Regis.Clouard@ensicaen.fr
Julien.Mahier@ensicaen.fr
Sylvain.Vernois@ensicaen.fr

Jeu de régate en réseau

Ce projet vise à la mise en œuvre des techniques de génie logiciel pour la réalisation d'un projet concret. Une attention particulière sera donc accordée à la qualité de la conception de votre logiciel. Il est difficile dans le cadre de séances de travaux pratiques de proposer un projet d'une taille suffisante pour se rendre compte qu'une conception mal structurée conduit inévitablement à un logiciel rigide, fragile et immobile qui vire à l'usine à gaz. Compte-tenu des contraintes de temps, nous ne pouvons que proposer un projet modeste et qui pourrait être conçu sans compétences particulières en génie logiciel. Néanmoins, dans un but pédagogique, il vous est demandé de bien vouloir jouer le jeu de la gestion de projet et de livrer un logiciel démontrant des qualités de robustesse, extensibilité et réutilisabilité, accompagné de ses tests unitaires. Un projet ne présentant pas toutes les fonctionnalités demandées mais conçu selon les règles de l'art sera beaucoup mieux évalué qu'un projet complètement fonctionnel mais bricolé.

1 Le sujet

Vous venez de créer une entreprise dans le domaine du jeu vidéo avec une offre spécialisée dans les jeux en réseau. Un client (rôle joué par votre encadrant de TP) qui représente un célèbre fabricant de voiliers vous demande de concevoir un jeu vidéo de régates virtuelles en réseau pour leur activité promotionnelle.

1.1 Description du jeu de voile

Le jeu simule une régate de voiliers sur un parcours fermé, délimité par des bouées tel qu'un parcours olympique présenté en Figure 1.

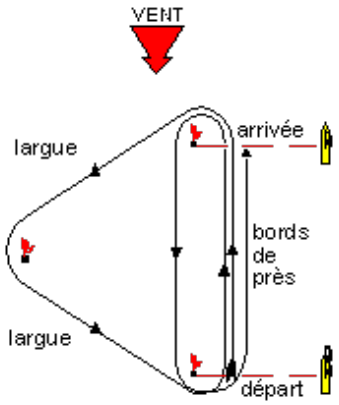

Figure 1: Parcours olympique.

Le plan d'eau est représenté par une carte qui contient tous les éléments du jeu :

- de l'eau,
- les bouées de marquage du parcours,
- le trait de côte,
- les voiliers.

Le plan d'eau est caractérisé par la présence d'un vent qui est identique en tout point du plan. Le vent est décrit par sa vitesse en nœuds et sa direction par rapport au nord.

Tous les voiliers sont identiques. Ce sont des Figaros Bénéteau de première génération. Ce bateau possède des caractéristiques propres qui définissent sa vitesse de déplacement en fonction de sa position par rapport au vent.

C’est ce que l’on appelle les *polaires de vitesses*. C’est un tableau qui donne pour chaque vitesse de vent réel, la vitesse de déplacement du bateau en fonction de sa direction rapport à la direction du vent réel en considérant un réglage optimal des voiles. On parle de vent réel par opposition au vent apparent qui est le vent mesuré sur le bateau et qui est la somme vectorielle du vent réel et du vent généré par le déplacement du bateau.

Dans le jeu à développer, le joueur ne peut influer que sur la direction de son bateau. On considérera que le réglage des voiles est toujours optimal. Le but du joueur est donc d'optimiser sa route. À chaque instant, le moteur de jeu calcule la nouvelle position de chaque voilier en course à partir de l’ancienne position, de la direction donnée par le joueur et de la vitesse de déplacement déduite des polaires des vitesses. La direction d’un bateau est conservée tant que le joueur n’a pas donné de contre-ordre.

Il a naufrage quand le voilier heurte un autre élément de jeu comme le trait de côte, un autre voilier ou une bouée. Dans ce cas, le voilier est retiré de la régate en cours.

