Game Design Document
- Dossier de conception -

Projet : Paths of Glory

Projet réalisé du 2 mars 2020 au 10 juin 2020.

Équipe de projet :

- Thomas CIANFARANI
- Océane CARPENTIER
- Alex TOMASIA
- Vanessa ASEJO CASPILLO

Sommaire

Game Design Document

Introduction **3**
- Présentation du document 3
- Présentation du jeu 4
- Origines du projet 5
- Consignes et contraintes du projet 6

Conception et Game Design **7**
- Définition du jeu 7
- Implémentation du Shifumi 8
- Benchmark et univers 9
- Déroulement d'une partie et d'une bataille 13
- Simulation 14
- Maquettes graphiques 15
- Description des intelligences artificielles 17

Gestion de projet **19**
- Macro planning 19
- Story mapping 20
- Trello 21
- Répartition du travail 23
- Estimation du temps de travail sur le projet post-réalisation 24

Génie logiciel **25**
- Technologies utilisées 25
- Architecture logicielle 26
- Explication technique de l'IA 27
- Documentation technique 28
- Organisation des tests 30

I. Introduction

Présentation du document

Ce document constitue un **recueil complet d'informations** sur le projet Paths of Glory.

Il détaille l'ensemble du processus de création du jeu vidéo en présentant le jeu et son contexte, la conception et le game design, l'infographie ainsi que le génie logiciel et la gestion de projets.

Des captures et extraits des documents produits y seront également présentés.

Présentation du jeu

Path of Glory est un jeu de stratégie “tour par tour”.

Le joueur incarne un commandant qui a été sélectionné pour prendre part à la Ligue des commandants d'Élite de son Empire.

Son objectif sera de devenir le nouveau champion de la Ligue en venant à bout des 10 commandants en faisant partie.

Le joueur fera donc face à une difficulté progressive lors de batailles de 15 manches où chacune s’appuiera sur le système de Shi-Fu-Mi étendu.

Afin de devenir le nouveau Grand Champion de la Ligue, le joueur devra développer au fil des batailles une méthodologie stratégique lui permettant de s’adapter à ses 10 adversaires uniques.

Origines du projet

Ce projet a été demandé pour le cours de "Génie Logiciel" afin de mettre en pratique les méthodes de génie logiciel vues et étudiées en cours.
De plus, ce projet s'appuie sur les cours de "Programmation et outils" et "Systèmes d'interaction".

Son **objectif** est le développement d'un jeu impliquant le système de Shifumi (Pierre-papier-ciseaux) lors de confrontations entre le joueur et l'ordinateur.

L'équipe de projet est composée d'un groupe impliquant 4 apprentis ingénieurs spécialisés dans la programmation dans le domaine des médias numériques.

D'un **point de vue pédagogique**, ce projet sert à évaluer la programmation multimédia avec le moteur de jeu Unity3D, la mise en place de génie logiciel, l'expérience utilisateur et la gestion de projet agile.

Voir la note de cadrage produite.

Consignes et contraintes du projet

Voici les consignes et contraintes du projet telles qu'elles ont été formulées pour la réalisation du jeu :

"Développer un jeu où un joueur doit jouer à Shifumi contre l'ordinateur."

- Le joueur joue contre une "IA"
- Cette IA donne la possibilité pour le joueur de développer des stratégies
- L'IA gère la progression de la difficulté
- Le joueur doit penser que le jeu est juste
- Fournir une interface attrayante s'appuyant sur une évaluation UX/UI auprès des étudiants du campus
- Architecture permettant de mettre de nouvelles IA

La conception du jeu a pris en compte ces contraintes et elles seront rappelées plus tard dans le document.

II. Conception et Game Design

Définition du jeu

Au début du projet, peu après la réception des consignes, l'équipe est entrée dans une phase de **brainstorming** portant sur le sujet du jeu qui allait être réalisé.

Le concept du jeu a été trouvé et certains éléments ont été définis :

Thème / genre :

- Jeu de stratégie "tour par tour" se déroulant dans un univers Médiéval-Fantastique.

Principales mécaniques de gameplay :

- Choix d'un commandant à affronter parmis 10 ennemis uniques
- Débloquer le prochain commandant après une victoire
- Affronter un commandant joué par l'ordinateur (IA)
- Utiliser des mécaniques inspirées du Shifumi pendant des batailles de 15 courtes manches

Plateformes de déploiement visées :

- Exécutable pour PC Windows
- (plus tard) WebGL pour l'exécution du jeu dans un navigateur web
- (plus tard) Android pour publication sur le Play Store

Ce qui différencie ce projet des autres :

- Originalité globale du jeu
- Ses mécaniques de gameplay basées sur le Shifumi
- Son style visuel unique
- Son gameplay se basant sur de la stratégie et un peu de hasard

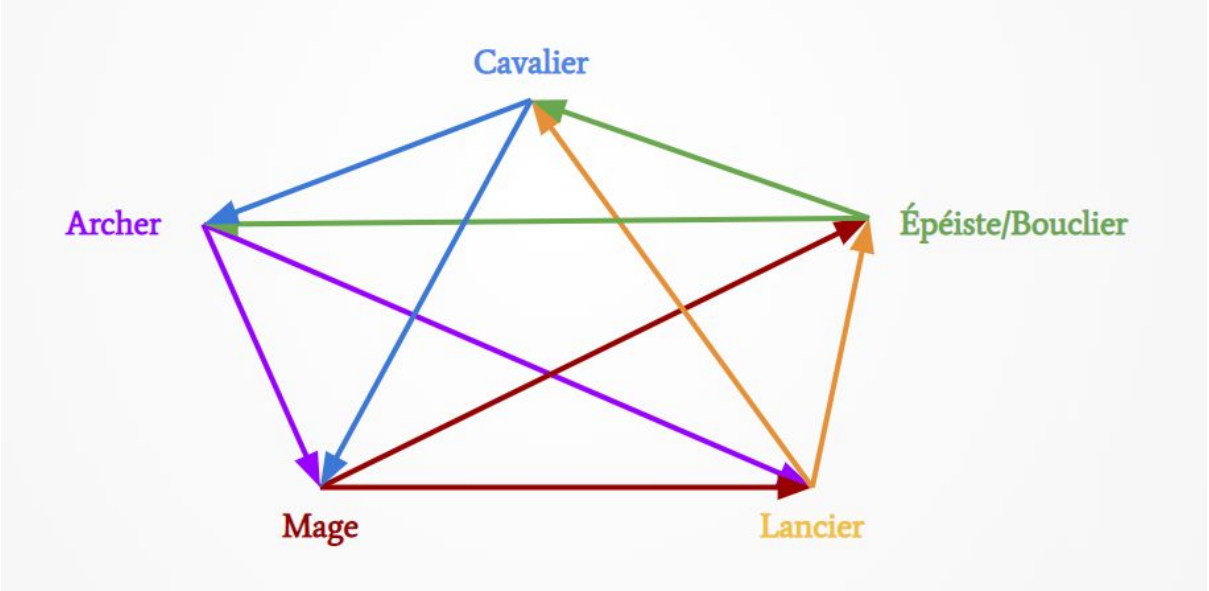
Voir le premier Game Design Document produit.

