Livrable 1

Modélisation UML

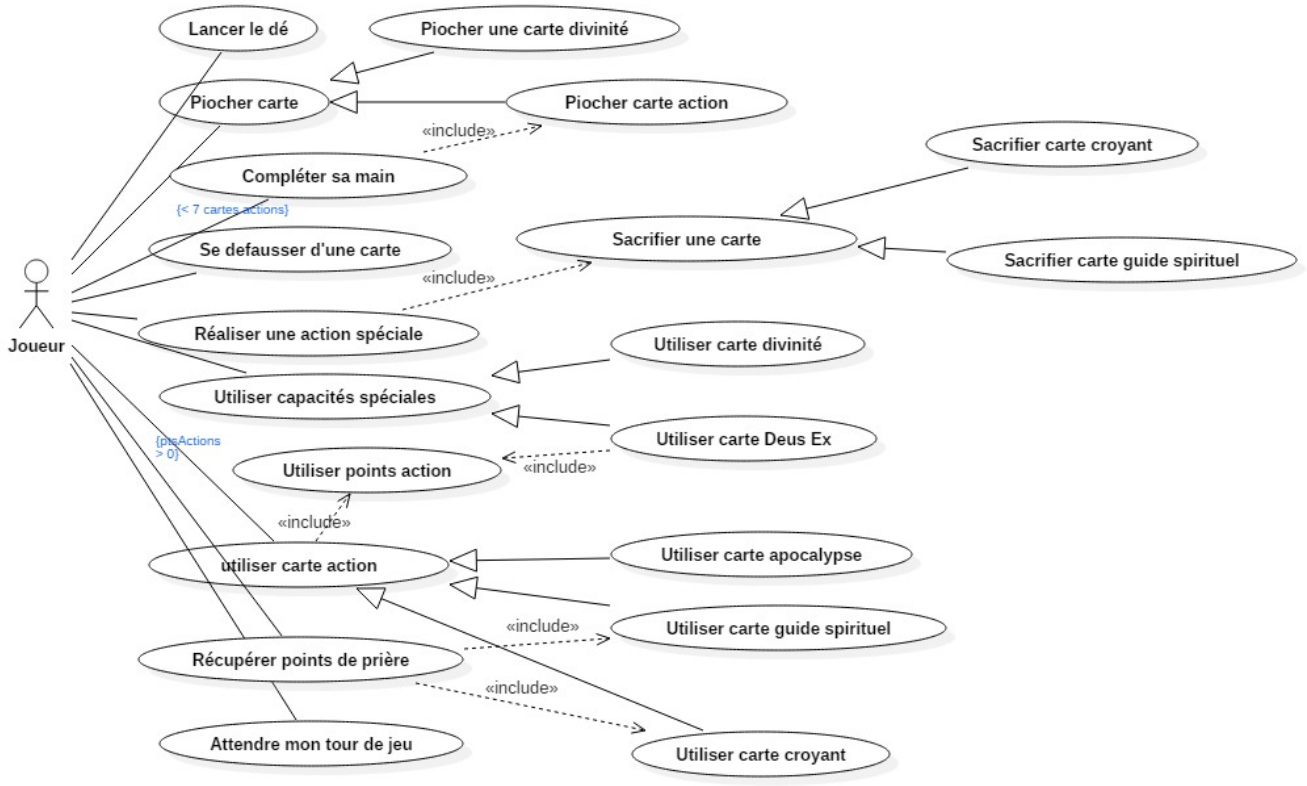
Rayane TALEB
Karim HAMDI

LO02 08/11/16

Introduction :

L’objectif de ce projet est de développer la version électronique du jeu de cartes pandocréon Diviane.
Contrairement à une partie physique de Pandocréon Divinae, la partie se jouera entre un joueur humain et un nombre variables d’intelligences artificiels géré par le programme.
Chaque joueur va incarner une « divinité » durant la partie et le but du jeu sera de récupérer un maximum de « prière » venant des « croyants » pour prendre la place du « haut dieu ».
Pour récupérer ces précieuses « prières », les joueurs devront utiliser des « guides spirituelles ».
Les cartes « Deus Ex » sont des cartes dont les effets peuvent bouleverser le cours d’une partie. Les cartes « Apocalypse » permettent d’éliminer des adversaires et de gagner la partie s’il y a au plus 3 adversaires.

