ENSSAT
LANNION

UNIVERSITÉ DE RENNES 1

Projet d'implémentation Java : Big Pirate

BACLET Anne-Charline, BIAU-GUILHEMBET Marie, THAO KY Christophe

Informatique 2 - Année scolaire 2016-2017

Table des matières

Introduction

Le système à numériser est un jeu de stratégie intitulé Big Pirate. Ce projet est commun aux deux modules Modélisation / Méthode de conception et Java / Interfaces graphiques. Durant la phase de conception du jeu informatisé Big Pirate, nous avons mis en oeuvre la méthode itérative de modélisation Unified Modeling Language, nécessaire au bon développement d'un logiciel orienté objet. Cela nous a permis de dégager et d'étudier les trois grandes itérations du projet. Celles-ci consistent à jouer le tour du moussaillon, du pirate et du fantôme. En regroupant les itérations, nous avons élaboré un diagramme de classe complet ainsi que dégagé un design pattern de type Modèle / Vue / Contrôleur. Notre jeu a été modélisé de façon évolutive et ergonomique afin de pouvoir implémenter des aménagements de la version de base.
Le risque majeur identifié est celui de la gestion du temps, c'est pourquoi nous allons planifier les étapes de l'implémentation. En s'appuyant sur la conception réalisée, nous allons maintenant effectuer l'implémentation en Java.

I. Conception

1. Système et itérations

Le système que nous allons étudier tout au long de ce projet est le jeu intitulé Big Pirate. Les acteurs qui interagissent avec notre système sont les joueurs. Le système rend un type de service aux joueurs qui est de jouer un tour. Le jeu est constitué de trois itérations : jouer le tour d'un moussaillon, jouer le tour du pirate et jouer le tour du fantôme.

Les moussaillons

Il y a de 1 à 3 moussaillons par partie. Les moussaillons commencent la partie sur une case de la barque. Un moussaillon doit pouvoir lancer un petit dé pour se déplacer, ramasser un trésor, abandonner son trésor, se cacher et rester caché dans une cachette grâce à des cartes cocotier, et doubler son score au dé grâce à une carte perroquet. Il doit également gagner la partie en revenant à la barque avec un trésor.

Précondition : le tour du personnage précédent (fantôme ou autre moussaillon) est terminé.
Postcondition: le moussaillon s'est déplacé ou est resté caché. S'il est retourné à la barque avec un trésor, il remporte la partie.

Le moussaillon a des comportements particuliers , qui sont les suivants :

- Au début du tour, il peut rester caché s'il l'est déjà et s'il lui reste une carte cocotier
- Utiliser une carte perroquet pour doubler son score au dé
- Abandonner son trésor pour retarder le pirate
- Se cacher s'il passe sur une case cocotier et qu'il possède une carte cocotier

Le pirate

Le pirate commence la partie sur la case grotte. Le pirate doit pouvoir lancer un dé allant de 1 à 6 et il doit avoir la possibilité de choisir sa direction lorsqu'il atteint une intersection.

Précondition : le tour du moussaillon est terminé.
Postcondition : le pirate s'est déplacé.
Il a des comportements particuliers :

- Le pirate ne peut se déplacer qu'en avant
- Lorsqu'il rencontre un moussaillon, il le tue et si ce dernier est en possession d'un trésor, celui-ci est renvoyé à la grotte. Le pirate doit également s'arrêter sur la case du moussaillon.
- Lorsqu'il rencontre un trésor, il s'arrête à l'emplacement de celui-ci et renvoie le trésor à la grotte

Une fois que le pirate s'est déplacé et que l'on a vérifié l'ensemble des cases sur lesquelles celui-ci s'est déplacé, le tour se finit.

Le fantôme

Le fantôme démarre la partie sur la case F. Aucun joueur ne déplace le fantôme. Le fantôme doit pouvoir lancer un dé allant de 1 à 3 et il doit avoir la possibilité de traverser toutes les cases du plateau.

