LES TROPHÉES NSI

Édition 2023

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET

document est l'un des livrables à fournir lors du dépôt de votre projet : 4 pages maximum (hors cumentation).

Pour accéder à la liste complète des éléments à fournir, consultez la page **Préparer votre participation**.

Vous avez des questions sur le concours ? Vous souhaitez des informations complémentaires pour déposer un projet ? Contactez-nous à **info@trophees-nsi.fr**.

NOM DU PROJET : 4 à la suite

> PRÉSENTATION GÉNÉRALE :

En ce début d'année calendaire et scolaire (puisque la rentrée en Nouvelle-Calédonie est en février), nous avons commencé la Programmation Orientée Objet (POO). Il nous a été demandé de créer des jeux. Ainsi la machine était lancée !
Notre objectif n'était pas déterminé d'emblée ; chacun développait son jeu de son côté pour le cours de NSI. D'un côté le démineur, et de l'autre la bataille. Après un temps de réflexion et compte tenu du délai très court, nous avons décidé d'optimiser notre travail pour les trophées NSI et de combiner nos deux jeux. Deux jeux, c'est bien, mais quatre c'est mieux : plus de choix, plus de temps de jeux possible, plus d'adaptation aux profils variés des joueurs… C'est à ce moment-là que le projet a vraiment pris forme.
L'idée était à la fois de développer nos connaissances des « Class », d'améliorer nos compétences sur Pygame et la programmation en Python plus globalement, tout en faisant quelque chose qui nous plaisait et dans lequel on s'identifiait.
Derrière ces classiques et populaires jeux vidéos choisis, nous avons volontairement mêlé des styles de jeux très différents : logique, lettre, chance et divertissement.

> ORGANISATION DU TRAVAIL :

Nous avons formé un binôme :

- Lison, qui a géré le développement des jeux Snake et La Bataille ;
- Sam le programmeur derrière le Démineur et le Wordle.

Nous sommes tous deux en Terminale.
Durant la durée de réalisation du projet, chacun était assez autonome dans la création de ses jeux, tout en s'assurant réciproquement que l'autre progressait au même rythme.
Toutes les semaines, nous nous retrouvions pour observer le travail de l'un et l'autre et échanger sur des potentiels problèmes, pour que l'autre apporte son point de vue, ses idées et ses éventuels « coup de génie ». Durant ces réunions, nous partagions également nos pistes d'améliorations pour les jeux, afin de les faire valider par le partenaire.
Au départ, nous travaillions en classe de NSI, les deux jeux faisant partie du travail demandé par le professeur. Après le chapitre sur la POO, nous avons poursuivi le développement chacun chez soi.
Pour le partage des programmes, nous utilisions un drive, qui garantissait un accès à toute heure, sur tout appareil.
Pour communiquer, nous utilisions particulièrement Discord et parfois Messenger suivant les besoins.
Nous travaillions principalement sur EduPython et Thonny.

LES ÉTAPES DU PROJET :

Au départ, on commence par découvrir la POO en classe et se familiariser avec ce type de programmation, pendant environ 2 semaines.

Puis on rentre dans le dur : le développement des deux premiers jeux commence, le démineur et la bataille font leur apparition. La première version est encore moins optimisée que celle actuelle. On utilise déjà Pygame pour l'interface graphique. A ce moment-là, on décide de créer une mini-console de jeu en réunissant nos deux jeux et dans la limite de temps imposé, en rajouter.
Les vacances scolaires font naître le Wordle et le Snake. Nos 4 jeux sont fonctionnels, il ne reste plus qu'à ajouter quelques plus par-ci par-là et la réunification commence.
Moins d'une semaine pour relier les jeux, faire les dossiers techniques, la vidéo… Mais ça a été réalisé, au détriment d'un délaissement des autres matières et de nuits bien plus courtes. Le jeu est fonctionnel, nous un peu moins 😉 !

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

Pour répondre à cette partie, nous vous présentons chaque jeu séparément.