Diagramme cas d’utilisation



Pandocreon divinae est un jeu de société qui peut se jouer à plusieurs personnes.
Chaque joueur a la possibilité d’effectuer une ou plusieurs actions lors de son tour de jeu.

Nous allons vous présenter les différents cas d'utilisation qu'aura un joueur lorsqu'il jouera à Pandocreon divinae.

1. Lancer le dé

Au début de chaque tour de jeu le dé cosmologie sera lancé afin de connaître la répartition des points d'action pour chaque joueur.

2. Piocher une carte

Chaque joueur aura la possibilité de piocher une carte au début du jeu ou au début du tour en piochant une carte divinité et/ou des cartes d'actions.

3. Compléter sa main

Chaque joueur doit posséder dans ses mains 7 cartes d'actions, il doit donc piocher une ou plusieurs cartes d'actions afin de compléter sa main, à condition d'avoir moins de 7 cartes d'action dans les mains

4. Se défausser d'une carte

Un joueur aura la possibilité lors de son tour de jeu, de se défausser d'une carte qui ne l'intéresse pas afin de pouvoir piocher une autre carte.

5. Sacrifier une carte

Chaque joueur aura en sa possession des cartes "croyants" et des cartes "guide spirituel" au cours de la partie, qu'ils pourront sacrifier.

6. Réaliser une action spéciale

Chaque joueur pourra réaliser des actions spéciales en fonction de leur carte, en sacrifiant une carte.

7. Utiliser capacités spéciales

Pour utiliser une ou plusieurs capacités spéciales, chaque joueur devra avoir dans ses mains, une ou plusieurs cartes "deus ex" à utiliser.

8. Utiliser carte action

Tout au long du jeu, chaque joueur aura dans sa main la possibilité d'utiliser une carte action qui sera de type "croyant","apocalypse", "deus ex", "guide spirituel".

9. Utiliser points action

Chaque joueur qui ayant au moins un point d'action après qu'un des joueurs ait lancé le dé, pourra les utiliser en jouant une ou plusieurs cartes.