Implémentation du Shifumi

L'implémentation du système de Shifumi au sein de notre jeu était au centre des contraintes pour sa réalisation.

Afin de diversifier les batailles davantage et pousser la réflexion et la stratégie un peu plus loin, nous avons décidé d'implémenter le Shifumi avec 5 éléments.

De plus ces éléments ont été adaptés à l'univers du jeu ainsi qu'au contexte des batailles, donnant le schéma suivant :

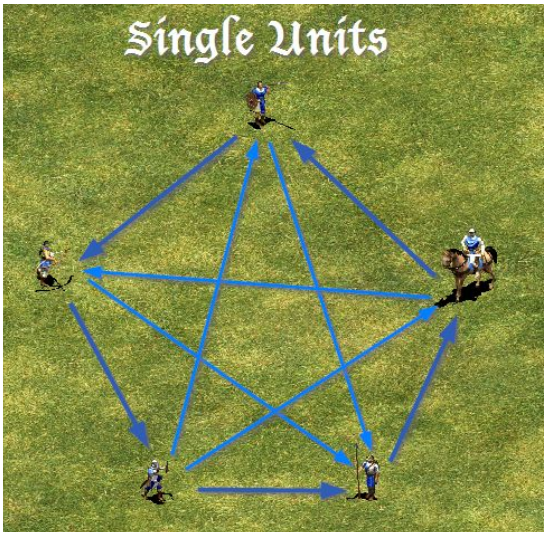


Benchmark et univers

Pour trouver de l'inspiration que ce soit pour le gameplay, l'univers ou les graphismes, nous avons réalisé une **étude sur les jeux et univers similaires à notre projet**.

Voici certains des éléments issus de l'étude de **benchmark** :

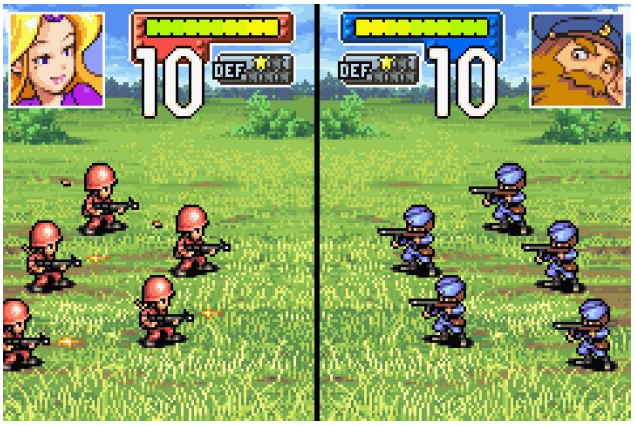
- Age of Empire, jeu de stratégie mythique. On y observe un schéma d'utilisation du système de Shifumi avec 5 types d'unités.



Lors des batailles dans notre jeu, nous utilisons également 5 types d'unités qui interagissent entre elles de manière similaire.

- Les jeux de combat 1vs1 opposant 2 personnages et implémentant un système de manches pour déterminer le vainqueur d'un affrontement :

Mortal Kombat



Advance Wars

Nous nous sommes inspirés de l'affichage avec un écran "séparé en 2" affichant le portrait des personnages et une barre d'équilibre similaire aux barres de vie de ces jeux.

- Pacman, avec les IAs uniques destinées à challenger le joueur :

Nous avons choisi d'implémenter une IA unique par commandant ennemi dans la Ligue, tout comme Pacman implémente une IA par fantôme.

- Duels d'insultes dans Monkey Island pour l'ajout de provocations et réactions de la part de l'IA tout au long de la bataille :

Nous avons implémenté une personnalité unique par commandant ennemi ainsi qu'un système leur permettant de communiquer avec le joueur. En nous inspirant de Monkey Island, nous pensons que cela a permis d'ajouter un caractère aux commandants et de diversifier l'expérience pour chaque bataille différente.

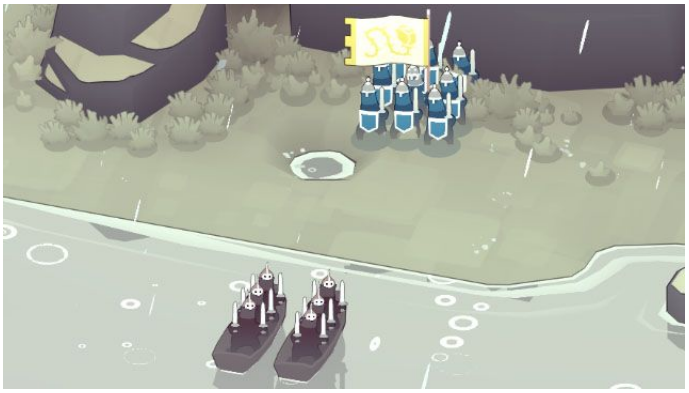
Cela conclut l'étude de benchmark.
Une étude sur l'univers de notre jeu nous a aidé à fixer une identité graphique pour celui-ci.

Voici certains de éléments issus de l'étude sur **l'univers du jeu** :

- TABS : le joueur décide du type de soldat qu'il envoie sur le champs de bataille, la couleur des unités diffère pour différencier les camps:

Nous nous sommes inspirés de la manière dont TABS différencie les soldats de chaque armées et avons décidés d'implémenter un code couleur pour chaque commandant qui est appliqué à leurs unités.

