TX-Moteur 3D :

Développement d'un moteur graphique pour la plateforme de réalité virtuelle du SeT

Cyril Crassin, Mathieu Heurtault
GI05 - Automne 2005
Suiveur : Olivier Lamotte

Table des matières

Introduction

Le laboratoire SeT de l'UTBM dispose d'une plateforme de réalité virtuelle composée d'un écran pour la vision stéréoscopique ainsi que d'un système de périphériques de tracking pour le déplacement et les interactions dans l'environnement. Cette plate forme est utilisée comme outil d'aide à la décision, d'aménagement du territoire ou encore de simulation de systèmes de transport ou de flux d'individus. Ces applications sont actuellement développées et mises en oeuvre sur la plateforme à l'aide de Virtools [VIR01].

Virtools est un produit distribué sous des licences "éducation" mais tout de même relativement coûteux pour des applications commerciales comme celles menées par les partenaires du laboratoire. De plus, les limites d'une telle application dédiée au prototypage commencent à se faire sentir sur certains projets et le laboratoire avait le souhait de s'orienter vers des solutions plus ouvertes et plus adaptables à leurs besoins.

Le but de ce projet était donc dans un premier temps de rechercher et d'étudier les solutions libres et gratuites de moteur 3D disponibles puis d'en utiliser une pour implémenter un visualiseur de scènes 3D capable de remplacer Virtools pour les travaux courants de visualisation du laboratoire.

Nous nous sommes rapidement orienté vers OpenSceneGraph, un moteur déjà utilisé sur d'autres plateformes similaires à celle du SeT. Développé sur le modèle communautaire, il commence maintenant à atteindre une certaine maturité (version 1.0RC publiée en décembre) et ses fonctionnalités permettent déjà de répondre à un nombre important de besoins du laboratoire.

Chapitre 1

OpenSceneGraph

1.1 Présentation

OpenSceneGraph est un toolkit de visualisation 3D Open source basé sur le modèle du graphe de scène. L'avantage principal d'utiliser un moteur Open Source est la disponibilité du code et l'extensibilité des fonctionnalités pour l'adaptation aux besoins spécifiques à la plateforme. Il est développé en C++ sur la base de la librairie graphique OpenGL et a été porté sur de nombreuses plateformes comme Windows, OSX, GNU/Linux, IRIX, Solaris et FreeBSD. De plus, il dispose d'une communauté de développeurs et d'utilisateurs importante avec plus de 1200 abonnés à la mailing list qui lui ait dédiée ce qui constitue une source relativement importante d'informations et permet d'assurer un certain support en cas de problème. L'auteur à la base d'OpenSceneGraph, *Robert Osfield*, est très impliqué dans son projet, et répond à la majorité des sujets envoyés.

1.2 Caractéristiques techniques

OpenSceneGraph propose un nombre très important de fonctionnalités qui vont de la visualisation pure au chargement de scènes 3D en passant par l'out-of-core et l'accélération matérielle :

- Bonnes performances générales grâce à l'implémentation de fonctionnalités telles que : - Frustum et occlusion culling - Niveaux de détail - Optimisation des changements d'état OpenGL - Accélération matérielle (GPU, extensions pour l'utilisation des shaders Cg et GLSL) - Multi-threading et optimisation des formats de données 3D utilisés.
- Chargement des formats image 2D, 3D et base de données standard comme OpenFlight, TerraPage, OBJ, 3DS, JPEG, PNG et GeoTiff.
- Fonctionnalités avancées telles que des systèmes de particules, l'affichage de texte en haute qualité avec anti-aliasing TrueType(R) ...
- Utilisation des VBO, PBO et FBO, pbuffers et fonctionnalités de rendu vers texture.
- Support de la pagination Multi-thread des formats de données, ces fonctionnalités peuvent être utilisées avec tous les formats supportés par l'API ce qui autorise l'exploitation temps réel de fichiers de données 3D de plusieurs Giga octets (pour l'exploitation et génération de larges étendues de terrain par exemple).
- Mécanisme d'introspection intégré au coeur de l'API qui permet la sérialisation et la modularité de l'ensemble de l'architecture. Un mécanisme de plug-ins permettant l'ajout dynamique de fonctionnalités et l'interaction avec des applications extérieures via une interface générique est ainsi disponible.

– Support multi-thread et hautement paramétrable pour les architectures multi-CPU voir multi-GPU.

Le mécanisme de plug-ins mis en place dans OpenSceneGraphe est particulièrement intéressant puisqu'il assure une modularité totale à l'API. Les plug-ins peuvent en effet être chargés et déchargés dynamiquement au moment de l'exécution et permettent d'étendre les fonctionnalités du graphe de scène. Il permettent également le support d'une grande variété de formats d'imports et d'exports de fichiers images 2D ou de modèles 3D.

OpenSceneGraphe est en outre particulièrement adapté aux environnements de Réalité Virtuelle en fournissant un support des principaux modes d'affichage stéréoscopiques (Anaglyphe, double Left/Right buffers, Horizontal/Vertical Split). Il inclut également des plug-ins pour la gestion des périphériques de réalité virtuelle via les toolkits VRPN (cf.section 2.3.2) et VR-Juggler.

1.3 Structure

L'architecture modulaire d'OpenSceneGraph s'articule autour de trois packages principaux nommés OSG, osgUtil et osgDB (cf. figure 1.1).

