

Rapport Match-3 HELA

Antonin Montagne, Enzo Mahoukou, Hiba L Moudden, Lou-Anne Gautherie

14 Avril 2021

Table des matières

1 Objectifs du projet	2
2 Les Débuts	3
3 Choix du niveau	3
4 Thème du jeu	4
5 Gemmes	5
6 Gestion de la grille	6
7 Interface	7
8 Pseudo	9
9 Assemblage des codes	10
10 Dernière ligne droite	10
11 Affichage Jeu	11
12 Diagramme des classes	13
13 Structure du Jeu	14
14 Conclusion	14

1 Objectifs du projet

Nous avions pour but de programmer un jeu match-3 sur le thème des gemmes ayant pour nom : "HELA's Gems".
"H" pour "Hiba", "E" pour "Enzo", "L" pour "Lou-Anne" et "A" pour "Antonin".

Le but du jeu est tout simplement d'aligner 3 mêmes gemmes afin qu'elles s'éliminent.

Description plus précise :

Il fallait créer une grille différentes selon le niveau, et remplir cette grille avec des gemmes. Nous devions mettre en place tout un système de gestion d'événements qui faisait en sorte que lorsque deux gemmes étaient cliquées, elles se permutaient, et que si 3 gemmes identiques se suivaient, elles se supprimaient et étaient remplacées par 3 nouvelles.

Nous devions enfin mettre en place un système de score qui calculait les points accumulés au long de la partie en fonction des matchs effectués et un système de temps qui faisait en sorte que lorsque le temps atteignait 0, le jeu s'arrêtait et renvoyait un écran de fin.

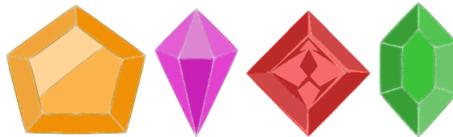


FIGURE 1 – Gemmes présentes dans le jeu

Nous devions donc créer une grille avec ces gemmes qu'Enzo a dessiné lui-même à l'intérieur et créer des fonctions de permutations.
Cela fait penser au célèbre jeu "Candy Crush" qui a le même principe.

2 Les Débuts

Nous avons commencé par coder la grille vide en Python le 20 Janvier, grâce aux bases que nous avions du 1er semestre, et nous l'avons, par la suite, transformé en grille Pygame.

Au départ, nous ne comprenions pas nos erreurs, le code ne marchait pas et la grille ne s'affichait pas. Ce n'est qu'après plusieurs essais que nous nous sommes rendu compte que le problème ne venait pas du code en lui même mais de la couleur de la grille qui était noire sur un fond noir.

Nous avons mis plusieurs séances à trouver comment créer la grille sur Pygame.

Hiba et Antonin ont testé plusieurs méthodes et nous avons finalement opté pour la création de lignes et de colonnes.

Hiba et Antonin ont créés les lignes de la grille grâce à deux fonctions : "dessinerColonnes" et "dessinerLignes" qui prenaient en paramètres la marge, le point de départ, le point d'arrivée et le nombre de colonnes/lignes souhaité. Ces lignes et colonnes étaient formées à partir de tuples de tuples qui représentent les coordonnées du point de départ et du point d'arrivée de la ligne auquel on a ajouté une marge pour centrer la grille.

Pour cela, Antonin s'est inspiré d'un tutoriel qui servait à tracer les lignes d'un morpion.

[Lien de la vidéo en question.](#)

La grille sous format Pygame était formée le 1er Février.

Après cela, Le groupe s'est scindé en deux : Hiba et Enzo travaillaient sur le code pure Python, tandis qu'Antonin et Lou-Anne s'occupaient de l'interface en Pygame.

3 Choix du niveau

Lou-Anne a ensuite créée une fonction pour le choix de niveau le 3 Février, qui prend en paramètres le niveau.

