ARTS ET MÉTIERS ParisTech
Bordeaux - Talence

UED SIM – SysIn

Programmation Orientée Objet : Manipulation de tableaux et d'images matricielles
La classe `'ndarray'` de `numpy`
Exemple d'environnement fenêtré (GUI `Qt`)

Jean-luc Charles (`jean-luc.charles@ensam.eu`)
Éric Ducasse (`eric.ducasse@ensam.eu`)

Préambule

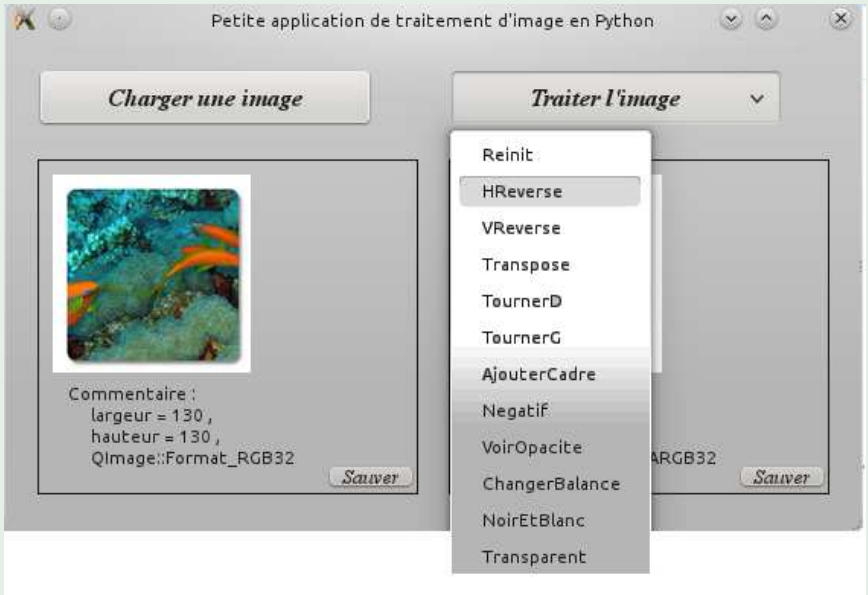
- L'archive `TD-OO-2.zip` est disponible sur SAVOIR (plate forme E-Learning de Arts & Métiers ParisTech)
- L'extraction du contenu de l'archive place tous les fichiers extraits dans le répertoire `TD-OO-2`.
- **Tout le travail qui suit doit être fait dans votre répertoire de travail `TD-OO-2`.**

- Ouvrir le fichier *TraiterImage.py* dans l'éditeur Python IDLE .
- Faire exécuter le fichier par F5. Une fenêtre doit apparaître en haut à droite de l'écran. . .

Introduction

L'objectif de cette séance est de manipuler des tableaux représentant une image matricielle, en complétant une application encapsulée dans une interface graphique.

La partie " *interface graphique* " est déjà écrite : on obtient un fenêtre avec à gauche l'image originale, que l'on va charger de manière interactive, et à droite l'image transformée par un traitement spécifié par le programmeur.



