Ecole Publique d'Ingénieurs en 3 ans

PROJET REGATE

Le 14 novembre 2022

Jordy Gelb
Etienne Langlois
Emeline Flicourt
Romain Cailly
Matéo Ducastel
Thomas Seng
Tangui Steimetz
Benjamin Maignan

Tuteur école : **Régis Clouard**

www.ensicaen.fr

TABLE DES MATIERES

TABLE DES FIGURES

TABLE DES TABLEAUX

RAPPORT DU PROJET

1. Présentation du projet

1.1. Rappel du sujet

En tant que nouvelle entreprise dans le domaine de la création de jeux-vidéo, Monsieur Régis Clouard nous a demandés, en la qualité de client, de créer un jeu vidéo de courses de régates virtuelles.

Ce jeu est destiné à l'apprentissage de la navigation en voilier. Il n'y a donc qu'un seul joueur humain. 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 un temps minimal.

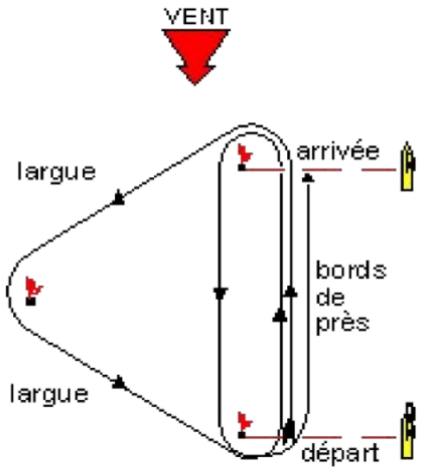

Figure 1 - Exemple de parcours olympique possible (Cf. sujet de projet)

À chaque carte, sont associés une suite de bouées spécifique et un vent donné récupéré à partir de coordonnées GPS stockées dans la carte. Le bateau, quant à lui, possède des caractéristiques précises telles que la taille de l'équipage ou la taille de la voile.

L'objectif de ce projet est d'appliquer les différentes règles de codage et patrons de conception étudiés durant les cours de Génie Logiciel, tout en répartissant judicieusement le travail entre les huit membres de notre équipe.

1.2. Objectifs et contraintes

Contraintes technologiques

Le langage choisi pour développer ce logiciel est le **JAVA** avec la bibliothèque **JavaFX** pour l'interface graphique. Ce langage, basé sur le paradigme orienté objet nous a été imposé par Monsieur Régis Clouard. De plus, celui-ci souhaitait que nous fassions des tests unitaires à l'aide des bibliothèques **JUnit** et **Mockito**.

La seule contrainte qui nous a été imposée concernant l'architecture de notre projet est le modèle **MVP** (modèle, vue, présentateur). Nous avons donc basé toute notre conception sur celui-ci pour satisfaire les volontés de notre client.

Pour ce qui est de la gestion du projet, notre client a souhaité que nous utilisions le gestionnaire de version Git avec la plateforme **Gitlab** de l'ENSICAEN. Enfin, nous avons utilisé le moteur de production **Gradle** qui nous a permis de lancer les tests unitaires facilement.

Contraintes fonctionnelles

D'un point de vue fonctionnel, trois contraintes ont été posées.

Tout d'abord, le logiciel se doit d'être multilingue, disponible obligatoirement en Français et en Anglais et ouvert à de nouvelles langues dans le futur.

Ensuite, le parcours et la carte sur lesquels la simulation s'effectue doivent être initialisés à partir d'un fichier. De plus, celui-ci doit comporter les coordonnées géographiques du lieu de la course afin de récupérer la vitesse et la direction du vent sur Internet.

Enfin, il devait être possible d'établir un score à la fin du parcours, basé sur le temps pris par la régate pour passer les différentes bouées, et de revoir intégralement sa course une fois celle-ci terminée.

Objectifs du logiciel

Dans un premier temps, l'utilisateur doit entrer son pseudonyme et choisir le type de voilier, ainsi que ses caractéristiques (taille de la voile, nombre de membres de l'équipage) dans une fenêtre sous forme de menu.