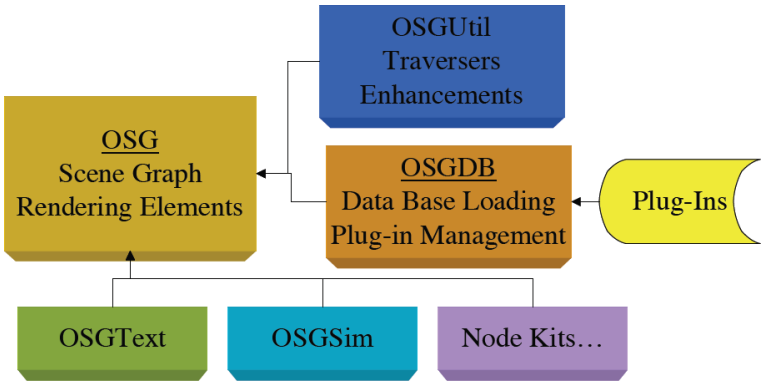


FIG. 1.1 – Schéma de principe de la structure modulaire d'OpenScenegraph

Le package OSG est le coeur de l'API et contient l'ensemble des classes permettant la construction et la gestion du graphe de scène, les structures de données pour la manipulation d'objets géométriques et de textures, les modules de visualisations, de transformations et d'animations...

Le package osgUtil contient pour sa part, un ensemble de classes utilitaires pour l'optimisation de graphes et de primitives de dessin, le culling, la triangulation de maillages, la génération de textures paramétriques...

Le package osgDB fournit le framework de lecture et d'écriture de sur ce qui est appelé base de données et correspond aux formats de données images 2D et de scènes ou modèles 3D. Il fournit le point d'entrée pour les plug-ins statiques dédiés aux différents formats de fichiers.

Il est possible d'ajouter soit des plugins pour les fichiers d'E/S, par l'intermédiaire de osgDB, soit des *Node Kits* étendant les fonctionnalités du scene graph, directement au coeur de OSG. Les Node Kits permettent d'ajouter des fonctions supplémentaires à OSG soit au moment du chargement soit directement pendant l'exécution.

Les *Node Kits* osgProducer et osgGA servent à la gestion du fenêtrage. OsgGA fournit un ensemble de classes constituant une couche d'abstraction du système de fenêtrage alors que osg-

Producer permet leur implémentation sur un système en particulier.

OsgText permet la manipulation et la visualisation de texte *True Type*. OsgSim fourni un ensemble de classes utiles dans le domaine de la visualisation scientifique de simulations. Il permet entre autre l'affichage de barres de contrôles (tableaux de bords) ou l'ajout de nouvelles transformations spécifiques à un domaine mais rien de dédié directement à la simulation (comme de la détection de collision ou de la simulation physique).

D'autres *Node Kits* existent et ajoutent le support de système de particules (osgParticle), le support du Bump Mapping, des shaders et de divers effets sur les matériaux (osgFX), ...

Chapitre 2

Travail réalisé

Le but de ce projet était de développer un visualiseur de scènes 3D basé sur OpenSceneGraph et répondant aux contraintes d'utilisation du laboratoire. Pour cela, notre travail s'est répartit autour de 3 axes principaux : l'importation de scènes modélisées sous 3D studio MAX [3DS01], la visualisation temps réel et l'exploration de ces scènes, la prise en charge de la plateforme de réalité virtuelle (périphériques et affichage stéréo).

2.1 Importation à partir de 3D studio MAX

Notre premier travail sur OpenSceneGraph a consisté à trouver un moyen de récupérer les scènes conçues sous 3D studio MAX afin de pouvoir les exploiter dans un visualiseur basé sur OpenSceneGraph.

Pour cela, nous avons utilisé le format de scène 3D propre à OpenSceneGraph appelé osg. Ce format existe en version texte (.osg) et compilé (.ive) et permet de stocker l'intégralité des composants d'un graphe de scène (même éventuellement des noeuds issus de *NodeKits*, cf. section 1.3) mis à part les textures ou les shaders stockés séparément pour la version non compilée. Des plugins d'exportation en OSG sont disponibles pour 3DsMAX, Maya et Blender.

2.1.1 Le plug-in OSGExp

Le plug-in 3D studio MAX pour l'exportation de scènes au format osg s'appelle OSGExp. Nous avons pu vérifier qu'il gère l'exportation d'un grand nombre de fonctionnalités avancées des scènes 3D studio MAX telles que les animations, les sources multiples d'éclairage, les matériaux multiples, l'environment mapping, textures procédurales (avec des limitations), le LOD, les systèmes de particules simples...

L'export se fait sous 3D studio MAX via le menu d'export standard (*File->Export*). Des options de sélection des objets et fonctionnalités exportées sont disponibles ainsi que l'activation d'optimisations de la scène, du mode de génération des primitives ou encore du format de textures utilisé (cf. figure 2.1). Les textures sont régénérés par le plug-in et placées dans un répertoire "images" à l'emplacement du fichier .osg exporté.

