410262327 資工三乙 黃予桐

**題目:**

期中考替代作業:2D打磚塊遊戲

**程式架構:**

在全域變數新增玩家橫條，所有磚塊，以及移動方塊球，及一些判斷資訊cnt計算關卡判斷，level場景…。

Renderscene 函式裡，包含初始化所有物件也就是橫條、磚塊、球的顏色，大小成型，再計算wincnt進入第二關 並且更新第二關背景顏色，cnt結束第二關。當球碰觸到視窗下方邊緣顯示出輸掉遊戲字樣，反之若兩關磚塊都消失則顯示出贏的字樣。

Updateautosquare 函式裡做出當球碰到磚塊時，取消磚塊，並將球的行進方向進行更新。

Specialkey 進行橫條左右操作。

Init 顯示視窗 且第一關黑色背景。

**討論:**

從一開始的環境就發生了很多問題，畢竟是第一次使用opengl，之後大略了解後，看完程式碼，對照講義的解說，其實不難理解，就是先宣告方塊的起始x y值，並且給予常跟寬，有需要移動的會多一個speed。後面就是在相對應的函式裡建立方塊，並上色，速度就要在update裡面進行原本xy+=速度 這樣的程式碼，最後要在if判斷當球的位置是否在磚塊的四個頂點內，就把磚塊的xy值調成視窗外的值。後面就是簡單的背景顏色，鍵盤操作…

這個作業讓我學到很多以前不知道的東西，尤其是用程式碼畫圖這件事，從來沒想過程式碼也能這樣做到，途中雖然遇到很多困難，但是只要持續找到對的方法，就能解決。回頭一看，其實也沒有當初想的這麼困難。

**截圖畫面:**

一張含有 螢幕擷取畫面, 文字, 陳列, 軟體 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 軟體, 作業系統 的圖片

自動產生的描述

**程式碼:**

#ifdef \_\_APPLE\_\_

#include <GLUT/glut.h>

#else

#include <GL/glut.h>

#endif

#include <stdlib.h>

#include <sstream>

int score =0;

int wincnt = 6;

int cnt = 16;

int level =1;

bool winbool = false;

float playerSquareX = 150.0f;

float playerSquareY = 10.0f;

float playerSquareWidth = 100.0f;

float playerSquareHeight = 5.0f;

float playerSquareSpeed = 10.0f;

float autoSquareX = 200.0f;

float autoSquareY = 300.0f;

float autoSquareWidth = 20.0f;

float autoSquareHeight = 20.0f;

float autoSquareSpeedX = 0.3f;

float autoSquareSpeedY = 0.3f;

float bonusX = 100.0f;

float bonusY = 150.0f;

float bonusWidth = 15.0f;

float bonusHeight = 15.0f;

float bonusSpeedX = 0.5f;

float bonusSpeedY = 0.5f;

float bonus2X = 20.0f;

float bonus2Y = 200.0f;

float bonus2Width = 10.0f;

float bonus2Height = 10.0f;

float bonus2SpeedX = 0.5f;

float bonus2SpeedY = 0.5f;

float s1x = 60.0f;

float s1y = 150.0f;

float s1Width = 15.0f;

float s1Height = 100.0f;

float s2x = 150.0f;

float s2y = 250.0f;

float s2Width = 90.0f;

float s2Height = 50.0f;

float s3x = 250.0f;

float s3y = 50.0f;

float s3Width = 200.0f;

float s3Height = 20.0f;

float s4x = 100.0f;

float s4y = 100.0f;

float s4Width = 60.0f;

float s4Height = 90.0f;

//second level square;

float s21x=10;

float s21y=380;

float s21Width=50;

float s21Height=10;

float s22x=70;

float s22y=380;

float s22Width=50;

float s22Height=10;

float s23x=130;

float s23y=380;

float s23Width=50;

float s23Height=10;

//

float s24x=190;

float s24y=380;

float s24Width=50;

float s24Height=10;

float s25x=250;

float s25y=380;

float s25Width=50;

float s25Height=10;

float s26x=310;

float s26y=380;

float s26Width=50;

float s26Height=10;

float ss21x=10;

float ss21y=350;

float ss21Width=70;

float ss21Height=10;

float ss22x=90;

float ss22y=350;

float ss22Width=70;

float ss22Height=10;

float ss23x=170;

float ss23y=350;

float ss23Width=70;

float ss23Height=10;

//

float ss24x=250;

float ss24y=350;

float ss24Width=70;

float ss24Height=10;

int windowWidth = 400;

int windowHeight = 400;

void updateScore() {

// ?自?方?被消除?，增加分?

