Année 2022—2023

Comic Book Reader/Writer

Une liseuse pour les passionnés de BDs

B. Monsuez

Comic Book Reader/Writer

Une liseuse pour les passionnés de BDs

But du projet

Ce projet consiste à réaliser un lecteur permettant de prendre en charge les différents formats utilisés pour les bandes dessinées, à l'instar de certains logiciels présents.

Cet outil devra permettre d'offrir une interface agréable et performante permettant de parcourir les différentes pages d'un livre au format .CBR[1] ou au format .CBZ[2].

Il devra de plus offrir à la fois un rendu graphique de qualité mais aussi un confort de lecture en essayant tant que possible de masquer le temps pris par les différentes tâches, en anticipant la lecture et le redimensionnement des pages, en permettant de débuter la lecture du document sans pour autant attendre que l'ensemble des pages soit chargé en mémoire, ainsi de suite.

Il devra en plus pouvoir assembler différentes images et les placer dans un fichier de type soit .CBZ, soit .CBZ, *ie*. créer des ouvrages au format .CBZ ou .CBR.

Enfin, il serait bien d'offrir la possibilité d'ajouter une table des matières (en proposant une extension du format de fichier, typiquement par l'ajout d'un fichier toc (table of content) au format XML par exemple dans la liste des fichiers contenus dans l'archive .CBZ ou .CBR.

Tâches demandées

Vous vous trouvez dans la situation où vous devez réaliser une analyse quant à la faisabilité du logiciel, proposer une architecture de ce logiciel et ensuite implanter un démonstrateur reposant sur cette analyse.

[1] https://fr.wikipedia.org/wiki/CBR_(format_de_fichier)

[2] https://fr.wikipedia.org/wiki/Comic_book_archive

1. Analysez le problème, sachant qu'il est souhaitable d'avoir une solution évolutive, permettant d'ajouter au fur et à mesure des fonctionnalités au logiciel, notamment d'étendre progressivement les formats supportés ainsi que les algorithmes de traitement d'image.

   Pour illustrer le propos, au niveau des formats d'images supportés, il est possible au départ de se contenter des formats d'images de type jpeg (.jpg) et de type bitmap (.bmp). Ensuite, il sera nécessaire d'ajouter pour le moins le format PNG (.png). Cependant, si vous avez bien conçu votre architecture, il sera possible d'encapsuler dans une classe « abstraite » l'ensemble des fonctions requises par votre programme relativement à la manipulation des images et ensuite de pouvoir traiter ces formats en créant pour chacun des formats une classe spécialisée.

   Il en va de même pour les formats d'archives. Au départ, vous pouvez vous contentez de gérer uniquement les formats .cbr (.rar) et .cbz (.zip) puis par la suite ajouter les formats .cb7 (.7z) ou .cbt (.tar)…

   De plus, comme il s'agit d'un outil devant apporter un bon confort d'utilisation, il est notamment demandé de faire :
   a. Une analyse d'usage du logiciel,
      i. Contexte d'utilisation,
      ii. Fonctions souhaitées par les utilisateurs
      iii. La définition des interactions entre les utilisateurs et le logiciel. Cela vous permettra de bien mettre au point les prérequis de votre interface graphique.

   b. Une analyse fonctionnelle du logiciel
      i. Les différents blocs fonctionnels du programme (le rendu des images, la navigation, le décodage et l'encodage des fichiers, …)
         C'est ici que vous allez définir les blocs qui vont :
         1. Accéder et manipuler les livres,
         2. Accéder et manipuler les images de livres, éventuellement la table des matières,
         3. Générer les images à afficher après avoir effectué des opérations de filtrage et conserver éventuellement ces images dans un cache pour ne pas avoir à les recalculer systématique.
         4. Les blocs de votre interface graphique. Par exemple, les blocs gérant l'affichage, les blocs gérant par exemple les chargements, les historiques des chargements de livres.
         5. Les blocs gérant éventuellement la conception d'un livre par assemblage ou en extrayant des pages d'un ouvrage pour le mettre dans un autre ouvrage.

