Multimedia Final Project - 2048 3D

Team 1
Member：103062104 邱思燈
103062202 廖重勳

● Abstraction

我們所選擇的主題是 OpenGL/Direct3D based game/application，建立 3D 模型物件搭配 model transformation、viewing transformation、Lighting，目的是讓玩家與物件之間互動，並且利用各種視覺與聽覺上的享受，來轟炸使用者的感官。

● Introduction

運用 Unity 遊戲開發平台來實作 2048 小遊戲，遊戲內包含兩種模式：經典模式與限時模式，並各有三種規模 3*3、4*4、3*3*3；另外遊戲元素還包含：遊戲選單、記分板、暫停、返回上一步等等，與一般 2D 2048 的差別在於 3*3*3 模式的玩法、鏡頭轉換以及限時模式的光源變化。

● Method

在 Unity 裡面，有分不同的場景(Scenes)，在我們的遊戲當中，3*3 遊戲為一個場景、4*4 為另一個場景、3*3*3 也是不同的場景，以此類推。因此我們的做法是事先建好場景，與該場景中所需要用到的所有 cube、music、lighting、canvas 等物件，在需要時才將其顯示出來，例如：在 3*3 中，我們會先建好 9 個 Cube Objects、在 3*3*3 中，則會先建好 27 個 Cube Objects 等等。

接著，以遊戲選單的形式，將這些不同的場景結合在一起，玩家可以根據想遊玩的遊戲模式來選擇，這些遊戲模式則會對應到不同的場景，

● Implementation

1. 腳本(Script)

首先，為了控制 Objects 的動作，必須創建腳本，使用的語言是 C#。每個腳本會有基本的兩個 function：Start()、Update()：Start()會在腳本開始時執行一次，以利參數初始化；而 Update()則為一個 loop，會在每一個 frame 不斷重複執行以利更新控制參數。

接著重要的是，必須在腳本裡取得每個所需的 Objects；當初在建場景時，已經給每一個 Objects 一個名字、或是一個標籤(tag)，因此可以透過名字或標籤來得到相對應的 Objects，如圖一。

```
string str = "Cube_" + i.ToString ();
Cube [i] = GameObject.Find (str);
cm [i] = Cube [i].GetComponent<CubeMove> ();
```

※圖一，以取得 Cube Object 為例

再來，即可透過取得的 Objects 來實作遊戲的操作，也就是 Cube 的上下左右移動、分數相同合併、Undo、Game Over、Reset 等等。

2. Cube 管理

Cube 管理之實作：先前提過我們會先建好所有的 Cube Objects，如圖二，對於每一個 Cube，我們給定一個編號，與固定的 xyz 座標位置，藉著再給予一個 Score 的參數。

實際上 Cube 的位置都不會變動，即使遊戲過程中看到 Cube 的移動，在移動完成之後，會馬上根據 Cube 編號，把它設回原始位置，所以實際上 1 號 Cube 還是在 1 號位置、2 號 Cube 也還是在 2 號、以此類推，目的在於方便管理這些 Cube。

做法是透過 Score 參數，所以我們可以知道每個 Cube 自身的分數，我們讓分數不等於零的 Cube 顯示出來，分數為 0 的則把它變成隱形，如此一來製造出 Cube 位置有改變的錯覺。

※圖二，3*3 遊戲模式示意圖

3. Model Transformation

接著介紹 Cube 移動與分數之合併，以 Cube 往下移動為例(上、下、左、右幾乎相同)，如圖三，當按下鍵盤的 A 或方向鍵下之後，我們必須將場景中所有的 Cube 往下移動，若遇到邊界、或是有其他 Cube 擋住，則停下。當初在建場景時，我們會知道整個場景 X、Y、Z 軸的方向，然後對應到 Cube 的上下左右方向，因此往下移動就是往-X 軸移動。此外，按下按鍵會播放音效也是在這邊判斷。

```
if (Input.GetKeyDown (KeyCode.A) || Input.GetKeyDown (KeyCode.LeftArrow)) {
    dir = "-X";
    All_Cube_Move ();
    temp = times;
    MOVING = true;
    if (isMoved)
        press_music.Play ();
```

※圖三，Cube 往下移動之操作

移動 Cube 可以分為兩步驟：

(1) 將空格補滿：

當所有 Cube 往某方向移動時，直到遇到邊界或其他 Cube 前，必須一直前進下去，所以移動完成時，Cube 之間是不應該有空間的。

(2) 將相同分數合併：

當步驟一完成後，根據移動方向，來判斷某個軸上，有無分數相同的 Cube，若有則合併。

在移動時，我們會去更新所有 Cube 的 Score、遊戲總分數、記錄哪些 Cube 有移動、移動距離等等，有了 Cube 的移動位置、距離，我們就可以做出 Cube 移動的動畫效果，但如同前面說的，雖然表面上看到 Cube 的移動，但實際上移動完成之後，會根據編號把 Cube 設回原始位置。以下為 Cube 移動完成之後的處理動作，如圖四。

```
MOVING = false;
ResetAndShow ();
RandomCreateCube ();
```

※圖四，Cube 移動完成之後的處理動作。

ResetAndShow()之實作就是將所有 Cube 設回原始位置，以及讓分數不等於 0 的 Cube 顯示出來，並且根據 Cube 的分數，貼上相對應的 texture(2、4、8、16 等，皆是我們自己畫的)。

