Navigation dans un environnement 3D

Marc BEYSECKER, Tom GIMENEZ, Valentin HIRSON, Léo RIZZON

Professeur encadrant : Mr Frédéric Koriche

Université Montpellier II

Master 1 Informatique

Table des matières

1 Remerciements **2**

2 Introduction **3**

3 Recherches préliminaires **4**
3.1 Une scène en 3D avec Blender 4
3.2 Récupérer notre scène en 3D 4
3.2.1 OpenSceneGraph 4
3.2.2 OpenGL avec nos propres structures de données 4

4 Implémentation finale **5**
4.1 Un parser pour récupérer notre scène 5
4.2 Nos structures de données 5
4.2.1 La scène 5
4.2.2 Le graphe géneré 5
4.3 Travaux effectués sur le graphe 5
4.3.1 Création du graphe des way-points 5
4.3.2 Simplification du graphe obtenu : merging 5

5 Conclusion **6**

Chapitre 1

Remerciements

Ma mère, ta mère. Merci mon cul bonsoir messieurs dames.

Chapitre 2

Introduction

Contexte

Dans les jeux vidéos, les personnages non joués par des humains doivent pouvoir se déplacer de manière autonome et cohérente. Un environnement 3D est constitué d'un graphe énorme avec des milliers de sommets. Même en ne prenant que le sol, le graphe est encore très gros et, surtout, n'est pas seulement constitué des points naviguables. C'est à dire que le personnage ne doit pas pouvoir se déplacer dessus.

Comme nous l'avons étudié, la recherche de chemins dans un graphe est un problème classique mais qui peut s'avérer très lourd sur de gros graphes. En prenant en compte que les jeux mettent en scène un grand nombre d'agents il s'agit de minimiser les temps de calculs.

Il s'agit donc d'une part de ne sélectionner que les points naviguables et de simplifier le graphe obtenu pour limiter les espaces de calculs. De ces opérations nait le graphe de way-points. Il s'agit donc du graphe sur lequel vont se déplacer les agents.

Dans la plupart des jeux actuels les environnements sont créés "à la main" par les créateurs du jeu et les game designers placent eux même les points du graphe des way-points. Cela représente un gros travail et c'est même impossible dans le cas d'environnements aléatoirement générés.

Dans le cadre de notre unité d'enseignement intitulée Algorithmes de l'Intelligence Artificielle, nous avons réalisé un projet qui consiste en l'implémentation d'un algorithme de génération automatique du graphe des way-points.

Objectifs

La base de notre travail est une scène 3D créée à l'aide d'un logiciel d'éditions d'objets 3D, par exemple Blender. Il s'agit alors de charger cette scène pour y appliquer nos traitements. Tout d'abord, il faut générer le graphe à partir de tous les points composant les différentes formes pour représenter les arêtes. Ensuite il faut épurer ce graphe pour ne garder que les sommets et les arêtes "emruntables". Enfin on va chercher à appliquer un algorithme de simplification pour ne garder que les points réellement utiles. C'est une procédure appelée merging.

Le résultat serait donc un graphe des way-points, simplifié, automatiquement généré.

Chapitre 3

Recherches préliminaires

3.1 Une scène en 3D avec Blender

3.2 Récupérer notre scène en 3D

Une fois notre scène créée sous Blender s'est posé le problème de l'exploiter. Nous avons alors cherché du côté des librairies existantes, en C/C++ surtout. En effet, Blender nous proposait divers format de fichier de sortie. Nous devions faire en sorte que les informations soient récupérables et exploitable pour nos algorithmes. L'idée générale est d'utiliser un parser qui va lire les données du fichier pour les transformer en données compréhensibles par notre programme.

Voici les solutions que nous avons envisagées.

3.2.1 OpenSceneGraph

3.2.2 OpenGL avec nos propres structures de données

Une fois l'expérience OpenSceneGraph terminée et mise de côté, nous avons dû de nouveau chercher une solution. Après de longues recherches, avec quelques possibilités comme OGRE (Object-Oriented Graphics Rendering Engine), nous avons finalement décidé de créer nous même ce dont nous avions besoin.

L'utilisation d'OpenGL, conseillée par notre professeur encadrant

Chapitre 4

Implémentation finale

4.1 Un parser pour récupérer notre scène

4.2 Nos structures de données

4.2.1 La scène

4.2.2 Le graphe géneré

4.3 Travaux effectués sur le graphe

4.3.1 Création du graphe des way-points

Parcours

Heuristique

4.3.2 Simplification du graphe obtenu : merging

Chapitre 5

Conclusion

Bilan technique

Bilan personnel