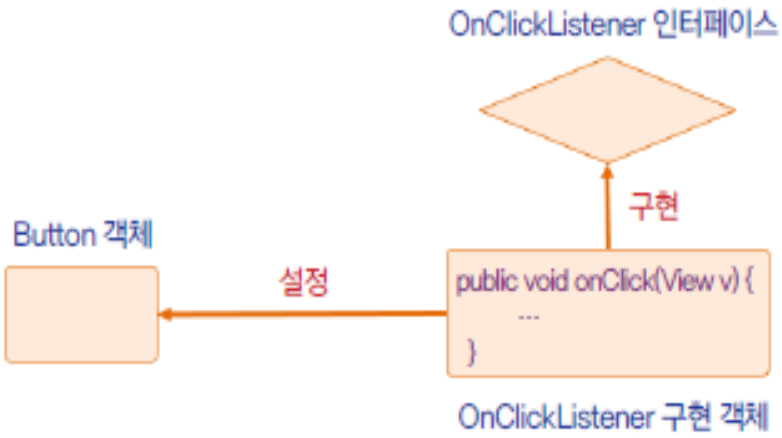
5. 이벤트 처리

1. 이벤트 처리 방법
2. 이벤트 처리 메소드 재정의
3. 키 이벤트
4. 터치 이벤트
5. 멀티터치 이벤트
6. 기타

1. 이벤트 처리 방법

뷰의 이벤트 처리하기



[버튼에 OnClickListener를 설정할 때의 패턴]

뷰를 상속할 때 이벤트 처리를 위한 메소드 재정의

boolean onTouchEvent (MotionEvent event)

boolean onKeyDown (int keyCode, KeyEvent event)

boolean onKeyUp (int keyCode, KeyEvent event)

뷰 객체에 전달되는 이벤트를 처리하기 위한 리스너 설정

View.OnTouchListener : boolean onTouch (View v, MotionEvent event)

View.OnKeyListener : boolean onKey (View v, int keyCode, KeyEvent event)

View.OnClickListener : void onClick (View v)

View.OnFocusChangeListener : void onFocusChange (View v, boolean hasFocus)

1. 이벤트 처리 방법

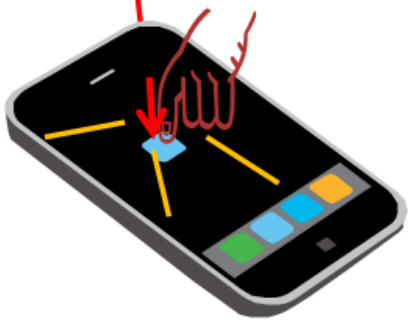
- **이벤트 등록 방법**
  - XML리소스에서 이벤트 함수 등록하여 이벤트처리
  - 이벤트 처리 객체를 생성하여 컴포넌트에 등록
    - 일반적인 방법
  - 뷰 클래스의 이벤트 처리 메소드를 재정의
    - 커스텀 뷰를 작성하는 경우: (예) 게임

1. 이벤트 처리 방법

- XML리소스에서 이벤트 메소드 등록하여 이벤트처

```xml
<?xml version="1.0" encoding="utf-8"?>
<Button
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="sendMessage"
    android:text="@string/button_send" />
```

사용자가 클릭하면 호출된다.



XML 파일에 적었던 메소드를 구현해주면 된다.

```java
public class MyActivity extends Activity {
    @Override
    public void onCreate(…) {
    //...
    }
    public void sendMessage(View view)
    {

    }
}
```

1. 이벤트 처리 방법

- 이벤트 처리 객체 사용


```
Class MyClass
{
  class Listener implements OnClickListener {
    public void onClick(View v) {
      ...
    }
  }
  ...
  Listener lis = new Listener();
  button.setOnClickListener(lis);
  ...
}
```

버튼에 붙은 리스너 객체가 이벤트를 처리한다.

1. 이벤트 처리 방법

이벤트 리스너 클래스


전체적인 구조

```
class MyClass
{
    class Listener implements OnClickListener {
        public void onClick(View v){
            ...
        }
    }
    ...
    Listener lis = new Listener();
    button.setOnClickListener(lis);
    ...
}
```

인터페이스를 구현한 클래스 정의

이벤트 리스너 객체 생성

버튼에 이벤트 리스너 객체를 등록

1. 이벤트 처리 방법

리스너의 종류

리스너	콜백 메소드	설명
View.OnClickListener	onClick()	사용자가 어떤 항목을 터치하거나 내비게이션 키나 트랙볼로 항목으로 이동한 후에 엔터키를 눌러서 선택하면 호출된다.
View.OnLongClickListener	onLongClick()	사용자가 항목을 터치하여서 일정 시간 동안 그대로 누르고 있으면 발생한다.
View.OnFocusChangeListener	onFocusChange()	사용자가 하나의 항목에서 다른 항목으로 포커스를 이동할 때 호출된다.
View.OnKeyListener	onKey()	포커스를 가지고 있는 항목 위에서 키를 눌렀다가 놓았을 때 호출된다.
View.OnTouchListener	onTouch()	사용자가 터치 이벤트로 간주되는 동작을 한 경우에 호출된다.
View.OnCreateContextMenuListener	onCreateContextMenu()	컨텍스트 메뉴가 구축되어 있는 경우에 호출된다.

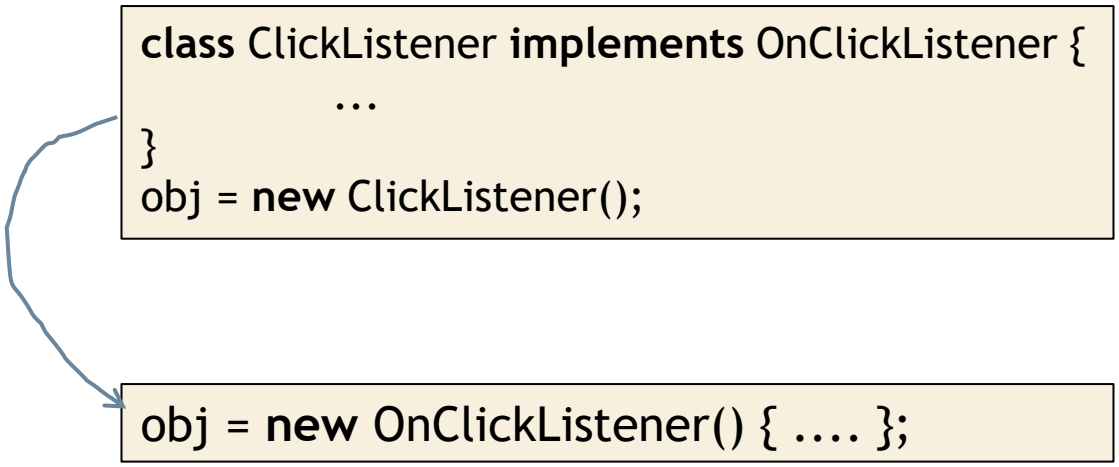
1. 이벤트 처리 방법

- **리스너 객체를 생성하는 방법**
  - 리스너 클래스를 내부 클래스로 정의.
  - 리스너 클래스를 무명 클래스로 정의.
  - 리스너 인터페이스를 액티비티 클래스에 구현.

