DOSSIER DE CONCEPTION : PROJET JEU

Mercredi 5 mars 2014

Groupe A

L3 info SPI

Sommaire

1 Présentation général

1.1 Objectif du document

Ce document est rédigé dans le cadre du projet de troisième année de licence SPI option informatique de l'Université du Maine. Le Dossier de Conception vient en complément du Cahier des Charges. Il a pour objectif de monter la modélisation du système à partir des besoins et contraintes exprimés dans le Cahier des Charges. Il informe également sur les outils et les normes utilisés pour la conception du produit.

1.2 Objectif du projet

L'objectif de ce projet est de concevoir et de développer une application de Picross, ainsi que de rédiger les documents inhérant à la gestion et la finalisation d'un projet informatique. Les utilisateurs doivent pouvoir, bien évidemment, jouer, mais il doivent également pouvoir créer des grilles, modifier la taille des grilles, et charger les partie préalablement sauvegarder. Il doit également être possible de jouer sous son propre profil afin d'avoir accès aux options les plus poussées de l'application.

1.3 Documents de références

Afin de réaliser le présent Dossier de Conception nous nous sommes appuyés sur le Cahier des Charges ainsi que sur les exemples que nous avons récupérés grâce à un ancien élève de DUT informatique.

2 Modélisation du projet

2.1 Les cas d'utilisations

Partie de Kévin

2.2 Contraintes de conception

Partie de Kévin

2.3 Le scénario du système

Partie de Kévin

2.4 Schéma de navigation

Partie de Kévin

3 Modélisation de la Sauvegarde des données

3.1 Dictionnaire de données

Nom de l'entité	Type	Commentaires
Grille	Class Ruby Grille	Une collection de case, jointe à un nom et des paramètres ("ALEA20", 20/11/2014)
Profil	Chaîne de caractère	Le profil joueur, constitué uniquement du nom du profil ("joueur1")
Scores	Tableau	Un tableau à trois colonnes, comprenant le nom de la grille, le profil et le score
Parties Sauvegardées	Class Ruby Partie	Partie en cours, sauvegardée avec tous ses paramètres (profil = "", grille ="", temps= – :–, ...)

3.2 Description des données sauvegardées et de leur stockage

L'ensemble des données sauvegardées, présentées brièvement ci-dessus, est ici décrite plus en détail.

Grille : les grilles sont les supports du jeu(cf Cahier des Charges 3.1.1). Pour permettre à plusieurs utilisateurs de jouer sur une même grille, il est nécessaire de sauvegarder ces objets. Les grilles sont constituées d'un collection de case, d'un nom, attribué à la création et de divers paramètres, stockés sous formes d'entiers. Elles sont stockés chacune dans un fichier propre, à leur création, qu'elle soit manuelle ou aléatoire. Le dossier des grilles comporte trois fichiers de grilles par défaut.

Profil : Les profils sont les noms que choisissent les utilisateurs(cf Cahier des Charges 3.1.2, qui les définissent dans le jeu, et qui les lient aux grilles créées, aux parties jouées et aux scores. Un profil est principalement constitué d'une chaîne de caractères, choisie par l'utilisateur à la création du profil. Cette chaîne est utilisée à la sérialisation du profil en tant que nom du fichier sur lequel sont sauvegardée les informations du profil. Cette chaîne, de nouveau entrée par l'utilisateur, servira à retrouver et charger son profil

Scores : Les scores sont des entiers, dépendant du temps réalisé pour finir la Grille. La donnée à sauvegarder "Scores" est un tableau à trois colonne, comprenant le score lui même, le nom de la grille sur lequel il a été réalisé, et le nom du profil l'ayant réalisé.

Parties Sauvegardées : Elles correspondent à la sérialisation d'une instance de la classe Partie, contenant un certin nombre de paramètres et d'informations, comme la grille utilisée, l'utilisateur, le temps écoulé, etc. Ces paramètres sont conservés sur un fichier lors de la sérialisation. L'utilisateur peut ainsi quitter une partie, fermer le programme, et reprendre cette partie plus tard.

3.3 Méchanismes de Sauvegarde des Données

Cette partie détaille l'ensemble des méchanismes de sauvegarde des données précédemment décrites, c'est à dire quand et comment ces données seront sauvegardées ou chargées.

- Les Grilles sont sauvegardées après leurs initialisation. Elles sont générées de deux manières : soit aléatoirement, sur demande d'un utilisateur, soit manuellement dans un éditeur. À chaque fois, après avoir été initialisée, la grille est sérialisée via le module YAML et sauvegardée dans un fichier propre au format nom_createur.nom_grille. Les grilles sont stockées dans un même dossier, afin de permettre aux utilisateurs de jouer à toutes les grilles, et pas uniquement celles qu'ils sont crées. Lorsqu'un utilisateur souhaite charger une grille, il a le choix entre voir la liste des ses grilles et voir la liste de toutes les grilles(tous créateurs confondus). Une fois qu'il en a choisie une, elle est chargée dans la partie.

