Les Capteurs Android

Cet article est rédigé par Android2EE, Expertise et Formation Android.

Il est associé à un ensemble de tutoriaux vous montrant comment utiliser chacun de ces capteurs. Pour plus d'information (Tutoriels, EBooks, Formations), une seule adresse :

Android2EE : http://www.android2ee.com

1 Introduction

Pour être honnête depuis que j'ai douze ans, je rêve d'avoir un objet (une montre à l'époque) avec tout un tas de capteurs. Et voilà que vingt ans plus tard mon rêve est exaucé avec les smartphones Android.

C'est donc avec plaisir que je vais essayer d'expliquer comment les utiliser. Par nature, il y aura des notions mathématiques associées à ces explications.

Je vais tout d'abord me pencher sur deux capteurs simples, celui de la lumière et celui de la proximité.

J'expliquerai les capteurs d'accélération qui sont le capteur d'accélération, le capteur de gravité et le capteur d'accélération linéaire.

Nous aborderons le capteur magnétique, le capteur d'orientation (à se pendre), le gyroscope et le vecteur de rotation.

Les capteurs de pression atmosphérique et de température ne seront pas traités (vous n'en aurez pas besoin quand vous aurez fini de lire l'article).

À cet article sont associés neuf tutoriels vous expliquant chacune de ces notions. Vous pouvez les retrouver sur android2ee.com.

Mais avant toute chose, le premier chapitre est dédié à ce qui est commun à l'ensemble des capteurs. Nous abordons ainsi les capteurs en eux-mêmes (l'objet Sensor). Nous montrons comment les instancier, les utiliser pour récupérer leur changement de valeurs. Ce qui nous amène à l'étude du SensorEvent. Et nous concluons ce chapitre par la correction des valeurs du capteur en fonction de l'orientation de l'écran.

2 Les notions communes à tous les capteurs

Ce chapitre rassemble l'ensemble des notions communes aux capteurs.

2.1 Le repère utilisé

La plupart des capteurs à trois dimensions sont associés au repère orthonormé suivant :

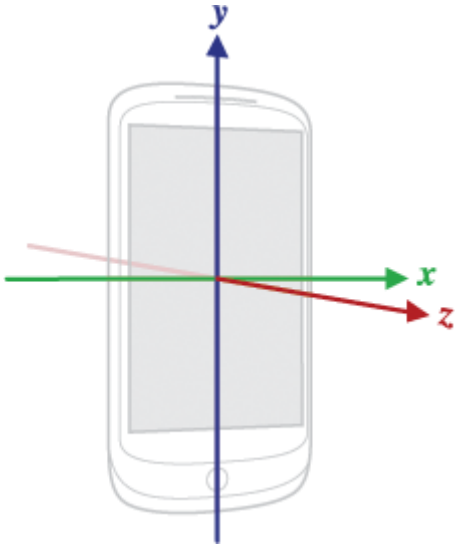


Figure 1_:provenant de google (http://developer.android.com/images/axis_device.png)

Ce repère est celui utilisé par les capteurs accéléromètre, gyroscope, électromagnétisme. L'orientation n'utilise pas ce repère.

2.2 Lister les capteurs présents et écouter leur spécificité

Pour lister ou récupérer un capteur particulier, il suffit d'instancier le SensorManager et ses méthodes getSensorList ou getDefaultSensor.

```
/** * The sensor manager */
SensorManager sensorManager;

/* * (non-Javadoc) *  * @see android.app.Activity#onCreate(android.os.Bundle) */
@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	// Instanicer le SensorManager
	sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
	// Faire la liste des capteurs de l'appareil
	listSensor();
}


/*****************************************************************************************/
/** Lister les capteurs existant ***************************************************************/
/*****************************************************************************************/
/** * Trouve la liste de tous les capteurs existants, trouve un capteur spécifique ou l'ensemble des capteurs d'un
type fixé. */
private void listSensor() {
	// Trouver tous les capteurs de l'appareil :
	List<Sensor> sensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
	// La chaîne descriptive de chaque capteur
	StringBuffer sensorDesc = new StringBuffer();
	//pour chaque capteur trouvé, construire sa chaîne descriptive
	for (Sensor sensor : sensors) {
		sensorDesc.append("New sensor detected : \r\n");
		sensorDesc.append("\tName: " + sensor.getName() + "\r\n");
		sensorDesc.append("\tType: " + getType(sensor.getType()) + "\r\n");
		sensorDesc.append("Version: " + sensor.getVersion() + "\r\n");
		sensorDesc.append("Resolution (in the sensor unit): " + sensor.getResolution() + "\r\n");
```

```
                sensorDesc.append("Power in mA used by this sensor while in use" + sensor.getPower() +
                        "\r\n");
                sensorDesc.append("Vendor: " + sensor.getVendor() + "\r\n");
                sensorDesc.append("Maximum range of the sensor in the sensor's unit." +
                        sensor.getMaximumRange() + "\r\n");
                sensorDesc.append("Minimum delay allowed between two events in microsecond"
                                + " or zero if this sensor only returns a value when the data it's measuring
                                changes"
                                + sensor.getMinDelay() + "\r\n");
        }
        //Faire quelque chose de cette liste
        Toast.makeText(this, sensorDesc.toString(), Toast.LENGTH_LONG).show();

        // Pour trouver un capteur spécifique :
        Sensor gyroscopeDefault = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
        // Pour trouver tous les capteurs d'un type fixé :
        List<Sensor> gyroscopes = sensorManager.getSensorList(Sensor.TYPE_GYROSCOPE);
}

/**
 * @param type
 *          the integer type
 * @return the type as a string
 */
private String getType(int type) {
        String strType;
        switch (type) {
        case Sensor.TYPE_ACCELEROMETER: strType = "TYPE_ACCELEROMETER";break;
        case Sensor.TYPE_GRAVITY:strType = "TYPE_GRAVITY";break;
        case Sensor.TYPE_GYROSCOPE: strType = "TYPE_GYROSCOPE";break;
        case Sensor.TYPE_LIGHT:strType = "TYPE_LIGHT";break;
        case Sensor.TYPE_LINEAR_ACCELERATION:strType = "TYPE_LINEAR_ACCELERATION";
                break;
        case Sensor.TYPE_MAGNETIC_FIELD:strType = "TYPE_MAGNETIC_FIELD";break;
        case Sensor.TYPE_ORIENTATION:strType = "TYPE_ORIENTATION";break;
        case Sensor.TYPE_PRESSURE:strType = "TYPE_PRESSURE";break;
        case Sensor.TYPE_PROXIMITY:   strType = "TYPE_PROXIMITY"; break;
        case Sensor.TYPE_ROTATION_VECTOR:  strType = "TYPE_ROTATION_VECTOR";break;
        case Sensor.TYPE_TEMPERATURE:strType = "TYPE_TEMPERATURE";break;
        default: strType = "TYPE_UNKNOW";break;
        }
        return strType;
}
```

Le tutoriel se trouve ici : Page des tutoriels des capteurs sur Android2ee

2.3 Utiliser un capteur et écouter ses changements de valeurs

Quand on souhaite travailler avec les capteurs, la première chose à faire est de savoir écouter leur changement d'état (de valeurs). Pour cela, c'est assez simple, il suffit de les instancier puis de s'enregistrer en tant qu'écouteur dans la méthode onResume et de se désenregistrer dans la méthode onPause. Pour les écouter il suffit d'implémenter l'interface SensorEventListener (souvent au niveau de votre activité).

