WebAssembly

G4 virtual workshop 2021

Guy Barrand, CNRS/IN2P3/IJCLab

WebAssembly

An interesting technology to have on the web a C++ application, that does WebGL for its graphics, and this without having to deploy a specific web server.

The secret

In your web browsers, there is a virtual machine (some kind of)! (yes, yes):

the wasm

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

How it works?

- In your web browsers, there is a virtual machine!
- You install on your beloved mac, or your preferred Linux, or your forever-upsetting Windows, the « emsdk » toolkit. (em is for « emscripten ». (No idea of what it means)).
- You cross-compile your app with « em++ » to build a « .wasm » (binary), some .js and one index.html. (em++ uses clang and LLVM).
- You deploy {index.html, .wasm, .js} in static web pages in any web host (for exemple gbarrand.github.io for me).
- Then no need to deploy a specific server.

How it works (2) ?

When you load the index.html from your web browser, the .wasm is loaded and executed in the virtual wasm machine, then on your local machine.

ET VOILA !

Graphics?

- The main idea is to use WebGL. Then we have 3D.
- There is a poor implementation of GL-ES-1 on WebGL in the emsdk toolkit, and I do not recommend it (it is very incomplete and bugged).
- WebGL: from the C++ we do then “string programming” = we build javascript code which is then passed to the browser to be compiled and executed, and that itself will finish to use compiled OpenGL.
- Then we have obviously an inefficiency compared to the same app compiled and run locally and using straight the compiled OpenGL.

My apps

- C++98: ok with em++. What I use of STL/STD works here.
- My scene graph logic is ok too. (And then my plotting too).
- My code to read .root files is ok.
- Mastering of the externals: I bring, since ever, the code of my critical externals: freetype, expat, zlib, jpeg, png. (There is nothing of these coming with the emscripten SDK!). And these are ok too.
- I did the port of cfitsio, hdf5.
- Geant4-10.03 core is ok!
- I can run my apps! ☺☺☺
- In particular due to the fact that I do my GUI with my scene graphs logic, and then in WebGL (unified graphics).

To be known

- It is in 32 bits.
- Usage of sockets is forbidden. To do some http, someone has to pass by the browser that will do the requests asynchronously: it complicates the life.
- We can upload a local data file in the wasm. We can also download a file from the wasm (for exa a .png) on the local machine.
- There is no true file system in the wasm, but we can encapsulate files (for exa fonts, icons) in a .data container seen by the standard C API (fopen, etc…)
- The windowing is done in a HTML canvas and can be inserted in any web page. In particular we can mix with some GUI done in javascript.

