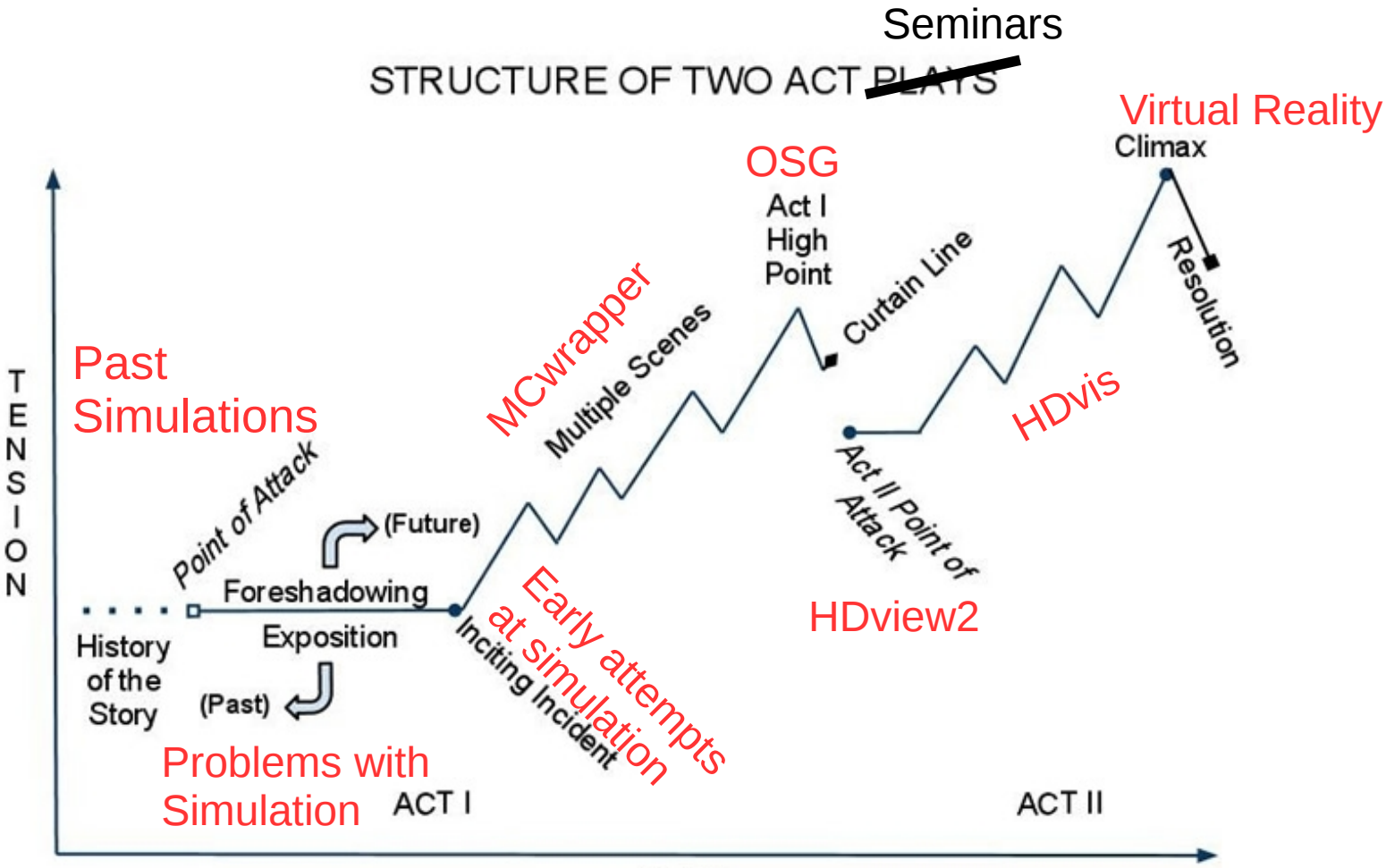
Simulation and Visualization in Hall-D (a seminar in two acts)

Thomas Britton

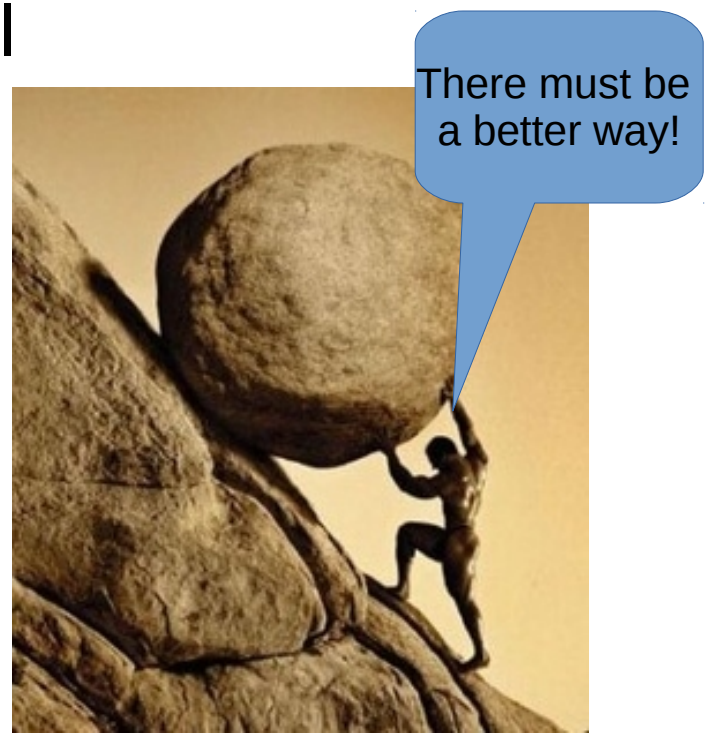
Overview



Simulation **Visualization**

Past Trials

- MC simulations are integral to analyses of all kinds, there is no way around it
- Previously in GlueX each step of simulation required careful configuration and running
  - Duplication of certain parameters
  - Many parameters derived from data from multiple different locations
  - “Fraught” system



