Towards a common software framework
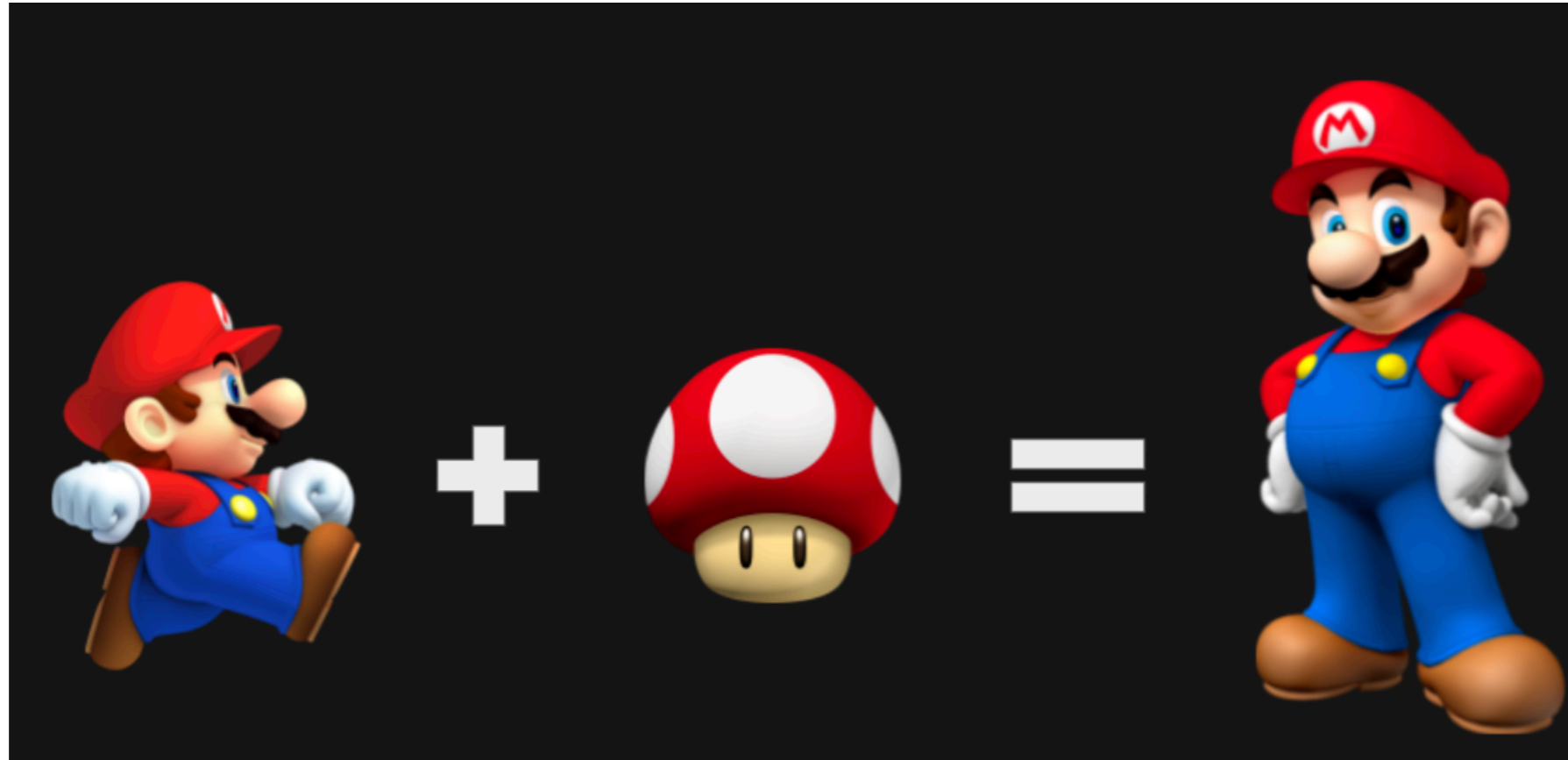

Many people* ... (Harm Schoorlemmer)