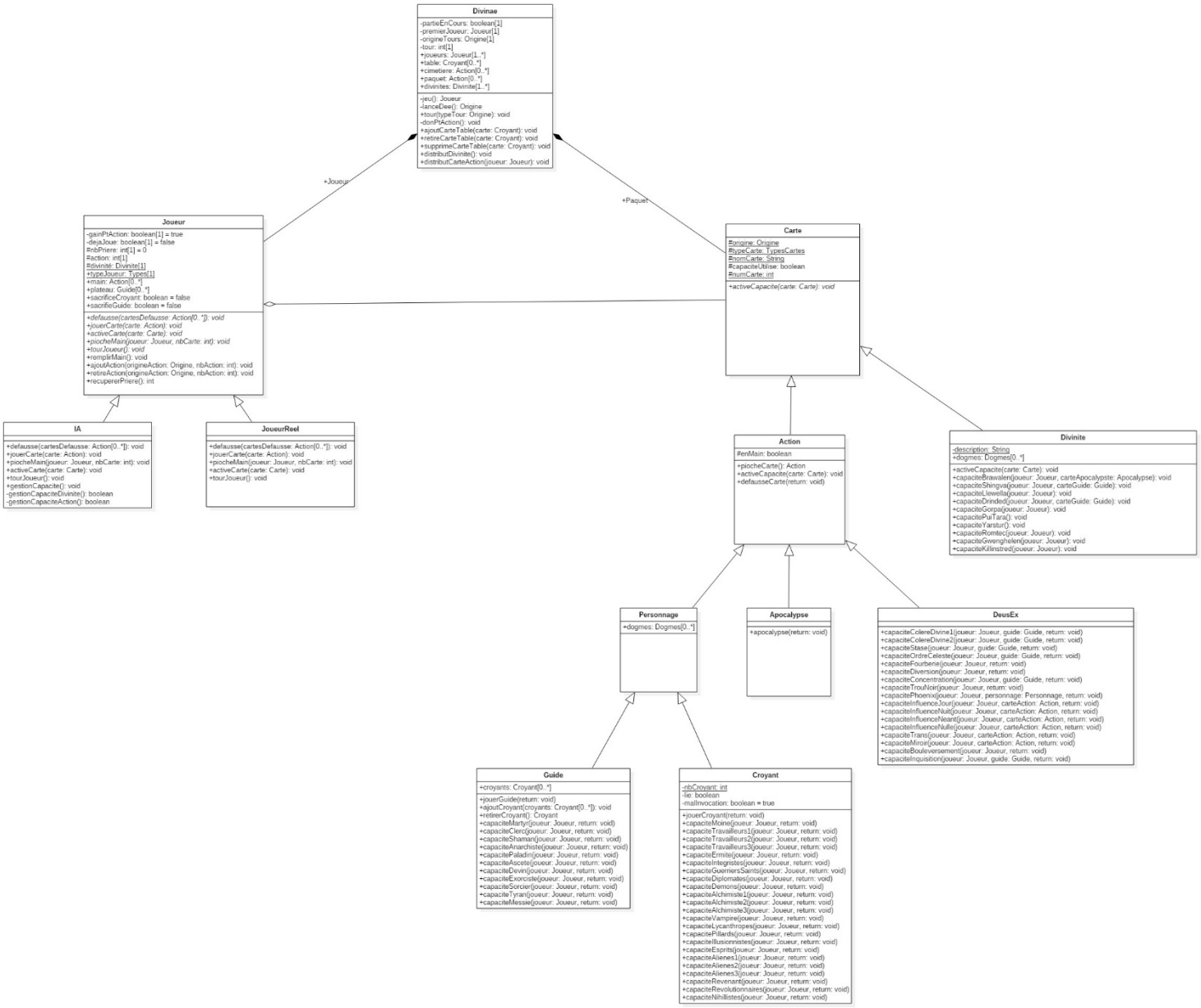
10. Récupérer points de prière

En utilisant une carte “guide spirituel” qui se situe dans sa main, avec une carte “croyant” qui se situe au milieu du jeu, le joueur pourra récupérer des points de prière du nombre indiqué sur la carte “croyant”.

11. Attendre mon tour de jeu

Durant le jeu, chaque joueur seront inactifs en attendant leur tour.

Diagramme de Classe :



On a divisé le projet en 11 classes différentes reliées entre elles.

La classe de base, celle qui va contrôler tout le jeu **Divinae**. C'est la classe qui intègrera la boucle principale du jeu. Elle va gérer les tours de jeu, vérifier si la partie est terminée et déterminer qui est le gagnant. Ses attributs sont publics pour que toutes les autres classes puisse y accéder. Les attributs sont : une liste de *joueurs*, le *paquet* de carte d'action là où les joueurs vont piocher à leur tour, le *cimetière* pour recueillir les cartes défaussées, la *table* de jeu où il y aura toutes les cartes de Croyants, les cartes *divinités* que l'on va distribuer aléatoirement une carte à chaque joueur.

La classe **Divinae** a également des attributs privés qui seront accessibles uniquement depuis cette classe, dont un booléen pour savoir si la *partieEnCours* et définir le gagnant du jeu. On garde un lien sur le premier joueur du tour, de cette manière on peut repérer la fin du tour quand on retombe sur lui après un tour complet et on peut définir le premier joueur du prochain tour. On garde *l'origine du tour* qui va définir comment les points d'action vont être distribué et a qui, ainsi que le numéro du *tour* actuel.

Les méthodes de la classe **Divinae** sont : jeu qui gère la boucle principale; lanceDee qui choisit aléatoirement une Origine entre jour, nuit et néant; tour qui gère la boucle du tour actuelle; donPtAction qui distribue les points d'action aux joueurs en fonction de l'origine du tour, ajoutCarteTable, retireCarteTable et supprimeCarteTable qui gèrent les cartes du centre de la table, distribueDivinite qui distribue les divinités à chaque joueur et enfin distribueCarteAction qui va distribuer au début 7 cartes d'action du paquet à chaque joueur.

La classe **Carte** est une classe abstraite car les cartes du jeu partagent beaucoup d'attributs. Comme toutes les cartes ont une *origine*, le fait de ne pas avoir d'origine sera implémenter comme une origine spéciale. Chaque carte a un *typeCarte* (divinité, deus ex, guide spirituel, apocalypse ou croyant), un *nomCarte*, un *numCarte* qui nous permettra de les différencier et un booléen pour savoir si la *capacite Utilisé* pour éviter de réutiliser. Cette classe a une méthode abstraite activeCapacite qui sera implémenter dans les classes filles.

