android CountDownTimer - additional milliseconds delay between ticks

1

android

From my observation the android CountDownTimer countDownInterval between ticks happens to be not accurate, the countDownInterval is regularly a few milliseconds longer than specified. The countDownInterval in my specific app is 1000ms, just counting down a certain amount of time with one second steps.

Due to this prolonged ticks I end up having less ticks then wanted when the the countdowntimer runs long enough which screws up the displayed countdown of the time (a 2 second step happens on the UI level when enough additional ms have summed up)

Looking into the source of CountDownTimer it seems possible to twist it so it corrects this unwanted inaccuracy yet I was wondering if there is already a better CountDownTimer available in the java/android world.

Thanks buddies for any pointer ...

share improve this question

dorjeduck
1,571 ●1 ●12 ●34

Asked
Oct 6 '12 at 17:47

3 Answers

order by votes

6

Rewrite

As you said, you also noticed that the next time in `onTick()` is calculated from the time the previous `onTick()` ran, which introduces a tiny error on *every* tick. I changed the CountDownTimer source code to call each `onTick()` at the specified intervals from the start time.

I build this upon the CountDownTimer framework, so cut & paste the source code into your project and give the class a unique name. (I called mine MoreAccurateTimer.) Now make a few changes:

1. Add a new class variable:

```
private long mNextTime;
```

2. Change `start()`:

```
public synchronized final MoreAccurateTimer start() {
    if (mMillisInFuture <= 0) {
        onFinish();
        return this;
    }

    mNextTime = SystemClock.uptimeMillis();
    mStopTimeInFuture = mNextTime + mMillisInFuture;

    mNextTime += mCountdownInterval;
    mHandler.sendMessageAtTime(mHandler.obtainMessage(MSG), mNextTime);
    return this;
}
```

3. Change the Handler's `handlerMessage()`:

```
        if (millisLeft <= 0) {
            onFinish();
        } else {
            onTick(millisLeft);

            // Calculate next tick by adding the countdown interval from the original start time
            // If user's onTick() took too long, skip the intervals that were already missed
            long currentTime = SystemClock.uptimeMillis();
            do {
                mNextTime += mCountdownInterval;
            } while (currentTime > mNextTime);

            // Make sure this interval doesn't exceed the stop time
            if(mNextTime < mStopTimeInFuture)
                sendMessageAtTime(obtainMessage(MSG), mNextTime);
            else
                sendMessageAtTime(obtainMessage(MSG), mStopTimeInFuture);
        }
    }
}
```

share improve this answer

Sam
47.3k●7●60●90

Answered
Oct 6 '12 at 18:06

Edited
Nov 25 '12 at 22:14

Interesting idea, will digest it and post here what i came up with for further discussion.. – **dorjeduck** Oct 6 '12 at 18:18

1 you mentioned other problems with CountDownTimer - was it the missing last tick due to ms delay? already had to adjust the class for that issue ... – **dorjeduck** Oct 6 '12 at 18:25

Yes. Assume it's counting down from three, the output was often: 3, 2, (*long pause*) 0... That was infuriating. – **Sam** Oct 6 '12 at 18:28

yep - it is build in if you look into the source code (no tick if the remaining time is even 1ms smaller than the given intervall) had a discussion here on stackoverflow before, some people say it makes sense yet for me it is poor design (within an overall very well done android ...) – **dorjeduck** Oct 6 '12 at

18:33

1 Thanks for your kind words. This version prevents that cumulative error (just like yours), but I also did away with the quirk that would sometimes skip the last tick. – **Sam** Oct 6 '12 at 22:09

show **8** more comments

2 So this is what I came up with. It is a small modification of the original CountDownTimer. What it adds is a variable mTickCounter which counts the amount of ticks called. This variable is used together with the new variable mStartTime to see how accurate we are with our ticks. Based on this input the delay to the next tick is adjusted ... It seems to do what I was looking for yet I am sure this can be improved upon.

Look for

```
// ************AccurateCountdownTimer***************
```

in the source code to find the modifications I have added to the original class.

```
package com.dorjeduck.xyz;

import android.os.Handler;
import android.os.Message;
import android.os.SystemClock;
import android.util.Log;

/**
 * Schedule a countdown until a time in the future, with regular notifications
 * on intervals along the way.
 *
 * Example of showing a 30 second countdown in a text field:
 *
 * <pre class="prettyprint">
 * new CountDownTimer(30000, 1000) {
 *
 *  public void onTick(long millisUntilFinished) {
 *      mTextField.setText(&quot;seconds remaining: &quot; + millisUntilFinished / 1000);
 *  }
 *
 *  public void onFinish() {
```

share improve this answer

dorjeduck
1,571 ●1 ●12 ●34

Answered
Oct 6 '12 at 19:35

(Upvote) Looks like we implemented the same logic, we must be on the right path. – **Sam** Oct 6 '12 at 19:45

I think both of our solution do yet yours seems more clean and straight forward, thanks a lot for sharing. – **dorjeduck** Oct 6 '12 at 20:02

add a comment

1

In most systems the timers are never perfectly accurate. The system only executes the timer when it doesn't have anything else to do. If the CPU is busy with a background process or a different thread then it won't get around to calling the timer till it has finished.

You might have better luck changing the interval to something smaller like 100ms and then only redrawing the screen if something has changed. With this approach the timer isn't directly causing anything to occur, it's just redrawing the screen periodically.

share improve this answer

Nathan Villaescusa
6,920 ● 17 ● 30

Answered
Oct 6 '12 at 17:54

Edited
Oct 6 '12 at 18:19

Thx Nathan, I understand that we cant expect 100% accuracy. Yet I am wondering why classes like the CountDownTimer have no accuracy adjustment logic build in, it seems straight forward to count the ticks, check what time has been gone since start and adjust the next intervall accordingly. Well I will just give it a try and implement it myself yet I just hope and expect that his has been done already, it seems so straight forward to me. – **dorjeduck** Oct 6 '12 at 17:59

I just dont like this making the intervall smaller solutions, it seems not satisfying at all as you will have shorter ticks and sometimes longer ticks (which are made of two ticks) in the UI user experience .. – **dorjeduck** Oct 6 '12 at 18:02

I see, you just want to know how long it has been since the last tick. You can always do long System.currentTimeMillis() after each tick and compare it to the time of the previous tick. – **Nathan Villaescusa** Oct 6 '12 at 18:03

not really, my problem is that the ticks are ofter a bit longer, never shorter, which sums up - so i just want to count down in seconds steps on UI level, 87, 86, 85 ... and so on. with the summing up i get occasionally 87, 86, 84, ... internally the complete duration is accurate yet the UI display is not proper – **dorjeduck** Oct 6 '12 at 18:08

if i would see a countdown timer with a missing second on UI level i would uninstall the app ;-) – **dorjeduck** Oct 6 '12 at 18:09

show **6** more comments

Your Answer

log in

or

Name

Email

Home Page

By posting your answer, you agree to the privacy policy and terms of service.

Post Your Answer