The Geant4 User Interface and Visualisation Part 1

Introduction
Installing and building
A basic app
The Qt GUI

INTRODUCTION

Contributors

Aims

Available User Interfaces and Graphics Drivers

Contributors

- Main contributors listed (with approximate start year) – but there is much cross-fertilization, including from the Geometry Category
- **UI systems**
  - Core framework: Hajime Yoshida, Koichi Murakami, and Guy Barrand (1998)
  - Xm GUI: Guy Barrand (1998)
  - Qt GUI: Laurent Garnier (2007), Guy Barrand (2007) and John Allison (2023)
  - Win32 GUI: Guy Barrand (1998), O. Pena-Rodrigue and Gabriele Cosmo (2021)
- **Graphics Reps** (low-level library of graphics primitives, etc)
  - Core primitives and vis manager virtual interface: John Allison (1994)
  - Polyhedron and Boolean processors: Evgueni Tchernaev (1998)
- **Modeling**
  - Core models: John Allison (1994)
  - Trajectory models and filters: Jane Tinslay (2005)
- **Graphics**
  - Core vis manager: John Allison (1994)
  - OpenGL drivers
    - Core scene handling: Andy Walkden and John Allison (1996)
    - Qt extension: Laurent Garnier (2007)
  - Open Inventor drivers: Joe Boudreau (1996), Guy Barrand (2004), Fred Jones (2012)
  - Ray Tracer: Makoto Asai (2000)
  - DAWN and VRML: Satoshi Tanaka (1996)
  - HepRepFile: Joseph Perl (2001)
  - VTK drivers: Stewart Boogert (2021)
  - ToolsSG drivers: Guy Barrand (2021)
  - ASCII Tree: John Allison (2001)
  - Others...

Aims

- The Geant4 UI and vis systems have evolved over many years
- The primary aim: to help G4 app developers
  - Check the geometry
  - Check the interactions
  - Understand and validate their app
  - Provide a custom app for your users
- Secondly, to take advantage of modern graphics libraries
  - An object-oriented interface
  - A multi-functional manager designed around the users' needs
- Each UI and graphics system, and even the manager itself, are effectively plug-ins, and an ambitious user may write his/her own to the abstract interfaces

Available User Interfaces and Graphics Drivers

- User Interfaces
  - Plain terminal
    - csh & tcsh
  - Graphical
    - Xm (motif)
      - In-window graphics, interactive help, and output
    - Win32 (Windows)
      - As above
    - Qt (cross-platform UI system)
      - As above, plus an interactive scene tree
        - Select visibility and colour of geometry objects...and more (described later)

- Graphics Drivers
  - Graphical
    - OpenGL (some advanced features, also export)
    - OpenInventor (some advanced features, VRLM export)
    - ToolsSG (GLES/Qt/ZB, export and plotting)
    - Qt3D (prototype)
    - VTK (many interactive options and export formats)
    - RayTracerX (export jpeg)
    - ...
  - File-writing (always registered):
    - DAWNFILE (browse with DAWN)
    - HepRepFile (browse with HepRApp)
    - VRML2FILE (browse with any web browser)
    - RayTracer (export jpeg)
    - ToolsSGOffscreen (numerous picture formats)
  - Diagnostic (always registered):
    - ASCIITree (always registered, dumps to G4cout)

BASICS

- Building your app, installing libraries, Geant4 build options
- Instantiating the UI and vis manager
- Choosing the UI and graphics driver
- Vis concepts
- Class diagram
- Vis attributes
- View parameters

Installing Qt

- MacOS: **brew install qt**
  - For Qt5: **brew unlink qt** (to avoid linking to Qt6)
- Linux: **apt install qtbase5-dev**
- Windows:
  - Binaries from Qt web site (qt.io)
  - Does **conda** work?

