Plan de travail pour de la visualisation pour LSST

LSST-France APC, Novembre 2018

G.Barrand, CNRS/IN2P3/LAL

G.B => mise à niveau astro !

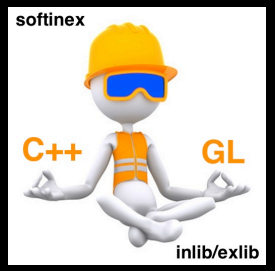
- Passif HEP => remise à niveau sur le domaine de science astro : problématiques scientifiques, instruments, traitement de données. (Youpi, je sais maintenant ce qu'est un « Cassegrain », un « survey » et une « coaddition » ☺. Merci Wikipedia, arXiv, YouTube, etc… !).
- Pages web de Pan-STARRS => meilleure vue sur le traitement de données pour des « surveys ». Documents LSST, meilleure vue sur le « DM », meilleure vue sur le stack LSST.
- Installation du stack sur mon Mac, faire les tutoriaux.

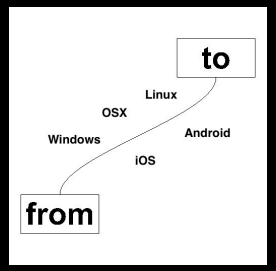
- Mais, pas sûr que je sois incollable en acronyme astro ☺ ! (LSST, STSCI, SDSS, VO, IVO, IVOA, MOC, RA, DEC, DESC, DR, DIA, CCD, PSF, FITS, HDF5, HDFS, TB, PB, DDF, WFD, MS, DAC, QSERV, SQL, RDBMS, ADQL, NED, IPAC, NASA, CFHT, CDS, NERSC, SIAP, SAMP, SSAP, TAP, UCD, MOND, CDM, CMB, HST, BAO, WMAP, WL, OTA, SCT, SSO, DSO, ESO, VLT, ELT, ESA, MAST, NEO, TNO, SKA, LSS, FLRW, PCA, ICA, IRSA, SIMBAD, SED,...!)

Et donc pour de la visu…

Tel que je vois les choses maintenant : utiliser toute l'expérience acquise à ce qui est encore le LAL pour :

- Offrir de la visu dans le cadre de Spark.
- Pour le stack LSST, offrir une autre possibilité de visu en parallèle à « display_ds9 », « display_firefly », « display_matplotlib »
- Autour de DC2 : se placer comme « un utilisateur externe en bout de chaine voulant voir des images et un gros catalogue ». Et donc ayant besoin d'accéder aux données au travers d'un « firefly serveur ». Et donc de faire un « firefly client » utilisant la technologie LAL pour visualiser.





Techno LAL : softinex

- Tout est dans ce qui s'appelle « softinex » (= software done with the inlib/exlib classes). Voir http://softinex.lal.in2p3.fr ou maintenant http://gbarrand.github.io. Source releases déposées sur github.

- C/C++, mais avec du SWIG wrapping pour Python (inexlib_py).
- Graphique basé sur une logique de « graphe de scène » (inspirée d'OpenInventor). On construit une scène en faisant un arbre de « nodes » : « separator », « matrix », « cube », « image », etc…
- Multiple « renderer » : un renderer « traverse un graphe » pour produire des images : renderer on-screen produisant du GL-ES, un renderer produisant du WebGL, mais aussi un renderer off-screen (pur C/C++) produisant direct des png, jpeg, ou PostScript.
- Exemple : inlib/sg/plotter : gros « sg » pour faire du « plotting ».

Softinex, inlib/sg sur…

- À peu près tout ce qui a un écran. Gros accent sur la portabilité (C++98) et le fait de pouvoir tourner en local pour coller aux CPU/GPU : Linux/X11/Mesa, Mac/Cocoa/Apple-GL, Windows-10/win32/gl32, mais aussi iOS/Apple-GL-ES et Android/GL-ES. (Un jour, je me ferai l'appli pmx (montrant LHCb) sur une Apple Watch; juste pour le fun).
- On sait faire des « apps » pour les « stores ».
- MAIS aussi du graphique offscreen. Probablement LA solution pour du « big data batchien ».
- On sait faire du WebGL et du docker… mais grosse préférence de rester en local pour avoir de l'« interactif très réactif ».

