UNIVERSITE DE LA MEDITERRANEE
AIX-MARSEILLE II
I.U.T. d'Aix-en-Provence
Département Informatique

RAPPORT DE STAGE

VSM
Interface Homme Machine : Le MCDU

Exemplaire personnel
Exemplaire entreprise
Exemplaire I.U.T

Maximilien MOUSSALLI

Enseignant Responsable : Vincent RISCH
Enseignant Lecteur : Marc LAPORTE

JUIN 2008

Remerciements

Je souhaite avant toute chose remercier plusieurs personnes pour leur aide dans le bon déroulement de mon stage.

Tout d'abord **Jean BENOIT**, Directeur général de VSM, pour m'avoir accueilli au sein de son entreprise et m'avoir confié un travail attrayant.

Gérard CHABERT, chef de projet, pour son aide dans la réalisation de mon travail.

Vincent RISCH, enseignant responsable, pour le suivi de mon stage.

Ainsi qu'à tous ceux qui ont contribués à l'ambiance chaleureuse qui règne dans cette petite entreprise : **Emmanuelle MERCIER**, **Arielle GIRAUD**, **Christophe CASTILLON**, **Laurent GALIBERT** et **Bastien MERINDOL**.

Fiche technique

- **Etudiant :** *Maximilien MOUSSALLI*
- **Année :** *2008*
- **Raison Sociale de l'entreprise :** *VSM Société anonyme*
- **Tuteur de stage :** *Gérard CHABERT*
- **Enseignant responsable :** *Vincent RISCH*

- **Sujet du stage :**
  Extension d'une interface homme machine entre un pilote et le calculateur de navigation d'un aéronef (ht1000[2]) appelé le « MCDU[1] ».

- **Plate-forme informatique :**
  - *Système d'exploitation : Windows*

- **Outils logiciels :**
  - *Logiciel de développement : Microsoft Visual Studio 2005 Professionnel*
  - *Langage : C/C++*

- **Mots clés :**
  - *MCDU*
  - *IHM[3]*
  - *C/C++*
  - *Analyses*
  - *Application graphique*
  - *Projet informatique*
  - *Responsabilité*
  - *Adaptation*
  - *Communication*
  - *Syntaxe*

SOMMAIRE

A) Introduction

Dans le cadre de ma seconde année en IUT dans le département Informatique d'Aix-en-Provence, j'ai été amené à effectuer un stage d'une durée de dix semaines en entreprise, du 7 avril au 13 juin 2008, afin d'acquérir une expérience professionnelle. Au cours de cette période, j'ai réalisé un journal de stage, décrivant mes activités journalières, au sein de l'entreprise qui m'a accueilli.

Le 16 juin 2008, un rapport de stage écrit, doit être rendu. Ce rapport sera lu et noté par un enseignant lecteur. Et enfin, entre le 18 et 23 juin 2008, une soutenance orale notée sera présentée devant un jury.

L'entreprise qui m'a accueilli est appelée « VSM », pour Visualisation Simulation et Modélisation. Elle se situe à Marseille et se charge globalement de réaliser diverses simulations graphiques pour d'autres entreprises. Cette entreprise est localisée dans plusieurs villes en France. L'unité dans laquelle je travaille est composée de trois employés et de cinq stagiaires dont deux de l'IUT Informatique d'Aix-en-Provence (moi inclus) et trois de l'IUT d'Arles.

Le travail qui m'a été confié consiste à implémenter/étendre une interface homme machine entre un pilote et un calculateur de vol d'aéronef appelé « MCDU ». Le langage de programmation utilisé dans le cadre du projet proposé est le C/C++, langage dans lequel l'IUT Informatique d'Aix-en-Provence, donne une importante formation. Ce qui m'a permis une adaptation au projet très rapide. À ma disposition j'avais, tout d'abord mon tuteur de stage Gérard CHABERT pour m'expliquer les détails du projet, des vidéos montrant le produit fini de l'interface et un projet déjà existant sur lequel m'appuyer.

Mon rapport de stage se découpe en trois parties bien distinctes. Tout d'abord, je tâcherai de présenter plus en profondeur l'entreprise dans laquelle j'ai réalisé mon stage, ses activités, ses ressources humaines et techniques. Ensuite, je parlerai plus précisément du travail que j'ai dû réaliser au sein de « VSM », en décrivant le contexte, les problèmes que j'ai rencontrés, les diverses solutions

trouvées, les résultats obtenus, les éléments qui m'ont soutenus dans mon travail qu'ils soient humains, matériels ou logiciels. Enfin, pour finir, je ferai une critique personnelle de mon travail, en me situant d'une part du coté de l'entreprise, d'une autre part de mon point de vue et puis sur la formation que j'ai acquise à l'IUT d'Aix-en-Provence au cours de ces deux dernières années.

B) Résumé de l'introduction en Anglais

As part of my second year at the Computing department of the IUT of Aix-en-Provence. I did a work placement of a duration of ten weeks in a company, from April 7th till June 13th, 2008, to acquire professional experience. At the end of this period, a written report must be produced, followed by an oral presentation.

My host company is called "VSM", for Visualization Simulation and Modelling, and is situated in Marseille. This company has branches in several cities in France. The part of this company in which I work has 3 employees and 5 trainees among which 2 from the IUT of Aix-en-Provence (including me). This company produces aircraft flight simulators.

My work consisted in designing a human-machine interface between a pilot and a flight computer used by aircraft pilots called "MCDU". I had at my disposal videos showing what this interface should look like and the services which it should offer to the user. The first stage of this work was to produce a hierarchical representation of the chain of pages which displays following the entry of the user. Then, I began programming the interface by implementing an already existing project coded in C++.

I met numerous problems, such as :

- A lack of vital information for the production of the interface. Indeed, the videos that were at my disposal did not describe everything! And the persons with whom I worked were not able to help me.
- One of the structures of my program was a little bit rickety on several points, which meant I had to make many changes to work already completed.
- Gaps concerning my knowledge in the programming. Certain parts of the project were hard for me to understand.

In spite of these numerous problems, I found this work placement to be an enriching experience as much in terms of communication as in terms of

programming. Many of my preconceptions proved false. Furthermore, there was a good atmosphere in the office. I noticed, besides, that the knowledge acquired at the IUT was indispensable every day in my work. I feel, now, more in accordance with the world of the work.

C) 1ère partie

Présentation de l'entreprise

et

contexte du stage

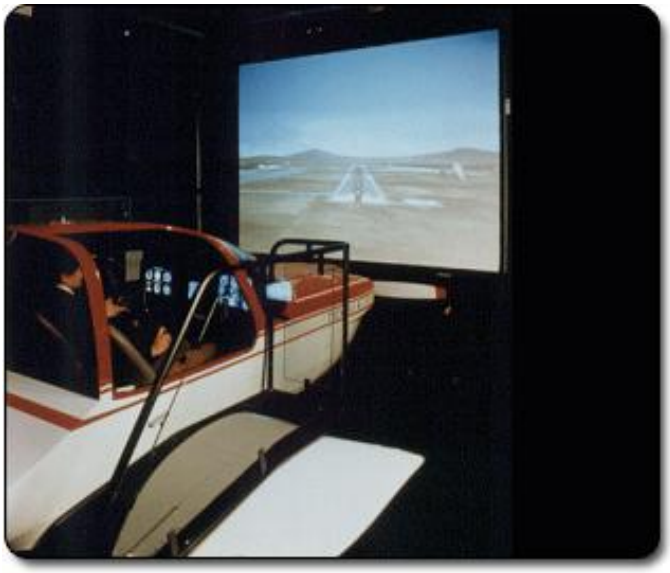
1. Introduction

Dans cette partie je vais présenter l'entreprise « VSM » (Visualisation Simulation Modélisation), société anonyme, répartie dans plusieurs villes en France, et plus précisément j'insisterai sur l'unité de l'entreprise située à Marseille où j'ai effectué mon stage.

Cette première partie de mon rapport de stage est divisée en deux sous-parties. Tout d'abord je ferai une présentation générale de « VSM », regroupant un petit historique de l'entreprise, son secteur d'activité et d'autres précisions importantes. Enfin, je décrirai les moyens informatiques, multimédias et audiovisuels dont dispose cette entreprise.

2. La Société VSM

2.1. Présentation de l'entreprise et de la cellule d'accueil



L'entreprise VSM pour Visualisation Simulation Modélisation est une Société Anonyme fondée en 1988 avec un capital de 245 000 €. Elle est implantée à Marseille, Angoulême et Pelissanne, où est situé le siège social.

En 1995, l'entreprise a reçu le prix Innovation Défense par la Délégation Générale pour l'Armement.

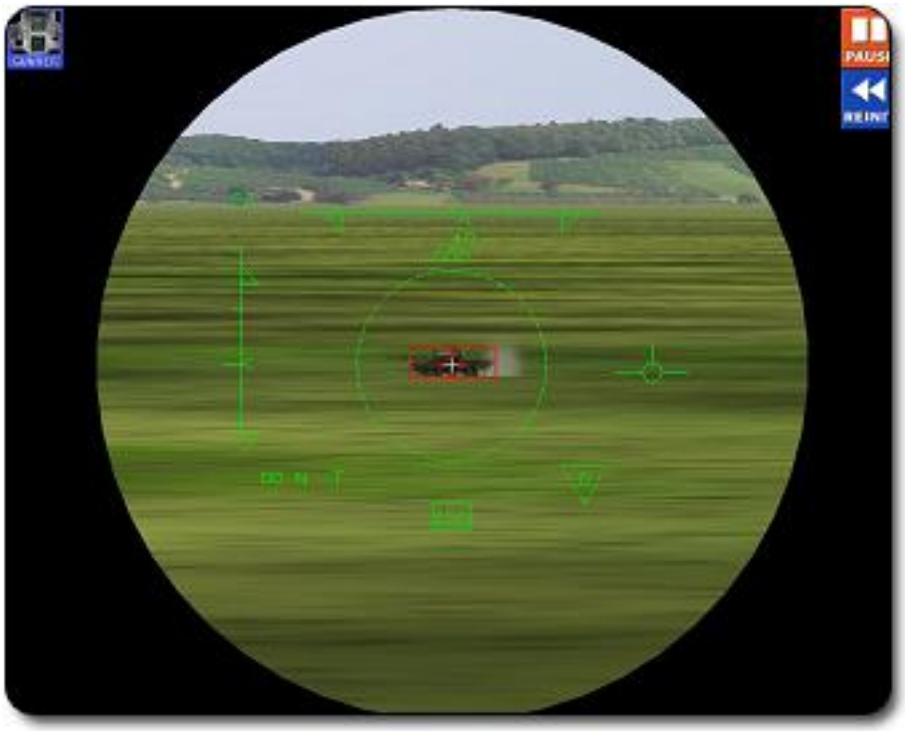
Le métier de VSM se compose de divers domaines tels que les systèmes de simulation temps réel (entraîneurs complets, cockpits et instrumentation simulée, modèles de simulation, structures logicielles temps réel et outils d'exploitation, systèmes visuels, postes instructeurs), les activités de suivi d'essais et les maquettes virtuelles pour l'aménagement du territoire.

Quelques exemples de leurs clients : SOGITEC, AIRBUS, EA-ALAT, QANTAS AIRWAYS, CROSAT, DASSAULT AVIATION, ONERA, DASSAULT, Conseils Généraux et Départementaux, DDE, Mairies, Entreprises du BTP.

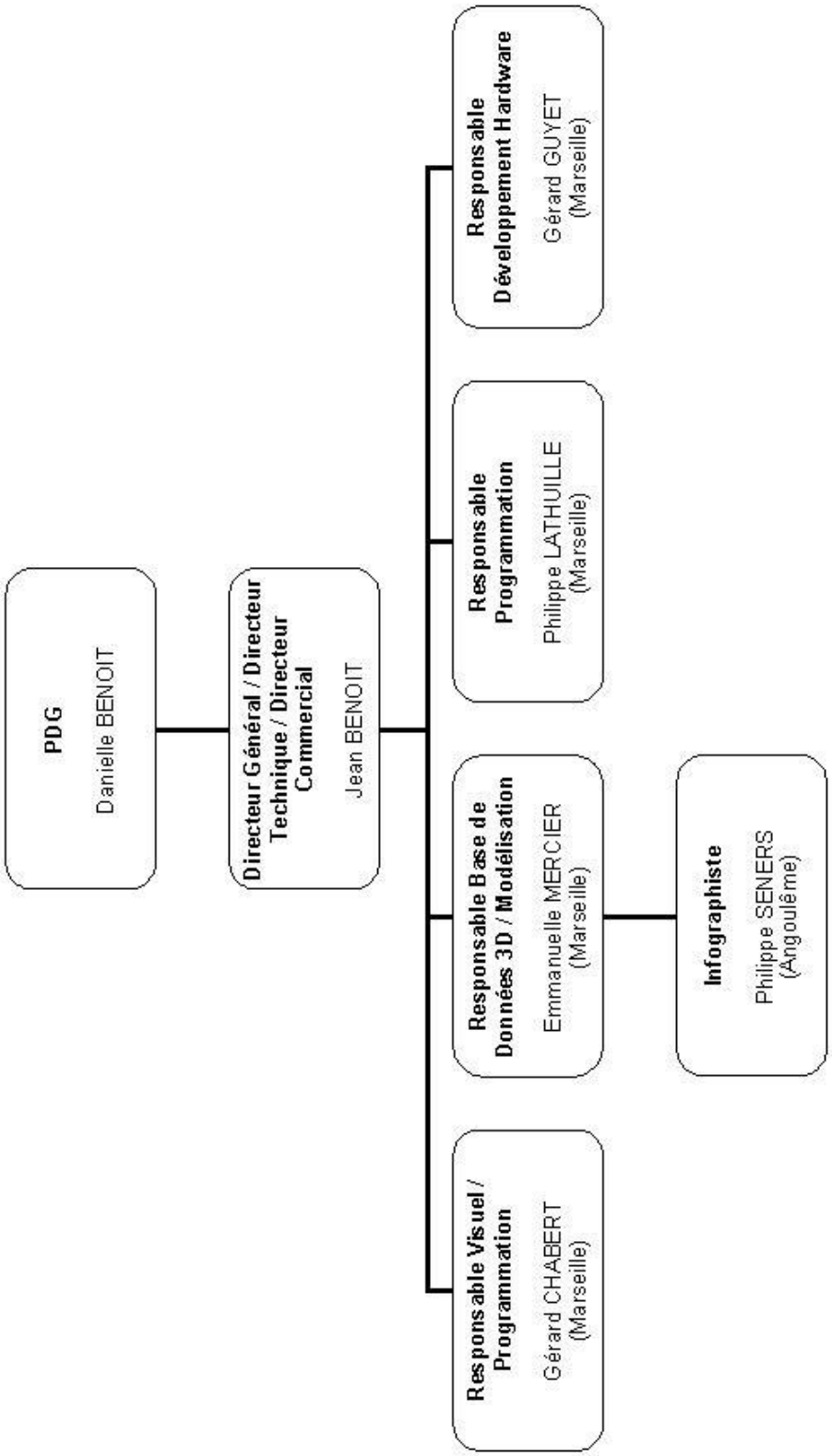
VSM coopère avec les industriels majeurs de la simulation. Quelques exemples de projets : « Projet SPHERE » pour EUROCOPTER (en main d'oeuvre avec BARCO en sous-traitance), « Entraîneurs FENNEC » pour l'Armée de l'Air (en sous-traitance de TT&S), « Projet SAGESSE » pour le CEV (en sous-traitance de CGEY),

« Simulation interarmes » (en sous-traitance de TT&S), et « GNSS Trainer » avec ATR. VSM a conclu une convention de portage avec EADS DEVELOPPEMENT.

