Synthèse personnelle de TPE

Porteries Tristan

31 janvier 2016

Table des matières

Résumé

1 Choix de l'équipe

Le choix de l'équipe ne m'a pas pris beaucoup de temps, en effet Thibaut Manceau, Alexis Gros et moi étions plutôt avancés au niveau des connaissance informatique, et nous avions déjà travaillés ensemble avant sur d'autre projets sans problèmes majeurs.

2 Choix de la problématique

Le choix du sujet et de la problématique a duré plusieurs séances, nous étions au début partie sur le domaine de la vidéo et plus précisément de la localisation d'une camera pour un rendu en temps réel des images de synthèses. Puis nous avons changé pour une TPE sur la sécurité sur internet mais trouvant ce sujet bien trop vague nous nous sommes mit d'accord sur la simulation de terrain. En effet depuis bien longtemps Alexis et mois travaillons sur une génération de terrains 3D optimisé pour les jeux vidéos. Nous nous sommes dit alors que nos travaux pourraient servir au TPE, mais pour créer et non simplement expliquer des travaux déjà finit, nous avons changé de cap pour la simulation d'un terrain basé sur la déformation des plaques tectoniques. Notre problématique finale à était : ≪ Comment simuler numériquement la géomorphologie de type alpine ≫.

3 Résumé du carnet de bord

Pour commencer nous nous sommes renseignés sur le sujet de la simulation géomorphologique, nous trouvâmes très peu d'informations. Les seules informations trouvées était quelque thèses du siècle dernier traitant principalement la parallélisation du calcule. Les thèses était toutes d'accord sur plusieurs points : les plaques tectonique sont composées de « cellules » (blocs carré) réparties sur un grille carrée et le système est discret : il n'y pas de transitions entre les positions des cellules ce qui réduit énormément la précision lors de la compression des cellules. De plus ces thèses indiquées rien sur les calcule utilisé pour la compression et la traction.

Partant avec presque rien et ne suivant pas le thèses, nous avons choisi un système bien différent. Nous avons décidés que le système ne serait pas discret permettant ainsi n'importe quelle configuration entre les cellules, de plus nous disposeront les cellules en « nid d'abeilles » (grille hexagonale) pour avoir au début de la simulation une équidistance entre les cellules. Ensuite nous utiliserons la « loi de Hook » pour la compression et traction entre les cellules, cette loi permet de savoir le coefficient de déformation d'une matière en fonction de la force et d'une constante définit dans le « module de Young ».

Pour commencer nous nous sommes dit qu'il fallait que chaque cellules connaissent toutes ses cellules adjacentes et que lorsque cette cellule recevrait une force lors d'une collision avec une cellule d'une autre plaque tectonique, elle la propagerait à toutes cellules adjacentes, qui, elles même la propageront à leur propre cellules adjacentes et ainsi de suite. Au première abord nous avons choisi un système de propagation récursif qui éviterait les double propagations sur les cellules qui ont déjà propagées leur force. Mais nous nous sommes rendu compte que cette propagation ne ressemblait a rien d'une onde, en effet la propagation faisait des tour autour du terrain de plus en plus petit. Pour résoudre se problème nous avons implémenté un système de front utilisant une liste pour stocker toutes les cellules sur front. A la fin de chaque propagation de la force des cellules dans le front vers les autres cellules, on demande aux cellules de front d'ajouter leur cellules adjacentes dans un nouveau front si elles non pas déjà propagées leur force, puis on recommence tout le calcul avec le nouveau front.

Étant très familier avec le code source d'un moteur de jeu nommé le « Blender Game Engine » et l'équipe étant d'accord j'ai alors commencer à programmer la simulation en Python avec ce moteur de jeu, mais nous nous sommes heurtés a un mur : le temps de calcule. En effet le simple calcule sur un bloque de 100 cellules prenait environs 5 secondes. Pour éviter cela nous avons changé de langage de programmation pour le C++, un langage plus bas niveau et donc pouvant être plus optimisé, de plus nous avons limité au maximum l'usage de listes (préférant une somme de vecteurs divisés qu'une liste de vecteurs) et les ajout doublon dans les listes.

Certes avec ces optimisations le calcul était 100 fois plus rapide mais nous étions toujours loin de nos exigences initiales.