// THREE CLICK.cpp : Defines the entry point for the console application.

#include "stdafx.h"

#include <Windows.h>

# include <glut.h>

#include<stdio.h>

#include<conio.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*GLOBALS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int tx = 10, ty = 10, dy = 2, dx =2;

double spin = 0.0, scale = 1.0;

int Radius = 10;

int click=0;

bool animate = false, won = false, timeup = false;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

|

| tx,ty... Translate x,y

| dx,dy... change in x,y

\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*display || The Default drawing function called at start up...\*/

void maze() {

glLineWidth(5.0f);

glBegin(GL\_LINE\_LOOP);

glVertex2f(5,5);

glVertex2f(5,395);

glVertex2f(595,395);

glVertex2f(595,5);

glEnd();

//Barrier 1

glBegin(GL\_LINES);

glVertex2f(100, 150);

glVertex2f(100,350);

//Barrier 2

glVertex2f(150, 50);

glVertex2f(250,50);

//Barrier 3

glVertex2f(350, 280);

glVertex2f(480,280);

//Barrier 4

glVertex2f(350, 5);

glVertex2f(350,100);

//Barrier 5

glVertex2f(380, 330);

glVertex2f(380,370);

glEnd();

}

void hole() {

/\* the hole \*/

glPushMatrix();

glTranslatef(550, 30, 0);

glScalef(1.5,1.5, 0.0);

glColor3f(.2,.2,.2);

glutSolidSphere(10, 12,12);

glColor3f(1.0,1.0,1.0);

glPopMatrix();

glFlush();

}

void displayCall(int a,int b,int c,char \*string) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glEnable(GL\_DEPTH\_TEST);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-2.0, 2.0, -2.0, 2.0, -2.0, 500.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(a, b, c, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glScalef(.005,.005,.005);

glRotatef(20, 0, 1, 0);

glRotatef(30, 0, 0, 1);

glRotatef(5, 1, 0, 0);

glTranslatef(-450, 0, 0);

glColor3f(1,1,1);

for(int unsigned i=0;i<strlen(string);i++)

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, string[i]);

glutSwapBuffers();

} /\* end func displayCall \*/

void display(void)

{ glClear (GL\_COLOR\_BUFFER\_BIT);

maze();

hole();

if(!won){

glPushMatrix(); /\* Put onto matrix stack? \*/

glTranslatef(tx,ty,0); /\* Move to/from tx,ty \*/

glRotatef(spin, 0.0, 0.0, 1.0); /\* rotate about z axis by spin degrees \*/

glScalef(scale, scale, 0.0);

glutSolidSphere(10, 12,12);

glPopMatrix(); /\* take off of stack\*/

glutSwapBuffers(); /\* swap drawn w/ shown \*/

}

else if(won)

{

displayCall(1,1,1," you won!");

Sleep(3000); //1000=1 second

exit(0);

}

}

void chkBarriers() {

// to bounce off the barriers.

/\* barrier 1 at (100, 150), (100,350) - Vertical Barrier\*/

if((ty+Radius)<=350 &&(ty + Radius)>=150 && (tx+Radius==100 || tx-Radius ==100)){

dx\*= -1;/\* reverse dx!\*/

if(dx<0)

tx -= (tx + Radius) - 100;

else

tx += (tx + Radius) - 100;

}

/\* barrier 2 at (150, 50), (250,50) - Horizontal Barrier\*/

if((tx+Radius)<=250 &&(tx + Radius)>=150 && (ty+Radius==50 || ty-Radius ==50)){

dy\*= -1;/\* reverse dy!\*/

if(dy<0)

ty -= (ty + Radius) - 50;

else

ty += (ty + Radius) - 50;

}

/\* barrier 3 at (350,280), (500,280) - Horizontal Barrier \*/

if((tx+Radius)<=480 &&(tx + Radius)>=350 && (ty+Radius==280 || ty-Radius ==280)){

dy\*= -1;/\* reverse dy!\*/

if(dy<0)

ty -= (ty + Radius) - 280;

else

ty += (ty + Radius) - 280;

