S10

Sokoban et IA

Etude préalable

Année 2020/2021

Anthony BRIOT - Benoit NICOL - Lucas SAKER - Quentin VRIGNON

Tuteur : Vincent THOMAS

Table des matières

Partie 1 - Description du projet, étude et acteurs intéressés

1.1) Description du projet

Description générale

Le projet consiste à **mettre en place un jeu similaire au jeu du Sokoban**, dans lequel un personnage va devoir déplacer des caisses sur des places prédéfinies afin de finir différents niveaux en rangeant l'entrepôt. Il pourra, s'il le souhaite, faire appel à une IA lorsqu'il voudra connaître la solution pour résoudre le niveau.

Objectif principal du projet

L'objectif principal de notre projet est de permettre la vulgarisation de l'IA. L'objectif à terme serait de montrer cette application (ce jeu) lors d'une fête de la science à des élèves qui ne connaissent pas encore la programmation et l'IA.

Description technique

Les déplacements du personnage :

Dans un premier temps, le personnage pourra **être déplacé par l'utilisateur** grâce au clavier dans les quatre directions cardinales ou grâce à la souris. Les déplacements se feront case par case. Un déplacement sera réalisé dans le cas où la case où le joueur souhaite aller est libre. Dans le cas où une caisse se trouve sur la case où nous voulons aller, alors il sera possible de la pousser si la caisse n'est pas bloquée par la case suivante. Il faudra aussi **gérer les collisions**.

Le déplacement des caisses :

Pour déplacer une caisse, le joueur ne peut que **pousser une caisse vers une case libre**, il faut donc faire attention à ne pas bloquer une caisse dans un coin. De plus, le personnage **ne peut pousser qu'une caisse en même temps**, s'il y a plusieurs caisses l'une derrière l'autre, il ne pourra pas les pousser.

L’interface graphique :

Par ailleurs, nous devrons **développer une interface graphique 2D** pour pouvoir visualiser le déplacement de notre personnage, les caisses, les murs ainsi que les emplacements spéciaux sur lesquels les caisses doivent être placées. Cette interface devra comporter un espace affichant le jeu en temps réel ainsi qu’un espace comprenant les boutons de retour ou encore d’aide pour l’IA.

L’IA :

Les IA sont aujourd’hui **omniprésentes**. En effet, on en **retrouve dans pratiquement tous les domaines** et c’est pour cela que l’application des IA est presque infinie.

L’un des objectifs de ce projet sera d’**implémenter une ou plusieurs IA** pouvant résoudre différents niveaux de Sokoban pour trouver la solution optimale à chaque niveau et l’afficher dans l’interface graphique, c'est-à-dire pouvoir visualiser les déplacements de la solution trouvée par l’IA.

La difficulté sera d’écarter les problèmes de déplacements inutiles pouvant rendre une partie infinie si l’IA rentre dans une boucle (si elle répète toujours le même déplacement). Nous pourrons alors présenter à la fin de notre projet dans notre dernier rapport **une comparaison des différentes IA** avec pour chacune sa particularité, ses points positifs et ses points négatifs. Nous pourrons potentiellement essayer de **généraliser certaines IA à d’autres jeux**.

Les différents niveaux :

Il faudra également pouvoir **charger plusieurs niveaux** avec des difficultés différentes, voire même ajouter des objets supplémentaires pour certains niveaux, comme par exemple des objets permettant de pousser deux caisses en même temps ou autre. De plus, nous pourrons potentiellement rajouter un système permettant de créer soi-même des niveaux qui pourront être joués par la suite.

Jeu en réseau :

Si le temps et la situation le permettent nous pourrions essayer de **développer un système de jeu en réseau** pour jouer à plusieurs et donner des actions spécifiques à chaque utilisateur. Chaque joueur pourra faire un ou plusieurs déplacements donnés et donc le jeu pourra se résoudre seulement grâce à un **travail d’équipe**.

1.2) Etude de l'existant

- Le site officiel du jeu Sokoban [1] nous indique le principe général du jeu tel qu'il a été créé à la base. Il présente donc des éléments intéressants que nous pourrions utiliser par la suite, que ce soit au niveau de la forme graphique ou bien au niveau du code.

- Un second type de Sokoban [2] qui a été créé par un internaute anonyme avec différents thèmes graphiques pourra nous être utile.

  Il y a également une interface permettant de créer son propre thème de jeu et la possibilité de créer ses propres niveaux.

  Conception de solveurs :

- Solveur créé par l'université d'Alberta [3] utilisant un algorithme de type IDA* qui permet de résoudre le jeu Sokoban dans des niveaux de complexité moindres, c'est à dire avec seulement des caisses à déplacer et non pas d'autres types d'objet.

- Mémoire réalisé par Michaël Hoste sur la résolution de Sokoban par solveur.[4]

  Il y présente différents moyens de résoudre Sokoban avec leurs avantages et leurs inconvénients.

  Il va donc falloir choisir quelles solutions existantes nous devons ou pouvons implémenter en fonction de leur pertinence et de leurs atouts.

1.3) Etude technique sur l'existence de solutions

Pour le projet en général, nous allons utiliser **le langage de programmation Java**, car c'est le langage principal que nous utilisons à l'IUT et dont on connaît les outils pour réaliser cette interface. Nous savons, dans un premier temps, que nous allons utiliser **le patron de conception MVC**.

