

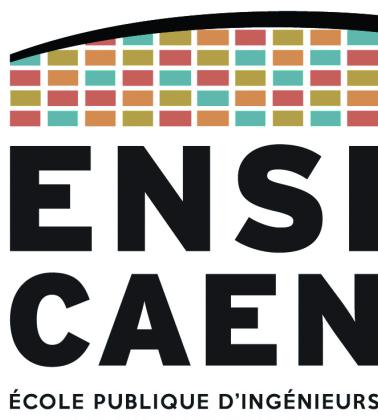
École Publique d'Ingénieurs en 3 ans

Rapport de fin de projet

# GÉNIE LOGICIEL ET PATRONS DE CONCEPTION PROJET RÉGATE

le 27 oct. 2022,  
version 1.1

Nom du projet : El Gama	Client : Régis CLOUARD
<a href="mailto:david.guo@ecole.ensicaen.fr">david.guo@ecole.ensicaen.fr</a>	Tuteur école : Régis CLOUARD
	<a href="mailto:regis.clouard@ecole.ensicaen.fr">regis.clouard@ecole.ensicaen.fr</a>



[www.ensicaen.fr](http://www.ensicaen.fr)

## Remerciements

Avant de présenter nos travaux, nous tenons à exprimer notre grande reconnaissance envers les personnes qui ont apporté leurs soutiens

Nous tenons aussi à remercier à notre encadrant M. Régis Clouard pour le temps qu'il nous a consacré pour la réalisation de ce projet, de nous avoir guidé au long de ce projet.

# Table des matières

---

Remerciements	2
<b>I Cadre du projet et méthodologie pour la gestion du projet</b>	<b>5</b>
1. Position du problème	6
2. Méthode utilisée pour la gestion du projet	6
3. Identification des acteurs	6
<b>II Analyse et spécification des besoins</b>	<b>8</b>
4. Analyse des risques	9
5. Étude des besoins	9
5.1. Besoin fonctionnel	9
5.2. Besoins non fonctionnels	11
6. Diagramme de cas d'utilisation	11
7. Diagramme de classes	12
<b>III Réalisation</b>	<b>14</b>
8. Architecture logicielle de la solution	15
9. Environnement de développement logiciel	15
10. Déroulement des sprints	15
10.1. Sprint 1 - Implémentation de la carte et du vent	15
10.1.1. Conception, objectifs et développement	15
10.1.2. Test relatif	16
10.2. Sprint 2 - Refactoring, correction de bugs et implémentation de la fin de course	16
10.2.1. Conception, objectifs et développement	16
10.2.2. Test relatif	16
11. Analyse des résultats produits	17

# Table des figures

---

1	Diagramme de cas d'utilisation	11
2	Diagramme de classe vue d'ensemble	12
3	Interface utilisateur : choix de la langue	17
4	Interface utilisateur : choix du voilier	17
5	Interface utilisateur : interface du jeu	18

## **Première partie**

# **Cadre du projet et méthodologie pour la gestion du projet**

## 1. Position du problème

Nous sommes une entreprise dans le domaine du jeu vidéo. M. Régis Clouard qui est un célèbre fabricant de voilier souhaite un jeu vidéo de régates virtuelles pour l'activité promotionnelle de son entreprise

Il fait appel à nos services pour concevoir ce jeu vidéo.

## 2. Méthode utilisée pour la gestion du projet

Nous sommes face à un projet où les données et les besoins du client sont incomplets au départ, notre équipe se dirige naturellement vers une méthode itérative.

Parmi les méthodes itératives, nous utiliserons une des méthodes agiles, qui sont celles utilisées par les ingénieurs logiciel de manière générale.

La méthode agile assure une meilleure communication avec le client et une certaine visibilité du produit livrable.

Parmi les méthodes agiles, nous adopterons la méthode scrum.

L'un des piliers de la méthode scrum est de développer un logiciel de manière incrémentale en listant de manière transparente une liste de tâches à effectuer.