Sur le plan de la recherche et du développement, VSM est sur divers projets comme l'étude d'un nouveau concept de visuel pour simulation (DGA – projet REI), l'étude d'aménagement 3D (ANVAR et Conseil Général PACA) ou encore VIPER « Virtuel Interface pour les Process en micro-Electronique » (CEE).



Organisation interne de VSM :



1

2.2. Fonctions et moyens informatiques, multimédias et audiovisuels

Technologies et outils de production :

- Utilisation des technologies standards :
  - WINDOWS et LINUX.
  - Génération des bases de données avec les outils de référence du domaine (MultiGen, Terravista en particulier).
  - Utilisation pour le rendu visuel des outils VEGA et VEGA PRIME.
  - Maîtrise de la bibliothèque Performer.
- Développement d'outils internes « métier » :
  - VISIR pour la visualisation Infra Rouge.
  - ILLUMIN pour la visualisation Intensification de Lumière.
- Expertise de l'environnement SGI / IRIX.
- Expertise du développement de bases de données pour systèmes Evans & Sutherland ESIG.

Outils utilisés pour le projet :

- Système d'exploitation : WINDOWS.
- Développement en C/C++ avec Visual Studio 2005 Professionnel.

D) 2ème partie

Nature de la tâche

et

Travail réalisé

1. Introduction

Dans cette partie, je parlerai du travail que j'ai réalisé au cours de ces dix semaines de stage au sein de « VSM ». Mon travail s'est déroulé en trois étapes, tout d'abord concevoir l'interface du « MCDU » sur un ordinateur classique, ensuite faire fonctionner l'interface en réseau, puis l'adapter avec le « MCDU ».

Ce chapitre est composé de trois sous-parties. La première, est une explication détaillée du sujet qui m'a été proposé pour ce stage. La deuxième, est une description du matériel et des outils logiciels qui m'ont aidé dans mes activités. Et la dernière est une présentation du cheminement de mon travail au cours de ce stage.

2. Présentation du sujet et problématique

Suite à plusieurs « couches » d'explications qui m'ont été fournies au cours du stage, allant de la rapide présentation lors de mon entretien préalable avec Jean BENOIT, aux diverses explications données ensuite par Gérard CHABERT, j'ai pu cerner le sujet qui m'était demandé.

image 1 : *Le « MCDU »*

Globalement le travail à réaliser consistait à réaliser/étendre, en langage C/C++, une interface homme machine (appelé le « MCDU » (voir image 1)) entre un pilote et un calculateur de navigation d'aéronef (le « ht1000 »). Le but du produit fini étant de servir dans une simulation. Le développement de l'interface était donc simplifié et ne communiquait pas vraiment avec le « ht1000 ».

Le « MCDU » comporte un écran sur lequel sont affichées des données relatives au « ht1000 », un clavier alphanumérique pour les saisies utilisateur ('A' sur l'image 2), douze touches disposées sur les cotés de l 'écran (appelées pour le coté gauche de 1L à 6L et pour le coté droit de 1R à 6R), qui permettent une sélection plus intuitive d'un menu, d'un mode, ou d'autres options apparaissant à l'écran ('B' sur l'image 2), de deux touches appelées NEXT et PREV qui permettent le défilement des pages ('C' sur l'image 2), d'autres touches dont je n'ai pas à me préoccuper lors de mon stage, et de quelques diodes (LED[7]) qui servent à définir le statut du « MCDU » ('E' sur l'image). Une partie de l'écran appelée « scratch pad [4]», est réservée à l'affichage des saisies utilisateur, et de certains messages d'erreurs ('D' sur l'image 2).

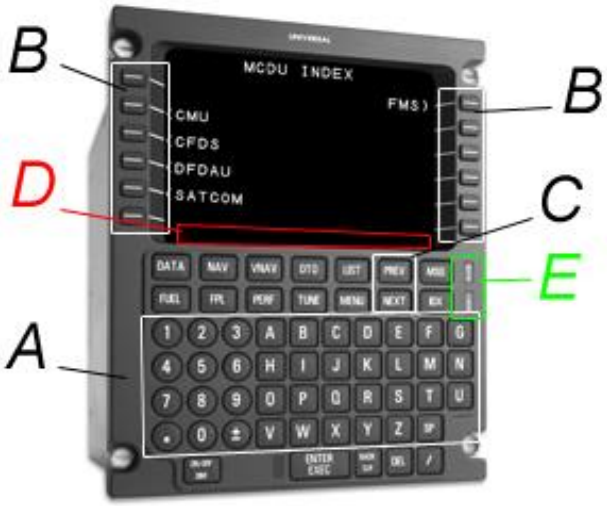


image 2 : *détails du « MCDU »*

Dans mon travail, je devais me charger dans un premier temps, de la création/gestion de pages du « MCDU » (environ une cinquantaine), dont la plupart devaient être « dynamiques ». Au début de mon travail, ces pages devaient être affichées sur l'écran d'un ordinateur classique, à l'aide d'un projet codé en C fourni par « VSM » qui s'occupe de la gestion de l'affichage. Ces pages pouvaient fournir toutes sortes de données relatives au « ht1000 » (date, numéro de vol, rapports, …). A terme, l'utilisateur doit pouvoir avoir une interaction avec ces pages grâce aux touches dont dispose le « MCDU » (dans cette partie du travail, des « boutons Windows » les remplacent). L'utilisateur doit pouvoir également insérer/modifier des données du « ht1000 » et naviguer au travers des pages disponibles dans le « MCDU ».

La seconde partie de mon travail consistait à rendre la partie « *gestion de l'affichage* » et « *acquisitions clavier* » indépendantes « physiquement » de la partie gestion des pages du « MCDU ». C'est à dire que chacune des deux parties doit fonctionner sur un ordinateur différent. Pour réaliser la communication entre les deux ordinateurs, une connexion Ethernet en protocole TCP doit être mise en place.

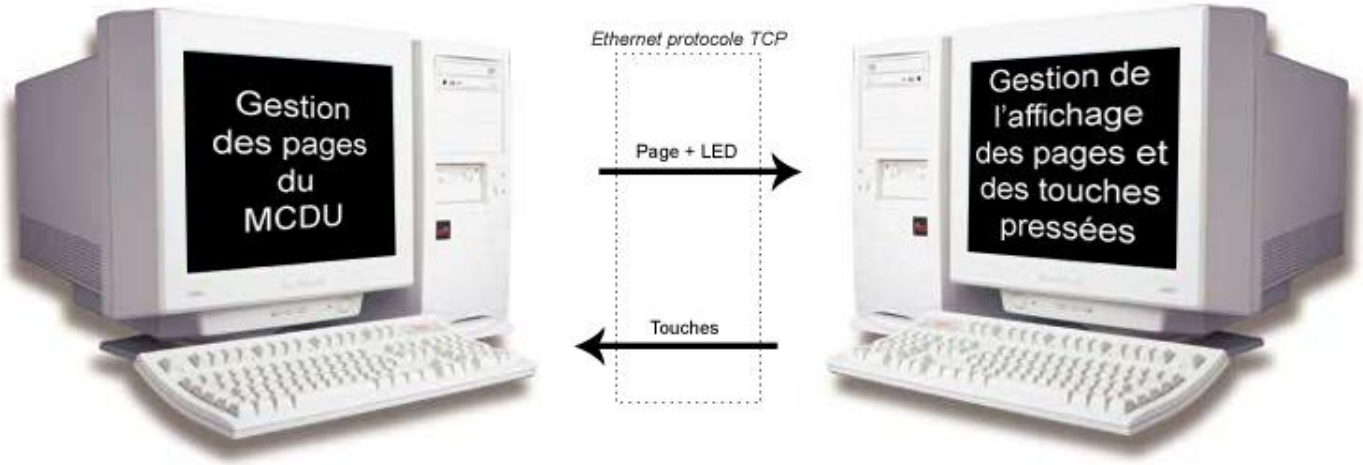


image 3 : *schéma de la connexion Ethernet entre les deux parties de l'interface*

En vérité, le « MCDU » n'est pas un « *système embarqué* », car il ne gère ni la gestion de l'affichage, ni l'analyse des touches, ni l'affichage de ses diodes (LED). En effet, son écran fonctionne, comme n'importe quel écran d'ordinateur, avec un câble « VGA[5] ». Il n'a donc aucune prise en compte de ce qu'il affiche. Les acquisitions clavier du « MCDU » sont, quant à elles, simplement transmises par un câble « USB[6] » (protocole R/S232). Le fonctionnement des « LED » est reçu par ce

même câble « USB ». C'est un ordinateur extérieur au « MCDU » qui devra s'occuper de la gestion des entrées/sorties (*E/S*) du « MCDU ». Il devra contenir les informations des pages, gérer leurs affichages graphiques et les transmettre à l'écran du « MCDU » via un câble « VGA ». Il s'occupera également, de la gestion des acquisitions clavier et des événements qui en résultent, ainsi que de la gestion des « LED ».

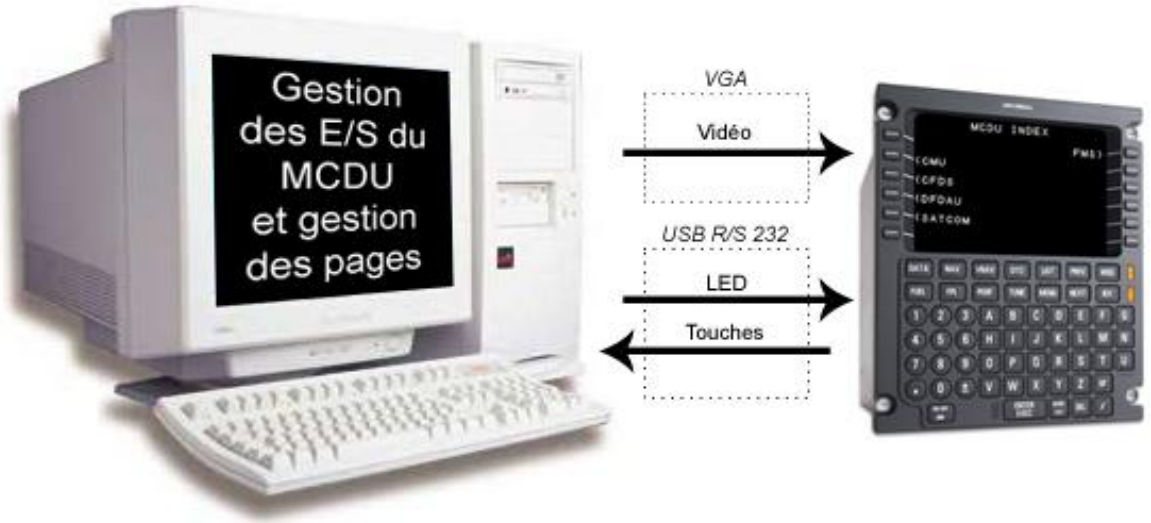


image 4 : *schéma de l'interaction entre un ordinateur et le « MCDU »*

La troisième et dernière partie de mon travail consistait, en me basant sur la partie précédente et d'un projet codé en C fourni par « VSM », chargé de la gestion des entrées/sorties du « MCDU », à réaliser l'interaction entre le « MCDU » et mon interface homme machine. C'est-à-dire, que l'affichage des pages devait se faire sur l'écran du « MCDU » et que je devais analyser les signaux de touches émis par celui ci pour provoquer les traitements adéquats.

