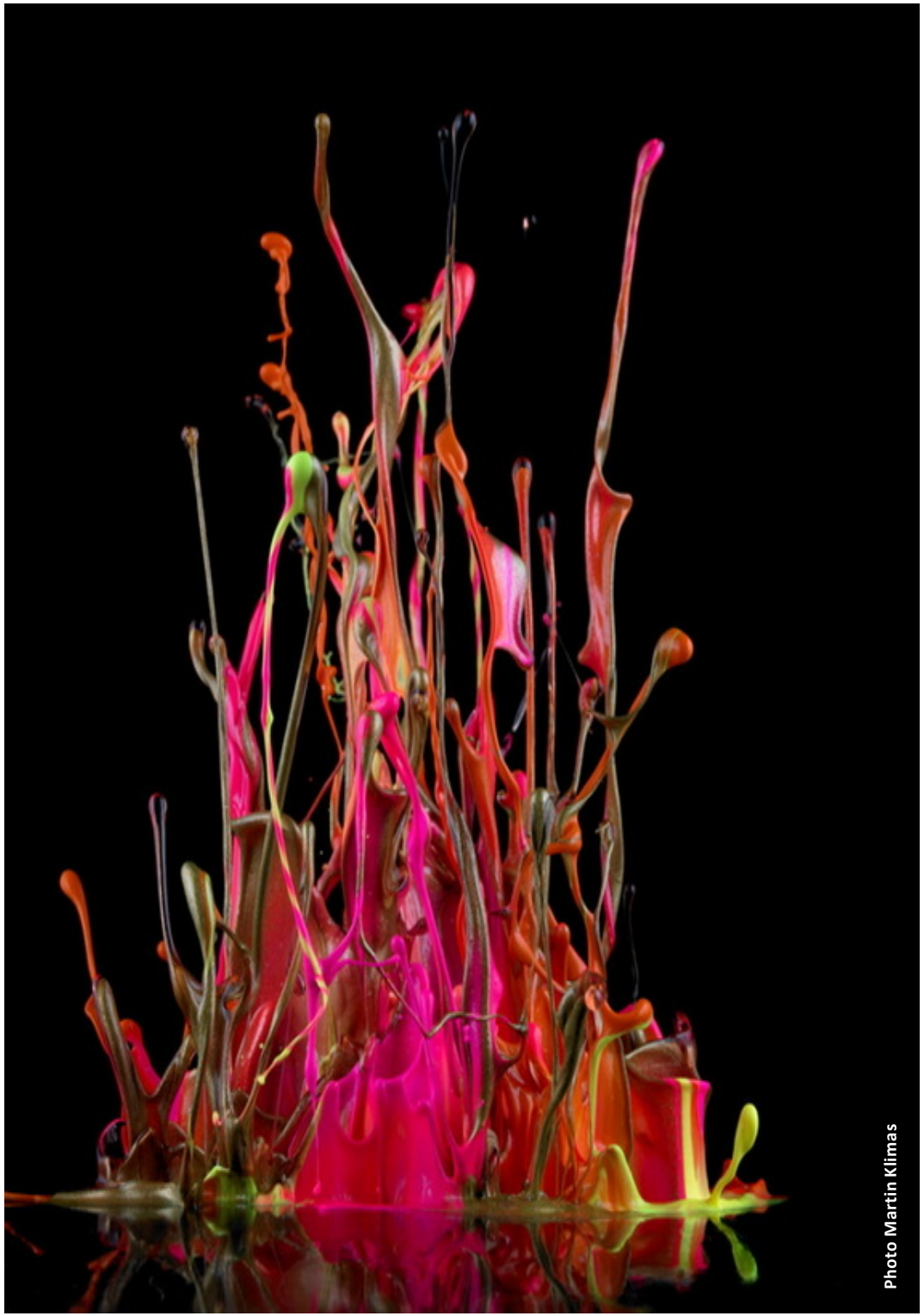


EPFL

Programmation pour Ingénieur

Exo LabVIEW - my Third VI

ME 4e semestre

Rev. 2025.1

Christophe Salzmann

Laboratoire d'Automatique

My Third VI

But:

Tester votre programme C depuis LabVIEW

Etapes:

