서강대학교 SOGANG UNIVERSITY

컴퓨터공학설계및실험I
최종 프로젝트

DFS, BFS의 시각화

컴퓨터공학과 김민준 / 20201559

CSE3013-05

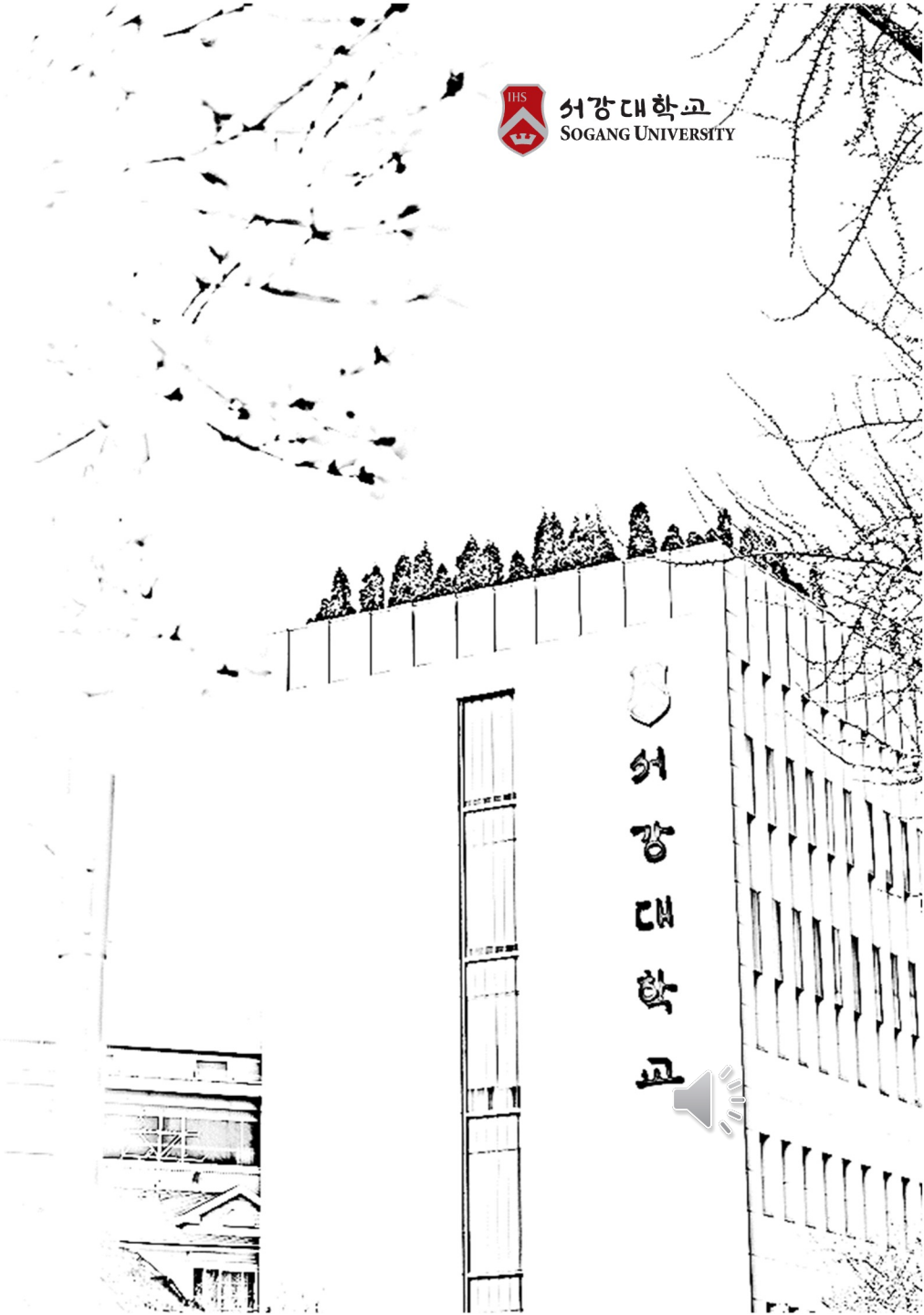
서강대학교 SOGANG UNIVERSITY

CONTENTS

SOGANG UNIVERSITY

01. 프로젝트 목표

DFS, BFS 알고리즘의 시각화



서강대학교 SOGANG UNIVERSITY

미로 생성

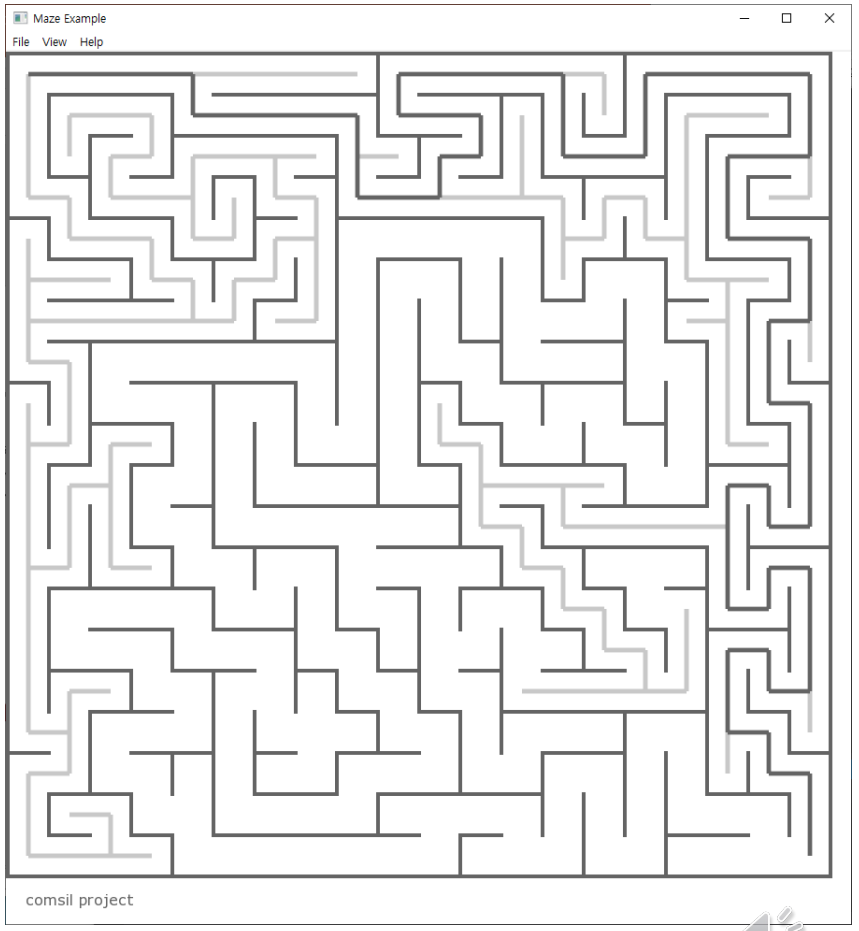
기존 프로젝트의 경우 새로운 미로를 만드는 기능이 없다.
11주차 실습에서 사용한 파일을 사용할때는 번거로운 측면이 있다.
해당 실습에서 사용한 Eller`s algorithm 을 사용하면 다소 수평 방향 벽이 많이 발생 한다.
이를 해결하기위하여 recursive backtracking algorithm 을 이용한 maze generator 기능을 추가하였다.

서강대학교 SOGANG UNIVERSITY

미로 탐색

기존 프로젝트의 경우 어떤 순서로 탐색을 진행하였는지 알 수 없다.