À chaque début de partie, le joueur doit choisir le niveau de la grille qui sera plus ou moins grande en fonction du niveau choisi : "easy", "medium" ou "hard".

```
In [1]: runfile('C:/Users/loulo/OneDrive/Documents/FAC/Informatique/Conception Logiciel/hello/niveau_grille.py', wdir='C:/Users/loulo/OneDrive/Documents/FAC/Informatique/Conception Logiciel/hello')
pygame 2.0.1 (SDL 2.0.14, Python 3.8.5)
Hello from the pygame community. https://www.pygame.org/contribute.html

Quel niveau voulez-vous choisir ? easy, medium ou hard:
medium
```

FIGURE 2 – Demande du niveau

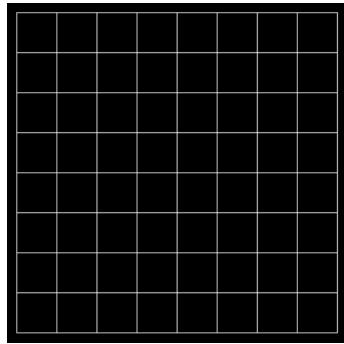


FIGURE 3 – Grille médium vide

Cette fonction est ensuite devenue "dessinNiveau" qui prend en paramètres l'écran, le niveau, et une grille. Elle nous permettait de dessiner notre grille en fonction de son niveau sur l'interface Pygame.

La fonction dessinNiveau utilise les fonctions "dessinerLignes" et "dessiner Colonnes". Nous avons donc créé une boucle pour chaque tracer ligne.

La fonction "dessinNiveau" appelle la fonction "remplissageGem" qui se trouve dans le module "icon.py".

4 Thème du jeu

La semaine du 8 Février était une semaine de recherches et de documentation sur les permutations, la mise en page du titre, le thème du jeu et le remplissage de grilles pygame.



FIGURE 4 – Fond d'écran du jeu

5 Gemmes

Enzo a créé lui même l'apparence des gemmes à partir de Paint le 15 Février, puis nous les avons ajoutées dans notre grille grâce à un groupe de "sprite".

Pour ce faire, nous avons créer une classe "Gem" (un objet de type sprite) qui prend les coordonnées de notre gemme (x, y) en paramètre ainsi que son identifiant (entre 1 et 4). La classe "Gem" contient les propriétés pour récupérer l'image de notre gemme grâce à une liste prédéfinie qui contient les URL des images de nos 4 gemmes ainsi que le rectangle pour afficher l'image.

Antonin a fait en sorte que les gemmes se positionnent aléatoirement dans la grille le 17 Février grâce à la fonction "remplissageGem". Cette fonction prend en paramètres les coordonnées x et y , le nombre de cases ainsi que la grille. Pour chaque case de la grille on construit une gemme que l'on ajoute à notre groupe de sprite et on incrémente nos coordonnées de 50 car les cases font 50 pixels. Puis pour afficher les gemmes sur l'écran Pygame nous avons dessiner notre groupe de sprite.

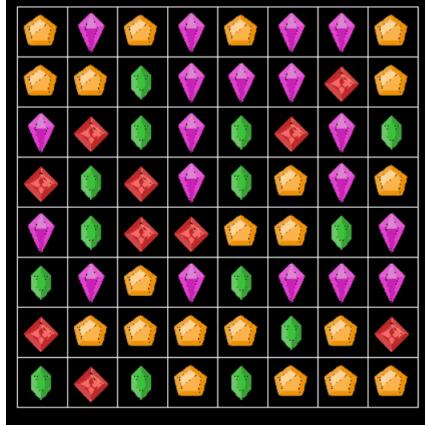


FIGURE 5 – Grille remplie avec les gemmes

6 Gestion de la grille

Hiba écrivait parallèlement le code permettant de créer la fonction de base pour le match-3.

Le 1er Mars, nous avons décidé de choisir des permutations par clique sans animations.

Hiba a créé un module "gestion_grille.py", avec une classe "gestionGrille" qui prend en paramètres les dimensions de la grille.

Elle a créé une fonction "remplissage" qui prend en paramètre une grille et qui remplit aléatoirement cette grille de chiffres entre 1 et 4. Pour cela elle a parcourue la grille en attribuant un chiffre à chaque case. Cette fonction renvoie une grille remplie de chiffres.

Elle a ensuite créé deux fonctions : "eliminerLignes" et "eliminerColonnes" qui prennent en paramètres des grilles et qui renvoient une grille où l'on a supprimé les mêmes chiffres qui s'alignaient. Elle a ensuite regroupé les deux fonctions en une "eliminerLignesColonnes".

Puis elle a créé une fonction "newtab" qui prend en paramètre une grille sur laquelle toutes les fonctions précédentes sont appelées afin d'obtenir une grille où nous avons supprimé tous les matchs et remplacer les chiffres éli-

miner par de nouveaux.

