Fire Buddy

Session: 2022 – 2026

Submitted by:

Muhammad Wali Ahmad 2022-CS-65

Supervised by:

Maida Shahid

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Contents

1. Short Description and Story Writing of your Game: 3
2. Game Characters Description: 3
2.1 Player: 3
2.2 Enemies: 3
3. Game Objects Description: 4
3.1 Door: 4
3.2 Score boosters: 4
3.3 Walls: 4
3.4 Magic Walls: 4
4 Rules & Interactions: 4
5 Goal of the Game: 4
6 Wireframes of the Game: 5
7 Data Structures: 9
8 Function Prototypes: 9
9 Complete Code: 12

1. Short Description and Story Writing of your Game:

Burning with the ambition of world domination, King Kratos of the Kingdom of Sparta with his companions abducted the beautiful princess of Sweetland as the beginning of the invasion and imprisoned them deep inside the castle of the Kingdom of Sparta. Sweetland, who lost the princess who controls the magic of fire, begins too cold little by little. At this rate, the Sweetland will soon disappear! Tom, the fireman, felt unprecedented resentment at being robbed of the princess in front of him. He vowed to rescue the princess from the hands of the villains, and set out for his country that is Sweetland, where the enemies were waiting. The home of the enemies, Sparta Castle, is a 2-store stone tower. A stone-breathing monster is roaming, and Kratos is worried about the fire and built the castle out of stone. It's more like a fortress than a castle because it's durable first, so it's not a very elegant castle. The fireman, who are not good at rocky places, tried to infiltrate the rocky castle of Kratos and his companions with courage.

2. Game Characters Description:

2.1 Player:

There is one human player in the Game.

Tom:

Tom is the main character in the game and is known for his fireballs. He is brave and loves to save his motherland, always searching the way to save his country from the enemy. Tom is brave, determined, and has a never-say-die spirit. He is the hero of the game, admired for his bravery and determination in the face of danger. Its health is 100.

2.2 Enemies:

There are 4 enemies in the game.

Botchan:

Botchan is one of the four monsters in the game and is known for his vertical movement in the game. Its health is 50.

Titchi:

Titchi is another monster and is known for his horizontal movement in the game. Its health is also 50 like Botchna.

Sarada:

Sarada is one of the four monsters. It throws incendiary bombs when the player is close which reduce the health of Tom. Its health is 100.When his fire collied with the tom, tom health decrease by 10.

Kratos:
Kratos is one of the four monsters in the game and is known for being aggressive and difficult to shake. He is always chasing Tom through the maze, trying to catch him at every turn. Kratos is fast and relentless, making him one of the most dangerous foes that Tom must face. Its health is 200.

3. Game Objects Description:

Following are the Objects in the Game

3.1 Door:
A Door, also known as Next Level, is an object used in the Fire Buddy game. When Tom Touches a Door, it causes the Tom to go into Next level, but it only used when all ghost is dead.

3.2 Score boosters:
Small Dollars are called “score boosters”. When Tom collects a score booster its score increases by 10.

3.3 Walls:
Walls are the barriers in the game which the Tom and the monsters cannot cross. In game these are represented by #

3.4 Magic Walls:
These walls are used by Tom to teleport from one end of the maze to other end.

4 Rules & Interactions:

Following are the rules and interaction of the game:
- Tom loses a life if he collides with any of the monsters.
- If Tom eats score booster, then the Score will increase by 10.
- Score increases when the Tom kills monsters by firing fireballs.
- When the fireballs hit the monster the health of the monster decreases and score increases
- When enemy fire on Tom then his health also decreases.
- When Tom touches the princess in last level then game finished.

5 Goal of the Game:

The goal of the game is to kill all of the monster that have been put across the maze while saving Tom lives and touches the door for next level. In the next level Tom fight with Kratos which is the king of Sparta. After beating him, the cage of princess opened and Tom catch her and game finished.

6 Wireframes of the Game:

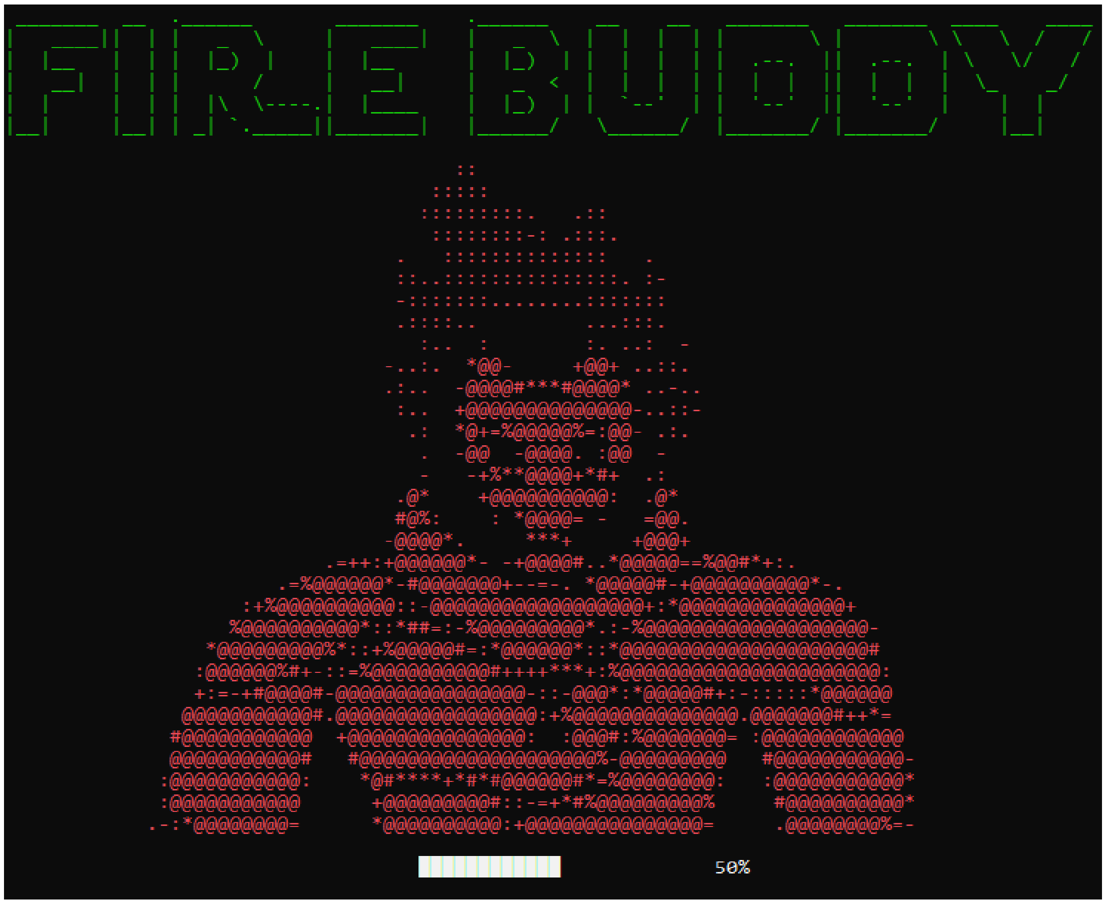
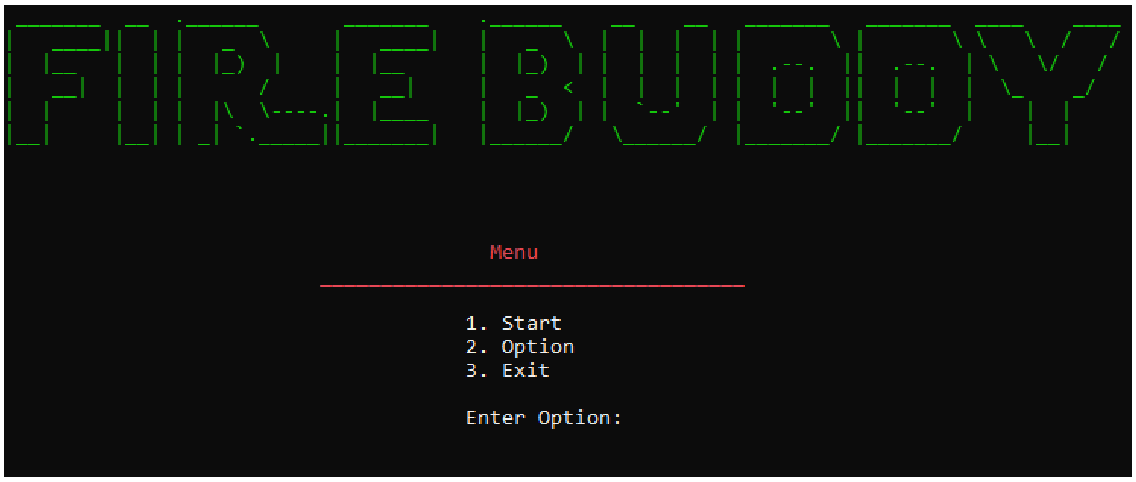
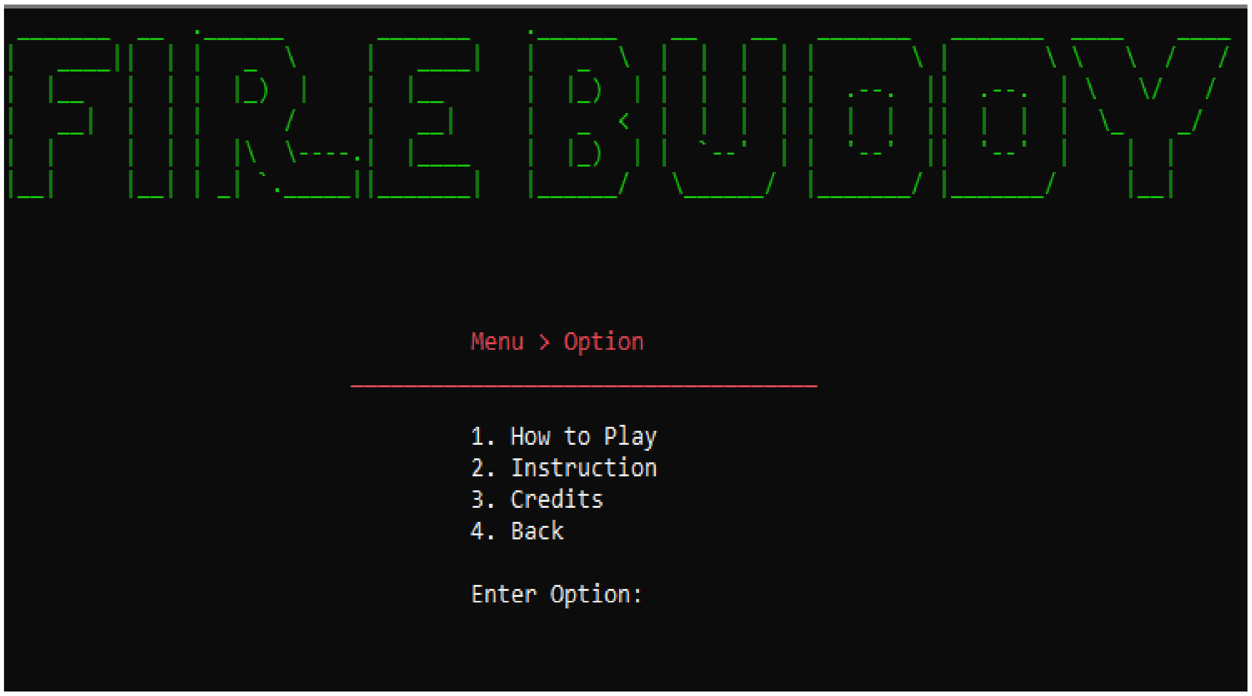


