



# Programmation Avancée

---

## Projet 2023

Poulpo Bowl

Bonnet Kloé • Donat-Bouillud Lallie • Grouiller Martin

Groupe de TD1



07/05/2023

## Sommaire

<b>Introduction</b>	<b>2</b>
<b>Le choix de l'univers du jeu et ses spécifications</b>	<b>3</b>
<b>Les possibilités des joueurs et le déroulement d'une partie</b>	<b>4</b>
Les actions possibles des joueurs	4
Les différents comportements des joueurs	5
Comportement des coureurs	5
Comportement des défenseurs	5
Comportement des brutes	6
Comportement des gardiens	7
Comportement de la balle	7
Déroulement d'une partie	10
Les différents modes de jeu	11
Interface et plateau de jeu	12
Plateau	12
Interface du jeu	13
Placement sur le plateau en début de partie	13
Visualisation des informations sur le plateau	14
Visualisation des informations durant la partie	15
Informations complémentaire textuel action du coach	16
<b>Modélisation UML :</b>	<b>17</b>
<b>Tests et gestion des erreurs</b>	<b>18</b>
Gestion des erreurs internes	18
Gestion des erreurs externes	19
<b>Gestion de projet : l'organisation de l'équipe</b>	<b>21</b>
Planning de réalisation des tâches	21
Communication au sein du groupe	25
Réflexion sur l'ajout de fonctionnalité bonus	25
<b>Bilan critique du projet</b>	<b>26</b>
Ressenti individuel	26
Ressenti de Kloé	26
Ressenti de Lallie	27
Ressenti de Martin	28

## Introduction

Dans le cadre du module de la programmation avancée, nous devions réaliser un petit jeu, grandement inspiré du jeu Blood Bowl, en programmation orientée objet sur C# via l'éditeur de code Visual Studio Code. Le sujet nous offrant une part de liberté, nous avons choisi de l'adapter à notre manière tout en gardant l'esprit du jeu original.

Ce nouveau jeu Poulpo Bowl proposé par notre groupe va affronter 2 équipes : à gauche les vilains trolls affamés ayant pour objectif de manger le poulpe de l'ENSC, et à droite les étudiants de l'école voulant défendre leur mascotte adoré. De la même façon que le jeu original, on retrouve une part d'aléatoire, des coachs pour chacune des deux équipes et même un inventaire.

Ce projet a notamment pour objectif de nous exercer à la programmation objet autour d'un sujet ludique. En plus de nous faire réfléchir à la manière de structurer nos idées à travers les classes, celui-ci nous oblige aussi à penser à l'aspect visuel du jeu, notamment l'interface, afin que le jeu soit facile à utiliser. Cela passe notamment par l'utilisation des fonds et écritures de couleurs pour mieux distinguer les éléments importants.

Ce rapport détaillera le principe de notre jeu, le déroulement d'une partie mais aussi la structuration des classes et une explication plus approfondie de certains choix dans le codage. Enfin, une partie sur la gestion de projet et nos différents ressentis permettra de conclure sur la réalisation de ce projet et de l'organisation au sein du groupe.

## Lancer le jeu

Pour lancer le jeu, il faut cloner le lien de ce projet GitHub sur un éditeur de code, comme Visual Studio Code, et rentrer “dotnet run” dans le terminal. Le jeu se lancera alors.

## Le choix de l'univers du jeu et ses spécifications

Pour l'univers de ce jeu, nous avons imaginé une bataille fictive mettant en scène la mascotte de notre école : le poulpe ! D'un côté, des étudiants aux couleurs de l'ENSC (violet) se battent pour protéger leur petit poulpe. D'un autre les Trolls, de vils créatures prêtes à tout pour mettre la main sur cet octopode pour un joyeux festin. Et quoi de mieux qu'une partie de foot pour les départager ?

Dans cette bataille acharnée, tous les coups sont permis, y compris frapper son adversaire pour récupérer le ballon. Chaque joueur possède des points de vie et, si ces derniers tombent à zéro, le joueur meurt. De plus, chacune des équipes est dirigée par un coach : le Roi des Trolls pour l'équipe des Trolls et Patate, le légendaire chat vivant sous le patio, pour l'équipe des étudiants (les noms des coachs peuvent être modifiable par l'utilisateur selon l'équipe qu'il choisit d'incarner). A chaque tour, les coachs pourront utiliser des objets de leur inventaire pour soigner ou réanimer leurs joueurs.

De plus, chaque action est soumise à l'aléatoire. D'abord avec un lancé de dé en début de partie pour déterminer quelle équipe à l'avantage. Puis tout au long de la partie avec les joueurs qui peuvent perdre le contrôle de la balle, tomber et se blesser tout seul, rater leurs tirs de ballon ou encore pour le gardien, ne pas réussir à l'attraper. En effet, chaque action des différents joueurs est soumise à l'aléatoire (se déplacer, frapper, contrôler la balle, lancer la balle et rattraper la balle).

Finalement, la partie prend lorsque le nombre de tours, définis au lancement d'une partie, est atteint. Le jeu peut aussi prendre fin si l'ensemble des joueurs d'une équipe, hormis le gardien, meurt.

## Les possibilités des joueurs et le déroulement d'une partie

### Les actions possibles des joueurs

Les joueurs peuvent faire ces différentes actions : se déplacer, contrôler la balle, lancer la balle, frapper un joueur adjacent et intercepter une balle lancée par un joueur.

Ils possèdent ou non la balle, peuvent la lancer quand elle est dans leur contrôle, ils perdent des points de vie (PV) et peuvent revenir à la vie ou être soignés par le coach de leur équipe.

Au niveau des déplacements, selon la classe des joueurs et la position du ballon ainsi que son possesseur, les joueurs ont plusieurs possibilités : ils peuvent avancer vers la balle, vers les buts ou vers un joueur choisi. Dans le cas où leur déplacement les mène à une case déjà occupée par un autre joueur, deux cas de figures s'offrent à eux : soit ils tapent le joueur, soit ils se déplacent aléatoirement sur une case adjacente inoccupée. Si le déplacement d'un joueur échoue, le joueur trébuche : il reste sur place et perd un point de vie.

Au niveau du contrôle de la balle, les joueurs sur la même case du ballon peuvent la récupérer. Dès qu'ils sont en possession de la balle, quand ils se déplacent la balle se déplacera aussi avec eux. Si jamais leur action échoue, il reste au-dessus de la balle sans réussir à la contrôler.

Au niveau des lancers de balle, selon la classe des joueurs, de la position du ballon et celle de son possesseur, les joueurs ont plusieurs possibilités : ils peuvent lancer la balle vers les

buts, vers un joueur ou aléatoirement. Si les joueurs ne réussissent pas, ils restent sur place avec le ballon.

Quand un joueur lance le ballon sur la case d'un autre joueur, celui-ci à une certaine probabilité de récupérer la balle selon la classe à laquelle il appartient. S'il ne réussit pas, la balle continue tranquillement son chemin selon la direction qui lui a été donnée.

## Les différents comportements des joueurs

### Comportement des coureurs

Si le coureur n'a pas la balle, il court vers elle (probabilité de succès : 90%)

Si le coureur est sur la balle, il peut la contrôler (probabilité de succès : 70%). Cette action est combinée en cas de succès avec une action de déplacement sur le même tour de jeu.

Si le coureur a la balle, il se déplace vers les buts (probabilité de succès : 90%)

Si le coureur a la balle et est proche des buts, il lance la balle vers les buts (probabilité de succès : 60%)

Si jamais le coureur veut se déplacer sur une case où un autre joueur est déjà présent :

- Soit le joueur a la balle et le joueur présent sur la case est un allié : dans ce cas le joueur se déplace aléatoirement sur une autre case vide adjacente (probabilité de succès : 90%)
- Soit le joueur n'a pas la balle et le joueur présent est un adversaire : dans ce cas il frappe le joueur qui l'empêche de se déplacer (probabilité de succès : 60%).