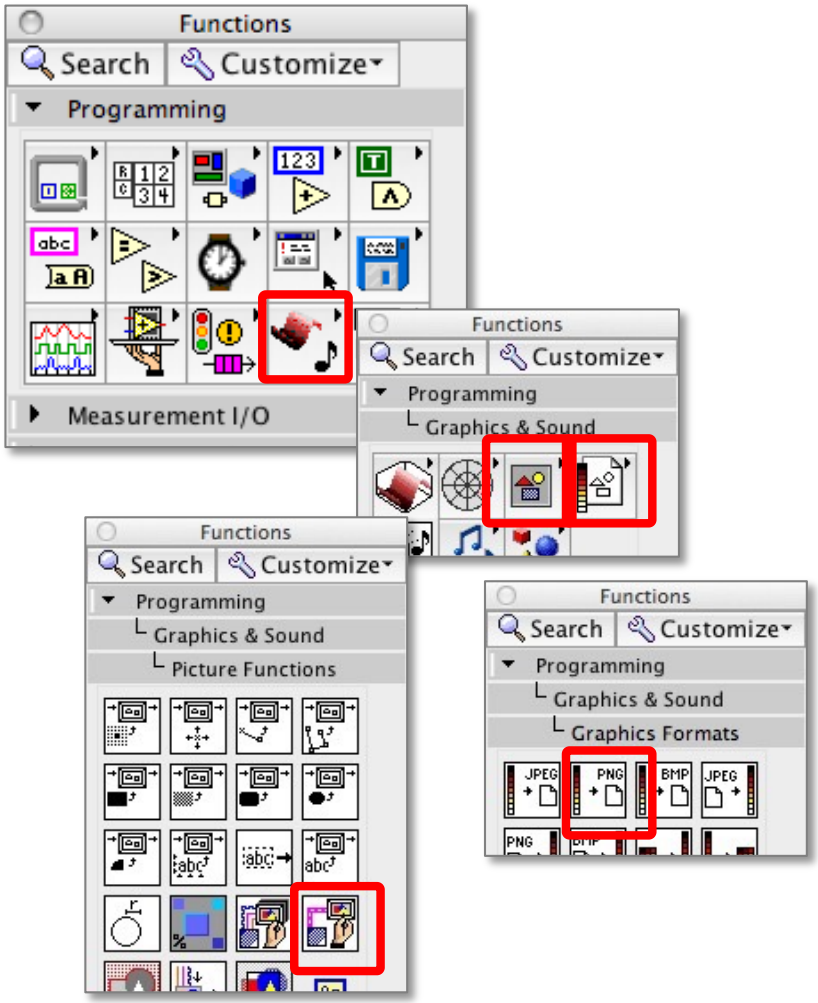
- Lecture du fichier contenant l'image png
- Affichage de l'image
- Accès aux éléments de l'image
- Création du vecteur avec largeur, hauteur et pixels
- Ecriture du vecteur
- Appel de votre code C
- Lecture du fichier **pos.txt**

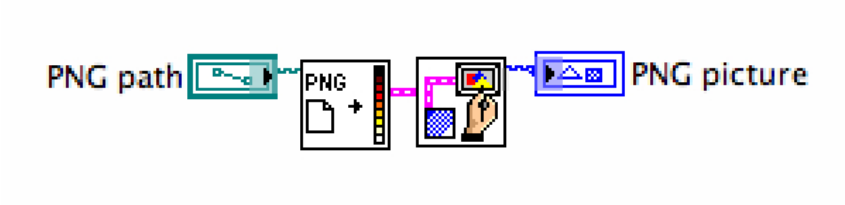
A faire:

- Conversion des 3 composants RGB des pixels en 1 pixel U32

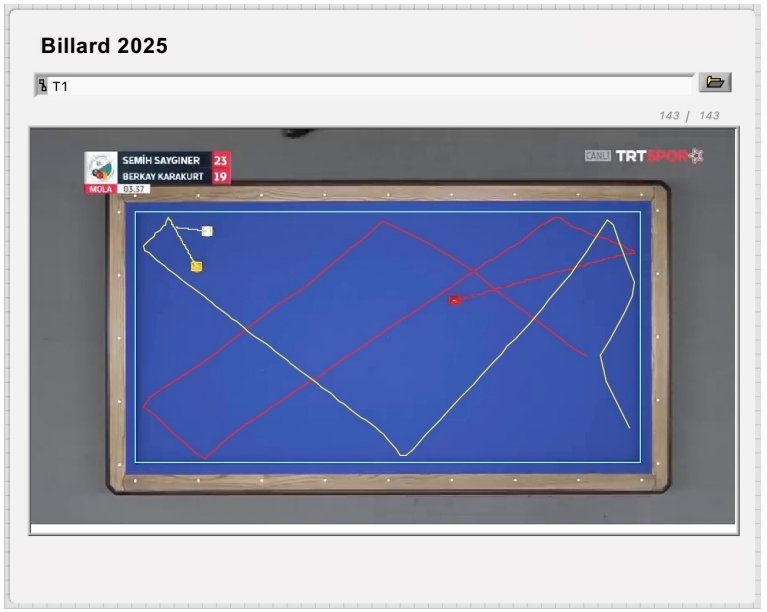
Lecture d'une image png

En seulement 2 Vis!!

