L3 Informatique · S6 Printemps 2020

Projet Intégrateur - Groupe OS

› Cahier des charges ‹

Khalida OUAMAR

Antoine PIERQUIN

Louis POLITANSKI

Hugo PORQUET

Victor ROECK

Choaïb ROSTAQI

Aymeric SCHAULI

Joren SCHUSTER (*chef de projet*)

Ce document constitue le cahier des charges du projet du groupe OS, réalisé dans le cadre de l'UE « Projet Intégrateur », suivie au semestre 6 en Licence d'Informatique à l'UFR de Mathématique et d'Informatique de Strasbourg.

Il a pour but de décrire le projet de manière globale ainsi que sa conception, en détaillant ses différents aspects. Ce cahier des charges présente donc dans un premier temps le jeu de plateau qui a été choisi pour être revisité et réalisé en jeu vidéo, ainsi que les objectifs du projet d'adaptation. Les choix technologiques sont ensuite décrits et justifiés, après quoi le déroulement prévisionnel du projet est présenté. L'organisation générale de l'équipe et les méthodes de travail sont également précisées ; enfin, un récapitulatif des différents outils utilisés pour la gestion du travail conclut le document.

SOMMAIRE

1 Jeu choisi et objectifs du projet

1.1 Présentation du jeu original

Le jeu choisi est « Les Colons de Catane » (également appelé « Catane »), créé par Klaus Teuber. Catane met en scène différentes factions arrivant sur l'île inhabitée de Catane, toujours en quête de gloire, les joueurs chercheront donc à être les premiers à coloniser l'île. Pour ce faire, les joueurs vont essayer d'exploiter un maximum de ressources sur l'île pour pouvoir mener à bien leurs différents projets d'aménagement tout en faisant attention à ne pas se faire devancer dans leur conquête par les autres colons.

Il s'agit donc d'un jeu de gestion/stratégie se jouant de trois à quatre joueurs sur un plateau hexagonal représentant l'île de Catane (*cf* ILLUSTRATION). Cette île est composée de six ports ainsi que de dix-neuf cases en forme d'hexagone contenant chacune l'une des ressources primaires du jeu : le bois, le blé, l'argile, la laine et le minerai, sauf l'une d'entre elle qui sera le désert. Ces ressources sont récupérées grâce aux colonies (ou villes) se trouvant à l'intersection de ces cases. La disposition des cases change de partie en partie.

ILLUSTRATION — *Plateau, cartes et pièces du jeu original*

Catane propose aussi un système de commerce, que ce soit entre joueurs, ou avec l'étranger. Le commerce entre joueurs se fait de manière orale, il n'y a pas de limite, tant que les deux joueurs sont d'accord. Le commerce avec l'étranger se fait en échangeant quatre ressources, pour en avoir une autre. Les ports servent à réduire ce nombre, soit trois ressources pour en avoir une au choix du joueur, soit deux pour en avoir une spécifique.

L'objectif du jeu est d'être le premier à arriver à dix points de victoire, pour arriver à ces dix points, les joueurs devront utiliser leurs ressources pour pouvoir construire différents aménagements (qui eux même permettent de récupérer des ressources) et/ou d'acheter des cartes développement qui auront divers effets.

Le déroulement d'une partie est décrit de manière plus détaillée dans un diagramme d'activité, disponible en document annexe (*cf* image `CDC-equipe-OS-Diagramme_activite.png`).

1.2 Choix d'adaptation et enjeux d'implémentation

Les Colons de Catane propose non seulement des éléments d'expérience de jeu très amusantes, mais il offre aussi un nombre conséquent d'éléments d'implémentation qui ont été jugés intéressantes de part leur modularité : deux parties ne sont jamais les mêmes, même si les joueurs prennent les décisions optimales.