Au lieu d'un long discours, quelques lignes de code éclaireront cette explication. Prenons l'exemple de l'accéléromètre :

```
public class SensorAccelerationTutoActivity extends Activity implements SensorEventListener {
```

```
/** * Déclaration de l'attribut en tant qu'attribut de l'activité */
/** * Le sensor manager (gestionnaire de capteurs)*/
SensorManager sensorManager;
/**
 * L'accéléromètre
 */
Sensor accelerometer;

/*************************************************************************/
/** Manage life cycle ******************************************************/
/*************************************************************************/
	/** Appelé à la création de l'activité. */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		// Faire quelque chose
		// Gérer les capteurs :
		// Instancier le gestionnaire des capteurs, le SensorManager
		sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
		// Instancier l'accéléromètre
		accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
		// Faire d'autres trucs
	}
	/* * (non-Javadoc) *
	 * @see android.app.Activity#onPause() */
	@Override
	protected void onPause() {
		// unregister the sensor (désenregistrer le capteur)
		sensorManager.unregisterListener(this, accelerometer);
		super.onPause();
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see android.app.Activity#onResume()
	 */
	@Override
	protected void onResume() {
		/* Ce qu'en dit Google dans le cas de l'accéléromètre :
		 * «  Ce n'est pas nécessaire d'avoir les évènements des capteurs à un rythme trop rapide.
		 * En utilisant un rythme moins rapide (SENSOR_DELAY_UI), nous obtenons un filtre
		 * automatique de bas-niveau qui "extrait" la gravité  de l'accélération.
		 * Un autre bénéfice étant que l'on utilise moins d'énergie et de CPU. »
		 */
		sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_UI);
		super.onResume();
	}
/*************************************************************************/
/** SensorEventListener****************************************************/
/*************************************************************************/
/*
 * (non-Javadoc)
 *
 * @see android.hardware.SensorEventListener#onAccuracyChanged(android.hardware.Sensor, int)
 */
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
	// Rien à faire la plupart du temps
}
```

```
/* * (non-Javadoc) *
 * @see android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent) */
@Override
public void onSensorChanged(SensorEvent event) {
// Récupérer les valeurs du capteur
float x, y, z;
if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
	x = event.values[0];
	y = event.values[1];
	z = event.values[2];
}
}
```

Cet exemple est celui de l'accéléromètre, mais il s'applique à tous les capteurs.

2.4 Comprendre le SensorEvent

2.4.1 Les méthodes d'écoute des changements des capteurs

Dans l'exemple suivant, je montre comment utiliser les trois champs du SensorEvent qui sont utiles : accuracy, timestamp et values.

Dans les exemples qui suivent, le seul objectif est de présenter les différents champs.

```
@Override
public void onSensorChanged(SensorEvent event) {
	// Tous d'abord vérifier que le capteur est bien celui que l'on écoute (il y a un champ sensor nommé
	// litenedSensor dans la classe qui est le capteur que l'on écoute).
	if (event.sensor.equals(listenedSensor)) {
		//On trouve sa précision
		int accuracy = event.accuracy;
		//on l'analyse
		switch (accuracy) {
		case SensorManager.SENSOR_STATUS_ACCURACY_LOW:
		case SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM:
		case SensorManager.SENSOR_STATUS_ACCURACY_HIGH:
		case SensorManager.SENSOR_STATUS_UNRELIABLE:
		default: break;
		}
		// On trouve le moment où l'évènement a été émis
		long timestamp = event.timestamp;
		//On récupère ses valeurs (cela sera expliqué plus tard)
		float[] val = event.values;
		//et on fait quelque chose de ces informations
	}
}
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
	// Tous d'abord vérifier que le capteur est bien celui que l'on écoute (il y a un champ sensor nommé
	// litenedSensor dans la classe qui est le capteur que l'on écoute).
	if(sensor.equals(listenedSensor)){
		//On analyse sa précision
		switch (accuracy) {
		case SensorManager.SENSOR_STATUS_ACCURACY_LOW:
		case SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM:
		case SensorManager.SENSOR_STATUS_ACCURACY_HIGH:
		case SensorManager.SENSOR_STATUS_UNRELIABLE:
		default:
		break;
		}
```

```
                //et on fait quelque chose avec ça
        }
}
        //Ci-dessous les différentes valeurs du rythme de récupération des données. En d'autres termes, à la
        // vitesse d'émission de l'évènement SensorEvent
        //La vitesse par défaut
        int rate=SensorManager.SENSOR_DELAY_NORMAL;
        //La vitesse à utiliser pour les jeux
         rate=SensorManager.SENSOR_DELAY_GAME;
         //La vitesse la plus rapide qui puisse être
         rate=SensorManager.SENSOR_DELAY_FASTEST;
         //La vitesse si l'on souhaite uniquement mettre à jour l'UI
         rate=SensorManager.SENSOR_DELAY_UI;
         //L'objectif étant de s'enregistrer avec cette vitesse :
        sensorManager.registerListener(sensorOrientationListener, listenedSensor, rate);
}
```

2.4.2 Unités et structures des différents capteurs

Chaque capteur renvoie un vecteur de données sous forme de flottant (ce vecteur peut être d'une à trois dimensions). Le tableau ci-dessous synthétise la sémantique associée à ses vecteurs :

Nom	Dimension du vecteur	Unité	Sémantique	Values[]
Accelerometer	3	m/s^2	Mesure de l'accélération (gravité incluse)	[0] axe x [1] axe y [2] axe z
Gyroscope	3	Radian/ seconde	Mesure la rotation en termes de vitesse autour de chaque axe	[0] vitesse angulaire autour de x [1] vitesse angulaire autour de y [2] vitesse angulaire autour de z
Light	1	Lux	Mesure de la luminosité	[0]valeur
Magnetic_Field	3	µTesla	Mesure du champ magnétique	[0] axe x [1] axe y [2] axe z
Orientation	3	degrés	Mesure l'angle entre le nord magnétique	[0] Azimut entre l'axe y et le nord [1] Rotation autour de l'axe x (-180,180) [2] Rotation autour de l'axe y (-90,90)
Pressure	1	KPascal	Mesure la pression	[0]valeur
Proximity	1	mètre	Mesure la distance entre l'appareil et un objet cible	[0]valeur
Temperature	1	Celsius	Mesure la température	[0]valeur

Pour utiliser certains de ces capteurs un minimum de connaissance en physique est nécessaire.