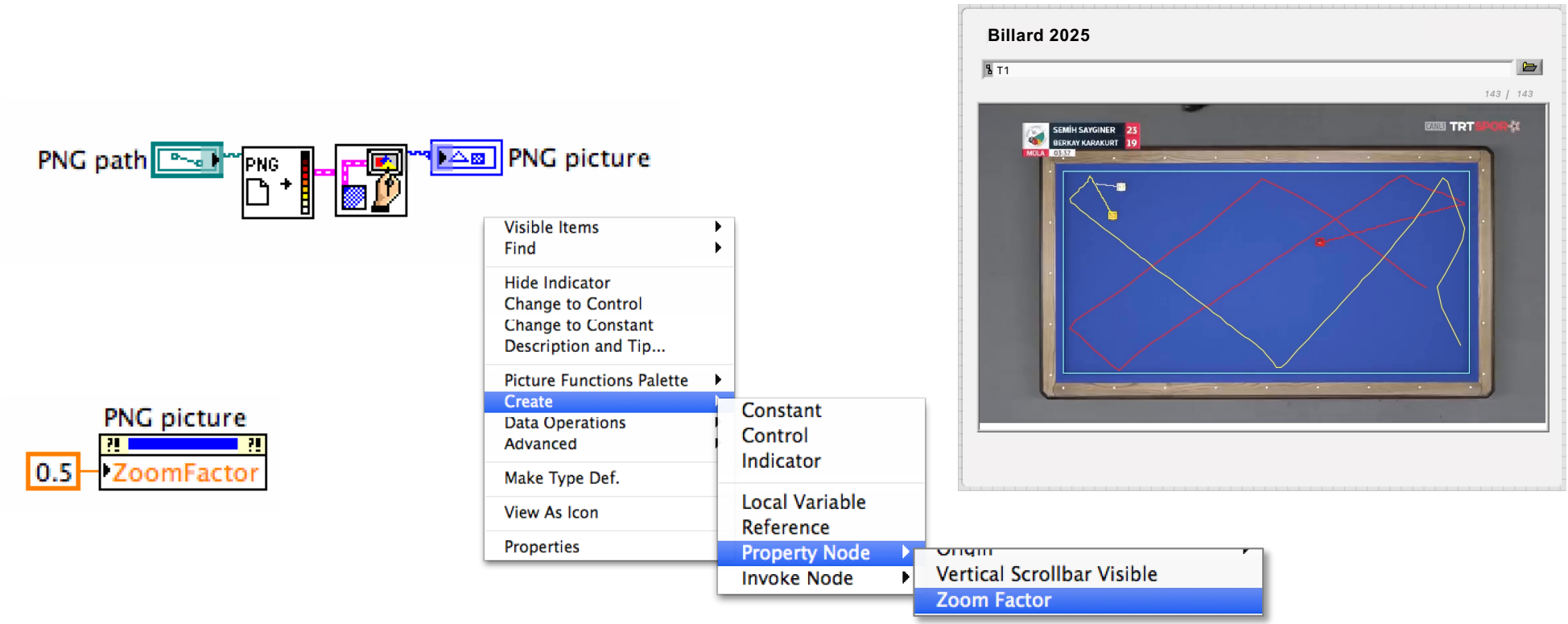




Cependant l'image est un peu grande

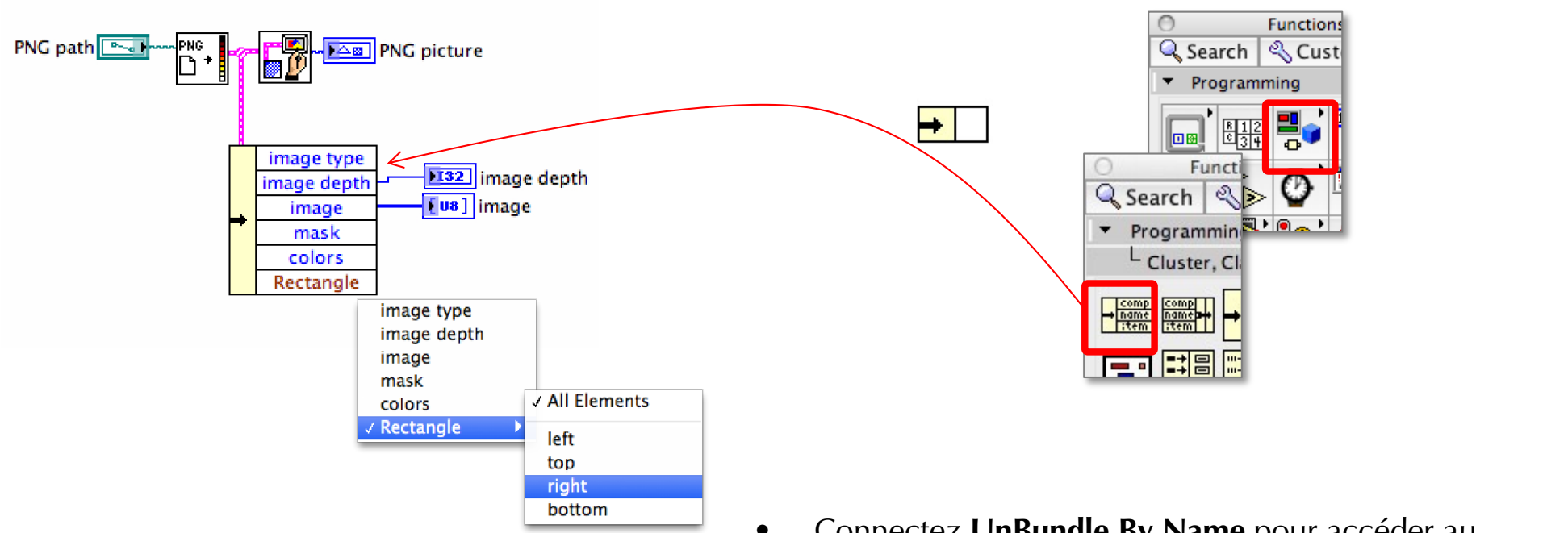


Echelle d'une image



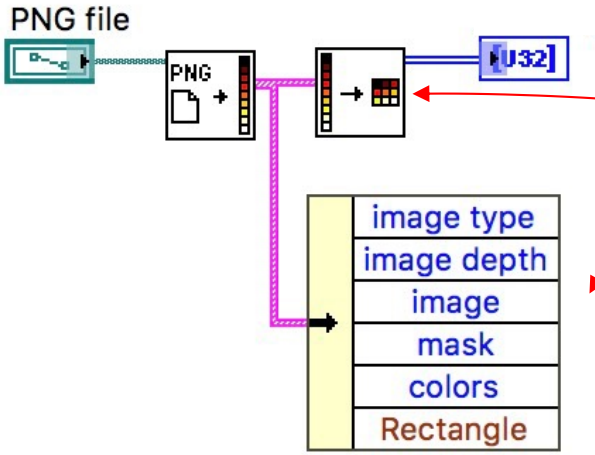
- Clic de droite sur l'icône de l'image **PNG Picture** pour créer un **Property Node : Zoom Factor**
- Connecter une constante qui defini la nouvelle échelle de l'image affichée, par ex. **0.5**
- L'échelle sera appliquée lors de la prochaine exécution du VI

Accès aux données de l'image

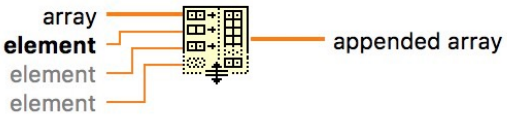


- Connectez **UnBundle By Name** pour accéder au donnée de l'image.
- Dès que vous connectez le fil de l'image, vous verrez apparaître les valeurs de celle-ci, au besoin agrandir **UnBundle By Name**
- Faire un clic de droite sur **Rectangle** pour sélectionner les éléments reprenstants la largeur et la hauteur

A faire: Image [U8]->[U32]->Pixmap

