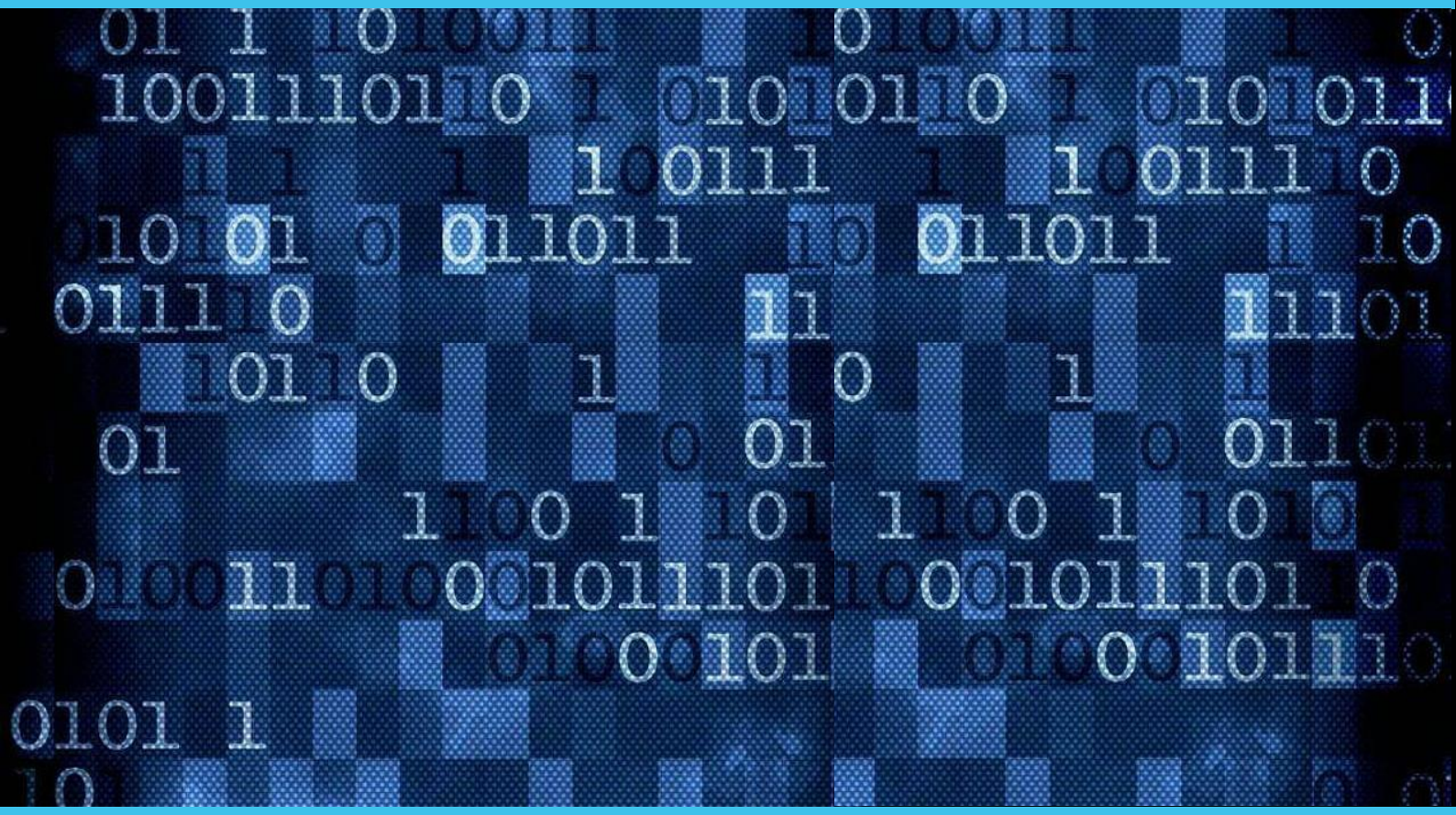
Livrable 1

Projet : Le Jeu De La Vie



Rayane OULDALI et Thaïs VAINES

Sommaire

Préambule

Rappel du contexte

Au cours de ce projet nous devons implémenter un algorithme du jeu de la vie.

Le jeu de la vie est un automate cellulaire (Le joueur n'intervient pas dans le jeu mais celui-ci est coder de manière à suivre son cours jusqu'à l'extinction du programme). Ce jeu issu d'un modèle mathématique proposé par John Conway en 1970 représente des cellules, blanches si elles sont vivantes et noires si elles sont mortes. A chaque itération du programme, les cellules sont analysées afin de savoir si elles doivent vivre, mourir ou rester dans leur état actuel.

