9주차 결과보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181288 이름: 윤성호

<Basic Shape 1>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 1]



[사진 2]

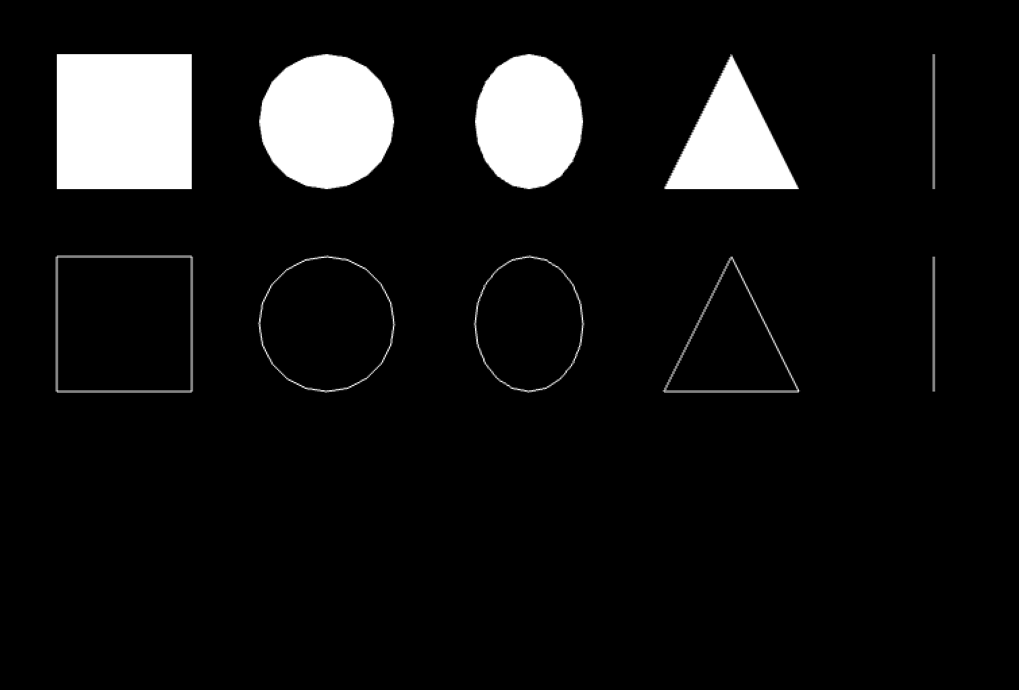
Basic shape 1은 검정색화면에서 하얀색 도형들을 그리는 것을 요구한다. 그림을 그리는 과정이기에 모든 코드는 draw() 함수 내부에 작성하였다. 가장 먼저, 배경색을 검정색으로 만들어주어야 하기에 ofBackGround(0)을 작성하였다. 여기서 0은 검정을 의미한다. 그 다음, 도형은 흰색으로 그려주어야 하기 때문에 ofSetColor(255)를 작성하였다. 여기서 255는 흰색을 의미한다. 이후 직사각형, 원, 타원, 삼각형, 선을 차례로 그린다. 먼저 직사각형은 ofDrawRectangle함수를 사용하여 그리며, 인자로는 (왼쪽 위 x좌표, 왼쪽 위 y좌표, 가로길이, 세로길이)로 사용한다. ofDrawRectangle(500,50,100,10)은 왼쪽 위 x좌표는 500, 왼쪽 위 y좌표는 50에 가로길이는 100, 세로길이는 10인 직사각형을 그린다. 다음으로는 원을 그리는데, ofDrawCircle함수를 사용한다. 인자로는 (중심의 x좌표, 중심의 y좌표, 반지름)의 형태로 사용한다. ofDrawCircle(250,100,50)은 중심이 (250, 100)이고 반지름이 50인 원을 그린다. 세번째로는 타원이다. ofDrawEllipse함수를 사용하며, 인자로는 (중심의 x좌표, 중심의 y좌표, 가로길이, 세로길이)의 형태로 사용한다. ofDrawEllipse(400,100,80,100)은 중심이 (400,100)이고, 가로길이가 80, 세로길이가 100인 타원을 그린다. 네번째로는 삼각형인데, ofDrawTriangle함수를 이용한다. 인자로는 (꼭짓점1의 x좌표, 꼭짓점1의 y좌표, 꼭짓점2의 x좌표, 꼭짓점2의 y좌표, 꼭짓점3의 x좌표, 꼭짓점3의 y좌표,)의 형태로 사용한다. OfDrawTriangle(500,150, 550,50, 600,150)은 세 점이 (500,150), (500, 50), (600,150)인 삼각형을 그린다. 마지막으로 선을 그리는데 ofDrawLine함수를 이용한다. 인자로는 (점1의 x좌표, 점1의 y좌표, 점2의 x좌표, 점2의 y좌표,)로 사용한다. ofDrawLine(700,50,700,150)은 양끝점이 각각 (700,50), (700,150)인 선을 그려준다. [사진 1]은 수행한 실습 코드이고, [사진 2]는 실습코드를 실행한 결과이다.

