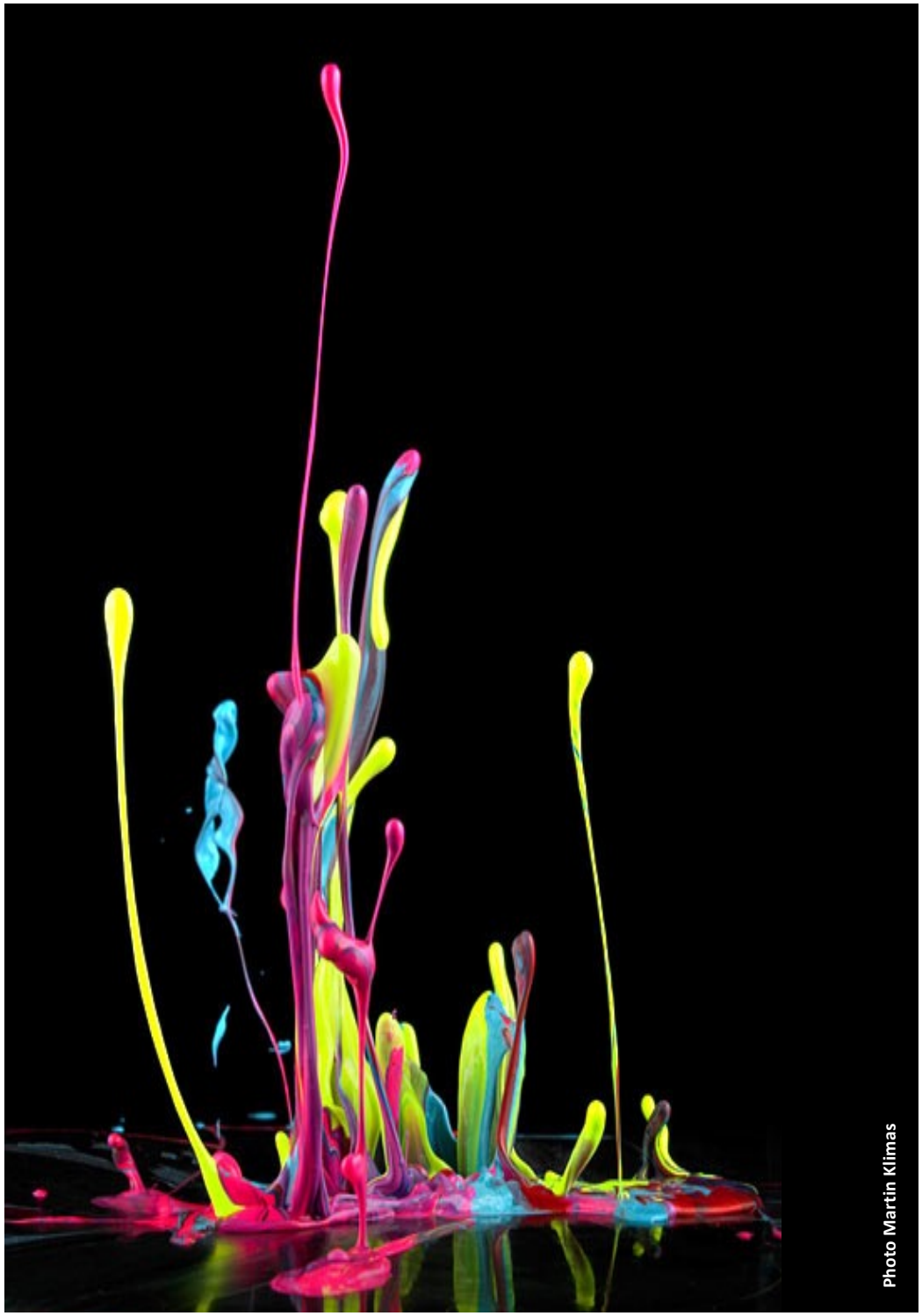


Photo Martin Klimas

EPFL

Programmation pour Ingénieur

Affichage + DAQ

ME 3e semestre

rev. 2025.1

Christophe Salzmann

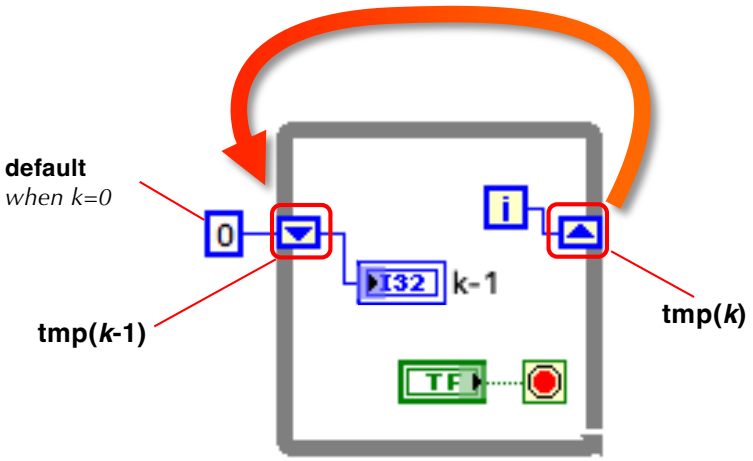
Laboratoire d'Automatique

Rappels LabVIEW 2

- Boucles
- Indexing
- Shift register
- Fils d'erreur défini l'ordre d'execussion
- Fichiers – idem C, Δ fermeture et endianess
- Call system exe

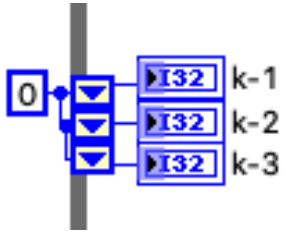
Dataflow & Shift register

Les shift registers sont des variables dépendantes du temps qui mémorisent la (les) valeur(s) précédente(s)



*k-1 = **0**,0,1,2,3, ...*

Valeurs précédentes



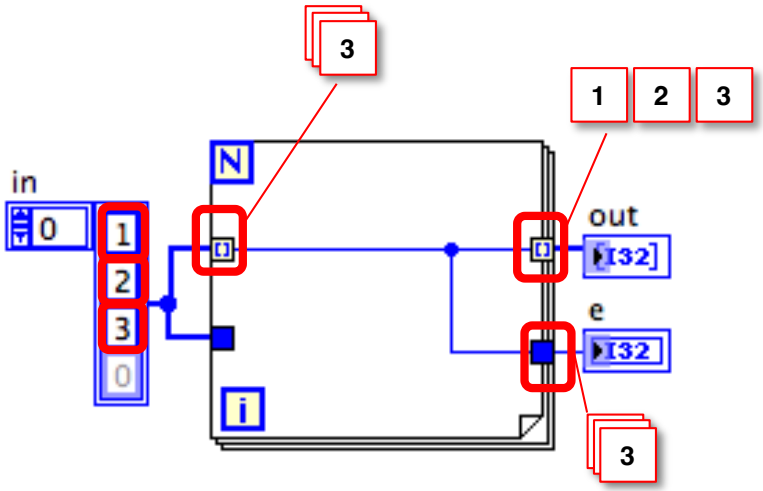
Boucle – auto-indexing

Auto indexing

- N nombre d'itérations définie par la **taille** du tableau d'entrée (in:3)

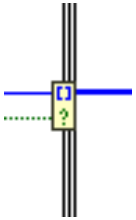
- auto indexing
- no auto-indexing

- Accède au i^ème élément du tableau d'entrée



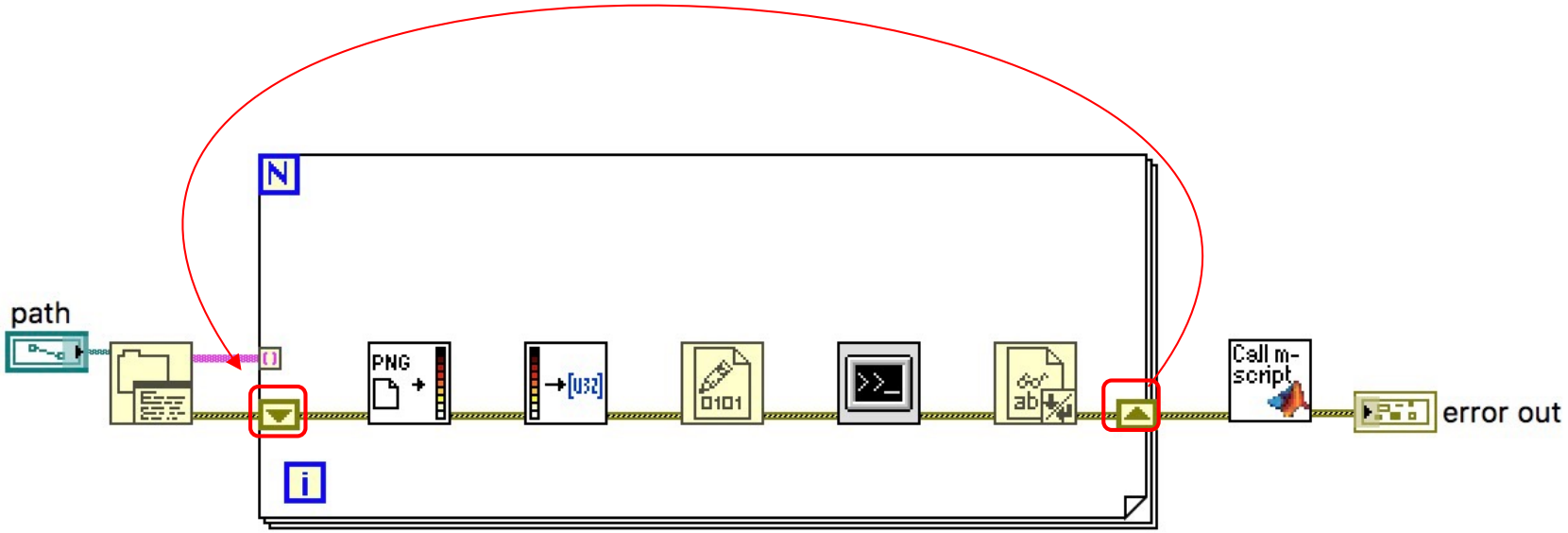
- out[] = {1, 2, 3}
- e = 3
- i = 2, N = 3

Ajout conditionnel



Rappel - gestion des Erreurs

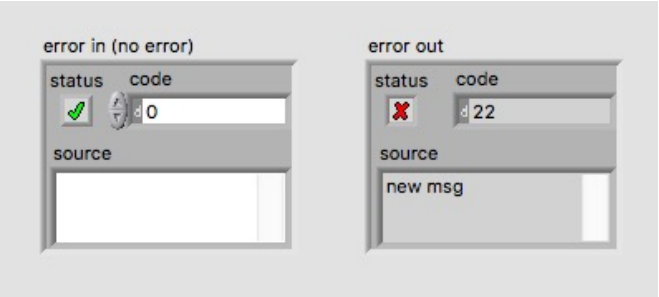
Chainez les Vis à l'aide des *errors* pour contrôler l'ordre d'exécution des VIs.
Dans les boucles (for, while), utilisez un *shift-registers* pour mettre à jour l'état des erreurs à chaque itération.

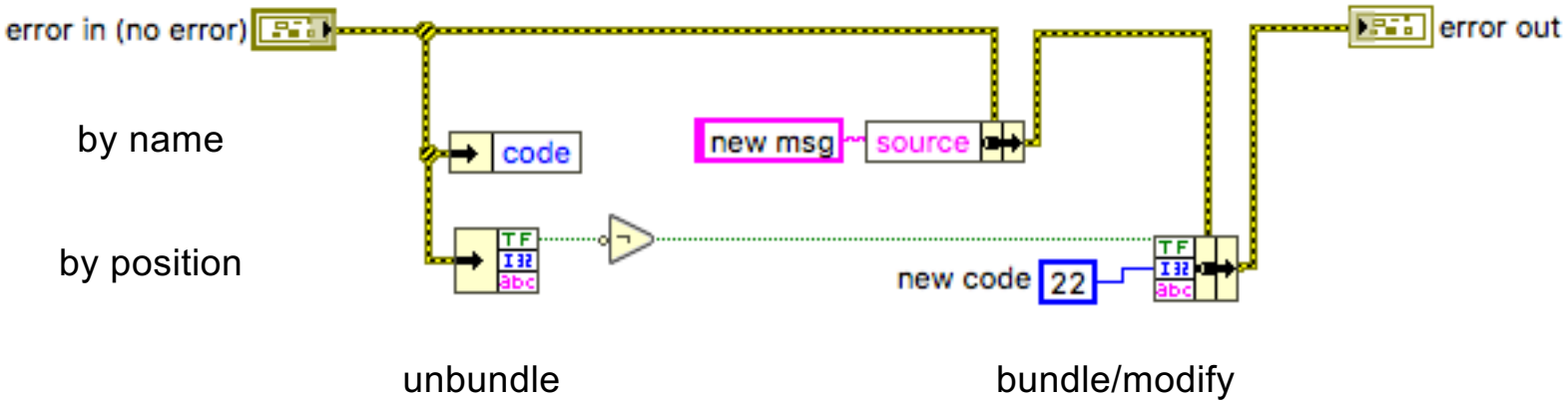


Cluster

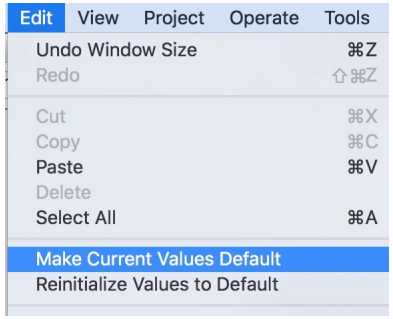
Cluster-> structure avec des éléments de types différents

- Equivalent des `struct` en C/C++
- Le nombre d'éléments des clusters est fixe
- La couleur des clusters indique si les éléments sont de taille fixe (brun) ou non (rose)
- Il est possible de mettre un Cluster dans un Cluster
- Les éléments (champs) du Cluster peuvent être accédés par leur nom (recommandé) ou leur position

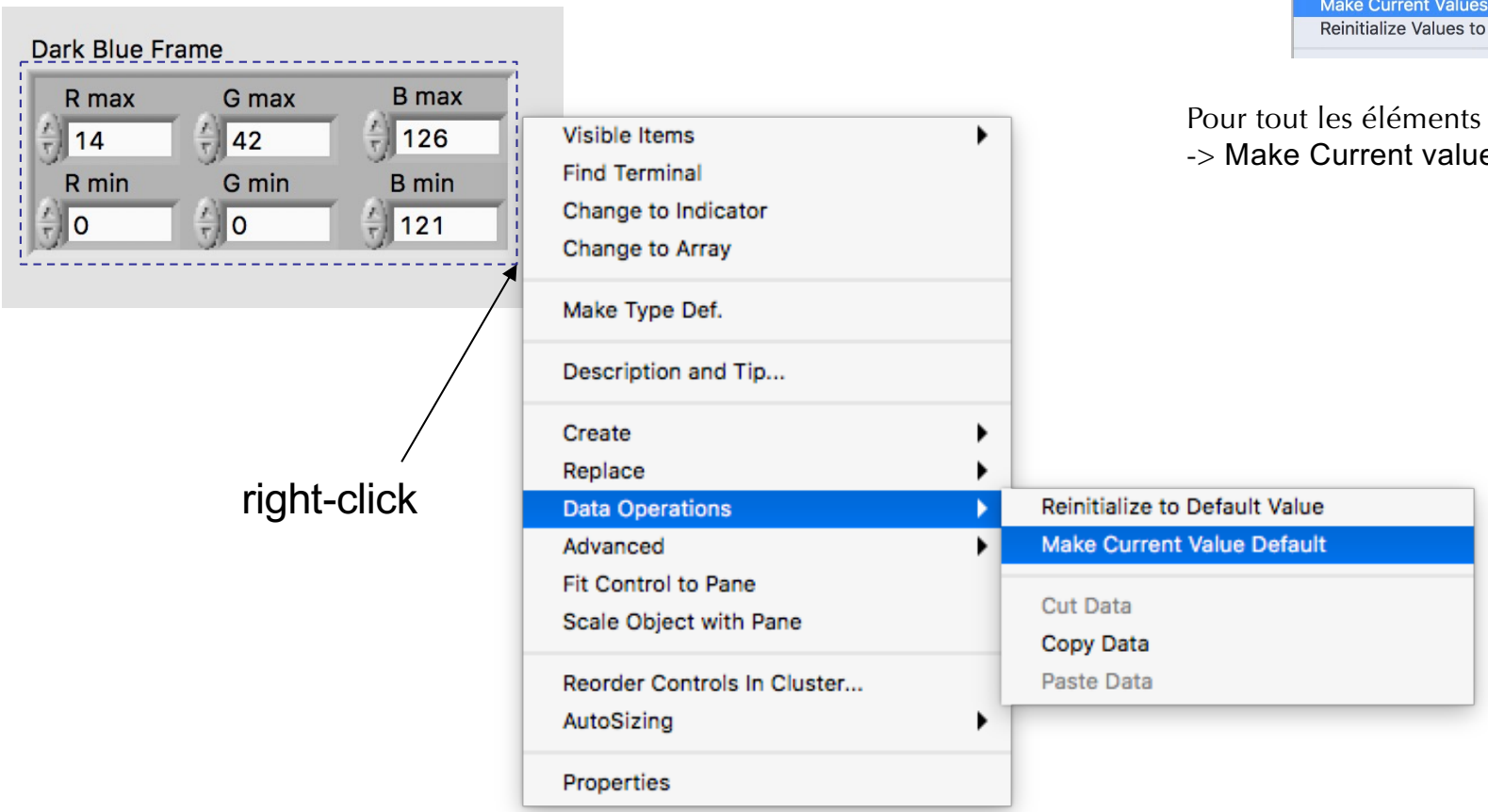




LabVIEW edition



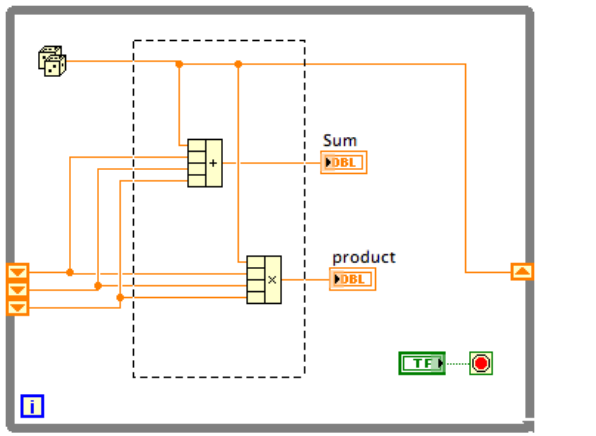
Make selected values default

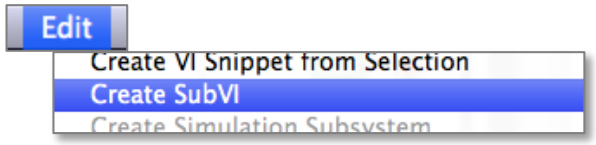


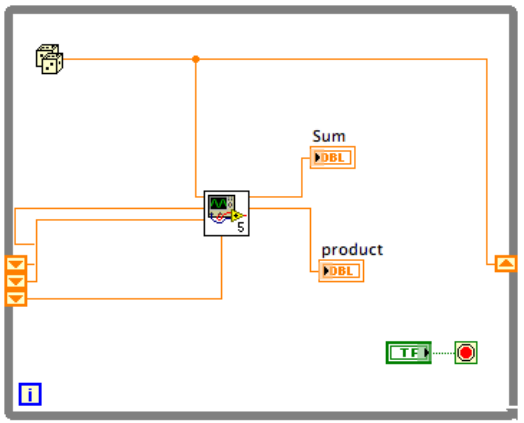
Pour tout les éléments de l'UI à la fois
-> Make Current values default

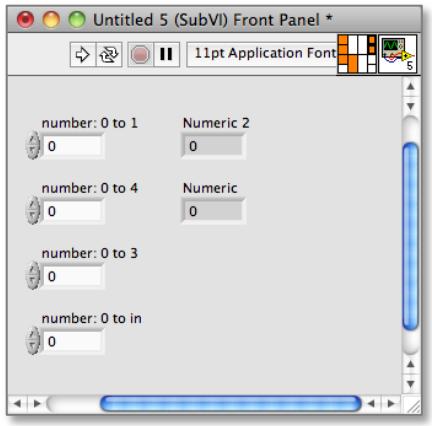
LabVIEW edition

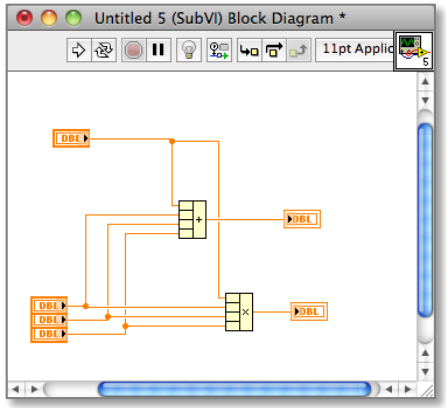
Create a sub-vi from a main vi in 2 clicks









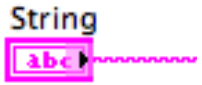


LabVIEW 3

- String
- Array
- Cluster, Matrix, (Waveform)
- Affichage
- Acquisition de signaux

Strings

- Les *strings* LabVIEW sont comme des string en C/C++.
- Les *strings* peuvent être redimensionnée dynamiquement, LabVIEW se charge de la gestion de la mémoire, elles ont une taille limites de 2^{32} ~4Millard de chars
- Les *strings* LabVIEW connaissent leur taille
- Un caractère est une *string* d'un élément, un caractère peut être représenté comme un unsigned 8bits Integer ou [U8].
- Les *strings* peuvent être employées comme des *streams* de bytes



C

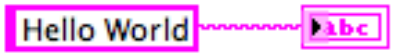
```
typedef struct
LVString {
  long size;
  char[] string;
}
```

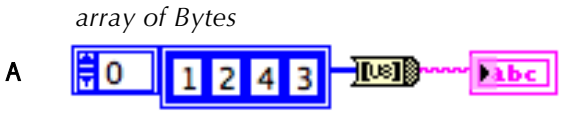
or C++

```
String string;
```

String functions

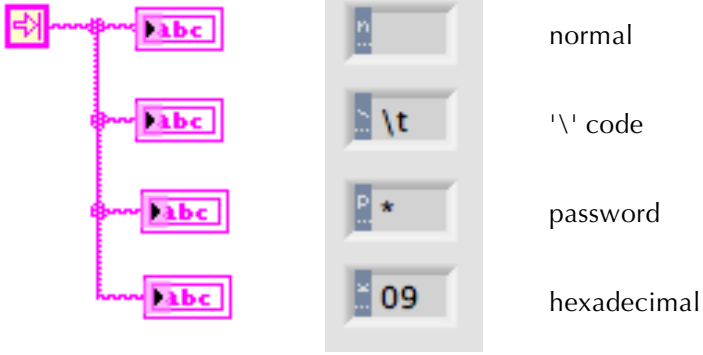
Initialize Strings





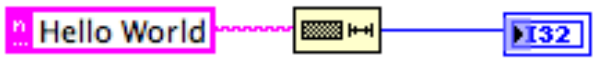

```
String S[] = '\r';

String S[] = 'Hello World';


UInt8 A[] = {1,2,4,3};

String S[] = (String)A;
```

