

# INTO the Unknown



**Session 2023 - 2027**

**Submitted by:**

Muhammad Omer 2023-CS-68

**Supervised by:**

Muhammad Awais

**Course:**

CSC-102 Programming Fundamentals

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

## Here you can find the major parts of your Proposal documentation

- **Short Description of your project**

I wanted to create a game that is fun to play. I had this idea a two to three years back and now I finally have the skills to fulfill my project plans. In this game, you play as a ship lost in space and your goal is to survive as long as possible. You need to collect space energy to charge your cannons, dodge meteors, use your cannon to break through blockades.

- **Game Characters Description**

1. Ship :- this is you the player.
2. Meteoroids :- These are space rocks moving towards you. You must dodge them or ship will get destroyed.
3. Moon :- These are large space bodies
4. Blockades :- These are here to stop you from moving and you can destroy them by shooting.

- **Game Objects Description**

1. Space energy :- You can collect this energy to charge your cannons.
2. Laser residue :- You can also collect the residue left by your lasers to charge them up again.

- **Rules**

1. Survive for as long as possible
2. Dodge all rocks
3. Try to collect all space energy
4. Try to survive

- **Goal of the Game**

Survive For Long as possible.

- **Wireframes**



Figure 1: Start Screen

```
char randArrays[3][screen_h][screen_l]
```

## ● Function Prototypes

```
int random_function(int total_options);
```

```
void cursor_hide();
```

```
void clearConsole();
```

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

```
void print_border();
```

```
void print_Array(char screen[screen_h][screen_l]);
```

```
void moveDown(char screen[screen_h][screen_l], char buffer[screen_h][screen_l], char  
randArrays[3][screen_h][screen_l]);
```

```
void printhead();
```

```
void erase_player(char screen[screen_h][screen_l]);
```

```
void print_player(char screen[screen_h][screen_l]);
```

```
void move_player(char screen[screen_h][screen_l], int direction, bool input);
```

```
void check_player_input(char screen[screen_h][screen_l]);
```

```
void player_visuals();
```

```
char checkCollision(char screen[screen_h][screen_l], int x, int y);
```

```
bool scoreCollision(char screen[screen_h][screen_l], int x, int y);
```

```
void fire_laser(char screen[screen_h][screen_l], int& laserY, bool& laserActive);
```

```
void move_laser(char screen[screen_h][screen_l], int& laserY, bool& laserActive);
```

```
#include <iostream>
```

```
#include <windows.h>
```

```
#include <conio.h>
```

```
#include <ctime>
```

```
#include <cstdlib>
```

```
using namespace std;
```

```
// screen dimensions
const int screen_w=51;
const int screen_h=37;

// player variables
const int return_coordinates[2]= {25,25};
int player_coordinates[2]={25,25};
const int accel_factor=1;
const int deaccel_factor=2;
int player_speed=0;
char player=219;
const int laser_speed = 2;
int score = 0;
int canon_fuel = 0;
char score_fuel = 30;
bool laserActive = false;
char blocker_enemy = 219;

HANDLE color = GetStdHandle(STD_OUTPUT_HANDLE);

//general use functions
int random_function(int total_options);
void cursor_hide();
void clearConsole();
void gotoxy(int x, int y);

//printing functions
```

```
void print_border();

void print_Array(char screen[screen_h][screen_l]);


//screen movement

void moveDown(char screen[screen_h][screen_l], char buffer[screen_h][screen_l], char
randArrays[3][screen_h][screen_l]);

void printhead();


// player functions

void erase_player(char screen[screen_h][screen_l]);

void print_player(char screen[screen_h][screen_l]);

void move_player(char screen[screen_h][screen_l], int direction,bool input);

void check_player_input(char screen[screen_h][screen_l]);

void player_visuals();

char checkCollision(char screen[screen_h][screen_l], int x, int y);

bool scoreCollision(char screen[screen_h][screen_l], int x, int y);

void fire_laser(char screen[screen_h][screen_l], int& laserY, bool& laserActive);

void move_laser(char screen[screen_h][screen_l], int& laserY, bool& laserActive);


// ui functions

void testcases();


int main()
{
    system("cls");    //to clear screen
    cursor_hide ();    //to hide cursor
    printhead();    //print start screen
```

```
system("cls");
```

```
//this is used to show the game screen
```

```
char screen[screen_h][screen_l] = {"#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           6           #",
```

```
        "#           #",
```

```
        "#           #",
```

```
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#};
```

