# Sokoban, un jeu de stratégie

Ahmat Mahamat Ahmat, Mathilde Boulland, Pierre Chretien

L1 MIASHS 2019/2020

## 1 Objectifs du projet

### 1.1 Formation du groupe et choix de l'application

Tout commence lors de la première séance par former un groupe, en sachant que peu de personnes se connaissent dans notre classe de TP. L'objectif premier est de faire connaissance et se familiariser séance après séance avec notre groupe. Celui-ci était composé de quatre personnes au départ, Ahmat Mahamat Ahmat, Mathilde Boulland, Alexandre Bertin et Pierre Chretien, mais est finalement réduit à trois personnes car Alexandre Bertin a décidé de ne plus suivre et de ne plus participer à la licence MIASHS. Plusieurs propositions d'applications nous sont proposées telles qu'un Puzzle Quest, un Block Puzzle multijoueur, un interface pour jeux de cartes, un jeu de labyrinthe à gravité, un WarGame, un jeu de Sokoban ou encore une simulation d'écosystèmes. Nous avons décidé de nous lancer dans la réalisation du jeu Sokoban car c'est celui que nous avons estimé être le plus abordable et c'est celui qui nous a le plus inspiré.

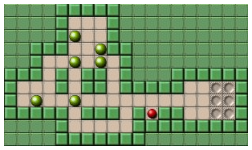
L'objectif du jeu Sokoban est de déplacer des caisses à l'aide d'un personnage et de les ranger à des cases cibles en sachant que notre personnage ne peut se déplacer que dans les quatre directions et que celui ci ne peut que pousser les caisses et non les tirer. Donc il est possible de se retrouver bloqué après un mouvement mal choisi. Notre projet consiste à réaliser un jeu Sokoban jouable pour un utilisateur sur différents niveaux qui n'ont pas la même difficulté et de permettre à l'utilisateur de comparer ses performances à celles d'une intelligence artificielle qui jouera en parallèle avec l'utilisateur. Notre première étape va consister à créer un jeu Sokoban jouable de manière simple et efficace pour un humain en lui ajoutant une interface graphique cohérente afin d'avoir un univers et de la vie dans notre application.

### 1.2 Applications similaires au Sokoban

Pour réaliser ce projet, nous nous sommes bien évidemment inspiré d'autres jeux Sokoban que l'on a trouvé sur internet. Il existe beaucoup de jeux Sokoban sur différents sites internet comme "jeu.fr", "jeu.com" ou encore "jeu.net " et nos recherches sur ces jeux nous ont permis de mixer plusieurs de leurs idées,

comme les niveaux et les décors, afin ne pas reprendre les idées d'un seul même jeu. Pour l'univers de notre jeu, nous avions décidé de reprendre les décorations, autrement appelé les "sprites", d'un jeu appelé "Mario sokoban" disponible sur le site "jeu.net".
Voici, ci-dessous, une image du jeu "Mario sokoban" :


Pour ce qui est des niveaux, nos principales inspirations se sont faites sur internet ou nous avons repris quelques idées de niveaux disponibles sur "Google image".
Voici, ci-dessous, un exemple de niveau que l'on a repris pour notre projet :



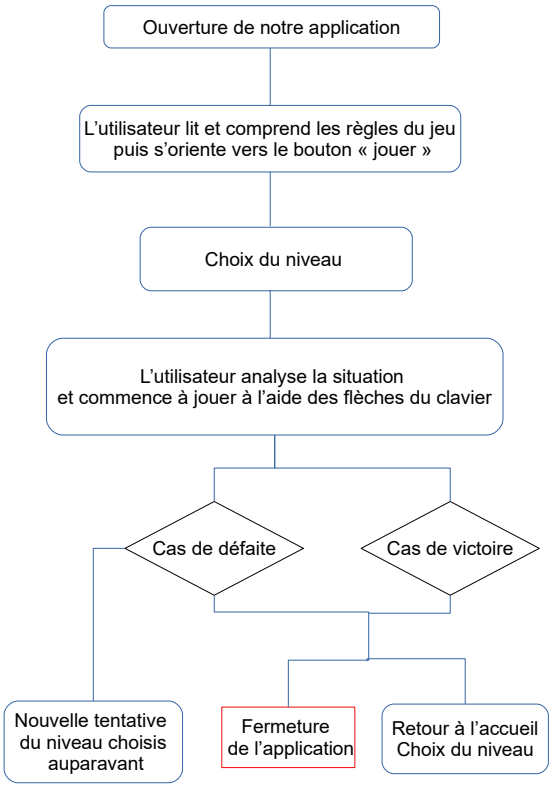
# 2 Analyse conceptuelle du projet

## 2.1 Fonctionnalités implémentées

Une fois notre application exécutée, l'utilisateur sera accueilli par un menu d'accueil. Ce menu d'accueil a pour objectif d'expliquer les règles du jeu du traditionnel jeu Sokoban à l'utilisateur et de savoir comment y jouer. Une fois ces règles comprises et maîtrisées, l'utilisateur pourra cliquer sur la rubrique "jouer" en bas de cette page.
Après que l'utilisateur ait cliqué sur la rubrique "jouer", il se retrouvera face à une nouvelle page incluant six choix de niveau et un niveau aléatoire à sa disposition. Ce niveau aléatoire sera différent des six niveaux que l'on peut choisir et sera d'une difficulté bien plus élevée. Il faut savoir que les niveaux sont dans un ordre de difficulté, c'est à dire qu'ils sont rangés du plus faible au plus difficile.
Une fois le niveau choisi, l'utilisateur se retrouvera face au jeu. Il pourra alors commencer à réfléchir et à analyser la situation afin de ne pas être bloqué dès les premiers mouvements. Ensuite, il pourra déplacer notre personnage "Mario" à l'aide des flèches du clavier et à pousser les caisses jusqu'à leurs objectifs qui sont les points verts. Lorsque l'utilisateur réussis à réunir toutes les caisses jusqu'à

leurs objectifs, c'est à dire les cases vertes, alors une nouvelle page lui proposera de cliquer sur l'une des deux rubriques suivantes : "accueil" et "quitter". Si l'utilisateur décide de cliquer sur la rubrique "accueil", alors il sera redirigé vers la page d'accueil comprenant les six choix de niveau et pourra alors refaire son choix parmi ces six possibilités. Si l'utilisateur décide de cliquer sur la rubrique "quitter", alors l'application se fermera.
En revanche, si l'utilisateur parvient à bloquer une caisse qui est synonyme de défaite, alors une nouvelle page lui proposera de retourner à l'accueil ou de fermer l'application comme expliqué précédemment ou de bien de retenter sa chance en réessayant le niveau.

