9주차 결과보고서

전공: 수학과,컴퓨터공학과 학년: 3학년 학번: 20181256 이름: 김도현

1.

1-1. Basic shape(1)(2)

가장 기본적인 도형들을 그리는 실습으로 처음에는 안쪽 색깔을 채우고 두번째 실습은 안쪽 색깔을 비워보았다.

draw()함수는 사용자가 원하는 도형을 그릴 때 사용하는 함수이다.

ofBackground(a) //int a를 받아 도형이 그려질 배경화면의 색깔을 결정하는 것으로 parameter int grayscale인 a가 0에 가까울수록 검은색, 255에 가까울수록 흰색이다. 이때 특정 색을 지정하고 싶다면 ofSetColor(a,b,c)로 하여 r,g,b를 (a,b,c) 의 색으로 칠하게 된다.

ofSetColor(a)// int a를 받아 도형 내부의 색깔을 r,g,b를 grayscale인 (a,a,a)인 색으로 칠한다. 이때 특정 색을 지정하고 싶다면 ofSetColor(a,b,c)로 하여 r,g,b를 (a,b,c) 의 색으로 칠하게 된다.

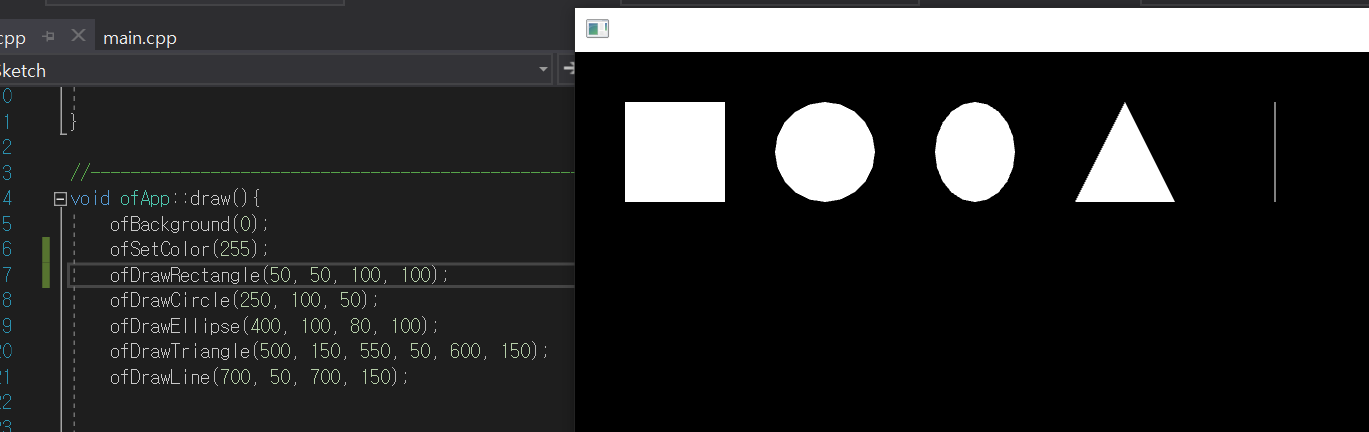
ofDrawRectangle(a,b,c,d) //float a,b,c,d를 받아 이름처럼 직사각형의 왼쪽 위 부분을 x좌표=a, y좌표=b 로 두고 width(가로)=c, height(높이)=d 로 두어 직사각형을 그린다.

ofDrawCircle(a,b,c) // float a,b,c 를 받아 원의 중심인 x 좌표를 a, y좌표를 b 로 두고 반지름이 c인 원을 그린다.

ofDrawEllipse(a,b,c,d) // float a,b,c,d를 받아 타원의 중심인 x 좌표를 a, y좌표를 b 로 두고 가로 길이를 c, 세로(높이) 를 d 로 두어 타원을 그린다.

ofDrawTriangle(a,b,c,d,e,f) //float a,b,c,d,e,f 를 받아 삼각형의 세 꼭짓점의 x,y 좌표를 각 (a,b) (c,d) (e,f)로 두어 삼각형을 그린다.

ofDrawLine(a,b,c,d) //float a,b,c,d를 받아 선분의 두 꼭짓점의 x,y 좌표를 각 (a,b) (c,d) 로 두어 (a,b)에서 (c,d)로 가는 선분을 그린다.



