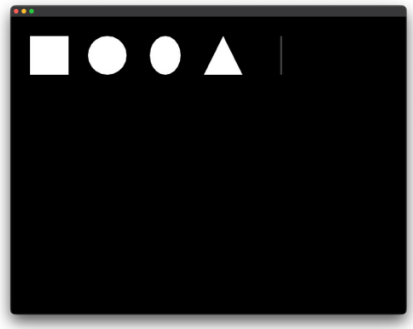
6주차 결과보고서

전공 : 경영학과 학년 : 4학년 학번 : 20190808 이름 : 방지혁

1. Basic shape (1), 함수

```
//--------------------------------------------------------------
void ofApp::draw(){
    ofBackground(0); // Clear the screen with a black color, 0 means completely white color in grayscale color.
    ofSetColor(255); // Set the drawing color to white.
    // Draw some shapes below.
    ofDrawRectangle(50, 50, 100, 100); // Top left corner at (50, 50), 100 wide x 100 high.
    ofDrawCircle(250, 100, 50); // Centered at (250, 100), radius of 50.
    ofDrawEllipse(400, 100, 80, 100); // Centered at (400 100), 80 wide x 100 high.
    ofDrawTriangle(500, 150, 550, 50, 600, 150); // Three corners: (500, 150), (550, 50), (600, 150).
    ofDrawLine(700, 50, 700, 150); // Line from (700, 50) to (700, 150).
}
```

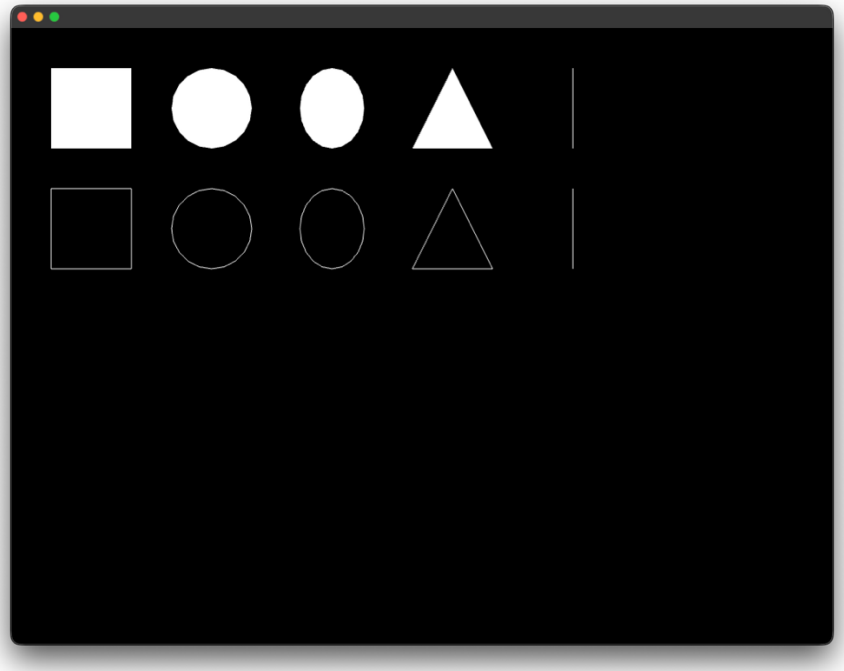
우선 첫번째 줄은 화면 배경색을 설정하는 함수 ofBackground입니다. 인자값으로 들어가는 0은 그레이 스케일 값으로 0이니 검은색입니다. 이 함수를 통해 화면을 검정색으로 설정하는 것입니다. 두번째 줄은 ofSetColor함수로 이후 그릴 도형들의 색상을 설정하는 함수입니다. 앞선 말했듯 0은 검정색이기에 255는 흰색이 될 것이며 도형이 흰색으로 나타나는 것을 기대할 수 있겠습니다. 세번째 줄은 ofDrawRectangle로 사각형을 그리는 함수입니다. 인자는 순서대로 가장 왼쪽 위의 x좌표, y좌표, 가로 길이, 세로 길이입니다. 네번째 줄은 ofDrawCircle로 원을 그리는 함수입니다. 인자는 순서대로 원의 중심 x좌표, y좌표, 반지름입니다. 다섯번째 줄은 ofDrawEllipse로 타원을 그리는 함수입니다. 인자는 순서대로 타원의 중심 x좌표, y좌표, 가로, 세로입니다. 여섯 번째는 ofDrawTriangle로 삼각형을 그리는 함수입니다. 인자는 순서대로 왼쪽 아래 꼭짓점의 x 및 y좌표, 위 꼭짓점의 x 및 y좌표, 오른쪽 아래 꼭짓점의 x 및 y 좌표입니다. 마지막은 ofDrawLine으로 직선을 그리는 함수입니다. 인자는 순서대로 시작점의 x 및 y 좌표, 끝나는 점의 x 및 y 좌표입니다. 결과는 이렇습니다.



