# Chapter 07

## 슈팅 게임 개발 (Advanced)

(Shooting Game - Advanced 1of2)

# Contents

# 7.1 터치 입력을 통한 게임 조작

## ❖ 터치 입력을 통한 게임 진행

### ◆ 게임 시작시 초기 메뉴 화면을 기준으로 구현할 때

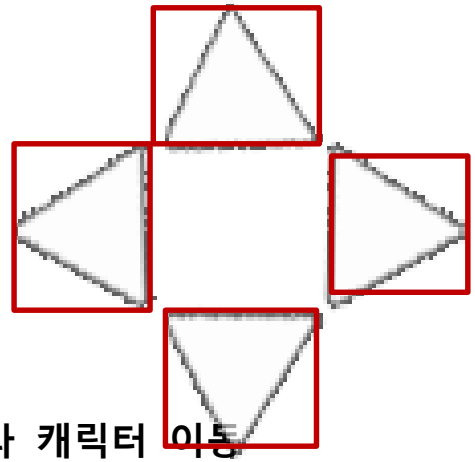
- ✓ 화면에 있는 메뉴의 Rect 값을 모두 구해서 리스트에 넣음
- ✓ onTouchEvent를 통해 넘어온 터치 이벤트의 좌표 x, y 값을 구함
- ✓ 루프를 돌면서 메뉴들의 Rect 리스트와 터치 이벤트의 좌표 값을 비교
- ✓ 비교 중에 특정 Rect 속에 터치 좌표가 있다면 그 Rect 에 해당하는 메뉴의 기능을 실행

### ◆ 버튼요소를 하나씩 직접 코드로 작성해도 되지만,

- ✓ 버튼 요소가 많은 것을 구현할 때는 클래스화해서 값을 비교하고, 그에 따른 수행으로 처리

# 7.1 터치 입력을 통한 게임 조작

## ❖ 플레이어의 이동과 미사일 발사 구현



### ◆ 터치 입력 처리

- ✓ 방향키에 따른 Rect 값을 저장
- ✓ 그에 따라 이동을 처리
- ✓ GameState 클래스에 onTouchEvent 메서드 재정의
- ✓ 버튼의 영역을 판별해서 해당 영역을 터치하면 그에 따라 캐릭터 이동

```
public class GameState implements IState  {
    … …
    @Override
    public boolean onTouchEvent ( MotionEvent event )  {
            int action = event.getAction( );
            int _x, _y;
            _x = (int) event.getX( );
            _y = (int) event.getY( );
            Rect rt = new Rect( );
            // 왼쪽 방향을 터치할 경우
            rt.set (5, 385, 45, 425);
            if (rt.contains(_x,, _y))  {  /* 플레이어 왼쪽으로 이동 */  }
            // 위쪽 방향을 터치할 경우
            rt.set (40, 345, 80, 385);
            if (rt.contains(_x, _y))  {  /* 플레이어를 위쪽으로 이동 */  }
            // 오른쪽 방향을 터치할 경우
            rt.set (80, 385, 120, 425);
            if (rt.contains(_x, _y))  {  /* 플레이어를 오른쪽으로 이동 */  }
            // 아래쪽 방향을 터치할 경우
            rt.set (40, 425, 80, 465);
            if (rt.contains(_x, _y))  {  /* 플레이어를 아래쪽으로 이동 */  }
    }
    … …
}
```

# 7.1 터치 입력을 통한 게임 조작

## ❖ 플레이어의 이동과 미사일 발사 구현 (cont.)

### ◆ 터치 입력 처리

#### ✓ 참고) D-Pad (Directional Pad) 구현

- 게임 하단에 동그란 원을 두고, 이를 누린 뒤 때지 않고 원하는 방향으로 손가락을 움직이면 플레이어가 원하는 방향으로 움직임
- 화면에 좌측 하단에 원을 두고 (D-Pad), D-Pad의 원점을 정해둔 뒤 onTouchEvent에서 DOWN 이벤트가 발생하면 플레이어의 움직임을 시작하고, DOWN 이벤트가 발생한 후에 MOVE 이벤트가 발생할 때 마다 플레이어가 움직이는 방향 값을 조정해주면서 구현