Si la balle est lancée sur la case du joueur, il a une probabilité de 80% de récupérer la balle.

Au niveau des paramètres des coureurs, ils possèdent 10 points de vie initiaux et leur force de frappe est de 3.

## Comportement des défenseurs

Si le défenseur n'a pas la balle, il regarde où elle se situe :

- Si un joueur ennemi est sur la même case que la balle, le défenseur se déplace jusqu'à ce joueur (probabilité de succès : 90%). Quand le défenseur arrive à côté de ce joueur , au lieu de se déplacer, il prend la balle au joueur (probabilité de succès : 70%)
- Si la balle est sur la case du défenseur, celui-ci la récupère sur sa case (probabilité de succès : 70%)
- Sinon, le défenseur ne bouge pas

Si le défenseur a la balle :

- Il lance la balle aux joueurs de son équipe le plus proches, sans prendre en compte les gardiens et les autres défenseurs (probabilité de succès : 60%)
- Si il n'y a pas d'autres joueurs dans son équipe que le gardien et que des défenseurs, il lance la balle vers les buts (probabilité de succès : 60%)

Si la balle est lancée sur la case du défenseur, il a une probabilité de 90% de récupérer la balle.

Au niveau des paramètres des défenseurs, ils possèdent 8 points de vie initiaux.

## Comportement des brutes

Si la brute n'a pas la balle, elle se dirige vers le joueur adverse le plus proche (probabilité de succès : 90%). La brute se déplace lentement car elle se déplace en diagonale seulement quand un joueur allié se trouve sur la case où elle souhaitait se déplacer. Quand elle est à proximité d'un joueur adversaire, elle le tape (probabilité de succès : 90%)

Si la brute est en possession de la balle, elle la lance aléatoirement.

Si la balle est lancée sur la case de la brute, elle a une probabilité de 40% de récupérer la balle.

Au niveau des paramètres des brutes, elles possèdent 15 points de vie initiaux et leur force de frappe est trois fois plus élevées que celle d'un joueur d'une autre classe, elle est donc de 9.

### Comportement des gardiens

Les gardiens ont plusieurs niveaux différents (niveau 1, niveau 2 et niveau 3), qui régissent leur façon de se déplacer.

Tant que la balle n'est pas dans les buts, les gardiens se déplacent.

- Niveau 1 : le gardien se déplace aléatoirement en bas ou en haut du plateau. Si le déplacement emmène le gardien hors de la zone des buts, il faut qu'il réussissent un lancé de dé pour se déplacer dans la zone des buts.
- Niveau 2 : le gardien se déplace aléatoirement en bas ou en haut du plateau. Si le déplacement emmène le gardien hors de la zone des buts, il change son déplacement pour se déplacer dans la zone des buts.
- Niveau 3 : le gardien suit le déplacement du ballon, il monte et descend en fonction de la position du ballon.

Quand la balle est lancée dans les buts, si le gardien est sur une case adjacente à la balle, il a une probabilité de l'arrêter (probabilité de succès : (niveau 1 = 50%), (niveau 2 = 65%), (niveau 3 = 85%)).

Si le gardien réussit à arrêter la balle, il fait la passe aux joueurs le plus proche de son équipe qui récupère directement la balle, peu importe la distance. Néanmoins, le gardien a du mal à savoir quels joueurs sont dans son équipe et s'il loupe son jet de dé, il fait la passe aux joueurs adverses le plus proche (probabilité de succès : 85%).

Si la balle est lancée pile sur la case du gardien, il a une probabilité de 70% de récupérer la balle.

Au niveau des paramètres des gardiens, ils possèdent 3 points de vie initiaux.

## Comportement de la balle

La balle à été codé de telle sorte qu'elle puisse avoir le comportement le plus communément admis dans les jeux de balles.

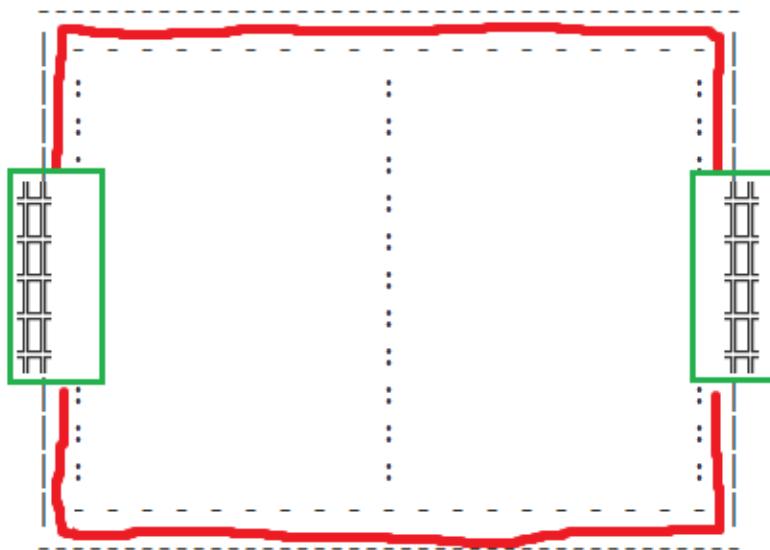
Les différentes méthodes sont :

**La remise à zéro** : pour celle-ci nous prenons en considération les dimensions du terrain sélectionné et l'équipe qui à l'avantage c'est-à-dire qu'on la replace au centre du côté de l'équipe désignée. Si il n'y a pas d'avantage, la balle se place au centre du plateau.

**La détection du hors zone et replacement du ballon** : on vérifie si la ballon est sorti du plateau, soit par un but (zone verte) soit hors du terrain (zone rouge). Si une de ces deux conditions est vraie, on replace la balle au centre du terrain de jeu.

**Le hors zone du ballon** : cette méthode permet de vérifier si le ballon est dans la zone rouge ou non (voir l'image ci-dessous). Elle retourne un booléen.

Terrain moyen :



Capture d'écran n°1 : Hors zone et zone de but sur le plateau

**Le déplacement du ballon** : si on aucun joueur n'a la balle, alors elle continue sa trajectoire en roulant sinon elle reste avec le joueur.

La balle **roule** : nous examinons si la balle a une impulsion. Si oui, elle va avancer comme une boule dans les sens d'orientation établis préalablement.

L'obtention de la **balle par au moins un joueur**, on analyse un à un les joueurs sur le plateau. Si au moins un des joueurs a la balle, on retourne vrai, faux sinon.

La **modification de la position** : permet à la balle de modifier sa position et sa direction. Elle peut aller dans toutes les directions du plan tant les points cardinaux que les diagonales.

Le **changement de coordonnées du ballon** : si on souhaite replacer à un endroit précis notre ballon c'est cette méthode qu'il faudra utiliser. Elle sert notamment pour suivre la position du joueur qui contrôle la balle.

Le **ballon statique** : on réinitialise les paramètres du ballon à zéro. En effet, lorsqu'il roule ou est contrôlé par un joueur, il obtient et conserve certaines caractéristiques. De fait, on remet les compteurs à zéro pour avoir un comportement neutre ou il n'aura aucun mouvement et il restera à son emplacement de façon statique.

**Marquer un but** : cherche à vérifier si le ballon est bien dans la zone verte de but, et si oui alors on examine de quel côté le but a été mis afin d'attribuer le point au bon groupe de joueurs.

Le **changement de score** : lorsque le ballon rentre dans les buts, on attribue un point à l'équipe ayant tiré dans le camp adverse.

## Déroulement d'une partie

Au début d'une partie, nous proposons à l'utilisateur(s) de regarder **Les règles du jeu** ou de **Choisir les paramètres** pour lancer une partie.

**Les règles de jeu** expliquent l'univers du jeu, donnent des indications sur les différentes classes des joueurs et de la gestion de l'inventaire.

