

2017

Yoan LE DUOT

Antoine FORYS

ECE

12/11/2017

Projet Java : Chat

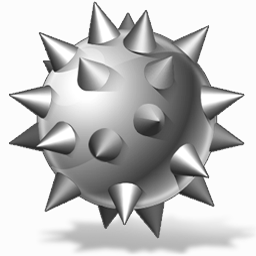


Table des matières

[I. Les fonctionnalités 2](#_Toc497996043)

[1. Les fonctionnalités implémentées 2](#_Toc497996044)

[2. Les fonctionnalités manquantes 2](#_Toc497996045)

[II. La conception 2](#_Toc497996046)

[1. Diagramme de classes 2](#_Toc497996047)

[2. Diagramme de séquences 2](#_Toc497996048)

[III. Implémentations expliquées 3](#_Toc497996049)

[1. Sous-titre 1 3](#_Toc497996050)

[2. Sous-titre 2 3](#_Toc497996051)

[IV. Manuel d’utilisation de l’application 7](#_Toc497996052)

[V. Evolutions possibles 7](#_Toc497996053)

# Les fonctionnalités

## Les fonctionnalités implémentées

Le groupe est constitué de deux étudiants en 1ère année du cycle d’ingénieur à l’ECE Paris : Antoine Forys et Quentin Garrigue. Nous sommes tous les deux issus d’une classe préparatoire aux grandes écoles.

Nos connaissances en informatique se sont enrichies depuis notre entrée à l’ECE. Nous sommes tous les deux débutants en langage C. C’est pourquoi nous avons décidé de nous associer afin de progresser tous les deux au même rythme et qu’il n’y ait pas de décalage entre nous.

## Les fonctionnalités manquantes

Quentin s’est concentré tout d’abord sur l’architecture du programme composé d’un main, d’un fichier header et d’un autre fichier « .c » comprenant le détail des fonctions. Il s’est ensuite chargé de déclarer les deux structures qui permettent au programme de fonctionner : la structure qui gère une case et la structure qui gère toute la grille. Il s’est ensuite occupé de toute l’allocation dynamique du programme. Il a programmé le placement des mines aléatoirement. Il a élaboré le menu ainsi que les instructions du jeu.

# La conception

Nous accueillons l’utilisateur avec un message de bienvenue. Ce message apparaît temporairement.