En effet, cette méthode s'occupe de l'architecture du code, en le séparant en trois parties : modèle, vue et contrôleur. Cela nous permettra de gérer au mieux les évènements créés par l'utilisateur et d'actualiser notre interface graphique en conséquence.
Le modèle MVC **possède de nombreux avantages** et peu d'inconvénients ce qui nous dirige donc vers ce choix.

Pour ce qui est des solveurs, nous allons nous **inspirer d'existants** afin d'en créer et de les ajouter à notre propre jeu. Ainsi, avec plusieurs solveurs, nous allons pouvoir les comparer dans le but de trouver le(s) plus performant(s) (nombre de coups, temps, etc.) à l'aide des données récoltées.

Ces solveurs **peuvent être réalisés dans différents langages**. Il faudra donc les comprendre clairement pour pouvoir ensuite les ajouter, si nous y trouvons un intérêt, à notre jeu.

Différents algorithmes de recherche peuvent être abordés comme par exemple :

- L'algorithme de **Dijkstra.** [5]
- L'algorithme de type **A*.** [6]
- L'algorithme de type **IDA***.

L'algorithme A* est expliqué dans le 5e point, à la page suivante.

Nous avons également vu l'existence de la **contraction hiérarchique**[7] et de **l'apprentissage par renforcement**[8] qui pourraient nous être utile mais sur lesquels nous ne nous sommes pas attardés pour le moment.

1.4) Acteurs intéressés

Plusieurs secteurs pourraient s'avérer être intéressés par ce projet, voire même devenir de futurs clients.

On pourrait imaginer, par exemple, les secteurs du **développement de jeux vidéo**, de la réception de commande, de la préparation de commande ou même de la mise en stock.

En effet, le domaine d'application de l'IA ne cesse de s'agrandir et c'est pour cela qu'il est encore possible de progresser dans ce domaine. L'adaptation de l'IA à un projet est donc multiple, voire **infinie** de nos jours.

Ce domaine est voué à trouver de nouveaux acteurs dans un futur proche.

1.5) Explication de A*

A* est un algorithme de résolution que l'on peut représenter sous forme d'arbre d'état.

Chaque état possède une valeur heuristique (composée du coût réel pour arriver au nœud courant et du coût prévisionnel pour arriver à l'état final) correspondant au coût de la distance entre l'état initial et l'état final.

A chaque itération, on se positionne sur l'état ayant la plus petite valeur heuristique parmi tous les nœuds de l'arbre non visités et fils d'un nœud déjà visité jusqu'à arriver à l'état final par le chemin le plus court possible.

Partie 2 : Conception initiale

2.1) Liste des fonctionnalités

Fonctionnalités du jeu de base :

- Création des objets du jeu (caisses, murs, personnage).
- L'application affiche une fenêtre principale.
- L'application affiche la fenêtre de choix de jeu.
- L'application affiche la fenêtre de jeu.
- L'application affiche le personnage et permet de le déplacer.
- L'application affiche les murs pour créer le labyrinthe.
- L'application affiche les caisses (aux places indiquées).
- Le jeu gère les collisions avec les caisses et les murs.
- Le jeu gère le fait de pousser les caisses.
- L'application affiche les cases où ranger les caisses.
- Le jeu gère le fait d'être bloqué.
- Le jeu gère la réussite du niveau (en vérifiant l'emplacement des caisses).

Fonctionnalités supplémentaires importantes

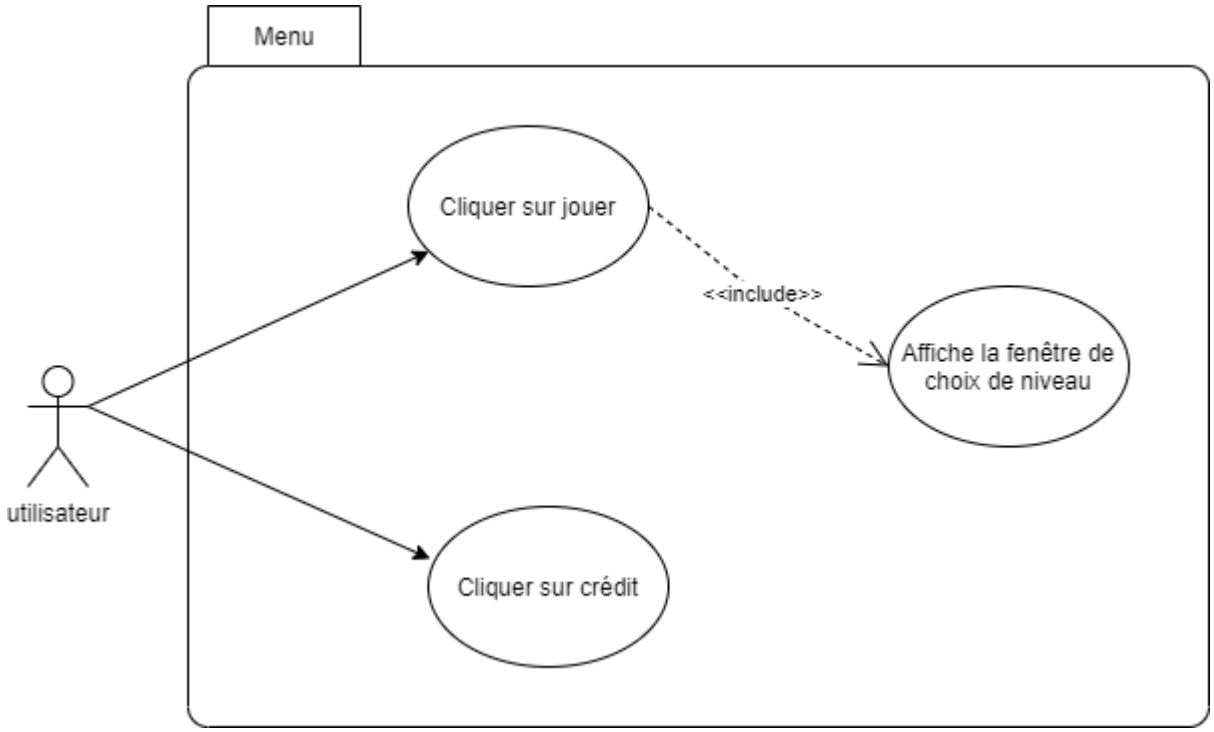
- Création d'une IA.
- Le jeu permet de choisir un niveau.
- L'application permet de retourner au menu principal à tout moment.
- L'application mémorise les coups joués (le nombre et quel coup en particulier dans l'ordre).
- L'application permet de revenir en arrière.
- L'application déplace le personnage pour finir le niveau seul (avec les données de l'IA).
- L'utilisateur peut dérouler l'IA (avec un menu déroulant et potentiellement un mode automatique).