Rappel des attentes du client

Règles du jeu :

- Une cellule morte possédant exactement trois cellules voisines vivantes devient vivante (elle naît) ;
- Une cellule vivante ne possédant pas exactement deux ou trois cellules voisines vivantes meurt.

Objectifs du livrable 1

Le premier objectif est de mettre en place des diagrammes simple (UML ou diagramme de classe) afin que l'on puisse avoir une vue d'ensemble du projet, de son fonctionnement et de la façon dont le code devra être agencé pour que le programme soit fonctionnel et le plus simple possible.

Les diagrammes

Diagramme Use Case

L'une des premières étapes pour se situer dans le projet et le présenter de façon simple est le diagramme Use Case. Ce diagramme permet d'identifier les possibilités d'interaction entre le système et les acteurs. C'est-à-dire toutes les fonctionnalités que doit fournir le système. Ici, l'acteur à savoir le joueur du jeu de la vie fournit un fichier au code afin que celui-ci se lance.

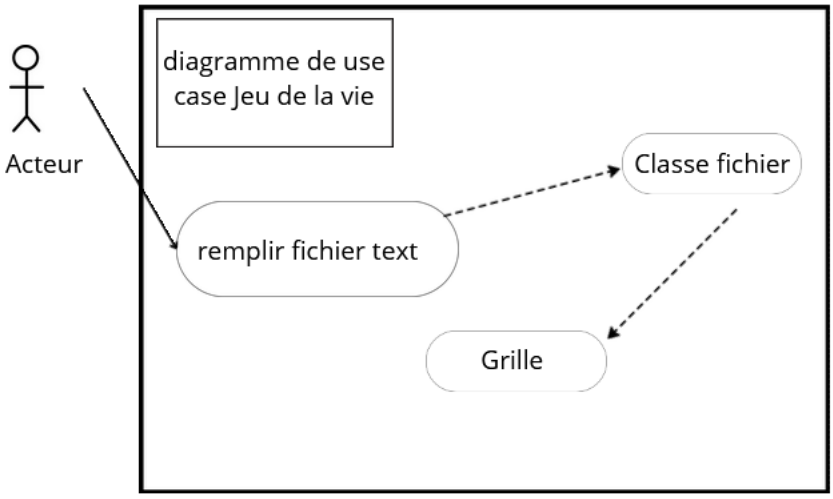


Diagramme de Séquence

Le diagramme de séquence permet de modéliser une fonctionnalité du système. Il va permettre de voir pas à pas les différentes étapes du fonctionnement du programme.

Dans notre cas le code permet de donner en entrer du code un fichier.txt composé du nombre de colonnes, de lignes ainsi que la disposition initiale des cellules mais aussi un nombre d'itération selon lequel le code doit fonctionner. Lors du lancement du code la grille est créée à l'état initial, puis, à chaque itération les cases changent de couleurs en fonction des critères du jeu.

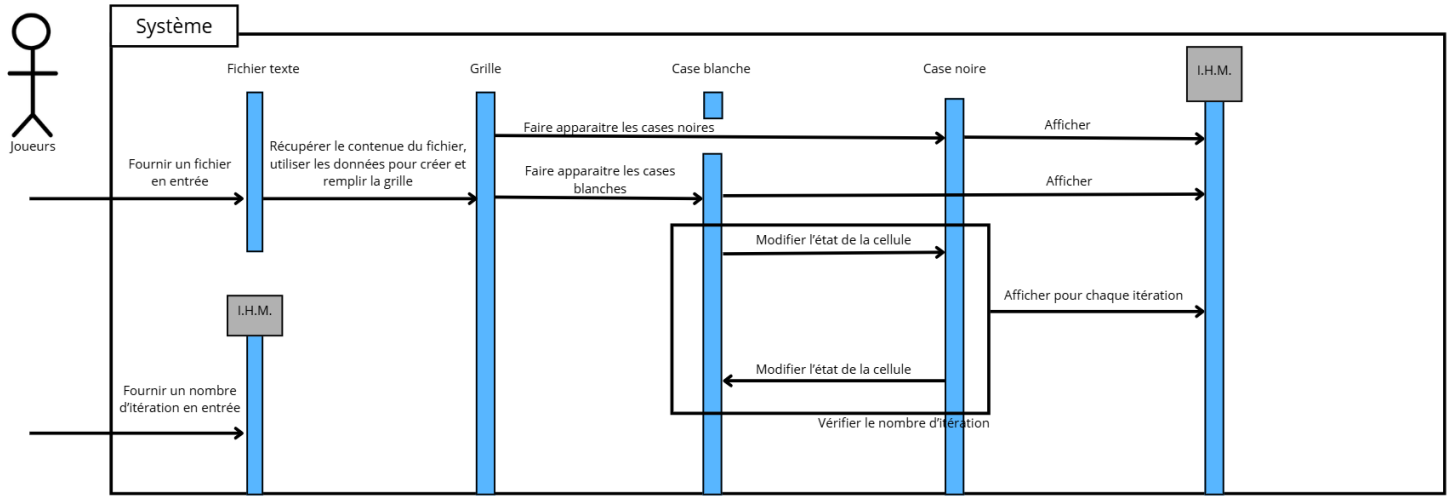


Diagramme d’Activité

Le diagramme d'activité est une représentation qui peut être apparentée à un algorigramme permettant de mettre en valeur les étapes du programme.