1.2 Exigences du client pour le jeu

Les exigences du client par rapport au logiciel de jeu sont les suivantes :

1. Une partie consiste en une régate à l'issue de laquelle sera établi un classement.
2. Un serveur centralise le jeu. Il initialise le plan d'eau à partir d’une configuration prédéfinie. Il contrôle le jeu en mettant à jour la position de chaque voilier et détermine le classement à l'issue de la régate.
3. Chaque joueur vient se connecter au serveur en tant que client. Le rôle du joueur est de contrôler la direction de son bateau. Il intervient quand il le souhaite pour changer le cap ; à défaut, le bateau garde le même cap.
4. Chaque joueur dispose d'une vue personnelle du plan d'eau courant montrant les conditions météorologiques, la position de tous les bateaux de la régate ainsi que de son compas indiquant sa direction de déplacement et sa vitesse de déplacement.

2 Organisation du projet

La réalisation et l'organisation du travail sont soumises à des contraintes que chaque groupe devra respecter.

2.1 Contraintes sur la réalisation du projet

- ☑ **JAVA** : Le langage d'implémentation sera le Java.
- ☑ **JAVAFX** : L'interface graphique se basera intégralement sur la bibliothèque JavaFX.
- ☑ **Test** : Le code source du projet doit s'accompagner de tests unitaires et de tests d'intégration programmés avec JUnit et potentiellement Mockito.
- ☑ **Gradle** : le projet sera géré par le moteur de production gradle.
- ☑ **GIT** : La gestion des versions sera réalisée avec Git. Le dépôt central sera localisé sur gitlab.ecole.ensicaen.fr. La récupération des livrables finaux sera faite par votre encadrant directement à partir du dépôt Gitlab dont vous lui communiquerez l'adresse.
- ☑ **UML :** Les modélisations seront réalisées avec le langage UML. La rédaction des diagrammes UML devra se faire avec un atelier de génie logiciel tel que ArgoUML (installé sur les machines) ou un autre atelier de votre choix.
- ☑ **Patrons de conception** : L'architecture devra bien entendu faire appel aux patrons de conception, dont la pertinence devra être justifiée dans le rapport et la soutenance.

2.2 Méthode de gestion de projet

Le projet sera réalisé par équipe de 8 personnes et se déroulera sur 8 séances de 2 heures.

Il vous est demandé d'utiliser une méthode de gestion de projet à base de **cycles de développement itératifs**.

- ➢ Dans notre cadre, une itération correspond à deux séances.
- ➢ Le projet commence par une première séance de définition des fonctionnalités du futur logiciel et une répartition globale sur les

itérations prévues. Cette phase associe normalement pleinement le client. Ici, votre client viendra simplement valider ce que vous aurez prévu.

- Chaque itération commence par la définition d'un sous-ensemble de tâches à réaliser (de granularité assez fine) qui correspondent aux fonctionnalités identifiées pour cette itération.
- Au cours de l'itération, les tâches sont réalisées par les développeurs de l'équipe avec une étanchéité la plus élevée possible entre les développeurs.
- Chaque itération se termine par une **démonstration** du prototype opérationnel montrant à votre client les fonctionnalités développées jusque-là. Cette démonstration est un moment d'évaluation par votre encadrant de votre capacité à concevoir le logiciel demandé. Elle doit donc être préparée et réalisée de manière formelle. Le trop fameux « effet démo » est ici une erreur qui sera sanctionnée.
- La démonstration s'accompagne alors d'une rétrospective qui vise à réviser le plan de développement, en supprimant ajoutant changeant les fonctionnalités de la liste prévue.

2.3 Rôles et fonctions dans chaque équipe

Chaque équipe doit répartir les responsabilités en nommant des personnes aux postes suivants :

- ☑ **Chef de projet** : son rôle est de coordonner les activités de l'équipe, de superviser les travaux et d'interagir avec le client.
- ☑ **Architecte** : son rôle est de concevoir l'architecture du logiciel, l'organisation de la conception en paquets et de prendre les décisions tactiques relatives à cette réalisation. Il a aussi en charge les tests d'intégration.
- ☑ **Développeur** : son rôle est de concevoir les classes et leur organisation pour réaliser les fonctionnalités qui lui sont attribuées. Le développeur doit aussi écrire les tests unitaires correspondants (*pourquoi pas en TDD*). Seules les classes réellement graphiques ne sont pas accompagnées de tests unitaires.
- ☑ **Responsable de version** : cet ingénieur est le responsable du

dépôt sur Gitlab. C'est lui qui fait l'intégration continue du travail des développeurs et réalise le prototype de démonstration qui sera présenté à l'issue de chaque itération. Il est aussi le garant de la propreté du code et de la présence des tests dans les contributions des développeurs.

