

**University of Dhaka**

**Department of Computer Science and Engineering**

**CSE – 1211**

**Project: The Stick Champ**

Submitted to:

Dr. Muhammad Asif Hossain Khan

Associate Proffesor

Ms. Anna Fariha

Lecturer

Submitted by:

Mehreen Rahman (SN-61)

Date of Submission: 3.12.15

December 3rd,2015.

Contents Page

1. Introduction 03

2. Project Name 03

3. Project Outline 03

4. Main Menu Features 03

5. Game Feature 04

6. Coding Implementation 05

7. Difficulties Encountered 06

8. Graphical Interface 07

9. Source code functions 08

10. Main Game Interface 10

11. Algorithmic approach 16

12. Source Code 18

13. Conclusion 27

References

**Introduction:**

The main objective of this project is to implement a basic graphics library named BGI (Borland Graphics Interface) built in C.BGI is a graphics library bundled with several Borland compilers for the DOS operating systems since 1987.BGI is less powerful than modern graphics libraries such as OpenGL since it was designed for presentation graphics rather than supporting 3D environment, making it simpler to code and implement.

**Project name:**

The Stick Champ

**Project outline:**

A 2D continuous game. It is inspired from a very popular game named “Stick Hero”. The game features keyboard navigation by the user. It’s basically a single player game where the user has to estimate the distance between the consecutive pillars, connect the pillars through a bridge & pass it successfully. The unsuccessful pass marks determine the end of this game.

**Main Menu Features:**

1. Play Now Tab :

After a mouse click by the user on this Tab, the main game play is activated & the game starts.

1. How To Play Tab:

This tab will show the user the keyboard navigation for the game play to the user.

1. Credit Tab :

This tab includes the information of the game maker.

1. Exit Game Tab :

By a click on this tab, user can exit the game.

**Game Features:**

1. **Character & story:**

In this game, there’s a character named Ninja. It is escaping using it’s stick to make a bridge between two pillars and thus get away.

1. **Continuous pillars printing:**

The pillars will be printing continuously and randomly in a loop. The game will continue until the user can manage to pass the pillars.

1. **Every pillar with random thickness:**

Every pillar will be printing with randomized thickness. This thickness will not be predictable. The randomness of thickness eliminates the probability of the user’s prediction.

1. **Keyboard navigation of the user:**

The user can form a stick of its own in order to make the bridge between the pillars which it can use to pass. The user can also increment the length of the stick of its own by giving input from the keyboard. The longer the user presses the up key, the longer the stick grows

1. **Keyboard Controls:**

The user controls are implemented by using ASCII codes for the keys. The keys used in this game for the user navigation is up arrow key, enter, space and escape key.

1. **Game over criteria& scores:**

If the user fails to judge the distance correctly, the game ends as the ninja fails to reach to the next pillar & falls. If reached, user gets a point.

1. **Bonus feature:**

If the stick can reach a length somewhere around 20 units from the center of the pillar, it will get a bonus point.

**Coding Implementations:**

* 1. **Division into modules:**

The whole coding needed proper organization and to ensure its comprehensibility it needed to be divided into modules. Several .cpp files are used to implement different components. Also, there are several defined functions in each .cpp file which are also used for different purposes.

The main menu containing main game play, How to play, Credits and Exit each having different modules and .cpp files

* 1. **The Character Designing:**

The first phase was the character designing. The Ninja was drawn using rectangles and circles, all parts drawn by different functions & passing variable.

* 1. **Movement of lines, bars & characters:**

All the movement and continuous animation such as falling of the stick has been implemented by passing different parameters to the functions assigned to do that particular module.

* 1. **Measurements of pillars & the character:**

This game needed a good estimation & measurement of the characters for the graphical enhancement. perfect measurement according to the window has been accomplished after several testing. The testing phase turned out to be a lengthy & crucial phase.

* 1. **Handling all the events simultaneously:**

In the main game play, all the events such as stick increment, taking input, stick falling are done in different functions. Similarly as done in the main menu features such as managing play now , how to play & credits window.

**Difficulties encountered:**

1. **Division into modules:**

The whole project should be well organized. In order to keep the whole code well arranged, the entire code must be divided into modules. But in this case, handling each of the modules and passing parameters among these functions and the main functions are not easy tasks either.

1. **Screen limit:**

The Ninja or the pillars were not to be moved out of the screen. So we had to put necessary limits in the code to debug.