가장 많이 사용되는 방법!

1. 이벤트 처리 방법

- **무명 클래스**
  - 클래스 몸체는 정의되지만 이름이 없는 클래스이다.
  - 무명 클래스는 클래스를 정의하면서 동시에 객체를 생성하게 됨



1. 이벤트 처리 방법-리스너를 무명의 클래스로 사용

```
...
public class ButtonEvent2Activity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "버튼
                    눌려졌습니다", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

무명 클래스 사용

1. 이벤트 처리 방법- 리스너를 내부 클래스로 정의

```
...
public class ButtonEvent2Activity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(this);
    }
    class MyOnClickListener extend OnClickListener{
        public void onClick(View v) {
            Toast.makeText(getApplicationContext(), "버튼
                눌려졌습니다", Toast.LENGTH_SHORT).show();
        }
    }
}
```

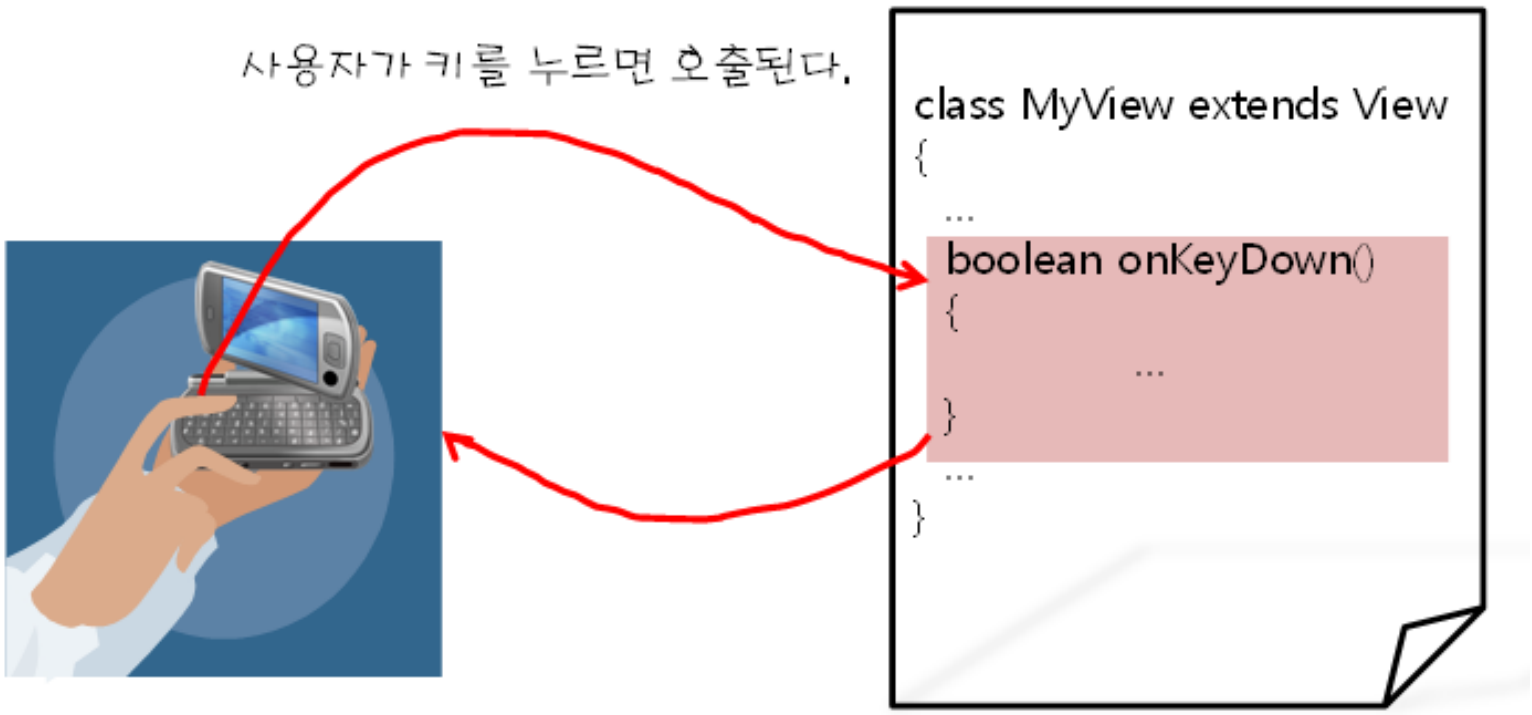
내부 클래스 정의 사용

1. 이벤트 처리 방법-액티비티 클래스를 리스너 인터페이스 구현

