Mini-Projet 1 : Un jeu en Python

1 L'objectif

Reproduire et s'approprier un jeu existant sur internet (voir http ://www.jeux.fr/jeu/fishy).

Ce jeu se compose d'un personnage que l'on déplace avec les touches fléchées qui doit manger les ennemis plus petit que lui pour grandir et éviter de se faire manger par les plus grand. L'objectif pour gagner est d'arriver à une taille fixée, et l'on perd lorsqu'on n'a plus de vie.

2 Cahier des charges

- Ce jeu doit comporter un personnage qui ne doit pas sortir de l'écran en haut et en bas. De plus, si on va trop à gauche ou trop à droite on se retrouve de l'autre côté de l'écran.
- Ce jeu doit comporter quatre ennemis qui se déplacent horizontalement à des vitesses différentes. De plus, on affichera les ennemis que l'on peut manger en vert et ce que l'on ne peut pas manger en rouge.
- Ce jeu doit comporter un bonus qui a une position fixe et qui fait gagner une vie lorsqu'on l'attrape.
- Ce jeu doit comporter une régénération des ennemis mangés et des ennemis qui sont sortis de l'écran.
- Ce jeu doit comporter un affichage lorsqu'on a perdu ou gagné.
- Ce jeu doit comporter deux boutons. L'un pour quitter le jeu, l'autre pour lancer ou recommencer une nouvelle partie lorsqu'on a perdu ou gagné.

3 Amélioration future

- Rendre modifiable le nombre d'ennemi.
- Changer les boules par des images de poissons et rajouter un décor d'aquarium derrière.
- Rajouter un compteur de score.
- Rajouter un compteur de temps
- Rajouter d'autres bonus/malus
- ...

4 Analyse

Pour les besoins de ce jeu, on utilisera trois modules. Le module « tkinter » pour le graphisme, la fonction « randint » du module « random » pour l'aléatoire et la fonction « time » du module « time » pour la gestion du temps.

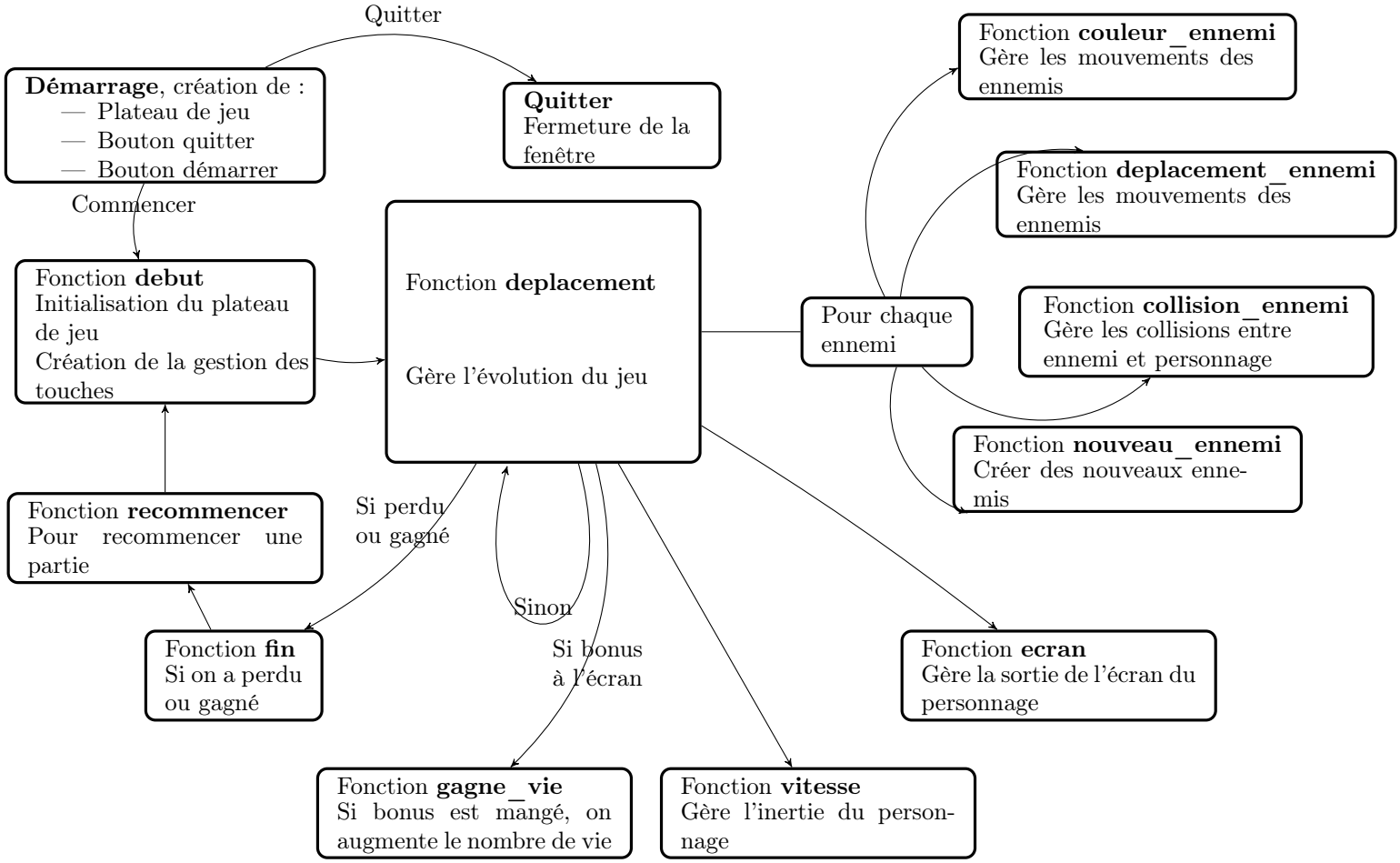
On utilisera les variables suivantes :

