Raport de stage Eleve Ingeniéur

Emmanuel Kervizic

10 septembre 2010

Table des matières

Chapitre 1

La DGAC

1.1 Présentation de l'entreprise

1.1.1 Description générale

Activités de l'entreprise et historique

Quelques chiffres

localisation

Ses domaines d'activités.

Organisation

Ses domaines d'activités.

Son pôle de recherche.

Ses compétences, son marché.

Présentation générale du service

La Direction des Services de la Navigation Aérienne est chargée de rendre le service de navigation aérienne pour l'Etat français. A ce titre, la DSNA est responsable de rendre les services de circulation aérienne, d'information aéronautique et d'alerte sur le territoire national et ceux d'outre-mer (DOM, TOM , POM). La DSNA s'appuie sur deux directions pour exécuter cette mission :

- La Direction des opérations ou DO,
- la Direction de la Technique et de l'Innovation ou DTI.

La DO est l'acteur opérationnel du contrôle aérien tandis que la DTI est chargé du volet technique. Celui-ci consiste à réaliser ou acquérir les systèmes qui participent à l'exercice du contrôle aérien. Il s'agit de systèmes informatiques permettant d'assister le contrôleur dans ses activités, de chaines radios pour communiquer avec les aéronefs, de systèmes de traitement de l'information météorologique...

La DTI réalise également de nombreuses études pour traiter les besoins des utilisateurs et les évolutions réglementaires. La DTI réalise le déploiement et le support opérationnel des systèmes qu'elle acquiert ou réalise.

Enfin la DTI fait viser ses systèmes, procédures et formation par l'autorité de surveillance nationale (Direction de la Sécurité de l'Aviation Civile ou DSAC).

La DTI est structurée en domaines qui sont chacun en charge de plusieurs pôles de compétences :
- Recherche & développement, R et D
- Exigences opérationnelles des systèmes, EOS
- Gestion du trafic aérien, ATM
- Communication, navigation, surveillance, CNS
- Déploiement et Support Opérationnel, DSO

Chaque pôles qui couvre un ensemble de fonctions et d'expertises. Pole ATM/-VIG : Le pôle « Vol et information générale » (VIG) est responsable de la maîtrise d'ouvrage systèmes de traitement des plans de vol et informations générales, à ce titre, le pôle assure le suivi industriel de leur réalisation ou de leur acquisition. Le pôle VIG est également chargé de leur maintien en condition opérationnelles lorsqu'ils sont déployés. Le pôle ATM/VIG est notamment responsable de la maîtrise d'ouvrage de systèmes déployés en outre-mer. L'aéroport de Tahiti (Polynésie française) a récemment été modernisé avec un système entièrement acquis auprès d'un industriel, couplé à un radar dans le cadre du projet TIARE, qui s'est terminé en 2009.

Les partenaires

Tissu local, sous-traitant.

Relations au plan local, nationnal...

1.1.2 Situation géographique

Son emplacement.

1.2 Environnement de travail

Chapitre 2

Le projet

2.1 Introduction

2.1.1 Le systeme de contrôle aérien mis en place à Tahiti

Le système TIARE

TIARE est un système de gestion du trafic aérien pour le centre de contrôle de Tahiti, en remplacement des systèmes vieillissants de visualisation du trafic (VIVO) et de gestion de plans de vol et d'informations générales (SIGMA). La superficie de l'espace aérien géré par le centre de contrôle de Tahiti s'étend sur 12 500 000 km2. Les situations de contrôle auxquelles doivent face les contrôleurs sont multiples, il y en a en effet à traiter les spécificités du contrôle océanique, du contrôle d'approche et inter-iles. Le système TIARE est construit à partir de plusieurs « produits sur étagère » :

- EUROCAT-X, système en charge du traitement radar et de la gestion plans de vols.
- ATALIS, système en charge de la préparation des vols, de la gestion des NOTAM, et de la présentation d'informations générales au contrôleur tour et approche.

Les systèmes EUROCAT-X et ATALIS sont connectés au commutateur CAGOU, raccordé aux liaisons externes (RSFTA). ATALIS reçoit également des informations météorologiques en provenance du système local d'acquisition de ces données appelé CAOBS. EUROCAT-X est raccordé au radar secondaire du mont Marau et au réseau ACARS.

La zone Aci :

Une fonction de contrôle spécifique, nommée Aci [1] ou zone Aci, a été développée dans le système EUROCAT-X pour répondre à des besoins de contrôle. Il s'agit d'une zone particulière limitrophe à la Fir [2] de Tahiti , dons la limite se situe à 50 miles nautiques de la Fir. La zone Aci encercle la Fir. Il est à noter que cette zone n'est pas sous la responsabilité des contrôleurs aériens français, cependant, les vols pénétrant dans cette région sont visualisés par le système Eurocat-X

1. Aci : Area Common Interest, soit une zonne d'intérets commun
2. la Fir est la zone dans laquelle les contrôleurs doivent assurer le contrôle des vols

Ainsi en visualisant le trafic aérien dans la zone ACI, les contrôleurs peuvent maintenir les séparations entre les aéronefs. C'est-à-dire vérifier que les vols qui sont à l'extérieur et longent la FIR de Tahiti sont séparés des vols évoluant dans cette FIR.

