Chapter 09

그래픽과 이미지

모바일앱프로그래밍1
(2021년 2학기)

학습목표

- 캔버스에 도형을 그리는 방법을 익힌다.
- 이미지 파일을 처리하는 방식을 이해한다.
- 영상 처리 앱을 작성한다.

차례

01 캔버스와 페인트의 기본

- **Canvas, Paint**
  - 화면에 도형을 그릴 때 사용하는 클래스
  - 캔버스와 페인트는 도화지, 붓 과 같은 개념

그림 9-1 Canvas와 Paint 클래스

01 캔버스와 페인트의 기본

- android.graphics.Canvas 클래스의 점을 찍는 메소드의 원형

```
public void drawPoint (float x, float y, Paint paint)
```

- x와 y 좌표에 점을 찍음
  - 화면 왼쪽 상단 : (0,0)
  - Paint 개체에 설정된 색상, 두께 등에 따라 다르게 그려짐

- android.graphics.Paint 클래스에서 색상을 지정하는 속성과 사용법

```
var paint = Paint()
paint.color = Color.RED
```

01 캔버스와 페인트의 기본

- View 클래스를 재정의하는 형태로 그래픽 표현

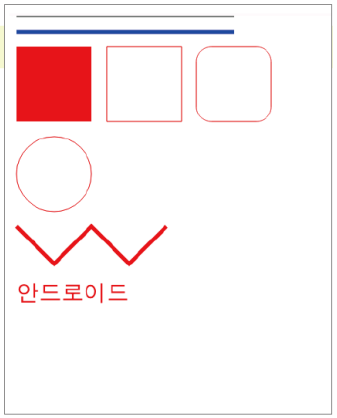
```
public override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(재정의한 클래스 이름(this))
}

private class 재정의한 클래스 이름(context: Context) : View(context) {
    override fun onDraw(canvas: Canvas) {
        super.onDraw(canvas)
        // 화면에 그려질 내용을 여기에 코딩
    }
}
```

02 그래픽 처리의 기본

- View.onDraw() 메소드를 오버라이딩하여 그래픽 출력

예제 9-1 그래픽 기본의 Kotlin 코드

```
1  public override fun onCreate(savedInstanceState: Bundle?) {
2      super.onCreate(savedInstanceState)
3      setContentView(MyGraphicView(this))
4  }
5
6  private class MyGraphicView(context: Context) : View(context) {
7
8      override fun onDraw(canvas: Canvas) {
9          super.onDraw(canvas)
10         val paint = Paint()
11         paint.isAntiAlias = true
12         paint.color = Color.GREEN
13         canvas.drawLine(10f, 10f, 300f, 10f, paint)
14
15         paint.color = Color.BLUE
16         paint.strokeWidth = 5f
17         canvas.drawLine(10f, 30f, 300f, 30f, paint)
18
19         paint.color = Color.RED
20         paint.strokeWidth = 0f
21
22         paint.style = Paint.Style.FILL
23         val rect1 = Rect(10, 50, 10 + 100, 50 + 100)
24         canvas.drawRect(rect1, paint)
25
```



02 그래픽 처리의 기본

```
26          paint.style = Paint.Style.STROKE
27          val rect2 = Rect(130, 50, 130 + 100, 50 + 100)
28          canvas.drawRect(rect2, paint)
29
30          var rect3 = RectF(250f, 50f, 250f + 100f, 50f + 100f)
31          canvas.drawRoundRect(rect3, 20f, 20f, paint)
32
33          canvas.drawCircle(60f, 220f, 50f, paint)
34
35          paint.strokeWidth = 5f
36          val path1 = Path()
37          path1.moveTo(10f, 290f)
38          path1.lineTo(10f + 50f, 290f + 50f)
39          path1.lineTo(10f + 100f, 290f)
40          path1.lineTo(10f + 150f, 290f + 50f)
41          path1.lineTo(10f + 200f, 290f)
42          canvas.drawPath(path1, paint)
43
44          paint.strokeWidth = 0f
45          paint.textSize = 30f
46          canvas.drawText("안드로이드", 10f, 390f, paint)
47      }
48  }
```

02 그래픽 처리의 기본

▶ **직접 풀어보기 9-1**

그림과 같은 화면을 출력하도록 다음 메소드를 사용하여 Kotlin을 코딩하라.