Précondition : le tour du pirate est terminé.
Postcondition : le fantôme s'est déplacé.

Le fantôme ne se déplace qu'en avant. Durant un tour, le fantôme peut accéder à certaines cases autour de lui. Si un moussaillon possédant un trésor se trouve sur l'une de ces cases, alors le fantôme se déplace obligatoirement dans sa direction, s'arrête sur cette case et s'empare du coffre, qui est déposé aléatoirement sur une case sentier du plateau (cette case doit être vide, sans personnage, ni trésor). Sinon, s'il ne peut atteindre aucun case avec un moussaillon possédant un trésor, alors il se déplace aléatoirement.

2. Diagramme de classes

L'étude des itérations précédentes nous a amené à établir le diagramme de classe détaillé suivant :

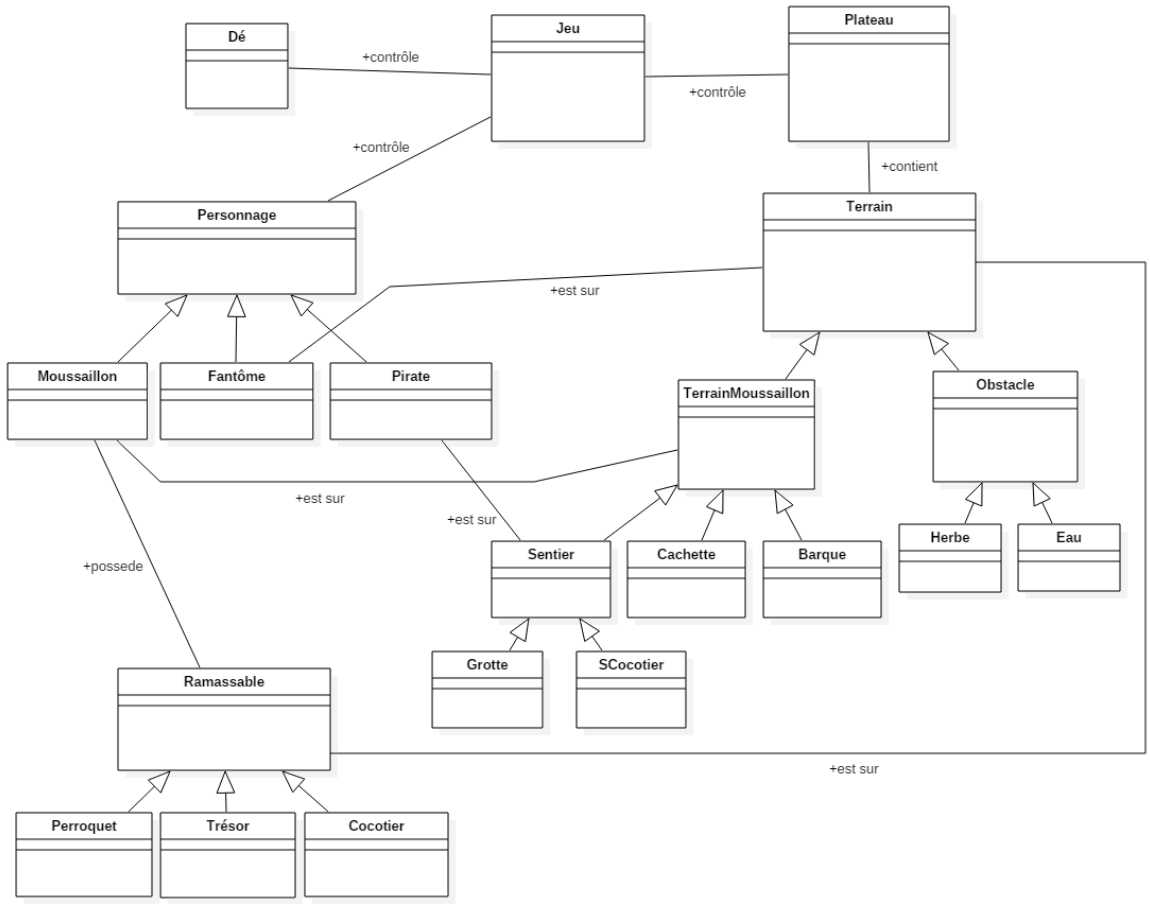


Diagramme de classe

Dans ce diagramme, nous considérons 5 classes principales et dont les autres classes héritent. La classe Plateau regroupe toutes les cases du jeu. La classe Terrain modélise les différents types de cases sur le plateau. La classe Personnage contient les personnages du jeu qui sont situés sur une case du plateau. La classe Dé modélise les deux dés différents en fonction des personnages, l'un allant de 1 à 3 et l'autre de 1 à 6. La classe Ramassable modélise tous les objets pouvant être ramassés sur le terrain (les types de cartes possédées par un moussaillon ainsi que les trésors).

3. Modèle

Le design pattern que nous avons identifié est du type Modèle / Vue / Contrôleur. Le modèle correspond au diagramme de classe, la vue est l'interface graphique et le contrôleur correspond à la classe Jeu.

Le déroulement de la partie s'effectue de la manière suivante. On lance une partie en choisissant le nombre de moussaillons. La boucle de jeu (itérations 1 à 3) s'effectue tant qu'aucun moussaillon n'a atteint la barque et qu'il reste des moussaillons en jeu. Dès qu'une de ces conditions n'est plus remplie, le jeu se termine et on affiche le vainqueur.

II. Organisation

1. Priorisation