2. Basic shape (2)

```
void ofApp::draw(){
    ofBackground(0); // Clear the screen with a black color, 0 means completely white color in grayscale color.
    ofSetColor(255); // Set the drawing color to white.

    ofFill(); // If we omit this and leave ofNoFill(), all the shapes will be outlines!
    // Draw some shapes below.
    ofDrawRectangle(50, 50, 100, 100); // Top left corner at (50, 50), 100 wide x 100 high.
    ofDrawCircle(250, 100, 50); // Centered at (250, 100), radius of 50.
    ofDrawEllipse(400, 100, 80, 100); // Centered at (400 100), 80 wide x 100 high.
    ofDrawTriangle(500, 150, 550, 50, 600, 150); // Three corners: (500, 150), (550, 50), (600, 150).
    ofDrawLine(700, 50, 700, 150); // Line from (700, 50) to (700, 150).

    ofNoFill(); // If we omit this and leave ofFill(), all the shapes will be filled!
    // Draw some shapes below
    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);
}
```

첫 두 줄은 이전과 같기에 설명은 생략하겠습니다. ofFill은 새로 추가된 함수로 모든 도형이 계속 채워진채로 그려지게 됩니다. 사실 default가 ofFill이여서 1번처럼 ofFill을 하지 않아도 채워져서 나온 것입니다. 이후 5줄은 이전과 같습니다. ofNofill함수는 도형을 그릴때 채우지 않고 외곽선만 그리도록 설정하는 함수입니다. 이후 다음 5줄은 이전과 같지만 그리는 위치만 다릅니다. 참고로 선은 채우기 개념이 없어서 ofFill()과 상관이 없습니다. 결과는 다음과 같습니다.

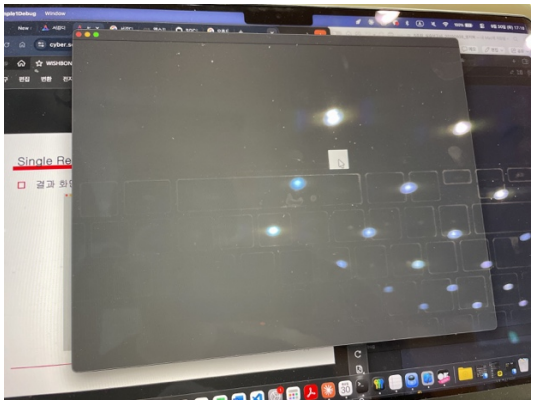


3. Single rectangle brush

처음 추가할 코드는 다음과 같습니다.

```
void ofApp::draw(){
    if (ofGetMousePressed(OF_MOUSE_BUTTON_LEFT)) { // If the left mouse button is pressed...
        ofSetColor(255);
        ofSetRectMode(OF_RECTMODE_CENTER);
        ofDrawRectangle(ofGetMouseX(), ofGetMouseY(), 50, 50);
        // Draw a 50 x 50 rect centered over the mouse
    }
}
```

ofGetMousePressed(OF_MOUSE_BUTTON_LEFT)는 왼쪽 마우스 버튼이 눌려있는지 체크하는 함수이며 이를 통해 왼쪽 마우스가 눌렸는지 확인하고 다음 코드를 수행합니다. ofSetColor(255)는 그리기 색상을 흰색으로 설정하고 ofSetRectMode(OF_RECTMODE_CENTER)는 사각형 그리는 방식을 사각형의 중심을 기준으로 그리도록 설정합니다. 기본은 왼쪽 위 모서리입니다. 이후 ofDrawRectangle(ofGetMouseX(), ofGetMouseY(), 50, 50)는 마우스의 x좌표, y좌표를 가져와 이 위치를 중심으로 50x50 크기의 흰색 사각형을 그립니다. 실행 결과는 다음과 같으며 마우스가 꾹 눌려있을 때만 사각형이 그려지는 것을 확인할 수 있습니다.

