Geant4

Lecture 1: Introduction

What it does

GEometry ANd Tracking

Simulates the passage of particles through matter

Applications:

- Particle physics - what will detectors register?
- Accelerators - what radiation will escape?
- Nuclear power - what happens in reactors?
- Isotope production - what is created? (Good and bad)
- Medical physics - doses and damage to DNA
- Space science - what damage do cosmic rays do to satellites?

Lots of documentation: see http://geant4.cern.ch. Including “Introduction to Geant4” manual

What can it do?

Lots!

- Different geometrical shapes
- Different materials
- Different particles
- Cross section libraries from eV to TeV
- Different recording and digitising methods

Making the right choices is very important. Incorrect choices can give very slow computations and/or very wrong results

How does it do it?

Totally object oriented. Main program just defines a few classes and tells them to get on with it.

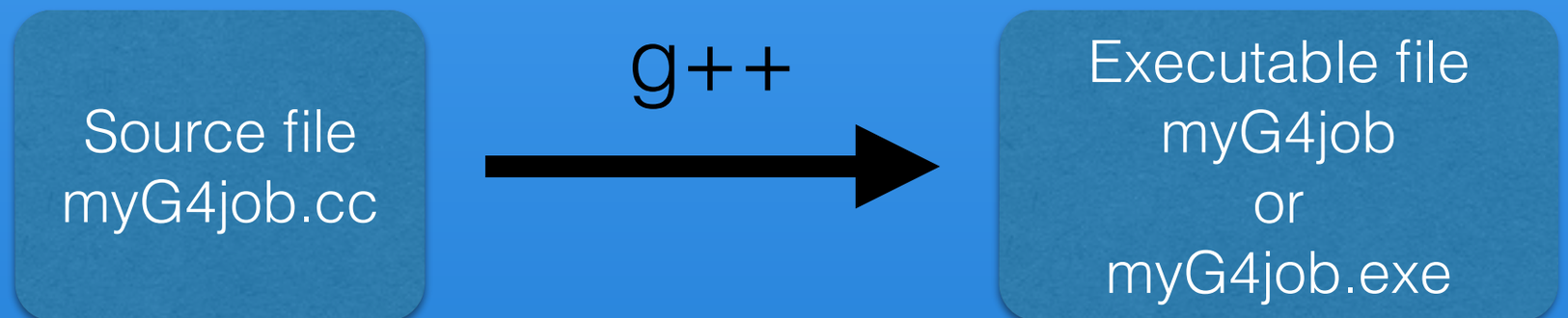
- DetectorConstruction
- PrimaryGeneratorAction
- PhysicsList
- SteppingAction

Example of a main program

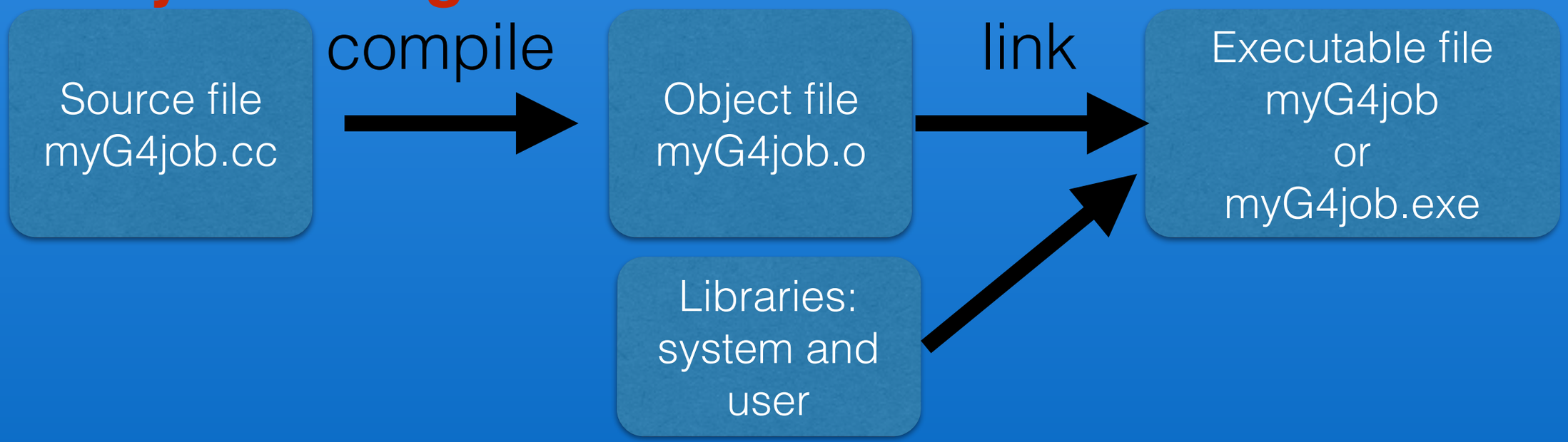
```
#include "B1DetectorConstruction.hh"
#include "B1ActionInitialization.hh"
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "QBBC.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"
#include "Randomize.hh"

int main(int argc,char** argv)
{
  // Detect interactive mode (if no arguments) and define UI session
  //
  G4UIExecutive* ui = 0;
  if ( argc == 1 ) {
    ui = new G4UIExecutive(argc, argv);
  }

  // Choose the Random engine
  G4Random::setTheEngine(new CLHEP::RanecuEngine);

  // Construct the default run manager

  G4RunManager* runManager = new G4RunManager;


  // Set mandatory initialization classes
  //
  // Detector construction
  runManager->SetUserInitialization(new B1DetectorConstruction());

  // Physics list
   G4VModularPhysicsList* physicsList = new QBBC;
   physicsList->SetVerboseLevel(1);
   runManager->SetUserInitialization(physicsList);

   // User action initialization
   runManager->SetUserInitialization(new B1ActionInitialization());

   // Initialize visualization
   //
   G4VisManager* visManager = new G4VisExecutive;
   visManager->Initialize();

   // Get the pointer to the User Interface manager
   G4UImanager* UImanager = G4UImanager::GetUIpointer();

   // Process macro or start UI session
   //
   if ( ! ui ) {     // batch mode
     G4String command = "/control/execute ";
     G4String fileName = argv[1];
     UImanager->ApplyCommand(command+fileName);
   }
   else {    // interactive mode
     UImanager->ApplyCommand("/control/execute init_vis.mac");
     ui->SessionStart();
     delete ui;
   }
   delete visManager;
   delete runManager;
}
```

