Objectif 1 - Référencement des avions
A - Choisir l'architecture

Si on crée une nouvelle classe AircraftManager qui s'occupera de garder tout les avions, cela
sera plus pratique et plus lisible. On aura du coup une classe qui s'occupe d'un objet,
C'est la délégation, on va donner une "tache" à une classe pour s'occuper de celle ci, ca sera
plus propre. Mais si on donne ca directement à une classe existante, ca nous facilitera la tache car on
pourra l'appeler assez facilement dans notre code et on aura pas besoin de créer une nouvelle instance d'objet quand on aura besoin d'un avion.

B - Déterminer le propriétaire de chaque avion

1. le destructeur de aircraft, qu'on appel dans timer qui détruit le dynamic objet (l'avion car c'est le seul qui a un booléen qui change)
sans oublier qu'on l'enleve de la queue. Sans oublier que on change le booléen dans la fonction get_instructions.

2. La queue qui s'occupe de stocker les dynamicObject donc un avion. Après on a d'autres fonctions qui crée les avions,
qui les deplaces (move) etc...

3. On va mettre des consignes dans les destructeurs des classes qui s'occupent de stocker les avions (DynamicObjcet et Displayable)
donc on va erase l'avion dans cette liste en appelant le destructeur de l'avion qui appel aussi le destructeur de Displayable.

4. Car le aircraftManager s'occupera totalement des avions lui meme, donc on voudras surement garder les avions meme qui sont détruis car on voudras peut etre les relancer,
ceux qui fais que on veut pas les détruire de notre programme, on veut juste plus qu'ils apparaissent et bouges. On veut les garder au cas ou, on aura besoin d'eux.

C - C'est parti !

On va faire une classe qui s'occupera de stocker les avions et de les créer pour l'instant. On va faire un vecteur de unique_ptr d'avion car chaque avion à sa propre adresse unique, donc on va stocker de cette maniere les avions, on aura besoin de récupérer les fonctions de tower_sim et de les mettres dans notre aircraft_manager,
on va récupérer create_aircraft, create_random_aircraft, airlines (pour pouvoir créer les avions avec un nom) et aircraft_types qui est une variable globales qu'on est pas obligé de mettre dans la classe, on s'en occupera plus tard avec la factory.
On va du coup faire notre vecteur en privé, créer l'avion et le mettre dans notre vecteur et en utilisant move car on travaille sur des unique_ptr, il nous faut quand meme l'aéroport donc on va le mettre en parametre et le récupérer dans
TowerSim directement. Le soucis c'est que maintenant on peut arreter de les mettres dans la move_queue car on a deja les avions dans notre vecteur d'avion.
Donc pour faciliter la tache, on va faire une fonction move (sans oublier le override car c'est la fonction virtual de DynamicObject et que aircraftManager herite de celle ci) qui va bouger tout les avions de notre vecteur, donc on récupère la fonction timer et celle la deviens notre move dans AircraftManager, et timer reviens la meme que celle qu'on avait au tout départ,on adapte move de AircraftManager car on a des unique_ptr d'avion, on appel move sur les avions (notre dynamic objet dans notre ancienne fonction) et voila.
Sans oublier que TowerSim doit avoir un champs AircraftManager qu'on ajoute dans la move_queue (et qu'on appel pour ajouter des avions), et pour que cela marche bien il faut aussi faire en sorte que AircraftManager hérite de

DynamicObject pour pouvoir ranger celui ci dans la move_queue.

Objectif 2 - Usine à avions
A - Création d'une factory

On va faire une factory qui va prendre tout ce qui prend en compte la création d'avion, donc create_aircraft, create_random_aircraft, airlines et aircraft_types.
On va changer le type de retour de create_aircraft et create_random_aircraft car c'est aircraftManager qui s'occupe de supprimer ou d'ajouter des avions dans son vecteur.
Donc on va juste retourner un unique_ptr d'avion, pareil pour create_random et dans notre aircraftManager on va ajouter une fonction qui ajoute des avions dans notre vecteur avec move.
Elle prend en parametre un unique_ptr d'avion encore et move dans le vecteur celle ci. Rien ne change pour les fonctions et champs qu'on a mis dans la classe,
on retire le inline pour les champs car on veut plus de champs globales et on enleve le push_back dans create_aircraft. Puis on ajoute une méthode create_aircraft dans towersim qui va juste faire un add_aircraft sur l'aircraftmanager
qui va add un create_random_aircraft de notre factroy avec la factory qu'on a crée dans towersim.
Ensuite, on enleve dans aircraftManager les fonctions qu'on a mis dans la factory, et on crée un champs factory dans notre classe TowerSim.
Le problème c'est que on appel la factory avant le constructeur de towersim, donc on va faire une struct ContextInitializer et mettre dans son constructeur tout ceux qu'il y avait
dans le constructeur de TowerSim et créer 2 champs privé pour stocker le nombre d'argument et les arguments car on en a besoin, on va du coup prnedre ces 2 la en arguments du constructeur
de contextInitializer, et ranger dans nos champs privés les parametres et initialiser Mediapath, gl, et srand dans le constructeur. Puis dans le constructeur de TowerSim, on prend
un contextInitializer, donc on va créer un contextInitializer dans le main pour pouvoir bien tout créer avant de lancer la simulation et d'avoir des soucis, et après on le met dans
l'appel de la création de towerSim dans le main. Sans oublier de créer un champs privé qui represente ce contextInitializer dans Towersim et lui mettre dedans le contextInitializer mis en parametre.
Il y a juste help et create_keystrokes qu'on met dans le contructeur de Towersim car on peut les appeler que la et que c'est des champs et méthode de towersim, donc on oublie pas de faire un get
sur les champs privés de contextInitializer, pour définir help.

B - Conflits

On va faire un set des noms d'avions dans notre Factory pour vérifier s'ils existent ou pas en fonction de leurs noms qui sont des string.
Donc on a un set de string, car on peut pas vérifier en fonciton de leurs place en mémoire, parcre que meme si ils
ont le meme nom, en mémoire leur adresse est différentes. On va mettre la set en privé et faire une methode get qui renvoit cette set
et une méthode add_name qui ajoute le nom de l'avion en question dans la set, j'ai choisi aussi une set pour etre sur de n'avoir aucun
doublon. Après on va juste changer notre fonction, create_aircraft dans towersim qui va vérifier si on trouve le nom de l'avion
que l'on vient de créer dans la set, on n'ajoute ni le nom dans notre set, ni l'avion dans le vecteur de unique_ptr en affichant un message
qui explique que le nom a déjà été utilissé et on sort de create_aircraft (avec un return), sinon on peut ajouter.