Document de conception

Bilel

Leila

Valentin

Théo

Mini-Logiciel de traitement d’image

**alias : «stardust»**

Ce logiciel doit permettre de réaliser des opérations basiques de traitement d’image sur trois formats d’image qui sont .pbm, .pgm et .ppm. L’environnement de développement est Python 3.x.

# Présentation des fichiers .pbm, .pgm et .ppm

Ces extensions font référence à trois types de fichiers texte permettant de coder une image.

## Format de fichier .pbm

Ce format permet de coder une image en noir et blanc en format texte. L’extension signifie « Portable Bit Map ». Bit signifie que chaque pixel est codé sur un seul bit.

|  |  |
| --- | --- |
| **Exemple :** | **Description :** |
| P1  8  3  0 0 0 0 1 1 1 1  0 1 1 1 1 0 0 0  0 0 0 0 1 1 1 1 | La première ligne code le format de l’image P1 correspond à .pbm  La deuxième ligne donne le nombre de colonne  La troisième ligne donne le nombre de ligne  Les lignes suivantes contiennent des valeurs codant chaque pixel de l’image : 1 (noir) ou 0 (blanc) |

## Format de fichier .pgm

Ce format permet de coder une image en niveau de gris. L’extension signifie (Portable Grey Map).

|  |  |
| --- | --- |
| **Exemple :** | **Description :** |
| P2  5  2  255  0  50  45  40   0  0  55    0    0   0 | La première ligne contient le format de l’image P2 pour .pgm  La deuxième ligne donne le nombre de colonne  La troisième ligne donne le nombre de ligne  La quatrième ligne contient la valeur maximale pour l’échelle de teinte  Les lignes suivantes contiennent des valeurs codant chaque pixel de l’image : 0 (noir) et 255 (blanc) |

## Format de fichier .ppm

Ce format permet de coder une image en couleur. L’extension signifie (Portable Pix Map).

|  |  |
| --- | --- |
| **Exemple :** | **Description :** |
| P3  1  3  255  103 255 13  9 63 112  235 201 89 | La première ligne contient le format de l’image P3 pour .ppm  La deuxième ligne donne le nombre de colonne  La troisième ligne donne le nombre de ligne  La quatrième ligne contient la valeur maximale pour l’échelle de teinte  Les lignes suivantes contiennent des valeurs codant 3 par 3 pour un pixel. Ces trois valeurs comprises entre 0 et 255 indiquent respectivement la quantité de rouge, de vert et de bleu du pixel. |

# Structure de donnée

Dans tout le mini-logiciel, les images seront sous la forme d’une structure de donnée basée sur un dictionnaire. Dans toute la suite, cette structure sera appelée le « dictionnaire-image » et abrégé en « dict-img ». Voici ses caractéristiques :

|  |  |
| --- | --- |
| **Exemple :** | **Description :** |
| {"meta" : { "titre" : "titre",  "extension" : ".ppm",  "col" : 2,  "lig" : 3,  "mod" : "" },  "pix" : [[(255, 255, 255), (253, 0, 34)],  [(…, …, …), (…, …, …)],  [(…, …, …), (…, …, …)]]} | str : Titre de l’image  str : Extension du fichier (".pbm" ou ".pgm" ou ".ppm")  int : Nombre de colonne  int : Nombre de ligne  str : Explication de la modification apportée à l’image  list(list) : Tableau des pixels. Chaque pixel est codé par un tuple de trois valeurs comprises entre 0 et 255 respectivement codant pour l’intensité du rouge, du vert et du bleu. |

**Remarques :** Pour un dictionnaire-image correspondant à un format .pbm, les tuples-pixels ne peuvent prendre que deux valeurs (255, 255, 255) pour un pixel blanc et (0, 0, 0) pour un pixel noir. Pour un format PGM, les tuples-pixels ont forcément leurs trois valeurs égales pour coder un niveau de gris.

# Fichiers

## 1. Module : main.py

Gère l’interface graphique pour l’utilisateur.

## 2. Module : support.py

### ouvrir(fichier)

Description : Ouvre le fichier.pbm, .pgm et .ppm et renvoie l'image au format dictionnaire-image décrit ci-dessus.

Param :

- fichier : str - Chemin d’accès du fichier. Les formats lisibles sont .pbm, .pgm et .ppm.

Return : dict-img

### afficher(img)

Affiche l'image dans une fenêtre Tkinter.

:param img : dict-img

### comparer(img1, img2)

Affiche deux images côte à côte dans une fenêtre Tkinter.

:param img1 : dict-img

Img2 : dict-img

### sauver(img)

Enregistre l’image sous la forme d’un fichier de même nom suffixé par « \_modifié » dans le même dossier.