- Paint.setStrokeCap( )
- Canvas.drawOval( )
- Paint.setColor(Color.argb( ))

그림 9-2 그래픽 메소드의 활용

03 터치 이벤트

- View 클래스의 onTouchEvent() 메소드를 오버라이딩하여 터치 이벤트 구현

```
override fun onTouchEvent(event: MotionEvent): Boolean {
    when (event.action) {
        MotionEvent.ACTION_DOWN -> {
            // 손가락으로 화면을 누르기 시작했을 때 할 일
        }
        MotionEvent.ACTION_MOVE -> {
            // 터치 후 손가락을 움직일 때 할 일
        }
        MotionEvent.ACTION_UP -> {
            // 손가락을 화면에서 뗄 때 할 일
        }
        MotionEvent.ACTION_UP -> {
            // 터치가 취소될 때 할 일
        }
    }
    return true
}
```

03 터치 이벤트

- **<실습 9-1> 간단 그림판 앱 만들기**
  - 화면을 손가락으로 터치, 드래그하여 선 또는 원을 그리는 간단한 앱 만들기
  - 옵션 메뉴로 선, 원 그리기 선택

- 안드로이드 프로젝트 생성
  - (1) 새 프로젝트 만들기
    - 프로젝트 이름 : 'Project9_1'
    - 패키지 이름 : 'com.cookandroid.project9_1'
    - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름

- 화면 디자인 및 편집
  - (2) 이번 프로젝트는 Kotlin 코드로만 작성
    - activity_main.xml은 필요 없으므로 삭제

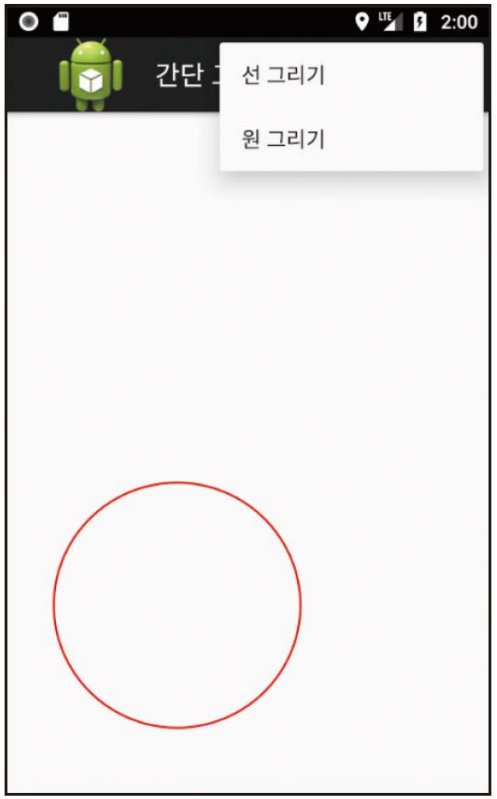


그림 9-3 간단 그림판 앱 결과 화면

03 터치 이벤트

- Kotlin 코드 작성 및 수정
  - (3) MainActivity.kt를 코딩
    - View 클래스의 상속을 받는 MyGraphicView 클래스 만들기

예제 9-2 Kotlin 코드 1

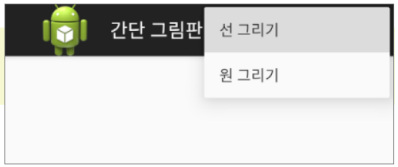
```
1  ~~~ 생략(import 문) ~~~
2  class MainActivity : AppCompatActivity() {
3      companion object {
4          const val LINE = 1
5          const val CIRCLE = 2
6          var curShape = LINE
7      }
8
9      override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(MyGraphicView(this))
12         title = "간단 그림판"
13     }
14
15     private class MyGraphicView(context: Context) : View(context) {
16
17     }
18 }
```

03 터치 이벤트

- (4) 선 그리기, 원 그리기 옵션 메뉴 작성
  - 항목을 클릭하면 curShape 변수에 선택한 전역상수를 대입
  - onCreate( ) 메소드와 같은 레벨로 만들어야 함
    : onCreateOptionsMenu( )와 onOptionsItemSelected( ) 메소드를 오버라이드함
    - » 완성 후에 화면 오른쪽 위의 메뉴(⋮)버튼을 클릭하여 확인해 보기

예제 9-3 Kotlin 코드 2

```
override fun onCreateOptionsMenu(menu: Menu?) : Boolean {
    super.onCreateOptionsMenu(menu)
    menu!!.add(0, 1, 0, "선 그리기")
    menu!!.add(0, 2, 0, "원 그리기")
    return true
}

override fun onOptionsItemSelected(item: MenuItem) : Boolean {
    when (item.itemId) {
        1 -> {
            curShape = LINE // 선
            return true
        }
        2 -> {
            curShape = CIRCLE // 원
            return true
        }
    }
    return super.onOptionsItemSelected(item)
}
```



