資工三 電腦圖學 期中考替代作業

1. 資工二乙 / 411262075 / 程聖哲

2. 題目：老鼠走迷宮

3. 程式架構

generateMaze：用於迷宮的隨機生成。

drawCell：根據cell的類型繪製對應目標（起始點、終點、牆壁、通道、路徑）。

drawMaze：使用drawCell在固定範圍繪製整個迷宮。

drawText：繪製文字。

setAgentStart：變更老鼠的起始點。

toggleWall：切換當前cell的狀態（牆壁、通道）。

mouseDrag：滑鼠拖曳狀態下切換路徑的cell狀態。

mouseClick：根據當前狀態（setWall, setMousePosition, findingPath），按下滑鼠的相對應事件。

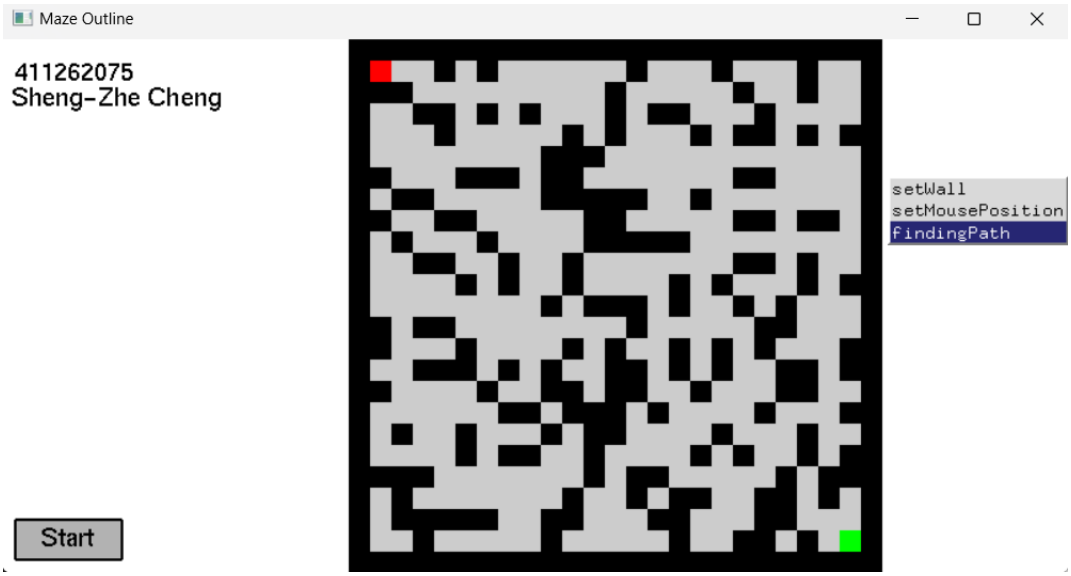
findingPath_BFS：使用 BFS

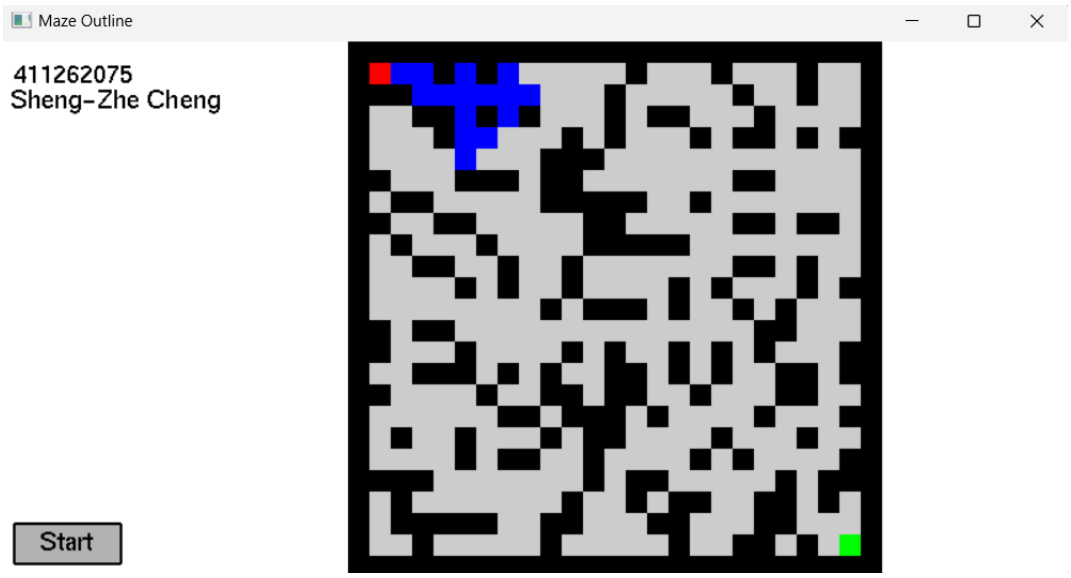
演算法，尋找期間不斷切換cell狀態（正確路徑設為2；正在探索的路徑設為3）

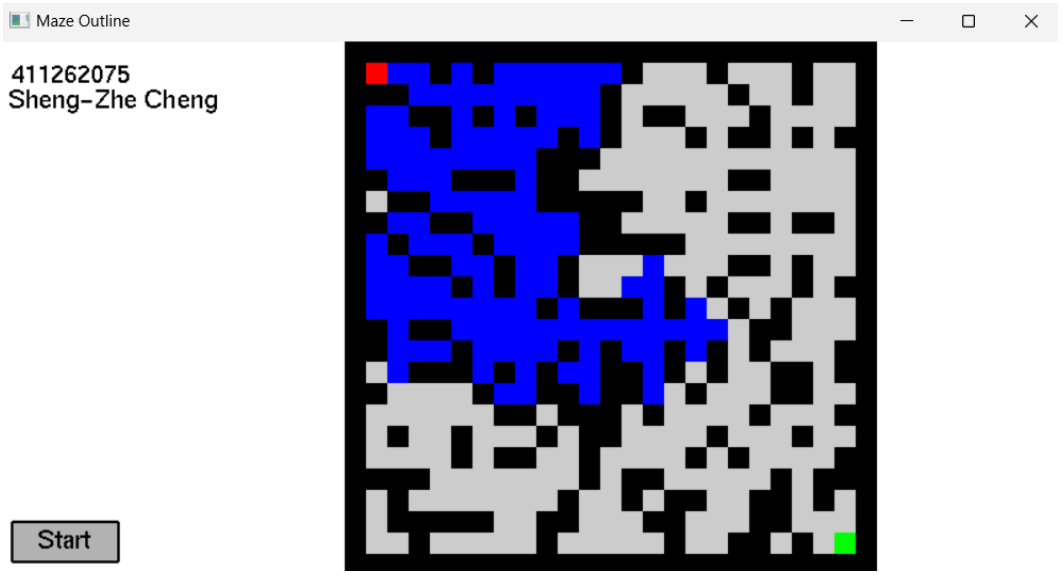
4. 討論

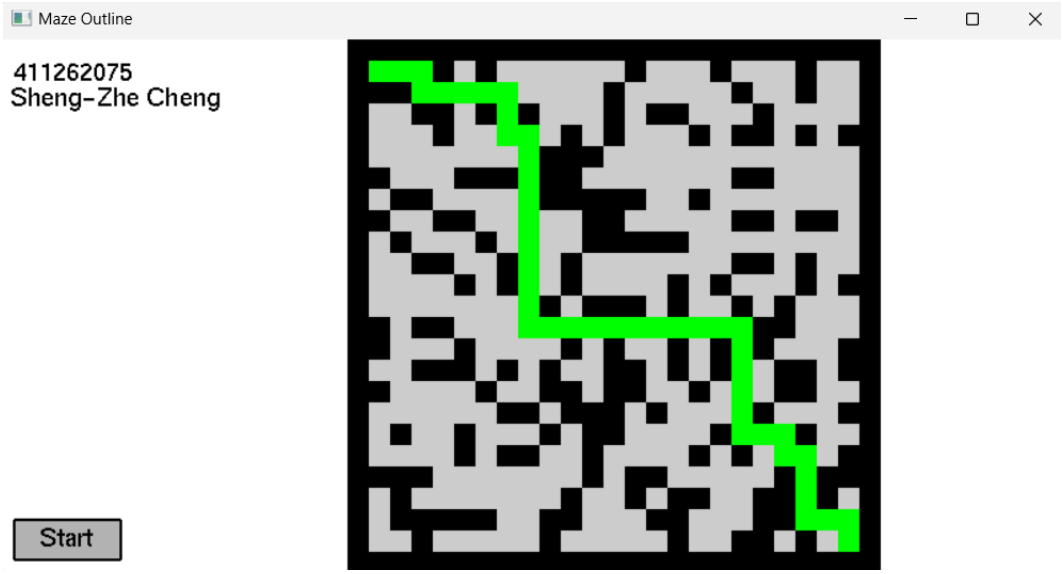
一開始因為是第一次實作繪圖的程式，環境的設定上出現不少問題，所以花了許多時間琢磨。起初程式的方向就是想要著重於可視化的搜尋效果，而後選擇採用BFS是因為相較於其他演算法，BFS相較簡單且繪圖是以起始點向外擴散，路徑尋找的過程容易清晰的表現。

