Rapport de Projet

Bataille Navale

Par Mehdi Hafi et Jérémy Lafosse, élèves en L3 MIAGE à l'UEVE.

Introduction

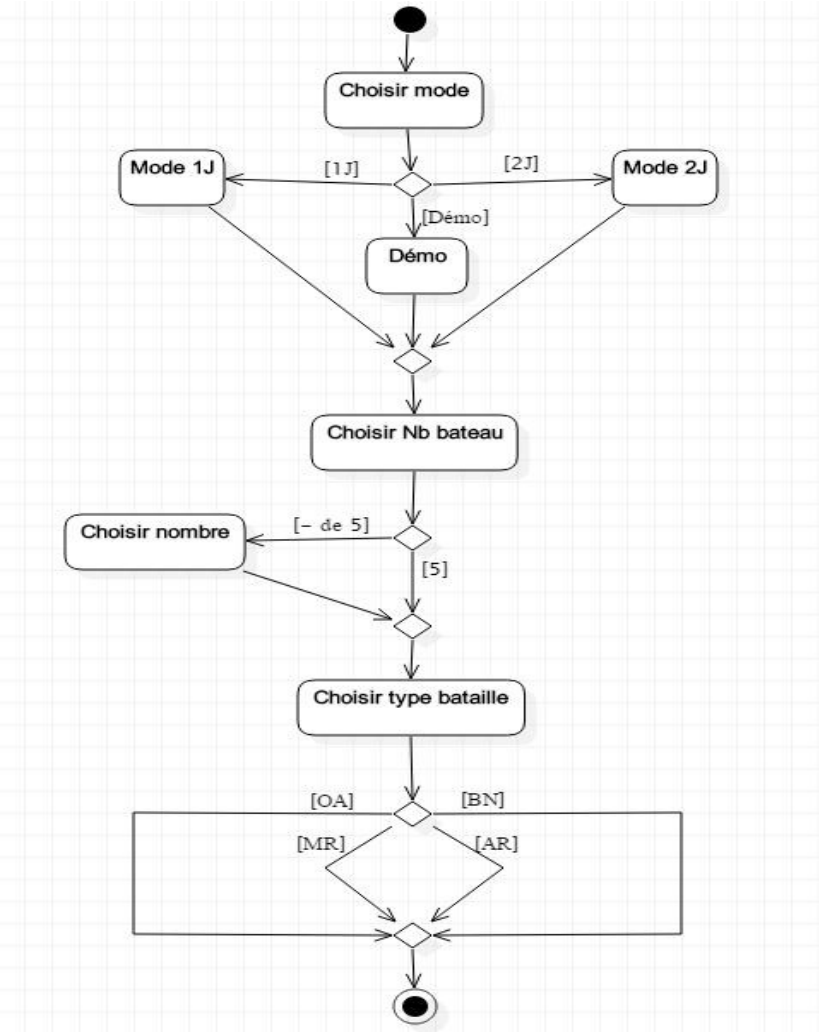
À travers ce rapport, nous aborderons notre projet de Bataille Navale accompli pendant ce second semestre.
Pour présenter ce projet, nous aborderons les différentes parties : dans l'ordre, la conception et modélisation, le développement et nous verrons ensuite les difficultés rencontrées.

Divers images seront présentes tout au long du rapport, telles que celles des Diagrammes, ou de l'interface graphique afin de compléter nos idées et explications.