RandomCreateCube()則是利用 Random 找到分數為 0 的 Cube 並給此 Cube 2 分或 4 分，這裡我們是按照 2048 的規則，每移動一次，都會隨機產生一個 Cube，產生的同時，若場景中 Cube 的數量已經達到飽和，則會去判斷有沒有 Game Over(判斷任何方向皆無法移動)。

以上為 3*3、4*4 遊戲的大致過程。

4. Viewing Transformation

另外，在 3*3*3 的模式中，移動的方法、原理與上述完全相同，比較特別的地方在於我們新增了 Camera 之轉動操作。玩法是：Camera 總共可以面對四個地方，分別是 3*3*3 的四個面(不包含上下兩面)，Camera 面對到的那一面，則可以操作那一面 3*3 的 Cube，其他面的 Cube 則不能去移動，也就是我們實際只會操作到 27-3 = 24 個 Cube，因為我們都是去操作 3*3*3 表面的 Cube，中間會有 3 個 Cube 不會去操作到。

3*3*3 模式中，按下 Z、X 鍵是轉動 Camera，在人的主觀視覺看起來就會像是方塊往左、往右轉。由於 Camera 面對的方向不同，看到的 X、Y、Z 軸的方向也會跟著改變，因此必須記錄 Camera 目前的位置，如圖五。

```
} else if (Input.GetKeyDown (KeyCode.Z)) {
    curCamera--;
    if (curCamera == 0)
        curCamera = 4;
    cam.move ("Z", curCamera);
    camSwap (curCamera);

} else if (Input.GetKeyDown (KeyCode.X)) {
    curCamera++;
    if (curCamera == 5)
        curCamera = 1;
    cam.move ("X", curCamera);
    camSwap (curCamera);
}
```

※圖五，Camera 之操作

前面提到，Camera 面對到的那個面，才能去操作那個面的 Cube，但是不同的面對方向，會看到不同方向的 X、Y、Z 軸，也就是每個面按上下左右的方向都會不一樣，當初在 3*3、4*4，方向都是固定的，例如 ： 按左就是往-Z 軸移動。但是在 3*3*3 要特別去處理這種問題，處理方法就是寫四種上下左右，每個鏡頭會對應到一個上下左右，所以要去記錄目前 Camera 位置。

以上為 3*3*3 的遊戲方式以及實作方法。

5. Lighting

最後，在限時模式中我們新增了光源的控制，透過場景周遭的光之變化，如黑夜、清晨、白晝、黃昏，以實現時間改變，讓玩家即使在沒有時間 UI 可以參考的情況下，仍可以透過這些光的變化順利地遊玩。

實作的方法是設置了一個無窮遠光(Directional Light)，由於無窮遠光無論如何改變座標位置，都不會影響其最後在場景中展現的效果，而為了實現黑夜與白晝的變化，我們使用 Rotate 的技術實作：給定旋轉的角度，透過 Update()，在每個 frame 更新無窮遠光的旋轉角；最後當無窮遠光旋轉完一圈之後，遊戲立即結束。

6. 其他遊戲基本元素

Undo： 可以回到移動 Cube 前的狀態，只能回到前一次的狀態，每次移動前，會先去記錄目前狀態，包括分數、Cube 位置等等。一旦使用者按下 SPACE 之後，則可以根據你記錄的前一步的狀態來做 Undo。作法就是必須要有額外的參數來去記錄前一步的狀態。

Reset： 使用 Reset 之後，遊戲從頭開始，作法就是直接去呼叫 Start()，因為 Start()就是執行初始化的動作，只是已經創好的 Objects 不會再去重創一次。

Pause：若在遊戲中，當按下按鍵 ESC 之後，則會暫停遊戲，此時除了 Reset 與返回主選單操作之外，其他控制都不會有任何反應，直到再次按下按鍵 ESC 方可繼續遊戲。

返回主選單：若在遊戲中，當按下按鍵 B 之後，則會載入主選單之場景，並切換。

記分板：紀錄當前玩家透過合併分數字相同的 Cube 所取得之遊戲總分。

● Experiments

雖然 Cube 的實際位置不改變，動畫是一種錯覺，但是實際上玩起來的感覺就像是每個 Cube 真的有在移動它的位置，而且分數合併、變換 texture 的時候很自然。在 3*3*3 的 Camera 部分，我們 Camera 也是有移動動作的，而不是瞬間就切換到某個位置,如此一來在換Camera 的時候 ，帶給玩家一種有在轉的視覺，讓玩家不會迷失方向感，我們一開始的做法是直接建 4 個 Camera，然後 Cameras 之間互相切換，但是這樣會沒有轉的感覺，很容易就搞不清楚方向。

再來是時間模式的部分，光源的位置變化帶給玩家一種太陽升起、落下的感覺，從光源的變化搭配背景音樂來帶給玩家一種時間緊迫的緊張刺激。

● Discussion

我們這一組之前沒有接觸過 Unity，這次的 final project 是我們 Unity 的首作。我們之前聽過 Unity 是一個開發遊戲的平台，且很適合我們 Graphics 3D model 的主題，所以認為用它來寫一款 2048 的遊戲應該不是太難，但事實上十分複雜，寫法真的可以是千變萬化，不管你遊戲內部是怎麼運作的，目的都是要讓使用者感覺起來是一款富有視覺、聽覺、趣味性的遊戲。

在課程中學習到的 lighting、3D model、texture mapping、viewing transformation 等等，都可以透過 Unity 來整合實作，讓我們更了解這些不同的元素，混合在一起會帶來什麼樣的感受。