



– Porphyre 2002 –

DOSSIER DE CONCEPTION

Serveur de Documents

Version 1.1

Référence du document : OC-SERVDOC-CONCEPTION-v1.1

Date de création : 06/05/02

Dernière modification : 18/07/02

Etat du document : validé

Nombre de pages : 30

Rédacteur : Olivier CHADENAT

Équipe : Informatique

Validation : Chef de Projet

Responsable thèse

SOMMAIRE

1	INTRODUCTION	3
1.1	OBJET DU DOCUMENT	3
1.2	PRE-REQUIS	3
1.2.1	<i>Documents</i>	3
1.2.2	<i>Existants</i>	3
1.3	EXIGENCES	3
1.4	GLOSSAIRE	3
2	(PRE-) TRAITEMENTS	4
2.1	IMAGES	4
2.1.1	<i>Etapas du pré-traitement</i>	4
2.1.2	<i>Les différentes fonctionnalités</i>	6
2.1.3	<i>Choix de la technologie à employer</i>	9
2.1.4	<i>Reconnaissance de caractères :</i>	10
2.1.5	<i>Conception de « traitement »</i>	12
2.2	TEXTES	16
2.2.1	<i>Principes du pré-traitement</i>	16
2.2.2	<i>Exemple</i>	17
2.2.3	<i>Caractères spéciaux</i>	19
2.2.4	<i>Nombre de caractères des noms de répertoires</i>	19
2.2.5	<i>Mode d'emploi</i>	19
2.3	CONCEPTION DU CLIENT WEB DE PRE-TRAITEMENT	20
2.3.1	<i>Principe</i>	20
2.3.2	<i>Fonctionnalités</i>	20
2.3.3	<i>IHM</i>	20
3	ARCHIVAGE	21
3.1	PRINCIPE.....	21
3.2	ORGANISATION DES REPERTOIRES	21
3.3	TRAITEMENT : NUMEROTATION	22
3.4	DEMON	23
4	EMISSION	24
4.1	PRINCIPE.....	24
4.2	FONCTIONNALITES	24
4.2.1	<i>Obtenir une description (type, chemin logique et icône)</i>	24
4.2.2	<i>Obtenir le document à l'écran</i>	25
4.2.3	<i>Manipulation sur le document.</i>	25
4.2.4	<i>Inscriptions d'informations sur l'image : Droits d'auteur etc.</i>	28
4.3	IHM	29
	CONCLUSION.....	30

1 Introduction

1.1 Objet du document

Ce document est le dossier de conception du système qui traite les documents et les requêtes qui portent sur les contenus documentaires de Porphyre (image, texte etc..). On appellera ce système serveur de documents par la suite. C'est la suite logique du dossier de spécification. Ce rapport traitera de la conception préliminaire, mais également de la conception détaillée. C'est pourquoi on y inclura les algorithmes principaux et des comparaisons entre logiciels.

1.2 Pré-requis

1.2.1 Documents

L'ensemble de ce document a été rédigé à partir du dossier de spécification rédigé par moi-même.

1.2.2 Existants

Il existe une version réalisée en Java. Cette version ne donne pas pleinement satisfaction, c'est pourquoi on essaiera de faire une nouvelle version qui répond à toutes les exigences.

1.3 Exigences

On rappelle les exigences du cahier des charges qui ont été retenues dans le dossier de spécification:

- La forme d'archivage doit être copiable et modifiable de manière simple
- L'architecture doit être légère, modulaire et utiliser de préférence des bibliothèques et des protocoles pré-existants (c'est à dire standardisés)
- Le serveur doit pouvoir implanter de nombreuses fonctions permettant de réaliser des traitements sur les données
- Le serveur doit pouvoir gérer plusieurs formats de documents et on doit pouvoir intégrer facilement les formats futurs
- Les documents générés par le système doivent être d'assez bonne qualité pour être lisibles. On pense en particulier aux fac-similés de textes.

<p>Pour résumer, le serveur doit être très performant et doit être optimisé pour pouvoir faire des traitements sur un gros volume de données sans que l'utilisateur soit pénalisé. Ceci est très important et on en tiendra compte dans la conception du système et pour les choix qui vont être faits.</p>

1.4 Glossaire

OCR : *Optical Character Recognition (Reconnaissance optique de caractères)*. Logiciel permettant de transformer une image contenant des caractères en un fichier texte éditable.

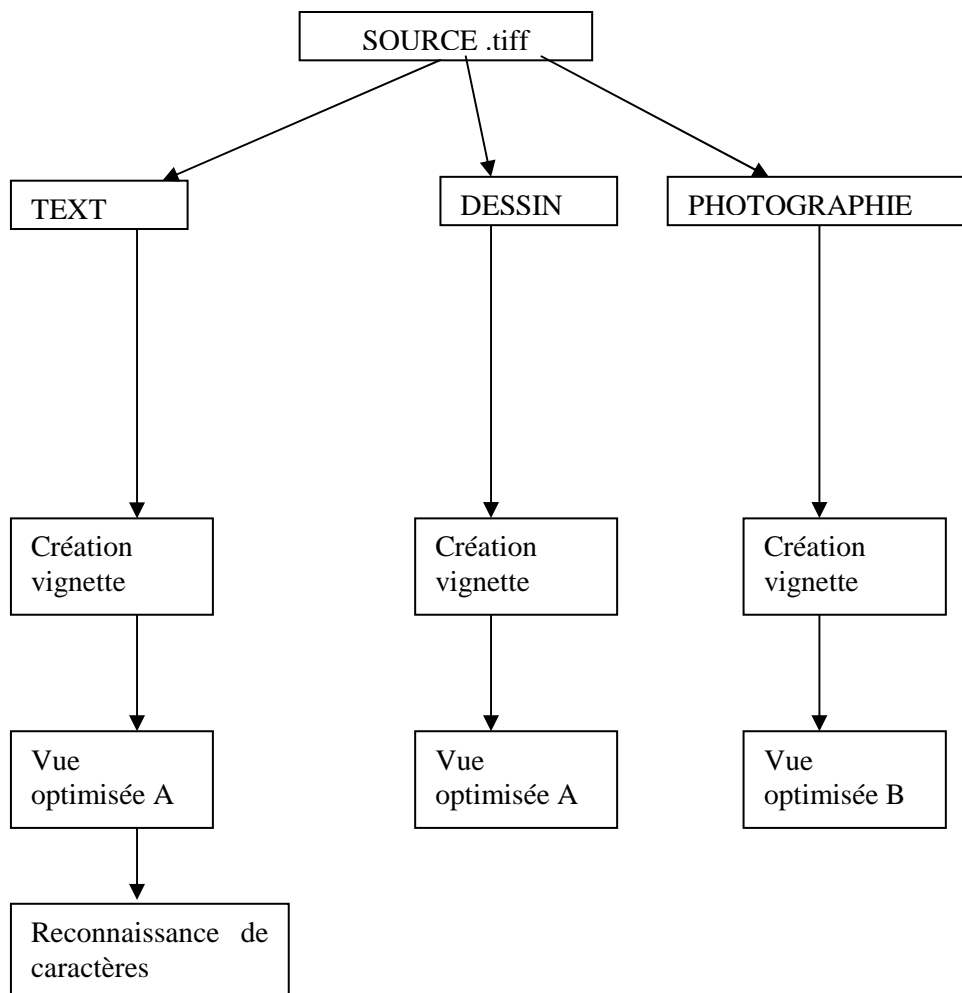
pré-traitement : *Action dont le but est de préparer les documents afin de les traiter de manière optimale par la suite*

2 (Pré-) traitements

Il est plus logique de le faire par scrutation : chaque fois que le système détecte un nouveau document, le pré-traitement adéquat est effectué. Par contre, cela peut poser des problèmes de performances : des essais et des tests doivent être faits. C'est pourquoi, on commencera d'abord par concevoir le système en considérant qu'il doit fonctionner "à la demande". On essaiera d'optimiser au maximum les algorithmes afin que les performances ne s'effondrent pas lorsque le système fonctionnera en scrutation.