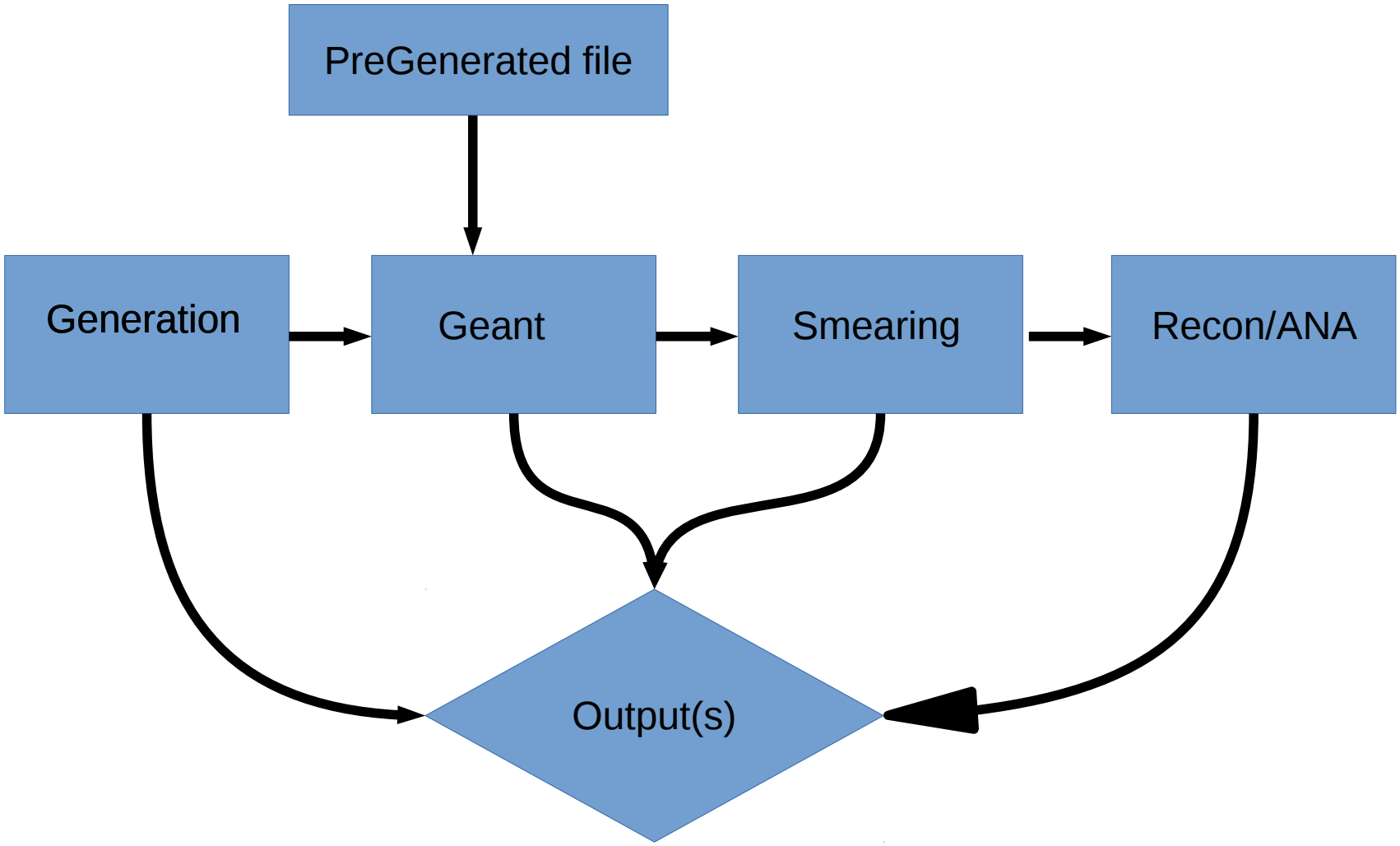
Enter MCwrapper

- Spawned from personally attempting to generate MC
  - Example scripts and configuration files passed around were often incomplete and not well understood by the user
  - Started taking notes on the process and laziness took over: “I'll make the computer do all of the tedious work”
  - A combination of python and shell scripting means NO COMPILED CODE