2.5 Déclarer le capteur dans votre AndroidManifest

Ce détail est d'importance, en effet, l'Android Market analyse votre manifeste et proposera votre application uniquement aux appareils possédant le capteur que vous avez déclaré. Cette déclaration s'effectue comme suit :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.android2ee.android.tuto.sensor.light" android:versionCode="1"
        android:versionName="1.0">
        <uses-sdk android:minSdkVersion="10" />
        <uses-feature android:name="android.hardware.sensor.accelerometer"
                android:required="true" />
        <uses-feature android:name="android.hardware.sensor.barometer"
                android:required="true" />
        <uses-feature android:name="android.hardware.sensor.compass"
                android:required="true" />
        <uses-feature android:name="android.hardware.sensor.gyroscope"
                android:required="true" />
        <uses-feature android:name="android.hardware.sensor.light"
                android:required="true" />
        <uses-feature android:name="android.hardware.sensor.proximity"
                android:required="true" />

        …
</manifest>
```

L'exemple précédent déclare tous les capteurs connus mais spécifier uniquement celui dont vous avez besoin pour votre application.

2.6 Gérer le changement d'orientation de l'écran (du portrait au landscape)

Une problématique qui s'applique à la plupart des capteurs est la gestion du mode de l'écran (landscape ou portrait). L'idée est de savoir, quand on récupère les valeurs du capteur, l'état de l'orientation de l'écran. En fonction de cet état, on corrige les valeurs :

```
/** En attribut de la classe (de l'activité)
 * Le seul qui connaisse l'orientation de l'appareil
 */
private Display mDisplay;

/** Appelé à la création de l'activité. */
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Faire quelque chose
        // Gérer les capteurs :
        // Instancier le gestionnaire des capteurs, le SensorManager
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        // Instancier l'accéléromètre
        accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        // Faire d'autres trucs
}
/* * (non-Javadoc) *
 * @see android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent) */
@Override
public void onSensorChanged(SensorEvent event) {
// Récupérer les valeurs du capteur
```

```
float x, y, z;
if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
// Log.d(LOG_TAG, "TYPE_ACCELEROMETER");
// En fonction de l'orientation de l'appareil, on corrige les valeurs x et y du capteur
switch (mDisplay.getRotation()) {
case Surface.ROTATION_0:
        x = event.values[0];
        y = event.values[1];
        break;
case Surface.ROTATION_90:
        x = -event.values[1];
        y = event.values[0];
        break;
case Surface.ROTATION_180:
        x = -event.values[0];
        y = -event.values[1];
        break;
case Surface.ROTATION_270:
        x = event.values[1];
        y = -event.values[0];
        break;
}
// la valeur de z
z = event.values[2];
// faire quelque chose, par exemple un Log :
Log.d(LOG_TAG, "Sensor's values ("+x+","+y+","+z+") and maxRange : "+maxRange);
        }
}
```

Vous remarquerez qu'il n'y a pas deux états (landscape ou portrait) mais quatre, landscape, landscape inversé, portrait, portrait inversé. L'exemple s'appuie sur l'utilisation de l'accéléromètre qui est l'exemple typique.

3 Le capteur de lumière

Ce capteur est extrêmement facile d'utilisation, mais sa sensibilité dépend du fabricant. Typiquement le mien ne renvoie que des puissances de 10 (10, 100, 1000, et ainsi de suite). Ce capteur permet de savoir quelle est l'intensité lumineuse détectée par votre téléphone (l'unité est le Lux).

Ainsi pour écouter les changements de valeur de ce capteur, il vous faut :

- le déclarer dans votre fichier AndroidManifest le capteur de lumière ;
- faire étendre votre activité (ou la classe qui écoute le capteur) de SensorListener ;
- déclarer et instancier un objet Sensor de type light ;
- s'enregistrer/se désenregistrer en tant qu'écouteur de ce capteur ;
- faire quelque chose lors d'un changement de valeur.

Ce qui donne :

```
public class SensorLightTutoActivity extends Activity implements SensorEventListener {
/*********************************************************************/
/** Attribut du capteur*************************************************/
/*********************************************************************/
/** * Valeur courante de la lumière */
float l;
```

```
…
/** * Le sensor manager */
SensorManager sensorManager;
/** * le capteur de lumière */
Sensor light;
/*****************************************************************/
/** Gestion du cycle de vie ****************************************/
/*****************************************************************/
/** Appelé à la création de l'activité. */
@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	// construire l'IHM
	setContentView(R.layout.main);
	//
	// Instancier le SensorManager
	sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
	// Instancier le capteur de lumière
	light = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
	…
}
/* * (non-Javadoc) *  * @see android.app.Activity#onPause() */
@Override
protected void onPause() {
	// désenregistrer notre écoute du capteur
	sensorManager.unregisterListener(this, light);
	super.onPause();
}
/* * (non-Javadoc) *  * @see android.app.Activity#onResume() */
@Override
protected void onResume() {
	/* * enregistrer notre écoute du capteur*/
	sensorManager.registerListener(this, light, SensorManager.SENSOR_DELAY_GAME);
	super.onResume();
}
/*******************************************************************/
/** SensorEventListener ********************************************/
/*******************************************************************/
/** (non-Javadoc)* * @see
android.hardware.SensorEventListener#onAccuracyChanged(android.hardware.Sensor, int) */
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
	// Faîtes quelque chose ou pas…
}
/** (non-Javadoc)* * @see
android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent) */
@Override
public void onSensorChanged(SensorEvent event) {
	// Mettre à jour uniquement dans le cas de notre capteur
	if (event.sensor.getType() == Sensor.TYPE_LIGHT) {
		// La valeur de la lumière
		l = event.values[0];
		// faire autre chose…
	}
}
}
```

Le tutoriel se trouve ici : Page des tutoriels des capteurs sur Android2ee

4 Le capteur de proximité

Ce capteur est extrêmement facile d'utilisation, mais sa sensibilité dépend du fabricant. Typiquement le mien ne renvoie que soit 0, soit 5 mètres. Pour comprendre pourquoi il faudrait le nommer « capteur qui détecte la présence du corps humain au niveau de l'écouteur de l'appareil ». S'il détecte une présence il renvoie 0 sinon il renvoie 5. Comment ça marche ? Sûrement comme la détection de nos doigts sur l'écran, mais je n'en suis pas certain.

Ainsi pour écouter les changements de valeur de ce capteur, il vous faut (comme d'habitude) :

- déclarer dans votre fichier AndroidManifest le capteur de proximité ;
- faire étendre votre activité (ou la classe qui écoute le capteur) de SensorListener ;
- déclarer et instancier un objet Sensor de type proximity ;
- s'enregistrer/se désenregistrer en tant qu'écouteur de ce capteur ;
- faire quelque chose lors d'un changement de valeur.

Bon, le principe est identique à celui de la lumière, ce qui donne :

```
public class SensorProximityTutoActivity extends Activity implements SensorEventListener {
/*********************************************************************/
/** Attribut du capteur*************************************************/
/*********************************************************************/
/** * Valeur courante de la proximité */
float p;
…
/** * Le sensor manager */
SensorManager sensorManager;
/** * le capteur de proximité */
Sensor proximity;
/*****************************************************************/
/** Gestion du cycle de vie ***************************************/
/*****************************************************************/
/** Appelé à la création de l'activité. */
@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	// construire l'IHM
	setContentView(R.layout.main);
	//
	// Instancier le SensorManager
	sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
	// Instancier le capteur de lumière
	proximity = sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
	…
}
/* * (non-Javadoc) *  * @see android.app.Activity#onPause() */
@Override
protected void onPause() {
	// désenregistrer notre écoute du capteur
	sensorManager.unregisterListener(this, proximity);
	// and don't forget to pause the thread that redraw the xyAccelerationView
	super.onPause();
}
/* * (non-Javadoc) *  * @see android.app.Activity#onResume() */
@Override
protected void onResume() {
	/* * enregistrer notre écoute du capteur*/
```