# 7.1 터치 입력을 통한 게임 조작

## ❖ 멀티 터치 구현

### ◆ 멀티 터치 예제 프로젝트

- ✓ 안드로이드 아이콘 이미지를 두 개 그림
- ✓ 하나만 터치하면 터치 이벤트가 발생한 좌표로 아이콘이 이동
- ✓ 멀티터치 시에는 각각 터치 이벤트가 발생한 좌표로 이동

- ✓ MultiTouchExample 프로젝트 생성
- ✓ MultiTouchExampleView 뷰 클래스를 생성한 후 코드 작성

# 7.1 터치 입력을 통한 게임 조작

```
public class MultiTouchExample extends Activity  {
    @Override
    public void onCreate (Bundle savedInstanceState)  {
            super .onCreate (savedInstanceState);
            setContentView( new MultiTouchExampleView( this ));
    }
}
```

```
public class MultiTouchExampleView extends View  {
    public MultiTouchExampleView(Context context)  {
            super (context);
    }
}
```

# 7.1 터치 입력을 통한 게임 조작

## ❖ 멀티 터치 구현 (cont.)

### ◆ 멀티 터치 예제 프로젝트

- ✓ 안드로이드 아이콘 그리기

# 7. 1 [illegible]

```
public class MultiTouchExampleView extends View  {
    //  아이콘 1
    int m_x_1;
    int m_y_1;
    // 아이콘 2
    int m_x_2;
    int m_y_2;
    … …
    @Override
    protected void onDraw (Canvas canvas)  {
            canvas.drawColor(Color. BLACK);
            // 두 아이콘이 겹치지 않게
            // 첫번째 아이콘은 터치 이벤트의 왼쪽에 그려지게 함
            canvas.drawBitmap(BitmapFactory.decodeResource(getResource( ),
                                R.drawable. icon), m_x_1-50, m_y_1, null );
            // 두번재 아이콘 그리기
            canvas.drawBitmap(BitmapFactory.decodeResource(getResource( ),
                                R.drawable. icon), m_x_2, m_y_2, null );
            super .onDraw(canvas);
    }
}
```

# 7.1 터치 입력을 통한 게임 조작

## ❖ 멀티 터치 구현 (cont.)

### ◆ 멀티 터치 예제 프로젝트

✓ 멀티 터치를 구현하기 전

- onTouchEvent 메서드를 재정의하고 코딩 → 컴파일하고 실행

```
public class MultiTouchExampleView extends View  {
    … …
    @Override
    public boolean onTouchEvent(MotionEvent event)  {
            // 이벤트에 따른 처리
            m_x_1 = (int) event.getX( );
            m_y_1 = (int) event.getX( );
            m_x_2 = (int) event.getX( );
            m_y_2 = (int) event.getX( );
            // 화면 갱신
            invalidate( );
            return true;
    }   }
```

# 7.1 터치 입력을 통한 게임 조작

## ❖ 멀티 터치 구현 (cont.)

### ◆ 멀티 터치 예제 프로젝트

- ✓ 멀티 터치 구현 위해 실제 기기 사용 (Android 2.0 이상)
- ✓ 기기를 위한 테스팅 방법
  - 안드로이드폰에서 환경설정(setting)을 실행
  - 응용 프로그램(Applications) – 개발(Development)을 선택
  - USB 디버깅(USB debugging)을 활성화
  - 안드로이드폰을 PC와 USB로 연결 → 드라이버 설치
  - 이클립스에서 프로그램 실행시 Run – Android Application을 선택
  - ‘Android Device Chooser’ 대화상자에서 실제 기기를 선택하고 OK
  - 프로그램이 기기에서 자동으로 설치되고 실행

# 7.1 터치 입력을 통한 게임 조작

## ❖ 멀티 터치 구현 (cont.)

### ◆ 멀티 터치 예제 프로젝트

#### ✓ 기기를 위한 테스팅 방법

- **참고) 오류**

```
Failed to upload project.apk on device 'emulator-5554'
java.io.IOException: Unable to upload file: timeout
Launch canceled!
```

