Irrlicht

Étude d'un moteur de jeu

David Odin

CPE Lyon

2016

Irrlicht — Plan

PRÉSENTATION

- **Définition** : ensemble de bibliothèques permettant de développer rapidement des afficheurs 3D ou des jeux

PRÉSENTATION

- **Définition** : ensemble de bibliothèques permettant de développer rapidement des afficheurs 3D ou des jeux
- Fonctions de haut niveaux manipulant des objets graphiques (!= primitives)

Présentation

- **Définition** : ensemble de bibliothèques permettant de développer rapidement des afficheurs 3D ou des jeux
- Fonctions de haut niveaux manipulant des objets graphiques (!= primitives)
- Fonctions permettant de simplifier plein de trucs : math, portabilité, gestion de scène, optimisations, etc.

DIFFÉRENTES BIBLIOTHÈQUES

Moteur de jeu = ensemble de bibliothèques.
On trouve souvent :

- Mémoire, logs, portabilité

Différentes bibliothèques

Moteur de jeu = ensemble de bibliothèques.
On trouve souvent :

- Mémoire, logs, portabilité
- Math

Différentes bibliothèques

Moteur de jeu = ensemble de bibliothèques.
On trouve souvent :

- Mémoire, logs, portabilité
- Math
- Input

Différentes bibliothèques

Moteur de jeu = ensemble de bibliothèques.
On trouve souvent :

- Mémoire, logs, portabilité
- Math
- Input
- Graphique

Différentes bibliothèques

Moteur de jeu = ensemble de bibliothèques.
On trouve souvent :

- Mémoire, logs, portabilité
- Math
- Input
- Graphique
- Importation de données : images, modèles, décor, etc.

Différentes bibliothèques

Moteur de jeu = ensemble de bibliothèques.
On trouve souvent :

- Mémoire, logs, portabilité
- Math
- Input
- Graphique
- Importation de données : images, modèles, décor, etc.
- GUI

Différentes bibliothèques

Moteur de jeu = ensemble de bibliothèques.
On trouve souvent :

- Mémoire, logs, portabilité
- Math
- Input
- Graphique
- Importation de données : images, modèles, décor, etc.
- GUI
- Physique

Différentes bibliothèques

Moteur de jeu = ensemble de bibliothèques.
On trouve souvent :

- Mémoire, logs, portabilité
- Math
- Input
- Graphique
- Importation de données : images, modèles, décor, etc.
- GUI
- Physique
- Musique / effets sonores

GÉNÉRALITÉS

- Logger,

GÉNÉRALITÉS

- Logger,
- Gestion mémoire,

GÉNÉRALITÉS

- Logger,
- Gestion mémoire,
- Design patterns

MATHÉMATIQUES

- vec2, vec3, vec4

MATHÉMATIQUES

- vec2, vec3, vec4
- matrices (perspective, rotation, translation)

MATHÉMATIQUES

- vec2, vec3, vec4
- matrices (perspective, rotation, translation)
- quaternion

Portabilité / Input

- Clavier

Portabilité / Input

- Clavier
- Souris

Portabilité / Input

- Clavier
- Souris
- Joystick

Portabilité / Input

- Clavier
- Souris
- Joystick
- Temps

Import de données

- Image / texture, beaucoup de formats

Import de données

- Image / texture, beaucoup de formats
- Modèles 3D

Import de données

- Image / texture, beaucoup de formats
- Modèles 3D
- Décor / environnement

Import de données

- Image / texture, beaucoup de formats
- Modèles 3D
- Décor / environnement
- Sons

Scène manager

Manipulation d'un graphe où chaque noeud peut-être :

- Un personnage,

Scène manager

Manipulation d'un graphe où chaque noeud peut-être :

- Un personnage,
- Un objet décoratif,

Scène manager

Manipulation d'un graphe où chaque noeud peut-être :

- Un personnage,
- Un objet décoratif,
- Un objet scénarique,

Scène manager

Manipulation d'un graphe où chaque noeud peut-être :

- Un personnage,
- Un objet décoratif,
- Un objet scénarique,
- Un élément de décor (une pièce),

Scène manager

Manipulation d'un graphe où chaque noeud peut-être :

- Un personnage,
- Un objet décoratif,
- Un objet scénarique,
- Un élément de décor (une pièce),
- Un décor entier,

Scène manager

Manipulation d'un graphe où chaque noeud peut-être :

- Un personnage,
- Un objet décoratif,
- Un objet scénarique,
- Un élément de décor (une pièce),
- Un décor entier,
- Une caméra,

Scène manager

Manipulation d'un graphe où chaque noeud peut-être :

- Un personnage,
- Un objet décoratif,
- Un objet scénarique,
- Un élément de décor (une pièce),
- Un décor entier,
- Une caméra,
- Déplacé,

Scène manager

Manipulation d'un graphe où chaque noeud peut-être :

- Un personnage,
- Un objet décoratif,
- Un objet scénarique,
- Un élément de décor (une pièce),
- Un décor entier,
- Une caméra,
- Déplacé,
- Testé,

Scène manager

Manipulation d'un graphe où chaque noeud peut-être :

- Un personnage,
- Un objet décoratif,
- Un objet scénarique,
- Un élément de décor (une pièce),
- Un décor entier,
- Une caméra,
- Déplacé,
- Testé,
- etc.

