

## **INFO 424**

### **Projet en Informatique**

Équipe n°2 : **TeamHVC**

Membres : - **Ciminera Zack** : ciminera.zack@gmail.com  
- **Vouillamoz Frédéric** : frederic.vouillamoz@etu.univ-savoie.fr  
- **Hurstel Aurore** : hurstel.aurore@gmail.com

# TABLE DES MATIERES

1. Présentation générale du projet.....	3
2. Description des formats pgm et ppm.....	4
3. Tutoriel d'utilisation de matrices en Python.....	4
4. Tutoriel d'utilisation de matrices en Java.....	4
5. Tutoriel d'utilisation de nos deux logiciels à partir du terminal.....	4, 5, 6, 7
6. Récapitulatif des séances de TP.....	8
7. Description des fonctionnalités supplémentaires.....	8

## 1. Présentation générale du projet

Dans cette matière, nous avons pour but de réaliser un projet de développement logiciel d'envergure sur le thème de la synthèse et l'analyse d'images.

Nous avons donc créé deux programmes, le premier s'occupant de la synthèse d'images, le second de l'analyse d'images.

### Programme n°1 : Générateur (Synthèse d'images)

- Génère des images à partir d'arguments passés en ligne de commande.
- Encode l'image dans un format standard (pgm ou ppm).
- Produit l'image via la sortie standard.

Ex :



Nous avons choisi de programmer ce générateur en langage Python.

### Programme n°2 : Analyseur (Analyse d'images)

- Lit un fichier image via l'entrée standard.
- Décode le fichier pour produire l'image.
- Identifie des formes géométriques dans l'image.

Ex :



Nous avons choisi de programmer cet analyseur en Java.

### Fonctionnalités :

#### Programme n°1 : Générateur (Synthèse d'images)

Permet de dessiner les formes suivantes :

- Point
- Segment
- Rectangle (vide ou plein)
- Cercle (vide ou plein)
- Triangle (vide ou plein) (triangles à bases non horizontales gérés également)

425 couleurs sont disponibles.

#### Programme n°2 : Analyseur (Analyse d'images)

Permet de reconnaître les formes suivantes :

- Point
- Rectangle
- Cercle
- Triangle (triangles à bases non horizontales gérés également)

## 2. Description des formats pgm et ppm

Format PGM :

La première ligne d'un fichier PGM contient soit la ligne « p2 » si c'est un fichier codé en ASCII, ou la ligne « p5 » si c'est un fichier codé en binaire.

Le format PGM est un format d'images en niveau de gris. Chaque pixel est codé par une valeur comprise entre 0 (noir) et 256 (blanc).

Chaque pixel correspond à une case d'un tableau bidimensionnel (ou matrice), qui définit la taille de l'image.

Ex : Image de dimension 4\*4 (carré noir)

```
p2
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```

Format PPM :

Le format PPM est un format d'images en couleurs.

La première ligne d'un fichier PPM contient la ligne « P3 », ce qui signifie que les couleurs sont codées en ASCII, sur la deuxième est écrit le nombre de colonnes et de lignes, et sur la 3ème est écrit « 255 » ce qui signifie que les images ont 255 pour valeur maximum et qu'elles sont en RGB.

Ex : Image représentant un rectangle le 3 pixels : bleu, blanc et rouge

```
P3
3 1
255
0 0 255    255 255 255    255 0 0
```

## 3. Tutoriel d'utilisation de matrices en Python

Une matrice (ou tableau bidimensionnel) est définie par des colonnes et des lignes.

En Python, une matrice est un tableau de tableaux, ces derniers représentant les lignes, leur nombre d'éléments étant leur colonnes.

Chaque tableau doit donc avoir le même nombre d'éléments.

Ex : Déclarer et afficher une matrice en Python

```
matrice = [[1,2,3],[4,5,6],[7,8,9]]    déclare la matrice « matrice »
print (matrice)                        affichera la matrice 3*3 suivante :
                                         1 2 3
                                         4 5 6
                                         7 8 9
```

## 4. Tutoriel d'utilisation de matrices en Java

Une matrice (ou tableau bidimensionnel) est définie par des colonnes et des lignes.

En Java, une matrice est un tableau de tableaux, ces derniers représentant les lignes, leur nombre d'éléments étant leur colonnes.

Chaque tableau doit donc avoir le même nombre d'éléments.

Ex : Déclarer une matrice d'entiers en Java


```
int[][] matrice = new int[x][y]        x étant le nombre de lignes, y le nombre de colonnes
```

## 5. Tutoriel d'utilisation de nos deux logiciels à partir du terminal

### Tutoriel d'utilisation du Générateur (Synthèse d'images)

Ouvrir un terminal UNIX.

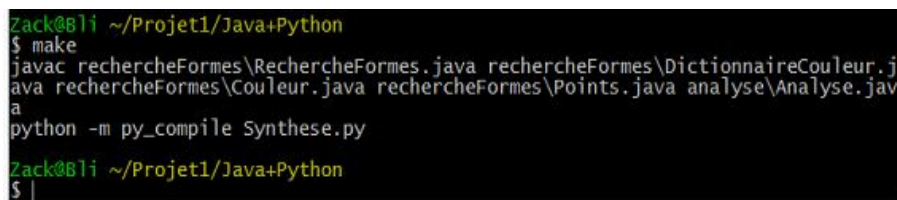
Aller dans le répertoire où se trouve le programme à partir de la commande **cd**.



