UNIVERSITÉ D'ARTOIS

SUSHI BAR

Auteur: Dubuis Jonathan

Maîtres de stage: Cardon Stéphane
Lagrue Sylvain

29 Mars – 4 Juin 2010

Table des matières

Introduction

Durant cette première année de master informatique il nous est demandé de réaliser un stage en entreprise ou un projet au sein du laboratoire de recherche du CRIL. Trouvant l'idée de concevoir un jeu avec des intelligences artificielles intéressante, j'ai choisi le sujet du Sushi bar.

Depuis les débuts de l'informatique, l'homme cherche à rendre les ordinateurs capables de réaliser des tâches complexes tel que la prise de décisions. Dés lors un nouveau domaine de recherches est né, celui de l'intelligence artificielle. De nombreuses recherches pour améliorer les connaissances dans le domaine des intelligences artificielles utilisent des jeux pour progresser. Cependant les jeux les plus utilisés sont ceux dit déterministes c'est à dire dont toutes les possibilités sont prévisibles comme aux échecs.

Certains chercheurs ont eu l'idée de tester les limites de la machine dans des domaines faisant intervenir le hasard. C'est dans ce contexte que des membres du CRIL ont conçu une adaptation de Pickomino un jeu de plateau où le hasard a une grande importance pour pouvoir tester et améliorer le fruit de leurs recherches.

Le Sushi bar reprend cette idée d'adapter un jeu de plateau sous forme de jeu vidéo pour confronter des intelligences artificielles à des joueurs humains. Ainsi qu'un mode permettant de joueur avec un vrai plateau de jeu.

Tout comme son prédécesseur, le Sushi bar doit permettre à un joueur de se mesurer à divers IAs mais a aussi l'ambition de faire combattre les IAs entre elles pour déterminer la plus efficace.

L'application réalisée en java est basée sur le modèle MVC pour une découpe claire.

Chapitre A: Le CRIL

1 Présentation

Le Centre de Recherche en Informatique de Lens est un laboratoire de recherche attaché à l'Université d'Artois et associé au CNRS. Il rassemble plus de cinquante chercheurs, enseignants-chercheurs, doctorants et personnel administratif et technique.

2 Les recherches

Les recherches menées au sein du CRIL concernent les intelligences artificielles. Ces derniers sont des programmes capables de prendre des décisions, pour satisfaire au mieux un problème donné selon les informations qu'on lui aura données.

Pour résoudre les problèmes les pistes de recherche abordées par le laboratoire sont : l'algorithmique pour l'inférence et la prise de décisions et les domaines du traitement des informations imparfaites, dynamiques, contextuels et multi-sources.

Dans le cadre de ces recherches de nombreux logiciels d'aide à la décisions ont étés développés allant du planificateur temporel, à la bibliothèque SAT4J pour résoudre les problèmes SAT utilisé dans le projet Alloy.

Chapitre B: Présentation du projet

1 Sushi bar le jeu

1.1 Présentation du jeu

Sushi-bar est un jeu de plateau pour 2 à 5 joueurs créé par Reiner Knizia et édité par Gigamic.

Le jeu consiste à lancer les dés dont le résultat déterminera quel sushis ou quelle arête pourra être pris dans le bar ou être volé à un adversaire. A la fin de la partie on établie le score de chaque joueur en fonction des arêtes et des sushis qu'il a réussi a rassemblé. Le joueur ayant le plus grand score gagne la partie.

Illustration 1: Le Sushi bar

1.2 Organisation du plateau de jeu

Le plateau de jeu est composé de deux rangés de 12 portions chacune. La première contenant les sushis et la seconde les arêtes. Chaque joueur possède face à lui une pile de sushis et une d'arête initialement vide où le joueur empilera les portions qu'il gagnera.

Type de sushi						
Nombre	2	2	2	2	2	2

Tableau 1: Liste des sushis

Type d'arête				
Nombre	5	4	2	1

Tableau 2: Liste des arêtes

Chaque portion possède un nombre de points. Ainsi les sushis rapportent des points alors que les arêtes en font perdre.

La sélection de ces portions se fait par des lancés de 5 dés à 6 faces :

Face	 Sushi	 arête	 Baguettes bleues	 Baguettes rouges
Nombre par dé	2	2	1	1

Tableau 3: Faces des dés

Les faces sushi indiquent la position du sushi prenable dans la barre des sushi, l'arête indique quelle arête est prenable dans la barre des arêtes. Tandis que les baguettes permettent de voler une portion sushi pour les bleues et une portion d'arête pour les rouges dans la pile d'un adversaire.

1.3 Règles

1.3.1 Préparation

Les joueurs commencent par mélanger les sushis et les arêtes pour ensuite former au milieu de la table deux rangés de portions : l'une contenant les sushis et l'autre les arêtes.

Ensuite les participent jouent chacun leur tour dans le sens horaire.

1.3.2 Phase de jeu

Lors de son tour, le joueur dispose au maximum de 3 lancés. Après son lancé les dés indiquent quelles actions sont réalisables.

a. Prise de portion dans le bar

Le nombre de dés dont la face visible indique un sushi détermine la position du sushi prenable dans le bar. De même le nombre de dés indiquant une arête révèle l'arête prenable dans le bar. Le joueur peut bien évidemment prendre cette portion que si elle existe.

Par exemple :

Lors de son premier lancer Alphonse obtient le lance suivant :

Illustration 2: Résultat du premier lancer d'Alphonse

Les portions du bar sont les suivantes :

Illustration 3: État du bar lors du premier lancer d'Alphonse

Alphonse peut seulement prendre la seconde arête qui a pour valeur -3. En effet comme il possède 3 dés sushis et 2 dés arêtes il a le choix entre prendre le 3éme sushis ou la seconde arête du bar. Hors le 3ème sushis n'existe plus, il ne peut donc pas prendre l'arête -3.

b. Relancer

Ne voulant pas prendre l'arête -3 et étant au premier tour, Alphonse choisit de relancer.

Pour se faire il doit choisir au moins un dé à conserver et en laisser au moins un de libre pour le lancer au tour suivant. Bien que le dé conservé ne sera pas lancé au tour suivant, la face qu'il indique sera tout de même comptabilisée dans le résultat du lancer.

Alphonse décide donc de conserver un dé sushi et de relancer les autres dés.

Illustration 4: Résultats du second lancer d'Alphonse

Avec ce lancer Alphonse peut prendre le second sushi ou la première arête. Mais il remarque qu'il dispose de 2 baguettes bleus et comme il lui reste encore un tour et qu'il a encore 4 dés de libres. Il choisit de conserver les baguettes et de relancer dans l'espoir d'avoir assez de baguettes pour pouvoir voler un sushi a l'un de ses adversaire.

c. Action par défaut

Pour son 3ème lancer Alphonse obtient les résultats suivants :

Illustration 5: Dernier lancer d' Alphonse

N'ayant pas de chance Alphonse n'a pas réussi a obtenir le nombre de dés minimum pour voler (il faut 3 dés baguettes de la même couleur). De plus, le sushi indiqué par les dés n'existe plus dans le bar et il est dans l'impossibilité de relancer (les trois lancés sont réalisés).

Dans ce cas Alphonse est dans l'obligation de prendre la pire arête ou dans le cas où il n'y aurait plus d'arête dans le bar le plus petit sushi. Dans notre configuration le joueur doit prendre l'arête -3 et la placer au sommet de sa pile d'arêtes.

NB : Lorsque le joueur prend une arête ou un sushi il le pose au sommet de sa pile correspondante. Seul la portion se trouvant au sommet de la pile est visible. Ni le propriétaire

de la pile ni un autre joueur n'est autorisé a regarder les portions qui sont dans la pile.

d. Voler une portion

Comme Alphonse a terminé son tour, il donne la main a son voisin. Bernard lance les dés et obtient le lancé suivant :

Illustration 6: Premier lancer de Bernard

Les dés lui indiquent qu'il peut prendre la 3ème arête dans le bar hors cette dernière n'existe plus. De plus il lui faudrait 3 baguettes bleus pour prendre le sushis se trouvant au sommet de la pile de l'un de ses adversaire et au moins 4 pour prendre n'importe quel sushis chez ses adversaire.

NB : Les baguettes rouges se comportent de la même manière pour les arêtes.

Bernard choisit de conserver les baguettes bleues puis de relancer.

Illustration 7: Second lancé de Bernard

Maintenant Bernard possède assez de dés pour voler la portion de son choix. Toutefois il n'est pas autorisé à regarder le contenu des piles.

Il décide de prendre le 3ème sushi d'Alphonse qu'il ajoute au sommet de sa pile de sushis.

NB : Lorsqu'aucun joueur ne possède pas de portion il est alors impossible de voler et dans le cas où aucune autre action n'est réalisable c'est l'action par défaut qui est réalisée

1.3.3 Fin de partie

Lorsqu'il ne reste plus aucune portion dans le bar la partie est terminée. On procède alors au calcul des points et le joueur possédant le plus de points est nommé vainqueur.

Pour calculer les points de chaque joueur on commence par retirer tout les sushis qui dépassent de la pile des arêtes jusqu'à ce que le nombre de sushis soit inférieur ou égal au

nombre d'arêtes possédées par le dit joueur. Ensuite on ajoute au score les points rapportés par chaque sushis auquel on va retrancher les points des arêtes.

Par exemple à la fin de la partie Alphonse possède les portions suivantes:

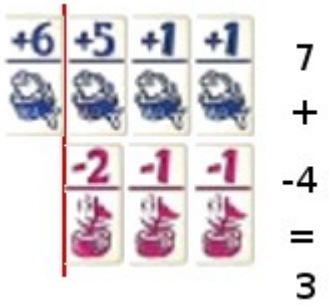


Illustration 8: Score d'Alphonse

On retire donc le sushis ayant pour valeur 6 et on fait les totaux.

De la même manière pour J2 :

Illustration 9: Score de Bernard

Bernard remporte donc la partie.

2 Cahier des charges

2.1 Présentation

Le sushi bar doit être l'adaptation du jeux de plateau du même nom pour développer et tester des intelligences artificielles qui se confronteront au problème non déterministe du jeu de Sushi bar.

2.2 Description de la demande

L'application doit permettre de confronter de 2 à 5 joueurs humains ou intelligences artificielles.

L'application est conçue pour trois modes d'utilisation :

→ Humain vs IA : qui oppose un joueur humain à des IAs qu'il aura choisies à la création de la partie.

→ Opérateur : qui permet de confronter un ou plusieurs joueurs à une IA tout en jouant sur un plateau réel. L'IA étant informée et s'exprimant à travers l'interversion d'un opérateur.

→ Arène : plusieurs IAs s'affrontent sur un nombre de tour donné.

De plus la création ou l'ajout de nouvelles IAs devra se faire facilement.

2.3 Contraintes

Cette adaptation doit reprendre les mêmes règles que la version plateau du jeu.

L'application doit être développée en java, en suivant le modèle MVC.

La version finale devra être livrée au bout des 10 semaines du stage.

