Projet T3.
Graphe PERT et diagramme de Gantt

Sommaire

Introduction

Contexte

Le projet T3 consiste à développer un *serious game* en équipe de 3 ou de 4, en utilisant un moteur de jeu choisi par l'équipe, sur un des quatre sujets proposés. Nous avons opté pour un jeu ayant pour vocation de sensibiliser le public au rôle du directeur de l'IUT dans l'entretien et la rénovation des bâtiments et des infrastructures du campus.

L'application s'appellera RenovIUT. En 2D, elle sera basée sur un plan réel du campus d'Illkirch, vue du dessus. Le joueur incarnera le directeur et devra gérer au mieux les ressources dont il dispose, tout en tenant compte de celles qu'il ne contrôle pas.

Objectifs

Le projet a été annoncé le 3 septembre 2024, à la rentrée universitaire. L'échéance impérative est le 20 décembre 2024, où les jeux développés par les équipes de 2e année seront présentés aux étudiants de 1ère année, ainsi qu'aux enseignants de l'IUT.

Enjeux

En tant que jeu sérieux, RenovIUT ne se limite pas à divertir le joueur, mais a également des objectifs pédagogiques, à savoir :

- Comprendre le rôle du directeur de l'IUT, dont l'administration est relativement autonome, dans l'entretien et la rénovation de ses locaux, en adéquation avec ses engagements sociaux ;
- Découvrir les bonnes pratiques de gestion du budget d'un IUT, alimenté par des sources de financement certes multiples mais limitées, afin d'assurer les meilleures conditions de travail possibles pour les étudiants et le corps enseignant ;
- Expérimenter la prise de décision en situation d'urgence, qui nécessite une remise en question des priorités de l'établissement.

Le risque principal, avec n'importe quel jeu sérieux, c'est que le joueur décroche. Pour l'éviter, nous avons imaginé RenovIUT comme un jeu très rapide, mais prenant. Il va de soi que ce n'est pas un objectif facile à atteindre ; la planification devrait simplifier la dimension organisationnelle de ce projet pour qu'on puisse se concentrer sur le développement.

Livrables

Les livrables suivants sont attendus dans le cadre de ce projet, dans l'ordre chronologique :

- Le poster pédagogique, expliquant la problématique à la base du jeu ;
- Le code source, l'exécutable accompagné d'un tuto et le cahier des charges sur un dépôt Git accessible au grand public ;
- La grille d'évaluation, à l'attention du public testeur.

Graphe PERT

Tâches

Afin de planifier et de coordonner au mieux notre projet T3, il nous a été conseillé d'établir un graphe PERT (*Program Evaluation and Review Technique*), méthode graphique permettant de visualiser un agencement optimisé des tâches constituant le projet, compte tenu de leur durée et des dépendances entre elles.

L'avantage principal du PERT est de faire ressortir la durée maximale du projet, obtenue si on réalise en parallèle toutes les tâches qui peuvent l'être. Aussi peut-on se concentrer sur les tâches cruciales, celles qui prennent le plus de temps et qui doivent impérativement être accomplies à temps, sous peine de compromettre le délai final.

La base de tout graphe PERT est une liste des tâches, accompagnées chacune de sa durée, de son délai et, surtout, de la ou des tâches antérieures (i.e. celles dont la réalisation est indispensable pour poursuivre).

Nous avons pu identifier 18 tâches de durée inégale, allant de 4 heures à trois semaines de travail, comme suit :