Le terrain de jeu est disposé aléatoirement, que ce soit au niveau des cases, ou des numéros associés à celles-ci. L'agencement et la disposition des cases constitue un point de réflexion important, les cases sont aux maximums adjacentes à six autres cases, de plus elles doivent être disposées de manière aléatoire et être associées à des ressources et des nombres. Le plateau doit aussi être agréable à voir et à utiliser pour l'utilisateur, que ce soit pour y installer des aménagements ou pour visualiser le déroulement de la partie.

Les sessions d'échanges (le commerce) sont elles aussi non négligeables, et leur réussite repose

notamment sur leur capacité à ne pas casser le rythme du jeu. De plus, une attention particulière doit être portée sur d'autres éléments qui font de ce jeu une expérience multijoueur agréable : la facilité de création et de connexion à un salon en réseau, la discussion via un *chat* pour les joueurs...

La facilité d'utilisation du jeu et l'amusement en cours de partie ainsi que l'ergonomie des menus doivent être constamment gardées à l'esprit afin que le jeu final puisse assurer une expérience agréable à l'utilisateur.

Tout l'objectif de cette adaptation est donc d'implémenter un jeu vidéo fidèle à la version physique du jeu de plateau, mais qui doit aussi tirer profit des possibilités offertes par l'informatique et s'adapter aux supports numériques pour lesquels l'implémentation sera faite.

1.3 Spécification et fonctionnalités envisagées

Afin de modéliser le déroulement d'une partie des Colons de Catane, un diagramme d'activité du jeu a été réalisé. Ce dernier est disponible en annexe de ce cahier des charges (*cf* image `CDC-equipe-OS-Diagramme_activite.png`).

Par ailleurs, plusieurs fonctionnalités ont été envisagées afin de proposer une expérience de jeu amusante. Il est à noter que seules les plus notables sont récapitulées au sein du tableau ci-dessous : celui-ci n'inclut pas les fonctionnalités de base évidemment indispensables pour que le jeu puisse fonctionner (par exemple : détail des différents menus, possibilité de créer un salon en ligne... ; en jeu : possibilité de faire des constructions, obtention des ressources...).

Un niveau de priorité de 3 correspond à une fonctionnalité très importante et prioritaire, 2, à une fonctionnalité envisageable, alors qu'une fonctionnalité de niveau 1 consiste en une idée émise, éventuellement faisable, mais qui reste à voir.

Les *wireframes* (*cf* partie **1.4 Wireframes des menus et de l'interface**) permettent de compléter la vision globale de ce qui est prévu.

Catégorie	*Fonctionnalité*	*Niveau de priorité*	*Description éventuelle*
Fonctionnement général	Serveur central	3	Serveur permettant d'héberger les parties en ligne, les IA en ligne et la BDD
	Partie en local / solo	3	
	Partie en local / multi	3	
	Partie en ligne / multi	3	
	Jeu disponible sur web ou mobile	2	
	Extensions	1	Partie à 6 joueurs, ...
Menus et interface graphique	Réalisation des différents menus	3	
	Animations, transitions	2	
	Améliorations graphiques	2	
Partie de jeu	Jeu minimal fonctionnel et jouable	3	
	Chat écrit	3	
	Système d'échanges	3	Commerce
	Historique des actions de la partie, dans le chat	2	
	Chat vocal	2	
	Lancer de dés évolué	1	
Options	Réglages sonores	3	Effets sonores, musique
	Langues	3	Traduction en Anglais, Allemand, voire Arabe
	Fenêtrage	3	Plein écran ou non
	Réglages graphiques	1	Luminosité, contraste
	Réglages du contrôleur de jeu	1	Sensibilité de la souris, possibilité de jouer au tout clavier ou avec une manette
Base de données et système de comptes	Création de compte et personnalisation (avatar, pseudo)	3	
	Classement et scores	2	
	Achievements et trophées	1	
Intelligence artificielle	IA minimale	3	
	Autres niveaux de difficulté d'IA	2	

	Prise en charge d'un joueur se déconnectant	2	(Dans le cadre d'une partie en ligne)
	Capacité à gérer les échanges	1	
Divers	Travail sur les musiques et l'ambiance du jeu	1	
	Réalisation d'un tutoriel pour les novices	1	

1.4 *Wireframes* des menus et de l'interface

Un document illustrant les différents menus et l'interface, sous la forme de *wireframes*, se trouve en annexe de ce cahier des charges (*cf* fichier `CDC-equipe-OS-Wireframes.pdf`).

