Rapport du projet K-tan

Sommaire:

I- Bilan Hebdomadaire

Première semaine: 30/01/24-06/02/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	Implémentation de la classe Player	Implémentation des cartes ressources
Claire Chambaz	Ajout de classes	Début de logique des tours
Marc Robin	Ajout de classes	Création du menu
Gabriel Treca	Map en cours	Continuer la modélisation de la map
Zhula XXX	Ajout de classes	Implémentation des cartes ressources

Semaine: 06/02/24-13/02/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	Implémentation des cartes ressources	Débuter la logique d'échanges de ressources
Claire Chambaz	Implémentation de la logique des tours	Graphique primitif de la Map
Marc Robin	Création Menu primitif	Reprendre l'implémentation des tours
Gabriel Treca	Modélisation de la Map en cours	Finir la Map
Zhula XXX	Implémentation des cartes ressources	Interface pour les cartes de jeu

Semaine: 13/02/24-20/02/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	Echanges de cartes et de ressources	Ajout de l'affichage des villes et routes
Claire Chambaz	Interface graphique de la carte en cours	Continuer la carte graphique
Marc Robin	Implémentation du système de tours	Finir le système de tours
Gabriel Treca	Prototype fonctionnel de la Map	Ajout de méthodes pratiques pour utiliser interagir avec la map
Zhula XXX	Echanges de cartes et de ressources	Interface graphique des cartes de jeu

Semaine: 20/02/24-27/02/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	Ajout de l'affichage des villes et routes	Implémenter graphiquement la pose de construction de villes et routes
Claire Chambaz	Carte graphique en cours et Création d'un menu esthétique	Finir la Map graphique
Marc Robin	Implémentation du système de tours en cours	Finir de coder le système de tours
Gabriel Treca	Ajout de méthodes pratiques pour utiliser interagir avec la map	Débuter l'IA
Zhula XXX	Interface graphique des cartes de jeu en cours	Finir l'affichage des cartes ressources

Semaine: 27/02/24-05/03/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	Mettre en place le système d'interaction graphique avec le plateau (poser constructions)	Ajout d'un menu options pour choisir la difficulté, le nom et la couleur des joueurs
Claire Chambaz	Fin de l'affichage de la Map graphique	Concevoir l'extension météo
Marc Robin	Implémentation du système monétaire	Créer des classes pour l'extension météo (Matrices pour chaînes de Markov)
Gabriel Treca	Débuter l'IA	Début de prototypes
Zhula XXX	Fin de l'affichage des cartes ressources	

Semaine: 05/03/24-12/03/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	Ajout d'un menu options pour choisir la difficulté, le nom et la couleur des joueurs	Amélioration de l'affichage graphique du jeux
Claire Chambaz	Conception de l'extension et début d'implémentation	Gérer l'affichage de la GameView (fusion de tous les éléments du jeu) et création de dé graphiques
Marc Robin	Création de classes matrices et conception de l'extension	
Gabriel Treca	Début de prototype d'IA	Continuer l'IA

Zhula XXX	Affichage des cartes	

Semaine: 12/03/24-19/03/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	Amélioration de l'affichage graphique du jeu et du système de tour	Création d'un mode joueur contre joueur
Claire Chambaz	Affichage complet de la gameView et création de dés graphiques	Reprendre l'extension météo
Marc Robin	Enrichissement des classes matrix et création des constantes (matrix de météo)	
Gabriel Treca	IA en cours	Continuer IA
Zhula XXX	Design cartes en cours	finir design et logique des cartes

Semaine: 19/03/24-26/03/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	Refonte de l'interface graphique + mode joueur VS joueur en cours	Continuer le mode joueur VS joueur
Claire Chambaz	Météo quotidienne ajoutée	Continuer l'extension météo
Marc Robin	Merge des conflits	
Gabriel Treca	IA en cours	Continuer IA
Zhula XXX	Design des cartes finit	Faire un système de sauvegardes

