第 1 部 Android プログラミング入門

徐　丙鉄

Android 開発環境

Java SE 6 + SDK ADT Bundle (Eclipse)
ADT：Android Development Tools
AVD：Android Virtual Device（Android エミュレーター）
Windows では，エミュレータのメモリ割り当ては
769MB 以下にする。ヒープメモリは 64MB 程度。

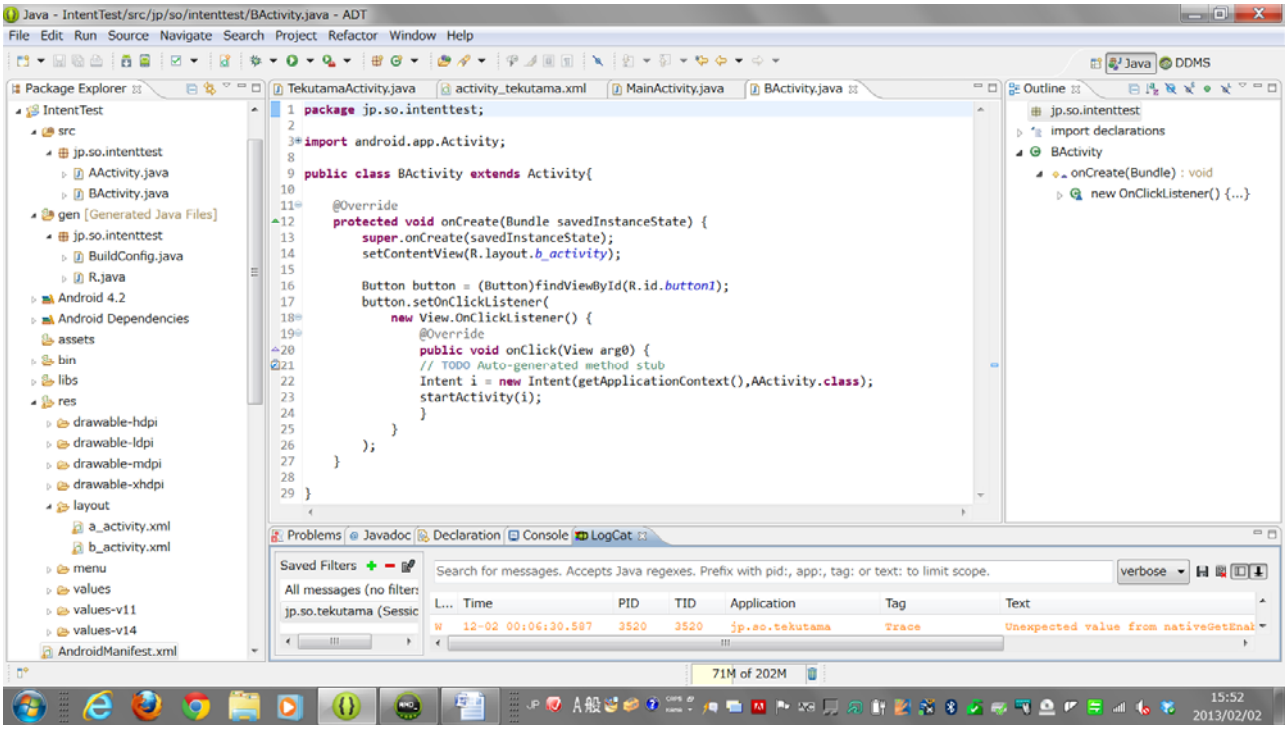
Eclipse：統合開発環境

ショートカット

自動インポート	Windows：Ctrl + Shift + O
補完	Ctrl + space
クイックフィックス	Ctrl + 1
クイックアシスト	Ctrl + 2

デバグ

Log.d(“TekutamaLevel” , ”accel” + value);



開発用 Android 端末の準備

開発では，PC に USB で Android を接続して，アプリの動作確認をするのが，手っ取り早い。

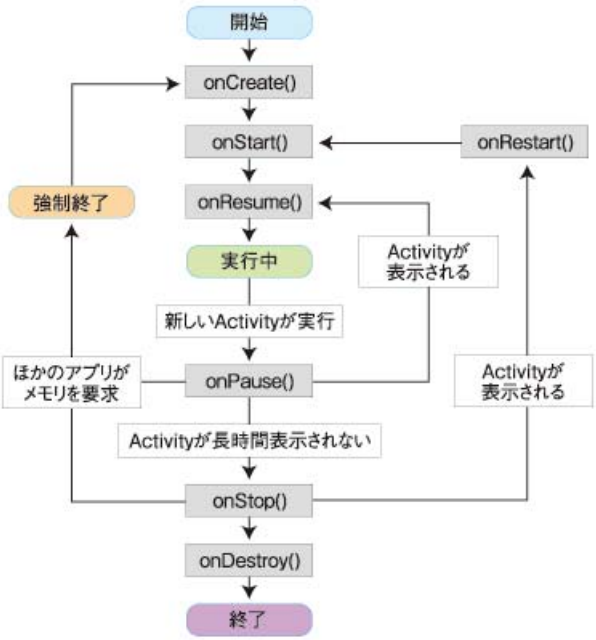
Android 端末の設定：　「設定」→「セキュリティ」→「提供元不明のアプリ」をチェック
「設定」→「開発者向けオプション」→「USB でバッグ」をチェック

Activity：Android のプログラムの最小基本単位のクラス。一つの Activity は一つの View を持つ。
また，一般にプログラムは複数の Activity から構成される。

