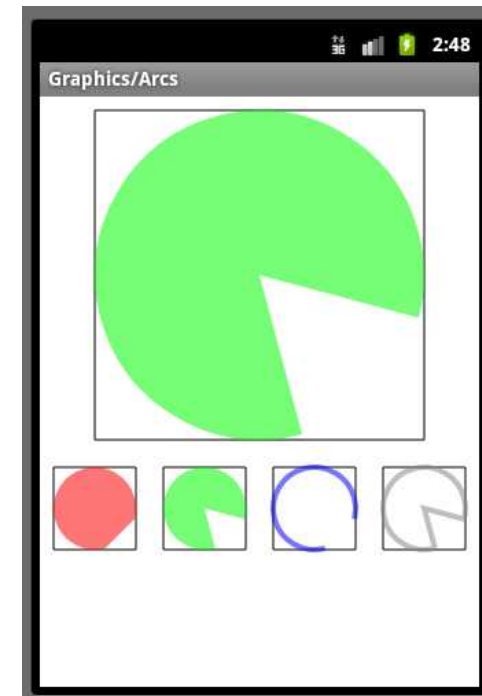


## CHAP 6. 그래픽



# 안드로이드에서의 그래픽

- XML 파일로 그래픽이나 애니메이션을 정의한다. 그리는 작업은 안드로이드 시스템이 담당한다.
- `onDraw()` 메소드 안에 `draw...()`와 같은 메소드를 호출하여 직접 그린다.



# 전체적인 구조

```
class MyView extends View {  
    ...  
    protected void onDraw(Canvas canvas) {  
        Paint paint = new Paint();  
  
        ...  
    }  
}  
  
public class MainActivity extends AppCompatActivity {  
    public void onCreate(Bundle is) {  
        ...  
        MyView w = new MyView(this);  
        setContentView(w);  
    }  
}
```



전체적인 구조

여기에 그림을 그리는 코드를 넣는다.

MyView를 생성하고 이것을 Activity의 콘텐츠 뷰로 설정한다.

# Canvas 클래스와 Paint 클래스

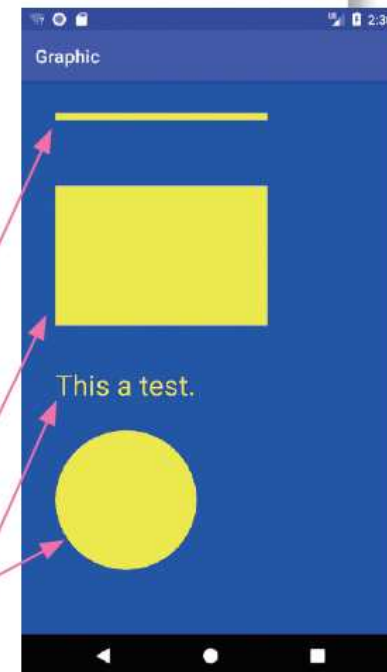
- Canvas 클래스는 그림을 그리는 캔버스(화포)에 해당
- Paint 클래스는 색상이나 선의 스타일, 채우기 스타일, 폰트, 안티앨리어싱 여부 등과 같은 그리기 속성을 가지고 있는 클래스



# 예제

MainActivity.java

```
package kr.co.company.graphic;  
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.  
  
class MyView extends View {  
    public MyView(Context context) {  
        super(context);  
        setBackgroundColor(Color.BLUE);  
    }  
  
    @Override  
    protected void onDraw(Canvas canvas) {  
        Paint paint = new Paint();  
        paint.setColor(Color.YELLOW);  
        paint.setStrokeWidth(20);  
        canvas.drawLine(100, 100, 700, 100, paint);  
        canvas.drawRect(100, 300, 700, 700, paint);  
        canvas.drawCircle(300, 1200, 200, paint);  
        paint.setTextSize(80);  
        canvas.drawText("This is a test.", 100, 900, paint);  
    }  
}
```



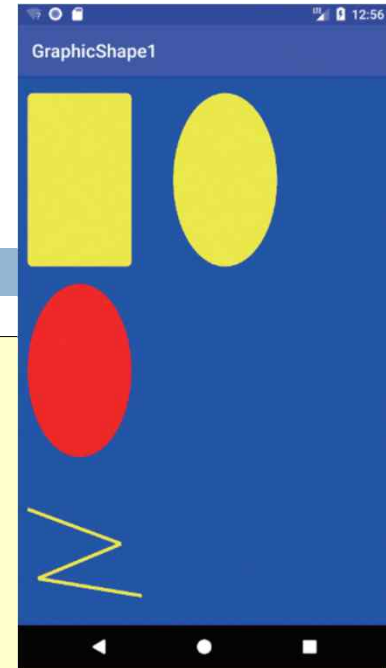
# 예제

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        MyView w = new MyView(this);  
        setContentView(w);  
    }  
}
```

# 몇 개의 기초 도형 그리기

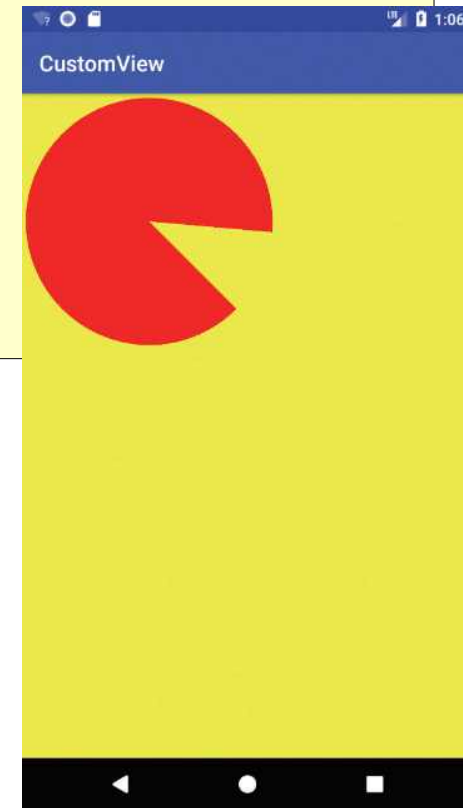
@Override

```
protected void onDraw(Canvas canvas) {  
    Paint paint = new Paint();  
    paint.setColor(Color.YELLOW);  
  
    canvas.drawColor(Color.BLUE);  
    canvas.drawRoundRect(new RectF(30,50,330,550), 15, 15, paint);  
    canvas.drawOval(new RectF(450,50,750,550), paint);  
    paint.setColor(Color.RED);  
    canvas.drawArc(new RectF(30,600,330,1100), 360, 1000,  
        true, paint);  
    paint.setColor(Color.YELLOW);  
    float[] pts={ 30, 1250, 300, 1350, 300, 1350, 60, 1450,  
        60, 1450, 360, 1500};  
    paint.setStrokeWidth(10);  
    canvas.drawLines(pts, paint);  
}
```



# 커스텀 뷰를 XML에서 참조하기

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <kr.co.company.customview.CustomView
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```





# 커스텀 뷰를 XML에서 참조하기

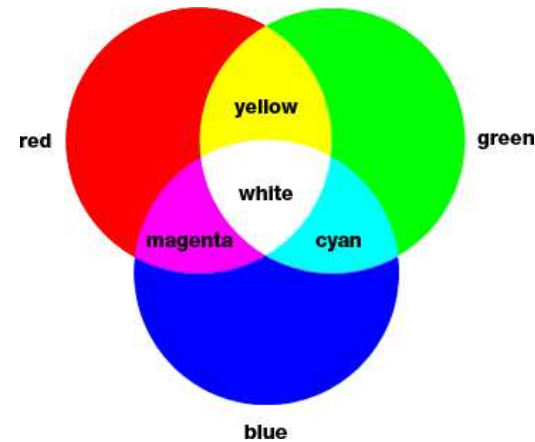
```
...  
public class CustomView extends View {  
    public CustomView(Context context) {  
        super(context);  
        setBackgroundColor(Color.YELLOW);  
    }  
    public CustomView(Context context, AttributeSet attrs) {  
        super(context);  
        setBackgroundColor(Color.YELLOW);  
    }  
    @Override  
    protected void onDraw(Canvas canvas) {  
        Paint paint = new Paint();  
        paint.setColor(Color.RED);  
        canvas.drawArc(new RectF(10, 120, 110, 220), 120, 270, true, paint);  
    }  
}
```



# 색상

- 색의 3원색인 Red, Green, Blue 성분을 8비트로 표시

- `paint.setColor(0xFF0000);`
- `paint.setColor(Color.RED);`



# 선의 스타일

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

# 원호그리기