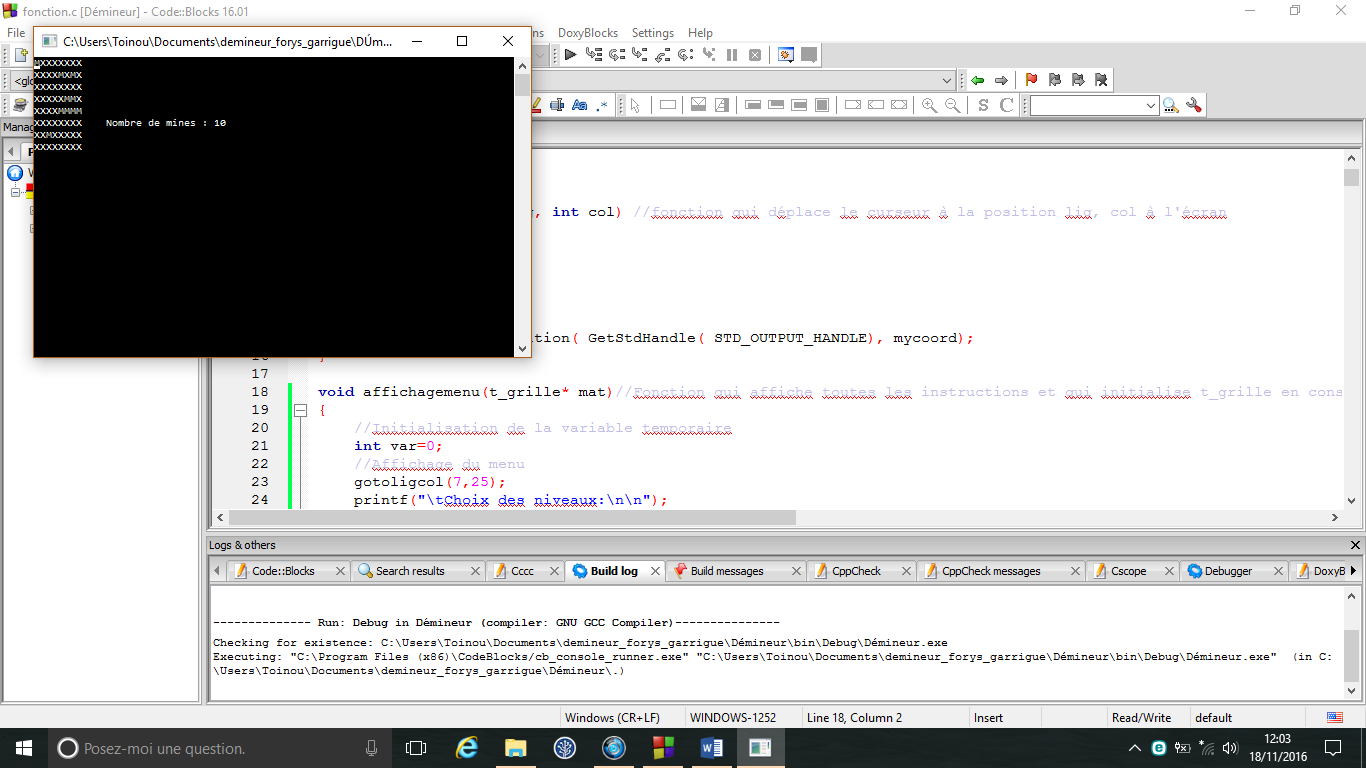
## Diagramme de classes

### 

L’utilisateur peut choisir en quatre niveaux de difficulté comme spécifié dans le cahier des charges :

## Diagramme de séquences

Pour illustrer nos propos nous avons laissé apparentes les mines, représentées par un « M » sur les captures d’écran de notre jeu. Ici notre curseur est donc sur une mine non dévoilée.