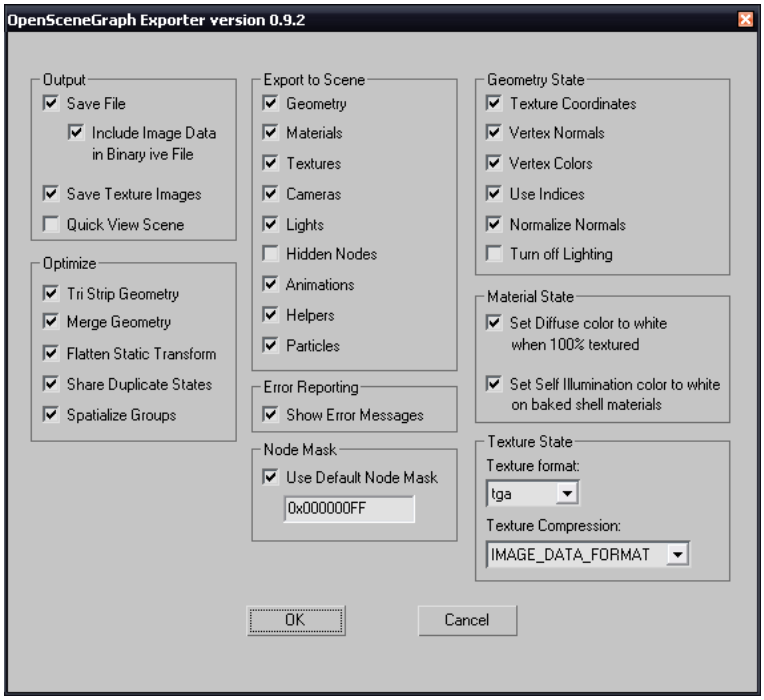


FIG. 2.1 – Fenètre d'export d'OSGExp

2.2 Visualiseur OpenSceneGraph

Une fois un moyen d'exporter les scènes à partir de 3D studio MAX trouvé, un visualiseur capable de lire et de permettre l'exploration de ces scènes a du être implémenté. Pour cela, nous nous sommes basé sur l'exemple *osgViewer* distribué avec OpenSceneGraph.

L'étape de définition d'une application OpenSceneGraph se décompose en deux grandes parties : la mise en place du système de fenêtrage et la création du graphe de scène.

La mise en place du système de fenêtrage passe par la classe *osgProducer* qui permet de créer la fenêtre, donc la surface de rendu du graphe de scène (contexte OpenGL), à partir d'une liste d'arguments passés en paramètres de l'application. Ces paramètres permettent entre autre de définir la profondeur de couleur et le format de la surface utilisée mais également le mode de vision stéréoscopique souhaité. Ce mode est passé via l'argument `"-stereo <mode>"` et peut être configuré par les variables d'environnement : `"OSG_SCREEN_DISTANCE"` qui définit la distance moyenne de l'utilisateur à l'écran, `"OSG_SCREEN_HEIGHT"` qui définit la hauteur de l'écran, `"OSG_SCREEN_WIDTH"` sa largeur et `"OSG_EYE_SEPARATION"` la distance inter-oculaire.

La définition du graphe de scène est faite à l'aide de la fonction de chargement automatique du package osgDB, celle si se charge de charger si nécessaire le plug-in correspondant au format 3D utilisé et crée le graphe de scène directement à partir du fichier. Ce graphe est ensuite optimisé à l'aide de la classe *Optimizer* du package *osgUtil* et passé comme noeud racine à *osgProducer*.

2.3 Prise en charge des périphériques de Réalité Virtuelle

2.3.1 Présentation

Une des contraintes principales de notre projet était de s'intégrer parfaitement avec la plate forme du laboratoire. Cette intégration passe par la prise en charge des périphériques de réalité virtuelle qui y sont connectés.

2.3.2 Première approche : VRPN

VRPN (Virtual-Reality Peripheral Network, [VRPN]) est une bibliothèque libre fournissant une interface transparente entre une application de réalité virtuelle tournant sur une machine, et des dispositifs physiques (trackers, wand, haptic device) connectés à d'autres machines d'un réseau. L'utilisation de VRPN permet de connecter une application à une multitude de périphériques de réalité virtuelle, simplement en modifiant la configuration du serveur VRPN. OpenSceneGraph possède un plugin d'interfaçage avec VRPN appelé osgVRPN (disparu dans la version 1.0 d'OpenSceneGraph sortie à al fin de notre TX) qui vient se connecter à osgDB, qui s'occupe de la gestion des plugins. Ainsi, nous avons pu tester VRPN sur la plateforme avec les périphériques de tracking du laboratoire. Cependant, le serveur VRPN pose un important problème d'utilisation de ressources. Nous avons donc cherché à simplifier au maximum le traitement des informations de tracking en nous exonérant du serveur VRPN.

2.3.3 Introduction à DTrack

La plate-forme du Set utilise le logiciel Dtrack de A.R.T. GmbH pour la détection des mouvements perçus au centre de la plateforme. Il s'agit d'un système de capture optique basé sur des repères sphériques réfléchissants les infra-rouges disposées en structures 3D reconnaissables appelées Body (cf. figure 2.2). Le système analyse les mouvements relatifs, d'une part des Standard Bodies, fixés sur les lunettes de l'utilisateur, d'autre part des bodies du flystick dont il transmet également l'état des boutons. Les données sont mises à disposition sur le port 5000 d'une machine serveur Dtrack (sur laquelle le logiciel s'exécute) par le protocole UDP. Les trames contiennent d'autres informations décrites dans le rapport de TX1. Les informations qui nous sont nécessaires et doivent être traitées sont celles relatives aux Standard Bodies et aux Flysticks : les positions des bodies, leurs angles et les matrices de rotation ainsi que l'état des boutons du Flystick. Celles-ci permettent de savoir sur quoi et comment effectuer une transformation de l'espace virtuel.