- Le démineur :

  Il a été codé avec la POO, et comporte deux classes :

  - La classe « Case », qui attribue diverses valeurs concernant chaque case : leurs coordonnées dans la grille, leurs coordonnées dans le « Plateau » (l'autre classe) soit l'un des attributs les plus utilisés, leur contenance (si elle contient une « Bombe », une « Case_Vide », des « Chiffres »…).
  - La classe « Plateau », composée de plusieurs cases. Elle prend en argument la difficulté puisque à celle-ci est attribuée la taille du plateau et le nombre de bombes. La classe a des méthodes intéressantes. La méthode « Repartition_Bombe » n'était pas compliquée à mettre en place, c'était plutôt celle qui attribue les données aux cases sans bombes et celle qui révèle plusieurs cases d'un coup qui ont provoqué de nombreux bug. Le problème venait des coordonnées des cases dans le plateau, il a fallu du temps et de nombreux essais pour paramétrer une condition prenant en compte toutes les exceptions.

  J'avais pour objectif de départ d'intégrer un chronomètre qui aurait rajouté du challenge, mais je m'y suis pris trop tard. Le problème principal étant que la taille du plateau en mode « Facile » ne permettait pas l'ajout d'un chronomètre visible, sans changer de nombreuses lignes de codes éparpillées dans le programme.

- Le wordle :

Codé sans les classes, l'aperçu graphique a été une des premières choses faites. Je considère ce jeu comme le plus abouti, avec des « effets graphiques » (Les lettres ne se colorent pas toutes en même temps), un relevé des scores qui peut être conservé après la fermeture du logiciel grâce au module « pickle », un accès à la définition du mot à trouver à la fin de la partie. Dès le début, j'ai été confronté à un problème qui me semblait très pénible à résoudre : le clavier. Au départ, lorsqu'une touche était pressée, je regardais à laquelle elle correspondait avec « K_a » pour la lettre « a » par exemple. Pygame étant paramétré sur un clavier qwerty, quand je tapais « a », Pygame comprenais « q ». J'ai réglé ce problème avec la méthode « event.unicode » qui renvoie un string correspondant à la lettre associé à la touche.
L'importation des 7980 mots a été faite dans une liste, même si j'hésitais à utiliser une autre méthode. J'ai fini par laisser les mots dans la liste car c'est la méthode avec laquelle j'étais le plus à l'aise.

J'ai paramétré en dernier le fait que le clavier du bas, sur l'interface graphique, soit cliquable. J'utilise une méthode à peu près similaire à celle du démineur, cette fonctionnalité m'a donc pris moins de temps que je ne l'avais imaginé.

- Le snake :

Ce jeu est codé sans classe, à partir de 4 codes python différents : la page d'accueil et les 3 niveaux. Ainsi chaque niveau est écrit sous la forme d'une fonction importée dans la page d'accueil. (exemple : from snake.niveau1 import niveau1_fonction)
Spécificités de chaque niveau :
Chaque niveau apporte de nouvelles fonctionnalités tout en imageant l'avancement du jeu lui-même et des nouvelles techniques/idées d'amélioration.

- Le 1er niveau correspond à un serpent immobile qu'on fait déplacer avec la fonction pygame.key.get_pressed() selon les flèches sur lesquelles le joueur appuie. Il peut s'auto manger (et même faire demi-tour sur lui-même).

- Dans le 2e, le serpent avance en continue… on ne peut plus l'arrêter ! Pour coder cela on initialise sa direction vers la droite dès le début du code (right=True) et cette direction change quand le joueur change de sens.
  Cette vitesse imposée au serpent rend le jeu plus compliqué, de plus si le serpent sort de la fenêtre il perd un point, et il est éliminé s'il s'auto mange en faisant demi-tour sur lui-même : event.key==K_LEFT and right==True:# s'il allait dans une direction et qu'il prend la direction inverse (de droite à gauche sans faire de demi-tour), alors : run=False.

- Enfin le dernier niveau met en scène un adversaire.
  Créer ce deuxième serpent a été vraiment intéressant. Et bien qu'il ne soit pas complètement perfectionné, il est déjà capable de nous battre ! En effet pour éviter que le serpent adverse s'auto mange, on regarde ses coordonnées et s'il est tout proche de se manger, il prend une autre direction. Or cette technique n'a pas été suffisamment approfondie pour permettre au serpent de ne jamais rentrer en collision avec lui-même. C'est une des améliorations qu'on pourrait ajouter à ce jeu.

Autres améliorations à apporter au jeu : dans le dernier niveau toujours, les 2 serpents peuvent se chevaucher ou se couper la route. Dans ce cas il faudrait analyser lequel entre dans lequel pour déterminer qui perd un point.