```
...
public class ButtonEvent2Activity extends Activity implements OnClickListener{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(this);
    }
    public void onClick(View v) {
            Toast.makeText(getApplicationContext(), "버튼
                    눌려졌습니다", Toast.LENGTH_SHORT).show();
    }
}
```

2. 이벤트 처리 메소드 재정의

- **커스텀 컴포넌트**
  - 개발자가 직접 View 클래스를 상속받아서 필요한 위젯을 개발
  - 커스텀 컴포넌트에 이벤트 처리 메소드를 재정의

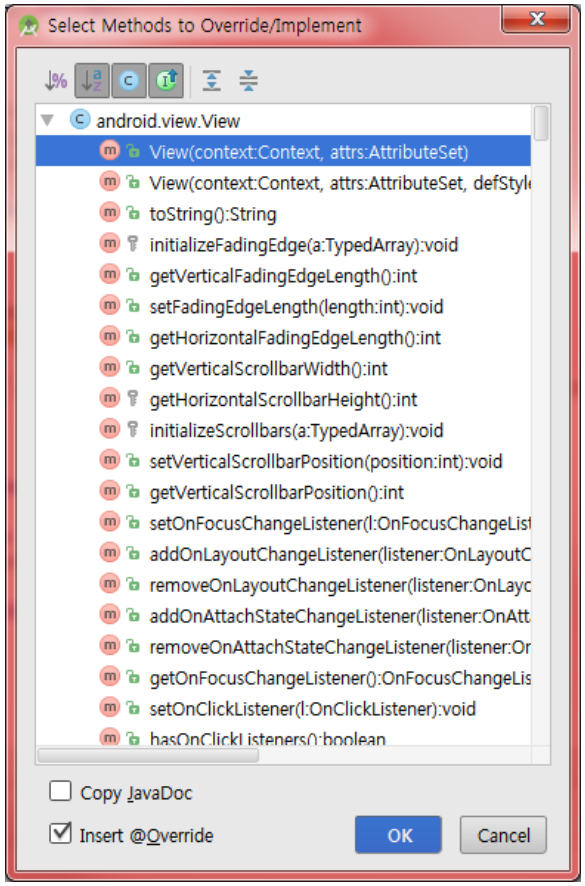


2. 이벤트 처리 메소드 재정의

- **View에 재정의할 수 있는 콜백 메소드**
  - onKeyDown(int, KeyEvent)
  - onKeyUp(int, KeyEvent)
  - onTrackballEvent(MotionEvent)
  - onTouchEvent(MotionEvent)
  - onFocusChanged(boolean, int, Rect)
  - onDraw(Canvas canvas)

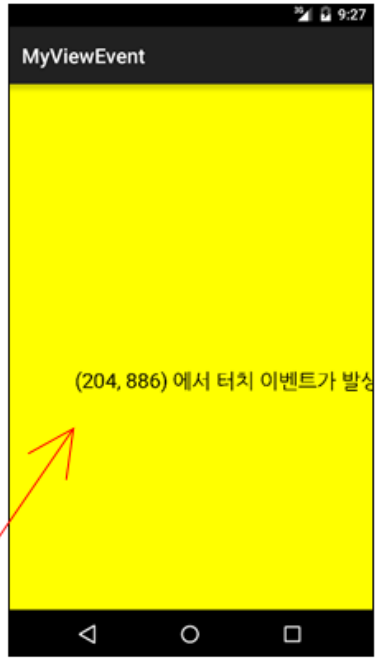
2. 이벤트 처리 메소드 재정의

- **콜백 메소드를 재정의할 때 편리한 기능**
  - [Code]->[Override Methods...] 사용!



예제

```
class MyView extends View {
    int key;
    String str;
    int x, y;

    public MyView(Context context) {

        super(context);
        setBackgroundColor(Color.YELLOW);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        x = (int) event.getX(0);
        y = (int) event.getY(0);
        invalidate();
        return super.onTouchEvent(event);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        paint.setTextSize(60);
        canvas.drawText("("+x+ ", "+y+") 에서 터치 이벤트가 발생하였음", x, y, paint);
    }

}
```

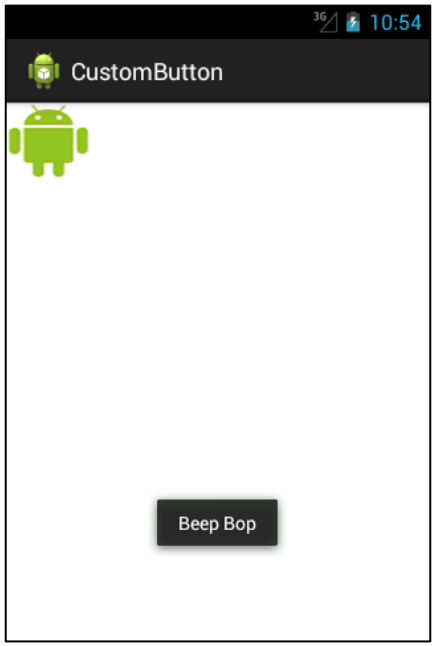


예제

```
public class MyViewEventActivity extends ActionBarActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        MyView w = new MyView(this);
        setContentView(w);
    }
}
```

2. 이벤트 처리 메소드 재정의

- **커스텀 버튼**
  - 버튼 위에 텍스트 대신에 이미지가 그려져 있는 버튼

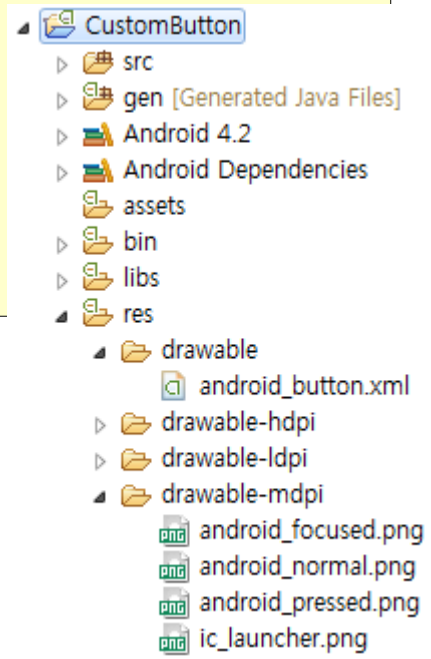


2. 이벤트 처리 메소드 재정의

- **커스텀 버튼 정의**
  - XML로 버튼에 사용되는 이미지를 등록한다.

  /res/drawable/android_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/android_pressed"
          android:state_pressed="true" />
    <item android:drawable="@drawable/android_focused"
          android:state_focused="true" />
    <item android:drawable="@drawable/android_normal" />
</selector>
```



2. 이벤트 처리 메소드 재정의

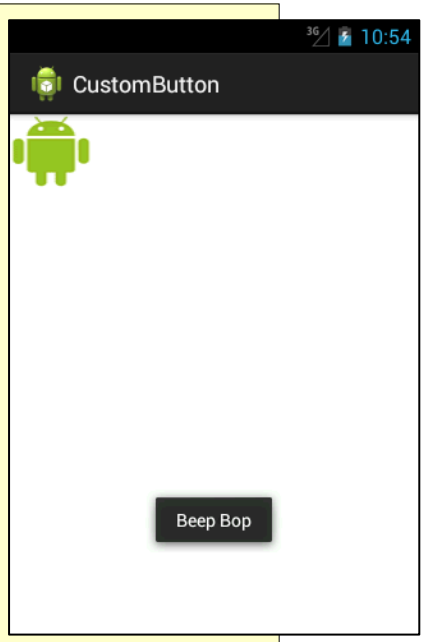
- 레이아웃 파일에 버튼을 정의한다.