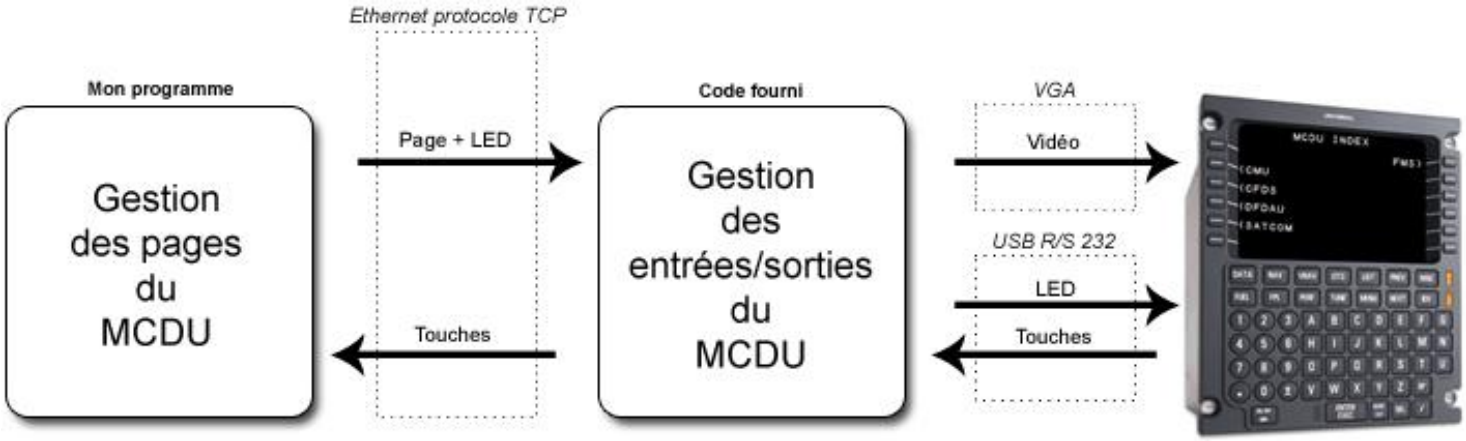


image 5 : *schéma de l'interaction entre les différentes parties de mon travail et le « MCDU »*

3. Le matériel et les outils logiciels

Dans mon travail, plusieurs outils m'ont été indispensables, qu'ils soient matériels ou logiciels. Tout d'abord, les ordinateurs sur lesquels j'ai travaillé, dotés du système d'exploitation « *Windows* » et de l'indispensable « *Microsoft Visual Studio 2005 Professionnel* », logiciel de développement, qui m'a été nécessaire pour coder mon projet en langage C/C++. Ensuite le « MCDU » qui, bien évidemment, m'a permis de tester les résultats de mon travail final. Et enfin les vidéos de soutien qui m'ont présenté les différentes pages du « MCDU » et leurs interactions avec les activités de l'utilisateur.

4. Travail réalisé

Je vais maintenant exposer, en détail , la continuité de mon travail au cours de ces dix semaines de stage dans l'entreprise « VSM », les chemins que j'ai choisis, qu'ils soient bons ou mauvais, la rectification de mes erreurs, les résultats obtenus et autres observations. Ce travail est divisé en trois étapes et elles seront donc présentées séparément. La première étape a été la plus longue, et donc constituera la part la plus volumineuse de ce chapitre.

4.1. Réalisation de l'interface du « MCDU »

Je le rappelle, la première partie de mon travail consistait à réaliser l'interface du « MCDU » mais sur un ordinateur et non sur le « MCDU ». A l'aide de vidéos que l'on m'avait fournies je pouvais voir le produit fini de l'interface. Ces vidéos ont été réalisées par le client, « ATR[8] », de « VSM » qui a commandé la simulation du « MCDU ».

On m'a demandé, dans un premier temps, de vérifier/corriger une représentation graphique en arbre (image 6), déjà existante montrant l'enchaînement des pages en fonction des touches pressées par l'utilisateur, afin de me permettre ainsi qu'à « VSM » plus tard, de s'y retrouver plus facilement dans la hiérarchie des pages.

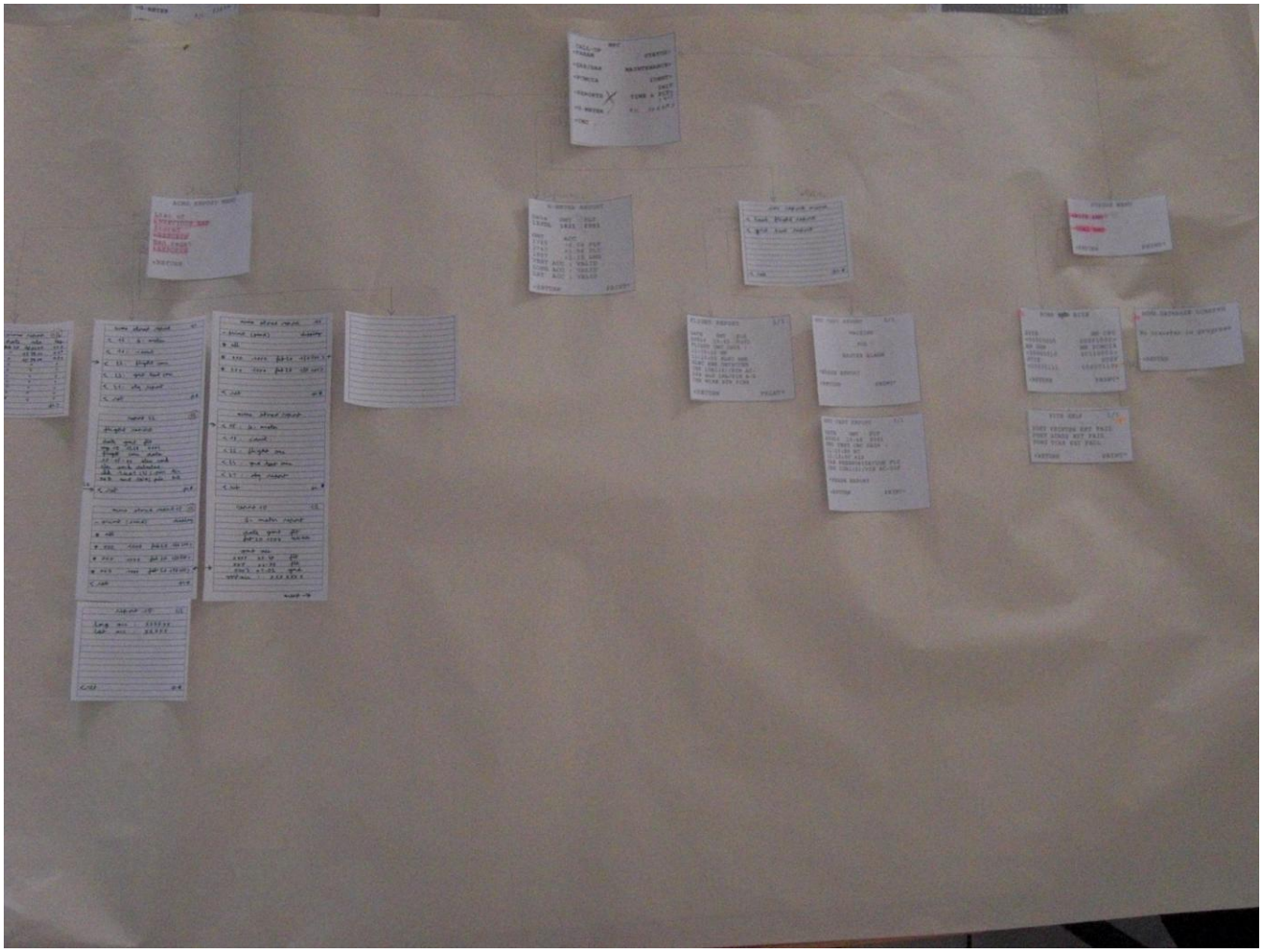


image 6 : *une partie de la représentation graphique de la hiérarchie des pages du « MCDU »*

Après quelques rectifications sur certaines pages et l'ajout de quelques unes manquantes, j'ai constaté que sur les vidéos, des liens n'étaient jamais utilisés et donc les pages qui se cachaient derrière restaient inconnues. J'ai donc dû dresser la liste des pages manquantes, pour que le client « ATR » puisse amener des précisions sur ces pages.

Ce travail étant fait, j'ai pu me lancer vraiment dans le sujet principal : créer l'interface. En plus des vidéos et de la hiérarchie des pages, je disposais d'un petit projet codé en C réalisé par « VSM » qui s'occupait de la Gestion de l'Affichage des Pages, appelons le « GAP[9] ». Après une analyse du code « GAP » et l'aide de Gérard CHABERT, j'ai compris que pour afficher les pages, celles-ci étaient soumises à des contraintes telles qu'être stockées dans un « buffer[10] » de caractères de 14 lignes sur 128 colonnes, et voir leur contenu respecter une syntaxe propre à « GAP » pour permettre des effets de couleur ({cb} ⇒ couleur bleu) ou de taille sur des lettres ({t1} ⇒ taille 1). Il est à noter que dans chaque ligne d'une page, seuls les 24 premiers caractères qui ne participent pas à la syntaxe sont affichés à l'écran. Le mode d'affichage par défaut de « GAP » est une écriture blanche sur fond noir en taille 0.

Exemple :

- Si l'on écrit les lignes suivantes dans le « buffer » :
  - Je suis blanc,
  - {cb}moi je suis bleu,
  - {t1}et en plus je suis BOLD
  - {t0}{cw}Fin

- Le résultat affiché sera :

Après avoir réalisé quelques tests d'affichage pour découvrir les différentes couleurs et tailles de caractères proposées par la syntaxe du projet, j'ai réfléchi sur la meilleure manière d'organiser les pages afin de rendre les liens entre elles faciles à coder, et permettre des actions variables selon les pages.

J'avais donc prévu de faire trois parties de code :

- Une partie appelée « **PageAbstr** » qui servirait de base pour chaque page et qui s'occuperait de stocker/restituer le lien associé à chacune des touches (de 1L à 6R), permettant la restitution de la page correspondante, en fonction des touches pressées par l'utilisateur.
- Une partie appelée « **PageNOMPAGE** » spécialisée pour chaque page et qui s'occuperait de tous les traitements internes de la page et d'écrire quand on lui demande son contenu dans le « buffer ».
- Une partie appelée « **CPage** » qui me servirait d'interface dans mon travail, c'est à dire qui regrouperait l'ensemble des pages et s'occuperait de faire la gestion des changements de pages et de demander à la page courante d'écrire son contenu dans le « buffer ».

Pour les initiés voir *Annexe 1*

Après la réflexion : l'action. J'ai commencé à écrire le code de la structure que j'avais prévue auparavant, mais sans « **CPage** », pour vérifier le fonctionnement de l'imbrication entre les deux autres parties. Le résultat que j'ai obtenu après la réalisation de la première page « statique » (PageMPC) me semblait satisfaisant :

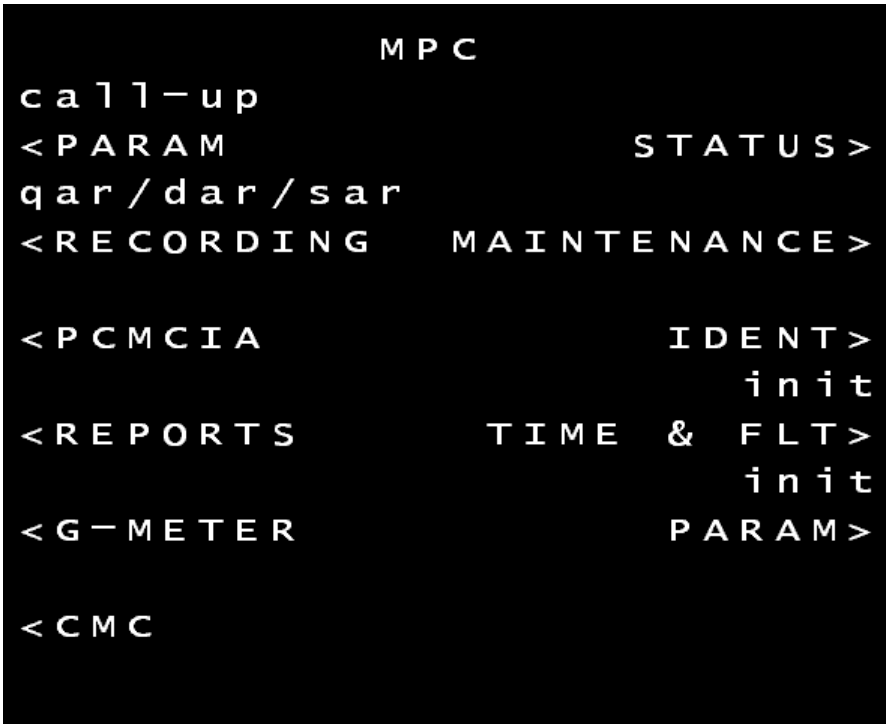


image 7 : *résultat des premiers tests*

j'ai ensuite, rajouté « **CPage** », ainsi qu'une deuxième page « statique » à coder (PageMPC_PAR_C_U) pour essayer de faire la liaison entre ces deux pages. J'ai dû modifier « **CPage** » pour permettre les changements de lien en fonction des touches pressées par l'utilisateur (voir *Annexe 2* pour les initiés).

