PART 1 앱 인벤터 프로젝트

02 페인트 통

2차원 그래픽 프로그램

1970년대의 2차원 그래픽 프로그램

- 개인용 컴퓨터 잠재력을 보여주는 초창기 프로그램
- 매우 어렵고 조잡한 그래픽 화면

앱 인벤터에서는 아주 쉽게 화려한 그래픽이 가능

- 여기서는 <페인트 통>이라는 간단한 그림 그리기 앱을 작성
- 이후 3장에서 <두더지 잡기>, 5장에서 <무당벌레 추적> 게임 앱 작성

무엇을 개발하는가?

<페인트 통> 앱에서 개발하는 기능들

- 가상의 페인트 통에 손가락을 담가 색깔 선택
- 손가락으로 화면에 선을 그림
- 화면을 찍어 점을 그림
- 버튼을 클릭하여 화면을 깨끗이 지움
- 버튼을 클릭하여 점의 크기 조절
- 카메라로 사진을 찍고 그 위에 그림을 그림

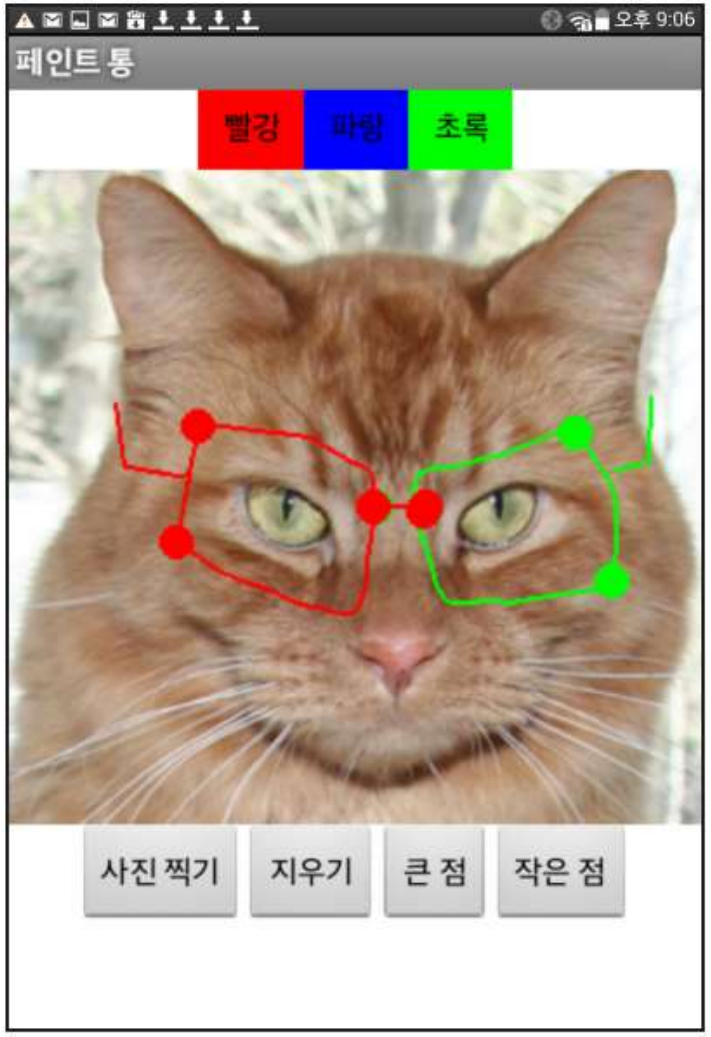


그림 2-1 〈페인트 통〉 앱

무엇을 배우는가?

<페인트 통> 앱을 만들며 배우는 것들

- 그림을 그릴 Canvas 컴포넌트
- 화면을 드래그하거나 터치할 때 발생하는 이벤트 처리
- Arrangement 컴포넌트로 여러 컴포넌트를 배치
- 매개변수를 갖는 이벤트 처리기
- 변수 사용

프로젝트 생성

프로젝트를 만들고, 라이브 테스팅 연결

- 프로젝트 이름은 "PaintPot"
- Screen1의 Title 속성은 "페인트 통"

앱 인벤터에서 중요한 세 가지 이름

① 프로젝트 이름

② 컴포넌트 이름 Screen1

③ 화면 제목

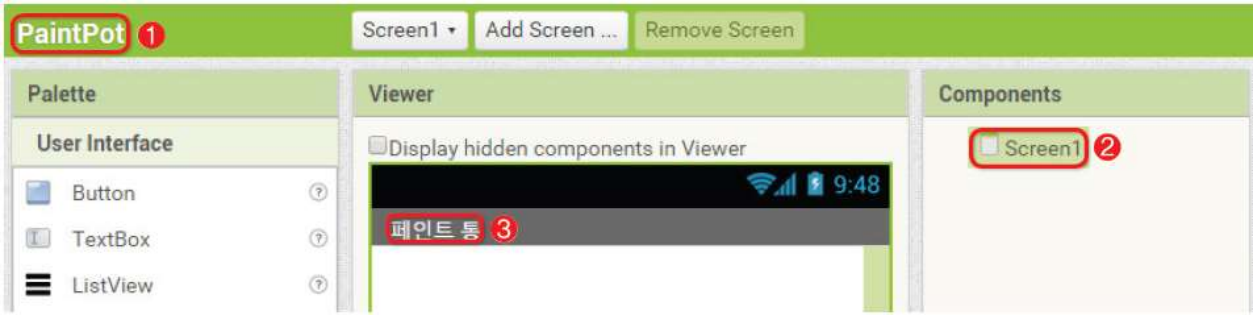


그림 2-2 앱 인벤터에서 중요한 세 가지 이름

Note _ 기본적으로 프로젝트 이름이 앱 이름이 되는데, Screen1의 AppName 속성을 설정하여 원하는 이름으로 바꿀 수 있다. 이는 한글 이름도 가능하다.

