

# Qt et Box2D c'est facile

*Qt by Nokia*

par Andreas traducteur : Jonathan Courtois ([Site web](#)) ([Blog](#)) Qt Labs

Date de publication : 26/02/2010

Dernière mise à jour : 10/09/2010

Vous trouvez vos vues graphiques trop statiques ? Vous avez envie de l'animer mais vous ne voulez pas gérer le moteur physique vous-même ? Cet article vous présente Box2D qui s'occupe de toute la partie physique et qui peut facilement être lié avec Qt.

Cet article est une traduction autorisée de  **Qt + Box2D is easy!**, par Andreas.


N'hésitez pas à commenter cet article !

I - L'article original.....	3
II - Introduction.....	3
III - Installation.....	3
IV - Fonctionnement.....	3
V - Conclusion.....	4
VI - Divers.....	5

## I - L'article original

Les *Qt Labs Blogs* sont des blogs tenus par les développeurs de Qt, concernant les nouveautés ou les utilisations un peu extrêmes du framework.

Nokia, Qt et leurs logos sont des marques déposées de *Nokia Corporation* en Finlande et/ou dans les autres pays. Les autres marques déposées sont détenues par leurs propriétaires respectifs.

Cet article est la traduction du billet  **Qt + Box2D is easy!**, par Andreas.

## II - Introduction

Box2D est un moteur physique 2D de corps rigide (rigid body) open source pour le langage C++. Il est actuellement (2.0.1) disponible sous la licence MIT, qui est plutôt tolérante. Box2D est utilisé, entre autres, par Gluon (<http://gluon.tuxfamily.org/>), qui est une bibliothèque de jeux pour KDE.

## III - Installation

L'intégration de Box2D dans votre application Qt est plutôt facile et cet article vous montre par où commencer. Tout d'abord :

- étape 1 : télécharger Box2D sur Google Code : <http://code.google.com/p/box2d/> ;
- étape 2 : le compiler (j'ai dû insérer quelques `#include <cstring>` pour réussir à le compiler) ;
- étape 3 : compiler et essayer l'application test bed : Box2D/Examples/TestBed/ ;
- étape 4 : lire le manuel : <http://www.box2d.org/manual.html> ;
- étape 5 : continuer de lire cet article pour lier ces deux frameworks...

La bibliothèque ne semblait pas s'installer alors j'ai juste compilé le code source et je l'ai utilisé directement.

## IV - Fonctionnement

Ce que j'ai découvert pendant mes deux heures d'études aujourd'hui, c'est que Box2D gère un monde avec des corps semblables à la gestion des objets de **QGraphicsScene**. En bref, vous créez un objet monde (world object) et vous le remplissez avec des éléments. Certains corps sont statiques, par exemple le sol, les autres sont dynamiques, comme une balle rebondissante. Vous pouvez définir des articulations, des masses, des frottements et d'autres paramètres, définir un vecteur de gravité, puis démarrer la simulation. Box2D ne nécessite pas de système graphique, tout graphe de scène avec des éléments que vous pouvez déplacer et faire pivoter devrait fonctionner. **La vue graphique (QGraphicsView)** fonctionne très bien. J'ai basé ce code sur l'exemple "Hello World" qui est fourni avec Box2D.

L'objet monde définit les limites du système de coordonnées et le vecteur de gravité. Il paraît très proche de **QGraphicsScene**. Les limites ne sont pas appliquées selon la documentation mais j'ai eu de nombreux échecs d'exécution lorsque les éléments sont en dehors de ces limites. Vous devriez donc définir le monde suffisamment large pour englober l'ensemble de vos éléments.

```
// Define world, gravity
b2AABB worldAABB;
worldAABB.lowerBound.Set(-200, -100);
worldAABB.upperBound.Set(200, 500);
b2World world = new b2World(worldAABB,
    /* gravity = */ b2Vec2(0.0f, -10.0f),
    /* doSleep = */ true);
```

Les corps dans Box2D ont une position et un angle et vous pouvez leur assigner une forme (polygone convexe ou circulaire). Cela semble similaire à **QGraphicsItem**, qui possède également une position et une transformation. En fait, avec 4.6 la propriété de rotation correspond parfaitement à celle de l'angle dans Box2D (à l'exception que Box2D utilise des radians et effectue les rotations dans le sens opposé à **QGraphicsItem**). Cet exemple montre comment créer un corps, puis lui attribuer une forme rectangulaire :

```
b2BodyDef bodyDef;
bodyDef.position.Set(0.0f, -10);
b2Body *body = world->CreateBody(&bodyDef);

b2PolygonDef shapeDef;
shapeDef.SetAsBox(100.0f, 10.0f);
body->CreateShape(&shapeDef);
```

Les corps peuvent être statiques ou dynamiques. Les corps statiques ne se déplacent pas, tout simplement. Par défaut, les corps sont statiques. Pour rendre un corps dynamique, vous lui affectez une masse positive. Le moyen le plus simple de faire cela est de demander à Box2D de calculer la masse et l'inertie de rotation en regardant la forme du corps. Ainsi on peut légèrement modifier le code ci-dessus :

```
shapeDef.density = 1.0f;
shapeDef.friction = 0.5f;
body->CreateShape(&shapeDef);
body->SetMassFromShapes();
```

C'est vraiment tout ce qu'il y a à faire. Lorsque vous êtes prêt, avancez dans la simulation étape par étape en appelant `b2World::Step` comme ceci :

```
world->Step(B2_Timestep, B2_Iterations);
```

Après avoir appelé cette fonction, Box2D aura les positions et les angles ajustés pour tous les corps. Donc, si vous avez des items correspondants dans **la vue graphique**, vous pouvez simplement mettre à jour leur position et rotation comme ceci :

```
void adjust()
{
    // Update QGraphicsItem's position and rotation from body.
    b2Vec2 position = _body->GetPosition();
    float32 angle = _body->GetAngle();
    setPos(position.x, -position.y);
    setRotation(-(angle * 360.0) / (2 * PI));
}
```

Notez la valeur négative de Y (comme **la vue graphique** et le reste de Qt, Box2D a une composante Y orientée vers le bas) et la rotation négative qui est convertie en degrés.

## V - Conclusion

C'est vraiment tout ce qu'il y a à faire. Créer le monde, ajouter des corps, leur attribuer une masse et lancer la simulation. Utilisez l'angle et la position pour ajuster vos **QGraphicsItems** et amusez-vous avec.

*Cliquer sur l'image ci-dessus pour lancer l'animation flash*

La vidéo ci-dessus montre ma première application Box2D + Graphics View en action. Vous pouvez trouver les sources complètes ici : [qgv-box2d.tar.gz](#). J'ai essayé d'expérimenter un peu la façon dont un binding de Box2D pour Qt pourrait être faite. Pour l'instant je vais la laisser en tant qu'expérience.

## VI - Divers

Merci à [gbdivers](#), [dourouc05](#) et [jacques\\_jean](#) pour leur relecture !