컴퓨터공학실험1　9주차 결과보고서

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

1. OpenFramework 실습 코드들을 수행하고, 각 line이 어떤 기능을 수행하는지 작성한다.

<ofApp.h>

```
#pragma once

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


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

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

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

       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);


    //int xPos;
    // WaterFall-related member variables
        //int draw_flag;
        //int load_flag;

    // WaterFall-related member functions
        void processOpenFileSelection(ofFileDialogResult openFileResult);
        void UserLine(WaterLine line);
        void UserCircle(WaterPoint point);

};
```

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

WaterLine: 물받이용 선반의 정보를 저장하는 structure.
시작 x, y좌표와 끝 x, y좌표로 구성된다.

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

WaterPoint: 물이 흘러나오는 점들의 좌표 정보 저장하는 structure. (x, y 좌표)
+ 활성화 상태를 저장한다.

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

#include <vector>를 통해 vector 사용.
draw_flag는 현재 프로그램에서 도형 (점, 선)이 그려져 있는 상태인지 여부를 저장.
activeIdx는 활성화된 점의 index를 나타냄. (처음엔 0으로 초기화)
vector<WaterLine> lines를 통해 연속된 선 정보 structure 할당.
vector<WaterPoint> points를 통해 연속된 점 정보 structure 할당.

<ofApp.cpp>

```
#include "ofApp.h"
#include <stdio.h>
#include <fstream>

//--------------------------------------------------------------
void ofApp::setup(){
    ofSetLineWidth(5);
    ofSetBackgroundColor(255, 255, 255);
    this->draw_flag=0;
}
```

ofSetLineWidth 함수로 선의 굵기를 5로 설정하고, ofSetBackgroundColor 함수로 배경색을 흰색으로 설정한다.
this 포인터-> draw_flag를 0으로 초기화하여 처음엔 그림을 그리지 않았음을 나타낸다.

update 함수는 빈 함수로 두었으므로 생략한다.

```
void ofApp::draw(){
    if(this->draw_flag==1){
        ofSetColor(143, 0, 25);
        for(int i=0; i<this->lines.size(); i++)
            UserLine(this->lines[i]);

        ofSetColor(0, 0, 0);
        for(int i=0; i<this->points.size(); i++)
            UserCircle(this->points[i]);

      }
}
```

draw 함수는 this->flag가 set 되었을 때 점과 선을 그린다. line은 그리기 전 색깔을 붉은색 (143, 0, 25)로 설정한 후, 입력받은 lines의 수만큼 반복하여 출력한다. Userline 함수는 후술할 예정. circle은 그리기 전 색깔을 검은색으로 설정하고, 입력받은 circle의 수만큼 반복해서 출력한다.

```
void ofApp::keyPressed(int key){
    //cout << key << endl;
    if (key=='l'){
        cout << "L key Pressed" << endl;
        ofFileDialogResult fileResult=ofSystemLoadDialog("Load input
file");
        processOpenFileSelection(fileResult);
    }

    if (key=='d'){
        this->draw_flag=1;
        cout << "Draw key Pressed" << endl;
    }

    if (key==OF_KEY_LEFT){
        cout << "Left key Pressed" << endl;
        //
        //this->draw_flag++;
        if(this->draw_flag==1){
            //최소까지 이동
            if(activeIdx>0){
                this->points[activeIdx--].active=false;
                this->points[activeIdx].active=true;
            }
            else{
                this->points[activeIdx].active=false;
                this->points[this->points.size()-1].active=true;
                activeIdx=this->points.size()-1;
            }
        }

    }
```

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

key가 l일 경우, “L key Pressed”라는 문구를 출력하고, processOpenFileSelection함수를 이용하여 파일 선택을 위한 Finder 창을 연다.

