Rapport AP4A

FRAISSE Tom
MALEWICZ Nicolas

Semestre d'Automne 2024

Table des matières

1 Introduction

1.1 But de ce projet

Ce projet a été réalisé dans le cadre de l'UE AP4B afin de mettre en application les notions de conceptions UML et l'utilisation du langage Java.
Nous avons donc choisi le projet "Troy Dice", un jeu de plateau où les joueurs doivent obtenir le plus de points afin de gagner. Ce projet est donc une version numérique de ce jeu de société en s'inspirant de l'UTBM.
Cela nous a permis de mettre en place les méthodes de conception orientée objet, principalement avec la conception de diagrammes UML avant l'implémentation en Java.

1.2 Fonctionnement du jeu

Initialement, le jeu Troy Dice est un jeu de société stratégique inspiré de la mythologie grecque, où les joueurs se battent pour contrôler et défendre la ville légendaire de Troie. Ce jeu combine des mécaniques de lancers de dés et de gestion de ressources, tout en mettant en avant le choix crucial des constructions.

A première vue ce jeu n'est pas si simple à comprendre, car il possède de nombreuses mécaniques, notamment avec les effets de chaque bâtiment et de chaque section.
Ainsi ici le but est d'obtenir le maximum de points possibles durant les 8 jours de jeu, en construisant les différents bâtiments de la ville à l'aide de 3 monnaies d'échanges. Chacun de ces bâtiments permettra d'obtenir des points et/ou un effet spécial influençant le déroulement du jeu.

Chaque jour de jeu est divisé en deux tours, où les dés seront lancés aléatoirement par le crieur (un des joueurs). Puis ensuite placés sur les différentes places du plateau selon l'ordre croissant des valeurs des dés.
Un des quatre dés sera de couleur noire et ne pourra pas être choisi lors du tour de jeu.
De plus, à chaque tour, l'ordre des places bougera d'un tour dans le sens des aiguilles d'une montre.

Pendant chaque tour de jeu, les joueurs auront donc le choix entre plusieurs possibilités :

— choisir un des dés afin d'obtenir le nombre de ressources indiqué par la valeur du dé et la couleur de la place.
— Construire un batiment qui a couleur de la place et la section de la valeur du dé

De plus, les joueurs peuvent modifier la couleur et la valeur du dé en échange de monnaies.

2 Choix de conception

Cette deuxième partie explique le processus de conception, et les différents choix que nous avons faits durant la conception de ce projet.

Note : Les différents diagrammes seront fournis dans le fichier `Diagramme_AP4B_UTBM_DICE.asta`

2.1 Cas d'utilisations

Après avoir analysé les différentes règles de ce jeu, nous avons dressé l'inventaire de l'ensemble des cas d'utilisation de ce projet afin d'obtenir une vue d'ensemble des différents composants.
Ce premier diagramme nous a permis de modéliser l'ensemble des actions accessibles à l'utilisateur ainsi que leur influence sur l'environnement du jeu.

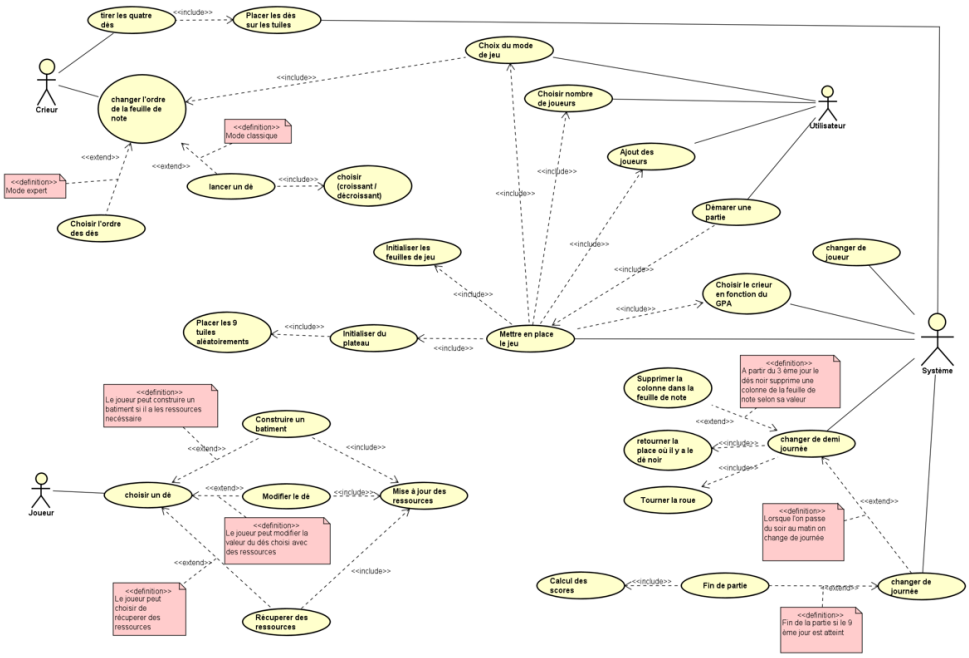


FIGURE 2.1 – Diagramme de cas d'utilisation

2.2 Modélisation du Modèle

Après avoir effectué notre diagramme de cas d'utilisation, nous avons commencé à modéliser le premier diagramme de classe afin de représenter la structure statique du système.
Ce diagramme met en évidence les principales classes identifiées, leurs attributs, leurs méthodes, ainsi que les relations entre elles, telles que l'héritage, l'association ou la composition.
Cette modélisation a permis de clarifier les responsabilités de chaque classe et leurs interactions. Elle a également aidé à identifier les dépendances critiques et à mieux structurer notre système pour répondre aux exigences fonctionnelles identifiées dans les cas d'utilisation.