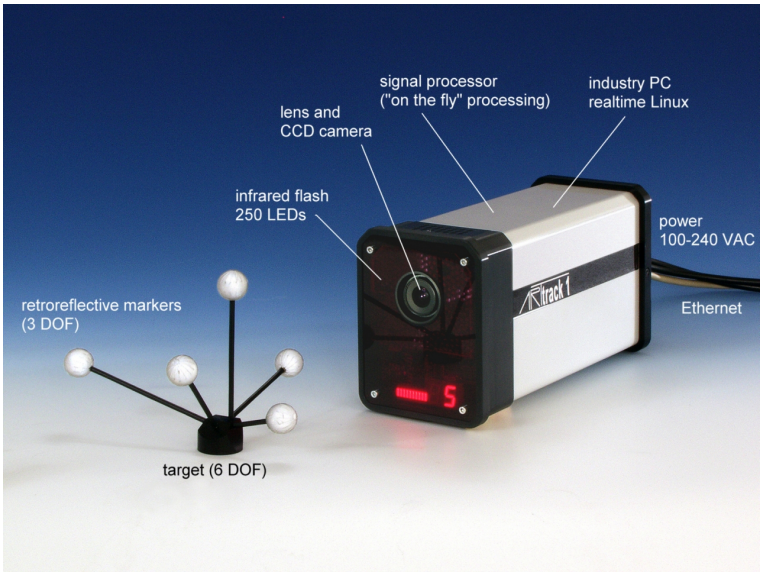


FIG. 2.2 – Composants du système DTrack

2.3.4 Intégration dans OpenSceneGraph

Notre démarche a été de rendre la prise en charge de Dtrack dans OpenSceneGraph la plus générique et réutilisable possible. Pour cela, nous avons créé un nouveau package sur le modèle d'osgVRPN et appelé osgDTrack, permettant la prise en charge directe du système Dtrack (cf. diagramme de classes en Annexe A). Des objets Body et Flystick ont été implémentés pour représenter ces périphériques et permettre leur connection aux objets de manipulation de noeuds (pour le déplacement d'objets dans la scène) ou de manipulation de caméra (pour la navigation). L'objet Body s'intègre dans l'API en implémentant le mécanisme de comptage de référence via l'héritage de l'objet Referenced. Le Flystick est une spécialisation d'un Body générique auquel il ajoute un ensemble de 8 boutons pouvant posséder un état activé ou désactivé. Notre objet Flystick hérite donc tout naturellement de Body et fournit en plus une interface pour récupérer l'état des boutons. Chaque Body et Flystick sont identifiés de manière unique (par type) par un entier correspondant à l'identifiant Dtrack du périphérique.

Les données de position et d'orientation des Body sont stockées sous forme de vecteurs (osg : :Vec3f) et également d'une matrice de transformations linéaires. L'état des boutons du Flystick est stocké sur 1 bit dans un octet encodant les 8 boutons et des flags ont été définit pour leur manipulation (de FLYSTICK_BUTTON_0 à FLYSTICK_BUTTON_7). Ces classes ont été conçues de manière Thread-safe pour pouvoir être mises à jour à partir du module de lecture de trames Dtrack présenté dans la section suivante.

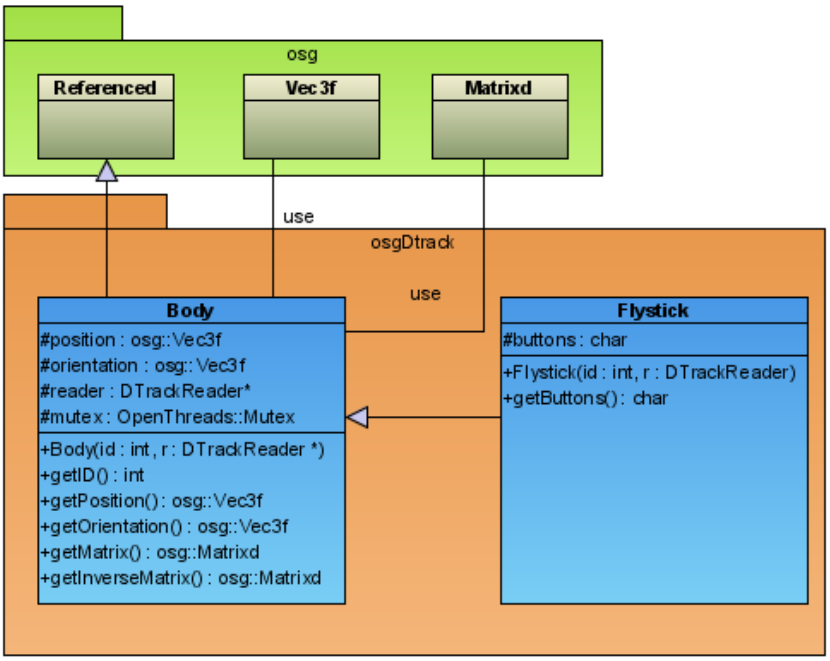


FIG. 2.3 – Classes Body et Flystick et leurs dépendances

2.3.5 Lecture des trames DTrack

La récupération des informations sur les périphériques Dtrack a été implémentée dans un module appelé DtrackReader. Ce module est chargé de récupérer les informations sur les périphériques directement à partir d'un port UDP sur lequel le serveur Dtrack inscrit les données. Ces données sont ensuite transmises aux différentes instances de Body et Flystick utilisées dans l'application. Afin de La lecture des trames UDP est threadée en asynchrone. Cet objet s'inscrit dans l'architecture d'OpenSceneGraph en héritant de la classe osg : :Referenced. Il a été mis en oeuvre de manière totalement désynchronisée du processus d'affichage afin de ne pas perturber la visualisation par l'attente de trames du serveur. Pour cela, nous avons utilisé la bibliothèque de gestion de threads distribuée avec OpenSceneGraph appelée OpenThreads. L'instanciation d'un Body entraîne l'enregistrement automatique au près du DtrackReader. La transmissions des informations se fait ensuite en 3 étapes :

- Lecture des données au format texte sur le port UDP.
- Parsing et extraction des positions, orientations, matrices de transformation et informations propres aux bodies spécialisés (i.e. boutons des flystick). pour cela nous nous sommes basé sur les informations fournis dans le rapport [Mi05].
- Inscription des données dans les instances enregistrées de périphériques.