```
class MyView extends View {  
    private Paint mPaints, mFramePaint;  
    private RectF mBigOval;  
    private float mStart, mSweep;  
    private static final float SWEEP_INC = 2;  
    private static final float START_INC = 15;  
    public MyView(Context context) {  
        super(context);  
        mPaints = new Paint();  
        mPaints.setAntiAlias(true);  
        mPaints.setStyle(Paint.Style.FILL);  
        mPaints.setColor(0x88FF0000);  
        mFramePaint = new Paint();  
        mFramePaint.setAntiAlias(true);  
        mFramePaint.setStyle(Paint.Style.STROKE);  
        mFramePaint.setStrokeWidth(3);  
        mFramePaint.setColor(0x8800FF00);  
        mBigOval = new RectF(100, 40, 900, 1000);  
    }  
}
```

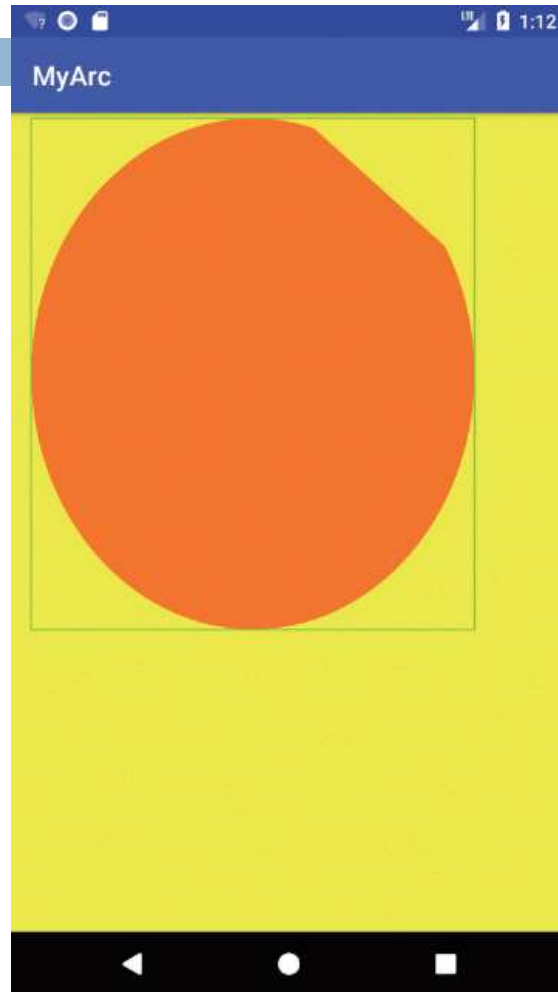
# 원호그리기

@Override

```
protected void onDraw(Canvas canvas) {
    canvas.drawColor(Color.YELLOW);
    canvas.drawRect(mBigOval, mFramePaint);
    canvas.drawArc(mBigOval, mStart, mSweep, false, mPaints);
    mSweep += SWEEP_INC;
    if (mSweep > 360) {
        mSweep -= 360;
        mStart += START_INC;
        if (mStart >= 360)
            mStart -= 360;
    }
    invalidate();
}
}

public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new MyView(this));
    }
}
```

# 원호그리기



# 폰트

## 메소드

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

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

Typeface 객체는 Typeface 클래스 내부의 create() 메소드로 생성된다.

**family:** 폰트 이름 , DEFAULT, DEFAULT\_BOLD, MONOSPACE, SANS\_SERIF, SERIF 중의 하나

**style:** 폰트 스타일, NORMAL, BOLD, ITALIC, BOLD\_ITALIC 중의 하나

위의 메소드들은 주어진 폰트와 스타일에 가장 일치하는 Typeface 객체를 생성한다. 이 메소드는 주로 이미 존재하는 폰트로부터 새로운 스타일의 폰트를 생성하고자 할 때 호출한다. 만약 **family**가 null이면 디폴트 폰트 패밀리를 선택한다.

**Typeface setTypeface (Typeface typeface)**

폰트를 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)**

텍스트를 화면에 그린다.

## 폰트 예제

```
...
@Override
protected void onDraw(Canvas canvas)
{
    Paint paint = new Paint();
    paint.setAntiAlias(true);
    paint.setTextSize(100);
    Typeface t;
    t = Typeface.create(Typeface.DEFAULT, Typeface.NORMAL);
    paint.setTypeface(t);
    canvas.drawText("DEFAULT 폰트", 10, 400, paint);

    t = Typeface.create(Typeface.DEFAULT_BOLD, Typeface.NORMAL);
    paint.setTypeface(t);
    canvas.drawText("DEFAULT BOLD 폰트", 10, 600, paint);

    t = Typeface.create(Typeface.MONOSPACE, Typeface.NORMAL);
    paint.setTypeface(t);
    canvas.drawText("MONOSPACE 폰트", 10, 800, paint);

    t = Typeface.create(Typeface.SERIF, Typeface.NORMAL);
    paint.setTypeface(t);
    canvas.drawText("SERIF 폰트", 10, 1000, paint);

    t = Typeface.create(Typeface.SANS_SERIF, Typeface.NORMAL);
    paint.setTypeface(t);
    canvas.drawText("SANS SERIF 폰트", 10, 1200, paint);
}
```

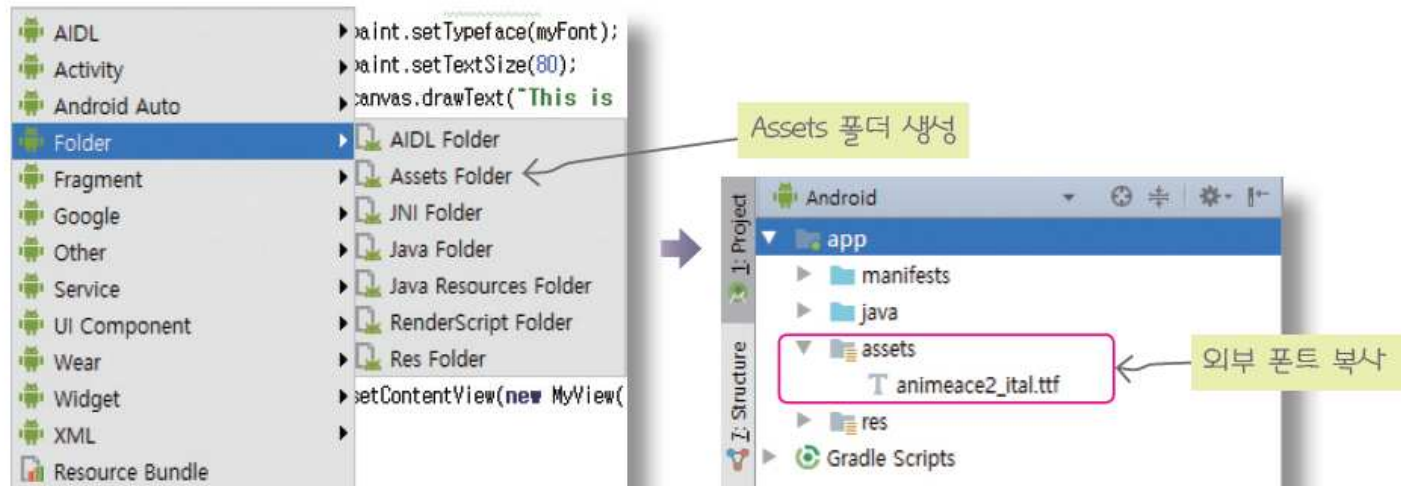


# 폰트 예제 실행 결과



# 외부 폰트

- 폰트 파일을 구하여 프로젝트의 **asset** 폴더로 복사



# 외부 폰트 사용

