TSAMEZA Armand Groupe 14

THSIAMA Pierre  
NOEL Jonathan

Rapport : Bataille Navale

*Dans le cadre du cours de « Développement informatique avancé : orienté application »*

EPHEC 2016 - 2017

Table des matières

Sommaire ..........................................................................

Introduction........................................................................

Cahier de charge et mode d’emploi du jeu.............................

Diagramme UML et explication.............................................

Explication du choix de l’implémentation...............................

Description des difficultés rencontrés....................................

Les pistes d’améliorations du jeu…………………………………………

Conclusions personnels et générales......................................

**Introduction**

Lors de nôtres deuxième année d’étude en informatique, il nous est demandé dans le cadre du cours de développement informatique orienté application de réaliser un projet ; ceci étant au choix, nous avons opté pour le jeu « Bataille Navale ». L’objectif de ce projet était d’utiliser progressivement les notions théoriques d’un langage de programmation orienté objet, Java, afin de développer l’application.

Mais que signifie bataille navale ? quels sont les différents points et les différentes notions qui constituent nôtres application ? La suite de ce rapport aura pour objectif de répondre à ces questions.

Cahier des charges

L'application "Bataille Navale" est un jeu à 2 joueurs, l'un étant un Humain, l'autre l'Ordinateur.

Chaque joueur dispose de plusieurs bateaux placés sur une grille de 10x10 cases qui lui est propre.

A tour de rôle, les joueurs indiquent une case à attaquer de la grille de leur adversaire, afin de tenter de couler les bateaux adverses. Dès que l'un des joueurs a coulé tous les bateaux de son adversaire, il a gagné la partie.

***1ère étape du jeu : le placement des bateaux***

Pour chacun des 2 joueurs, l'application place aléatoirement des bateaux sur une grille de 10x10 cases qui lui sont propre. Pour chaque joueur, les bateaux à placer sont les suivants :

● 1 porte-avion de 5 cases

● 1 croiseurs de 4 cases

● 1 frégates de 3 cases

● 1 zodiacs de 2 cases

Les cases des bateaux doivent bien entendu être en ligne droite. Deux bateaux ne peuvent évidemment pas se superposer, par contre ils pourront être placés l'un à côté de l'autre (avoir des côtés communs).

***2ème étape du jeu : la partie proprement dite***

Le joueur Humain commence la partie en indiquant une case où il joue.

Il s'agit pour lui d'indiquer une case située sur la grille de l'Ordinateur, case qu'il "attaque". Si un bateau est présent sur cette case, l'application signale qu'un bateau est touché. Si aucun bateau n'est présent, l'application signale qu'il s'agit d'un coup "à l'eau".

Puis le joueur Ordinateur, à son tour, "attaque" une case du joueur Humain, en suivant le même principe.

***Fin du jeu***

Dès que l'un des 2 joueurs a touché toutes les cases de tous les bateaux de son adversaire, ce joueur est le gagnant de la partie : l'application signale que ce joueur a gagné, puis se termine.

***Utilisation***

Le jeu sera utilisable non seulement en interface console, mais aussi sur en interface graphique qui aura comme taille +- 600px et qui aura l’avantage d’avoir une vue plus détaillée du jeu.

Le jeu pourra être joué par deux personnes via deux terminaux différents.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I | J |
| 1 |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |

Diagramme UML

Untitled Diagram.png

Explication du diagramme UML

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.  Il montre la structure interne et permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation ainsi comme on peut le voir sur le diagramme UML de notre application il s’articule autour de trois classes principales à savoir :

* La classe bateau sur laquelle on associe la taille et donc les classe Zodiac, Frégate, Croiseur, et Porte-avions héritent.
* La classe carte mère Carte avec ses classes filles CarteController et CarteView ; et, le bateau est associé à la carte car c’est sur la carte qu’on dessine les bateaux.
* La classe InteractionUtilisateur à laquelle on associe la classe joueur et donc les classes InteractionController et InteractionView héritent. Et, ces interactions utilisateurs sont associés à la classe bateau et carte car c’est sur ces classes qu’il y aura des actions de l’utilisateur.

