7주차 결과보고서

전공: 신문방송학과 학년: 3학년 학번: 20191150 이름: 전현길

1. 물이 흐르는 것을 표현하기 위해서, 본인이 구현한 알고리즘과 자료구조를 기술한다.

가. ofApp의 멤버 변수(ofApp.h)

a. **LineSegment* lineseg**: 직선의 정보를 저장한다.

b. **Dot* dot**: 점의 정보를 저장한다.

c. **int num_of_line**: 직선의 개수를 저장한다.

d. **int num_of_dot**: 점의 개수를 저장한다.

e. **int draw_flag**: 화면에 점, 선을 그릴 때 set되는 flag

f. **int load_flag**: 입력 파일을 성공적으로 불러왔을 때 set되는 flag

g. **int waterfall_start_flag**: 물줄기를 흐르게 할 때 set되는 flag

h. **int selection_dot**: 현재 선택된 점을 가리키는 idx

i. **ofPoint target_dot**: 현재 선택된 점에 관한 정보를 ofPoint 변수를 저장한다.

j. **water_radius**: 물줄기를 표현하는 반지름을 저장한다.

k. **dot_diameter**: 점의 지름을 저장한다.

l. **vector <WaterLine> wl**: 각 점(=수구)에 대한 물줄기의 정보를 저장하는 1차원 vector이다.

나. WaterLine.h의 클래스

a. **LineSegment**: 직선의 클래스, 두 끝 점의 x, y좌표(x1/y1, x2/y2), 기울기(slope), (-1) * x의 이동 거리(y_coef), y의 이동거리(x_coef), 상수 값(constant)을 멤버 변수로 저장한다.

b. **Dot**: 점의 클래스, x, y좌표(x1/y1)를 멤버 변수로 저장한다.

c. **WaterLine**

[변수]

1. **int path_idx**: 현재 물줄기의 경로에 대한 idx

2. **int draw_complete**: path를 화면에 표시할 때 set되는 flag

3. **int calc_complete**: path의 계산이 완료되었을 때 set되는 flag

4. **ofPoint start_dot**: 물의 흐름이 시작되는 점 또는 물줄기의 현재 위치를 실시간으로 추적하는 변수(calculate_path() 함수)

5. **float dot_radius**: 점의 크기를 저장하는 변수

6. **float uniqueColor_r, uniqueColor_g, uniqueColor_b**: 물줄기의 고유한 색을 저장하는 변수

7. **Dot* path**: 물줄기가 흐를 경로를 저장하는 Dot 변수의 배열, 각각 x좌표와 y좌표를 저장하고 있으며, path 경로에 저장된 선을 순서대로 이어서 물줄기의 흐름을 표현할 수 있음

8. **int scale, int hexcolor**: 코드에서 실제로 사용되지 않는 변수

[함수]

1. **WaterLine(int num_of_line)**

직선의 개수를 입력값으로 받아 각종 flag, 색을 초기화하고, path 변수에 대해 메모리를 직선의 개수와 점의 수에 따라 동적으로 할당한다.

2. **~WaterLine()**

동적 할당한 메모리를 해제한다

3. **void reset()**

물줄기의 색을 초기화하고, calc_complete(경로 계산 완료), draw_complete(물줄기 그리기 완료) flag와 path_idx를 reset한다.

직선의 배열 LineSegment* lineseg와 직선의 개수 num_of_line을 입력값으로 받아 경로를 계산하는 calculate_path(), 물줄기를 실제로 그리는 draw()를 갖는다.

4. **void draw()**

calc_complete이 set되어 있다면 path 변수에 저장된 점들을 설정된 고유색에 따라 연결하는 직선을 순서대로 그린다. 완성된 선을 통해 물줄기가 표현되며, 그리기가 완료되면 draw_complete flag를 set한다.

5. **void calculate_path(LineSegment* lineseg, int num_of_line)**

선의 정보를 담은 배열 lineseg와 선의 개수를 입력값으로 받아 한 점에 대한 물줄기의 경로를 계산한다. 경로를 계산하는 알고리즘은 아래에 line-by-line으로 주석을 달아 설명했다.

```
void WaterLine::calculate_path(LineSegment *lineseg, int num_of_line) {
    // 각 점에 따라 물줄기의 경로를 계산
    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++){
            // 점의 시작 좌표보다 위에 있는 선분을 skip
            if( start_dot.y >= lineseg[i].y1 && start_dot.y >= lineseg[i].y2 ) continue;

            // 직선의 x좌표가 물줄기의 x좌표보다 왼쪽/오른쪽에 있다면 skip
            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;
            }
            // 현재 점(=물줄기)에서 선의 한 끝점에 대한 기울기가
            // 선의 기울기와 같아질 때까지 start_dot.y를 증가시키며 이동
            // 선의 기울때의 절댓값이 EPSILON보다 작을 때 path값 갱신
            double temp_slope = (double)(start_dot.y - lineseg[i].y1)/(start_dot.x - lineseg[i].x1);
            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++; // 물줄기와 선의 충돌지점의 좌표를 저장

                // 기울기가 음수라면 선의 오른쪽 끝점을 다음 경로로 저장
                if( lineseg[i].slope < 0){
                    path[path_idx-1].x1++;
                    start_dot.x = lineseg[i].x1;
                    start_dot.y = lineseg[i].y1-2;
                }
                // 기울기가 양수라면 선의 왼쪽 끝점을 다음 경로로 저장
                else{
                    path[path_idx-1].x1--;
                    start_dot.x = lineseg[i].x2;
                    start_dot.y = lineseg[i].y2-2;
                }
            }
        }
    }
    // 마지막 경로를 저장
    path[path_idx].x1 = start_dot.x;
    path[path_idx].y1 = start_dot.y;
    path_idx++;

    calc_complete = 1; // calc_complete flag를 set
}
```

1) 물줄기의 시작점을 path의 첫 경로로 설정한다.

