|  |
| --- |
| Computer Programming Lab |
| **Project Report** |
| Submitted to:  Dr. Muhammad Asif Hossain Khan  Ms.Anna Fariha |

Submitted by:

Sharmin Mahzabin (03)

Melody Soptaka (43)

**Index of contents:**

-Name of project

-How we got the idea

-Work-plan

-Description of the game

-How to play

-Variables and data types

-User defined function and algorithm

-Graphics library functions

-A flow chart of user-defined functions

-A flow chart of the game

-Source code

Presenting…

**Bouncing Ball**



Bouncing\_Ball: The idea

In the broad arena of computer programming, most prefer contest programming as their field of interest. But we all know, traditional coding despite of being a huge part, there are parts like software sectors and graphics sectors. As this project focuses on our interest for graphics sector, no matter how much reluctant we were, we had to think about it.

Sharmin came along with the idea first. We both were in quite some confusion about doing a simple project like this. But we thought that, It was better to finish our simple game perfectly in time than attempting something that requires much time and has risks.

So the basic idea is to land a object safely from one place to another. It’s simple as it sounds but of course some difficulties might work just fine to take the game to a brand new level.

The game is not dependent on time. As long as put the object on safe places, you survive. And when you fall, you fail.

At first we wanted to make a game like rapid roll by Nokia. But then we thought it’s too simple! So, instead of rolling, we made our ball bouncing! The direction of jump is set by the user. Initial velocity of the ball is constant.



Figure 1: Rapid roll by Nokia



Figure 2: Bouncing ball

Work-plan…

When ma’am asked for the proposals, we didn’t have a plan. We thought about doing something too simple. After submitting the proposal, we both were quite reluctant about making the game. But again when the defense day was fixed, we started working all over. Even this simple game seemed a lot!

Then we decided to work by parts. We learnt about some functions which we didn’t even touch in the graphics lab. We divided the game into quite a few sectors. While Rakhi made the structure and the calculations, I decided the backgrounds and rest of the surroundings.

The Game…

Well, as said before our game is about landing safely on the bars. Here, you’ll see a screen where some bars rise in the background and one of those bars contains the ball on it. When the ball is on a bar or a thorn, an arc is shown which has an indicator inside it, which decides the direction the ball might jump to. The player can change the direction by moving cursor.



While bouncing, if the ball touches a bar and it’s going down, it sits on the bar .But if it touches a bar while going up, it bounces back. The ball bounces back whenever it hits the side walls too.



Figure 3: The ball being prepared for a jump



Figure 4: Ball hitting a wall



Figure 5: Bouncing back

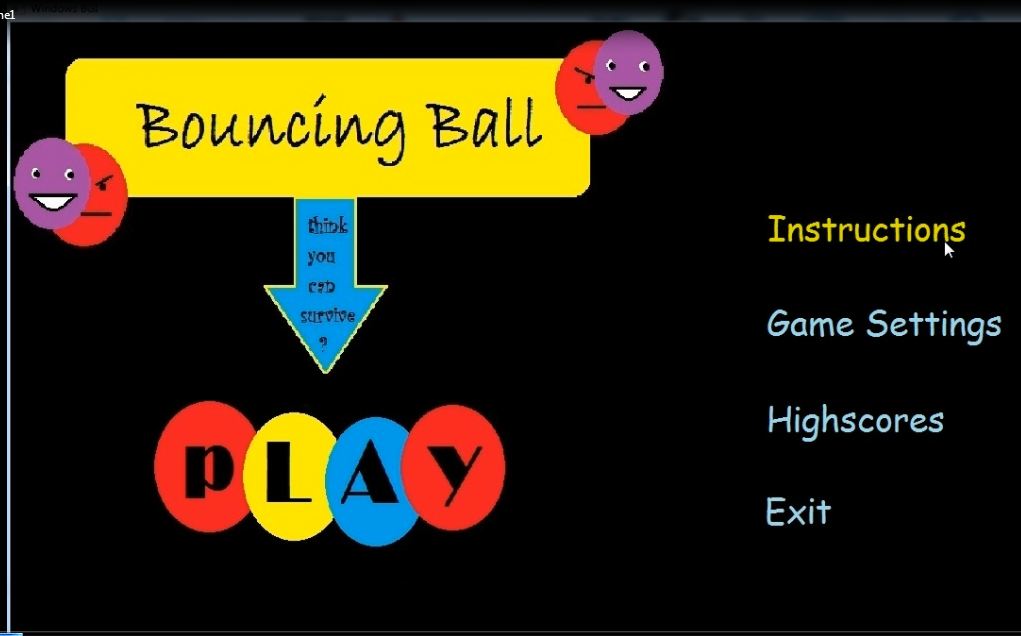
If the ball touches the upper wall or floor, then the game is over. We also arranged a scoreboard which updates itself on each of the safe landings. Each time the ball jumps on a bar , score increases by 10. On the other hand, if it sits on a thorn score decreases by 10 if score is greater than 0. Otherwise the game is over. High scores, instructions are also available.