5. 執行畫面









程式碼

```
#include <GL/glut.h>

#include <stdlib.h>

#include <time.h>

#include <queue>

#include <vector>

#include <cstdlib>

#include <windows.h>
```

```
using namespace std;

const int mazeWidth = 25;
const int mazeHeight = 25;
int windowWidth = 500;
int windowHeight = 500;
float offsetX, offsetY;
int cellSize;

int maze[mazeHeight][mazeWidth] = {};

int mode = 0;
bool isMousePressed = false; // 追蹤滑鼠按下狀態
int lastMouseRow = -1;
int lastMouseCol = -1;

int agentStartX = 1;
int agentStartY = 1;

bool visited[mazeHeight][mazeWidth] = {}; // 紀錄是否走過
bool searchCompleted = false;

struct Node {
    int row, col;
```

```
    std::vector<std::pair<int, int>> path; // 儲存當前位置
};

void drawPath() {
    for (int i = 0; i < mazeHeight; ++i) {
        for (int j = 0; j < mazeWidth; ++j) {
            if (maze[i][j] == 2) {
                int x = j * cellSize + offsetX;
                int y = (mazeHeight - 1 - i) * cellSize + offsetY;
                glColor3f(0.0, 1.0, 0.0);
                glBegin(GL_QUADS);
                glVertex2i(x, y);
                glVertex2i(x + cellSize, y);
                glVertex2i(x + cellSize, y + cellSize);
                glVertex2i(x, y + cellSize);
                glEnd();
            }
        }
    }
    glFlush();
}

void generateMaze() {
    for (int i = 0; i < mazeHeight; ++i) {
```

```
        for (int j = 0; j < mazeWidth; ++j) {
            if (i == 0 || i == mazeHeight - 1 || j == 0 || j == mazeWidth - 1) {
                maze[i][j] = 1; // 邊界固定為牆壁
            } else {
                maze[i][j] = (rand() % 100 < 30) ? 1 : 0; // 30% 機率為牆壁
            }
        }
    }
}

void init() {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    cellSize = windowWidth / mazeWidth < windowHeight / mazeHeight ? windowWidth / mazeWidth : windowHeight / mazeHeight;

    offsetX = (windowWidth - mazeWidth * cellSize) / 2.0f;
    offsetY = (windowHeight - mazeHeight * cellSize) / 2.0f;

    gluOrtho2D(0.0, windowWidth, 0.0, windowHeight);

    srand(time(NULL));
```

```
    generateMaze();
}

void reshape(int  w, int h) {
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    windowWidth = w;
    windowHeight = h;

    // 重新計算格子大小，使迷宮適應視窗
    cellSize = windowWidth / mazeWidth < windowHeight / mazeHeight ? windowWidth / mazeWidth : windowHeight / mazeHeight;

    // 預留空間，讓迷宮不遮蓋左側文字
    int leftMargin = 120; // 預留 120 像素的空間
    offsetX = leftMargin + (windowWidth - leftMargin - mazeWidth * cellSize) / 2.0f;
    offsetY = (windowHeight - mazeHeight * cellSize) / 2.0f;

    gluOrtho2D(0.0, windowWidth, 0.0, windowHeight);
    glMatrixMode(GL_MODELVIEW);
}
```

```
void drawCell(int row, int col) {
    int x = col * cellSize + offsetX;
    int y = (mazeHeight - 1 - row) * cellSize + offsetY;

    // 根據格子類型設置顏色
    if (maze[row][col] == 3) {
        glColor3f(0.0, 0.0, 1.0); // 藍色表示正在訪問的節點
    }
    else if (maze[row][col] == 2) {
        glColor3f(0.0, 1.0, 0.0); // 綠色表示正確路徑
    }
    else if (row == mazeHeight - 2 && col == mazeWidth - 2) {
        glColor3f(0.0, 1.0, 0.0); // 綠色表示終點
    } else if (row == agentStartY && col == agentStartX) {
        glColor3f(1.0, 0.0, 0.0); // 老鼠位置
    } else if (maze[row][col] == 1) {