L'implémentation doit se faire suivant la priorisation suivante :
- classe Dé
- classes Personnage et Terrain
- classe Ramassable
- classe Plateau
- classe Jeu
- interface graphique

En effet, nous devons d'abord créer le modèle, puis le contrôleur, et enfin la vue.

2. Planification

	45	46	47	48	49	50
Tâche 1	Christophe					
Tâche 2	Christophe					
Tâche 3	Marie					
Tâche 4	Anne-Charline					
Tâche 5		Toute l'équipe				
Tâche 6		Toute l'équipe				
Tâche 7				Toute l'équipe		
Tâche 8						Toute l'équipe

Tâche 1 : implémentation de la classe Dé
Tâche 2 : implémentation de la classe Terrain
Tâche 3 : implémentation de la classe Personnage
Tâche 4 : implémentation de la classe Ramassable
Tâche 5 : implémentation de la classe Plateau
Tâche 6 : implémentation de la classe Jeu
Tâche 7 : création de l'interface graphique
Tâche 8 : rédaction du rapport

III. Implémentation et validation

1. Modèle

La classe De

La classe De modélise tous les dés que le jeu peut proposer aux joueurs. Chaque dé possède en attribut le score maximal que l'on peut obtenir en le lançant grâce à la méthode seLancer().

La classe Terrain et ses sous-classes

La classe Terrain modélise toute les cases du plateau et toutes les cases sur lequel le fantôme peut se déplacer.
Un Terrain possède en attributs :

- un booléen nommé fantome modélisant la présence du fantôme, true signifiant que le fantôme se trouve sur cette case.
- un entier x modélisant la coordonnée x de la case.
- un entier y modélisant la coordonnée y de la case.

Il existe 2 classes héritant de la classe abstraite Terrain : Obstacle et TerrainMoussaillon.

La classe Obstacle modélise toutes les cases sur lesquelles les personnages humains ne peuvent se déplacer. Seul le fantôme peut y accéder. Elle hérite de Terrain car c'est une case du plateau.
Il existe 2 classes héritant de la classe abstraite Obstacle : Herbe et Eau.

La classe Herbe modélise les cases herbes du plateau. Elle hérite de Obstacle car le pirate et les moussaillons ne peuvent s'y trouver.

