Markdown sheet

Installation Guide

1. Opencv2 Required. Link : <https://opencv.org/releases/>

2. Copy opencv/build/include to your solutiondir.

3. C/C++ -> General -> Additonal Dependencis : input the path where you placed the include downloaded from opencv

4. Linker -> input -> Additional Dependencis : opencv_world420.lib & opencv_world420d.lib.

5. Copy opencv_world420.dll & opencv_world420d.dll from opencvdir to solutiondir.

The concept of the Maze Game

1. Using an array to create a map

2. Render the maze by for-loop, the maze is combined by walls

3. Spawn the player in the place of '2'

4. The player can play in first-person and third-person

5. allow the player to move around the map using the WASD keys

6. when the player manages to get to '3' the finish the map (win)

7. the player can not pass through or move walls, '1' but can walk on empty tiles '0' as well as over the exit 'E'

Controller

F1: first-person view

F2: third-person view

UP: move forward

DOWN: move back

LEFT: left turn

RIGHT: right turn

Program brief

OpenCV 

1. Using Opencv lib to develop which is also included glut and glew.

```
Vertex vertexes[8];
xyzToVertex(vertexes[0], wall.x, wall.y, wall.z);
xyzToVertex(vertexes[1], wall.x + wall.size, wall.y, wall.z);
xyzToVertex(vertexes[2], wall.x + wall.size, wall.y + wall.size, wall.z);
xyzToVertex(vertexes[3], wall.x, wall.y + wall.size, wall.z);
xyzToVertex(vertexes[4], wall.x, wall.y, wall.z + wall.size);
xyzToVertex(vertexes[5], wall.x + wall.size, wall.y, wall.z + wall.size);
xyzToVertex(vertexes[6], wall.x + wall.size, wall.y + wall.size, wall.z + wall.size);
xyzToVertex(vertexes[7], wall.x, wall.y + wall.size, wall.z + wall.size);

//The edge of the wall
Quad4 quads[6];
vertexesToQuad(quads[0], vertexes[0], vertexes[1], vertexes[2], vertexes[3]);
vertexesToQuad(quads[1], vertexes[0], vertexes[1], vertexes[5], vertexes[4]);
vertexesToQuad(quads[2], vertexes[2], vertexes[3], vertexes[7], vertexes[6]);
vertexesToQuad(quads[3], vertexes[1], vertexes[2], vertexes[6], vertexes[5]);
vertexesToQuad(quads[4], vertexes[0], vertexes[3], vertexes[7], vertexes[4]);
vertexesToQuad(quads[5], vertexes[4], vertexes[5], vertexes[6], vertexes[7]);
```

2. The wall is rendered by 8 vertexes which are the corner of a rectangle. (and 6 edges of a rectangle, edge have consisted 4 vertexes.)

```
void drawMaze(Map map) {
    Wall wall;
    wall.size = MAP_BLOCK_LENGTH;
    for (int i = 0; i < map.width; i++) {
        for (int j = 0; j < map.height; j++) {
            if (map.blocks[i][j] == MAP_BLOCK_CUBE) {
                wall.x = j * wall.size;
                wall.y = map.height * wall.size - (i + 1) * wall.size;
                wall.z = 0;
                drawCube(wall, false);
            }
        }
    }

}
```

3. The maze is combined with many walls. (Using for-loop to read an array and render a map by a wall)

```
#define MAP_BLOCK_EMPTY 0        //Path
#define MAP_BLOCK_CUBE 1         //Wall
#define MAP_BLOCK_START 2        //Start of the Maze
#define MAP_BLOCK_END 3          //Exit of the Maze
#define MAP_BLOCK_LENGTH 30

// The design of map
const GLint MAP1_WIDTH = 20;
const GLint MAP1_HEIGHT = 20;
const GLint MAP1_BLOCKS[MAP1_WIDTH][MAP1_HEIGHT] = {
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
    1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
    1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
    1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
    1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
    1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
};
```

4. The map has 4 elements which are start position, end position, path and wall

```
if (viewMode == VIEW_MODE_FRIST_PERSON) {
    switch (key) {
    case GLUT_KEY_UP:
        switch (player.face) {
        case PLAYER_FACE_UP:
            if (player.x > 0) {
                if (map.blocks[player.x - 1][player.y] != MAP_BLOCK_CUBE) {
                    player.x -= 1;
                }
            }
            break;
        case PLAYER_FACE_DOWN:
            if (player.x < map.height - 1) {
                if (map.blocks[player.x + 1][player.y] != MAP_BLOCK_CUBE) {
                    player.x += 1;
                }
            }
```

5. Using if-else func to check collision, if there is a wall in front of the play who can't move forward.

```
int main(int argc, char *argv[]) {
    init();

    // init glut
    glutInit(&argc, argv);

    // init window
    glutInitWindowPosition(WINDOW_POSITION_X, WINDOW_POSITION_Y);
    glutInitWindowSize(WINDOW_SIZE_WIDTH, WINDOW_SIZE_HEIGHT);
    glutCreateWindow("Maze");
```

6. Generate the window and model by glut instruction/code.

A video playlist regarding the point produced at here: [PlayList]