plan

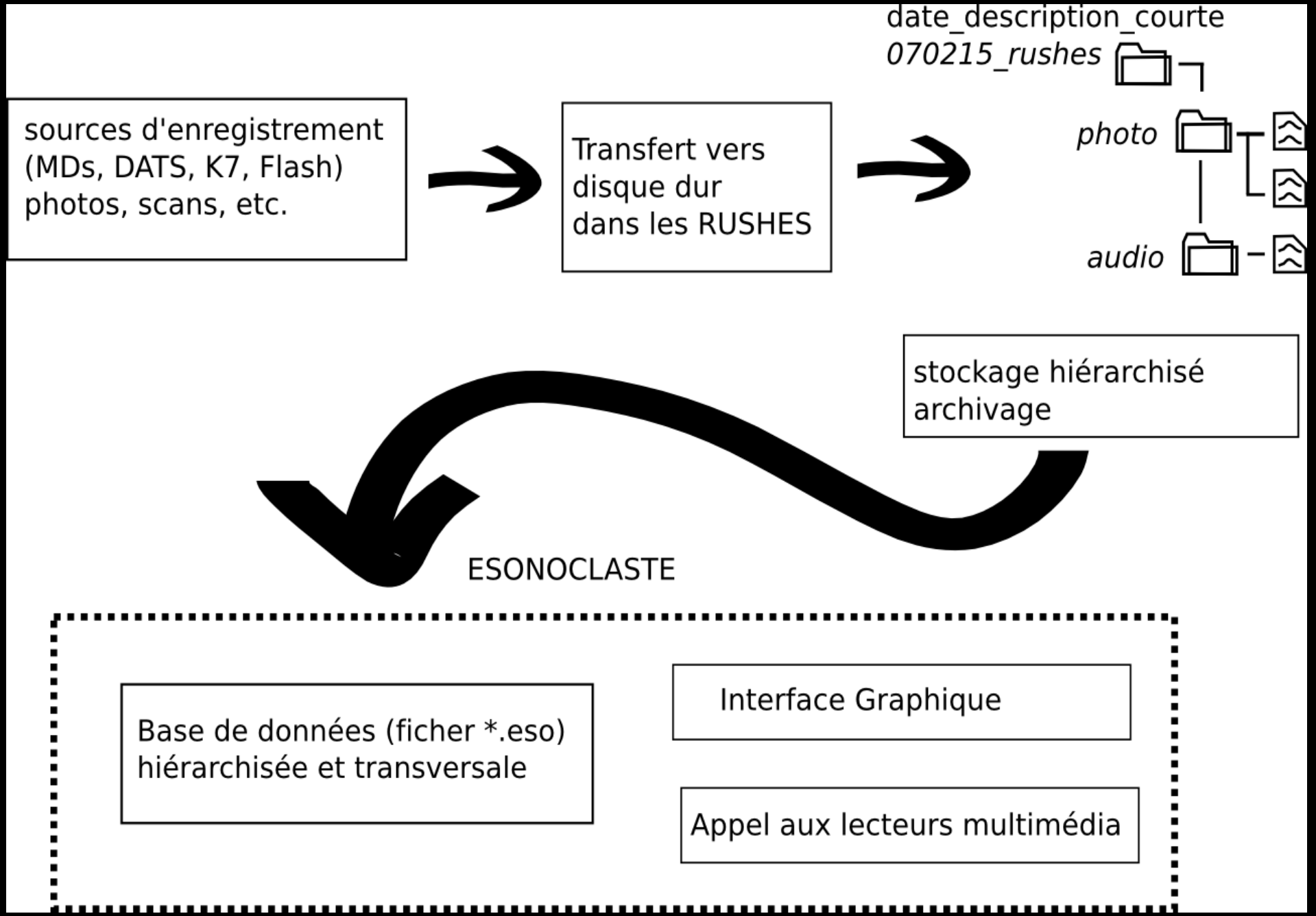
- *e-sonoclaste* ?
- Pourquoi Python?
- Petit historique d'implémentation
- Architecture
- Utilisation de la zodb en « standalone »
- Une démo rapide

Les idées de départ

- Gérer des archives sonores constituées par des enregistrements binauraux.
- Pouvoir se repérer dans plusieurs dizaines d'heures de son.
- Annoter certains moments ou événements (description verbale du sonore).
- Esquisser des montages.
- Nécessité d'une forme d'onde?