2.4 Planning

N° semaine	Tâches planifiées	Tâches réalisées
1	Prise de connaissance du projet, et étude.	Étude des règles, cahier des charges, cas d'utilisation, états transition séquences.
2	DCU, CET, DCL	Rectification des diagrammes de séquences suite a la réunion. Modélisation partie modèle et IA et interfaces des vues et contrôleur.
		Modélisation du contrôleur et de la vue. Création de la vue de base de la partie.
3		Création de la vue de base de la partie. Conception du modèle et testes fonctionnels pour le modèle.
4		Rectifications sur la conception du contrôleur. Création du contrôleur.
5	V1 du projet avec un joueur et une IA.	Création du contrôleur. Création de la première IA.
6	Mode opérateur	Ajout de la désélection d'une portion après sa prise. Mode opérateur (non terminé).
7	Mode Arène.	Modification de l'organisation du contrôleur et de la vue. Gestion de l'historique pour le mode opérateur et ajout de l'édition du bar en début de mode opérateur (Bug rencontré)
8	Préversion du rapport	Préversion du rapport. Écriture de nouveau tests pour les contrôleur. Rectification d'un bug sur le retour en arrière.
9		Révision du rapport. Amélioration de l'interface graphique.
10		Révision du rapport. Amélioration de l'interface graphique.

Tableau 4: Planning

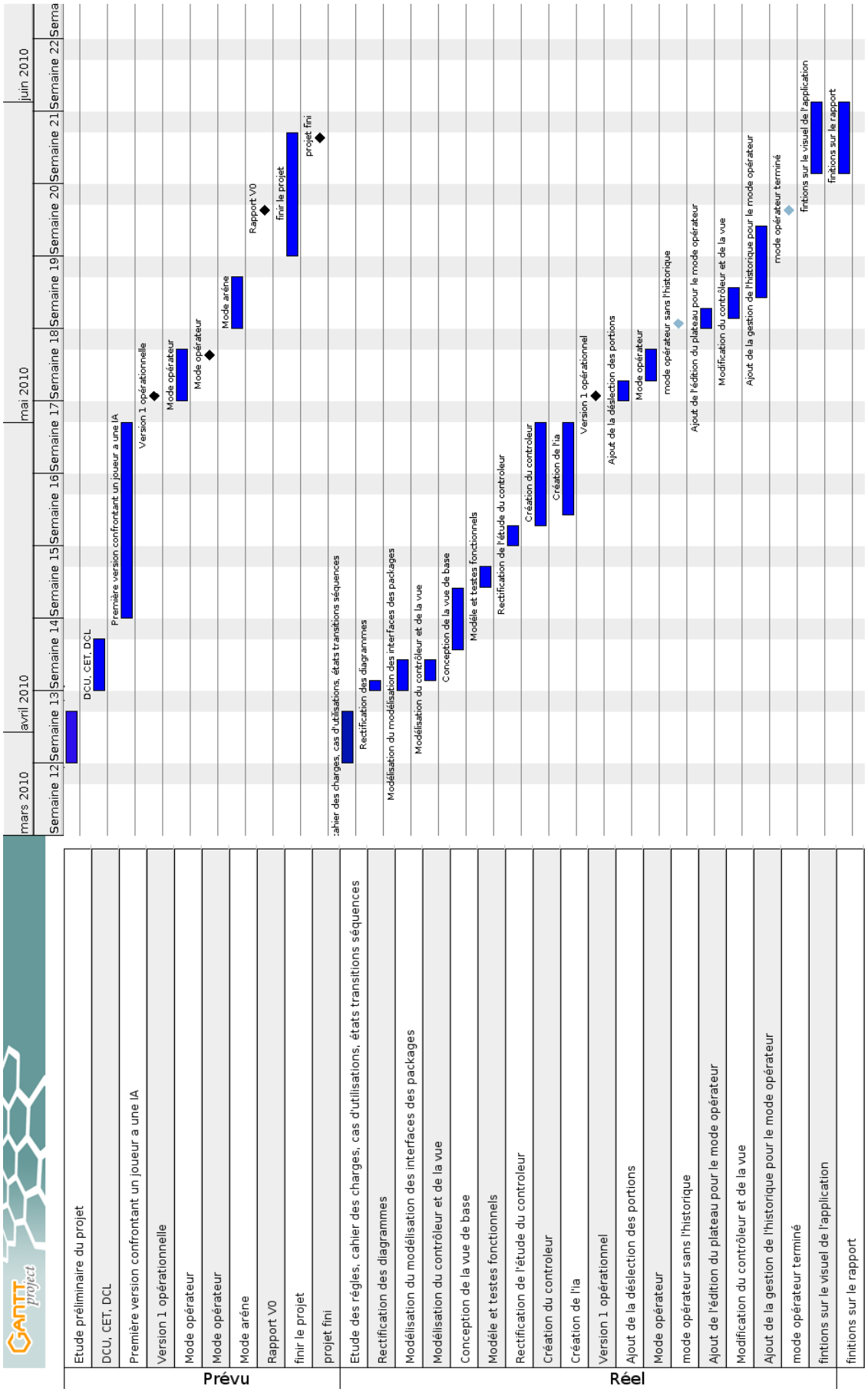


Illustration 10: Diagramme de gantt du déroulement du projet

Chapitre C: L'application

1 Analyse

1.1 Acteurs et cas d'utilisations

Les divers modes de fonctionnement et besoins de l'application permettent d'établir le diagramme des cas d'utilisations ci-dessous.

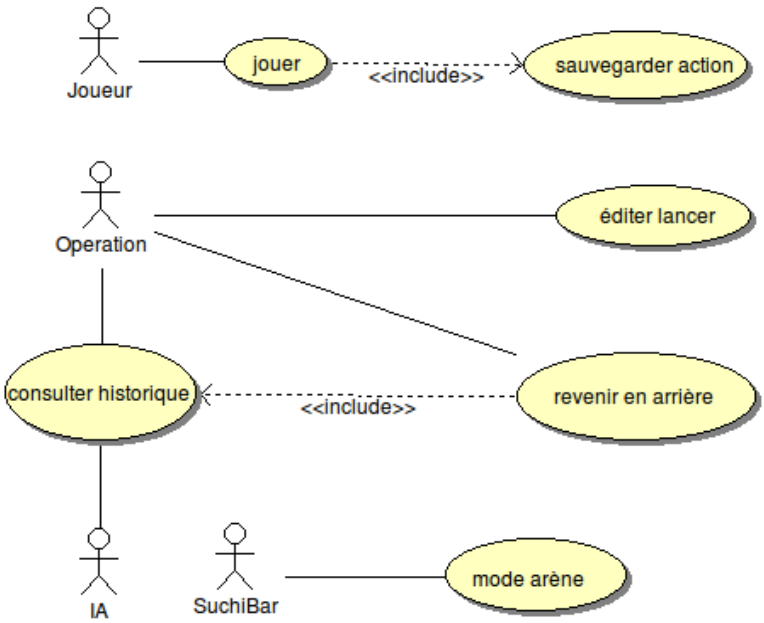


Illustration 11: Diagramme des cas d'utilisations de l'application

Les trois modes d'opérations suggèrent l'existence de 4 types d'utilisateurs qui sont :

→ joueur humain : il est le principal acteur de l'application, comme son nom l'indique, son rôle dans l'application est de joueur. Au cas d'utilisation jouer s'ajoute la sauvegarde des actions qui permettra de revenir sur un choix.

→ opérateur : il est l'acteur principal de l'application en mode opérateur. C'est lui qui fera le lien entre l'application et le plateau réel du jeu. Il peut ainsi éditer les lancés de dés mais aussi consulter l'historique et revenir en arrière en cas mauvaise manipulation.

→ IA (Intelligence Artificielle) reprend le rôle du joueur mais pourra aussi consulter l'historique en cas de besoin pour établir une stratégie.

→ sushi bar : il gère le mode arène en faisant intervenir plusieurs IAs au cours d'un nombre de parties décidées par l'utilisateur.

1.2 Déroulement général de la partie:

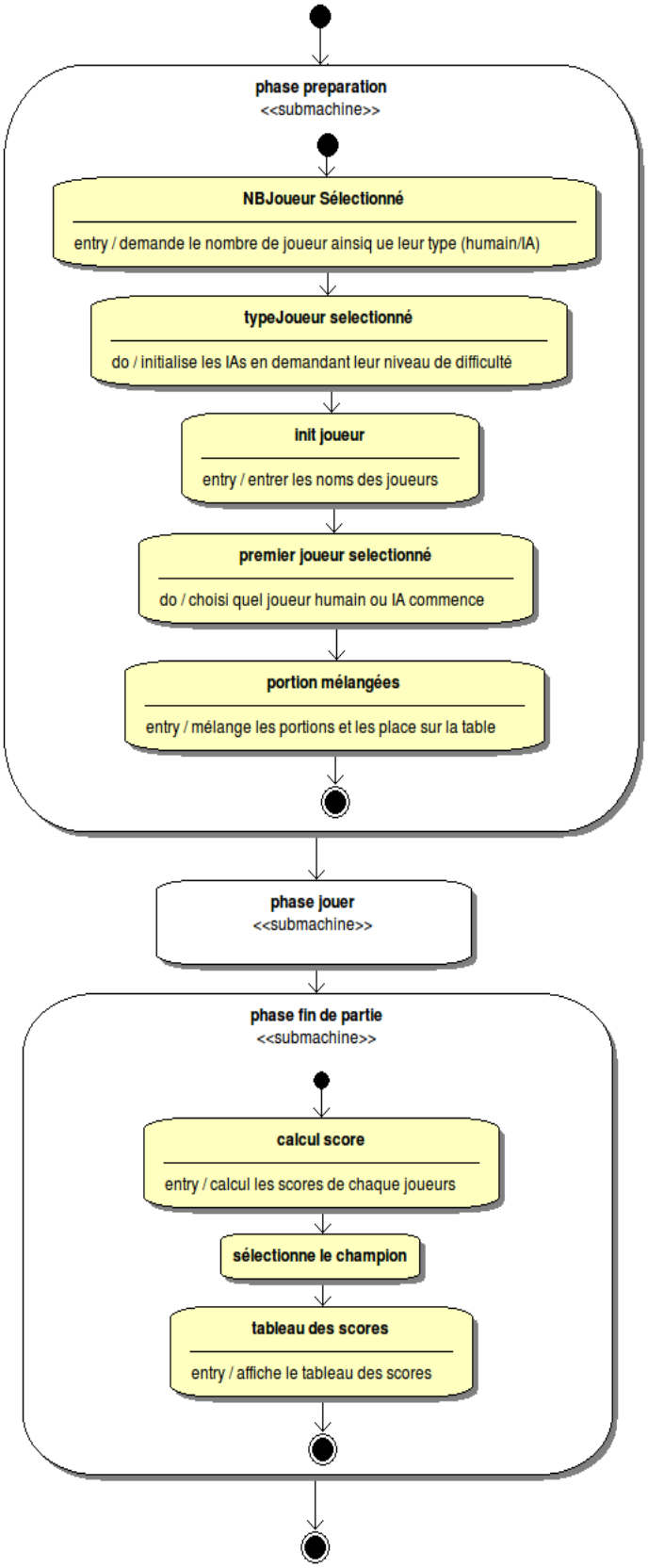


Illustration 12: Diagramme d'états transitions d'une partie

Quelque soit le mode de jeu choisi une partie se déroule en 3 phases :

→ la phase d'initialisation: on indique le nombre le type des participants et le plateau de jeu est préparé.

→ la phase de jeu .

→ la fin de partie où le score des joueurs est calculé et le vainqueur est nommé.

1.2.1 Préparation

Lors de la préparation on indique le nombre et le type (humain ou IA) des participants. Puis on prépare le plateau en mélangeant la barre de sushis et la barre des arêtes.

