GUI, Qt, OnX, etc…

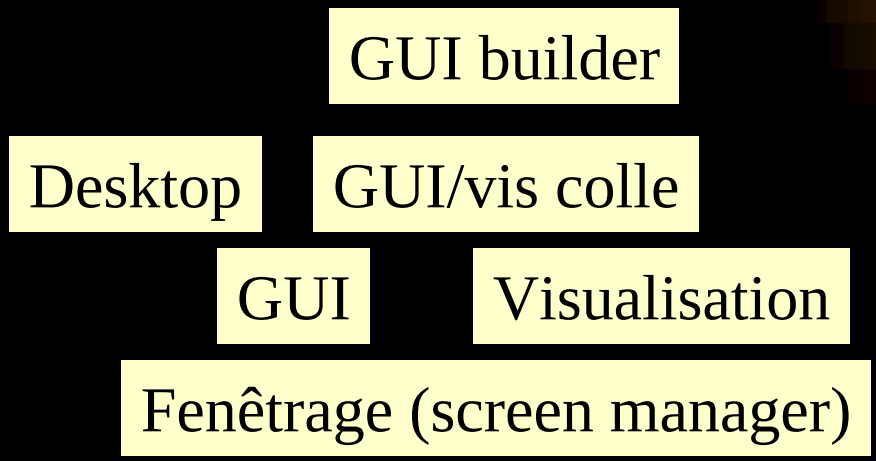
Problème

- Les « desktop providers » ne fournissent pas un API commun pour gérer l'interface utilisateur (les boutons, menus, etc…) (GUI : Graphical User Interface).
- Il n'y a pas eu de miracle comme pour la visualisation avec OpenGL (couche rendering).
- Les trois principaux « desktops » :
  - Microsoft (Bill Gates) Windows
  - Apple (Steve Jobs) NextStep (Cocoa)
  - GNU/Linux :
    - GNOME : Richard Stallman GTK
    - KDE : « ? » Qt (TrollTech)
- Le « GUI » est encore un cheval de bataille !!!

Problème

- D’autres acteurs / produits :
  - Xt / Motif (maintenant open source (OpenMotif))
  - Java / AWT, SWING
  - FLTK
  - ….
- Problématique du langage :
  - pygtk, pyqt
  - Javagtk (ca doit exister)
  - …
- Produits maisons :
  - OnX
  - L’autre ; ROOT dans lequel on trouve TButton, TMenu, etc..

Couches



- Produits : X11, Windows, Aqua, Xt, Motif, Qt, GTK, NextStep, Cocoa, Carbon, FLTK, AWT, SWING, SWT,glade, qbuilder, pyqt, pygtk, KDE, GNOME, VTK, Inventor, java3d, OpenGL, SoXt, SoQt, SoWin, SoGtk, SC2I.

Problème

Que choisir ?

Choix important :

- une grosse application (genre « event display ») peut comporter des centaines de « widgets » (widget = élément d'interface utilisateur)
- on a pas envie de changer tous les jours.

Critères de choix

- Portabilité : Windows, Mac, Linux.
- Coût : collaboration internationales mettent hors jeu les solutions payantes.
- Fenêtrage (a ne pas confondre avec le GUI et le desktop) :
  - Windows
  - Aqua
  - X11
- Intégration du graphique :
  - OpenGL
  - Mais aussi des « surcouches » Inventor, VTK
  - Java3d
- Langage : on se simplifie la vie en prenant une « toolkit » écrite dans le langage de l'application :
  - C++ : Qt, FLT. GTK ecrit en C.
  - Java : AWT/SWING.
  - Python : en général wrappers a gtk, Qt, etc…
- Existence d'interface builder.

Lorsque l'on fait le tri…

- C++ / Inventor : GTK (mais pas native C++)
- Java / java3d : AWT / SWING
- C++ / TGraphic : TGUI

Qt

- Qt (TrollTech) :
  - écrit en C++
  - Sur les fenêtrages natifs des « desktop providers » (Windows, X11, Aqua)
  - Intégration d'OpenGL et d'Inventor (SoQt package)
  - GPL sous Linux et MacOSX

- Qt n'était pas open source sous Windows

Donc l'annonce de TrollTech de mettre Qt-4 GPL sous Windows est importante ; cela change la donne.

Autour du LHC

- ALICE : C++, TGUI, TGRAPHIC. Plateformes ?
- CMS : C++, Qt, Inventor. Mais ne supportant pas Windows (on comprend maintenant pourquoi)
- LHCb : C++, OnX (Xt/Motif et Windows), Inventor. Mais on trouve aussi d'autres applications en Qt, gtk et java. Support de Windows.
- ATLAS : data framework en C++, mais ATLANTIS en java (!). Plateformes ?

« Evidement » il n'y jamais eu de « rtag » pour le choix d'une GUI toolkit et d'une librairie pour le graphique.
CMS a forcé Qt sans jamais avoir été suivi par les autres.
On sent ici l'intelligence profonde, le sens collaboratif inné, l'extraordinaire vision a long terme du CERN vis a vis des problèmes de software…

Autour du LHC (2)

- On voit que d'avoir Qt GPL sous Windows change beaucoup de choses.
- Le choix forcé de CMS va devenir un choix naturel pour LHCb.

  En particulier, technologies communes pour le GUI, le graphique et le scripting (Qt, Inventor/Coin, python).
- Le choix TGUI (et TGraphic, CINT) d'ALICE va devenir difficilement justifiable. (super)
- Est ce qu'ATLANTIS en java est tenable face a un environnement interactif en Qt/Inventor/Python, sachant que le « data framework » est en C++ ?
  Je pense que non. Julius pense que oui.

À la maison

- OnX : au dessus des partis.
- Description du GUI en XML (hiérarchie de widgets et callbacks)
- « factories » (drivers) pour :
  - Xt/Motif (en fait le plus performant des GUIS sous X11 !!!)
  - WIN32
  - NextStep
  - Qt
  - GTK
- Forcement le plus rapide (puisque exploitant les GUI des constructeurs).
- Le GUI builder vient avec l'application.
- Donc philosophiquement différent de Qt (qui n'exploite pas les GUIs des constructeurs).

- Mais est ce que cela vaut le coup de continuer ?????

Le futur ?

- .NET (Microsoft)
- DOTNET (GNU .NET)
- CLI, IL : C#, java, C++
- System.Windows.Forms classes C# communes a .NET et DOTNET sur l'API Win32.
- donc supportées en natif par au moins un constructeur (Microsoft).
- Sous Linux (MacOSX ?) : wine : émulation de WIN32 sur X11. Même nombre de couche que gtk ou Qt ; donc a priori même performances.
- mais DOTNET (GNU / Stallman) fait la promotion de GTK# !!!
- Integration du graphique ???

- Cela vaut le coup de regarder.

Conclusions

- L’interface utilisateur : un merd… depuis 20 ans.
- Qt GPL sous Windows est un événement.
- Cela devrait changer le paysage LHC autour des softs interactifs.
- Mais dans le fond le problème ne sera pas réglé tant que les « desktops providers » ne viendront pas avec une solution commune pour décrire un GUI.

Références (Goggle) :

Qt : http://www.trolltech.com

GTK : http://www.gtk.org

Apple / Cocoa, Microsft / Windows : en cherchant bien ca se trouve…

OnX : http://www.lal.in2p3.fr/OpenScientist

ROOT : http:://root.cern.ch