PROJET THEORIE DES GRAPHES : SIMULATION DU CONTROLE AERIEN

Sommaire

1. Objectifs et planning du projet 2

2. Documentation PowerPoint (ou Prezi) à rédiger 3

3. Introduction du sujet 3

4. Première mission : système de contrôle aérien dans un réseau d'aéroports 4
   - Les paramètres locaux d'un aéroport et du réseau d'aéroports 4
   - Les paramètres locaux d'un avion 5
   - Mise en place du « monde » 5
   - Déroulement de la simulation 6
   - Visualisation et IHM de la simulation : 6

5. Missions plus ambitieuses 7
   - Perturbation météo sur une zone. 7
   - Eviter les collisions d'avions. 7
   - Fuite réservoir. 8
   - Etude des flots des avions 8
   - Crise mondiale énergétique 8
   - Ouvertures 9

6. Contraintes matérielles et logicielles 9

7. Critères d'évaluation 9
   - Barème 9

8. Modalités de dépôt sur campus 9

9. Annexe : exemple de visualisation de la simulation 10

1. Objectifs et planning du projet

Ce second projet informatique a cette fois pour but de travailler la **Théorie des graphes** développée dans le **langage orienté objet C++**, intégrée à un environnement console (pour la saisie et l'affichage) ou/et graphique**.**

Le projet en mode "piscine" de Théorie des graphes se déroule **dans la semaine du 4 avril 2022** (1 semaine complète). Le Cahier Des Charges (CDC) du sujet vous est fourni la veille du démarrage des longueurs de nage. Deux conséquences pour vous :

- Il vous sera nécessaire d'étudier attentivement et rapidement le CDC pour le décomposer en étapes, identifier celles que vous pouvez traiter dès maintenant et commencer à les réaliser (comme vue en cours avec la méthode Pert).

- Pour les autres tâches, vous aurez également à identifier les besoins et à rechercher les connaissances nécessaires pour y répondre. Pour cela, vous aurez à utiliser tous les outils à votre disposition (web, tutoriaux, livres…) pour identifier, comprendre et vous approprier les techniques utiles au jeu et qui ne vous auraient pas été encore enseignées. Ainsi, à l'occasion des futurs cours, vous aurez les idées claires, un besoin précis, de nombreuses questions, et surtout une capacité accrue de comprendre la théorie enseignée.

Ce projet est à faire de **2 à 4 étudiants par équipe du même groupe de TD, telle que mentionnée dans le lien** Drive des équipes du projet Théorie des graphes

Afin de répondre à vos questions, un « **suivi de projet** » pourra se faire sur le serveur discord du tutorat https://discord.gg/epB96JA et son salon textuel **#ing2-theorie-graphes-cplusplus** qui se trouve dans la catégorie « COURS » de ce serveur.

Le livrable sera à déposer sur le lien Livrable à déposer : code en C++ et documentation PowerPoint (ou Prezi) deadline le 10 avril qui se trouve dans la section de la page campus Cours : Projet info4 Théorie Graphes, Section : Livrable à déposer (campusonline.me). Comme indiqué dans cette section : « 2 pts de pénalité sur la note globale de projet par heure de retard. Vérifiez AVANT la soutenance que la version déposée fonctionne. »

Les **soutenances** auront lieu **la semaine du 11 avril 2022.** Elles seront de **15 minutes au maximum par équipe et 5 minutes de questions posées par le jury**. Chaque équipe présentera son projet déposé sur campus avec le vidéoprojecteur au créneau fixé sur le Drive des équipes du projet Théorie des graphes. Chaque étudiant d'une équipe doit intervenir lors de la présentation orale et avoir une connaissance globale du projet de l'équipe. Le jury pourra interroger l'équipe et individuellement certains étudiants. Un étudiant trop hésitant sur une question le concernant ou n'ayant pas une vision globale du projet pourra se voir pénaliser par rapport à ses coéquipiers. Tous les étudiants du groupe **doivent être présents sous peine d'être pénalisés par 0 au projet, toujours selon Pert** ☺

2. Documentation PowerPoint (ou Prezi) à rédiger

Votre documentation PowerPoint (ou Prezi) d'environ 10 slides, à rédiger et présenter lors de votre soutenance devra contenir les slides suivants :