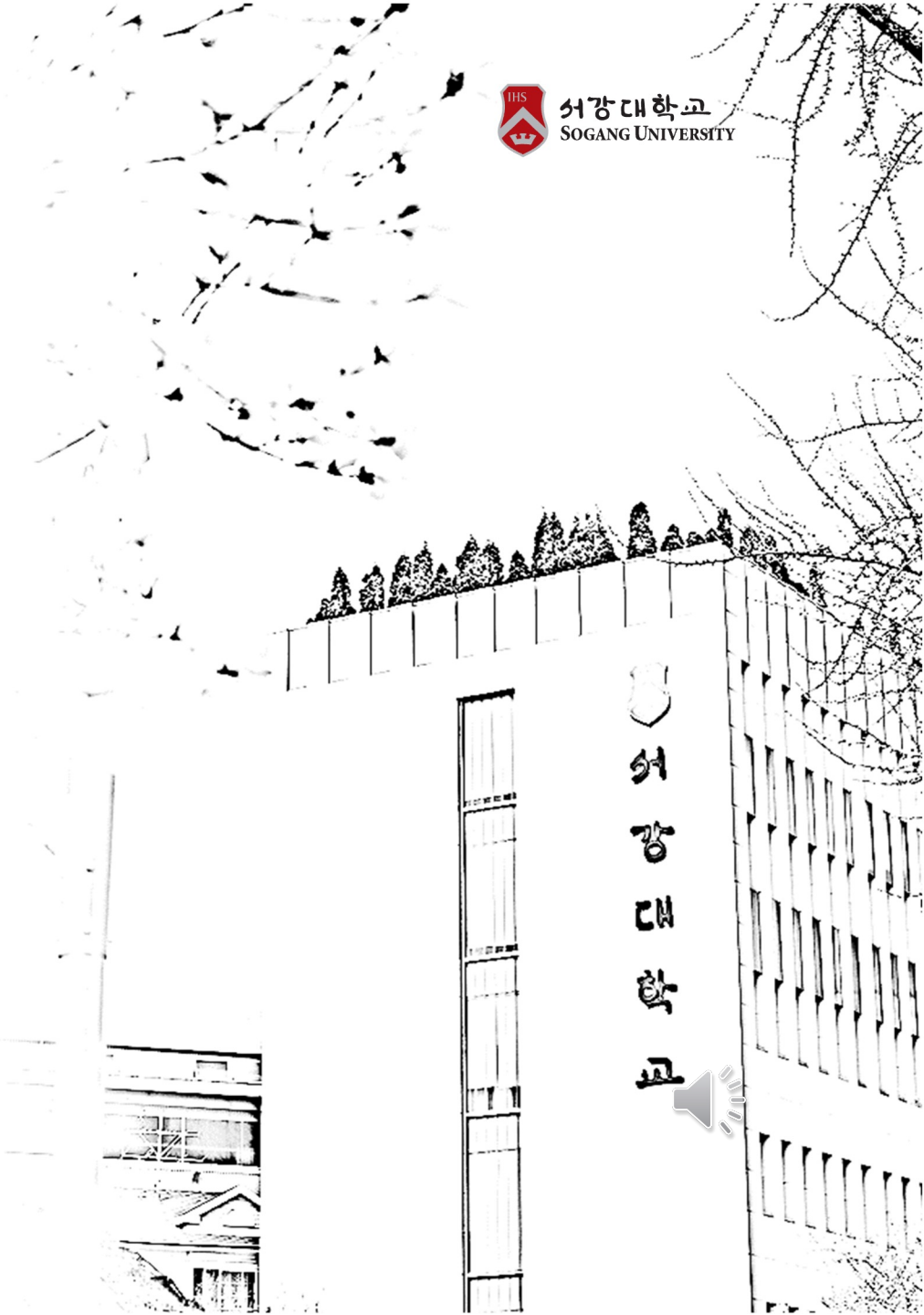
탐색의 과정을 시각화 하여 BFS와 DFS알고리즘에 대한 이해를 돕는 것이 이 프로젝트의 목표이다.



