ORIENTACION IMPORTANTE

Android: implementación de brújula: cálculo del azimut  
  
Con esta publicación quiero resumir algunas técnicas para una brújula simple desarrollada para Android. Mientras implementaba una aplicación de Android usando el valor de acimut para determinar la orientación direccional relativa a un objetivo, revisé múltiples implementaciones. No quiero presentarte una aplicación completa, sino mostrarte las diferentes implementaciones de los sensores y los algoritmos de recopilación de datos. Algunos de ellos se mejoran con algún tipo de filtro de paso bajo o promedio de cola. También debe tener en cuenta el hecho de que los valores de acimut dependen de la alineación del dispositivo. Por lo tanto, si el dispositivo no se mantiene plano (desviación de ∓45 °), debe usar remapCoordinateSystem () de manera útil para obtener resultados correctos.  
  
Brújula con sensores Android relevantes  
  
Dado que Gingerbread 2.3 Android es compatible con varios tipos de sensores nuevos. Utilizar este sensor en combinación con el otro puede permitir a los desarrolladores rastrear el movimiento tridimensional del dispositivo y el cambio de orientación con gran precisión y precisión. La siguiente tabla se basa en la guía para desarrolladores de Android.  
  
Sensores apreciables para calcular la orientación

|  |  |
| --- | --- |
| Sensor API Name | Sensor Type |
| TYPE\_ORIENTATION | Sensor Fusion – Deprecated |
| TYPE\_ACCELEROMETER | Hardware |
| TYPE\_MAGNETIC\_FIELD | Hardware |
| TYPE\_GYROSCOPE | Hardware |
| TYPE\_GRAVITY | Sensor Fusion / Hardware |
| TYPE\_ROTATION\_VECTOR | Sensor Fusion / Hardware |

Recopila datos de orientación usando sensores de bajo nivel  
  
Calcular los datos de orientación utilizando el sensor de hardware es sencillo. Registre los sensores TYPE\_ACCELEROMETER y TYPE\_MAGNETIC\_FIELD y obtenga los datos medidos.

private int mAzimuth = 0; // degree

private SensorManager mSensorManager = null;

private Sensor mAccelerometer;

private Sensor mMagnetometer;

boolean haveAccelerometer = false;

boolean haveMagnetometer = false;

private SensorEventListener mSensorEventListener = new SensorEventListener() {

float[] gData = new float[3]; // accelerometer

float[] mData = new float[3]; // magnetometer

float[] rMat = new float[9];

float[] iMat = new float[9];

float[] orientation = new float[3];

public void onAccuracyChanged( Sensor sensor, int accuracy ) {}

@Override

public void onSensorChanged( SensorEvent event ) {

float[] data;

switch ( event.sensor.getType() ) {

case Sensor.TYPE\_ACCELEROMETER:

gData = event.values.clone();

break;

case Sensor.TYPE\_MAGNETIC\_FIELD:

mData = event.values.clone();

break;

default: return;

}

if ( SensorManager.getRotationMatrix( rMat, iMat, gData, mData ) ) {

mAzimuth= (int) ( Math.toDegrees( SensorManager.getOrientation( rMat, orientation )[0] ) + 360 ) % 360;

}

}

}

@Override

protected void onCreate( Bundle savedInstanceState ) {

this.mAccelerometer = this.mSensorManager.getDefaultSensor( Sensor.TYPE\_ACCELEROMETER );

this.haveAccelerometer = this.mSensorManager.registerListener( mSensorEventListener, this.mAccelerometer, SensorManager.SENSOR\_DELAY\_GAME );

this.mMagnetometer = this.mSensorManager.getDefaultSensor( Sensor.TYPE\_MAGNETIC\_FIELD );

this.haveMagnetometer = this.mSensorManager.registerListener( mSensorEventListener, this.mMagnetometer, SensorManager.SENSOR\_DELAY\_GAME );

if ( haveAccelerometer && haveMagnetometer ) {

// ready to go

} else {

// unregister and stop

}

}

Mejore la implementación de bajo nivel  
  
Debido a que estos valores medidos provienen directamente de la fuente, incluyen algunas imprecisiones de medición importantes. Estas imprecisiones están haciendo que nuestro valor de azimut sea inestable e inutilizable, por lo que debemos mejorarlo utilizando algunas técnicas adecuadas. Por un lado, podemos (paso bajo) filtrar los valores del acelerómetro. Debe determinar un buen valor alfa para el filtrado, ya que causa un retraso grave si el valor es demasiado bajo. Por otro lado, podemos mantener un historial o más bien una cola de matrices de rotación y calcular el promedio.  
  