Un autre pilier de la méthode est que l'on livre l'avancée du logiciel de manière fréquente (1 semaine pour notre cas) en faisant de l'intégration continue à chaque itération. C'est un logiciel 'fini' qui est livré à chaque itération. A la fin de chacune, le logiciel peut être considéré comme une version finale du produit. Plus on avance dans le projet, plus le logiciel est complet et plus il possède de fonctionnalités.

## 3. Identification des acteurs

La réalisation d'un projet débute par la détermination des différentes parties prenantes. On y distingue notamment :

- L'utilisateur : un client de l'entreprise du product owner
- Product owner (responsable produit) : Régis Clouard
- Chef de projet : David Guo
- Gestionnaires de versions : Baptiste Alix, Elliot Gaudron-Parry, David Guo
- Architectes : Gatien Bouyer, Marceau Combet

— Développeurs : Baptiste Alix, Elliot Gaudron-Parry, David Guo, Gatien Bouyer, Marceau Combet, Kylian Picque, Teo Dallier, Basil Courbayre

## **Deuxième partie**

# **Analyse et spécification des besoins**

## 4. Analyse des risques

Risque	Gravité	Mesure de prévention
<b>Risque humain</b> : Absentéisme des développeurs dans le groupe : lié à la période de l'année (maladies ou soirées d'intégration)	Importante	Essayer de ne pas mettre de séance après une soirée d'intégration
<b>Risque humain</b> : Conflits dans le groupe	Limitée	Organiser un événement en commun.
<b>Risque intrinsèque au projet</b> : mauvaise affectation des tâches	Importante	Demander conseil au responsable pédagogique
<b>Risque technique</b> : Mauvaise connaissance de framework tel que JavaFX	Importante	Documentation
<b>Risque technique</b> : Absence du gestionnaire de version	Négligeable	Affecter plusieurs gestionnaire de version au projet
<b>Risque technique</b> : Absence de l'architecte	Négligeable	Affecter plusieurs architectes au projet

## 5. Étude des besoins

### 5.1. Besoin fonctionnel

Voici les exigences du client sur le jeu :

- Le logiciel doit être bilingue, français et anglais, selon le choix de l'utilisateur. Le français est la langue par défaut
- Une partie consiste en une régate à l'issue de laquelle sera établi un score basé sur le temps de parcours
- Le jeu est initialisé à partir du choix d'une configuration du plan d'eau et du parcours stocké dans un fichier

- Au début du jeu, le joueur choisit le type de son bateau qui conditionne la polaire des vitesses.  
Nous donnons ici les polaires des vitesses pour le Figaro de première génération et l'Oceanis 37
- La vitesse et l'orientation du vent doivent être prises comme celles de la prévision météorologique du jour sur le plan d'eau utilisé
- Les coordonnées géographiques du plan d'eau doivent donc être intégrées au fichier de configuration du plan d'eau. S'il n'y a pas de connexion, la vitesse et l'orientation doivent être prises au hasard
- 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
- De façon à marquer les virements de bords, le temps de virement sera pénalisé et la vitesse ramenée à 0
- Le joueur peut choisir deux caractéristiques d'armement du bateau : la taille des voiles et le nombre d'équipiers
- 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 apparent
- À l'issue de la course, il doit être possible de visionner la course qui vient d'être faite
- Le joueur est en compétition avec des bateaux pilotés par l'ordinateur. L'enjeu est ainsi d'être plus rapide qu'eux

## 5.2. Besoins non fonctionnels

- L'ergonomie et la convivialité : l'application fournira une interface conviviale et simple d'utilisation, qui ne requiert pas de pré-requis, c'est-à-dire qu'elle pourra être utilisée par n'importe qui (même des personnes n'ayant pas suivi de cours d'informatique)
- L'extensibilité : l'architecture de l'application permettra l'évolution et la maintenance (ajout ou suppression des fonctionnalités) d'une manière flexible, tel que le requièrent les méthodes agiles

## 6. Diagramme de cas d'utilisation

Ce diagramme décrit les cas d'utilisation de notre logiciel.