Que fait *esono*?

- joue et organise des fichiers multimédias
- indexation à l'aide de marqueurs textuels
- marqueurs structurés (champs)
- arborescence et liste
- écoute et analyse (par ex. ethnomusicologie)
- dérushage / prémontage
- montage de diaporamas sonores

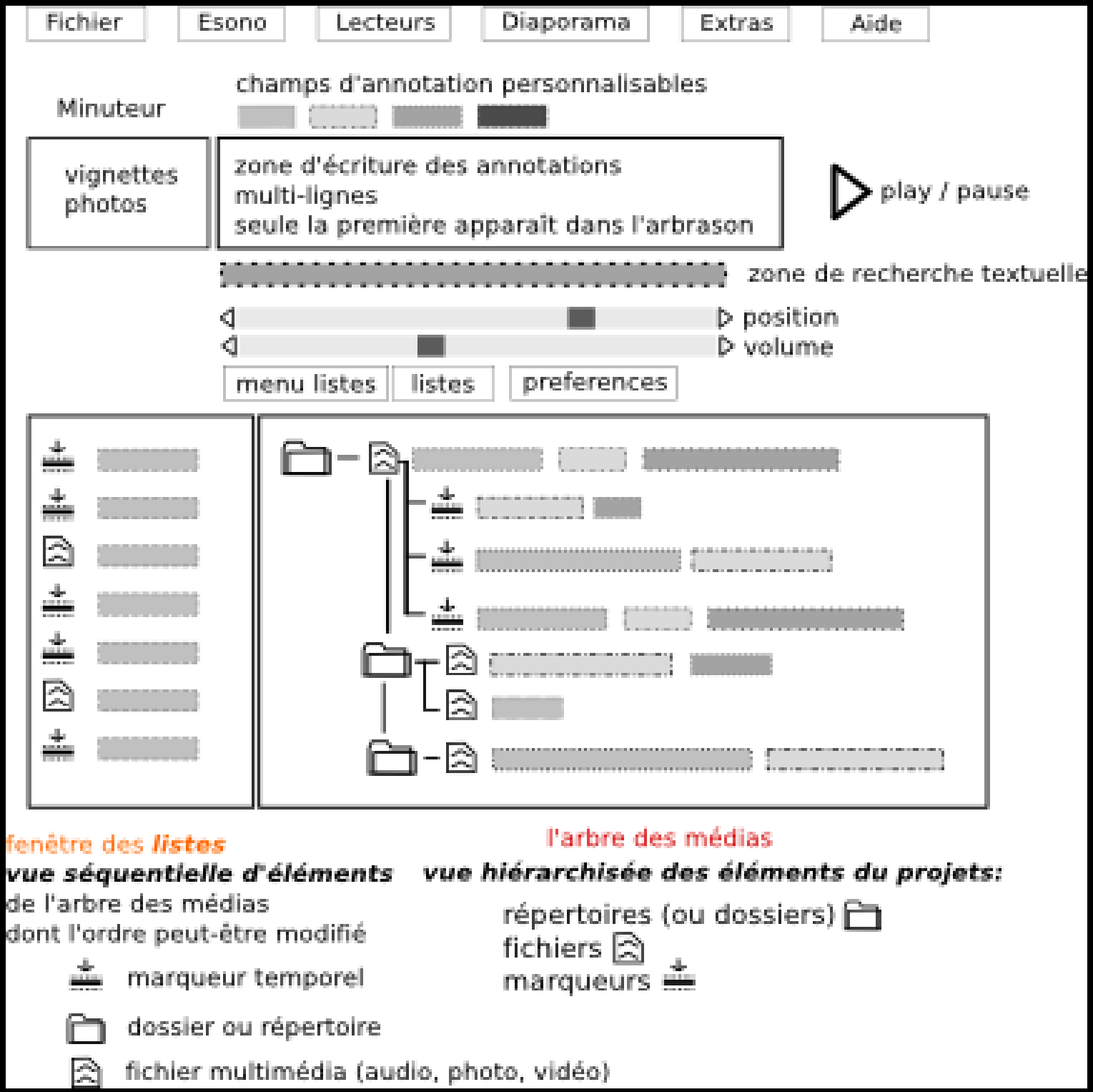

date_description_courte
070215_rushes
sources d'enregistrement
(MDs, DATS, K7, Flash)
photos, scans, etc.
Transfert vers
disque dur
dans les RUSHES
photo
audio
stockage hiérarchisé
archivage
ESONOCLASTE
Base de données (ficher *.eso)
hiérarchisée et transversale
Interface Graphique
Appel aux lecteurs multimédia