      ii. Les interactions avec l'environnement au niveau fonctionnel
         1. Un livre est un ensemble de pages. Comment la gestion des livres interagit avec la gestion des pages.
         2. Comment interagit le bloc d'affichage d'image avec les images stockées dans un livre ? mais aussi avec le bloc effectuant le filtrage des images et enfin le bloc servant à stocker temporairement les images dans un cache afin d'éviter à avoir à les recalculer systématiquement à chaque visualisation de l'image.
         3. Les blocs gérant le chargement et le stockage des livres, l'interaction avec les blocs dédiés à la décompression des formats .cbr/.cbz ou à sa compression.

   c. Une architecture gros grains du logiciel.
      i. Les différentes classes principales

Pour donner quelques exemples, typiquement, une classe « BOOK », une classe « IMAGE » ou « PAGE ». Mais aussi des classes fournissant les algorithmes de filtrage, ceux accédant aux formats des fichiers compressés.

ii. Les concepts de modularité et d'extensibilité,
Comme mentionné en premier, il est intéressant de pouvoir étendre le logiciel avec de nouveaux formats d'image, de nouveaux formats de livre et aussi de nouveaux algorithmes de filtrage d'image sans avoir besoin de recoder mais uniquement en dérivant des objets.

iii. Les interactions entre ces éléments.
Les différentes interactions entre les classes principales, vous permettant d'avoir un modèle de type requête.
DISPLAY => demande à FILTRE d'appliquer Algorithme à l'IMAGE pour être affiché et stocké dans le cache.

avant même de commencer réellement à étudier l'implantation de ce logiciel.

2. Réalisez un démonstrateur pour valider l'approche et les choix techniques et s'assurer que ces derniers sont pertinents. Dans le cas d'une approche évolutive, il est conseillé d'implanter un noyau applicatif :
3.
   a. Prise en compte des images au format .bmp et .jpeg
   b. Prise en compte du format d'archive .rar et/ou .zip
   c. Redimensionnement de l'image sans filtrage.

   Puis de faire évoluer ce noyau avec de nouveaux formats d'images supportés mais aussi de nouveaux formats d'archive et de nouveaux algorithmes de redimensionnement avec filtrage.

   Il est notamment demandé de faire :
   d. Une description du modèle d'architecture
   e. De bien documenter les choix des classes,
   f. De réaliser une implantation documentée du logiciel.

L'outil devra fournir au minimum les services suivants :

Ouverture d'un livre de manière asynchrone en permettant de consulter les premières pages avant que le chargement du livre soit terminé.

Redimensionnement des images à l'échelle de l'écran.

Possibilité d'afficher plusieurs pages à l'écran (au moins deux pages). Penser notamment à soigner le redimensionnement des images si les deux images ne sont pas exactement à la même taille pour donner l'impression de deux images au même format.

Pouvoir sélectionner un filtre de redimensionnement adapté au document (ie. plus adapté au texte ou plus adapté aux graphiques).

Pouvoir feuilleter l'ouvrage, aller à la page suivante, précédente, première page et dernière page de l'ouvrage, et ce le plus naturellement possible.

Pouvoir extraire des pages d'un ouvrage.

Pouvoir créer un ouvrage en assemblant des pages.

Pouvoir ajouter une table de matière aux ouvrages existant.

Quelques éléments de Réflexion sur les différentes Etapes

Étape 1 : Analyse du problème

Phase A :

Analysez d'une part les formats de base des différents fichiers au format .cbr et/ou .cbz. Déterminer les spécificités des différents formats d'images pouvant être utilisés.

Rechercher les différents algorithmes de redimensionnement.[3]

Phase B :

Déterminer les interactions de votre logiciel avec l'utilisateur. Dans le cadre de ce projet, vous concevez un applicatif 'user-centric', c'est le confort d'utilisation qui doit être au centre de vos préoccupations.

