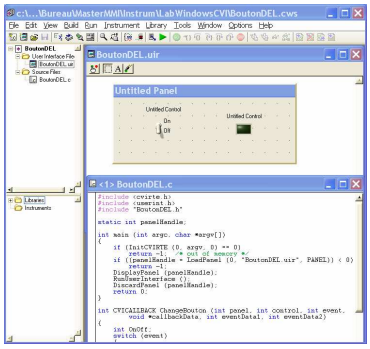
LabWindows/CVI en 5 minutes (ou presque)

F. Chollet

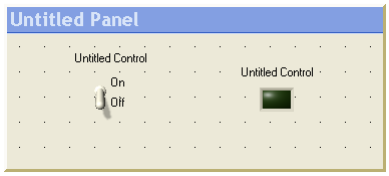
LabWindows CVI

- De façon similaire à LabView, un projet LabWindows CVI comporte au moins deux parties
  - Une face avant (fichier .uir)
  - Un fichier source en C (fichier .c)
- Bien sur la différence majeure avec Labview réside dans l'usage du C au lieu du langage graphique G de Labview.



Face avant

- On démarre en général un projet en créant une face avant (ou User Interface) File > New > User Interface (fichier .uir)
- Placer sur la face avant un interrupteur et une DEL (click droit)



- LabwindowsCVI va prendre en charge toute la gestion des éléments de la face avant, particulièrement graphique, et ne laisser au programmeur que la tache de traiter l'information
- Sauvegarder le fichier .uir

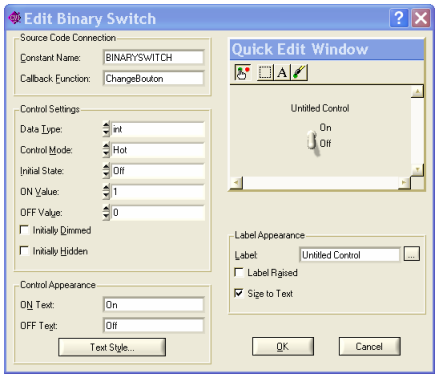
Génération automatique de code

- On peut ensuite générer le code correspondant à la face avant (Code > Generate > Code all)
- Labwindows crée un projet et un fichier source pour afficher la face avant et arrêter proprement:

```
#include <cvirte.h>
#include <userint.h>
#include "BoutonDEL.h"

static int panelHandle;

int main (int argc, char *argv[])
{
        if (InitCVIRTE (0, argv, 0) == 0)
                return -1;   /* out of memory */
        if ((panelHandle = LoadPanel (0, "BoutonDEL.uir", PANEL)) < 0)
                return -1;
        DisplayPanel (panelHandle);
        RunUserInterface ();
        DiscardPanel (panelHandle);
        return 0;
}
```

Créer des évènements

- Toute action sur les éléments de la face avant génère un (ou plusieurs) évènement(s)
- Chaque élément peut alors appeler une fonction dans le code source pour traiter ses propres évènements
- En face avant double-clicker l'interrupteur et ajouter une callback function: **ChangeBouton**
- Noter le champ **Constant Name** (BINARYSWITCH) qui identifiera l'élément dans le code source



Créer le code évènement

- Générer le code correspondant au traitement des évènements en utilisant Code > Generate > All Callbacks (ou plus tard Control Callbacks)
- Le code par défaut correspondant à l'élément s'ajoute au code source:

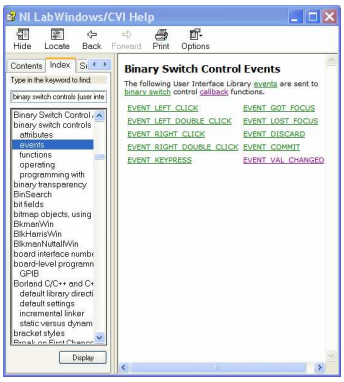
```
int CVICALLBACK ChangeBouton (int panel, int control, int event,
                              void *callbackData, int eventData1, int eventData2)
{
        switch (event)
        {
                case EVENT_COMMIT:

                        break;
        }
        return 0;
}
```

Traiter l'évènement

- Dans la fonction callback, on peut alors utiliser la variable event pour déterminer quel évènement a déclenché l'appel de la fonction et prendre alors toutes les actions nécessaires
- L'évènement COMMIT (identifié par la constante EVENT_COMMIT) correspond à un simple click gauche sur l'élément
- L'évènement VAL_CHANGED (indiqué par la constante avec préfixe EVENT_VAL_CHANGED) correspond à un changement de valeur, pour l'interrupteur c'est à dire à son basculement: c'est l'évènement que nous voulons pour commander la DEL

Propriétés des éléments

- Chaque élément possède une liste de propriétés consultable avec l'aide en ligne:
  - Attributes: décrivent l'état ou l'aspect de l'élément
  - Event: les évènement liés à l'élément
  - Function: les fonctions permettant de manipuler l'élément
- Par exemple pour connaître la valeur retournée par un élément on utilisera la fonction **GetCtrlVal()** et pour affecter une valeur **SetCtrlVal()**



Programmation

- Ainsi pour allumer et éteindre la DEL (de nom LED) on pourra utiliser:

```
int CVICALLBACK ChangeBouton (int panel, int control, int event,
                              void *callbackData, int eventData1, int eventData2)
{
        int OnOff;
        switch (event)
        {
                case EVENT_VAL_CHANGED:
                        GetCtrlVal(panelHandle, PANEL_BINARYSWITCH, &OnOff);
                        SetCtrlVal(panelHandle, PANEL_LED, OnOff);
                break;
        }
        return 0;
}
```

- On remarquera que le nom de l'élément (provenant du champ Constant Name entré en face avant), *comme tout les noms de constantes*, a un préfixe, ici, le terme **PANEL_**

Test et exécutable

- Il suffit alors de lancer la compilation (Run – flèche verte) pour que le programme s'exécute

- On est alors en mode de déboguage interactif et il est possible d'arrêter le programme avec le bouton stop (bouton rouge) pour observer les variables et faire d'autres opérations. Un second appui sur stop arrêtera vraiment le programme.
- Pour obtenir un exécutable indépendant (.exe dans le répertoire du projet) on sélectionne Build > Configuration > Release avant Run
- Le précédent programme manque de bouton fin (qui devrait alors appeler la fonction QuitUserInterface(0)) et ne s'arrête qu'avec un click droit sur la barre des programmes Windows puis Fermer :-)