Les classes **Action** et **Divinité** héritent de **Carte**. La classe **Divinite** représente les cartes divinité qui seront joués par les joueurs, on rajoute aux attributs de la classe mère **Carte** un attribut *description* qui décrit la divinité du jeu ainsi qu'un attribut *dogmes* qui liste les dogmes vénérés par la divinité. Cette classe implémente la méthode définie dans activeCapacite qui activera la capacité spéciale de la divinité ce qui est différent que d'activer une capacité spéciale d'une carte action dans le sens où par exemple on ne défausse pas la divinité une fois la capacité utilisée. Il y a aussi toute les méthodes de capacité des différentes divinités car elles ont tous une action différente.

La classe **Action** représente les cartes actions qui seront représentés par les classes filles **Croyant**, **Guide Spirituel**, **Apocalypse** et **DeusEx.** Dans cette classe on va ajouter aux attributs de la classe **Carte** un booléen pour savoir si la carte est *enMain.* On ajoutera également les méthodes piocheCarte et defausseCarte qui vont interagir avec le paquet de carte d'action du jeu ainsi que l'implémentation de activeCapacite pour les capacités des cartes Action.

Les classes **DeusEx**, **Apocalypse** et **Personnage** héritent directement de la classe **Action**. Les classes **DeusEx** ainsi que la classe **Apocalypse** n'ont aucun nouvel attribut. Ils implémentent juste les méthodes liées à leurs différentes cartes.

La classe **Personnage** rassemble les classes **Guide Spirituel** et **Croyant** qui se ressemblent dans le sens où elles sont des cartes d'action qui interagissent sur la table de jeu. Cette classe rajoute l'attribut *dogmes* qui liste tous les dogmes que les « cartes personnages » vénèrent.

Notre classe **Croyant** a pour attributs supplémentaires le *nbCroyant* qu'elle représente ; un booléen pour savoir si la carte est *lie* à un guide, cet attribut nous permettra de connaitre la position du croyant (au centre de la table ou relié à un guide) ainsi que l'attribut *malInvocation* pour savoir si la carte vient d'être jouée, on évitera ainsi de transgresser la règle qui stipule que l'on ne peut pas mettre un croyant et le relier instantanément à un guide. Elle a comme méthode jouerCroyant qui permettra de mettre en place les actions pour jouer un croyant et des méthodes d'activation spécifique à un croyant spécial.

Enfin la classe **Guide,** qui est la dernière classe qui hérite de **Carte**. Elle a la liste de ses *croyants* en attribut, en plus de tous les attributs dont elle hérite. Comme pour la classe **Croyant,** la classe **Guide** implémente les méthode jouerGuide, ajoutCroyant et retirerCroyant qui effectuent des opérations sur la liste des croyants d'un guide, elle implémente aussi toutes les méthodes pour les capacités spécifique à un guide spécial.

Pour finir ils nous restent la classe **Joueur** et ses classes filles **JoueurReel** et **IA. Joueur** est une classe abstraite. Elle est composée des attributs *gainPtAction* qui nous indique si le joueur peut gagner des points d'action durant le tour, *dejaJoue* pour savoir si on peut agir sur un joueur avec certaines capacités de carte d'action, l'attribut *nbPriere* qui indique le nombre de prière du joueur, *action* pour savoir combien de point d'action le joueur dispose, chaque joueur se voit attribuer une *divinité*, on doit savoir le *typeJoueur* pour savoir si c'est une IA ou un vrai joueur, tous les joueurs ont une *main* de carte d'action, ils ont également un *plateau* qui représente les cartes devant eux et deux booléens *sacrificeCroyant* et *sacrificeGuide* qui vont nous permettre de savoir s'ils ont le droit de sacrifier un guide ou un croyant durant ce tour.

