COMPTE RENDU PPOOIG – ASSO ALI MULLUD, FELIX OLLIVIER

Notre projet PPOOIG remplit l'intégralité du cahier des charges minimales. En effet, une fois le jeu lancé, une fenêtre s'ouvre, nous permettant de choisir le jeu (dominos ou Carcassonne), le nombre de joueurs, et sélectionner le nombre d'IA parmi les joueurs.

Notre jeu de dominos est complet, autant sur le terminal qu'avec une interface graphique :

Lorsque l'on lance une partie de domino avec interface graphique, nous obtenons une fenêtre affichée en grand écran, sur laquelle on peut observer deux parties :

- la partie gauche servant aux statistiques et aux choix du joueur, c'est-à-dire les points de tous les joueurs, le nombre de tuiles restantes dans le sac, mais aussi les boutons pour piocher, tourner une tuile (désactivé tant qu'une tuile n'est pas affichée), ou abandonner.
- La partie droite est un plateau de 10 cases par 10 cases où se trouve déjà une tuile, située aléatoirement à chaque nouvelle partie.

Le jeu de domino a été source de plusieurs problèmes pendant notre projet :

- Tout d'abord, force est de constater que l'esthétique actuelle de ce jeu est bien plus agréable que lorsqu'à nos débuts du projet. Nous avons utilisé un combiné de plusieurs JPanel, permettant ainsi de placer les différents éléments où nous le souhaitions. Cependant, cela n'a pas été une simple tâche, il fut très ardu de déterminer la bonne structure, étant donné les faibles possibilités que nous donnent les JPanel. Nous avons eu besoin de nous représenter la forme à l'aide d'un schéma, que nous avons joint à notre archive. Il a été également difficile de faire apparaître correctement la tuile piochée à cause de ce même problème de structure qui était imparfaite.
- Nous avons également rencontré un problème assez contraignant lorsque nous voulions poser une nouvelle tuile sur le plateau : la tuile n'affichait qu'un seul tableau sur les 4. Nous avons ensuite compris par la suite qu'il fallait utiliser la fonction removeAll() sur les différents panel afin de la voir entièrement.
- A la base, nous souhaitions créer un plateau de Tuile, mais nous nous sommes rendus compte que nous ne parviendrons pas à créer un plateau de la sorte,

étant donné qu'au début de la partie les tuiles sont nulles. Ainsi nous avons décidé pour remédier à ce problème de créer un tableau non pas de Tuile mais de Case, et chaque case possédaient un attribut Tuile qui pouvait être null. Ainsi lorsque nous comparons deux tableaux de Tuile pour poser une nouvelle tuile sur le plateau, nous comparons en réalité des tuiles de cases.

- Un autre problème concernant la pose d'une tuile était que lorsque nous souhaitions poser une tuile sur un bord, notre code renvoyait des erreurs, expliquant que les tuiles adjacentes étaient null (en parlant donc de la tuile inexistante sur le plateau). Après plusieurs heures de recherche nous avons fini par résoudre le problème en créant différentes conditions dans les fonctions pour ajouter une tuile et modifiant légèrement le plateau : nous avons créé une case supplémentaire à côté de tous les bords du jeu, ainsi la case adjacente ne sera pas null tandis que la tuile le sera (ce qui était le problème rencontré), ainsi pas besoin de vérifier de tableaux adjacents.
- Mais résoudre un problème en crée un autre : maintenant que nous avions un tableau de taille 12 par 12 et non pas 10 par 10, il était beaucoup plus difficile d'associer un numéro à la case que nous souhaitions remplir. Autrement dit, la première case du plateau visible était en réalité la 14^{e}. Ceci a engendré beaucoup de remises en question concernant le format adéquat à utiliser pour notre plateau. Mais nous sommes finalement parvenus à modifier les numéros des cases à l'aide de coordonnées modifiées : si le numéro de la colonne n'est pas 10, alors le numéro correspond aux coordonnées du plateau (ainsi la 14^{e} case est en réalité la case du plateau de coordonnées [1][1], donc on associe la case au numéro 11 ; si en revanche le numéro de la colonne est 10, alors on l'inscrit comme étant 9 et on ajoute 1 au numéro. Ainsi la 20^{e} case est sur la case [1][10] du plateau, donc on l'associe au numéro 19+1, soit 20.

Malgré ces différents problèmes, certains points nous ont en revanche parut plus simple qu'ils n'en avaient l'air :

- L'implémentation du code pour faire fonctionner les IA ne nous a posés aucun problème apparent. En effet nous n'avons passé que très peu de temps sur cette particularité du projet, et elle a tout de suite bien marché.
- La création du jeu de domino dans le terminal nous était également assez intimidante, par exemple pour l'affichage du plateau et des différentes fonctionnalités (sans interface graphique, aucune ergonomie, etc.), mais finalement, nous avons été très efficace sur ce point. Nous avions également peur que le plateau soit illisible, mais finalement nous avons trouvé une méthode très ergonomique et agréable à lire pour l'utilisateur, et ce assez rapidement.