Además, podemos extender el filtro (y la cola) para que sean adaptables. Eso significa que, al permanecer en posición, el algoritmo usa un valor de filtro muy fuerte (tamaño alfa o de cola) para reducir el ruido. Si el algoritmo reconoce algún movimiento serio en una dirección, reducirá el valor del filtro alfa para reunir datos más rápidos.  
  
Otro punto de partida para mejorar es usar el giroscopio conjuntamente con el acelerómetro y el campo magnético. Debido a que el giroscopio tiene un muy buen tiempo de respuesta, puede mejorar dramáticamente la medición.  
Recopilar datos de orientación utilizando sensores de alto nivel (sensor-fusión)  
  
La técnica para crear sensores accionados por software, recibir datos obteniendo la entrada de varios sensores se denomina fusión de sensores. Los datos medidos de los sensores de fusión del sensor son en muchos aspectos mejores de lo que sería posible cuando estas fuentes se usaran individualmente. Mejor puede significar, más preciso, más completo o más confiable. Android viene con algunos sensores que no se basan en un solo sensor de hardware. Estos sensores incluidos están completamente implementados, son estables y están bien probados. Por lo general, es mejor utilizar sensores de fusión de sensores en lugar de implementar sus propios algoritmos de fusión de sensores.  
Mejore la implementación de bajo nivel utilizando el sensor TYPE\_GRAVITY  
  
En la sección anterior usamos sensores de hardware de bajo nivel para calcular el azimut. Con el campo magnético y el acelerómetro, podemos obtener la orientación. Desafortunadamente, si su teléfono sufre alguna aceleración lineal o si hay interferencias magnéticas, los valores medidos se vuelven ruidosos. Como se muestra arriba, el uso adicional de un giroscopio mejoraría drásticamente el tiempo de respuesta. También (paso bajo) filtramos los valores del acelerómetro porque no son muy exactos. Lo que intenta hacer este filtro de paso bajo es aislar el componente de gravedad del acelerómetro. En lugar de hacer esto, puede verificar si el sensor TYPE\_GRAVITY está disponible en el dispositivo y usarlo en lugar de TYPE\_ACCELEROMETER.

private int mAzimuth = 0; // degree

private SensorManager mSensorManager = null;

private Sensor mGravity;

private Sensor mAccelerometer;

private Sensor mMagnetometer;

boolean haveGravity = false;

boolean haveAccelerometer = false;

boolean haveMagnetometer = false;

private SensorEventListener mSensorEventListener = new SensorEventListener() {

float[] gData = new float[3]; // gravity or accelerometer

float[] mData = new float[3]; // magnetometer

float[] rMat = new float[9];

float[] iMat = new float[9];

float[] orientation = new float[3];

public void onAccuracyChanged( Sensor sensor, int accuracy ) {}

@Override

public void onSensorChanged( SensorEvent event ) {

float[] data;

switch ( event.sensor.getType() ) {

case Sensor.TYPE\_GRAVITY:

gData = event.values.clone();

break;

case Sensor.TYPE\_ACCELEROMETER:

gData = event.values.clone();

break;

case Sensor.TYPE\_MAGNETIC\_FIELD:

mData = event.values.clone();

break;

default: return;

}

if ( SensorManager.getRotationMatrix( rMat, iMat, gData, mData ) ) {

mAzimuth= (int) ( Math.toDegrees( SensorManager.getOrientation( rMat, orientation )[0] ) + 360 ) % 360;

