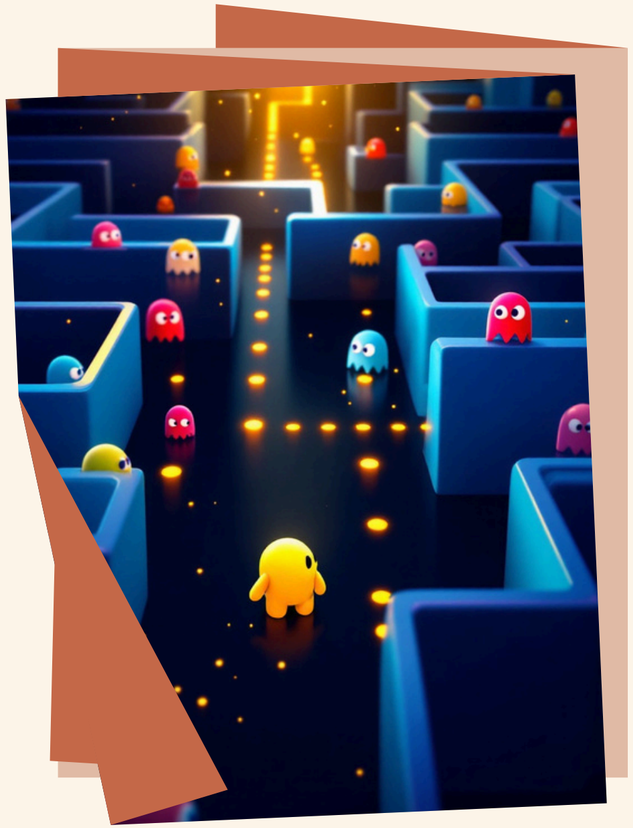
C++小遊戲

Part 4

PACMAN



經典的Pacman遊戲，玩家控制Pacman在迷宮中移動，吃掉點數和能量球，同時躲避鬼魂。玩家目標是吃掉所有點數並不要被追趕你的鬼抓到。

架構與邏輯

Part 4-1

迷宮和遊戲元素

迷宮設計

遊戲使用字符數組表示迷宮，其中 # 代表牆壁，
. 代表點數，* 代表能量球，**0** 代表Pacman，
字母代表鬼魂。迷宮大小為15 x 66。

```
########
#......#
#.0##.##
#..##.##
#*..a.##
########
```

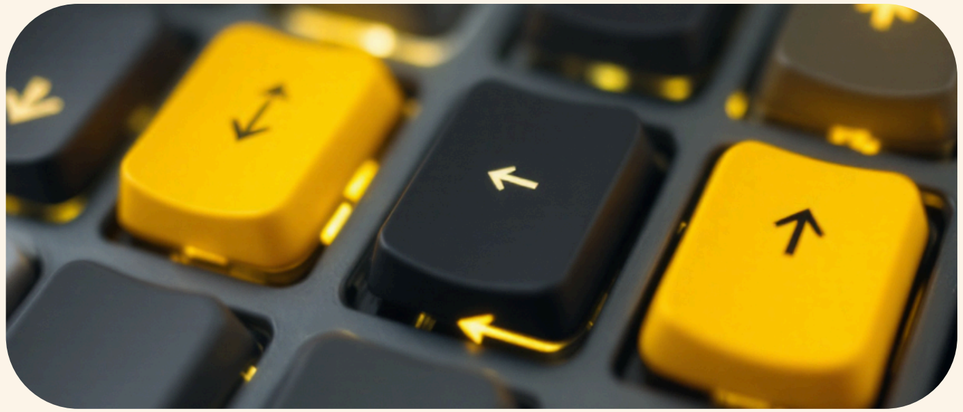
遊戲元素

- Pacman：由玩家控制，可以吃點數和能量球
- 點數：吃掉獲得10分
- 能量球：吃掉獲得50分並進入驚嚇模式
- 鬼魂：四種不同類型，各有不同的追蹤策略

鍵盤輸入處理

KeyManager結構

創建一個獨立線程持續記錄最後按下的方向鍵。支持方向鍵和WASD兩種控制方式，將按鍵轉換為對應的方向索引。



輸入處理

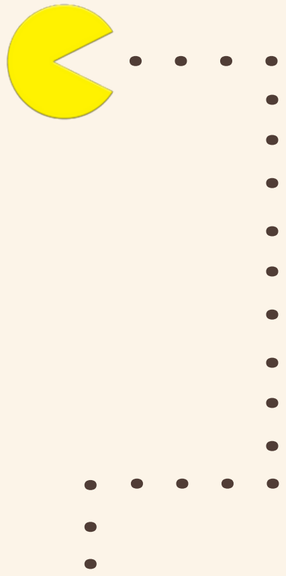
通過Windows API讀取控制台輸入，識別方向鍵和WASD鍵，更新lastDir變數，供遊戲主循環使用。

