

De la visu (3D) pour de l'astronomie

(et non pas « la visu » pour LSST)
(mais avec un accent sur la visu de « grandes images »)

Notre approche, états d'âme



- Approche Googlienne du net : tout est dans les serveurs (chez Google !) (et donc à la NSA), les données mais aussi l'intelligence. Pour la visu et l'interactif on ne donne aux gens que des devices « montrant une image finale ».
- Approche Applienne du net : on donne aux gens des devices survitaminés : l'intelligence et la puissance de visu et interactive sont dans les mains des gens, seules certaines données sont dans les serveurs. On tient beaucoup à la « great user experience » avec les devices.

Je suis un Apple Guy ! ☺

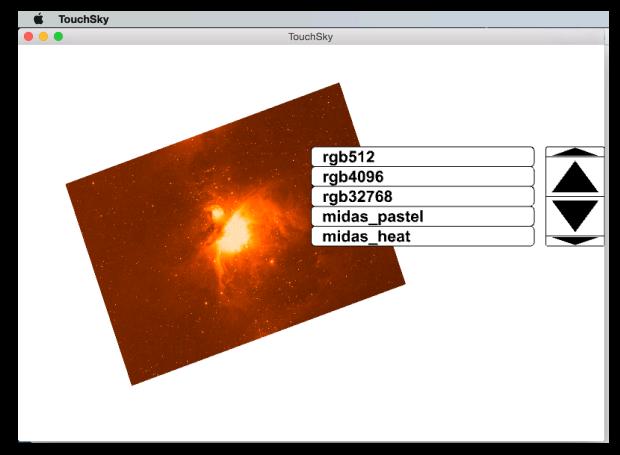
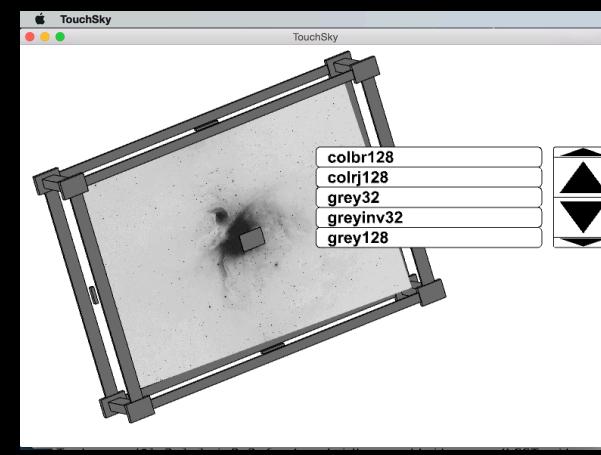
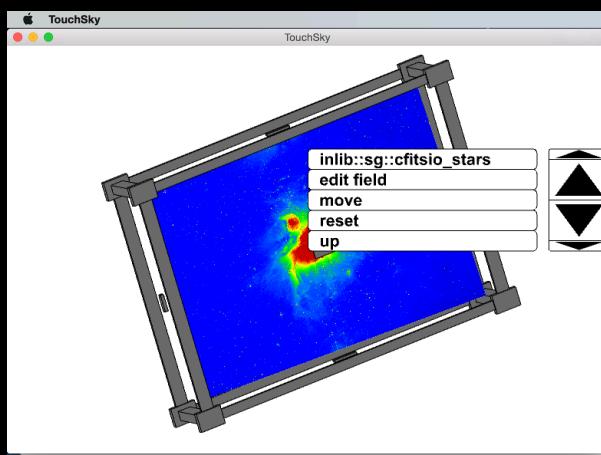
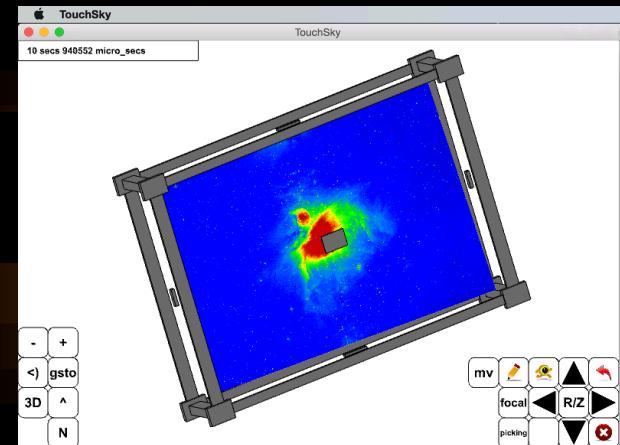
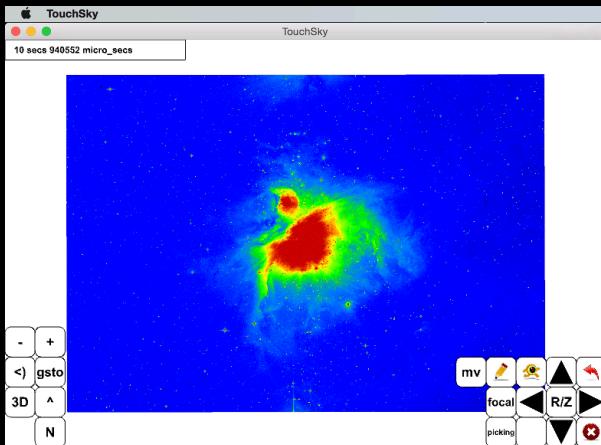
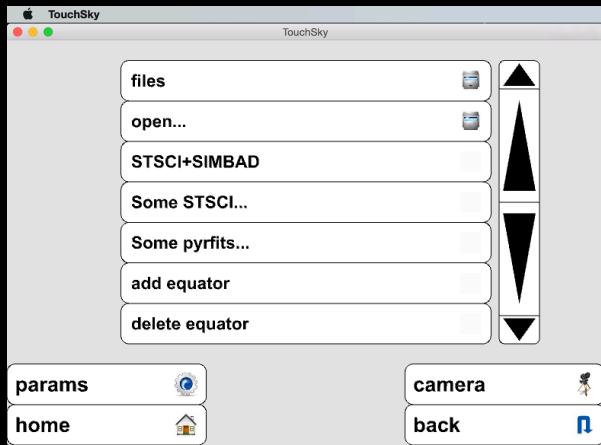
Techno...

