

Rapport 1

Machi Koro



Table des matières

1	Introduction	2
2	Explication d'une partie de <i>Machi Koro</i>	3
2.1	De quoi est composé le jeu ?	3
2.1.1	Les cartes	3
2.1.1.1	Les bâtiments	3
2.1.1.2	Les monuments	4
2.1.2	Les dés	4
2.1.3	Les pièces	4
2.2	Quel est le but du jeu ?	4
2.3	Comment atteindre ce but ?	5
2.3.1	Début de partie	5
2.3.2	Déroulé de partie	5
2.3.3	Fin de partie	6
2.3.4	Illustration	6
3	Analyse du sujet	7
3.1	Analyse des modules	7
3.2	Analyse des classes	7
3.2.1	Carte	7
3.2.2	Monument	7
3.2.3	Batiment	7
3.2.4	Joueur	8
3.2.5	Piece	8
3.2.6	Partie	8
3.2.7	Effet	8
3.2.7.1	Transaction	8
3.2.7.2	Changer_Val_De	8
3.2.7.3	Echange	9
3.2.7.4	Rejouer	9
3.2.8	EditionDeJeu	9
3.2.9	Dé	9
3.3	Analyse des concepts	9
3.3.1	Classe et objet	9
3.3.2	Membres statiques	10
3.3.3	Héritage et héritage multiple	10
4	Première version de l'UML	11
5	Liste des tâches	12
6	Conclusion	13

1 Introduction

Durant ce semestre et dans le cadre de l'UV LO21, nous allons travailler sur un projet de groupe. Notre projet de groupe répond au sujet disponible [ici](#). Le but de ce dernier est de nous familiariser avec la programmation orientée objet en produisant une version numérique du jeu *Machi Koro*. Nous mettrons en application dans ce projet les différentes notions vues en cours de LO21.

Dans ce premier rapport, nous présenterons les premières analyses que nous avons faites sur le sujet. Nous les expliciterons à l'aide d'un premier schéma UML. Ce schéma sera amené à être modifié durant nos différentes réunions jusqu'à atteindre une version suffisamment proche de la réalité.

Nous allons dans un premier temps expliquer brièvement le fonctionnement d'une partie de jeu de *Machi Koro*. Ensuite, nous expliquerons notre compréhension du sujet vis-à-vis des modules, des classes et des concepts à utiliser. Enfin, nous présenterons notre première version du schéma UML ainsi que la répartition des tâches.

2 Explication d'une partie de *Machi Koro*

Nous allons dans un premier temps présenter les principaux aspects d'une partie de *Machi Koro*. Pour cela, nous allons répondre à trois questions essentielles.

Les questions sont les suivantes :

- De quoi est composé le jeu ?
- Quel est le but du jeu ?
- Comment atteindre ce but ?

2.1 De quoi est composé le jeu ?

Machi Koro est un jeu de table se jouant avec des dés, des cartes et des pièces. Les différentes extensions permettent d'ajouter de nouvelles cartes. Nous allons voir dans le détail ces différents éléments.

2.1.1 Les cartes

Les cartes sont les éléments centraux d'une partie de *Machi Koro*. Elles sont présentes dans la pioche, dans les "villes" de chaque joueur et dans le paquet central. Ce dernier est composé de 9 tas de cartes distinctes : si une carte piochée est déjà présente dans le paquet central, on l'empile sur celle-ci et on pioche à nouveau jusqu'à obtenir une carte qui n'est pas déjà présente. Les cartes constituant un jeu peuvent varier en fonction de l'édition et des extensions.

Les cartes sont divisées en deux types distincts :

- les bâtiments
- les monuments

2.1.1.1 Les bâtiments

Les bâtiments sont des cartes qui s'achètent au fur et à mesure de la partie. Ces cartes ont chacune une propriété, un effet, qui consiste le plus souvent à de l'échange de cartes et/ou d'argent. La condition d'activation de l'effet des bâtiments est un score à faire en lançant les dés : si la somme des dés correspond au numéro ou à l'intervalle de numéros spécifié sur la carte, alors la carte active son effet.