Semaine: 26/03/24-02/04/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	Mode joueur VS joueur en cours + amélioration de la GameView	Terminer le mode joueur VS joueur
Claire Chambaz	Extension météo en cours	Continuer l'extension météo
Marc Robin	Résolution des conflits	
Gabriel Treca	IA en cours	Continuer IA
Zhula XXX	Sauvegarde en cours	Continuer sauvegarde et s'occuper des points de victoire

Semaine: 02/04/24-23/04/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	Fin du mode joueur VS joueur	implémentation logique et graphique du voleur
Claire Chambaz	Graphique de la météo fini	Terminer l'extension météo (appliquer les effets du temps)
Marc Robin		Réorganisation du code météo
Gabriel Treca	Ajout du port dans la map	Ajout graphique des ports + Continuer IA
Zhula XXX	Graphique des points de victoire en cours	Effet des cartes developments

Semaine: 23/04/24-30/04/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	implémentation du voleur	débogage
Claire Chambaz	Extension météo fini	ajouter du son pour l'ambiance
Marc Robin		
Gabriel Treca	mise en place des ports	Connexion de ports en cours
Zhula XXX	Graphique GameOver	

Dernière semaine: 30/04/24	**Ce qui a été fait cette semaine**	**Objectifs pour la semaine prochaine**
Radouane Aouini	débogages + ajout de méthodes pour l'IA	Finaliser le projet: - terminer le rapport - écrire le readme - faire un exécutable - ajout de tous les derniers éléments en cours
Claire Chambaz	Ajout du son (de météo) + fin de jeu	
Marc Robin	exécutable	
Gabriel Treca	Connexion de ports en cours	
Zhula XXX	cartes de développement	

II- Règles du jeu

Les règles du jeu de société Catan

Catan est un jeu de stratégie et de négociation dans lequel les joueurs tentent de coloniser une île en développant leurs colonies et en échangeant des ressources. Voici les règles de base du jeu :

1. Objectif : Le but de Catan est d'être le premier joueur à atteindre un certain nombre de points de victoire. Habituellement, cela se fait en construisant des colonies, des routes et en réalisant des objectifs spécifiques.

2. Mise en place : La carte de l'île de Catan est composée de tuiles hexagonales représentant différents types de terrain, tels que les forêts, les champs, les montagnes, et les prairies. Chaque tuile produit des ressources spécifiques (bois, blé, pierre, brique, et mouton). Les joueurs placent leurs colonies et leurs routes sur les intersections et les arêtes de ces tuiles.

3. Tour de jeu : Le jeu se déroule en tours. À chaque tour, les joueurs lancent deux dés pour déterminer quelles tuiles produisent des ressources ce tour-ci. Tous les joueurs qui possèdent une colonie adjacente à une tuile qui correspond au résultat du lancer de dés reçoivent une ressource de cette tuile.

4. Construction : Pendant leur tour, les joueurs peuvent construire des colonies, des routes, des villes et acheter des cartes de développement en dépensant les ressources qu'ils ont collectées. Les colonies et les villes rapportent des points de victoire, tandis que les routes permettent d'étendre le réseau des joueurs.

