**NATIONAL PUBLIC SCHOOL**

**KORAMANGALA**

**GALAGA**



*ADITYA NAIDU (XII A)*

*RAJAT AGARWAL (XII A)  
AYUSH AGRAWAL (XII A)*

ACKNOWLEDGEMENTS

We are deeply indebted to Chandita Ma’am and Kavitha Ma’am for their invaluable inputs on helping us to decide the right project and sketching out the program flow and content.

We would also like to thank Jyotsna Ma’am for giving us the opportunity to hone our skills.

INDEX

1. ACKNOWLEDGEMENTS
2. INTRODUCTION
3. GAME OVERVIEW
4. HEADER FILES
5. CLASSES AND OBJECTS
6. FUNCTIONS
7. STRUCTURES
8. SOURCE CODE
9. OUTPUT SCREENS
10. BIBLIOGRAPHY AND REFERENCES

INTRODUCTION

Our game is a reconceptualization of the classic ‘80s arcade game, Galaga. The user plays as Solo – a pilot aboard the Millenium Falcon on his way to the ruined planet Jakku. Gather resources, avoid asteroids and navigate your way across the galaxy to deliver your consignment to Jabba the Hutt.

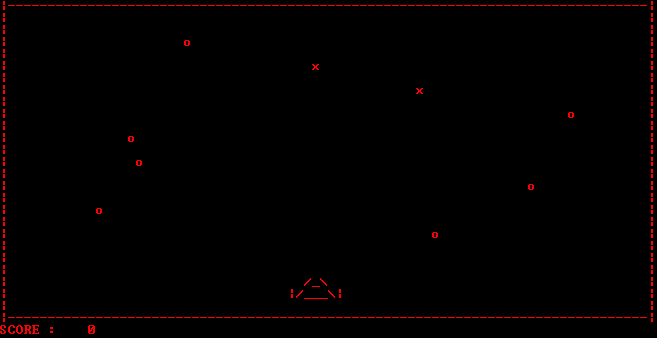
GAME OVERVIEW

Enter 1, 2 or 3 to choose an option to proceed.

Input:

* ‘1’ to play the game
* ‘2’ to display high scores
* ‘3’ to display instructions

The game screen consists of falling asteroids (‘x’) and coins (‘o’) and a spacecraft at the bottom which shoots out bullets.



The user can use the ‘a’ and ‘d’ keys to move left and right respectively.

Use the spacebar to shoot a single bullet directly above the spacecraft.

The objective is to collect as many coins (‘o’) as possible by shooting them while avoiding or destroying all asteroids (‘x’). Your score is determined by the number of coins collected.

<High score screen>

**HEADER FILES**

1. iostream
2. fstream
3. conio.h
4. stdio.h
5. string.h
6. windows.h
7. string
8. math.h
9. process.h
10. sstream
11. iomanip
12. ctime

**CLASSES AND OBJECTS**

Space:

## Data Members:

1. area[rows][cols]; Stores the game screen values

2. spacec[2][7]; Stores the spacecraft structure

3. static char area[rows][cols];

4. static char spacec[2][7];

## Methods:

1. void Move(char, Pos&, int, int);

2. void emove(Pos &pos, int h, int w);

3. static void DisplaySpace();

# Spacecraft: (Inherits publicly from Space)

## Data Members:

1. Pos pos; stores position of the spaecraft

2. int height, width; stores dimensions of the spacecraft

## Methods:

1. void Control(char); describes the movement of the spacecraft

2. void Shoot(); describes how the bullets are shot

3. void bullet\_c(); describes the structure of the bullets

# Enemy: (Inherits publicly from Spacecraft)

## Methods:

1. void disp\_enemy(int cole); displays the enemy

2. void control(); describes the movement of the enemies

## Global Functions

1. void sortsc() sorts the scores

2. void sortsc() writes the scores

3. void dispsc() displays all the high scores

4. void initfile() initializes all scores to 0

5. void dinst() displays instructions

6. void init() initializes all global variables

7. void gotoxy modifies the coordinates of enemies, spacecraft

**SOURCE CODE**

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* PROJECT NAME : GALAGA AKA SPACE INVADERS \*

\* PROJECT MONTH: MARCH \*

\* AUTHORS : AYUSH AGRAWAL , RAJAT AGARWAL , ADITYA NAIDU \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

//-------------------------------------------------------------------------------------------------------------------------------