Figure 6 : Game over

How to play…

Menu :



Game settings :

This game has two levels- Fair bounce and feisty bounce. In feisty bounce , we have added thorns.



Ours is an easy game to play. Pressing the left button of mouse on the “Play” bar starts the game. On the left, is the scoreboard which initially shows zero. On the right there are bars coming randomly and one of those has the ball. When the cursor is moved, direction of the ball changes. One needs to choose a direction and then press left button to jump. One can make the ball jump even when it’s on the air. In that case, direction of jump will be the last direction set by the player when the ball was on a bar.



Variables and Data types:

The variables and data we used were mostly of integer type. But we had to use double and char too. Because the direction of the ball is stored as the slope of it’s velocity and slope is often fractional number. We used char type variables to store the names of top ten scorers. However, we didn’t use any user defined data type. Most of the variables are gloabal.

User Defined Functions and algorithm:

void drawball(int x, int y) :

This function draws a ball the centre of which is (x,y). Radius of the ball is a global variable. This function draws the ball with happy face if the ball is on a bar, an expressionless face if it’s on a thorn and a sad face when it’s on air.

void select(void);

The whole screen is divided in to several regions. This function selects whether a region will be empty or contain a bar or a throne.

The selection of the regions are pseudorandom. Two adacent regions will never contain bars or thorns. One of them must be empty. The number of consecutive empty regions doesn’t exceed 3. If a region contains a thorn, the next non-empty region must contain a bar, not a thorn again!

void drawbar(int y,int row);

This funtion draws bars in the selected regions. The width and height of the bars are global variables. Here, y represents the y co-ordinate of the lowest part of the bar. There will be one or two rows of bars on the screen. The variable ‘row’ is the number of row where the bar will be drawn.

The regions are stored in a two dimensional array. A non-zero element

Represents that this region is non-empty.

In a non-empty region, the probability of its containing a bar is 60%

and 40% for a throne.

void setdirection(void);

This function draws an arc and an indicator around the ball. The indicator is actually a line joining the center of the ball and the cursor. But it’s length is constant regardless of the position of the position of the cursor. The ball cannot jump downwards. In those positions of cursor, the direction is set to 0 degree or 180 degree depending on whether the cursor is in the right or left side of the ball. The direction is stored as the slope of the indiactor in a global variable of double data type.

When the player wants to make the ball jump in 90 degree, the slope of the indicator theoretically becomes infinite. So, we need to manage that special case seperately.

void moveball(void);

This function finds out the next x and y position of the ball depending on the current velocity of the ball.

But sometimes, if we move the ball according to it’s velocity it passes throw the bars or goes outside the screen. But while moving, there must be a moment when the ball hit the bar or the wall. So, in case of bars, we made the ball sit on the bar and change x position which should be changed in that fractional change of time. And in case of side walls, we set the x so that ball just touches the walls and then made the little change in y position.

void bounce(int x, int towards);

Here x is the current x position of the ball and towards is a variable that represents whether the ball should move left, right or downwards after bouncing.