03 터치 이벤트

- (9) MyGraphicView 클래스에 터치와 관련된 메소드를 완성함.
  - MyGraphicView의 전역변수 시작x, 시작y, 끝x, 끝y, 반지름 변수 선언
  - MyGraphicView 클래스 안에서 onTouchEvent( ) 메소드를 오버라이드함

예제 9-4 Kotlin 코드 3

```
1  private class MyGraphicView(context: Context) : View(context) {
2      var startX = -1
3      var startY = -1
4      var stopX = -1
5      var stopY = -1
6      override fun onTouchEvent(event: MotionEvent?) : Boolean {
7          when (event!!.action) {
8              MotionEvent.ACTION_DOWN -> {
9                  startX = event.x.toInt()
10                 startY = event.y.toInt()
11             }
12             MotionEvent.ACTION_MOVE, MotionEvent.ACTION_UP -> {
13                 stopX = event.x.toInt()
14                 stopY = event.y.toInt()
15                 this.invalidate()
16             }
17         }
18         return true
19     }
20
21 }
```

03 터치 이벤트

- (6) 실제로 화면에 도형이 그려질 onDraw() 메소드를 완성
  - MyGraphicView의 내부에 onDraw( )를 자동 완성하고 나머지 코딩
  - 페인트에 선의 두께, 채우기 여부, 선의 색상 지정
  - when 문으로 메뉴에서 선택한 내용에 따라 선 또는 원 그림

예제 9-5 Kotlin 코드 4

```
override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    val paint = Paint()
    paint.isAntiAlias = true
    paint.strokeWidth = 5f
    paint.style = Paint.Style.STROKE
    paint.color = Color.RED

    when (curShape) {
        LINE -> canvas.drawLine(startX, startY, stopX, stopY, paint)
        CIRCLE -> {
            var radius = Math.sqrt(Math.pow((stopX - startX).toDouble(), 2.0)
                    + Math.pow((stopY - startY).toDouble(), 2.0))
            canvas.drawCircle(startX, startY, radius.toFloat(), paint)
        }
    }
}
```

03 터치 이벤트

- 프로젝트 실행 및 결과 확인
  - (7) 완성된 코드를 실행해봄. [그림 9-3]과 같이 선이나 원이 그려짐

▶ **직접 풀어보기 9-2**

[실습 9-1]을 다음과 같이 수정하라.

- 클릭한 두 점을 끝점으로 하는 사각형이 추가로 그려진다.
- 옵션 메뉴에서 색상을 선택하게 한다. 색상은 서브 메뉴로 나오게 한다.

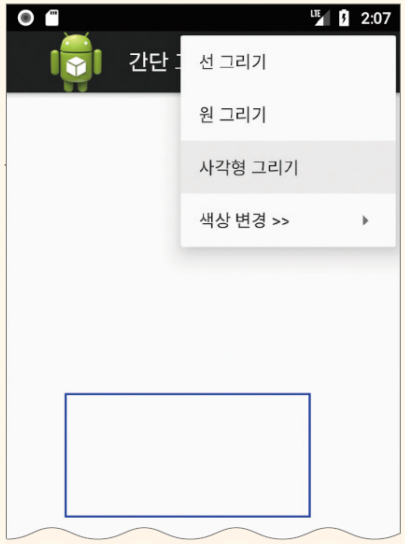


그림 9-4 수정된 간단 그림판 앱

01 비트맵의 기본

- **Bitmap 클래스**
  - 캔버스에 이미지 파일을 보여주는데 사용됨
    - /res/drawable 폴더의 이미지 파일을 보여주는 onDraw( ) 메소드

```
override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)

    var picture = BitmapFactory.decodeResource(resources, R.drawable.그림id)
    canvas.drawBitmap(picture, 시작x, 시작y, null)
    picture.recycle()
}
```

01 비트맵의 기본

- SD 카드의 이미지 파일을 보여주는 onDraw( ) 메소드

```
override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)

    var picture = BitmapFactory.decodeFile("파일경로 및 파일")
    canvas.drawBitmap(picture, 시작x, 시작y, null)
    picture.recycle()
}
```

- drawBitmap(picture, 시작x, 시작y, null) 메소드로 화면 중앙에 이미지 출력

```
시작x = (View 너비 - 이미지 너비) / 2
시작y = (View 높이 - 이미지 높이) / 2
```

01 비트맵의 기본

예제 9-6 이미지를 화면 중앙에 출력하는 Kotlin 코드

```
1  public override fun onCreate(savedInstanceState: Bundle?) {
2      super.onCreate(savedInstanceState)
3      setContentView(MyGraphicView(this))
4  }
5
6  private class MyGraphicView(context: Context) : View(context)
7
8      override fun onDraw(canvas: Canvas) {
9          super.onDraw(canvas)
10         var picture = BitmapFactory.decodeResource(resources, R.drawable.jeju14)
11         var picX = (this.width - picture.width) / 2f
12         var picY = (this.height - picture.height) / 2f
13         canvas.drawBitmap(picture, picX, picY, null)
14         picture.recycle()
15     }
16 }
```