Figure 1: Loading Screen



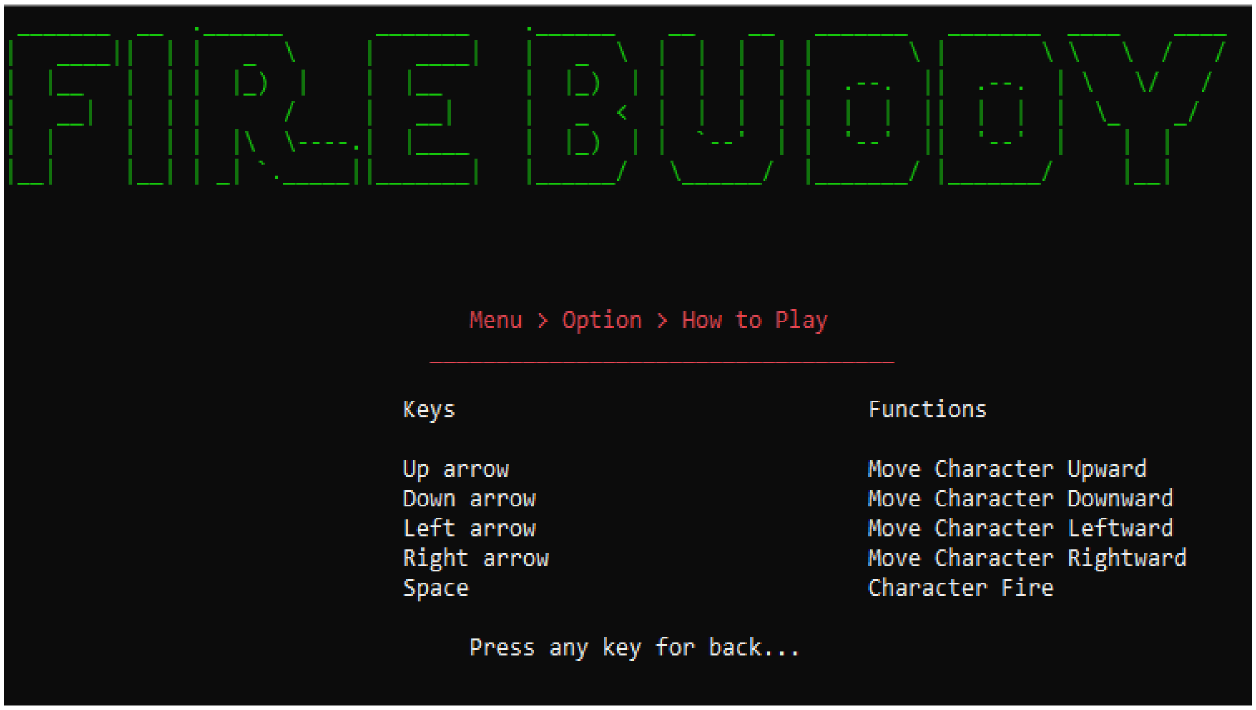
```
Menu
_______________________________

1. Start
2. Option
3. Exit

Enter Option:
```

Figure 2: Main Menu



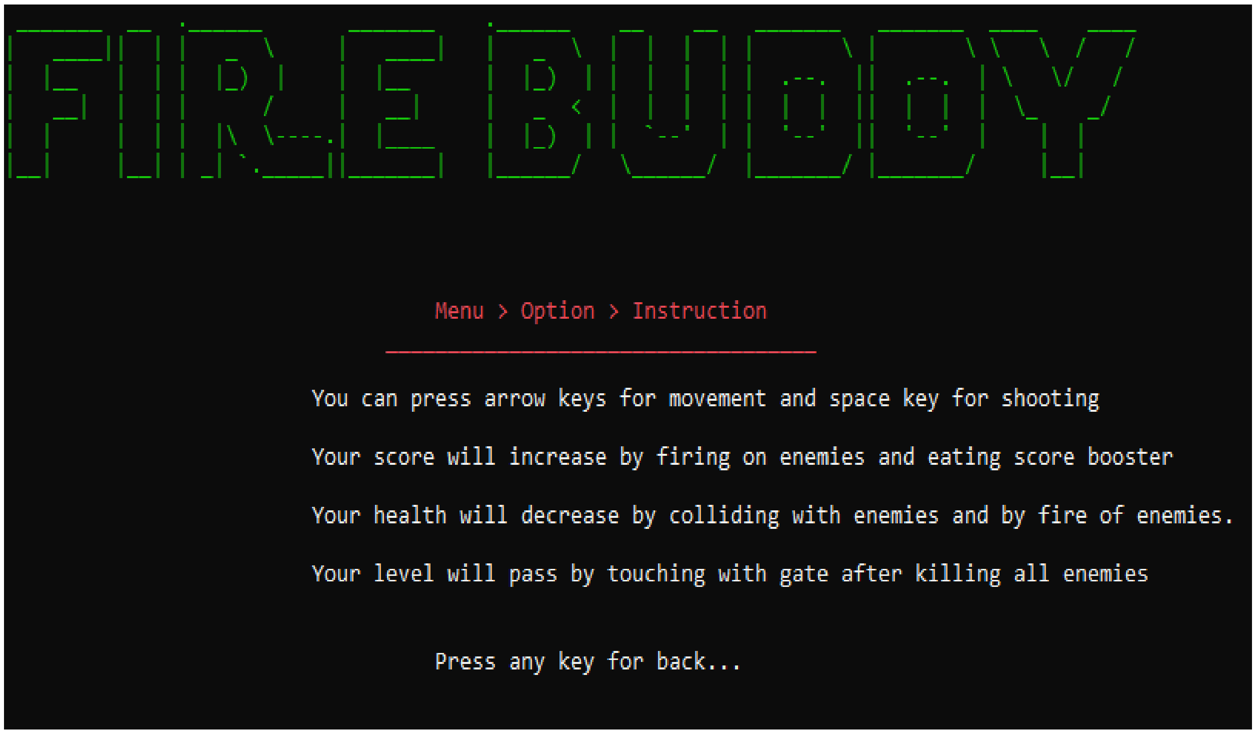
```
Menu > Option
________________________________

1. How to Play
2. Instruction
3. Credits
4. Back

Enter Option:
```

Figure 3: Option Menu



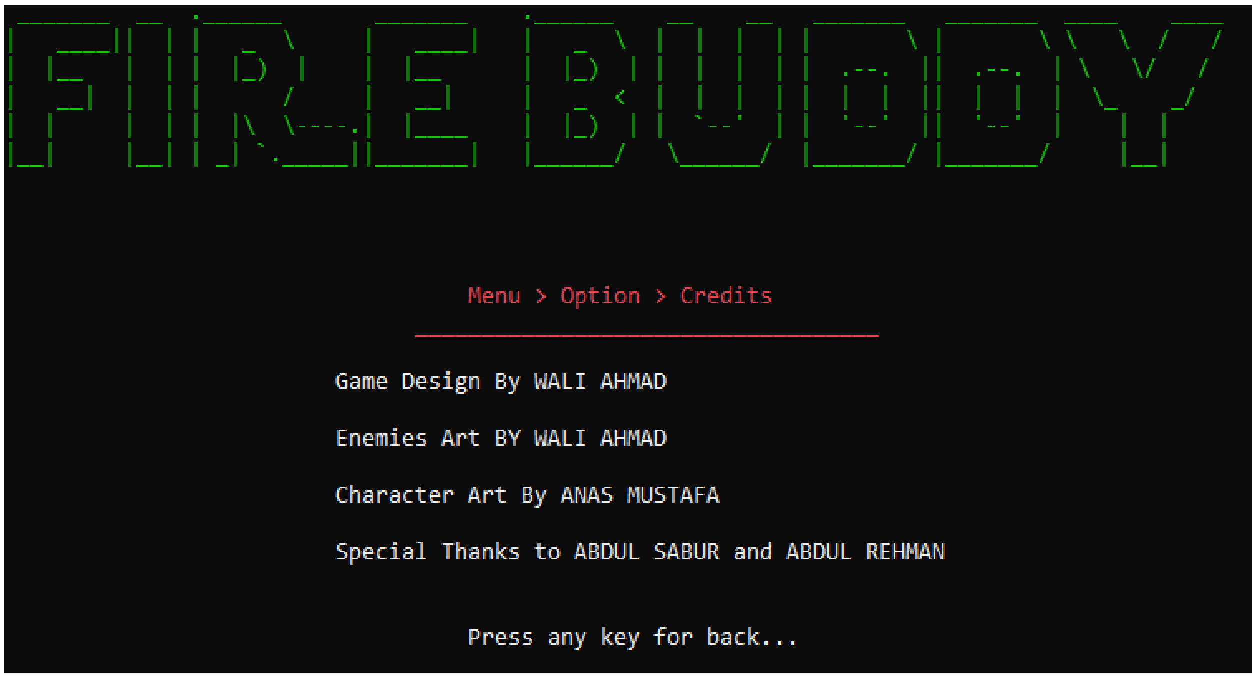
```
     Menu > Option > How to Play
   ________________________________

Keys                               Functions

Up arrow                           Move Character Upward
Down arrow                         Move Character Downward
Left arrow                         Move Character Leftward
Right arrow                        Move Character Rightward
Space                              Character Fire

     Press any key for back...
```

Figure 4: How to Play



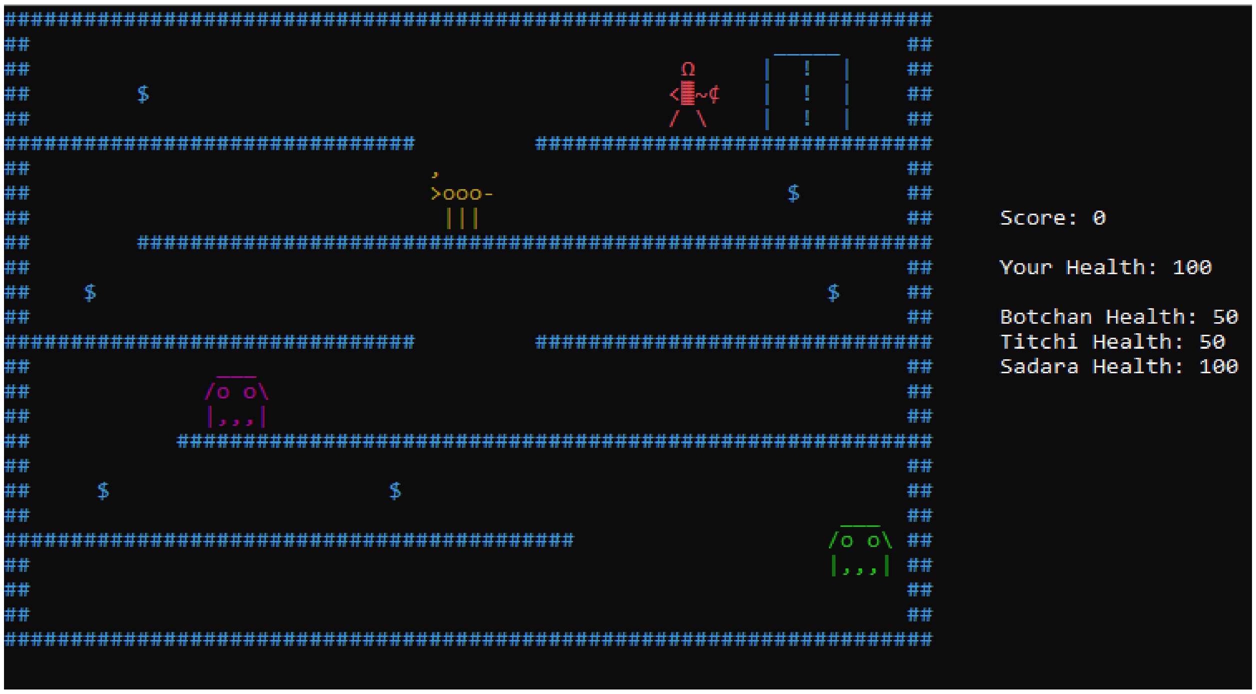
```
Menu > Option > Instruction

You can press arrow keys for movement and space key for shooting

Your score will increase by firing on enemies and eating score booster

Your health will decrease by colliding with enemies and by fire of enemies.

Your level will pass by touching with gate after killing all enemies


Press any key for back...
```

