PROJECT PROPOSAL

# DEPARTMENT OF COMPUTER ENGINEERING

# PULCHOWK CAMPUS

# INSTITUTE OF ENGINEERING

# TRIBHUVAN UNIVERSITY

­

**PROJECT: ASCII SPACE INVADERZ**

SUBMITTED BY: DATE OF SUBMISSION:

BIGYAPTI BASHYAL - PUL076BCT016 17/02/2017

ISHANI MALLA - PUL076BCT028

MAHIMA DHALKA - PUL076BCT033

NISAN SHRESTHA - PUL076BCT040

1. Introduction:

Our project is a restoration of the classic retro game space invaders in C using basic ascii characters as a substitute for all graphical elements. In short it is a remake of the game. Users can choose difficulty via the main menu displayed as soon as the game lunches. Gameplay is fairly simple too. Players simply use the standard keyboard to interact with the program.

1. Dependencies:

Currently our code works with the GCC compiler shipped with Code::Blocks. Although our course is based on the legacy compiler TURBO C++ , we decided to use Code::Blocks and GCC as it shipped all the header files and libraries which we required.

Currently our code encounters few errors in TURBO C++ as it lacks header files ‘windows.h’. Due to this functions like Sleep() do not work out of the box. Also some user defined functions like gotoxy() and fullscreen() encounter some problems.

1. Run-Time Details:

As soon the application/code is run, users are greeted with a main menu via which they can choose the difficulty level they want to attempt (pressing 9 on the difficulty menu enables god/debug mode). To do so they use the numpad to enter the options displayed on the screen. Once the play game option has been selected, players will be told to get ready and the main game will start after the playing window of console is cleared.

The gameplay is pretty straightforward as user use the following keys to play the game:

1. ‘a’ – To move left.
2. ‘d’ – To move right.
3. ‘s’ – To shoot/fire bullet.

Players are allowed limited no of active bullets at a time depending upon the difficulty chosen. The goal is simple; to destroy invading aliens who move left and right while simultaneously travelling downwards.

The game is won when all enemies are destroyed while it is lost if any of the enemy crosses the player row.

Both of the last two features are achieved using phantom object whose coordinates are equivalent to the bottom most - left and right enemies during the initialization. Their coordinates are updated as the game goes on. If any of the phantom object hits the side borders the position of all enemies are moved one step down and their horizontal velocity is reversed. If the phantom object reaches the player row game is lost. Also, the position of phantom object is moved two steps up if bottom row of enemy is destroyed and so on for second row.

Following inputs are also valid during gameplay:

1. ‘p’ – To pause game.
2. ‘x’ – To quit game.
3. Additional Information:

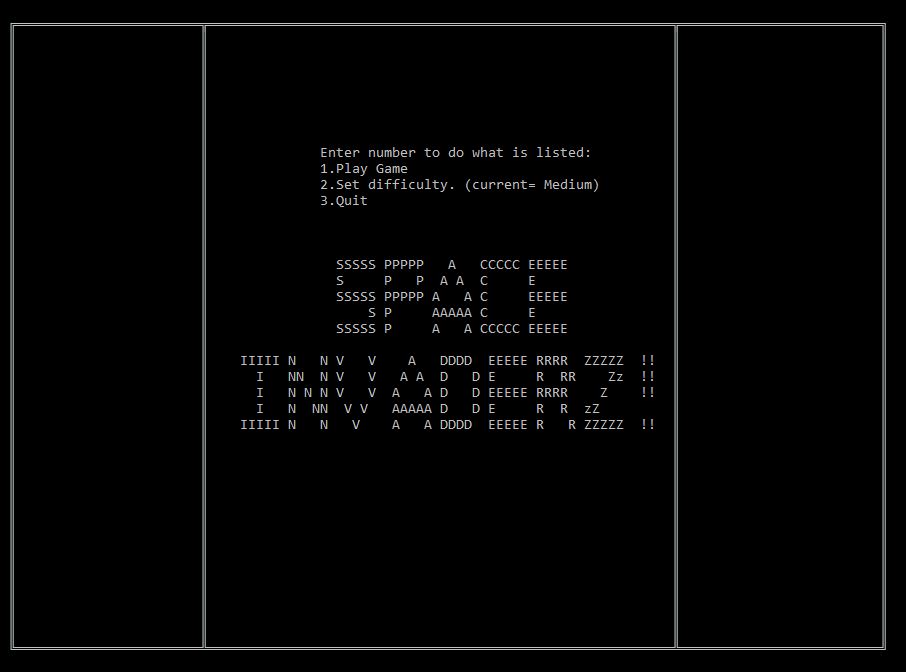
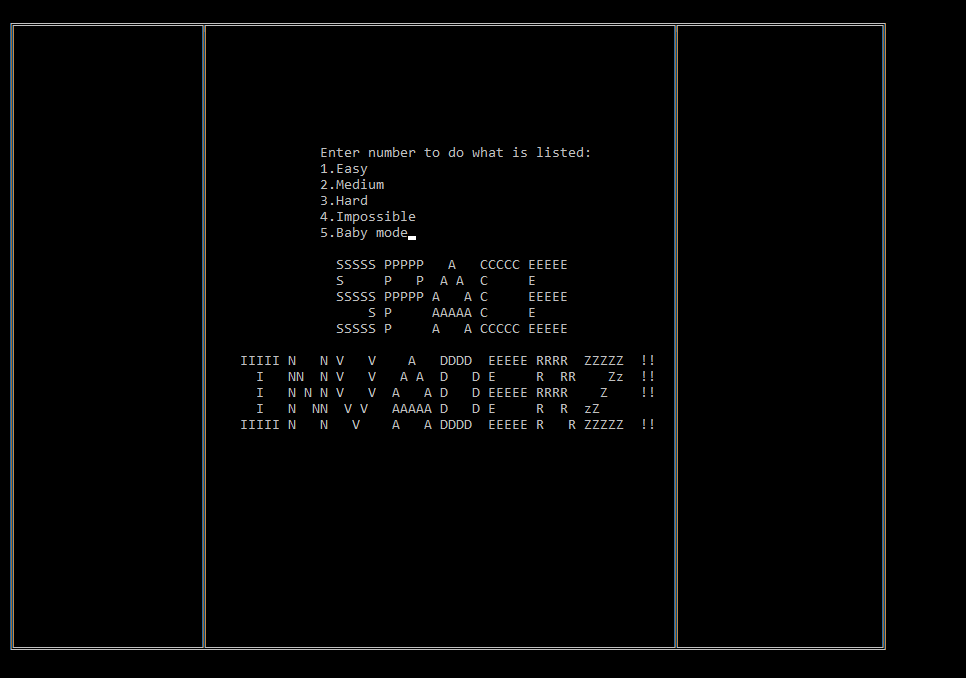
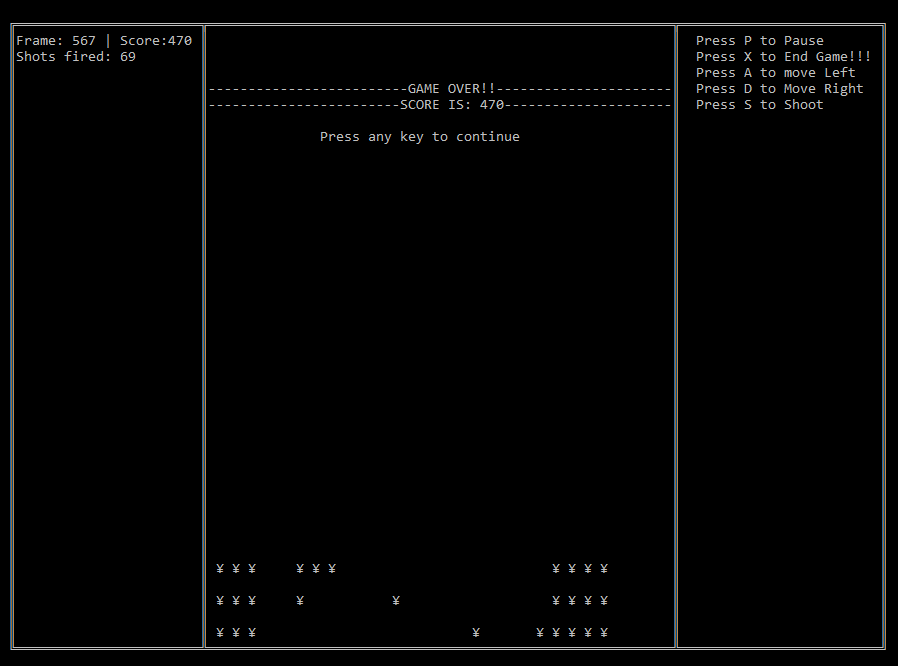
Few lines in the code have been copied via various sources to make the program more user friendly. Notedly functions like fullscreen(); and gotoxy(); have been defined using statements unfamiliar to us as students.