Building with Qt

- You will get the Qt GUI
- You get several graphics drivers
  - OGLI, OGLS, TSG-GLES, Qt3D
- (You can choose at run-time – see later)

```
cmake -Wno-dev --log-level=ERROR \
-DCMAKE_INSTALL_PREFIX=`pwd` \
-DGEANT4_USE_GDML=ON \
-DGEANT4_USE_QT=ON \
-DGEANT4_USE_QT_QT6=ON \
-DCMAKE_PREFIX_PATH=\
"$(brew --prefix qt@6)" \
-DGEANT4_ENABLE_TESTING=ON \
-DGEANT4_USE_FREETYPE=ON \
-G Xcode \
~/Geant4/geant4-dev
```

Other libraries

- Installing X11
  - MacOS: Xquartz
  - Linux: native
  - Windows: X11 not available, but special drivers use the native windowing system.
- Installing VTK
  - MacOS: **brew install vtk**
  - Linux: **apt install vtk**
  - Windows: **conda install vtk**
- Installing Coin3D (Open Inventor)
  - Coind3D (Open Inventor): See https://www.coin3d.org

The UI and vis build options

- From the Installation Guide, Build Options:
  - GEANT4_USE_INVENTOR (DEFAULT : OFF)
  - GEANT4_USE_INVENTOR_QT (DEFAULT : OFF)
  - GEANT4_USE_OPENGL_WIN32 (DEFAULT : OFF, Windows Only)
  - GEANT4_USE_OPENGL_X11 (DEFAULT : OFF, Unix Only)
  - GEANT4_USE_QT (DEFAULT : OFF)
  - GEANT4_USE_QT_QT6 (DEFAULT : OFF)
  - GEANT4_USE_RAYTRACER_X11 (DEFAULT : OFF, Unix only)
  - GEANT4_USE_VTK (DEFAULT : OFF)
  - GEANT4_USE_XM (DEFAULT : OFF, Motif, Unix Only)
- Don't mix X11 and Qt (unless you know what you're doing!!)
- GEANT4_USE_QT brings in:
  - OpenGLQt
  - ToolsSGQt
  - Qt3D (under development)

main()

- A minimal **main()**
- A simple **vis.mac**
- And **gps.mac**
  - Uses **G4GeneralParticleSource**

```
int main(int argc,char** argv) {
  auto ui = new G4UIExecutive(argc,argv);
  auto runMan = G4RunManagerFactory::CreateRunManager();
  runMan->SetUserInitialization(new DetectorConstruction);
  auto physList = G4PhysListFactory().ReferencePhysList();
  runMan->SetUserInitialization(physList);
  runMan->SetUserInitialization(new ActionInitialization);
  auto visMan = new G4VisExecutive;
  visMan->Initialise();
  auto UImanager = G4UImanager::GetUIpointer();
  UImanager->ApplyCommand("/control/execute vis.mac");
  ui->SessionStart();
  delete ui;
  delete visMan;
  delete runMan;
}
```

```
vis.mac
-------
/control/verbose 2
/run/initialize
/vis/open
/vis/drawVolume
/vis/viewer/set/viewpointThetaPhi 110 170 deg
/vis/scene/add/axes
/vis/scene/add/trajectories
/vis/scene/endOfEventAction accumulate
/control/execute gps.mac
```

```
gps.mac
-------
/gps/particle proton
/gps/position 0 0 -20 cm
/gps/direction 0 0 1
/gps/energy 70 MeV
```

At start-up

- On instantiation of **G4UIExecutive**
  - Before instantiation of actual session, so on std::cout

Available UI session types: [Qt, tcsh, csh]

- On instantiation of **G4VisExecutive**
  - Some drivers always registered
    - They do not need an eternal graphics library

You have successfully registered the following graphics systems.
Registered graphics systems are:
ASCIITree (ATree)
DAWNFILE (DAWNFILE)
G4HepRepFile (HepRepFile)
RayTracer (RayTracer)
VRML2FILE (VRML2FILE)
gMocrenFile (gMocrenFile)
TOOLSSG_OFFSCREEN (TSG_OFFSCREEN, TSG_FILE)
OpenGLImmediateQt (OGLIQt, OGLI)
OpenGLStoredQt (OGLSQt, OGL, OGLS)
Qt3D (Qt3D)
TOOLSSG_QT_GLES (TSG_QT_GLES, TSGQt, TSG)
TOOLSSG_QT_ZB (TSG_QT_ZB, TSGQtZB)
You may choose a graphics system (driver) with a parameter of
the command "/vis/open" or "/vis/sceneHandler/create",
or you may omit the driver parameter and choose at run time:
- by argument in the construction of G4VisExecutive;
- by environment variable "G4VIS_DEFAULT_DRIVER";
- by entry in "~/.g4session";
- by build flags.
- Note: This feature is not allowed in batch mode.
For further information see "examples/basic/B1/exampleB1.cc"
and "vis.mac".