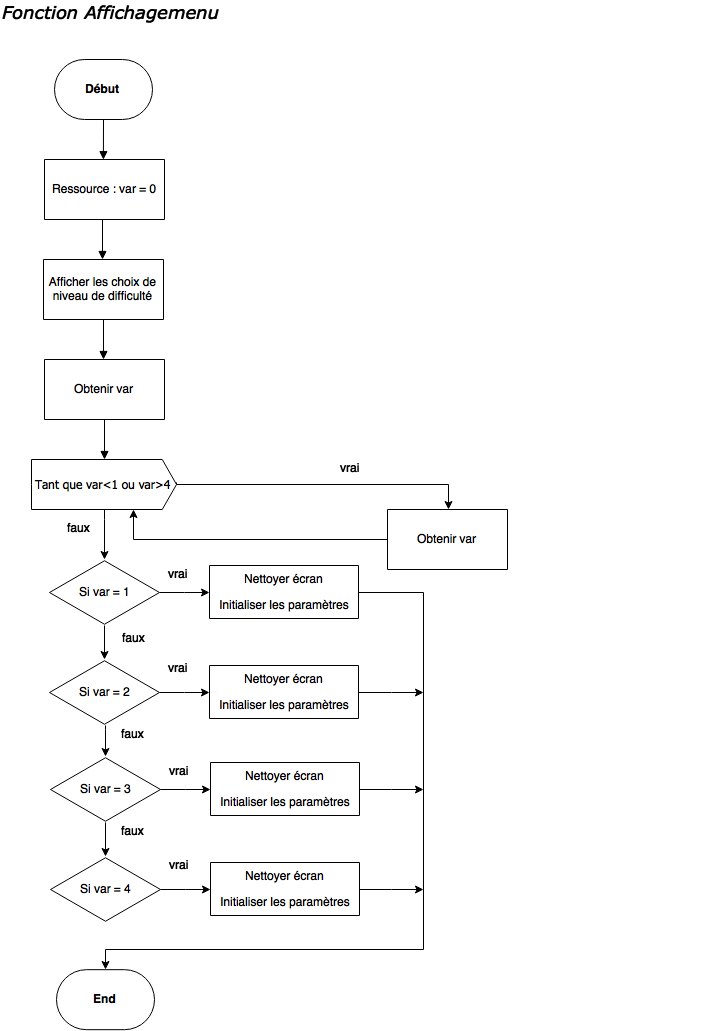
# Implémentations expliquées

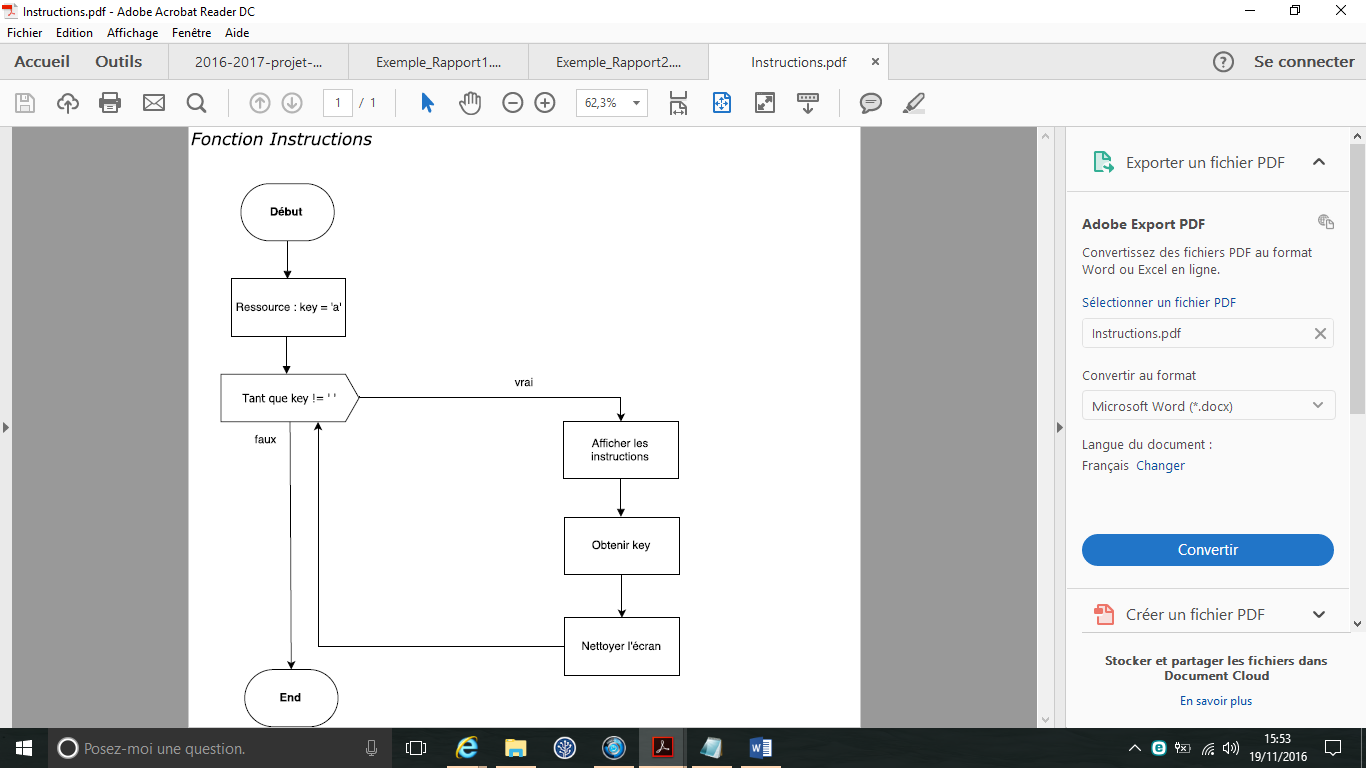
Nous allons décrire dans cette partie les choix de programmation que nous avons effectué afin de mener à bien notre projet.