N°	***Libellé de la tâche***	***Antériorité***	***Durée (semaines)***	***Délais***
T1	Choix entre GameLab et un jeu sérieux	/	1	13/09/24
T2	Etude de l'ancien cahier des charges	T1	0,25	13/09/24
T3	Création du poster	T1	1	19/10/24
T4	Elaboration d'un schéma du gameplay	T2	1	13/09/24
T5	Choix du moteur de jeu (Godot)	T2	0,1	13/09/24
T6	Elaboration du schéma détaillé du jeu	T4	1,4	20/09/24
T7	Formation à Godot	T5	2	13/10/24
T8	Elaboration d'un diagramme UML	T6	2	13/10/24
T9	Elaboration du cahier des charges	T7	1	20/10/24
T10	Elaboration d'une grille d'évaluation	T1	0,1	20/10/24
T11	Développement des modèles (MVC)	T8, T9	2	04/11/24
T12	Développement des contrôleurs (MVC)	T11	3	25/11/24
T13	Développement des vues (MVC) : UI	T12	3	13/12/24
T14	Développement des vues (MVC) : UX	T11	1	13/12/24
T15	Tests expérience utilisateur	T13, T14	1	19/12/24
T16	Tests élémentaires du code	T15	1	19/12/24
T17	Compilation et publication	T16	0,2	20/12/24
T18	Présentation du jeu	T3, T10, T17	0,1	20/12/24

Fig. 1. Liste des tâches

Une question se posait sur l'unité de mesure du temps. Notre premier réflexe a été de compter en heures, afin d'être le plus précis possible. Cependant, il était difficile de chiffrer la durée

totale en heures, car elle nous avait été donnée en semaines : 16 semaines entre le 3 septembre, où le projet nous avait été annoncé à la rentrée, et le 20 décembre, jour J où tous les jeux créés par notre promotion seraient présentés aux étudiants de première années et à nos enseignants ; 15 semaines si on ne compte pas la semaine de vacances. Dans tous les schémas ci-dessous, les durées sont calculées sur 16 semaines.

Par contre, on aurait pu également compter en jours – on verra par ailleurs plus tard, avec le diagramme Gantt, que c'est la mesure proposée par défaut par plusieurs progiciels dédiés. Compter plutôt en semaines permet de ne pas se préoccuper du week-end et des jours fériés, tout en laissant la possibilité à la personne d'utiliser ces « heures sup » si nécessaire.

Pour résumer, ce choix de l'échelle nous permet de respecter la contrainte du temps tout en restant flexibles : en effet, le nombre d'heures par semaine n'étant pas précisé, on a une certaine liberté de manœuvre.

Niveaux

Les tâches, basées sur leurs antécédents, se répartissent donc sur 12 niveaux, formant des « méridiens » sur la liste. Le travail préalable sur les antériorités et les niveaux facilite la réalisation du graphe :

Niveau	***Tâches***
1	T1
2	T2, T3, T10
3	T4, T5
4	T6, T7
5	T8, T9
6	T11
7	T12, T14
8	T13
9	T15
10	T16
11	T17
12	T18

Fig. 2. Niveaux

On remarque que les niveaux sont relativement nombreux par rapport au total des tâches. Cela veut dire qu'on ne pourra réaliser que peu de tâches en parallèle, mais il était difficile de pressentir, à ce stade, si même une parallélisation partielle apporterait un gain de temps. Reportons-nous au graphe PERT pour en avoir une idée plus claire.

