2D Graphic (I)

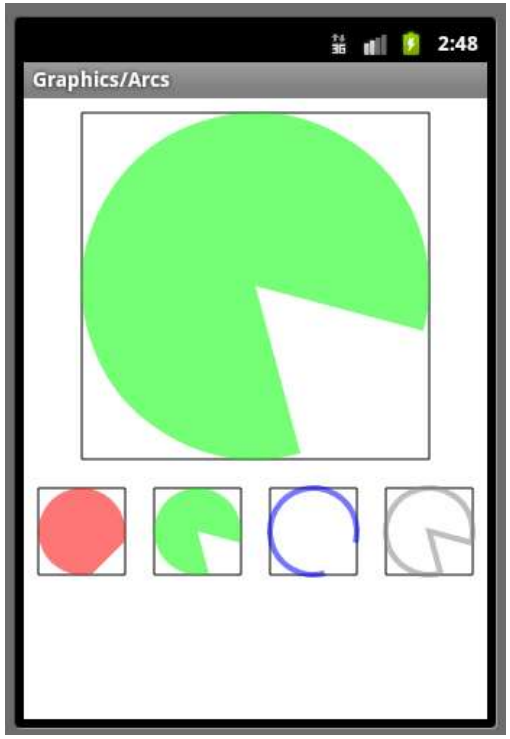
Mobile Software
2019 Fall

What to do next?

- **그래픽 기초**
  - **custom view를 사용하여 도형 그리기**
- 속성, 색상, 폰트
- 이미지 출력 및 변환
- Nine-patch 이미지

2D 그래픽 구현 방법

- Canvas에 직접 그림
  - **onDraw**() 에서 그래픽 메소드 호출
- XML 파일에서 그래픽 또는 애니메이션을 정의



Canvas에 직접 그릴 때 구현 방법

- UI activity에서 custom view를 생성
  - **invalidate** ( ) 호출 ➔ **onDraw**( )
    - **onDraw**()를 직접 호출할 수 없기 때문에

- 별도 화면(**SurfaceView**)과 thread 생성
  - **UI thread와 그래픽을 각각 따로 처리**
    - invalidate ( ) 를 호출할 필요가 없음

Custom view 방식 coding style

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // setContentView(R.layout.activity_main)

        val myView = MyView(this)
        setContentView(myView)
    }

    class MyView(context: Context): View(context) {
        override fun onDraw(canvas: Canvas) {...}
    }
}
```

MyView 객체의 내용을 화면에 출력

중첩 클래스: View 클래스 상속

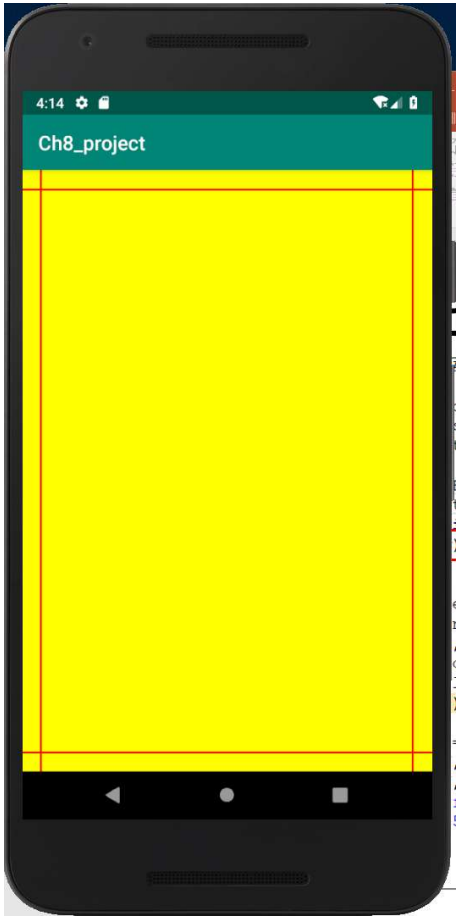
onDraw 메소드의 parameter는 Canvas 객체!

그래픽 코드 삽입

실습 1: 간단한 도형

MainActivity.kt

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // setContentView(R.layout.activity_main)

        val myView = MyView(this)
        setContentView(myView)
    }

    class MyView(context: Context): View(context) {
        override fun onDraw(canvas: Canvas) {
            super.onDraw(canvas)

            canvas.drawRGB(255, 255, 0)
            val w = width.toFloat()
            val h = height.toFloat()
            val paint = Paint()
            paint.setARGB(255, 255, 0, 0)
            paint.strokeWidth = 5f
            canvas.drawLine(0f, 50f, w, 50f, paint)
            canvas.drawLine(0f, h-50f, w, h-50f, paint)
            canvas.drawLine(50f, 0f, 50f, h, paint)
            canvas.drawLine(w-50f, 0f, w-50f, h, paint)
        }
    }
}
```



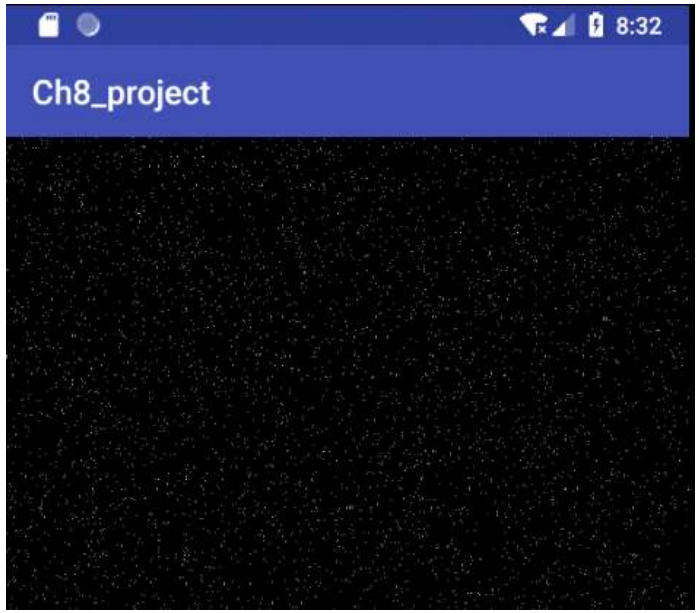
실습 2: 은하수

MainActivity.kt

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // setContentView(R.layout.activity_main)

        val myView = MyView(this)
        setContentView(myView)
    }
}
```

MyView.kt

```
class MyView(context: Context): View(context) {
    init {
        setBackgroundColor(Color.BLACK)
    }