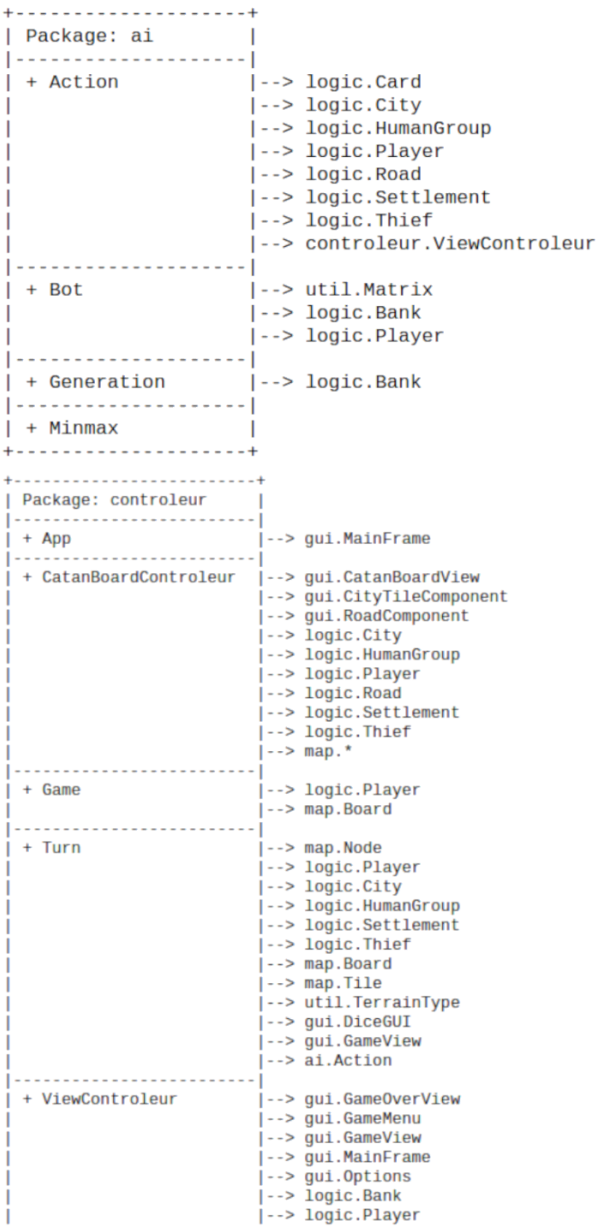
5. Échanges : Les joueurs peuvent également négocier entre eux en échangeant des ressources, des cartes de développement ou même des positions sur la carte. Les négociations font partie intégrante de la stratégie du jeu.

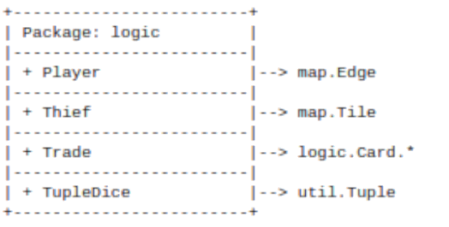
6. Cartes de développement : Les cartes de développement offrent des avantages variés aux joueurs, tels que la construction gratuite de routes, le vol de ressources à un adversaire, ou même des points de victoire supplémentaires.

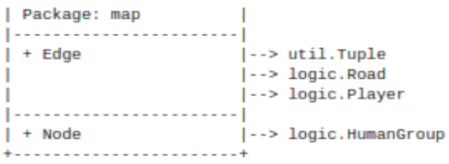
7. Le voleur : Un élément central de Catan est le voleur, représenté par un pion qui peut être déplacé sur une tuile par un joueur qui lance un 7. Lorsque le voleur est placé sur une tuile, cette tuile cesse de produire des ressources, et le joueur qui a placé le voleur peut voler une ressource à un joueur voisin.

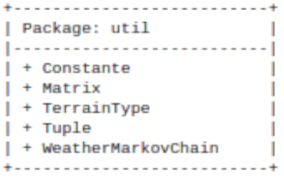
8. Fin de partie : Le jeu se termine dès qu'un joueur atteint le nombre requis de points de victoire. Ce nombre peut varier en fonction du nombre de joueurs et des règles spécifiques à chaque partie.

III- Architecture De Notre Projet









IV- Evolution De Notre Projet

A - Création du plateau de jeu

1. Logic

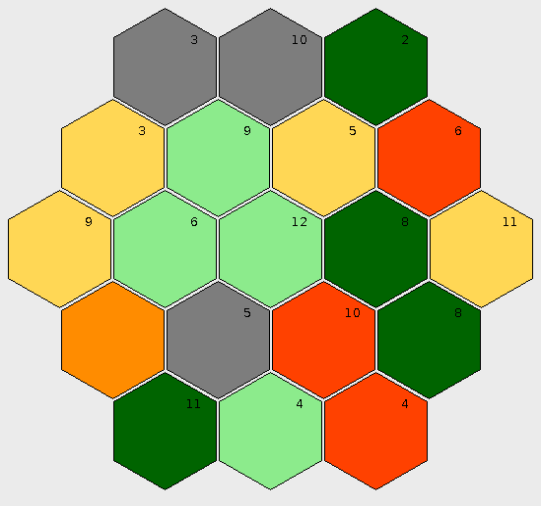
Pour la partie logique j'ai opté pour un mélange entre un graph et un système de coordonnées que nous avons travaillé au fur et à mesure pour répondre au besoin de l'équipe, je me suis aussi appuyé sur un système d'urne pour sélectionner au hasard des éléments et les répartir sur la carte.

