

Indoor Tracking App

GPS less tracking system

Lionel Cominelli

Université de la Réunion / Master 1 informatique

December 9 2021

Plan

- ▶ Problematic
- ▶ The App solution
- ▶ Zoom on inertial principle
- ▶ The implementation
- ▶ Perspectives

Problematic

Since the 2001, US government has open the Global Positioning System (GPS) military tracking system to the world.

-

People take possession of the technology and intensively use it.

-

Today, we cannot imagine living without GPS !

-

GPS works with satellites and outcomes your position by distance triangulation between you and the satellites.

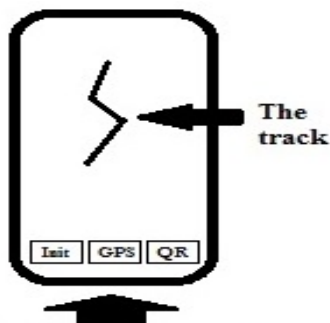
-

But, what's happen inside a building ????

-

GPS does not work.... to keep tracking indoor we must find another way.

The App Solution

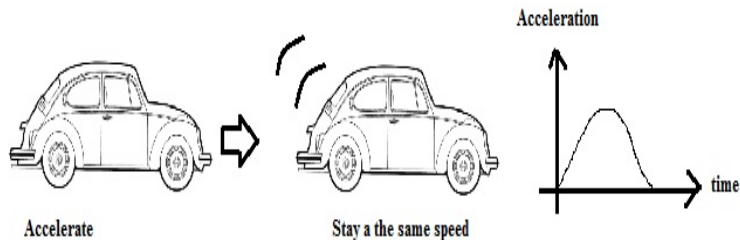


The initialisation buttons

- ▶ Init button : reset the displayed track, localize the position in the center of the screen
- ▶ GPS button : reset the displayed track and localize the position into a background map
- ▶ Button QR ; reset the displayed track and localize the position into a building background map
- ▶ The track : the path performed by the user

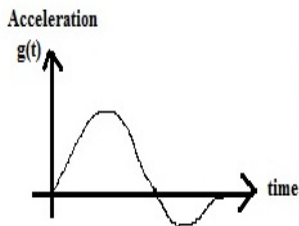
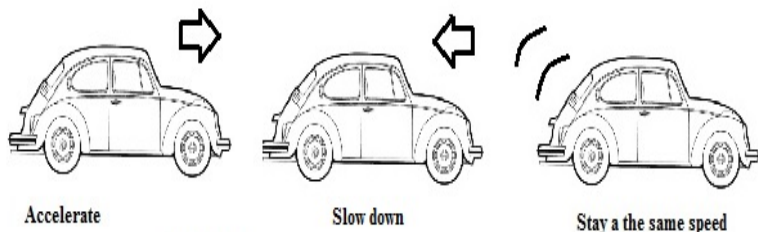
Zoom on inertial principle

- How the position is determined without GPS ?



Zoom on inertial principle

- How the position is determined without GPS ?

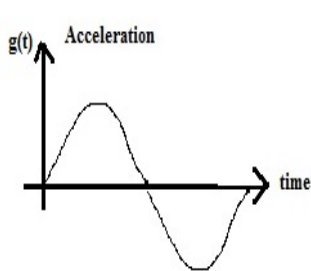


$$\text{vitesse} = \int g(t) dt > 0$$

$$T * \int g(t) dt = \text{distance}$$

Zoom on inertial principle

- How the position is determined without GPS ?

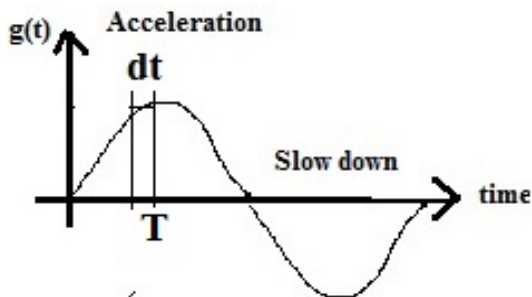


$$\text{vitesse} = \int g(t) dt = 0$$

$$T * \int g(t) dt = \text{distance} \neq 0$$

Zoom on inertial principle

- How the position is determined without GPS ?



$$d = dt * \left\{ g(t) * dt \right.$$

$$D(T) = \left\{ d = \left\{ dt \right\} g(t) * dt = T dt \left\{ g \right.$$

The implementation

```
public float[] computeDistancesBuffer(int mstime) {
```

```
    float[] Ig = new float[]{0,0,0};
```

```
    float dt;
```



```
    if (numberOfSamples != 0) {
```

```
        // save current buffer index
```

```
        int captureBufferIndex = currentBufferIndex;
```

```
        // save the number of samples
```

```
        int captureNumberOfSample = numberOfSamples;
```

```
        // switch the current buffer Index
```

```
        currentBufferIndex = (currentBufferIndex == 0) ? 1 : 0;
```

```
        // reset the number of samples
```

```
        numberOfSamples = 0;
```

```
        //dt = 0.1f;
```

```
        // make the integration
```

```
        for (int i = 0; i < captureNumberOfSample; i++) {
```

```
            Ig[0] += gxBuffer[captureBufferIndex][i] ;
```

```
            Ig[1] += gyBuffer[captureBufferIndex][i] ;
```

```
            //Ig[2] += gzBuffer[captureBufferIndex][i];
```

```
        }
```



dt →

```
        // compute the dt /1000 because mstime is in ms
```

```
        dt = ((float)mstime) / ((float)(1000 * captureNumberOfSample));
```

```
        // multiply the g integration by the T*dt to get distance
```