2.1 Images

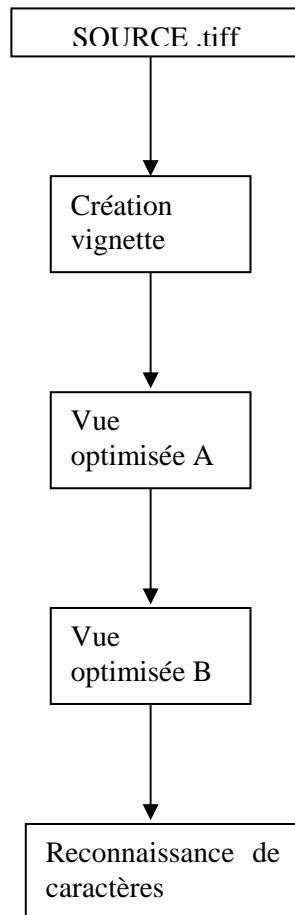
2.1.1 Etapes du pré-traitement



ETAPES THEORIQUES DU PRE-TRAITEMENT

Comme cela a déjà été dit dans le dossier de spécification, il n'est pas prévu que l'on puisse déterminer le type de document à partir des fichiers qui nous sont fournis. La chaîne de production sera donc modifiée : il sera proposé à l'utilisateur une vue optimisée A et une vue optimisée B pour chaque image.

Il faut prendre conscience que ceci handicape le système au niveau des performances (surtout lorsqu'il fonctionne en mode scrutation) et il est conseillé de réfléchir à un moyen de définir le type de document : **le format tiff permet de définir un commentaire dans l'entête...**



ETAPES PRATIQUES DU PRE-TRAITEMENT

2.1.2 Les différentes fonctionnalités

- **Création d'une vignette**

Cette opération est assez simple : elle consiste à prendre une image et à lui faire subir une réduction de taille. On choisira un algorithme assez rapide : la qualité de la vignette n'est pas primordiale : une vignette sert juste à donner un aperçu à l'utilisateur. Une vignette de taille 100*100 apparaît de manière correcte chez la plupart des utilisateurs : il est inutile de créer plusieurs vignettes selon la résolution de l'écran. Le format en sortie sera le format png avec une compression qui reste à définir.

Afin d'améliorer la représentation de la vignette, on ne réalisera pas une vignette de 100*100 mais on procédera de la manière suivante (redimensionnement proportionnel) :

- détermination du côté le plus long : taille X
- on change la taille telle que le côté le plus long soit de 100 pixels
- on change l'autre côté proportionnellement

On peut également se demander si on peut optimiser la taille du fichier. Voici les résultats avec un fichier de test :

Paramètres :

Caractéristiques initiales de l'image

Dimension : 1044 X 1733

Taille : 1164 Ko

Caractéristiques de la vignette sans compression

Dimension : 60X100

Taille : 4Ko

Les essais avec compression n'ont pas donnés de résultats satisfaisants, la taille du fichier restait à 4Ko. On n'appliquera donc pas de compression à la vignette.

- **Conversion en 256 niveaux de gris**

Cette opération peut paraître assez simple, mais on trouve assez peu de bibliothèques ou d'outils qui réalisent cette fonction. Par exemple, Gd n'est pas capable d'effectuer cette conversion. Cette opération consiste à changer la palette de l'image courante. Il existe des algorithmes assez simple à mettre en œuvre, mais ils donnent des résultats assez médiocres c'est pourquoi on essaiera de trouver un outil plus performant pour réaliser cette opération.

- **Réduction en fonction de la largeur des écrans.**

A l'heure actuelle, les grands écrans se démocratisent , et le 17 pouces est devenu l'entrée de gamme. Pourtant, il existe un parc important de machines plus anciennes disposant d'écran 14 ou 15 pouces. Les utilisateurs de porphyre peuvent donc posséder les résolutions suivantes : 640*480, 800*600, 1024*768, 1152*864, 1280*1024, 1400*1050, 1600*1200.

Il est évident qu' il existe une grande différence entre un écran possédant une résolution de 640*480 et un écran affichant en 1600*1200.

Les 2 cas à éviter sont les suivants :

- Il faut éviter qu'une image occupe une trop grande partie de l'écran dans les basses résolutions
- Il faut éviter qu'une image apparaisse trop petite dans les hautes résolutions.

De plus, on notera que pour éviter toutes déformations de l'image, la réduction est proportionnelle (en fonction de la largeur choisie).

Toutefois, il paraît improbable lors d'un pré-traitement par scrutation de réaliser un vue adaptée à chaque résolution puisque l'on ne possède pas forcément l'espace de stockage mais également les ressources machines (vitesse de traitement) pour créer 7 versions d'une même vue.

On peut envisager la solution suivante :

- on réalise pendant le pré-traitement une réduction à 1600 pixels de large
- on réalise les autres réductions à la volée et on peut éventuellement les stocker.

- **conversion en jpeg compressé avec perte**

Paramètres :