Ensuite elle a crée une autre fonction "recup_case" qui prend en paramètres le niveau, x et y. Elle permet de savoir dans quelle case nous avons cliqué. Elle retourne un tuple qui correspond aux coordonnées des cases dans la grille.

Finalement elle a crée une dernière fonction "permut" qui prend en paramètres une grille, click1 et click2 qui permet d'échanger les deux cases sur lesquelles nous avons cliqué si elles répondent à certaines conditions. Cette fonction renvoie une nouvelle grille avec les deux cases inversées.

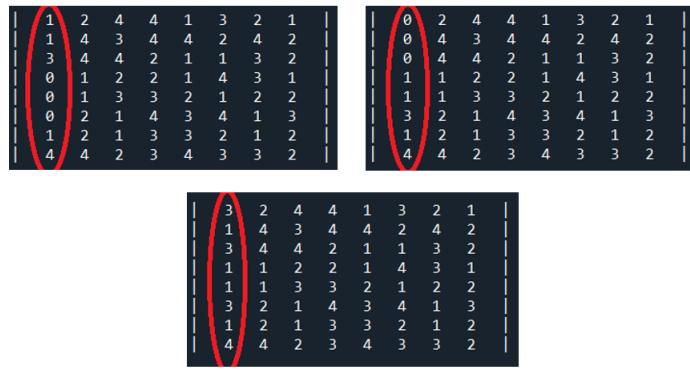


FIGURE 6 – Changement de gemmes suite à un match

7 Interface

Antonin et Lou-Anne ont décidé de s'occuper de l'interface le 10 Mars.

Ils ont créés une classe "Bouton" (un objet de type sprite) qui prend les coordonnées de notre bouton (x, y) en paramètre ainsi que son chemin d'accès.

La classe "Bouton" contient les propriétés pour récupérer l'image de notre bouton grâce à une liste prédefinit qui contient les URL des images de nos 4 boutons ainsi que le rectangle pour afficher l'image.

Ils ont décidé de créer un bouton play qu'ils ont ajoutés à un groupe de sprite destiné à contenir les boutons et ont cherché comment faire pour que lorsque l'on appuie sur le bouton, la fonction "dessinNiveau" se lance.

Le 15 Mars, ils ont réussi à faire en sorte que lorsque l'on appuie sur le bouton play, la console nous demande quel niveau nous voulons choisir.

Pour cela, ils ont récupérer les coordonnées de la souris quand le clique était effectué, et ils ont vérifié que lorsque nous cliquions les coordonnées de la souris étaient contenues dans celle du rectangle du bouton.

Antonin et Lou-Anne ont donc décidé de créer 3 boutons avec les différents niveaux inscrits dessus qu'ils ont ajoutés au groupe de sprite. De la même façon ils ont vérifié que la souris était contenue dans le rectangle du bouton lors du clique. Si c'était le cas, ça lançait la fonction "dessinNiveau" avec le niveau correspondant au bouton.



FIGURE 7 – Boutons représentants les niveaux

Lorsqu'ils cliquaient sur le bouton, ils avaient un problème car la fonction "dessinNiveau" se lançait bien mais la grille se dessinait par dessus les boutons, qui ne disparaissaient pas de l'écran.

Pour corriger cela, ils ont donc créé une fonction "supprimer" qui prend en paramètre un groupe de sprite et 3 boutons, et qui supprime les boutons du groupe de sprite.

Après avoir appelé la fonction il ont ré-affiché le fond d'écran par dessus puis dessiné la grille sur le fond d'écran. Le problème était enfin réglé.

Avant de trouver la solution, Antonin et Lou-Anne se sont demandés si il ne valait pas mieux lancer le jeu en appuyant sur une touche du clavier, Lou-Anne a donc tester cette méthode mais celle du bouton était finalement la bonne.

Enzo a dessiné lui même les boutons pour les niveaux.



FIGURE 8 – Bouton Play et Boutons Niveaux

8 Pseudo

Le 31 Mars, Lou-Anne a créée la partie de code qui demande à l'utilisateur de marquer son pseudo et celui ci s'affiche en tant que "nouveau joueur".

