成圖技術與應用期末報告

題目 : GG 飛車

4104056003 陳鐿文

4104056028 許丰譯

Member Contribution

陳鐿文　　100

許丰譯　　80

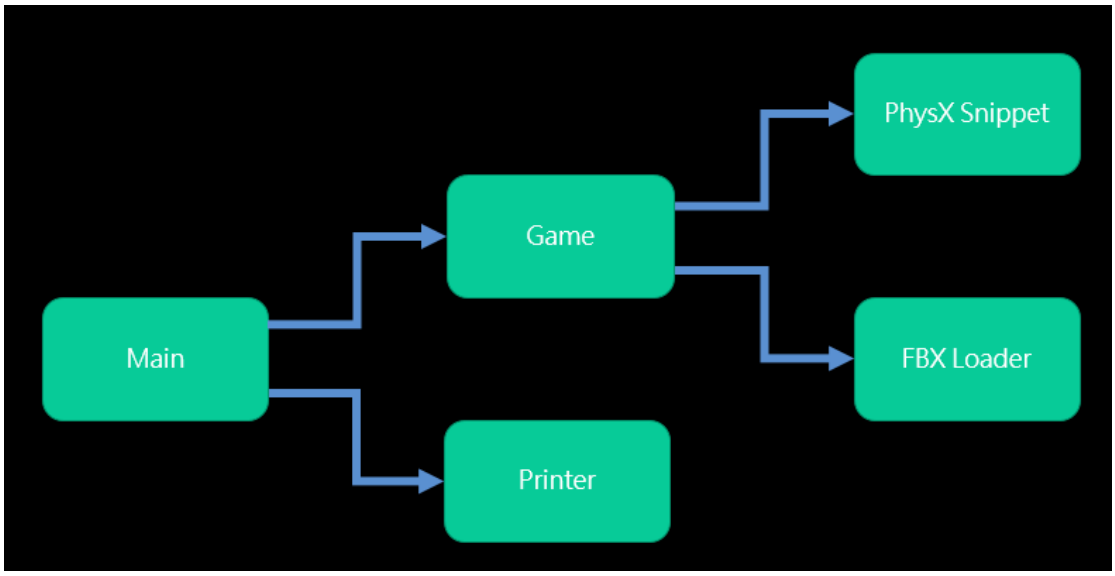
Goal

最近的一款手機遊戲 <極速領域> ，勾起了我們小時候玩過的 <跑跑卡丁車> 的記憶，而它的遊戲性真的做得很不錯，讓我們想要挑戰一下，看看我們能夠做到何種程度的內容。

Introduction

GG 飛車使用本課堂的主角 OpenGL 來繪製視窗內容，物理的部分因為想讓我們的作品有更高的可玩性，所以使用了現在 Nvidia 公司負責開發及維護的 PhysX 函式庫作為遊戲的物理引擎。遊戲內容目前只有做到一個賽道及一輛賽車，也只能進行單人遊戲，遊戲也還有許多內容還沒完成，像是飄移後小噴以及氮氣加速之類的，但已經足夠進行簡單的計時賽，操控部分可用 WASD 鍵控制車子移動，利用 L 鍵控制手煞車鎖死後輪來進行飄移，如果翻車或被地形卡住可以用 R 鍵來將賽車恢復到前一個儲存的位置，J 跟 K 鍵提供作弊功能，讓老師可以瞬移到前方或後方。

架構



程式碼

Printer

將傳入的 x, y 座標依照對齊方式先轉換成左下對齊形式，方便後面處理。

```
//Transform to Left-Bottom Form
switch (horizontal)
{
case Printer::eLEFT:
    break;
case Printer::eRIGHT:
    x -= length * letter_width + (length - 1) * margin;
    break;
case Printer::eCENTER:
    x -= length / 2 * letter_width + length / 2 * margin + (length % 2 == 0 ? -((int)margin / 2) : letter_width / 2);
    break;
default:
    return nullptr;
}
switch (vertical)
{
case Printer::eTOP:
    y -= letter_height;
    break;
case Printer::eBOTTOM:
    break;
case Printer::eCENTER:
    y -= letter_height / 2;
    break;
default:
    return nullptr;
}
```