Le scénario d'interaction entre l'utilisateur et notre application peut se représenter par un schéma ci-dessous :



## 2.2 Organisation du projet

Lors de la seconde séance consacrée à notre projet, notre but était de bien comprendre le fonctionnement du jeu et d'attribuer un rôle à chaque membre du groupe pour la réalisation de notre projet. Le but n'étant pas de s'isoler et de travailler chacun dans son coin, c'est pourquoi on a décidé d'attribuer des rôles complémentaires les uns des autres pour pouvoir contribuer à sa manière au projet. Deux membres de notre groupe estiment avoir des difficultés dans la programmation et le codage informatique qui sont Mathilde et Pierre, c'est pourquoi il était plus judicieux et astucieux d'attribuer la tâche de programmation principale à Ahmat et à Alexandre. Comme on l'a évoqué précédemment, Alexandre a décidé de d'arrêter la licence MIASHS et donc s'est retiré du groupe pour ce projet mais en aucun cas Ahmat sera seul à effectuer le codage. En effet Pierre et Mathilde vont se consacrer, dans un premier temps, à réfléchir au thème du jeu par rapport aux décorations, aux images, au "design" donc à l'univers qui entourera notre application et à comment procéder pour le réaliser. Nous avons décidé de nous mettre d'accord assez rapidement sur l'univers de notre application, c'est à dire d'être au point après cinq séances maximum, afin de ne pas avoir trop de retard sur la programmation du jeu. Une fois cela fait, on a tous pu se concentrer sur la programmation.
Durant toutes les séances qui ont suivi, Ahmat a programmé la plus grande partie de notre travail avec l'aide de Pierre et Mathilde mais comme on s'est aperçu d'un déséquilibre dans la répartition des tâches, Pierre et Mathilde ont décidé de réaliser le rapport et le diaporama du projet.
Voici nos numéros de téléphones :
- Chretien Pierre : 06 86 42 44 42
- Ahmat Mahamat Ahmat : 07 51 01 85 42
- Boulland Mathilde : 07 71 14 72 50

# 3 Éléments techniques du projet

Algorithme global :
Dans cette partie on va décrire les algorithmes les plus importants de notre jeu, ces algorithmes se basent sur les anciens principes du SOKOBAN des années 90. Dans les parties précédentes, on a détaillé ces principes et le mode de jeu. D'après ces principes, on constate que le jeu se résume sur le déplacement du personnage et celui des caisses. Dans cette partie on va décrire notamment l'algorithme de déplacement du personnage qui va engendrer le déplacement des caisses et deux autres algorithmes qui nous permettent de savoir si le joueur a gagné ou a perdu dans le jeu.

Remarque :
D'après les règles du jeu:
–le personnage ne peut pousser qu'une seule caisse, ne peut pas tirer les caisses et le personnage ne peut se déplacer que dans les 4 directions;

–le joueur gagne s'il a rangé toutes les caisses sur les objectifs marqués par des points verts;
–le joueur perd si une caisse se trouve bloquée et s'il n'y a pas de résolution possible.

Structures et Bibliothèques :
Les bibliothèques utilisées dans le jeu sont Random, Time et Pygame.
Les structures utilisées dans l'application et dans les algorithmes que l'ont décrira ci-dessous sont les tuples et les listes.

Algorithme du déplacement du personnage :
En entrée notre algorithme va recevoir deux choses, une liste de dimension 2 qui représentera le niveau à jouer et une variable qui représentera la direction (Droite, Gauche, Haut, Bas) du personnage. Soient Niveau cette liste et Direction la variable.
Notre niveau est constitué du personnage, personnage_ sur_ objectif, des caisses, des caisses_ sur_ objectif, de l'espace vide, des objectifs et du mur. On utilise une procédure pour cet algorithme.
Pour faire du commentaire, on utilise # comme en python et pour la différence on utilise ! = .

```
Procédure DEPLACEMENT(Niveau, Direction)
DECLARATION:
I de type nombre Entier naturel
J de type nombre Entier naturel
L de type nombre Entier naturel # L => ligne et C => colonne
C de type nombre Entier naturel

Début
    # on parcourt notre liste Niveau pour récupérer les coordonnées initiales du personnage

    Pour I allant de 1 à longueur de Niveau faire
    Début Pour
        Pour J allant de 1 à longueur de Niveau[I] faire
        Début Pour
            # on teste toutes les positions une par une pour trouver celle du personnage
            Si Niveau[I][J] = Personnage ou Niveau[I][J] = Personnage_ sur_ Objectif alors
            Début Si
                L --> I
                C --> J
            fin Si
        Fin Pour
    Fin Pour
```

```
# En sortant de la boucle, on va tester si Niveau[L][C] = Personnage
ou bien Niveau[L][C] = Personnage_ sur_ Objectif et que Niveau[L][C+1] ! = Mur
        Si Niveau[L][C] = Personnage alors
        Début Si
                # On va tester la variable Direction pour savoir dans quelle direction
                le personnage va se diriger.

                Si Direction = Droite alors
                Début Si
                        # on teste la position suivante qui est (L,C+1) pour voir si c'est possible de se déplacer.
                        # le personnage peut se déplacer si la position suivante est différente d'un Mur
                        Si Niveau[L][C+1] = Vide alors
                        Début Si
                                Niveau[L][C] -- > Vide # on remet l'ancienne position du personnage à vide
                                Niveau[L][C+1] -- > Personnage # on déplace le personnage à
                                la position (L,C+1)
                        Fin Si

                        Si Niveau[L][C+1] = Objectif alors
                        Début Si
                                Niveau[L][C] -- > Vide # on remet l'ancienne position du personnage à vide
                                Niveau[L][C+1] -- > Personnage_ sur_ Objectif # on déplace le personnage à
                                la position (L,C+1)
                        Fin Si

                        Si Niveau[L][C+1] = Caisse alors
                        Début Si
                                # On va encore tester la position suivante qui est (L,C+2).
                                # le personnage peut se déplacer si cette position est différente de Caisse,
                                Mur ou bien Caisse_ sur_ Objectif
                                Si Niveau[L][C+2] = Vide alors # le personnage et la caisse peuvent se déplacer
                                Début Si
                                        Niveau[L][C] -- > Vide # on remet l'ancienne position du
                                        personnage à vide
                                        Niveau[L][C+1] -- > Personnage # on déplace le personnage
                                        à la position (L,C+1)
                                        Niveau[L][C+2] -- > Caisse # on déplace la Caisse à la position (L,C+2)
                                Fin Si

                                Si Niveau[L][C+2] = Objectif alors # le personnage et la caisse peuvent
                                se déplacer
                                Début Si
                                        Niveau[L][C] -- > Vide # on remet l'ancienne position du
                                        personnage à vide
                                        Niveau[L][C+1] -- > Personnage # on déplace le personnage
                                        à la position (L,C+1)
```

```
                    Niveau[L][C+2] −− > Caisse_ sur_ Objectif # on déplace
                    la Caisse sur l'objectif
                Fin Si
            Fin Si
            Si Niveau[L][C+1] = Caisse_ sur_ Objectif alors

            Début Si
                # On va encore tester la position suivante qui est (L,C+2).
                # le personnage peut se déplacer si cette position est différente de Caisse,
                Mur ou bien Caisse_ sur_ Objectif
                Si Niveau[L][C+2] = Vide alors # le personnage et la caisse peuvent se déplacer
                Début Si
                    Niveau[L][C] −− > Vide # on remet l'ancienne position du
                    personnage à vide
                    Niveau[L][C+1] −− > Personnage_ sur_ Objectif # on déplace le
                    personnage à la position (L,C+1)
                    Niveau[L][C+2] −− > Caisse # on déplace la Caisse à la position (L,C+2)
                Fin Si

                Si Niveau[L][C+2] = Objectif alors # le personnage et la caisse
                peuvent se déplacer
                Début Si
                    Niveau[L][C] −− > Vide # on remet l'ancienne position du
                    personnage à vide
                    Niveau[L][C+1] −− > Personnage_ sur_ Objectif # on déplace le
                    personnage à la position (L,C+1)
                    Niveau[L][C+2] −− > Caisse_ sur_ Objectif # on déplace la Caisse
                    sur l'objectif
                Fin Si
            Fin Si
        Fin Si
```

