***Projet d’informatique :***

***Baba Is You***

******

**(Arnaud De Lena)**

**Romain Eloy**

**Julien Ladeuze**

**2022**

**Bachelier année 1, sciences de l’informatiques**

**Tables des matières**

***1.Introduction (page 3)***

-présentation du jeu

-répartition des tâches

***2.Structucture du projet***

**2.1.Model**

-fonctionnalités de bases

- principe OO\* utilisé

- structure de donnée utilisé

- complexité globale

**2.2.View**

- système de fenêtre utilisé

- ???

**2.3.Presenter**

- liens effectués entre le Model et le Présenter

**2.4.Ressources**

Hiérarchie des fichiers dans ressources

***3.Ajouts apportés***

***4.Difficultés rencontrées***

***5.Points forts et Points faibles***

***6.Erreurs connues***

***7.Mini guide***

***8.Conclusion***

***9.Bibliographie***

***10.Remerciements***

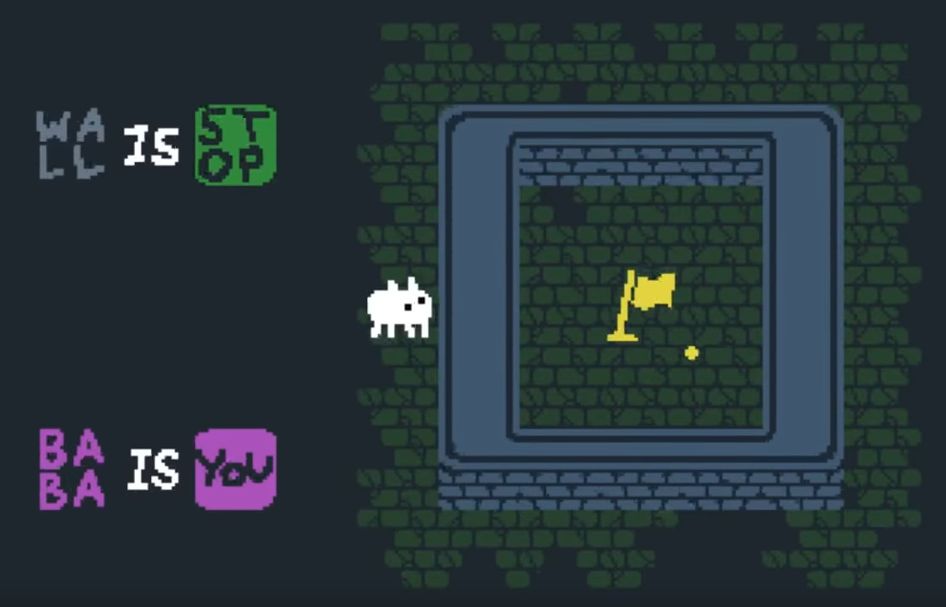
\*Orienté Objet

*1.INTRODUCTION*

Présentation du jeu

Dans le cadre de notre cours « projet d’informatique », il nous est demandé de reproduire un jeu vidéo. Ce jeu a été fait pour la première fois durant le Nordic Games Jam (Un endroit où les développeurs peuvent exprimer leur talent sur des jeux vidéo) de 2017 où il a d’ailleurs remporté le prix. Ce jeu est « Baba Is You ». Un jeu vidéo de style « Sokoban\*» puzzle/casse-tête. La particularité de ce jeu est que les règles se trouve elle-même dans le niveau et peuvent être déplacés afin de réussir le niveau. Ceci est l’essence même du jeu.

A noter que des fonctionnalités de bases devaient être implémentés : chargement d’une carte à partir d’un fichier texte, système de sauvegarde, implémentation des 4 premiers niveaux de la version jam\*



Répartition des tâches

-Romain : partie interface graphique

-Julien : partie logique + rédaction

-Arnaud : absent (dernier commit sur github : 01/03/2022)

\*https://fr.wikipedia.org/wiki/Sokoban

\*https://hempuli.itch.io/baba-is-you

*2.Structure du projet*

Lors du cadre de notre projet, nous avons eu un cours concernant l’architecture MVP

(Model View Presenter). Nous avons décidé d’adopter cette architecture car elle convenait bien pour l’implémentation de l’application. Les parties du MVP seront présentés dans les points suivants.