Apart from these instances the rest of the code including collision logic was written from scratch without using templates of any kind. However, it can be noted that various sources were referred to in order to gain knowledge on game development process.

1. Known Bugs:

There are no currently known bugs.

1. Sample Output:

Since games are dynamic programs with continuously updating output screen following few screen grabs don’t do full justice to the almost 620 lines of code, we encourage you to download the source from the project GitHub over at “tiny.cc/Project\_github” and try it for yourself. Here are few screengrabs anyways:

1. Source Code:

While it is to be noted that it may change (available on GitHub over at “tiny.cc/Project\_github”) at following is the current version of the source code:

#include<stdio.h>

#include<conio.h>

#include <Windows.h>

char difficulty[10]="Medium";

long frame=1;

int bullet\_shot=0;

int score=0;

int moving\_frame=10;

int gameover=0;

int pause=0;

int width=60;

int height=40;

int shoot=0;

int player\_bullet\_used=0;

int player\_max\_bullet=3;

int dir\_player=0; //direction of movement

int enemy\_dir\_y=0; //direction of movement

int enemy\_dir\_x=1; //direction of movement

int ending=0;

typedef struct bullet

{

int x,y; //coordinates

int last\_y; //last position to be cleared

int dead; //if 1 initiates kill status

int used\_or\_not; //bullet slot used or not

};

typedef struct object

{

int x,y; //coordinates

float dx; //direction of movement

int life; //if 1 alive if 0 dead

int last\_x,last\_y; //last position to be cleared

};

void gotoxy(short x, short y)

{

COORD pos={x,y};

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),pos);

}

void fullscreen()

{

keybd\_event(VK\_MENU,0x38,0,0);

keybd\_event(VK\_RETURN,0x1c,0,0);

keybd\_event(VK\_MENU,0x38,KEYEVENTF\_KEYUP,0);

keybd\_event(VK\_RETURN,0x1c,KEYEVENTF\_KEYUP,0);

}

void init\_bullet(struct bullet player\_bullet[])

{

int i;

for (i=1;i<10;i++)

{

player\_bullet[i].x=0;

player\_bullet[i].y=0;

player\_bullet[i].used\_or\_not=0;

player\_bullet[i].dead=0;

}

}

void init\_struct\_enemy(struct object enemy[],struct object \*phantom\_left,struct object \*phantom\_rig)

{

int c,i=5,j=3;

for (c=1;c<=75;c++)

{

enemy[c].life=1;

i+=2;

enemy[c].x =i+25;

enemy[c].y =j;

enemy[c].last\_x =enemy[c].x ; //to prevent flicker if not initialized

enemy[c].last\_y =enemy[c].y ;

if (c%25==0)

{

i=5;

j+=2;

}

}

phantom\_left-> x = enemy[1].x;

phantom\_rig -> x = enemy[25].x;

phantom\_left-> y = enemy[51].y;

phantom\_rig -> y = enemy[75].y;

}

void draw\_init\_splash()

{

int i,j;

system("cls");

//prints top border

gotoxy(1,1); //goto start of border

for (i=1;i<=width+25+25;i++)

{

if(i==1 ) printf("%c",201);

else if(i==(width+50)) printf("%c",187);

else if(i==25 || i==25+width-1 ) printf("%c",203);

else printf("%c",205);

}

//prints vertical border,

for(j=2;j<=height-1;j++)

{

gotoxy(1,j); //goto start of each line

for(i=1; i <= (width+25+25) ; i++)

{

if(i==1 || i==25 || (i==25+width-1) || i==(width+50))

printf("%c",186);

else

printf(" ");

}

printf("\n");

}

//lower border

gotoxy(1,height); //goto last line

for (i=1;i<=(width+25+25);i++)

