ISAAC Benoit
LE MAITRE Aurélie

PROJET VRML

Option Robotique

Collisions de robots

Avril 2004

SOMMAIRE

❖ Description du sujet choisi

❖ Manuel d'utilisation

❖ Explication du code

I. Description du sujet choisi

Comme prévu, nous avons réalisé une interface graphique mettant en scène deux robots en mouvement sur une piste quadrillée et comportant des obstacles.

Les trajectoires des deux robots sont calculées de manière aléatoire en temps réel sous certaines contraintes ; ainsi le robot doit toujours rester à l'intérieur de la piste et doit éviter les obstacles (deux barres verticales représentants les palmiers) et les positions bloquantes.

Le but final est de visualiser de manière ergonomique et agréable à l'œil dans quelles conditions et au bout de combien de temps apparaît une collision, celle-ci étant détectée dès lors que les deux robots se « rentrent dedans ».

Ce but s'inscrit dans le cadre plus large de la Coupe de France de Robotique 2004, En effet, la configuration que nous avons représentée correspond aux règles de cette année (« Coconut Rugby ») que vous pouvez consulter à l'adresse suivante : http://www.planete-sciences.org/robot/ .

Une des plus grosses difficultés de la Coupe de France de robotique est le positionnement absolu du robot dans la zone de jeu.

Si celui-ci n'a pas d'adversaire, il est relativement facile d'obtenir la position actuelle du robot, à l'aide d'un bon asservissement et de roues codeuses.

Mais dès qu'il y a un adversaire, le risque de collision apparaît, et lorsqu'il y a collision, les roues patinent et le robot se « perd » (il peut même arriver que le choc déplace le robot) Ce problème est un de ceux auquel l'équipe de robotique de l'IIE est actuellement confrontée.

Dans cette optique, nous avons voulu visualiser dans quelles conditions les collisions se produisent, et avec quelle fréquence.

II. Manuel d'utilisation

Démarrage :

Le fichier principal est le fichier **index.wrl**, les autres contiennent des PROTOS ou du code javascript. Il y a également un fichier son et des fichiers images.

La vue initiale est celle du côté de la piste, avec les boutons au premier plan. Vous pouvez voir la piste et les deux robots.

- Le bouton rouge « GO », qui devient vert lorsqu'il est actionné, permet de faire démarrer les robots et de les arrêter.
- Le bouton « RESET » permet de remettre les robots en position initiale.
- Le bouton « PALM » modifie aléatoirement la position des palmiers (en respectant les contraintes des règles officielles, c'est-à-dire qu'ils doivent être placés symétriquement par rapport au centre de la piste, et à une intersection du damier).

Fonctionnement :

Les robots font des trajectoires aléatoires, déterminées comme ceci :

- on génère aléatoirement une position dans un carré de côté 1 autour de la position actuelle du robot.
- si cette position est atteignable (dans la piste, pas de palmiers dans la trajectoire), alors elle est conservée et elle devient le prochain but à atteindre.
- À intervalles de temps réguliers, le script DeplaceRobots calcule la position actuelle de chaque robot, les deux ayant des trajectoires indépendantes.
- quand le but de l'un d'eux est atteint, on lui calcule une autre position à atteindre.
- Si les deux robots entrent en collision (c'est-à-dire si la distance qui les sépare est inférieure à un diamètre de robot), ils s'arrêtent, un son est déclenché et il faut cliquer sur le bouton « reset » pour les remettre en position initiale. **On peut alors éventuellement changer la position des palmiers pour étudier une autre configuration d'obstacles.**

Le comportement des boutons dépend de ce qui se passe dans la scène : ainsi si les robots fonctionnent, ils faut les arrêter (STOP) avant de pouvoir faire un RESET ou modifier la position des palmiers. Les deux boutons RESET et PALM ne sont pas cliquables pendant qu'un déplacement est en cours. De même, après une collision entre les deux robots, il faut nécessairement cliquer sur RESET pour pouvoir à nouveau re-démarrer l'animation.

Tout ceci est géré à l'aide du script **DeplaceRobots, ainsi que des différentes ROUTES.**

III. Explication du code

La plupart de nos structures sont définies dans des Protos auxquels on fait éventuellement appel dans d'autres fichiers à l'aide d' « externProto ».

Nous avons tenté d'utiliser le plus possible de nœuds différents dans notre projet, afin de tester un peu toutes les fonctionnalités du VRML : (Nous reprenons ici la classification de la Cheat Sheet)

➢ Sensors

- **TouchSensor** : Nous en avons utilisé pour les boutons, à un ou deux états. Lorsqu'un bouton est à deux états, il a été nécessaire d'ajouter un petit code Javascript permettant de conserver la valeur de l'état en cours, afin que celle-ci puisse être « routée » vers d'autres fonctions*.
- **TimeSensor :** Il permet de gérer l'animation des robots dans le temps. Lorsque l'on clique sur un TouchSensor, cela déclenche l'activation d'un TimeSensor qui à son tour déclenche à intervalles réguliers une fonction Javascript.

➢ Transform and Special Groups

- **Transform :** Utilisés un peu partout
- **Anchor :** utilisé à titre d'illustration pour que le logo « ORIGINE » pointe vers le site web de l'équipe.
- **Switch :** utilisé pour les boutons à deux états notamment, dont le texte peut changer. Selon la valeur de l'état en cours (obtenu par la méthode expliquée plus haut), le choix **whichchoice** est modifié ce qui provoque un affichage différent.

➢ Script

- **Script :** Quasiment toute l'interactivité est gérée grâce à des scripts Javascripts, qui modifient des variables qui sont ensuite « routées » vers les différents éléments de la scène.

➢ Bindables

- **Viewpoint :** plusieurs points de vue ont été définis. Ils sont accessibles, avec le client Cortona, soit avec un clic-droit, soit en cliquant sur les flèches en bas du player.
- **NavigationInfo :** ce nœud nous permet de définir le type de navigation.

➢ Lights

- **SpotLight :** Quelques spots coniques sont situés autour de la scène pour l'éclairer.
- **DirectionalLight :** Nous avons également utilisé une lumière directionnelle éclairant l'ensemble de la piste.

➢ Sound

Nous avons ajouté un son déclenché au moment de la collision, faisant appel à un AudioClip contenu dans un fichier externe.

➢ Shape / Appearance / Textures

Nous avons testé les nœuds « appearance » et « material » et appliqué des textures (images externes) aux robots.

➢ Prototypes

Dans la mesure du possible, nous avons allégé le fichier principal en utilisant des prototypes définis dans d'autres fichiers.

Il y a des prototypes pour la table et les boutons à 1 et 2 états.

De manière globale, tous les évènements sont gérés par le script DeplaceRobots qui affecte les nouvelles positions aux robots, aux palmiers (si l'on clique sur « PALM ») et qui vérifie dans quel état on est (démarrage, reset, arrêt temporaire,...) pour activer ou non les boutons (à l'aide des champs **enabled** et des ROUTE).