» 음악파일이나 이미지 파일이 많은 경우 기기에 업로드하는데 걸리는 시간이 이클립스에 정해준 시간을 초과해서 발새하는 에러 메시지

» 이클립스에서 window → preferences → Android → DDMS

» 'ADB connection time out (ms) :' 에서 시간을 설정

# 7.1 터치 입력을 통한 게임 조작

## ❖ 멀티 터치 구현 (cont.)

### ◆ 멀티 터치 예제 프로젝트

#### ✓ 안드로디의 멀티 터치

- 터치되는 접촉면을 각각 포인터로 정의
- 소프트웨어로는 무한 개 지원 가능하나, 기기마다 최소 1개에서 최근에는 4개까지 포인터를 지원
- 멀티터치의 확인 여부
  - » `MotionEvent.getPointerCount( );`

# 7.1 터치 입력을 통한 게임 조작

## ❖ 멀티 터치 구현 (cont.)

### ◆ 멀티 터치 예제 프로젝트

#### ✓ 멀티 터치 처리

```
public class MultiTouchExampleView extends View  {
    … …
    @Override
    public boolean onTouchEvent(MotionEvent event)  {
            // 접촉면의 개수만큼 터치 입력을 처리
            if (event.getPointerCount( ) > 1)  {
                // 멀티 터치 이벤트의 경우
            } else  {
                // 싱글터치 이벤트의 경우
            }
            // 화면 갱신
            invalidate( );
            return true;
    }   }
```

# 7.1 터치 입력을 통한 게임 조작

## ❖ 멀티 터치 구현 (cont.)

### ◆ 멀티 터치 예제 프로젝트

#### ✓ 멀티 터치 처리

- 접촉면이 하나가 아니기 때문에 포인터 인덱스를 인자로 전달해서 해당 접촉면의 좌표를 알려주는 메서드 사용

```
MotionEvent.getX(포인터인덱스);
MotionEvent.getY(포인터인덱스);
```

- 멀티 터치 이벤트이면 각 접촉면의 좌표 값을 얻어 각 아이콘의 위치 값으로 대입
- 싱글 터치 이벤 한 곳으로 두 아이콘을 모아주게 작성

- 컴파일하고 실행

```
public class MultiTouchExampleView extends View  {
    … …
    @Override
    public boolean onTouchEvent(MotionEvent event)  {
            // 접촉면의 개수만큼 터치 입력을 처리
            if (event.getPointerCount( ) > 1)  {
                // 멀티 터치 이벤트의 경우
                // 첫번째 접촉면
                m_x_1 = (int) event.getX(0);
                m_y_1 = (int) event.getY(0);
                // 두번째 접촉면
                m_x_2 = (int) event.getX(1);
                m_y_2 = (int) event.getY(1);
            } else  {
                // 싱글터치 이벤트의 경우
                m_x_1 = (int) event.getX( );
                m_y_1 = (int) event.getY( );
                m_x_2 = (int) event.getX( );
                m_y_2 = (int) event.getY( );
            }
            // 화면 갱신
            invalidate( );
            return true;
    }   }
```