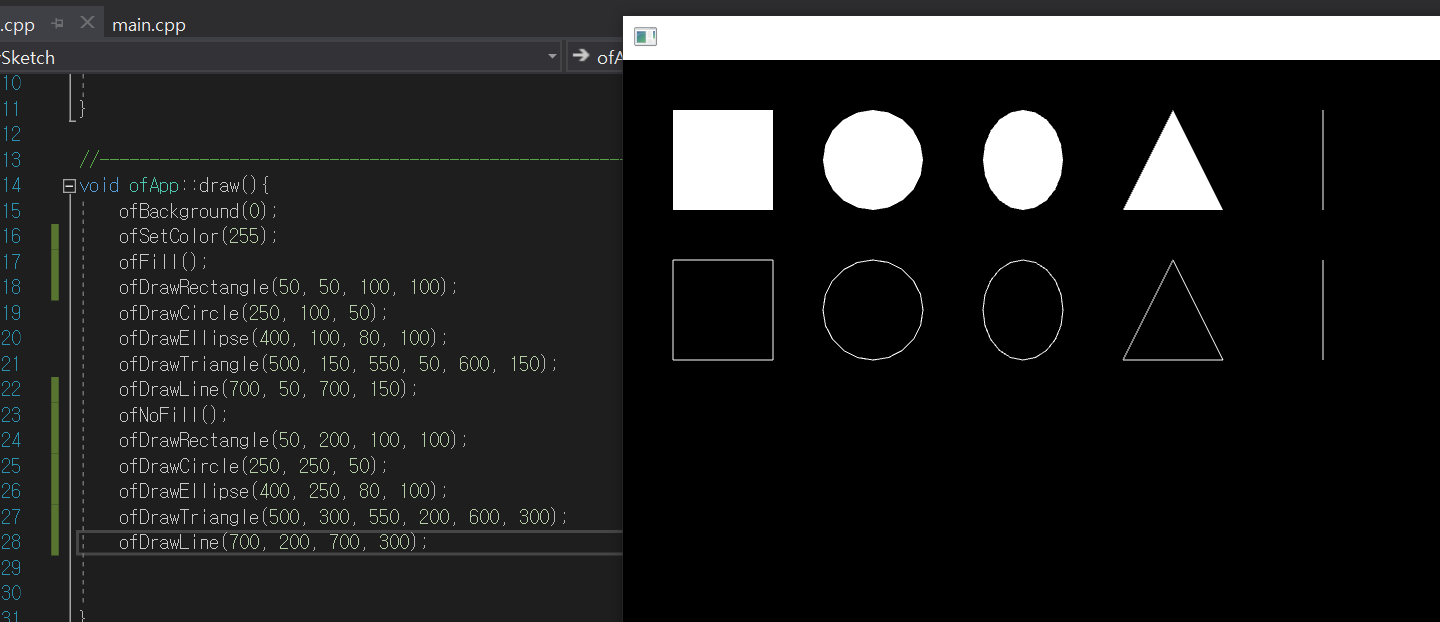
1-2.

이후 Basic Shape(2) 에서는 ofFill, ofNoFill를 추가해보았다.

추가된 함수들만 정리했다. 만약 draw() 안에 둘 중 하나의 함수만 있다면 그 함수가 Draw() 안에 있는 모든 함수에 적용되므로 주의해야 한다. 즉 ofNoFill()이 중간에 있어도 그 위의 도형들 역시 내부가 비어있다.

ofFill() //도형 내부의 색을 칠하라는 의미의 함수이다.

ofNoFill() //도형 내부의 색깔을 칠하지 말라는 의미의 함수이다.



