**Waterfall-1 결과보고서**

전공: 수학 학년: 4 학번: 20161255 이름: 장원태

**OpenFramework 실습 코드들을 수행하고, 각 line이 어떤 기능을 수행하는지 작성한다.**

이번 실습에서 작성한 코드들은 다음과 같다.

1. 파일에서 데이터 읽어오기

void ofApp::processOpenFileSelection(ofFileDialogResult openFileResult) {

//Path to the comma delimited file

//string fileName = "input.txt";

string fileName = openFileResult.getName();

ofFile file(fileName);

if( !file.exists()) cout << "Target file does not exists." << endl;

else cout << "We found the target file." << endl;

ofBuffer buffer(file);

/\* This variable is for indicating which type of input is being received.

IF input\_type == 0, then work of getting line input is in progress.

IF input\_type == 1, then work of getting dot input is in progress.

\*/

int input\_type = 0;

/\* COMSIL1-TODO 1 : Below code is for getting the number of line and dot, getting coordinates.

You must maintain those information. But, currently below code is not complete.

Also, note that all of coordinate should not be out of screen size.

However, all of coordinate do not always turn out to be the case.

So, You have to develop some error handling code that can detect whether coordinate is out of screen size.

\*/

int dotindex = 0; int lineindex = 0;

// 점과 선분의 정보를 배열에 저장하기 위해 index를 선언한다.

// Read file line by line

for (ofBuffer::Line it = buffer.getLines().begin(), end = buffer.getLines().end(); it != end; ++it) {

// 각 줄을 읽어온다.

string line = \*it;

// Split line into strings 그 줄을 스페이스바를 기준으로 나눈다.

vector<string> words = ofSplitString(line, " ");

if( words.size() == 1){ // 점이나 선분의 개수에 대한 정보를 저장하는 경우이다.

if( input\_type == 0){ // Input for the number of lines.

num\_of\_line = atoi(words[0].c\_str()); // 선분에 대한 경우이므로, 선분의 개수에 해당 데이터를 저장한다.

cout << "The number of line is: " << num\_of\_line << endl; // 콘솔에 선분의 개수를 출력해준다.

line\_info = new waterline[num\_of\_line]; // waterline이라는 배열을, 크기는 선분의 개수가 되도록 새롭게 할당한다.

}

else{ // Input for the number of dots.

num\_of\_dot = atoi(words[0].c\_str()); // 점에 대한 경우이므로, 점의 개수에 해당 데이터를 저장한다.

cout << "The number of dot is: " << num\_of\_dot << endl; // 콘솔에 점의 개수를 출력해준다.

dot\_info = new waterdot[num\_of\_dot]; // waterdot이라는 배열을, 크기는 점의 개수가 되도록 새롭게 할당한다.

}

}

else if (words.size() >= 2){ // 점이나 선분에 대한 상세한 정보를 받은 경우이다.

int x1,y1,x2,y2; // 점의 좌표나, 선분의 양 끝 점 좌표를 저장하기 위한 변수를 선언한다.

if( input\_type == 0){ // Input for actual information of lines

x1 = atoi(words[0].c\_str()); // 선분의 한 쪽 끝 점의 x좌표를 저장한다.

y1 = atoi(words[1].c\_str()); // 선분의 해당 끝 점의 y좌표를 저장한다.

x2 = atoi(words[2].c\_str()); // 선분의 다른 쪽 끝 점의 x좌표를 저장한다.

y2 = atoi(words[3].c\_str()); // 선분의 해당 끝 점의 y좌표를 저장한다.

line\_info[lineindex].x1 = x1; // x1값을 lineindex번째의 line\_info에 저장한다.

line\_info[lineindex].y1 = y1; // y1값을 lineindex번째의 line\_info에 저장한다.

line\_info[lineindex].x2 = x2; // x2값을 lineindex번째의 line\_info에 저장한다.

line\_info[lineindex].y2 = y2; // y2값을 lineindex번째의 line\_info에 저장한다.

lineindex++; // 저장이 완료된 후, 다음 값을 받기 위해 lineindex에 1을 더해준다.

if (lineindex == num\_of\_line) input\_type = 1; // 모든 선분에 대해 저장이 완료되면, 이후에는 점에 대한 정보를 저장하면 되므로 input\_type을 그에 맞게 설정한다.

}

else{ // Input for actual information of dots.

x1 = atoi(words[0].c\_str()); // 점의 x좌표를 저장한다.

y1 = atoi(words[1].c\_str()); // 점의 y좌표를 저장한다.

dot\_info[dotindex].x1 = x1; // x1값을 dotindex값의 dot\_info에 저장한다.

dot\_info[dotindex].y1 = y1; // y1값을 dotindex값의 dot\_info에 저장한다.

dotindex++; // 저장이 완료된 후, 다음 값을 받기 위해 dotindex에 1을 더해준다.

}

} // End of else if.

} // End of for-loop (Read file line by line).

//initializeWaterLines();

}

1. ‘D’ 키를 누른 경우 그림을 그릴 수 있게 설정

if (key == 'd' || key == 'D'){

if( !load\_flag) return; // load가 되지 않은 상태라면 이 과정을 수행하지 않는다.

draw\_flag = 1; // 이후에 그릴 과정을 위한 flag을 1로 set한다.