La classe Eau modélise les cases herbes du plateau. Elle hérite de Obstacle car le pirate et les moussaillons ne peuvent s'y trouver.

La classe TerrainMoussaillon modélise toute case du plateau accessible par des moussaillons. Elle hérite de Terrain car c'est une case du plateau.
La Classe TerrainMoussaillon possède en attributs :

- un tableau de booléens nommé moussaillons modélisant la présence de chaque moussaillon. La case 1 correspond au moussaillon 1 etc, true à la case 1 signifiant que le moussaillon 1 se trouve sur la case etc.

Il existe 3 classes héritant de la classe abstraite TerrainMoussaillon : Cachette, Barque et Sentier.

La classe Cachette modélise toute case sur laquelle un moussaillon se cache à l'aide d'un atout cocotier. Elle hérite de TerrainMoussaillon car un moussaillon peut s'y trouver.

La classe Barque modélise toute case permettant à un moussaillon de gagner s'il possède un trésor. Elle hérite de TerrainMoussaillon car un moussaillon peut s'y trouver.

La classe Sentier modélise toute case du plateau accessible par le pirate. Elle hérite de TerrainMoussaillon car un moussaillon peut également s'y trouver.
La classe Sentier possède en attributs :

- un booléen nommé pirate modélisant la présence du pirate, true signifiant que le pirate est sur la case.
- une collection de Tresor nommée tresors modélisant la présence d'un trésor sur le Sentier. En effet, un Moussaillon peut uniquement abandonner son trésor sur un Sentier.

Ainsi, tout Sentier doit pouvoir :

- ajouter ou retirer un trésor de sa collection. Il le fait grâce aux méthodes : deposer et ramasserTresor qui prennent en argument et retournent, respectivement, une collection de Ramassable contenant ce que l'on veut déposer et le trésor à ramasser.

Il existe 2 classes héritant de la classe Sentier : Grotte et SCocotier.

La classe SCocotier modélise toute case Sentier permettant à un Moussaillon d'utiliser un atout cocotier. Une case Cachette se trouve nécessairement à côté d'une case SCocotier et vice versa. Elle hérite de Sentier car un Pirate peut s'y déplacer.

La classe Grotte modélise la case où se trouvent les trésors au début de la partie et la case où les trésors retournent lorsque le pirate tue un Moussaillon ou trouve un trésor. Elle hérite de Sentier car un Pirate peut s'y déplacer.

La classe Personnage et ses sous-classes

La classe abstraite Personnage modélise tous les personnages du jeu.
Il existe 3 classes héritant de la classe Personnage : Pirate, Fantome et Moussaillon.

La classe Pirate modélise le joueur pirate. Elle hérite de Personnage car il s'agit d'un personnage du jeu.

La classe Fantome modélise le personnage fantôme. Elle hérite de Personnage car il s'agit d'un personnage du jeu. La fantôme n'est contrôlé par aucun joueur, il doit donc décider par lui-même du chemin qu'il va emprunter une fois qu'il a lancé le dé. Il possède donc une méthode choisirParcours prenant en paramètre une collection de tableaux de Terrain contenant tous les déplacements qui lui sont possibles et retournant un de ces déplacements choisi aléatoirement.

La classe Moussaillon modélise les personnages moussaillons. Elle hérite de Personnage car il s'agit d'un personnage du jeu.
La classe Moussaillon possède en attribut :

- un entier static nommé nombreM correspondant au nombre de moussaillons dans la partie.
- un booléen nommé enJeu permettant de savoir si le moussaillon est en jeu ou non.
- trois collections de Ramassable nommées tresor, cocotiers et perroquets contenant respectivement les trésors, les atouts cocotier et les atouts perroquet que possède le moussaillon.

