|  |
| --- |
|  |
| tower\_control |
| Master 1 Informatique Programmation générique en C++  Étudiant : Decilap Jason Groupe1  Professeur : Noël Céline & Marsault Victor 2021-2022 |

Table des matières

[Réponses 3](#_Toc96178686)

[Task0 3](#_Toc96178687)

[A - Exécution 3](#_Toc96178688)

[Task1 3](#_Toc96178689)

[Task2 3](#_Toc96178690)

[Task3 3](#_Toc96178691)

[Task4 3](#_Toc96178692)

[Architecture 3](#_Toc96178693)

[Difficultés 3](#_Toc96178694)

[Ressentis 3](#_Toc96178695)

[Apprentissage 4](#_Toc96178696)

# Réponses

## Task0

### A – Exécution

Allez dans le fichier `tower\_sim.cpp` et recherchez la fonction responsable de gérer les inputs du programme.

Sur quelle touche faut-il appuyer pour ajouter un avion ?

Comment faire pour quitter le programme ?

A quoi sert la touche 'F' ?

La fonction TowerSimulation::create\_keystrokes est responsable de gérer les inputs du programme.

La touche 'C' permet d'ajouter un avion.

La touche 'X' ou 'Q' permet de quitter le programme.

La touche 'F' permet de mettre le plein écran.

Lorsqu’il y a un avion son comportement est le suivant :

* 1. L'avion vole
  2. L'avion atterrie
  3. L'avion se fait entretenir
  4. L'avion redécolle

(On recommence le cycle)

Ajoutez un avion à la simulation et attendez.

Quel est le comportement de l'avion ?

Quelles informations s'affichent dans la console ?

Pour l’avion BA9850 les informations suivantes s’affichent dans la console :

* 1. BA9850 is now landing... (Atterrissage)
  2. now servicing BA9850... (Début de l'entretient)
  3. done servicing BA9850 (Fin de l'entretient)
  4. BA9850 lift off (Décollage)

(On recommence le cycle)

Ajoutez maintenant quatre avions d'un coup dans la simulation.

Que fait chacun des avions ?

Lorsqu’il y a 4 avions :

* + Il y a 3 avions maximum qui peuvent atterrir et aller à l'entretient.
  + Les avions supplémentaires attendent en continuant de voler puis atterrissent quand il y a une place de libre pour l’entretient.

### B - Analyse du code

Listez les classes du programme à la racine du dossier src/.

Pour chacune d'entre elle, expliquez ce qu'elle représente et son rôle dans le programme.

|  |  |  |
| --- | --- | --- |
| **Class** | **Représentation** | **Rôle** |
| Aircraft | Avion | Créer des avions avec ses différentes propriétés. |
| AirportType | Type d’aéroport | Avoir différent type d’aéroport. |
| Airport | Aéroport | Créer un aéroport avec ses éléments |
| Terminal | Terminal de l’aéroport | Gérer les fonctionnalités apporter par un terminal sur les avions. |
| TowerSimulation | Action de l’utilisateur | Déclencher les fonctionnalités en fonction de la touche appuyée. |
| Tower | Tour de contrôle | Contrôler les fonctions des terminaux et des avions |
| Waypoint | Point de la trajectoire d’un avion | Point par lequel un avion va passer, une suite de ces points sera une trajectoire. |

Pour les classes Tower, Aircaft, Airport et Terminal, listez leurs fonctions-membres publiques et expliquez précisément à quoi elles servent. Réalisez ensuite un schéma présentant comment ces différentes classes interagissent ensemble.

