成圖技術與應用 Final Project

組員：資工四 4104056004 李家駿 資工四 4104056034 柯冠名

組員貢獻：

李家駿：遊戲機制，物理效果，戰鬥場景，關卡設計，作弊模式
柯冠名：開頭畫面，過場畫面，勝利與死亡動畫，遊戲背景，人物模組
貢獻度：李家駿 = 50%
柯冠名 = 50%

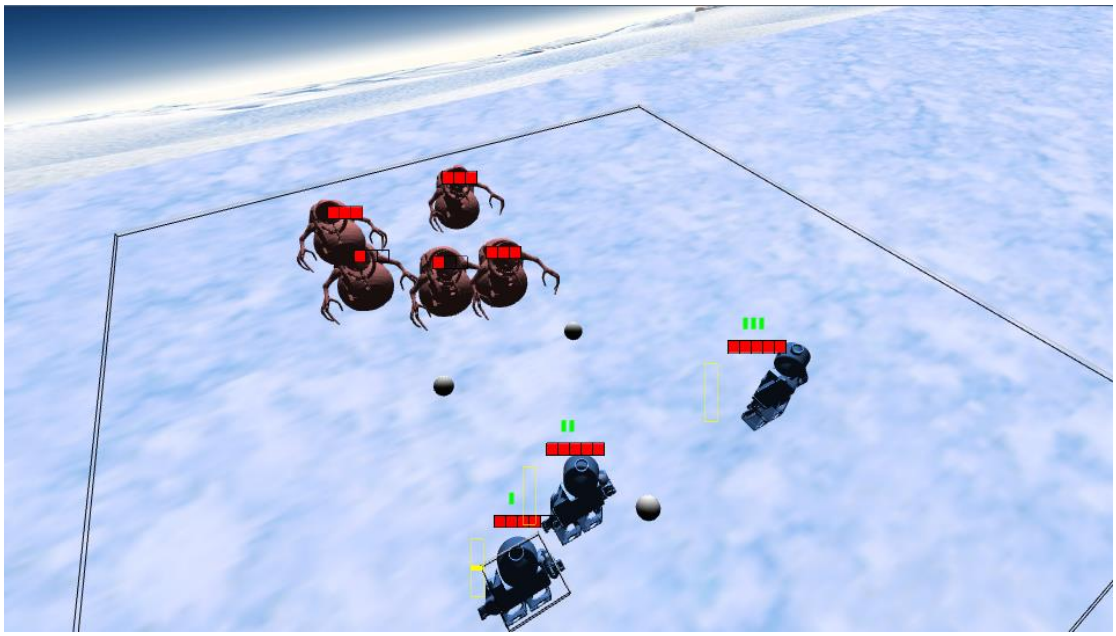
主題：

在一個天寒地凍，充滿著皚皚白雪的冰天雪地，三位英雄靠著地上的雪做出僅有的武器，扔向對面一個又一個邪惡的化身，這是一個充滿激情與熱血的闖關遊戲。

作品介紹：

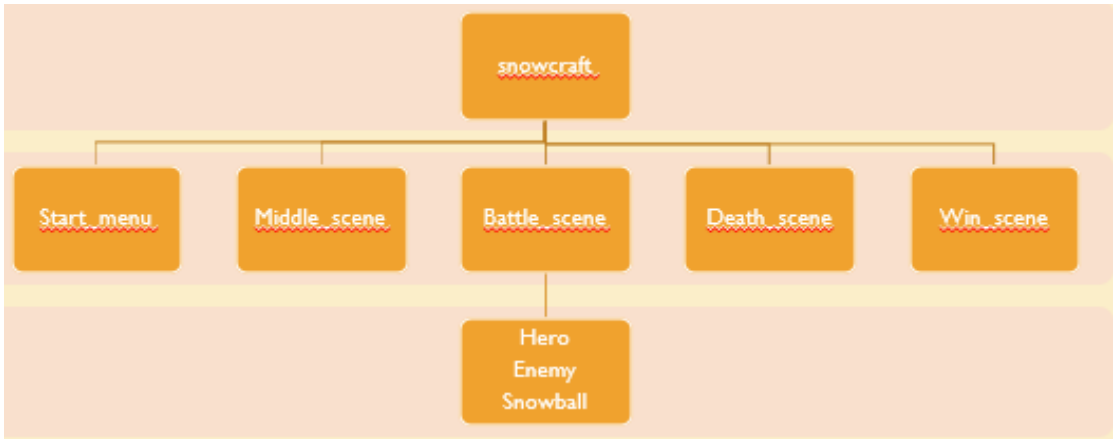
名稱：Snowcraft(打雪仗)
簡介：以打雪仗為主軸的闖關遊戲，共有五道關卡，己方三人，對手則隨著關卡持續增加。
過關條件：在己方 3 人全數陣亡之前，擊倒所有敵人
遊戲規則與操作方法：

