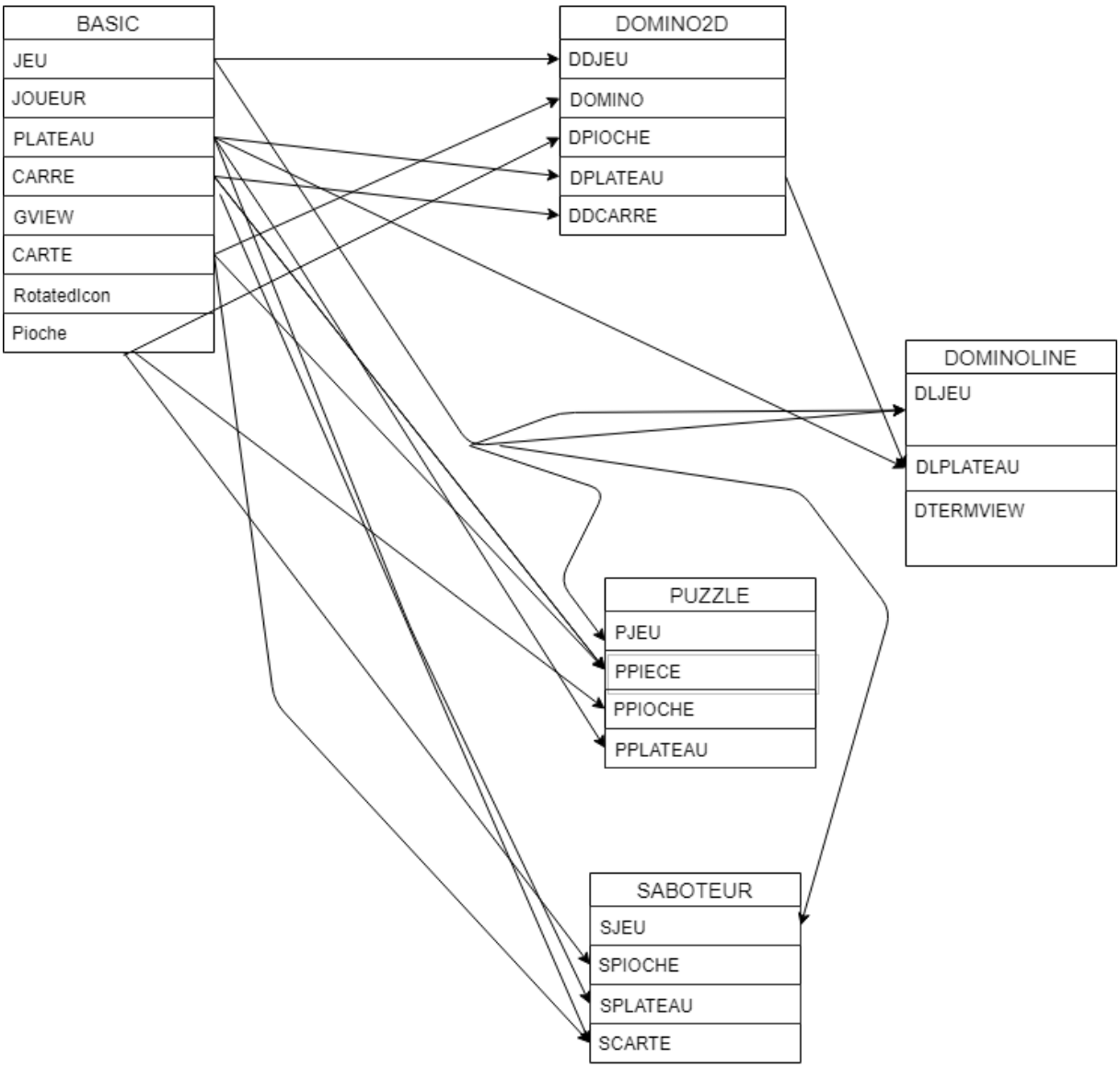
Julien BERSAUTER et Hugo SPINAT

RAPPORT

DIAGRAMME

Flèche = héritage / implémentation



Basic

Chaque autre jeu hérite et/ou implémente forcément des fichiers de BASIC. Il possède aussi la vue.
Plateau -> Défini un plateau de jeu, bidimensionnel.
Joueur -> Défini un joueur avec toutes les options nécessaires
Carte -> Défini une carte avec ses attributs
Carres -> Défini un carré (utile uniquement pour les Dominos)
Jeu -> Permet d'initialiser (et choisir ?) le jeu.
Pioche -> Correspond à la pioche « commune » entre les joueurs.
GView -> Correspond à la vue.
RotatedIcon -> Permet de faire tourner un Icon.