```
~/Projet1
Zack@Bli ~
$ cd Projet1/
Zack@Bli ~/Projet1
$
```

Effectuer la commande **make** afin de compiler le programme.

(En Python, **make** ne compile pas vraiment. Il vérifie surtout les erreurs de syntaxe.)



```
~/Projet1/Java+Python
$ make
javac rechercheFormes\RechercheFormes.java rechercheFormes\DictionnaireCouleur.j
ava rechercheFormes\Couleur.java rechercheFormes\Points.java analyse\Analyse.jav
a
python -m py_compile Synthese.py
Zack@Bli ~/Projet1/Java+Python
$
```

Écrire ensuite la ligne de commande correspondant à l'action voulue.

Voici une copie d'écran de notre fichier README.txt où se trouve la liste des commandes et actions disponibles.

4 - Taper ./NomduFichier.py suivi des arguments correspondant à l'action désirée.

Les arguments sont les suivants :

- POINT : --point <nom de l'image à créer>  
          <coordonnéeX> <coordonnéeY> <Couleur\*\*>  
          <largeurImageàCréer> <hauteurImageàCréer>
- SEGMENT : --segment <nom de l'image à créer>  
          <coordonnéeX 1erpoint> <coordonnéeY 1erpoint> <coordonnéeX 2emepoint> <coordonnéeY 2emepoint> <Couleur\*\*>  
          <largeurImageàCréer> <hauteurImageàCréer>
- RECTANGLE : --rectangle <nom de l'image à créer>  
          <coordonnéeX Pointdedépart> <coordonnéeYPointdedépart> <hauteur> <largeur> <Booléen\*> <Couleur\*\*>  
          <largeurImageàCréer> <hauteurImageàCréer>
- TRIANGLE : --triangle <nom de l'image à créer>  
          <X 1erpoint> <Y 1erpoint> <X 2emepoint> <Y 2emepoint> <X 3emepoint> <Y 3emepoint> <Booléen\*> <Couleur\*\*>  
          <largeurImageàCréer> <hauteurImageàCréer>
- CERCLE : --cercle <nom de l'image à créer>  
          <coordonnéeX Centre> <coordonnéeY Centre> <rayon> <Booléen\*> <Couleur\*\*>  
          <largeurImageàCréer> <hauteurImageàCréer>

\*Booléen : True pour Plein  
          False pour Vide

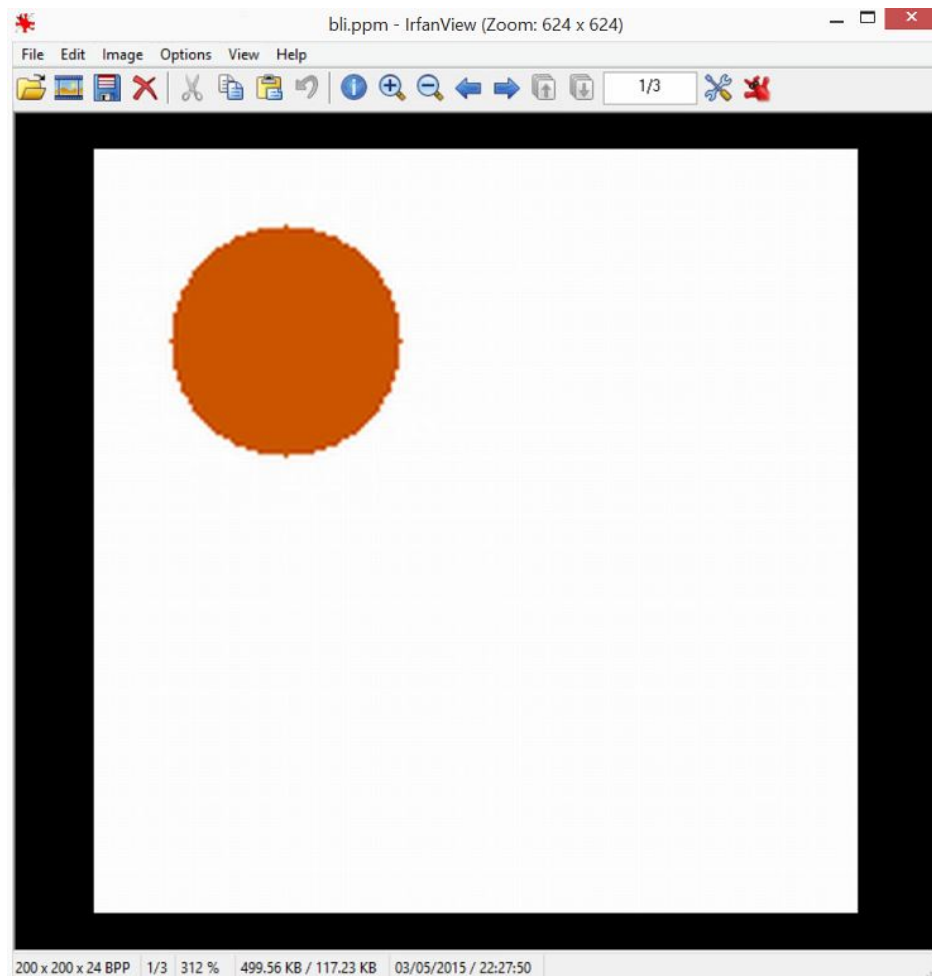
\*\*425 couleurs disponibles

Précéder le nom de la couleur par une majuscule

Pour les couleurs composées, les deux mots ne sont pas séparés d'un espace

(Ex : Rosebonbon)