Figure 5: Instruction



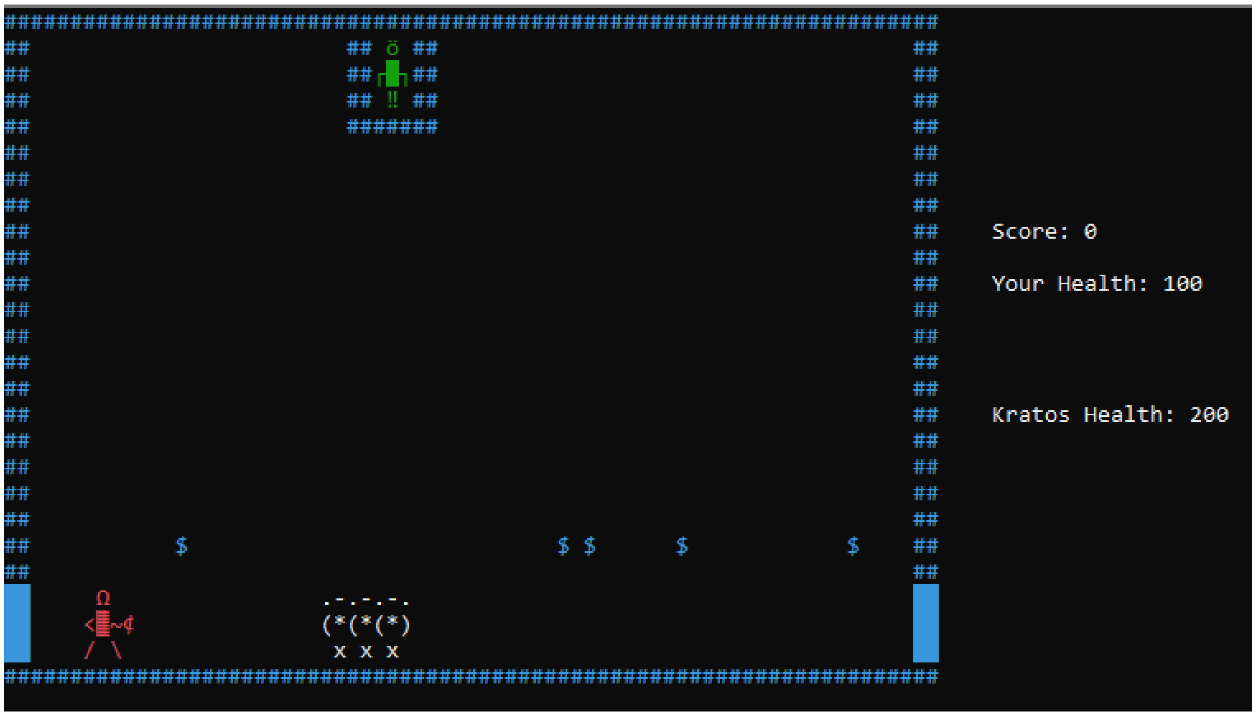
```
Menu > Option > Credits

Game Design By WALI AHMAD

Enemies Art BY WALI AHMAD

Character Art By ANAS MUSTAFA

Special Thanks to ABDUL SABUR and ABDUL REHMAN


Press any key for back...
```

Figure 6: Credits



```
Score: 0

Your Health: 100

Botchan Health: 50
Titchi Health: 50
Sadara Health: 100
```

Figure 7: Level 1



```
Score: 0

Your Health: 100


Kratos Health: 200
```

Figure 8: Level 2

7 Data Structures:

```
char Maze1[26][70]; // All Arrays
char Maze2[26][71];
char Buddy[3][4];
char BuddyLeft[3][4];
char princess[3][3];
char Enemy[3][5];
char Ant[3][5];
char Kratos[3][7];
```

8 Function Prototypes:

```
// functions

// helping functions
void gotoxy(int x, int y);
string setcolor(unsigned short color);
char getCharAtxy(short int x, short int y);
void ShowConsoleCursor(bool showFlag);

void header(); // first page functions and sub-functions
void mainPic();
void loading();
void firstPage();

void option(); // second page functions and sub-functions
void howToPlay();
void instruction();
void credits();
int menu();
void Lastcredits();

void printMaze1(char Maze1[26][70]); // maze functions
void printMaze2(char Maze2[26][71]);

void printBuddy(char Buddy[3][4], int &BuddyX, int &BuddyY); // buddy
control, print functions with passing arrays
void printBuddyLeft(char BuddyLeft[3][4], int &BuddyX, int &BuddyY);
void teleportBuddy(char Buddy[3][4], char BuddyLeft[3][4], int
&BuddyX, int &BuddyY);
void eraseBuddy(int &BuddyX, int &BuddyY);
void moveBuddyDown(char Buddy[3][4], char BuddyLeft[3][4], int
&BuddyX, int &BuddyY, string &printDirection, int &score);
void moveBuddyUp(char Buddy[3][4], char BuddyLeft[3][4], int &BuddyX,
int &BuddyY, string &printDirection, int &score, int &Buddyhealth);
void moveBuddyLeft(char BuddyLeft[3][4], int &BuddyX, int &BuddyY,
int &princessX, int &princessY, string &printDirection, int &check,
bool &game, bool &game2, int &score);
void moveBuddyRight(char Buddy[3][4], int &BuddyX, int &BuddyY, int
&princessX, int &princessY, string &printDirection, int &check, bool
&game, bool &game2, int &score, int &Enemy1health, int &Enemy2health,
int &Anthealth);
void printBuddyHealth(int &Buddyhealth);
void controlBuddy(char Buddy[3][4], char BuddyLeft[3][4], int
&BuddyX, int &BuddyY, int &princessX, int &princessY, string
&printDirection, bool &isJump, int &jumpTick, int bulletX[100], int
bulletY[100], char bulletDirection[100], int &bulletCount, int
&check, bool &game, bool &game2, int &score, int &Enemy1health, int
&Enemy2health, int &Anthealth, int &Buddyhealth);
bool canJump(int &BuddyX, int &BuddyY);

void printPrincess(char princess[3][3], int &princessX, int
&princessY); // Princess control, print functions with passing arrays
void erasePrincess(int &princessX, int &princessY);
void movePrincessDown(char princess[3][3], int &princessX, int
&princessY);
void jailOpen(int &jailX, int &jailY);

void createBullet(int &BuddyX, int &BuddyY, string &printDirection,
int bulletX[100], int bulletY[100], char bulletDirection[100], int
&bulletCount); // buddy bullets functions, arrays and array counter
void movebullet(int bulletX[100], int bulletY[100], char
bulletDirection[100], int &bulletCount);
void printBullet(int x, int y);
void eraseBullet(int x, int y);
void deleteBullet(int index, int bulletX[100], int bulletY[100], char
bulletDirection[100], int &bulletCount);

void printEnemy1(char Enemy[3][5], int &enemy1X, int &enemy1Y); //
horzontal enemy functions and health variable
void eraseEnemy1(int &enemy1X, int &enemy1Y);
void moveEnemy1(char Enemy[3][5], int &enemy1X, int &enemy1Y, string
&enemy1direction, int &Enemy1health);

void printEnemy2(char Enemy[3][5], int &enemy2X, int &enemy2Y); //
vertical enemy functions and health variable
void eraseEnemy2(int &enemy2X, int &enemy2Y);
void moveEnemy2(char Enemy[3][5], int &enemy2X, int &enemy2Y, string
&enemy2direction, int &Enemy2health);

void printAnt(char Ant[3][5], int &AntX, int &AntY); // Ant and its
bullet functions and health variable
void eraseAnt(int &AntX, int &AntY);
void AntBullet(int &BuddyX, int &BuddyY, int &AntX, int &AntY, int
AntBulletX[10], int AntBulletY[10], int &AntCount, int &Anthealth,
bool &AntFlag);
void moveAntBullet(int AntBulletX[10], int AntBulletY[10], int
&AntCount);
void printAntBullet(int x, int y);
void eraseAntBullet(int x, int y);
void deleteAntBullet(int index, int AntBulletX[10], int
AntBulletY[10], int &AntCount);

void printkratos(char Kratos[3][7], int &KratosX, int &KratosY); //
kratos functions and health variable
void erasekratos(int &KratosX, int &KratosY);
void movekratos(char Kratos[3][7], int &BuddyX, int &BuddyY, int
&KratosX, int &KratosY, int &kratosHealth);

void Printenemyhealth(bool &game, bool &game2, int &Enemy1health, int
&Enemy2health, int &Anthealth, int &kratosHealth); // printing
enemies health

void collision(int &Buddyhealth, int &enemy1X, int &enemy1Y, int
&enemy2X, int &enemy2Y, int &AntX, int &AntY, int &KratosX, int
&KratosY, int bulletX[100], int bulletY[100], char
bulletDirection[100], int &bulletCount, int &Enemy1health, int
&Enemy2health, int &Anthealth, int &kratosHealth, int &score); //
bullet collision function

void addScore(int &score); // score print and increment function
void printScore(int &score);

void gameoverCollsion(int &BuddyX, int &BuddyY, int
&Buddyhealth, int &enemy1X, int &enemy1Y, int &enemy2X, int &enemy2Y,
int &KratosX, int &KratosY, int AntBulletX[10], int AntBulletY[10],
int &AntCount, int &Enemy1health, int &Enemy2health, int
&kratosHealth); // check gameover function

void gameover(int &Buddyhealth, bool &game, bool &game2);
// print gameover function
void complete(int &check, bool &game, bool &game2); // Game complete
function
```

```
void generateRandomCoin(int &count); // random coin generator

void loadMaze1(char Maze1[26][70]); // loading arrays functions
void loadMaze2(char Maze2[26][71]);
void loadBuddy(char Buddy[3][4]);
void loadBuddyLeft(char BuddyLeft[3][4]);
void loadPrincess(char princess[3][3]);
void loadEnemy(char Enemy[3][5]);
void loadAnt(char Ant[3][5]);
void loadKratos(char Kratos[3][7]);
void loadLevel(char &Level);
void writeLevel(char &Level, char number); // level saver
```