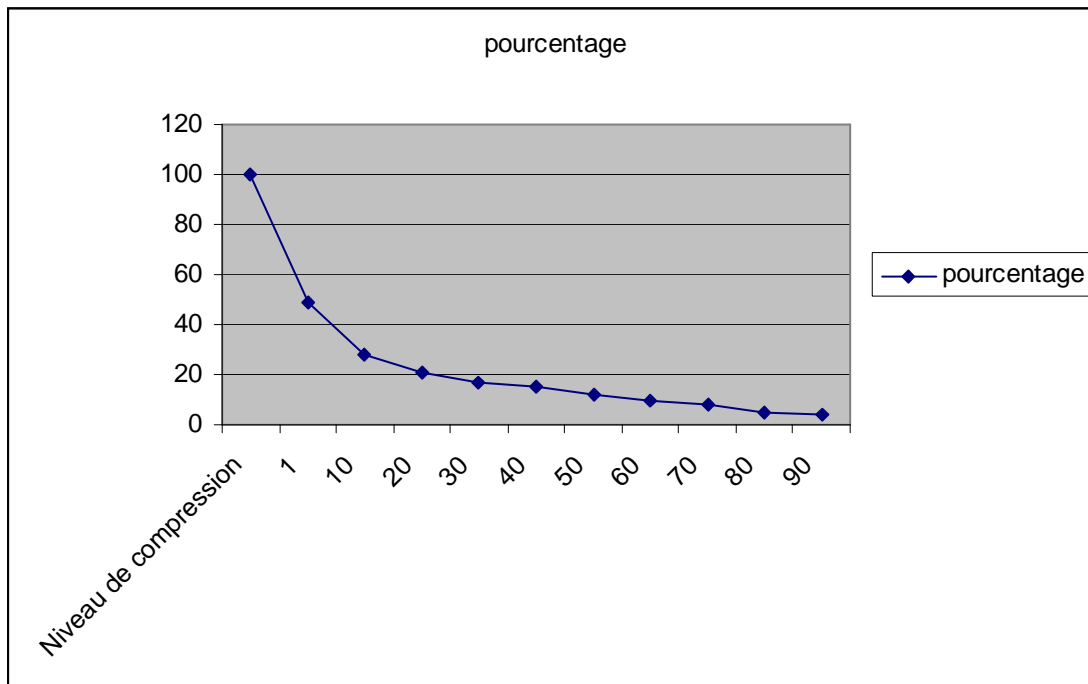
Caractéristiques initiales de l'image

Dimension : 1280 X 960

Taille : 488 Ko

Avec paint shop pro

Niveau de compression	Taille du fichier (en ko)	Pourcentage
1	488	100
10	239	49
20	135	28
30	102	21
40	84	17
50	71	15
60	60	12
70	49	10
80	37	8
90	26	5
99	21	4



D'après le graphique ci dessous et les observations visuelles des images compressées, on peut prendre la décision suivante :

- Si l'espace disque est primordiale, il est recommandé de compresser vers 50-60. A ce niveau là, la qualité de la photographie reste correcte.
- Si on recherche le meilleur compromis qualité-place occupée, on recommande une compression comprise entre 20 et 40.

• **Détachage**



Le détachage consiste à supprimer les tâches noirs qui peuvent apparaître sur les images qui proviennent de documents numérisés. Il existe très peu d'outils capable d'effectuer cette opération.

L'algorithme n'est pas très compliqué mais il demande du temps machine. La technique qui vient à l'idée pour supprimer les grosses taches est la suivante :

Pour chaque pixel, il faut regarder la couleur des pixels voisins. A partir d'un certain seuil de pixel voisin de la même couleur, on peut en déduire la présence d'une tache et on peut donc la supprimer.



Les images ci contre illustre l'opération que l'on veut réaliser.

- **Dé-tramage**

Source d'information : <http://joel.laclavere.online.fr/Pages/Graphisme/Detramage.html>

Lors de la récupération d'images d'origine presse (impression tramée), un tramage devient rapidement visible. Il est impératif de réduire ce dernier afin d'éviter l'apparition d'une résonance de trame lors de du changement de résolution de l'image.

Il n'y a pas de miracle : la diminution de trame ne s'obtient qu'en rendant l'image un peu floue.

Le Flou Gaussien sera préféré dans le cas d'images de type photographique, car il ne rajoute qu'un flou qui "recolle" les points de trames.

Par contre, le filtre Antipoussière est plus efficace sur des dessins ou illustrations, mais il altère légèrement les tracés de l'image en apportant (si on le pousse) un coté un peu "cartoons" au dessins.

2.1.3 Choix de la technologie à employer

Le cahier des charges suggère l'emploi du PHP en ligne de commande ou du C/C++ pour les traitement de haut niveau. Pour des raisons de performances, on choisira le C/C++. Les images à notre disposition sont au format tiff. La bibliothèque GD ne supporte pas ce type d'image et possède dans sa version 2.0 que très peu de fonctions qui nous intéressent. Il manque notamment les fonctions de passage en 256 niveaux de gris, de conversion de format, de compression, de dé tramage et de détachage. De plus, la librairie GD existe sous Windows mais il n'existe aucun support de la part des développeurs.

Conclusion : GD ne convient pas pour le pré traitement, il convient de faire une étude du marché des librairies graphiques.

Les contraintes sont les suivantes :

- la librairie doit être multi-plateforme
- la librairie doit être open source
- la librairie doit être gratuite
- la librairie doit fournir les principales fonctionnalités qui permettent de répondre aux besoins fonctionnels du système. (changement de taille, passage en 256 niveau de gris)

Le meilleur choix apparaît être la librairie ImageMagick : www.imagemagick.org

NB : GD sera toutefois testé pour une utilisation future si cette bibliothèque évolue.

Remarque :

L'utilisation de Gimp (<http://www.gimp.org>) peut s'avérer un choix intéressant.

Temps de réduction en 1600X1200 : 17 s

Temps de réduction en 100X100 : moins d'une seconde

<p>Le principal avantage est le fait que le fichier généré est de très petite taille par rapport au même fichier généré par GD</p>

2.1.4 Reconnaissance de caractères :

Une partie des documents (les fac-similés de texte) nécessite un traitement OCR. Il convient donc de comparer les solutions qui sont disponibles. Les critères de choix sont les mêmes que pour la librairie graphique, mais il faut également que le logiciel est de bonne performance. Le tableau comparatif est donné page suivante.

Il ressort de cette comparaison qu'il existe que 2 solutions sérieuses qui soient gratuites : JOCR/GOOCR et Clara OCR. On testera les 2 pour évaluer leur performance.

	XOCR	JOCR/GOCR	LOCR	ORCchie	OCRE	Clara OCR
sion		0.3.5	0.1.0	?	0.008	0.9.9
ence	Shareware	GPL	GPL	?	GPL	GPL
ix	oui	oui	oui	oui	oui	oui
indows	non	oui	non	non	non	oui
code	?	oui	?	non	?	non
II	?	oui	?	oui	oui	oui
-8859	?	?	?	?	?	partiel
entrée	?	pnm,pgm,pbm,ppm,pcx	pnm	Tiff	PGM/PBM (Tiff)	PBM/PGM (Tiff)
ortie	?	ISO8859, HTML	ASCII	ASCII	ASCII	?
h	?	oui	oui	oui	oui	oui
[?	non	non	oui	non	oui
o client	?	non	non	non	non	oui
ing	?	?	non	oui	oui	oui
sion	?	?	non	?	non	oui
ifont	?	?	?	?	non	non
issement	/	2	5	3	4	1

2.1.5 Conception de « traitement »

Le programme sera écrit en C++. En effet, le C++ a plusieurs avantages par rapport au C dans le cadre de ce projet :

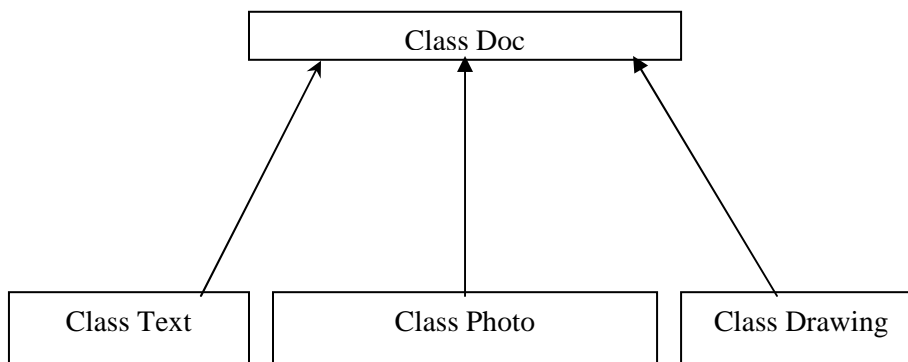
- Les documents auxquels ont fait subir des transformations peuvent être très facilement être gérés comme des objets
- On peut facilement réutiliser des méthodes et l'arbre d'héritage est très facile à obtenir.

