# Pemrograman Qt 14 – QPropertyAnimation untuk Animasi GUI

Bismillahirrahmanirrahim.

Ubiquity, program pemasang sistem operasi Ubuntu yang kita pakai selalu ketika menginstal, adalah sumber inspirasi program ini. Animasi berbentuk slider layaknya slider di web yang bergerak ketika tombol panah diklik, adalah sesuatu yang gagal diimplementasikan pada Otodidak versi 1. Kegagalan itu disebabkan oleh ketidaktahuan mengenai hard coding di Java terutama pemakaian animasi di dalamnya. Kini dengan Qt, animasi sliding seperti layaknya Ubiquity di Ubuntu serasa dekat untuk dibuat. Kita bisa membuat objek-objek GUI bergerak melalui klik tombol dengan memakai kelas animasi di dalam Qt. Salah satu kelas tersebut adalah QPropertyAnimation. Tulisan ini bukan hendak membuat implementasi Ubiquity versi Qt (karena belum mampu) melainkan hanya pengantar menuju ke sana. Suatu saat saya ingin membuat Ubiquity sendiri dalam Qt. Semoga tulisan ini bermanfaat.

## 1. Spesifikasi Sistem

- Ubuntu 12.04
- Qt Creator 2.4.1
- Qt 4.8.0 (32 bit)

## 2. Daftar Kelas

QpropertyAnimation

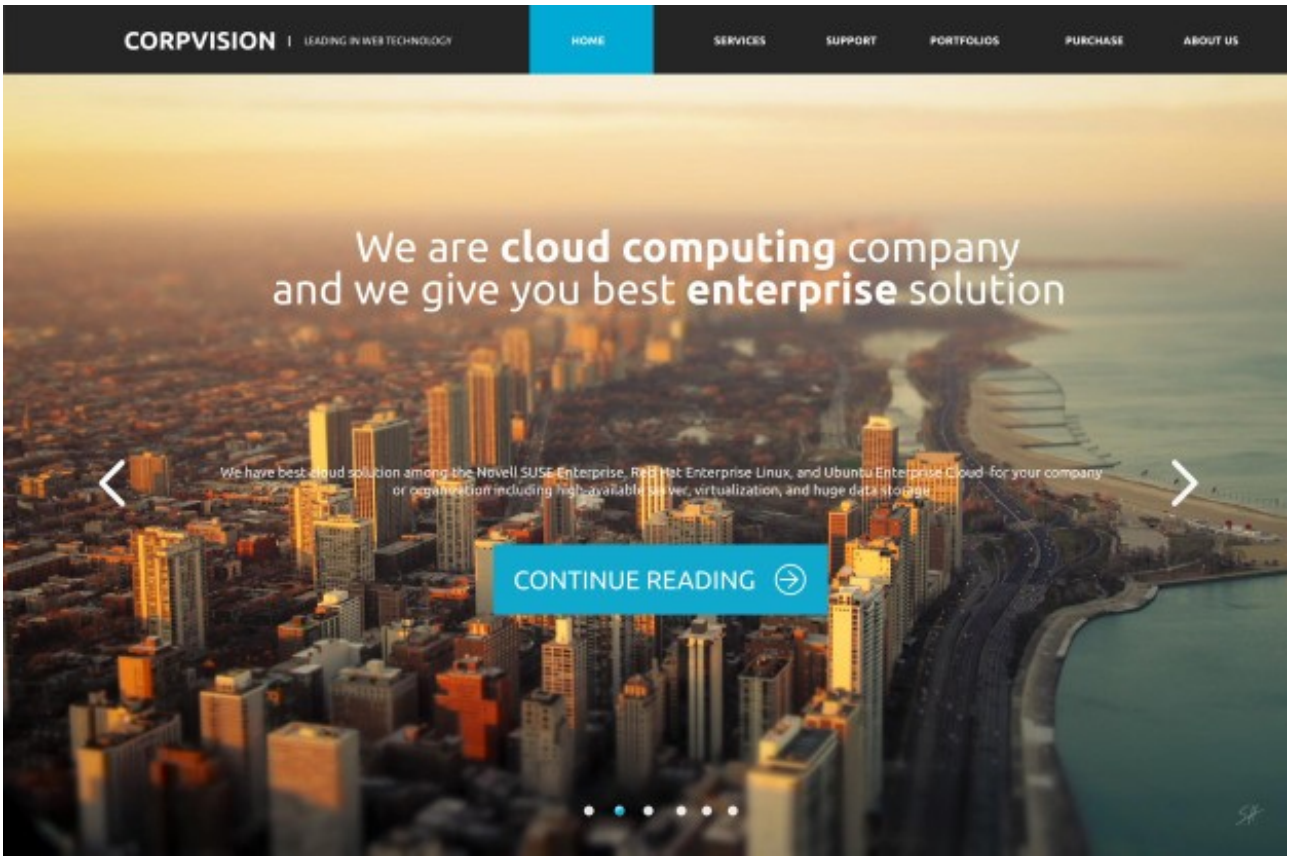
## 3. Daftar Method

- setStartValue
- setEndValue

## 4. Arah Tulisan Ini

Bayangkan Otodidak dengan antarmuka seperti Ubiquity. Itulah tujuan antarmuka aslinya Otodidak. Slider dalam Ubiquity itu mirip slider di web yang biasanya dibuat dengan jQuery atau Flash. Karena ketidaktahuan, maka antarmuka yang indah itu tidak dapat diimplementasikan. Mari mengingat kembali bagaimanakah Ubiquity dan bagaimanakah slider di web.