1) Page de garde avec titre, groupe de TD, numéro d'équipe et noms (1 slide)

2) Sommaire (1 slide)

3) **Diagramme de classes** (1 slide) avec l'outil Draw.io ou équivalent, et présentant les attributs, les méthodes, les cardinalités, MAIS sans constructeurs ni getters/setters

4) **Versioning GIT en bonus** : screenshot montrant la bonne utilisation et répartition des tâches du code entre coéquipiers. (1 slide)

5) **Méthodologie de votre travail** : (environ 5 slides)
   - tableau de la répartition des tâches et **l'ordonnancement du projet avec la méthode Pert**
   - présentation synthétique de vos approches (inspirez-vous des slides du cours avec des algos courts et "en français")
   - tableau comparatif de la complexité des différents algorithmes
   - La mise à jour commentée et argumentée des fichiers avec des exemples concrets
   - remarques, schémas, diagrammes, formules mathématiques justifiées, screenshots annotés : tout ce qui peut permettre de rendre votre présentation informative, efficace et dynamique
   - présentation claire, concise et précise du travail réalisé
     -> qu'avez-vous entrepris ? -> comment ? -> quels sont les résultats ?

6) **Bilan individuel et collectif sans blabla** de l'état du travail effectué. (1 slide)

7) **Bibliographie précise de toutes vos sources (web, livres etc.)**. (1 slide). Toute source non citée est considérée de facto comme un plagiat

3. Introduction du sujet

Le but de votre projet est de réaliser un simulateur de trafic aérien.

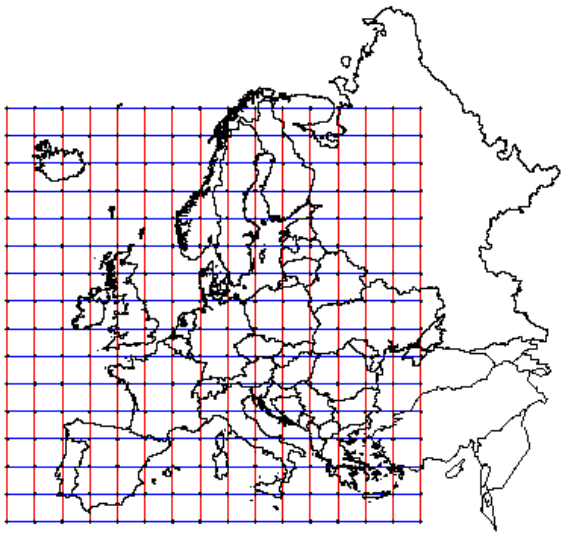
Ce simulateur devra permettre de suivre les vols et les stations au sol d'un certains nombre d'avions et de se confronter aux problèmes de contrôle aérien. Il mettra en pratique quelques algorithmes bien sentis de théorie des graphes afin d'apporter une solution à cette tâche complexe.

La réalité d'aujourd'hui

Selon Rivière et Brisset [1]

L'augmentation du trafic aérien dans les aéroports pose un réel problème puisqu'il faut simultanément gérer un nombre toujours croissant d'avions dans un espace aérien déjà saturé et maintenir un haut niveau de sécurité.

Chaque aéroport dispose de son propre contrôle aérien lui permettant de surveiller son propre espace aérien, et éventuellement de le réguler. Le contrôle est effectué « par secteur aérien », c'est-à-dire qu'une paire de contrôleurs contrôle un nombre limité d'avions dans un volume d'espace restreint dont ils ont la charge. La grille de la figure suivante représente un réseau aérien européen, chaque case correspondant à un secteur aérien.



4. Première mission : système de contrôle aérien dans un réseau d'aéroports

Le simulateur qu'il vous est proposé de réaliser ici ne reprendra pas toutes les tâches du contrôle aérien décrit ci-dessus.

Vous êtes en charge de simuler un système de contrôle, via un écran de contrôle (IHM), en prenant en compte certains paramètres utiles pour le contrôleur aérien local à chaque aéroport et pour le suivi du déplacement des avions.