02 이미지의 기하학적 변환

- 많이 사용되는 Canvas 클래스의 기하학적 메소드
  - 회전: rotate( )
  - 확대/축소: scale( )
  - 이동: translate( )
  - 기울이기: skew( )
    - 캔버스(도화지)에 대해 기하학적 변환을 한 후에 이미지 파일을 변환된 캔버스에 출력하는 형태

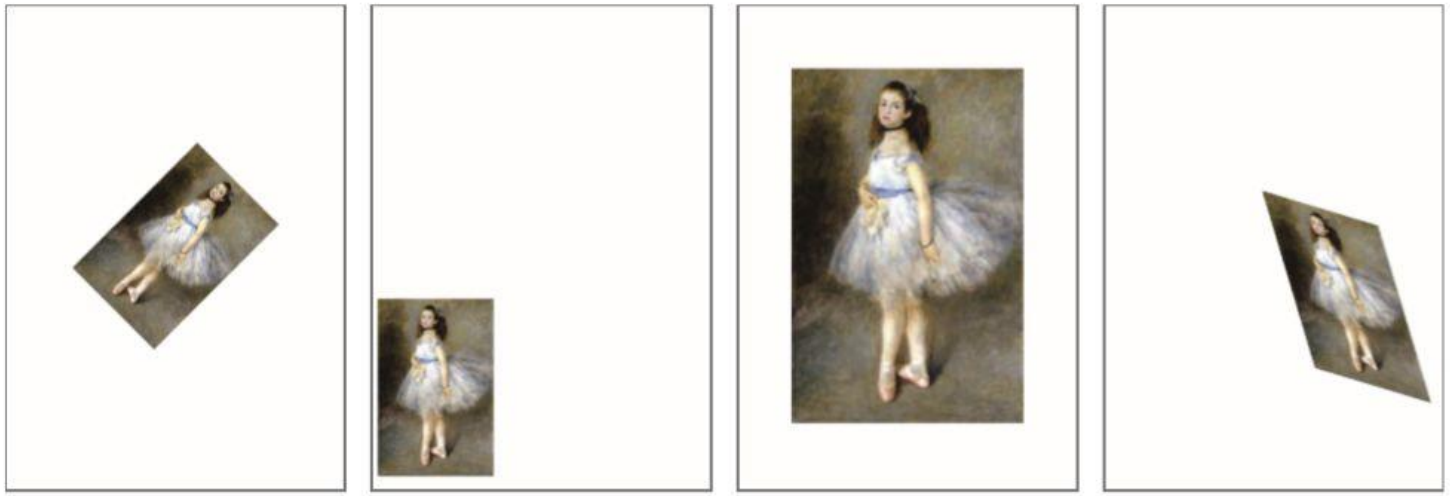


그림 9-5 기하학적 변환

02 이미지의 기하학적 변환

예제 9-7 기하학적 변환의 Kotlin 코드

```
override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    var picture = BitmapFactory.decodeResource(resources,R.drawable.small)

    var cenX = this.width / 2f
    var cenY = this.height / 2f
    var picX = (this.width - picture.width) / 2f
    var picY = (this.height - picture.height) / 2f

    canvas.rotate(45f, cenX, cenY)
    canvas.drawBitmap(picture, picX, picY, null)

    canvas.translate(-150f, 200f)
    canvas.drawBitmap(picture, picX, picY, null)

    canvas.scale(2f, 2f, cenX, cenY)
    canvas.drawBitmap(picture, picX, picY, null)

    canvas.skew(0.3f, 0.3f)
    canvas.drawBitmap(picture, picX, picY, null)

    picture.recycle()
}
```

03 이미지 활용

- 블러링(blurring)
  - 이미지를 뿌옇게 만듦
  - BlurMaskFilter 클래스를 제공

```
BlurMaskFilter(반지름, 스타일)
```

  - 반지름 : 블러링이 될 너비
    - 반지름이 클수록 이미지의 가장자리가 크게 블러링됨
  - 스타일 : NORMAL, INNER, OUTER, SOLID