```
		sensorManager.registerListener(this, proximity, SensorManager.SENSOR_DELAY_GAME);
		super.onResume();
}
/*****************************************************************/
/** SensorEventListener ***********************************************/
/*****************************************************************/
/** (non-Javadoc)* * @see
android.hardware.SensorEventListener#onAccuracyChanged(android.hardware.Sensor, int) */
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
		// Faites  quelque chose ou pas…
}
/** (non-Javadoc)* * @see
android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent) */
@Override
public void onSensorChanged(SensorEvent event) {
		// Mettre à jour uniquement dans le cas de notre capteur
		if (event.sensor.getType() == Sensor.TYPE_PROXIMITY) {
			// La valeur de la lumière
			p = event.values[0];
			// faire autre chose…
		}
}
}
```

Le tutoriel se trouve ici : Page des tutoriels des capteurs sur Android2ee

5 L'accéléromètre, l'accéléromètre linéaire et la gravité

Ce capteur est celui qui me plait le plus, il est capable de donner le champ de forces qui s'appliquent sur l'appareil (je peux détecter la présence des Jedis… ok, je sors). À partir de ce capteur Android en dérive deux de plus, le Linear Acceleration et la Gravity.

L'accéléromètre fournit le vecteur de force (ou d'accélération, c'est la même chose) tridimensionnel (x,y,z). À partir de ce vecteur le système déduit la composante gravitationnelle (toujours un vecteur tridimensionnel). Celle-ci est celle renvoyée par le capteur Gravity. Le capteur Linear Acceleration déduit du champ de force cette composante fournissant un vecteur (tridimensionnel) épuré de la gravité. Cela explique pourquoi les capteurs Gravity et Linear Acceleration n'ont pas de numéro de série et sont fournis par Google.

Il est important de comprendre le repère utilisé par ce capteur :

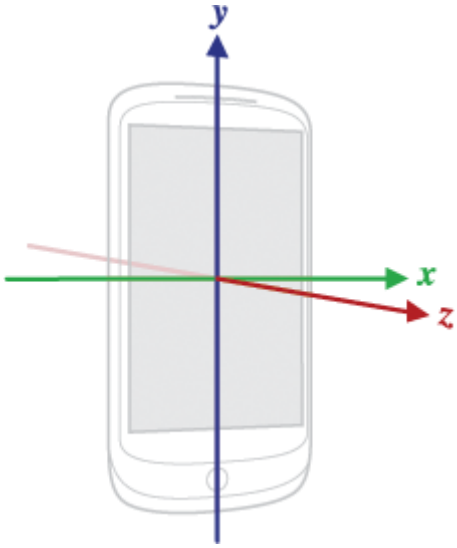


Figure 2_: provenant de google (http://developer.android.com/images/axis_device.png)

Ainsi le vecteur d'accélération (x,y,z) est toujours donné en fonction du repère de l'écran. Si vous bougez votre écran ces valeurs vont changer. L'axe des X est l'axe horizontal de l'écran, l'axe des Y est le vertical et l'axe des z est orthogonal à l'écran.

Au repos, la gravité est dans le repère terrestre (0,0,$\sqrt{x^2+y^2+z^2}$), dans le repère de l'appareil vous aurez donc des valeurs qui dépendent de l'inclinaison de l'appareil dans l'espace.

De ce capteur vous pouvez effectuer une multitude de chose (niveau à bulle, jeux contrôlés par l'accélération…)

Dans le tutoriel, j'ai mis en place deux balles qui sont manipulées par le champ de forces ; l'une considère ce champ comme son vecteur d'accélération, l'autre comme son vecteur vitesse. Si vous souhaitez faire un jeu dont les trajectoires des objets sont contrôlées par le champ de force, le mieux est d'inventer vos propres formules.

Le code permet d'écouter l'un des trois capteurs. Les trois capteurs sont enregistrés et un entier sensorType permet d'écouter spécifiquement l'un de ces capteurs.

Dans le fichier de l'activité (j'ai épuré le code, un copier-coller ne marchera pas, utiliser le tutorial plutôt) :

```
public class SensorAccelerationTutoActivity extends Activity implements SensorEventListener {
/…/
/*** valeur courante de l'accéléromètre*/
float x, y, z;
/*****************************************************************************************/
/** Sensors and co ***********************************************************************/
/*****************************************************************************************/
/** * Celui qui connait l'orientation de l'appareil */
private Display mDisplay;
/** * Le sensor manager */
SensorManager sensorManager;
/** * l'accéléromètre */
Sensor accelerometer;
/** * la gravité */
```

```
Sensor gravity;
/** * l'accéléromètre linéaire */
Sensor linearAcc;
/*****************************************************************/
/** Sensors Type Constant ****************************************/
/*****************************************************************/
/** * Le capteur sélectionné */
private int sensorType;
/** * l'accéléromètre */
private static final int ACCELE = 0;
/** * la Gravité */
private static final int Gravity = 1;
/** * l'accéléromètre linéaire */
private static final int LINEAR_ACCELE = 2;
/**********************************/
/** Gestion du cycle de vie ***************/
/**********************************/
/** Création de l'activité */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
// Construire l'IHM
setContentView(R.layout.main);
// Gérer les capteurs
// Instancier le SensorManager
sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
// Instancier l'accéléromètre
accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
// Instancier la gravité
gravity = sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY);
// Instancier l'accélération linéaire
linearAcc = sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
// Et enfin instancier le display qui connaît l'orientation de l'appareil
mDisplay = ((WindowManager) getSystemService(WINDOW_SERVICE)).getDefaultDisplay();
}

/* * (non-Javadoc) *  * @see android.app.Activity#onPause() */
@Override
protected void onPause() {
// désenregistrer tous le monde
sensorManager.unregisterListener(this, accelerometer);
sensorManager.unregisterListener(this, gravity);
sensorManager.unregisterListener(this, linearAcc);
super.onPause();
}
/* * (non-Javadoc) *  * @see android.app.Activity#onResume() */
@Override
protected void onResume() {
/* Ce qu'en dit Google :
* «  Ce n'est pas nécessaire d'avoir les évènements des capteurs à un rythme trop rapide.
 * En utilisant un rythme moins rapide (SENSOR_DELAY_UI), nous obtenons un filtre
 * automatique de bas-niveau qui "extrait" la gravité  de l'accélération.
 * Un autre bénéfice étant que l'on utilise moins d'énergie et de CPU. »
 */
sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_UI);
sensorManager.registerListener(this, gravity, SensorManager.SENSOR_DELAY_UI);
sensorManager.registerListener(this, linearAcc, SensorManager.SENSOR_DELAY_UI);
super.onResume();
}
/**********************************/
```

```
/** SensorEventListener *************/
/**********************************/
/* * (non-Javadoc) *  * @see
android.hardware.SensorEventListener#onAccuracyChanged(android.hardware.Sensor, int) */
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
		// Nothing to do
}
/* * (non-Javadoc) *  * @see
android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent) */
@Override
public void onSensorChanged(SensorEvent event) {
		// update only when your are in the right case:
if (((event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) && (sensorType == ACCELE))
|| ((event.sensor.getType() == Sensor.TYPE_GRAVITY) && (sensorType == Gravity))
 || ((event.sensor.getType() == Sensor.TYPE_LINEAR_ACCELERATION) && (sensorType ==
LINEAR_ACCELE))) {
// Corriger les valeurs x et y en fonction de l'orientation de l'appareil
switch (mDisplay.getRotation()) {
case Surface.ROTATION_0:
		x = event.values[0];
		y = event.values[1];
		break;
case Surface.ROTATION_90:
		x = -event.values[1];
		y = event.values[0];
		break;
case Surface.ROTATION_180:
		x = -event.values[0];
		y = -event.values[1];
		break;
case Surface.ROTATION_270:
		x = event.values[1];
		y = -event.values[0];
		break;
}
// la valeur z
z = event.values[2];
// faire quelque chose
}
}

