CNAM – Aix en Provence

Valeur C - RSM

Conception d'Applications Multimedia

Programmation Multimedia

Session 4

Erwan TRANVOUEZ
Maître de Conférences

erwan.tranvouez@polytech.univ-mrs.fr
http://erwan.tranvouez.free.fr

Objectifs de la session :

- Appréhender les concepts sur lesquels s'appuient la numérisation d'images en mouvement ainsi que leur intégration dans un environnement logiciel.
- Mise en œuvre avec JMF…

Plan du cours

1. Principe généraux

Elements d'Architecture …

2. Le son et Java : l'API JavaSound

3. La vidéo et Java : l'API JMF

1. Généralités

- Vidéo, compression, formats vidéos, …

Capture d'images en mouvement

- **Capacité à re-transmettre sur un support (visuel ou de stockage) des images en mouvements.**
- **Principe général :**
  - capturer des « images » à un rythme rapide.
  - les afficher au même rythme pour donner l'impression du mouvement (persistance rétinienne).
- **Type de capture :**
  - Film « cinéma » : procédé physico-chimique: via un obturateur, des images sont fixées sur un film photosensible à un rythme de 24 images/seconde.
  - Film vidéo analogique : au travers de tubes cathodiques transformation d'une image en signal électrique.
  - Film vidéo numérique : capteurs photosensibles traduisant une intensité lumineuse en signal numérique.

Signal analogique vs. Signal numérique

Analogique

- **Principe** : image(s) codée(s) sous la forme d'un signal électrique, magnétique (K7) ou électro-magnétique (ondes télévision).

-
  - Données traitées en temps réel au niveau matériel => performance

-
  - Dégradation image à la recopie
  - Qualité limitée
  - Sensibilité au bruit et aux parasites

Numérique

- Principe : image(s) codée(s) sous la forme de séquences de 0 et de 1

-
  - Capacité de traitement (modification, montage, effets…)
  - Recopie données sans pertes
  - Qualité contrôlable et paramétrable

-
  - Traitement des données « coûteux » en capacité de calcul

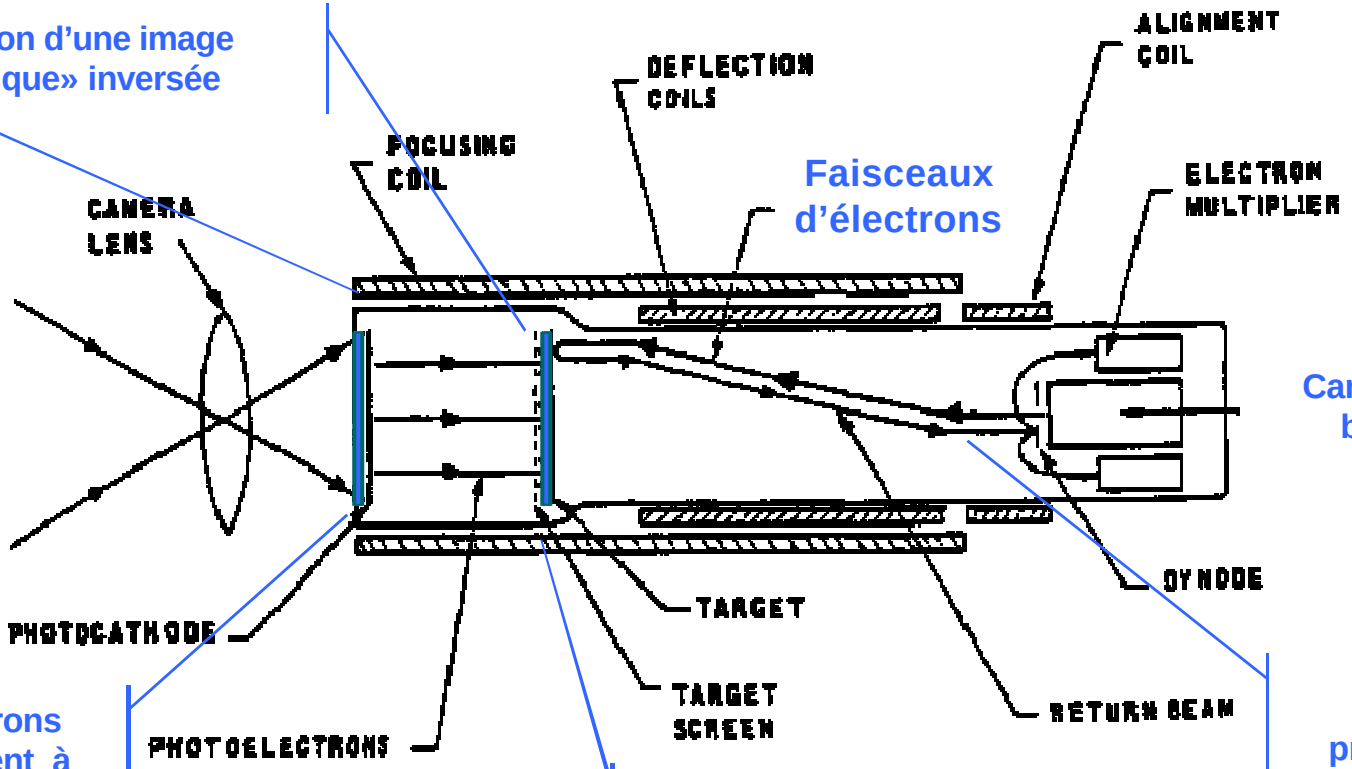
Fonctionnement d'une caméra vidéo

Exemple de tube cathodique : l'orthicon qui mesure la *luminance* d'une image .

Formation d'une image «électrique» inversée

Formation d'une image

Monde « réel » & objets du monde



Faisceaux d'électrons

Canon à électrons balaye la cible

Produit des électrons proportionnellement à l'intensité de la lumière

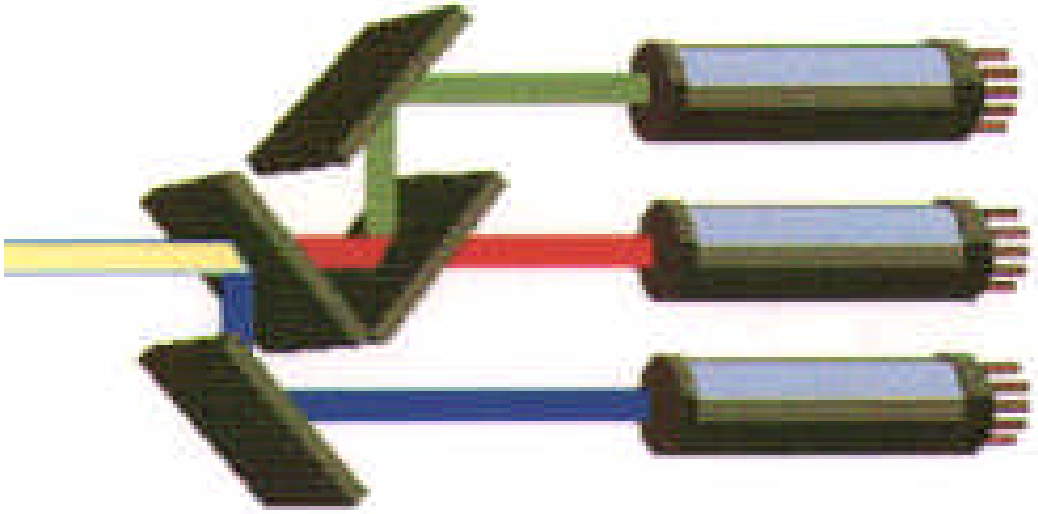
La *cible* est en un matériau photosensible qui produit des électrons proportionnellement à l'intensité de la lumière

Au retour : nb d'électrons proportionnel à la luminosité

Source image Wikipedia

… et la couleur

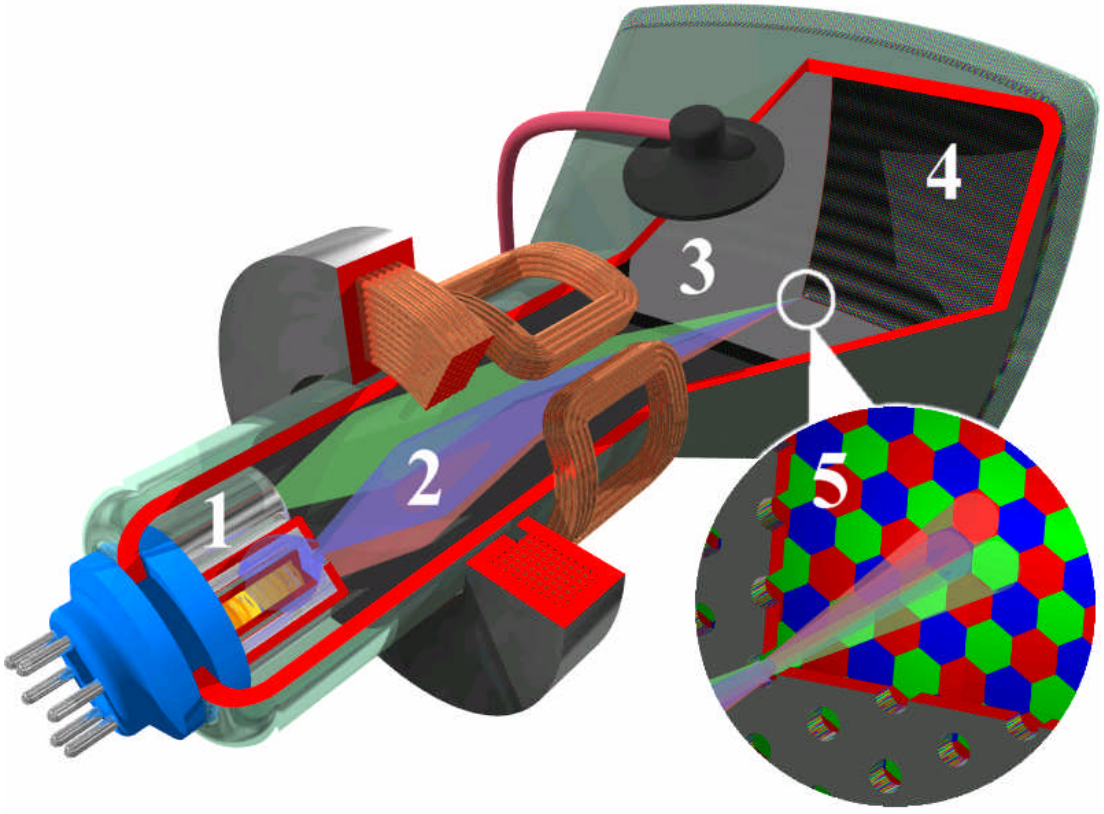
- **Même principe la *chrominance* avec décomposition du signal lumineux en 3 composantes Rouge/Vert/Bleu (RGB) via des filtres (miroir qui ne laissent passer qu'une couleur et rejette les autres)**
- **Il suffit alors de mesurer pour chaque couleur l'intensité lumineuse.**



Source image Wikipedia

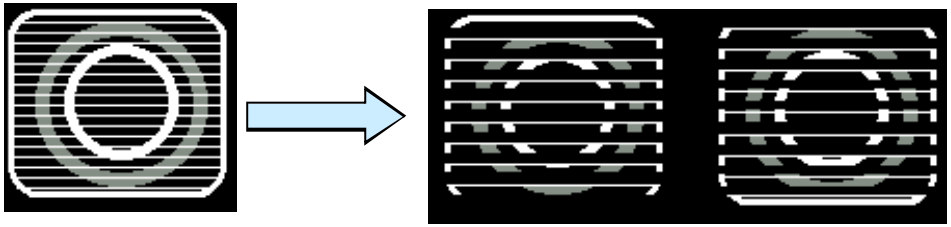
Restitution et affichage du signal

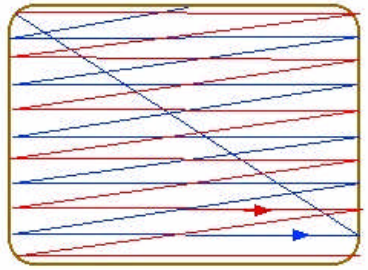
- **Le signal reçu active du tubes cathodiques qui éclairent chacun une zone particulière de l'écran …**
- **Un point de l'écran est en fait constitué de 3 points accolés correspondant aux 3 couleurs RGB.**

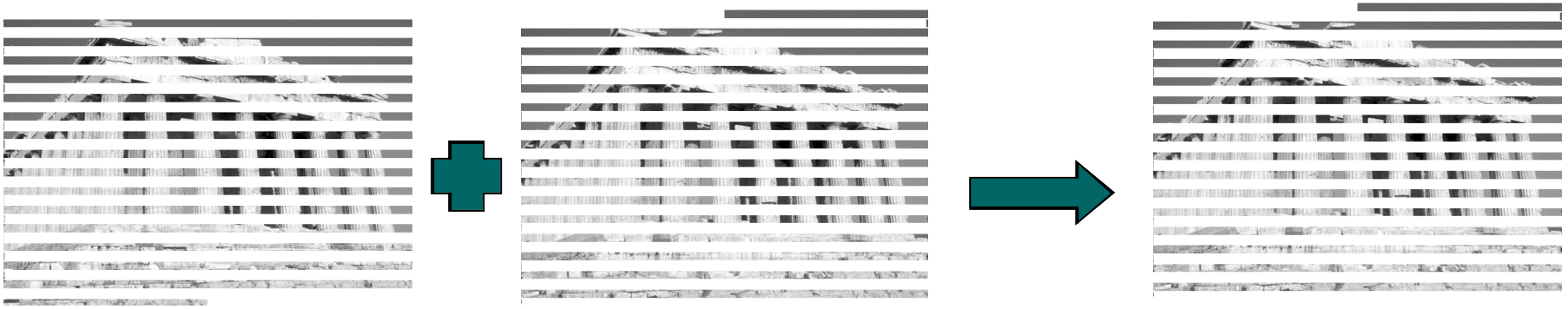


Zoom sur l'entrelacement

Principe d'entrelacement: le faisceau ne dessine pas image par image mais affiche une ligne sur deux d'une image et sur l'autre ligne… une ligne sur 2 de l'autre image. La combinaison des deux forme une trame. Une image est donc affichée en 2 x. Cette technique permet d'éviter les scintillements liés à une fréquence d'affichage insuffisante.

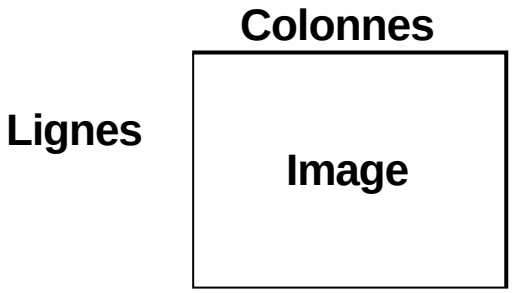






Source http://www.danalee.ca/ttt/index.htm . Rque: GIF animé.