|  |  |  |
| --- | --- | --- |
| **Class** | **Fonction** | **Utilité** |
| Tower | Tower | Constructeur qui affecte un Airport à la Tower. |
| get\_instructions | Prend en paramètres une référence Aircraft et renvoie une deque de la trajectoire que l’Aircraft devra prendre :  Se diriger vers l’aéroport si l’Aircraft est loin  Faire un tour en attendant qu’un Terminal se libère  Sortir du Terminal pour s’envoler. |
| arrived\_at\_terminal | Prend en paramètre une référence constante Aircraft et ne renvoie rien, lance le service du Terminal sur lequel se situe l’Aircraft. |
| Aircraft | Aircraft | Constructeur qui affecte un type, identifiant, position, vitesse, Tower à l’Aircraft. |
| get\_flight\_num | Renvoi une référence constante de l’identifiant du Aircraft. |
| distance\_to | Prend en paramètre une référence constante d'un Point3D et  renvoie la distance entre l'avion et ce point. |
| display | Affichage de l'Aircraft. |
| move | Fait déplacer l'avion vers son prochain waypoints, s’il n’y en a aucun alors on demande a la Tower la prochaine destination. |
| Airport | Airport | Constructeur qui affecte un type, une position, Terminal etc… |
| get\_tower | Renvoie une référence Tower qui représente la tour de contrôle. |
| display | Affichage de l'Airport. |
| move | Fait appel à la méthode move de chaque Terminal |
| Terminal | Terminal | Constructeur qui affecte une position a un Terminal |
| in\_use | Renvoie un boolean qui dit si le Terminal est libre. |
| is\_servicing | Renvoie un boolean qui dit si le Termial est en train de travailler. |
| assign\_craft | Prend en paramètre une référence constante Aircraft et lui assigne la référence à son current\_aircraft. |
| start\_service | Prend en paramètre une référence constante Aircraft, cette fonction annonce que le Terminal commence le service. |
| finish\_service | Annonce la fin du service et libère son current\_aircraft. |
| move | Fait avancer le processus de service de l'avion, incrémente son  service\_progress. |

has

move

lance le

service

trajectoire

service

Aircraft

Terminal

Tower

Airport

Quelles classes et fonctions sont impliquées dans la génération du chemin d'un avion ? Quel conteneur de la librairie standard a été choisi pour représenter le chemin ? Expliquez les intérêts de ce choix.

Lors de l’appel de Aircraft::move si la deque est vide on fait appel à Tower::get\_instruction.

Tower::get\_instruction :

* Appelle Tower::get\_circle() qui renvoie une trajectoire circulaire pour faire patienter l’Aircraft si il n’y a pas de Terminal disponible.
* Appelle Airport::reserve\_terminal pour avoir la trajectoire vers le Terminal réserver pour l’Aircraft
* Appel Airport::start\_path pour faire décoller l’Aircraft de l’Airport.

Le conteneur std::deque a été choisit pour représenter un chemin car il permet :

* L'ajout d'éléments à la fin du conteneur.
* Réception d’éléments au début du conteneur.
* Complexité O(1).

### C - Bidouillons !

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

Dans le fichier aircraft\_types.hpp dans la struct AircraftType nous avons les champs :

* float max\_air\_speed qui représente la vitesse maximale de l'Aircraft
* float max\_accel qui représente l'accélération maximale de l’Aircraft

1. Identifiez quelle variable contrôle le framerate de la simulation. Ajoutez deux nouvelles entrées au programme permettant d'augmenter ou de diminuer cette valeur. Essayez de maintenant mettre en pause le programme en manipulant ce framerate. Que se passe-t-il ?

Ajoutez une nouvelle fonctionnalité au programme pour mettre le programme en pause, et qui ne passe pas par le framerate.

Dans le fichier config.hpp la variable constexpr unsigned int DEFAULT\_TICKS\_PER\_SEC permet de contrôler le framerate de la simulation.

La touche 'U' permet d'augmenter le framerate.

La touche 'D' permet de diminuer le framerate

Dans la méthode GL::timer nous avons une division par ticks\_per\_sec, donc si nous dé-incrémentons trop nous aurons une division par 0 et cela provoquera des évènement inattendue. De plus il faut faire attention au dépassement de capacité.

La touche 'P' permet de mettre le programme en pause et de reprendre.

1. Identifiez quelle variable contrôle le temps de débarquement des avions et doublez-le.

C’est la variable constexpr unsigned int SERVICE\_CYCLES dans le fichier config.hpp

1. Lorsqu'un avion a décollé, il réattérit peu de temps après. Faites en sorte qu'à la place, il soit retiré du programme.

