Projet 1 - Compte Rendu

Jeu de la Vie

Par Jérémie Dumas et Julien Herrmann

Sommaire

I.Manuel d'utilisation.....1
II.Historique du développement.....2
III.Représentation, Structure de donnée.....2
IV.Bilan.....3

I. Manuel d'utilisation

- **Écran de Paramétrage**
  En plus du jeu de la vie "classique", il est possible de paramétrer différents éléments de l'automate tels que :

  - *Taille de la colonie* (10*10 par défaut)

L'utilisateur choisit les dimensions du tableau représentant les cellules de la colonie.

  - *Hauteur maximale de la colonie* (1 par défaut)

L'état des cellules n'est plus nécessairement un booléen (vivant ou mort) mais un entier correspondant à sa hauteur qui évoluera au fur et à mesure du jeu.

  - *Voisinage actif* (complet par défaut)

L'utilisateur peut choisir quels voisins seront vérifiés par le programme pour décider de l'évolution d'une cellule. On peut par exemple décider d'orienter l'évolution dans une direction particulière.

  - *Laisser un squelette* (désactivée par défaut)

Sur le modèle des récifs coralliens, si cette option est activée, chaque cellule meurt laissera un squelette. Sur une case occupée par un squelette, aucune cellule ne peut naître.

  - *Conditions de naissance* ({3} par défaut)

L'utilisateur choisit combien une cellule vide doit avoir d'éléments vivants dans son voisinage actif pour naître.

  - *Conditions de croissance* ($\varnothing$ par défaut)

L'utilisateur choisit combien une cellule de hauteur *h* doit avoir d'éléments de hauteur $h' \geqslant h$ dans son voisinage actif pour que sa hauteur augmente de 1.

  - *Condition de décroissance/mort* ({0,1,4,5,6,7,8} par défaut)

L'utilisateur choisit combien une cellule de hauteur *h* doit avoir d'éléments de hauteur $h' \geqslant h$ dans son voisinage actif pour que sa hauteur diminue de 1 ou (si *h*=1) meure. En cas de conflit avec la règle précédente, seule la règle de croissance est appliquée.

  - *Sauvegarder/Charger les paramètres*

Permet de sauvegarder les paramètres affichés afin de les charger ultérieurement.

- **Scène de jeu**
  - *Jouer/Stop*

Permet de démarrer/arrêter l'évolution de la colonie de façon continue en itérant chaque fois du nombre de pas défini par l'option "*Nombre de pas d'une itération*". La vitesse d'évolution est définie par l'option "*Vitesse du jeu*".

  - *Une étape*

Permet de visionner l'évolution de la colonie pas à pas. Un clic sur ce bouton fait évoluer la colonie un certain nombre de fois définie par l'option "*Nombre de pas d'une itération*".

  - *Sauvegarde/Chargement*

Permet de sauvegarder la grille actuelle dans un fichier ou d'en charger une, sous réserve de compatibilité des dimensions (X, Y et Z).

  - *Nombre de pas d'une itération*

Permet de définir le nombre de pas menées par la colonie lors de l'exécution d'une étape.

  - *Vitesse du jeu*

Définit la rapidité d'exécution du mode "*Jouer*". Correspond au nombre de mise à jour par seconde (FPS).

II. Historique du développement

Après avoir opté pour le "jeu de la vie" comme projet python, nous nous sommes mis d'accord sur la direction que prendrait notre application (choix des paramètres à inclure, organisation de l'interface et de la structure du programme, etc.). Nous nous réunissions régulièrement le mercredi afin de travailler de manière synchrone. Ainsi, sans avoir noté de date particulières, on peut cependant présenter les différentes étapes du développement de ce projet :

