Bridge Pattern Game

Théo Coutaudier, Eric Peronetti, Jérémie Santoro

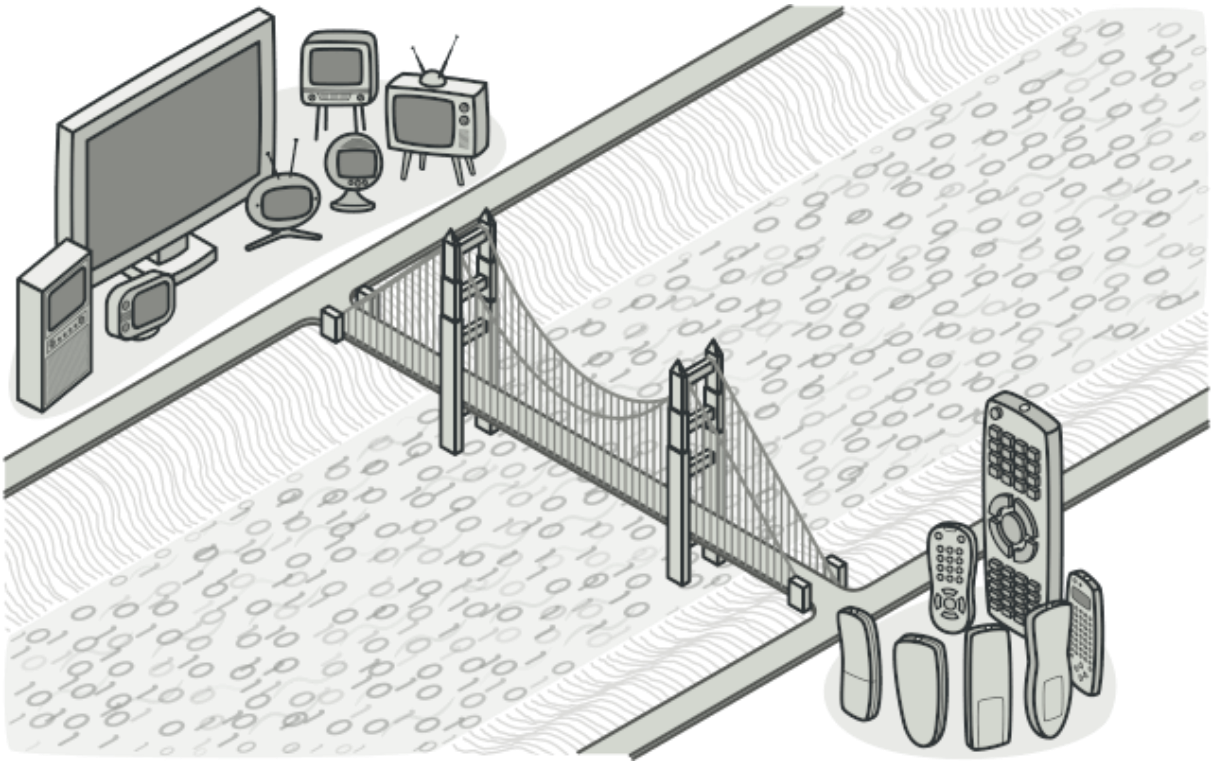
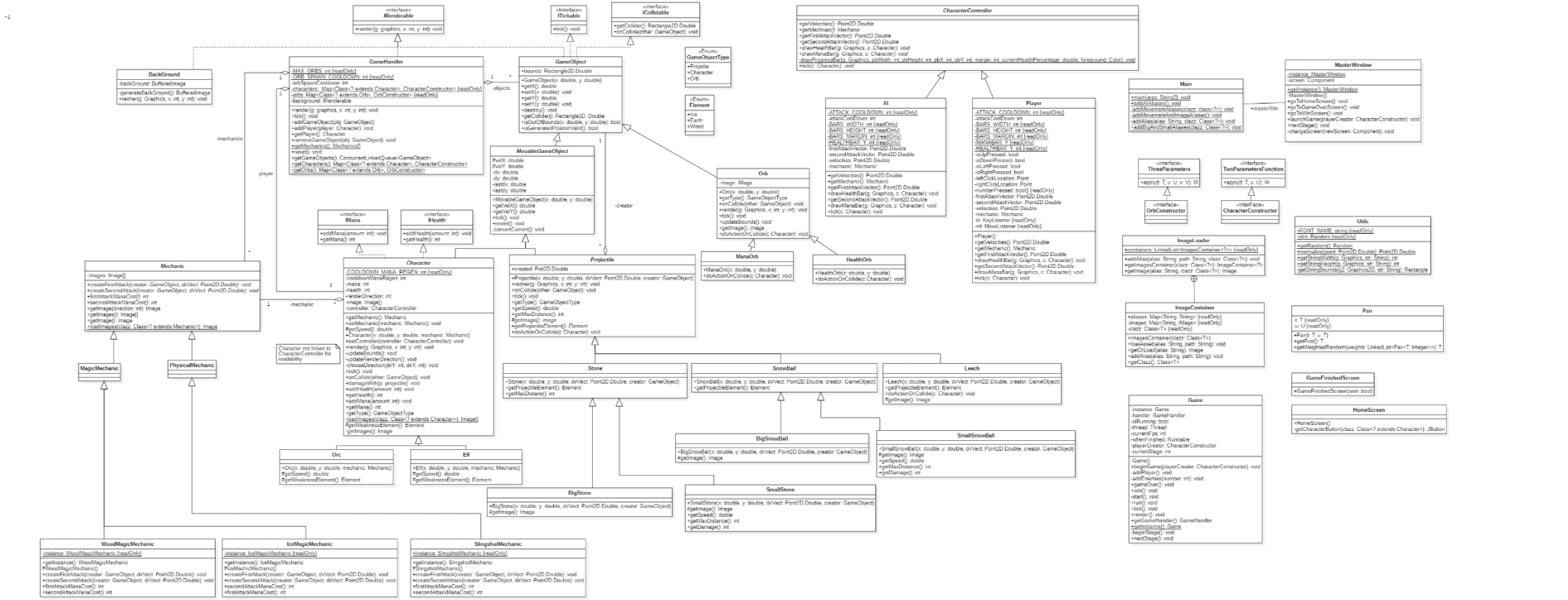