1.2.2 Phase de jeu

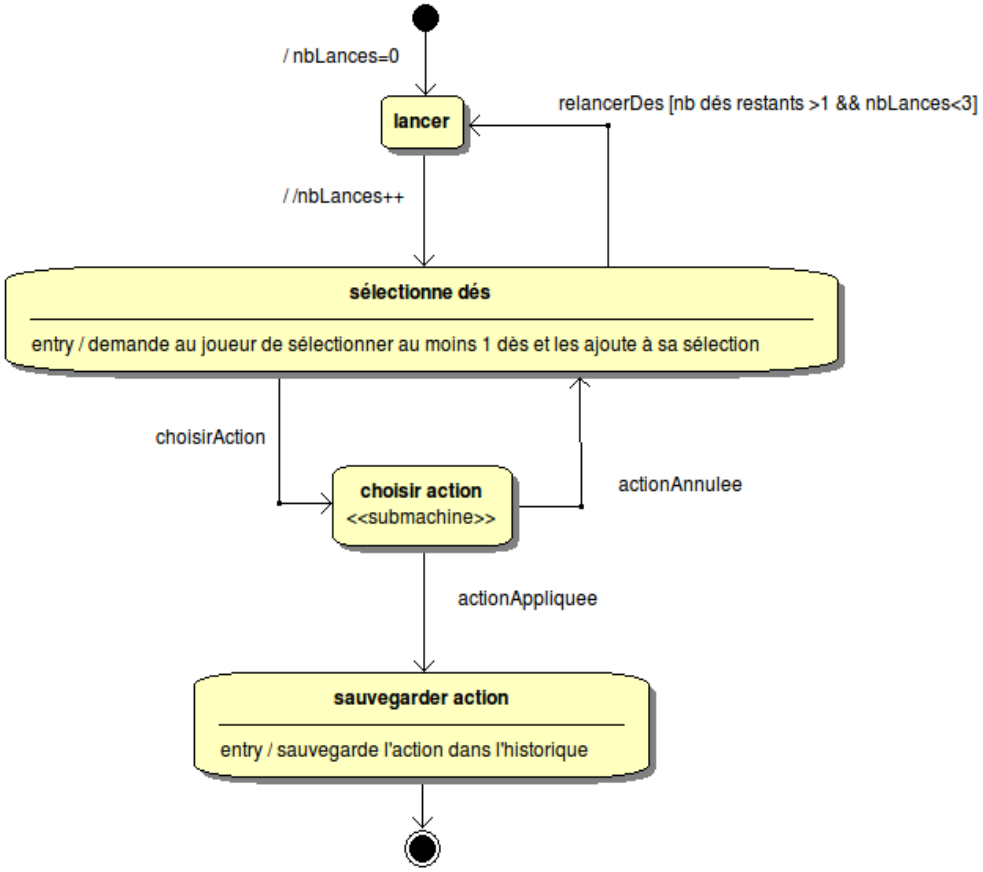


Illustration 13: Diagramme d'états transitions d'un tour

Les joueurs jouent chacun leur tour dans le sens des aiguilles d'une montre.

Le tour d'un joueur est décrit par le cycle suivant :

→ lancer les dés libres

→ choisir au moins un dé à conserver (la face du dé n'a pas d'importance)

→ relancer si aucune action n'a été exécutée, que le nombre limite de lancer n'a pas été atteint et qu'il reste au moins un dé libre.

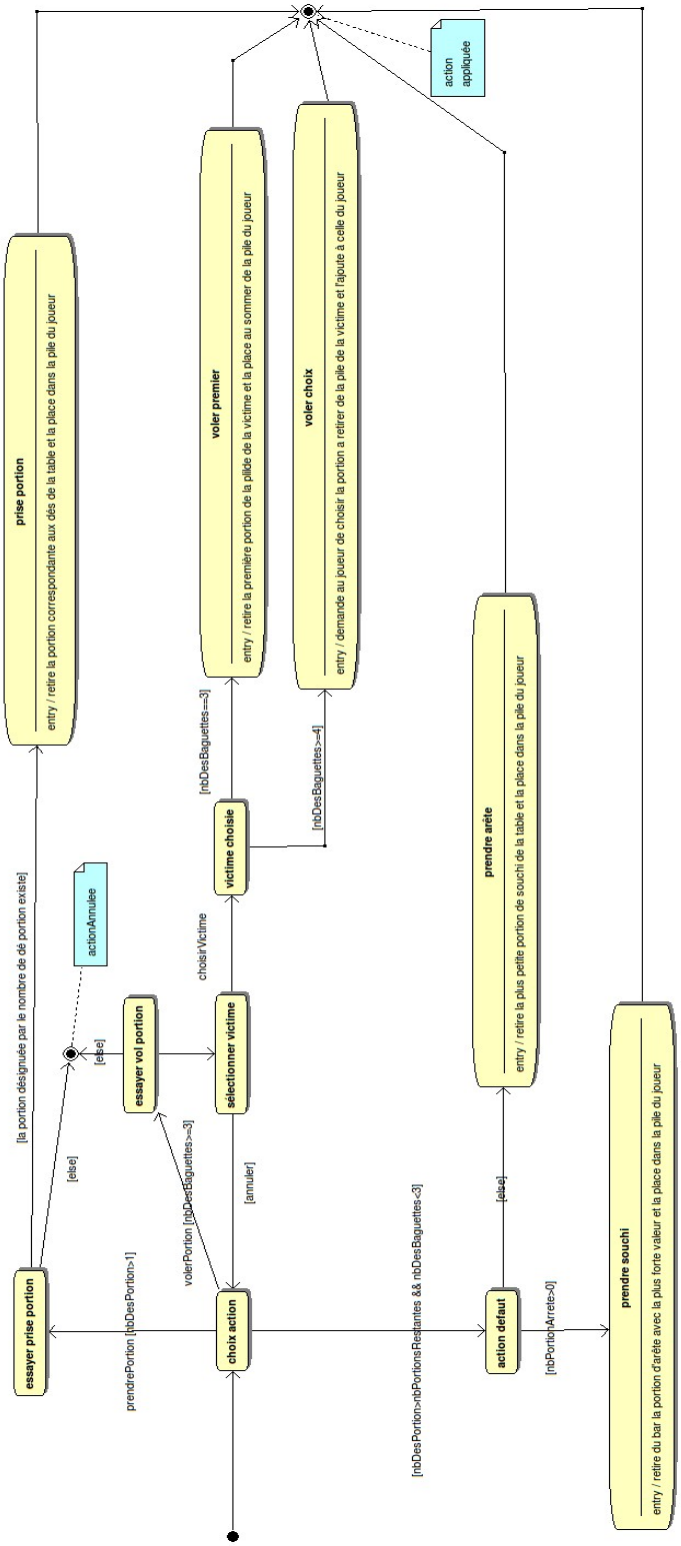


Illustration 14: Diagramme d'états transitions du choix de l'action

Le cycle prend fin lorsque le joueur ne peut plus relancer les dés ou s'il réalise une action.

A chaque cycle les actions réalisables par le joueur sont déterminées par les faces des dés qu'il vient de lancer et celles des dés qu'il a choisit de conserver.

Le nombre de face sushi indique la position du sushi prenable qui se trouve dans la barre en partant de droite, de même les faces arêtes indiquent la position de l'arête prenable dans la barre des arêtes. Une fois la portion choisie elle est transférée du bar vers celle du joueur.

Les baguettes permettent de voler les sushis des autres joueurs pour les bleues et les arêtes pour les rouges. Si le joueur possède 3 baguettes de la même couleur il peut prendre la portion se trouvant au sommet de la pile du joueur de son choix. Si il possède au moins 4 baguettes de la même couleur il peut prendre la portion de son choix dans la pile.

Si le nombre limite de lancer est atteint et que le joueur ne peut réaliser une action (portion désignée par les dés n'existant plus dans le bar ou aucun joueur ne pouvant être volé), le jeu rentrera dans l'état de l'action par défaut qui contraindra le joueur à prendre l'arête avec la plus forte valeur négative ou le plus petit sushi s'il n'y a plus d'arête dans le bar.

Une fois la portion prise, le joueur la place au sommet de sa pile.

NB : Seule la potion qui se trouve en sommet de pile est visible, ni le joueur ni l'un de ses adversaires peut regarder le contenu de la pile.

1.2.3 Fin de la partie

La partie se termine lorsqu'il ne reste plus aucune portion dans le bar. On calcul alors le score de chaque joueur en retirant les sushis dépassant la pile des arêtes et en additionnant les points des portions restantes.

Le joueur ayant le score le plus élevé remporte la partie.

1.3 Organisation du code

Comme il a été présenté dans l'introduction l'application est développée sur la base du patron de conception Modèle Vue Contrôleur pour rendre la maintenance et la reprise du code plus accessible.

Le modèle constitue la base de l'application, il contient toutes les données relative à une partie, il contient les règles et les fonctions nécessaires aux opérations de base (les transferts de portion entre le bar et le joueur par exemple).

La vue est l'intermédiaire entre l'utilisateur et l'application en étant la représentation visuelle du modèle.

Le contrôleur est quant à lui le coordinateur de l'application c'est lui qui va faire le lien entre le modèle et la vue. Vérifiant que telle action est applicable ou non avant de modifier le modèle.

Ce qui se traduit par le diagramme de séquences suivant:

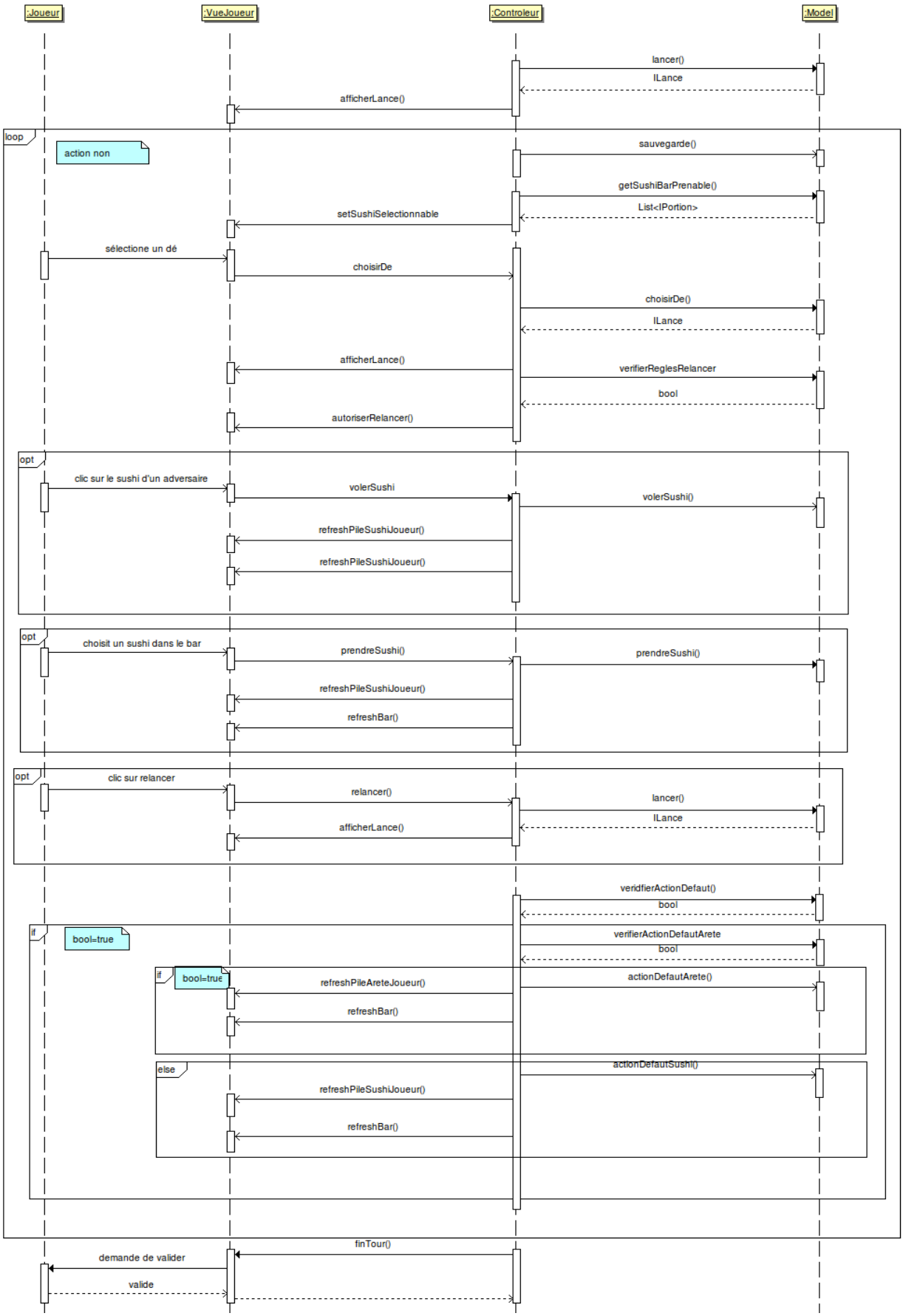


Illustration 15: Diagramme de séquences de la face de jeu

Au début du tour le contrôleur va demander au modèle de lancer les dés puis va afficher le résultat pour ensuite entrer dans la routine de vérification des actions possible tant qu'une action n'a pas été réalisée.

Au début de la routine le contrôleur lance la sauvegarde du modèle. Ensuite le joueur choisi un dé en cliquant sur un dé libre ou qui a été sélectionné. Le contrôleur va alors par le biais de la procédure void choisirDe(IDe) sélectionner le dé ou le dé-sélectionner.

Le contrôleur va ensuite vérifier les règles en interrogeant le modèle, il va ainsi dans un premier temps demander au modèle si la règle pour relancer un dé est satisfaite le contrôleur autorisera la vue de relancer. De la même manière le contrôleur va vérifier si les règles pour prendre ou voler une portion sont satisfaites et dans le cas échéant il rendra les bonnes portions prenables.

Le joueur pourra alors choisir la portion disponible qu'il souhaite ou relancer si cette action lui est autorisée. La vue indiquera au contrôleur quel est l'action choisie et indiquera si nécessaire quelle est la portion concernée.

Une fois l'action appliqué le contrôleur demande a l'utilisateur de terminer via un bouton de la vue.

2 Réalisation

2.1 MVC et IA

L'application s'articule ainsi autour de 4 packages : le modèle, la vue, le controleur et IA.

Le contrôleur, la vue et l'IA consultent le modèle grâce à l'interface IModelLecture. Pour autoriser le joueur ou l'IA à réaliser telle où telle action la vue et l'IA implémentent l'interface IPlayable. Une fois que le joueur ou l'IA choisit de réaliser une action, il en fait part au contrôleur par l'appel de la fonction correspondante à l'action à réaliser se trouvant dans l'interface IDesision.

La mise à jour de la vue se fait avec l'interface IVuePartie. Le contrôleur vérifie si les règles sont bien respecté en interrogeant le model avec l'interface IRegle puis si l'opération est permise, il modifie le modèle avec l'interface IModel et sauvegarde éventuellement les modification avec l'interface ImodelSave.

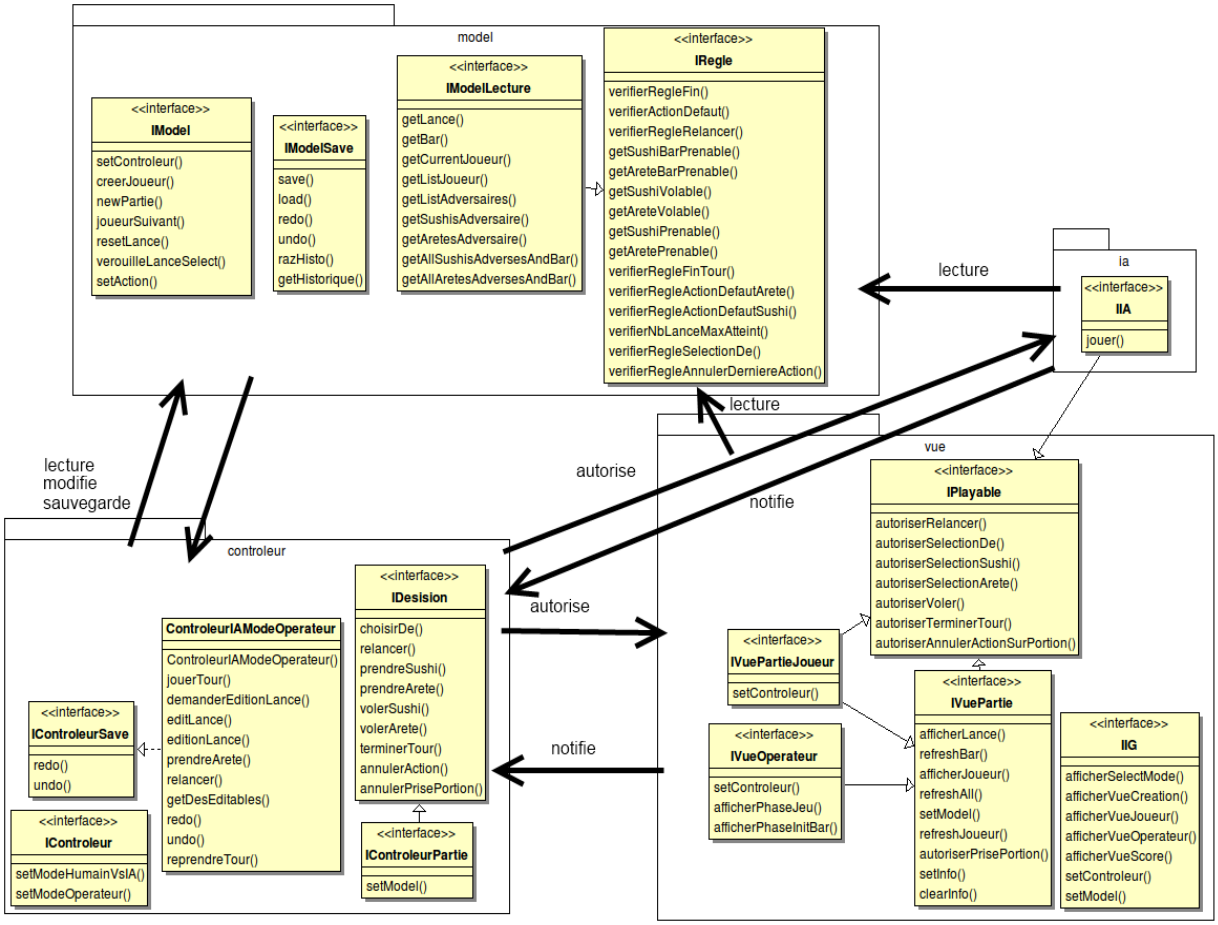


Illustration 16: Diagramme de classes général

2.1.1 Modèle

Pour rappel, le modèle est le fondement même de l'application il est par conséquent indispensable qu'il représenté les divers éléments du plateau sous forme d'objet.

a. Les dés

Le fonctionnement du jeu rend indispensable l'existence d'un objet dé lançable qui possède différentes faces. Les dés sont rassemblés dans un lancé qui lui aussi est lançable. Pour différencier les dés libres de ceux qui sont sélectionnés ou verrouillés un état est assigné au dé.

Les interfaces IDe et ILance sont visibles de l'extérieure par la vue et le contrôleur. Alors que ILancable est prévu uniquement pour le contrôleur.

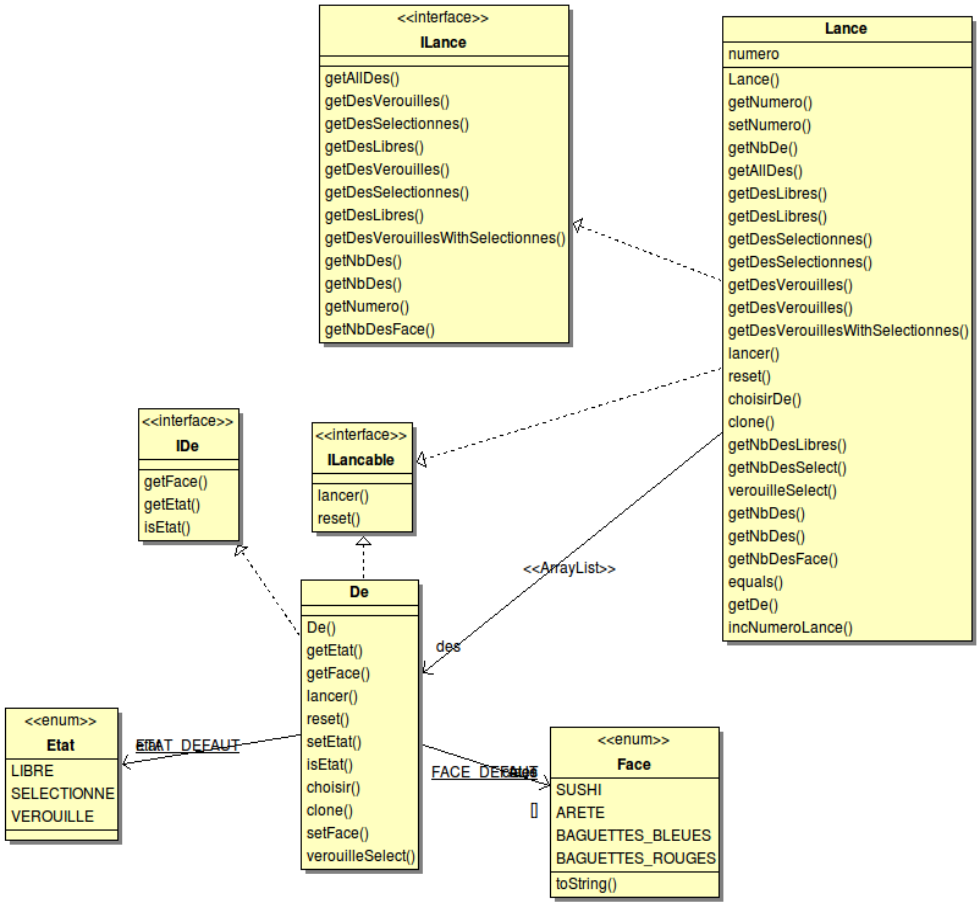


Illustration 17: Diagramme de classes des dés

b. Bar et portions

Les sushis et les arêtes sont tous deux représentés par l'objet Portion. Ces portions sont stockées dans deux listes du bar(l'une pour les arêtes et l'autre pour les sushis). Leur lecture se fait par les interfaces IPortion et IBar.

Deux énumérations contiennent le nombre et la valeur des sushis et des arêtes.

L'interface IPortionTransfert est utilisée à l'intérieur du modèle pour transférer les portions du bar vers le joueur ou de joueur en joueur.