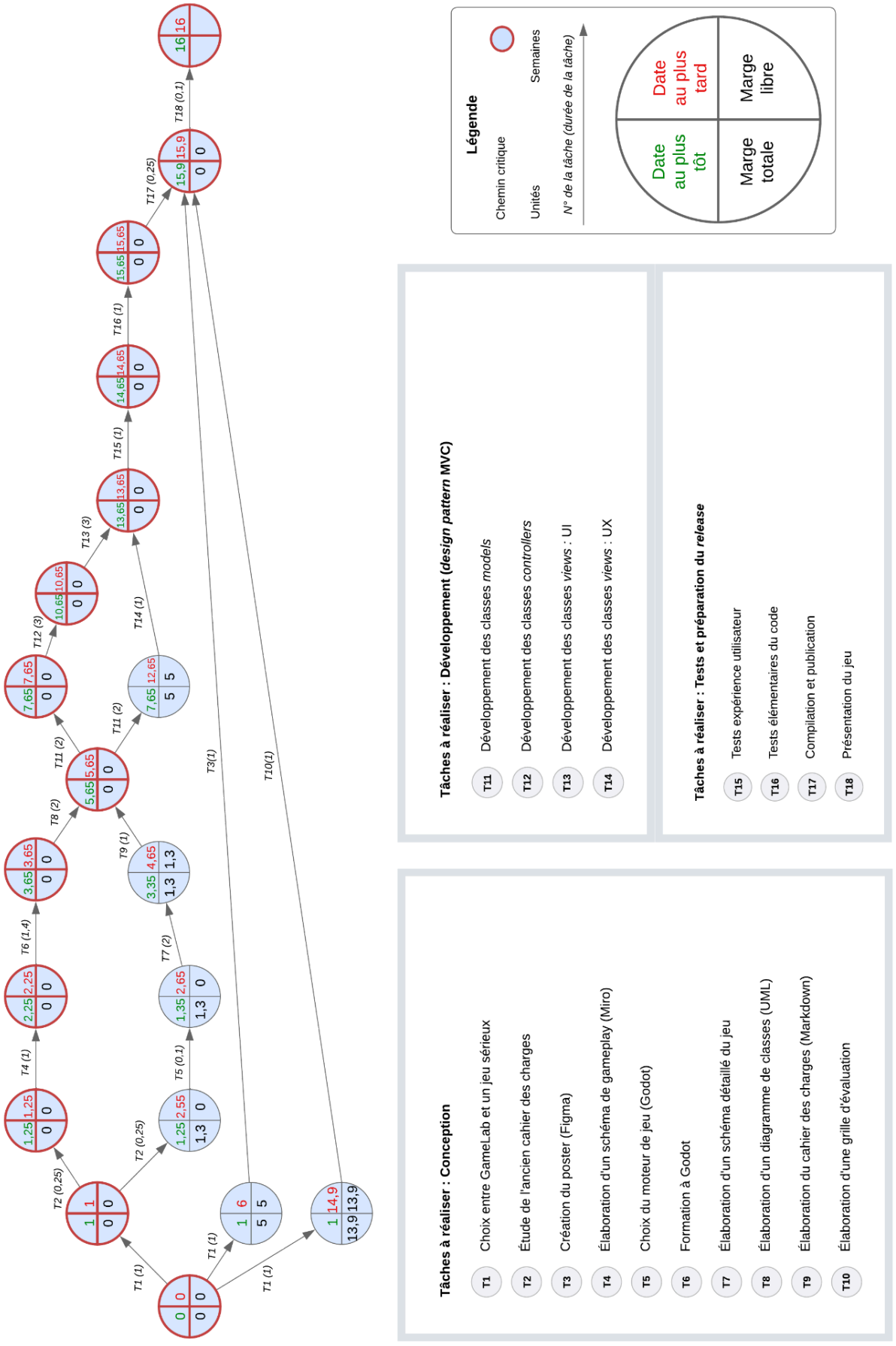

Fig. 3. Graphe PERT du projet

Chemin critique

Le graphe PERT est donc une traduction graphique des tableaux précités. L'agencement de l'ensemble des tâches à remplir pour mener le projet à bon port est montré par des flèches reliant des camemberts divisés en quatre zones, dénotant la date de début au plus tôt (en vert), la date de début au plus tard (en rouge), la marge totale (en bas à gauche) et la marge libre (en bas à droite).

On constate que le ***chemin critique***, c'est à dire le délai incompressible pour réaliser notre projet T3 passe par les tâches suivantes (entourées de rouge foncé sur le graphe) :

1. Tâche 1 – Choix entre le développement d'un projet GameLab et d'un jeu sérieux ;
2. Tâche 2 – Étude du cahier des charges fourni ;
3. Tâche 4 – Création d'un gameplay ;
4. Tâche 6 – Formation au moteur de jeu choisi ;
5. Tâche 8 – Élaboration d'un diagramme de classes ;
6. Tâche 11 – Développement des classes *models* ;
7. Tâche 12 – Développement des classes *controllers* ;
8. Tâche 13 – Développement des classes *views* relevant de l'interface utilisateur ;
9. Tâche 15 – Tests expérience utilisateur ;
10. Tâche 16 – Tests élémentaires du code ;
11. Tâche 17 – Compilation et publication ;
12. Taâche 18 – Présentation du jeu aux étudiants et aux enseignants de l'IUT.