方向索引

D , →	S , ↓	A , ←	W , ↑
0	1	2	3

PACMAN移動邏輯

1. **獲取輸入** 從KeyManager獲取最後按下的方向鍵
2. **檢查碰撞** 使用checkNotCollideWall函數檢查是否會與牆壁碰撞
3. **更新方向** 更新curDir變數，用於後續的鬼魂移動計算
4. **更新位置** 如果不會碰撞，則根據方向更新Pacman位置



鬼魂追蹤邏輯

紅色鬼

直接追蹤Pacman的當前位置，最具攻擊性

橘色鬼

當距離Pacman較遠時追蹤，距離近時保持4格距離

粉紅鬼

瞄準Pacman前方4格的位置，試圖預測並攔截

淺藍鬼

以Pacman前方2格為中心，與紅鬼位置相對，形成包圍

鬼魂追蹤邏輯

紅色鬼(a)

直接追蹤Pacman的當前位置，最具攻擊性

```
#############################
#.....###.....###########
#.#######..#..##.a....##
#.....###..#..##..###.##
#.###.######.....###..#
#.###...#p....#######..#
#.......................#
#############################
```

鬼魂追蹤邏輯

粉紅鬼(b)

瞄準Pacman前方4格的位置，試圖預測並攔截

```
#############################
#.....###.....###########
#.#######..#..##.b....##
#.....###..#..##..###.##
#.###.######.....###..#
#.###...#0...#######..#
#.......................#
#############################
```

鬼魂追蹤邏輯

淺藍鬼(c)

以Pacman前方2格為中心，與紅鬼位置相對，形成包圍

```
#########################
#.....###.....##########
#.#######c.#..##......##
#.....###..#..##..###.##
#.###.######.....##...#
#.###...#0.....#######..#
#.a.......................#
#########################
```

鬼魂追蹤邏輯

橘色鬼(d)

當距離Pacman較遠時追蹤，距離近時保持4格距離

```
##########################
#.....###.....##########
#.#######..#..##......##
#.....###..#..##..###.##
#.###.######.....d###..#
#.###...#o....#######..#
#                dis >= 4
#......................#
##########################
```

鬼魂追蹤邏輯

橘色鬼(d)

當距離Pacman較遠時追蹤，距離近時保持4格距離

```
##########################
#.....###.....##########
#.#######..#..##......##
#.....###..#..##..###.##
#.###.######.....###..#
#.###...#0.d.X#######..#
#.......dis < 4.........#
##########################
```

驚嚇模式

被驚嚇的鬼

所有鬼魂隨機移動，
當遇到多個可行進方
向時將會隨機選擇

```
##########################
#A....###.....##########
#A#######..#..##C.....##
#....B###..#..##CC###.##
#.###B######.....###..#
#.###...#O....#######..#
#.............DD....#
##########################
```