{

if(i==1 ) printf("%c",200);

else if(i==(width+25+25)) printf("%c",188);

else if(i==25 || i==25+width-1 ) printf("%c",202);

else printf("%c",205);

}

//prints styled name.

gotoxy(30,16);

printf(" SSSSS PPPPP A CCCCC EEEEE\n");

gotoxy(30,17);

printf(" S P P A A C E \n");

gotoxy(30,18);

printf(" SSSSS PPPPP A A C EEEEE\n");

gotoxy(30,19);

printf(" S P AAAAA C E \n");

gotoxy(30,20);

printf(" SSSSS P A A CCCCC EEEEE\n");

gotoxy(30,21);

printf("\n");

gotoxy(30,22);

printf("IIIII N N V V A DDDD EEEEE RRRR ZZZZZ !!\n");

gotoxy(30,23);

printf(" I NN N V V A A D D E R RR Zz !!\n");

gotoxy(30,24);

printf(" I N N N V V A A D D EEEEE RRRR Z !!\n");

gotoxy(30,25);

printf(" I N NN V V AAAAA D D E R R zZ \n");

gotoxy(30,26);

printf("IIIII N N V A A DDDD EEEEE R R ZZZZZ !!\n");

main\_menu();

gotoxy(45,14);

if(ending==0)

printf(" GET READY!!! ");

else

printf("Quitting Please Wait");

Sleep(500);

gotoxy(45,14);

printf(" ");

Sleep(500);

}

void main\_menu(){

char ch;

int i,j,switch\_complete=0;

for(i=26;i<84;i++)

{

for(j=5; j<15; j++)

{

gotoxy(i,j);

printf(" ");

}

}

gotoxy(40,9);

printf("Enter number to do what is listed:");

gotoxy(40,10);

printf("1.Play Game");

gotoxy(40,11);

printf("2.Set difficulty. (current= %s)",difficulty);

gotoxy(40,12);

printf("3.Quit ");

while(switch\_complete!=1)

{

ch=getch();

switch(ch)

{

case '1':

clear\_console(); //clears loading screen

switch\_complete = 1;

break;

case '2':

ch = difficulty\_select();

switch\_complete = 1;

break;

case '3':

gotoxy(30,13);

printf("THANK YOU FOR CHECKING OUT ASCII SPACE INVADERZ!!!");

switch\_complete = 1;

ending=1;

break;

}

}

}

void difficulty\_select()

{ int i,j,switch\_complete=0;

char ch;

for(i=26;i<84;i++)

{

for(j=5; j<15; j++)

{

gotoxy(i,j);

printf(" ");

}

}

gotoxy(40,9);

printf("Enter number to do what is listed:");

gotoxy(40,10);

printf("1.Easy");

gotoxy(40,11);

printf("2.Medium");

gotoxy(40,12);

printf("3.Hard");

gotoxy(40,13);

printf("4.Impossible");

gotoxy(40,14);

printf("5.Baby mode");

while(switch\_complete!=1 )

{

ch=getch();

switch(ch)

{

case '1':

moving\_frame=16;

player\_max\_bullet=5;

strcpy(difficulty,"Easy");

switch\_complete = 1;

break; //continue without doing anything

case '2':

moving\_frame=10;

player\_max\_bullet=4;

strcpy(difficulty,"Medium");

switch\_complete = 1;

break;

case '3':

moving\_frame=6;

player\_max\_bullet=3;

strcpy(difficulty,"Hard");

switch\_complete = 1;

break;

case '4':

moving\_frame=3;

player\_max\_bullet=2;

strcpy(difficulty,"Impossible!");

switch\_complete = 1;

break;

case '5':

moving\_frame=26;

strcpy(difficulty,"Baby mode");

player\_max\_bullet=8;

switch\_complete = 1;

break;

case '9':

moving\_frame=2;

strcpy(difficulty,"debug");

player\_max\_bullet=8;

switch\_complete = 1;

break;

default:

break;

}

}

main\_menu(); //returns to menu screen

}

void clear\_console()

{

int i,j;

for(i=26;i<84;i++)

{

for(j=2; j<35 ; j++)

{

gotoxy(i,j);

printf(" ");

}

}

}

void draw\_struct\_enemy(struct object enemy[])