Un peu d'arithmétique …



Colonnes=4/3 x Lignes

Standard	Nb lignes	Nb lignes "utiles"	Fréquence (trame/s)
PAL/SECAM	625	476	27
NTSC	525	423	30

- **Le standard Rc601(*) de l'UIT (Union Internationale des télécommunications) définit la résolution d'une ligne à 720 points actifs. Globalement le système PAL/SECAM requiert 864 points par ligne au total (resp. 858 pour NTSC).**
- **Calcul du débit le système PAL :**
  - Nb de points = 625 l x 864 pt/l x 25i/s = 13 500 000 pt/sec
  - Volume données : 13 500 000 x 3 x 1 o ˜ 40 Mo / s !
  - Ainsi un film d'1h30 occuperait ˜ 90 x 60 x 40 ˜ 210 Go
  - soit plus de 40 DVDs …

? **Compression des données indispensable**

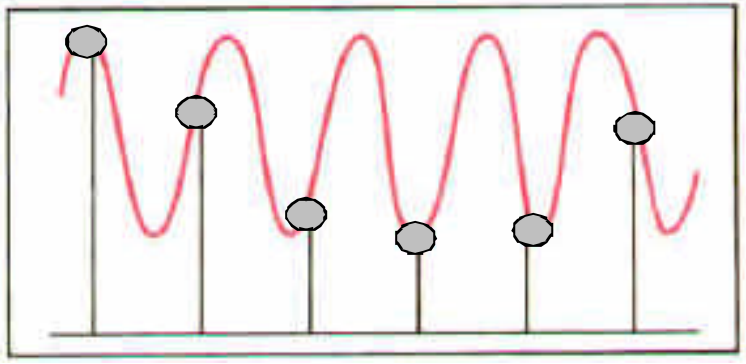
(*) ex. CCIR 601 (Comité Consultatif International des radiocommunications)

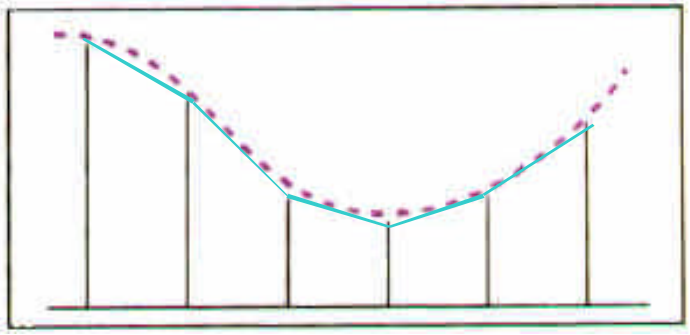
Aparté: conversion Analogique->Numérique

- **Signal analogique :**
  - analogie avec une courbe : y=f(x) avec donc théoriquement un nombre infini de valeur (x, f(x)).
- **Signal numérique :**
  - nombre fini de valeur avec l'intervalle de valeur qui dépend de la mémoire (ie précision) consacrée (désirée).
  - au final codé en bits ou octets : 2^n par valeur.
- **Conversion Analogique / Numérique** :
  - Extraire du signal numérique un nombre fini de valeurs représentatives du signal initial
  - Calcul de discrétisation => approximation

Illustration

- **Echantillonage :** principe consistant à retenir un sous-ensemble de valeur représentative d'un ensemble beaucoup plus grand (ex. sondage).
- **Echantillonnage et numérisation**
  - f(x) pris à intervalle régulier et stocké
  - on a donc discrétisé f(x)
- **Problème :**
  - Quelle intervalle prendre ?
  - Les points perdus sont ils importants ?
  - Exemple d'un mauvais intervalle





Concepts

- **Fréquence d'échantillonnage = nb de mesures par sec hz.**
- **Donc ici : 5 Hz**

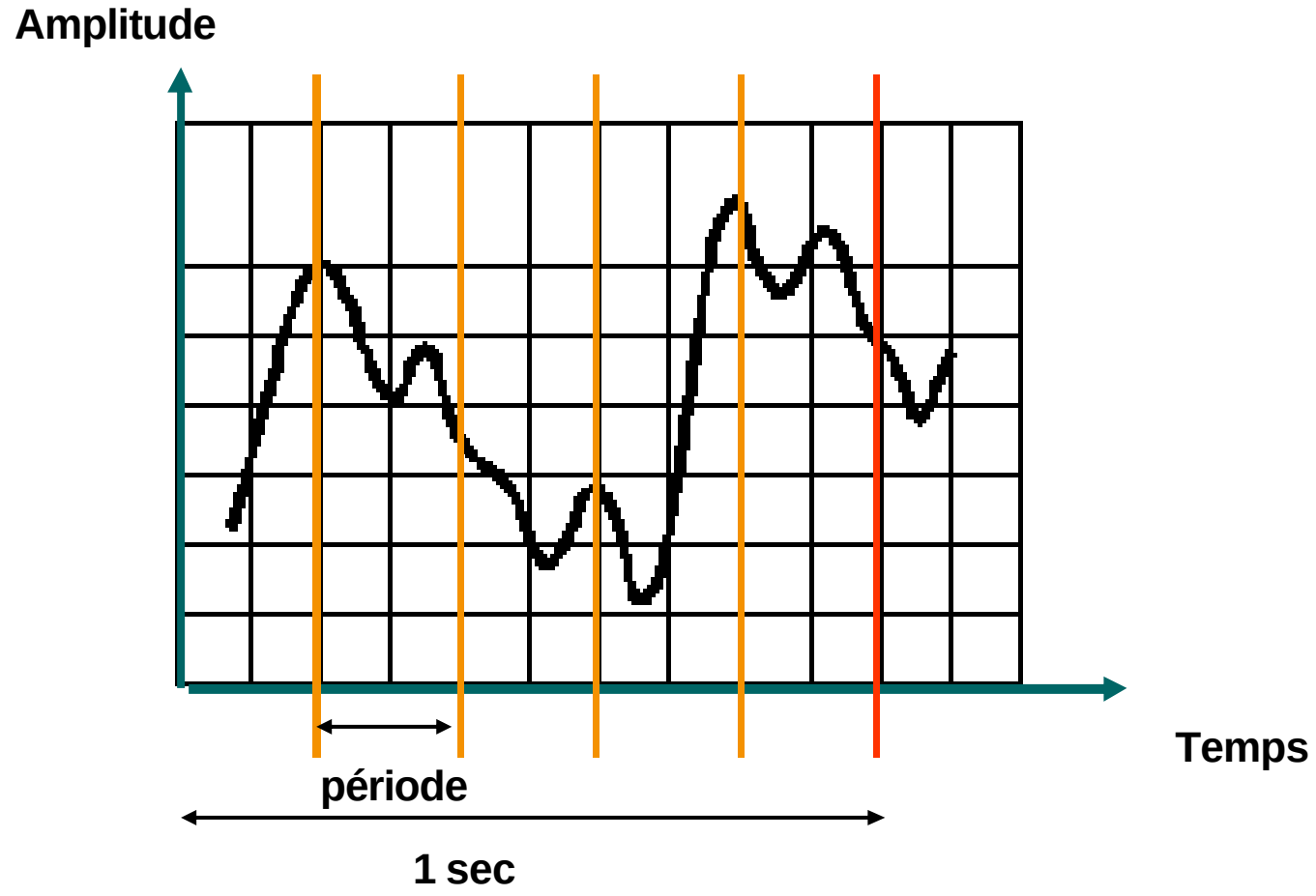
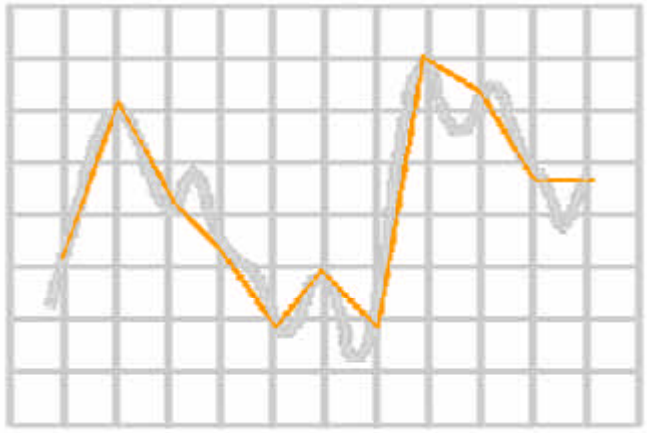
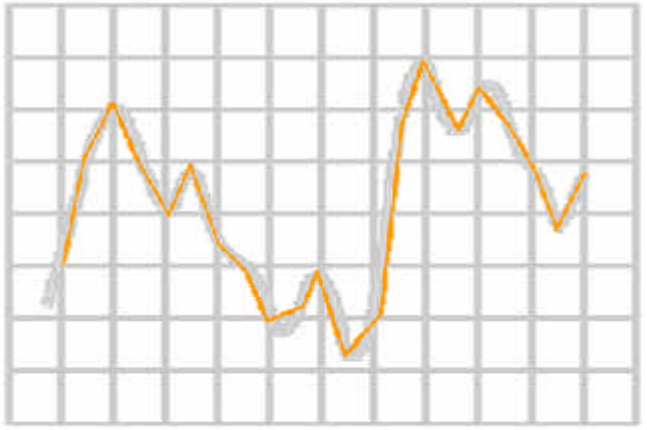


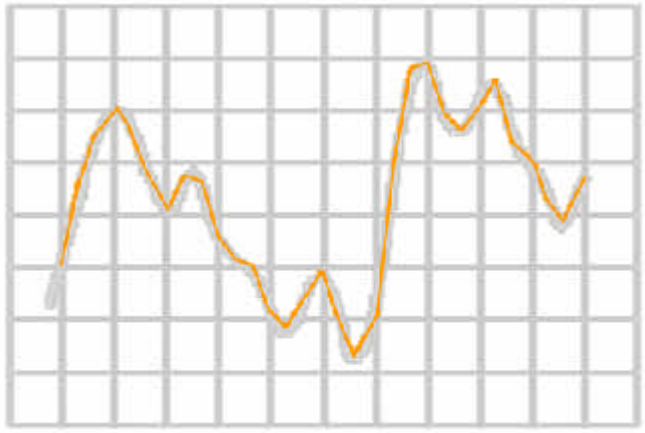
Illustration d'échantillonnage



11 points



22 points



33 points

Où s'arrêter ?

=> Théorème de Shannon :

La fréquence d'échantillonnage doit être 2x plus grandes que la plus grande des fréquences

Principe: approximation par une transformation de fourrier

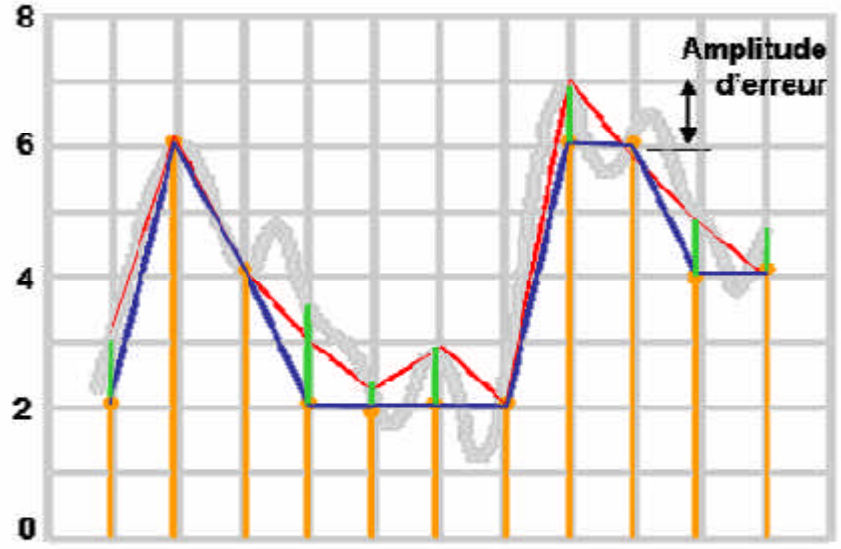
Exemple de fréquences d'échantillonage…

Fréquence	Applications
8 kHz	Téléphonie
11,025 kHz	Echantillonnage Mac
22,05 kHz	Echantillonnage PC
32 kHz	HDTV (High Definition TeleVision), radio numérique et Nicam
44,1 kHz	CD Audio
48 kHz	DAT : Digital Audio Tape
96 kHz	DVD Audio – 6 canaux (5.1)
128 kHz	DVD Audio – 2 canaux (Stéréo)

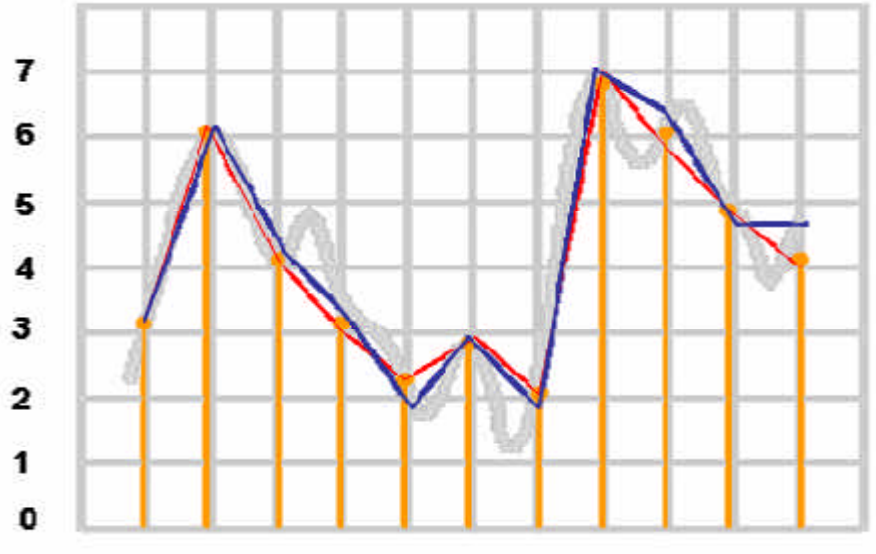
Résolution des valeurs à stocker

- **Le *nombre* de valeur à stocker étant fixé, avec qu'elle *précision* stocker chaque valeur ?**
  - 1 bit : {0, 1}, 1 octet [0, 255], 2 octets : [32768, 32767] etc…
- **Définition de l'amplitude désirée ie l'écart entre 2 valeurs successive** (ex. Nb entier =1)
- **=> nécessité d'approximer :**
  - par troncature : ie partie entière
  - par arrondi : diminue l'erreur maximum à ½ amplitude
- **C'est une manière de compresser !**
  - Nb de points par image
  - Nb de niveau de luminance
  - Nb de niveau de chrominance
  - => Exemple application à la numérisation du signal analogique reçu par une télévision (acquisition)

Illustration quantification

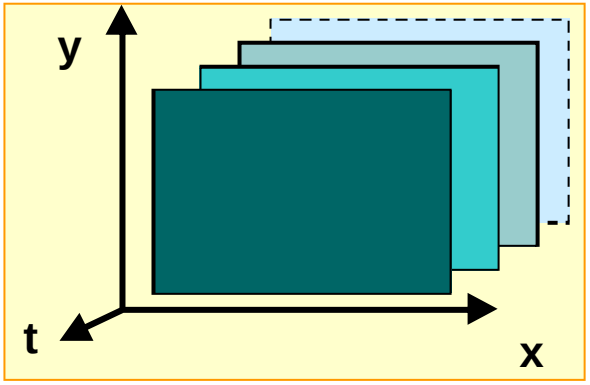


Bruit de quantification



Compression de flux vidéos

- **Objectif: gain de mémoire en factorisant l'information afin d'éviter des redondances inutiles :**
  - vieil antagonisme Données <-> Traitement : Plutôt que "d'apprendre par cœur", on va définir des algorithmes permettrant de retrouver une information à partir d'un ensemble d'information réduit.
- **Compression spatiale :**
  - Optimiser la mémoire consacrée à la représentation d'1 image
  - => Compression du film vidéo revient à compresser chaque image (ex. JPG) … puis à les assembler
  - Principe employé par le format MJPEG (M pour Motion) et DV (Digital Video) qui permet ainsi un montage image/image



- **Compression temporelle**
  - Profiter des similitudes entre 2 images filmées sur un même plan afin d'éviter de stocker des informations «inutiles»
  - => Calculer les différences entre 2 images successives et en déduire des opérations de transformation
  - Stocke quelques images et des opérations de transformation

Sous Echantillonnage…

- **Passage d'une représentation RGB au format YUV profitant des caractéristiques de l'œil humain**
- L'œil a des sensibilités différentes en fonction
  - **des couleurs (a énergie égale) :**
    - **Vert : 1 - Rouge : 0,5 - Bleu : 0,2**
  - **de la luminance (parametre gamma):**

    perception non linéaire

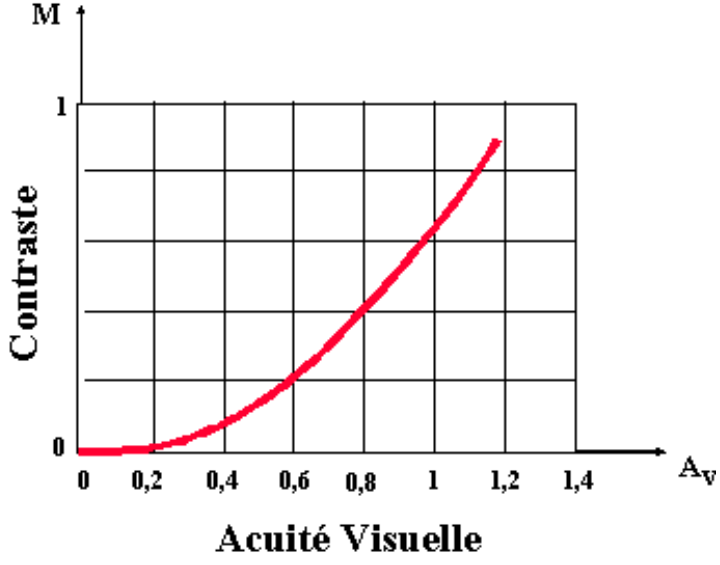


Illustration YUV

Formules (avec RGB « gammatisé ») :