<Basic Shape 2>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 3]



[사진 4]

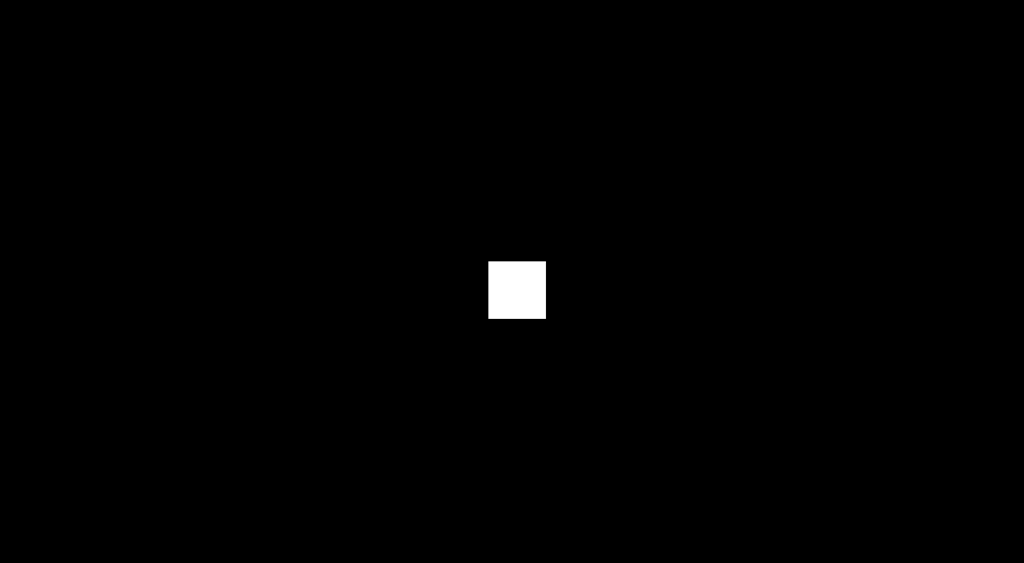
Basic shape 1의 그림에 추가적으로 채워지지 않은 도형을 그리는 것을 요구한다. 기본적으로 오픈프레임워크에서 도형을 그릴때는 채워져있는 것이 default이다. 따라서, 채워지지 않은 도형을 그릴때는 추가적인 옵션을 주어야하고, 이는 ofNoFill이다. ofNoFill() 함수를 호출하면 그 이후에 그려지는 도형들은 내부가 채워지지 않고 테두리만 그려진다. 다시 채워진 도형을 그리고 싶다면 ofFill()함수를 호출하면 된다. 첫번째 줄에 그려지는 도형들은 채워진 도형들이다. 이는 basic shape 1 과정에서 설명했으니 생략하고, 두번째 줄에 그려지는 도형들에 대해 설명하도록 하겠다. 먼저, 내부를 채우지 않은 도형을 그려야 하므로 ofNoFill() 함수를 호출해주고, ofDrawRectangle 함수를 통해 직사각형을, ofDrawCircle을 통해 원을, ofDrawEllipse를 통해 타원을, ofDrawTriangle을 통해 삼각형을, 그리고 마지막으로 ofDrawLine함수를 통해 선분을 그린다. 도형들의 y좌표를 충분히 크게 설정하여 기존에 그려진 도형과 겹치지 않게 한 후, 같은 모양의 도형을 그려주면 된다. [사진 3]는 basic shape2 를 수행한 실습코드이고, [사진 4]는 실습코드를 실행한 결과이다.

