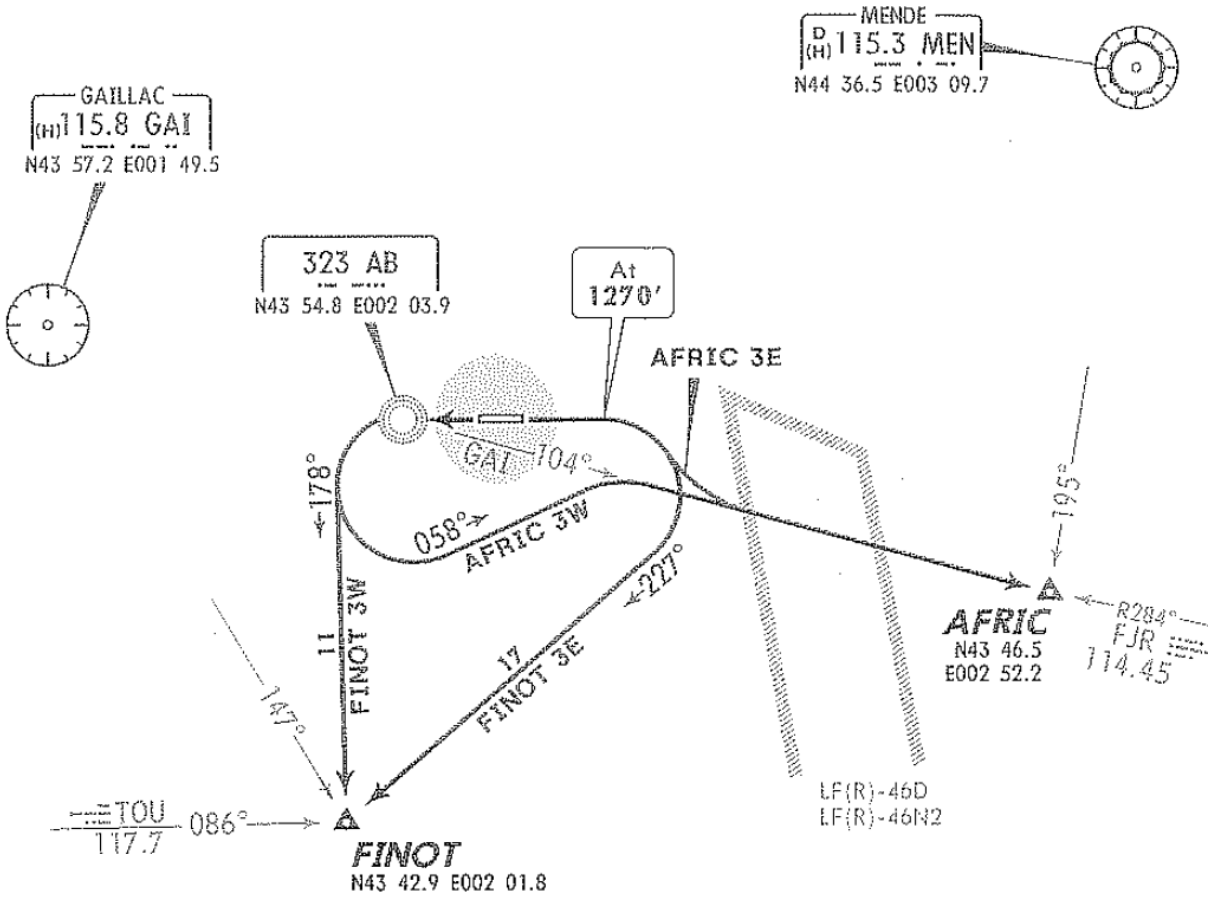
Construction de plans de vol

Développez un outil graphique de construction de plan de vol



Le plan de vol est obligatoire pour un vol aux instruments, mais aussi dans un certain nombre de cas pour des vols à vue.

On se propose ici de réaliser un outil graphique permettant de construire la trajectoire associée à un plan de vol, y compris les phases de départ et d'approche.

Sur les avions de ligne, le FMS (Flight Management System) récupère un plan de vol déjà enregistré dans sa base de données, et construit la trajectoire correspondante. Le FMS permet aussi de construire un plan de vol manuellement à partir de saisies sur le clavier alphanumérique d'un Control Display Unit du FMS : cette saisie est facilitée par le fait que le FMS connait déjà tous les points de route (waypoints) dans sa base de données, et qu'il suffit alors d'indiquer les éléments de la route grâce à un nombre réduit de "legs" reliant les points de route. L'interface est malgré tout très peu ergonomique, mais rarement utilisée, car le FMS est un énorme logiciel (plusieurs millions de lignes de code) qui va calculer une trajectoire suivant le plan de vol tout en prenant en compte de nombreuses contraintes avion et en tentant d'optimiser la consommation de carburant ou le bruit...