# 7.1 터치 입력을 통한 게임 조작

## ❖ 멀티 터치 구현 (cont.)

### ◆ 게임에 적용 과제

# 7.2 안드로이드의 센서

## ❖ 센서 종류

### ◆ 가속도 센서 / 방향 센서

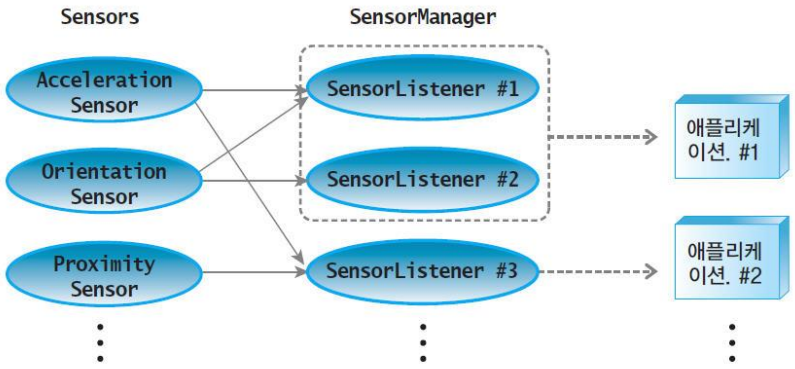
| 상수 | 설명 |
|---|---|
| TYPE_ALL | 모든 센서 타입을 정의한 상수 |
| TYPE_ACCELEROMETER | 가속도를 감지하는 센서의 타입 |
| TYPE_GYROSCOPE | 모션 센서의 정밀한 교정을 위해 쓰이는 자이로스코프 센서의 타입 |
| TYPE_LIGHT | 주변의 빛을 감지하는 센서의 타입 |
| TYPE_MAGNETIC_FIELD | 주변의 자기장을 감지하는 센서의 타입 |
| TYPE_ORIENTATION | 기기의 방향을 감지하는 센서의 타입 |
| TYPE_PRESSURE | 기기에 적용되는 압력을 감지하는 센서의 타입 |
| TYPE_PROXIMITY | 특정 물체와 근접한 정도를 감지하는 센서의 타입 |
| TYPE_TEMPERATURE | 기기 근처의 온도를 감지하는 센서의 타입 |

# 7.2 안드로이드의 센서

## ❖ 방향 센서

### ◆ 센서 예제 애플리케이션

- ✓ 기본 아이콘
- ✓ 기기를 움직이는 것에 따라 아이콘이 움직임
- ✓ 화면에 방향 센서의 정보 표시
- ✓ SensorExample 프로젝트 생성
- ✓ SensorExampleView 클래스 생성



# 7.2 안드로이드의 센서

```
public class SensorExample extends Activity  {
    @Override
    public void onCreate (Bundle savedInstanceState)  {
            super .onCreate (savedInstanceState);
            setContentView( new SensorExampleView( this ));
    }
}
```

```
public class SensorExampleView extends View  {
    public SensorExampleView(Context context)  {
            super (context);
    }

    @Override
    protected void onDraw (Canvas canvas)  {

    }
}
```

# 7.2 안드로이드의 센서

## ❖ 방향 센서 (cont.)

### ◆ 센서 예제 애플리케이션

✓ 화면에 아이콘 띄우기

```
public class SensorExampleView extends View  {
    // 아이콘의 좌표
    int m_x = 0;
    int m_y = 0;
    … …

    @Override
    protected void onDraw (Canvas canvas)  {
            // 아이콘 그리기
            canvas.drawBitmap(BitmapFactory.decodeResource(getResource( ),
                                R.drawable. icon), m_x, m_y, null );
    }
}
```

# 7.2 안드로이드의 센서

## ❖ 방향 센서 (cont.)

### ◆ 방향 센서 사용 과정

- ✓ SensorEventListener의 역할을 할 클래스 생성
- ✓ Device에서 SensorManager를 가져옴
- ✓ SensorManager에 Listener로 생성한 클래스를 등록
- ✓ 받아온 센서 값을 처리

# 7.2 안드로이드의 센서

## ❖ 방향 센서 (cont.)

### ◆ 방향 센서 사용 과정

✓ SensorEventListener의 역할을 할 클래스 생성

- 콜백 방식으로 이벤트를 받아와서 처리 가능
- 클래스 따로 생성 가능하지만, 뷰 자체 포함해서 사용도 가능
- 가상 메서드 재정의

# 7. 2 안드로이드의 센서

```
public class SensorExampleView extends View implements SensorEventListener  {
    … …

    @Override
    public void onSensorChanged (SensorEvent event)  {
            // TODO Auto-generated method stub
    }

    @Override
    public void onAccuracyChanged (Sensor arg0, int arg1)  {
            // TODO Auto-generated method stub
    }
}
```

