

The benefits of particle transport simulations in a web browsers.

Leszek Grzanka^{1,2,3}, Szymon Kania², Jakub Niechaj²

1 IFJ PAN Kraków, 2 ACK Cyfronet, 3 AGH Kraków

GATE Scientific Meeting 2023 25.04.2023

GATE and Jupyter Notebooks

Jupyter Notebooks eliminate the need for console use and provide a more accessible interface for GATE users.

Access GATE from any web browser with the convenience of Jupyter Notebooks.

However, users may still face challenges with installation, including downloading Python and large GATE binaries.

Compatibility issues with different operating systems (e.g. Windows, Mac OSX with M1 CPUs) may also be a concern for users.



Full web migration

Apps that run on the internet

Accessible through a web browser

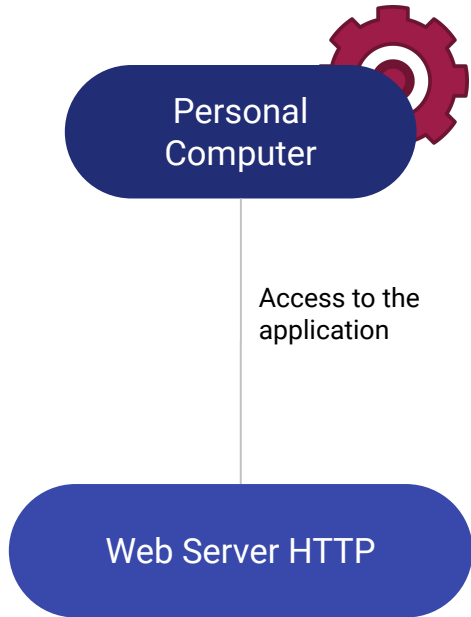
No installation or console required

Works on any OS, easy to use and update

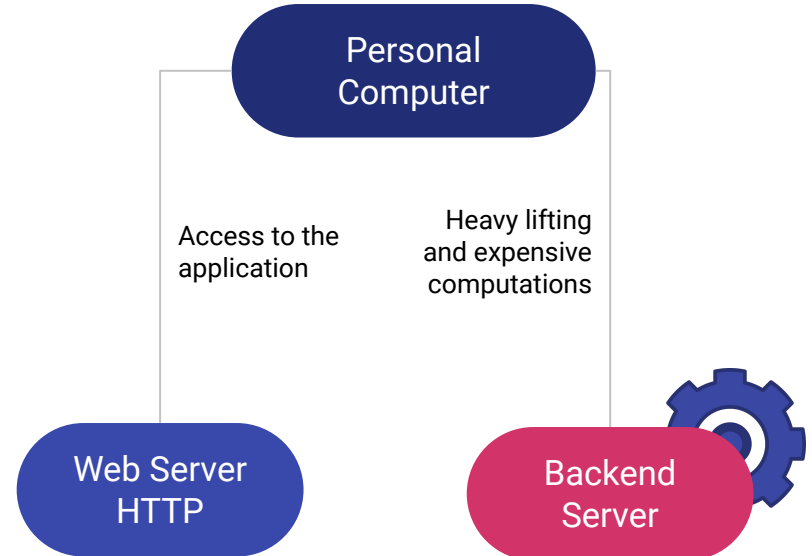


Types of Web Applications

Fully client side applications



Client + backend server applications

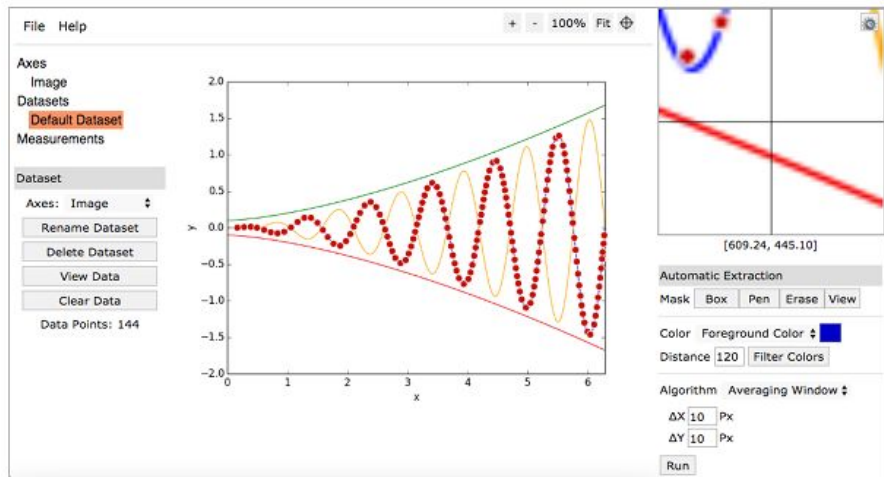


Types of Web Applications

Fully client side applications



WebPlot Digitizer



Client + backend server applications



Google Collab

File Edit View History Tools People Help

Hello, Collaboratory x Overview of Colab x

Secure | <https://colab.research.google.com/notebooks/welcome.ipynb#scrollTo=9J7p406abzgl>

Hello, Collaboratory

File Edit View Insert Runtime Tools Help

CODE TEXT CELL CELL COPY TO DRIVE DISCARD CHANGES CONNECT

Table of contents Code snippets

Welcome to Collaboratory!