Premièrement, nous avons identifié les différentes classes mises en jeu dans ce projet, afin d'avoir une vue d'ensemble et de définir la structuration de notre projet. Cela nous a permis de poser les bases de l'organisation et de comprendre les entités clés ainsi que leurs responsabilités.
Puis, dans un second temps, nous avons réfléchi aux différents liens entre chacune des classes en termes d'association, d'héritage et de composition. Ces relations ont été analysées pour garantir une modélisation cohérente et logique, reflétant les interactions attendues entre les éléments du système.
De plus, en parallèle, nous avons approfondi la réflexion sur le rôle et le fonctionnement propres à chacune des classes, afin de définir clairement leurs responsabilités. Cela a conduit à l'identification précise des attributs nécessaires pour stocker l'état de chaque classe, ainsi que des méthodes pertinentes pour assurer leur comportement attendu et leur contribution aux fonctionnalités globales du système.

2.2.1 Packages

Les classes sont organisées en quatre packages distincts. Tout d'abord, on retrouve le package "gui", qui regroupe la gestion de la vue et du contrôleur. Nous avons donc séparé les différentes fenêtres de notre application comme on peut le voir sur la figure suivante. De plus, chaque fenêtre possèdera son propre contrôleur afin de dissocier chaque étape de notre programme.
Ensuite, trois autres packages décrivent le fonctionnement du modèle. Le premier est le package "core", responsable du cœur du fonctionnement de l'application, qui gère la logique principale du jeu et les joueurs. Le deuxième est le package "feuille", qui gère les feuilles de jeu et l'ensemble des fonctionnalités qui leur sont associées, mais aussi la progression de chaque joueur. Enfin, le troisième est le package "plateau", qui manipule les dés et leur interaction avec le plateau.

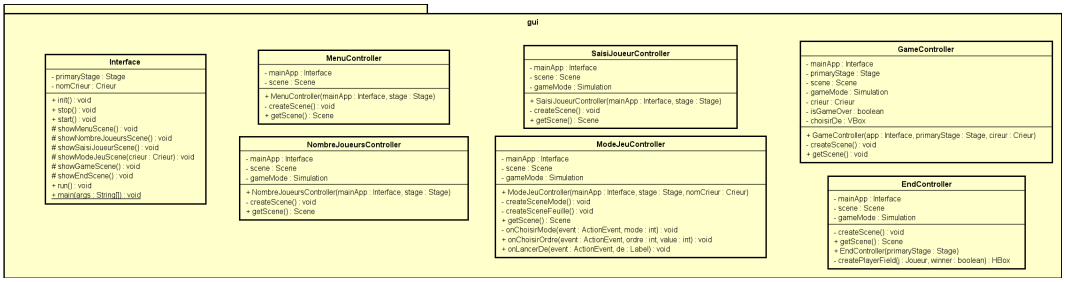


FIGURE 2.2 – Paquetage de la Vue et du Controller

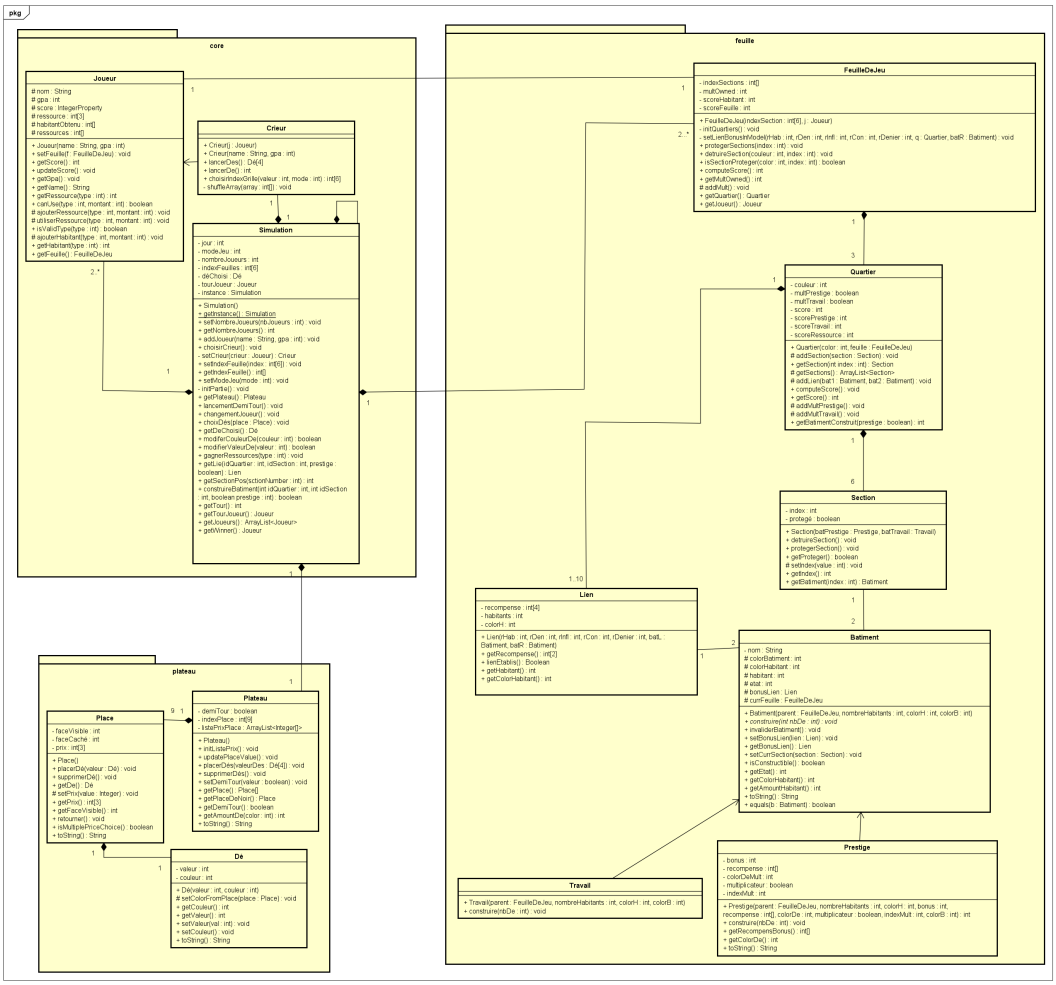


FIGURE 2.3 – Diagramme de classe et paquetage du Modèle

2.2.2 Classes

Nous allons maintenant présenter brièvement chacune de nos classes, leur rôle dans le projet, ainsi que leur objectif principal.