* and hopefully more after this meeting

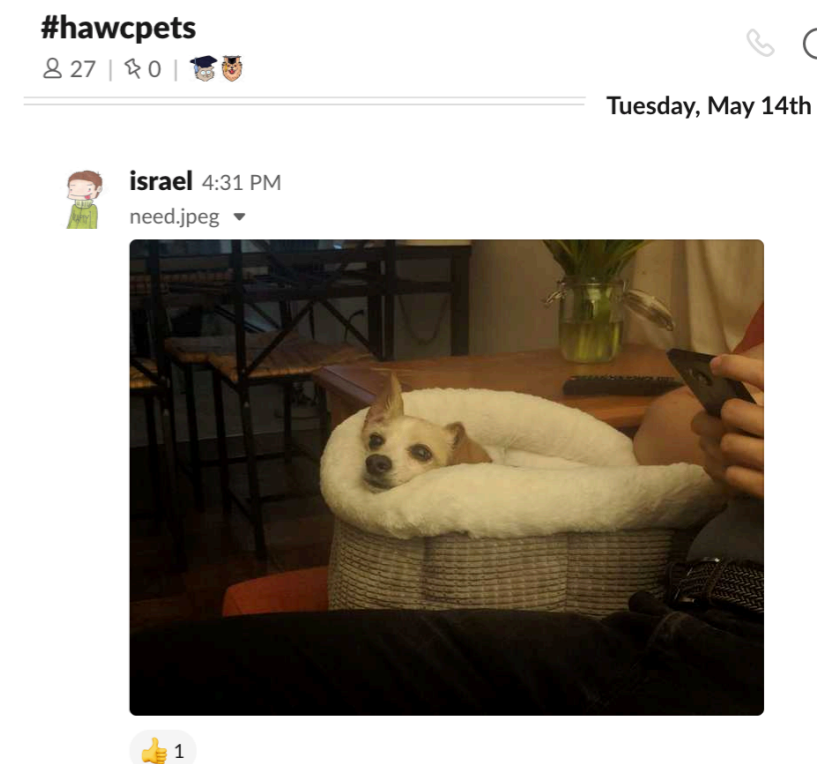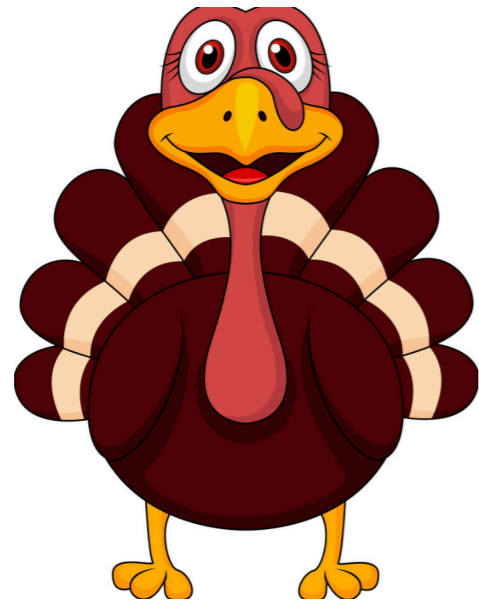# The tools

https://gitlab.com/sgso-alliance

‣ Repository (a lot like github)

‣ Issue tracking

‣ Code review

‣ Controlled access (not public yet)

https://southerngamma.slack.com

‣ Multi-channel chat program

‣ Post results / ask questions

‣ Discuss

#hawcpets
⚇ 27 | ⚡ 0 | 👮🐶

Tuesday, May 14th

israel 4:31 PM
need.jpeg ▾

👍 1

Joining efforts

- ‣ Use HAWC software package as joint starting point
- ‣ Merge independent efforts
- ‣ Develop together and compare different designs

Joint software & simulation development...

‣ Meeting in Heidelberg

- https://indico.in2p3.fr/event/18564/

- Start of joined development based on the HAWC software framework

- Installation difficult

- Defined milestones

Milestones
1. put the SW to run in **LIP**/**MPIK** clusters (currently in progress)
2. Perform a critical review of HAWC SW
   1. To assess if the simulation output structure is adequate
   2. Check the simulation of noise and electronics and trigger
   3. Check interface with G4
3. Adapt HAWCsim
   - Create an abstract layer to select different units (G4 simulations)
   - Implement different units in the SW and develop steering cards for these different units
4. Validate implementation at station level using stand-alone simulations
5. Perform a critical review of HAWCrec
   - Adapt to each unit
   - Build a simple reconstruction chain (including trigger)
   - Adapt/create an event display to check for problems
6. Compute the instrument response functions
7. Create/adapt/use module to compute the sensitivity

Shared (private) repository on gitlab

Shared (private) repository on gitlab

Subgroups and projects   Shared projects   Archived projects

> 📁 **D** **documentation** (Owner)
A place to put documentation related to software and repository

> 📁 **S** **sandboxes** (Owner)
Sandboxes to play around in...