/\* COMSIL1-TODO 2: This is draw control part.

You should draw only after when the key 'd' has been pressed.

\*/

}

1. 그림을 그릴 수 있게 설정된 경우, 선분과 점을 화면에 그리기

void ofApp::draw(){

ofSetColor(127,23,31); // Set the drawing color to brown

int i = 0;

// Draw shapes for ceiling and floor

ofDrawRectangle(0, 0, 1024, 40); // Top left corner at (50, 50), 100 wide x 100 high

// 위쪽의 갈색 선을 그린다.

ofDrawRectangle(0, 728, 1024, 40); // Top left corner at (50, 50), 100 wide x 100 high

// 아래쪽의 갈색 선을 그린다.

ofSetLineWidth(5);

ofSetLineWidth(5);

if( draw\_flag ){ // draw\_flag가 ‘D’ 키에 의해 1로 set되었을 때 그린다.

/\* COMSIL1-TODO 3 : Draw the line segment and dot in which water starts to flow in the screen.

Note that after drawing line segment and dot, you have to make selected water start dot colored in red.

\*/

for (i = 0; i < num\_of\_line; i++) // 모든 선에 대한 for문이다.

ofDrawLine(line\_info[i].x1, line\_info[i].y1, line\_info[i].x2, line\_info[i].y2); // 각 line\_info에 저장된 데이터를 토대로 선을 그린다.

for (i = 0; i < num\_of\_dot; i++) { // 모든 점에 대한 for문이다.

if (i == selected\_dot) ofSetColor(255, 0, 0); // 현재 선택된 점인 selected\_dot 값과 현재 dot\_info에서의 index가 같다면, 해당 점에서는 빨간색으로 색을 설정한다.

else ofSetColor(0, 0, 0); // 현재 선택된 점이 아닌 다른 점이라면, 해당 점에서는 검은색으로 색을 설정한다.

ofDrawCircle(dot\_info[i].x1, dot\_info[i].y1, dot\_diameter); // 각 dot\_info에 저장된 데이터를 받아 원의 중심을 설정하고, dot\_diameter만큼의 지름을 가진 원으로 점을 표시하도록 그린다.

}

// 2nd week portion.

ofSetLineWidth(2);

}

}

1. 그림을 그린 상태에서, 방향키를 이용하여 점 선택하기(선택된 점 표시하기)

void ofApp::keyReleased(int key){

if( key == 'l'){

// Open the Open File Dialog

ofFileDialogResult openFileResult= ofSystemLoadDialog("Select a only txt for Waterfall");

// Check whether the user opened a file

if( openFileResult.bSuccess){

ofLogVerbose("User selected a file");

// We have a file, so let's check it and process it

processOpenFileSelection(openFileResult);

load\_flag = 1;

}

}

/\* COMSIL1-TODO 4: This is selection dot control part.

You can select dot in which water starts to flow by left, right direction key (<- , ->).

\*/

if (key == OF\_KEY\_RIGHT){ // 오른쪽 방향키를 누른 경우

selected\_dot = (selected\_dot + 1) % num\_of\_dot; // selected\_dot 인덱스값을 1 올려준다. 그 값이 점의 개수와 같아지는 경우, 즉 맨 마지막 점에서 오른쪽 점으로 이동하려고 시도하는 경우는 인덱스를 0으로 만들어준다.

cout << "Selected Dot Coordinate is (" << dot\_info[selected\_dot].x1 << ", " << dot\_info[selected\_dot].y1 << ")" << endl; // 선택된 인덱스에 대한 dot\_info 데이터를 출력한다.

}

if (key == OF\_KEY\_LEFT){ // 왼쪽 방향키를 누른 경우

if (selected\_dot == 0) selected\_dot = num\_of\_dot - 1; // 가장 첫 점에서 왼쪽 점으로 이동하려고 시도하는 경우는 맨 마지막 점으로 이동해야 하므로, 인덱스를 점의 개수에서 1을 뺀 값으로 설정한다.

else selected\_dot--; // 이외의 경우는 selected\_dot 인덱스값을 1 내려준다.

cout << "Selected Dot Coordinate is (" << dot\_info[selected\_dot].x1 << ", " << dot\_info[selected\_dot].y1 << ")" << endl; // 선택된 인덱스에 대한 dot\_info 데이터를 출력한다.

}

}

1. ‘Q’ 키를 누른 경우, 메모리를 모두 해제하고 종료

keyPressed 메소드의 일부이다.

if (key == 'q' || key == 'Q'){

// Reset flags

draw\_flag = 0; // 다시 draw가 되지 않도록 해야하므로, draw\_flag를 0으로 초기화해준다.

// Free the dynamically allocated memory exits.

delete[] dot\_info; // 할당했던 dot\_info 배열을 다시 반환한다.

delete[] line\_info; // 할당했던 line\_info 배열을 다시 반환한다.

cout << "Dynamically allocated memory has been freed." << endl; // 반환이 완료됨을 확인하기 위해 출력한다.

\_Exit(0); // 프로그램을 종료한다.

}