Projet de POOIG

Les dominos, et la famille des jeux qu'on peut leur associer, est la source d'inspiration de ce sujet.

- Le projet est à faire en binôme. Les monômes sont interdits sauf pour des questions de parité. Vous pouvez aussi vous associer à quelqu'un qui n'est pas de votre groupe de TD (utilisez le forum dédié sur moodle pour entrer en contact). Il n'y aura absolument pas de trinômes, et il est entendu que si des travaux se ressemblent trop nous prendrons les sanctions qui s'imposent.
- Il vous faut déclarer la constitution de votre binôme avant le 9 novembre (sur moodle dans la section projet)
- La note de soutenance pourra être individualisée, chacun doit donc maîtriser l'ensemble du travail présenté.
- La soutenance aura lieu sur la période prévue pour les examens, et votre travail sera à rendre quelques jours avant. Nous vous donnerons les dates exactes lorsque les réservations seront confirmées.
- Sauvegardez régulièrement votre travail, et lorsque vous envisagez une modification importante conservez bien la version antérieure.

I) Introduction

Dans la suite de ce document vous trouverez la présentation de plusieurs jeux et de certaines variantes. Il vous faut bien comprendre que l'objectif du projet est de développer une sorte de plateforme qui permette la mise en commun d'un ensemble de concepts que l'on retrouve dans cette famille de jeux. Ainsi on pourra réutiliser un maximum de code pour écrire un jeu ou un autre comme une extension. Ayez bien en tête cet objectif. On souhaite que vous soyez capables de dégager des concepts utiles à la résolution des problèmes, à modéliser en décomposant en objets et méthodes. Ce projet doit vous servir à illustrer au maximum les notions vues en cours.

Il nous semble que ces jeux ont suffisamment de points communs pour pouvoir faire un travail de modélisation intéressant. Relevez bien leurs points communs et leurs différences.

Concernant les affichages, vous pouvez commencer par proposez une solution en mode texte, puis isoler les affichages en mettant en oeuvre une architecture qui sépare la vue et le modèle. Plus tard au fur et à mesure de l'avancement du cours sur les interfaces graphiques, vous pourrez proposer un autre type de vue intégrant le graphisme.

Il est important de se concentrer tout d'abord sur la modélisation du jeu. En règle générale compartimentez votre code, documentez le ...

II) Le jeu des dominos

Le jeu de base consiste à déposer de façon linéaire des pièces qui se correspondent bord à bord. Elles sont en nombre limité, prédéfinies et initialement réparties entre plusieurs joueurs. Les questions de savoir qui commence, comment le jeu se termine et que faire lorsqu'un joueur ne peut pas jouer font l'objet de plusieurs variantes :

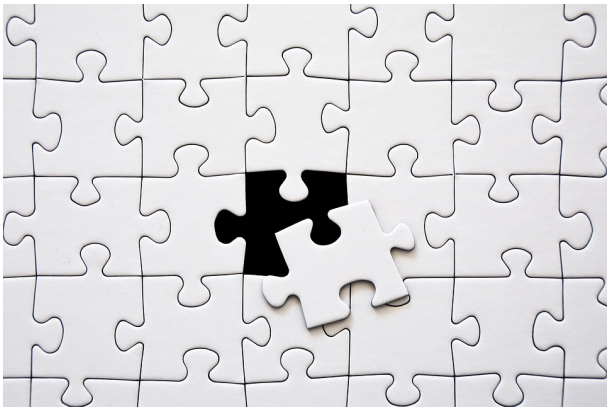
- pour commencer, on peut choisir le joueur le plus jeune, ou celui qui possède le double le plus grand, piocher une pièce et décider en fonction du total des points, etc ...
- Lorsqu'un joueur ne peut pas jouer on peut décider d'arrêter le jeu, de piocher parmi les pièces non distribuées, ou, pourquoi pas, imaginer de prendre une pièce chez un autre joueur, pourvu que la règle soit fixée au départ.
- En général le gagnant est celui qui réussi à placer toutes ses pièces en premier, mais on peut aussi décider de s'arrêter lorsqu'un joueur est bloqué et classer les joueurs en fonction du total des valeurs des pièces.

III) Variantes dans le plan

a) Domino-gommettes & Puzzle

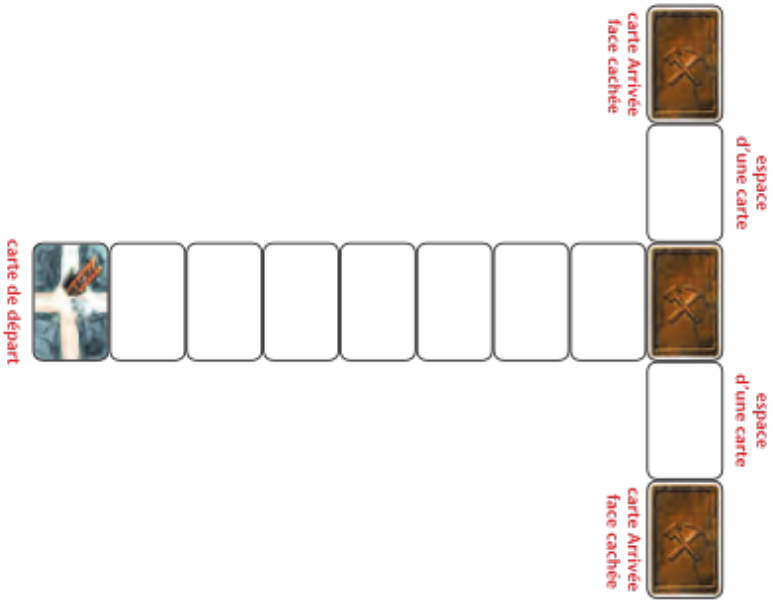
Dans le cas des *dominos-gommettes* l'association se fait selon la forme ou la couleur, ce qui permet d'occuper l'espace différemment.