Premièrement nous avons associé trois classes au paquet "core" (Simulation,Joueur,Crieur).
La classe Simulation est centrale dans ce projet et peut être représentée comme le GameMode. Elle a pour but de coordonner le déroulement global de la partie, elle inclut des fonctionnalités telles que l'ajout de joueurs, la gestion des tours, la sélection du crieur et le lancement du jeu.
La classe Joueur contient elle les informations essentielles des joueurs, telles que leur nom, leur score, leurs ressources et leur feuille de jeu associée. Elle offre des méthodes permettant de manipuler les points et de gérer les ressources des joueurs.
La classe Crieur est dédiée au personnage qui choisit le mode de jeu de la partie et lance les dés pour l'initialisation de la partie. Sa méthode de lancé de dés sera réutilisée par le système à tous les tours.

Deuxièmement, le paquet plateau contient l'ensemble de la gestion du plateau de jeu.
La classe Place représente les différents emplacements du plateau où les dés peuvent être placés en fonction de l'état du jeu. Elle gère les actions liées aux dés qui y sont positionnés.
La classe Dé représente les dés utilisés lors du jeu. Elle inclut des propriétés telles que la valeur et la couleur du dé, ainsi que des méthodes pour lancer les dés ou en modifier les propriétés.
La classe Plateau se concentre sur la gestion des élements du plateau, notamment en positionnant les dés sur les bonnes places mais aussi en mettant à jour l'état du plateau.

Enfin le paquet feuille modélise les feuilles de chacun des joueurs au cours de la partie en tenant compte de leur progression.
La classe FeuilleDeJeu est une feuille complète divisée en quartiers, qui contient les sections de jeu et gère le calcul des scores des joueurs. Chaque feuille est subdivisée en Quartiers, qui contiennent des sections et permettent des opérations telles que l'ajout de sections ou le calcul des scores partiels.
La classe Sections, à son tour, représente une partie d'un quartier et contient deux bâtiments.
La classe Bâtiment gère les informations des bâtiments, notamment leur type, leurs ressources nécessaires et leurs récompenses. Elle est déclinée en deux sous-classes.

- Travail, pour les bâtiments dit classique.
- Prestige, pour ceux apportant des bonus spécifiques.

2.3 Modélisation des transitions

Dès lors, nous avons pu commencer à réfléchir à l'aspect dynamique de notre projet, notamment en modélisant les transitions et les interactions qui

se produiraient au cours de son cycle de vie. Cette étape nous a permis de mieux comprendre le comportement du système dans différents scénarios et de définir les états clés ainsi que les événements déclencheurs.

Tout d'abord, à travers le diagramme d'état-transition, nous avons visualisé les différents états par lesquels le système peut passer au cours de son cycle de vie. Ce diagramme nous a permis de représenter de manière claire et intuitive les états clés du système, ainsi que les transitions possibles entre ces états, déclenchées par des événements spécifiques ou des conditions particulières.

Grâce à cette approche, nous avons pu identifier les scénarios critiques, les éventuelles boucles ou impasses, et les mécanismes nécessaires pour garantir la fluidité et la robustesse des transitions. Ce travail a également fourni une base solide pour concevoir les comportements dynamiques du système et assurer une gestion correcte des changements d'état en réponse aux interactions des utilisateurs ou aux événements externes.