1-3. Single rectangle brush

마우스에 따라 직사각형이 움직이도록 하는 실습으로 처음에는 마우스만 따라 움직이고 두번째는 마우스를 떼어도 계속 도형이 남아있고 3번째는 직사각형이 random한 색깔들로 나오도록 한다.

If(ofGetMousePressed(OF\_MOUSE\_BUTTUN\_LEFT){//이름 그대로 마우스의 왼쪽 버튼이 눌렸는지 눌리지 않았는지 check 하는 코드이다. 만약 눌렸다면 아래의 문장들을 실행한다.

OfSetColor(a) //Basic shape 에서 설명한 것 처럼 int a를 받아 도형 내부의 색을 r,g,b 의 a 색으로 칠한다.

ofSetRectMode(OF\_RECTMODE\_CENTER) //직사각형을 그릴 때 default값은 왼쪽 위 꼭짓점 기준인데 이 코드를 통해 기준을 직사각형의 중심으로 결정한다.

ofDrawRectangle(ofGetMouseX(),ofGetMouseY(),a,b) //마우스의 위치를 ofGetMouse로 가져와서 직사각형의 X축은 마우스의 X 좌표, 직사각형의 Y좌표는 마우스의 Y 좌표로 설정하고 가로, 세로길이 역시 float 값인 a,b로 두고 직사각형을 그린다.