Il est évident que nombreux de ces problèmes ont aussi été présent sur notre jeu Carcassonne, dont nous allons étayer les propos ci-dessous.

Notre jeu de Carcassonne remplit l'entièreté des fonctionnalités demandées par le cahier des charges. En effet, lorsqu'on lance une partie de Carcassonne, on arrive sur un plateau très similaire à celui des dominos (une tuile est déjà placée), et le fonctionnement est également le même. Malgré la similarité avec les dominos, nous avons rencontré de nombreux problèmes pendant le développement de ce jeu :

- Le premier problème évident est la différence entre afficher de simples chiffres à l'aide de JLabel et une image entière. Ceci nous a valu un bon moment de recherche, jusqu'à trouver la fonctionnalité ImageIcon qui nous permet d'afficher proprement une image.
- Mais le prochain problème était de la tourner, ce qui ne nous paraissait pas possible avec ImageIcon. C'est alors qu'après plusieurs heures de recherche, nous avons découvert la classe AffineTransform, qui nous a permis d'utiliser sa fonction rotate(). Nous avons donc créé un objet AffineTransform avec en argument notre image, puis nous avons utilisé rotate(), et enfin notre ImageIcon initiale est remplacée par cette nouvelle image tournée.
- Nous avons rencontré un léger problème lorsque nous ajoutions des tuiles sur le plateau, qui était que les dimensions de l'image n'étaient pas bonnes, d'un côté car l'image était trop grande par rapport à la tuile, mais aussi parce que lorsque nous tournions la tuile, comme nos cases étaient initialement rectangulaires, cela causait un problème de forme. C'est donc à ce moment la que nous avons changé d'affichage en mettant le JPanel jeu non pas en BorderLayout comme il l'était avant, mais en GridLayout, afin d'accorder autant de place à la partie gauche des statistiques qu'au plateau. Cela nous a donc donné un plateau avec des cases carrées, et nous avons également réduit les dimensions des images (concernant leurs pixels), afin d'obtenir la même définition pour les cases et les images.
- Le dernier problème, et probablement l'un des plus ennuyeux, a été l'implémentation des pions. Nous n'avons malheureusement pas trouvé d'autre moyen que d'afficher sur le côté des statistiques la case et la position des pions posés par le joueur courant, l'implémentation des pions sur les tuiles étant hors de nos connaissances, car il s'agit d'ajouter une image sur une image, ce que les JPanel ou ImageIcon (etc.) ne permettent pas. Ainsi nous avons dû réfléchir à une autre façon de procéder, et nous en sommes arrivés à la conclusion qu'afficher les pions sur le côté des statistiques avec toutes ses données (numéro de la case et position sur la tuile) était la meilleure option qui s'offrait à nous. Cependant cela n'a pas exclu un nouveau problème de structure de JPanel, qui nous a coûté légèrement en temps.

Tout comme pour les dominos, nous avons également été agréablement surpris par la rapidité de certains points sur lesquelles nous sommes passés :