« GAP » ne possédant pas de bouton graphique pour me permettre de déclencher les changements de page, j'ai donc dû les créer moi-même. Voici le résultat obtenu :

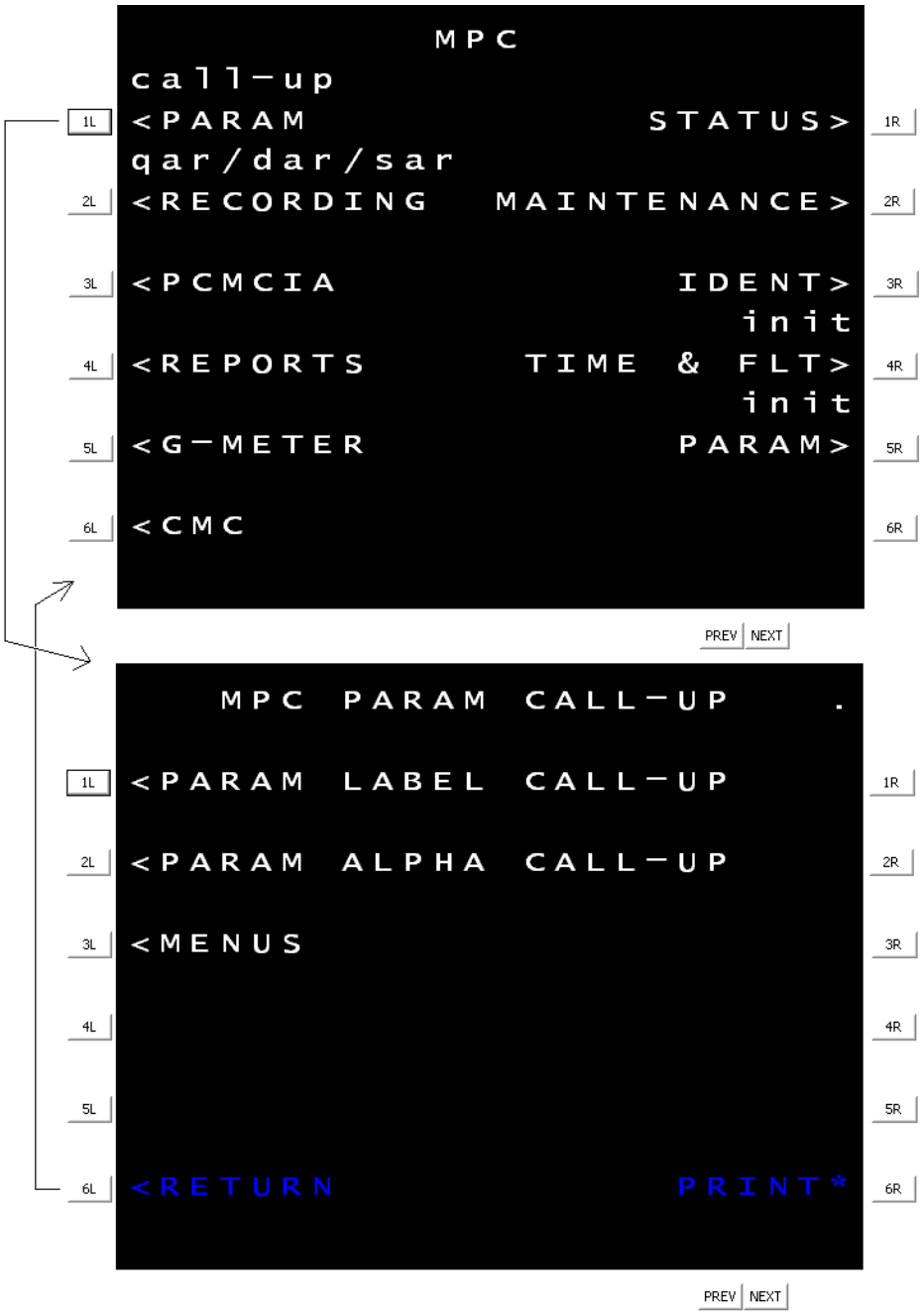


image 8 : *résultat du changement de page*

J'ai ensuite créé une demi-douzaine de pages « statiques » afin de tester la solidité de ma structure globale. Je n'ai rencontré aucun problème, ma structure était donc une bonne base à ce moment là.

Mais qu'en était t'il des pages « dynamiques » ? J'ai donc cherché une page « dynamique » comportant des variables mais restant, en même temps, facile à mettre en oeuvre. J'en ai trouvé une qui permettait à l'utilisateur de régler la date, l'heure et le numéro de vol du « MCDU » que j'appellerai PageT&F.

J'ai commencé à faire PageT&F, sachant que ses variables ne sont pour l'instant (et ce ne sera pas mon travail pour ce stage) pas chargées dans une base de données ou autres. L'utilisateur ne peut pas encore changer ces variables. J'ai donc dû modifier ma structure de la gestion des pages, pour qu'elle prenne en compte des modifications en fonction du texte saisi par l'utilisateur (voir *Annexe 3* pour les initiés). Pour que les modifications soient possibles, j'ai dû ajouter après les boutons Windows, de 1L à 6R, une boite de saisie Windows (textbox) sur l'interface graphique.

Il me fallait, à ce stade, vérifier la validité de la saisie de utilisateur. Par exemple pour l'heure, vérifier qu'il s'agisse exclusivement de chiffres, qu'il comporte bien le bon nombre de chiffres (dans ce cas il nous en faut quatre) et vérifier la validité de l'heure (24h15 n'existe pas).

J'ai eu ensuite quelques questions qui me venaient à l'esprit et dont les vidéos ne me donnaient pas de réponse, telles que :

- Y'a-t-il des messages d'erreurs lors d'une mauvaise saisie ? si oui lesquels et dans quels cas ?
- Quelles sont les analyses syntaxiques qu'il y aurait à faire sur le numéro de vol ?

J'ai posé ces questions à Jean BENOIT et Gérard CHABERT qui n'ont pas pu y répondre. Il faut demander au client « ATR » les réponses. J'ai donc pris l'initiative en attendant, de rajouter des messages d'erreurs lors des mauvaises saisies.

Les résultats obtenus après ce travail :

NB : la saisie utilisateur apparaît sur la dernière ligne au bas de l'écran (scratch pad)

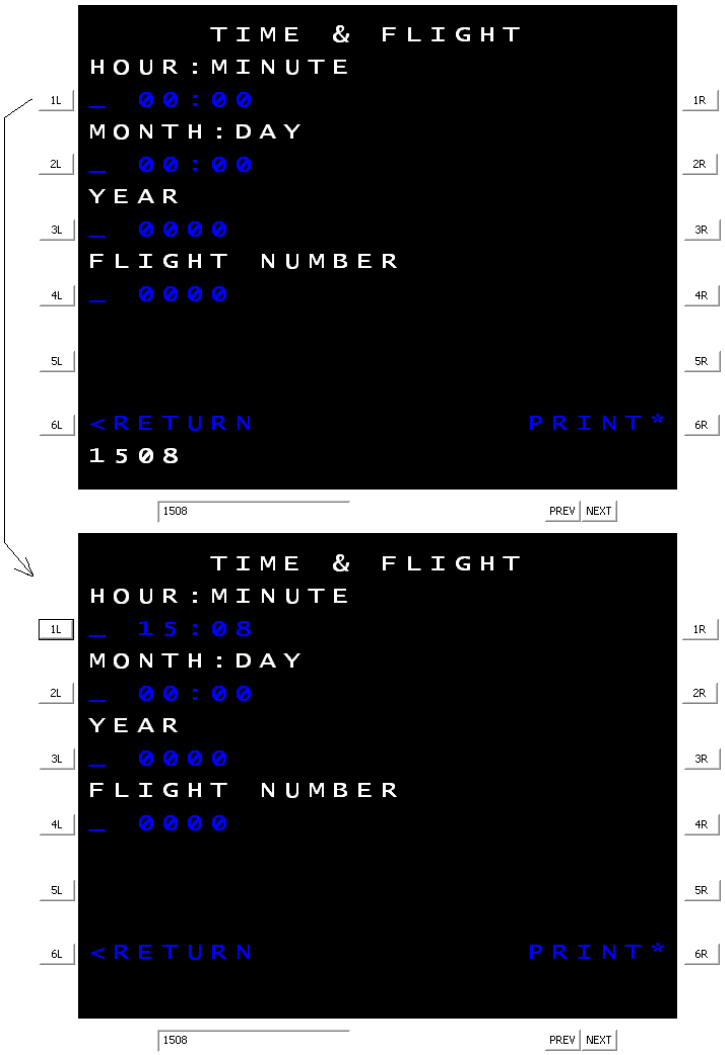


image 9 : *résultat de la modification de l'heure avec 15h08*

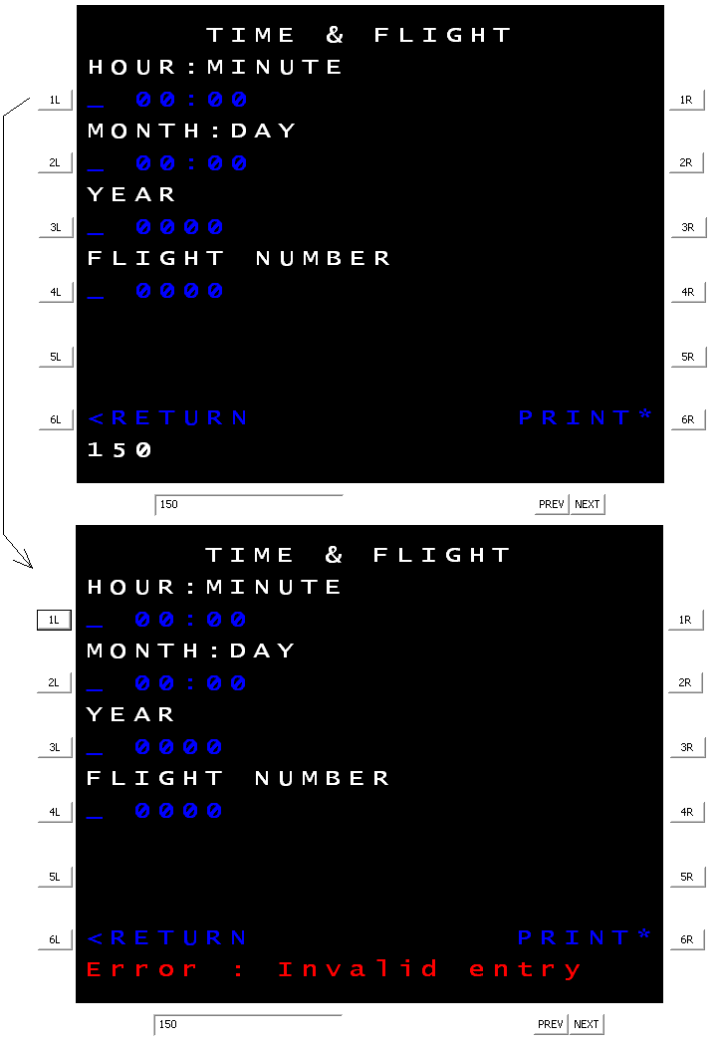


image 10 : *résultat de la modification de l'heure avec une saisie incorrecte*

Je me suis lancé ensuite dans un autre style de page « dynamique ». Il est bon de savoir que si le contenu d'une page dépasse les 14 lignes « autorisées », celle ci se présentera sur plusieurs pages.

J'ai fait en sorte que la page sur laquelle je travaillais, stocke son nombre total de pages et le numéro de la page courante. Ensuite à l'appui de la touche NEXT ou PREV j'ai changé le numéro de la page courante. Ainsi, lorsque « **CPage** » demande à cette page d'écrire dans le « buffer » , apparaissent alors :

- une en-tête, contenant le titre et une indication du numéro de la page par rapport au nombre total de pages
- le contenu de la page correspondante
- le pied de page

Voici le résultat obtenu :

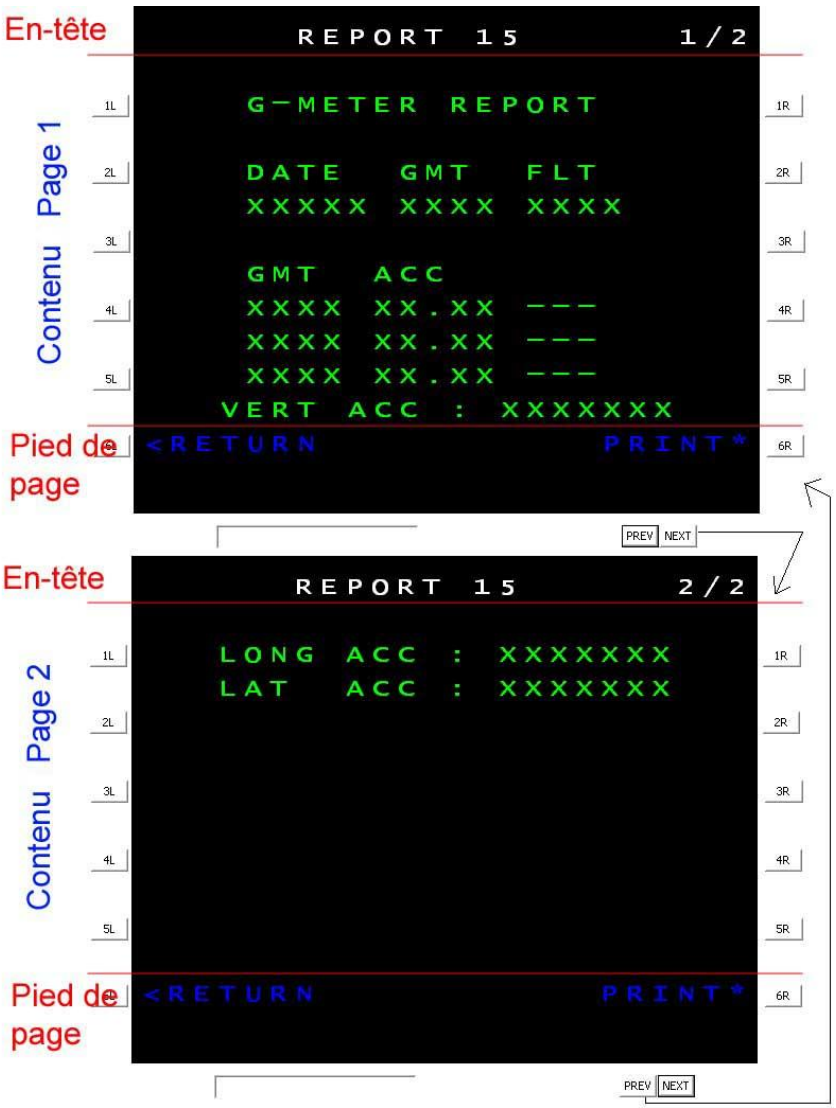


image 11 : *résultat d'une page « dynamique » contenant deux pages*

Dans ma structure, chaque page devait écrire elle même son contenu dans le « buffer », j'ai alors décidé de centraliser la procédure d'écriture dans la partie « **PageAbstr** » pour éviter qu'il y ait redondance.

Ensuite, j'ai dû faire une page qui avait comme propriété de permettre à l'utilisateur d'entrer une valeur pour recevoir un résultat sous forme hexadécimale, décimale, octale et binaire. Donc un problème se pose à moi : Quel est le résultat que renvoie la valeur saisie par l'utilisateur ? J'ai essayé d'avoir une réponse à cette question auprès des personnes qui travaillent dans l'entreprise, mais sans résultat. Encore une question pour le client « ATR ».