Les bâtiments peuvent être de différents types. Ce dernier va influencer sur leur condition d'activation et leur effet, par exemple pour l'édition *Deluxe* :

1. **Type Rouge** : Une carte de ce type s'active quand un autre joueur que celui qui la possède fait un certain score aux dés. Ce joueur devra une certaine somme d'argent au propriétaire de la carte.
2. **Type Vert** : Une carte de ce type s'active quand le joueur qui la possède fait un certain score aux dés. Ce joueur recevra une certaine somme d'argent de la banque.
3. **Type Bleu** : Une carte de ce type s'active quand n'importe quel joueur fait un certain score aux dés. Le joueur possédant la carte recevra une certaine somme d'argent de la banque.

4. **Type Violet** : Une carte de ce type s'active quand le joueur qui la possède fait un certain score aux dés. En fonction de la carte, ce joueur recevra une certaine somme d'argent du joueur de son choix, de chaque joueur ou encore pourra échanger avec le joueur de son choix un bâtiment qui n'est pas de type violet.

Il est possible d'acheter plusieurs fois un bâtiment afin de l'obtenir en plusieurs exemplaires et ainsi dupliquer son effet. Cela est possible pour tous les bâtiments, sauf ceux de type **violet**, qu'on ne peut acheter qu'en un seul exemplaire chacun.

2.1.1.2 Les monuments

Les monuments sont des cartes que chaque joueur possède dès le début de la partie. Ces monuments seront d'abord "inactifs", "en construction". Les joueurs doivent payer pour les débloquent. Un joueur ne peut pas acheter d'autres monuments que ceux qu'il possède déjà en début de partie. Ces cartes ont chacune une propriété, un effet, qui est en général varié et différent de ceux des bâtiments, de part sa nature ou son moment d'activation (ex : pouvoir rejouer si on a fait un double, pouvoir lancer deux dés au lieu d'un seul, etc). L'effet de ces cartes ne peut être activé que si la carte elle-même activée.

2.1.2 Les dés

Chaque joueur a la possibilité de lancer un dé ou, s'il possède le monument le permettant, deux dés par tour. Les dés utilisés sont des dés à six faces, ce qui nous laisse penser que les "meilleures" cartes bâtiment à choisir sont celles qui ont un numéro d'activation autour de 6, 7 et 8, nombres ayant le plus de "chance" d'être obtenus en sommant le score de deux dés de six faces.

2.1.3 Les pièces

Le dernier élément du jeu est l'argent fictif permettant d'acheter des cartes (acheter des bâtiments ou activer des monuments). Pour l'argent, nous prévoyons dans un premier temps de le représenter comme une simple valeur numérique rattachée à chaque joueur, puis, dans un second temps, de modéliser réellement les pièces en implémentant dans l'interface graphique un affichage similaire à celui qu'on retrouve dans la réalité : des pièces de 1, 5 et 10.

2.2 Quel est le but du jeu ?

Les joueurs sont chacun maire d'une ville fictive. En conséquence, ils vont acheter des cartes qui représentent divers types d'établissements (des monuments et bâtiments) qui vont composer leur ville. Le premier joueur à avoir activé un nombre prédéfini de cartes "Monument" gagne instantanément la partie. Le nombre de monuments présents dans le jeu (et donc le nombre de monuments nécessaires pour gagner) va dépendre de l'édition et des différentes extensions utilisées pour la partie.

2.3 Comment atteindre ce but ?

2.3.1 Début de partie

Chaque joueur commence la partie avec 2 cartes "établissement" :

- le "Champs de blé"
- la "Boulangerie"

Les joueurs ont également devant eux leurs monuments, initialement non activés, sur leur face "construction". Leur nombre varie selon les éditions et extension, par exemple pour l'édition *Deluxe*, il y a six monuments :

- Le "Port"
- La "Gare"
- Le "Centre Commercial"
- Le "Parc d'Attractions"
- La "Tour Radio"
- L' "Aéroport"

Ces monuments et bâtiments constituent la ville de départ de chaque joueur.

Il faut ensuite placer au centre des joueurs 9 cartes différentes en grille 3*3. Comme expliqué plus haut, si on tire une carte qui est déjà présente sur le tas central, empile les deux cartes correspondantes et on continue à piocher jusqu'à obtenir 9 piles de cartes différentes.