Le rond plein représente le début du programme, celui avec la croix représente la fin. Le losange représente une condition, la flèche partant de la droite du losange est le résultat si la condition n’est pas remplie, sur le bas on retrouve la suite du code lorsque la condition est validée. Au centre des ovales se trouvent les évènements du code (changement d’état, modification d’une donnée, appel d’une fonction, etc.).

Les conditions de la bonne ouverture du fichiers et la présence de contenu dans ce fichier sont des conditions immuables au bon fonctionnement du code, si ces conditions ne sont pas remplies le code s’arrêtera avant de faire de calculs.

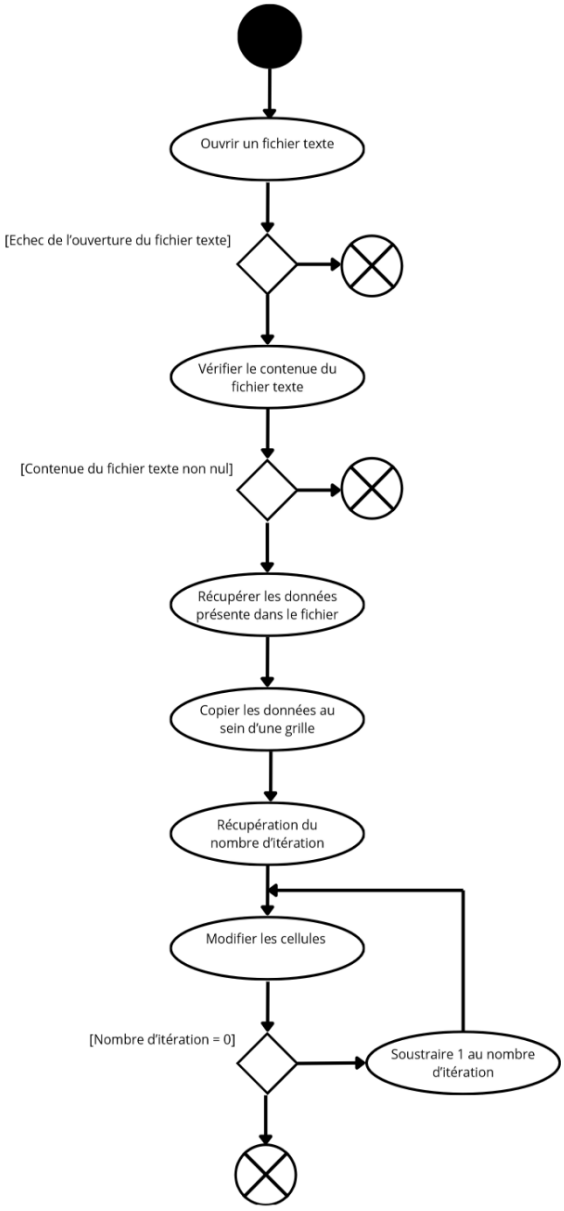
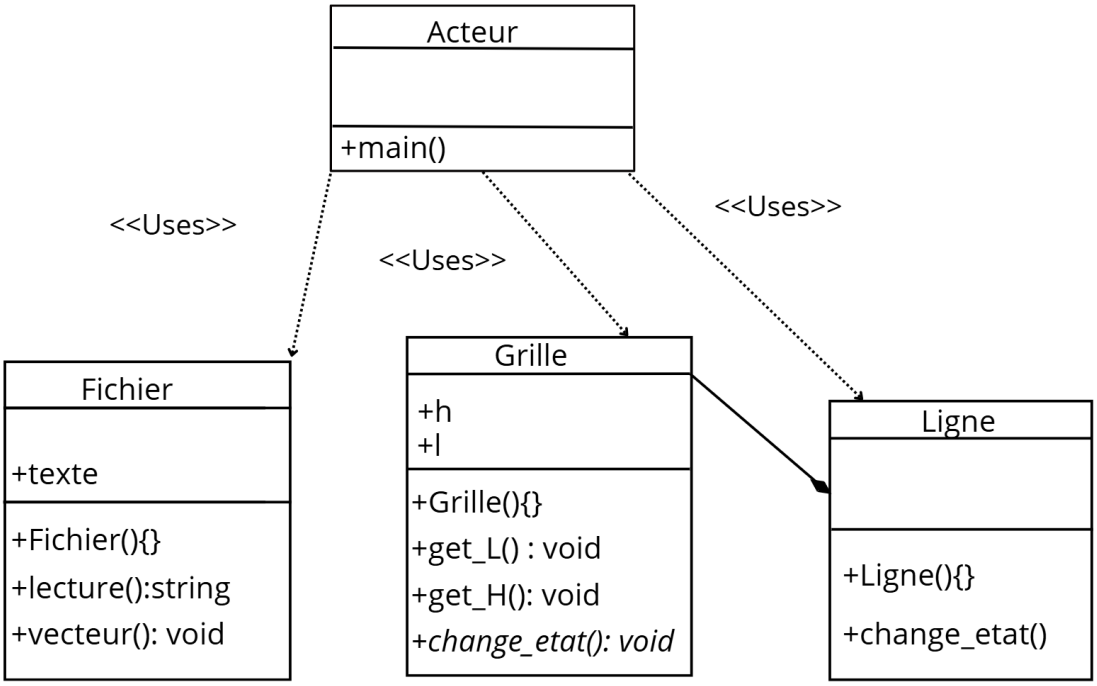