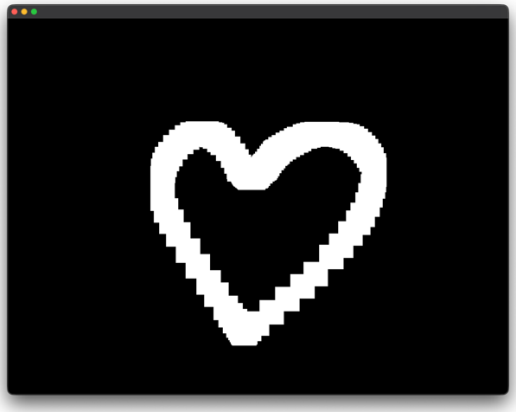


금새 사라지는 것을 방지하기 위해 setup 함수에 다음과 같이 추가해줍니다.

```
void ofApp::setup(){
    ofSetBackgroundAuto(false);
    // We still want to draw on a black background, so we need to draw
    // the background before we do anything with the brush
    ofBackground(0);
}
```

ofSetBackgroundAuto(false)는 배경을 자동으로 지우는 것을 끄는 것입니다. 기존에는 default가 true라 화면을 완전히 지우고 draw()함수를 실행하고 새로 그리는 것을 매번 반복했지만 false로

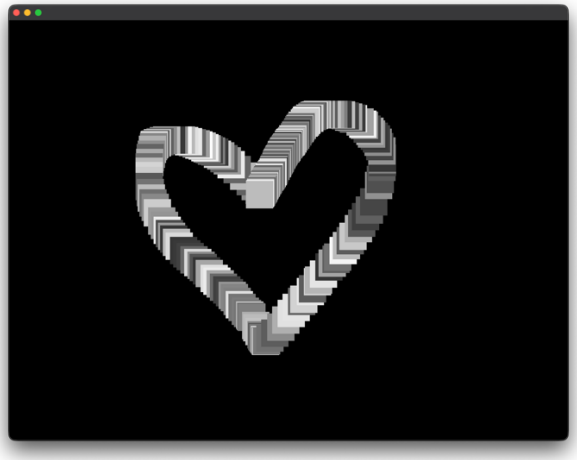
설정해서 이전에 그린 것들이 계속 남아있게 됩니다. ofBackground(0)은 검은 배경을 한 번만 그린다는 뜻입니다. Setup()함수 내부에서 호출하기 때문에 프로그램을 시작할 때 딱 한번만 검정색으로 칠합니다. 결과는 다음과 같습니다. 사각형이 안 사라지는 것을 목격할 수 있습니다.



이후 draw함수를 다음과 같이 수정합니다.

```
void ofApp::draw(){
    if (ofGetMousePressed(OF_MOUSE_BUTTON_LEFT)) { // If the left mouse button is pressed...
        float randomColor = ofRandom(50, 255);
        ofSetColor(randomColor);
        // Exclude dark grayscale values (0 - 50) that won't show on black background
        ofSetRectMode(OF_RECTMODE_CENTER);
        ofDrawRectangle(ofGetMouseX(), ofGetMouseY(), 50, 50);
        // Draw a 50 x 50 rect centered over the mouse
    }
}
```

새로 추가된 2줄이 존재합니다. float randomColor = ofRandom(50, 255)는 50부터 255사이의 랜덤한 숫자를 만들어서 randomColor 변수에 저장합니다. 그리고 이 변수(랜덤 색상)으로 그리기 색상을 설정하는 것이 ofSetColor(randomColor)입니다. 최종적인 결과는 다음과 같습니다.



4. bursting rectangle brush

```
void ofApp::setup(){
    ofSetFrameRate(60);
    ofSetBackgroundAuto(false);
    ofBackground(0);
}
```

ofSetFramRate함수는 frame rate를 설정하는 함수입니다. 1초에 60번 draw()함수를 실행하겠다고 설정하는 것으로 우리가 게임에서 사양을 말할 때 그 프레임과 같습니다. 그 외 setup()함수의 코드는 기존과 같습니다.

```
void ofApp::draw(){
    if (ofGetMousePressed(OF_MOUSE_BUTTON_LEFT)) { // If the left mouse button is pressed...
        ofSetRectMode(OF_RECTMODE_CENTER);
        int numRects = 10;
        for (int r=0; r<numRects; r++) {
                ofSetColor(ofRandom(50, 255));
            float width = ofRandom(5, 20);
            float height = ofRandom(5, 20);
            float xOffset = ofRandom(-40, 40);
            float yOffset = ofRandom(-40, 40);
            ofDrawRectangle(ofGetMouseX()+xOffset, ofGetMouseY()+yOffset, width, height);
        }
    }
}
```