驚嚇模式

觸發條件

Pacman吃掉能量球

鬼魂變化

鬼魂變為可被食用狀態，記錄為大寫字母

鬼魂特性

鬼魂會隨機移動，且速度減半。

持續時間

20個時間單位（10秒）

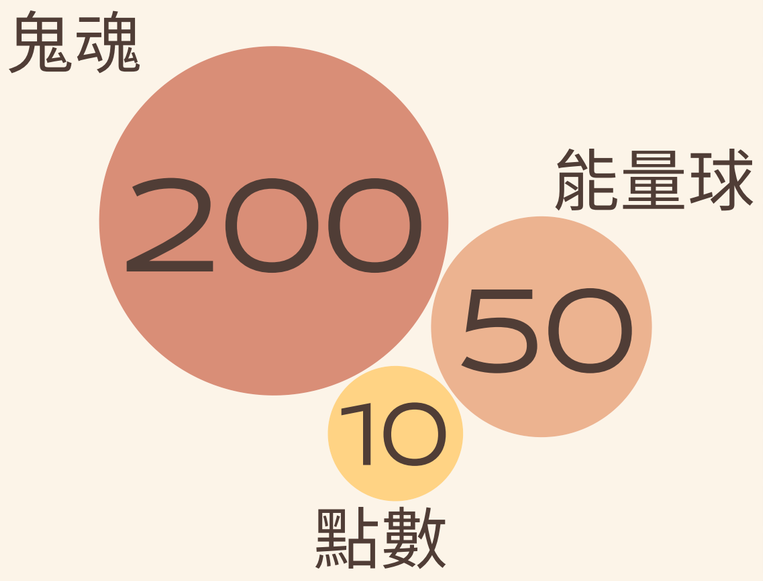
碰撞檢測與得分系統

遊戲中有三種碰撞檢測：

1. **Pacman與牆壁**(checkNotCollideWall)
   檢查是否會與牆壁碰撞

2. **Pacman與點數/能量球**
   吃掉點數時得10分，吃掉能量球時得50分並進入驚嚇模式

3. **Pacman與鬼魂**(checkCollideGhost)
   正常模式:碰觸鬼魂時遊戲結束
   驚嚇模式:吃掉鬼魂時得200分。



輸出美化

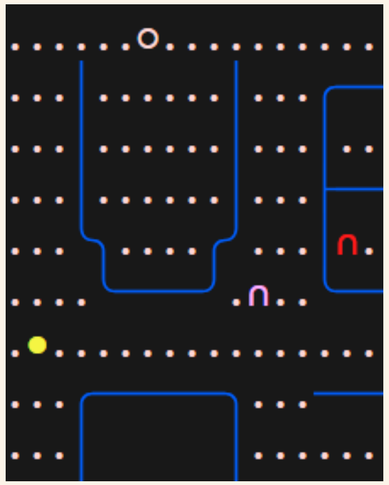
字符藝術

使用不同的Unicode字符表示遊戲元素，如"○"表示能量球。"╮ ", "═"等表示牆壁

ANSI顏色

使用ANSI轉義序列為不同元素添加顏色，如黃色Pacman，藍色牆壁，彩色鬼魂

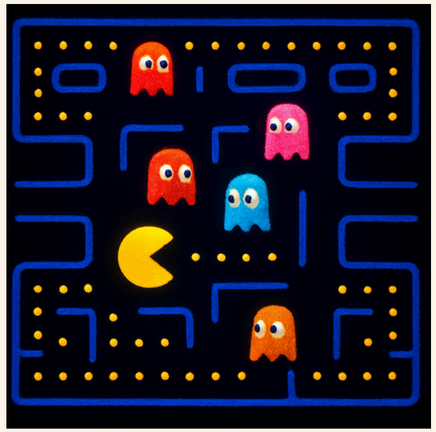
```
########
#......#
#.0##.##
#..##.##
#*..a.##
########
```



