Sorbonne Université - Sciences

2i013 - Projet

Vie artificielle

Enzo Wesquy | Quentin Serreau
2018

Table des matières

1. Architecture et principes de base

a. Choix graphiques

Nous avons décidé d'utiliser OpenGL en 3D pour représenter notre monde en nous inspirant de l'exemple « World of Cells ». La navigation n'a pas beaucoup évolué, les touches Q et D permettent de faire tourner la caméra vers la gauche et la droite, les touches Z et S permettent de la faire monter et descendre et les flèches droites et gauche permettent une translation horizontale.

Les arbres sont représentés comment des triangles, les agents et les humains comme des cubes.

b. Architecture logicielle

Notre avons conçu une architecture permettant l'indépendance de la partie métier. En effet, les parties métier et graphique sont complètement dissociables ce qui permet de changer d'interface graphique sans tout reconstruire. Un autre avantage est que l'on peut lancer le programme sans interface graphique, ce qui permet lancer le programme sur un grand nombre d'itérations dans des délais plus courts, pour réaliser des statistiques par exemple.

Les parties métiers et graphique sont exécutées dans deux threads différents, la partie métier envoie périodiquement les informations nécessaires à l'affichage par le biais d'évènements. Un bloc de synchronisation était donc indispensable pour éviter de corrompre les données.

Tous les éléments de notre monde (humains, agents, forêt) ont été conçus comme des automates cellulaires à deux dimensions. Ils partagent tous le même tableau d'entier à deux dimensions représentant notre monde. Pour que chaque case puisse contenir plusieurs éléments (par exemple de l'herbe et un agent), nous avons utilisé des champs de bits. Cette solution a pour avantage d'être peu couteuse en mémoire et en temps de calcul mais ajoute une certaine complexité.

Absolument tous les automates sont mis à jour de manière asynchrone randomisée.

c. Configuration

Nous avons mis en place un système permettant d'enregistrer la configuration des automates dans un fichier XML. Ce fichier contient tous les paramètres pouvant altérer significativement l'exécution du programme. Ce choix d'implémentation a été motivé par la simplification de la configuration (il n'est plus nécessaire de modifier les constantes à plusieurs endroits différents dans les sources), de plus, cela évite la recompilation de tout le programme.

2. Monde et environnement

a. Génération de terrain et biomes

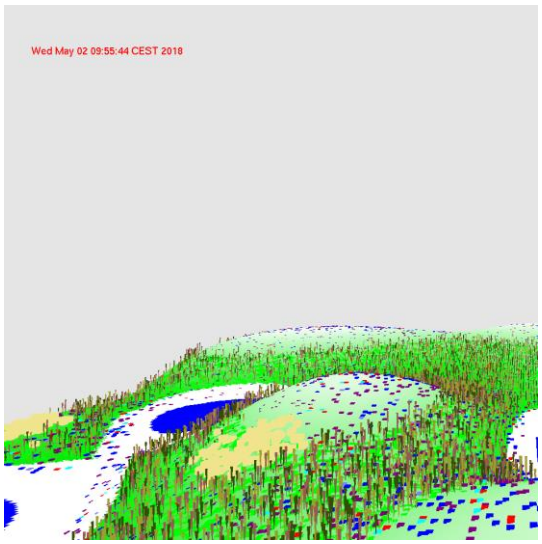
Pour créer un terrain aléatoire à chaque nouveau lancement, nous avons opté pour l'utilisation d'algorithme de génération procédurale. Nous nous sommes d'abord renseigné sur le bruit de Perlin pour finalement choisir le bruit de Simplex, générant des terrains plus uniformes. Nous avons utilisé l'implémentation du bruit de Simplex de Kurt Spencer que l'on peut trouver sur GitHub (https://gist.github.com/KdotJPG/b1270127455a94ac5d19).

Pour générer une nouvelle carte, le programme fait appel à une fonction génératrice utilisant le bruit de Simplex qui génère un tableau d'altitude. Ce dernier nous permet de placer les éléments de l'environnement : l'eau se trouvera dans les altitudes basses, le sable se trouvera autour de l'eau, les montagnes en hauteur, etc.