그림 9-6 블러링 효과

03 이미지 활용

예제 9-8 블러링 효과의 Kotlin 코드

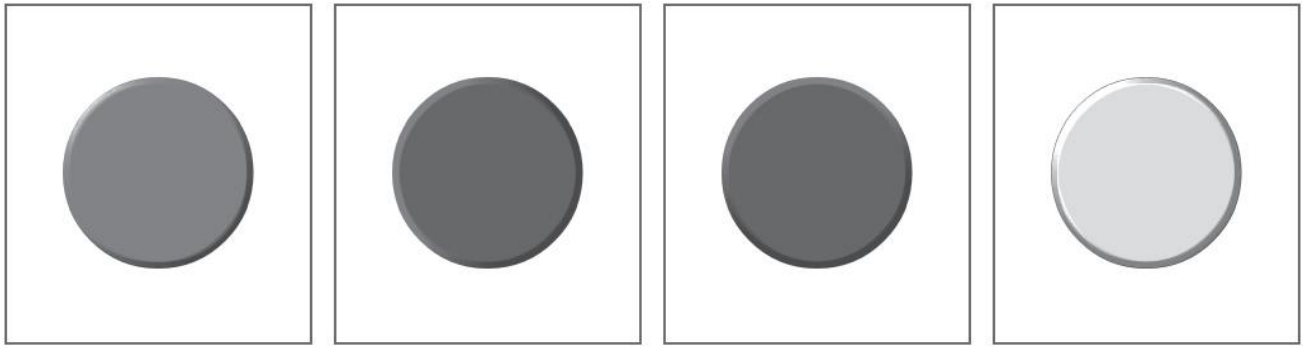
```
override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    var picture = BitmapFactory.decodeResource(resources,R.drawable.lena256)

    var picX = (this.width - picture.width) / 2f
    var picY = (this.height - picture.height) / 2f

    var paint = Paint()
    var bMask : BlurMaskFilter

    bMask = BlurMaskFilter(30f, BlurMaskFilter.Blur.NORMAL)
    paint.maskFilter = bMask
    canvas.drawBitmap(picture, picX, picY, paint)
    picture.recycle()
    ~~~ 생략(INNER, OUTER, SOLID 스타일) ~~~
}
```

03 이미지 활용

- 엠보싱(embossing)
  - 이미지가 볼록하게 튀어나와 보이는 효과를 줌
  - EmbossMaskFilter 클래스를 제공

```
EmbossMaskFilter(빛의 xyz 방향 1차 배열, 빛의 밝기, 반사 계수, 블러링 크기)
```

  - 첫 번째 파라미터 : 빛이 비추는 x, y, z 방향 배열
  - 두 번째 파라미터 : 빛의 밝기로 0부터 1까지 지정
  - 반사 계수 : 5~8 정도가 적당함
  - 블러링 크기 : 볼록하게 표현하기 위한 가장자리의 크기를 나타냄



(a) 빛 방향 {3, 3, 3} (b) 빛 방향 {10, 3, 3} (c) 빛 방향 {3, 10, 3} (d) 빛 방향 {3, 3, 10}

그림 9-7 엠보싱 효과

03 이미지 활용

예제 9-9 엠보싱 효과의 Kotlin 코드

```
override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)

    var cenX = this.width / 2f
    var cenY = this.height / 2f

    var paint = Paint()
    paint.color = Color.GRAY
    var eMask : EmbossMaskFilter

    eMask = EmbossMaskFilter(floatArrayOf(3f, 3f, 3f), 0.5f, 5f, 10f)
    paint.maskFilter = eMask
    canvas.drawCircle(cenX, cenY, 150f, paint)
    ~~~ 생략(빛의 방향을 바꾼 세 가지 스타일) ~~~
}
```

03 이미지 활용

- 컬러매트릭스
  - 색상이나 밝기 조절
  - ColorMatrix 클래스와 ColorMatrixColor Filter 클래스를 제공

```
var paint = Paint()
var array = floatArrayOf(4 × 5 배열)
var cm = ColorMatrix(array)
paint.colorFilter = ColorMatrixColorFilter(cm)
canvas.drawBitmap(···)
```

03 이미지 활용

- ColorMatrix에 사용할 배열

```
Red (1)     0           0           0           Brightness(0)
0           Green (1)   0           0           Brightness(0)
0           0           Blue (1)    0           Brightness(0)
0           0           0           Alpha(1)    0
```

- Red, Green, Blue, Alpha의 값은 기본적으로 1이 설정
  - 이 값을 몇 배로 올리면 각 색상의 대비(contrast)가 커짐
- Brightness
  - 양수 : 색상이 밝아짐
  - 음수 : 색상이 어두워짐