2. Graphic

La création graphique de la carte est passée par plusieurs étapes avant d'aboutir à quelque chose. Tout d'abord, c'était un travail de collaboration car il a fallu relier logique et graphique et donc comprendre ce que mon collègue avait fait. Il a donc fallu que l'on s'entretienne quelques fois. D'abord, j'ai essayé de dessiner les hexagones en calculant les coordonnées de chaque tuile grâce au code de mon collègue. Ça ne marchait pas, les hexagones en eux même avaient la bonne forme mais n'étaient pas alignés comme il fallait. Après quelques tentatives pour essayer de les mettre en place j'ai finalement d'opter pour une stratégie beaucoup plus simple: au lieu de calculer les coordonnées de chaque tuile, j'ai simplement déterminer à l'avance les coordonnées de chaque tuiles puis elles sont distribuées aléatoirement lors de la création du plateau de jeu.

```
private static final Vector[] TILE_COORDINATES = {
        new Vector(x:1.5, y:0), new Vector(x:2.5, y:0), new Vector(x:3.5, y:0),
        new Vector(x:1, y:1), new Vector(x:2, y:1), new Vector(x:3, y:1), new Vector(x:4, y:1),
        new Vector(x:0.5, y:2), new Vector(x:1.5, y:2), new Vector(x:2.5, y:2), new Vector(x:3.5, y:2), new Vector(x:4.5, y:2),
        new Vector(x:1, y:3), new Vector(x:2, y:3), new Vector(x:3, y:3), new Vector(x:4, y:3),
        new Vector(x:1.5, y:4), new Vector(x:2.5, y:4), new Vector(x:3.5, y:4)
};
```

Cet attribut m'a ensuite permis de très facilement terminer le graphique de la carte. Il ne restait plus qu'à assigner leur couleur à chaque tuile selon leur type de terrain et afficher leur numéro. Ainsi on obtient le résultat suivant. C'était la première carte graphique utilisable (elle change par la suite, des images remplacent les couleurs).



B - Mise en place du système de tours

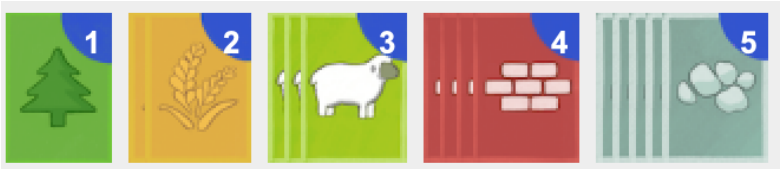
L’implémentation de la logique des tours a été une étape importante dans l’avancée du jeu. Il a été important de scinder en plusieurs classes plus courtes qui avaient des tâches plus spécifiques. Quelques problèmes ont été rencontrés notamment au moment de la récupération des ressources au moment des lancers de dés car tout n’avait pas fini d'être connecté à ce moment.

C - Création des cartes de ressources et développement et Mise en place de l’interface d’échange

Système de cartes
Cartes : Il existe deux types de cartes dans le jeu Catan, les cartes ressources et les cartes développement. Ces cartes ont une classe d'énumération qui définit un ensemble de constantes de carte. La classe d'énumération offre l'avantage de parcourir toutes les cartes dans leur ensemble, et distingue également les cartes de ressources des cartes de développement en fonction de la valeur entière des constantes.
Gestion des cartes : la gestion des cartes est réalisée via une classe de boîtes à cartes, comprenant l'attribution de cartes en tant que banque, la génération de cartes de développement aléatoires, le contrôle de la quantité, ainsi que l'échange et l'application de cartes en tant que joueur.