key가 d일 경우, “Draw key Pressed”라는 문구를 출력하고, this->draw_flag를 1로 set한다.
key가 ←일 경우, “Left key Pressed”라는 문구를 출력하고, activeIdx가 0보다 크다면 활성화된 점을 한 칸씩 앞으로 당긴다. 아니라면 activeIdx를 입력받은 점의 개수-1로 바꾼다. (this->points[activeIdx].active를 set한다.)

```
    if (key==OF_KEY_RIGHT){
        cout << "Right key Pressed" << endl;
        //cout << "Num of point: " << this->points.size() << endl;
        //
        //this->draw_flag++;
        if(this->draw_flag==1){
            //최대까지 이동
            if(activeIdx< this->points.size()-1){
                this->points[activeIdx++].active=false;
                this->points[activeIdx].active=true;
            }
            else{
                this->points[activeIdx].active=false;
                this->points[0].active=true;
                activeIdx=0;
            }
        }
    }

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

        //free(this);
        //delete this;
        this->lines.clear();
        this->points.clear();

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

        OF_EXIT_APP(0);
    }
}
```

key가 →일 경우, “Right key Pressed”라는 문구를 출력하고, activeIdx가 입력받은 점의 개수-1보다 작다면 활성화된 점을 한 칸씩 뒤로 민다. 아니라면 activeIdx를 0으로 바꾼다. (this->points[activeIdx].active를 set한다.)
key가 q일 경우, “Quit”라는 문구를 출력하고, 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->points, this->lines를 swap한다. 이를 통해 메모리 해제를 이룰 수 있다. 마지막으로 OF_EXIT_APP(0)을 통해 프로그램을 종료한다.
keyReleased, mouseMoved… dragEvent 함수는 빈 함수로 두었으므로 생략한다.

```
void ofApp::processOpenFileSelection(ofFileDialogResult openFileResult) {
    this->lines.clear();
    this->points.clear();

    if (openFileResult.bSuccess){
        string line;
        int lines, startX, startY, toX, toY, pointX, pointY;

        //파일 열기
        string file_location=openFileResult.getPath();
        ifstream input_file;
        input_file.open(file_location);

        if(input_file.is_open()){
            //선분
            input_file >> lines;
            for(int i=0; i<lines; i++){
                input_file >> startX;
                input_file >> startY;
                input_file >> toX;
                input_file >> toY;

                cout << "[read line] stX: " << startX << ", stY: " << startY <<
", toX: " << toX << ", toY: " << toY <<endl;

                WaterLine line;
                line.startX=startX;
                line.startY=startY;
                line.toX=toX;
                line.toY=toY;
                this->lines.push_back(line);
            }

            //점
            input_file >> lines;
            for(int i=0; i<lines; i++){
                input_file >> pointX;
                input_file >> pointY;
                cout << "[read point] x: " <<pointX << ", y: " << pointY <<
endl;

                WaterPoint point_tmp;
                point_tmp.X=pointX;
                point_tmp.Y=pointY;
                //
                if(i==0) point_tmp.active=true;
                else point_tmp.active=false;

                //
                this->points.push_back(point_tmp);
            }
        }
        else cout << "Failed to open file" << endl;
    }
}
```

processOpenFileSelection 함수는 선택된 파일을 입력받아 미리 지정한 구조체에 순차적으로 점, 선 정보를 저장하는 역할을 수행한다.

string file_location=openFileResult.getPath(), ifstream input_file, input_file.open(file_location) 을 통해 파일 열기를 수행한 후 그 다음부터 본격적으로 상수 정보를 입력받는다. 선의 개수를 입

력받은 후, for문 안에서 선의 시작 좌표, 끝 좌표 입력+저장, 터미널 창에 입력받은 값 출력을 한다.
이후 WaterLine형으로 line을 만들어 push_back 시킨다. (vector 이용)
점의 개수를 입력받은 후, for문 안에서 점의 좌표 입력+저장, 터미널 창에 입력받은 값 출력을 한다.
이후 WaterPoint형으로 point_tmp를 만들어 push_back 시킨다. (vector 이용)
맨 처음, 0번째 점을 active로 설정해 준다.

openFileResult.bSuccess가 아니라면 "Failed to open File"을 출력해준다.

```
void ofApp::UserLine(WaterLine line) {
    ofDrawLine(line.startX, line.startY, line.toX, line.toY);
}

void ofApp::UserCircle(WaterPoint point) {
    if(point.active){
        ofSetColor(255, 0, 0);
        ofDrawCircle(point.X, point.Y, 10);
    }
    else{
        ofSetColor(0, 0, 0);
        ofDrawCircle(point.X, point.Y, 10);
    }

}
```

Userline 함수는 선의 시작좌표부터 끝 좌표까지 선분을 그린다.
UserCircle 함수는 해당 점이 active인 경우는 점을 붉은색으로 그려주고, 아닌 경우에는 검은색으로 그려준다. 입력받은 좌표의 위치에 그린다. (반지름 10)