由於統一了對齊方式，一個迴圈便能產生所有點座標。

```
for (int i = 0; i < length; i++)
{
    //v1
    vertices[i * 8 + 0] = (GLfloat)x;
    vertices[i * 8 + 1] = (GLfloat)y;

    //v2
    vertices[i * 8 + 2] = (GLfloat)x;
    vertices[i * 8 + 3] = (GLfloat)(y + letter_height);

    //v3
    vertices[i * 8 + 4] = (GLfloat)(x + letter_width);
    vertices[i * 8 + 5] = (GLfloat)(y + letter_height);

    //v4
    vertices[i * 8 + 6] = (GLfloat)(x + letter_width);
    vertices[i * 8 + 7] = (GLfloat)y;

    //Next
    x += margin + letter_width;
}

return vertices;
```

文字紋理座標先由 ASCII 編碼檢查是否在可印範圍，若不可印，將四點座標設

為 0，印出空白；若在範圍內，定位 x, y 座標到目標文字左下角。

```
if (content[i] < 33 || content[i] > 126)
{
    for (int j = i * 8; j < i * 8 + 8; j++)
        texture_vertices[j] = 0.0f;
    continue;
}
x = ((content[i] - 33) % 47) * (ALPHABET_LETTER_WIDTH + ALPHABET_MARGIN_WIDTH);
y = (content[i] - 33) > 46 ? 0 : (ALPHABET_LETTER_HEIGHT + ALPHABET_MARGIN_HEIGHT);
```

定位好左下座標後，根據定義的文字寬高，取得四點座標儲存。

```
//v1
texture_vertices[i * 8 + 0] = (GLfloat)x / (GLfloat)ALPHABET_BITMAP_WIDTH;
texture_vertices[i * 8 + 1] = (GLfloat)y / (GLfloat)ALPHABET_BITMAP_HEIGHT;

//v2
texture_vertices[i * 8 + 2] = (GLfloat)x / (GLfloat)ALPHABET_BITMAP_WIDTH;
texture_vertices[i * 8 + 3] = (GLfloat)(y + ALPHABET_LETTER_HEIGHT) / (GLfloat)ALPHABET_BITMAP_HEIGHT;

//v3
texture_vertices[i * 8 + 4] = (GLfloat)(x + ALPHABET_LETTER_WIDTH) / (GLfloat)ALPHABET_BITMAP_WIDTH;
texture_vertices[i * 8 + 5] = (GLfloat)(y + ALPHABET_LETTER_HEIGHT) / (GLfloat)ALPHABET_BITMAP_HEIGHT;

//v4
texture_vertices[i * 8 + 6] = (GLfloat)(x + ALPHABET_LETTER_WIDTH) / (GLfloat)ALPHABET_BITMAP_WIDTH;
texture_vertices[i * 8 + 7] = (GLfloat)y / (GLfloat)ALPHABET_BITMAP_HEIGHT;
```

準備繪製文字內容，儲存先前矩陣，並更改投影矩陣成正交投影，再儲存，並切換成 Modelview。

```
glPushMatrix();
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (GLfloat)window_width, 0.0, (GLfloat)window_height);
    glPushMatrix();
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
```

綁定 Texture 成傳入字體顏色對象

```
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, mAlphabetTexture[colorID]);
```

生成 VBO 並綁定點座標、紋理座標、索引陣列，繪製。

```
//Generate Buffers
glGenBuffers(1, &vertex_VBO);
glGenBuffers(1, &texture_VBO);
glGenBuffers(1, &indice_VBO);

//Set Vertex Buffer
glBindBuffer(GL_ARRAY_BUFFER, vertex_VBO);
glBufferData(GL_ARRAY_BUFFER, 2 * 4 * length * sizeof(GLfloat), vertices, GL_STATIC_DRAW);
glVertexPointer(2, GL_FLOAT, 0, 0);
glEnableClientState(GL_VERTEX_ARRAY);

//Set Texture Buffer
glBindBuffer(GL_ARRAY_BUFFER, texture_VBO);
glBufferData(GL_ARRAY_BUFFER, 2 * 4 * length * sizeof(GLfloat), texture_vertices, GL_STATIC_DRAW);
glTexCoordPointer(2, GL_FLOAT, 0, 0);
glEnableClientState(GL_TEXTURE_COORD_ARRAY);

//Set Indice Buffer
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, indice_VBO);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, 4 * length * sizeof(GLuint), indices, GL_STATIC_DRAW);

//Draw
for(int i = 0; i < length; i++)
    glDrawElements(GL_QUADS, 4, GL_UNSIGNED_INT, (void*)(4 * i * sizeof(GLuint)));
```