2) 점의 시작 좌표의 y좌표 값보다 위에 있는 선분을 skip한다.

3) 직선이 물줄기의 현재 x좌표보다 왼쪽에 있거나, 오른쪽에 있어 닿지 않을 경우 skip한다.

4) 현재 점에서 선의 한 끝점에 대한 기울기가 선의 기울기와 거의 같아질 때까지(temp_slope)의 절댓값이 EPSILON보다 작아질 때까지 start_dot.y를 증가시키며 이동한다.

5) 물줄기와 선의 충돌지점의 x, y좌표를 path에 저장한다.

6) 선의 기울기가 음수라면 선의 오른쪽 끝점을, 양수라면 왼쪽 끝점을 path의 다음 경로에 저장한 뒤 path값을 갱신한다.

7) start_dot.y의 값이 ofGetHeight - 50보다 커질 때까지 반복한다.

8) 끝점의 경로를 path에 저장한 뒤, calc_complete flag를 set한다.

3. ofApp의 멤버 함수(ofApp.cpp)

a. void setup()

fps를 60fps로 고정(원본 코드의 경우 15fps)하며, 배경색을 흰색으로 set하고, 선의 두께를 4로 set하며, 각종 flag(draw_flag, load_flag, waterfall_start_flag, selection_dot)를 0으로 초기화한다.

b. void draw()

화면에 테두리, 점, 선, 물줄기를 그리는 함수이다.

먼저 화면의 위와 아래에 갈색 테두리를 그리고, 선의 두께를 5로 set하며, draw_flag가 set되어 있다면 점, 선을 개수만큼 그리며, waterfall_start_flag가 set되어있을 시 점에 따라 각각 물의 경로를 계산한 뒤 물줄기를 그린다. 현재 선택된 점의 경우 붉은색으로 표시하며, 그렇지 않을 경우 검은색으로 표시한다.

c. void keyPressed(int key)

사용자로부터 키보드로 입력을 받으면 이에 따라 명령을 수행하는 함수이다.

입력은 ‘q’, ‘d’, ‘v’, ‘s’, ‘e’로 구분된다.

‘v’를 입력받을 경우, 스크린샷을 찍은 뒤 .png 확장자로 저장한다.

‘q’를 입력받을 경우, load_flag가 set되어 있지 않다면 return하고, 그렇지 않을 경우 각종 flag를 초기화한 뒤 동적 할당된 메모리(lineseg, dot)를 해제한 뒤 프로그램을 종료한다.

‘d’를 입력받을 경우, load_flag가 set되어 있지 않다면 return하고, 그렇지 않을 경우 draw_flag를 set하고 target_dot의 x좌표, y좌표를 설정한다(=점과 선이 그려진다).

‘s’를 입력받을 경우, resetWater()함수로 물줄기 정보를 초기화한 뒤 경로를 다시 계산한다. draw_flag가 set되어 있을 경우 waterfall_start_flag를 set한다(=물줄기가 그려진다).

‘e’를 입력받을 경우, waterfall_start_flag를 reset한다(=물줄기가 멈춘다).

c. void keyReleased(int key)

사용자로부터 키보드로 입력을 받으면 이에 따라 명령을 수행하는 함수이다.

입력은 ‘l’, ‘←’, ‘→’로 구분된다.

‘l’을 입력받을 경우, 파일 입력창을 표시하며 파일 입력이 성공했을 경우 load_flag를 set한다.

‘←’를 입력받을 경우, selection_dot(=현재 선택된 점에 대한 idx)을 감소시키며 overflow될 시 예외 처리하여 마지막 점의 idx로 이동한다.

‘→’를 입력받을 경우, selection_dot(=현재 선택된 점에 대한 idx)을 증가시키며 overflow될 시 예외 처리하여 첫 번째 점의 idx로 이동한다.

d. void processOpenFIleSelection(ofFileDialogResult openFileResuilt)

파일의 이름을 fileName string 변수에 저장한 뒤, 파일이 존재하지 않을 시 오류 처리한다. 파일이 성공적으로 입력됐다면, input_flag와 idx를 0으로 초기화한다. input_flag는 현재 선에 대한 데이터를 입력받고 있는 중인지, 점에 대한 데이터를 입력받고 있는 중인지 결정한다.

it 변수를 입력 파일의 첫 번째 줄을 가리키는 변수로 초기화한 뒤, it 값을 증가시켜 it 변수가 마지막 줄에 도달할 때까지 아래의 작업을 반복한다.

line 변수에 it 변수가 가리키는 문자열(한 줄의 문자열)을 복사한 뒤, 공백

문자를 구분문자로 파싱하여 words 문자열 벡터에 저장한다. 만약 words의 원소가 1개라면 input_flag에 따라 num_of_line, num_of_dot에 값을 저장한다. 원소가 2개 이상이라면, input_flag에 따라 lineseg 배열에 차례대로 선의 정보(두 끝점의 좌표, 기울기, x_coef, y_coef, constant)를 저장하거나, dot 배열에 점의 정보를 저장한다.

점의 입력까지 종료되었다면 input_flag를 reset한다.

WaterLine 벡터 wl에 50개의 원소를 생선한 뒤, dot_diameter 변수를 20.0f로 저장한다. 시작점의 x, y좌표와 점의 반지름의 길이를 설정한 뒤 물줄기 관련 정보를 초기화한다.