Explication du choix de l’implémentation

Comme l’explique l’UML, notre application s’articule sur plusieurs volets à savoir :

Premier volet consiste à la construction du plan de jeu qui ici est la carte ; pour s’y faire nous avons défini une classe mère « Carte » permettant d’instancier deux cartes, et à travers les autres classes héritant de la classe mère nous avons pu construire et afficher le plan de jeu en console ; il s’agit ainsi de la classe « CarteControlleur » qui a pour objectif non seulement de dessiner les bateaux sur les cartes (ceci par la méthode draw Bateau()) , mais aussi de par un Radom de générer les tirs de l’adversaires et d’afficher physiquement les cellules attaqués « showAversaireAttaque() » ; et aussi de la classe « CarteView » qui par la méthode « afficheControlleur() » , présente tous les actions effectuer sur les cartes.

Deuxième volet est la création des bateaux. Comme nous l’avons précisé sur le cahier des charges, les bateaux sont de taille différente et un bateau peut être placé de manière horizontale ou verticale sur la carte ; ainsi pour avoir la structure bateau, nous créer les classes suivantes :

-la classe « Taille » qui permettait d’avoir les coordonnées du bateau ; une classe mère « Bateau » à laquelle s’associe la taille ; et, permettant de préciser l’orientation du bateau l’action sur un bateau. Aussi, a cette classe mère bateau, sont associé des différentes classe fille permettant de spécifier les différentes longueurs des bateaux. Il s’agit de « Zodiac » avec une longueur de deux cellules, de « Frégate » avec une longueur de trois cellules, de « porte-avion » avec une longueur de cinq cellules, de « Croiseur » avec une longueur de quatre cellules.

Troisième volet porte sur l’interaction utilisateur. Cette rubrique avait pour objectif de contrôler toute les entrer de l’utilisateur ceci par les classes « Joueur » « InteractionUtilisateur », « interactionController », et « interactionView ». Ainsi pour jouer l’utilisateur commence par choisir un nom d’utilisateur d’où la création de la classe « Joueur » ; la classe « interactionController » s’articule sur la découpe des différentes cellules, la vérification des lettres et des numéros choisis pour la cellule car celle-ci était compris entre « A et J » pour les lettres et entre « 1et 10 » pour les chiffres.

Avec ce qui précède, on voit que notre application est centrée sur trois classe principale à savoir : la classe « Bateau », la classe « Carte », et la classe « interactionUtilisateur » ; et, en combinant toute ceci dans une classe de test appelle jeu, nous avons pu réaliser entre autres une interface console. Et à ce stade, l’utilisateur ne peut jouer qu’avec un adversaire qui est l’ordinateur ; mais que se passe-t-il si deux utilisateurs veulent jouer ? Ainsi entre la notion de socket.

Les sockets servent à communiquer entre deux hôtes appelés Client / Serveur à l'aide d'une adresse IP et d'un port ; ces sockets permettront de gérer des flux entrant et sortant afin d'assurer une communication entre les deux (le client et le serveur), soit de manière fiable à l'aide du protocole TCP/IP, soit non fiable mais plus rapide avec le protocole UDP. Nous allons utiliser le premier mode, le mode TCP/IP. Dans notre cas l’un des joueurs est considéré comme serveur et l’autre comme client. L’un des joueurs(serveur) attend une connexion sur un port bien défini ; lorsque l’autre joueur(client) initie une connexion depuis un de ces port vers le port défini de l’autre joueur(serveur), le joueur(serveur) accepte la connexion et la partie peut commencer entre les deux joueurs en outre les attaques des bateaux peuvent commencer entre les deux joueurs distants.

A ce stade l’application fonctionne plus ou moins bien en interface console ; mais en accord avec le cahier de charge, il fallait implémenter une vue graphique pour une visualisation plus physique du jeu par un utilisateur, et pour s’y faire, nous avons utilisé l’interface graphique SWING et associe à cet interface la notion de programmation évènementiel pour pouvoir gérer lors des attaques sur les bateaux les actions des différents joeurs.