Les paramètres locaux d'un aéroport et du réseau d'aéroports

Les paramètres locaux suivants de chaque aéroport et du réseau des aéroports, si nécessaire à compléter, doivent être stockés dans un fichier :

- Le **nom de l'aéroport**.
- La **localisation de l'aéroport** (GPS ou autre comme ses coordonnées sur votre plan graphique tel que fourni en annexe).
- Le **nombre de pistes**. Elles servent toutes au décollage et à l'atterrissage.
- Le **nombre de places au sol** (station), permettant de parquer un avion.
- Le **délai d'attente au sol** pour un avion parqué, le temps de charger et décharger passagers et bagages.
- Le **temps d'accès aux pistes**, « station vers piste » ou « piste vers station ».
- Le **délai anticollision**, permettant à deux avions de se suivre sur la voie d'accès aux pistes mais pas de s'y trouver en même temps.
- Le **temps de décollage ou d'atterrissage**. Un seul avion sur la piste durant ces opérations.
- La **durée de la boucle d'attente en vol**, permettant à un avion dont le carburant le permet de se mettre en attente d'une station en tournant en boucle au dessus de l'aéroport.
- La **distance** aux autres aéroports.

Contraintes :
Ne s'engagent dans la phase d'atterrissage que des avions certains de pouvoir trouver une place de stationnement.

Les paramètres locaux d'un avion

Les paramètres locaux suivants des avions, si nécessaire à compléter, doivent être aussi stockés dans un fichier :

- Le **type d'avion** : court, moyen ou long courrier quel que soit le trajet emprunté, y compris avec des escales même pour les avions de court courrier.
- Sa **consommation** : à vous de fixer une consommation cohérente selon le type d'avion, en nombre de litres pour 100 km parcourus et selon la vitesse moyenne de parcours.
- Sa **capacité carburant** : à vous de fixer cette capacité en nombre de litres qui doit être cohérente, selon le type d'avion.

Contraintes :
La **vitesse de vol en situation normale** sera considérée **constante** durant la quasi-totalité du vol (même si c'est une hypothèse peu probable), mais variable au décollage et l'atterrissage. Elle correspondra en moyenne à **200 km parcouru en une Unité de Temps UT (un carré sur le schéma en annexe)**. Une **UT correspond à un tour de votre simulateur** et devra être calé sur **2 secondes au minimum** afin de voir l'évolution des différents appareils en jeu.
La **vitesse de vol en boucle d'attente** autour d'un aéroport sera **la moitié** de la vitesse normale. La consommation sera également **réduite de 20% (80 litres pour 100 km)** dans cette situation.

- *Les informations présentes dans les fichiers servent de « base de données » à votre plan du site. Tout en les respectant, vous devrez les enrichir à chaque nouvelle règle : par exemple, les informations fournies dans le déroulement de la simulation qui suit.*
- *Ces informations pourront être modifiées de manière interactive par l'utilisateur grâce à une interface console ou/et graphique évènementielle.*

Toute autre information que vous jugez nécessaire peut être ajoutée dans ces fichiers. Par exemple, vous pouvez ajouter si vous le souhaitez les coordonnées de chaque point de votre plan graphique.

Mise en place du « monde »

- Dans un premier temps, vous devez afficher un menu en mode console (pour ne pas perdre de temps inutile avec le graphique juste pour un menu) présentant toutes les options liées aux fonctionnalités qui vont suivre tout au long de ce sujet aérien.

- Puis vous devez charger les paramètres des fichiers du réseau d'aéroports et des types d'avions mentionnés au-dessus. Puis vous devez calculer les plus courts chemins inter-aéroports, en fonction de ces paramètres.

- Une fois ces premières fonctionnalités mises en place ci-dessus, vous devez effectuer celles qui suivent :

1) Permettre d'afficher la liste des aéroports et les différents types d'avion avec leurs caractéristiques, ceci en mode console seulement (toujours pour ne pas perdre de temps avec le graphique pour ces informations générales).
2) Proposer une visualisation graphique du réseau d'aéroports, en vous inspirant du schéma en annexe.
3) Proposer de calculer, d'afficher et de visualiser graphiquement sur le réseau l'arbre des plus courts chemins (PCC) pour un type d'avion à partir d'un aéroport. Le choix de l'aéroport de départ et du type d'avion seront à saisir par l'utilisateur en mode console.