**Le choix des paramètres** demande à l'utilisateur le mode de jeu qu'il veut sélectionner (mode solo, mode deux joueurs ou mode automatique), puis il demande le choix du terrain avec lequel il souhaite jouer (petit, moyen ou grand). Ensuite, il lui demande s'il souhaite entrer les prénoms des huit joueurs qui vont s'affronter sur le terrain ou reprendre les prénoms sauvegardés d'une partie précédente. S'il change les prénoms des joueurs, il peut choisir de sauvegarder les nouveaux prénoms sélectionnés. Enfin, le joueur peut choisir de

modifier la classe des différents joueurs (hors gardien). Finalement, il sélectionne le nombre de tours de jeu qu'il veut et la partie commence.

A chaque tour de jeu, les joueurs se déplacent. En dessous du plateau, les phrases expliquant les déplacements de chaque joueur apparaissent, tandis que les informations relatives aux joueurs apparaissent à droite. Puis on demande le choix des deux coachs (soigner leurs joueurs ou ne rien faire). La partie se poursuit ainsi jusqu'à la fin de la partie.

La partie prend fin quand le nombre de tours de jeu est épuisé ou quand tous les joueurs d'une équipe (hors gardien) sont morts. Pour déterminer le gagnant, on regarde quelle équipe à marquer le plus de but. Si les équipes ont le même score, c'est l'équipe avec le plus de joueurs en vie qui l'emporte. Si le nombre de joueurs en vie est égal entre les deux équipes, alors la partie se conclut par une égalité.

## Les différents modes de jeu

Il existe trois modes de jeu : le mode solo, le mode multijoueur et le mode automatique.

### Mode solo

Dans le mode solo, on demande à l'utilisateur de choisir l'équipe qu'il souhaite faire gagner (équipe des trolls ou des étudiants de l'ENSC) et de sélectionner son nom. Lors de ce mode, après chaque tour de jeu, on demande à l'utilisateur s'il souhaite soigner son équipe ou ne rien faire. Le coach de l'équipe adverse est gérée par une IA qui a une probabilité de 30% de réanimer un de ses joueurs morts (si plusieurs joueurs sont morts elle soigne en priorité le joueur 1, puis le 2, puis le 3) et une probabilité de 30% de soigner un de ses joueurs qui a perdu au moins 3 points de vie (elle soigne dans l'ordre le joueur avec le moins de point de vie).

### Mode deux joueurs

Dans le mode deux joueurs, on demande à chaque utilisateur de donner leur nom à l'équipe de leur choix. Ensuite à chaque tour de jeu, on demande à chaque utilisateur s'il souhaite soigner des membres de leur équipe ou ne rien faire.

### Mode automatique

Dans le mode automatique, à chaque tour de jeu, les IA coachs de chacune des équipes jouent automatiquement. Elles ont une probabilité de 30% de réanimer un de leurs joueurs morts (si plusieurs joueurs sont morts elle soigne en priorité le joueur 1, puis le 2, puis le 3) et une probabilité de 30% de soigner un de leurs joueurs qui a perdu au moins 3 points de vie (elle soigne dans l'ordre le joueur avec le moins de point de vie).

Dans ce mode de jeu on ne demande plus à l'utilisateur de rentrer un choix à chaque tour, les actions s'enchaînent les unes après les autres.

## Interface et plateau de jeu

### Plateau



Capture d'écran n°2 : Différentes tailles de plateau disponible

Il existe 3 tailles de plateau différentes : un petit (9x23), un moyen (15x39) et un grand (21x55). Ces derniers sont divisés en 2 par une ligne fictive pour faciliter la lecture ainsi que de but.

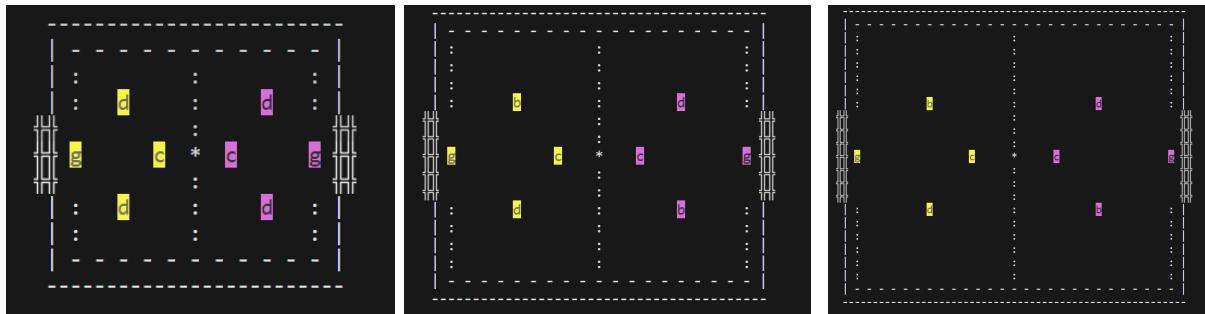
Le plateau est un tableau de int (int[,]) où l'on inscrit les symboles ":" et "—" là où c'est nécessaire. C'est ensuite avec la méthode ToString() que l'on fait apparaître les contours du terrain comme par exemple les buts. Une distinction est faite entre le lieu où les joueurs et le ballon peut se placer et les décors autours où les joueurs et le ballon n'iront jamais.

Passons maintenant au choix des dimensions : pour les lignes, il faut choisir un multiple de trois. En effet, le plateau est constitué de 3 parties (de haut en bas) dont la centrale contient les espaces pour les cages de but. Pour le nombre de colonnes, il faut un nombre impair pour que la ligne du centre soit bien placée de façon esthétique.

## Interface du jeu

### Placement sur le plateau en début de partie

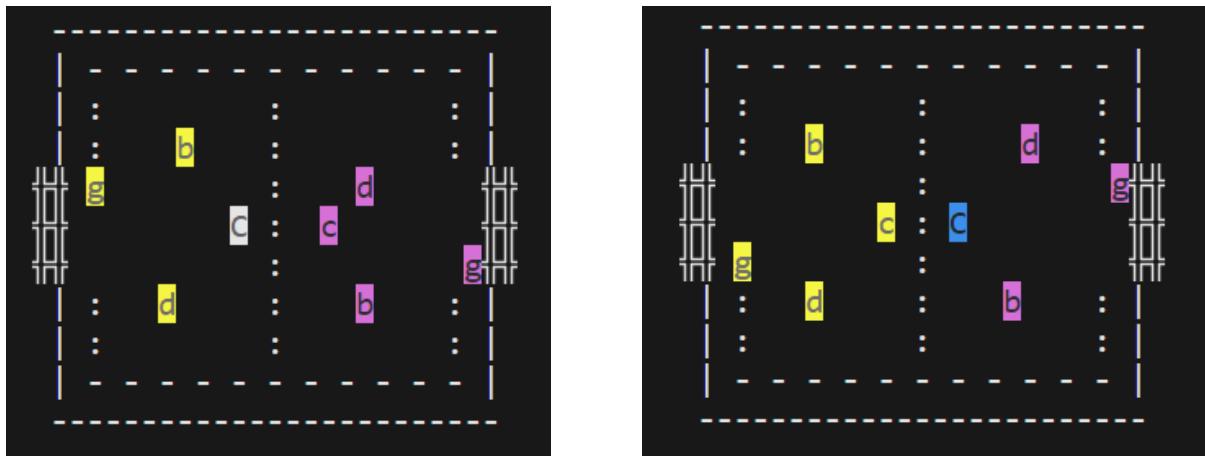
C'est dans la simulation que l'on va placer les joueurs sur le plateau. Or, en fonction de sa taille, les coordonnées des joueurs vont varier. Il faut donc réfléchir à une manière optimale d'écrire le code. Les différents calculs exposés ci-dessous ont été trouvés après plusieurs tests. Pour simplifier l'écriture ici, nous utiliserons "i" pour le nombre de lignes et "j" pour le nombre de colonnes.