Luminance (*luma*) :

- Y=219x(0.299xR + 0,587xG + 0.144xB) + 16

  avec Y ? [16, 235] (220 niveau de gris : Noir=16 & Blanc=235)

Chrominance Bleu (U):

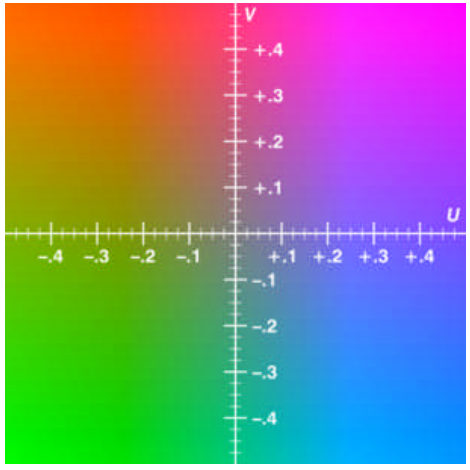
- $C_B = 128 + 112x\ 0{,}886^{-1}x\ (B-Y)$

  C_B ? [-0,436,0.436]

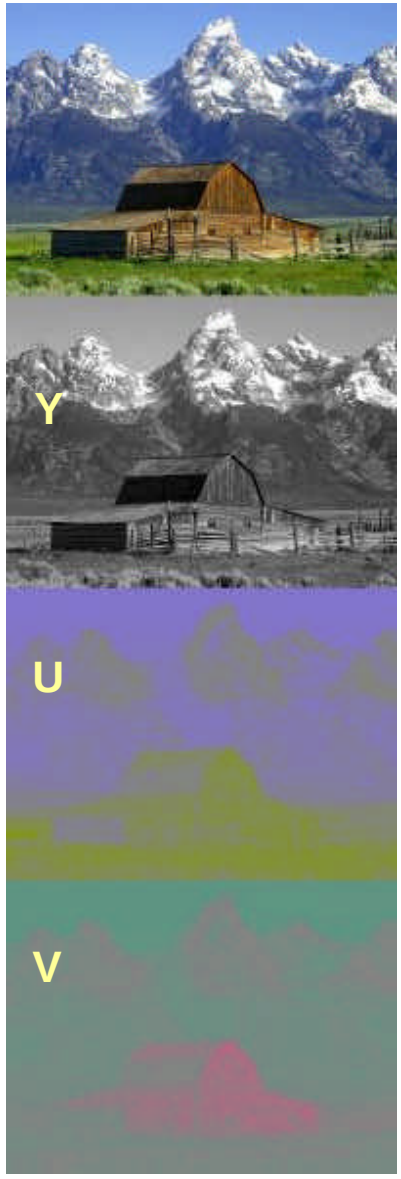
Chrominance Rouge (V) :

- $C_R = 128 + 112x\ 0{,}701^{-1}x\ (R-Y)$

  C_R ? [-0.615,0.615]



C_B eq. U en et C_R eq. V en Sécam



Compression spatiale

- **Avantages** :
  - Décompose RGB en 1 signal noir&blanc et 2 autres signaux => compatible avec téléviseur Noir&Blanc
  - Du fait de l'insensibilité de l'œil à certaine nuances de rouge et bleu l'intervalle de valeur peut être réduit (ex. NTSC gain sur les couleurs initiale ie RGB 11% pour le bleu et 30% pour le rouge).
  - Il est possible alors de se permettre de sous-échantillonner C_R et C_B. et par exemple décrire 4 pixel par une seule valeur UV (cf. ci-après).
- L'image est échantillonnée aux taux a:b:c. Par exemple 4:2:2 signifie que C_B et C_R sont 2x moins échantillonnés que Y (nb de points inférieur).

? **Constitue une forme de compression !!!**

? **… qui n'interdit pas après d'autres mode de compression…**

(Rappel) Compression spatiale

- **Exemple de**
- **sous-échantillonnage**

L,CrCb	L	L,CrCb	L	L,CrCb	L	L,CrCb	L
L,CrCb	L	L,CrCb	L	L,CrCb	L	L,CrCb	L
L,CrCb	L	L,CrCb	L	L,CrCb	L	L,CrCb	L
L,CrCb	L	L,CrCb	L	L,CrCb	L	L,CrCb	L

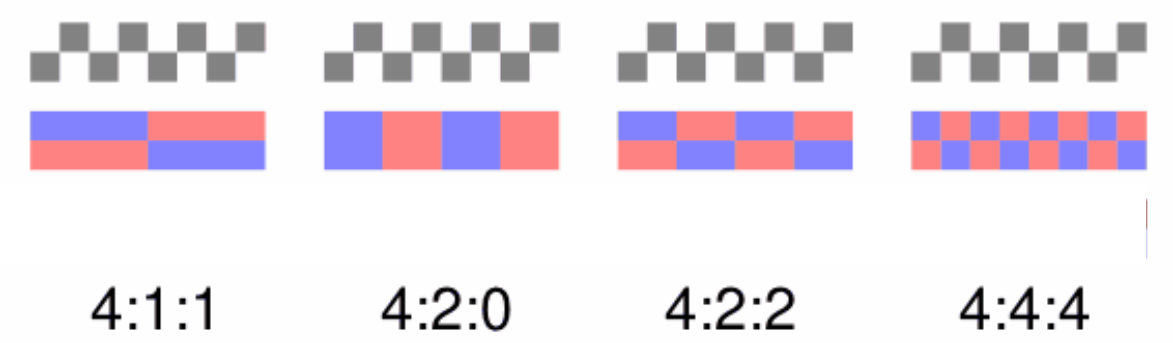
Format 4:2:2

L,CrCb	L	L	L	L,CrCb	L	L	L
L,CrCb	L	L	L	L,CrCb	L	L	L
L,CrCb	L	L	L	L,CrCb	L	L	L
L,CrCb	L	L	L	L,CrCb	L	L	L

Format 4:1:1

L CrCb	L	L CrCb	L	L CrCb	L	L CrCb	L
L	L	L	L	L	L	L	L
L CrCb	L	L CrCb	L	L CrCb	L	L CrCb	L
L	L	L	L	L	L	L	L

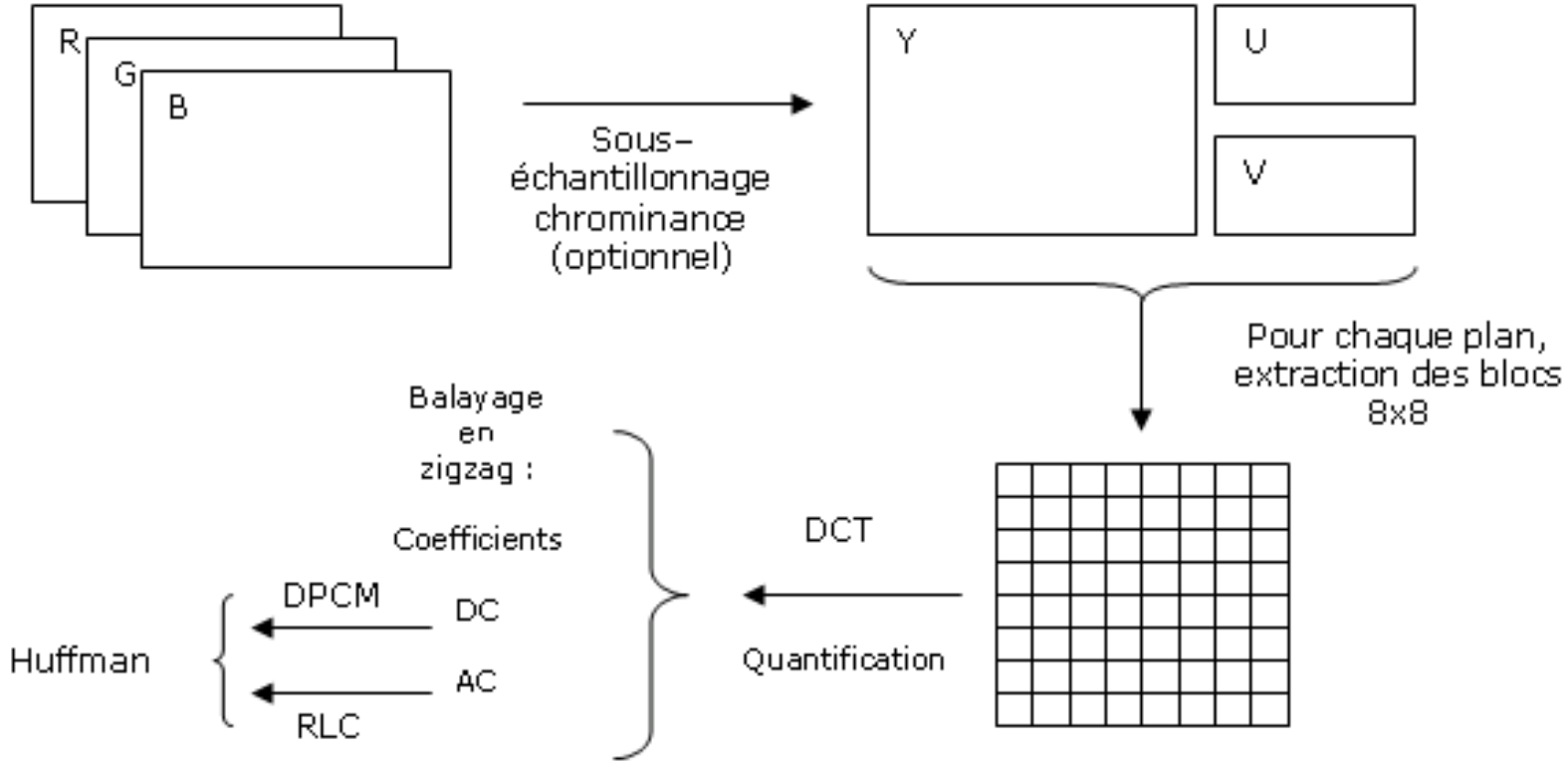
Format 4:2:0



Rque : 4:2:0 signifie en fait 4:2:2 avec sous échantillonnage vertical 2:1 ~ 4:1:1

(Rappel) Compression spatiale

Rappel Compression JPEG



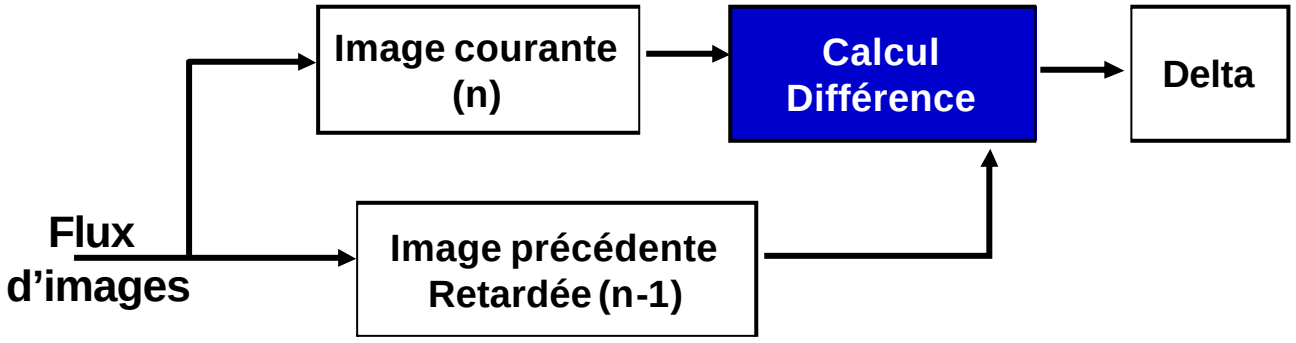
Conséquence de la compression spatiale

Standard	Résolution	Nb images / s	Nb images sur 1h30 (x 5400)
NTSC	640 x 480	29,97	161 838
PAL	768 x 576	25	135 000
SECAM	768 x 576	25	135 000
HD 720 p	1280 x 720	24	129 600
HD 1080i	1920 x 1080	24	129 600

? **nb important d'image : compression spatiale insuffisante (hormis usages spécifiques)**

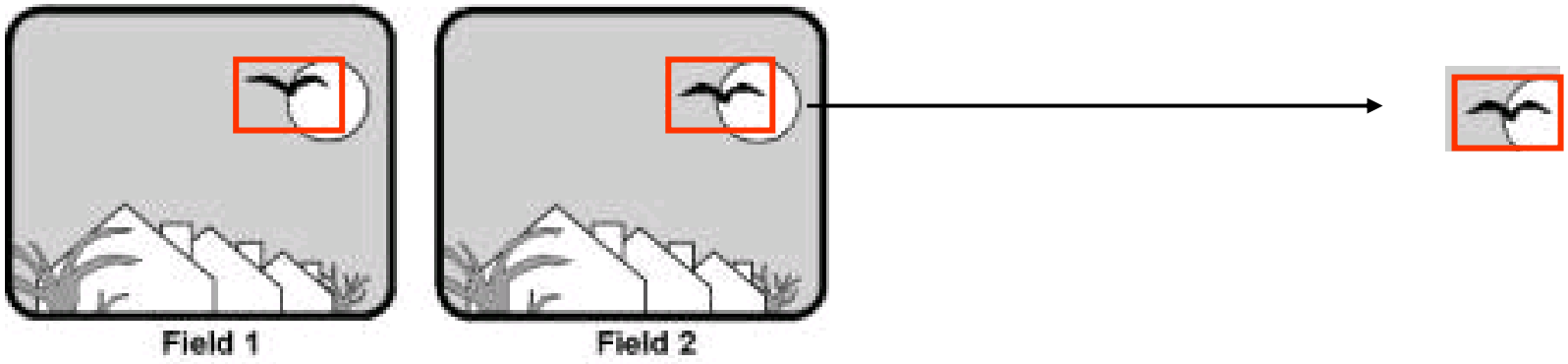
Compression temporelle **uni**directionnelle

- **Principe général :**
- **CODAGE**
  - Sélection d'une image clé (appelée ***intra***) qui ne sera pas compressée temporellement
  - Sélection de l'image suivante
  - Comparaison et extraction des régions différentes => image ***Delta***
  - Éventuellement compression spatiale de l'image Delta



- **DéCODAGE**
  - Décompression image Delta puis transposition à l'image précédente.