Toutefois, les quelques paragraphes suivants décrivent de manière globale les principaux écrans (dans les menus et en cours de jeu).

Menus

Les menus contenant des informations concernant le jeu (crédits et tutoriel) ainsi que le menu des options sont placés de façon à ce qu'ils soient directement accessibles depuis le menu principal. Ces trois menus sont ceux qui sont le plus susceptibles d'être recherchés par un joueur lors de son premier lancement du jeu, ainsi il n'aura pas à les chercher.
À ces trois boutons menant à ces menus, s'ajoutent un bouton pour jouer en local et un bouton pour jouer en ligne.
Si l'utilisateur souhaite jouer en local, il sera redirigé vers le menu de création de jeu en local. L'utilisateur peut y entrer le nombre de joueur et d'IA ainsi que le pseudo des différents joueurs ; dans le cas où les données entrées sont invalides (moins de 3 ou plus de 4 joueurs) le menu change automatiquement la valeur du nombre d'IA ou de joueurs (selon la valeur qui a été changée en dernier) pour que la composition soit toujours valide.

Si l'utilisateur souhaite jouer en ligne, il sera redirigé vers le menu d'accueil du jeu en ligne. Ce menu se décompose en deux « sous-parties », la première permet de rejoindre ou de créer un serveur, la deuxième permettant de gérer si l'utilisateur est connecté avec un compte ou non. S'il n'est pas connecté, il peut rejoindre et créer des serveurs mais son pseudo lui est assigné ; s'il est connecté, il pourra consulter et modifier son profil ainsi que se déconnecter.
Si l'utilisateur décide de créer un serveur, il est redirigé vers le menu de création d'un serveur. Ici, de manière similaire au menu de création d'un jeu en local, l'utilisateur peut choisir le nombre de joueurs et d'IA. Il doit de plus entrer un nom pour son serveur et un mot de passe si l'option salon privé est activée.
Quand un utilisateur crée ou rejoint un serveur, il est redirigé vers le menu d'attente des joueurs. Une fois que tous les joueurs sont connectés et prêts, l'hôte peut lancer la partie.

Jeu en Local

L'écran de jeu en local est composé :

- Du « plateau », positionné au centre de l'écran
- D'une zone fixe à droite de l'écran, permettant à l'utilisateur d'accéder aux options, règles, et de quitter la partie.
- De trois ou quatre cartes de joueurs : ces cartes affichent les ressources de chaque joueur, leur score et trophées et le nombre de cartes développement qu'ils ont en leur possession. La carte du joueur courant est surélevée de façon à montrer toutes les informations mais les cartes des autres joueurs peuvent aussi être dévoilées en cliquant dessus.
- De cinq boutons, sur la gauche, permettant au joueur dont c'est le tour de construire (une colonie, ville, route), d'acheter une carte de développement ou encore de lancer une session d'échange.

Jeu en Ligne

L'écran de jeu en ligne est très similaire à celui du jeu en local, la zone de texte à droite permet toutefois de dialoguer via un *chat*, et le jeu affiche uniquement la carte de chaque joueur localement :

pour consulter les informations d'un adversaire, il faut cliquer sur son avatar situé au-dessus de la carte.

Échanges / Commerce

Les sessions d'échanges ont lieu dans une fenêtre qui s'affiche en lieu et place du plateau de jeu : chaque utilisateur peut proposer des ressources à l'utilisateur dont c'est le tour, et cet utilisateur peut lui aussi proposer des ressources en échange. Une fois qu'un des utilisateurs et l'utilisateur dont c'est le tour se sont mis d'accord, ils peuvent confirmer les ressources qui vont être échangées. Si les deux parties acceptent, l'échange est réalisé et la session se finit. Si aucune offre ne convient, l'utilisateur ayant lancé la session peut l'interrompre. En « globalisant » les échanges de cette manière, cela évite d'avoir des sessions seulement entre deux joueurs, ce qui ralentit nettement le rythme de la partie et nécessiterait autant de sessions d'échanges que de joueurs souhaitant faire du commerce. De cette manière, l'échange a lieu entre tous les joueurs au même moment, tout le monde peut voir ce qui se passe et communiquer.