GPU Support (NEW!)

Python 3

TensorFlow execution

Visualization

Welcome to Collaboratory!

Collaboratory is a Google research project created to help disseminate machine learning and research. It's a Jupyter notebook environment that requires no setup to use and run the cloud.

Collaboratory notebooks are stored in [Google Drive](#) and can be shared just as you would Docs or Sheets. Collaboratory is free to use.

For more information, see our [FAQ](#).

Key benefits of running simulations in a browser

1 No installation or setup required

There is no need to install any software on your computer, making it a convenient option for users with no technical experience.

2 Small scale testing and improvements

Users can test their input and optimize their parameters on a small scale before submitting larger jobs to high-performance computing systems.

3 Easy to share or present results

Scientists can easily share simulation results and the input used to generate those results, which can foster collaboration and more efficient research.

4 Benefits outreach and education

Prepared examples can be used to showcase the simulations, making it easier for people to understand the concept and get started with using the tool.

How can we go full client side?

Javascript

- High level programming language designed to manage content for websites (animation, form submit etc.)
- Modernly used to create web applications of various sizes. From simple games to powerful editors.
- Performance is limited by web browser and you can't easily do parallel computing.

WebAssembly

- Low level binary format that is designed to be more efficient than JavaScript and can also run in a web browser.
- Application usually written first in other language.
- Application has to be downloaded every time when you (re)open tab in a browser.

How can we go full client side?

Javascript

T-Rex Chrome



WebAssembly

WASM Doom



Early attempts

Compiling Geant4 to WebAssembly

Single ExampleB1

Google Summer of Code

Saurav Sachidanand

2018

<https://medium.com/@saurvs/compiling-geant4-to-webassembly-cb124b75600d>

CernVM-FS

POSIX read-only I/O on CernVM-FS

Reduced app size

2018

<https://github.com/cvmfs-contrib/cvmfs-emscripten>

Softindex - g4view

Multiplatform toolkit

Guy Barrand

<https://github.com/gbarrand/g4view>

Performance of Geant4 in WASM

Text based input

Szymon Kania

2023

<https://github.com/ostatni5/geant4-wasm-performance>

Our early results

Single process Native vs WebAssembly in browser

Initialization time ~x2 slower than Native

Simulation time ~x2.7 slower than Native

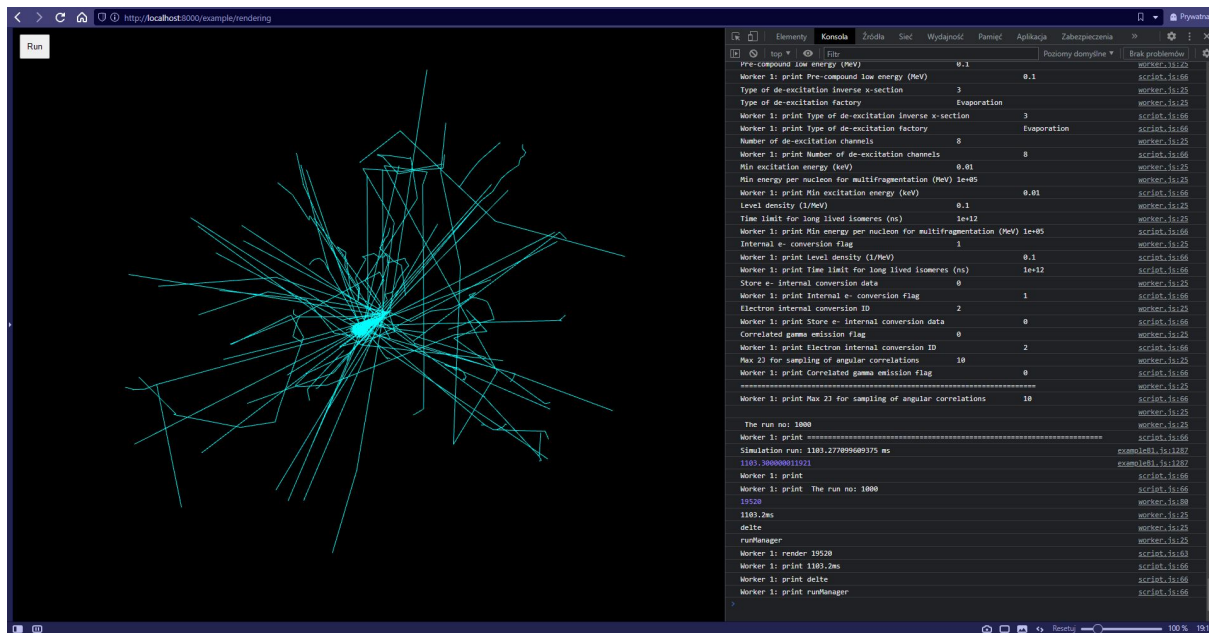
Scaling of time with problem size almost the same as Native