2.2 Les objectifs et besoins du projet

2.2.1 L'objectif initial du projet

L'objectif principal est de pouvoir réaliser un logiciel banalisé et ergonomique permettant de représenter l'ensemble des données de contrôle (repères, balises, secteurs...) afin de pouvoir visualiser le trafic aérien circulant dans la FIR et la zone ACI. Les bénéfices attendus de cet outil sont :

- l'amélioration de l'analyse et de la compréhension visuelle du trafic aérien de Tahiti,
- la possibilité d'élaborer de statistiques à partir des fonctions de calcule du logiciel,
- une aide dans le travail de définition des points d'entrée dans la zone ACI que réalise le service de contrôle de Tahiti.

2.2.2 Le besoin

Le besoin au début du projet

Le besoin était initialement de pouvoir visualiser les plans de vol des avions afin de pouvoir aider les contrôleurs à déterminer l'heure d'entrée approximative de l'avion dans la zone ACI (cf. 2.1.1 page précédente) Le logiciel devant être portable et adaptable, l'utilisation langage Python a été défini comme l'une des meilleur alternative. Tout au long du projet de nouveau besoin se sont fait ressentir. Ceux-ci comme nous pourrons le constater dans la suite du rapport nous a amener à modifier les objectifs du projet.

Un besoin redéfinit tout au long du projet

On a très vite constaté, ce e dès les premières représentations, que la création d'un logiciel qui pouvait représenter les données du système TIARE dans Google Eath pourrait avoir plusieurs intérêts :

- Visualiser comment le Système interprète les données
- Comparer les plan de vol déposé avec les vols réalisé
- Visualiser les zone de contrôles
- Évaluer la mise en place du système ADS

C'est pourquoi les besoins on été redéfinit en cours de projet.

Des besoins redéfinis

Les nouveaux besoins définis sont donc de pouvoir disposer d'une maquette (Appellée en anglais : Poc[3]) qui serrait représentative de toutes les possibilités que pourrait apporter un logiciel de ce type.

Cette maquette doit être capable de :

- définir un point d'entré approximatif d'un plan de vol dans la zone ACI.
- Visualiser les zone de contrôles
- Visualiser tous les points caractéristique du système (Aéroport, balises ...)
- Visualiser les routes définie dans le système.
- Visualiser les plan de vol en fonction du temps
- Visualiser les report ADS ainsi que les points calculés par le système TIARE
- Différencier les vols interne, sortant, entrant et en transit.

L'intérêt de cette maquette serrait de définir les spécification d'un logiciel qui pourrait être ensuite réalisé et mis en production dans des sites ou la sécurité n'est pas à négliger. Il aurait aussi l'intérêt de définir les fonctionnalités nécessaires à un utilisateur donné.

2.3 La réalisation du projet

2.3.1 Les prérequis

Avant chaque intervention sur le code une étape apprentissage a été nécessaire, dans cette partie serra décrite les partie qui m'ont le plus sollicitées.

EUROCATX

Il faut bien comprendre comment marche le système afin de bien visulaliser d'où proviennent les informations. Comme décrit grossièrement dans le schéma (cf. Fig. 2.1 page suivante), EUROCATX récupere les information sur les plans de vol par l'intermediaire de CAGOU[4]. Il rcupere aussi le positionement émis par l'avion a l'aide de la transmition Satelite, VHF[5] ou des données radar lors de son approche. Le système EUROCATX donne un acces à la bureautique protégé par un parfeu (FireWall) afin de rendre disponible sur ce réseau un certain nombre d'information. Dans notre cas nous y recupererons :

- toutes les données de configuration du système tels que les nom et coordonnées des balise referancée, la position des zone de controle et des zone ACI ou encore les route utilisée pour décrire les plans de vols.
- Les fichiers de log du Commutateur CAGOU afin de pouvoir exploiter les plans de vol reçus par le réseaux RSFTA.
- Tous les report ADS reçu par satelite et traité par le systeme.

Le système envoye les information recoltées et celle calculées au visues[6] situées dans la tour de controle au niveau de la Vigie ou de la salle CCR ainsi que de la position

3. Poc : Proof of concept, soit une démonstration de faisabilité
4. CAGOU : nom donné au commutateur RSFTA
5. VHF : Very High Frequency, soit une bande radio de très haute frequance
6. Visue : Nom pour décrire les ordinateur utilisés pour visualiser les données de controles