Display formats



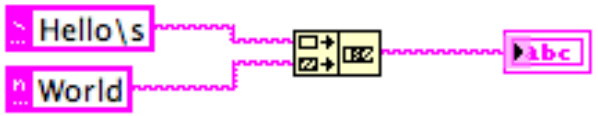
String functions

String Length



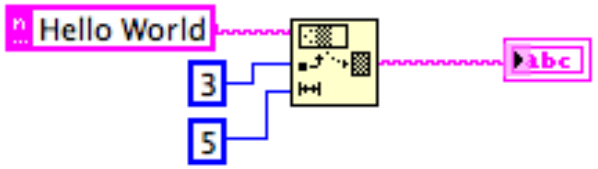

```
Length = 11
```

Concatenate



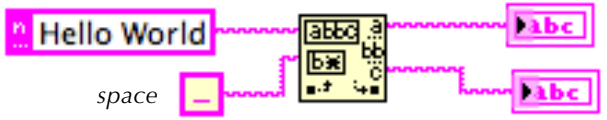

```
S = "Hello World"
```

String Subset



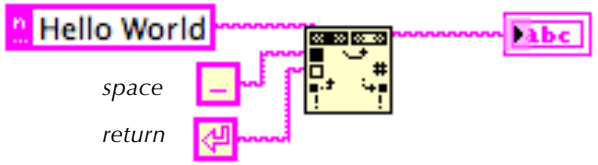

```
S = "lo Wo"
```

Match Pattern




```
S1 = "Hello"
S2 = "World"
```

Search & Replace

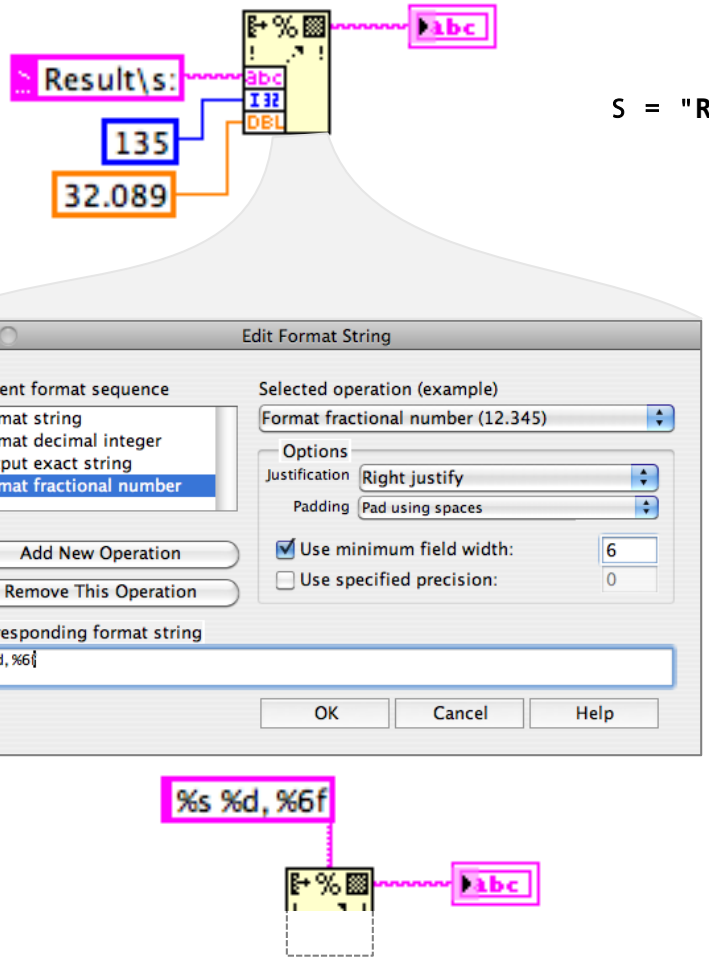



```
S = "Hello
     World"
```

```
Length = sizeof("Hello World");


S = S + "Hello " + "World";
strcat(S,"Hello ");
strcat(S,"World");


 S = StrSubset("Hello World",3,5);


 S[2] = SplitStr("Hello World","\s");


 S = ReplaceStr("Hello World","\s","\r");
```

String functions

Format Into String




```
S = "Result : 135, 32.089000"
```

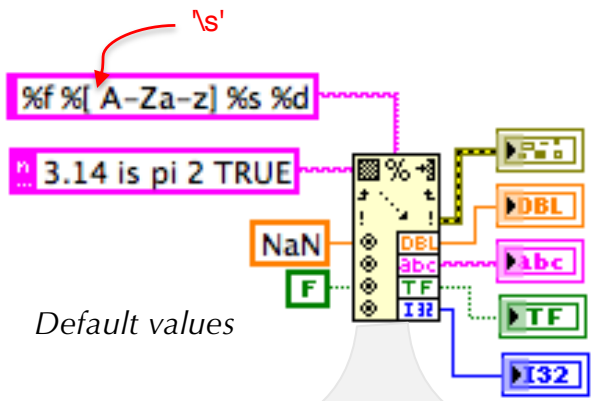
Right – click la fonction **Format into String** *pour éditer précisément le format de la conversion*

String format will be automatically added to the node

```
sprintf(S,"Result :%d, %6f",
                        135,32.089);
```

String functions

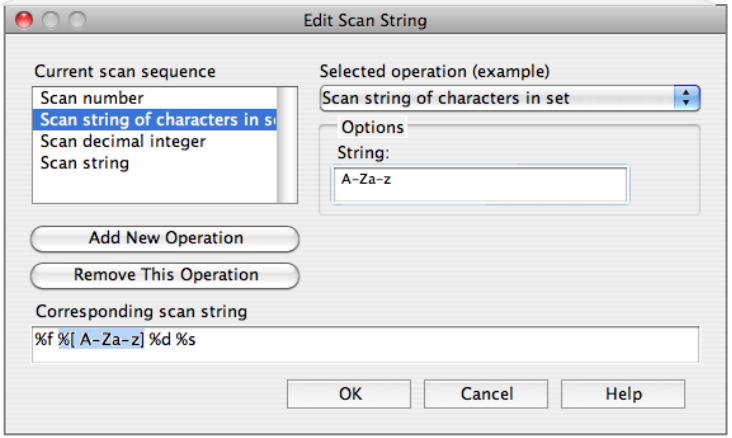
Scan from String



Default values

```
Error = "Scan From String (arg 4) in Strings.vi"
D = 3.14
S = "is pi"
B = FALSE
I = 0
```

Error because the last two elements are inverted



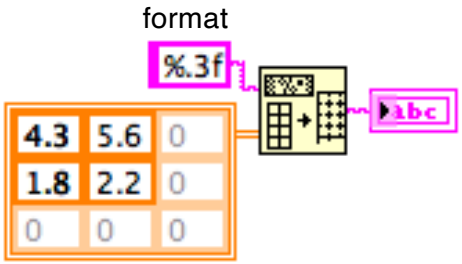
Right – click la fonction **Format into String** *pour éditer précisément le format de la conversion*

```
cin >> D >> S >> B >> I;

If (cin.fail()) ...


sscanf(S,"%f %[ A-Za-z] %s %d",
                    &D, S, &B, &i);
```

Use %s *for a single string (no delimiter like* \s, \r, *etc) otherwise specify the allowed characters in* [], *in the above example* [\s A-Za-z]

String functions

Array to Spreadsheet String

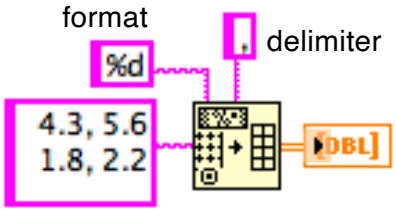



```
S = "4.300 5.600
     1.800 2.200"
```

%.3f -> double with 3 digits after '.'

```
Double A = {{4.3, 5.6} {1.8, 2.2}};
String F = "%.3f";
S = ArrayToSpreadSheetStr(A,F);
```

Spreadsheet String to Array



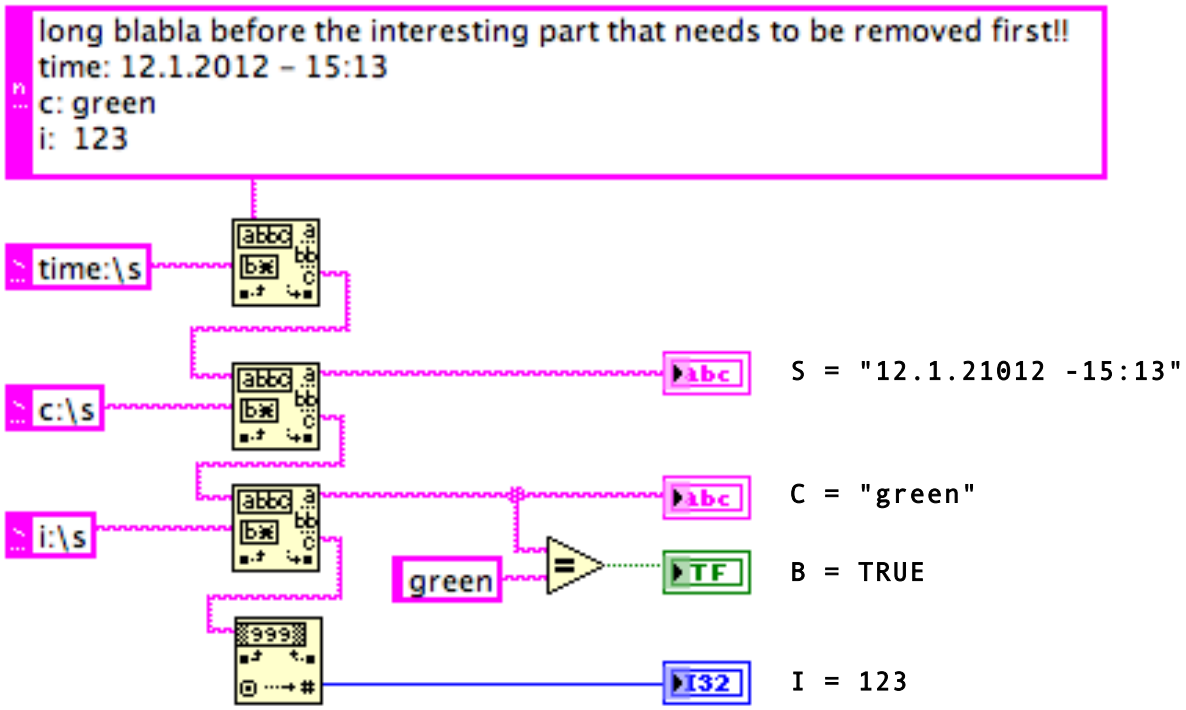

```
A = {{ 4   6 }
     { 2   2 }}
```

%d -> convert into integer (IEEE 754)

```
String S = "4.3, 5.6\r1.8,2.2";
String F = "%d";
String D = ',';
A = SpreadSheetStrToArray(S,F,D);
```

String functions

Example of string parsing

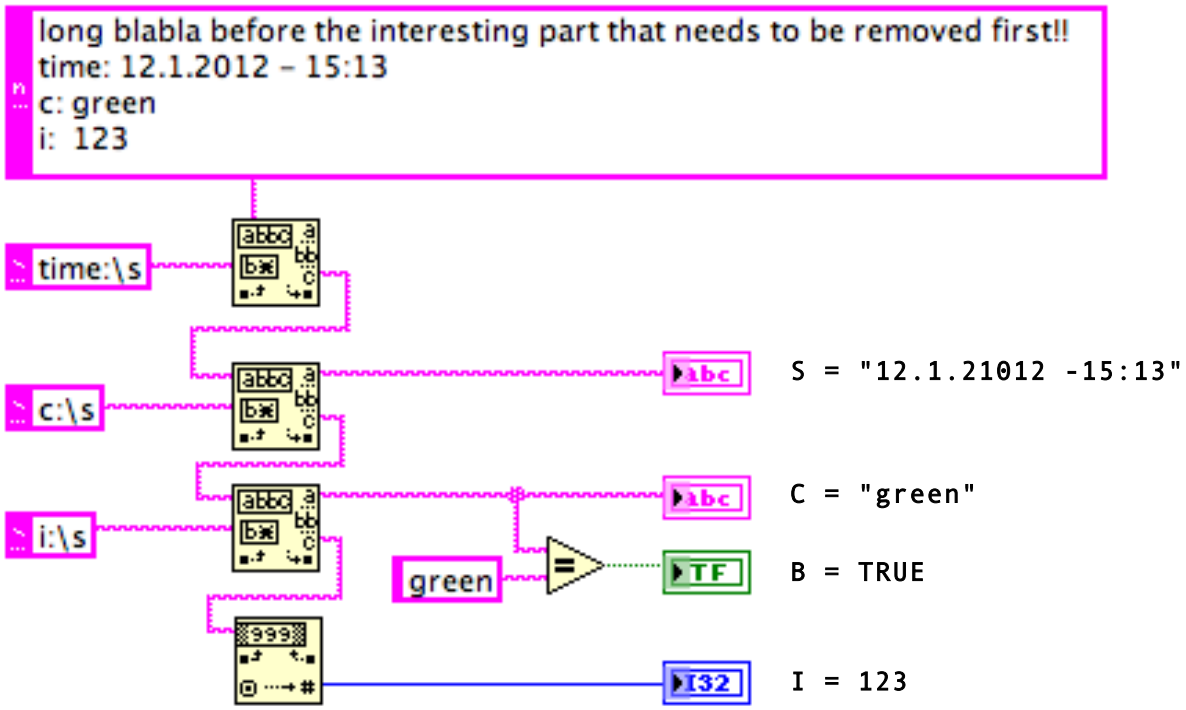


Decimal String to number

```
String LS = "long blabla before the
   interesting part that needs to be
   removed first!!\rtime: 12.1.2012 -
   15:13\rc: green\ri:  123"


MatchPattern(LS, "time:\s",
                 NULL,NULL,After);

MatchPattern(After, "c:\s",
                 S,NULL,After2);

MatchPattern(After2, "i:\s",
                 C,NULL,After3);

B = (C=="green");

atoi(After3,I)
```

String functions

Example of string parsing

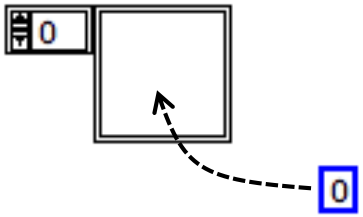


Decimal String to number

```
// -- c++ version --

string LS = "long blabla before the
   interesting part that needs to be
   removed first!!\rtime: 12.1.2012 -
   15:13\rc: green\ri:  123"

int P, I, B;
string T,S,C;



P = LS.find("time: ");
T = LS.substr(P + 6,LS.size());

P = T.find("c: ");
S = T.substr(0,P);
T = T.substr(P + 3,T.size());

P = T.find("i: ");
C = T.substr(0,P);


B = C.compare(0,5,"green");
T = T.substr(P+3,T.size());

I =atoi(T.c_str());
```

Tableau *array*

Tableau (Array) -> structure contenant des éléments du même type



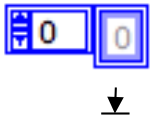


Tableau vide
0 est grisé
1 visible

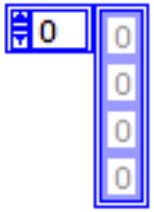
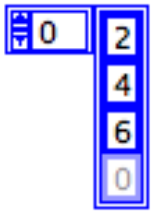
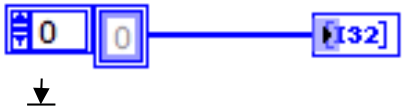


Tableau vide
0s sont grisés
4 visibles

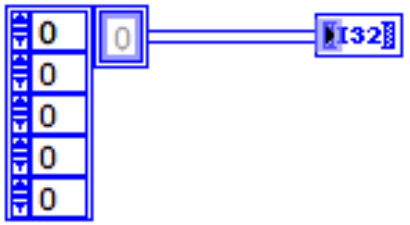


3 éléments
1 vide
4 visibles



1D array

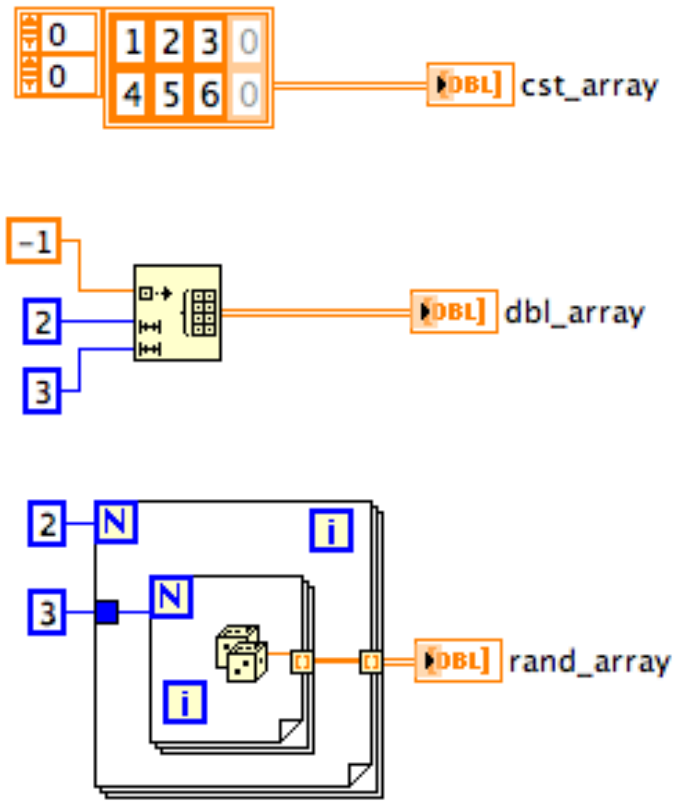
2D array



5D array

Array functions

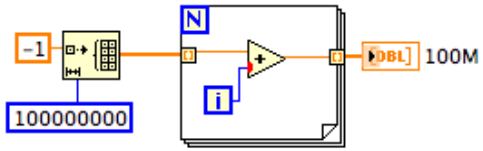
Initialize arrays



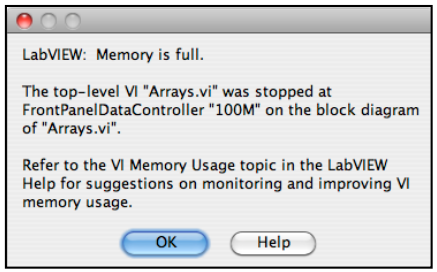

```
const dbl cst_array[2][3] =
      {{1,2,3} , {4,5,6}};


dbl dbl_array [2][3] =
      {{-1,-1,-1},{-1,-1,-1}};


dbl rnd_array[][];
for (r=0; r<2; r++)
   for (c=0;c<3;c++)

rand_array[r][c]=rand();
```

How big an array ?

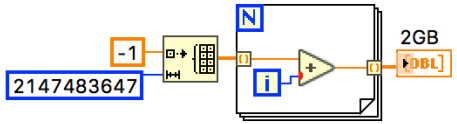
In LabVIEW 2013 32bit



100'000'000 pts x 8 Bytes ~ 800 MBytes



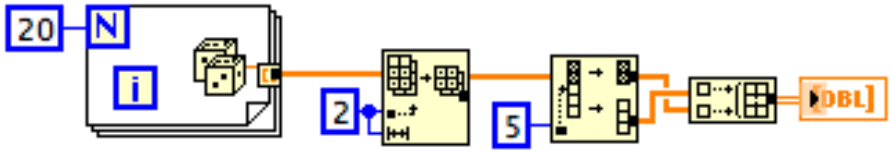
In LabVIEW 2017 64bit



2147483647 = MaxInt = 2^{31}-1 ~ 2 GBytes

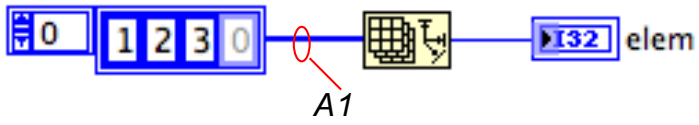
LabVIEW 2017-64bit can handle such size

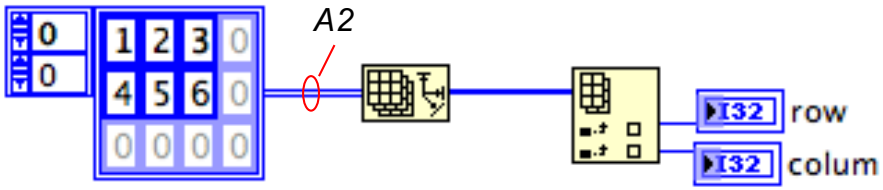
- Utilisez quand c'est possible des primitives *in-place* qui ne travaillent directement sur les tableaux sans les copier
- *Check the Buffer Allocations,* Tools»Profile»Show Buffer Allocations



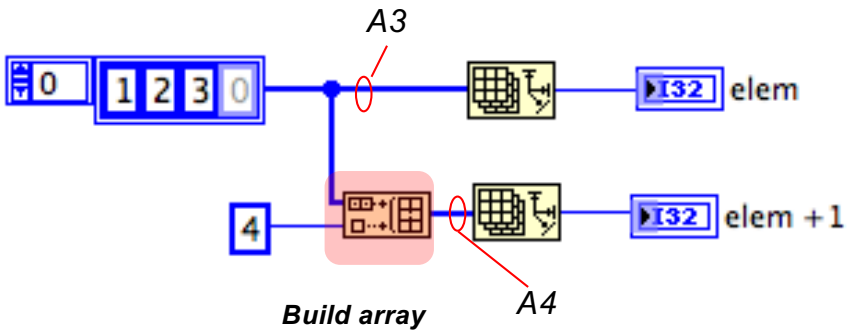
Array functions

Les tableaux LabVIEW connaissent leur(s) taille(s)





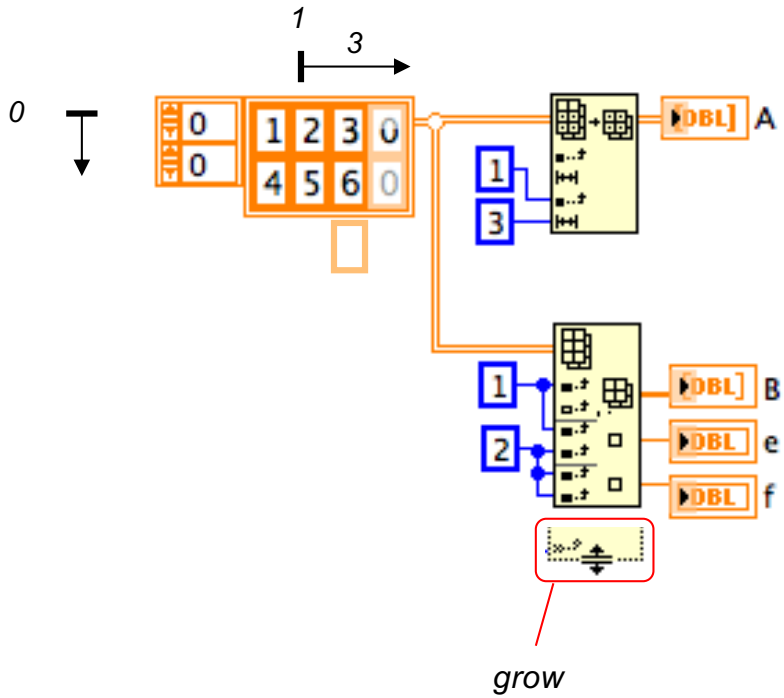
Les tableaux LabVIEW sont dynamiques




```
int A1[3] = {1,2,3};

elem = ArraySize(A1);



int A2[3] = {{1,2,3},
{4,5,6}};

row = Get(ArraySize(A2),0);

Colum = Get(ArraySize(A2),1);



int A3[3] = {1,2,3};

elem = ArraySize(A3);

A4= BuildArray(A3,4);

elem_1 = ArraySize(A4);
```

Array functions

Accès aux éléments du tableau équivalent aux [ ] en c



A[]={{2,3}, {5,6}}

B[]={4,5,6}

e = 6

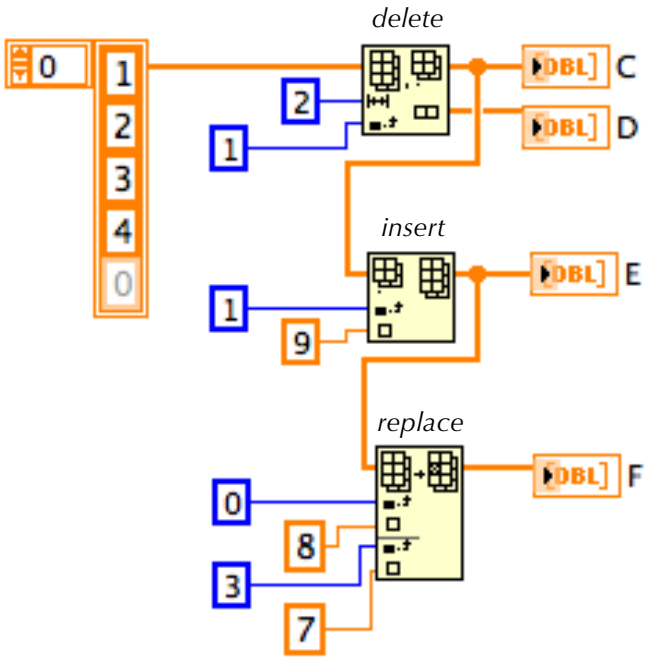
f = 0

Unconnected inputs are set to default (see help window for default values)

```
int Array[6] = {{1,2,3},
                {4,5,6}};
A = GetSubArray(Array,def,all,
                          1,3);


B = GetIndexArray(Array, 1,
all);
e = GetIndexElem(Array, 1, 2);
f = GetIndexElem(Array, def,
2);
```