---

```
        Si Direction = Gauche alors
        Début Si
            # on teste la position suivante qui est (L,C-1) pour voir si c'est possible de se déplacer.
            # le personnage peut se déplacer si la position suivante est différent d'un Mur
            Si Niveau[L][C+1] = Vide alors
            Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position du personnage à vide
                Niveau[L][C-1] −− > Personnage # on déplace le personnage
                à la position (L,C-1)
            Fin Si

            Si Niveau[L][C-1] = Objectif alors
            Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position du personnage à vide
```

```
        Niveau[L][C-1] −− > Personnage_ sur_ Objectif # on déplace le personnage
        à la position (L,C-1)
Fin Si

Si Niveau[L][C-1] = Caisse alors
Début Si
        # On va encore tester la position suivante qui est (L,C-2).
        # le personnage peut se déplacer si cette position est différente de Caisse,
        Mur ou bien Caisse_ sur_ Objectif
        Si Niveau[L][C-2] = Vide alors # le personnage et la caisse peuvent se déplacer
        Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position du
                personnage à vide
                Niveau[L][C-1] −− > Personnage # on déplace le personnage
                à la position (L,C-1)                                   Niveau[L][C-2] −− >
        Fin Si

        Si Niveau[L][C-2] = Objectif alors # le personnage et la caisse
        peuvent se déplacer
        Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position
                du personnage à vide
                Niveau[L][C-1] −− > Personnage # on déplace le personnage
                à la position (L,C-1)
                Niveau[L][C-2] −− > Caisse_ sur_ Objectif # on déplace
                la Caisse sur l'objectif
        Fin Si
Fin Si
Si Niveau[L][C-1] = Caisse_ sur_ Objectif alors
Début Si
        # On va encore tester la position suivante qui est (L,C-2).
        # le personnage peut se déplacer si cette position est différente de Caisse,
        Mur ou bien Caisse_ sur_ Objectif
        Si Niveau[L][C-2] = Vide alors # le personnage et la caisse peuvent se déplacer
        Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position du
                personnage à vide
                Niveau[L][C-1] −− > Personnage_ sur_ Objectif # on déplace le personnage
                à la position (L,C-1)
                Niveau[L][C-2] −− > Caisse # on déplace la Caisse à la position (L,C-2)
        Fin Si

        Si Niveau[L][C-2] = Objectif alors # le personnage et la caisse
        peuvent se déplacer
        Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position
```

```
                    du personnage à vide
                    Niveau[L][C-1] −− > Personnage_ sur_ Objectif # on déplace le
                    personnage à la position (L,C-1)
                    Niveau[L][C-2] −− > Caisse_ sur_ Objectif # on déplace
                    la Caisse sur l'objectif
                Fin Si
            Fin Si
        Fin Si
```

---

```
        Si Direction = Bas alors
        Début Si
            # on teste la position suivante qui est (L,C+1) pour voir si c'est possible de se déplacer.
            # le personnage peut se déplacer si la position suivante est différente d'un Mur
            Si Niveau[L][C+1] = Vide alors
            Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position du personnage à vide
                Niveau[L+1][C] −− > Personnage # on déplace le personnage
                à la position (L+1,C)
            Fin Si

            Si Niveau[L+1][C] = Objectif alors
            Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position du personnage à vide
                Niveau[L+1][C] −− > Personnage_ sur_ Objectif # on déplace le personnage
                à la position (L+1,C)
            Fin Si

            Si Niveau[L+1][C] = Caisse alors
            Début Si
                # On va encore tester la position suivante qui est (L+2,C).
                # le personnage peut se déplacer si cette position est différente de Caisse,
                Mur ou Caisse_ sur_ Objectif
                Si Niveau[L+2][C] = Vide alors # le personnage et la caisse peuvent se déplacer
                Début Si
                    Niveau[L][C] −− > Vide # on remet l'ancienne position
                    du personnage à vide
                    Niveau[L+1][C] −− > Personnage # on déplace le personnage
                    à la position (L+1,C)
                    Niveau[L+2][C] −− > Caisse # on déplace la Caisse à la position (L+2,C)
                Fin Si

                Si Niveau[L][C+2] = Objectif alors # le personnage et la caisse
                peuvent se déplacer
                Début Si
                    Niveau[L][C] −− > Vide # on remet l'ancienne position
                    du personnage à vide
```

Niveau[L+1][C] −− > Personnage # on déplace le personnage
à la position (L+1,C)
Niveau[L+2][C] −− > Caisse_ sur_ Objectif # on déplace la Caisse
sur l'objectif
Fin Si
Fin Si
Si Niveau[L+1][C] = Caisse_ sur_ Objectif alors
Début Si
# On va encore tester la position suivante qui est (L+2,C).
# le personnage peut se déplacer si cette position est différente de Caisse,
Mur ou Caisse_ sur_ Objectif
Si Niveau[L+2][C] = Vide alors # le personnage et la caisse peuvent se déplacer
Début Si
Niveau[L][C] −− > Vide # on remet l'ancienne position
du personnage à vide
Niveau[L+1][C] −− > Personnage_ sur_ objectif # on déplace le
personnage à la position (L+1,C)
Niveau[L+2][C] −− > Caisse # on déplace la Caisse à la position (L+2,C)
Fin Si

Si Niveau[L+2][C] = Objectif alors # le personnage et la caisse
peuvent se déplacer
Début Si
Niveau[L][C] −− > Vide # on remet l'ancienne position du
personnage à vide
Niveau[L+1][C] −− > Personnage_ sur_ Objectif # on déplace
le personnage à la position (L+1,C)
Niveau[L+2][C] −− > Caisse_ sur_ Objectif # on déplace
la Caisse sur l'objectif
Fin Si
Fin Si
Fin Si

---

Si Direction = Haut alors
Début Si
# on teste la position suivante qui est (L-1,C) pour voir si c'est possible de se déplacer.
# le personnage peut se déplacer si la position suivante est différente d'un Mur
Si Niveau[L-1][C] = Vide alors
Début Si
Niveau[L][C] −− > Vide # on remet l'ancienne position du personnage à vide
Niveau[L-1][C] −− > Personnage # on déplace le personnage
à la position (L-1,C)
Fin Si