SOGANG UNIVERSITY

02. 구현 과정

DFS, BFS 알고리즘의 시각화



서강대학교 SOGANG UNIVERSITY

DFS

DFS와 BFS 모두 한번 방문한 방은 다시 방문하지 않는다.
하지만 탐색 과정을 시각화 하기 위해서는 되돌아 가는 과정을 추가할 필요가 있다.
방문하는 순서를 저장하였다.

```
bool ofApp::DFS(int x, int y)//DFS탐색을 하는 함수
{
    //path 변수는 dfs에서 이동한 점들의 위치를 저장하는 변수이다.
    path.push_back(make_pair(x, y));
    if (x == WIDTH - 2 && y == HEIGHT - 2) {
        maze_arr[y][x] = 2;
        return true;
    }
    else
        for (int i = 0;i < 4;i++)
            if (x + dx[i] * 2 > 0 && x + dx[i] * 2 < WIDTH && y + dy[i] * 2 > 0 && y + dy[i] * 2 < HEIGHT && maze_arr[y + dy[i]][x + dx[i]] == 0)
                if (maze_arr[y + 2 * dy[i]][x + 2 * dx[i]] == 0) {
                    maze_arr[y][x] = 3;;
                    maze_arr[y + 2 * dy[i]][x + 2 * dx[i]] = 3;
                    if (DFS(x + 2 * dx[i], y + 2 * dy[i])) {
                        maze_arr[y][x] = 2;
                        return true;
                    }
                    path.push_back(make_pair(x, y));
                    maze_arr[y][x] = 3;
                }
    maze_arr[y][x] = 3;
    return false;
}
```

서강대학교 SOGANG UNIVERSITY

DFS

시각화 하는 과정에서 각 프레임 path에 저장된 정보를 하나씩 출력해준다.

```
void ofApp::update() {
    if (isdfs && !dfs_draw_finish) {
        dfs_print_idx++; // DFS 의 경로를 한단계씩 더 그려준다.
    }
```

```
void ofApp::draw_DFS_PATH(void) { // path에 저장된 점을 기준으로 그려준다.
    if (dfs_print_idx >= path.size() - 1) {
        dfs_draw_finish = true;
        return;
    }
    for (int i = 0;i < dfs_print_idx;i++) { //path의 점을 이어서 그려준다.
        int x1 = path[i].first;
        int y1 = path[i].second;
        int x2 = path[i + 1].first;
        int y2 = path[i + 1].second;
        ofSetLineWidth(5);
        ofSetColor(200);
        ofDrawLine(x1 * (LINEH + LINEW) / 2 + (LINEW / 2), y1 * (LINEH + LINEW) / 2 + (LINEW / 2), x2 * (LINEH + LINEW) / 2 + (LINEW / 2), y2 * (LINEH + LINEW) / 2 + (LINEW / 2));
    }
    ofSetColor(100);
    ofDrawCircle(path[dfs_print_idx].first * (LINEH + LINEW) / 2 + (LINEW / 2), path[dfs_print_idx].second*(LINEH + LINEW) / 2 + (LINEW / 2), LINEH/4);
}
```

서강대학교 SOGANG UNIVERSITY

DFS

path의 모든 점을 출력하였다면 올바른 길을 다른 색으로 출력해준다.

```
if (isdfs)
{
    ofSetColor(200);
    ofSetLineWidth(5);
    if (isOpen) {
        if (!dfs_draw_finish)  //dfs가 완료되지 않았다면 중간 과정 까지 그려준다.
            draw_DFS_PATH();
        else
            dfsdraw(); // dfs가 완료되었다면 전체를 그려준다.
    }
    else
        cout << "You must open file first" << endl;
}
```

