Propagation du bâillement chez le porc - Manuel utilisateur

Gardette Auguste, Delauney Théo

A. Packages utilisés

tkinter : Bibliothèque graphique libre d'origine de Python. Permet la création d'interfaces graphiques.
math : Fournit l'accès à un panel de fonctions mathématiques.
random : Implémente différents générateurs de nombres pseudo-aléatoires.

B. Fonctionnalités développées

1. Création d'une classe cochon

Chaque cochon sera considéré comme un objet défini par son sexe, son age, sa forme et ses coordonnées. Il peut réaliser un ensemble de méthodes privées et une méthode publique.

a. Méthodes Privées

__direction(self) : Permet de proposer un déplacement DX, DY d' un mètre (10 pixels) tout autour de l'objet. Cela est permis grâce à la détermination aléatoire d'un angle en utilisant les modules random.uniform et math. cos/sin.

__distance(self, cochon2) : Création d'un attribut longueur calculant la distance entre le bord du cochon considéré s' il se déplace de DX, DY et le bord d'un second cochon. La longueur peut renvoyer une valeur négative si le déplacement implique une superposition des deux objets.

__control_parois(self) : Méthode de contrôle des extrémités du canevas. Si le déplacement DX, DY implique de sortir du canevas alors on redéfinit DX ou DY à 0 pour empêcher la sortie.

__baillement_spontané(self) : Déclenchement d'un bâillement spontané pour un cochon considéré, selon une probabilité d'un pour mille, si celui-ci n'est pas protégé. Modifie la couleur du rond en rouge pour le cochon considéré et initialise un décompte à 3 correspondant à la durée de la protection du cochon pour un futur bâillement.

__baillement_transmission(self) : Déclenchement d'un bâillement pour un cochon considéré, selon la probabilité "proba" de se le voir transmis et si son décompte est à 0, c'est-à-dire si le cochon n'est pas protégé. Si les conditions sont vérifiées alors le rond du cochon est colorié en rouge et le décompte est initialisé à 3.

__baillement_protection(self) : Si le décompte est supérieur à 0 alors cela signifie que le cochon considéré a déjà baillé ou qu'il est déjà protégé, alors le rond de ce cochon restera orange et le décompte sera décrémenté. Si le décompte est nul, le cochon redevient vert.

__probabilité(self,cochon2) : Si le second cochon est rouge alors on ajoute à l'attribut "proba" la probabilité que ce cochon transmette au cochon considéré le bâillement selon son sexe et sa distance .

b. Méthode mouvement (publique)

Méthode faisant appel à l'ensemble des méthodes privées permettant de faire déplacer le cochon à chaque itération et de le faire bailler si les conditions sont réunies.
Si la variable stop est égale à 1 cela permet d'interrompre la fonction after répétant la méthode mouvement. Si le cochon n'était pas en pause avant (DX,DY =0) alors il garde la direction précédente, sinon il prend une nouvelle valeur pour DX,DY.
Une boucle for permet de :

- Vérifier que le déplacement de DX, DY n'entre pas en conflit avec les autres cochons sinon le cochon marque une pause.
- Calculer la probabilité de se voir transmettre le bâillement.

L'âge du cochon considéré est ensuite pris en compte dans le calcul de la probabilité de bailler et le cochon et déplacé d'une valeur DX, DY.

2. Fonctions

getScale() : Permet de récupérer la valeur de la barre de scroll, et donc le nombre de cochons choisi par l'utilisateur, pour être utilisé en tant que variable.

pause() : Fonction appelée par le bouton Start/Stop. Quand la variable stop est égale à 1, les cochons ne bougent pas. Ils bougent quand la variable stop est nulle. Cette fonction donne à la variable stop la valeur 1 quand elle est nulle et inversement.

generateur_de_cochon() : Le jeu est mis en pause. La fonction utilise le diamètre des cochons, la taille du canvas et le nombre de cochons récupéré par getScale pour générer une grille de cochon à intervalle régulier sur le canvas. Ajoute chaque individu dans une liste ensemble_cochon. Une fois les cochons créés, la fonction lance les mouvements.

3. Main

On initialise différentes variables. Elles sont globalisées pour être utilisées dans les fonctions et les méthodes. Sont initialisées dans l'ordre :

- la variable stop, qui prendra la valeur de 0 (start) ou 1 (pause)
- Taille_canva, la taille du canvas dans la fenêtre
- diametre et rayon, des caractéristiques du cochon
- dico_age, dictionnaire où chaque âge possible pour un cochon est associé à une probabilité de transmettre le bâillement à un autre cochon
- fen, la fenêtre
- can, un canvas de fond blanc et de taille dépendant de Taille_canva placé dans fen
- value, conteneur mémorisant un flottant (valeur par défaut 0.0)

On initialise trois boutons dans la fenêtre, placés en bas de celle-ci :

- scale, bouton de scroll horizontal allant de 1 à 64 et dont la valeur est stockée dans value
- bouton_pause, bouton-pressoir “Start/Pause” lié à la fonction pause
- bouton_generateur, bouton-pressoir “Création” lié au générateur de cochons

On initialise aussi deux labels dans la fenêtre, placés en haut de celle-ci.

C. Instruction pour utilisateur

1. Presser le bouton de scroll et le faire glisser horizontalement. Ce bouton qui va de 1 à 64 permet de renseigner le nombre de cochons que l’utilisateur veut générer.
2. Cliquer sur le bouton “Création”. Une grille de cochons de l’effectif choisi devrait apparaître.
3. Cliquer sur “Start/Pause” pour lancer la simulation.

Chaque rond est un cochon, et le canevas est la porcherie. Les cochons verts sont des cochons qui ne baillent pas. Les cochons ont une certaine probabilité de bailler spontanément. Lorsqu’un cochon est à proximité d’un cochon qui baille, il a une certaine probabilité de bailler en retour. Un cochon qui baille est un cochon de couleur rouge. Après avoir baillé, un cochon a un certain temps de repos avant de pouvoir rebailler. Un cochon en repos est un cochon de couleur orange.

4. Cliquer sur “Start/Pause” pour interrompre la simulation.
5. Recommencer depuis l’étape 1 pour reconfigurer le nombre de cochons.