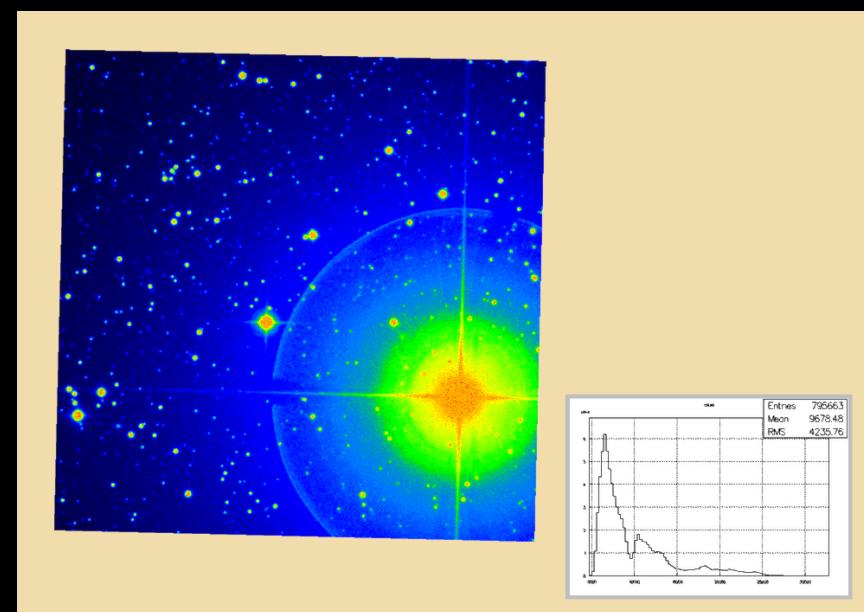
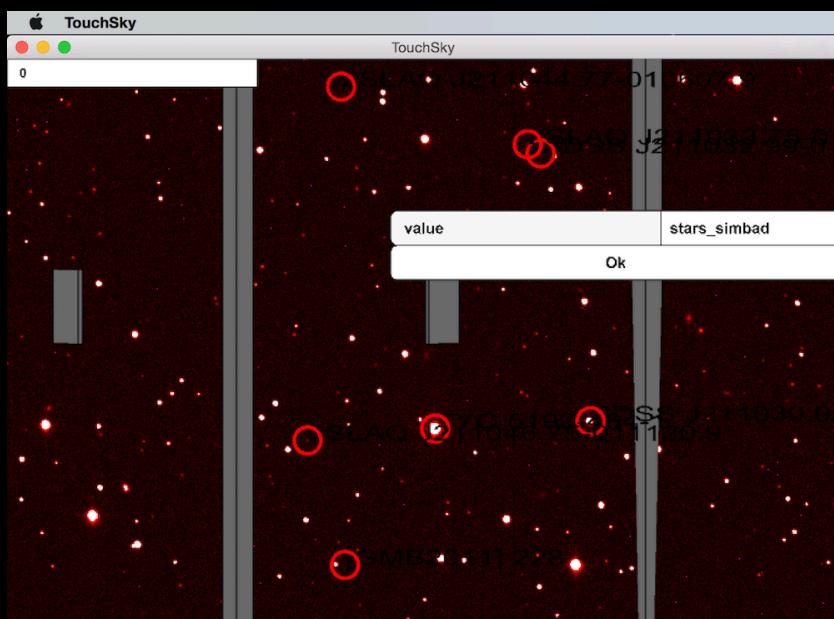
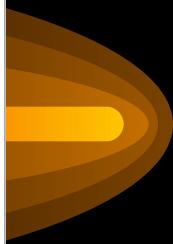
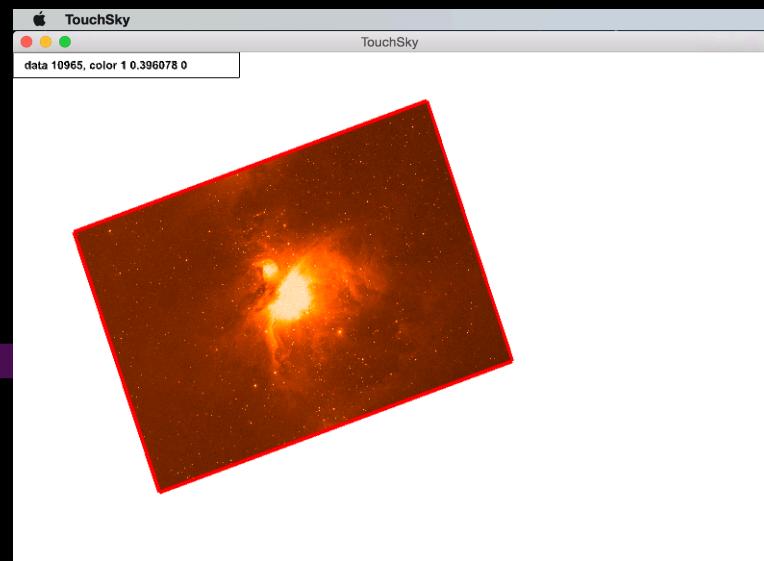
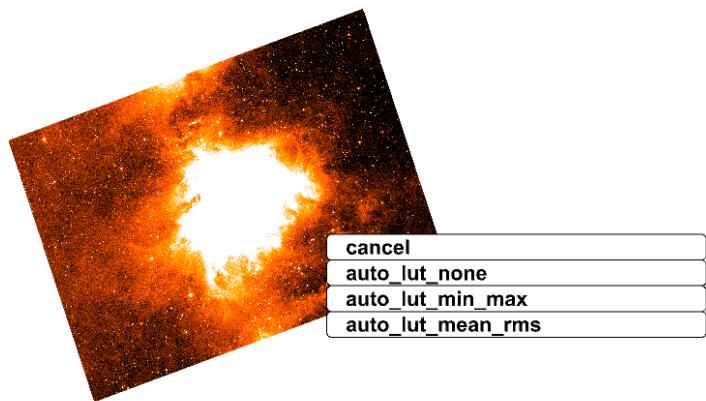
- Et donc je laisse les développements d'une interactivité « 100% » web à d'autres... (Avec tout leurs web-bazarre JS, WebGL, etc...). (De toute façon je n'ai aucun mandat pour le déploiement en masse de serveurs !) (On fait avec ce que l'on a : moi mon iPad ! ☺).
- On exploite iOS, Android, OSX, Windows-10, Linux avec leurs capacités natives communes : GL-ES, C++. On colle au silicium.
- J'exploite l'expérience acquise depuis 2010 (sortie de l'iPad) en HEP* et thésaurisée dans mes softinex : <http://softinex.lal.in2p3.fr>
- On rappelle la règle des « 3 secs » : s'il n'a pas une réponse au clic/touch après 3 secs, l'utilisateur (vous, moi !) trouve rapidement qu'une appli est insupportable. (Pour certains problèmes, peut-on avoir la réactivité nécessaire avec du 100% web ?)