서강대학교 SOGANG UNIVERSITY

BFS

BFS의 경우 DFS와 다르게 다음 번 방문하는 방이 인접한 방이 아니다.
이를 DFS처럼 돌아가는 경로를 표시하는 방식 구현하는 것은 적합하지 않다.
BFS탐색 중 queue 에 들어가는 지점을 시각화 하여 해결하였다.

```
if (isbfs) {
    ofSetColor(200); ofSetLineWidth(5);
    if (isOpen) {
        bfsdraw(); // 현재까지 bfs경로를 그려준다.
        if (!bfs_draw_finish) {
            // 현재 탐색중인 깊이에 있는 모든 점을 표한하기 위해
            // 첫 원소(지금 움직이는 원소)는 초록색으로 그려준다
            pair<int, int> temp = q_bfs.front(); q_bfs.pop();
            ofSetColor(0, 255, 0);
            ofDrawCircle(temp.first * (LINEH + LINEW) / 2 + (LINEW / 2), temp.second * (LINEH + LINEW) / 2 + (LINEW / 2), LINEH / 4);
            q_bfs.push(temp);
            //나머지 원소들은 하나씩 빼서 회색으로 그려주고 다시 큐에 넣어준다.
            for (int i = 1;i < q_bfs.size();i++) {
                temp = q_bfs.front(); q_bfs.pop();
                ofSetColor(100);
                ofDrawCircle(temp.first * (LINEH + LINEW) / 2 + (LINEW / 2), temp.second * (LINEH + LINEW) / 2 + (LINEW / 2), LINEH/4);
                q_bfs.push(temp);
            }
            ofSetColor(100);
        }
    }
    else
        cout << "You must open file first" << endl;
}
```

서강대학교 SOGANG UNIVERSITY

BFS

DFS는 모든 탐색을 진행 한 후 경로를 저장하고 출력함과 다르게 BFS에서는 한 프레임에 한번의 탐색을 진행한다.
queue에서 제거된 값을 기록하는 방법보다 효율적이기 때문이다.
ofApp::update() 함수에 선언된 내용이다.

```
if (isbfs && !bfs_draw_finish) //BFS의 단계는 DFS와는 다르게 update함수를 통해 진행한다.
    if (!q_bfs.empty()) {
        pair<int, int> curr = q_bfs.front(); q_bfs.pop();
        int x = curr.first, y = curr.second;
        if (x == WIDTH - 2 && y == HEIGHT - 2) {
            while (!q_bfs.empty()) q_bfs.pop(); //BFS가 종료된다면 queue에 있는 모든 원소를 빼준다.
            bfs_draw_finish = true;
        }
        else
            for (int i = 0; i < 4; i++) {
                if (y + 2 * dy[i] < 0 || x + 2 * dx[i] < 0) continue;
                if (maze_arr[y + dy[i]][x + dx[i]] == 1) continue;
                if (maze_arr[y + 2 * dy[i]][x + 2 * dx[i]] != 0) continue;
                maze_arr[y + 2 * dy[i]][x + 2 * dx[i]] = maze_arr[y][x] + 1;
                q_bfs.push(make_pair(x + 2 * dx[i], y + 2 * dy[i]));
            }
    }
```