Pour les tests de la page j'ai donc considéré que le résultat de la valeur saisie serait la même valeur (en décimale). Je n'ai donc plus qu'à convertir cette valeur sous forme hexadécimale, octale et binaire.

Donc, pour l'exemple, si l'utilisateur saisit « 10 » le résultat sera :

Hexadécimal :	A
Décimal :	10
Octal :	12
Binaire :	1010

J'ai donc essayé d'utiliser les services offerts par le C++ (classe « **streamstring** ») pour récupérer les valeurs sous forme de chaînes de caractères hexadécimales, octales et binaires, mais je ne sais pas pour quelles raisons, lors de la compilation, Visual Studio 2005 ne trouvait pas ces services. J'ai donc dû réaliser moi-même des fonctions qui transforment un entier non signé, en chaînes de caractères sous forme hexadécimales, décimales, octales ou binaires. J'ai créé au passage, une autre fonction qui transforme une chaîne de caractères valides en entier non signé (voir code en *Annexe 4* pour les initiés).

Ensuite le problème avec cette page était que le bloc de saisie/résultat se répétait six fois. J'aurais pu réécrire six fois le même code pour chaque bloc mais cela n'aurait pas été très performant. J'ai donc décidé de permettre de choisir le

nombre de blocs à afficher lors de l'initialisation de la page. Ensuite la page crée « dynamiquement » le nombre adéquat de blocs.

Il y avait une petite partie d'analyse syntaxique à faire pour cette page aussi. J'ai donc empêché la saisie d'autres valeurs que celles numériques. Il y a peut être d'autres contraintes à prendre en compte pour cette page mais c'est encore une question à poser au client « ATR ».

image 12 : résultat de la saisie d'une valeur numérique

Je me suis ensuite, retrouvé face à une page (« PageACMS_STO_REP ») qui me posait quelques petits problèmes. En effet, cette page affichait « dynamiquement » des liens vers d'autre pages. Elle indiquait en fait, la liste de tous les rapports d'erreurs disponibles actuellement dans le « MCDU ». Le nombre de ces liens n'était donc pas connu.

J'ai donc décidé que la partie « **PageAbstr** », qui normalement s'occupe de fournir les liens des pages en fonction des boutons pressés par l'utilisateur, laisserait faire la gestion des liens à « PageACMS_STO_REP » pour qu'elle les fournisse en fonction du nombre de rapports disponibles et du numéro de la page courante, car une même touche chargera deux liens différents selon le numéro de la page.

Pour les initiés voir *Annexe 5*

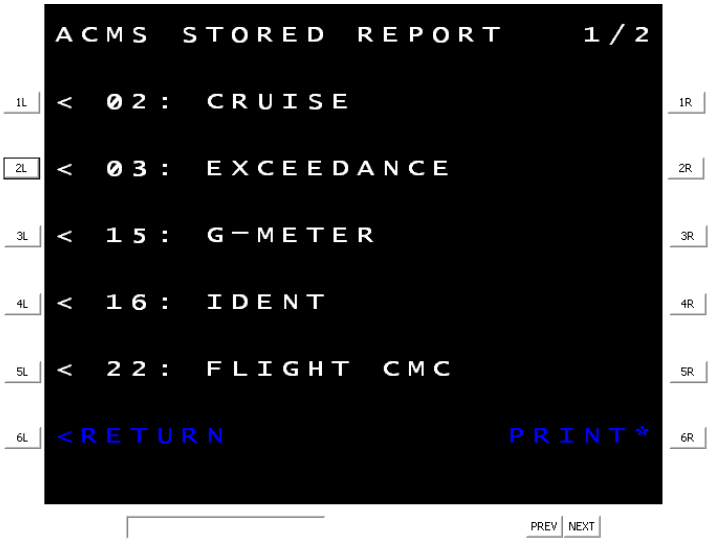


image 13 : *résultat de « PageACMS_STO_REP » page 1/2*

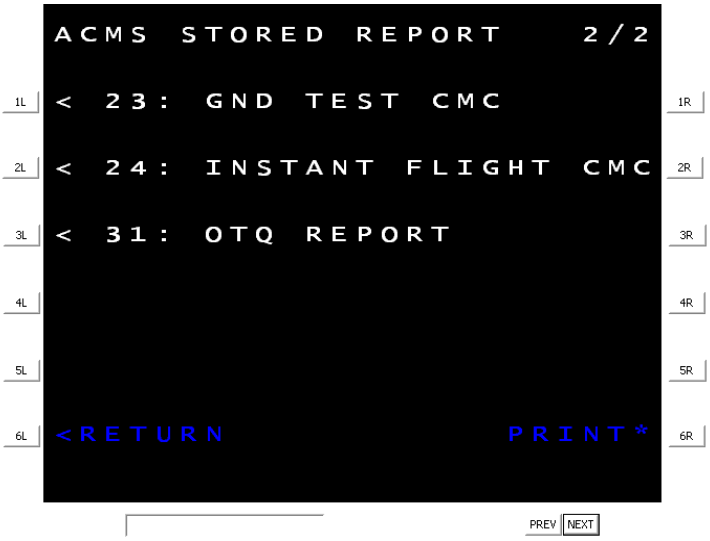


image 14 : *résultat de « PageACMS_STO_REP » page 2/2*

En reprenant les vidéos j'ai remarqué que quelques pages se ressemblaient beaucoup si l'on excluait leur contenu. Leurs titres se résumaient au mot « REPORT » avec un numéro qui l'accompagnait. Et surtout j'ai remarqué que le contenu de ces pages était le contenu d'une autre page. J'ai compris alors, que cette page pouvait charger le contenu d'autres pages.

Je me suis retrouvé bloqué car je n'avais aucun moyen efficace de récupérer le contenu d'une autre page. Je devais donc réaliser une approche légèrement différente de la structure actuelle de mes pages, pour pouvoir permettre à une page de lire le contenu d'une autre page.

Pour réaliser cela, j'ai dû changer la partie qui s'occupait d'écrire dans le « buffer » par une partie qui fournissait juste le contenu de la page. C'est dans « **PageAbstr** » que l'écriture de la page courante dans le « buffer » était effectuée grâce au bout de code qui fournissait le contenu de la page. Cette structure était beaucoup plus souple et j'aurais dû l'appliquer dès le début, car j'ai dû modifier l'ensemble des pages que j'avais déjà créées.

Pour les initiés voir *Annexe 6*

Après les modifications qui ont été apportées à la structure des classes, réaliser la page qui affichait le contenu d'une autre, devenait un jeu d'enfant.

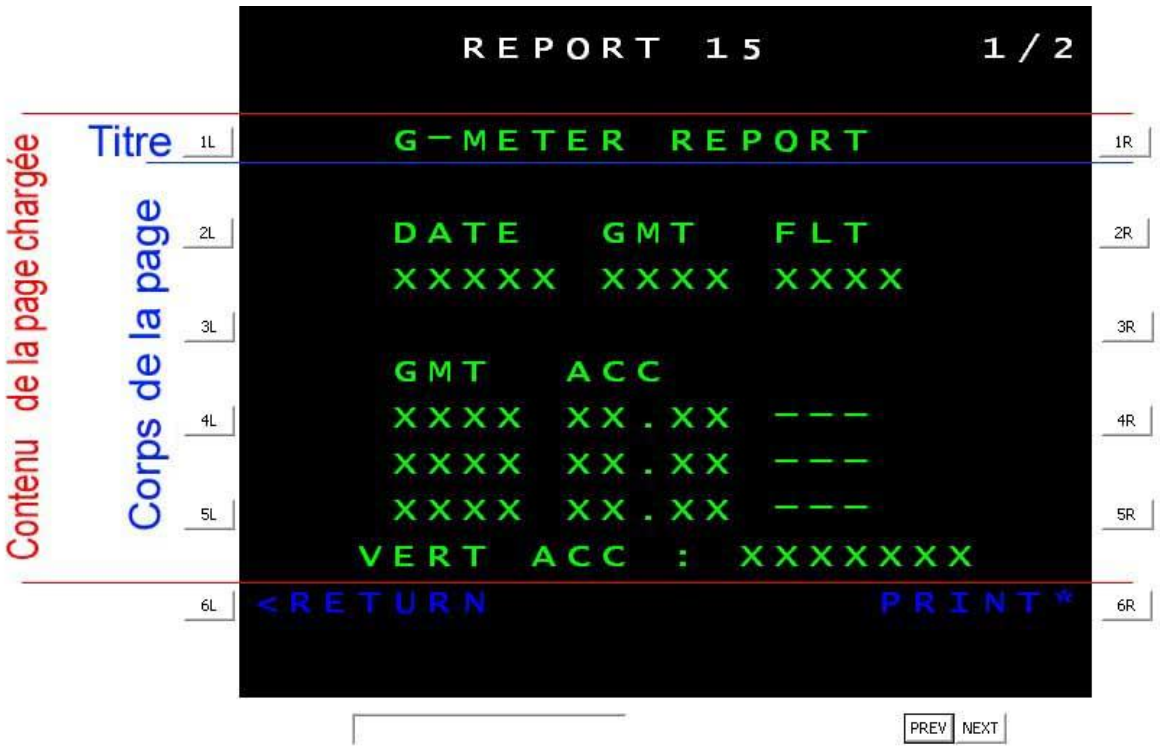


image 15 : *résultat d'une page qui affiche le contenu d'une autre*

Les informations qui suivent sont des petits travaux toujours en rapport avec le premier travail, qui ont été effectués après que j'ai stoppé le premier travail demandé, réalisable à partir des vidéos dont je disposais, et avant qu'on me propose un deuxième travail.

Mon travail étant bloqué, face à toutes les questions sans réponse, on m'a proposé de réaliser un questionnaire à envoyer au client « ATR ». J'ai donc rédigé des questions sur les pages manquantes et des interrogations exposées plus haut.

En réexaminant la structure de mes pages, j'ai constaté que jusqu'à maintenant c'était les pages, elles mêmes, qui s'occupaient d'afficher le système des numérotations de page, dès qu'elles en avaient plusieurs. J'ai donc décidé que ce serait dans « **PageAbstr** » que cette opération s'effectuerait. Cela a permis d'alléger le code des pages.

J'ai fait une nouvelle approche de la structure des pages. A la place de la partie de code qui fournit le contenu de la page entière il aurait été plus intéressant de faire une partie de code qui fournit l'en-tête, une autre le corps et une dernière le pied de page.

Pour les initiés voir *Annexe 7.*

La première partie de mon travail est terminée faute d'informations complémentaires, compte tenu de la durée de mon stage et du temps que mettra le client « ATR » à répondre aux questions que je lui ai posé.

Pour les initiés voir *Annexes 8 et 9*

4.2. Séparation « physique » des systèmes de gestions des pages et de l'affichage

La seconde partie de mon travail consistait à rendre la partie gestion de l'affichage et des acquisitions clavier indépendantes « physiquement » de la partie gestion des pages du « MCDU ». C'est à dire que chacune des deux parties tourne sur un ordinateur différent. Pour réaliser la communication entre les deux ordinateurs, une connexion Ethernet en protocole TCP a été mise en place.

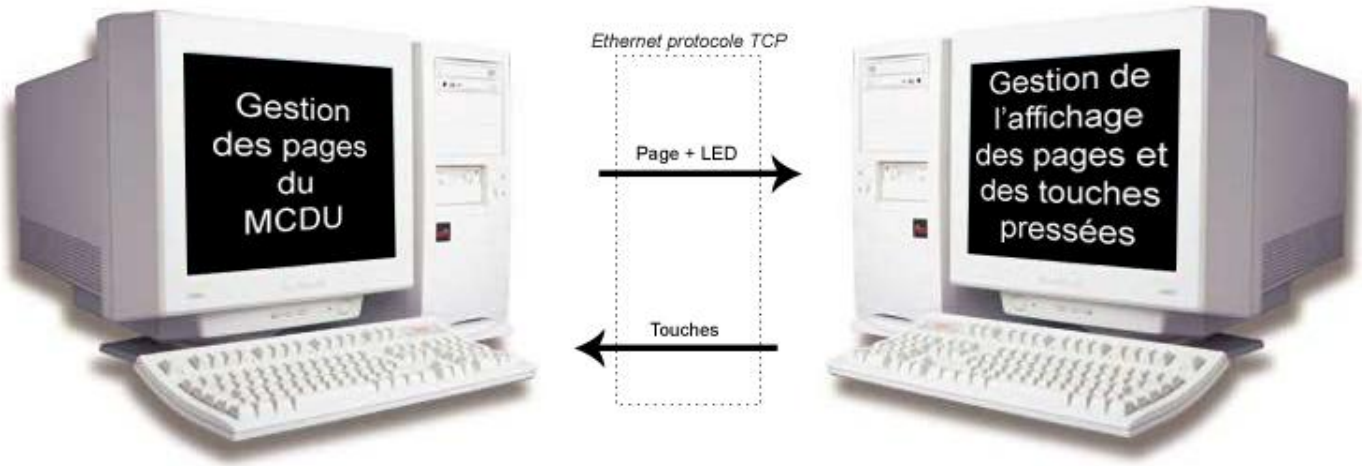


image 3 : schéma de la connexion Ethernet entre les deux parties de l'interface

Heureusement, Gérard CHABERT m'a fourni un projet codé en C qui s'occupe de l'affichage et de la connexion en mode TCP, pour récupérer les pages à afficher. C'est en fait la partie Gestion de l'affichage des pages et des touches pressées présentée sur l'image 3 mais sans une vraie acquisition des touches (j'ai fait des simulations de touches pressées pour les tests) avec un système de communication réseau déjà en place, appelons cette partie « GAP&TP[11] ». Je n'avais donc, plus qu'à me calquer sur le code de « GAP&TP », pour adapter ma partie de Gestion des pages du MCDU (« GP MCDU[12] »), pour qu'elle puisse communiquer avec « GAP&TP ». Je n'ai pas à m'occuper de la partie gestion des LED.