<Single Rectangle Brush>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 5]



[사진 6]

왼쪽 마우스를 클릭하면 그 위치에 한 변의 길이가 50인 정사각형을 그리는 것을 요구한다. 가장 먼저, 왼쪽 마우스 클릭 이벤트를 포착하기 위해서 ofGetMousePressed함수를 이용하고, 인자로 OF\_MOUSE\_BUTTON\_LEFT를 주어서 마우스 클릭 중 왼쪽마우스 클릭으로 제한하였다. 이러한 조건문 안에 그릴 사각형은 하얀색이므로 ofSetColor(255)를 이용해 도형이 그려질 색깔을 흰색으로 설정한다. 그 이후 ofSetRectMode(OF\_RECTMODE\_CENTER) 을 작성하여 마우스 클릭한 위치를 중심으로 정사각형이 그려지도록 설정한다. 만약, 이부분을 제거하고 코드를 실행시킨다면 클릭위치는 그려지는 정사각형의 왼쪽 위가 될 것이다. 이러한 설정을 해놓은 다음 ofDrawRectangle함수를 이용해 정사각형을 그린다. 정사각형을 그리는 위치는 마우스 클릭위치이고, 사이즈는 한 변의 길이가 50이다. [사진 5]는 수행한 실습 코드이고, [사진 6]은 실습 코드를 실행한 결과이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 7]



[사진 8]

위의 실습에서는 마우스가 클릭되는 위치에 정사각형을 그렸다. 그러나, 오픈프레임워크에서는 draw함수가 매 frame마다 새롭게 실행되므로, 마우스의 왼쪽 클릭을 누르다 뗀 이후에는 그려진 정사각형이 사라지게 된다. 그러나, 여러 프로그램을 만들다 보면 이미 그려진 도형을 화면에 계속 남아있게 할 필요가 있다. 이는, setup() 함수에 추가적인 설정을 해주면 가능해진다. ofSetBackgroundAuto는 새로운 frame 마다 화면을 clear 할지 말지 설정가능한 함수이다. 화면을 지우지 않고 그대로 유지하려면 인자로 false를 주면 된다. 그 다음, background의 색은 검정색으로 해주기 위해 ofBackground(0)을 해주었다. 0은 검정색을 의미한다. [사진 7]은 수행한 실습코드이고, [사진 8]은 실습코드를 실행한 결과이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 9]

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 10]

위의 실습을 진행할때는 정사각형의 색은 항상 하얀색이다. 도형의 색을 ofSetColor(255)를 이용하여 하얀색으로 고정해주었기 때문이다. 그러나, 색을 매번 랜덤하게 변화하게 하는 것도 가능한데, 오픈프레임워크에서 기본적으로 제공해주는 ofRandom함수를 이용하면 된다. ofRandom함수는 난수를 생성해주는 함수인데, 이는 인자를 하나주면 [0, 인자), 인자를 두개 주면 [인자1, 인자2) 사이의 floating point number를 리턴해주는 함수이다. 이를 이용해 randomColor를 50에서 255 사이의 임의의 수로 주었다. 0부터 하지 않은 이유는 현재 배경화면이 검정색이기에 정사각형이 배경색에 묻혀 잘 보이지 않을 수 있기 때문이다. 따라서, 검정배경에서도 잘 보이는 정도의 색을 랜덤 선택하기 위해 50을 최솟값으로 설정하였다. 그렇게 구해진 수를 ofSetColor의 인자로 넣어주어 매 정사각형이 그려질때마다 다른 색으로 그려지게 하였다. 그 이후, 클릭 위치가 정사각형의 중앙이 되도록 ofRectMode함수를 이용하였고, 정사각형을 클릭위치에 그리기 위해 ofDrawRectangle함수를 이용하였다. [사진 9]는 수행한 실습 코드이고, [사진 10]은 실습 코드를 실행한 것이다.