Si Niveau[L-1][C] = Objectif alors
Début Si

```
        Niveau[L][C] −− > Vide # on remet l'ancienne position du personnage à vide
        Niveau[L-1][C] −− > Personnage_ sur_ Objectif # on déplace le personnage
        à la position (L-1,C)
Fin Si

Si Niveau[L-1][C] = Caisse alors
Début Si
        # On va encore tester la position suivante qui est (L-2,C).
        # le personnage peut se déplacer si cette position est différente de Caisse,
        Mur ou bien Caisse_ sur_ Objectif
        Si Niveau[L-2][C] = Vide alors # le personnage et la caisse peuvent se déplacer
        Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position du
                personnage à vide
                Niveau[L-1][C] −− > Personnage # on déplace le personnage
                à la position (L-1,C)
                Niveau[L-2][C] −− > Caisse # on déplace la Caisse à la position (L-2,C)
        Fin Si

        Si Niveau[L-2][C] = Objectif alors # le personnage et la caisse
        peuvent se déplacer
        Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position
                du personnage à vide
                Niveau[L-1][C] −− > Personnage # on déplace le personnage
                à la position (L-1,C)
                Niveau[L-2][C] −− > Caisse_ sur_ Objectif # on déplace la Caisse
                sur l'objectif
        Fin Si
Fin Si
Si Niveau[L-1][C] = Caisse_ sur_ Objectif alors
Début Si
        # On va encore tester la position suivante qui est (L-2,C).
        # le personnage peut se déplacer si cette position est différente de Caisse,
        Mur ou bien Caisse_ sur_ Objectif
        Si Niveau[L-2][C] = Vide alors # le personnage et la caisse peuvent se déplacer
        Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position
                du personnage à vide
                Niveau[L-1][C] −− > Personnage_ sur_ Objectif # on déplace le
                personnage à la position (L-1,C)
                Niveau[L-2][C] −− > Caisse # on déplace la Caisse à la position (L-2,C)
        Fin Si

        Si Niveau[L-2][C] = Objectif alors # le personnage et la caisse
        peuvent se déplacer
```

```
                    Début Si
                        Niveau[L][C] −− > Vide # on remet l'ancienne position
                        du personnage à vide
                        Niveau[L-1][C] −− > Personnage_ sur_ Objectif # on déplace le
                        personnage à la position (L-1,C)
                        Niveau[L-2][C] −− > Caisse_ sur_ Objectif # on déplace la Caisse
                        sur l'objectif
                    Fin Si
                Fin Si
            Fin Si
        Fin Si
```

---

```
        Si Niveau[L][C] = Personnage_ sur_ Objectif alors
        Début Si
            # On va tester la variable Direction pour savoir dans quelle direction
            le personnage va se diriger.
```

---

```
            Si Direction = Droite alors
            Début Si
                # on teste la position suivante qui est (L,C+1) pour voir si c'est possible de se déplacer.
                # le personnage peut se déplacer si la position suivante est différente d'un Mur
                Si Niveau[L][C+1] = Vide alors
                Début Si
                    Niveau[L][C] −− > Objectif # on remet l'ancienne position du personnage à vide
                    Niveau[L][C+1] −− > Personnage # on déplace le personnage
                    à la position (L,C+1)
                Fin Si

                Si Niveau[L][C+1] = Objectif alors
                Début Si
                    Niveau[L][C] −− > Objectif # on remet l'ancienne position du personnage à vide
                    Niveau[L][C+1] −− > Personnage_ sur_ Objectif # on déplace le personnage
                    à la position (L,C+1)
                Fin Si

                Si Niveau[L][C+1] = Caisse alors
                Début Si
                    # On va encore tester la position suivante qui est (L,C+2).
                    # le personnage peut se déplacer si cette position est différente de Caisse,
                    Si Niveau[L][C+2] = Vide alors # le personnage et la caisse peuvent se déplacer
                    Début Si
                        Niveau[L][C] −− > Objectif # on remet l'ancienne position du
                        personnage à vide
                        Niveau[L][C+1] −− > Personnage # on déplace le personnage
                        à la position (L,C+1)
                        Niveau[L][C+2] −− > Caisse # on déplace la Caisse à la position (L,C+2)
```

```
                Fin Si

                Si Niveau[L][C+2] = Objectif alors # le personnage et la caisse
                peuvent se déplacer
                Début Si
                    Niveau[L][C] −− > Objectif # on remet l'ancienne position du
                    personnage à vide
                    Niveau[L][C+1] −− > Personnage # on déplace le personnage
                    à la position (L,C+1)
                    Niveau[L][C+2] −− > Caisse_ sur_ Objectif # on déplace la Caisse
                    sur l'objectif
                Fin Si
            Fin Si
            Si Niveau[L][C+1] = Caisse_ sur_ Objectif alors
            Début Si
                # On va encore tester la position suivante qui est (L,C+2).
                # le personnage peut se déplacer si cette position est différente de Caisse,
                Mur ou bien Caisse_ sur_ Objectif
                Si Niveau[L][C+2] = Vide alors # le personnage et la caisse peuvent se déplacer
                Début Si
                    Niveau[L][C] −− > Objectif # on remet l'ancienne position
                    du personnage à vide
                    Niveau[L][C+1] −− > Personnage_ sur_ Objectif # on déplace le
                    personnage à la position (L,C+1)
                    Niveau[L][C+2] −− > Caisse # on déplace la Caisse à la position (L,C+2)
                Fin Si

                Si Niveau[L][C+2] = Objectif alors # le personnage et la caisse
                peuvent se déplacer
                Début Si
                    Niveau[L][C] −− > Objectif # on remet l'ancienne position
                    du personnage à vide
                    Niveau[L][C+1] −− > Personnage_ sur_ Objectif # on déplace le
                    personnage à la position (L,C+1)
                    Niveau[L][C+2] −− > Caisse_ sur_ Objectif # on déplace
                    la Caisse sur l'objectif
                Fin Si
            Fin Si
        Fin Si
-------------------------------------------------------------------------------
        Si Direction = Gauche alors
        Début Si
            # on teste la position suivante qui est (L,C-1) pour voir si c'est possible de se déplacer.
            # le personnage peut se déplacer si la position suivante est différente d'un Mur
            Si Niveau[L][C+1] = Vide alors
            Début Si
```

```
        Niveau[L][C] -- > Vide # on remet l'ancienne position du personnage à vide
        Niveau[L][C-1] -- > Personnage # on déplace le personnage
        à la position (L,C-1)
Fin Si

Si Niveau[L][C-1] = Objectif alors
Début Si
        Niveau[L][C] -- > Vide # on remet l'ancienne position du personnage à vide
        Niveau[L][C-1] -- > Personnage_ sur_ Objectif # on déplace le personnage
        à la position (L,C-1)
Fin Si

Si Niveau[L][C-1] = Caisse alors
Début Si
        # On va encore tester la position suivante qui est (L,C-2).
        # le personnage peut se déplacer si cette position est différente de Caisse,
        Mur ou bien Caisse_ sur_ Objectif
        Si Niveau[L][C-2] = Vide alors # le personnage et la caisse peuvent se déplacer
        Début Si
                Niveau[L][C] -- > Vide # on remet l'ancienne position
                du personnage à vide
                Niveau[L][C-1] -- > Personnage # on déplace le personnage
                à la position (L,C-1)
                Niveau[L][C-2] -- > Caisse # on déplace la Caisse à la position (L,C-2)
        Fin Si

        Si Niveau[L][C-2] = Objectif alors # le personnage et la caisse
        peuvent se déplacer
        Début Si
                Niveau[L][C] -- > Vide # on remet l'ancienne position
                du personnage à vide
                Niveau[L][C-1] -- > Personnage # on déplace le personnage
                à la position (L,C-1)
                Niveau[L][C-2] -- > Caisse_ sur_ Objectif # on déplace la Caisse
                sur l'objectif
        Fin Si
Fin Si
Si Niveau[L][C-1] = Caisse_ sur_ Objectif alors
Début Si
        # On va encore tester la position suivante qui est (L,C-2).
        # le personnage peut se déplacer si cette position est différente de Caisse,
        Mur ou bien Caisse_ sur_ Objectif
        Si Niveau[L][C-2] = Vide alors # le personnage et la caisse peuvent se déplacer
        Début Si
                Niveau[L][C] -- > Objectif # on remet l'ancienne position
                du personnage à vide
```

```
                    Niveau[L][C-1] −− > Personnage_ sur_ Objectif # on déplace le
                    personnage à la position (L,C-1)
                    Niveau[L][C-2] −− > Caisse # on déplace la Caisse à la position (L,C-2)
                Fin Si

                Si Niveau[L][C-2] = Objectif alors # le personnage et la caisse
                peuvent se déplacer
                Début Si
                    Niveau[L][C] −− > Objectif # on remet l'ancienne position
                    du personnage à vide
                    Niveau[L][C-1] −− > Personnage_ sur_ Objectif # on déplace le
                    personnage à la position (L,C-1)
                    Niveau[L][C-2] −− > Caisse_ sur_ Objectif # on déplace la Caisse
                    sur l'objectif
                Fin Si
            Fin Si
        Fin Si
```