Array functions

Delete/insert/replace array elements



C[] = {1,4}

D[] = {2,3}

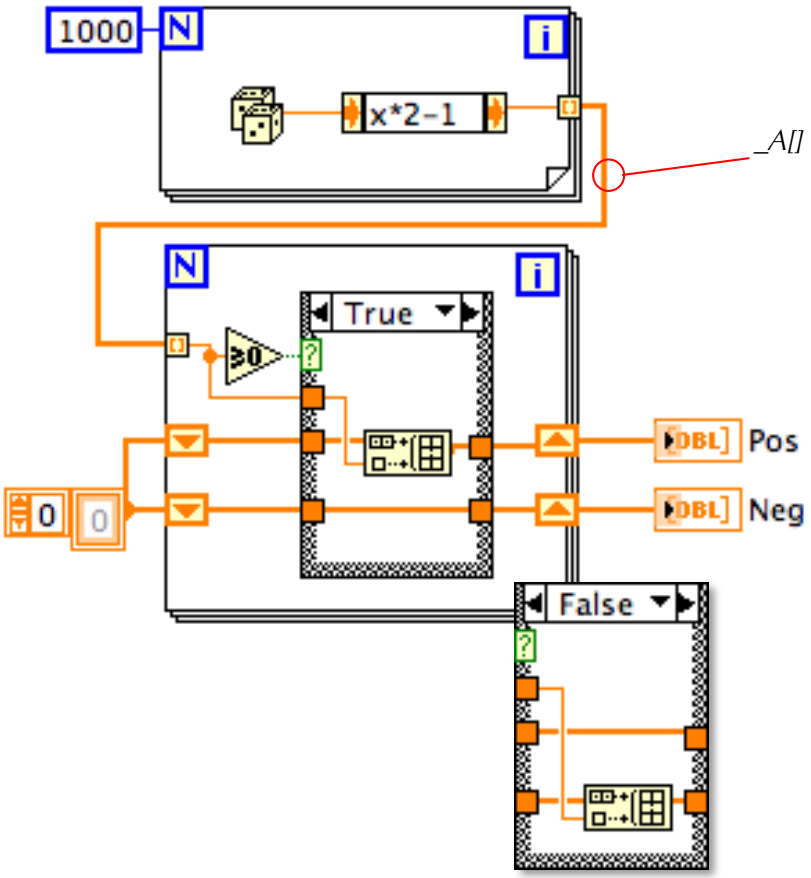
E[] = {1,9,4}

F[] = {8,9,4}

```
int Array[4] = {1,2,3,4};

Delete(Array,2,1,C,D);

Insert(C,1,9.0,E);

Replace(E,0,8.0,F);
Replace(F,3,7.0,F);
```

Array functions

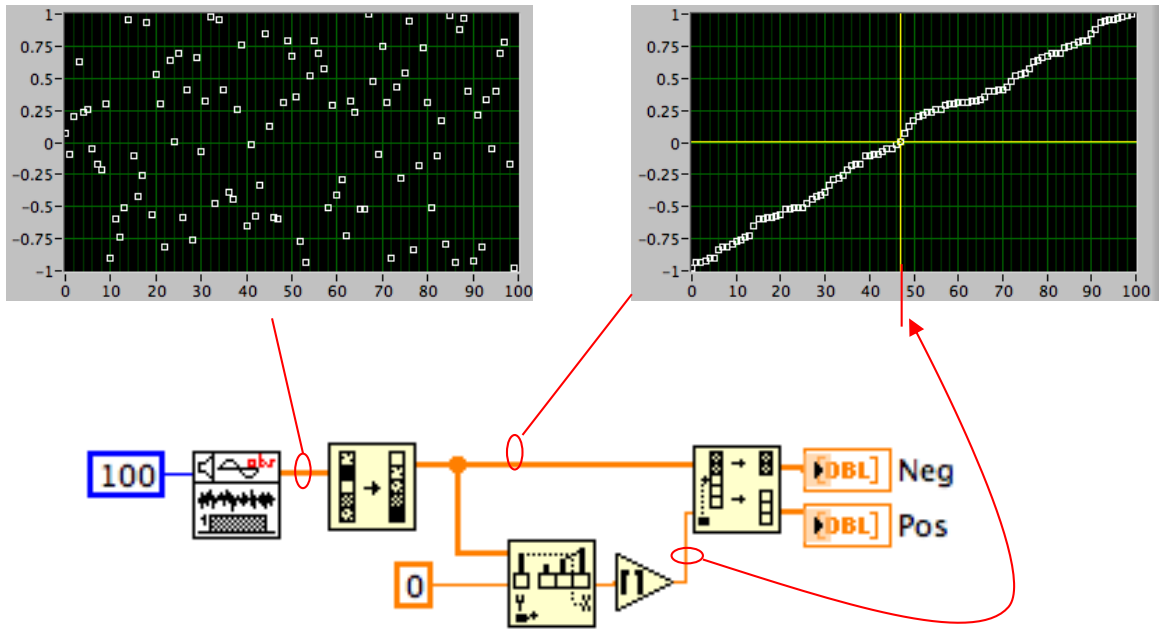
Ex: sort in 2 arrays positive from negative values




```
double  Pos[] = {};
double  Neg[] = {};
double  _A[];

for (i=0; r<1000; i++)
    _A[i]=rand() * 2 - 1;

for (j=0; j<size(_A),j++) {
  if (_A[j]≥ 0)
        AppendArray(Pos,
_A[j]);
      else
        AppendArray(Neg,
_A[j]);
```

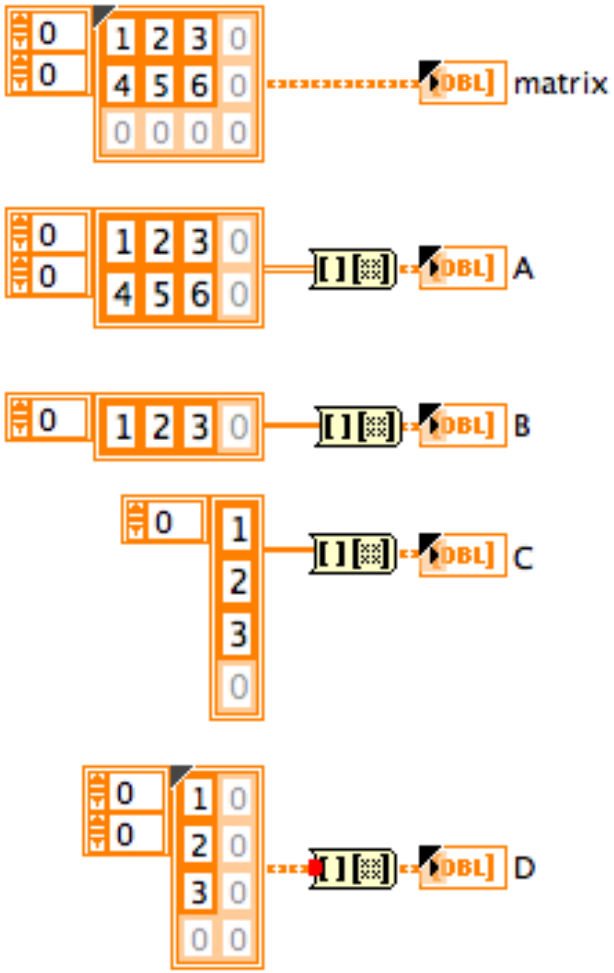
Array functions

Ex: sort positive from negative values, alternative solution (values indexes differ)




```
double A[100];

double B[100];

double Neg[], Pos[];

Double  T=0;

int i;


GenRand(100, &A);

Sort(A, &B);

for (i=0, i<SizeArray(B), i++){

   if (B[i]<T) && (B[i+1]>=T))

        break;

   i++;

}

Split(B, i, &Neg, &Pos);
```

G data types - Matrix

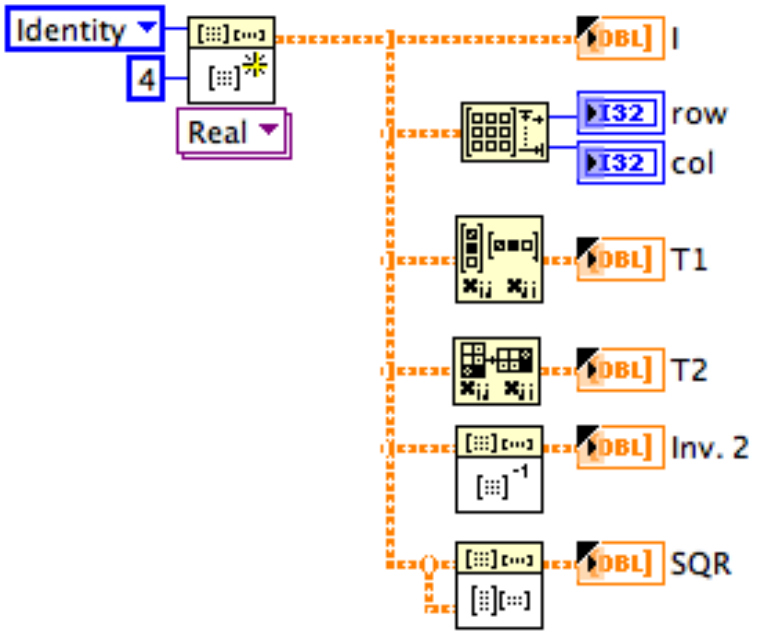


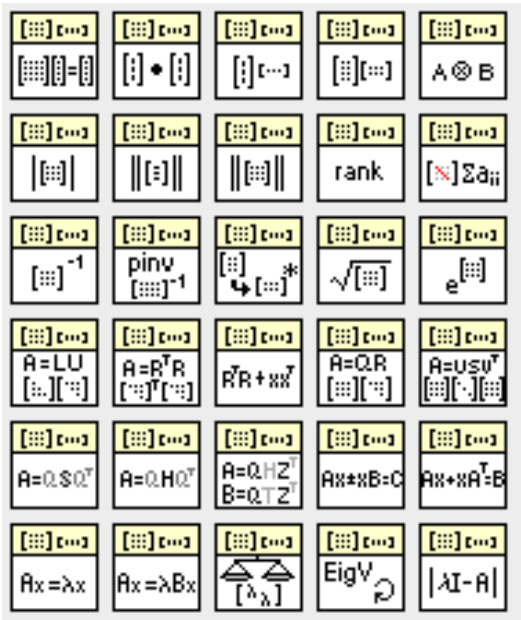
$$B \overset{?}{=} C \overset{?}{=} D$$

$B = C$

$C \neq D$

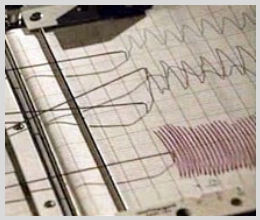
Matrix functions

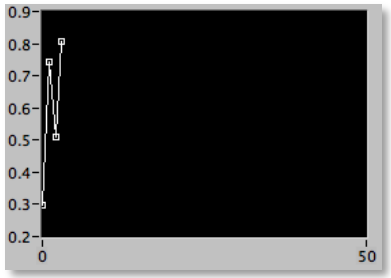




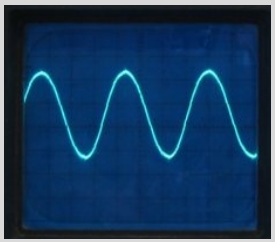
GUI - display data

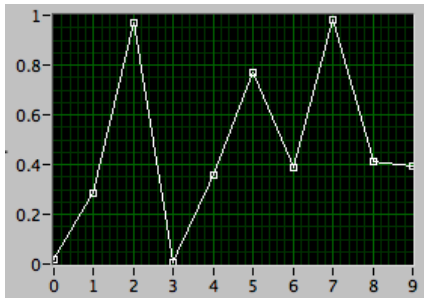
chart

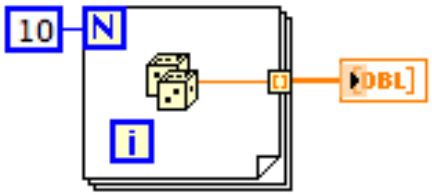




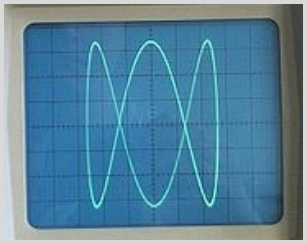
graph

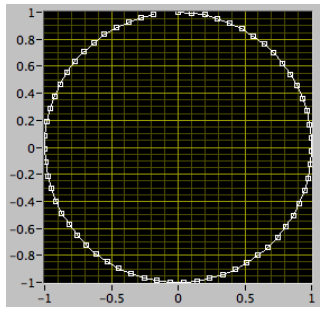


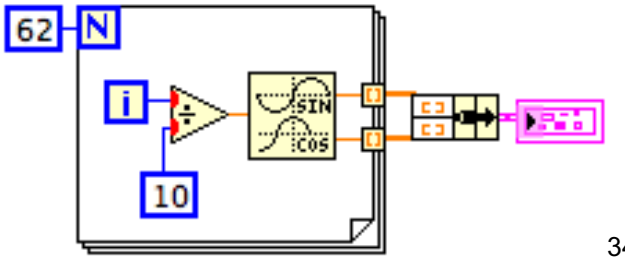




x-y graph

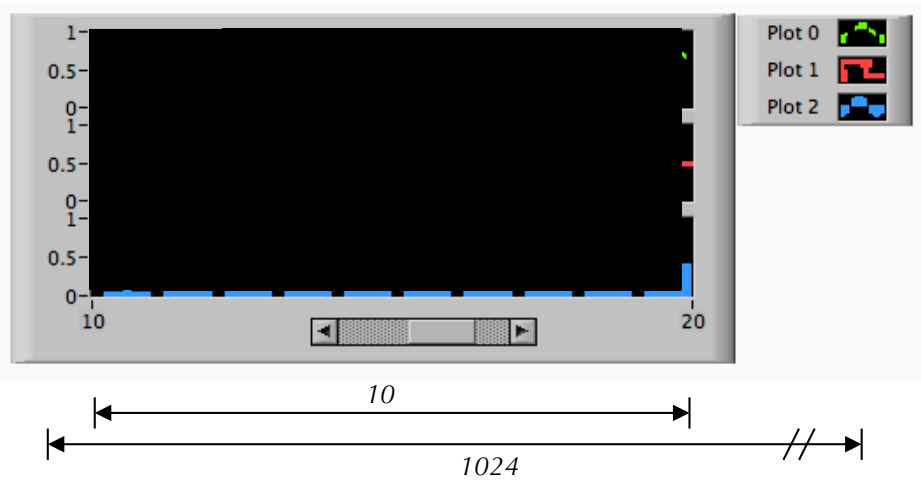


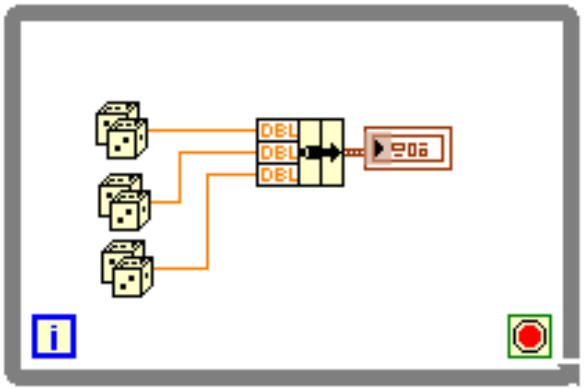


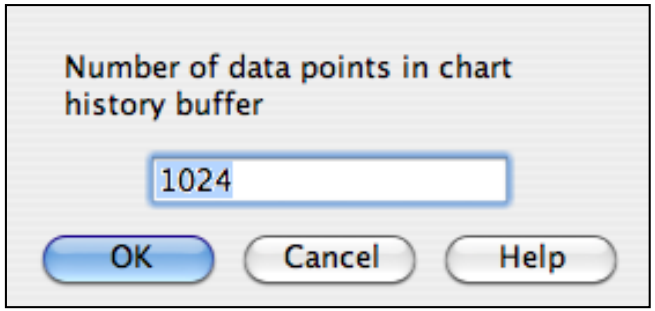


Chart

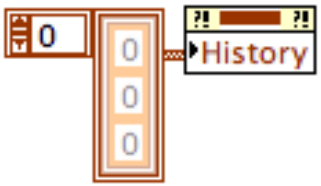
Ajoute une (set de) valeur(s) à la fois, i.e. à chaque iteration/appel







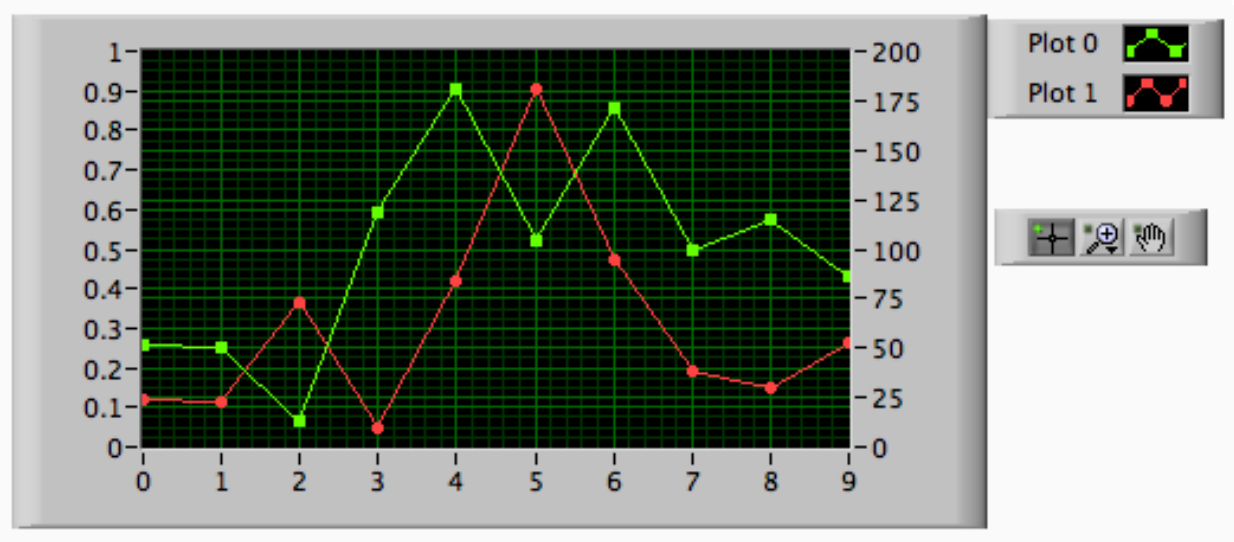
Utilisez un *attribute node* pour remettre le chart à 0

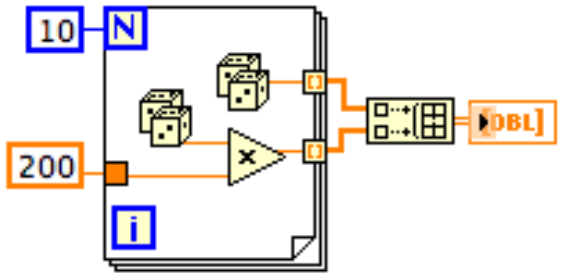


Graph

Ajoutes** n **plots à chaque appel/itération, le graph est renouvelé chaque fois

Si les données connectées sont au format waveform, l'échelle des x s'adapte automatiquement

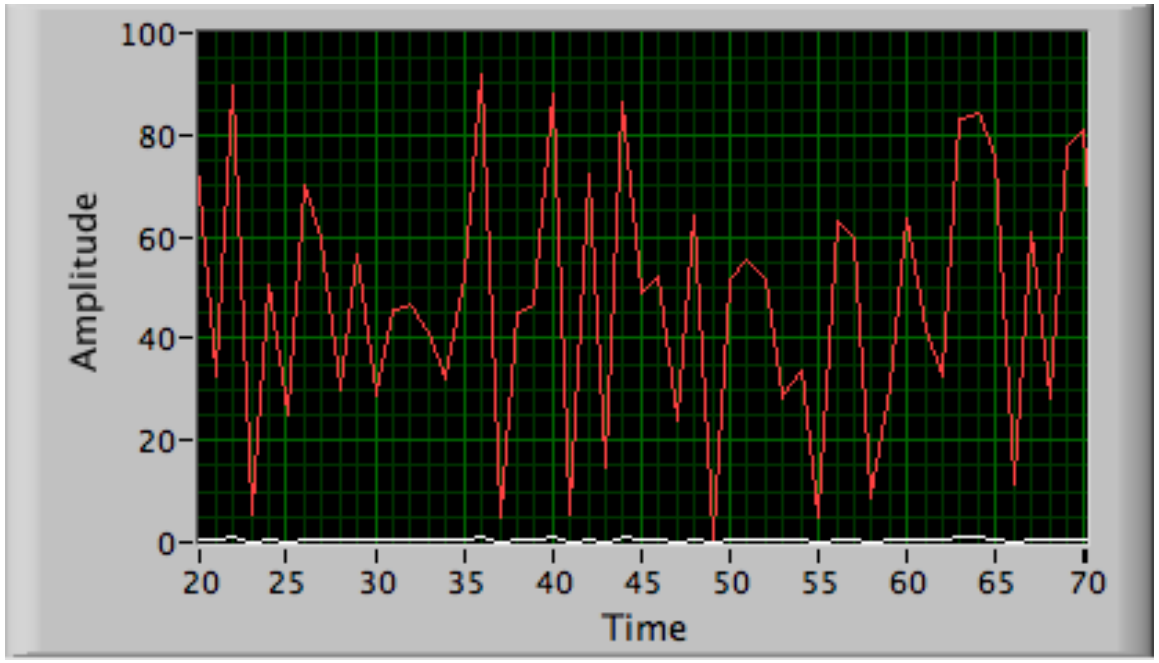


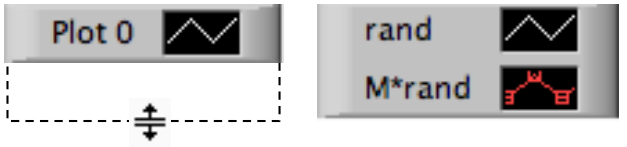


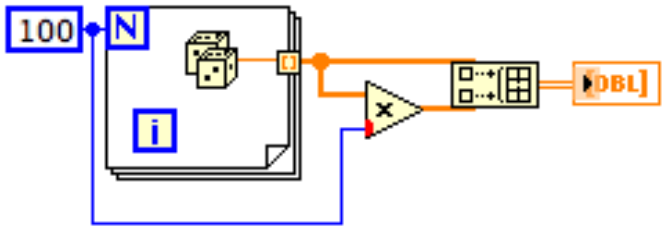
Problème de mise à l'échelle ?