/\* if (autoSquareX + autoSquareWidth >= s1x && autoSquareX <= s1x + s1Width &&

autoSquareY + autoSquareHeight >= s1y && autoSquareY <= s1y + s1Height) {

score += 10; // 增加分?

wincnt--; // ?少消除的方??量

}\*/

// 在?里可以添加更多的消除方?的判?，以及??的增加分?的操作

glutPostRedisplay(); // 通知 OpenGL 重新?制

}

void renderScene() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

// score present

glColor3f(1.0f, 1.0f, 1.0f); // ?置文字?色?白色

glRasterPos2f(10, 10); // ?置文字位置

std::stringstream ss;

ss << "Score: " << score; // ?分????字符串

std::string scoreStr = ss.str();

for (size\_t i = 0; i < scoreStr.length(); ++i) {

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, scoreStr[i]); // 逐?字符?示分?

}

//

if(!winbool){

// Draw player square (red)

glBegin(GL\_QUADS);

glColor3f(1.0f, 0.0f, 0.0f); // 紅色

glVertex2f(playerSquareX, playerSquareY);

glVertex2f(playerSquareX + playerSquareWidth, playerSquareY);

glVertex2f(playerSquareX + playerSquareWidth, playerSquareY + playerSquareHeight);

glVertex2f(playerSquareX, playerSquareY + playerSquareHeight);

glEnd();

// Draw auto-moving square (yellow)

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 0.0f); // 黃色

glVertex2f(autoSquareX, autoSquareY);

glVertex2f(autoSquareX + autoSquareWidth, autoSquareY);

glVertex2f(autoSquareX + autoSquareWidth, autoSquareY + autoSquareHeight);

glVertex2f(autoSquareX, autoSquareY + autoSquareHeight);

glEnd();

glBegin(GL\_QUADS);

glColor3f(0.0f, 1.0f, 1.0f); //淺藍

glVertex2f(s1x, s1y);

glVertex2f(s1x + s1Width, s1y);

glVertex2f(s1x + s1Width, s1y + s1Height);

glVertex2f(s1x, s1y + s1Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(0.5f, 0.5f, 1.0f);

glVertex2f(s2x, s2y);

glVertex2f(s2x + s2Width, s2y);

glVertex2f(s2x + s2Width, s2y + s2Height);

glVertex2f(s2x, s2y + s2Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(0.5f, 0.5f, 0.25f);

glVertex2f(s3x, s3y);

glVertex2f(s3x + s3Width, s3y);

glVertex2f(s3x + s3Width, s3y + s3Height);

glVertex2f(s3x, s3y + s3Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(0.2f, 0.2f, 0.25f);

glVertex2f(s4x, s4y);

glVertex2f(s4x + s4Width, s4y);

glVertex2f(s4x + s4Width, s4y + s4Height);

glVertex2f(s4x, s4y + s4Height);

glEnd();

// 檢查黃色方塊是否超出視窗底部

if (autoSquareY <= 0) {

glColor3f(1.0f, 0.0f, 0.0f); // 設置文字顏色為紅色

glRasterPos2f(150, 200); // 設置文字位置

const char \*loseMessage = "You Lose"; // 設置要顯示的文字

// 逐個字元繪製文字

for (int i = 0; loseMessage[i] != '\0'; i++) {

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

}

glutSwapBuffers();

return; // 停止遊戲

}

// 檢查是否所有方塊都被消失

if (wincnt == 0) {

glColor3f(0.0f, 1.0f, 0.0f); // 設置文字顏色為綠色

glRasterPos2f(150, 200); // 設置文字位置

const char \*victoryMessage = "Victory"; // 設置要顯示的文字

// 逐個字元繪製文字

for (int i = 0; victoryMessage[i] != '\0'; i++) {

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

}

glutSwapBuffers();

