Manuel du Simulateur Physique de Quadrupèdes en Temps-Réel

Étudiant : **Grégori Clauzel** — Encadrant : **Lionel Revéret**
elgoret@online.fr — lionel.reveret@inria.fr

Evasion - Laboratoire GRAVIR - INRIA Rhône-Alpes
ZIRST - 655 avenue de l'Europe
Montbonnot - 38334 Saint Ismier Cedex

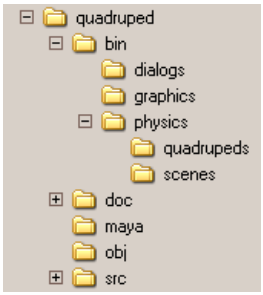
30 août 2006

Table des matières

1 Côté utilisateur

1.1 Fichiers et répertoires

- Le fichier exécutable et les binaires utilisés par le logiciel se trouvent dans le répertoire **bin**. On lance l'application avec **quadruped.exe**.
- Le fichier **quadruped.log** répertorie les différents problèmes que l'application a pu rencontrer lors de l'exécution.
- Les fichiers **.dll** sont les librairies propriétaires d'*AGEIA*[1] pour faire fonctionner le moteur physique *NovodeX*[2] utilisé. A l'heure de la rédaction de ce document, la version utilisée est la 2.3.1.



- Dans le dossier **dialogs**, se trouve le fichier décrivant la police de caractère utilisée pour afficher du texte dans le logiciel. Ce sont les polices du projet *fnt*[7] de *plib*[4].
- Dans le dossier **graphics** se trouvent les textures **generic** et **sky**. **Sky** est appliqué sur le ciel (sphérique), **generic** pour tous les objets de l'environnement.
- Dans le dossier **physics**, se trouve deux dossiers : **quadruped** et **scene**.
  - Dans le dossier **quadrupeds** se trouve les fichiers **.pml** (au format xml) décrivant les caractéristiques physiques de l'animal.
  - Dans le dossier **scene**, les fichiers **.pml** définissent les objets présents dans l'environnement.

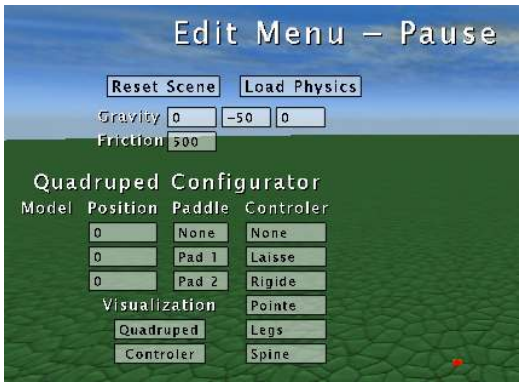
1.2 Scripts Maya

2 scripts maya ont été programmés, ils se trouvent dans le dossier **maya** :

- **qdpDefinePhysics.mel** pour modifier certaines informations dans les fichiers maya des quadrupeds
- **qdpExportPhysics.mel** pour exporter les données des quadrupèdes vers un format xml défini dans **quadruped.cpp**.

Pour s'en servir, il suffit d'ouvrir avec *Maya*[8] le fichier de *Christine Depraz* couremment nommé **newskel_16.mb**. Puis ouvrir le **Script Editor** (icône tout en bas à droite), et dans le menu **File**, choisir **Source Script..**. Reste plus qu'à ouvrir le fichier script voulu, qui s'exécutera alors.
Selon les versions du fichier Maya, les noms des os ou des animaux peuvent être différents. Il faut donc s'assurer que les noms correspondent, pour que le script se déroule correctement.

1.3 Menu de configuration



Une fois le logiciel exécuté, un menu de configuration général apparait.

- **Reset Scene** permets de remettre à zéro la scène physique. Cela efface tous les objets précédemment chargés.
- **Load Physics** ouvre un explorateur de fichiers, qui permets de charger un fichier xml contenant la description d'objets physiques. On peut choisir indiféremment un fichier représentant un quadruped, ou des objets génériques pour peupler l'environnement.
- La section **Gravity** permets de modifier au cours de la simulation la gravité du monde.
- **Friction** modifie la force de friction de tous les objets de la scène.