Le joueur ne contrôle que la direction de la régate (bâbord et tribord). En outre, s'il n'y a pas d'action, la régate garde le même cap, et l'utilisateur est pénalisé lors d'un virement de bord.

Le joueur doit pouvoir voir le cap, la vitesse, ainsi que la vitesse du vent par l'intermédiaire d'un tableau de bord.

En cas de collision avec le sable ou les bouées, la vitesse devient nulle et le bateau est immobilisé durant un laps de temps.

D'autres bateaux (des intelligences artificielles) participent à la régate et interagissent avec le joueur (collision, aspiration, etc.).

1.3. Diagramme des cas d'utilisation

Une fois le logiciel lancé, l'utilisateur doit choisir son pseudo pour accéder au menu de configuration de la course de régate. Sur ce menu, il peut sélectionner le bateau qu'il veut utiliser ainsi que ses caractéristiques. Une fois la carte sélectionnée parmi la liste de cartes disponibles, l'utilisateur peut démarrer sa partie.

Le menu laisse alors la place à la fenêtre de jeu, où le joueur peut contrôler l'orientation de son navire à l'aide des touches directionnelles. Le voilier est soumis à un vent, affiché au-dessus de la carte, qui influence sa vitesse selon son orientation.

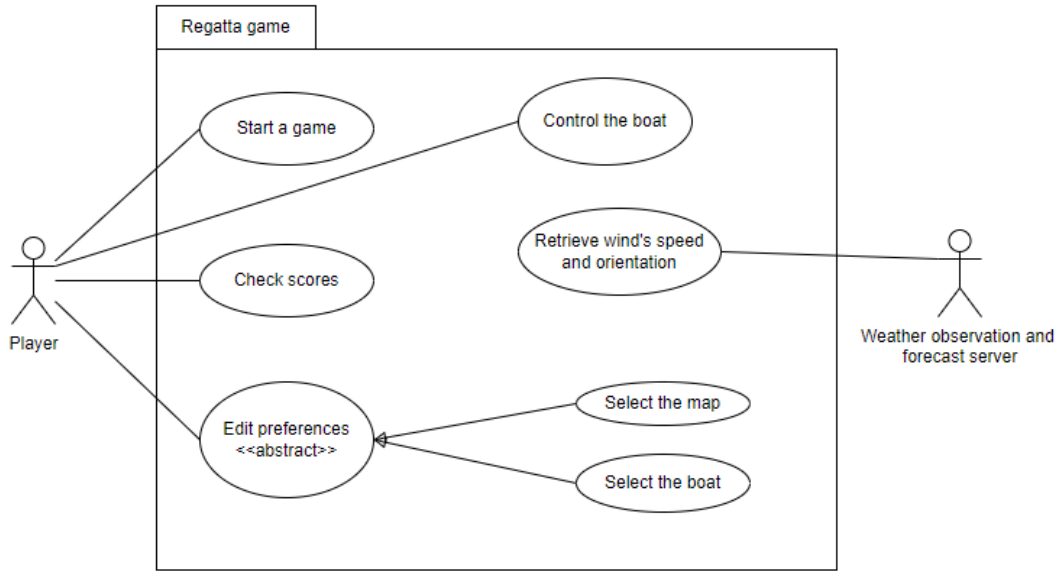

Figure 2 - Diagramme des cas d'utilisation

1.4. Analyse des risques

Avant de commencer à travailler sur le projet, il est nécessaire d'évaluer ce qui pourrait nuire au bon déroulement du projet. Ces risques peuvent être extérieurs tels que des imprévus ou intérieurs comme des problèmes d'organisation ou de connaissances.

Nous avons donc décidé de réaliser une analyse des risques. Pour chaque risque, nous avons considéré son impact, sa criticité (à partir de la gravité et de la probabilité) ainsi que le responsable. De plus, nous avons également réfléchi à des actions préventives pouvant limiter l'apparition de ces problèmes.