1. 使用滑鼠左鍵進行集氣與射擊，集氣時間越長射程越遠
2. 數字 1、2、3 可以進行切換人物
3. 如果被對方的雪球擊中，會有 3 秒無法發射雪球
4. 密技：詳見開始畫面中的說明(Konami code)



程式碼架構與描述：

1. 概要



在這個遊戲中，總共有五個場景，分別是:

開始畫面(Start_menu)

過場畫面(Middle_scene)

戰鬥畫面(Battel_scene)

死亡畫面(Death_scene)

勝利畫面(Win_scene)

這五個場景都包含在 Class:Snowcraft 物件 snowcraft 中，其中 Battel_scene 之下還包含了三個物件。

2.Class:Snowcraft

Class:Snowcraft 是一個包含五個 Class:Scene 物件的物件。Constructor 會宣告並初始化五個場景，並預先載入 OBJ model。之後會負責接收玩家的指令輸入，並操作人物或轉換場景。

3.Class:Scene

Class:Scene 負責建立遊戲中的場景，並繪製出來。snowcraft 物件之下的五個 Scene 物件都是繼承自此 class，再個別建立。這五個場景中，又可細分成兩類:

(1) Start_menu,Middle_scene,Death_scene 與 Win_scene:

這四個場景的繪製模式類似，都是在視窗中繪製一個填滿畫面的矩形，再貼上 Texture。而這些場景中的動畫效果，則是透過載入多張 Texture，並依次變換，做出類似 GIF 的動畫效果。

(2) Battle_scene:

這個場景與上四個場景不同，為 3D 場景。此場景的鏡頭被設置在 Y 軸逆時針旋轉 120 度、Z 軸逆時針旋轉 45 度的位置，以還原原版遊戲的畫面感。這個場景中總共會出現三種物件:Class:Hero、Class:Enemy 與 Class:Snowball，下面會詳細介紹這三種物件的特性。

4.Class:Moving_object

Class:Moving_object 是一個可移動的物件，物件擁有自己的座標與速度。以下三個物件皆是繼承自此物件:

(1) Class:Hero:

為我方的可操作人物，透過擷取滑鼠在視窗中的位置，在經由函式將滑鼠座標轉換成空間座標，來順暢移動人物位置。

```
Vec GetCursorPlane(int x, int y)
{
    GLfloat mouseX, mouseY;

    GLdouble posX1, posY1, posZ1;
    GLdouble posX2, posY2, posZ2;

    GLdouble modelview[16],projection[16];
    GLint viewport[4];
    Vec to;
    /*Get matrix*/
    //glMatrixMode(GL_VIEWPORT);
    glGetIntegerv(GL_VIEWPORT,viewport);
    glGetDoublev(GL_MODELVIEW_MATRIX, modelview);
    glGetDoublev(GL_PROJECTION_MATRIX, projection);
```

(圖為將滑鼠位置轉換成空間座標的 funciton，宣告於 include/snowcraft/scene/scene.h)