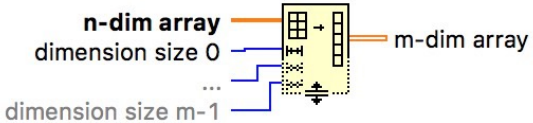


Build Array



Concatenates multiple arrays or appends elements to an n-dimensional array.

Reshape Array



Changes the dimensions of an array according to the values of **dimension size 0..m-1**.

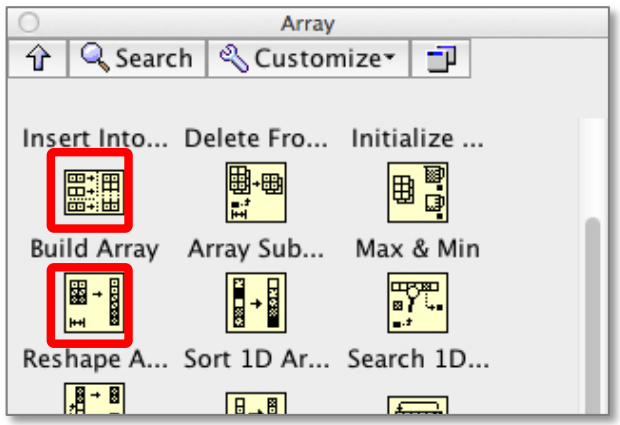
- Le vecteur **image** [U8] contient les valeurs RGB des pixels (soit 3 bytes) ordonné sous la forme:

  $R_0,G_0,B_0,\quad R_1,G_1,B_1,\quad \ldots R_{n-1},G_{n-1},B_{n-1}$

- Le VI **Unflatten Pixmap.vi** convertir ce vecteur 3 x [U8] en un tableau 2D [U32]

- *A faire*: créér le VI **Pix2U32.vi**.

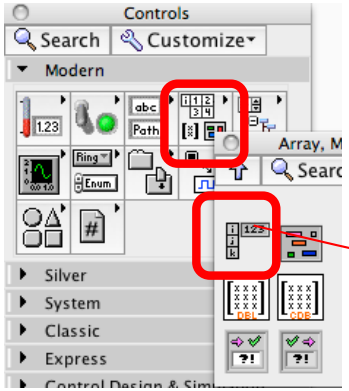
  A l'aide du VI **Build Array** vous devez ajouter la largeur et la hauteur au début de ce tableau 2D que vous aurez préalablement redimensionné en tableau 1D à l'aide de **Reshape Array**. La largeur et la hauteur sont disponible dans **Rectangle.**