- Les Profils sont sauvegardés à leur création. À chaque fois qu'un utilisateur crée un nouveau profil, celui ci est sérialisé et sauvegardé sur un fichier propre via le module YAML. Il est de nouveau sauvegardé à chaque modification. Les utilisateurs entrent leur nom de profil au lancement du jeu, et une recherche dans les dossiers correspondant à chaque profil permet de determiner si le profil en question existe déjà. Une fois le profil trouvé, il est chargé.

- Les Scores sont sauvegardés à chaque fin de parties. Ils sont stockés dans un tableau, comme indiqué ci-dessus. À chaque fois qu'un nouveau score est réalisé, il est ajouté au tableau, et le tableau est sérialisé et sauvegardé dans un fichier propre via le module YAML. Le tableau est chargé depuis ce fichier à chaque lancement du jeu via le module YAML.

- Les Parties Sauvegardées sont sauvegardées sur demande de l'utilisateur en cours de parties. Elles sont ajoutées à une collection de Parties Sauvegardées, permettant de trier ces Parties, notemment en fonction du Profil joueur, pour proposer à ce dernier de continuer la(les) dernière(s) Partie(s) Sauvegardée(s). À chaque fois qu'une Partie Sauvegardée est ajoutée, la collection est sérialisée et sauvegardée sur un fichier propre via le module YAML. Cette collection est chargée depuis le fichier à chaque lancement du jeu.

4 Interface Homme-Machine

4.1 Introduction

Nous allons réaliser l'interface homme-machine concrète correspondant au schéma de navigation ci-dessus. Les principales maquettes à représenter seront donc:

- Accueil
- Crédits
- Options
- Profil
- Édition
- Jouer
- Menu de pause

4.2 Représentations

La première interface apparaissant au lancement de l'application est l'acceil.

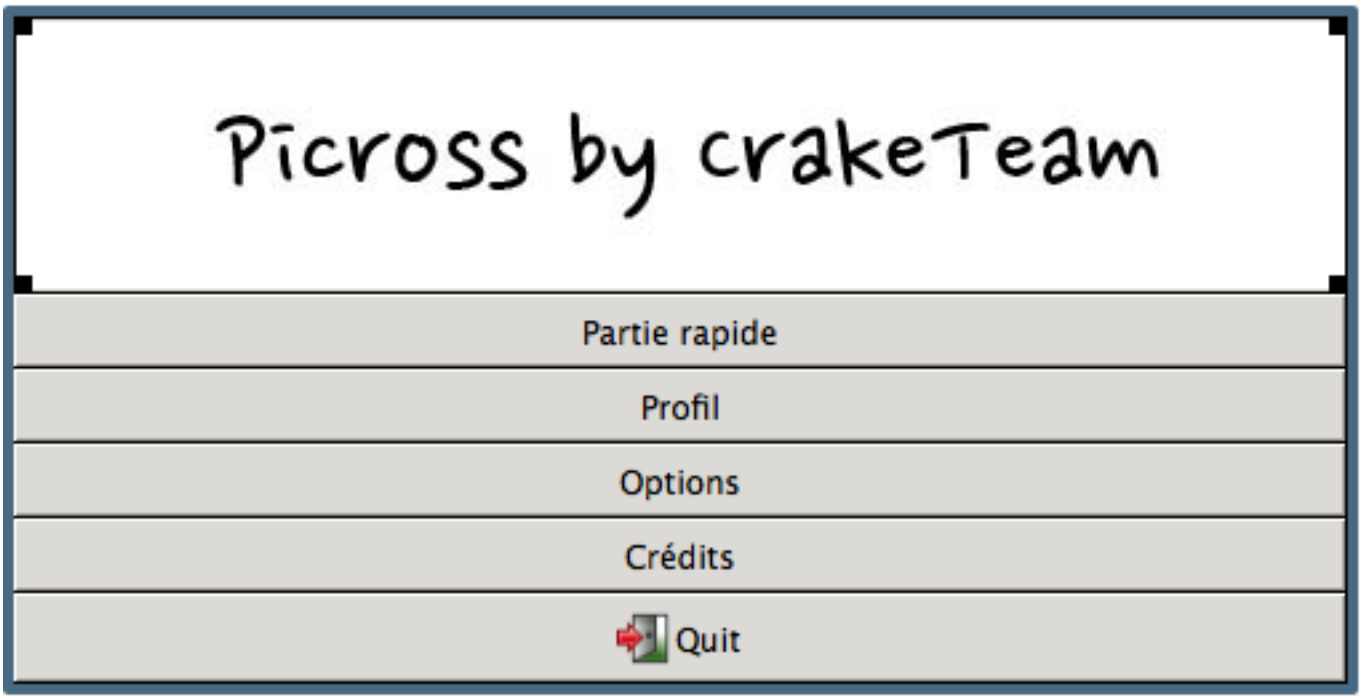

Figure 1: Accueil principal

On peut alors voir les différents choix disponibles. Nous allons commencer par les Crédits dans lesquels on peut consulter la liste des développeurs.