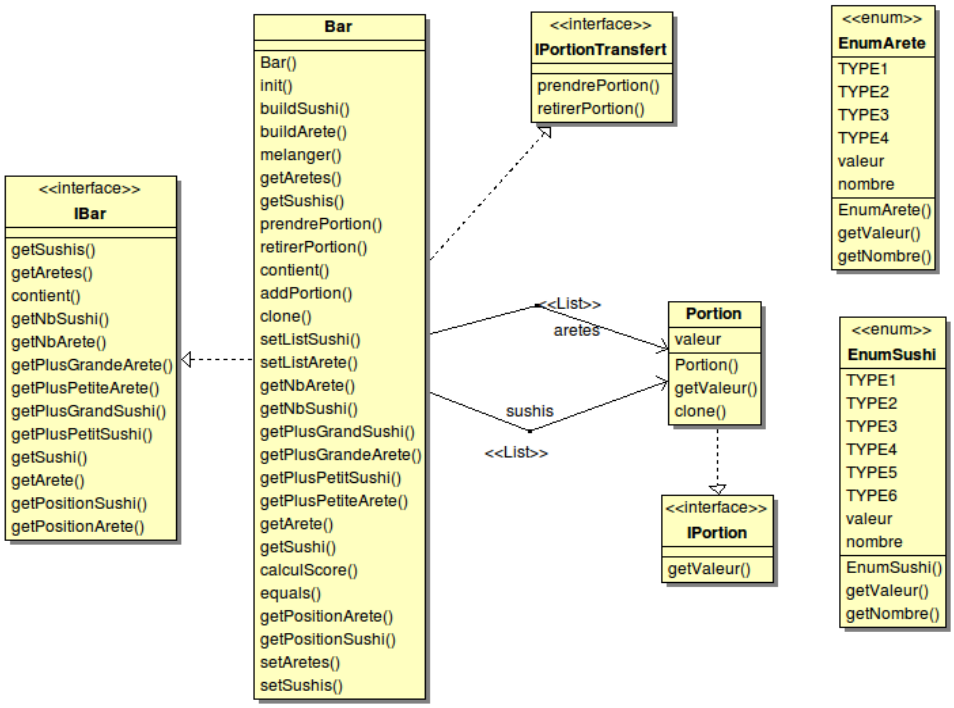


Illustration 18: Diagramme de classes du bar

c. Joueur

Les informations relatives aux joueurs sont stockés dans la structure joueur.

Les piles de sushis et d'arêtes du joueur sont stockés dans un bar personnel.

Pour distinguer les joueurs humains des IAs on attribut une énumération correspondant au type de joueur.

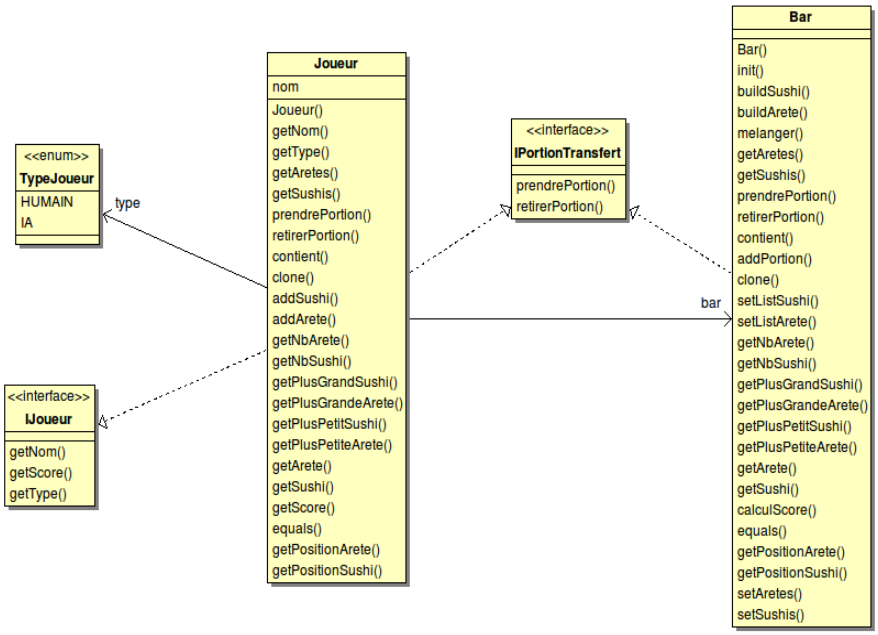


Illustration 19: Diagramme de classes des joueurs

d. Tour, modèle et sauvegarde

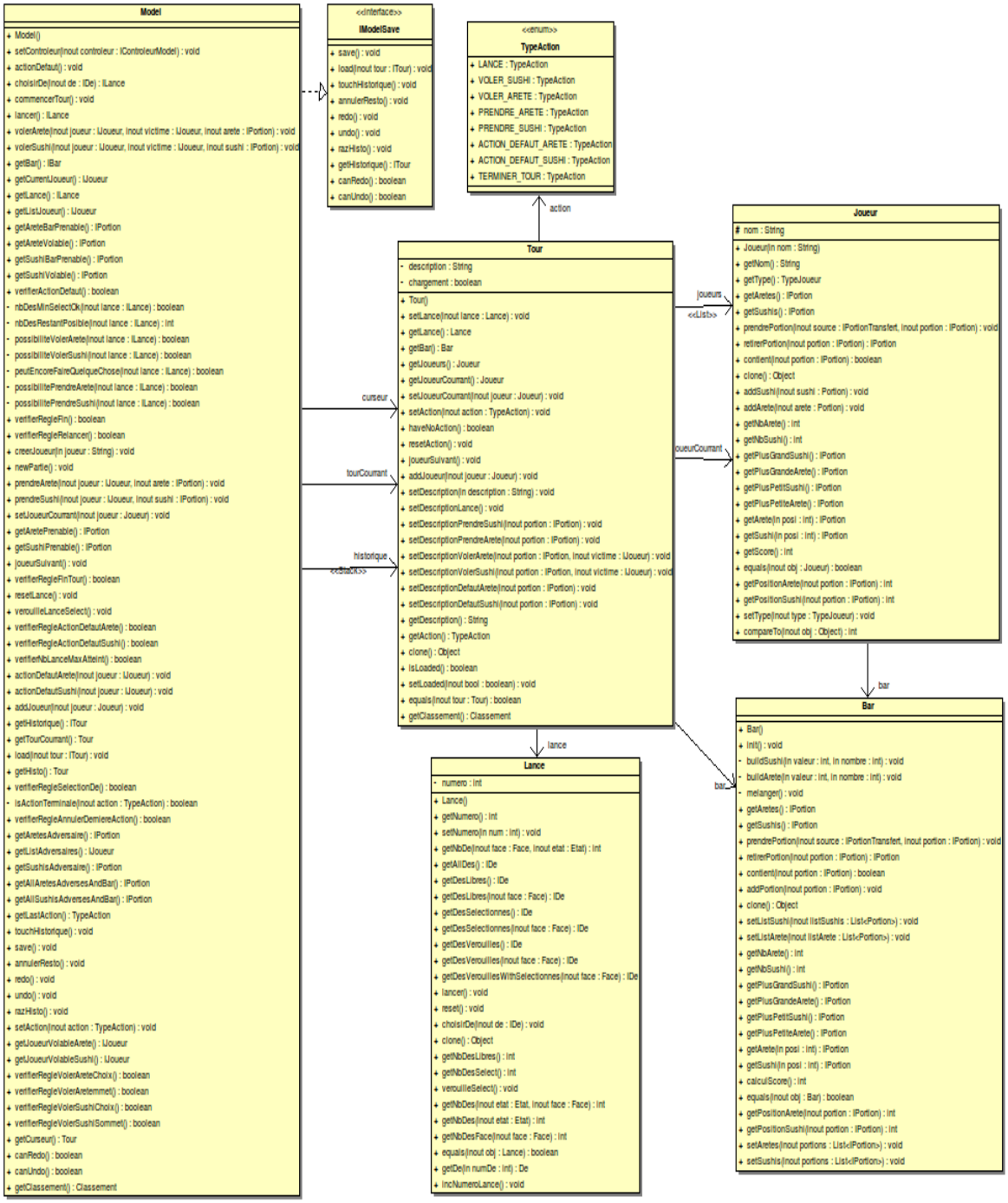


Illustration 20: Diagramme de classes du model et du tour

Pour faciliter les sauvegardes des informations contenues dans le modèle, le patron de conception memento a été utilisé. La mise en œuvre de ce patron se fait dans l'interface IModelSave via la méthode save qui va créer une image du modèle et undo / redo vont restaurer une image enregistrée.

Pour faciliter la réalisation des images et leur restauration, les diverses informations du modèle sont stockées dans une structure Tour qui nous permet de sauvegarder les divers éléments(en les clonant via la surcharge de la méthode clone).

La sauvegarde est effectuée à chaque début de tour par le contrôleur avant qu'une quelconque action n'a été réalisée puis renouvelée a chaque lancer.

Les sauvegardes ainsi effectuées sont stockées dans une pile se trouvant dans le modèle. Si aucune sauvegarde n'a été restaurée, le pointeur de sommet de la pile de sauvegarde se trouve au sommet qui est la copie conforme du tour courant. Si un retour en arrière via un *undo* n'est pas effectué, le pointeur de sommet descend d'un élément et remplace le tour courant par une copie du tour pointé par le pointeur de sommet.

Si l'utilisateur annule son retour arrière (*redo*) le pointeur de sommet remonte d'un cran et on replace le tour courant par une une copie de l'élément pointé par le pointeur de sommet.

Maintenant si l'utilisateur décide de réaliser une action et que le pointeur de sommet ne correspond pas au sommet de la pile de sauvegarde toutes les sauvegardes recouvrant ce pointeur sont supprimées avant de placer une copie du tour courant sur cette pile.

Cette sauvegarde est ainsi utilisée dans le mode normal pour permettre à un joueur d'annuler la prise d'une portion et dans le mode opérateur pour rectifier un lancé ou une prise de portion.

En plus des éléments du plateau, on attribut a chaque tour une énumération correspondant à l'action réalisée au cours du tour (nul en début de tour).

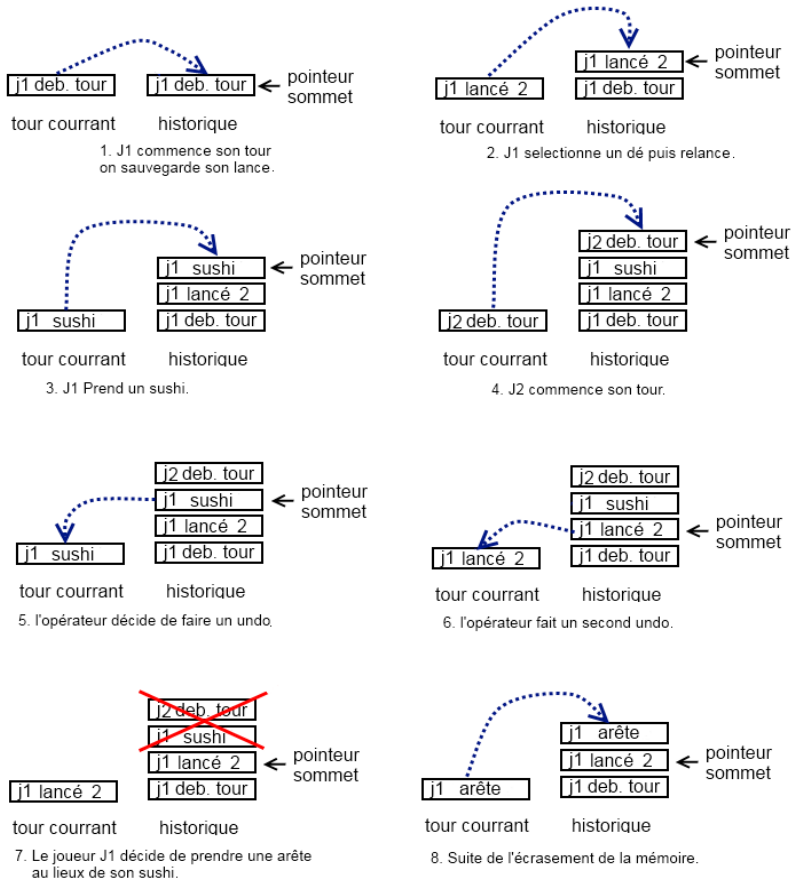


Illustration 21: Gestion exemple de sauvegarde

e. Vérification du bon fonctionnement du modèle

Pour vérifier le bon fonctionnement du modèle et pour faciliter le *refactoring* un certain nombre de tests unitaires on été créés. Ces tests se trouvent dans le package sushibar.modele du répertoire de travail tstModel. Ils s'assurent que les opérations de bases telles que les sauvegardes, les transferts de portions entre joueur et le bar sont bien réalisées par le modèle.