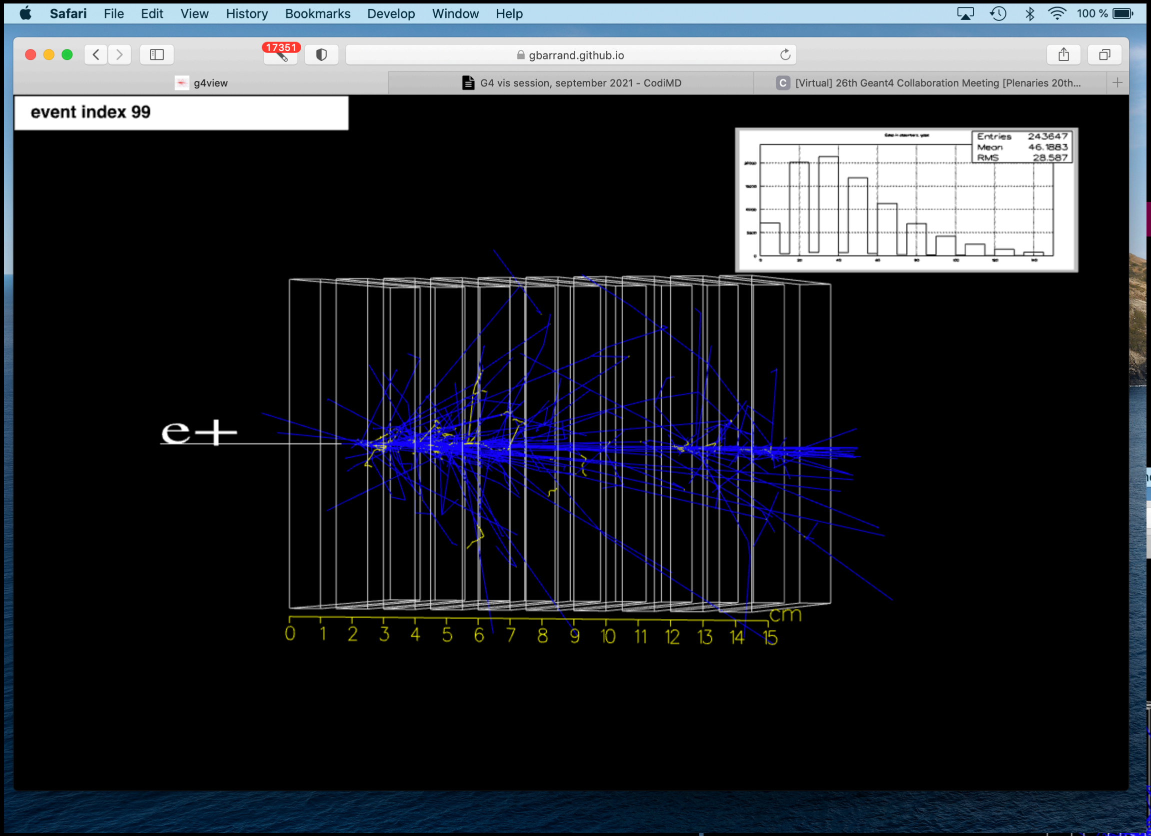
Physics: Geant4, HEP

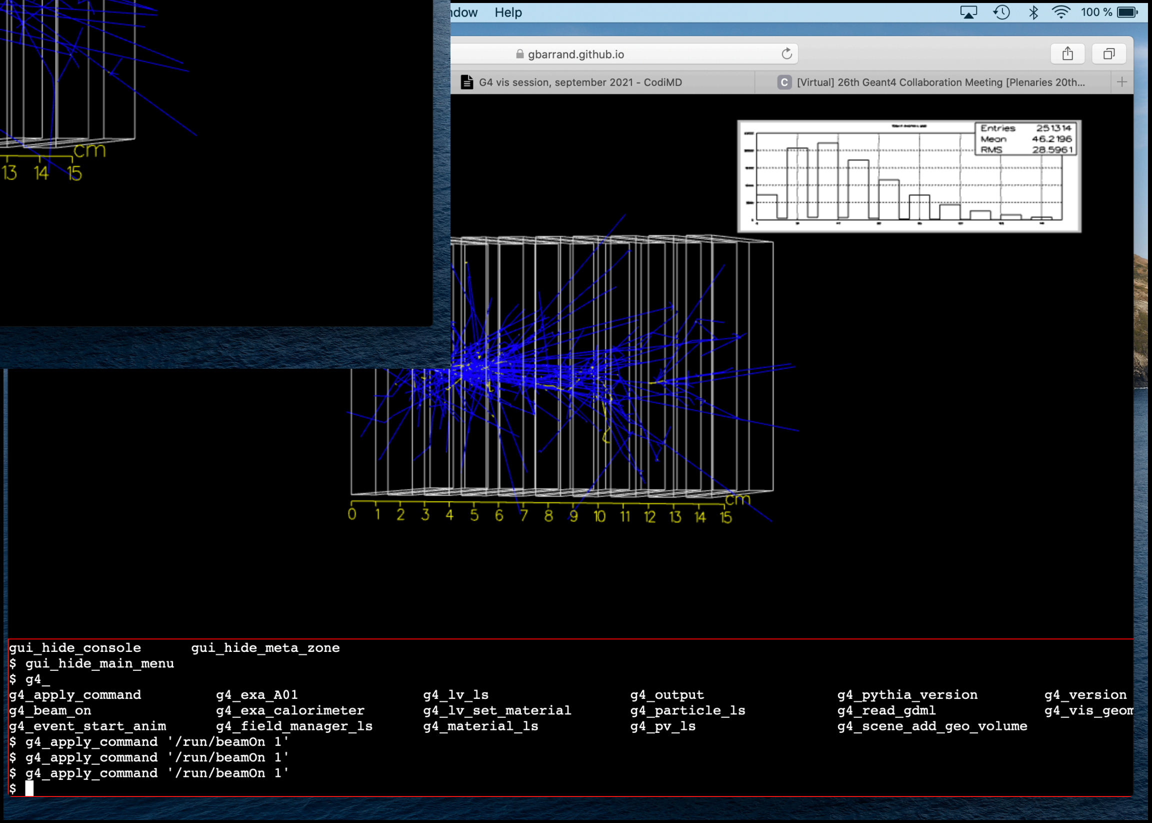
- I ported g4view, g4exa: and then have Geant4 (10.3) in .wasm!
- I can read .root files with geometries, histos and ntuples.
- pmx is ok: then an embryo of LHCb event display (to show, again and again, that we can do highly portable C++ HEP apps without having to embark… the rest of the world).
- EsbRootView: R&D display for ESSnuSB (ESS is an accelerator at Lund). Presented at vCHEP-2021 in May. There is now a paper.
- I have a terminal mode (done with xterm.js) to type "insh" commands, or… G4 commands, then from the web browser!
- All these are testable from gbarrand.github.io, under the sections « WebAssembly » of each apps.

