
# START'AIR SAFE PROJECT

## AF 10/04/2014

Par corentin RICOU & William BOBO

# Table des matières

Introduction 1
- les besoins 1
- platforme de travail 2

Conception 3
- Architecture 3

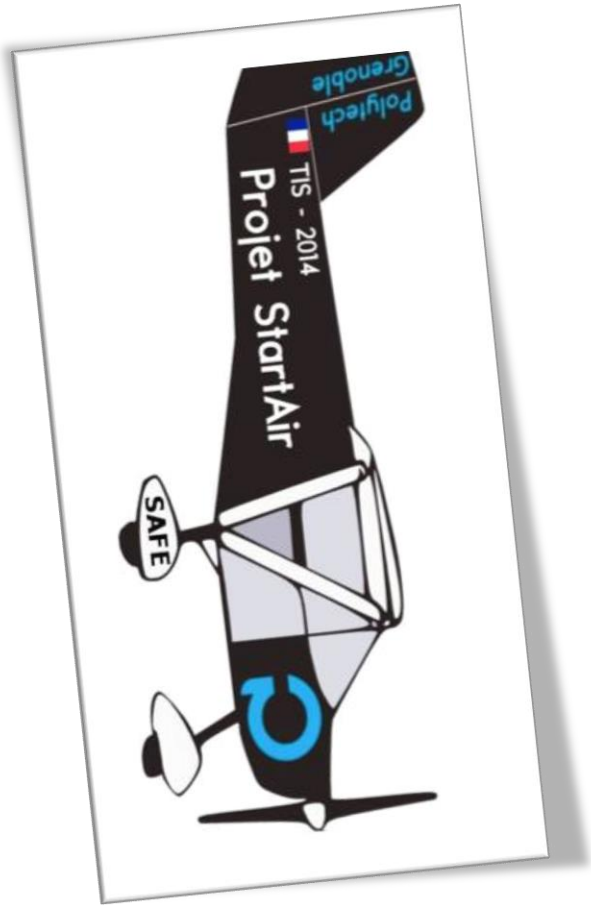
Réalisation 5
- travail préparatoire 5
- microsoft flight simulator 5
- simconnect sdk 5
- client / server 6
  - Récupération des données de vol 6
  - Transfert server-client 6
  - Affectation des données au simulateur pour l' affichage 6
- performances 7

Le projet 8
- gestion du temps 8

Conclusion 9
- perspectives 9

# Introduction

Start'Air Safe est un projet multi-filière entre les départements TIS, 3I, Matériaux et RICM de Polytech'Grenoble. Ce projet a pour but la construction d'un simulateur de vol le plus réaliste possible. Pour ce faire, la conception a été divisée en plusieurs parties : l'aspect électronique et reconstitution du tableau de bord pour les 3I, l'analyse des données médicales pour les TIS, l'amélioration de la structure pour les Matériaux et la création d'un plugin pour FSX pour les RICM. Nous allons maintenant vous expliquer en quoi a consisté notre partie du projet.




## LES BESOINS

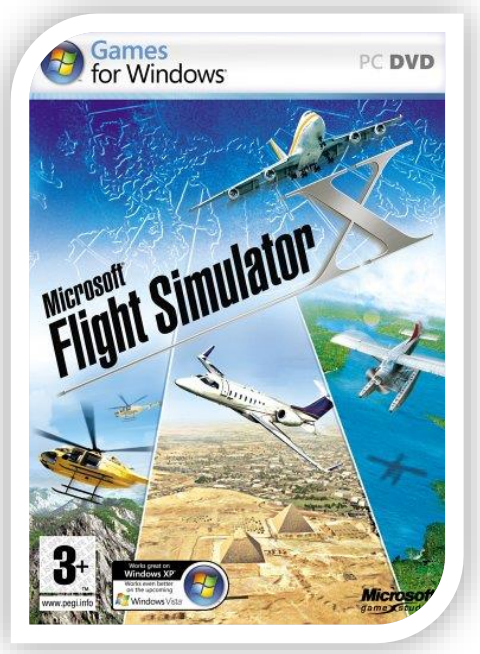
Pour bien comprendre notre projet, il faut savoir que l'association Start'Air s'occupe en priorité de personnes atteintes d'une forme d'autisme appelé « trouble du spectre autistique » plus connu sous le nom de syndrome d'asperger. Les « asperger » sont spéciaux dans le sens où ils ont du mal à comprendre la « réalité » et pour échapper à cela, ils se réfugient dans un domaine particulier dans lequel ils excellent (imagination, musique, aéronautique etc…). En conséquences, de certain « asperger » adorent voler. Notre projet dans son ensemble a pour but d'observer ces personnes lorsqu'elles sont « dans leur monde » pour mieux comprendre le phénomène et ainsi les aider à surmonter leur peur de la réalité.

