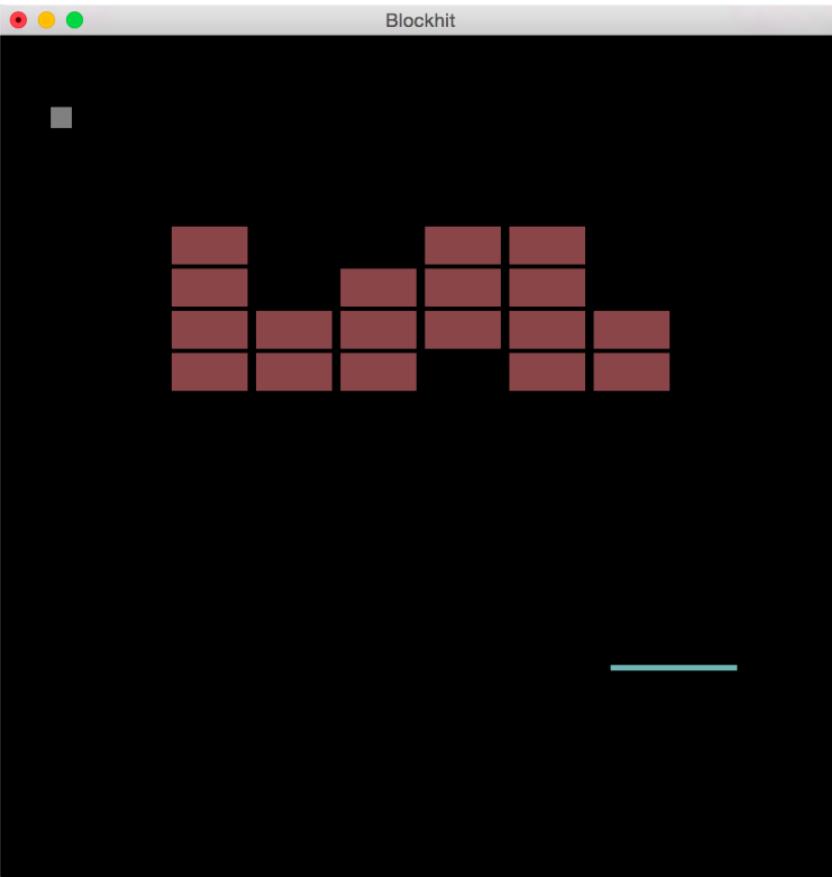


小遊戲



小遊戲需要？

小遊戲需要？

- 控制一個圖形視窗

小遊戲需要？

- 控制一個圖形視窗

圖形介面 vs 終端機介面

小遊戲需要？

- 控制一個圖形視窗

圖形介面 vs 終端機介面

- 繪製圖形

小遊戲需要？

- 控制一個圖形視窗

圖形介面 vs 終端機介面

- 繪製圖形
- 處理鍵盤和滑鼠

小遊戲需要？

- 控制一個圖形視窗

圖形介面 vs 終端機介面

- 繪製圖形
- 處理鍵盤和滑鼠

可是這些要怎麼用 c++ 達成??

小遊戲需要？

- 控制一個圖形視窗

圖形介面 vs 終端機介面

- 繪製圖形
- 處理鍵盤和滑鼠

可是這些要怎麼用 c++ 達成??

使用 **GLUT**

OpenGL是什麼？

OpenGL是什麼？



OpenGL是什麼？



- 一個跨平台，跨程式語言的應用程式介面

OpenGL是什麼？



- 一個跨平台，跨程式語言的應用程式介面
- GL = Graphic library

OpenGL是什麼？



- 一個跨平台，跨程式語言的應用程式介面
- GL = Graphic library
- 簡單講，就是提供了一堆和顯卡溝通、進行圖形渲染的函數

GLUT是什麼？

GLUT是什麼？

- 全名: OpenGL Utility Tool

GLUT是什麼？

- 全名: OpenGL Utility Tool
- 光只有OpenGL是不行的（很不方便）

GLUT是什麼？

- 全名: OpenGL Utility Tool
- 光只有OpenGL是不行的（很不方便）
- OpenGL只有做圖形渲染，但應用上通常需要其他周邊的搭配。
 - 像是 圖形視窗

GLUT是什麼？

- 全名: OpenGL Utility Tool
- 光只有OpenGL是不行的（很不方便）
- OpenGL只有做圖形渲染，但應用上通常需要其他周邊的搭配。
 - 像是 圖形視窗
- GLUT延伸了其他功能
 - 控制一個圖形視窗
 - 處理鍵盤和滑鼠