# 7.2 안드로이드의 센서

## ❖ 방향 센서 (cont.)

### ◆ 방향 센서 사용 과정

✓ SensorEventListener의 역할을 할 클래스 생성

- OnSensorChanged : 센서의 값이 바뀔 때 호출
- OnAccuracyChanged : 센서의 정확도 값이 바뀔 때 호출
  - » 정확한 센서 값을 요하는 애플리케이션에서 센서 데이터를 조정하는 코드를 넣어 사용

✓ Device에서 SensorManager를 가져옴

- SensorManager sensorManager =
  (SensorManager)context.getSystemService(Context.SENSOR_SERVICE);
- SensorManager를 멤버 변수로 추가하고, 생성자에서 SensorManager를 가져오면 됨

# 7. 2 안드로이드의 센서

```
public class SensorExampleView extends View implements SensorEventListener  {
    SensorManager m_sensorManager;
    … …

    public void SensorExampleView (Context context)  {
            super (context);
            m_sensorManager = (SensorManager) context.getSystemService
                                    (Context. SENSOR_SERVICE);
    }

    … …
}
```

# 7.2 안드로이드의 센서

## ❖ 방향 센서 (cont.)

### ◆ 방향 센서 사용 과정

✓ SensorManager에 Listener로 생성한 클래스를 등록

```
SensorManager.registerListener(센서이벤트리스너, 센서타입, 센서속도);
```

- 여기서는 SensorExampleView가 SensorEventListener를 포함하고 있으므로 this
- 센서타입은 방향센서(TYPE_ORIENTATION)
- 센서속도

| 상수 | 설명 |
|---|---|
| SensorManager.SENSOR_DELAY_FASTEST | 가장 빠른 센서 업데이트 속도 |
| SensorManager.SENSOR_DELAY_GAME | 게임 제어에 적합한 업데이트 속도 |
| SensorManager.SENSOR_DELAY_NORMAL | 기본적인 업데이트 속도 |
| SensorManager.SENSOR_DELAY_UI | UI 처리에 적합한 업데이트 속도 |

# 7. 2 안드로이드의 센서

```
public class SensorExampleView extends View implements SensorEventListener  {
    SensorManager m_ensorManager;
    … …

    public void SensorExampleView (Context context)  {
            super (context);
            m_sensorManager = (SensorManager) context.getSystemService
                                    (Context. SENSOR_SERVICE);
            m_sensorManager .registerListener( this,
                        m_sensorManager .getDefalutSensor (Sensor. TYPE_ORIENTATION),
                        SensorManager. SENSOR_DELAY_GAME);
    }

    … …
}
```

# 7.2 안드로이드의 센서

## ❖ 방향 센서 (cont.)

### ◆ 방향 센서 사용 과정

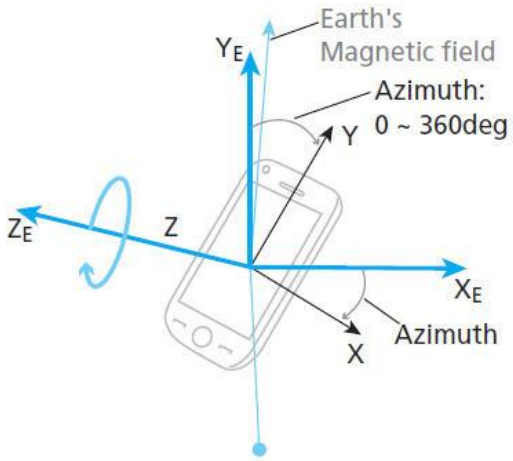
#### ✓ 받아온 센서 값을 처리

- onSensorChanged 메서드에서 인자로 넘어오는 SensorEvent를 처리
- SensorEvent의 대표적인 정보

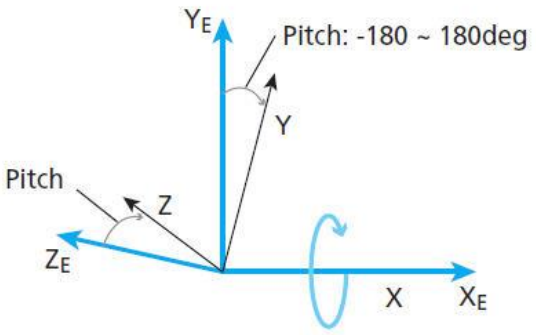
  SensorEvent.sensorgetType : 센서 이벤트의 종류

