//

// main.cpp

// BouncyBallUpdated

//

// Created by ARAFAA on 11/24/19.

// Copyright © 2019 ARAFAA. All rights reserved.

//

// Includes

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <cmath>

#include<GLUT/glut.h>

///////////////////////////////////// GLOBAL VARIABLES ///////////////////////////////////

**float** rotAng;

GLboolean flag = **true**;

GLboolean flag2 = **true**;

GLboolean shoot = **false**;

GLboolean turnDone = **true**;

GLboolean cannonMotion = **true**;

GLboolean firstMotion = **true**;

GLboolean reflectLeft = **false**;

GLboolean reflectRight = **false**;

GLboolean c = **true**;

GLint turn = 0;

GLint ref1 = -1;

GLint ref2 = -1;

GLint i = 0;

GLint score1Total[] = { 0,0,0 };

GLint score1Final=0;

GLint score2Final =0;

GLint score2Total[] = { 0,0,0 };

GLfloat yc = -0.8;

GLfloat zc = 0;

GLfloat yf = 0;

GLfloat zf = 3.5;

///////////////////////////////////////////////// Colors ///////////////////////////////////////////////

GLfloat white[] = { 1,1,1 };

GLfloat red[] = { 1,0,0 };

GLfloat green[] = { 0,1,0 };

GLfloat blue[] = { 0,0,1 };

GLfloat Cannonbase1[] = { 0.3, 0.3, 0.3 };

GLfloat Cannonbase2[] = { 0.5, 0.5, 0.5 };

GLfloat turquoise[] = { 0.1,0.1,0.1 };

GLfloat black[] = { 0,0,0 };

GLfloat cyan[] = { 0.6,0,0.5 };

///////////////////////////////////////////////////////////////////////////////////////// CANNON //////////////////////////////////////////////////////////////////////////////////////////////////////////////

**void** scoredisplay(**int** posx, **int** posy, **int** posz, **int** space\_char, **int** scorevar)

{

**int** j = 0, p, k;

GLvoid\* font\_style1 = GLUT\_BITMAP\_TIMES\_ROMAN\_24;

p = scorevar;

j = 0;

k = 0;

**while** (p > 9)

{

k = p % 10;

glRasterPos3f(posx, posy, posz + (j \* space\_char));

glutBitmapCharacter(font\_style1, 48 + k);

j++;

p /= 10;

}

glRasterPos3f(posx, posy, posz + (j \* space\_char));

glutBitmapCharacter(font\_style1, 48 + p);

}

**void** print(**int** x, **int** y, **int** z, **char**\* string)

{

**int** len, i;

glRasterPos3f(x, y, z);

len = (**int**)strlen(string);

**for** (i = 0; i < len; i++)

{

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, string[i]);

}

}

**void** update\_Cannon(GLfloat yc, GLfloat zc) {

**if** (zc < -2) {

zc = -2;

}

**if** (zc > 2) {

zc = 2;

}

**if** (yc < -2) {

yc = -2;

}

**if** (yc > 2) {

yc = 2;

}

}

//////////////////////////////////////////////////////////////////////////////// CAMERA //////////////////////////////////////////////////////////////////////////////////////////////////////////

**class** CameraUpdated {

GLfloat x;

GLfloat y;

GLfloat z;

**public**:

CameraUpdated() : x(20.5), y(3.5), z(3.5) {}

**double** getx() {

**return** (x);

}

**void** setx(**double** k) {

x = k;

}

**void** sety(**double** k) {

y = k;

}

**void** setz(**double** k) {

z = k;

}

**double** gety() {

**return** y;

}

**double** getz() {

**return** (z);

}

**void** XmovePlus() {

**if** (x < 30) {

x = x + 0.5;

}

}

**void** XmoveMinus() {

**if** (x > 2)

x = x - 0.5;

}

**void** YmovePlus() {

**if** (y < 30) {

y = y + 0.5;

}

}

**void** YmoveMinus() {

**if** (y > -10)

y = y - 0.5;

}

**void** ZmovePlus() {

**if** (z < 30) {

z = z + 0.5;