#include <iostream>

#include <stdio.h>

#include <string.h>

#include <fstream>

#include <math.h>

#include <windows.h>

#include <conio.h>

#include <string>

#include <process.h>

#include <sstream>

#include <iomanip>

#include <ctime>

//--------------------------------------------------------------------------------------------------------------------------------

#define rows 25

#define cols 80

//--------------------------------------------------------------------------------------------------------------------------------

using namespace std;

//----------------------------------------------------------------

//GLOBALS

char b[20];

int temp\_score=0;

int fl=0;

int srow=22;

int scol=37;

int srow1=10;

int scol1=10;

int count =0;

int erow=srow+2;

int erow1=srow1+2;

int ecol=scol+5;

int score=0;

int score\_check=-1;

int ecol1=scol1+5;

//--------------------------------------------------------------------------------------------------------------------------------

void fscore(int c,string k) // performs operations with scores

{

ifstream infiles;

ifstream infilen;

infiles.open("score.txt");

infilen.open("name.txt");

//error handling

if(infiles.fail())

{

cerr <<"Error Opening File"<<endl;

exit(1);

}

int a[20];

string s[20];

for(int i=0;i<10;i++)

{

infiles>>a[i];

infilen>>s[i];

}

infiles.close();

infilen.close();

ofstream outfiles;

ofstream outfilen;

outfiles.open("score.txt");

outfilen.open("name.txt");

//error handling

if(outfiles.fail())

{

cerr <<"Error Opening File"<<endl;

exit(1);

}

for(int j=0;j<10;j++)

{

if(c>a[j])

{ for(int o=10;o>j;o--)

{

a[o]=a[o-1];

s[o]=s[o-1];

}

a[j]=c;

s[j]=k;

break;}

}

for(int l=0;l<10;l++)

{

outfiles<<a[l]<<endl;

outfilen<<s[l]<<endl;

}

outfiles.close();

outfilen.close();

}

//--------------------------------------------------------------------------------------------------------------------------------

void dscore() //display scores

{

system("cls");

cout<<"\t\t\t \_\_\_ \_\_\_ .\_\_ .\_\_ \_\_\_\_\_\_\_\_\_ "<<endl;

cout<<"\t\t\t / | \\|\_\_| \_\_\_\_ | |\_\_ / \_\_\_\_\_/ \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ \_\_\_\_\_\_"<<endl;

cout<<"\t\t\t/ ~ \\ |/ \_\_\_\\| | \\ \\\_\_\_\_\_ \\\_/ \_\_\_\\/ \_ \\\_ \_\_ \\\_/ \_\_ \\ / \_\_\_/"<<endl;

cout<<"\t\t\t\\ Y / / /\_/ > Y \\ / \\ \\\_\_( <\_> ) | \\/\\ \_\_\_/ \\\_\_\_ \\ "<<endl;

cout<<"\t\t\t \\\_\_\_|\_ /|\_\_\\\_\_\_ /|\_\_\_| / /\_\_\_\_\_\_\_ /\\\_\_\_ >\_\_\_\_/|\_\_| \\\_\_\_ >\_\_\_\_ >"<<endl;

cout<<"\t\t\t \\/ /\_\_\_\_\_/ \\/ \\/ \\/ \\/ \\/ "<<endl<<endl<<endl<<endl<<endl<<endl;

ifstream infiles;

ifstream infilen;

infiles.open("score.txt");

infilen.open("name.txt");

//error handling

if(infiles.fail())

{

cerr <<"Error Opening File"<<endl;

exit(1);

}

int a[10];

string s[10];

for(int i=0;i<10;i++)

{

infiles>>a[i];

infilen>>s[i];

}

cout.left;

for(int j=0;j<10;j++)

{

if(j!=9)

cout<<"\t\t\t\t"<<j+1<<". "<<left<<setw(20)<<s[j]<<a[j]<<endl;

else

cout<<"\t\t\t\t"<<j+1<<". "<<left<<setw(20)<<s[j]<<a[j]<<endl;}

infiles.close();

infilen.close();

getch();

}

//--------------------------------------------------------------------------------------------------------------------------------

void dinst()

{

ifstream infile;

infile.open("inst.txt");

//error handling

if(infile.fail())

{

cerr <<"Error Opening File"<<endl;

exit(1);

}

string s;

system("cls");

while (getline(infile,s))

{

cout<<s<<endl;

}

getch();

}

//--------------------------------------------------------------------------------------------------------------------------------

SYSTEMTIME st;