- Bad North, un jeu de stratégie aux graphismes minimalistes :



Nous nous sommes inspirés du style minimaliste des unités de Bad North pour effectuer le design des unités de notre jeu.

- The Witcher pour l'univers médiéval fantastique (présence de mages, ainsi que de toutes les autres types d'unités de notre jeu) :

Cela conclut l'étude de l'univers du jeu.

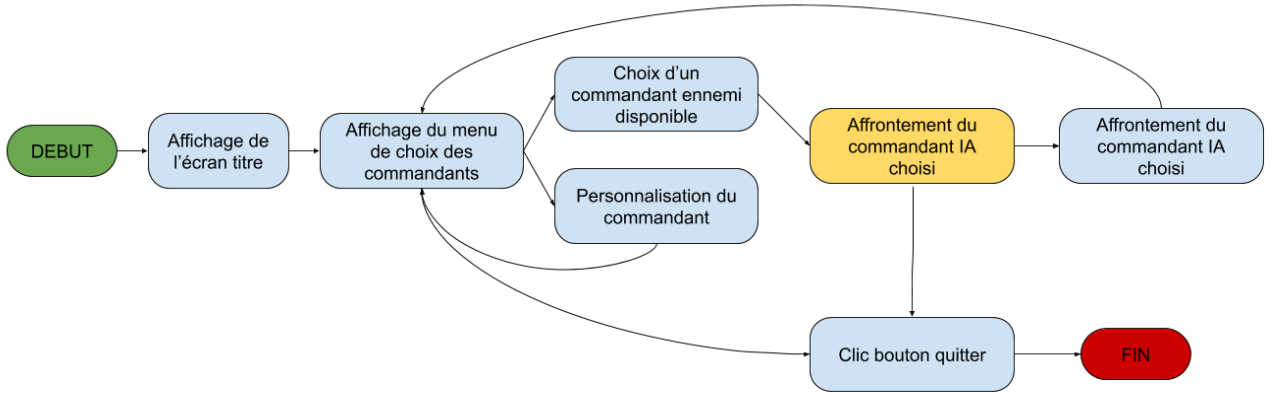
Ainsi, on peut constater que pour la conception de notre jeu nous nous sommes inspirés d'éléments existants similaires et appréciés des joueurs afin de les mélanger pour constituer l'univers de notre jeu et détailler son gameplay ainsi que les éléments qui le composent.

Voir le document de benchmark et univers pour y trouver quelques inspirations supplémentaires.

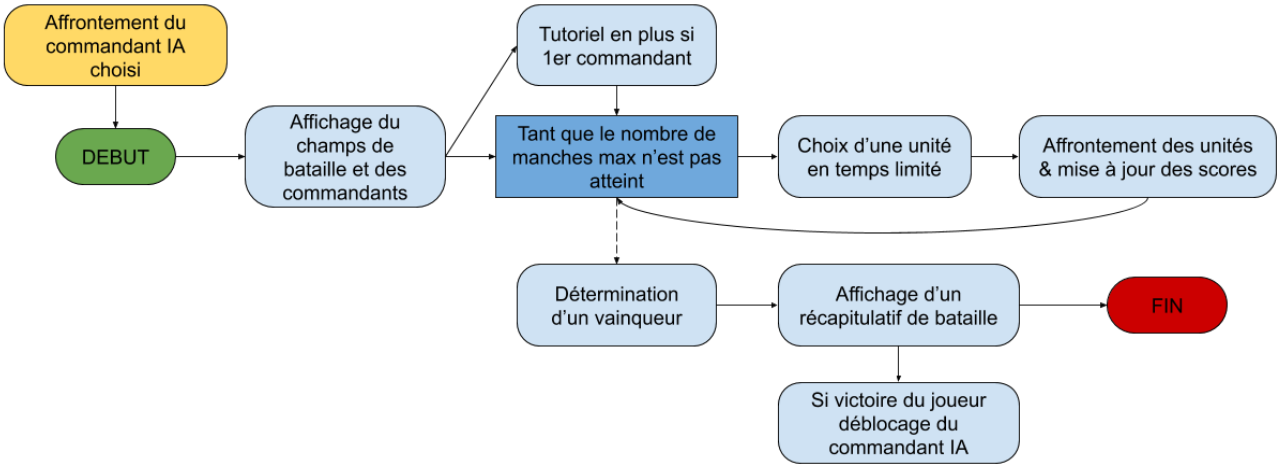
Déroulement d'une partie et d'une bataille

Dans l'optique d'avoir une représentation claire du déroulement d'une partie ou d'un bataille du point de vue du joueur, nous avec réalisé deux schémas comportant des transitions et représentant l'utilisation de notre jeu :