Outils de production

Certains moteurs proposent des outils de productions de données liées :

- Un modelleur plus ou moins complet

Outils de production

Certains moteurs proposent des outils de productions de données liées :

- Un modelleur plus ou moins complet
- Un éditeur de scénario / d'éléments scénariques

Irrlicht — Plan

Un choix très vaste

- Unity

Un choix très vaste

- Unity
- Ogre3D

Un choix très vaste

- Unity
- Ogre3D
- Id2, Id3, Id4

Un choix très vaste

- Unity
- Ogre3D
- Id2, Id3, Id4
- Crytek

Un choix très vaste

- Unity
- Ogre3D
- Id2, Id3, Id4
- Crytek
- Unigine

Un choix très vaste

- Unity
- Ogre3D
- Id2, Id3, Id4
- Crytek
- Unigine
- Des dizaines d'autres sur `https://fr.wikipedia.org/wiki/Liste_de_moteurs_de_jeu`

Les avantages de Irrlicht

- C++ (connu des étudiants)

Les avantages de Irrlicht

- C++ (connu des étudiants)
- Facile à installer

Les avantages de Irrlicht

- C++ (connu des étudiants)
- Facile à installer
- Open Source

Les avantages de Irrlicht

- C++ (connu des étudiants)
- Facile à installer
- Open Source
- Assez représentatif

Les avantages de Irrlicht

- C++ (connu des étudiants)
- Facile à installer
- Open Source
- Assez représentatif
- Bien documenté

Les avantages de Irrlicht

- C++ (connu des étudiants)
- Facile à installer
- Open Source
- Assez représentatif
- Bien documenté
- Didacticiels

Les inconvénients de Irrlicht

- Plus vraiment développé (peu de développeurs actifs)

Les inconvénients de Irrlicht

- Plus vraiment développé (peu de développeurs actifs)
- Code un peu vieillot

Les inconvénients de Irrlicht

- Plus vraiment développé (peu de développeurs actifs)
- Code un peu vieillot
- Limité à OpenGL 3.0 / GLSL 1.3

Irrlicht — Plan

Compilation et édition de lien

```
CPPFLAGS = -I/usr/include/irrlicht
CXXFLAGS = -Wall -Wextra -O2 -g
```

Compilation et édition de lien

```
CPPFLAGS = -I/usr/include/irrlicht
CXXFLAGS = -Wall -Wextra -O2 -g

irrlicht-exemple1: main.o
        g++ $^ -o $@ -lIrrlicht
```

Les espaces de noms

- `irr::` Tout est dans cet espace de noms

Les espaces de noms

- `irr::` Tout est dans cet espace de noms
- `irr::core` Structures de maths, containers, etc.

Les espaces de noms

- `irr::` Tout est dans cet espace de noms
- `irr::core` Structures de maths, containers, etc.
- `irr::video` Les classes liées au driver vidéo, qui réalisent l'affichage 2D et 3D.

Les espaces de noms

- `irr::` Tout est dans cet espace de noms
- `irr::core` Structures de maths, containers, etc.
- `irr::video` Les classes liées au driver vidéo, qui réalisent l'affichage 2D et 3D.
- `irr::scene` manipulation de maillages (chargement, transformations, animation), de caméra, regroupement en un graphe, etc.

Les espaces de noms

- `irr::` Tout est dans cet espace de noms
- `irr::core` Structures de maths, containers, etc.
- `irr::video` Les classes liées au driver vidéo, qui réalisent l'affichage 2D et 3D.
- `irr::scene` manipulation de maillages (chargement, transformations, animation), de caméra, regroupement en un graphe, etc.
- `irr::io` Chargement de fichiers, utilisation transparente de fichiers compressés, etc.

Les espaces de noms

- `irr::` Tout est dans cet espace de noms
- `irr::core` Structures de maths, containers, etc.
- `irr::video` Les classes liées au driver vidéo, qui réalisent l'affichage 2D et 3D.
- `irr::scene` manipulation de maillages (chargement, transformations, animation), de caméra, regroupement en un graphe, etc.
- `irr::io` Chargement de fichiers, utilisation transparente de fichiers compressés, etc.
- `irr::gui` Les classes permettant de créer une interface utilisateur (boutons, textes, images 2D, menus, etc.)