//--------------------------------------------------------------------------------------------------------------------------------

void init(){

srow=22;

scol=37;

srow1=10;

scol1=10;

count =0;

erow=srow+2;

erow1=srow1+2;

ecol=scol+5;

score=0;

score\_check=-1;

ecol1=scol1+5;

}

//--------------------------------------------------------------------------------------------------------------------------------

int main();

//--------------------------------------------------------------------------------------------------------------------------------

void gotoxy( int line, int column )

{

COORD coord;

coord.X = column;

coord.Y = line;

SetConsoleCursorPosition(

GetStdHandle( STD\_OUTPUT\_HANDLE ),

coord

);

}

//----------------------------------------------------------------

struct Pos

{

int i, j;

};

//----------------------------------------------------------------

class Space{

static char ReturnChar(int);

protected:

static char area[rows][cols];

static char spacec[2][7];

void Move(char, Pos&, int, int);

void emove(Pos &pos, int h, int w);

public:

static void DisplaySpace();

Space()

{

for(int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

area[i][j] = ' ';

}

}

}

};

char Space::area[rows][cols] = {' '};

char Space::spacec[2][7] = {' '};

//shifts the values by one

void Space::Move(char ch, Pos &pos, int h, int w)

{

gotoxy(erow,ecol+3);

//cout<<"t";

cout<<"\b\b\b\b\b\b\b\b\b";

cout<<" ";

//cout<<"\b\b\b\b\b\b\b\b";

gotoxy(erow-1,ecol+1);

cout<<"\b\b\b\b\b";

cout<<" ";

//cout<<"\b\b\b\b";

//left - a

if (ch == 'a' && pos.j >= 0)

{

scol-=1;

ecol=scol+3;

for(int r = pos.i; r < pos.i + h; r++)

{

for (int c = pos.j; c <= pos.j+w; c++)

{

if(area[r][c+1]!='o' && area[r][c]!='o' && area[r][c+1]!='x' && area[r][c]!='x')

area[r][c] = area[r][c+1];

//area[22][c-1]=area[22][c];

}

}

pos.j--;

}

//right - d

else if (ch == 'd' && pos.j + w + 1 < cols)

{

scol+=1;

ecol=scol+3;

for(int r = pos.i; r < pos.i+h; r++)

{

for (int c = pos.j+w+1; c >= pos.j; c--)

{

if(area[r][c-1]!='o' && area[r][c]!='o' && area[r][c+1]!='x' && area[r][c]!='x')

area[r][c] = area[r][c-1];

}

}

pos.j++;

}

gotoxy(srow+1,scol-1);

for (int i = 0; i < 2; i++)

{

count=0;

for (int j = 0; j < 7; j++)

{

cout << spacec[i][j];

count++;

if(count == 7)

{

gotoxy(erow,ecol-4);

//for(int sp=1;sp<=scol;sp++)

// cout<<" ";

}

}

}

}

//--------------------------------------------------------------------------------------------------------------------------------

void Space::DisplaySpace()

{

cout << "|";

for(int i = 0; i < cols; i++)

cout << "-";

cout << "|\n";

for (int i = 0; i < rows; i++)

{

cout << "|";

for (int j = 0; j < cols; j++)

{

cout << area[i][j];

}

cout << "|\n";

}

cout << "|";

for(int i = 0; i < cols; i++)

cout << "-";

cout << "|\n";

}

class Spacecraft : public Space

{

public:

void Control(char);

void Shoot();

void bullet\_c();

Pos pos;

int height, width;

Spacecraft()

{

height = 2;

width = 7;

pos = {22, 34};

spacec[0][2] = '/';

spacec[1][1] = '/'; // forward slash - '/'

spacec[0][4] = '\\'; // backward slash - '\'

spacec[1][5] = '\\';

spacec[1][2] = '\_'; // underscore - '\_'

spacec[1][3] = '\_';

spacec[1][4] = '\_';

spacec[0][3] = '\_';

spacec[1][0] = '|'; // vertical bar - '|'

spacec[1][6] = '|';

area[22][37] = '/';

area[23][36] = '/'; // forward slash - '/'

area[22][39] = '\\'; // backward slash - '\'

area[23][40] = '\\';

area[23][37] = '\_'; // underscore - '\_'

area[23][38] = '\_';

area[23][39] = '\_';

area[22][38] = '\_';

area[23][35] = '|'; // vertical bar - '|'

area[23][41] = '|';

}

};

void Spacecraft::Control(char ch)

{

if(ch == 'a' || ch == 'd')

{

Move(ch, pos, height, width);

}

else if(ch == ' ')

Shoot();