- ◇ deux entiers `centre_x` et `centre_y` pour le centre et un entier `rayon` pour le rayon.
- ◇ deux entiers `pas_x` et `pas_y` qui évoluent en fonction de la touche de direction utilisée et trois entiers `vit_x`, `vit_y` et `tps` qui gerent la vitesse du personnage.
- ◇ un entier `vie` pour le nombre de vie restante, un réel `debut` pour savoir depuis quand un bonus est ou n'est pas à l'écran et un booléen `bonus` pour savoir si le bonus est ou pas à l'écran.
- ◇ un réel `intouchable` pour savoir si on vient de perdre une vie et si oui, depuis quand
- ◇ une liste contenant les quatre ennemis, chaque ennemi étant constitué à partir d'une liste contenant (dans l'ordre) les données de l'abscisse du centre, de son ordonnée, du rayon, de la vitesse de déplacement et de l'objet graphique.
  REMARQUE : `ennemis[2]` permettra d'avoir toutes les informations du troisième ennemi.
  En enregistrant cet ennemi dans une nouvelle variable (`ennemi = ennemis[2]`), alors on peut accéder à la vitesse de déplacement avec `ennemi[3]` par exemple.

On utilisera également les objets tkinter suivants :

- ◇ un bouton `quitter`.
- ◇ un bouton `commencer`.
- ◇ un texte qui affiche le nombre de vie.
- ◇ un canvas contenant les informations du plateau de jeu :
  - le personnage.
  - les quatre ennemis (dans la liste des ennemis).
  - le bonus (pour la vie supplémentaire).

L'organisation du jeu se déroulera avec l'organigramme suivant :



5 Description des différentes fonctions

◇ *La phase de création :* (ce n'est pas une fonction à proprement parler)
Il faut contruire la fenêtre tkinter et y placer les boutons (dont le bouton *commencer* qui lance la fonction *debut*), le texte et le canvas.
Le fonctionnement se fera à l'aide de la boucle infinie `mainloop`.

◇ *debut()*
On initialise toutes les variables pour bien débuter.
On crée les ennemis de manière alétoire.
On active les touches fléchées.
On passe à la fonction *deplacement*

◇ *deplacement()* : c'est la fonction principale
On gère le timer pour l'affichage ou non du bonus. Le bonus s'affichera au bout de 7 secondes sans bonus et restera à l'écran pendant 4 secondes.

- On exécute la fonction *vitesse()*
  Si une touche est appuyée (avec la fonction *appuie_ touche(event)*) et si ça fait cinq fois que cette fonction est exécutée alors on augmente la vitesse.
  Si une touche est relâchée (avec la fonction *relache_ touche(event)*) et si ça fait cinq fois que cette fonction est exécutée alors on baisse la vitesse.
  On vérifie que la vitesse maximale n'est pas atteinte.
  On remet au début le compteur d'exécution si on a dépassé 5 exécutions.

On modifie les coordonnées du centre avec cette nouvelle vitesse.

- On exécute la fonction *ecran()*
  On vérifie que le personnage ne sort pas de l'écran en haut et en bas.
  On vérifie que si on sort à gauche, on rentre par la droite et réciproquement.