}

}

**void** ZmoveMinus() {

**if** (z > -10)

z = z - 0.5;

}

};

CameraUpdated camera2;

**void** key2(**unsigned** **char** key1, **int** x, **int** y) {

**if** (!c) {

**if**(key1=='v' ){

camera2.XmovePlus();}

**if**(key1=='z' ){

camera2.XmoveMinus();

}

**if**(key1=='u' ){

camera2.YmovePlus();}

**if**(key1=='t' ){

camera2.YmoveMinus();

}

**if**(key1=='s' ){

camera2.ZmovePlus();}

**if**(key1=='a' ){

camera2.ZmoveMinus();

}

**if**(key1=='y'){

camera2.XmoveMinus();

camera2.ZmoveMinus();

}

**if**(key1=='x'){

camera2.YmoveMinus();

camera2.ZmoveMinus();

}

**if**(key1=='z'){

camera2.XmoveMinus();

camera2.YmoveMinus();

}

}

**if** (key1 == 'f') {

shoot = **true**;

}

**if** (key1 == 'c') {

c = !c;

}

glutPostRedisplay();

}

/////////////////////////////////////////////////////////////////////////////// BALLS /////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**void** setScore(**float** x, **float** y, **float** z) {

**if** (i < 3) {

**if** (z <= 0.4) { //Right

**if** (**int**(x) % (2) != 0 && **int**(y) % 2 != 0) { ///blue

score1Total[i] -= 5;

std::cout << 567;

} **if** (**int**(x) % (2) == 0 && **int**(y) % 2 != 0) { //green

score1Total[i] += 10;

std::cout << 568;

} **if** (**int**(x) % (2) != 0 && **int**(y) % 2 == 0) { //white

score1Total[i] += 15;

std::cout << 569;

} **if** (**int**(x) % (2) == 0 && **int**(y) % 2 == 0) { //red

score1Total[i] += 20;

std::cout << 570;

}

}

**else** **if** (y <= 0.4) {//down

**if** (**int**(x) % (2) != 0 && **int**(z) % 2 != 0) { ///blue

score1Total[i] -= 5;

std::cout << 567;

} **if** (**int**(x) % (2) != 0 && **int**(z) % 2 == 0) { //green

score1Total[i] += 10;

std::cout << 568;

} **if** (**int**(x) % (2) == 0 && **int**(z) % 2 != 0) { //white

score1Total[i] += 15;

std::cout << 569;

} **if** (**int**(x) % (2) == 0 && **int**(z) % 2 == 0) { //red

score1Total[i] += 20;

std::cout << 570;

}

}

**else** **if** (z >= 6.5) {//left

**if** (**int**(x) % (2) != 0 && **int**(y) % 2 != 0) { ///blue

score1Total[i] -= 5;

std::cout << 567;

} **if** (**int**(x) % (2) == 0 && **int**(y) % 2 != 0) { //green

score1Total[i] += 10;

std::cout << 568;

} **if** (**int**(x) % (2) != 0 && **int**(y) % 2 == 0) { //white

score1Total[i] += 15;

std::cout << 569;

} **if** (**int**(x) % (2) == 0 && **int**(y) % 2 == 0) { //red

score1Total[i] += 20;

std::cout << 570;

}

}

**else** **if** (y >= 6.5) {//up

**if** (**int**(x) % (2) != 0 && **int**(z) % 2 != 0) { ///blue

score1Total[i] -= 5;

std::cout << 567;

} **if** (**int**(x) % (2) != 0 && **int**(z) % 2 == 0) { //green

score1Total[i] += 10;

std::cout << 568;

} **if** (**int**(x) % (2) == 0 && **int**(z) % 2 != 0) { //white

score1Total[i] += 15;

std::cout << 569;

} **if** (**int**(x) % (2) == 0 && **int**(z) % 2 == 0) { //red

score1Total[i] += 20;

std::cout << 570;

}

}

}

**if** (i >= 3 && i < 6) {

**if** (z <= 0.4) { //Right

**if** (**int**(x) % (2) != 0 && **int**(y) % 2 != 0) { ///blue

score2Total[i-3] -= 5;

std::cout << 667;