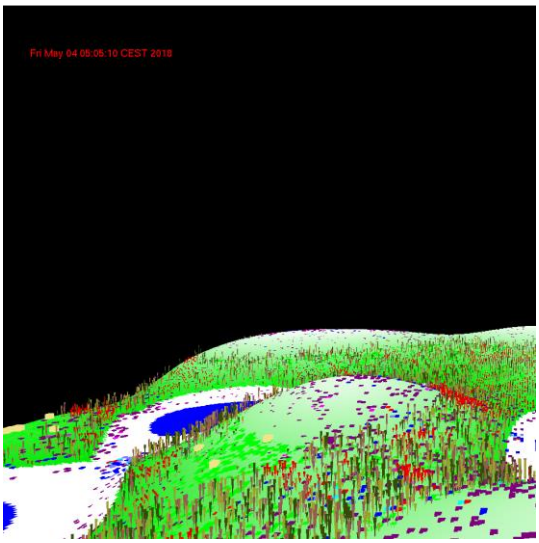
b. Saisons et temps

Lors du lancement du programme, une horloge est créée avec le temps actuel. Il est possible d'accélérer ou de ralentir le temps pendant l'exécution, il suffit d'appuyer respectivement sur les touches 1 et 2.

Nous avons implémenté un cycle jour/nuit. Il est représenté par un ciel noir pendant la nuit et blanc pendant la journée. Pour obtenir un cycle réaliste, nous avons récupérer les heures de lever et de coucher de Soleil à Paris de tous les jours de l'année 2018. Ces données sont stockées dans un fichier CSV chargé au démarrage.



Le monde pendant la journée



Le monde pendant la nuit

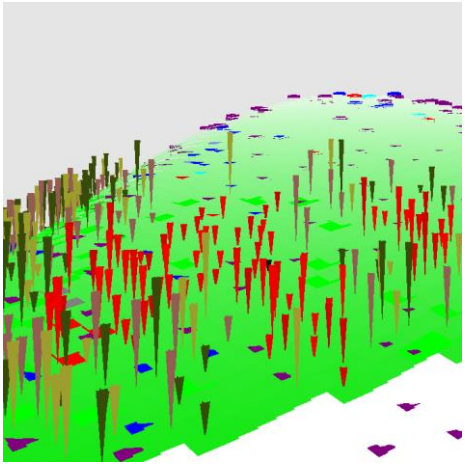
Les quatre saisons ont aussi été implémentées. Nous utilisons l'horloge pour déterminer la date actuelle et donc la saison. Le cycle jour/nuit s'appuyant sur des données réelles, les journées sont plus courtes en hiver qu'en été. Les saisons ont un impact sur l'environnement. Par exemple, la probabilité qu'un arbre prenne feu est plus forte en été qu'en hiver. Graphiquement, le changement de saison se traduit par un simple changement de couleur des éléments.

c. Forêt

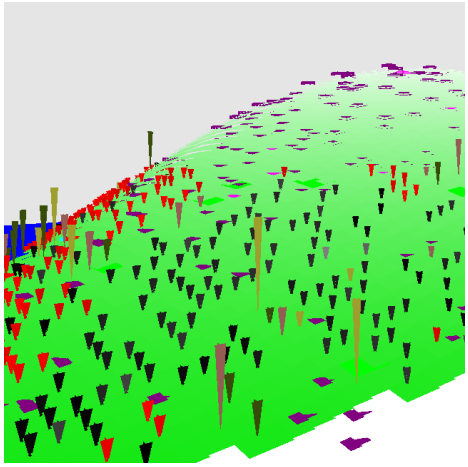
Comme dit précédemment, la forêt est modélisé grâce à un automate cellulaire. Les arbres poussent uniquement sur certaines zones définies lors de la génération de terrain. Ils peuvent grandir (jusqu'à un certain seuil), s'enflammer et être coupés par les humains. La forêt contient aussi des herbes qui peuvent être mangées par des proies (cf. 3.a. Type d'agents) ou s'enflammer ; elles ne grandissent pas.

A l'initialisation, un nombre d'arbres est placé sur la carte selon une densité définie dans le fichier de configuration ; leur taille est alors initialisée de manière aléatoire. Le même processus est effectué pour les herbes, hormis la taille. Une case « herbe » peut être superposée par un autre élément. Elle peut, par exemple, être traversée par un agent ou un humain. C'est le seul élément pouvant cohabiter avec un autre sur une même case.

