실험 WaterFall-2주차 결과보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20201635 이름: 전찬

**1. 목적**

WaterFall 1주차의 구현에 이어서 물이 흐르는 형태를 구현하며, 구현하기 위해서 어떤 알고리즘과 자료구조를 사용했는지 설명한다.

**2. 구현한 알고리즘과 자료구조**

이번 실습에서는 WaterFall 1주차 실습에서 만들어낸 그림에서 각 점에서 물이 흐르기 시작할 때 어떤 형태로 흐르는지를 OpenFrameWorks를 통해서 표현하는 실습을 진행했다. 개인적으로 물 객체를 만들어내는 것보다 각 line에 대한 정보를 저장해서 ofDrawLine을 이용해서 구현하는 것이 쉬울 것 같아 각 line의 정보를 배열로 저장하는 형태를 선택했다.

우선 물 줄기의 각 선분에 대한 정보를 저장하는 배열을 동적 할당을 통해서 만들어주어야 했다. 여기에서는 이미 존재하는 직선과 현재 test하는 점의 위치와의 비교를 통해서 일정 조건에 만족하는 직선에 대해서, 그 직선과 test하는 점의 위치와의 교점~y값이 높은 직선의 끝점 + 끝점의 위치~y=728으로 수직으로 내려가는 위치 두 값을 저장해주었다. 간단히 말하자면, 저장되어 있는 직선의 정보(line\_data)의 모든 직선들에 대해서 비교를 진행하며, 가장 먼저 만나는 직선을 저장하는 것이다. 직선이 서로 교차하지 않는 가정 하에, 한 번 수직인 물줄기(직선과 만나지 않고 아래로 떨어지는 물줄기)가 있다면, 그 다음의 물줄기는 직선과 만나서 기울어진 형태일 것임을 토대로 위와 같이 만들어주었다. 쉽게 말하자면, 교점이 존재할 때마다, 직선 두개의 정보를 동적 할당으로 추가해 준 것이다. 위와 같은 방식으로 물줄기의 모든 직선을 동적 할당을 통해서 waterline\_data 에 저장해주었다. 또한 ‘s’ 키의 입력과 함께 water\_flag = 1로 만들어주어, 좌우 방향키를 제한하는 역할과 함께, draw()에서 그림을 그려낼 수 있도록 수행했다.

이후에 수행해야 할 일은 물줄기가 나오는 것을 표현해주는 것이었다. 이는 update()를 활용해서 수행해주었다. waterline\_counter라는 변수를 이용해주었는데, 0~waterline\_counter까지는 그대로 waterline\_data[i]에 대해서 ofDrawLine을 수행하고, 그 이후에 ofDrawLine을 한 번 더 수행해주는 형태로 진행했다. 여기에서 ofDrawLine(~~, ~~, lastx2, lasty2) 형태로, lastx1과 lastx2를 update() 에서 바꿔주며, lasty2가 waterline\_data[waterline\_counter][3] 보다 큰 경우(저장되어 있는 직선을 넘어가는 경우)에 lastx2와 lasty2를 적절하게 초기화해주며, waterline\_counter++; 형태로 진행했다. 마지막으로 lastx2 와 lasty2를 바꿔줄 때, slope를 바탕으로 총 길이가 1이 되는 형태로 이동시켜주었다. 또한 waterspeed라는 변수를 설정해 물의 속도를 조절할 수 있도록 만들었다.