Conception

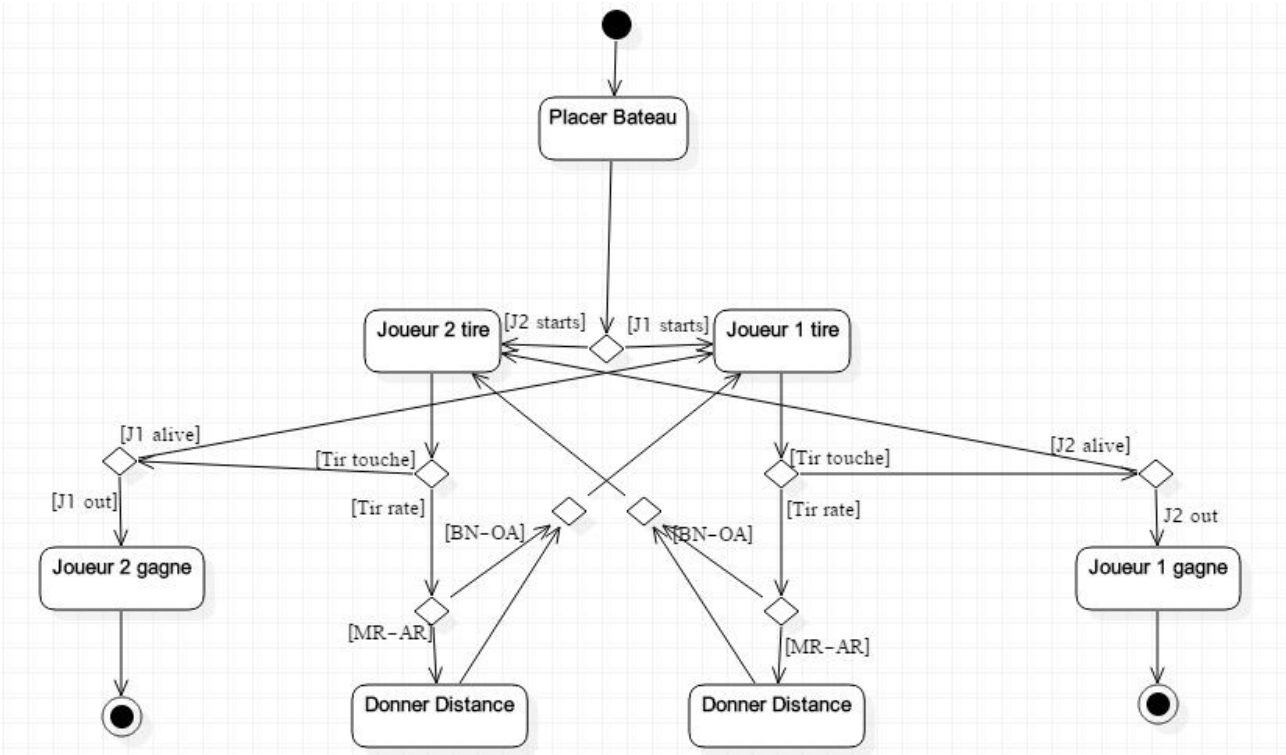
Dans cette partie, nous parlerons de la réflexion avant le développement. Nous avons fait divers diagrammes UML afin d'alléger la durée de réflexion du code.

Ils seront affichés un à un dans cette partie avec des commentaires sur le contenu, et les divers choses qui n'ont pas convenues au final.



Voici le diagramme d'activité du choix des paramètres. Pas grand chose à redire sur celui-ci, si ce n'est que malgré le fait qu'il y ai un sens d'entrée des paramètres sur le diagramme, au final, tous les paramètres sont choisis « au même moment » avec l'aide de jradiobutton et d'une fenêtre regroupant tous les choix.

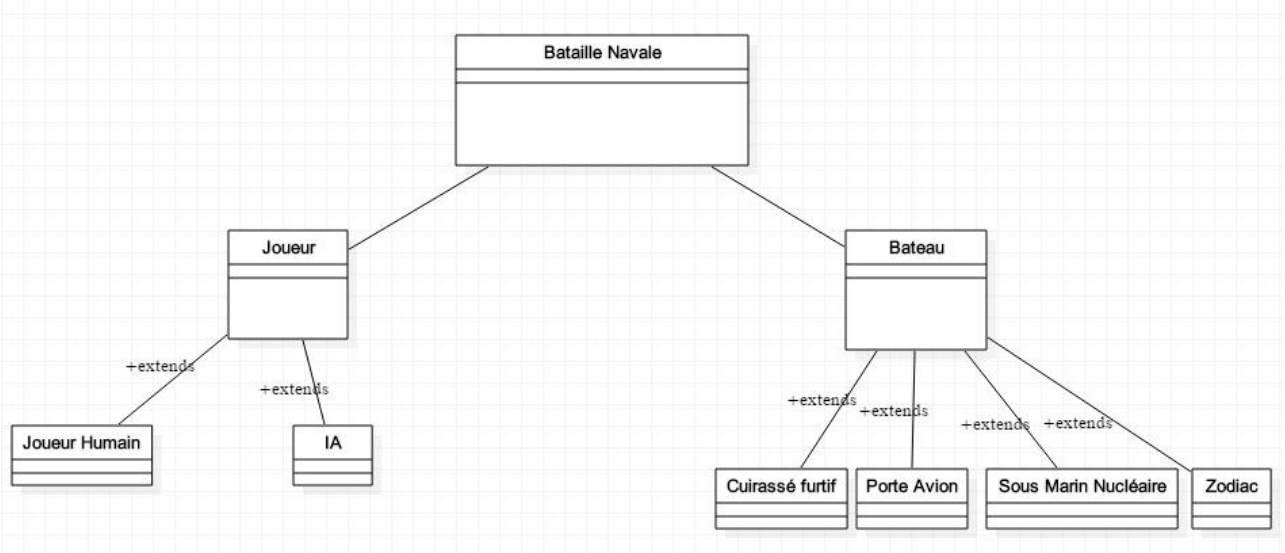
Les abréviations OA, BN, MR et AR sont les différents modes de jeu.