Capture d'écran n°3 : Placement des joueurs en début de partie en fonction du terrain choisi

- Les joueurs n°1 (les coureurs sur l'image, symbolisés par un “c”) sont les plus proches du ballon. Leurs coordonnées sont pour la ligne  $\frac{1}{2}*i$  et pour la colonne  $\frac{1}{2}*j \pm \frac{1}{3}*i$ .
- Pour les joueur n°2 (défenseurs les plus haut), les coordonnées sont pour la ligne  $\frac{1}{3}*i - 1$  et pour la colonne  $\frac{1}{2}*j \pm \frac{2}{3}*i$ .
- Pour les joueurs (défenseurs les plus bas), les coordonnées sont pour la ligne  $\frac{2}{3}*i$  et pour la colonne  $\frac{1}{2}*j \pm \frac{2}{3}*i$ .
- Pour les gardiens (au niveau des cages, symbolisés par la lettre g), les coordonnées sont pour la ligne  $\frac{1}{2}*i$  et pour la colonne 1 ou j-2.

Visualisation des informations sur le plateau



Capture d'écran n°4 : Coloration des joueurs en fonction de leur équipe

Chacun des joueurs est représenté par une lettre. Celle-ci change en fonction de sa classe : “c” pour le Coureur, “d” pour le Defenseur, “b” pour la Brute et enfin “g” pour le Gardien. Les équipes sont distinguées visuellement grâce au couleur : du jaune pour l'équipe 1 des Trolls et le violet pour l'équipe 2 des humains. Lorsque qu'un joueur possède la balle, sa lettre passe en majuscule et sa couleur change (le jaune devient blanc et le violet devient bleu).

## Visualisation des informations durant la partie

```
*** SCORE ***
-----
| 000 : 000 |
-----
|-----|-----|
|   :   :   :   |
| :   b   :   d   :   |
|   :   :   :   :   |
|-----|-----|
|   :   :   :   |
|   g   :   c   *   c   :   g   :   |
|   :   :   :   :   |
|-----|-----|
|   :   d   :   b   :   |
|   :   :   :   :   |
|-----|-----|-----|-----|
INFORMATIONS | TOURS : 1/50
-----
Equipe n°1 | Nb de Baie = 1 | Nb de Baie Magique = 1
J1 : coureur Goh : 10/10      | J2 : brute Gah : 8/8
J3 : défenseur Guh : 8/8     | J4 : gardien Guih : 3/3

Equipe n°2 | Nb de Baie = 1 | Nb de Baie Magique = 1
J1 : coureur Kloé : 10/10     | J2 : défenseur Lallie : 8/8
J3 : brute Martin : 8/8       | J4 : gardien Cookie : 3/3
```

*Capture d'écran n°5 : Informations durant la partie*

A coté sur plateau sont présentes de nombreuses informations tel que le le nombre de tour joués et le score. On trouve aussi pour chaque équipe l'inventaire des coachs mais aussi des informations plus précises sur les joueurs tels que leur numéro, leur prénom, leur classe et enfin leur nombre de points de vie. Les prénoms sont écrits en bleu pour mieux les distinguer.

INFORMATIONS	TOURS	: 45/50
<hr/>		
Equipe n°1	Nb de Baie = 1	Nb de Baie Magique = 1
J1 : coureur Goh	: 6/10	J2 : brute Gah : 7/8
J3 : défenseur Guh	: 2/8	J4 : gardien Guih : 3/3
Equipe n°2	Nb de Baie = 0	Nb de Baie Magique = 0
J1 : coureur Kloé	: 0/10	J2 : défenseur Lallie : 0/8
J3 : brute Martin	: 4/8	J4 : gardien Cookie : 3/3

Capture d'écran n°6 : Colorisation différentes des points de vie des joueurs en fonction de leur état

Le nombre de points de vie restant pour chaque joueur s'affiche d'une certaine couleur en fonction de l'état du joueur. Si celui =-ci est en pleine forme, alors ses PV seront en vert, en jaune si son état est moyen, rouge si sont était est crotique et enfin gris si le joueur est mort.

## Informations complémentaire textuel action du coach

```

*** SCORE ***
-----
| 000 : 000 |
-----

INFORMATIONS | TOURS : 1/50
-----
Equipe n°1 | Nb de Baie = 1 | Nb de Baie Magique = 1
J1 : coureur Goh : 10/10 | J2 : brute Gah : 8/8
J3 : défenseur Guh : 8/8 | J4 : gardien Guih : 3/3

Equipe n°2 | Nb de Baie = 1 | Nb de Baie Magique = 1
J1 : coureur Kloé : 10/10 | J2 : défenseur Lallie : 8/8
J3 : brute Martin : 8/8 | J4 : gardien Cookie : 3/3

Le match affrontant l'équipe des Trolls dirigé par Roi des Trolls et
l'équipe des Etudiants de l'ENSC coaché par Patate est sur le point de commencer !

Appuyer sur 'Entrée' pour déterminer quelle équipe commence. █

```

Capture d'écran n°7 : Commentaires sous le plateau de jeu pour donner davantage d'informations sur la parité en cours

Enfin, en dessous du plateau, on retrouve des commentaires sur la partie. A chaque tour, les commentaires indiquent ce qu'il s'est passé. Le coach (l'utilisateur du jeu) peut ensuite continuer la partie ou de soigner ses joueurs.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
*** SCORE ***
-----
| 000 : 000 |
-----

INFORMATIONS | TOURS : 5/50
-----
Equipe n°1 | Nb de Baie = 1 | Nb de Baie Magique = 1
J1 : coureur Goh : 6/10 | J2 : brute Gah : 8/8
J3 : défenseur Guh : 8/8 | J4 : gardien Guih : 3/3

Equipe n°2 | Nb de Baie = 1 | Nb de Baie Magique = 1
J1 : coureur Kloé : 10/10 | J2 : défenseur Lallie : 8/8
J3 : brute Martin : 8/8 | J4 : gardien Cookie : 3/3

'(T_T)'!. Goh tombe, il a perdu 1 point de vie. :-D
'Tu ne mangeras pas les poulpes, saleté de troll !'. Kloé frappe Goh.
'Bien fait!'. Goh a perdu 3 points de vie

-----
C'est au tour de l'équipe 1. Roi des Trolls à toi de jouer !