---

```
        Si Direction = Bas alors
        Début Si
            # on teste la position suivante qui est (L+1,C) pour voir si c'est possible de se déplacer.
            # le personnage peut se déplacer si la position suivante est différente d'un Mur
            Si Niveau[L+1][C] = Vide alors
            Début Si
                Niveau[L][C] −− > Objectif # on remet l'ancienne position du personnage à vide
                Niveau[L+1][C] −− > Personnage # on déplace le personnage
                à la position (L+1,C)
            Fin Si

            Si Niveau[L+1][C] = Objectif alors
            Début Si
                Niveau[L][C] −− > Objectif # on remet l'ancienne position du personnage à vide
                Niveau[L+1][C] −− > Personnage_ sur_ Objectif # on déplace le personnage
                à la position (L+1,C)
            Fin Si

            Si Niveau[L+1][C] = Caisse alors
            Début Si
                # On va encore tester la position suivante qui est (L+2,C).
                # le personnage peut se déplacer si cette position est différente de Caisse,
                Mur ou Caisse_ sur_ Objectif
                Si Niveau[L+2][C] = Vide alors # le personnage et la caisse peuvent se déplacer
                Début Si
                    Niveau[L][C] −− > Objectif # on remet l'ancienne position
                    du personnage à vide
                    Niveau[L+1][C] −− > Personnage # on déplace le personnage
```

```
                à la position (L+1,C)
                Niveau[L+2][C] −− > Caisse # on déplace la Caisse à la position (L+2,C)
            Fin Si

            Si Niveau[L][C+2] = Objectif alors # le personnage et la caisse
            peuvent se déplacer
            Début Si
                Niveau[L][C] −− > Objectif # on remet l'ancienne position
                du personnage à vide
                Niveau[L+1][C] −− > Personnage # on déplace le personnage
                à la position (L+1,C)
                Niveau[L+2][C] −− > Caisse_ sur_ Objectif # on déplace la Caisse
                sur l'objectif
            Fin Si
        Fin Si
        Si Niveau[L+1][C] = Caisse_ sur_ Objectif alors
        Début Si
            # On va encore tester la position suivante qui est (L+2,C).
            # le personnage peut se déplacer si cette position est différente de Caisse,
            Mur ou Caisse_ sur_ Objectif
            Si Niveau[L+2][C] = Vide alors # le personnage et la caisse peuvent se déplacer
            Début Si
                Niveau[L][C] −− > objectif # on remet l'ancienne position
                du personnage à vide
                Niveau[L+1][C] −− > Personnage_ sur_ objectif # on déplace le
                personnage à la position (L+1,C)
                Niveau[L+2][C] −− > Caisse # on déplace la Caisse à la position (L+2,C)
            Fin Si

            Si Niveau[L][C+2] = Objectif alors # le personnage et la caisse
            peuvent se déplacer
            Début Si
                Niveau[L][C] −− > Objectif # on remet l'ancienne position
                du personnage à vide
                Niveau[L+1][C] −− > Personnage_ sur_ Objectif # on déplace le
                personnage à la position (L+1,C)
                Niveau[L+2][C] −− > Caisse_ sur_ Objectif # on déplace la Caisse
                sur l'objectif
            Fin Si
        Fin Si
    Fin Si
```

---

```
    Si Direction = Haut alors
    Début Si
        # on teste la position suivante qui est (L-1,C) pour voir si c'est possible de se déplacer.
        # le personnage peut se déplacer si la position suivante est différente d'un Mur
```

```
Si Niveau[L-1][C] = Vide alors
Début Si
        Niveau[L][C] −− > Vide # on remet l'ancienne position du personnage à vide
        Niveau[L-1][C] −− > Personnage # on déplace le personnage
        à la position (L-1,C)
Fin Si

Si Niveau[L-1][C] = Objectif alors
Début Si
        Niveau[L][C] −− > Vide # on remet l'ancienne position du personnage à vide
        Niveau[L-1][C] −− > Personnage_ sur_ Objectif # on déplace le personnage
        à la position (L-1,C)
Fin Si

Si Niveau[L-1][C] = Caisse alors
Début Si
        # On va encore tester la position suivante qui est (L-2,C).
        # le personnage peut se déplacer si cette position est différente de Caisse,
        Mur ou bien Caisse_ sur_ Objectif
        Si Niveau[L-2][C] = Vide alors # le personnage et la caisse peuvent se déplacer
        Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position
                du personnage à vide
                Niveau[L-1][C] −− > Personnage # on déplace le personnage
                à la position (L-1,C)
                Niveau[L-2][C] −− > Caisse # on déplace la Caisse à la position (L-2,C)
        Fin Si

        Si Niveau[L-2][C] = Objectif alors # le personnage et la caisse
        peuvent se déplacer
        Début Si
                Niveau[L][C] −− > Vide # on remet l'ancienne position
                du personnage à vide
                Niveau[L-1][C] −− > Personnage # on déplace le personnage
                à la position (L-1,C)
                Niveau[L-2][C] −− > Caisse_ sur_ Objectif # on déplace la Caisse
                sur l'objectif
        Fin Si
Fin Si

Si Niveau[L-1][C] = Caisse_ sur_ Objectif alors
Début Si
        # On va encore tester la position suivante qui est (L-2,C).
        # le personnage peut se déplacer si cette position est différente de Caisse,
        Mur ou bien Caisse_ sur_ Objectif
        Si Niveau[L-2][C] = Vide alors # le personnage et la caisse peuvent se déplacer
```