9 Complete Code:

```
main()
{
    char Maze1[26][70]; // All Arrays
    char Maze2[26][71];
    char Buddy[3][4];
    char BuddyLeft[3][4];
    char princess[3][3];
    char Enemy[3][5];
    char Ant[3][5];
    char Kratos[3][7];
    char Level;

    loadMaze1(Maze1); // file load functions
    loadMaze2(Maze2);
    loadBuddy(Buddy);
    loadBuddyLeft(BuddyLeft);
    loadPrincess(princess);
    loadEnemy(Enemy);
    loadAnt(Ant);
    loadKratos(Kratos);
    loadLevel(Level);
    srand(time(0));

    int BuddyX = 3; // buddy coordinates,health and other helping
variables
    int BuddyY = 22;
    int Buddyhealth = 100;
    string printDirection = "right";
    bool isJump = false;
    int jumpTick = 0;
    int bulletX[100]; // buddy bullets coordinates and counter
    int bulletY[100];
    char bulletDirection[100];
    int bulletCount = 0;

    int princessX = 28; // princess coordinates
    int princessY = 1;

    int jailX = 26; // jail coordinates
    int jailY = 4;

    int enemy1X = 62; // enemy1 coordinates and health
    int enemy1Y = 22;
    string enemy1direction = "down";
    int Enemy1health = 50;

    int enemy2X = 14; // enemy2 coordinates and health
    int enemy2Y = 14;
    string enemy2direction = "left";
    int Enemy2health = 50;

    int AntX = 32; // Ant coordinates and health
    int AntY = 6;
    int Anthealth = 100;
    bool AntFlag = true;

    int AntBulletX[100]; // Ant bullets Arrays
    int AntBulletY[100];
    int AntCount = 0;

    int KratosX = 62; // Kratos coordinates and health
    int KratosY = 22;
    int kratosHealth = 200;

    int score = 0; // score

    int check = 0; // flags for game
    bool game = false;
    bool game2 = false;
    bool credit = false;
    int count = 0;
    int tick = 0;

    ShowConsoleCursor(false); // remove cursors
    firstPage(); // loading page and options menu
    int x = menu();
    system("cls");

    if (Level == '1') // level one loader
    {
        game = true;
        game2 = true;
    }
    if (x == 1 && game == true) // printing enemies and buddy for
level one
    {
        printMaze1(Maze1);
        printEnemy1(Enemy, enemy1X, enemy1Y);
        printEnemy2(Enemy, enemy2X, enemy2Y);
        printAnt(Ant, AntX, AntY);
        printBuddy(Buddy, BuddyX, BuddyY);
        KratosX = 0;
        KratosY = 0;
    }

    while (x == 1 && game) // level one loop
    {
        printScore(score);
        printBuddyHealth(Buddyhealth);

        gameover(Buddyhealth, game, game2);

        moveEnemy1(Enemy, enemy1X, enemy1Y, enemy1direction,
Enemy1health);
        moveEnemy2(Enemy, enemy2X, enemy2Y, enemy2direction,
Enemy2health);

        AntBullet(BuddyX, BuddyY, AntX, AntY, AntBulletX, AntBulletY,
AntCount, Anthealth, AntFlag);

        moveAntBullet(AntBulletX, AntBulletY, AntCount);
        gameoverCollsion(BuddyX, BuddyY, Buddyhealth, enemy1X,
enemy1Y, enemy2X, enemy2Y, KratosX, KratosY, AntBulletX, AntBulletY,
AntCount, Enemy1health, Enemy2health, kratosHealth);
        movebullet(bulletX, bulletY, bulletDirection, bulletCount);
        collision(Buddyhealth, enemy1X, enemy1Y, enemy2X, enemy2Y,
AntX, AntY, KratosX, KratosY, bulletX, bulletY, bulletDirection,
bulletCount, Enemy1health, Enemy2health, Anthealth, kratosHealth,
score);
```

```
        Printenemyhealth(game, game2, Enemy1health, Enemy2health,
Anthealth, kratosHealth);
        controlBuddy(Buddy, BuddyLeft, BuddyX, BuddyY, princessX,
princessY, printDirection, isJump, jumpTick, bulletX, bulletY,
bulletDirection, bulletCount, check, game, game2, score,
Enemy1health, Enemy2health, Anthealth, Buddyhealth);
        Sleep(50);
    }
    system("cls");
    writeLevel(Level, '2'); // level two loader
    if (Level == '2')
    {
        game2 = true;
    }
    if (game2 == true) // printing enemies and buddy for level two
    {
        printMaze2(Maze2);
        BuddyX = 3;
        BuddyY = 22;
        KratosX = 62;
        KratosY = 22;
        printBuddy(Buddy, BuddyX, BuddyY);
        printkratos(Kratos, KratosX, KratosY);
        printPrincess(princess, princessX, princessY);
        enemy1X = 0;
        enemy1Y = 0;
        enemy2X = 0;
        enemy2Y = 0;
        AntX = 0;
        AntY = 0;
    }

    while (game2) // level two loop
    {
        movePrincessDown(princess, princessX, princessY);
        controlBuddy(Buddy, BuddyLeft, BuddyX, BuddyY, princessX,
princessY, printDirection, isJump, jumpTick, bulletX, bulletY,
bulletDirection, bulletCount, check, game, game2, score,
Enemy1health, Enemy2health, Anthealth, Buddyhealth);
        printScore(score);
        movekratos(Kratos, BuddyX, BuddyY, KratosX, KratosY,
kratosHealth);
        printBuddyHealth(Buddyhealth);
        gameover(Buddyhealth, game, game2);
        gameoverCollsion(BuddyX, BuddyY, Buddyhealth, enemy1X,
enemy1Y, enemy2X, enemy2Y, KratosX, KratosY, AntBulletX, AntBulletY,
AntCount, Enemy1health, Enemy2health, kratosHealth);
        movebullet(bulletX, bulletY, bulletDirection, bulletCount);
        collision(Buddyhealth, enemy1X, enemy1Y, enemy2X, enemy2Y,
AntX, AntY, KratosX, KratosY, bulletX, bulletY, bulletDirection,
bulletCount, Enemy1health, Enemy2health, Anthealth, kratosHealth,
score);
        Printenemyhealth(game, game2, Enemy1health, Enemy2health,
Anthealth, kratosHealth);
        teleportBuddy(Buddy, BuddyLeft, BuddyX, BuddyY);
        if (tick == 10)
        {
            generateRandomCoin(count);
            tick = 0;
        }
        if (kratosHealth == -1)
        {
            jailOpen(jailX, jailY);
            credit = true;
        }
        tick++;
        Sleep(50);
    }
    system("cls");
    writeLevel(Level, '1');
    if (x == 1 && credit == true) // last credits
    {
        Lastcredits();
    }
}

void header()
{
    setcolor(10);
    cout << " _______  __  .______       _______     .______    __    __   _______   _______  ____    ____ " << endl;
    cout << "|   ____||  | |   _  \     |   ____|    |   _  \  |  |  |  | |       \ |       \ \   \  /   / " << endl;
    cout << "|  |__   |  | |  |_)  |    |  |__       |  |_)  | |  |  |  | |  .--.  ||  .--.  | \   \/   /  " << endl;
    cout << "|   __|  |  | |      /     |   __|      |   _  <  |  |  |  | |  |  |  ||  |  |  |  \_    _/   " << endl;
    cout << "|  |     |  | |  |\  \----.|  |____     |  |_)  | |  `--'  | |  '--'  ||  '--'  |    |  |     " << endl;
    cout << "|__|     |__| | _| `._____||_______|    |______/   \______/  |_______/ |_______/     |__|     " << endl;
  cout << endl;
    setcolor(15);
}

void mainPic()
{
    setcolor(12);
    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;
    setcolor(15);
}

void printMaze1(char Maze1[26][70])
{
    setcolor(03);
    for (int i = 0; i < 26; i++)
    {
        for (int j = 0; j < 70; j++)
        {
            cout << Maze1[i][j];
        }
        cout << endl;
    }
    setcolor(15);
}

void printMaze2(char Maze2[26][71])
{
    setcolor(03);
    for (int i = 0; i < 26; i++)
    {
        for (int j = 0; j < 71; j++)
        {
            cout << Maze2[i][j];
        }
        cout << endl;
    }
    setcolor(15);
}

void loading()
{
    int x = 0;
    char loading = 219;
    for (int i = 0; i <= 10; i++)
    {
        gotoxy(35 + x, 39);
        Sleep(300);
        cout << loading << loading;
        x = x + 2;
        gotoxy(60, 39);
        cout << i << "0%";
    }
    gotoxy(40, 40);
    cout << "COMPLETE ";
    Sleep(200);
    cout << "!";
    Sleep(200);
    cout << "!";
    Sleep(500);
}

void firstPage()
{
    system("cls");
    header();
    mainPic();
    loading();
}

int menu()
{
    int choice = 0;
    while (choice != 3)
    {

        system("cls");
        header();

        setcolor(12);
        gotoxy(40, 10);
        cout << "Menu";
        gotoxy(26, 11);
        cout << "___________________________________";

        setcolor(15);
        gotoxy(38, 13);
        cout << "1. Start";
        gotoxy(38, 14);
        cout << "2. Option";
        gotoxy(38, 15);
        cout << "3. Exit";
        gotoxy(38, 17);
        cout << "Enter Option: ";
        cin >> choice;

        if (choice == 1)
        {
            return choice;
        }

        if (choice == 2)
        {
            option();
        }
    }
    if (choice == 3)
    {
        system("cls");
    }
    return 0;
}

void option()
{
    int choice = 0;

    while (choice != 4)
    {
        system("cls");
        header();
        setcolor(12);
        gotoxy(35, 10);
        cout << "Menu > Option";
        gotoxy(26, 11);
        cout << "___________________________________";
        setcolor(15);
        gotoxy(35, 13);
        cout << "1. How to Play";
        gotoxy(35, 14);
        cout << "2. Instruction";
        gotoxy(35, 15);
        cout << "3. Credits";
        gotoxy(35, 16);
        cout << "4. Back";
        gotoxy(35, 18);
        cout << "Enter Option: ";
        cin >> choice;

        if (choice == 1)
        {
            howToPlay();
        }

        if (choice == 2)
        {
            instruction();
        }

        if (choice == 3)
        {
            credits();
        }
    }
}

void howToPlay()
{
    system("cls");
    header();
    setcolor(12);
    gotoxy(35, 10);
    cout << "Menu > Option > How to Play";
    gotoxy(32, 11);
    cout << "___________________________________";
    setcolor(15);
    gotoxy(30, 13);
    cout << "Keys\t\t\t\t Functions";
    gotoxy(30, 15);
    cout << "Up arrow\t\t\t\t Move Character Upward";
    gotoxy(30, 16);
    cout << "Down arrow\t\t\t Move Character Downward";
    gotoxy(30, 17);
    cout << "Left arrow\t\t\t Move Character Leftward";
    gotoxy(30, 18);
    cout << "Right arrow\t\t\t Move Character Rightward";
    gotoxy(30, 19);
    cout << "Space\t\t\t\t Character Fire";
    gotoxy(35, 21);
    cout << "Press any key for back...";
    getch();
}

void instruction()
{
    system("cls");
    header();
    setcolor(12);
    gotoxy(35, 10);
    cout << "Menu > Option > Instruction";
    gotoxy(31, 11);
    cout << "___________________________________";
    setcolor(15);
    gotoxy(25, 13);
    cout << "You can press arrow keys for movement and space key for
shooting";
    gotoxy(25, 15);
    cout << "Your score will increase by firing on enemies and eating
score booster ";
    gotoxy(25, 17);
    cout << "Your health will decrease by colliding with enemies and
by fire of enemies.";
    gotoxy(25, 19);
    cout << "Your level will pass by touching with gate after killing
all enemies";
    gotoxy(25, 21);
    cout << "Your Last level will pass by touching with Princess
after killing Kratos";
    gotoxy(35, 24);
    cout << "Press any key for back...";
    getch();
}

void credits()
{
    system("cls");
    header();
    setcolor(12);
    gotoxy(35, 10);
    cout << "Menu > Option > Credits";
    gotoxy(31, 11);
    cout << "___________________________________";
    setcolor(15);
    gotoxy(25, 13);
    cout << "Game Design By WALI AHMAD";
    gotoxy(25, 15);
    cout << "Enemies Art BY WALI AHMAD";
    gotoxy(25, 17);
    cout << "Character Art By ANAS MUSTAFA";
    gotoxy(25, 19);