Pour la modélisation du feu, nous avons utilisé un voisinage de Moore. En effet, pour savoir si un élément de la forêt va s'embraser (herbe ou arbre), nous faisons une moyenne pondérée de ses voisins en feu (les coefficients dépendent de la taille des arbres, une herbe compte pour 1) puis, la moyenne est comparée à un nombre aléatoire : plus la moyenne est élevée, plus les chances que



Début de feu



Dispersion de cendres

l'élément s'enflamme sont fortes. Un élément peut aussi s'enflammer spontanément mais les chances sont très faibles, elles dépendent de la saison.

Un feu peut aussi se propager au-delà du voisinage de Moore : en fonction de sa taille, l'arbre peut enflammer un autre arbre jusqu'à cinq cases plus loin. Ceci est valable uniquement pour les arbres. Enfin, la vitesse de combustion des arbres dépend de leur taille, les herbes brûlent en une itération. Tous les éléments de la forêt deviennent des cendres après avoir brûlé. Les cendres se dispersent au fur et à mesure des itérations. Graphiquement, les éléments en feu sont rouges, les cendres évoluent en nuances de gris, les arbres sont verts, orangés, marrons, blanc ou gris en fonction de la saison et les herbes sont verts vif.

3. Agents et interactions

a. Type d'agents

Pour les agents, nous avons choisi de ne prendre que deux types d'animaux aux caractéristiques indéterminés qui se trouvent être les proies (en bleu) et les prédateurs (en violet) avec des humains (en beige) à côté. Les agents peuvent se reproduire (avec une probabilité définie dans le fichier de configuration) s'ils sont assez proches et qu'ils ont mangé.

Tous les agents (proies, prédateurs et humains) ont des caractéristiques communes : ils ont une "barre de faim" qui leur donne un temps limité de vie sans nourriture, un arbre de décision qui détermine leurs actions à chaque tour et des interactions pouvant changer leurs états.

Pour ce qui est des proies, elles ont un comportement qui va de la recherche de nourriture (les proies sont herbivores et mange l'herbe qui poussent vite et partout) quand ils ont faim à la fuite quand un prédateur est à côté d'elle. En effet, les proies ont deux états : un état normal où sa préoccupation est de chercher de la nourriture et se reproduire, et un état de fuite où elle tente d'échapper à un prédateur en chasse, elles changent alors de couleur pour devenir cyan.

Au niveau des prédateurs le comportement est très similaire mais leur recherche de nourriture est plus fréquente et invoque la recherche de proies. Quand un prédateur rencontre une proie, il se mettent tous les deux en état "chasse" où la proie tente de s'échapper et où le prédateur la poursuit, il devient rouge. Il y a une probabilité que la proie s'épuise et cesse de fuir et une autre que le prédateur abandonne la poursuite.

b. Humains

Nous avons choisi d'implémenter des humains dans le but d'avoir des comportements plus complexes comme la création de bâtiments après avoir coupé un certain nombre d'arbres ou la chasse en groupe. Malheureusement, dû à un manque de temps et une architecture trop limitante, nous n'avons pu introduire que certains de ces aspects.

D'abord nous avons créé un système de tribu où les humains apparaissent en groupe autour d'un point central représentant leur base/point de départ. Ils bougent ensuite aléatoirement à la recherche de bois pour construire des bâtiments et tue les proies comme les prédateurs dès qu'ils sont en surnombre.

c. Interactions

Tous les agents ont une fonction leur permettant de rechercher leur nourriture dans leur environnement. Elle permet de regarder dans les quatre directions cardinales et de déterminer l'endroit avec le plus de nourriture. Pour les prédateurs cela équivaut à l'endroit contenant le plus de proies. Pour les proies, cela est légèrement plus compliqué, elle regarde dans les directions contenant

le plus d'herbes mais également celles qui n'ont peu ou pas de prédateurs à l'aide d'un système de score.

4. Annexe

a. Compilation et lancement

Pour compiler le projet sous Linux, il suffit simplement d'exécuter le script « build.sh ». Pour le lancer, il suffit d'exécuter le script « run.sh », si le projet n'est pas compilé, il le compilera puis lancera l'application.

b. Utilisation

Pour se déplacer dans le monde, utilisez les touches Q et D pour faire tourner la caméra horizontalement, les touches Z et S pour monter ou descendre la caméra et enfin, les flèches gauche et droite pour faire bouger la camera horizontalement. Vous pouvez utiliser les touches 1 et 2 pour augmenter ou diminuer la vitesse d'exécution.

Vous pourrez voir des humains regroupés entre eux, des agents se poursuivre ou se reproduire ou des arbres prendre feu.