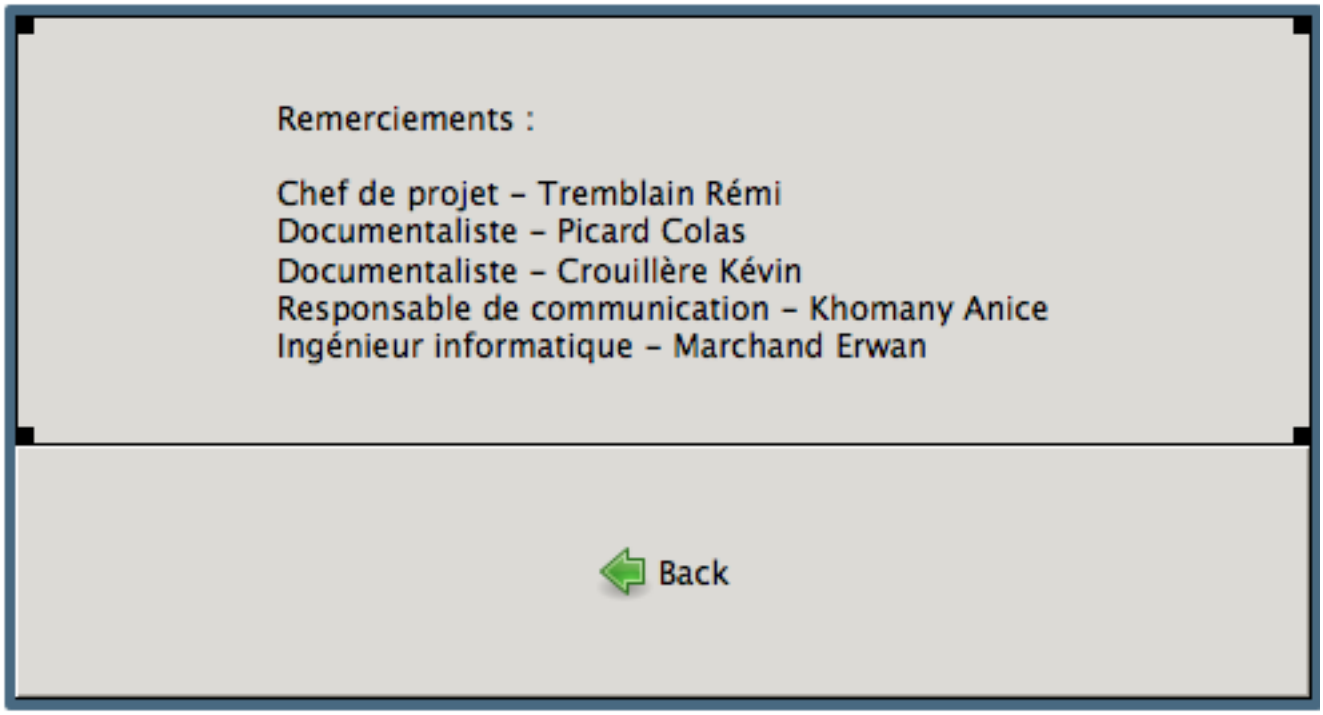

Figure 2: Crédit

Nous pouvons ensuite accéder aux options dans lesquels on peut changer la langue par exemple.