    cout << "Special Thanks to ABDUL SABUR and ABDUL REHMAN";
    gotoxy(35, 22);
    cout << "Press any key for back...";
    getch();
}
void Lastcredits()
{
    string line[5] = {"          Game Design By WALI
AHMAD          ", "          Character Art By ANAS MUSTAFA       ",
"          Enemies Art BY WALI AHMAD          ", "          Princess
Art BY WALI AHMAD         ", "Special Thanks to ABDUL SABUR and ABDUL
REHMAN"};
    system("cls");
    header();
    setcolor(12);

    for (int i = 0, j = 0; i < 10; i = i + 2, j++)
    {
        gotoxy(25, 13 + i);
        cout << line[j];
        Sleep(500);
    }

    gotoxy(20, 26);
    cout << "Thanks for Your Precious Time. Press Any Key for
Exit...";
    setcolor(15);
    getch();
}

void printBuddy(char Buddy[3][4], int &BuddyX, int &BuddyY)
{
    setcolor(12);
    int PrintBX = BuddyX;
    int PrintBY = BuddyY;
    for (int i = 0; i < 3; i++)
    {
        gotoxy(PrintBX, PrintBY);
        for (int j = 0; j < 4; j++)
        {
            cout << Buddy[i][j];
        }
        cout << endl;
        PrintBY++;
    }
    setcolor(15);
}

void printBuddyLeft(char BuddyLeft[3][4], int &BuddyX, int &BuddyY)
{
    setcolor(12);
    int PrintBX = BuddyX;
    int PrintBY = BuddyY;
    for (int i = 0; i < 3; i++)
    {
        gotoxy(PrintBX, PrintBY);
        for (int j = 0; j < 4; j++)
        {
            cout << BuddyLeft[i][j];
        }
        cout << endl;
        PrintBY++;
    }
    setcolor(15);
}

void teleportBuddy(char Buddy[3][4], char BuddyLeft[3][4], int
&BuddyX, int &BuddyY)
{
    char wall = 219;
    char nextleft = getCharAtxy(BuddyX - 1, BuddyY);
    char nextleft1 = getCharAtxy(BuddyX - 1, BuddyY + 1);
    char nextleft2 = getCharAtxy(BuddyX - 1, BuddyY + 2);
    if (nextleft == wall && nextleft1 == wall && nextleft2 == wall)
    {
        eraseBuddy(BuddyX, BuddyY);
        BuddyX = 64;
        printBuddyLeft(BuddyLeft, BuddyX, BuddyY);
    }

    char nextright = getCharAtxy(BuddyX + 4, BuddyY);
    char nextright1 = getCharAtxy(BuddyX + 4, BuddyY + 1);
    char nextright2 = getCharAtxy(BuddyX + 4, BuddyY + 2);
    if (nextright == wall && nextright1 == wall && nextright2 ==
wall)
    {
        eraseBuddy(BuddyX, BuddyY);
        BuddyX = 3;
        printBuddy(Buddy, BuddyX, BuddyY);
    }
}

void eraseBuddy(int &BuddyX, int &BuddyY)
{
    setcolor(12);
    int EraseBX = BuddyX;
    int EraseBY = BuddyY;
    for (int i = 0; i < 3; i++)
    {
        gotoxy(EraseBX, EraseBY);
        for (int j = 0; j < 4; j++)
        {
            cout << ' ';
        }
        cout << endl;
        EraseBY++;
    }
    setcolor(15);
}

void moveBuddyDown(char Buddy[3][4], char BuddyLeft[3][4], int
&BuddyX, int &BuddyY, string &printDirection, int &score)
{
    char next = getCharAtxy(BuddyX, BuddyY + 3);
    char next1 = getCharAtxy(BuddyX + 1, BuddyY + 3);
    char next2 = getCharAtxy(BuddyX + 2, BuddyY + 3);
    char next3 = getCharAtxy(BuddyX + 3, BuddyY + 3);
    if (next == ' ' && next1 == ' ' && next2 == ' ' && next3 == ' ')
    {
        eraseBuddy(BuddyX, BuddyY);
        BuddyY++;
        if (printDirection == "right")
        {
            printBuddy(Buddy, BuddyX, BuddyY);
        }
        else
        {
            printBuddyLeft(BuddyLeft, BuddyX, BuddyY);
        }
    }

    else if (next == '$' || next1 == '$' || next2 == '$' || next3 ==
'$')
    {
        score = score + 10;
        eraseBuddy(BuddyX, BuddyY);
        BuddyY++;
        if (printDirection == "right")
        {
            printBuddy(Buddy, BuddyX, BuddyY);
        }
        else
        {
            printBuddyLeft(BuddyLeft, BuddyX, BuddyY);
        }
    }
}

void moveBuddyUp(char Buddy[3][4], char BuddyLeft[3][4], int &BuddyX,
int &BuddyY, string &printDirection, int &score, int &Buddyhealth)
{
    char next = getCharAtxy(BuddyX, BuddyY - 1);
    char next1 = getCharAtxy(BuddyX + 1, BuddyY - 1);
    char next2 = getCharAtxy(BuddyX + 2, BuddyY - 1);
    char next3 = getCharAtxy(BuddyX + 3, BuddyY - 1);
    if (next == ' ' && next1 == ' ' && next2 == ' ' && next3 == ' ')
    {
        eraseBuddy(BuddyX, BuddyY);
        BuddyY--;
        if (printDirection == "right")
        {
            printBuddy(Buddy, BuddyX, BuddyY);
        }
        else
        {
            printBuddyLeft(BuddyLeft, BuddyX, BuddyY);
        }
    }

    if (next == '$' || next1 == '$' || next2 == '$' || next3 == '$')
    {
        score = score + 10;
        eraseBuddy(BuddyX, BuddyY);
        BuddyY--;
        if (printDirection == "right")
        {
            printBuddy(Buddy, BuddyX, BuddyY);
        }
        else
        {
            printBuddyLeft(BuddyLeft, BuddyX, BuddyY);
        }
    }
    if (next == '.' || next1 == '.' || next2 == '.' || next3 == '.')
    {
        Buddyhealth = Buddyhealth - 10;
        eraseBuddy(BuddyX, BuddyY);
        BuddyY--;
        if (printDirection == "right")
        {
            printBuddy(Buddy, BuddyX, BuddyY);
        }
        else
        {
            printBuddyLeft(BuddyLeft, BuddyX, BuddyY);
        }
    }
}

void moveBuddyLeft(char BuddyLeft[3][4], int &BuddyX, int &BuddyY,
int &princessX, int &princessY, string &printDirection, int &check,
bool &game, bool &game2, int &score)
{
    printDirection = "left";
    char next = getCharAtxy(BuddyX - 1, BuddyY);
    char next1 = getCharAtxy(BuddyX - 1, BuddyY + 1);
    char next2 = getCharAtxy(BuddyX - 1, BuddyY + 2);
    if (next == ' ' && next1 == ' ' && next2 == ' ')
    {
        eraseBuddy(BuddyX, BuddyY);
        BuddyX--;
        printBuddyLeft(BuddyLeft, BuddyX, BuddyY);
    }
    if (next == '$' || next1 == '$' || next2 == '$')
    {
        score = score + 10;
        eraseBuddy(BuddyX, BuddyY);
        BuddyX--;
        printBuddyLeft(BuddyLeft, BuddyX, BuddyY);
    }
    if ((BuddyX - 1 == princessX + 2) && (BuddyY == princessY))
    {
        eraseBuddy(BuddyX, BuddyY);
        erasePrincess(princessX, princessY);
        complete(check, game, game2);
        game2 = false;
    }
}

void moveBuddyRight(char Buddy[3][4], int &BuddyX, int &BuddyY, int
&princessX, int &princessY, string &printDirection, int &check, bool
&game, bool &game2, int &score, int &Enemy1health, int &Enemy2health,
int &Anthealth)
{
    printDirection = "right";
    char next = getCharAtxy(BuddyX + 4, BuddyY);
    char next1 = getCharAtxy(BuddyX + 4, BuddyY + 1);
    char next2 = getCharAtxy(BuddyX + 4, BuddyY + 2);
    if (next == ' ' && next1 == ' ' && next2 == ' ')
    {
        eraseBuddy(BuddyX, BuddyY);
        BuddyX++;
        printBuddy(Buddy, BuddyX, BuddyY);
    }
    if (next == '$' || next1 == '$' || next2 == '$')
    {
        score = score + 10;
        eraseBuddy(BuddyX, BuddyY);
        BuddyX++;
        printBuddy(Buddy, BuddyX, BuddyY);
    }

    if ((next == '|' || next1 == '|' || next2 == '|') && Enemy1health
== -1 && Enemy2health == -1 && Anthealth == -1)
    {
        eraseBuddy(BuddyX, BuddyY);
        complete(check, game, game2);
    }
    if ((BuddyX + 3 == princessX - 1) && (BuddyY == princessY))
    {
        eraseBuddy(BuddyX, BuddyY);
        erasePrincess(princessX, princessY);
        complete(check, game, game2);
        game2 = false;
    }
}
```

```
void controlBuddy(char Buddy[3][4], char BuddyLeft[3][4], int
&BuddyX, int &BuddyY, int &princessX, int &princessY, string
&printDirection, bool &isJump, int &jumpTick, int bulletX[100], int
bulletY[100], char bulletDirection[100], int &bulletCount, int
&check, bool &game, bool &game2, int &score, int &Enemy1health, int
&Enemy2health, int &Anthealth, int &Buddyhealth)
{
    if (GetAsyncKeyState(VK_LEFT))
    {
        moveBuddyLeft(BuddyLeft, BuddyX, BuddyY, princessX,
princessY, printDirection, check, game, game2, score);
    }

    if (GetAsyncKeyState(VK_RIGHT))
    {
        moveBuddyRight(Buddy, BuddyX, BuddyY, princessX, princessY,
printDirection, check, game, game2, score, Enemy1health,
Enemy2health, Anthealth);
    }

    if (GetAsyncKeyState(VK_UP))
    {
        if (canJump(BuddyX, BuddyY))
        {
            isJump = true;
        }
    }
    if (isJump)
    {
        moveBuddyUp(Buddy, BuddyLeft, BuddyX, BuddyY, printDirection,
score, Buddyhealth);
        jumpTick += 1;
        if (jumpTick == 4)
        {
            isJump = false;
            jumpTick = 0;
        }
    }
    else
    {
        moveBuddyDown(Buddy, BuddyLeft, BuddyX, BuddyY,
printDirection, score);
    }

    if (GetAsyncKeyState(VK_SPACE))
    {
        createBullet(BuddyX, BuddyY, printDirection, bulletX,
bulletY, bulletDirection, bulletCount);
    }
}

bool canJump(int &BuddyX, int &BuddyY)
{
    char below1 = getCharAtxy(BuddyX, BuddyY + 3);
    char below2 = getCharAtxy(BuddyX + 1, BuddyY + 3);
    char below3 = getCharAtxy(BuddyX + 2, BuddyY + 3);
    char below4 = getCharAtxy(BuddyX + 3, BuddyY + 3);
    if ((below1 == '#' || below2 == '#' || below3 == '#' || below4 ==
'#'))
    {
        return true;
    }
    else
    {
        return false;
    }
}

void printEnemy1(char Enemy[3][5], int &enemy1X, int &enemy1Y)
{
    setcolor(10);
    int EnemyP1X = enemy1X;
    int EnemyP1Y = enemy1Y;
    for (int i = 0; i < 3; i++)
    {
        gotoxy(EnemyP1X, EnemyP1Y);
        for (int j = 0; j < 5; j++)
        {
            cout << Enemy[i][j];
        }
        cout << endl;
        EnemyP1Y++;
    }
    setcolor(15);
}

void eraseEnemy1(int &enemy1X, int &enemy1Y)
{
    setcolor(10);
    int EnemyP1X = enemy1X;
    int EnemyP1Y = enemy1Y;
    for (int i = 0; i < 3; i++)
    {
        gotoxy(EnemyP1X, EnemyP1Y);
        for (int j = 0; j < 5; j++)
        {
            cout << ' ';
        }
        cout << endl;
        EnemyP1Y++;
    }
    setcolor(15);
}

void moveEnemy1(char Enemy[3][5], int &enemy1X, int &enemy1Y, string
&enemy1direction, int &Enemy1health)
{
    if (Enemy1health > 0)
    {
        if (enemy1direction == "up")
        {
            char next = getCharAtxy(enemy1X, enemy1Y - 1);
            if (next == ' ')
            {
                eraseEnemy1(enemy1X, enemy1Y);
                enemy1Y--;
                printEnemy1(Enemy, enemy1X, enemy1Y);
            }
            if (next == '#')
            {
                enemy1direction = "down";
            }
        }
        if (enemy1direction == "down")
        {
            char next = getCharAtxy(enemy1X, enemy1Y + 3);
            if (next == ' ')
            {
                eraseEnemy1(enemy1X, enemy1Y);
                enemy1Y++;
                printEnemy1(Enemy, enemy1X, enemy1Y);
            }
            if (next == '#')
            {
                enemy1direction = "up";
            }
        }
    }
    if (Enemy1health == 0)
    {
        eraseEnemy1(enemy1X, enemy1Y);
        Enemy1health = -1;
        enemy1X = 0;
        enemy1Y = 0;
        gotoxy(75, 12);
        cout << "Botchan Health: KILL ";
    }
}

void printEnemy2(char Enemy[3][5], int &enemy2X, int &enemy2Y)
{
    setcolor(13);
    int EnemyP2X = enemy2X;
    int EnemyP2Y = enemy2Y;

    for (int i = 0; i < 3; i++)
    {
        gotoxy(EnemyP2X, EnemyP2Y);
        for (int j = 0; j < 5; j++)
