HELHa
Haute École Louvain en Hainaut

Sokoban

Giorgio Caculli LA196672, Guillaume Lambert LA198116, Tanguy Taminiau LA199566, Nathan Thaon LA188132
Groupe B01

2 décembre 2021

ÉCONOMIQUE

Table des matières

1 Description générale de l'existant

1.1 Le Sokoban original

Sokoban est un jeu vidéo japonais de type puzzle sorti en 1986. Le but est de déplacer des caisses à des emplacements précis en les poussant grâce à notre petit personnage. Une fois toutes les caisses en place le niveau s'achève et un nouveau commence.

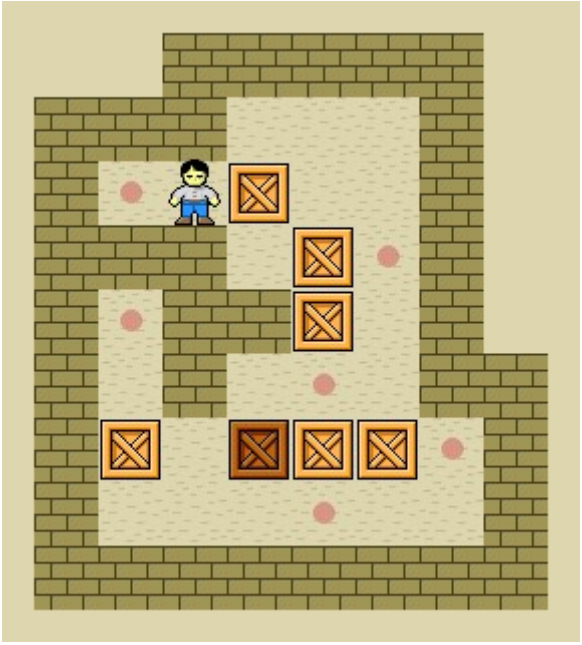

FIGURE 1 – Sokoban original de 1986

1.2 Slimoban

Slimoban est un jeu dérivé du Sokoban original, dans cette version, il s'agit toujours d'un jeu de puzzle mais cette fois, le but est de récupérer une pièce d'or en résolvant les énigmes jusqu'à la pièce d'or.

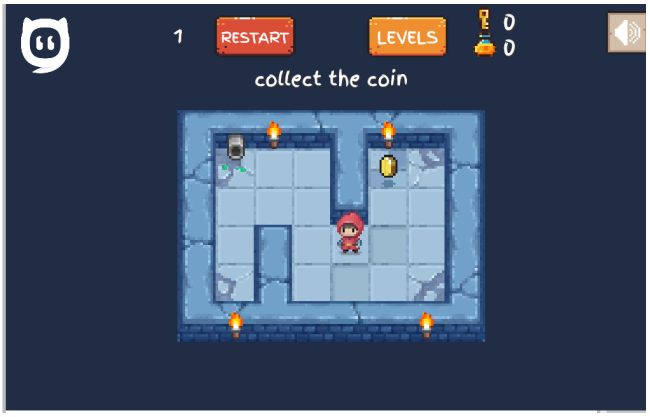

FIGURE 2 – Slimoban

1.3 BlupiMania

BlupiMania est également un jeu dérivé du Sokoban original mais dans celui-ci, le jeu est en 3D, de plus le jeu est agrémenté de nouveaux objets (tels que des accélérateurs, détonateurs, des trous, ...) modifiant ainsi la façon de jouer.

FIGURE 3 – BlupiMania

2 Description générale de l'application

Notre jeu sera un jeu de type puzzle. Il sera en vue 2D, vue du haut. Le but du jeu sera d'atteindre un objectif, en se frayant un chemin via la résolution d'un puzzle. La mécanique principale de ce jeu sera de pouvoir pousser une caisse pour nous permettre d'atteindre notre objectif qui sera soit de pousser ces caisses sur certains points pour finir le niveau, soit de se frayer un chemin via les caisses pour permettre au personnage d'atteindre son objectif et de finir le niveau. Les mouvements du personnage et des caisses se feront en case par case et les caisses ne pourront pas être poussées deux par deux. Il y aura aussi la présence d'un compteur de mouvements, de poussées de caisses, un chronomètre et un compteur de vie. De plus, il y aura plusieurs niveaux, variant au niveau de la difficulté. Il pourra aussi y avoir des ennemis qui nous font perdre de la vie et nous repoussent en rentrant dans leurs champs de vision et qu'on pourra éliminer en allant dans leurs dos. Enfin, le jeu alternera les mécaniques précédemment expliquées dans ces niveaux (un niveau où il faut atteindre un objectif suivi d'un niveau où les caisses doivent atteindre un certain point pour finir sur un niveau où il faut s'échapper d'ennemis, par exemple).

3 Product Backlog

US-01	En tant qu'utilisateur je voudrais reset une partie.
US-02	En tant qu'utilisateur je voudrais mettre en pause la partie.
US-03	En tant qu'utilisateur je voudrais sauvegarder une partie.
US-04	En tant qu'utilisateur je voudrais charger une partie.
US-05	En tant qu'utilisateur je voudrais une musique de fond.

