Application de création et d'aide à la résolution de puzzle *picross*

Cahier d'analyse et conception

Étudiants :

Brinon Baptiste
Brocherieux Thibault
Cohen Mehdi
Debonne Valentin
Lardy Anthony
Mottier Emeric
Pastouret Gilles
Pelloin Valentin

Groupe n°2
Licence Informatique
Le Mans Université
25 janvier 2018

Sommaire

1 | Présentation

1.1 Introduction

Dans le cadre de l'unité d'enseignement "Génie logiciel 2" de la Licence d'informatique de Le Mans Université, les étudiants de troisième année sont amenés à travailler sur un projet de développement d'une application.
Ce document permet de présenter les solutions au cahier des charges, ainsi que l'architecture du programme demandé. Celui-ci est composé en plusieurs parties : la gestion d'une partie, l'aide à la resolution du picross, la gestion des statistiques du joueur (scores, niveaux débloqués,...) ainsi qu'une interface graphique permettant l'utilisation de ces fonctionnalités. Ce document décrit aussi l'utilisation du temps qui nous est imparti.

1.2 Objectif de l'application

Nous devons réaliser un jeu de type picross (aussi appelé nonogramme, logigramme ou hanjie) permettant à un utilisateur de résoudre des grilles et de l'aider dans sa réalisation.

1.3 Outils

Afin de réaliser notre application de Picross, nous utilisons plusieurs outils. Ruby est le langage utilisé pour programmer le logiciel, incluant une documentation généré par Rdoc et l'utilisation des bibliothèques de Gtk pour la la réalisation des interfaces graphiques. LaTeX est utilisé pour rédiger et mettre en forme les différents livrables avant de les convertir en format PDF. Git et Github permettent le partage d'informations et la mise en commun du code ainsi que les livrables associés. Nous nous servons de Discord afin de communiquer au sein du groupe afin de partager nos idées, nos informations et notre travail. Pour concevoir les différents diagrammes UML, nous avons recours au logiciel Astah et au site internet Tom's planner. Afin de créer les maquettes du logiciel (disponibles en Annexe) permettant d'avoir un rendu de la futur interface graphique, nous avons utiliser le logiciel Balsamik. Pour simplifier la vérification de la qualité du code produit, nous utilisons Codeclimate (maintenabilité) ainsi que Travis (compilation et tests).