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

        val paint = Paint()
        paint.setARGB(255, 255, 255, 255)

        for (x in 1..10000) {
            var dx = (Math.random() * width).toFloat()
            var dy = (Math.random() * height).toFloat()
            canvas.drawPoint(dx,dy, paint)
        }
    }
}
```



Canvas 와 Paint

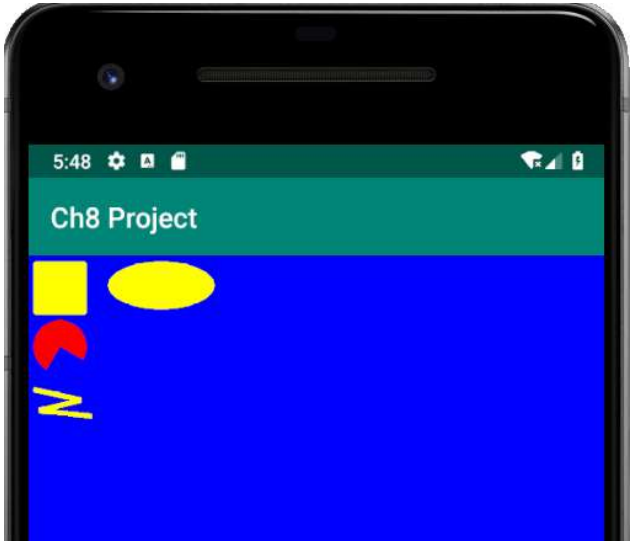
- **Canvas** 클래스
  - 그림이 그려지는 캔버스
    - 캔버스는 Bitmap 객체를 갖고 있음.

- **Paint** 클래스
  - 그리기 속성을 정의 (색상, 선 스타일, 채우기 등)
    - 그림을 그리는 붓이나 물감에 해당

실습 3: 복잡한 도형

MyView.kt

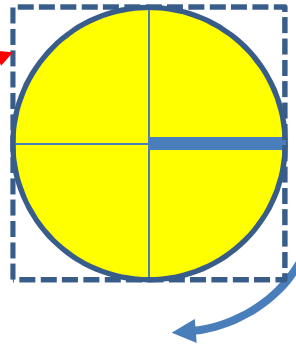
```
class MyView(context: Context) : View(context) {
    override fun onDraw(canvas: Canvas) {
        canvas.drawRGB(0, 0, 255)

        val paint = Paint()
        val r1 = RectF(10f,  10f, 110f, 110f)
        val r2 = RectF(150f, 10f, 350f, 100f)
        val r3 = RectF(10f, 120f, 110f, 220f)

        paint.color = Color.YELLOW
        canvas.drawRoundRect(r1, 5f, 5f, paint)
        canvas.drawOval(r2, paint)

        paint.color = Color.RED
        canvas.drawArc(r3, 120f, 270f, true, paint)

        paint.color = Color.YELLOW
        val pts = floatArrayOf(
                    10f,  250f, 100f, 270f,
                    100f, 270f,  20f, 290f,
                    20f,  290f, 120f, 300f)
        paint.strokeWidth = 10f
        canvas.drawLines(pts, paint)
    }
}
```



잠깐! drawArc(호), drawOval(타원)

drawArc

```
void drawArc (RectF oval,
              float startAngle,
              float sweepAngle,
              boolean useCenter,
              Paint paint)
```



startAngle = 0
(*3 o'clock on a watch*)

sweepAngle 은
시계방향으로 증가

useCenter = *false* ➔ *호 (외곽선)*
useCenter = *true* ➔*부채꼴 (채우기)*

0,90 0,180 0,270

useCenter = *true*

---

drawOval

```
void drawOval (RectF oval,
               Paint paint)
```

사각형에 내접하는 타원

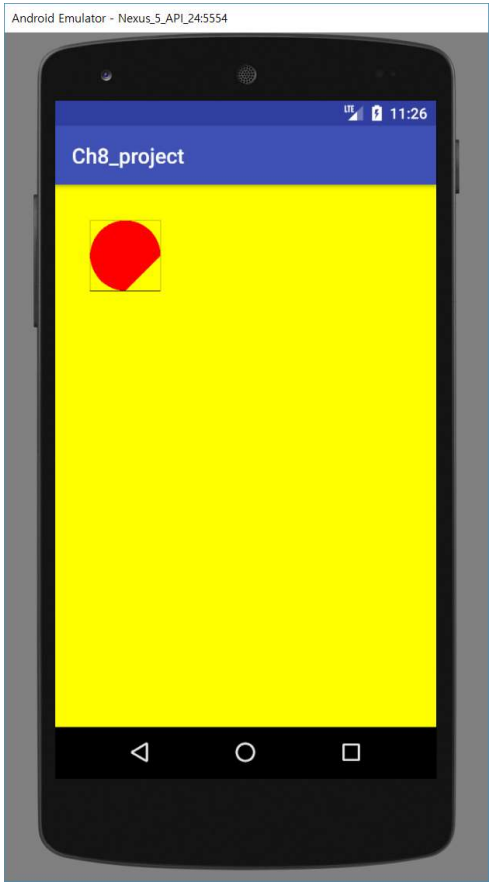
실습 4: Custom view

custom view를 XML layout 파일에서 참조

MyView.kt

```
class MyView(context: Context, attrs:AttributeSet):
            View(context, attrs) {
    override fun onDraw(canvas: Canvas) {
        canvas.drawRGB(255, 255, 0)

        val paint = Paint()
        val r = RectF(100f, 100f, 300f, 300f)

        paint.style = Paint.Style.STROKE
        canvas.drawRect(r, paint)

        paint.color = Color.RED
        paint.style = Paint.Style.FILL
        canvas.drawArc(r, 90f, 270f, false, paint)
    }
}
```

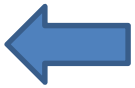
The constructor with *Context* and *AttributeSet* is used **when your view is inflated from xml**.



실습 4: Layout + Activity

activity_main.xml

