**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 backs and now I finally have the skills to fulfill my project plans.In this game u 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 u the player.
2. Meterors :- These are space rocks moving towards u. You must dodge them or ship will get destroyed.
3. Moon :- These are large space bodies
4. Blockades:- These are here to stop u from moving and u can destroy them by shooting.

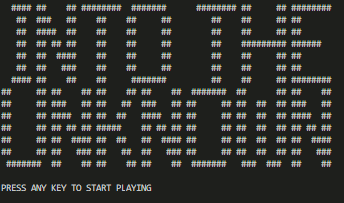
* **Game Objects Description**

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

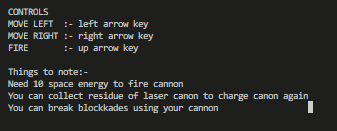
* **Rules** 
  1. Survive for as long as possible
  2. Dodge all rocks
  3. Try to collect all space energy
  4. Try to survive
* **Goal of the Game**

Survive For Long as possible.

* **Wireframes**

****

**Figure 1: Start Screen**

****

**Figure 2: Controls menu**

****

**Figure 3: Game Play**

**Data Structures (Parallel Arrays)**

char screen[screen\_h][screen\_l]

char buffer[screen\_h][screen\_l]

char Gameover\_screen[screen\_h][screen\_l]

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 printheader();

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\_l=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\_fator=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 printheader();

// 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

printheader(); //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

char Gameover\_screen[screen\_h][screen\_l] = {"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# GAME OVER #",

"# #",

"# #",

"# #",

"# #",

"# press any key to continue #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #",

"# #"};

//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 #",

"# #",

"# 6 #",

"# 6 #",

"# #",

"# +\*#\*\*+=--. 6 #",

"# @##@%\*\*\*\*=-- #",

"# \*@@%\*@%\*\*\*\*=- #",

"#99999999999999.@@@%#\*\*\*\*\*\*+=: 6 #",

"# :@@@@@@%\*\*@%\*\* #",

"# %@@@@@@@@@@@\*= #",

"# .%@@@@@@%@# 6 #",

"# #",

"# 6 #",

"# 6 #",

"# 6 #",

"# 6 #",

"# 6 #",

"# 6 #",

"# 6 #"},

{"# 6 %@@@@@#",

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

"# 6 +%@%##@@@@@@@#",

"# 6 :#%@#\*%@@@#@@@@@#",

"# 6 :%:#\*\*###@@@@@@@@@@#",

"# 6 .\*\*=+\*%@@@@=\*@@@@@@@@#",

"# 6 :====\*=@@@@##@@@@@@@@@#",

"# 6 :\*==+#\*-=@@@@@=%@@@@@@@#",

"# 6 --+@@%\*++@#@@#\*#@@@@@@@@#",

"# 6 :=@@%@@@@@@%\*#@##@@@@@@@@@#",

"# 6 -#@%\*##@@@#@%@%@@@@@@@%@@@#",

"# 6 ::%##@\*#@@%\*@@@\*%@@@@@@@@@@#",

"# 6 :-= =#@\*\*%##@%\*#@@@@%@@@@@@@#",

"# 6 ::--=\*#@%@###%-#@@@@%=@@@@@@@#",

"# 6 :: --+%\*#@%\*@%\*\*@@@@\*\*@@@@@@@#",

"# 6 +-:::=%\*@##@@@%\*#@@@@@@@@@@@@#",

"# 6 :\*+=#\*:=%@%#@@@%@\*@@@@@@@@@@@@#",

"#999999999999999999:#\*\*+==%@@@\*%\*@@@@@@@@@@@@@@@@#",

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

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

"# 6 +@+%##\*#@@@@@@@@@%#@@@@@@@@@@#",

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

"# 6 @@%@\*%@@@#@@%@@@@@@@@@@@@@@@#",

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

"# 6 :%@@@@@@@@@@@@@@@@@@@@@@@@#",

"# 6 :%@@@@@@@@@@@@@@@@@@@@@@@#",

"# 6 @@@@@@@@@@@@@@@@@@@@@@@@#",

"# 6 @@@@@@@@@@@@@@@@@@@@@@@#",

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

"# 6 @@@@@@@@@@@@@@@@@@@@#",

"# 6 =@@@@@@@@@@@@@@@@@@#",

"# 6 .=%@@@@@@@@@@@@@@#",

"# 6 -@@@@@@@@@@@#",

"# 6 .#%@#",

"# 6 #",

"# 6 #",

"# 6 #"},

{"# 6 #",

"# 6 #",

"# :===. 6 #",

"# :#######+= 6 #",

"# =\*\*###\*+======: 6 #",

"# =============\*\*++====: 6 #",

"# :=====\*##\*+=============: 6 #",

"#.=====\*####\*============== 6 #",

"#=+\*====+++================- 6 #",

"#=\*+==============\*\*=====\*== 6 #",

"#=+==========+\*\*===+====\*\*==999999999999999999999#",

"#===+###+=====\*\*=========== 6 #",

"# ===\*##\*================= 6 #",

"# :=========+\*====\*#+==: 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 efficent 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 printheader() //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 toprint 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;

}