6주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231515 이름: 김다은

**1-1. Basic shape(1)**

ofBackground(0); // 화면의 배경 색을 검정색으로 지정한다.

// 화면의 배경 색깔을 지정하는 함수이다. 0은 검정색을 의미하며, argument로 RGB 색상값을 받는다. 위와 같이 1개의 argument만이 입력된 경우 R, G, B 값이 모두 해당 숫자인 것으로 처리된다.

ofSetColor(255); // 이후 그릴 때 색상을 흰색으로 지정한다.

// 이후 그릴 물체의 색상을 설정하는 함수이다. ofBackground 함수와 마찬가지로 RGB 값을 argument로 받는다. 해당 코드의 경우 R: 255. G: 255, B: 255인 색깔인 흰색으로 처리된다.

ofDrawRectangle(50, 50, 100, 100); // 좌측 상단의 좌표가 (50, 50)이고 가로, 세로의 길이가 100인 직사각형을 그린다.

// argument로 위치와 직사각형의 크기를 입력받아 직사각형을 그리는 함수이다. arg1과 arg2는 직사각형의 좌측 상단 꼭짓점의 x좌표와 y좌표이다. 또한 arg3은 좌측 상단의 점으로부터의 가로 길이를 의미한다. arg4는 좌측 상단의 점으로부터 직사각형의 세로 길이를 전달한다. 이때 단위는 pixel이다.

ofDrawCircle(250, 100, 50); // (250, 100)을 중심으로 하고 반지름이 50인 원을 그린다.

// argument로 원의 중심과 반지름을 입력 받아 원을 그리는 함수이다. arg1과 arg2는 각각 원의 중심의 x, y좌표를 의미한다. arg2는 반지름의 길이를 뜻한다.

ofDrawEllipse(400, 100, 80, 100); // 중심이 (400, 100)이고 가로, 세로의 길이가 각각 80, 100인 타원을 그린다.

// argument로 타원의 중심 좌표와 가로, 세로 길이를 입력 받아 타원을 그리는 함수이다. arg1, arg2는 각각 타원의 중심의 x, y좌표를 뜻한다. arg3은 타원의 가로 길이이고, arg4는 타원의 세로 길이를 뜻한다.

ofDrawTriangle(500, 150, 550, 50, 600, 150); // 세 꼭짓점의 좌표가 (500, 150), (550, 50), (600, 150)인 삼각형을 그린다.

// 삼각형의 세 꼭짓점의 좌표를 입력받아 삼각형을 그리는 함수이다. arg1, arg2는 첫번째 꼭짓점의 x, y 좌표를 의미한다. 그 뒤로 arg3과 arg4, arg5와 arg6은 두번째, 세번째 꼭짓점의 x, y좌표를 나타낸다.

ofDrawLine(700, 50, 700, 150); // 두 좌표 (700, 50), (700, 150)을 잇는 선분을 그린다.

// 한 선분의 양 끝점의 좌표를 입력 받아 선을 그리는 함수이다. arg1, arg2는 첫번째 점의 x, y좌표를 의미한다. arg3, arg4는 두 번째 점의 x, y 좌표를 의미한다.

**1-2. basic Shapes (2)**

ofBackground(0); // 배경의 색깔을 검정으로 한다.

ofSetColor(255); // 이후 그리는 색을 흰색으로 한다.

ofFill(); // 이후 그릴 때, 도형 안의 색도 칠하는 것으로 한다.

ofDrawRectangle(50, 50, 100, 100); // (50,50)을 좌측 상단의 꼭짓점으로 하고, 가로, 세로의 길이가 100인 직사각형을 그린다.

ofDrawCircle(250, 100, 50); // (250, 100)을 중심의 좌표로 하는 반지름 50인 원을 그린다.

ofDrawEllipse(400, 100, 80, 100); // (400, 100)을 중심의 좌표로 하고, 가로 길이와 세로 길이가 각각 80, 100인 타원을 그린다.

ofDrawTriangle(500, 150, 550, 50, 600, 150); // 세 꼭짓점의 좌표가 (500, 150), (550, 50), (600, 150)인 삼각형을 그린다.