Il est à noter qu'une même personne peut endosser plusieurs rôles et qu'un rôle peut être assumé par plusieurs personnes.

2.4 Livrables

Première séance

- ☑ **Analyse des risques** : la liste des risques majeurs de ne pas parvenir à développer le logiciel avec les moyens d'y remédier.
- ☑ **Prédiction des itérations** : une description de la liste des fonctionnalités prévues réparties les 4 itérations.
- ☑ **Dépôt git** : l'adresse gitlab du projet.

Rendu final

Différents documents, regroupés dans un rapport, devront être produits. Le rapport devra au moins contenir les documents suivants :

- ☑ **Analyse des risques** : reprise de l'analyse faite en première séance.
- ☑ **Analyse des besoins** : les cas d'utilisation du logiciel, certains détaillés si nécessaires.
- ☑ **Conception UML** : La modélisation du logiciel en UML où les patrons de conception seront mis en exergue.
- ☑ **Diagramme de paquet** : Il décrit vos choix d'organisation de la conception en paquets.
- ☑ **Code source** : Le projet gradle avec le code Java du logiciel et le code des tests unitaires et des tests d'intégration.

Cette liste n'est en aucun cas exhaustive et pourra être complétée avec d'autres documents qui vous paraîtront pertinents dans la limite du

raisonnable (au sens Agile du terme).

2.5 Soutenance

À l'issue des huit séances de travail, une soutenance sera organisée pour la validation finale du logiciel. La soutenance devra au moins faire apparaître clairement :

- ☑ L'organisation de l'équipe et la gestion de projet conduite.
- ☑ L'architecture de votre système.
- ☑ La réponse de votre conception à la robustesse, la réutilisabilité et la maintenance.

La soutenance sera conclue par une démonstration finale.

Si la planification de votre projet s'est avérée irréaliste et que l'avancement réel en est très éloigné, il vous est demandé de prendre du recul et d'analyser les causes du dysfonctionnement. Le travail d'une équipe qui saurait analyser des dysfonctionnements et proposer des solutions sera bien mieux apprécié que celui d'une équipe qui chercherait à dissimuler d'éventuelles déconvenues derrière un produit fini, même spectaculaire.

Lors de la présentation orale, tous les membres de l'équipe doivent présenter l'aspect du projet dont ils sont responsables. Il ne sera pas accepté qu'un seule personne présente l'ensemble du projet de l'entreprise.

2.6 Évaluation

La note de TP de chaque membre d'une équipe sera la moyenne des notes obtenues pour chacune des quatre évaluations suivantes :

- ☑ La perception du client sur votre capacité à produire le logiciel demandé.
- ☑ La pertinence de la conception.
- ☑ La « propreté » du code et la présence des tests automatiques.
- ☑ La qualité de la soutenance.

3 Ressources

Afin de vous aider dans la réalisation de votre projet vous trouverez sur la

plateforme pédagogique plusieurs ressources documentaires et matérielles.

3.1 Exemples de code Java

- ☑ Un projet gradle avec un exemple d'interface graphique en JavaFX basée sur le patron d'architecture MVP.
- ☑ Un exemple en Java de communication client-serveur utilisant les sockets.

3.2 Fichier de données

- ☑ Le fichier des polaires de vitesse pour le Figaro.

3.3 Documentation

- ☑ Le travail de développement de projet avec Git (Git Workflow).
- ☑ Les tests unitaires avec JUnit.
- ☑ Les doublures avec Mockito.
- ☑ Un exemple de compte-rendu de projet.