}
```

Le tutoriel associé vous montre comment :

- mettre en place un spinner permettant de choisir quel type de capteur l'on souhaite écouter ;
- mettre en place trois progressBars présentant les valeurs x,y,z du vecteur de force ;
- afficher un point qui représente ce vecteur (et surtout comment mettre une thread qui re-dessine régulièrement l'écran) ;
- et deux points qui bougent en fonction des valeurs du capteur.

Il se trouve ici : Page des tutoriels des capteurs sur Android2ee

6 Le capteur électromagnétique

Le capteur électromagnétique permet de connaître les valeurs du vecteur électromagnétique qui s'applique à votre téléphone. Le référentiel est le même que celui utilisé par le capteur d'accélération.

Les valeurs sont en micro-tesla.

Au-delà du vecteur en lui-même, sa norme permet de savoir ce que l'on appelle la valeur électromagnétique.

Si vous placez votre appareil à proximité d'un aimant (par exemple des baffles) vous verrez les valeurs augmenter.

Mais place au code, ci-dessous la méthode onSensorChanged (les méthodes onCreate, onPause, onResume et onAccuracyChanged étant semblables je ne les répéterai plus) :

```
public void onSensorChanged(SensorEvent event) {
	// Lire les données quand elles correspondent à notre capteur:
	if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
		// Valeur du vecteur du champ magnétique (x,y,z)
		xMagnetic = event.values[0];
		yMagnetic = event.values[1];
		zMagnetic = event.values[2];
		// Valeur de la norme de ce vecteur
		magneticStrenght=Math.sqrt((double)
			 (xMagnetic*xMagnetic+
			yMagnetic*yMagnetic+
			zMagnetic*zMagnetic));
		// faire quelque chose, demander à mettre à jour l'IHM, par exemple :
		redraw();
	}

}
```

Le tutoriel se trouve ici :Page des tutoriels des capteurs sur Android2ee

7 Le capteur d'orientation

Comme son nom l'indique ce capteur donne l'orientation (le nord pour être grossier). Ce capteur est la boussole de votre appareil. Il y a trois notions à comprendre avec ce capteur :

- l'azimut : donne l'angle avec le Nord magnétique ;
- le pitch : donne l'angle autour de l'axe des x. En français cela se dit le tangage, mais comme cela embrouille plus qu'autre chose, je garde la nomenclature anglaise pour être en cohérence avec la SDK ;
- le roll : donne l'angle autour de l'axe des y. En français cela se dit le roulis, mais comme cela embrouille plus qu'autre chose, je garde la nomenclature anglaise pour être en cohérence avec la SDK.

L'azimut varie entre 0 et 360°, il représente l'angle avec le nord dans le sens des aiguilles d'une montre.

Le pitch varie entre -180 et 180, il représente l'inclinaison haut-bas de l'appareil selon l'axe Y (parallèle au sol, perpendiculaire au sol). On obtient les valeurs suivantes :

- la valeur 0, l'appareil est parallèle au sol, face vers le ciel ;
- la valeur +/- 180 est l'appareil parallèle au sol, face vers le sol ;
- la valeur 90 est l'appareil perpendiculaire au sol face, tête vers le bas ;
- la valeur -90 est l'appareil perpendiculaire au sol face, tête vers le haut.

Enfin le roll varie entre -90 et 90, il représente l'inclinaison droite-gauche de l'appareil selon l'axe des X (si l'appareil penche à gauche ou à droite). On obtient les valeurs suivantes (quand sa face est vers le haut) :

- la valeur 0, l'appareil ne penche pas ;
- la valeur 90 est l'appareil penche à gauche ;
- la valeur -90 est l'appareil penche à droite.

Il y a deux façons distinctes pour obtenir cette orientation, soit en écoutant directement le capteur d'orientation (ce qui n'a pas l'air d'être la bonne pratique), soit en utilisant le champ magnétique et le champ de force.

7.1 Écoute directe du capteur d'orientation

Il faut utiliser le capteur d'orientation :

```
// Le SensorManager
sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
// Le capteur d'orientation
orientation = sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);

...

/*
* (non-Javadoc)
*
* @see android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent)
*/
@Override
public void onSensorChanged(SensorEvent event) {
// update only when your are in the right case:
if (event.sensor.getType() == Sensor.TYPE_ORIENTATION) {
        //l' azimuth
        x = event.values[0];
        //le pitch
        y = event.values[1];
        //le roll
        z = event.values[2];
        }
}
```

Cette méthode n'est pas conseillée et d'après Google n'est là que pour des raisons « legacy », c'est-à-dire provenant de l'histoire de la plateforme.

7.2 Récupération de l'orientation avec le champ magnétique et l'accélération

Cette méthode est celle conseillée par Google. Pour la mettre en place au lieu d'écouter le capteur d'orientation, il faut écouter les capteurs d'accélération et d'électromagnétisme.

```
// Attribut de la classe pour calculer l'orientation
```

```
float[] acceleromterVector=new float[3];
float[] magneticVector=new float[3];
float[] resultMatrix=new float[9];
float[] values=new float[3];



// Instantiate the magnetic sensor and its max range
magnetic = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
// Instantiate the accelerometer
accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

...