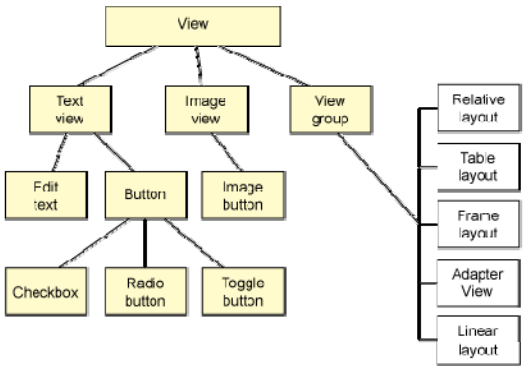
View：アプリの画面の描画クラス

void onDraw(Canvas canvas)
invalidate ()　　～を無効にする、、invalidate すると onDraw が呼び出される。

Activity のライフサイクル



http://www.atmarkit.co.jp/ait/articles/0901/19/news122.html

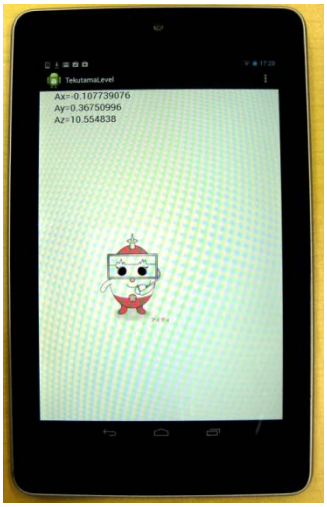


http://www.ibm.com/developerworks/jp/opensource/tutorials/os-eclipse-androidwidget/

Google play 登録料　$ 25
（参考：App Store,
OS Developer Program　$ 99/年）

Android アプリ（.apk）

てくたま水準器：Android 端末を面に合わせると「てくたま」が傾きの方向に動きます。



```
package kindai.so.tekutamalevel;
import java.util.List;
import com.example.tekutamalevel.MyView;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.TextView;

public class MainActivity extends Activity implements SensorEventListener{

    SensorManager sensorManager;        //センサーを管理する SensorManager
    MyView myView;                      //てくたまを描画する View

        @Override
        protected void onCreate( Bundle savedInstanceState ) {
                super.onCreate( savedInstanceState );
                myView = new MyView( getApplication( ) );
                setContentView( myView );
                sensorManager = (SensorManager)getSystemService( SENSOR_SERVICE);
        }
```

```
	@Override
	public boolean onCreateOptionsMenu( Menu menu ) {
		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate( R.menu.main, menu );
		return true;
	}

	@Override
	protected void onResume( ) {
		// TODO Auto-generated method stub
		super.onResume( );
		// Listener の登録
		List<Sensor> sensors =
			sensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER);
		if( sensors.size() > 0 ) {
			Sensor s = sensors.get(0); //加速度センサー
			sensorManager.registerListener( this , s ,
					SensorManager.SENSOR_DELAY_UI );
		}
	}

	@Override
	protected void onStop( ) {
		// TODO Auto-generated method stub
		super.onStop();
		// sensor の Listener の登録解除
		sensorManager.unregisterListener(this);
	}

	@Override
	public void onSensorChanged( SensorEvent event ) {
		// TODO Auto-generated method stub
		if( event.sensor.getType() == Sensor.TYPE_ACCELEROMETER ) {
			// myView へ加速度の値をセット，値は float で順番に x,y,z 成分
			myView.setA( event.values[0] , event.values[1] , event.values[2] );
	    }
	}

	@Override
	public void onAccuracyChanged( Sensor sensor, int accuracy ) {
		// TODO Auto-generated method stub
	}

}


package kindai.so.tekutamalevel;
```

```
import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.view.MotionEvent;
import android.view.View;

public class MyView extends View{
	float x, y;			//てくたまの座標
	Bitmap img;			//てくたまの Bitmap イメージ
	Paint paint;			//View の Paint
	float Ax, Ay, Az;		//加速度の成分

	public MyView(Context context) {
		super(context);
		// TODO Auto-generated constructor stub
		x = (float)200.0;  y = (float)300.0;
		paint = new Paint();
			paint.setARGB(255,0,0,0); paint.setTextSize(32);
		Resources res = context.getResources();
		img = BitmapFactory.decodeResource( res, R.drawable.tekutama );
	//てくたまの画像ファイル tekutama.gif は，res フォルダ内の drawable フォルダに置く
	}

	// タッチするとその位置をてくたまの座標 x,y に設定
	public boolean onTouchEvent(MotionEvent event){
		switch( event.getAction() ){
			case MotionEvent.ACTION_MOVE:
				x = event.getX();
				y = event.getY();
				break;
			}
		return true;
	}

	//描画指示
	protected void onDraw(Canvas canvas){
		super.onDraw( canvas );
		canvas.drawBitmap(img, x -100, y -100, paint );
		canvas.drawText( "Ax=" + Ax , 50, 30, paint );
		canvas.drawText( "Ay=" + Ay , 50, 70, paint );
		canvas.drawText( "Az=" + Az , 50, 110, paint );
	}

	//Activity から加速度の値をセットする関数
	public void setA( float Ax , float Ay , float Az ){
```

```
                this.Ax = Ax;   this.Ay = Ay;   this.Az = Az;
                x -= 5*Ax;   y += 5*Ay;
                invalidate( );      //invalidate() すると onDraw() が呼び出される
        }
}
```