Galli Evan
Lecard Maxence
Groupe 5

Rapport projet : Memory

Répartition et organisation du travail :

Nous avons utilisé l'IDE Visual Studio 2022 afin de profiter de la richesse de ses outils et de son confort d'utilisation.

Nous avons aussi utilisé GitHub afin de nous faciliter la tâche en termes de communication du code et notamment pour le suivi du travail grâce à un historique nous permettant de suivre les modifications apportées par notre binôme.

Nous avons aussi essayé de respecter au maximum les conventions de code, nous avons pour cela utilisé pylint, un outil de vérification de code qui nous a rapporté les changements nécessaires. Pylint attribue à notre projet une note de 9.48 / 10

Pour la répartition du travail, Evan s'est occupé du système de jeu, alors que Maxence s'est occupé du décor. Nous avons aménagé des horaires de travail pendant les vacances, notamment par appels discord.

Bien que ce rapport soit destiné à apporter des explications quant à notre programme, le code contient de nombreux commentaires afin d'expliquer plus en détail certains passages.

Thème graphique et niveaux de difficulté :

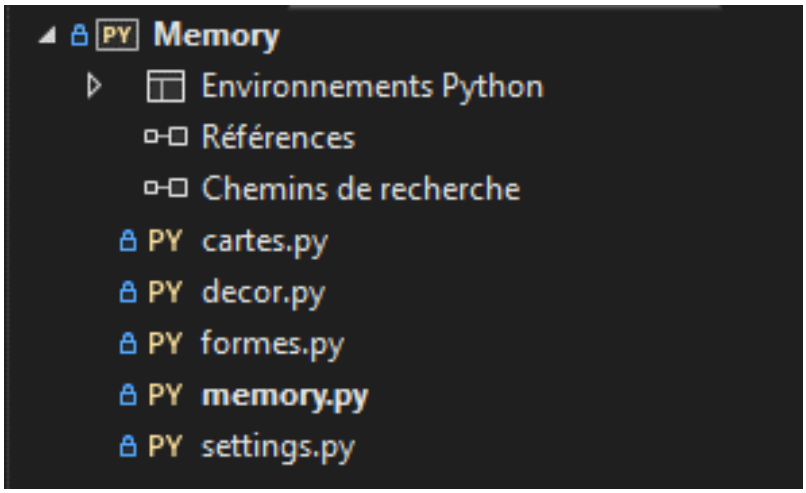
Nous avons choisi le niveaux 2+ pour le projet, l'utilisateur peut donc cliquer sur l'objet de son choix. Nous avons aussi implémenté une barre de progression permettant au joueur de visualiser le nombre de tentatives restantes (nous reviendrons plus en détail sur ce point).
Le thème de notre jeu est l'hiver.
Concernant les éléments du décor nous avons mis deux bonhommes de neige au premier plan et les éléments du décor qui se répètent sont les étoiles, celles-ci scintillent et sont positionnées aléatoirement à chaque redémarrage du jeu. **Les éléments graphiques comme le reste du système de jeu ont été créé par nos soins.**

Règle du jeu :

Le joueur clique une première fois pour retourner la carte, puis il clique sur une seconde carte. Si les deux cartes ont la même forme et la même couleur alors le couple de carte reste retourné, (sinon, les cartes se retournent après un délai d'une seconde). On continue ainsi de suite jusqu'à avoir trouvé toutes les paires de la partie.
Il y a un nombre maximum de tentatives qui dépend du nombre de cartes dans la partie (il correspond au double du nombre de couple de cartes), on peut visualiser ce nombre d'essai grâce à la barre de progression située au milieu en bas de l'écran. Au-delà de ce nombre d'essai, la partie s'arrête, et on a perdu.

Structure du code :

Notre projet est divisé en 5 modules dont nous allons détailler le rôle ci-après



Programme principal :

Le programme principal se trouve dans le fichier `memory`.