Ainsi, tout moussaillon doit pouvoir :

- ajouter et retirer un trésor de sa collection de trésors, retirer un atout cocotier ou un atout perroquet, de ses collections cocotiers et de perroquets, respectivement. Il le fait grâce aux méthodes : recupererTresor, abandonnerTresor, utiliseCocotier et utilisePerroquet.
- savoir s'il possède un trésor, un atout cocotier ou un atout perroquet.

La classe Ramassable et ses sous-classes

La classe Ramassable modélise tout objet qu'un joueur peut posséder et peut utiliser. Elle modélise également les objets ramassables sur les sentiers.
Il existe trois classes héritant de la classe abstraite Ramassable : Cocotier, Perroquet et Tresor.

La classe Cocotier représente les atouts cocotier, la classe Perroquet représente les atouts perroquets et la classe Tresor représente les trésors. Ces classes ne nécessitent aucune méthode particulière. Elles seront utilisées lorsque des personnages ou des sentiers possèdent des instances de ces classes.

La classe Plateau

La classe Plateau modélise le plateau du jeu. Elle implémente l'interface Observable car elle doit modifier la vue lorsqu'elle change.
La classe Plateau possède en attribut :

- un tableau de tableaux de Terrain nommé plateau correspondant au plateau de 12x12 cases.
- la case du pirate nommée caseP.
- la case du fantôme nommée caseF.
- les cases des moussaillons 1 à 3 nommées respectivement caseMoussaillon1, caseMoussaillon2 et caseMoussaillon3.
- la case où se trouve la grotte nommée caseGrotte.
- un tableau de case Barque nommé casesBarque contenant toutes les cases Barque du plateau.
- une collection d'Observer nommée observers contenant les observeurs.

La classe Plateau peut être considérée comme un sous-contrôleur du jeu. En effet, puisqu'il s'agit de la classe connaissant le plateau du jeu, elle doit pouvoir :

- calculer les différents déplacements de chaque personnage lorsqu'ils lancent un dé. Elle le fait grâce aux méthodes : cheminsFantome, cheminsPirate et cheminsMoussaillon.
- poser aléatoirement un trésor sur le plateau lorsque la case du fantôme et la case d'un moussaillon coïncident. Elle le fait grâce à la méthode : aleaTresor.
- remettre un trésor dans la grotte lorsque la case du pirate et la case d'un moussaillon possédant un trésor coïncident. Elle le fait grâce à la méthode : retourGrotte.
- savoir si un moussaillon est caché derrière un cocotier.
- mettre à jour la vue à chaque modification du plateau.

Les calculs des différents chemins de chaque personnage se font globalement de la même manière. Seules les conditions liées au comportement des différents personnages les différencient ce qui explique la présence de trois méthodes de calcul différentes.

Ces méthodes retournent une collection de tableaux de Terrain. Ces tableaux contiennent une suite de cases contigües correspondant à un déplacement possible du personnage qui est fonction de la case actuelle du personnage et du score du dé. Cette

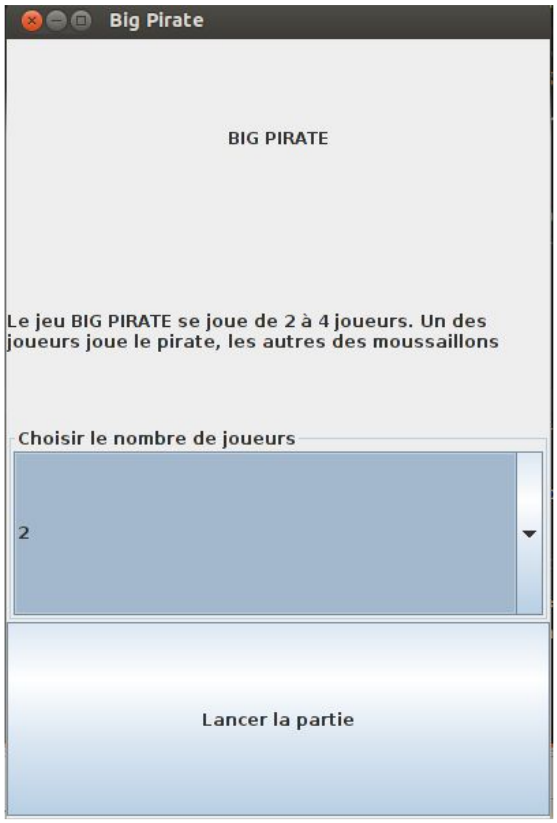
collection est envoyée à la vue, chargée d'afficher chacun de ces déplacements pour que le joueur puisse en choisir un.