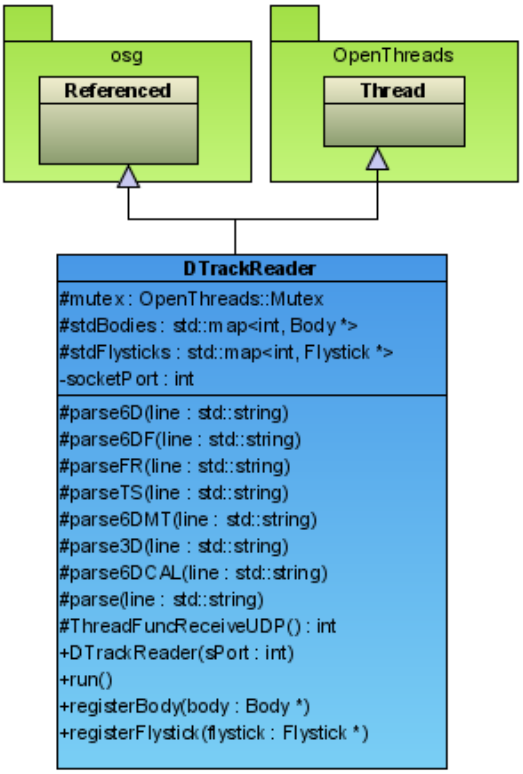


FIG. 2.4 – Classe DTrackReader et ses dépendances

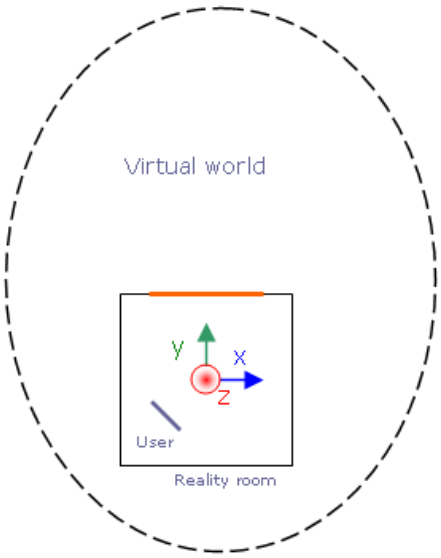
2.3.6 Transformations

Dans OpenSceneGraph, les transformations sur les objets graphiques sont manipulées via l'objet osg : :MatrixTransform qui peut être associé à n'importe quel noeud du graphe de scène. Afin de permettre la manipulation d'objets via les périphériques de Dtrack, nous avons du spécialiser cette classe afin qu'elle applique une transformation en fonction de la position et de l'orientation d'un Body. Pour cela, nous avons développé la classe BodyTransform qui hérite de osg : :MatrixTransform et à laquelle est associée un Body de manipulation. Un travail particulier a du être réalisé afin d'adapter le repère utilisé par Dtrack à celui d'OpenSceneGraph ainsi qu'a l'échelle des scènes. Une classe FlystickTransform a également été implémentée afin d'ajouter la prise en compte des boutons pour les manipulations. Cette classe pourrait être utilisée par exemple pour des opérations de picking ou autre.

2.3.7 Manipulation de la caméra

Dans OpenSceneGraph, les caméras sont gérées via le NodeKit osgProducer en charge du système de fenêtrage. Leur manipulation se fait au travers de la classe osgGA : :MatrixManipulator. Pour permettre le déplacement de la caméra en fonction des périphériques Dtrack, nous avons donc spécialisé cette classe par une classe BodyManipulator associée à un Body. De la même manière que pour les transformations, une classe FlystickManipulator héritant de BodyManipulator et ajoutant la prise en charge des boutons a également été mise en oeuvre.

2.3.8 Déplacements dans l'environnement



Il est difficile de mettre en correspondance l'espace du monde virtuel avec celui de la salle de réalité virtuelle. En d'autres termes, si en traversant la salle l'utilisateur parcourait l'intégralité du monde, l'immersion serait médiocre. Pour prendre cela en compte, le modèle de déplacement utilisé jusqu'à maintenant sur la plateforme fonctionne sur l'image d'une salle virtuelle plongée dans l'environnement visualisé. Cette salle correspond à la salle réelle dans laquelle les utilisateurs se trouvent, et l'écran de visualisation à une fenêtre de la salle virtuelle sur le monde virtuel. Cette salle est déplacée dans le monde virtuel par l'intermédiaire du Flystick. A l'intérieur de cette salle virtuelle, l'utilisateur peut se déplacer pour observer le monde virtuel par différents angles. Ses déplacements sont captés et simulés par l'intermédiaire des lunettes trackées (cf. figure ci-dessus). C'est le principe de déplacement déjà utilisé avec Virtools et que nous avons réimplémenté pour OpenSceneGraph. Ce mode de déplacement a été mis en oeuvre comme un module de manipulation de caméra appelé PlatformManipulator qui spécialise MatrixManipulator.