*Ubiquity*




*Web*

Seperti itulah slider yang beranimasi. Tulisan ini hanya akan membuat animasinya tanpa fungsi-fungsi lain slider.

## 5. Kode

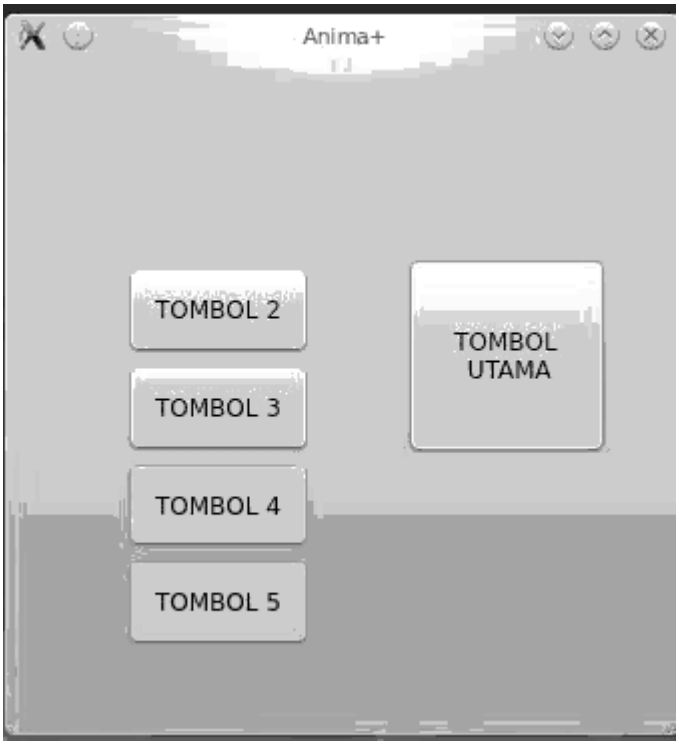
***mainwindow.h***

```
#include <mainwindow.h>

namespace ui {
 class MainWindow;
 };

class MainWindow : public QMainWindow
 {
 Q_OBJECT

public:
 explicit MainWindow(QWidget *parent = 0);

public slots:
 void animasi();

private:
 QPushButton *tombol_utama;
 QPushButton *tombol_anakan[5];
 QWidget *objek_yang_dianimasikan;
 QPropertyAnimation *objek_yang_menganimasikan;
 };
```

***mainwindow.cpp***

```
//ANIMA+
#include <mainwindow.h>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent)
{
    this->setGeometry(333,333,333,333);

    QVBoxLayout *vlayout = new QVBoxLayout;
    tombol_utama = new QPushButton("TOMBOL \n UTAMA", this);
    objek_yang_dianimasikan = new QWidget(this);

    objek_yang_dianimasikan->setGeometry(QRect(QPoint(-111,99), QSize(99,199)));
    objek_yang_dianimasikan->setLayout(vlayout);
    objek_yang_dianimasikan->show();

    tombol_utama->setGeometry(QRect(QPoint(199,99), QSize(99,99)));

    for(int i=1; i<5; i++){
        tombol_anakan[i] = new QPushButton(tr("TOMBOL %1").arg(i+1));
        tombol_anakan[i]->setSizePolicy(QSizePolicy::Expanding,
QSizePolicy::Expanding);
        vlayout->addWidget(tombol_anakan[i]);
    }

    connect(tombol_utama, SIGNAL(clicked()), this, SLOT(animasi()));
}

void MainWindow::animasi(){
    QPropertyAnimation *objek_yang_menganimasikan = new
QPropertyAnimation(objek_yang_dianimasikan, "geometry");
    objek_yang_menganimasikan->setDuration(1111);

    QRect posisiAwal(-111,99,99,199);
    QRect posisiAkhir(55,99,99,199);

    objek_yang_menganimasikan->setStartValue(posisiAwal);
    objek_yang_menganimasikan->setEndValue(posisiAkhir);
    objek_yang_menganimasikan->start();
}
```

## 6. Hasil




Mohon maaf, gambar GIF ini jelek. Saya tidak mau menggunakan video sedangkan GIF hanya mampu menerima sedikit warna. Saya pun menjadikan gambar ini grayscale.

## 7. Analisis

Pembahasan program kali ini mungkin yang terpanjang dibanding program-program sebelumnya. Ada banyak teknik baru yang perlu diperhatikan untuk membuat animasi sederhana di atas.