	Native	Wasm
Init:	~1.8s	~3.5s
Sim:	~5.7s	~13s
Init + Sim:	~7.6s	~17s

Szymon Kania - <https://github.com/ostatni5/geant4-wasm-performance>



Our visualization attempt



Data processing requires a lot of work

Differences in data format implementation could impact performance

Yaptide - research platform

The aim:

To provide a platform assisting MC simulations.

Focus on:

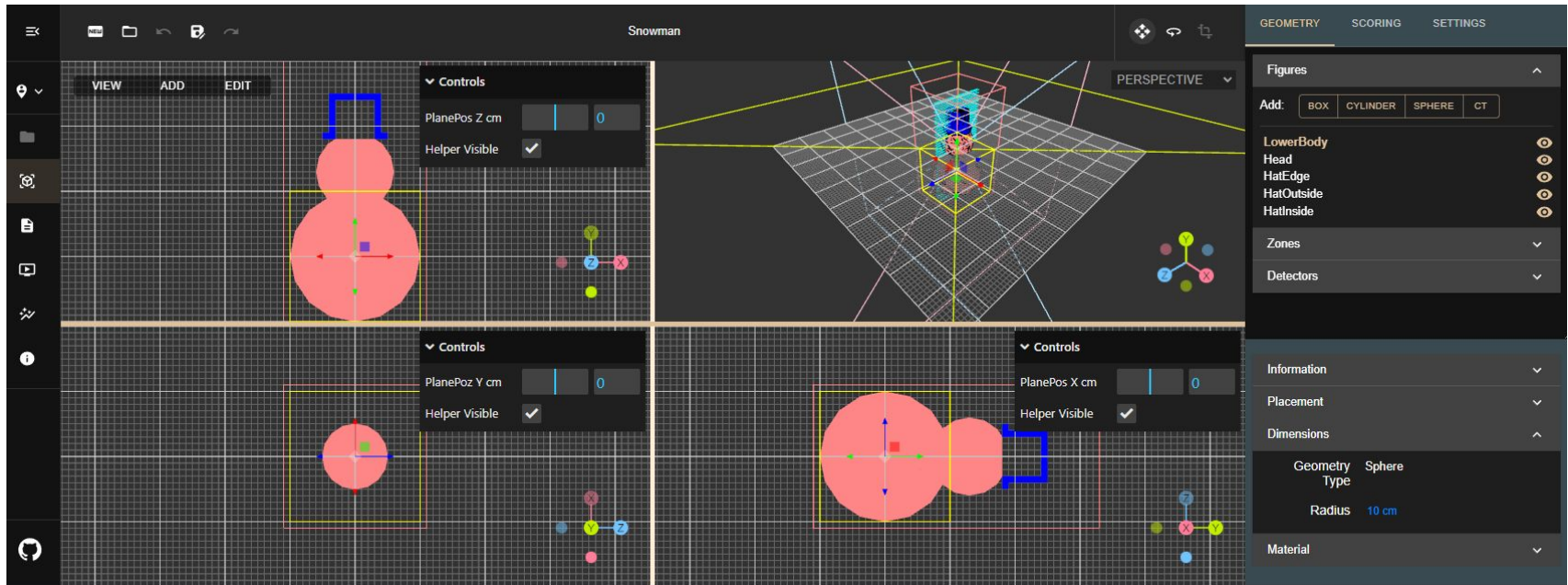
- advancing hadrontherapy
- particle transport simulation

Main features:

- Simulations of particle interaction with matter
- Pushing forward research in medical physics: optimising not only dose
- Evaluation of treatment plan quality
- In-silico studies for design of experiments