1. **Flickering:**

BGI displays too much flickering when any image is taken as input as a background. So the diagrams had to be drawn manually and then convert them in respect of only two points just to reduce flickering, which was not an easy task at all.

1. **Maintaining the speed:**

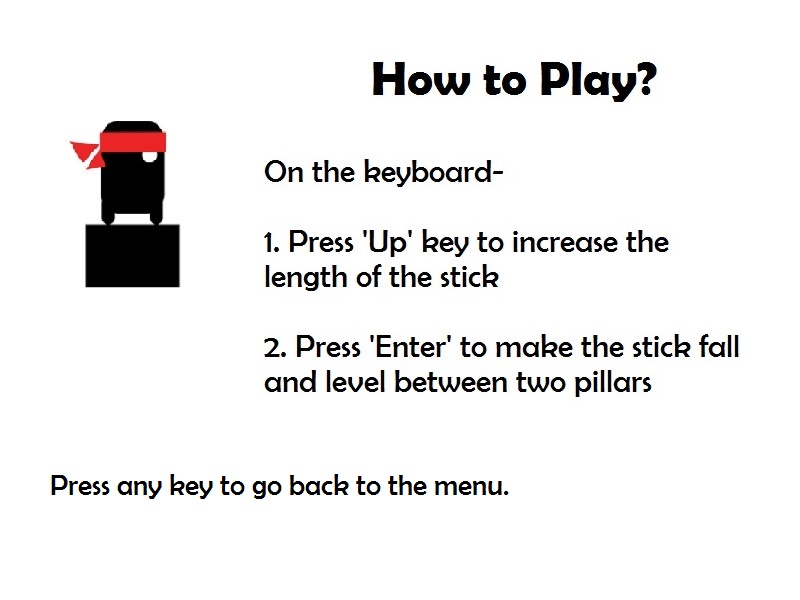
When too much delay() function is used in order to reduce flickering, it slows down the game. So to maintain the speed and reduce flickering at the same time was also a challenge when coding.

**Graphical interface:**

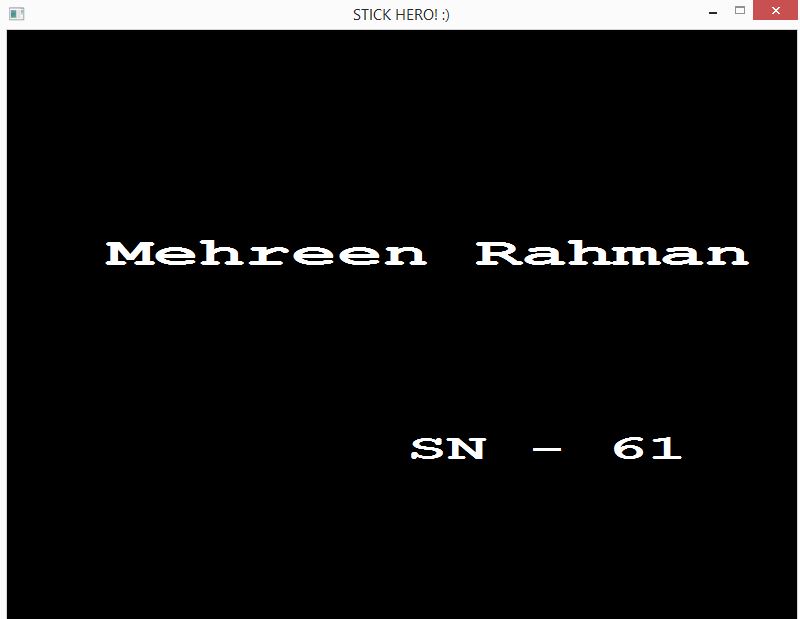
1. **Main Menu:**



1. **How to play:**



1. **Credits Window:**

****

**Source Code Functions:**

1. Functions of the header file “graphics.h”:

void initwindow (int width, int height):

The function initializes the graphics system by opening a graphics window of the specified size.

void delay (int millisec):

The function pauses the computation for the the specified number of milisec

void fillellipse (int x, int y, int xradius, int yradius):

Draws an ellipse using (x,y) as a center point and xradius and yradius

and vertical axes, and fills it with the current fill color and fill pattern.

void fillpoly (int numpoints, int \*polypoints):

It draws the outline of a polygon with numpoints points in the current line style and color (just as drawpoly does), then fills the polygon using the current fill pattern

and fill color.

void settextstyle (int font, int direction, int charsize):

This function sets the text font, the direction in which text is displayed, and the size of the characters.

void outtextxy (int x, int y, char \*textstring):

This function shows message on a given coordinate.

void readimage(int x,int y,int x2,int y2):

This function takes only JPEG and BITMAP image.

void setfillstyle (int pattern, int color):

This function imposes the color and pattern of filling.

void rectangle(void rectangle (int left, int top, int right, int bottom);

This function can draw rectangles.