-----
| Continuer = 1 | Soigner = 2 |
-----

Choix : █

```

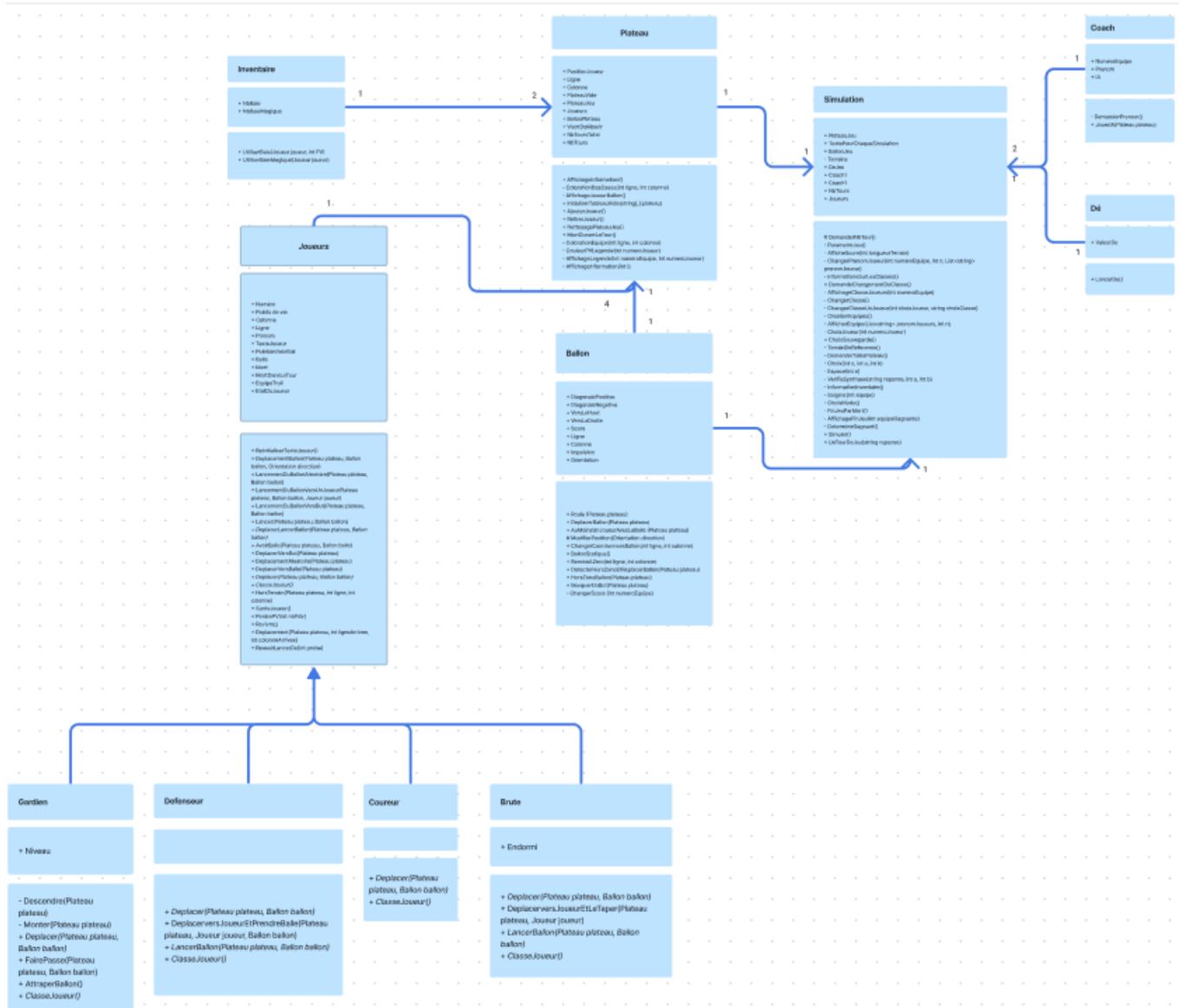
Capture d'écran n°8 : Exemple de visualisation du jeu durant une partie

## Modélisation UML :

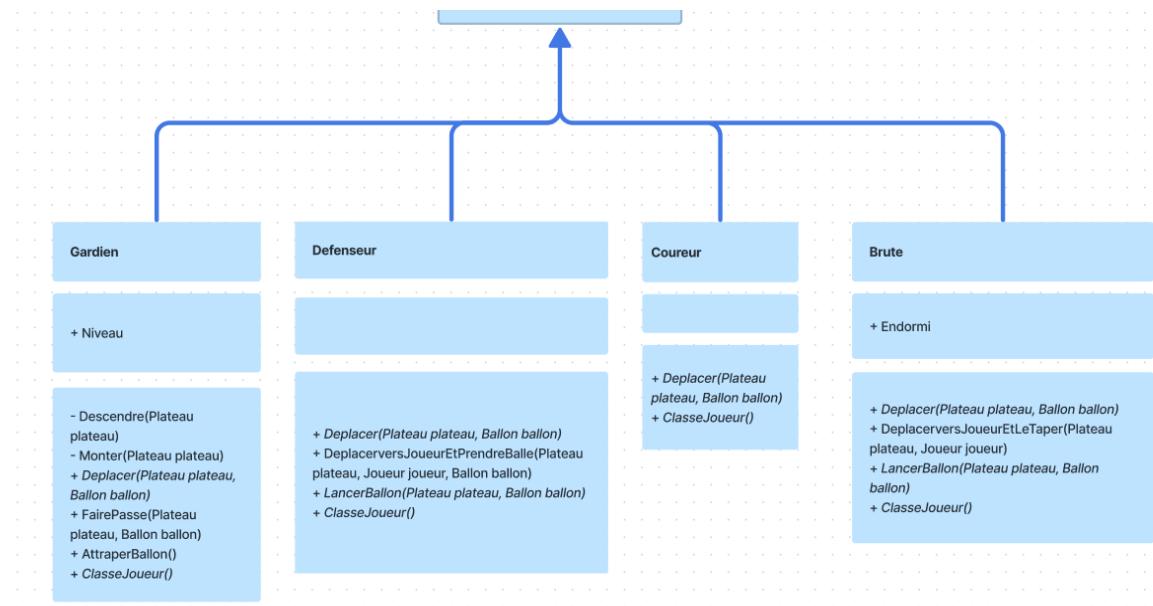
Voici les diagrammes UML que nous avons réalisé sur Figma (disponible au lien suivant :

<https://www.figma.com/file/p6y9gMujoveOR8tggmEERV/Diagramme-UML-du-projet-PoulyphFoot?type=whiteboard&node-id=0%3A1&t=cBqOpQCvZPfp70Mj-1> ).

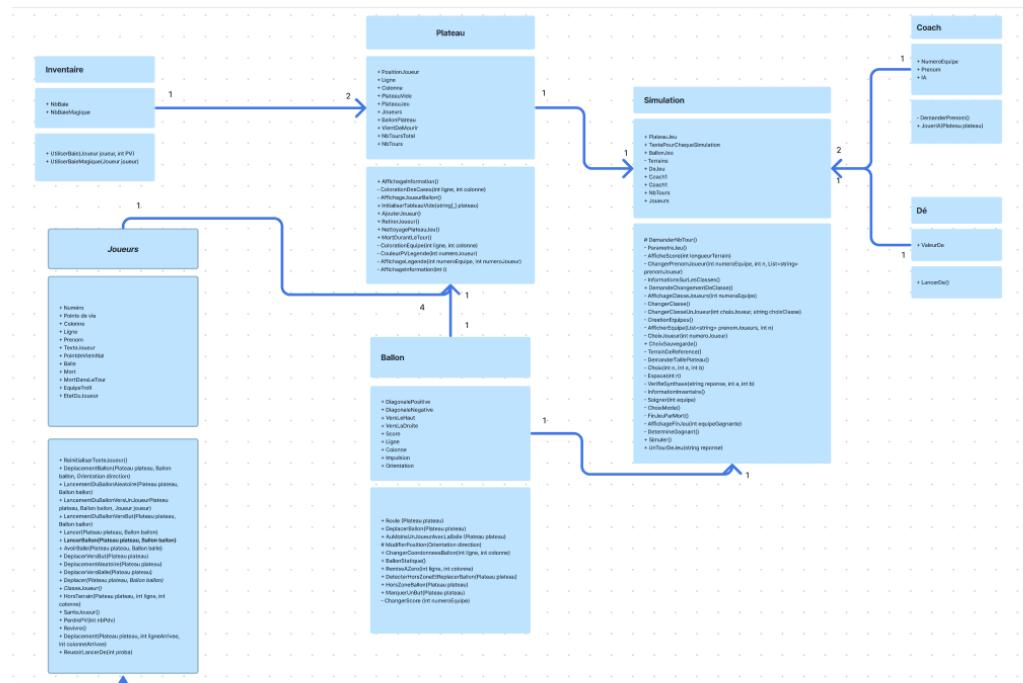
Diagramme UML du projet PouppyFoot



Capture d'écran n°9 : Diagramme de UML global du projet



Capture d'écran n°10 : Diagramme des classes héritées de la classe Joueurs : Gardien, Défenseur, Coureur et Brute.



Capture d'écran n°11 : Diagramme global sans les classes héritées de joueurs

# Tests et gestion des erreurs

## Gestion des erreurs internes

Dans ce projet nous n'avons pas rencontré de gros bug qui nécessite de recommencer entièrement une partie du code.

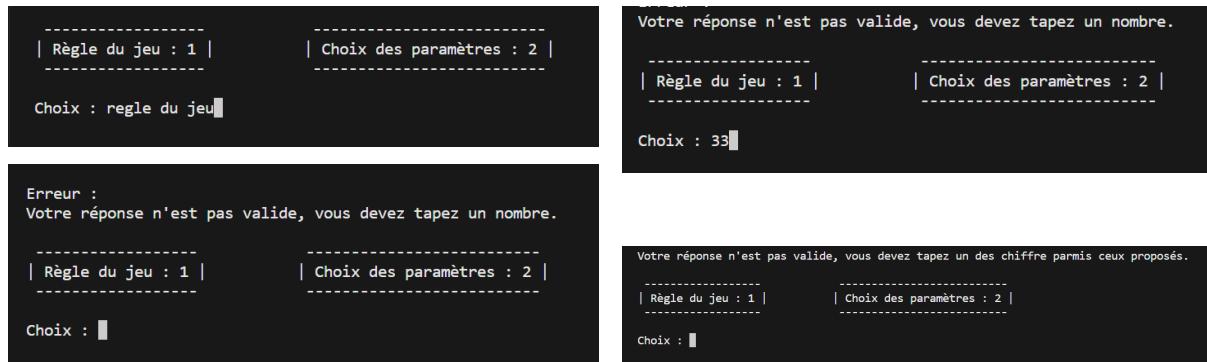
Il arrivait de rencontrer quelques conflits lorsque l'on voulait merge une branche sur le main sur GitHub, mais la plupart du temps, ces conflits étaient bénins. Nous avions toujours la possibilité de regarder l'historique de la branche pour comprendre les changements ou encore, demander à aux membres du groupe leurs avis si nous n'étions pas sûrs.

Nous avons pensé intéressant de faire tester notre code à plusieurs reprises. Tout d'abord, par des membres du groupe pour leur proposer de nouvelles fonctionnalités, par exemple avec la création de l'inventaire ou des modes de jeux pour voir si l'interface est intuitive et claire. Faire tester à d'autres membres du groupe permet aussi de découvrir d'autres bugs que nous n'aurions pas remarqué individuellement.