```
@Override
protected void onDraw(Canvas canvas) {
    Typeface myFont;
    Paint paint = new Paint();
    paint.setAntiAlias(true);
    myFont = Typeface.createFromAsset(getContext().getAssets(),
        "animeace2_ital.ttf");
    paint.setTypeface(myFont);
    paint.setTextSize(25);
    canvas.drawText("This is a New Font!!!", 10, 100, paint);
    canvas.drawText("Have Fun!!!", 10, 200, paint);
}
```



# 패스 그리기

- 패스(path)는 복잡한 기하학적인 경로를 표현
- 패스는 직선과 타원, 곡선으로 이루어질 수 있다

MainActivity.java

```
@Override
protected void onDraw(Canvas canvas) {
    Path path = new Path();
    Paint paint = new Paint();

    paint.setStyle(Paint.Style.STROKE);

    path.moveTo(20, 400);
    path.lineTo(300, 800);
    path.cubicTo(450, 120, 600, 1200, 900, 800);

    paint.setColor(Color.BLUE);
    canvas.drawPath(path, paint);

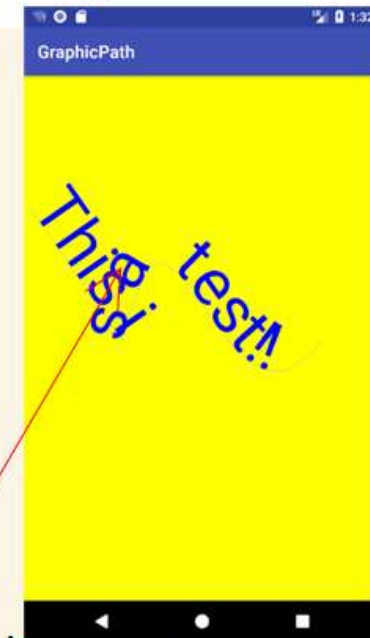
    paint.setStyle(Paint.Style.FILL);
    paint.setTextSize(200);
    canvas.drawTextOnPath("This is a test!!", path, 0, 0, paint);
}
```

패스 생성.

패스에 도형을 추가  
한다.

패스를 그린다.

패스 위에 텍스트를  
그린다.



# 이미지 표시

- 리소스 폴더에 이미지 파일을 복사한다.
  - ▣ 예를 들어서 `android.png` 파일을 프로젝트의 `res/drawable` 폴더에 복사
  - ▣ 프로그램에서는 `R.drawable.android`로 참조
- 지원되는 파일 형식은 **PNG (선호), JPG (가능), GIF (권장되지 않음)**



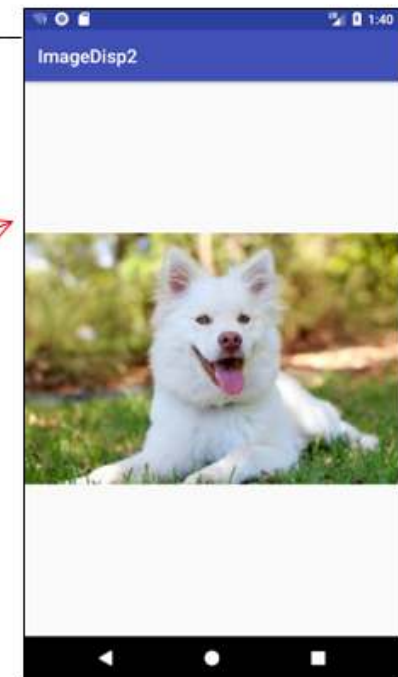
# 이미지 표시하기: ImageView 사용

activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

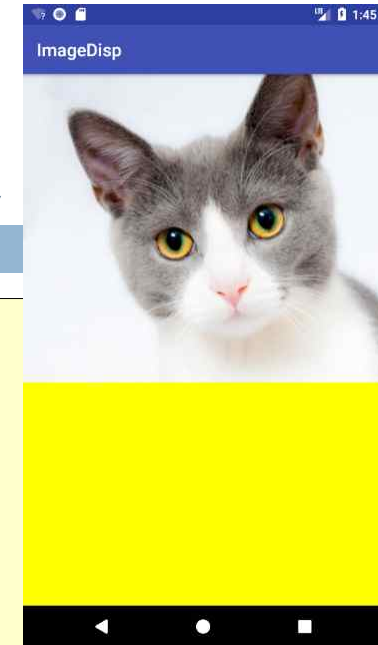
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/dog" />

</LinearLayout>
```



# 코드로 화면에 이미지 표시

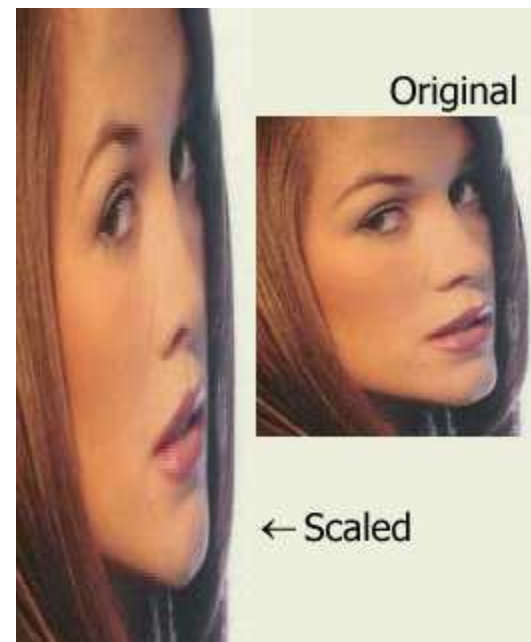
```
class MyView extends View {  
    public MyView(Context context) {  
        super(context);  
        setBackgroundColor(Color.YELLOW);  
    }  
    @Override  
    protected void onDraw(Canvas canvas) {  
        Paint paint = new Paint();  
        Bitmap b = BitmapFactory.decodeResource(getResources(),  
            R.drawable.cat);  
        canvas.drawBitmap(b, 0, 0, null);  
    }  
}  
  
public class ImageDispActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        MyView w = new MyView(this);  
        setContentView(w);  
    }  
}
```