}

텍스트이(가) 표시된 사진

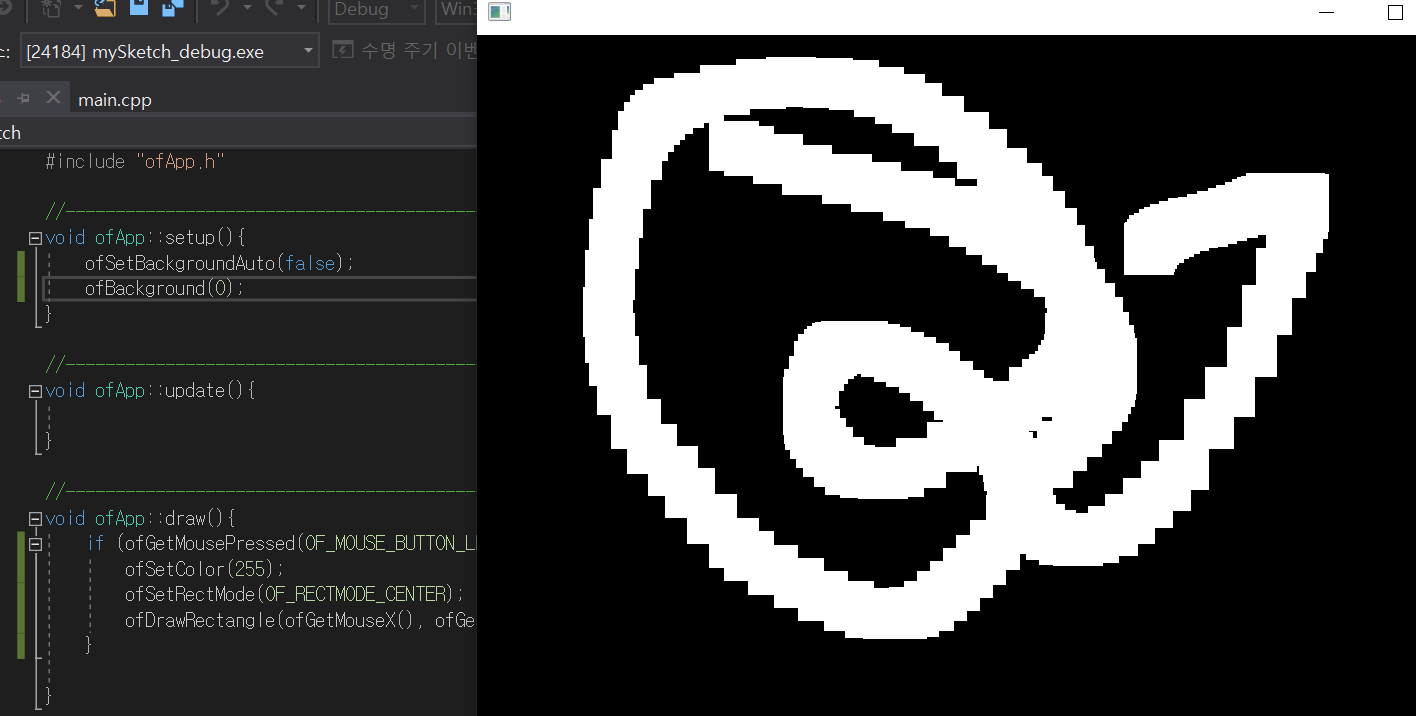
자동 생성된 설명

캡쳐할 때는 마우스의 왼쪽 키를 screen에 클릭하지 못하기 때문에 직사각형이 나오는 것은 캡쳐하지 못했다. 이 말은 왼쪽 키를 띄면 프레임이 일정 시간동안 계속 실행되므로 화면이 clear 된다는 의미이다.

이후 setup() 함수에서 다음 코드를 추가하여 왼쪽 마우스 키를 떼도 그림이 남아있도록 해보았다.

ofSetBackgroundAuto(false) //기본적으로 그림을 그려주는 screen은 draw가 실행될 때 마다 초기화(clear)되기 때문에 만약 계속 그림을 유지하고 싶다면 이 것을 false로 둬야 한다.

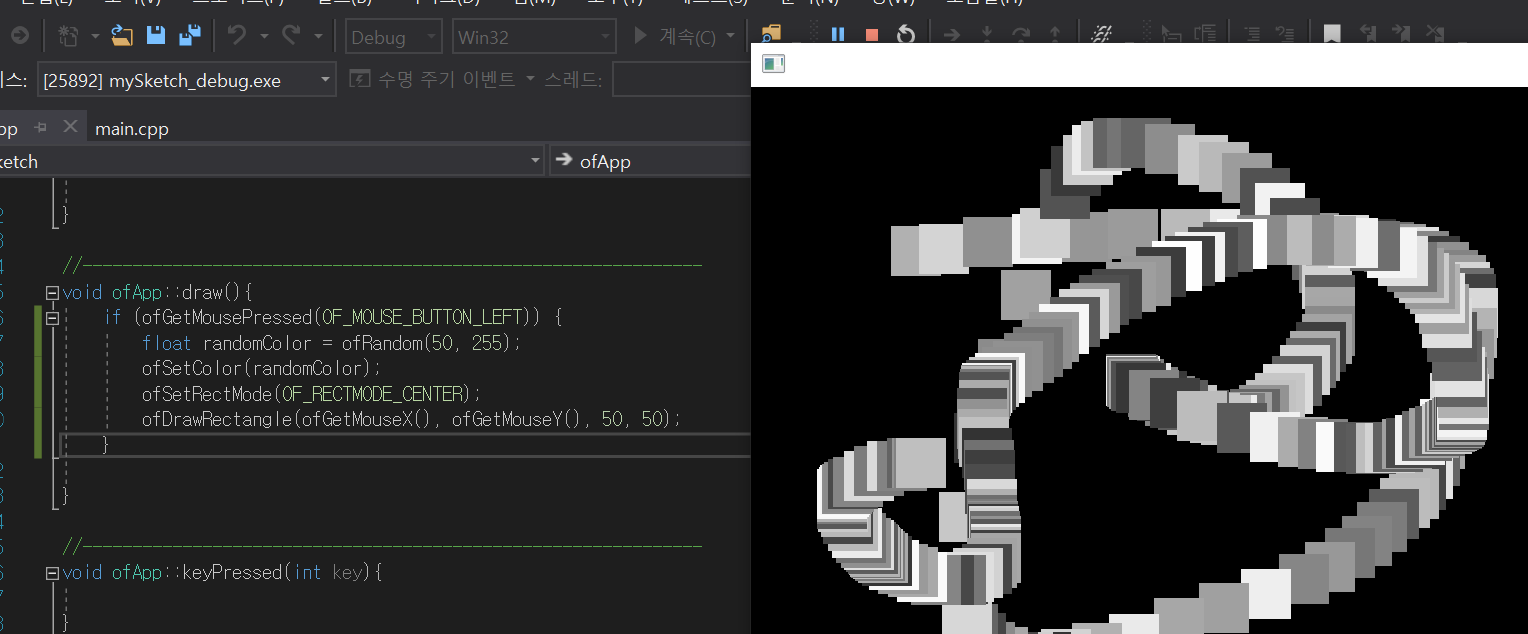
ofBackground(a) //int a를 받아 뒷 배경의 색을 r,g,b의 grayscale인 a 색으로 칠한다.