Table des matières

L'UML est disponible sur la page suivante mais est aussi disponible en tant qu'image PNG dans le zip (BridgeGameUML.png)

18.06.2023

Bridge Pattern – MCR



Comment jouer

Au début, il vous faudra choisir entre 2 personnages, soit un Orc soit un Elf. Ils sont faibles contre un élément différent et ont une vitesse différente (l'Elf est plus rapide que l'Orc).

Voici les personnages avec leurs vitesses et leurs faiblesses :

Personnage	**Vitesse**	**Faiblesse**
Orc	1.8	Glace
Elf	2.5	Terre

Après avoir choisi votre personnage, le jeu débutera. Vous allez apparaître à une position aléatoire (vous pouvez reconnaître lequel est votre personnage car ça sera le seul sans barre de vie et de mana au-dessus de sa tête) avec une mécanique (une arme que vous pouvez utiliser ou changer, il y en a 3 différente – Par défaut ça sera la mécanique qui peut être choisie avec la touche 1 (mécanique de magie de glace actuellement)).

Il vous faudra éliminer tous les ennemis autour de vous pour passer au niveau suivant. Chaque mécanique fait plus ou moins de dégâts en fonction de l'ennemi sur lequel on tire. Car chaque mécanique est d'un élément différent (terre, glace ou bois) et les ennemis tout comme vous ont un élément de faiblesse. Choisir la bonne arme pour maximiser vos chances de tuer les ennemis tout en conservant votre mana est crucial pour gagner rapidement et facilement.

Lorsque vous tirez le projectile va partir dans la direction de votre souris (par rapport à en haut à gauche de votre personnage).

Attention les ennemis peuvent également vous tirez dessus, ils auront une mécanique aléatoire attribuée à leur création, n'oubliez pas que vous avez un élément de faiblesse aussi. Certaine mécanique vous fera donc plus mal que d'autre.

Voici l'ensemble des commandes que vous pouvez faire :

- Pour vous déplacer utilisez : WASD
- Pour tirer : Clique gauche de la souris pour la première attaque et clique doit pour la deuxième attaque
- Pour changer d'armes : 1, 2 ou 3
- Pour quitter le jeu : ALT+F4

Voici la liste des attaques :

Mécanique	**Première attaque (clique gauche)**	**Deuxième attaque (clique droit)**
Magie de glace	Lance une grosse boule de neige (type glace – 20 dégâts) Coût : 30 MP	Lance une petite boule de neige (type glace – 16 dégâts) Coût : 20 MP
Fronde	Lance une grosse pierre (type terre – 20 dégâts) Coût : 35 MP	Lance une petite pierre (type terre – 15 dégâts) Coût : 20 MP
Magie de bois	Lance une sangsue (type bois – 20 dégâts - 4 HP régénérés lors du contact avec un ennemi pour le lanceur du projectile) Coût : 20 MP	Régénère 30 HP directement Coût : 90 MP

Vous pouvez ramasser des orbes qui vous permettent de regagner de la vie (Orbe verte) ou du mana (Orbe bleu). Vous pouvez voir votre vie et votre mana en bas à gauche de l'écran.

Voici la liste des orbes :

Orbe	**Effet**
Orbe verte	+20 HP
Orbe bleue	+20 MP

Une fois tous les ennemis éliminer vous passez au niveau suivant, qui contiendra plus d'ennemis encore (2 -> 4 -> 6 -> 8 -> …). Si vous mourrez vous recommencez le jeu depuis le début.

Structure du projet

Pour ce projet nous avons la class abstraite Character qui permet de représenter les différents personnages. Elle possède 2 sous-classes : Orc et Elf. Ces classes permettent de définir toutes leurs caractéristiques comme leurs vies, leurs images graphiques ou encore leurs mécaniques.

