gemc 2 portfolio

Ingredients to cook a geant4 simulation
gemc 2 portfolio

abstraction of "detector" to table of parameters
gemc 2 portfolio

standardized api for all components

Generator
Solid Volume
Logical Volume
Physical Volume
Mirrors
Materials
Sensitivity
Hit Definition
Maps
Production Cuts

Digitization
Elements, Composite
Magn. Field
Physics
True Info
Multipoles Field
Region
Steps
Bank Definition

GEMC
gemc API status

- Complete description of a detector outside code.
- Detector is a C++ "object".
- Current APIs:
  - Perl
  - Python (not completed)
  - Java (in the works)
  - Wanted: root?

Database: TEXT and MYSQL factories.
Both support "variation"
MYSQL supports "run number"
gemc API status

# Configuration file for ctof

# Detector name and variation
detector_name: ctof
variation: original

# Factory can be MYSQL or TEXT
factory: TEXT

# run ranges and variation will apply to geometry, materials and parameters
rmin: 1
rmax: 10000

Configuration file selects system name, factory, variation, run ranges

api will read conf. file, build detector
gemc API status

Working?

- External database: YES
- API: perl works but perl is old. python or java are modern languages. Both not completed.
- ROOT api wanted.
generators

- Internal: up to 3 particles
  One primary two "beam"

- Lund Format (text)

- SLAC Formats
  (StdHep, IXDR)

GEMC

Easy to add others
generators

Working? yes, but not elegant.

Needed: factory mechanism.

Wanted: general geant4 MPrimaryGenerator plugin that could be loaded by gemc, fairroot, etc.
luminosity

Generating Background

To add background coming from the beam the following quantities must be defined:

1. a time window: the total time of one event
2. the number of beam particles for each event
3. the number of beam bunches

These quantities are defined with the LUMI_EVENT option. For example for clas12 $10^{35}$ luminosity on 5cm LH2 target:

```
<option name="LUMI_EVENT" value="124000, 250*ns, 2*ns" />
<option name="LUMI_P" value="e-, 11*GeV, 0*deg, 0*deg" />
<option name="LUMI_V" value="(0.,0.,-10.)cm" />
<option name="LUMI_SPREAD_V" value="(0.01, 0.01)cm" />
```

Adds 124000 e- in 250 ns time window, grouped in 2 ns bunches. That would produce 125 bunches with 992 particles each bunch. The beam is 100 micron wide and starts 10 cm upstream of the center of the target.

Working?

Yes, with one or 2 beams.

Wanted:
- gemc optimisation
- geant4 multithreading
electronic time window

Purpose: detector response is same as ADC/TDC/FADC signal.

All steps in the same time window, same detector cell form a hit

Working: awesome.
“advanced digitization”:

- Digitization should be AGARA: As Good As Reasonably Achievable

- Calibration Constant, mechanism should not slow down or complicate GEMC. Same DB.
advanced digitization

ADC:
- Attenuation according to exponential law
- Conversion from energy to ADC based on MIP signal
  \( (dE/dx_{\text{MIP}} = 2 \text{ MeV/cm, countsForAMinimumIonizing}=2000) \)

TDC:
- Delay due to light propagation in the paddle (effective velocity)
- Parameterized Time Walk
- Gaussian time spread based on parameters that will be matched to data
  \( (\sigma^2 = \sigma_0^2 + \sigma_1^2 / \sqrt{E}) \)
- EPMT Conversion from time to TDC
  \( (\text{time2tdc}=20\text{ns}^{-1}) \)

Output: both “smeared” and “unsmeared” TDCs

Status:
0 - fully functioning
1 - noADC
2 - noTDC
3 - noADC, noTDC (PMT is dead)
5 - any other reconstruction problem
advanced digitization

DC Efficiency for g12 runs contributed proportionally to their FC Q

dead 50% of the time, correlated!

The right thing to do: Kill dead channels based on individual runs (logistic nightmare)
advanced digitization

-RUN_WEIGHTS="runs.txt"

2 0.1
13 0.6
22 0.2
30 0.1

-N=100,000 generates:

10,043 with constants from run 2
59,901 with constants from run 13
20,034 with constants from run 22
10,022 with constants from run 30

Run number put in header bank
(events are also ordered by run)
advanced digitization

TEST: FTOF paddle

-RUN_WEIGHTS="runs.txt" -N=100000

> Run weights table loaded:
  - run: 2 weight: 0.1 n. events: 10043
  - run: 13 weight: 0.6 n. events: 59901
  - run: 22 weight: 0.2 n. events: 20034
  - run: 30 weight: 0.1 n. events: 10022

DB status “3”

for paddle 11 in run 13 for paddle 13 in run 22 for paddle 15 in run 30

7.8K

3156

(40.4%)
advanced digitization

Working: prototype stage. Awesome

- parameters are the same as used in reconstruction, calibration (same DB)

- can “label” instead of “kill”

- both smeared and unsmeared output

- can always do digitization at a second stage by external software
detector factories

- MYSQL
  (perl, python, java)

- TEXT
  (perl, python, java)

- GDML

- C++ library

- Easy to add others (ROOT?), plugins

Can have EC from a factory, ctof from another.

