**Waterfall-2 결과보고서**

전공: 수학 학년: 4 학번: 20161255 이름: 장원태

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

물이 흐르는 것을 표현할 때, 이번 실습에서는 ‘원’으로 물방울을 표현함을 통해 구현하였다. 우선, 별도의 boolean 변수로서 작용하는 waterfall 변수를 사용하여 물이 흘러야 하는지, 흐르지 않아야 하는지를 판단하게 한다. 그리고, 현재 물방울의 중심의 x, y좌표를 클래스 변수인 x, y에 저장하며, 해당 x, y를 올바르게 update시키며 물방울을 그려나가 물줄기가 표현되게 한다.

다음은 draw() 함수에서 추가로 구현한 코드이다. 해당 코드는 draw\_flag가 1로 set되어 있는 경우에 작동하게 된다.

|  |
| --- |
| if (waterfall) {  ofSetColor(0, 0, 255);  ofDrawCircle(x, y, 5.0f);    } |

해당 과정은 물줄기의 색깔을 ‘파란색’으로 set하고, x, y 좌표를 가지는 직경 5만큼의 물방울을 그리는 간단한 코드이다. 단, x와 y값이 update() 함수에 의해 변화되며 이에 따라 계속해서 물줄기가 그려지게 된다.

다음은 update() 함수에서 구현한 코드이다.

|  |
| --- |
| void ofApp::update(){  if (waterfall){  if (x == 0 && y == 0) {  x = dot\_info[selected\_dot].x1;  y = dot\_info[selected\_dot].y1;  }  else {  for (loop = 0; loop < num\_of\_line; loop++) {  if (line\_info[loop].y1 < y && line\_info[loop].y2 < y) continue;  if (min(line\_info[loop].x1, line\_info[loop].x2) > x || max(line\_info[loop].x1, line\_info[loop].x2) < x) continue;  slope = ((float)line\_info[loop].y1 - (float)line\_info[loop].y2) / ((float)line\_info[loop].x1 - (float)line\_info[loop].x2);  float on\_line\_y = ((float)x - (float)line\_info[loop].x1) \* slope + (float)line\_info[loop].y1;  if (abs((float)y - on\_line\_y) < 1.0f) break;  }  if (loop == num\_of\_line) y++;  else {  if (slope > 0.0f) x++;  else x--;  y = ((float)x - (float)line\_info[loop].x1) \* slope + (float)line\_info[loop].y1;  }  }  }  } |

이 코드는 기본적으로 update() 함수에 들어가 있으며, 물줄기가 생성되어야 하는 경우에만 작동해야 하므로, waterfall이 1로 set되어 있는 경우에만 update를 진행할 수 있도록 한다. 이는 후술할 ‘s’ 키를 누른 경우에만 x와 y 값들이 update되어야 한다는 이유와, 특별히 update를 시작할 시점을 잡아주지 않으면 코드 내에 있는 dot\_info와 line\_info를 비롯한 여러 변수들은 아직 null인 상태이므로 코드가 정상적으로 작동되지 않아, 윈도우 자체가 생성되지 않는 오류가 생기는 이유가 있다.

그래서, 물줄기가 구현되기 시작해야 할 시점은 waterfall이 0이 아닌 값으로 set되어 있을 때로, 그 때부터 update가 시작된다. 이 때, x와 y는 setup 함수에서 0으로 set되어 있다. 물줄기는 선택된 점에서부터 시작되어야 하므로, 0으로 set되어 있는 경우라면 선택된 점의 좌표로 각 x, y를 update해준다.