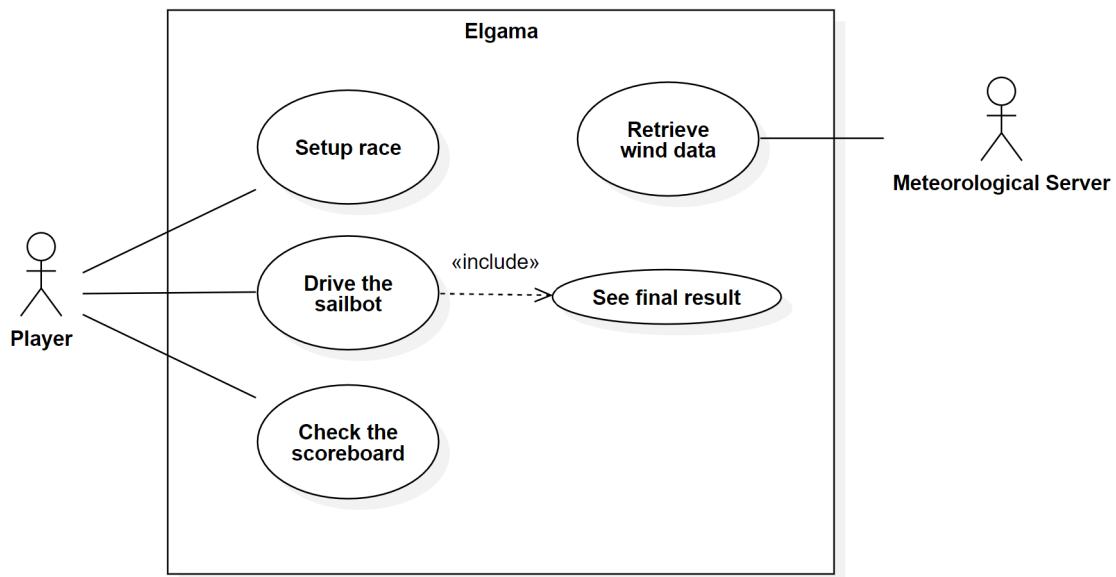


Figure 1. Diagramme de cas d'utilisation

On peut voir sur le diagramme ci-dessus que l'utilisateur devrait avoir la possibilité de faire trois actions principaux. En effet, il peut paramétriser la course, la jouer et enfin vérifier son score. Lorsque la course se lance, il faut que les données sur le vent soient renseignées. Ceci se fait par un intermédiaire externe au programme qui est un serveur.

## 7. Diagramme de classes

Ce diagramme décrit les classes de notre logiciel.