ofDrawLine(700, 50, 700, 150); // (700, 50)과 (700, 150)인 점을 잇는 선분을 그린다.

ofNoFill(); // 이후 그릴 때, 도형 내부에 색을 채우지 않는다.

ofDrawRectangle(50, 200, 100, 100); // (50, 200)을 좌측 상단의 꼭짓점으로 하고, 가로, 세로의 길이가 100인 직사각형을 그린다.

ofDrawCircle(250, 250, 50); // (250, 250)을 원의 중심으로 하고 반지름의 길이가 50인 원을 그린다.

ofDrawEllipse(400, 250, 80, 100); // 중심이 (400, 250)이고 가로, 세로의 길이가 80, 100인 타원을 그린다.

ofDrawTriangle(500, 300, 550, 200, 600, 300); // 세 꼭짓점의 좌표가 (500, 300), (550, 200), (600, 300)인 삼각형을 그린다.

ofDrawLine(700, 200, 700, 300); // 두 좌표 (700, 200), (700, 300)을 잇는 선분을 그린다.

**1-3. Single Rectangle Brush**

// draw 함수: 시작된 이후 계속 반복되는 함수이다. update()와는 다르게 화면에 원이나 사각형 등 뭔가를 그리는 역할을 한다. 다시 말해 프로그램의 그래픽 요소를 렌더링하는 역할을 한다.

void ofApp::draw(){

// ofGetMousePressed 함수: 마우스의 버튼이나 휠이 눌렸는지 여부를 반환한다.

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT)) { // 마우스의 왼쪽 버튼이 눌린 경우에

// ofRandom 함수: arg1과 arg2의 범위 사이의 값에서 무작위로 값을 리턴한다.

// 50~255 범위의 값을 랜덤으로 산출하여 randomColor에 저장한다.  
float randomColor = ofRandom(50, 255);

// ofSetColor 함수: 이후 그릴 물체의 색을 지정한다.

// RGB 값을 randomColor 변수의 값으로 하는 색상으로 이후 그릴 물체의 색을 설정한다.

ofSetColor(randomColor);

// ofSetRectMode 함수: 그릴 직사각형의 기준을 설정하는 함수이다.  
ofSetRectMode(OF\_RECTMODE\_CENTER); // 직사각형의 중심을 기준으로 설정한다.

// ofGetMouseX, ofGetMouseY 함수: 각각 마우스 위치의 x, y좌표를 리턴한다.

ofDrawRectangle(ofGetMouseX(), ofGetMouseY(), 50, 50); // arg1, arg2를 각각 기준점의 x, y좌표로 하고 가로 세로의 길이가 50인 직사각형을 그린다.

}

}

// setup 함수: 프로그램이 시작되었을 때 한번 호출되는 함수이다. 배경색이나 윈도우 크기, 프레임 속도를 설정가능하다.

void ofApp::setup(){

// 화면이 갱신되는 과정에서 배경 상태가 유지되도록 한다.

// 만약 아래 코드가 없는 경우 ofClear()함수가 호출될 때마다 배경이 자동으로 설정된 색상으로 초기화된다.  
ofSetBackgroundAuto(false);  
ofBackground(0); // 화면의 배경 색깔을 검정색으로 한다.

}

**1-4. Bursting Rectangle Brush**

void ofApp::draw(){

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT)) { // 마우스의 왼쪽 버튼이 눌린 경우

ofSetRectMode(OF\_RECTMODE\_CENTER); // 직사각형을 그릴 때, 직사각형의 중심을 기준으로 그린다.  
int numRects = 10; // 그릴 사각형의 개수를 10으로 한다.  
for (int r = 0; r < numRects; r++) { // 10번(numRects) 반복한다.

ofSetColor(ofRandom(50, 255)); // 50~255 사이의 랜덤 값으로 색상을 지정한다.  
float width = ofRandom(5, 20); // 너비를 5~20사이의 랜덤 값으로 지정한다.  
float height = ofRandom(5, 20); // 높이를 5~20 사이의 랜덤 값으로 지정한다.  
float xOffset = ofRandom(-40, 40); // x좌표 시작점을 -40~40 사이의 랜덤 값으로 지정한다.  
float yOffset = ofRandom(-40, 40); // y좌표 시작점을 -40~40 사이의 랜덤 값으로 지정한다.  
ofDrawRectangle(ofGetMouseX() + xOffset, ofGetMouseY() + yOffset, width, height);

// 현재 마우스 위치에서 각각 xOffset, yOffset만큼 떨어진 위치에 width, height의 크기를 가진 직사각형을 그린다.