Notre partie dans ce projet consiste donc à créer un simulateur de vol pour mettre les personnes en situation réelle. Plus l'immersion est total meilleur seront les résultats des analyses.

# INTRODUCTION

## PLATFORME DE TRAVAIL

Pour être fonctionnel à 100%, notre projet a besoin d'une plateforme particulière qui se compose des instruments suivants :




- Des écrans pour afficher la simulation. Les caractéristiques et le nombre importent peut. Pour noter prototype, nous utilisons 3 écrans 32 pouces de la marque «

- Autant d'ordinateur que d'écran tournant sous Windows XP minimum. Nous utiliserons ici 3 ordinateurs DELL sous Windows XP.

- Un hub USB pour créer notre réseau local

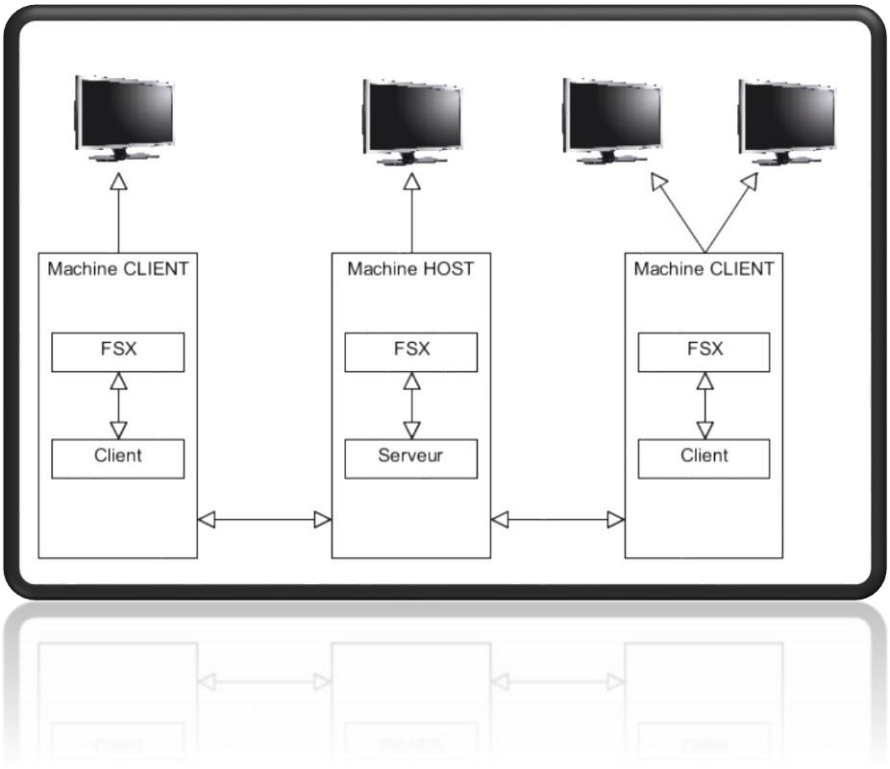
- Autant de licences Microsoft Flight simulator que d'ordinateur. Nous utilisons pour le moment des versions d'essais sur tous les ordinateurs.