2. Vue

La vue est ce qui apparaît à l'écran pour l'utilisateur. Ainsi, il est naturel de considérer que l'interface graphique du jeu est la vue du modèle.

La classe LauncherGUI

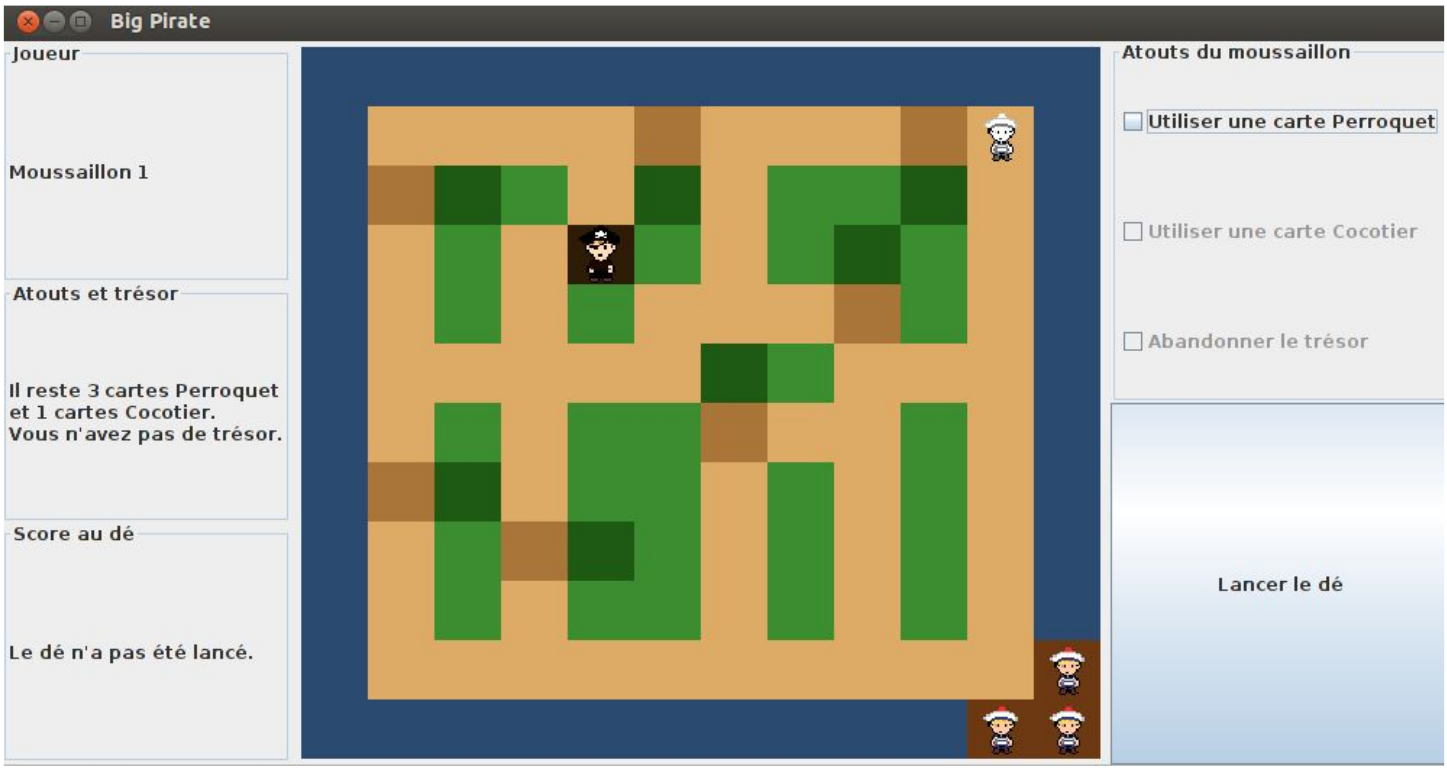
La classe LauncherGUI est une fenêtre permettant de choisir le nombre de joueurs dans la partie et de lancer la partie.