{

int i;

for (i=1;i<=75;i++)

{

if(enemy[i].y<height)

{

gotoxy(enemy[i].last\_x,enemy[i].last\_y);

printf(" ",157);

}

if (enemy[i].life != 0)

{

gotoxy(enemy[i].x,enemy[i].y);

printf("%c",157);

}

}

}

void shoot\_check(struct bullet player\_bullet[],struct object \*player)

{

int i;

if (shoot==1 && player\_bullet\_used < player\_max\_bullet) //to ensure limited bullet

{

player\_bullet\_used++; //increases counter which tracks no of active bullets

for(i=1;player\_bullet[i].used\_or\_not==1;i++); //finds empty bullet slot at i

if (player\_bullet[i].used\_or\_not != 1) //to ensure bullet is empty at i found

player\_bullet[i].used\_or\_not=1; //changes empty bullet slot status to used

player\_bullet\_release(player\_bullet,player,i); //Sets position & direction of fired bullet and i= bullet index

//display no of bullets successfully fired in game

gotoxy(2,3);

printf("Shots fired: %d ",++bullet\_shot);

}

}

void enemy\_move\_compute(struct object enemy[],struct object \*phantom\_left,struct object \*phantom\_rig)

{

int i;

if (phantom\_left-> x <= (25+1))

{

gotoxy(1,26);

enemy\_dir\_y = 1;

enemy\_dir\_x = 1;

phantom\_rig->y += 1;

}

if (phantom\_rig-> x >= (25+width-2))

{

enemy\_dir\_y = 1;

enemy\_dir\_x = -1;

phantom\_rig->y += 1;

}

for (i=0;i<=75;i++)

{

enemy[i].last\_x=enemy[i].x;

enemy[i].last\_y=enemy[i].y;

enemy[i].x += enemy\_dir\_x;

enemy[i].y += enemy\_dir\_y;

}

phantom\_left->x +=enemy\_dir\_x;

phantom\_rig->x +=enemy\_dir\_x;

enemy\_dir\_y=0;

if (phantom\_rig->y == height-1 )

gameover=1;

}

void draw\_player(struct object \*player)

{

gotoxy(player->last\_x,player->y);

printf(" ");

gotoxy(player->x,player->y);

printf("%c",142);

gotoxy(1,26);

}

void input()

{

if(\_kbhit())

{

switch(getch())

{ //if user presses stuff then only getch() activates and hence takes input

case 'a':

dir\_player=-1;

break;

case 'd':

dir\_player=1;

break;

case 's':

shoot = 1;

break;

case 'p':

pause = 1;

break;

case 'x':

gameover=1;

break;

default:

shoot = 0;

dir\_player = 0;

break;

}

}

}

void player\_bullet\_release(struct bullet player\_bullet[],struct object \*player, int i)

{

int j;

player\_bullet[i].x = player->x;

player\_bullet[i].y = (player->y-1);

}

void draw\_player\_bullet(struct bullet player\_bullet[])

{

int i,cursor\_debug=1;

for(i=1;i<=player\_max\_bullet;i++)

{

if (player\_bullet[i].used\_or\_not==1)

{

player\_bullet[i].last\_y=player\_bullet[i].y; //store current line of bullet in temp

player\_bullet[i].y -= 1; //update bullet one step up

gotoxy(player\_bullet[i].x,player\_bullet[i].last\_y); //clear last position of bullet

printf(" ");

gotoxy(player\_bullet[i].x,player\_bullet[i].y); //print new positon

printf("1");

if ( player\_bullet[i].y <= 2)

player\_bullet[i].dead=1;

}

//if dead remove||free bullet

if (player\_bullet[i].dead==1)

{

gotoxy(player\_bullet[i].x,player\_bullet[i].y); //clear last position of bullet

printf(" ");

player\_bullet[i].y=26; //prevent unintended false positive collision

player\_bullet[i].used\_or\_not=0; //reset used status

player\_bullet[i].dead=0; //reset freeing algorithm

player\_bullet\_used--; //update number of active bullet

}

}

gotoxy(1,26);

}

void collision(struct bullet player\_bullet[],struct object enemy[])

{

int ec,bc,ey,ex,bx,by;

for(ec=1;ec<=75;ec++)