Certains peuvent dire que les programmes en C++ s'exécute moins rapidement que les programmes en C, mais notre système n'est pas critique à ce point et il est évident que d'autres optimisations à d'autres endroits permettra d'améliorer de manière plus significative le temps d'exécution.

Le programme devra être portable. Pour vérifier cela, le développement se fera sous Kdevelop 2 (Linux), et on vérifiera fréquemment que la compilation s'exécute bien sous Visual C++ (Windows).

Remarque :

Les noms donnés aux attributs et au méthodes seront en anglais.



ARBRE D'HERITAGE

Cet arbre peut sembler ne pas correspondre aux pistes données par le cahier des charges, mais c'est la solution qui m'a paru la plus logique et la plus évolutive. D'après le cahier des charges, on aurait pu faire hériter la classe Text et la classe Photo d'une même classe puisqu'elles ont la création de la vue A en commun. Mais ce n'est pas très logique.

Classe DOC

Rôle : *C'est la classe ancêtre. Elle gère les documents de tout type et propose de nombreuses méthodes*

Méthodes :

```
// default constructor
Doc ();
// constructor
Doc (string path);
// destructor
virtual ~Doc();

// return the name of the document
string GetName ();
// set the name of the document
void SetName (string nom);
// return the path of the document
string GetPath ();
// set the path of the document
void SetPath (string path);
// create the thumbnail of the document
void Thumb();
// create an optimized view for the document
virtual void View();
```

Attributs :

```
//name of the document
string Name;
// relatif path of the document
string Path;
// file which contain the doc
FILE* DocFile;
// file which contain the thumbnail of the doc
FILE* ThumbFile;
```

Classe DRAWING

Rôle : *C'est la classe qui gère les dessins*

Méthodes :

```
// default constructor
Drawing();
// constructor
Drawing(string path) : Doc(path){View();};
// destructor
~Drawing();
// create an optimized view for the document (view A)
void View();
```

Classe PHOTO

Rôle : *C'est la classe qui gère les photographies*

Méthodes :

```
// constructor
Photo(string path) : Doc(path){View();};
// default constructor
Photo();
// destructor
~Photo();
// create an optimized view for the document (view B)
void View();
```

Classe TEXT

Rôle : *C'est la classe qui gère les images contenant du texte*

Méthodes :

```
// default constructor
Text();
// constructor
Text(string path) : Doc(path){View();OCR();};
// destructor
~Text();
// create an optimized view for the document (view A)
void View();
// processing OCR
void OCR();
```

Attributs :

```
// file which contain the content of the doc after OCR processing
FILE * Content;
```

Le programme doit avoir plusieurs modes d'utilisations :

Soit on l'appelle avec un nom de répertoire pour qu'il fasse le pré traitement sur toutes les images contenu dans le répertoire (sans récursivité). ***traitement -d doc***

Soit on l'appelle avec un nom de répertoire pour qu'il fasse le pré traitement sur toutes les images contenu dans le répertoire et récursivement dans ses sous-répertoires ***traitement -dr doc***

Soit on l'appelle avec une succession de nom de fichier. ***traitement doc/insa.tif doc/olivier.tif***

On peut également envisager de nombreuses autres options comme :

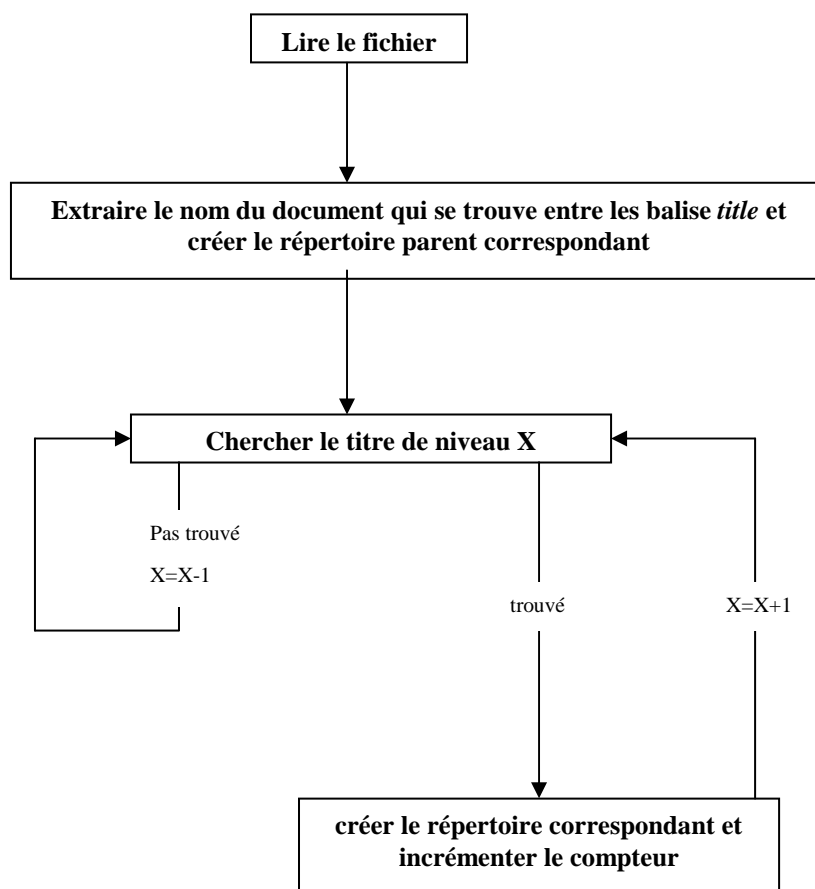
-f : Ecrase les fichiers s'ils existaient déjà

2.2 Textes

2.2.1 Principes du pré-traitement

Le pré-traitement consiste à extraire la structure du document à partir des textes au format XHTML et à créer une hiérarchie de répertoires. Il est nécessaire de s'attarder sur le fonctionnement du XHTML Élémentaire (XHTML Basic) afin de mieux comprendre les besoins fonctionnels du système à mettre en place. Puisque aucun fichier xhtml n'a encore été créé pour porphyre, il convient d'en créer un pour faire les tests mais aussi pour expliquer ce que l'on veut.