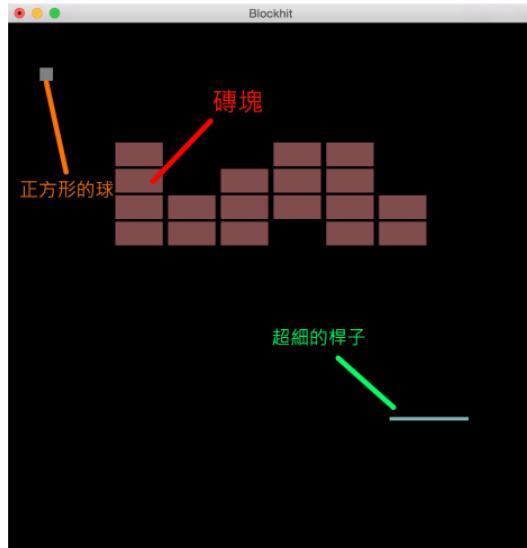
作業部分

大概

- 一個未完成且陽春的打磚塊小遊戲

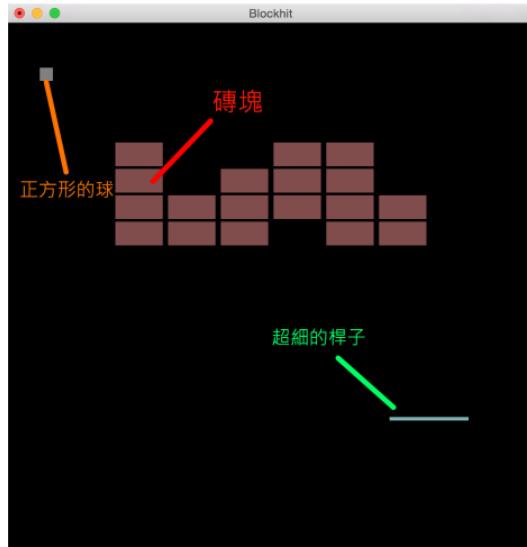
大概

- 一個未完成且陽春的打磚塊小遊戲



大概

- 一個未完成且陽春的打磚塊小遊戲



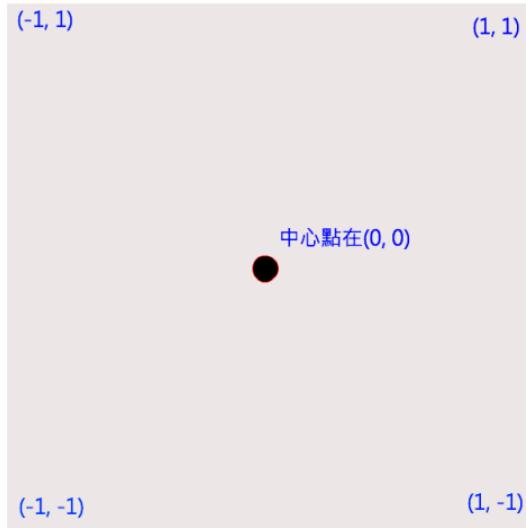
- 要寫的部分是三塊挖空的函數

座標系

- 既然需要繪製東西，那需要座標系：笛卡兒座標系

座標系

- 既然需要繪製東西，那需要座標系：笛卡兒座標系



- 範圍: $x \in [-1, 1]$, $y \in [-1, 1]$

三個函數

這三個挖空的函數不需要調用 **GLUT** 函數，在每個函數的前面都有關於這函數的Spec，只要把實作三個函數正確，就可以正確的跑出一個打磚塊的遊戲。

三個函數

這三個挖空的函數不需要調用 **GLUT** 函數，在每個函數的前面都有關於這函數的Spec，只要把實作三個函數正確，就可以正確的跑出一個打磚塊的遊戲。

```
bool IsInBox(float x, float y, float w, float h,  
            float cx, float cy);  
bool IsLegalPoint(float x, float y);  
void BlockDelete(float x, float y);
```

IsInBox

給一個點和一個長方形，判斷是否在裡面

```
bool IsInBox(float x, float y, float w, float h,  
            float cx, float cy);
```

IsInBox

給一個點和一個長方形，判斷是否在裡面

```
bool IsInBox(float x, float y, float w, float h,  
             float cx, float cy);
```

- 參數：

- `x, y`：點的x座標, y座標
- `w, h`：長方形寬度, 高度
- `cx, cy`：長方形中心x座標, y座標

IsInBox

給一個點和一個長方形，判斷是否在裡面

```
bool IsInBox(float x, float y, float w, float h,  
             float cx, float cy);
```