It works…

- It works for me with Safari, Firefox and Chrome on my (beloved) Mac, on iOS and Android, on Windows-10 and on Linux VMs (at least centos7 and ubuntu).
- WARNING: {.wasm, data related to the app} can be big, then the loading at startup could be slow on a poor connection from a remote web host.
- But soon everybody will be on 5G :-)
- For Geant4 apps, problem with the data files. (In g4exa, g4view I bring only what is needed). (Problem if deploying on github that limits file size to 50 Mbytes. I have to find a way for that. Hmmm, would be great to arrange that G4/processes get themselves their needed files through the web!)

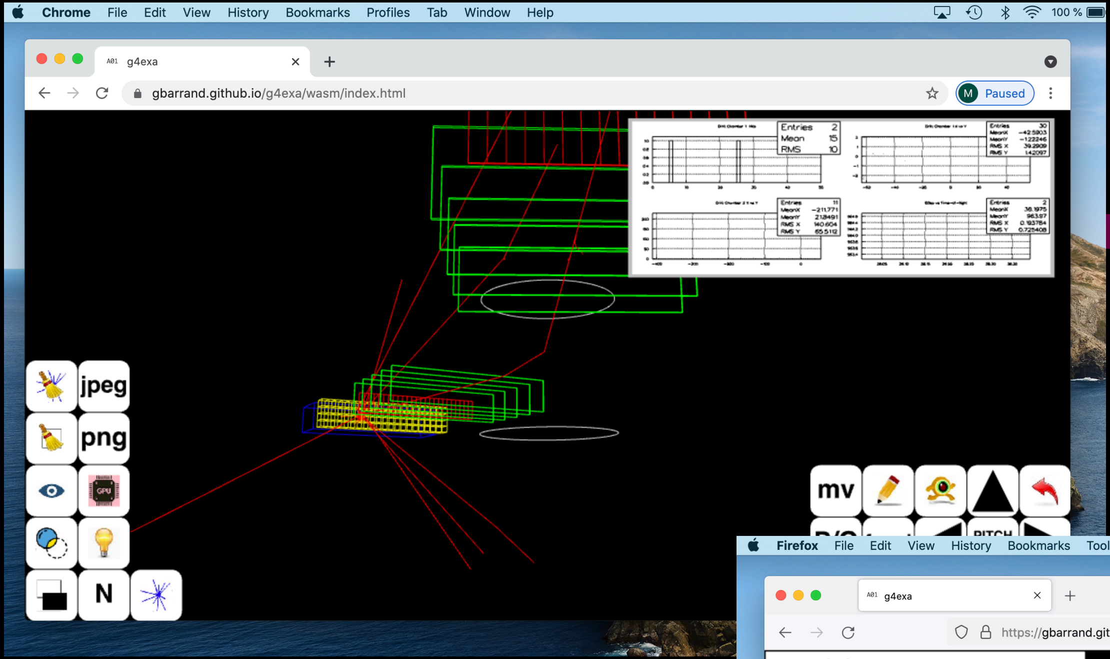
g4view

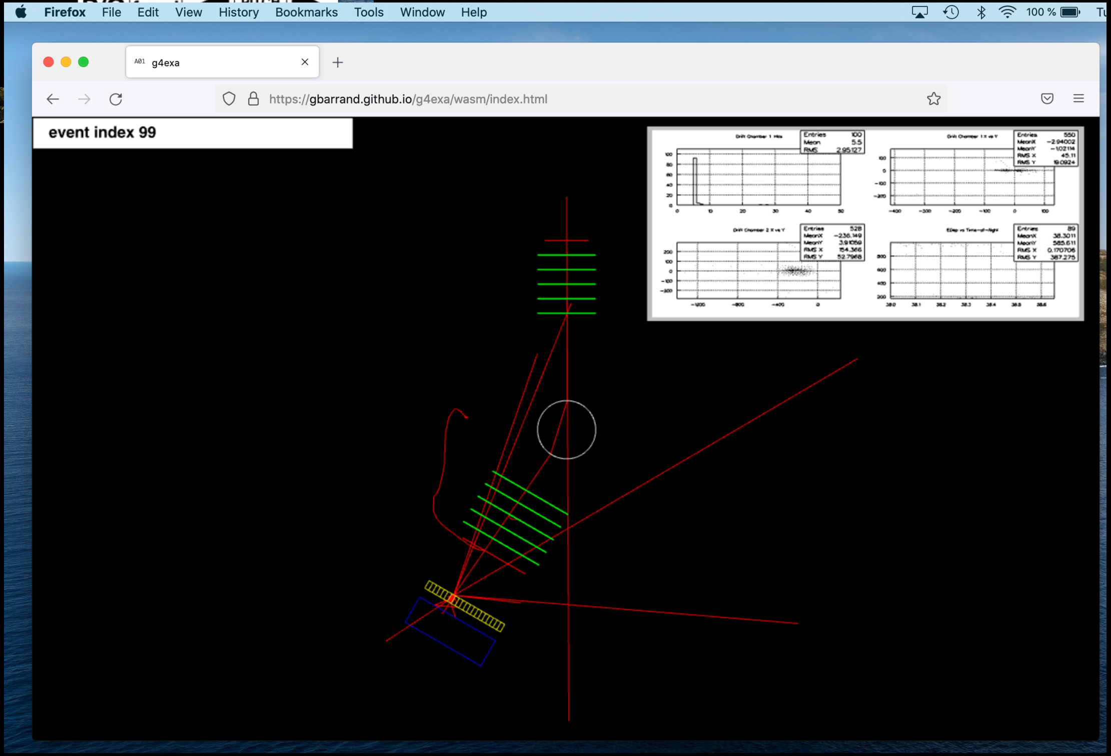



gbarrand.github.io
G4 vis session, september 2021 - CodiMD
[Virtual] 26th Geant4 Collaboration Meeting [Plenaries 20th...
Entries 251314
Mean 46.2196
RMS 28.5961
cm
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

```
gui_hide_console      gui_hide_meta_zone
$ gui_hide_main_menu
$ g4_
g4_apply_command         g4_exa_A01               g4_lv_ls                 g4_output                g4_pythia_version        g4_version
g4_beam_on               g4_exa_calorimeter       g4_lv_set_material       g4_particle_ls           g4_read_gdml             g4_vis_geom
g4_event_start_anim      g4_field_manager_ls      g4_material_ls           g4_pv_ls                 g4_scene_add_geo_volume
$ g4_apply_command '/run/beamOn 1'
$ g4_apply_command '/run/beamOn 1'
$ g4_apply_command '/run/beamOn 1'
$ 
```