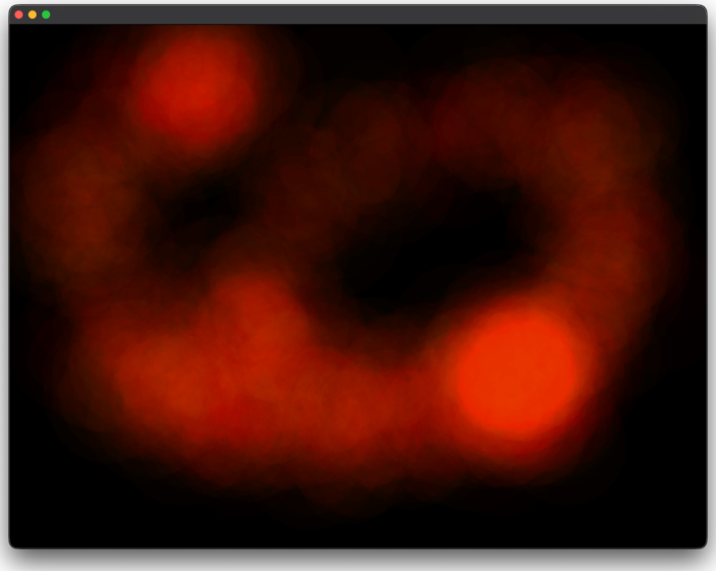
이전에 설명했다시피 왼쪽 마우스 버튼이 눌려있으면 코드를 실행하는 것입니다. ofSetRectMode(OF_RECTMODE_CENTER)는 사각형을 그릴 때 중심점을 기준으로 그리도록 하겠다는 것입니다. for(int r=0; r<numRects; r++)안에서 반복문을 실행하도록 했는데 여기서 numRects는 한 번에 몇 개의 사각형을 그릴지 설정하는 변수로 이 값이 높아질수록 더 촘촘하게 보이게 될 것입니다. 이후 반복문 안에선 color – 매 반복마다 랜덤한 밝기의 색상, width – 가로 크기를 5픽셀부터 20 픽셀 사이의 랜덤한 값, height – 세로 크기를 5픽셀부터 20 픽셀 사이의 랜덤한 값, xOffset – 기존 마우스 위치에서 가로 방향으로 얼마나 멀리 떨어뜨릴지, yOffset - 기존 마우스 위치에서 세로 방향으로 얼마나 멀리 떨어뜨릴지를 설정하는 코드가 5줄 있습니다. 이후 이 변수들을 인자로 삼아 ofDrawRectangle 즉 사각형을 그립니다. 결과는 다음과 같습니다.

5. glowing circle brush

```
void ofApp::draw(){
    // If the left mouse button is pressed...
    if (ofGetMousePressed(OF_MOUSE_BUTTON_LEFT)) {
        int maxRadius = 100;  // Increase for a wider brush
        int radiusStepSize = 5;  // Decrease for more circles (i.e. a more opaque brush)
        int alpha = 3;  // Increase for a more opaque brush
        int maxOffsetDistance = 100;  // Increase for a larger spread of circles
        for (int radius=maxRadius; radius>0; radius-=radiusStepSize) {

            // Formula for converting from polar to Cartesian coordinates:
            //    x = cos(polar angle) * (polar distance)
            //    y = sin(polar angle) * (polar distance)
            // We need our angle to be in radians if we want to use sin() or cos()
            // so we can make use of an openFrameworks function to convert from degrees
            // to radians
            float angle = ofRandom(ofDegToRad(360.0));
            float distance = ofRandom(maxOffsetDistance);
            float xOffset = cos(angle) * distance;
            float yOffset = sin(angle) * distance;

            // Using the ofColor class, we will randomly select a color between orange and red
            ofColor myOrange(255, 132, 0, alpha);
            ofColor myRed(255, 6, 0, alpha);
            ofColor inBetween = myOrange.getLerped(myRed, ofRandom(1.0));
            ofSetColor(inBetween);

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

    // If the right mouse button is pressed...
    if (ofGetMousePressed(OF_MOUSE_BUTTON_RIGHT)) {
        ofBackground(0);  // Erase the screen with a black background
    }
}
```