03 이미지 활용

예제 9-10 컬러매트릭스의 Kotlin 코드

```
override fun onDraw(canvas: Canvas) {

    super.onDraw(canvas)

    var picture = BitmapFactory.decodeResource(resources, R.drawable.lena256)

    var picX = (this.width - picture.width) / 2f
    var picY = (this.height - picture.height) / 2f

    var paint = Paint()
    var array = floatArrayOf( 2f, 0f, 0f, 0f, -25f,
                              0f, 2f, 0f, 0f, -25f,
                              0f, 0f, 2f, 0f, -25f,
                              0f, 0f, 0f, 1f,   0f )
    var cm = ColorMatrix(array)
    paint.colorFilter = ColorMatrixColorFilter(cm)
    canvas.drawBitmap(picture, picX, picY, paint)
    picture.recycle()
}
```

03 이미지 활용

- **<실습 9-2> 미니 포토샵 앱 만들기**
  - 앞에서 배운 다양한 이미지 처리 방법을 적용하여 포토샵과 비슷한 기능을 하는 간단한 앱 만들기

- 안드로이드 프로젝트 생성
  - (1) 새 프로젝트 만들기
    - 프로젝트 이름 : 'Project9_2'
    - 패키지 이름 : 'com.cookandroid.project9_2'
    - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름

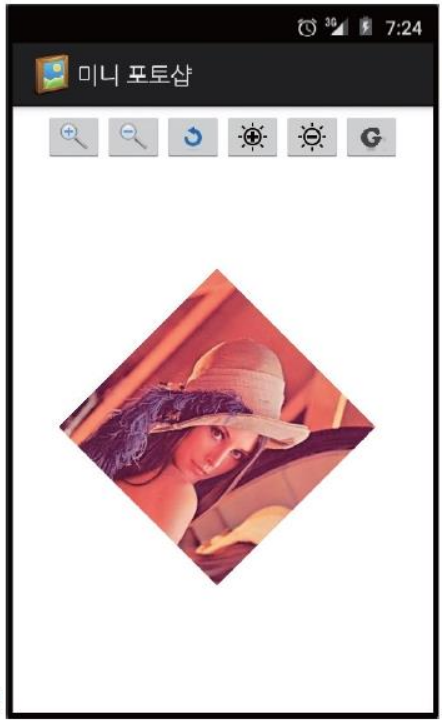


그림 9-8 미니 포토샵 앱 결과 화면

03 이미지 활용

- 화면 디자인 및 편집
  - (2) 확대, 축소, 회전, 밝게 하기, 어둡게 하기, 회색 영상 등 6개 아이콘으로 사용할 그림 파일과 앱 제목의 아이콘으로 사용할 그림 파일, 화면에 출력할 그림 파일을 [res]-[drawable] 폴더에 복사해놓음
  - (3) AndroidManifest.xml 파일에서 아이콘으로 사용할 그림 파일의 id로 변경하고, 블러링 및 엠보싱 효과를 위해 하드웨어 가속기 기능 끔

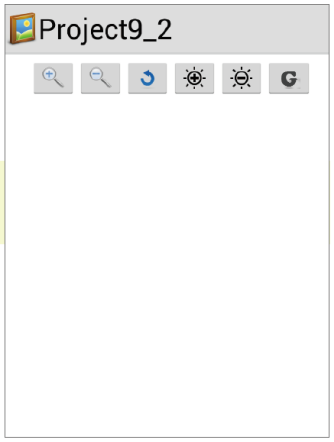
```
android:icon="@drawable/그림 파일 id"
android:hardwareAccelerated="false"
```

  - (4) activity_main.xml을 다음과 같이 수정
    - 바깥 리니어레이아웃 안에 리니어레이아웃 2개 생성
    - 두 리니어레이아웃의 layout_weight : 1:9
    - 위쪽 리니어레이아웃에 이미지버튼 6개 생성
    - 위젯의 id를 다음과 같이 선언함
      - » 리니어레이아웃: iconLayout, pictureLayout
      - » 이미지버튼: ibZoomin, ibZoomout, ibRotate, ibBright, ibDark, ibGray

03 이미지 활용

예제 9-11 activity_main.xml

```
1  <LinearLayout>
2      <LinearLayout
3          android:id="@+id/iconLayout"
4          android:layout_width="fill_parent"
5          android:layout_height="0dip"
6          android:layout_weight="1"
7          android:gravity="center"
8
9          <ImageButton
10             android:id="@+id/ibZoomin"
11             android:src="@drawable/zoom_in" />
12
13         ~~~ 생략(이미지버튼 5개) ~~~
14
15      </LinearLayout>
16
17     <LinearLayout
18         android:id="@+id/pictureLayout"
19         android:layout_width="fill_parent"
20         android:layout_height="0dip"
21         android:layout_weight="9"
22         android:gravity="center" >
23     </LinearLayout>
24 </LinearLayout>
```