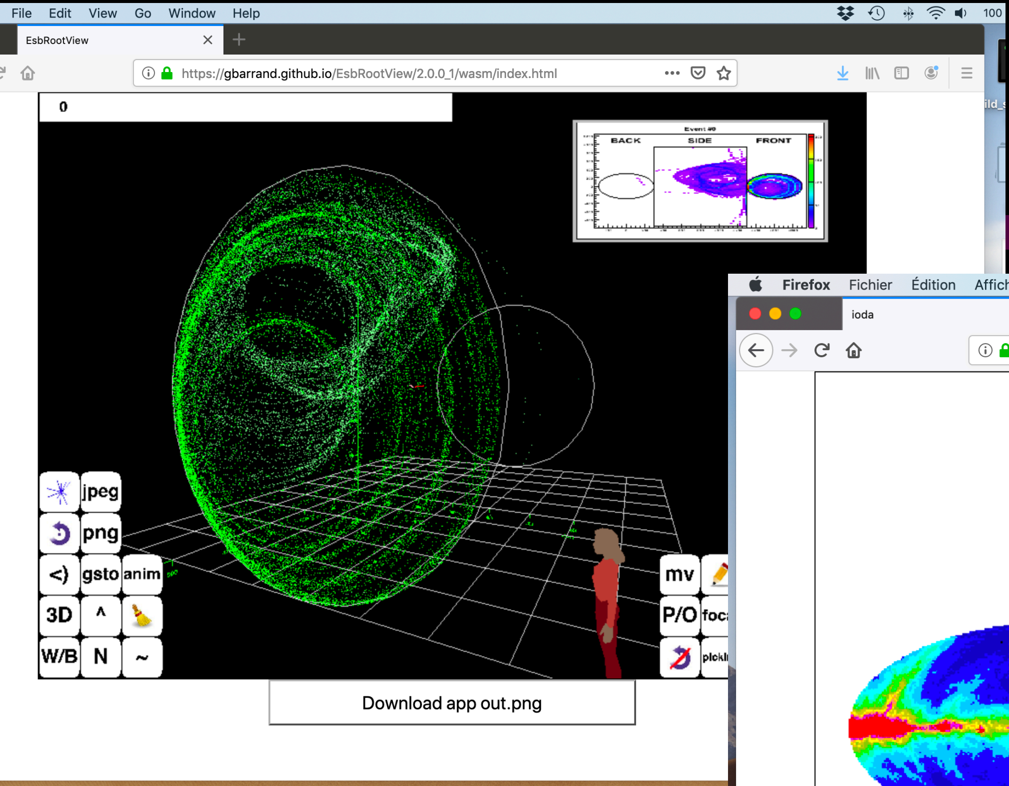
g4exa

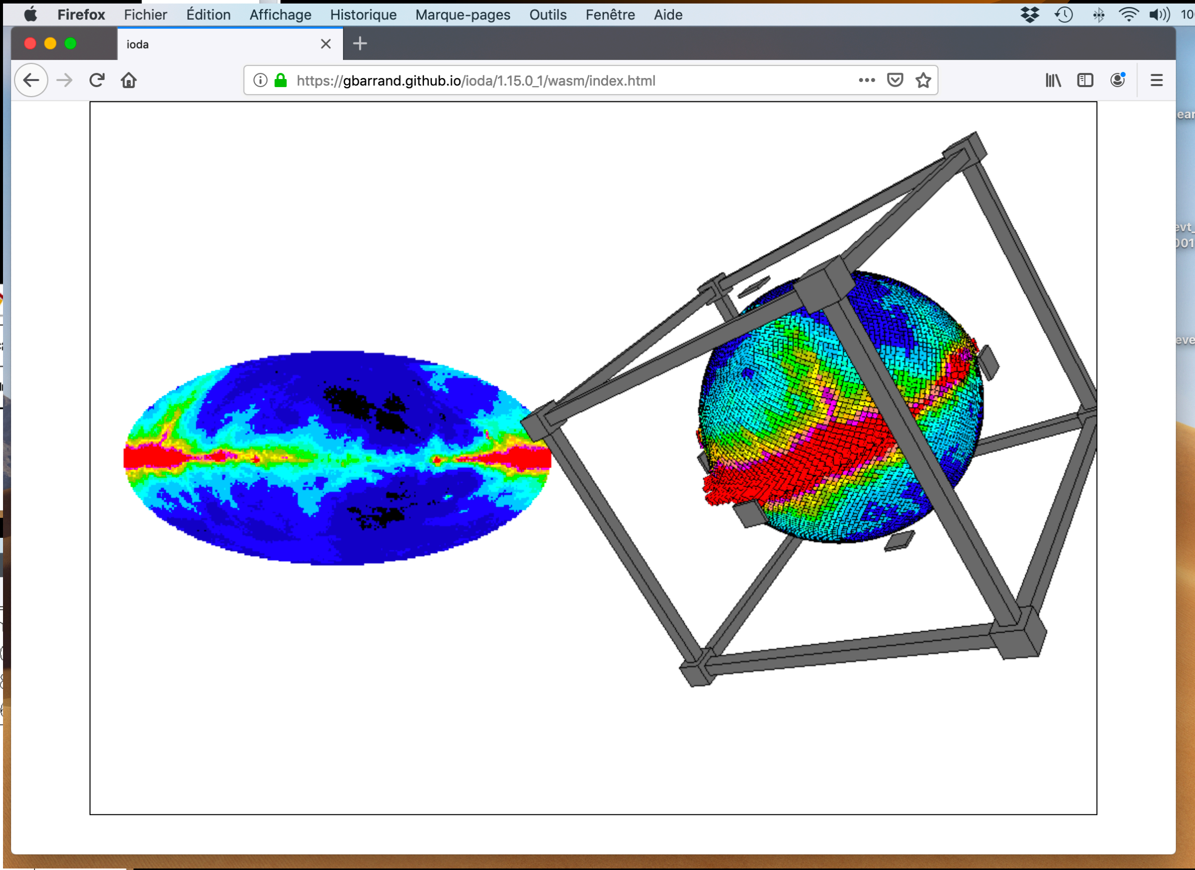

Chrome File Edit View History Bookmarks Profiles Tab Window Help
g4exa
gbarrand.github.io/g4exa/wasm/index.html
Paused
jpeg
png
N
mv
PITCH

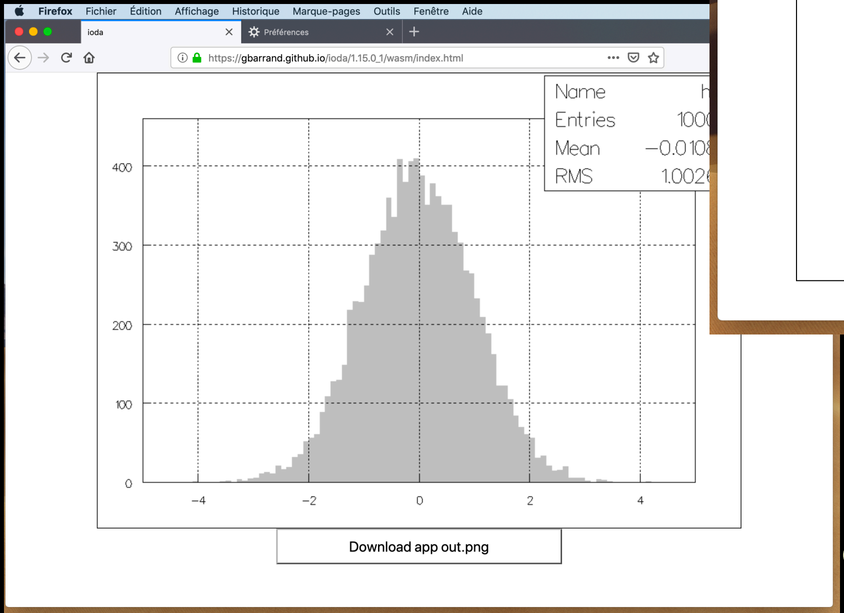

Firefox File Edit View History Bookmarks Tools Window Help
g4exa
https://gbarrand.github.io/g4exa/wasm/index.html
event index 99

Guy Barrand, CNRS/IN2P3/IJCLab

Etc..


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
https://gbarrand.github.io/ioda/1.15.0_1/wasm/index.html


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 100
Mean −0.010
RMS 1.002
400
300
200
100
0
−4
−2
0
2
4
Download app out.png

Conclusions ☺☹

- At last an interesting connection « C++/WebGL » with the Web!
- (Then no need to rewrite everything in javascript just for the web).
- Due to my “strategical choices” my C++ apps run here.
- BUT I feel, at usage, that the web browsers are though more to execute « little tasks asynchronously », than to run big synchronous tasks.
- Some web browsers (Safari) block these kind of tasks.
- Anyway the interactivity is less reactive that in « pure local ».
- My feeling is that the wasm would be just great to do outreach or highly targeted physics tasks bringing « only what is needed » with not so much needs in graphics. I think hat we must continue to fight to run locally so that « big apps » stay close of the silicium.

Demos…