winbool=true;

level+=1;

return; // 停止遊戲

}

}

else{ //seconde level

glClearColor(0.0f, 0.0f, 1.0f, 1.0f);

glBegin(GL\_QUADS);

glColor3f(1.0f, 0.0f, 0.0f); // 紅色

glVertex2f(playerSquareX, playerSquareY);

glVertex2f(playerSquareX + playerSquareWidth, playerSquareY);

glVertex2f(playerSquareX + playerSquareWidth, playerSquareY + playerSquareHeight);

glVertex2f(playerSquareX, playerSquareY + playerSquareHeight);

glEnd();

// Draw auto-moving square (yellow)

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 0.0f); // 黃色

glVertex2f(autoSquareX, autoSquareY);

glVertex2f(autoSquareX + autoSquareWidth, autoSquareY);

glVertex2f(autoSquareX + autoSquareWidth, autoSquareY + autoSquareHeight);

glVertex2f(autoSquareX, autoSquareY + autoSquareHeight);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 1.0f);

glVertex2f(s21x, s21y);

glVertex2f(s21x + s21Width, s21y);

glVertex2f(s21x + s21Width, s21y + s21Height);

glVertex2f(s21x, s21y + s21Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 1.0f);

glVertex2f(s22x, s22y);

glVertex2f(s22x + s22Width, s22y);

glVertex2f(s22x + s22Width, s22y + s22Height);

glVertex2f(s22x, s22y + s22Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 1.0f);

glVertex2f(s23x, s23y);

glVertex2f(s23x + s23Width, s23y);

glVertex2f(s23x + s23Width, s23y + s23Height);

glVertex2f(s23x, s23y + s23Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 1.0f);

glVertex2f(s24x, s24y);

glVertex2f(s24x + s24Width, s24y);

glVertex2f(s24x + s24Width, s24y + s24Height);

glVertex2f(s24x, s24y + s24Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 1.0f);

glVertex2f(s25x, s25y);

glVertex2f(s25x + s25Width, s25y);

glVertex2f(s25x + s25Width, s25y + s25Height);

glVertex2f(s25x, s25y + s25Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 1.0f);

glVertex2f(s26x, s26y);

glVertex2f(s26x + s26Width, s26y);

glVertex2f(s26x + s26Width, s26y + s26Height);

glVertex2f(s26x, s26y + s26Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 1.0f);

glVertex2f(ss21x, ss21y);

glVertex2f(ss21x + ss21Width, ss21y);

glVertex2f(ss21x + ss21Width, ss21y + ss21Height);

glVertex2f(ss21x, ss21y + ss21Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 1.0f);

glVertex2f(ss22x, ss22y);

glVertex2f(ss22x + ss22Width, ss22y);

glVertex2f(ss22x + ss22Width, ss22y + ss22Height);

glVertex2f(ss22x, ss22y + ss22Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 1.0f);

glVertex2f(ss23x, ss23y);

glVertex2f(ss23x + ss23Width, ss23y);

glVertex2f(ss23x + ss23Width, ss23y + ss23Height);

glVertex2f(ss23x, ss23y + ss23Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 1.0f, 1.0f);

glVertex2f(ss24x, ss24y);

glVertex2f(ss24x + ss24Width, ss24y);

glVertex2f(ss24x + ss24Width, ss24y + ss24Height);

glVertex2f(ss24x, ss24y + ss24Height);

glEnd();

glBegin(GL\_QUADS);

glColor3f(0.2f, 0.2f, 0.2f);

glVertex2f(bonusX, bonusY);

glVertex2f(bonusX + bonusWidth, bonusY);

glVertex2f(bonusX + bonusWidth, bonusY + bonusHeight);

glVertex2f(bonusX, bonusY + bonusHeight);

glEnd();

glBegin(GL\_QUADS);

glColor3f(1.0f, 0.5f, 0.5f);

glVertex2f(bonus2X, bonus2Y);

glVertex2f(bonus2X + bonus2Width, bonus2Y);

glVertex2f(bonus2X + bonus2Width, bonus2Y + bonus2Height);

glVertex2f(bonus2X, bonus2Y + bonus2Height);

glEnd();