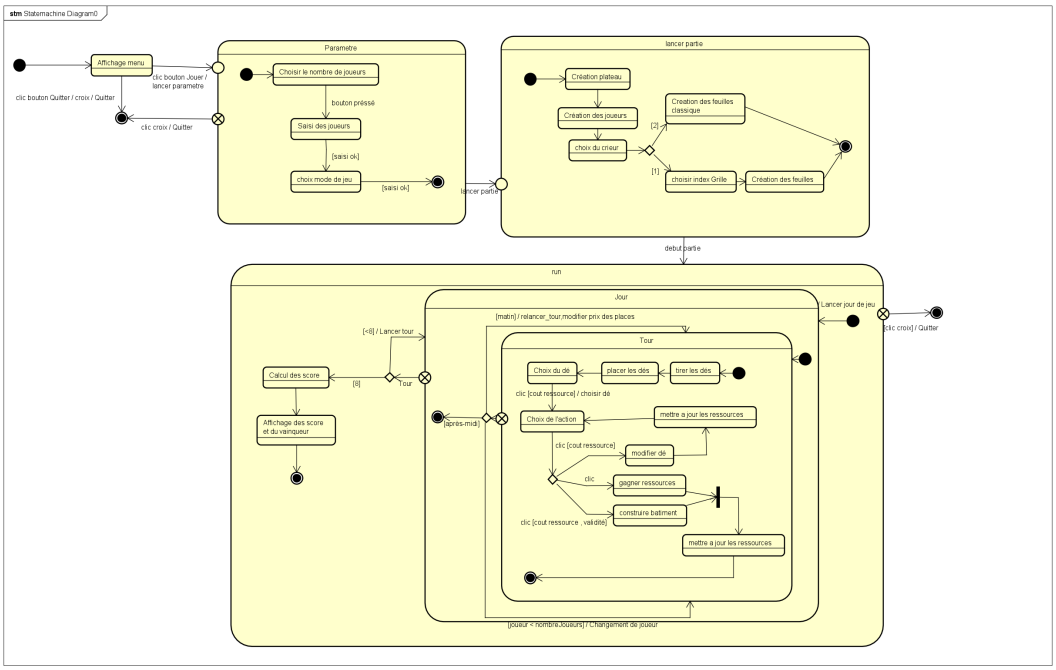


FIGURE 2.4 – Diagramme d'état transition

2.4 Elaboration de scénarii

Dans un second temps, à travers des scénarii, nous avons approfondi notre analyse en décrivant des cas spécifiques d'utilisation du système. Ces scénarii nous ont permis de détailler les interactions entre les acteurs et le système, en mettant en lumière les séquences d'actions nécessaires pour atteindre les

objectifs définis.