1. Functions in “MyHeader.h” & other defined functions:

void dellipse()

void dpoly()

void gameload();

int printmenu();

int printhowtoplay();

void game();

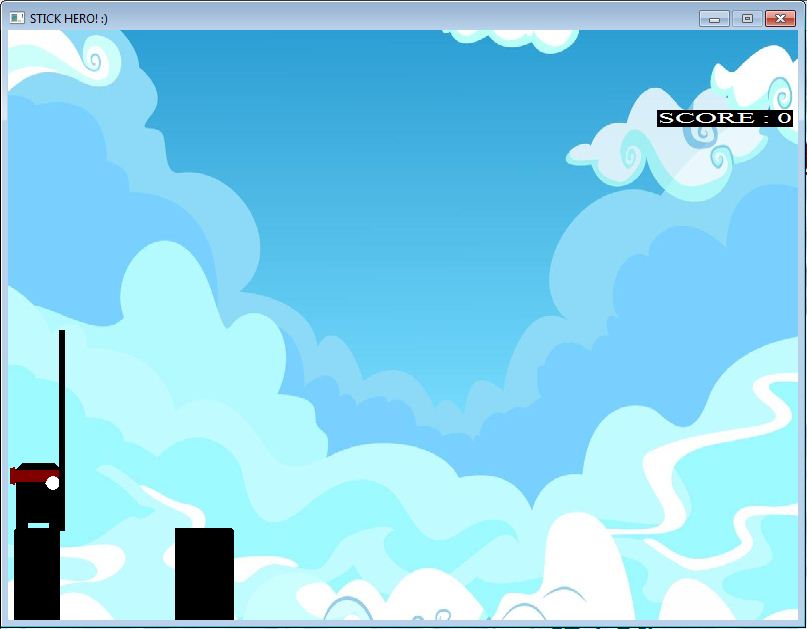
int credits();