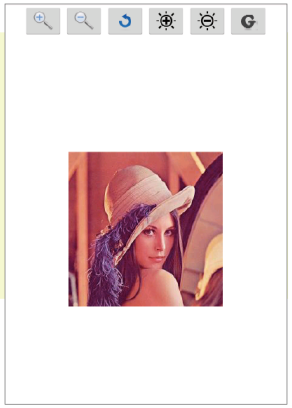
03 이미지 활용

- Kotlin 코드 작성 및 수정
  - (5) MainActivity.kt를 코딩함
    - 사용할 전역변수와 MyGraphicView 클래스를 만들고 그림이 중앙에 위치하는지 확인
    - 이미지버튼에 대응할 위젯 변수 6개 선언
      - » ibZoomin, ibZoomout, ibRotate, ibBright, ibDark, ibGray
    - MyGraphicView 클래스 변수 선언
    - MyGraphicView 클래스를 정의함
      - » 그림을 중앙에 비트맵으로 출력함
    - pictureLayout을 인플레이트한 후 MyGraphicView 추가

03 이미지 활용

예제 9-12 Kotlin 코드 1

```
1  class MainActivity : AppCompatActivity() {
2      lateinit var ibZoomin : ImageButton
3      lateinit var ibZoomout : ImageButton
4      lateinit var ibRotate : ImageButton
5      lateinit var ibBright : ImageButton
6      lateinit var ibDark : ImageButton
7      lateinit var ibGray : ImageButton
8      lateinit var graphicView : MyGraphicView
9
10     override fun onCreate(savedInstanceState: Bundle?) {
11         super.onCreate(savedInstanceState)
12         setContentView(R.layout.activity_main)
13         title = "미니 포토샵"
14
15         var pictureLayout = findViewById<LinearLayout>(R.id.pictureLayout)
16         graphicView = MyGraphicView(this)
17         pictureLayout.addView(graphicView)
18
19     }
20
21     class MyGraphicView(context: Context) : View(context) {
22         override fun onDraw(canvas: Canvas) {
23             super.onDraw(canvas)
24
25             var picture = BitmapFactory.decodeResource(resources,
26                         R.drawable.lena256)
27             var picX = (this.width - picture.width) / 2f
28             var picY = (this.height - picture.height) / 2f
29             canvas.drawBitmap(picture, picX, picY, null)
30
31             picture.recycle()
32         }
33     }
34 }
```

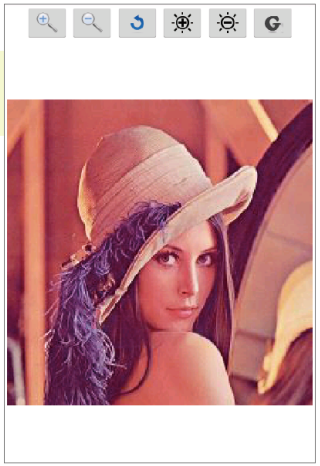


03 이미지 활용

- (6) 확대 아이콘에 대해 코딩
  - 6개의 아이콘을 클릭할 때의 내용을 onCreate() 안에 모두 코딩하면 복잡해지므로 clickIcons() 메소드 만들기
    - 축척에 사용될 전역변수 선언
    - clickIcons( ) 메소드를 정의하고 확대 아이콘 클릭 람다식 생성
    - clickIcons( ) 메소드를 호출
    - onDraw( )에 Cavas.scale( ) 메소드 추가

예제 9-13 Kotlin 코드 2

```
1  ~~~ 생략 ~~~
2  companion object {
3      var sX = 1f
4      var sY = 1f
5  }
6
7  ~~~ 생략 ~~~
8      clickIcons()
9  }
10
11 private fun clickIcons() {
```



03 이미지 활용

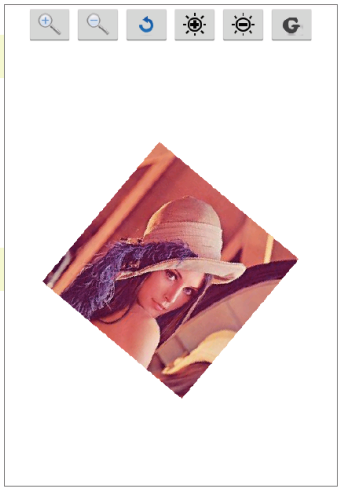
```
12      ibZoomin = findViewById<ImageButton>(R.id.ibZoomin)
13      ibZoomin.setOnClickListener {
14          sX = sX + 0.2f
15          sY = sY + 0.2f
16          graphicView.invalidate()
17      }
18
19  }
20      ~~~ 생략 ~~~
21      var cenX = this.width / 2f
22      var cenY = this.height / 2f
23      canvas.scale(sX, sY, cenX, cenY)
```