Type de risques	Risque	Impact	Gravité	Probabilité	Criticité	Action préventive	Responsable
Risques technologiques	Apprentissage du JAVAFX difficile	Ralentissement de l'avancement du projet, Contrainte technique	Moyenne	Probable	Modéré	Se documenter en amont sur le JAVAFX	Développeurs
Risques techniques	Bug dans l'application	Ralentissement de l'avancement du projet	Grave	Probable	Elevé	Bien comprendre le modèle MVP et respecter les patrons de conception	Développeurs
Risques techniques	Problème de Git	Non mise en commun de la bonne version.	Moyenne	Peu probable	Basse	Vigilance des responsables git lors de la validation d'une mise en commun.	Responsable git
Risques technologiques	Apprentissage de Git	Ralentissement de l'avancement du projet	Faible	Faible	Basse	Voir des tutos	Développeurs
Risque lié au degré d'intégration	Problème lors de la récupération du vent sur le serveur correspondant	Non prise en compte du vent dans le logiciel qui est pourtant une contrainte du client.	Importante	Peu probable	Modéré	Tester sur différentes machines. Prendre en compte les erreurs.	Développeurs
Risques sur les délais	Mauvaise estimation du temps	Ralentissement du projet	Importante	Très probable	Très Élevé	Mise en place d'un planning avec objectif sur le court terme	Chef de projet
Risques intrinsèques à la gestion de projet	Mauvaise dispersion des tâches	Perte de temps, non-respect de la deadline.	Importante	Probable	Elevé	Bonne compréhension en amont du développement du projet et communication entre les membres du projet	Chef de projet
Risques fonctionnels	Mauvaise compréhension du sujet	Produire un résultat ne convenant pas au client	Maximal	Peu probable	Elevé	Lire plusieurs fois le sujet et confronter nos compréhensions pour répondre à certains problèmes.	Ensemble de l'équipe.
Risques humains	Absence d'un ou plusieurs membres	Ralentissement du projet.	Importante	Probable	Elevé	Prévenir dès que possible ses absences pour gérer les différentes tâches.	Ensemble de l'équipe.
Risques humains	Conflits entre plusieurs membres au sein de l'équipe	Ralentissement du projet.	Importante	Peu probable	Modéré	Faciliter la communication entre les différents pôles du projet.	Chef de projet

Tableau 1 - Analyse de risques

2. Élaboration d'une solution

2.1. Architecture

Le patron d'architecture employé pour ce projet est le modèle MVP (Modèle, Vue, Présentateur). Celui-ci est adapté à la création d'un logiciel avec une interface graphique, car il permet de séparer la partie interface utilisateur (Vue) de la partie logique (Modèle) du programme.

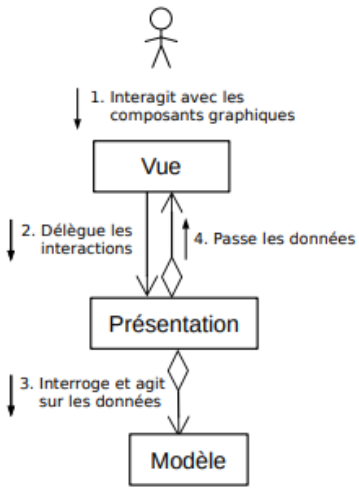

Figure 3 - Schéma du patron d'architecture MVP (issu du cours de génie logiciel)

La partie **Modèle** est centrée sur la logique métier et contient le modèle du domaine ainsi que les données. Dans le cas du projet, elle est composée des classes relatives à la carte (*map*), au joueur (*player*), au bateau (*ship*), ainsi qu'aux collisions et au passage de bouées. En somme, il s'agit là de toutes les fonctions permettant la gestion du bateau, c'est-à-dire la possibilité d'avancer, la vitesse, le vent, le tout sur une carte donnée.

La partie **Présentation** sert à interconnecter la partie **Modèle** et la partie **Vue**. Elle contient la logique de présentation. Les événements sur la **Vue** y sont traités puis les données du **Modèle** sont modifiées et la **Vue** est actualisée en conséquence. Notre partie présentation est composée des classes *GamePresenter, IGameView, ILoginView, LoginPresenter* et *UserAction*.

La partie **Vue**, quant à elle, ne gère que l'aspect graphique et ne contient donc aucune logique. Elle est composée des classes relatives à l'espace de jeu ainsi que les classes relatives au menu.