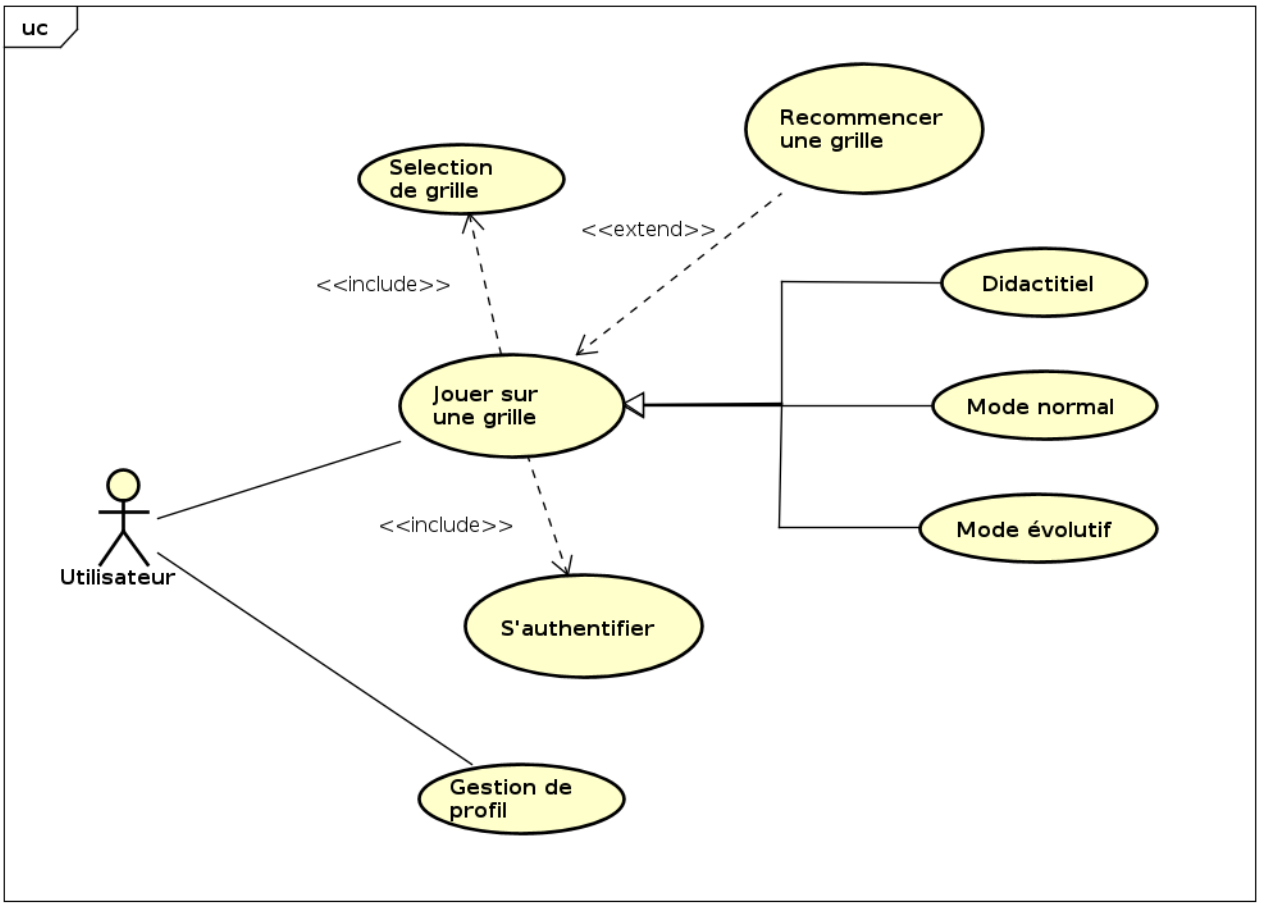
2 | Conception générale

2.1 Diagramme de Gantt détaillé

À travers ce diagramme de Gantt, nous observons l'organisation du projet sur les trois mois à venir. Le projet se découpe en plusieurs modules : carte, gameplay, utilisateur, IHM, test et améliorations. Chaque module a une durée de vie, nous lui accordons un temps de travail et un certain nombre de personnes, les modules sont également divisés en différentes étapes ce qui correspond aux fonctionnalités de l'application. Les modules sont imbriqués les uns dans les autres, c'est-à-dire que certains dépendent d'autres modules. Sans les modules gameplay et carte réalisés, le module IHM ne peut être réalisé.

2.2 Diagramme de cas d'utilisations

FIGURE 2.1 – Diagramme de *Cas d'utilisation*



Le but de l'utilisateur est de jouer sur une grille de Picross. Pour cela, it doit avoir un profil. Il peut donc en créer un nouveau s'il en a envie, ou s'authentifier à un profil déjà existant. Le joueur doit ensuite selectionner une grille à l'aide du menu. Il a le choix entre trois types de jeu : - Le didacticiel, permettant d'apprendre les règles du jeu. - Le mode classique, permettant de jouer sur plusieurs grilles de tailles et difficultés variables. - Le mode évolutif, agrandissant la taille de la grille au fur et à mesure que Pierre résoudra celle-ci. A tout moment, celui-ci peut utiliser l'aide intégrée au jeu pour se débloquer. Il peut aussi choisir de recommencer la grille s'il le souhaite.

