7주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20221532 이름: 김민기

1. Water\_Line Calculating 함수

**void** WaterLine::calculate\_path(LineSegment \*lineseg, **int** num\_of\_line) {

path[path\_idx].x1 = start\_dot.x;

path[path\_idx].y1 = start\_dot.y;

path\_idx++;

**for**( ; start\_dot.y <=ofGetHeight()-50 ; start\_dot.y++){

**for**( **int** i=0 ; i<num\_of\_line ; i++){

// 1) Ignore line that located higher than water particle.

**if**( start\_dot.y >= lineseg[i].y1 && start\_dot.y >= lineseg[i].y2 ) **continue**;

// 2) Check whether the dot point exists between the each end of line segment.

// If pos.x exists btw. y1 and y2 then, eventually hit the line segment.q

**if**( lineseg[i].x1 < lineseg[i].x2){

**if**( start\_dot.x <= lineseg[i].x1 || start\_dot.x >= lineseg[i].x2)

**continue**;

}

**else** **if** ( lineseg[i].x1 > lineseg[i].x2){

**if**( start\_dot.x <= lineseg[i].x2 || start\_dot.x >= lineseg[i].x1)

**continue**;

}

// Slope 계산

**double** temp\_slope = (**double**)(start\_dot.y - lineseg[i].y1)/(start\_dot.x - lineseg[i].x1);

// Dot exists in line

**if**( abs(temp\_slope - lineseg[i].slope) <= EPSILON){

path[path\_idx].x1 = start\_dot.x;

path[path\_idx].y1 = start\_dot.y+2;

path\_idx++;

// Debug output

//cout << "[" << i << "]" << "x: " << start\_dot.x << " " << "y: " << start\_dot.y << endl;

**if**( lineseg[i].slope < 0){

path[path\_idx-1].x1++;

start\_dot.x = lineseg[i].x1;

start\_dot.y = lineseg[i].y1-2;

}

**else**{

path[path\_idx-1].x1--;

start\_dot.x = lineseg[i].x2;

start\_dot.y = lineseg[i].y2-2;

}

path[path\_idx].x1 = start\_dot.x;

path[path\_idx].y1 = start\_dot.y;

path\_idx++;

// Debug output

cout << "[" << i << "]" << "x: " << start\_dot.x << " " << "y: " << start\_dot.y << endl;

}

}

}

// Last path

path[path\_idx].x1 = start\_dot.x;

path[path\_idx].y1 = start\_dot.y;

path\_idx++;

calc\_complete = 1;

}

Water Line을 계산하는 함수이다. 물이 지나갈 선은 path배열에 저장된 점들을 이어 그려지므로 이 함수에선 path값을 계산하여 저장한다. 현재 계산하는 점은 start\_dot 변수를 사용하며 모든 선분들을 계산한다. 해당 선분이 위에 있거나, start\_dot과 겹치지 않는다면 해당 선분은 계산하지 않는다. 먼저 만약 점이 선분위에 존재한다면 선분을 따라서 물이 흐르므로 선분의 기울기가 음수인지 양수인지에 따라 각각 path를 저장한다. 모든 path를 저장했다면 플래그를 1로 설정하여 draw를 실행한다.

Diagram

Description automatically generated플로우 차트:

2. Draw 함수

**void** WaterLine::draw() {

**if**( calc\_complete){

ofSetLineWidth(5);

ofSetColor(uniqueColor\_r, uniqueColor\_g, uniqueColor\_b);

**for**( **int** i=0 ; i<path\_idx-1 ; i++){

uniqueColor\_r = ofRandom(0,100);

uniqueColor\_g = ofRandom(0,100);

uniqueColor\_b = ofRandom(185, 255);

ofSetColor(uniqueColor\_r, uniqueColor\_g, uniqueColor\_b);

ofDrawLine(path[i].x1-1, path[i].y1-1, path[i+1].x1+1, path[i+1].y1+1);

}

}

draw\_complete = 1;

}

Calculate함수에서 저장된 path값을 모두 돌며 선을 그린다. 색은 파란색으로 설정한다.