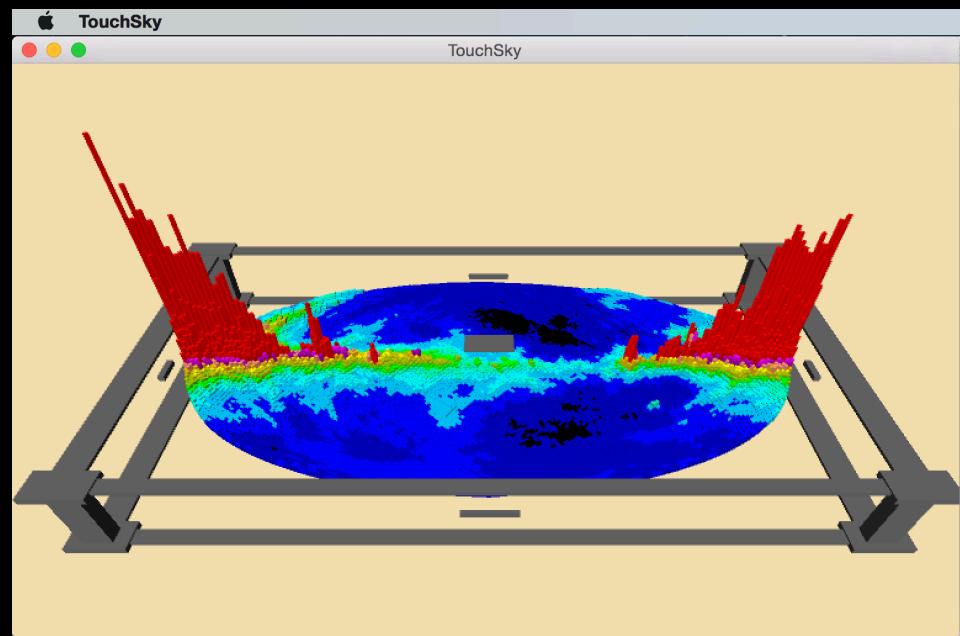
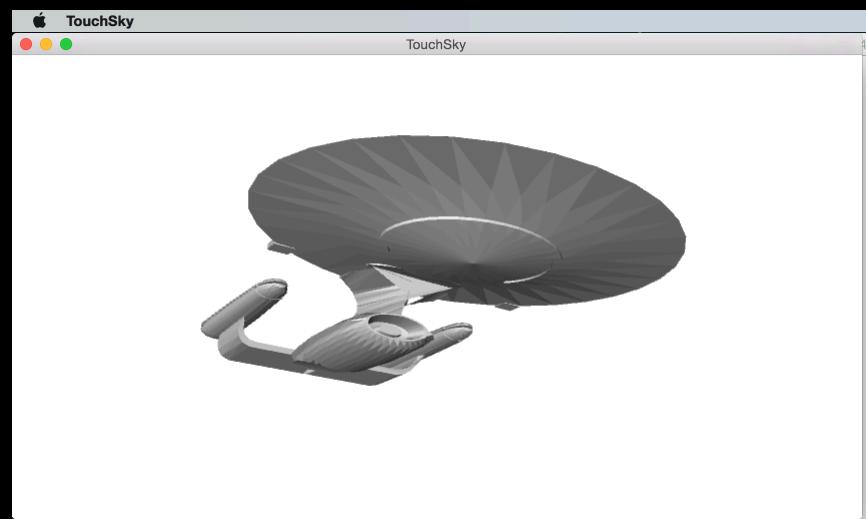
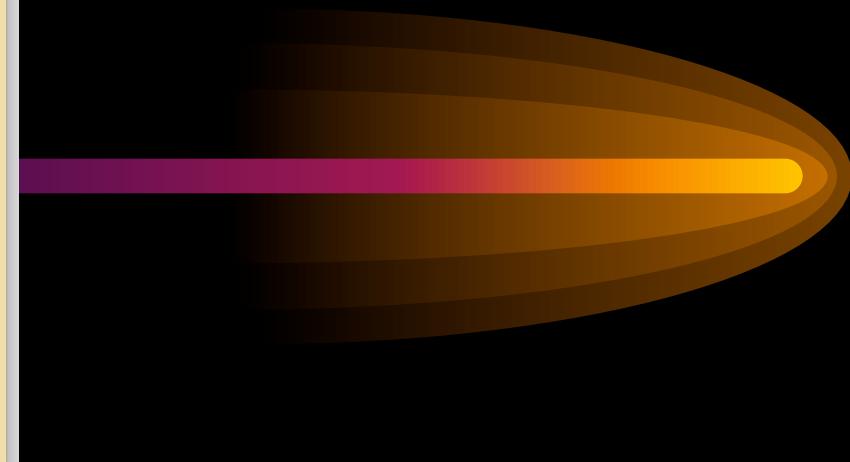
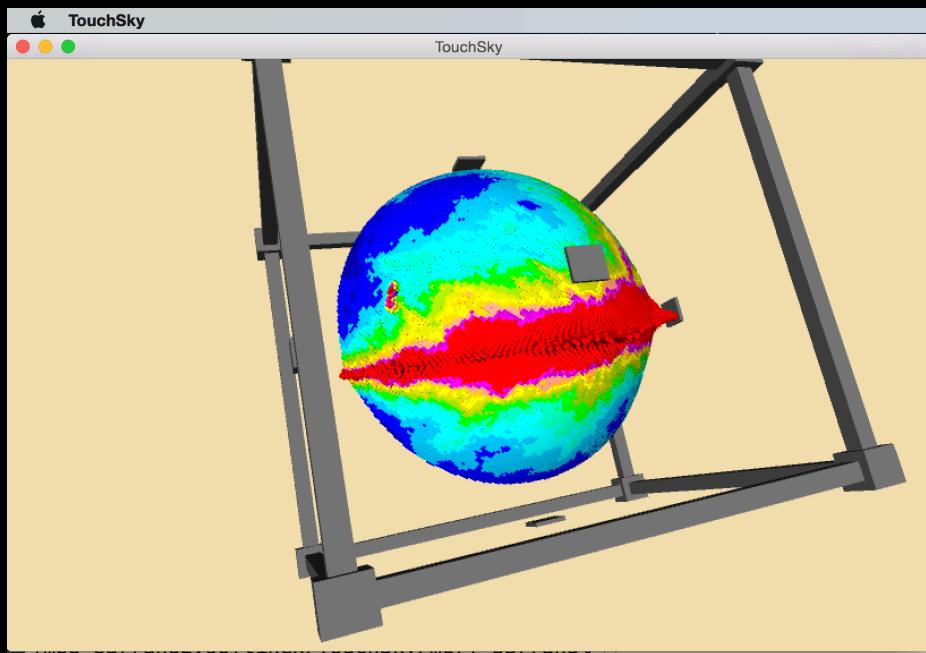
* HEP = Highly Exotic Programming