```
<android.support.constraint.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

    <edu.ourincheon.ch8project.MyView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```



Custom view 객체

MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

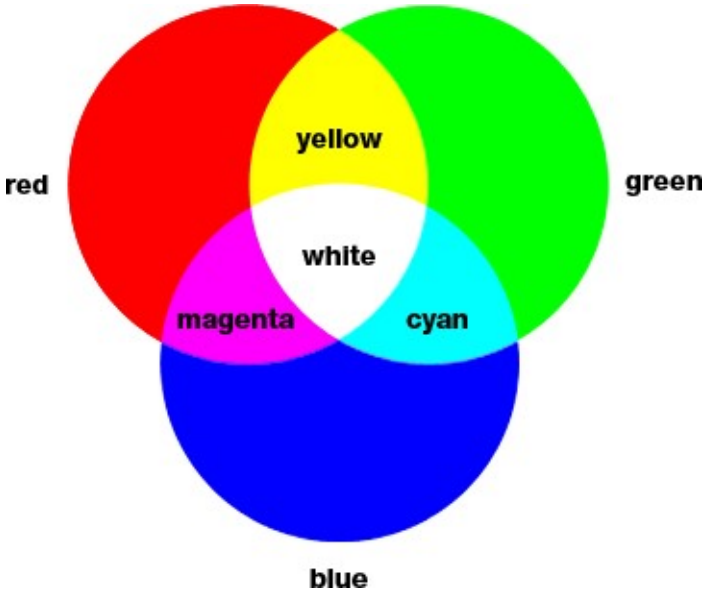
What to do next?

- 그래픽 기초
  - custom view를 사용하여 도형 그리기
- **속성, 색상, 폰트**
- 이미지 출력 및 변환
- Nine-patch 이미지

색상

- 색의 3원색인 Red, Green, Blue 성분을 8비트(0~255)로 표시

```
paint. color = 0xFF0000
paint. color = Color. RED
var c = paint. color
paint. alpha = 255
```



선 스타일

- paint. **style** = *Paint. Style.* ***FILL***
- Paint. **strokeWidth** = 10f

FILL	도형 내부를 채운다.
FILL_AND_STROKE	도형 내부를 채우면서 외곽선도 그린다.
STROKE	도형 외곽선만 그린다.

폰트(font)

method

static Typeface **create** (Typeface family, int style)

static Typeface **create** (String familyName, int style)

Typeface 객체는 Typeface 클래스 내부의 create() 메소드로 생성된다.

family: **DEFAULT, DEFAULT_BOLD, MONOSPACE, SANS_SERIF, SERIF**

style: **NORMAL, BOLD, ITALIC, BOLD_ITALIC**

지정한 폰트와 스타일에 가장 일치하는 Typeface 객체를 생성한다.

Typeface **setTypeface** (Typeface typeface)

Font 를 typeface로 변경한다.

void **drawText** (String text, float x, float y, **Paint paint**)

void **drawText** (String text, int start, int end, float x, float y, **Paint paint**)

void **drawText** (char[ ] text, int index, int count, float x, float y, **Paint paint**)

화면에 텍스트를 그린다.

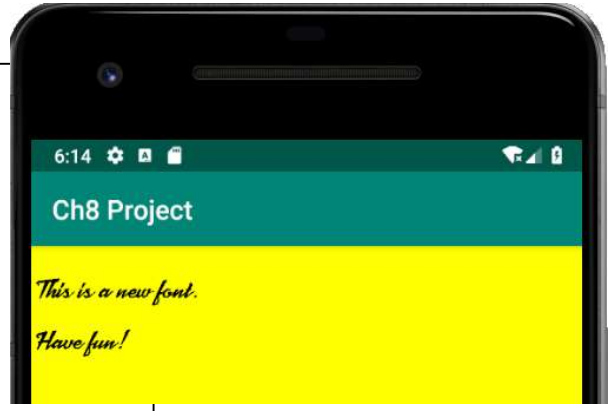
실습 5: font

```
class MyView(context: Context): View(context) {
    init {
        setBackgroundColor(Color.YELLOW)
    }

    override fun onDraw(canvas: Canvas) {
        val paint = Paint()
        paint.textSize = 50f

        var t = Typeface.create(Typeface.DEFAULT, Typeface.NORMAL)
        paint.typeface = t
        canvas.drawText("DEFAULT 폰트", 10f, 100f, paint)

        t = Typeface.create(Typeface.DEFAULT_BOLD, Typeface.NORMAL)
        paint.typeface = t
        canvas.drawText("DEFAULT_BOLD 폰트", 10f, 200f, paint)

        t = Typeface.create(Typeface.MONOSPACE, Typeface.NORMAL)
        paint.typeface = t
        canvas.drawText("MONOSPACE 폰트", 10f, 300f, paint)

        t = Typeface.create(Typeface.SERIF, Typeface.NORMAL)
        paint.typeface = t
        canvas.drawText("SERIF 폰트", 10f, 400f, paint)

        t = Typeface.create(Typeface.SANS_SERIF, Typeface.NORMAL)
        paint.typeface = t
        canvas.drawText("SANS_SERIF 폰트", 10f, 500f, paint)
    }
}
```

실습 6: 외부 폰트

- Google 검색 : "android font download"
  - http://www.fontspace.com → Scriptonite Demo.zip 다운로드
- 압축을 푼 다음 ScriptoniteDemo.ttf 를 assets 폴더에 복사

