WebAssembly (pour notre physique ?)

Groupe énergie noire

16 Décembre 2019

Guy Barrand, CNRS/IN2P3/LAL

WebAssembly

- Une idée, technologie intéressante pour passer sur le Web une application en C++, qui fait de l'OpenGL pour son graphique, et ce sans avoir à faire et déployer un serveur spécial.

Le secret

Dans vos navigateurs de toile, il y a une machine virtuelle ! (si, si)

(une « portable virtual stack machine », dixit Wikipedia)

Comment ça marche ?

- Dans vos navigateurs de toile, il y a une machine virtuelle !
- On installe sur son mac adoré, ou son Linux bien-aimé, ou son Windows abhorré, la boite à outil « emsdk ». (em est pour « emscripten ». (Aucune idée de ce que ça veut dire)).
- On cross compile son appli avec « em++ » pour fabriquer un « .wasm » (binaire), quelques .js et un index.html. (em++ utilise clang et LLVM).
- On déploie {index.html, .wasm, .js} dans des pages statiques chez n'importe quel hébergeur (par exemple gbarrand.github.io pour moi).
- Donc pas besoin de déployer un serveur spécial.

Comment ça marche (2) ?

- Lorsqu‘on charge le index.html depuis son navigateur, le .wasm est chargé et executé dans la machine virtuelle, donc sur votre machine en local.

ET VOILA !

Graphique?

- La base est de faire du WebGL. Donc on a du 3D.
- Il y a une pauvre implémentation de GL-ES-1 sur WebGL dans la boite à outil emsdk.
- Je peux faire tourner mes applis ! ☺ ☺ ☺
- (En particulier du fait que je fais mon interface utilisateur aussi avec GL-ES).

Physique : analyse, astro

- ioda fonctionne : et donc mon plotting, une partie de SophyaLib, cfitsio, hdf5, lua et mon lecteur de fichier .root.
- Une partie de TouchSky : et donc une visu de HEALPIX et de projection mollweid.
- MAIS, du fait de limitations de emsdk sur les sockets, je ne peux pas faire pur l'instant des requêtes en synchrone sur STSCI ou le CDS.

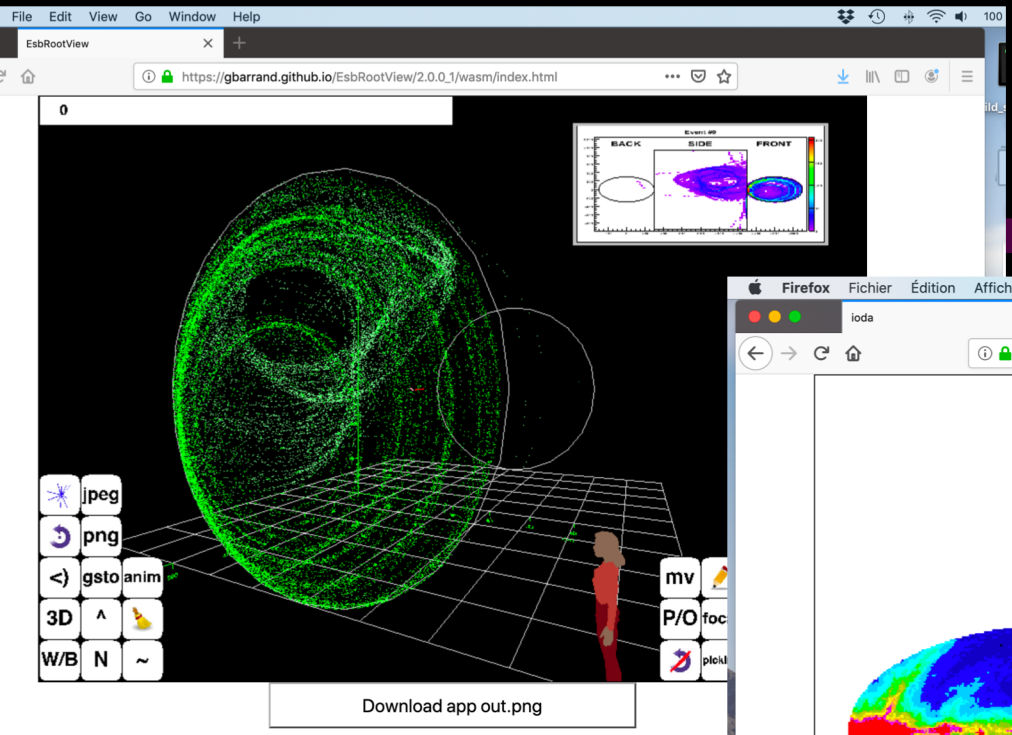
- Ce qui marche aussi est de pouvoir uploader un fichier de données de la machine locale à la machine wasm. Et de pouvoir aussi récupérer un fichier produit sur la machine wasm sur la machine locale.