/res/layout/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" >

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/android_button"
        android:padding="10dp" />

</LinearLayout>
```

2. 이벤트 처리 메소드 재정의

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
            Toast.makeText(getApplicationContext(), "Beep Bop",
                        Toast.LENGTH_SHORT).show();
            }
        });

    }
}
```



3. 키 입력 이벤트 처리하기

뷰를 상속할 때 키 이벤트 처리를 위한 메소드 재정의

```
boolean onKeyDown (int keyCode,
KeyEvent event)
boolean onKey (View v, int keyCode,
KeyEvent event)
```

위젯에 키 포커스설정

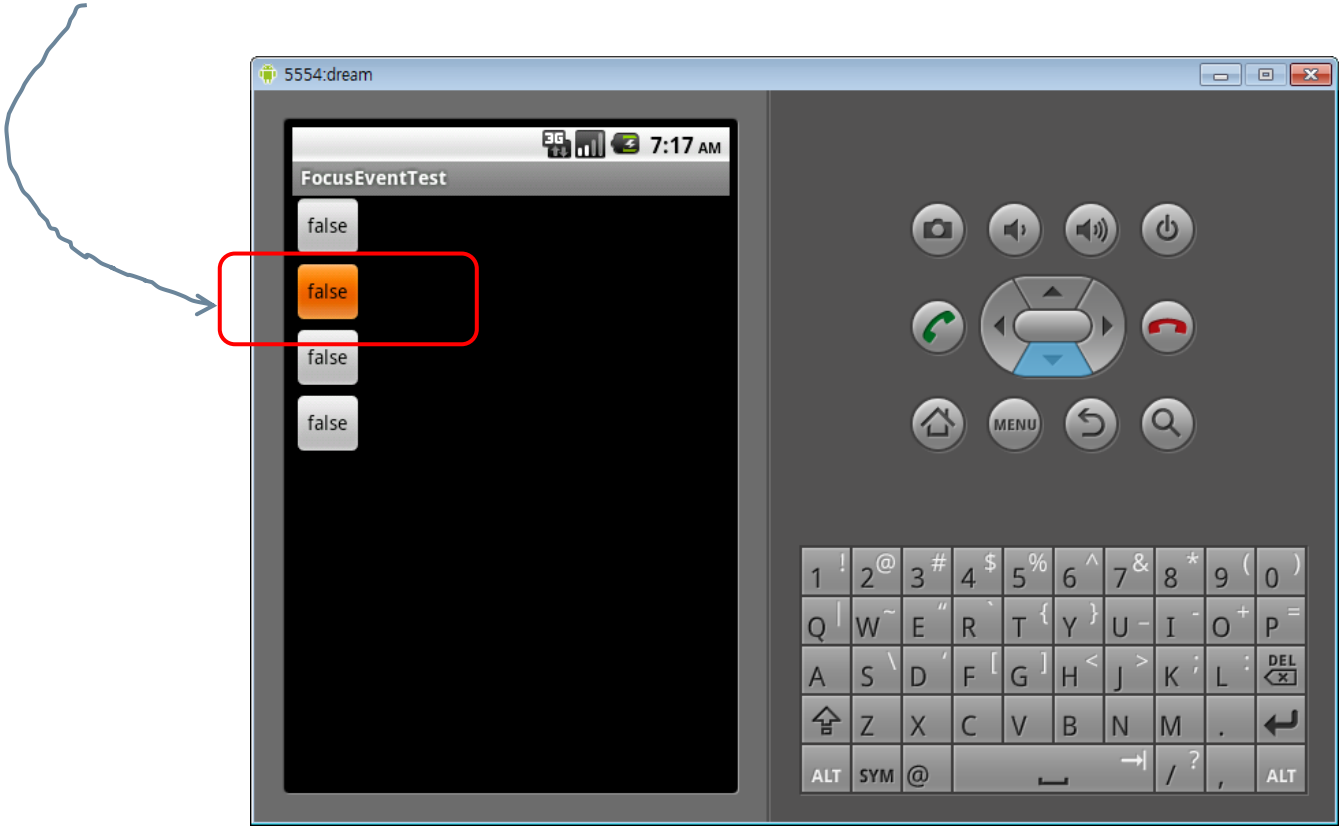
```
w.setFocusable(true);
w.setFocusableInTouchMode(true);
```

[키를 눌렀을 때 전달되는 대표적인 키값]

키 코드	설 명
KEYCODE_DPAD_LEFT	- 왼쪽 화살표
KEYCODE_DPAD_RIGHT	- 오른쪽 화살표
KEYCODE_DPAD_UP	- 위쪽 화살표
KEYCODE_DPAD_DOWN	- 아래쪽 화살표
KEYCODE_DPAD_CENTER	- [중앙] 버튼
KEYCODE_CALL	- [통화] 버튼
KEYCODE_ENDCALL	- [통화 종료] 버튼
KEYCODE_HOME	- [홈] 버튼
KEYCODE_BACK	- [뒤로 가기] 버튼
KEYCODE_VOLUME_UP	- [소리 크기 증가] 버튼
KEYCODE_VOLUME_DOWN	- [소리 크기 감소] 버튼
KEYCODE_0 ~ KEYCODE_9	- 숫자 0부터 9까지의 키값
KEYCODE_A ~ KEYCODE_Z	- 알파벳 A부터 Z까지의 키값