Début Si
Niveau[L][C] −− > Objectif # on remet l'ancienne position
du personnage à vide
Niveau[L-1][C] −− > Personnage_ sur_ Objectif # on déplace le
personnage à la position (L-1,C)
Niveau[L-2][C] −− > Caisse # on déplace la Caisse à la position (L-2,C)
Fin Si

Si Niveau[L-2][C] = Objectif alors # le personnage et la caisse
peuvent se déplacer
Début Si
Niveau[L][C] −− > Objectif # on remet l'ancienne position
du personnage à vide
Niveau[L-1][C] −− > Personnage_ sur_ Objectif # on déplace le personnage
à la position (L-1,C)
Niveau[L-2][C] −− > Caisse_ sur_ Objectif # on déplace la Caisse
sur l'objectif
Fin Si
Fin Si
Fin Si
Fin Si
Fin

---

ALGORITHME DE VICTOIRE:

---

Dans cette algorithme va utiliser trois variables, l'une va stocker le nombre de caisses, l'autre le nombre des objectifs et une variable booléenne qui va stocker soit 0, soit 1 ( 0−− > faut et 1 −− > vrai ).
Dans cette algorithme on utilise une fonction pour faire les opérations et cette fonction va retourner la variable booléenne.
En entrée la fonction a besoin d'une liste qui représente le niveau à jouer.

FONCTION victoire(Niveau)
DECLARATION:
I de type nombre Entier naturel
J de type nombre Entier naturel
Nombre_ Caisse de type entier et Nombre_ Caisse −− > 0
Nombre_ Objectif de type entier et Nombre_ Objectif −− > 0
Gagner de type booléen et Gagner −− > 0
Début
# on parcourt notre liste Niveau

Pour I allant de 1 à longueur de Niveau faire
Début Pour
Pour J allant de 1 à longueur de Niveau[I] faire
Début Pour

```
                # on teste toutes les positions une par une pour trouver celles des caisses et Objectifs
                Si Niveau[I][J] = Caisse alors
                Début Si
                    Nombre_ Caisse −− > Nombre_ Caisse + 1
                fin Si

                Si Niveau[I][J] = Objectif alors
                Début Si
                    Nombre_ Objectif −− > Nombre_ Objectif + 1
                fin Si
            Fin Pour
        Fin Pour
        # A la sortie de la boucle, on vérifie que si le Nombre_ Objectif = 0 et Nombre_ Caisse = 0
        alors le joueur gagne.
        Si Nombre_ Objectif = 0 et Nombre_ Caisse = 0 alors
        Début Si
            Gagner −− > 1
        Fin Si
        # A la sortie de la fonction, on retourne la variable Gagner
        Retourne(Gagner)
Fin
```

---

ALGORITHME D'ECHEC:

---

Cet algorithme à besoin en entrée une liste et donne en sorti retourne une variable booléenne.
Dans cet algorithme on vérifie à chaque moment si une caisse se trouve bloquée ou non.

```
FONCTION échec(Niveau)
DECLARATION:
        I de type nombre Entier naturel
        J de type nombre Entier naturel
        Echec de type booléen et Gagner −− > Faux
Début
        # on parcourt notre liste Niveau

        Pour I allant de 1 à longueur de Niveau faire
        Début Pour
            Pour J allant de 1 à longueur de Niveau[I] faire
            Début Pour
                # on teste toutes les positions une par une pour trouver celles des caisses
                Si Niveau[I][J] = Caisse alors
                Début Si
                # si une caisse se trouve bloquée dans un coin formé par deux murs
                    Si Niveau[I][J+1] = Mur et
                    (Niveau[I+1][J] = Mur ou Niveau[I-1][J] = Caisse) alors
```

```
                    Début si
                        ECHEC −− > Vrai
                    Fin Si
                    Si Niveau[I][J-1] = Mur et
                    (Niveau[I+1][J] = Mur ou Niveau[I-1][J] = Caisse) alors
                    Début si
                        ECHEC −− > Vrai
                    Fin Si
                    Si une caisse forme un carré avec d'autres caisses
                    ou de Mur ou bien Caisse_ sur_ Objectifs alors
                    Début si
                        ECHEC −− > Vrai
                    Fin Si
                    Si une caisse se trouve bloquée dans une ligne
                    et que le personnage ne peut pas la sortir alors
                    Début si
                        ECHEC −− > Vrai
                    Fin Si
                fin Si
            Fin Pour
        Fin Pour

        RETOURNE ( ECHEC )
Fin
```

# 4 Architecture du projet

## 4.1 Réalisation technique

Dans cette partie on va détailler la réalisation technique, l'implémentation de notre application et les étapes à suivre par l'utilisateur pour pouvoir utiliser correctement l'application, donc dans cette partie on ne va pas vraiment détailler les algorithmes et les fonctions car on les a bien détaillé dans la partie 3.
Pour réaliser ce fameux jeu de Sokoban, on a partagé le code dans trois différents fichiers python, un fichier nommé "classe.py" dans lequel on a utilisé une classe "Game" et certaines fonctions( Déplacement des objets, dessin des objets...) pour réaliser l'application, un fichier " CONSTANTES.py" que l'on ne va pas trop détailler car ce fichier ne contient que quelques constantes telles que les objets(Mur, PLayer, CAISSE...) d'un niveau, les couleurs, les tailles des objets et un dernier fichier "main1.py" qui est le fichier principale de notre application. Dans cette partie, on va plutôt détailler uniquement le fichier "main1.py" car c'est le principal pour relier tout le code. En ce qui concerne le fichier "classe.py", on a tout détaillé dans le fichier lui même avec des commentaires et aussi dans la partie 3.
Pour les graphismes on a utilisé Pygame. Pour pouvoir lancer l'application et

jouer le jeu, l'utilisateur doit appeler le fichier principale "main1.py" et non les autres fichiers. Dans les lignes qui suivent, on va décrire brièvement le contenu du fichier "main1.py" avant d'illustrer les étapes que l'utilisateur doit suivre pour pourvoir jouer.
Le fichier "main.py" est composé de trois grandes parties, une partie pour les importations des bibliothèques et des modules, une partie pour les initialisations de certaines variables et bibliothèques et une dernière partie pour les boucles qui permettent de garder l'application active.