/*
 * (non-Javadoc)
 *
 * @see android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent)
 */
@Override
public void onSensorChanged(SensorEvent event) {
// Mettre à jour la valeur de l'accéléromètre et du champ magnétique
if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        acceleromterVector=event.values;
} else if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
        magneticVector=event.values;
}
// Demander au sensorManager la matric de Rotation (resultMatric)
SensorManager.getRotationMatrix(resultMatrix, null, acceleromterVector, magneticVector);
// Demander au SensorManager le vecteur d'orientation associé (values)
SensorManager.getOrientation(resultMatrix, values);
// l'azimuth
x =(float) Math.toDegrees(values[0]);
// le pitch
y = (float) Math.toDegrees(values[1]);
// le roll
z = (float) Math.toDegrees(values[2]);
}
```

En utilisant cette méthode, il faut être attentif, en effet le pitch et le roll ne sont plus les mêmes.

Le pitch varie entre -90 et 90, il représente l'inclinaison haut-bas de l'appareil selon l'axe Y (parallèle au sol, perpendiculaire au sol). On obtient les valeurs suivantes :

- la valeur 0, l'appareil est parallèle au sol, face vers le ciel ou l'appareil est parallèle au sol ; face vers le sol ;
- la valeur 90 l'appareil est perpendiculaire au sol face, tête vers le bas ;
- la valeur -90 l'appareil est perpendiculaire au sol face, tête vers le haut.

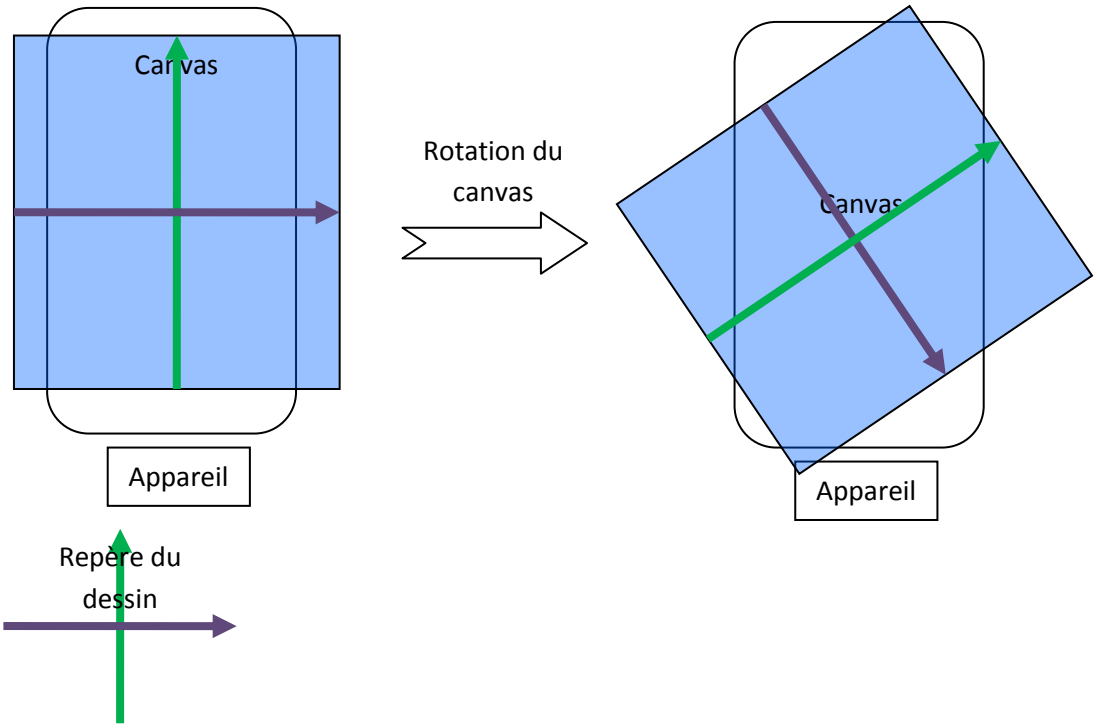
Enfin le roll varie entre -180 et 180, il représente l'inclinaison droite-gauche de l'appareil selon l'axe des X (si l'appareil penche à gauche ou à droite). On obtient les valeurs suivantes (quand sa face est vers le haut) :

- la valeur 0, l'appareil ne penche pas, il est face au ciel ;
- la valeur 90 est l'appareil penche à gauche ;
- la valeur -90 est l'appareil penche à droite ;

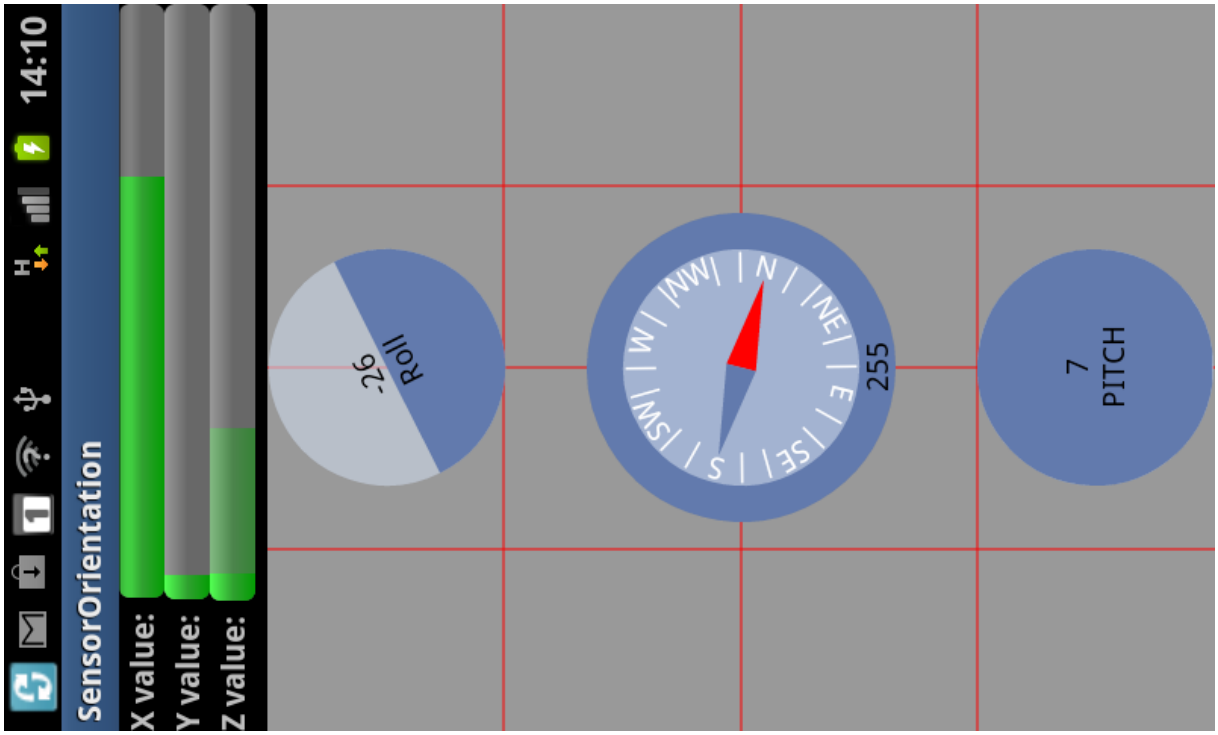
- La valeur 180, l'appareil ne penche pas, il est face contre sol.

7.3 Digression graphique

Ce capteur amène naturellement la question de dessiner ces valeurs. Pour cela quelques notions de dessins 2D sont nécessaires. Pour comprendre la philosophie de tels dessins, il faut comprendre les transformations de l'objet Canvas. En effet, on commence par tourner le Canvas (on lui applique la rotation qui correspond à la valeur que l'on affiche), ensuite on effectue le dessin (cercle, demi-cercle, flèche) dans ce nouveau référentiel. Ce qui nous permet de faire un dessin simple (toujours dessiner verticalement) qui sera tourné automatiquement par la rotation que nous appliquons au Canvas. L'idée étant de faire toujours le même dessin puis de tourner la feuille sur laquelle on a fait ce dessin, pour qu'il corresponde à notre souhait. Et ça c'est malin comme idée.



Le dessin à effectuer est le suivant :



Ainsi pour dessiner le roll, le code suivant est le bon :

```
// Sauver la configuration du canvas (orientation, translation)
canvas.save();
// trouver la valeur du roll à afficher
float roll = activity.z;
// faire tourner le canvas, de manière à ce que la description de notre dessin s'effectue dans un repère
//orthonormé normal (y vers le haut) mais à ce que son affichage soit le y vers l'angle du roll
canvas.rotate(roll, 0, 0);
// Définir le rectangle contenant le cercle que l'on va dessiner
RectF pitchOval = new RectF(0,0, lenght, lenght);
// Dessiner le cercle de fond
paint.setColor(backgroundCircleColor);
canvas.drawArc(pitchOval, 0, 360, false, paint);
// dessiner le demi-cercle qui représente le roll (et là grâce à la rotation du canvas, on le dessine comme étant le
demi-cercle inférieur)
paint.setColor(circleColor);
canvas.drawArc(pitchOval, 0, 180, false, paint);
// Pour finir, revenir à la configuration du canvas initiale 'il ne prend plus en compte notre rotation dans sa future
//utilisation)
canvas.restore();
```

Pour le pitch :