{

ex=enemy[ec].x; //storing coordinate for easy view

ey=enemy[ec].y; //storing coordinate for easy view

for(bc=1;bc <= player\_max\_bullet;bc++) //for checking each enemy with each bullet

{

bx=player\_bullet[bc].x; //storing coordinate for easy view

by=player\_bullet[bc].y; //storing coordinate for easy view

if(ex==bx && ey==by && player\_bullet[bc].dead == 0 && enemy[ec].life==1)

{

score+=10; //basic score

enemy[ec].life=0; //empty bullet

player\_bullet[bc].dead=1; //initialize bullet freeing algorithm in draw\_bullet

gotoxy(player\_bullet[bc].x,player\_bullet[bc].y);

printf("x"); //replace last position of bullet with collision indicator

}

}

}

}

void phantom\_update(struct object enemy[],struct object \*phantom\_left,struct object \*phantom\_rig)

{

int bottom\_alive=0;

int mid\_alive=0;

int top\_alive=0;

int i;

for (i=26;i<=50;i++)

{

if(enemy[i].life!=0)

mid\_alive++;

}

for (i=51;i<=75;i++)

{

if(enemy[i].life!=0)

bottom\_alive++;

}

if(bottom\_alive==0)

{

phantom\_left -> y = enemy[26].y;

phantom\_rig -> y = enemy[50].y;

if(mid\_alive==0)

{

phantom\_left -> y = enemy[1].y;

phantom\_rig -> y = enemy[25].y;

}

}

}

void main()

{

int i,j;

struct object player; //player object

struct object enemy[76]; //enemies (76 but 75 used)

struct object phantom\_left; //non printing object for game over and direction check

struct object phantom\_rig; //non printing object for game over and direction check

struct bullet player\_bullet[100]; //player bullet (100 defined to avoid a struct array overflow error

char replay;

fullscreen();

start:

init\_struct\_enemy(enemy,&phantom\_left,&phantom\_rig); //initializes position of all enemies

init\_bullet(player\_bullet);

player.x=40; // player position initialization

player.y=height-1; // player position initialization

score = 0;

frame = 1;

pause = 0;

bullet\_shot=0;

draw\_init\_splash(); // Border and get ready

if (ending==1)

goto end;

fflush(stdin); //clear input buffer

while(!gameover && score < 750)

{

if (frame % moving\_frame==0) //so that enemy moves every (moving\_frame) frames

enemy\_move\_compute(enemy,&phantom\_left,&phantom\_rig); //changes position of enemy and phantom object

input(); //take input

fflush(stdin); //clear input buffer

if (pause==1)

{

gotoxy(27,27);

printf("Press any key to continue.");

pause=0;

getch();

gotoxy(27,27);

printf(" ");

//to pause until user presses enter.

}

collision(player\_bullet,enemy); //Check for collision and kill bullet and enemy

draw\_player(&player); //update player position

draw\_struct\_enemy(enemy); //draws all enemies

draw\_player\_bullet(player\_bullet); //draws player bullet

phantom\_update(enemy,&phantom\_left,&phantom\_rig);

if (score>750)

moving\_frame = 3; //last 10 enemy move with very high speed.

//compute position of player

if(player.x>26 || dir\_player != -1)

{

if(player.x < (25+ width - 2) || dir\_player!=1 )

{

player.last\_x = player.x; //save last position clean screen at the point ; refer to void draw\_player();

player.x += dir\_player; //update to new position according to input(); from user

dir\_player=0; //reset move direction

}

}

shoot\_check(player\_bullet, &player);//compute shoot status of player

shoot=0; //resets shooting status for player

gotoxy(2,2);

printf("Frame: %d | Score:%d", ++frame,score);

Sleep(5); //frame rate control

}

gotoxy(26,4);

printf(" ");

gotoxy(26,5);

if(score==750)

{

printf(" Congrats You won ");

}

else

{

printf("-------------------------GAME OVER!!----------------------");

gotoxy(26,6);

printf("------------------------SCORE IS: %3d---------------------",score);

}

gotoxy(26,7);

printf(" ");

gotoxy(40,8);

printf("Press any key to continue");

getch();

gameover=0;

goto start;

end:

gotoxy(1,height+1);

}