Arnaud Curie | Pascaline Guichard | Juliette Gagnepain |

Guillaume Clerc | Johann Fouchard | Noa Barbosa

IUT DIJON | DEPARTEMEnt informatique

Livrable n°2

Dossier de conception



# Introduction

Dans ce dossier de conception, vous verrez ce que nous avons fait pour la partie conception de notre projet. Mais avant tout, il nous faut rappeler quel est notre projet :

Notre projet consiste en la réalisation d’un clone du jeu *The Binding of Isaac*. C’est un jeu indépendant d’action-aventure en 2D isométrique. Le style du jeu est volontaire gore et effrayant mais est adouci par des designs cartoons. Nous avons décidé pour notre clone de reprendre un style similaire dans l’univers d’Alice au pays des merveilles. L’objectif du projet est de réaliser la version la plus abouti possible du jeu.

Le projet étant d’envergure et nos idées pouvant changer, il serait irréaliste d’essayer de faire une conception complète. En effet, d’une part nous oublierons certainement des éléments mais aussi nous comptons rajouter des éléments au fur et à mesure de notre progression. Nous avons donc fait une conception très large pouvant accueillir beaucoup d’élément.

# Backlog produit

Nous avons commencé par réaliser un backlog produit. Un backlog produit a la même utilité qu’un Use-Case mais sous une forme différente. Nous allons lister les fonctionnalités qu’attend notre client en les rangeant par grande catégories et par importance. De plus nous rajoutons les dépendances entre chaque fonctionnalité.

Une image contenant table

Description générée automatiquement

Une image contenant table

Description générée automatiquementUne image contenant table

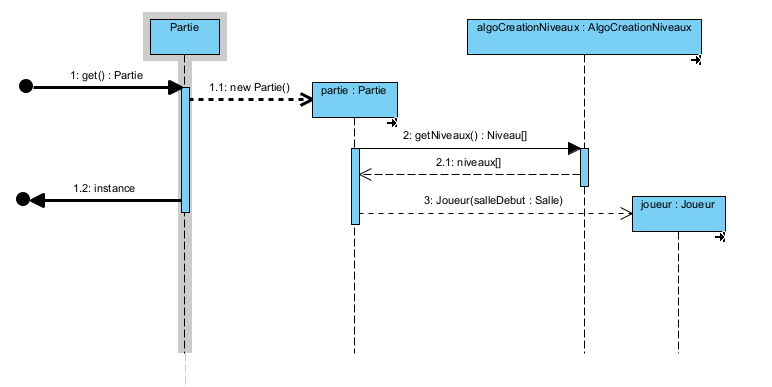
Description générée automatiquement

# Diagramme de cas d’utilisation

# Diagrammes de séquence

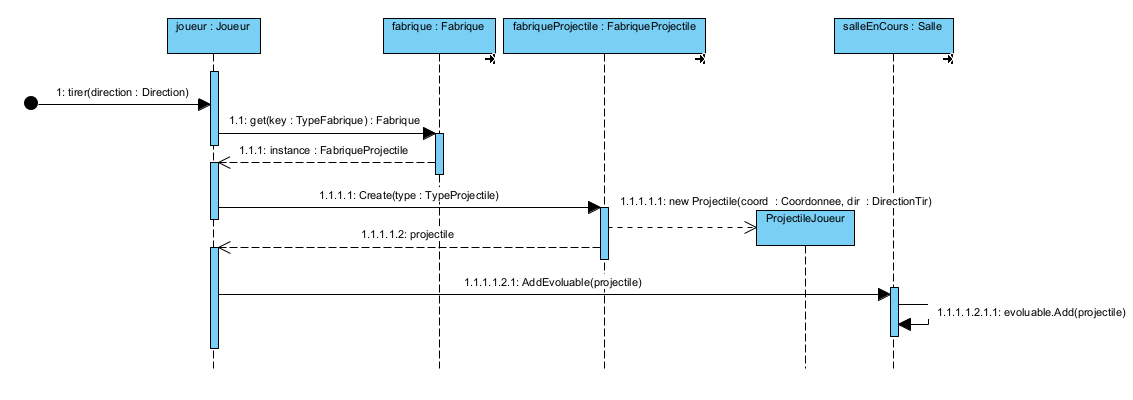
Les cas d’utilisations étant fait nous avons réaliser les diagrammes de séquences pour ces mêmes cas d’utilisation.