2.3 utilisation de profils

2.4 partie classique

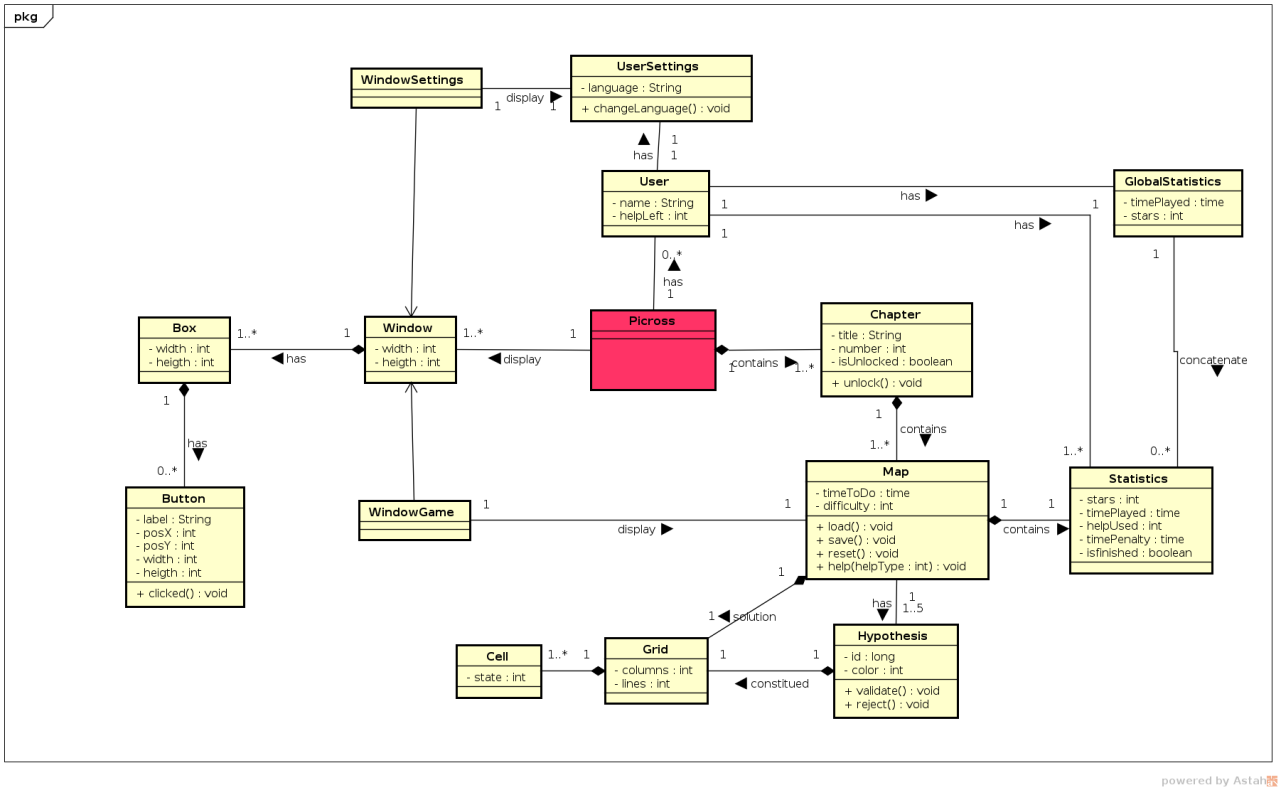
2.5 parties evolutives et didacticiel

// Schéma liens entre écrans // Description

3 | Conception détaillé

3.1 Diagramme de classe

FIGURE 3.1 – Diagramme de *Classes*



Le diagramme de classe ci-dessus présente les différentes classes que nous allons implémenter afin de programmer notre application. Nous avons tout d'abord la classe Picross qui est la racine permettant de rassembler toutes les autres classes et qui représente l'ensemble du jeu. Cette classe est liée à 3 modules principaux : l'Interface Homme-Machine, les données utilisateur, ainsi que le système de jeu.

L'IHM commence par la classe Window correspondant à la fenêtre de l'application et possédant une taille définie et modifiable. Cette fenêtre contient plusieurs objets de classe Box possédant chacune une taille définie et représentant les différentes parties graphiques de la fenêtre (la grille, le chronomètre, l'affichage des informations, etc...). Chacune de ces boîtes peut contenir des objets de classe Button correspondant aux boutons nécessaires à l'interaction entre l'utilisateur et le logiciel (valider la grille, retourner au menu, etc...).

Les données de l'utilisateur sont rassemblées dans la classe User possédant un pseudo, le nombre d'aides auxquelles il a le droit, et des données séparées en deux parties. La première correspondant aux différents paramètres du logiciel comme la langue utilisée ou la taille de la fenêtre (classe WindowSetting) et modélisée grâce à la classe UserSettings. La deuxième est liée aux statistiques générales de l'utilisateur (classe GlobalStatistics) comme le temps joué ou le nombre total d'étoiles ainsi que les statistiques liées à chaque grille finie par le joueur (classe Statistics)

comme le nombre d'étoiles reçues, le temps mis à la finir ainsi que le nombre d'aides utilisées.

Le système de jeu est rassemblé dans la classe Chapter possédant un titre et un numéro ainsi qu'un booléen indiquant si ce chapitre est verrouillé ou non. Chaque chapitre est composé de plusieurs grilles représentées dans le diagramme par la classe Map possédant une difficulté ainsi que le temps de base à réaliser la grille permettant ensuite de calculer le nombre d'étoiles obtenues par le joueur en fonction de son temps. Chacun de ses objets Map est capable de réaliser plusieurs actions comme s'initialiser, sauvegarder ou se réinitialiser. Lors d'une partie sur une grille, l'utilisateur peut utiliser le mode hypothèse représenté par la classe Hypothesis composé d'un numéro et d'une couleur. Chacune de ses hypothèses est composée d'une sauvegarde de la grille (classe Grid) avec l'état de chaque case (colorié, non colorié, croix) représenté par la classe Cell.

4 | Annexes

// maquettes