Redéfinition des types de bases

Pour la portabilité, les types classiques sont redéfinis :

- `s8`, `u8`, `c8`

Redéfinition des types de bases

Pour la portabilité, les types classiques sont redéfinis :

- `s8`, `u8`, `c8`
- `s16`, `u16`

Redéfinition des types de bases

Pour la portabilité, les types classiques sont redéfinis :

- `s8`, `u8`, `c8`
- `s16`, `u16`
- `s32`, `u32`

Redéfinition des types de bases

Pour la portabilité, les types classiques sont redéfinis :

- `s8`, `u8`, `c8`
- `s16`, `u16`
- `s32`, `u32`
- `s64`, `u64`

Redéfinition des types de bases

Pour la portabilité, les types classiques sont redéfinis :

- `s8`, `u8`, `c8`
- `s16`, `u16`
- `s32`, `u32`
- `s64`, `u64`
- `f32`, `f64`

Redéfinition des types de bases

Pour la portabilité, les types classiques sont redéfinis :

- `s8`, `u8`, `c8`
- `s16`, `u16`
- `s32`, `u32`
- `s64`, `u64`
- `f32`, `f64`

Redéfinition des types de bases

Pour la portabilité, les types classiques sont redéfinis :

- **s8**, **u8**, **c8**
- **s16**, **u16**
- **s32**, **u32**
- **s64**, **u64**
- **f32**, **f64**

Exemples :

```
s32 a; // équivalent à int a en général
f64 b; // équivalent à double
```

Structures génériques

Pour les structures simples (vecteurs, dimensions, rectangles, etc.) Irrlicht a fait le choix de code templatisé.

- `irr::core::vector3d<float>` ou `irr::core::vector3df` pour un `vec3`

Structures génériques

Pour les structures simples (vecteurs, dimensions, rectangles, etc.) Irrlicht a fait le choix de code templatisé.

- `irr::core::vector3d<float>` ou `irr::core::vector3df` pour un `vec3`
- `irr::core::dimension2d<u32>` pour quelque chose ayant une largeur et une hauteur

Structures génériques

Pour les structures simples (vecteurs, dimensions, rectangles, etc.) Irrlicht a fait le choix de code templatisé.

- `irr::core::vector3d<float>` ou `irr::core::vector3df` pour un `vec3`
- `irr::core::dimension2d<u32>` pour quelque chose ayant une largeur et une hauteur
- `irr::core::rect<T>`, `rectf`, `recti`, pour un rectangle

Structures génériques

Pour les structures simples (vecteurs, dimensions, rectangles, etc.) Irrlicht a fait le choix de code templatisé.

- `irr::core::vector3d<float>` ou `irr::core::vector3df` pour un `vec3`
- `irr::core::dimension2d<u32>` pour quelque chose ayant une largeur et une hauteur
- `irr::core::rect<T>`, `rectf`, `recti`, pour un rectangle
- etc.

INITIALISATION DU MOTEUR DE RENDU

L'initialisation du moteur Irrlicht peut-être aussi simple que :

```
irr::IrrlichtDevice *device = irr::createDevice();
```

INITIALISATION DU MOTEUR DE RENDU

L'initialisation du moteur Irrlicht peut-être aussi simple que :

```
irr::IrrlichtDevice *device = irr::createDevice();
```

Équivalent à :

```
irr::IrrlichtDevice *device =
  irr::createDevice(irr::video::EDT_SOFTWARE,
                    irr::core::dimension2d<u32>(640, 480)),
                    16, false, false, false, nullptr);
```

INITIALISATION DU MOTEUR DE RENDU

L'initialisation du moteur Irrlicht peut-être aussi simple que :

```
irr::IrrlichtDevice *device = irr::createDevice();
```

Équivalent à :

```
irr::IrrlichtDevice *device =
  irr::createDevice(irr::video::EDT_SOFTWARE,
                    irr::core::dimension2d<u32>(640, 480)),
                    16, false, false, false, nullptr);
```

Comme premier paramètre, on préférera `irr::video::EDT_OPENGL` pour les TP.