Échange de carte
Logique : En permettant aux joueurs de mettre en place l'interface commerciale, d'échanger des cartes avec les banques et d'autres joueurs.

Graphique : L'affichage des cartes utilise principalement cardsuit() pour superposer des cartes du même type. A travers trois groupes de cartes --- les cartes que vous possédez, les cartes que vous souhaitez obtenir et les cartes proposées, l'écouteur de la souris sert de connexion pour réaliser l'échange de cartes.



D - Sauvegarde du jeu

En rendant les classes où se trouvent les informations sur les joueurs et les informations sur l'état du jeu sérialisables, toutes les informations sur le jeu peuvent être stockées dans la base de données. Dans ce projet, les informations du jeu sont finalement implémentées en enregistrant la classe Game. Lorsque le jeu est ouvert, il sera vérifié s'il existe un fichier qui stocke les dernières informations du jeu. S'il existe, le jeu continuera. continuer à exécuter le jeu.

```java
public class GameSaveManager {

    2 usages
    public static final String SAVE_FOLDER_PATH = "game_save";
    2 usages
    public static final String SAVE_FILE_PATH = SAVE_FOLDER_PATH + "/game_save.txt";

    no usages  Zhula
    public static void saveGame(Game gameData) {
        try {
            File saveFolder = new File(SAVE_FOLDER_PATH);
            if (!saveFolder.exists()) {
                saveFolder.mkdirs();
            }

            ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(SAVE_FILE_PATH));
            outputStream.writeObject(gameData);
            outputStream.close();

            System.out.println("Game saved successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    1 usage  Zhula
    public static Game loadGame() {
        try {
            ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(SAVE_FILE_PATH));
            Game gameData = (Game) inputStream.readObject();
            inputStream.close();
```

E- Implémentation du mode joueur VS joueur

L'implémentation du mode joueur contre joueur (JcJ) a constitué une étape cruciale dans le développement de notre projet. Ce mode a permis de mettre en avant les interactions directes entre les joueurs, ajoutant une dimension stratégique et compétitive au jeu. Le mode JcJ a été ajouté pour garantir une jouabilité même en l'absence de bots fonctionnels.

Nous avons commencé par la mise en place de l'interface pour les échanges entre les joueurs. Cette interface permet aux joueurs de proposer des échanges de ressources, intégrant ainsi une partie essentielle des interactions JcJ. Les premières versions de l'affichage des échanges entre les joueurs, incluant les interactions avec la banque, ont été mises en place, et le code a été simplifié lors de l'intégration pour assurer une meilleure fluidité des interactions, améliorant ainsi la clarté et l'efficacité des échanges. L'affichage des échanges entre les joueurs a été complété, permettant une interaction visuelle claire et intuitive, incluant les propositions d'échange, et l'acceptation ou le refus des échanges.

Un aspect crucial de l'implémentation a été de réfléchir à l'ordre des actions et au fonctionnement du tour par tour. Cette réflexion a été nécessaire pour garantir une expérience de jeu fluide et stratégique. Nous nous sommes également posés des questions d'interface utilisateur (UI) pour déterminer comment les joueurs allaient interagir avec l'application de manière intuitive et efficace.