Visu astro...

- Grosse phase d'apprentissage pour moi depuis 2014 : visualisation « diverses et variées » de « tas de trucs » astro. On se fait les dents.
- Et donc thésaurisation de code autour de l'astro :
 - autour du format fits (portage de cfitsio sur iOS, Android, etc...)
 - visu 3D d'images avec du texture mapping.
 - manipulation interactive des images.
 - retour de picking (sur des images dans une scène 3D !).
 - changement de colormap.
 - interrogation depuis les devices de sites comme STSCI pour des images fits et de SIMBAD pour les catalogues.
 - superposition d'objets SIMBAD sur les images.
 - projection « dynamique à fenêtre » sur des sphères (HEALPIX).
 - histogrammation, plotting.
 - animation.
 - etc...

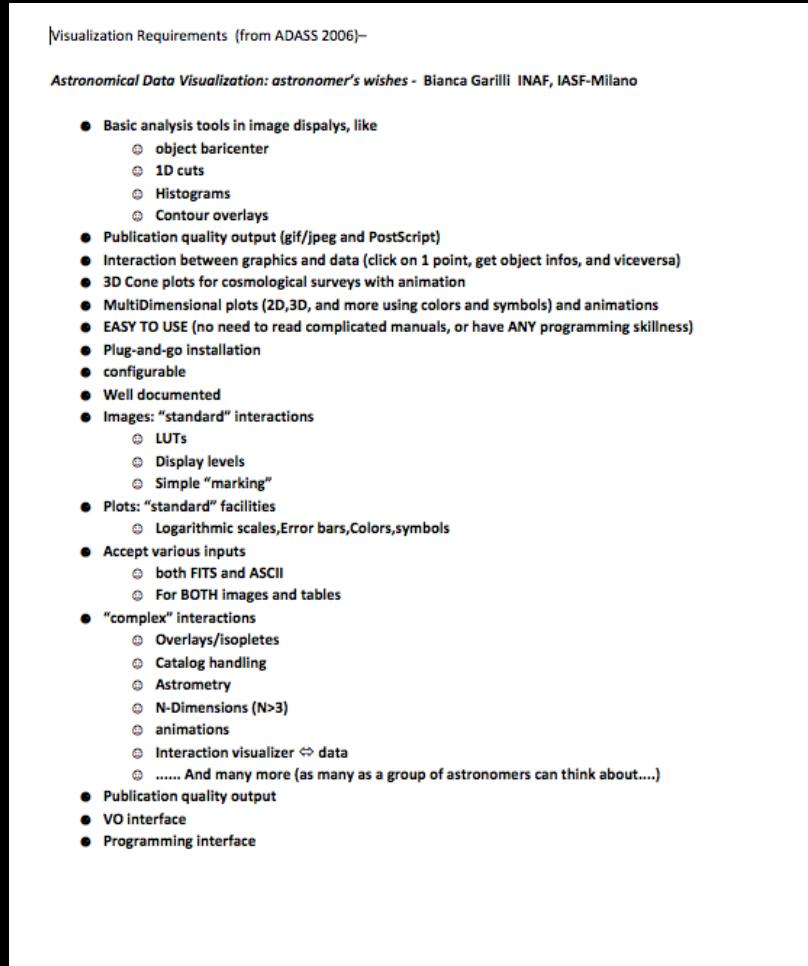






G.Barrand, CNRS/IN2P3/LAL

- On couvre beaucoup de points du document :



Visu grandes images

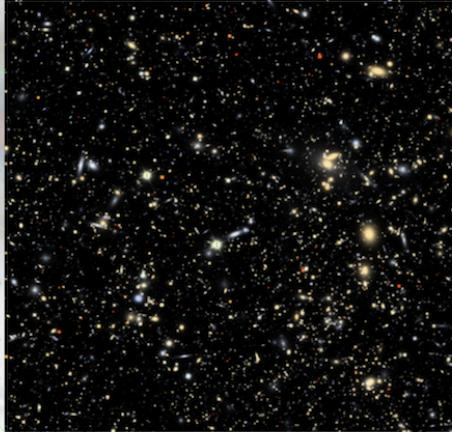
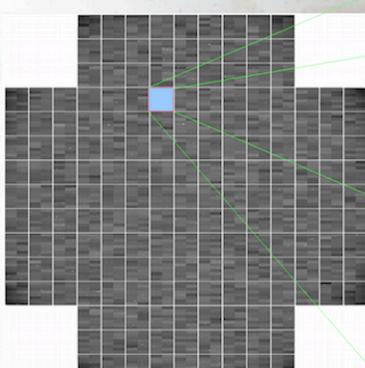
- Et où je comprends (enfin ! ☺) ce que veut dire « grandes images » pour LSST ! Partie centrale : 40000x40000 ! Soit un mur de 20x20 écrans tels ceux de notre « murino » (ou 2 LRI/WILD ou 4 LRI/WILDER)

 Le flot de données LSST

PetaSky

Caméra : 198 CCD (16 Mpix) en parallèle
→ 3,2 G pixels !
~ 6 Gbyte / 17 secondes
→ 15 TB / nuit

Comment visualiser de telles images ?
~ 1/1 000 000 000 des données LSST !



23/114 Emmanuel Gangler – Workshop Mastodons 7/27



Travail sur :

- CFHTLS : 20k x 20k (1/4 partie centrale LSST).
- « sky_40 » : image de 40k x 40k simulée par Astromatic/
SkyMaker.
- SDSS/Stripe82 (mais en fait un « stripe », donc pas très proche
du problème LSST, mais intéressant en soi car les images ont
des superpositions).