<Bursting Rectangle Brush>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 11]

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 12]

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

[사진 13]

앞의 실습에서는 마우스 왼쪽 클릭 위치에 정사각형이 하나 그려보았는데, 이번 실습에서는 작은 정사각형이 클릭 위치 주변에 여러개 생성되어 클릭 지점이 폭발하는 듯한 애니메이션을 구현하는 것을 요구한다. 일단, 왼쪽 마우스 클릭 이벤트를 포착하기 위해 ofGetMousePressed(OF\_MOUSE\_BUTTON\_LEFT)를 조건문의 조건으로 작성하였다. 즉, 왼쪽 마우스를 클릭하면 내부의 코드를 실행하라는 의미이다. 내부로 들어오면 가장 먼저 ofRectMode함수를 이용하여 마우스 클릭 위치가 도형이 그려지는 정중앙이 되도록 한다. 그 이후, numRect 변수를 10으로 주어 매 frame마다 for문을 이용하여 직사각형을 10개씩 그리도록 한다. 이때 10개의 직사각형이 그려지는 위치는 클릭위치를 기준으로 x좌표와 y좌표가 최대 20 떨어질 수 있도록 하였다. 또한, 직사각형의 모양은 가로 세로길이가 최소 5, 최대 20이 되도록 하였다. 좌표들과 가로세로 길이는 모두 특정 범위 내부의 수를 랜덤선택하는 것이므로 ofRandom 함수를 이용하였다. 그렇게 선택한 x좌표, y좌표, 가로길이, 세로길이는 ofDrawRectangle의 인자로 들어가여 직사각형을 화면에 그린다. 그런데 여기서는 매 프레임마다 직사각형이 10개씩 그려지므로, 이전 프레임과 새로운 프레임의 시간간격이 애니메이션의 스무스함을 결정한다. 즉, 시간간격이 짧다면 일정 시간동안 그려지는 직사각형의 개수가 증가하고, 시간간격이 길다면 그려지는 직사각형의 개수는 감소할 것이다. 이번 실습에서는 초당 프레임의 횟수를 60번으로 설정하는 것을 요구하고 있기 대문에 ofSetFrameRate(60) 을 setup 함수에서 호출해준다. [사진 11]은 수행한 실습코드의 draw함수부분의 코드이고, [사진 12]는 수행항 실습코드의 setup함수부분의 코드이고, [사진 13]은 실습코드를 실행한 결과이다.

<Glowing Circle Brush>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 14]

흐림이(가) 표시된 사진

자동 생성된 설명

[사진 15]