:param img : dict-img

## 3. Module : traitement.py

Ce module contient les fonctions de traitement de l’image :

### symHori(img)

Réalise une symétrie horizontale de l’image.

:param img : dict-img

:retour : dict-img

### symVert(img)

Réalise une symétrie verticale de l’image.

:param img : dict-img

:retour : dict-img

### rotation180(img)

Réalise une rotation de 180° de l’image.

:param img : dict-img

:retour : dict-img

### rotation90(img)

Réalise une rotation de 90° de l’image.

:param img : dict-img

:retour : dict-img

### gris(img)

Converti une image au format PPM (couleur), en une image au format PBM (noir et blanc).

:param img : dict-img

:retour : dict-img

### noir(img)

Converti une image au format .ppm (couleur) ou .pgm (niveau de gris) au format .pbm (noir et blanc).

:param img : dict-img

:retour : dict-img

Ainsi qu’éventuellement d’autres idées de fonctions :

### luminosite(img, taux)

Modifie la luminosité de l’image.

param img : dict-img

taux: int (Compris entre 0 et 100 inclus)

:retour : dict-img

### saturation(img, taux)

Modifie la saturation des couleurs de l’image.

:param img: dict-img

taux: int (Compris entre 0 et 100 inclus)

:retour: dict-img

**rotation(img , degree)**

Descriptions : Permet de faire une rotation à n’importe degrés , attention cependant le taille de l’image est conservée .

Paramètres :

- img = dict-img - image\_data

- degree = int – degrés de rotation

return : dict-img

**4. Module ui.py :**

Description : Ce module contient toute les classes pour créer des composants d’UI comme par exemple : Boutton , sélecteur , entry ...

**5. Module window.py :**

Description : Ce module contient toute les classes pour créer des fenêtres

**6. Module image.py :**

Description : Ce module contient toute les classes pour stocker , et visionner une image .

**Classe Image :**

**@static\_method rgb\_to\_hsl(rgb)**

Description : Convertit la valeur rgb en valeur hsl ( Hue , Saturation , Luminosity)

Paramètres :

- rgb = list – valeur rgb à convertir

retour : hsl = list

**@static\_method hsl\_to\_rgb(hsl)**

Description : Convertit la valeur hsl en valeur rgb

Paramètres :

- hsl = list – valeur hsl à convertir

retour : rgb = list

**7. Module panel.py :**

Description : Ce module gère les panels

**8. Module scrollbar.py :**

Description : Ce module gère les scrollbars dans les panels

**9. Module filters.py :**

Description : Ce module contient toute les fonctions de filtres

**10. Module events.py :**

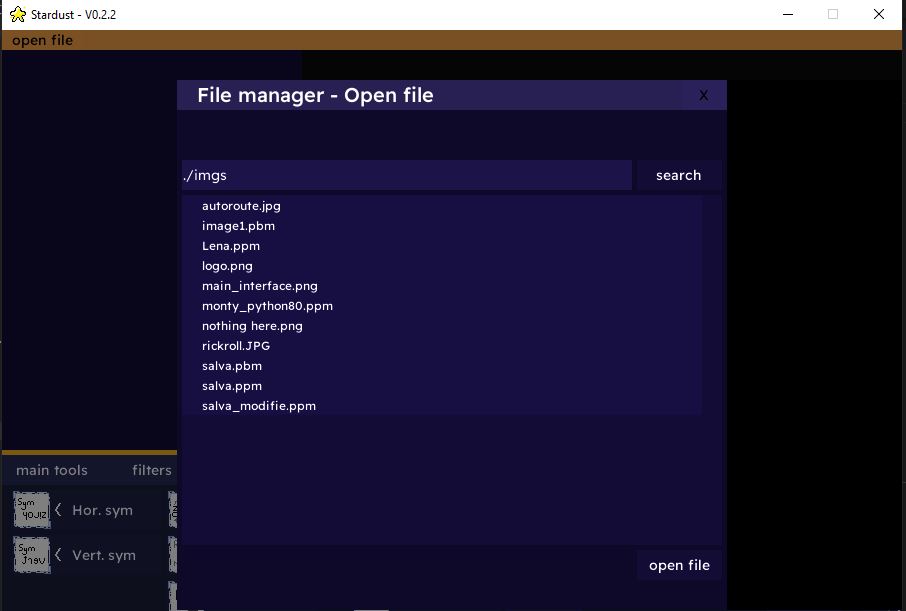
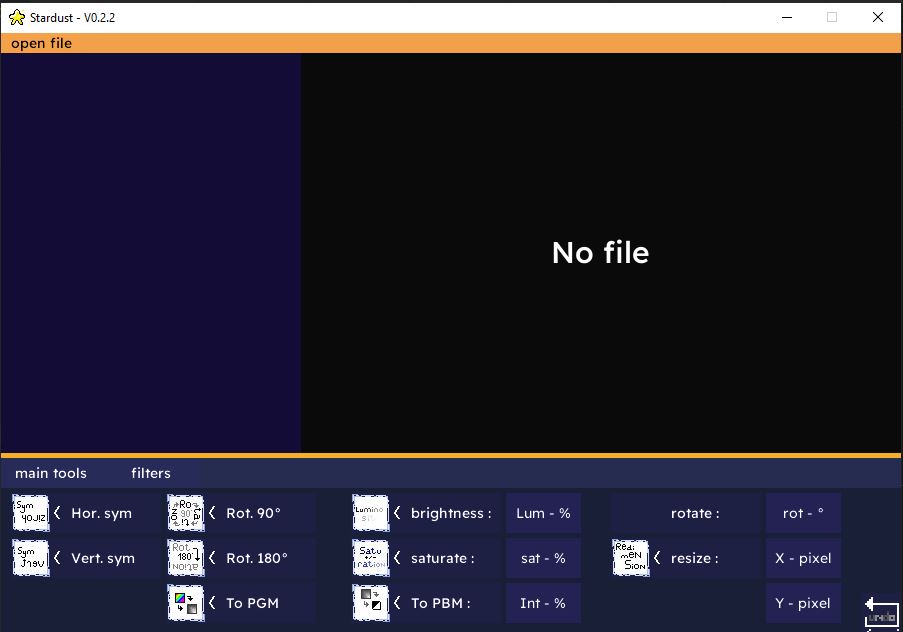
Description : Ce module contient toute les fonctions événements appelés par les boutons dans l’interface

