Université Gustave Eiffel

Master Informatique

Rapport Technique

C++

Jimmy Teillard

Groupe 2

Chargé de TD: Mathias Weller

M1

Avril 2022

Contents

1 Introduction

Ce rapport technique à pour but de compiler mes résultats quant au projet de C++ de la matière échéante.

Il sera divisé en plusieurs parties, chacun traitant respectivement des différents aspects dudit projet.

2 Réponses aux questions

Les différentes questions des tâches données ont directement été répondues dans les fichiers concernés (c'est à dire les fichiers TASK_*.md. Il est à noter que les réponses sont toutes représentées par un quote block (en markdown, un quote block est un paragraphe précédé d'un chevron pointant vers la droite), ou bien par un code block (un paragraphhe entouré de triple backticks).

3 Choix d'implémentation

J'en reparlerai dans la suite du rapport, mais l'un des problèmes de ce projet est que l'on a pas vraiment beaucoup le choix, en terme d'implémentation.

Cela dit, il est à noter qu'UN choix s'est tout de même présenté quant à la façon de gérer le mouvement des avions ainsi que leur durée de vie, ou plutôt, ce qui indique qu'un avion doit être supprimé.

En effet, pour suprimer un avion, il est nécessaire de faire remonter une information à la structure qui s'occupe de gérer chaque frame/tour de jeu. En l'occurence, cette structure, à la fin du projet, est l'AircraftManager. Une façon de transmettre cette information est, entre autres, de changer la signature de la fonction DynamicObject::move, en faisant en sorte qu'elle retourne un booléen, au lieu de void. Ainsi, lorsqu'un avion est voué à être supprimé, il retournera false, au lieu de true, par exemple. Cependant, ce n'est pas ce que j'ai fait à terme : cette solution possède plusieurs problèmes, notamment en terme de responsabilité.

Il est important, en Programmation Orientée Objet, de bien séparer les responsabilité, et, en plus de cela, modulariser correctement ces fonctionnalités. Cela se traduit par le fait d'éviter au mieux que les fonctions effectuent trop de choses en même temps dans un objet. Or, donner la responsabilité à TOUT les DynamicObject de dire s'ils doivent

être supprimés après un move est une erreur de conception, car seulement les avions sont supprimables.

Par conséquent, j'ai dérivé de ce fonctionnement, et, à la place, ai opté pour un champs privé booléen dans Aircraft : `finished`. En effet, ce booléen est en temps normal à false. Mais à n'importe quel moment dans le programme (dont l'implémentation de move de Aircraft), cette valeur peut être inversée pour indiquer que ledit avion est prêt à être supprimé. D'autre part, puisque AircraftManager est amie de Aicraft, elle a accès à ce champs.

4 Difficultés

Je n'ai pas réellement rencontré de difficultés (si ce n'est me battre contre la syntaxe de C++ et ses fonctions d'ordre supérieur qui laissent à désirer comparé à des langages qui possèdent des structures fonctionnelles plus matures et naturelles comme Kotlin ou même Java).

On pourra cela dit noter la "difficulté" de la toute dernière question du projet sur le perfect forwarding ainsi que le bonus suivant (SFINAE), sur laquelle je vais revenir dans la partie suivante.

5 Appréciations (ou pas)

Cette section est vouée à expliciter ce que j'ai pu aimer (ou non) dans ce projet, ainsi que ce que j'ai pu personnellement en retirer en tant qu'étudiant.

Malheureusement, j'ai bien peur de dire que je n'ai absolument pas apprécié ce projet – et sans exagérer – en aucun point.

Je dirais que le problème majeur de ce projet, qui est source de tout les autres, et la philosophie de base qu'il propose, que j'appellerai celle du "**faites-ci, en fait non**". En effet, à énormément de points dans ce projet, on nous demande d'implémenter une solution, pour ensuite la remplacer par "la bonne chose à faire". Je comprends bien que ce genre d'approche à pour but de donner des points sur les premières implémentations, dans le cas où un élève ne parviendrait pas à implémenter "la vraie solution" ; cependant, le problème de cette approche est qu'elle amène à des situations qui causent confusion, ou font perdre du temps (parce qu'il faut refactor à plein d'endroits) sans pour autant que cela soit intéressant d'un point de vu apprentissage.

D'autre part, il est à noter qu'un dériver du problème précédent est simplement le fait que l'on parte d'un projet déjà implémenté en majorité, avec une codebase volontairement "incorrecte". J'aurai personnellement préféré partir de zero, sur un projet potentiellement moins complexe (sans pour autant être plus simple), et qui amènerai alors a comprendre le langage lui même. Cela à amené à des questions pénibles et peu naturelles, telles que celles concernant l'ajout d'assertions dans le programmes. C'est normalement un procédé auquel on se prend avant même de coder (tests unitaires) ou, au plus tard, **pendant** que l'on code (et non après). Les assertions servent à éviter de se perdre lorsqu'un bug apparaît, et non pas à les ajouter une fois qu'on a trouvé un bug et qu'on s'y perd après coup, dans un gros projet.

La culmination de mon manque d'intérêt dans ce projet résulte de tout cela : ce n'est pas un projet de C++, c'est un projet d'une matière qui s'appellerait "correction de mauvais code qui n'est absolument pas spécifique à C++". Je considère qu'étant étudiants (j'inclus mes camarades) en Master, nous sommes en mesure de transférer la majorité de nos savoirs quant aux langages qu'on a pu apprendre jusque là (On a appris à apprendre, si cela fait sens). C'est pour ça que, quand je vois des questions du genre "quelle structure de donnée utiliser ici", pour répondre "une map", alors que ce n'est absolument pas spécifique à C++, je suis surpris du détachement complet de l'intérêt d'une matière au stade ou nous en sommes, et du peu de sens que cela a.

Pour revenir à la "question difficile" mentionnée précédemment, c'est seulement à ce moment là que j'ai trouvé un intérêt conséquent au projet, c'était enfin un problème et une solution spécifique à C++, et non un problème générique ; au final c'est à ce moment que j'ai appris quelque chose.