Joueur possède aussi deux types de méthode, les méthodes qui seront les même pour **IA** et **JoueurReel,** remplirMain qui vas compléter la *main* des joueurs peu importe leurs type, ajoutAction et retireAction qui vont interagir avec *action* et recupererPriere qui calcule le nombre de prière de chaque guide pour mettre à jour *nbPriere.* Et les méthodes abstraites qui seront implémenté dans les classes filles car elles auront des comportements différents en fonction du type de joueurs. Les méthodes abstraites sont défausse qui vas mettre des cartes actions dans le cimetière, jouerCarte qui permet de jouer la carte en fonction de son type, activeCarte qui vas définir la façon d'activer la capacité de carte, piocheMain qui permet de gérer quand on doit piocher une carte dans la main d'un autre joueur ou quand on doit piocher dans notre main et pour finir avec les méthodes abstraites de la classe **Joueur,** la méthode tourJoueur qui gère tout le tours d'un joueur avec les appels aux bonnes méthodes

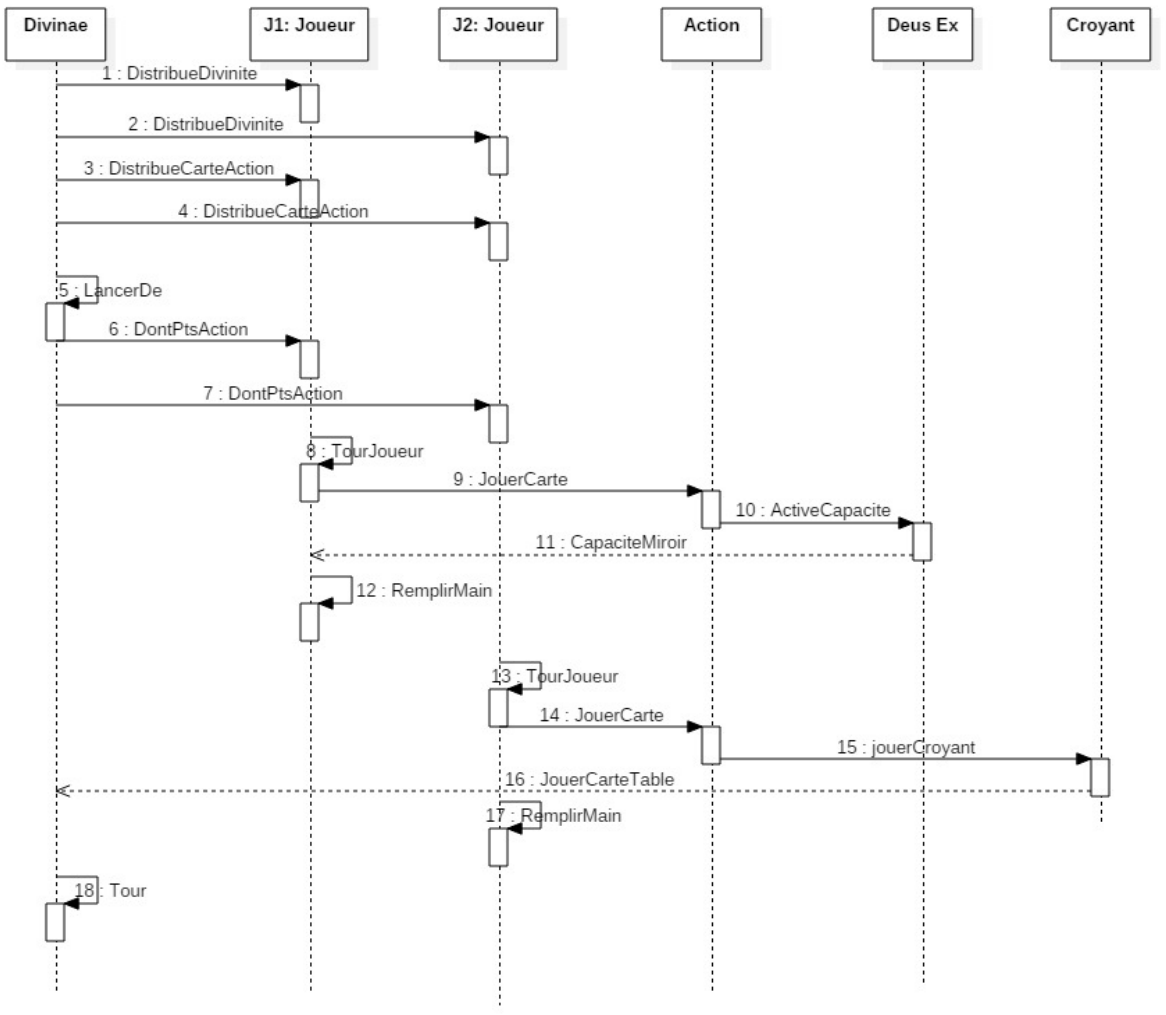
au bon moment. Tous ces attributs ainsi que ses méthodes abstraites sont récupérés et les méthodes sont implémentés par les classes **JoueurReel** et **IA.**

Pour **JoueurReel** on va laisser le choix des cartes à défausser, à activer, à piocher ou encore à jouer au Joueur directement.