} **if** (**int**(x) % (2) == 0 && **int**(y) % 2 != 0) { //green

score2Total[i - 3] += 10;

std::cout << 668;

} **if** (**int**(x) % (2) != 0 && **int**(y) % 2 == 0) { //white

score2Total[i - 3] += 15;

std::cout << 669;

} **if** (**int**(x) % (2) == 0 && **int**(y) % 2 == 0) { //red

score2Total[i - 3] += 20;

std::cout << 670;

}

}

**else** **if** (y <= 0.4) {//down

**if** (**int**(x) % (2) != 0 && **int**(z) % 2 != 0) { ///blue

score2Total[i - 3] -= 5;

std::cout << 667;

} **if** (**int**(x) % (2) != 0 && **int**(z) % 2 == 0) { //green

score2Total[i - 3] += 10;

std::cout << 668;

} **if** (**int**(x) % (2) == 0 && **int**(z) % 2 != 0) { //white

score2Total[i - 3] += 15;

std::cout << 669;

} **if** (**int**(x) % (2) == 0 && **int**(z) % 2 == 0) { //red

score2Total[i - 3] += 20;

std::cout << 670;

}

}

**else** **if** (z >= 6.5) {//left

**if** (**int**(x) % (2) != 0 && **int**(y) % 2 != 0) { ///blue

score2Total[i - 3] -= 5;

std::cout << 667;

} **if** (**int**(x) % (2) == 0 && **int**(y) % 2 != 0) { //green

score2Total[i - 3] += 10;

std::cout << 668;

} **if** (**int**(x) % (2) != 0 && **int**(y) % 2 == 0) { //white

score2Total[i - 3] += 15;

std::cout << 669;

} **if** (**int**(x) % (2) == 0 && **int**(y) % 2 == 0) { //red

score2Total[i - 3] += 20;

std::cout << 670;

}

}

**else** **if** (y >= 6.5) {//up

**if** (**int**(x) % (2) != 0 && **int**(z) % 2 != 0) { ///blue

score2Total[i - 3] -= 5;

std::cout << 667;

} **if** (**int**(x) % (2) != 0 && **int**(z) % 2 == 0) { //green

score2Total[i - 3] += 10;

std::cout << 668;

} **if** (**int**(x) % (2) == 0 && **int**(z) % 2 != 0) { //white

score2Total[i - 3] += 15;

std::cout << 669;

} **if** (**int**(x) % (2) == 0 && **int**(z) % 2 == 0) { //red

score2Total[i - 3] += 20;

std::cout << 670;

}

}

}

}

**class** Ball {

**double** radius;

GLfloat\* color;

**double** maxh;

**double** maxX;

**double** x;

**double** y;

**double** z;

**double** directionY;

**double** directionX;

**public**:

Ball(**double** r, GLfloat\* c, **double** h, **double** x, **double** z) :

radius(r), color(c), maxh(6.5), maxX(10), directionY(0.8), directionX(-1.3), y(h), x(x), z(z) {

}

**float** getRadiusBall() {

**return** radius;

}

**float** getBallX() {

**return** x;

}

**float** getBallY() {

**return** y;

}

**float** getBallZ() {

**return** z;

}

**void** update() {

**if** (shoot) {

**if** (firstMotion) {

cannonMotion = **false**;

x = x + 0.05 \* directionX;

z = z + 0.05 \* zc;

y = y + 0.05 \* (0.8 + yc);

**if** (c) {

camera2.setx(x + 10.5);

// camera2.sety(y);

camera2.setz(z);

}

**if** (x < radius) {

x = 10;

z = 3.5;

y = 1.8;

yc = -0.8;

zc = 0;

ref1 = -1;

ref2 = -1;

i++;

shoot = **false**;

firstMotion = **false**;

cannonMotion = **true**;

camera2.setx(20.5);

camera2.sety(3.5);

camera2.setz(3.5);

// sndPlaySound(TEXT("splat.wav"), SND\_ASYNC);

