

Afficher une vidéo avec Qt et MPlayer (utilisation de QProcess)

par Denys Bulant ([Tutoriels Qt](#))

Date de publication :

Dernière mise à jour :

Comment visualiser une vidéo dans Qt ?

Comment intégrer MPlayer et Qt ?

Qt, dans son souci de framework généraliste, n'implémente pas toujours tout ce dont on peut avoir besoin, surtout lorsque cela n'est pas d'usage courant. Les vidéos font partie de ce qui n'était pas couvert par Qt (maintenant supporté depuis la 4.4 par le biais de Phonon). Cependant, un grand nombre de bibliothèques et autres backends existent. Nous allons voir comment utiliser l'un d'entre eux : MPlayer.

- I - Introduction
- II - Généralités sur MPlayer en tant que backend...
- III - Exemple complet
- IV - Si vous avez des remarques...
- V - Annexes

I - Introduction




Qt, dans son souci de framework généraliste, n'implémente pas toujours tout ce dont on peut avoir besoin, surtout lorsque cela n'est pas d'usage courant. Les vidéos font partie de ce qui n'est pas couvert par Qt. Cependant, un grand nombre de bibliothèques et autres backends existent. Nous allons voir comment utiliser l'un d'entre eux : MPlayer.

Version de Qt : Toutes versions (exemple fait avec C++/Qt4 et une version pyQt disponible).

II - Généralités sur MPlayer en tant que backend...

La particularité de MPlayer est de ne pas s'intégrer par le biais d'une api, mais par un process externe. Les applications désirant l'utiliser en tant que tel, doivent communiquer avec lui par le biais de son flux d'entrée. Les flux de sorties (standard/erreur) peuvent bien sûr être analysés pour récupérer diverses infos sur la vidéo ainsi que les réponses à des requêtes pouvant être formulées par votre application.

Voici ce dont vous aurez besoin :

-  **MPlayer**, installé et fonctionnel ;
- La page des  **options de MPlayer** ;
- La liste des  **options en slave mode** pour mplayer (ce sont toutes les manipulations qui vous sont permises, donc considérez ce fichier comme votre référence ;)).

Pour utiliser MPlayer comme backend, il y a 2 arguments à lui passer impérativement :

- L'id du widget à utiliser pour le rendu par le biais de "-wid" ;
- "-slave" qui permet de le mettre en mode esclave, d'où le nom ;-)


Exemple de code lançant MPlayer comme backend :

```
QStringList args;
// On demande à utiliser mplayer comme backend
args << "-slave";
#ifdef Q_WS_WIN
// reinterpret_cast<unsigned int> obligatoire, HWND ne se laissant pas convertir gentiment ;)
args << "-wid" << QString::number(reinterpret_cast<unsigned int>(renderTarget->winId()));
#else
// Sur linux, pas de manip pour Wid :)
args << "-wid" << QString::number(renderTarget->winId());
#endif
```

À savoir aussi que sur Windows, je n'ai trouvé que **directx** comme driver de sortie ./

Cela ne concerne pas les autres OS ; lors de mon test sur Linux, je n'ai eu aucun driver à spécifier et cela fonctionne parfaitement.

Apparemment, les seuls drivers compatibles pour un embarquement de MPlayer sous Linux sont **xv**, **x11** et **gl**.

 *Sous Linux, il faut faire attention à avoir un widget de taille raisonnable pour contenir la vidéo. En effet, contrairement à Windows, un resize à la volée n'est pas possible ./*

Ensuite, le mode esclave permet de commander MPlayer pour :

- Obtenir des infos (ex. : get_video_resolution/get_time_length/...);
- Donner un ordre (ex. : play/pause/quit/...).

Tout ce qui lui est transmis doit être terminé par un retour chariot. Si vous demandez une information, elle est renvoyée sur la sortie standard avec un début de ligne propre à chaque commande (ex. : **get_video_resolution** renvoie **ANS_VIDEO_RESOLUTION='resX x resY'**).

III - Exemple complet



Voici un programme proposant les fonctionnalités suivantes :

- Lecture d'une vidéo ;
- Arrêt ;
- Avancer/reculer en déplaçant le slider ;
- Afficher le log de tout ce qui sort sur les flux **stdout** et **stderr** de mplayer.

Vous le trouverez en annexe de ce tuto. Seuls les points essentiels sont repris ici par souci de clarté. Pour vos tests, pensez à changer les valeurs de **mPlayerPath** et **movieFile** , en haut du fichier.