//this is used to store the next screen

```
char buffer[screen_h][screen_l] = {"#           "#,
```

```
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,  
"#           "#,
```



```
//this is used to show the game over screen
```

---

Muhammad Omer 2023-CS-68

CSC-102 Programming Fundamentals



```
"#           #",
"#           #",
"#           #",
"#           #",
"#           #",
"#           #",
"#           #",
"#           #"};
```

//this is used to store seed for random level generator

```
char randArrays[3][screen_h][screen_l] = {"#           6           #",
"#           6           #",
"# :=+:           6           #",
"# :***=-           6           #",
"#           6           #",
"#           6           #",
"#           6           #",
"#           %#:.....*: #",
"#           6           *:.*==:- :.* #",
"#   %#:.....*           -:.....-=:.*:=#",
"# .@@@%#*****           6           =:.*=:+#::=-:#",
"# =:.*=:+#::-99999999999999999999==:.....-:.....=#",
"#   -*:::-=:.*           6           %=-*:::-=:.*#",
"#           :##+=--=#. #",
"#           6           #",
"#           #",
"#           6           #",
"#           #",
```

---

Muhammad Omer 2023-CS-68

CSC-102 Programming Fundamentals

"#99999999999999999999:#\*\*+==%@@@\*%\*@@@@@@@@@@@@@@@@@@@",

```
"# 6
%#@#%*=+%@@@%@@@@@@@@@@@@@@@@@#",
```

```
"# 6
.:%#@%#@@@@%@@@@#@@@@@@@@@@@@@@@@@@#",
```

```
"# 6
=@@%#@@@@#@#@#@#@#@#@#@#@#@#",
```

```
"# 6
:%@@@@@@@@@@@@@@@@@@"
```

```
"# 6
@@@@@@@@@@@@@@@@@@@@@@"
```

```
"# 6 @@@%@@@@@@@@@@@@@@@@@#",
```



```
"#          6          #",
"#          6          #",
"#          6          #",
"#          6          #",
"#          6          #",
"#          6          #",
"#          6      :=+:   #",
"#          6      :***=-...  #",
"#          6      :***=-###:=+#",
"#          6      :***=-##:***=#",
"#          6      ..!@@@##::#",
"#          6      :=+:   #",
"#          6      :***=-   #",
"#          6          #",
"#          6          #",
"#          6          #",
"#          6          #"};
```

```
bool game_on=true;

int laserY = player_coordinates[0] - 1; // Initial position of the laser

while (game_on)
{
    clearConsole();

    check_player_input(screen);

    // Move and erase the laser if active
    if (laserActive)
```

```
{
    move_laser(screen, laserY, laserActive);
}

if (GetAsyncKeyState(VK_SPACE) && !laserActive && (canon_fuel/10) > 0)
{
    fire_laser(screen, laserY, laserActive);
    canon_fuel-=10; // Reduce canon fuel when firing
}
print_Array(screen);
player_visuals();
moveDown(screen, buffer, randArrays);
char collidedChar = checkCollision(screen, player_coordinates[0], player_coordinates[1]);

if (collidedChar != ' ' && collidedChar != '6' && collidedChar != '|')
{
    game_on = false;
}

if (scoreCollision(screen, player_coordinates[0], player_coordinates[1]))
{
    score += 1;
    canon_fuel += 1;
}

SetConsoleTextAttribute(color, 8);
testcases();
```



```
    Beep(100,80);
}

clearConsole();          //used to bring the cursor back to the start of the screen
print_Array(Gameover_screen); //used to print the game over screen
getch();
return 0;
}

//general use functions
int random_function(int total_options)    //used to generate random numbers
{
    int return_int;
    srand(time(0));
    return_int = rand() % total_options ;
    return return_int;
}

void cursor_hide()          //used to hide cursor
{
    /*
        For Removing Blinking Cursor on Screen
    */
    HANDLE hStdOut = NULL;
    CONSOLE_CURSOR_INFO curInfo;

    hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
    GetConsoleCursorInfo(hStdOut, &curInfo);
    curInfo.bVisible = FALSE;
```

```
    SetConsoleCursorInfo(hStdOut, &curInfo);
}

void clearConsole()          //used to clear screen in efficient manner
{