}

if (autoSquareY <= 0) {

glColor3f(1.0f, 0.0f, 0.0f); // 設置文字顏色為紅色

glRasterPos2f(150, 200); // 設置文字位置

const char \*loseMessage = "You Lose"; // 設置要顯示的文字

// 逐個字元繪製文字

for (int i = 0; loseMessage[i] != '\0'; i++) {

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

}

glutSwapBuffers();

return; // 停止遊戲

}

if (cnt == 0) {

glColor3f(0.0f, 1.0f, 0.0f); // 設置文字顏色為綠色

glRasterPos2f(150, 200); // 設置文字位置

const char \*victoryMessage = "Victory"; // 設置要顯示的文字

// 逐個字元繪製文字

for (int i = 0; victoryMessage[i] != '\0'; i++) {

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

}

glutSwapBuffers();

winbool=true;

return; // 停止遊戲

}

glutSwapBuffers();

}

void updateAutoSquare() {

// Move the auto-moving square

autoSquareX += autoSquareSpeedX;

autoSquareY += autoSquareSpeedY;

bonusX +=bonusSpeedX;

bonusY +=bonusSpeedY;

bonus2X +=bonus2SpeedX;

bonus2Y +=bonus2SpeedY;

// Check collision with window edges

if (autoSquareX + autoSquareWidth >= windowWidth || autoSquareX <= 0) {

autoSquareSpeedX = -autoSquareSpeedX;

}

if (bonusX + bonusWidth >= windowWidth || bonusX <= 0) {

bonusSpeedX = - bonusSpeedX;

}

if (bonus2X + bonus2Width >= windowWidth || bonus2X <= 0) {

bonus2SpeedX = - bonus2SpeedX;

}

if (autoSquareY <= 0) {

glColor3f(1.0f, 0.0f, 0.0f); // Set color to red

glRasterPos2f(150, 200); // Set position to display text

glutSwapBuffers();

return; // Stop the game

}

if (autoSquareY + autoSquareHeight >= windowHeight || autoSquareY <= 0) {

autoSquareSpeedY = -autoSquareSpeedY;

}

if (bonusY + bonusHeight >= bonusHeight || bonusY <= 0) {

bonusSpeedY = -bonusSpeedY;

}

if (bonus2Y + bonus2Height >= bonus2Height || bonus2Y <= 0) {

bonus2SpeedY = -bonus2SpeedY;

}

// Check collision with player square

if (autoSquareX + autoSquareWidth >= playerSquareX && autoSquareX <= playerSquareX + playerSquareWidth &&

autoSquareY + autoSquareHeight >= playerSquareY && autoSquareY <= playerSquareY + playerSquareHeight ) {

// autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

}

if (autoSquareX + autoSquareWidth >= s1x && autoSquareX <= s1x + s1Width &&

autoSquareY + autoSquareHeight >= s1y && autoSquareY <= s1y + s1Height) {

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

s1x = -1000.0f; // Move s1 out of visible area (delete)

s1y = -1000.0f;

wincnt--;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= s2x && autoSquareX <= s2x + s2Width &&

autoSquareY + autoSquareHeight >= s2y && autoSquareY <= s2y + s2Height){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

s2x = -1000.0f;

s2y = -1000.0f;

wincnt--;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= s3x && autoSquareX <= s3x + s3Width &&

autoSquareY + autoSquareHeight >= s3y && autoSquareY <= s3y + s3Height){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

s3x = -1000.0f;

s3y = -1000.0f;

wincnt--;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= s4x && autoSquareX <= s4x + s4Width &&

autoSquareY + autoSquareHeight >= s4y && autoSquareY <= s4y + s4Height){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

s4x = -1000.0f;

s4y = -1000.0f;

wincnt--;

cnt--;

score++;