- 參數：

- `x, y`：點的x座標, y座標
- `w, h`：長方形寬度, 高度
- `cx, cy`：長方形中心x座標, y座標

- 定義：

IsLegalPoint

給一個點，判斷是否**不在**任何磚塊裡面且沒有出界。

```
bool IsLegalPoint(float x, float y);
```

IsLegalPoint

給一個點，判斷是否**不在**任何磚塊裡面且沒有出界。

```
bool IsLegalPoint(float x, float y);
```

- 參數：
 - x, y : 測試x座標, y座標

IsLegalPoint

給一個點，判斷是否**不在**任何磚塊裡面且沒有出界。

```
bool IsLegalPoint(float x, float y);
```

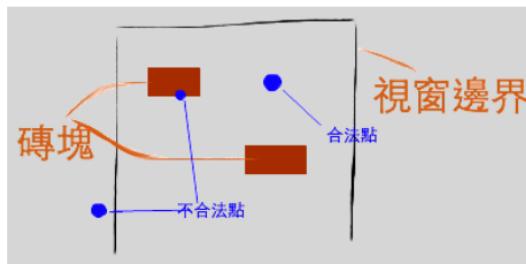
- 參數：
 - x, y ：測試 x 座標, y 座標
- 出界定義： $x \notin [-1, 1]$ 或 $y > 1$ (就是視窗邊界，不過把 $-1 > y$ 拿掉的原因是要判斷球是否掉下去)

IsLegalPoint

給一個點，判斷是否**不在**任何磚塊裡面且沒有出界。

```
bool IsLegalPoint(float x, float y);
```

- 參數：
 - x, y ：測試 x 座標, y 座標
- 出界定義： $x \notin [-1, 1]$ 或 $y > 1$ (就是視窗邊界，不過把 $-1 > y$ 拿掉的原因是要判斷球是否掉下去)



BlockDelete

給一個點，把覆蓋到該點的磚塊都移除

```
void BlockDelete(float x, float y);
```

BlockDelete

給一個點，把覆蓋到該點的磚塊都移除

```
void BlockDelete(float x, float y);
```

- 參數：
 - x, y : x座標, y座標

BlockDelete

給一個點，把覆蓋到該點的磚塊都移除

```
void BlockDelete(float x, float y);
```

- 參數：
 - `x, y` : x座標, y座標
- 對應到遊戲的主要功能：磚塊消除

可能需要操作的全域變數

- `IsLegalPoint`, `BlockDelete` 需要用到的全域變數

```
float block_x[100], block_y[100];
float block_width, block_height;
int block_cnt;
```

- `block_width` 記錄磚塊的寬，`block_height` 記錄磚塊的長(或高)
- `block_cnt` 記錄當前有多少個磚塊
- `block_x[0], block_y[0] ~ block_x[block_cnt - 1], block_y[block_cnt - 1]` 記錄了磚塊的中心座標

IsLegalPoint-Example

```
block_width = 0.1, block_height = 0.4
```

block_cnt = 2	i = 0	i = 1
block_x[i]	0.2	-0.3
block_y[i]	0.3	-0.4

IsLegalPoint-Example

```
block_width = 0.1, block_height = 0.4
```

block_cnt = 2	i = 0	i = 1
block_x[i]	0.2	-0.3
block_y[i]	0.3	-0.4

- IsLegalPoint(0.8, 0.9) : True

IsLegalPoint-Example

```
block_width = 0.1, block_height = 0.4
```

block_cnt = 2	i = 0	i = 1
block_x[i]	0.2	-0.3
block_y[i]	0.3	-0.4

- IsLegalPoint(0.8, 0.9) : True
- IsLegalPoint(0.24, 0.4) : False

IsLegalPoint-Example

```
block_width = 0.1, block_height = 0.4
```

block_cnt = 2	i = 0	i = 1
block_x[i]	0.2	-0.3
block_y[i]	0.3	-0.4

- IsLegalPoint(0.8, 0.9) : True
- IsLegalPoint(0.24, 0.4) : False
- IsLegalPoint(-1.01, 0.4) : False

BlockDelete-Example

```
block_width = 0.1, block_height = 0.4
```

block_cnt = 2	i = 0	i = 1
block_x[i]	0.2	-0.3
block_y[i]	0.3	-0.4

BlockDelete-Example

```
block_width = 0.1, block_height = 0.4
```

block_cnt = 2	i = 0	i = 1
block_x[i]	0.2	-0.3
block_y[i]	0.3	-0.4

- BlockDelete(0.24, 0.4)

BlockDelete-Example

block_width = 0.1, block_height = 0.4

block_cnt = 2	i = 0	i = 1
block_x[i]	0.2	-0.3
block_y[i]	0.3	-0.4

- BlockDelete(0.24, 0.4)
- 發現 i = 0 的block覆蓋到了(0.24, 0.4)

BlockDelete-Example

block_width = 0.1, block_height = 0.4

block_cnt = 2	i = 0	i = 1
block_x[i]	0.2	-0.3
block_y[i]	0.3	-0.4

- BlockDelete(0.24, 0.4)
- 發現 i = 0 的 block 覆蓋到了(0.24, 0.4)

block_cnt = 1	i = 0
block_x[i]	-0.3
block_y[i]	-0.4