Indices :

A quel endroit pouvez-vous savoir que l'avion doit être supprimé ?

Pourquoi n'est-il pas sûr de procéder au retrait de l'avion dans cette fonction ? A quel endroit de la pile d'appels pourriez-vous le faire à la place ?

Que devez-vous modifier pour transmettre l'information de la première à la seconde fonction ?

On change le nom de la méthode GL::DynamicObject::move en GL::DynamicObject::update qui renvoie true si on a bien mis l’objet a jour et false si on doit supprimer l’objet.

Pour cela on ajoute un champs bool service\_done qui indique si on a fini l’entretient de l’Aircraft dans un terminal de la base.

Par défaut il est à false et on le met à true dans Tower::get\_instructions dans la condition qui indique que l’entretient de l’avion est terminé.

Dans le méthode GL::timer on parcourt la liste des DynamicObject puis au fur et à mesure des updates on supprime ou non les Aircraft dont la méthode Aircraft::update renvoie true.

1. Lorsqu'un objet de type Displayable est créé, il faut ajouter celui-ci manuellement dans la liste des objets à afficher. Il faut également penser à le supprimer de cette liste avant de le détruire. Faites en sorte que l'ajout et la suppression de display\_queue soit "automatiquement géré" lorsqu'un Displayableest créé ou détruit. Pourquoi n'est-il pas spécialement pertinent d'en faire de même pour DynamicObject?
2. La tour de contrôle a besoin de stocker pour tout ce qui lui est actuellement attribué, afin de pouvoir le libérer une fois que l'avion décolle Aircraft. TerminalCette information est actuellement enregistrée dans un std::vector<std::pair<const Aircraft\*, size\_t>>(size\_t représentant l'indice du terminal). Cela fait que la recherche du terminal associé à un avion est effectuée en temps linéaire, par rapport au nombre total de terminaux. Cela n'est pas grave tant que ce nombre est petit, mais pour préparer l'avenir, on aimerait bien remplacer le vecteur par un conteneur qui garantira des opérations efficaces, même s'il y a beaucoup de terminaux.

Modifiez le code afin d'utiliser un conteneur STL plus adapté. Normalement, à la fin, la fonction find\_craft\_and\_terminal(const Aicraft&)ne devrait plus être nécessaire.

### D- Théorie

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

Dans la classe Tower il y a un champ privé AircraftToTerminal reserved\_terminals qui liste les terminaux utilisés.

1. En regardant le contenu de la fonction `void Aircraft::turn(Point3D direction)`, pourquoi selon-vous ne sommes-nous pas passer par une référence ?

Pensez-vous qu'il soit possible d'éviter la copie du `Point3D` passé en paramètre ?

Celons-moi il n’est pas nécessaire de passer par une référence car :

* Nous ne voulons pas d’effet de Bord.
* Ici sera copié le résultat direct de la création d’un Point3D dans la fonction turn et si nous voulions passer par référence le résultat sera non pas enregistré directement dans la méthode turn mais dans une variable, donc pas de réel changement concernant l’espace mémoire.

Il est possible d’évité la copie du Point3D en créant une variable pour stocker le point puis de passer sa référence à turn.

### E-Bonus

## Task1

## Task2

## Task3

## Task4

# Architecture

# Difficultés

# Ressentis

# Apprentissage

Vous avez rédigé un petit rapport pour ce projet contenant :

* les réponses aux questions ‘non-code’ dans les différentes tâches, si vous n’y avez pas répondu directement dans les fichiers TASK\_#.md
* les choix auxquels vous avez été confrontés par rapport à l’architecture du programme, et les raisons qui ont fait que vous êtes parti dans une direction plutôt qu’une autre,
* les situations où vous avez bloqué, et si applicable, la solution que vous avez trouvée pour vous en sortir,
* ce que vous avez aimé et/ou détesté dans ce projet,
* ce que vous pensez en avoir appris.