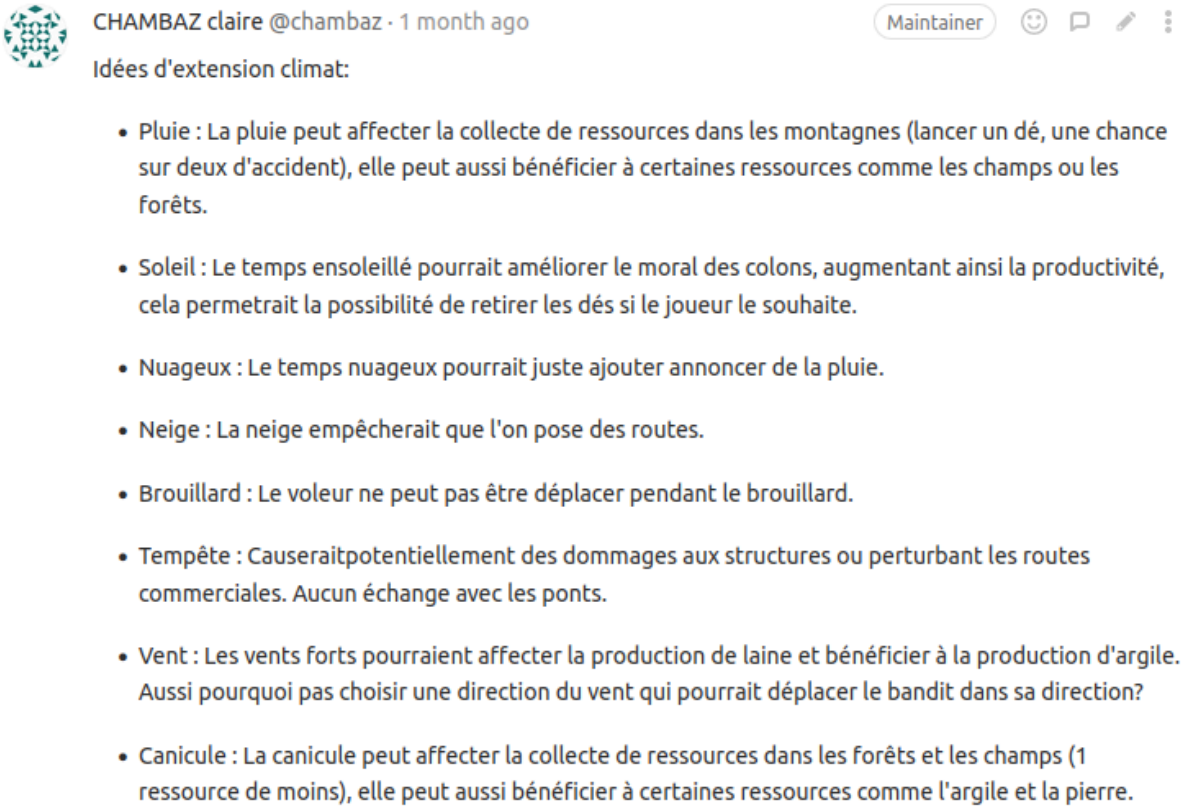
Principaux défis et solutions :

- Compréhension et intégration du code existant :
Le principal défi a été de comprendre le code écrit par les autres membres de l'équipe pour assurer une intégration harmonieuse des nouvelles fonctionnalités. Des réunions régulières ont été organisées pour faciliter cette compréhension et résoudre les problèmes rapidement.

F- Mise en place de l'extension

1. Conception

Il a fallu que l’on réfléchisse à une extension pour notre jeu. On s'est très vite mis d’accord sur le fait qu’une extension météo était réalisable et ajoutait une nouvelle dimension au jeu. On s’est posé de nombreuses questions: Est ce que la météo allait être purement aléatoire et ajouterait uniquement des effets temporaires ? Est ce que les joueurs pourraient contrôler la météo ? Est ce que le joueur pourrait la prendre en compte dans sa stratégie grâce à des prédictions météorologiques ? On a finalement opté pour la chose suivante: créer un système de météo en utilisant des chaînes de Markov pour pouvoir donner aux joueurs les prévisions du lendemain. On a ensuite réfléchi aux effets de chaque météo.



CHAMBAZ claire @chambaz · 1 month ago Maintainer

Idées d'extension climat:

- Pluie : La pluie peut affecter la collecte de ressources dans les montagnes (lancer un dé, une chance sur deux d'accident), elle peut aussi bénéficier à certaines ressources comme les champs ou les forêts.
- Soleil : Le temps ensoleillé pourrait améliorer le moral des colons, augmentant ainsi la productivité, cela permetrait la possibilité de retirer les dés si le joueur le souhaite.
- Nuageux : Le temps nuageux pourrait juste ajouter annoncer de la pluie.
- Neige : La neige empêcherait que l'on pose des routes.
- Brouillard : Le voleur ne peut pas être déplacer pendant le brouillard.
- Tempête : Causeraitpotentiellement des dommages aux structures ou perturbant les routes commerciales. Aucun échange avec les ponts.
- Vent : Les vents forts pourraient affecter la production de laine et bénéficier à la production d'argile. Aussi pourquoi pas choisir une direction du vent qui pourrait déplacer le bandit dans sa direction?
- Canicule : La canicule peut affecter la collecte de ressources dans les forêts et les champs (1 ressource de moins), elle peut aussi bénéficier à certaines ressources comme l'argile et la pierre.

On a finalement opté pour les suivantes uniquement:

- Soleil: donne la possibilité au joueur de relancer les dés s'il le souhaite.
- Nuage: empêche le voleur d'être déplacé.
- Pluie: permet de gagner une ressource supplémentaire de bois ou de blé (si les dés activent une tuile forêt ou champ)
- Vent: permet de gagner une ressource supplémentaire de pierre ou d'argile (si les dés activent une tuile montagne ou collines)
- Neige: empêche le joueur de construire une route.

2. Logique

La particularité principale du jeu est donc l'ajout d'effets météorologiques. Rapidement, la question de la modélisation s'est posée. Nous nous sommes donc dit qu'il serait intéressant de procéder de la manière suivante : considérer un état initial avec un objet de transition qui nous permet de passer d'un état à un autre.

Mais alors, quel objet utiliser pour modéliser ces transitions météorologiques ?

Un outil mathématique adapté serait les chaînes de Markov, qui de surcroît sont assez simples à implémenter et assez réalistes. Il ne restait plus qu'à mettre en place cette logique. À ce titre, j'ai implémenté une classe Matrix représentant des matrices.

Dans un premier temps, j'ai fait cette classe générique. Cependant, j'ai rencontré un problème de taille, à savoir la multiplication de matrices. En effet, si on prend un objet générique, il n'a pas forcément de méthode de multiplication. Le problème ne réside pas là, car il suffit d'imposer que l'objet ait une méthode de multiplication. Sauf qu'aucune méthode (telles que `times` ou `add`) n'est commune à toutes les classes qui sont munies de la multiplication. Je me suis donc rabattu sur une classe Matrix qui prenait des doubles en entrée.

Il a donc fallu ensuite, une fois la classe Matrix créée, initialiser les matrices de transition permettant d'appliquer le principe des chaînes de Markov. Après un temps de réflexion et de concertation avec une camarade pour obtenir des transitions les plus réalistes possible, nous avons ajouté ces matrices constantes. Il est à noter que chaque saison est associée à une matrice de transition.

Ce choix d'utiliser les chaînes de Markov est peut-être un peu lourd pour l'utilisation envisagée, mais il se justifie par le fait que nous voulions ajouter des prévisions météorologiques au jeu. C'est-à-dire qu'à chaque tour, nous aurions annoncé la météo prévue sur n jours. Pour ce faire, il est très pratique d'utiliser les chaînes de Markov. Il suffit de considérer une matrice ligne A de l'état actuel. Par exemple, s'il fait soleil, on considère un simplexe de dimension (le nombre d'états différents) et on indique qu'il fait soleil. On utilise ensuite la matrice M de transition associée à la saison actuelle, et on calcule M^n, puis A*M^n pour obtenir les prévisions au jour j+n.

Cependant, ce dispositif n'a pas été mis en place par manque de temps et incompréhension.

Voici nos quatres matrices de transition (une pour chaque saison).

