9주차 결과보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181294 이름: 임승섭

**1.**

1 - 1. Basic Shape (1)

사각형, 원, 타원, 삼각형을 화면에 그린다. of함수를 사용하며 함수 실행 시 필요한 인자에 대해 학습하였다.

void ofApp::draw(){

ofBackground(0); // 배경은 검정색으로 설정한다.

ofSetColor(255); // 도형의 색깔을 설정한다. 255 는 흰색을 의미한다.

ofDrawRectangle(50, 50, 100, 100); 왼쪽 위 꼭짓점의 위치가 (50, 50)이고 가로 100 세로 100 size의 직사각형을 그린다.

ofDrawCircle(250, 100, 50); // 중심의 위치가 (250, 100)이고 반지름이 50인 원을 그린다.

ofDrawEllipse(400, 100, 80, 100); // 중심의 위치가 (400, 100)이고 가로 80 세로 100 size의 타원을 그린다.

ofDrawTriangle(500, 150, 550, 50, 600, 150); // (500, 150), (550, 50), (600, 150) 을 꼭짓점으로 갖는 삼각형을 그린다.

ofDrawLine(700, 50, 700, 150); // (700, 50)부터 (700, 150)까지 이어지는 선분을 그린다.

}

1 - 2. Basic Shape (2)

위에서 그렸던 도형들을 다시 한 번 그리는데, 속이 뚫려있는 도형을 그린다. 즉, 테두리만 존재하고 도형 속은 배경 색과 같은 도형을 그린다. ofFill()과 ofNoFill() 함수를 사용한다. 주의해야 할 점은 두 함수 중 하나만 draw() 안에 있을 시, draw에서 그려지는 모든 도형이 그 함수에 따른다.

void ofApp::draw(){

ofBackground(0); // 배경은 검정색으로 설정한다.

ofSetColor(255); // 도형 내부의 색이 있다면, 흰색으로 설정한다.

ofFill(); // 도형 내부의 색을 칠한다.

ofDrawRectangle(50, 50, 100, 100);

ofDrawCircle(250, 100, 50);

ofDrawEllipse(400, 100, 80, 100);

ofDrawTriangle(500, 150, 550, 50, 600, 150);

ofDrawLine(700, 50, 700, 150);

ofNoFill(); // 도형 내부의 색을 칠하지 않는다. 결국 도형 내부의 색은 배경과 동일하다.

ofDrawRectangle(50, 200, 100, 100);

ofDrawCircle(250, 250, 50);

ofDrawEllipse(400, 250, 80, 100);

ofDrawTriangle(500, 300, 550, 200, 600, 300);

ofDrawLine(700, 200, 700, 300);

}

1 - 3. Single rectangle brush

마우스를 왼쪽 버튼을 클릭했을 때, 직사각형 모양이 나오도록 하는 코드를 작성한다. setup() 내의 코드를 작성하기 전에는 손을 떼면 직사각형이 바로 사라지고, 코드 작성 후에는 화면에 표시된 직사각형이 계속 유지된다. 직사각형의 색깔을 특정 색으로 지정하고 싶으면 draw 함수 내의 ofSetColor()의 인자로 원하는 색의 number를 적어주고, ofRandom() 함수를 통해 색이 계속 바뀌게 할 수도 있다.

void ofApp::setup(){

ofSetBackgroundAuto(false); //인자로 false값을 넣어줌으로서 화면이 자동으로 다시 그려지도록 하는 것을 막는다.

ofBackground(0); // 배경 색을 검정색으로 변하지 않게 설정한다.

}

void ofApp::draw(){

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT)) { //마우스 왼쪽 버튼이 클릭되면 조건문 안으로 들어간다.

float randomColor = ofRandom(50, 255); //ofRandom함수를 이용해서 50부터 255까지 랜덤으로 float형 숫자를 받는다. 0부터 50까지의 색깔은 화면에 거의 표시되지 않기 때문에 제외한다.

ofSetColor(randomColor); // 직사각형의 색깔을 randomColor로 설정한다.

ofSetRectMode(OF\_RECTMODE\_CENTER); // 클릭한 지점을 중앙으로 하는 직사각형을 생성한다. OF\_RECTMODE\_CORNER을 인자로 주면 클릭한 지점을 왼쪽 위 꼭짓점으로 하는 직사각형이 생성된다.

ofDrawRectangle(ofGetMouseX(), ofGetMouseY(), 50, 50); // 마우스로 클릭한 위치에 size 50x50 의 직사각형을 생성한다.

