MARATRAT Alexandre, ABARA Elias, COURTE Hugo, MURA Florian

Mini-Projet d'informatique : *MyPuzzle*

I.Cahier des charges du programme

L'utilisateur choisit l'image qu'il souhaite en la téléchargeant au format jpeg depuis un moteur de recherche puis il sélectionne le chemin menant à cette image sur la fenêtre de lancement du jeu. Une image différente peut donc être ouverte à chaque nouvelle partie. Après avoir choisi un pseudo, un niveau de difficulté…, le jeu se lance en appuyant sur le bouton jouer et l'image choisie est découpée en un nombre de pièces qui dépend de la difficulté choisie. Dans la fenêtre de jeu, l'utilisateur a alors une image constituée des morceaux de l'image initiale de manière désordonnée ainsi qu'un emplacement pour la reconstituer. Un timer est lancé pour suivre la progression du joueur afin de lui attribuer un score en fin de partie. Il peut ainsi sélectionner avec la souris les pièces du puzzle pour les placer au bon endroit. Si une pièce est au mauvais endroit, il n'a qu'à cliquer dessus pour la remplacer dans le tas de pièces aléatoires. Si l'utilisateur bloque, il peut à tout moment demander un indice grâce à un bouton indice qui le redirige vers une autre fenêtre (Fenetre_Indices dans le code). Dans cette fenêtre, il doit répondre à un quizz. Parmi les quatre propositions, une seule est correcte et si l'utilisateur sélectionne la bonne, il recevra des points bonus qui lui permettront d'accéder aux indices (placement d'une pièce choisie ou affichage de la réponse pendant une courte durée).

La réponse du puzzle s'affiche durant une courte durée puis disparaît et l'utilisateur est redirigé vers la fenêtre du jeu. Il peut également, en l'échange d'une diminution de ses points en fin de partie, demander à voir le puzzle pendant une certaine durée.

Celui ci pourra utiliser autant de fois l'indice qu'il veut (tant qu'il possède des points bonus). Néanmoins son score à la fin de la partie en sera diminué. En effet, une fois le puzzle résolu, une fenêtre de fin de jeu s'affiche avec le score de l'utilisateur qui dépend de plusieurs facteurs qui seront détaillés par la suite.

II.Description du problème posé

La principale difficulté rencontrée durant notre projet était la manipulation et le découpage d'une image que nous n'avions jamais abordé auparavant à savoir comment l'utilisation du BufferedImage notamment.
Il a également fallu gérer et coordonner les quatre différentes fenêtres pendant leur création, en effet chaque fenêtre nécessitait des éléments qui se trouvaient dans d'autres fenêtres, il fallait donc être en communication permanente pour savoir qui modifiait quel fichier à tout moment, et l'utilisation de RapidSVN en début de projet nous a bien aidé à résoudre ce problème. Chaque version était en ligne et était récupérable à tout moment. RapidSVN permettait également de fusionner les fichiers modifiés par deux personnes à des endroits différents.
Nous avons rencontré beaucoup de soucis importants de prime abord : impossibilité de déplacer la pièce, disparition du tableau aléatoire, algorithme de validation inefficace qui ont pris beaucoup de temps à être résolus alors que souvent ils résultaient de petites erreurs. La coopération a permis de prendre du recul sur le programme et de déceler les erreurs fondamentales de codage. Quand on est seul face à son programme, on réfléchit toujours de la même façon, ce qui ne permet pas de résoudre les problèmes. Il paraît donc évident que la coopération dans un projet informatique est nécessaire.

III.Fonctionnement de l'algorithme

Dans ce paragraphe seront décrites les méthodes les plus importantes de notre code et qui nous ont demandé le plus de travail.
Tout d'abord pour découper l'image en puzzle, on utilise la méthode *DecoupeEasy* qui renvoie un tableau de Pièce (objet extends d'un JPanel). On prend chaque image que l'on met dans une pièce (classe héritière de JPanel) et on fait un tableau de JPanel avec toutes les pièces.