When a ball hits a side wall, the direction of its Vx is reversed. But hits a bar while going up, its Vy is set to 0.

void updatescore(void);

This function shows the updated score on the upper left corner of the screen. Score is stored in a global variable.

void sort(int len);

This function reads the scores made by the players who played earlier and finds the position of current scorer in the list. Length is the the number of topscorers which doesn’t exceed 10. If two people make same scores, then the one who played the game first will obtain better position in the list.

void intro(void);

This is a function that shows the main menu of the game. It shows some buttons on the screen for selecting an option. Then it calls other functions depending on which option is selected.

void showscore(void);

‘When the player selects “Highscore” from the menu, this function is called. It shows the names and scores of top ten scorers.

void showins(void);

When the player selects “Instruction” from the menu, this function is called and it shows a page containing instructions to play the game.

int play(void);

When the “PLAY” button is pressed, this function is called. This function returns 0 when the game is over. It calls other functions to controll the game and changes a global variable depending on whether the ball jumping or staying on a bar.

**Functions of header file “graphics.h” and “winbgim.h:**

delay():

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

fillellipse():

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.

fillpoly():

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.

settextstyle():

This function the style, direction and size of a text.

outtextxy():

This function shows message on a given coordinate.

circle():

This function draws a circle.

readimage():

This function takes only JPEG and BITMAP image.

kbhit():

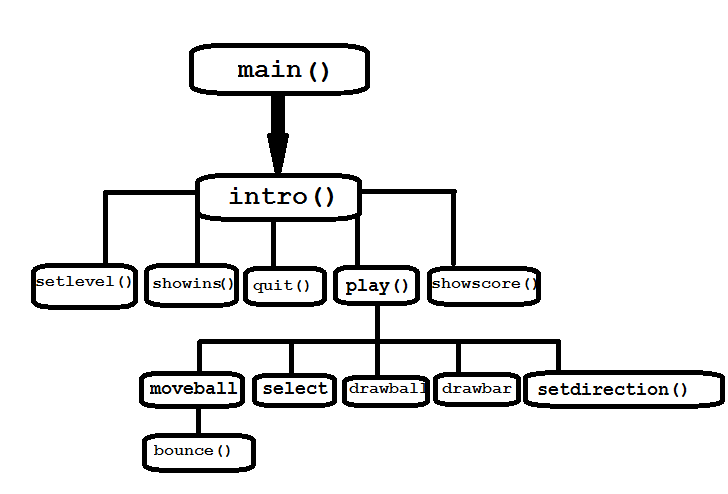
The function returns true (nonzero) if there is a character in the input buffer ready to read. Otherwise it returns false.

In order to work, the user must click in the graphics window.

setfillstyle():

This function imposes the color and pattern of filling.

Flow chart of Functions:



Flow chart of the game:

Program Code:

#include"graphics.h"

#include"winbgim.h"

#include<string.h>

#include<ctime>

#include<stdio.h>

#include<math.h>

#define INF 9999999

#define rightx -9999

#define leftx 9999

#define xmin 200

//#define bounce 4

#define xmax 1024

#define ymax 700

#define ymin 0

#define bar\_width 100

#define bar\_height 2

#define segment 8

#define rad 16

#define sit 1

#define fall 2

#define jump 3

#define vi 20 // vy/vx = slope,m(dir)

// (m\*m+1)\*vx\*vx = vi\*vi

#define g 1 // (g=)vy/vx = slope = dir; vx = vy/slope

int ballstate,ballx,bally,stop,nrow,xpoint,ypoint,vy,vx,yrow,ballregion,score,exiti,click,level,thorn;

char s[20],name[15],players[15][15];

int scores[15];

double dir,jumpto;

int region[2][15]; //whole screen is divided into 9 bar regions (X axis) : 300-400,400-500,...,1200-1300

/\*

region[][]=0 : no bar

1,2,3 : normal bar

4,5 : thorn

6 : bonus

\*/

void drawheart(int x, int y); // x= middle point on x axis

//y = lowest point on y axis

void drawball(int x, int y); //(x,y) is centre

void select(void); //selects whether a region will be empty or contain a bar

void drawbar(int y,int row); /\*draws bars at the selected regions and y co-ordinate

(lowest y), (xmin+(region-1)\*100 = lowest x)

There will be at least 1 and at most 2 rows of bar on the screen

the upper/older row is 1, lower/newer is 0

\*/

void setdirection(void);

void moveball(void);

int obstacle(int x1,int y1, int x2,int y2); //return 0 = no obstacle, return 1 = sit on upper row, return 2 = sit on lower row, return 3 = bounce

void updatescore(void);

void bounce(int x, int towards);

int pos,low;

void sort(int len);

void intro(void);

void showscore(void);

void showins(void);

int play(void);

void setlevel(void);

void quit(void);

void quit(void)

