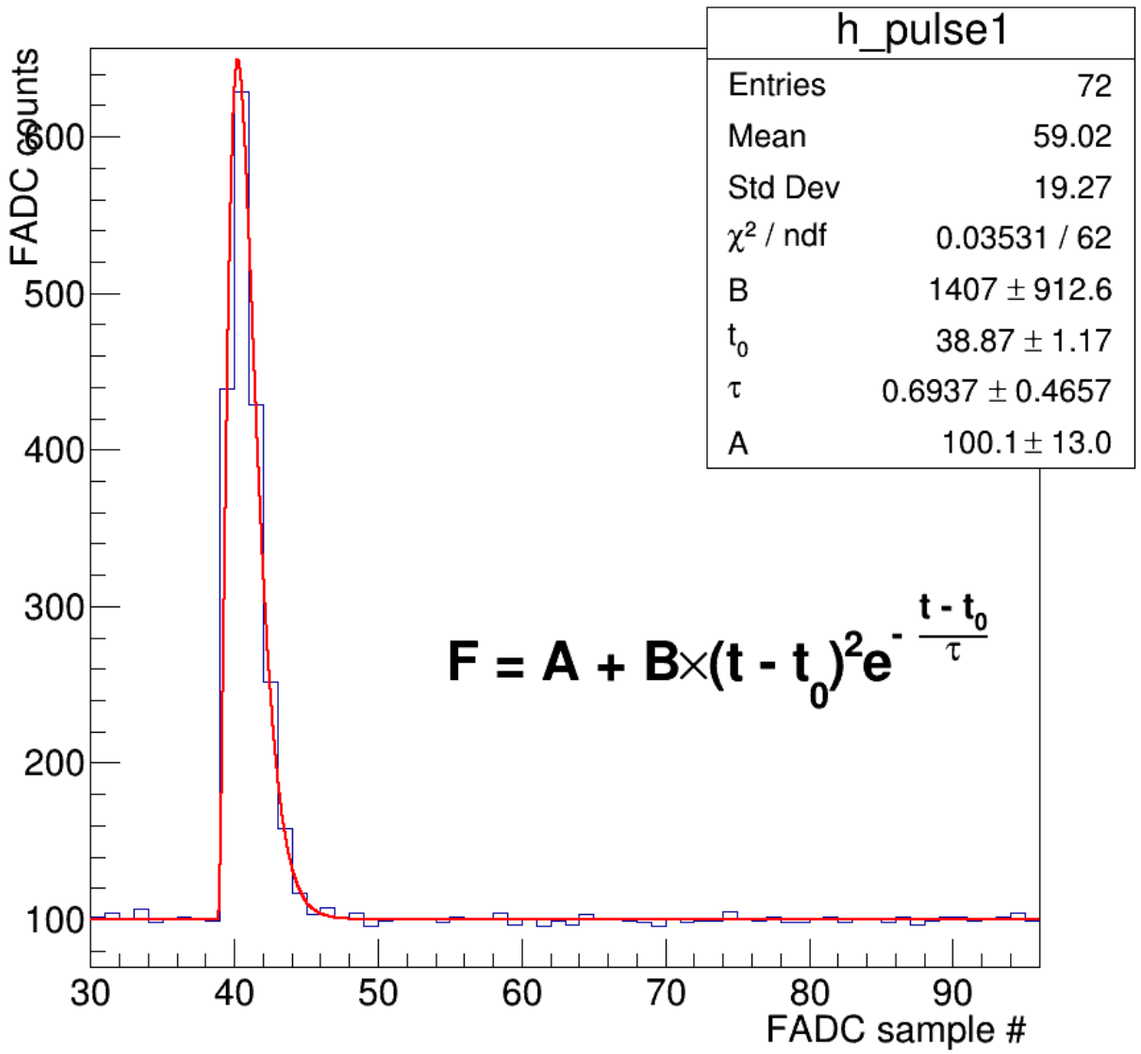
GEMC Present, Future

2.X reaching end of life.

Last “big” changes:

- gdml geometry; cad and “native” geant4 can be mixed and matched.
- FADC Output.
- Magnetic field user requests in progress…
- Background merge possible by October.



In the output:

- crate/slot/channel,
- FADC mode1 or mode7.
- TT from CCDB.

1 strip in CLAS12 calorimeter
pedestals mean and sigma from DB
delay can be tuned

output identical to DAQ
can be used to feed FPGA trigger logic

GEMC 3.0

geant4 improvements:

- Run/Event streamline and API for user actions
- Analysis tools
- Parallel worlds
- **Multithreading**

game changer. Need transition 2.x > 3.0

- better memory management
- code modularization
- code optimization, use modern C++
- cmake support
- better output model
- plugins for input, output, digitization

GEMC 3.0

Multithreading: at the event level

Now thread local:

- Physics list
- Sensitive detector mechanism. Process ID, Energy sharing, Noise, Constants, Background merging, Touchable, Digitization
- Hit definition
- Magnetic field
- Generator and input from files
- Random numbers mechanism

plugin

plugin:
ROOT, HDF5, proMC, JSon

Non thread local:

- merge events info
- output to file

plugin:
ROOT, HDF5, JSon, Excel, EVIO

GEMC 2.X/Geant4 Dependencies

XercesC
for GDML,
XML Parsing

CLHEP for
algebra, units

QT
for graphics

C++11

Geant4

QT
for graphics, XML
parsing

CLHEP for
algebra, units

C++11

CADMESH
For CAD import

GEMC

EVIO
for output

CCDB
For CLAS12
digitization

SCONS
To build it

MYSQL for
CCDB

GEMC 3.X/Geant4 Dependencies

XercesC
for GDML,
XML Parsing

CLHEP for
algebra, units

QT
for graphics

C++11

Geant4

Only additional requirement.
Coming with MLIBRARY.