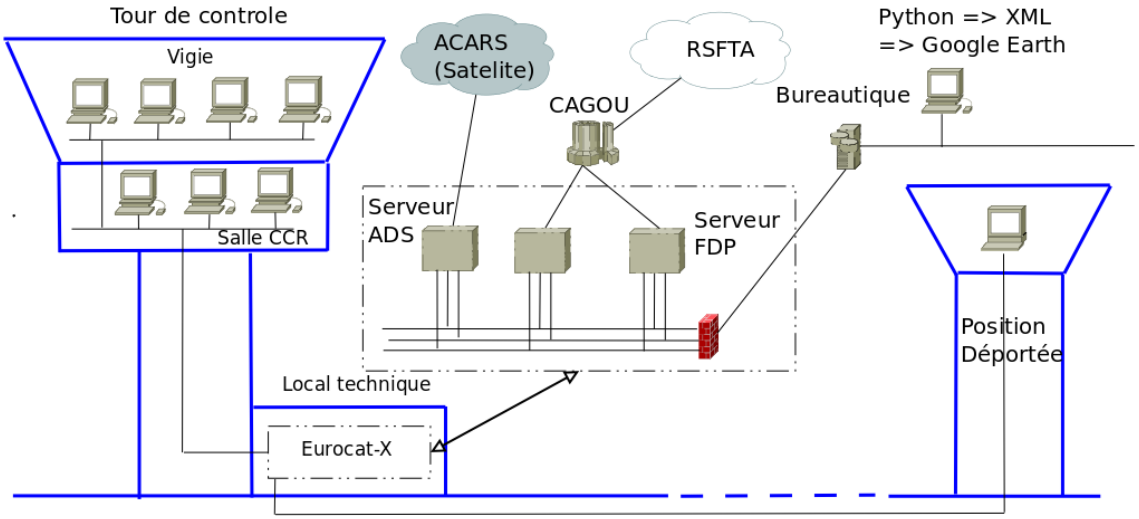

FIGURE 2.1 – Schématisation du système EUROCATX au niveau des tour de controle

deportée à MOREA.

Les données seront donc recupérée dans les fichiers ".asf" pour tous ce qui est de la configuration du système, dans les fichier du FDP pour les plans de vol et dans les fichiers du serveur ADS pour les repport ainsi que pour la position calculée des aéronefs

Le langage Python

Bien coder : Afin de pouvoir apprendre les bonne pratique de la programmation Python[4] j'ai lu (en diagonale) un livre tres bien expliqué intitulé Programation Python, conception et Optimisation. Celui-ci m'a permit de pouvoir d'une part revoir ce qui avait été appliquer lors de mes etudes et d'autre part avoir une vue global sur le langage et ainsi pouvoir prendre du recule lors du codage.

Celui ci m'a par exemple appris le nouveau style de programmation qui part du principe que chaque nouvel objet définit est basé sur un Objet existant, et que par la même occasion tout en python était Objet (même une simple variable booléene). Ou encore la maniere de verifier si un objet etait faux, egale à 0 ou encore une chaine vide simplement en demandant si il existait (ex : `"if x != 0:"` devient `"if not x:"`)

Utiliser les expression régulière : L'apprentissage de l'utilisation des expression réguliere[7], m'a été grandement facilité garce au site : `http://www.dsimb.inserm.fr/`[1] et a la documentation en ligne de Python[3]. Il s'est avéré après apprentissage que ces expression régulière auron grandement facilité la faisabilité du projet.

L'optimisation : Je pourrais cité un passage du livre qui dit :

7. Une expression régulière est en informatique une chaîne de caractères que l'on appelle parfois un motif et qui décrit un ensemble de chaînes de caractères possibles selon une syntaxe précise.

> Fourni dès le départ avec des modules de tests, Python est un langage agile. Le terme agile est originellement issu de la méthodologie de programmation agile (Beck et Al.), très proche de la programmation itérative. Cette méthodologie, qui réduit les risques liés à la conception de logiciels, introduit entre autres des principes de tests continus du code. Vincent LOZANO.

En effet il m'a été rapidement nécessaire de réalisé des test, aussi bien pour vérifier que mon code etait valide que pour vérifier que celui-ci s'exécutait normalement. Il c'est avéré a plusieurs reprise que certaine partie de mon code était très gourmande en processus . L'apprentissage des fonction de test poussé du code tel que le module hotshot décrit plus tard m'a été rapidement nécessaire.

Le domaine de l'aviation

Il m'a aussi été nécessaire de prendre connaissance de touts les terme, unité, convention et j'en passe utilisé dans le domaine aéronautique.

Les coordonnées et unités : Tout d'abord est vite venu le problème de conversion de coordonnées, J'ai donc du revoir les conversions de coordonnées sphériques ainsi que les conversions de distances. J'ai également du, comme expliquée ci dessus me remémoré les manière de calculé le point d'intersection de deux arc de cercle en coordonnées sphériques.

Convention : Plusieurs conventions on du être acquise comme celle utilisé par le système TIARE pour décrire les report ADS ou encre celle utilisée par les compagnie pour le dépôt de plan de vole. NE PAS OUBLIER DE FAIR REF AU DOCUMLENT 4444 ...

GOOGLE ERATH

GOOGLE ERATH est un logiciel, propriété de la société GOOGLE, permettant une visualisation de la terre en 3 dimensions avec un assemblage de photographies aériennes ou satellitaires. Ce logiciel donne la possibilité de configurer un environement, ajouter des lignes, des points ou encore des polygone en 3D en passent par des fichier de configuration au format KML[8].