Les tests et les explications se feront sur mon CV au format xhtml. Une version du fichier sans les entêtes est donnée page suivante. Le fichier se trouve à l'adresse suivante : <http://porphyre/dev/traitement/text/cv.html>



Pour simplifier, le pré traitement sur les textes xhtml nécessite de réaliser un *parser* xhtml. Vu que la tâche n'est pas insurmontable, il est préférable de le réaliser nous-même et de ne pas en prendre un sur Internet. De cette façon, le *parser* répondra exactement aux exigences. Il est clair qu'il va falloir mettre en place un algorithme récursif que tente de schématiser le dessin ci-dessus.

Notre fichier exemple a été validé comme étant <http://validator.w3.org/>

2.2.2 Exemple

```
<h1>Intro</h1>
<h1>Goal</h1>
<h1>Content</h1>
  <h2>Educational Background</h2>
    <h3>INSA de Lyon</h3>
    <h3>Lycée Claude Fauriel</h3>
  <h2>Foreign Languages</h2>
    <h3>French</h3>
    <h3>English</h3>
    <h3>German</h3>
  <h2>Professional experience</h2>
    <h3>The French School at Athens</h3>
      <p>Design and Development within the Porphyre project (Digital Library)
      Porphyre is a system of research of information to help the reader of scholarly publications.
      </p>
    <h3>Etranges Libellules</h3>
      <p>Analysis of the characteristics of the GameBoy. Designed and carry out a software in order to develop
      video games.
      </p>
    <h3>France Télécom</h3>
      <p>Development of a web site Modelling of the system information , setting up of a data base and development of
      HTML, ASP, Javascript, SQL, Access 97
      </p>
    <h3>Casino</h3>
  <h2>Computer Skills</h2>
    <h3>Languages and DBMS</h3>
      <p>Java, C/C++, Assembly for 68000, Pascal, Oracle, SQL, Prolog, HTML, ASP, PHP, MySQL, Access</p>
    <h3>Operating systems</h3>
      <p>Linux (Red Hat/Mandrake), VxWorks (real time OS), Windows (NT4-2000)</p>
    <h3>Networks</h3>
      <p>ISO architecture, Internet architecture, LAN</p>
    <h3>Design methods</h3>
      <p>Entity-relationship, Object oriented approach (UML language, USDP method), Merise</p>
    <h3>Tools</h3>
      <p>Visual C++, Kdevelop (AGL C/C++ pour Unix), Delphi, Jbuilder, Matlab</p>
    <h3>Mathematics</h3>
      <p>probability, statistics, numerical analysis, signal processing, artificial intelligence, discrete
      mathematics, encryption</p>
  <h2>Personal information</h2>
    <h3>Sport</h3>
      <p>body building, Football</p>
    <h3>Driving licence</h3>
<h1>Conclusion</h1>
```

<p><i>an outline for this document</i> généré par le validateur</p>	<p>Ce que l'on veut obtenir</p>
<ul style="list-style-type: none"> • Intro • Goal • Content <ul style="list-style-type: none"> ○ Educational Background <ul style="list-style-type: none"> ▪ INSA de Lyon ▪ Lycée Claude Fauriel ○ Foreign Languages <ul style="list-style-type: none"> ▪ French ▪ English ▪ German ○ Professional experience <ul style="list-style-type: none"> ▪ The French School at Athens ▪ Etranges Libellules ▪ France Télécom ▪ Casino ○ Computer Skills <ul style="list-style-type: none"> ▪ Languages and DBMS ▪ Operating systems ▪ Networks ▪ Design methods ▪ Tools ▪ Mathematics ○ Personal information <ul style="list-style-type: none"> ▪ Sport ▪ Driving licence • Conclusion 	

Il faut aussi noter que l'on doit retrouver dans les répertoires terminaux des fichiers qui disposent du contenu.

Le système de numérotation permet de retrouver facilement un contenu. Par exemple, si je désire avoir accès aux OS que je maîtrise, il suffit que je fasse une requête en indiquant : 2.3.1

Le moteur de recherche de document sera développer en même temps que le *parser* pour vérifier la cohérence de l'ensemble.

2.2.3 Caractères spéciaux

Les titres des documents XHTML sont transformés en répertoire. Ces titres peuvent contenir des caractères qui ne sont pas acceptés dans les noms de répertoires. Il convient donc de lister ces caractères spéciaux et de les convertir. Voici la table de conversion :

Caractère	Interdit sous	Remplacé par
\	Win	blanc
/	Win	blanc
:	Win	blanc
*	Win	blanc
?	Win	blanc
“	Win	blanc
<	Win	blanc
>	Win	blanc
	Win	blanc

Il est à noter que dans le moteur de recherche, on sera confronté à un autre problème : celui des noms de répertoires contenant des espaces. En effet, un lien hypertexte ne peut pas contenir d'espace. Il faudra remplacer les espaces dans le lien pas %20

2.2.4 Nombre de caractères des noms de répertoires

Extrait de l'aide Windows 2000

*Un nom de fichier peut contenir jusqu'à **215 caractères**, y compris des espaces. Cependant, il n'est pas conseillé de créer des noms de fichiers de 215 caractères. La plupart des [programmes](#) ne peuvent pas interpréter les noms de fichiers extrêmement longs. Les noms de fichiers ne peuvent contenir aucun des caractères suivants :*

`\ / : * ? " < > |`

Extrait de l'aide Windows NT

*Un nom de fichier peut contenir jusqu'à **255 caractères**, y compris des espaces. Mais il ne peut contenir aucun des caractères suivants : \ ? : * ? " < > |*

Il faut donc limiter les noms de répertoires à 215 caractères. Comme le pré traitement est automatisé, si un nom dépasse les 215 caractères, il sera tronqué.

2.2.5 Mode d'emploi

Le parser prend en entrée un fichier XHTML valide. En effet, le parser ne valide pas le document. C'est à celui qui produit le fichier XHTML de s'assurer de la validité de son document.

2.3 Conception du Client Web de pré-traitement

2.3.1 Principe

Nous avons vu que le pré-traitement pouvait se faire à la demande ou par scrutation. Avec la réalisation d'un client WEB nous avons à notre la possibilité de réaliser ce pré-traitement. Le principe est de demander à l'utilisateur qu'elles sont es fichiers qu'il veut pré-traiter. Si ce sont des fichiers XHTML, on fera appel au script précédemment décrit. Si ce sont des fichiers images, on exécutera notre programme traitement.

2.3.2 Fonctionnalités

Le client Web est assez simple, puisqu'il s'agit d'une interface entre l'utilisateur et les scripts/programmes qui ont été réalisé pour faire le pré traitement.