🔖 **E** **examples**

🔖 **H** **hawcsim-gui-example**
Basic example on how to use the HAWCSim graphical interface

🔖 **A** **aerie-install**
Installing externals and aerie.

🔖 **A** **aerie**
SGSO's Aerie fork.

🔖 **H** **HAWC limited code release**

🔖 **A** **aerie documentation**

Slack chat: #simulation channel

Recent code development

Recent code development

Recent code development: three amigos

HAWC
Outrigger

Double-Layer
WCD

HAWC
Main-Array

https://gitlab.com/sgso-alliance/hawcsim-gui-example
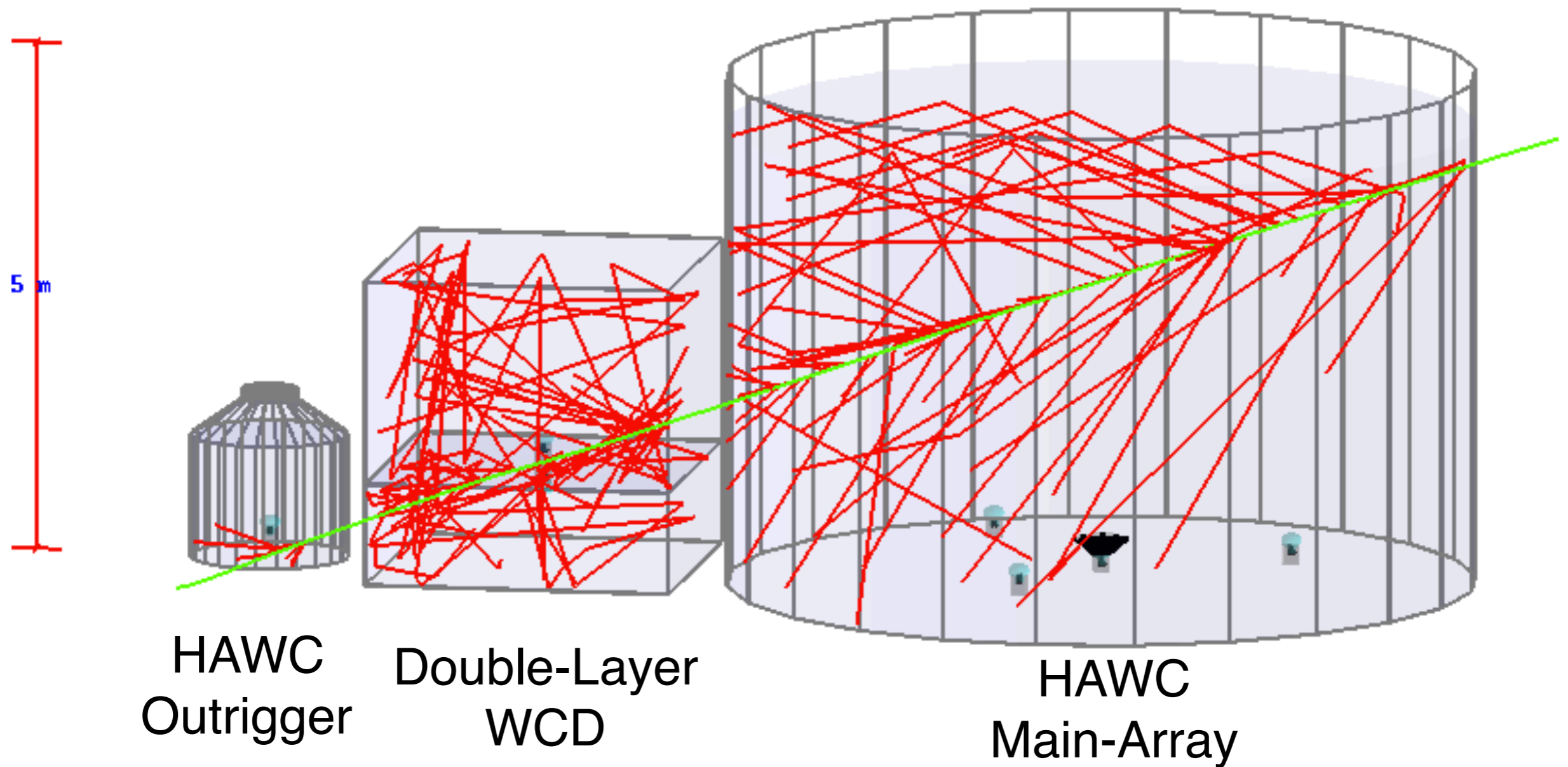
Recent code development: Straw man design array