```
// Sauver la configuration initiale du canvas
canvas.save();
// Trouver le pitch à afficher
float pitch = activity.y;
// Mettre le centre au milieu de l'écran, dans son quart bas
canvas.translate(cx, cy + 3 * lenght / 2);
// Définir le rectangle contenant le cercle à dessiner
RectF pitchOval = new RectF(-lenght/2, -lenght/2, lenght/2, lenght/2);
// Dessiner le cercle du fond
paint.setColor(backgroundCircleColor);
canvas.drawCircle(0, 0, lenght / 2, paint);
```

```
// Dessiner le pitch
paint.setColor(circleColor);
if(pitch<=0) {
	canvas.drawArc(pitchOval, -90-pitch, 360+2*pitch, false, paint);
}else {
	canvas.drawArc(pitchOval, 90+pitch, 360-2*pitch, false, paint);
}
// restaurer l'état initial du canvas
canvas.restore();
```

Pour l'azimut, ou comment dessiner une boussole :

```
canvas.save();
//Définir la flèche nord
Path northPath = new Path();
Path southPath = new Path();
northPath.moveTo(0, -60);
northPath.lineTo(-10, 0);
northPath.lineTo(10,0);
northPath.close();
// Définir la flèche sud
southPath.moveTo(-10,0);
southPath.lineTo(0,60);
southPath.lineTo(10,0);
southPath.close();
float fontHeight = paint.getFontMetrics().ascent  +  paint.getFontMetrics().descent;
// Retrouver l'azimut à afficher
float azimut = -activity.x;
//centrer le compas au milieu de l'écran
canvas.translate(cx, cy);
// Effectuer une rotation de canvas, de sorte que la flèche indique le nord
// quand on la dessine verticalement
canvas.rotate(azimut);
// dessiner le cercle de la boussole
paint.setColor(backgroundCircleColor);
canvas.drawCircle(0, 0, lenght / 2, paint);

// Dessiner la graduation du compas
paint.setColor(Color.WHITE);
float hText = - lenght/2 - fontHeight+3;
// Tous les 15° faire une graduation
int step = 15;
for (int degree = 0; degree < 360; degree = degree + step) {
	// Si ce n'est pas un point cardinal, dessiner une graduation
	if ((degree % 90) != 0) {
		canvas.drawText("|", 0, hText, paint);
	}
	canvas.rotate(-step);
}

// Dessiner les points cardinaux
canvas.drawText("N", 0, hText, paint);
canvas.rotate(-90);
canvas.drawText("W", 0, hText, paint);
canvas.rotate(-90);
canvas.drawText("S", 0, hText, paint);
canvas.rotate(-90);
canvas.drawText("E", 0, hText, paint);
canvas.rotate(-90);
```

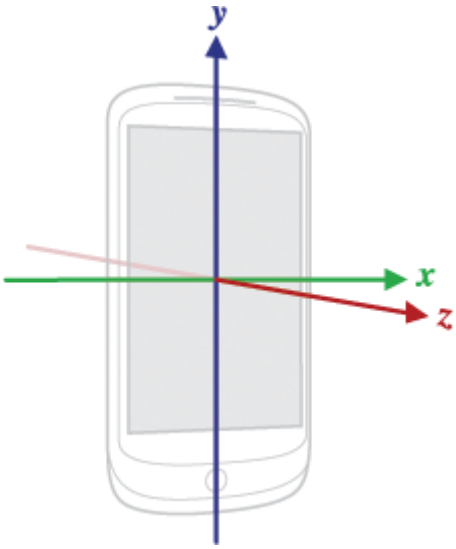
```
// Dessiner les flèches
paint.setStyle(Paint.Style.FILL);
paint.setColor(Color.RED);
canvas.drawPath(northPath, paint);
paint.setColor(circleColor);
canvas.drawPath(southPath, paint);
// Restaurer l'état initial du canvas
canvas.restore();
```

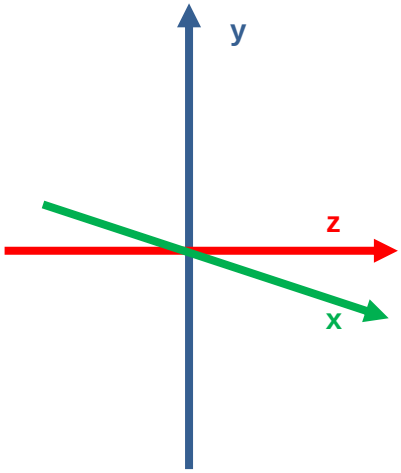
Les tutoriels se trouvent ici : Page des tutoriels des capteurs sur Android2ee

Vous trouverez celui qui utilise le capteur d'orientation et celui qui utilise les capteurs d'accélération et de champ magnétique.

7.4 La méthode remapCoordinateSystem

Cette méthode permet de réorienter l'appareil. En d'autres termes, au lieu d'être dans le système de coordonnées usuel, il est possible de définir où se trouve l'axe des X, des Y et des Z.





vers celui-ci :

Ce changement correspond à dire à l'appareil que la position de repos (usuellement, à plat au sol, écran vers le haut) correspond maintenant à : sur le coté droit. Inverser les axes y et z signifiant que la position de repos est verticale, écran face à l'utilisateur.

Pour cela, il suffit de modifier sa méthode onSensorChanged ainsi :

```
/* * (non-Javadoc) *
 * @see android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent) */
@Override
public void onSensorChanged(SensorEvent event) {
	//Récupérer les valeurs de l'accélération et du champ magnétique:
	if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
		acceleromterVector=event.values;
	} else if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
		magneticVector=event.values;
	}
	//retrouver la matrice de rotation
	SensorManager.getRotationMatrix(resultMatrix, null, acceleromterVector, magneticVector);
	//Effectuer le changement de coordonnées en utilisant la méthode remap:
	float[] outR=new float[9];
```

```
		SensorManager.remapCoordinateSystem(resultMatrix, SensorManager.AXIS_Y,
			SensorManager.AXIS_Z, outR);
		//retrouver la matrice d'orientation en lui passant outR et non pas resultMatrix
		SensorManager.getOrientation(outR, values);
		//Récuperer les différentes valeurs maintenant corrigées dans ce nouveau système.
		// l'azimut
		x =(float) Math.toDegrees(values[0]);
		// le pitch
		y = (float) Math.toDegrees(values[1]);
		// le roll
		z = (float) Math.toDegrees(values[2]);
}
```

8 Le gyroscope

Le gyroscope est un capteur qui calcule la vitesse angulaire de votre téléphone. Pour cela rien de bien magique. Il suffit comme d'habitude avec les capteurs de le créer, s'enregistrer/se désenregistrer en tant qu'écouteur et d'implémenter la méthode onSensorChanged :