```
Zack@Bli ~/Projet1/Java+Python
$ ./Synthese.py --cercle bli 50 50 30 True Orangebrulee 200 200
Les arguments sont :
./Synthese.py
--cercle
bli
50
50
30
True
Orangebrulee
200
200
```



Le fichier créé se situe dans le même répertoire que le programme.

### Tutoriel d'utilisation de l'Analyseur (Analyse d'images)

Ouvrir un terminal UNIX.

Aller dans le répertoire où se trouve le programme à partir de la commande **cd**.

```
~/Projet1
Zack@Bli ~
$ cd Projet1/
Zack@Bli ~/Projet1
$
```

Effectuer la commande **make** afin de compiler le programme.

```
Zack@bli ~/Projet1/Java+Python
$ make
javac rechercheFormes\RechercheFormes.java rechercheFormes\DictionnaireCouleur.j
ava rechercheFormes\Couleur.java rechercheFormes\Points.java analyse\Analyse.jav
a
python -m py_compile Synthese.py
Zack@bli ~/Projet1/Java+Python
$ |
```

Écrire ensuite la ligne de commande correspondant à l'action voulue.

(Il faut pour cette étape disposer d'un fichier .PPM à analyser.)

Voici une copie d'écran de notre fichier README.txt où se trouve la liste des commandes et actions disponibles.

- 5 - Taper `java analyse.Analyse <NomImage.ppm >NomFichierTransformé.txt`  
ce qui va transformer le .ppm en .txt afin de pouvoir mieux l'analyser
- 6 - Taper `java rechercheFormes.RechercheFormes <NomFichierTransformé.txt`  
Les résultats de l'analyse s'affichent dans le terminal.

Il est possible de stocker les résultats de l'analyse dans un fichier.txt

Pour cela, rajouter à la fin de la ligne de commande de l'étape 6 :

`>NomFichierOùStockerLeRésultat.txt`

```
Zack@bli ~/Projet1/Java+Python
$ java analyse.Analyse <bli.ppm >bli.txt

Zack@bli ~/Projet1/Java+Python
$ java rechercheFormes.RechercheFormes <bli.txt
largeur de l'image = 200
hauteur de l'image = 200
La forme est un cercle de centre x=50 y=50 et de rayon 30 pixels de couleur Oran
gebrulee
Zack@bli ~/Projet1/Java+Python
$ |
```

L'analyse du fichier est détaillée dans le terminal, ou dans un fichier.txt selon les choix de l'utilisateur lors de l'étape 6 ci-dessus.

## 6. Récapitulatif des séances de TP (à partir de la séance 6)

- Séance 6 : mardi 15 Mars

**Frédéric** : Début du code sur Eclipse (Java) : Affichage d'une matrice à partir d'un fichier pgm.

**Zack & Aurore** : Travail du code sur Python : Fonction « triangle plein » (encore en cours, bug non fixé).

- Séance 7 : mercredi 25 Mars

**Frédéric** : Travail du code sur Eclipse (Java). Problème rencontré : mettre un fichier en argument.

**Zack** : Nettoyage du code Python, fin de la fonction « triangle plein », insertion des couleurs.

**Aurore** : Comprendre comment exécuter un programme à partir de la ligne de commande, écriture des fonctions permettant d'exécuter un programme Python à partir de l'invite de commande.

- Séance 8 : mercredi 1<sup>er</sup> Avril

On a tout fini. POISSON D'AVRIL !

- Séance 9 : mardi 7 Avril

**Aurore** : Création de makefiles afin de pouvoir compiler nos deux programmes en ligne de commande.

**Frédéric & Zack & Aurore** : Travail du code Java (analyse), insertion du code permettant de lire les arguments.

## 7. Description des fonctionnalités supplémentaires

- Création et reconnaissance de la forme Point.

- Création et reconnaissance de la forme Triangle à base autre qu'horizontale.

Notre fonction Triangle appelle 3 fois la fonction Segment. Cette méthode nous permet de faire des triangles de n'importe quelle forme.

- Création de la forme Segment.

- Choix entre des formes géométriques Pleines ou Vides.

Pour remplir une forme, on se sert de 2 tableaux où l'on stock les points à droite et les points à gauche. Il suffit d'ensuite relier entre eux les points de ces tableaux ayant le même y.

- Très large choix de couleurs : 425 couleurs disponibles (détail dans le fichier couleurs.py).