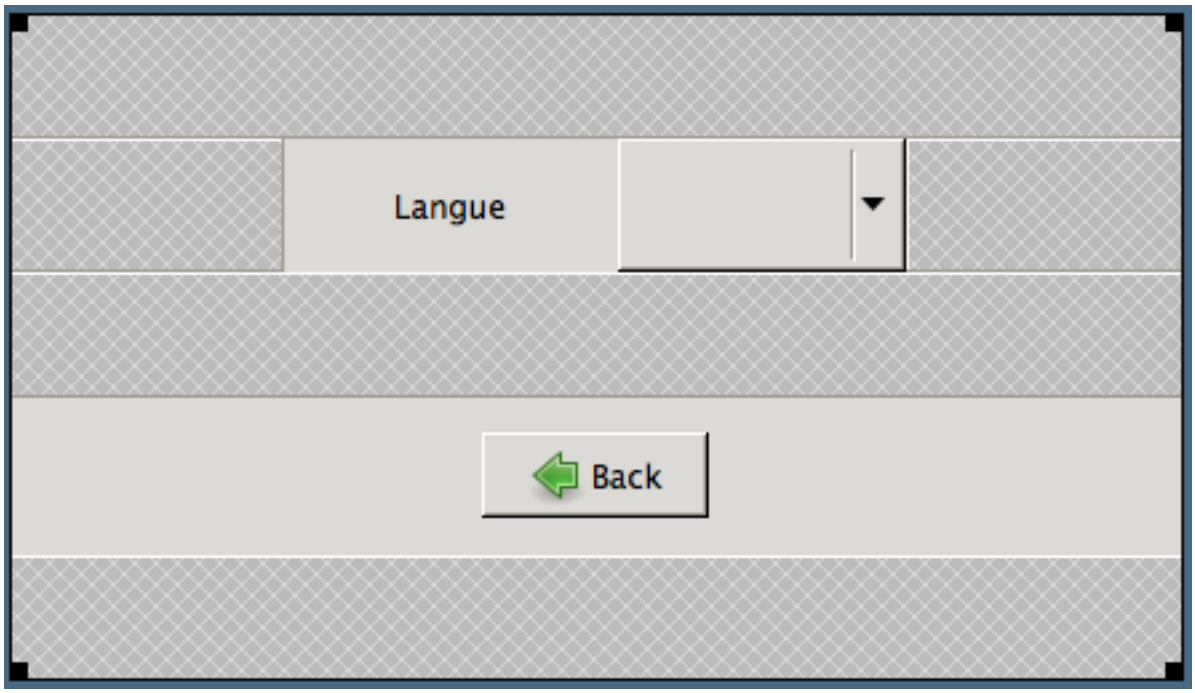

Figure 3: Options

On peut alors charger ou créer un profil en saisissant son nom.

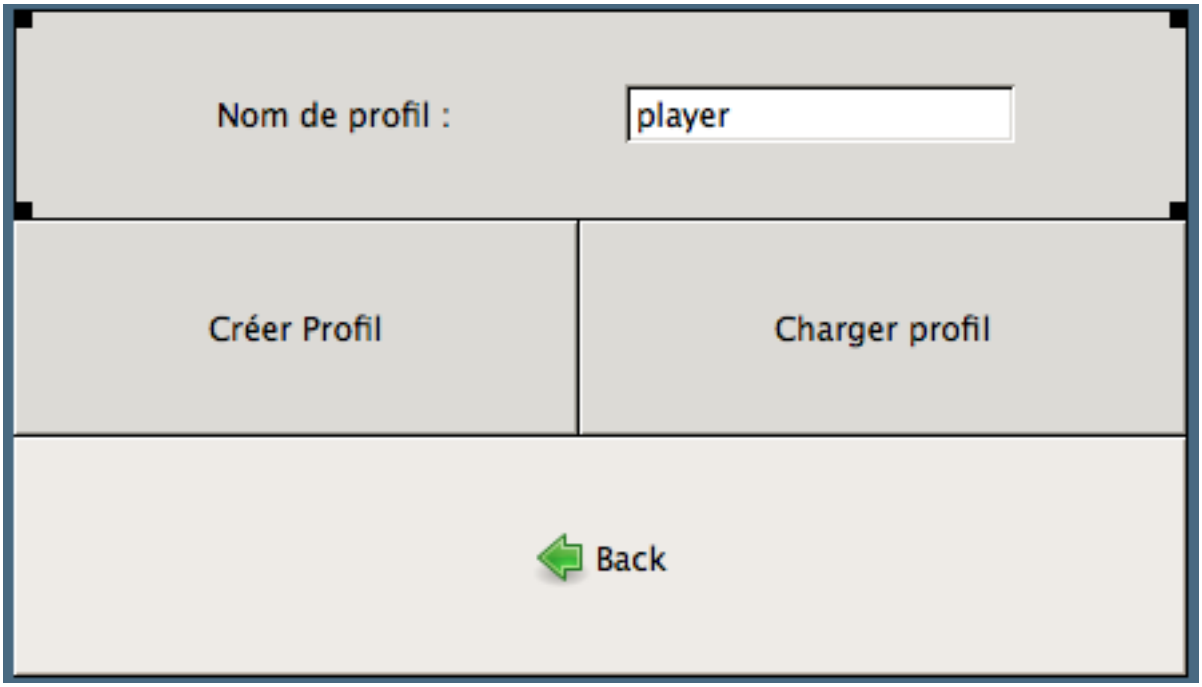

Figure 4: profil

Le menu principal change alors pour accueillir la nouvelle option "Édition".

