Animasi di Canvas dengan Thread

Asynctask ditujukan untuk task dalam periode pendek, dan kontrol pada thread terbatas. Padahal pada animasi dibutuhkan loop yang terus berulang. Untuk mengatasi hal ini dapat digunakan thread sendiri. Di dalam thread inilah dilakukan loop update posisi dan menggambar canvas. Kelemahan dari pendekatan ini adalah code yang relatif lebih rumit.

Program dibagi menjadi tiga class, pertama adalah class GameRunnable yang mengimplement Runnable, kedua adalah class GameView yang merupakan turunan dari class SurfaceView dan terakhir adalah activity-nya sendiri.

Berikut adalah kode untuk class GameRunnable. Method run() pada class ini yang akan dijalankan di thread terpisah. Di dalam method run dilakukan update posisi gambar dan penggambaran canvas.

```
import com.example.animasicanvas2.R;
import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.Log;
import android.view.SurfaceHolder;

//menggunakan runnable, karena hanya run yang dibutuhkan untuk dioverride
public class GameRunnable implements Runnable {

	//jika true, loop di run() berakhir
	boolean mRun=false;
	float posX=10;
	float posY=100;
	Bitmap bmp;
	private Paint cat = new Paint();

	//akses ke surface dan canvas
	private SurfaceHolder mSurfaceHolder;

	//akses ke context (untuk ambil resource)
	Context mContext;

	private void updatePosisi() {
		//geser ke kiri dan kalau sudah max, kembali ke awal
		posX= posX+10;
		if (posX>200) {
			posX=10;
		}

		//tidur 0.2 detik, agar animasi tdk terlalu cepat
		try {
				Thread.sleep((long)(1000*0.2));
		} catch (InterruptedException e) {
				e.printStackTrace();
				Log.e("yw","error saat mencoba sleep"+e.getMessage());
```

```
            }
        }


        //siapkan resources
        public GameRunnable(SurfaceHolder vSurfaceHolder, Context vContext) {
                mSurfaceHolder = vSurfaceHolder;
                mContext = vContext;
                Resources res = mContext.getResources();
                bmp = BitmapFactory.decodeResource(res, R.drawable.ic_launcher);
        }

        //bersihkan layar dan gambar objek
        public void doDraw(Canvas c) {
               //canvas perlu dicek, karena pada saat
               //user pindah ke activity lain, c bisa
               //berisi null (proses shutdown belum selesai dan
               //thread masih jalan, tapi canvas sudah didestroy)
               if (c!=null) {
                     //clear screen
                     c.drawColor(Color.WHITE);
                     //gambar bitmap
                     c.drawBitmap(bmp,posX,posY,cat);
              }
        }


    @Override
    public void run() {
        //loop forever selama tidak dishutdown (mRun diset false)
        while (mRun) {
            Canvas c = null;
            try {
                c = mSurfaceHolder.lockCanvas(null);
                synchronized (mSurfaceHolder) {
                    updatePosisi();
                    doDraw(c);
                }
            } finally {
                //kalau ada masalah
                if (c != null) {
                    mSurfaceHolder.unlockCanvasAndPost(c);
                }
            }
        }
    }


      //jika diset false maka akan shutdown (lihat method run())
      public void setRunning(boolean b) {
             mRun = b;
      }
}
```

Bagian kedua adalah class GameView yang merupakan turunan dari SurfaceView. Dalam class ini thread dimulai saat surface dicreate dan dishutdown saat surface didestroy.

```java
import android.content.Context;

import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

//surfaceView digunakan karena GUI akan seriung diupdate
//dan cepat untuk animasi
public class GameView extends SurfaceView implements SurfaceHolder.Callback  {


        public Thread mThread;
        private GameRunnable gr;


        public void startThread() {
                //runnable jadi parameter thread
                mThread = new Thread(gr);
                gr.setRunning(true);
                //saat thread dimulai, maka GameRunnable.run() dipanggil
                mThread.start();
        }

        public void shutDownThread() {
                boolean retry = true;
                gr.setRunning(false);  //matikan,

                //loop terus sampai thread berhenti
                //setRunning(false) tidak menghentikan seketika
                //coba lihat method GameThread.run()
                while (retry) {
                     try {
                         mThread.join();
                         retry = false;
                     } catch (InterruptedException e) {
                     }
                 }
        }

        public GameView(Context context) {
                super(context);
                SurfaceHolder holder = getHolder();
                holder.addCallback(this);
                gr = new GameRunnable(holder,context);
        }

        @Override
        public void surfaceCreated(SurfaceHolder holder) {
                 // jalankan thread saat surface sudah ada
                startThread();
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
                shutDownThread();
        }

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width,
                        int height) {
                // bisa digunakan untuk memberi tahu GameThread ada perubahan dimensi
```

```
                // surface
        }
}
```

Bagian yang terakhir adalah Activity biasa. Pada class ini GameView dicreate dan ditampilkan.

```
import com.example.animasicanvas2.R;

import android.os.Bundle;
import android.app.Activity;


public class MainActivity extends Activity {

GameView gv;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                gv= new GameView(this);
                setContentView(gv);
        }

         @Override
         protected void onPause() {
            super.onPause();
            //nantinya perlu diisi dengan penyimpanan state
            //melalui bundle
         }

         @Override
        protected void onResume() {
                super.onResume();
                //nantinya perlu diisi dengan load state
         }

}
```