1.5 Description de l'architecture globale du jeu

Le jeu est basé sur une architecture MVC, en distinguant trois modules (paquets) différents du côté du client, et deux du côté du serveur.

Côté client

Noyau

Noté "Core", il contient toutes les classes nécessaires pour réaliser le noyau du jeu.
Dans la partie modèle, on modélise la classe d'un Joueur noté "Player". Un joueur est caractérisé par un identifiant unique et un nom, et a un certain nombre de points. Il a une colonie qui est une classe qui regroupe les routes, les villes et les colonies construites par ce joueur. Ce dernier a également un

tas de cartes, où une carte appartient au paquet "Card" pour une meilleure organisation.
On distingue spécialement deux types de cartes, les cartes "Ressource" qui ont un type spécifique (roche, argile, blé, bois ou laine) et les cartes développement qui à leur tour sont divisées en trois sous-catégories.
En outre, il y a un paquet “Mapping” qui représente les différents terrains de la plateforme du jeu. Un terrain est spécialement reconnu grâce à sa position dans le jeu et le type de matière première qu’il produit. Les terrains ne produisent pas tous ces matières premières, c’est pour cela qu’on distingue encore une fois les ports et le désert des autres terrains. Il est à noter que le port se décompose lui-même en plusieurs types, certains ports étant plus avantageux par rapport à d’autres vis-à-vis des échanges.
Du côté du Controller, on retrouve la classe qui modélise une partie de jeu. “Game” est un contrôleur d'entités du modèle “Player”. Il se compose d'une carte de terrains, d’un couple de dés, de la position du brigand, du tour courant et les différentes positions des jetons associées aux terrains existants.
Pour pouvoir utiliser cette classe par d’autres modules, on doit passer par une interface “IGame” dans la vue du paquet Noyau.

UI

Ce paquet regroupe les différents scripts utilisés pour l’interface graphique, et fait appel aux éléments de la vue du Noyau pour l’état logique du jeu.

Réseau

Du côté du client, ce paquet permet de faire une connexion au réseau et de recevoir ainsi qu’envoyer des informations (événements spéciaux). On ne peut instancier qu’un seul objet de la classe “Client” grâce au patron Singleton, qui est utile pour limiter l’utilisateur à une seule connexion par machine.

Côté serveur

Serveur

Ce paquet est très similaire au module réseau dans le client. Cela permet d'instancier un serveur sur un port et une adresse donnée, de lancer et de fermer le serveur mais aussi de diffuser les flux d'informations qu'il reçoit des clients.

Réseau

Dans le paquet Model, on retrouve les classes qui modélisent un client et une partie de jeu. Chaque client a un identifiant, un nom, un identifiant du groupe de jeu auquel il appartient, mais aussi un "TcpClient" qui permet de faire circuler les informations entre le client et le serveur.
Quant à un groupe de jeu, il est composé d'un identifiant, d'un type d'accès (privé ou public – il s'agit du statut du serveur), d'un mot de passe si l'accès est privé ainsi qu'une liste de joueurs (ici considérés comme clients).
On distingue deux contrôleurs dans ce module : un contrôleur de parties de jeu et un autre pour les différents clients à gérer. Grâce à ces contrôleurs, on peut ajouter ou supprimer une partie de jeu du serveur, et faire les différentes vérifications sur chaque nouveau client (s'il existe dans la base de données, s'il est déjà connecté, ...).

Toutes ces classes sont accessibles depuis l'extérieur du module grâce à la vue, avec des interfaces implémentées par les classes du modèle et du contrôleur.