Par définition, ce sont les tâches les plus chronophages, dont la marge libre et la marge totale sont nulles : en effet, on ne saurait en retarder aucune sans repousser la réalisation de la tâche suivante, ni compromettre la réalisation du projet en entier dans les délais. Sans surprise, les tâches de développement en font partie. Bien que – comme on le verra plus tard sur le diagramme de Gantt – le développement puisse et doive être mené en parallèle, réduisant ainsi, proportionnellement, le temps de cette étape du projet, ce ne serait pas judicieux de mélanger les phases différentes de développement dans le cadre d'une programmation orientée objet (POO) selon le patron MVC (*Model – View – Controller)*. Ainsi les « modèles », par exemple, doivent nécessairement être implémentés avant les « contrôleurs », et on attend d'avoir l'implémentation complète de l'algorithme principal que les contrôleurs formalisent pour développer les « vues ».

Les tâches qui présentent une marge, c'est-à-dire qui admettent un retard, sont :

- Tâche 3 – Création du poster ;
- Tâche 5 – Choix du moteur du jeu ;
- Tâche 7 – Élaboration d'un schéma détaillé du jeu, sous une forme libre ;
- Tâche 9 – Élaboration du cahier des charges ;
- Tâche 10 – Élaboration d'une grille d'évaluation, à l'attention des personnes qui testeront le jeu au cours de sa présentation ;
- Tâche 14 – Développement des classes *views* relevant de l'expérience utilisateur.

Il est à noter que les résultats des tâches 3 et 10 ne seront exploitables que par la tâche 18, la dernière. Toutefois, leurs marges diffèrent, car un délai de rendu bien antérieur à la phase finale du projet est imposé à la tâche 3 (le poster devant être préparé pour l'impression avant le 20/10, alors que la présentation du jeu, où il sera accroché au mur, est prévue le 20/12), alors qu'aucune date butoir n'est fixée pour la tâche 10 - élaboration d'une grille

d'évaluation. De ce chef, les marges respectives de ces deux tâches diffèrent de manière considérable.

En étudiant notre graphe PERT, on se rend compte que les optimisations que cette approche apporte ne peuvent être exploitées qu'au début du projet. Ainsi, s'il est judicieux et même utile de se former à Godot, choisi tout en s'efforçant d'imaginer la conception globale du jeu, ou encore dessiner le diagramme UML, qui montre les relations entre les différents éléments du jeu, en rédigeant le cahier des charges en parallèle, la parallélisation devient impossible à partir du moment où il faut coder les fonctionnalités qui se greffent les unes sur les autres !

En revanche, comme on le verra sur le diagramme de Gantt, on arrive à travailler à plusieurs sur une même phase de développement si on change de granularité (à travers les *sprints* Scrum, pour ne citer que cet exemple). Ce n'est donc pas deux phases de développement qui se font en parallèle, mais l'implémentation d'une même fonctionnalité complexe, en toute transparence. Ce résultat intermédiaire sera évalué par la suite, ce qui nous permettra d'adapter notre démarche ultérieure en conséquence, en repartant sur un nouveau cycle dans un esprit d'agilité.

Quantitativement, le chemin critique est de 16 semaines, ce qui correspond tout juste au délai qui nous est imposé, alors que la durée totale des tâches est de 21,5 semaines. Le gain est évident : malgré les limites de notre organisation, le PERT permet d'économiser pas loin d'un mois et demi de travail !