{

FILE \*f;

setfillstyle(SOLID\_FILL,BLACK);

fillellipse(750,400,1300,1300);

if(score<100)

readimagefile("loser.jpg",400,200,800,400);

else if(score<150)

readimagefile("good.jpg",400,200,800,400);

else if(score<200)

readimagefile("verygood.jpg",400,200,800,400);

else

readimagefile("excellent.jpg",400,200,800,400);

setcolor(WHITE);

settextstyle(1,0,20);

sprintf(s,"%d",score);

outtextxy(400,400,"Your Score is ");

outtextxy(660,400,s);

char c;

int i,j;

if((f=fopen("score.txt","r"))==NULL)

{

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

fclose(f);

}

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

if(low >= score)

delay(2000);

if(low<score)

{

i=0;

outtextxy(360,440,"Please enter your name : ");

while(fscanf(f,"%d",&scores[i])==1)

{

fscanf(f,"%s",players[i]);

++i;

}

sort(i);

printf("position %d\n",pos);

if(i>=10)

j=9;

else

j=i;

fclose(f);

i=0;

for(i=0; i<20; ++i)

s[i]=' ';

s[19]='\0';

i=0;

while(1)

{

c=s[i]=getch();

if((c>='a' && c<='z') || (c>='A' && c<='Z') || (c>='0' && c<='9') || c=='.' || c=='\_')

{

++i;

outtextxy(360,470,s);

break;

}

}

while(((c=getch()) != 13 && i<10) || (i==0))

{

if((c>='a' && c<='z') || (c>='A' && c<='Z') || (c>='0' && c<='9') || c=='.' || c=='\_')

{

s[i]=c;

++i;

outtextxy(360,470,s);

}

else if(c == '\b'&& i>0)

{

s[--i]=' ';

outtextxy(360,470,s);

//setfillstyle(SOLID\_FILL,CYAN);

printf("i %d\n",i);

//bar(560+i\*10,470,570+i\*10,510);

}

}

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

fprintf(f,"%d\n",low);

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

{

fprintf(f,"%10d %s\n",scores[i],players[i]);

}

low=i;

fprintf(f,"%10d %s\n",score,s);

fprintf(f,"\n");

i=low;

for(; i<j; ++i)

{

fprintf(f,"%10d %s\n",scores[i],players[i]);

}

fclose(f);

}

}

void setlevel (void)

{

int x,y;

clearmouseclick(WM\_LBUTTONDOWN);

readimagefile("level.jpg",0,0,xmax,ymax);

while(1)

{

x=mousex();

y=mousey();

if(x>=500 && x<=741 && y>=200 && y<=250)

{

readimagefile("fair\_light.jpg",500,200,500+241,200+50);

readimagefile("feisty\_def.jpg",500,300,500+279,300+49);

if(ismouseclick(WM\_LBUTTONDOWN))

{

level=1;

break;

}

}

else if(x>=500 && x<=779 && y>=300 && y<=349)

{

readimagefile("feisty\_light.jpg",500,300,500+279,300+49);

readimagefile("fair\_def.jpg",500,200,500+241,200+50);

if(ismouseclick(WM\_LBUTTONDOWN))

{

level=2;

printf("level %d\n",level);

break;

}

}

else

{

readimagefile("feisty\_def.jpg",500,300,500+279,300+49);

readimagefile("fair\_def.jpg",500,200,500+241,200+50);

if(ismouseclick(WM\_LBUTTONDOWN))

clearmouseclick(WM\_LBUTTONDOWN);

}

}

clearmouseclick(WM\_LBUTTONDOWN);

}

void showins(void)

{

clearmouseclick(WM\_LBUTTONDOWN);

readimagefile("instruct.jpg",0,ymin,xmax,ymax);

while(!ismouseclick(WM\_LBUTTONDOWN ));

clearmouseclick(WM\_LBUTTONDOWN);

}

void showscore(void)

{

char name[15],val[15];

int num,i=0;

FILE \*f;

clearmouseclick(WM\_LBUTTONDOWN);

readimagefile("highscores.jpg",0,ymin,xmax,ymax);

if((f=fopen("score.txt","r")) != NULL)