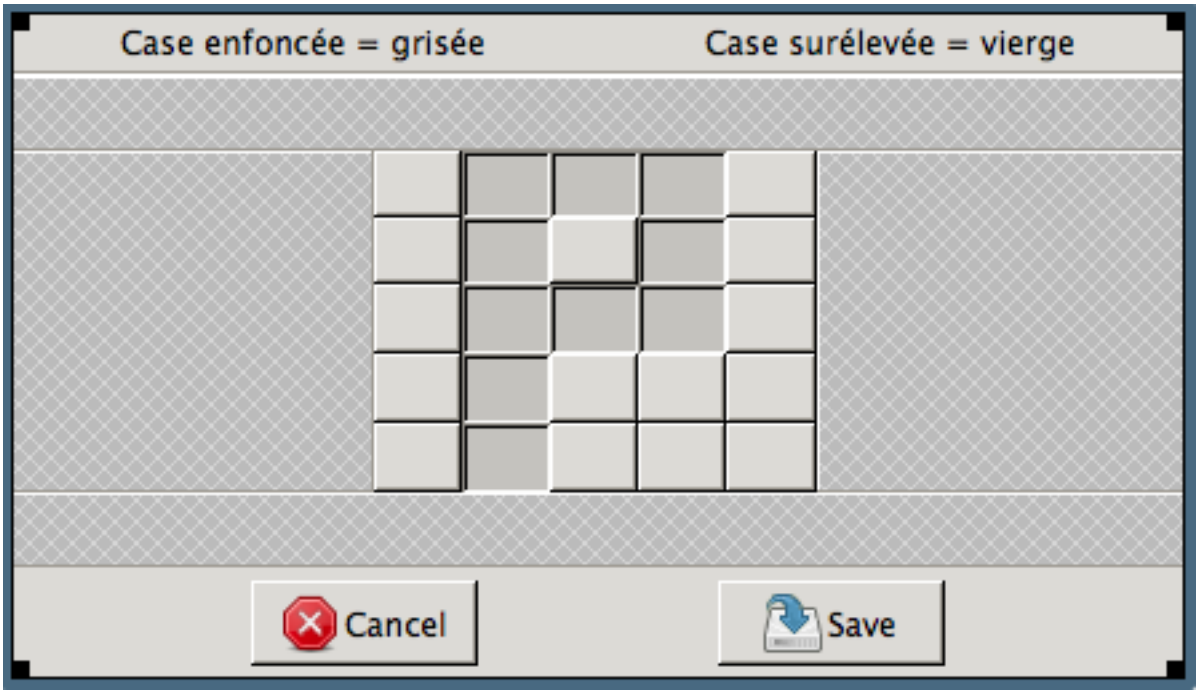

Figure 5: Édition

La principale option reste tout de même le jeu. On doit premièrement choisir entre charger une ancienne partie ou en commencer une nouvelle.

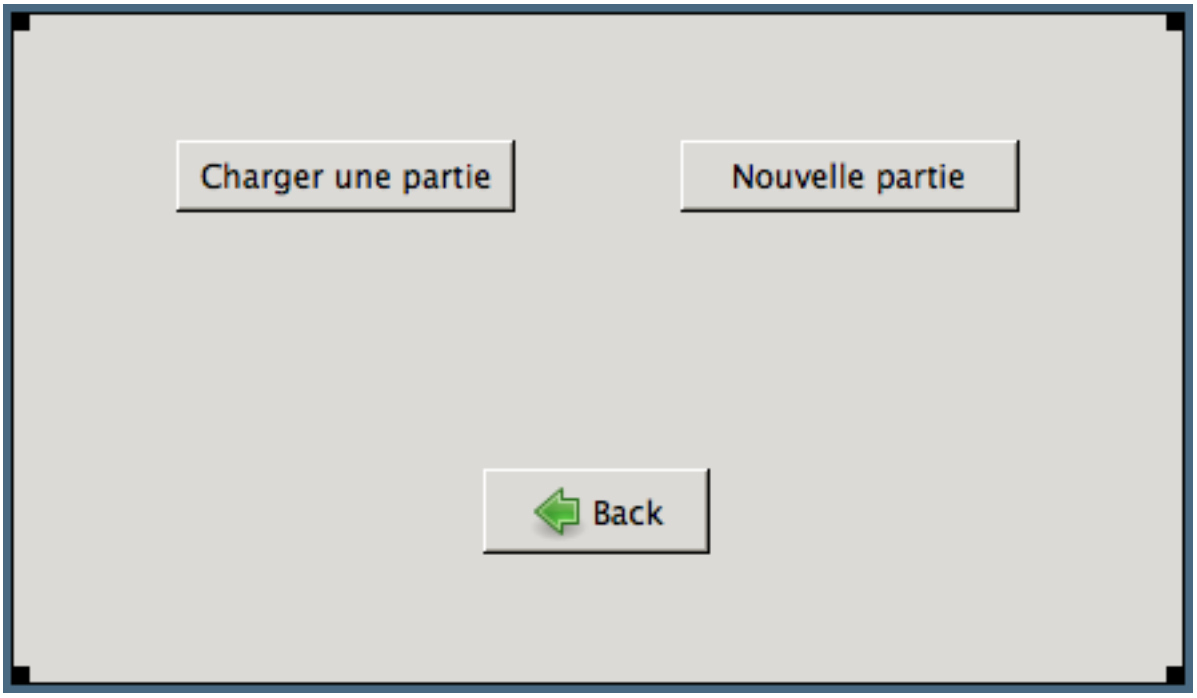

Figure 6: Choix d'ancienne ou nouvelle partie

Si on désire charger une ancienne partie, on pourra le faire grâce au menu de sélection de sauvegardes.