Ce n'est pas un frontend complet, mais cela vous permettra de voir fonctionner Qt 4 avec MPlayer en utilisant **QProcess** :)

Comme dit dans l'introduction, une version **pyQt** a été réalisée par **alteo_gange** . Elle est disponible sur **ce forum** . À noter qu'une version multi-plateforme a été postée par **egaudrain** ; le code est dispo à **ce post** .

```

class PlayerWidget: public QWidget
{
    Q_OBJECT

public:
    PlayerWidget(QWidget *parent = 0)
        :QWidget(parent), isPlaying(false)
    {
        [...]
        renderTarget = new QWidget;
        renderTarget->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);
        renderTarget->setAttribute(Qt::WA_PaintOnScreen);
        renderTarget->setMinimumSize(320, 240);
        [...]

        mplayerProcess = new QProcess(this);

        [...]
        connect(mplayerProcess, SIGNAL(readyReadStandardOutput()),
            this, SLOT(catchOutput()));
    }
}

```

```

        connect(mplayerProcess, SIGNAL(finished(int, QProcess::ExitStatus)),
               this, SLOT(mplayerEnded(int, QProcess::ExitStatus)));
        [...]
    }

protected:
    virtual void closeEvent(QCloseEvent *e)
    {
        [...]
    }

private:
    bool startMPlayer()
    {
        [...]
        QStringList args;
        // On demande à utiliser mplayer comme backend
        args << "-slave";
        // Et on veut ne pas avoir trop de chose à parser :)
        args << "-quiet";

#ifdef Q_WS_WIN
        // reinterpreter_cast<qlonglong> obligatoire, winId() ne se laissant pas convertir gentiment
        ;)
        args << "-wid" << QString::number(reinterpreter_cast<qlonglong>(renderTarget->winId()));
        args << "-vo" << "directx:noaccel";
#else
        // Sur linux, aucun driver n'a été nécessaire et pas de manip pour Wid :)
        args << "-wid" << QString::number(renderTarget->winId());
        log->append("Video output driver may not be necessary for your platform. \
                    Check: http://www.mplayerhq.hu/DOCS/man/en/mplayer.1.html \
                    at the VIDEO OUTPUT DRIVERS section.");
#endif

        args << movieFile;
        qDebug(args.join(" ").toUtf8());

        // On parse la stdout et stderr au même endroit, donc on demande à "fusionner" les 2 flux
        mplayerProcess->setProcessChannelMode(QProcess::MergedChannels);
        mplayerProcess->start(mplayerPath, args);
        if(!mplayerProcess->waitForStarted(3000))
        {
            qDebug("allez, cherche le bug :o");
            return false;
        }

        // On récupère les infos de base
        mplayerProcess->write("get_video_resolution\n");
        mplayerProcess->write("get_time_length\n");
        [...]
    }

    bool stopMPlayer()
    {
        [...]
        mplayerProcess->write("quit\n");
        if(!mplayerProcess->waitForFinished(3000))
        {
            qDebug("ZOMG, ça plante :(");
            return false;
        }

        return true;
    }

private slots:
    void catchOutput()
    {

```

```
while(mplayerProcess->canReadLine())
{
    QByteArray buffer(mplayerProcess->readLine());
    log->append(QString(buffer));

    // On vérifie si on a eu des réponses
    // réponse à get_video_resolution : ANS_VIDEO_RESOLUTION='176 x 144'
    if(buffer.startsWith("ANS_VIDEO_RESOLUTION"))
    {
        buffer.remove(0, 21);
        buffer.replace(QByteArray(""), QByteArray(""));
        buffer.replace(QByteArray(" "), QByteArray(""));
        buffer.replace(QByteArray("\n"), QByteArray(""));
        buffer.replace(QByteArray("\r"), QByteArray(""));
        int sepIndex = buffer.indexOf('x');
        int resX = buffer.left(sepIndex).toInt();
        int resY = buffer.mid(sepIndex+1).toInt();
        renderTarget->setMinimumSize(resX, resY);
    }
    // réponse à get_time_length : ANS_LENGTH=45.28
    if(buffer.startsWith("ANS_LENGTH"))
    {
        buffer.remove(0, 11);
        buffer.replace(QByteArray(""), QByteArray(""));
        buffer.replace(QByteArray(" "), QByteArray(""));
        buffer.replace(QByteArray("\n"), QByteArray(""));
        buffer.replace(QByteArray("\r"), QByteArray(""));
        float maxTime = buffer.toFloat();
        timeLine->setMaximum(static_cast<int>(maxTime+1));
    }
    // réponse à get_time_pos : ANS_TIME_POSITION=2.4
    if(buffer.startsWith("ANS_TIME_POSITION"))
    {
        buffer.remove(0, 18);
        buffer.replace(QByteArray(""), QByteArray(""));
        buffer.replace(QByteArray(" "), QByteArray(""));
        buffer.replace(QByteArray("\n"), QByteArray(""));
        buffer.replace(QByteArray("\r"), QByteArray(""));
        float currTime = buffer.toFloat();
        timeLine->setValue(static_cast<int>(currTime+1));
    }
}

void pollCurrentTime()
{
    mplayerProcess->write("get_time_pos\n");
}

// Dirige la timeline
void timeLineChanged(int pos)
{
    mplayerProcess->write(QString("seek " + QString::number(pos) + " 2\n").toUtf8());
}


// Play/stop
void switchPlayState()
{
    [...]
}

void mplayerEnded(int exitCode, QProcess::ExitStatus exitStatus)
{
    [...]
}

private:
[...]
```

```
QWidget *renderTarget;  
QProcess *mplayerProcess;  
[...]  
};
```

IV - Si vous avez des remarques...

Si vous avez besoin d'un éclaircissement, vous pouvez le faire ici :  [Forums Qt](#) .

L'article original est situé ici :  [Un conteneur pour MPlayer \(utilisation de QProcess\)](#) .

V - Annexes

Source [FTP](#) ou [HTTP](#)

