Christopher, Nareg, Nam

Rapport de semestrielle

1.But: **Nareg**

L'objectif de notre projet était de créer un jeu PONG simplifié qui peut se jouer tout seul. Nous avons imaginé un PONG où le but est de faire rebondir la balle le plus de fois possible sur la raquette. Le programme permet de lancer le jeu, de le mettre sur pause, de l'arrêter et de le relancer grâce à des boutons positionnés sur les bords du terrain. Nous avons affiché un score qui évolue en temps réel avec les rebonds de la balle. Le programme permet le déplacement aléatoire de la balle et laisse l'opportunité à l'utilisateur de déplacer la raquette à la verticale sur l'interface que nous avons créé. Nous avons utilisé Tkinter pour écrire le programme de ce jeu.

2.Répartition du travail: **Nareg**

Nous avons essayé de nous partager le travail le plus équitablement possible. Nous avons tous participé à la conception du code sur le site replit et à son explication. Nous avons essayé de donner un programme des plus complets possible. L'oral a été réparti sur les différents points clés du rapport entre nous.

3.Code: **Christopher**

Structure générale:

Nous avons essayé de découper notre programme en différentes parties distinctes pour aider à la compréhension de celui-ci. Nous avons séparé le code en 3 parties. La première partie contient l'initialisation de l'interface du terrain. Nous avons importé pour ce faire le module Tkinter qui nous permet de créer l'interface graphique de notre jeu, le module colorama qui permet de mettre en couleur le texte affiché sur la console pour être plus esthétique et le module os qui nous a été utile pour pouvoir relancer le programme directement depuis la console sans avoir à le redémarrer. Nous avons aussi ajouté des conseils utiles aux utilisateurs pour que le jeu soit le plus simple possible à utiliser.
Ensuite, dans la deuxième partie nous retrouvons les fonctions. C'est une partie très importante, car c'est grâce à ces fonctions que le jeu va pouvoir fonctionner. Il y a en l'occurrence les fonctions pour le déplacement de la balle et de la raquette ou entre autres pour les différents boutons qui vont permettre à l'utilisateur de pouvoir réaliser plusieurs actions telles qu'arrêter, démarrer, relancer et mettre sur pause le jeu. Ces fonctions seront utilisées plus tard dans le code.
Finalement, nous avons la partie principale du code qui permet de créer les objets dont nous avons besoin pour le jeu tel que la balle, la raquette, la couleur ou même les lignes du terrain. Cette partie sert à mettre en place l'interface graphique. Nous avons aussi écrit des bouts de codes qui permettent de vérifier l'entrée des coordonnées de déplacement et de la couleur du terrain pour éviter tout bug dans le programme. Les fonctions sont appelées afin de pouvoir les utiliser pour les boutons ou le déplacement de la balle et de la raquette. Pour finir, il y a encore un bout de code qui permet de donner des instructions générales à l'utilisateur quand aux touches utiles pour pouvoir jouer. Cela permet de conseiller et d'aider l'utilisateur.

Usage de variables: **Christopher**
Comme nous pouvons le voir, nous avons utilisé des variables globales. Pour cela nous avons mis dans le code la fonction “global” qui permet de changer une variable en une variable globale. Nous étions obligés d’utiliser des variables globales car sinon notre programme ne marchait pas. Il fallait que la variable “flag” par exemple varie en même temps dans plusieurs fonctions pour que le programme puisse marcher. Cette variable peut s’étendre à tout le programme et nous simplifie la tâche. Nous avons donc privilégié les variables globales aux variables locales même si l’on sait que les variables locales permettent d’éviter de multiples bugs.

Explication de chaque fonction:

a) fonction déplacement: **Christopher**
Nous allons vous parler de la fonction déplacement qui est la partie la plus importante de notre programme. Cette fonction comme nous pouvons le constater permet le rebond de la balle contre les parois et la raquette ainsi que son déplacement dans l’interface. La fonction déplacement permet aussi l’augmentation du score au fur et à mesure des rebonds de la balle sur la raquette.
variable global: dx = vitesse de déplacement sur l’axe des x.
variable global: dy = vitesse de déplacement sur l’axe des y.