**11. Module data.py :**

Description : Ce module stocke grâce un dictionnaire , toute les textures

**IV – Fonctionnement interface graphique**

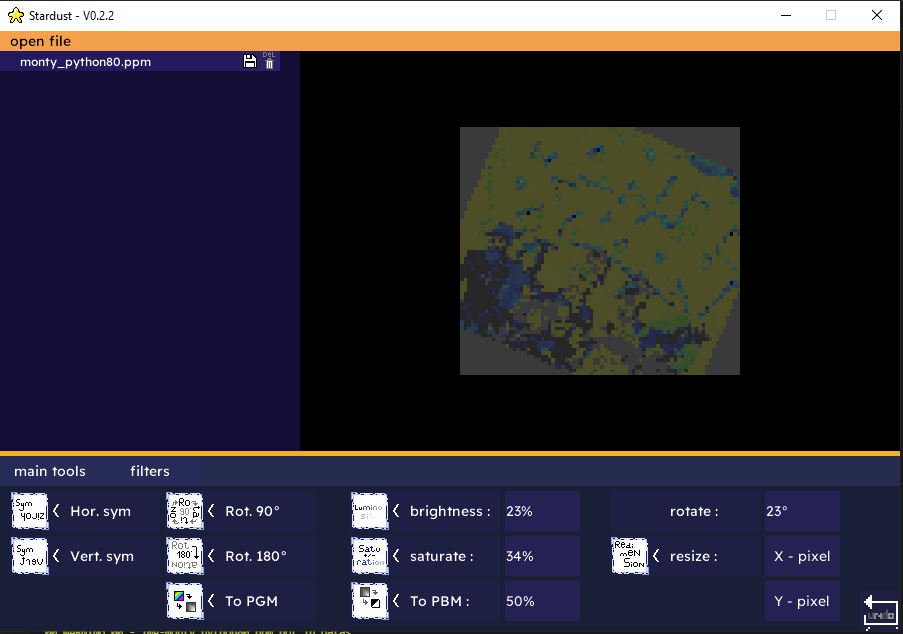
**I – Ouvrir un fichier**

****

Après être arrivé dans cette interface , pour ouvrir une image , il suffit d’entrer le chemin vers le dossier contenant l’image , après avoir lancer la recherche d’image en appuyant sur «search» , cela vous donne une liste de toute les images lisibles par le programme (png , jpg , ppm , pbm , pgm) .

Enfin vous aurez juste à sélectionner une image en cliquant pour la sélectionner et cela va vous l’ouvrir , si l’image ne s’ouvre pas , c’est que le fichier en question est certes d’un type de fichier qui est lisible par le programme , mais le contenu est encodé différemment .

**II – Appliquer des modifications**



Après avoir ouvert l’image , vous pouvez changer d’image avec la liste des images ouvertes à gauche , et appliquer des modifications grâce au panel en bas . L’image sera affiché dans la zone noir de l’interface .

Vous pouvez aussi remove le fichier de la liste , en survolant son sélecteur et en appuyant sur le bouton avec une poubelle , vous pouvez aussi la save avec l’autre bouton .