La construction d'un plan de vol par un outil graphique pourrait donc être intéressante pour l'aviation légère : quelques applications sur tablette commencent à arriver, mais elles ne sont pas encore très répandues.

Spécifications attendues de l'outil

Construction interactive de plan de vol

On veut que l'outil permette de construire un plan de vol délimité par des "Fix" (points de route) et des "Legs" reliant ces Fix.

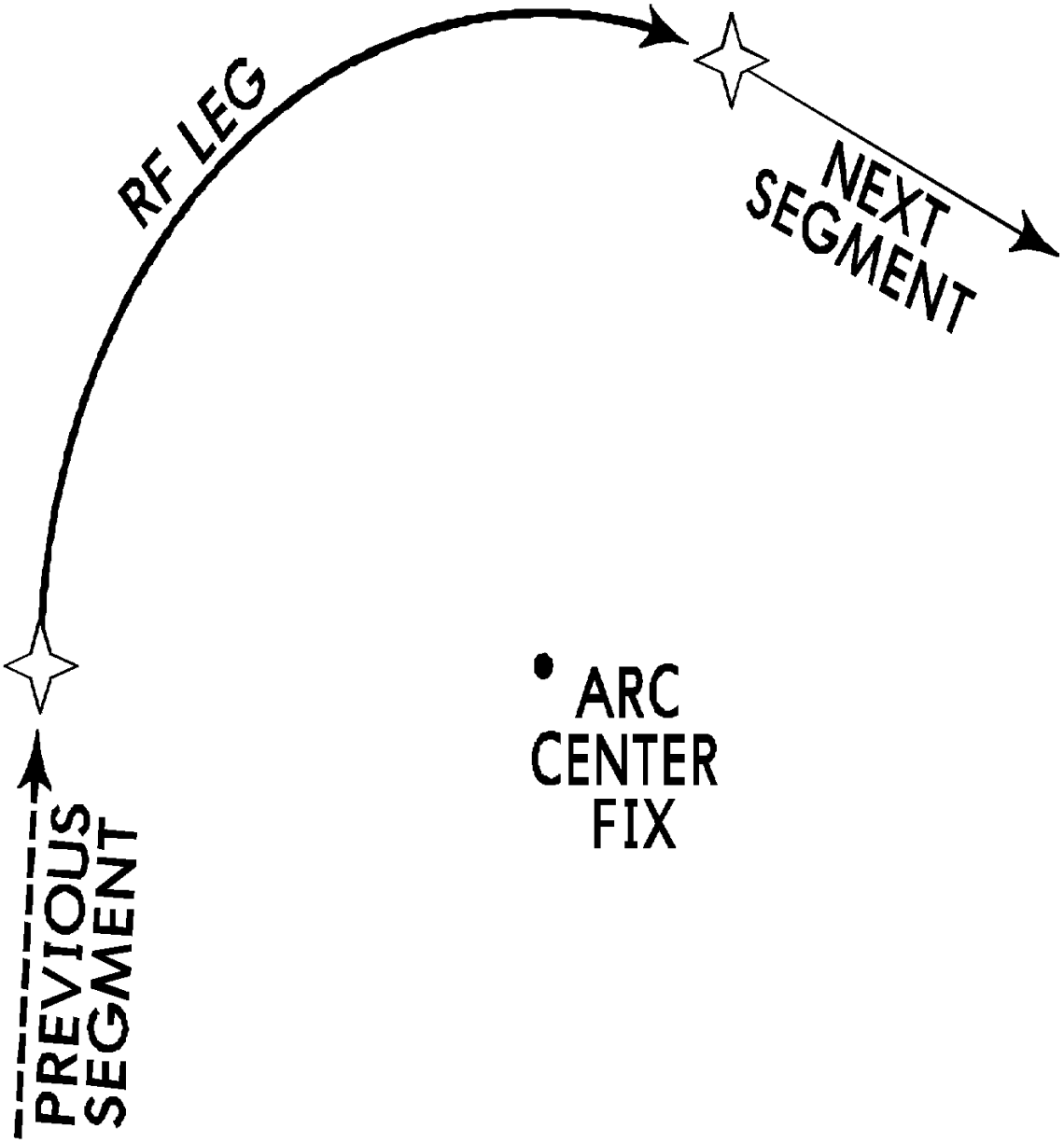
L'interface permettra donc de définir un nouveau Fix de deux façons :
- soit en entrant ses coordonnées (ex: N43 42.9 E002 01.8 pour le point "FINOT")
- soit par un simple clic dans la zone graphique

Dans les deux cas, l'interface permettra de saisir le nom du point et un type permettant de lui associer un symbole graphique (station sol par exemple).