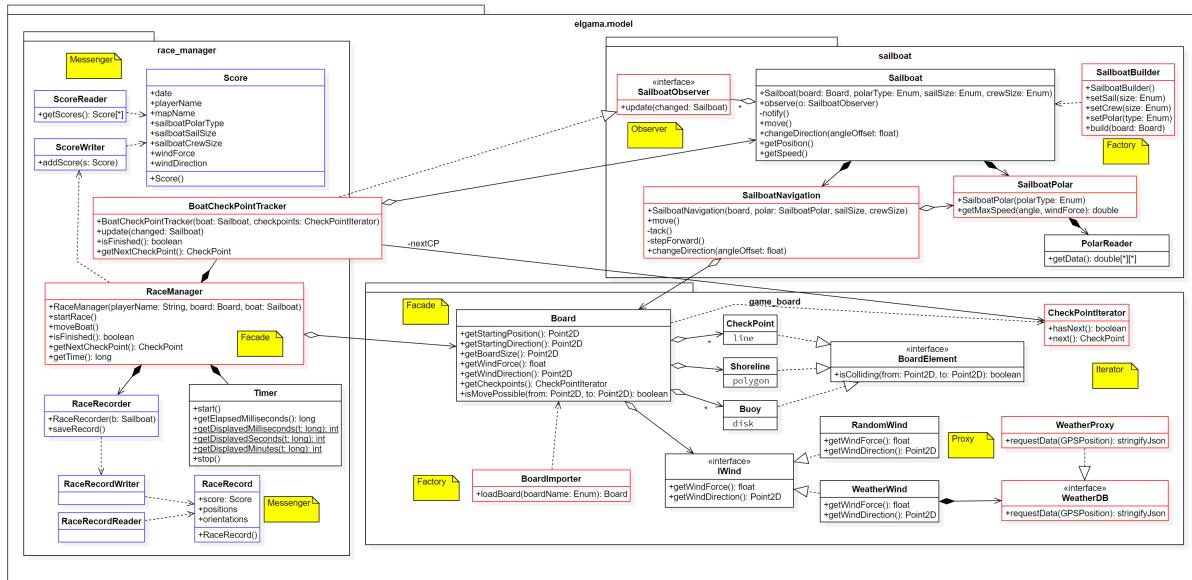


Figure 2. Diagramme de classe vue d'ensemble

La figure 2 présente ci-dessus les classes qui ont servi à l'implémentation de notre application : en bleu les classes qui n'ont pas été implémenté, en noir celles qui sont implémentées et en rouge les dernières classes implémentées ou en cours d'implémentation par manque de temps.

### 1. Dans le package race\_manager :

- La classe Score qui suit le patron **messenger** puisque cette classe ne possède pas de méthode et rend ses données publiques.
- La classe ScoreReader qui lit le score
- La classe ScoreWriter qui écrit le score
- La classe BoatCheckPointTracker qui recense l'avancée d'un bateau sur la liste des points de contrôle : si le bateau a passé un point de contrôle ou non
- La classe RaceManager qui est le directeur de course : il va implémenter les méthodes qui vont être propre d'une course (départ, mouvement etc...)
- La classe Timer qui va implémenter un chronomètre
- La classe RaceRecorder qui va implémenter l'enregistreur de course
- La classe RaceRecorderWriter qui va permettre l'écriture de l'enregistrement de la course dans un fichier

- La classe RaceRecord (patron de conception Messenger) qui stocke les données d'un enregistrement d'une course
- La classe RaceRecorderReader qui va permettre la lecture d'un enregistrement d'une course depuis un fichier

## 2. Dans le package sailboat

- La classe Sailboat qui modélisera un bateau à voile
- La classe SailboatNavigation qui permettra le mouvement du bateau à voile
- La classe SailboatBuilder qui permettra de construire le bateau à voile. Cette classe reprend le patron de conception **Builder** afin de construire la classe SailBoat qui possède beaucoup d'attributs facultatifs.
- La classe SailboatPolar qui va lire la polaire du bateau à partir d'un angle et d'une force de vent
- La classe PolarReader qui va lire les coordonnées polaires à partir d'un fichier
- La classe Sailboat pour informer aux autres objets les déplacements du bateau à voile. Cette classe suit le patron **Observer** puisqu'elle permet de centraliser les informations concernant le bateau.

## 3. Dans le paquet game\_board

- La classe Board qui va modéliser la carte ou "planche" de jeu. Elle représente le patron **Facade** car elle permet de fournir une seule interface(celle de la planche de navigation) qui contient toutes les fonctionnalités de la carte et du bateau.
- La classe CheckPoint qui va modéliser un point de contrôle
- La classe Buoy qui va modéliser une bouée
- La classe Shoreline qui va modéliser un trait de côte
- La classe RandomWind qui va implémenter un vent aléatoire
- La classe WheatherProxy est un **Proxy** qui va permettre d'interfacer avec le serveur météorologique. L'interaction entre le proxy et le serveur se fait qu'une fois au lancement de la partie.
- La classe WheatherDB qui va acquérir les données météorologiques du serveur
- L'interface BoardElement qui va modéliser un élément de la planche
- La classe CheckPointIterator qui va permettre de parcourir la suite ordonnée des checkpoints sans exposer sa représentation
- L'interface IWInd qui modélisera le vent
- La classe WeatherWind qui va implémenter les conditions météorologiques du jour sur le plan d'eau utilisé
- La classe BoardImporter respecte le patron **Factory** qui permet de construire la "planche" de jeu.Ce patron nous est utile ici parce qu'il y a plusieurs types de planches de jeu mais que