# 이미지 크기 변환

## □ 변환 행렬 사용

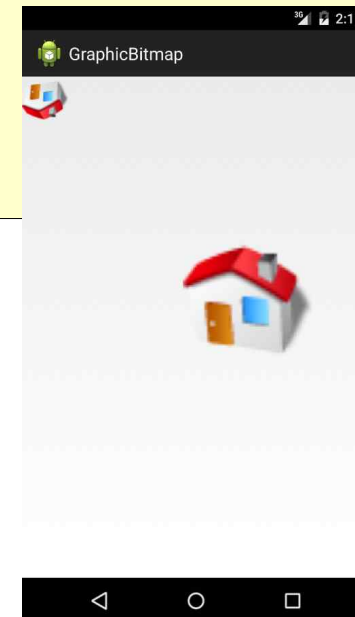
```
...  
Matrix m = new Matrix();m.preScale(1, -1);  
Bitmap b = BitmapFactory.decodeResource(getResources(), R.drawable.android);  
Bitmap mb=Bitmap.createBitmap(b, 0, 0, b.getWidth(), b.getHeight(), m, false);  
...
```





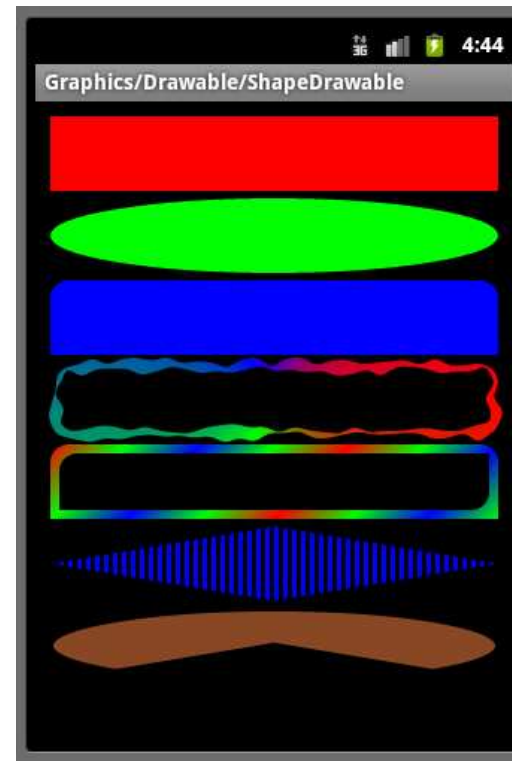
# 이미지 크기 변환

```
...  
protected void onDraw(Canvas canvas) {  
    Paint paint = new Paint();  
    Matrix m= new Matrix();  
    m.preScale(1, -1);  
    Bitmap b = BitmapFactory.decodeResource(getResources(), R.drawable.house);  
    Bitmap mb = Bitmap.createBitmap(b, 0, 0, b.getWidth(), b.getHeight(), m, false);  
    Bitmap sb = Bitmap.createScaledBitmap(b, 200, 200, false);  
    canvas.drawBitmap(mb, 0, 0, null);  
    canvas.drawBitmap(sb, 100, 100, null);  
}  
...
```



# 도형 객체

- 사각형이나 원 같은 도형을 객체로 표시한다.
- Drawable 객체
  - ▣ XML로 객체를 생성
  - ▣ 코드로 객체를 생성



# XML로 도형 객체 정의

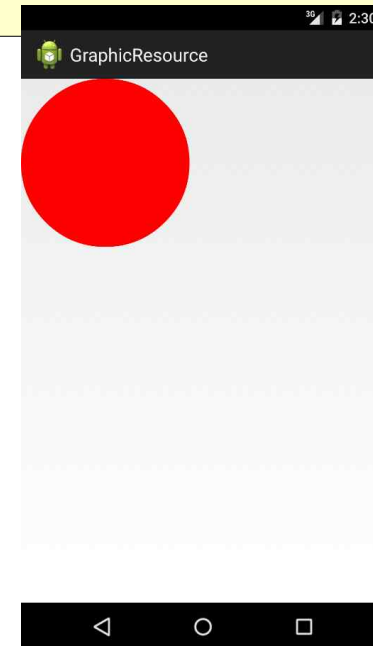
```
<?xml version="1.0" encoding="utf-8"?>

<shape
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:shape=["rectangle" | "oval" | "line" | "ring"] >
  <corners .... />
  <gradient .... />
  <padding .... />
  <size .... />
  <solid android:color="color" />
  <stroke android:width="integer" .... />
</shape>
```

# XML로 도형 객체 정의: 예제

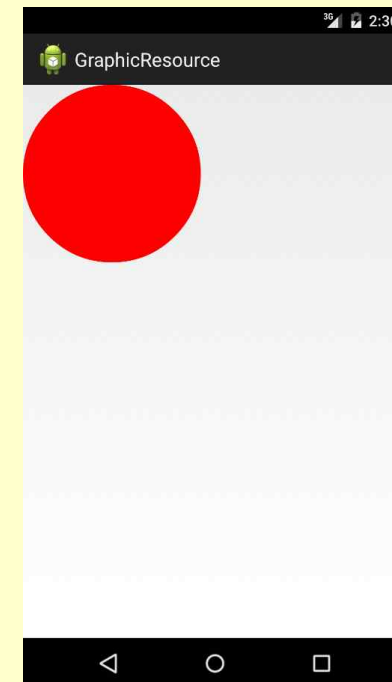
```
<?xml version="1.0" encoding="utf-8"?>

<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="#ff0000"/>
</shape>
```



# XML로 도형 객체 정의: 예제

```
...  
LinearLayout mLinearLayout;  
  
protected void onCreate(Bundle savedInstanceState)  
{  
    super.onCreate(savedInstanceState);  
    mLinearLayout = new LinearLayout(this);  
    ImageView i = new ImageView(this);  
    i.setImageDrawable(R.drawable.oval);  
    i.setMinimumHeight(100);  
    i.setMinimumWidth(100);  
  
    mLinearLayout.addView(i);  
    setContentView(mLinearLayout);  
}  
...
```

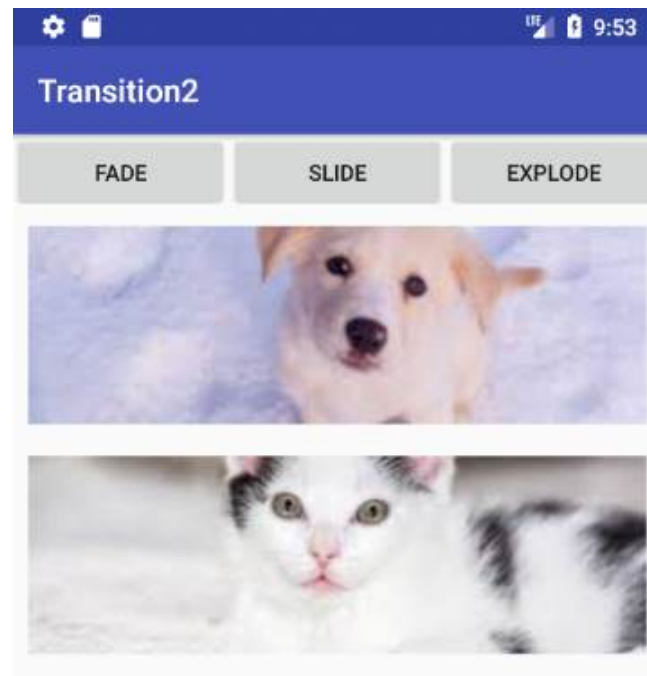


# Transition API 애니메이션

- **Fade** - 페이드 인 및 페이드 아웃과 같은 가장 보편적인 애니메이션을 수행
- **Slide** - 한 방향으로 움직여서 사라진다.
- **Explode** - 폭발하는 것과 같은 효과를 낸다.
- **Auto Transition** - **Fade-out**, **ChangeBounds**, **Fade-in**이 순차적으로 포함된 **TransitionSet**이다. 첫 번째 뷰가 페이드 아웃된 후에 위치 및 크기가 변경되고 마지막으로 새로운 뷰가 페이드 인으로 나타난다.
- **ChangeBounds** - 위치 및 크기를 변경하는 애니메이션
- **TransitionSet** - 여러 개의 전환들을 묶는다.