Déroulé global d'une session de jeu



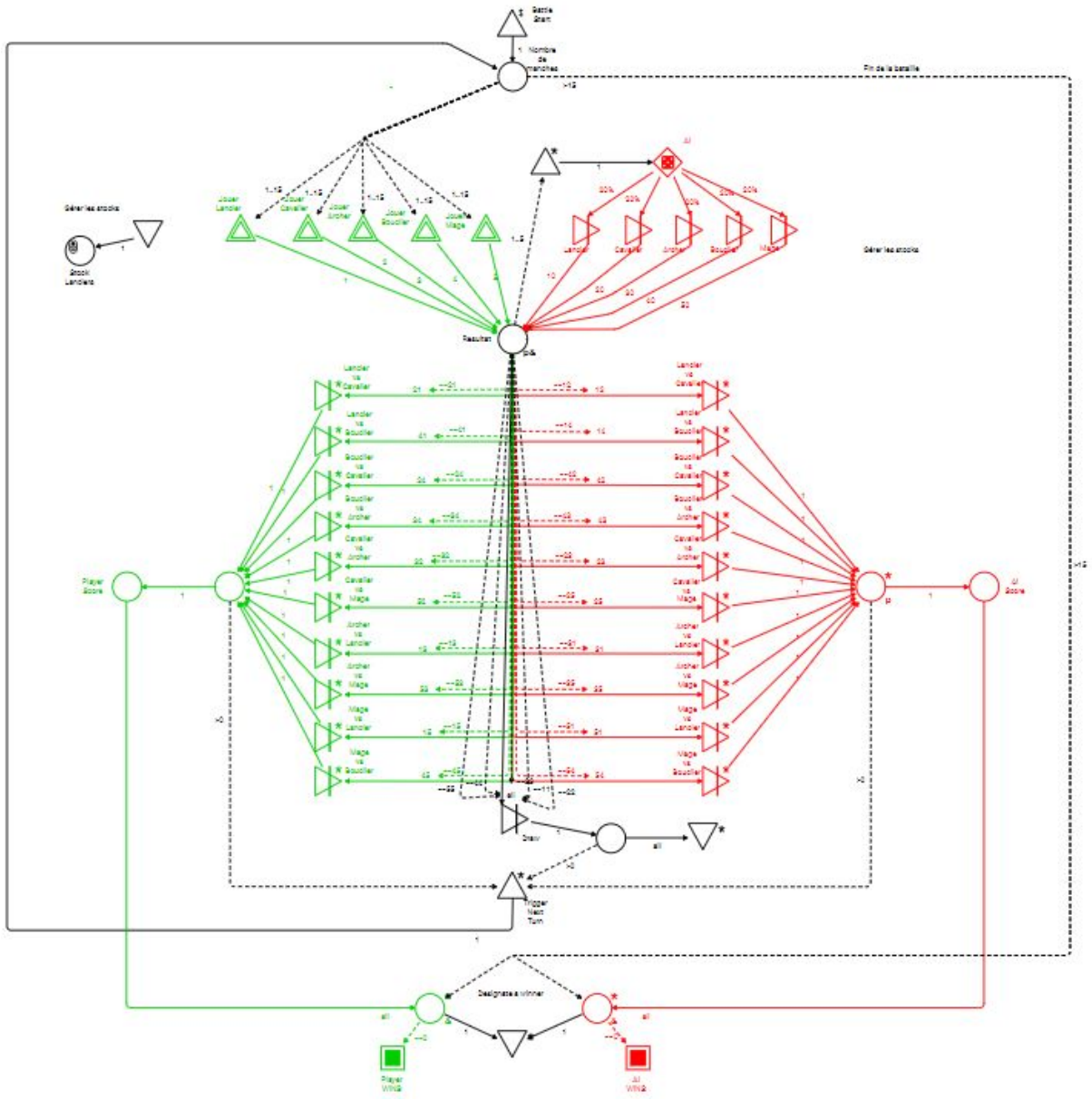
Déroulé d'un affrontement contre un commandant IA



Disposer de ces documents a permis de renforcer notre compréhension du parcours utilisateur pour notre notre jeu. Ils ont également été utiles pour apprécier les tâches à réaliser pour le développement du projet.

Simulation

Dans l'optique d'aller plus loin dans la représentation graphique du fonctionnement de notre jeu, nous avons utilisé le logiciel de Game Design "Machination" afin de simuler une bataille d'un utilisateur contre une IA :



Cette simulation est disponible à cette adresse. Bien qu'elle représente très bien une bataille et est exécutable, des soucis de performances liés au logiciel empêchent malheureusement la simulation d'une bataille entière.

Maquettes graphiques

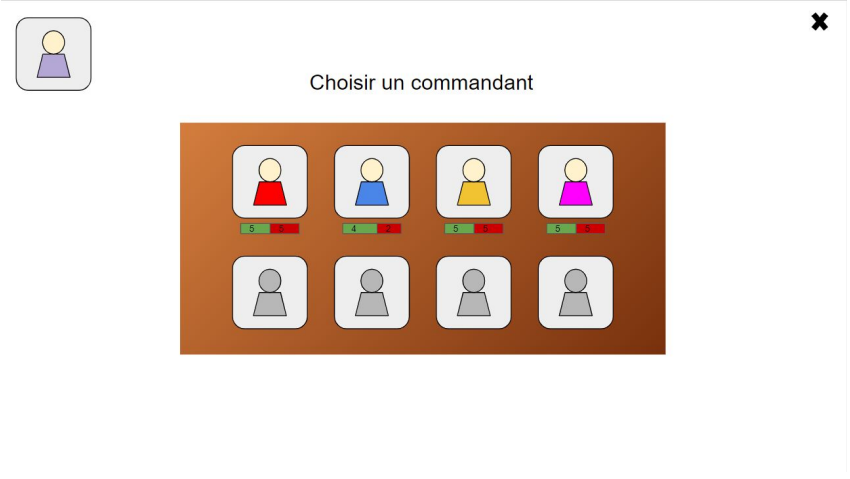
Nous sommes conscients que l'interface utilisateur et les graphismes d'un jeu sont essentiels et doivent être bien étudiés avant d'être implémentés.

Ainsi, nous avons réalisé des mock-ups graphiques représentant chaque écran de notre jeu.

De plus, nous avons ajouté pour chaque maquette des commentaires détaillant chaque élément ainsi que leur utilité (que se passe-t-il si on clique sur tel bouton par exemple ?).

Voici quelques unes des maquettes produites, comparées à leur implémentation plus tard dans Unity :

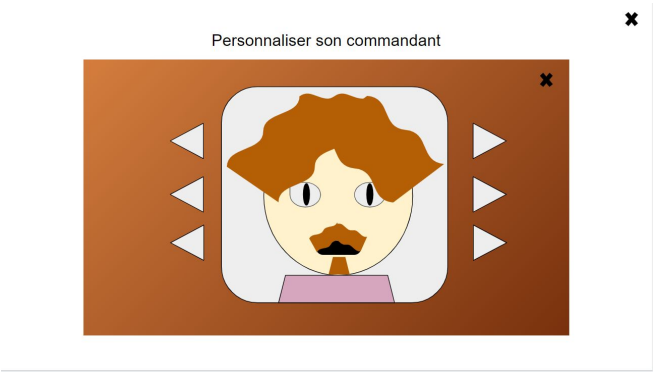
- Écran de choix d'un commandant :



maquette

implémentation

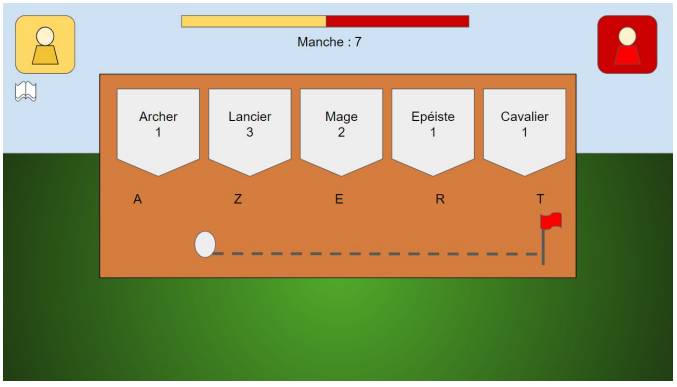
- Écran de personnalisation du commandant du joueur :



maquette

implémentation

- Écran de choix d'une unité pendant une bataille :



maquette

implémentation

L'ensemble des maquettes réalisées avec commentaires est disponible ici.