Nous avons ici un diagramme d'activité encore, mais celui du jeu. Les abréviations sont toujours les mêmes.
Pas beaucoup de détails nécessaires ici, une fois les bateaux placés, la partie démarre. Nous avons sûrement oublié de placer une case qui détermine aléatoirement quel joueur commence (ou alors démarrer directement sur Joueur 1, si jamais on veut toujours commencer sur celui-ci).

Il a fallu faire une boucle qui fut compliquée à retranscrire afin d'effectuer la transition de changement de tour.

C'est sûrement le diagramme dont le fonctionnement est le plus proche du fonctionnement optimal de la bataille navale.



Ceci est le Diagramme de Classe de notre projet. N'ayant pas énormément d'expérience en interface graphique, nous avions imaginé les classes telles quelles.

Le diagramme de classe du projet est celui qui s'éloigne le plus du résultat final.

En dehors de la partie Bateau, la partie Joueur finale n'a rien a voir, et il manque quelques classes. Il a fallu créer certaines classes pour des choses telles que le sens des bateaux dans la grille, chose que nous n'envisagions pas avant de nous impliquer dans l'interface graphique.

Avec plus d'expérience, nous aurions très probablement fait un Diagramme de classe qui s'approche plus du résultat final.

D'autre diagrammes étaient faisables, mais la plupart faisaient doublons avec ceux déjà faits, ou n'étaient pas (ou peu) nécessaires au développement, donc nous sommes directement passé à la réflexion et au code.

Développement

Dans cette partie, nous parlerons du procédé que nous avons suivi lors du développement, et de différents bouts de codes jugés intéressants qui seront commentés.

Nous avons codé en Java, car les autres langages autorisés ne nous étaient pas familiers, même si certains, comme le C#, semblaient plus utiles.

Nous avons préféré commencer par nous familiariser avec la partie graphique avant d'aborder le reste de manière à ne pas perdre de temps sur des incohérences par la suite. Cela nous a pris pas mal de temps, car nous ne savions pas quel type de boutons utiliser et nous avons du apprendre les propriétés de chacun. Nous sommes passés d'une fenêtre par paramètre à une grande fenêtre contenant tous les paramètres de la bataille navale.

En utilisant des ButtonGroup regroupant chaque Jradiobutton afin d'éviter qu'on ne coche plusieurs fois les mêmes paramètres, ce fut vite réglé.

Faire la partie « en jeu » a été plus dur, notamment la partie grille mais à la suite de nombreuses recherches et essais, de discussions avec nos camarades, nous avons fait un tableau 10x10 plutôt qu'essayer de le faire uniquement de manière graphique.