- *Ajoute une nouvelle échelle des Y*
- *Assigne chaque plot à son échelle respective*

Graph - customization

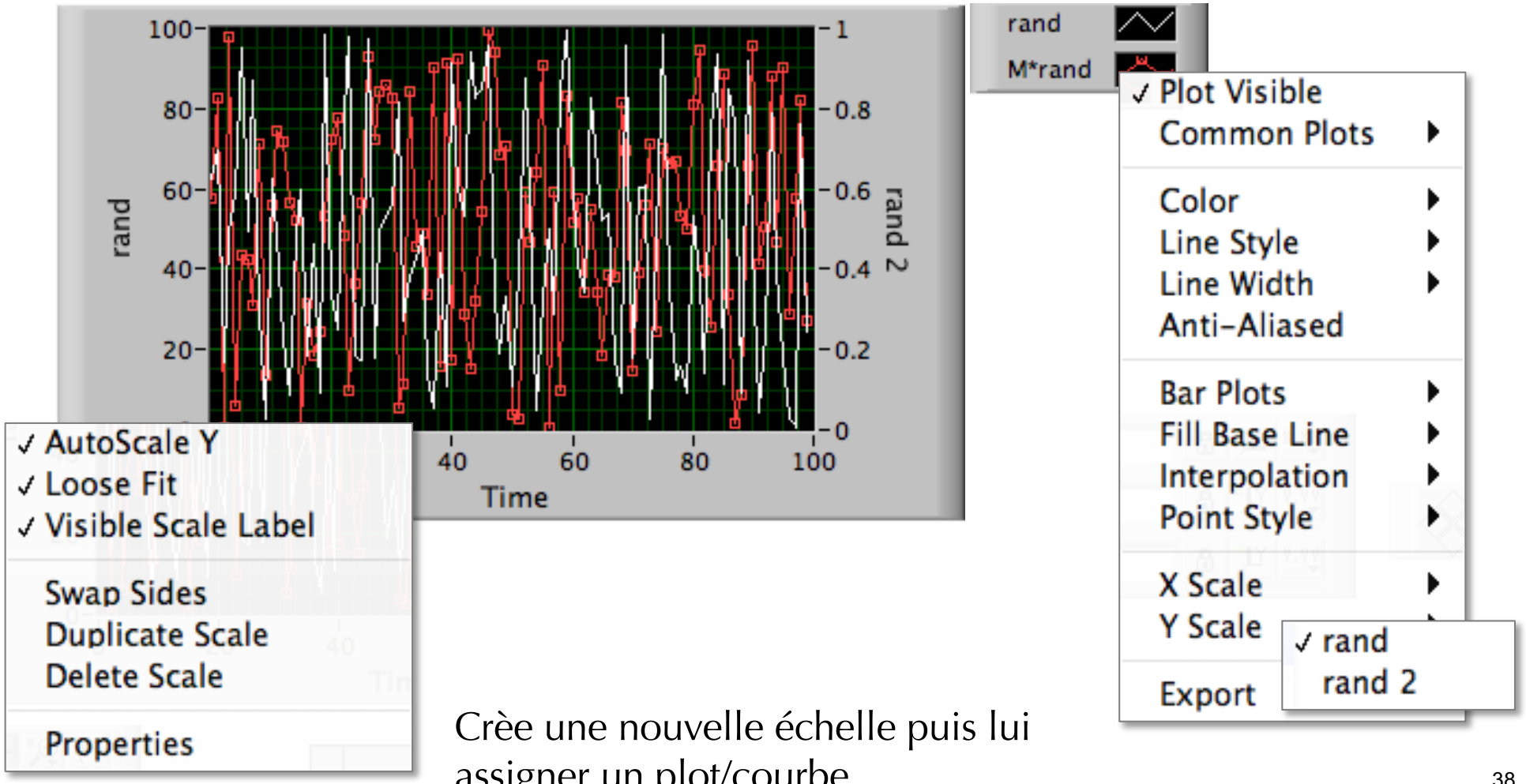






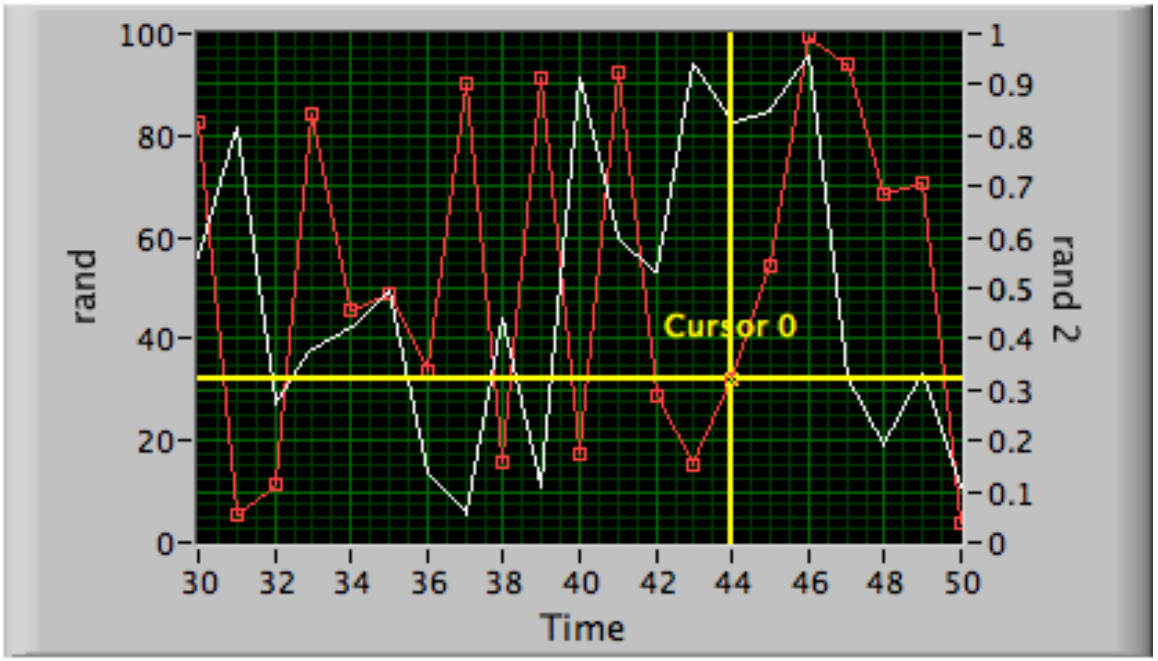
Click ou right-click pour accéder ou modifier les propriétés du grap

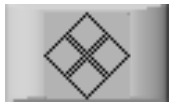
Graph - customization

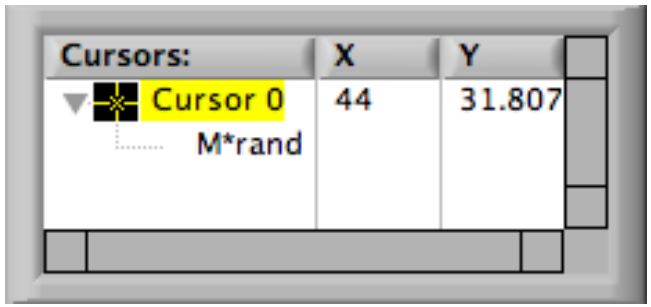


Crèe une nouvelle échelle puis lui assigner un plot/courbe

Graph - customization



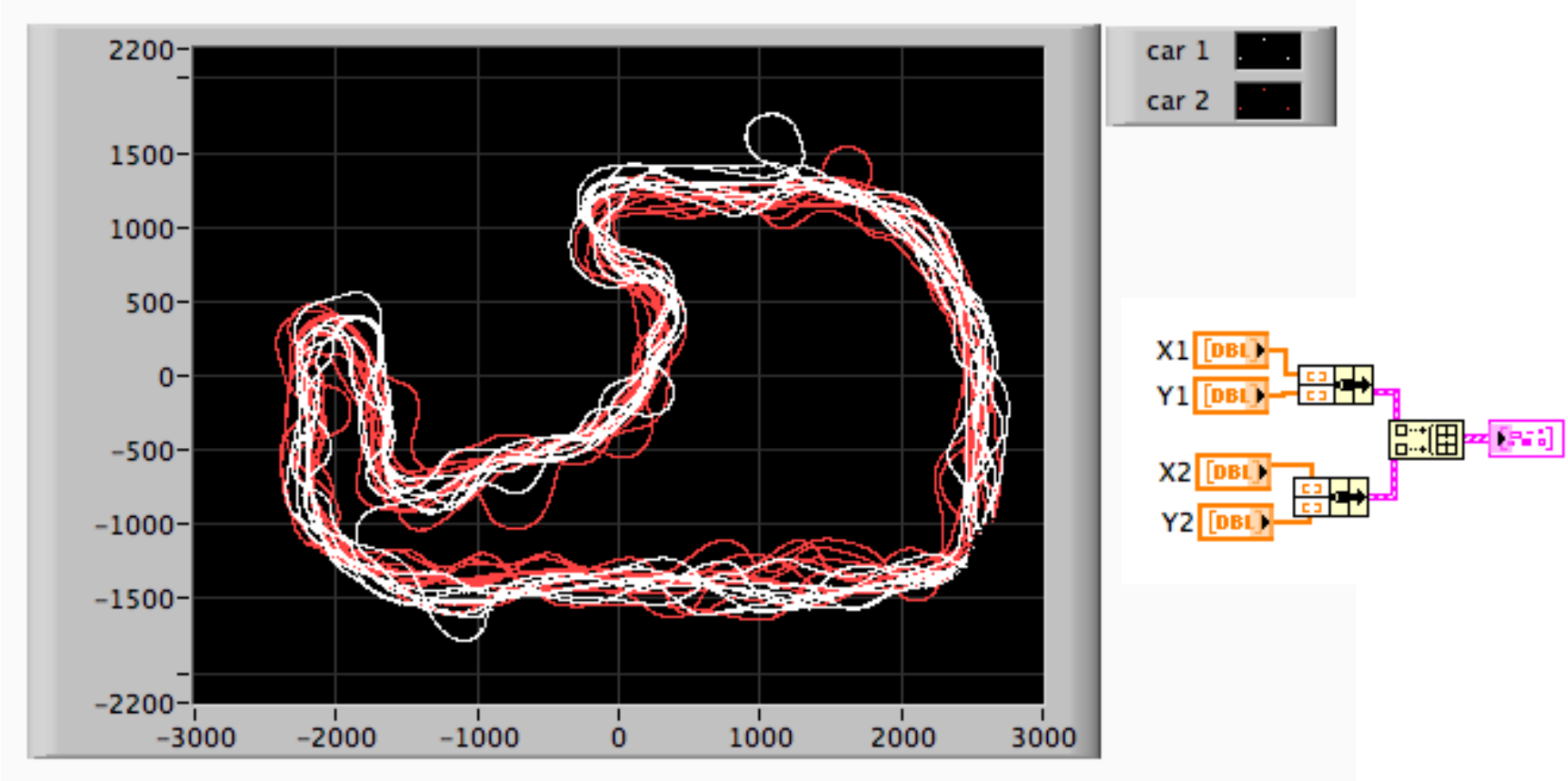




Les curseurs peuvent être libres (*free*) or attachés (*attached)* à un plot donné

X-Y Graph

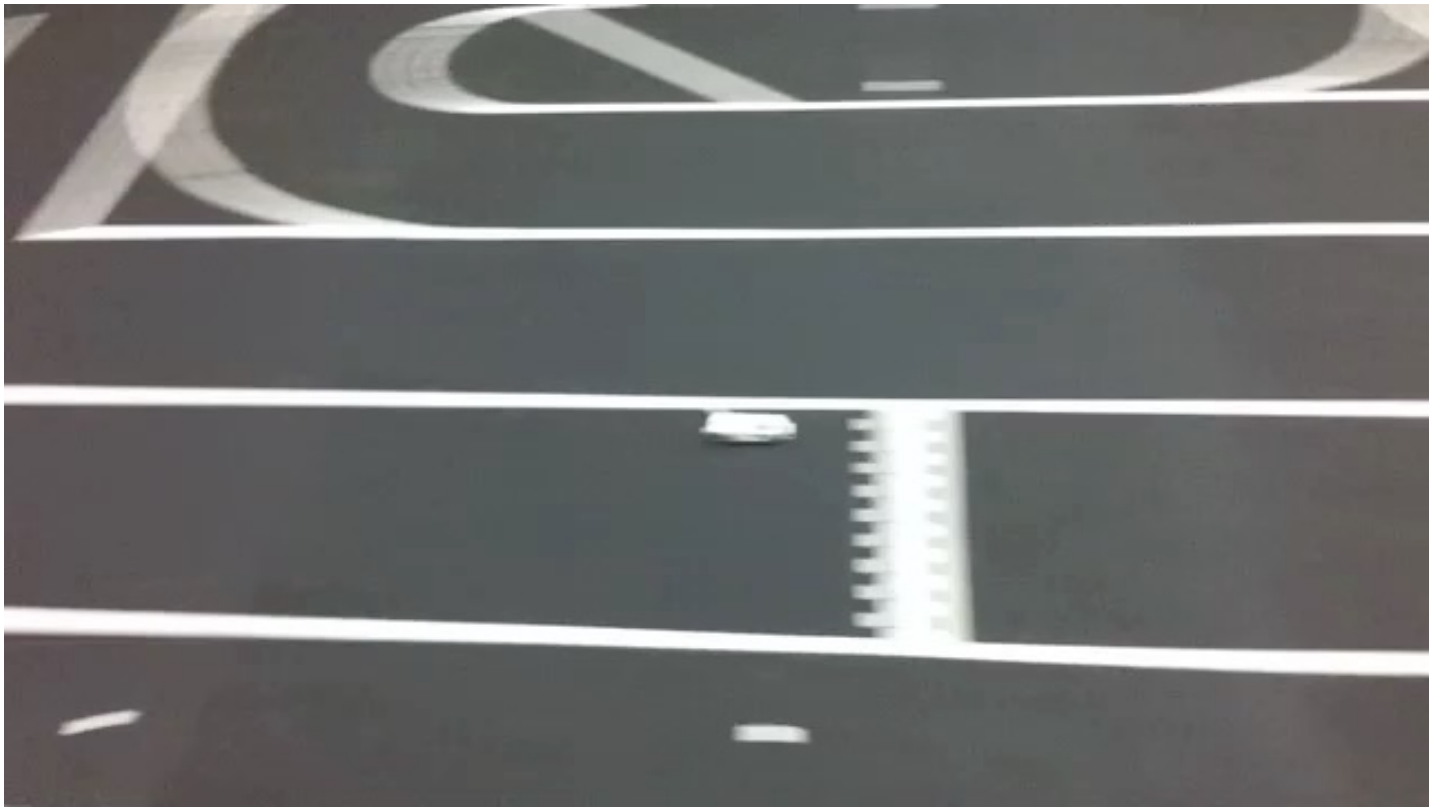
Similaires aux Graphs pour afficher une courbe paramétrique ☺



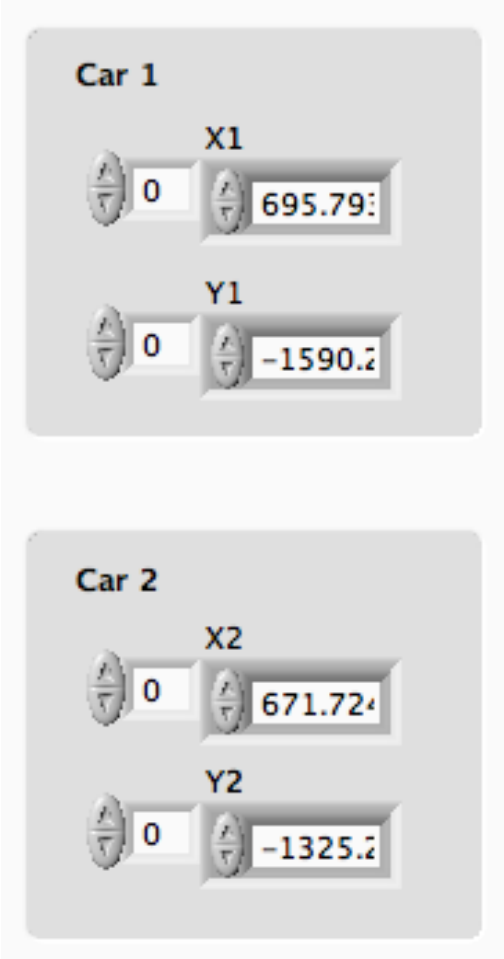
The famous first vortex recording, cars race

LA nano cars' race

- Acquisition, supervision & control via LabVIEW.
- Nano car: ~10 cm long, max speed ~4 m/s.
- 16 cameras give absolute car's positions and angle.

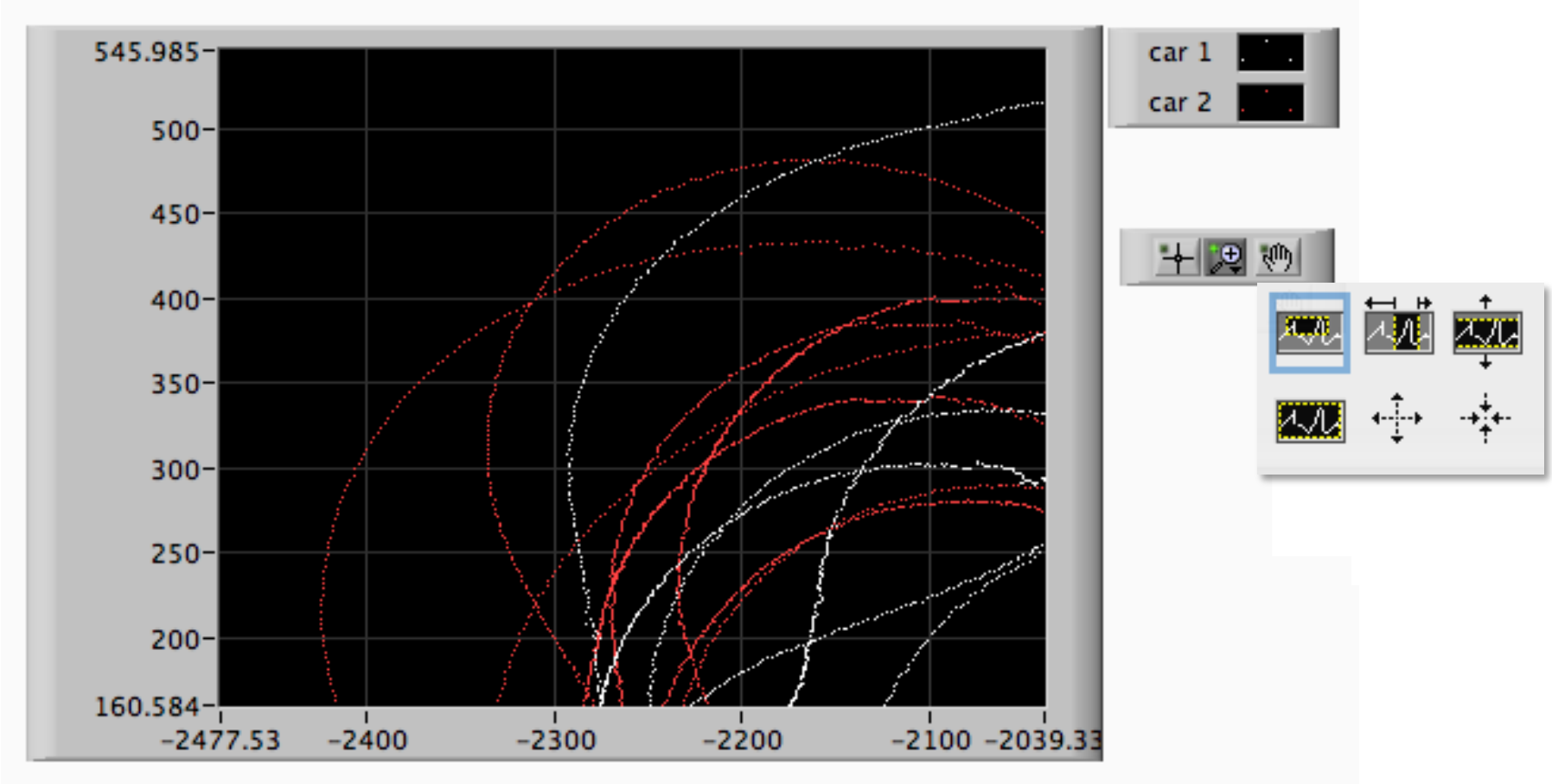


X-Y Graph – an example



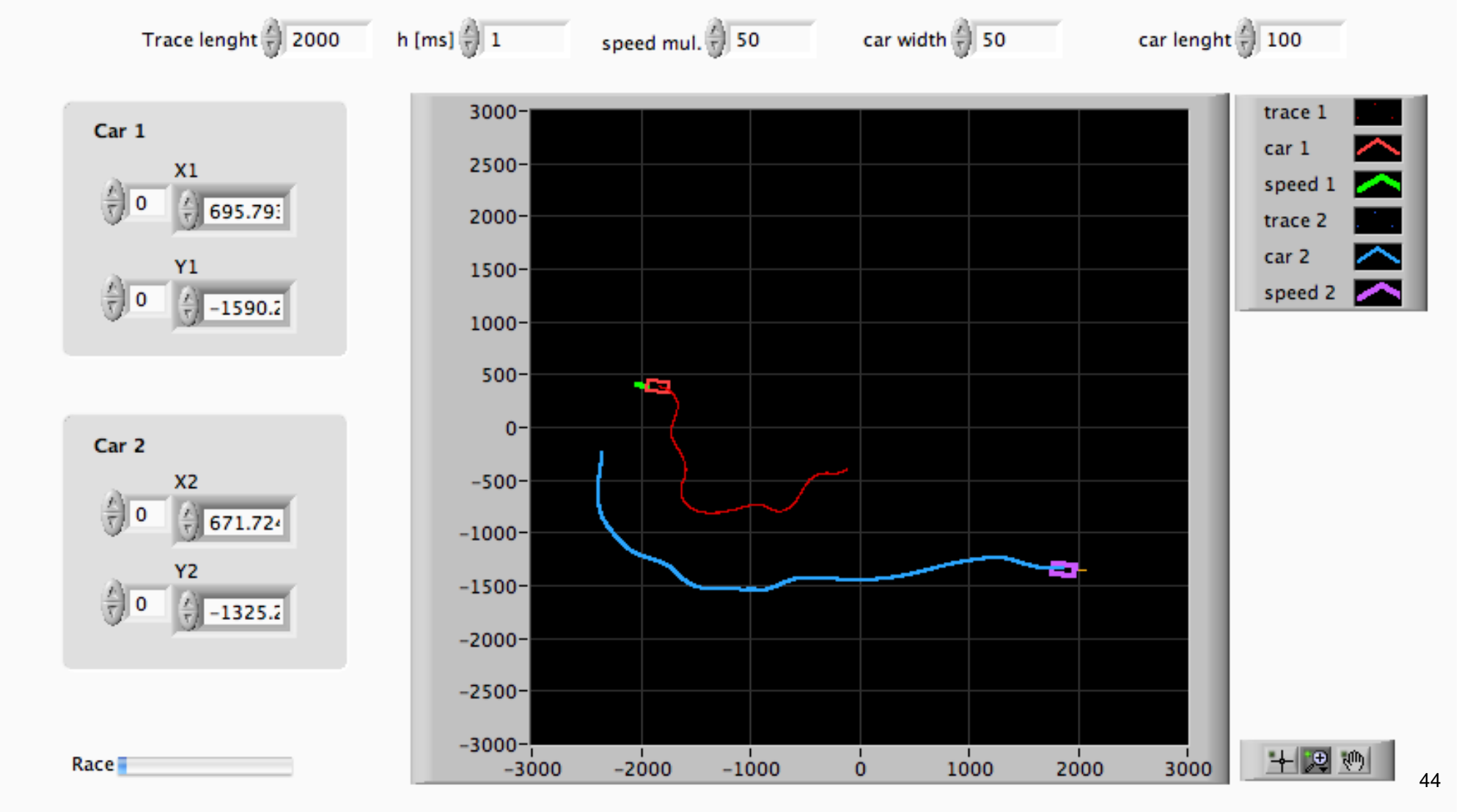
X-Y Graph

Similaires aux Graphs pour afficher une courbe paramétrique ☺



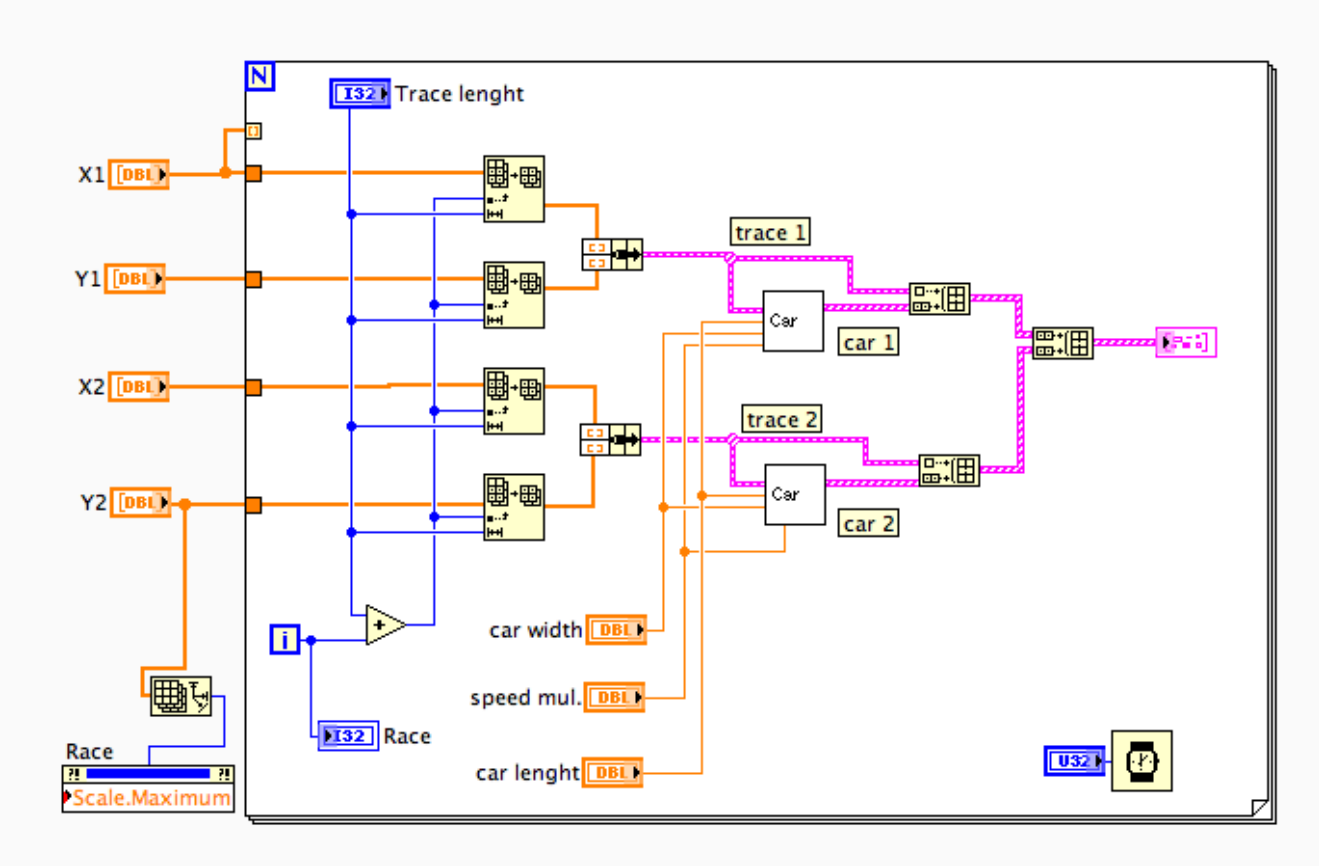
The famous first vortex recording, cars race