On utilise ensuite la méthode *AddTabJPanel* qui permet, en prenant comme paramètres un tableau de Pièces, un JPanel, une hauteur, une largeur et une taille (la taille est le nombre de cases sur une colonne ou une ligne, taille*taille donne le nombre total de pièces) : elle permet à partir du tableau de pièces de créer un Jpanel les contenant.

La méthode *MelangePuzzle* nous a posé aussi quelques problèmes de conception mais était essentielle au fonctionnement du programme. Elle permet de créer à partir d'un tableau de pièces placé en paramètre (représentant l'image à découper), de créer un nouveau tableau où les pièces sont ordonnées de manière aléatoire et avec lequel interagit l'utilisateur.

La méthode *DeplacerPiece* permet à un utilisateur de cliquer dans un premier temps sur une pièce du tableau aléatoire. Si l'utilisateur clique sur une pièce du tableau vide, elle sera alors transférée dans le tableau vide. Cela permet de faire progresser le jeu. On rafraîchit la fenêtre quand c'est fini.

La méthode *Recommencer* devait servir à recommencer une partie mais les délais ne nous ont pas permis de la rendre fonctionnelle. Si on appuyait dessus, les pièces n'étaient plus déplaçables. Ceci est dû à un problème de MouseListener.

La méthode *refresh* est très importante car elle permet d'actualiser visuellement les puzzles en reconstruisant les JPanel à partir de leur tableau de pièces (AddTabJPanel).

Enfin*, VérifPuzzle* est la méthode permettant de vérifier que le puzzle est fini. Pour cela on vérifie que les cases sont placées dans l'ordre croissant grâce à leur attribut num qui permet de suivre leur placement initial et ainsi le comparer. On place donc ces numéros dans un tableau 1D que l'on parcourt puis si les termes sont dans l'ordre croissant

Calcul du score : elle prend en compte trois facteurs: le temps mis pour reconstituer le puzzle, la difficulté choisie (pas eut le temps) et l'utilisation des indices (chaque demande d'indices diminue le score et s'ajoutent en plus de malus en cas de mauvaises réponses à la question permettant d'accéder à l'indice. Pour que cette fonctionnalité soit la plus juste possible, il a fallu trouver des compromis. Par exemple un joueur résolvant un puzzle de 9 pièces ne devrait pas avoir un meilleur score qu'un autre qui terminera le 16 pièces même si le temps mis est légèrement supérieur. Il a également fallu estimer l'influence des indices dans la vitesse de résolution du puzzle afin que ceux ci ne désavantagent ou n'avantagent pas trop l'utilisateur.

IV. Structuration des données

Pour la structuration des données on s'est notamment appuyé sur l'héritage. En effet, on peut retrouver une classe *Fenetres* dans notre dossier qui est une classe abstraite permettant de faire circuler les différentes variables nécessaires pour plusieurs fenêtres, chaque fenêtre hérite donc de la classe abstraite *Fenetres* où sont contenues les différentes variables.

Pour l'ensemble des panneaux de la fenêtre de jeu principale, nous avons fait le choix, quitte à faire peut être plus de calculs qu'avec une Array ou LinkedList, d'utiliser des **tableaux 2D** de JPanel car cela nous paraissaît plus instinctif et facile à visualiser pour des pièces de puzzle.
Pour la fenêtre "Fenetre_Indice", nous avons privilégié ArrayList pour manier les JButton car nous allons beaucoup sélectionner des éléments de manière **aléatoire** dans ces listes.