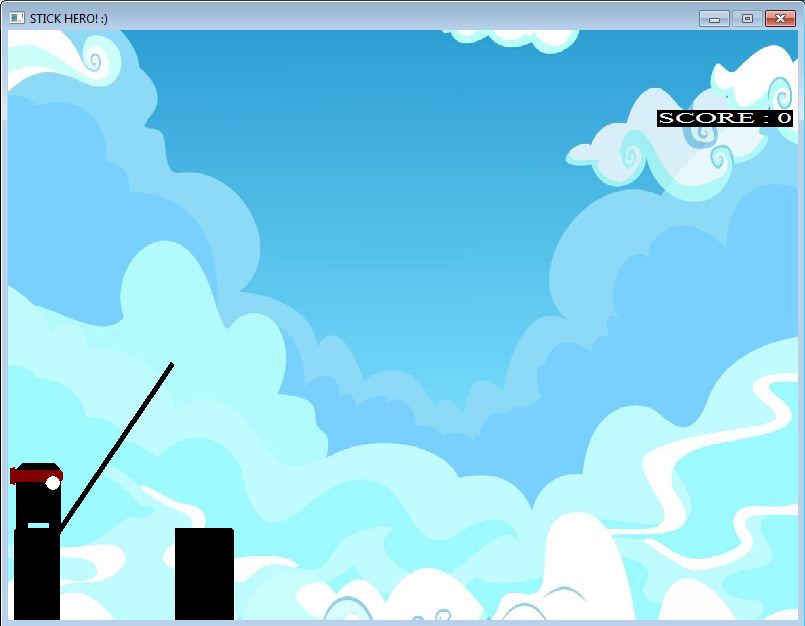
**Main game interface:**

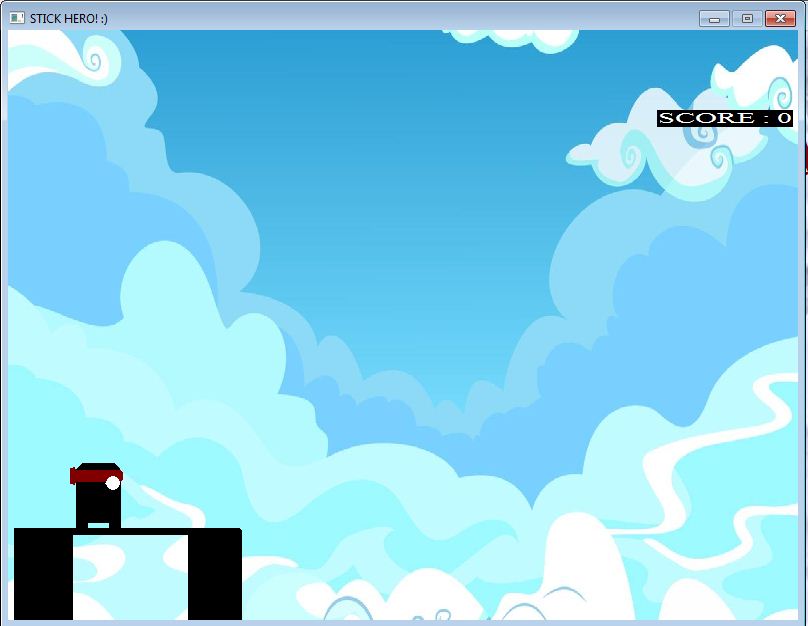
****

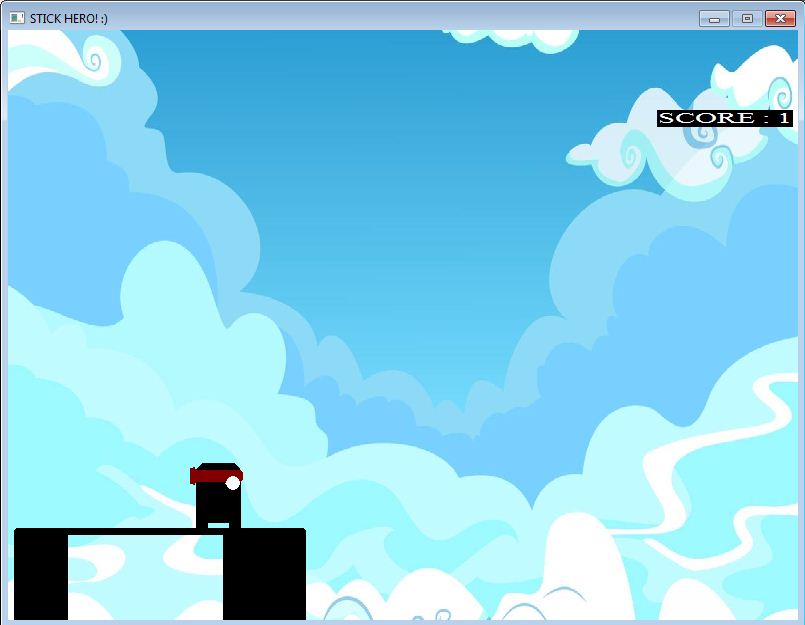
****

**­**

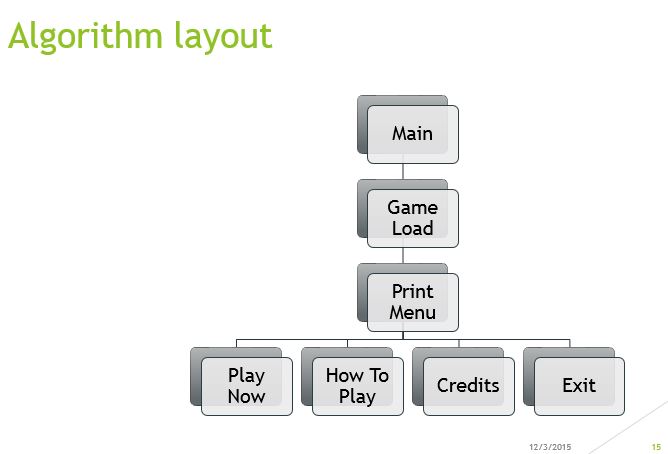
****

****

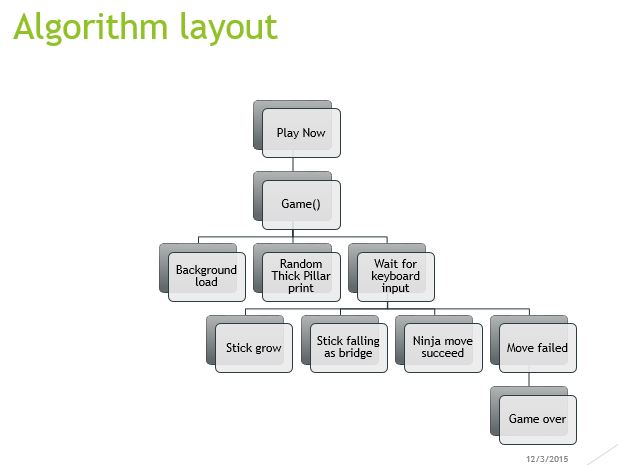
****

****

**Algorithmic approach:**



Flow chart1: The inter relation of functions of the outer structure



Flow chart2: The inter relation and order of functions of levels

**Source Code**

MyHeader.h:

#include "graphics.h"

#include "stdio.h"

int printmenu();

int credits();

int printhowtoplay();

void gameover();

void gameload();

int game();

game.cpp:

#include "graphics.h"

#include <stdio.h>

#include<math.h>

char score[100];

void delipse(int x, int y, int a)

{

setcolor(15);

setfillstyle(SOLID\_FILL, WHITE);

fillellipse(x, y, a, a);

}

void dpoly(int c, int p, int a[10])

{

setcolor(c);

setfillstyle(SOLID\_FILL, c);

fillpoly(p, a);

}

void linesinc(int x, int y)

{

line(10+x, 500, 10+x, 500-y);

line(11+x, 500, 11+x, 500-y);

line(12+x, 500, 12+x, 500-y);

line(13+x, 500, 13+x, 500-y);

line(14+x, 500, 14+x, 500-y);

line(15+x, 500, 15+x, 500-y);

}

void printscore(int value)

{

setcolor(15);

settextstyle(TRIPLEX\_FONT, HORIZ\_DIR, 1);

sprintf(score, "SCORE : %d", value);

outtextxy(650, 80, score);

}

void highscore(int value)

{

FILE \*f;

f = fopen("input.txt","r");

int d;

fscanf(f,"%d",&d);

if(d < value) {

d = value;

}

fclose(f);

f = fopen("input.txt","w");

fprintf(f,"%d",d);

fclose(f);

char lame[50];

setcolor(WHITE);

sprintf(lame,"%d\n",d);

outtextxy(350,500,lame);

char arr[50];

sprintf(arr,"%d\n",value);

outtextxy(350,350,arr);

printf("%s\n",arr);

getch();