Sans grands efforts d'imagination on peut aussi constater qu'un puzzle peut être vu comme une forme de jeu de cette famille.



b) Le *Saboteur*

Dans ce jeu on fixe sur la table une position de départ (à gauche sur la figure de la page suivante) et des positions d'arrivées (les trois à droites). Sur le dessin on a reporté les distances avec des espaces blanc, mais toute la surface de la table fait partie du jeu.



Les joueurs ont en main des cartes, certaines peuvent être posées et d'autres correspondent à des actions de jeu. Sur les cartes posables sont représentés des portions de routes, qui doivent se correspondre bord à bord pour que les routes se prolongent harmonieusement.

Aux positions d'arrivées on prétend qu'il y a des trésors d'une valeur cachée. Dans la version simplifiée qui nous suffira ici, lorsqu'un joueur réussi à connecter par une route le départ avec l'une des arrivée, il prend alors connaissance de sa valeur et collecte ce trésor.

Initialement toutes les cartes sont mises dans une seule pioche, et chaque joueur en prend 5 au hasard pour constituer sa main. A son tour il devra en jouer une. Selon le cas il pourra :

— la poser sur la table pour continuer une route
— la jeter définitivement
— s'il s'agit d'une carte action, l'imposer au joueur de son choix pour le perturber (ou l'aider)

Pour finir son tour, il complète sa main en piochant une carte s'il en reste dans la pioche, et on passe au joueur suivant.

Les cartes actions ont pour but de retarder les autres joueurs en les empêchant de progresser vers les positions d'arrivées ; elles permettent également de parer ces attaques. Il existe trois type de sabotages : pour le transport, les outils, ou l'éclairage, et trois possibilités d'effectuer les réparations correspondantes.

Dans un jeu standard il y a 4 copies de chaque carte action possible, et disons 44 cartes du type chemin. Vous en trouverez des images sur moodle pour vous permettre de faire les affichages correspondants.

Remarquez que les cartes sont des rectangles, elles n'ont donc que 2 orientations possibles. Remarquez également qu'il y a une propriété particulière à détecter : le chemin entre l'arrivée et le départ doit être une route qui permette effectivement d'y accéder.

IV) Séparer la vue du modèle

Le développement des règles du jeu est largement indépendant des aspects graphiques. Si on souhaite faire évoluer notre programme pour l'adapter avec une autre présentation, par exemple sur un téléphone ou une console, l'essentiel du jeu sera tout de même préservé, les changements à faire relèvent tous de ce qu'on appelle *la vue*.

Que le programmeur ait bien isolé l'ensemble du code concerné simplifiera grandement le développement ultérieur. C'est pourquoi on choisit, très classiquement, de séparer dès la conception les aspects relatifs à la *vue*, des aspects propres au modèle de données.

Dans notre cas, vous allez dans un premier temps tout faire en mode texte. Puis, lorsque vous serez plus avancé dans votre réflexion, que vous aurez quelques connaissances d'interface graphique, et que vous aurez mieux cerné ce que vous souhaitez apporter au projet il sera temps de redéfinir les vues, via une interface, ou une classe plus abstraite. L'essentiel est que vous sépariez modèles en général, et vues en général.

L'association entre les modèles et les vues se fera en introduisant un champs typé `VueGenerale` dans les modèles de votre jeu, et réciproquement si besoin. Pour rendre compte d'une modification, graphique ou textuelle, le jeu s'adressera à sa vue.

V) Travail demandé, rapport, soutenance

Vous devez réaliser tous les jeux décrits : domino classiques, domino gommette, puzzle, saboteur. Cela n'est possible que si vous développez votre code autour d'une architecture réutilisable, extensible, modulaire etc... Réfléchissez, sans les réaliser toutes, aux variantes des règles. Abordez cette question dans votre rapport.

La séparation de la vue et du modèle sera illustrée au moins une fois. C'est à dire que pour au moins un des jeux vous aurez réalisé à la fois une vue graphique et une vue textuelle.

Les travaux sont à rendre sur `Moodle` sous la forme d'une archive nommée *`nom1-nom2`* selon la constitution de votre binôme, et qui s'extraira dans un répertoire *`nom1-nom2`*. Elle devra contenir :

- les sources et ressources (images ...) de votre programme. Ne polluez pas votre dépots avec des fichiers `.class` inutiles ou d'autres
- Idéalement après avoir décompressé votre archive, nous devrions simplement faire : `javac Main.java` puis exécuter. Assurez vous que ce soit bien le cas, en particulier utilisez des chemins relatifs pour vos images.
- un fichier nommé `README` qui indique comment on se sert de votre programme
- un rapport au format *PDF* d'au moins cinq pages expliquant intelligemment les parties traitées, les problèmes connus, et les pistes d'extensions que vous n'auriez pas encore implémentées. Il devra contenir impérativement une représentation graphique du modèle. (Le plus simple est qu'elle soit manuscrite, puis scannée). Le rapport n'est en aucun cas une impression illisible de votre code.
- pensez à préparer toutes choses utiles pour rendre la soutenance fluide.
- la soutenance devra pouvoir se faire sur une machine du script à partir des sources que vous avez déposées. Elle se déroulera devant un ou deux enseignants dans un mélange de questions et de tests. Il pourra vous être demandé de modifier votre code pour répondre à une question spécifique.
- Enfin, organisez-vous pour avancer petit à petit, sans vous perdre complètement sur les aspects graphiques.