Lorsce que l'on charge un quadruped, son nom s'affiche sur la gauche, dans la section **Model**. En cliquant dessus, on le selectionne, et la camera pointe sur lui.

- on peut alors modifier certains de ces attributs, comme sa **position** dans l'espace.
- **Visualisation** permets d'activer ou de désactiver la visualisation de certaines informations propres à la structure du quadruped, ou à sa methode de controle.
- **Paddle** permets de choisir si l'on veut affecter un contrôle à partir d'un joystick sur l'animal.
- **Controler** permets d'affecter un type de controleur programmé sur l'animal choisi.

1.4 Contrôles de la simulation

La touche **Echap** permets d'activer ou de désactiver le menu de configuration. Quand le menu est désactivé, la simulation tourne, sinon elle est en pause.

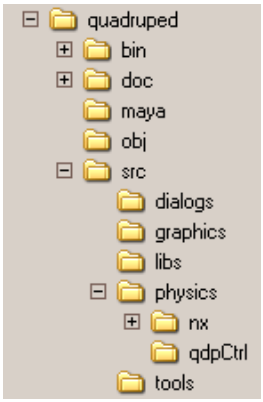
- Au cours de la simulation, on peut sélectionner un objet en cliquant sur le **bouton gauche** de la souris.
- Maintenir le **bouton droit** permets d'appliquer une force sur l'objet sélectionné, en direction du pointeur de la souris.
- Si l'on maintient le **bouton du milieu** enfoncé, tout en déplacant la souris, on fait tourner la camera autour de l'objet. Si aucun objet n'est sélectionné, cela fait pivoter la caméra.
- la **molette** de la souris permets de se rapprocher ou de s'éloigner de l'objet sélectionné, ou bien de faire avancer ou reculer la camera.
- Certaines **touches du clavier** permettent de faire des actions précises selon le contrôleur actif.
  Par exemple , **R F** et **V** permettent de controller la posture de l'animal avec le contrôleur laisse.
  G permets d'enclencher la marche pour le contrôleur spine.

2 Côté développeur

2.1 Fichiers et répertoires

Dans le dossier racine se trouve les fichiers projet. Il faut ouvrir **quadruped.vcproj** avec *Microsoft Visual C++ 2003* (version 7). Le projet peut être converti pour *2005 Express* (version 8)[9] qui est gratuite. Les sources sont dans le dossier **src**. **Main.cpp** est le fichier principal. C'est à partir de celui-ci que tous les différents composants sont activés. Généralement, chaque objet possède une fonction **init**, **maj**, **exit**, et souvent un **draw**.

- Dans le fichier **clock** se trouve la gestion du temps. L'application est bloquée à 30fps, pour obtenir un rendu graphique synchronisé avec les calculs physiques (Voir main.cpp ligne 29).
- Dans le fichier **dialogs**, est défini l'interface utilisateur (rendu du texte et des boutons par exemple).
- Le fichier **graphics** s'occupe de l'affichage, il stocke le graphisme de l'environnement, et initialise *opengl*[10].
- Le fichier **keyboard** tiens à jour les évènements du clavier. On accède directement à cette structure pour connaître l'état d'une touche. Le fichier **mouse** à le meme rôle pour la souris. Le fichier **paddle** aussi, pour les joysticks.
- Le fichier **physics** stocke les données physiques, et initialise *novodex*[2].
- Le fichier **tools** inclue divers fichiers présents dans le répertoire **tools**. On y trouve divers outils mathématiques, ainsi que le parser xml.
- Le fichier **window** s'occupe de créer la fenêtre de rendu, la mettre à jour, et la fermer.



2.2 Librairies externes

Dans le dossier **libs** se trouve toutes librairies externes qui ont été utilisées pour accélérer la programmation :

- *Glfw*[3] pour les entrées/sorties (fenetre, clavier, souris, joypad)
- *Ul*[5] pour la gestion système (explorateur de fichiers, horloge)
- *Fnt*[7] (qui dépends de *sg*[6])pour l'affichage de police de caractères

2.3 Graphics

On trouve dans ce dossier 3 composants graphiques :

- le fichier **texture**, pour appliquer des textures au ciel et aux objets.
- le fichier **camera**, qui définit la caméra *opengl*[10], avec une méthode pour la déplacer, et projeter le curseur de la souris dans l'espace 3D.
- le fichier **shadowmap**, qui calcule et affiche une shadow map, et qui gère aussi la lumière *opengl*[10].