GEMC
hit processes factories

Drift Chambers
Time of Flights
Cherenkovs
RICHs
Micromegas
Silicon Trackers
FLUX

GEMC

association of c++ algorithm and detector performed at run time
gemc factories

Working?
- MYSQL, TEXT, C++: awesome
- GDML: beta
- Wanted: ROOT

Working?
- Mechanism: yes but can be optimized (string comparison maybe expensive)
- Plugin: NOT CODED YET. Routines still compiled with gemc core (ouch).
output, bank structure

> BST 100, 0

> True Step by Step infos (101, 0)
  - Edep (101, 1)
  - Pid (101, 2)
  - positions (101, 3)

> Dgtz Step by Step infos (102, 0)
  - ADCL (102, 1)
  - ADCR (102, 2)

> True Integrated infos (103, 0)
  - Edep (103, 1)
  - Pid (103, 2)
  - positions (103, 3)

> Dgtz Integrated infos (104, 0)
  - ADCL (104, 1)
  - ADCR (104, 2)

> Voltage as a function of time (105, 0)
  - Identifier (105, 1)
  - Time (105, 2)
  - Voltage (105, 3)

> Trigger Bank (106, 0)
  - Identifier (106, 1)
  - Time (106, 2)
  - Voltage (106, 3)
output, bank structure

> BST 100, 0

> True Step by Step infos (101, 0)
  - Edep (101, 1)
  - Pid (101, 2)
  - positions (101, 3)

> Dgtz Step by Step infos (102, 0)
  - ADCL (102, 1)
  - ADCR (102, 2)

> True Integrated infos (103, 0)
  - Edep (103, 1)
  - Pid (103, 2)
  - positions (103, 3)

> Dgtz Integrated infos (104, 0)
  - ADCL (104, 1)
  - ADCR (104, 2)

> Voltage as a function of time (105, 0)
  - Identifier (105, 1)
  - Time (105, 2)
  - Voltage (105, 3)

> Trigger Bank (106, 0)
  - Identifier (106, 1)
  - Time (106, 2)
  - Voltage (106, 3)

> Header 10, 0

- Event number
- Timestamp
- Event Type
  [...]

> Generated Particles

- pid
- p
- v
- vector<Det. Summary>
  - EC, 22 hits, 3 GeV
  - DC, 29 hits, 0.6 MeV
  - HTCC, 2 hits, 23 nphe

Fast MC Mode
  - ENERGY_CUT
  - “fast MC routines”
output, bank structure

Working?

- awesome. Allows for integrated over electronic time window (pile ups) or step by step output for external digitization.

- Plugin: not coded yet but not crucial.

- Wanted: ROOT output plugin.
Voltage Signal

Rise time: 1ns  
Fall time: 2ns  
Delay: 50ns  
1 MeV = 100 mV  

Total Signal is integral (over electronic time window) of all the step-signals. Signal processing time is small.

Working: YES, but parameters are common for all channels. May need to introduce channel-by-channel pars
Dipole, Quadrupoles, Multipoles

- Working: yes
- Needed: superimposing fields
- Wanted: better more general description (useful for reconstruction as well)
# Optical Properties

- **surface**
- **type**
- **optical properties:**
  - photonEnergy
  - indexOfRefraction
  - reflectivity
  - efficiency
  - specularlobe
  - specularspike
  - backscatter