Remarques importantes : Bien entendu, le calcul des PCC doit tenir compte de la capacité du réservoir du type d'avion choisi et donc déterminer les escales éventuellement nécessaires. Certains aéroports ne sont peut-être pas accessibles (par exemple, aéroports fermés pour cause d'intempéries, de guerre...)

Déroulement de la simulation

1. A l'initialisation, le simulateur **remplit aléatoirement ou via un module de paramétrage stocké dans un fichier** chaque aéroport avec des avions de différents types (selon les possibilités de l'aéroport). Il gère ensuite tous les avions en vol, les ajoute et les enlève des files d'attentes des différentes pistes et stations des aéroports ... Lors de la simulation, le simulateur fait à chaque fois le tour des aéroports pour faire décoller/atterrir et il fait avancer les avions en vol. Certaines stations peuvent bien sûr rester vides afin d'autoriser l'atterrissage d'avions se présentant à l'aéroport.

2. Avant toute éventuelle modification de paramétrage, vous **proposerez en soutenance une démonstration prenant en compte les paramètres d'aéroport et d'avion de vos fichiers, tels que présentés et visualisés graphiquement sur le schéma en annexe de la dernière page**.
   Conseil :
   Commencez par tester votre simulateur avec un seul avion. Puis avec un avion par aéroport…

3. Pensez aux caractéristiques d'un vol : un vol a un aéroport de départ, un aéroport d'arrivée, une heure de départ. Il faut établir son plan de vol (correspondant au PCC jusqu'à son aéroport d'arrivée) ...

4. Dans un second temps, la boucle de simulation vous permet de faire vivre votre simulateur et d'observer le bon déroulement des vols et la gestion des aéroports. Chaque tour de boucle correspond à une UT (quelques secondes par UT mais trop longtemps pour avoir le temps de visualiser lors de la démonstration en soutenance).

   A chaque UT, vous déterminerez aléatoirement les avions décollant de chaque aéroport, ainsi que leur destination parmi les destinations accessibles (via un trait noir sur le schéma en annexe). Vous serez vigilant, en fonction du type d'appareil, à **déterminer le plus court chemin viable** pour atteindre la destination. Il est hors de question de laisser décoller un avion dont la capacité de vol ne permettrait pas de mener à bien un tronçon du trajet.

A chaque escale, vous considérerez que les avions refont le plein de carburant. A chaque UT, vous symboliserez la position de chaque avion.

   Vous déterminerez (paramètre éventuel stocké dans un fichier ou valeur fixe dans votre code) le nombre d'UT d'une simulation (une centaine par exemple).

Visualisation et IHM de la simulation :

La visualisation et votre IHM (Interface Homme-Machine) de la simulation seront à réaliser en partie **en mode console et en partie en mode**.

Pour que vous ne perdiez pas trop de temps concernant l'affichage du menu, des listes d'aéroports, de types d'avion et du contenu des aéroports, ils doivent être affichés en mode console. De même, toute saisie (aéroport de départ, type d'avion...) doit être blindée et se faire en console. Par contre, la visualisation du réseau, des PCC et des vols en cours de simulation doit se faire en mode graphique avec **une librairie graphique compatible C++** (par exemple, Allegro ou autre librairie graphique à préciser lors de la soutenance).

Cette séparation de visualisation et IHM entre mode console et mode graphique vous permettra d'au moins montrer le mode console, surtout si la visualisation graphique ne fonctionne pas le jour de la soutenance. Preuve que le moteur du jeu en mode console est opérationnel et que votre code est **modulaire (séparation du moteur en mode console et de la visualisation de la simulation en mode graphique).**

Votre IHM devra permettre à l'utilisateur de paramétrer un certain nombre de contraintes pour la phase d'initialisation (tout peut être aléatoire au début).

Au lancement de la simulation, votre simulateur devra nous permettre de suivre le déplacement des avions en vol, mais aussi au sol et d'en visualiser ou masquer la valeur des paramètres. Un code couleur pourra éventuellement traduire certains états d'un aéroport et d'un avion, mais tout autre mode de représentation des états est autorisé (texte, info bulle sur un aéroport, un avion…). L'utilisateur ne devrait pas intervenir une fois la simulation démarrée, sauf éventuellement pour déclencher une situation d'urgence. A vous d'en gérer le protocole **à préciser dans la documentation et la démo, lors de la soutenance**.