2.4.1 Initialisation du jeu

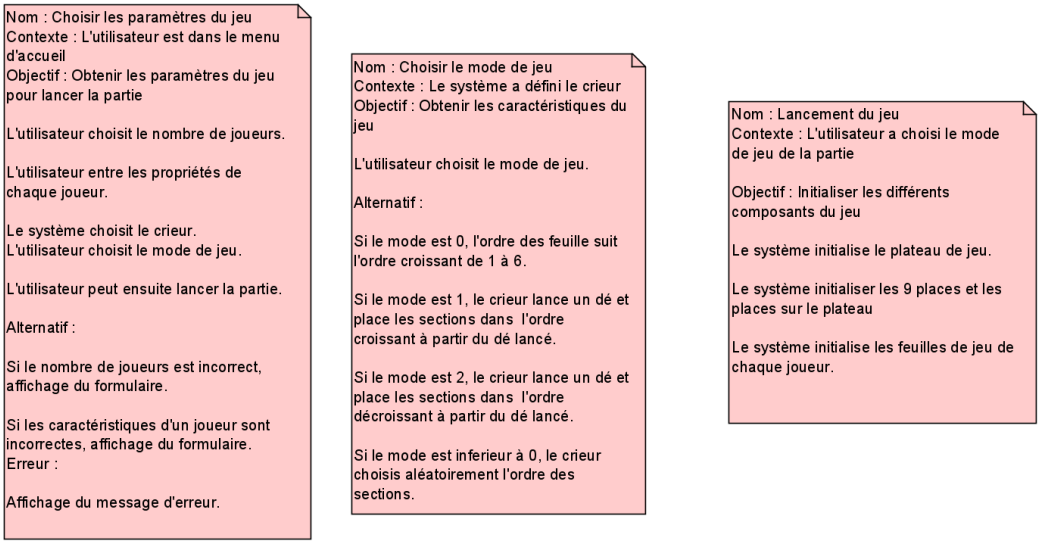


FIGURE 2.5 – Scenarii des paramètres et du lancement du jeu

2.4.2 Déroulement du jeu

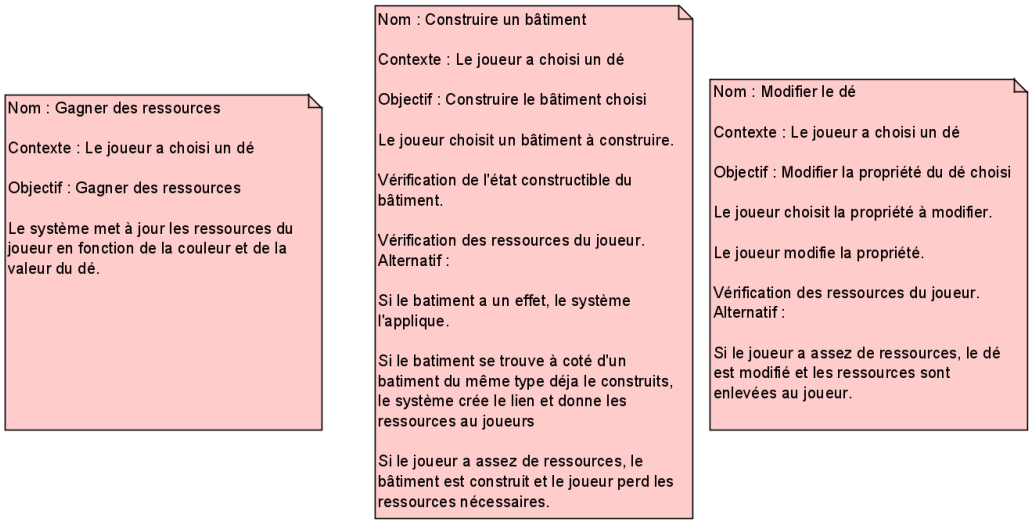


FIGURE 2.6 – Scenarii des actions possible

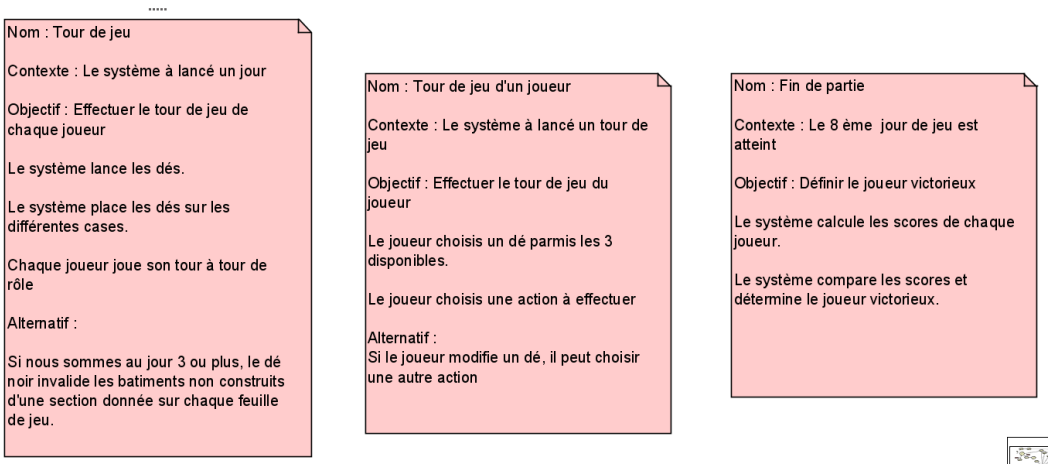


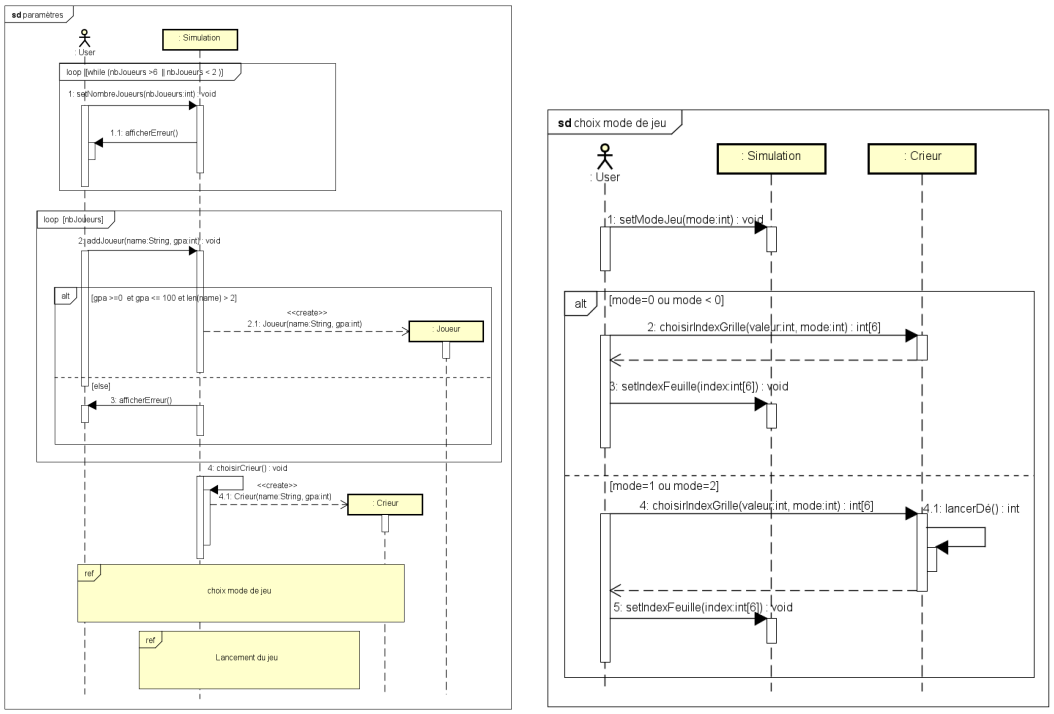
FIGURE 2.7 – Scenarii du déroulement du jeu

2.5 Diagramme de séquences

À partir de ces scénarii, nous avons élaboré des diagrammes de séquence pour modéliser les échanges entre les différentes classes et objets. Cela nous a permis d'identifier les responsabilités de chaque entité, de clarifier l'ordre des interactions, et de vérifier la cohérence des relations définies dans le diagramme de classe. Ces scénarii ont ainsi joué un rôle crucial pour valider et affiner la dynamique du système tout en garantissant qu'il réponde aux besoins fonctionnels identifiés.

2.5.1 Initialisation

Les diagrammes suivants illustrent les étapes nécessaires pour initialiser la partie. Cela inclut la définition des paramètres, le choix du mode de jeu et la création des objets du jeu.
Le diagramme (a) détaille le processus par lequel l'utilisateur entre les paramètres principaux, comme le nombre de joueurs et leurs caractéristiques. Le système vérifie la validité des données et les enregistre pour préparer la partie.
Le diagramme (b) montre comment l'utilisateur sélectionne le mode de jeu parmi plusieurs options (classique, rapide, etc.). Le système ajuste ensuite les règles et les conditions en conséquence.



(a) Initialisation des paramètres (b) Choix du mode de jeu

FIGURE 2.8 – Diagrammes de séquences liés aux paramètres du jeu.

Une fois que les différents paramètres sont définis, l'initialisation de la partie peut être effectuée. Le diagramme suivant décrit la création des instances nécessaires pour démarrer le jeu, telles que les joueurs, le plateau et les dés. Une fois cette étape terminée, la partie est prête à commencer.