Fonctionnalités supplémentaires facultatives

- L’application affiche un message lorsque l’on gagne.
- L’application nous avertit si l’on quitte le jeu en pleine partie.
- L’application propose des niveaux triés par difficulté.
- L’application demande le nom du joueur avant la partie (et l’enregistre).
- L’application enregistre le nom et le score pour un niveau donné.
- L’application affiche une interface pour créer des niveaux (et les tester).
- L’application permet de choisir différents thèmes pour jouer.
- L’application propose de sauvegarder la partie (fichier).
- L’application permet de reprendre le jeu enregistrer (en fonction du nom entrée).
- L’application permet de charger des niveaux externes.

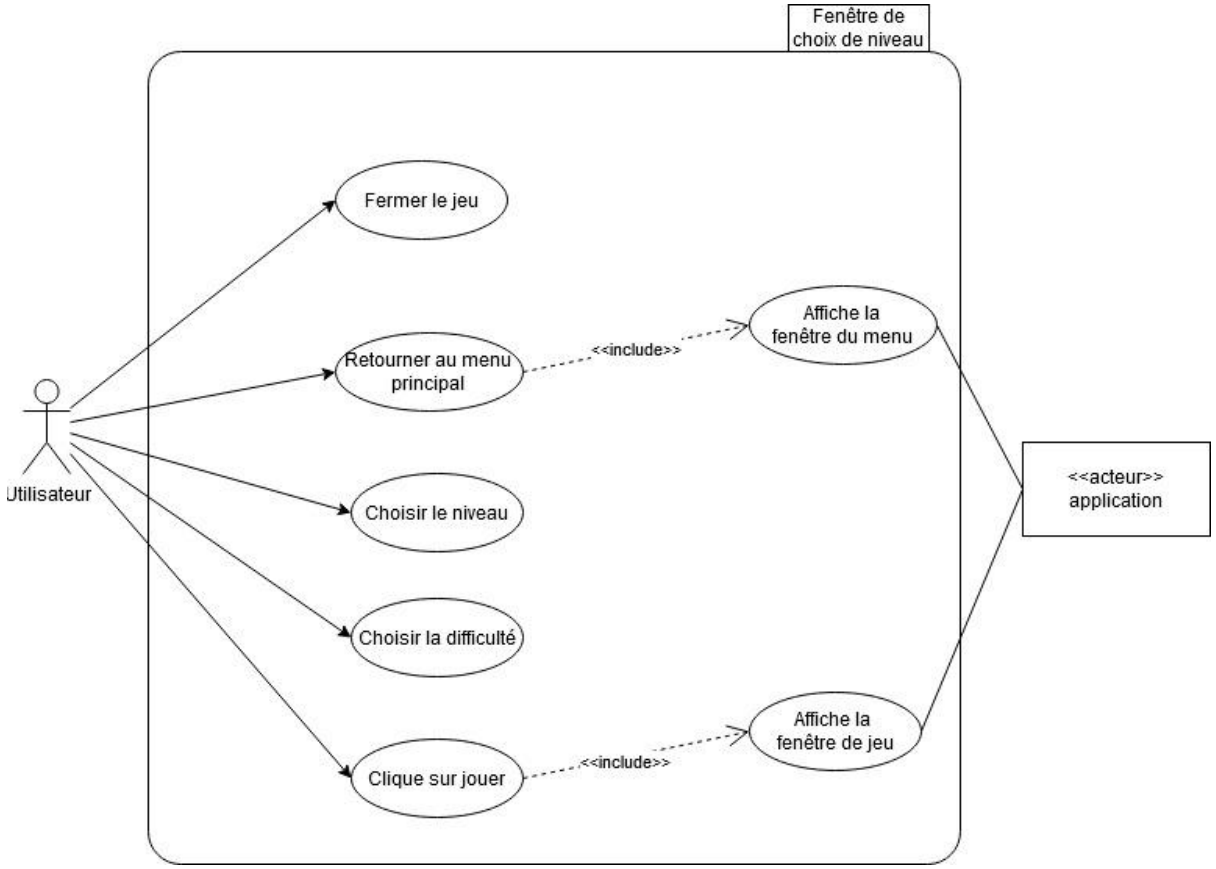
2.2) Diagrammes de cas d’utilisation

Diagramme de cas d’utilisation du menu principal



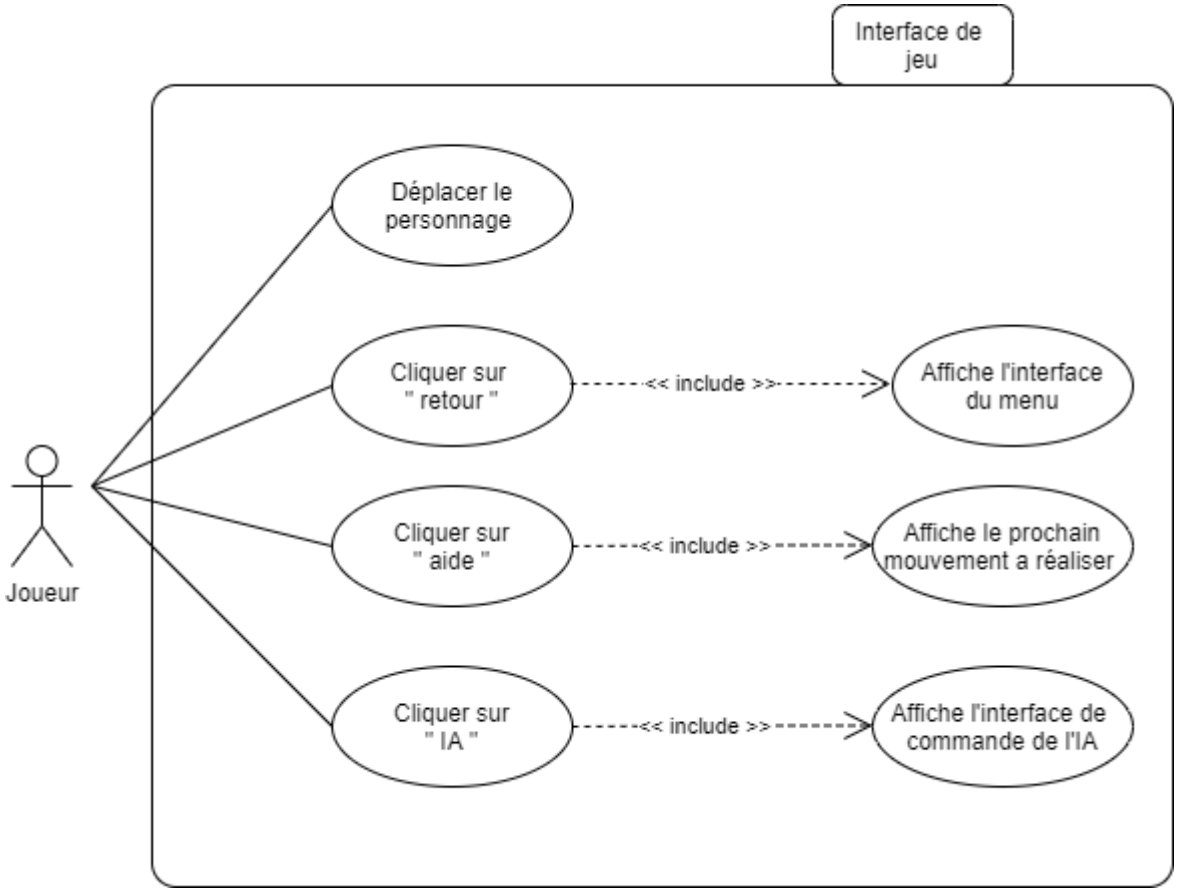
- L’utilisateur peut cliquer sur le bouton “Crédits” afin d’afficher la fenêtre des crédits avec le détail des créateurs du projet.
- Il peut cliquer sur le bouton “Jouer”, afin d’afficher la fenêtre de sélection de niveau pour jouer.