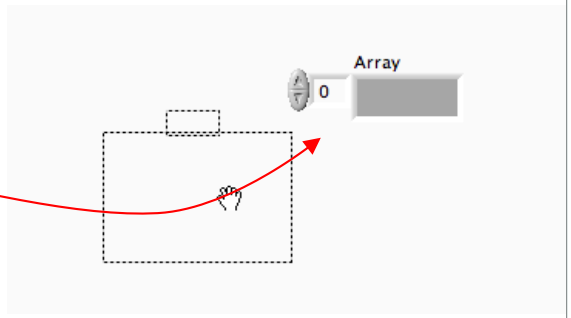


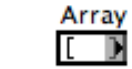
Array creation

Front Panel

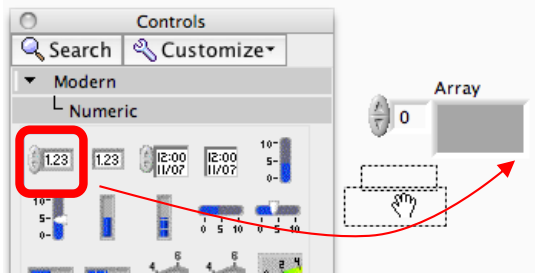
diagram

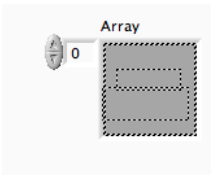




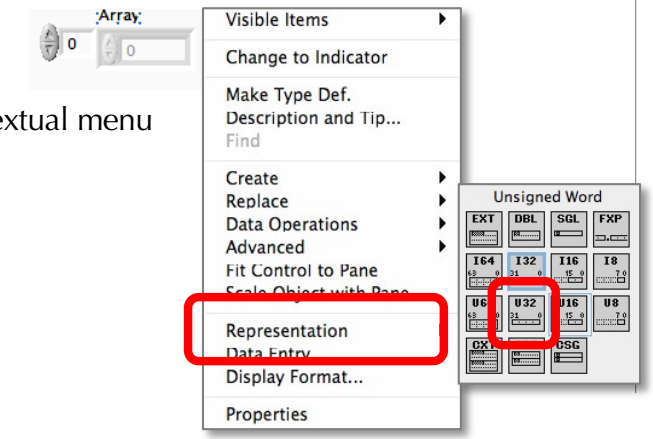


Créer un tableau vide

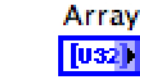




Glisser un élément pour lui donner un type



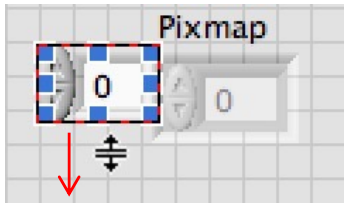
right click for contextual menu

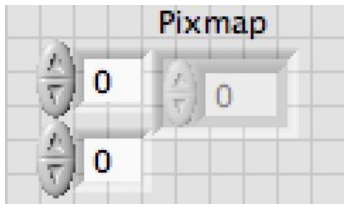


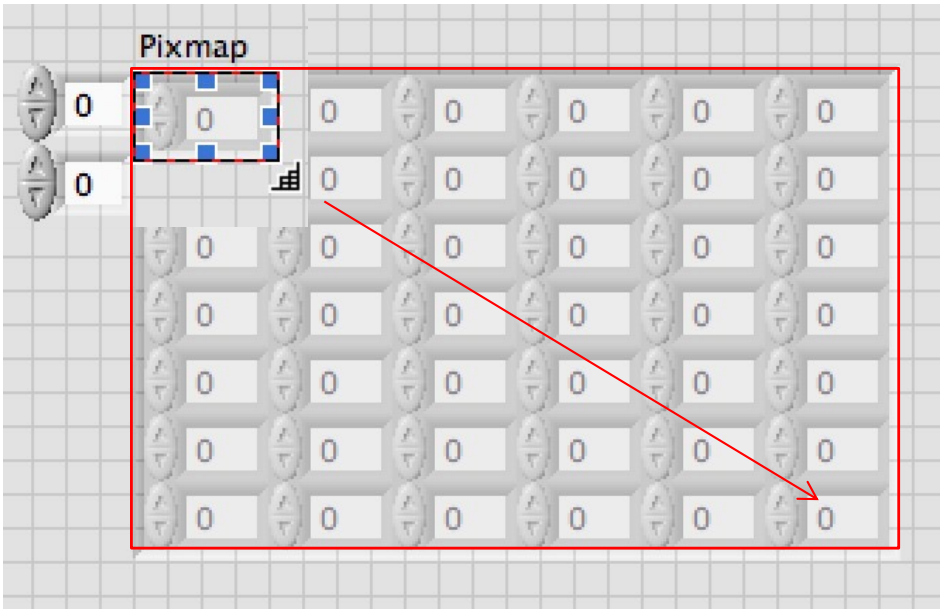
Changer le type des éléments du tableau en UInt 32