<table>
<thead>
<tr>
<th>polished</th>
<th>smooth perfectly polished surface</th>
</tr>
</thead>
<tbody>
<tr>
<td>polishedfrontpainted</td>
<td>smooth top-layer (front) paint</td>
</tr>
<tr>
<td>polishedbackpainted</td>
<td>same as 'polished' but with a back-paint</td>
</tr>
<tr>
<td>polishedlumirrorair</td>
<td>mechanically polished surface, with lumirror</td>
</tr>
<tr>
<td>polishedlumirrorglue</td>
<td>mechanically polished surface, with lumirror &amp; meltmount</td>
</tr>
<tr>
<td>polishedteflonair</td>
<td>mechanically polished surface, with teflon</td>
</tr>
<tr>
<td>polisheddioair</td>
<td>mechanically polished surface, with dio paint</td>
</tr>
<tr>
<td>polishedtyvekair</td>
<td>mechanically polished surface, with tyvek</td>
</tr>
<tr>
<td>polishedvm2000air</td>
<td>mechanically polished surface, with esr film</td>
</tr>
<tr>
<td>polishedvm2000glue</td>
<td>mechanically polished surface, with esr film &amp; meltmount</td>
</tr>
<tr>
<td>etchedlumirrorair</td>
<td>chemically etched surface, with lumirror</td>
</tr>
<tr>
<td>etchedlumirrorglue</td>
<td>chemically etched surface, with lumirror &amp; meltmount</td>
</tr>
<tr>
<td>etchedteflonair</td>
<td>chemically etched surface, with teflon</td>
</tr>
<tr>
<td>etcheddioair</td>
<td>chemically etched surface, with dio paint</td>
</tr>
<tr>
<td>etchedtyvekair</td>
<td>chemically etched surface, with tyvek</td>
</tr>
<tr>
<td>etchedvm2000air</td>
<td>chemically etched surface, with esr film</td>
</tr>
<tr>
<td>etchedvm2000glue</td>
<td>chemically etched surface, with esr film &amp; meltmount</td>
</tr>
<tr>
<td>groundlumirrorair</td>
<td>rough-cut surface, with lumirror</td>
</tr>
<tr>
<td>groundlumirrorglue</td>
<td>rough-cut surface, with lumirror &amp; meltmount</td>
</tr>
<tr>
<td>groundteflonair</td>
<td>rough-cut surface, with teflon</td>
</tr>
<tr>
<td>grounddioair</td>
<td>rough-cut surface, with dio paint</td>
</tr>
<tr>
<td>groundtyvekair</td>
<td>rough-cut surface, with tyvek</td>
</tr>
<tr>
<td>groundvm2000air</td>
<td>rough-cut surface, with esr film</td>
</tr>
<tr>
<td>groundvm2000glue</td>
<td>rough-cut surface, with esr film &amp; meltmount</td>
</tr>
</tbody>
</table>
Optical Properties

- surface
- type
- optical properties:
  - photonEnergy
  - indexOfRefrac4on
  - reflectivity
  - efficiency
  - specularlobe
  - specularspike
  - backscatter

- dielectric_metal: dielectric-metal interface
- dielectric_dielectric: dielectric-dielectric interface
- dielectric_LUT: dielectric-Look-Up-Table interface
Optical Properties

- surface
- type
- optical properties:
  - photonEnergy
  - indexOfRefrac4on
  - reflectivity
  - efficiency
  - specularlobe
  - specularspike
  - backscatter

Table of optical photon energies (wavelengths) from 190-650 nm:

```perl
my $penergy = "  1.9074494*eV  1.9372533*eV  1.9680033*eV  1.9997453*eV  2.0325280*eV " .
"  2.0664035*eV  2.1014273*eV  2.1376588*eV  2.1751616*eV  2.2140037*eV " .
"  2.2542584*eV  2.2960039*eV  2.3393247*eV  2.3843117*eV  2.4310630*eV " .
"  2.4796842*eV  2.5302900*eV  2.5830044*eV  2.6379619*eV  2.6953089*eV " .
"  2.7552047*eV  2.8178230*eV  2.8835377*eV  2.9520050*eV  3.0240051*eV " .
"  4.9593684*eV  5.1660088*eV  5.396179*eV  5.6356459*eV  5.9040100*eV " ;
```

Reflectivity of AlMgF2 coated on thermally shaped acrylic sheets, measured by AJRP, 10/01/2012:

```perl
my $reflectivity = "  0.8331038     0.8309071     0.8279127     0.8280742     0.8322623 " .
"  0.837572      0.8481834     0.8660284     0.861336    0.8826233 " .
"  0.8563167     0.8667431     0.85955      0.8722461     0.8728122 " .
"  0.8771635     0.879907     0.879761      0.8831943     0.8894673 " .
"  0.8984234     0.9009531     0.8910166     0.8887382     0.8869093 " .
"  0.8941976     0.8948479     0.887356      0.9026919     0.899685  " .
"  0.901617      0.9090085     0.991694      0.9990897     0.9900493 " .
"  0.9065833     0.9028855     0.895184      0.9009736     0.9006968 " .
"  0.9815145     0.8914838     0.8816829     0.8666895     0.8496298 " .
"  0.9842583 ";
```
Optical Properties

Working?

- awesome.

- geant4 propagation of optical photons not perfect
How to contribute

1. Create an “issue”
2. Fork
3. Modify
4. Pull request

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 whom, 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 if we have a great feature idea (or we found a bug), we opened an issue to check with the author, and they asked for input. 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 step 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 comments 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 an extremely poor taste.

Create a Pull Request

To create the pull request, navigate to 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 information:

- 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 #XX at the end of the description on a newline

source: https://github.com/gemc
Outlook

Projects:

Overlapping Fields
Multichannel ADC/TDC
C++11
G4 Multithreading
Code optimisation
ROOT detector factory
ROOT output plugin

Improvements:

API for detector
Common generator library
Hit process should be plugins
Field superposition