# Compte-rendu - Tower Control Simulation

Crée par SAVANE Kévin

https://github.com/LyKell/CPP_Learning_Project

Le 23 avril 2021

# Task-0 | Se familiariser avec l'existant

## C- Bidouillons !

1. Déterminez à quel endroit du code sont définies les vitesses maximales et accélération de chaque avion. Le Concorde est censé pouvoir voler plus vite que les autres avions. Modifiez le programme pour tenir compte de cela.

> Les vitesses maximales et accélération de chaque avion sont définies dans le fichier `aircraft_types.hpp`.
>
> ```
> inline void init_aircraft_types()
> {
>  aircraft_types[0] = new AircraftType { .02f, .05f, .02f, MediaPath { "l1011_48px.png" } };
>  aircraft_types[1] = new AircraftType { .02f, .05f, .02f, MediaPath { "b707_jat.png" } };
>  aircraft_types[2] = new AircraftType { .02f, .08f, .04f, MediaPath { "concorde_af.png" } };
> }
> ```
>
> Pour que le Concorde soit plus rapide que les autres avions, il suffit simplement de modifier les deux dernières valeurs flottantes de `aircraft_types[2]` qui sont la vitesse de croisière et l'accélération.

2. Identifiez quelle variable contrôle le framerate de la simulation. Que se passe-t-il [lorsque vous essayez de mettre en pause le programme en manipulant ce framerate] ?

> La variable contrôlant le framerate est `constexpr unsigned int DEFAULT_TICKS_PER_SEC` du fichier `config.hpp`. Je ne me souviens plus de ce qu'il se passe lorsqu'on essaye de mettre en pause le programme via le framerate.

3. Identifier quelle variable contrôle le temps de débarquement des avions et doublez-le.

> La variable contrôlant le temps de débarquement des avions est `constexpr unsigned int SERVICE_CYCLES` du fichier `config.hpp`.

5. Pourquoi n'est-il pas spécialement pertinent [de gérer automatiquement l'ajout et la suppression d'objet] pour `DynamicObject` ?

> Si on décide de supprimer automatiquement les `DynamicObject` alors qu'on le fait déjà pour les `Displayable`, on risque de supprimer un objet déjà supprimé.

> Je n'ai pas de réponse pour l'ajout.

### D- Théorie

1. Comment a-t-on fait pour que seule la classe `Tower` puisse réserver un terminale de l'aéroport ?

> La classe `Tower` est une **friend class** de `Airport`. C'est la seule classe à pouvoir accéder aux membres et fonctions-membres privés de `Airport`.

2. En regardant le contenu de la fonction `void Aircraft::turn(Point3D direction)`, pourquoi selon-vous ne sommes-nous pas passé par une référence ? Pensez-vous qu'il soit possible d'éviter la copie du `Point3D` passé en paramètre ?

> La méthode `Point3D::cap_length()` modifie le point appelant. Peut-être qu'on ne souhaite pas modifier le point passé en paramètre ?
>
> Je dirais qu'il n'est pas possible d'éviter la copie car `Aircraft::turn()` est utilisée uniquement dans `Aircraft::turn_to_waypoint()`, mais avec une expression (`target - pos - speed`), qui n'est pas stockée dans la mémoire. Il faudrait qu'elle le soit pour pouvoir accéder à sa référence.

# Task-1 | Gestion des ressources

## Objectif 1 - Référencement des avions

### B- Déterminer le propriétaire de chaque avion

1. Qui est responsable de détruire les avions du programme ?

> La méthode `GL::timer()` dans `opengl_interface.cpp`.

2. Quelles autres structures contiennent une référence sur un avion au moment où il doit être détruit ?

> `Terminal::current_aircraft`, `GL::Displayable::display_queue` et `GL::move_queue` possède des références sur un avion au moment de sa destruction.