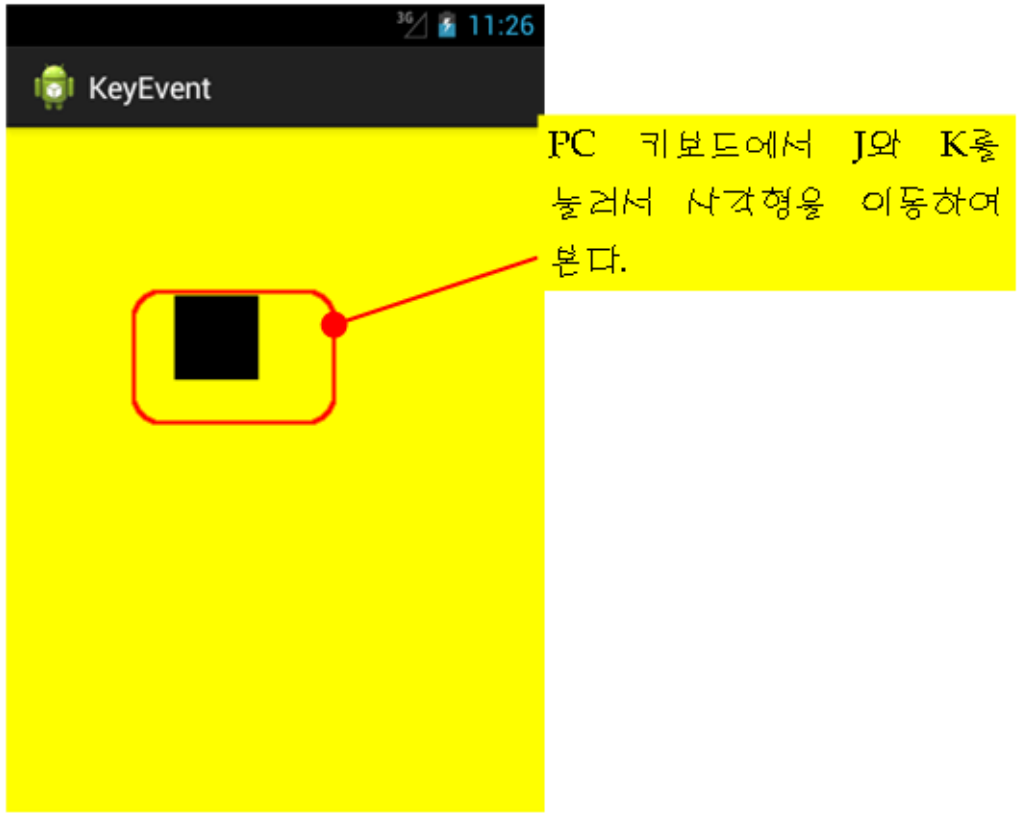
3. 키 이벤트

- **키 포커스**
  - 포커스를 가진 위젯 만이 키패드를 통하여 입력을 받을 수 있다.



3. 키 이벤트

- **키 이벤트 처리**
  - OnKeyListener 리스너를 구현
- **키를 누르면 사각형이 이동하는 예제를 작성.**



3. 키 이벤트

커스텀 뷰를 정의

```
...
public class MainActivity extends Activity {
        int x = 100, y = 100;
        protected class MyView extends View {
                public MyView(Context context) {
                        super(context);
                        setBackgroundColor(Color.YELLOW);
                }
                @Override
                protected void onDraw(Canvas canvas) {
                        Paint paint = new Paint();
                        canvas.drawRect(x, y, x+50, y+50, paint);
                }
        }
```

3. 키 이벤트- 커스텀 뷰를 정의

```
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        MyView w = new MyView(this);
        w.setFocusable(true);
        w.setFocusableInTouchMode(true);
        setContentView(w);

        w.setOnKeyListener(new OnKeyListener() {
            public boolean onKey(View v, int keyCode, KeyEvent event) {
                if (event.getAction() == KeyEvent.ACTION_UP) {
                    switch (keyCode) {
                    case KeyEvent.KEYCODE_J:
                        x -= 30;
                        break;
                    case KeyEvent.KEYCODE_K:
                        x += 30;
                        break;
                    }
                    v.invalidate();
                    return true;
                }
                return false;
            }
        });
    }}
```

4. 터치 이벤트

- 일반적으로 커스텀 뷰를 정의하고 onTouchEvent() 재정의 드래그 등 클릭이랑 다름.

```
class MyView extends View {
        ...
        @Override
        public boolean onTouchEvent(MotionEvent event) {
            x = (int) event.getX();
            y = (int) event.getY();
            ...
        }
}
```

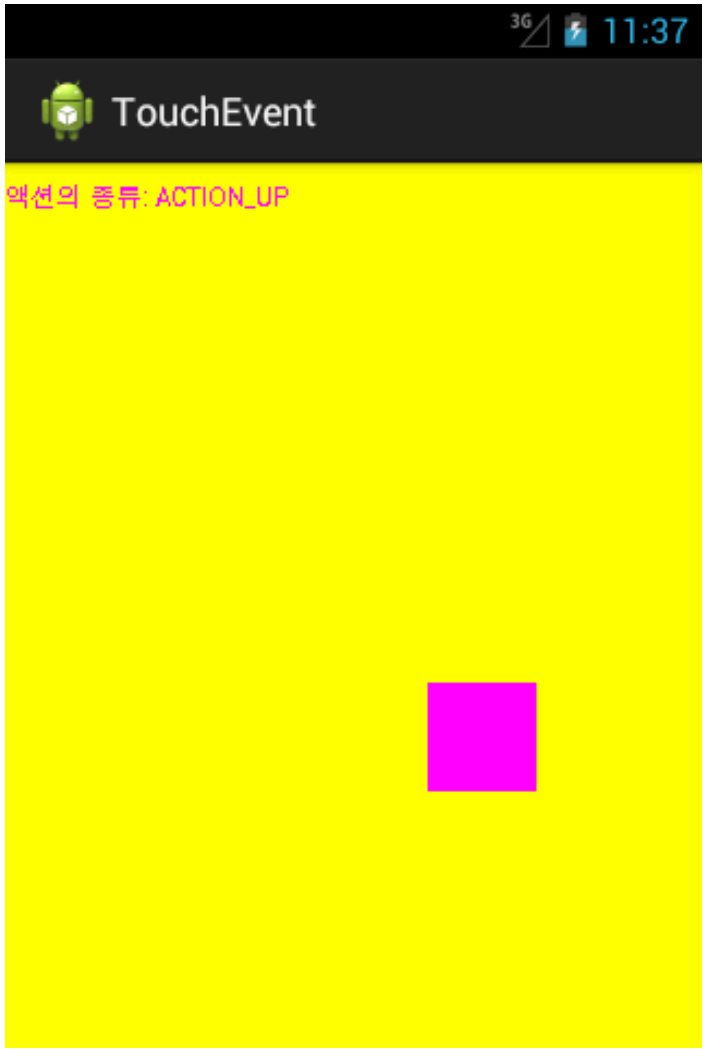
4. 터치 이벤트

터치 이벤트의 종류

액션	설명
ACTION_DOWN	누르는 동작이 시작됨
ACTION_UP	누르고 있다가 뗄때 발생함
ACTION_MOVE	누르는 도중에 움직임
ACTION_CANCEL	터치 동작이 취소됨
ACTION_OUTSIDE	터치가 현재의 위젯을 벗어남

4. 터치 이벤트

- 터치로 사각형 움직이기



4. 터치 이벤트-예제

```
public class TouchEventActivity extends Activity {
        protected class MyView extends View {
                int x = 100, y = 100;
                String str;
                public MyView(Context context) {
                        super(context);
                        setBackgroundColor(Color.YELLOW);
                }
                @Override
                protected void onDraw(Canvas canvas) {
                        Paint paint = new Paint();
                        paint.setColor(Color.MAGENTA);
                        canvas.drawRect(x, y, x + 50, y + 50, paint);
                        canvas.drawText("액션의 종류: " + str, 0, 20, paint);
                }
```