# 예제

## □ 사용자 인터페이스



# 예제

```
public class MainActivity extends AppCompatActivity {  
  
    private LinearLayout layout;  
    private Button fadeButton, slideButton, explodeButton;  
    private ImageView imageView, imageView2;  
    boolean visible;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        layout = (LinearLayout) findViewById(R.id.layout);  
        fadeButton = (Button) findViewById(R.id.fade);  
        slideButton = (Button) findViewById(R.id.slide);  
        explodeButton = (Button) findViewById(R.id.explode);  
        imageView = (ImageView) findViewById(R.id.imageview);  
    }  
}
```



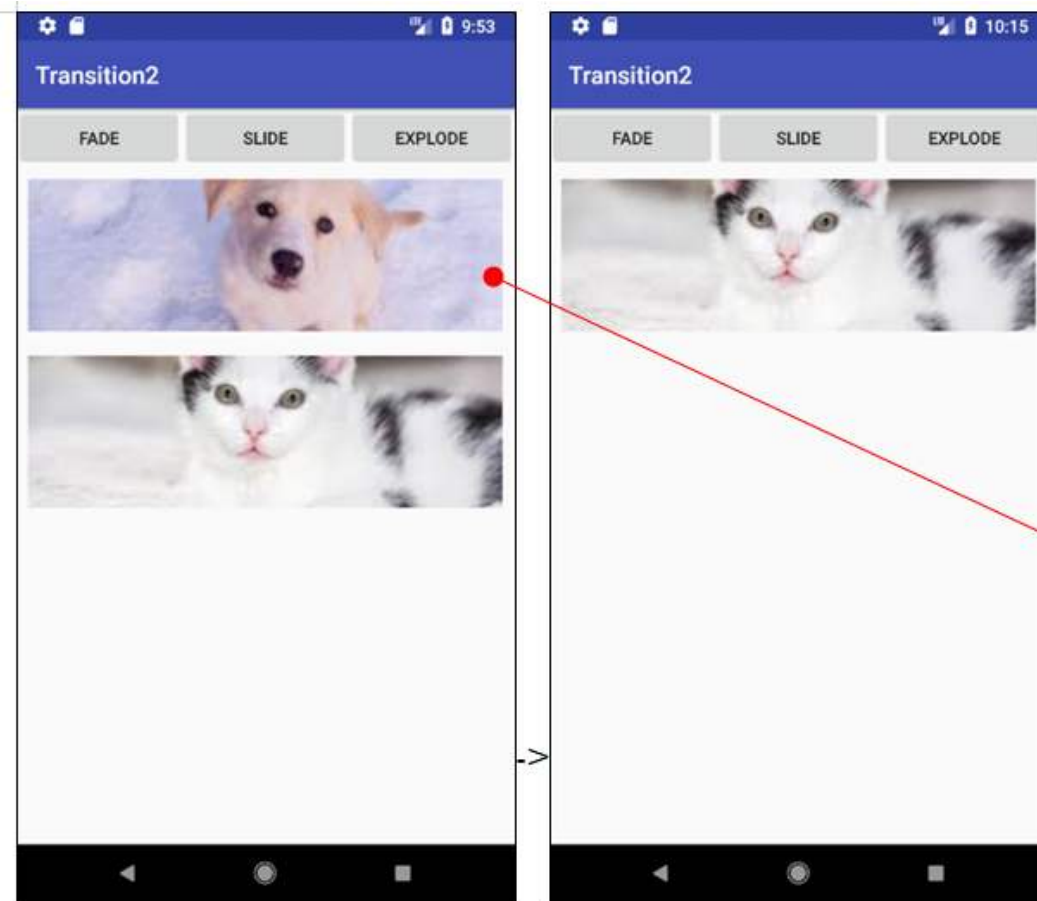
# 예제

```
fadeButton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        TransitionManager.beginDelayedTransition(layout, new Fade());  
        visible = !visible;  
        imageView.setVisibility(visible ? View.VISIBLE : View.GONE);  
    }  
});  
slideButton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        TransitionManager.beginDelayedTransition(layout, new Slide());  
        visible = !visible;  
        imageView.setVisibility(visible ? View.VISIBLE : View.GONE);  
    }  
});
```

# 예제

```
explodeButton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        TransitionManager.beginDelayedTransition(layout, new Explode());  
        visible = !visible;  
        imageView.setVisibility(visible ? View.VISIBLE : View.GONE);  
    }  
});  
}
```

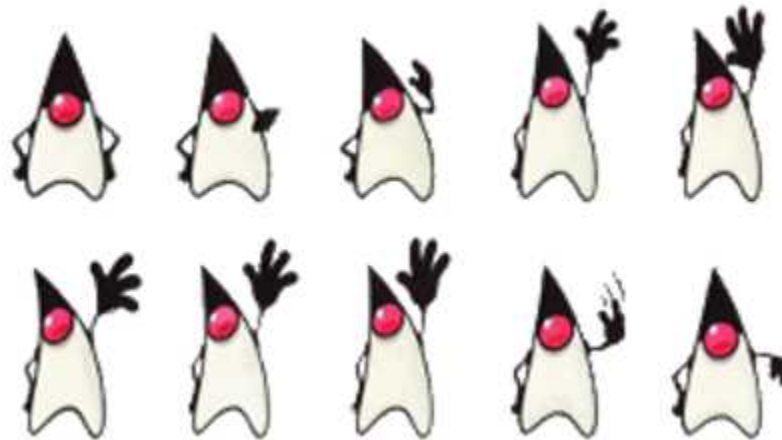
# 실행 결과



전환 효과는 첫 번째 그림에만 적용하였다. 전환 효과는 XML 파일로도 기술이 가능하다. 자세한 내용은 안드로이드 레퍼런스를 참조한다.