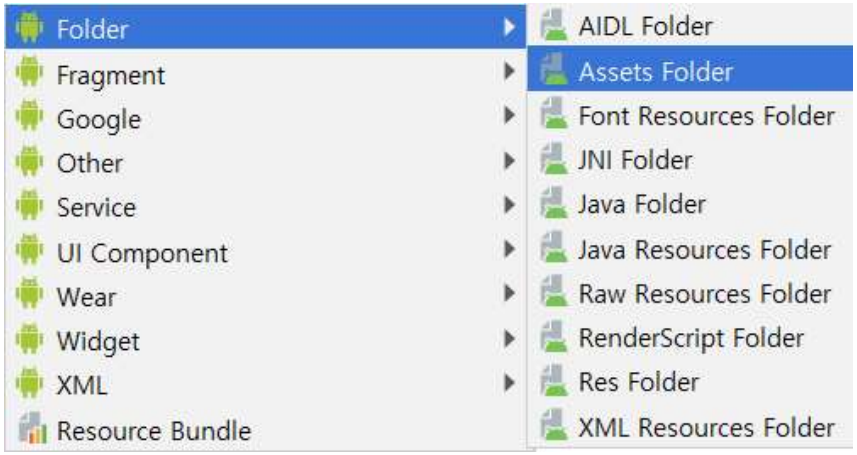
```
class MyView(context: Context): View(context) {
    init {
        setBackgroundColor(Color.YELLOW)
    }

    override fun onDraw(canvas: Canvas) {
        val paint = Paint()
        paint.textSize = 50f

        try {
            val myFont = Typeface.createFromAsset(
                context.assets, "ScriptoniteDemo.ttf"
            )
            paint.typeface = myFont
            canvas.drawText("This is a new font.", 10f, 100f, paint)
            canvas.drawText("Have fun!", 10f, 200f, paint)
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }
}
```

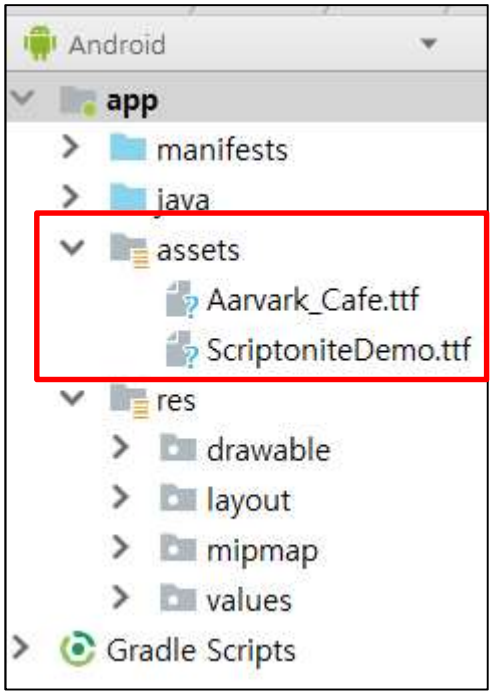


폰트 파일 이름에 공백이 포함되면 안되므로, 복사할 때 공백을 제거

잠깐! assets 폴더 만드는 방법

Android view로 전환
app ➔ 오른쪽 버튼
➔ **New** ➔ (메뉴 아래쪽) ➔ **Folder**
➔ **Assets Folder** ➔ **Finish**





What to do next?

- 그래픽 기초
  - custom view를 사용하여 도형 그리기
- 속성, 색상, 폰트
- **이미지 출력 및 변환**
- Nine-patch 이미지

Image 리소스를 화면에 출력

- Drawable 리소스에 포함된 이미지를
  - Bitmap 객체로 변환

```
val b: Bitmap = BitmapFactory.decodeResource(resources,
                    R.drawable.harubang)
```

  - Bitmap 객체로 변환한 이미지를 메모리에 저장

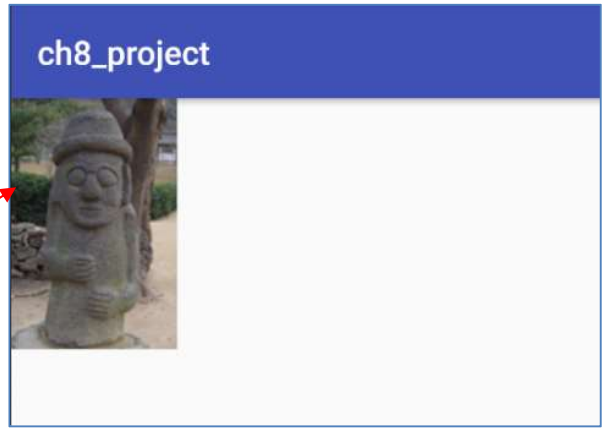
- Bitmap 객체를 canvas에 출력

```
canvas.drawBitmap(b, 0f, 0f, null)
```

실습 7(a): 이미지 리소스를 bitmap 객체로 변환하여 출력

MyView.kt

```
class MyView(context: Context): View(context) {
    override fun onDraw(canvas: Canvas) {
        canvas.drawRGB(255, 255, 255)

        val b = BitmapFactory.decodeResource(resources,
            R.drawable.harubang)
        canvas.drawBitmap(b, 0f, 0f, null)
    }
}
```



실습 7(b): bitmap 객체 확대 및 축소

MyView.kt

```
class MyView(context: Context): View(context) {
    val b: Bitmap = BitmapFactory.decodeResource(
        resources, R.drawable.harubang)

    init {
        setBackgroundColor(Color.LTGRAY)
    }

    override fun onDraw(canvas: Canvas) {
        super.onDraw(canvas)
        val w = b.width
        val h = b.height

        var px = 10f
        var py = 10f     // 원본 이미지 출력 위치
        canvas.drawBitmap(b, px, py, null)

        px += w          // 원점을 w 만큼 왼쪽으로 이동

        // 원본 이미지를 1/2로 축소해서 출력
        var pi = px.toInt()
        var pj = py.toInt()
        val shape1 = Rect(pi, pj, pi + w/2, pj + h/2)
        canvas.drawBitmap(b, null, shape1, null)

        pj += h      // // 원점을 h 만큼 아래로 이동
        // 원본 이미지를 2배로 확대해서 출력
        val shape2 = Rect(pi, pj, pi + 2*w, pj + 2*h)
        canvas.drawBitmap(b, null, shape2, null)
    }
}
```

잠깐! drawBitmap의 4개 parameters

```
drawBitmap(Bitmap bitmap, Rect src, Rect dst, Paint paint)
```

Draw the specified bitmap, scaling/translating automatically to fill the destination rectangle.