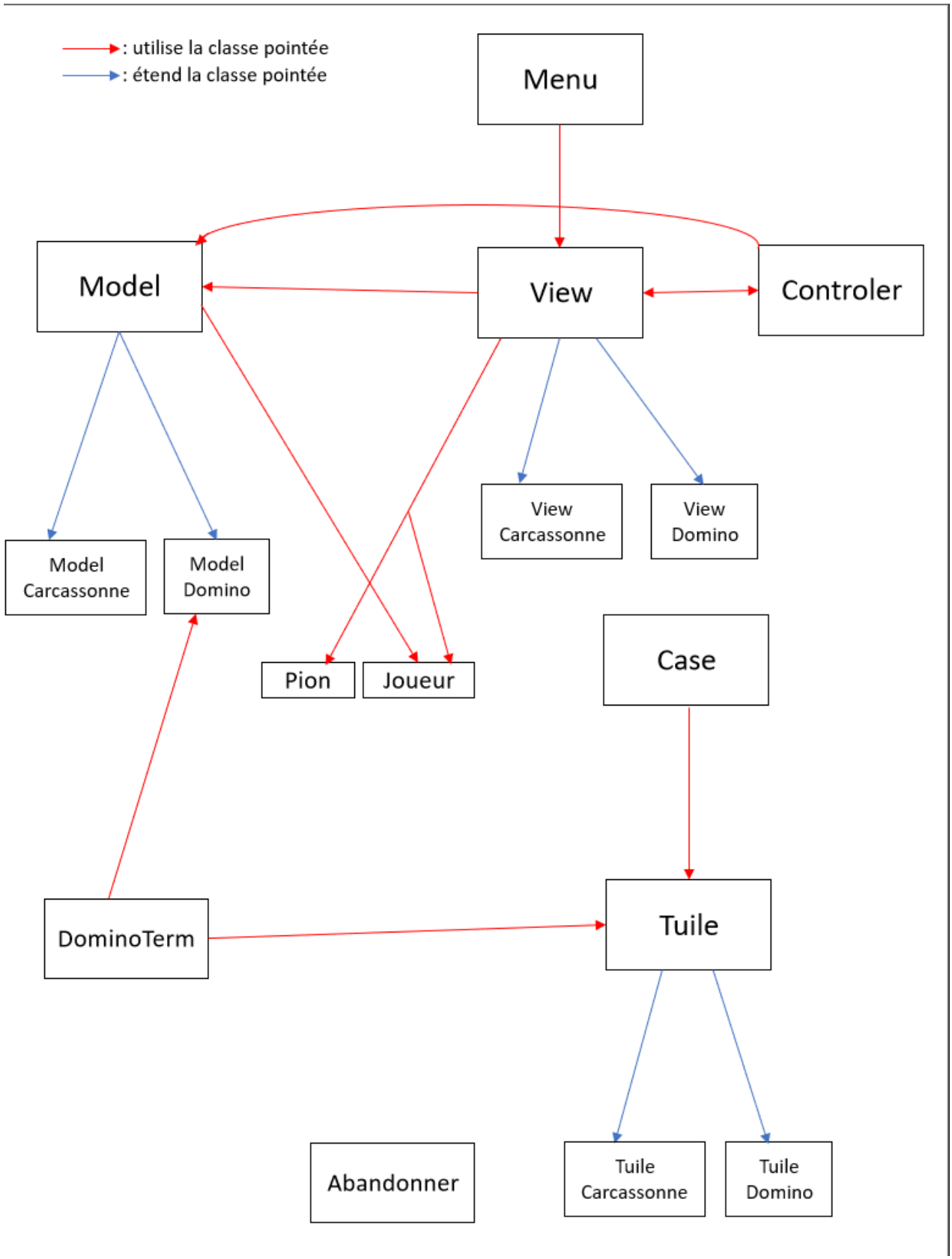
- Une fois le jeu de dominos terminés, nous avons été très rapides sur le début de Carcassonne et très efficaces sur la réutilisation de code, comme ce qui était demandé.
- Les tuiles ont étonnement été beaucoup plus simples à implémenter, dans le sens où contrairement aux dominos où il faut regarder en détail tous les chiffres des tableaux pour voir s'ils correspondent, nous n'avions ici qu'une simple chaîne de caractère à comparer (« route », « pré », « ville » etc.).

Nous avons pensé à ajouter 2 fonctionnalités supplémentaires à notre projet :

Tout d'abord, nous souhaitions implémenter l'extension HAL 9000 (l'IA meilleure que le hasard). La fonctionnalité est assez implicite, mais notre compréhension de cette dernière réside dans le fait que le bot doit obligatoirement placer sa tuile s'il existe une case où sa tuile est plaçable. Ainsi, pour réaliser cette extension, il suffit lors de la définition de la méthode pour qu'une IA joue un tour, de modifier le fait qu'elle possède 50 essais pour la poser à « parcourir tout le tableau en tournant 4 fois sa tuile et étudier à chaque fois si le placement est possible ou non ». Ce code se crée très facilement, avec notamment une double boucle for pour parcourir tout le tableau, et à chaque case on vérifie s'il est possible de poser la tuile, et si ce n'est pas possible on tourne la tuile et on réessaye, le tout 3 fois.

La deuxième extension que nous souhaitions ajouter à notre projet est Hexa-Carcassonne. Cette extension réside dans la création du même jeu mais avec des tuiles hexagonales. Pour réaliser cette extension, il nous suffirait de créer un plateau avec des cases hexagonales, ce qui est possible en utilisant la classe Graphics et en utilisant la fonction drawPolygon. Puis dans la classe TuileCarcassonne, il suffit de rajouter 2 String pour les deux côtés ajoutés, et lorsque l'on pose la tuile on vérifie si tous les côtés adjacents sont concordants.

Voici la représentation graphique du modèle des classes de notre projet :



Ce projet a été pour nous le moyen de découvrir d'une part un grand nombre de nouvelles méthodes de développement ainsi que l'étendue des possibilités de projets faisables avec Java et ses différentes bibliothèques, et d'autre part de créer pour la première fois un projet complet entièrement créé par nos soins, qui nous donne un sentiment de satisfaction et de fierté après avoir vu une mise en forme conforme à nos attentes. Merci d'avoir accordé ce temps à l'étude de notre rapport de projet, nous vous agréons, Madame, Monsieur, l'expression de nos salutations les plus distinguées.

M. Félix Ollivier, M. Asso ALI MULLUD