  SensorEvent.values[ ] : 센서 이벤트의 타입에 따른 센서 데이터 값
- 방향 센서의 값
  - » 헤딩 (heading) ; Z축 주변을 향하고 있는 장치의 방향 ( Yaw 또는 Bearing)
  - » 피치 (pitch) ; Y축 주변 장치각을 나타내는 장치의 방햔
  - » 롤 (roll) ; X축 주변 장치각을 나타내는 장치의 방향

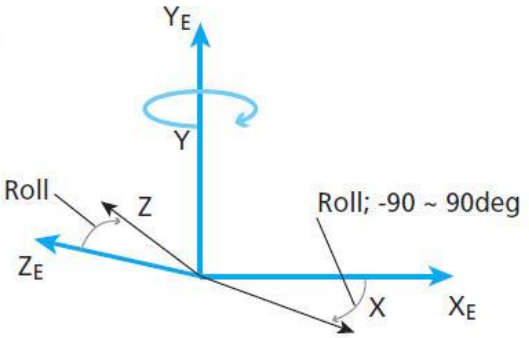
# 7. 2 안드로이드의 센서




헤딩(heading)




피치(pitch)




롤(roll)

# 7.2 안드로이드의 센서

## ❖ 방향 센서 (cont.)

### ◆ 방향 센서 사용 과정

✓ 받아온 센서 값을 처리

- 헤딩은 0 ~ 360°의 값을 가지며, 이를 이용해 북쪽은 0/360, 90 간격으로 동, 남, 서로 나침반 정보를 가질 수 있음
- 피치는 Y축의 방향으로 장치의 머리가 하늘을 향할 때 -90, 장치가 바닥에 뒤집혀 있을 때 180/-180의 값을 갖음
- 롤은 기기를 손으로 잡고 좌우로 기울이는 것에 따라 -90 ~ 90°의 값을 갖음
- 피치와 롤 값은 기기를 평평한 곳에 두었을 때 0

✓ 실제 기기에서 실행

# 7. 2 안드로이드의 센서

```
public class SensorExampleView extends View implements SensorEventListener  {
    … …
    String m_str = “”;
    … …
    @Override
    protected void onDraw (Canvas canvas)  {
            canvas.drawBitmap (BitmapFactory.decodeResource(getResource( ),
                    R.drawable. icon), m_x, m_y, null);
            // 방향 센서 값을 화면에 표시
            Paint p = new Paint( );
            p.setTextSize(20);
            p.setColor(Color. WHITE);
            canvas.drawText( m_str, 0, 20, p);
    }
```

# 7. 2 안드로이드의 센서

```
@Override
public void onSensorChanged (SensorEvent event)  {
        synchronized (this)  {
            switch (event. sensor .getType( ))  {
                    // 방향 센서 값의 경우
                    case Sensor. TYPE_ORIENTATION:
                        float Heading = event. values [0];
                        float Pitch = event. values [1];
                        float Roll = event. values [2];

                        m_str = "(Orientation ) ";
                        m_str += "Heading : " + Float.toString(Heading);
                        m_str += " Pitch : " + Float.toString(Pitch);
                        m_str += " Roll : " + Float.toString(Roll);
            }
        }
        // 화면 갱신
        invalidate( );
}
... ...
}
```