rebond mur: **Nam**
Nous avons dans un premier temps mis des conditions quant au limite du terrain pour que la balle puisse changer de direction lorsqu’elle arrive à la coordonnée du bord du terrain en fonction de l’axe x (horizontale) et y (verticale). La coordonné de la balle doit toujours être plus grande ou plus petite que les coordonnées du mur. Nous pouvons apercevoir le game over qui a été inséré pour le mur de gauche derrière la raquette donc à la place de rebondir, un game over s’affiche sur l’écran. La balle est un carré et il me semble que c'est une liste avec 4 côtés qui sont: gauche: coord 0; haut: coord 1; droite: coord 2; bas: coord 3. C’est grâce à cela que l’on arrive à créer les rebonds de la balle.

rebond raquette: **Nam**
Ensuite vient le rebond de la balle sur la raquette. Nous avons utilisé une méthode avec les coordonnées de la balle et de la raquette qui va permettre à la balle de détecter quand elle est sur les coordonnées de la raquette pour partir après dans l’autre sens. Nous avons fait cela pour les côtés haut et bas de la raquette pour améliorer notre programme afin d’éviter tout bug.

compteur: **Nam**
Comme nous pouvons le voir, il va y avoir une variable compteur qui va augmenter en fonction du nombre de rebond sur la raquette, car elle est dans le **if du rebond**. Nous l'avons mis entre les tests de collisions de la raquette, car les conditions du rebond du côté haut et bas de la raquette sont les mêmes que celles du rebond par défaut mais en ajoutant quelques conditions en plus pour éviter des bugs.

déplacement: **Nam**
La variable flag va nous être très utile pour le déplacement et l'arrêt de la balle qui variera grâce aux boutons. Les éléments importés tel que **interface.move** permet le déplacement et le **fen.after** déclenche l'appel de la fonction déplacement et permet d'exécuter celle-ci chaque 20 milliseconde.

b) fonction haut: **Nam**

Cette fonction permet de bouger la raquette vers le haut de l'interface. Dans cette fonction, nous avons une variable **event** qui va nous permettre de lier plus tard cette fonction avec les touches du clavier de l'ordinateur qui sera fait avec la méthode **fen.bind**.
Nous avons mis une condition si le flag est égal à 1 la raquette peut bouger vers le haut jusqu'au mur du haut, mais sans le toucher, car la coordonnée de la raquette doit être plus petite que celle du mur. Cela évite des bugs qui auraient pu apparaître dans le programme. La valeur -30 représente sa vitesse de déplacement.

c) fonction relancer: **Christopher**

Cette fonction permet de relancer le programme depuis le jeu. Nous avons utilisé **sys.executable** qui est une chaîne donnant le chemin vers l'interpréteur Python, un fichier binaire exécutable. Si Python n'est pas capable de récupérer le chemin réel de son exécutable, **sys.executable** sera une chaîne vide ou None.
Nous avons utilisé pour cela la fonction **os.excl** du module os. Cette fonction permet d'exécuter tout un nouveau programme, remplaçant le processus actuel qu'elle ne renvoie pas. La fonction **sys.argv** permet de passer la liste des arguments de la ligne à un script python. Il faut lui donner un script qui est ici la variable python. Cela va permettre de relancer le jeu à l'infini grâce au bouton relancer.
(module os: diverses interfaces du système d'exploitation)

d) fonction démarrer: **Nam**

Nous avons utilisé la variable flag qui nous a permis de gérer le déplacement de la balle et de la raquette. Cette variable est utile car elle permet de mettre des conditions au programme qui réagissent en fonction de la valeur de ce flag. Comme nous pouvons le voir sur le code, si la variable est égale à 1, la balle et la raquette peuvent avancer. Cette variable est très utile pour les boutons démarrer et pause. On évite ainsi le démarrage de plusieurs boucles concurrentes car initialement

lorsqu'on appuie sur le bouton démarrer le flag passe de 0 à 1. Lorsqu'on essaie d'appuyer plusieurs fois sur le bouton démarrer, la fonction démarrer ne peut plus être appelée car le flag est déjà à 1. Nous avons aussi utilisé un flag qui revient à 0 lorsqu'on appuie sur le bouton pause et il permet à la boucle de s'interrompre.