}

//second

if(autoSquareX + autoSquareWidth >= s21x && autoSquareX <= s21x + s21Width &&

autoSquareY + autoSquareHeight >= s21y && autoSquareY <= s21y + s21Height &&level==2){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

s21x = -1000.0f;

s21y = -1000.0f;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= s22x && autoSquareX <= s22x + s22Width &&

autoSquareY + autoSquareHeight >= s22y && autoSquareY <= s22y + s22Height &&level==2){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

s22x = -1000.0f;

s22y = -1000.0f;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= s23x && autoSquareX <= s23x + s23Width &&

autoSquareY + autoSquareHeight >= s23y && autoSquareY <= s23y + s23Height&&level==2){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

s23x = -1000.0f;

s23y = -1000.0f;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= s24x && autoSquareX <= s24x + s24Width &&

autoSquareY + autoSquareHeight >= s24y && autoSquareY <= s24y + s24Height &&level==2){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

s24x = -1000.0f;

s24y = -1000.0f;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= s25x && autoSquareX <= s25x + s25Width &&

autoSquareY + autoSquareHeight >= s25y && autoSquareY <= s25y + s25Height &&level==2){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

s25x = -1000.0f;

s25y = -1000.0f;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= s26x && autoSquareX <= s26x + s26Width &&

autoSquareY + autoSquareHeight >= s26y && autoSquareY <= s26y + s26Height&& level==2){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

s26x = -1000.0f;

s26y = -1000.0f;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= ss21x && autoSquareX <= ss21x + ss21Width &&

autoSquareY + autoSquareHeight >= ss21y && autoSquareY <= ss21y + ss21Height &&level==2){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

ss21x = -1000.0f;

ss21y = -1000.0f;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= ss22x && autoSquareX <= ss22x + ss22Width &&

autoSquareY + autoSquareHeight >= ss22y && autoSquareY <= ss22y + ss22Height && level==2){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

ss22x = -1000.0f;

ss22y = -1000.0f;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= ss23x && autoSquareX <= ss23x + ss23Width &&

autoSquareY + autoSquareHeight >= ss23y && autoSquareY <= ss23y + ss23Height&&level==2){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

ss23x = -1000.0f;

ss23y = -1000.0f;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= ss24x && autoSquareX <= ss24x + ss24Width &&

autoSquareY + autoSquareHeight >= ss24y && autoSquareY <= ss24y + ss24Height &&level==2){

autoSquareSpeedX = -autoSquareSpeedX;

autoSquareSpeedY = -autoSquareSpeedY;

ss24x = -1000.0f;

ss24y = -1000.0f;

cnt--;

score++;

}

if(autoSquareX + autoSquareWidth >= bonusX && autoSquareX <= bonusX + bonusWidth &&

autoSquareY + autoSquareHeight >= bonusY && autoSquareY <= bonusY + bonusHeight &&level==2){

bonusX = -1000.0f;

bonusY = -1000.0f;

cnt--;

score+=5;

}

if(autoSquareX + autoSquareWidth >= bonus2X && autoSquareX <= bonus2X + bonusWidth &&

autoSquareY + autoSquareHeight >= bonus2Y && autoSquareY <= bonus2Y + bonusHeight &&level==2){

bonus2X = -1000.0f;

bonus2Y = -1000.0f;

cnt--;

score+=5;

}

glutPostRedisplay();

}

void specialKeys(int key, int, int) {

switch (key) {

case GLUT\_KEY\_UP:

playerSquareY += playerSquareSpeed;

break;

case GLUT\_KEY\_DOWN:

playerSquareY -= playerSquareSpeed;

break;

case GLUT\_KEY\_LEFT:

playerSquareX -= playerSquareSpeed;

break;

case GLUT\_KEY\_RIGHT:

playerSquareX += playerSquareSpeed;

break;

case 27: // ESC 鍵

exit(0);

break;

}

glutPostRedisplay();

}

void init() {

glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // 黑色背景

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, windowWidth, 0, windowHeight, -1, 1);

glMatrixMode(GL\_MODELVIEW);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(windowWidth, windowHeight);

glutCreateWindow("Square");

glutDisplayFunc(renderScene);

glutSpecialFunc(specialKeys);

glutIdleFunc(updateAutoSquare);

init();

glutMainLoop();

return 0;

}