컴퓨터공학실험1　10주차 결과보고서

전공: 컴퓨터공학과　　학년: 2　　학번: 20191559　　이름: 강상원

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

<ofApp.h>

```
#pragma once

#include "ofMain.h"
#include <vector>


struct WaterLine{
    int startX;
    int startY;
    int toX;
    int toY;

    float slope;
};

struct WaterPoint{
    bool active;
    int X;
    int Y;
};

struct WaterDrop{
    float xpos;
    float ypos;

    //double distance;
};

class ofApp : public ofBaseApp{
    private:
        int draw_flag;
        int flow_flag;
        int activeIdx=0;
        vector<WaterLine> lines;
        vector<WaterPoint> points;
        vector<WaterDrop> drops;

    public:
        void setup();
        void update();
        void draw();

        void keyPressed(int key);
        void keyReleased(int key);
        void mouseMoved(int x, int y );
        void mouseDragged(int x, int y, int button);
        void mousePressed(int x, int y, int button);
        void mouseReleased(int x, int y, int button);
        void mouseEntered(int x, int y);
        void mouseExited(int x, int y);
        void windowResized(int w, int h);
        void dragEvent(ofDragInfo dragInfo);
        void gotMessage(ofMessage msg);


        void processOpenFileSelection(ofFileDialogResult openFileResult);
        void UserLine(WaterLine line);
        void UserCircle(WaterPoint point);
        void drawDrop(WaterDrop water);

        void moveCal(WaterPoint activePoint, vector<WaterLine> lines);

};
```

```
struct WaterDrop{
    float xpos;
    float ypos;
};
```

WaterDrop: 물방울의 정보를 저장하는 structure.
x좌표와 y좌표로 구성된다.

```
struct WaterLine{
    int startX;
    int startY;
    int toX;
    int toY;

    float slope;
};
```

WaterLine: 물받이용 선반의 정보를 저장하는 structure. 시작 x, y좌표와 끝 x, y좌표로 구성된다.
1주차에 비해 새롭게 추가된 것은 float slope;
해당 물받이용 선반의 기울기이다.

```
slope = (double)(line.toY-line.startY) / (double)
(line.toX-line.startX);
```

```
class ofApp : public ofBaseApp{
    private:
        int draw_flag;
        int flow_flag;
        int activeIdx=0;
        vector<WaterLine> lines;
        vector<WaterPoint> points;
        vector<WaterDrop> drops;
```

#include <vector>를 통해 vector 사용.
1주차에 비해 새롭게 추가된 것은 vector<Waterdrop>drops와 flow_flag이다. flow_flag는 현재 프로그램에서 물이 흐르고 있는지 여부를 나타낸다.

vector<WaterDrop> drops를 통해 연속된 물방울 정보 structure 할당.

<ofApp.cpp>

```
void ofApp::update(){
    if(this->flow_flag==1)
        moveCal(this->points[this->activeIdx], this->lines);
}
```

update 함수에 moveCal 함수를 실행시키는 구문을 넣어 활성화된 점에서의, 물 흐름에 따른 물방울 위치를 계산한다. moveCal 함수에 대해서는 밑에서 다룰 예정이다.

```
void ofApp::draw(){
    …앞 내용 생략…
    if(this->flow_flag==1 && this->draw_flag==1){
        for(int i=0; i<idx; i++)
            drawDrop(this->drops[i]);
        idx++;
    }

}
```

draw 함수에서, drawDrop 함수를 호출한다. flow_flag가 set되었을 때만 호출되며, 물받침대와 구멍이 그려진 후 물이 흘러야 되므로 draw_flag 또한 set된 상태여야 한다. 0번째부터, 물방울이 그려진 개수 idx까지 반복해서 호출하여 물 흐름이 보이도록 한다.

```
void ofApp::keyPressed(int key){

    …생략…

    if (key=='s'){
        if(this->draw_flag==1)
            this->flow_flag=1;
        cout << "Start Flow key Pressed" << endl;

    }

    if (key=='e'){
        this->flow_flag=0;
        cout << "End Flow key Pressed" << endl;
        idx=0;
    }

    …생략…

    if (key=='q'){
        cout << "Quit" << endl;

        this->lines.clear();
        this->points.clear();
        this->drops.clear();



        vector<WaterPoint>().swap(this->points);
        vector<WaterLine>().swap(this->lines);
        vector<WaterDrop>().swap(this->drops);

        OF_EXIT_APP(0);
    }


}
```

keyPressed 함수는 int형 변수 key를 받아 각 key에 맞는 동작을 수행한다.