Array creation

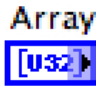
Front Panel



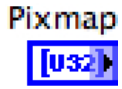




diagram



Ajouter une dimension à votre tableau
Et changer son nom

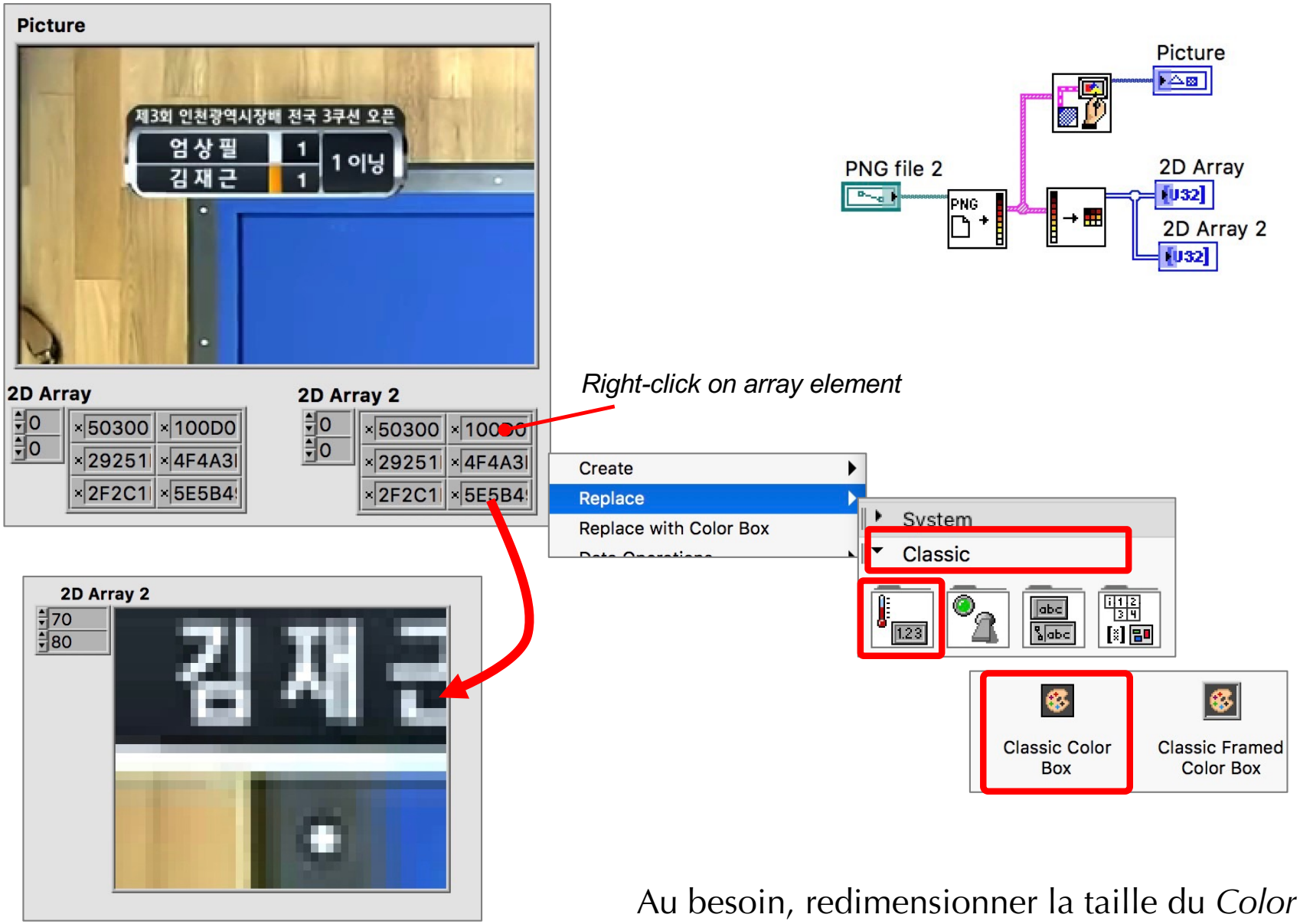


Notez que les [..] sont plus larges

Agrandir le tableau pour voir plusieurs éléments

Les cases grisées sont vides

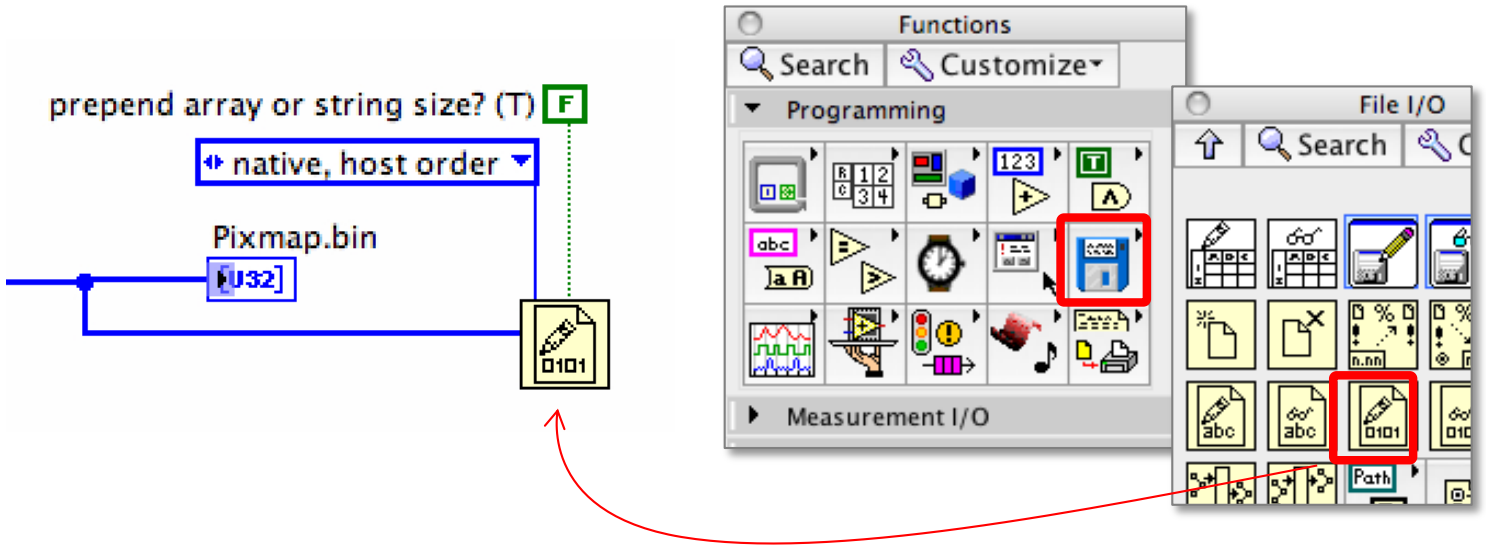
View pixels as U32 or color



Au besoin, redimensionner la taille du *Color Box*

Sauvegarde de pixmap.bin

Connectez votre array **Pixmap.bin** (contenant largeur, hauteur et pixels) au VI **Write to Binary file**. Afin de ne pas ajouter la taille du tableau au début du fichier mettez **`prepend array or string size? (T)`** à faux (F). De même vous devez ajuster *l'endianess* des données sauvées sur votre disque dur (**`little-endian`** ou **`native, host order`**). Lors de la prochaine exécution de votre VI, vous aurez un dialogue qui vous demande où sauver le fichier.