Description des intelligences artificielles

Cette partie, qui peut aussi intégrée au génie logiciel, aborde la définition des intelligences artificielles du jeu.

En effet, les contraintes du projet imposent plusieurs choses à propos des IA que le joueur affrontera dans le jeu. Voici comment les IA de notre jeu répondent à ces contraintes :

- **Cette IA donne la possibilité pour le joueur de développer des stratégies**
  ⇒ les 10 IA de notre jeu ont leur propre stratégie que le joueur devra identifier et contrer à l'aide de nombreux outils mis à sa disposition.

- **L'IA gère la progression de la difficulté**
  ⇒ les 10 IA représentent dans l'univers le top 10 des meilleurs commandants de la Ligue. Ainsi, elles ont été conçues pour que leur difficulté augmente plus le joueur se rapprochera du top de la Ligue.

- **Le joueur doit penser que le jeu est juste**
  ⇒ les IA du jeu ne basent leur stratégie strictement que sur des éléments disponibles de manière égale pour chaque commandant de la bataille. Cela est même rappelé par le premier commandant qui fait office de tutoriel pour le joueur.

- **Architecture permettant de mettre de nouvelles IA**
  ⇒ Sera abordé lors de la présentation du diagramme de classes du jeu.

Un document décrivant l'ensemble des IA a été réalisé afin de centraliser la manière dont nous allons répondre à ces contraintes. Son but principal cependant est de recenser et décrire les 10 IA du jeu (***SPOILERS***) :

1. **CLOCKWISE AI :**
   Choisit les unités dans l'ordre horaire.

2. **COUNTER CLOCKWISE AI :**
   Choisit les unités dans l'ordre anti-horaire.

3. **DRUNK RESILIENT AI :**
   Choisit une unité aléatoirement parmi celles en stock et la joue jusqu'à épuisement.

4. **DRUNK AI :**
   Choisit une unité aléatoirement parmi celles en stock.

5. **SELF COUNTER AI :**
   Choisit une première unité aléatoire, puis choisit toujours une unité qui contre son unité précédemment jouée (2 choix possibles).

6. **PLAYER COUNTER AI :**
   Choisit toujours une unité qui aurait contrée celle que le joueur a utilisé lors de la manche précédente.

7. **COMMON HUMAN AI :**
   Choisit une première unité aléatoirement.
   Si l'IA gagne ⇒ choisit la même unité à nouveau.
   Si l'IA perd ⇒ choisit une autre unité aléatoire.

8. **PLAYER STOCK AI :**
   Choisit toujours une unité contrant celle dont le joueur dispose du stock maximal. Si plusieurs unités ont un stock correspondant au stock max, choisit aléatoirement parmis celles-ci.

9. **THROWBACK AI :**
   Choisit une unité qui contre le coup du joueur d'il y a 2 manches.

10. **SMART HUMAN AI :**
    Se base sur de vrais études à sur comment gagner au SHIFUMI, 2 règles :
    Si l'IA gagne ⇒ choisit ce que le joueur a joué à la manche précédente.
    Si l'IA perd ⇒ choisit une unité qui n'a pas été jouée à la manche précédente.

III. Gestion de projet

Cette partie va aborder les notions de génie logiciel et de gestion de projet employées pour ce projet. Les documents réalisés à ce sujets y seront présentés et référencés.

Macro planning

Lors de la réalisation de la note de cadrage pour le projet, nous avons également effectué un macro planning for apprécier de manière globale les grandes parties du projet et les estimer temporellement :

Macro-planning et Macro-budget du projet	Phases du projet	Date de début	Date de fin	Charge (en j*h)
	Definition du projet et conception	27/02/2020	02/03/2020	10
	Spécifications	03/03/2020	22/03/2020	70
	Réalisation du jeu vidéo	23/03/2020	07/06/2020	250
	Test et validation du jeu vidéo	08/06/2020	14/06/2020	25
	Livraison du projet	15/06/2020	16/06/2020	10
	Ensemble du projet :	27/02/2020	16/06/2020	365

Une estimation de la charge jour-homme de chaque phase du projet a également été réalisée à partir des dates estimées.

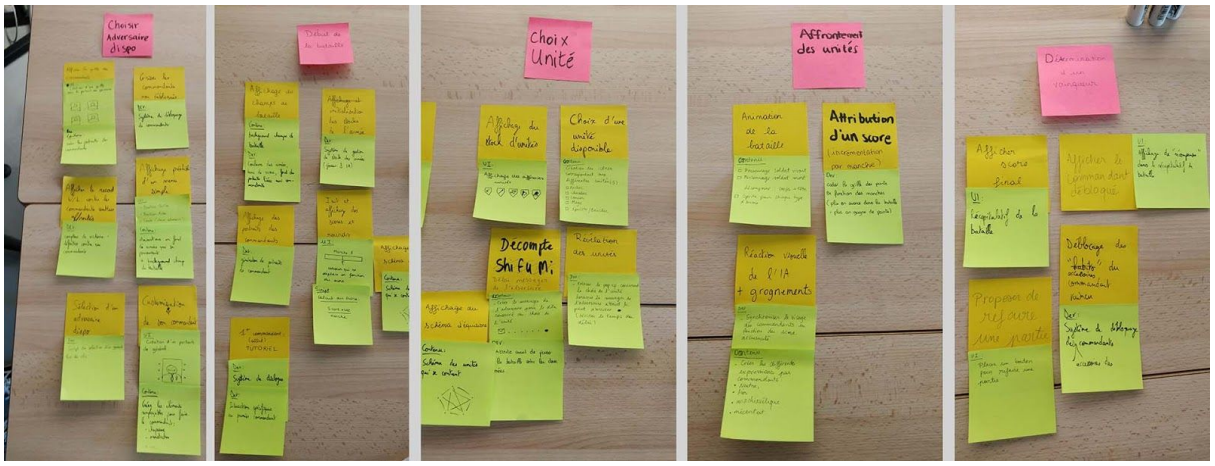
Par exemple, explication pour les spécifications : tâche durant 14 jours de travail, impliquant toute l'équipe (5 personnes) ⇒ 14 * 5 = 70 j-h.