historique d'implémentation

- matlab
- python1.5
- emacs, pymacs, regexp
- structures de données sauvegardées en xml (pyxml)
- tkinter (widget texte tableur/traitement texte)
- structures de données gérées par la **zodb**
- télécommande multimédia (xmms, winamp, mplayer, vlc, quicktime)
- intégration de l'unicode
- PIL (photos)
- réécriture du code et dépôt tuxfamily (svn)
- packages précompilés py2app, py2exe
- intégration puredata (et pyext), pour les POMs
- annulation / retour arrière (zodb)
- pysvn, auto-update

pourquoi Python ?

- +
  - orienté objet simple à mettre en oeuvre
  - interprété (*ipython, débuggage, reload*)
  - des modules d'extensions dans tous les domaines
  - multi-plateforme
  - relativement facile à distribuer
- -
  - implémentation unicode peu « pythonique »
  - tkinter seul GUI « pythonique » un peu ancien
  - manque un « case » (?)
  - peut-être moins élégant que *ruby*
  - passage à python 3000...


Fichier
Esono
Lecteurs
Diaporama
Extras
Aide
Minuteur
champs d'annotation personnalisables
vignettes photos
zone d'écriture des annotations multi-lignes
seule la première apparait dans l'arbrason
play / pause
zone de recherche textuelle
position
volume
menu listes
listes
preferences
fenêtre des listes
vue séquentielle d'éléments
de l'arbre des médias
dont l'ordre peut-être modifié
marqueur temporel
dossier ou répertoire
fichier multimédia (audio, photo, vidéo)
l'arbre des médias
vue hiérarchisée des éléments du projets:
répertoires (ou dossiers)
fichiers
marqueurs

Architecture

- classes éléments (rép, fichier, marqueur)
- classes GUI éléments, arbre, listes
- classes télécommande médias
- classes et fonctions
  - découper un son/vidéo
  - retailler une/des images
  - faire une galerie de photo ou un diaporama
  - faire des recherches dans la base
  - ...

utilisation de la zodb (1)

- base de données orientée objet de zope
- un peu magique
- très simple
- s'intègre naturellement dans les classes

```
from persistent import Persistent
class MaClasse(Persistent):
    def __init__(self, mavar):
        self.mavar = 0
    def change(self, valeur):
        self.mavar = valeur
        self._p_changed = 1
```

```
class Db_zodb:
    def __init__(self, path):
        self.initialize(path)
        self.read()

    def initialize(self, path):
        self.storage     = FileStorage.FileStorage(path)
        self.db          = DB(self.storage)
        self.connection  = self.db.open()
        self.root        = self.connection.root()
        self.db_tuple = (self.root, self.connection, self.db, self.storage)

    def read(self):
        if 'maclasse' in self.root.keys():
            maclasse = self.root['maclasse']
        else:
            self.root['maclasse'] = MaClasse(10)

    def save(self):
        transaction.commit()

    def close(self):
        transaction.abort()
        self.db.close()
        self.connection.close()
        self.storage.close()
```

utilisation de la zodb (3)

- implémentation d'un *Undo*

```
self.db.undo(id_undo)
transaction.commit()
```

- compactage

```
self.db.pack()
```

autres fonctionnalités de la zodb (non utilisées à l'heure actuelle dans esono)

- utiliser Zcatalog pour les recherches
- extraire une sous-partie d'une zodb
- regrouper des zodb
  - mount points

autres développements

- import / export xml
- choix d'une autre GUI (pygtk, wxwin ou pyqt)
- traductions en anglais
- compléter la doc

notes conclusives

- site web

http://esonoclaste.net

- si vous êtes intéressés à participer au développement, à la doc ou à la traduction, merci de rejoindre la liste de diffusion

http://groups.google.com/group/esonoclaste/