Adrian Amaglio
Groupe C2
N° 11711172

Le jeu de balle au prisonnier

Le modèle MVC

Dans ce projet, le choix a été fait de placer la vue comme élément actif. Les classes qui la composent se charge de la gestion des événements utilisateur et notifient le contrôleur lorsque c'est nécessaire. Chaque objet ayant une interaction directe avec la vue, les joueurs et la balle pour commencer, ont un proxy chargé d'assurer la cohérence des comportements entre le cœur du jeu et son interface graphique. Cette pratique permet de laisser le soin à l'interface de rythmer le programme, ce qui est bien plus efficace et confortable dans un jeu en temps réel.

Les design patterns utilisés

Le proxy

Les objets appartenant au modèle et au contrôleur étant complètement indépendant de ceux de la vue, les évènements ne remontent pas naturellement à l'interface graphique. Les classes GraphicBall et GraphicPlayer permettent d'intercepter les actions du contrôleur et de leur faire correspondre un évènement graphique.
Les interfaces PlayerInterface et BallInterface s'assurent que les objets de type Graphic- aient bien les méthodes nécessaires au bon fonctionnement du jeu.

Observateur

Afin de faire remonter les actions des utilisateurs au moteur de jeu, les différents éléments du contrôleur suivent le patron observateur/observé. Ils implémentent une méthode « event » qui est appelée à chaque évènement significatif de l'utilisateur. L'interface PlayerInterface s'assure que la méthode de notification existe bien dans chaque objet joueur.
Ces objets s'enregistrent à la création du terrain, dans la classe Field, pour obtenir ces notifications.

Travail sur la généricité du code

Afin de pouvoir mieux réutiliser le code et le faire évoluer plus facilement, de nombreux concepts du jeu ont été affranchis de leur type initial.

Les positions.

Tout élément sur le terrain de jeu a une position. La classe Position permet de l'abstraire. Les objets représentant des entités du jeu ne sont plus chargés de travailler avec leurs coordonnées X et Y sur le terrain. La responsabilité de gérer ces informations est déléguée à la classe position.
Dans de nombreux cas d'évolution du programme, cette architecture est bénéfique. Si les déplacements

verticaux devenaient possibles, si une troisième dimension devait être ajoutée (se baisser pour éviter la balle), seul le code de la classe Position viendrait à changer, l'interface offerte aux entités sur le jeu resterait la même.

Les équipes et les côtés du terrain.

Afin de ne plus manipuler de chaînes de caractères pour définir les équipes et leur place sur le terrain, pour prévenir les erreurs de frappe et faciliter d'éventuelles modifications, les classes Side et Team ont été ajoutées. La classe Team permettant également de gérer la couleur de l'équipe.

Les joueurs

Pour créer une interface robuste et pouvoir ajouter de nouvelles façons de jouer au jeu (en réseau, avec différentes difficultés d'intelligence artificielle…) toutes les classes permettant de contrôler un joueur sur le terrain doivent implémenter les méthodes de l'interface PlayerInterface.