        {
            cout << Enemy[i][j];
        }
        cout << endl;
        EnemyP2Y++;
    }
    setcolor(15);
}

void eraseEnemy2(int &enemy2X, int &enemy2Y)
{
    setcolor(10);
    int EnemyP2X = enemy2X;
    int EnemyP2Y = enemy2Y;
    for (int i = 0; i < 3; i++)
    {
        gotoxy(EnemyP2X, EnemyP2Y);
        for (int j = 0; j < 5; j++)
        {
            cout << ' ';
        }
        cout << endl;
        EnemyP2Y++;
    }
    setcolor(15);
```

```
}

void moveEnemy2(char Enemy[3][5], int &enemy2X, int &enemy2Y, string
&enemy2direction, int &Enemy2health)
{

    if (Enemy2health > 0)
    {
        if (enemy2direction == "left")
        {
            char next = getCharAtxy(enemy2X - 1, enemy2Y + 1);
            if (next == ' ')
            {
                eraseEnemy2(enemy2X, enemy2Y);
                enemy2X--;
                printEnemy2(Enemy, enemy2X, enemy2Y);
            }
            if (next == '#')
            {
                enemy2direction = "right";
            }
        }

        if (enemy2direction == "right")
        {
            char next = getCharAtxy(enemy2X + 5, enemy2Y + 1);
            if (next == ' ')
            {
                eraseEnemy2(enemy2X, enemy2Y);
                enemy2X++;
                printEnemy2(Enemy, enemy2X, enemy2Y);
            }
            if (next == '#')
            {
                enemy2direction = "left";
            }
        }
    }
    if (Enemy2health == 0)
    {
        eraseEnemy2(enemy2X, enemy2Y);
        Enemy2health = -1;
        enemy2X = 0;
        enemy2Y = 0;
        gotoxy(75, 13);
        cout << "Titchi Health: KILL ";
    }
}

void printAnt(char Ant[3][5], int &AntX, int &AntY)
{
    setcolor(06);
    int AntPX = AntX;
    int AntPY = AntY;

    for (int i = 0; i < 3; i++)
    {
        gotoxy(AntPX, AntPY);
        for (int j = 0; j < 5; j++)
        {
            cout << Ant[i][j];
        }
        cout << endl;
        AntPY++;
    }
    setcolor(15);
}

void eraseAnt(int &AntX, int &AntY)
{
    setcolor(06);
    int AntPX = AntX;
    int AntPY = AntY;

    for (int i = 0; i < 3; i++)
    {
        gotoxy(AntPX, AntPY);
        for (int j = 0; j < 5; j++)
        {
            cout << ' ';
        }
        cout << endl;
        AntPY++;
    }
    setcolor(15);
}

void AntBullet(int &BuddyX, int &BuddyY, int &AntX, int &AntY, int
AntBulletX[10], int AntBulletY[10], int &AntCount, int &Anthealth,
bool &AntFlag)
{
    if ((AntY == BuddyY) && AntFlag)
    {
        setcolor(06);
        AntBulletX[AntCount] = AntX - 1;
        AntBulletY[AntCount] = AntY + 1;
        gotoxy(AntBulletX[AntCount], AntBulletY[AntCount]);
        cout << ".";
        AntCount++;
        setcolor(15);
    }

    if (Anthealth == 0)
    {
        eraseAnt(AntX, AntY);
        Anthealth = -1;
        AntX = 0;
        AntY = 0;
        gotoxy(75, 14);
        cout << "Sadara Health: KILL ";
        AntFlag = false;
    }
}

void moveAntBullet(int AntBulletX[10], int AntBulletY[10], int
&AntCount)
{
    for (int x = 0; x < AntCount; x++)
    {
        char next = getCharAtxy(AntBulletX[x] - 1, AntBulletY[x]);
        char next1 = getCharAtxy(AntBulletX[x], AntBulletY[x] - 1);
        char head = 234;
        char hand = 178;
        char body = 155;
        char leg = '\\';
        if (next != ' ' || (next1 == head || next1 == hand || next1
== body))
        {
            eraseAntBullet(AntBulletX[x], AntBulletY[x]);
            deleteAntBullet(x, AntBulletX, AntBulletY, AntCount);
        }
        else
        {
            eraseAntBullet(AntBulletX[x], AntBulletY[x]);
            AntBulletX[x]--;
            printAntBullet(AntBulletX[x], AntBulletY[x]);
        }
    }
```

```
}

void printAntBullet(int x, int y)
{
    setcolor(06);
    gotoxy(x, y);
    cout << ".";
    setcolor(15);
}

void eraseAntBullet(int x, int y)
{
    gotoxy(x, y);
    cout << " ";
}

void deleteAntBullet(int index, int AntBulletX[10], int
AntBulletY[10], int &AntCount)
{
    int x = index;
    while (x < AntCount)
    {
        AntBulletX[x] = AntBulletX[x + 1];
        AntBulletY[x] = AntBulletY[x + 1];
        x++;
    }
    AntCount--;
}

void printPrincess(char princess[3][3], int &princessX, int
&princessY)
{
    int princessPX = princessX;
    int princessPY = princessY;
    setcolor(02);
    for (int i = 0; i < 3; i++)
    {
        gotoxy(princessPX, princessPY);
        for (int j = 0; j < 3; j++)
        {
            cout << princess[i][j];
        }
        cout << endl;
        princessPY++;
    }
    setcolor(15);
```

```
}

void erasePrincess(int &princessX, int &princessY)
{
    int princessPX = princessX;
    int princessPY = princessY;
    for (int i = 0; i < 3; i++)
    {
        gotoxy(princessPX, princessPY);
        for (int j = 0; j < 3; j++)
        {
            cout << ' ';
        }
        cout << endl;
        princessPY++;
    }
}

void movePrincessDown(char princess[3][3], int &princessX, int
&princessY)
{
    char next = getCharAtxy(princessX, princessY + 3);
    char next1 = getCharAtxy(princessX + 1, princessY + 3);
    char next2 = getCharAtxy(princessX + 2, princessY + 3);
    if (next == ' ' && next1 == ' ' && next2 == ' ')
    {
        erasePrincess(princessX, princessY);
        princessY++;
        printPrincess(princess, princessX, princessY);
    }
}

void jailOpen(int &jailX, int &jailY)
{
    int jailPX = jailX;
    int jailPY = jailY;
    while (jailPX < 34)
    {
        gotoxy(jailPX, jailPY);
        cout << ' ';
        jailPX++;
    }
}
```

```
void createBullet(int &BuddyX, int &BuddyY, string &printDirection,
int bulletX[100], int bulletY[100], char bulletDirection[100], int
&bulletCount)
{
    if (printDirection == "right")
    {
        Beep(2000, 5);
        char next = getCharAtxy(BuddyX + 4, BuddyY + 1);
        if (next == ' ')
        {
            bulletX[bulletCount] = BuddyX + 4;
            bulletY[bulletCount] = BuddyY + 1;
            bulletDirection[bulletCount] = 'R';
            gotoxy(BuddyX + 4, BuddyY + 1);
            setcolor(12);
            cout << "*";
            bulletCount++;
            setcolor(15);
        }
    }

    if (printDirection == "left")
    {
        Beep(2000, 5);
        char next = getCharAtxy(BuddyX - 1, BuddyY + 1);
        if (next == ' ')
        {
            bulletX[bulletCount] = BuddyX - 1;
            bulletY[bulletCount] = BuddyY + 1;
            bulletDirection[bulletCount] = 'L';
            gotoxy(BuddyX - 1, BuddyY + 1);
            setcolor(12);
            cout << "*";
            bulletCount++;
            setcolor(15);
        }
    }
}

void movebullet(int bulletX[100], int bulletY[100], char
bulletDirection[100], int &bulletCount)
{
    for (int x = 0; x < bulletCount; x++)
    {
        if (bulletDirection[x] == 'R')
        {
            char next = getCharAtxy(bulletX[x] + 1, bulletY[x]);
            if (next != ' ')
            {
                eraseBullet(bulletX[x], bulletY[x]);
                deleteBullet(x, bulletX, bulletY, bulletDirection, 
bulletCount);
            }
            else
            {
                eraseBullet(bulletX[x], bulletY[x]);
                bulletX[x]++;
                printBullet(bulletX[x], bulletY[x]);
            }
        }

        if (bulletDirection[x] == 'L')
        {
            char next = getCharAtxy(bulletX[x] - 1, bulletY[x]);
            if (next != ' ')
            {
                eraseBullet(bulletX[x], bulletY[x]);
                deleteBullet(x, bulletX, bulletY, bulletDirection, 
bulletCount);
            }
            else
            {
                eraseBullet(bulletX[x], bulletY[x]);
                bulletX[x]--;
                printBullet(bulletX[x], bulletY[x]);
            }
        }
    }
}

void deleteBullet(int index, int bulletX[100], int bulletY[100], char 
bulletDirection[100], int &bulletCount)
{
    int x = index;
    while (x < bulletCount)
    {
        bulletX[x] = bulletX[x + 1];
        bulletY[x] = bulletY[x + 1];
        bulletDirection[x] = bulletDirection[x + 1];
        x++;
    }
    bulletCount--;
```

```
}

void printBullet(int x, int y)
{
    setcolor(12);
    gotoxy(x, y);
    cout << "*";
    setcolor(15);
}

void eraseBullet(int x, int y)
{
    gotoxy(x, y);
    cout << " ";
}

void printkratos(char Kratos[3][7], int &KratosX, int &KratosY)
{
    int KratosPX = KratosX;
    int KratosPY = KratosY;
    for (int i = 0; i < 3; i++)
    {
        gotoxy(KratosPX, KratosPY);
        for (int j = 0; j < 7; j++)
        {
            cout << Kratos[i][j];
        }
        cout << endl;
        KratosPY++;
    }
}
void erasekratos(int &KratosX, int &KratosY)
{
    int KratosPX = KratosX;
    int KratosPY = KratosY;
    for (int i = 0; i < 3; i++)
    {
        gotoxy(KratosPX, KratosPY);
        for (int j = 0; j < 7; j++)
        {
            cout << ' ';
        }
        cout << endl;
        KratosPY++;
    }
}
```

```
void movekratos(char Kratos[3][7], int &BuddyX, int &BuddyY, int
&KratosX, int &KratosY, int &kratosHealth)
{
    char wall = 219;
    if (kratosHealth > 0)
    {

        bool canMove = true;
        int xDiff = BuddyX - KratosX;
        int xOffset = 0;
        int yOffset = 0;
        erasekratos(KratosX, KratosY);
        if (xDiff < 0)
        {
            xOffset = -1;
        }
        else
        {
            xOffset = 1;
        }
        char next;
        for (int i = 0; i < 7; i++)
        {
            next = getCharAtxy(KratosX + xOffset + i, KratosY +
yOffset);
            if (next == '#' && next == wall)
            {
                canMove = false;
                break;
            }
        }
        if (canMove)
        {
            KratosX += xOffset;
        }
        printkratos(Kratos, KratosX, KratosY);
    }
    if (kratosHealth == 0)
    {
        erasekratos(KratosX, KratosY);
        kratosHealth = -1;
        KratosX = 0;
        KratosY = 0;
        gotoxy(75, 15);
        cout << "Kratos Health: KILL ";
    }
```

```
}

void collision(int &Buddyhealth, int &enemy1X, int &enemy1Y, int
&enemy2X, int &enemy2Y, int &AntX, int &AntY, int &KratosX, int
&KratosY, int bulletX[100], int bulletY[100], char
bulletDirection[100], int &bulletCount, int &Enemy1health, int
&Enemy2health, int &Anthealth, int &kratosHealth, int &score)
{
    for (int x = 0; x < bulletCount; x++)
    {
        if (Enemy1health > 0)
        {
            if (bulletX[x] + 1 == enemy1X && (bulletY[x] == enemy1Y
|| bulletY[x] == enemy1Y + 1 || bulletY[x] == enemy1Y + 2))
            {
                eraseBullet(bulletX[x], bulletY[x]);
                Enemy1health = Enemy1health - 5;
                addScore(score);
                deleteBullet(x, bulletX, bulletY, bulletDirection,
bulletCount);
            }
        }

        if (Enemy2health > 0)
        {
            if ((bulletX[x] - 1 == enemy2X + 5 || bulletX[x] + 1 ==
enemy2X) && (bulletY[x] == enemy2Y || bulletY[x] == enemy2Y + 1 ||
bulletY[x] == enemy2Y + 2))
            {
                eraseBullet(bulletX[x], bulletY[x]);