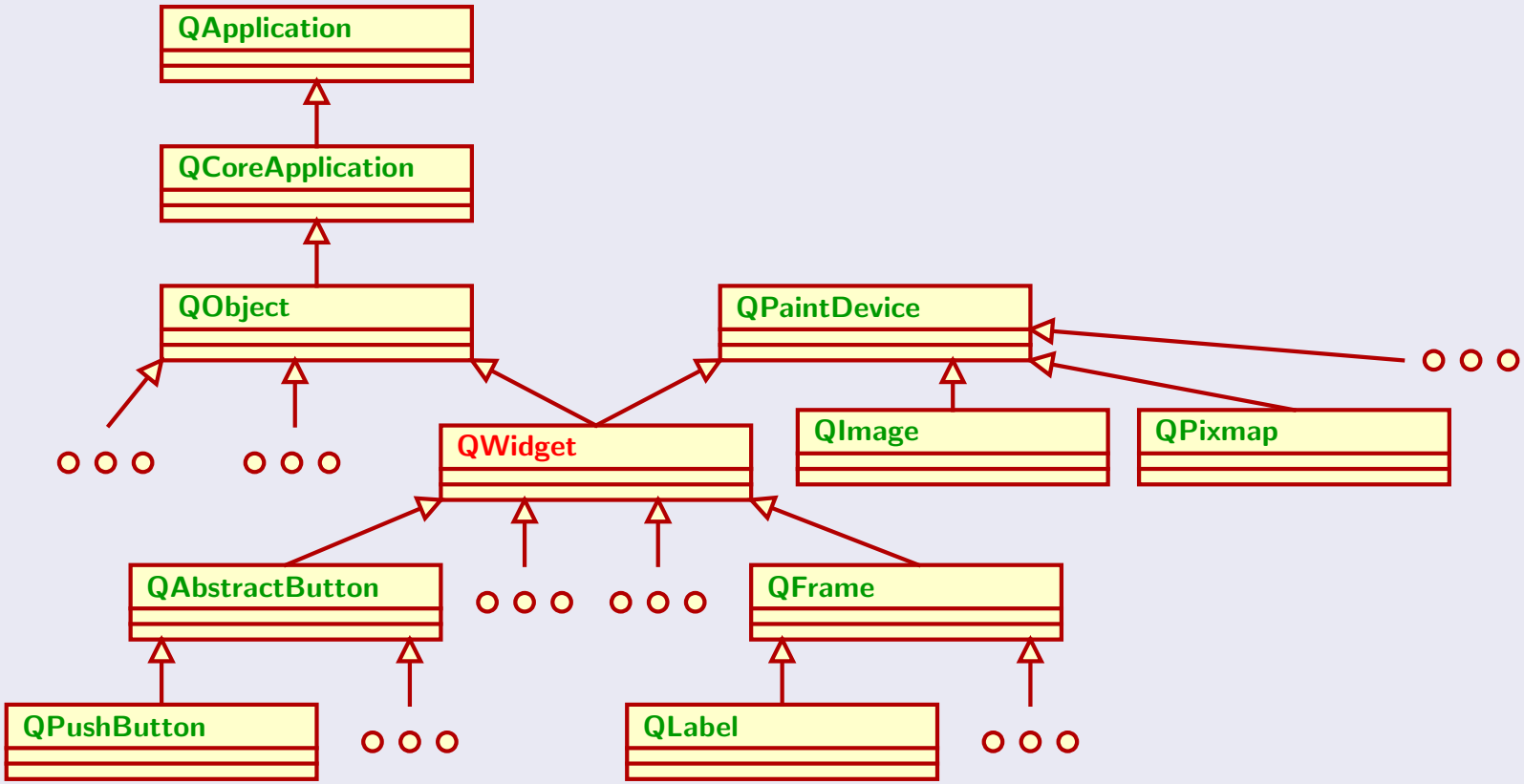
Une image matricielle comporte des pixels. Chaque pixel est codé au format ARGB (Alpha, Red, Green, Blue) sur 4 octets, soit 32 bits. Chaque octet représente un niveau entier de 0 à 255, pour l'opacité (" le niveau α ", valant 1 par défaut) puis les trois couleurs primaires (rouge, vert et bleu).

Après un exposé succinct de la structure du programme et de l'architecture des classes utilisées, le travail à réaliser par l'étudiant est spécifié à la fin du présent document.

Présentation de la bibliothèque Qt

Bibliothèque libre (en partie), complète, multilangage (C++, Python...) et multiplateforme.
Deux modules principaux :

- **QtCore** : objets ne concernant pas l'interface graphique (classes *QPoint*, *QLine*, *QRect*...)
- **QtGui** : tout ce qui concerne le "**G**raphical **U**ser **I**nterface"



Aperçu de l'architecture de classes du module QtGui

Pour en savoir plus sur les classes Qt : http ://pyqt.sourceforge.net/Docs/PyQt4/classes.html

Architecture de l'application

La classe **FenetrePrincipale** contient :

- deux instances de la classe **ZoneImage** pour afficher l'image initiale et l'image traitée,
- une instance de la classe **ARGBArray**, qui stocke les pixels de l'image traitée dans des tableaux de type **ndarray** (classe importée du module **numpy**).