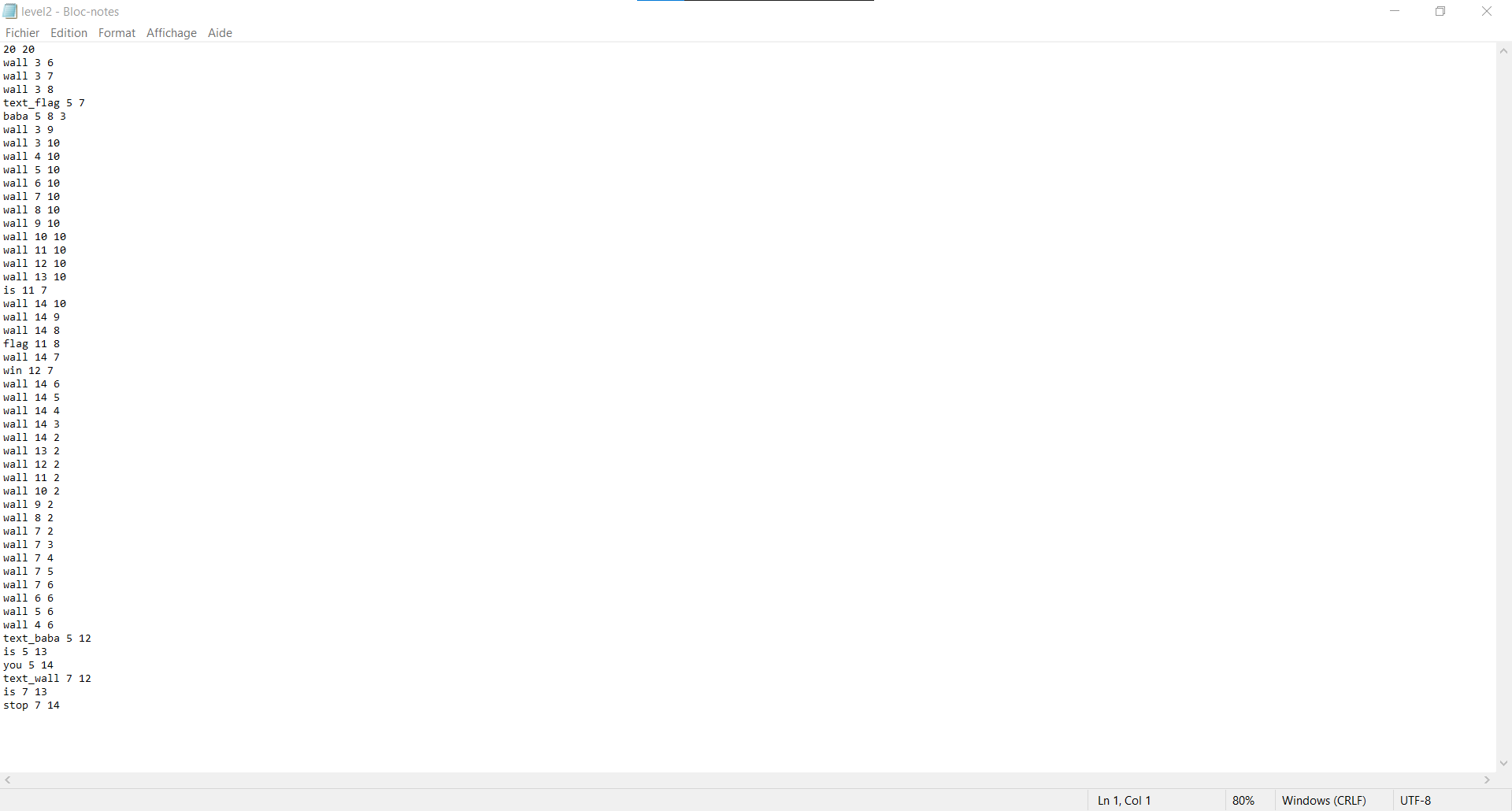
2.1.Model

La partie Model correspond à la « machinerie » de l’application. C’est ici que le travail de données va se jouer.

Fonctionnalités de bases

Niveau = fichier texte :

Comme dit dans l’introduction, nous avons dû implémenter les niveaux comme des fichiers texte d’un format bien précis voir ci-dessous.



Ce fichier représente le 2eme niveau du jeu

La première ligne représente la taille de la carte. Dans ce cas-ci, 20 en longueur et 20 en largeur. Pour le reste, on y trouve des éléments du jeu comme baba. A côté de leur nom, on y trouve les coordonnés associées en x et y. La quatrième valeur possible est un chiffre compris entre 0 et 3 représentant la direction initiale de l’objet (dans notre cas, on ne l’a pas utilisé car on y a pas trouvé d’utilité).

Pour implémenter tout ça dans notre jeu, nous avons eu recours à un tableau à double dimensions de type String. Nous avons fait ce choix car le développement de cette partie s’est faite aux alentours du cours de programmations et algorithmiques 2 sur les tableaux.

La structure du tableau se fait de cette manière Un sous-tableau représente une ligne du fichier. Celui-ci est de taille 4 pour accueillir le maximum d’information et toujours dans le même ordre (sauf la premier sous tableaux <-> 1 ere ligne du fichier): indice 0 : élément du jeu, indice 1 : coordonné en x, indice 2 : coordonné en y (et indice 3 : direction initiale).

L’initialisation se fait dans la classe « Extract » qui est indépendante des autres classes. Après que le tableau aura été initialisé, on l’appliquera dans la classe « Map » (à voir par la suite).