Après avoir lu le code de « GAP&TP » sur les connexions TCP en C, je me suis lancé dans le travail. J'ai gardé pour faciliter mon travail, la partie d'affichage des pages de mon projet afin de contrôler les pages que j'affichais et aussi pourvoir me déplacer dans la hiérarchie des pages grâce aux boutons Windows. Après avoir résolu quelques problèmes de transfert via Ethernet, le résultat obtenu est

satisfaisant. Lors du lancement de « GAP&TP », une petite fenêtre verte se lance. « GAP&TP » est en attente d'une connexion avec « GP MCDU » (voir image 16). Ensuite on lance « GP MCDU », un bip sonore retentit pour chaque partie lorsque qu'elles reçoivent une connexion. Ensuite « GP MCDU » envoie les pages à afficher sur « GAP&TP » et reçoit de « GAP&TP » les touches pressées (aucune pour l'instant) (voir image 17).

image 16 : *après lancement de « GAP&TP »*

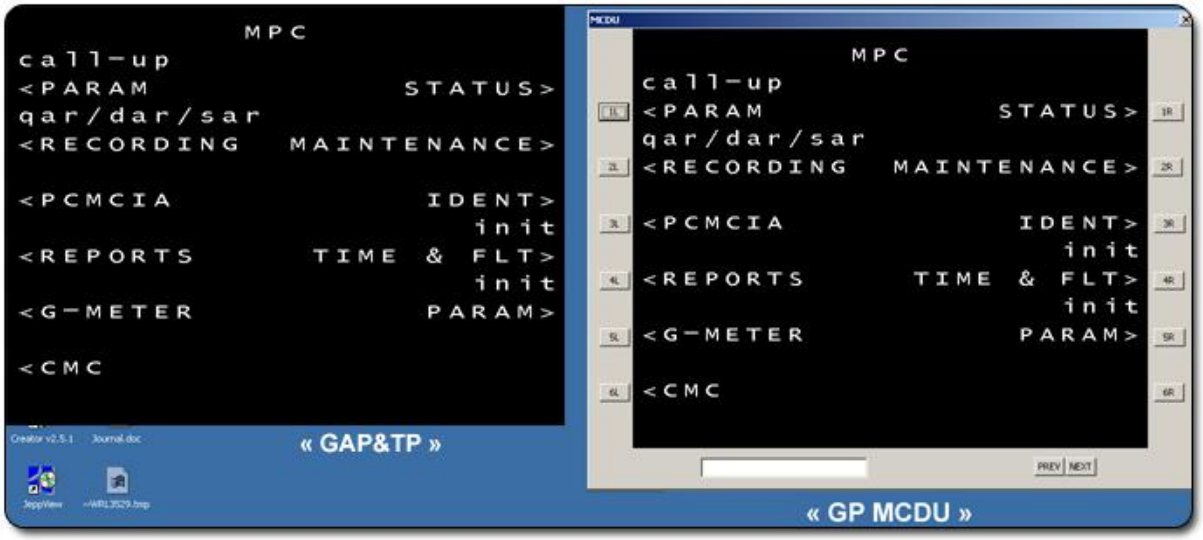


image 17 : *après lancement de « GAP&TP » et de « GP MCDU »*

Après avoir vérifié le fonctionnement de l'affichage des pages, même lors des déplacements dans la hiérarchie (grâce aux touches disponibles dans « GP MCDU »), j'ai testé avec des numéros de touches que« GAP&TP » envoie à « GP MCDU ». J'ai dû résoudre là encore des problèmes de transfert via Ethernet, suivis

d’un résultat concluant. J’ai donc pu me lancer dans la troisième partie de mon travail.

4.3. Adaptation de l'interface sur le « MCDU »

La troisième et dernière partie de mon travail consistait, en me basant sur la partie précédente et d'un projet codé en C fourni par « VSM », chargé de la gestion des entrées/sorties du « MCDU » (« GES MCDU »), à réaliser l'interaction entre le « MCDU » et mon interface homme machine(« GP MCDU[13] »). C'est-à-dire, que l'affichage des pages devait se faire sur l'écran du « MCDU » et que je devais analyser les signaux de touches émis par celui ci pour provoquer les traitements adéquats.

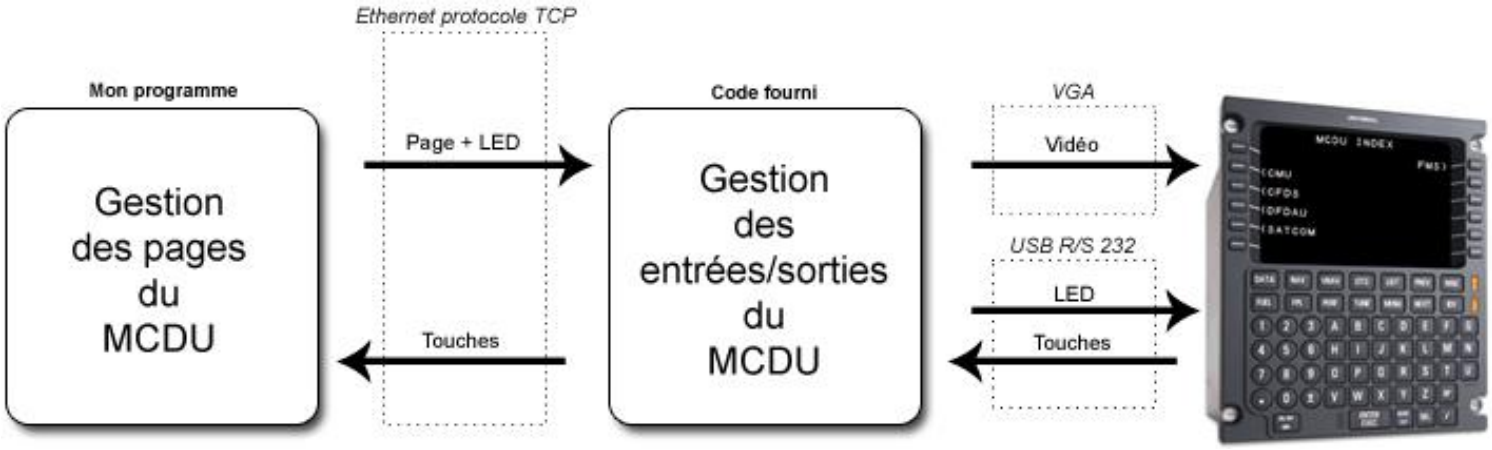


image 5 : schéma de l'interaction entre les différentes parties de mon travail et le « MCDU »

La partie communication via Ethernet a été réalisée dans la deuxième étape de mon travail. Je devais juste réunir « GES MCDU » et « GAP&TP » pour que l'affichage des pages se fasse sur le « MCDU » et que les touches pressées par le « MCDU » soit envoyées à « GP MCDU ».

L'affichage des pages sur le « MCDU » se faisait en considérant son écran comme le deuxième écran de l'ordinateur sur lequel est exécuté « GES MCDU ». Il fallait donc que la carte graphique de l'ordinateur supporte la gestion d'un deuxième écran et enfin d'indiquer la position (X et Y) d'affichage de la fenêtre qui affichait les pages du « MCDU » dans « GES MCDU », pour qu'elle s'affiche sur l'écran du « MCDU ».

image 18 : *Le « MCDU » avant connexion avec « GP MCDU »*

image 19 : *Le « MCDU » après connexion avec « GP MCDU »*

Le « MCDU » utilise une syntaxe pour décrire les touches qui sont pressées :

<01><XX><00>

ou <> représente un octet (unsigned char) représenté sous forme hexadécimale et XX le numéro d'une touche.

Il peut y avoir aussi :

<01><08><00>

qui signale le lâché d'une touche (la fin de pression sur une touche).

Donc « GES MCDU » reçoit du « MCDU » ce genre de signaux et se limite à les transférer à « GP MCDU ». C'est donc dans « GP MCDU » que je devais effectuer une analyse syntaxique qui vérifierait si la syntaxe était correcte avant de valider la touche indiquée. Ensuite, les numéros des touches envoyés par le « MCDU » ne correspondaient pas aux numéros des touches que j'avais attribués dans « GP MCDU », j'ai donc dû les faire correspondre avant que le projet ne soit opérationnel.

Après de nombreux tests, pour vérifier le fonctionnement de chaque touche, mon travail chez « VSM » était terminé.

5. Conclusion

Au cours de ces dix semaines dans l'entreprise « VSM », j'ai pu réaliser le travail qui m'était imparti ainsi que quelques travaux supplémentaires dans les délais demandés.

Par contre, j'ai été grandement surpris de m'apercevoir que l'entreprise ne pouvait pas répondre à grand nombre des questions concernant le travail demandé par « ATR ». En effet, le seul support pour concevoir le produit fini (vidéo fournie par le client), constituait, un élément technique insuffisant pour une telle commande. J'ai trouvé cette situation frustrante.

J'ai été aussi étonné de voir que les projets fournis par « VSM » étaient uniquement codés en langage C et j'ai donc eu quelques problèmes d'adaptation pour comprendre le code étant formé principalement sur le langage C++.

Néanmoins, j'ai beaucoup appris grâce au travail réalisé, et aux obstacles engendrés me permettant de progresser, autant dans le domaine de la programmation (techniques, architectures, …), que dans celui de la communication professionnelle, voire même dans le milieu de l'aéronautique…

E) 3ème partie

Bilan du travail

1. Bilan de l'expérience professionnelle liée à la situation du stage

Lors de mon stage mes capacités ont été mises à rude épreuve : des journées de sept heures devant un ordinateur au cours desquelles le travail à effectuer était un projet assez vaste, qui comportait des parties de code que je ne comprenais pas forcément.

J'ai été confronté à d'importants problèmes lors du développement de mon travail. En effet, j'ai dû parfois modifier, avec beaucoup d'agacement, une grande quantité du travail que j'avais déjà effectué à cause d'éléments que je n'avais pas prévus ou pris en compte dès le début. De plus la moitié de mon travail étant basé sur des données inconnues, sera sûrement pour moitié faux/inutile après réception des réponses d' « ATR » aux questions que je leur ai posé.

Ce stage a été pour moi, autant une épreuve d'endurance que technique.

J'ai pu évaluer mes performances au cours de ses dix semaines, et il m'a semblé avoir effectué, pour un étudiant non expérimenté, un travail consciencieux et correct au sein de cette entreprise, puisque même dans les parties difficiles, je trouvais toujours un moyen d'atteindre les objectifs demandés. De plus j'ai réalisé du travail supplémentaire qui n'avait pas été prévu au début, témoignant que mes compétences étaient en adéquation avec les objectifs demandés. Je me reproche cependant de ne pas avoir suffisamment anticipé les problèmes que la structure de mon travail pouvait engendrer, démontrant mon manque d'expérience dans l'architecture d'un programme.

Je pense que l'entreprise « VSM » à été satisfaite de mon travail, que j'ai essayé de réaliser avec soin. Je n'ai en tout cas pas reçu de critique négative de mon travail. J'ai pu par ailleurs apporter mes connaissances et rendre des services à certaines personnes de l'entreprise.

Malgré la difficulté, ce stage m'a vraiment beaucoup apporté. J'ai pu apprendre à me servir de Microsoft Visual Studio 2005 Professionnel, qui est un outil de développement très répandu et il me sera utile de savoir m'en servir plus tard. J'ai pu mieux assimiler les bases de la programmation qui m'ont été enseignées à l'IUT. J'ai appris comment pouvait fonctionner une interface homme machine. J'ai vu un exemple de milieu professionnel. J'ai expérimenté la difficulté d'exposer mon travail, et les difficultés rencontrées à des personnes non initiées à ce projet.

Je pense donc que ce stage a été une étape importante pour ma formation, et sera un élément clé pour ma future vie professionnelle.

2. Bilan de la formation initiale

Après stage, je réalise à quel point notre formation à l'IUT Informatique est importante. Elle nous fournit tous les éléments pour réaliser un projet dans le développement en langage C++ principalement, mais aussi avec d'autres langages tels que JAVA, C#, SQL.

Je regrette cependant le peu de formation sur le langage C, car je suppose qu'il y a encore beaucoup d'entreprises comme « VSM » qui sont restées axées sur ce langage, ce qui m'a posé quelques problèmes. Mais il est évident qu'en seulement deux années, étudier tous ces langages est déjà proche du miracle et donc apprendre le langage C, plus en détails, deviendrait un acte « divin »…

Je pense que l'IUT Informatique serait plus performant s'il disposait d'une troisième année afin d'éviter de survoler certaines parties de notre formation telles que les langages web PHP, (x)HTML, CSS, AJAX qui bizarrement sont les moins étudiées mais les plus demandées dans les stages, de voir également plus en profondeur certaines parties plutôt lourdes sur le langage C++, (fonctions système, manipulation de pointeurs, masques, fonctions virtuelles, …), ou simplement sur l'algorithme (récursivité, listes chaînées,…) et enfin de disposer de modules supplémentaires de spécialisations tels que des cours sur l'architecture d'un programme, les intelligences artificielles, l'infographie et bien d'autres...

Cependant la formation que j'ai reçu à l'IUT est déjà bien plus importante que je ne l'avais imaginé il y a deux ans quand je m'y suis inscrit. Je reste donc largement satisfait du choix de mon orientation post-bac et je suis fier d'avoir été étudiant à l'IUT Informatique d'Aix-en-Provence et de tout ce que j'y ai reçu.

3. Conclusion

L'IUT informatique, nous apporte de solides bases, mais malgré tout, sans la réalisation d'une expérience professionnelle, ces connaissances resteraient trop fragiles. Le stage que j'ai suivi m'a servi de ciment, pour réunir les morceaux de mes connaissances, qui me semblaient éparpillées. C'est seulement en appliquant régulièrement ce que l'on nous a enseigné qu'on assimile vraiment les choses. L'apprentissage se fait plus instinctivement en entreprise car des responsabilités nous sont confiées et que nous sommes directement confrontés aux fruits de nos erreurs et nous apprenons à travers elles à les résoudre et à ne plus les reproduire.