How to put a program together…

Basic idea

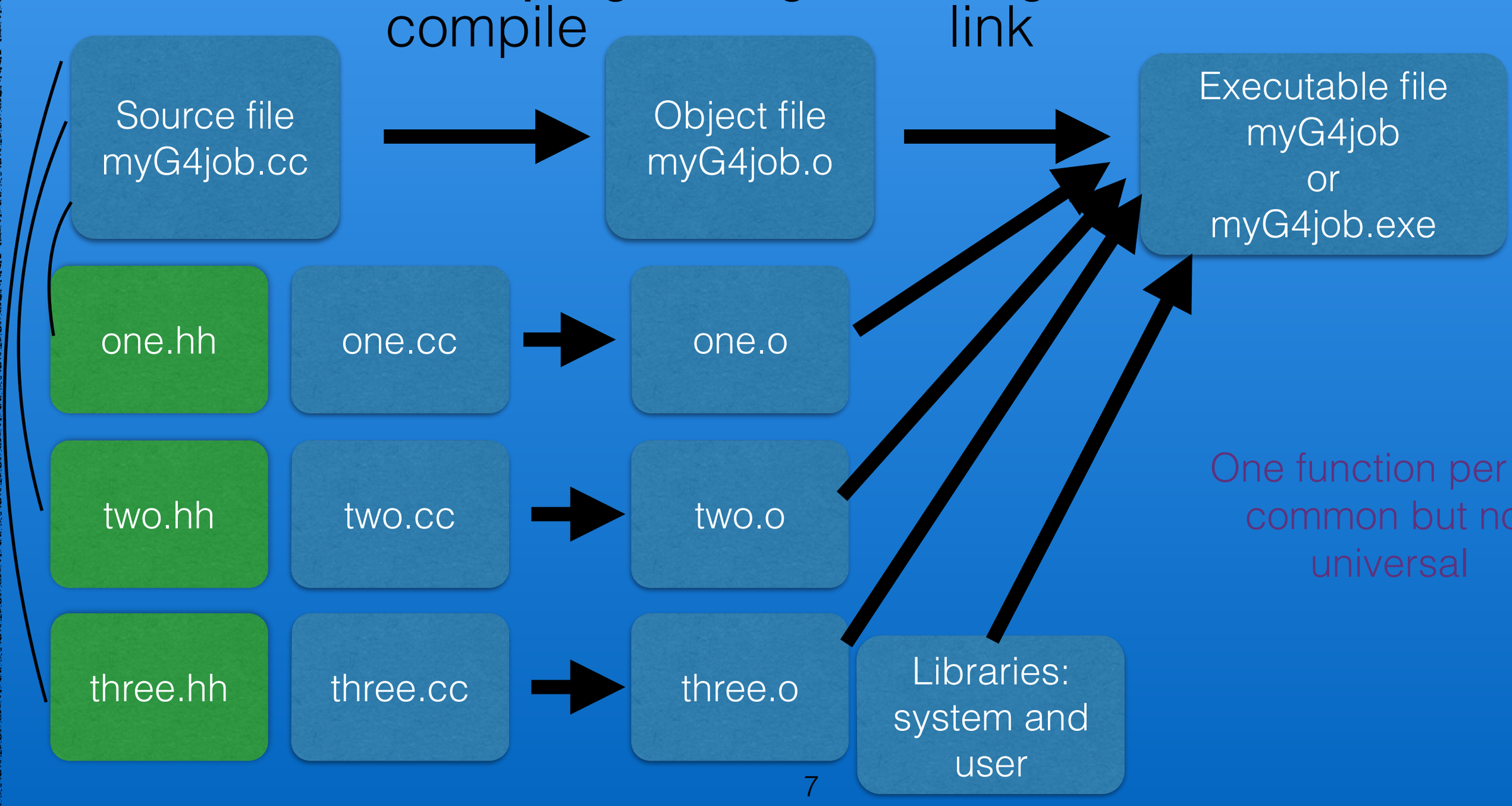


Actually two stages



Which get complicated

g++ command has to specify where to find the include files, what libraries to link and where to find them, and compiler options for optimisation etc. And you need to know what needs recompiling/relinking when changes have been made



One function per file common but not universal

Makefiles to the rescue