2.2.Choix d'interface

Une fois toutes les contraintes et objectifs listés, nous avons travaillé sur une maquette de notre interface qui nous a servie de base au développement de notre réelle interface utilisateur.

Maquette

Pour que notre interface utilisateur corresponde aux besoins de l'utilisateur, Les éléments suivant devaient être visible :

- Pour le menu :
    - Choisir son pseudonyme.
    - Choisir son bateau.
    - Choisir la taille de la voile.
    - Choisir le nombre d'équipiers.
    - Afficher les précédents scores en fonction du choix du pseudonyme.
    - Choisir la carte.
- Pour l'écran de jeu :
    - La carte avec les bouées et le voilier.
    - Un indicateur pour la direction et la force du vent.
    - Une liste des balises passées et le temps associé.
    - Le chronomètre représentant le temps depuis le début de la partie.
    - La détection de fin de courses avec affichage du score et possibilité de visionner la course effectuée.

Figure 4 - Prototype du menu d'accueil de l'interface utilisateur

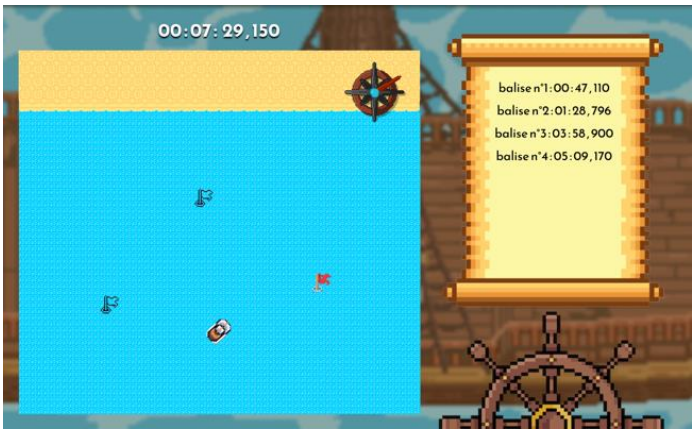

Figure 5 - Prototype de l'espace de jeu de l'interface utilisateur

Nous avons décidé d'utiliser pour l'interface des éléments graphiques unis sous le thème du « **Pixel Art** » pour que celle-ci soit conviviale sans être trop chargée. Les éléments graphiques utilisés sont tous libres de droit.

Implémentation de l'interface

Nous avons donc implémenté l'interface en respectant la maquette. Cependant, nous avons fait face à des difficultés et avons dû renoncer à certains détails pour respecter les délais.

Nous pouvons notamment citer le cas du vent qui est retranscrit sous forme de texte à la place de l'idée originale. En effet, l'implémentation voulue était d'actualiser la rose des vents en temps réel, mais cela était trop difficile à réaliser dans le temps imparti. De plus, le chronomètre a changé de place pour que le vent puisse se positionner au-dessus de la carte.

La détection de fin de course n'a pas été réalisée par manque de temps, impliquant que ni le score de fin, ni le bouton de replay ne sont disponibles dans la version actuelle du logiciel (un autre moyen a été mis en place pour visualiser la course effectuée.). De plus, les résultats n'étant pas implémentés, la restauration des précédents scores ne peut donc pas être présente.

