Algorithmes

Objectif 1 - Refactorisation de l'existant

A - Structured Bindings
On va tout simplement donnée un nom au first et second du pair dans le for avec le structure binding,
on va du coup faire un [nom du first, nom du second] : pair.

B - Algorithmes divers
1. On va utiliser, remove_if et erase pour pouvoir enlever les avions supprimés quand ils ont finis
leurs parcours si celui ci à son champs "if_destroy()" à vrai. Donc on utilise l'iterator,
et on code un prédicat avec function qui renvoit un booléen et prend en paramètre un
unique_ptr de Aircraft, rien n'est capturé ([]), et un unique_ptr de aircraft en paramètre et va
move() l'avion et renvoyer si on doit le détruire ou pas.

2. On va ajouter un champ int dans les avions qui represente l'indice de l'airline qu'il represente
dans le tableau d'airlines et du coup faire une fonction dans aircraft_manager qui compter le nombre d'avion
qui a comme indice dans son champ, le x en paramètre. Quand on crée l'avion dans la factory, on va créer
un int qui represente l'indice aléatoire dans airlines, et le mettre dans la création de l'avion.

C - Relooking de Point3D

1.On va utiliser std::transform pour changer chaque valeur de l'array, comme on veut juste les multiplier par un float
On a juste à faire un iterator, partir de begin de l'array qui represente les coordonnées du point, on va
jusau'à la fin (end() de l'array), et on modifie l'array donc le 3eme paramètre est le début de l'array
(begin()), et on finis par la lambda qui va etre appliquer pour chaque valeur donc la multiplier
par le scalairen, donc on capture scalar, on prend en parametre un float (la valeur de l'array) et on renvoi
la modification donc le float mulitplier par le scalaire.

2. Pour les opérateur += et -= c'est du meme style, on utilise transform car on veut modifier le tableau.
On parcours du coup le debut de values, jusq'à la fin et on parcours les valeurs dans le Point en paramètre
do nc on pars des valeurs du points en paramètres et de notre points et on les additionne avec std::plus<float>
et pareil avec -= juste qu'on utilise minus<float>.

3. Pour celui ci, on va modifier ce qu'on a dans std::sqrt par std::accumulate qui parcours un tableau
et ajoute toutes les valeurs de celui ci en fonction de la lambda (ou fonction) qu'on a en dernier paramètre
et en 3ème la valeur initial, ici 0 vu qu'on veut une somme, on par de values.begin() jusqu'à sa fin avec end()
puis pour la lambda on prend le compteur (un long) et la valeur du tableau (le float), et on renvoit
le compteur additionner à la valeur du tableau au carré.

Objectif 2 - Rupture de kérosène
A - Consommation d'essence

On va faire 2 fonctions et 2 champ privé (un booléen qui regarde si il a encore de l'essence et fuel qui
 vaut l'essence de l'avion), la fonction update va decrementé le fuel de l'avion si il est différent de 0,
 si fuel est égale à 0 on passe le booléen à true, et dans le move de AircraftManager on va regarder aussi
 si l'avion a de l'essence ou non, donc on appel le getter du booléen qu'on va mettre dans la classe aircraft
 du coup on regarde si on doit le détruire ou si il a plus d'essence. On update que quand l'avion est dans les airs
, donc dans le else du if qui vérifie si l'avion est sur l'aéroport.

B - Un terminal s'il vous plaît

1. On va regarder si dans notre map reserve_terminal qui reserve les terminaux dans tower, (control est le champ
Tower de l'avion en question) contient l'avion courant, si on le trouve avec la fonciton find on renvoi vrai
sinon on renvoit faux.

2. On renvoit le contraire de la fonction qu'on a en 1, car si on a pas reservé un terminal pour lui, on attend de
se faire assigner un terminal.

3. Pour reserver le terminal on va faire appel à reserve_terminal de l'aéroport et on va juste regarder si il est
possible de renvoyer un chemin pour aller au terminal (si il y en a un de libre), sinon on renvoit un chemin vide.

4. On va juste ajouter une condition qui nous facilitera la tache, si notre fonction is_cycling renvoit vrai, ce qui
veut dire que l'avion attend d'etre placer sur un terminal ou si l'avion n'est pas détruis. Du coup on appel
notre fonction reserve terminal qu'on range dans une variable et si celle ci n'est pas vide on la met dans notre
waypoints (celui de l'avion).
Le problème c'est que des fois un avion va reserver un terminal et que entre les 2 il n'aura plus d'essence et sera
détruis, et du coup reservera un terminal sans jamais y sortir, du coup on va faire une methode dans Tower qui va desallouer
l'avion qui sera bloqué et on va aller l'appeler dans le destructeur de l'avion. Donc à chaque fois qu'un avion va
etre detruis on appelera cettre fonction, on verifi dans la fonction si l'avion a bien reservé un terminal, si oui on
recupérer le terminal correspondant au numero du terminal de l'avion qu'on recupère avec la map et on finis son service
avec la fonction finish-service() du terminal qu'on a recupéré. Et on oublie pas de le supprimer de la map des terminaux reservé.

C - Minimiser les crashs

On va faire une fonction dans move qui va verifier nos conditions pour trier le vaceteur et on va appeler sort dessus,
et appelé le tout au debut de la fonction. On va juste coder la fonction qui prendra 2 const de unique ptr d'avion
pour pouvoir les comparer et regarder si l'un d'entre 2 est dans un terminal, si c'est le premier alors on renvoit vrai
si c'est le deuxième on renvoit faux et sinon on renvoit le booléen que renvoit l'expression qui verifie si
le premiere avion a moins d'essence que le deuxième (en utilisant get_fuel).

D - Réapprovisionnement

1. On va faire notre fonction is low on fuel tout simplement comme demandé, et pour qu'il reste bloqué on renvoit
le resultat de cette en fonction (avec un ou) dans is_servicing de terminal.

2. On peut utiliser un algorithme, du coup on va faire un accumulate comme ce qu'on avait fais avant mais on compte
la somme d'essence des avions qui sont sur l'aéroport et que leur essence est faible.

3. On va mettre le aircraftManager en constructeur dans notre Airport pour pouvoir l'utiliser.

4. On va dire que fuel_stock est un unsigned int comme ca impossible qu'il soit negatif, puis on va
regarder si on à assez pour remplir notre reservoir et le faire, sinon on remplis avec ce qu'il reste
dans le reservoir.

5. On a juste a appelé la fonction refill sur l'avion courant si celui ci manque d'essence (is low on fuel)

6. On a juste à faire ce qui est demandé, juste pour la derniere, on va se faciliter la tache et modifier
le move de terminal pour qu'il appel refill_aircraft_if_needed dans le if et le move prendra en paramètre
le fuel_stock en référence et il appelera refilll_aircraft_needed. Sans oubler que le move des terminaux est
appelé juste après ce qui nous ai demandé avec fuel_stock. Mais quand on va détruire l'avion, au lieu de faire
un finish_service dessus car il ne rentrera pas dans la condition car nous l'avons changé pour bloqué les avions
dans le terminal pour qu'il se recharge. On va plus tot faire une fonction qui va passer l'avion courant du terminal
à null sans passer par une condition, et on remplace dans la fonction terminal_finish de tower la fonction finish_service
par celle qui passe à null l'avion courant (del_aircraft).

On remarque qu'on a un petit truc pas terrible qui détruis les avions quand ils atterrissent ou quand
ils vont vers le terminal ou quand ils décollent parce que ils sont détruis à ces moments la donc on va
faire en sorte que si ils ont pas assez d'essence pour aller au bout, il continuera à faire des tours
jusqu'à qu'il soit détruis.