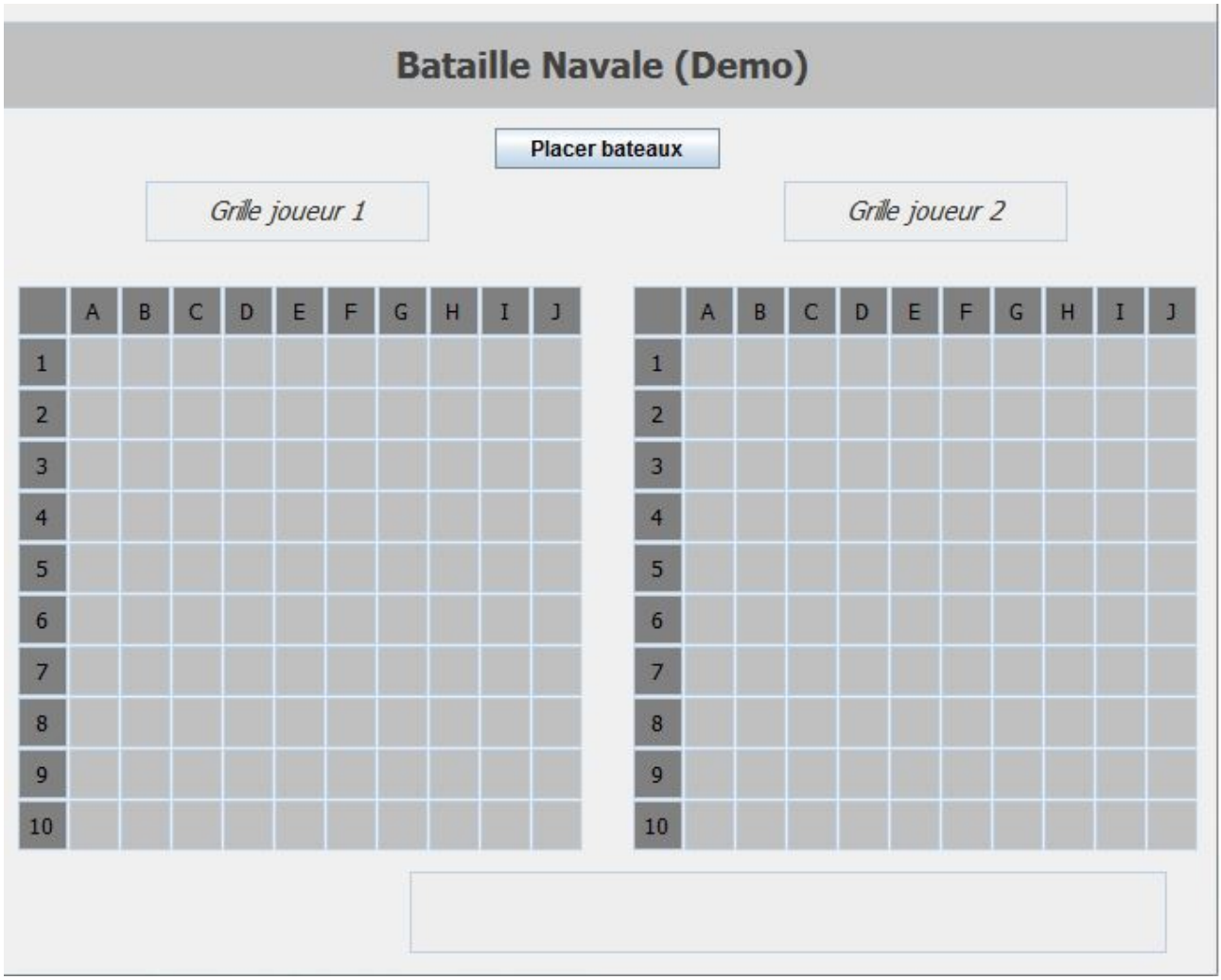
Notre soucis fut ensuite la direction des bateaux lors du placement aléatoire de l'IA. Nous avons investi beaucoup de temps sans succès dans différents moyens de placement, voici notre solution finale :

```
public enum Direction {
	VERTICAL,
	HORIZONTAL;

	public static Direction getRandom(){
    return values()[(int) (Math.random() * values().length)];
}
```