優化渲染

只更新變化的部分，通過比較當前網格和前一個網格，減少重繪量提高效率

遊戲流程與結束條件



初始化

設置遊戲地圖、放置能量球、初始化PACMAN和鬼魂位置、開始監聽鍵盤。

遊戲循環

每0.5秒循環更新PACMAN和鬼魂位置，檢查碰撞，更新遊戲狀態，渲染畫面。

勝利條件

點數和能量球都被吃完時，玩家獲勝。

失敗條件

在非驚嚇模式下被鬼魂碰到時，遊戲結束。

遊戲實作

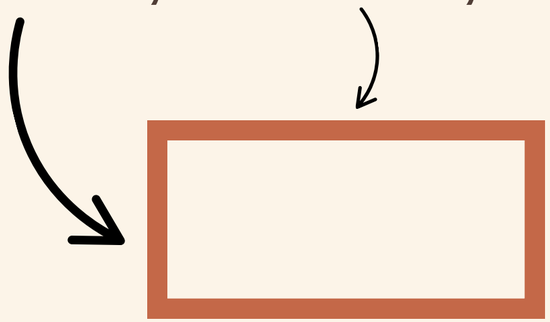
-main.cpp

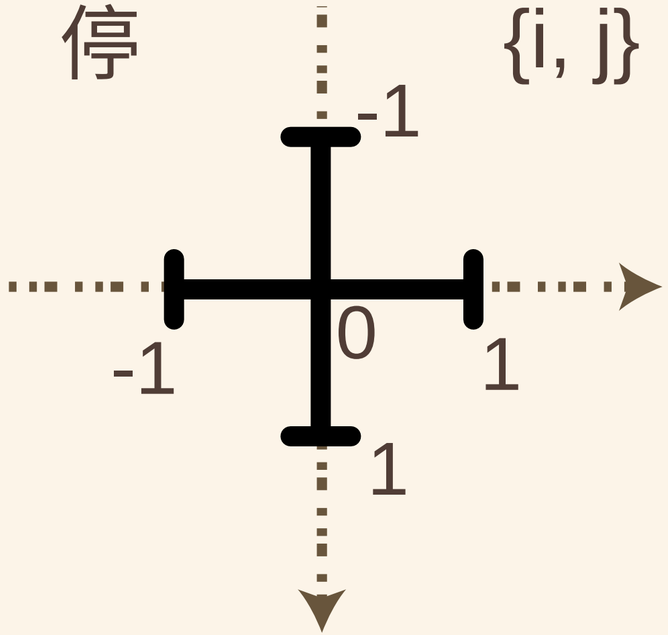
Part 4-2

全域變數

1. int dir[5][2] =
{{0, 1}, {1, 0}, {0, -1}, {-1, 0}, {0, 0}};
右 下 左 上 停

2. int HEIGHT, WIDTH;

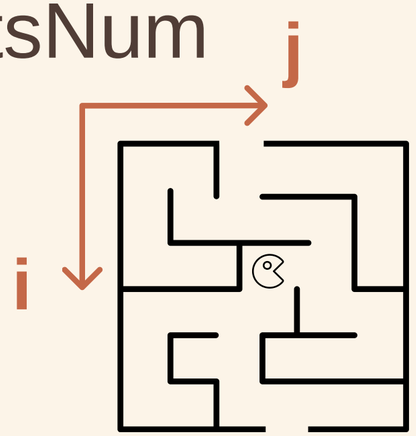




變數

1. **int** powerNum, ghostNum, pointsNum

2. **int** playerPos[2], playerDir

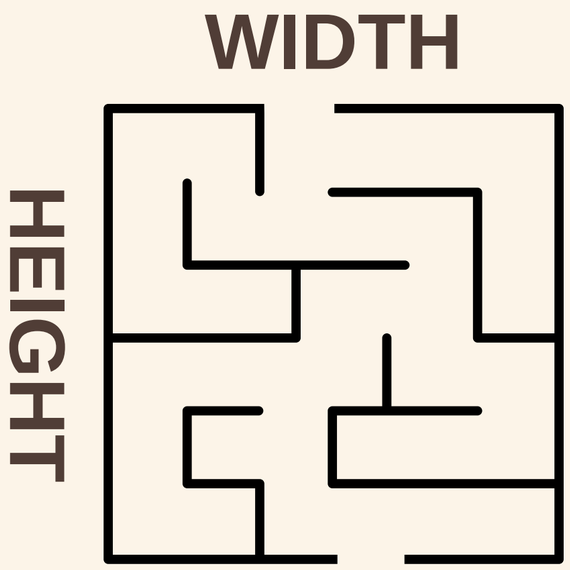


3. **int** ghostPos[ghostNum][2], ghostDir[ghostNum]
char ghostType[ghostNum]

變數

4 **int** score, scaredTime
bool scaredMode

5 **string** pointsGrid[HEIGHT],
gridDisplay[HEIGHT],
prevGridDisplay[HEIGHT]



main函式

功能

遊戲的主循環，初始化遊戲狀態，處理玩家輸入，更新遊戲狀態，並顯示遊戲畫面。

流程

1. 初始化遊戲區域、玩家和鬼的位置。
2. 隨機放置能量球和鬼。
3. 顯示遊戲開始提示，等待玩家按鍵。
4. 進入遊戲循環，直到遊戲結束：
   移動 PACMAN。
   檢查是否與鬼碰撞。
   移動鬼。
   檢查是否與鬼碰撞。
   更新顯示遊戲狀態。
5. 遊戲結束後顯示最終分數。