다시 draw() 함수에서 다음 코드를 추가 및 수정하였다.

float randomColor=ofRandom(50,255)//50에서 255사이의 random한 숫자를 하나 받아 randomColor에 저장한다. 0~50의 색은 배제하여 검은색에 가까운 색은 없다. 즉 배경과 거의 비슷한 색이 없기때문에 구분하기가 쉬울 것이다.

ofSetColor(randomColor) //이 randomColor를 이용하여 도형 내부의 색을 칠하도록 한다. Draw 함수가 실행될 때마다 색이 ofRandom 역시 실행되기 때문에 색이 계속 바뀌는 것을 확인할 수 있다. 나머지 코드는 기존의 코드와 같다.



1-4. Bursting Rectangle Brush

랜덤하게 한 지점을 중심으로 주변의 지점까지 서로 크기가 다른 사각형들이 폭발적으로 나타나도록 그리도록 한다.

If(ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT){

ofSetRectMode(OF\_RECTMODE\_CENTER);// 직사각형을 그릴 때 default값은 왼쪽 위 꼭짓점 기준인데 이 기준을 직사각형의 중심으로 결정한다.

int numRects=10;

for(int r=0;r<numRects;r++){ // 한번 왼쪽 마우스가 눌릴 때마다 10번의 for문을 돌며 다음 코드를 실행하며 10개의 직사각형을 그린다.

ofSetColor(ofRandom(50,255)); //직사각형 내부 색은 grayscale 50~255사이로 random 하게 정한다.

float width=ofRandom(5,20); //직사각형 가로 길이는 5~20 사이의 수로 random 하게 한다.

float height=ofRandom(5,20); //직사각형 세로 길이는 5~20 사이의 수로 random 하게 한다.

float xOffset=ofRandom(-40,40); //마우스의 x축에 더하는 값으로 왼쪽으로 최대 40, 오른쪽으로 최대 40만큼 가도록 random하게 설정한다.

float yOffset=ofRandom(-40,40);// 마우스의 y축에 더하는 값으로 위쪽으로 최대 40, 아래쪽으로 최대 40만큼 가도록 random하게 설정한다.

ofDrawRectangle(ofGetMouseX()+xOffset,ofGetMouseY()+yOffset, width,height); //직사각형의 중심의 x좌표를 마우스의 x좌표에 xOffset을 더하고 중심의 y좌표도 마우스의 y좌표에 yOffset을 더한다. 그리고 위에서 설정한 것처럼 가로 세로 길이도 준다.

}}

이후 setup() 함수에

ofSetFrameRate(60) //refresh rate를 60으로 하여 초당 60번 refresh하도록 설정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1-5. Glowing Circle Brush

기존의 직사각형이었던 도형을 원으로 바꾸되 원의 색, 반지름 등을 조정해주고 투명도 역시 조절을 하는 실습이다. 여기서는 ppt에 특정 코드가 없어 실험자가 직접 코드를 짜고 투명도를 조절하였다. 만약 ppt 예시 사진과 비슷하게 하려면 투명도를 더욱 낮추고(10 이하 수준) 색 역시 더욱 어두운 색으로 해야 할 것이다. (하지만 결국 이 실습의 목적은 색, 투명도 바꾸기 이기 때문에 자신이 원하는 방향으로 코드를 작성했다.)