Chaque tour de boucle du simulateur devra permettre à tous les avions d'évoluer d'une UT. Veillez à ce qu'une UT dure quelques secondes, mais pas trop longtemps quand même, pour nous laisser le temps d'observer l'évolution de l'ensemble.
Par exemple, à chaque UT les avions en vol normal évoluent d'un carré sur le schéma en annexe. Dans le cas où votre simulateur utiliserait un mode graphique différent de celui suggéré sur le schéma ci-dessous, vous veillerez à représenter les changements à chaque UT.

Les avions pouvant avoir chacun une altitude différente des autres, un même espace aérien (un carré sur le schéma en annexe) peut contenir plusieurs avions. **Il faudra éviter les collisions entre avions : voir le chapitre plus bas Eviter les collisions d'avions.**
Le schéma en annexe ne propose pas de visualisation de la gestion des aéroports (boucle d'attente, accès aux pistes, station…). A vous d'imaginer le meilleur moyen pour tout visualiser !

5. Missions plus ambitieuses

Lorsque votre simulateur sera suffisamment performant sur la première partie, vous pourrez gérer des évènements inattendus venant perturber le bon déroulement des vols.

Perturbation météo sur une zone.

A la suite d'un violent orage, les avions de cette zone ne peuvent maintenir leur vitesse de vol. Ils passent alors en situation d'urgence. Leur vitesse et leur consommation s'en trouvent réduites. A vous de déterminer de manière cohérente cette réduction en fonction du type d'avion, de la gravité de la turbulence et tout autre critère qui vous semble réaliste. Si l'avion doit contourner sa trajectoire, pour cause de trop forte turbulence, vous devrez recalculer les chemins en tenant compte de la baisse de vitesse et de consommation quand il rentre dans la zone.
Vous déterminerez le moyen de permettre à l'utilisateur ou au générateur aléatoire d'événements de définir la zone concernée (ensemble connexe de carrés sur le schéma en annexe par exemple).

En cas de déviation de la zone, vous devrez appliquer **l'algorithme de A*** en considérant que dans le schéma de l'annexe, chaque case correspond à une déviation possible de la turbulence. Il faudra minimiser le nombre de cases parcourues lors de cette turbulence, en tenant compte de la baisse de vitesse et de consommation quand il rentre dans la zone. A vous de déterminer une heuristique cohérente pour le poids des cases où se trouvent la turbulence. Ce choix d'heuristique sera à préciser et à justifier lors de la soutenance.

Eviter les collisions d'avions.

Afin de limiter les risques de collision entre avions, une altitude de vol différente doit être attribuée à des avions dont les trajectoires se croisent (à + ou - de km, et + ou - unité de temps, marges à déterminer).

Autre cas plus complexe et pas obligatoire à intégrer dans votre simulateur :
Les avions ne sont pas forcément sur la même trajectoire, mais l'un d'eux ralentit ou change d'altitude (par exemple à cause d'une perturbation météo) qui risque de percuter un autre avion.

Pour éviter ce genre de collision, on peut implémenter l'un des algorithmes de coloration **naïf ou de Welsh & Powell**.

Fuite réservoir.

La consommation s'en trouve doublée. Un algorithme de décision peut alors mettre l'avion en situation d'urgence (la vitesse plus faible réduit alors la consommation par exemple de 20% mais la fuite la multiplie par exemple par 2, on a donc un avion qui vole moins vite mais qui consomme plus qu'un vol normal) ou maintenir l'avion en vol normal (tout en consommant par exemple 2 fois plus).

La survenue de ces imprévus vous amènera à **rechercher l'aéroport le plus proche** de la position actuelle de l'avion sur la grille, vous permettant si possible une escale de sécurité. Malgré cette escale vous veillerez à acheminer l'avion jusqu'à sa destination initialement prévue.

Si ces événements inattendus ne permettent pas à l'avion de terminer sa course, votre simulateur devra malheureusement nous en symboliser le crash !

