# *Apple/Metal (hurlant)*

IJCLab réunion développement
15 Septembre 2020

# *2018 :-(*

WWDC Juin 2018 : Apple, dans une session //, annonce l'obsolescence de leur OpenGL.

# *2018 :-(*

- Mauvaise nouvelle pour les gens voulant un standard pour faire de la visualisation.
- Mauvaise nouvelle pour moi et Geant4, et pour beaucoup de logiciels scientifiques.
- À cause de l'impact d'Apple sur les questions d'interactivité, on ne peut pas ignorer cela…
- Apple fait la promotion de leur librairie propriétaire Metal en remplacement de leur OpenGL sur leurs appareils. On doit regarder !

- (Pas de date donnée pour un enlèvement en dur d'Apple-OpenGL sur macOS et iOS).

# *logique de graphes de scène inlib/exlib/sg*

- Dans l'esprit, même logique que le grand OpenInventor.
- Une scène est décrite par un graphe de "nodes" dans lequel, par exemple, un cube rouge tourné est décrit par un "matrix node", un "color node" et un "shape node".
- Un graphe est rendu sur l'écran (ou offscreen !) en utilisant l'implémentation d'un "renderer" pour une technologie de graphique donnée, par exemple GL-ES.
- Voir softinex à http://gbarrand.github.io

# *inlib/exlib/sg renderers*

- GL-ES : cela permet (aujourd'hui) avec le MÊME CODE de visualiser sur Linux, macOS, Windows, iOS, Android.
- offscreen : pour produire un fichier .png, .jpeg, .ps, .pdf sans avoir à se lier à aucun système graphique (c'est du pur code C/C++ basé sur les librairies STD/STL).
- wasm : une version web assembly (utilisant WebGL). Cela permet de visualiser sur tous les web browsers du moment (y compris sur iOS et Android).
- Donc je dois fair un renderer pour Apple/Metal…

# *Pas si facile que ça à faire !*

- L'API est en Objective-C ou en Swift (le langage "meilleur que Python", dixit Apple :-) ).
- Les exemples d'Apple sont en Swift constructibles avec Xcode.
- Rien en C++ constructible depuis un "simple makefile".
- Bloqué…

# *…jusqu'à la fin du mois de Juin 2020*

- Un nième googling désespéré m'a donné un hit sur GitHub : naleksiev/mtlpp
- mtlpp : un C++ wrapper pour Metal
- avec, important, un exemple pour dessiner un triangle constructible avec make : bingo!

- (Comme dit un vieux (ben oui) proverbe LALois : donnez moi un triangle et je visualiserai le monde).

# *Été 2020 à la forge…*

- Après deux mois de codage très pénible, j'ai maintenant une application (l'event display pour ESSnu) qui marche sur macOS.
- Et ceci en utilisant directement l'API Objective-C de Metal depuis du C++ (Apple clang permet de mixer les deux langages).
- Et ce sans librairies tierces.
- (Cela suit mon "software least action principle").
- Pénible car la logique de Metal n'est pas la même que GL-ES, même si les idées de "rendering pipline", buffers", etc… sont les mêmes. On doit repenser un nouveau renderer (ce qui n'était pas le cas pour les renderers offscreen et wasm).

# *Été 2020 à la forge… (2)*

- J’ai maintenant un rendu 3D pour les primitives de base (points, lignes, segments, triangles, triangle-fan and strip).
- J’ai l’éclairage. (Je vois la lumière : enfin !)
- J’ai aussi le texture mapping.
- (Rappel : pas de gestion de texte, ni dans OpenGL, ni dans Metal)
- Avec cela mes apps fonctionnent (y compris le GUI, puisque fait avec inlib/sg)

Et soyez sûr que cela n’a pas été facile à avoir !

# *Et donc "ouf" !*

- Je dois faire une version iOS (windowing API pas pareil que sur macOS)
- En Principe je suis prêt pour ce qu'Apple nous prépare dans le futur.
- (Je suppute fortement qu'ils vont enlever leur OpenGL lors de la mise à jour macOS supportant leur processeur Ax pour les Macs).

# *Est ce que cela peut aider pour Geant4 ?*

- Mes apps g4exa, g4view devraient tourner avec Metal, et je vais faire des mises à jour pour celles ci.
- Mais elles ne sont pas basées sur le “G4 vis system” largement utilisé world wide.
- Le G4/vis système est en principe conçu pour gérer plusieurs systèmes graphiques hétérogènes (= drivers).
- Par exemple il y a un driver OpenInventor et des drivers offscreens (HepRep, VRML).

# *Est ce que cela peut aider pour Geant4 ? (2)*

- Pour le moment ce qui est promu par Geant4 est Qt pour le GUI et le “vieux” OpenGL (même pas GL-ES) pour le graphique.
- Pas si clair comment avancer…
- Peut-on mixer du Qt avec QWidget faisant du Metal ? Cela demanderait une programmation spéciale pour macOS/iOS : beaucoup de travail.
- Une librairie Qt-vis ayant un QMetal renderer en dessous pour macOS, iOS? Cela demanderait d’avoir un G4 “Q-vis-lib” driver : beaucoup de travail.
- Un driver inlib/exlib ? beaucoup de travail. (Sachant que mes très grandes vacances pointent à l’horizon).

# *En attendant…*

- Je vais releaser mon code et apps. (Il y a probablement très peu d'applis scientifiques C++ tournant directement sur Metal en ce moment).
- Cela mériterait un papier dans un futur C(OVID)HEP (probablement mon dernier papier "R&D").
- Ce qui est sûr c'est que suis impatient de avoir ce qu'Apple nous réserve dans les prochaines années…
- Et à ce propos, il y a keynote ce soir à 19h…