RAPPORT DU PROJET

C++

Présenté Par :

Moussa Ndiaye

Université Gustave Eiffel

2021/2022

Présentation de projet

Le bute de ce projet et de changer et améliorer un projet qui existe déjà et qui représente une implémentation d'une bibliothèque graphique qui affiche un aéroport et des avions qui bougent au tour de lui et qu'ils atterrissent et plusieurs classes qui le gèrent.

Ce projet est codé en C++(Programmation Objet) ça permet à comprendre comment prendre un code existant et l'améliorer sans changer les bonnes fonctionnalités qui extensité déjà

Réalisation

j'ai réussi à répondre a toutes les questions et faire touts le code demander mais la parties Optionnels j'ai faits juste quelques questions par manque de temps j'ai pas réussi à faire les autres .

Task0

A- Exécution

Dans le fichier tower_sim.cpp la fonction responsable de gérer les inputs du programme est create_keystrokes().

→ pour ajouter un avion if faut appuyer sur la touche « c »
→ pour quitter le programme if faut appuyer sur la touche « q » ou « x »
→ La touche 'F' sert à activer et désactiver le plein écran

Après l'ajout d'un avion à la simulation on remarque que l'avion vole et atterrit puis il vas dans le terminal et fait des tours au tour de l'aéroport.

Dans la console s'affichent des informations sur le déroulement de déplacement de l'avion .

Après l'ajout de quatre avions d'un coup dans la simulation on remarque que ils volent et ils atterrissent mais juste 3 parmi ces 4 qui peuvent atterrir en même temps

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.

- Aircraft : qui représente l'objet Avion.
- Airport : qui représente l'aeroport.
- AircraftType : represente le type d'un avion elle stocke les vitesse et l'accélération des avions.
- AirportType :contient les positions de l'aeroport comme le debut et fin des runways (il peut en avoir plusieurs), chaque AirportType peut indiquer des chemins pour l'avion.
- Terminal : gère le débarquement d'un avion pendant un certain temps un terminal ne peut débarquer qu'un seul avion au même temps.

- Tower : une classe qui gère les instructions donnés aux avions comme les fonctionnalités du tour de contrôle .
- TowerSimulation : une classe pour la gestion de la simulation et es actions utilisateur comme création de l'aéroport et des avions, affichage de l'usage sur la ligne de commande… ect.
- Waypoint : un point 3D qui indique s'il est au sol ou non chez un terminal ou dans l'air, on voit aussi qu'un "chemin" (WaypointQueue) est un deque de Waypoints.
- Displayable : classe abstraite qui représente les objets qu'on peut dessiner dans le projet. Avec la coordonné z on peut trier les objets de cette classe.
- DynamicObject : classe abstraite qui forme la base pour tous les objets qui ont la spécificité de bouger.
- opengl-interface : contient les quelques fonctions importantes pour interagir avec OpenGL dans le cadre du projet.
- Texture2D :contient un pointeur vers un img::Image qui contient les octets vrac de l'image.
- Image : une classe qui gère les octets d'une image.
- MediaPath : gère l'accès aux PNGs qui sont dans le code.
- stb_image : c'est une bibliothèque C qui sait lire les PNGs correctement.
- Runway : stocke le début et le fin d'un piste de décollage.

→ Pour les classes Tower, Aircaft, Airport et Terminal, la liste de leurs fonctions-membre publiques et à quoi elles servent

, listez leurs fonctions-membre publiques et expliquez précisément à quoi elles servent.

Tower :

- get_instructions(Aircraft&) : cette fonction sert à génèrer des instructions pour l'avion
- l'avionarrived_at_terminal(Aircraft&) : cette fonction sert à récupèrer le terminal ou il atterrit l'avion et après le passer à l'avion.

Aircraft :