Deux diagrammes UML, récapitulant l'architecture globale côté client et côté serveur, sont disponibles en annexes du cahier des charges (*cf* images `CDC-equipe-OS-UML_client.png` et `CDC-equipe-OS-UML_serveur.png`).

En outre, un troisième document présente un schéma conceptuel et un schéma logique de la base de données (*cf* image `CDC-equipe-OS-Schemas_base_de_donnees.png`).

1.6 Justification de la difficulté d'implémentation

Le jeu choisi étant relativement complet, offrant beaucoup de possibilités, et les fonctionnalités étant nombreuses, les 150 heures de travail minimum requises par personne seront, de l'avis des membres de l'équipe, atteintes.

Le diagramme de Gantt prévisionnel étant maintenant établi, il est possible d'avoir une idée générale de la répartition des tâches et du temps que celles-ci peuvent prendre pour chaque membre.

En effet, l'estimation a donné, au minimum, environ 15 heures de travail personnel chaque semaine, pour chaque personne. Ceci inclut les séances de TPs d'une durée de 4h, les éventuelles autres réunions, le temps de travail consacré au développement, aux tests et à la mise au point, ainsi que le temps nécessaire pour la documentation et la mise en œuvre des différentes fonctionnalités. En outre, les membres qui ont bien avancé et sont disponibles pourront contribuer aux différents tests sur l'application et à l'amélioration du jeu afin d'obtenir une meilleure expérience, par exemple.

2 Choix technologiques du projet

2.1 Technologies

Nous avons choisi d'utiliser le moteur de jeu Unity3D. En effet, Unity présente de nombreux avantages, dont principalement le fait qu'il intègre ensemble toutes les fonctionnalités et librairies nécessaires à la création d'un jeu complet :

- Un moteur de jeu 2D/3D
- Des librairies/outils:
  - de création d'interface graphique
  - de gestion de périphérique (manettes, son, etc.)
- La possibilité de compiler pour différentes plateformes

Le fait de pouvoir compiler un même projet vers différentes plateformes, notamment pour le Web (et mobile) est un gros avantage puisqu'il est entre autres requis pour le projet de réaliser une version du projet accessible soit depuis le Web soit sur mobile, ceci nous permettra donc de gagner en temps lorsqu'il s'agira de passer à la version Web après avoir implémenté la version *desktop*.
De plus, plusieurs membres de notre équipe ayant déjà utilisé Unity auparavant, le temps d'adaptation sera moins grand pour l'équipe.
Unity est également gratuit pour tous les projets non commerciaux et à petits budgets.
Il s'agit donc d'un outil parfaitement adapté à nos besoins pour ce projet.

2.2 Langages

Pour ce qui est du langage, le langage est ici en partie dicté par le framework utilisé puisque Unity impose d'utiliser le C# pour ses scripts.
Le C# est un langage de haut niveau avec une syntaxe proche de celle du Java. Il offre un bon

compromis entre la performance et la facilité d'utilisation, de maintenance. Il dispose en effet d'un système de gestion des ressources (RAII) avec ramasse-miettes qui demande une moindre gestion de la mémoire de la part du programmeur. Il dispose également de nombreuses fonctionnalités intégrées au langage pour la conception d'applications Windows, Linux, Web, mobiles et autres par le biais du framework .NET. Le C# permet également d'interagir avec les bases de données grâce aux requêtes LINQ (Langage-Integrated Query).
Là encore, plusieurs membres de l'équipe ayant déjà programmé en C# et celui-ci ayant une syntaxe proche du Java, ce choix ne devrait pas poser de problèmes d'adoption.
Enfin, bien que le C# soit un langage de relativement haut niveau, sa performance reste largement suffisante pour la création d'un tel jeu, d'autant qu'il y a ici peu ou pas de contraintes de temps réel au niveau du jeu.

2.3 Base de données

Nous utiliserons une base de données SQLite pour stocker les données liées aux comptes des joueurs en ligne. L'interface avec une base de données SQLite en C# peut se faire facilement puisqu'il existe une librairie dédiée dans le framework .NET.