    COORD cursorPosition;

    cursorPosition.X = 0;

    cursorPosition.Y = 0;

    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), cursorPosition);
}

void gotoxy(int x, int y)    //used to go to a specific coordinate
{
    static HANDLE h = NULL;

    if(!h)

        h = GetStdHandle(STD_OUTPUT_HANDLE);

    COORD c = { x, y };

    SetConsoleCursorPosition(h,c);
}


void printhead()            //used to print start screen
{
    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 <<"PRESS ANY KEY TO START PLAYING";
getch();
system("cls");
cout <<"CONTROLS"<<endl;
cout <<"MOVE LEFT :- left arrow key"<<endl;
cout <<"MOVE RIGHT :- right arrow key"<<endl;
cout <<"FIRE    :- up arrow key"<<endl;
cout <<endl;
cout <<"Things to note:-";
cout <<endl;
cout <<"Need 10 space energy to fire cannon"<<endl;
cout <<"You can collect residue of laser canon to charge canon again"<<endl;
cout <<"You can break blockkades using your cannon"<<endl;
getch();
system("cls");
}

//Printing functions
void print_border()          //used to print borders
{
```

```
    cout<<"#####"<<endl;
}

void print_Array(char screen[screen_h][screen_l]) //used to print arrays
{
    string temp = "";
    print_border();
    for (int i = 0; i < screen_h; ++i)
    {
        for (int j = 0; j < screen_l; ++j)
        {
            if (screen[i][j]=='6')
            {
                temp += score_fuel;
            }
            else if (screen[i][j]=='9')
            {
                temp += blocker_enemy;
            }

            else {temp += screen[i][j];}
        }
        temp += "\n";
    }
    cout << temp;
    print_player(screen);
    print_border();
}
```

```
//Screen movement

void moveDown(char screen[screen_h][screen_l], char buffer[screen_h][screen_l], char
randArrays[3][screen_h][screen_l]) //used to move screen down
{
    static int count = 0; // Counter to track the number of times moveDown is called

    // Move each column one step down
    for (int j = 0; j < screen_l; ++j) {
        // Shift elements down in the buffer
        for (int i = screen_h - 1; i > 0; --i) {
            buffer[i][j] = buffer[i - 1][j];
        }
        // Move the last row from screen to the top of the buffer
        // buffer[0][j] = screen[screen_h - 1][j];

        // Shift elements down in the screen
        for (int i = screen_h - 1; i > 0; --i) {
            screen[i][j] = screen[i - 1][j];
        }
        // Move the last row to the top
        screen[0][j] = buffer[screen_h - 1][j];
    }

    // Check if all elements in the buffer have moved down
    if (++count % screen_h == 0) {
        // Reset the counter
    }
}
```

```
count = 0;

// Randomly select an array from randArrays and move it to the buffer
int randIndex = random_function(3);
for (int i = 0; i < screen_h; ++i) {
    for (int j = 0; j < screen_l; ++j) {
        buffer[i][j] = randArrays[randIndex][i][j];
    }
}
}

//player functions
void check_player_input(char screen[screen_h][screen_l]) //used to see if the user presses any
keys
{
    bool input;
    if(GetAsyncKeyState(VK_LEFT))
    {
        input = true;
        move_player(screen,-1,input);
    }
    else if(GetAsyncKeyState(VK_RIGHT))
    {
        input = true;
        move_player(screen,1,input);
    }
}
```

```
    else
    {
        input = false;
        move_player(screen,0,input);
    }
}

void erase_player(char screen[screen_h][screen_l])    //used to erase player
{
    screen[player_coordinates[0]][player_coordinates[1]] = ' ';
}

void print_player(char screen[screen_h][screen_l])    //used to print player
{
    screen[player_coordinates[0]][player_coordinates[1]] = '*';
}

void move_player(char screen[screen_h][screen_l], int direction, bool input)    //used to move
player
{
    // Erase the player from the current position
    erase_player(screen);