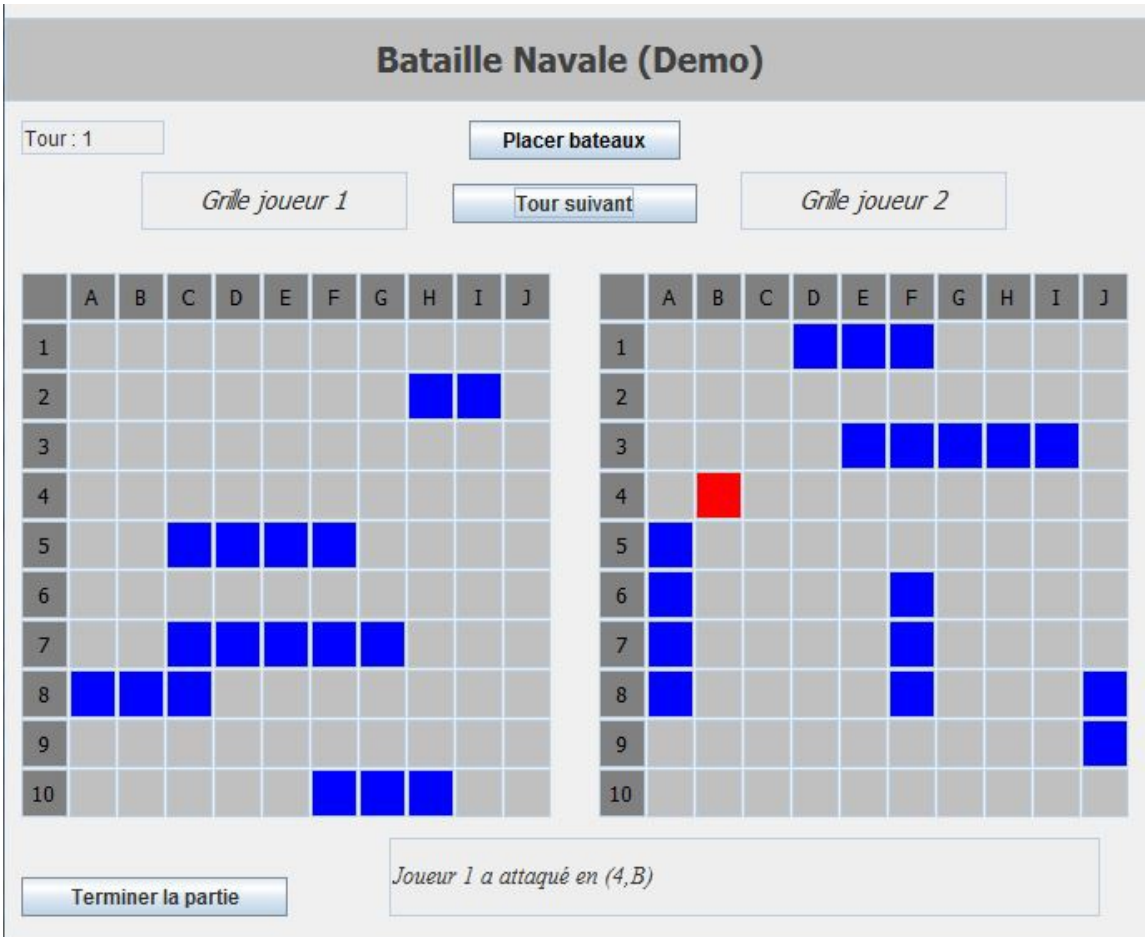
Après des séries de tests sur l'IA, nous n'avons constaté aucun soucis, les bateaux ne sortent pas du tableau et se placent toujours correctement.

Voici un rendu du projet en jeu mode Démo avant le lancement d'une partie :



Il y a une grille pour chaque joueur, un bouton pour initialiser le placement des bateaux. Le rectangle du bas contiendra des informations sur qui a tiré, et d'autres informations sur la partie (ce bloc pourrait être utilisé pour donner des informations selon le type de partie choisi par exemple).

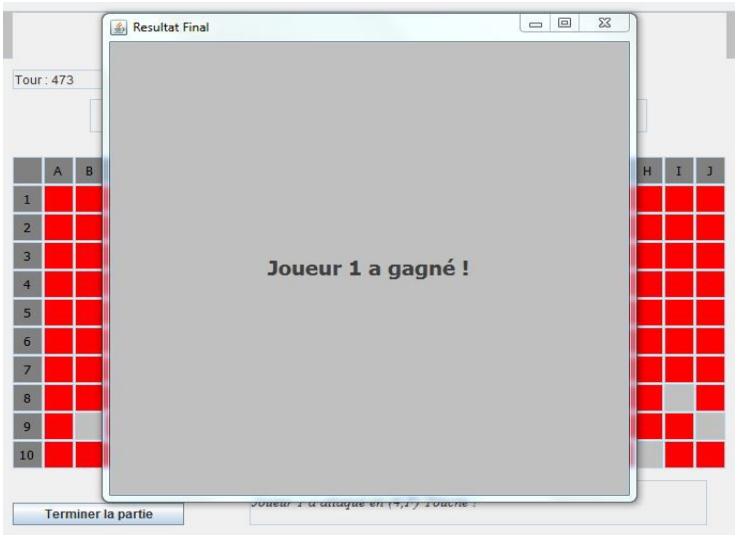
Et maintenant le même mode, mais une fois le premier tir effectué :