500 GeV gamma rays
Altitude 5 km

https://gitlab.com/sgso-alliance/examples
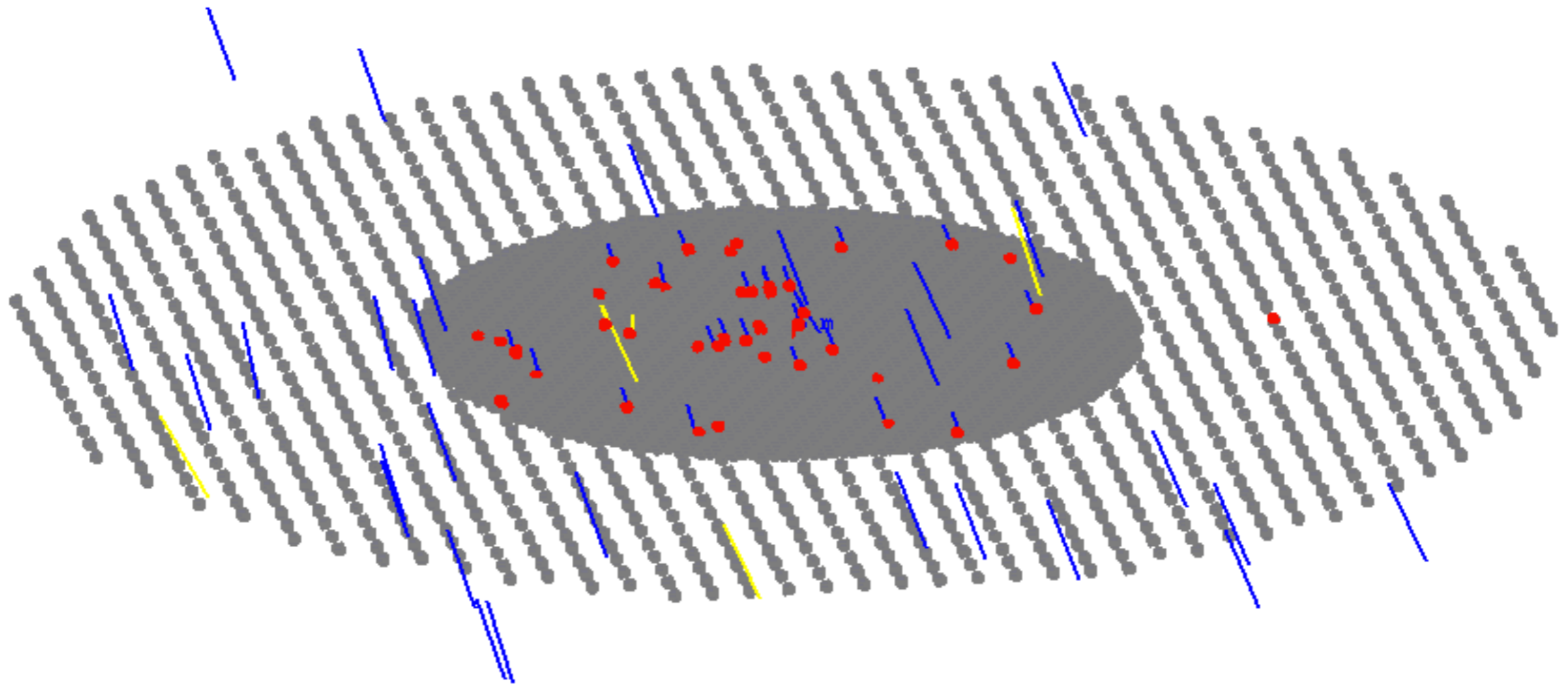
Recent code development: Straw man design array

500 GeV
showers are
not compact! *

Need for
a big array!