Note _ 라이브 테스팅에 대한 문서는 http://appinventor.chonbuk.ac.kr

컴포넌트 설계

<페인트 통> 앱을 만들 때 사용하는 컴포넌트

- 빨강, 파랑, 초록에 해당하는 세 개의 Button 컴포넌트
- 버튼을 한 줄에 배치하는 데 사용하는 HorizontalArrangement 컴포넌트
- 화면을 지우는데 쓰는 Button 컴포넌트
- 점의 크기를 바꾸는 데 사용할 Button 컴포넌트 두 개
- 사진을 찍으라는 명령을 내리는 데 사용할 Button 컴포넌트
- 그림을 그리는 데 사용할 Canvas 컴포넌트
- 사진을 찍는데 사용할 Camera 컴포넌트 (보이지 않는 컴포넌트)

색깔 버튼 만들기

절차

1. Button 컴포넌트 끌어오기 → Text 속성을 "빨강", BackgroundColor 속성을 "빨간색"
2. Button1이던 버튼 이름을 RedButton으로 바꿈
3. 같은 방법으로 BlueButton과 GreenButton을 만듦

컴포넌트에게 의미 있는 이름을 붙여주면 프로그램을 이해하는데 크게 도움이 됨

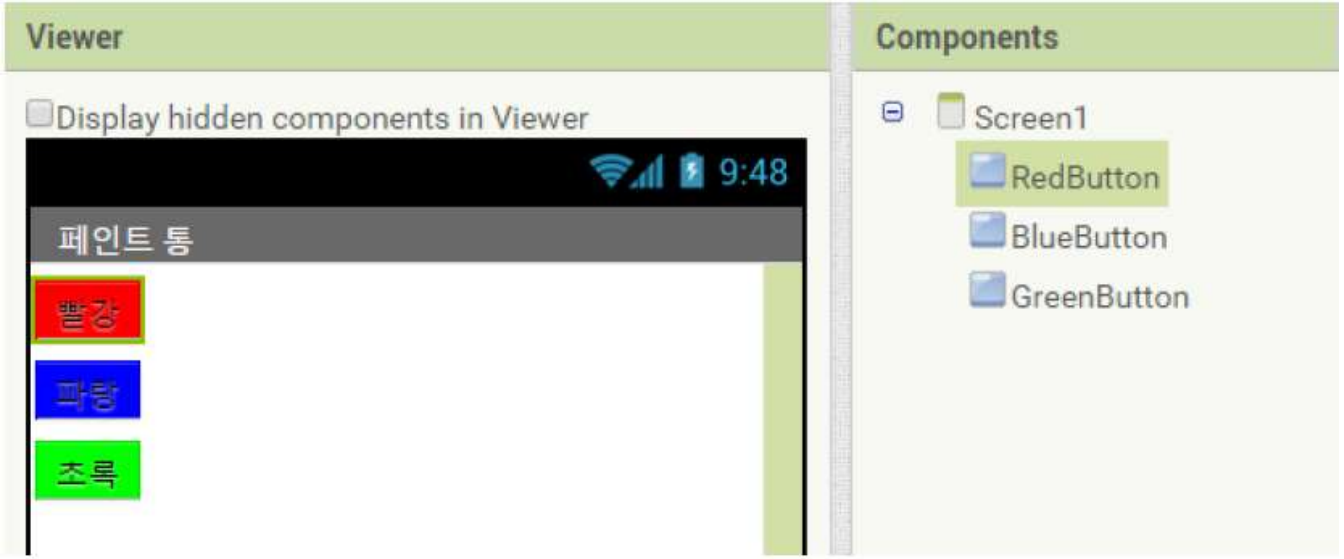


그림 2-3 세 개의 버튼이 배치된 Viewer 화면

라이브 테스팅

▲ **Test your APP**

아직까지 폰을 라이브 테스트 상태로 접속하지 않았다면, 바로 접속해 본다. 폰이 없다면 에뮬레이터를 사용하여 지금까지 만든 화면 인터페이스가 제대로 나타나는지 확인해 본다.

Note _ 에뮬레이터 사용에 대한 문서는 http://appinventor.chonbuk.ac.kr

Arrangement 컴포넌트로 깔끔하게 버튼 배치하기

절차

1. Layout 서랍에서 HorizontalArrangement 컴포넌트를 끌어옴
2. Width 속성을 Fill parent로 함(화면 전체 너비 차지)
3. 색깔 버튼을 HorizontalArrangement로 옮김 (세 버튼 컴포넌트는 부속 컴포넌트가 됨)

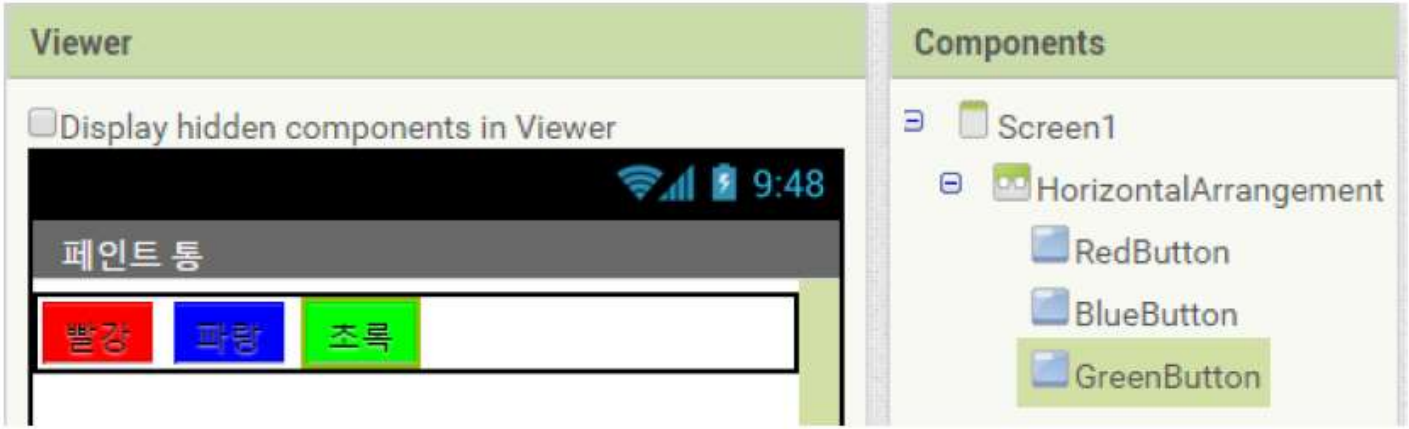


그림 2-4 수평으로 배치된 세 개의 버튼

▲ Test your APP

폰을 확인하면 세 개의 버튼이 한 줄에 나타날 것이다. 앱 인벤터 화면과는 약간 다를 수 있는데, 예를 들어 앱 인벤터에서는 HorizontalArrangement의 테두리가 표시되지만, 폰에서는 이것이 나타나지 않는다.