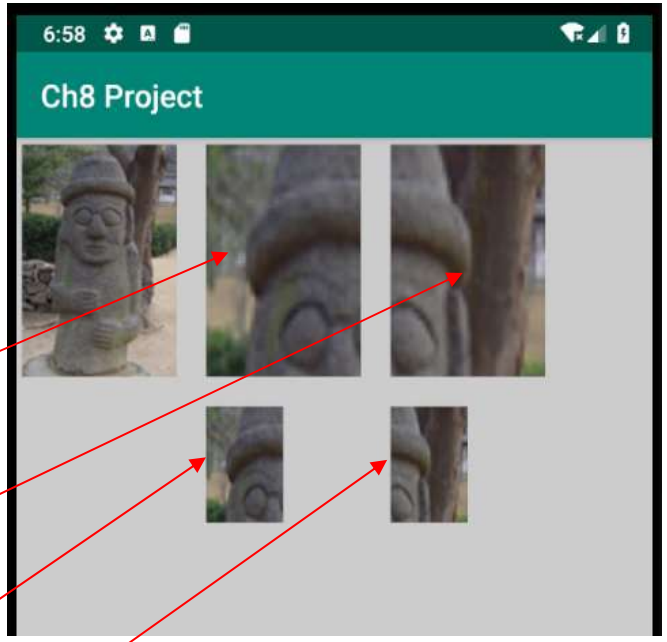
Parameters	
`bitmap`	`Bitmap`: The bitmap to be drawn This value must never be `null`.
`src`	`Rect`: May be null. The subset of the bitmap to be drawn
`dst`	`Rect`: The rectangle that the bitmap will be scaled/translated to fit into This value must never be `null`.
`paint`	`Paint`: May be null. The paint used to draw the bitmap

첫 번째, 세 번째 parameter는 절대 null이 되면 안 됨.

2번째, 4번째 parameter는 null이 될 수 있음.

실습 7(c): 부분 이미지 확대, 축소

```
override fun onDraw(canvas: Canvas) {
    val w = b.width // 하루방 이미지 너비
    val h = b.height// 하루방 이미지 높이
    var px = 10f    // 하루방 이미지 원점 x 좌표
    var py = 10f    // 하루방 이미지 원점 y 좌표
    canvas.drawBitmap(b, px, py, null)

    val src = Rect(40, 40, 140, 140)     // 하루방 왼쪽 눈 부분
    val src2 = Rect(140, 40, 240, 140)   // 하루방 오른쪽 눈 부분

    val incWidth = w + 50
    var new_pi = (px + incWidth).toInt()

    var pi = new_pi
    var pj = py.toInt()

    val dst = Rect(pi, pj, pi + w, pj + h)
    canvas.drawBitmap(b, src, dst, null)

    pi += incWidth
    val dst3 = Rect(pi, pj, pi + w, pj + h)
    canvas.drawBitmap(b, src2, dst3, null)

    pi = new_pi
    pj += (h + 50)
    val dst2 = Rect(pi, pj, pi + w/2, pj + h/2)
    canvas.drawBitmap(b, src, dst2, null)

    pi += incWidth
    val dst4 = Rect(pi, pj, pi + w/2, pj + h/2)
    canvas.drawBitmap(b, src2, dst4, null)
}
```



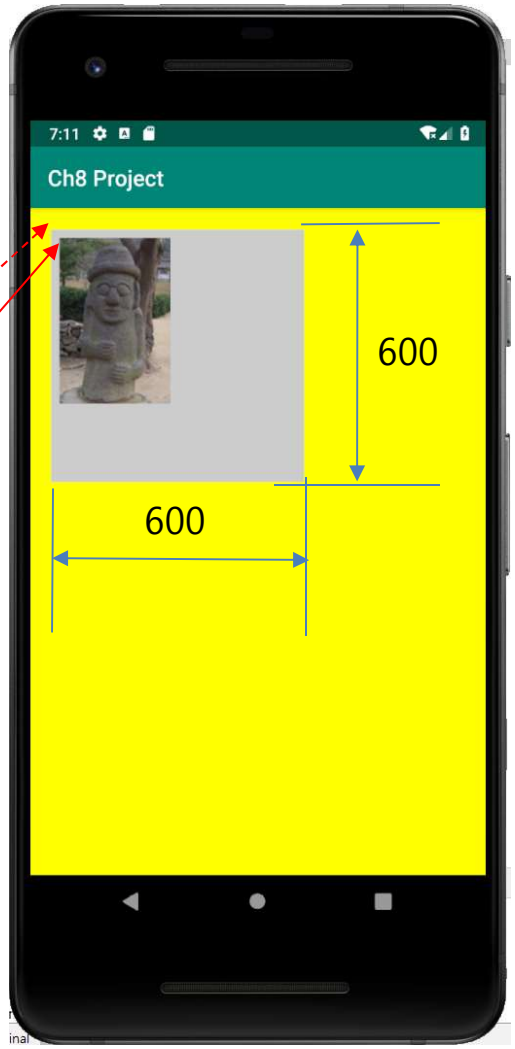
Bitmap image를 직접 만들고 캔버스에 출력

- bitmap 이미지를 출력할 공간을 만들고 이미지를 출력
  - **createBitmap**
    - Bitmap 크기와 색상 encoding 방식 지정 ➔ bitmap 객체 생성
    - Canvas 객체 **mCanvas**를 생성하고,
      - **mCanvas**에 그림 ➔ **mCanvas**에 그리는 그림은 메모리에 저장됨.
    - **drawBitmap** 메소드를 사용하여 화면에 출력

- **createBitmap**을 실행하는 2가지 방법
  - **onDraw** 메소드에서 **createBitmap** 실행
    - Layout이 화면에 나타난 다음 ➔ 메모리에 저장된 bitmap 이미지 출력
  - **onSizeChanged** 메소드에서 **createBitmap** 실행
    - Layout이 화면에 나타나기 전에 실행
      - drawBitmap을 사용해 미리 메모리에 bitmap 이미지를 그림

