# LAL-Murino
# softinex

G.Barrand, Ingénieur de Recherche. CNRS / IN2P3 / LAL

http://wall.lal.in2p3.fr
http://softinex.lal.in2p3.fr

2010 : Apple donne au LAL/SI quatre machines et huit écrans dans le cadre d'un projet ARTS (Apple Research and Technological Support).
2010 : L'arrivée des tablettes : Apple/iPad, Android (Samsung Galaxy Tab, etc…).
2007/2010 : L'arrivée des "stores" (AppleStore, AndroidMarket) qui permettent une distribution mondiale aisée d'applications.

Des logiciels on été écrit pour gérer "comme une seule machine" (un mur d'écrans) les écrans et machines ARTS, et ce pour visualiser des données scientifiques. Mais des logiciels pensés aussi pour permettre la visualisation de ces données sur tablettes (iOS, Android) avec le même jeux de code.

Plateforme, le "petit mur" (murino) du LAL :

- 4 machines ayant chacune : deux 2.26 GHz Quad-Core Interl Xeon Nehalem hyperthreading (16 coeurs apparants), 2G memoire, disque de 1 TB 7200-rpm Serial ATA 3gb/s, 4xNVIDIA GeForce GT 120 512 Mbytes.
- 12 écrans de 1920x1200 pixels avec, hélas, de gros bords.
- Un iMac servant au pilotage de l'ensemble.
- Un switch gigabit pour faire un réseau privé rapide.
- Une borne Airport pour faire un lien WIFI avec les tablettes.
- Un support mécanique, fait au LAL, pour monter le tout sur deux chariots à roulettes.

Version 2010 à huit écrans (visualisation d'un gros jpeg du détecteur ATLAS) :

Logiciels, choix globaux :

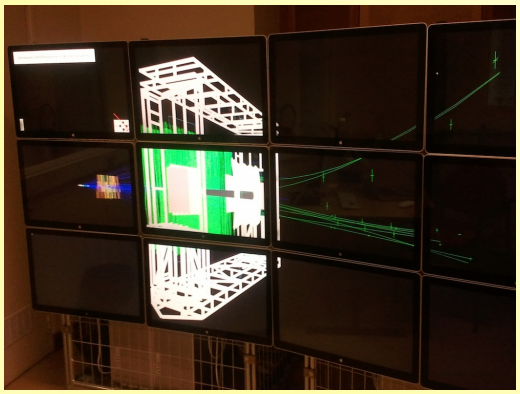
- Le LAL s'est impliqué depuis longtemps dans la visualisation pour les expériences (LEP/ALEPH, NA31, NA48, LHC/LHCb) et ce en utilisant différentes librairies comme OpenGL, OpenInventor, OpenMotif, gtk, Qt, etc…
- La volonté de maitriser les tablettes et smartphones a impliqué une profonde révision de la manière de gérer le graphique et l'interface utilisateur, puisque la plupart des librairies existantes sur desktop/laptop n'étaient pas disponibles sur iOS et Android.
- Volonté d'avoir le même système graphique pour gérer notre murino que pour les tablettes.
- Qu'y a-t-il de commun ? C++ et GL-ES. On construit sur cela.
- L'interface utilisateur ? On tente de faire le "GUI" avec les même classes que celles utilisées pour visualiser les données. La fonctionnalité est la, même si l'ergonomie est clairement à améliorer.
- Pour le contrôle du mur, on ajoute une couche réseau basée sur un protocole TCP/IP simple.

softinex = {inlib, exlib, ourex + applications comme "wall", ioda, g4view, etc…} :

- Logique centrale : avoir un gestionnaire de "graphes de scène" construit sur GL-ES (et C++/STL) ; ce sont les classes inlib/sg (SG=Scene Graph).
- inlib/sg : un ensemble de "noeuds". Certains sont des feuilles comme les noeuds de géométrie (inlib::sg::cube, inlib::sg::vertices), des noeuds de texture d'images sur une surface ou une sphère, de définition de caméra (inlib::sg::ortho, perspective, frustum, lrbt), du noeud inlib::sg::matrix pour positionner des objets. D'autres noeuds sont des containers (inlib::sg::separator, group). (Nous nous sommes fortement inspiré de l'architecture d'OpenInventor). Un assemblage de containers et de noeuds feuilles forment un graphe de scène.
- Actions : des classes de inlib/sg sont des "actions" qui permettent de traverser un graphe de scène pour faire quelque chose de spécifique . Une classe action importante est l'action "render_GL" qui produit le GL-ES pour faire le rendu. (inlib::sg::pick_action permet de gérer le picking)
- exlib : jeux de classes qui font le lien avec des librairies "externes" comme X11, expat, freetype, jpeg, cfitsio, dcmtk, etc…
- Pour assurer la portabilité et le même comportement sur toutes les plateformes, on embarque une version de certains packages externes (comme expat, jpeg, cfitsio, etc…) dans la librairie "ourex".

Diaporama de fichiers jpeg, png :

Visualisation de géometries (ici du détecteur LHCb) :



Une scène avec des images DICOM, et une gerbe electromagnétique dans des tranches de plomb et d'argon provenant d'une simulation (simple) faite avec GEANT4 :

La logique du mur :

- Un processus par écran (soit douze en tout et donc trois par machine). Un processus maître sur la machine de contrôle.
- Chaque processus-écran reçoit du processus-maître un même graphe de scène contenant un noeud camera, des noeuds de géométrie, de lecture d'images, de positionnement d'objets, etc…
- Mais le noeud camera (classe inlib::sg::lrbt) de chaque processus-écran est paramétrisé pour voir un douzième de la projection de la scène. C'est la technique du rendu tuilé.
- Le processus-maître peut envoyer des ordres de déplacement, zoom de caméra à tous les processus-écrans de manière à faire des animations.
- Pour certaines géométries, on télécharge dans les GPUs les triangles, segments, points d'une scène en une seule fois, ce qui permet des animations particulièrement fluides, même sur de grosses géométries.

Demandez une démo !!!!!

cnrs déployer les frontières
IN2P3 Institut National de Physique Nucléaire et de Physique des Particules
LAL Laboratoire de l'Accélérateur Linéaire
JDEV2013