Enfin nous avons réalisé des tests avec des personnes extérieures du projet. Ces personnes ont un point de vue beaucoup plus intéressant que le nôtre puisqu'elles ne sont pas biaisées. Il est donc intéressant de savoir ce qu'elles n'ont pas compris. Par exemple, un des participants ne comprenait pas combien de prénoms il devait inscrire pour son équipe, c'est pourquoi nous avons ajouté cette information en indiquant "4 Joueurs". De la même façon, un autre participant indiquait ne pas bien comprendre où était le ballon sur le plateau lorsqu'il était possédé par un joueur. Même si, à ce moment-là, le joueur possédant le ballon avait sa lettre en majuscule, ce n'était pas assez visible. C'est pourquoi, nous avons réalisé un affichage supplémentaire en changeant aussi la couleur du joueur.

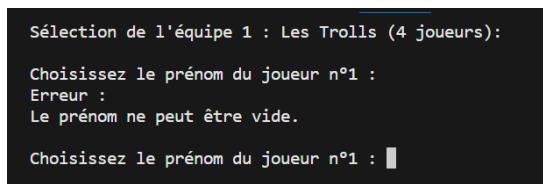
## Gestion des erreurs externes

Des erreurs peuvent aussi arriver à cause des actions de l'utilisateur. En effet, propose une petite interaction avec l'utilisateur, mais il faut encore que ce dernier n'écrit pas n'importe quoi.



Capture d'écran n°12 : Messages d'erreurs différents en fonction de l'erreur de l'utilisateur

Aussi la méthode `VerifieSynthaxe` va permettre de vérifier ce que l'utilisateur écrit pour ne pas faire planter le jeu. Si ce dernier se trompe effectivement, un message d'erreur s'affiche. Le message diffère en fonction de l'erreur : si le texte entré n'est pas un nombre ou si le nombre entré n'est pas valide.



Capture d'écran n°13 : Message d'erreur si le prénom choisi est vide

On empêche aussi l'utilisateur de choisir des prénoms vides pour ses joueurs, cela ne générera pas d'erreur mais un manque de compréhension durant la partie.