PARTIE 1 : Importation

Dans cette partie on va importer la bibliothèque pygame pour l'utiliser ensuite dans les deux autres parties pour faire des animations. On importe aussi le module time et random qu'on utilisera dans la partie des boucles pour interrompre le programme pour quelques secondes et pour choisir des nombres aléatoirement. Pour pouvoir utiliser toutes les constantes et les fonctions de nos deux fichiers, on importera les fichiers "classe.py" et "CONSTANTES.py".

PARTIE 2 : Initialisation

Cette partie est réservée uniquement pour l'initialisation de la bibliothèque Pygame et de certaines variables très indispensables pour la troisième partie telles que les variables booléennes qui servent de conditions à nos boucles et deux autres variables pour télécharger l'image et la musique.

PARTIE 3 : Boucles

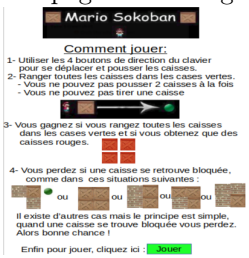
C'est la partie la plus compliquée à coder du fichier "main.py", elle est composée d'une boucle principale nommée "Game" qui va contenir 4 autres boucles. On va pas détailler tout le code qui compose ces boucles mais on détaillera la fonction de chaque boucle. Au début de chaque boucle on a une condition qui doit être remplie pour pouvoir entrer dans la boucle et exécuter le code de la boucle. On a mis cette condition pour ne pas créer des fenêtres quand l'utilisateur fermera l'application et c'est aussi dans ces conditions que l'on a créé nos fenêtres. Pour chaque boucle on a défini une variable booléenne et cette variable va représenter la boucle dans nos explications.
Une fois qu'on est dans la boucle principale, nous allons jouer la musique puis on entre dans la boucle guide, cette boucle a pour rôle de garder active la fenêtre de l'écran d'accueil dans laquelle on explique les principes du jeu et les boutons à utiliser pour jouer, c'est une sorte de guide. C'est dans cette boucle qu'on appellera notre Class "Game" avec tous ses attributs, méthodes et certaines fonctions du fichier "classe.py".
Puis la boucle "Home" qui permet d'ouvrir une fenêtre à l'utilisateur en lui proposant plusieurs niveaux, c'est la boucle qui permet de garder active la fenêtre de choix de niveau.
Ensuite, dès que l'utilisateur choisit un niveau, on entre dans la boucle "play".

Comme son nom l'indique, cette boucle permet de garder l'écran du jeu active pour permettre à l'utilisateur de jouer. Enfin, la dernière boucle nommée "Win_ or_ lose" qui s'occupe de la victoire ou de la défaite de l'utilisateur dans le jeu. En cas de victoire, cette boucle va générer une fenêtre avec deux options pour permettre à l'utilisateur de revenir à la fenêtre de choix de niveau ou bien de quitter l'application. En cas d'échec, en plus des deux autres options, l'utilisateur a une autre option pour pouvoir réessayer le niveau actuel.

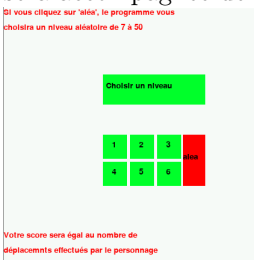
## 4.2 Chaine de traitement

Dans cette partie, nous allons parler de toute la séquence de traitement qu'un utilisateur fait une fois sur notre application. Cette chaine de traitement sera illustrée par des images.

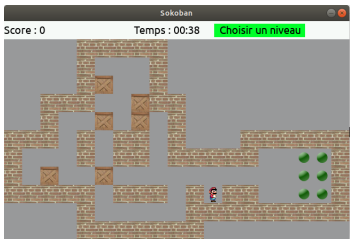
Une fois que l'utilisateur aura lancé l'application, une fenêtre va s'ouvrir accompagné de l'image ci-dessous :




Cette fenêtre explique les principes du jeu, si l'utilisateur clique sur le rectangle vert accompagné du texte "jouer" alors une nouvelle fenêtre s'ouvrira et sera accompagnée de l'image ci-dessous :



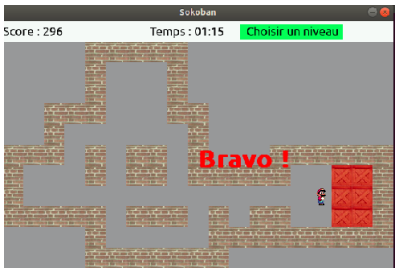

Dans cette fenêtre l'utilisateur a plusieurs options pour choisir un niveau en choisissant un des six premiers nombres entiers ou bien de choisir un niveau aléatoirement entre le niveau 7 au niveau 50 en cliquant sur "Aléa".
Supposons que l'utilisateur ait choisi le niveau 1, alors c'est cette fenêtre qui s'ouvrira comme l'indique l'image suivante :




En haut à droite de la fenêtre, il y a une option représentée par un rectangle vert accompagné du texte "Choix de niveau", si l'utilisateur clique dessus alors on le renvoie vers la fenêtre de choix de niveau pour choisir un autre niveau. Après cette fenêtre on a deux cas, soit l'utilisateur a gagné soit il a commis un échec.
En cas d'échec, c'est l'image suivante qui s'affichera :


Puis après 3 secondes, une autre fenêtre s'ouvrira avec l'image suivante :


Dans cette fenêtre l'utilisateur a trois options, soit de quitter l'application, soit de réessayer le niveau actuel ou bien de retourner à la fenêtre de choix de niveau en cliquant sur "Accueil".
En cas de victoire, c'est l'image suivante qui s'affichera :



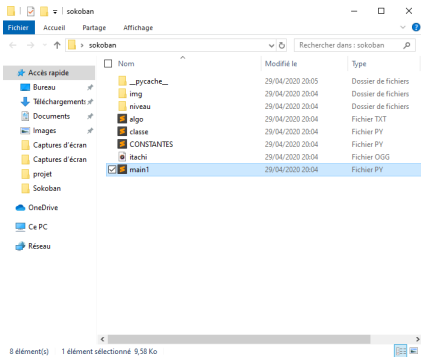

Puis après 3 secondes, une autre fenêtre s'ouvrira avec l'image suivante :


Dans cette fenêtre l'utilisateur a deux options, soit de quitter l'application, soit de retourner a la fenêtre de choix de niveau en cliquant sur "Accueil".