{

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

while(fscanf(f,"%d",&num)==1)

{

sprintf(val,"%d",num);

outtextxy(800,200+i\*30,val);

fscanf(f,"%s",name);

outtextxy(200,200+i\*30,name);

++i;

}

fclose(f);

}

while(!ismouseclick(WM\_LBUTTONDOWN));

clearmouseclick(WM\_LBUTTONDOWN);

}

void sort(int len)

{

int i,j,k;

for(i=len-1; i>=0; --i)

{

if(score <= scores[i])

{

pos=i+1;

break;

}

}

if(i<0)

pos=0;

if(pos==9)

low=score;

else if(len>=9)

low=scores[8];

else

low=0;

}

void bounce(int x, int towards)

{

vx\*=-1;

if(towards == leftx)

ballx = x-rad;

else

ballx = x+rad;

if(vy>0)

jumpto\*=-1;

}

void updatescore(void)

{

setcolor(WHITE);

settextstyle(1,0,20);

sprintf(s,"%d",score);

outtextxy(10,20,"Score:");

outtextxy(10,70,s);

}

void moveball(void)

{

int oldx,oldy,state,xx,yy,leftreg,rightreg;

oldx=ballx;

oldy=bally;

if(ballstate == sit)

{

bally-=5;

}

if(ballstate == jump)

{

vy+=g;

bally += vy;

ballx+=vx;

if(ballx+rad >= xmax)

{

bounce(xmax,leftx);

}

else if( ballx-rad <= xmin)

{

bounce(xmin,rightx);

}

else

{

if(bally > oldy )

{

if((bally >= yrow && oldy <= yrow ) || (bally >= yrow+4 && oldy <= yrow+4))

{

yy = yrow-rad;

if(yy<oldy)

xx=oldx;

else if(vy)

xx = oldx+((yy-oldy)/((double)(vy)))\*vx;

else

xx=ballx;

ballregion=1+(xx-xmin)/100;

leftreg=1+(xx-xmin-rad+2)/100;

rightreg=1+(xx-xmin+rad-2)/100;

if((region[1][leftreg] != 0 ) || (region[1][rightreg] != 0))

{

bally=yy;

ballx=xx;

if((region[1][leftreg] != 0 && region[1][leftreg] != 4 && region[1][leftreg] != 5) || (region[1][rightreg] != 0 && region[1][rightreg] != 4 && region[1][rightreg] != 5))

thorn=0;

else

thorn=1;

if(ballstate != sit)

{

if(thorn==0)

score+=10;

else

{

if(score>0)

score-=10;

else

{

exiti=1;

return;

}

}

updatescore();

}

ballstate=sit;

}

}

else if(nrow == 2 && (bally >= yrow+400 && oldy <= yrow+400) || (bally >= yrow+400+4 && oldy <= yrow+400+4))

{

yy = yrow-rad+400;

if(yy<oldy)

xx=oldx;

else if(vy)

xx = oldx+((yy-oldy)/((double)(vy)))\*vx;

else

xx=ballx;

leftreg=1+(xx-xmin-rad+2)/100;

rightreg=1+(xx-xmin+rad-2)/100;

if((region[0][leftreg] != 0) || (region[0][rightreg] != 0 ))

{

ballx=xx;

bally=yy;

if((region[0][leftreg] != 0 && region[0][leftreg] != 4 && region[0][leftreg] != 5) || (region[0][rightreg] != 0 && region[0][rightreg] != 4 && region[0][rightreg] != 5))

thorn=0;

else

thorn=1;

if(ballstate != sit)

{

if(thorn==0)

score+=10;

else

{

if(score>0)

score-=10;

else

{

exiti=1;

return;

}

}

updatescore();

}

ballstate=sit;

}

}

}

else if(bally < oldy)

{

if((bally <= yrow && oldy >= yrow ) || (bally <= yrow-4 && oldy >= yrow-4))

{

yy = yrow+rad;

if(vy)

xx = oldx + ((yy-oldy)/((double)(vy)))\*vx;

else

xx=ballx;

ballregion=1+(xx-xmin)/100;

leftreg=1+(xx-xmin-rad+2)/100;

rightreg=1+(xx-xmin+rad-2)/100;

if((region[1][leftreg] != 0 ) || (region[1][rightreg] != 0 ))