A **makefile** contains the command to make a particular file, and a list of the files it depends on - i.e. if any of them are modified, this file must be re-made

Contains stuff like:

```
myG4job.o:  myG4job.cc,one.hh,two.hh,three.hh
g++ -c -O3 -I/home/my/inc myG4job.cc
```

Then just say **`make myG4job.o`** and it will do what's necessary, if necessary.

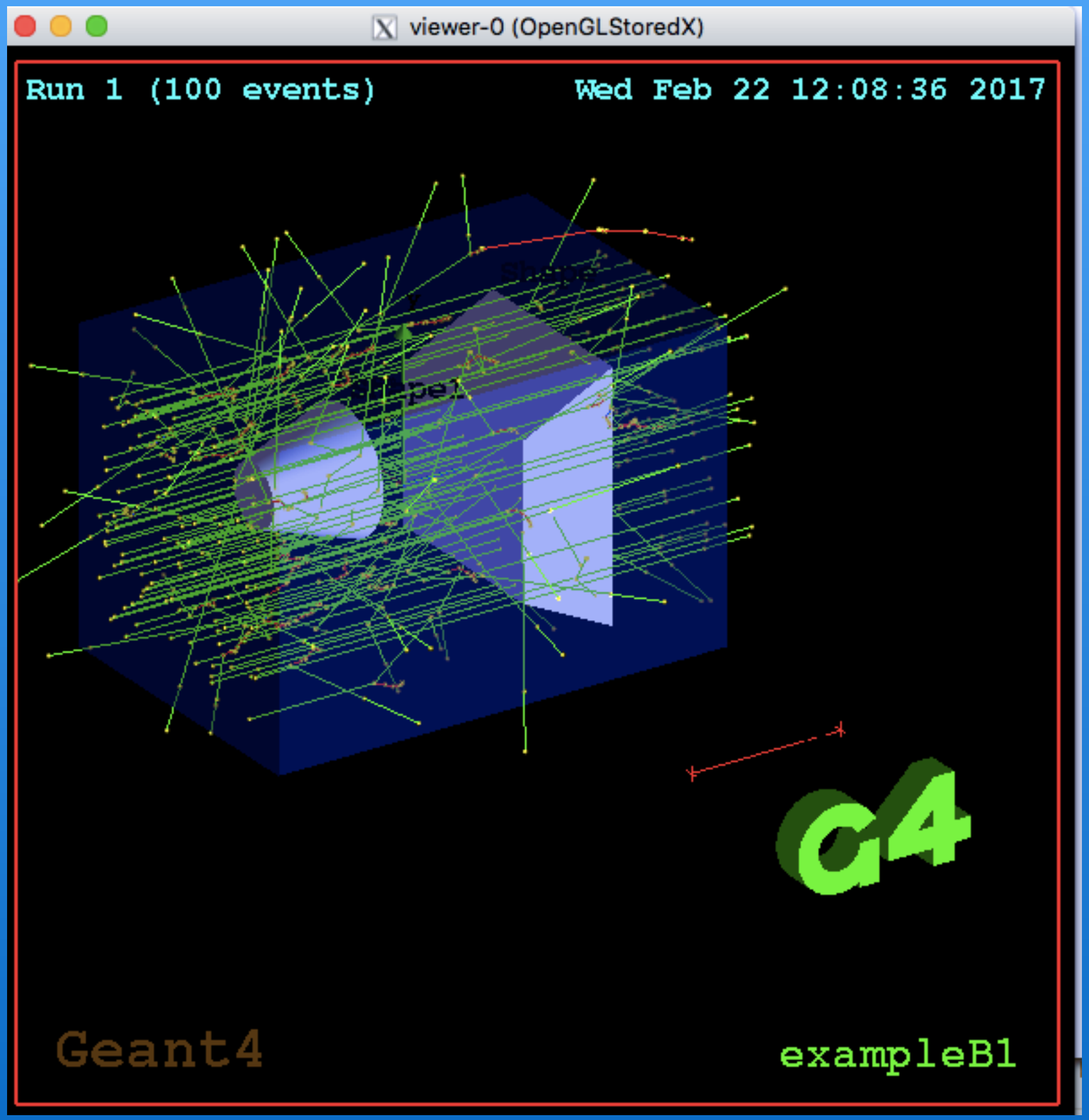
Next problem: writing the make files and keeping them up to date…