4. 터치 이벤트-예제

```
        @Override
        public boolean onTouchEvent(MotionEvent event) {
            x = (int) event.getX();
            y = (int) event.getY();
            if (event.getAction() == MotionEvent.ACTION_DOWN)
                str = "ACTION_DOWN";
            if (event.getAction() == MotionEvent.ACTION_MOVE)
                str = "ACTION_MOVE";
            if (event.getAction() == MotionEvent.ACTION_UP)
                str = "ACTION_UP";
            invalidate();
            return true;
        }
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        MyView w = new MyView(this);
        setContentView(w);
    }
}
```

4. 터치 이벤트

- 터치로 곡선 그리기

4. 터치 이벤트

터치로 곡선 그리기 #1

SingleTouchActivity.java

```
package kr.co.company.singletouch;
// 소스만 입력하고 Ctrl-Shift-O를 눌러서 import 문장을 자동으로 생성한다.

public class SingleTouchActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new SingleTouchView(this, null));
    }

}
```

4. 터치 이벤트

터치로 곡선 그리기 #2

SingleTouchView.java

```
package kr.co.company.singletouch;
// 소스만 입력하고 Ctrl-Shift-O를 눌러서 import 문장을 자동으로 생성한다.

public class SingleTouchView extends View {
    private Paint paint = new Paint();
    private Path path = new Path();

    public SingleTouchView(Context context, AttributeSet attrs) {
        super(context, attrs);

        paint.setAntiAlias(true);
        paint.setStrokeWidth(10f);
        paint.setColor(Color.BLUE);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeJoin(Paint.Join.ROUND);
    }

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

선분을 매끄럽게 그리기 위하여 앤티 에일리어싱을 설정한다.

선분의 두께를 10으로 한다.

현재까지의 경로를 모두 그린다.

4. 터치 이벤트

터치로 곡선 그리기 #3

```
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        float eventX = event.getX();
        float eventY = event.getY();

        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            path.moveTo(eventX, eventY);
            return true;
        case MotionEvent.ACTION_MOVE:
            path.lineTo(eventX, eventY);
            break;
        case MotionEvent.ACTION_UP:
            break;
        default:
            return false;
        }

        invalidate();
        return true;
    }
}
```

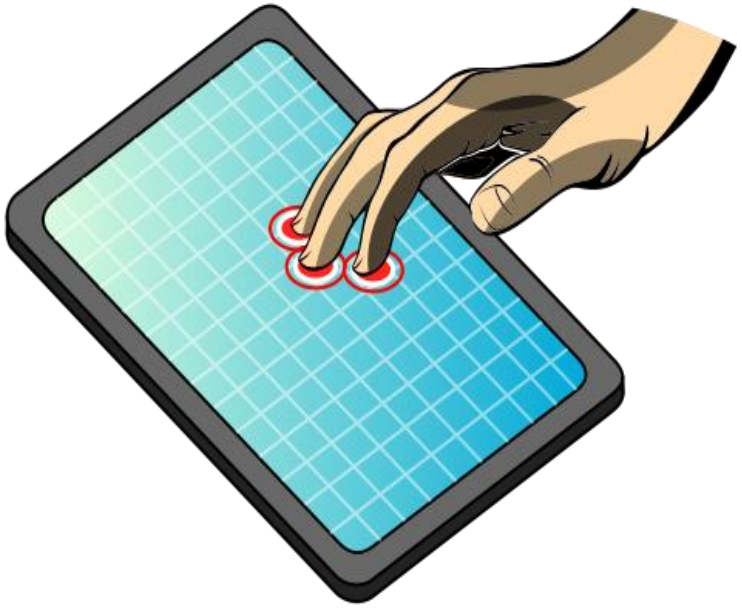
마우스가 터치된 위치를 얻는다.

터치가 눌려지면 경로에 위치를 저장한다.

터치가 떼어지면 경로에 직선그리기를 저장한다.

5. 멀티 터치

- 여러 개의 손가락을 이용하여 화면을 터치하는 것으로 이미지를 확대/축소할 때 많이 사용된다.



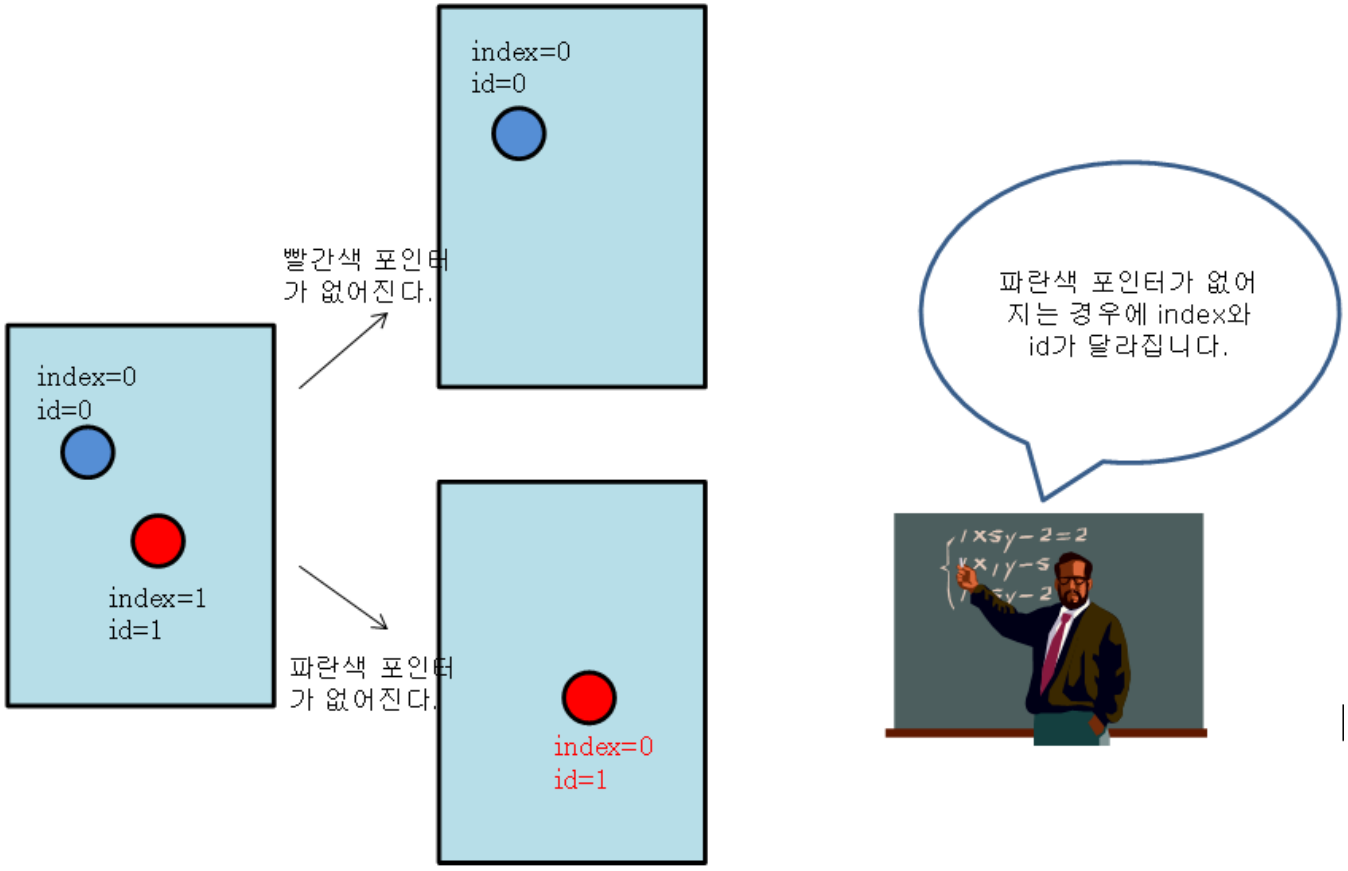
5. 멀티 터치

□ 터치 이벤트의 종류

- ACTION_DOWN – 화면을 터치하는 첫 번째 포인터에 대하여 발생한다. 제스처 인식이 시작된다. 첫 번째 터치는 항상 MotionEvent에서 인덱스 0번에 저장된다.
- ACTION_POINTER_DOWN – 첫 번째 포인터 이외의 포인터에 대하여 발생된다. 포인터 데이터는 getActionIndex()이 반환하는 인덱스에 저장된다.
- ACTION_MOVE – 화면을 누르면서 이동할 때 발생한다.
- ACTION_POINTER_UP – 마지막 포인터가 아닌 다른 포인터가 화면에서 없어지면 발생된다.
- ACTION_UP – 화면을 떠나는 마지막 포인터에 대하여 발생된다.

5. 멀티 터치

인덱스와 아이디



5. 멀티 터치

□ 터치된 위치에 원을 그리는 예제

MultiTouchActivity.java