2.4 Physics

Dans **physics**, on trouve drawshapes, qui contient les routines *opengl*[10] pour afficher les formes physiques simples.
Puis, l'on trouve tous les objets physiques définis dans le moteur :

- **Plane** est un plan infini qui sert pour le sol.
- **Box**, **capsule** et **sphere** sont les 3 formes les plus simples de *novodex*[2]. Elles sont abstraites de **rigidbody**, ce qui permet de les manipuler sans forcément connaître leur type.
- **Joint** défini les joints utilises : ce sont des joints génériques à 6 degrès de liberté configurables.
- **Bone** défini un os. Il y a 3 types d'os, qui derivent de **rigidbody**. Ils ont simplement un joint en plus de défini dans leur structure.
- **Quadruped** est la structure qui représente les animaux.
  Elle est principalement composée de **bones** reliés entre eux.

L'initialisation de tous ces objets s'effectue en chargeant un fichier xml qui respecte la syntaxe pré-établie.
(Voir dans le fichier **physics**, la méthode **ImportPML**)

2.4.1 Novodex

Dans le dossier **nx** se trouve les headers et le **physxloader.lib** du moteur *Novodex*[2] (récemment renommé *PhysX*[2]).

2.4.2 Quadruped Controlers

Dans **physics**, se trouve un dossier nomme **qdpCtrl**. Ici sont définis les contrôleurs appliqués aux quadrupèdes.
Le fichier principal est **qdpCtrl**. Il liste les différents contrôleurs disponibles dans le simulateur.
Le nombre de contrôleurs est défini par **define NBCTRL**. Il est important de mettre à jour ce nombre quand on change le nombre de contrôleurs. Il est nécessaire entre autre pour l'initialisation de l'interface. (Voir fichier **qdpCtrl.cpp**, ligne 13)

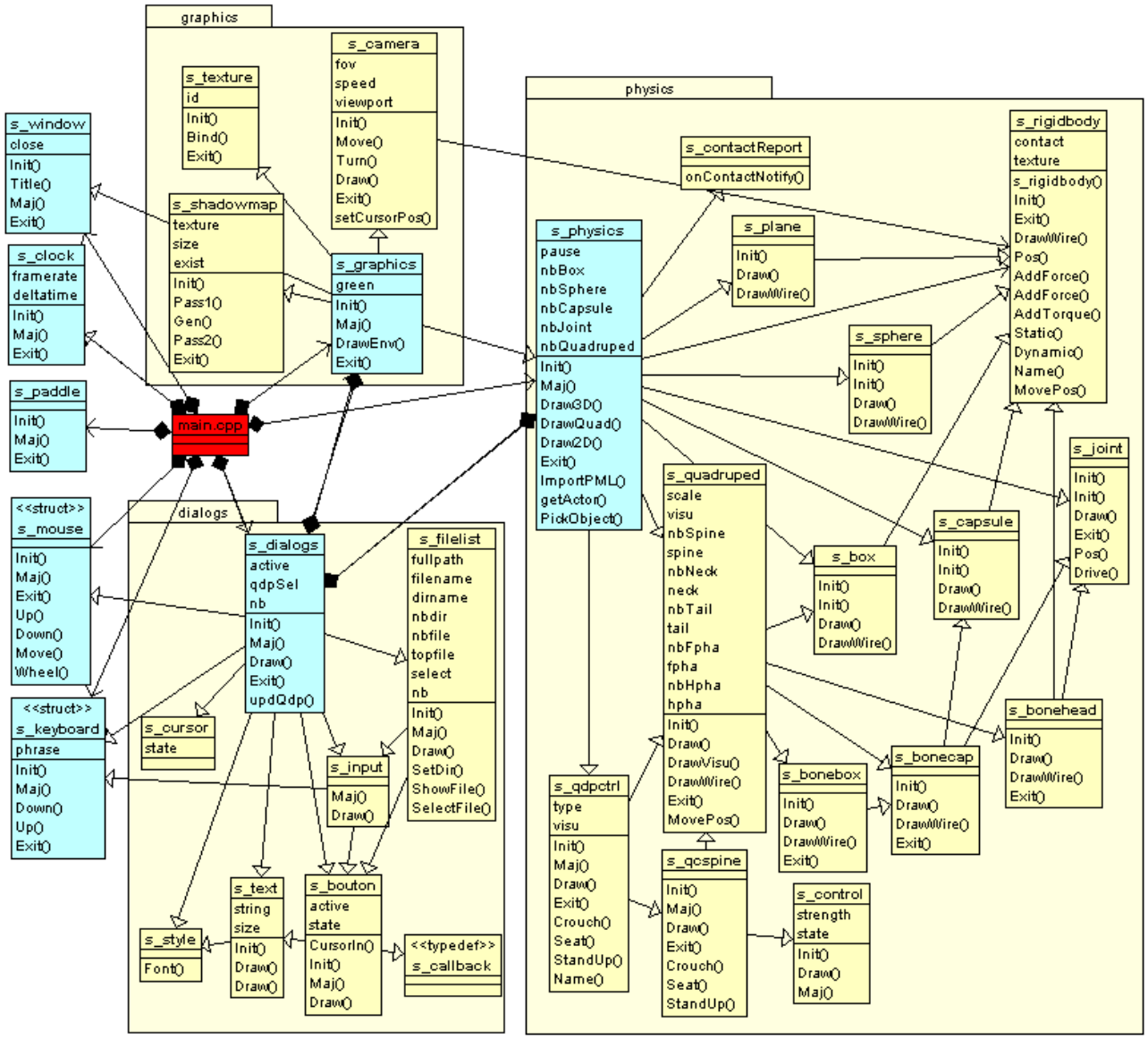
Le controleur **qcSpine** défini des segments de contrôle, qui correspondent aux classes **s_control**. Ils sont créés dans la méthode **Init** de **qcSpine**. Lors de la mise à jour, un certain nombre de forces sont appliquées pour faire marcher le quadrupède. Le code est légèrement commenté dans la fonction **Maj** pour expliquer le processus.