Story mapping

Pour notre projet, nous avons fait le choix de nous orienter vers une gestion de projet agile kanban. Cela se justifie par notre rythme d'alternance qui n'est pas propice à la mise en place d'une méthode telle que Scrum.

A la suite de la définition du jeu présentée dans la partie précédente de ce document, nous avons effectué en classe un exercice de "Story Mapping" afin d'extraire des tâches de notre concept de jeu.

Voici ce qui a été produit en appliquant le Story Mapping pour notre concept de jeu :



L'image est visible en haute résolution à cette adresse.

Nous avons recensé les étapes majeures d'une partie du jeu (post-its roses), puis nous les avons ordonnées chronologiquement et en avons extrait les éléments à réaliser (post-its jaunes) ainsi que les sous-tâches les composant (post-its verts).

Cet exercice a été très intéressant pour nous car nous avons réussi à extraire les éléments nécessaires pour constituer notre backlog de produit et mettre en place notre gestion du projet.

Trello

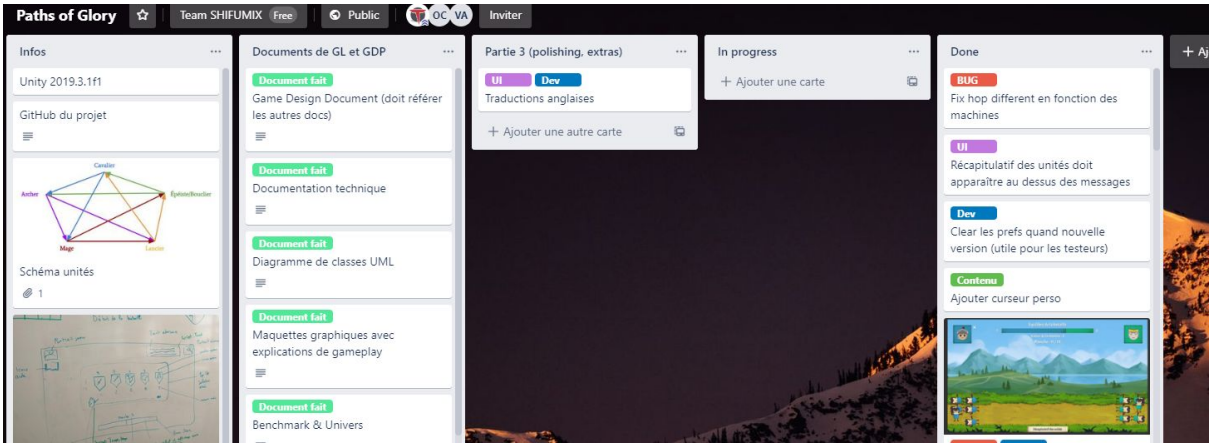
Disposant de notre backlog de produit sous la forme d'un ensemble de tâches à réaliser pour le développement du projet, nous avons choisi d'utiliser un tableau Trello pour effectuer notre gestion de projets.

Trello est un outil que l'équipe a l'habitude d'utiliser et qui convient très bien à notre projet et à la manière dont nous avons prévu d'effectuer sa gestion de projet. De plus, un de nos professeurs nous l'a recommandé et nous a donné quelques instructions utiles pour mieux l'utiliser.

Nous avons organisé les colonnes du tableau de la manière suivante :

- **Infos** ⇒ contient quelques informations et liens essentiels pour le projet
- **Documents GL et GDP** ⇒ référence l'ensemble des documents de génie logiciel et de gestion de projet produits
- 3 parties, avec 1 colonne par partie :
    - **Partie 1** ⇒ les tâches du backlog estimées essentielles pour avoir une base incomplète mais jouable du jeu
    - **Partie 2** ⇒ les tâches qui permettront de compléter le jeu presque entièrement
    - **Partie 3** ⇒ les tâches bonus qui sont programmées pour une date dépassant celle du rendu du projet (pour continuer le projet après la soutenance)
- **In progress** ⇒ partie classique d'un kanban indiquant les tâches en cours
- **Done** ⇒ autre partie classique d'un kanban indiquant les tâches terminées

Voici à quoi ressemble le tableau à la fin du développement :



Lors de la création des tâches, nous avons attribué à chacune un label ainsi qu'une description détaillant la tâche. Pour certaines tâches où cela était applicable, nous avons également ajouté un liste d'éléments à cocher contenant des sous-tâches à effectuer.

Voici l'aperçu d'une tâche du projet sur Trello :

Le tableau Trello de notre projet est disponible publiquement ici.

Répartition du travail

Enfin, nous avons réparti le travail sur ce projet en fonction des profils de chacun des 4 membres de l'équipe.

Voici les rôles principaux de chaque membre sur le projet :

- **Thomas CIANFARANI :**
  Game Design, Lead dev Unity, Gestion de projet

- **Alexandre TOMASIA :**
  Game Design, Dev Unity, Sound Design

- **Océane CARPENTIER :**
  Graphismes 2D, Interfaces utilisateur, Level Design

- **Vanessa ASEJO CASPILLO :**
  Graphismes 2D, Interfaces utilisateur, Level Design

Estimation du temps de travail sur le projet post-réalisation

Le projet ayant débuté le 2 mars 2020 a vu son développement prendre fin le 8 mai 2020. La période de confinement liée au COVID-19 a permis à certains membres de l'équipe de disposer de beaucoup de temps libre pour réaliser ce jeu.

Après la réalisation du projet, nous avons utilisé un outil nommé "git hours" pour estimer le temps de travail sur le projet.