}

}

1 - 4. Bursting Rectangle Brush

마우스 클릭시 여러 개의 작은 직사각형 조각들이 폭발하듯이 그려지는 코드를 작성한다. for loop을 이용해 랜덤한 정보를 같은 여러 사각형을 생성한다.

void ofApp::setup(){

ofSetFrameRate(60); // //frame당 비율을 설정함으로서 프레임의 속도를 60으로 맞춰준다.

ofSetBackgroundAuto(false);

ofBackground(0);

}

void ofApp::draw(){

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

ofSetRectMode(OF\_RECTMODE\_CENTER);

int numRects = 10;

for (int r = 0; r < numRects; r++) { //10번의 loop을 돈다. loop을 도는 횟수가 많아질수록 클릭 시 작은 직사각형의 개수가 많아진다.

ofSetColor(ofRandom(50, 255));

float width = ofRandom(5, 20); // 직사각형의 가로 길이를 random으로 설정한다.

float height = ofRandom(5, 20); // 직사각형의 세로 길이를 random으로 설정한다.

float xOffset = ofRandom(-40, 40); // 직사각형이 생성되는 위치(x좌표)를 random으로 설정한다.

float yOffset = ofRandom(-40, 40); // 직사각형이 생성되는 위치(y좌표)를 random으로 설정한다.

ofDrawRectangle(ofGetMouseX() + xOffset, ofGetMouseY() + yOffset, width, height); // 마우스로 클릭하는 위치에서 xOffset, yOffset만큼 떨어져 있는 위치에 가로 width, 세로 height 만큼의 직사각형을 생성한다.

}

}

}

1 - 5. Growing Circle Brush

원하는 색깔과 투명도를 가진 원을 그려주는 코드를 작성한다. Bursting Rectangle brush 와 거의 유사하나, 도형을 원으로 바꾸고, 투명도와 색깔을 바꿔준다는 점에서 차이가 있다. 코드는 강의자료의 코드와, 링크로 걸려있는 깃허브의 코드를 참고하였다.

void ofApp::draw(){

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

int maxRadius = 100; // for loop을 돌 때 변수의 최댓값으로 사용한다.

int radiusStepSize = 5; // for loop을 돌 때 변수의 step 값으로 사용한다.

int maxOffsetDistance = 100; // 원의 반지름의 최댓값으로 사용한다.

for (int radius = maxRadius; radius > 0; radius -= radiusStepSize) { // maxRadius와 radiusStepSize를 이용해 loop을 얼마나 돌 지 결정한다.

float angle = ofRandom(ofDegToRad(360.0)); // ofDegTorad()함수를 사용해 실제 각도를 radian 으로 변한다. 여기선 360도를 radian으로 변환하여 2pi를 반환한다. 이를 최댓값으로 해서 ofRandom을 통해 random 각도를 받아온다.

float distance = ofRandom(maxOffsetDistance); // 원의 반지름을 최댓값 maxOffsetDistanc로 하여 random으로 받아온다.

float xOffset = cos(angle) \* distance; // 극좌표계를 직교좌표계로 변환하는 과정이다. x좌표를 나타낸다.

float yOffset = sin(angle) \* distance; // 극좌표계를 직교좌표계로 변환하는 과정이다. y좌표를 나타낸다.

ofColor myOrange(255, 132, 0); // myOrange라는 color를 지정한다. 원하는 RGB 값을 입력한다.

ofColor myBlue(0, 0, 255, 50); // myBlue라는 color를 지정한다. 원하는 RGB값과 불투명도 값을 입력한다.

ofColor myGreen(0, 0, 255, 255); // myGreen이라는 color를 지정한다. 원하는 RGB값과 불투명도 값을 입력한다.

// myGreen의 R, G, B, A 값이 맞는지 확인하여 이를 출력한다.

cout << "Red channel" << myGreen.r << endl;

cout << "Green channel" << myGreen.g << endl;

cout << "Blue channel" << myGreen.b << endl;

cout << "Alpha channel" << myGreen.a << endl;

// myYellow라는 color를 지저한다. 원하는 RGB값과 불투명도 값을 입력한다. 이처럼 먼저 선언 후 나중에 값을 넣어줄 수도 있다.

ofColor myYellow;

myYellow.r = 255; myYellow.b = 0; myYellow.g = 255; myYellow.a = 255;

// 사용자가 지정한 색이 아닌, 원래 of에 만들어져 있는 색을 받아올 수도 있다.

ofColor myAqua = ofColor::aqua;

ofColor myPurple = ofColor::plum;

ofColor inBetween = myOrange.getLerped(myBlue, ofRandom(1.0)); // getLerped 함수를 사용해 두 색(myOrange, myBlue) 사이의 임의의 색을 받아온다. ofRandom(1.0)의 값에 따라 어느 색에 가까운지 정해진다.

ofSetColor(inBetween); // 도형의 색을 inBetween으로 설정한다.

ofDrawCircle(ofGetMouseX() + xOffset, ofGetMouseY() + yOffset, radius); // 랜덤으로 원을 그린다. 위치는 위에서 random으로 받은 변수들을 이용하였고, radius는 현재 for loop을 돌고 있는 변수를 이용하였다.