checkNotCollideWall

功能

檢查指定位置是否與牆壁碰撞。

參數

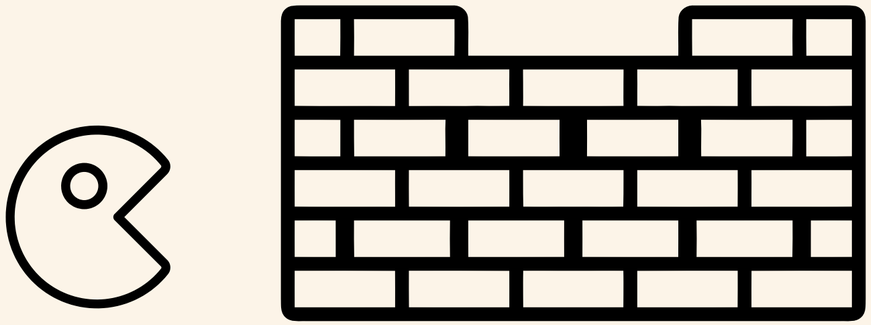
初始的地圖，i j 索引

回傳值

布林值 (TRUE/FALSE)

流程

1. 檢查索引是否超出邊界。
2. 檢查指定位置是否為牆壁 ('#')。
3. 回傳檢查結果。



checkNotCollideWall

```
bool checkNotCollideWall(string grid[], int i, int j) {
    int n = HEIGHT, m = WIDTH;
    if (i < 0 || i >= n || j < 0 || j >= m || grid[i][j] == '#')
        return false;  // 如果超出邊界或碰到牆壁，回傳false
    return true;  // 否則回傳true
}
```

checkCollideGhost

功能

檢查玩家是否與鬼碰撞。

參數

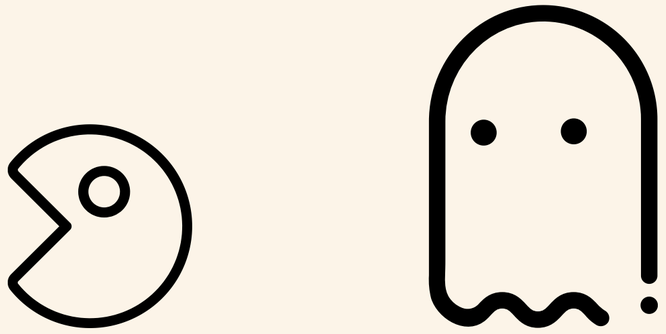
所有鬼的位置，玩家位置，
鬼的數量

回傳值

布林值 (TRUE/FALSE)

流程

1. 遍歷所有鬼的位置。
2. 檢查玩家位置是否與任何鬼的位置相同。
3. 回傳檢查結果。



checkCollideGhost

```
bool checkCollideGhost(int (*ghostPos)[2], int playerPos[2], int ghostNum) {
    for (int i = 0; i < ghostNum; i++) {
        if (ghostPos[i][0] == playerPos[0] && ghostPos[i][1] == playerPos[1]) {
            return true;  // 如果pacman與鬼的座標相同，回傳true
        }
    }
    return false;  // 否則回傳false
}
```

bestDir

功能

計算從當前位置到目標位置的最佳移動方向。

參數

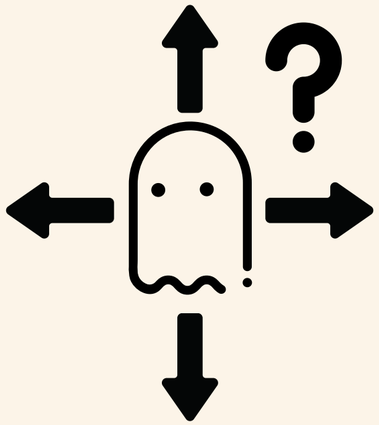
初始的地圖，當前位置方向，目標位置

回傳值

整數 (代表最佳方向)

流程

1. 初始化最佳方向和最小距離。
2. 遍歷所有可能的方向。
3. 檢查每個方向是否有效且不會碰到牆壁。
4. 計算到目標的距離，更新最佳方向。
5. 回傳最佳方向。



bestDir

```
int bestDir(string grid[], int curPos[2], int curDir, int target[2]) {
    int n = HEIGHT, m = WIDTH;

    int ans = 4;  // 預設為停止
    int minDis = INT16_MAX;  // 最小距離初始化為最大值
    for (int i = 0; i < 4; i++) {
        if (i == (curDir + 2) % 4)
            continue;  // 不考慮反方向

        int newI = curPos[0] + dir[i][0]; //新的i座標
        int newJ = curPos[1] + dir[i][1]; //新的j座標
        int dis = (newI - target[0]) * (newI - target[0]) +
                  (newJ - target[1]) * (newJ - target[1]);  // 計算直線距離的平方
        if (dis < minDis && checkNotCollideWall(grid, newI, newJ)) {
            ans = i;  // 更新最佳方向
            minDis = dis;  // 更新最小距離
        }
    }
    return ans;  // 回傳最佳方向
}
```