}

}

}

void ofApp::setup(){

ofSetFrameRate(60); // 프레임을 1초당 60으로 제한한다.

}

**1-5. Glowing Circle Brush**

void ofApp::setup(){

ofSetFrameRate(60); // 프레임을 1초당 60으로 제한한다.

ofSetBackgroundAuto(false); // 화면이 갱신될 때 배경이 매번 초기화되지 않도록 한다.

ofBackground(0); // 배경 색깔을 검정으로 지정한다.

}

void ofApp::draw(){

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT)) { // 마우스 왼쪽 버튼이 눌린 경우

int maxRadius = 100; // 그릴 원의 최대 반지름을 100으로 한다.

int radiusStepSize = 5; // 줄어드는 반지름의 크기를 5로 한다.

int alpha = 3; // 투명도를 3으로 한다.

int maxOffsetDistance = 100; // 마우스 위치로부터 최대로 떨어져 있을 수 있는 거리를 100으로 한다.

for (int radius=maxRadius; radius>0; radius-=radiusStepSize) { // 최대 반지름에서부터 반지름을 radiusStepSize만큼 줄여가면서 radius가 0보다 클 때까지 반복한다.

float angle = ofRandom(ofDegToRad(360.0)); // 0의 라디안 값과 260의 라디안 값 사이의 랜덤 값을 angle에 저장한다. 이때 ofDegToRad는 각도를 라디안값으로 변환해준다.

float distance = ofRandom(maxOffsetDistance); // 0~maxOffsetDistance 사이의 랜덤 값을 저장한다.

float xOffset = cos(angle) \* distance; // 마우스에서 떨어진 길이에 cos값을 곱하여 x 좌표의 관점에서 얼마나 떨어져 있는지를 저장한다.

float yOffset = sin(angle) \* distance; // 마우스에서 떨어진 길이에 sin값을 곱하여 y 좌표의 관점에서 얼마나 떨어져 있는지를 저장한다.

// Using the ofColor class, we will randomly select a color between orange and red

ofColor myOrange(255, 132, 0, alpha); // R: 255, G: 132, B:0 투명도: alpha를 가지는 색을 myOrange에 저장한다.

ofColor myRed(255, 6, 0, alpha); // R: 255, G: 6, B: 0. 투명도: alpha를 가지는 색을 myRed에 저장한다.

ofColor inBetween = myOrange.getLerped(myRed, ofRandom(1.0)); // getLerped 함수를 이용해 myRed와 myOrange를 랜덤하게 섞은 값을 저장한다.

ofSetColor(inBetween); // 이후 그릴 색을 inBetween으로 지정한다.

ofDrawCircle(ofGetMouseX()+xOffset, ofGetMouseY()+yOffset, radius);

// 마우스 커서에서 랜덤한 길이로 떨어진 좌표에 radius의 반지름을 가지는 원을 그린다.

}

}

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_RIGHT)) { // 마우스의 오른쪽 버튼이 눌린 경우

ofBackground(0); // 화면을 모두 검정색으로 초기화한다.

}

}

**1-6. Fleeing Triangle Brush**

void ofApp::setup(){

ofSetWindowShape(1024,768); // 프로그램 화면의 크기를 1024 \* 768로 설정한다.

ofSetFrameRate(60); // 화면의 프레임을 1초당 60으로 제한한다.

ofSetBackgroundAuto(false); // 초기화될 때마다 화면이 지워지는 것을 막는다.

ofBackground(0); // 배경 색을 검정으로 한다.