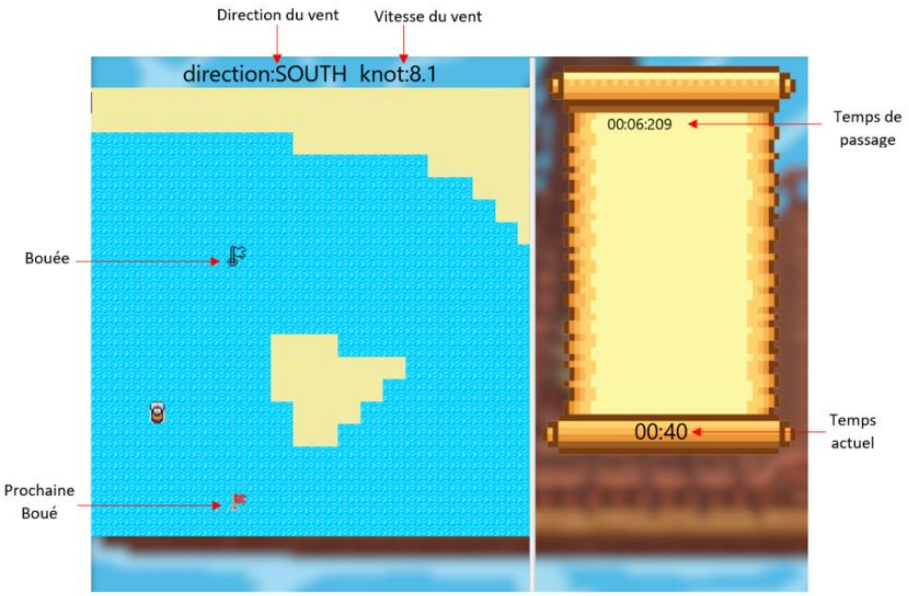

Figure 4 – Visuel de l'interface de jeu

2.3. Moyen et organisation globale

Avant le début du projet, chaque membre de l'équipe s'est vu attribuer un rôle. Nous avons donc :

- Jordy Gelb, en tant que **Chef de projet**.
- Romain Cailly et Etienne Langlois, comme **Architectes**.
- Matéo Ducastel et Thomas Seng remplissant le rôle des **Responsables de version**.
- Emeline Flicourt, Benjamin Maignan et Tangui Steimetz en tant que **Développeurs**.

Ces rôles ne restent qu'à caractère indicatif. En effet, chaque membre du groupe a travaillé en tant que développeur au cours du projet et tout le monde a pu exposer son point de vue par rapport à l'architecture de celui-ci.

Nous avons décidé d'affecter deux développeurs à l'équipe travaillant sur la partie graphique et les six autres sur la partie modèle. Le travail a été divisé entre chaque individu de l'équipe de sorte qu'ils puissent travailler sans dépendre de la progression des autres. Les modifications et mises à jour ont été ajoutées à la branche principale au cours de l'avancée du projet. Nonobstant, il faut noter que les développeurs travaillant sur l'interface graphique dépendaient en partie des développeurs travaillant sur le **Modèle,** car ils devaient adapter l'interface aux changements du **Modèle**.

Pour réaliser cette application, nous avons utilisé la méthode **Agile**. Le développement s'est donc organisé autour de deux MVP. Le premier visait à livrer une application fonctionnelle permettant au bateau de se déplacer sur une carte donnée avec un vent fixe. Pour le second **MVP**, nous devions implémenter les caractéristiques du bateau (composition de l'équipage et taille de la voile), les collisions, le passage des bouées et la possibilité de rejouer son chemin dans son entièreté. La course devait également se terminer, mais par manque de temps, cela n'est pas possible.

Nous avons aussi réalisé des tests visant à nous assurer que notre **Modèle** fonctionnait correctement et prévenir toute régression du code à la suite de changements.

2.4.Implémentation

2.4.1. Patrons de conception utilisés

Afin de créer un logiciel « propre », l'équipe devait implémenter les patrons de conception que nous jugions nécessaire pour rendre le logiciel le plus robuste possible et qu'il respecte les principes SOLID. Cette partie met en exergue les différents patrons mis en place et leurs utilités.

Procuration

L'interaction effectuée avec le serveur permettant de récupérer la direction et la vitesse du vent est implémentée à l'aide du patron de conception **Procuration**. Celui-ci permet d'interagir une unique fois avec le serveur et stocke les données reçues pendant 24 heures. Ainsi, il n'est pas obligatoire de faire une requête à chaque utilisation.

Décorateur

La sélection des caractéristiques de la régate est effectuée avec le patron de conception **Décorateur**. Ce patron permet de définir la taille de la voile et la composition de l'équipage. Les décorateurs modifient l'interaction de la régate avec son environnement extérieur et son fonctionnement interne.