moveGhost

功能

移動鬼的位置。

參數

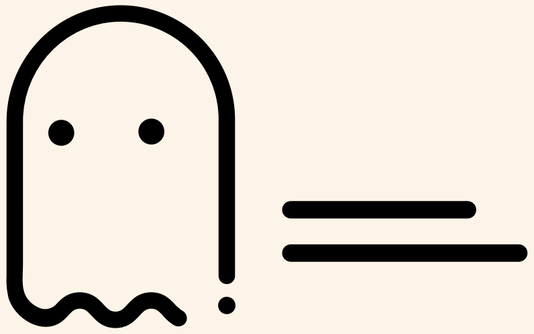
初始的地圖，所有鬼及玩家的位置及方向，驚嚇模式

回傳值

無

流程

1. 遍歷每個鬼。
2. 檢查每個鬼的合法移動方向。
3. 根據鬼的類型和驚嚇模式計算目標位置。
4. 更新鬼的位置。



moveGhost

```
template <int _N>
void moveGhost(string grid[], int (&ghostPos)[_N][2], int (&ghostDir)[_N],
                char ghostType[], int ghostNum, int playerPos[2],
                int playerDir, bool scaredMode) {
    //............
        int target[2] = {};
        if (ghostType[i] == 'a') {  // 紅鬼：目標pacman位置
            target[0] = playerPos[0];
            target[1] = playerPos[1];
        } else if (ghostType[i] == 'b') {  // 粉紅鬼：目標pacman位置前方4格
            target[0] = playerPos[0] + dir[playerDir][0] * 4;
            target[1] = playerPos[1] + dir[playerDir][1] * 4;
        } else if (ghostType[i] == 'c') {  // 淺藍鬼：與紅鬼相對，中心在pacman前方2格
            //...........
            int center[2];
            center[0] = playerPos[0] + dir[playerDir][0] * 2;
            center[1] = playerPos[1] + dir[playerDir][1] * 2;
            target[0] = center[0] * 2 - ghostPos[indexOfRed][0];
            target[1] = center[1] * 2 - ghostPos[indexOfRed][1];
        } else if (ghostType[i] == 'd') {  // 橙鬼：保持4格距離
            //............
        }
        ghostDir[i] = bestDir(grid, ghostPos[i], ghostDir[i], target);  // 計算最佳方向
    //............
}
```

eatGhost

功能

處理玩家吃掉鬼的情況。

參數

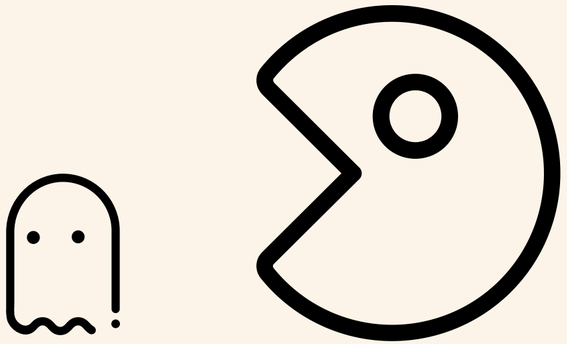
初始的地圖，鬼的位置、方向、種類及數量，玩家位置及分數

回傳值

無

流程

1. 遍歷所有鬼的位置。
2. 檢查玩家是否與鬼的位置相同。
3. 若吃掉鬼更新分數，重新生成鬼的位置，並將鬼的類型改回小寫。



eatGhost

```
void eatGhost(string grid[], int playerPos[2], int (*ghostPos)[2],
              int (&ghostDir)[], char (&ghostType)[], int ghostNum,
              int &score) {
    for (int i = 0; i < ghostNum; i++) {
        if (ghostPos[i][0] == playerPos[0] && ghostPos[i][1] == playerPos[1]) {
            score += 200;  // 吃掉鬼的得分
            respawnGhost(grid, ghostPos[i], playerPos);  // 重新生成鬼
            ghostDir[i] = 4;  // 設置鬼的方向為停止
            ghostType[i] = tolower(ghostType[i]);  // 將鬼的類型改為小寫
        }
    }
}
```