200m

150m
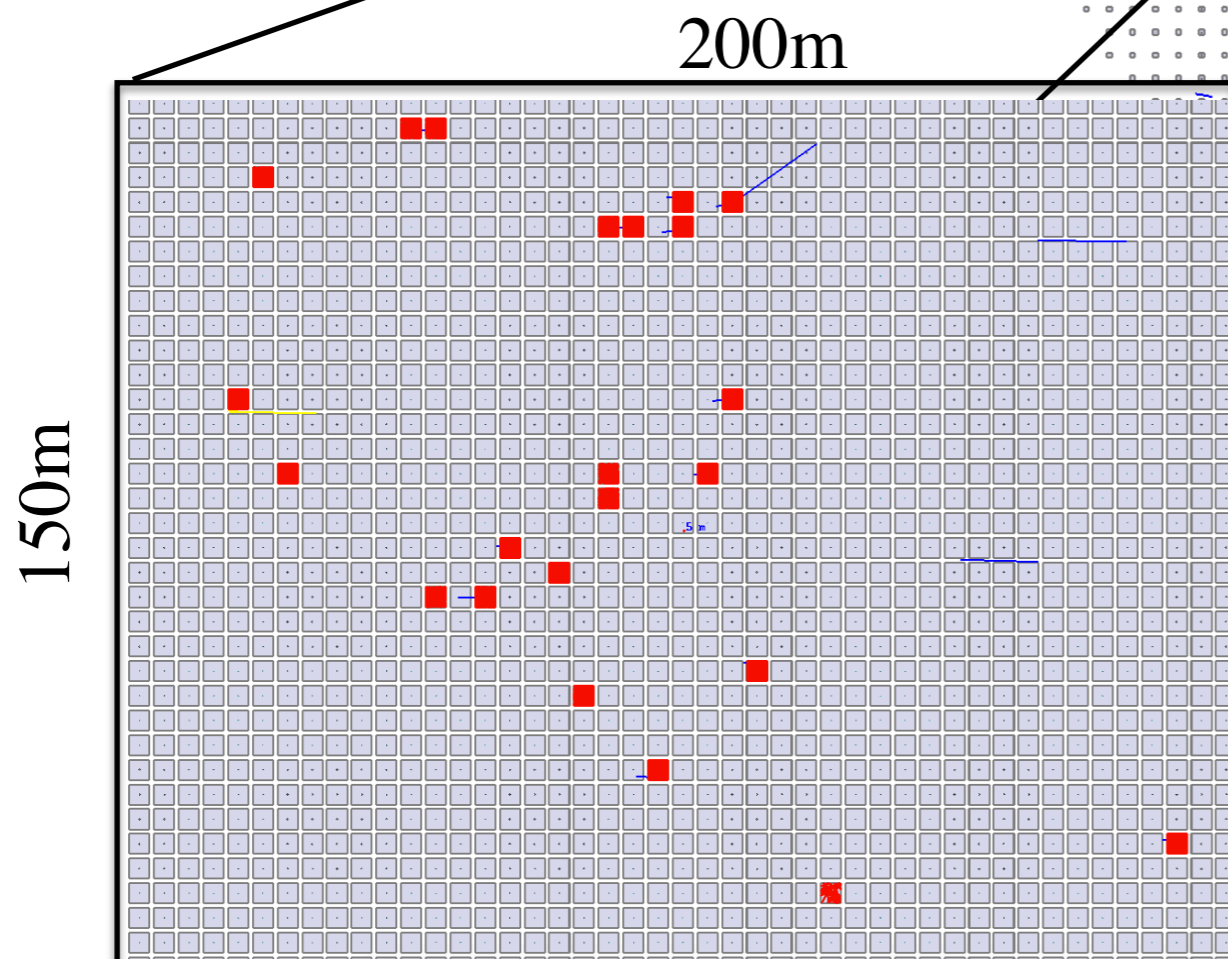
Accepted EPCJ,
HS, J. Hinton, R. López-Coto
https://arxiv.org/abs/1905.06816

Where are we at …

- Building upon HAWC framework
- Corsika -> GEANT 4 -> Output
- Repository / organization
- Installation procedure

Where do we go next…

- Add more designs
- Share CORSIKA production
- Trigger and noise modeling
- Reconstruction Chain
- Sensitivity estimates
- Make repository public, after HAWC will make theirs public

3ML
The Multi-Mission Maximum Likelihood framework

for high-level data analysis

Henrike Fleischhack
May 21st 2019

threeML/astromodels

- `threeML` provides a framework for multi-instrument likelihood analysis.