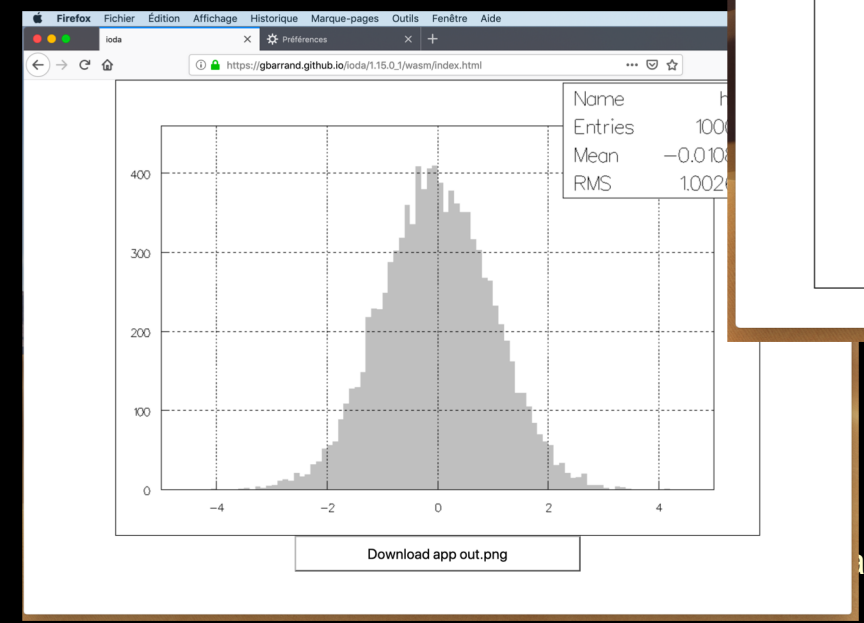
Physique : HEP

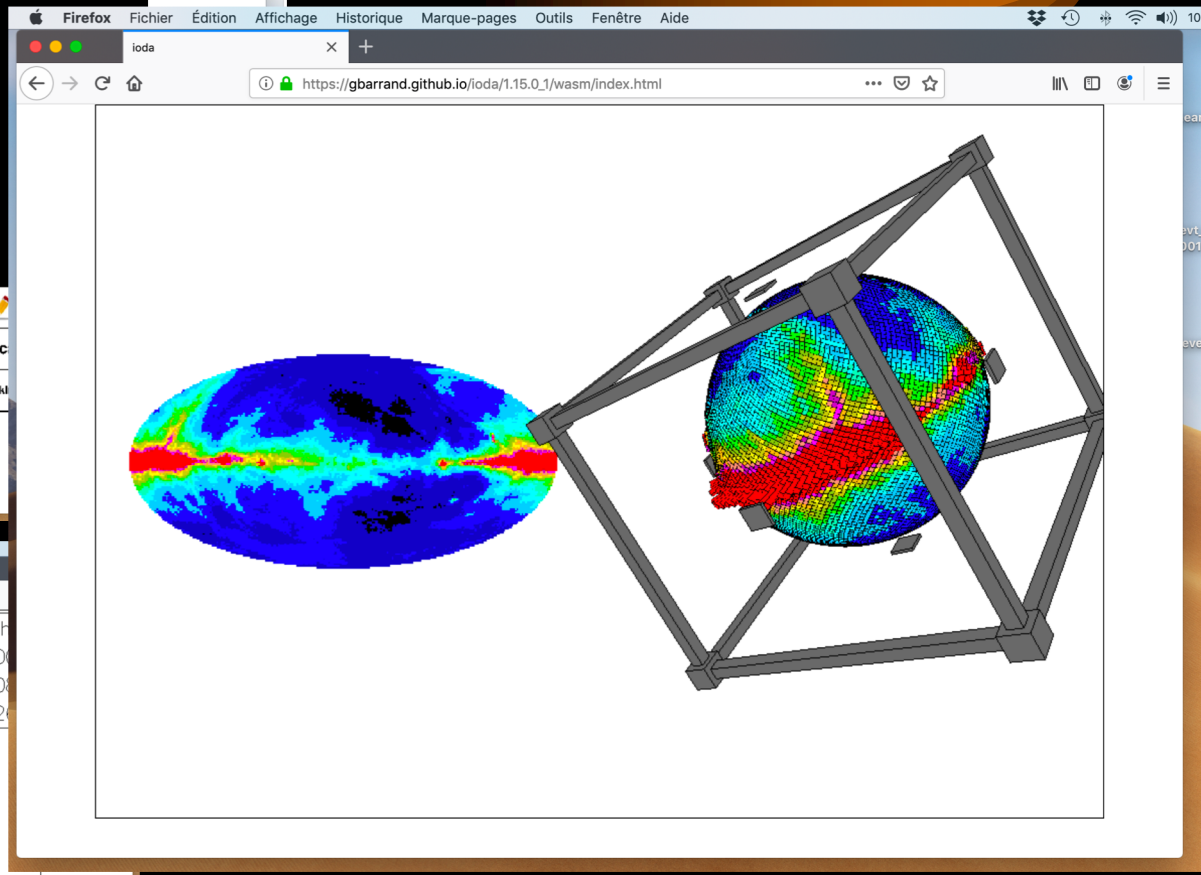
- J‘ai pu passer g4view, g4exa : et donc avoir Geant4 en .wasm.
- Je peux lire des fichiers .root de geométrie ou d‘analyse.
- pmx marche : donc un embryon d’event display d’LHCb (pour montrer, encore et toujours (soupir), à HEP et au CERN qu’on peut faire du soft potable portable).
- EsbRootView : display pour de la R&D sur ESSnuSB (accélérateur à Lund). (Une suite inattendue d’une visu MEMPHYS qu’on avait fait avec Jean-Eric il y a… une quinzaine d’année).
- Tout cela est essayable depuis gbarrand.github.io, depuis les sections « wasm (experimental) » de mes applications. Prendre Firexfox ou Chrome pour l‘instant.