Le modèle décrit est le modèle de déplacement déjà utilisé avec Virtools et que nous avons réimplémenté pour OpenSceneGraph. Ce mode de déplacement a été mis en oeuvre comme un module de manipulation de caméra appelé PlatformManipulator qui spécialise MatrixManipulator. Cette classe stock indépendamment la position et l'orientation de la salle virtuelle ainsi que

celles de l'utilisateur et reconstruit à chaque déplacement la matrice de transformation résultante. L'utilisation des quaternions permet d'éviter le 'Gimbal Lock' lors des rotations successives.

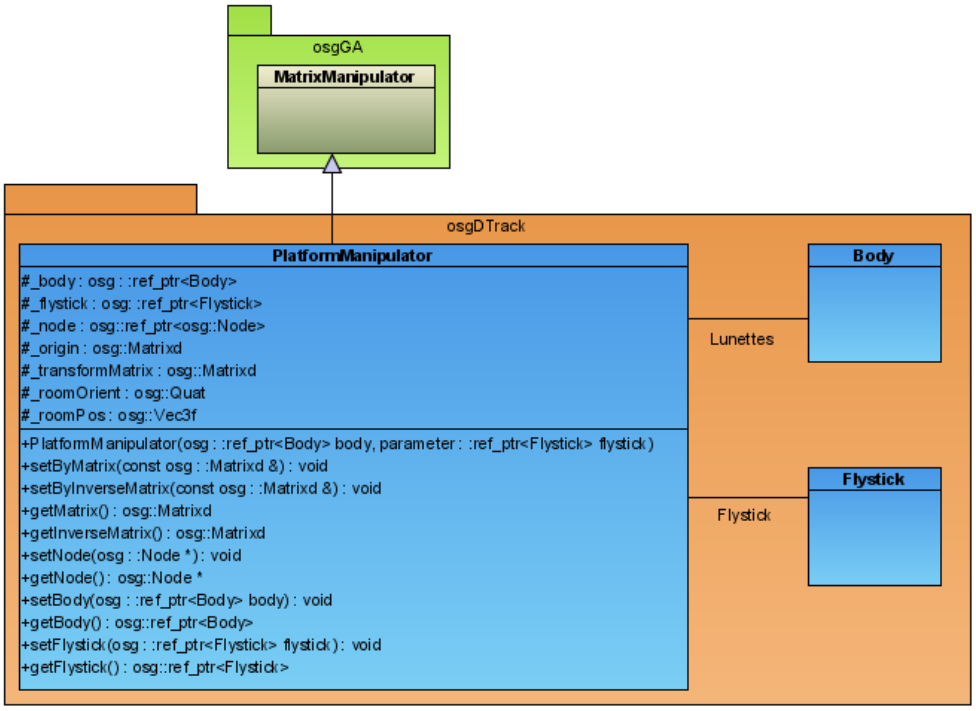


FIG. 2.5 – Classe de déplacement PlatformManipulator

Chapitre 3

Guide d'utilisation

3.1 Installation

3.1.1 OpenScenGraph

L'installation d'Open Scene Graph est relativement simple, en suivant les bonnes instructions : les fichiers d'aide inclus dans le dossier d'OpenSceneGraph. Il est d'abord nécessaire d'installer les fichiers de dépendances. Ensuite, il faut juste respecter l'ordre de compilation :

- OpenThread
- OpenProducer
- OpenSceneGraph.

Nous l'avons compilé sous Microsoft Visual Studio .net 7.0, et sous Linux, sans aucun problème. En effet, les sources contiennent à la fois le projet pour Visual Studio et le makefile pour la compilation sur les autres plateformes. Sous Visual Studio, la compilation de OpenSceneGraph s'effectue via le batch build, ce qui permet de choisir les différents éléments à compiler avec les versions Release ou Debug. Sous Linux, il suffit de mettre en commentaire les éléments qui ne sont pas souhaités. Cela permet de ne pas compiler l'ensemble des exemples, qui prennent beaucoup d'espace disque. Il est à noter que l'utilisation de Visual Studio dans sa médiocre version 6, requiert une bibliothèque supplémentaire : STLport. Le wiki du site internet OpenSceneGraph détaille la procédure d'installation de STLport1.

3.1.2 OSGexp

L'installation du plug-in pour 3dsMax est plus délicate. En effet, il est possible de trouver une version compilée du plugin sur le site Internet d'OSGExp, mais le développement du plugin ayant été interrompu en 2003, celui-ci ne fonctionne que pour la version 5 de 3DsMAX. De plus, les sources disponibles sur le site sont incompatibles avec les versions récentes d'OpenSceneGraph et il est nécessaire de recompiler le plug-in à partir des sources disponibles sur CVS et le SDK de la version utilisée de 3DsMAX.

OSGExport peut alors être compilé avec Microsoft Visual Studio 7, en portant une attention particulière aux librairies OSG utilisées. En effet, des problèmes de liaisons de librairies apparaissent si on utilise la version précompilée des bibliothèques d'OpenSceneGraph et il est donc nécessaire de les recompiler à partir des sources et de linker OSGExp à partir de ces versions.