Pour ce faire, elle a d'abord recherché la police qu'elle souhaitée utiliser, puis elle affiché sur l'écran le texte qui demande d'afficher le pseudo

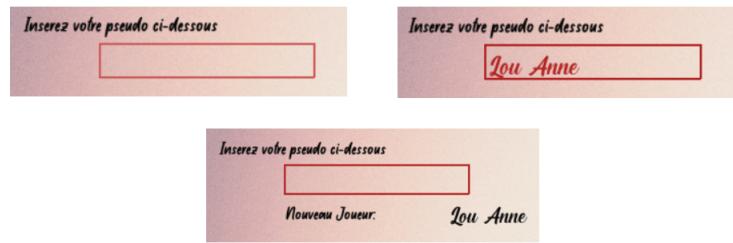


FIGURE 9 – Demande du pseudo

Elle a créée un rectangle où l'on pouvait écrire dedans. Quand le rectangle était cliqué, la couleur changée et nous pouvions écrire dedans. Lorsque l'on appuyait sur "Entrée", cela affichait le pseudo qui venait d'être saisi accompagné d'un texte "Nouveau Joueur :".

9 Assemblage des codes

Le 31 Mars, Hiba et Antonin ont commencé à joindre leurs codes. Hiba avait du code en Python et Antonin avait du code en Pygame. La tache fu très compliquée et pris plus d'une semaine.

En effet, les chiffres que Hiba utilisait de base causait problème. Il fallait les remplacer par des tuples afin d'avoir de vrais permutations.

Ensuite, lorsque 3 gemmes se suivaient, 3 nouvelles gemmes se superposaient sur les anciennes, sans effacer les précédentes.

Hiba et Antonin avaient un problème d'actualisation d'affichage.

Pour le régler, ils ont créer une fonction "vidageSprite" qui ne prend aucun paramètre et qui suppriment chaque gemmes du groupe de sprite. Après ça ils ont rappeler la fonction "dessinNiveau" pour dessiner les nouvelles gemmes. Ils ont répéter cette opérations grâce à une boucle à chaque permutation.

10 Dernière ligne droite

Lors de la dernière semaine, nous nous sommes occupés des derniers détails.

Lou-Anne s'est occupé de joindre la fonction qui demande le pseudo et le stocke, dans le fichier "main.py", ainsi que la mise en page du jeu. Elle a aussi trié tous les fichiers et dossiers, et s'est occupée du rapport.

Antonin a aidé Lou-Anne pour joindre la fonction du pseudo et pour faire le rapport. Il s'est aussi occupé de la fonction qui calcule et affiche le score en se basant sur les recherches d'Hiba.

Le score est calculé grâce aux fonctions de "GestionGrille", lors de la partie, des points sont attribués en fonction du match effectué.

Il a d'abord créé un fichier txt contenant un dictionnaire vide, il a ensuite créé la fonction "enregistrer_score".

Cette fonction contient deux paramètres : pseudo et score_partie. Elle ouvre

le fichier "score.txt", le lit, vérifie si il y a le pseudo du score parmi les clés du dictionnaire. Si un score existe, elle le compare avec le score de la partie actuelle, sinon elle enregistre le score actuel.

Il a ensuite créé un texte "Score :" qu'il a affiché sur l'écran avant le score.

Il a également créé un chrono qui, lorsqu'il se termine, affiche un écran de fin du jeu.

Il a créé une fonction "afficher_temps" qui prend en paramètres la police d'écriture, l'écran, le temps de la partie, et le fond d'écran du jeu.

Cette fonction soustrait au temps de partie initialement définit le temps qui passe grâce à la fonction "get_ticks()" du module "time" de pygame. Elle affiche un message "Temps :" suivi du temps qui reste.

Cette fonction a posé problème car le temps se superposait à chaque seconde qui passait. Il a donc blit le fond d'écran à chaque seconde pour que l'affichage s'actualise mais cela a eu pour effet d'effacer les lignes de la grille. Il a donc créé une fonction "dessinGrille" pour pouvoir redessiner les lignes de la grille.

Le problème était ensuite réglé.

Lou-Anne s'est ensuite occupée de l'apparence de l'écran de fin du jeu.

11 Affichage Jeu

Lorsque le jeu se lance, le titre est affiché en grand suivi du bouton "Play". Le pseudo est demandé en haut à droite avant de cliquer sur "Play".



FIGURE 10 – Ecran d'Accueil

Lorsque le bouton "Play" est cliqué, le menu s'affiche avec le choix du niveau au centre et le pseudo entrée en haut à gauche.