Diagramme de classe

Le diagramme de classe permet avec simplicité d'identifié l'ensembles des classes ainsi que leurs liens. (Héritage, Classe Abstraite, Classe virtuelle).

Ici la classe acteur contient le main() et les différents appel de classe et d'objet pour notre projet. La classe fichier lit le contenue d'un fichier dont nous avons au préalable donnée le chemin. La class fichier permet à l'utilisateur de pouvoir créer sa grille avec l'emplacement des cellules vivante et des cellules morte. La class grille est une classe abstraite étant donné que la méthode change_état() est abstraite et définie dans la classe ligne. Il y'a donc un notion d'héritage entre la classe Grille et la classe Ligne



Environnement de développement opérationnel

Afin de pouvoir travailler en équipe dans les meilleurs conditions, nous avons mis en place un environnement de développement. Celui-ci donne toutes les spécifications logistiques de notre projet pour que le travail soit efficace et la perte de temps due au merge et ajustement des codes et fichiers minimes.

Logiciel et outils

Nous allons travailler à l'aide du logiciel CLion qui est un environnement de développement intégré (IDE) sur lequel nous allons programmer en C++.

Tous les fichiers seront ordonnés et conservés sur GitHub dans un repository. Nous pourrons ainsi chacun travailler sur les documents de manière indépendante et tout rassembler une fois le travail achevé sur une des branches.

Langage et bibliothèque

Tout le projet sera codé en C++ en se basant sur les principes de la Programmation Orientée Objet (POO).

Afin de pouvoir générer une interface graphique nous utiliserons la bibliothèque externe STL disponible en C++.

La bibliothèque gtest de google nous permettra quant à elle de réaliser les tests unitaires nécessaires au projet.

Organisation du code

Les classes ainsi que les méthodes de notre code vont être déclarées dans des fichiers du « nom de la classe .h » et le contenue des méthodes seront placés dans les fichiers « nom de la classe .cpp »

Le fichier « main .cpp » abritera la classe main du programme contenant les appels de méthodes et les output ainsi que les lignes relatives à la création de l'interface et des test unitaires.

Conclusion :

Après avoir défini les attentes du client, nous avons pu définir des objectifs pour avancer sur la bonne voie.

Nous avons créé plusieurs diagrammes UML (Use case, séquence, activité, de classe) afin de ne rien oublier de prendre en compte.

- Le diagramme Use Case donne un aperçu des différentes fonctionnalités dont notre programme va disposer.
- Le diagramme de Séquence met en avant les différentes étapes du système nécessaires à son bon fonctionnement. On peut y voir les entrées et sortie au travers de l'Interface Homme-Machine (I.H.M.) mais aussi les étapes au sein même du programme, comme la création de la grille sur la base du fichier pris en entrée.
- Le diagramme d'Activité précise les étapes données précédemment par le diagramme de Séquence, il précise le début et la fin du programme ainsi que les boucles.
- Le diagramme de Classe quant à lui propose une architecture simplifiée du code. On y retrouve les classes : composées de leurs méthodes et attributs, mais aussi le lien entre celles-ci.

Pour finir, nous avons défini un environnement de travail (Logiciel et outils, Langage et libraire, Organisation du code).

Annexe

Lien vers le document partagé GitHub du projet : https://github.com/CESi-CPIA2-2425/Projet-2-Le-Jeu-De-La-Vie.git