        glColor3f(0.0, 0.0, 0.0); // 黑色牆壁
    } else {
        glColor3f(0.8, 0.8, 0.8); // 灰色通道
    }

    // 繪製單格
    glBegin(GL_QUADS);
    glVertex2i(x, y);
```

```
    glVertex2i(x + cellSize, y);
    glVertex2i(x + cellSize, y + cellSize);
    glVertex2i(x, y + cellSize);
    glEnd();
}

void drawMaze() {
    for (int i = 0; i < mazeHeight; ++i) {
        for (int j = 0; j < mazeWidth; ++j) {
            drawCell(i, j);
        }
    }
}

void drawText(const char* text, int x, int y) {

    glColor3f(0.0, 0.0, 0.0); // 黑色文字
    glRasterPos2i(x, y);

    while (*text) {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *text);
        text++;
    }
}
```

```
void setAgentStart(int  x, int y) {
    int col = (x - offsetX) / cellSize;
    int row = (y - offsetY) / cellSize;

    if (row >= 0 && row < mazeHeight && col >= 0 && col < mazeWidth) {
        agentStartX = col;
        agentStartY = row;
    }
}

void toggleWall(int  x, int y) {
    int col = (x - offsetX) / cellSize;
    int row = (windowHeight - y - offsetY) / cellSize;
    // 如果是邊界或終點，直接跳過
    if (row <= 0 || row >= mazeHeight - 1 || col <= 0 || col >= mazeWidth - 1 ||
        (row == mazeHeight - 1 && col == mazeWidth - 1)) {
        return;
    }
    maze[row][col] = (maze[row][col] == 1) ? 0 : 1; // 切換牆壁狀態
    drawCell(row, col); // 只更新改動的格子
}

void mouseDrag(int x, int y) {
```

```
    int col = (x - offsetX) / cellSize;
    int row = (y - offsetY) / cellSize;
    if (isMousePressed && (row != lastMouseRow || col != lastMouseCol)) {
        toggleWall(x, windowHeight - y); // 切換拖曳位置牆壁
        glFlush();
        lastMouseRow = row;
        lastMouseCol = col;
    }
}

void menu(int item) {
    switch (item) {
        case 0:
            for (int i = 0; i < mazeHeight; ++i) {
                for (int j = 0; j < mazeWidth; ++j) {
                    if (maze[i][j] == 2) {
                        maze[i][j] = 0;
                    }
                }
            }
            searchCompleted = false;
            glutPostRedisplay();
            mode = 0;
            break;
```

```
    case 1:
        for (int i = 0; i < mazeHeight; ++i) {
            for (int j = 0; j < mazeWidth; ++j) {
                if (maze[i][j] == 2) {
                    maze[i][j] = 0;
                }
            }
        }
        searchCompleted = false;
        glutPostRedisplay();
        mode = 1;
        break;
    case 2:
        for (int i = 0; i < mazeHeight; ++i) {
            for (int j = 0; j < mazeWidth; ++j) {
                if (maze[i][j] == 2) {
                    maze[i][j] = 0;
                }
            }
        }
        searchCompleted = false;
        mode = 2;
        break;
}
```

```
    glutPostRedisplay();
}

void drawButton(const char* text, int x, int y, int width, int height) {
    // 繪製按鈕背景
    glColor3f(0.7, 0.7, 0.7); // 灰色背景
    glBegin(GL_QUADS);
    glVertex2i(x, y);
    glVertex2i(x + width, y);
    glVertex2i(x + width, y + height);
    glVertex2i(x, y + height);
    glEnd();

    // 繪製按鈕邊框
    glColor3f(0.0, 0.0, 0.0); // 黑色邊框
    glLineWidth(2.0);
    glBegin(GL_LINE_LOOP);
    glVertex2i(x, y);
    glVertex2i(x + width, y);
    glVertex2i(x + width, y + height);
    glVertex2i(x, y + height);