서강대학교 SOGANG UNIVERSITY

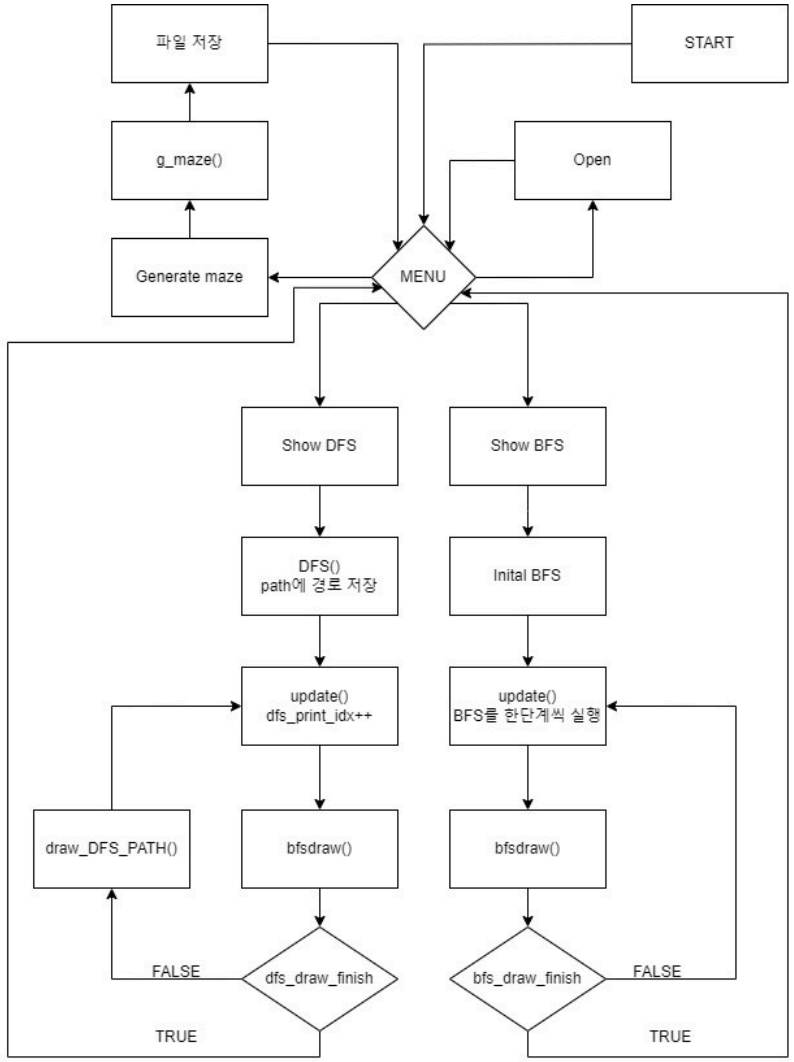
Maze Generator

사용자에게 미로의 크기 W, H를 입력 받아 recursive backtracking을 통하여 미로를 생성한다.
생성된 미로는 /bin/data/*.maz 로 저장되어 프로그램을 다시 시작해도 불러올 수 있도록 한다.
다음은 미로를 생성하는 코드이다.

```
void ofApp::g_maze(int x, int y) {
    int d = rand() % 4; //랜덤한 방향을 지정해주고 그 방향의 벽을 허물어준다.
    for (int i = 0;i < 4;i++) {
        int tx = x + dx[(i + d) % 4] * 2;
        int ty = y + dy[(i + d) % 4] * 2;
        if (tx > 0 && tx < WIDTH && ty>0 && ty < HEIGHT && new_maze[ty][tx] == 1) {
            new_maze[ty][tx] = 0;
            new_maze[y + dy[(i + d) % 4]][x + dx[(i + d) % 4]] = 0;
            g_maze(tx, ty); // 재귀적으로 이를 반복한다.
        }
    }
}
```