}

void Spacecraft::Shoot()

{

area[srow-2][scol+2]='|';

gotoxy(srow-2,scol+2);

cout<<"|";

}

void Spacecraft::bullet\_c()

{

for(int cher=1;cher<=22;cher++)

{

for(int chec=1;chec<=79;chec++)

{

if(area[cher][chec]=='|')

{

gotoxy(cher,chec+2);

cout<<"\b\b\b\b";

cout<<" ";

if(area[cher-1][chec]==' ')

{

gotoxy(cher-1,chec);

if(cher-1!=0)

cout<<"|";

area[cher][chec]=' ';

area[cher-1][chec]='|';

}

area[cher][chec]=' ';

//area[cher-1][chec]='|';

}

}

}

}

//----------------------------------------------------------------

class enemy : public Spacecraft

{

public:

void disp\_enemy(int cole)

{

int type=rand()%100;

if(type%5!=0)

{

area[2][cole]='o';

gotoxy(2,cole);

cout<<"o";

}

if(type%5==0)

{

area[2][cole]='x';

gotoxy(2,cole);

cout<<"x";

}

}

void control();

};

void enemy::control()

{

for(int cher=rows;cher>0;cher--)

{

for(int chec=cols;chec>0;chec--)

{

if(area[cher][chec]=='o' && cher<=20)

{

gotoxy(cher,chec+2);

cout<<"\b\b";

cout<<" ";

gotoxy(cher+1,chec);

area[cher][chec]=' ';

if((area[cher+1][chec]=='|' || area[cher+2][chec]=='|' /\*|| area[cher+3][chec]=='|'\*/ ))

{

if(score\_check==-1)score\_check=0;

}

if(cher+1!=20 && area[cher+1][chec]!='|' && area[cher+1][chec]!='/' && area[cher+2][chec]!='|' &&area[cher+1][chec]!='\\' && area[cher+1][chec]!='\_')

{cout<<"o";

area[cher+1][chec]='o';

}

area[cher][chec]=' ';

}

else if (area[cher][chec]=='x')

{

gotoxy(cher,chec+2);

cout<<"\b\b";

cout<<" ";

gotoxy(cher+1,chec);

area[cher][chec]=' ';

if((area[cher+1][chec]=='|' || area[cher+2][chec]=='|'))

{

if(cher<=20)

if(score\_check==-1)score\_check=1;

}

area[cher][chec]=' ';

if(cher+1!=rows)

{

if(area[cher+1][chec-1]!='|' && area[cher+1][chec-1]!='/' && area[cher+1][chec-1]!='\\' &&

area[cher+1][chec-1]!='\_' && area[cher+1][chec-1]!='\\')

{

if(area[cher+1][chec]==' ' && cher+1!=rows && area[cher+2][chec]!='|' && area[cher+1][chec]!='/'

&& area[cher+1][chec]!='\\' && area[cher+1][chec]!='\_')

{

cout<<"x";

area[cher+1][chec]='x';

;

}

}

else

{

if(cher+1>=21){

system("cls");

cout<<"\t\t \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ "<<endl;

cout<<"\t\t / \_\_\_\_\_/ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_ \\_\_\_\_\_ \\ \_\_\_ \_\_ \_\_\_\_ \_\_\_\_\_\_\_ "<<endl;

cout<<"\t\t/ \\ \_\_\_ \\\_\_ \\ / \\ \_/ \_\_ \\ / | \\\\ \\/ /\_/ \_\_ \\\\\_ \_\_ \\"<<endl;

cout<<"\t\t\\ \\\_\\ \\ / \_\_ \\\_| Y Y \\\\ \_\_\_/ / | \\\\ / \\ \_\_\_/ | | \\/"<<endl;

cout<<"\t\t \\\_\_\_\_\_\_ /(\_\_\_\_ /|\_\_|\_| / \\\_\_\_ > \\\_\_\_\_\_\_\_ / \\\_/ \\\_\_\_ >|\_\_| "<<endl;

cout<<"\t\t \\/ \\/ \\/ \\/ \\/ \\/ "<<endl;

cout<<"\t\t\t\t\t\t SCORE :"<<score<<endl;

//getch();

temp\_score=score;

string s;

cout<<"\t\t\t\t\t\t SCORE :"<<score<<endl;

cout<<"\t\t\t\t\t\t Enter Name : ";

getline(cin,s);

fscore(score,s);

init();

main();

}} }}}}}

enemy e[100];

//--------------------------------------------------------------------------------------------------------------------------------

int main()

{

// for(int y=0;y<10;y++)

// initfile();

init();

GetSystemTime(&st);

srand(st.wMilliseconds);

char o=1;

do {

system("cls");

init();

string STRING;

ifstream infile;

infile.open ("welcome.txt");

while(!infile.eof()) // To get you all the lines.