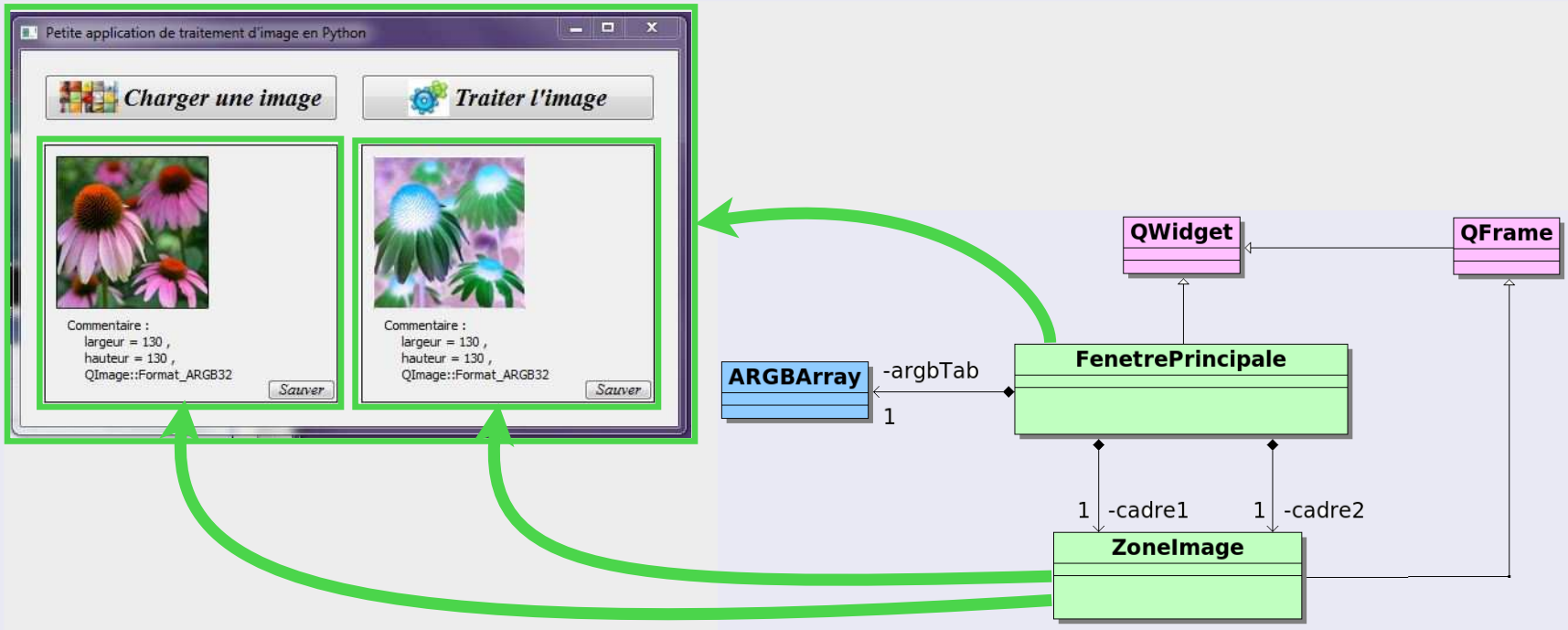