Chaque joueur commence la partie avec 3 pièces de valeur 1. Les pièces restantes sont placées au centre de la table, formant ainsi la Banque.

Selon les règles officielles, le joueur ayant visité un parc d'attractions le plus récemment commence la partie. Toutefois, cela n'est pas obligatoire, le joueur qui débute peut être décidé avec un lancer de dé ou autre.

2.3.2 Déroulé de partie

Les joueurs jouent chacun leur tour dans le sens horaire, un tour se déroule comme suit :

1. Le joueur lance le ou les dés
2. On déclenche les établissements qui correspondent au résultat du jet de dé
3. Ensuite, le joueur choisit soit :
 - De construire un nouveau bâtiment
 - D'activer un nouveau monument
 - De ne rien construire pour économiser
4. On renouvelle les piles de cartes au centre des joueurs

Il arrive que lors d'un tour, beaucoup de transactions se fassent, ces dernières doivent suivre un certain ordre.

Lors du tour d'un joueur, ce dernier effectue en premier les paiements qu'il doit aux autres joueurs (cartes rouges). Si ce dernier n'a pas assez pour tout payer, il donne ce qu'il a et le reste de la dette est annulée. Si le joueur doit payer plusieurs joueurs lors d'un tour, il les paye dans le sens anti-horaire. Si un joueur possède 3 pièces, en doit 5 à son voisin directement à sa droite et 2 à son deuxième voisin à droite, alors il donnera ses 3 pièces à son voisin de droite et rien à l'autre voisin, le reste de la dette étant annulée.

Une fois les paiements effectués, le joueur reçoit l'argent de ses cartes (si elles ont été activées). Si au cours d'un tour, un joueur n'a pas pu payer une dette et reçoit de l'argent par la suite, il ne doit rien du tout au joueur envers qui il avait une dette. Par exemple, si un joueur a 0 pièces, avait une dette de 3 à un joueur qui a donc été annulée, puis par la suite du tour reçoit 2 pièces grâce à un de ses bâtiments, il gardera ses deux pièces pour lui et ne devra rien à l'autre joueur.

Si un joueur n'a rien construit lors d'un tour et dispose de moins de deux pièces, alors il reçoit des pièces de la banque jusqu'à avoir deux pièces.

2.3.3 Fin de partie

La partie se termine lorsqu'un joueur a obtenu le nombre de monuments nécessaire pour remporter la partie. Ce nombre varie selon l'édition et les extensions, il faut par exemple quatre monuments pour gagner dans la version standard du jeu, tandis que cinq sont nécessaires dans la version *Deluxe*.

2.3.4 Illustration

Pour illustrer les éléments que nous venons de décrire, nous avons mis en annexe une image tirée des règles du jeu représentant les cartes d'un joueur (monuments et bâtiments), son argent, le tas central, les dés, la pioche et la banque. [Voir l'annexe](#)

3 Analyse du sujet

3.1 Analyse des modules

Les modules sont d'après notre interprétation du sujet un groupement de classes. Voir le projet comme un groupement de classes simplifiera par la suite leur modification ainsi que leur implémentation. Nous allons donc pouvoir regrouper les classes entre elles par rapport à une notion commune.

Nous avons ainsi identifié trois principaux modules : un module Joueur, un module Carte, et un module Partie/Jeu. Nous allons par la suite étudier les différents liens qui les relient.

3.2 Analyse des classes

Dans notre première analyse des classes dont nous allons avoir besoin pour modéliser le jeu, nous avons identifié 13 classes. Le nombre de classes nécessaires est amené à évoluer au fil des premières semaines sur le projet.

3.2.1 Carte

Nous avons choisi de représenter par une classe le composant indispensable de notre jeu : les cartes. Les cartes possèdent toutes un nom, un prix d'achat (ou d'activation dans le cas des Monuments) et une image qui correspond au rendu visuel de la carte dans le jeu. Le visuel de la carte ne sera développé que si le temps disponible nous le permet, nous allons d'abord rendre le jeu fonctionnel. Chaque carte peut déclencher un effet, que nous présenterons ultérieurement. Les cartes peuvent être de deux types et sont soit des Bâtiments, soit des Monuments. Ainsi, nous avons choisi un héritage exclusif pour modéliser ce phénomène. Bâtiment et Monument hériteront tous deux de la classe Carte. Nous avons choisi de distinguer ces deux types de carte, car leur fonctionnement diffère trop pour pouvoir nous permettre de les représenter au sein de la classe Carte.