Mais ☹☹☹

- Ça marche pour moi avec :
  - Firexfox et Chrome sur mon mac (donc adoré).
  - Firefox sur des VM Linux (centos7 et MINT).
- MAIS, marche mal sur :
  - Safari (qui impose une limite sur la mémoire utilisée).
  - Tout navigateurs sur iOS et Android (canvas noir, à comprendre).
  - Firefox sur mon Windows abhorré (points pas dessinés !)
  - Chrome sur Windows (démarre pas, ici aussi limite sur la mémoire).
- Et avec GL-ES de emsdk :
  - glLight pas implémenté.
  - VBO (utilisation du GPU) est buggé.


File Edit View Go Window Help
EsbRootView
https://gbarrand.github.io/EsbRootView/2.0.0_1/wasm/index.html
0
BACK
SIDE
FRONT
jpeg
png
gsto
anim
3D
W/B
N
mv
P/O
Download app out.png


Firefox Fichier Édition Affichage Historique Marque-pages Outils Fenêtre Aide
ioda
Préférences
https://gbarrand.github.io/ioda/1.15.0_1/wasm/index.html
Name
Entries
Mean
RMS
400
300
200
100
0
−4
−2
0
2
4
Download app out.png


Firefox Fichier Édition Affichage Historique Marque-pages Outils Fenêtre Aide
ioda
https://gbarrand.github.io/ioda/1.15.0_1/wasm/index.html

WebAssembly ☺☹

- Enfin une connexion « C++ GL-ES » avec le Web intéressante !
- MAIS pas mal de problèmes encore autour du graphique. (Il faut probablement faire du WebGL en direct et zapper le emsdk/GL-ES).
- MAIS on sent bien, à l'usage, que les navigateurs de toile sont clairement « pensés » pour exécuter des « petites tâches de manière asynchrones », et pas pour tourner des grosses tâches synchrones.
- Certaines navigateurs bloquent ces tâches.
- De toute façon le graphique, est moins réactif qu'en « pur local ».
- Mon sentiment est que le wasm sera très bien pour de l'outreach ou des tâches de physique très ciblées et n'embarquant « que ce qu'il faut ». Et qu'on doit encore cibler de tourner en local pour les « grosses applis » (bref, rester proche du silicium).
- Demo….

Wasm et python

- Visiblement on peut cross compiler aussi pour Go(Sport), C#, Java et Python.
- Pyodide : « bringing the python scientific stack to the browser ».