2.1.2 La vue

a. Fonctionnement

Ce package repose sur la classe IG qui allie la fenêtre principale de l'application à une fabrique de vue qui permettra d'interchanger les différentes vue selon le mode sélectionné ou le phase durant laquelle la partie se trouve (vue de la partie, tableau des scores etc ..) .

Illustration 22: Diagramme de classes de la vue

b. Images

Le chargement des images se fait part la classe Image qui est accessible de n'importe ou grâce à l'utilisation du template singleton une fois qu'elle a été instanciée et initialisée. Pour récupérer une image il suffira d''appeler la méthode getImage en spécifiant le nom de l'image ainsi que les dimension souhaitées.

c. Internationalisation

Pour rendre l'application plus attrayante il est intéressant de rendre plus facile le changement de langue. Pour ce faire le package i18n a été créée. Il permet de récupérer un texte donné en spécifiant sa localisation.

Tous les textes de bouton et dialogues sont répertoriés dans des fichier .properties. Pour ajouter une nouvelle langue il suffit de créer un nouveau fichier qui contiendra une traduction des textes dans la langue cible.

d. Vue de la partie

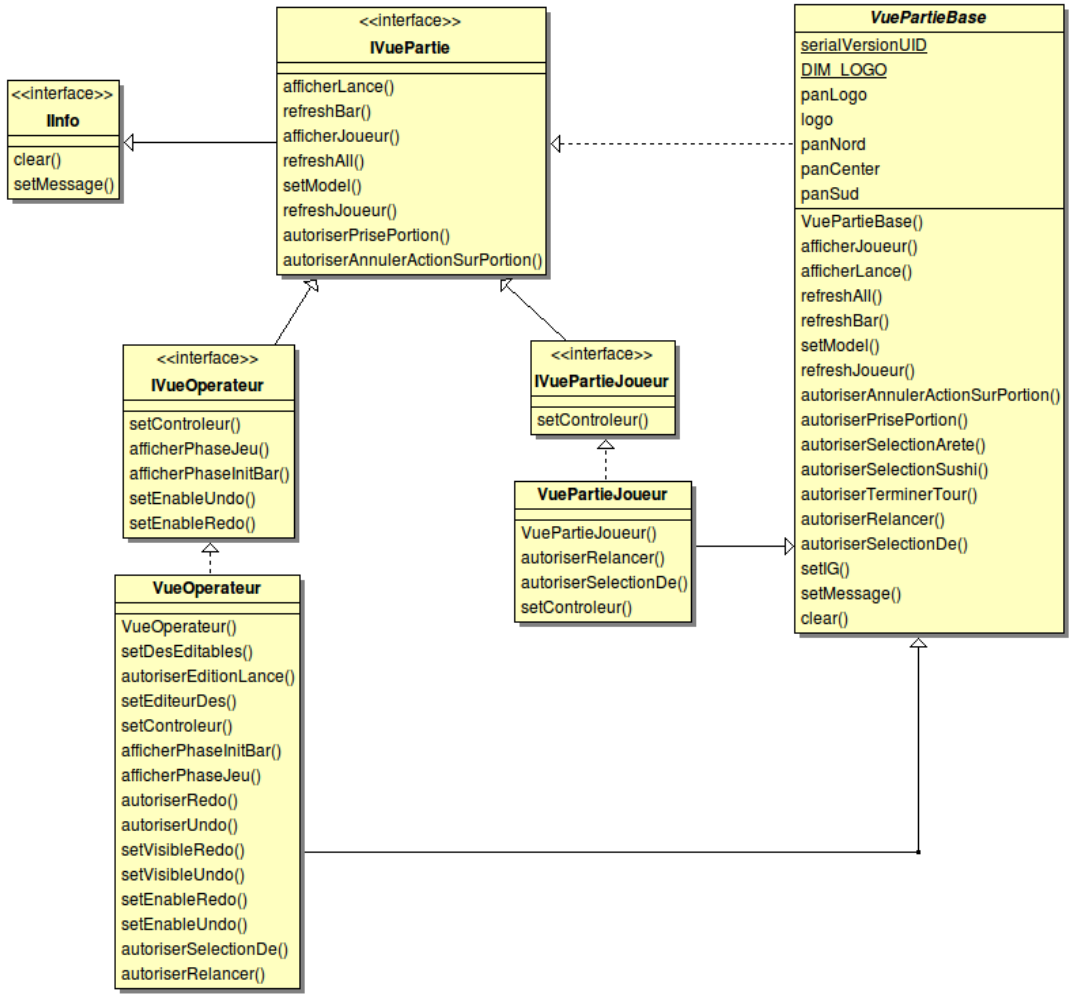


Illustration 23: Diagramme de classes de la vue d'une partie

Les vues de partie du mode normal et du mode opérateur sont plus ou moins identique. En effet elles ont toutes deux le même aspect par contre la vue de l'opérateur doit permettre de retourner en arrière pour modifier une action ou encore d'éditer les lancés de dés.

Ainsi les vues normal et opérateur héritent d'une vue abstraite vue partie base qui contient elle même une version abstraite de l'élément.

e. Structure de la vue

Illustration 24: Composition de la VuePartieJoueur

Comme on peut le remarquer la vue est composée de différentes zones (barre, groupe de dés a lancer, joueurs), il est donc naturel d'ajouter ces éléments à la vue sous la forme de panneau. Chacun des ces panneaux sont responsables des éléments qu'ils contiennent ainsi le panneau bar écoutera les chacune de ses portions pour notifier le contrôleur si le joueur décide de prendre une portion.

Tous ces éléments étant tout désactivable et ont besoin de notifier le contrôleur gérant la partie, héritent d'AbstractExecuteur qui possède en attribut une interface de IcontroleurPartie

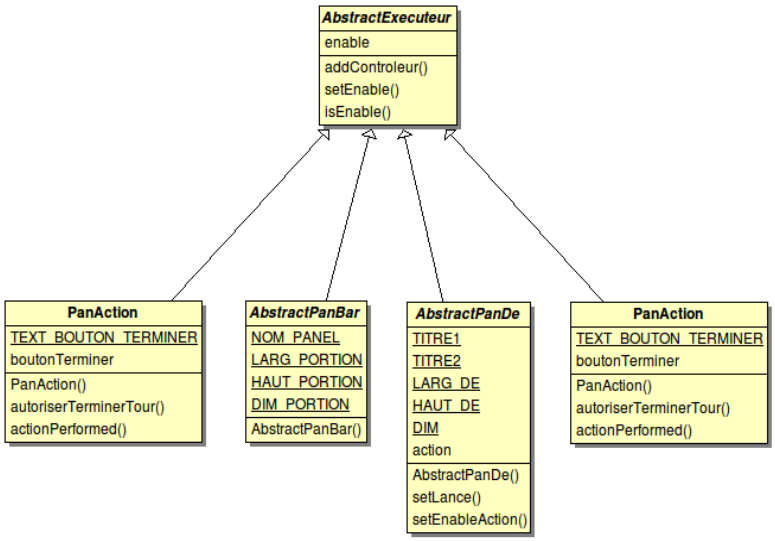


Illustration 25: Diagramme de classes de l'organisation des panneaux

2.1.1 Le contrôleur

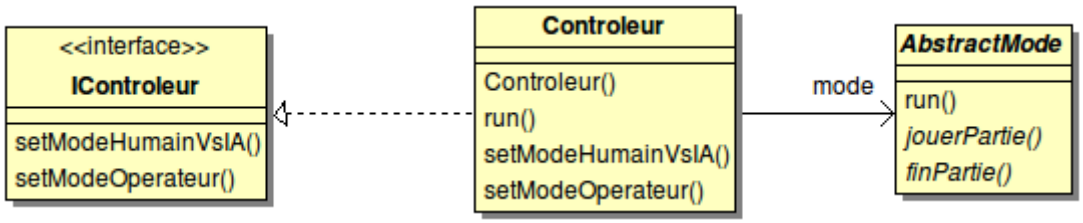


Illustration 26: Diagramme de classes du controleur

Ce package est orchestré par un contrôleur principale. C'est lui qui va lancer l'application par la méthode run et va afficher les différentes vues pour sélectionner le mode et créer la partie.

Pour gérer les différents mode les méthodes le contrôleur va se comporter comme une fabrique pour créer le bon contrôleur et va afficher la bonne vue via les méthodes setModeHumainVsIa et setModeOperateur.

a. Le contrôleur de la partie

Le contrôleur de la partie a pour rôle d'autoriser le jouer a réaliser telle ou telle action, lettre a jour la vue et vérifier si les conditions requise à l'exécution d'une action sont vraies et le cas échéant mettre a jour le modèle.

Toutes ces actions en dehors de la mise à jour de la vue sont effectuées par le contrôleur de base : ControleurPartie. Le ControleurGrfx va en plus mettre a jour la vue. Et les contrôleurs héritant de ControleurPartieGrfx aborderont une stratégie différentes selon le cas ou l'utilisateur est humain, ou s'il s'agit d'une IA ou encore si on est en mode opérateur ou non.

Quelque soit le mode d'utilisation ou que le joueur courant est humain ou une IA, le fonctionnement global est toujours le même. Par contre dans certain cas sont spécifique a un mode et a un utilisateur. Par exemple pour le mode opposant un humain a une IA, il faut réactualiser la vue et en fonction qu'il s'agisse du joueur humain ou de l'IA il faut autoriser au non l'interaction graphique.

Pour résoudre ce problème la solution retenue est la mise en œuvre du pattern decorator qui permet d'ajouter a une classe des fonctionnalités supplémentaire sans modifier son fonctionnement global.