기존의 Rectangle Brush에서 직사각형을 원으로 바꿔주고 다음과 같이 일부 색을 바꿔주었다.

ofColor myOrange(255,132,0); // 여기서는 사용자가 직접 색깔을 설정하는 것으로 r,g,b를 마음대로 지정할 수 있다.

ofColor myBlue(0,0,255,50) //myBlue를 투명도까지 주어 지정할 수 있다. 여기서 투명도는 r,g,b 다음에 쓰이는 a 이다. 50% 정도의 투명도를 주려면 a를 127로 설정하면 된다.

ofColor myGreen(0,0,255,255) //myGreen이지만 B에 255, 투명도 255로 B=255로 꽉 채워진 색으로 볼 수 있다.

cout<<”Red channel:”<<myGreen.r<<endl; // print하는 함수로 myGreen.r 인 Red Channel이 0이 맞는지 확인하는 것이다.

cout<<”Green channel:”<<myGreen.g<<endl; // print하는 함수로 myGreen.g 인 Green Channel이 0이 맞는지 확인하는 것이다.

cout<<”Blue channel:”<<myGreen.b<<endl; // print하는 함수로 myGreen.b 인 Blue Channel이 255가 맞는지 확인하는 것이다.

cout<<”Alpha channel:”<<myGreen.a<<endl; // print하는 함수로 myGreen.a 인 Alpha Channel이 255가 맞는지 확인하는 것이다.

ofColor myYellow;

myYellow.r=255;myYellow.b=0;myYellow.g=255;myYellow.a=255; //이렇게 myYellow라는 색을 각각 따로 channel을 지정해 줄 수도 있다.

ofColor myAuqa=ofColor::aqua; //사용자가 따로 색을 지정해주지 않아도 미리 of에서 만들어져 있는 색을 받아올 수 있다.

자세한 구현 코드는 ppt에 나오지 않아 임의로 작성하였다.

위를 확인했으면 앞에서 직사각형을 그린 것처럼 코드를 작성해 주되 여기서는 ofColor에 대한 실험이기 때문에 색깔을 임의로 설정하고 getLerped 함수를 이용해 정한 두 색깔 사이의 임의의 random한 새로운 색을 A에 저장하여 이 색을 사용하였다. 그리고 radius는 (5~20) 사이의 random한 반지름으로 잡았다. 나머지는 Bursting Rectangle Brush와 같기 때문에 설명은 생략한다.

ofDrawCircle(ofGetMouseX()+xOffset,ofGetMouseY()+yOffset,radius)//이 코드는 마우스의 x,y 좌표를 해당 원의 중심 좌표로 하고 랜덤한 xOffset, yOffset을 더해주어 bursting한 모양이 나오도록 하였고 radius 는 위에 언급한 것 처럼 5~20 사이의 수로 random하게 지정했다.(임의로 코드를 짜보았다.)

만약 투명도를 50이 아닌 10 이하로 낮추었다면 ppt의 그림대로 매우 희미하게 구름과 같은 모양이 나오겠지만 결국 이 실습은 코드를 작성해보고 연습을 해야 하기 때문에 투명도를 50으로 설정하여 해보았다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1-6. Fleeing triangle brush

기존의 Rectangle Brush에서 직사각형을 삼각형으로 바꿀 때 vector를 이용하여 도형을 그려보았다. 이 역시 왼쪽 마우스 클릭의 위치에 따라 삼각형들이 자유분방하게 발산하는 형상을 띈다.