3.2.2 Monument

Un monument étant une Carte, la classe Monument hérite de la classe Carte. La classe Monument dispose d'un attribut `momenteffet`, qui représente le moment où l'effet se déclenchera. Cela peut être avant que le joueur joue (ex : possibilité de lancer 2 dés), pendant que le joueur joue (ex : rajouter 2 si le total des dés est supérieur à 10) ou après que le joueur joue (ex : rejouer). Quant à l'attribut `actif`, il indique si le joueur a payé pour activer le monument.

3.2.3 Bâtiment

La classe Bâtiment hérite, elle aussi, de Carte. Un bâtiment est caractérisé par sa couleur (les types dont nous avons parlé plus haut), ses numéros d'activation (nombres qu'il faut faire pour déclencher l'effet), et un type (situé à gauche du nom de la carte, il y a des sous-types au sein des couleurs des cartes).

3.2.4 Joueur

Un joueur est modélisé par l'attribut nom, qui va être la clé de la classe (ainsi, deux joueurs ne pourront pas avoir le même pseudo, beaucoup de jeux numériques utilisent le pseudo comme clé primaire). La classe comporte aussi une liste de bâtiment et une liste de monuments. Ainsi, nous avons décidé que chaque joueur n'aurait pas une liste de cartes, mais une liste de bâtiments et une liste de monuments. Ce choix a été effectué pour simplifier le traitement informatique. En effet, on pourra itérer sur la liste des monuments (qui contiendra un nombre fixe de cartes) et sur la liste de bâtiments (qui sera une liste susceptible d'évoluer au fil de la partie). Un joueur possède plusieurs méthodes. En effet, il peut acheter une carte ou appeler la méthode Argent() pour savoir quel est son solde.

3.2.5 Piece

La classe Pièce représente une pièce du jeu. Une pièce est représentée par sa valeur (qui peut être 1, 5 ou 10). Un joueur peut donc posséder plusieurs pièces tandis qu'une pièce n'appartient qu'à un joueur maximum (0 si la pièce appartient à la banque).

3.2.6 Partie

La classe Partie est la classe qui gère une partie de jeu. Entre autres, elle va organiser les tours de jeu. La classe agit directement sur la classe Joueur, car une partie est une suite de tours, et un tour est une suite d'actions faite par les joueurs.

3.2.7 Effet

La classe Effet est dans notre modèle une classe à part entière intrinsèquement liée à la classe Carte, un effet est spécifique à une carte et une carte possède un seul effet. Étant donné cela, nous ferons probablement un héritage par classe mère par la suite. Un effet possède une description. Un effet peut être de différentes natures, nous les avons représentés en différentes classes dans l'UML mais en pratique, dans l'implémentation, ces classes ne seront pas représentées, la méthode DeclencherEffet() sera juste différente en fonction de la carte. Voici une liste non exhaustive de différents types d'effet :

3.2.7.1 Transaction

La classe Transaction modélise l'effet de transaction qu'une carte peut avoir sur le jeu. Cette dernière peut avoir effet soit sur un joueur ou depuis la banque (selon la carte). Beaucoup de cartes ont un effet de ce type.

3.2.7.2 Changer_Val_De

Cette classe modélise un effet assez rare. En effet, cas où l'on peut modifier la valeur obtenue par les dés juste lancés.

3.2.7.3 Echange

Cette classe représente le cas des échanges. Ces échanges peuvent être de différentes natures. Par exemple, il est possible d'échanger une carte avec un autre joueur, en réceptionnant de l'argent de la banque ou non. Parfois, les échanges considèrent des cartes violettes rares.

3.2.7.4 Rejouer

Cette classe modélise l'effet de jouer à nouveau pour un Joueur venant de finir son tour. Il pourra à nouveau jouer, lui permettant d'acheter de nouvelles cartes et d'espérer remporter des gains.

Bien entendu, toutes ces classes énumérant les différents effets de cartes héritent de la classe Effet.