2.4 Périphériques

Les périphériques permettant de jouer au jeu seront, de base, le clavier et la souris.
Si les fonctionnalités du jeu de base sont achevées suffisamment tôt, nous comptons cependant permettre une utilisation basique d'une manette de jeu (déplacement du pointeur de souris avec le stick et clic avec un bouton de la manette) ainsi qu'une utilisation au tout clavier (dans la version de base, notamment en cours de partie, l'utilisation de la souris sera requise).
Bien que cela ne soit pas une priorité, cela devrait être une fonctionnalité relativement aisée à implémenter, puisque Unity permet de gérer assez facilement les manettes les plus populaires.

3 Déroulement prévisionnel

3.1 Diagramme de Gantt prévisionnel du développement

Un diagramme de Gantt prévisionnel a été réalisé et se trouve en annexe du cahier des charges (*cf* image `CDC-equipe-OS-Diagramme_Gantt.png`).

Le fichier à partir duquel a été généré l'image, au format `.gan`, est également laissé à disposition, au besoin (*cf* fichier `CDC-equipe-OS-Diagramme_Gantt.gan`). Il est possible d'ouvrir un tel fichier par l'intermédiaire du logiciel GanttProject.

3.2 Jalons : prototypes/versions et critères

Il est prévu, au cours de la réalisation du projet, de réaliser différents prototypes jouables permettant de valider notre avancement dans le projet.

Le premier prototype jouable sera la version *pre-alpha* : elle sera prête une fois que la partie noyau sera terminée et qu'une interface graphique basique aura été réalisée. Le jeu sera jouable en local, sans module réseau, IA ni BDD : l'intérêt sera simplement de pouvoir vérifier qu'une partie locale fonctionne.

Le deuxième prototype, la version *alpha*, pourra sortir une fois que la partie réseau sera opérationnelle. L'intérêt sera alors de pouvoir tester une partie en ligne.

Le troisième prototype, la version *beta*, sera disponible une fois que la partie IA sera terminée et que la BDD aura été implémentée. Des tests en local et en ligne permettront de s'assurer que les bots sont fonctionnels, et que la BDD est bien faite.

Enfin, un dernier prototype (une sorte de version *pre-release*) regroupera l'ensemble de tout ce qui a été fait, en apportant un support web ou mobile, des améliorations graphiques ainsi qu'une traduction.

La version finale (*release*) sortira rapidement après la *pre-release*.

4 Organisation générale

4.1 Gestion de l'équipe et méthodes de travail

Répartition du travail

Le projet est découpé en trois modules principaux:

- le module noyau
- le module réseau
- le module interface graphique (menus et jeu en lui-même)

Pour chaque module, un responsable a été désigné : il sera au cœur de cette équipe et sera chargé de communiquer avec le chef de projet sur l'état du développement et l'avancement de ce module. Cela évite, dans l'idée, que tous les membres se retrouvent "multi-équipe" : un responsable pour chaque équipe permet d'avoir une personne qui sera, à priori, toujours membre de cette équipe et par conséquent connaîtra bien « son » module.

Le chef de projet sera amené à travailler au sein des différentes équipes, mais il aura cependant moins de tâches de développement. L'objectif est qu'il ait une bonne connaissance de chaque partie et qu'il puisse communiquer avec toutes les équipes en s'assurant de diffuser les informations entre elles.
De cette manière, il pourra à tout moment disposer d'une vision globale du projet, de son organisation et de son avancement, afin qu'il puisse faire les bons choix en conséquence : réorganisation des équipes selon l'avancement, changement dans le planning, décisions pour le développement... Cela lui permettra également d'assurer la maîtrise de l'ensemble des aspects du jeu et de pouvoir rédiger des comptes-rendus précis.

La répartition pour les modules développés ultérieurement (intelligence artificielle, base de données...) reste à définir, et sera établie en fonction de l'avancement des modules « initiaux » et des

membres de chaque équipe.

