커스텀 뷰 만들기

기초 원리 학습

OneKick

- 액티비티(Activity)
  - 안드로이드 프로그램을 구성하는 주요 구성요소
  - 화면에 보이지 않고, 실제 사용자가 보는 것은 뷰(View)이다.

- 뷰 여러 개가 모여서 액티비티를 구성
  - 액티비티 여러 개가 모여 안드로이드 응용 프로그램 구성

- 커스텀 뷰
  - 자기만의 특수한 뷰를 만들 수 있음
  - 여기서는 전체 화면을 차지하고, 내부에 사진을 그리는 뷰 구현 예정

기초 원리 학습

OneKick

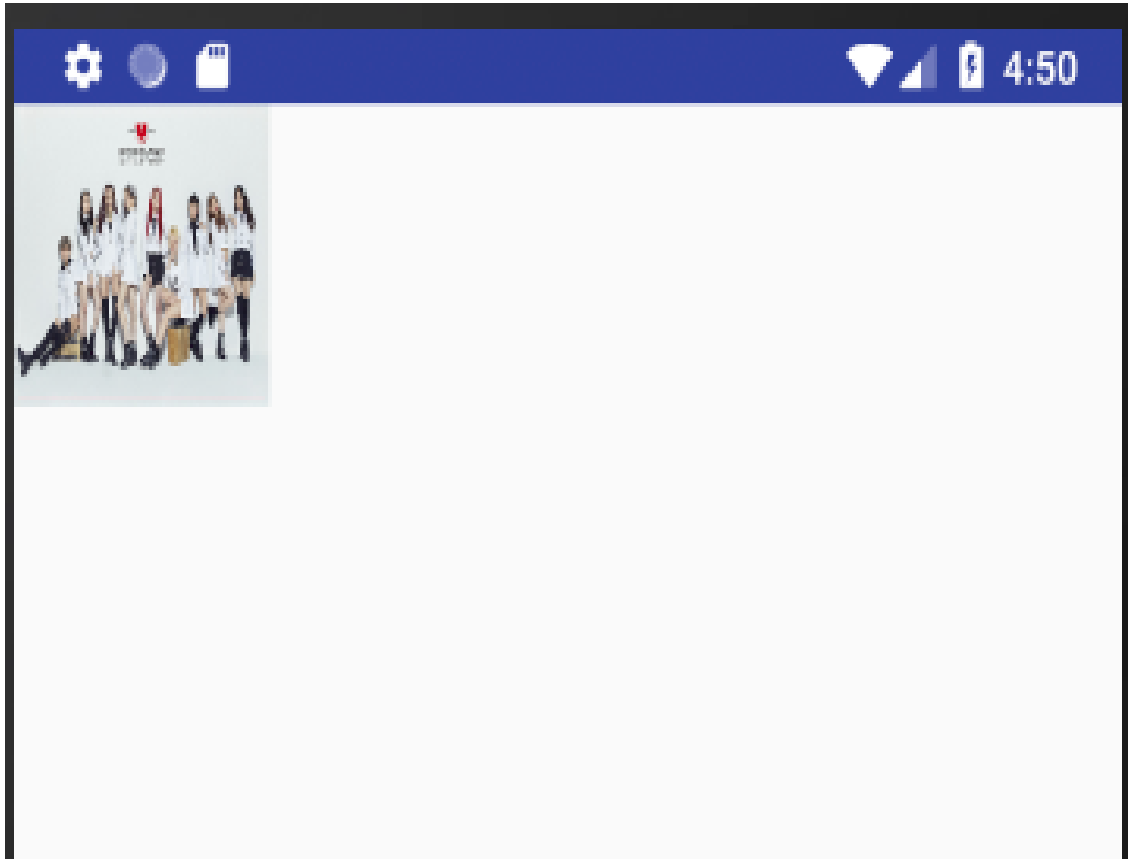
- 뷰(View)
  1. 위젯(Widget)
     - 버튼, 텍스트 뷰, 에디트 등
     - 흔히 컨트롤(Control)이라 함

  2. 뷰그룹(ViewGroup)
     - 직접적으로 보이지 않으며 뷰(View)를 담는 컨테이너 역할
     - 레이아웃들

프로그램 구현

OneKick

- 화면 전체를 차지하는 커스텀뷰 만들기
  - 처음에 사진을 화면 중앙에 표시
  - 터치하면 터치한 곳으로 그림이 이동
  - 핸들러를 이용하여 그림이 아래로 움직이는 애니메이션 예제 구현



새로운 View 클래스 생성

OneKick

- 현재 패키지 위에서 새 클래스 추가
  - 클래스명: CustomView
  - 부모클래스: android.view.View

- 메소드 오버라이드
  - 인자가 2개인 생성자
  - onDraw()
    - 화면을 갱신해야 할 때 불림
    - Canvas를 이용해 그림을 그릴 수 있음
  - onSizeChanged()
    - 화면이 만들어 질 때 가장 먼저 불림
    - 여기서 화면의 가로세로 크기를 알아낼 수 있다.
  - onTouchEvent()
    - 터치한 위치를 저장하고, 화면을 갱신 시킨다.