Remarques :



- Une fois Delta identifiée on peut « reconstruire » l'image 2 puis refaire l'opération avec l'image 3 avec cette image…

Remarques :

- **Bien que tentant, la généralisation de cette approche n'est pas souhaitable : ie sélection de la 1ère image puis calcul de différences sur toutes les suivantes du film**
  - Car alors faire une avance rapide de l'image i à l'image i+n suppose de calculer les n images après i !
  - Une erreur sur 1 image se répercute sur toutes les autres
  - Passage d'une image à une autre peut être complètement différente (ex. générique)
  - Rend impossible toute opération de montage… (cf. MJPEG)
- **Solution :**
  - Alterner image Intra & Images delta : la fréquence dépendra du type de séquence.
    Ex. opposés: image de paysage et
    image d'événements sportif (avec caméra mobile)
  - Complément : la fréquence d'image peut etre variable ie définie en fonction de la video et non déterminée à l'avance…

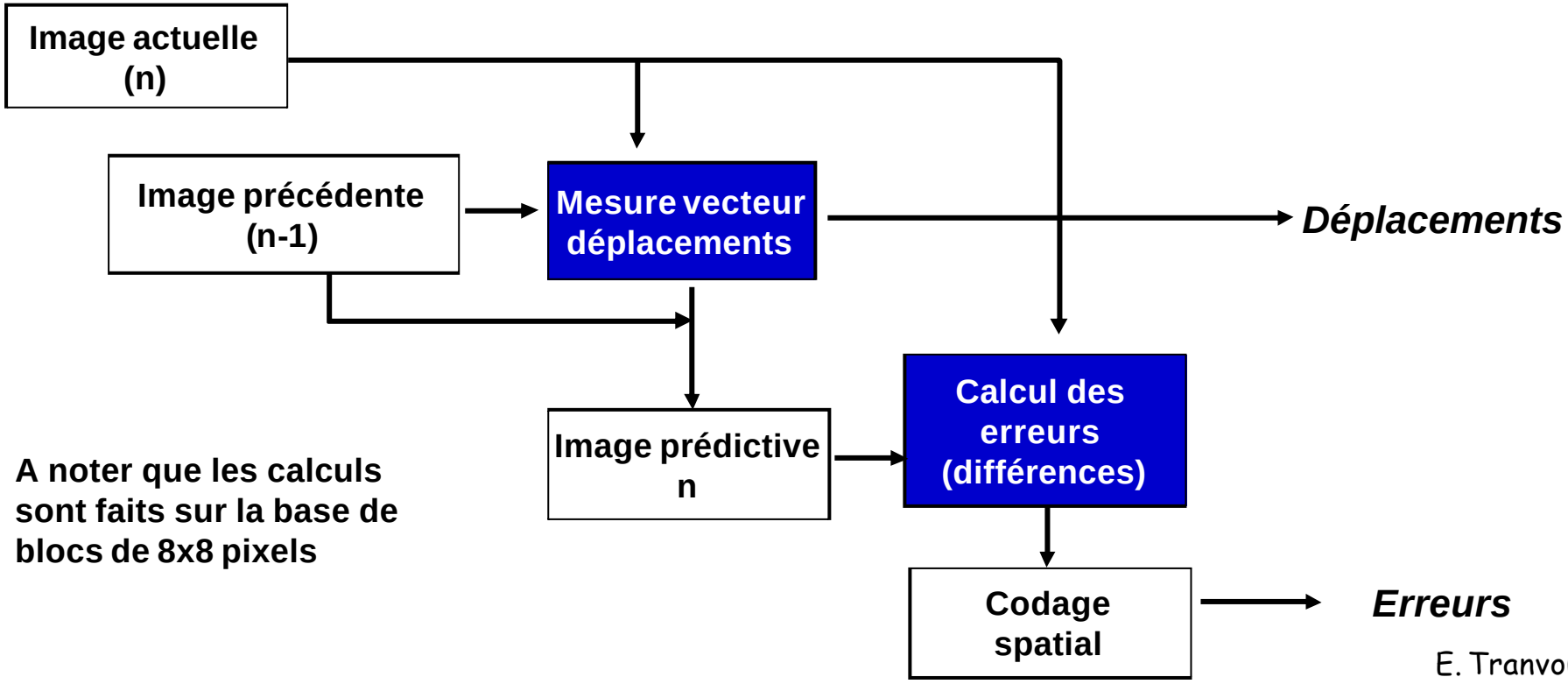
? **Ex; de codec : Cinepak, Indeo (Intel Video)**

Compression temporelle **bi**directionnelle

- **Principe général :**
  - Considérons un mouvement de caméra de gauche à droite: au fur et à mesure la différence entre l'image intra et la suivante est importante (en tout cas + qu'avec un plan fixe)
  - *Idée:* considérer l'image d'après afin de mieux mesurer ce qui reste commun
- **Contrainte:**
  - Il ne s'agit plus seulement d'identifier des zones identiques mais également de les reconnaître à une position éventuellement différente (déplacement de la zone)
  - La phase de comparaison est donc plus lourde: elle met en œuvre des calculs de ***compensation de mouvement*** *(motion compensation)***.**
  - Donc Delta = régions (blocs) identiques + translations

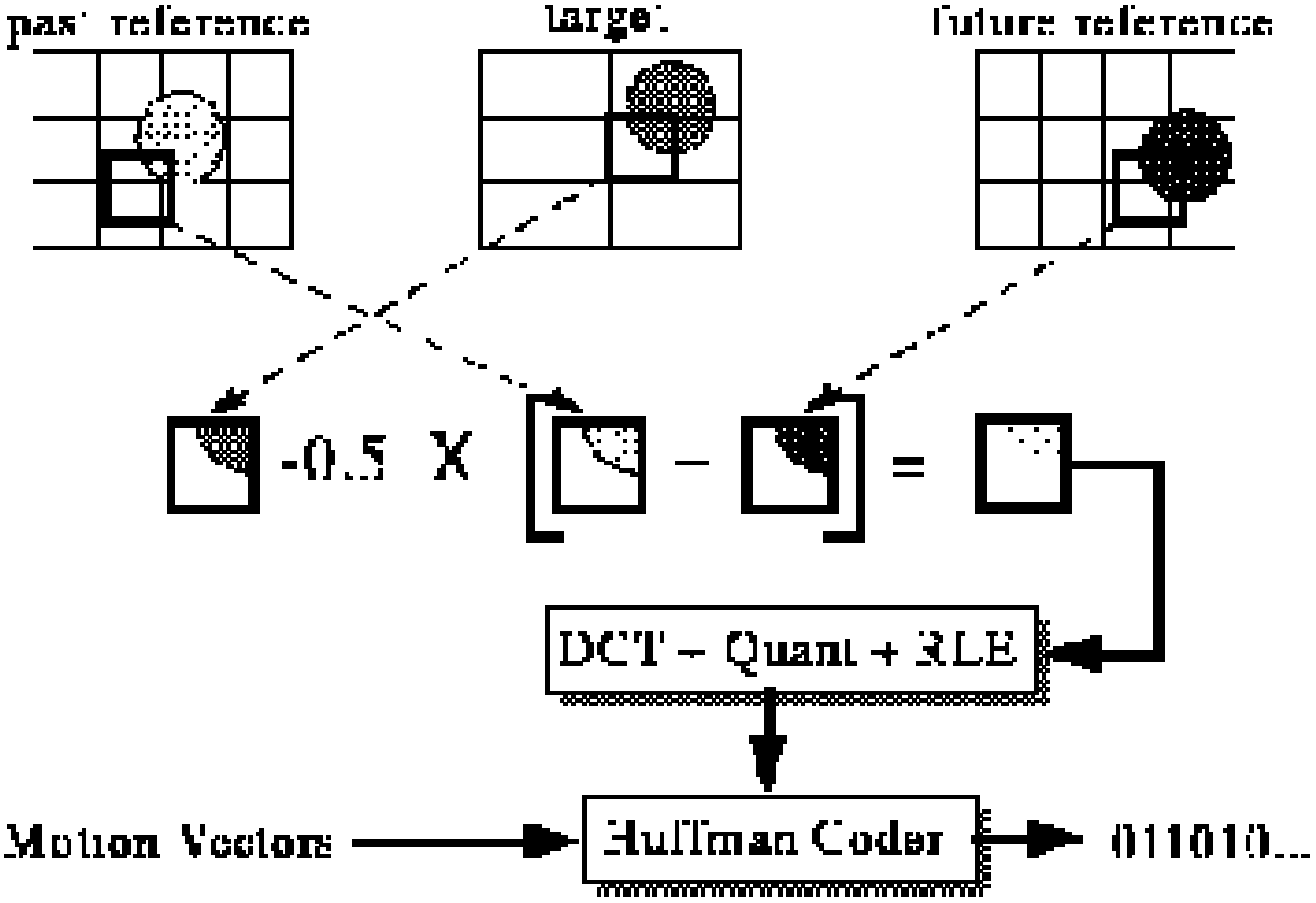
La norme MPEG

- **Norme reconnue par l'ISO (International Standard Organization) établie à fin des années 80 par le Moving Picture Experts Group.**
- **Combine un ensemble de techniques de compression pour optimiser le volume final de données**

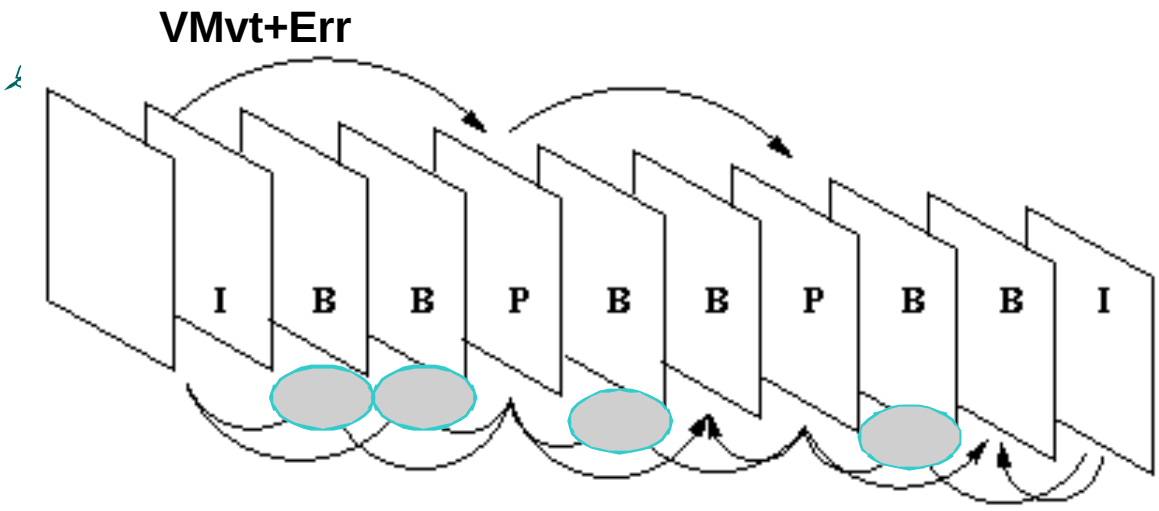


A noter que les calculs sont faits sur la base de blocs de 8x8 pixels

Mise en œuvre de la compression bi directionnelle

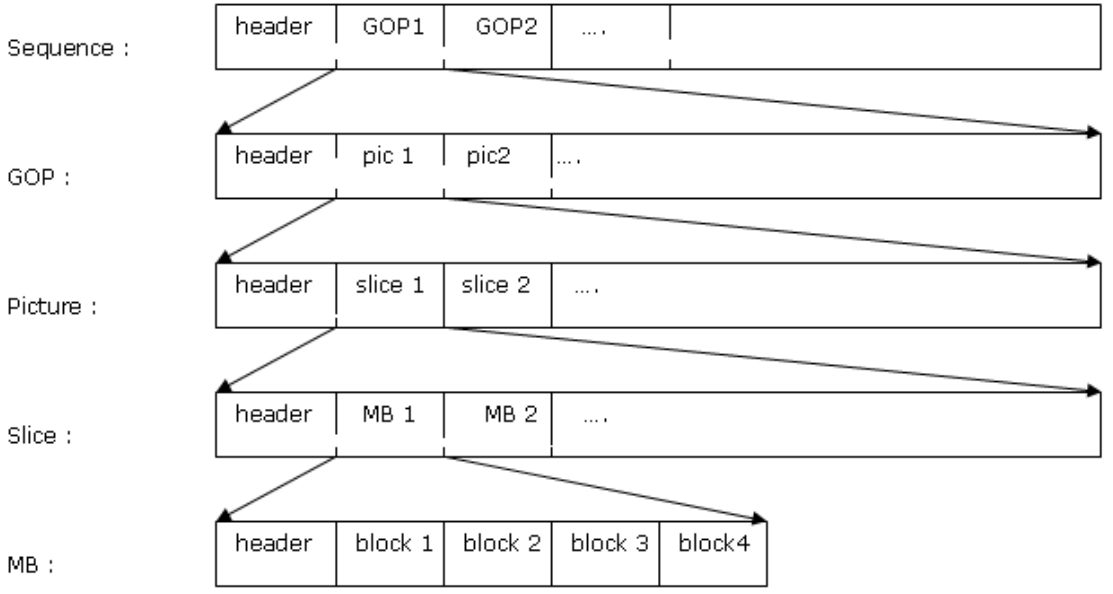


Séquence GOP : Group of Pictures



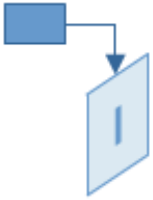
B= image bidirectionnelle ou interpolée

Type Image	Taille	Compression°
I	**18 ko**	**7:1**
P	**6 ko**	**20:1**
B	**2,5 ko**	**50:1**
Moyenne	***4,8 ko***	***27:1***



Liste d’algorithmes de compression vidéo

Image INTRA | Image PREDITE | Image BIDIRECTIONNELLE | Hierarchie MPEG



Evolution de la norme MPEG

- **MPEG 2 :**
  - Gestion des images entrelacées (MPEG 1 limité à video numérique sur CD)
  - Sous échantillonnage UV (limité à 4:2:0) étendu a 4.2.2 & 4.4.4
  - A intégré les objectifs de la version 3 pour la télévision HD
  - Augmentation de la résolution (high level : 1920x1152)
- **MPEG 4**
  - Approche orienté objet pour la description de scènes *Visual Object* (audio : voix, synthese – video : meuble, ecran, conférencier…), localisé relativement entre eux et dans le temps (ouverture vers XML, SMIL), pouvant se superposer…)
  - Ouvre de large possibilité : adaptabilité (du débit, du nombre d'images par secondes, …), mélange de contenu (image, son, objet 3D,…)
  - Interactivité: possibilité de manipuler chaque objet
- **MPEG 7**
  - Poursuite de l'évolution de MPEG 4 avec l'ajout de descripteur; d'un langage de description… (ouverture sémantique)

Liste d'algorithmes de compression vidéo

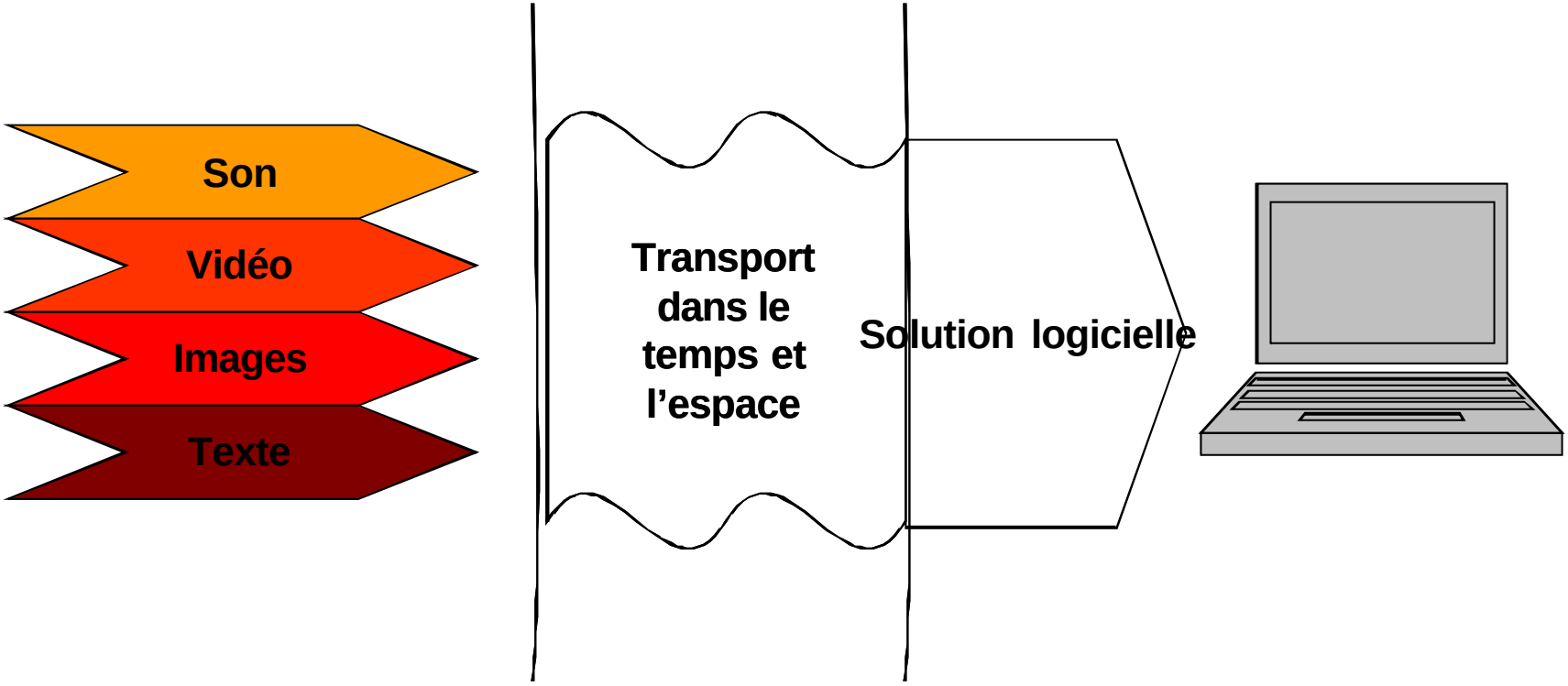
Algorithmes	Taux moyen de compress°	Description
Intel RTV	3:1	Sous échantillonnage des couleurs,
Motion JPG	10:1	DCT…
Fractals	10:1	Efficace pour les images « naturelles » ie bcp d'information… couteux en terme de calcul
Ondelettes (*Wavelets*)	20:1	Compression vidéo
H261/H263	50:1	Basé sur DCT ->
MPEG	30:1	Basé sur DCT (cf JPG) Couteux en compression mais efficace en décompression

2. Programmation multimédia

- Principe et enjeux technologique…

Gérer du contenu multimedia ?