Diagramme de cas d'utilisation de la fenêtre de choix de niveau



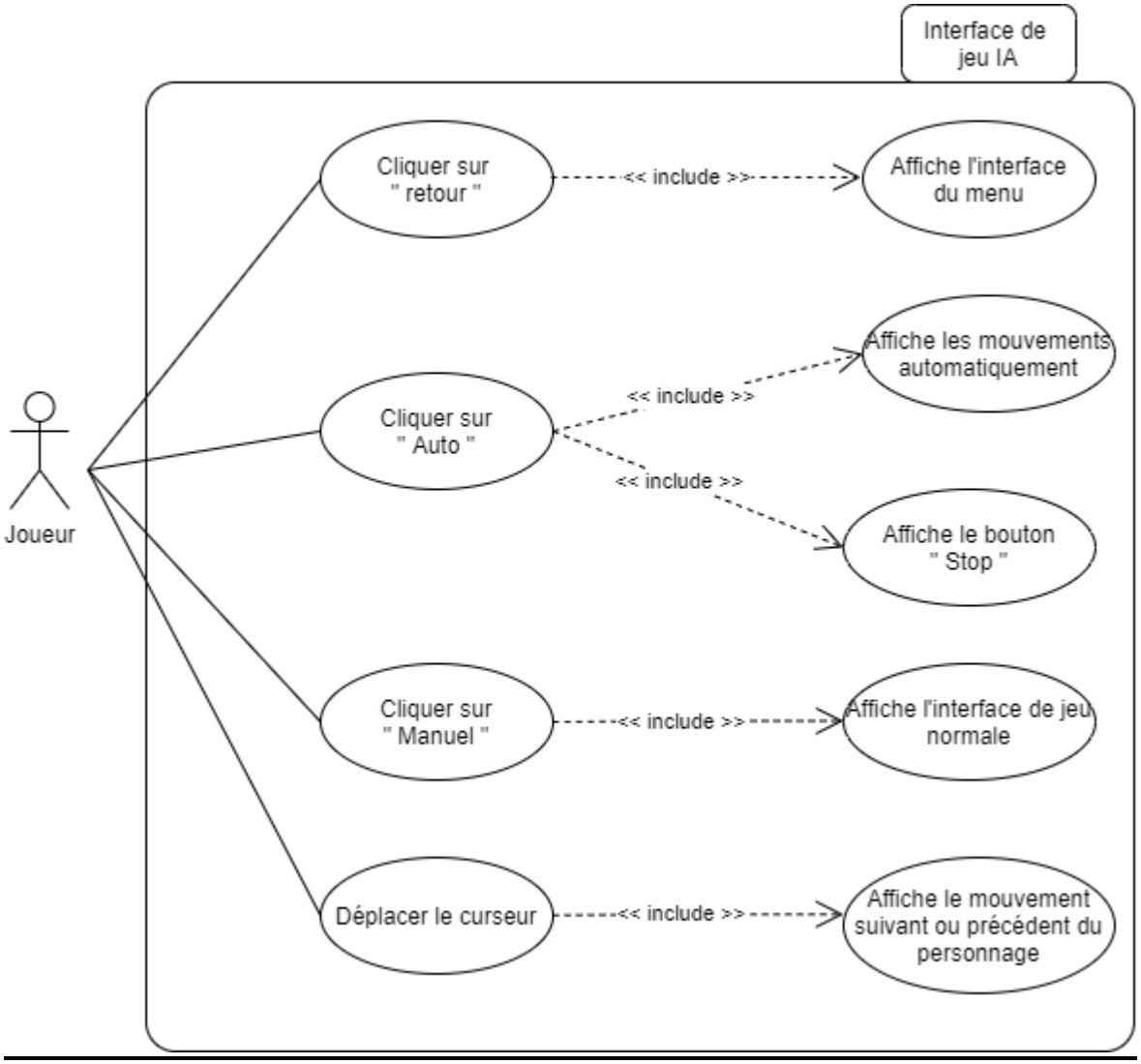
- L'utilisateur peut fermer la fenêtre du jeu à tout moment.
- Il peut cliquer sur "Retour" afin de retourner en arrière, c'est-à-dire au menu principal.
- Il peut choisir son niveau et peut également choisir la difficulté de celui-ci.
- Il peut cliquer sur jouer afin d'afficher la fenêtre de jeu avec le niveau qu'il a choisi.
- Les fenêtres sont affichées par l'application lors de l'appui sur le bouton.

Diagramme de cas d'utilisation de la fenêtre de jeu (mode sans IA)



- L'utilisateur peut déplacer le personnage avec les touches du clavier.
- Il peut également cliquer sur "Retour" pour retourner à la page du menu principal.
- Il peut cliquer sur "Aide" pour afficher le prochain mouvement à réaliser (calculé par l'IA)
- Il peut cliquer sur "IA" afin d'afficher l'interface de commande de l'IA sur le bas de l'écran.

Diagramme de cas d'utilisation de la fenêtre de jeu (mode avec IA)