Au début de notre programme, nous importons les modules et les librairies dont nous besoin. Nous avons désactivé le redimensionnement de la fenêtre de jeu car celle-ci nous aurait posé problème pour le positionnement du décor. Nous nous sommes servis de trois tortues différentes pour tracer le décor, les cases et la barre de progression. C'est aussi dans ce programme que nous appelons la fonction `main` qui trace le décor.

Génération aléatoire des paires de cartes :

Le nombre de carte est déterminée à partir du nombre de colonnes et de lignes de la grille.
Le code génère une liste de l'ensemble des couples forme/couleur.
La génération aléatoire privilégie les couples dont la couleur et la forme ne sont jamais sortis (pour éviter de jouer avec des objets qui sont tous de même couleur ou tous de la même forme). Puis on sélectionne parmi les couples forme/couleur non tirée le nombre manquant de couples.
Nous avons alors généré l'ensemble des couples forme/couleur, il nous suffit de dupliquer la liste obtenue pour obtenir des paires de cartes. Enfin on mélange la liste de manière à répartir les cartes sur la grille.

Le jeu :

Le programme de jeu est contenu dans une fonction nommée `clickCases` celle-ci est appelée lorsque l'utilisateur effectue un clic.
Elle vérifie tout d'abord grâce à la fonction `obtenirCase` présente dans `cartes` si le clic s'est produit sur une carte et, qui plus est, est non retournée. Si ce n'est pas le cas, elle s'arrête.
Une fois la première carte sélectionnée par le joueur, le programme affiche le contenu de la carte en traçant l'objet correspondant puis attend l'instruction de joueur pour la deuxième carte.
Si les cartes retournées sont de même couleur et de même forme alors le joueur a trouvé une paire. On garde alors les cartes face visible, sinon on attend une seconde (toute action du joueur est désactivée durant cette période) pour laisser le temps au joueur de visualiser le contenu avant retourner les cartes (c'est à dire redessiner la carte sans l'objet).
Enfin si les cartes ont toutes été retournée alors la partie est gagné et un écran de fin apparait.
Si au contraire, le joueur a dépassé le nombre de tentatives autorisées, la partie est perdu et autre écran de fin apparait.
Dans les autres cas, le jeu se prépare à recevoir un nouveau couple et à répéter les mêmes instructions.

Settings :

C'est le fichier qui contient les paramètres du jeu (comme le nombre de cartes, la couleur et les formes des objets de la carte) et que l'utilisateur peut modifier.

Cartes :

C'est le module qui chargé du bon fonctionnement de la grille. C'est lui qui positionne et dessine les cartes et leur contenu.
Dans le fichier il y a plusieurs fonctions :

- La fonction `dessineCase` qui dessine la carte.
- La fonction `positionCase` qui calcule la position de la carte d'index i.
- La fonction `obtenirCase` qui calcule l'index de la carte présente aux coordonnées x, y.
- La fonction `afficheContenu` qui dessine les cartes non sélectionnées et qui révèle le contenu des cartes sélectionnées et de celles qui ont déjà été révélées (les paires de cartes déjà trouvée).

Formes :

Le module `formes` est composé de nombreuses fonctions qui ont pour objectifs de tracer les différents éléments graphiques du jeu comme le décor ou bien les objets sous les cartes.
Ce module contiens notamment la fonction `dessine` qui est appelée pour chacune des fonctions de `formes`, cette fonction `dessine` permet d'initialiser la position, la couleur, l'épaisseur et l'orientation de la tortue.
L'ensemble des fonctions (à l'exception de `texte`) prennent en paramètre les coordonnées du point en bas à gauche de la figure. Ainsi pour la fonction `rond` par exemple, les coordonnées ne sont pas celles du centre mais celle du point en bas à gauche d'un carré (imaginaire) dans lequel le rond serait circonscrit.
Chacune de fonctions sont assez détaillées et nécessite pas forcement beaucoup plus d'explications que celle fournies dans le code.