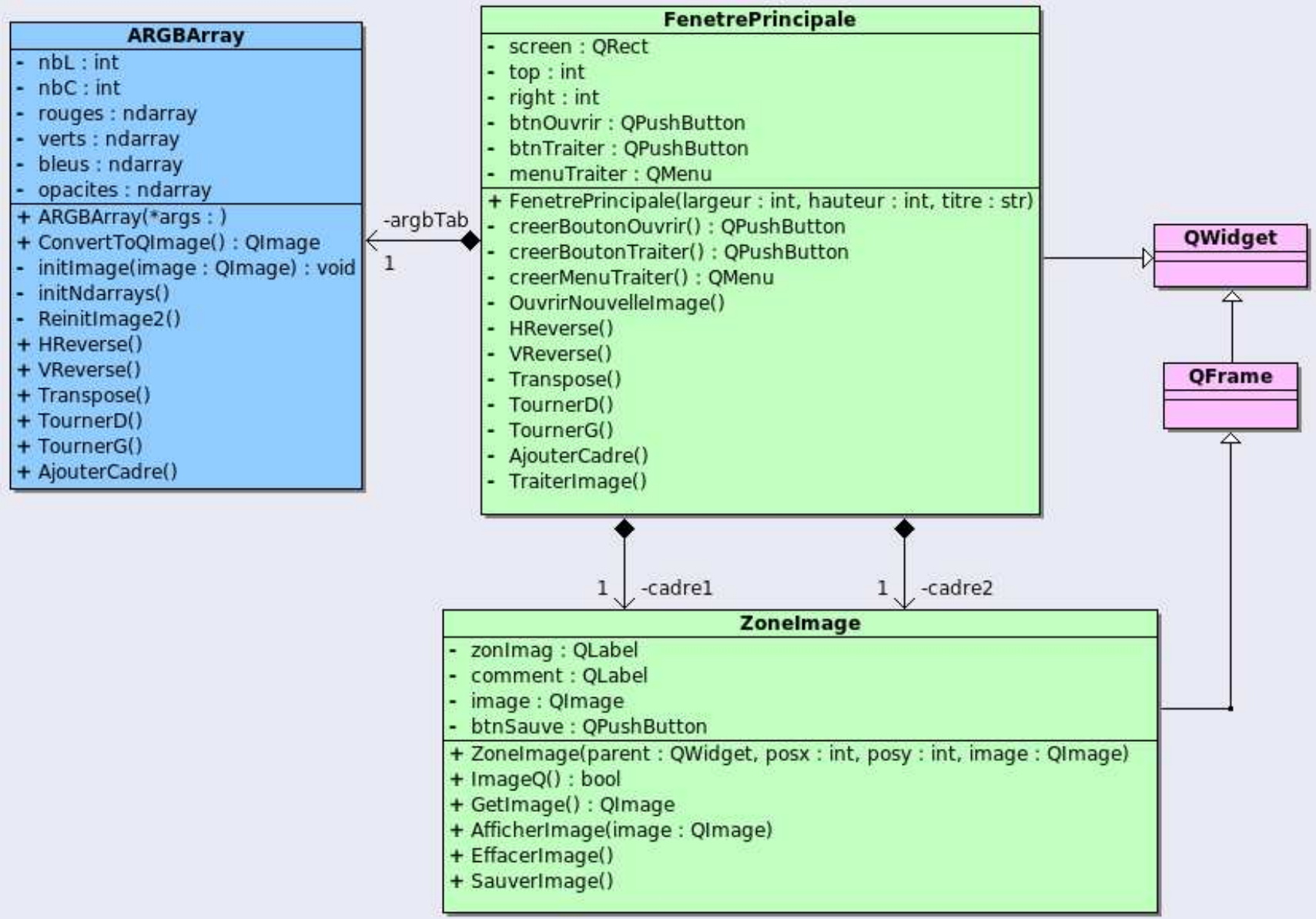