2.5 Dialogs

Au début du fichier **dialogs.cpp**, se trouvent les différents callbacks pour l'interface. Ils sont passés en paramètres lors de la création de boutons, ou autres éléments de l'interface. Dans le dossier dialogs, on trouve tous les différents composants.

- **Text** est l'objet le plus simple, il affiche seulement un texte.
- Il faut lui spécifier un objet **style**, qui contient les informations sur la couleur, et la police de caractère. Cet objet est utilisé par tous les éléments de l'interface. Pour changer par exemple la couleur de l'interface, il suffit de changer ce paramètre dans **style**, pour que cela s'affecte à tous les boutons.
- **bouton** dérive de **text**. On peut spécifier un callback à **bouton**, pour exécuter une action lorsque le curseur clique dessus.
- Pour cela, il faut définir un **cursor** que l'on passe en paramètre lors de la mise à jour. Ceci est géré dans **dialogs.cpp**, ligne 176 par exemple.
- **Input** est un **bouton** dont on peut modifier le contenu texte en cliquant dessus.
- **filelist** est l'explorateur de fichiers.

3 Diagramme de Classes



J'espère que cette documentation succinte permettra de faciliter la reprise de mon travail.
Pour toute question ou incompréhension, n'hésitez pas à me contacter par mail sur elgoret@online.fr.

4 Bibliographie

Références

[1] *AGEIA*
http://www.ageia.com

[2] *NovodeX / PhysX API*
http://www.ageia.com/developers/downloads.html

[3] *glfw*
http://glfw.sourceforge.net/

[4] *PLIB*
http://plib.sourceforge.net/

[5] *plib : UL*
http://plib.sourceforge.net/util/index.html

[6] *plib : SG*
http://plib.sourceforge.net/sg/index.html

[7] *plib : FNT*
http://plib.sourceforge.net/fnt/index.html

[8] *Maya*
http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=7635018

[9] *Microsoft Visual C++ 2005 Express Edition*
http://msdn.microsoft.com/vstudio/express/visualc/

[10] *OpenGL*
http://www.opengl.org/