Diagramme de Gantt

Exigences

Identifier les **exigences** pour un projet de développement est une étape essentielle de la planification. On peut les classer en plusieurs catégories. Dans le cadre de notre projet T3, les exigences imposées par le client / tuteur, complétées par celles formulées par l'équipe elle-même sont :

- ***Fonctionnelles :*** il s'agit d'un jeu sérieux dont la visée pédagogique est de sensibiliser les étudiants aux fonctions d'un directeur de l'IUT, notamment dans le domaine de l'entretien des bâtiments et des infrastructures. Le jeu doit être rapide, d'une durée de quelques minutes (environ une minute par année), et donc très prenant. Il est souhaitable que le joueur puisse choisir entre plusieurs scénarios et/ou niveaux de difficulté. Le jeu se fait en solo contre la montre. La logique du jeu doit devenir transparente pour le joueur au fur et à mesure, à des fins d'apprentissage.
- ***Non-fonctionnelles :*** le jeu doit être facile à prendre en main, avec une interface graphique simple, avec un minimum de zones d'affichage étant donné sa durée. Il importe d'assurer que son installation et son lancement ne présentent aucun problème ni ralentissement. Il importe de vérifier la portabilité du jeu Windows / Linux.
- ***Techniques :*** le développement est réalisé avec le système de versionnement Git, codé en Godot Script suivant le choix de l'équipe. Un cahier des charges sera bien entendu établi. À la fin du développement, le dépôt Git doit être accessible à tous les étudiants venus le tester le jour de la présentation (ouvert au grand public ou, au moins, aux étudiants et enseignants enregistrés sur le serveur GitLab de l'Université de Strasbourg). Il sera précédé d'un court tutoriel graphique et accompagné d'une grille d'évaluation, à remplir par les personnes concernées. Un poster sera accroché à côté du poste, présentant la problématique de manière synthétique mais illustrée au mieux par des éléments graphiques.
- ***De design :*** en absence de contraintes de la part du client, le choix revenait à l'équipe. Nous optons pour un jeu 2D, vue de dessus, avec des designs réalistes inspirés par la topologie réelle du campus d'Illkirch. Une charte graphique a été établie pour le poster, qui sera également respectée pour l'interface du jeu (tuto, carte, écrans, notifications pop-up).
- ***De test :*** une fois la phase de développement et de débogage de routine complétée, nous procéderons nous-mêmes aux tests, tant sur le plan de l'interface que sur celui de l'expérience utilisateur. Pour minimiser le risque de biaiser leurs résultats, nous envisageons également de faire essayer le jeu à des personnes de notre connaissance, en dehors de l'IUT, si possible étrangères au domaine informatique.
- ***De calendrier :*** le projet a été lancé le 3 septembre 2024. L'équipe se réunit de manière encadrée tous les vendredis après-midi pendant 4h, et sinon à toute heure à la convenance de l'ensemble des membres. Le jeu fonctionnel sera présenté au grand public le 20 décembre 2024, dans l'après-midi, dûment préparé et accompagné d'un poster de formation accroché au mur. Ce poster doit être soumis à l'impression pour le 19/10/2024 au plus tard, le dépôt, le cahier des charges définitif et la grille d'évaluation rendus accessibles au grand public au plus tard dans la matinée du 20/12/2024, avec toutes les explications nécessaires.

À partir du moment où ces exigences ont été identifiées, un diagramme de Gantt peut être élaboré pour visualiser et planifier les tâches qui permettront de les atteindre. Les exigences sont des éléments de l'objectif visé ; le diagramme, une feuille de route.

Fig. 4. Diagramme de Gantt du projet

Ce diagramme utilise le code couleur suivant :

- les tâches liées à la conception sont mises en bleu,
- les tâches de développement, en vert,
- les tâches de test, en orange,
- les tâches relatives à la présentation du projet, en rouge.

La frise chronologique est établie en semaines, les dates étant marquées tout en haut. Dans un souci de concision, nous n'avons pas mis la liste des tâches en parallèle, mais simplement identifié chaque tâche sur le diagramme avec son numéro. Par contre, nous avons bien tenu à indiquer, pour chaque tâche, les personnes qui en étaient chargées.