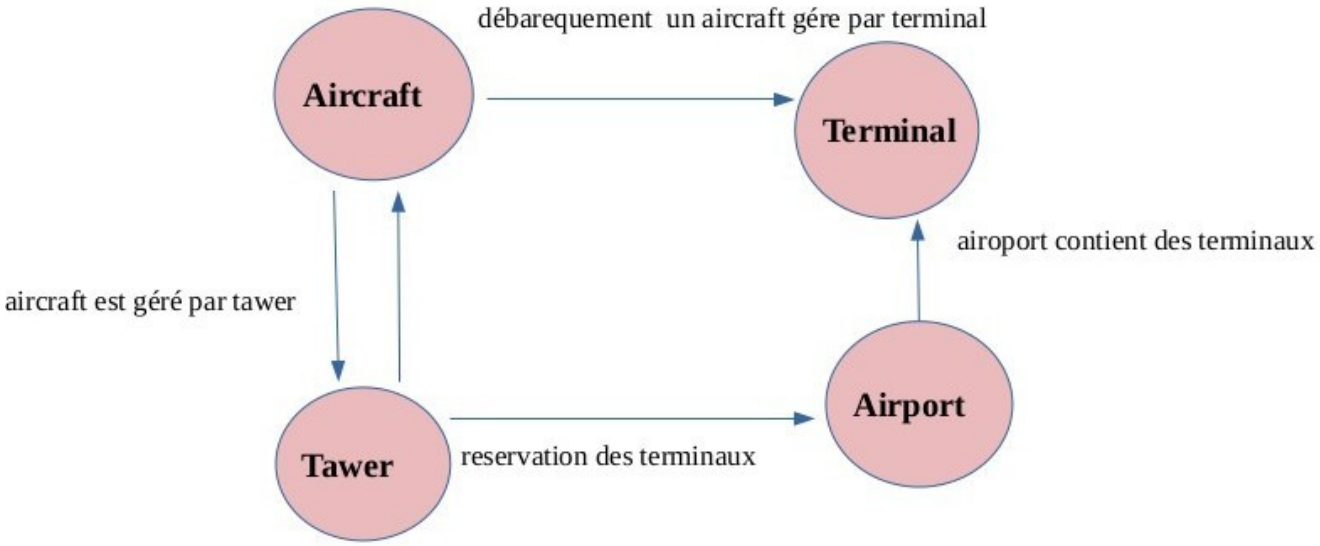
- get_flight_num() : sert à renvoyer le numéro de vol de cet avion
- distance_to(): sert à renvoyer la distance entre cet avion et un point
- display() : sert à afficher l'avion
- move() : sert à mettre à jours la position de l'avion

Airport :

- get_tower() : sert à renvoyer la tour de contrôle
- display() : sert à afficher l'aéroport
- move() : sert à faire bouger tous les terminaux

Terminal :

- in_use() : sert à renvoyer true s'il est en cours d'utilisation
- is_servicing() : sert à renvoyer true tant qu'il n'a pas fini de s'occuper de l'avion
- assign_craft() : sert à assigner un nouvel avion
- start_service() : Affiche un message et elle s'occuper de l'avion
- finish_service() : Affiche un message et elle supprime sa référence sur l'avion
- move() : Ajoute 1 au compteur de traitement de l'avion



→ Les classes et les fonctions qui sont impliquées dans la génération du chemin d'un avion sont
Aircraft::move() ,Tower::get_instruction() , Airport::reserve_terminal() Terminal::assign_craft() .

→ Pour représenter le chemin on a choisi une queue comme conteneur de la librairie standard parce que la queue nos donne la possibilité d'ajouter des points à la fin et les récupérer de début de la queue

C- Bidouillons !

1) l'endroit du code ou sont définies les vitesses maximales et accélération de chaque avion est dans AircraftTypes, le concorde peut plus vite que les autres avions en changeant les valeurs.
2) La variable qui contrôle le framerate de la simulation est GL::tick_per_frame initialisé à config::DEFAULT_TICKS_PER_SEC
les deux nouveaux inputs au programme permettant d'augmenter ou de diminuer cette valeur sont : « z » pour augmenter et « a » pour le réduire

3) la variable qui contrôle le temps de débarquement des avions est
: config.cpp -> SERVICE_CYCLES -> 40u

4) l’endroit pour savoir que l'avion doit être supprimé est :Aircraft::move
n'est pas sûr de procéder au retrait de l'avion dans cette fonction car
c’est mieux de le faire dans GL::move_queue
l’endroit de la callstack ou on peut le faire à la place est dans GL::timer
pour transmettre l'information de la première à la seconde fonction il faut
changer la signature de la fonction pour renvoyer un booléen.

5) Il n’est pas pertinent de faire de même pour un DynamicObject car ces derniers ont un comportement moins prévisible. Les avions peuvent crasher par exemple ou simplement finir leur service et quitter la scène etc.

6) j’ai modifié le code afin d'utiliser un conteneur STL plus adapté et pour cela j’ai utilisé une std::map pour supprimer ou chercher un élément en O(1).

D- Théorie

1) pour que seule la classe Tower puisse réserver un terminal de l'aéroport il faut que juste Airport qui peut appeler la méthode assign_craft.

2) En regardant le contenu de la fonction void Aircraft::turn(Point3D direction), ne nous sommes pas passer par une référence Parce qu'on veut appliquer un traitement à direction avant de l'ajouter à la vitesse c’est possible d'éviter la copie du Point3D passé en paramètre mais pour cela il faudrait faire les traitements différemment.

Task1

Objectif 1 - Référencement des avions

A - Choisir l'architecture

il faut avoir une classe qui référence tous les avions renvoie celui qui nous intéresse soit créer une nouvelle classes soit donner ce rôle à une classe existante.
Pour moi c’est avantageux d’avoir une classe est de lui donné toute la responsabilité de la gestion de ce qu’on a besoin donc j’ai opté pour ce choix malgré que il faut rajouter une référence vers le manager dans plusieurs classe.