}

}

}

@Override

protected void onCreate( Bundle savedInstanceState ) {

this.mGravity = this.mSensorManager.getDefaultSensor( Sensor.TYPE\_GRAVITY );

this.haveGravity = this.mSensorManager.registerListener( mSensorEventListener, this.mGravity, SensorManager.SENSOR\_DELAY\_GAME );

this.mAccelerometer = this.mSensorManager.getDefaultSensor( Sensor.TYPE\_ACCELEROMETER );

this.haveAccelerometer = this.mSensorManager.registerListener( mSensorEventListener, this.mAccelerometer, SensorManager.SENSOR\_DELAY\_GAME );

this.mMagnetometer = this.mSensorManager.getDefaultSensor( Sensor.TYPE\_MAGNETIC\_FIELD );

this.haveMagnetometer = this.mSensorManager.registerListener( mSensorEventListener, this.mMagnetometer, SensorManager.SENSOR\_DELAY\_GAME );

// if there is a gravity sensor we do not need the accelerometer

if( this.haveGravity )

this.mSensorManager.unregisterListener( this.mSensorEventListener, this.mAccelerometer );

if ( ( haveGravity || haveAccelerometer ) && haveMagnetometer ) {

// ready to go

} else {

// unregister and stop

}

}

Utilizando la fusión de sensor de alto nivel TYPE\_ROTATION\_VECTOR  
  
TYPE\_ROTATION\_VECTOR es el conector de todos los sensores para medir la información de orientación de los dispositivos. Los datos son estables y tenemos un gran tiempo de respuesta. Utiliza el acelerómetro, el giroscopio y el magnetómetro si están disponibles. Necesita orientarse inicialmente y luego eliminar la deriva que viene con el giroscopio a lo largo del tiempo. Se puede encontrar más información sobre el sensor de vector de rotación en la guía para desarrolladores de Android.

private int mAzimuth = 0; // degree

private SensorManager mSensorManager = null;

private SensorEventListener mSensorEventListener = new SensorEventListener() {

float[] orientation = new float[3];

float[] rMat = new float[9];

public void onAccuracyChanged( Sensor sensor, int accuracy ) {}

@Override

public void onSensorChanged( SensorEvent event ) {

if( event.sensor.getType() == Sensor.TYPE\_ROTATION\_VECTOR ){

// calculate th rotation matrix

SensorManager.getRotationMatrixFromVector( rMat, event.values );

// get the azimuth value (orientation[0]) in degree

mAzimuth = (int) ( Math.toDegrees( SensorManager.getOrientation( rMat, orientation )[0] ) + 360 ) % 360;

}

}

}

@Override

protected void onCreate( Bundle savedInstanceState ) {

}

Compatibilidad con versiones anteriores con TYPE\_ORIENTATION  
  
El sensor TYPE\_ORIENTATION está en desuso desde Android Froyo 1.5 (SDK 3) pero aún se puede usar hasta Android Gingerbread 2.3 (SDK 9). Es un método fácil de usar para admitir dispositivos con una versión de Android debajo de Gingerbread. Sin embargo, simplemente no hace más que usar TYPE\_ACCELEROMETER y TYPE\_MAGNETIC\_FIELD con algunos filtros como se muestra arriba. Tenga en cuenta que se devolverá un valor de grado.

private int mAzimuth = 0; // degree

private SensorManager mSensorManager = null;

private Sensor mOrientation;

private Sensor mHaveOrientation;

private SensorEventListener mSensorEventListener = new SensorEventListener() {

public void onAccuracyChanged( Sensor sensor, int accuracy ) {}

@Override

public void onSensorChanged( SensorEvent event ) {

if( event.sensor.getType() == Sensor.TYPE\_ORIENTATION ){

newAzimuth = ( event.values[0] + 360 ) % 360;

}

}

}

@Override

protected void onCreate( Bundle savedInstanceState ) {

// check android Version

if ( Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.GINGERBREAD ) {

// enhanced compass implementation for Android 2.3+ here

} else {

mOrientation = mSensorManager.getDefaultSensor( Sensor.TYPE\_ORIENTATION);

mHaveOrientation = mSensorManager.registerListener( mSensorEventListener, mOrientation, SensorManager.SENSOR\_DELAY\_NORMAL );

}

}