GRAPHIQUE 3D

Mégane Bati & Paul Bouton & Thomas Loiodice

Ensimag 2A

Avril 2017

Spécificités de notre mini-jeu

Exécution et Pilotage du vaisseau

Pour lancer notre jeu, il suffit de compiler le programme et de lancer le main 1. Le pilotage du vaisseau s'effectue via les touches directionnelles du clavier. L'utilisateur ne peut pas monter ou descendre (le vaisseau suit automatiquement la courbure du terrain). En outre, lorsque le vaisseau tourne et possède une vitesse non nulle, ce dernier vrille dans la direction du virage.

Terrain

Le terrain a été implémenté en utilisant le bruit de Perlin. Une texture a ensuite été appliquée pour obtenir un effet de neige.
Nous obtenons ainsi des bosses assez lisses et d'autres plus abruptes : celles-ci sont visibles lorsque le terrain est très grand comme le montre la figure suivante :

Neige

Pour notre effet de neige, nous avons opté pour des chutes de neige. Ces dernières sont réalisées à l'aide de Keyframes appliquées à des cylindres simplifiés (nombre de faces réduit).

SkyBox

La skybox a été liée au vaisseau afin de le suivre tout au long de son déplacement. Cela présente l'avantage de toujours avoir un ciel visible ne sortant pas du champ de la camera.

Problèmes rencontrés et pistes d'amélioration

Nous avions d'abord créé des arbres beaucoup plus détaillés que ceux affichés dans le mini-jeu (ils sont disponibles en exécutant le main 4). Une explication plus détaillée est donnée en Annexe 1. Nous aurions aussi aimé ajouter de nouvelles fonctionnalités à notre vaisseau (déplacement en hauteur, tirs de boules de neige (traitées comme des particules par exemple) pour détruire la nature,…).

En ce qui concerne la gestion de la physique, nous avons choisi de modifier le système proposé et d'applique nos forces à des hierarchicalRenderables. Ainsi, pour chaque objet héritant de cette classe, nous pouvons gérer se physique et ses collisions (via le calcul de sphères de collisions semblables à des particules).

Annexe 1 : Réduction de la taille de l'arbre

Au début nous avions des arbres bien plus détaillés, composés de bouts de tronc, branches et épines, comme sur la figure suivante. Elle montre un arbre de longueur 15 avec des longueurs de branches aléatoires et "cagneuses" pour sembler plus naturelles, l'angle par rapport au sol est aussi aléatoire.

Le nombre de FPS étant trop faible (environ 15), nous avons eu plusieurs idées pour l'augmenter :

- quadrilatères au lieu de cylindres pour les aiguilles
- triangles au lieu de quadrilatères pour les aiguilles
- réduction de la hauteur du tronc
- réduction du nombre de divisions d'une branche
  et ajout de plus d'épines (pour que ça ne se voit pas trop)
- réduction du nombre de faces des cylindres

Les triangles représentant les épines ont été colorés en vert et passés au flatShader (au lieu de phongShader) pour éviter d'utiliser un matériau.
L'arbre est moins lourd mais toujours trop pour créer des forêts et un décor plus riche.
Il aurait été possible d'utiliser des segments avec une épaisseur pour simuler les cylindres et réduire largement le nombre de FPS. Nous n'avons pas implémenté cette solution, nous avons préféré simplifier l'architecture de nos arbres.

De plus, les arbres ne sont pas optimisés puisqu'une partie du dessin des branches se fait à l'intérieur du tronc et une partie de celui des épines se fait dans les branches. Ces parties cachées ne sont donc pas rentable et il faudrait translater les objets par rapport à leur parent. Cependant une amélioration de ce type n'aurait pas d'influence majeure sur le nombre de FPS.

En conclusion, après ces diverses réflexions et tentatives, les arbres restent toujours trop chers donc nous avons créé un nouveau type d'objets ArbreLegerRenderable. Un cône en bronze forme le tronc et des cônes en émeraudes forment les branches (leur nombre est disposition est plus ou moins aléatoire pour respecter certaines contraintes de la nature (pour éviter d'avoir des superpositions). Cette technique permet d'augmenter grandement le nombre de FPS associé à l'objet et donc de créer des forêts générées elles aussi aléatoirement. En effet, dans une grille carrée, les arbres sont disposés aléatoirement dans des sous-cases pour éviter que les arbres soient trop proches ou entre en collision.

Nous avons choisi de garder le phongShader et les matériaux Bronze et Emeraude pour ce nouveau type de sapin afin d'avoir de jolies couleurs à défaut de jolis arbres.
Nous aurions aimé implémenter une imitation de ParticleListRenderable pour générer une forêt à partir de quelques arbres aléatoires et augmenter ainsi les FPS.