바뀐 부분의 코드만 살펴보았다.

ofVec2f mousePos(ofGetMouseX(),ofGetMouseY()) // 마우스의 x,y좌표 역시 나중의 벡터 연산을 위해 ofVec2f로 지정해두었다.

ofVec2f p1(0,25.0) //vector의 기준은 (0,0) 이므로 다음은 (0,0)에서 (0,25) 로 가는 vector를 만들었다.

ofVec2f p2(100,0)//(0,0)에서 (100,0) 으로 가는 vector를 만들었다.

ofVec2f p3(0,-25.0) //(0,0)에서 (0,-25)로 가는 vector를 만들었다.

float rotation=ofRandom(360)//rotation을 줄 경우 360까지의 숫자 중 random하게 하나를 뽑아

p1.rotate(rotation)// p1, p2, p3 에 random한 rotation을 준다.

p2.rotate(rotation)

p3.rotate(rotation)

float offsetDistance=ofRandom(5,70); //5~70사이의 random한 숫자를 하나 뽑아 한 클릭에 여러 곳으로 삼각형이 그려지도록 한다.

ofVec2f Offset(offsetDistance,0,0);//위의 offsetDistance를 x좌표로 두어 벡터를 만든다.

Offset.rotate(rotation) //Offset 벡터에도 같은 rotation을 준다.

p1+=mousePos;//p1,p2,p3에 각각 마우스의 x,y 좌표의 벡터를 만든 mousePos를 더해준다.

p2+=mousePos;

p3+=mousePos;

p1+=Offset; //p1,p2,p3에 여러곳에서 삼각형이 그려지도록 Offset 값도 각각 더해준다.

p2+=Offset;

p3+=Offset;

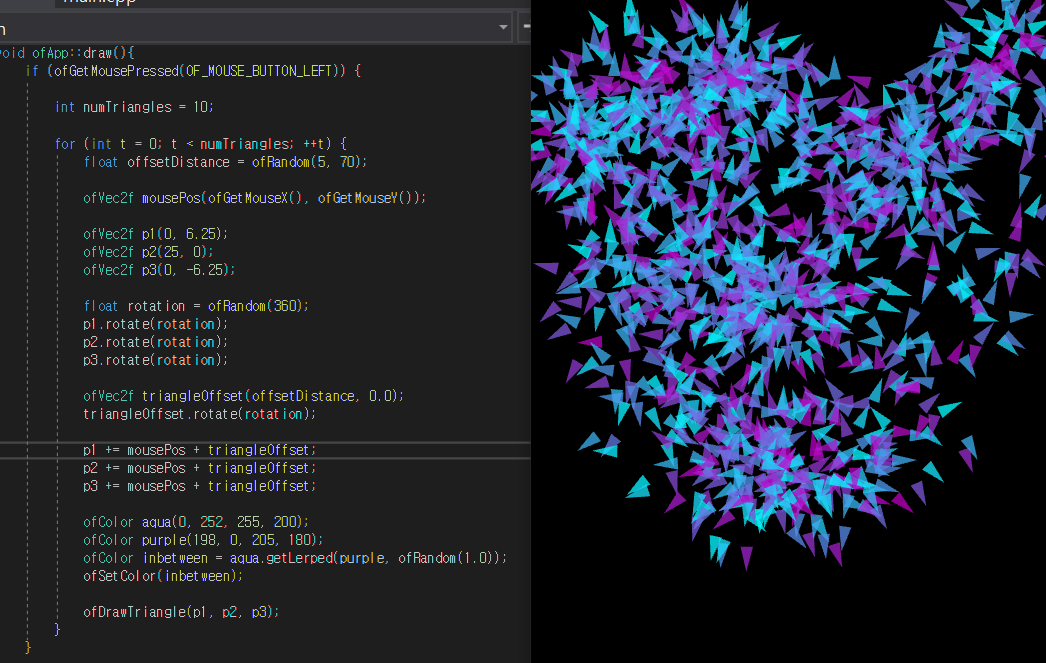
ofColor aqua(0,252,250)// Glowing circle Brush에서 색을 지정한 것처럼 임의로 색을 지정해준다.