C'est déjà plus complet. Le bouton « Placer bateaux » est à retirer, ou à griser, car il n'est plus utile (il sera certainement modifié d'ici la version finale, le code continue d'avancer). Un bouton « Tour suivant » est apparu, qui fait tirer directement le joueur suivant. En bleu, vous avons les bateau placés automatiquement et en rouge, une case touchée par un tir.

Un compteur du nombre de tour est apparu en haut à gauche, et un bouton « Terminer la partie » est en bas à gauche. Ce bouton effectue automatiquement le nombre de coups nécessaires pour terminer la partie, et affiche automatiquement le gagnant final.

Un des soucis restant (notre plus important) est le fait que l'IA peut tirer plusieurs fois au même endroit. Si l'on veut terminer la partie manuellement via le bouton « Tour Suivant », cela peut prendre très longtemps pour finir une partie, car quand il ne reste qu'une case à toucher, il y a une chance sur 100 pour toucher cette case. Une fois ce soucis réglé, les parties ne dureront pas plus de 100 tours alors qu'elles durent plus du triple actuellement.



Difficultés rencontrées

Nous avons fait face à de nombreuses difficultés. Tout d'abord sur l'interface graphique où le manque de connaissance s'est fait très vite ressentir, en effet nous n'avons jamais eu de cours sur la création de celles-ci, nous avons dû improviser et beaucoup nous renseigner sur l'utilisation et la création de celle-ci.. Nous avons perdu beaucoup de temps à chercher des tutoriels sur Internet, et à chercher des solutions à de nombreux petit soucis liés au graphique. Deuxièmement, nous avions tout les deux un emploi du temps très chargé hors de l'université, ce qui n'a pas facilité la communication et le travail « continu ».

Conclusion

Avec du recul, nous avons avancé trop lentement en début de projet pour ensuite se retrouver à devoir régler plusieurs soucis en un intervalle de temps très court. Nous aurions mieux fait de commencer à développer des classes métiers une séance ou deux plus tôt, sans perdre de temps sur les diagrammes, ou en nous séparant mieux les tâches de manière à économiser notre temps.

Ce projet nous a aidé à mieux comprendre le développement en Java, mais également à comment aborder un projet, développer l'esprit d'équipe et éviter les erreurs dans la gestion du temps. Nous avons également progressé en matière d'interface graphique.

Ce projet qui nous paraissait de prime abord le plus sympathique à faire (le coté jeu, sûrement) s'est révélé être plus complique que d'autres qui semblaient plus rébarbatifs tel que celui sur la gestion de stage.

Aussi, l'usage de Github semblait pratique au premier abord, mais il s'est révélé superflu étant donné que nous étions par groupe de 2. Nous aurions sûrement plus apprécié ce support si nous étions par groupe de 4-5, minimum, sur un projet plus grand et certainement plus long.

ci-dessous le tableau d'avancement de projet !

Date	Réalisé	Contraintes	Tâches à réaliser	Durée prévue	Durée effective
13/01/2016	Aborder le sujet		Aborder le sujet	1 Séance	1 Séance
20/01/2016	Squelette de Diagramme		Commencer les diagrammes	1 Séance	2 Séance
27/01/2016	Diagramme de classe		Diagramme d'activité	2 Séance	2 Séance
03/02/2016	Diagramme d'activité		Commencer/Terminer le code	6 Séances	9 Séances (grâce aux 3 ajoutées au final!)
10/02/2016	Diagramme d'activité				
17/02/2016	Aborder le code	Appréhension de la partie graphique			
24/02/2016	Avancée des classes/fenêtre				
02/03/2016	Avancée des classes/fenêtre				
16/03/2016	Menu des paramètres				
23/03/2016	Ecran « ingam e »				
30/03/2016	Résolution de certains soucis d'IA	Manque de connaissances sur le sujet			
07/04/2016	Résolution de certains soucis d'IA				

Les cases vides signifient qu'elles sont identiques à celles du dessus.