Projet Java-2

Le but de ce projet est de fournir une version simplifiée du jeu de plateau *Dice Wars*. Le projet vous permettra de mettre en application les connaissances acquises en cours et de développer des compétences en programmation Java. Le travail est fait en équipe de 3 étudiants (ou moins si pas possible). La composition des équipes doit être faite au début de la première séance et envoyée à votre enseignant par mail.

Chaque équipe doit remettre, à la fin du projet, une copie des fichiers contenant le code du programme commenté (un dossier par version) et un rapport expliquant le travail fait. Aucune soutenance n'est prévue pour ce projet. La note du projet représentera 75% de la note finale.

Règles du jeu

Il est nécessaire de bien lire les règles du jeu, cela vous aidera dans la conception et l'implémentation de votre programme.

Générale

- Une carte contenant des territoires est partagée par plusieurs joueurs
- Chaque territoire appartient à un seul joueur
- Chaque joueur possède un nombre de territoire
- Chaque joueur doit conquérir tous les territoires représentés sur la carte
- La force de chaque territoire est représentée par un nombre de dés
- Il ne peut y avoir plus de 8 dés sur un territoire

Déroulement

- Au début de la partie, tous les joueurs possèdent le même nombre de dés répartis aléatoirement sur tous leurs territoires
- Lors de son tour de jeu, le joueur choisi parmi ses territoires celui qu'il veut étendre puis un territoire voisin pour l'attaquer
- Les territoires à un seul dé ne peuvent pas attaquer
- La bataille se fait en additionnant les points des dés en attaque et les comparant au score des dés en défense
- Si l'attaquant fait plus de points, il déplace ses dés sur le territoire conquis, sauf un qui reste sur le territoire de départ et les dés vaincus de l'adversaire disparaissent
- Si l'attaquant perd, il ne conserve qu'un seul dé sur son territoire et le territoire attaqué reste inchangé
- Pendant son tour, un joueur peut attaquer autant de fois qu'il le souhaite avant de finir son tour
- A la fin du tour, le joueur gagne autant de dés que le plus grand nombre de territoires contigus de sa couleur
- Ces dés de renfort sont répartis au hasard sur les territoires

L’image ci-dessous représente une partie en cours avec 6 joueurs représentés par les différents couleurs.

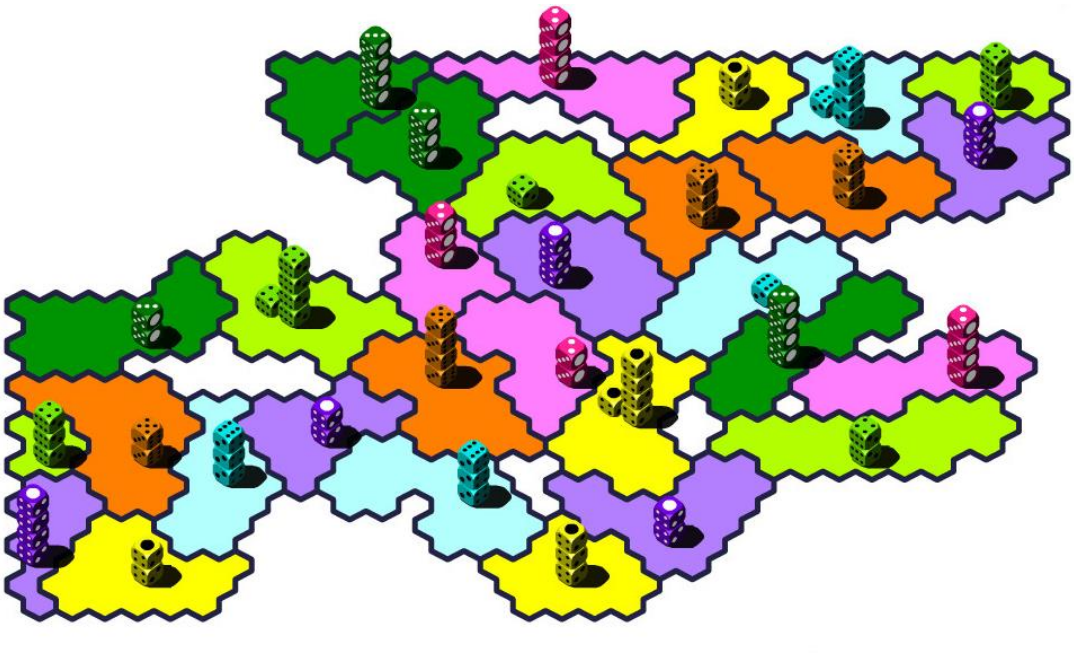

Figure 1. Exemple d'une partie en cours avec 6 joueurs

Une première version

Remarque

Il est nécessaire d’aller « au bout » de cette première partie pour pouvoir passer à la suite. Le schéma proposé ci-dessus vous donne une structure générale de votre programme pour vous guider dans l’avancement, cela ne vous limite pas aux classes est méthodes proposées. Vous pouvez définir d’autres classes et méthodes si besoin.

1. Créez une Classe `Joueur` représentant un joueur d’une partie avec un attribue `int ID` (unique) et une méthode `jouer()` qui permet à un joueur de récupérer la main (entrée clavier). Pour l’instant la seule action possible c’est de terminer son tour en entrant «q»

2. Pour cette version on se contente de modéliser la carte des territoires par un simple tableau à deux dimensions où un territoire est représenté par une case et partage les frontières avec les territoires présents sur les cases adjacentes. Une case peut être vide.