Afin d'éviter de devoir chaque fois spécifier le nom et chemin du fichier, vous devez construire le chemin du fichier (slides suivants) et le connecter au VI ci-dessus.

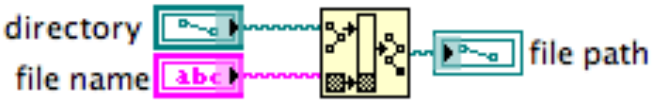
Chemin courant

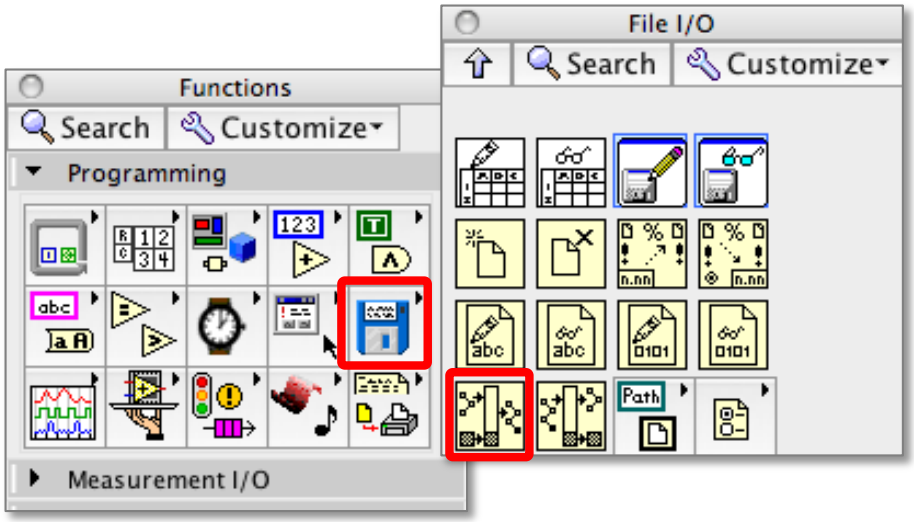
- La constante **`Current VI's path`** donne le chemin complet sur le VI courant (i.e. le chemin de celui qui contient la constante).
- De cette manière il sera possible d'avoir un accès relatif à vos fichiers et exécutable.
- **TOUS vos fichiers doivent se trouver dans le même dossier!**

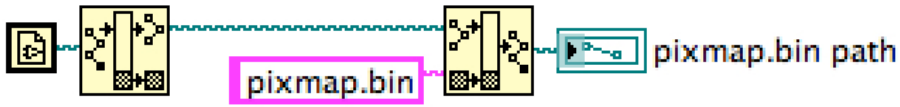
Chemin courant

- La fonction **`Build path`** permet de créer un chemin complet sur un fichier.
- De la même manière il est possible de créer un chemin complet relatif au VI courant,

