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

Jeu de régate

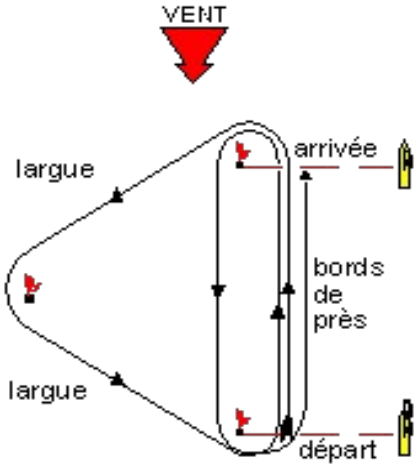
Ce projet vise la mise en œuvre des techniques de génie logiciel pour la réalisation d'un projet concret. Une attention particulière sera donc accordée à la qualité de la conception de votre logiciel. Il est difficile dans le cadre de séances de travaux pratiques de proposer un projet d'une taille suffisante 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. 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 pour leur activité promotionnelle.

1.1 Description du jeu

Le jeu est destiné à l'apprentissage de la navigation en voilier. Il n'y a donc qu'un seul joueur. Le but du joueur est d'optimiser sa route pour effectuer un parcours fermé, délimité par des bouées, tel que le parcours olympique présenté ci-dessous en moins de temps possible.

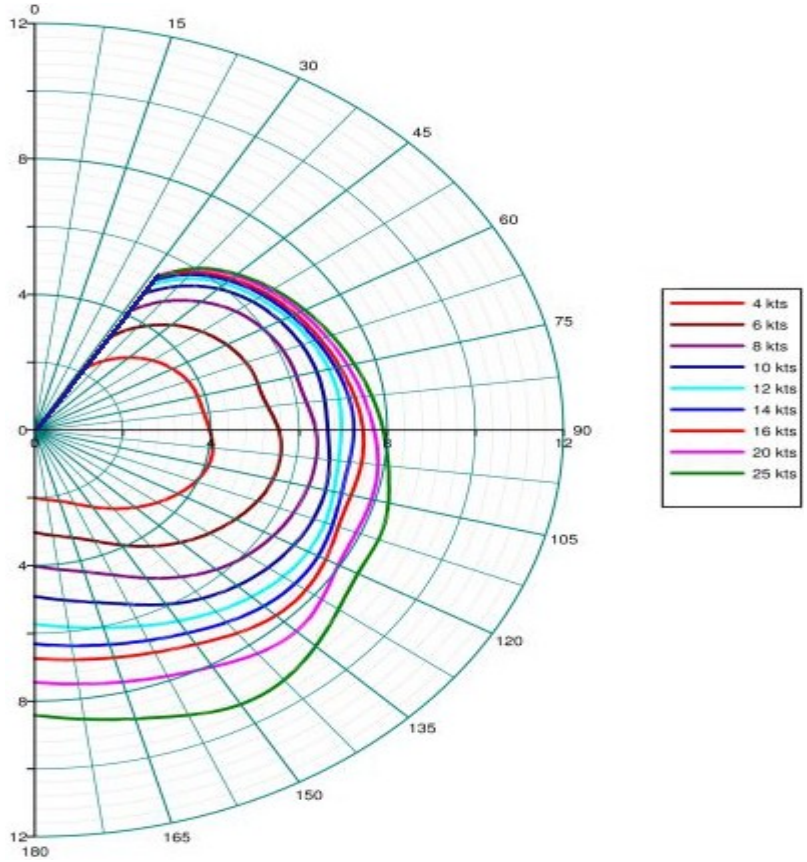


Le plan d'eau est représenté par une carte qui contient tous les éléments du jeu :

- 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.

Le bateau utilisé pour la régate est un Figaro 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 la *polaire de vitesses*. La polaire est une représentation graphique, exprimant la vitesse d'un bateau en fonction de la direction et de la force du vent. En général, ce graphique a la forme d'un diagramme semi-circulaire, car on considère les résultats comme identiques sur les deux amures.



Dans le projet, elle est donnée sous la forme d'un tableau qui indique 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. Par exemple, l'extrait ci-dessous indique qu'avec 3 nœuds de vent et un angle de 30° par rapport à la direction du vent, la vitesse atteinte est de 1,7 nœud mais avec un angle de 90°, la vitesse passe à 3,3 nœuds.