# Conception

Pour mettre en place une architecture solide, il nous a fallu faire plusieurs prototypes simples. Ils nous ont permis d'identifier les principales fonctionnalités et les principales difficultés que posait l'interface du simulateur.

## ARCHITECTURES

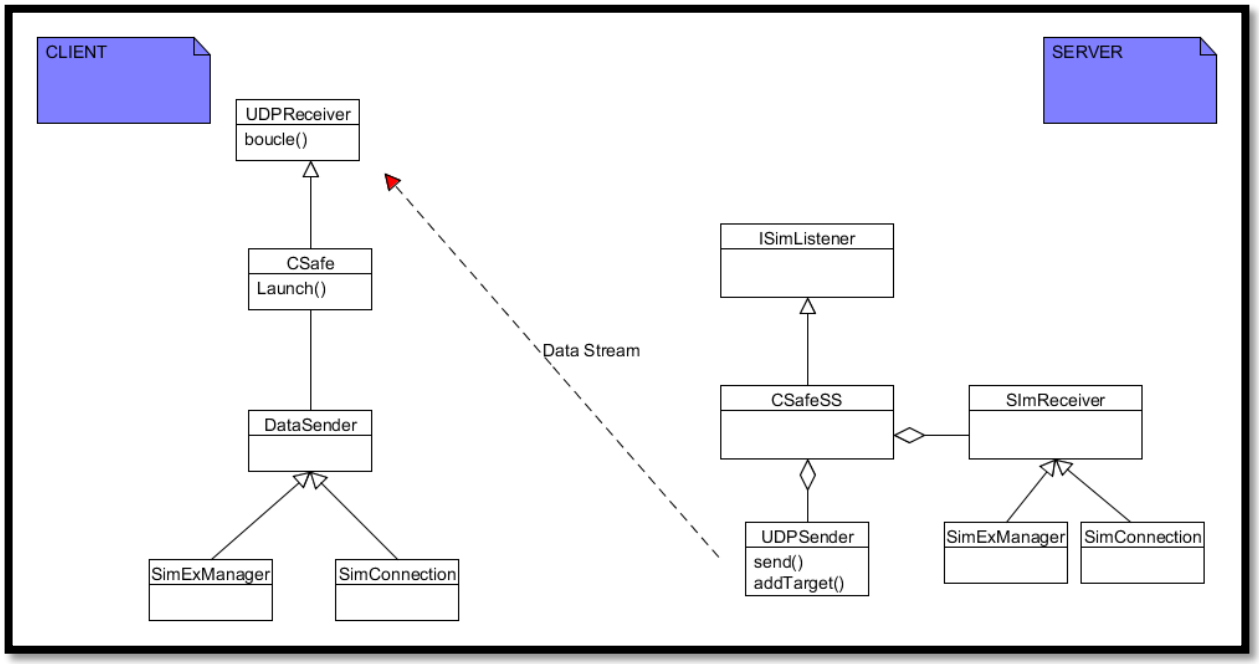
Ces notions en tête, nous avons prévu l'architecture globale du simulateur.




Il y aura donc une machine principale ou « HOST » qui exécuterait la simulation. C'est cette machine qui reçoit les commandes de manipulation de l'ULM et qui ensuite calcule le vol. Les autres machines dites « CLIENT » sont toutes reliées à la machine principale mais pas entre

elles. Les simulateurs de ces machines ne font pas de calcul de position et ne reçoivent pas de contrôle pour l'appareil, ils ne font qu'appliquer les données reçues. Les machines « HOST » ou « CLIENT » peuvent afficher la simulation sur un nombre d'écran indéfini. La limitation étant les capacités de la machine et la complexité de la configuration des vues lors du multi-écrans. Les applications client et serveur sont les parties principales de notre projet. L'application serveur récupère les données du simulateur principal pour les envoyées aux applications clients qui envoie respectivement les données au simulateur de leur machine.

L'architecture de ces applications est décrite en dessous.



# Réalisation

## TRAVAIL PREPARATOIRE

Nous avons fait le choix de Visual Studio comme IDE. La mise en place du projet en a été simplifiée, un tutoriel étant déjà disponible pour préparer l'environnement. Notre application ne pouvant fonctionner que sous Windows, la création d'un projet sur un IDE Windows nous semblait logique.