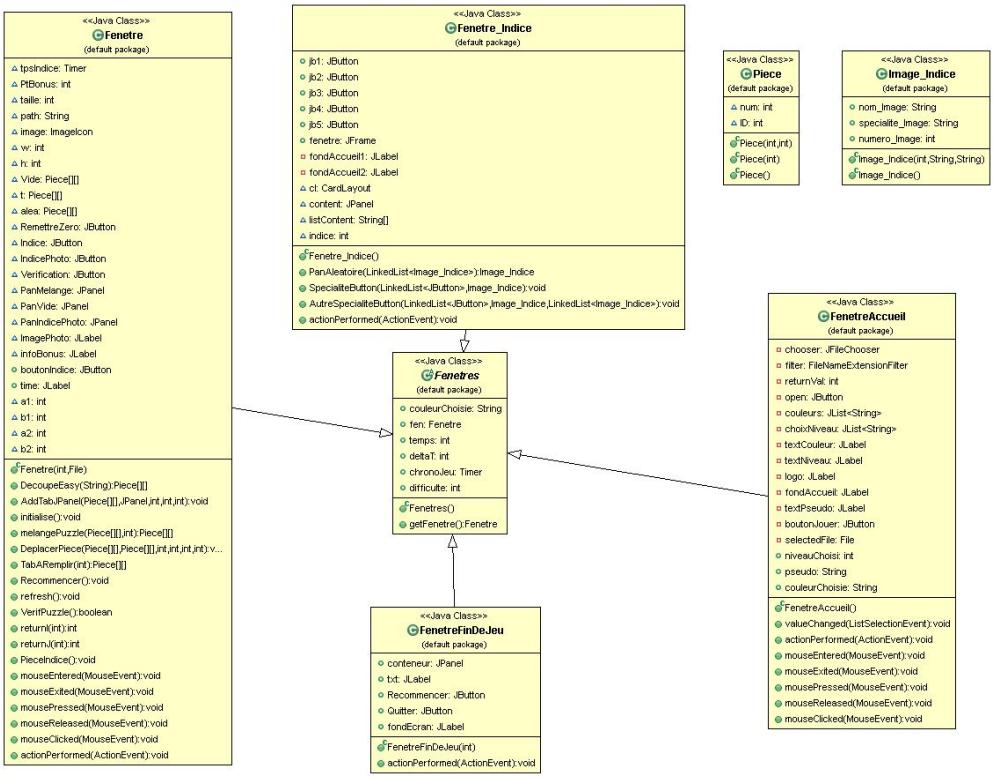


Diagramme UML

V.Suggestions d'améliorations du projet

Bien que les objectifs que nous nous étions fixé au début semblent avoir été atteints, de nombreuses corrections et améliorations seraient à faire :

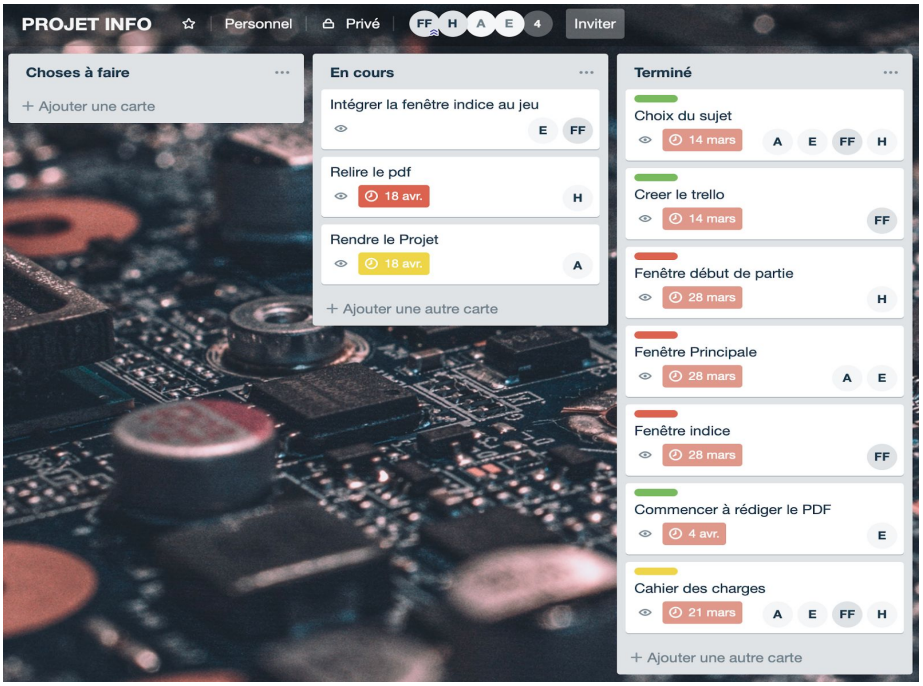
Corrections/bug	Améliorations
Supprimer les bandes blanches horizontales qui apparaissent lors du découpage du Puzzle liées à la taille de l'image choisie (qui n'est pas multiple de l'entier « taille » donc il y a du tronquage	**Enrichir le quiz** permettant d'accéder à l'indice (on aura vite fait le tour au bout de quelques parties) et supprimer les questions ayant déjà été traitées dans une partie si l'utilisateur a répondu juste.