la méthode pour les faire sont identiques.

# **Troisième partie**

# **Réalisation**

## 8. Architecture logicielle de la solution

A travers ce projet, nous adopterons une architecture de type Modèle - Vue - Présentateur (MVP).

Elle consiste à distinguer trois entités distinctes qui sont :

- Le modèle qui est totalement centré sur la logique métier
- La vue qui concerne la gestion graphique sans connaissance du modèle.
- Le présentateur qui contient toute la logique de présentation et qui fait le pont entre le modèle et la vue

Ces trois entités distinctes jouent donc un rôle précis dans l'interface

## 9. Environnement de développement logiciel

Pour pouvoir développer notre logiciel, nous utiliserons :

- Linux comme environnement de travail
- IntelliJ pour le développement de l'application
- TexMaker pour la rédaction de ce rapport
- SceneBuilder pour la création des vues
- Git avec Gitlab pour versionner le logiciel et mutualiser le code

## 10. Déroulement des sprints

### 10.1. Sprint 1 - Implémentation de la carte et du vent

#### 10.1.1. Conception, objectifs et développement

Les objectifs et conception pour ce premier sprint sont multiples, il s'agit tout d'abord :

- Implémenter un vent aléatoire
- Amélioration de l'interface graphique : page d'accueil avec le logo du groupe
- Implementations des éléments de la carte
- Implémentation de la polaire : lecture du fichier et lecture de la vitesse appliquée sans plus d'effet au bateau à voile

#### 10.1.2. Test relatif

- Ce qui s'est bien passé : l'interface graphique et l'implémentation des éléments de la carte
- Ce qui peut être mieux fait : Ce qui peut être mieux fait : Implémentation du vent aléatoire bugué et lecture de la polaire erronée
- Améliorations : correction du vent aléatoire et refactoring du code relatif au bateau à voile

### 10.2. Sprint 2 - Refactoring, correction de bugs et implémentation de la fin de course

#### 10.2.1. Conception, objectifs et développement

Les objectifs et conception pour ce second sprint sont multiples, il s'agit tout d'abord :

- D'effectuer la correction du vent aléatoire et faire un refactoring du code relatif au bateau à voile
- Implémenter un chrono
- Implémenter la collision du bateau avec les obstacles
- Implémenter le choix de la langue pour l'interface graphique utilisateur
- Implémentation des checkpoints
- Implémentation de la fin du jeu lorsque l'utilisateur a traversé tous les checkpoints
- Améliorer l'interface graphique de la côte (ou plage)
- Réaliser l'interface graphique de la fin de jeu
- Implémentation de la configuration du bateau
- Implémentation de la lecture de la carte depuis un fichier

#### 10.2.2. Test relatif

- Ce qui s'est bien passé : les tâches de ce module ont toutes étées implémentées outre la lecture de la carte depuis un fichier. Il n'y a pas eu de problèmes majeurs dans ce sprint
- Ce qui peut être mieux fait : le rendu visuel de l'écran de configuration du bateau et de la carte
- Améliorations : utiliser des images et des motifs pour le bateau, la plage et les bouées, choisir une nouvelle texture pour l'eau, et faire un refactoring de la classe Board qui commence à être plus imposante que initialement prévue