- Créez la classe *Carte* contenant la matrice des territoires
    - A ce niveau, la matrice doit être construite à partir d'un fichier CSV à l'instanciation. La valeur d'une case indique seulement si le territoire est présent ou pas. La carte doit être connectée et assez dense.

- Proposez un moyen pour attribuer un entier unique à chaque territoire par lequel il sera identifié

- Modifiez la classe *Carte* pour maintenir les informations nécessaires sur les territoires :
    - Nombre de dés par territoire
    - Joueur possédant le territoire

- Définissez les méthodes permettant de :
    - Récupérer les voisins d'un territoire donné
    - Récupérer/modifier le nombre de dés d'un territoire
    - Récupérer/modifier l'ID du joueur possédant un territoire

3. Créez une classe *Partie* (main) qui initialise une partie et gère l'ordonnancement du jeu :
    - Création des joueurs
        - Le nombre de joueurs est donné en argument au programme

    - Création d'une carte
    - Ordonnancement des rôles des joueurs durant la partie
    - Affichage de la carte sous forme de matrice après chaque tour avec les valeurs *ID* du joueur, nombre de dés sur chaque case

4. Testez l'exécution du programme

5. Créez une classe *Jeu* qui gère les actions, les calculs et l'avancement de la partie selon les règles :
    - A l'instanciation
        - Une attribution aléatoire des territoires aux joueurs
        - Répartition initiale des dés sur les territoires (aléatoire avec au moins un dé sur chaque territoire)

    - Une action de lancée de dés représentée par une méthode qui calcule la somme des points d'un nombre donné de dés. Cette méthode utilisera une autre méthode privée *int aleatoire()* qui retourne un entier entre 1 et 6
        - Une méthode permettant de mettre à jour les deux territoires après une attaque et vérifie si la partie est terminée

- Une autre méthode qui à la fin du tour :
    - Calcule le nombre de dés rapportés par le joueur
    - Répartie aléatoirement les dés sur les territoires du joueur

6. Définissez la méthode `void attaquer (int territoireAttaquant, int territoireAttaquée)` de la classe `Joueur:`

    - Cette méthode est appelée lorsqu'un joueur réalise une attaque en entrant le numéro du territoire attaquant et du territoire attaqué séparer par un espace

7. Utilisez les *Exceptions* pour gérer les actions interdites lors d'un tour de jeu, comme :
    - Sélectionner un territoire attaquant qui n'appartient pas au joueur
    - Attaquer avec un territoire contenant un seul dé
    - Attaquer un territoire non voisin
    - Attaquer un territoire appartenant au joueur
    - …

8. Modifier La classe `Partie` pour prendre en compte la classe `Jeu` et la fin de la partie

9. Votre programme doit être capable maintenant de lancer une partie complète. Testez son exécution sur différents scénarios

C'est plus fun avec une souris

Dans cette version, vous allez améliorer votre programme pour permettre un affichage graphique. Le programme doit permettre l'interaction de plusieurs jours avec un plateau de jeu où :

- Chaque joueur est identifié par une couleur
- La carte est représentée par une matrice de territoire
- Chaque territoire est représenté par un bouton de la couleur du joueur et affichant le nombre de dés
- Le joueur sélectionne les territoires en cliquent sur les boutons correspondants et finit son tour en cliquant sur le bouton « fin de tour »
- L'affichage doit être mis à jour après chaque attaque et chaque fin de tour

1. Modifiez votre programme en ajoutant les modules nécessaires pour la gestion des fenêtres, des différents composants et des actions. Votre conception doit respecter le *design pattern Modèle-Vue-Contrôleur (MVC).* Votre programme doit aussi gérer les actions interdites.

2. Modifiez votre classe `Carte` pour permettre une génération aléatoire de cartes respectant la contrainte de connexion (pas de déconnexion dans la carte). Votre proposition doit prendre en

compte des paramètres pour contrôler la taille de la carte (le nombre de territoires qui doit être multiple du nombre de joueurs) et sa densité (en terme de connexions entre les territoires).

N'oubliez pas de sauvegarder

Dans cette partie, vous allez programmer la persistance de votre application. Il s'agit de pouvoir arrêter la partie à n'importe quel moment, la sauvegarder sur un support persistant, un fichier en l'occurrence, puis il doit être possible de la reprendre : c'est à dire restaurer tous les objets, avec toutes les valeurs de leurs attributs mais aussi toutes les relations entre les objets.

1. Modifiez votre programme pour permettre cette fonctionnalité de sauvegarde
2. Testez cette version sur différents scénarios

Affronter l'IA

Améliorez votre programme pour avoir un joueur contrôlé par votre programme avec différents niveaux de difficulté.

Une vraie carte

Utilisez une structure en *Graphe* pour une implémentation plus souple de la carte avec une version graphique améliorée.

Jouer en ligne (bonus)

Développer une version permettant à deux joueurs de jouer à distance (un joueur par machine) en utilisant un client et un serveur TCP. Pour simplifier, un joueur sera associé au serveur et l'autre au client (asymétrique).