3.2.8 EditionDeJeu

Cette classe permet de déterminer le cadre de la partie, avec quel ensemble de cartes la partie va se dérouler. Selon les éditions, *Machi Koro* se joue avec un nombre de joueurs différent, a des modalités de victoire différentes, un nombre de dés différent... Tout cela est pris en compte dans cette classe. Ainsi, EditionDeJeu a une influence et donc un lien sur la classe Carte, car une édition est avant tout un ensemble de cartes. Mais EditionDeJeu influe aussi sur le nombre de dés jouables (selon les éditions, certaines cartes permettent de jouer plus de dés).

3.2.9 Dé

La classe Dé est une classe très simple, mais non pas moins essentielle, car c'est elle qui cadence nos tours. Elle est en lien direct avec la classe Joueur (car c'est un joueur qui lance les dés). Elle possède les attributs min (toujours égal à 1) et max qui peuvent varier selon les éditions de jeu, ce qui explique son lien avec la classe EditionDeJeu. Nous avons choisi d'en faire une classe à part entière, car selon nous, un dé est un objet dissocié d'un joueur, les dés sont les mêmes pour tout le monde. Nous ne considérons pas dé directement comme une méthode de Joueurs.

3.3 Analyse des concepts

Nous allons utiliser de nombreux concepts que nous avons vus en cours. Nous allons dans cette partie les aborder un par un et expliquer leur utilisation dans notre projet.

3.3.1 Classe et objet

Ces concepts sont à la base de notre projet. Ils viendront, par leur nature, diviser les différentes notions liées au projet afin d'en expliciter leur existence. Dans notre cas, les objets physiques cartes (par exemple) seront modélisés par des objets d'une classe que l'on appellera carte. Ainsi, une carte devient un objet manipulable,

associable, destructible, assignable (nous aborderons ces thématiques dans un prochain rapport quand nous aurons une idée plus claire des structures de nos classes). Chaque objet ou concept du jeu (une partie n'est pas physiquement manipulable, mais sera un objet dans notre jeu) sera modélisé sous forme de classe. Les liaisons que nous établirons d'abord en UML puis implémenterons en *C++* formera notre jeu. C'est pourquoi le concept d'objet et de classe est fondamental dans notre sujet.

3.3.2 Membres statiques

Les membres statiques seront utilisées dans notre projet, notamment lorsque nous devrons implémenter des objets communs à toutes les classes. Par exemple, la classe *EditionDeJeu* risque (nous n'avons pas encore déterminé de manière sûre) d'être un singleton, ce qui nécessitera l'utilisation de membres statiques. Une autre utilisation des membres statiques sera aussi pour définir des attributs "constants" à toutes les instances d'une classe. C'est le cas de l'attribut `prix_min` d'une carte.

3.3.3 Héritage et héritage multiple

L'héritage sera fondamental dans notre projet. En effet, par la structure du jeu, il existe différents types de cartes (monuments et bâtiments) et dans ces types de cartes, il existe les boulangeries, les épiceries, etc. Autant de cartes qui nécessiteront probablement leur propre classe (héritant de bâtiment ou de monument en fonction de la carte), pour permettre définir leur effet. C'est ici que le concept d'héritage nous aidera à la tâche, car les attributs et la structure de chaque classe mère n'aura pas à être redéfinie dans les classes filles, ces dernières ne contiendront que les ajouts par rapport à leur classe mère. De plus, l'héritage présente un avantage pour l'extensibilité de l'application. Dans le cas de l'ajout d'une nouvelle extension, nous allons devoir ajouter de nouvelles cartes. Il n'y aura donc pas besoin de modifier le code et les fonctions déjà existantes, tout ce qu'il y aura à faire sera de définir ces nouvelles cartes dans un fichier, en indiquant les caractéristiques de la carte (prix, effet, etc), puis d'inclure ce fichier au reste du code.

4 Première version de l'UML

Bien que l'UML ne soit pas obligatoire pour le premier livrable, nous avons choisi de le développer afin de condenser et d'organiser nos différentes idées sur le projet. De ce fait, le schéma UML que nous présentons ci-dessous sera amené à changer.

L'UML ici présent comporte les premières notions vues en cours : composition, agrégation et modules. Elles seront amenées à évoluer et à être remise en question.