                Enemy2health = Enemy2health - 5;
                addScore(score);
                deleteBullet(x, bulletX, bulletY, bulletDirection,
bulletCount);
            }
        }

        if (Anthealth > 0)
        {
            if (bulletX[x] + 1 == AntX && (bulletY[x] == AntY ||
bulletY[x] == AntY + 1 || bulletY[x] == AntY + 2))
            {
                eraseBullet(bulletX[x], bulletY[x]);
                Anthealth = Anthealth - 5;
                addScore(score);
                deleteBullet(x, bulletX, bulletY, bulletDirection, 
bulletCount);
            }
        }

        if (kratosHealth > 0)
        {
            if ((bulletX[x] - 1 == KratosX + 7 || bulletX[x] + 1 == 
KratosX) && (bulletY[x] == KratosY || bulletY[x] == KratosY + 1 || 
bulletY[x] == KratosY + 2))
            {
                eraseBullet(bulletX[x], bulletY[x]);
                kratosHealth = kratosHealth - 5;
                addScore(score);
                deleteBullet(x, bulletX, bulletY, bulletDirection, 
bulletCount);
            }
        }
    }
}

void printScore(int &score)
{
    gotoxy(75, 8);
    cout << "Score: " << score;
}

void addScore(int &score)
{
    score++;
}

void printBuddyHealth(int &Buddyhealth)
{
    gotoxy(75, 10);
    cout << "Your Health:    ";
    gotoxy(75, 10);
    cout << "Your Health: " << Buddyhealth;
}

void Printenemyhealth(bool &game, bool &game2, int &Enemy1health, int 
&Enemy2health, int &Anthealth, int &kratosHealth)
{
    if (Enemy1health > 0 && game == true)
    {
        gotoxy(75, 12);
        cout << "Botchan Health:    ";
        gotoxy(75, 12);
        cout << "Botchan Health: " << Enemy1health;
    }
    if (Enemy2health > 0 && game == true)
    {
        gotoxy(75, 13);
        cout << "Titchi Health:    ";
        gotoxy(75, 13);
        cout << "Titchi Health: " << Enemy2health;
    }

    if (Anthealth > 0 && game == true)
    {
        gotoxy(75, 14);
        cout << "Sadara Health:     ";
        gotoxy(75, 14);
        cout << "Sadara Health: " << Anthealth;
    }

    if (kratosHealth > 0 && (game2 == true && game == false))
    {
        gotoxy(75, 15);
        cout << "Kratos Health:     ";
        gotoxy(75, 15);
        cout << "Kratos Health: " << kratosHealth;
    }
}

void gameoverCollsion(int &BuddyX, int &BuddyY, int &Buddyhealth, int
&enemy1X, int &enemy1Y, int &enemy2X, int &enemy2Y, int &KratosX, int
&KratosY, int AntBulletX[10], int AntBulletY[10], int &AntCount, int
&Enemy1health, int &Enemy2health, int &kratosHealth)
{
    // enemy1
    if (Enemy1health > 0)
    {
        for (int i = -2; i < 3; i++) // right
        {
            if (BuddyX + 3 == enemy1X - 1 && BuddyY == enemy1Y + i)
            {
                Buddyhealth = 0;
            }
        }

        if (BuddyX + 3 == enemy1X - 1 && BuddyY == enemy1Y) // right
        {
            Buddyhealth = 0;
        }

        for (int i = -3; i < 8; i++) // enemy up
        {
            if (BuddyX == enemy1X + i && BuddyY - 1 == enemy1Y + 2)
            {
                Buddyhealth = 0;
            }
        }

        for (int i = -3; i < 8; i++) // Enemy down
        {
            if (BuddyX == enemy1X + i && BuddyY + 2 == enemy1Y - 1)
            {
                Buddyhealth = 0;
            }
        }
    }

    if (Enemy2health > 0)
    {
        // enemy2
        if (BuddyX + 3 == enemy2X - 1 && BuddyY == enemy2Y) // right
        {
            Buddyhealth = 0;
        }
        if (BuddyX - 1 == enemy2X + 4 && BuddyY == enemy2Y) // left
        {
            Buddyhealth = 0;
        }

        for (int i = -3; i < 8; i++) // enemy  up
        {
            if (BuddyX == enemy2X + i && BuddyY - 1 == enemy2Y + 2)
            {
                Buddyhealth = 0;
            }
        }

        for (int i = -3; i < 8; i++) // Enemy Down
        {
            if (BuddyX == enemy2X + i && BuddyY + 2 == enemy2Y - 1)
            {
                Buddyhealth = 0;
            }
        }
    }
    // ant
    for (int i = 0; i < AntCount; i++)
    {
        if (AntBulletX[i] == BuddyX + 4 && (AntBulletY[i] == BuddyY
|| AntBulletY[i] == BuddyY + 1 || AntBulletY[i] == BuddyY + 2))
        {
            Buddyhealth = Buddyhealth - 10;
            eraseAntBullet(AntBulletX[i], AntBulletY[i]);
        }
    }

    // kratos
    if (kratosHealth > 0)
    {
        // kratos
        if (BuddyX + 4 == KratosX - 1 && BuddyY == KratosY) // right
        {
            Buddyhealth = 0;
        }
        if (BuddyX - 1 == KratosX + 7 && BuddyY == KratosY) // left
        {
            Buddyhealth = 0;
        }
        for (int i = -3; i < 10; i++) // Enemy Down
        {
            if (BuddyX == KratosX + i && BuddyY + 2 == KratosY - 1)
            {
                Buddyhealth = 0;
            }
        }
    }
}

void gameover(int &Buddyhealth, bool &game, bool &game2)
{
    if (Buddyhealth == 0)
    {
        game = false;
        game2 = false;
        system("cls");
        setcolor(12);
        gotoxy(20, 15);
        cout << "GAME OVER";
        cout << endl;
        cout << endl;
        cout << endl;
        setcolor(15);
    }
}

void complete(int &check, bool &game, bool &game2)
{
    game = false;
    // game2 = false;
    system("cls");
    setcolor(12);
    gotoxy(20, 15);
    cout << "STAGE COMLETE :)";
    cout << endl;
    cout << endl;
    cout << endl;
    setcolor(15);
    // if (check == 0)
    // {
    //     game2 = true;
    //     check = 1;
    // }
}

void generateRandomCoin(int &count)
{
    setcolor(03);
    if (count < 5)
    {
        int x;
        x = 3 + rand() % 65;
        gotoxy(x, 20);
        cout << '$';
        count = count + 1;
    }
    setcolor(15);
}

void loadMaze1(char Maze1[26][70])
{
    fstream MazeOne;
    MazeOne.open("Maze1.txt", ios::in);
    string line;
    int row = 0;
    while (getline(MazeOne, line))
    {
        for (int i = 0; i < line.length(); i++)
        {
            Maze1[row][i] = line[i];
        }
        row++;
    }
    MazeOne.close();
}
void loadMaze2(char Maze2[26][71])
{
    fstream MazeTwo;
    MazeTwo.open("Maze2.txt", ios::in);
    string line;
    int row = 0;
    char wall = 219;
    while (getline(MazeTwo, line))
    {
        for (int i = 0; i < line.length(); i++)
        {
            if (line[i] == 'W')
            {
                Maze2[row][i] = wall;
            }
            else
            {
                Maze2[row][i] = line[i];
            }
        }
        row++;
    }
    MazeTwo.close();
}
void loadBuddy(char Buddy[3][4])
{
    char fireBuddyHead = 234;
    char fireBuddyBody = 178;
    char fireBuddyHand = 155;
    fstream rightBuddyFile;
    rightBuddyFile.open("BuddyRight.txt", ios::in);
    string line;
    int row = 0;
    while (getline(rightBuddyFile, line))
    {
        for (int i = 0; i < 4; i++)
        {
            if (line[i] == 'H')
            {
                Buddy[row][i] = fireBuddyHead;
            }
            else if (line[i] == 'B')
            {
                Buddy[row][i] = fireBuddyBody;
            }
            else if (line[i] == 'L')
            {
                Buddy[row][i] = fireBuddyHand;
            }
            else
            {
                Buddy[row][i] = line[i];
            }
        }
        row++;
    }
    rightBuddyFile.close();
}
void loadBuddyLeft(char BuddyLeft[3][4])
{
    char fireBuddyHead = 234;
    char fireBuddyBody = 178;
    char fireBuddyHand = 155;
    fstream leftBuddyFile;
    leftBuddyFile.open("BuddyLeft.txt", ios::in);
    string line;
    int row = 0;
    while (getline(leftBuddyFile, line))
    {
        for (int i = 0; i < 4; i++)
        {
            if (line[i] == 'H')
            {
                BuddyLeft[row][i] = fireBuddyHead;
            }
            else if (line[i] == 'B')
            {
                BuddyLeft[row][i] = fireBuddyBody;
            }
            else if (line[i] == 'L')
            {
                BuddyLeft[row][i] = fireBuddyHand;
            }
            else
            {
                BuddyLeft[row][i] = line[i];
            }
        }
        row++;
    }
    leftBuddyFile.close();
}
void loadPrincess(char princess[3][3])
{
    char head = 148;
    char center = 219;
    char rightHand = 191;
    char leftHand = 218;
    char legs = 19;
    fstream PrincessFile;
    PrincessFile.open("Princess.txt", ios::in);
    string line;
    int row = 0;
    while (getline(PrincessFile, line))
    {
        for (int i = 0; i < 3; i++)
        {
            if (line[i] == 'H')
            {
                princess[row][i] = head;
            }
            else if (line[i] == 'B')
            {
                princess[row][i] = center;
            }
            else if (line[i] == 'L')
            {
                princess[row][i] = legs;
            }
            else if (line[i] == 'r')
            {
                princess[row][i] = rightHand;
            }
            else if (line[i] == 'l')
            {
                princess[row][i] = leftHand;
            }
            else
            {
                princess[row][i] = line[i];
            }
        }
        row++;
    }
    PrincessFile.close();
}
void loadEnemy(char Enemy[3][5])
{
    fstream EnemyFile;
    EnemyFile.open("Enemy.txt", ios::in);
    string line;
    int row = 0;
    while (getline(EnemyFile, line))
    {
        for (int i = 0; i < 5; i++)
        {
            Enemy[row][i] = line[i];
        }
        row++;
    }
    EnemyFile.close();
}
void loadAnt(char Ant[3][5])
{
    fstream AntFile;
    AntFile.open("Ant.txt", ios::in);
    string line;
    int row = 0;
    while (getline(AntFile, line))
    {
        for (int i = 0; i < 5; i++)
        {
            Ant[row][i] = line[i];
        }
        row++;
    }
    AntFile.close();
}
void loadKratos(char Kratos[3][7])
{
    fstream KratosFile;
    KratosFile.open("Kratos.txt", ios::in);
    string line;
    int row = 0;
    while (getline(KratosFile, line))
    {
        for (int i = 0; i < 7; i++)
        {
            Kratos[row][i] = line[i];
        }
        row++;
    }
    KratosFile.close();
}
void loadLevel(char &Level)
{
    fstream LevelFile;
    LevelFile.open("Level.txt", ios::in);
    string line;
    getline(LevelFile, line);
    Level = line[0];
    LevelFile.close();
}
void writeLevel(char &Level, char number)
{
    fstream LevelFile;
    LevelFile.open("Level.txt", ios::out);
    LevelFile << number;
    LevelFile.close();
}
void gotoxy(int x, int y)
{
    COORD coordinates;
    coordinates.X = x;
    coordinates.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),
coordinates);
}

string setcolor(unsigned short color)
{
    HANDLE hcon = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hcon, color);
    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 : ' ';
}

void ShowConsoleCursor(bool showFlag)
{
    HANDLE out = GetStdHandle(STD_OUTPUT_HANDLE);

    CONSOLE_CURSOR_INFO cursorInfo;

    GetConsoleCursorInfo(out, &cursorInfo);
    cursorInfo.bVisible = showFlag; // set the cursor visibility
    SetConsoleCursorInfo(out, &cursorInfo);
}
```