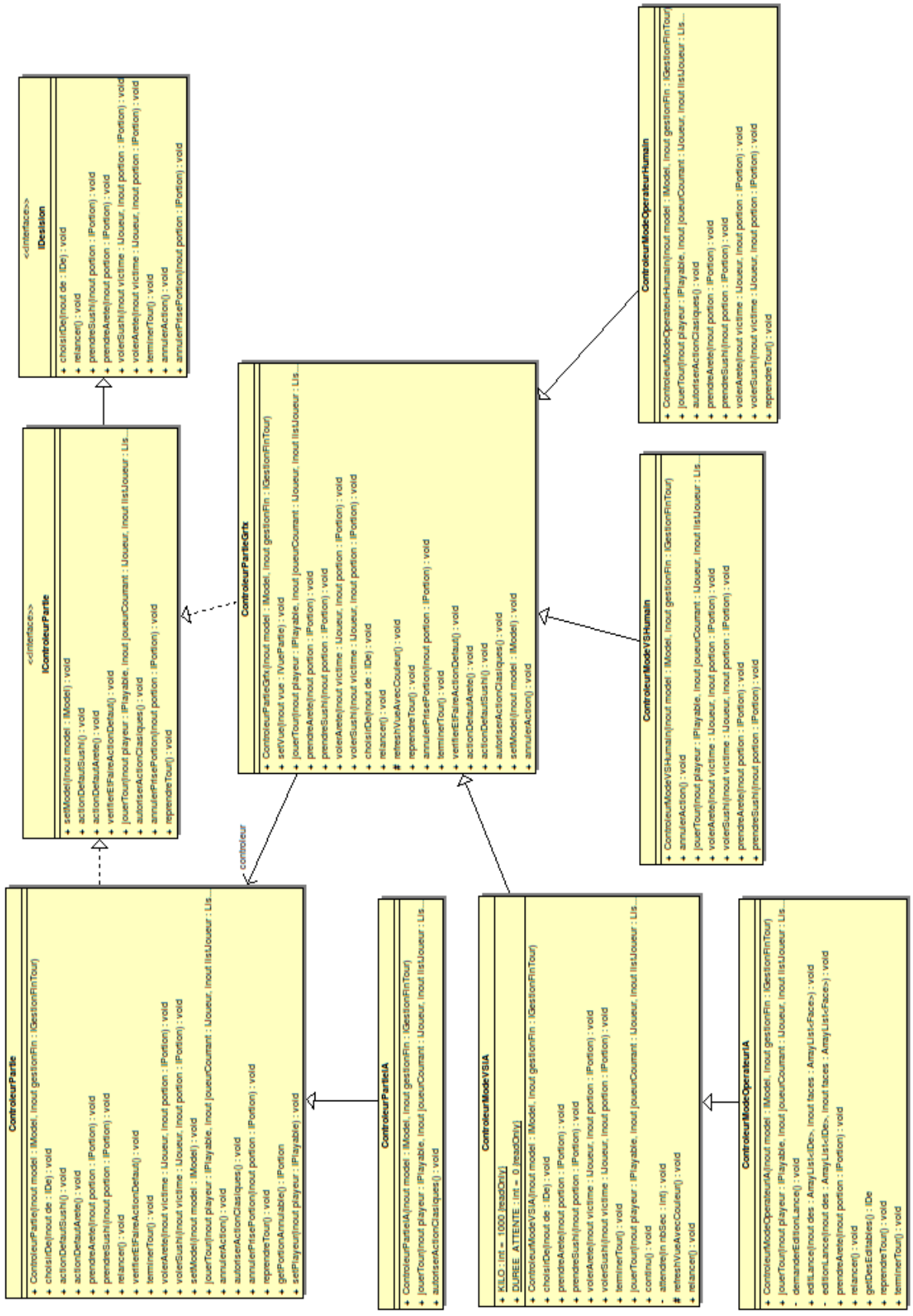


Illustration 27: Diagramme de classes du controleur de la partie

b. Les modes dans le contrôleur

Les modes sont exécutes par les classes ModeOperateur et ModeHumVsIA qui possèdent chacun la vue de la partie correspondante(vue de l'opérateur ou joueur) ainsi que les contrôleur de la partie spécifique au joueur ou à l'IA.

En début de partie lorsque l'utilisateur sélectionne le mode de fonctionnement, un objet correspondant à la classe du mode sélectionné est crée. Puis la partie débute à l'aide de l'appel de la méthode *void run().*

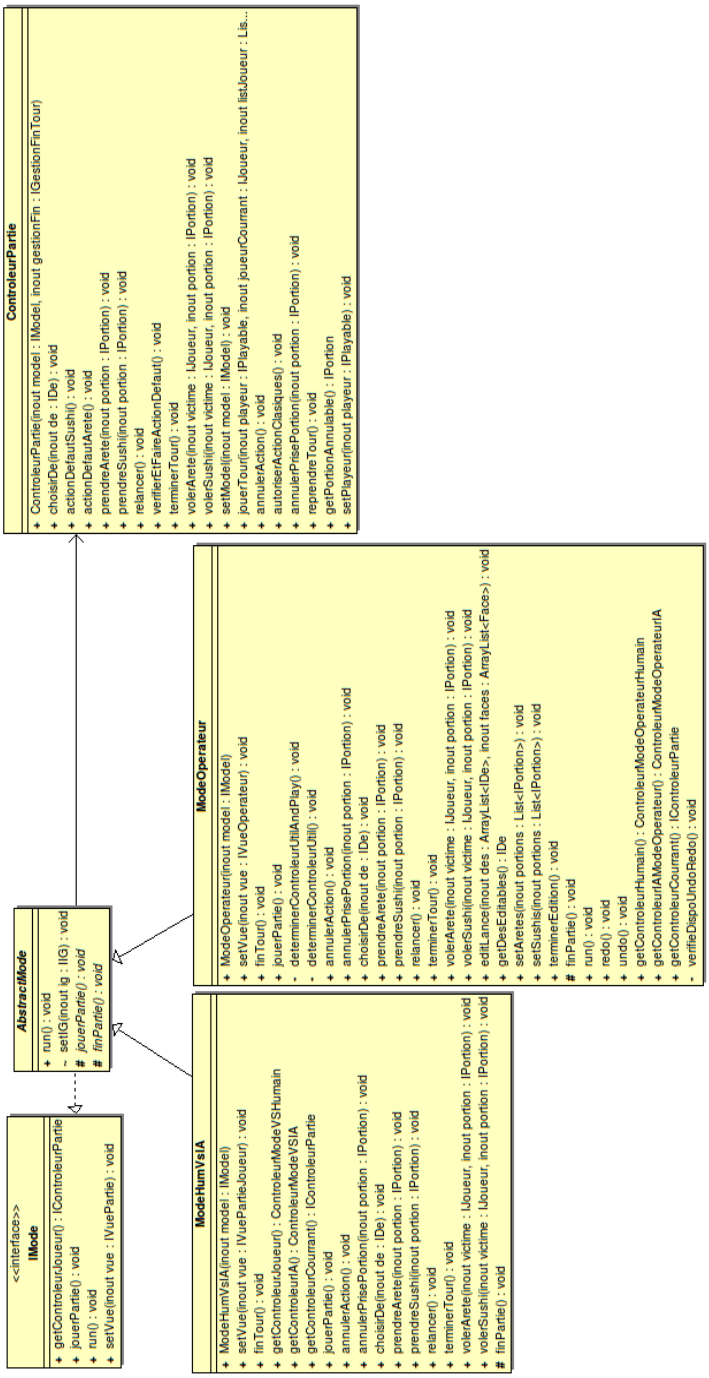


Illustration 28: Diagramme de classes des Modes

Chacun des modes posséde un fonctionnement qui lui est propre et aborde le fonctionnement de l'application sous une stratégie différente. En fonction du type de joueur courant (IA ou humain). Pour se faire les modes possèdent des *ControleurPartie* spécifique au type de joueur.

c. Spécificité du mode opérateur : l'édition

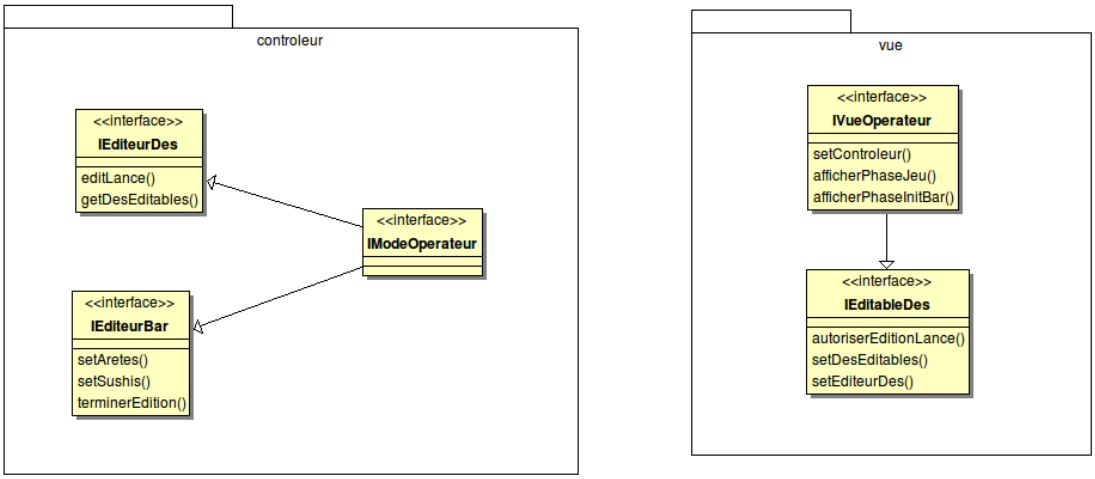


Illustration 29: Diagramme de classes précisions sur la sauvegarde

L'édition est rendu possible grâce aux interfaces IEdtieurDe et IEditeurBar du package controleur et de l'interface IvueOperateur.

2.1.2 IA

a. Présentation

En recherchant le mot Intelligence Artificielle dans le dictionnaire on trouve la définition suivante : « l'Intelligence Artificielle est la recherche de moyens susceptibles de doter les systèmes informatiques de capacités intellectuelles comparables à celles des êtres humains ».

L'application doit être capable de prendre par elle même des décisions et ainsi rivaliser avec les joueurs.

Pour parvenir à cela les mécanismes nécessaires à la prise de décision se trouve dans le package IA.

Illustration 30: Diagramme de classes de l'IA

b. Principe de fonctionnement des IAs

L'IA doit se comporter de la même façon qu'un joueur humain. Elle doit donc en reprendre les caractéristiques et avoir les mêmes actions possibles.

Pour rendre la conception des IAs facile il faut minimiser les tâches exercées par l'A. Ainsi il faut peut de chose pour créer une IA. En effet pour créer une nouvelle IA il suffit de créer une nouvelle classe qui redéfinira la méthode : *void stratChoisirAction().* Dans laquelle on indiquera les actions a réaliser dans telle ou telle condition.

Le système a été conçu de telle manière que les IAs n'ont pas besoin de connaitre les règles. C'est le contrôleur qui indiquera lors du tour de l'IA si telle ou telle action et disponible. Cette notification se fait par l'interface IPlayable qui est également utilisé par la vue pour autoriser le joueur a réaliser les actions possibles..

Mais les besoins d'information de l'IA sont plus grands que ceux de la vue. En effet pour autoriser un joueur a prendre un sushi chez un joueur ou dans le bar une seule méthode est utilisée. Mais dans le soucie de donner plus d'information à l'IA (pour adopter une meilleur stratégie), il est important de l'informer que tel ou tel joueur peut être volé. Ces notifications se font par l'intermédiaire l'interface IIA.

2.1.3 IA en action

Lorsque c'est le tour de l'IA de joueur, le contrôleur l'informe des actions possibles à réaliser selon le résultats du lancé, puis demande à l'IA de faire son choix via la méthode *void jouer(IModelLecture model, IDesision controleur).* Celle-ci va choisir une action et informer le controleur à l'aide l'interface Idessision (tout comme le fait la vue),en fonction de la stratégie programmée au sein de sa méthode vo*id stratChoisirAction().* Une fois que le contrôleur a reçu le choix de l'IA il redonne les actions réalisable par l'IA, jusqu'à ce que l'IA décide de terminer son tour.