Ce format, qui repose sur le XML[9], a l'avantage d'être simple à manipuler. Ça sémantique est définie sur le de google (cf. Biliographie [2])

8. KML : Keyhole Markup Language, est un format de fichier et de grammaire XML pour la modélisation et le stockage de caractéristiques géographiques comme les points, les lignes, les images, les polygones et les modèles pour l'affichage dans GOOGLE ERATH, dans GOOGLE MAPS et dans d'autres applications.

9. XML : Extensible Markup Language («langage extensible de balisage»), est un langage informatique de balisage générique.

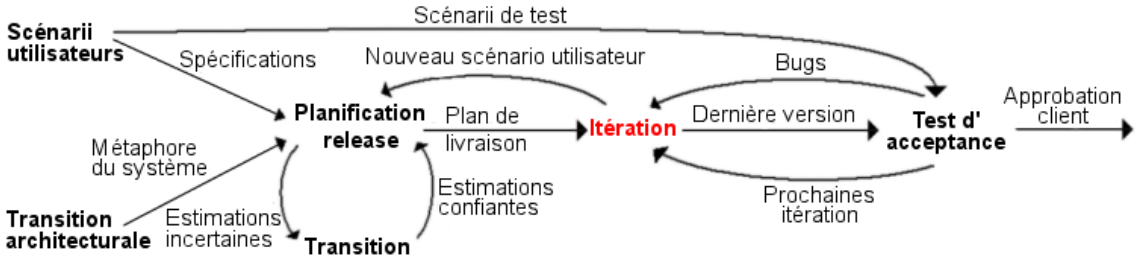

FIGURE 2.2 – Cycle de l'Exreme Programing.

2.3.2 La méthodologie appliqué

Pour la réalisation de ce projet, vue les circonstance, une méthodologie existante c'est mise en place automatiquement. Celle ci est l'Extreme programming [10] décrite ci-après.

Dans les méthodes traditionnelles, les besoins sont définis et souvent fixés au départ du projet informatique ce qui accroît les coûts ultérieurs de modifications. Extreme programming s'attache à rendre le projet plus flexible et ouvert au changement en introduisant des valeurs de base, des principes et des pratiques.

L'Extreme Programming repose sur des cycles rapides de développement (des itérations de quelques semaines voir dans notre cas quelques jours seulement) dont les étapes sont les suivantes :

- une phase d'exploration détermine les scénarios clients qui seront fournis pendant cette itération,
- la transformation des scénarios en tâches à réaliser et en tests fonctionnels,
- lorsque tous les tests fonctionnels passent, le produit est livré.

Lorsqu'une tâche est terminée, les modifications sont immédiatement intégrées dans le produit complet. On évite ainsi la surcharge de travail liée à l'intégration de tous les éléments avant la livraison. Les tests facilitent grandement cette intégration : quand tous les tests passent, l'intégration est terminée.

Le cycle se répète tant que le client peut fournir des scénarios à livrer (cf. Fig. 2.2). Généralement le cycle de la première livraison se caractérise par sa durée et le volume important de fonctionnalités embarquées. Après la première mise en production, les itérations peuvent devenir plus courtes (par exemple la séparation des plans de vol en catégories tel que : transit, interne . . .)

Il est bien entendu qu'avant de passer a la pratique un apprentissage théorique a du etre réalisé comme par exemple :

- L'apprentissage des bonnes manières de coder en python (cf. 2.3.1 page 8),
- Le fonctionnement des fichier KML (cf.
- Ou encore les formule mathématique permettant de trouver l'intersection de deux arc de cercle en coordonnées sphérique (cf.

10. L'Extreme Programming a été inventée par Kent Beck, Ward Cunningham et Ron Jeffries pendant leur travail sur un projet « C3 » de calcul des rémunérations chez Chrysler. Kent Beck, chef de projet en mars 1996 commença à affiner la méthodologie de développement utilisée sur le projet. La méthode est née officiellement en octobre 1999 avec le livre Extreme Programming Explained de Kent Beck. "Wikipedia"

Bibliographie

[1] Patrick Fuchs and Pierre Poulain. Expressions régulières et parsing. http://www.dsimb.inserm.fr/~fuchs/python/python-node13.html, june 2010.

[2] Google. Documentation en ligne sur la sémentique des documents kml. http://code.google.com/apis/kml/documentation/kmlreference.html, june 2010.

[3] Python v2.7. Documentation en ligne de python. http://docs.python.org/, june 2010.

[4] Tareck Ziadé. *Programmation Python, Conception et optimisation*. Eyrolles, 2009.