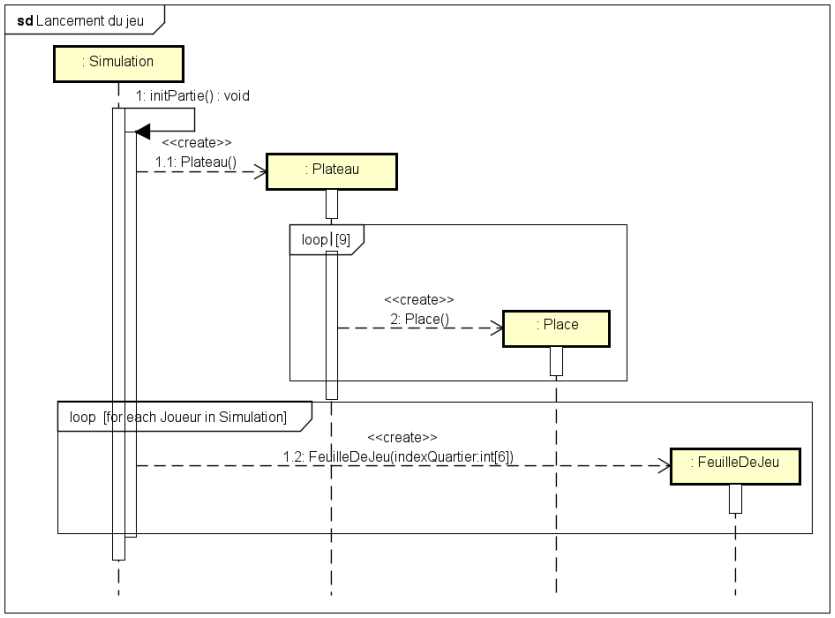


FIGURE 2.9 – Lancement de la partie

À ce stade, tous les composants nécessaires au jeu sont opérationnels, et la boucle principale du jeu peut commencer.

2.5.2 Actions disponibles

Les diagrammes suivants présentent les trois actions principales qu'un joueur peut effectuer pendant son tour. Ces actions permettent au joueur d'interagir avec le jeu de manière stratégique.

La figure 2.10 décrit l'action par laquelle un joueur peut dépenser des ressources pour construire un bâtiment sur le plateau. Cela modifie l'état du plateau et peut rapporter des points au joueur.
La figure 2.11 illustre comment un joueur utilise un dé pour collecter des ressources spécifiques. Ces ressources peuvent être utilisées pour d'autres actions, comme la construction de bâtiments.
La figure 2.12 décrit comment un joueur peut changer la valeur ou les propriétés d'un dé pour optimiser ses actions futures. Cela peut être fait avant d'effectuer une autre action pendant le tour.