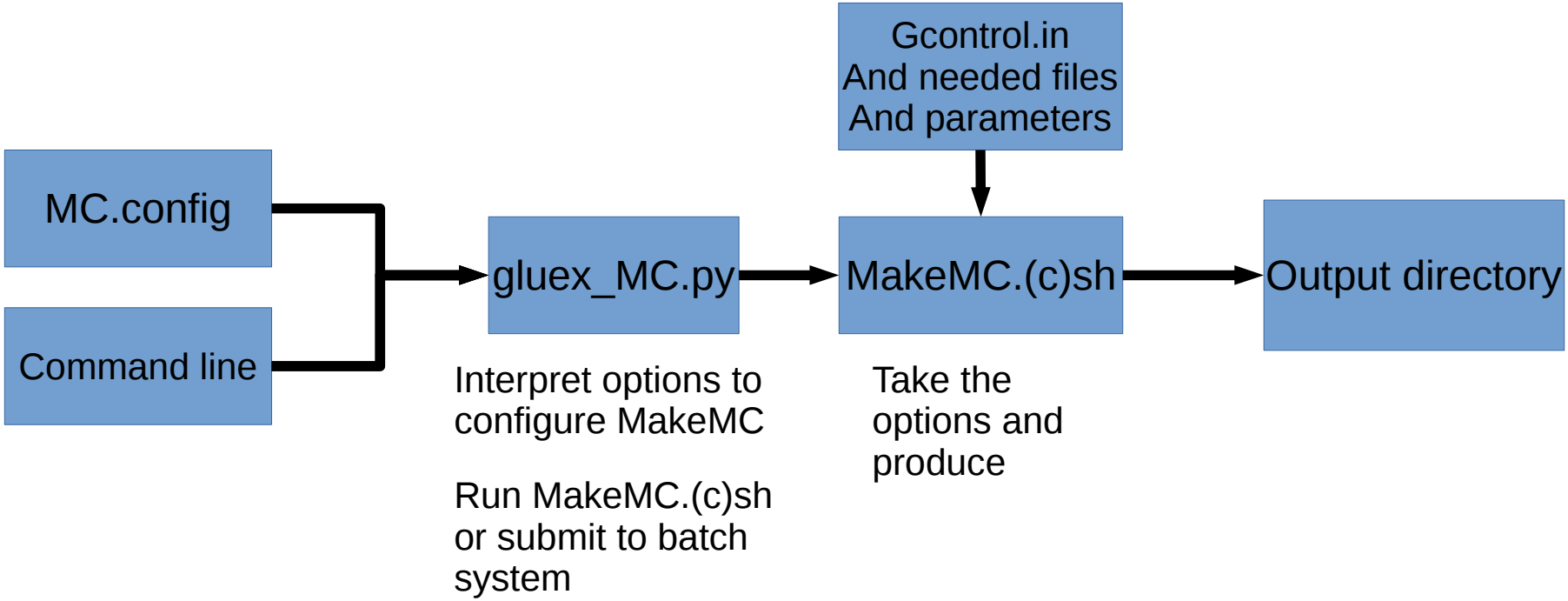
Philosophy

- MCwrapper seeks to be a one-stop-shop for simulation for glue-x/hall-d. As such it needs to be able to:
  - Complete the production chain; from generation through early analysis
  - Run both geant3 and geant4 easily
  - Provide basic standards of simulation
  - Be customizable for individual studies
  - Utilize various batch systems to scale
  - Provide support for new users

MC Production Line



MCwrapper Flow



Integration

- Utilizes the Run Conditions DataBase (RCDB) and the supplied run number to fill out various parameters as desired such as:
  - Radiator thickness
  - Coherent Peak Position
  - Electron beam energy
- This takes the leg work out of grabbing the right values for a given run number and removes a common source of human error: typos

Configuration file and command line

```
#THESE TWO ARE OPTIONAL IF THE STANDARD RUNNING DOESN'T SUIT YOUR NEEDS
#CUSTOM_MAKEMC=use-this-script-instead
#CUSTOM_GCONTROL=use-this-Gcontrol-instead
#=====================================================================

#VARIATION=mc calibtime=timegoes here #set your jana calib context here with or without calibtime Default is variation=mc

#RUNNING_DIRECTORY=/run/in/this/directory #where the code should run.  This is defaulted to ./

#SQLITEPATH=/your/sqlite/path  #if you use SQLITE and it is not part of the environment file that gets sourced

#TAG=my-custom-prefix-tag

DATA_OUTPUT_BASE_DIR=OUTPUT-LOCATION#your desired output location

NCORES=4      # Number of CPU threads to use or nodes:node-id:ppn or nodes:ppn depending on your system

GENERATOR=generator-to-use #or you may specifile file:/.../file-to-use
GENERATOR_CONFIG=config file for generator

#common parameters for generators
#eBEAM_ENERGY=12 #either use =rcdb or do not set to pull the run number from the rcdb
#COHERENT_PEAK=9 #either use =rcdb or do not set to pull the run number from the rcdb
#GEN_MIN_ENERGY=4
#GEN_MAX_ENERGY=12

GEANT_VERSION=3
BKG=None #[None, BeamPhotons, DEFAULT, custom e.g bg.hddm:1.8]

#optional additional plugins that will be run along side danarest and hd_root.  This should be a comma separated list (e.g. plugin1,plugin2)
#CUSTOM_PLUGINS= #or file:/.../file-to-use which is a configuration file for jana/hd_root
#=================================================================================
#EVERYTHING BELOW FOR BATCH ONLY

#VERBOSE=True

BATCH_SYSTEM=swif #can be swif or qsub  adding :[name] will pass -q [name] into PBS.

#environment file location
ENVIRONMENT_FILE=your-environment-files     #change this to your own environment file

WORKFLOW_NAME=WORKFLOW-NAME #SWIF WORKFLOW NAME
PROJECT = gluex          # http://scicomp.jlab.org/scicomp/#/projects
TRACK= simulation          # https://scicomp.jlab.org/docs/batch_job_tracks

# RESOURCES for swif jobs
DISK=5GB            # Max Disk usage
RAM=5GB            # Max RAM usage
TIMELIMIT=300minutes      # Max walltime.  This may be of the form xx:xx:xx depending on your system
```