3. Comment fait-on pour supprimer la référence sur un avion qui va être détruit dans ces structures ?

> On appelle `move_queue.erase()` pour `GL::move_queue`, le destructeur de `Displayable` pour `GL::Displayable::display_queue` et `Terminal::current_aircraft` devient `nullptr` lorsque le terminal ne sert plus l'avion.

4. Pourquoi n'est-il pas très judicieux d'essayer d'appliquer la même chose pour votre `AircraftManager` ?

> `AircraftManager` devrait s'occuper de la destruction des avions alors que le travail est déjà fait. Il faudrait en plus pouvoir accéder à `Terminal::current_aircraft` qui est un membre privé de `Terminal`.

# Task-4 | Templates

## Objectif 1 - Devant ou derrière ?

2. Que devez-vous changer dans l'appel de la fonction pour que le programme compile ?

> Il faut ajouter `constexpr` entre le `if` et la condition :

```
1 if constexpr (front)
```

3. Comparez le code-assembleur généré par [deux] fonctions

> La deuxième fonction ne génère pas de code-assembleur sur GodBolt

# Difficultés

Lorsque je faisais la Task-2, le réapprovisionnement de kérozène, je me suis retrouvé bloqué une bonne heure sur la question 6, modifier la fonction `Airport::update`. En fait, le sujet que je lisais n'était pas celui à jour, j'ai récupéré les derniers commits du projet, mais je n'ai jamais rafraîchis le sujet, et je me suis retrouvé à chercher pendant un bon moment comment modifier `Aircraft::update`, alors que c'était `Airport::update` qu'il fallait modifier.

La gestion des numéros de vol unique m'a aussi bloqué pendant un moment, alors que la solution était toute bête. J'avais peut-être plus la tête pour travailler à ce moment-là.

J'ai eu beaucoup de mal à trouver la réponse pour la Task-0 C-4, car il y avait beaucoup de fichiers à modifier et beaucoup de code à rajouter. La faute vient de moi parce que je n'étais pas présent aux TPs, notamment le TP pour cette Task. J'ai fini par regardé la solution de la Task-0 que vous avez fournie.

# J'aime

Le projet était très ludique et donc très amusant à compléter. Ca peut paraître relativement enfantin, mais je trouve les avions, et de manière générale les appareils permettant à l'Homme de voler, absolument fascinants.

# Apprentissage

La syntaxe par moment très bizarre à première vue, mais assez simple en vrai. Le langage est évidemment très proche du C, et comme nos bases en C sont plutôt solides, coder en C++ était relativement aisé. L'avantage, c'est qu'on n'a pas besoin de déclarer tous les membres et fonctions-membres publique ou privée, il suffit juste des les mettre dans le bloc adéquat. En Java, on est obligé de mettre le modificateur de visibilité devant chaque attribut ou méthode.

La quantité gigantesque de headers et de fonctions de la STL, quel plaisir de pouvoir parcourir une librairie si riche !

Les erreurs de compilation sont une HORREUR absolue. M. Borie nous disait que le compilateur de C nous insultait quand on avait des erreurs, j'ai l'impression que le compilateur de C++ veut ma mort à chaque erreur. Il y a beaucoup d'erreurs que j'ai eu beaucoup de mal à comprendre, des erreurs que j'avais l'impression n'avaient aucunement lieu d'être (alors que très

certainement si, mais je ne sais pas pourquoi).

Les pointeurs et les références, même avec le C et le C++, j'ai toujours un peu de mal à comprendre.

La gestion des ressources est aussi très difficile à appréhender. Je comprends le principe, mais déterminer qui est owner de telle ou telle ressource est plutôt compliqué.

La possibilité de surcharger les opérateurs est géniale. C'est très simple et très logique, alors qu'en Java, on est obligé d'implémenter une interface.

Comme dans tous les langages que j'ai appris : **Je. N'arrive. Pas.** A retenir la syntaxe pour construire un objet ou initialiser des conteneurs.