- Des fois, il faut bosser encore et toujours les « couches de base » : exemple, la stratégie GL-ES a fonctionné depuis 2010, mais Apple lâche OpenGL… ☹. (La il va falloir faire du Metal… hurlant (parce qu'API en Objective-C))

Et donc, « le plan »

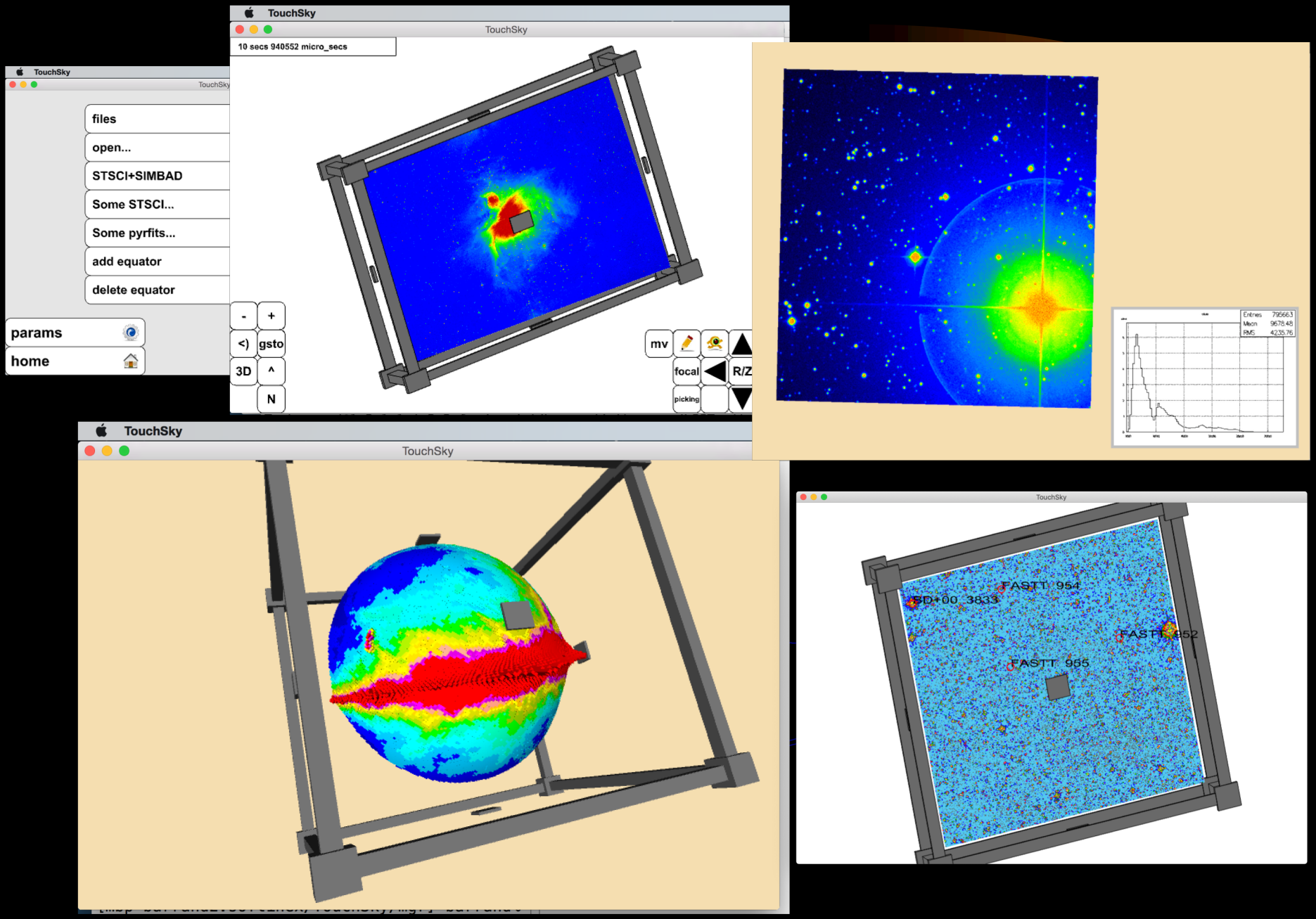
- Spark : inexlib_py : plotting, mode « off screen » et mode « on screen ». (La connexion est déjà faite. 1.0.0 releasé. Voir demo sur YouTube (channel « Guy Barrand »))
- Spark : mode client/serveur : l'idée est d'avoir un « scene graph viewer serveur » en local (une sorte de « X11 à graphes de scène »), alimenté par un client (spark) lui envoyant, par exemple, des plots (des inlib/sg/plotter).
- Pour LSST/stack : display_inexlib : probablement un « API stack » à inexlib_py. (en // à display_ds9, display_firefly, display_matplotlib).
- DC2 : donc là, le point chaud sera de voir comment attaquer un « firefly serveur ». Et donc probablement de faire un mode « firefly client » au « scene graph viewer » ci-dessus.