遊戲實作

-pacman.hpp

Part 4-3

gridPrint

功能

輸出遊戲當前地圖。

參數

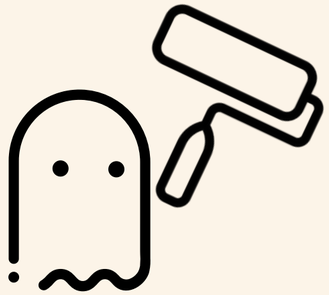
當前輸出用的地圖及前一個輸出用的地圖。

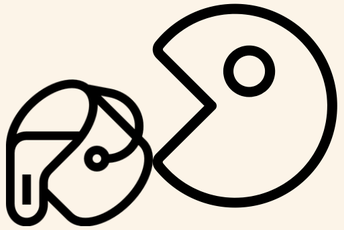
回傳值

無

流程

1. 初始化顯示網格。
2. 根據當前網格和前一個網格的變化輸出顏色和符號。
3. 更新前一個網格。





movePacman

功能

移動 PACMAN。

參數

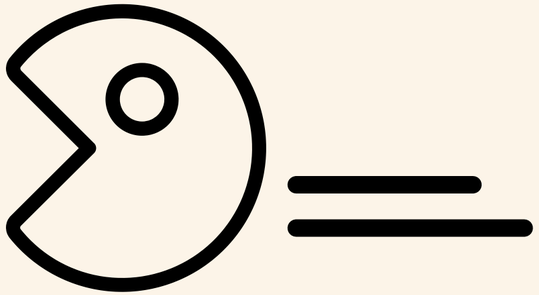
初始的地圖，玩家位置及方向，KeyManager

回傳值

無

流程

1. 讀取最後按下的鍵。
2. 檢查移動後的位置是否合法。
3. 更新玩家位置和當前方向。



respawnGhost

功能

重新生成被吃掉的鬼的位置。

參數

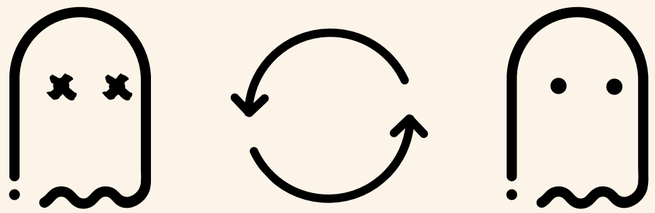
初始的地圖，被吃掉的鬼的位置，玩家位置

回傳值

無

流程

1.找到距離玩家最遠的位置。
2. 更新鬼的位置為該位置。



update

功能

更新遊戲狀態。

參數

初始的地圖，鬼的位置、方向、種類、數量，玩家位置，點數地圖，點數數量，輸出用地圖，玩家分數，驚嚇模式，驚嚇時間

回傳值

無

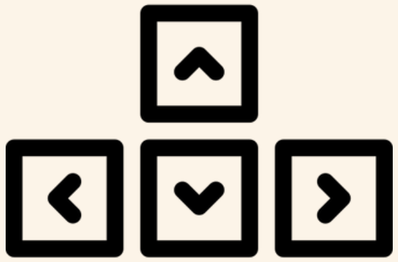
流程

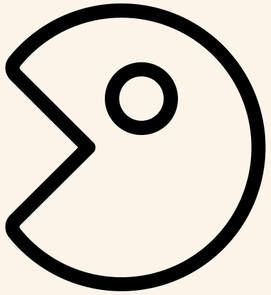
1. 重置顯示地圖。
2. 更新玩家和鬼的位置。
3. 檢查玩家是否吃掉了點數或能量球，並更新分數。
4. 更新驚嚇模式的狀態。

KeyManager 結構

功能

管理鍵盤輸入。
(箭頭或WASD)





成員函數

1. getLastKey():
   獲取最後被按下的鍵。
2. anyKeyToContinue():
   等待任意鍵繼續。
3. updateStatus():
   持續監聽鍵盤輸入並更新最新輸入。
4. readKey():
   讀取當前按鍵。

練習時間

Part 4-4

Part 4-4

練習時間

https://pse.is/7dgpmr

PACMAN :video_game

2025 NYCU EEcamp

HackMD