Singleton

Le chronomètre utilise le patron de conception **Singleton** afin qu'il ne soit instancié qu'une seule fois et qu'il puisse être accessible depuis n'importe quelle classe. Celui-ci est démarré au début de la course et permet notamment d'indiquer les temps réalisés par les bateaux lors du passage des bouées.

Monteur

Le patron de conception **Monteur** a pour responsabilité de créer correctement le voilier avec ses paramètres correctement instanciés suivant le choix de l'utilisateur lors de la sélection dans le menu. Il est donc utilisé en complément du patron de conception **Décorateur.**

Commande

Le patron de conception **Commande** est utilisé pour implémenter le replay de la course. Celui-ci permet de stocker les commandes successivement entrées par l'utilisateur. Lorsque la partie est terminée, l'utilisateur peut appuyer sur la touche R et le bateau rejoue sa course à l'identique en suivant les commandes stockées précédemment.

2.4.2. Diagramme de paquets

Afin de mieux représenter l'architecture de notre projet, nous avons créé un diagramme de paquet. Celui-ci nous a notamment permis de bien nous organiser dans la répartition du travail en délimitant les tâches de chacun pour éviter les conflits.

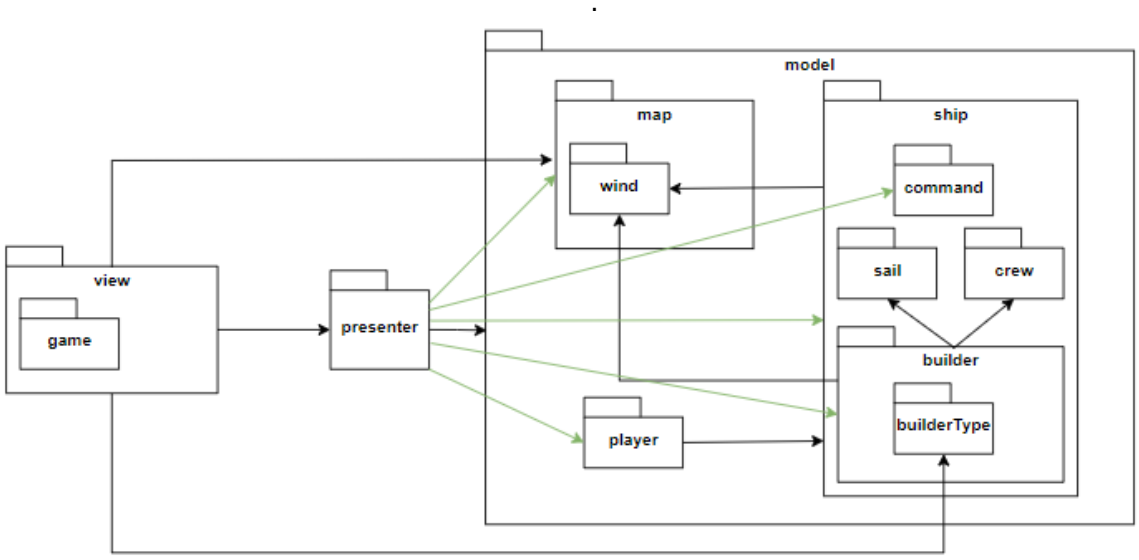

Figure 6 - Diagramme de paquets

3. Retour sur expérience

3.1. Difficultés rencontrées lors du projet

3.1.1.1. Post-analyse des risques

Une fois le projet terminé, nous sommes revenus sur la liste des risques que nous avions définie au préalable. Deux catégories ont retenu notre attention : les risques technologiques et les risques sur les délais.

Les risques technologiques

Au nombre de deux, les risques technologiques représentaient une menace importante au bon déroulement du projet.

Le premier concernait l'apprentissage du **JavaFX**. Il a été évité en confiant cette partie à une équipe de deux développeurs, dont un ayant déjà travaillé auparavant sur des projets utilisant cette technologie. Cependant, il y a eu des problèmes de couplage fort entre le modèle et la vue lors du **MVP 1** qui ont été corrigés lors du **MVP 2** afin de répondre aux critiques exposées par notre client.

