Projet de POOIG: Jeux dominos

Pour la conception nous avions commencé par une approche assez proche des jeux réels, nous sommes concertés et avions décidé de définir les pièces de domino « solides » dont on pourrait par exemple changer l’orientation avant de les poser sur un plateau. Nous nous sommes réparti les tâches et avons commencé à construire un squelette de code qui permettrait de faire certains tests et de nous familiariser avec ce que nous aurons à faire.

Après un certain temps nous avons vu que la structure de notre code n’était pas suffisamment modulable pour pouvoir étendre ce que nous avions développé pour les différents types de pièces qui sont nécessaires au différents jeux et particulier le domino gommette qui avait 6 connexions par pièces rendant la représentation textuelle quasi impossible avec notre code.

Nous avons ensuite réellement recommencé à la conception avec une nouvelle approche plus dynamique, cette fois nous voulions avoir comme classe parente des différentes pièces une classe avec comme seul attribut les connexions. Cette fois ci nous nous sommes efforcés à avoir une structure du codes en couches successives qui permettrait non seulement d’interagir avec les objets indirectement avec les règles dissimulés dans les méthodes qui ajouteraient les pièces ensemble non seulement pour faciliter la construction de l’interface graphique et textuelle mais aussi parce nous nous étions dit qu’il serait possible si nous avions le temps d’ajouter des joueurs adverses avec quelques éléments de hasard pour le choix du placement des pièces



Attributs de Connexion.java

Chaque connexion fonctionne similairement à une liste chainée, serait unique et comporterait l’information sur la valeur que la connexion pouvait accepter, la connexion qui lui serait associé quand deux pièces sont reliées ainsi que la pièce à laquelle la connexion se rattache. Ainsi la valeur acceptante peut être différente que la valeur qu’aura la pièce, ce qui est utile pour domino gommettes avec la forme et la couleur pour chaque valeur mais surtout pour le puzzle où nous avons décidé de donner à chaque pièce une valeur qui indique quelle image doit être associée à la pièce et un lien unique entre deux voisins ce qui permet une seule façon de poser les pièces. L’attribut parent ne changera pas donc son implémentation dans le code sera facile puisqu’il est donné automatiquement quand une pièce est créée.

Revenons aux pièces définies dans Piece.java, cette classe parente des différents types de pièces qui seront nécessaires aux jeux n’a comme seul attribut une liste de connexion dont la longueur n’est pas fixée, ainsi nous avons pu écrire toutes les méthodes qui gèrent les pièces dans Piece.java sans avoir peur de devoir les modifier beaucoup quand on créait de nouvelles classes filles pour les différents types tout en permettant au déroulement du jeu de ne pas prendre en compte les valeurs des pièces qui seraient uniquement pour les différentes interfaces. Ainsi nos classes filles pour Piece.java : PieceD, PieceP, PieceG pour les pièces respectivement du domino normal, le puzzle et le domino gommettes ne contiennent que les constructeurs utilisant des méthodes statiques.

La couche suivante est celles des groupes de pièces dont la classe parente est GroupePiece.java, cette famille de classe sert à la gestion en suivant les règles des connexions possibles en fonction des différents jeux. Ici la décision avait été prise pour garder notre approche dynamique était d’avoir une seule pièce racine à partir de laquelle un ensemble de méthodes donneraient les connexions libres et celles avec lesquelles une pièce pouvaient être connectés ainsi que des méthodes qui récupèreraient l’ensemble des pièces qui seraient ensuite formattés pour être affichables. Cette idée d’avoir un objet contenant un groupe de pièces connectés permettrait aussi d’avoir pour le puzzle différents groupes de pièces affichées en même temps ce qui se rapprocherait plus de l’expérience du jeu.