Sachant que…

Depuis 2014, une « marche d'approche astro » a déjà permis :

- d'avoir des « nodes » ciblés images.
- d'intégrer la lecture de FITS (et HDF5) (y compris sur iOS et Android).
- d'intégrer la possibilité de faire des requêtes http sur STSCI ou SIMBAD.
- d'avoir une navigation « multi-résolution pyramidale » d'une « grande image ».
- d'avoir du code pour faire de la navigation à « calotte sphérique » sur une grande image sphérique. (HEALPIX, HIPS).
- faire une version 1.0, d'une « app » TouchSky (voir YouTube).

Mais, jusqu'à présent, ce n'était pas clair comment intégrer cela dans l'environnement LSST : maintenant cela se précise.


TouchSky
files
open...
STSCI+SIMBAD
Some STSCI...
Some pyrfits...
add equator
delete equator
params
home
10 secs 940552 micro_secs
mv
focal
R/Z
picking
Entries 79566
Mean 9678.48
RMS 4235.76
FASTT 954
FASTT 965

Ce plan dans le monde…

- Point de vue du « science program » : web + python. (Firefly-server + firefly-client ciblé web/python).
- CDS : Aladin : web + java, Aladin-light : local + java.
- Softinex : local + C++. Donc effort complémentaire aux autres.

Ensemble, on peut réfléchir aux « représentations, scènes, projections, plots, animations, etc… » adéquates.

- IMPORTANT : vu le turnover des technologies autour de l'interactif, il vaut mieux ne pas mettre tous les œufs dans le même panier, avoir plusieurs cordes à son arc, faire feu de tout bois, surtout autour de grands programmes scientifiques sensés durer plusieurs décennies. (En plus, cette politique offre une certaine garantie d'ouverture, d'accès aux données, importante sur le long terme).

VR, AR, etc…

- VR : programmation difficile pour le moment. (Que des « toolkits » propriétaires).
- AR : ça j'aime beaucoup ! Le mieux serait de croquer la pomme. Apple met clairement les moyens de ce côté la. (Et le dernier iPad à l'air d'être une bête de course interactive…).
- Mur d'écrans : l'installation du LAL se désintègre doucement : pas de volonté de ma gouvernance de faire une mise à jour. (Cela va probablement disparaitre dans le haut fourneau de la refondation *).

* Est-ce qu'au GLO **, la visualisation, l'interactivité et le software « pour la recherche » auront « enfin » les structures, les moyens et les gouvernances adéquates ?

** Grand Laboratoire d'Orsay (?). (GLO = aussi Graphics Lib d'Orsay ! ☺)