Annexe A

Les codes sources du projet

A.1 Main

Ce fichier sert à executer tout le programme :

```
#coding: utf-8
__author__ = "KERVIZIC Emmanuel (kervizic@hotmail.com)"
__version__ = "0.0.1"
__license__ = ""
__copyright__ =""

import sys
sys.path.append(r'modules')
import CharacteristicPoints, Routes, KML, Fdp, GetOfFiles, MakeKML, Ads
import MakeKMZ, Aoi, os

def main():
    print ("""
###############################################################################
## Representation des traces et routes des avions dans Google Earth ##
#    Programme realise par: KERVIZIC Emmanuel                                 #
#       Pour :   La DTI de l'aviation civile                                  #
##      Le :     17/06/2010                                                  ##
###############################################################################
\nC'est un bon debut ;)\n""")

    allObjects = GetOfFiles.getOfFile()
    makeFile = allObjects['makeFile']
    if makeFile['characteristicsPoints'] == 'yes':
        kmlCP = MakeKML.addAllCharacteristicsPoints(allObjects)
    if makeFile['routes'] == 'yes':
        kmlRT = MakeKML.addRoutes(allObjects)
    if makeFile['fir'] == 'yes':
        kmlST = MakeKML.addSector(allObjects)
    if makeFile['fpl'] == 'yes':
        kmlFP = MakeKML.addFpl(allObjects)
    if makeFile['ads'] == 'yes':
        kmlFP = MakeKML.addAds(allObjects)
    if makeFile['main'] == 'yes':
        kmlMain = MakeKML.addMain(allObjects)

    kmz = MakeKMZ.makeFile(allObjects)
    print "Et ca fini bien !"

# Execute only if this file is main
if __name__ == '__main__':
    main()
```