Canvas 컴포넌트 추가하기

절차

1. Canvas 컴포넌트를 끌어옴
   - 이름은 DrawingCanvas
   - Width는 Fill parent
   - Height는 300으로 설정
2. kitty.png 다운로드
3. DrawingCanvas의 BackgroundImage속성으로 kitty.png 업로드
4. Canvas 컴포넌트의 PaintColor 속성을 빨간색으로 설정

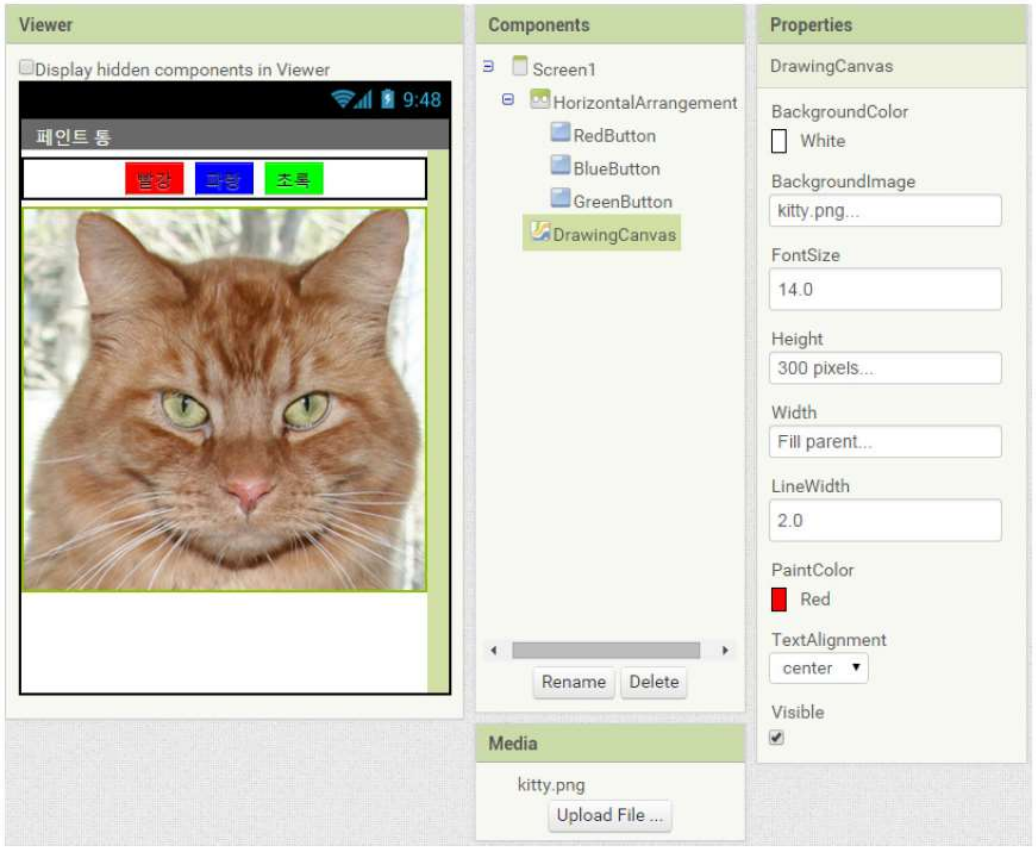


그림 2-5 고양이 사진이 배경인 DrawingCanvas 컴포넌트

아래쪽 버튼과 Camera 컴포넌트 추가하기

절차

1. HorizontalArrangement 컴포넌트를 끌어옴. 두 개의 버튼을 배치 (Text 속성을 "사진 찍기"와 "지우기"로 함)
2. 두 개 버튼을 추가 배치
3. Text 속성을 "큰 점"과 "작은 점"으로 설정
4. Camera 컴포넌트 끌어오기 (Non-visible components 영역에 나타남)