Vous maintiendrez la visualisation des crashs jusqu'à la fin de la simulation.

Etude des flots des avions

Vous avez encore rempli brillamment la mission qui vous a été confiée pour cette simulation ! ☺

Ce simulateur à l'usage des contrôleurs aériens que vous avez réalisé fonctionne. On souhaite optimiser le nombre de vols inter-aéroports.

Pour cela, on peut étudier les vols inter-aéroports dont il faudrait augmenter le nombre de vols (capacité) par heure entre 2 aéroports.

Le but est d'optimiser le flot de vols circulant entre deux aéroports. Puis connaissant un flot réalisable entre deux aéroports, il sera sans doute intéressant de savoir trouver un plus court chemin dans le graphe d'écart correspondant ou les plus longs circuits en termes de vols ou encore le plus long vol

- Ajoutez ces capacités maximales de vol inter-aéroports dans un fichier.
- Ajouter le flot dans vos classes.
- Ecrivez l'algorithme de résolution du flot maximum : cet algorithme permet de calculer l'augmentation possible du flot d'avions entre deux aéroports.
- Affichez le graphe d'écart des chemins trouvés en testant la valeur du flot de chaque arc du graphe réel par rapport à sa capacité. Puis calculez et affichez le plus court chemin en nombre d'arcs entre deux points dans le graphe d'écart.
- Je fais confiance à votre créativité et à votre esprit de futurs ingénieurs pour enrichir votre simulateur aérien.

Crise mondiale énergétique

Une crise mondiale du pétrole réduit très fortement le nombre d'avions pouvant voler. Certains aéroports sont même obligés de fermer, créant des zones d'exclusion aérienne. Pour réduire au maximum la consommation de kérosène des rares avions qui peuvent voler, il faut minimiser la distance parcourue dans l'ensemble des aéroports parcourus avec escales.

Pour minimiser cette distance parcourue inter-aéroports, vous appliquerez le fameux algorithme de **Kruskal**.

Ouvertures

Pour les plus à l'aise d'entre vous, il sera possible de laisser à l'utilisateur le paramétrage de certains critères actuellement fixés ci-dessus (consommation des appareils par exemple, vitesse non-constante…).

Vous pourrez également nous inventer quelques événements surprenants, comme un détournement d'avion par exemple ;-)

Vous pourrez mettre en place un historique des événements pour chaque avion (lieu et date de décollage, atterrissage, événements inattendus…).

6. Contraintes matérielles et logicielles

Langage orienté objet C++ en mode console ou/et graphique compatible C++ sur CodeBlocks.

Méthode de conduite de projet : voir le chapitre Objectifs et planning du projet

Résolution graphique : compatible avec les vidéoprojecteurs de l'école donc **vérifiez avant la soutenance**.

7. Critères d'évaluation

Votre travail sera jugé sur les critères suivants :

- Le respect rigoureux des règles du projet énoncé précédemment.
- L'analyse chronologique et descendante des traitements afin de mieux structurer votre code orienté objet en C++, avec des classes et des méthodes correctement paramétrées.
- La bonne répartition des tâches entre les membres de l'équipe.
- L'intérêt, l'originalité, la jouabilité et toutes les caractéristiques que vous prendrez soin de mettre en avant lors de la soutenance.

Barème

- **Démo sur 20 points coefficient 2**
- **Documentation PowerPoint (ou Prezi) sur 20 points coefficient 1**

8. Modalités de dépôt sur campus

Un livrable à déposer **en respect des consignes spécifiées sur campus** dans la section de la page campus Cours : Projet info4 Théorie Graphes, Section : Livrable à déposer (campusonline.me).