# 드로워블 애니메이션

- 영화 필름처럼 여러 개의 이미지가 순서대로 재생되어서 생성되는 전통적인 애니메이션



# 드로워블 애니메이션

- 애니메이션을 구성하는 프레임들을 나열하는 XML 파일을 생성

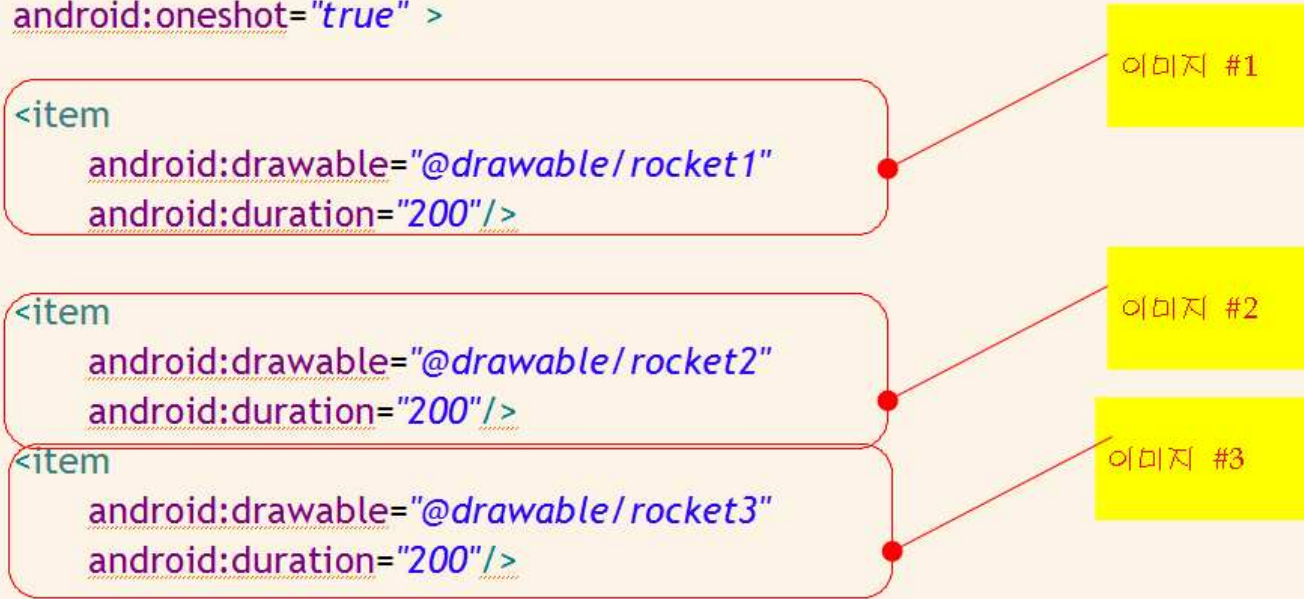
```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true" >

    <item
        android:drawable="@drawable/rocket1"
        android:duration="200"/>

    <item
        android:drawable="@drawable/rocket2"
        android:duration="200"/>

    <item
        android:drawable="@drawable/rocket3"
        android:duration="200"/>

</animation-list>
```



# 예제

```
public class MainActivity extends AppCompatActivity {
    AnimationDrawable rocketAnimation;

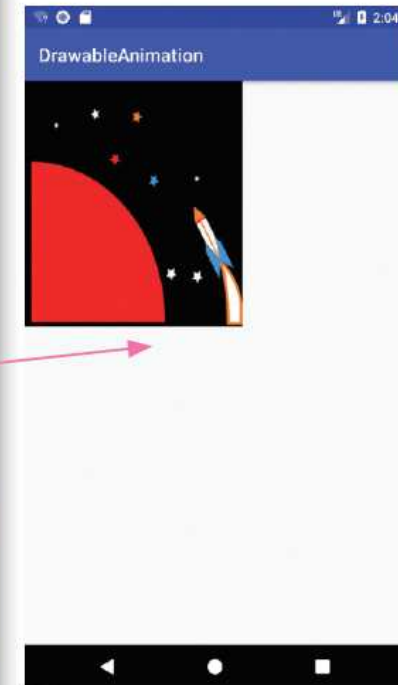
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
        rocketImage.setBackgroundResource(R.drawable.rocket);
        rocketAnimation = (AnimationDrawable) rocketImage.getBackground();
    }

    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            rocketAnimation.start();
            return true;
        }
        return super.onTouchEvent(event);
    }
}
```

애니메이션 리소스를 이미지  
뷰의 배경으로 설정한다.

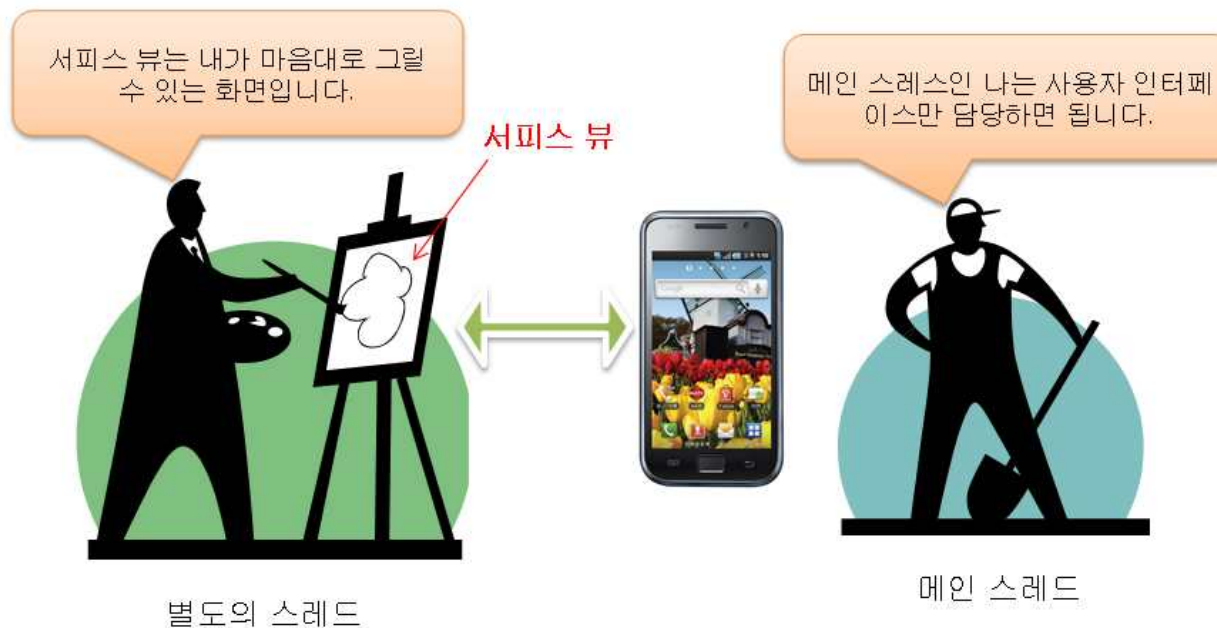
애니메이션 객체를 얻는다.

화면이 터치되면 애니  
메이션을 시작한다.



# 서피스 뷰

- 서피스뷰는 사용자 인터페이스와는 별도로 애플리케이션에게 그림을 그릴 수 있는 화면을 제공



# 서피스 뷰의 구조

- SurfaceView 클래스를 상속한 뷰를 생성한다.