L'application devant être adaptable pour un nombre de machine variable, nous avons recherché une librairie gérant les fichiers de configurations. Nous avons choisis « libconfig » qui ne nous semblait pas difficile à implémenter ni trop compliquer dans sa syntaxe du fichier de configuration.

Les prototypes qui ont été créés pour la compréhension du SDK ont été en partie réutilisés dans l'application sous forme de classe ou de bout de code.

## MICROSOFT FLIGHT SIMULATOR

Le simulateur de Microsoft, Flight Simulator X a été choisis par l'association principalement car elle possède déjà une clé de ce simulateur. Il existe plusieurs forums d'utilisateur de ce simulateur mais ils ne sont pas tous développeur dessus.

## SIMCONNECT SDK

SimConnect est le kit de développement associé à Flight Simulator. Il peut être utilisé en C/C++ ou C#. Il permet l'accès en écriture comme en lecture des données du simulateur.

Microsoft fournit une documentation assez complète de SimConnect avec plusieurs exemples même si certaines lacunes d'informations peuvent mener à de longue heure de debug.

# REALISATION

## CLIENT / SERVER

### ❖ Récupération des données de vol

Le SDK permet la récupération de données de vol. Dans un premier temps, il faut définir qu'elles données récupérer avec quelles types et l'indiquer au simulateur. La requête de donnée peut alors être envoyée avec la fréquence de rafraichissement. La donnée est alors reçu à la fréquence demandée et sous la forme définit plus tôt.

### ❖ Transfert server-client

Initialement, le client pouvait directement faire une requête de donnée au simulateur principal. Cela nous à poser problème car le simulateur envoyai ses données par lot dans un paquet TCP ce qui nous empêchais de faire du temps réel.

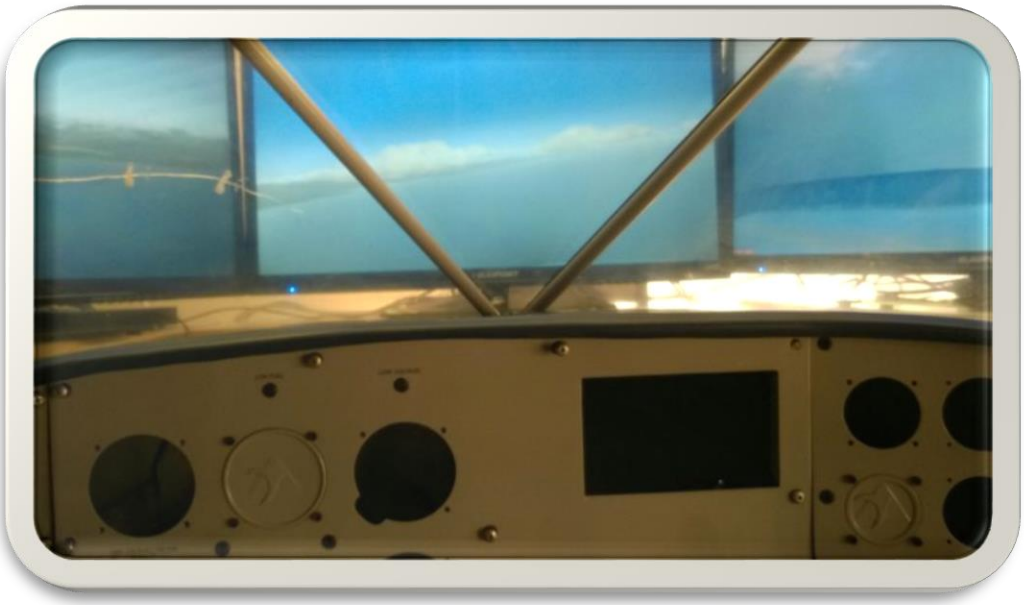
Ayant remarqué dans un prototype que cet envoi par lot n'était pas effectif si la requête de donnée était local, nous avons réalisé une couche UDP. Celle-ci transfert à la volé des données reçu et nous permet d'atteindre une exécution temps réel suffisante pour ne pas remarquer de délai.