Pour ce qui est du second risque technologique, celui-ci était lié l'utilisation de l'outil Git pour la gestion de version qui a eu un impact important sur l'avancée du projet. En effet, certains membres de l'équipe, non-familiers avec l'outil, ont rencontré des difficultés lors de la prise en main. De plus, nous avons dû faire face à de nombreux conflits quand il a fallu lier les fonctionnalités développées en parallèle, ce qui nous a fortement ralenti dans notre avancement.

Les risques sur les délais

Les risques en rapport avec les délais se sont confirmés, ce qui nous a empêché de mener à terme le projet et de rendre une version du logiciel respectant toutes les exigences du client. En effet, il reste à implémenter les bateaux contrôlés par des IA qui ont pour rôle de se confronter à l'utilisateur. De plus, de nombreux tests unitaires n'ont pas été réalisés ou mis à jour à la suite de changements dans le code. Cette situation est arrivée, car nous avons décidé de nous concentrer sur l'implémentation des fonctionnalités de bases et la correction du programme par manque de temps, ce qui a grandement modifié une partie du code et donc rendu inutiles certains tests.

3.1.2. Gestion de projet

Ce projet a été pour nous une occasion d'expérimenter le travail dans une équipe de huit personnes en employant la méthode agile. Nous nous sommes donc fixés des objectifs à court terme au travers de l'élaboration de deux **MVP** (livrables) représentant chacun 48 heures-homme.

Nous commencions chaque séance de deux heures par une réunion d'équipe visant à donner l'état d'avancement de chaque fonctionnalité, planifier le travail à réaliser dans l'immédiat et discuter de solutions aux possibles problèmes rencontrés. Cet exercice, difficile car limité dans le temps, demande de résumer brièvement le travail de tous, tout en laissant le champ libre au dialogue. Nous avons donc pu observer une nette amélioration entre les premières réunions prenant plus de temps que prévu et les dernières ou nous arrivions à aborder tous les points efficacement.

Parmi les erreurs observées, il nous a été reproché d'avancer sur des tâches du **MVP 2** alors que nous étions encore en train de travailler sur le **MVP 1**. Nous avons corrigé ce problème au cours du **MVP 2** en nous concentrant uniquement sur les fonctionnalités en rapport avec celui-ci.

Nous pouvons noter comme point positif la répartition du travail qui a été faite au sein de notre équipe. En effet, nous avons réussi à donner à chaque membre de l'équipe une charge de travail variant en fonction de ses compétences et de l'état d'avancement des fonctionnalités qu'il implémentait.

Toutefois, nous avons rencontré des difficultés lorsqu'il a fallu regrouper le travail de chacun, menant à des conflits entre les différents espaces de travail. En effet, des modifications ont été apportées aux mêmes fichiers par plusieurs personnes différentes, entraînant des conflits importants et une perte de temps majeure. Ceci souligne l'importance de la communication au sein d'un projet afin d'éviter des problèmes de ce type.

3.2. Retour sur le travail fourni

3.2.1. Amélioration (d'un point de vue gestion de projet)

À la fin du **MVP 2**, nous avons pu prendre du recul sur notre organisation et définir des axes d'amélioration d'un point de vue gestion de projet.

Le premier point majeur sur lequel l'équipe de projet pourrait s'améliorer est un respect plus strict des espaces de travail de chacun afin de ne pas créer de conflits trop importants lors de la mise en commun du travail de chacun.

Le second point d'amélioration serait la communication entre les différents pôles du projet. Assurément, malgré les réunions de début de projet nous permettant de rester à jour sur les modifications, il était difficile pour le groupe de concevoir des modules pouvant facilement s'incorporer à l'ensemble du projet. Ce problème peut être résolu si les différents pôles impliqués communiquent directement entre eux sur leurs besoins.

Le troisième et dernier axe d'amélioration serait une meilleure analyse des besoins du client afin d'éviter des retours nécessitant des changements majeurs dans le code. En effet, au cours du projet, les remarques du client ont nécessité d'importantes corrections dans le logiciel, ce

qui a entraîné une charge de travail supplémentaire qui nous a forcée à revoir la planification des tâches mise en place, et donc la liste des fonctionnalités à implémenter pour le **MVP 2**.