- Independent of format for data/instrument response files (data access encapsulated in 'plugins').

- Lots of functionality e.g. plotting, source injection, …

- `astromodels` enables detailed spectral/spatial modeling of astrophysical sources.

- See https://github.com/threeML/ and https://arxiv.org/abs/1507.08343

- Actively being used/developed for HAWC analyses, X-ray, …

- Plugins available also for Fermi-LAT, VERITAS, general spectra, …

- Still growing with new plugins, bayesian samplers etc. being added.

2

3ML is different

3ML architecture
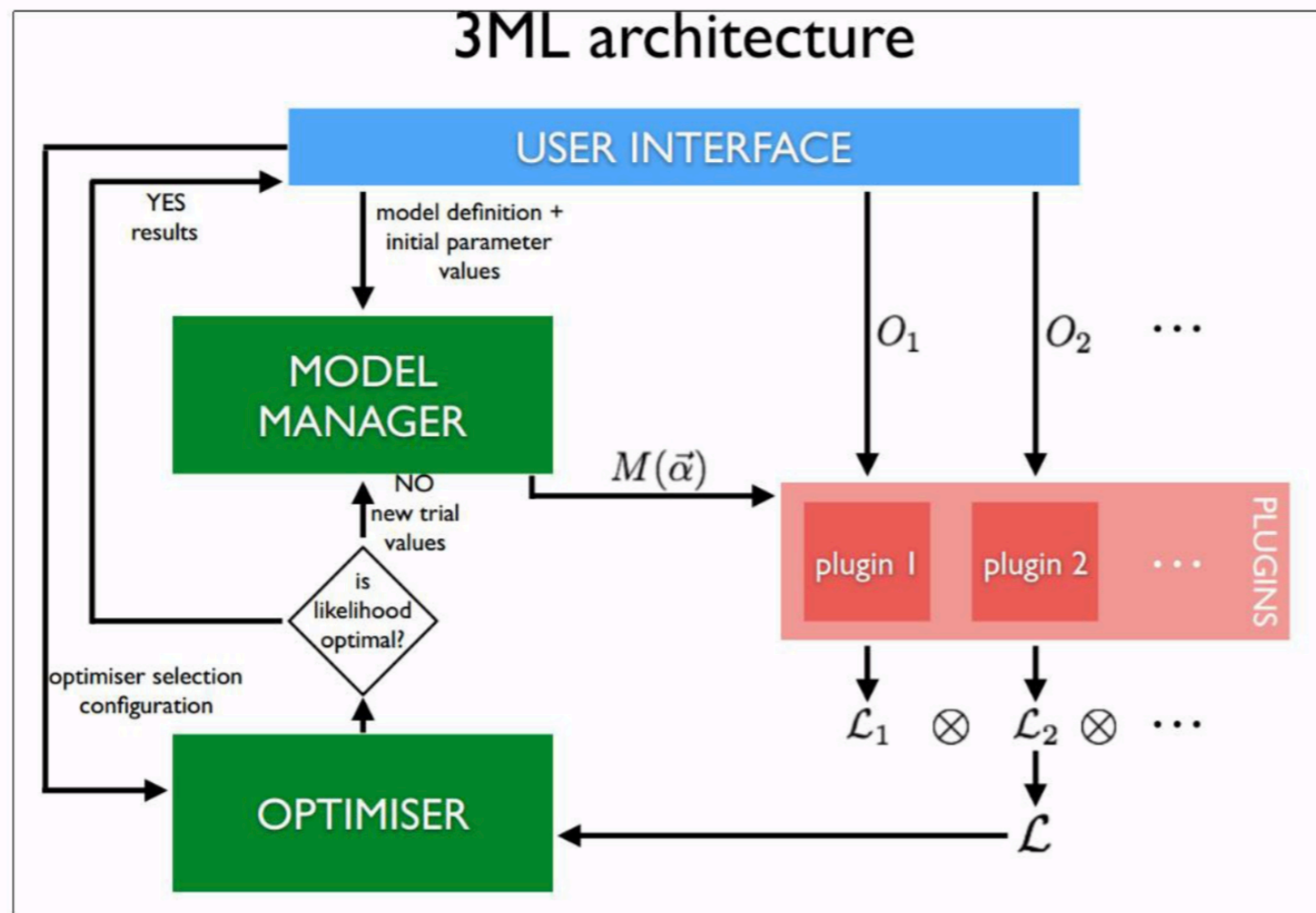
- short pieces of software connecting the framework with the instrument-specific software
  - receive in input the model, give in output the likelihood value
- no constraints on:
  - messenger, data formats, response specification, likelihood implementation, background estimation and handling, language (C++, fortran...)
- existing and new solutions (ST, sherpa, xspec, gammapy, isis) can be used as plugins