{

bally=yy;

ballx=xx;

vy=g;

}

}

else if(nrow == 2 && (bally <= yrow+400 && oldy >= yrow+400) || (bally >= yrow+400-4 && oldy <= yrow+400-4))

{

yy = yrow+rad+400;

if(vy)

xx = oldx + ((yy-oldy)/((double)(vy)))\*vx;

else

xx=ballx;

ballregion=1+(xx-xmin)/100;

leftreg=1+(xx-xmin-rad+2)/100;

rightreg=1+(xx-xmin+rad-2)/100;

if((region[0][leftreg] != 0 ) || (region[0][rightreg] != 0 ))

{

ballx=xx;

bally=yy;

vy=g;

}

}

}

if(ballx+rad >= xmax)

{

bounce(xmax,leftx);

}

else if( ballx-rad <= xmin)

{

bounce(xmin,rightx);

}

}

}

}

void setdirection(void)

{

int x,y,dis,tox,toy,xtoball,ytoball;

double m,n;

x = mousex();

y = mousey();

dis = sqrt((double)((x-(ballx))\*(x-(ballx))+(y-bally)\*(y-bally)));

if(dis > 100)

{

m = 100.0/dis;

n = 1-m;

tox = m\*x+n\*(ballx);

toy = m\*y+n\*(bally);

}

else

{

m=1;

n=1-(dis/100.0);

if(m != n)

{

tox = (m\*x-n\*(ballx))/(m-n);

toy = (m\*y-n\*(bally))/(m-n);

}

else

{

tox=x;

toy=y;

}

}

if(toy > bally)

{

toy = bally;

if(tox > (ballx))

{

tox = (ballx)+100;

}

else

{

tox = (ballx)-100;

}

}

setcolor(BLACK);

if(ballx >= xmin+100)

arc(ballx,bally,0,180,100);

else

{

arc(ballx,bally,0,180-(acos((ballx-xmin)/100.0)\*(180/acos(-1.0))),100);

}

if(tox >= xmin)

line(ballx, bally,tox,toy);

else

line(ballx, bally,xmin,toy);

if(ballx == tox)

dir = INF;

else if(bally == toy && tox >= ballx)

dir = rightx;

else if(bally==toy && tox < ballx)

dir = leftx;

else

dir = ((double)(toy-(bally)))/((ballx)-tox);

}

void drawball(int x, int y)

{

setcolor(BLACK);

setfillstyle(SOLID\_FILL,YELLOW);

fillellipse(x,y,rad,rad);

setcolor(BLACK);

setlinestyle(0,0,2);

if(ballstate==sit && thorn==0)

arc(x,y+3,180,360,5);

else if(ballstate != sit)

arc(x,y+7,20,160,5);

setfillstyle(SOLID\_FILL,BLACK);

fillellipse(x-8,y-5,3,2);

fillellipse(x+8,y-5,3,2);

}

void drawheart(int x, int y)

{

int a[12]= {x,y,x-10,y-20,x-10+5,y-20-5,x,y-20,x+5,y-20-5,x+10,y-20};

setfillstyle(SOLID\_FILL,RED);

fillpoly(6,a);

}

void drawbar(int y,int row)

{

int i,left,x,j;

setfillstyle(SOLID\_FILL,RED);

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

{

if(region[row][i] ==4 || region[row][i] == 5) //thorn

{

setcolor(BLACK);

x= xmin+(i-1)\*100;

for(j=0; j<5; ++j)

{

line(x,y,x+10,y-20);

x+=10;

line(x,y-20,x+10,y);

x+=10;

}

}

else if(region[row][i]) //not thorn

{

left=xmin+(i-1)\*100;

setfillstyle(SOLID\_FILL,RED);

bar(left,y-bar\_height,left+bar\_width,y);

if(region[row][i]==6)

drawheart(left+(bar\_width/2),y-bar\_height);

}

}

}

void select(void)

{

int i,last=1,pick;

for(i=1+rand()%4; i<segment; )

{

if(last < 4)

{

if(level==1)

last = region[0][i]=1+rand()%3;

else

last = region[0][i]=1+rand()%4;

}

else if(last==6)

last = region[0][i]=1+rand()%3;

else

{

pick = rand()%2;

if(!pick) //normal bar

last = region[0][i]=1;

else

last=region[0][i]=1;