  par exemple pour *Pixmap.bin*

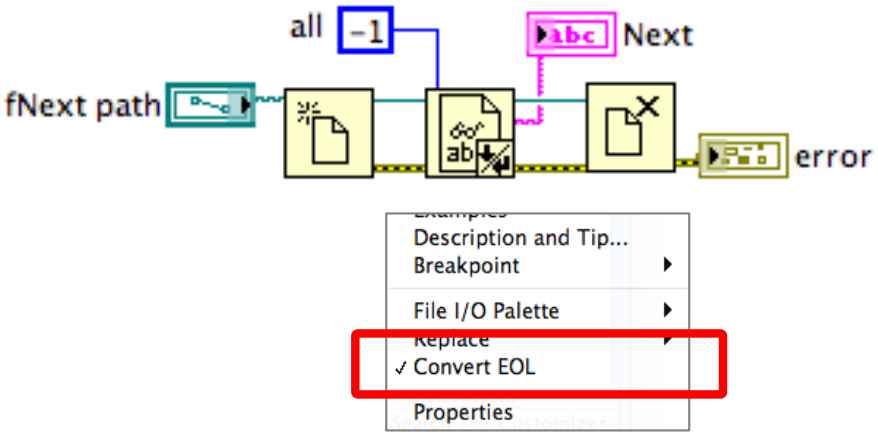


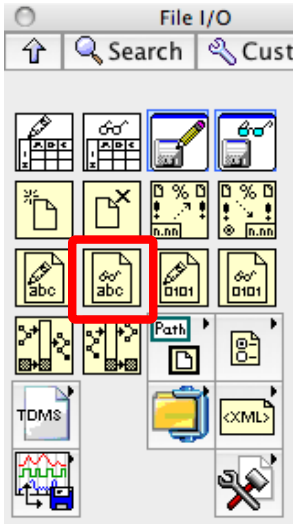




Lecture d'un fichier text

- Comme en C/C++vous devez ouvrir le fichier, lire les données et fermer le fichier.
- LabVIEW gère les fichiers au format txt (**`read from text file`**) et au format binaire.
- LabVIEW adapte le cartère de fin de ligne (clic de droite, option **Convert EOL**) en fonction de la plateforme sur laquelle est éxécuté le VI.
- Il est possible de lire un nombre défini de caratères dans le fichier, mettre '-1' pour lire l'entier du ficheir

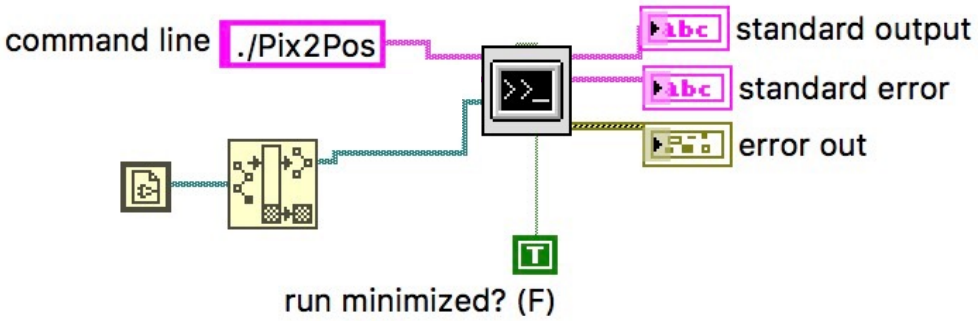


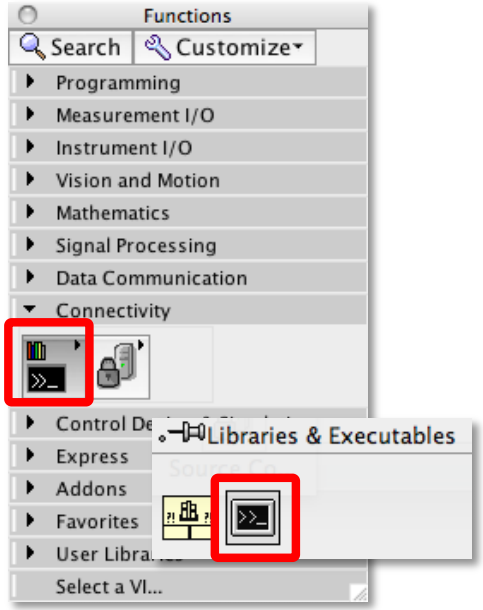


Regarder l'aide en ligne pour voir sous quelles conditions ***read from text file*** *peut être employé sans avoir à ouvrir et fermer explicitment le fichier.*

Appel d'un programme C

- **`System exec`** permet d'entrer une commande de la même manière que vous le feriez dans le terminal.
- La ligne de commande (**command line**) pour executer votre programme C commence par:
  ./ sous OSX/linux
  cmd /c sous windows
- Vous devez ajouter les 29 paramètres (pour le moment prendre ceux déjà fourni) à **command line**
- Cette commande est éxécutée dans le dossier courant (si cela a un sens)
- Il est possible de récupérer le contenu de 'stdout' et 'stderr'
- Les erreurs d'execussions sont retournées dans le cluster **error out**, Attention **stderr** ≠ cluster **error out** !
- Référez vous à l'aide pour les options supplémentaires





Ordre d'exécution

Assurez-vous de l'ordre d'exécution des Vis en connectant le cluster d'erreur entre les Vis.

1) Lecture du png
2) Création du tableau pixmap
3) Ecriture du tableau pixmap
4) Appel du programme C
5) Lecture des coordonnées

Ne oublier d'afficher l'erreur

mockup

A tester :

- Le tableau **pixmap.bin** est bien lu
- Le vecteur (L,H,Pixs) est bien construit et sauvegardé
- L'appel à votre exécutable fonctionne
- Les coordonnées retournées sont correctes
- Les erreurs retournées (ou non) sont correctes
- Que se passe-t-il si vous entrez des valeurs erronées ?

Done!

- L'étape suivante est la lecture et gestion de toutes les images d'une séquence