```
Que voulez utiliser ?  
-----  
| Baie (0) = 1 | | Baie Magique (1) = 2 | | Retour = 0 |  
-----  
Choix : |
```

```
Vous n'avez plus de baie !  
*** SCORE ***  
-----  
| 000 : 000 |  
-----|
```

Capture d'écran n°14 : Message d'erreur si l'utilisateur choisit un objet qu'il n'a plus

L'utilisateur ne peut pas utiliser de baie s'il n'en a plus. On l'informe aussi avec un message d'erreur.

## Gestion de projet : l'organisation de l'équipe

Notre groupe étant composé de trois étudiants de formations différentes, nous avons choisi d'adapter le travail en fonction des savoir-faire de chacun des membres pour optimiser le temps qui nous était donné. Durant ce projet, nous avions à réaliser quatres grandes tâches : l'écriture des trois classes principales (le plateau, les joueurs et le ballon) avec en parallèle celle de la simulation, la correction de bug/aide et enfin la rédaction du rapport. Il nous a semblé important de répondre en priorité au sujet du projet avant d'ajouter des compléments bonus. Dès que nécessaire, nous faisions des réunions afin d'harmoniser le code et sa structure.

### Planning de réalisation des tâches

Dans un premier temps, nous avons discuté globalement de l'idée finale de ce que nous voulions faire. En effet, avant que chacun des membres du groupe travaille en autonomie, il est important de se mettre d'accord sur certains points afin de pouvoir se coordonner rapidement et de partir dans la même direction. C'est notamment le résultat visuel du plateau qui nous a semblé important de clarifier rapidement. En effet, celui-ci portera les joueurs et le ballon durant les parties et a donc un impact direct sur la réflexion des coordonnées de chacun.

Ainsi, durant la première séance, nous nous sommes mis d'accord sur l'aspect du plateau de jeu.

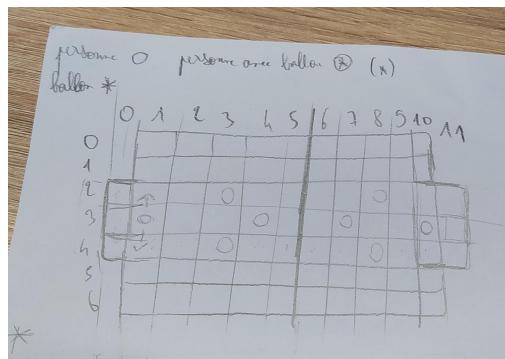


Figure n°1 : Dessin du terrain de jeu imaginé lors de la première séance

Cette première séance nous a aussi permis de réfléchir à la structure globale du programme et aux futurs classes et méthodes à produire. Pour simplifier la représentation, nous avons choisi de travailler sur Miro.

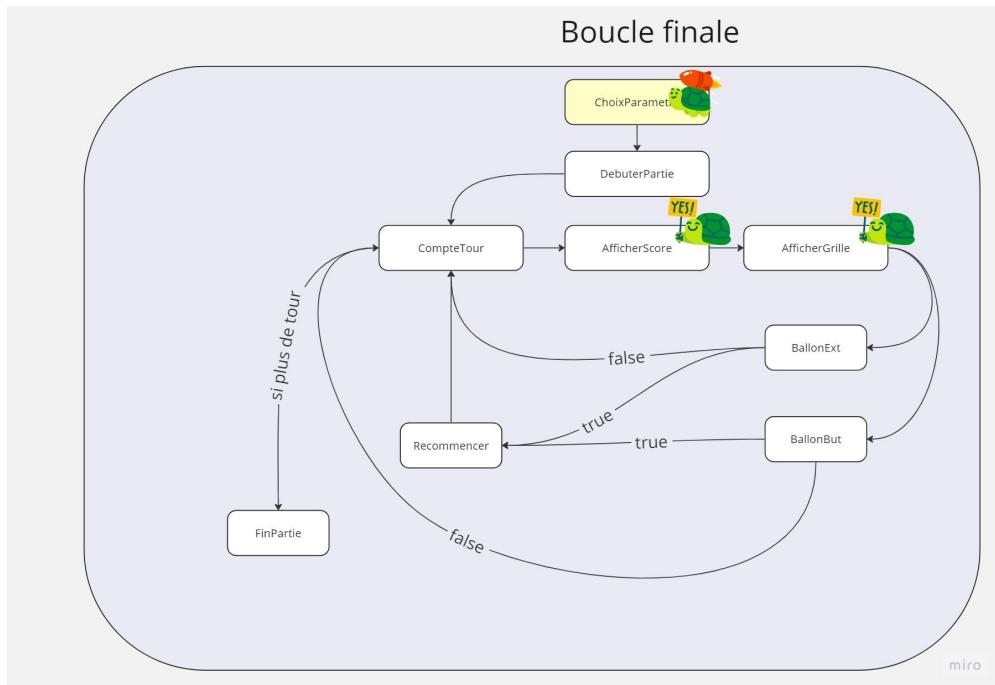


Diagramme n°1 : Représentation simplifiée de la structure du programme final réalisée, lors de la première séance sur Miro

Le code final est en vérité bien plus fourni bien sûr, mais faire ce petit diagramme nous a permis de structurer nos idées et a servi de base lors des premières heures de codage.

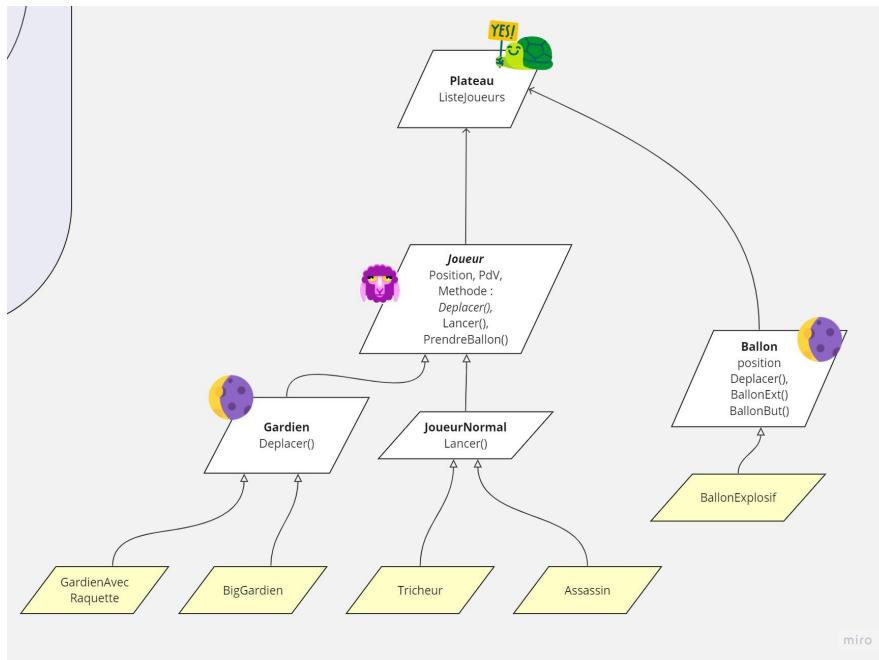


Diagramme n°2 : Représentation simplifiée de la structure des classes principales, réalisée lors de la première séance sur Miro

Nous avons enfin cherché les classes indispensables pour que le jeu soit jouable et intéressant assez rapidement. Nous avons alors identifié le Ballon, les Joueurs comportant des héritages avec Gardien et JoueurNormal, et enfin le Plateau dans lequel les ballons et les joueurs prendront vie. Certaines de ces classes et associations seront amenées à bouger par la suite comme par exemple le Ballon qui n'est plus dans Plateau mais dans Simulation ou encore JoueurNormal qui va se diviser en plusieurs héritages de Joueurs différents.

Faire ce schéma sur Miro nous a aussi permis de nous répartir les tâches et d'indiquer, grâce à l'utilisation d'emoji, l'avancement de l'écriture de chaque classe et de connaître rapidement d'un coup d'œil comment le projet avance.

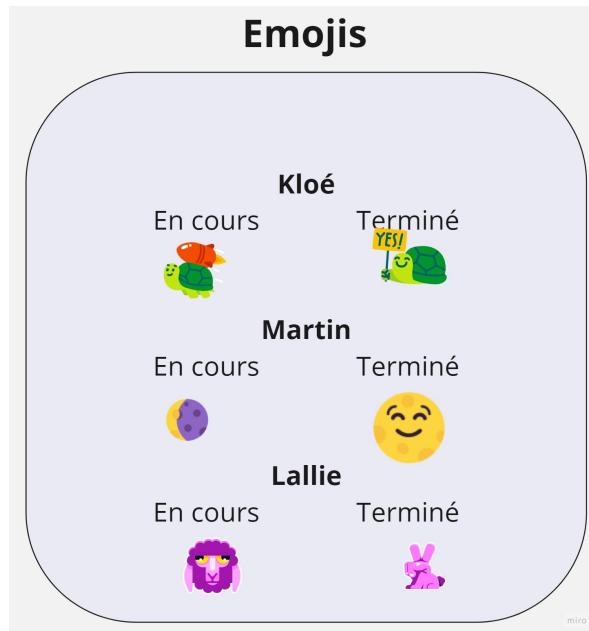


Diagramme n°3 : Utilisation des emojis sur Miro pour connaître facilement l'avance du projet

Ainsi, des emojis sont attribués à chacun des membres du groupe avec notamment une distinction entre une tâche en cours de réalisation et une tâche terminée.

A terme, nous avons finalement arrêté d'utiliser Miro et ce pour 2 raisons : D'abord, la version gratuite bloque rapidement l'équipe à réaliser des modifications, ce qui oblige à créer un nouveau document régulièrement. De plus, le code évoluant très rapidement, les liens entre les méthodes évoluent aussi rapidement. Il est alors beaucoup plus simple de communiquer directement avec les membres du groupe pour parler d'éventuelles remarques ou interrogations.

L'utilisation de la matrice d'implication a finalement été le moyen de plus simple pour connaître l'avancement du projet. Grâce à la mise en commun des idées durant la première séance pour la structuration du code, nous avons ensuite pu nous répartir sur les trois principales classes du jeu : le Plateau, les Joueurs (et leur héritage) et le Ballon.

## Communication au sein du groupe

Pour ce projet, nous avons utilisé GitHub pour partager régulièrement nos codes. L'utilisation des branches permet à chaque membre du groupe de pouvoir travailler en parallèle. Chacun possédait sa branche personnelle dans laquelle chacun avançait sur une des trois classes moteur du projet. Par la suite, de plus petites branches sont apparues, que ce soit pour la correction de petits bugs, pour la correction de syntaxe, pour le rangement de fichier pour rendre le tout plus lisible ou encore pour l'ajout des commentaires ou d'autres fonctionnalités.

Si des séances de TP étaient dédiées à la réalisation du projet, ce dernier demande aussi une grande part d'autonomie. Aussi, il nous arrivait d'avoir des interrogations ou des remarques à faire lorsque nous n'étions pas à l'école. Nous avons donc utilisé une conversation de groupe sur Messenger pour communiquer facilement et rapidement à distance des avancées du projet. Ce dernier outil permet aussi l'envoie de photos et de vidéos ce qui permet un échange plus dynamique entre les membres du groupe

Les séances de TP étaient utilisées comme un créneau dédié aux partages d'idées et d'évolutions du projet plus complets que lors des conversations écrites. Elles permettaient aussi de demander des conseils ou de l'aide aux autres membres du groupe si nécessaire.

## Réflexion sur l'ajout de fonctionnalité bonus

L'idée de bonus a rapidement été évoquée avec par exemple un ballon explosif, des cases de laves dans lesquelles les joueurs pourraient se noyer et mourir instantanément des joueurs "Tricheur" ou "Assassin"... Ces idées sont symbolisées dans le schéma par des cases en jaune pour indiquer qu'elles sont facultatives et ne sont donc pas indispensable.

Finalement, ces dernières idées n'ont pu être réalisées, mais d'autres comme le choix de la dimension du terrain, les différents types de joueurs ou encore le niveau du gardien ont rapidement vu le jour. Plus tard, ce sont les différents modes de jeu, les coachs et leur inventaire respectif qui naîtront.

# Bilan critique du projet

## Ressenti individuel

### Ressenti de Kloé

Ce projet m'a permis de découvrir une autre facette de la programmation à savoir celle Orientée Objet. Ce que je trouve intéressant avec ce type de programmation c'est cette manière de pouvoir mieux structurer les idées et les choses.