}

int game()

{

int pillarthick1, heromoves, x = 0; //pillarthick1 defines the width of the previous pillar

int stickinc=20, stepcnt;

//stickinc defines how much the stick will increase and it is fixed to increase from the length 20units

int pillarthick2=(rand()%40)+40;

//pillarthick2 defines the width of the next pillar. rand is used here to make sure that the next pillar is

//of a random width

pillarthick1=pillarthick2;

//this is required as randomly generated pillarthick2 swaps and becomes pillarthick1

setlinestyle(SOLID\_LINE, 0, 5); //to ensure the thickness of the stick

stepcnt=(rand()%(24))+2;

//stepcnt is randomized so that the distance between the two pillars (which is later initialed as dist1)

//is randomized as well

while(1)

{

heromoves=20;

//how many steps the character going to take after the stick's been replaced between the two pillars

readimagefile("background.jpg", 0, 0, 800, 600);

printscore(value);

pillarthick1=pillarthick2;

pillarthick2=(rand()%40)+40;

dpoly(0, 15, points); //draws the black part of the body

dpoly(RED, 9, recs); //draws the red part of the body

dellipse(i, j, r); //draws the eyes

int strtrec[10]= {8, 600, 8+pillarthick1, 600, 8+pillarthick1, 500, 8, 500, 8, 600};

//the array for the previous pillar

dpoly(0, 5, strtrec); //draws the previous pillar

stepcnt=(rand()%(24))+2;

while((x+(stepcnt\*heromoves)-8)<=0 || (x+(stepcnt\*heromoves)+pillarthick1+pillarthick2)>=588)

{

stepcnt=stepcnt-1;

//if the distance between the starting point of the previous pillar and the ending point of the next pillar

//goes out of the window, then stepcnt is decreased so that the total distance mentioned remains inside