Enfin, en ce qui concerne les tests, chaque équipe s'occupe de réaliser les tests spécifiques au module concerné, et ce tout au long du développement et de la réalisation du projet.

La répartition initiale des membres pour le développement est la suivante :

Module	***Responsable du module***	***Autre(s) membre(s)***
Noyau	Choaïb	Louis
Réseau	Khalida	Antoine
Interface graphique	Victor	Aymeric, Hugo

Réunions de travail

Les premières réunions ont permis de discuter et de mettre au point de tous les aspects importants du projet à définir au plus tôt, qu'ils concernent le jeu ou l'équipe : choix du jeu, outils et technologies, fonctionnalités, menus et interfaces (*wireframing*), architecture globale du jeu ; organisation, plateformes, répartition des tâches…

Une fois le cahier des charges validé par M. DECOR, les réunions ultérieures seront tout aussi importantes, et permettront notamment de récapituler et *débriefer* les dernières avancées, ce qui est à faire prochainement et plus tard, l'état du planning : en bref, l'avancement du jeu. Ces réunions permettront également de discuter certaines propositions (concernant des fonctionnalités, un aspect du développement, une amélioration...).

Par ailleurs, les équipes seront rééquilibrées au besoin, si certaines équipes prennent du retard ou rencontrent des difficultés, vis-à-vis de l'avancement par rapport au diagramme de Gantt. Dans le cas contraire, ce sera plutôt l'occasion d'envisager l'ajout de fonctionnalités à priorité moins élevée, voire d'extensions. Les décisions de ré-affectation des membres seront prises par le chef de projet, après discussion et accord avec les membres des équipes concernées.

4.2 Rôle et missions du chef de projet

Le chef de projet est au cœur de l'équipe, et ses missions sont nombreuses :

- gestion de l'équipe (organisation, cohérence, répartition) et communication entre tous les membres mais aussi entre le tuteur, M. DECOR, et les membres de l'équipe
- vision globale du projet, de son organisation, de son avancement, suivi des tâches ; afin de faire les bons choix en conséquence : réorganisation des équipes, changement dans le planning, décisions pour le développement
- travail avec les différentes équipes, pour avoir une bonne connaissance de chaque partie et faire le lien entre elles
- prise en compte des remarques des membres de l'équipe (concernant l'organisation, le développement...)
- gestion des éventuels conflits ou difficultés, qu'elles soient techniques ou humaines
- préparation pour chaque réunion d'une "feuille de route" récapitulant où nous en sommes (l'état actuel du projet), ce qui a été fait dernièrement ainsi que ce qui est prévu pour la réunion (et éventuellement ultérieurement)
- réalisation de comptes-rendus détaillés des réunions et d'échanges avec M. DECOR
- réflexions et travail, relativement tôt, sur le rapport final, la documentation technique et le dossier de communication
- gestion des rendus

4.3 Gestion des facteurs humains

La répartition du travail et l'organisation en équipes ont été faites relativement vite : ainsi, dès le début de la troisième semaine du projet, chaque membre connaît déjà la partie sur laquelle il va travailler.

Nous avons tâché de définir une stratégie globale pour la gestion du projet au long du semestre,

notamment en prenant en compte nos obligations au cours des trois mois qui vont suivre : nous sommes conscients que plus nous avancerons dans le semestre, plus nous aurons de choses à faire à côté du projet intégrateur : examens, projets...
La stratégie mise en place consiste donc à travailler intensivement dès le début du semestre, avant une période plus chargée qui débutera vers le milieu du mois de mars.
En parallèle, le chef de projet compte garder à tout moment une vue globale du travail réalisé et de l'avancement, lui permettant notamment d'effectuer des réaffectations de membres entre équipes.