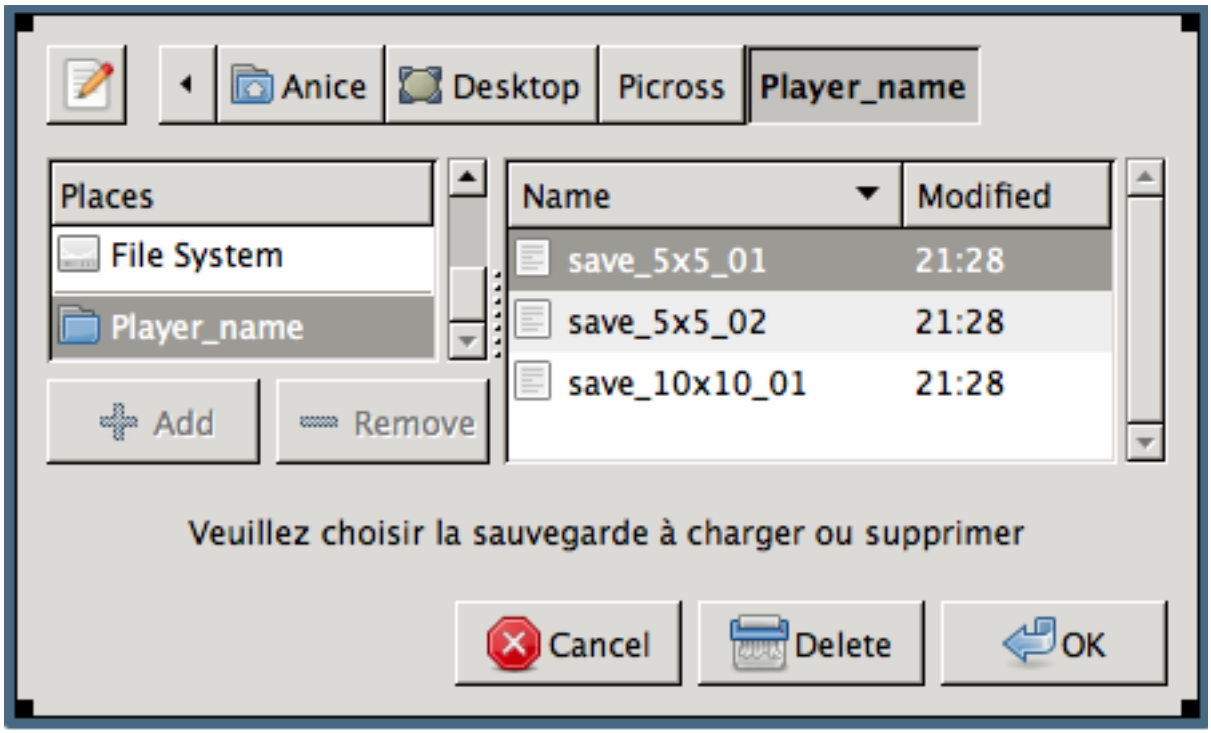

Figure 7: Charger une partie sauvegardée

En commençant une nouvelle partie, on accède au choix de la taille de grille.

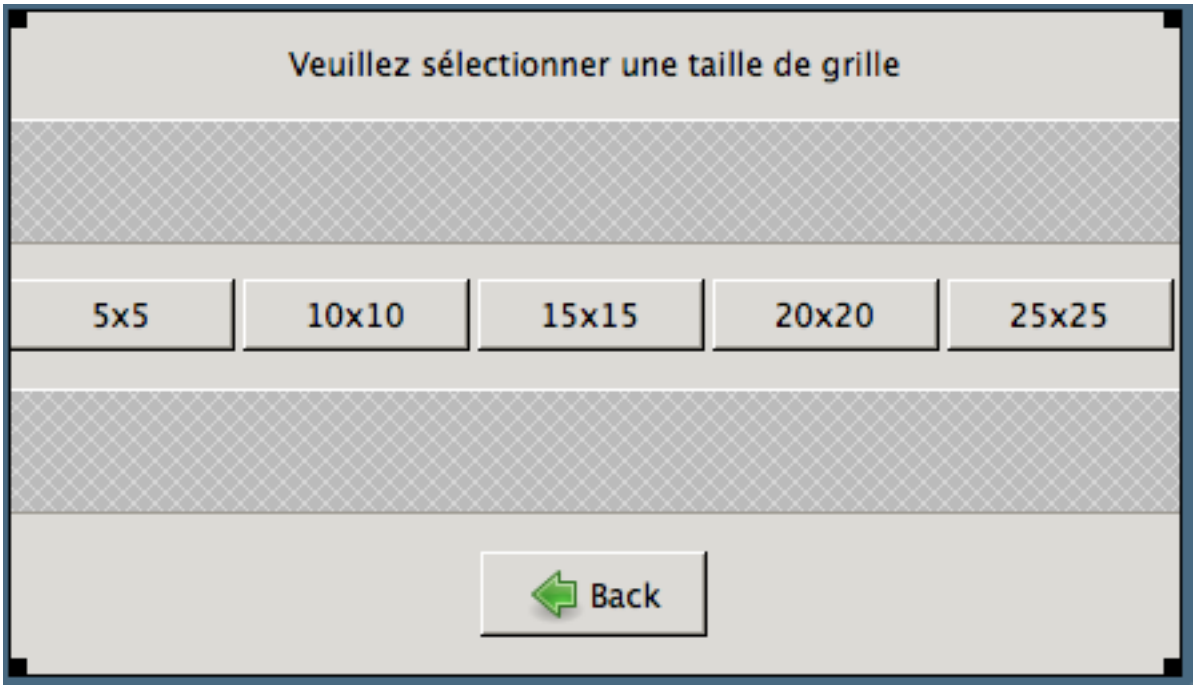

Figure 8: Choix de la taille

Une fois la taille sélectionnée, on pourra se lancer dans la partie.