- All parameters found in one location
  - Hide the stuff that should not be changed or derived values
- Collects parameters once to avoid parameter misalignment
- Configuration file for parameters that are seldom changed
  - e.g. generator
- Command line for parameters changed more frequently
  - Run number

Command line

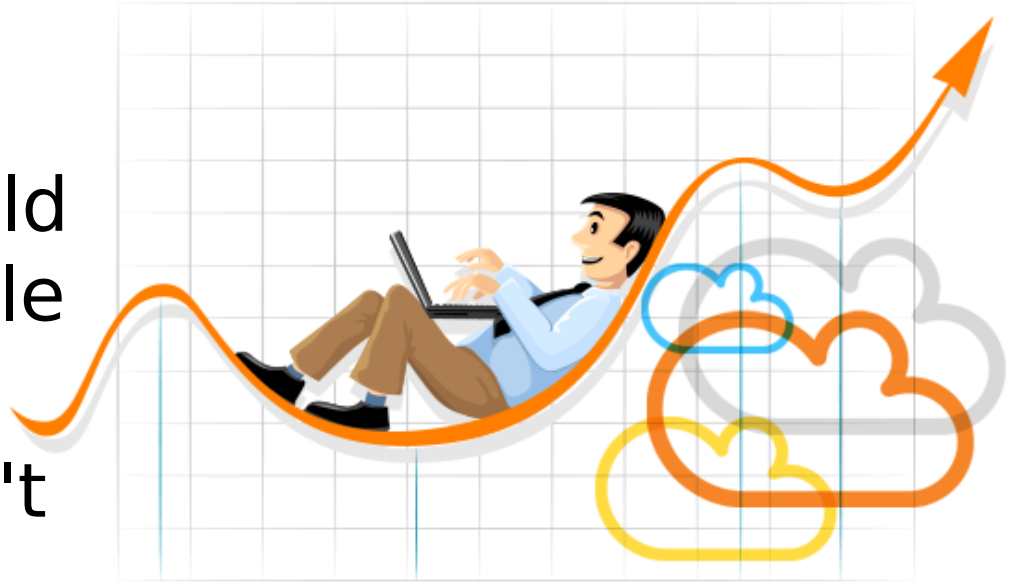
- Usage: **gluex_MC.py config_file [Run_Number or Range] [num_events] [all other options]**
- where [all other options] are:

variation=%s where %s is a valid jana_calib_context variation string (default is "mc")
per_file=%i where %i is the number of events you want per file/job (default is 10000)
numthreads=%i sets the number of threads to use to %i. Note that this will overwrite the NCORES set in MC.config
generate=[0/1] where 0 means that the generation step and any subsequent step will not run (default is 1)
geant=[0/1] where 0 means that the geant step and any subsequent step will not run (default is 1)
mcsmear=[0/1] where 0 means that the mcsmear step and any subsequent step will not run (default is 1)
recon=[0/1] where 0 means that the reconstruction step will not run (default is 1)
cleangenerate=[0/1] where 0 means that the generation step will not be cleaned up after use (default is 1)
cleangeant=[0/1] where 0 means that the geant step will not be cleaned up after use (default is 1)
cleanmcsmear=[0/1] where 0 means that the mcsmear step will not be cleaned up after use (default is 1)
cleanrecon=[0/1] where 0 means that the reconstruction step will not run (default is 1)
batch=[0/1/2] where 1/2 means that it will submit to the specified batch system (default is 0)
logdir=[path] will direct the .out and .err files to the specified path in qsub

Example: gluex_MC.py MC.config 11366 1000000 cleanmcsmear=0 batch=1
Or gluex_MC.py MC.config 11366-11555 1000000 cleanmcsmear=0 batch=1

Useful but...

- MCwrapper could let people, unfamiliar with the halld/glueX software stack to produce MC that would be directly comparable to others
- This, however, doesn't scale
  - Can't run a billion events



Batch jobs

- To handle large volume jobs MCwrapper is designed to run on a variety of batch systems
  - At Jlab It will create a swif workflow and add the necessary jobs to it or, if the workflow exists, will create enough jobs and submit them all
  - In the case of swif it will prompt you with the command to call to run the workflow or if chosen will run the workflow for you
- No more battling with a submission script. MCwrapper will write one for you
  - Swif
  - Qsub
  - Condor
  - OSG
- **OSG SUPPORT!**