Il doit proposer les fonctionnalités suivantes :

- traiter un fichier XHTML
- traiter un répertoire contenant des fichiers XHTML
- traiter une image
- traiter un répertoire contenant des images
- informer l'utilisateur sur le bon déroulement des opérations

2.3.3 IHM

Pré-Traitement

Pour les textes	Veuller indiquer le repertoire dans lequel se trouve les fichiers : <input type="text" value="doc"/> <input type="button" value="Traiter"/> Veuller indiquer le fichier que vous vouler parser : <input type="text" value="test1.html"/> <input type="button" value="Parser"/>
Pour les images	Veuller indiquer le repertoire dans lequel se trouve les images : <input type="text" value="doc"/> <input type="button" value="Traiter"/> Veuller indiquer le fichier que vous vouler traiter : <input type="text" value="Test_trait_01.tif"/> <input type="button" value="Traiter"/>

3 Archivage

3.1 Principe

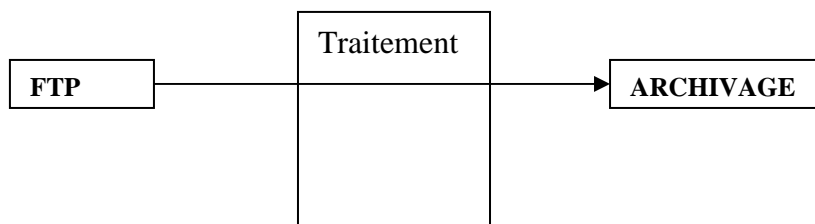
L'archivage consiste à prendre des données à un certain endroit, de faire des modifications et de les classer à un autre endroit. Dans le cas du système porphyre, les données arrivent sur le serveur de document de plusieurs façon différentes :

- par FTP
- par copie à parti du réseau local

Les traitements pour l'archivage sont assez basiques puisqu'il s'agit de numéroter les répertoires. Le point critique de ce module est le fonctionnement en mode scrutation à l'aide d'un démon qui tourne en permanence. En effet, il ne faut pas paralyser le serveur avec le démon. On notera qu'on commencera à faire fonctionner le programme d'archivage en mode « à la demande » et ensuite, on créera le démon.

3.2 Organisation des répertoires

L'archivage doit reposer sur une organisation des répertoires rigoureuses. Tout d'abord pour simplifier le problème on suppose que les données qui doivent être archivées sont dans un seul répertoire FTP



FTP et ARCHIVAGE peuvent être le même répertoire. Il y a plusieurs options possible que l'on va détailler :

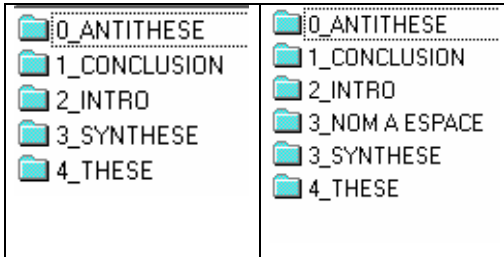
- FTP et ARCHIVAGE sont le même répertoire. Il y a de nombreux problèmes qui se posent :
- ARCHIVAGE est un sous-répertoire de FTP
- FTP est un sous-répertoire de ARCHIVAGE
- FTP et ARCHIVAGE sont des répertoires différents

De plus, il se pose une question importante : est ce que l'on veut garder les fichiers qui on été archivé ?

3.3 Traitement : numérotation

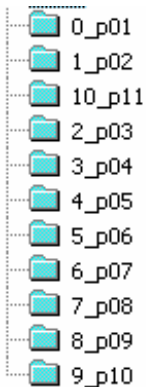
L'algorithme est le même que celui utilisé pour le pré-traitement des fichiers XHTML. Il faut ajouter X_ au nom des répertoires (Avec X le numéro du répertoire). Pour faire ce traitement, on réalisera un petit programme en C ou en script batch . Le C a l'avantage d'être portable , c'est pourquoi on choisira de préférence ce dernier.

On notera qu'il faut recommencer la numérotation si un nouveau répertoire est ajouté, comme le montre les captures ci-après :

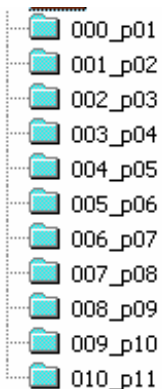


Remarque importante à propos de la numérotation :

Après avoir réalisé ce module et l'avoir testé sur un grand nombre de fichier, j'ai été confronté à un problème assez important et qui peut être illustré par la capture d'écran suivante :



En numérotant de cette façon, dans le système de fichiers de Windows, 10_p11 est avant 2_p03 par exemple. Hors, une des contraintes que l'on s'est fixée est d'obtenir une forme d'archivage modifiable de manière simple. Ici ce n'est pas le cas, car l'utilisateur est obligé de faire un effort supplémentaire pour localiser le répertoire qui l'intéresse.



La numérotation finalement choisie est la suivante. On code le numéro du répertoire sur 3 chiffres. En effet, cela nous semble suffisant car on ne devrait pas avoir de documents dépassant les 1000 pages. On décide également de commencer de numéroter à partir de zéro. (pour suivre le modèle américain)

3.4 Démon

Le démon doit pouvoir être lancé, arrêté et redémarré assez facilement. Une application console ou avec une IHM doit pouvoir contrôler le démon. Le démon sera lancé au démarrage de l'application et arrêté lorsqu'on quittera l'application.

La principale difficulté de la réalisation du démon est de savoir tout les combien il doit vérifier l'arrivée de nouveaux fichiers. En effet, à ce niveau là se pose de nombreux problèmes :

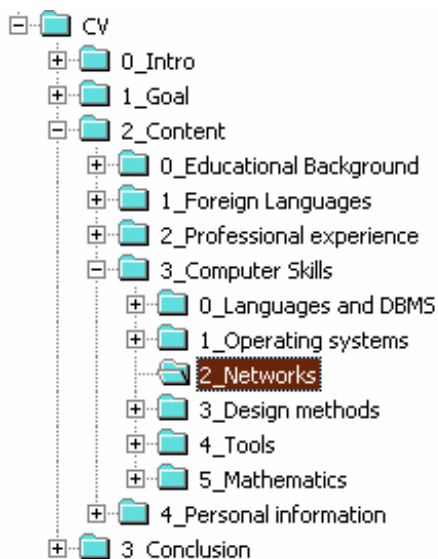
Comment faire puisque pour numéroter les répertoires il faut attendre que l'utilisateur est fini de transférer ces fichiers ?

4 Emission

4.1 Principe

Ce module est en fait en client Web dont le but est d'aller chercher un fragment de document identifié par un identifiant : cet identifiant représente le chemin qu'il faut parcourir dans l'arborescence pour obtenir le fragment de document.

Ex :



Le fragment de document qui contient les compétences en réseau est désigné par le code suivant : 2.3.2

Remarques :

Il y a un seul fragment de document par répertoire sinon il est impossible de retrouver le fragment.

Avant de trouver un fragment de document, il faut se placer dans le bon document parent. Par exemple dans cet exemple ci-dessus, il faut sélectionner le document CV.

Il n'y a pas de contraintes concernant la profondeur de la hiérarchie.

Remarque : pour simplifier, on emploiera souvent le terme document pour désigner un fragment de document.