}

i = i+ 2+rand()%3;

}

putchar('\n');

}

void intro()

{

int x,y,sel=0;

//clearmouseclick(WM\_LBUTTONDOWN);

while(1)

{

readimagefile("intro\_default.jpg",0,0,xmax,ymax);

while(!ismouseclick(WM\_LBUTTONDOWN ))

{

sel=0;

x=mousex();

y=mousey();

if(x>=600 && x<=600+215 && y>=200 && y<=200+35)

{

readimagefile("instructions\_light.jpg",600,200,600+215,200+35);

sel=2;

}

else

{

readimagefile("instructions\_def.jpg",600,200,600+215,200+35);

}

if(x>=600 && x<=600+252 && y>=300 && y<=300+44)

{

readimagefile("settings\_light.jpg",600,300,600+252,300+44);

sel=3;

}

else

{

readimagefile("settings\_def.jpg",600,300,600+252,300+44);

}

if(x>=600 && x<=600+190 && y>=400 && y<=400+46)

{

readimagefile("highscores\_light.jpg",600,400,600+190,400+46);

sel=4;

}

else

{

readimagefile("highscores\_def.jpg",600,400,600+190,400+46);

}

if(x>=600 && x<=600+71 && y>=500 && y<=500+34)

{

readimagefile("exit\_light.jpg",600,500,600+71,500+34);

sel=5;

}

else

{

readimagefile("exit\_def.jpg",600,500,600+71,500+34);

}

if(x >= 150 && x <= 150+381 && y >= 400 && y <= 400+158 )

{

readimagefile("play\_light.jpg",150,400,150+381,400+158);

sel=6;

}

else

{

readimagefile("play\_def.jpg",150,400,150+381,400+158);

}

}

clearmouseclick(WM\_LBUTTONDOWN);

if(exiti==1)

return;

if(sel==6)

{

play() ;

quit();

clearmouseclick(WM\_LBUTTONDOWN);

}

else if(sel==2)

{

showins();

clearmouseclick(WM\_LBUTTONDOWN);

}

else if(sel==3)

{

setlevel();

clearmouseclick(WM\_LBUTTONDOWN);

}

else if(sel==4)

{

showscore();

clearmouseclick(WM\_LBUTTONDOWN);

}

else if(sel==5)

{

break;

}

}

}

int main()

{

srand(time(0));

initwindow(xmax,ymax);

settextstyle(1,0,20);

level=1;

intro();

return 0;

}

int play(void)

{

int y,x,i,maxy;

char c;

FILE \*f;

//initwindow(xmax,ymax);

exiti=0;

while(1)

{

memset(region,0,sizeof(region));

memset(s,0,sizeof(s));

memset(scores,0,sizeof(scores));

ballstate=ballx=bally=nrow=xpoint=ypoint=vy=vx=yrow=ballregion=score=pos=low=stop=thorn=0;

dir=jumpto=0;

//intro();

if(exiti==1)

{

delay(50);

break;

}

clearmouseclick(WM\_LBUTTONDOWN);

delay(200);

readimagefile("scoreboard.jpg",0,0,xmin,ymax);

setcolor(BLACK);

updatescore();

select();

ballstate = sit;

nrow=1;

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

{

region[1][i]=region[0][i];

if(((region[1][i]>=1 && region[1][i]<=3) || region[1][i]==6) && !ballx && !bally)

{

ballx = xmin+(bar\_width/2)+(i-1)\*100;

bally = ymax+5-rad;

if(region[1][i]==6)

region[1][i]=1;

}

}

maxy=ymax;

while(1)

{

printf("%d %d\n",ballx,1+(ballx-xmin)/100);

nrow=1;

for(y=maxy; y>= ymin+bar\_height; y-=5)

{

//updatescore();

yrow=y;

readimagefile("back.jpg",xmin,ymin,xmax,ymax);

drawbar(y,1); ////for the only or the older/upper row

if(y==300) ////if older\_row(y) == 300, select regions for new row

{

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

region[0][i]=0;

select();

}

if(y <= 300)

{

//draw new row

drawbar(y+400,0);

nrow=2;

}

if(ballstate==sit)

setdirection();

if(ismouseclick(WM\_LBUTTONDOWN ))