3.2.2. Possible MVP 3

Dans le cadre d'un possible **MVP 3**, plusieurs changements sont à prévoir afin de corriger les problèmes rencontrés au cours des deux **MVP** précédents, et d'ajouter des fonctionnalités.

Parmi les changements à apporter, il faudrait faire apparaître graphiquement le vent à l'utilisateur au travers d'une flèche à la place d'un simple texte. De plus, il serait nécessaire d'améliorer le passage des bouées pour s'assurer que le bateau passe bien derrière celle-ci, et non simplement s'en rapprocher à une certaine distance.

Concernant les ajouts à faire, il serait nécessaire de mettre en place un écran de fin de partie permettant au joueur de rejouer sa course. Il faudrait aussi permettre à l'utilisateur de choisir dans le menu la carte sur laquelle il souhaite jouer. En outre, nous souhaiterions ajouter un écran des scores pour que l'utilisateur puisse comparer ses différentes courses de bateau. Pour finir, une fonctionnalité qui était voulue par le client, mais que nous n'avons pas eu le temps d'implémenter, est la mise en place de bateaux dirigés par des IA pour se confronter à l'utilisateur.

Pour conclure, nous considérons que les tests unitaires sont insuffisants sur notre projet. Nous souhaitons donc en ajouter et mettre à jour ceux existants pour qu'ils testent au mieux notre code. Ce projet nous montre non seulement leur importance, mais également la difficulté de les mettre en place. Pour cela, il faut que notre code soit testable, ce qui demande de programmer le plus simplement possible chaque méthode pour qu'elle n'effectue qu'une seule action. De plus, pour éviter les erreurs, il faut développer ces tests en parallèle de l'écriture du code.

4. Conclusion

Ce projet de Génie Logiciel, réalisé en équipe de huit étudiants, nous a offert l'opportunité d'appliquer les connaissances acquises sur la gestion de projet et les patrons de conception.

L'objectif principal de ce projet était de réaliser un logiciel de régate simulant une course de bateaux. Le programme se devait de respecter les principes **SOLID** et d'implémenter de nombreux tests pour éviter toute malfaçon et régression. Nous avons donc implémenté de nombreux patrons, dont le patron d'architecture **MVP**, qui a été la base de tout le projet en divisant la Vue, la Présentation et le Modèle.

La méthode de travail **Agile** imposait ainsi de livrer une première version fonctionnelle rapidement (MVP 1) puis d'ajouter des fonctionnalités initialement demandées par le client (MVP 2). Nous avons réussi à respecter les objectifs du premier **MVP,** mais n'avons pas respecté ceux du second. Cela est principalement dû à une mauvaise estimation du temps nécessaire pour le développement de chaque fonctionnalité.

Nous avons su faire face à la majorité des risques, bien que certaines difficultés liées à l'organisation globale et le regroupement des travaux de chacun ont considérablement ralenti la dynamique de travail. Il a été déterminé qu'un effort au niveau de la communication entre les différents pôles de travail aurait pu diminuer l'impact de ce risque sur notre projet.

Pour conclure, ce projet nous a permis de mettre en pratique la méthode **Agile**, ainsi que les enseignements du cours de Génie Logiciel, tout en nous confrontant aux problèmes organisationnels qu'implique le travail en équipe de huit personnes. Avec l'expérience acquise durant ce projet, nous serons plus en mesure de réaliser un projet en prenant mieux en compte les différents risques et contraintes.

ANNEXE

Le diagramme de classe du projet est trouvable en format PDF sur le Gitlab du projet, dans le même dossier que ce rapport. Nous avons fait ce choix dû à une forte complexité de celui-ci qui empêchait une bonne lisibilité au format PDF.

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04

ministère éducation nationale enseignement supérieur recherche

Normandie Université

Cti Commission des titres d'ingénieur

cdefi Conférence des Directeurs des Écoles Françaises d'Ingénieurs

European Accreditation of Engineering Programmes EUR-ACE®

CONFÉRENCE DES GRANDES ÉCOLES