Class:Hero 物件能夠透過滑鼠點擊發射雪球，滑鼠按壓時間越長，雪球的飛行距離越遠。

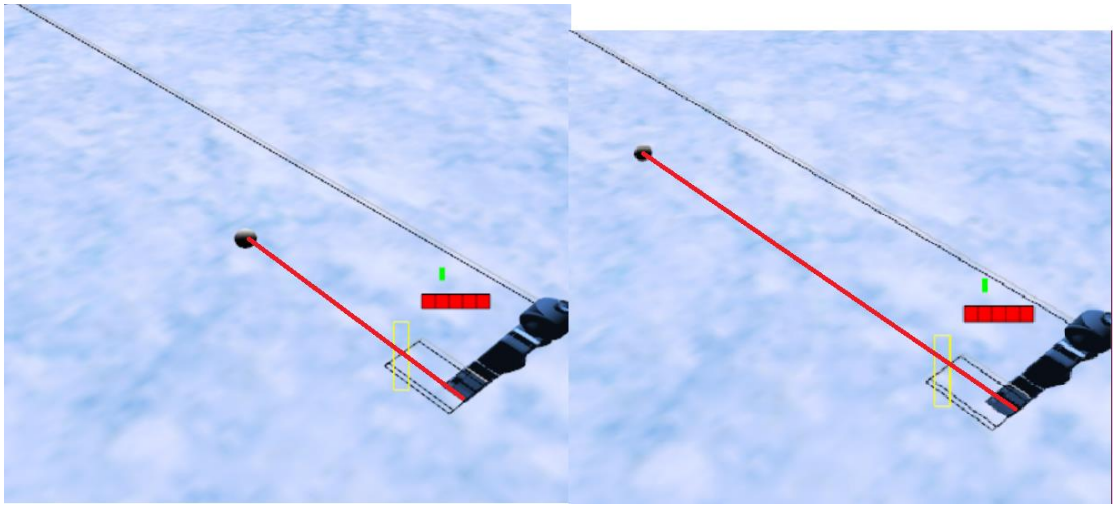
```
Snowball chargeShot(char key) {
    if (!hitback) {
        Vec v(0.4, -0.07, 0);
        switch (key) {
        case 'c':   //Charging
            if (charge < 1)
                charge += 0.03;
            else
                charge = 1;
            break;
        case 's':   //Shot
            cout << "charge:" << charge << "\n";
            v.plus(Vec(0.7*charge, 0, 0));
            charge = 0;
            return Shot(v);
            break;
        default:
            break;
        }
    }
}
```

雪球初速

集氣每次增加 3%，最多至 100%

雪球速度=初速+0.7*集氣%數

(雪球集氣射擊 function，宣告於 include/snowcraft/objects/hero.cpp)



(集氣射擊距離比較)

(2) Class:Enemy

此為遊戲中敵人的物件，會依關卡文件設定決定出現位置與移動模式。移動模式分為三種:

```
#define MOVE_MODE_LINE 0
#define MOVE_MODE_TRIANGLE 1
#define MOVE_MODE_SQUARE 2
```

直線
三角形
矩形

敵人會依照自己的移動模式，建立一個頂點陣列，並依序走過這些頂點。走到頂點時，敵人會發射一顆雪球，並朝著下一個頂點前進。

```
/*Initialize enemy path*/
switch (move_mode) {
    case MOVE_MODE_LINE:
        path = vector<Vec>{ Vec(_pos
        //cout << "0path:" << path[0].x() <
        break;
    case MOVE_MODE_TRIANGLE:
        path = vector<Vec>{ Vec(_pos
        //cout << "1path:" << path[0].x() <
        break;
    case MOVE_MODE_SQUARE:
        path = vector<Vec>{ Vec(_pos
        //cout << "2path:" << path[0].x() <

        break;
}
```

(圖為敵人路徑頂點設定，此段程式碼位於 include/snowcraft/objects/enemy.cpp 的 funtion:init()中)

(3) Class:Snowball

此為雪球的物件。雪球在初始化時需要給予位置，讓雪球可以從發射者的位置射出。雪球與人物不同的地方在於，雪球的初速會包含一個向下(y 軸負方向)的速度，這個方向的速度讓雪球進行一個斜下方向的等速運動。雪球的 y 座標=0 時，系統將會判定其落在地面，將這個雪球清除。

4.其他遊戲機制

(1)雪球擊中偵測:

因為此遊戲的物件移動都被限制於一個平面，且人物之間並沒有碰撞偵測，雪球擊中偵測的部分就不使用 bounding box 的方法實作，而是直接判斷雪球的 xz 座標與人物的座標是否相同，如相同就判斷是否為敵方所射出，條件符合則判斷雪球擊中。

(2) 雪球擊退效果:

為了加強擊中的畫面感，我們在遊戲中實作了擊退的系統。首先，人物在被雪球擊中後，會接收這個雪球物件的速度，並以這個速度向後移動。

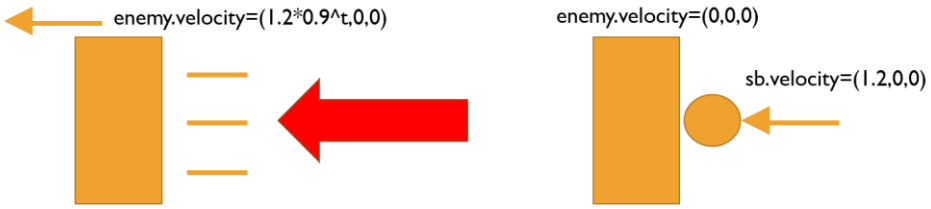
```
bool HitByBall(Snowball sb) {    //Return true if still alive.
    if (!immortal) {
        HP--;
        if (HP == 0)
        {
            alive = false;
            immortal = true;
            //Display death animation.
            return false;
        }
        else
        {
            //Stun for 1 sec.
            timer = time(0);     //Get now time
            _velocity = Vec(sb.velocity().x(), 0, 0);
            hitback = true;
            immortal = true;
            stoping = true;
            //Display stun animation.
            return true;
        }
    }
    return false;
}
```

人物被擊中後將會開始計時，不可動彈。

(圖為人物被雪球擊中後觸發的 function，Enemy 與 Hero 物件中均有此 function)

```
bool HitbackING() {
    if (hitback) {
        if (time(0) - timer == 3) { //一秒之後解除擊退狀態
            hitback = false;
            immortal = false;
            stoping = false;
            _velocity = Vec(0.05, 0, 0);
            return false;
        }
        else
        {
            Moveto(_position.plus(_velocity));
            _velocity = Vec(_velocity.x()*0.9, 0, 0);
            return true;
        }
    }
    return false;
}
```

(圖為被擊退時觸發的 funciton，在 3 秒內，人物將會以雪球的速度向後移動，這個速度每次都會減少 10%，直到 3 秒結束，人物始可移動。)



(擊退效果示意圖)

(3) 作弊模式

作弊模式會讓人物發射大量的雪球，且不會被雪球砸中。作弊模式的開啟方式為:在開始畫面中，依次輸入指令(↑ ↑ ↓ ↓ ⟷ ⟷ba)。此指令名為"Konami 密技"，是出現在日本遊戲公司 Konami 旗下遊戲《宇宙巡航艦》與《魂斗羅》中的作弊碼。

結論：

製作此遊戲的目的為致敬小時候玩過的小遊戲 Snowcraft 雪球大戰，在原版遊戲中，人物是透過按住滑鼠左鍵進行拖移的，但我們希望能將操作變得簡單順暢，故在我們的遊戲中，將操作模式改成了現在的樣子，其他地方也做了一些變動，讓遊戲更加順暢。希望日後能再為這遊戲加上音效與人物動畫，甚至加上 BOSS 關卡，讓整個遊戲更加完整有趣。

Feedback：

柯冠名：

老實說當初沒有想選這門課，因為上過老師的動態網頁設計，也體會到老師

的作業沒有那麼好混，相對於其他老師的作業麻煩不少，但是還是被同學說服選了這門課呵呵。

不過也剛好是大四下學期了，說實話可以忙的事也忙完了，學分也修的差不多了，成圖的作業剛好能夠填滿我的空閒時間，讓我沒那麼樣的無聊，也不會滿腦子只想著要玩樂或是待會要吃什麼的無聊問題。

做作業的過程中，每當完成一項功能，即使可能只是一個不怎麼重要的部分，心裡也會覺得一點點的開心，感覺自己又進步了一些，學到了新的東西。最後的這個專題算是自己第一個參與實作的遊戲，雖然只算是一個輕鬆的小遊戲，但也是感謝我的組員的大力幫忙，沒有他我大概只能生出一個更糟糕的作品出來吧。

只有老師的課是這樣，整個學期邊抱頭煩惱邊 coding，永遠都在跟時間賽跑，最後作完期末專題回過頭來看，會是滿滿的成就感，不單純只是為了學分應付了事。

李家駿:

雖然以前有過使用 Unity 寫遊戲的經驗，但對於遊戲引擎的基礎完全不了解。這學期修了這門課，從 opengl 的基礎開始學起，對於遊戲設計的基礎有了更深刻的了解，而且在期末專題中，運用到了很多以前學到的知識，讓我感受到大學這四年的經驗累積。在大學時期的最後，能夠寫出自己小時候喜歡玩的遊戲，讓我感覺到非常的開心，感謝老師這學期的用心教導，希望以後也能運用這些知識，充實自己的人生。