4.2 Fonctionnalités

4.2.1 Obtenir une description (type, chemin logique et icône)

La description est composée du type du document ainsi que de son chemin logique. On peut également considérer que l'obtention d'une icône fait partir de la description. Les documents peuvent être du type suivant : Image, texte ou épure. Comme ce système d'archivage s'appuie sur le système de fichiers et non pas sur une base de données, il est nécessaire de se servir de l'extension du fichier pour déterminer le type du document. Il est à noter que ceci est un handicap. Car si on change de format ou que de nouveaux formats de fichiers apparaissent, il faut changer les scripts en conséquence.

Il faut également faire la distinction entre chemin logique et chemin physique :

Exemple avec le cas précédemment cité.

- le chemin physique du document est CV\2_Content\3_Computer Skills\2_Networks
- le chemin logique du document est CV\Content\Computer Skills\Networks

L'obtention de l'icône a 2 objectifs :

- Déterminer rapidement le type du document : en effet, on doit afficher une icône différente selon la nature du document.
- Pour les images, l'icône sera en fait la vignette qui a été créée pendant le pré-traitement ou que l'on aura créée à la volée si elle n'était pas disponible pour une raison quelconque.

4.2.2 Obtenir le document à l'écran

Lors de sa recherche de document, l'utilisateur se voit présenter une description du document. Si ce document correspond bien à ses attentes il peut décider de le visualiser. Comme la visualisation du document varie selon son type, il convient de séparer

Pour les images :

On rappelle que l'utilisateur aura le choix entre une vue optimisée photo et une vue optimisée dessin. De plus, pour le confort de l'utilisateur, la taille de l'image affichée sur son écran sera adaptée à sa résolution. Les résolutions supportées sont : 640*480, 800*600, 1024*768, 1152*864, 1280*1024, 1400*1050, 1600*1200. Il reste à faire des tests de performance pour déterminer si les images adaptées à ces résolutions seront créées pendant le pré-traitement ou à la volée lors des requêtes client. De la même façon, on devra décider si les images un fois qu'elles ont été créées doivent être conservées pour resservir. (compromis occupation disque/performance).

Pour les textes :

Cela consiste simplement à afficher le contenu d'un fichier sur l'écran de l'utilisateur.

Lorsque l'utilisateur fait une requête sur un document, il y a 2 solutions :

- On génère la vue A et la vue B en même temps que la vignette. De cette manière là, si l'utilisateur est intéressé par le document il n'a pas à attendre de nouveau pour obtenir le document à l'écran. Par contre, s'il n'est pas intéressé, l'utilisateur aura patienté pour rien.
- On peut également attendre que l'utilisateur fasse une requête sur la vue pour la générer. L'avantage est de pouvoir parcourir plusieurs documents assez vite en ayant seulement un aperçu grâce à la vignette. Par contre, si l'utilisateur veut avoir accès aux vues, il devra attendre à chaque fois.

La première solution est à privilégier si le temps de traitement est faible.

4.2.3 Manipulation sur le document.

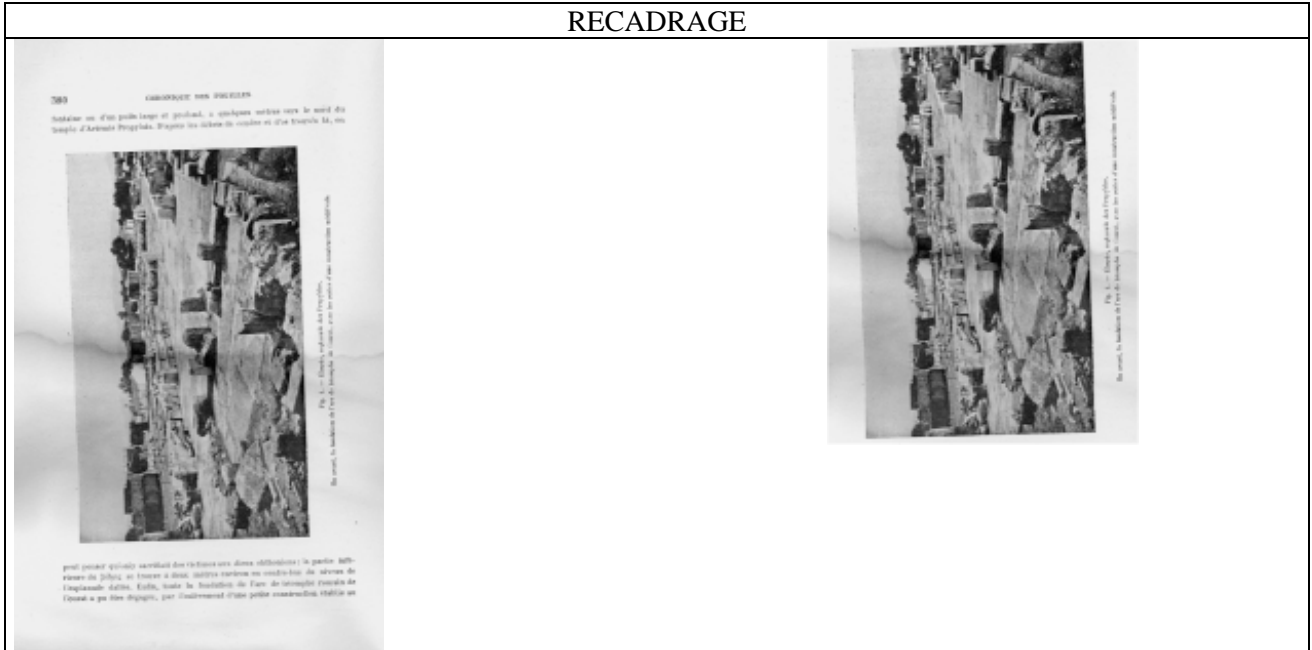
Une fois que l'utilisateur a obtenu son document, il peut lui faire subir des modifications et ainsi créer un nouveau document en quelques sortes.

Pour les images :

L'utilisateur peut extraire une partie de l'image comme dans l'exemple ci après. Il faut noter que ce n'est pas une option de zoom mais une option de recadrage. On notera que le but de ce client Web n'est pas de faire de la manipulation d'image (dans ce cas là, un applet Java serait plus adapté).

L'utilisateur peut également surligner certains fragments. Cela revient à dessiner un cadre coloré sur l'image (avec de la transparence bien évidemment). On pourra laisser à l'utilisateur le choix de la couleur. Un exemple de ce l'on veut pouvoir faire est donné par la suite.

RECADRAGE



SURLIGNAGE DE FRAGMENTS



Pour les textes :

Comme pour les images l'utilisateur peut extraire des fragments de texte.