}

}

**if** (z<radius || y<radius || z>maxh || y>maxh) {

firstMotion = **false**;

setScore(x, y, z);

}

**if** (!firstMotion) {

**if** (c) {

camera2.setx(x + 10.5);

// camera2.sety(y);

camera2.setz(z);

}

**if** (z > maxh) {

ref1++;

}

**else** **if** (z < radius) {

ref1++;

}

**if** (y > maxh) {

ref2++;

}

**else** **if** (y < radius) {

ref2++;

}

/////////////////////////////////////////////////////////////////////

**if** (ref1 % 2 == 0) {

z = z - 0.05 \* zc;

}

**else** **if** (ref1 % 2 != 0) {

z = z + 0.05 \* zc;

}

**if** (yc > -0.8) {

**if** (ref2 % 2 != 0) {

y = y - 0.05 \* (-2.8 - yc);

}

**else** **if** (ref2 % 2 == 0) {

y = y + 0.05 \* (-2.8 - yc);

}

}

**else** **if** (yc < -0.8) {

**if** (ref2 % 2 != 0) {

y = y + 0.05 \* (-2.8 - yc);

}

**else** **if** (ref2 % 2 == 0) {

y = y - 0.05 \* (-2.8 - yc);

}

}

**else** **if** (yc == -0.8) {

y = y + 0.05 \* (0.8 + yc);

}

x = x + 0.05 \* directionX;

**if** (x > maxX) {

x = maxX - 1;

}

**else** **if** (x < radius) {

x = 10;

z = 3.5;

y = 1.8;

yc = -0.8;

zc = 0;

ref1 = -1;

ref2 = -1;

i++;

shoot = **false**;

firstMotion = **true**;

cannonMotion = **true**;

camera2.setx(20.5);

camera2.sety(3.5);

camera2.setz(3.5);

// sndPlaySound(TEXT("splat.wav"), SND\_ASYNC);

}

}

}

**if** (zc < -2) {

zc = -2;

}

**if** (zc > 2) {

zc = 2;

}

**if** (yc < -2) {

yc = -2;

}

**if** (yc > 2) {

yc = 2;

}

glPushMatrix();

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, color);

glTranslated(x, y, z);

glutSolidSphere(radius, 30, 30);

glPopMatrix();

}

};

Ball balls[]{ Ball(0.3,cyan,1.8,10,3.5), Ball(0.3, red, 1.8,10,3.5), Ball(0.3, blue, 1.8,10,3.5),Ball(0.3,cyan,1.8,10,3.5), Ball(0.3, red, 1.8,10,3.5), Ball(0.3, blue, 1.8,10,3.5),Ball(0.3, red, 1.8,10,3.5) };

////////////////////////////////////////////////////////////////////////////////////////// WALLS /////////////////////////////////////////////////////////////////////////////////////////////////////

**class** ColoredWall {

**int** displaylistid;

**int** width;

**int** depth;

**public**:

ColoredWall(**int** width, **int** depth) :width(width), depth(depth) {}

**double** centerx() {

**return** width / 2;

// return 4;

}

**double** centerz() {

**return** depth / 2;

// return 4;

}

**void** create() {

displaylistid = glGenLists(1);

glNewList(displaylistid, GL\_COMPILE);

glBegin(GL\_QUADS);

glNormal3d(0, 1, 0);

**for** (**int** x = 0; x < width - 1; x++) {//floor

**for** (**int** z = 0; z < depth - 1; z++) {

**if** (flag) {

**if** (x % 2 == 0) {

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, blue);

}

**else**

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, white);

}

**if** (!flag) {

**if** (x % 2 == 0) {

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, green);

}

**else**

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, red);

}

glVertex3f(x, 0, z);

glVertex3f(x + 1, 0, z);

glVertex3f(x + 1, 0, z + 1);

glVertex3f(x, 0, z + 1);

flag = (!flag);

}

flag = (!flag);

}

**for** (**int** x = 0; x < width - 1; x++) {//floor

**for** (**int** z = 0; z < depth - 1; z++) {

**if** (flag) {

**if** (x % 2 == 0) {

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, blue);