La classe abstraite Mechanic représente les armes avec lesquels le joueur et les ennemis tirent. La structure des Mechanic se décompose en une classe abstraite Mechanic, puis 2 autres classes abstraites pour chaque type (Physique ou Magique). Toutes les sous-classes vont ensuite représenter les différentes mécaniques possibles à utiliser. Ces sous-classes sont des singletons, cela évite de créer plusieurs instances de mécaniques alors que l'on pourrait très bien n'en avoir qu'un seul étant donné qu'aucune information n'est stockée dans la mécanique (à part les différentes images qui resterons inchangées pour le même type de mécanique).

Nous avons également l'énumération Element permet de définir les éléments de notre Jeu (Terre, Glace ou Bois). Ils sont notamment utilisés dans les class Character et Mechanic.

Il y a deux types de joueur, les Player et les AI, dans notre projet ce sont deux classes qui héritent de la classe abstraite CharacterController et qui nous permet de s'occuper du dessin des barres de vie, ainsi que de la gestion de AI et également de la gestion du Joueur qui comporte entre autres les entrées clavier et souris du joueur.

La class abstraite Projectile va gérer tous les projectiles des Mechanic. C'est-à-dire son image, son dessin, son déplacement (notamment sa distance maximale et sa vitesse), la gestion des collisions ou encore des dégâts.

Les classes Game, GameFinishedScreen, HomeScreen gèrent les différents écrans dans le jeu. Le premier avec un canevas, les deux derniers avec JPanel. La classe MasterWindow, permet de gérer le changement entre les différents screens.

Nous avons plusieurs interfaces pour faciliter la gestion de différente caractéristique dans les objets du jeu :

- ICollidable : Pour les éléments qui peuvent occasionner des collisions entre eux.
- IHealth : Pour les éléments qui ont de la vie
- IMana : Pour les éléments qui ont du mana
- IRenderable : Pour les éléments qui peuvent être affichés
- ITickable : Pour les éléments qui sont tickable, c'est-à-dire qui s'actualise très régulièrement au fur à mesure du jeu.

La class abstraite Orb possède les sous-classes HealthOrb et ManaOrb. Ce sont les 2 types d'orbes que nous avons dans le jeu. Elles sont générées aléatoirement sur le terrain au fur et à mesure du temps (avec un maximum de 5 sur le terrain en même temps).

Nous avons une énumération GameObjectType qui représente les types d'objet de notre jeu : Character, Projectile, Orb. Tous décrits précédemment.

Pour la gestion des objets dans les espaces et des collisions. Nous avons fait une class abstraite GameObject. MovableGameObject est une sous class (abstraite également) qui traite les objets qui ont une vélocité, elle est utilisée par les personnages (Character) et les projectiles. Les orbes quant à elles, utiliseent GameObject directement car elles n'ont pas de vélocité.

Il y a aussi la class Background pour la génération aléatoire de l'image de fond.

Finalement c'est la classe GameHandler, qui va s'occuper de gérer la plupart des objets mentionné précédemment et les faire interagir entre eux. C'est lui qui ajoute ou enlève des objets, c'est lui qui gère le Player, c'est lui qui déclare les singleton, etc…

Bridge Pattern

Le bridge pattern est utilisé entre la classe abstraite Character et la class abstraite Mechanic. Character possède un attribut de type Mechanic, ensuite ces deux classes possèdent différente sous-classe.

Une sous-classe de Character pourra donc utiliser une sous class de Mechanic grâce au bridge pattern, voici un exemple :

Si nous créons donc un Elf par exemple, nous lui donnerons une Mechanic, disons la WoodMagicMechanic. Il pourra donc utiliser cette armes comme il le souhaite, notamment via la fonction getMechanic() qui permet de savoir quelle mécanique un personnage utilise. Et il pourra également la changer via la fonction setMechanic(). Pendant la partie il pourra donc changer en tout temps de mécanique pour prendre la IceMagicMechanic par exemple et l'utiliser. Un Orc pourrait faire de même.

L'avantage du bridge pattern ici du point de vue du jeu est qu'il est possible de changer d'armes pendant que le programme run. Un personnage n'est pas bloqué sur la même mécanique dès qu'il est créé. Du point de vue du développeur, il y a 3 avantages. Premièrement il n'y a pas de duplication du code. C'est beaucoup plus lisible. Deuxièmement, modifier un personnage ou une mécanique est très simple, car il suffit juste de modifier la classe en question. Dernièrement, ajouter un nouveau personnage ou une nouvelle mécanique ne peut pas être plus simple. Il faut juste rajouter une sous-classe à la class abstraite Mechanic ou Character.

Déploiement

Pour le déploiement il suffit d'avoir une version supérieure ou égale à Java 19, d'aller dans ce dossier : BridgeGame\out\production\BridgeGame\

Et de lancer la commande : >java -jar BridgeGame.jar

Le fichier jar peut être déplacé comme bon vous semble, toutes les images / ressources nécessaires sont stockées directement dans le fichier jar.