실습 8: bitmap 공간 생성 ➔ bitmap 이미지 출력 : onDraw

```
class MyView(context: Context): View(context) {
    val b: Bitmap = BitmapFactory.decodeResource(resources,
        R.drawable.harubang)

    init {
        setBackgroundColor(Color.YELLOW)
    }

    override fun onDraw(canvas: Canvas) {
        val width = 600
        val height = 600

        val mBitmap = Bitmap.createBitmap(
            width, height, Bitmap.Config.ARGB_8888)
        val mCanvas = Canvas(mBitmap)
        mCanvas.drawColor(Color.LTGRAY)
        mCanvas.drawBitmap(b, 20f, 20f, null)

        canvas.drawBitmap(mBitmap, 50f, 50f, null)
    }
}
```



실습 9: bitmap 공간 생성 ➔ bitmap 이미지 출력 : onSizeChanged

```
class MyView(context: Context): View(context) {
    private val b: Bitmap = BitmapFactory.decodeResource(
                        resources, R.drawable.harubang)
    private val mBitmap: Bitmap = Bitmap.createBitmap(
        600, 600, Bitmap.Config.ARGB_8888)

    init {
        setBackgroundColor(Color.WHITE)
    }

    override fun onDraw(canvas: Canvas) {
        super.onDraw(canvas)
        canvas.drawBitmap(mBitmap, 50f, 50f, null)
    }

    override fun onSizeChanged(w: Int, h: Int, oldw: Int, oldh: Int) {
        super.onSizeChanged(w, h, oldw, oldh)

        val mCanvas = Canvas(mBitmap)
        mCanvas.drawColor(Color.LTGRAY)
        mCanvas.drawBitmap(b, 20f, 20f, null)
    }
}
```

잠깐! ARGB_8888이 뭐예요?

```
Bitmap bmp_space =
        Bitmap.createBitmap(width, height,
                Bitmap.Config.ARGB_8888);
```

Possible bitmap configurations. A bitmap configuration describes how pixels are stored. This affects the quality (color depth) as well as the ability to display transparent/translucent colors.

ALPHA_8 Each pixel is stored as a single translucency (alpha) channel.
ARGB_4444 *This field was deprecated in API level 13. Because of the poor quality of this configuration, it is advised to use* ARGB_8888 *instead.*
ARGB_8888 Each pixel is stored on 4 bytes.
RGB_565 Each pixel is stored on 2 bytes and only the RGB channels are encoded: red is stored with 5 bits of precision (32 possible values), green is stored with 6 bits of precision (64 possible values) and blue is stored with 5 bits of precision.

실습 10: bitmap 공간 생성 ➔ bitmap 출력 : Activity에서 직접 그리기

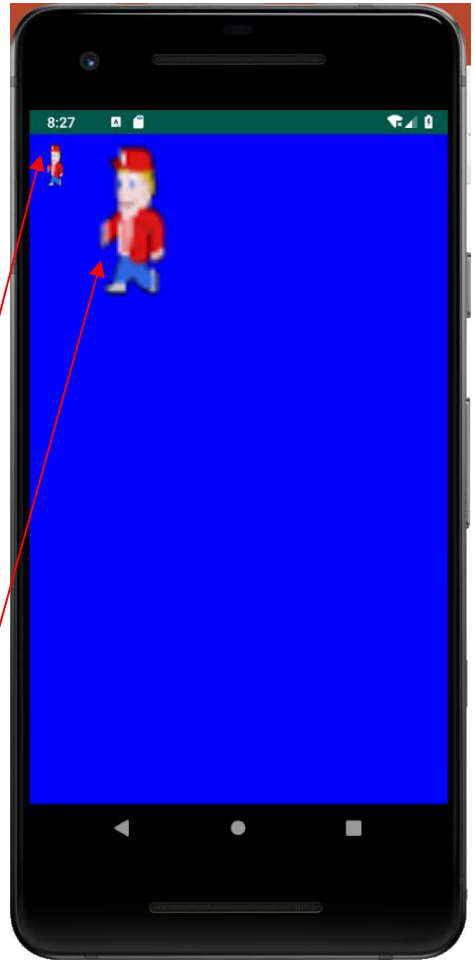
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    val myImageView = ImageView(this)

    val myBlankBitmap = Bitmap.createBitmap(
        600, 600,
        Bitmap.Config.ARGB_8888)

    val dm = applicationContext.resources.displayMetrics
    val screenBitmap = Bitmap.createBitmap(
        dm.widthPixels,
        dm.heightPixels,
        Bitmap.Config.ARGB_8888 )

    val scCanvas = Canvas(screenBitmap)
    scCanvas.drawColor(Color.argb(255, 255, 255, 0))

    val myCanvas = Canvas(myBlankBitmap)
    myCanvas.drawColor(Color.LTGRAY)

    val harubang = BitmapFactory.decodeResource(
        resources, R.drawable.harubang)
    myCanvas.drawBitmap(harubang, 20f, 20f, null)
    scCanvas.drawBitmap(myBlankBitmap, 50f, 50f, null)

    myImageView.setImageBitmap(screenBitmap)
    setContentView(myImageView)
}
```

작업 공간 (600x600)

전체 공간

실습 11(a): 이미지 확대

```
class MyView(context: Context): View(context) {

    var myBlankBitmap = Bitmap.createBitmap(
        2000, 2000,
        Bitmap.Config.ARGB_8888)

    var bobBitmap = BitmapFactory.decodeResource(
        resources, R.drawable.bob)
    var myCanvas = Canvas(myBlankBitmap)

    override fun onDraw(canvas: Canvas) {
        myCanvas.drawColor(Color.BLUE)

        drawEnlargedBitmap()
        canvas.drawBitmap(myBlankBitmap, 0f, 0f, null)
    }

    fun drawEnlargedBitmap() {
        val w = bobBitmap.width
        myCanvas.drawBitmap(bobBitmap, 25f, 25f, null)

        bobBitmap = Bitmap
            .createScaledBitmap(
                bobBitmap,
                300, 400, false
            )
        myCanvas.drawBitmap(bobBitmap, w+25f, 25f, null)

    }
}
```



잠깐! drawBitmap, createScaledBitmap

drawBitmap

```
void drawBitmap (Bitmap bitmap,
                 float left,
                 float top,
                 Paint paint)
```

(*left, top*)

Parameters	
bitmap	Bitmap: The bitmap to be drawn
left	float: The position of the left side of the bitmap being drawn
top	float: The position of the top side of the bitmap being drawn
paint	Paint: The paint used to draw the bitmap (may be null)

createScaledBitmap

```
Bitmap createScaledBitmap (Bitmap src,
                           int dstWidth,
                           int dstHeight,
                           boolean filter)
```

dstWidth	int: The new bitmap's desired width.
dstHeight	int: The new bitmap's desired height.
filter	boolean: true if the source should be filtered.