# 7.2 안드로이드의 센서

## ❖ 방향 센서 (cont.)

### ◆ 센서 예제 애플리케이션

✓ 센서 방향 값에 따른 처리

- 움직이는 제스처를 잡아내는 것은 X, Y축을 나타내는 롤과 피치
- 이를 이용해 아이콘을 움직이는 코드

✓ 기기에서 실행

# 7. 2 안드로이드의 센서

```
public class SensorExampleView extends View implements SensorEventListener {
    … …
    @Override
    public void onSensorChanged (SensorEvent event) {
            synchronized (this) {
                switch (event. sensor .getType( )) {
                        // 방향 센서 값의 경우
                        case Sensor. TYPE_ORIENTATION:
                            float Heading = event. values [0];
                            float Pitch = event. values [1];
                            float Roll = event. values [2];

                            m_str = "(Orientation ) ";
                            m_str += "Heading : " + Float.toString(Heading);
                            m_str += " Pitch : " + Float.toString(Pitch);
                            m_str += " Roll : " + Float.toString(Roll);
```

# 7. 2 안드로이드의 센서

```
                        // 아이콘 움직이기
                        m_x -= Roll;
                        m_y -= Pitch;

                        // 화면 크기에 대한 처리
                        if ( m_x <= 0 )      m_x = 0;
                        if ( m_y <= 0 )      m_y = 0;
                        if ( m_x >= getWidth( )-50 )  m_x = getWidth( );
                        if ( m_y >= getHeight( )-50)  m_y = getHeight( );
                        break;
                    } // switch
                } // synchronized
                // 화면 갱신
                invalidate( );
    }
    ... ...
}
```

# 7.3 바이브레이터 시스템

## ❖ 진동 시스템 (Vibrator System)

### ◆ 바이브레이터 예제 프로그램

- ✓ 화면을 터치하면 진동
- ✓ VibratorExample
- ✓ VibratorExampleView

# 7.3 바이브레이터 시스템

```
public class VibratorExample extends Activity  {
    @Override
    public void onCreate (Bundle savedInstanceState)  {
            super .onCreate (savedInstanceState);
            setContentView( new VibratorExampleView( this ));
    }
}
```

```
public class VibratorExampleView extends View  {
    public VibratorExampleView(Context context)  {
            super (context);
    }

    @Override
    public boolean onTouchEvent (MotionEvent event)  {
            // 터치 이벤트 발생할 시
            return super .onTouchEvent(event);
    }
}
```

# 7.3 바이브레이터 시스템

## ❖ 진동 시스템 (Vibrator System)

### ◆ 바이브레이터 예제 프로그램

- ✓ Vibrator 클래스의 인스턴스 필요
- ✓ 시스템 서비스이므로 운영체제에서 얻어옴

```
Vibrator vibrator = (Vibrator)getSystemService(Context.VIBRATOR_SERVICE);
```

```
public class VibratorExampleView extends View  {
    Vibrator m_vibrator;
    public VibratorExampleView(Context context)  {
            super (context);
            // 시스템에서 Vibrator를 얻어옴
            m_vibrator = (Vibrator) context.getSystemService
                                    (Context. VIBRATOR_SERVICE);
    }
    … …
}
```

# 7.3 바이브레이터 시스템

## ❖ 진동 시스템 (Vibrator System) (cont.)

### ◆ 바이브레이터 예제 프로그램

✓ 진동을 일으키는 메서드

```
Vibrator.vibrate(진동할 시간);
```

- 진동 시간은 밀리초 단위

```
public class VibratorExampleView extends View  {
    … …
    @Override
    public boolean onTouchEvent (MotionEvent event)  {
            // 터치 이벤트 발생할 시에 진동 효과
            m_vibrator .vibrate(100);
            // false를 반환하여 DOWN일 때만 이벤트를 발생하게 함
            return false;
    }
}
```

# 7.3 바이브레이터 시스템

## ❖ 진동 시스템 (Vibrator System) (cont.)

### ◆ 바이브레이터 예제 프로그램

- ✓ 실행을 위해 하드웨어 접근 허가 필요
- ✓ 접근 허가
  - 메니페스트 파일을 수정
  - AndroidManifest.xml 파일에 사용자 권한 추가

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... ...>
    ... ...
    <uses-permission android:name="android.permission.VIBRATE" />
</manifest>
```

# 7.3 바이브레이터 시스템

## ❖ 슈팅 게임

- 기기의 방향 센서를 이용하여 움직이도록
- 화면 터치시 미사일 발사
- 생명이 -1일 때마다 진동