Bilan et conclusion

Dans la mesure où l'application permet a un ou plusieurs joueurs humains de se confronter à des IAs, que la création d'une nouvelle IA se fait de façon facile et que le mode opérateur permet à une IA de joueur sur un vrai plateau, on peut considérer que les objectifs principaux du cahier des charges ont été tenus et les délais pour réaliser les différentes tâches ont été respectés.

L'une des partie les plus délicate du projet a été la conception du modèle pour qu'il soit facilement réutilisable et générique. En effet il n'est pas toujours facile de concilier généricité et spécificité de certains objets, notamment pour les modes de fonctionnements et les vues.

Un problème a été rencontré lors de la création des animations du jeu causé par l'utilisation des threads dans la bibliothèque Swing.

L'application trouve ses limites par l'absence du mode arène et l'IA proposée ne prend pas en considération les règles de probabilité dans la prise de ses décisions.

Néanmoins la découpe du projet et les tests unitaires existants permettront d'implémenter les fonctionnalités manquantes ou apporter d'éventuelles modifications tout en s'assurant que les contraintes existantes exercées par les différents objets sont bien satisfaites.

Je suis satisfait du projet Sushi bar car il a été l'occasion d'améliorer mes connaissances dans la modélisation des applications orientées objet reposant plus particulièrement sur le modèle MVC et fut ainsi un bon complément aux cours de génie logiciel.

Il m'a aussi montré que l'écriture des tests unitaires s'avère très utile pour vérifier le bon fonctionnement du modèle et ainsi écarter les sources d'erreurs. De plus les design patterns se révèlent être de précieux alliés pour implémenter certaines fonctionnalités.

Remerciements

Je tiens à remercier Monsieur Lagrue et Monsieur Cardon qui sont les auteurs du sujet projet ainsi que les tuteurs qui m'ont suivi et conseillé tout au long des dix semaines de ce projet.

Quelques liens

Intelligence Artificielle :	*http://fr.wikipedia.org/wiki/Intelligence_artificielle*
Le CRIL :	*http://www.cril.univ-artois.fr/*
	http://fr.wikipedia.org/wiki/Centre_de_recherche_en_informatique_de_Lens
Gigamic :	*http://www.gigamic.com/*
Pickomino:	*http://www.cril.univ-artois.fr/~lagrue/pickomino/*
Sushi bar :	*http://www.gigamic.com/sushibar-jeux-de-societe-c-28-p-514.html*

Annexes

[A] Les règles du jeu

ZickeZacke

Sushizock im GOCKELWOK

de Reiner Knizia

Mettez les bouchées doubles !

Pour 2 à 5 amateurs de sushis à partir de 8 ans.

Il n'y a pas que les vers grillés dans la vie ! Songeait le héron Harry Patapon après sa dernière visite à la basse-cour, quand soudain il a eu une idée culinaire de génie : des sushis ! Les poules branchées sont enthousiasmées par cette nouvelle carte des menus. Mais la qualité n'est pas encore au rendez-vous. La cuisine tourne vite au cauchemar et rapidement se pose la question des arêtes : combien vont se retrouver cette fois dans le gosier de notre volaille ?

But du jeu

À l'aide des dés, les joueurs essaient de s'emparer du maximum de ces délicieux sushis. Encore faut-il éviter les arêtes pour marquer des points. Bien entendu, les petites restent moins en travers de la gorge que les grandes ...

Le vainqueur est celui qui possède le plus de points à la fin de la partie.

1

Contenu

• 5 dés • 24 portions { 12 bouchées de sushi, 12 arêtes

Les dés

Ils indiquent 4 symboles différents :

sushi	arêtes	baguettes bleues	baguettes rouge
x10 (2 par dé)	x10 (2 par dé)	x5 (1 par dé)	x5 (1 par dé)

Les 12 bouchées de sushi (bleues, rapportent des points)

Les 12 arêtes (rouges, font perdre des points)

2

Préparation du jeu

Mélanger séparément les bouchées de sushi et les arêtes et former deux rangées, face visible.

Disposition

Déroulement de la partie

Les joueurs jouent à tour de rôle dans le sens des aiguilles d'une montre. Le dernier à avoir mangé des sushis reçoit les 5 dés et commence.

A son tour, le joueur dispose de trois lancers de dés au maximum pour s'emparer d'une portion. Une fois obtenue, son tour est terminé.

Le joueur commence par lancer les 5 dés. S'il ne peut ou ne veut pas prendre de portion, il laisse au minimum 1 dé de côté. (Les dés mis de côté viendront s'ajouter au résultat obtenu.)

Il relance une deuxième fois les dés restants. S'il ne veut ou ne peut toujours pas prendre une portion, il relance les dés une dernière fois. Là encore, il met de côté au minimum 1 des dés de son deuxième lancer. (Celui qui n'a plus qu'un dé au deuxième jet ne dispose pas de troisième lancer.)

Le joueur est obligé de prendre une portion au plus tard à l'issue de son troisième jet, soit sur la table, soit à l'un de ses adversaires.

3

Prise des portions sur la table

Si au moins un des dés indique un **sushi**, le joueur peut s'emparer d'un sushi. Le nombre de sushis obtenus indique si le joueur peut s'emparer, respectivement, de la 1re, 2e, 3e, 4e ou 5e bouchée de sushi sur la table, en comptant de gauche à droite (voir exemple).

Si au moins un des dés indique une **arête**, le joueur peut prendre une arête. Le nombre d'arêtes obtenues indique si le joueur peut s'emparer, respectivement, de la 1re, 2e, 3e, 4e ou 5e arête sur la table, en comptant de gauche à droite (voir exemple). Les espaces qui apparaissent sont resserrés au fur et à mesure.

Exemple

Nick peut choisir de prendre, soit la 2e bouchée de sushi, soit la 3e arête sur la table. À la place, il décide de garder des dés de côté et de relancer les autres.

Chaque joueur forme deux piles devant lui avec les portions récupérées :

- L'une composée uniquement de bouchées de sushi;
- L'autre composée uniquement d'arêtes.

Les portions récupérées sont placées sur le dessus de la pile correspondante (face visible). Il est interdit de regarder les portions recouvertes.

4

Vol de portions chez ses adversaires

Le joueur qui obtient au minimum trois baguettes de même couleur aux dés, gagne le droit de voler une portion à l'un de ses adversaires :

- **Trois dés indiquant des baguettes bleues :**
  - ➡ Le joueur peut voler une bouchée de sushi à l'adversaire de son choix !
- **Trois dés indiquant des baguettes rouges :**
  - ➡ Le joueur peut voler une arête à l'adversaire de son choix !

Mais seule la portion située au sommet de la pile correspondante peut être volée.

- **Quatre ou cinq dés indiquant des baguettes de même couleur :**

  - ➡ Le joueur peut voler **n'importe quelle** bouchée de sushi correspondant à la couleur obtenue. Il annonce quel adversaire il souhaite voler, puis la position de la portion convoitée dans la pile *(Par exemple : « Je voudrais la deuxième bouchée de sushi en partant du haut dans la pile de Sophia. »)*. Il est interdit de regarder avant !

pile de Sophia

L'embarras du choix ...

Si le résultat du dé offre plusieurs possibilité à un joueur, il choisit l'une d'entre elles.

Exemple

À l'issue de son 3e jet, Nick a obtenu trois baguettes rouges, une portion de sushi et une arête. Il doit alors se décider entre : prendre la 1re portion de sushi ou la 1re arête sur la table ou voler l'arête au sommet de la pile de l'un de ses adversaires.

Cf. fig.

Pas le choix ...

Si, à la fin de son tour, un joueur ne peut ni prendre une portion sur la table, ni voler un de ses adversaires, il est obligé de prendre l'arête avec la plus forte valeur négative encore sur la table. S'il ne reste plus d'arêtes, il prend la bouchée de sushi avec la plus petite valeur.

Exemple

À l'issue de son 3e jet, Luc ne peut ni voler, ni s'emparer d'une bouchée de sushi. (Il ne reste plus qu'une bouchée de sushi sur la table, mais ses dés indiquent 2 bouchées de sushi.) Il doit donc prendre l'arête avec la plus forte valeur négative encore sur la table. À ce moment du jeu, il s'agit du -3 car le -4 a déjà été pris.

Cf. fig.

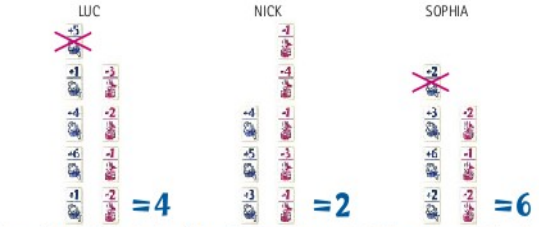
Fin de la partie et vainqueur

La partie s'arrête dès que la dernière portion sur la table est prise.

Les joueurs rapprochent alors leurs deux piles l'une de l'autre. Toutes les bouchées de **sushi** qui dépassent ne comptent pas et sont retirées du jeu. C'est pourquoi il faudra nécessairement prendre des arêtes durant la partie. Les arêtes comptent **toujours**.

Chacun additionne ensuite la valeur de ses bouchées de sushi, puis retranche la valeur de ses arêtes. Le joueur qui totalise le plus de points remporte la partie.

Exemple de décompte



Luc possède 6 bouchées de sushi mais a également récupéré 4 arêtes. Il doit donc se débarrasser de la bouchée de sushi au sommet de sa pile (+5). Elle ne compte pas. Il additionne alors la valeur de ses quatre bouchées de sushi (+12) et retranche la valeur de ses arêtes (-8). Il totalise donc 4 points.

Nick possède plus d'arêtes que de bouchées de sushi. Contrairement aux sushis, les arêtes en trop ne sont pas retirées du jeu mais sont prises en compte dans le résultat. Il conserve toutes ses bouchées de sushi et totalise tout de même 2 points.

Sophia ne peut pas compter la bouchée de sushi au sommet de sa pile (+2) et la retire du jeu. Elle remporte cependant la partie avec 6 points.

Exemple de tour de jeu

Après son 1er jet, Sophia pourrait prendre la 2e bouchée de sushi ou la 1re arête sur la table et avoir ainsi fini son tour. Mais elle décide, à la place, de mettre les baguettes bleues de côté.

Elle relance les autres dés et obtient deux autres baguettes bleues, un sushi et une arête. Elle pourrait décider de s'arrêter en prenant la 1re portion de sushi ou la 1re arête sur la table ou en volant le sushi au sommet de la pile de l'un de ses adversaires. Mais elle préfère garder de côté les deux autres baguettes bleues pour en avoir trois et relancer les deux dés restants.

pile de Nick

Sophia obtient des nouvelles baguettes bleues et un sushi. Comme elle possède maintenant 4 baguettes bleues, elle peut voler le sushi de son choix dans la pile de l'un de ses adversaires. Elle pourrait également s'emparer du 1er sushi sur la table. Elle réclame le sushi situé sous la pile de Nick et le place sur sa pile de sushis.

Zoch GmbH
Brienner Straße 54a
80333 München
www.zoch-verlag.com

Distribution Suisse : CARLETTO AG
Moosacherstraße 14
Postfach
CH-8820 Wädenswil
www.carletto.ch