}

**else**

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, white);

}

**if** (!flag) {

**if** (x % 2 == 0) {

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, green);

}

**else**

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, red);

}

glVertex3f(x, 7, z);

glVertex3f(x + 1, 7, z);

glVertex3f(x + 1, 7, z + 1);

glVertex3f(x, 7, z + 1);

flag = (!flag);

}

flag = (!flag);

}

**for** (**int** x = 0; x < width - 3; x++) { //blank

**for** (**int** z = 0; z < depth - 1; z++) {

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, black);

glVertex3f(0, x + 1, z);

glVertex3f(0, x + 1, z + 1);

glVertex3f(0, x, z + 1);

glVertex3f(0, x, z);

}

}

**for** (**int** x = 0; x < width - 1; x++) { //right

**for** (**int** z = 0; z < depth - 1; z++) {

**if** (!flag2) {

**if** (x % 2 == 0) {

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, white);

}

**else**

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, red);

}

**if** (flag2) {

**if** (x % 2 == 0) {

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, blue);

}

**else**

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, green);

}

glVertex3f(x, z, 0);

glVertex3f(x + 1, z, 0);

glVertex3f(x + 1, z + 1, 0);

glVertex3f(x, z + 1, 0);

flag2 = (!flag2);

}

flag2 = (!flag2);

}

**for** (**int** x = 0; x < width - 1; x++) { //left

**for** (**int** z = 0; z < depth - 1; z++) {

**if** (!flag2) {

**if** (x % 2 == 0) {

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, white);

}

**else**

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, red);

}

**if** (flag2) {

**if** (x % 2 == 0) {

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, blue);

}

**else**

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, green);

}

glVertex3f(x, z, 7);

glVertex3f(x + 1, z, 7);

glVertex3f(x + 1, z + 1, 7);

glVertex3f(x, z + 1, 7);

flag2 = (!flag2);

}

flag2 = (!flag2);

}

glEnd();

glEndList();

}

**void** draw() {

glCallList(displaylistid);

}

};

ColoredWall checkboard(10, 8);

///////////////////////////////////////////////////////////////////// // DISPLAY /////////////////////////////// ////////////////////////////////////////////////////////////////////////////////////////////

**void** Display(**void**) {

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

glLoadIdentity();

gluLookAt(camera2.getx(), camera2.gety(), camera2.getz(), balls[0].getBallX(), 3, balls[0].getBallZ(), 0.0, 1.0, 0.0);

**if** (i < 6) {

checkboard.draw();

print(5, 7, 12, "Player 1 SCORE");

scoredisplay(5, 6, 10, 1, score1Total[0]);

scoredisplay(5, 5, 10, 1, score1Total[1]);

scoredisplay(5, 4, 10, 1, score1Total[2]);

scoredisplay(5, 6, -5, 1, score2Total[0]);

scoredisplay(5, 5, -5, 1, score2Total[1]);

scoredisplay(5, 4, -5, 1, score2Total[2]);

print(5, 7, -3, "Player 2 SCORE");

print(5.5, 8, -2, "Bouncing Ball 43");

print(5.5, 2, 13, "Created By AR(k)AFAA");

/\*print(5, 5, 13, "Instructions:");

print(5, 4.5, 13, "1.Press 'S' To Shoot");

print(5, 3, 13, "2.Press UP/DOWN/RIGHT/LEFT");

print(5, 2.5, 13, "To Adjust Cannon Position And");

print(5, 1, 13, "Ball Direction");

print(5, 0, 13, "3.The Score Is Calculated By ");

print(5, -1, 13, "Hitting The Bricks");

print(5, -2, 13, "4.Each Player Has 3 Rounds");\*/

**if** (c) {

print(5, 3, -2, "Normal Camera Mode is On");

print(5, 2, -2, " ");

}

**else** **if** (!c) {

print(5, 3, -2, "Enhanced Camera Mode is On");

print(5, 2, -2, " ");