3.2 Lancement

Le visualiseur s'exécute via un fichier .bat, qui comporte le fichier à visualiser et les commandes nécessaires pour l'établissement de la stéréovision. Il est aussi possible de lancer le visualiseur sans la stéréovision, en modifiant la ligne de commande. Cependant, cela requiert que les variables d'environnement soient définies.

Un autre moyen de lancer l'application est de lancer le shell de OpenSceneGraph (osgShell.bat), et de lancer l'application en mode ligne de commande.

3.3 Conseils de Conception de scènes pour l'export

Nous avons fait une série de tests sur la scène des eurockéennes, le technopole autour de l'UTBM, la vielle ville de Belfort, des bâtiments Alstom et nous avons relevé les problèmes suivant :

Au cours de l'utilisation de OSGExport, il est apparu qu'il était préférable d'utiliser la conversion de textures au format tga. En effet certaines textures exportées en png présentent des problèmes de transparence (nous avons observé le phénomène sur des textures de feuillage par exemple) une fois importées avec OpenSceneGraph.

OpenSceneGraph procède à un classement de primitives par objets ce qui entraîne des problèmes de transparence. En effet le classement des primitives nécessaire à la bonne composition d'objets translucides n'est pas fait à l'intérieur des objets, les primitives du fond ne sont alors pas forcément dessinées avant les objets translucides les masquants. Ceci entraîne donc des problèmes sur de gros objets composés de plusieurs éléments qu'il est préférable de décomposé lors de la modélisation. Un problème similaire se produit également lors de la visualisation de grands objets concaves contenant de plus petits objets translucides dans sa concavité (comme pour la bande texturée d'arbres entourant la scène des eurockéennes ou certaines barrières sur se site). Le moteur ne classe pas correctement les parties des objets se recouvrant ou non et la composition n'est alors pas correcte. Il faut donc également éviter ce genre d'objets lors de la modélisation.

Certains problème de mapping ont également été décelés. Il faut éviter absolument d'utiliser les modifications (décalage et répétition) des coordonnées uvw dans l'éditeur de matériaux. Celles ci ne sont en effet pas du tout prises en compte lors de l'export. Il faut également éviter d'utiliser les coordonnées de texture par défaut sur les objets qui sont parfois mal exportées et, afin d'éviter tout problème, il est préférable d' appliquer un modificateur uvw sur les objets.

Chapitre 4

Possibilités futures

Le plugin OSGExp est plutôt aboutit mais présente encore un certain nombre de limitations qui peuvent s'avérer handicapantes dans une utilisation sur des projets de production. Il exporte par exemple la scène avec des chemins de fichiers spécifiques à Windows. Il serait donc judicieux de modifier les « \ » par des « / ». Cela permettrait de pouvoir visionner les scènes exportées sous Linux, par exemple. Par ailleurs, il manque la prise en compte et l'exportation des modifications faites sur le mapping uvw des textures dans l'éditeur de matériaux. Cette option semble importante pour faciliter la conception des scènes 3D. Ensuite de nombreuses autres possibilités de 3D studio MAX, que ce soit au niveau des textures paramétriques par shaders par exemple, des animations complexes par squelettes, des systèmes de particules évolués ou des effets atmosphériques, ne sont pas prises en compte et pourraient êtres intégrés (quitte à étendre également OpenSceneGraph en conséquence).

Au niveau du visualiseur en lui même, une première fonctionnalité intéressante serait la possibilité d'import et de composition de scènes à partir de multiples fichiers osg. Ceci permettrait d'adresser les problèmes de limites mémoires rencontrées sur certaines scènes trop volumineuses. De plus, notre application ne prend pas en compte la détection de collisions. En effet, ni le sol, ni les murs ne s'opposent aux mouvements de l'utilisateur. Des idées sur ce problème peuvent être trouvées dans les archives de la mailling list d'OpenSceneGraph. En particulier, le projet nommé de « BoostManipulator » , utilise a priori la détection de collision. Ensuite, de très nombreuses extensions sont possibles, le déplacement de l'utilisateur pourrait par exemepl être amélioré, afin de le rendre plus réaliste. Un mouvement simulant le pas du personnage tel que celui implémenté dans virtools pourrait être envisageable. Afin de permettre à l'utilisateur d'interagir avec l'environnement, la mise en place de fonctions de « picking » pourrait aussi être intéressante. La mise en oeuvre d'une gestion du son serait peut être également intéressante et améliorerait l'immersion de l'utilisateur.

Une solution générique pour tout cela serait de proposer l'ensemble de ces fonctionnalités sous la forme de plug-ins à l'interface unifiée sur le modèle des building-blocks de virtools.

Conclusion

D'un point de vue personnel, ce projet de TX nous a permis tout d'abord de découvrir la plateforme de réalité virtuelle du SeT, les logiciels qui l'exploitent et ses dispositifs d'interaction. Il nous a également permis d'étudier le fonctionnement et la structure interne d'un scène graphe qui commence à être assez largement utilisé dans le domaine de la recherche scientifique en visualisation et même dans certaines solutions de production. Nous avons aussi pu étudier et analyser un plug-in pour 3DsMAX et un travail important a été mené sur les transformation géométriques afin de fournir un mode d'interaction au plus proche de celui existant actuellement avec virtools.