4 Spécification technique

4.1 GUI : SFML

Figure 4 – Logo SFML

SFML est une librairie qui donne accès à une vaste variété de fonctionnalités purement écrites en C++. Les cinq fonctionnalités dont nous disposons sont les gestions suivantes :

— Toute interaction avec le système d'exploitation
— Fenêtrage
— Graphismes
— Son
— Réseau

SFML permet le cross-platforming, soi-disant, un logiciel codé avec SFML aura le même visuel indépendamment du système d'exploitation sur lequel le jeu tourne.

4.2 Librairies

Figure 5 – Logo Boost

La librairie de logging que nous utiliserons se nomme Boost. En quelques mots, la librairie Boost est elle-même un ensemble de librairies permettant d'étendre les fonctionnalités de C++. Dans notre cas, nous utiliserons les fonctionnalités prédéfinies de Boost.Log, qui nous donne accès à la possibilité d'enregistrer les différentes interactions qui ont eu lieu lors de l'exécution du jeu.

4.3 Fichiers

Les niveaux et les sauvegardes seront stockés dans des fichiers purement textuels. Ces fichiers ne stockeront que le design des niveaux ou de la partie en cours. Comme déclaré précédemment, Boost est un ensemble de librairies, dans cet ensemble il existe la librairie Boost.JSON. Grâce à cette librairie, nous serons capable de stocker des informations en format JSON, comme par exemple, une liste des scores.

4.4 OS

Les systèmes d'exploitation sur lesquels nous testerons notre jeu sont les suivants :

— Linux
— Mac OSX
— MS Windows 10

Étant donné que toutes les librairies citées précédemment sont cross-platform, nous pouvons assurer que notre jeu sera entièrement cross-platform.

5 Diagramme de classe

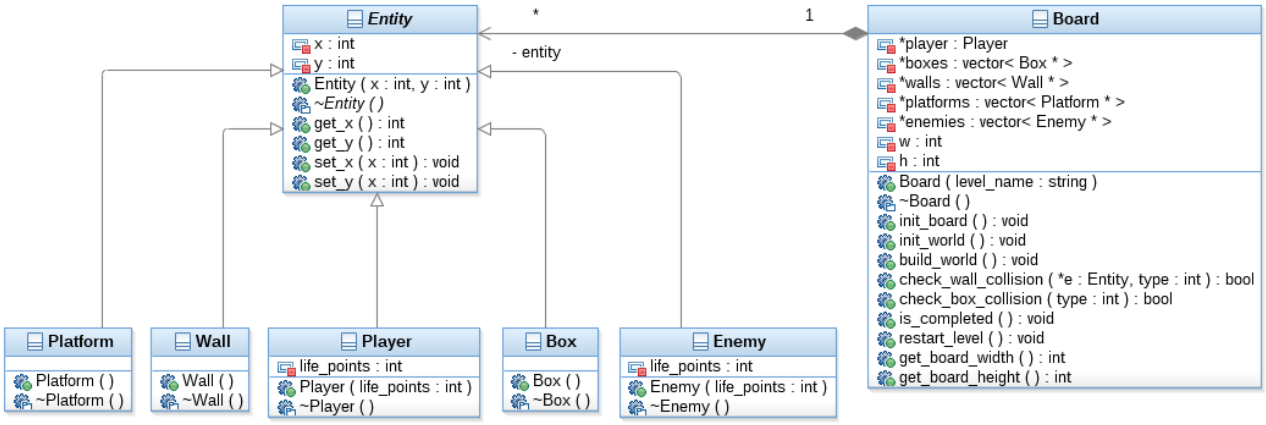

FIGURE 6 – UML du jeu

6 Planification

08/10/2021	Implémenter les modèles des objets, personnages.
15/10/2021	Réaliser un menu principal permettant de lancer la première interface du jeu, de quitter le jeu , ... et des animations du personnage et des objets et implémenter le design du niveau.
22/10/2021	Réaliser un niveau fonctionnel (implémentation des modèles dans le niveau + interaction du personnage avec les caisses + possibilité de finir le niveau).
29/10/2021	Affinage de l'interface en jeu en rajoutant le timer, compteur de vie, ... + implémenter des ennemis et dangers mettant en péril la vie du joueur.
05/11/2021	Implémenter les trois types de niveaux (pousser des caisses sur des points + atteindre un objectif + échapper à des ennemis).
12/11/2021	Réaliser une série de niveaux, avec une difficulté moindre dans les premiers niveaux qui s'accroit au fur et à mesure qu'on avance dans ceux-ci.
19/11/2021	Affinage du menu principal + menu pause dans les niveaux avec possibilité d'accéder aux options, de revenir au jeu et de quitter le jeu + rajout de musiques lors des niveaux et du menu principal + implémentation d'un $\text{step}_s ounddesign(choixdanslesmenus, poussedescaisses, findesniveaux, dgtsreus, \ldots).Derniresmodifica$