Rémi LANGDORPH TD J 13/04/2023

Elyess KHADRAOUI

**Compte Rendu Projet Scientifique Informatique**

# Projet

Dans le cadre du projet scientifique informatique, nous avons eu l’idée de créer « Pinte », une version parodique du logiciel « Paint ™» de Microsoft™.

Pinte est une application C# composée de 3 projets :

* Un projet de Test (Tests)
* Un projet de librairie (PSILib) : c’est ici qu’est développée toute la logique autours de l’image.
* Un projet WPF (PinteUI) : c’est ici que toute l’interface utilisateur est présente.

Les innovations sont surlignées.

# Tests statiques

Il est très complexe de tester efficacement les changements sur une librairie qui manipule des images : c’est pourquoi nous avons mis en place une méthode de tests statiques qui sont des suites d’opérations effectuées sur des images qui sont comparées avec des images de référence (stockées dans le dossier ref/).   
Nous pouvons ainsi identifier instantanément le moindre changement subtil, la moindre régression.

Afin de lancer les tests : il suffit de lancer la commande « $ dotnet test » dans le dossier du projet.

# Fonctionnalités de la librairie

## Lecture/écriture/Fractale (Huffman1D ou sans compression)

Nous avons implémenté la lecture et l’écriture des fichiers BMP, en rajoutant un mode de compression HUFFMAN1D : dans ce mode, l’extension de fichier lors de l’écriture est remplacée par .beermp même si ils gardent la structure des fichiers .bmp classiques à l’exception de la matrice de pixel qui est remplacée par l’arbre de Huffman ainsi que la matrice des identifiants des nœuds de l’arbre. Ce mode de compression n’est pertinent que sur les images avec peu de couleurs.

De plus, nous avons ajouté des mesures de sécurité pour éviter les fichiers malicieux aux dimensions fausses supposés faire de larges allocations de RAM en limitant le nombre de pixel à 109 lors de la lecture du fichier (classe ImageSettings).

Au lieu d’importer un fichier existant, l’utilisateur peut générer une fractale de la taille de son choix ou une toile unie de la couleur et de la taille de son choix.

## Opérations sur les images

Nous avons implémenté plusieurs opérations sur les images :

* La rotation (n’importe quel angle)
* Le redimensionnement (n’importe quel facteur)
* Retrait des bords : nous identifions les bords d’une couleur spécifique de tous les côtés de l’image et le retirons, l’épaisseur est automatiquement détectée.
* Ajout des bords : nous ajoutons un bord d’épaisseur et couleur définie par l’utilisateur de chaque côté de l’image.
* Rognage : nous ne gardons qu’une partie de l’image aux dimensions et coordonnées sélectionnées par l’utilisateur.
* Différence : nous prenons 2 images et pour chaque couleurs d’un pixel, nous prenons la différence absolue entre les valeurs des 2 couleurs. Par exemple Pixel(11, 22, 33) et Pixel(33, 22, 11) donnent le Pixel(22, 0, 22).

## Effets sur les images

Nous avons implémenté plusieurs effets sur les images, ceux-ci ne modifient que la valeur des pixels :

* Nuance de gris (GrayScale)
* Négatif (Negative)
* Détection de bords (Edge detection)
* Affinage (Sharpen)
* Flou gaussien (Gaussian Blur) : 3x3, 5x5
* Masque de netteté (Unsharp Mask)
* Filtre linéaire à caisson (Box Blur)
* Matrice personnalisée

## Outils d’espions / Stéganographie

Nous avons implémenté plusieurs outils d’espionnage :

* Cacher/Récupérer une image dans l’image : nombre de bits réglable
* Cacher/Récupérer du texte dans l’image
* Générer un fichier « BMP Bomb » avec de fausses dimensions qui a pour but de prendre un maximum de RAM. Ces fichiers peuvent être utilisés afin de créer des attaques par déni de service (DoS) lors de la lecture du fichier, car le programme cherche à allouer une matrice de Largeur\*Hauteur éléments, souvent sans vérifier si le nombre de pixel est possible.

# Interface Utilisateur (WPF)

Nous avons décidé d’utiliser le framework Microsoft™ WPF afin de créer l’interface utilisateur.

Il suffit de lancer la commande « $ dotnet run –project PinteUI » dans le dossier du projet pour lancer l’interface.  
Celle-ci se compose de plusieurs fenêtres :

* 1. Ecran de lancement (Window1)

L’écran de lancement est une simple fenêtre permettant de choisir entre importer une image, générer une toile vide ou une fractale.

* 1. Console de débogage (DebugConsole)

Les applications WPF n’ayant pas de sortie console, nous avons décidé de créer une console contenant des informations de debug.

Pour lancer avec la console, il suffit d’ajouter l’argument « --debug » après la commande de lancement de Pinte.

* 1. Fenêtre d’édition (MainWindow)

La fenêtre d’édition contient l’image ainsi que les outils nécessaires.

* 1. Fenêtre de popup (OperationSettingsPopup)

La fenêtre de popup permet de demander des paramètres à l’utilisateur de manière simple pour chaque opération.

Cela nous permet d’avoir une granularité de réglages dans toutes les fonctions disponibles.