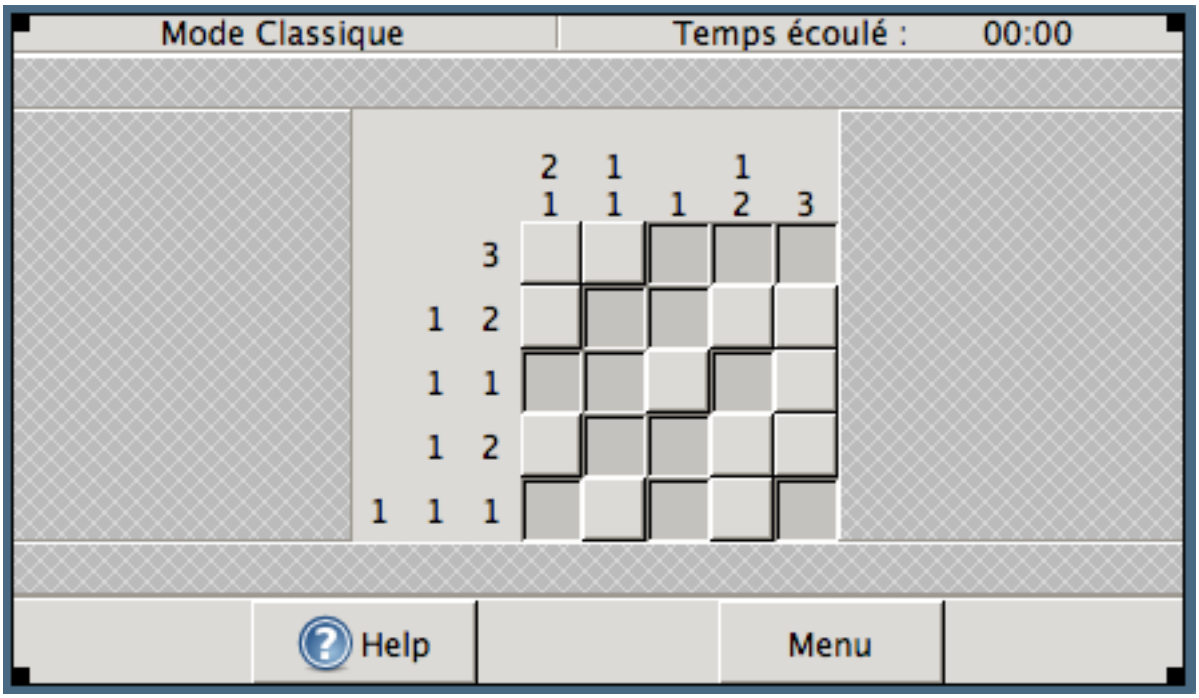

Figure 9: Interface de jeu

Un menu de pause est disponible lors de la partie. Le chronomètre s'arrête et quelque options

apparaissent.