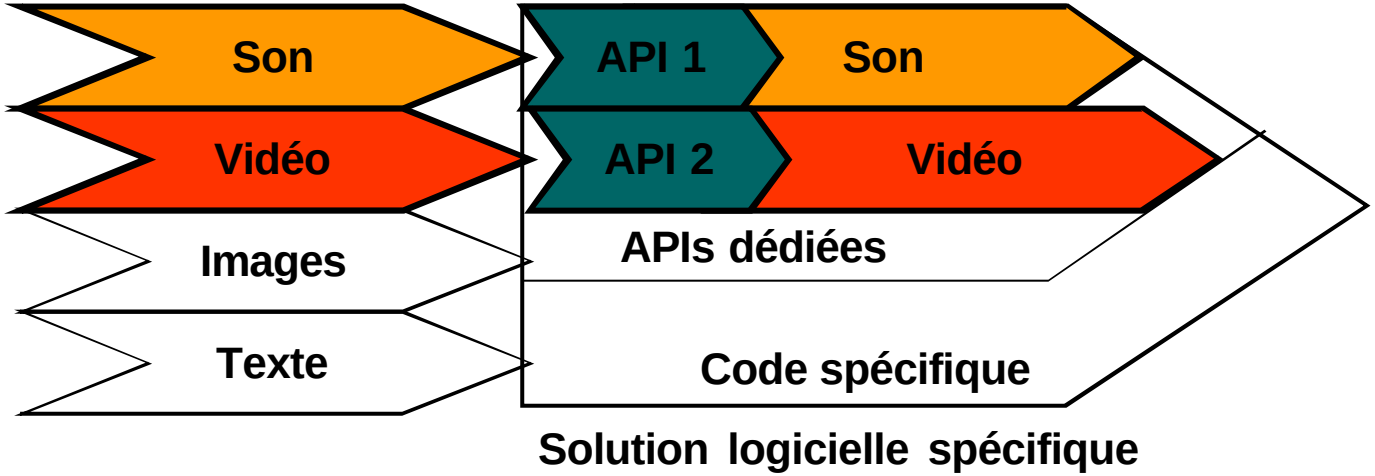
- **= gérer différent type de contenu…**



- **Cad développer une enveloppe logicielle capable d'assurer cette gestion**

Solution 1 : Tabula rasa

développement d'une solution logicielle particulière pour un problème particulier



Avantage:

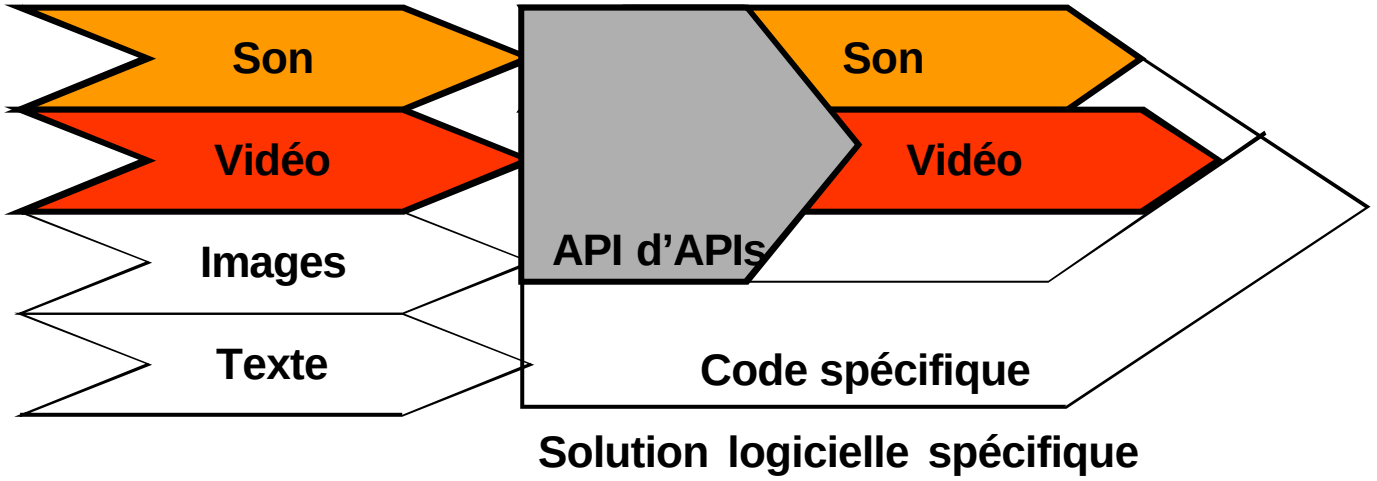
- utilisation des APIs limitée aux besoins exprimés

Inconvénient :

- Pas de réutilisation d'un projet à un autre (hormis l'expérience)
- dépendant des évolutions technologiques (mais c'est souvent le cas en informatique)

Solution 2 : one for all

- **utilisation d'environnement intégrés, ie solution 1 généralisée… utilisation d'une API faisant l'interface avec des APIs spécifiques. Ex. JMF**

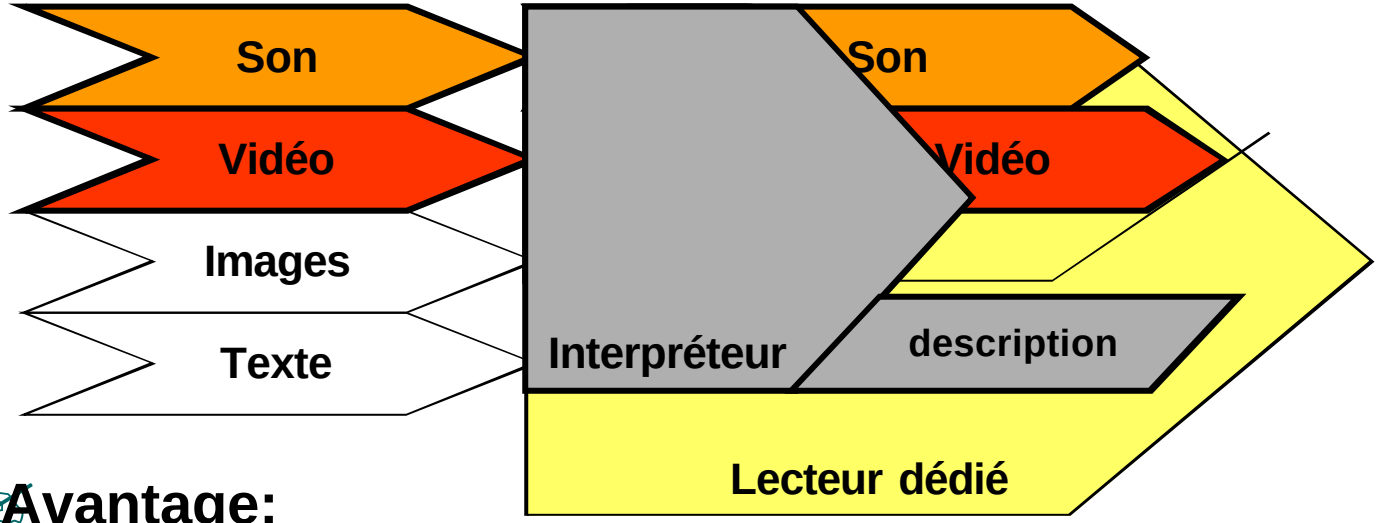


Solution logicielle spécifique

- **Avantage:**
  - Environnement unique cohérent … 1 seul «interlocuteur» entre le contenu multimédia et l'application
- **Inconvénient :**
  - Code, capacité de l'environnement …

Solution 3 : langage de manipulation

- **On ne programme plus la gestion des contenus, mais on décrit comment ces contenus interagissent (synchronisation). Ex: SMIL)**



- **Avantage:**
  - Code réduit !
- **Inconvénient (dans l'exemple de SMIL)**
  - limité au web
  - suppose lecteur adapté

2. « Lire » du son

- Solution ‘de base’ …

Ajouter du son à une application en Java

- Défini au départ comme moyen d'améliorer le contenu du web via les Applets (jdk 1.0).
  - **Aparté:** une applet est une petite application destinée à être télécharger depuis un site pour exécutée sur un poste client via un navigateur web. Elle s'exécute donc depuis une page HTML.
- Utilisation simple car il suffit d'utiliser une classe implémentant l'interface java.Applet.AudioClip, qui propose les méthodes `loop()`, `play()` et `stop()`.
- Ne s'utilise pas directement. On passe par une méthode qui nous renvoie une instance d'une classe implémentant cette interface : Applet.newAudioClip(url).

Apparté sur la définition de chemin en Java

- La manipulation de fichier soulève le problème des formats différents suivant les plateforme (Windows – Mac – Linux : / vers \ etc…).
- Une solution est de contourner le problème en utilisant le protocole FILE: lequel utilise la syntaxe web pour décrire un chemin :
  - **FILE:///C:/toto.txt**
- L'autre solution est d'utiliser les classes File et FileLocator

Apparté sur les propriétés Systèmes

System.getProperty("chaine") permet d'accéder à des propriétés systèmes de la plateforme sur laquelle s'exécute une application Java. "chaine" correspond au nom d'une variable système parmis lesquelles on peut citer :

Clés publiques

Clé	Signification
`"file.separator"`	Caractère de séparation de fichier ('\' '/')
`"java.vendor"`	Vendeur de la JVM
`"java.vendor.url"`	Adresse vendeur
`"java.version"`	Version de la JVM
`"line.separator"`	Séparateur de ligne
`"os.name"`	Nom de l'OS (sic)
`"path.separator"`	Caractère de séparation de chemin.

Clés Protégées

Clé	Signification
`"java.class.path"`	Valeur de la variable d'environnement CLASSPATH
`"java.home"`	Chemin de base de la JVM
`"user.dir"`	Chemin courant de l'utilisateur
`"user.home"`	Racine du compte utilisateur
`"user.name"`	Nom utilisateur

Exemple (tiré du tutorial java) …


```
public class SoundApplication extends Jpanel implements ActionListener, ItemListener {
    SoundList soundList;
    String auFile = "spacemusic.au"; /* …  suite chargement des fichier audio */
    String chosenFile;

    AudioClip onceClip, loopClip;
    URL codeBase;

    JComboBox formats;
    JButton playButton, loopButton, stopButton;
    JLabel status;

    boolean looping = false;

    public SoundApplication() {
        String [] fileTypes = {auFile, aiffFile, midiFile, rmfFile, wavFile};
        formats = new JComboBox(fileTypes);
        formats.setSelectedIndex(0);
        chosenFile = (String)formats.getSelectedItem();
        formats.addItemListener(this);
        playButton = new JButton("Play");  playButton.addActionListener(this);
/* .. Chargement des autres composants graphiques */
        startLoadingSounds();    }

    public void itemStateChanged(ItemEvent e){
        chosenFile = (String)formats.getSelectedItem();
        soundList.startLoading(chosenFile);                }
```

Exemple (tiré du tutorial java) ...


```
void startLoadingSounds() {
        try {
            codeBase = new URL("file:" + System.getProperty("user.dir") + "/");
        } catch (MalformedURLException e) { System.err.println(e.getMessage());  }
        soundList = new SoundList(codeBase);
     /* chargement des autres fichiers */    }

    public void stop() { onceClip.stop();  if (looping) {loopClip.stop();  }    }

    public void start() { if (looping) { loopClip.loop();  }    }

    public void actionPerformed(ActionEvent event) {
        //Bouton PLAY
        Object source = event.getSource();
        if (source == playButton) {
            onceClip = soundList.getClip(chosenFile);
            stopButton.setEnabled(true);
            onceClip.play();     //Play it once.
            status.setText("Playing sound " + chosenFile + ".");
            if (onceClip == null) {
                status.setText("Sound " + chosenFile + " not loaded yet.");   }
            return;
        } //suit le traitement des autres évenements
    }
    public static void main(String s[]) {
       JFrame f = new JFrame("SoundApplication");
        f.addWindowListener(l);
        f.getContentPane().add(new SoundApplication());
        f.setSize(new Dimension(400,100));      f.show();
    }
}
```

Code complet et démo

3. Java Media Framework

- Historique
- Concepts
- Mon premier player

Historique de l'API Java Media Framework

- Lancé par Sun, Silicon Graphics, Intel, IBM en 1996.
  - **`http://java.sun.com/products/java-media/jmf`**
- 1ère spécification 1996, 1ère implémentation février 1997.
- Propose 2 mode :
  - **J2SE Compliant : ie 100% Java**
  - **Performance pack : s'appuie sur des API Natives**

Formats supportés

- Objectif :
  - **Lecture vidéo multiplateforme (PC, Mac, PDA, Téléphone)**
  - **Capture média (vidéo, audio)**
  - **Vidéo Conférence**

