Rapport Final

Répartition des rôles :

- Stanislas Nerot : Responsable environnement et Generation du monde par perlin noise
- Basile Arlandis : Responsable Agent et Rendu graphique

Langage choisi :

Java

Mode graphique choisi :

Java 3D, JavaOpenGL

Détails du Projet :

Pour le manuel d'installation et d'utilisation, se référer au fichier aide.txt

Stanislas Nerot :

- Perlin noise :

Importation de classes pour calculer un perlin noise dans la classe ImprovedNoise de LandscapeGenerator.
Multiples tentatives dans la classe PerlinNoiseLandscapeGenerator de LandscapeGenerator :
-additionner des bruits en diminuant l'amplitude et en incrémentant la fréquence ce qui ne marche pas avec un monde torique pour les effets de bord.
-quadrupler un quart du monde et le bruiter ce qui corrige les effets de bord mais donne un monde trop géométrique.
-génération avec des puissances des fonctions cosinus, sinus, et tangeante ce qui rends encore plus géométrique que la version d'avant.
-méthode trouvé sur un forum (dont je n'ai pas eu la présence d'esprit de noter le nom) qui nivelle plus le terrain qui est la fonction fBm qui fonctionne et qui est retenue.

- Environnement :

Pour des raisons de lisibilité, il y a un automate par type d'objet et par type d'éléments physique
de plus les états ont des numéros différents pour pouvoir se retrouver, il y a donc des modification dans les classes WorldOfTrees, World et Lanscape afin de tout afficher.

- arbres : mise en place d'un système de reproduction d'arbres s'ils sont en cendre par Math.random().
  États: 0 rien, 1 arbre, 2 arbre en feu, 3 arbre en cendre

Dessine les objets de la classe Tree.

- herbe : l'automate GrassCA consiste a faire comme les arbres mais mise en place de feu différent (les arbres peuvent mettre en feu l'herbe mais l'inverse n'est pas possible), utile pour les vaches comme sources de nourritures.
  États: 0 rien, 4 herbe, 5 herbe en feu, 6 herbe en cendre
  Dessine les objets de la classe Grass.

- lave : l'automate LavaCA initialise la lave sur la plus haute case du monde avec un compteur pour la plus haute case (car world.getmaxeverheigth() ne donnait pas les bonnes valeurs), puis a une chance d'être en irruption a l'initialisation et s'il n'est pas en irruption, retente d'être en irruption a chaque itération du step() et il ne sort jamais de son irruption après avoir débuté, de plus la lave s'écoule sur un voisinage de moore uniquement sur les cases plus basses s'il n'y a pas d'eau dessus (s'il y a de l'eau créé un état de roche qui ne fait rien entre l'eau et la lave).
  La lave brûle également l'herbe et les arbres.
  États: 0 rien, 7 lave, 8 roche
  Ne dessine pas d'objets.

- eau : l'automate EauCA initialise le monde avec des points d'eau aléatoires s'ils sont assez haut (pour faire des sortes de cascades) avec écoulement similaire a la lave.
  La décision a été prise de garder le feu d'arbres et d'herbe au dessus des cases aqueuses car théoriquement il s'agit d'un fine couche d'eau qui ne pourront donc pas éteindre un feuillage d'arbre en feu ou les brins d'herbes embrasés.
  Rencontre avec la lave identique a celle avec l'eau sur le sol.
  États: 0 rien, -1 eau
  Ne dessine pas d'objets.

- pluie : l'automate RainCA initialise la pluie comme l'irruption de la lave mais peut s'arrêter contrairement à elle, de plus elle éteins la lave, et empêche la propagation des feux
  États: 0 rien, 9 et 10 pluie (initialement pour faire 2 états pour un affichage plus propre et une alternance par case pour une animation).
  Dessine les objets de la classe Rain.