Description des difficultés rencontrés

Selon les consignes du projet nous devions respecter plusieurs points parmi lesquelles l’architecture MVC (model vue contrôleur). Après lecture du cours théorique sur le MVC, nous avons cru développer notre application sur cette technique mais, après une esquisse d’exemple d’architecture MVC fournit par le professeur encadrant, nous avons constaté que notre application ne répondait pas du tout à cette architecture. Ensuite, après plusieurs lecture cette notion est toujours rester incompréhensible pour nous.

Les pistes d’améliorations du jeu

Au niveau des pistes d’améliorations envisageables, il y en a une infinité. Dans notre cas, nous avons réussi à mettre en œuvre une partie de la base du jeu, mais cela n’est qu’une petite partie de tout ce qu’il serait possible de faire.

Premièrement, et dans le cas le plus urgent, il serait bien d’améliorer l’interface graphique pour qu’il soit le plus « user-friendly » possible et de traiter les gros bugs.

Ensuite, nous avons pensés à ajouter une petite zone de discussion afin de permettre aux joueurs une certaine communication, cela est toujours plus agréable. De plus, nous savons qu’un projet portant sur la communication a été réalisé par une autre équipe, cela pourrait donc nous permettre de collaborer afin d’optimiser notre logiciel.

Conclusions personnelles et générales

* **Pierre Tshiama**

La réalisation de ce travail était très intéressante. J’ai eu l’opportunité d’apprendre à m’intégrer dans un groupe dans lequel il y avait différents niveaux d’expérience en programmation. Grace à notre travail d’équipe, nous avons su mettre nos connaissances en commun afin de réaliser un jeu qui s’appelle Bataille Navale. Durant le développement de l’application, pour ma part, j’ai rencontré des difficultés, mais la bonne qualité du cours et les explications sur les slides en lignes et lors des TPs m’ont bien aidé. Nous avons réussi à réaliser le jeu sur interface console, mais notre interface graphique pourrait encore être améliorée.

* **Noël Jonathan**

De mon côté, je trouve que le projet prenait vraiment beaucoup de temps. Après seulement quelques cours de Java, étant de la matière initialement basique, voilà un projet. Malheureusement, je trouve que nous n’étions pas assez prêts pour recevoir un projet aussi rapidement. De ce fait, j’ai pris une quantité énorme de temps à simplement essayer de comprendre le langage et à essayer des tas de commandes. Au jour d’aujourd’hui, voici le projet sur lequel j’ai vraiment appris des choses mais le Java n’est toujours pas mon langage de prédilection. Ne parlons même pas des GUI qui étaient horribles à mettre en place.  
Pour terminer, je trouve que ce projet m’a pris énormément de temps mais c’était quand même une bonne expérience.

* **Armand Tsameza**

Au terme de ce projet bien que cette matière fût des connaissances nouvelles pour notre bachelier, je dirai que, j’ai aussi pu apprendre beaucoup de choses notamment le travail en groupe, comment mener à bien un projet ; et, de tous ceci, je dirai ce fut une bonne expérience pour ce langage même si pas mal de notions reste pour moi incompris.

Conclusions générales

Tout d’abord, nous souhaitons remercier nos encadreurs pour l’aide qu’ils nous ont apporté tout au long de ce quadrimestre pour ce projet ; pour avoir répondu à nos questions et pour avoir eu la gentillesse de nous enseigner ce nouveau langage de programmation. Le bilan du projet par rapport à ses objectifs est en demi-teinte. Nous n’avons certes pas rempli l’ensemble des fonctions attendues au cahier des charges il s’agit entre autres de : l’architecture MVC (model vue contrôleur) ; néanmoins, nous avons indéniablement beaucoup appris, depuis l’avancé de l’applications, jusqu’à l’aperçu que nous avons pu avoir de ce à quoi ressemble un mini projet. Il a été l’occasion de découvrir de nouvelles notions informatiques en langage de programmation orienté objet ; et, fessant ainsi la continuité avec celui étudier en 1ère année notamment le java script. Nous avons de nouveau constaté qu’il était parfois difficile de respecter les consignes demandées par l’équipe encadrante car des imprévues sont toujours possibles. En toute on peut dire que cela fut une très bonne expérience.