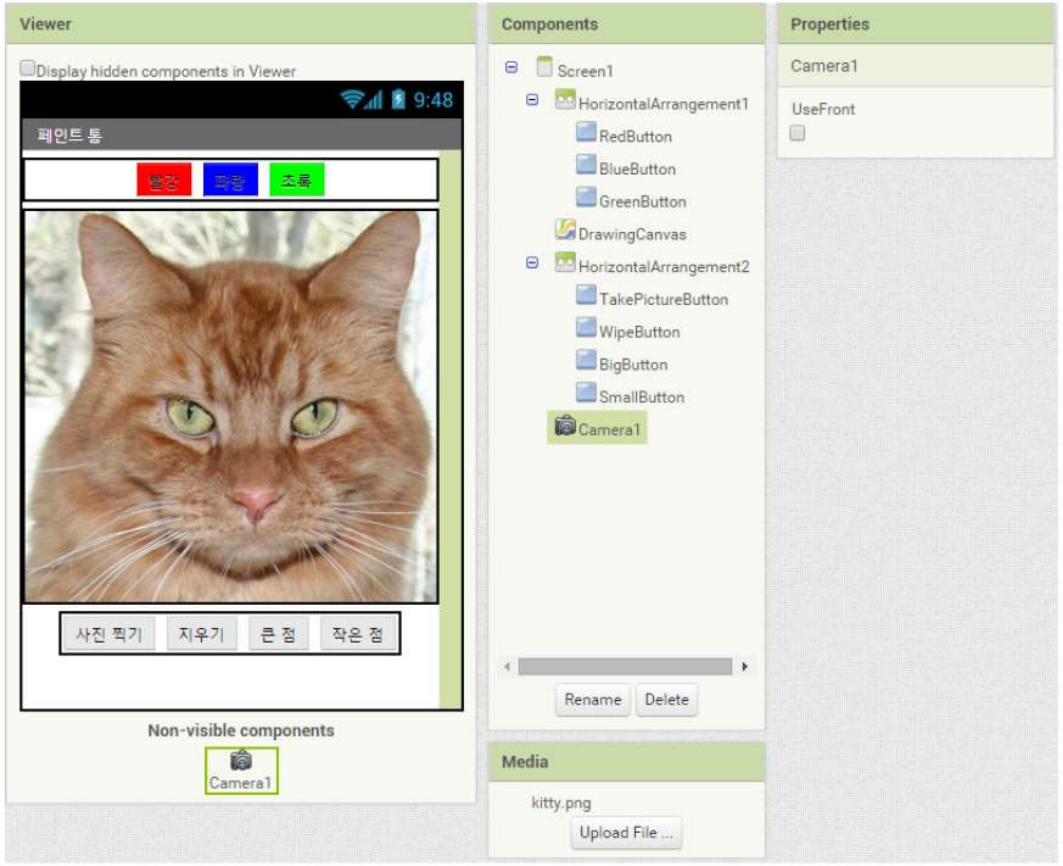


그림 2-6 〈페인트 통〉 앱의 사용자 인터페이스

▲ Test your APP

지금까지 작업한 내용이 폰에 제대로 나타나는지 확인해 보자. 캔버스에 고양이 사진이 나타나는가? 그 밑에 네 개의 버튼이 한 줄로 보이는가?

컴포넌트 동작 프로그래밍

프로그래밍할 일들

- DrawingCanvas 컴포넌트
  - 화면을 터치하면 점을 찍어줌
  - 화면을 드래그하면 선을 그려줌
- 버튼 컴포넌트
  - 색을 바꿈
  - 사진 찍음
  - 화면을 지움
  - 점의 크기를 바꿈

터치하여 점 찍기

절차

1. DrawingCanvas 서랍에서 DrawingCanvas.Touched 이벤트 처리기 블록 끌어옴
   - 사용자가 터치한 곳을 저장하고 있는 x와 y라는 매개변수

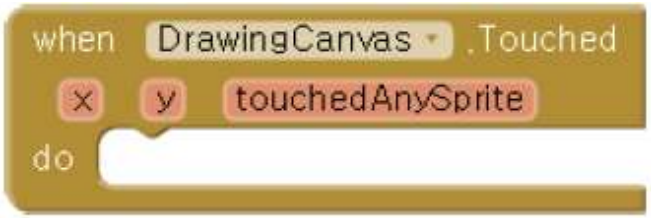


그림 2-7 터치 이벤트가 발생하면 x와 y에는 터치한 곳의 위치 정보가 저장된다.

Note _ 1장에서는 Button.Click과 같은 버튼 이벤트를 사용해 보았지만, 캔버스 이벤트는 이 장에서 처음으로 다루는 기능이다. Button.Click은 이벤트가 발생했다는 사실만 알면 되므로 프로그래밍하기가 쉬운 편이다. 하지만 이벤트 발생 시 생성된 정보가 매개변수에 저장되는 이벤트 처리기도 있다. 예를 들어, DrawingCanvas.Touched 이벤트 처리기에서는 터치된 지점의 위치 좌표가 매개변수 x와 y에 저장된다. 또한 매개변수 touchedAnySprite는 스프라이트라 불리는 물체를 터치했는지에 대한 정보를 가진다. touchedAnySprite는 3장에서 사용해 보기로 하고, 터치한 곳에 점을 찍는 데 필요한 x와 y는 바로 이어서 살펴보자.

터치하여 점 찍기

절차

2. DrawingCanvas 서랍에서 DrawingCanvas.DrawCircle 명령어를 끌어와 끼움
   - 원의 중심을 나타내는 centerx와 centery 매개변수
   - 반지름을 나타내는 radius 매개변수
   - 채울지 여부를 나타내는 fill 매개변수

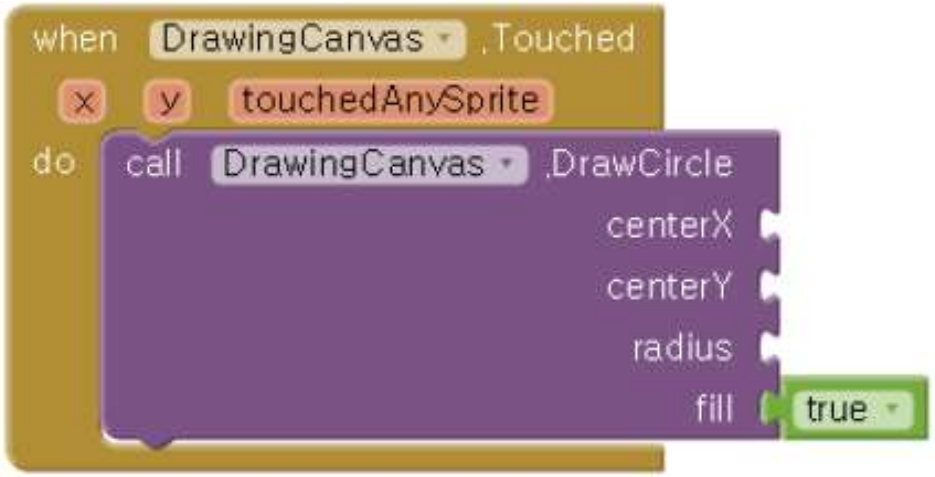


그림 2-8 사용자가 캔버스를 터치하면 원을 그려준다.

터치하여 점 찍기

절차

3. x와 y의 get 블록을 끌어와 DrawCircle의 해당 홈에 끼움

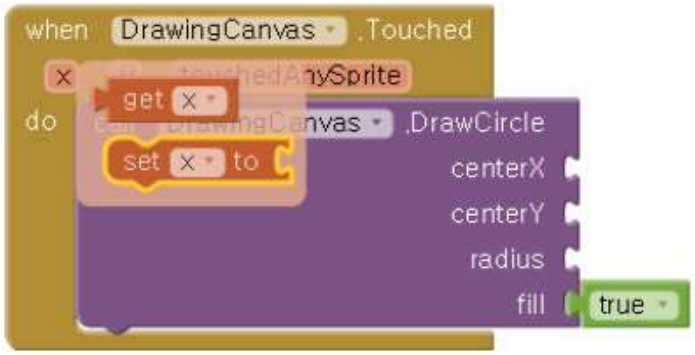


그림 2-9 매개변수 x의 get 블록을 얻으려면 매개변수 위에 마우스를 올려 둔다.