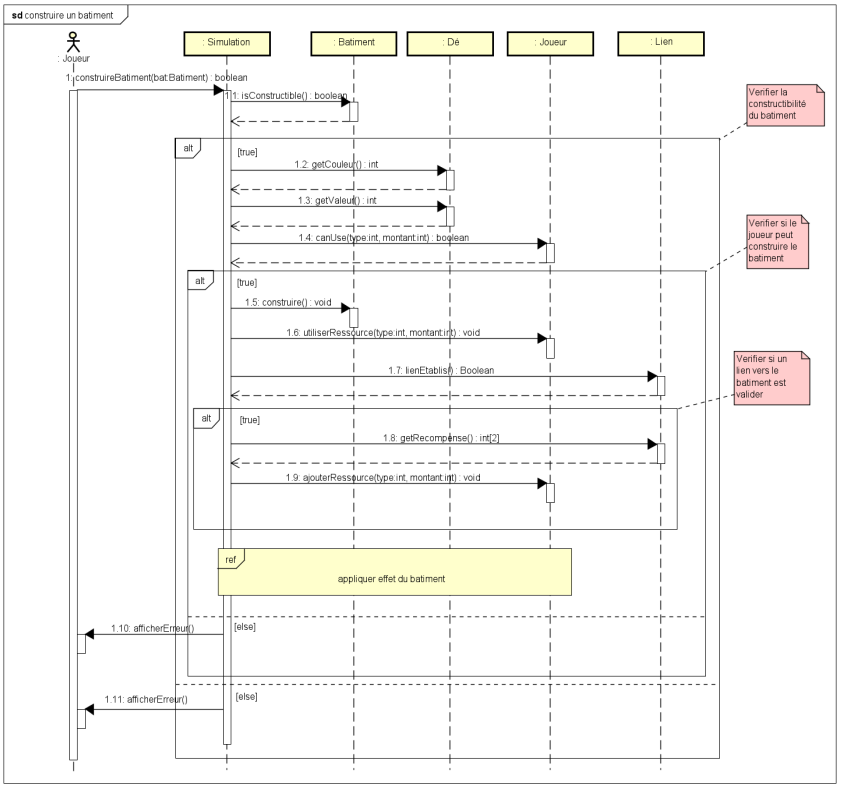


FIGURE 2.10 – Construire un bâtiment

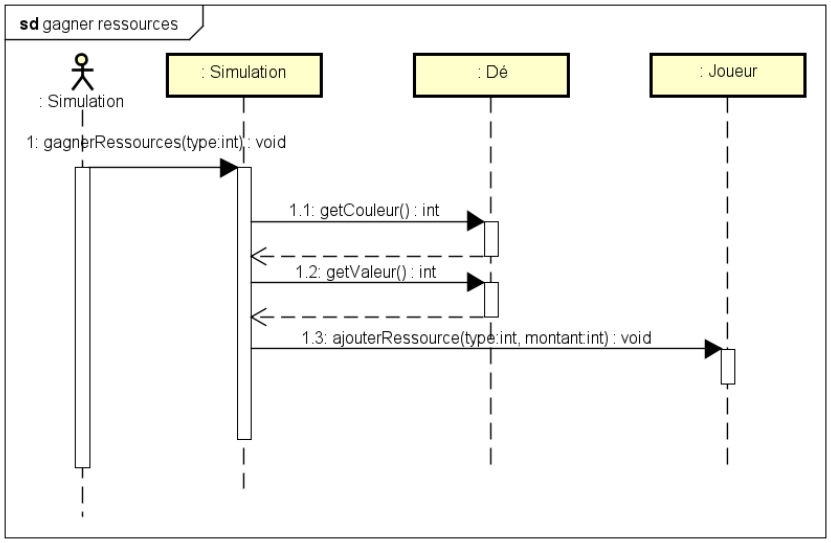
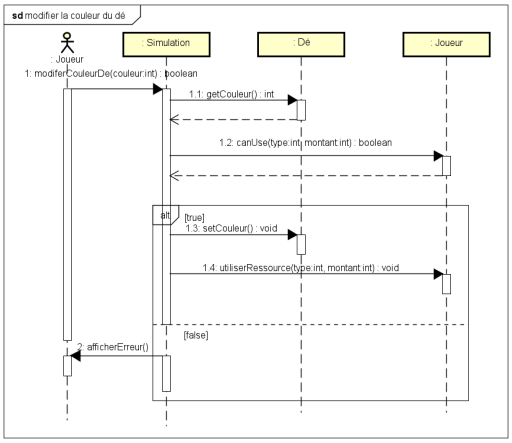
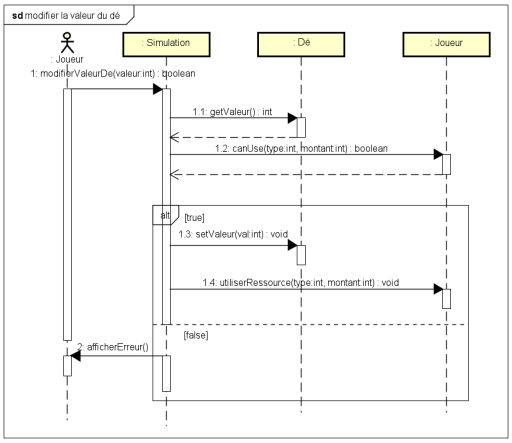


FIGURE 2.11 – Gagner des ressources



(a) Modification de la couleur



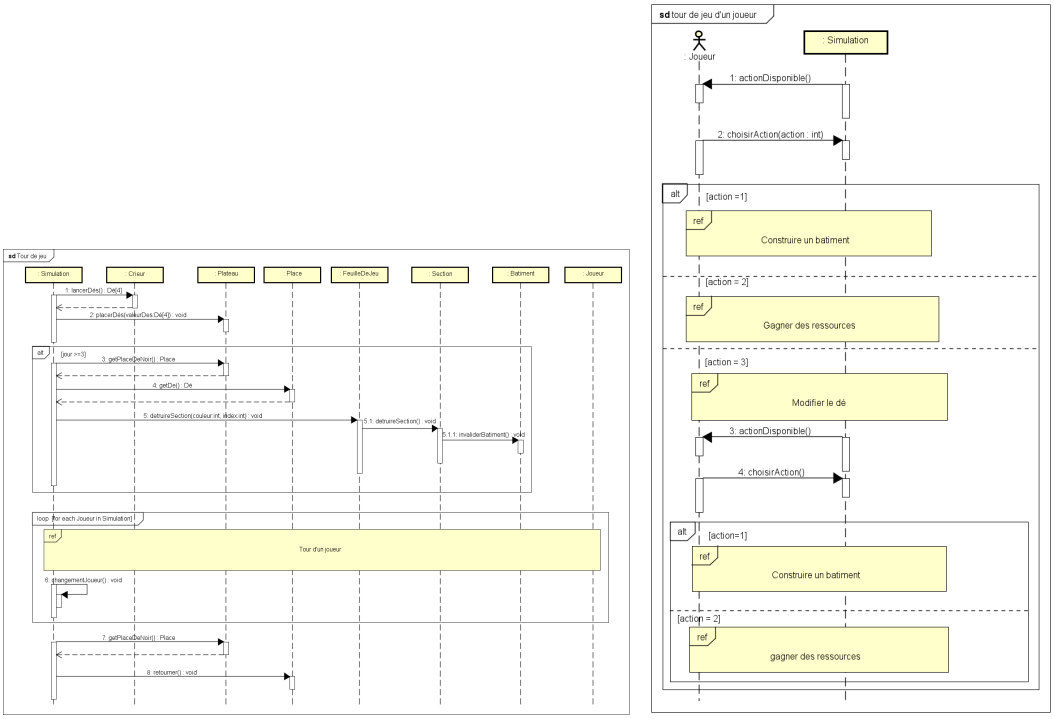
(b) Modification de la valuer

FIGURE 2.12 – Diagrammes de séquences liés à la modification des dés.

2.5.3 Déroulement du jeu

Le déroulement du jeu est divisé en plusieurs étapes importantes, décrites dans les diagrammes suivants.

Le diagramme (a) décrit la structure générale d'un tour. Chaque joueur effectue ses actions dans l'ordre, et le système met à jour les états du jeu après chaque tour.
Le diagramme (b) détaille les différentes étapes qu'un joueur suit pendant son tour, notamment le choix des actions et la résolution de leurs effets.

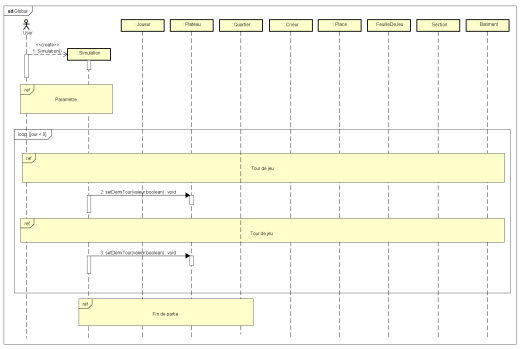


(a) Tour de jeu global (b) Tour de jeu d'un joueur

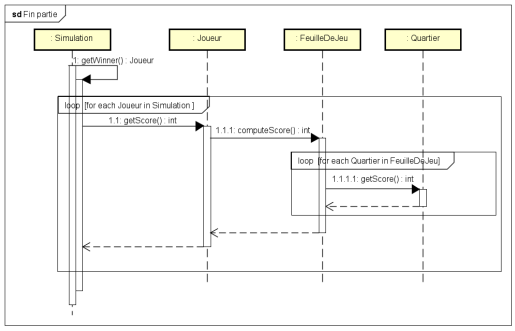
FIGURE 2.13 – Diagrammes de séquences liés au déroulement d'un tour de jeu.