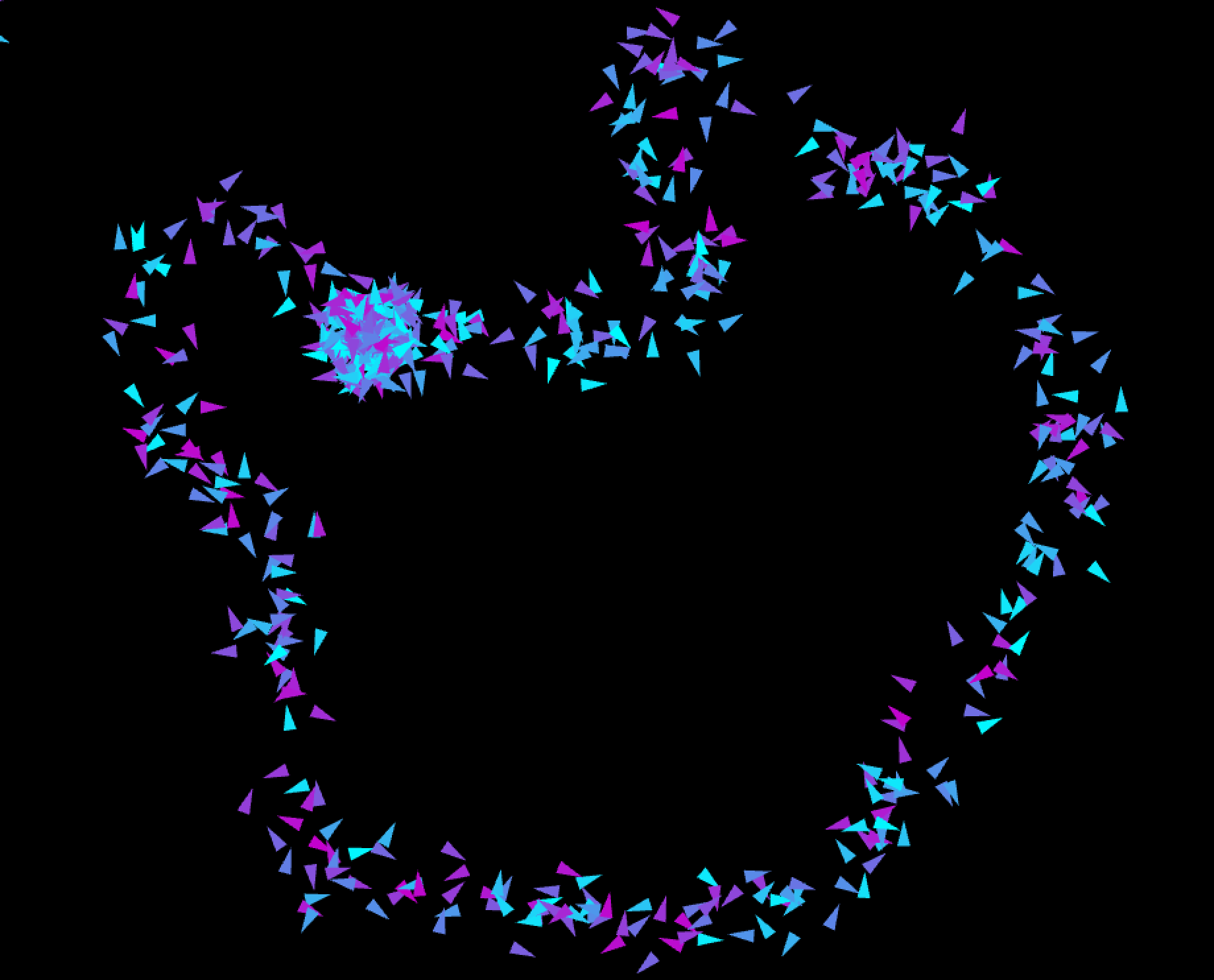
이번 실습은 적당히 희미한 원을 그려서 구름같은 모양을 화면에 출력하는 것을 요구한다. 오픈프레임워크에서는 ofColor 클래스를 이용해 사용자가 색을 정의할 수 있다. 이를 이용하여 myorange라는 변수로 새로운 색을 정의하였고, 인자로 (255, 132,0,3)을 주었다. 이때 앞에 세개의 인자는 R, G, B의 정도인데, 가장 뒤에 인자는 불투명도를 나타내는 인자다. 3을 주었기 때문에 default 불투명도보다 낮다. 즉, 더 투명하다는 의미이다. 그렇기에 이정도의 투명도를 가지는 원을 랜덤하게 그리면 구름같은 효과를 낼 수 있다. 이러한 색을 가지고 왼쪽 마우스를 클릭하게 되면, 클릭 한 위치를 기준으로 일정 offset만큼 떨어진 지점에 랜덤하게 선택된 원의 반지름을 가지는 원을 그린다. 더 구름같은 효과를 주기 위해서 cartesian coordinate보다는 polar coordinate을 이용하는 것이 좋다. 매 프레임마다 10개의 원을 그리는데, 반지름의 길이는 1에서 100 사이의 수로 랜덤하게 선택하고, 새로운 원을 그리기 위해 필요한 각과 길이를 ofRandom 함수를 이용해 얻어낸다. 그렇게 얻어난 각과 길이는 클릭 위치를 기준으로 이동할 offset으로 활용하여 ofDrawCircle 함수를 이용해 클릭 위치 기준 어느정도 떨어진 위치에 랜덤하게 선택한 반지름을 인자로 설정해주어 원을 그린다. [사진 14]는 수행한 실습 코드이고, [사진 15]는 실습코드를 실행한 결과이다.

