7주차 결과보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20181688 이름: 조태연

if (key == 'd') {

if (!load\_flag) return;

draw\_flag = 1;

target\_dot.x = dot[selection\_dot].x1;

target\_dot.y = dot[selection\_dot].y1;

}

if (key == 's') {

if (load\_flag) {

resetWater();

for (unsigned int i = 0; i < wl.size(); i++) {

if (wl[i].calc\_complete == 0)

wl[i].calculate\_path(lineseg, num\_of\_line);

}

if (draw\_flag) {

waterfall\_start\_flag = 1;

}

}

}

if (key == 'e') {

if (draw\_flag)

waterfall\_start\_flag = 0;

}

}

* d,s,e키가 눌릴 때를 구현한 부분이다. D를 누르면 draw\_flag를 활성화 해주고 s를 누르면 waterfall\_start\_flag를 활성화 해준다.

void ofApp::draw() {

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

// Draw shapes for ceiling and floor

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

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

ofSetLineWidth(5);

if (draw\_flag) {

int i;

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

ofDrawLine(lineseg[i].x1, lineseg[i].y1, lineseg[i].x2, lineseg[i].y2);

}

for (i = 0; i < num\_of\_dot; i++) {

if (selection\_dot == i)

ofSetColor(255, 0, 0);

else

ofSetColor(0);

ofDrawCircle(dot[i].x1, dot[i].y1, dot\_diameter / 2);

}

}

ofSetLineWidth(2);

if (waterfall\_start\_flag) {

for (unsigned int i = 0; i < wl.size(); i++) {

if (wl[i].path == NULL) continue;

if (wl[i].calc\_complete == 0)

wl[i].calculate\_path(lineseg, num\_of\_line);

wl[i].draw();

}

}

}

* **waterfall\_start\_flag가 활성화 되면 w[i].calculate\_path를 이용해 경로를 계산하고 w[i].draw를 호출해 그림을 그린다.**

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;

}

* **경로를 계산하는 부분이다. 물이 흐르게 된다면 꺾이거나 선분과 만나는 점들을 path에 저장한다.**

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;

}

* **그림을 그리는 부분이다 path에 저장된 점들을 순서대로 연결하여 그려준다**