커스텀 뷰의 터치 기능

OneKick

- 그림을 터치 했을 때 터치한 위치로 그림이 이동하도록 구현
  - 터치를 처리인식하기 위해 onTouchEvent() 메소드 오버라이드
  - X,Y 좌표는 실수 형태로 반환
    - int 형으로 사용하려면 형변환
  - 안드로이드 플랫폼에서 좌표계
    - 왼쪽 위를 0,0으로 잡고 사용
- 멤버변수에 터치한 x,y 좌표를 저장할 point 객체 생성
  - onDraw()에서 화면을 그릴 때 x,y 를 기준으로 그림 그리기
  - 시스템에 화면을 다시 갱신하라는 메시지를 보내기 위해 invalidate() 호출

터치 이후 화면 갱신하기

OneKick

- CustomView의 point 필드
  - 사용자가 화면을 터치 할 때 마다 그 좌표를 멤버 변수로 저장
    - canvas가 있어야 그림을 그릴 수 있으므로 onDraw()를 호출해야함
  - invalidate() 메소드 호출
    - 시스템에 의해 onDraw()가 불려서 화면을 다시 그리게 됨
    - 직접 onDraw()를 부를 순 없음
- onDraw()
  - 필드로 보관했던 x,y 좌표를 이용해서 화면에 그리는 메소드
  - 마치 터치하는 곳으로 그림이 옮겨 가는 듯한 효과를 구현할 수 있다.

CustomView.java 파일의 소스코드-1/4

OneKick

```
package com.iot.customview;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Point;
import android.graphics.Rect;
import android.graphics.drawable.Drawable;
import android.os.Handler;
import android.os.Message;
import android.support.annotation.Nullable;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;

public class CustomView extends View {
    private static final String TAG             = "CustomView";
    private static final int IMAGE_SIZE         = 250;    // 그릴 그림 크기
    private static final int WHAT_UPDATE        = 1;      // 핸들러 반복 메시지
    private static final long DELAY_MS          = 33;     // 30프레임(33mSec)
    private static final int DELTA              = 20;     // 그림 이동 픽셀
```

CustomView.java 파일의 소스코드-2/4

OneKick

```
    private int direction      = 1;                    // -1(up) or 1(down) 이동 방향
    private final Drawable drawable;                   // 그릴 그림
    private Rect rect          = new Rect();           // 그릴 영역
    private Point point        = new Point();          // 터치 좌표
    private Point size         = new Point();          // 화면의 크기

    private Handler handler  = new Handler(){
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if(direction==1)                                    // down
            if(point.y+ DELTA +IMAGE_SIZE<=size.y)          // can move
                point.y+= DELTA;
            else direction*=-1;
        if(direction==-1)                                   // up
            if(point.y + DELTA>=0)                          // can move
                point.y-= DELTA;
            else direction*=-1;
        invalidate();                       // 화면 갱신
        handler.sendEmptyMessageDelayed(WHAT_UPDATE, DELAY_MS); // 반복
    }
};
```

CustomView.java 파일의 소스코드-3/4

OneKick

```
public CustomView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        Log.i(TAG, "CustomView() called");
        drawable = getResources().getDrawable(R.drawable.momoland);
}

@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(w, h, oldw, oldh);

        size.x      = getWidth();                    // 그림을 화면 중앙에 표시
        size.y      = getHeight();
        point.x     = size.x/2 - IMAGE_SIZE/2;
        point.y     = size.y/2 - IMAGE_SIZE/2;

        Log.i(TAG, "size="+size);
        Log.i(TAG, "pointer="+point);

        handler.sendEmptyMessageDelayed(WHAT_UPDATE, DELAY_MS); // 시작
}
```

CustomView.java 파일의 소스코드-4/4

OneKick

```
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        rect.left       = point.x;                  // 그릴 영역
        rect.top        = point.y;
        rect.right      = point.x+IMAGE_SIZE;
        rect.bottom     = point.y+IMAGE_SIZE;
        drawable.setBounds(rect);                   // 그릴 영역 지정
        drawable.draw(canvas);                      // 도화지에 그리기
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        point.x = (int)event.getX() - IMAGE_SIZE/2;     // 터치 좌표 보관
        point.y = (int)event.getY() - IMAGE_SIZE/2;
        Log.i(TAG,"onTouchEvent "+point);
        invalidate();                               // onDraw() call

        return super.onTouchEvent(event);
    }
}
```

뷰를 액티비티에 표시하기

OneKick

- 구현을 완료한 뷰를 액티비티에 표시하기
  - XML에 포함 시킨다.
    - Activity가 XML을 부르고, XML이 View를 부름
  - 표준 View가 아니므로, 패키지 경로까지 포함시켜 주어야 함
  - XML에서 생성되는 View
    - 파라미터가 2개인 생성자를 반드시 오버라이드 해야 함
    - 없으면 죽음

activity_main.xml 파일 구현

OneKick

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="안드로이드 APK
파일의 스키마 경로는 생략합니다."