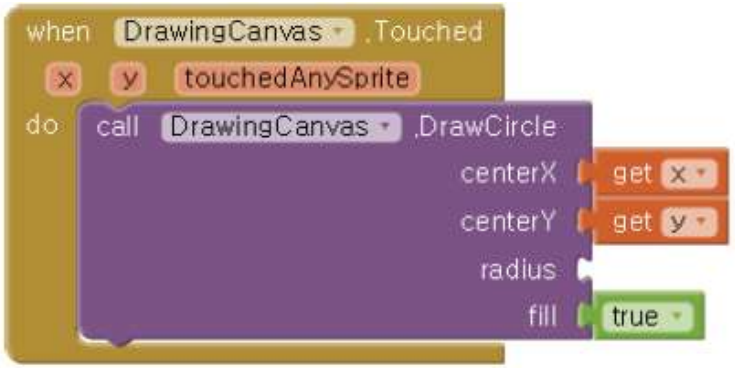


그림 2-10 x와 y로 원의 위치는 지정했으나 원의 크기는 아직 정하지 않은 상태

터치하여 점 찍기

절차

4. radius 매개변수를 5로 지정

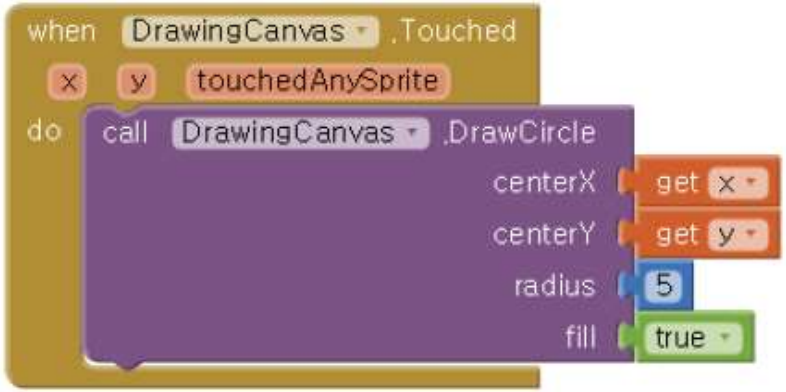


그림 2-11 사용자가 캔버스를 터치하면 터치한 곳(x,y)에 반지름이 5인 원을 그린다.

▲ Test your APP

지금까지 코딩한 내용을 테스트해 보자. 화면의 이곳저곳을 터치하고 그곳에 제대로 점이 찍히는지 확인해 본다. 앞에서 DrawingCanvas의 PaintColor 속성을 빨간색으로 지정해 두었으므로 빨간 점이 찍힌다. 색을 변경해 두지 않았다면, 기본값(검정)이 적용되어 검은 점이 찍힌다.

드래그하여 선 긋기

터치와 드래그의 차이점

- 화면에 손가락을 댄 다음 이동하지 않은 채 떼면 터치 이벤트 발생
- 화면에 손가락을 댄 다음 떼지 않은 채 이동하면 드래그 이벤트 발생

절차

1. DrawingCanvas 서랍에서 DrawingCanvas.Dragged 이벤트 처리기 블록 끌어옴
   - startX와 startY: 드래그가 처음 시작된 곳의 위치
   - prevX와 prevY: 손가락의 바로 이전 위치
   - currentX와 currentY: 손가락의 현재 위치