/home/manu/DTI/Manu.py

A.2 Config

Nous avons ic le fichier de configuration. Celui ci sert notament à se passer temporairement d'une interface graphique.

```
################################################################################
##                                                                            ##
#    File of configuration                                                    #
#     Here is complete all the fields needed to run the program               #
#                                                                             #
#    WARNING !!!                                                              #
#     Do not edit this file without considering the consequences              #
##                                                                            ##
################################################################################
// option
# enable the deletion of the area during the approach : yes or no
deleteZone                      = no


// update or crate KML File
# if you want to update the KML file, set to: yes, the variable
# Otherwise, set it to no

# All characteristics points, very long to write
characteristicsPoints           = yes
#characteristicsPoints           = no
# All routes
routes                          = yes
#routes                          = no
# All fir
#fir                             = yes
fir                             = no
# All Flight Plan and ADS position
fpl                             = yes
#fpl                             = no
# All ADS
ads                             = yes
#ads                             = no
# The main KML File
main                            = yes
#main                            = no



// path of files
# warning, all directories used must be created

## WINDOWS
#mainPath                        = D:\Documents de Manu\DTI\
#characteristicsPointsFilePath   = .\Sources\CHARACTERISTIC_POINTS.ASF
#routesFilePath                  = .\Sources\ROUTES.ASF
#fdpFilePath                     = .\Sources\FDP_VOLUMES_DEFINITION.ASF
#aoiFilePath                     = .\Sources\AOI_VOLUMES.ASF
#fplFilePath                     = .\Sources\FDX
#absRoutesPath                   = .\AC\KML\Routes.kml
#absCharacteristicsPointsPath    = .\AC\KML\CharacteristicsPoints.kml
#absFirPath                      = .\AC\KML\Fir.kml
#absFplPath                      = .\AC\KML\Fpl.kml
#absMainPath                     = .\AC\doc.kml
#absKmlPath                      = .\AC
#absKmzPath                      = AirControl.kmz

# OTHER
mainPath                        = /media/manu/Documents de Manu/DTI/
characteristicsPointsFilePath   = SourcesAsf/CHARACTERISTIC_POINTS.ASF
routesFilePath                  = SourcesAsf/ROUTES.ASF
fdpFilePath                     = SourcesAsf/FDP_VOLUMES_DEFINITION.ASF
aoiFilePath                     = SourcesAsf/AOI_VOLUMES.ASF
fplFilePath                     = Sources/
adsFilePath                     = Sources/
```

```
absRoutesPath                   = AC/KML/Routes.kml
absCharacteristicsPointsPath    = AC/KML/CharacteristicsPoints.kml
absFirPath                      = AC/KML/Fir.kml
absFplPath                      = AC/KML/Fpl.kml
absAdsPath                      = AC/KML/Ads.kml
absMainPath                     = AC/doc.kml
absKmlPath                      = AC
absKmzPath                      = AirControl.kmz
# used in KMZ archive
kmlFilePath                     = files/
kmlRoutesPath                   = KML/Routes.kml
kmlCharacteristicsPointsPath    = KML/CharacteristicsPoints.kml
kmlFirPath                      = KML/Fir.kml
kmlFplPath                      = KML/Fpl.kml
kmlAdsPath                      = KML/Ads.kml


// Object for point KML Style
circle                          = placemark_circle.png - 0.2 - FFFFFFFF
redCircle                       = placemark_circle_highlight.png - 0.4 - FF0000FF
polygon                         = polygon.png - 0.2 - FF00FFFF
triangle                        = triangle.png - 0.2 - FF00FFFF
airports                        = airports.png - 0.2 - FFFF6633
avion                           = avion.png - 0.6 - FF888888
avion2                          = avion2.png - 0.4 - FFFFFFFF
avion3                          = avion2.png - 0.2 - FFFFFFFF
avion4                          = avion.png - 0.3 - FFFFFFFF
redRound                        = redround.png - 0.4 - FF0000FF
greenRound                      = greenround.png - 0.3 - FFFFFFFF
blueRound                       = blueround.png - 0.3 - FFFFFFFF
glossyRound                     = glossyround.png - 0.1 - FFFFFFFF
orangeRound                     = orangeround.png - 0.2 - FFFFFFFF
yellowRound                     = yellowround.png - 0.3 - FFFFFFFF
blackRound                      = blackround.png - 0.2 - FF888888

// General LookAt
longitude                       = -138.5
latitude                        = -13.0
altitude                        = 0
range                           = 6000000
tilt                            = 0
heading                         = 0

// list of Color
# FIR EXT
NZZO                            = 4D000000
SCIZ                            = 4D0000FF
KZAK                            = 4D00FF00
SCTZ                            = 4D00FFFF
NCRG                            = 4DFF0000
# FIR NTTT
VMOR                            = 4DFF00FF
VIWR                            = 4DFFFF00
VAPP                            = 4DFFFFFF
VCC1                            = 4D00CCFF
VCC2                            = 4DCCFF00
VCC3                            = 4DFF00CC
# AOI
VCC1_AOI                        = 4D00CCCC
VCC2_AOI                        = 4D0000CC

# Unused colors
#4D00CC00
#4DCC0000
#4DCC00CC
#4DCCCCCC
#4DCCFFFF
#4DCCCCFF
#4DFFFFCC
#4DFFCCCC
```

```
// Comment, name or description

CharacteristicsPointsFileName    = CharacteristicsPoints
CharacteristicsPointsFileDescription = Includes all Characteristics points
AirportFolderName                = Airport
AirPortFolderDescription         = The online CSCIs use AIRPORT_I, AIRPORT_II and
    AIRPORT_III to distinguish airports from the other kind of points.
DummyFolderName                  = Dummy
DummyFolderDescription           = The DUMMY type is used to define a sector
    crossing point associated to a sector for FDP. Dummy point coordinates are not
    significant, but have to be input for the point to be correctly validated
ReportFolderName                 = Report
ReportFolderDescription          = not defined
VorFolderName                    = Vor
VorFolderDescription             = not defined
RoutesFileName                   = Routes
RoutesFileDescription            = Includes all Routes referenced
FplFileName                      = Fpl
FplFileDescription               = Flight plan
AdsFileName                      = Ads
AdsFileDescription               = Automatic Dependant Surveillance
CodedRoutesFolderName            = Simple coded routes
CodedRoutesFolderDescription     = Composed of a list of points defined in the
    CHARACTERISTIC_POINTS file.
SidFolderName                    = SID
SidFolderDescription             = Standard Instrument Departure coded routes
StarFolderName                   = STAR
StarFolderDescription            = Standard Arrival coded routes.
SectorFileName                   = FIR
SectorFileDescription            = Includes all FIR
FirNTTTFolderName                = FIR NTTT
FirNTTTFolderDescription         = All sector of NTTT FIR
FirExtFolderName                 = FIR EXT
FirExtFolderDescription          = All sector of NTTT FIRAll sector of other FIR
MainFileName                     = AirControl
MainFileDescription              = FIR de Tahiti
AoiFolderName                    = AOI
AoiFolderDescription             = AOI
VCC1_AOI                         = ACI ACC1 TAHITI a 50NM de la FIR
VCC2_AOI                         = ACI ACC2 TAHITI a 51NM de la FIR
in                               = Entrant
out                              = Sortant
transit                          = Transit
internal                         = Interne


// Folder are open or not in Google Earth at the start
# 0 for False and 1 for True
Airport                          = 0
```

/home/manu/DTI/manu.cfg

A.3 KML

Afin de pouvoir réaliser les document KML, un module a été implémenté. Celui-ci a pour objectif de mettre en forme le document final. Il ne réalise aucun calcul. Lors de l'initiation une variable est instencié. Celle-ci accumulera toute la mise en forme du document jusqu'a l'appel de la foction de fin qui permettera de clore cette variable et de l'enregistrer dans un fichier texte.

```
__author__ = "KERVIZIC Emmanuel (kervizic@hotmail.com"
__version__ = "0.0.2"
__license__ = ""
__copyright__ =""