03 이미지 활용

- (7) 축소 아이콘에 대해서도 코딩
- (8) 회전 아이콘 코딩
  - 회전에 사용될 전역변수 선언
  - 회전 아이콘 클릭 람다식 생성
  - onDraw( )에 Cavas.rotate( ) 메소드 추가

예제 9-14 미니 포토샵의 Kotlin 코드 3

```
1  ~~~ 생략 ~~~
2  var angle = 0f
3
4  ~~~ 생략 ~~~
5  ibRotate = findViewById<View>(R.id.ibRotate) as ImageButton
6  ibRotate.setOnClickListener {
7      angle = angle + 20
8      graphicView.invalidate()
9  }
10
11 ~~~ 생략 ~~~
12     canvas.rotate(angle, cenX, cenY)
```



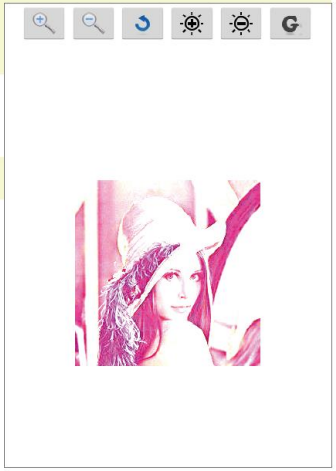
03 이미지 활용

- (9) 밝게 하기 아이콘 코딩
  - 화면 밝기에 사용될 전역변수 선언
  - 밝게 하기 아이콘 클릭 람다식 생성
  - onDraw(　)에 컬러매트릭스 적용

03 이미지 활용

예제 9-15 미니 포토샵의 Kotlin 코드 4

```
1  ~~~ 생략 ~~~
2  var color = 1f
3
4  ~~~ 생략(clickIcons() 메소드와 동일) ~~~
5  ibBright = findViewById<View>(R.id.ibBright) as ImageButton
6  ibBright.setOnClickListener {
7      color = color + 0.2f
8      graphicView.invalidate()
9  }
10
11 ~~~ 생략 ~~~
12     val paint = Paint()
13     val array = floatArrayOf(  color , 0f    , 0f    , 0f    , 0f,
14                                0f    , color , 0f    , 0f    , 0f,
15                                0f    , 0f    , color , 0f    , 0f,
16                                0f    , 0f    , 0f    , 1f    , 0f )
17     val cm = ColorMatrix(array)
18     paint.colorFilter = ColorMatrixColorFilter(cm)
19
20 ~~~ 생략 ~~~
21     canvas.drawBitmap(picture, picX, picY, paint)
```

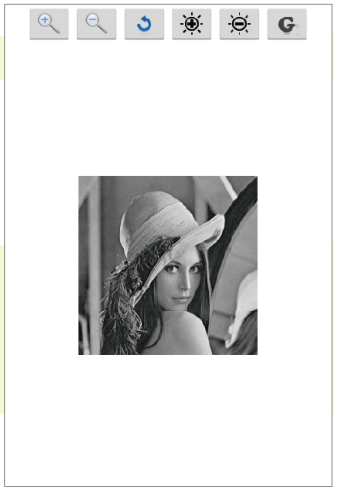


03 이미지 활용

- (10) 어둡게 하기 아이콘 코딩
- (11) 회색 영상으로 이미지를 변경하는 아이콘 코딩
  - 채도에 사용될 전역변수 선언
  - 회색 영상 아이콘 클릭 람다식 생성
  - onDraw( )에 채도 설정 적용

예제 9-16 Kotlin 코드 5

```
1  ~~~ 생략 ~~~
2  var satur = 1f
3
4  ~~~ 생략 ~~~
5  ibGray = findViewById<ImageButton>(R.id.ibGray)
6  ibGray.setOnClickListener {
7      if (satur == 0f)
8          satur = 1f
9      else
10         satur = 0f
11     graphicView.invalidate()
12 }
13
14     ~~~ 생략 ~~~
15     if (satur == 0f) cm.setSaturation(satur)
```



03 이미지 활용

- 프로젝트 실행 및 결과 확인
- (12) 완성된 코드를 실행해봄. [그림 9-8]과 같이 간단한 이미지 처리가 실행됨

직접 풀어보기 9-3

[실습 9-2]를 다음과 같이 수정하라.

- 회색 영상 아이콘을 없앤다.
- 밝게 하기 아이콘을 클릭하면 채도가 높아지고, 어둡게 하기 아이콘을 클릭하면 채도가 낮아지게 한다.
- 블러링, 엠보싱 아이콘을 추가하고, 클릭하면 블러링 또는 엠보싱 기능이 온/오프되게 한다.

그림 9-9 수정된 미니 포토샵 앱

Thank You !