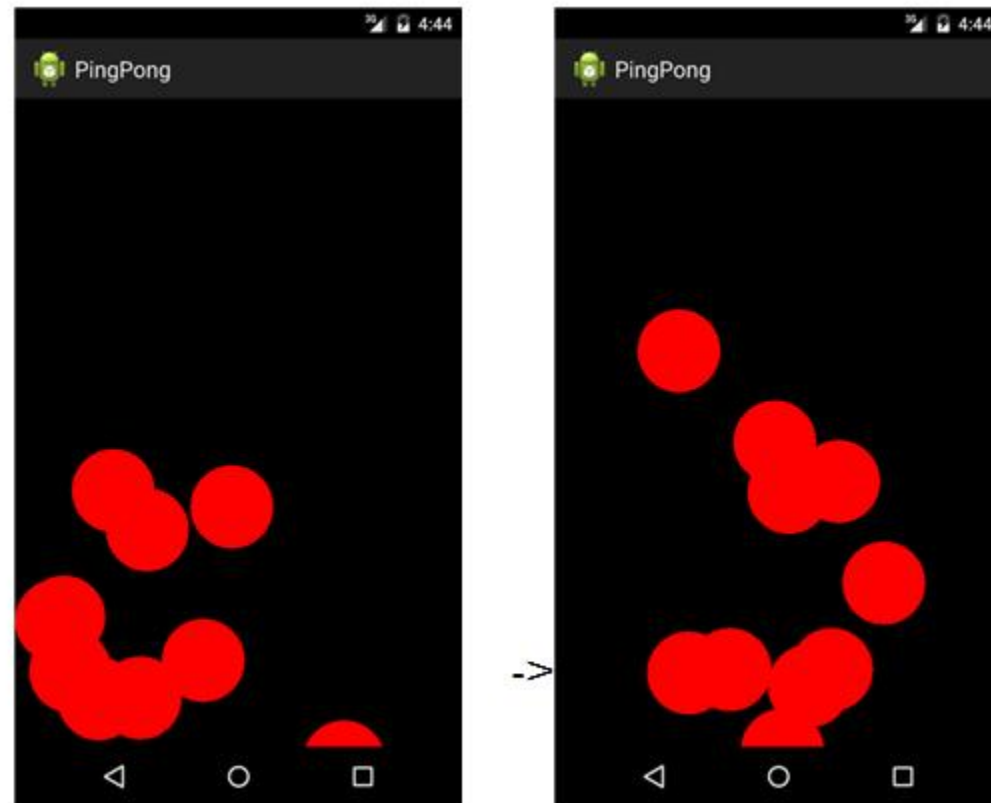
```
class MyView extends SurfaceView implements SurfaceHolder.Callback {  
    public void surfaceCreated(SurfaceHolder holder) {  
        // 서피스가 준비되었으므로 스레드를 시작한다.  
        ...  
    }  
    public void surfaceDestroyed(SurfaceHolder holder) {  
        // 서피스가 소멸되었으므로 스레드를 종료한다.  
        ...  
    }  
    public void surfaceChanged(SurfaceHolder holder, int format,  
                               int width, int height) {  
        // 서피스가 변경  
        ...  
    }  
}
```



# 스레드를 정의한다.

```
class MyThread extends Thread {  
    SurfaceHolder holder;  
    ...  
    public void run()  
    {  
        canvas = holder.lockCanvas();  
        // 캔버스에 그림을 그린다.  
        ...  
        holder.unlockCanvasAndPost(canvas);  
    }  
}
```

# 서피스 뷰 예제



# 핵심적인 코드

// 서피스 뷰 정의

**public class** MySurfaceView **extends** SurfaceView **implements**

SurfaceHolder.Callback {

...

**public** MySurfaceView(Context context) { // 생성자

**super**(context);

SurfaceHolder holder = getHolder(); // 서피스 뷰의 홀더를 얻는다.

holder.addCallback(**this**); // 콜백 메소드를 처리한다.

thread = **new** MyThread(holder); // 스레드를 생성한다.

// Ball 객체를 생성하여서 배열에 넣는다.

**for** (int i = 0; i < 10; i++)

    basket[i] = **new** Ball(20);

}

**public** MyThread getThread() {

**return** thread;

}

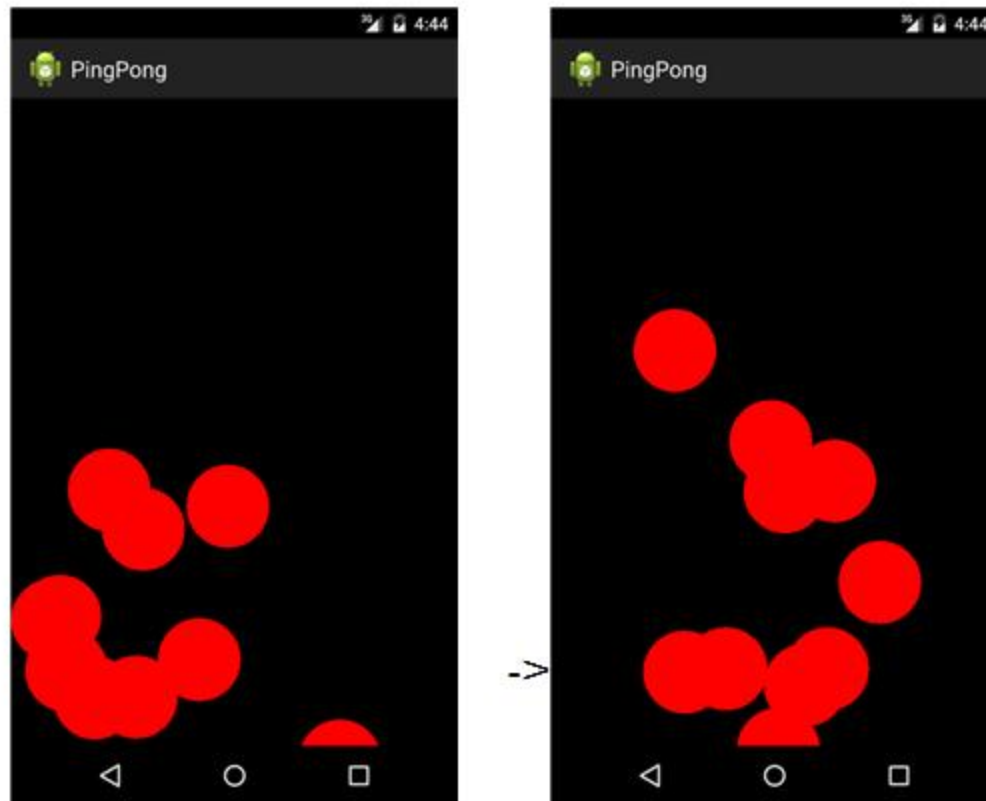
# 핵심적인 코드

```
public void surfaceCreated(SurfaceHolder holder) {  
    // 스레드를 시작한다.  
    thread.setRunning(true);  
    thread.start();  
}  
  
public void surfaceDestroyed(SurfaceHolder holder) {  
    boolean retry = true;  
    // 스레드를 중지시킨다.  
    thread.setRunning(false);  
    while (retry) {  
        try { thread.join(); // 메인 스레드와 합친다.  
            retry = false;  
        } catch (InterruptedException e) { }  
    }  
}  
  
public void surfaceChanged(SurfaceHolder holder, int format,  
    int width,    int height) {  
}
```