- foudre : l'automate ThunderCA n'instancie la fou!dre que s'il pleut, met la case en jaune et brûle arbre ou herbe dessus puis la case (l'affiche en grise) et un éclair un tombe pas deux fois (ou itérations) d'affilées au même endroit.
  États: 0 rien, 11 foudre et 12 case brûlée.
  Dessine les objets de la classe Thunder (pour l'instant inexistant).

- climat : l'automate Climat met un climat sur le monde en fonction du nombre d'itérations du monde pour faire un semblant de saisons et chaque saisons a deux états de température (s'il pleut ou non). (initialement prévu pour la gestion des agents)

Basile Arlandis :

- Rendu graphique :

Documentation sur la librairie OpenGL, modification des touches pour un déplacement plus aisé :

. touches directionnelles pour le déplacement

.Q/D maintenue pour une rotation à vitesse fixe

Suppression des scintillements en enlevant les Math.random() présents dans la génération des couleurs des objets.
Modification de la forme des arbres, et création graphique de l'herbe qui se distingue facilement des arbres.
Création graphique des Agents.
Tentative non aboutie de modéliser la pluie (problèmes rencontrés sur la transparence de l'eau et l'affichage graphique de gouttes d'eau).

- Agents :

Il y a trois différents agents :

- les Vaches :
  .Proie : aucune, est herbivore
  . Prédateurs : les Raptors et les Trex
  .Santé : de bonnes capacités de stockage de nourriture et d'eau
  .Détection : détection de la nourriture et de l'eau correcte, détection d'un prédateur assez réduite
  .Déplacement : déplacement lent, aucune endurance (ne peut courir)

- les Raptors:
  .Proies : les Vaches
  . Prédateurs : les Trex
  .Santé : faibles capacités de stockage de nourriture et d'eau
  .Détection : détection de la nourriture et de l'eau correcte, détection d'un prédateur correcte
  .Déplacement : déplacement rapide, endurance élevée

- les TRex :
  .Proies : tous les Agents (y compris les autres TRex)
  . Prédateurs : les Trex
  .Santé : très grandes capacités de stockage de nourriture et d'eau
  .Détection : détection de la nourriture et de l'eau élevée, détection d'un prédateur élevée
  .Déplacement : déplacement rapide, mais endurance réduite

Vie des Agents :

Vie individuelle :

Chaque Agent veille à se nourrir et boire pour ne pas mourir de faim ou de déshydratation. À chaque itération, ce-dernier perd légèrement des points de nourriture et d'eau qui, s'ils sont trop bas, lui font perdre des points de vie.
Pour se nourrir, un Agent doit se trouver sur la même case que sa nourriture/proie.
Pour boire, un Agent doit se trouver sur une case adjacente à l'eau.
Lorsqu'il pleut, tous les Agents peuvent boire, même s'ils ne se trouvent pas au bord d'un cours d'eau.
Si un Agent n'a ni faim ni soif, il se déplace aléatoirement sur la carte en suivant une direction globale.
Si un Agent perçoit un danger (écoulement de lave, feu de forêt ou autre prédateur), il change de direction pour s'en éloigner. La sécurité d'un Agent prime sur son

alimentation : tant qu'un danger sera présent l'Agent ne se nourrira pas et pourra potentiellement mourir de faim.
Si un Agent se retrouve au contact de la lave, il meurt instantanément. S'il se trouve au contact d'un feu de forêt, il perd des points de vie à chaque itération où il est dans l'incendie.
Si un éclair à lieu sur la case d'un Agent, ce dernier est foudroyé et meurt instantanément.

Interactions collectives

Les prédateurs chassent les proies, qui elles fuient les prédateurs. Lorsqu'un prédateur repère une proie, il se déplace plus rapidement si son endurance le lui permet, afin de pouvoir rattraper sa proie. Toutefois, si la chasse est trop longue, le prédateur s'épuise et la proie peut s'échapper.
Encore une fois la sécurité prime sur l'alimentation des Agents, si un prédateur se retrouve confronté à un danger (un de ses prédateurs apparaît, ou un feu est en vu), il abandonne sa chasse.