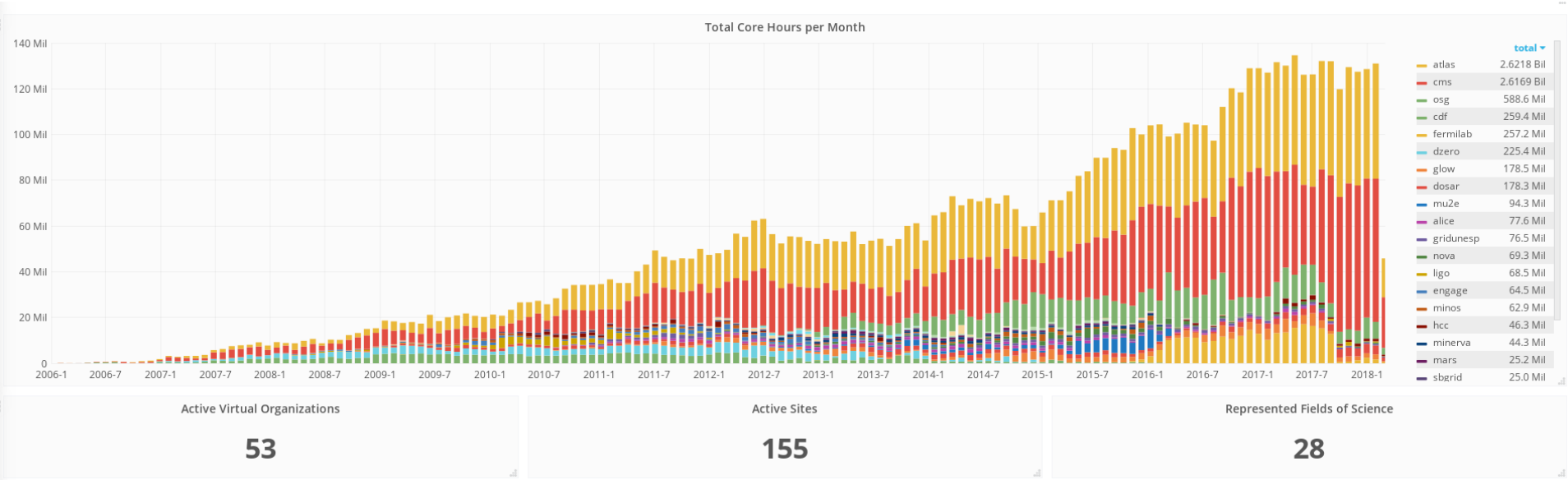
What is the OSG?

- The Open Science Grid:
  - "The OSG provides common service and support for resource providers and scientific institutions using a distributed fabric of high throughput computational services. The OSG does not own resources but provides software and services to users and resource providers alike to enable the opportunistic usage and sharing of resources. The OSG is jointly funded by the Department of Energy and the National Science Foundation. "
- Basically it is a distributed grid of computers where you can donate computational resources and use grid resources
- Used by CMS and ATLAS and many many more
  - Jlab has a submit node
  - The layer is transparent to the MCwrapper user

Open Science Grid

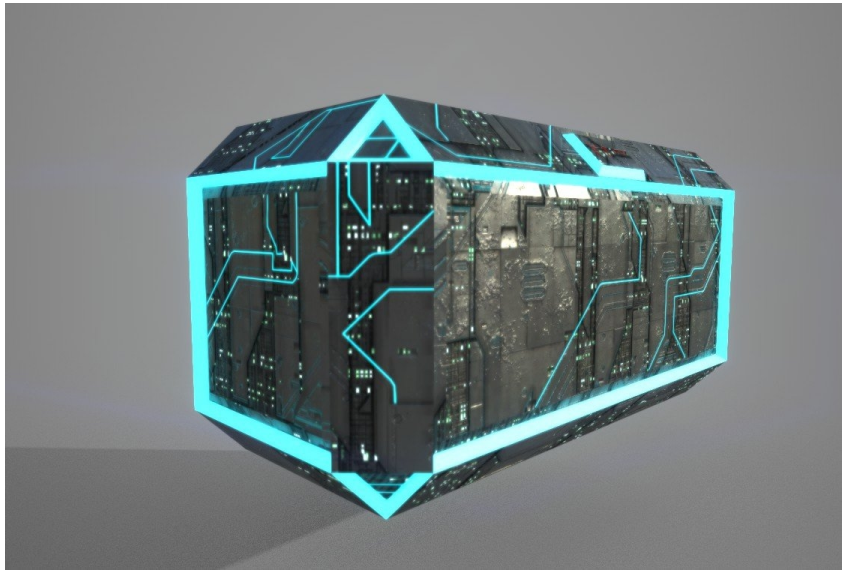
What does this mean?

- No more battling for resources here at jlab and ease of use means it is easier to put people in the environment (no additional training or how-to's)
- Simply have a certificate, be on the submit node, and tell MCwrapper you want to use the osg...done



Future Work

- Generalization of some components
- Wrap to the point of providing an entirely new interface to the simulation chain
- Take over the world
  - Or at least the creation of MC in GlueX by integrating work done on containers



History of visualization

- Hdview(2) was never intended to be the "final" event display for GlueX
  - It uses a root based interface and a series of 2D projections
- There was a past debate about using a 2D vs 3D display
- There was talk of using Hall-B display
  - A student even worked on it...
- I was tasked in ~Feb 2017 with creating a new event viewer to replace hdview2
- Quickly Root's “Eve” event viewer rises to forefront for consideration

Why Eve?