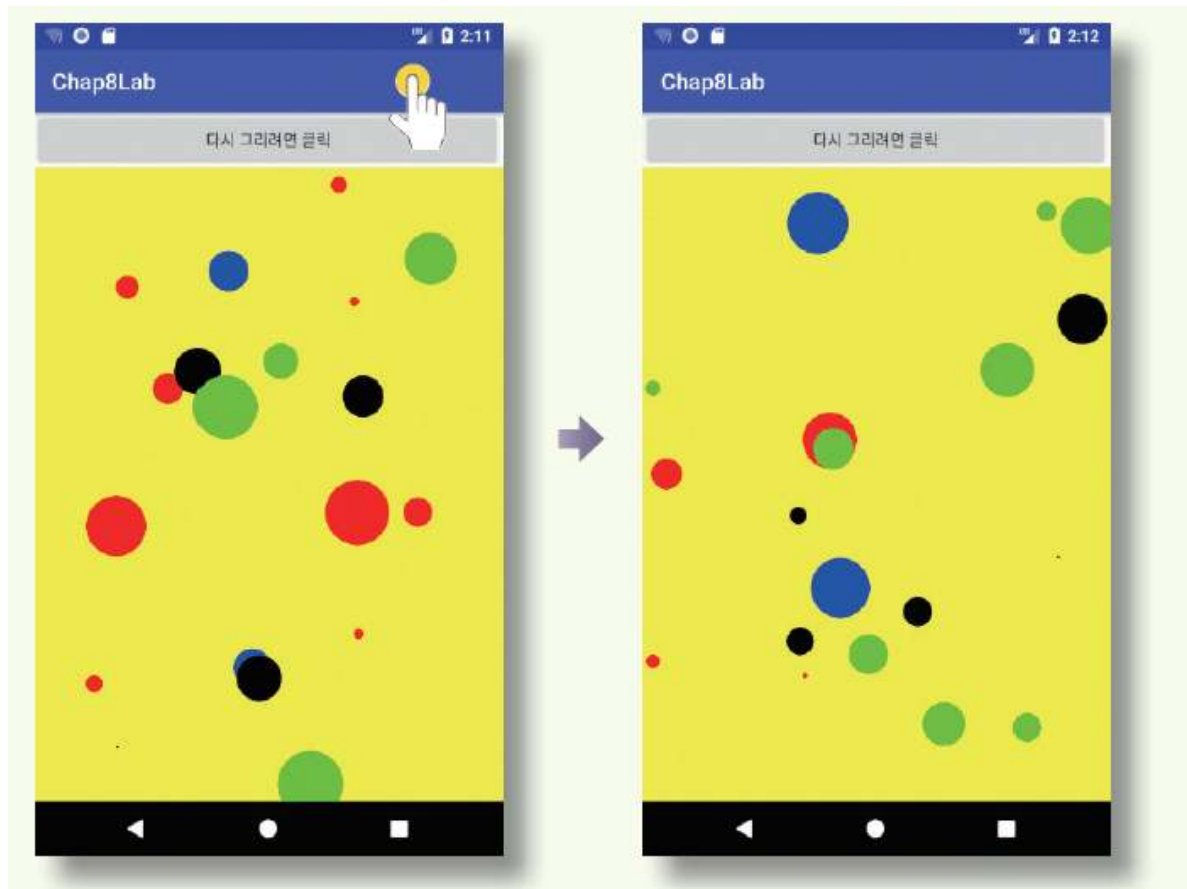
# 핵심적인 코드

```
public class MyThread extends Thread {  
    ...  
    @Override  
    public void run() {  
        while (mRun) {  
            Canvas c = null;  
            try {  
                c = mSurfaceHolder.lockCanvas(null);  
                c.drawColor(Color.BLACK);           // 캔버스의 배경을 지운다.  
                synchronized (mSurfaceHolder) {  
                    for (Ball b : basket) { // basket의 모든 원소를 그린다.  
                        b.paint(c);  
                    }  
                }  
            } finally {  
                if (c != null) { mSurfaceHolder.unlockCanvasAndPost(c); }  
            }  
        }  
    }  
}
```

# 서피스 뷰 예제



# Lab: 랜덤 그래픽 작성



```

public class MyView extends View {
    private Paint[] mForegrounds = { makePaint(Color.BLACK),
                                     makePaint(Color.BLUE), makePaint(Color.GREEN), makePaint(Color.RED)
    };
    private static Random r = new Random();

    public MyView(Context context) {
        super(context);
    }

    public MyView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawColor(Color.YELLOW);
        int width = getWidth();
        int height = getHeight();
        for (int i = 0; i < 20; i++) {
            float x = r.nextInt(width);
            float y = r.nextInt(height);
            float radius = r.nextInt(80);
            Paint circleColor = mForegrounds[r.nextInt(mForegrounds.length)];
            canvas.drawCircle(x, y, radius, circleColor);
        }
    }

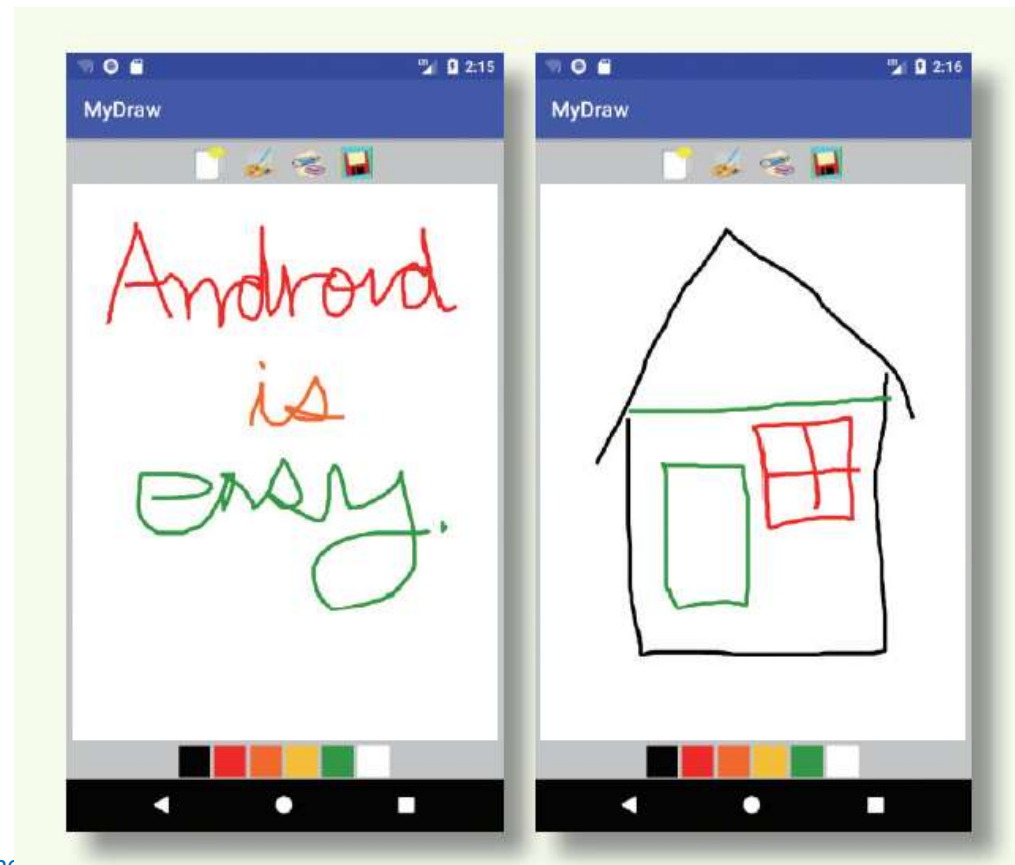
    private Paint makePaint(int color) {
        Paint p = new Paint();
        p.setColor(color);
        return (p);
    }
}

```



# Lab: 그림판 앱 작성

- 윈도우에서 기존 제공되는 그림판과 유사한 애플리케이션을 작성해보자



# 해싱 코드

```
@Override
protected void onDraw(Canvas canvas) {
    canvas.drawBitmap(canvasBitmap, 0, 0, canvasPaint);
    canvas.drawPath(path, paint);
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    // detect user touch
    float touchX = event.getX();
    float touchY = event.getY();
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            path.moveTo(touchX, touchY);
            break;
        case MotionEvent.ACTION_MOVE:
            path.lineTo(touchX, touchY);
            break;
        case MotionEvent.ACTION_UP:
            drawCanvas.drawPath(path, paint);
            path.reset();
            break;
        default:
            return false;
    }
    invalidate();
    return true;
}
```