Appliqué sur le dépôt Git de notre projet, la durée estimée est de :

175 heures.

Attention : ce temps concerne uniquement le temps passé sur Unity pour la réalisation du projet. Il ne comporte pas le temps passé à réaliser les assets graphiques, le game design, les documents de projet, etc.

Le résultat est toutefois cohérent par rapport au temps passé sur Unity.

IV. Génie logiciel

Cette partie aborde quelques points technique à propos du projet et présente les documents clés de génie logiciel ayant été produits.

Technologies utilisées

Pour la partie **spécification et conception technique**, nous avons utilisé les technologies suivantes :

- **Google Drive :** drive facile d'accès avec une capacité de stockage suffisante pour organiser les documents et éléments utiles au projet.

- **Google Doc :** utilisé pour la réalisation de documents partagés tels que la documentation technique, permet la modification et création facile depuis n'importe quelle appareil.

- **LucidChart :** logiciel utilisé pour créer le diagramme UML du projet.

Pour la **réalisation technique** de notre jeu, les principales technologies utilisées sont les suivantes :

- **Unity3D :** imposé, version 2019.3.1f1.
  Moteur de jeu de référence pour les indépendants et les petits studios de développement. Certains des membres de l'équipe ont également de l'expérience avec ce logiciel ce qui a permis d'accélérer le développement.

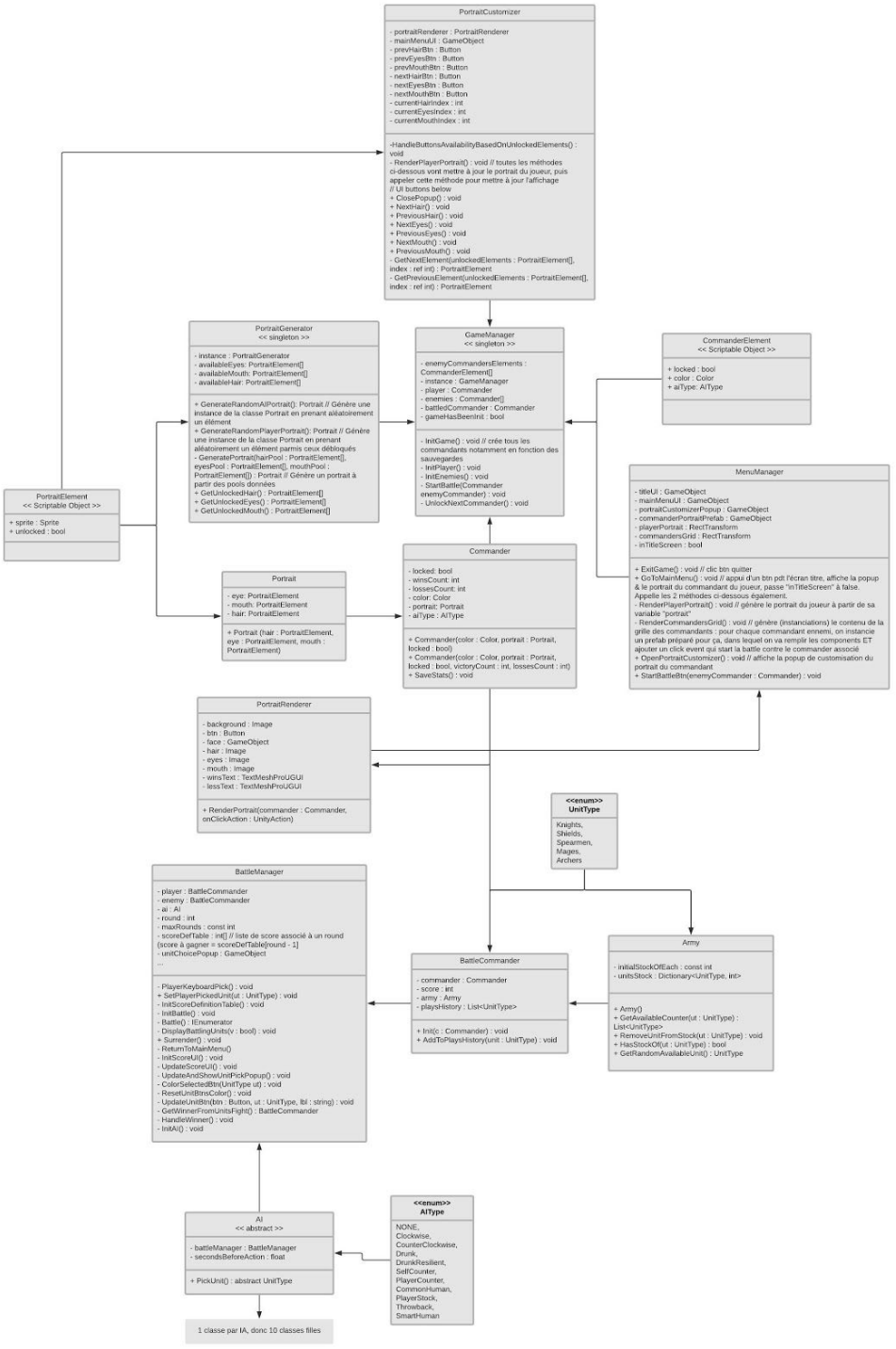
- **C# :** langage de programmation imposé par Unity3D. Une fois de plus, bien connu de l'équipe et assez facile à maîtriser de manière globale.

- **Git :** logiciel de gestion de version, permettant de collaborer à plusieurs sur le projet en partageant code et assets.

- **GitHub :** hébergeur de projets Git utile pour stocker progressivement le code source produit sur le cloud et le rendre accessible à toute l'équipe.

Architecture logicielle

Avant de nous lancer dans la réalisation du projet, en particulier dans la phase de codage, nous avons réalisé un **diagramme de classes UML** afin de mettre en place une architecture logicielle.



Ce diagramme de classes est plus consultable en haute résolution ici.

Bien que ce diagramme ne soit plus entièrement à jour avec la dernière évolution du jeu, il représente toutefois la structure de la **majeure** partie du code présent dans le jeu.