**mainwindow.cpp**

Ada banyak hal yang mesti diperhatikan. Saya membagi penjelasan kali ini menjadi paling penting dan kurang penting.

### *1. Paling Penting*

Yang diperhatikan: QPropertyAnimation, QWidget, show(), setDuration, setStartValue, setEndValue.

#### 1.1. Yang Dianimasi dan Yang Menganimasi

Inti program ini adalah menggerakkan QWidget dengan QPropertyAnimation. Dalam

mainwindow.cpp, ada fungsi baru bernama animasi(). Animasi yang dilakukan berjenis motion tween. Di dalam fungsi inilah terdapat deklarasi objek QPropertyAnimation. Objek ini menggerakkan QWidget yang dibuat pada fungsi MainWindow() di atasnya, dengan method setStartLocation dan setEndLocation.

```
QPropertyAnimation *objek_yang_menganimasikan = new
QPropertyAnimation(objek_yang_dianimasikan, "geometry");
```

Bentuk yang perlu diperhatikan adalah QPropertyAnimation( QObject, “geometry”). Jadi Anda bisa menggunakan QPushButton, QWidget, QLineEdit, maupun objek lain selama mereka adalah anak kelas QObject. Inilah langkah memasukkan objek untuk dianimasikan. Sedangkan parameter sebelahnya harus “geometry” seperti itu. Maaf, saya belum mampu menjelaskan dengan detail.

### 1.2. Durasi Animasi

Sebagaimana yang kita ketahui, motion tween perlu diatur durasinya sesuai kebutuhan. Maka durasi diatur oleh method setDuration(). Satuan yang dipakai adalah milisekon.

```
objek_yang_menganimasikan->setDuration(1111);
```

### 1.3. Posisi Awal dan Posisi Akhir

Setelah semua diatur, tentu animasi dilakukan dengan memanggil posisi awal dan akhir objek. Lalu dipanggillah method start() supaya animasi berjalan.

```
objek_yang_menganimasikan->setStartValue(posisiAwal);
objek_yang_menganimasikan->setEndValue(posisiAkhir);
objek_yang_menganimasikan->start();
```

Sekian saja bagian terpenting program ini.

## *2. Kurang Penting*

Yang diperhatikan: QRect, QPoint, QSize, tr(), arg(i+1).

### 2.1. QRect, QPoint, QSize

Program ini memakai QRect untuk menentukan ukuran objek QMainWindow, QPushButton, dan QWidget. QRect bisa diperluas perhitungan ukurannya dengan menentukan dahulu satu titik koordinat dengan QPoint lalu ditentukan panjang dan lebarnya dengan QSize. Ini dilakukan dalam satu parameter.

### 2.2. tr() dan arg(i+1)

Dua method khas Qt ini dipakai untuk otomatisasi pembuatan nama tombol-tombol secara otomatis. Oleh karena ini, maka kita bisa membuat banyak tombol dengan 1 perulangan for() saja. Jika tidak memakai ini, kita harus membuat satu per satu tombol dengan menomorinya satu per satu juga.

```
tombol_anakan[i] = new QPushButton(tr("TOMBOL %1").arg(i+1));
```

### 2.3. QRect Koordinat Posisi

Bagaimana bisa diketahui oleh program, posisi awal dan posisi akhir? Tentu harus ditentukan dahulu. Hal ini dilakukan dengan menulis koordinat pada objek QRect.

```
QRect posisiAwal(-111,99,99,199);
QRect posisiAkhir(55,99,99,199);
```

## 8. Kesimpulan

1. Animasi bisa dilakukan dengan QPropertyAnimation untuk menggerakkan QObject.
2. QPropertyAnimation menghasilkan animasi berjenis motion tween.

## 9. Unduh Kode Sumber

Program kali ini bernama Anima+. Silakan unduh dan buka di Qt Creator Anda.

Alamat: http://otodidak.freeserver.me/tarball/Anima+.tar.gz
Ukuran: 3 KB

## 10. Referensi

https://qt-project.org/forums/viewthread/22835/

## 11. Tentang Dokumen Ini

Dokumen ini adalah versi PDF dari posting asli http://malsasa.wordpress.com/2013/12/16/pemrograman-qt-14-qpropertyanimation-untuk-animasi-gui/. Dokumen ini ditulis dengan fonta Ubuntu 12pt. Dokumen ini disusun ulang dengan Libreoffice Writer 3.5. Dokumen ini selesai disusun pada 5 Maret 2014. Penulis mohon maaf jika terdapat kesalahan dalam dokumen ini.

## 12. Tentang Penulis

Penulis adalah warga Forum Ubuntu Indonesia. Penulis mendukung pendidikan perangkat lunak legal (terutama FOSS) untuk masyarakat. Penulis menyediakan buku-buku panduan Linux untuk pemula maupun ahli untuk diunduh secara gratis[1]. Penulis bisa dihubungi via SMS di nomor 0896 7923 7257.

1 http://malsasa.wordpress.com/pdf