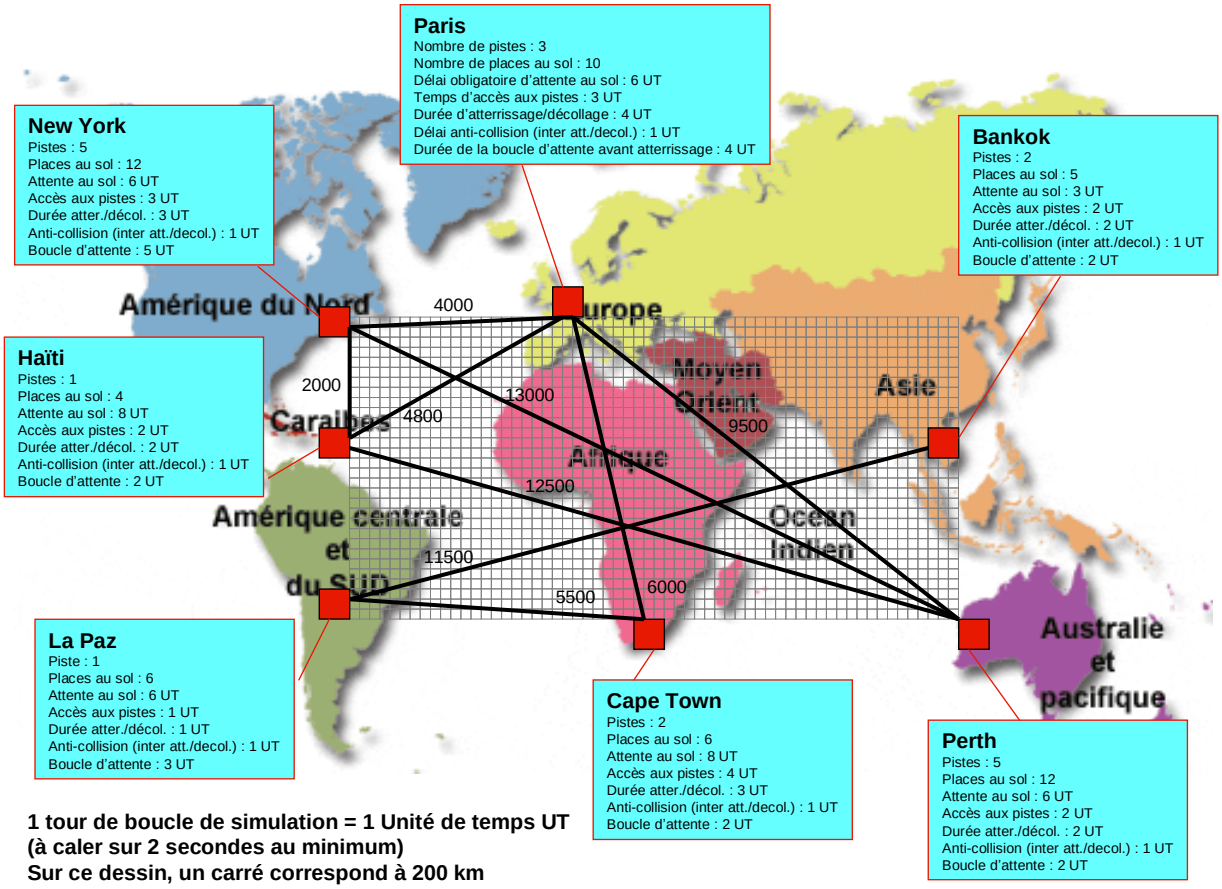
Dans le lien Livrable à déposer : code en C++ et documentation PowerPoint (ou Prezi) deadline le 10 avril : comme nom d'archive les noms de votre équipe sous la forme "nom1-nom2-nom3-nom4.zip ou .rar" (exemple : fornier-rebai-segado.zip), avec un seul dépôt par équipe, contenant le code C++ et la documentation PowerPoint (ou Prezi). Capacité maximale de dépôt : 1Go.

Ne déposez pas à la dernière minute !!! Tout retard sera pénalisé de -2 points par heure de retard à partir du lendemain. Tout plagiat sera sévèrement pénalisé par 0 et un avertissement.

Bon vol à tous en avion ou en hélicoptère ☺

JPS avec des remerciements particuliers à Mme Palasi et M. Debize responsable pédagogique des ING2 à ECE Lyon pour leurs conseils constructifs à votre attention, chers nageurs dans le ciel ☺

9. Annexe : exemple de visualisation de la simulation



1 tour de boucle de simulation = 1 Unité de temps UT
(à caler sur 2 secondes au minimum)
Sur ce dessin, un carré correspond à 200 km