<Fleeing Triangle Brush>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 16]



[사진 17]

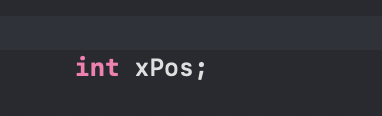
이번 실습에서는 마우스의 위치를 기준으로 어느정도 떨어진 지점에 랜덤하게 회전된 삼각형들을 그리는 것을 요구한다. 여러개의 값을 한번에 처리하기 위해 오픈프레임워크는 ofVec2f 클래스를 지원한다. 이를 통해 2차원 벡터를 정의하면, 좌표를 처리하는 데 아주 용이하다. mousePos 라는 변수에 현재 마우스의 위치벡터를 정의하고, 삼각형의 기본적인 세 꼭짓점을 정의한다. 그리고 glowing circle brush 실습에서와 마찬가지로 극좌표를 이용하여 offset을 만들어주기 위해 ofRandom함수로 각과 길이를 랜덤하게 선택한다. 이렇게 선택된 각과 길이는 offset을 만들어주는 데 사용한다. 그리고 이를 offset라는 2차원 벡터로 정의해놓아 나중에 사용할 수 있도록 하였다. 또한, 삼각형이 회전되어있으므로, 회전된 정도를 rotation이라는 변수에 0부터 360 사이의 수를 랜덤배정해준다. 그렇게 하여 그렇게 구한 rotation은 삼각형의 세점 p1, p2, p3를 rotate매서드를 이용하여 회전하는데 사용한다. 그 이후, 회전된 삼각형은 현재 마우스의 위치에서 offset만큼 더 이동한 위치에 그려지므로 p1, p2, p3에 mousePos와 offset을 더해준다. 또한, 삼각형은 아쿠아색과 퍼플색의 적당한 중간색이 랜덤하게 선택되며 그 색으로 그려지기 때문에 색 또한 설정해주어야 한다. ofColor 클래스를 이용하여 myaqua와 mypurple 색을 정의하였고, getLerped 매서드를 통해 그 두 색의 중간색을 랜덤으로 선택하여 이 색을 inbetween 변수에 정의하였다. 그리고 ofSetColor(inbetween)을 이용해 선택된 색으로 삼각형을 그릴 수 있는 준비를 한다. 그릴 준비가 다 되었으므로, ofDrawTriangle 함수를 이용하여 삼각형을 그린다. 인자로는 p1, p2, p3가 들어간다. [사진 16]은 수행한 실습코드이고, [사진 17]은 실습코드를 실행한 결과이다.

<update>

텍스트이(가) 표시된 사진

자동 생성된 설명

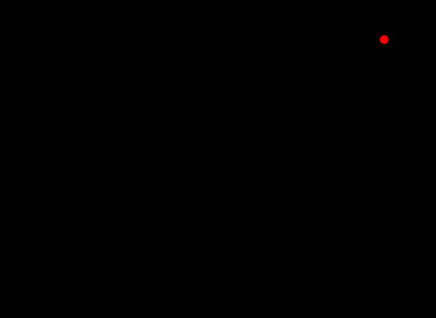
[사진 18]



[사진 19]

밤하늘이(가) 표시된 사진

자동 생성된 설명밤하늘이(가) 표시된 사진

자동 생성된 설명

[사진 20]

Update함수는 non-drawing 작업의 코드를 삽입하는 부분으로 매 프레임마다 새롭게 실행된다. 이를 확인해보기위해 ofApp.h에 xPos 변수를 정의하고 시작한다. 이번 실습은 빨간 원이 오른쪽으로 이동하다가 벽을 만나면 다시 가장 왼쪽으로 위치가 바뀌고 오른쪽으로 이동하는 애니메이션을 구현해본다. 이를 위해 setup함수에서는 가장 초기 xPos 값을 5로 설정해주고, 뒷배경을 검정색으로 설정해주기 위해 ofBackground함수를 사용하였다. 그리고, update함수에서는 xPos를 매 프레임마다 2씩 증가시키고, 만약 화면의 크기보다 xPos의 값이 더 크다면, 즉, 원이 화면에서 벗어난다면 xPos를 5로 설정해주어 다시 가장 왼쪽에 위치하게 한다. draw 함수에서는 ofSetColor을 이용해 빨간색으로 도형을 그리기 위한 설정을 하였고, ofDrawCircle함수를 이용해 중심이 (xPos, 100), 반지름이 10인 원을 그린다. [사진 18]은 수행한 실습코드의 ofApp.c 부분이고, [사진 19]는 수행한 실습코드의 ofApp.h이고, [사진 20]은 시간이 지날수록 변하는 빨간원의 위치를 나타낸 것이다.