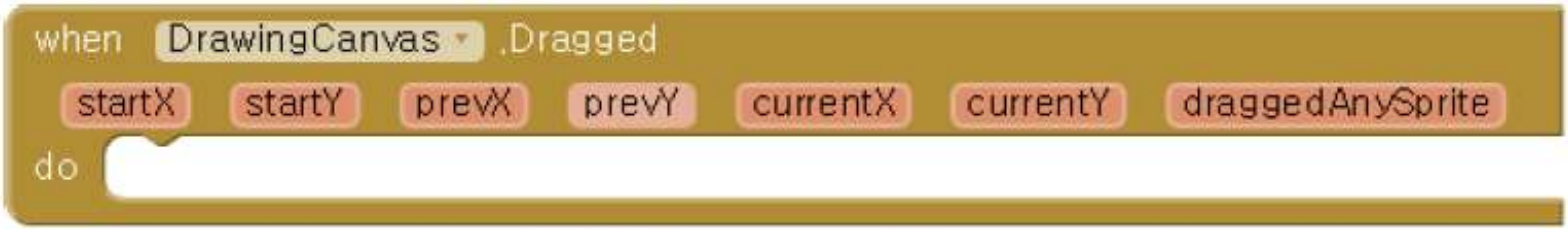


그림 2-12 Touched 이벤트보다 매개변수가 많은 Dragged 이벤트

드래그하여 선 긋기

절차

2. DrawingCanvas 서랍에서 DrawingCanvas.DrawLine 블록 끌어오기
3. get 블록 끼우기

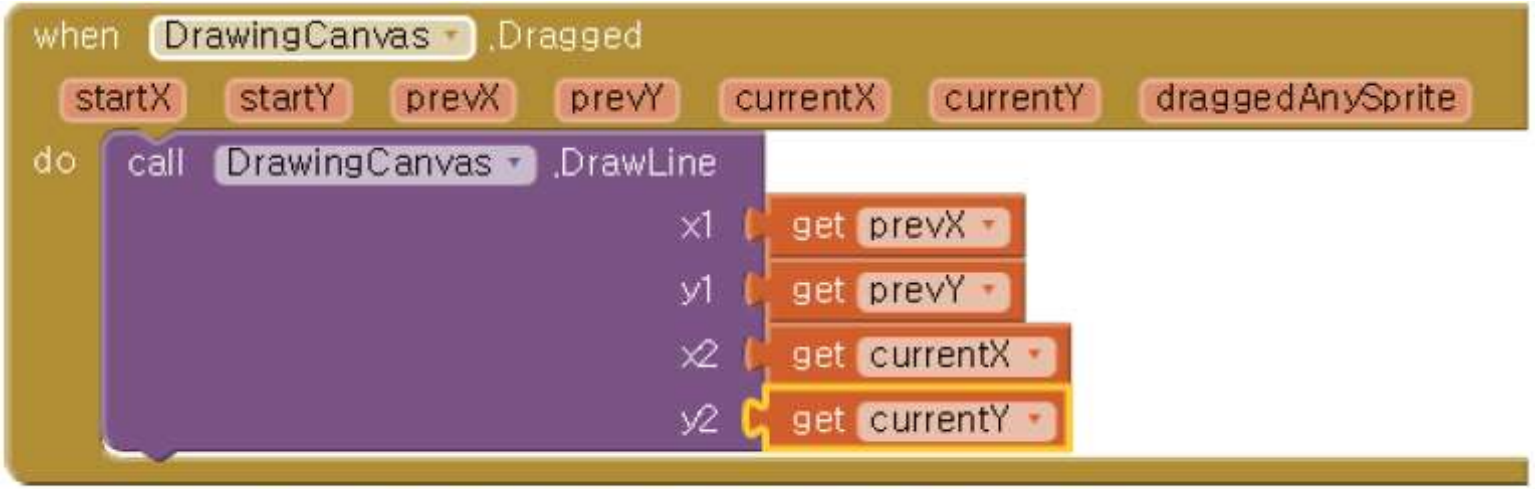


그림 2-14 사용자가 드래그하는 동안 이전 위치에서 현재 위치까지 짧은 선을 그려준다.

▲ Test your APP

화면에서 손가락을 드래그하여 직선과 곡선을 그려 보고, 터치하여 점도 찍어 본다.

색 바꾸기

절차

1. RedButton.Click 블록 끌어오기
2. set DrawingCanvas.PaintColor to 블록을 RedButton.Click에 끼우기
3. Color 서랍에서 빨간 블록을 끌어와 set DrawingCanvas.PaintColor to 블록에 끼움
4. 1~3을 반복하여 파랑과 초록 버튼도 코딩함
5. 지우기 버튼 코딩

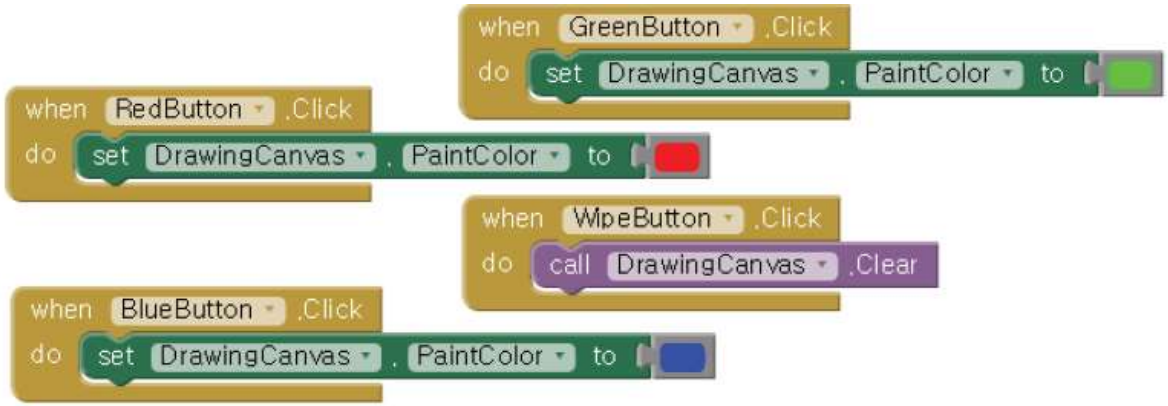


그림 2-15 색깔 버튼을 누르면 PaintColor 속성을 바꾸고, 지우기 버튼을 누르면 화면을 깨끗하게 지운다.

▲ **Test your APP**

색깔 버튼을 이것저것 눌러보고 선택한 색으로 점과 선이 그려지는지 살펴보자. [지우기] 버튼도 눌러보고 캔버스가 깨끗이 지워지는지 확인한다.

사진 찍기

절차

1. TakePictureButton.Click 이벤트 처리기 블록을 끌어오기
2. Camera1.TakePicture블록을 끌어와 TakePictureButton.Click에 끼우기
3. Camera1.AfterPicture 이벤트 처리기 블록을 끌어오기
4. set DrawingCanvas.BackgroundImage to 블록을 Camera1.AfterPicture에 끼우기
5. get Image 블록을 DrawingCanvas.BackgroundImage 블록에 끼우기

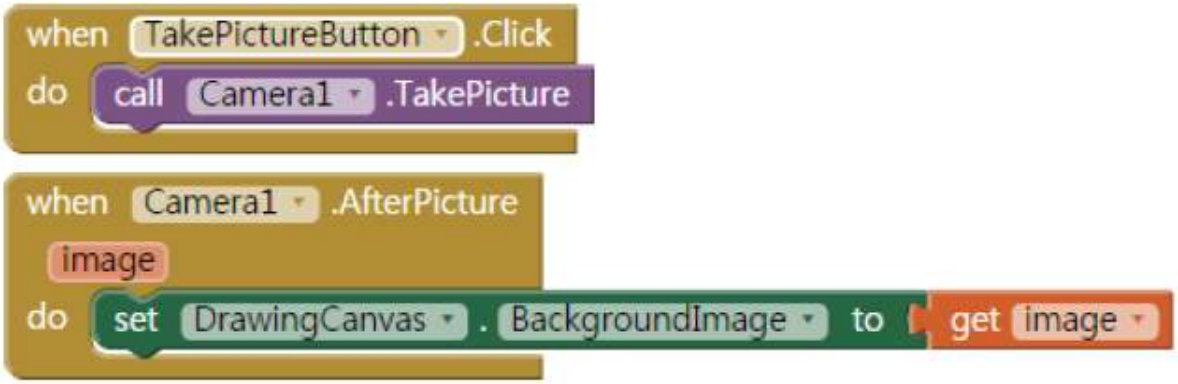


그림 2-16 카메라로 찍은 사진이 캔버스의 배경 영상이 된다.

사진 찍기

▲ **Test your APP**

앱에서 [사진 찍기] 버튼을 누른 다음 실제로 사진을 찍어 보자. 고양이 사진이 방금 찍은 사진으로 바뀌는가? 그렇다면 사진 위에 이것저것 그려 보자. [그림 2-17]은 학생들이 이 책의 저자인 올버 교수의 얼굴에 악살스런 그림을 그려 넣은 것이다.

그림 2-17 올버 교수 사진에 악살스런 그림 그리기

점 크기 바꾸기

점의 크기는 언제 정해질까?