QT
for graphics, XML
parsing

CLHEP for
algebra, units

C++11

CADMESH
For CAD import

GEMC

Output, digitization plugin will have additional dependencies

Web Interface

EIC Summer Project: Markus Diefenthaler,
Sam Markelon

Develop a web interface to run gemc on a server and/or on the JLAB farm

- select detectors / run conditions
- select/upload generator
- select gemc / geant4 version
- submit jobs on farm
- download results

Contributing to development

Feel free to ask

So let's say that you have an idea for a great feature. It's a good idea to open an issue describing the feature and its implementation and ask the code author's opinion. If they agree, go for it! They might even have some good suggestions for changes or additions to the feature as well.

If it's a bug you found, occasionally it can be ok to just create a **pull request** (PR), as long as it's clearly a bug with a straightforward fix, but it's also not a bad idea to file the bug as an issue first.

Finally, if you want to contribute but are not sure where, you can ask the author if they need help with anything — it could be as simple as helping improve the documentation.

Forking the repo

Ok, so we have a great feature idea (or we found a bug), we opened an issue to check with the author, and they signed off on it. Whoo! Time to get to coding. First thing you do is create a fork, that is a copy of the main repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Forking a repository is a simple two steps process:

1. On GitHub, navigate to the gemc repository.
2. In the top-right corner of the page, click Fork.

You now have a copy of the repo you just forked, available in your GitHub account; its **fork-url-address** can be found on the right menu.

You can create a pull request based on this fork. If you are working on several new features at once, you can create a branch for each feature.

Code Standards

When writing both commits and code, it's important to do so in harmony with a project's existing style. If the project uses camelCase variable naming, this is how you should name your variables as well. If the project has a test suite, you should be writing tests for any changes you make.

Even if you don't agree with some of the author's stylistic decisions, you should adhere to them in your PR. If you have a solid reason why they should be changed, open up an issue and discuss it there. Never ever change an author's existing code style to something you prefer, this is in extremely poor taste.

Create a Pull Request

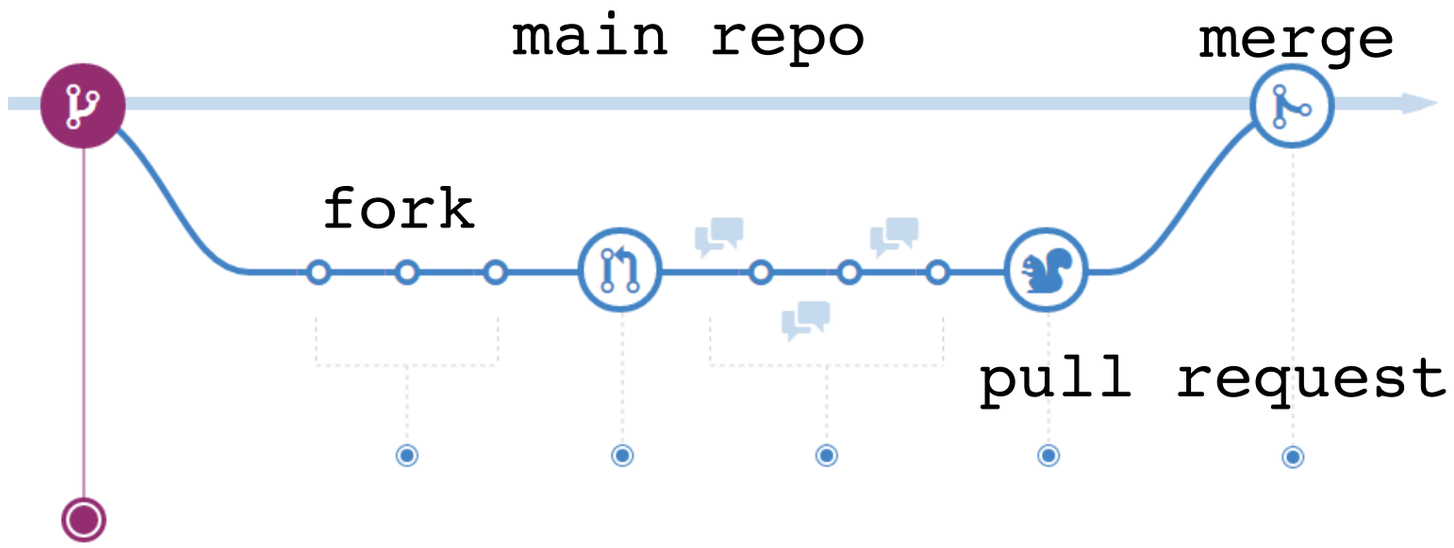
To create the pull request, navigate in github to your fork, and click on the PR button:

You will be presented with a page with a summary of your changes. Once you're ready, go ahead and press the PR button to provide additional informations:

- Make sure you selected the correct branch name ("master" if it's the main fork)
- Make sure the title and description are clear and concise
- If the change is visual, make sure to include a screenshot or gif
- If the PR closes an issue, make sure to put Closes #X at the end of the

description on a newline

1. Discuss an "issue"
2. Fork
3. Modify
4. Pull request



https://github.com/gemc

Solid Specific Requests

ROOT Output:
Currently: evio2root converter.
ROOT will be part of output plugin in 3.0.
Plugin work ready to start ~october (volunteers?)

Save / Reuse random seed
Needs investigation. Thread-local in 3.X.

Full Trajectory History
Can be included in 3.X.
Could be independent project for a geant4 / c++11 expert

Select variables to be stored.
Native in 3.X. Could be done in 2.X.

Self describing output.
Should be native in 3.X. Open to details on the output model, implementation, etc.

CMAKE support
Native in 3.X. Could be done in 2.X. Need help with this.