最後，解除綁定 VBO，刪除 VBO、文字座標，解除綁定紋理，並恢復投影及 Modelview 矩陣，完成文字繪製。

```
        //Unbind Buffers
        glBindBuffer(GL_ARRAY_BUFFER, 0);
        glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);

        //Delete Buffers
        glDeleteBuffers(1, &vertex_VBO);
        glDeleteBuffers(1, &texture_VBO);
        glDeleteBuffers(1, &indice_VBO);

        //Delete Data
        delete vertices;
        delete texture_vertices;
        delete indices;

        glBindTexture(GL_TEXTURE_2D, 0);
        glMatrixMode(GL_PROJECTION);
    glPopMatrix();
    glMatrixMode(GL_MODELVIEW);
glPopMatrix();
```

Game rendering

首先計算相機位置：由賽車位置往車頭反方向向後 8 單位，再向上 5 單位，最後根據賽車側滑速度向賽車左右方向偏移。

```
//Compute Camera Position
mCameraPosition = mRaceCarPosition - mRaceCarDirection * glm::vec4(8.0f) +
glm::vec4(0.0f, 5.0f, 0.0f, 0.0f) + mRaceCarRight * glm::vec4(mRaceCar->computeSidewaysSpeed() / 3.0);
```

繪製賽道，由於物理形狀及模型形狀有偏差，將賽車往下位移，使模型符合視覺位置，取得賽車位置及方向乘上矩陣後繪製。

```
//Race Track
glPushMatrix();
    mRaceTrackFBX->draw();
glPopMatrix();

//Race Car
glPushMatrix();
    lVecBuffer = mRaceCarUp * glm::vec4(-1.3f);
    glTranslatef(lVecBuffer.x, lVecBuffer.y, lVecBuffer.z);
    glMultMatrixf(glm::value_ptr(PxMatToGlmMat(PxMat44(mRaceCar->getRigidDynamicActor()->getGlobalPose()))));
    glScalef(-1.0f, 1.0f, 1.0f);
    mRaceCarFBX->draw();
glPopMatrix();
```

輪胎繪製前，同樣進行位移、縮放，使符合視覺位置。

```
//Align Front Tire
if (i == 0 || i == 1){ ... }

//Align Back Tire
if (i == 2 || i == 3){ ... }

//Align Right Tire
if (i == 0 || i == 2){ ... }

//Align Left Tire
if (i == 1 || i == 3){ ... }
glMultMatrixf(glm::value_ptr(PxMatToGlmMat(lTempSavePose)));

//Scale Back Tire To Mach
if (i == 2 || i == 3)
    glScalef(1.143f, 1.143f, 1.143f);
//Flip Tire To Right Side
if(i == 1 || i == 3)
    glScalef(-1.0f, 1.0f, -1.0f);
mRaceCarTireFBX->draw();
```

Physics simulate

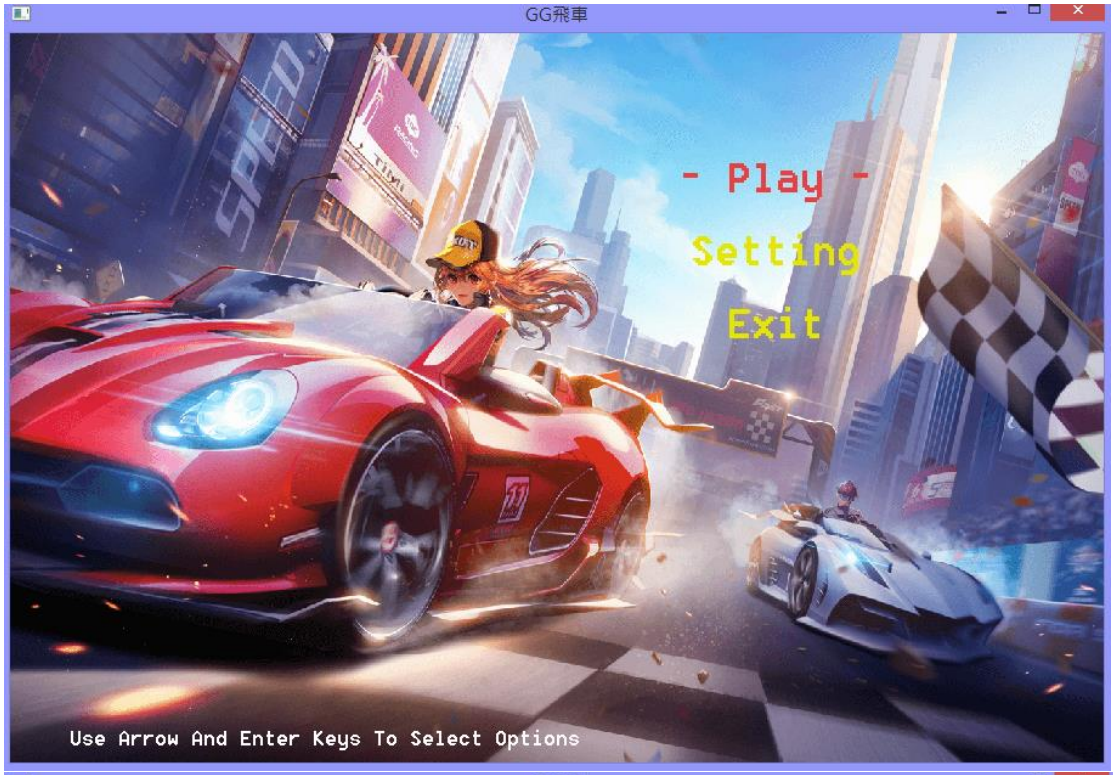
先重置賽車的所有操控，根據按鍵陣列輸入控制項目給賽車物件。

```
//Reset Control
releaseAllControls();

//Process Keyboard Input
{
    //Direction
    if (mKeyPress['W'] || mKeyPress['w'])
    {
        mRaceCar->mDriveDynData.forceGearChange(PxVehicleGearsData::eFIRST);
        mVehicleInputData.setDigitalAccel(true);
    }
    if (mKeyPress['A'] || mKeyPress['a']) { ... }
    if (mKeyPress['S'] || mKeyPress['s']) { ... }
    if (mKeyPress['D'] || mKeyPress['d']) { ... }

    //Handbrake
    if (mKeyPress['L'] || mKeyPress['l']) { ... }

    //Reset Race Car To Saved Point
    if (mKeyPress['R'] || mKeyPress['r'] || mRaceCarPosition.y < 0.0) { ... }

    //Cheating
    if (mKeyPress['J'] || mKeyPress['j']) { ... }
    if (mKeyPress['K'] || mKeyPress['k']) { ... }
```