    if (input)
    {
        player_speed += static_cast<int>(direction * accel_factor);
        // Limit the player_speed to a maximum value of 5
        player_speed = min(player_speed, 5);
        player_coordinates[1] += player_speed;
    }
}
```

```
else
{
    // If there is no input, move the player towards return_coordinates[1]
    if (player_coordinates[1] < return_coordinates[1])
    {
        direction = 1;
    }
    else if (player_coordinates[1] > return_coordinates[1])
    {
        direction = -1;
    }

    player_speed += static_cast<int>(direction / deaccel_fator);
    // Limit the player_speed to a maximum value of 5
    player_speed = min(player_speed, 5);
    player_coordinates[1] += player_speed;
}

// Check for boundaries to prevent the player from going off the screen
if (player_coordinates[1] < 1)
{
    player_coordinates[1] = 1;
    player_speed = 0;
}
else if (player_coordinates[1] >= screen_l - 3)
{
    player_coordinates[1] = screen_l - 3;
```



```
    player_speed = 0;
}

print_player(screen);
}

void fire_laser(char screen[screen_h][screen_l], int& laserY, bool& laserActive)//used to fire
cannon
{
    laserY = player_coordinates[0] - 1; // Set the initial position of the laser
    laserActive = true; // Activate the laser
}

void move_laser(char screen[screen_h][screen_l], int& laserY, bool& laserActive)//used to
move bullet
{
    // Erase the current position of the laser
    screen[laserY][player_coordinates[1]] = ' ';
    screen[laserY - 1][player_coordinates[1]] = ' ';

    // Move the laser up
    laserY -= laser_speed;

    // Check if the laser hits an obstacle or goes out of bounds
    if (laserY < 0 || (screen[laserY][player_coordinates[1]] != ' ' &&
screen[laserY][player_coordinates[1]] != '6'))
    {
        if (screen[laserY][player_coordinates[1]] == '9' ||
            screen[laserY + 1][player_coordinates[1]] == '9' ||
            screen[laserY - 1][player_coordinates[1]] == '9' ||
```

```
        screen[laserY + 1][player_coordinates[1]] == '9' ||
        screen[laserY - 2][player_coordinates[1]] == '9')
    {
        // Open a space of 5 units
        screen[laserY][player_coordinates[1]] = ' ';
        screen[laserY][player_coordinates[1] - 1] = ' ';
        screen[laserY][player_coordinates[1] + 1] = ' ';
        screen[laserY][player_coordinates[1] - 2] = ' ';
        screen[laserY][player_coordinates[1] + 2] = ' ';
        laserActive = false;
    }
    laserActive = false; // Deactivate the laser
}
else
{
    // Print the laser at its new position
    screen[laserY][player_coordinates[1]] = '|';
    screen[laserY - 1][player_coordinates[1]] = '|';
}
}

void player_visuals()
{
    gotoxy(player_coordinates[1]-1,player_coordinates[0]);
    SetConsoleTextAttribute(color, 15);
    cout<<char(220)<<char(219)<<char(220);
    gotoxy(player_coordinates[1],player_coordinates[0]-1);
    cout<<char(206);
```

```
}

// Collision detection function

char checkCollision(char screen[screen_h][screen_l], int x, int y) //used to check collisions
{
    // Checking if the player collides with any character at position (x, y)
    if (screen[x][y] != ' ') {
        return screen[x][y];
    }

    // If no collision, return a space character
    return ' ';
}

bool scoreCollision(char screen[screen_h][screen_l], int x, int y) //used to check score collisions
{
    if ((screen[x][y] == '6' || screen[x][y-1] == '6' || screen[x][y+1] == '6') || (screen[x][y] == '|' ||
screen[x][y-1] == '|' || screen[x][y+1] == '|'))
    {
        screen[x][y] = ' ';
        screen[x][y-1] = ' ';
        screen[x][y+1] = ' ';
        return true ;
    }
    else return false;
}

// ui functions

void testcases( )
{
```

```
gotoxy(screen_l+2,0);  
cout<<"Score:- "<<score;  
gotoxy(screen_l+2,1);  
cout<<"canon fuel "<<canon_fuel;  
}
```