Il est possible de définir un cahier des charges plus larges que ce que vous ferez dans le cadre de votre projet mais qui vous permettra par contre par la suite de bien structurer votre architecture de programme (choix des classes, choix des interactions, fonctions asynchrones) pour pouvoir faire évoluer votre applicatif de manière à ce qu'il respecte le cahier des charges intégral.

Pour la question B, une des personnes du projet peut travailler à l'aide de diagrammes UML (c'est simplement une option parmi d'autres), pendant que les autres réfléchissent à ce qui peut bouger entre les différentes phases et comment ce sera exploité.

L'analyse architecturale du projet étant terminée, voici une introduction rapide aux quelques difficultés algorithmiques de ce projet.

La représentation de l'ouvrage

Même si les formats .cbz et/ou .cbz sont relativement simples, de fait, la multiplicité des formats d'archive et la multiplicité des formats d'images utilisés pour stocker les images composant les pages d'un ouvrage nécessité de bien réfléchir au modèle objet à utiliser.

Il est d'évidence qu'il est important de définir un modèle objet « abstrait » d'un fichier archive qui fournira l'ensemble des fonctionnalités communes nécessaires pour manipuler un fichier archive sans connaître le type du fichier archive. Notamment la liste des tous les fichiers contenus dans l'archive, leur ordre, quelle est le premier fichier images contenant la première page, la seconde page, ainsi de suite.

[3] De fait, le plus souvent la taille de l'image doit-être réduite. Dans ce cas, il faut corriger le sous-échantillonnage en appliquant un filtre passe-bas, par exemple un filtre moyenneur.

L'utilisation de bibliothèques de manipulation de ces fichiers (.zip et/ou .rar) est conseillée pour effectuer ces opérations. Cependant il sera nécessaire de fournir un modèle objet commun, chacune des bibliothèques n'offrant pas le même modèle d'abstraction, éventuellement, n'offrant que des fonctions « C ».[4]

De même il sera nécessaire de définir un modèle objet « abstraits » d'une image permettant d'accéder à une image, de décoder celle-ci pour la redimensionner et l'afficher. Il faudra aussi pouvoir l'encode éventuellement vers un format différent pour la stocker.

L'utilisation d'une bibliothèque de manipulation d'images peut-être recommandé pour effectuer ces opérations. Certaines offrent des modèles objets.[5]

A côté de ces classes de manipulation de fichiers « image » et de fichiers « archive », il faudra ensuite réfléchir à la notion de livre (un objet 'Book' par exemple) et aussi la notion de page d'un ouvrage (un objet 'Page'). Peut-être vous sera-t-il possible d'aller plus loin et de penser à la notion de collection d'ouvrages correspondant à une sérier (un objet 'BookCollection') ?

L'occupation mémoire et la préparation des « pages » à afficher à l'écran

La plupart du temps les images sont présentes dans un format compressé sans pertes .gif, .bmp ou avec pertes .jpeg, .png, … La lecture des images imposent de les décompressés et elles peuvent devenir jusqu'à 10 fois plus importantes en termes d'occupation mémoire.

De même, l'image ainsi décompressée n'est pas au bon format. Il faudra donc la redimensionner pour l'affichage. Ce redimensionnement est souvent une réduction de la taille de l'image ce qui conduit à sous-échantillonner, ce qui n'est pas toujours très esthétique avec l'apparition d'un phénomène d'aliasing et peut aussi dégrader l'affichage des caractères. De fait, il existe plusieurs types de filtre passe-bas qui sont appliqués pour moyenner les nouveaux points de l'image. Cependant en fonction de la nature de l'image, du type de caractères, il faut sélectionner un type de filtre ou un autre type de filtre.

Il faut réfléchir au moment de l'analyse à l'architecture extensive permettant de supporter l'adjonction de nouveaux filtres.