Par ailleurs, nous sommes conscients qu'il y aura forcément des imprévus, des aléas techniques et/ou humains, ou des retards. Ceux-ci seront pris en compte par le chef de projet qui tâchera de les analyser, de les gérer et de prendre, après discussion avec les membres de l'équipe, des décisions en fonction de leur impact sur le planning.
Enfin, nous allons faire en sorte de finir notre projet pour le 21 avril, soit deux semaines avant le rendu. Cela permet de « nous laisser » un peu de marge compte-tenu des éventuels problèmes que nous pourrons être amenés à rencontrer.

4.4 Intégration et mise en commun du code

Lors de la phase de développement, qui débutera une fois la spécification du jeu réalisée, chaque membre pourra déposer son code source sur le GitLab créé au fur et à mesure de son écriture. Étant donné que l'architecture utilisée est basée sur le modèle *MVC*, chaque membre ayant besoin d'un module spécifique pourra faire appel aux interfaces dédiées pour cela.
L'intégration se fera donc tout au long du développement : il a été jugé important d'"intégrer" les différents modules entre eux au fur et à mesure, afin de ne pas commencer à vouloir faire communiquer les modules entre eux à quelques semaines de la fin.

5 Outils pour la gestion du travail

5.1 GitLab Unistra

L'outil de versionnage GitLab de l'Université de Strasbourg sert de dépôt pour les fichiers de code source écrits par les membres du groupe. Cet outil permet le partage, et le versionnage de ces fichiers. Cela permet notamment de garder une trace de toute modification, et de travailler sur des branches différentes du projet, voire sur des versions antérieures.
Le système d'*issues* permet également d'assurer un bon suivi des *bugs*, éléments non fonctionnels et erreurs. Une attention importante est dédiée à la gestion du GitLab et à son organisation, et les modifications, améliorations et changements sont discutés par l'ensemble de l'équipe.
Par ailleurs, l'existence de versions du projet sauvegardées localement par chacun des huit membres permet d'apporter de la robustesse pour la gestion du projet, notamment en cas d'indisponibilité du service GitLab de l'Unistra.

5.2 Discord

Le groupe communique principalement via un serveur Discord, application permettant d'échanger par écrit, les annonces, réflexions, commentaires et propositions des membres, ce qui permet à l'ensemble du groupe de s'organiser. Les réunions de groupes, quand elles sont programmées à distance, se font notamment via l'outil de discussion vocale de groupe.

Ce serveur est aussi accessible à l'enseignant en charge de notre groupe de projet intégrateur, M. DECOR, et sert donc de plus d'outil d'échange entre les membres du groupe et le tuteur.

En cas d'indisponibilité temporaire des services Discord, il reste possible de communiquer entre nous, notamment via la messagerie Partage de l'Université.

5.3 Trello

Trello est un outil de gestion de projet en ligne, permettant de définir des tâches à accomplir, et d'y affecter des personnes pour un temps déterminé. Cet outil sert notamment pour la coordination de l'équipe, et donne au chef de projet comme au reste du groupe une vision d'ensemble de l'avancement du projet, des tâches à accomplir et des membres qui sont affectés à chacune d'entre elles. En fonction de la priorité et de l'évolution du projet, Trello permet également de ré-affecter rapidement des membres d'une équipe à une autre et de changer l'organisation simplement.

5.4 Outil de rédaction de documents en commun

Lors des phases ou la rédaction de documents en commun est nécessaire ou simplement plus pratique, nous utilisons *Google Docs,* un logiciel de bureautique accessible en ligne. Il permet aux membres du groupe de travailler simultanément sur des fichiers texte avec mise en forme, permettant notamment de générer des rapports, ainsi que divers documents relatifs au projet tels que des spécifications, ou le guide utilisateur.

5.5 Outil de suivi du travail

Un dernier outil nous permet d'assurer le suivi du travail par chacun des membres, et de mettre à jour au fur et à mesure de l'avancement du projet qui a fait quoi, à quel moment, en spécifiant en outre le numéro des commits GitLab associés, s'il s'agit de code.
Il s'agit d'un *Google Docs* spécifique, mis en commun, qui nous permet une mise à jour régulière et pratique de son contenu, assurant un suivi précis du travail de chacun.