On actualise la position du personnage.

- Lorsqu'un bonus est à l'écran, on exécute la fonction *gagne_ vie(attrape_ moi, centre_ x, centre_ y, rayon)*
  On vérifie que le bonus est mangé ou pas, et si oui on le supprime de l'écran et on augmente le nombre de vie.

On vérifie si on est en mode invincible lorsqu'on vient de perdre une vie, et si oui on active le clignotement.
On stoppe l'invincibilité au bout d'une seconde après avoir perdu une vie.

Pour chaque ennemi :

- On exécute la fonction *couleur_ ennemi(ennemis[i], rayon)*
  On vérifie que l'ennemi est plus petit, et si oui on change la couleur de l'ennemi en vert.
- On exécute la fonction *deplacement_ ennemi(ennemis[i])*
  On modifie l'abscisse de l'ennemi en fonction de sa vitesse de déplacement.
  On actualise la position de l'ennemi.
- On exécute la fonction *collision_ ennemi(ennemis[i], centre_ x, centre_ y)*
  On vérifie si on rentre en contact avec l'ennemi, et si oui on regarde si on est mangé ou pas.
  Si on mange l'ennemi, on déplace l'ennemi à l'abscisse -500 et on augmente de 2 le rayon du personnage.
  Si on est mangé par l'ennemi, on vérifie qu'on n'est pas invincible. Si on est pas invincible, on date le moment où on vient de se faire manger, on perd une vie, et on modifie l'affichage de la vie.
- On exécute la fonction *nouveau_ ennemi(ennemis[i], rayon)*
  Si l'abscisse de l'ennemi est trop à gauche ou à droite de l'écran, on recrée un nouvel ennemi aléatoirement.

Fin de la boucle pour les ennemis.

On vérifie si on a gagné (le rayon est plus grand que 100) ou perdu (plus de vie). Si oui, on passe à la fonction *fin*.
Si non, après 10 millisecondes, on recommence la fonction *deplacement*

◇ *fin()*
On vérifie si on a perdu ou gagné, et on affiche un message adéquat.
Après 3 secondes, on passe à la fonction *recommencer*

◇ *recommencer()*
On efface tout le plateau de jeu.
On réactive le bouton *commencer*.

6 Exercices

Exercice 1 :

Compléter, en lieu et place de `pass`, dans la fonction `gagne_vie` un script qui permet (en utilisant uniquement les variables données) de gagner une vie lorsqu'on attrape le bonus.
Voir la description plus haut.

Exercice 2 :

Compléter, en lieu et place de `pass`, dans la fonction `ecran` un script qui permet (en utilisant uniquement les variables données) de ne pas sortir de l'écran ou suivant les conditions énoncées.
Voir la description plus haut.

Exercice 3 :

Compléter, en lieu et place de `pass`, dans la fonction `deplacement_ennemi` un script qui permet (en utilisant uniquement les variables données) de déplacer l'ennemi passé en argument.
Voir la description plus haut.

Exercice 4 :

Compléter, en lieu et place de `pass`, dans la fonction `couleur_ennemi` un script qui permet (en utilisant uniquement les variables données) de passer en vert la couleur de l'ennemi, passé en argument, s'il est plus petit que le personnage.
Voir la description plus haut.

Remarque : On pourra s'aider du clignotement pour l'instruction de changement de couleur (voir ligne 70 ou 72).

Exercice 5 :

Compléter, en lieu et place de `pass`, dans la fonction `nouveau_ennemi` un script qui permet (en utilisant uniquement les variables données) de recréer un nouvel ennemi aléatoirement comme lors de l'initialisation, si celui passé en argument est en dehors de l'écran.
Voir la description plus haut.

Remarque : On pourra, à la ligne 29, remplacer `x = randint(0, 1) * 800` par `x = -500`, `x = 600` ou `x = 1500` par exemple.
Ainsi on pourra vérifier si les ennemis en dehors sont recréés.

Exercice 6 :

Compléter, en lieu et place de `pass`, dans la fonction `collision_ennemi` un script qui permet (en utilisant uniquement les variables données) de tester le contact avec l'ennemi passé en argument.
En fonction de la taille, effectuer les modifications nécessaires.
Voir la description plus haut.

Remarque : On pourra, à la ligne 19, passer le nombre de vie intiale à 2 ou 3 par exemple.
Ainsi on pourra vérifier si le clignotement dû à l'invincibilité est correct.