- Familiarisation avec python et, pour Julien, la programmation orientée objet d'une manière générale
- Écriture de la classe gérant la colonie de cellule avec l'algorithme naïf (tableau bidimensionnel) pour Jérémie.
- Apprentissage progressif des options fournies par Tkinter, module utilisé pour l'interface graphique
- Julien s'occupe de la fenêtre des paramètres, il a plusieurs objets à afficher :
  - ✗ Réglettes pour la taille de la colonie
  - ✗ Boutons à cocher pour les conditions d'évolution
  - ✗ Boutons-radio pour d'autres options
  - ✗ Mise à jour des boutons en fonction des paramètres, remise à défaut …
- Jérémie s'occupe de tracer la colonie à l'écran (utilisation d'un canevas) :
  - ✗ La mise à jour du canevas est lente, ce qui n'aide pas à la vitesse de l'exécution.
  - ✗ On programme les boutons qui permettent d'effacer la grille, d'en générer une aléatoirement
  - ✗ Gestion du clic de la souris pour dessiner des cellules
  - ✗ Boutons Lecture/Pause, mise à jour étape par étape
- On regroupe les différentes parties du programme. Unification des notations / classes / variables utilisées
- Jérémie ajoute également la possibilité de charger/sauver des paramètres ou des grilles (avec le module pickle)
- Afin de figurer des hauteurs différentes, ajout d'un semblant de vue en perspective, paramétrable (inclinaison). Ajout aussi d'une coloration des cellules en fonction de leur hauteur (utile surtout en vue plane).
- Julien cherche à implémenter des info-bulles, mais cela nécessite l'utilisation d'un nouveau module à installer (Pmw), donc on choisit de s'en passer.
- Rédaction du manuel en première partie à défaut des info-bulles.

III. Représentation, Structure de donnée

Le choix de la structure de donnée pour représenter la colonie de cellule a été un tableau bidimensionnel stockant l'état de chaque case de la colonie. Ce n'est pas forcément la meilleure solution en terme de performance, mais comme c'est la première fois que nous faisons du python tous les deux, il nous a paru difficilement envisageable de travailler sur deux fronts en parallèle (efficacité de l'algorithme et construction en python). En pratique on utilise même deux tableaux pour faire la mise à jour de chaque cellule correctement.

Le jeu de la vie de Conway a aussi été étendu de différentes manières. Les nouvelles conditions d'évolution ne permettent pas une paramétrisation exhaustive de tous les automates cellulaires à n états (si n est la hauteur de notre colonie), laquelle serait combinatoire, mais constituent un compromis acceptable.

Il est possible également, comme le suggère le modèle des récifs coralliens, de laisser un squelette si une cellule meurt. On peut aussi favoriser une direction de propagation en cochant les voisins "actifs" d'une cellule (et reproduire par exemple l'automate "Déplacement vers l'Est").

L'idée des récifs coralliens nous a fait envisager la création de cellules à plusieurs hauteurs, car un récif de corail est généralement une structure en 3D. De fait, afin de ne pas trop complexifier le problème, on s'est limité à des règles d'évolution simples (cellules voisines hauteur supérieure par exemple).

L'ajout d'une dimension au problème nécessitait aussi l'implémentation d'une vue en perspective. Celle-ci est relativement basique (pas de sinus ou autres trucs du genre), mais permet de se faire une bonne idée de l'aspect de la colonie. Il faut noter toutefois que la vue en perspective a un impact sur les performances, car elle entraine plus de calculs. Se reporter au paragraphe suivant pour plus de détails.

IV. Bilan

D'un point de vu technique, l'utilisation d'un tableau bidimensionnel ne permet pas la gestion de colonie de grandes tailles (comme Golly qui utilise des tables de hachage). On se limite donc aux grilles de 100*100. La mise à jour du canevas par Tkinter est aussi une opération qui prend du temps, plus que de parcourir le tableau de la colonie lors de la mise à jour. Enfin un troisième facteur de ralentissement quand on utilise la vue en perspective vient des calculs des transformations des coordonnées des coins des cellules.

De fait (et cela paraît logique), plus la grille sera grande, plus le jeu risque de subir de ralentissements, surtout si on la visionne en 3D.

L'application est fournie avec quelques fichiers de configuration de base, ainsi que des grilles montrant quelques figures connus (pour le jeu de la vie notamment). Il faudrait plus de temps pour en exhiber d'autre, avec des règles personnalisée. Le fonctionnement actuel du programme permet l'utilisation d'une même grille pour des règles différentes, on peut donc chercher à voir comment évolue certaines figures connues du jeu de la vie avec d'autres règles.

Le code fourni est également commenté et expliqué, on pourra donc s'y référer pour plus de détails sur son fonctionnement (même s'il n'est pas détaillé ligne par ligne).