```
package kr.co.company.multitouch;
// 소스만 입력하고 Ctrl-Shift-O를 눌러서 import 문장을 자동으로 생성한다.

public class MultiTouchActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new MultiTouchView(this, null));
    }

}
```

액티비티의 화면을 MultiTouchView 객체로 설정한다.

5. 멀티 터치

MultiTouchView.java

```
package kr.co.company.multitouch;
// 소스만 입력하고 Ctrl-Shift-O를 눌러서 import 문장을 자동으로 생성한다.

public class MultiTouchView extends View {

    private static final int SIZE = 60;

    final int MAX_POINTS = 10;
    float[] x = new float[MAX_POINTS];
    float[] y = new float[MAX_POINTS];
    boolean[] touching = new boolean[MAX_POINTS];

    private Paint mPaint;

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

    private void initView() {
        mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        mPaint.setColor(Color.BLUE);
        mPaint.setStyle(Paint.Style.FILL_AND_STROKE);
    }
```

최대 10개 포인터의 위치와 상태를 저장할 수 있다.

상태정보

5. 멀티 터치

```
@Override
public boolean onTouchEvent(MotionEvent event) {

    int index = event.getActionIndex();
    int id = event.getPointerId(index);
    int action = event.getActionMasked();

    switch (action) {

    case MotionEvent.ACTION_DOWN:
    case MotionEvent.ACTION_POINTER_DOWN:
        x[id] = (int) event.getX(index);
        y[id] = (int) event.getY(index);
        touching[id] = true;
        break;
    case MotionEvent.ACTION_MOVE:
        break;
    case MotionEvent.ACTION_UP:
    case MotionEvent.ACTION_POINTER_UP:
    case MotionEvent.ACTION_CANCEL:
        touching[id] = false;
        break;
    }
    invalidate();

    return true;
}
```

인덱스로부터 포인터의 아이디를 구한다.

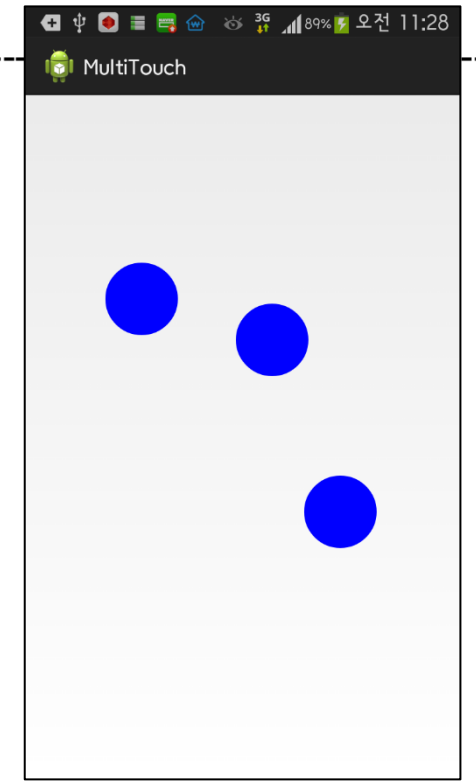
화면이 터치되면 위치를 계산하여 배열에 저장한다. touching[] 배열에 true를 저장하여서 현재 터치가 되어 있다는 것을 표시한다.

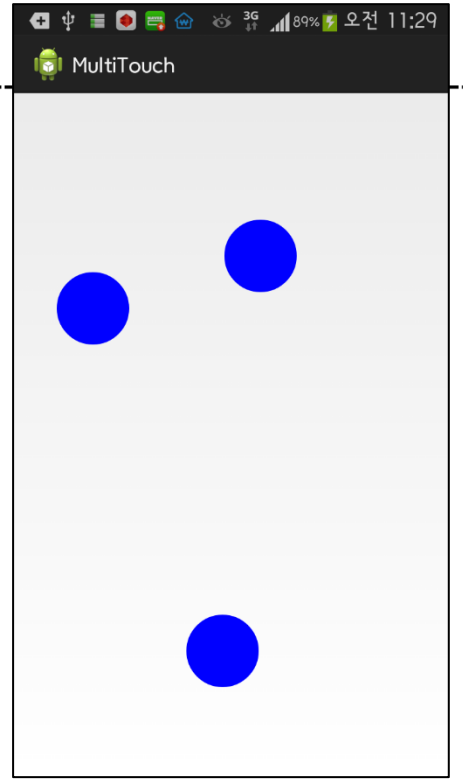
처리가 종료되었음을 저장한다.

5. 멀티 터치

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

    for (int i = 0; i < MAX_POINTS; i++) {
        if (touching[i]) {
            canvas.drawCircle(x[i], y[i], SIZE, mPaint);
        }
    }
}
}
```

현재 터치되어 있는 포인터 위치에 원을 그린다.