우선 아랫부분에 있는 if (ofGetMousePressed(OF_MOUSE_BUTTON_RIGHT))는 오른쪽 마우스 버튼이 눌려 있는지 체크하는 부분으로 만약 그러한 경우 ofBackground(0)으로 모든 그림을 지워버리는 것입니다. 윗부분부터 차근차근 설명하자면 앞서 말했듯 왼쪽 마우스 버튼이 눌렸을 때 다음과 같은 코드를 실행합니다. Int maxRadius = 100는 최대 반지름이 100 픽셀, int radiusStepSize = 5는 반지름을 최대 반지름에서부터 얼만큼 줄여 나갈지 정하는 것입니다. 만약 값을 작게 하면 더 많은 원이 그려져 더 부드럽게 보일 것입니다. Int alpha = 3은 투명도를 설정하는 코드입니다. Int maxOffsetDistance = 100는 원을 마우스로부터 얼마만큼 멀리 떨어질 수 있는지 설정하는 코드입니다. 이후 반복문은 for (int radius=maxRadius; radius>0; radius-=radiusStepSize)은 앞서 언급한 maxRadius와 radiusStepSize를 사용하는 반복문으로 바깥쪽 큰 원부터 radiusStepSize만큼 줄여나가며 안쪽 작은 원까지 0보다 클 동안 그려나가는 것입니다. 다음 4줄은 랜덤하게 각도, 마우스로부터의 거리, x좌표, y좌표 오프셋 값을 설정하는 것입니다. 다음 3줄은 주황색과 빨간색을 정

의한 다음 해당 두 색깔 사이의 색상을 랜덤하게 선택하는 것입니다. 그리고 앞서 말한 랜덤 변수들을 이용해 원을 그립니다. 결과는 다음과 같습니다.



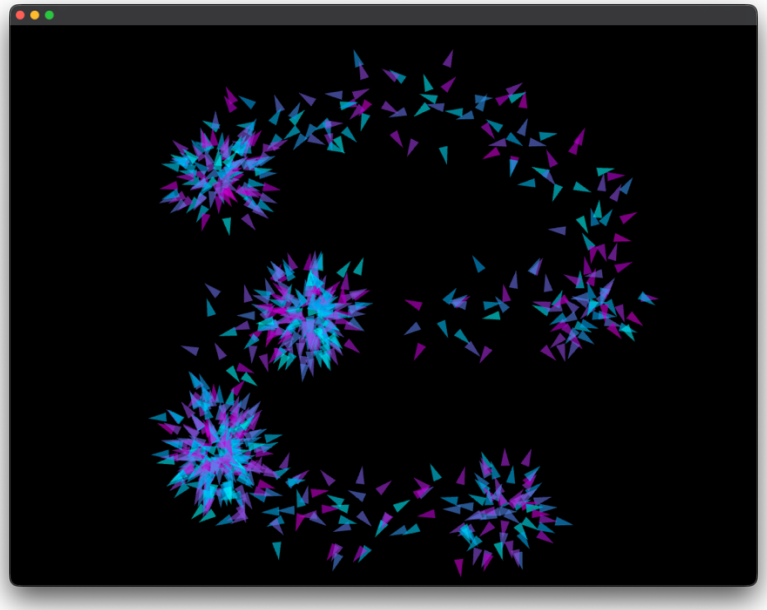
이어서 다음 장에

6. Fleeing triangle brush

```
void ofApp::draw(){
    // If the left mouse button is pressed...
    if (ofGetMousePressed(OF_MOUSE_BUTTON_LEFT)) {

        // Code for the final version of the brush

        int numTriangles = 10;
        int minOffset = 5;
        int maxOffset = 70;
        int alpha = 150;

        for (int t=0; t<numTriangles; ++t) {
            float offsetDistance = ofRandom(minOffset, maxOffset);

            ofVec2f mousePos(ofGetMouseX(), ofGetMouseY());

            // Define a triangle at the origin (0,0) that points to the right
            ofVec2f p1(0, 6.25);
            ofVec2f p2(25, 0);
            ofVec2f p3(0, -6.25);

            float rotation = ofRandom(360); // The rotate function uses degrees!
            p1.rotate(rotation);
            p2.rotate(rotation);
            p3.rotate(rotation);

            ofVec2f triangleOffset(offsetDistance, 0.0);
            triangleOffset.rotate(rotation);

            p1 += mousePos + triangleOffset;
            p2 += mousePos + triangleOffset;
            p3 += mousePos + triangleOffset;

            ofColor aqua(0, 252, 255, alpha);
            ofColor purple(198, 0, 205, alpha);
            ofColor inbetween = aqua.getLerped(purple, ofRandom(1.0));
            ofSetColor(inbetween);

            ofDrawTriangle(p1, p2, p3);
        }

    }

    // If the right mouse button is pressed...
    if (ofGetMousePressed(OF_MOUSE_BUTTON_RIGHT)) {
        ofBackground(0);  // Erase the screen with a black background
    }
}
```