key가 s일 경우, “Start Flow key Pressed”라는 문구를 출력하고, draw_flag가 set되어 있는 경우 flow_flag를 set한다. 이를 통해 물 흐름이 시작된다.

key가 e일 경우, “End Flow Pressed”라는 문구를 출력하고, flow_flag를 0으로 설정한다. 이를 통해 물 흐름이 끊기게 된다.

위 코드에는 생략되어 있지만, key가 ←, → 인 경우 idx=0으로 설정하여 물 흐름을 다시 처음부터 시작하게 만든다.

key가 q일 경우, “Quit”라는 문구를 출력하고, this->drops.clear(), this->lines.clear(), this->points.clear()를 이용해 size를 0으로 설정한다. 그 이후 capacity를 0으로 만들기 위해 swap method를 사용한다. 컨테이너 객체끼리 교환할 수 있는 swap() 멤버 함수를 이용하여, 임시로 생성한 (기본 생성자에 의해 size, capacity가 0인) 컨테이너 객체와 this->drops, this->points, this->lines를 swap한다. 이를 통해 메모리 해제를 이룰 수 있다. 마지막으로 OF_EXIT_APP(0)을 통해 프로그램을 종료한다.

```
void ofApp::drawDrop(WaterDrop water){
    ofSetColor(0, 0, 255);
    ofDrawCircle(water.xpos, water.ypos, 2);
}
```

drawDrop 함수는 색깔 파란색, 반지름 2인 원 모양으로 물방울을 그린다.

```
void ofApp::moveCal(WaterPoint activePoint, vector<WaterLine>
lines){
    this->drops.clear();

    float xPos, yPos;

    xPos=activePoint.X;
    yPos=activePoint.Y;

    WaterDrop tmpWater;

    for(int i=0; i<10000; i++){
        yPos++;

        for(int j=0; j<lines.size(); j++){
            if((int)yPos == (int)(lines[j].startY+ (xPos-
lines[j].startX)*lines[j].slope)){
                if(lines[j].startX<=xPos && xPos<=lines[j].toX){
                    xPos += (1/lines[j].slope);
                    yPos--;
                    yPos += lines[j].slope;
                }
            }
        }
        tmpWater.xpos=xPos;
        tmpWater.ypos=yPos;

        this->drops.push_back(tmpWater);
    }
}
```

moveCal 함수는 물방울들의 경로를 계산하여 구조체 vector에 순차적으로 저장하는 역할을 한다. 임시 좌표값 xPos, yPos를 만들어 yPos 값이 1씩 증가함에 따라 주어진 조건에 따라 xPos가 slope에 따라 증감하는 방식으로 진행된다. (int)yPos == (int)(lines[j].startY+(xPos-lines[j].startX)

*lines[j].slope)를 조건으로 하는데, for 문에서 j의 값을 0부터 물 받침대 개수-1 까지 반복하며 비교를 한다. 위 반복문의 골자는 물방울 점의 y좌표를 int형으로 형변환한 값과 현재 물방울의 x좌표에서의 물받침 선의 y좌표를 int형으로 형변환한 값이 같은지를 비교하는 것이다. 간단히 말해, x=k와 y=ax+b의 그래프의 교점을 생각하는 것이다. 만약 같다면 물방울 점이 해당 물받침 (lines[j]) 위(근방)에 있다고 해석할 수 있다. 물받침 점 위에 있다는 것이 판별되면, 물방울 점이 lines[j].startX와 lines[j].toX 사이에 있을 때 까지만 물이 선을 따라 흘러내리도록 한다.

계산, 조건 확인이 끝나면 this->drops.push_back(tmpWater)을 통해 vector에 원소를 추가한다.

가장 바깥쪽 for문은 yPos를 연속적으로 증가시키며 반복하는 역할을 하는데, main.cpp에 창의 높이가 768로 설정되어 있으므로 충분한 반복 횟수이다.