X-Y Graph – an example



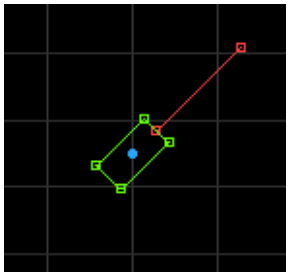
X-Y Graph – an example

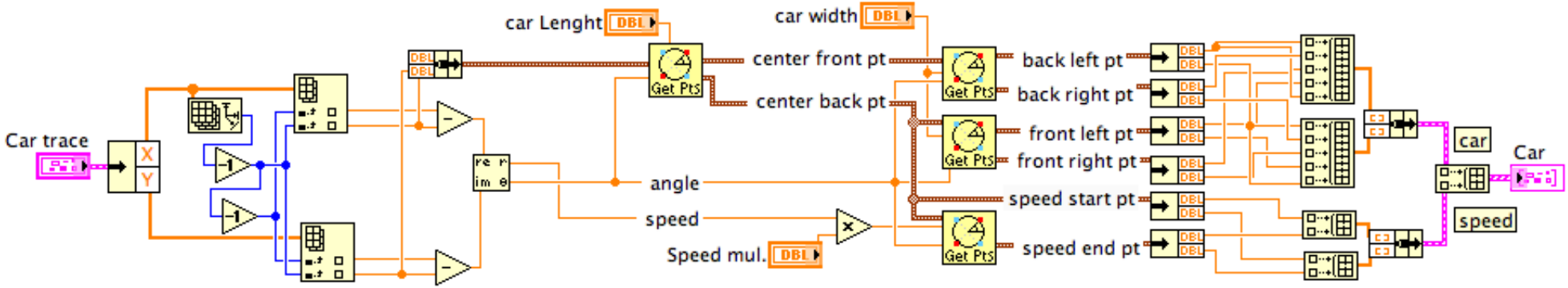
“draw” the cars and the tails



X-Y Graph – an example

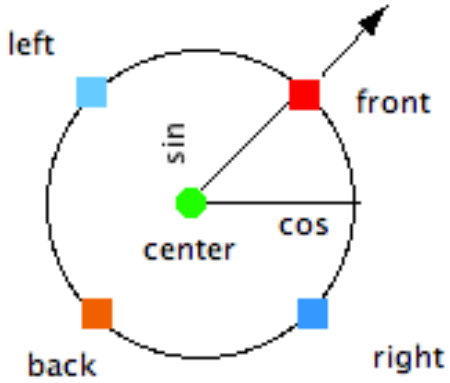
“draw” the car and the speed given 2 pts

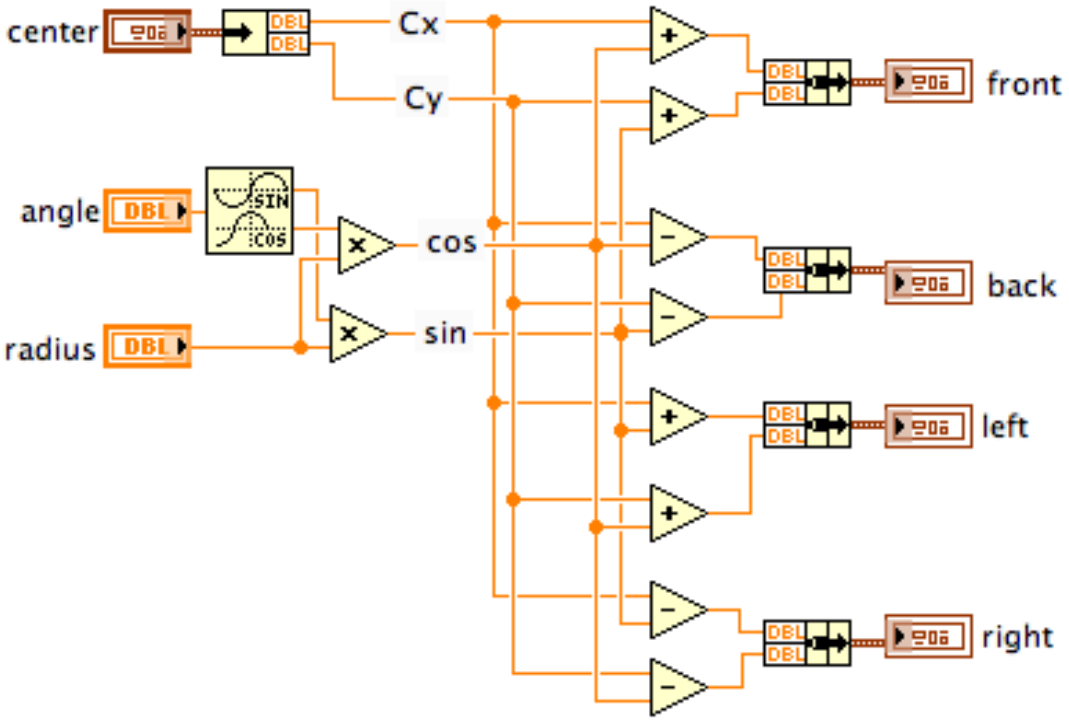




X-Y Graph – an example

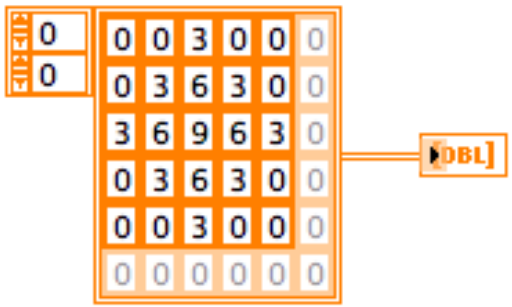
Compute the car's points, given angle and radius

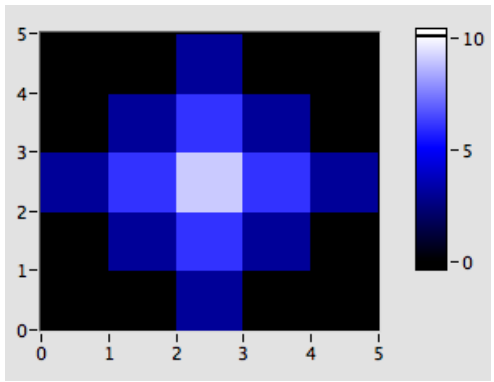




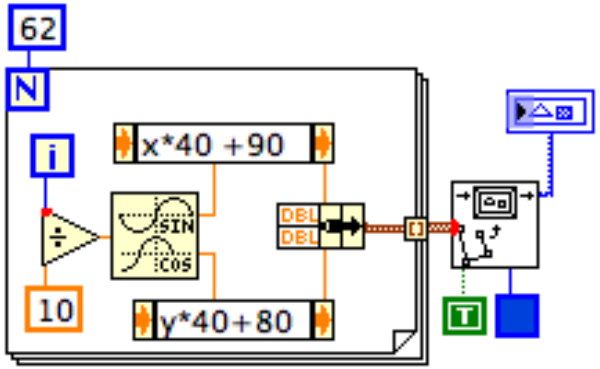
GUI - display data

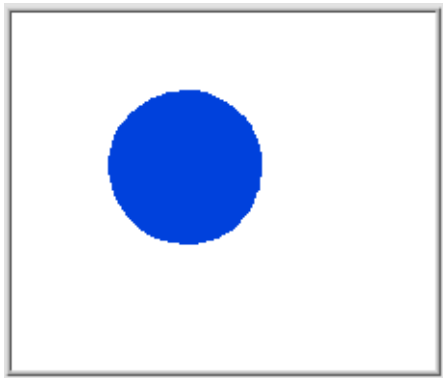
Intensity chart



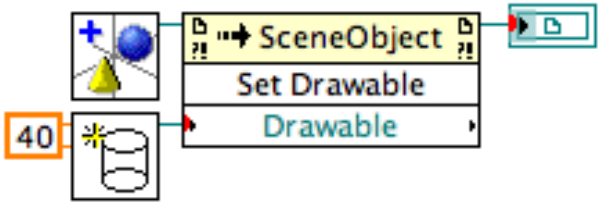


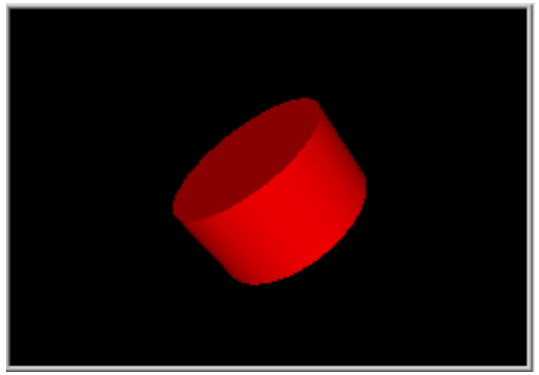
2D picture





3D picture



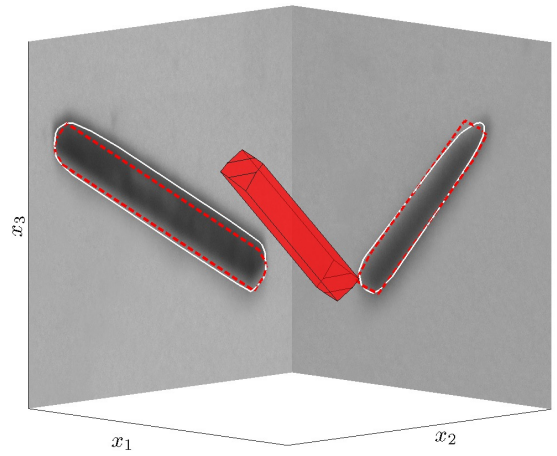


Demo

The famous Jean-Hubert's image selection problem

Staring:

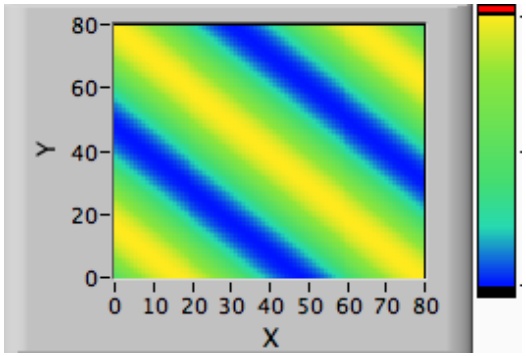
- *Displaying Image*
- *Image computation*
- *Graph*
- *Cursor*

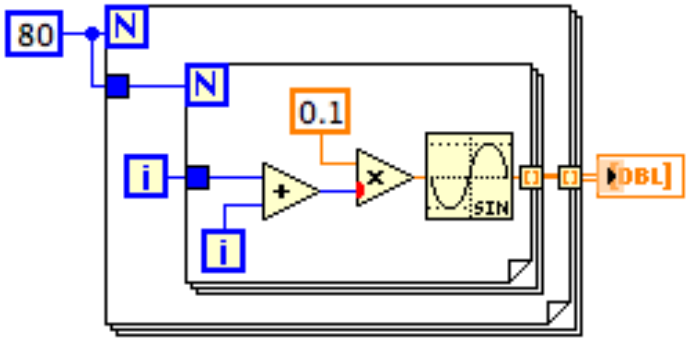


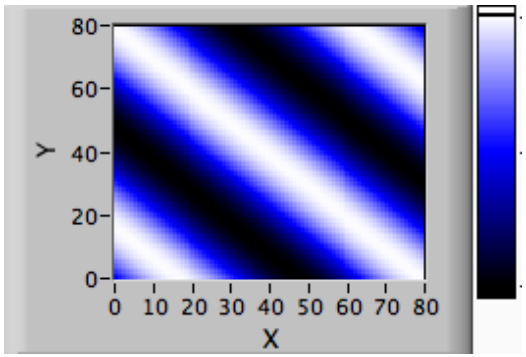
Intensity Graph

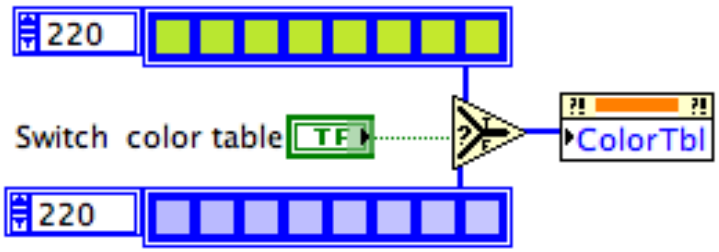
X-Y graph with Z as a color

User can switch Color Look Up Table





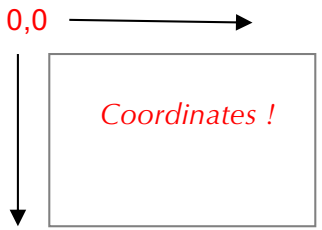




2D Picture

The best way to draw fully customized element in the front panel, can be tedious!

- Index image can be 1, 4, 8 bits/pixel with color table
- RGB image are 24 bits/pixel
- You can programmatically compose your picture
- You can programmatically access any pixel
- Many image primitives are provided, still tedious



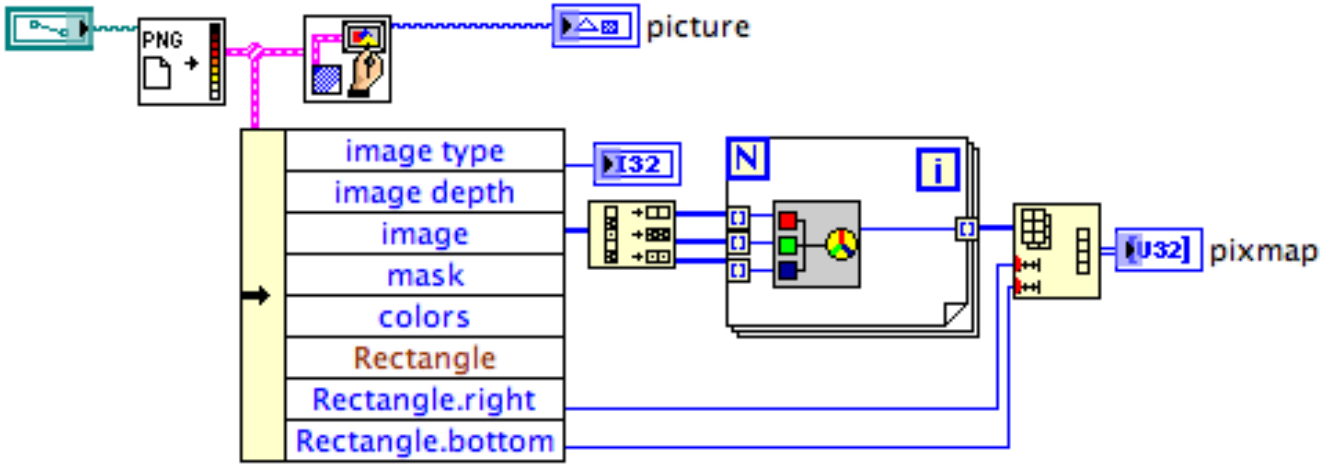
RGB (24bit) pixmap

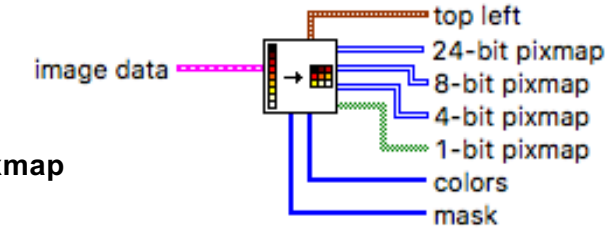
8bit image + gray CLUT

1bit image

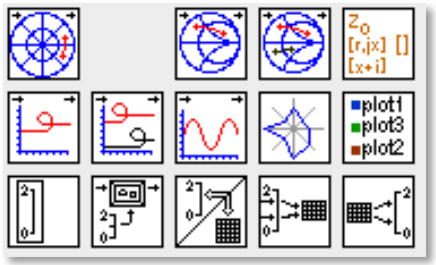
2D Picture

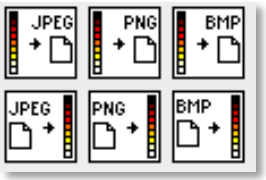
Many image primitives are provided, especially for chart/graph