進行完物理模擬後，確認賽車位置是否在檢查點，並設置 Flag。(游標處有修改)

```
//Update Check Point
{
    //Goal Check
    if (isCarInArea(mGoalLineAreaVerticeMin, mGoalLineAreaVerticeMax, glm::vec3(mRaceCarPosition)))
    {
        if (mCheckPoint1Flag && mCheckPoint2Flag)
        {
            mCurrentRound++;
            mCheckPoint1Flag = false;
            mCheckPoint2Flag = false;
        }
    }

    //Check Point 1
    if (isCarInArea(mCheckPoint1VerticeMin, mCheckPoint1VerticeMax, glm::vec3(mRaceCarPosition)))
    {
        mCheckPoint1Flag = true;
        mCheckPoint2Flag = false;
    }
    //Check Point 2
    if (mCheckPoint1Flag && isCarInArea(mCheckPoint2VerticeMin, mCheckPoint2VerticeMax, glm::vec3(mRaceCarPosition)))
    {
        mCheckPoint2Flag = true;
    }
}
```

Result


GG飛車
- Play -
Setting
Exit
Use Arrow And Enter Keys To Select Options


GG飛車
Press WASD To Control
Press L For Handbrake
Press R To Go Back To Saved Point
Press JK To Cheat
Round : 0 / 3
FPS : 66.67
17 km/h

Feedback

許丰譯：

像這次這麼大的工程我是第一次做，當初想法很美好，可是事實上一點點小小的細節就會燒掉很多時間，另外，要兩個人一起寫也是一大困難，還要感謝隊友包容我雜亂沒註解的程式碼，低落的生產率，以及包攬了大量的工作。

陳鐿文：

這次使用了很多函式庫，從中學習到了很多，Visual Studio 的環境還有許多不熟的地方，人家寫好的範例都花很久的時間才懂如何順利執行，明明包裝得很完善，真的除非碰到不然不會自己發現，PhysX 的範例 Solution 裡包含有許多 Project，一開始都執行失敗，結果是沒有選好要執行的專案，選在 ALL_BUILD 專案，裡面只有 CMAKE List。網路上幾乎找不到 PhysX 的教學，找到的都是舊版本的，所以只能看官方範例程式碼，跟讀官方文檔。看了許多程式碼後，也漸漸知道大型程式怎麼分散架構，cpp 檔與 header 檔如何互相配合，把各個功能封裝起來。總而言之，花了一些時間繞遠路，但我相信沒有白費，很多細節趁現在注意到，或許以後就能避免更多類似情況，雖然爆肝了一個多月，體重也掉了幾公斤就是了。