On voit que la réalisation du projet, **programmée du 3/09 au 20/12,** reste parfaitement réaliste. Pour l'instant, nous sommes **légèrement en retard sur la phase de développement** (commencée le 21/10 et non pas le 13/10 comme prévu), ce qui pourrait compromettre la qualité du livrable, le jour du rendu qui n'est pas susceptible d'être repoussé, mais c'est également ces tâches critiques, les plus longues et impliquant toute l'équipe, qui peuvent être réalisées plus rapidement au besoin. Au 25/10, le projet est **accompli à 31%**.

Les losanges gris sur le diagramme Gantt sont les jalons.

Jalons

Tout projet est ponctué de **jalons** (*milestones,* dans les ouvrages anglo-saxons), indicateurs de progression qui n'ont pas de durée mais permettent d'identifier une fin de phase, une avancée importante, une validation qui donne le feu vert à la suite.

Sur Git, les jalons peuvent être matérialisés par des « étiquettes », références pointant vers les *commits* qui contiennent des versions ou des *releases,* ou encore une fonctionnalité clé. Sur un diagramme de Gantt, un jalon est représenté par une forme distincte, par exemple, un losange sur la Fig.4 ci-dessus, pour le distinguer des rectangles correspondant aux tâches.

Nous avons pu identifier cinq jalons dans notre projet, dont deux seulement ont été atteint au jour de l'élaboration de ce dossier :

- ☑ ~~Choix du moteur de jeu (savoir en quel langage on va programmer) ;~~
- ☑ ~~Validation du diagramme UML (une conception globale du jeu) ;~~
- ☐ Validation du cahier des charges (liste des fonctionnalités à implémenter) ;
- ☐ Fin du développement (la formaliser et donner le feu vert aux tests) ;
- ☐ Tests UI validés (fin du projet officielle, précédant directement sa publication) ;

Force est de constater que les jalons sont concentrés au début et à la fin, ce qui semble normal car les tâches de développement actif sont plus continues et plus étalées dans le temps.

On remarquera qu'un jalon pourtant essentiel, la validation du cahier des charges n'a toujours pas été atteint, moins de deux mois avant la date de délivrance ! L'avenir montrera si notre intuition avait été bonne, mais, à notre connaissance, il est courant dans des méthodologies agiles de commencer le développement avec un cahier des charges flexible et évolutif, permettant de travailler sur le développement des fonctionnalités de manière *itérative* et *incrémentale*, avec des feedbacks réguliers des utilisateurs et des parties prenantes. Dans un domaine aussi exigeant sur le plan de l'expérience utilisateur qu'est l'univers des jeux, pouvoir adapter le processus du développement aux besoins et au ressenti du client (notre tuteur, d'une part, nos confrères d'autre part) peut être un atout non négligeable.

Conclusion

Malgré les perturbations des premières semaines, liées au départ inopiné du membre qui se posait comme chef de l'équipe, le projet se porte bien et avance – parce qu'on a su suppléer rapidement à son absence, certes, mais également grâce à une gestion rigoureuse.

Même si celle-ci ne trouve son expression complète que dans ce dossier, elle était présente dès le début. Nous étions conscients que certaines tâches pouvaient et devaient être réalisées en parallèle, et bien sûr qu'il était préférable de ne pas prendre de retard par rapport au *début* attendu des tâches à réaliser.

Nous savons également que réaliser une tâche critique *avant* le délai stipulé nous ferait gagner du temps pour les tâches qui en dépendent, directement ou indirectement ; ceci dit, il est extrêmement difficile de le faire étant donné que ce sont celles qui prennent déjà le plus de temps (et si en plus on applique la loi de Parkinson !). En revanche, nous avons déjà pu expérimenter l'effet bénéfique d'une réalisation précoce d'une tâche *non critique* (la création du poster), car cette démarche relativement facile permet de libérer des ressources pour les tâches ultérieures. Dans le modèle des *Sept habitudes des gens efficaces*, cela correspond au conseil de prioriser ce qui est important sans être urgent (pour éviter que cela devienne important ET urgent).