{

click=0;

jumpto=dir;

clearmouseclick(WM\_LBUTTONDOWN );

vx= (vi\*vi)/((jumpto\*jumpto)+1);

if(jumpto==0)

{

if(vx>=0)

jumpto=rightx;

else

jumpto=leftx;

}

if(jumpto==INF)

{

ballstate=jump;

vx = 0;

vy=-vi;

}

else if(jumpto == rightx)

{

vy = g+bar\_height;

ballx+=20;

if(ballstate==sit)

{

if(bally<yrow && region[1][1+((ballx-xmin)/100)])

{

if(region[1][1+((ballx-xmin)/100)] != 4 && region[1][(1+((ballx-xmin)/100))] != 5)

thorn=0;

else

thorn=1;

ballstate=sit;

}

else if(nrow==2 && bally>yrow && region[0][1+((ballx-xmin)/100)])

{

if(region[0][1+((ballx-xmin)/100)] != 4 && region[0][(1+((ballx-xmin)/100))] != 5)

thorn=0;

else

thorn=1;

ballstate=sit;

}

else

{

ballstate=jump;

bally= bally+rad+bar\_height+2+4;

}

}

else

{

ballstate=jump;

bally= bally+rad+bar\_height+2+4;

}

vx = vi;

}

else if(jumpto==leftx)

{

vy=g+bar\_height;

ballx-=20;

if(ballx<xmin+rad)

ballx=xmin+rad;

if((ballx+20)/100 == (ballx/100) && ballstate==sit)

{

if(bally<yrow && region[1][1+((ballx-xmin)/100)])

{

if(region[1][1+((ballx-xmin)/100)] != 4 && region[1][(1+((ballx-xmin)/100))] != 5)

thorn=0;

else

thorn=1;

ballstate=sit;

}

else if(nrow==2 && bally>yrow && region[0][1+((ballx-xmin)/100)])

{

if(region[0][1+((ballx-xmin)/100)] != 4 && region[0][(1+((ballx-xmin)/100))] != 5)

thorn=0;

else

thorn=1;

ballstate=sit;

}

else

{

ballstate=jump;

bally= bally+rad+bar\_height+2+4;

}

}

else

{

ballstate=jump;

bally= bally+rad+bar\_height+2+4;

}

vx = -vi;

}

else if(jumpto > 0)

{

vx = sqrt((double)(vx));

vy=-sqrt((double)((vi\*vi)-(vx\*vx)));

ballstate=jump;

}

else if(jumpto < 0)

{

vx = -sqrt((double)vx);

vy=-sqrt((double)((vi\*vi)-(vx\*vx)));

ballstate=jump;

}

}

if(bally <(rad) || bally > ymax-bar\_height+5)

{

if(bally<rad)

bally=rad;

else

bally = ymax-rad;

drawball(ballx,bally);

printf("%d\n",bally);

stop=1;

break;

}

moveball();

if(exiti==1)

{

exiti=0;

return 0;

}

if(y == bally+rad+bar\_height && region[1][1+((ballx-xmin)/100)])

{

if(region[1][1+((ballx-xmin)/100)] != 4 && region[1][(1+((ballx-xmin)/100))] != 5)

thorn=0;

else

thorn=1;

if(ballstate != sit)

{

if(thorn==0)

score+=10;

else

{

if(score>0)

score-=10;

else

{

return 0;

}

}

updatescore();

}

ballstate=sit;

}

else if(nrow==2 && y == bally+rad+bar\_height-400 && region[0][1+((ballx-xmin)/100)] )

{

if(region[0][1+((ballx-xmin)/100)] != 4 && region[0][(1+((ballx-xmin)/100))] != 5)

thorn=0;

else

thorn=1;

if(ballstate != sit)

{

if(thorn==0)

score+=10;

else

{

if(score>0)

score-=10;

else

{

return 0;

}

}

updatescore();

}

ballstate=sit;

}

ballregion=1+(ballx-xmin)/100;

drawball(ballx,bally);

delay(30);

}

if(stop)

break;

maxy=y+400;

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

{

region[1][i]=region[0][i];

}

}

return 0;

}

exiti=0;

return 0;

//intro();

//if(exiti==1)

//break;

exiti=0;

return 0;

}