# 5 Manuel d'utilisation de l'application

## 5.1 Mise en place et lancement

Tout d'abord, avant de pouvoir faire l'expérience de notre application, l'utilisateur devra télécharger python et Pygame sur son ordinateur ainsi que notre dossier comprenant tout le codage informatique de notre projet.
Le dossier de notre projet se présente ainsi :




Afin de pouvoir lancer notre application, l'utilisateur devra rentrer notre fichier python appelé " main1 " sur python car c'est ce fichier qui contient notre programme principal du jeu. Ensuite, il va devoir taper sur la touche F5 du clavier pour que l'ordinateur puisse lire le programme.
Une fois l'application lancée, l'utilisateur sera accueilli par notre menu d'accueil et d'une bande son. Cette bande son est infinie donc elle l'accompagnera, peu importe la situation dans laquelle il se trouvera dans notre jeu. Notre menu d'accueil se compose principalement des règles détaillées de notre jeu. Ces règles sont définies en 4 étapes de manière précise et sont illustrées par des décors et des situations que l'on peut retrouver dans le jeu actif afin de plonger l'utilisateur dans le vif du sujet. Une fois avoir compris les règles du jeu, l'utilisateur pourra cliquer sur la rubrique "jouer" en bas de notre menu d'accueil mais attention il ne pourra pas revenir en arrière donc si par malheur il lui manque un détail concernant le jeu, il devra fermer l'application et la relancer comme expliqué précédemment.
Notre menu d'accueil est présent ci-dessous :


Après avoir cliqué sur la rubrique "jouer", une nouvelle page se présentera devant l'utilisateur. Il est maintenant confronté à choisir le niveau dans lequel il veut s'exercer. L'utilisateur a le choix entre 6 niveaux et un niveau aléatoire totalement différent des six niveaux que l'on peut choisir. Ce niveau aléatoire

est un niveau parmi 44 niveaux disponibles et est d'une difficulté bien plus élevée que les 6 premiers niveaux à disposition. Il faut savoir que les 6 premiers niveaux sont dans un ordre de difficulté, c'est à dire qu'ils sont rangés du plus faible au plus difficile. Par exemple, le niveau 1 sera plus facile à terminer que le niveau 2, le niveau 2 sera plus facile à terminer que le niveau 3 etc... Cet ordre de difficulté a été réalisé selon nos avis et nos compétences donc il se peut très bien qu'un utilisateur soit plus à l'aise sur un niveau supérieur à un autre. Il serait préférable pour l'utilisateur de commencer par le niveau 1 afin de manier notre application à la perfection !
Il est indiqué aussi sur cette page notre système de score, chaque déplacement effectué augmentera votre score de 1. Donc plus le score sera faible et plus l'utilisateur sera fort ! Nous avons voulu faire une présentation simple et efficace de cette page afin de ne pas être trop dépasser par le temps, c'est pourquoi cette page est simplement composée de niveaux instaurés dans des cases vertes avec le niveau aléatoire instauré dans une case rouge ainsi que des indications de couleur rouge. Tous ces éléments sont déposés sur un fond blanc afin de les mettre en valeur.
Voici cette page ci-dessous :


## 5.2 Jouabilité et limites

Une fois le niveau choisi, l'utilisateur va se retrouver face au jeu. C'est alors qu'une phase d'analyse va commencer. En effet, avant tout déplacement, l'utilisateur sera contraint à réfléchir sur le choix de ses actes et de ses mouvements car chaque mouvement peut être synonyme de défaite. Cette phase d'analyse va poursuivre l'utilisateur durant tout son parcours lorsqu'il effectuera un niveau car il faudra être vigilant et attentif pour avoir le mérite de finir vainqueur. Un chronomètre ainsi que le score sera à la disposition de l'utilisateur durant cette phase de jeu afin de connaître le résultat de sa performance s'il parvient à réussir le niveau. Il aura aussi la possibilité de changer de niveau si le niveau qu'il a choisi auparavant ne lui convient pas où s'il le trouve très difficile à accomplir.
Une fois le niveau analysé, l'utilisateur va pouvoir déplacer le personnage "Mario" verticalement et horizontalement à l'aide des flèches du clavier. Ce personnage ne peut se déplacer que de case en case tout comme les objets à déplacer. On

ne peut maintenir une direction choisie pour faire avancer le personnage de plusieurs cases, chaque déplacement doit être justifié par un choix de direction. Chaque mouvement augmentera le score de 1, c'est à dire que ce score augmentera de 1 à chaque fois que l'utilisateur pressera l'une des flèches du clavier. Pour pousser une caisse le principe est très simple, il suffit de déplacer le personnage "Mario" à côté de celle-ci et de la pousser dans la direction souhaitée. L'utilisateur ne peut tirer une caisse alors il faut faire attention aux rebords et aux coins. En effet, si par malheur l'utilisateur décide de pousser une caisse de sorte à ce qu'elle touche un rebord, ses déplacements vont se retrouver limités et la défaite peut être présente plus vite que prévue. Comme c'est le cas dans cette situation ci-dessous :


Sur cette situation, le cas de défaite est inévitable même si le jeu ne le fait pas savoir. C'est le seul point qui, à nos yeux, fait défaut sur notre application. Ce défaut pourrait être une légère perte de temps à l'utilisateur qui tenterait tant bien que mal de réussir le niveau alors qu'il a déjà échoué dans sa mission. Nous pensons tout de même que ce défaut ne va pas pour autant gâcher toute l'expérience de jeu de l'utilisateur. Dans un autre cas de défaite un peu plus concret, comme la caisse qui est bloquée dans un coin, le jeu va s'arrêter et peu de temps après (trois secondes), l'application va proposer à l'utilisateur de retenter sa chance ou bien de revenir à l'accueil ( sur le choix des niveaux ) ou encore de fermer l'application.
Si, par chance ou par talent, l'utilisateur parvient à ranger toutes les caisses jusqu'à leurs objectifs, l'application lui annoncera sa gloire et peu de temps après lui proposera de retourner à l'accueil afin qu'il puisse tenter de réussir un autre niveau ou bien de fermer l'application. Il n'aura pas beaucoup de temps, très exactement trois secondes, pour retenir son score et le temps épuisé à la réussite du niveau.
Voici ce qui arrive quand l'utilisateur parvient à gagner :


Une fois que l'utilisateur ait appréhendé et fait l'expérience des six premiers niveaux ainsi que de plusieurs niveaux appartenant à la catégorie du niveau aléatoire, nous pensons qu'il n'y aura plus de secret entre notre application et l'utilisateur. Son but sera alors d'améliorer son score et d'améliorer son temps sur chacun des niveaux qui lui sont proposés afin de devenir un expert de notre jeu Sokoban !

# 6 Conclusion du projet

Pour conclure toute la réalisation de notre projet, qui nous a accompagné durant 4 mois, nous pouvons dire que ce fût une belle expérience. En effet, cette période de travail nous a beaucoup appris à savoir comment fonctionne un travail de groupe sur, ce qui est pour nous, un travail d'un assez long terme. Il a fallu gérer de nombreuses situations comme la perte d'un de nos camarades dans le groupe, les absences que l'on a pu avoir lors des séances de travaux, s'organiser dans le temps, nos difficultés à réaliser telle ou telle tâche ou encore le fait que l'on ne se connaissait pas couramment lors de la création de notre groupe. Mais nous sommes très satisfaits du contenu de notre projet, notamment sur le codage qui est la partie principale de notre projet mais aussi sur le rendu final de notre application. Notre jeu réunit correctement tous les éléments d'un jeu Sokoban " classique " et nous le trouvons très complet et très plaisant pour la personne qui en fera l'expérience.