Nous voyons bien nos goulots d'étranglement, concentrés dans la phase de développement : autant de points de vigilance pour cette période essentielle qui vient de commencer.

Récapitulatif du temps consacré au projet

Comme nous ne pouvions rendre compte que du temps effectif, et étant donné qu'un récapitulatif nous était demandé à mi-parcours, le tableau ci-dessous, bien que exhaustif, n'est rempli que dans la partie déjà réalisée.

Les durées sont données en heures de travail (heures de cours plus heures de travail personnel).

Nous avons jugé utile de regrouper les tâches autour d'objectifs concrets. Ainsi, la colonne *Cahier des charges* inclut, certes, le temps de rédaction du cahier des charges effectif, mais également ceux de l'étude de l'ancien et des échanges qui ont précédé et ceux qui ont suivi. La *Conception générale* inclut l'élaboration du schéma visuel d'origine, puis du diagramme des classes UML, mais aussi la création du poster, etc.

Nous y avons rajouté le temps nécessaire pour établir ce dossier (*Dossier PERT + Gantt*).

Les colonnes correspondant aux tâches qui restent à faire sont remplies de chiffres prévisionnels, que nous avons mis en gris pour plus de visibilité. Le tableau devrait être rempli dans son intégralité à la fin du projet.

	Formation à Godot	***Cahier des charges***	***Dossier PERT + Gantt***	***Conception générale***	***Dév[1]***	***Tests***	***Finalisation et démonstration***
A. AKGÜL		8,5		2,5			1
Y. CHETTATI	8	8	4	14,5	70	15	6
E. FRISON	8	14,5	8	12,5	70	15	8
M. FRISON	8	10	4	18,5	70	15	8

Fig. 5. Temps consacré au projet par personne

Webographie sélective

Documentation

1. Diagramme de Gantt : le guide complet https://www.canva.com/fr_fr/tableau-blanc/diagramme-de-gantt/
2. Diagramme PERT ou réseau : définition et étapes simplifiées pour le mettre en place : https://blog-gestion-de-projet.com/technique-pert/
3. Modèle de conception MVC : https://www.geeksforgeeks.org/mvc-design-pattern/
4. Stephen Covey, “The Seven Habits of Hightly Effective People”. Part 7 : https://web.archive.org/web/20180127032055/http://www.stafforini.com/docs/Covey%20-%20The%207%20habits%20of%20highly%20effective%20people.pdf
5. The 12 principles behind the Agile manifesto : https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/
6. What is Scrum ? https://www.scrum.org/learning-series/what-is-scrum/

Outils et langages utilisés dans le cadre du projet

- Moteur de jeu :
  - ***Godot Engine*** https://godotengine.org/
- Logiciel de gestion de version (VCS) :
  - ***Git*** https://git-scm.com/
- Poster :
  - ***Figma*** https://www.figma.com
  - ***GIMP*** https://www.gimp.org
- Charte graphique :
  - ***Canva*** https://www.canva.com/
  - ***Color Hunt*** https://colorhunt.co/
- Schéma du déroulement du jeu :
  - ***Miro*** https://miro.com/app/dashboard/
- Diagramme de classes :
  - ***UML*** https://www.uml.org/
- Cahier des charges :
  - ***Markdown*** https://www.markdownguide.org/
- Graphe PERT :
  - ***LucidChart*** https://lucid.app/lucidchart/
- Diagramme de Gantt :
  - ***Online Gantt*** https://www.onlinegantt.com/
- Communication au sein de l’équipe :
  - ***Discord*** https://discord.com/
- Environnement de développement :
  - ***VS Code*** *https://code.visualstudio.com/*
  - ***Geany*** https://www.geany.org/
  - éditeur de code intégré de ***Godot*** https://docs.godotengine.org/fr/4.x/tutorials/editor/external_editor.html