- **Protocoles :** FILE (local), HTTP, FTP, RTP

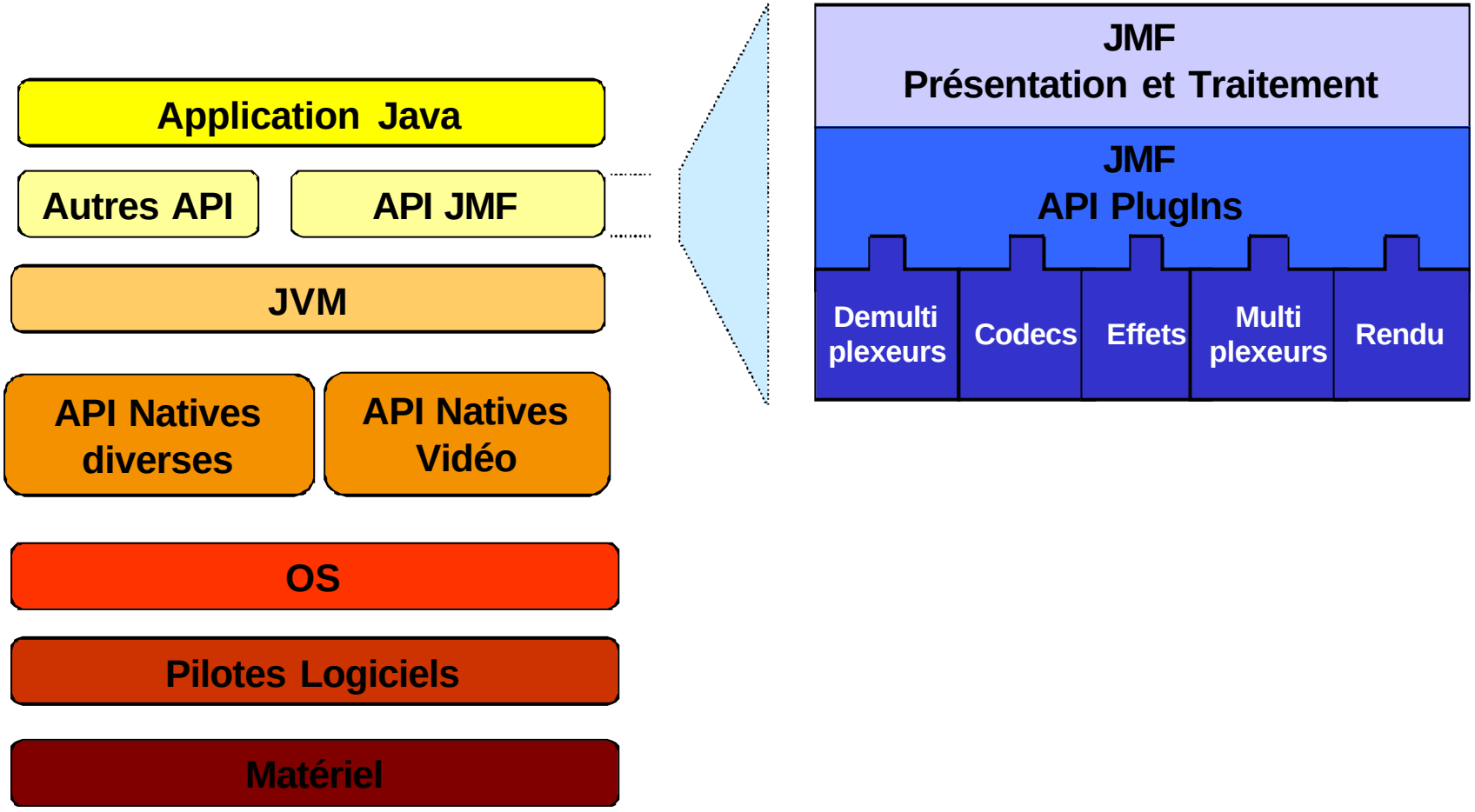
- **Audio : Au, Wav, MIDI, MP3**

- **Vidéo : AVI, MPEG, QuickTime, …**

- **… et ouvert …**

Architecture

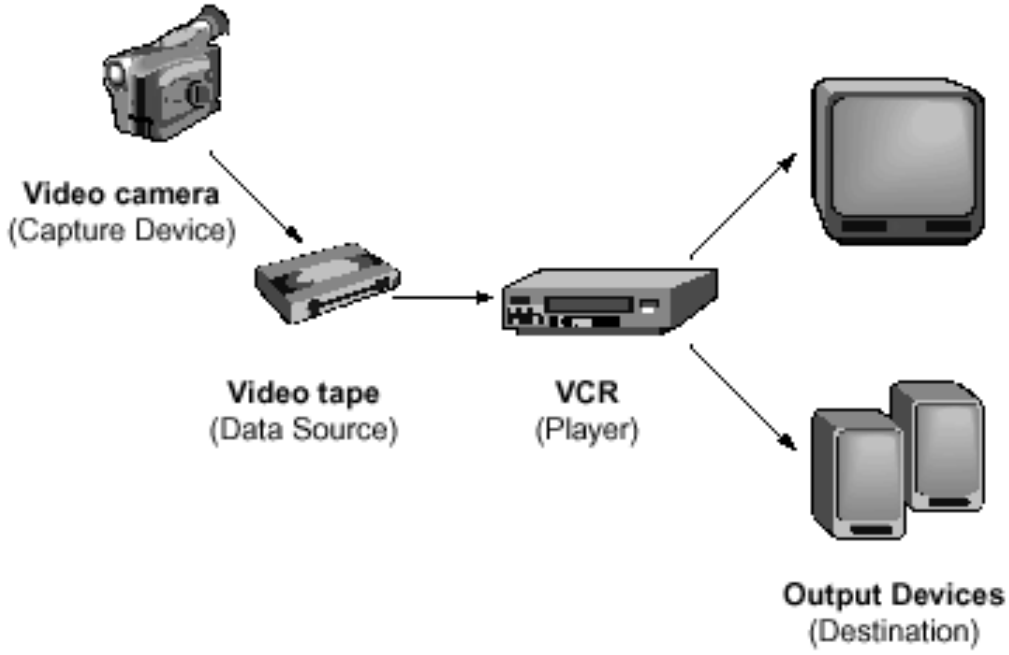
Couches logicielles



Adapté de JFM API Guide

Lecture vidéo

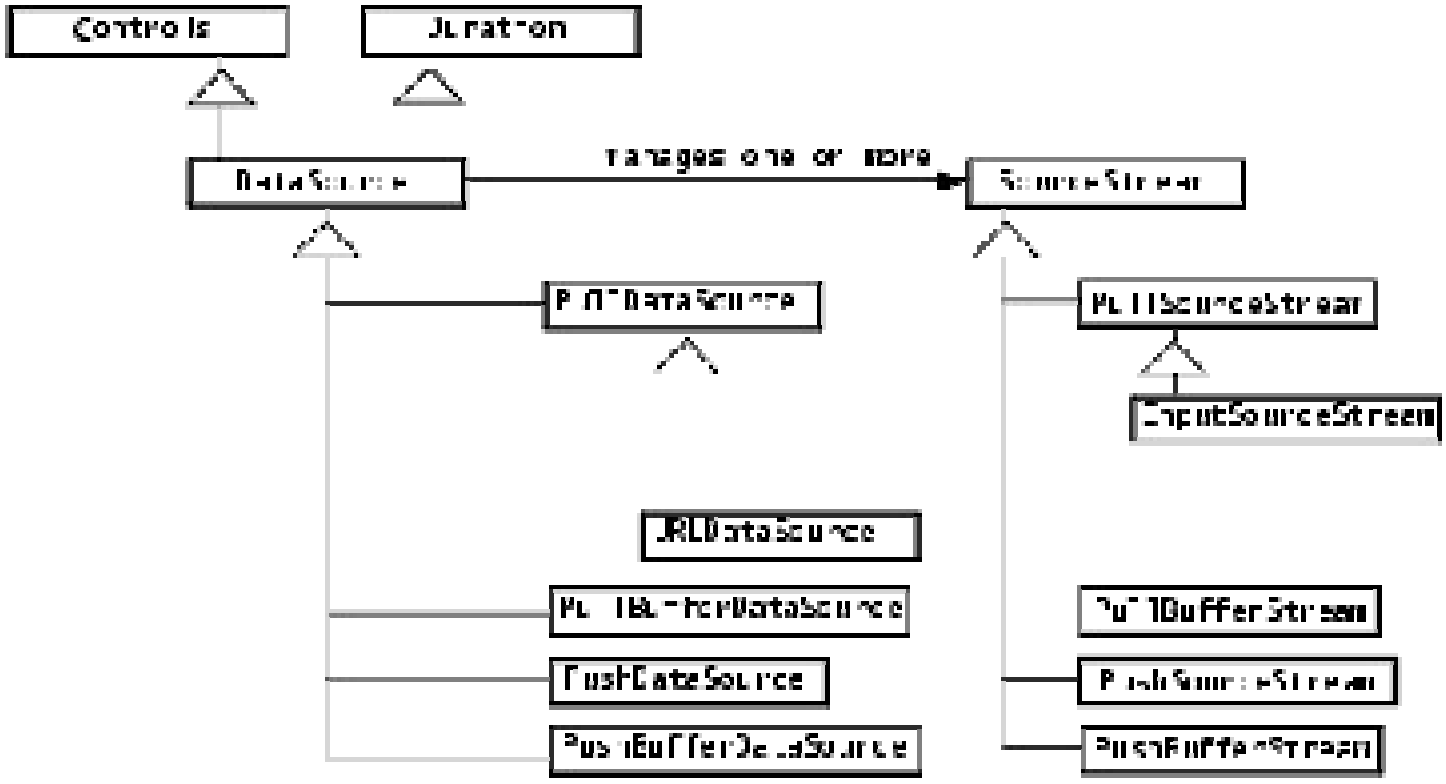
- Concepts JMF:
  - **Player**
  - **Processor**
  - **Data Source**
  - **DataSink**



DataSource

- Source de donnée : cad une classe encapsulant le média que l'on désire manipuler.
- JMF évoque la notion de Time Media, cad dont le contenu évolue au cours du temps, et on considère le média comme un flux.
- 2 grand types de DataSource :
  - Pull (flux tiré) : l'application va chercher le média et décide de ce qu'il veut récupérer et quand. Ex: fichier vidéo ou son.
  - Push (ou flux poussé) : les données sont diffusées et c'est à l'application de s'adapter pour pouvoir les gérer (ou non). Ex: microphone branché, Vidéo à la demande (VOD).

DataSource


Controls
Duration
manages one or more
JRLDataSource
PushBufferDataSource
PushDataSource
PushBufferStream
PushSourceStream

Localisation d'une DataSource

- Comment localiser un média :
  - **URL**
  - **MediaLocator**

Format d'une DataSource

=> javax.media.format.**AudioFormat**

- **Constructeur se base sur l'encodage et si désiré sur le taux d'échantillonnage (sampling) …**
- **Exemple d'encodage : DOLBYAC3, MPEG**

=> javax.media.format.**VideoFormat**

- **Exemple d'encodage : INDEO32, MPEG …**

DataSink

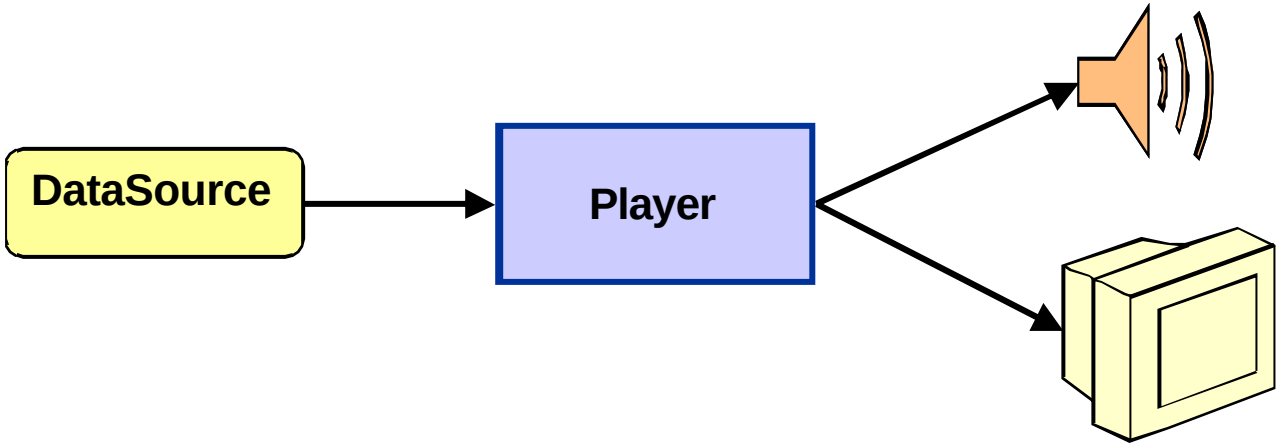
- Sink = Evier: cad conteneur destiné à recueillir et stocker un flux.
- Application directe : sauvegarde du flux dans un fichier.

Gestionnaires JMF

- Ensemble de classe *service* destinées à fournir les objets capables de manipuler les flux multimedia (Player, Processor …)
  - Manager : classe permettant la création de Player, Processors, DataSource et DataSinks. => Pattern Factory.
  - PackageManager : maintien une liste de paquetages JMF contenant des classes utilitaires (ex. implémentation d'un Player particulier).
  - CaptureDeviceManager : maintient une liste des périphériques de capture (ex: Audio : micro + couche logiciel).
  - PlugInManager : maintien une liste de plugin JMF ex. (De)MultiPlexer, Codecs, …

Player : principe

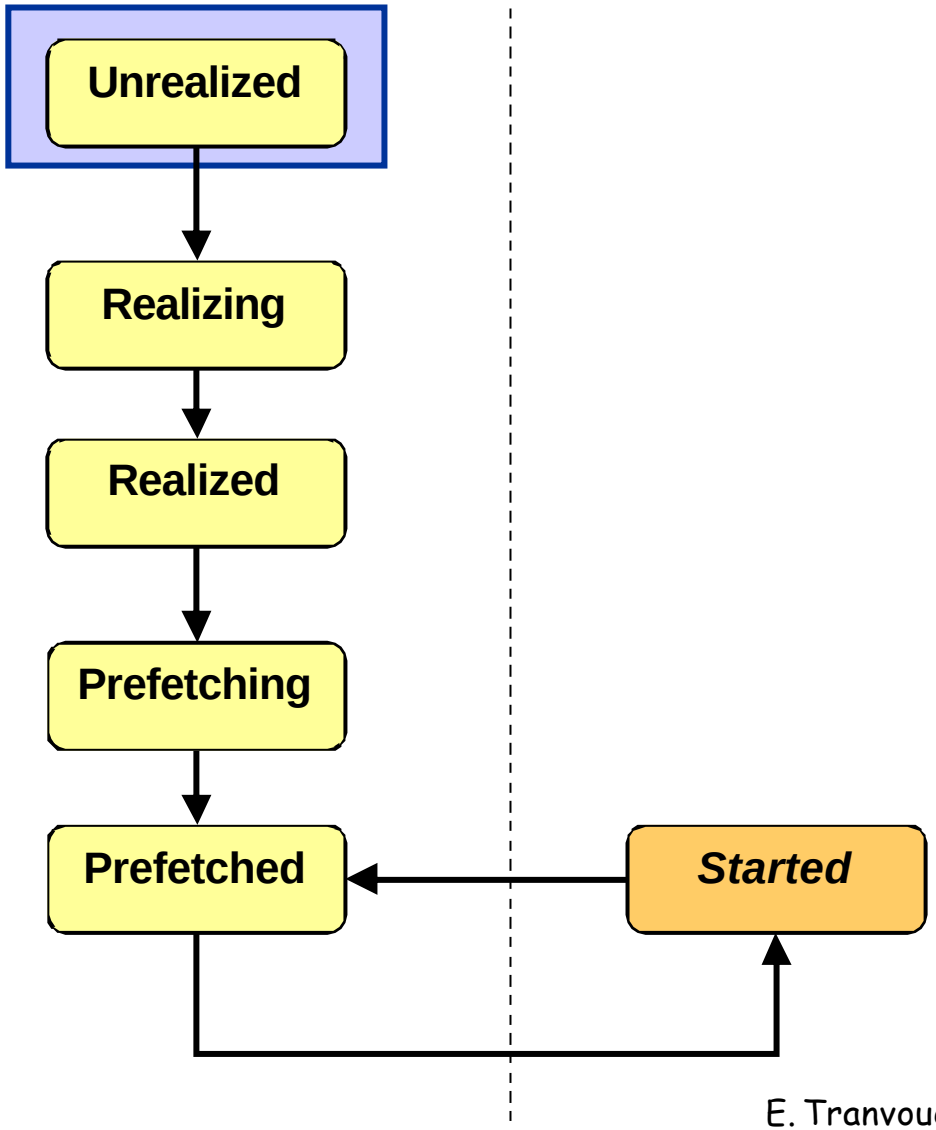
- Un player effectue un traitement sur une source de donnée et en effectue une présentation.
- Si on veut effectuer des modification de la source il faut passer par autre chose…
- Dispose de contrôles de base sur la source (lecture …).
- Avant de « jouer » le flux audio ou vidéo passé en paramètre il passe par différentes phases.



Player : les différents états - unrealized

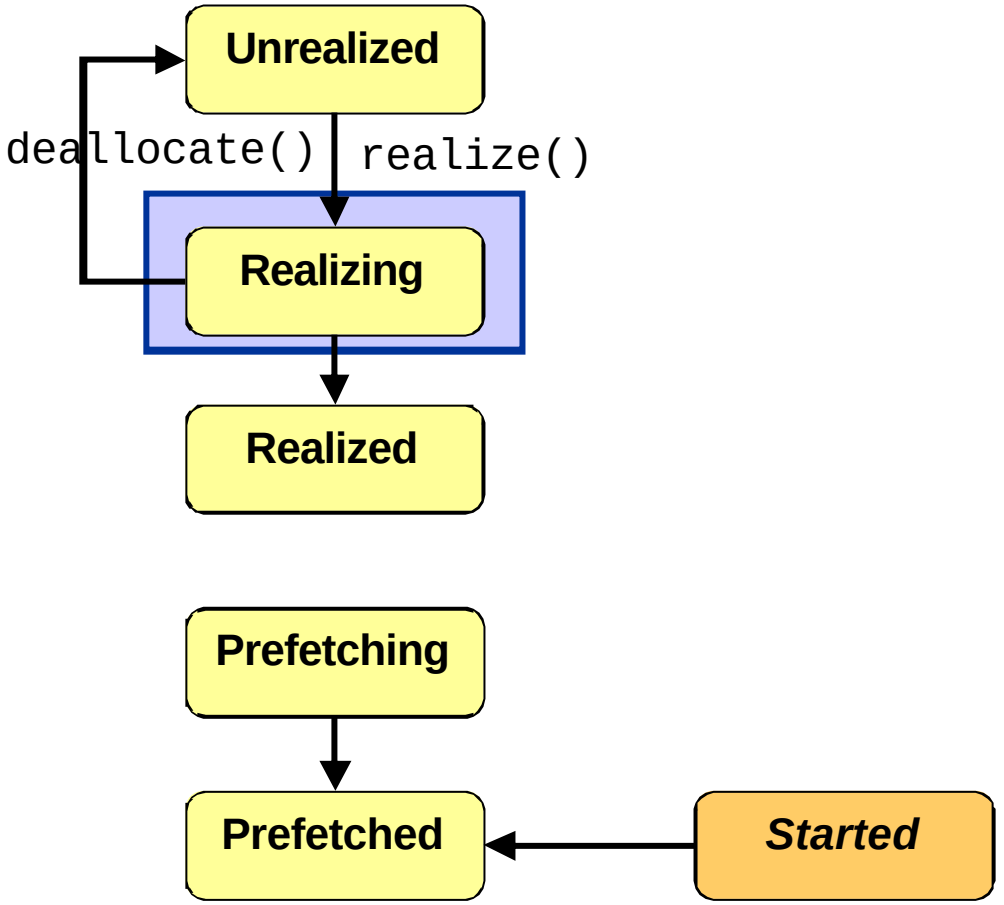
- Unrealized :

État initial du player (nouveau né).
Ne connaît pas le média à jouer.