Decor :

La fonction `main` du module `decor` est appelé dans le programme principale. Celle-ci est chargée d'appeler le reste des fonctions de décor qui ont pour rôle de tracer des objets. Le programme principal étant indépendant du décor il serait possible de créer plusieurs décors sans avoir à changer le reste du programme (on pourrait créer par exemple plusieurs saisons), pour cela il suffirait de changer le module décor.
Le fichier `decor` est décomposé en plusieurs fonctions :

- La fonction `main` : c'est la fonction qui est appelée par le programme principale et qui appelle les différentes fonctions chargées de tracer le décor.
- La fonction `generateEtoiles` qui créer une liste de n tuples (x, y, taille), en respectant un espacement minimum entre les étoiles afin d'éviter qu'elle se superposent.
- La fonction `etoiles`, cette fonction permet de tracer les étoiles en fonction des n tuples obtenus grâce à la fonction `generateEtoiles`.
- La fonction `bonhommeDeNeige` : trace un bonhomme de neige
  Pour la construction du bonhomme de neige, nous nous sommes servis de la fonction `rond` présente dans le module `formes` pour tracer les corps du bonhomme de neige ainsi que ses yeux et les boutons qui sont sur son corps. Nous nous sommes servis de la fonction `triangle` qui se trouve dans `formes`, afin de tracer un triangle orange à l'horizontale qui représente la carotte (le nez du bonhomme de neige). Enfin, nous avons utilisé la fonction `rectangle` de `formes` pour tracer les bâtons qui forment les bras du bonhomme de neige, cette fonction appelé de façon à tracer un rectangle marron incliné d'un angle respectif de 30 et 150 degrés.

Informations supplémentaires :

La durée du projet :

Ce projet a été réalisé en une quarantaine d'heures réparties sur 6 semaines.

Problèmes rencontrés :

Durant le développement de ce projet nous n'avons pas rencontré de problèmes majeurs. Voici une liste non exhaustive des soucis que l'on a eu ainsi que la solution que l'on a trouvée :

- L'affichage qui ne s'actualise pas : Utilisation de la fonction `update` de `turtle`
- La récupération de la taille de la fenêtre :
  Utilisation de `turtle.Screen().window_width()` et `turtle.Screen().window_height()`
- Le redimensionnement de la fenêtre qui cause un problème de positionnement du décor : Désactivation du redimensionnement grâce à la fonction `turtle.Screen().cv._rootwindow.resizable`
- La génération aléatoire des paires de cartes qui privilégie les couples dont la couleur et la forme n'est pas sortie : Utilisation d'`itertools.product` pour calculer les combinaisons possible et choix des paires en fonction de leur position dans la liste

Améliorations possibles :

Bien que notre jeu soit parfaitement fonctionnel, il pourrait être améliorer sur certains points :

- Avoir la possibilité de choisir la difficulté au démarrage. Celle-ci pourrait par exemple consister en une diminution du nombre de tentatives autorisées, ou encore par une augmentation du nombre de cartes dans la partie.
- Pouvoir recommencer la partie sans avoir à redémarrer le jeu
- L'algorithme de calcul du nombre de tentatives pourrait être ajusté pour être plus en raccord avec la difficulté réelle : 4 tentatives pour 2 couples, c'est gagné d'avance, mais 60 tentatives pour 30 couples, ça l'est beaucoup moins !
- Enfin, on pourrait imaginer implémenter un système de saison dans le jeu qui changerait les cartes et le décor en fonction de celles-ci.

Ce que le projet nous a apporté :

Finalement ce projet a été une réelle façon de s'entraîner à la programmation python. Devoir rechercher des solutions aux problèmes auxquels nous avons dû faire face, nous a aussi permis d'imaginer le travail de développeur en entreprise avec un travail précis à réaliser tout en devant respecter une date limite. De plus cela nous a aussi permis d'utiliser et de découvrir certains logiciels et plateforme de programmation. Mais encore cela nous a aussi permis d'apprendre le travail en équipe et nous appris à être coordonnée par exemple en répartissant les tâches de travail et en mettant en place des horaires de travail en communs notamment par appels discord pendant les vacances par exemple. Enfin ce projet nous à permit d'améliorer notre niveau en python peut importe notre niveau de base.