Grandes images : pyramides tuilées

- Construction « offline » d'une pyramide tuilée multi-résolution. Chaque niveau est une 2-dépixelisation du niveau inférieur, concaténant le contenu de 4 fichiers du niveau précédent.
- En fait la même logique que le mipmapping dans OpenGL !
- Exa :
 - sky-40 : niveau 1 : 4096 tuiles/fichiers (625x625) , niveau 2 : 1024, niveau 7 : 1 fichier (625x625). Soit 8.4 Gbytes.
 - CFHTLS : 1 : 1024 fichiers (604x604) avec 6 niveaux. Soit 2 Gbytes.
- Un programme optimisé multi-threads construit la pyramide pour sky_40 en 15 secondes sur mon MacBookPro. (Bottleneck sur l'IO).
- IMPORTANT : Toutes les tuiles ont la même taille et plus petite que 1024x1024 pour faire du texture mapping avec GL-ES sur toutes les plateformes. (Ça optimise aussi les couches réseaux).

Chargement dynamique des tuiles

- On ne charge que les tuiles dans le scope d'une fenêtre. Cela fonctionne aussi (et surtout !) en 3D.
- Pour l'instant le changement de niveau n'est pas automatique ; étude du changement de niveau suivant la taille du rendu d'un « pixel data » en cours...

Client/serveur : le repserv

- Une pyramide peut être remote sur un serveur « repserv » (C++ et tntnet).
- Notre « repserv » se charge d'envoyer les tuiles vues.
- Un repserv avec quelques tuiles tourne sur « clrlsstweb.in2p3.fr » mais sans port public.
- Un repserv tourne maintenant sur l'OpenStack du LAL avec un port public.
- Tout cela marche plutôt bien, en particulier avec notre petit mur d'écrans où les repserv (Clermont et OpenStack/lAL) sont hautement sollicités.

TouchSky

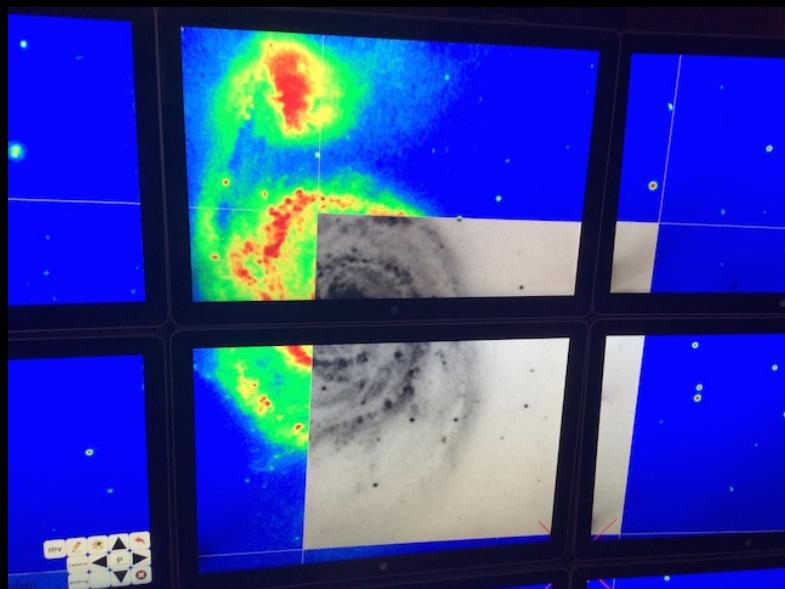
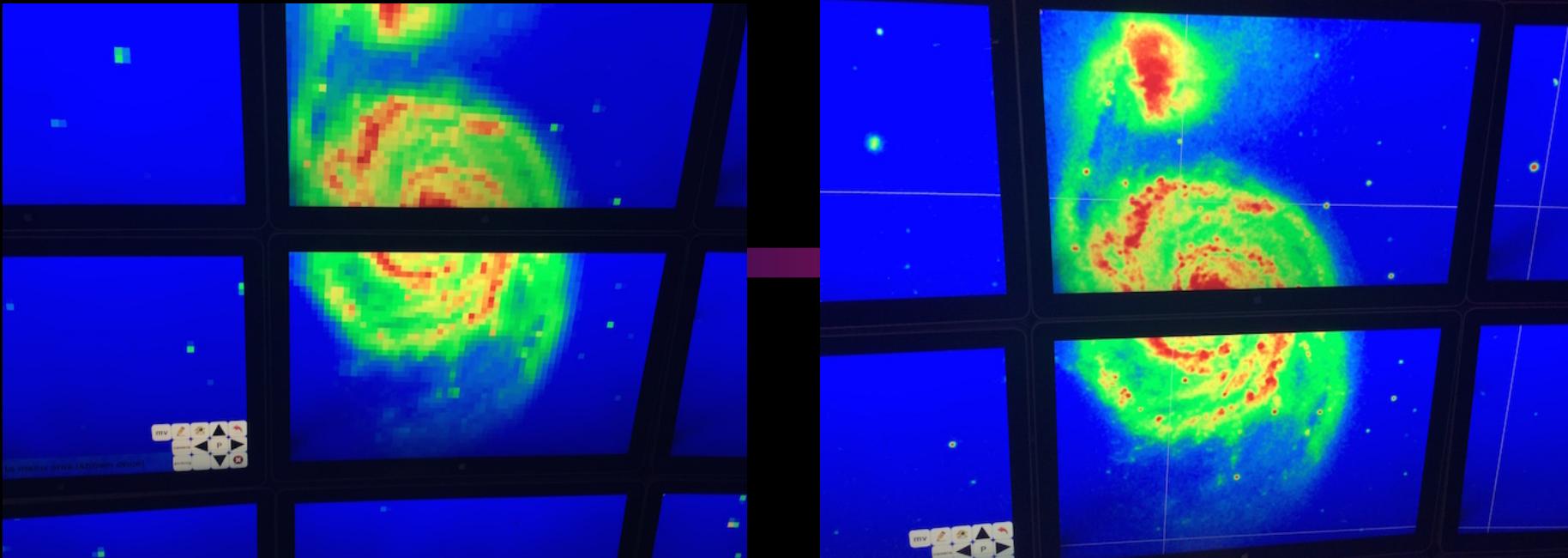