De plus, grâce au stage, on se familiarise avec le monde du travail. La communication dans une entreprise n'est pas simple à réaliser correctement mais elle est vitale pour que l'entreprise soit dynamique.

Lors de mon stage j'ai appris deux éléments essentiels : la vie en « communauté professionnelle » et la responsabilité.

F) Conclusion

Le stage, un moyen

de

faire le point

Dix semaines se sont écoulées dans l'entreprise « VSM » depuis le début de mon stage, ma vision du monde du travail s'est remodelée jour après jour.

Ce stage a été un gros défi car j'ai été confronté à mes propres limites physiques et intellectuelles, mais j'en suis sorti grandi. J'ai rencontré des personnes sympathiques, qui m'ont aidé dans mon travail et avec qui j'ai passé d'agréables moments pendant mes temps libres. J'ai surtout acquis une expérience professionnelle importante, en terme de communication, de développement et même dans d'autres sujets tel que les IHM ou les aéronefs. Je peux maintenant me baser sur mes propres idées du monde du travail plutôt que sur des idées reçues, je sais à quoi m'attendre pour ma future vie professionnelle.

Grâce à ce stage j'ai pu maîtriser au mieux les connaissances qui m'ont été enseignées lors de ma formation initiale à l'IUT Informatique d'Aix-en-Provence et avoir une vision critique sur celle-ci. Je la trouve relativement complète pour une formation de deux ans.

Ce stage m'a permis de réaliser deux points essentiels. Le premier est que je me suis rendu compte que, pour moi, une future formation en alternance aurait beaucoup d'aspects positifs, puisque ce que j'apprendrais lors de ma formation serait utilisé et donc assimilé dans l'entreprise d'accueil. Le second point est que j'ai réalisé que le développement informatique est un chemin professionnel qui me correspond parfaitement, je me sens alors plus léger car j'ai trouvé ma voie.

G) Lexique

« MCDU [1] » : Le MCDU pour Multifunction Control Display Unit est un instrument qui affiche seulement du texte. Il sert d'interface homme machine, entre un pilote et un calculateur de vol d'aéronef. Il affiche des messages concernant l'appareil et traite les saisies de l'utilisateur à l'aide d'un clavier intégré. le MCDU doit être connecté à un système qui génère son propre affichage de pages et traite les saisies clavier.

« ht1000 [2] » : Le ht1000 est un calculateur de navigation d'aéronef.

« IHM [3] » : Interface Homme Machine.

« scratch pad [4] » : zone sur l'écran où apparaissent les saisies clavier de l'utilisateur.

« VGA [5] » : Un câble VGA (Video Graphics Array) est un câble utilisé pour connecter une carte graphique à un moniteur informatique.

« USB [6] » : L'Universal Serial Bus (USB) est un moyen de communication informatique plug-and-play servant à brancher des périphériques informatiques à un ordinateur pour communiquer en série.

« LED [7] » : Une diode électroluminescente, couramment abrégée sous le sigle DEL, et parfois sous l'anglicisme LED (pour light-emitting diode), est un composant électronique, une diode, capable d'émettre de la lumière lorsqu'il est parcouru par un courant électrique.

« ATR [8] » : est un client de « VSM » spécialisé dans l'aéronautique.

« GAP [9] » : pour Gestion de l'Affichage des Pages est une désignation personnelle pour décrire une partie de code.

« buffer [10] » : système de stockage d'informations

« GAP&TP [11] **» :** pour Gestion de l'Affichage des Pages et des Touches Pressées est une désignation personnelle pour décrire une partie de code.

« GP MCDU [12] **» :** pour Gestion des Pages du MCDU est une désignation personnelle pour décrire une partie de code.

« GES MCDU [13] **» :** pour Gestion des Entrées/Sorties du MCDU est une désignation personnelle pour décrire une partie de code.

H) Bibliographie

Pour le rapport de stage :

Le rapport de stage par *Rober LANKESTER* et *Martine BROCHE.*

Pour les diverses définitions :

Wikipédia le site internet.

Pour les images :

« **VSM** »

I) Journal de stage

Journal de stage

Lundi 7 avril 08

Début du stage à 14h30.

Présentation de l'équipe :

Jean BENOIT, directeur général de « VSM »
Gérard CHABERT, chef de projet
Emmanuelle MERCIER, responsable base de données 3d et modélisation

Une heure d'explication du travail :

Extension d'une interface homme machine (appelé le MCDU (voir image 1)) entre un pilote et le ht1000 (calculateur de navigation).
Grâce à des vidéos je dispose d'une représentation de ce que doit faire le MCDU.
Logiciel de développement : Visual studio 2005 Professionnel.
Langage de programmation : C/C++

Travail effectué : poursuite du travail d'une autre personne. Représentation graphique en arbre des pages à afficher pour le MCDU en fonction des touches pressées.

mardi 8 avril 2008

Plus d'explication sur le projet :

Je vais devoir créer la structure des pages et de leur contenu et gérer les liens entre elles.
Pour afficher les pages, celles-ci sont soumises à la contrainte d'être stockées dans un « buffer » de 14 lignes et 128 colonnes.
Ainsi qu'une sorte de syntaxe pour gérer des effets tels que la couleur et la taille des lettres (ex {cg}, {cb}, {cw}, {t0}, {iv}).
Ex :
Je suis blanc, {cb} moi je suis bleu, {t1} et en plus je suis BOLD {t0}{cw}Fin
Donne :

Toute la partie affichage est déjà créée et utilise le « buffer » en question.
Pour l'instant cela ressemble à un simple écran noir (le buffer est vide).

Travail effectué : vérification de la présence des pages prévues pour le MCDU et correction des éventuelles erreurs sur la représentation des liens des pages qui ont été effectués par la personne qui avait déjà commencé ce travail

Mercredi 9 avril 2008

Fin du premier travail. L'ensemble des pages connues a été traité. J'ai dressé la liste des pages manquantes et M. BENOIT a envoyé un mail pour avoir des données sur celles-ci.

Début du code : Lecture du code existant pour me situer.
Test d'affichage d'un texte en écrivant dans le « buffer ».
Je n'ai pas encore réussi à faire fonctionner l'affichage de ce texte.

jeudi 10 avril 2008

Test d'affichage réussi. Le problème venait d'un mauvais remplissage du « **buffer** ».
Test des différentes couleurs, de la taille des caractères et de l'inverse couleur écran (qui marche mal) que me propose la syntaxe du projet.

J'ai réfléchi sur la meilleur manière d'organiser les pages pour rendre les liens entre elles faciles à coder ainsi que de permettre des actions variables selon la page.

Voir Annexe 1 pour la structure

Ensuite j'ai commencé le codage de la classe « **PageAbstr** », ainsi que d'une classe fille pour faire une page facile (la première page du MCDU) qui joue le rôle de menu.

J'ai testé l'affichage de cette page sans la classe « **CPage** », et le résultat semble satisfaisant (voir image 7)

Vendredi 11 avril 2008

J'ai ensuite codé la classe « **CPage** ».
J'ai fait le lien entre cette classe et la classe de la page que j'avais déjà créée.

Elle semble fonctionner correctement pour l'instant.

J'ai donc ensuite créé une autre page facile à faire pour tester la liaison entre ces deux pages.

J'ai dû modifier la classe « **CPage** » pour permettre de gérer le changement de page.
J'ai rajouté la fonction **Action** qui prend en paramètre le numéro d'une touche et qui grâce à la fonction **GetPage** effectuée sur la page courante, me permet de récupérer le pointeur de l'éventuelle page à laquelle correspond la touche pressée.
Si le pointeur récupéré n'est pas « nul » alors la page courante devient le pointeur récupéré.

Le code déjà existant que l'on ma fourni, ne possède malheureusement pas de boutons pour permettre de déclencher les actions.
J'ai donc dû créer les boutons moi-même et gérer les actions qu'ils devaient provoquer.
Ces boutons finalement ne font qu'appeler la fonction **Action** de la classe « **CPage** » avec comme paramètre le numéro qui représente le bouton.
C'est ensuite cette fonction qui se charge de changer la page courante.

A ma grande surprise mon système marche du premier coup avec cependant un petit problème d'affichage qui doit être réglé. (voir image 8)

lundi 14 avril 2008

J'ai résolu le problème d'affichage.

J'ai ensuite créé une demi-douzaine de pages « statiques » (qui ne font qu'afficher du texte qui ne changera jamais) afin de tester la solidité de ma structure globale.

Je ne rencontre aucun problème, ma structure est donc une bonne base pour l'instant.

Pour les pages « statiques », le travail est fait. Maintenant je dois m'occuper de pages plus ou moins dynamiques.

Je cherche donc une page qui affiche des variables et reste assez facile à mettre en oeuvre.
Je trouve une page qui permet à l'utilisateur de régler la date, l'heure et le numéro de vol du MCDU.

Je commence par faire la page en question qui affiche des variables qui sont des données membres de la classe fille de cette page, qui ne sont pour l'instant (et ce ne sera pas mon travail normalement pour ce stage) pas chargées vraiment. Je les initialise donc toutes à zéro. L'utilisateur ne peut pas changer ces variables pour l'instant.

Je commence donc à visualiser les modifications à faire sur mes différentes classes pour permettre à l'utilisateur de modifier ces variables : Voir annexe 3

La classe fille de ma page qui règle l'heure du MCDU va donc avoir la fonction virtuelle SetAction qui lui permettra en fonction de la touche pressée par l'utilisateur de changer soit l'heure, soit la date, soit le numéro du vol.

Il reste cependant à vérifier la validité de la saisie de l'utilisateur.
Par exemple pour l'heure, de vérifier qu'il s'agit exclusivement de chiffres.
Qu'il y ait bien le bon nombre de chiffres (dans ce cas il nous en faut quatre).
Et enfin de vérifier la validité de l'heure (24h15 n'existe pas).

Pour la date il reste à vérifier que cette date existe.
Pour le numéro de vol je ne sais pas s'il y a une syntaxe particulière.

J'ai donc quelques questions qui me viennent, telles que :
- Y'a-t-il des messages d'erreurs lors d'une mauvaise saisie ? si oui lesquels et dans quels cas ?
- Quelles sont les analyses syntaxiques qu'il y aurait à faire en plus sur le numéro de vol ?

Je les pose donc à M. BENOIT et M.CHABERT, qui malheureusement ne savent pas y répondre.

Je laisse donc de coté pour l'instant ces questions.

Ensuite j'ai dû ajouter un « **textbox** » (boite de saisie pour l'utilisateur) afin de permettre à l'utilisateur une saisie plus facile.
Normalement sur le MCDU il y a un clavier qui écrit directement à l'écran mais cela serait compliqué à mettre en œuvre dans mon travail.
J'ai donc fait que pour chaque modification du « **textbox** » appelle la fonction **SetMessage** de « **CPage** » en lui passant en paramètre la chaîne de caractères que lui-même contient.

Après quelques erreurs que j'ai réglées la page fonctionne correctement.

J'ai pour finir pris l'initiative de rajouter des messages d'erreurs selon les cas.

Voir images 9 et 10

Mardi 15 avril 2008

D'autres pages sont à faire, je choisis une autre sorte de page non dynamique cette fois mais qui comporte plusieurs pages à elle seule. C'est-à-dire qu'on ne change pas vraiment de page lorsqu'on appuie sur les touches NEXT ou PREV pour naviguer entre les différentes pages de cette page.

Pour arriver à faire cela j'ai donc créé la classe fille correspondante qui possède deux données membres :
- Le numéro de la page courante (valeur par défaut : 1).
- Le numéro de la dernière page (valeur par défaut : 1).

Ensuite la fonction **WriteBuffer** de la classe fille écrit dans le buffer :
- Le titre de la page avec le numéro de la page courante et de la dernière page sur la première ligne (ex : Titre 1/2).
- Le corps de la page correspondant à la page courante
- Le « pied de page » qui permet d'indiquer le retour à la page précédente et d'imprimer la page courante sur l'avant dernière ligne (la dernière étant réservée pour la saisie de l'utilisateur et pour d'éventuels messages d'erreurs).

La fonction **SetAction** de la classe fille s'occupe de gérer la transition des pages en incrémentant/décrémentant la donnée membre du numéro de la

page actuelle (en empêchant, bien évidemment, de sortir des limites des pages disponibles). (voir image 11)

J'ai créé quelques pages basiques pour finir la journée.

Mercredi 16 avril 2008

Création d'une page dynamique par son nombre de pages et de variables à afficher et création de quelques pages plus basiques.

jeudi 17 avril 2008

Nettoyage de printemps dans la structure de mes classes, beaucoup de modifications sont effectuées pour optimiser le code. Par exemple j'ai créé la fonction **Write** dans la classe « **PageAbstr** » qui sera appelée par la fonction **WriteBuffer** d'une classe fille. Car avant, chaque classe filles devaient écrire elle même dans le « **buffer** ». Maintenant grâce à cette fonction ce n'est plus le cas. J'ai donc centralisé la fonction d'écriture dans le « **buffer** » dans la classe mère.

J'ai aussi mis en place une gestion de l'affichage entre les messages d'erreurs et les saisies clavier de l'utilisateur qui apparaissent au même endroit.
En fait, avant cela, j'avais décidé qu'un message d'erreur se stocke dans la chaîne de caractères que l'utilisateur a saisie. Je remplaçais donc, le texte par un message d'erreur. Mais ce n'était pas très propre. Maintenant j'ai rajouté une autre chaîne de caractères en donnée membre de la classe « **CPage** » pour les messages d'erreurs.

Ensuite, j'ai dû faire une page qui avait comme propriété de permettre à l'utilisateur d'entrer une valeur et de recevoir un résultat sous forme hexadécimale, décimale, octale et binaire.