    glEnd();

    // 在按鈕內部顯示文字
```

```
    glColor3f(0.0, 0.0, 0.0); // 黑色文字
    glRasterPos2i(x + width / 4, y + height / 3);
    while (*text) {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *text);
        text++;
    }
}

bool findingPath_BFS(int startX, int startY) {
    std::queue<Node> q;
    q.push({startY, startX, {{startY, startX}}});
    visited[startY][startX] = true;

    // 上下左右四個方向
    int dx[] = {0, 1, 0, -1};
    int dy[] = {-1, 0, 1, 0};

    bool pathFound = false;

    while (!q.empty() && !pathFound) {
        Node current = q.front();
        q.pop();

        // 如果找到終點，標記路徑並停止
```

```
if (current.row == mazeHeight - 2 && current.col == mazeWidth - 2) {
    for (auto& p : current.path) {
        maze[p.first][p.second] = 2; // 標記為正確路徑
        drawCell(p.first, p.second);
        glFlush();
        Sleep(20); // 暫停以觀察
    }
    pathFound = true;
    searchCompleted = true;
    while(!q.empty()){ q.pop(); }
    break;
}
// 更新當前節點繪圖
if(!searchCompleted && maze[current.row][current.col] == 0 &&
   !(current.row == startY && current.col == startX)){
    maze[current.row][current.col] = 3; // 暫時標記為正在訪問的格子
    drawCell(current.row, current.col); // 重新繪製整個迷宮
    glFlush(); // 刷新畫面
    Sleep(25);
}

for (int i = 0; i < 4; ++i) {
    int newRow = current.row + dy[i];
    int newCol = current.col + dx[i];
```

```
            if (newRow >= 0 && newRow < mazeHeight && newCol >= 0 && newCol <
mazeWidth &&
                maze[newRow][newCol] == 0 && !visited[newRow][newCol]) {
                visited[newRow][newCol] = true;

                std::vector<std::pair<int, int>> newPath = current.path;
                newPath.push_back({newRow, newCol});
                q.push({newRow, newCol, newPath});
            }
        }

        if (maze[current.row][current.col] != 2) {
            maze[current.row][current.col] = 0;
        }
    }
    return pathFound;
}

void mouseClick(int button, int state, int x, int y) {

    if (button == GLUT_LEFT_BUTTON) {
        if(mode == 0){
            if (state == GLUT_DOWN) {
```

```
            isMousePressed = true;
            lastMouseCol = (x - offsetX) / cellSize;
            lastMouseRow = (y - offsetY) / cellSize;
            toggleWall(x, windowHeight - y); // 切換點擊位置牆壁
        } else if (state == GLUT_UP) {
            isMousePressed = false;
        }
        glFlush(); // 確保單個更新立即刷新
    }
    else if(mode == 1){
        setAgentStart(x,y);
    }
    else if (mode == 2 && (x >= 10 && x <= 90 && windowHeight - y >= 10 &&
windowHeight - y <= 40)) {
        if(searchCompleted)  return;
        // mode 2 且 按鈕位置被按下
        memset(visited,  false, sizeof(visited));
        if (findingPath_BFS(agentStartX,  agentStartY)) {
            drawPath();
        } else {
            drawText("No path found!", 10, windowHeight - 70);
        }
        glFlush();
    }
```

```
    }
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    drawMaze();
    glPushMatrix();
    glLoadIdentity();
    drawText("411262075", 10, windowHeight - 30);
    drawText("Sheng-Zhe Cheng", 8, windowHeight - 50);
    if(mode == 2){
        drawButton("Start", 10, 10, 80, 30);
    }
    glFlush(); // 單緩衝刷新
}

int main(int argc, char** argv) {
```