//the window

}

//x=8;

int nxtrec[10]= {8+pillarthick1+(stepcnt\*heromoves), 600, (8+(stepcnt\*heromoves)+pillarthick1+pillarthick2), 600, (8+(stepcnt\*heromoves)+pillarthick1+pillarthick2), 500, 8+pillarthick1+(stepcnt\*heromoves), 500, 8+pillarthick1+(stepcnt\*heromoves), 600};

//the array for the next pillar

dpoly(0, 5, nxtrec); //draws the next pillar

while(1)

{

c=getch();

delay(1);

if(c==72 && stickinc<=499) //72= up key

{

stickinc=stickinc+20; //stick will increase when the up key's going to be pressed and cannot go out of the window screen for which

//the above condition's been fixed

}

if(c==13) //13= enter

{

break;

//after the stick has been lengthened, it will break the loop and conduct the next loop which makes

//fall down to the ground with an angular motion

}

linesinc(pillarthick1, stickinc); //lines increased using the pre-defined function

}

double angle = 3.1416/2;

while(angle>=0)

{

xi =(int)stickinc\*1.0\*cos(angle),

yi =(int)stickinc\*1.0\*sin(angle); //changes the x and y co-ordinates according to the changing angle

readimagefile("background.jpg", 0, 0, 800, 600);

printscore(value);

setcolor(0);

line(10+pillarthick1, 500, 10+pillarthick1+xi, 500-yi);

dpoly(0, 5, strtrec);

dpoly(0, 5, nxtrec);

dpoly(0, 15, points);

dpoly(RED, 9, recs);

dellipse(i, j, r);

angle -= 0.05;

}

int fstickl=stickinc; //defines the final stick length

int dist1=stepcnt\*heromoves; //the distance in between the two pillars

int dist2=dist1+pillarthick2; //previous distance including the next pillars width

int moveinc=0; //how many moves the character can make according to the final stick length

while((moveinc\*heromoves)<=fstickl)

{

readimagefile("background.jpg", 0, 0, 800, 600);

printscore(value);

dpoly(0, 4, strtrec);

dpoly(0, 4, nxtrec);

a=heromoves; //the amount of change of the x coordinate of the character

b=0;

moveinc++;

int cpypoints[50]= {p1+a\*moveinc, p2+b, p3+a\*moveinc, p4+b, p5+a\*moveinc, p4+b, p6+a\*moveinc, p2+b, p6+a\*moveinc, p7+b, p8+a\*moveinc, p9+b, p5+a\*moveinc, p9+b, p10+a\*moveinc, p7+b, p10+a\*moveinc, p11+b, p12+a\*moveinc, p11+b, p12+a\*moveinc, p7+b, p13+a\*moveinc, p9+b, p14+a\*moveinc, p9+b, p1+a\*moveinc, p7+b, p1+a\*moveinc, p2+b};

int cpyrecs[20]= {q1+a\*moveinc, q2+b, q3+a\*moveinc, q2+b, q3+a\*moveinc, q4+b, q1+a\*moveinc, q4+b, q1+a\*moveinc, q5+b, q6+a\*moveinc, q7+b, q6+a\*moveinc, q8+b, q1+a\*moveinc, q9+b, q1+a\*moveinc, q2+b};

//cpypoints and cpyrecs are the changed arrays which will make the character movement possible

dpoly(0, 15, cpypoints);

dpoly(RED, 9, cpyrecs);

dellipse(i+a\*moveinc, j+b, r); //the character's been drawn after the movement

setcolor(0);

setfillstyle(SOLID\_FILL, 0);

rectangle(8+pillarthick1, 500, 8+pillarthick1+fstickl, 502);

floodfill(8+pillarthick1+1, 501, 0);

//the final stick, which is drawn using a rectangle, leveled between the two pillars according

//to their increased length

delay(200); //to make the movement of the character visible

}

if(fstickl>=dist1 && fstickl<=dist2) //required condition, to point a score

{

value++;

printscore(value);

if(fstickl>=(dist1+20) && fstickl<=(dist2-20)) //

{

char bonus[10];

settextstyle(TRIPLEX\_FONT, HORIZ\_DIR, 3);

sprintf(bonus, "+1!!");

outtextxy(380, 150, bonus);

value++;

delay(500);

}

printscore(value);

}

else

{

gameover();

highscore(value);

return 0;

}

stickinc=20; //this makes the stick length back to it's starting self and thus repeats the process for the next pillars to appear

}

}

Source.cpp:

#include "graphics.h"