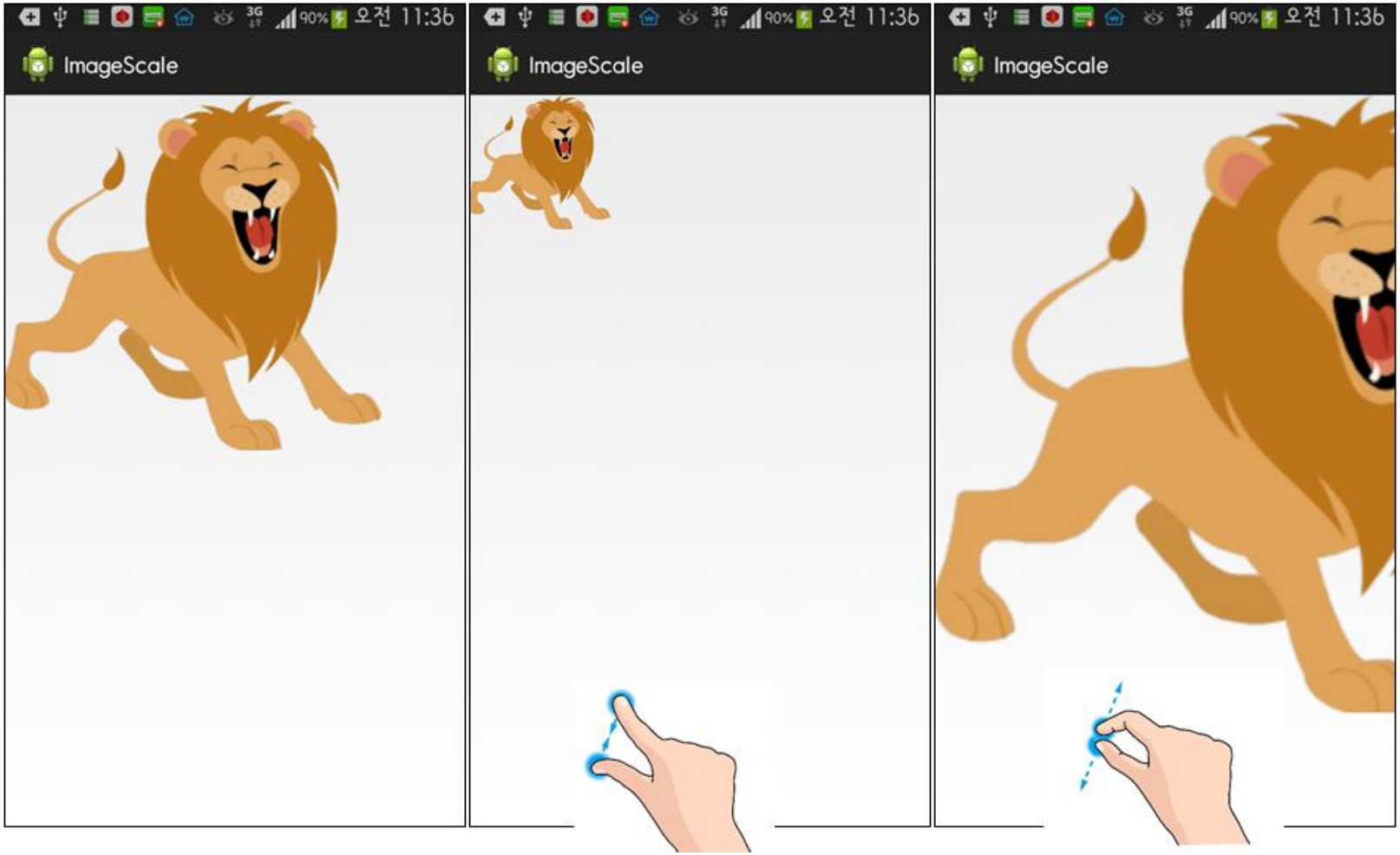
6. 제스처 이벤트

제스처를 통해 처리할 수 있는 이벤트

메소드	이벤트 유형
onDown()	- 화면이 눌렸을 경우
onShowPress()	- 화면이 눌렸다 떼어지는 경우
onSingleTapUp()	- 화면이 한 손가락으로 눌렸다 떼어지는 경우
onSingleTapConfirmed()	- 화면이 한 손가락으로 눌려지는 경우
onDoubleTap()	- 화면이 두 손가락으로 눌려지는 경우
onDoubleTapEvent()	- 화면이 두 손가락으로 눌려진 상태에서 떼거나 이동하는 등 세부적인 액션을 취하는 경우
onScroll()	- 화면이 눌린 채 일정한 속도화 방향으로 움직였다 떼는 경우
onFling()	- 화면이 눌린 채 가속도를 붙여 손가락을 움직였다 떼는 경우
onLongPress()	- 화면을 손가락으로 오래 누르는 경우

6. 제스처 이벤트

- 핀치줌 구현



6. 제스처 이벤트

액티비티 정의

ImageScaleActivity.java

```
package kr.co.company.imagescale;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class ImageScaleActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new MyImageView(this));
    }
}
```

6. 제스처 이벤트 –뷰 정의

```
public class MyImageView extends View {
    private Drawable image;
    private ScaleGestureDetector gestureDetector;
    private float scale = 1.0f;

    public MyImageView(Context context) {
        super(context);
        image = context.getResources().getDrawable(R.drawable.lion);
        setFocusable(true);
        image.setBounds(0, 0, image.getIntrinsicWidth(),
                image.getIntrinsicHeight());
        gestureDetector = new ScaleGestureDetector(context, new ScaleListener());
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.save();
        canvas.scale(scale, scale);
        image.draw(canvas);
        canvas.restore();
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        gestureDetector.onTouchEvent(event);
        invalidate();
        return true;
    }
```

제스처 인식기 객체를 생성한다.

캔버스에 신축 연산을 적용한다. 좀 더 자세한 내용은 다음 장을 참조한다.

제스처 인식기의 터치 이벤트 처리 메소드를 호출해준다.

6. 제스처 이벤트 –리스너 구현

```
    private class ScaleListener extends
            ScaleGestureDetector.SimpleOnScaleGestureListener {
        @Override
        public boolean onScale(ScaleGestureDetector detector) {
            scale *= detector.getScaleFactor();

            if (scale < 0.1f)
                scale = 0.1f;
            if (scale > 10.0f)
                scale = 10.0f;

            invalidate();
            return true;
        }
    }
}
```