Launcher du jeu Big Pirate

La classe GameGUI

La classe GameGUI, ou interface graphique du jeu ou IHM, est la fenêtre principale du jeu où est affichée le plateau. Elle dispose de plusieurs espaces contenant des informations liées au jeu et des boutons permettant d'interagir avec celui-ci.



Fenêtre du jeu

L'IHM implémente l'interface Observer et observe le modèle : le Plateau. Elle se met à jour à chaque appel de la méthode notifyAll(). Le Plateau appelle cette méthode lors de chaque modification. Il est modifié lorsque l'utilisateur interagit avec lui via les boutons de l'IHM. Ses boutons traduisent les clics de l'utilisateur en appel de méthodes du contrôleur qui va modifier le modèle.

3. Contrôleur

Le contrôleur est l'entité qui récupère les données de l'interface graphique manipulée par l'utilisateur. Il n'agit que lorsque l'utilisateur agit sur l'IHM. Ces données lui permettent de modifier le modèle.

La classe Jeu

La classe Jeu est le contrôleur du jeu. Elle doit connaître le modèle afin de pouvoir le modifier. Elle doit également pouvoir réagir à tous les appels de méthodes de l'IHM.

Conclusion

Le projet Big Pirate nous a permis de mettre en oeuvre les concepts appris lors des modules UML et Java. Nous avons utilisé l'UML pour la conception du projet. Nous avons ainsi pu réfléchir à tous les besoins du projet avant de progressivement arriver à l'implémentation de celui-ci. Cependant, l'implémentation en Java a révélé des failles dans la conception ce qui nous a amenés à modifier la modélisation.

Au niveau de l'organisation, les tâches étaient équitablement réparties. Cependant, l'utilisation de GitHub a rendu le travail beaucoup plus facile. Bien que la prise en main de cet outil a été complexe, une fois maîtrisé il a rendu le partage du code très simple. Cela nous a permis, notamment, de pouvoir toujours travailler avec la dernière version du projet tout en sachant qu'elle était partagée par toute l'équipe.

En revanche, nous utilisions cet outil de manière débutante ce qui nous empêchait de travailler sur le même fichier au même moment. Ainsi, alors que le début du projet était constitué de travaux individuels, la fin du projet était, elle, constituée d'un travail de groupe. Toute l'équipe réfléchissait à l'implémentation mais une unique personne codait réellement. L'utilisation de logiciels plus performants que GitHub aurait pu rendre le travail encore plus efficace.

Au niveau de l'avancement du projet, le code fourni correspond à la version minimale. Une réalisation supplémentaire que nous aurions souhaité implémenter est la possibilité de pouvoir déposer et ramasser des atouts perroquet et cocotier. Des méthodes permettant de le faire sont implémentées dans la classe et les sous-classes de Terrain. Cependant, aucun atout n'est généré au début de la partie ce qui ne permet pas aux moussaillons d'en ramasser. D'autres réalisations sont également possibles, notamment l'utilisation d'atouts par le pirate, l'ajout d'autres types d'atouts ou encore l'amélioration de l'intelligence artificielle du fantôme.

Grâce à ce projet, nous sommes à présent capables d'effectuer la conception et l'implémentation d'un projet informatique.