Il existe 23 types de legs définis par la norme ARINC 424, mais on pourra se limiter ici à un sous-ensemble de ces legs :

- TF (Track to Fix) : c'est un leg qui relie directement deux Fix. On veut pouvoir créer ce leg TF simplement en cliquant d'abord **près** d'un premier point de route, puis près d'un second. Ce type de leg permet de créer très rapidement un plan de vol passant par différents points de route.

- RF (Radius to Fix) : c'est un leg en arc de cercle reliant deux Fix, à même distance d'un centre. La saisie se fera en cliquant successivement près du premier point, puis du second, puis un point près de la médiatrice des deux points (on prendra comme centre du cercle le point de la médiatrice le plus proche du point cliqué).



- DF (Direct to Fix) : c'est un leg qui permet de rejoindre directement un Fix. Ce leg semble donc identique au leg TF mais vous trouverez une différence lorsque vous aurez lu la section suivante...
- CF (Course to Fix) : c'est un leg qui permet de rejoindre un Fix avec un cap donné. La saisie se fera en cliquant près du point à rejoindre, puis en entrant au clavier le cap souhaité. Notez que ce leg ne dit pas comment rejoindre la route de cap donné si le leg précédent se terminait sur un point de route qui n'est justement pas sur la route... Ce défaut de raccordement est traité dans la section qui suit...

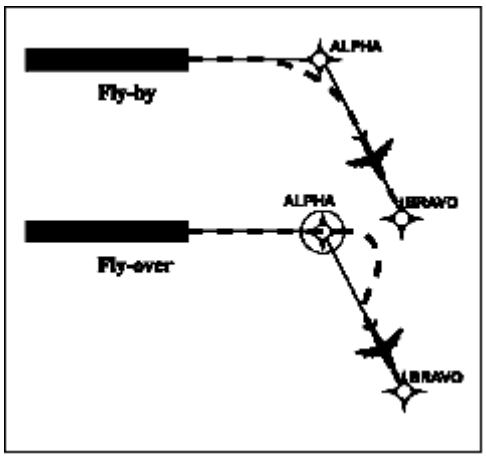
Calcul d'une trajectoire réaliste

Un avion ne peut pas tourner en suivant des angles vifs, on précisera donc pour chaque point de route du plan de vol s'il doit être passé en "fly-by" ou en "fly-over" (par défaut ce sera toujours "fly-by", on indiquera explicitement lorsqu'on veut passer sur un point de route (fly-over)).

Dans le cas "fly-by", la trajectoire ne passera pas sur le point de route : elle "coupera" le virage en entamant (avant d'arriver au point de route) un arc de cercle qui permettra de rejoindre ensuite le segment allant vers le point de route suivant. L'arc de cercle aura un rayon de courbure réaliste pour l'avion (un virage normal pour l'avion).

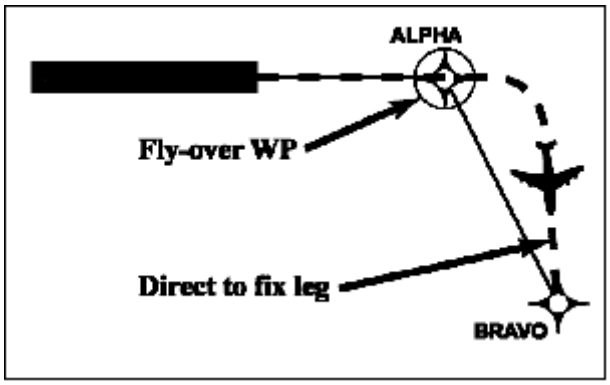
Dans le cas "fly-over", la trajectoire passera bien sur le point de route, et tournera **à partir** de ce point de route (toujours avec un rayon de courbure réaliste pour l'avion) pour rejoindre le segment suivant.

La figure suivante illustre donc les deux trajectoires résultantes lorsque le leg qui suit est un TF (Track to Fix) :



Notez que dans le second cas, il faut calculer deux virages pour rejoindre sans angle le leg TF qui suit !

On peut maintenant voir la différence entre un leg TF et un leg DF (Direct to Fix) : dans le cas d'un leg DF, on ne cherchera pas à rejoindre le segment, mais à la place on partira directement vers le point de route suivant. C'est ce qui est montré sur la figure suivante :



Enfin, vous noterez que pour les legs CF, il y aura plusieurs cas à traiter suivant la position du point d'intersection de la demi-droite définie par le leg CF avec le leg précédent (éventuellement prolongé). Evidemment, s'il se trouve que ce point d'intersection est exactement le point de route (ce qui veut dire que le cap indiqué dans le leg CF correspond effectivement à la direction entre les deux Fix), alors on a les mêmes cas de figure que pour le leg TF. Par contre, si cette intersection se trouve avant le passage du point de route, alors il faudra amorcer le virage plus tôt pour un point de route "fly-by", tandis qu'il faudra faire un virage plus important après un point de route "fly-over", puisque l'intersection est déjà passée. Inversement, si l'intersection se trouve dans le prolongement du premier leg, il faudra considérer cette intersection comme le vrai point de route intersection des deux legs...