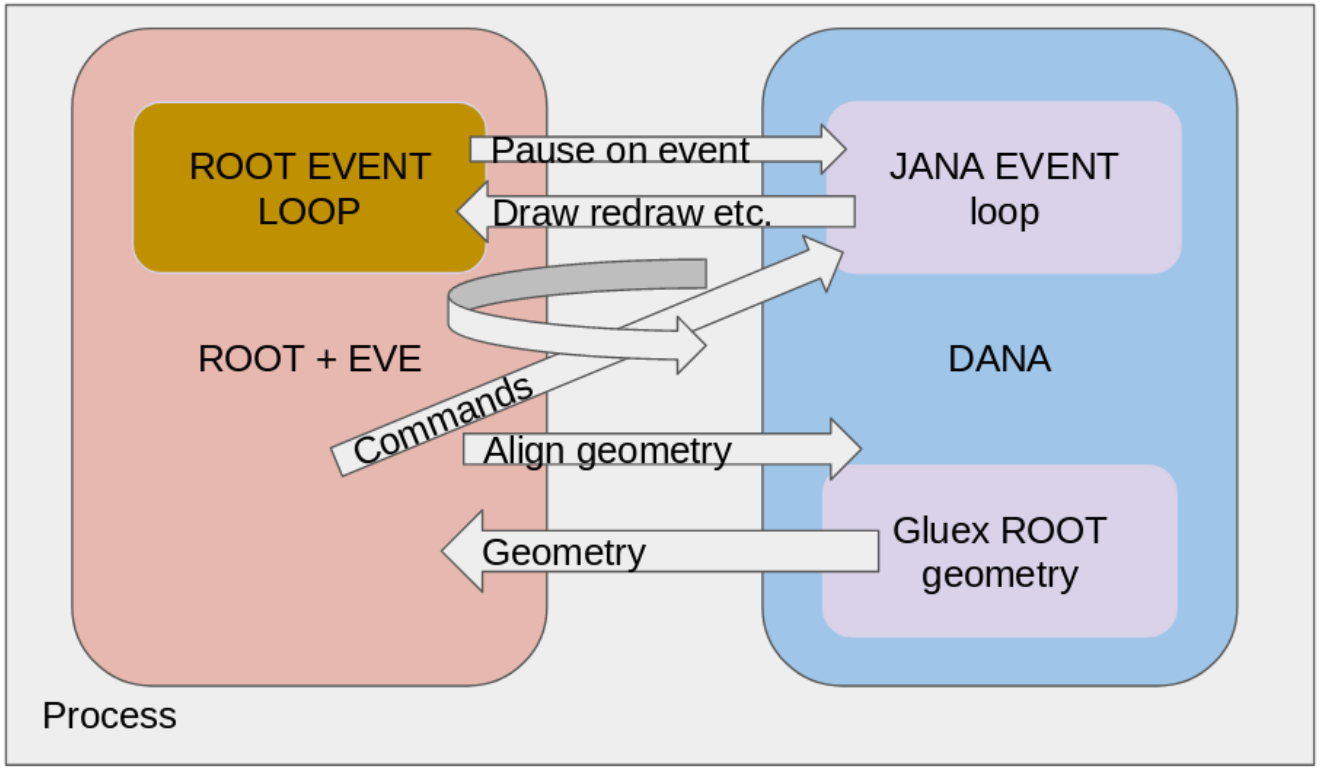
- Eve is a root based event viewer that promises to be a framework for viewing detectors and associated hits
- Hdview was already a root based viewer so it looked like not a lot of work needed to be done
- Root based geometry had been around in Hall-D for many years
- Early prototyping work had been done by Dmitry Romanov
- After consideration and discussion ground was broken March 16th 2016 on a new event viewer
- The plan:
  - Replace hdview2 and run root's eve
  - Use jana to transform data objects from "raw" data into root objects and add them to the canvas/scene

Eve: Epilogue

- Started with eve for event viewing
  - It's terrible and I'd rather not talk about it….
  - Primitives seemed built specifically for the collaboration that created them (boxes and cylinders and jets)
  - Had tutorials that did not function
  - Very little documentation
  - No apparent development
  - Non-functioning alphas
  - Global variables (why?!)
  - Frequent crashes (see above and below)
  - Complex threading (see next slide)
- Ended eve ~ Jun 13th