}

void ofApp::draw(){

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT)) { // 마우스 왼쪽 버튼이 눌린 경우

int numTriangles = 10; // 그릴 삼각형의 개수를 10으로 한다.

int minOffset = 5; // 마우스 위치로부터 떨어진 최소 길이를 5로 한다.

int maxOffset = 70; // 마우스 위치로부터 떨어진 최대 길이를 70으로 한다.

int alpha = 150; // 투명도를 150으로 한다.

for (int t=0; t<numTriangles; ++t) { // 그릴 삼각형의 개수만큼 반복한다.

float offsetDistance = ofRandom(minOffset, maxOffset); // minOffset~ maxOffset 사이의 랜덤 값을 저장한다.

ofVec2f mousePos(ofGetMouseX(), ofGetMouseY()); // 2차원 백터를 선언하고 현재 마우스의 위치를 저장한다.

ofVec2f p1(0, 6.25); // 2차원 백터 p1에 삼각형의 한 꼭짓점의 좌표를 저장한다.

ofVec2f p2(25, 0); // 2차원 백터 p2에 삼각형의 한 꼭짓점의 좌표를 저장한다.

ofVec2f p3(0, -6.25); // 2차원 백터 p3에 삼각형의 한 꼭짓점의 좌표를 저장한다.

float rotation = ofRandom(360); // 0~360 사이의 랜덤값을 저장한다.

p1.rotate(rotation); // rotation 각도 만큼 p1을 회전한다.

p2.rotate(rotation); // rotation 각도 만큼 p2을 회전한다.

p3.rotate(rotation); // rotation 각도 만큼 p3을 회전한다.

ofVec2f triangleOffset(offsetDistance, 0.0); // 2차원 벡터를 선언하고 아까 구한 offsetDistance를 저장한다.

triangleOffset.rotate(rotation); // 거리를 저장한 벡터도 rotation 각도만큼 회전한다.

p1 += mousePos + triangleOffset; // p1 좌표에 마우스위 위치와 삼각형이 떨어져 있는 거리를 더해준다.

p2 += mousePos + triangleOffset; // p2 좌표에 마우스위 위치와 삼각형이 떨어져 있는 거리를 더해준다.

p3 += mousePos + triangleOffset; // p3 좌표에 마우스위 위치와 삼각형이 떨어져 있는 거리를 더해준다.

ofColor aqua(0, 252, 255, alpha); // RGB값을 (0, 252, 255)로 가지고 투명도가 alpha인 색을 저장한다.

ofColor purple(198, 0, 205, alpha); // RGB값을 (198, 0, 205)로 가지고 투명도가 alpha인 색을 저장한다.

ofColor inbetween = aqua.getLerped(purple, ofRandom(1.0)); // getLeperd 메서드를 통해 aqua와 purple의 색을 섞어준 뒤 값을 저장한다.

ofSetColor(inbetween); // 앞으로 그릴 도형의 색을 inbetween으로 지정한다.

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

}

}

if (ofGetMousePressed(OF\_MOUSE\_BUTTON\_RIGHT)) { // 마우스의 오른쪽 버튼이 눌린 경우

ofBackground(0); // 화면을 모두 검정색으로 초기화한다.

}

}

**1-7. Update 함수**

프로그램이 시작된 이후 계속 반복 호출되는 함수이다. 보통 변수 값의 변화를 업데이트한다. 사용자 입력을 처리하거나 애니메이션 및 모션 처리를 할 수 있는 함수이다.

`ofApp.h`의 클래스의 멤버 변수로 int xPos;가 선언되어 있다고 하자.

void ofApp::setup(){

xPos = 5; // xPos를 5로 초기화한다.

ofBackground(ofColor::black); // 화면의 배경 색상을 검정으로 한다.

}

void ofApp::update(){

xPos += 2; // update 함수가 호출될 때마다(매 프레임마다) xPos를 2만큼 증가시킨다.

if(ofGetWidth()<xPos) { // 프로그램 창의 너비가 xPos보다 작은 경우

xPos = 5; // xPos를 5로 다시 초기화한다.

}

}

void ofApp::draw() {

ofSetColor(ofColor::red); // 이후 그릴 도형의 색을 빨강으로 지정한다.

ofDrawCircle(xPos, 100, 10); // (xPos, 100)을 중심으로 하는 반지름 10인 원을 그린다.

}