Existing plugins

- HAWC: both point-like sources and extended sources
- Fermi/LAT:

  NEW
  - Fermipy: point-like sources, extended source in the working
  - Unbinned analysis: for GRB-type analysis of point sources

NEW
- VERITAS: prototype for point sources ready
- Swift, Chandra, XMM, Fermi/GBM, Konus... (all spectra you can read in XSpec): ready (only point sources)

NEW
- Optical with ~2000 filters (thanks to the Spanish Virtual Observatory's Filter Profile service)

NEW
- SED / "xy" data
- Still need a volunteer for Radio data

HAWC source fitting: forward folding

- We have events and background maps for several analysis bins (nHit or energy).

- We have a detector response which gives the PSF and the expected gamma-ray counts expectation for a point source at a given position and fixed spectrum

- HAL:

  - computes the expectation for a model (set of sources with given morphologies and spectra):

    - re-weights to arbitrary spectra

    - convolves source morphologies with PSF

  - computes the likelihood of the data given the model+background, using Poisson statistics.

- 3ML:

  - provides the model to HAL, runs the likelihood maximization
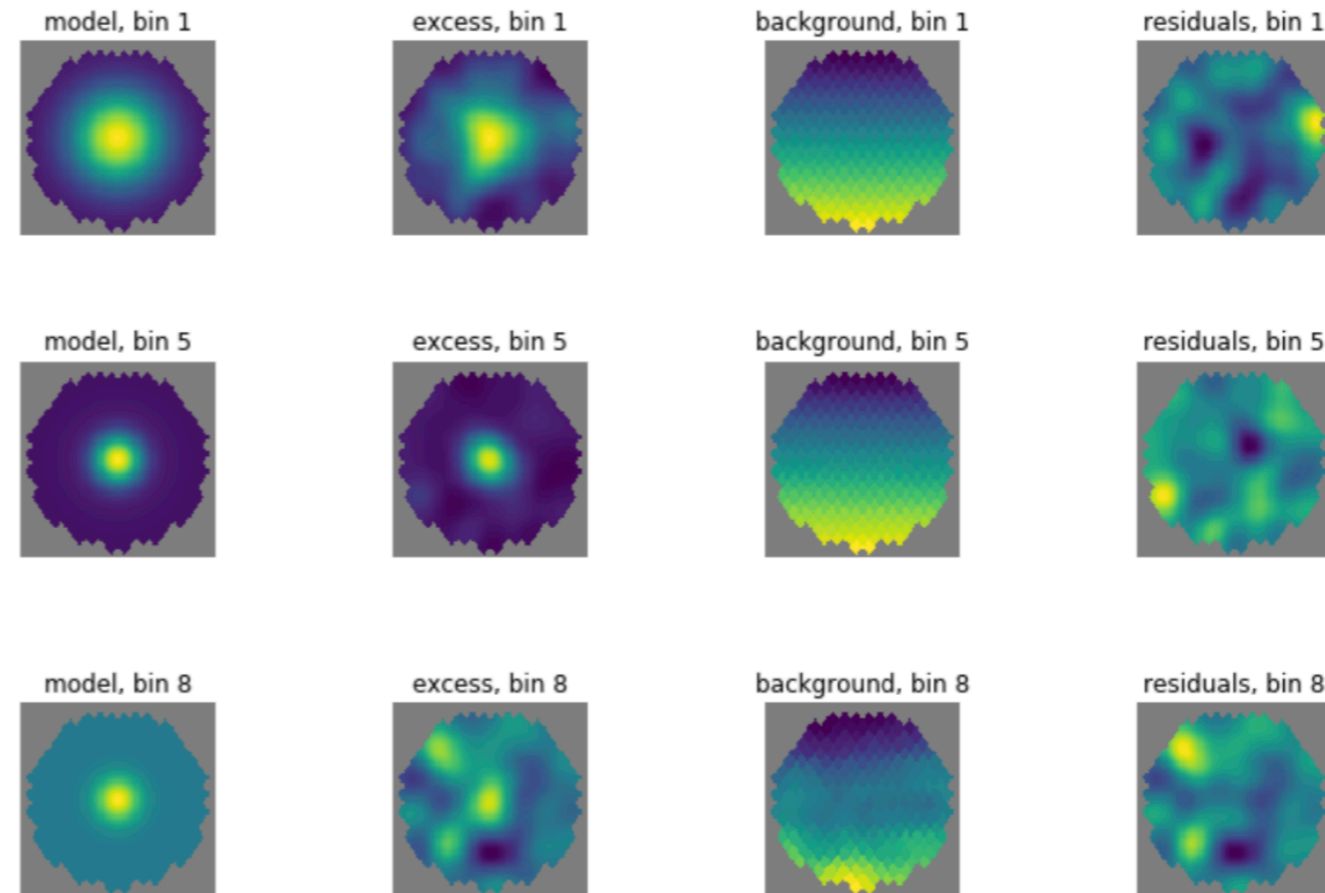
this slide by C. Riviere