	3 nœuds	**10 nœuds**	**12 nœuds**	**20 nœuds**	**32 nœuds**
30°	1.7	4.2	4.4	4.7	4.6
90°	3.3	7.4	7.9	8.7	9.3

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. À chaque instant, le moteur de jeu calcule la nouvelle position du voilier à partir de l'ancienne position, de la direction donnée par le joueur et de la vitesse de déplacement déduite de la polaire des vitesses. La direction d'un bateau est conservée tant que le joueur n'a pas donné de contre-ordre.

1.2 Exigences initiales du client pour le jeu

1. Une partie consiste en une régate à l'issue de laquelle sera établi un score basé sur le temps de parcours.
2. Le jeu est initialisé à partir de configurations de plan d'eau et de

parcours prédéfinies et stockées dans des fichiers.

3. La vitesse et la direction du vent doivent être prises comme celles du jour sur le plan d'eau utilisé. Le site www.prevision-meteo.ch permet de récupérer les conditions météorologiques sous la forme d'un fichier Json. Par exemple l'url suivante permet de récupérer les conditions à Ouistreham (coordonnées : 49.283, -0.25) : https://www.prevision-meteo.ch/services/json/lat=49.283lng=-0.25 Les coordonnées géographiques du plan d'eau doivent donc être intégrées au fichier de configuration du plan d'eau.

4. 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.

5. De façon à marquer les virements de bords, le temps de virement sera pénalisé et la vitesse ramenée à 0. Cela peut, par exemple, se faire en diminuant l'angle de rotation donné par la barre et ainsi allonger le temps de virement.

6. Le joueur peut choisir deux caractéristiques d'armement du bateau : la taille des voiles et le nombre d'équipiers. On se contentera de deux valeurs pour chacune de ces options (surface de voile : grande, moyenne) et équipiers (2 ou 4). Ces valeurs influent sur deux performances du bateau : le temps de virement de bord et la vitesse. Plus il y a d'équipiers plus les virements de bord sont rapides et mieux le voilier remonte au vent, mais moins il avance vite. La taille de la voile permet d'aller plus vite mais aux allures de près, le bateau gîte plus et donc moins il va vite et moins bien il remonte le vent.

7. Le joueur doit avoir à sa disposition un tableau de bord indiquant le cap, la vitesse de déplacement, la force et la direction du vent.

8. À l'issue de la course, il doit être possible de rejouer automatiquement la régate pour ce bateau seul.

9. Quand le bateau touche une bouée ou un trait de côte, il se retrouve neutralisé pendant un temps fixé correspondant au temps d'une réparation de fortune. Il ne peut plus virer de bord ni avancer. Puis, le bateau reprend sa marche normale.

10. D'autres bateaux NPC (Non-Player Character) programmés à l'aide d'algorithmes d'IA participent à la régate. Il faut aussi mettre en place le respect des règles de priorité et gérer les conséquences du déventage d'un bateau par autre placé devant lui face au vent.

2 Organisation du projet

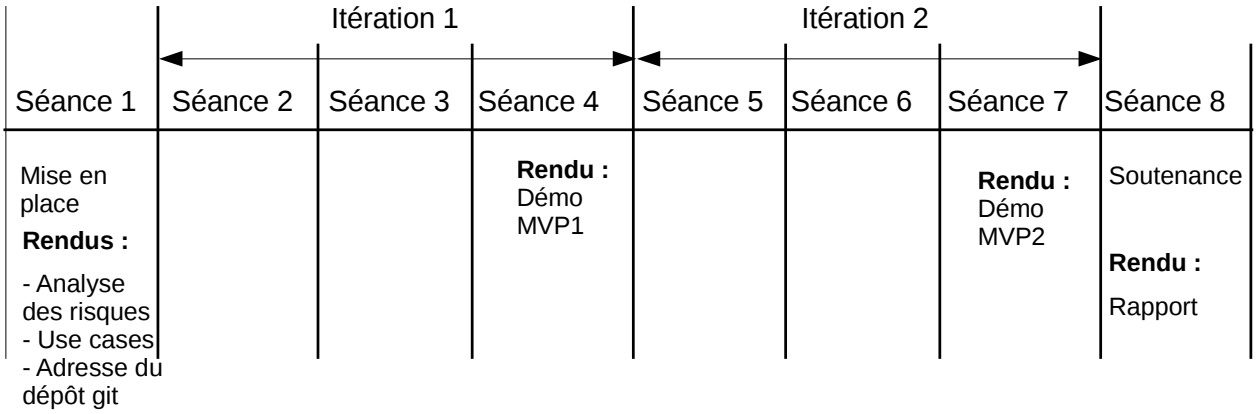
La réalisation et l'organisation du travail sont soumis à des contraintes impératives.