## Lancer une partie



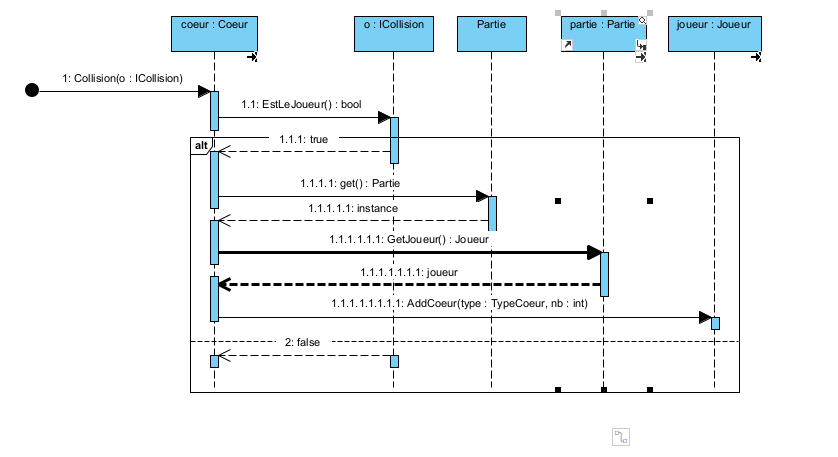
Dans ce diagramme nous souhaitons lancer une partie. Nous avons mis la classe Partie en singleton car nous allons avoir besoin de l’appeler régulièrement dans notre code. Nous l’appelons donc et à sa création elle demande un tableau de Niveau à une interface nommé AlgoCreationNiveaux. Pour notre clone d’Isaac, il nous faudra des algorithmes de créations de niveaux, de cartes, de salles… C’est à cela que sert cette classe. N’ayant pas encore les compétences pour ces algorithmes, nous les modélisons sous la forme d’une classe qui contient simplement un tableau de Niveaux qu’elle nous renvoie.

## Le joueur tire

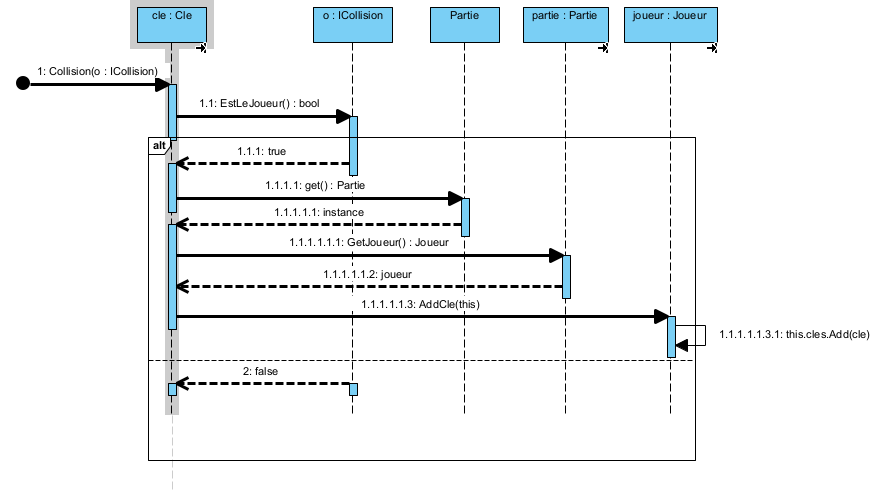


Voici le diagramme de séquence du joueur qui tire. Au moment où on demande au joueur de tirer, il appel l’instance de la fabrique correspondante pour instancier un projectile. Puis le joueur ajoute à la salle en cours le projectile. Le joueur connait ça salle en cours et la salle contient plusieurs interface nommé Evoluable.

## Collision du joueur avec un cœur

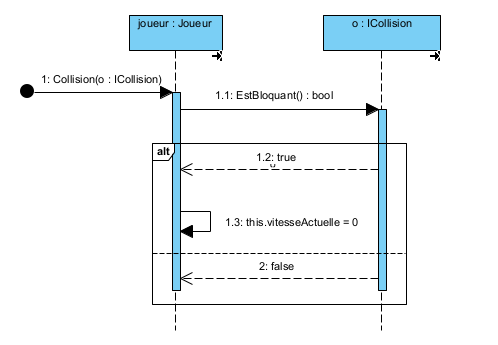
Dans ce diagramme, l’IHM va appeler le cœur pour lui dire qu’il est actuellement en collision avec un autre objet. Le cœur va donc demander si cet objet est le joueur. Si non, alors la fonction s’arrête. Mais si oui, alors le cœur récupère l’instance du singleton Partie pour ainsi récupérer l’instance du joueur. Le cœur pourra donc ajouter son type et le nombre de cœur qu’il est. Le joueur ne stocke pas directement une instance de cœur car nous prévoyons qu’il y est plusieurs types de cœur et qu’il soit géré en demi-cœurs comme sur le jeu de base. Cette modélisation rendra les algorithmes plus faciles.

## Collision avec une clé



Ce diagramme est très similaire au précédent. La clé reçoit l’information qu’elle touche un objet. Elle lui demande s’il est le joueur. Si nan, il ne se passe rien. Si oui, alors elle récupère l’instance du joueur par le biais du singleton partie et ainsi elle s’ajoute elle-même aux joueurs. Remarquez que cette fois le joueur contient directement les instances des clés. Cela nous permettra un code plus simple étant donné que nous n’avons pas à différencier différents types de clé.

## Collision du joueur



Ce diagramme de séquence nous présente ce qu’il se passe lorsque le joueur est en collision avec un autre objet. Il lui demande simplement s’il est bloquant. Si oui, alors nous mettons sa vitesse à 0. Si non, il ne se passe rien.

# Diagrammes de classe

## Diagramme métier

## Multiton de fabriques abstraites

Nous avons décidé d’utiliser le design pattern de la fabrique abstraite, le trouvant particulièrement adapté à la situation. Nous avons remarqué que nous aurons besoin de nombreuses fabriques pour notre projet. Et en sachant que la fabrique abstraite est elle-même un singleton, nous avons décidé de combiner la fabrique abstraite avec un autre design pattern : le multiton. Cela nous permet d’appeler depuis n’importe où dans notre code nos fabriques abstraites.