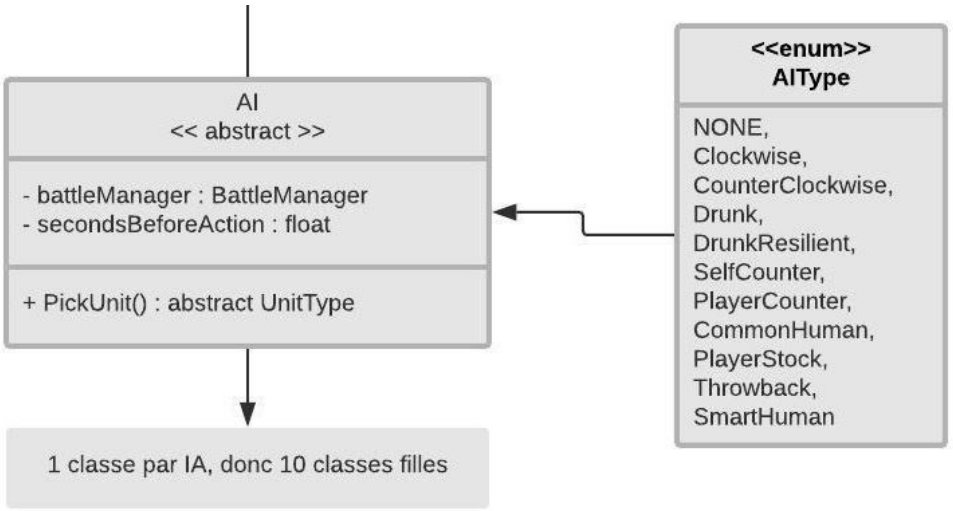
La réalisation de ce document nous a permis de développer de manière organisée et même plus rapide.

Explication technique de l'IA

Les IA dans notre jeu n'ont véritablement qu'**une seule action** à effectuer plusieurs fois pendant chaque bataille : il s'agit du choix de l'unité à envoyer sur le champs de bataille.

La stratégie d'une IA correspond ainsi aux **opérations que celle-ci va effectuer afin de déterminer quelle unité envoyer sur le champs de bataille lors de la manche courante**. Se référer à la partie descriptive des IAs du jeu présentée ci-dessus.

Voici la partie du diagramme UML décrivant les IA du jeu :



Ainsi, nous avons Ici une classe mère abstraite contenant quelques attributs et la méthode **PickUnit()** qui est celle qui sera implémentée par chaque IA spécifique à sa propre manière. Les IA sont également identifiées par une énumération AIType.

Documentation technique

Afin d'encadrer davantage le développement, et imposer un certain nombre de normes de codage à respecter à l'équipe, une documentation technique a été réalisée.

Celle-ci a pour but d'uniformiser le code produit par chaque collaborateur, ainsi que de proposer quelques instructions sur notre usage de la gestion de version avec Git et GitHub.

Voici le **sommaire** de cette documentation technique :

Documentation technique **1**
Normes pour le projet 2
Partie Unity 2
Partie C# 2
Gestion de version 3
Généralités 3
Fonctionnement de notre GitHub 4
Récupérer le projet 7
Mettre en place la résolution de conflits entre scènes automatique 8

Pour ce qui est des **normes pour le projet**, l'idée était de rappeler quelques norme fondamentales concernant le développement avec Unity3D. On y trouve alors le contenu suivant :

Normes pour le projet

Partie Unity

⇒ **Utiliser Unity 2019.3.1f1**

Voici les différentes normes à respecter pour le travail dans l'éditeur Unity:

- Bien placer les ressources dans les dossiers présent dans le dossier Asset
- Bien placer les GameObjects dans les "onglets" pré-créés dans l'inspecteur
- UI : Utiliser les ancres pour que l'UI soit responsive

Partie C#

Voici les différentes normes à respecter pour la programmation C#:

- Noms de méthodes et variables en anglais
- Commentaire de préférence en français mais ok en anglais
- Respecter un minimum les conventions de codage C#, surtout le camel case
- Pour chaque méthode créée, ajouter au dessus un commentaire expliquant ce que la méthode fait (utiliser les commentaires /// <summary> …. </summary>
- Faites attention à ce vous mettez dans les méthodes Update() car elle est appelée bcp de fois par secondes
- Pour ce qui est Physique, utiliser FixedUpdate() plutôt que Update()

Ces quelques instructions simples reprennent des normes officielles pour Unity3D et le langage C#.

Pour ce qui est de la partie **gestion de version**, la documentation technique aborde les points suivants :

- Lien vers le dépôt GitHub
- Syntaxe de numérotation des versions du jeu
- Fonctionnement adopté pour l'usage de Git : le Git-Flow
- Exemple détaillé de l'utilisation de Git pour le travail sur une fonctionnalité
- Comment récupérer le projet depuis GitHub
- Comment résoudre les conflits lors de la mise en commun des travaux

Voir la documentation technique dans son intégralité.

Organisation des tests

L'importance des tests dans tout projet est indiscutable et c'est pourquoi nous avons décidé de passer du temps sur cette partie afin de s'assurer de disposer d'une méthodologie fiable tout au long du développement du jeu.

Ainsin, nous avons rédigé un document d'**organisation des tests et revues de code**.

Ce document aborde les 3 points suivants :

- Les **tests unitaires** :
  Indique qu'il faut mettre en place des tests unitaires sur les fonctions développées lorsque cela est possible. Avec Unity et C#, il est possible de déclencher facilement la batterie de tests unitaires depuis l'éditeur et visualiser les résultats.

- Les **smoke tests** :
  Smoke test = test de "fumée" pour voir si les fonctionnalités principales du jeu sont OK. Idéalement un Google Form, doit être clair pour être utilisé par n'importe qui, et assez rapide pour vérifier brièvement la fonctionnalité du jeu.
  Au début de chaque sprint, créer/mettre à jour un smoke test par rapport aux fonctionnalités qui seront développées dans le sprint.
  A la fin de chaque sprint, effectuer le smoke test pour vérifier que les fonctionnalités produites sont conformes aux spécifications du sprint.

- La **relecture de code** :
  Pour maintenir une bonne qualité de code, s'appuyer sur le système de **pull request** de GitHub afin d'effectuer, **lors de chaque merge de branche dans develop**, une relecture de code avec au moins 1 collaborateur du projet.

Voir le document complet sur l'organisation des tests.
Voir le smoke test réalisé pour le projet et progressivement mis à jour.