Thread structure of Eve

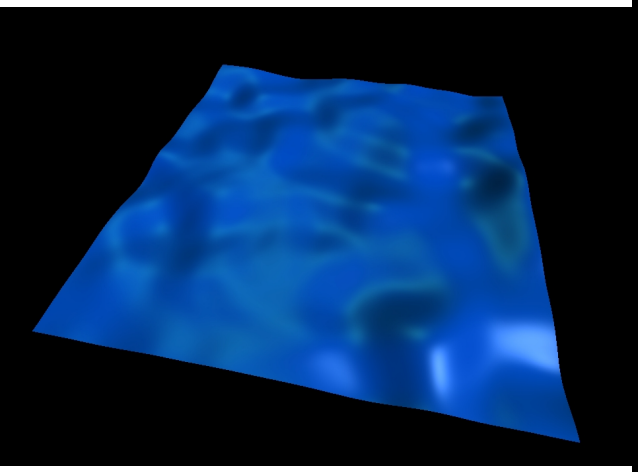
Before

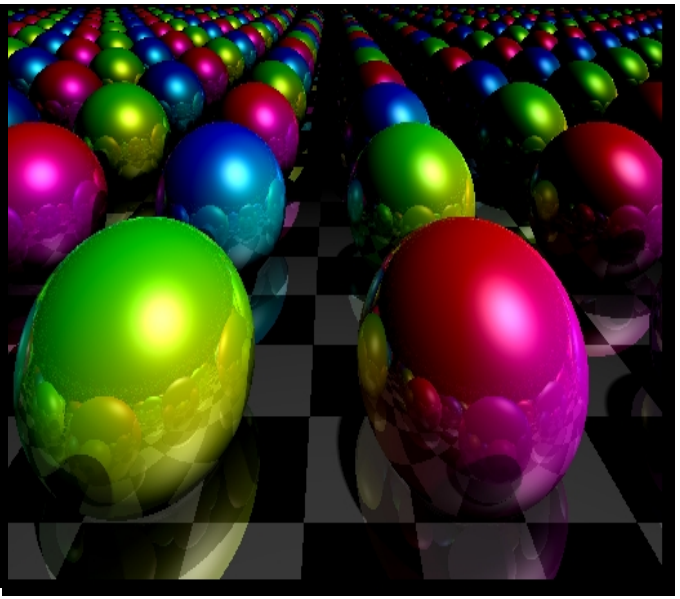


*Thanks to Dmitry Romanov for the diagram

Introducing threejs

- Threejs is a javascript library allowing for the creation of GPU-accelerated 3D animations without browser plugins (thanks WebGL!).
- First released to github in April of 2010





- What took about 2-4 weeks to visualize with root/eve was accomplished in a matter of days

Pros and Cons of threejs

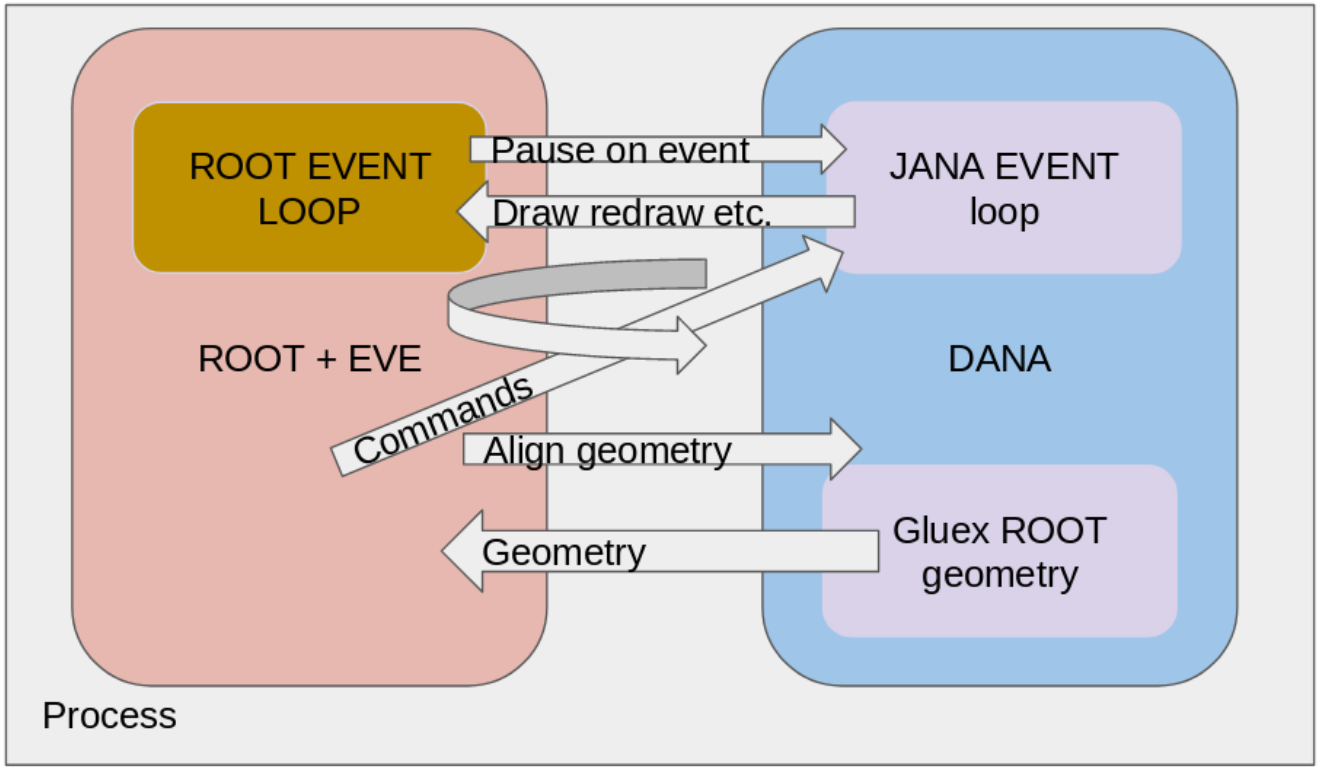
- Pros:
  - Simple primitives
  - Readily available documentation
  - Active development
  - Functioning tutorials
  - Runs in the browser
  - “Free” Features (including VR!)
  - Much simpler threading scheme
  - Also it isn’t eve

Eve Three.js

- Cons:
  - Geometry had to be developed/translated to JS
  - Event information needs conversion to JSON