Domino

Le jeu a été très simple et sans difficulté. Il initialise un plateau de jeu, avec une seule dimension.

Chaque pièce est posée soit à l'extrémité droite, soit à l'extrémité gauche.

Un Domino est posé aléatoirement dès le départ.

L'ordre de départ des joueurs est l'ordre naturel d'ajout.

- ➔ Pose de manière linéaire des Dominos
- ➔ Dès qu'un joueur n'a plus de Domino ou que les joueurs sont bloqués, la partie est terminée.
- ➔ Les joueurs adverses ne peuvent pas voir vos dominos.
- ➔ Disponible en mode TERMINAL.

Difficulté : Aucune.

Domino_2D

Un peu plus complexe. Cette fois-ci, il est laissé la liberté d'essayer de poser une pièce là où on le souhaite. Nous avons donc fait là même chose mais les fonctions de vérifications s'assurent qu'on ne pose pas une pièce dans le vide.

Pose sur un espace en 2 dimensions (haut/bas, droit/gauche) des Dominos. Il est possible de tourner les dominos. Dès que tout le monde est bloqué ou qu'il n'y a plus de joueurs, la partie est terminée.

Les joueurs adverses ne peuvent pas voir vos dominos.

- ➔ Disponible avec une interface GRAPHIQUE.

SABOTEUR

Le saboteur nécessite plus de travail.

Trois trésors sont posés aux extrémités du terrain (haut droit, milieu droit et bas droit). Le trésor renferme un score numérique défini en début de partie. Les pièces sont unique et énumérées dès la création de la classe SCARTE.

Il nécessite de vérifier à chaque tour de jeu si le joueur actuel à pu avoir accès à un trésor.

- ➔ Pose sur un espace en 2 dimensions (haut/bas, droit/gauche) des cartes
- ➔ Dès que tout le monde est bloqué ou que tous les trésors sont découverts, la partie est terminée.
- ➔ Les joueurs adverses ne peuvent pas voir vos cartes.
- ➔ Disponible avec une interface GRAPHIQUE.

PUZZLE

A part pour le découpage, le puzzle a été plutôt simple. On peut poser sur n'importe quelle case disponible n'importe quelle pièce.

Le jeu prend une image en entrée, et construit automatiquement un puzzle, qui est découpé en un nombre de pièces définis par le joueur. Les pièces sont carrées pour faciliter le découpage.

Il est possible de poser ou de retirer une pièce. Il n'est pas possible de les faire tourner.

Dès que le puzzle est complet, dans l'ordre, c'est fini.

Le jeu est mono-joueur.

Difficultés :

Le découpage a été difficile à mettre en œuvre.

Problèmes généraux :

Étant donné que chaque jeu hérite/implémente des classes de basic, alors les erreurs de typages ont été fréquentes. Ainsi, nous devions utiliser à de nombreuses reprises des CAST.

Nous devions aussi corriger mutuellement nos codes, ajouter nos propres fonctions en plus des modèles préétablis pour les classes, et nettoyer le superflu, simplifier le tout ce qui a pris un temps considérable.

Le modèle de base a été revu à plusieurs reprises (changement entre abstract et interfaces par, changement des interactions entre les différentes classes etc.)

Ensuite, le plus gros problème a été de rendre les données cohérentes entre chaque jeu pour que la vue soit la plus simple à mettre en place possible. C'est-à-dire qu'il fallait trouver un maximum de points commun pour les exploiter, et ce le plus succinctement possible.

De plus, redimensionner les images pour chaque case dans la vue a été impossible.

Piste d'extensions à implémenter :