Diagramme de classes

Diagramme UML détaillé



Organisation du programme pour le traitement d'une image

Les méthodes de traitement **HReverse()**, **VReverse()**, ... de la classe **ARGBArray** sont appelées par la méthode **TraiterImage** de la classe **FenetrePrincipale**.

```
...
class FenetrePrincipale(QWidget):
    def __creerMenuTraiter(self):
        # Menu 'traiter' avec ses item
        m = QMenu()
        m.addAction('Reinit', self.__Reinit)
        m.addAction('HReverse', self.__HReverse)
        m.addAction('VReverse', self.__VReverse)
        ...
        return m
    ...
    def __HReverse(self):
        self.__TraiterImage(self.__argbTab.HReverse)
    def __VReverse(self):
        self.__TraiterImage(self.__argbTab.VReverse)
    ...
    def __TraiterImage(self,traitement,*args):
        ...
        # effectuer le traitement choisi
        traitement(*args)
        ...
```

```
...
class ARGBArray :
    ...
    def HReverse(self):
        u'renversement gauche/droite'
        print u"Méthode ARGBArray.HReverse() à compléter"

    def VReverse(self):
        u'renversement haut/bas'
        print u"Méthode ARGBArray.VReverse() à compléter"

    def Transpose(self):
        u'permutation des axes x et y'
        print u"Méthode ARGBArray.Transpose() à compléter"
    ...
```

Attributs de la classe ARGBArray

- **__nbL** : nombre de lignes de pixels
- **__nbC** : nombre de colonnes
- **__rouges** : tableau ndarray de float entre 0 et 1, à deux dimensions
- **__verts** : idem
- **__bleus** : idem
- **__opacites** : idem

La classe ndarray de la bibliothèque numpy

scipy est une bibliothèque de **calcul scientifique** qui utilise numpy pour le **calcul numérique**. La classe ndarray permet de définir un tableau à *n* dimensions (matrices, hypermatrices) contenant des coefficients qui sont tous de même type (tableau homogène).

Méthodes de la classe ndarray pour un tableau à deux dimensions

Constructeur d'un tableau à deux dimensions (matrice) :

- `tableau = ndarray( (nbLignes, nbColonnes), type)`

Extraction du coefficient de la *i*-ème ligne et de la *j*-ème colonne :

- `tableau[i,j]`

Extraction de la *i*-ème ligne :

- `tableau[i]`

Copie de(s) valeur(s) et pas simplement d'une référence :

- tableau entier : `newtab = tableau.copy()`
- ligne entière : `newligne = tableau[i].copy()`
- coefficient : `newcoef = tableau[i,j]`

Permutation des lignes et des colonnes (*attention* : transpose ne modifie pas l'objet) :

- `tableau = tableau.transpose()`

Remplissage :

- `tableau.fill(valeur)`

Pour en savoir plus sur `numpy.ndarray` : http ://docs.scipy.org/doc/numpy/reference/arrays.ndarray.html

Question 1 : retournements

Écrire les méthodes **HReverse** et **VReverse** qui font subir à chaque tableau `__rouges`, `__verts`, `__bleus`, et `__opacites`, un renversement respectivement de chaque ligne et de chaque colonne. Les tester.

Question 2 : rotations

En utilisant la méthode **transpose** de **ndarray**, créer la méthode **Transpose** qui fait une symétrie de l'image par rapport à l'axe $x = y$. Penser à mettre à jour tous les attributs de la classe **ARGBArray**. Tester la méthode sur une image rectangulaire.

En combinant les méthodes **VReverse** et **Transpose**, écrire deux nouvelles méthodes **TournerD** et **TournerG** qui font tourner l'image de 90 degrés respectivement dans le sens horaire et dans le sens trigonométrique. Les tester.

Question 3 : encadrement

On veut pouvoir entourer une image d'une bande de couleur (cadre).

Écrire la méthode **TracerCadre** dont les arguments sont un quadruplet [a, r, g, b] représentant la couleur du cadre, nbx, sa largeur en pixels, et nby, sa hauteur en pixels.

Le même tableau intermédiaire sera utilisé plusieurs fois, en prenant garde à bien faire des recopies par valeurs et non pas de simples *alias*.

Question 4 : autre transformation

Écrire une autre méthode qui opère une transformation de son choix.

Propositions :

- création d'un quadrillage de couleur, de pas selon x et selon y que l'on peut régler
- effets de transparence
- modifier la balance des couleurs, convertir en noir et blanc, obtenir un négatif
- déformer l'image (multiplier sa largeur par deux, ou sa hauteur, ou les deux)
- *etc.*