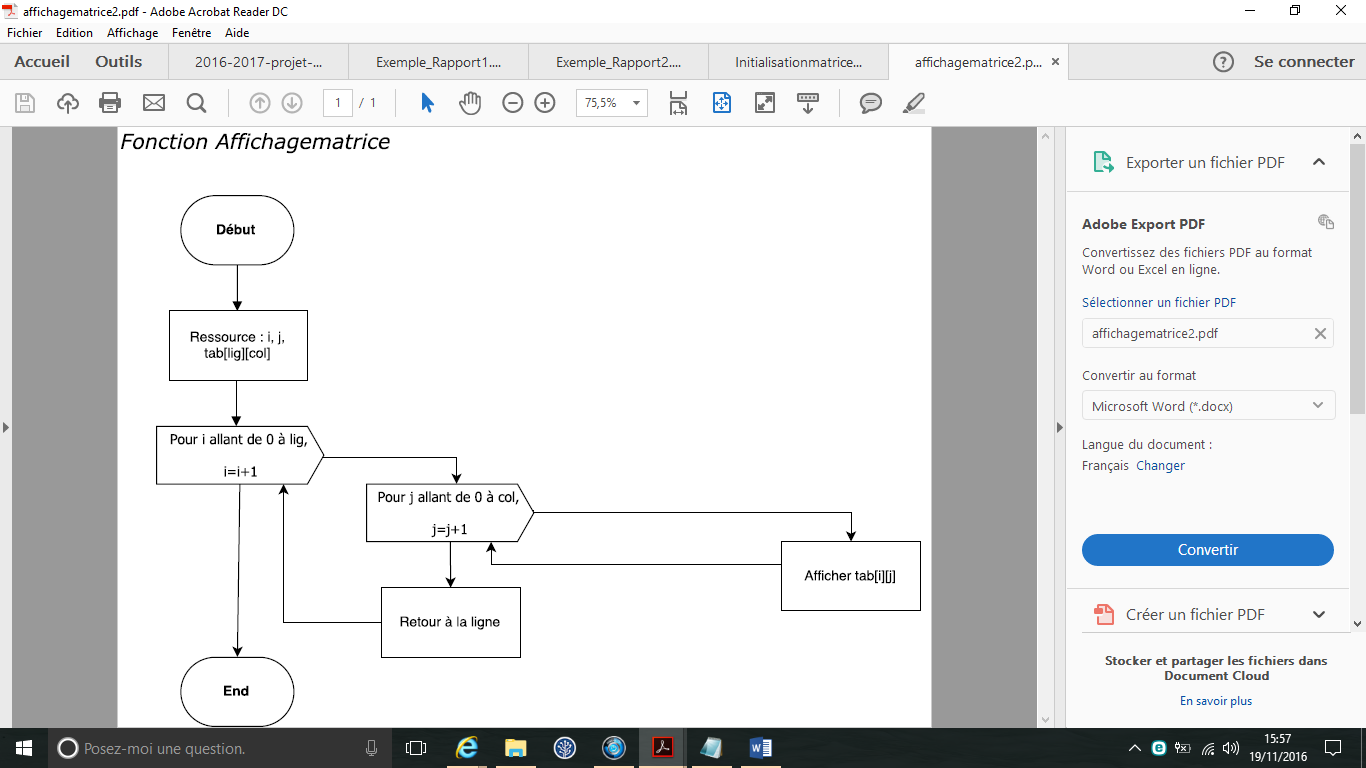
## Sous-titre 1

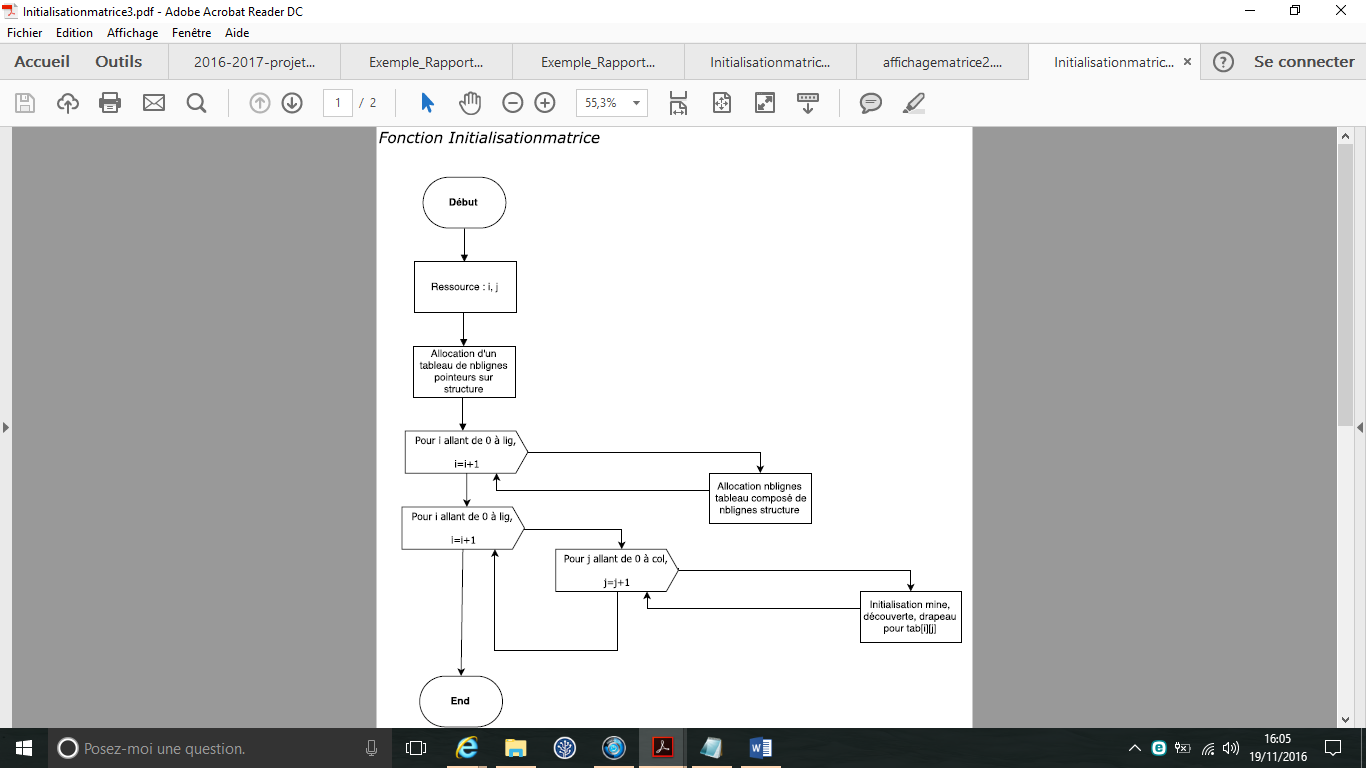
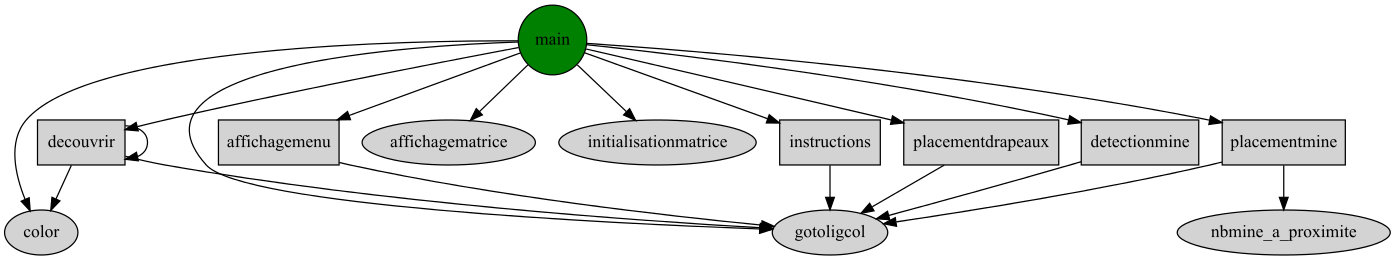
## Sous-titre 2

Nous avons décidé d’organiser notre projet autour de deux sous-programmes et une bibliothèque :









# Manuel d’utilisation de l’application

Nous avons passé un certain temps pour gérer le dévoilement des cases lorsqu’elles n’avaient pas de mines à proximité. Nous étions partis au début sur l’approche suivante : lorsque l’utilisateur appuyait sur une case : on initialisait le nombre de mines à proximité dans cette case et dans les cases adjacentes. Cependant lorsque nous avons voulu dévoiler plusieurs cases à la fois, on s’est aperçu qu’une grande partie des champs « minesproxi » n’étaient pas initialisés car l’utilisateur ne pouvait parcourir toute la grille. Nous avons donc modifié notre programme pour que tous les champs « minesproxi » soient initialisés dès que les mines sont placées aléatoirement dans la grille.

Le fait de compter le nombre de mines à proximité nous a également pris beaucoup de temps. En effet le code devait détecter les bornes de la grille pour ne pas conduire à un plantage du programme. Nous avons donc distingué 9 cas : le cas où une case est sur une bordure (haute, basse, gauche, droite), le cas où une case est dans un coin (haut gauche, bas gauche, haut droit, bas droit), enfin le cas où une case est dans la partie centrale du programme. Nous avons néanmoins pu réutiliser cette architecture pour la fonction qui dévoile les cases. D’où un gain de temps.

# Evolutions possibles

Nous nous sommes fortement mobilisés sur ce projet car nous avons tous les deux apprécié cette manière d’appliquer nos connaissances. Nous sommes tous les deux très fiers d’avoir réussi à coder ce démineur en ayant démarré notre apprentissage du langage C mis-Septembre.

Les objectifs de ce démineur étaient donc de proposer un démineur qui fonctionne. Quatre niveaux de jeu devaient être intégré ainsi que le respect des règles du démineur original. Nous avons donc réalisé ces trois principales exigences, ce qui nous a permis d’ajouter quelques fonctions supplémentaires.