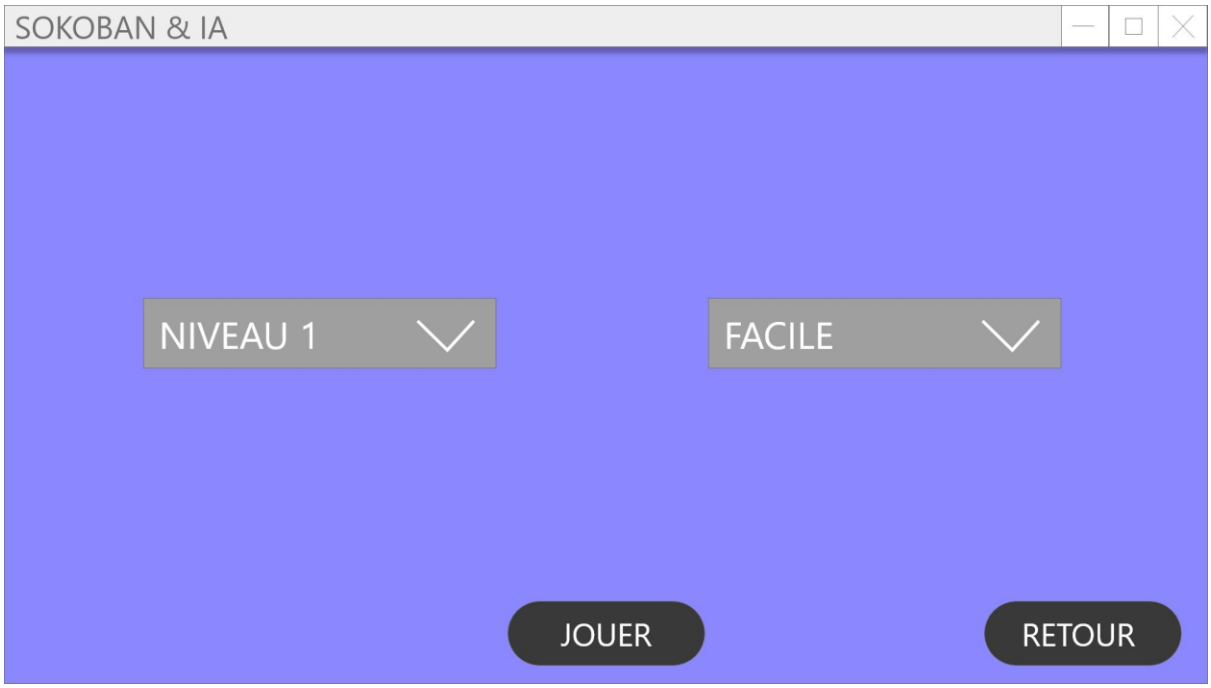


- L'utilisateur peut cliquer sur "Retour" pour retourner au menu principal.
- Il peut également cliquer sur "Auto" afin d'afficher automatiquement les mouvements que l'IA a préalablement calculé et le bouton devient alors un bouton "Stop" permettant d'arrêter les déplacements de l'IA.
- Il peut cliquer sur "Manuel" pour retourner à l'interface de jeu sans l'IA.
- Il peut, pour finir, déplacer le curseur de gauche à droite pour afficher le mouvement suivant ou précédent de la solution trouvée par l'IA.

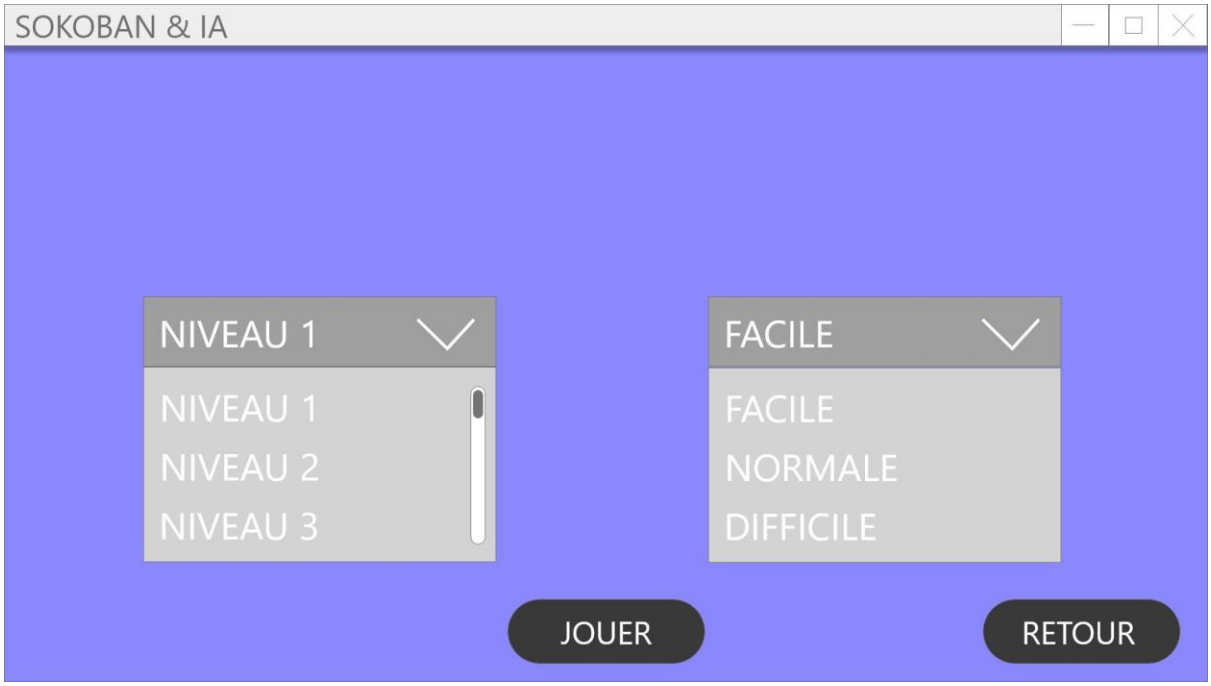
2.3) Maquettes

Maquette du menu principal

Maquette du menu du choix de niveau



Maquette du menu du choix de niveau 2



Maquette de l'interface de jeu simple

Maquette de l’interface de jeu avec IA

Maquette de l’interface de jeu avec IA 2

2.4) Recensement et évaluation des risques

Niveaux de risque :

Faible ●

Moyen ●

Elevé ●

Risques possibles :

a) Risques relationnels

- Problèmes liés à la situation sanitaire actuelle. ●
- Problèmes liés à la communication entre les membres du groupe. ●
- Problèmes liés à une mauvaise compréhension de la demande. ●

b) Risques techniques

- Problèmes liés à la conception de base du jeu. ●
- Problèmes lors de la création du jeu de base. ●
- Problèmes lors de la création de l'IA. ●
- Problèmes lors de l'intégration de l'IA dans le jeu. ●
- Problèmes lors de la création de la fonction heuristique. ●

2.5) Vision du produit

1. **Qui va être intéressé par le produit ? Qui est la cible ?**

Les personnes qui seront intéressées par notre produit seront principalement les personnes voulant présenter le principe de l'IA dans le contexte d'un jeu ou alors pour un usage purement personnel.