La gestion d'une table de matière.

Il faut déterminer comment ajouter une table des matières tout en maintenant la compatibilité avec le format .cbr et .cbz. Une option est d'ajouter dans un répertoire un fichier contenant des informations

[4] Pour le format 7zip en compression et plein de formats (.zip, .tar, . rar, …) en lecture : : http://7-zip.org/sdk.html

[5] Une telle bibliothèque est par exemple ImageMagik http://www.imagemagick.org/script/index.php

complémentaires comme la table des matières mais pourquoi pas d'autres informations concernant le document.

Il pourrait être intéressant de récupérer à partir du titre et/ou du numéro ISBN les informations à partir d'un site comme bedetheque.com.

Étape 2 : Réalisation du démonstrateur

Phase A :

Procédez à une première implantation de l'algorithme qui ouvre un ouvrage au format .cbz ou .cbz et qui lit l'ensemble des images. La première version sera séquentielle, ensuite, vous tâcherez de mettre au point la version asynchrone qui vous prévient chaque fois qu'une image est chargée et peut-être affichée.

Phase B :

Procédez à une implantation de l'interface graphique. Cette implantation sera réalisée en s'appuyant sur une bibliothèque d'interface graphique comme QT[6] ou wxWidget[7] par exemple.

Phase C :

Procéder à l'affichage des pages (une page par une page) dans votre environnement utilisateur.

Phase D :

Procéder aux améliorations des fonctionnalités en traitant dans l'ordre :

- L'implantation des algorithmes de redimensionnement de l'image (fonction de zoom, etc)
- L'implantation des algorithmes de chargement en avance de l'image. L'image suivante ou précédente est décodée et redimensionnée de manière asynchrone afin de pouvoir l'afficher immédiatement si l'utilisateur passe à la page suivante.
- L'affichage de deux pages à l'écran avec leur redimensionnement relatif pour donner l'impression qu'il s'agit des deux pages ayant exactement la même taille[8].

Étape 3 : Amélioration de l'outil

Vous avez le choix pour améliorer votre outil, mais n'oubliez pas de conserver le résultat de l'étape 2.

[6] https://www.qt.io/

[7] http://wxwidgets.org/

[8] Il arrive souvent que dans un ouvrage de type Comic Book Reader, qu'après la numérisation des images, un post-traitement a été appliqué qui conduit à ce que deux images correspondant à deux pages qui se suivent, n'ont plus exactement la même dimension.

Les améliorations sont pas exemple d'offrir des fonctions d'ajustement de la page à l'écran, toute la page dans l'écran, toute la largeur de la page à l'écran, ainsi de suite[9].

- L'affichage des pages en continue ;
- Le support des gestes tactiles sur l'écran.
- Le support des BDs japonaises (affichage en sens inverse).
- La prise en compte d'autres formats.
- La détermination des fichiers d'archive sans se fier au type[10].

Vous pouvez ensuite améliorer l'interface utilisateur, augmenter les performances en utilisant la carte graphique pour le redimensionnement d'image, etc..

Enfin, vous pouvez tout à fait utiliser une bibliothèque de gestion des documents .pdf pour ajouter le support en lecture des fichiers .pdf.

Bref, n'hésitez pas à laisser libre court à votre imagination, tout en restant dans le domaine du réalisable dans le temps que vous avez décidé de consacrer au projet.

[9] L'interface d'Abobe Acrobat Reader peut donner des idées de fonctionnalités agréables pour lire un livre page par page.

[10] https://en.wikipedia.org/wiki/List_of_file_signatures contient la liste des octets qui permettent de déterminer le type d'un fichier, que ce soit un fichier image ou un fichier archive. Ceci permet de ne pas se fier au type d'extension et de connaître le vrai type du fichier. Il arrive en fait que des fichiers .cbr soient des fichiers .cbz (format zip et non rar). De même, parfois des images n'ont pas d'extension dans les archives.