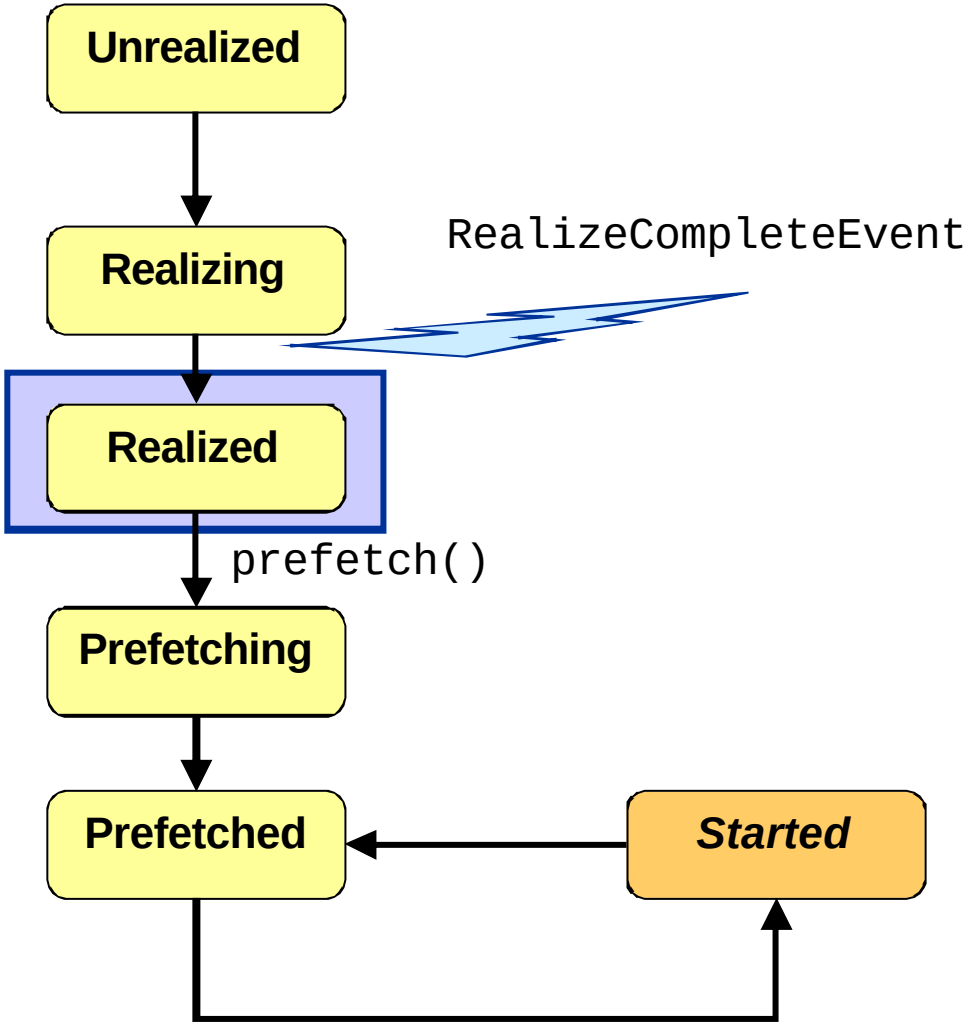
Player : les différents états - realizing

- Realizing : détermine les ressources nécessaires.



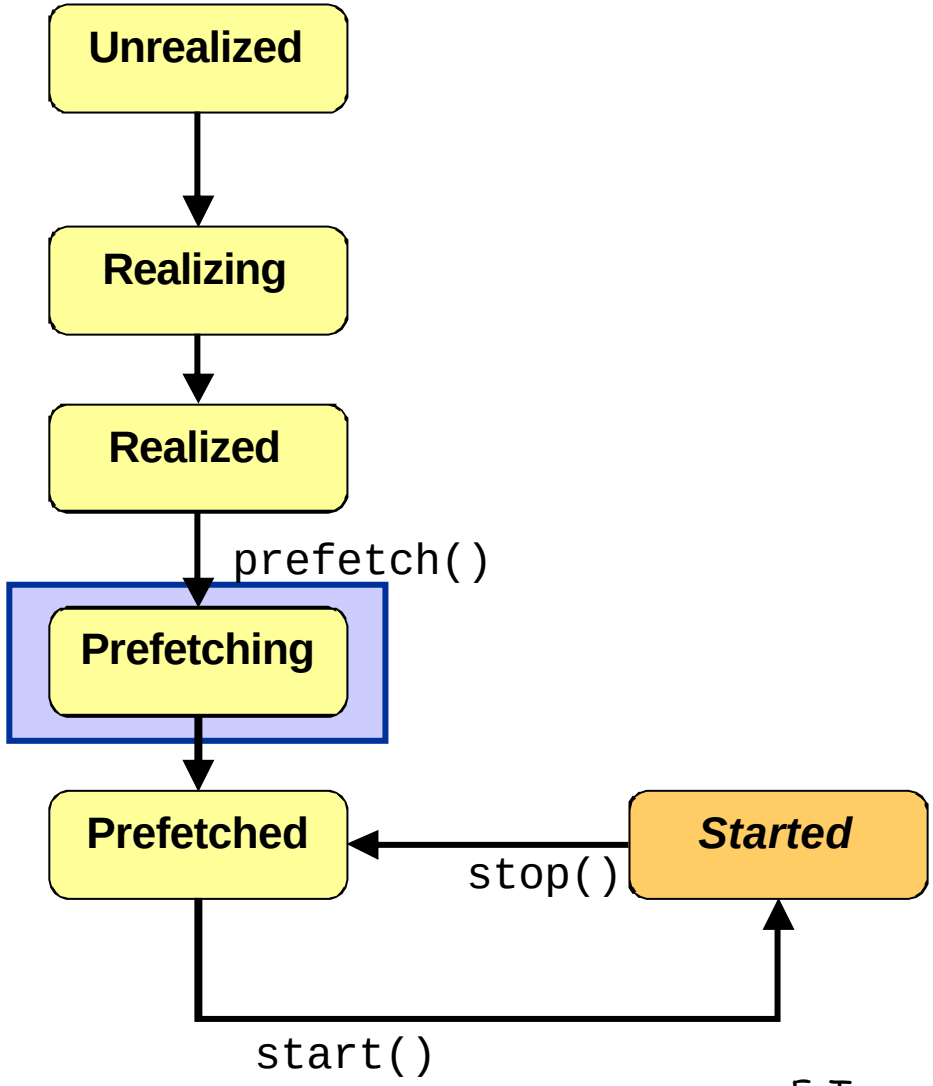
Player : les différents états - realized

- Realized : dispose des informations nécessaires sur le média et les ressources requises … mais ne les a pas réservées.
- Ex: Film non démarré mais « chargé »



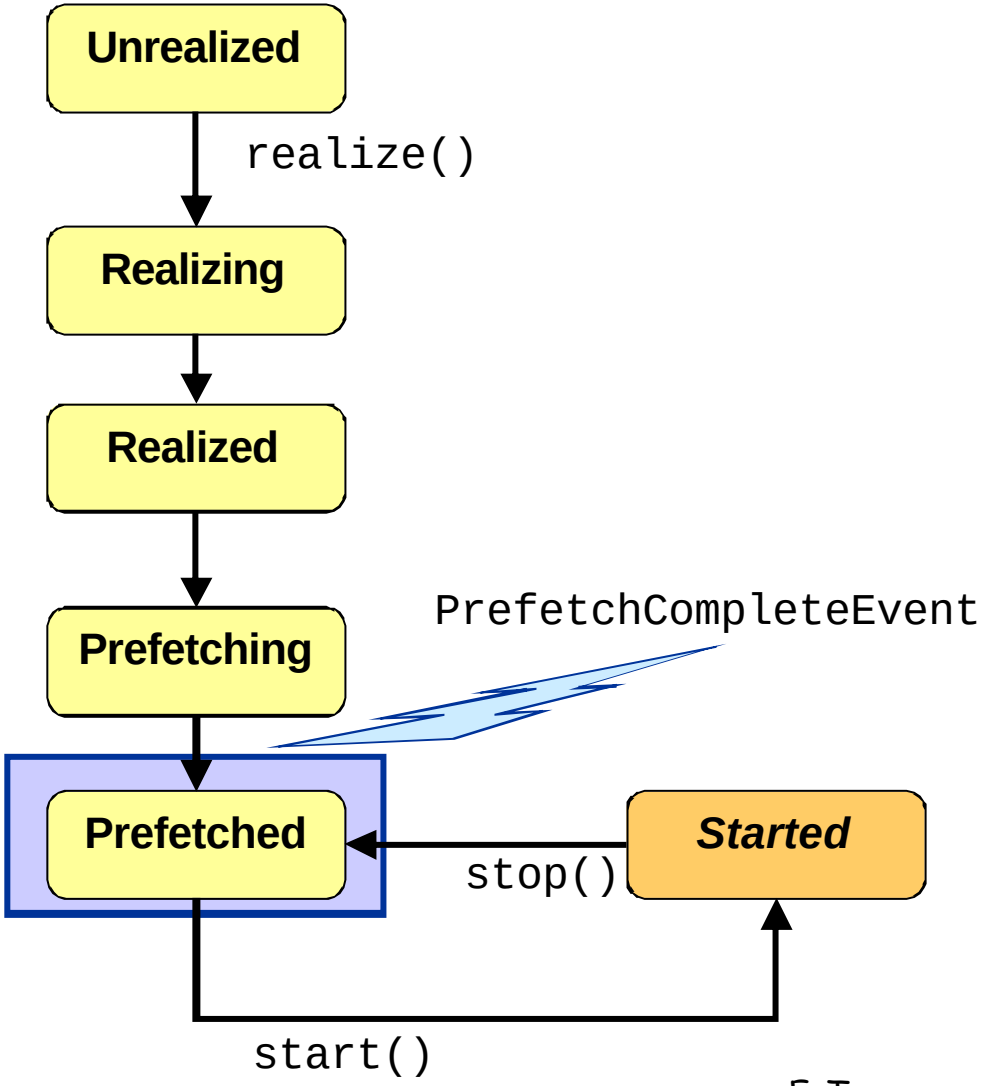
Player : les différents états – prefetching

- Prefetching : prépare la présentation du média :
  - **Réserver les ressources**
  - **Charge le media (ex. film)**



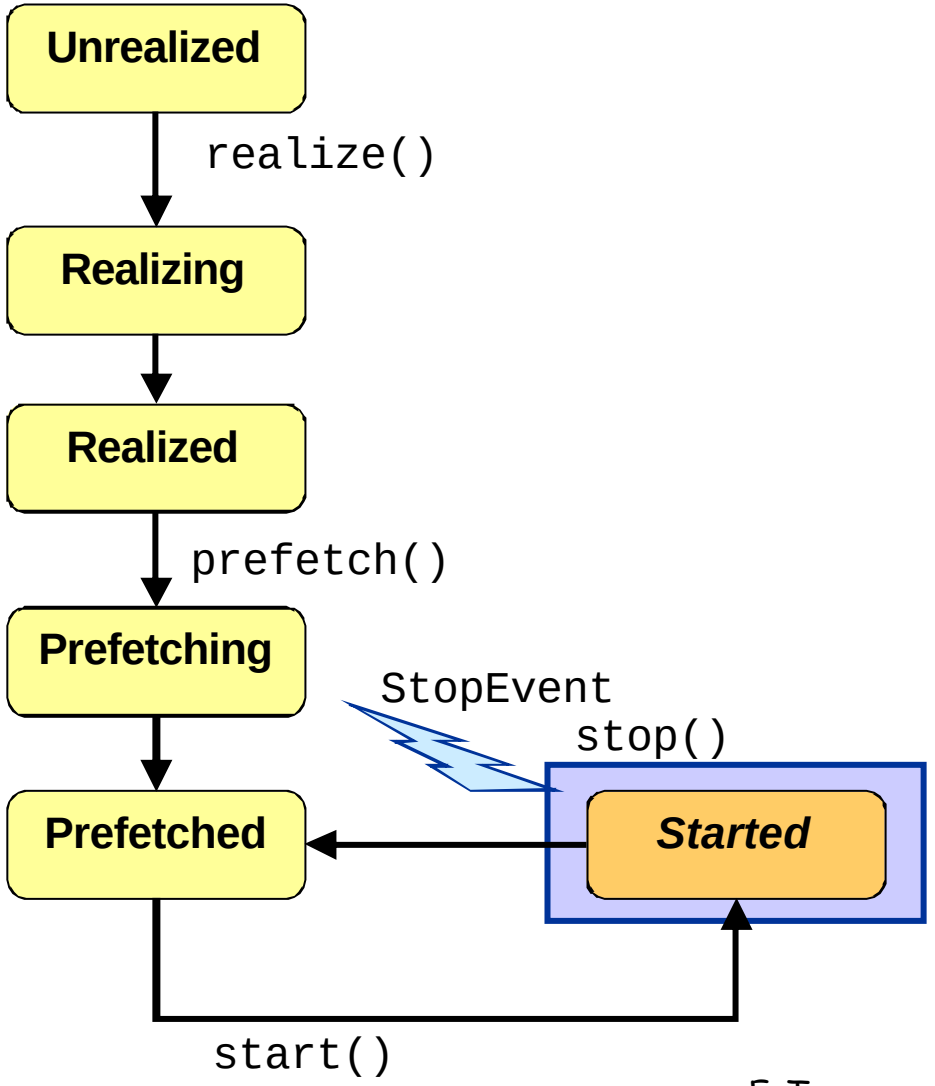
Player : les différents états - prefetched