}

/\* barrier 4 at (350, 5), (350,100) - Vertical Barrier\*/

if((ty+Radius)<=100 &&(ty + Radius)>=5 && (tx+Radius==350 || tx-Radius ==350)){

dx\*= -1;/\* reverse dx!\*/

if(dx<0)

tx -= (tx + Radius) - 350;

else

tx += (tx + Radius) - 350;

}

/\* barrier 5 at (380, 330), (380,370) - Vertical Barrier\*/

if((ty+Radius)<=370 &&(ty + Radius)>=330 && (tx+Radius==380 || tx-Radius ==380)){

dx\*= -1;/\* reverse dx!\*/

if(dx<0)

tx -= (tx + Radius) - 380;

else

tx += (tx + Radius) - 380;

}

}

void chkHole() {

if((tx+Radius <=570 && tx+Radius>=535) && (ty+Radius <=50 && ty+Radius>=10)){

animate = false;

/\*printf("You won!!!!!!!!!!!");\\*/

won=true;

}

}

/\*BounceDisplay || The Idle Function... and bouncing function.\*/

void BounceDisplay(void)

{

if(animate)

{

tx += dx; /\* Move center over \*/

ty += dy; /\* by dx,dy (applied by translation) \*/

/\* to bounce of the walls \*/

chkHole();

if(!won){

chkBarriers();

/\* Now check for the maze walls, for collision and bouncing \*/

/\* Right wall \*/

if((tx + Radius) > 600)

{

dx\*= -1;/\* reverse dx!\*/

tx -= (tx + Radius) - 600;

}

/\* Left Wall \*/

if((tx - Radius) < 0 )

{

dx\*= -1;

tx += -1 \* (tx - Radius); /\* Boing! \*/

chkBarriers();

}

/\* Bottom wall ,since window coordinate start from TOP-LEFT(0,0)\*/

if((ty + Radius) > 400)

{

dy \*= -1;

ty -= (ty + Radius) - 400; /\*Boing! same as dx \*/

}

/\*Top wall\*/

if ((ty - Radius) < 0 )

{

dy \*= -1;

ty += -1 \* (ty - Radius); /\*Boing! same as dx \*/

}

Sleep(10); // to slow things down

}// end if not won

glutPostRedisplay();/\* Make Sure you redraw everything \*/

}// end if animate

}

/\*init() || GL initialization... called at start up...\*/

void init (void)

{

glClearColor (0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, 600, 0, 400, -1.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\

| mouse() || GLUT event loop handler nifty huh?

\\*\*\*\*\*\*\*\*\*\*/

void mouse(int button, int state, int Mx, int My) {

/\*flip the y coordinate... because screen origin starts from top right! so no need to change tx and change ty as ty=400-My\*/

switch(button){

case GLUT\_LEFT\_BUTTON:

if (state == GLUT\_DOWN)

{

click+=1;

if(click!=4)

{

animate = true;

won = false;

if(Mx%2!=0) Mx+=1;

if(My%2!=0) My+=1;

tx = Mx; ty = 400-My;

printf("\n%d %d", tx, ty);

break;

}

else

{

displayCall(3,4,5," You Lose!");

Sleep(2000);

exit(0);

}

}

case GLUT\_RIGHT\_BUTTON:

if(state == GLUT\_DOWN){

break;

}

default:

break;

} /\* switch \*/

}

/\* main \*/

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

printf("Left - Click on the screen to start/ change position \n Lead the Ball to hole in 3 CLICKS ");

glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize (600, 400);

glutInitWindowPosition (300, 300);

glutCreateWindow ("3 clicks");

init();

glutDisplayFunc(display);

glutIdleFunc(BounceDisplay);

glutMouseFunc(mouse);

glutMainLoop();

return 0;/\* ANSI C requires main to return int. \*/

}