Compte rendu Projet C

Julie Lemaître Virgile Daugé

Recherche du sujet :

Nous portions tous les deux un intérêt à la simulation de phénomènes physiques et naturels, d'une part pour le côté très visuel et d'autre part pour le côté pratique, s'appliquant au réel. Plusieurs sujets s'offraient alors à nous : simulation de gravité, algorithmes de recherche du plus court chemin inspiré par les fourmis, algorithmes de constructions de termitières.... Mais notre choix a finalement porté sur un algorithme de birdoid, suivit d'une simulation visuelle en trois dimentions. En effet notre intérêt pour l'intelligence artificielle à primé sur les autres possibilités, notamment la « swarm intelligence ».

Une fois le sujet déterminé, il restait à choisir quelle type de représentation utiliser (3D, 2D...) ainsi que les technologies à mettre en œuvre pour gérer entrées et l'affichage. Nous avons choisit d'utiliser la librairie Simple DirectMedia Layer dans sa version 1.2 qui permet à la fois de gérer aisément les entrées et d'intégrer openGl. Nous avons donc penché pour une représentation 3D, utilisant la librairie openGL dans ça deuxième mouture, cela pour éviter de devoir reprogrammer les fonctions simples même si elles ne sont pas forcément optimisées. L'openGL étant une nouveauté pour nous nous avons préféré jouer la carte de la sécurité. Nous avons choisi de travailler avec l'éditeur de texte sublime text ainsi que les options de débogage de gcc pour pouvoir vraiment apprendre et comprendre ce que l'on fait.

Mise en place du modèle :

Nous avons choisit de développer le modèle petit à petit en testant de façon systématique chaque fonction ajoutée, dans le but d'éviter au maximum de construire le modèle sur des briques douteuses. On as ainsi réalisé un fichier vecteur.c accompagné de son .h ayant pour but de simplifier l'utilisation des calculs vectoriels de déplacement. Il contient une fonction d'allocation, les opérations de base et enfin une fonction de copie.

Par la suite nous avons créé un fichier boid.c modélisant un boid, et permettant quelques opérations basique comme calculer la distance entre le boid et un point de l'espace et bien sur l'allocation mémoire.

Nous avons ensuite logiquement établi le modèle. Ce dernier comporte de nombreux éléments le composent :

-tableau de boids

-tableau de prédateurs (modélisées sous forme que les boids)

-tableau de nouriture (tableau de vecteurs)

-des entiers contenant les valeurs actuelles et maximum de chaque groupe

-tableau de boids a proximité

-tableau de prédateurs à proximité

-vecteur position de la nouriture a plus proche

Le principe de calcul de déplacement est le suivant : pour chaque boid, on calcule dans un premier temps ce qu'il peut voir. Ensuite on applique des règles qui définissent son comportement comme le regroupement ou encore la fuite du prédateur. Cet ensemble de règles sont pondérées, pour pouvoir paramétrer des priorités. Elles sont ensuite additionnées une à une a un vecteur qui représente le déplacement à effectuer. On limite ensuite les coordonnées de ce vecteur pour éviter un déplacement trop important voir incohérent.Enfin, on fait une moyenne pondérée entre l'ancien vecteur de déplacement et celui nouvellement calculé, ce dans le but de donner une inertie, et donc une certaine continuité aux mouvements. Le principe est le même pour les prédateurs, avec des règles qui leur sont spécifiques.

Il restait maintenant à réaliser l'interface graphique, ainsi que la prise en compte des entrées. La boucle principale du programme vérifie en permanence les events d'entrée, et compte le nombre de millisecondes depuis le lancement de l'application. Nous avons choisit de recalculer et de réafficher le modèle en fonction d'une variable fps (fixée à 60) . L'affichage en lui même est décomposé en trois parties :

-La préparation comprenant le vidage des buffers, la réinitialisation de la matrice de modèle ainsi que la mise à jour de la position de la caméra.

-Le calcul de l'affichage des boids/prédateurs/nourriture

-Et enfin on swap les buffers pour afficher les modifications à l'écran.

Nous avons également implanté une caméra basique qui regarde toujours le centre de la scéne mais dont les coordonnées varient (on tourne ainsi autour du centre).

Difficultés rencontrées :

La première des difficultés rencontrées fut la décomposition du modèle en tâches à réaliser. Comment s'organiser, comment réaliser le modèle ? Nous avons répondu à ces deux question de la manière décrire ci-dessus.

Julie à rencontré des difficultés pour l'implantation des règles du modèle. Après un premier jet non concluant, nous avons décidé de les retravailler ensemble pour de meilleurs résultats. Décision payante, puisque nous avons pu avancer de nouveau dans le projet.

Nous avons également été confrontés à un problème récurrent en c, les segmentation fault. Fort heureusement nous avons limité leur apparition en adoptant une méthode de développement sûre . Nous avons simplement codé des fonctions de test pour chaque fonction, en même temps si ce n'est avant les fonctions en question. Les seuls segfault restant étaient alors simples à débusquer(erreur d'index, faute de frappe....).

La réalisation de l'affichage en 3D avec openGL à demandé un apprentissage préalable, ainsi qu'une étude des modes de projections et de leurs subtilités. Ensuite nous avons pu réaliser quelques fonctions d'affichage simples permettant de créer un rendu de notre modèle. Puis est venu la gestion de la caméra. En revanche, même si nous avons pu réaliser un rendu rapidement, il n'était pas vraiment explicite, il nous a fallut changer les paramètres d'affichage afin de mieux visualiser les mouvements effectués par les boids.

Enfin, la dernière difficulté à été d'affiner les paramètres du modèle afin d'assurer une cohérence au modèle, en effet, celui-ci étant fortement paramétrable, quelques petites variations peuvent entrainer un changement radical du modèle. Nous avons donc du, de manière empirique détérminer des paramètres optimaux pour grandement améliorer le comportement du modèle.

Volume horaire

	Julie	Virgile
Travaux préliminaires	5h	5h
développement	10h	50h
Réglage paramètres	10h	10h