7주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231632 이름: Jumagul Alua

이번 실습에는 waterfall 프로그램에서 물이 흐르도록 하는 일을 해야 됐다. 저번 실습에 함수를 두개를 추가해서 프로그램이 작동하게 만들었지만 물이 흐리지 않았다. 그래서 waterline 클래스에서의 Dot* path 부분을 초기화시켰다.

Dot 구조체: 특정 좌표를 나타내는 데 사용된다. 각 점은 흐르는 물의 경로를 정의하는 데 사용된다. LineSegment 구조체는 선분을 나타내는 데 사용된다. 각 선분은 물의 흐름 경로에 영향을 준다. path 포인터는 동적으로 할당된 메모리를 가리키는 포인터이다. 이 포인터는 물의 경로를 저장한다.

Calculate_path() 함수는 물 흐름의 경로를 계산하고, update() 함수는 시뮬레이션 상태를 업데이트하고, draw()는 시뮬레이션을 그립니다. 따라서 선형 세그먼트가 주어지면 이 코드를 사용하여 물 흐름을 시뮬레이션한 것이다.

일단, draw() 함수를 살펴봅시다:

```
void WaterLine::draw() {
    if (calc_complete) { //계산이 완료되었는지 확인함
        ofSetLineWidth(5); //선의 너비를 설정함

        ofSetColor(uniqueColor_r, uniqueColor_g, uniqueColor_b); //색상을 설정함
        for (int i = 0; i < path_idx - 1; i++) { //모든 경로 점을 연결하여 선을 그린다
            uniqueColor_r = ofRandom(0, 100);
            uniqueColor_g = ofRandom(0, 100);
            uniqueColor_b = ofRandom(185, 255);//무작위 색상을 생성함
            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); //선을 그린다
        }
    }
    draw_complete = 1; //완료됨
}
```

계산이 완료되었을 때만 경로를 그리고, 랜덤한 색상을 생성하여 경로를 그린다. ofSetColor() 함수로 색상을 설정하고, ofDrawLine() 함수로 선을 그린다.

그 다음, calculate_path() 함수:

```
void WaterLine::calculate_path(LineSegment* lineseg, int num_of_line) {
    //경로의 첫 번째 점을 설정함
    path[path_idx].x1 = start_dot.x;
    path[path_idx].y1 = start_dot.y;
    path_idx++;

    //화면 하단까지 각 점마다 경로를 계산함
    for (; start_dot.y <= ofGetHeight() - 50; start_dot.y++) {
        for (int i = 0; i < num_of_line; i++) {
            //1) 물 흐름보다 위에 있는 라인은 무시함
            if (start_dot.y >= lineseg[i].y1 && start_dot.y >= lineseg[i].y2) continue;
            //2) 라인 세그먼트의 양 끝 사이에 점이 있는지 확인함
            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;
            }
            //라인에 점이 존재하는 경우, 새로운 점을 경로에 추가함
            double temp_slope = (double)(start_dot.y - lineseg[i].y1) / (start_dot.x - lineseg[i].x1);
            if (abs(temp_slope - lineseg[i].slope) <= EPSILON) {
                //경로에 점을 추가함
                path[path_idx].x1 = start_dot.x;
                path[path_idx].y1 = start_dot.y + 2;
                path_idx++;
```

```
            double temp_slope = (double)(start_dot.y - lineseg[i].y1) / (start_dot.x - lineseg[i].x1);
            if (abs(temp_slope - lineseg[i].slope) <= EPSILON) {
                //경로에 점을 추가함
                path[path_idx].x1 = start_dot.x;
                path[path_idx].y1 = start_dot.y + 2;
                path_idx++;
                //다음 위치로 이동함
                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;
                }
                //경로에 점을 추가함
                path[path_idx].x1 = start_dot.x;
                path[path_idx].y1 = start_dot.y;
                path_idx++;
            }
        }
    }
    //마지막 경로를 추가함
    path[path_idx].x1 = start_dot.x;
    path[path_idx].y1 = start_dot.y;
    path_idx++;

    calc_complete = 1; //완료됨
```

start_dot에서 시작하여 경로를 계산하고 각 점마다 주어진 라인 세그먼트들과의 상호작용을 고려하여 새로운 점을 경로에 추가했다. 경로 계산이 완료되면 calc_complete 변수를 설정하여 이를 표시한다.