FIGURE 11 – Ecran du Menu

Lorsque le niveau est cliqué, la grille est affichée au centre, avec le pseudo en haut à gauche, le score et le temps en haut à droite.

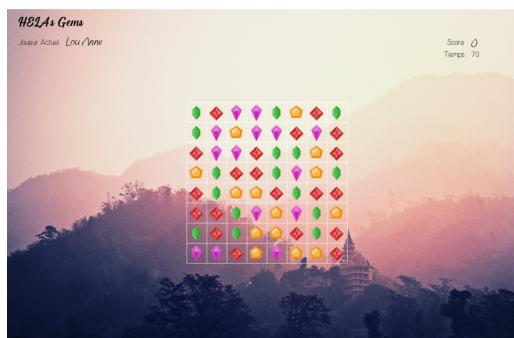


FIGURE 12 – Ecran de la Partie

Lorsque le temps se termine, un texte "Fin du Jeu" s'affiche avec le nom du jeu en dessous, le pseudo et le score en haut à gauche.



FIGURE 13 – Ecran de Fin

12 Diagramme des classes

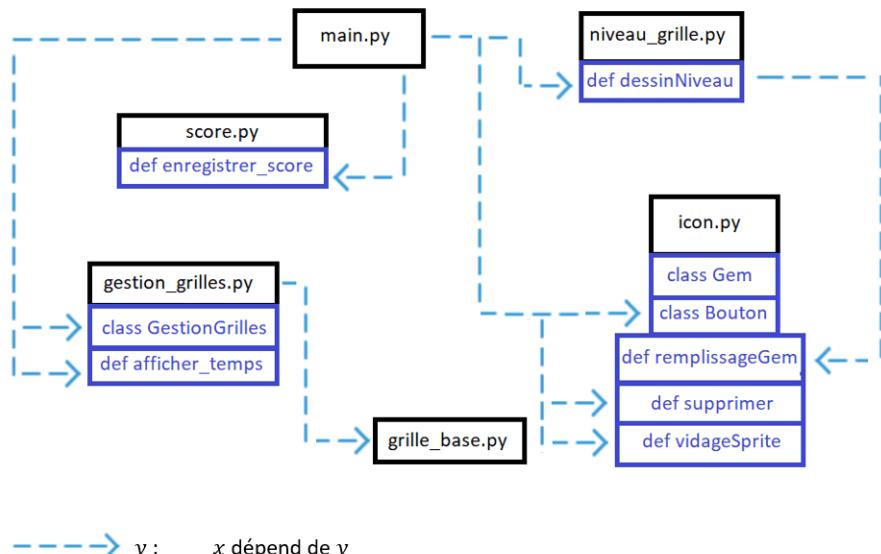


FIGURE 14 – Diagramme des classes

13 Structure du Jeu

Nous avons donc un fichier "main.py" d'où le jeu débute.

Le fichier "niveau_grille.py" regroupe les fonctions qui permettent de dessiner la grille en fonction du choix du niveau, comme la fonction "def dessinNiveau".

Le fichier "icon.py" possède toutes les gemmes et les boutons sous forme de "sprite", dans des classes "Gem" et "Bouton". Ce fichier possède aussi des fonctions qui permettent de supprimer les boutons une fois qu'ils ont été cliqués. Ce sont les fonctions "def supprimer" et "def vidageSprite". La fonction "def remplissageGem" sert à remplir la grille de gemmes.

Les fonctions qui permettent de faire fonctionner le jeu (permutation, élimination des gemmes... etc) se trouvent dans le fichier "gestion_grilles.py", dans la classe "GestionGrilles". Ce fichier contient également la fonction "afficher_temps" qui créé et affiche un chrono. C'est aussi de ce fichier que le score se calcule.

Le fichier "score.py" contient la fonction qui compare deux scores et garde le meilleur grâce à un dictionnaire.

Il y a aussi un fichier "score.txt", qui enregistre le score de chaque joueur.

14 Conclusion

Le jeu se lance parfaitement avec un petit temps de latence par moment. L'apparence du jeu et son fonctionnement corresponde aux attentes que nous avions.

Avec un peu plus de temps, nous aurions pu créer une fonction qui affichait le meilleur score de chaque joueur avec le classement de tous les joueurs. Nous aurions pu aussi retravailler le chrono pour qu'il se lance après avoir lancé le niveau.