CMake to the rescue

- CMake reads high-level instructions and writes a makefile.
- Do not try to edit the makefile! But it is worth looking at

Getting started

1. On a linux desktop/laptop you need X11. On Windows you need putty and Xming
2. From an X11 terminal, log in to IIAA1 with `ssh -XY` yourid@iiaa1.hud.ac.uk (The `-XY` lets you send back pictures. Check with `xclock` or `xeyes`). Or putty with Xauth set and Xming running. [Have you set up your rsa keys yet??]
3. Create a new directory `test` (or whatever) with `mkdir test` and `cd` into it. [Learning some linux will help…]
4. Copy an example job with `cp -r /home/software/geant/geant4.10.03/examples/basic/B1 B1`
5. You need a newer version of cmake than provided with the standard release. So type `alias cmake=/home/software/cmake/cmake-3.7.2/bin/cmake`
6. Set some vital environment variables by `source /home/software/geant/geant4.10.03-install/bin/geant4.sh`
7. To save hassle later, you may want to put steps 5 and 6 in your `.bashrc` file
8. Still in the test directory, do `cmake B1`
9. then do `make B1`
10. then do `./exampleB1` You should get a picture and a prompt saying `idle>`
11. Type `/run/beamOn 100` and then `exit`


viewer-0 (OpenGLStoredX)
Run 1 (100 events)
Wed Feb 22 12:08:36 2017
G4
Geant4
exampleB1

Activity…

Investigate the user interface (the idle> prompt). Try out:

- help
- run
- gun
- vis

Look at the vis.mac file to see some commands - try editing it to see what happens.

Now look at Joe Perl's tutorial

http://geant4.slac.stanford.edu/Presentations/vis/G4VisCommands.pdf

These commands are used interactively and in 'macro' and input files, and from inside the program, so they're worth mastering

More activity

Investigate non-interactive mode by running
`./exampleB1 exampleB1.in`
Try editing the input file using `gedit` (or any other editor):
and seeing what happens. Increase the numbers of particles to simulate.
Now prepare a script file of 3 lines to

1. run the geant4.sh script
2. cd to your directory
3. run exampleB1 with an input file, as above

and submit your job to the batch queue by typing `qsub` followed by the name of the script file
Follow progress with `qstat` and `top`
The output will appear in your directory
type `man qsub` and try out the -N -joe and -W options

Even more activity

`cd` to `B1/src`

Edit (carefully!) `B1DetectorConstruction.cc`

change some dimensions or other shape parameters

remember to `cd` up 2 levels, and type `make B1`

Run the new program and observe the difference

Changing shapes

3-stage definition:

- Solid - the shape and its dimensions
- Logical volume - include materials
- Physical volume - position it in space

These may not make sense - but you get used to them

More on this in the next lecture. For now, look at B1 and undertand it…

Assignment

- In B1, remove the truncated pyramid and change the cone into a sphere, cube or cylinder (to be assigned in class). Plot it.
- Change the colour scheme and the text, and plot it again, giving views from the side and end-on

- Run some 100 MeV protons, neutrons and electrons into your geometry: draw pictures
- Adjust the proton energy until they stop about halfway through the object: draw pictures. What is this energy?
- Write a batch job that will simulate many such protons. Run it several times with different numbers to establish the relation between the number of particles and the time it takes for the job to run. (the linux `time` command may be useful)