ofColor crystal(150,0,200,190)// 색깔에 투명도를 약간 줄 수도 있다.

ofColor A=aqua.getLerped(crystal,ofRandom(1.0));// getLerped는 0~1 까지의 float 가 들어올 때 두 aqua와 crystal 사이의 색을 나타내준다. Float는 새로운 색이 원래의 색과 얼마나 비슷한지 나타내는 지표이다.

ofSetColor(A) //두 색 사이의 색으로 지정한다.

ofDrawTriangle(p1,p2,p3)//위에서 계산한 P1,p2,p3를 가지고 해당 마우스의 왼쪽 키가 눌러졌을 때 random하게 삼각형이 그려지도록 한다.



1-7.update()

일정시간동안 계속해서 화면(프레임)을 update시켜 계산된 결과에 따라 도형을 계속 draw 하도록 하는 실습이다. 이 update() 함수는 나중에 10주차 실습인 waterfall 구현의 물이 떨어질 때 일정시간마다 물이 떨어질 위치를 계산하는 계산식을 구하는데 중요할 것으로 예상된다.

우선 ofApp.h 에 int xPos라는 멤버 변수를 넣었다.(draw()에 그릴 원의 x좌표를 의미한다.)

ofApp.cpp

setup() 함수

xPos=5// x좌표를 5로 지정하고

ofBackground(ofColor::black)// 배경색은 검은색으로 지정한다

update() 함수 //frame당 한번씩 update 되는 것으로 frame당 한 번 이 함수는 호출된다.

xPos+=2 //x좌표를 2를 증가시킨다

if(ofGetWidth()<xPos) //만약 screen의 가로 길이보다 xPos가 더 크다면

xPos=5 //xPos를 5로 재설정한다.

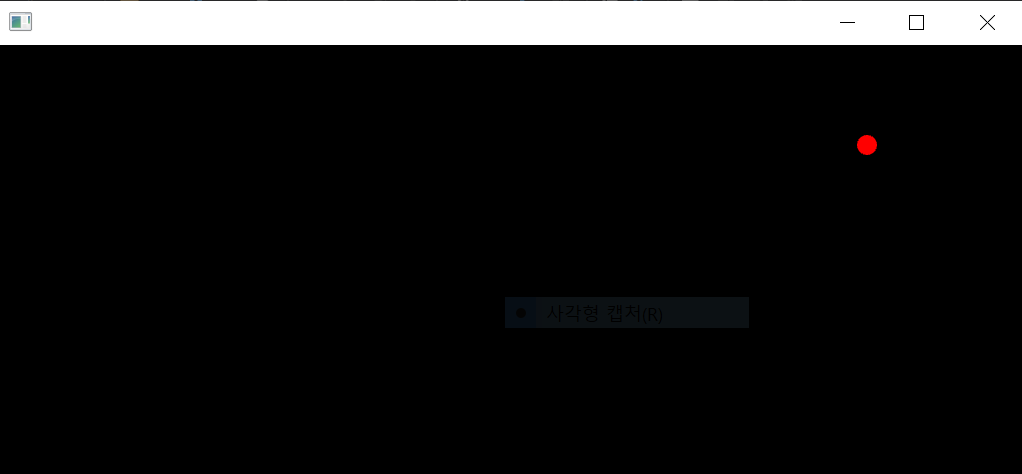
draw() 함수

ofSetColor(ofColor::red) //색을 빨간색으로 지정하여 그린다

ofDrawCircle(xPos,100,10)//원의 중심을 (xPos,100) 으로 그리고 반지름을 10인 원을 그리도록 한다.

Xpos 가 update함수에 의해 계속 update 되기 때문에 시간이 지나면 빨간 원이 계속 왼쪽으로 오른쪽으로 갔다가 다시 xPos가 5로 돌아오는 것이 반복될 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

정상적으로 일정시간동안 update 함수가 계속 실행되어 왼쪽에서 오른쪽으로 원이 이동하는 것을 알 수 있다.