- 사용자가 화면을 터치하면 DrawingCanvas.Touched 이벤트가 발생하며, 아래 이벤트 처리기 블록이 실행되고, 이때 radius가 5가 되어 반지름이 5인 원이 그려짐
- 5를 10으로 바꾸고 점이 커지는지 라이브 테스팅으로 실험해 보자.

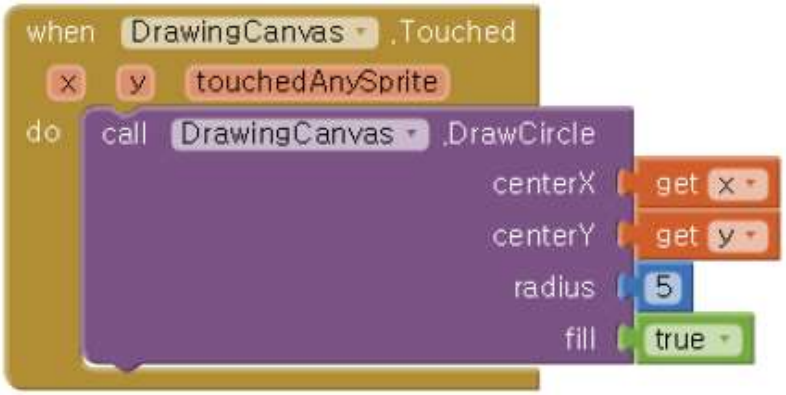


그림 2-11 사용자가 캔버스를 터치하면 터치한 곳(x,y)에 반지름이 5인 원을 그린다.

점 크기 바꾸기

한계점

- 점의 크기를 바꾸려면, 프로그래머가 매번 코드를 바꾸고 다시 실행해야 함
- 이런 방식의 프로그램을 하드 코딩(hard coding)되어 있다고 말함
- 프로그래머가 아니라 사용자가 자유롭게 점의 크기를 바꿀 수 있게 하려면?

해결책

- 변수(variable) 사용
- 사용자가 점의 크기를 지정하는 버튼을 누를 때마다 변수 값을 변경함
- DrawingCanvas.Touched 블록은 변수 값을 radius로 사용

점 크기 바꾸기

변수 dotSize 정의하기

- Variables 서랍에서 initialize global name to 블록을 끌어와 name을 dotSize로 변경
- 초깃값 지정

그림 2-18 변수 dotSize를 2로 초기화한다.

DrawCircle에서 dotSize 참조하기

절차

1. get global dotSize 꺼내오기
2. DrawingCanvas.Touched 블록에 get global dotSize 끼우기

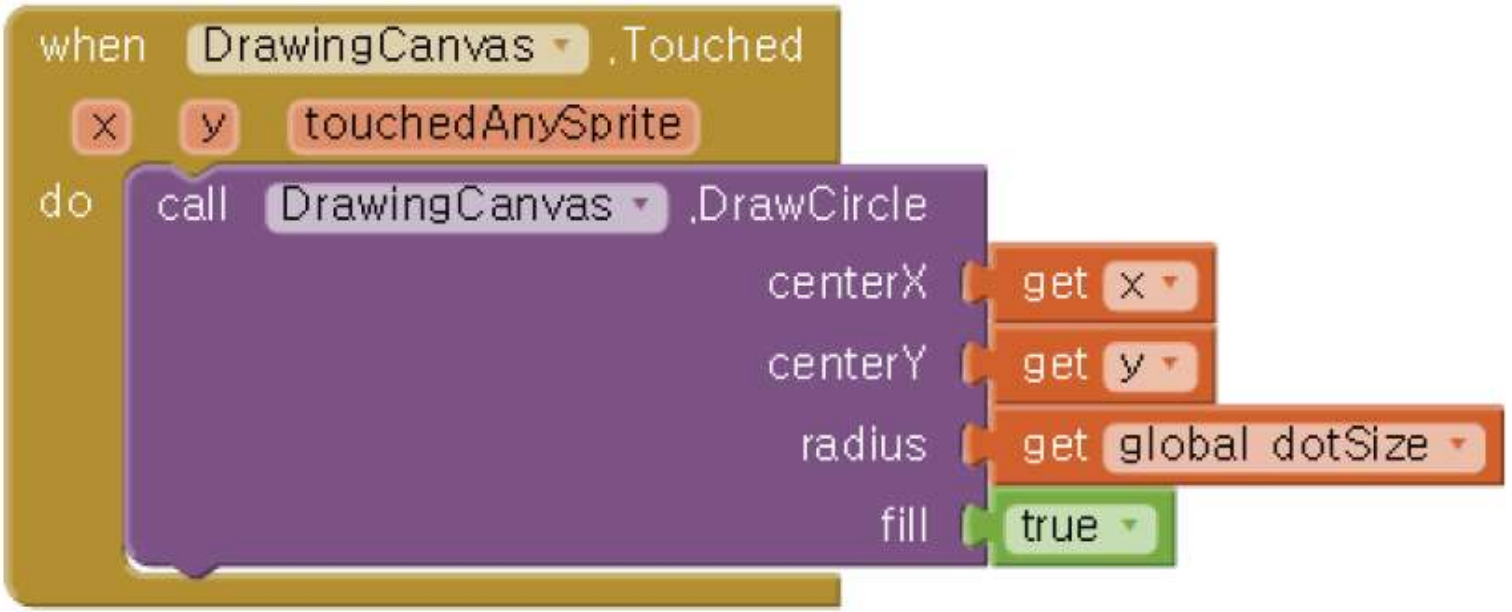


그림 2-19 점의 크기는 dotSize라는 변수에 따라 결정된다.

dotSize의 값 변경하기

절차

1. SmallButton.Click 이벤트 처리기에 set global dotSize to 끼우기

   → set global dotSize to에 숫자 블록 2 끼우기

2. BigButton.Click 이벤트 처리기에 set global dotSize to 끼우기

   → set global dotSize to에 숫자 블록 8 끼우기

그림 2-20 이들 버튼을 클릭하면 dotSize가 바뀌고, 해당 크기로 점이 그려진다.

Note _ set global dotSize에서 global은 전역을 의미하며, 프로그램에 있는 모든 이벤트 처리기가 이 변수를 사용할 수 있음을 뜻한다. 앱 인벤터는 프로그램의 특정 영역에서만 사용 가능한 '지역' 변수도 정의할 수 있다. 보다 자세한 내용은 21장을 참조한다.