Fonction utile trouvée : colliderect(), qui a permis de ne pas avoir à écrire les intervalles de valeurs correspondant à la nourriture et au serpent et qui détecte si un élément en touche un autre.

Curiosité : Pour ce jeu on a importé des fonctions qui contiennent elle-même des fonctions et du pygame. Cela permet de structurer et de rendre plus compréhensible la page d'accueil du jeu snake.

- Les cartes :

Le jeu de la bataille est composé de 2 classes que nous avions étudié en cours de NSI : la classe Carte et celle JeuDeCartes.
Ce qui m'a intéressé c'était de pouvoir rendre les cartes visuelles en utilisant Pygame, puis ensuite de pouvoir choisir la carte que je veux jouer, abandonnant les simulations de bataille aléatoires.

Le 1er problème a été d'importer les cartes, sans les faire une par une. Pour cela on utilise une variable qui correspondra à son emplacement. Et puis quand les cartes s'affichaient, assez vite la fonction blint m'a posé problème, car lorsque les cartes se superposent et qu'on joue, certaines se retrouvaient cachées. C'est une des raisons pour lesquels on n'a que 5 cartes (10 à la fin de la partie qui sont disposées sur les 1000px de la fenêtre), puisque davantage ne seraient pas entrées dans la fenêtre d'affichage.

Curiosité : Créer un menu où l'on peut changer les fonds et les musiques du jeu. Pour cela, dès l'entrée dans la boucle de jeu, tous les éléments faisant parti du menu sont affichés, mais hors cadre (y=700). Et lorsqu'on clique sur le menu alors leur ordonnée change et ils deviennent visibles. Cette méthode a également été utilisée pour les cartes de dos et de faces. En effet les cartes de l'adversaire sont de dos mais ses cartes de faces sont également dans le programme mais non visible dans la fenêtre, elles le deviennent que lors de la bataille quand les 2 cartes en jeu sont au milieu.

> OUVERTURE :

A terme, l'idée serait de rassembler le plus de jeux possibles. Comme nous les importons de sous-dossier, leur intégration sera facile et rapide à effectuer. Le système d'image en récompense à été intégré rapidement vers la fin. Par la suite, il serait intéressant de faire un partenariat avec le Festival Sublimage (https://www.festivalsublimage.nc/)par exemple, un festival qui récompense les meilleurs clichés pris dans un milieu marin, et de rajouter des images périodiquement.
Ou bien on pourrait même envisager de relier ce projet à l'Office du tourisme, afin de promouvoir le territoire, et peut être diffuser le jeu à l'international, donc implémenter des langues différentes.
Il parait nécessaire, pour l'attractivité, de rajouter des effets spéciaux et du son dans chacun des jeux. Qu'est ce qui différencie les jeux de nos jours finalement, si ce n'est quelques paillettes et froufrous, accompagnés de petits animaux mignons.
Pouvoir modifier la taille des fenêtres serait intéressant mais très compliqué à réaliser étant donné le nombre d'affichages différents.
Bien sûr, l'entièreté du code est à améliorer en termes d'optimisation, pourquoi pas dans quelques années, quand nous aurons évolués en maître de la programmation ?
Si nous devions le refaire, nous utiliserions probablement plus les classes, dans chaque jeu. Si nous ne l'avons pas fait, c'est parce que nous n'y sommes pas encore habitués et que cela reste abstrait dans notre esprit de « poussin codeur ».
Avoir un groupe plus important serait intéressant, géré le code, la documentation, le montage vidéo… représente un travail colossal, qui aurait pu être mieux réparti avec une équipe plus nombreuse. Dans l'ensemble, cette expérience reste très instructive.

DOCUMENTATION

Nous avons utilisé Python3.

Plusieurs modules sont importés : pygame, random, time, pickle, webbrowser.

L'entièreté du code est en Python, une partie du code a une architecture correspondant à la Programmation Orientés Objet (POO).

Les images utilisées sont stockées dans des sous dossier des dossiers des jeux.
La liste de mot du Wordle vient du site https://www.listesdemots.net/mots5lettres.htm

Le programme à lancer est sources/4aLaSuite.py

Le menu principal qui permet d'accéder aux jeux :