```
Bitmap sb = Bitmap.createScaledBitmap(b, 60, 80, false);
Bitmap sb = Bitmap.createScaledBitmap(b, b.getWidth()*2, b.getHeight()*2, true);
```

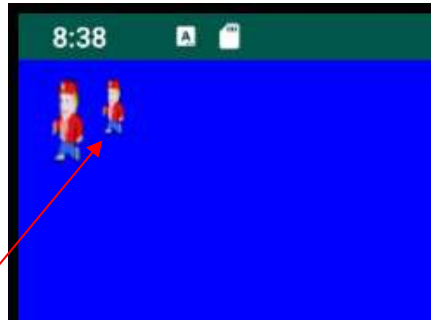
원본 이미지보다 축소할 때는 상관없지만, 크게 확대할 때

Passing `filter = false` will result in a blocky, pixellated image.

Passing `filter = true` will give you smoother edges.

실습 11(b): 이미지 축소

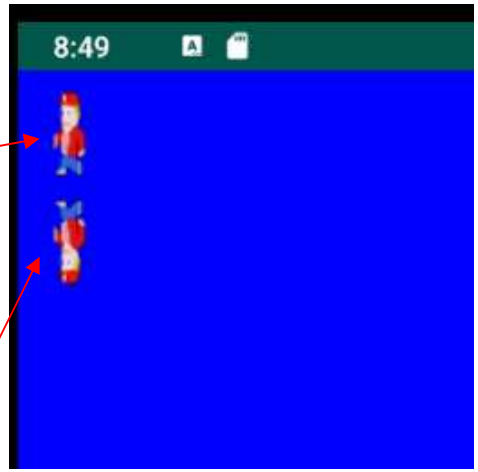
```
fun drawShrunkenBitmap() {
    val w = bobBitmap.width
    myCanvas.drawBitmap(bobBitmap, 25f, 25f, null)

    bobBitmap = Bitmap
        .createScaledBitmap(
            bobBitmap,
            50, 75, false
        )
    myCanvas.drawBitmap(bobBitmap, w+25f, 25f, null)
}
```



실습 12(a): 이미지 상하 반전 – matrix 사용

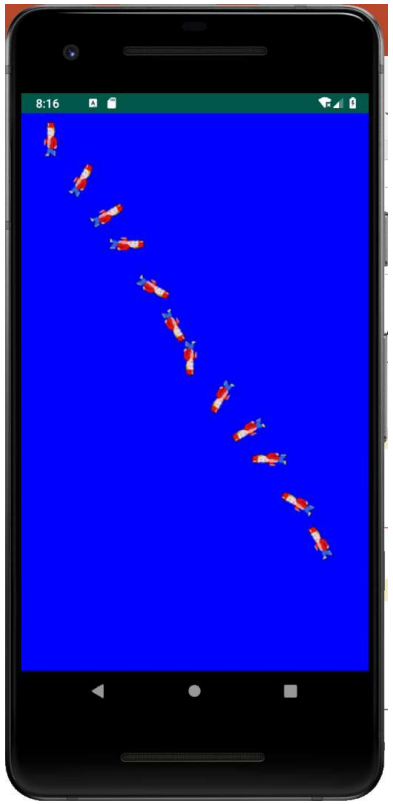
- 변환 행렬 사용 : x값은 그대로, y값은 -1을 곱함

```
fun drawMirrorBitmap() {
    val w = bobBitmap.width
    val h = bobBitmap.height
    myCanvas.drawBitmap(bobBitmap, 25f, 25f, null)

    val matrix = Matrix()
    matrix.preScale(1f, -1f)
    bobBitmap = Bitmap.createBitmap(
            bobBitmap,
            0, 0, w, h, matrix, false)

    myCanvas.drawBitmap(bobBitmap, 25f, h+50f, null)
}
```



실습 12(b): 이미지 회전 – matrix 사용

```
fun drawRotatedBitmap() {
    val w = bobBitmap.width
    val h = bobBitmap.height
    var rotatedBitmap: Bitmap

    var rotation = 0f
    var horizontalPos = 50f
    var verticalPos = 25f
    val matrix = Matrix()

    while (rotation < 360) {
        matrix.reset()

        // M' = M*R(degrees), 시계방향 회전
        matrix.preRotate(rotation)

        // 원점(0,0), 너비(w), 높이(h)
        // 위치 이동만이 아닌 다른 변환도 포함되었을 경우에만 true 설정
        rotatedBitmap = Bitmap.createBitmap(
                bobBitmap, 0, 0, w, h, matrix, true)

        // 이미지 출력 위치(horizontalPos, verticalPos)
        // 회전할 때마다 오른쪽으로 120, 아래쪽으로 70씩 이동
        myCanvas.drawBitmap(
            rotatedBitmap,
            horizontalPos, verticalPos, null)

        horizontalPos += w
        verticalPos += h
        rotation += 30f
    }
}
```



잠깐! createBitmap

createBitmap

```
Bitmap createBitmap (Bitmap source,
                     int x,
                     int y,
                     int width,
                     int height,
                     Matrix m,
                     boolean filter)
```

width	int: The number of pixels in each row
height	int: The number of rows
m	Matrix: Optional matrix to be applied to the pixels