이전의 항목과 같이 아랫부분에 있는 if (ofGetMousePressed(OF_MOUSE_BUTTON_RIGHT))는 오른쪽 마우스 버튼이 눌려 있는지 체크하는 부분으로 만약 그러한 경우 ofBackground(0)으로 모든 그림을 지워버리는 것입니다. 윗부분부터 차근차근 설명하자면 왼쪽 마우스 버튼이 눌러있는 경우 다음과 같은 코드들을 수행합니다. Int numTriangles = 10; 은 한 번에 10개의 삼각형을 그릴 것이라는 뜻입니다. Int minOffset=5는 마우스로부터 최소 5픽셀은 떨어지게 해서 삼각형이 그려지는 것입니다. Int maxOffset = 70은 아까와는 반대로 마우스로부터 최대 70픽셀까지 떨어지게 그릴 수 있다는 것입니다. Int alpha= 150은 투명도를 결정하는 것입니다. 이제 10번 동안 반복문을 수행할 것인데,

반복문 안에서부터 코드에 대해 설명하자면 float offsetDistance = ofRandom(minOffset, maxOffset)는 마우스로부터의 거리를 랜덤하게 설정하는 것입니다. 그리고 ofVec2f mousePose(ofGetMouseX(), ofGetMouseY())는 마우스의 x, y좌표를 mousePos라는 2차원 벡터로 묶어 관리합니다. 다음 세 줄은 삼각형의 세 꼭짓점을 정의하는 것입니다. 이후 4줄은 삼각형의 세 점을 랜덤하게 0에서 360도 사이의 rotation 각도만큼 회전시킵니다. 이후 2줄은 삼각형을 마우스로부터 떨어뜨릴 오프셋 벡터를 똑같이 회전시키는 것입니다. 이후 이를 이용해 마우스 위치와 오프셋을 더해 최종 위치를 계산합니다. 그렇게 마우스 근처에서 랜덤한 방향으로 날아가는 삼각형을 그릴 수 있게 됩니다. 이후 다음 코드들은 청록색, 보라색을 지정해 이 사이의 색상을 랜덤하게 선택해서 삼각형을 그리는 것입니다. 결과는 다음과 같습니다.



다음 장에 이어서

7. update()

```
//--------------------------------------------------------------
void ofApp::setup(){
    // ofSetFrameRate(240);
    xPos = 5; // horizontal start position
    ofBackground(ofColor::black); // black background
    // We still want to draw on a black background, so we need to draw
    // the background before we do anything with the brush
}

//--------------------------------------------------------------
void ofApp::update(){
    xPos += 2;
    if(ofGetWidth() < xPos){
        // if horizontal position is off the screen (width)
        xPos = 5; // reset horizontal position
    }
}

//--------------------------------------------------------------
void ofApp::draw(){
    // If the left mouse button is pressed...
    ofSetColor(ofColor::red); // draw everything in red
    ofDrawCircle(xPos, 100, 10);
    // draw a circle at the (variable) horizontal position
    // 100 pixels from the top with a 10 pixel diameter
}
```

Setup함수에 대해 설명하자면 원의 가로 위치인 xPos에 5를 저장합니다. 이는 클래스안에 멤버 변수로 선언되어 있으며 처음에 화면 왼쪽 끝에서 시작하겠다는 의미입니다. 이후 다음 코드는 배경을 검정색으로 설정하겠다는 것입니다. 이후 update함수에 대해 설명하자면 이는 매프레임마다 실행됩니다. 코드를 보자면 X좌표 값을 2씩 증가시킵니다. 그리고 조건문을 통해 원이 화면 밖을 나갔는지 안 나갔는지 체크해서 나갔다면 리셋해서 다시 처음부터 시작합니다. Draw함수에 대해 설명하자면 이는 update다음에 매 프레임마다 실행됩니다. 단순하게 빨간색으로 그린다는 것입니다. 대신 x좌표는 매 프레임마다 다르지만 y좌표와 반지름은 고정됩니다.