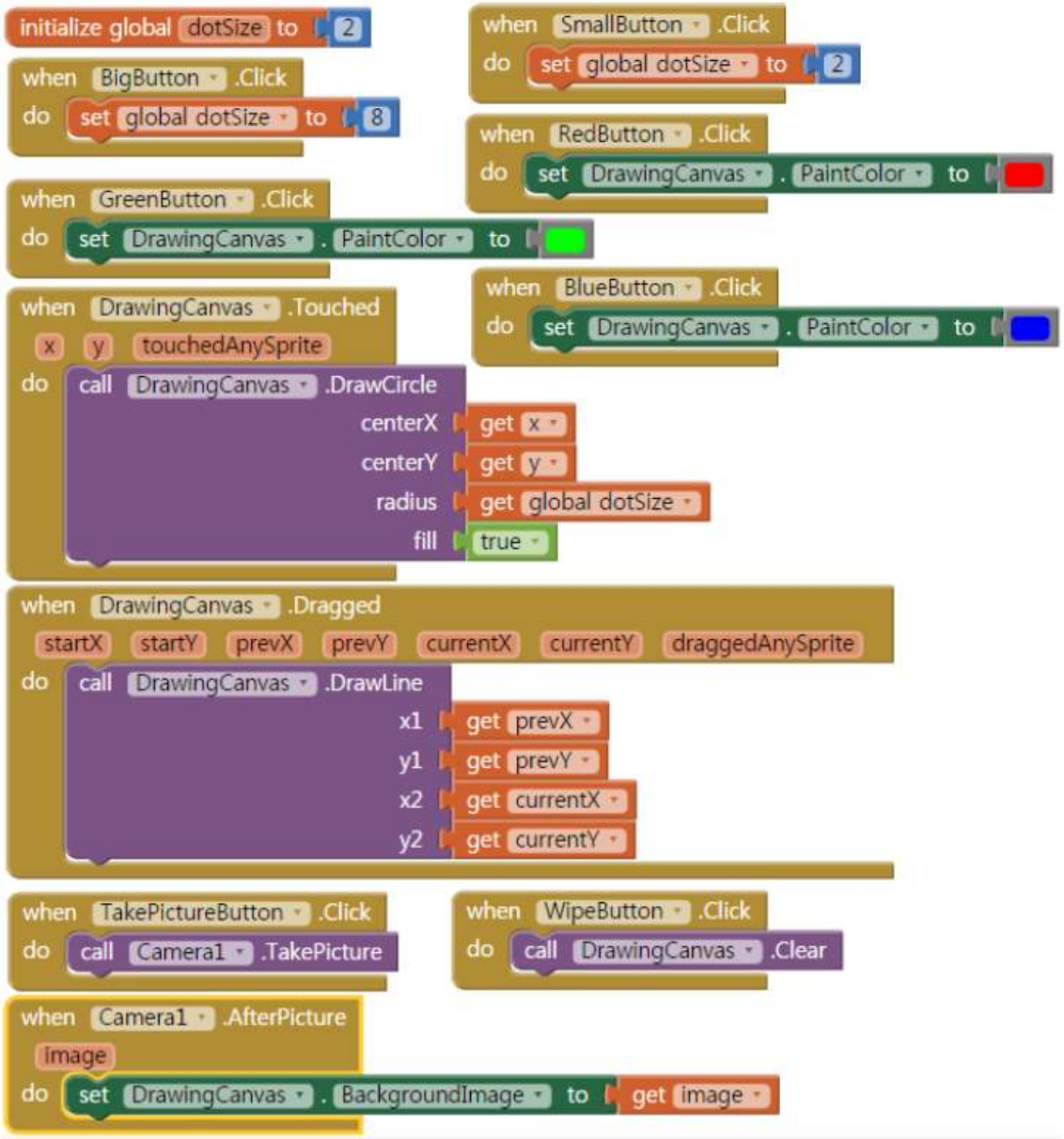
dotSize의 값 변경하기

▲ **Test your APP**

[큰 점]과 [작은 점] 버튼을 번갈아 누르며 화면을 터치해 점을 그려 보자. 점의 크기가 제대로 바뀌는가? 화면을 드래그하여 선도 그려본다. 선의 두께도 버튼에 따라 변하는가? DrawingCanvas.DrawCircle만 dotSize를 사용하고 있기 때문에 여기서 선 두께는 변하지 않아야 한다. 사용자가 선 두께도 변경할 수 있게 프로그램을 확장하려면 무엇을 추가해야 할지 생각해 보자.

Tip DrawingCanvas 컴포넌트에는 LineWidth 속성이 포함되어 있다.

전체 앱 프로그램


initialize global dotSize to 2
when BigButton .Click
do set global dotSize to 8
when SmallButton .Click
do set global dotSize to 2
when RedButton .Click
do set DrawingCanvas . PaintColor to
when GreenButton .Click
do set DrawingCanvas . PaintColor to
when BlueButton .Click
do set DrawingCanvas . PaintColor to
when DrawingCanvas .Touched
x y touchedAnySprite
do call DrawingCanvas .DrawCircle
centerX get x
centerY get y
radius get global dotSize
fill true
when DrawingCanvas .Dragged
startX startY prevX prevY currentX currentY draggedAnySprite
do call DrawingCanvas .DrawLine
x1 get prevX
y1 get prevY
x2 get currentX
y2 get currentY
when TakePictureButton .Click
do call Camera1 .TakePicture
when WipeButton .Click
do call DrawingCanvas .Clear
when Camera1 .AfterPicture
image
do set DrawingCanvas . BackgroundImage to get image

확장해 보기

다음의 아이디어를 구현하여 앱을 개선해 보자.

- 사용자 인터페이스는 현재 설정된 정보를 제공하지 않는다. 예를 들어 점의 크기나 색깔을 알아내려면 점을 찍어보는 수밖에 없다. 이러한 정보가 화면에 나타나도록 사용자 인터페이스를 개선한다.
- 사용자가 점의 크기로 2와 8뿐 아니라 임의의 값을 입력할 수 있게 확장한다. 이는 Slider 컴포넌트를 사용하면 된다.

감사합니다.