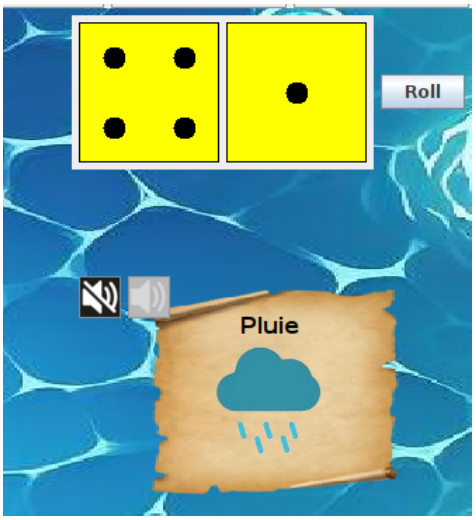
```
private final static double[][] matriceHivernale = { {0.2, 0.1, 0.4, 0.2, 0.1},
                                                     {0.1, 0.3, 0.3, 0.2, 0.1},
                                                     {0.2, 0.1, 0.4, 0.2, 0.1},
                                                     {0.1, 0.2, 0.2, 0.4, 0.1},
                                                     {0.2, 0.1, 0.3, 0.1, 0.3} };

private final static double[][] matricePrintemps = { {0.3, 0.2, 0.3, 0.0, 0.2},
                                                     {0.2, 0.3, 0.3, 0.0, 0.2},
                                                     {0.2, 0.2, 0.4, 0.0, 0.2},
                                                     {0.1, 0.2, 0.1, 0.1, 0.5},
                                                     {0.2, 0.2, 0.3, 0.0, 0.3} };

private final static double[][] matriceEstivale = { {0.4, 0.2, 0.3, 0.0, 0.1},
                                                    {0.2, 0.4, 0.2, 0.0, 0.2},
                                                    {0.1, 0.2, 0.5, 0.0, 0.2},
                                                    {0.0, 0.1, 0.0, 0.0, 0.9},
                                                    {0.2, 0.2, 0.2, 0.0, 0.4} };

private final static double[][] matriceAutomnale = { {0.3, 0.2, 0.3, 0.1, 0.1},
                                                     {0.2, 0.3, 0.2, 0.2, 0.1},
                                                     {0.1, 0.2, 0.4, 0.1, 0.2},
                                                     {0.1, 0.2, 0.2, 0.3, 0.2},
                                                     {0.2, 0.2, 0.2, 0.1, 0.3} };
```

3. Graphique et Audio



La classe **WeatherDisplay** est une composante graphique qui affiche les informations météorologiques et gère la partie audio de l'extension météo dans le jeu. Voici une description des principales fonctionnalités ajoutées pour l'aspect graphique et audio.

1. Composants Graphiques :

- **weatherLabel** : Un label qui affiche le type de conditions météorologiques actuelles.
- **weatherIconLabel** : Un label qui affiche une icône représentant la météo actuelle.
- **muteButton** et **unmuteButton** : Deux boutons pour couper ou rétablir le son.

2. Mise à Jour des Informations Météorologiques :

- La méthode **updateWeather()** est appelée pour mettre à jour les informations météorologiques. Elle utilise les chaînes de Markov pour déterminer la météo suivante en fonction de la météo actuelle et de la saison.
- La méthode charge ensuite l'icône correspondante et met à jour l'affichage.

3. Gestion Audio :

- La méthode **playMusic(String weather)** est appelée pour jouer la musique d'ambiance correspondant aux conditions météorologiques. Elle charge le fichier audio approprié en fonction de la météo actuelle.
- Le son est arrêté lorsque la météo change ou lorsque l'utilisateur appuie sur le bouton de coupure.
- Les boutons **muteButton** et **unmuteButton** permettent à l'utilisateur de couper ou de rétablir le son.
- Lorsque le son est coupé, la méthode **stopCurrentMusic()** est appelée pour arrêter la musique en cours de lecture.

V- Conclusion

En fin de compte, notre projet de développement du jeu de plateau Catan en Java a été une expérience enrichissante et formatrice. La collaboration en équipe de cinq personnes nous a permis de surmonter de nombreux défis, tant techniques qu'organisationnels. L'utilisation de GitLab pour la gestion de notre code et la coordination des tâches a été un atout précieux, nous permettant de structurer notre travail de manière efficace. Bien que nous n'ayons pas pu intégrer toutes les fonctionnalités souhaitées dans le temps imparti, nous avons réussi à créer une version complète et fonctionnelle du jeu, en y ajoutant même une nouvelle extension.