La cible pourrait être également des développeurs voulant utiliser notre application des IA. Ou bien, comme le principe du jeu Sokoban est similaire à un robot exécutant des tâches dans un entrepôt, notre produit pourrait servir à une personne voulant automatiser l'entrepôt en question et ainsi retranscrire les comportements de l'IA.

2. **A quels besoins le produit va-t-il répondre ?**

Notre produit va principalement répondre au besoin de montrer le déroulement d'une IA sur un jeu simple comme Sokoban.

Il pourra aussi par la suite être repris pour proposer des solutions pour des besoins d'automatisation, de gain de temps et d'optimisation. En effet, avec l'utilisation de l'IA pour remplacer la manutention par exemple, qui sera moins rapide, moins efficace et permettra de limiter les imprévus.

3. **Quelles sont les fonctionnalités critiques pour répondre aux besoins de façon à avoir un produit réussi ?**

Le but de ce projet est de réaliser une vulgarisation de l'IA en utilisant le jeu Sokoban, donc nous devons avoir toutes les fonctionnalités liées à l'interface du jeu, la fenêtre du jeu, les collisions, les déplacements du personnage, etc... Mais aussi et surtout toutes celles liées aux IA, comme le choix des IA ou bien les déplacements du personnage par l'IA.

Il faudrait également un système permettant d'afficher le mouvement suivant ou précédent de la solution que l'IA aura trouvé afin de l'analyser.

4. **Comment le produit se situe-t-il par rapport aux produits existants sur le marché ?**

Aujourd'hui, notre produit ne se situe sur aucun marché, ou bien les produits concurrents ne sont pas rendus publics. Donc nous ne savons pas nous placer par rapport aux produits qui sont potentiellement existants.

De plus, le produit devra être utilisé dans un cadre purement pédagogique ou personnel et ne sera donc pas vendu.

2.6) Prévision des itérations

Nous prévoyons un plan généralisé de développement pour les trois prochaines itérations :

Itération 1 : Nous souhaitons coder toutes les fonctionnalités liées au jeu Sokoban de base. Nous voulons également commencer à réfléchir sur la création d'une IA et son implémentation, notamment en écrivant un algorithme général. Enfin, nous commencerions à réfléchir sur la façon de calculer la fonction heuristique dont nous avons parlé dans la partie d'explication de A*.

Itération 2 : Durant cette itération, nous prévoyons de nous focaliser sur la création de l'IA et l'implémentation dans notre jeu. Nous pourrions nous concentrer sur le mode IA et sur l'affichage du déroulement de l'IA.

Itération 3 : Durant cette itération, nous pensons devoir continuer l'implémentation de l'IA afin de fournir une aide au joueur. Nous pourrions également coder quelques fonctionnalités supplémentaires facultatives.

Partie 3 : Prévision de l'itération 1

3.1) Planification et répartition des tâches

Plan des tâches à réaliser :

I) Création du jeu de base

1. Conception
    - Création des objets du jeu (caisses, murs, personnage)
    - Le jeu gère les collisions avec les caisses et les murs
    - Le jeu gère le fait de pousser les caisses
    - Le jeu gère le fait d'être bloqué.
    - Le jeu gère la réussite du niveau (en vérifiant l'emplacement des caisses)

2. Affichage
    - L'application affiche une fenêtre principale
    - L'application affiche la fenêtre de choix de jeu
    - L'application affiche la fenêtre de jeu
    - L'application affiche le personnage et permet de le déplacer
    - L'application affiche le labyrinthe
    - L'application affiche les caisses (au places indiquées)
    - L'application affiche les cases où ranger les caisses

II) Réflexion sur l'IA

1. Conception
    - Mise en place de squelettes de classes
    - Visualisation des liens entre les classes
2. Réflexion
    - Algorithme général A*

III) Commencer la fonction heuristique

- Voir les possibilités de fonction heuristique
- Ecrire un algorithme
- Réfléchir à comment l'implémenter