{

getline(infile,STRING); // Saves the line in STRING.

cout<<STRING<<endl; // Prints our STRING.

}

infile.close();

o= getch();

if (o=='1')

{

system("cls");

//user input

char inp;

//Space::DisplaySpace();

Spacecraft spacecraft;

//gen\_en();

int c;

Space::DisplaySpace();

gotoxy(27,0);

cout<<"score : ";

for (int i = 0; i < 1000; i++)

{

gotoxy(27,10);

cout<<"\b\b\b";

cout<<" ";

if(score\_check==0)score++;

if(score\_check==1)score+=2;

score\_check=-1;

cout<<score;

Sleep(100);

spacecraft.bullet\_c();

c=rand()%74 +3;

if(i%2==0)

e[i].disp\_enemy(c);

if(i==99)

i=0;

e[1].control();

if (kbhit())

{

inp = getch();

spacecraft.Control(inp);

}

//system("cls");

}

}

else if (o=='2')

{

system("cls");

dscore();

}

else if(o=='3')

{

system("cls");

dinst();

}

}while (1);

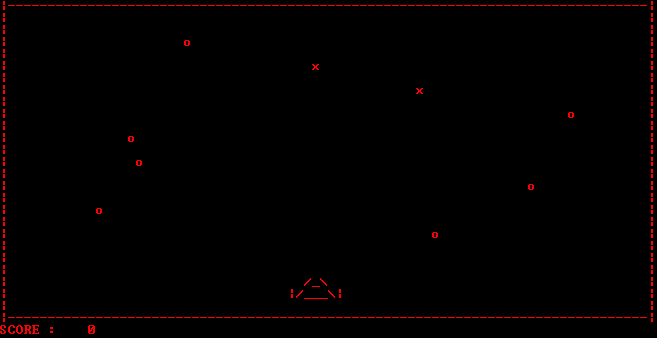
return 0;

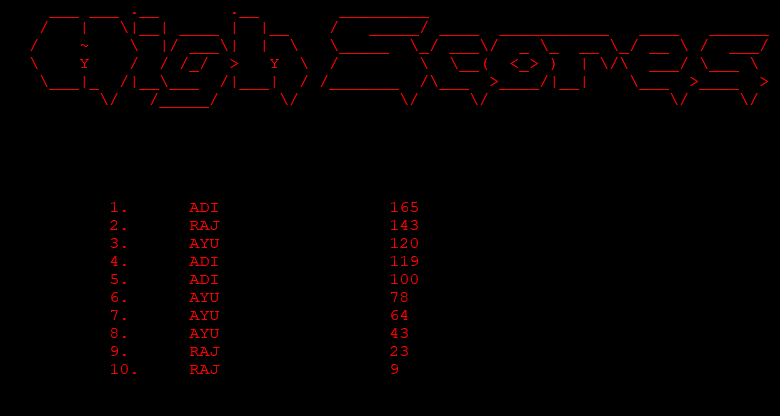
}

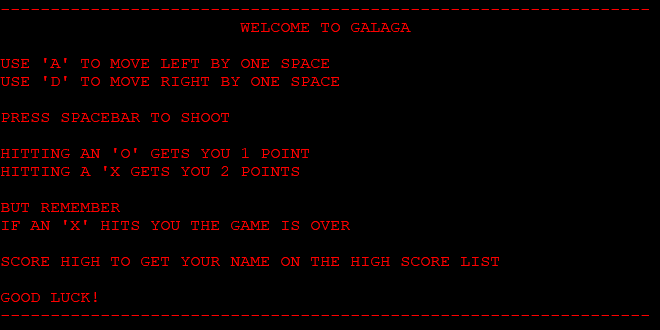
//--------------------------------------------------------------------------------------------------------------------------------

**OUTPUT SCREENS**









**BIBLIOGRAPHY AND REFERENCES**

1. https://stackoverflow.com
2. Computer Science with C++ - Sumita Arora

**FUNCTIONS**

# void fscore(int c,string k) Handling scores

# void dscore() Displaying scores

# void dinst() Displaying instructions

# void gotoxy( int line, int column ) To eliminate Screen Refresh

**STRUCTURES**

# Struct Pos Storing Position Coordinates