ROMON Pierre
BOCQUET Anthony
SR01

TP3
Programmation Python

Table des matières

Le but de l'exercice est de créer une interface graphique pour le jeu de la vie. Cette interface permet le suivi des évolutions des cellules ainsi que certaines manipulations sur les paramètres propres au jeu de la vie.

Interface graphique

On utilise ici la bibliothèque graphique *tkinter*. Ce module de base intégré à Python permet la création d'interface graphique très variées et portables sur la majorité des systèmes d'exploitation. *tkinter* contient de nombreux widgets à personnaliser pour mettre en place les différents éléments graphiques de notre fenêtre. La première chose à faire est d'importer la totalité du module.

Constructeur

On définit ensuite un objet *Interface* héritant de *Frame* afin de mettre en place les différents éléments graphiques de la fenêtre. Tous ces éléments graphiques sont initialisés dans le constructeur de la classe.

On commence par créer 2 frames afin de compartimenter notre fenêtre en 2 parties distinctes : la partie principale *main* (à gauche) qui accueillera le damier du jeu de la vie et la partie latérale *side* (à droite) qui contiendra les différents widgets permettant de manipuler le déroulement du jeu de la vie.

La barre latérale de l'interface contient 4 différents boutons : un pour lancer le jeu de la vie, un pour l'arrêter, un autre pour l'initialiser et un dernier pour quitter l'interface. Pour définir graphiquement ces différents boutons, on utilise le widget *Button*. Ce widget possède différentes options personnalisables intéressantes comme *label* (intitulé du bouton) et *command* (la fonction à exécuter lorsque le bouton est pressé). On affecte donc à ces boutons les fonctions *cycle*, *stop* et *init* qui permettent la manipulation du jeu de la vie. Ces fonctions seront définies et explicitées plus tard. Il y a également dans cette barre latérale 3 curseur qui permettent de gérer la taille du damier, le nombre de cellules vivantes et la vitesse d'évolution des cellules. On crée donc ces curseurs avec le widget *Scale*. On utilise ici les options *from_* (valeur minimale), *to* (valeur maximale), *orient* (orientation du curseur) et *label* (intitulé du curseur). Les valeurs, et donc positions, de ces curseurs peut être récupérées avec la fonction *get* ou établies avec la fonction *set*. On utilise d'ailleurs *set* pour affecter des valeurs par défaut au curseur.

Il ne reste plus qu'à gérer le frame propre au damier à l'aide du widget *Canvas*. Avant l'initialisation, celui-ci doit être vide, on ne fait donc rien de plus que de préciser les dimensions de ce canevas *canDim*.

Fonction affichage

La fonction *affichage* utilise la méthode *create_rectangle* issu de *Canvas* pour créer des cellules carrées en parcourant la matrice. La cellule prend la couleur rouge ou blanche en fonction de sa valeur et donc de son état. Afin de remplir correctement le frame *main*, on fait varier la taille des cellules en fonction de la taille de canevas *canDim* et de la taille du damier récupéré depuis le curseur *taille*.

Jeu de la vie

On importe ici un autre module : *random* pour la génération aléatoire. On définit une cellule vivante par la valeur 1 et une cellule morte par la valeur 0. La matrice contenant les cellules est modifiée au fil de l'évolution. À chaque nouveau stade d'évolution, on appelle la fonction *affichage* pour modifier le damier graphique.

Fonction bernoulli

Cette fonction prend en paramètre le pourcentage de cellule vivante à l'initialisation, paramètre récupéré grâce au curseur *vie*. Ce pourcentage est interprété comme une probabilité qu'on utilise afin de calculer la probabilité de l'état d'une cellule à l'aide d'une loi de Bernoulli.

Fonction initialisation

Tout d'abord, on crée une matrice nulle possédant les dimensions maximales du damier. Le nettoyage du damier est ainsi facile en appelant la fonction *affichage*. On récupère ensuite la dimension voulu par l'utilisateur grâce au curseur et on génère une matrice de cette dimension dont les composantes sont, aléatoirement grâce à *randrange*, 0 ou 1. On n'oublie pas de compter les cellules vivantes afin d'actualiser le curseur gérant le pourcentage de cellule vivante.

Fonction nbVoisin

Cette fonction prend en paramètre les coordonnées *i* et *j* d'une cellule afin d'en calculer les voisins. Pour cela, on inspecte les cellules adjacentes et on incrémente le compteur dans le cas où elles sont vivantes. Étant dans une matrice torique, on traite différemment les cas de bordure de matrice.

Fonction evol

Ici, une matrice temporaire est nécessaire pour le stockage des nouvelles valeurs de la matrice. Pour chaque cellule, on calcule son nombre de voisin avec l'appel de la fonction *nbVoisin*. On applique ensuite les différentes règles du jeu de la vie afin de déterminer la mort, la naissance ou la survie d'une cellule en fonction de son nombre de voisin. Ici aussi, on compte le nombre de cellule vivante à l'issu de l'évolution afin d'actualiser le curseur donnant le pourcentage de cellule vivante.

Fonction cycle

Cette fonction permet, tant que le jeu de la vie est en cours (*running* = *True*), de faire subir des évolutions à la matrice grâce à la fonction *evol*. Ces évolutions sont espacées par un laps de temps défini par la vitesse d'évolution de la matrice que l'utilisateur a pu définir grâce au curseur. Pour cela, on utilise la méthode *after* associée avec une récurrence sur *cycle*.

Fonction stop

L'arrêt du jeu de la vie se fait tout simplement en changeant la variable *running* à *False*. Cette variable changé, la fonction *cycle* ne peut plus appeler la fonction *evol*.

Exécution

On crée tout d'abord la fenêtre avec la fonction *Tk*. On change ensuite le nom et l'icône de cette fenêtre grâce aux fonctions *title* et *iconbitmap*. On crée un objet *Interface* et on lance la boucle infinie propre à *tkinter* avec *mainloop*.