On pourrait par exemple créer une nouvelle méthode qui recadrer l'image choisie pour que celle-ci soit à la bonne taille.	
Dysfonctionnement de la méthode *recommencer* de la fenêtre principale de jeu "Fenetre"> produit un bug qui ne permet plus de déplacer les pièces	**Sélection des pièces plus interactive** en intégrant du drag and drop pour déplacer les pièces
Le jeu n'est pas très réactif, si l'on sélectionne trop vite une pièce et sa destination, elle ne bouge pas	Possibilité de **mettre le jeu en pause** à tout moment
il est **possible de sélectionner plusieurs réponses dans le quizz** pour trouver la bonne ce qui ne devrait pas être possible	**Intégrer du traitement d'image** (par exemple, rendre l'image indice floue ou sombre pour ne pas trop aider le joueur)
Il ne se passe rien quand on appuie sur le bouton vérifier et que le puzzle est faux	**Ajouter une limite de temps**: le joueur perd si il n'arrive pas à résoudre le puzzle avant
La partie concernant l'ajout de points bonus en répondant au quizz a été abandonnée suite à un problème de variables inter-fenêtre : malgré plusieurs tentatives vaines, nous n'arrivons pas à proposer quelque chose de stable. Nous avons donc gardé le principe de bonus en donnant une quantité initiale de point d'indice qui défavoriseront le score final. (la partie quizz est néanmoins consultable sur l'archive)	Intégrer la possibilité à l'utilisateur de **changer de fichier**
	Intégrer le **niveau de difficulté dans le calcul du score**

VI.Carnet de route et échéancier

Au début du projet, nous avons essayé de nous répartir de façon équitable le travail. Etant donné que nous sommes quatre et que notre programme comprend quatre fenêtres, nous avons travaillé chacun sur une fenêtre au début mais nous nous sommes rapidement rendu compte que les fenêtres ne seraient pas de la même difficulté à coder : la fenêtre *Fenêtre* étant bien plus compliquée en faisant appel à toutes les méthodes complexes à coder. Cependant une fois toutes les autres fenêtres finies, nous nous sommes focalisés sur la fenêtre principale.

Afin de suivre l'évolution du travail et de savoir précisément qui doit faire quoi, nous avons choisi d'utiliser le logiciel Trello, en accès gratuit, ses fonctionnalités sont largement suffisantes. On a donc créé au départ des cartes avec des tâches à effectuer attribuées à des personnes du groupe. (Voir ci-dessous état d'avancement à la fin du projet).



Cette méthode de suivi de l'avancement du projet nous a permis de suivre en temps et en heure ce que nous devions faire chaque semaine.

VII.Bibliographie et Webographie

- La JavaDoc : https://docs.oracle.com/javase/7/docs/api/
- Les diapositives de cours (S2,S3,S4) : https://moodle.insa-lyon.fr/course/index.php?categoryid=173
- Openclassroom (cours et forum) : https://openclassrooms.com/fr/
- Developper.net : https://www.developpez.net/forums/