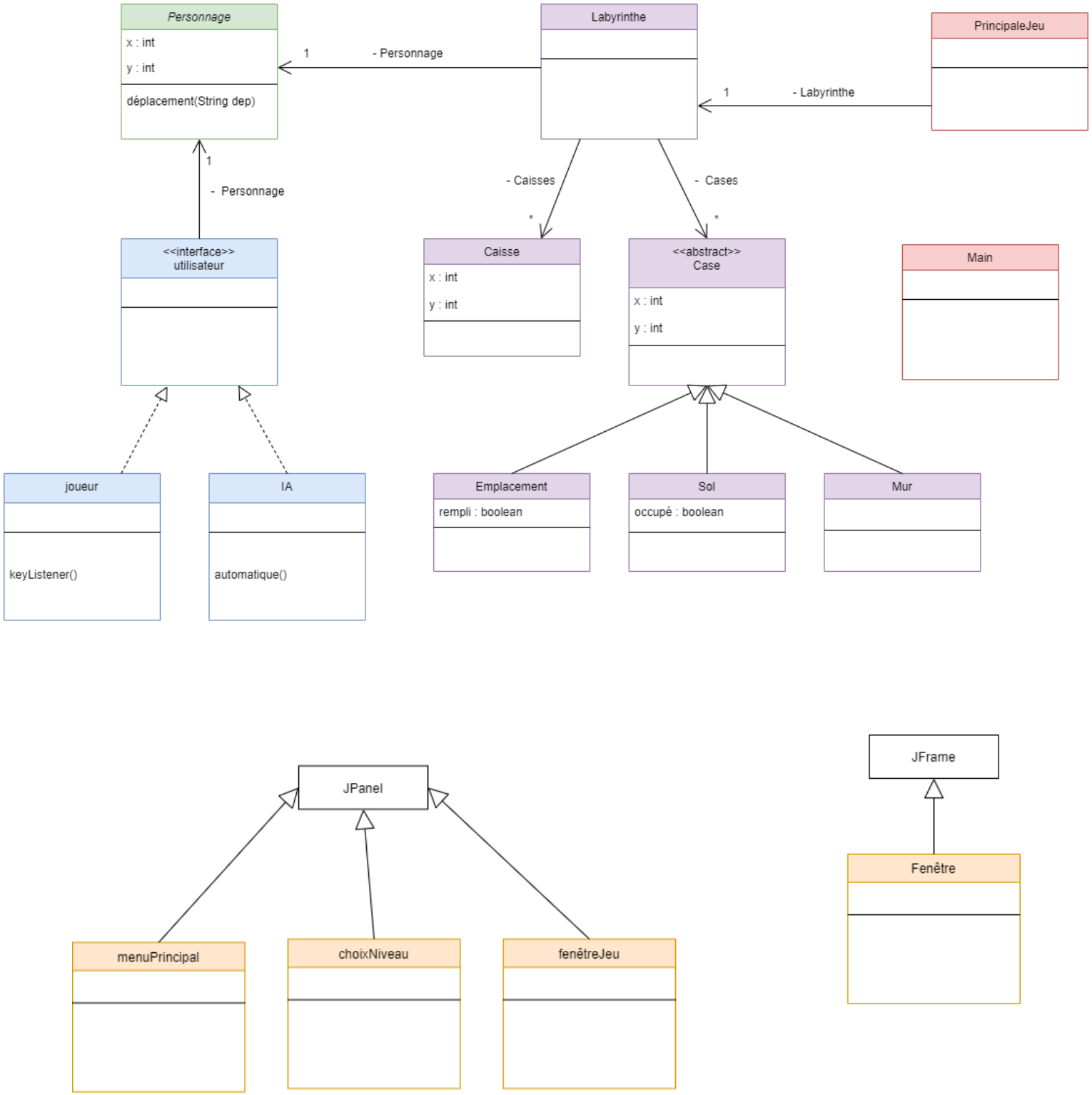
Répartition des tâches :

Nous n'avons pas encore prévu la répartition exacte des tâches de chacun des membres du groupe. Cependant, nous avons déjà réfléchi à la façon de le faire. Pour ce qui est de la création du jeu de base de l'application, nous pensons nous répartir en deux binômes : dans chaque binôme, il y aurait une personne qui coderait une classe, et une autre qui écrirait la classe de test de cette dernière. Ainsi, nous aurions quatre personnes travaillant sur quatre classes différentes, ce qui nous permettrait d'optimiser le temps de travail sur le projet.
Pour ce qui est des diagrammes (de classes et de séquence), il y aura une seule personne par diagramme. Pour la partie d'IA et la fonction heuristique, comme il s'agit principalement de réflexion, nous comptons les réaliser en groupe. Cependant, il y aurait, là encore, une seule personne par algorithme réalisé.

3.2) Attendus de fin d'itération

A la fin de l'itération, nous devrons impérativement avoir un jeu fonctionnel où le joueur pourra déplacer son personnage, pousser les caisses et finir le niveau. Nous devrons aussi avoir un algorithme de type A* rédigé et avoir réfléchi au calcul de la fonction heuristique dans le cadre du jeu Sokoban.

3.3) Premier diagramme de classes



Références

1. https://www.sokoban.jp/rule.html
2. https://gamejolt.com/games/push-crate-sokoban/56655
3. https://webdocs.cs.ualberta.ca/~games/Sokoban/program.html
4. http://informatique.umons.ac.be/ftp_infofs/2008/Hoste2008-memoire.pdf
5. https://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra
6. https://fr.wikipedia.org/wiki/Algorithme_A*
7. https://fr.wikipedia.org/wiki/Contractions_hi%C3%A9rarchiques
8. https://fr.wikipedia.org/wiki/Apprentissage_par_renforcement

Liens annexes

http://www-connex.lip6.fr/~baskiotisn/index.php/2016/10/19/sokoban-solveur/

http://www.dechelotte.com/fr/sokodan.php

https://github.com/LilianBordeau/Sokoban

http://www-connex.lip6.fr/~baskiotisn/wp/wp-content/uploads/2017/09/projet-sokoban.pdf

https://weetu.net/Timo-Virkkala-Solving-Sokoban-Masters-Thesis.pdf

Différents solveurs de Sokoban :
http://sokobano.de/wiki/index.php?title=Links

https://chamilo.univ-grenoble-alpes.fr/courses/IUT1RT1M2109/document/1718-Sokoban/build/principe.html