Enfin, les diagrammes suivants offrent une vue globale du fonctionnement du jeu et de sa conclusion.

Le diagramme (a) présente une vision globale des interactions et des composants du jeu, montrant comment ils collaborent pour offrir une expérience cohérente.
Le diagramme (b) illustre les étapes menant à la conclusion du jeu, y compris la vérification des conditions de victoire et l'affichage des résultats.



(a) Vue d'ensemble



(b) Fin de partie

FIGURE 2.14 – Diagrammes de séquences globaux du jeu. Ils mettent en évidence le flux global du jeu, depuis le début jusqu'à la fin.

2.6 Adaptation du sujet à l'UTBM

Afin d'adapter ce jeu le plus possible à l'utbm nous avons décidé de modifier quelques caractéristiques de Troy Dice afin de le rapprocher de l'univers de l'UTBM.
De plus, nous avons retravailler l'aspect graphique du jeu, afin qu'il ait une charte graphique lié à l'UTBM.

2.6.1 Caractéristiques du jeu

Tout d'abord, nous avons décidé de renommer notre jeu pour l'appeler **UTBM Dice**.
En parallèle, certaines caractéristiques du jeu ont été modifiées, notamment les quartiers et les différentes monnaies utilisées.

Nous avons ajusté le type de chaque quartier afin de les rapprocher davantage de l'univers de l'UTBM. Ainsi, les quartiers sont désormais associés aux Alumni, Étudiants, et Chercheurs. Chaque quartier dispose de sa propre monnaie en lien avec son domaine : l'expérience, les crédits ECTS, et la connaissance, respectivement.

En outre, nous avons introduit une nouvelle variable pour les joueurs, le GPA. Cette valeur déterminera quel joueur assumera le rôle de crieur et sélectionnera le mode de jeu.

2.6.2 Aspect visuel

L'aspect visuel du jeu a été revisité afin de l'intégrer pleinement dans l'univers de l'UTBM. Pour ce faire, nous avons conçu nos propres illustrations pour chacun des éléments présents dans l'espace de jeu.

Ainsi, nous avons recréé le plateau de jeu, la feuille de jeu, les différents quartiers, ainsi que les dés, en nous inspirant de l'identité et de l'univers de l'UTBM.

FIGURE 2.15 – Menu d'acceuil - UTBM Dice

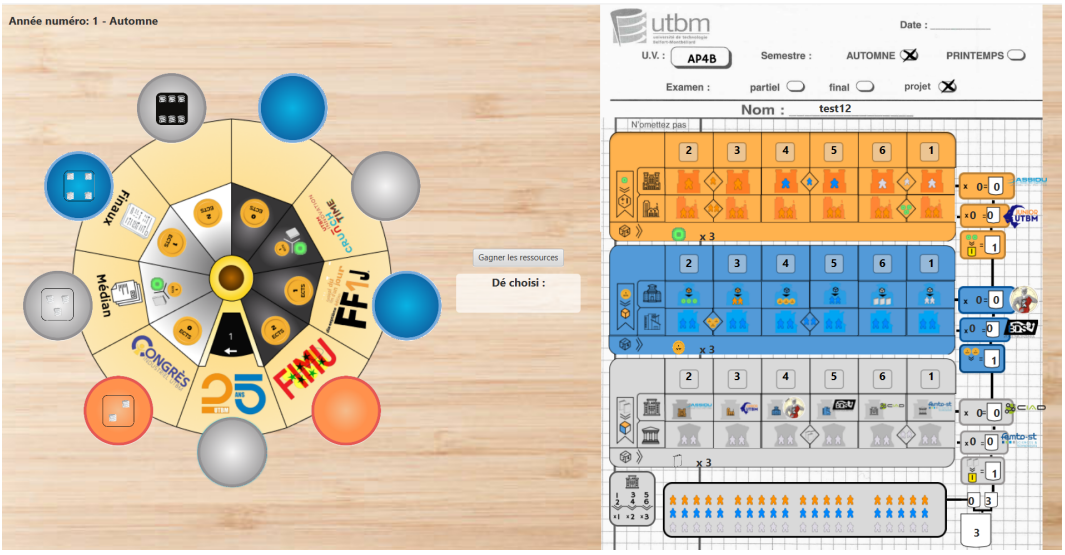


FIGURE 2.16 – Espace de jeu - UTBM Dice

FIGURE 2.17 – Ecran de fin de partie

2.7 Répartition du travail

2.7.1 Tom

J'ai travaillé sur l'élaboration des diagrammes de cas d'utilisation, de classe et de séquence, ainsi que sur la rédaction des scénarios pour clarifier les différentes interactions dans le projet.
J'ai également développé le code Java nécessaire pour créer l'intégralité de l'environnement de jeu.

2.7.2 Nicolas

J'ai travaillé sur les différents diagrammes pour vérifier qu'ils correspondent bien à nos attentes dans le respect des règles du jeu original. Je me suis aussi occupé de chercher des manières originales d'amener le thème de l'UTBM dans notre version du jeu.

2.8 Conclusion

Pour conclure ce rapport, ce projet nous a permis d'apprendre à concevoir l'architecture d'un projet élaboré en respectant le modèle MVC (Model - View - Controller) et les principes de la Programmation Orientée Objet. De plus, ce projet a été l'occasion d'apprendre à utiliser la librairie graphique JavaFx, ce qui nous a permis de concevoir une interface graphique élaborée tout en respectant strictement le modèle MVC.