Student Reg. No: 2022-CS-65 **Student Name:** Muhammad Wali Ahmad

	A-Extensive Evidence	**B-Convincing Evidence**	**C-Limited Evidence**	**D-No Evidence**
Documentation Formatting **Grade:**	All the documentation meets all the criteria.	Documentation is well formatted but some of the criteria is not fulfilled.	Documentation is required a lot of improvement.	Documentation is not Available
Documentation Formatting Criteria: In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented				
Documentation Contents **Grade:**	Documentation includes all of the criteria.	Documentation meet more than 80% of the criteria given.	Documentation meet more than 50% of the criteria.	When the documentation meet less than 50% of the criteria.
Documentation Contents Criteria: **Title** Page - **Table** of Contents - Project **Short Description and Story Writing of Game** - **Game Characters** Description - **Rules** & Interactions - **Goal** of the Game - **Screenshot** of the Game - **Data Structures** Used in the Game - **Functions** Prototype - **Full Code**				
Project Complexity **Grade:**	Project has at least 1 Player and 3 enemies. Proper use of gotoxy() function. Health system, Firing System and lives decreasing system.	Project complexity meet 80% criteria given in extensive evidence	Project complexity meet 50% criteria given in extensive evidence	Project complexity meet less than 50% criteria given in extensive evidence
Randomness **Grade:**	Objects are produced randomly in the game.	meet more than 80% of the criteria given.	meet more than 50% of the criteria given.	Objects are appearing in the same pattern
Code Style **Grade:**	All Code style criteria is followed	All code style criteria followed but some improvements required	lot of improvements required in coding style.	**Did not follow** code style,
Code Style Criteria: Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added.				
Code Documentation Mapping **Grade:**	Code and documentation is synchronized.	Code and documentation does not synchronized at **some** places	Code and documentation does not synchronized at **many** places	Code and documentation **does not** synchronized.
Idea Novelty and Creativity **Grade:**	Idea is unique of the game	Idea is merged by combining other different games	Same idea as a previous game	Could not implement the existing game idea.
Data Structure (2D Arrays) **Grade:**	Data structure is sufficient for the project requirements	Data Structure is sufficient but require improvement to meet project requirements.	Data structure is not sufficient and need a lot of improvement	Data Structure is not properly identified and declared.
File Handling **Grade:**	Game maze is loaded and the updated maze is stored in the file	Game maze is loaded and partial data is stored in the file.	Game maze is just loaded but the updated game configuration is not stored in the maze.	Project do not contain file handling
Modularity **Grade:**	Meet all Modularity criteria	Meet all Modularity criteria but at some places it is missing	Do not sufficiently meet the modularity criteria.	No modularity or very minimum modularity.
Modularity criteria: Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)- There is no global variable defined. Arrays and variables are passed as parameters to the functions. Functions exhibit single responsibility principle.				
Screen flickering **Grade:**	There is no Screen flickering.	Maze is not flickering but the characters are flickering at great speed	Flickering is done at lot of places	Screen is flickering at all places
Presentation and Demo **Grade:**	Presentation and Demo was 100% working	Presentation and Demo require some improvements	Presentation and Demo require a lot of improvements	Presentation was not ok and Demo was not working
Student Understanding with the Code. **Grade:**	Student has complete understanding how the code is working and knows the concept.	Student has good understand but some place he does not know the concepts	Student has a very little understand and lack the major concepts.	Student does not have any level of understanding of the code.

Checked By: ____

Student Reg. No: 2022-CS-65 **Student Name:** Muhammad Wali Ahmad

	A-Extensive Evidence	**B-Convincing Evidence**	**C-Limited Evidence**	**D-No Evidence**
Documentation Formatting **Grade:**	All the documentation meets all the criteria.	Documentation is well formatted but some of the criteria is not fulfilled.	Documentation is required a lot of improvement.	Documentation is not Available
Documentation Formatting Criteria: In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented				
Documentation Contents **Grade:**	Documentation includes all of the criteria.	Documentation meet more than 80% of the criteria given.	Documentation meet more than 50% of the criteria.	When the documentation meet less than 50% of the criteria.
Documentation Contents Criteria: **Title** Page - **Table** of Contents - Project **Short Description and Story Writing of Game** - **Game Characters** Description - **Rules** & Interactions - **Goal** of the Game - **Screenshot** of the Game - **Data Structures** Used in the Game - **Functions** Prototype - **Full Code**				
Project Complexity **Grade:**	Project has at least 1 Player and 3 enemies. Proper use of gotoxy() function. Health system, Firing System and lives decreasing system.	Project complexity meet 80% criteria given in extensive evidence	Project complexity meet 50% criteria given in extensive evidence	Project complexity meet less than 50% criteria given in extensive evidence
Randomness **Grade:**	Objects are produced randomly in the game.	meet more than 80% of the criteria given.	meet more than 50% of the criteria given.	Objects are appearing in the same pattern
Code Style **Grade:**	All Code style criteria is followed	All code style criteria followed but some improvements required	lot of improvements required in coding style.	**Did not follow** code style,
Code Style Criteria: Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added.				
Code Documentation Mapping **Grade:**	Code and documentation is synchronized.	Code and documentation does not synchronized at **some** places	Code and documentation does not synchronized at **many** places	Code and documentation **does not** synchronized.
Idea Novelty and Creativity **Grade:**	Idea is unique of the game	Idea is merged by combining other different games	Same idea as a previous game	Could not implement the existing game idea.
Data Structure (2D Arrays) **Grade:**	Data structure is sufficient for the project requirements	Data Structure is sufficient but require improvement to meet project requirements.	Data structure is not sufficient and need a lot of improvement	Data Structure is not properly identified and declared.
File Handling **Grade:**	Game maze is loaded and the updated maze is stored in the file	Game maze is loaded and partial data is stored in the file.	Game maze is just loaded but the updated game configuration is not stored in the maze.	Project do not contain file handling
Modularity **Grade:**	Meet all Modularity criteria	Meet all Modularity criteria but at some places it is missing	Do not sufficiently meet the modularity criteria.	No modularity or very minimum modularity.
Modularity criteria: Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)- There is no global variable defined. Arrays and variables are passed as parameters to the functions. Functions exhibit single responsibility principle.				
Screen flickering **Grade:**	There is no Screen flickering.	Maze is not flickering but the characters are flickering at great speed	Flickering is done at lot of places	Screen is flickering at all places
Presentation and Demo **Grade:**	Presentation and Demo was 100% working	Presentation and Demo require some improvements	Presentation and Demo require a lot of improvements	Presentation was not ok and Demo was not working
Student Understanding with the Code. **Grade:**	Student has complete understanding how the code is working and knows the concept.	Student has good understand but some place he does not know the concepts	Student has a very little understand and lack the major concepts.	Student does not have any level of understanding of the code.

Checked By: ____