Fonctionalités d'affichage

Le logiciel permettra de zoomer et dézoomer sur le plan de vol afin de bien montrer la trajectoire précise avion (notamment les courbures).

Maintenance d'une "base de données" de points et de plans de vol

A la fermeture du programme, on sauvegardera automatiquement sur fichier tous les points déjà entrés, de sorte qu'au démarrage du programme ces points soient rechargés en mémoire et puissent ainsi être utilisés pour créer des trajectoires.

De même, le programme permettra de gérer une petite "base" de plans de vol : il permettra de nommer un plan de vol créé et de le sauver dans un fichier, et inversement de recharger un plan de vol déjà sauvé.

Parties optionnelles

On pourra chercher à rendre le logiciel compatible avec les définitions textuelles de plans de vol : taper au clavier un plan de vol tel qu'il apparaît sur le document officiel de plan de vol construira le plan de vol et la trajectoire correspondante dans l'outil.

On pourra aussi implémenter plus de types de legs : par exemple les legs de caps (Heading to Distance, Heading to Intercept next leg, Heading to Intercept Radial, Heading to manual termination...) ou les legs de route (Course to Distance, Course from Fix to distance, Course from Fix to distance DME, Course from fix to Manual termination...).

Eventuellement, on pourra ajouter la prise en compte de la trajectoire dans le plan vertical...

Quelques indications pour le développement de votre programme

Le graphisme en 2D avec Java

La classe ImageN6K fournie à l'occasion des TPs tels que l'affichage de l'ensemble de Mandelbrot est une classe fournissant un accès simplifié au dessin 2D en Java. Vous pouvez l'utiliser mais vous aurez sans doute besoin de vous plonger dans la bibliothèque de dessin Java 2D pour obtenir une représentation graphique plus évoluée. Un moyen simple de se lancer dans le dessin sans avoir besoin de comprendre le fonctionnement complet des IHM

(Interfaces Homme-Machine) est d'utiliser la classe ImageN6K pour créer votre fenêtre graphique et d'ensuite récupérer un objet Graphics2D au moyen de la méthode getGraphics( ).

Avec cet objet vous pourrez ainsi accéder aux fonctionnalités de dessin de la classe java.awt.Graphics pour faire des dessins basiques (segments, arcs, écriture de texte), mais aussi aux fonctionnalités plus évoluées de la classe java.awt.Graphics2D. Cette dernière classe est un peu plus délicate à maîtriser à cause des nombreuses notions de dessin implémentées, mais si vous passez un peu de temps à comprendre la documentation, le rendu final en vaut la peine : vous pourrez ainsi contrôler le type de crayon utilisé pour le dessin (classe Stroke), définir des formes (classe Shape) ou utiliser les formes de base déjà existantes dans le paquetage java.awt.geom, faire des transformations de repère (translations, rotations, homothéties) avec la classe java.awt.geom.AffineTransform, etc.

Les accès fichiers en Java

Pour simplifier votre travail de sauvegarde et de récupération de données dans des fichiers, vous pouvez vous contenter d'utiliser des fichiers de données primitives qui seront lus ou écrits en totalité à chaque usage. De tels fichiers (séquentiels) mémorisent les informations dans l'ordre où vous les avez écrits. Par exemple, si vous avez un fichier qui contient une "base de données" de points (Fix), vous pourrez utiliser une boucle for pour écrire tous vos points dans le fichier.

Par exemple :

```
Fix [ ] fixDataBase; // le tableau qui contient tous vos points Fix
DataOutputStream fichierDePoints = new DataOutputStream(new
File("MyFixDataBase.dat"));
fichierDePoints.writeInt( fixDataBase.length ); // écrit d'abord le nombre
de points, ce sera pratique pour les relire
for (int i=0; i<fixDataBase.length; i++)
 fixDataBase.ecrireDans(fichierDePoints);
```

avec une méthode d'écriture de point Fix telle que :

```
public void ecrireDans(DataOutputStream fichier) throws IOException
{
    for (int i=0; i < TAILLE_NOMS_FIX ; i++)  // en supposant que les noms
des Fix ont une longueur fixe
        fichier.writeChar( nom.charAt(i) );
    fichier.writeDouble( longitude );
    fichier.writeDouble( latitude );
    fichier.writeInt( typeDeFix );
}
```

Pour la lecture vous pourrez procéder de même avec les méthodes de la classe DataInputStream.