서강대학교 SOGANG UNIVERSITY

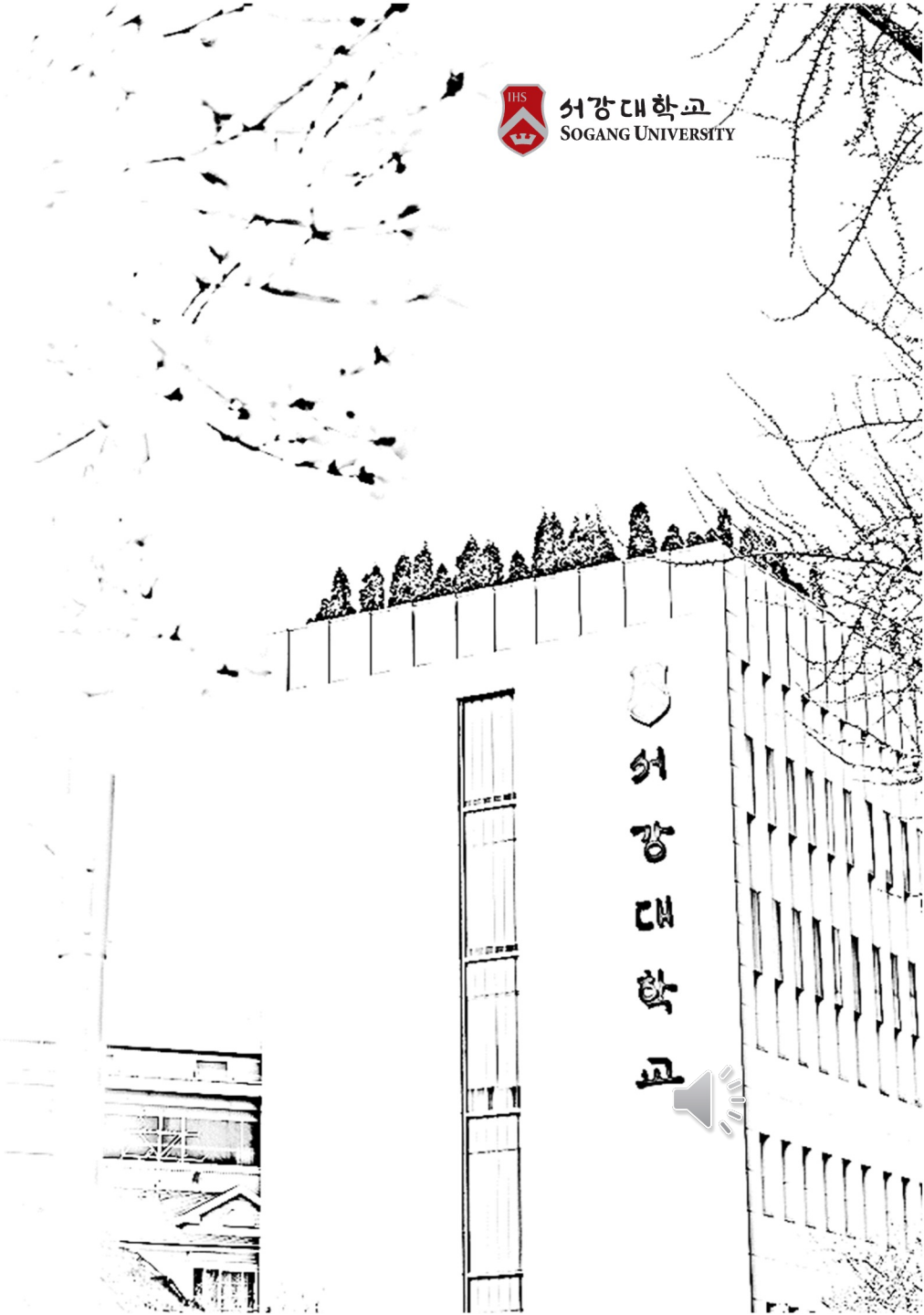
Flow Chart

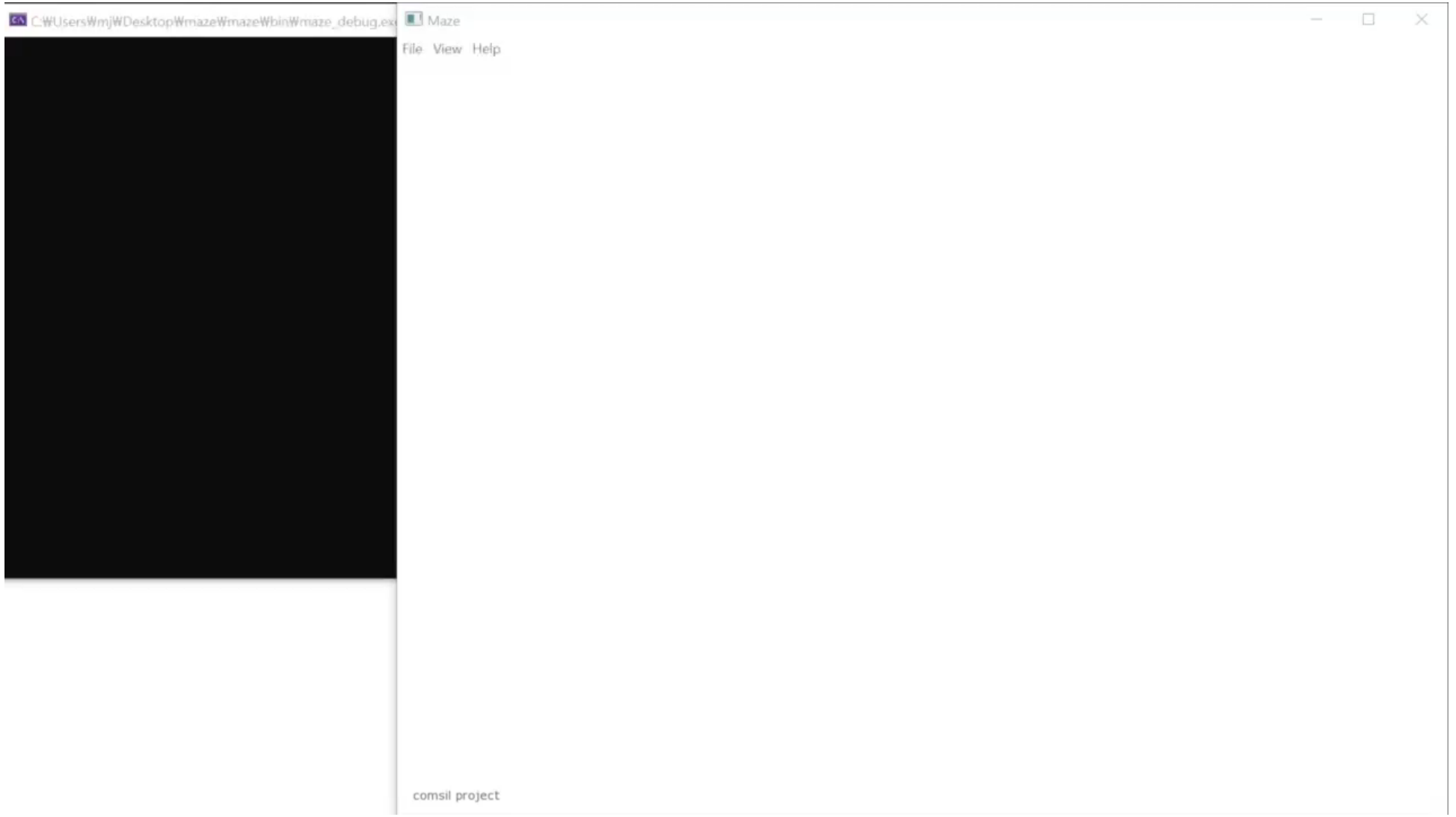


SOGANG UNIVERSITY

03. 시연 영상

DFS, BFS 알고리즘의 시각화




C:₩Users₩mj₩Desktop₩maze₩maze₩bin₩maze_debug.ex
Maze
File View Help
comsil project

SOGANG UNIVERSITY

04. 느낀 점 및 개선사항

DFS, BFS 알고리즘의 시각화

서강대학교
SOGANG UNIVERSITY

서강대학교 SOGANG UNIVERSITY

미로 생성 알고리즘의 시각화

Recursive backtracking algorithm 과 openframeworks의 특성상 미로가 생성되는 과정을 구현하기 어렵다.
다른 미로 생성 알고리즘을 사용한다면 이 또한 시각화 할 수 있을 것 같다.
이번 프로젝트를 통해 DFS, BFS에 대한 이해가 한층 더 높아진것 같다.

서강대학교 SOGANG UNIVERSITY

Thank you!

김민준 / 20201559