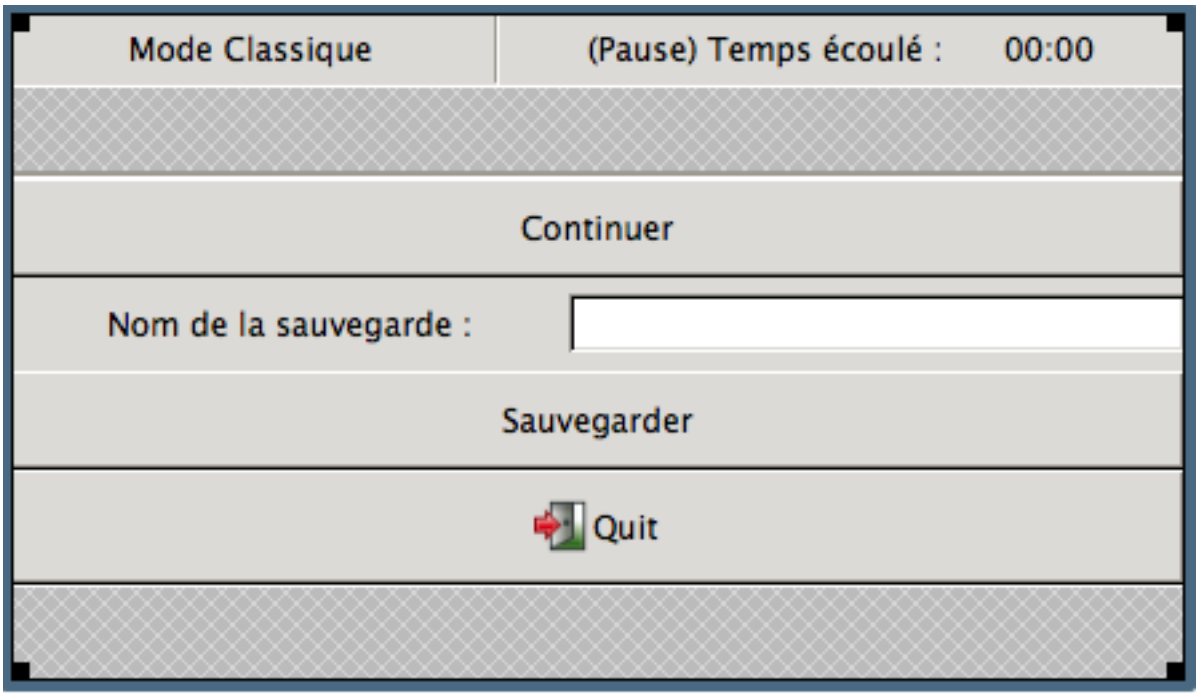

Figure 10: Menu de pause

La partie terminée, un menu apparaîtra afin de renouveler l'expérience ou revenir au menu principal.

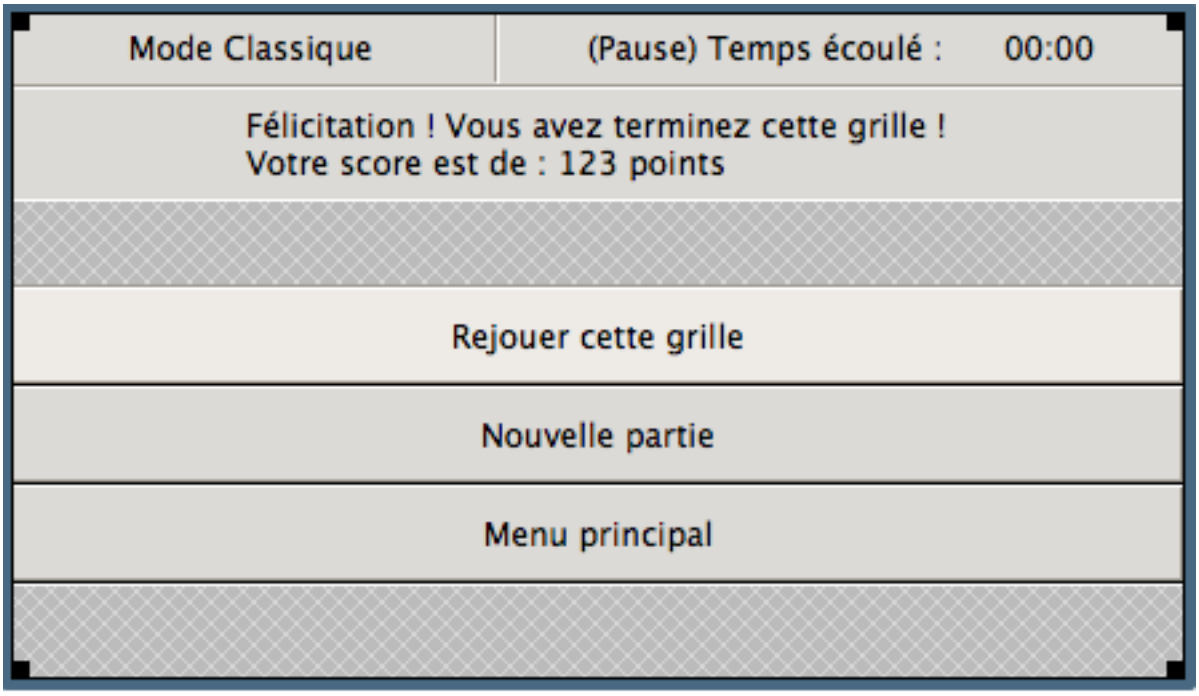

Figure 11: Édition

5 Outils, normes et standards de programmation

5.1 Languages utilisées

Ci dessous nous référençons la liste des différents langages utilisés dans le projet :

- Ruby : Langage de programmation orienté objet, utilisé en version 2.0.0 sur nos machines personnelles, avec une compatibilitée sur les versions antétrieurs,

5.2 Envrionnement et outils de développement

Le développement sera effectué sur des ordinateurs suffisamment puissants auxquelles nous avons accès à l'Université du Maine, ainsi que sur nos machines personnelles.

- Version utilisé de Ruby : 2.0,
- GTK : Bibliothèque graphique (The **G**IMP **T**ool**k**it) utilisé avec Ruby.

5.3 Standards de programmation

Durant le développement, nous respecterons les conventions de codages ci contre :

- Les commentaires doivent être rédigés en français et permettent d'expliquer de façon claire le code.
- Le code doit être rédigé de la manière la plus lisible possible (indenté et explicite).

- La documentation du code sera faites à l'aide de Ruby doc.