J'ai beaucoup aimé le sujet proposé puisqu'il était très ludique et offrait une grande liberté pour la proposition de nouvelles idées et bonus.

Pour ma part, j'ai principalement travaillé sur le Plateau et la Simulation, autrement dit l'interface du jeu visible par le(s) joueur(s). En plus des choix multiples lors du choix des paramètres, chose que j'avais déjà pu expérimenter durant le projet du S5, ce projet fut aussi un prétexte pour apprendre à utiliser les fonds et les polices de couleur.

Par rapport à notre organisation, puisque nous avions rapidement établi les grandes lignes du jeu et que nous nous étions répartis les tâches en fonction des classes, nous pouvions être assez autonome, du moins au début. Ainsi, nous pouvions produire indépendamment les classes Plateaux, Joueurs et Ballon pendant une longue période. Puis, lorsque les classes étaient assez fournies, nous pouvions retravailler légèrement le code pour que tout soit uniforme. Nous avons en général passé notre code à l'autre membre du groupe pour que ce dernier puisse éventuellement corriger à sa guise certaines choses pour que tout fonctionne normalement.

Travailler en groupe ne m'a pas semblé difficile puisqu'il est simple de retrouver des informations dans les codes en les relisant simplement. Cependant je reconnais pour ma part que je ne mettais pas tout de suite de commentaires pour expliquer mon travail, ce que j'aurais dû faire plus tôt pour alléger les lectures de mes camarades. C'est donc quelque chose que j'ai essayé de corriger rapidement lorsque je me suis occupé des nouvelles classes.

Finalement, ce projet me motive davantage à la réalisation du projet individuel qui aura lieu l'an prochain.

### Ressenti de Lallie

Ce projet de programmation avancée m'a permis de mettre plus en pratique les classes et de mieux comprendre comment elles peuvent s'agencer entre elles pour créer des structures qui offrent plus de possibilités de création. Elle m'a également fait réfléchir sur la façon de rendre les informations simples à comprendre par l'utilisateur quand il y a beaucoup d'informations à retranscrire sur l'interface graphique très simple du terminal de commande.

Ce projet étant un projet à trois, il m'a également permis de me rendre compte de la difficulté de travailler ensemble sur du code commun si l'on est pas organisé.

Heureusement, nous avons réussi à mon avis à être assez organisé pour pouvoir avancer chacun de notre côté indépendamment des autres sans se marcher dessus. Ainsi, il était assez plaisant de travailler ensemble et je trouvais assez fascinant que chacun ait une manière de coder différente alors que nous avons suivi les mêmes cours depuis le début de l'année.

Cependant, même en étant bien organisé, il est parfois un peu complexe de fusionner les branches quand on a différents codes créés par différentes personnes qui entrent en conflit. Ou alors de retrouver une erreur dans un programme faisant appel à plusieurs classes. On se rend alors vraiment compte de l'importance cruciale des commentaires car il peut être difficile de déchiffrer le code créé par d'autres personnes. Le débogueur est aussi très utile pour retrouver l'endroit où une erreur a été commise.

## Ressenti de Martin

Ce projet m'a permis de pratiquer la programmation orientée objets qui m'est familière et d'avancer progressivement dans mon apprentissage du langage C# qui, elle, m'est nouvelle. Les notions de classes, de constructeurs, d'attributs et de méthodes appliquées au C# sont comprises et en cours d'assimilation. Cependant, je me suis rendu compte que lorsque les tâches qui me sont confiées sont abstraites, j'ai plus de difficultés à être efficace dans la conception et réalisation. De fait, cela apporte de la confusion au code. Il est donc essentiel pour moi d'ajouter une étape de pré-conception où on y établit les grandes lignes des fonctionnalités à coder pour que chacun des membres puissent être autonomes et que cela apporte de la clarté à l'implémentation, mais aussi un respect des délais des tâches afin de ne pas impacter le projet. Au niveau de la rédaction du code, je me suis rendu compte que souvent, je le complexifie alors qu'il y avait plus simple au premier abord. Cela m'aurait (conditionnel je pense, si je comprends bien le sens de ce que tu veux dire avec la phrase précédente) permis d'appréhender la partie de codage avec plus de simplicité. La gestion de groupe s'est bien passée cependant la mise en commun des codes est une tâche complexe, notamment à trois car chacun à sa logique, ce qui demande l'utilisation des commentaires et de nombreuses discussions afin de dérouler uniformément notre stratégie.

## Conclusion

Le langage de programmation C# est utilisé dans le domaine du développement informatique pour les fonctionnalités complètes qu'il propose. De plus, l'orientée objet est une pratique couramment utilisée, notamment en entreprise, pour la conception d'application sous la forme d'un ensemble de briques logicielles, rendant alors la programmation plus flexible. Aussi, cela offre une meilleure maintenance du code, une bonne portabilité et une fluidité d'exécution qui permet de faire tout type de projet comme ici la création du jeu qui nous a été demandé.

L'aboutissement de ce projet avec la mise en place du jeu Poulpo Bowl nous a donc permis de mettre en pratique la programmation orienté objet.

Ce travail en groupe de trois nous a également permis de nous confronter à des obstacles d'organisation de groupe. Nous avons essayé de surmonter les obstacles en partageant et en ajustant nos compétences et également en faisant preuve d'écoute de chaque membre du groupe et de savoir-être .

Nous avons réussi à mettre en place de nombreuses fonctionnalités supplémentaires comme par exemple avec les différents modes de jeu, la création de l'inventaire ou encore les différents types de joueurs. Ce jeu pourrait néanmoins être amélioré comme par exemple avec les idées que nous avions évoqué précédemment avec un ballon explosif et des cases de lave. Par la suite, nous avons aussi pensé à mettre un nombre de joueur plus important sur le grand plateau, proposer plus d'objet dans l'inventaire, faire apparaître aléatoirement des objets sur le plateau durant la partie que les joueurs pourraient ramasser, créer d'autre classe de joueurs avec notamment un mage qui pourrait soigner ses coéquipier proche ou un joueur qui créerait des dégat de zone.

Même si nous avons tenté de rendre notre jeu le plus accessible et clair, avec notamment les informations sur le côté droit du plateau et les couleurs, il est possible que l'affichage ne soit pas encore optimal. La réalisation de tests utilisateur supplémentaires permettraient d'obtenir de nouvelles pistes d'améliorations.