```
/* * (non-Javadoc) *
 * @see android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent) */
@Override
public void onSensorChanged(SensorEvent event) {
		// Écouter le changement du gyroscope:
		if (event.sensor.getType() == Sensor.TYPE_GYROSCOPE) {
			// La vitesse angulaire autour de chaque axe
			xGyroscope = event.values[0];
			yGyroscope = event.values[1];
			zGyroscope = event.values[2];
			//Vous pouvez faire quelque chose de ça, mais quoi ?o?
		//see http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_GYROSCOPE
			// demander le rafraichissement de l'écran.
			redraw();
		}
}
```

Le tutoriel se trouve ici : Page des tutoriels des capteurs sur Android2ee

9 Le capteur vecteur d'orientation

Ce capteur représente le vecteur de rotation de l'appareil, de la même manière que l'orientation. Ce n'est pas un capteur physique mais plus un capteur interpolé (comme Gravity). Pour finir ce capteur peut renvoyer quatre valeurs au lieu de trois.

Un exemple d'utilisation de ce capteur se trouve dans les exemples Google (http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/os/RotationVectorDemo.html) et montre comment utiliser ce vecteur pour faire tourner un cube en accord avec le mouvement de l'appareil.

Ce qu'en dit Google :

« Le vecteur de rotation représente l'orientation de l'appareil comme une combinaison d'un angle et d'un axe, dans laquelle l'appareil a pivoté d'un angle θ autour d'un axe <x, y, z>.

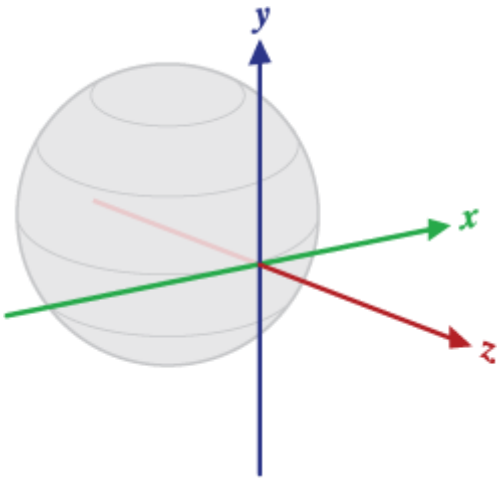
Les trois éléments du vecteur de rotation sont <x*sin(θ/2), y*sin(θ/2), z*sin(θ/2)>, tels que l'ampleur du vecteur de rotation est égale à sin (θ / 2) et sa direction est égale à la direction de l'axe de rotation.

Les trois éléments du vecteur de rotation sont égaux aux trois dernières composantes d'un quaternion unitaire <cos (θ / 2), x * sin (θ / 2), y * sin (θ / 2), z * sin (θ / 2)>.

Les éléments du vecteur de rotation sont sans unité. Les axes x, y et z sont définis de la même manière que le capteur d'accélération.

Le système de coordonnées de référence est défini comme une base orthonormée directe, où :

- X est défini comme étant le vecteur d'YZ produit (Il est tangentiel au sol à l'emplacement actuel de l'appareil et pointe à peu près vers l'est).
- Y est tangent à la terre à l'emplacement actuel de l'appareil et pointe vers le pôle nord magnétique.
- Z pointe vers le ciel et qui est perpendiculaire au sol.



values[0]: x*sin(θ/2)

values[1]: y*sin(θ/2)

values[2]: z*sin(θ/2)

values[3]: cos(θ/2) (optionnel: seulement si value.length = 4) »

Sinon, de la même manière que d'habitude, voilà la méthode onSensorChanged :

```
float[] mRotationMatrix;

/* * (non-Javadoc) *
 * @see android.hardware.SensorEventListener#onSensorChanged(android.hardware.SensorEvent) */
@Override
public void onSensorChanged(SensorEvent event) {
	// N'écouter que le capteur rotation vector:
	if (event.sensor.getType() == Sensor.TYPE_ROTATION_VECTOR) {
		// La valeur angulaire pour chaque axe
		xRotation = event.values[0];
		yRotation = event.values[1];
```

```
                zRotation = event.values[2];
                // La récupération du cosinus
                if (event.values.length == 4) {
                        cosRotation = event.values[3];
                }
                // see
                // http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_GYROSCOPE
                // redessiner
                redraw();
                //Pour l'uitliser avec en tant que matrice de rotation:
                //Convertir le vecteur de rotation en une matrice a 4x4 matrix. the matrix
                //Cette matrice est interprétée par OpenGl comme étant l'inverse
                //du vecteur de rotation.
                SensorManager.getRotationMatrixFromVector(mRotationMatrix, event.values);
        }
}
```

Le tutoriel se trouve ici : Page des tutoriels des capteurs sur Android2ee

10 Conclusion

Les capteurs nous permettent d'avoir une interaction plus grande avec l'environnement, j'espère que maintenant ils n'ont plus de secret pour vous.

11 Récupération des tutoriels

Cet article est associé à un tutoriel que vous trouverez en libre téléchargement sur le site Android2ee. À l'adresse suivante : Page des tutoriels des capteurs sur Android2ee

Liste des tutoriels disponibles :

Tutorial
SensorAccelerationTuto
SensorGyroscopeTuto
SensorLightTuto
SensorMagneticFieldTuto
SensorOrientationAMTuto
SensorOrientationTuto
SensorProximityTuto
SensorRotationVectorTuto
SensorList

12 Contact

Mathias Séguy

mathias.seguy.it@gmail.com

Auteur Android2ee.com

Docteur en Mathématiques Fondamentales

Directeur Technique & Avant-vente

ST Informatique *Services*

Expert Technique de l'Agence Nationale de la Recherche

Rédacteur sur Developpez.com

Android 2EE la programmation sous Android.

13 Android2ee vous propose des formations entreprise Android

Android2EE vous propose des formations Android intra-entreprises adaptées à vos besoins.

- Formation Initiale : Devenir autonome (3j).
- Formation Approfondissement (2j).
- Formation Spécificité Tablette (2j).
- Formation Multimédia (2j).
- Formation Applications complexes (2j).
- Formation chef de projet - responsable technique (1j).
- Mais aussi la possibilité d'effectuer des formations d'approfondissement sur mesure (2-3j).

Détails des formations.
Tarifs et considérations contractuelles.
Nous contacter :
mathias.seguy.it@gmail.com

14 Android2ee vous présente l'Ebook de programmation Android

Le nouveau système d'exploitation de Google pour les téléphones portables et les nouvelles tablettes est là. Sa réputation est solide, il envahit le monde de la téléphonie, il est ouvert et offre des outils de développement Java au monde des programmeurs. Il ouvre les portes du développement mobile à tous les développeurs objets avec un coût minime pour la montée en compétence. Une seule question se pose :

êtes-vous prêts ?

L'objectif de ces livres est très clair : vous permettre en un temps record d'être autonome en programmation Android. Si vous êtes un programmeur Java (débutant ou confirmé), le but est que vous soyez autonome en moins de dix jours. C'est cet objectif qui est à l'origine de ce livre, permettre aux collaborateurs de mon entreprise de monter en compétence sur cette technologie avec rapidité et efficience. Vous serez alors à même de concevoir une application, de l'implémenter, de la tester, de l'internationaliser et de la livrer à votre client.

Lancez-vous dans la programmation Android et faites-vous plaisir !

Vous serez aussi capable de connaître et comprendre quelles sont les considérations à avoir lorsque l'on a à charge une application Android en tant que professionnel de l'informatique. Quelle est la stratégie de tests à utiliser ? Comment signer son application ? Comment la déployer ? Comment

mettre en place la gestion du cycle de vie de l'application ? Comment implémenter l'intégration continue ?

Soyez efficient dans l'encadrement de vos projets Android d'entreprise.

15 Remerciements

J'adresse ici tous mes remerciements à Djug pour son aide et ses déploiements et à jacques_jean pour l'excellence de ses corrections orthographiques.

Je remercie les correcteurs techniques *** et *** pour la pertinence de leurs remarques.

Je remercie spécialement Monsieur Adam Daniel, DRH de développez.com, pour ses mails nocturnes, ses encouragements et son aide qui m'est précieuse.