```

```
#__author__ = "Jon Goodall <jon.goodall@gmail.com> - http://www.duke.edu/~jgl34"
#__version__ = "0.0.1"

class KML_File (object):
    """ For creating KML files used for Google Earth """

    def __init__(self, filePath, name, description, kmlFilePath, lookAt, isOpen):
        """
        Inport array whith general longitude, latitude and all
        data necessary for a general LookAt.
        lookAt['altitude'] for all lookat
        lookAt['longitude'] for the folder longitude
        lookAt['llatitude'] for the folder latitude
        lookAt['tilt'] for all lookat
        lookAt['range'] for all lookAt
        lookAt['heading'] for all lookAt
        """
        self.filePath = filePath
        self.lookAt = lookAt
        self.kmlFilePath = kmlFilePath
        self.kml = ''
        self.name = name
        self.isOpen = str(isOpen)
        self.ind = ''
        #adds the kml header to a file
        self.kml += """\
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<name> %s </name>
<open> %s </open>
<description> %s </description>
<LookAt>
    <longitude> %s </longitude>
    <latitude> %s </latitude>
    <altitude> %s </altitude>
    <range> %s </range>
    <tilt> %s </tilt>
    <heading> %s </heading>
    <altitudeMode>absolute</altitudeMode>
    <gx:altitudeMode>absolute</gx:altitudeMode>
</LookAt>
""" % (
name, # String
self.isOpen, # String
description, # String
lookAt['longitude'], # String
lookAt['latitude'], # String
lookAt['altitude'], # String
lookAt['range'], # String
lookAt['tilt'], # String
lookAt['heading'] # String
)

    def addPlacemark(self, name ,description , position, visibility,
    style = '', begin = '', end = ''):
        # adds the point to a kml file
        self.kml += """\
<Placemark>
<name> %s </name>
<visibility> %i </visibility>
<description><![CDATA[ %s ]]></description>
<styleUrl>#%s</styleUrl>
""" % (
name,
visibility,
description,
style,  # String
)
        if begin and end :
            self.kml += """\
```

```
77 <TimeSpan>
78 <begin> %s </begin>
79 <end> %s </end>
80 </TimeSpan>
81 """ % (
82 begin, # String
83 end # String
84 )
85         self.kml += """\
86 <Point>
87 <altitudeMode>relativeToGround</altitudeMode>
88 <coordinates>%f,%f</coordinates>
89 </Point>
90 </Placemark>
91 """ % (
92 position['longitude'], # Float
93 position['latitude'] # Float
94 )
95
96     def addLine (self, name ,description, color, points, visibility,
97     width, begin = '', end = '') :
98         # adds the line to a kml file
99         self.kml += """\
100 <Placemark>
101 <name> %s </name>
102 <visibility> %i </visibility>
103 <description><![CDATA[ %s ]]></description>
104 <Style>
105 <LineStyle>
106 <color> %s </color>
107 <width> %s </width>
108 </LineStyle>
109 </Style>
110 """ % (
111 name,
112 visibility,
113 description,
114 color,
115 width
116 )
117         if begin and end :
118             self.kml += """\
119 <TimeSpan>
120 <begin> %s </begin>
121 <end> %s </end>
122 </TimeSpan>
123 """ % (
124 begin, # String
125 end # String
126 )
127         self.kml += """\
128 <LineString>
129 <tessellate>1</tessellate>
130 <coordinates>
131 """
132         for i in sorted(points) :
133             self.kml += ('%f,%f\n') % (
134                 points[i]['longitude'], # Float
135                 points[i]['latitude'] # Float
136                 )
137         self.kml += """\
138 </coordinates>
139 </LineString>
140 </Placemark>
141 """
142
143     def writeLookAt (self, lookAtTmp) :
144         self.kml += """\
145 <LookAt>
146 <longitude> %s </longitude>
147 <latitude> %s </latitude>
```

```
148 <altitude> %s </altitude>
149 <range> %s </range>
150 <tilt> %s </tilt>
151 <heading> %s </heading>
152 </LookAt>
153 """ % (
154 lookAtTmp['longitude'], # String
155 lookAtTmp['latitude'], # String
156 lookAtTmp['altitude'], # String
157 lookAtTmp['range'], # String
158 lookAtTmp['tilt'], # String
159 lookAtTmp['heading'] # String
160 )
161
162     def addRegion (self) :
163         self.kml += """\
164 <Region>
165 <LatLonAltBox>
166 <north>90</north>
167 <south>-90</south>
168 <east>180</east>
169 <west>-180</west>
170 <minAltitude>0</minAltitude>
171 <maxAltitude>-1</maxAltitude>
172 </LatLonAltBox>
173 <Lod>
174 <minLodPixels>-1</minLodPixels>
175 <maxLodPixels>200000</maxLodPixels>
176 <minFadeExtent>0</minFadeExtent>
177 <maxFadeExtent>200000</maxFadeExtent>
178 </Lod>
179 </Region>
180 """
181
182     def addTimeSpan (self, begin, end):
183         self.kml += """\
184 <TimeSpan>
185 <begin> %s </begin>
186 <end> %s </end>
187 </TimeSpan>
188 """ % ( begin, end ) # String
189
190     def open_folder(self, name, description, isOpen):
191         self.kml += """\
192 <Folder>
193 <name> %s </name>