    xmlns:app="안드로이드 APK 파일의 스키마 경로는 생략"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.iot.customview.CustomView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"  />

</android.support.constraint.ConstraintLayout>
```

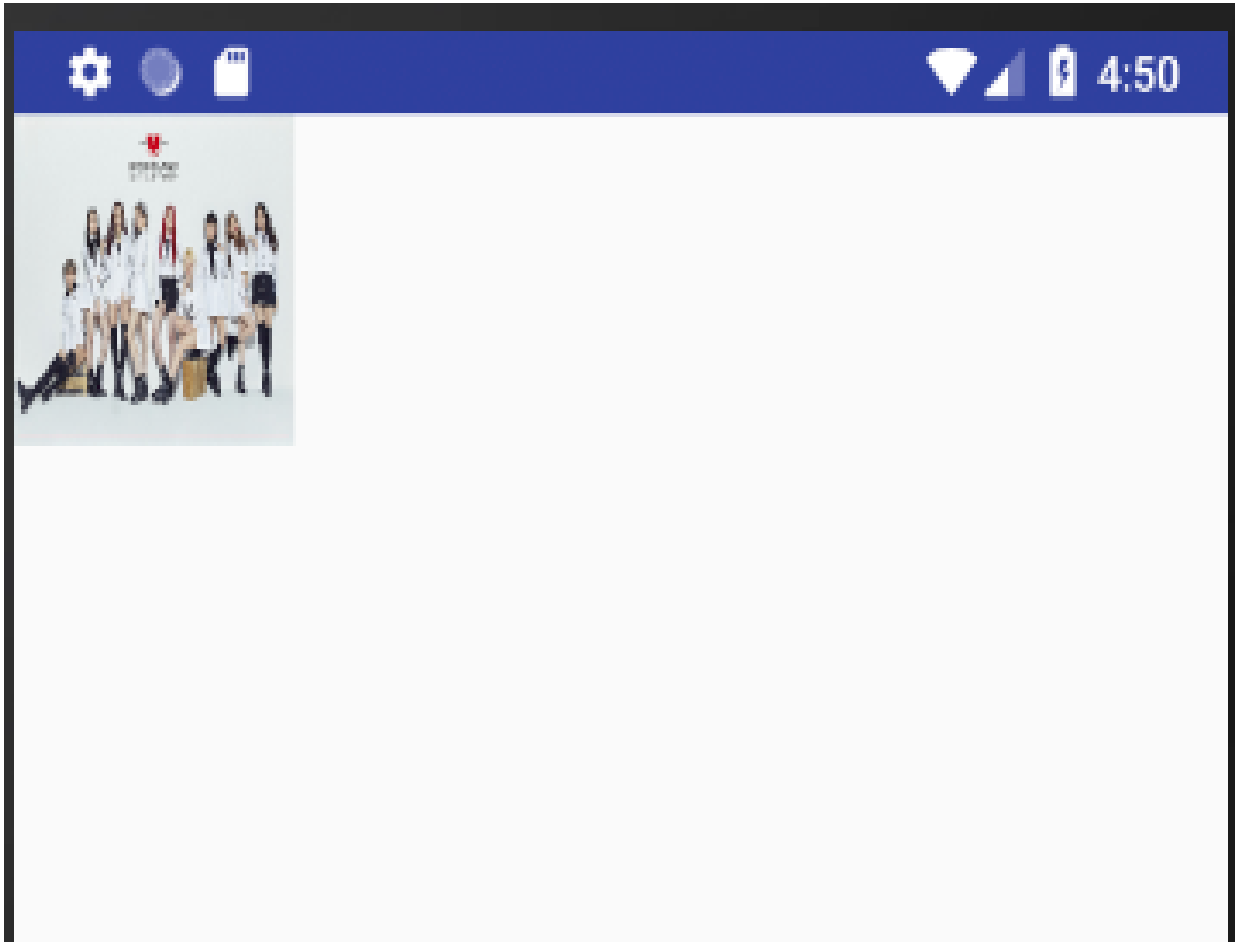
MainActivity.java 파일의 소스코드

- 핸들러를 이용하여 그림이 아래로 움직이는 예제 구현

```
package com.iot.customview;

import android.app.Activity;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Window;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);  // 타이틀바 제거
        setContentView(R.layout.activity_main);
    }
}
```

동작 테스트

- 자동으로 그림이 아래위로 움직인다.
  - 터치하면 그 위치로 그림이 이동한다.



심화학습

OneKick

- 아래위로 움직이는 그림을 대각선으로 돌아다니도록 구현해보자.
  - 지금은 방향이 UP, DOWN 밖에 없음
  - X축 방향과 Y축 방향을 따로 관리하면 가능
  - 수평 충돌 시 X에 -1곱하기
  - 수직 출돌 시 Y에 -1곱하기