Donc un problème se pose à moi :
- Quel est le résultat que renvoie la valeur saisie par l'utilisateur ?

J'ai essayé d'avoir une réponse à cette question auprès des personnes qui travaillent dans l'entreprise, mais sans résultat. Ils ne savent pas non plus.

Pour les tests de la page j'ai donc considéré que le résultat de la valeur saisie serait la même valeur (en décimale).

Je n'ai donc plus qu'à convertir cette valeur sous forme hexadécimale, octale et binaire.

Donc si l'utilisateur saisie « 10 » le résultat sera :

Hexadécimal :	A
Décimal :	10
Octal :	12
Binaire :	1010

J'ai donc essayé d'utiliser la classe « **streamstring** » pour récupérer les valeurs sous forme hexadécimale, octale et binaire, mais je ne sais pas pour quelles raisons, lors de la compilation, visual studio 2005 ne trouvait pas la classe.

J'ai donc dû faire moi même des fonctions qui transforment un entier non signé en chaîne hexadécimale, décimale, octale ou binaire. J'ai utilisé les opérateurs binaires « << » et « & » pour faire certaines de ces fonctions.
J'ai fait aussi une autre fonction qui transforme une chaîne de caractères en entier non signé si la chaîne est valide. (voir code en annexe 1)

Ensuite le problème avec cette page était que le bloc de saisie plus le résultat correspondant se répétaient six fois.
J'aurais pu réécrire six fois le même code pour chaque bloc mais ça n'aurait pas été très souple.
J'ai donc décidé de permettre de choisir le nombre de blocs à afficher lors de l'initialisation de la page.
Ensuite cette page créé dynamiquement le nombre de blocs.

Sur la vidéo qui me montre la page à faire, la zone que l'utilisateur peut modifier est écrite en rouge, mais cette couleur ne faisait pas partie de la syntaxe que l'on m'avait fourni. J'ai demandé l'accord avant de la rajouter.

Il y avait une petite partie d'analyse syntaxique à faire pour cette page aussi. J'ai donc empêché la saisie d'autres valeurs que celles numériques. Il y a peut être d'autres contraintes à prendre en compte pour cette page mais personne ne les connaît encore. (voir image 12)

Vendredi 18 avril 2008

J'ai créé une page quasiment similaire à celle de la veille ainsi que d'autres pages plus faciles à mettre en œuvre.

Lundi 21 avril 2008

Vérification des pages que j'ai déjà créées, et correction des erreurs trouvées aussi bien sur le code que sur le texte affiché sur chaque page.

Mardi 22 avril 2008

Création de pages provoquant beaucoup de questions de ma part, pour comprendre son fonctionnement. N'ayant toujours aucun élément pour répondre à mes questions, je remplis donc avec des valeurs temporaires qui n'ont aucune signification.

Mercredi 23 avril 2008

Création de pages quelconques.

Jeudi 24 avril 2008

Absent pour raison de santé

Vendredi 25 avril 2008

Je me retrouve face à une page qui me pose quelques petits problèmes. Cette page affiche dynamiquement des liens vers d'autres pages. Elle indique en fait, la liste de tous les rapports d'erreurs ou autres disponibles dans le MCDU. Le nombre de ces pages n'est donc pas connu. Je dois donc rendre dynamiques les liens de cette page.

J'ai donc décidé de rendre la fonction **GetPage** de la classe « **PageAbstr** » virtuelle afin de permettre, à cette classe fille, de rendre ses liens dynamiques en fonction de la page actuelle ainsi que des différents rapports disponibles.

Dans la classe « **PageAbstr** » la fonction **GetPage** reste la même qu'avant afin de permettre d'équiper les pages filles plus classiques de liens vers leurs pages.

Donc si l'on est sur la page un ou deux, le bouton « 1L » ne chargera pas le même lien.

J'ai dû créer par ailleurs une nouvelle liste de pointeurs de « **PageAbstr** » spécialement conçue pour cette page car celle disponible actuellement dans la classe mère « **PageAbstr** » est insuffisante. (voir images 13 et 14)

Lundi 28 avril 2008

Un autre problème apparaît, les différents rapports sont affichés dans une page « commune » qui ne fait que changer le numéro (représentant le numéro du rapport) contenu dans le titre et bien sûr, son contenu. De plus cette page charge le contenu exact de certaines autres pages.

Je me retrouve donc bloqué car je n'ai aucun moyen efficace de récupérer le contenu d'une autre page.

Je dois donc réaliser une approche légèrement différente de la structure actuelle de mes pages, pour pouvoir permettre à une page de lire le contenu d'une autre page.

Je décide de remplacer, dans toutes mes classes, la fonction virtuelle « **WriteBuffer »** par la fonction virtuelle « **GetBuffer »** qui ne s'occupe plus, elle, d'écrire dans le « **buffer** » mais renvoie seulement son contenu.

C'est ensuite dans la fonction **Write** de la classe « **PageAbstr** » que j'effectue l'écriture de la page dans le « **buffer** ». La fonction **WriteBuffer** n'existant plus, je profite de la situation pour renommer **Write** en **WriteBuffer**.

J'ai dû modifier beaucoup de mes classes.

Mardi 29 avril 2008

Après les modifications apportées à la structure de mes classes, réaliser la page qui affiche le contenu d'une autre, devient un jeu d'enfant à condition d'avoir le lien de la page concernée.
J'ai donc rajouté un pointeur en donnée membre vers la page qu'elle aura à afficher. Il m'a fallu ensuite créer une fonction qui permet d'indiquer à cette page la page qu'elle doit afficher en modifiant le pointeur.

A force de me retrouver face à des questions sans réponse, on m'a demandé de réaliser une série de questions concernant chaque page que je ne comprenais pas bien ou pas du tout.

Mercredi 30 avril 2008

Après avoir fini de lister les questions que je me posais sur les pages que j'avais déjà effectuées et celles que j'avais entraperçues, M. BENOIT s'est occupé d'envoyer le dossier aux personnes qui ont fait les vidéos afin qu'elles y répondent.

J'ai changé certains éléments dans mes pages pour qu'elles soient plus légères.

Vendredi 2 mai 2008

L'entreprise est fermée pour un pont

Lundi 05 mai 2008

Correction d'un problème qui persistait sur une page.

Jusqu'à maintenant c'était les pages filles qui s'occupaient d'afficher le système des numéros de page, si elles en avaient plusieurs.

J'ai donc décidé que ce serait dans la fonction **WriteBuffer** de la classe « **PageAbstr** » que le numéro de la page courante et de la dernière page de la page sur laquelle on se trouve serait écrit. Cela allége le code de mes pages.

Création d'autres pages dynamiques.

Mardi 06 mai 2008

Création d'une grosse page dynamique et d'autre pages plus basiques.

Mercredi 07 mai 2008

Création de pages.

Vendredi 09 mai 2008

L'entreprise est fermée pour un pont

Mardi 13 mai 2008

Création de pages dynamiques en rapport avec la maintenance.

Mercredi 14 mai 2008

Création des dernières pages pour lesquelles je dispose d'informations.

jeudi 15 mai 2008

Etablissement des nouvelles questions que je me pose sur les pages dernièrement créées, ainsi que la liste de toutes les variables ayant une analyse syntaxique inconnue à faire avant d'être modifiées lors d'une saisie utilisateur.

Vendredi 16 mai 2008

Réflexion sur une nouvelle approche de la structure de mes pages.
Il serait intéressant de faire une fonction pour le titre, une pour le corps et une pour le pied de page. Que j'appellerai **GetHeader**, **GetBody**, et **GetFooter** et qui renverront chacune leurs valeurs propres. Cette structure me parais plus pratique pour les pages contenant, elles mêmes, plusieurs pages, permettant ainsi la répétition du titre et du pied de page automatiquement. Et surtout, cela me permet d'éviter d'écrire 50 fois le même pied de page alors que 99% des pages possèdent le même.

lundi 19 mai 2008

Je travaille à nouveau sur les questions que j'avais réalisées afin de les rendrent plus agréables à lire avant de les envoyer.

Ajout d'un test de validité sur une date entrée grâce à la classe « **COleDateTime** ».

mardi 20 mai 2008

J'ai appliqué le changement de structure de mes pages concernant le remplacement de la fonction **GetBuffer** par les fonctions **GetHeader**, **GetBody**, et **GetFooter**.

Cela a provoqué beaucoup de changements, mais a allégé le code ainsi que simplifié de nombreuses pages dynamiques.

mercredi 21 mai 2008

N'ayant pour l'instant, plus aucune donnée pour continuer mon travail, j'aide M. BENOIT à réaliser des taches quelconques, telles que la modification de certains documents de présentation de l'entreprise.

On m'a confié un nouveau travail à faire. Je dois réaliser la connexion de ma partie avec une autre déjà existante afin d'intégrer ma partie au MCDU. Plus précisément, je doit faire dialoguer ma partie avec l'autre partie qu'on va me fournir via Ethernet (protocole TCP) pour permettre l'échange des touches pressées par l'utilisateur et l'envoie des pages à afficher sur le MCDU.

Jeudi 22 mai 2008

Plus de précision sur le nouveau travail :
Le MCDU possède une entrée vidéo de type VGA qui fonctionne exactement comme un écran et une entrée/sortie USB (R/S 232) qui s'occupe d'envoyer les touches pressées par l'utilisateur et de récupérer les LED (diodes) du MCDU à allumer ou éteindre.

La partie connexion avec le MCDU via le port USB ainsi que l'affichage des pages est géré dans le code que l'on m'a fourni. Ainsi qu'une partie faite pour dialoguer avec la partie que j'ai conçue

Appelons la partie qu'on m'a fournie :
« **Gestion des entrées/sorties du MCDU** » (**GES MCDU**)
et la partie que j'ai réalisée :
« **Gestion des pages du MCDU** » (**GP MCDU**)

Le but final, mais auquel je ne participerai pas, est d'intégrer « **GP MCDU** » dans un projet plus vaste avec une base de données, et tournera sur un ordinateur au sol. Tandis que « **GES MCDU** » tournera sur un ordinateur d'aéronef.

Pour ma part j'ai donc juste, en me calquant sur « **GES MCDU** », à faire la connexion entre les deux parties, pour que « **GP MCDU** » puisse récupérer les touches pressées et envoyer les pages à afficher et éventuellement les LED à allumer (mais j'ai pas à m'en occuper normalement). Une fois les touches récupérées je dois effectuer les actions qu'elles provoqueront sur « **GP MCDU** ».

Voir schéma qui explique l'interconnexion entre les deux parties et le MCDU : image 5.

J'ai, suite à l'explication de mon nouveau travail, lu le code de « **GES MCDU** » pour comprendre le fonctionnement des échanges TCP en C++.

Vendredi 23 mai 2008

Les premiers tests d'échange sont lancés. Pour l'instant le MCDU ne sera pas utilisé et l'affichage se fera sur l'écran de l'ordinateur.
Grâce au code de « **GES MCDU** » j'arrive facilement à faire la connexion entre les deux parties de code puisqu'il me suffit d'utiliser quasiment le même code que dans « **GES MCDU** ».
Je suis averti de la connexion grâce à un double bip sonore qui signale la réussite de la connexion dans les deux parties.

J'ai cependant, des problèmes dans la lecture de la trame de réception et/ou dans l'écriture de la trame d'envoi, car l'affichage d'une page envoyée est pour l'instant assez étrange.
Je résous finalement ce problème et mes pages s'affichent très bien maintenant.

(voir images 16 et 17)

Lundi 26 mai 2008

Je commence à coder la partie qui reçoit les touches envoyées que reçoit « **GP MCDU** ». Je modifie légèrement la partie « **GES MCDU** » pour qu'elle envoie malgré l'absence du MCDU une touche dans la trame ethernet.

Puis, après avoir vérifié que la réception marche avec succès, je code une partie qui analyse les touches reçues et effectue les actions que doivent engendrer ces boutons.
(ex : si la touche NEXT est reçue on essaye d'afficher la page suivante).

Ensuite j'apprends qu'il y a une sorte de syntaxe qui accompagne la touche reçue.
Du genre :

<01><XX><00>

ou <> représente un octet (unsigned char) et XX le numéro d'une touche.
Il peut y avoir aussi :

<01><08><00>

qui signale le lâché d'une touche (la fin de pression sur une touche).

Je code donc une partie analyse syntaxique qui vérifie si cette syntaxe est correcte avant de valider une touche du coté « **GP MCDU** ».
Après de nombreux tests, pour vérifier le fonctionnement de chaque touche, cette étape est terminée.

Mardi 27 mai 2008

Ayant plus ou moins fini l'interconnexion entres les deux parties, je teste l'adaptation du code sur le MCDU. Il faut d'abord installer des drivers pour gérer le dialogue entre le MCDU et l'ordinateur (via USB). Mais manque de chance pour moi, le câble USB (un peu particulier) est défectueux et il doit être réparé. Je ne peux donc pas tester la partie acquisition clavier. Il me reste cependant la partie affichage du MCDU à tester. Qui ne fait rien d'extraordinaire puisqu'elle agit tout simplement comme un deuxième écran. L'affichage fonctionne correctement quand je lance mon application. Je dois maintenant attendre que le câble USB soit réparé pour continuer mon travail.

Mercredi 28 mai 2008

Le câble USB a été réparé, j'ai donc pu tester la partie acquisition clavier. Après quelques rectifications d'erreurs, l'adaptation sur le MCDU est opérationnelle. Le résultat est concluant (voir images 18 et 19).

Du Jeudi 29 mai au Vendredi 13 Juin 2008

J'ai réalisé plusieurs petits travaux utiles à « VSM » puisque le travail que je devais effectuer est terminé tant que j'ai pas de réponse aux questions posées.

Vendredi 13 juin 2008

Petit apéritif festif de fin de stage, Jean BENOIT nous remercie , les autres stagiaires et moi-même du travail accompli et nous demande de lui laisser nos coordonnées, il a un projet d'embauche prochainement et nous contactera pour savoir si le poste qu'il propose nous intéresse.