Thread structure of Eve

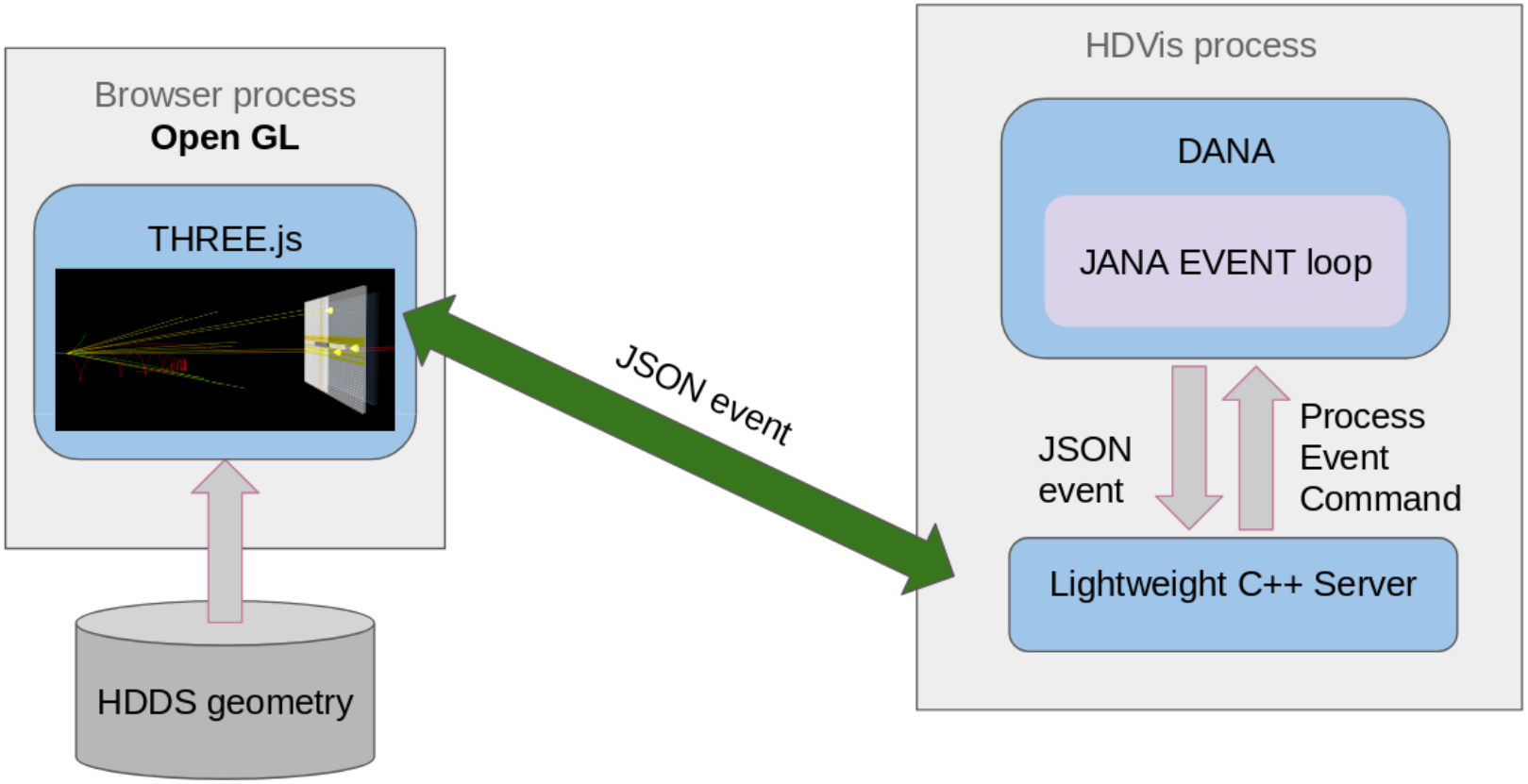
Before



*Thanks to Dmitry Romanov for the diagram

Thread structure of threejs

Now



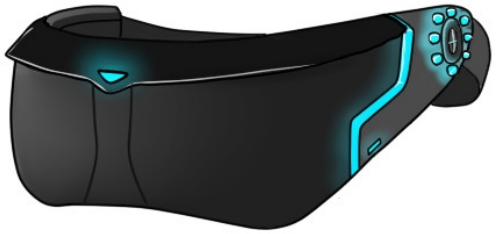
*Thanks to Dmitry Romanov for the diagram

Main Features

- Nothing but a jana program to install
  - Assuming you already have a web browser with WebGL
- Full 3D display with both perspective and orthographic cameras
- Runs in the Browser!
- Animations: Timing information is incorporated into all objects allowing for accurate animations of the event. This allows for some quick diagnosis of timing windows and the like
- Dynamically selectable objects: Simply mouse over any object to dump information about the object. Left click to “lock-on” to the object
- Jana server can, in principle, serve multiple web connections
  - Possibly stream current data taking to anywhere in the world

Future work

- More details:
  - Include lower level info if available
  - MC Truth information
  - Visual clues to associated objects eg “what hits formed this track?”
- Interactions
  - Dynamic track hypothesis switching
    - “What if this track is a Kaon not a Pion?”
  - Dynamic algorithm changes
    - “What would have been seen if…?”
- Web app
  - Single web application
    - Maybe Chromium under the hood?
  - Service multiple connections



Brief tour

https://halldweb.jlab.org/talks/2017/HDvis2/js/event.html

Cell phones out!

Full page

High Memory
May not work on most phones

https://halldweb.jlab.org/talks/2017/HDvis2/js/event.html

Reduced page

Should work on most phones

https://halldweb.jlab.org/talks/2017/HDvis3/js/event.html

Turn to landscape and double tap. If your screen goes black just move the phone around the room (no translations) to find the detectors

Stop by to view an event in Virtual Reality on the Oculus Rift