### ❖ Affectation des données au simulateur pour l'affichage

La donnée reçue par le client au travers de la couche UDP est affecté au simulateur local à l'aide de SimConnect.

Comme précédemment, il faut définir au simulateur le type de donnée que l'on souhaite utiliser. Elle peut alors être envoyée.

# REALISATION



## PERFORMANCES

Comme expliqué précédemment, nous avons dû changer de Protocol de transfert des données pour passer de TCP à UDP ce qui a entrainé une augmentation des performances :

TCP : 10paquets / sec avec 4 lots

UDP : 22 paquets / sec avec 1 lot

## LIGNES DE CODE

| TECHNOLOGIES | CHIFFRE D'AFFAIRES |
|---|---|
| **Client** | 157 |
| **Utils / Global** | 669 |
| **Server** | 140 |
| **CaméraTool** | 130 |

Nous avons un total de 1096 lignes de codes dans notre projet. Nous avons entièrement écris ce code dans lequel nous ne comptons pas les librairies utilisées.

# Le projet

## GESTION DU TEMPS

Lors de ce projet, nous avons séparé la gestion du temps en trois parties. Dans un premier temps nous nous sommes concentrés sur la recherche d'information sur le fonctionnement de FSX et de la SDK associé et nous avons réalisé un prototype puis un second plus performant après plusieurs tests. Dans un deuxième temps nous avons réalisé le logiciel suivant les technologies de notre deuxième prototype. Enfin nous avons rédigés une documentation Développeur ainsi qu'un guide d'utilisation de notre logiciel.

### TACHES

| TECHNOLOGIES | TEST |
|---|---|
| **Recherche des technologies** | W/C |
| **Prototype communication simulateur local [TCP]** | W |
| **Test de performance [mauvaise performance TCP]** | W/C |
| **Prototype communication simulateur a distance [passage à UDP]** | C |
| **Test des performances** | W/C |
| **Réalisation du Server** | C |
| **Réalisation du Client** | W |
| **Test de transmission Client ←→ Server** | W/C |
| **Test de transmission Client →** FSX local | W/C |
| **Génération de la documentation** | W |
| **Rédaction du rendu** | C |

# CONCLUSION

## Conclusion

Pour conclure, nous pouvons dire que notre partie du projet a été réalisée dans sa totalité. Il nous est possible d'afficher Microsoft flight simulator sur un nombre illimité d'écran (pour un nombre raisonnable) et ce sans différence de timing entre les différents affichages. Nous tenions à remercier … pour l'achat et le prêt des écrans et … pour le prêt des ordinateurs qui nous ont permis de réaliser le prototype.

Nous souhaitons que notre projet ne s'arrête pas la et qu'il soit amélioré dans les années futures car les « aspergers » sont des personnes très sensibles et intéressantes qui ont absolument besoins de notre aide.

Pour ce faire, nous proposons plusieurs perspectives d'évolution pour le simulateur et le projet Start'Air Safe en général.

### PERSPECTIVES

Comme expliqué plus haut, nous avons pensés à plusieurs évolutions possibles en rapport avec RICM pour rendre le simulateur plus réaliste :

- ✓ Ajouter un sons surround à l'intérieur de la cabine
- ✓ Relier le tableau de bord des 3I à notre server via Arduino / Rasberri
- ✓ Ajouter des elements à l'intérieur de la simulation pour ammélioreer les perfomances même du simulateur
- ✓ Utiliser la réalité virtuelle avec l'occulus rift pour etre totalement immergé. Attention, cette idée est à investiger avant un quelconque test car cette technologie a des effets secondaire qui pourraient etre trop puissant pour les « asperger » par exemple il dérègle l'orloge interne ce qui donne une sensation de nausée.
- ✓ Utiliser google glass et la réalité virtuelle pour le tracking des yeux ajouter des elements de vol.