- Appli tournant sur... tout ce qui a un écran !
- Pas encore sur GooglePlay ni les Apple stores car pour l'instant c'est plus un démonstrateur de toutes nos expériences « diverses et variées » de « tas de trucs » astro.
- La disponibilité d'un repserv stable avec un port public sur LAL/OpenStack permet maintenant de définir une ligne claire « visu de grandes images astro » et de tenter une soumission chez Apple. (Qui veut des applis qui « font quelque chose » (et non, justement, des démonstrateurs)).
- Depuis un iPad, TouchSky est « workable » sur une pyramide lointaine si... on est sous la borne ! ☺
- Il existe aussi un mode « batch graphique ».

LAL/murino

- Un process par écran gérant un même « graphe de scène » mais montrant une partie d'une projection orthographique/perspective commune.
- Pour les pyramides, rajout d'une logique spéciale pour qu'un process ne ramène que les tuiles nécessaires vues par son écran.
- Bien que l'installation vieillisse (2010), elle permet d'écrire du code de « visualisation parallèle multi écrans, multi ordis » intéressant . (Cela renforce l'architecture déjà hautement portable du code).



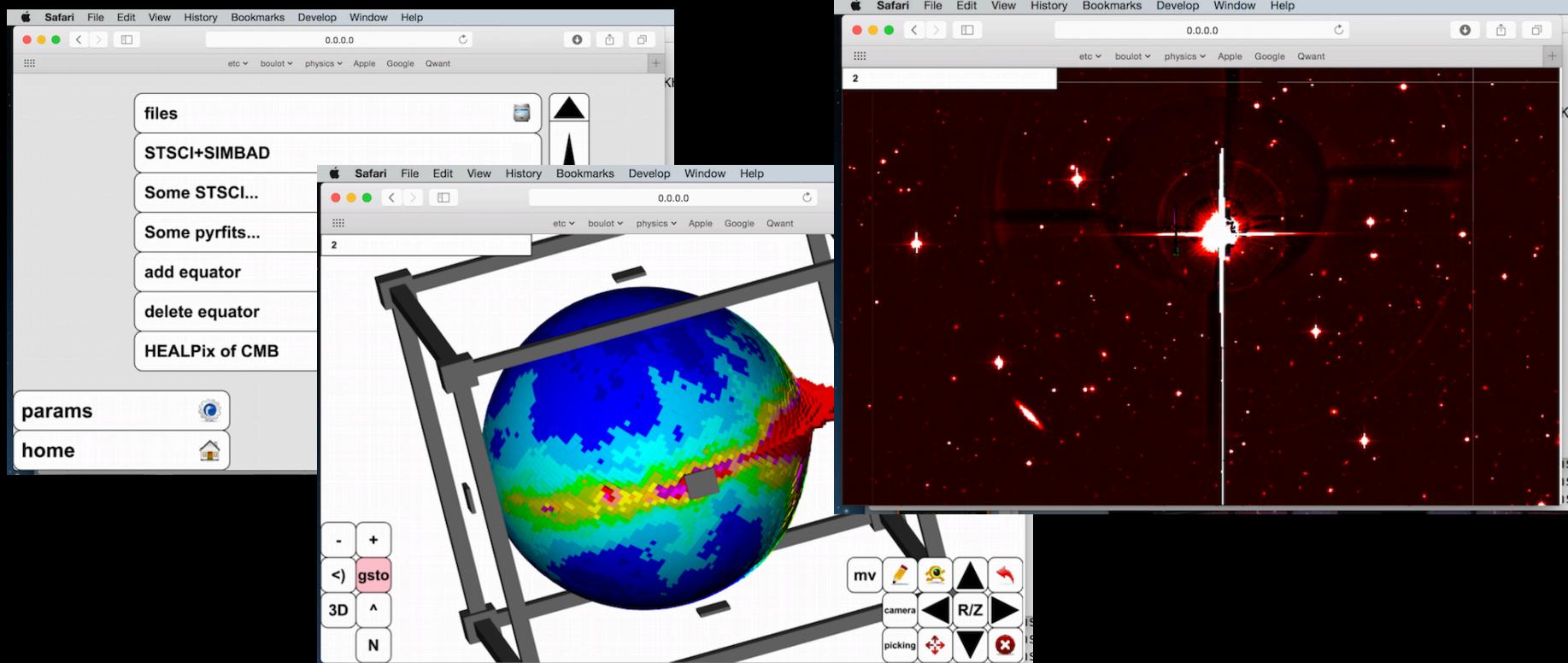


G.Barrand, CNRS/IN2P3/LAL

Pas de Web mais...

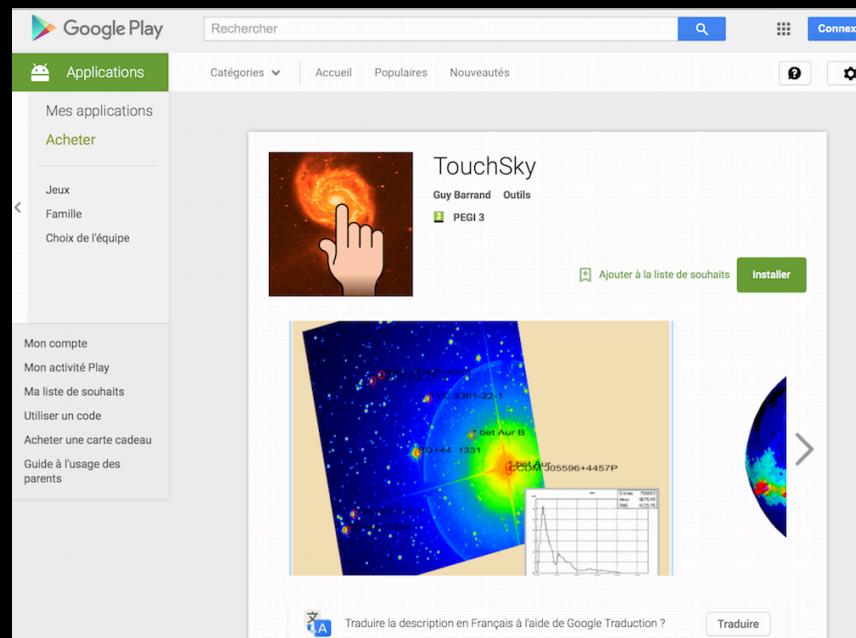
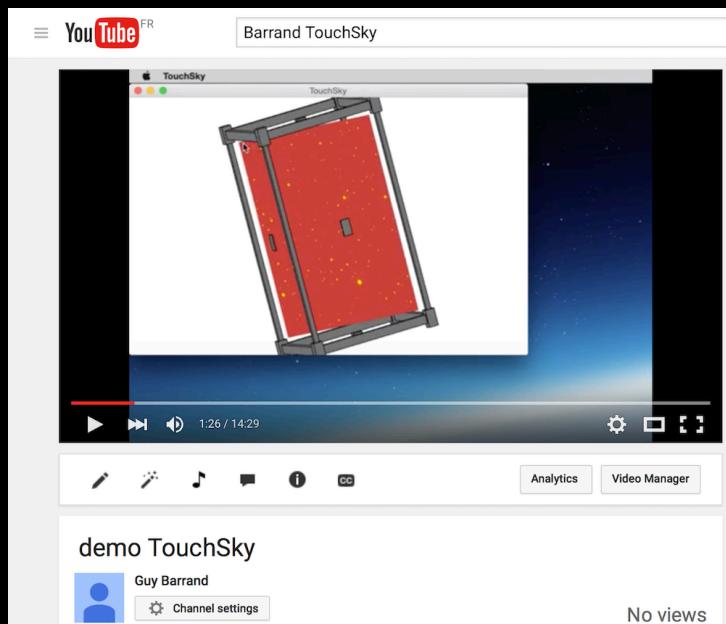
En fait notre gestionnaire de graphe de scène possède un driver WebGL.... Quelques vues de TouchSky dans Safari...

« Ça marche » mais c'est nettement moins réactif que l'appli en direct sur un device local !



Demos

- YouTube : chercher « Barrand TouchSky »
- <http://softinex.lal.in2p3.fr/download/TouchSky>
- Breaking news : TouchSky on GooglePlay !



Conclusions

Gros progrès depuis 2014, ça avance !

L'architecture se dessine. Des questionnements sur :

- Où, quand et sur quelles données on construit les pyramides ? Cela va prendre plus de trois secondes et des Gigas pour une pyramide, cela ne peut pas se faire par de l'interactif depuis les visualiseurs locaux...
- Où va se faire le déploiement des serveurs ?
- Des sous ?
 - Pour une mise à jour matériel du LAL/murino.
 - Achat d'une Microsoft Surface Pro 4. (Redmond vrai concurrent d'Apple dans le futur pour ce qui est de la visu et de l'interactivité ?)