Alors que pour l'**IA,** nous allons implémenter une stratégie de jeu. Pour defausse, je ne défausse pas les cartes de mêmes origines que moi, ni celle qui n'ont pas origine. Si je n'ai aucune carte de même origine que moi ou d'aucune origine alors je conserve la meilleure carte d'origine néant. Je jouerais toutes les cartes que je peux jouer en privilégiant les cartes de type croyant avec le même dogme que moi ainsi que les cartes croyants avec un grand nombre de croyants.

On rajoutera également à toutes les méthodes abstraites que l'on aura implémentées des méthodes de gestionCapacite pour savoir le moment auquel il faut activer la capacité en fonction des évènements durant la partie. On peut activer la capacité de la divinité à tout moment dans la partie. Pour cela, on doit trouver à quel moment le faire et contre qui le faire en fonction de chaque divinité. Pour cela on implémentera gestionCapaciteDivinite ainsi que gestionCapaciteAction qui gèreront et quel moment et contre qui utiliser les capacités des carte actions. Utiliser une carte apocalypse seulement quand on a un nombre de prière supérieur à celui des autres joueurs. Il faut à tout prix empêcher un joueur d'utilisé une carte apocalypse si l'utilisation de cette carte m'élimine ou fait gagner ce joueur. La stratégie sera simple, faire en sorte d'avoir le plus grand nombre de prière possible et empêcher les autre de gagner.

Diagramme de séquence



Comme nous venons de le voir Pandocreon Divinae est un jeu de société qui permet à plusieurs joueurs d'effectuer plusieurs cas d'utilisation différents. Nous allons maintenant vous montrer un exemple de déroulement d'un tour de jeu, en utilisant 2 joueurs. Le diagramme de séquence à la différence du diagramme de cas d'utilisation, permet d'ajouter une dimension temporelle et toutes les actions entre les différents objets ont été prélevés à partir de notre diagramme de classe.

1. Mise en situation :

Cette partie permet de commencer le jeu, à partir de l'objet "divinae". Cet objet va d'abord distribuer une carte divinité à chaque joueur présent, ensuite il va distribuer 7 cartes actions à chaque joueur. Pour cela "Divinae" utilise les méthodes : *distribueDivinite* et : *distribuCarteAction*. Par la suite, "Divinae" va pouvoir lancer le dé en donnant une valeur aléatoire et en changeant de lanceur à chaque tour. Le résultat du dé va ensuite affecter le tour en attribuant des points d'actions à chaque joueur, pour cela l'objet "Divinae" va utiliser la méthode : *distribuPtsAction.* Et lorsque tous les joueurs auront effectués leur tour "Divinae" va ensuite stocker toutes les informations qu'il y a eu durant ce tour et passer au tour suivant en utilisant la méthode *: tour*

2. Tour de jeu du Joueur 1

Le joueur 1 va utiliser ses points d'actions en utilisant la capacité d'une des cartes "Deus Ex", Le joueur va donc utiliser la capacité de sa carte Deus Ex qu'il a dans les mains ici nous avons donnés comme exemple : *capaciteMiroir,* cela va donc retourner une capacité au joueur. Après avoir joué une carte action le joueur devra remplir sa main en exécutant la méthode : *remplirMain*

3. Tour de jeu du joueur 2

Le joueur 1 va utiliser ses points d'actions en posant une carte "croyant" au centre de la table. Le joueur devra donc utiliser la méthode : *jouerCarte* en utilisant l'objet "Action" qui lui-même va utiliser l'objet "Croyant" en exécutant la méthode : *jouerCroyant.* Le joueur aura donc posé une carte croyant au centre de la table en utilisant l'objet "Divinae". Après avoir joué une carte action, le joueur devra remplir sa main en exécutant la méthode : *remplirMain*

Conclusion :

La partie de la modélisation des contraintes liées aux règles du jeu ne changeront pas. La modélisation virtuelle des cartes du jeu et le déroulement d'un tour de jeu resteront identiques. Cependant toute la partie liée à l'intelligence artificiel sera susceptible d'évoluer au cours du développement, dans le but d'équilibrer la difficulté du jeu pour le joueur humain.