Yaptide - research platform

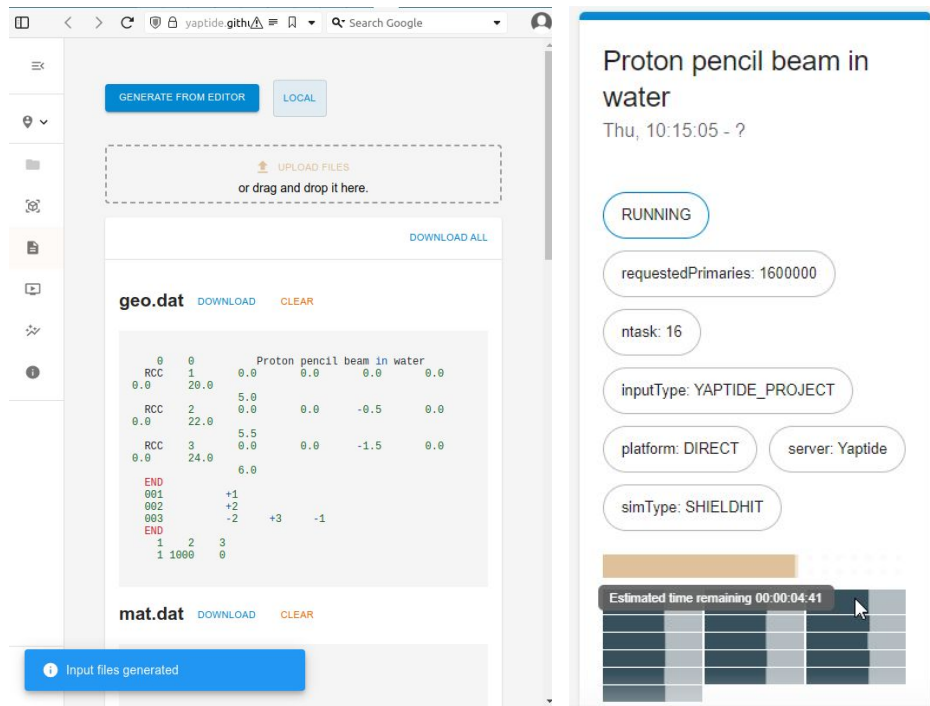
https://yaptide.github.io/web_dev/ - development version



Yaptide - research platform

Overview:

- no authentication needed
- works on github-pages
- input files can be manually executed locally or on **HPC**
- ongoing work to support **TOPAS** and **FLUKA MC** codes
- interactive results preview using **JSROOT** library



The screenshot displays the Yaptide web interface. The main content area shows a simulation job titled "Proton pencil beam in water" with a status of "RUNNING". The job details include "requestedPrimaries: 1600000", "ntask: 16", "inputType: YAPTIDE_PROJECT", "platform: DIRECT", "server: Yaptide", and "simType: SHIELDHIT". A progress bar at the bottom indicates an "Estimated time remaining 00:00:04:41".

On the left side, there is a sidebar with navigation icons. The main content area has a "GENERATE FROM EDITOR" button and a "LOCAL" button. Below these is an "UPLOAD FILES" section with a dashed box and the text "or drag and drop it here." and a "DOWNLOAD ALL" link. The main content area displays the output of a simulation, including a table of results for "geo.dat" and "mat.dat".

The "geo.dat" output shows the following data:

RCC	0	1	2	3	4	5
0.0	0.0	20.0	0.0	0.0	0.0	0.0
RCC	2	5.0	0.0	0.0	-0.5	0.0
0.0	22.0	0.0	0.0	0.0	0.0	0.0
RCC	3	5.5	0.0	-1.5	0.0	0.0
0.0	24.0	0.0	0.0	0.0	0.0	0.0
END						
001		+1				
002		+2				
003		-2	+3	-1		
END						
1	2	3				
1	1000	0				

The "mat.dat" output shows the following data:

RCC	0	1	2	3	4	5
0.0	0.0	20.0	0.0	0.0	0.0	0.0
RCC	2	5.0	0.0	0.0	-0.5	0.0
0.0	22.0	0.0	0.0	0.0	0.0	0.0
RCC	3	5.5	0.0	-1.5	0.0	0.0
0.0	24.0	0.0	0.0	0.0	0.0	0.0
END						
001		+1				
002		+2				
003		-2	+3	-1		
END						
1	2	3				
1	1000	0				

At the bottom of the main content area, there is a blue notification bar that says "Input files generated".

Towards GATE

- Technically: OpenGate Python code should be compiled to WASM (pyodide?)
Core Geant4 code stays the same
- probably a lot of effort to adjust all CMake configuration
- Downloading data files - to be handled efficiently



Final slide

This work was supported by the EuroHPC PL infrastructure funded at the Smart Growth Operational Programme (2014-2020), Measure 4.2 under the grant agreement no. POIR.04.02.00-00-D014/20-00

Projects **Performance of Geant4 in WASM** and **Yaptide** were developed for the purposes of Jakub Niechaj and Szymon Kania's diploma theses.