConsoleTextAttribute(h, 11);  cout << "( \_\_\_\_ )";  SetConsoleTextAttribute(h, 12);  cout << "( \_\_\_\_ \\";  SetConsoleTextAttribute(h, 11);  cout << "( \_\_\_\_ \\" << endl;  cout << "\t\t\t\t| ( \\/";  SetConsoleTextAttribute(h, 11);  cout << "| ( ";  SetConsoleTextAttribute(h, 12);  cout << ")|| ( \\/";  SetConsoleTextAttribute(h, 11);  cout << "| ( ";  SetConsoleTextAttribute(h, 12);  cout << "\\/| ( \\ )";  SetConsoleTextAttribute(h, 11);  cout << "| ( )";  SetConsoleTextAttribute(h, 12);  cout << " || () () | ";  SetConsoleTextAttribute(h, 11);  cout << "| ( ";  SetConsoleTextAttribute(h, 12);  cout << "\\/| ( ) |";  SetConsoleTextAttribute(h, 11);  cout << "| ( ";  SetConsoleTextAttribute(h, 12);  cout << ")|| ( \\/| ( \\/" << endl;  cout << "\t\t\t\t| (\_\_ ";  SetConsoleTextAttribute(h, 11);  cout << "| (\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << ")|| (\_\_ ";  SetConsoleTextAttribute(h, 11);  cout << "| (\_\_ ";  SetConsoleTextAttribute(h, 12);  cout << " | | ) || | ";  SetConsoleTextAttribute(h, 11);  cout << "| || ||";  SetConsoleTextAttribute(h, 12);  cout << " || | ";  SetConsoleTextAttribute(h, 11);  cout << "| (\_\_ ";  SetConsoleTextAttribute(h, 12);  cout << " | | | |";  SetConsoleTextAttribute(h, 11);  cout << "| (\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << ")|| | | (\_\_ " << endl;  cout << "\t\t\t\t";  SetConsoleTextAttribute(h, 11);  cout << "| \_\_) ";  SetConsoleTextAttribute(h, 12);  cout << " | \_\_)";  SetConsoleTextAttribute(h, 11);  cout << "| \_\_) ";  SetConsoleTextAttribute(h, 12);  cout << " | \_\_) |";  SetConsoleTextAttribute(h, 11);  cout << " | | ";  SetConsoleTextAttribute(h, 12);  cout << "|| | | |";  SetConsoleTextAttribute(h, 11);  cout << "| |(\_)|";  SetConsoleTextAttribute(h, 12);  cout << " | | \_\_) ";  SetConsoleTextAttribute(h, 11);  cout << "| | |";  SetConsoleTextAttribute(h, 12);  cout << " || \_\_)| ";  SetConsoleTextAttribute(h, 11);  cout << "| \_\_\_\_ ";  SetConsoleTextAttribute(h, 12);  cout << "| \_\_) " << endl;  cout << "\t\t\t\t";  SetConsoleTextAttribute(h, 11);  cout << "| ( ";  SetConsoleTextAttribute(h, 12);  cout << " | (\\ ( ";  SetConsoleTextAttribute(h, 11);  cout << "| ( ";  SetConsoleTextAttribute(h, 12);  cout << " | ( ";  SetConsoleTextAttribute(h, 11);  cout << "| | )";  SetConsoleTextAttribute(h, 12);  cout << " || | | |";  SetConsoleTextAttribute(h, 11);  cout << "| | |";  SetConsoleTextAttribute(h, 12);  cout << " | | ( ";  SetConsoleTextAttribute(h, 11);  cout << "| | |";  SetConsoleTextAttribute(h, 12);  cout << " || (\\ ( | ";  SetConsoleTextAttribute(h, 11);  cout << "| \\\_ ";  SetConsoleTextAttribute(h, 12);  cout << ")| ( " << endl;  cout << "\t\t\t\t";  SetConsoleTextAttribute(h, 11);  cout << "| ) ";  SetConsoleTextAttribute(h, 12);  cout << " | ) \\ \\\_\_";  SetConsoleTextAttribute(h, 11);  cout << "| (\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << "/\\| (\_\_\_\_";  SetConsoleTextAttribute(h, 11);  cout << "/\\| (\_";  SetConsoleTextAttribute(h, 12);  cout << "\_/ )| (\_\_\_) |";  SetConsoleTextAttribute(h, 11);  cout << "| ) (";  SetConsoleTextAttribute(h, 12);  cout << " | | ) ";  SetConsoleTextAttribute(h, 11);  cout << "| (\_\_\_)";  SetConsoleTextAttribute(h, 12);  cout << " || ) \\ \\\_\_";  SetConsoleTextAttribute(h, 11);  cout << "| (\_\_\_)";  SetConsoleTextAttribute(h, 12);  cout << " || (\_\_\_\_/\\" << endl;  cout << "\t\t\t\t";  SetConsoleTextAttribute(h, 11);  cout << "|/ ";  SetConsoleTextAttribute(h, 12);  cout << " |/ \\\_\_/";  SetConsoleTextAttribute(h, 11);  cout << "(\_\_\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << "\_/(\_\_\_\_\_\_\_/";  SetConsoleTextAttribute(h, 11);  cout << "(\_\_\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << "/ (\_\_\_\_\_\_\_)";  SetConsoleTextAttribute(h, 11);  cout << "|/ ";  SetConsoleTextAttribute(h, 12);  cout << "\\| |/ ";  SetConsoleTextAttribute(h, 11);  cout << "(\_\_\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << "\_)|/ \\\_\_/";  SetConsoleTextAttribute(h, 11);  cout << "(\_\_\_\_\_\_";  SetConsoleTextAttribute(h, 12);  cout << "\_)(\_\_\_\_\_\_\_/" << endl;  setConsoleColor(14);  cout << "\n\t\t\t\t\tPress Enter to start ";  setConsoleColor(14);  while (\_getch() != 13) // Wait for Enter key press  {  // Do nothing until Enter key is pressed  }  }  void loadingFunction()  {  int Mx = 70, My = 16;  string box = "\xDC"; // ASCII code for a filled box  // Display "Loading..." message  gotoxy(Mx + 27, My + 32);  cout << "Loading...";  setConsoleColor(10);  // Position the loading cursor at the start  gotoxy(Mx, My + 30);  cout << box; // Set color to green and display a filled box  setConsoleColor(7);  // Display loading animation  for (int i = 0; i < 60; i++)  {  Mx++;  setConsoleColor(10);  gotoxy(Mx, My + 30);  cout << box; // Display a filled box at the current position  setConsoleColor(7);  }  }  void playerLivesSystem(int &playerLives, int &playersHealth, int &pX, int &pY)  {  if (playersHealth <= 0)  {  playerLives--;  eraseJerry(pX, pY);  pX = 45, pY = 15;  printJerry(pX, pY);  playersHealth = 100;  }  gotoxy(10, 28);  cout << " ";  setConsoleColor(14);  gotoxy(10, 28);  cout << "Lives Remaining " << playerLives;  setConsoleColor(7);  }  void overHeader()  {  system("cls");  setConsoleColor(12);  cout << "\n\n\n\n\n\n\n";  cout << "\t\t\t (\_\_\_\_\_\_\_) (\_\_\_\_\_\_\_) " << endl;  cout << "\t\t\t \_ \_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_\_ \_ \_ \_ \_ \_\_\_\_\_ \_\_\_\_ " << endl;  cout << "\t\t\t | | (\_ (\_\_\_\_ | \\| \_\_\_ | | | | | | | | \_\_\_ |/ \_\_\_)" << endl;  cout << "\t\t\t | |\_\_\_) / \_\_\_ | | | | \_\_\_\_| | |\_\_\_| |\\ V /| \_\_\_\_| | " << endl;  cout << "\t\t\t \\\_\_\_\_\_/\\\_\_\_\_\_|\_|\_|\_|\_\_\_\_\_) \\\_\_\_\_\_/ \\\_/ |\_\_\_\_\_)\_| " << endl;  setConsoleColor(7);  setConsoleColor(14);  cout << "\n\n\n\n\t\t\t Press Enter to continue and any key to restart the game ";  setConsoleColor(7);  } |