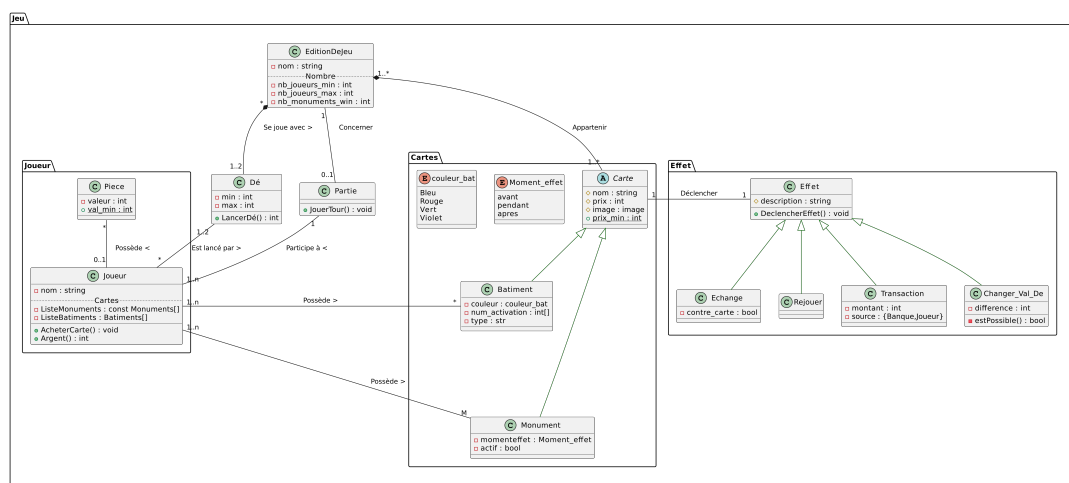


FIGURE 1 – Diagramme UML d'une partie de *Machi Koro*

Afin de sectionner notre projet en parties et gagner du temps par la suite, nous avons opté pour une représentation avec des packages. Ainsi, le jeu est composé d'un package **Joueur**, d'un package **Carte** et d'un package **Effet**. Chaque package représentera une partie qui pourra être traitée indépendamment des autres avant la mise en commun pour les lier.







5 Liste des tâches

Cette section est dédiée à l'avancée des tâches du projet. En particulier, on y trouvera les tâches à faire, celles en cours, et celles terminées. De semaine en semaine, nous reprendrons les différentes tâches et les feront changer de partie en fonction de leur état d'avancement.

Le format de chaque tâche est le suivant :

icone : tache ; personne en charge ; duree estime ; duree reelle



À faire :

-  : Planning Gantt créé et utilisable ; tous ; 3 heures
-  : Correction de l'architecture si besoin ; tous ; 2 heures
-  : Modification de l'architecture avec l'intégration des extensions ; tous ; 3 heures
-  : Ajout des attributs et méthodes nécessaires au bon fonctionnement du jeu ; tous ; 3 heures
-  : Intégrer les premières classes en *C++* ; tous ; 6 heures
-  : Créer l'interface graphique avec QtCreator et l'implémenter dans notre projet ; tous ; Durée Indéterminée

En cours :

-  : Seconde version de l'UML ; tous ; 1 semaine

Fait :

-  : Réalisation d'une esquisse d'UML ; tous ; 2 heures ; 2 heures
-  : Répartition des tâches ; tous ; 30 minutes ; 1 heure

6 Conclusion

Ces trois premières semaines de projet nous ont permis de nous familiariser avec le jeu Machi Koro. En particulier, elles nous ont permis de comprendre le fonctionnement du jeu en y jouant. Nous avons alors pu éclaircir les notions et concepts qui nous seront utiles pour mener à bien le projet.

Ce projet sera pour nous l'occasion de mobiliser les connaissances théoriques acquises lors des cours magistraux et travaux dirigés en les appliquant à un cas concret.

La réalisation du planning nous permettra de fixer des dates butoirs afin de réaliser le projet avec les contraintes temporelles imposées.

Annexe



FIGURE 2 – Représentation du jeu

Table des figures

1	Diagramme UML d'une partie de <i>Machi Koro</i>	11
2	Représentation du jeu	14