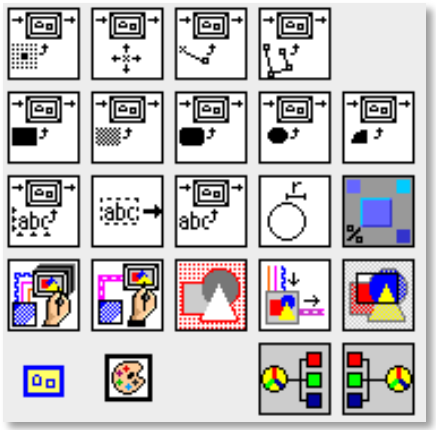




Unflatten pixmap

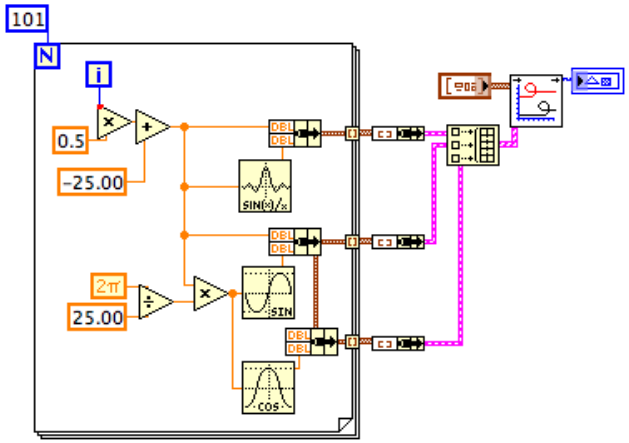


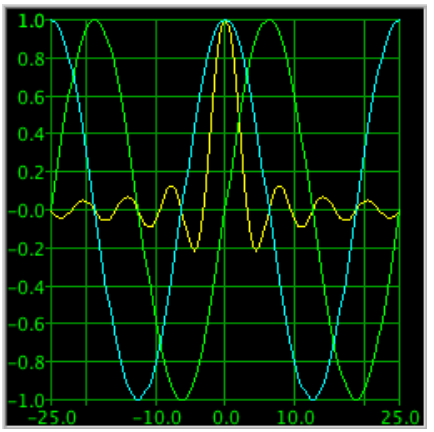


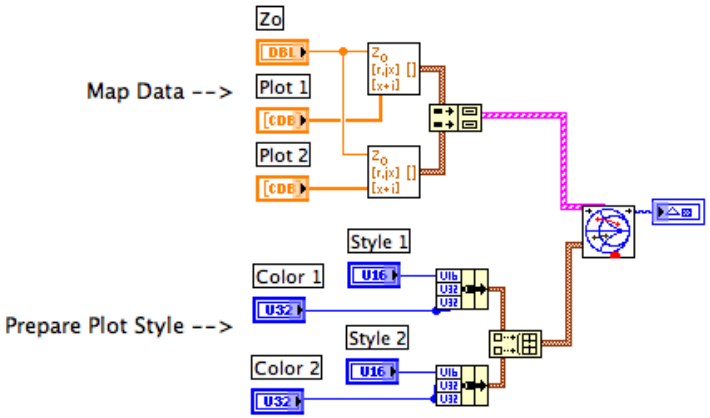


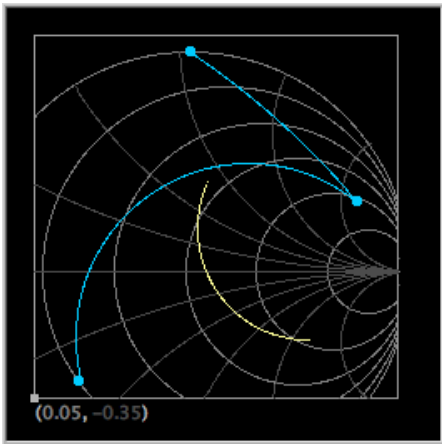
2D Picture

Chart/graph primitives



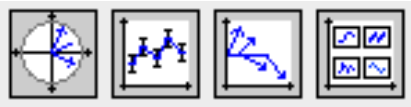


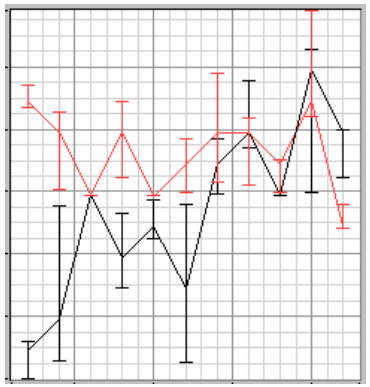


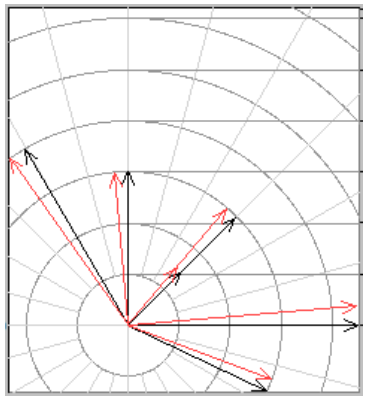


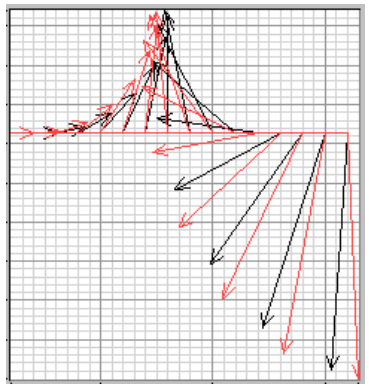
2D Picture

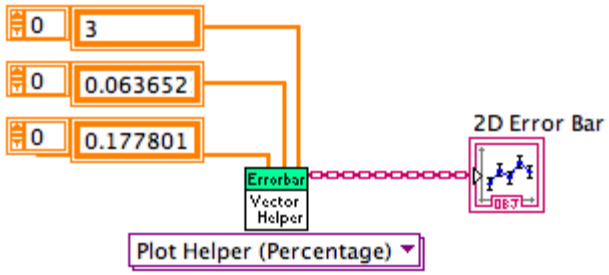
Object plot 2D primitives

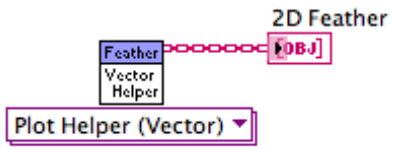


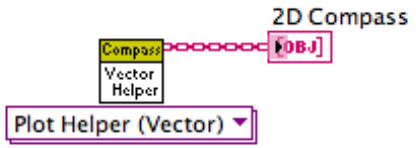








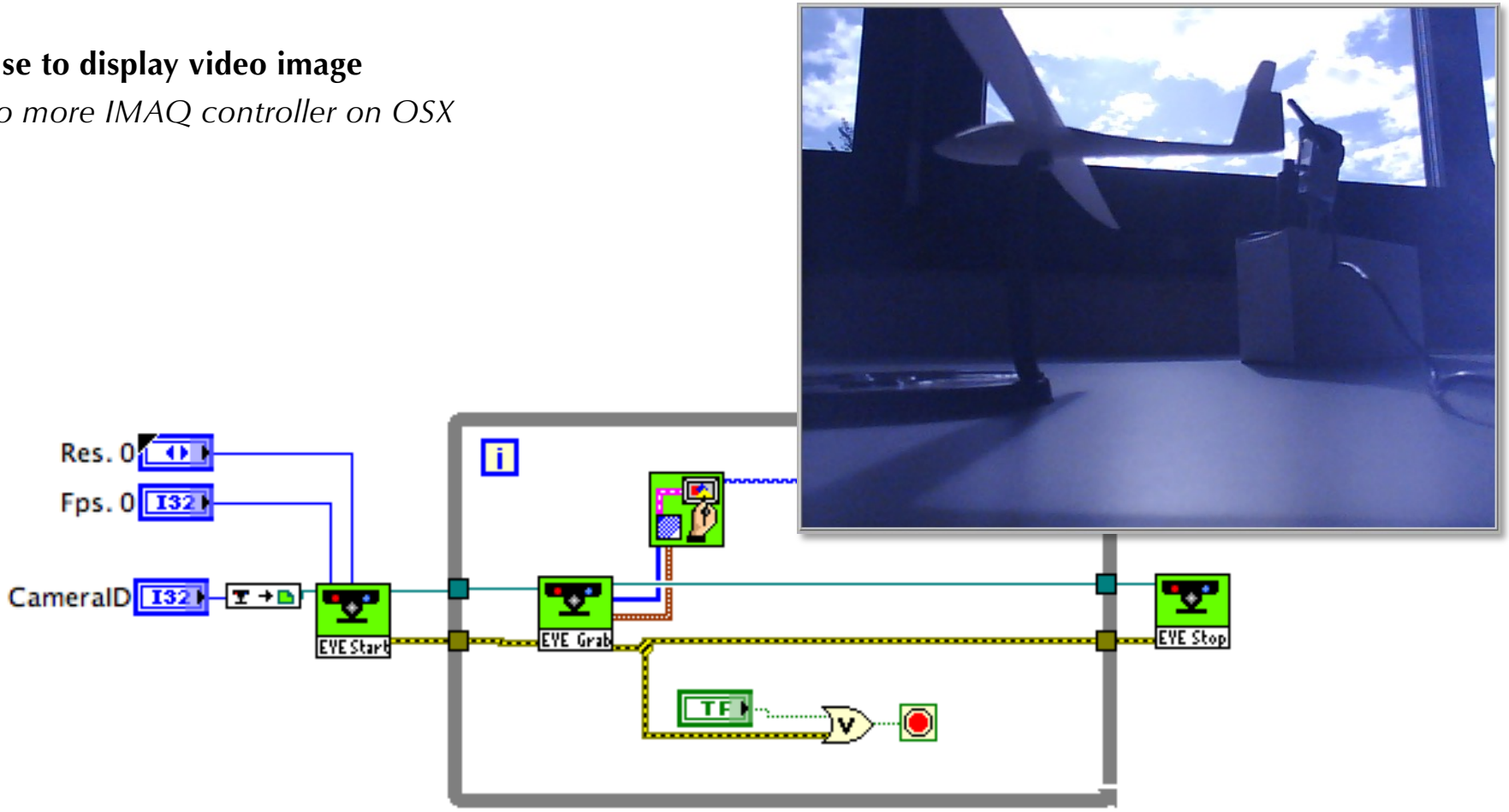




2D Picture

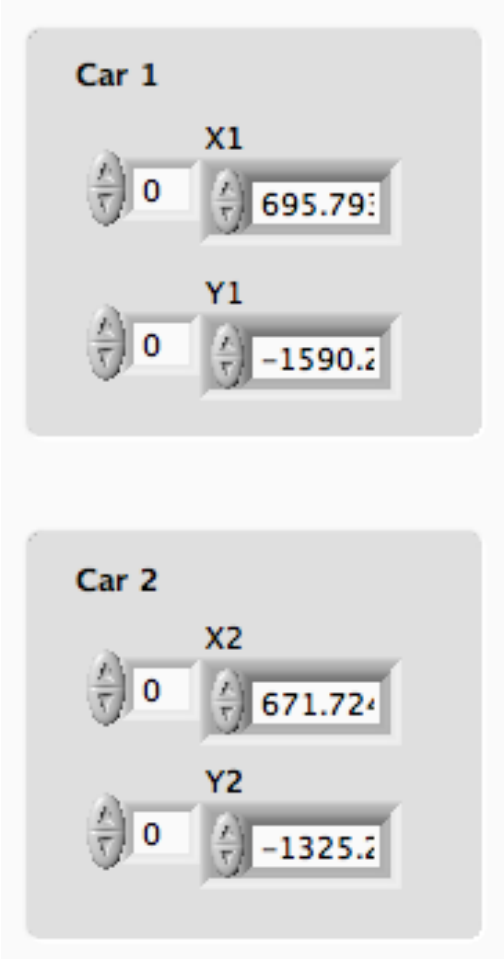
Use to display video image

no more IMAQ controller on OSX



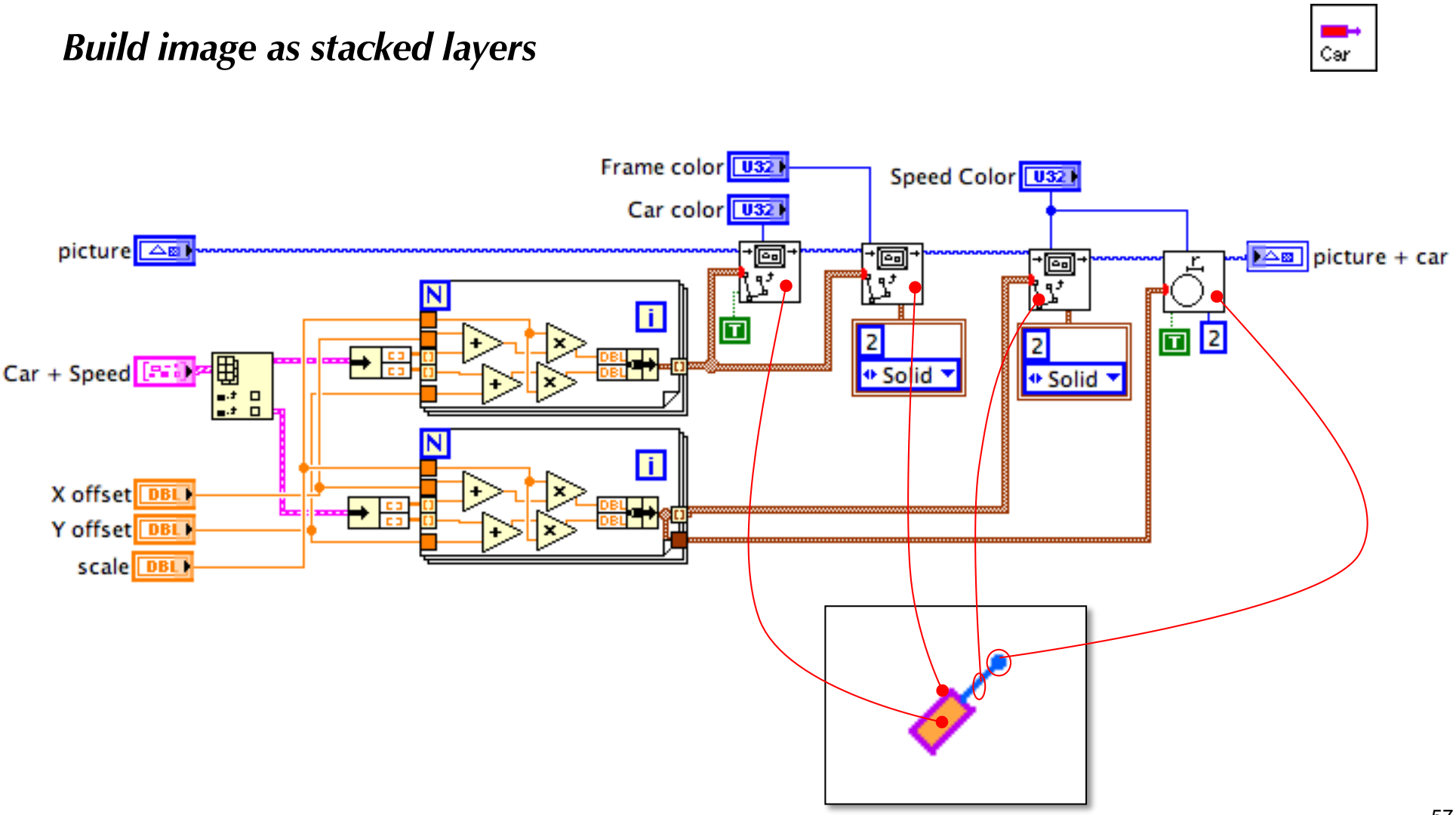
Use the LVsOCV or QTLib for additional video related functionalities on OSX!

Cars race revisited – 2D picture



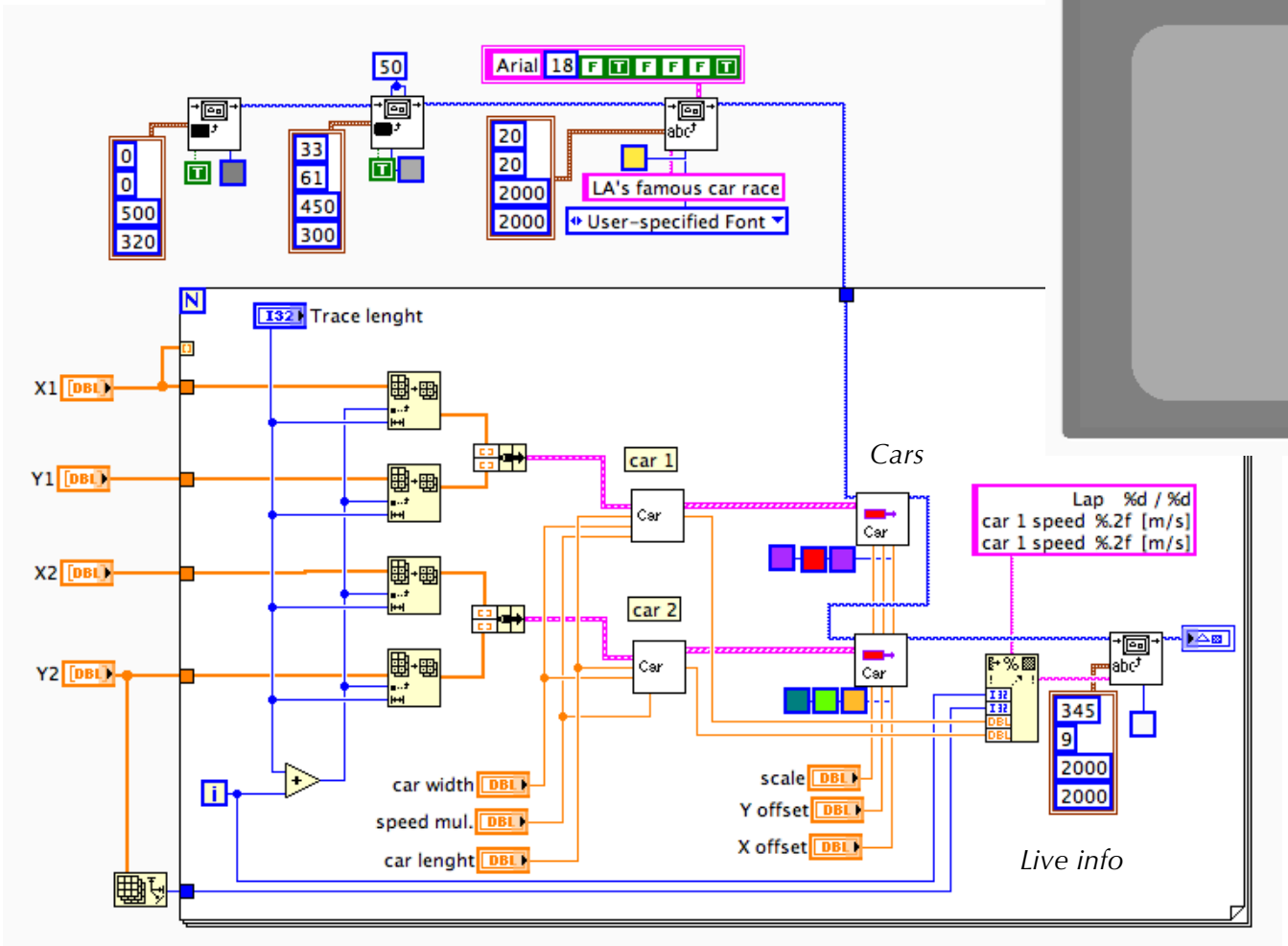
2D Picture

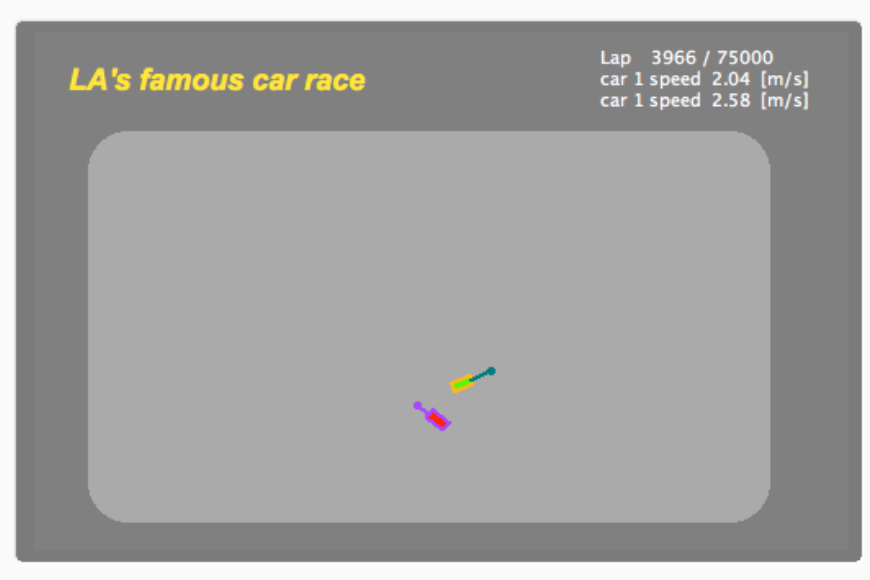
Build image as stacked layers



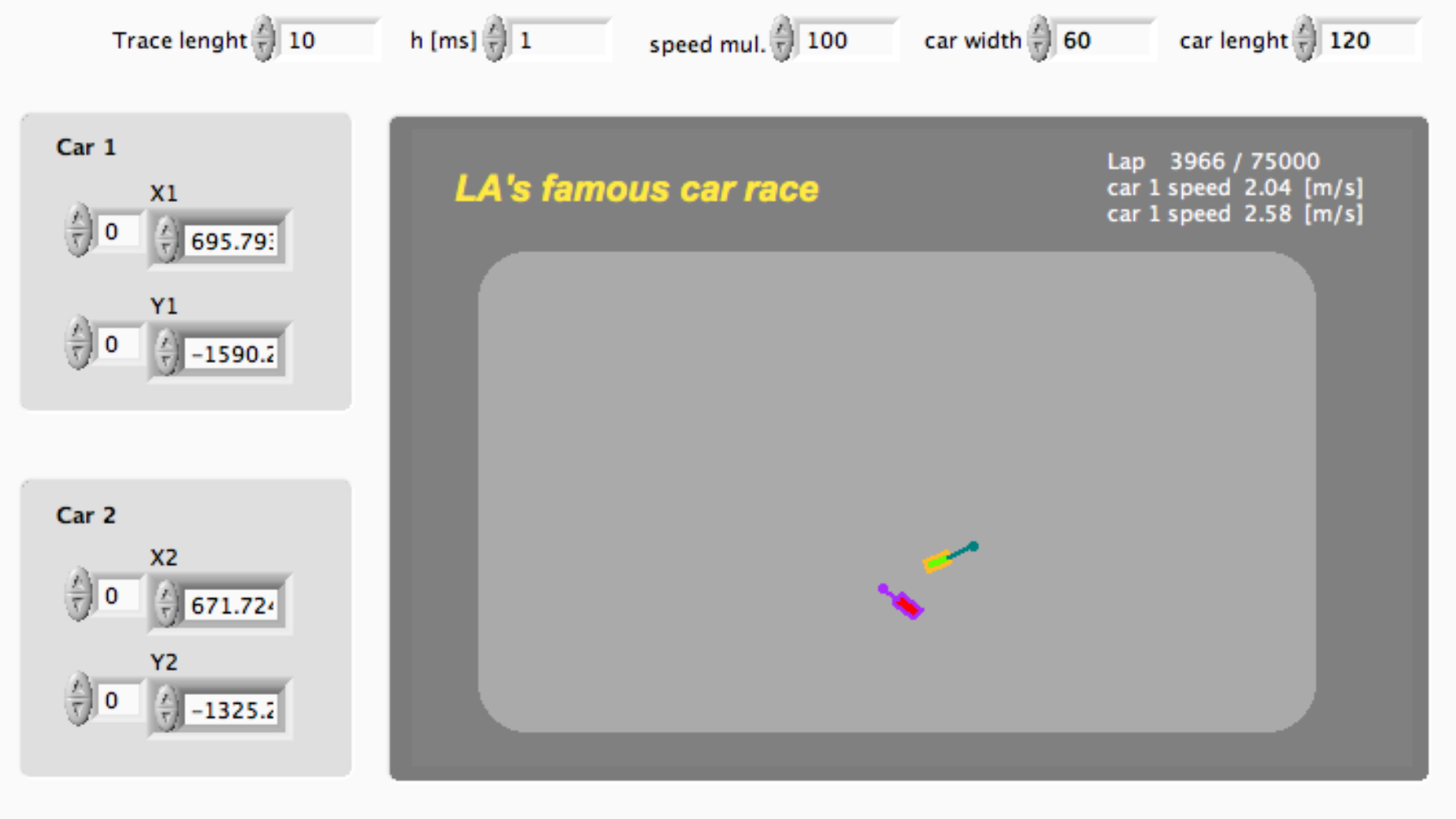
Cars race revisited

Background + Pub





Cars race revisited

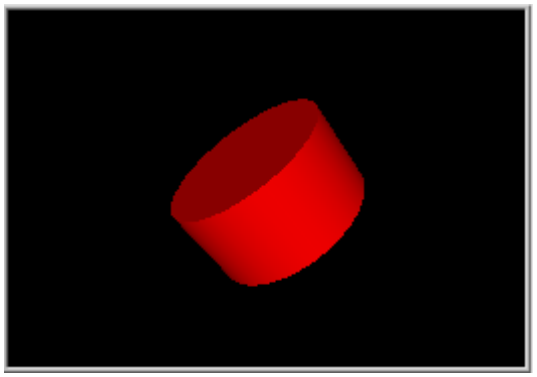


3D Picture

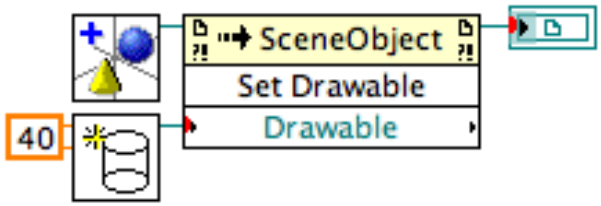
Similar to 2D pictures but in 3D can be tedious!