194 <description><![CDATA[ %s ]]></description>
195 <open> %s </open>
196 <LookAt>
197 <longitude> %s </longitude>
198 <latitude> %s </latitude>
199 <altitude> %s </altitude>
200 <range> %s </range>
201 <tilt> %s </tilt>
202 <heading> %s </heading>
203 </LookAt>
204 """ % (
205 name, # String
206 description, # String
207 str(isOpen), # String
208 self.lookAt['longitude'], # String
209 self.lookAt['latitude'], # String
210 self.lookAt['altitude'], # String
211 self.lookAt['range'], # String
212 self.lookAt['tilt'], # String
213 self.lookAt['heading'] # String
214 )
215
216     def close_folder(self):
217         self.kml += '</Folder>\n'
218
```

```
219     def openPlacemark (self, name, description, visibility):
220         self.kml += """\
221 <Placemark>
222 <name> %s </name>
223 <visibility> %i </visibility>
224 <description><![CDATA[ %s ]]></description>
225 """ % (
226 name,
227 visibility,
228 description
229 )
230 
231     def addIconStyle (self, name, imgName, scale = '1', color = 'FFFFFFFF') :
232         textScale = float(scale) + 0.3
233         self.kml += """\
234 <Style id="%s">
235 <IconStyle>
236 <Icon>
237 <href>%simages/%s</href>
238 </Icon>
239 <scale> %s </scale>
240 </IconStyle>
241 <LabelStyle>
242 <color> %s </color>
243 <scale> %f </scale>
244 </LabelStyle>
245 </Style>
246 """ % (
247 name,
248 self.kmlFilePath,
249 imgName,
250 scale,
251 color,
252 textScale
253 )
254 
255     def addLineStyle (self, color, width) :
256         self.kml += """\
257 <LineStyle>
258 <color> %s </color>
259 <width> %s </width>
260 </LineStyle>
261 """ % (
262 color, # String
263 width # String
264 )
265 
266     def addPolyStyle (self, color) :
267         self.kml += """\
268 <PolyStyle>
269 <color> %s </color>
270 </PolyStyle>
271 """ % (color)  # String
272 
273     def addPolygon (self, polyPoints, altitudeMode):
274         self.kml +="""\
275 <Polygon>
276 <altitudeMode> %s </altitudeMode>
277 <tessellate> %i </tessellate>
278 <outerBoundaryIs>
279 <LinearRing>
280 <coordinates>
281 """ % ( altitudeMode, 1 )
282         for i in xrange(len(polyPoints)) :
283             self.kml += '%s \n' % (polyPoints[i])  # String
284         self.kml += """\
285 </coordinates>
286 </LinearRing>
287 </outerBoundaryIs>
288 </Polygon>
289 """
```

```
290
291     def addSurface (self, name, description, points, visibility, color) :
292         # adds the line to a kml file
293         self.kml += """\
294 <Placemark>
295 <name> %s </name>
296 <visibility> %i </visibility>
297 <description><![CDATA[ %s ]]></description>
298 <Style>
299 <LineStyle>
300 <color> %s </color>
301 <width> %s </width>
302 </LineStyle>
303 <PolyStyle>
304 <color> %s </color>
305 </PolyStyle>
306 </Style>
307 """ % (
308 name,
309 visibility,
310 description,
311 'FF' + str(color[2:]),
312 '0.5',
313 color
314 )
315         self.kml +="""\
316 <Polygon>
317 <altitudeMode> %s </altitudeMode>
318 <tessellate> %i </tessellate>
319 <outerBoundaryIs>
320 <LinearRing>
321 <coordinates>
322 """ % ( 'clampToGround' , 1 )
323         for i in xrange( len(points) - 1 ) :
324             self.kml += ('%f,%f\n') % (
325                 points[i].coordinate['longitude'],
326                 points[i].coordinate['latitude']
327                 )
328         self.kml += """\
329 </coordinates>
330 </LinearRing>
331 </outerBoundaryIs>
332 </Polygon>
333 </Placemark>
334 """
335
336     def addNetworkLink (self, name, description, isOpen, path,
337                         refreshMode='onChange',refreshInterval=0 ):
338         self.kml += """\
339 <NetworkLink>
340 <name> %s </name>
341 <description><![CDATA[ %s ]]></description>
342 <open> %s </open>
343 <LookAt>
344 <longitude> %s </longitude>
345 <latitude> %s </latitude>
346 <altitude> %s </altitude>
347 <range> %s </range>
348 <tilt> %s </tilt>
349 <heading> %s </heading>
350 </LookAt>
351 <Url>
352 <href> %s </href>
353 <refreshMode> %s </refreshMode>
354 <refreshInterval> %s </refreshInterval>
355 </Url>
356 </NetworkLink>
357 """ % (
358 name, # String
359 description, # String
360 str(isOpen), # String
```

```
361 self.lookAt['longitude'], # String
362 self.lookAt['latitude'], # String
363 self.lookAt['altitude'], # String
364 self.lookAt['range'], # String
365 self.lookAt['tilt'], # String
366 self.lookAt['heading'], # String
367 path,  # String
368 refreshMode,  # String
369 refreshInterval # String
370 )
371
372     def close(self):
373         self.kml += '</Document>\n</kml>'
374         print 'creation du fichier : ' + str(self.name)
375         file = open(self.filePath,"w")
376         with file :
377             file.write(self.kml)
378         print 'fichier: ' + str(self.name) +  ' est cree'
```

/home/manu/DTI/modules/KML.py