#include "MyHeader.h"

#include <stdio.h>

#include <math.h>

int active=1;

int main( )

{

initwindow(800, 600, "The Stick Champ!");

gameload();

int active=1;

while (active==1)

{

cleardevice();

int x= printmenu();

if (x==3)

{

active=credits();

cleardevice();

}

else if (x==2)

{

active=printhowtoplay();

cleardevice();

}

else if (x==1)

{

active=game();

active = finalgame();

cleardevice();

}

else if (x==4)

{

exit(0);

}

}

}

Printmenu.cpp:

#include "graphics.h"

int printmenu()

{

readimagefile ("menu.jpg",0,0,800,600);

while (1)

{

clearmouseclick(WM\_LBUTTONDOWN);

int x=mousex();

int y=mousey();

if (x>=320 && x<=490 && y>=200 &&y<=240)

{

setcolor(RED);

setfillstyle (SOLID\_FILL,RED);

fillellipse(280,230,15,15);

setcolor(BLACK);

setfillstyle (SOLID\_FILL,BLUE);

fillellipse(280,320,15,15);

fillellipse(280,420,15,15);

fillellipse(280,520,15,15);

if (ismouseclick(WM\_LBUTTONDOWN))

{

clearmouseclick(WM\_LBUTTONDOWN);

return 1;

}

}

else if (x>=300 && x<=500 && y>=300 &&y<=340)

{

setcolor(RED);

setfillstyle (SOLID\_FILL,RED);

fillellipse(280,320,15,15);

setcolor(BLACK);

setfillstyle (SOLID\_FILL,BLUE);

fillellipse(280,230,15,15);

fillellipse(280,420,15,15);

fillellipse(280,520,15,15);

if (ismouseclick(WM\_LBUTTONDOWN))

{

clearmouseclick(WM\_LBUTTONDOWN);

return 2;

}

}

else if (x>=330 && x<=470 && y>=400 &&y<=440)

{

setcolor(RED);

setfillstyle (SOLID\_FILL,RED);

fillellipse(280,420,15,15);

setcolor(BLACK);

setfillstyle (SOLID\_FILL,BLUE);

fillellipse(280,230,15,15);

fillellipse(280,320,15,15);

fillellipse(280,520,15,15);

if (ismouseclick(WM\_LBUTTONDOWN))

{

clearmouseclick(WM\_LBUTTONDOWN);

return 3;

}

}

else if (x>=360 && x<=440 && y>=500 &&y<=540)

{

setcolor(RED);

setfillstyle (SOLID\_FILL,RED);

fillellipse(280,520,15,15);

setcolor(BLACK);

setfillstyle (SOLID\_FILL,BLUE);

fillellipse(280,230,15,15);

fillellipse(280,320,15,15);

fillellipse(280,420,15,15);

if (ismouseclick(WM\_LBUTTONDOWN))

{

clearmouseclick(WM\_LBUTTONDOWN);

return 4;

}

}

else if (x>=39 && x<=174 && y>=545 &&y<=662)

{

//setcolor(RED);

setfillstyle (SOLID\_FILL,BLUE);

fillellipse(280,230,15,15);

fillellipse(280,320,15,15);

fillellipse(280,420,15,15);

fillellipse(280,520,15,15);

if (ismouseclick(WM\_LBUTTONDOWN))

{

clearmouseclick(WM\_LBUTTONDOWN);

return 5;

}

}

else

{

setcolor(RED);

setfillstyle (SOLID\_FILL,BLUE);

fillellipse(280,230,15,15);

fillellipse(280,320,15,15);

fillellipse(280,420,15,15);

fillellipse(280,520,15,15);

}

}

return 0;

}

**Conclusion:**

Due to such a short time in such a busy schedule, it needed tremendous hard work to complete the game. However, it was fun to actually make something from scratch & implement a graphics library for animation & a wide range of tasks with C language for the first time.

Heartiest gratitude to the respected teachers and classmates who helped a lot to make this project.

**References:**

1. Programming in ANSI C By E.Balaguruswami
2. C The Complete Reference (4th Edition) By Herbert Schildt
3. <http://cprogrammingexpert.com/C/Tutorial/graphics.aspx>
4. <http://www.cs.colorado.edu>
5. <http://www.programmingsimplified.com>
6. <http://en.wikipedia.org/>
7. <http://www.mycplus.com/tutorials/c-programming-tutorials/graphics-programming>