4.Démonstration du programme: **Christopher**

1) montrer que les entrées de l'utilisateur pour la couleur et la vitesse marchent même s'il ne rentre pas une valeur appropriée

2) montrer les niveau de difficultés et la différence entre les niveaux avec les changements de couleurs du terrain

3) montrer le fonctionnement des différents boutons

4) montrer le game over lorsque l'on perd

5) montrer le rebond de la balle sur le côté de la raquette (bug corrigé)

6) une partie normale

5.Critique du travail: **Nam**

En se basant sur les différents tests que nous avons pu faire sur notre jeu PONG grâce à notre programme, nous pensons que notre programme est assez complet et réussi, mais pourrait être encore bien amélioré. Les différents essais du jeu nous ont permis de remarquer que le programme ne comprenait pas de bug important empêchant l'utilisateur de jouer au jeu. Nous avons essayé de réduire la majorité des bugs liés à l'utilisation du programme par un utilisateur et nous avons tenté d'afficher un code clair et lisible pour aider à sa compréhension pour ceux qui codaient. En séparant le code en plusieurs sections comme nous en avions parlé dans la structure générale. Nous savons que le rebond de la raquette peut encore être amélioré, ainsi que l'interface graphique, mais nous n'avions pas assez de temps pour cela.

Christopher, Nareg, Nam

Difficulté: **Nam**

Nous avons eu quelques difficultés concernant l'initialisation des lignes et de la balle au départ car nous n'avions pas compris que l'axe y était inversé par rapport à celui que l'on utilise normalement pendant les cours de math. Tkinter étant un nouveau module que nous n'avions jamais utilisé auparavant, il a fallu nous y habituer pour pouvoir réaliser les fonctions et l'interface graphique. Il fallait comprendre et réfléchir à comment certaines options du module de Tkinter pouvaient nous aider à réaliser notre programme. Cela était la principale difficulté avec la gestion des rebonds entre la raquette et la balle.

Point technique: **Nareg**

Nous pourrions parler de la fonction déplacement qui a été le point le plus important et technique de ce code. Le point technique a été le rebond de la balle sur les côtés de la raquette. Nous avions pensé au départ à utiliser la méthode find_overlapping qui permettait de renvoyer l'index des objets du canvas en chevauchement, mais il y avait plusieurs bugs avec le rebond sur le côté de la raquette. Donc nous avons changé le code et utilisé des conditions entre les coordonnés de la raquette et les coordonnés de la balle pour permettre le rebond. Il fallait penser à toutes les possibilités du rebond de la balle sur la raquette. Cela était un des points importants et techniques du programme.
Le programme a été amélioré de différentes manières tel que rajouter de la couleur ou vérifier ce que l'utilisateur rentre comme coordonnés ou couleur dans la console lorsqu'on lui demande. C'était des points qui ont permis d'améliorer notre code d'une certaine manière et d'empêcher des bugs importants qui pouvaient compromettre notre programme.

Amélioration du code: **Nareg**

Si nous avions plus de temps, nous aurions pu améliorer notre jeu en mettant un second utilisateur sur le côté droit de l'interface et ainsi créer un vrai PONG. Il faudrait donc mettre en place une seconde raquette se déplaçant aussi verticalement.
A ce moment-là, le but ne serait plus de faire rebondir un maximum de fois la balle sur la raquette, mais à la place nous pourrions dire que dès que la balle franchit le mur, il y a un point pour l'adversaire. Les murs verticaux seraient les "buts" et les murs horizontaux seraient ceux qui permettent le rebond de la balle.

Enfin, nous pourrions aussi ajouter une fonctionnalité où la balle se déplace directement au milieu de terrain sans avoir à relancer le programme après un but. Cela serait les améliorations que nous aurions pu faire si nous avions eu plus de temps à consacrer au programme.
Il serait aussi intéressant de mettre en place une condition où l'utilisateur peut choisir s'il veut jouer seul ou à plusieurs.

Sources et inspirations:

Le swinnen et mon python nous ont permis de comprendre le module Tkinter et comment l'utiliser. Nous avons copié du code pour la fonction relancer qui nous était compliqué à faire par nous même. Nous nous sommes aussi fortement inspirés de la vidéo youtube concernant les collisions pour pouvoir comprendre le rebond de la balle sur la raquette.

lien des sites en question:

- Gérard Swinnen, Apprendre à programmer avec python 3
- Mon Python, https://sites.google.com/site/pythonpasapas/
- vidéo youtube collision avec coordonnées Tkinter, https://youtu.be/Ui9mpu1zGTE
- Daniweb, restart your python program https://www.daniweb.com/programming/software-development/code/260268/restart-your-python-program