Nous avons ainsi fournit un moyen d'export de scènes 3D à partir de 3D studio MAX ainsi qu'un visualiseur permettant leur exploration sur la plateforme de réalité virtuelle du SeT. Notre travail constitue ainsi un premier pas vers la conception et al mise en oeuvre d'un remplaçant total à virtools pour la plateforme. Nous avons ainsi pu montrer l'intérêt et le potentiel dont dispose OpenSceneGraph pour cette tache. Les apports d'un tel moteur comme base de développement sont nombreux mais un travail important reste à faire afin de le rendre aussi flexible et rapidement utilisable dans des situations de production que virtools. Il prouve dans tout les cas déjà son efficacité aussi bien en terme de qualité de rendu que de réactivité et d'efficacité.

Bibliographie

[OSG01] Community work.
Site officiel d'OpenSceneGraph.
`http://www.openscenegraph.org/`.

[OSG02] Site officiel OpenSceneGraph.
Wiki OpenSceneGraph sur la compilation avec Visual Studio.
`http://www.openscenegraph.org/osgwiki/pmwiki.php/PlatformSpecifics/VisualStudio`.

[VIR01] Dassault Systèmes Virtools SA.
Site officiel de Virtools.
`http://www.virtools.com/`.

[3DS01] Autodesk inc.
Site officiel de 3D studio MAX.
`http://www.autodesk.com/3dsmax`.

[VRPN] Department of Computer Science at the University of North Carolina.
VRPN : Virtual Reality Peripheral Network. Site officiel.
`http://www.cs.unc.edu/Research/vrpn/`.

[Mi05] Nicolas Mignon. UTBM GI06.
Mise en Oeuvre d'un système de tracking optique sur la plateforme Open Source de réalité virtuelle "Smart".
Rapport de stage ST50.

Annexe A

Diagramme de classes

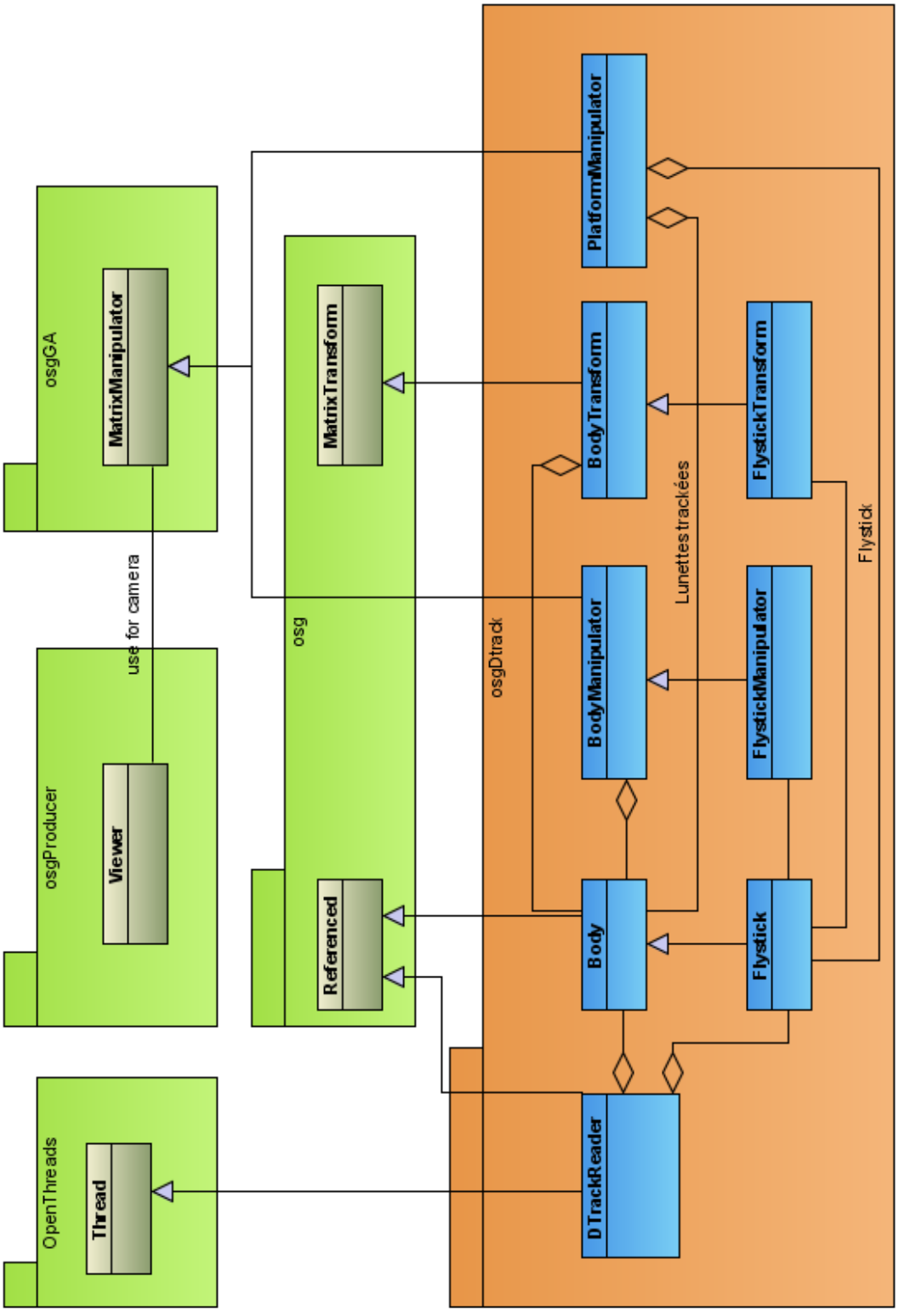


FIG. A.1 – Diagramme de classes des modules développés dans le package *osgDTrack*