2.1 Contraintes sur la réalisation du projet

- ☑ **UML :** Les modélisations seront réalisées avec le langage UML.
- ☑ **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 devra s'accompagner de **tests unitaires** développés 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[1].
- ☑ **Gitlab CI/CD** : Vous devez mettre en place une intégration continue qui sera gérée par Gitlab. Pour cela, un fichier yaml (`.gitlab-ci.yml`) vous est fourni dans le squelette de projet à disposition dans les ressources sur la plateforme pédagogique.

2.2 Méthode de gestion de projet

Le projet sera réalisé par équipe de 7/8 personnes et se déroulera sur **8 séances de 2 heures**. Il vous est demandé d'utiliser une méthode de gestion de projet Agile à base de **cycles de développement itératifs et incrémentaux**.



1 Vous ajouterez aussi votre encadrant à votre dépôt avec le rôle de manteneur afin qu'il puisse suivre votre progression.

- Une itération durera 3 séances et donc il n’y aura que 2 itérations.
- Chaque itération commence par la définition d'un sous-ensemble de tâches à réaliser (de granularité assez fine) qui correspondent aux fonctionnalités identifiées pour le MVP de cette itération : le sprint backlog.
- Au cours de l'itération, les tâches sont réalisées par les développeurs de l'équipe avec une étanchéité la plus élevée possible entre eux.
- Chaque itération se termine par une **démonstration** du MVP auprès de votre client. 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 fréquent « effet démo » est ici une erreur qui sera sanctionnée.
- La démonstration s’accompagne ensuite d’une rétrospective qui vise à réviser le plan de développement en supprimant, ajoutant ou changeant les fonctionnalités de la liste prévue afin de définir le MVP de l’itération suivante.

2.3 Rôles et fonctions des coéquipiers

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.
- ☑ **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 de son code fonctionnel.
- ☑ **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

Premier rendu au début de la séance 2

Un rapport à mettre sur le gitlab :

- ☑ **Analyse des risques** : la liste des risques majeurs de ne pas parvenir à développer le logiciel et les moyens de les prévenir ou d'y remédier.
- ☑ **Cas d'utilisation** : l'analyse des besoins sous la forme d'un diagramme des cas d'utilisation.
- ☑ **MVP** : une description textuelle du MVP n°1.
- ☑ **Dépôt git** : l'adresse gitlab du projet.
- ☑ **Répartition des rôles** au sein de l'équipe.

Rendu final avant la séance 8

Un rapport à mettre sur le gitlab :

- ☑ **Analyse des risques** : reprise de l'analyse faite en première séance.
- ☑ **Cas d'utilisation** : le diagramme des cas d'utilisation du logiciel révisé.
- ☑ **Diagramme de paquet** : uniquement les paquets et leurs dépendances.
- ☑ **Conception UML** : la modélisation du logiciel en UML où les patrons de conception seront mis en exergue.
- ☑ **Code source** : le projet Gradle avec le code Java du logiciel et le code des tests unitaires.

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 sept séances de travail, une soutenance sera organisée pour la validation finale du logiciel. La soutenance sera découpée en

- 10 minutes de présentation,

- 5 minutes de démonstration
- et 10 minutes de question.

La soutenance devra au moins faire apparaître clairement :

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

La soutenance sera conclue par une démonstration finale.

Lors de la présentation orale, chaque membre de l'équipe doit présenter l'aspect du projet dont il est responsable. 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 :

- ☑ la pertinence de l'architecture et de la conception ;
- ☑ la « propreté » du code et des tests ;
- ☑ la qualité de la soutenance ;
- ☑ la perception du client sur votre capacité à produire le logiciel demandé.

3 Ressources

Afin de vous aider dans la réalisation de votre projet vous trouverez sur la plateforme pédagogique plusieurs ressources documentaires et matérielles.

L'ensemble des ressources proposées est conforme à l'IDE IntelliJ IDEA, que nous vous encourageons à utiliser.