}

/\*print(5, 5, -3, "5.Camera: Simple Mode By Clicking");\*/

/\*print(5, 4, -3, "'j':Go To X-axis");

print(5, 3, -3, "'k':Go To Y-axis");

print(5, 2, -3, "'l':Go To Z-axis");

print(5, 1, -3, "'T':+X");

print(5, 0, -3, "'Y':-X");

print(5, -1, -3, "'U':+Y");

print(5, -1, -5, "'I':-Y");

print(5, -2, -3, "'O':+Z");

print(5, -2, -5, "'P':-Z");\*/

**if** (i < 3) {

print(9, -1, 7, "YOUR TURN");

}

**else** {

print(9, -1, 2, "YOUR TURN");

}

glPushMatrix(); // Cannon Direction

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, Cannonbase1);

glBegin(GL\_POLYGON);

glVertex3f(11.3, 1, 3);

glVertex3f(11.3, 1, 4);

glVertex3f(10, 1.8 + yc, 3.5 + zc);

// glVertex3f(10, 1.8 + yc, 3.2 + zc);

glEnd();

glPopMatrix();

glPushMatrix(); // Cannon Base

glMaterialfv(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, Cannonbase1);

glTranslatef(11.3, 0.7, 3.5);

glScalef(1, 0.5, 1);

glutSolidCube(1);

glPopMatrix();

balls[0].update();

}

**else** **if** (i == 6) {

score1Final=score1Total[0]+score1Total[1]+score1Total[2];

score2Final=score2Total[0]+score2Total[1]+score2Total[2];

scoredisplay(5,4,-5,1,score2Final);

scoredisplay(5,4,10,1,score1Final);

print(5, 3, 10, "Created by AR(K)AFAA");

**if** (score1Final > score2Final) {

print(5, 5, 3, "GAME OVER!!");

print(5, 5, -2, "PLAYER 2 LOSES");

print(5, 5, 12, "PLAYER 1 WINS");

}

**else** **if** (score2Final > score1Final) {

print(5, 3, 3, "GAME OVER!!");

print(5, 5, 12, "PLAYER 1 LOSES");

print(5, 5, -2, "PLAYER 2 WINS");

}

**else** {

print(5, 3, 5, "IT'S A TIE!!");

print(5, 5, -2, "GAME OVER!!");

print(5, 5, 12, "GAME OVER!!");

}

}

glFlush();

glutSwapBuffers();

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**void** reshape(GLint w, GLint h) {

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(40.0f, GLfloat(w) / GLfloat(h), 1.0f, 150.0f);

glMatrixMode(GL\_MODELVIEW);

}

**void** timer(**int** v) {

glutPostRedisplay();

glutTimerFunc(1000 / 60, timer, v);

}

//////////////////////////////////////////////////////////////////////////////////////// CANNON Directiom From Player ///////////////////////////////////////////////////////////////////////////////////////////

**void** special(**int** key, **int**, **int**) {

**if**(cannonMotion){

**switch** (key) {

**case** GLUT\_KEY\_LEFT:zc += 0.05; update\_Cannon(yc, zc); **break**;

**case** GLUT\_KEY\_RIGHT:zc -= 0.05; update\_Cannon(yc, zc); **break**;

**case** GLUT\_KEY\_UP:yc += 0.05; update\_Cannon(yc, zc); **break**;

**case** GLUT\_KEY\_DOWN:yc -= 0.05; update\_Cannon(yc, zc); **break**;

}

}

glutPostRedisplay();

}

///////////////////////////////////////////////////////////////////////////////////////// MAIN ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowPosition(50, 50);

glutInitWindowSize(1600, 900);

glutCreateWindow("Bouncy Ball");

glutDisplayFunc(Display);

glutReshapeFunc(reshape);

glutSpecialFunc(special);

glutTimerFunc(1000 / 60, timer, 0);

glutKeyboardFunc(key2);

glClearColor(0.7, 0.7, 0.7, 0.0f);

glEnable(GL\_DEPTH\_TEST);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, white);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, white);

GLfloat lightPos[] = { 5,5,5,0 };

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPos);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, white);

glMaterialf(GL\_FRONT, GL\_SHININESS, 50);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

checkboard.create();

glutMainLoop();

**return**(0);

}