그 후, 각 물방울에 대해서 다음에 이동해야 할 좌표를 찾는 과정을 거친다. 이는 각 line들을 해당 물방울이 닿는지를 체크하는 것을 우선적으로 행한다. 이를 위해, 각 line들에 대해 search를 진행하는 for loop를 생성하며, 이는 line의 개수만큼 행해진다. 그 중, 만약 한 line의 양 끝 점 모두 현재 물방울의 중심보다 더 위에 있다면, 즉 y값이 더 작다면, 고려할 필요가 없으므로 continue를 통해 통과한다. 또한, 만약 물방울의 x좌표가 한 line의 x값 범위에 속하지 않는다면, 즉 한 line의 양 끝 점의 두 x좌표 중 작은 것보다 더 작거나, 큰 것보다 더 큰 경우 역시 line을 만나지 않으므로 고려할 필요가 없어, continue를 통해 통과한다. 이 일련의 과정을 거치면, 물방울과 만날 가능성이 있는 line들만 남는다. 이 때, 현재 물방울의 x좌표에 따른 해당 line 상의 점을 직선의 방정식을 통해 구한다. 기울기(slope)를 (y1-y2)/(x1-x2)를 통해 구하고, 현재 x좌표에 해당하는 직선상의 점의 y좌표는 y = m(x-x1)+y1(m은 기울기)라는 직선의 방정식에 대입하여 구한다. 이렇게 구한 직선상의 y좌표와 현재 물방울의 중심의 y좌표가 만나게 된다면 해당 for loop를 벗어난다. 이 때, 직선상의 점의 y좌표는 float형으로 구한 반면, 물방울의 중심의 y좌표는 int형으로 되어 있기 때문에, “==”을 통해 확인하지 않고, 그 차이의 절댓값이 1보다 작아지는 경우로 검사하게 된다.

이 과정을 거친 후 loop 변수인 loop값이 line의 개수와 같다면, for문을 모두 지날 때까지 위와 같이 두 y값이 일치하는 경우를 찾지 못한 것이므로, 그대로 아래로 떨어지게 하기 위해 y값을 1 더해준다. 그렇지 않다면, 현재 loop값이 곧 만나는 line의 index값이 되므로, 추가적인 과정을 수행한다. 만약 기울기가 양수라면, 윈도우 상으로는 직선이 오른쪽 아래로 기울어 있는 상태이므로 물줄기는 오른쪽으로 떨어져야 한다. 그러므로, x값을 1 더해준다. 그렇지 않다면(이 실습에서 기울기가 0인 직선은 고려하지 않지만, 행여 기울기가 0인 직선이 있다면 왼쪽으로 흐르게 해준다), 윈도우 상으로는 직선이 왼쪽 아래로 기울어 있는 상태이므로 물줄기는 왼쪽으로 떨어져야 한다. 그러므로, x값을 1 빼준다. 그 후, y값은 y = m(x-x1)+y1(m은 기울기) 직선의 방정식에 새로 구한 x값을 대입해줌을 통해, 즉 직선상에 있도록 구해준다.

다음으로 keyPressed(int key) 메소드에서, ‘q’를 누른 경우 waterfall 변수 역시 0으로 초기화해준다.

다음은 ‘s’ 키를 눌렀을 때와 ‘e’ 키를 눌렀을 때에 대한 코드이다.

|  |
| --- |
| if (key == 's' || key == 'S'){  // 2nd week portion.  ofSetBackgroundAuto(false);  waterfall = 1;    }  if (key == 'e' || key == 'E') {  // 2nd week portion.  ofSetBackgroundAuto(true);  waterfall = 0; x = 0; y = 0;  } |

‘S’ 키를 누른 경우, 위에서 설명한 물줄기 구현 방식을 따라 물줄기를 그리게 한다. 이 때, 물방울을 연속적으로 그려주어야 하므로, ofSetBackgroundAuto(false) 과정을 추가해주고, waterfall 변수를 1로 set해준다. ‘E’ 키를 누른 경우, 그렸던 물줄기를 모두 다시 지워줘야 하므로 ofSetBackgroundAuto(true) 과정을 추가해주고, waterfall 변수를 0으로 reset해준다. 이 때, update 함수 역시 초기화해주어야 이후에 물줄기를 다시 그릴 때 오류가 발생하지 않으므로, x와 y값도 setup 상태와 같이 0으로 초기화해준다.

이 실습에서 구현한 자료구조는 각 점과 선에 대한 정보를 저장할 구조체 및 해당 구조체를 가지는 배열이다. Struct 구조체를 사용하여, 점의 경우 해당 점의 x좌표와 y좌표를, 선의 경우는 해당 선의 양 끝 점의 x좌표와 y좌표를 저장하도록 했다. 그리고, 그 구조체의 포인터를 가지는 dot\_info와 line\_info 배열을 선언하여 각 정보를 저장하도록 했다.

|  |
| --- |
| typedef struct{  int x1;  int y1;  } waterdot;  typedef struct {  int x1;  int y1;  int x2;  int y2;  } waterline;  waterdot \*dot\_info;  waterline\* line\_info; |