Proposal for SGSO

- Propose to use threeML for high-level analysis (spectra, morphology, sensitivity).

- Can use the HAL (HAWC Accelerated Likelihood) plugin for now (https://github.com/threeML/hawc_hal) and work on a dedicated plugin later.

- Would like to implement some mock instrument response files using our design sensitivity while we wait for a full simulation/analysis chain.

- No sensitivity calculation yet in threeML but can be implemented easily on top of existing infrastructure for source injection etc.

Collaborators welcome! Would like to hear of any other approaches to high-level analysis!

Join us!!!!!

3ML
The Multi-Mission Maximum Likelihood framework

for high-level data analysis

Henrike Fleischhack
May 21st 2019

3ML
The Multi-Mission Maximum Likelihood framework

for high-level data analysis

Henrike Fleischhack
May 21st 2019

Sensitivity estimates... Higher level public tools

- Would like to make quantitative statements about SGSO's sensitivity to different kinds of sources/physics questions.
- Need to know how detector performance affects sensitivity.
- For that, need high-level analysis tools.
- I propose to use `threeML`/`astromodels` with the `HAL` plugin.
- Should be developed in parallel to detector design, reconstruction algorithm etc.
- Any other plans for high-level analysis yet?

General remarks

‣ Update your code (installation)

‣ Use slack-channel to ask for help

‣ We need more people to make better use of merge procedure

‣ Many things to work on, see milestone list.

Running the gui example

Code development: merging

Recent code development

```cpp
class Tank
{
public:
  Tank();

  virtual ~Tank() {};
  //Abstract function that needs to be implemented
  // by derived classes
  virtual G4PVPlacement *AddTankToLogicWorld(G4LogicalVolume* LogicWorld, d

  TankType GetTankType() {
    return type_;
  }
  TankType type_;
};
```

```cpp
bool TankArray::LoadTanks(G4LogicalVolume* logicWorld)
{

  DetectorService& detSrvc = GetService<DetectorService>("det");

  DatBase &db (DatBase::GetDB() );

  double sGPSfrac;
  double sGPSdbl = modf(db.GetVal("GPSSeconds"),&sGPSfrac);
  double nsGPSdbl = floor(sGPSfrac*pow(10.,9.));
  TimeStamp ctime( static_cast<int>(sGPSdbl),
                   static_cast<int>(nsGPSdbl) );
  det::Detector det = detSrvc.GetDetector(ctime);

  //all different kind of tanks
  Tank
  *DLWCD = new TankDLWCD(),
  *NormalTank = new TankHAWC300(),
  *Outrigger = new TankHAWCOR(),
//  *DLWCD = new TankDLWCD(),
  *Tank = NULL;

  det::Detector::TankIterator tnkItr = det.TanksBegin();
  Vector arrayCenter = det.GetArrayCenter();

  for (;tnkItr != det.TanksEnd();++tnkItr) {
    //select tank type
    int tankType = tnkItr -> GetTankType();
    if (tankType == DLTank) {
      Tank = DLWCD;
    }
    else if (tankType == OutriggerTank) {
      Tank = Outrigger;
    }
    else {
      Tank = NormalTank;
    }
    //add tank to logical world
    Tank->AddTankToLogicWorld(logicWorld,*tnkItr,arrayCenter,PMTVolumes,logicPMTs);
  }

  return true;
}
```