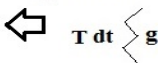
```
        for (int i = 0; i < 2; i++)
```

```
            Ig[i] = dt * Ig[i] * mstime / 1000;
```

```
        }
```

```
        return Ig;
```

```
    }
```



The implementation

```
public float[] computeDistancesBuffer(int mstime) {  
  
    float[] Ig = new float[]{0,0,0};  
    float dt;  
  
    if (numberOfSamples != 0) {  
        // save current buffer index  
        int captureBufferIndex = currentBufferIndex;  
        // save the number of samples  
        int captureNumberOfSample = numberOfSamples;  
        // switch the current buffer Index  
        currentBufferIndex = (currentBufferIndex == 0) ? 1 : 0;  
        // reset the number of samples  
        numberOfSamples = 0;  
  
        //dt = 0.1f;  
        // make the integration  
        for (int i = 0; i < captureNumberOfSample; i++) {  
            Ig[0] += gxBuffer[captureBufferIndex][i] ;  
            Ig[1] += gyBuffer[captureBufferIndex][i] ;  
            //Ig[2] += gzBuffer[captureBufferIndex][i];  
        }  
  
        // compute the dt /1000 because mstime is in ms  
        dt = ((float)mstime) / ((float)(1000 * captureNumberOfSample));  
  
        // multiply the g integration by the T*dt to get distance  
        for (int i = 0; i < 2; i++)  
            Ig[i] = dt * Ig[i] * mstime / 1000;  
    }  
    return Ig;  
}
```



Why gBuffers
are
float [2][i] ?

The implementation

```
@Override  
public void onSensorChanged(SensorEvent sensorEvent) {
```


← Runs every dt

```
    if (newborn == false) {  
        // compute g variation  
        //          3 for accelerometer  
        //          2 for linear accelerometer  
        for (int i = 0; i < 2; i++) {  
            gvar[i]=previousg[i]-sensorEvent.values[i];  
        }  
  
        // update the buffers  
        gxBuffer[currentBufferIndex][numberOfSamples]=  
            (Math.abs(gvar[0])>0.15)?sensorEvent.values[0]:0;  
        gyBuffer[currentBufferIndex][numberOfSamples]=  
            (Math.abs(gvar[1])>0.15)?sensorEvent.values[1]:0;  
        //gzBuffer[currentBufferIndex][numberOfSamples]=  
        //    (Math.abs(gvar[2])>0.15)?gvar[2]:0;  
  
        numberOfSamples=numberOfSamples+1;  
    } else {  
        newborn = false;  
    }  
  
    // overwrite the previousg  
    // do that instead of Buffer[i-1] because when buffer switching it  
    // is very difficult to get the previous values from the previous  
    // active buffer  
    for (int i = 0; i < 2; i++) {  
        previousg[i]=sensorEvent.values[i];  
    }  
}
```

← Double Buffering Technic

Perspectives

```
public float[] computeDistancesBuffer(int mstime) {  
  
    float[] Ig = new float[]{0,0,0};  
    float dt;  
  
    if (numberOfSamples != 0) {  
        // save current buffer index  
        int captureBufferIndex = currentBufferIndex;  
        // save the number of samples  
        int captureNumberOfSample = numberOfSamples;  
        // switch the current buffer Index  
        currentBufferIndex = (currentBufferIndex == 0) ? 1 : 0;  
        // reset the number of samples  
        numberOfSamples = 0;  
  
        //dt = 0.1f;  
        // make the integration  
        for (int i = 0; i < captureNumberOfSample; i++)  
            Ig[0] += gxBuffer[captureBufferIndex][i] ;  
            Ig[1] += gyBuffer[captureBufferIndex][i] ;  
            //Ig[2] += gzBuffer[captureBufferIndex][i];  
        }  
  
        // compute the dt /1000 because mstime is in ms  
        dt = ((float)mstime) / ((float)(1000 * captureNumberOfSample));  
  
        // multiply the g integration by the T*dt to get distance  
        for (int i = 0; i < 2; i++)  
            Ig[i] = dt * Ig[i] * mstime / 1000;  
    }  
    return Ig;  
}
```



Time greedy

Perspectives

```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {

    if (newborn == false) {
        // compute g variation
        //          3 for accelerometer
        //          2 for linear accelerometer
        for (int i = 0; i < 2; i++) {
            gvar[i]=previousg[i]-sensorEvent.values[i];
        }

        // update the buffers
        gxBuffer[currentBufferIndex][numberOfSamples]=
            (Math.abs(gvar[0])>0.15)?sensorEvent.values[0]:0;
        gyBuffer[currentBufferIndex][numberOfSamples]=
            (Math.abs(gvar[1])>0.15)?sensorEvent.values[1]:0;
        //gzBuffer[currentBufferIndex][numberOfSamples]=
        //    (Math.abs(gvar[2])>0.15)?gvar[2]:0;

        numberOfSamples=numberOfSamples+1;
    } else {
        newborn = false;
    }

    // overwrite the previousg
    // do that instead of Buffer[i-1] because when buffer switching it
    // is very difficult to get the previous values from the previous
    // active buffer
    for (int i = 0; i < 2; i++) {
        previousg[i]=sensorEvent.values[i];
    }
}
```

**Filtrering
implementation
to review
(digital FIR or IRR)**



Conclusion

- ▶ Accelerometer very sensitive
- ▶ Sensor driver implementation must be meticulous
- ▶ Embedded software is a real trade
- ▶ GUI Design too !

Thank you (Q&A ?)