- Prefetched : prêt à présenter le média



Player : les différents états - started

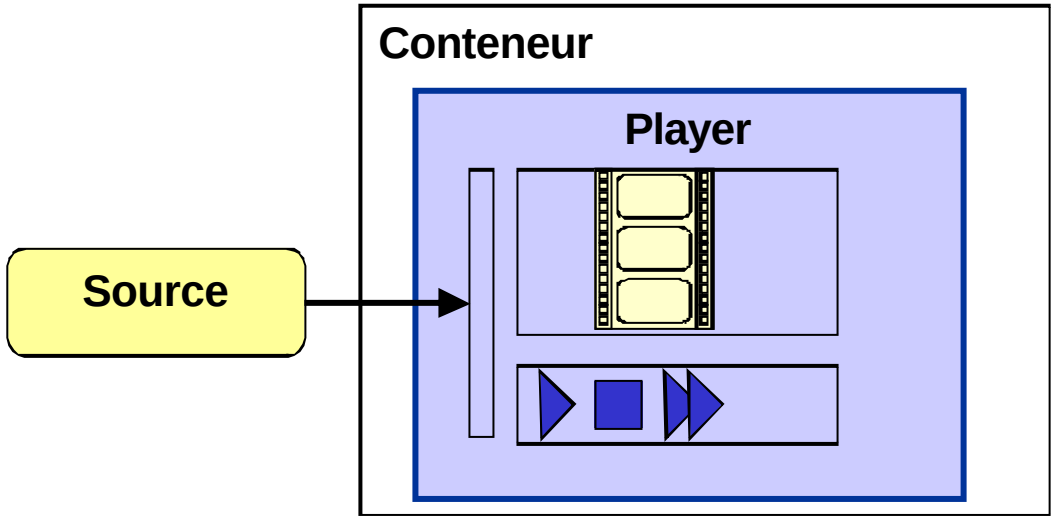
- Started : lecture du média en cours



Player : utilisation

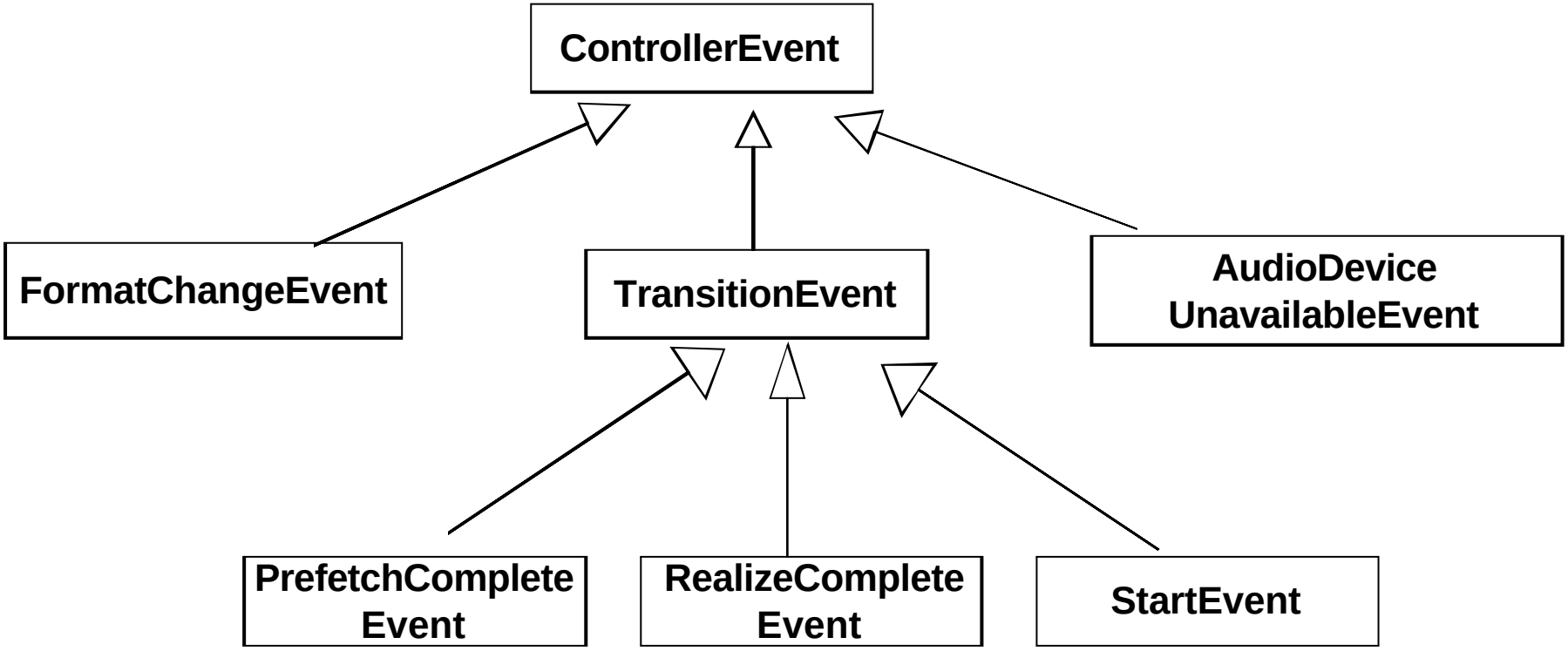
- N'est pas crée directement : on passe par un Manager
  - Manager.createPlayer(arg). Arg doit indiquer un flux.
  - Un fois crée, le player passe par les phases indiquées pcdemment. Il n'est pas tout de suite utilisable…
  - createRealizedPlayer() créé un player mais ne rend la main que lorsqu'il est réalisé (fonction bloquante).

! Bien mesurer les implications.



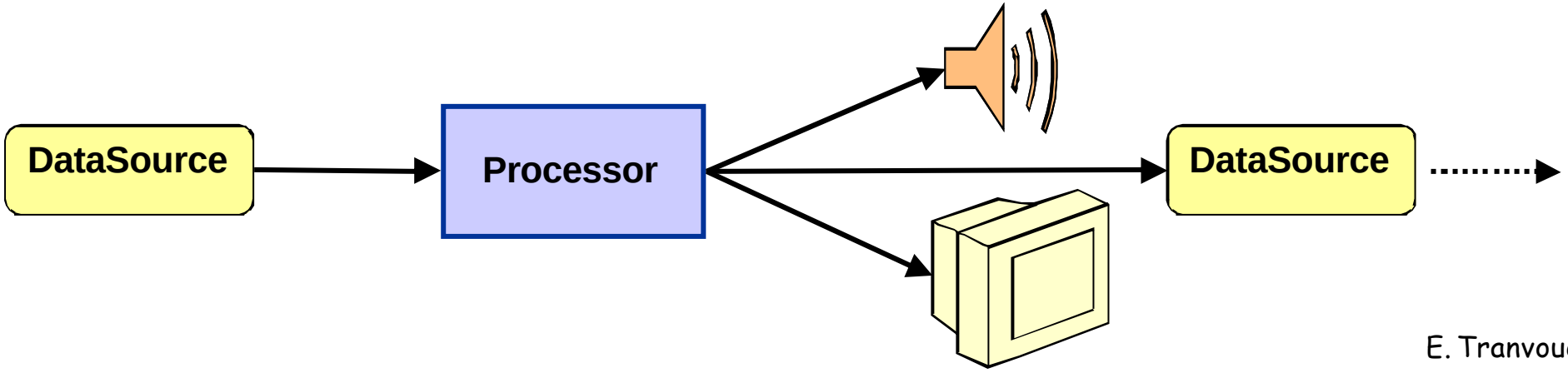
Player : l'interface ControllerListener

- Utilisée pour gérer les événements du player.
- Se limite à une méthode :
  - **controllerUpdate(ControllerEvent e) avec e pouvant instancier les sous classes:**



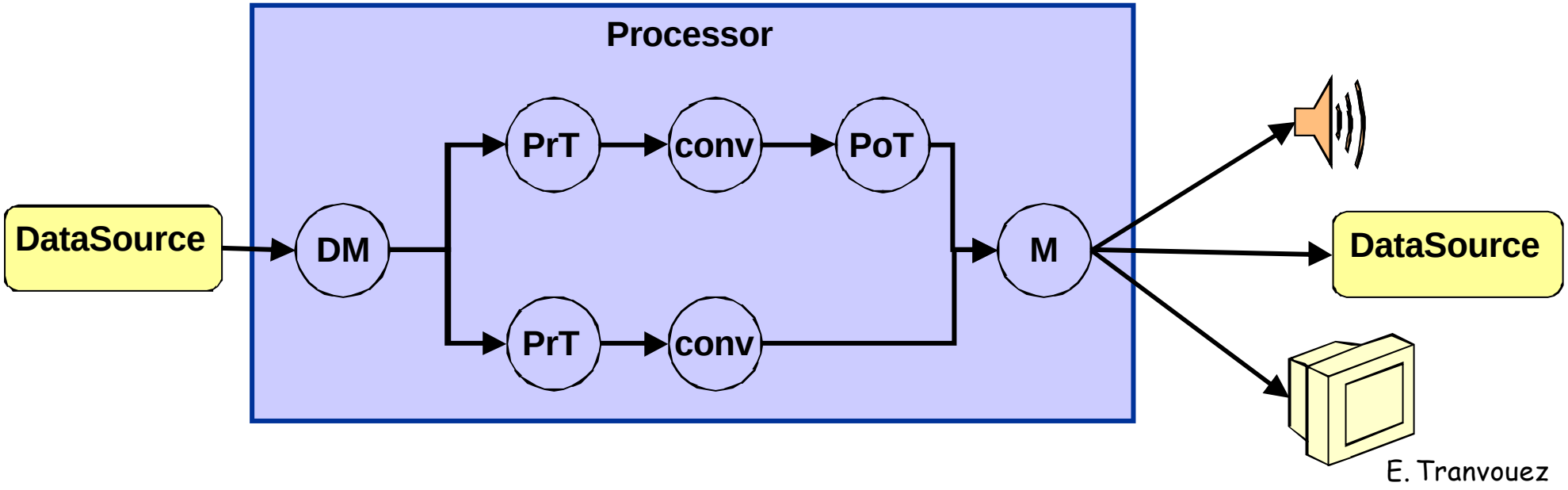
Processor : principe

- C'est un player évolué permettant un plus grand contrôle sur la source.
- Si on veut effectuer des modification de la source il faut passer par autre chose…
- Dispose de contrôles de base sur la source (lecture …).
- Avant de « jouer » le flux audio ou vidéo passé en paramètre il passe par différentes phases.



Processor : zoom

- [D/M] (De)multiplexage : (dé)composer un flux en plusieurs flux (ex: audio/vidéo). Action automatique.
- [PrT/PoT] Pré/Post Traitement (algorithme) appliqué sur le flux entrant ou le flux décodé.
- conv : transcodage vers un autre format de média ex: (dé)compression.



Processor : suite

États supplémentaires ./. Player :

- **Configuring :** connection sur la source, demultiplexage et identification format de la source.
- **Configured :** activé par un ConfiguredCompleteEvent. Attend le prochain appel de fonction realize().

Illustration d'un player simple

- Le bloc try – catch - finally

```
import java.awt.*;   import javax.swing.*;
import java.net.URL; import java.io.*;
import javax.media.*;

import javax.media.ControllerEvent;
import javax.media.ControllerListener;

public class FirstVideoReader extends JFrame
   implements ControllerListener {
   Player player;
   String fichier;
   Container contentPane;

   public synchronized void controllerUpdate(ControllerEvent   event) {
       if (event instanceof StartEvent) {
           Component comVisual = player.getVisualComponent();
           Component comControl = player.getControlPanelComponent();
           if (comVisual != null)
                contentPane.add(comVisual, BorderLayout.CENTER);
           if (comControl != null)
                this.contentPane.add(comControl, BorderLayout.SOUTH);
   validate();
   pack();
   setVisible(true);
          }
     }
```

Récupère des composants par défauts chargés d'afficher et contrôler le player. comVisual générera par exemple un StartEvent pour signaler l'appui sur la touche lecture …

Code


```
  public FirstVideoReader(String str){
    fichier = str;
    contentPane = getContentPane();
    contentPane.setLayout(new BorderLayout());
    URL url = null;
    try{
       url=new URL(fichier);
      player = Manager.createPlayer(url);
    }catch(Exception e){  System.out.println(e);  }
    player.addControllerListener(this);
    player.start();
  }

  public static void main(String[] args) {
 String fichier;
 if(args.length==0){
              fichier=« o.mpg";
 }else fichier = args[0];

          new FirstVideoReader(fichier);
  }
}
```

Appel a la méthode statique pour récupérer un player par défaut à partir d'une adresse …

La classe implémente l'interface ControllerListener peut donc s'autogérer…

Lancement du lecteur…

Exécution du player simple

Lecture de flux sur le réseau

- Ne change rien au principe …
- … puisqu'on passe par une URL pour spécifier la localisation du fichier.
- Concernant les flux dit *streamé* nécessite d'utiliser un autre protocole -> RTP …

4. JMF : Capture de média

- Principe
- Mon premier enregistreur

Principe général

- **Ne remet pas en cause les principes énoncés pcdt.**
- **Principe supplémentaire :**
  1. Localiser le périphérique de capture désiré en s'adressant au `CaptureManager`. (cf. exemple ci après).
  2. Récupérer une instance de `CaptureDeviceInfo` pour le périphérique (`cdi`).
  3. Récupérer un `MediaLocator` du `cdi` et créer une `DataSource`
  4. Créer un `Player` ou `Processor` pour utiliser le DataSource
  5. Démarrer le `Player` ou le `Processor` pour lancer la capture.

- **On retrouve des notions familières …**

Localisation d'un périphérique

2 solutions :

- méthode statique `CaptureDeviceManager.getDeviceList()` qui renvoie un `Vector` contenant des instances de `CaptureDeviceInfo`.
- appel 'nominatif' au périphérique via `CaptureDeviceManager.getDevice("nomPeriph")`

Exemple de périphérique :

- `DirectSoundCapture`
- `JavaSoundCapture`

Exemple de localisation d'un périphérique


```
import javax.media.*;
import java.util.Vector;

public class ListePeriph{
   public static void main(String s[]) {
     Vector v =CaptureDeviceManager.getDeviceList(null);
     for (int i=0;i<v.size();i++)
        System.out.println("Liste des périphériques "+i+" : " +
            ( CaptureDeviceInfo) v.get(i) + ".");
     CaptureDeviceInfo cdi =
  CaptureDeviceManager.getDevice("DirectSoundCapture");
     if(  cdi == null)
        System.out.println("Erreur, pas trouvé");
     else
        System.out.println("Ok pour "+cdi.getName());

  }

}
```

Enregistrement des données capturées

Implique d'utiliser un Processor :

1. A partir du `Processor` récupérer un `DataSource`.
2. Créer un flux d'écriture « pointant » sur un DataSink lui-même créé à partir d'un Manager : `Manager.createDataSink()`
3. Demander au `DataSink` d'ouvrir le fichier
4. Démarrer le `DataSink` (`start`)
5. Démarrer le `Processor`
6. Attendre un évènement de fin (`EndOfMediaEvent` ou evénement provenant de l'interface graphique).
7. Arreter le `Processor`
8. Fermer le `Processor`
9. Fermer le `DataSink` une fois l'évènement `EndOfStreamEvent` est arrivé

Illustration : création de DataSource


```
MediaLocator videoMediaLocator = captureVideoDevice.getLocator();
DataSource videoDataSource = null;
try   {
   videoDataSource =
   javax.media.Manager.createDataSource(videoMediaLocator);
   }catch (IOException ie) {Stdout.logAndAbortException(ie); }
   catch (NoDataSourceException nse) { Stdout.logAndAbortException(nse);
}

if (! DeviceInfo.setFormat(videoDataSource, captureVideoFormat))
{  Stdout.log("Error: unable to set video format  - program aborted")
   System.exit(0);
}   /….  /

DataSource mixedDataSource = null;
try{
   DataSource dArray[] = new DataSource[2];
   dArray[0] = videoDataSource;
   dArray[1] = audioDataSource;
   mixedDataSource = javax.media.Manager.createMergingDataSource(dArray);

}catch (IncompatibleSourceException ise) {
Stdout.logAndAbortException(ise); }
```

Illustration : création du processor


```
FileTypeDescriptor outputType = new
FileTypeDescriptor(FileTypeDescriptor.MSVIDEO);

Format outputFormat[] = new Format[2];
outputFormat[0] = new VideoFormat(VideoFormat.INDEO50);
outputFormat[1] = new AudioFormat(AudioFormat.GSM_MS);

      // create processor
ProcessorModel processorModel = new ProcessorModel(mixedDataSource,
outputFormat, outputType);
Processor processor = null;
try
{
   processor = Manager.createRealizedProcessor(processorModel);
   }
catch (IOException e) { Stdout.logAndAbortException(e); }
catch (NoProcessorException e) { Stdout.logAndAbortException(e); }
catch (CannotRealizeException e) { Stdout.logAndAbortException(e); }
```

Illustration : création du dataSink


```
DataSource source = processor.getDataOutput();

MediaLocator dest = new MediaLocator("file:testcam.avi");
DataSink dataSink = null;
MyDataSinkListener dataSinkListener = null;
try
{
   dataSink = Manager.createDataSink(source, dest);
   dataSinkListener = new MyDataSinkListener();
   dataSink.addDataSinkListener(dataSinkListener);
   dataSink.open();
}
   catch (IOException e) { Stdout.logAndAbortException(e); }
   catch (NoDataSinkException e) { Stdout.logAndAbortException(e); }
   catch (SecurityException e) { Stdout.logAndAbortException(e); }
     // now start the datasink and processor
   try
   {
     dataSink.start();
   }
   catch (IOException e) { Stdout.logAndAbortException(e); }
   processor.start();
```

Ressources Multimedia

Bibliographie

- *Technique de base pour le multimédia*, Gérad Weidenfeld et al., Ed. Masson 1997.
- *Java et le multimédia*, Jean-Marc Farinone, Edition 01 Informatique, DUNOD, 2003.
- *Java : Comment programmer.* Deitel & Deitel. Ed. Reynald Goulet. 2002.
- *Multimédia: les fondamentaux*. I. Roxin & D. Mercier. 2004.

Webliographie

- Java.sun.com/jmf
- API de JMF (très bon tutorial & présentation claire de JMF) : *JavaTM Media Framework API Guide*, Sun MicroSystems Inc. 1999. Disp. sur http://java.sun.com/product/java-media/jmf/
- JavaWorld. (généraliste) : http://www.javaworld.com
- http://www.sciences.univ-nantes.fr/info/perso/permanents/pmartin/Video
- http://www.tomshardware.com/1999/09/24/video_guide_part_3/page8.html
- http://www.cs.wustl.edu/~jain/cis788-97/ftp/video_over_atm/index.htm
- https://www.cs.sfu.ca/CC/365/mark/material/notes/Chap4/Chap4.3/Chap4.3.html

Ressources Multimedia

- ***Java et le multimédia*, Jean-Marc Farinone, Edition 01 Informatique, DUNOD, 2003.**
- ***JavaTM Media Framework API Guide*, Sun MicroSystems Inc. 1999. Disp. sur http://java.sun.com/product/java-media/jmf/**
- **JavaWorld. (généraliste) : http://www.javaworld.com**
- **Téléchargements :**
  - API JMF x.xx : java.sun.com
  - Plugin MP3 JMF : java.sun.com