RECADRAGE	
<p>Un avion de diplomates s'écrase dans la jungle, il y a trois rescapés : un Français, un Américain et un Belge. Ils sont immédiatement capturés par la tribu locale (comme toujours dans ces cas là). Le chef s'approche et dit "Voilà ! Comme aujourd'hui nous avons récupéré assez de viande pour passer l'hiver, je vous propose un marche. Si vous voulez rester en vie il faut passer deux épreuves. La première, nous ramener chacun 100 fruits de la forêt vierge. Pour la deuxième, on verra plus tard. "</p> <p>L'Américain refuse : "Je m'oppose à toute forme de chantage en vertu des droits de l'homme."</p> <p>Le chef du village attrape l'Américain et lui coupe la tête. Le Belge et le Français, pétrifiés, acceptent. Le chef donne à chacun un sac et les conduit à l'orée de la forêt vierge. "Allez ... revenez vite avec 100 fruits ! "</p> <p>Le Français revient le premier avec un sac plein de litchis. Le chef le félicite et réunit le conseil du village et lui dit : "Voilà ! Maintenant nous pouvons passer à la deuxième épreuve.</p>	<p>"Voilà ! Comme aujourd'hui nous avons récupéré assez de viande pour passer l'hiver, je vous propose un marche. Si vous voulez rester en vie il faut passer deux épreuves. La première, nous ramener chacun 100 fruits de la forêt vierge. Pour la deuxième, on verra plus tard. "</p>

SURLIGNAGE DE FRAGMENTS	
<p>Un avion de diplomates s'écrase dans la jungle, il y a trois rescapés : un Français, un Américain et un Belge. Ils sont immédiatement capturés par la tribu locale (comme toujours dans ces cas là). Le chef s'approche et dit "Voilà ! Comme aujourd'hui nous avons récupéré assez de viande pour passer l'hiver, je vous propose un marche. Si vous voulez rester en vie il faut passer deux épreuves. La première, nous ramener chacun 100 fruits de la forêt vierge. Pour la deuxième, on verra plus tard. "</p> <p>L'Américain refuse : "Je m'oppose à toute forme de chantage en vertu des droits de l'homme."</p> <p>Le chef du village attrape l'Américain et lui coupe la tête. Le Belge et le Français, pétrifiés, acceptent. Le chef donne à chacun un sac et les conduit à l'orée de la forêt vierge. "Allez ... revenez vite avec 100 fruits ! "</p> <p>Le Français revient le premier avec un sac plein de litchis. Le chef le félicite et réunit le conseil du village et lui dit : "Voilà ! Maintenant nous pouvons passer à la deuxième épreuve.</p>	<p>Un avion de diplomates s'écrase dans la jungle, il y a trois rescapés : un Français, un Américain et un Belge. Ils sont immédiatement capturés par la tribu locale (comme toujours dans ces cas là). Le chef s'approche et dit "Voilà ! Comme aujourd'hui nous avons récupéré assez de viande pour passer l'hiver, je vous propose un marche. Si vous voulez rester en vie il faut passer deux épreuves. La première, nous ramener chacun 100 fruits de la forêt vierge. Pour la deuxième, on verra plus tard. "</p> <p>L'Américain refuse : "Je m'oppose à toute forme de chantage en vertu des droits de l'homme."</p> <p>Le chef du village attrape l'Américain et lui coupe la tête. Le Belge et le Français, pétrifiés, acceptent. Le chef donne à chacun un sac et les conduit à l'orée de la forêt vierge. "Allez ... revenez vite avec 100 fruits ! "</p> <p>Le Français revient le premier avec un sac plein de litchis. Le chef le félicite et réunit le conseil du village et lui dit : "Voilà ! Maintenant nous pouvons passer à la deuxième épreuve.</p>

L'utilisateur peut surligner certains passages comme il le ferait avec un stylo. On peut également envisager de colorer seulement les lettres. Ceci a l'avantage d'être un peu plus visible.

Il est difficile de prévoir quelle sera la meilleure représentation d'un stylo sur une image ou à l'intérieur d'un texte. On fera quelques essais avant d'adopter une solution.

4.2.4 Inscriptions d'informations sur l'image : Droits d'auteur etc..

Les documents contenus dans une bibliothèque numérique sous soumis à certaines restrictions et droits d'auteurs. Il convient donc de trouver un moyen pour signaler ces informations à l'utilisateur. On s'intéressera ici seulement au cas des documents qui se présentent sous la forme d'une image (fac simulé de text, photographie).

On se posera 2 questions essentielles :

- Quelle apparence doit prendre ces informations : forme, emplacement
- A quel moment dans le processus, doit-on rajouter ces informations à l'image

Ces informations seront extraites d'une base de données. Les retours chariots seront gérés.

Ex :

Le contenu du copyright est :

"auteur : Olivier CHADENAT \n\r Tout droit réservé \n\r Copyright EFA"



auteur : Olivier CHADENAT
Tout droit réservé
Copyright EFA

Apparence du copyright :

La solution la plus simple serait d'inscrire ces informations en bas à droite par exemple. Toutefois, il paraît impossible de superposer le copyright sur l'image car cela risquerait de masquer des informations importantes contenues dans l'image. On s'oriente donc vers la solution suivante qui consisterait à « rallonger » l'image pour pouvoir y insérer le copyright. Voici la marche à suivre :

- Récupérer le texte du copyright et créer une image à partir de ce text
- Récupérer la hauteur de l'image contenant le copyright et la hauteur de l'image d'origine
- Créer une image contant l'image d'origine + le copyright

Ces opérations qui semblent simples sont gourmandes en temps machine, c'est pourquoi il convient de bien choisir le moment où il faut insérer le copyright

Processus :

Il y a 3 moments où l'on peut insérer le copyright ;

- Lors du pré-traitement (solution 1)
- Lors de la création de l'image adaptée à la résolution de l'utilisateur (solution 2)
- Lors de l'affichage final (solution 3)

Numéro de la solution	Avantages	Inconvénients
solution 1	Temps de traitement	Apparence du copyright Mise à jour difficile
solution 2	Apparence du copyright	Temps de traitement
solution 3	Apparence du copyright Mise à jour facile	Temps de traitement

L'insertion du copyright étant assez longue, la meilleure solution serait d'insérer le copyright lors du pré-traitement. De cette façon là, lors de la génération des images pour les différentes résolutions, il serait inutile d'insérer à chaque fois le copyright. Par contre, vue que l'image issue du pré-traitement sera modifiée (passage en niveau de gris, redimensionnement), il convient de s'assurer que le copyright sera toujours lisible.

Cette solution nous parait la meilleure compte tenu des contraintes qui nous sont apparues lors de cette étude:

- le copyright ne doit pas cacher des informations contenues dans l'image
- le copyright doit être lisible
- l'insertion du copyright doit le moins pénaliser le processus en temps machine

4.3 IHM

Dans le cadre de ces modules, il n'y aucune contrainte au niveau de l'IHM. On s'efforcera juste de réaliser des scripts performants.

Conclusion

Ce dossier n'aborde pas certains problèmes comme l'intégration de ces modules au projet Porphyre. En effet, ces modules ont été conçus pour être intégré dans porphyre. Toutefois, la structure de ce dernier et la documentation (ou plutôt l'absence de documentation sur certain module comme le serveur FTP) ne permettent pas d'assurer que les modules que j'ai conçu soient intégrables.

Comme je l'ai répété plusieurs fois dans ce dossier, il sera très important de faire des tests de performance lors de la réalisation des différents modules.