3D picture can be displayed in

- 3D picture indicator (scene)
- 3D graph
- External window *(much faster display)*

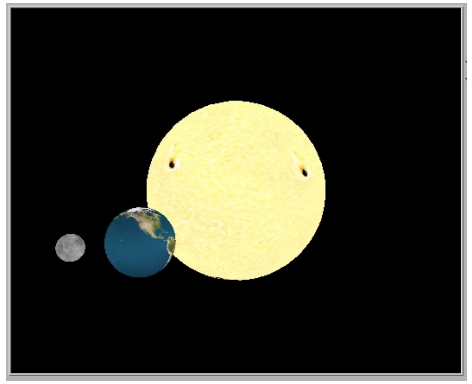


Build and animate your 3D world *à-la VRML*

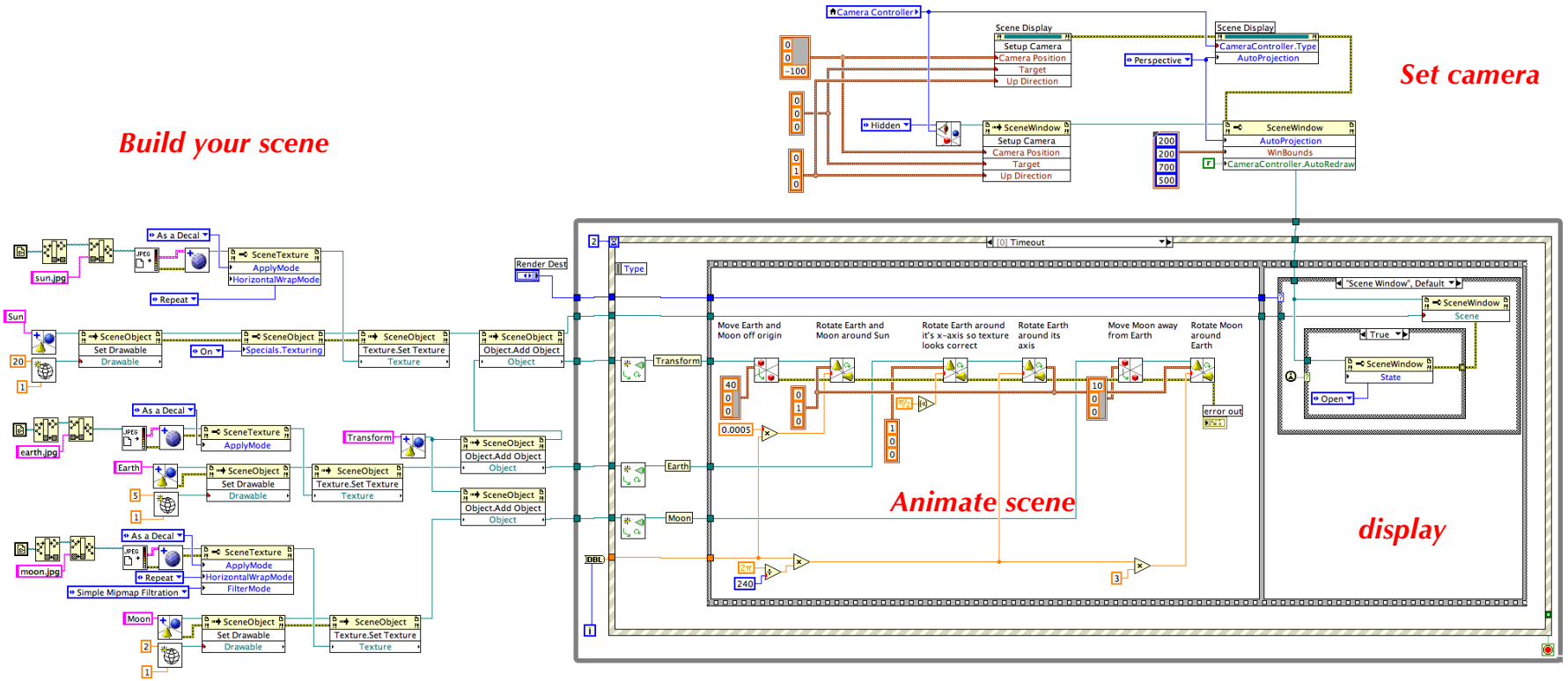


You can move the camera interactively or programmatically

3D Picture



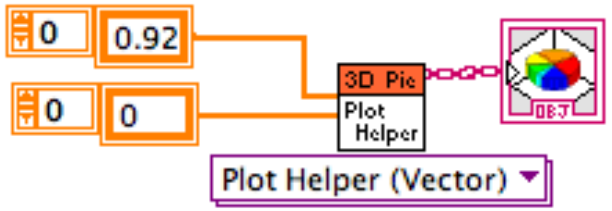
Ex: animated solar system

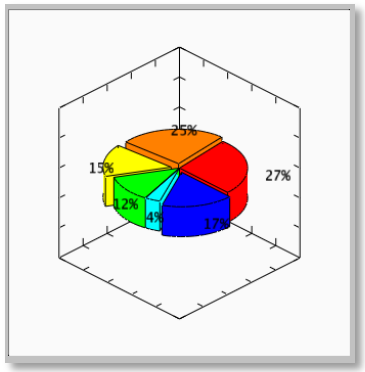


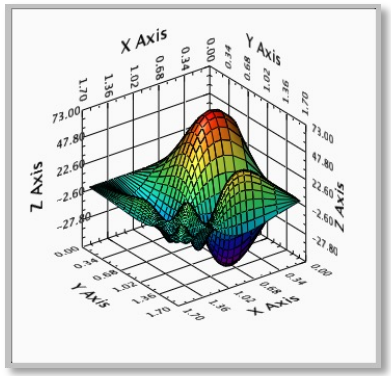
3D Graph

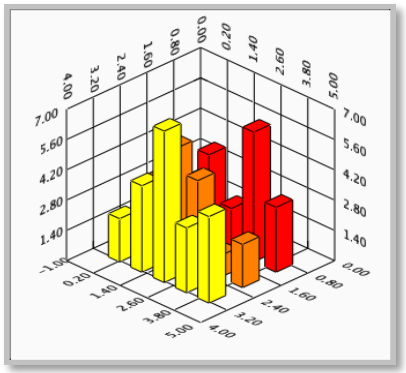
Replace the old ActiveX graphs

- Build with the 3D pictures in plain LV
- Many predefined graphs format
- When you drop the 3D graph control a Plot Helper vi is dropped as well
- Just need to feed data
- Can be animated same as 2D example

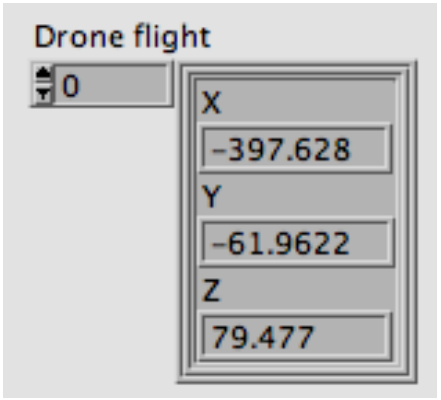


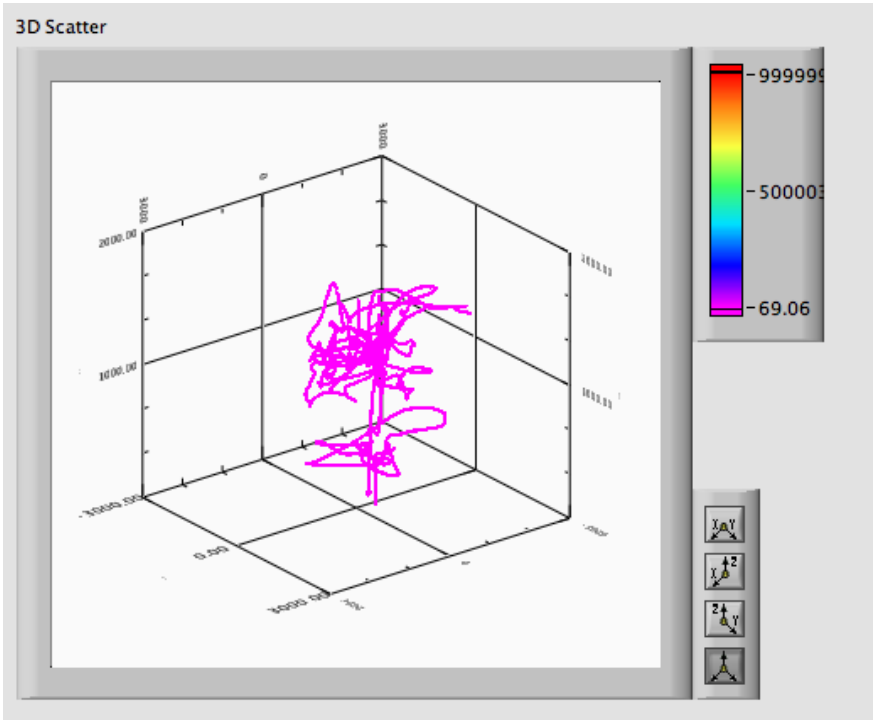






3D Graph

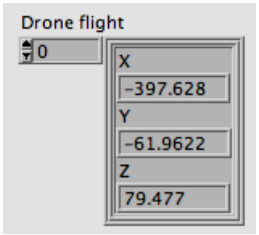


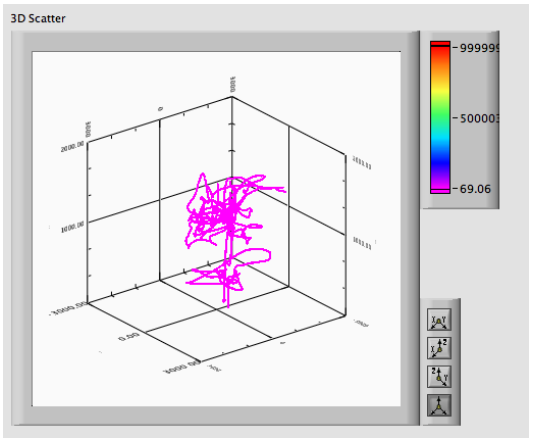


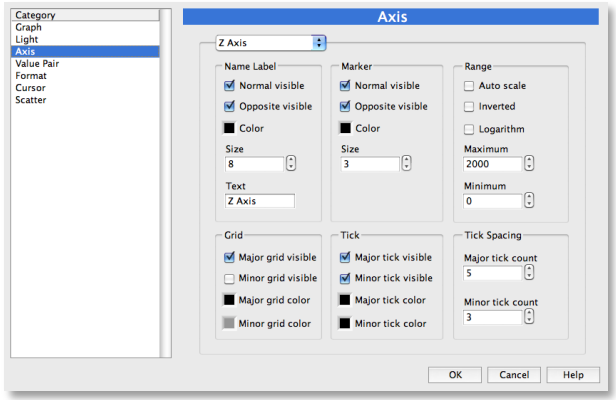
Ex: Drone flight

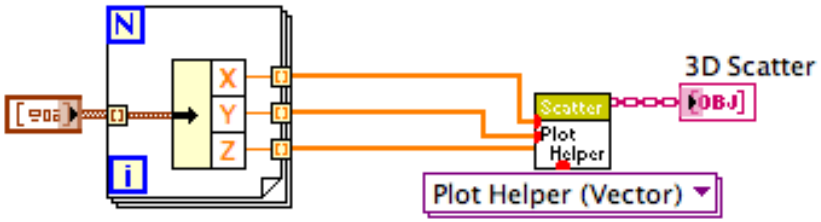
3D Graph

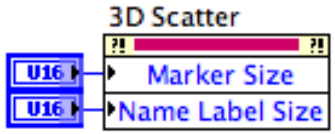
Ex: Drone flight – easy as pie!







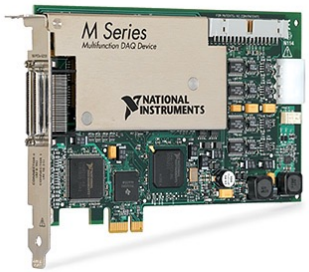




Mesure en moins de 10 slides

- LabVIEW a initialement été créé pour faire de la mesure, il a évolué en un environnement de développement complet
- LabVIEW peut accéder à un grand nombre d'appareils et d'interfaces de mesures (au travers de drivers)
- La puissance de LabVIEW vient du fait qu'il propose une interface standard qui cache la partie compliquée de l'accès au matériel

GPIB instrument



DAQ card

USB device

cRio controller

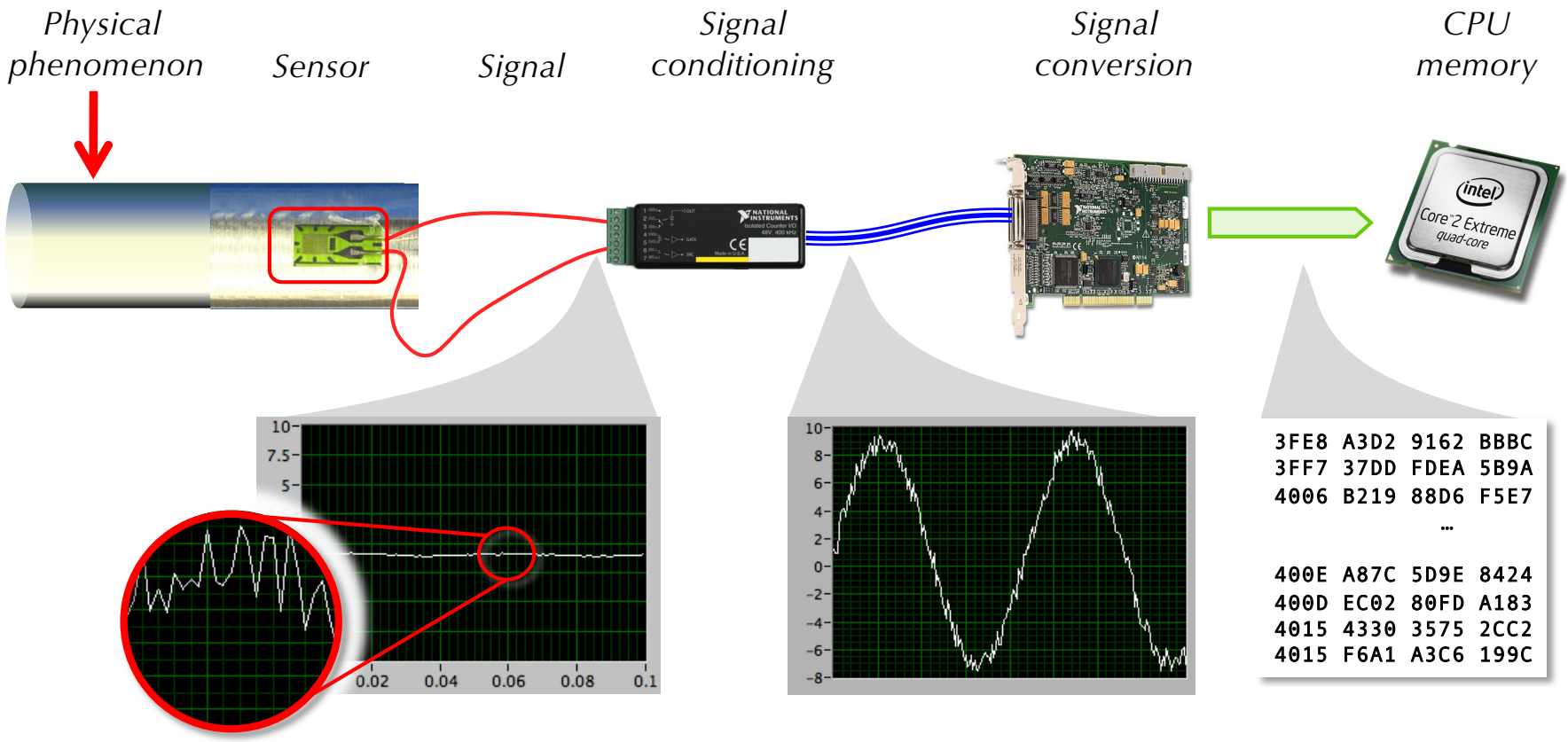
Choix d'une solution DAQ

Questions pour choisir une solution/hardware

- Nombre d'entrée et de sortie
- Types (digital/analogique) des entrées et des sorties
- Gamme d'utilisation des entrées et des sorties (volt)
- Résolution des entrées et des sorties (bits)
- Fréquence des entrées et des sorties (Hertz)
- Mesure (plusieurs points à la fois) ou control (un point après l'autre)
- Conditionnement du signal

Du capteur au processeur

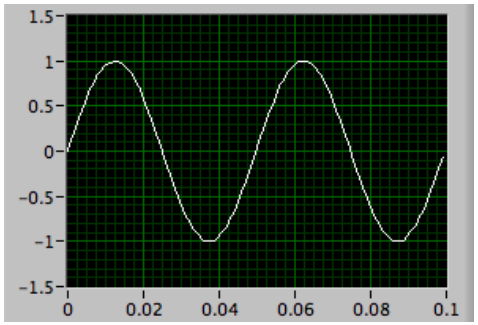
Le **phénomène physique** est acquis par le **capteur**, puis il est transmis au module de **conditionnement du signal** afin d'être amplifié. Il est finalement converti en un nombre par le **convertisseur analogique/digital** situé sur la carte d'acquisition



Capteur – types de signal

Analogue

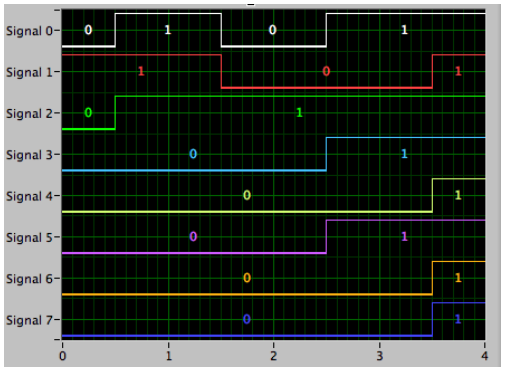
- Forme
- Amplitude
- Fréquence



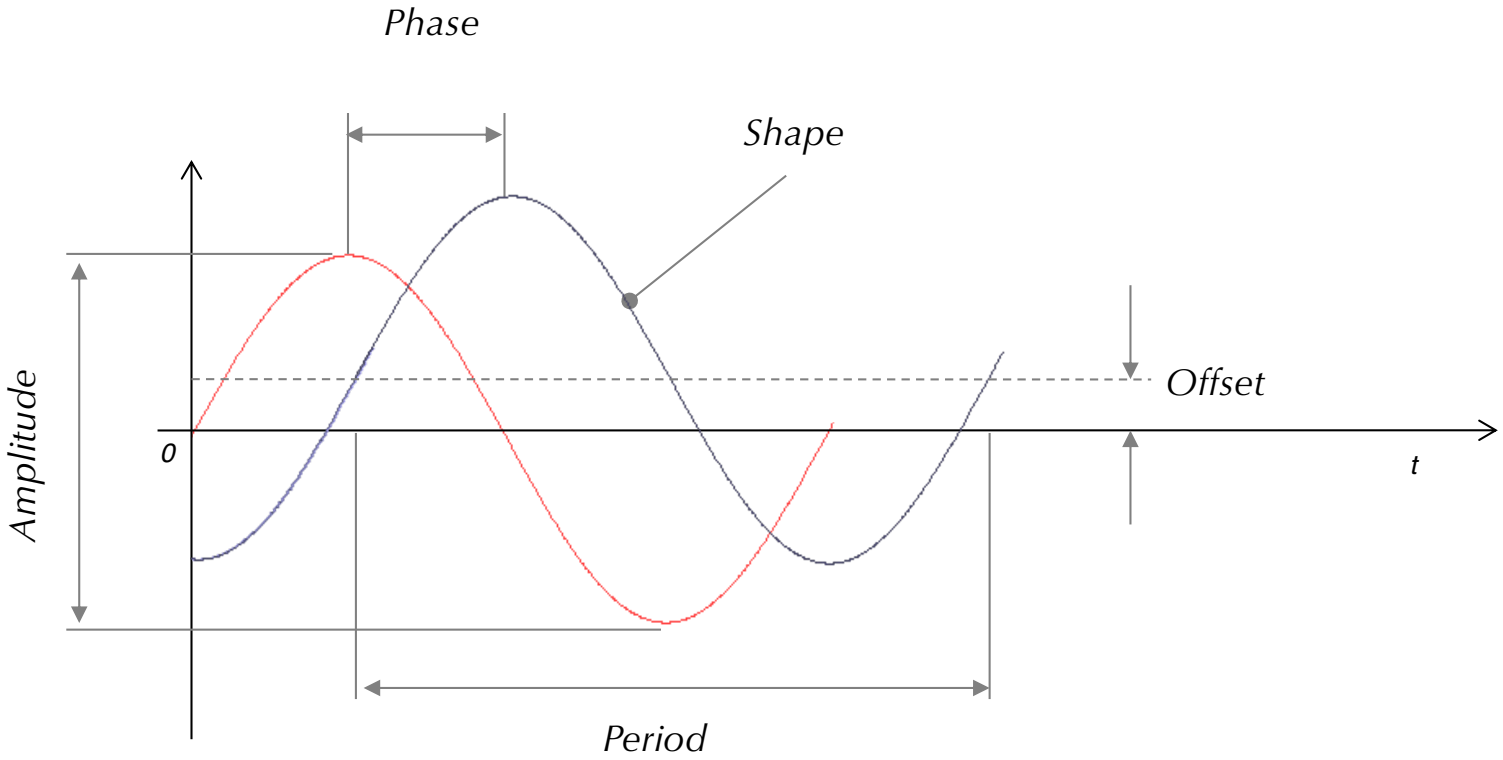
Digitale

- Etat
- Période

Les signaux digitaux peuvent être traités comme des signaux analogues



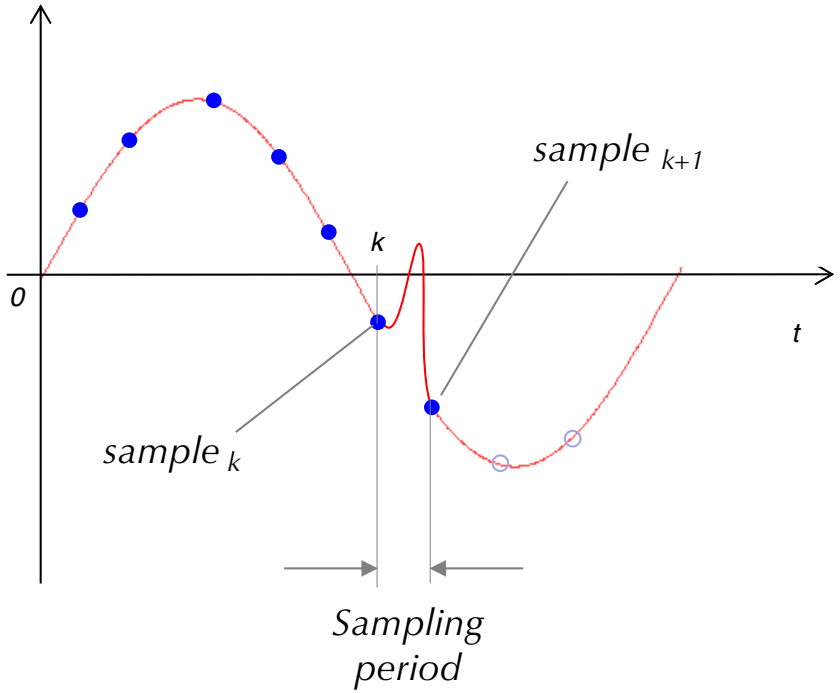
Signal



Frequency = 1/period

Conversion A->D

Convertir un signal analogique en une représentation digitale se fait en deux étapes. La première étape est de prendre un échantillon du signal analogique continu (*sample and hold circuit*). La deuxième étape est de convertir le signal analogue "mémorisé" en un nombre à l'aide du convertisseur A/D). Ce processus est répété après un certain temps appelé période d'échantillonage *sampling period* (or frequency).

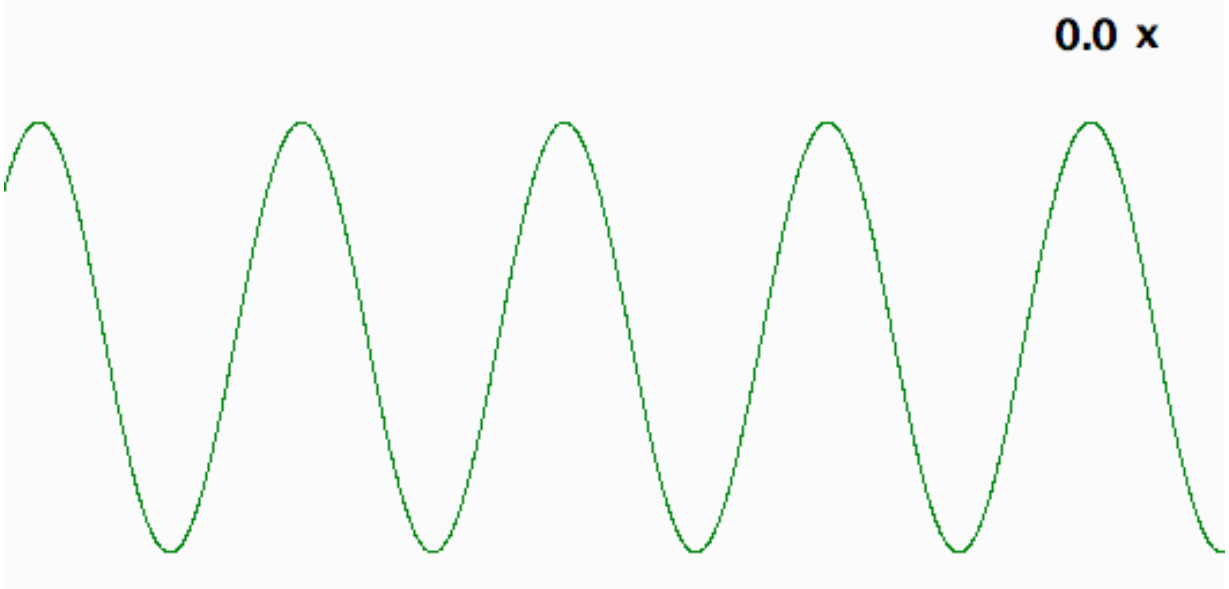


Le processus d'échantillonage est aveugle entre les échantillons.
La période d'échantillonage est généralement constante entre les échantillons.