Séparation scène / GUI

Le *device* crée trois entités importantes :

- Le *driver* qui gère l'affichage :

```
irr::video::IVideoDriver *driver =
                                device->getVideoDriver();
```

Séparation scène / GUI

Le *device* crée trois entités importantes :

- Le *driver* qui gère l'affichage :

```
irr::video::IVideoDriver *driver =
                                device->getVideoDriver();
```

- Le *scene manager* les objets de la scène et la caméra :

```
irr::scene::ISceneManager *s_m =
                              device->getSceneManager();
```

Séparation scène / GUI

Le *device* crée trois entités importantes :

- Le *driver* qui gère l'affichage :

```
irr::video::IVideoDriver *driver =
                                device->getVideoDriver();
```

- Le *scene manager* les objets de la scène et la caméra :

```
irr::scene::ISceneManager *s_m =
                               device->getSceneManager();
```

- Le *GUI Environment* qui gère tous les éléments de l'interface utilisateur 2D :

```
irr::gui::IGUIEnvironment *gui =
                               device->getGUIEnvironment();
```

SCENE MANAGER

Le *scene manager* gère un graphe composé de nœuds qui peuvent être :

- Un maillage (animé ou non) :

```
mesh = smgr->getMesh("data/tris.md2");
node = smgr->addAnimatedMeshSceneNode(mesh);
```

Chaque nœud peut ensuite être manipulé (voir documentation).

SCENE MANAGER

Le *scene manager* gère un graphe composé de nœuds qui peuvent être :

- Un maillage (animé ou non) :

```
mesh = smgr->getMesh("data/tris.md2");
node = smgr->addAnimatedMeshSceneNode(mesh);
```

- Une camera :

```
smgr->addCameraSceneNode();
```

Chaque nœud peut ensuite être manipulé (voir documentation).

SCENE MANAGER

Le *scene manager* gère un graphe composé de nœuds qui peuvent être :

- Un maillage (animé ou non) :

```
mesh = smgr->getMesh("data/tris.md2");
node = smgr->addAnimatedMeshSceneNode(mesh);
```

- Une camera :

```
smgr->addCameraSceneNode();
```

- Un décor

Chaque nœud peut ensuite être manipulé (voir documentation).

Scene manager

Le *scene manager* gère un graphe composé de nœuds qui peuvent être :

- Un maillage (animé ou non) :

```
mesh = smgr->getMesh("data/tris.md2");
node = smgr->addAnimatedMeshSceneNode(mesh);
```

- Une camera :

```
smgr->addCameraSceneNode();
```

- Un décor
- Un autre élément graphique...

Chaque nœud peut ensuite être manipulé (voir documentation).

Graphical User Interface 1/2

Le *GUI environment* permet l'affichage d'éléments graphiques 2D comme :

- Un label :

```
gui->addStaticText(L"Bonjour",
                   rect<s32>(50, 50, 150, 100));
```

Graphical User Interface 1/2

Le *GUI environment* permet l'affichage d'éléments graphiques 2D comme :

- Un label :

```
gui->addStaticText(L"Bonjour",
                   rect<s32>(50, 50, 150, 100));
```

- Une image :

```
texture = driver->getTexture("data/logo.tga");
gui->addImage(texture, position2d<s32>(10,20));
```

Graphical User Interface 2/2

Mais aussi des éléments interactifs comme :

- Un texte éditable :

```
gui->addEditBox(L"Texte éditable",
                rect<s32>(50, 50, 250, 100));
```

Graphical User Interface 2/2

Mais aussi des éléments interactifs comme :

- Un texte éditable :

```
gui->addEditBox(L"Texte éditable",
                rect<s32>(50, 50, 250, 100));
```

- Un bouton clickable :

```
gui->addButton(rect<s32>(50, 50, 150, 100),
               nullptr, id,
               L"Coucou", L"Dit coucou");
```

Graphical User Interface 2/2

Mais aussi des éléments interactifs comme :

- Un texte éditable :

```
gui->addEditBox(L"Texte éditable",
                rect<s32>(50, 50, 250, 100));
```

- Un bouton clickable :

```
gui->addButton(rect<s32>(50, 50, 150, 100),
               nullptr, id,
               L"Coucou", L"Dit coucou");
```

- Une barre de défilement :

```
gui->addScrollBar(true, rect<s32>(50, 50, 150,100));
```

BOUCLE DE RENDU

Une fois le tout mis en place, on passe la main au moteur avec la boucle d'affichage (qui peut être plus complète que cela) :

```
while (device->run())
{
   driver->beginScene();
   // Dessin de la scène :
   smgr->drawAll();
   // Dessin de l'interface utilisateur :
   gui->drawAll();
   //
   driver->endScene();
}
```

Irrlicht — Plan

BIBLIOGRAPHIE

- La documentation en ligne de irrlicht (avec les didacticiels) : `http://irrlicht.sourceforge.net/docu/index.html`

Bibliographie

- La documentation en ligne de irrlicht (avec les didacticiels) : `http://irrlicht.sourceforge.net/docu/index.html`
- Real-Time Rendering (ISBN-13: 978-1568814247) A K Peters/CRC Press

Bibliographie

- La documentation en ligne de irrlicht (avec les didacticiels) : `http://irrlicht.sourceforge.net/docu/index.html`
- Real-Time Rendering (ISBN-13: 978-1568814247) A K Peters/CRC Press