}

}

}

1 - 6. Fleeing Triangle Brush

Bursting rectangle brush와 거의 유사하나, 도형이 직사각형에서 삼각형으로 바뀌고 벡터를 이용하여 화면에 그림을 표시하는 코드를 구현한다.

void ofApp::draw(){

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

int numTriangles = 10; // for loop 을 돌 때 변수의 최댓값으로 설정한다.

int minOffset = 5; // 도형의 위치벡터를 만들 때 최솟값으로 사용한다.

int maxOffset = 70; // 도형의 위치벡터를 만들 때 최댓값으로 설정ㅎ나다.

int alpha = 150; // 색의 불투명도를 사용하는 변수이다.

for (int t = 0; t < numTriangles; ++t) { //원하는 만큼 loop을 돈다.

float offsetDistance = ofRandom(minOffset, maxOffset); // 위치벡터의 크기를 random으로 받는다.

ofVec2f mousePos(ofGetMouseX(), ofGetMouseY()); // 마우스 클릭 시 위치(x, y)를 계산을 위해 ofVec2f로 받는다.

ofVec2f p1(0, 6.25); // (0,0)에서 (0, 6.25)로 가는 vector

ofVec2f p2(25, 0); // (0, 0)에서 (25, 0)으로 가는 vector

ofVec2f p3(0, -6.25); // (0, 0)에서 (0, -6.25)로 가는 vector

float rotation = ofRandom(360); // 도형의 회전을 표현하는 변수 rotation을 설정하고 0부터 360 까지의 숫자를 랜덤으로 받는다. 여기선 radian이 아닌 실제 각도를 이용한다.

p1.rotate(rotation); // p1에 rotation만큼의 회전을 준다.

p2.rotate(rotation); // p2에 rotation만큼의 회전을 준다.

p3.rotate(rotation); // p3에 rotation만큼의 회전을 준다.

ofVec2f triangleOffset(offsetDistance, 0.0); // (0, 0)에서 (offsetDistance, 0.0)으로 가는 vector

triangleOffset.rotate(rotation); // triangleOffset에 rotation만큼의 회전을 준다.

p1 += mousePos + triangleOffset; // 마우스 클릭 위치와 triangleOffset의 위치를 더해서 p1값에 더한다.

p2 += mousePos + triangleOffset; // 마우스 클릭 위치와 triangleOffset의 위치를 더해서 p2값에 더한다.

p3 += mousePos + triangleOffset; // 마우스 클릭 위치와 triangleOffset의 위치를 더해서 p3값에 더한다.

ofColor aqua(0, 252, 255, alpha); // aqua라는 색 지정

ofColor purple(198, 0, 205, alpha); // purple라는 색 지정

ofColor inbetween = aqua.getLerped(purple, ofRandom(1.0)); // aqua부터 purple 사이의 임의의 색을 받아온다.

ofSetColor(inbetween); // 도형의 색은 inbetween.

ofDrawTriangle(p1, p2, p3); // p1, p2, p3를 꼭짓점으로 하는 삼각형을 그린다.

}

}

}

1 - 7. update()

일정 시간마다 화면을 update한다. frame마다 변하는 도형의 위치를 화면에 출력한다.

int xPos; // ofApp.h의 멤버 변수로 xPos 를 선언한다. 점의 x좌표를 나타낸다.

void ofApp::setup(){

xPos = 5; // 점의 x좌표의 초기값을 5로 지정한다.

ofBackground(ofColor::black); // 배경은 검정색으로 설정한다.

}

void ofApp::update(){ // frame당 한 번씩 호출된다.

xPos += 2; // x좌표를 2씩 키운다. 화면상으로는 계속 오른쪽으로 가는 모습을 확인할 수 있다.

if (ofGetWidth() < xPos) {

xPos = 5; // xPos가 계속 커져서 화면의 가로 길이보다 커진다면 초기값이었던 5로 재설정 해준다. 화면상으로는 다시 맨 왼쪽에서 시작하는 모습을 확인할 수 있다.

}

}

void ofApp::draw(){

ofSetColor(ofColor::red); // 점은 빨간색으로 설정한다.

ofDrawCircle(xPos, 100, 10); // (xPos, 100)위치에 반지름 10만큼의 원을 그린다.

}