Echantillonnage du signal

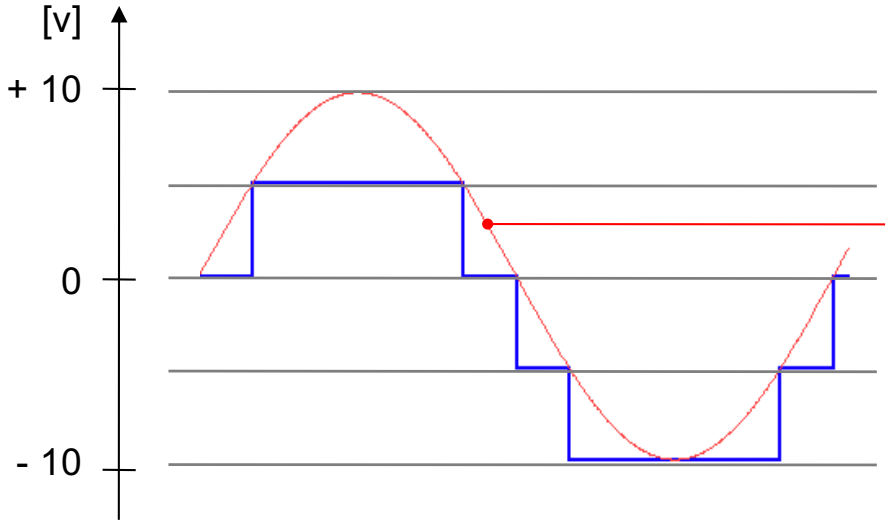
La fréquence à laquelle le signal analogique est échantillonné doit être en pratique 10 fois (2 x en théorie) plus rapide que la fréquence maximum que l'on veut observer. Sinon le signal peut ne pas être reconstruit correctement

Ex. Pour mesurer du 220v à 50Hz, la fréquence d'échantillonnage devrait être d'environs 500Hz.

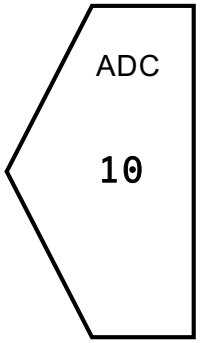


Conversion A/D

Le convertisseur AD (ADC) transforme le signal analogique en une représentation digitale. La résolution du convertisseur défini le nombre possible de valeurs que peut avoir le signal de sortie



11
10
01
00



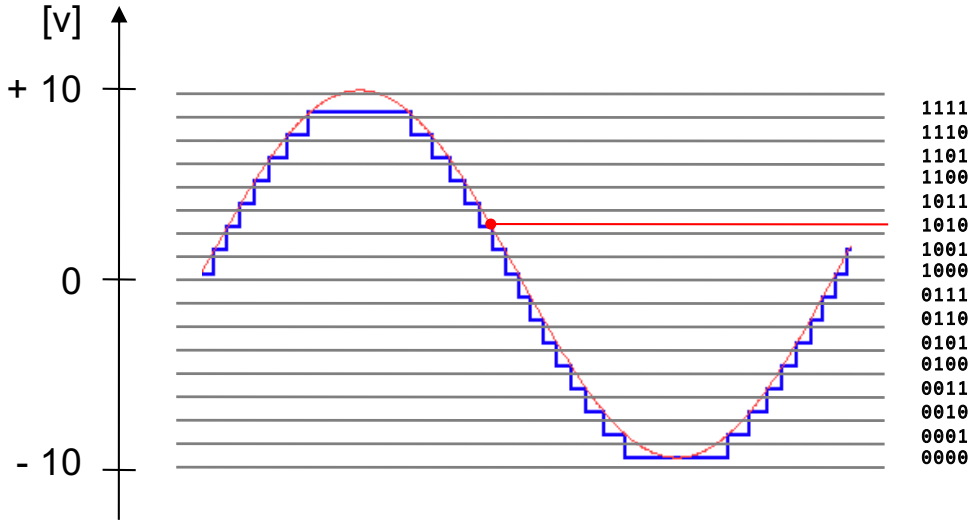
$$10_b \frac{10-(-10)}{2^2} - 10 =$$

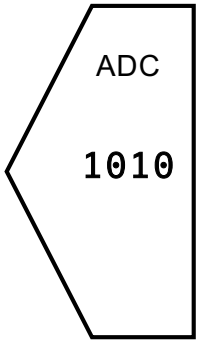
$$(2*5) - 10 = 0[v]$$

2 bits -> 2^2 = 4 levels

Conversion A/D

Le convertisseur AD (ADC) transforme le signal analogique en une représentation digitale. La résolution du convertisseur défini le nombre possible de valeurs que peut avoir le signal de sortie





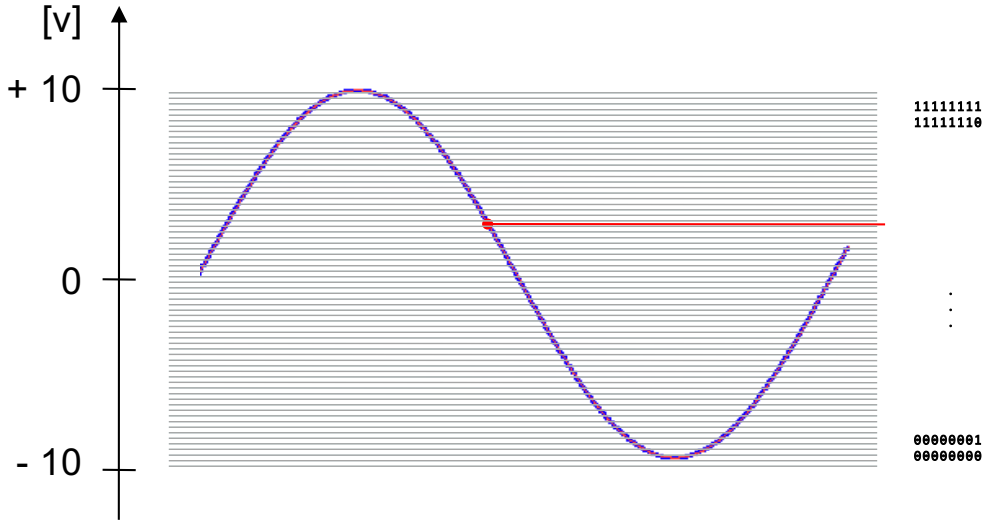
$$1010_b \frac{10-(-10)}{2^4} - 10 =$$

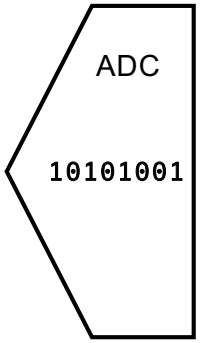
$$(10 * 1.25) - 10 = 2.5[\text{V}]$$

4 bits -> 2^4 = 16 levels

Conversion A/D

Le convertisseur AD (ADC) transforme le signal analogique en une représentation digitale. La résolution du convertisseur défini le nombre possible de valeurs que peut avoir le signal de sortie





$$10101001_b \frac{10-(-10)}{2^8} - 10 =$$

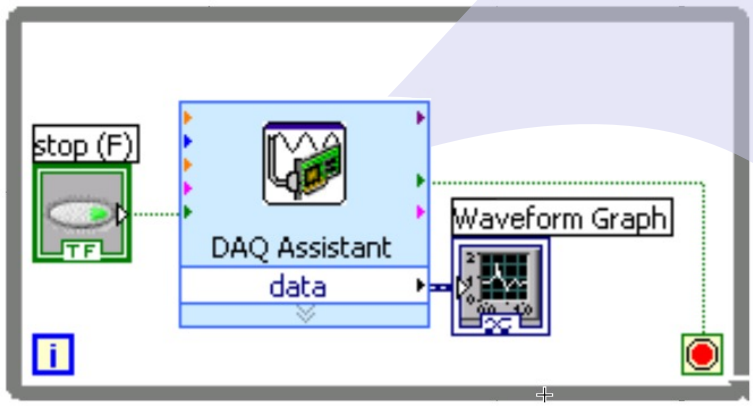
$$(169 * 0.078125) - 10 = 3.20[\text{V}]$$

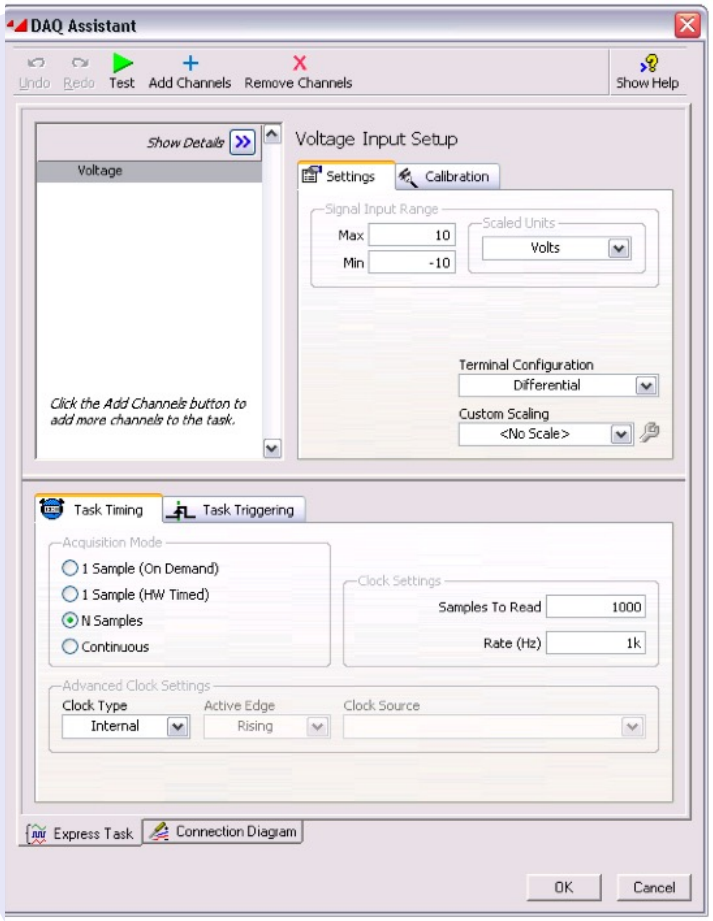
8 bits -> 2^8 = 256 levels

Windows – MAX & DAQ assistant

Configure tasks with
Measurement & **A**utomation
Explorer (MAX)

Or use DAQ assistant Express VI

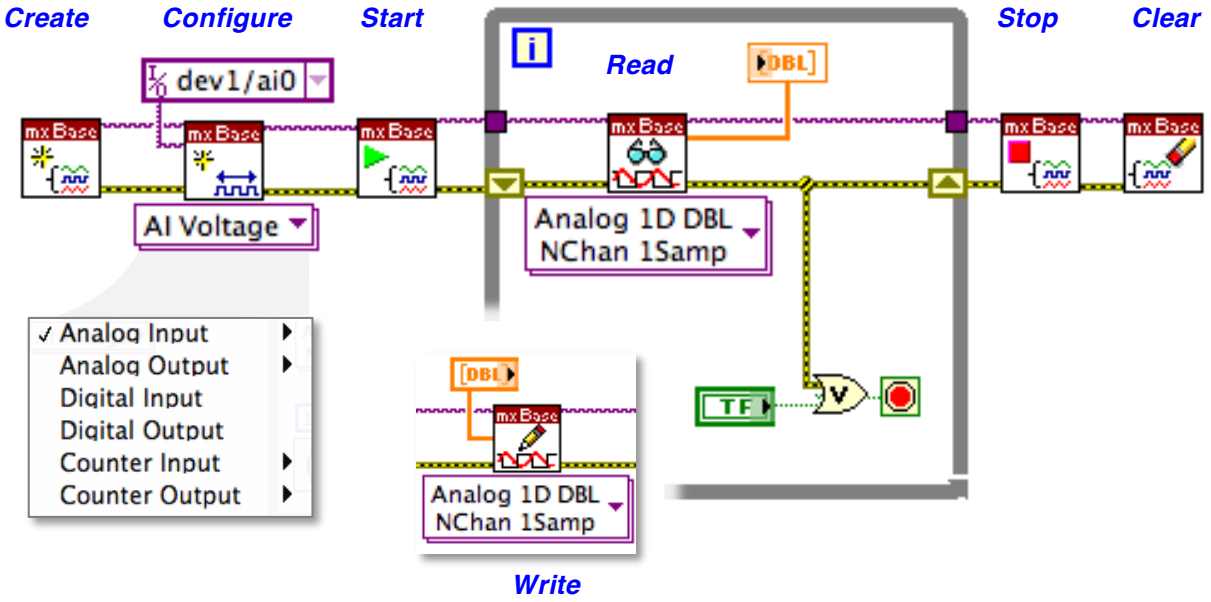




Data acquisition (DAQ)

Les opérations d'entrée/sortie LabVIEW sont basées sur des taches. Une tache représente **une** opération sur les entrées ou les sorties. Les étapes suivantes sont nécessaires:

- Create
- Configure
- Start
- Read/write
- Stop
- Clear

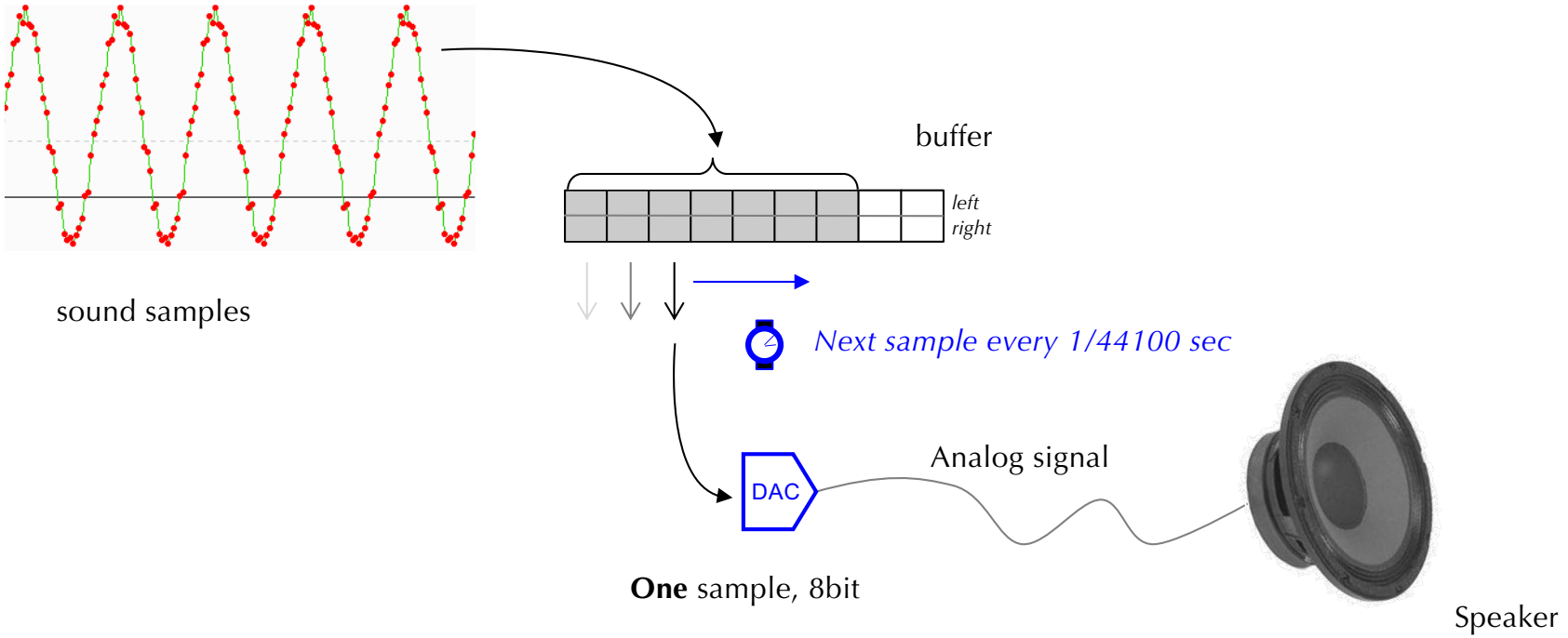


Slides suplémentaires

Playing a sound

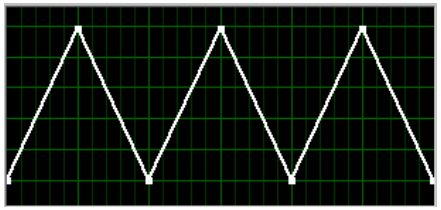
The sound DAs converter has some specificities:

- the output rates is fixed (11025, 22050 and 44100 S/s)
- the DA sample resolution can be set to 8 or 16 bit
- two or more channels (mono, stereo)
- the converter(s) is/are connected to speaker(s)



Playing a sound

Outputting a sinusoidal at a given frequency given the output rate of 44100 S/s.
Maximum theoretical playable frequency 44100/2 = 22050 Hz.
Human ear: baby ~20hz – 20khz, after 25y.o. ~16khz, computer speaker : 200Hz -12khz

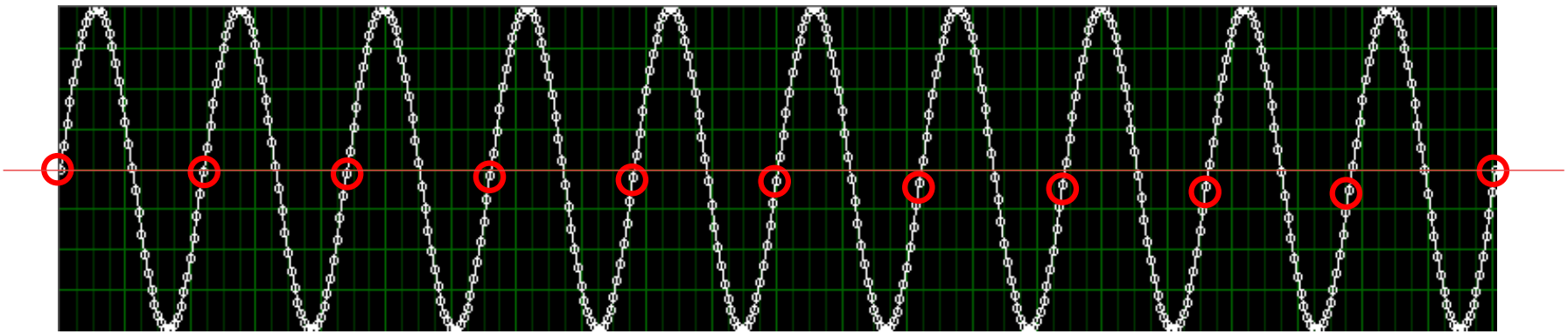


Example:

Desired output frequency: 1000Hz -> 1000 sinusoidal per second
Nbr samples per sinusoidal for 1000Hz -> 44100/1000 -> 44.1 samples per sinusoidal

How do we deal with the ".1" in 44.1 ?

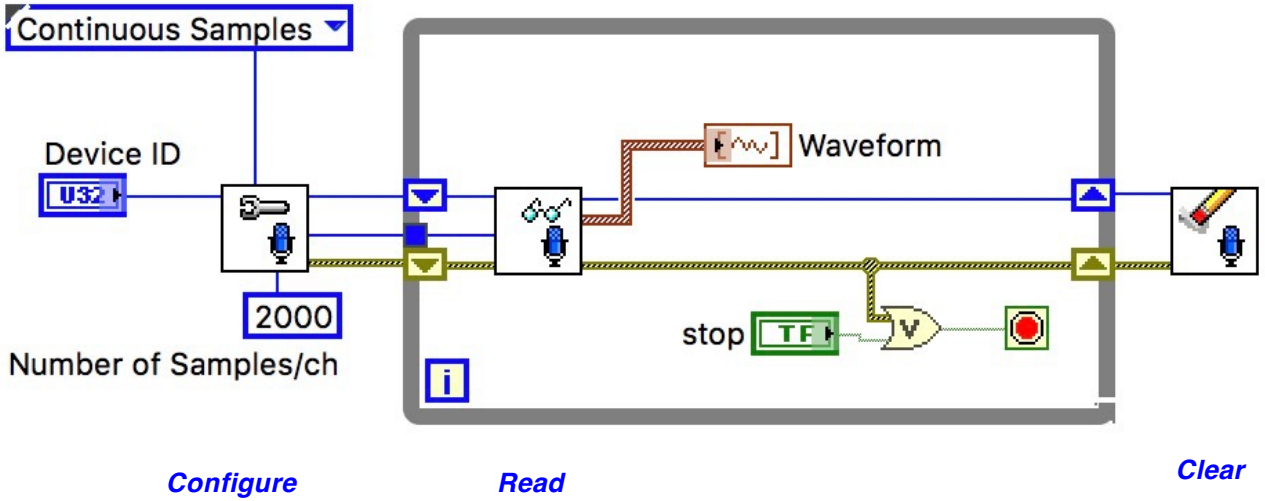
Solution: *generate more than 1 sinusoidal and shift the phase of the next sinusoidal until the phase goes back to 0. In our example every 10 sinusoidals the phase goes back to 0.*



Note: in the project we'll define the number of samples, not the frequency, it will calculated

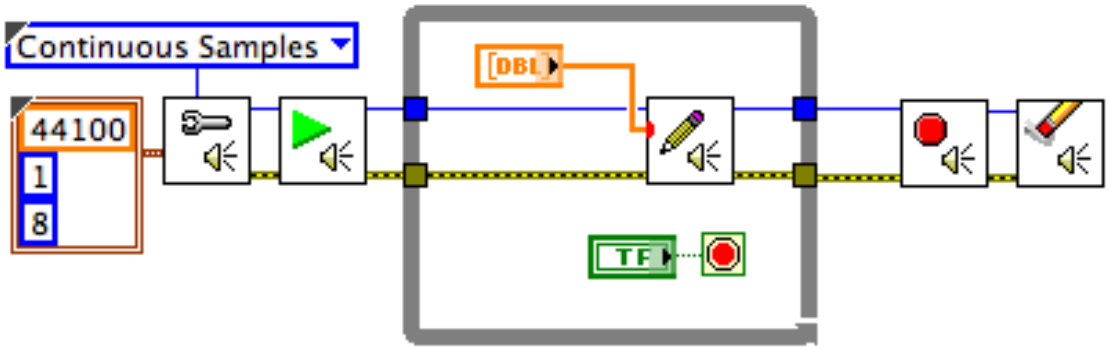
Acquiring a sound

As easy as other DAQ operations ☺ !



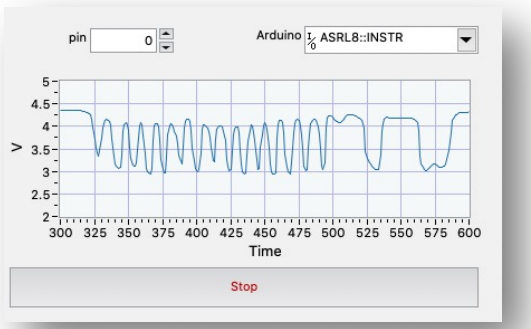
Playing a sound

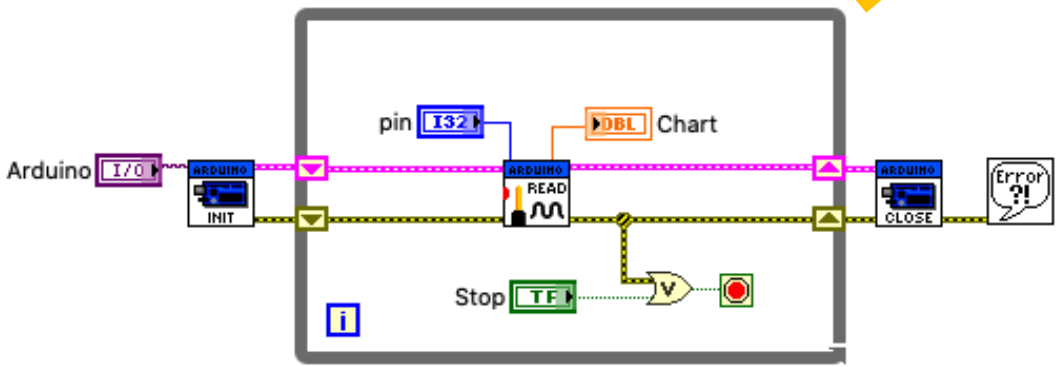
As easy as other DAQ operations ☺ !

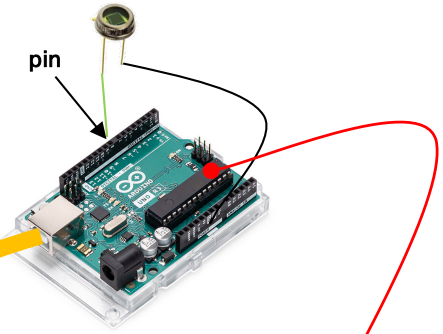


Note: similar set of VIs exists for sound input operations

Talk to an Arduino







LIFA.ino

Configure *Read* *Clear*

https://arduino.cc

LabVIEW Interface for Arduino
via VIMP

LabVIEW community edition (free)