B - Déterminer le propriétaire de chaque avion

Dans cette partie j’ai introduit une nouvelle liste de références sur les avions du programme et mettre à jour le gestionnaire d'avions lorsque l'un d'entre eux disparaît.
Répondez aux questions suivantes :

1. le responsable de détruire les avions du programme
   est : opengl_interface.cpp dans la move_queue
2. les autres structures qui contiennent une référence sur un avion au
   moment où il doit être détruit est : GL::display_queue et reserved_terminals

3. pour supprimer la référence sur un avion qui va être détruit dans ces structures On doit la chercher et la supprimer manuellement
4. n'est pas très judicieux d'essayer d'appliquer la même chose pour AircraftManager car AircraftManager détiendra l'ownership des avions, c'est donc elle qui contrôlera leur durée de vie.

C - C'est parti !

Pour le type qui met bien en avant le fait que AircraftManager est propriétaire des avions j'ai choisi std::unique_ptr

Pour que le gestionnaire supprime les avions après qu'ils aient décollé.
J'ai utilisé l'héritage tel que Aircraft_Manager hérite DynamicObject et avoir le type de retour de la fonction update .

Objectif 2 - Usine à avions

A - Création d'une factory

la création de la classe AircraftFactory dont le rôle est de créer des avions a été bien fait.

B - Conflits

j'ai fait en sorte qu'il ne soit plus possible de créer deux fois un avion avec le même numéro de vol.

Task2

Dans cette partie de Task2 on nous demande de faire beaucoup de code et changer plusieurs méthode dans le bute d'ajouter des fonctionnalités au projet et ajouter la gestion du kérosène des avions donc j'ai
réimplémenté plusieurs méthodes en utilisant les algorithme de STL et les structures binding

dans l'objectif 1
j'ai ajouté quelque modifications dans la classe AircraftManager et statistiques pour être bien capable de compter tous les avions de chaque airline
de plus j'ai réimplémenté la classe Point3D en changeant les méthodes des operator
dans l'objectif 2
contient plusieurs parties pour la gestion kérosène et le s crashs
d'avion j'ai réussi cette parie en suivant ce qui est demandé dans l'annoncé question par question

Task3

Objectif 1 - Crash des avions

Dans l'objectif 1 on traite les exceptions de crach d'un avion et on rajoute quelques améliorations exemple une touche « m » qui indique le nombre d'avion qui a craché depuis le lancement du projet

un avion s’écrase si il n’a plus du Kérozéne ou s’il a atterrit trop vite et donc on léve des exceptions pour gérer ça et on supprime l’avion et toutes ces liaison exemple s’il a réservé un terminal on le libère

Objectif 2 - Détecter les erreurs de programmation

Dans l’objectif 2 on rajoute quelques changement pour sécuriser le code, on repasse sur les différentes fonctions de programme et on rajoute des assertions permettant de vérifier qu'elles sont correctement utilisées.

Task4

Objectif 1 – Devant ou derrière ?

Il faut modifier Aircraft::add_waypoint afin que l'évaluation du flag ait lieu à la compilation et non à l'exécution. Et pour cela il faut rendre la méthode générique

dans l'appel de la fonction on le change à :

const bool front = false;

add_waypoint<front>(wp);

pour que le programme

compile

Objectif 2 - Points génériques

le bute de Objectif2

pour l’Objectif 2 le bute est de changer la classe geometry en créant une classe Point générique pour instancier un point de tout dimension et pas juste 2 et 3 dimension existant déjà

pour cela on supprime les deux structures Point2D et Point3D et générer une classe Point avec un structure générique et de mettre toutes les méthodes existe déjà en Point2D et Poni3D mais d’une façon a traiter n'importe quel nombre d'arguments de n'importe quel type

puis on peut créer des alias de Point avec la dimension qu’on souhaite et dans ce projet c’est Point2D et Point3D

4) Dans la fonction test_generic_points**,** si on essaie d'instancier un Point2D avec 3 arguments y’aura un problème car le size de Point2D est égale à 2 et on a mis 3 arguments alors que e tableau supporte pas 3 arguments

5) on aura toujours des problèmes si on essaie d'instancier un Point3D avec 2 arguments. Et pour régler ce problème on utilise un static_assert afin d’assurer que personne ne puisse initialiser un Point3D avec seulement deux éléments. Et de même dans les fonctions y() et z(), pour vérifier que l'on ne puisse pas les appeler sur des Point qui n'ont pas la dimension minimale requise.

Difficultés

J’ai rencontré pas mal de difficultés :
→ des difficultés à comprendre le code fournit dans le projet
→ quelques complication dans la TAsk2 (ça m’a pris beaucoup du temps)
→ La syntaxe de C++ qui est un peu compliqué par apport aux autres langages
→ quelques difficultés dans la connexion privée
le manque du temps à cause du travail des autres matières.

Mais grâce aux travail en groupe dans les salons de discord et l’aide des profs et beaucoup de recherche
J’ai réussi a faire touts ce qui est demandé dans toues les Tasks

Ce que j’ai appris (aimer/pas aimer):

Ce projet fut pour moi un moyen de mettre en pratique les connaissances acquises sur ce 2ème semestre en C++. De plus il m’a permis de bien comprendre certaines notions de la programmation objet en C++.
J’ai bien aimé ce projet car je trouve qu’il est intéressant et très riche des notions et il m’a trop aidé à bien assimiler ce que nous avons fait dans le cours