```
Matrix m = new Matrix();
m.preScale(1, -1);
Bitmap mb = Bitmap.createBitmap(b, 0, 0,
        b.getWidth(), b.getHeight(), m, false);
```

기타(1/2): Filter

- Filter – mask filter, color filter
- **Mask filter**
  - alpha 채널(투명도)만을 변경
    - BlurMaskFilter – 흐릿하게 만듦
    - EmbossMaskFilter - 올록볼록하게 만듦

- **Color Filter**
  - RGB 채널의 색상 요소만을 변경
    - LightingColorFilter
    - **ColorMatrixColorFilter**
      - Color matrix : 4x5 행렬 ➔ 각 행은 R, G, B, A에 해당
    - PorterDuffColorFilter

기타(2/2): transformation(변환)

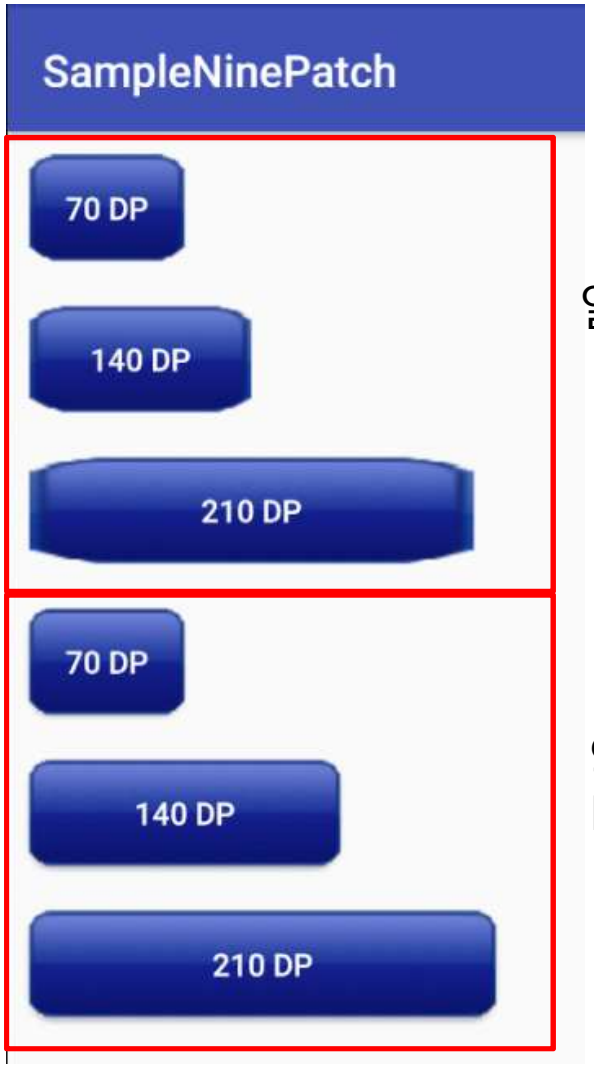
- Canvas 객체에서 제공하는 변환 메소드
  - 이동(translate)
    - void **translate** (*float dx, float dy*);
  - 확대(scale)
    - void **scale** (*float sx, float sy* [, *float px, float py*]);
      - (px, py) : 확대 기준 좌표
      - 생략하면 원점(0,0)이 기준 좌표
  - 회전(rotate)
    - void **rotate** (float degrees [, *float px, float py*]);
      - 시계 방향(clockwise)으로 회전
  - save, restore
    - int **save** ([*int saveFlags*]);
      - 변환 전 정보를 저장
    - void **restore** ( );
    - void **restoreToCount** (*int saveCount*);

What to do next?

- 그래픽 기초
  - custom view를 사용하여 도형 그리기
- 속성, 색상, 폰트
- 이미지 출력 및 변환
- **Nine-patch 이미지**

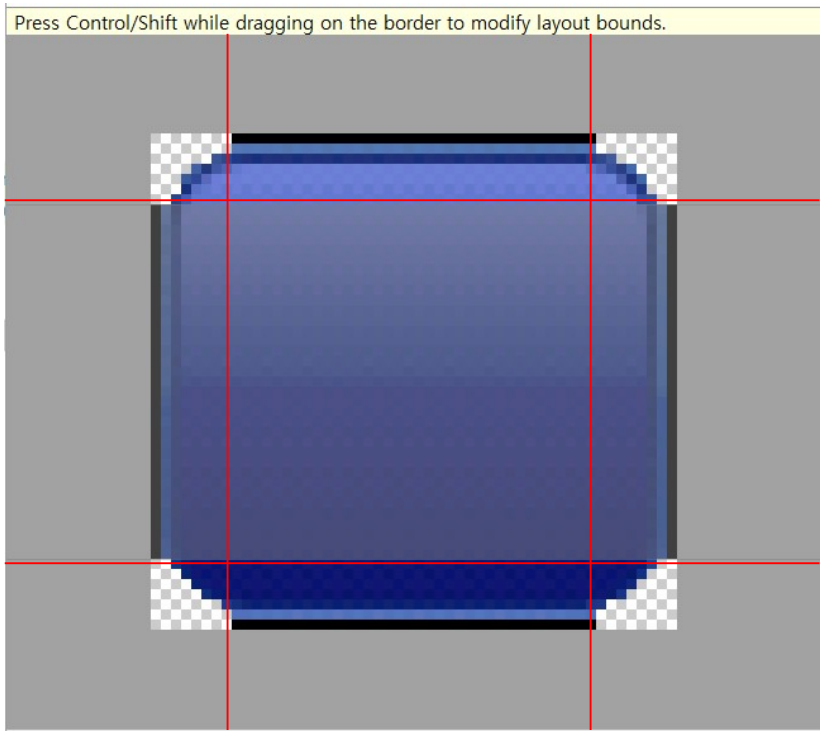
Nine-patch 이미지란?

- 원본 이미지를 크게 하거나 작게 할 때
  - 4군데 모서리에 나타나는 이미지 굴곡(distortion)을 해결하기 위한 방법
- 파일 확장자
  - xxx.9.png
    - 이미지 리소스를 참조할 때는 숫자 9 생략



Nine-patch 파일 = resizable bitmap

- PNG 파일 클릭
  - 오른쪽 버튼 > **Create 9-patch file** > 파일 이름 지정
    - Drawable 폴더에 xxx.9.png 파일이 새로 만들어짐
- 9-patch 파일 **더블 클릭**
  - 수평 또는 수직 라인을 움직여 patch 크기를 pixel단위로 조절
  - 4곳 모서리(patch)는 이미지 resizing 결과에 상관없이 크기가 일정하게 유지.



Stretchable patches