## 11. Analyse des résultats produits



Figure 3. Interface utilisateur : choix de la langue

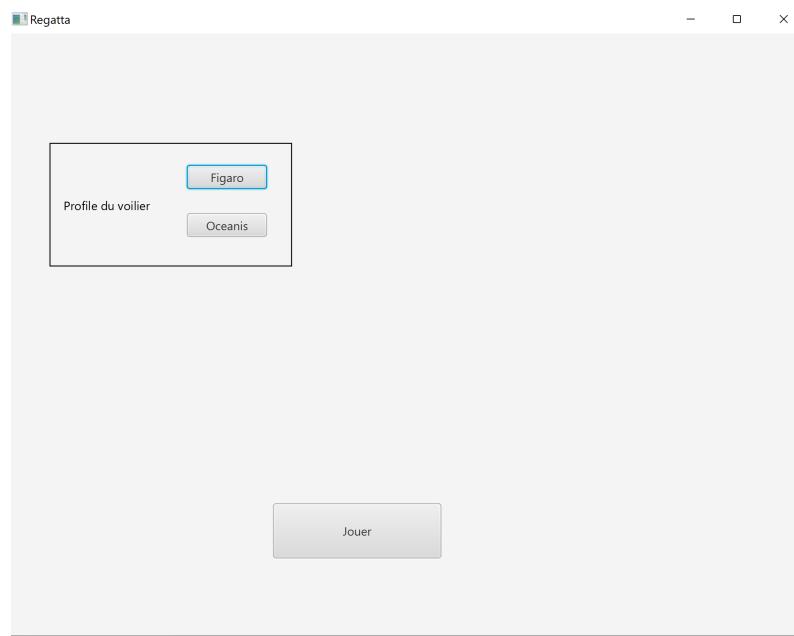


Figure 4. Interface utilisateur : choix du voilier

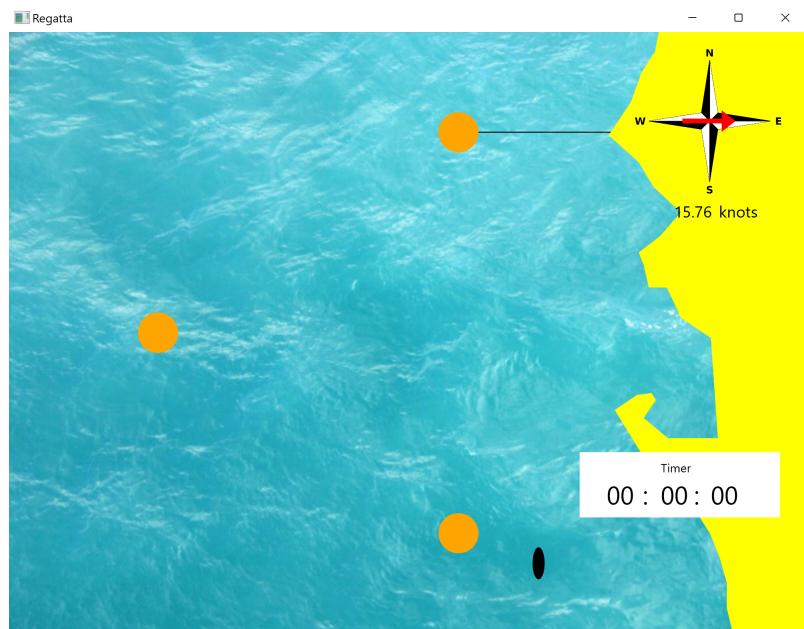
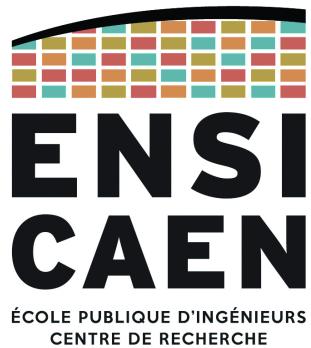


Figure 5. Interface utilisateur : interface du jeu



f  
@  
in

## École Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 4505

14050 CAEN cedex 04