Always registered

Depending on options

Choosing the UI and Vis Driver at run time

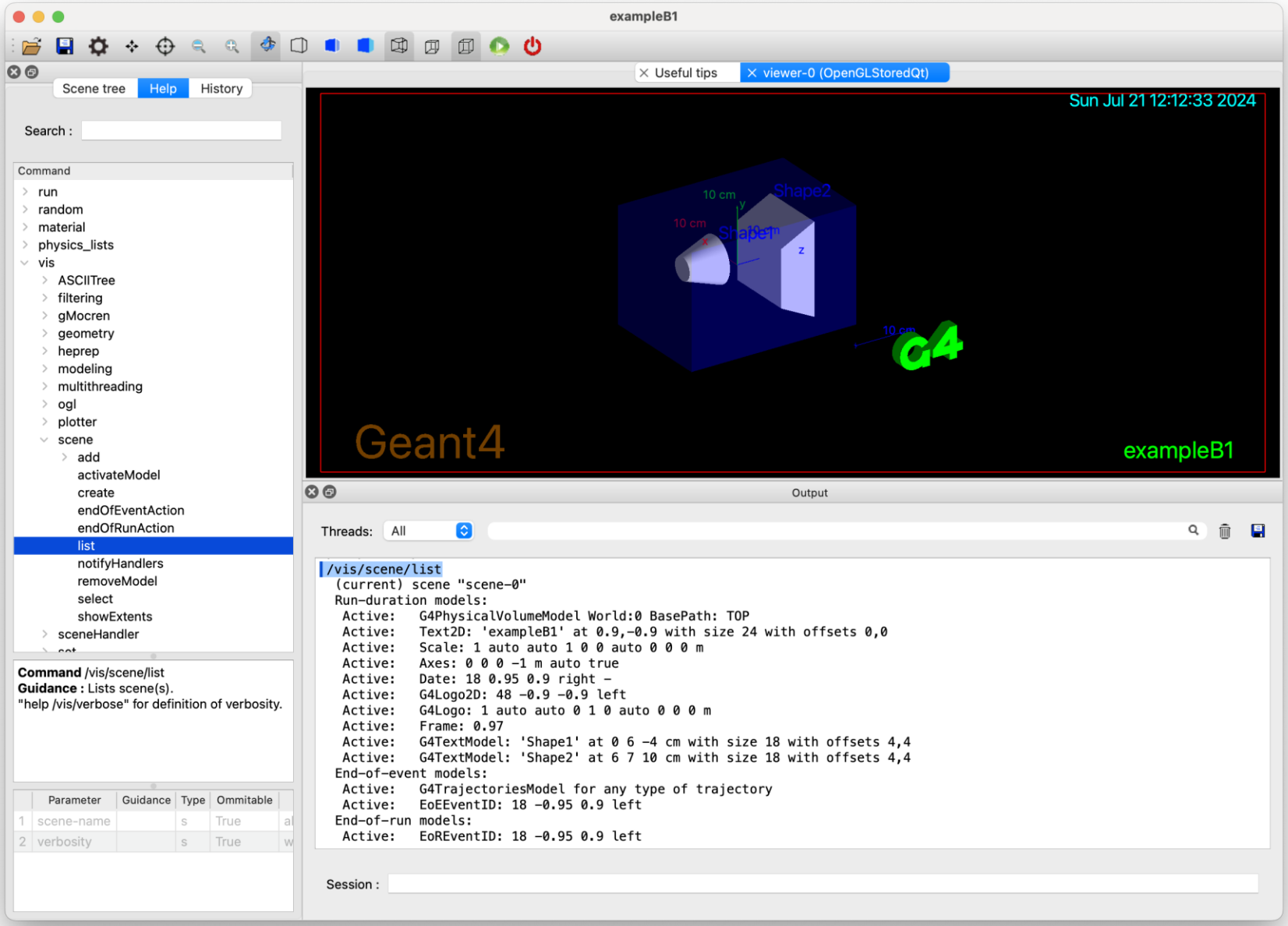
- If you do nothing before start-up, the "best" UI and Vis are chosen according to your original build flags
- Choosing with environment variables
  - **UI**: G4UI_USE_TCSH=1, G4UI_USE_WIN32=1, G4UI_USE_XM=1, or G4UI_USE_QT=1
  - **Vis**: G4VIS_DEFAULT_DRIVER=TSG (or any from the list of registered drivers – previous page)
- Using **~/.g4session**
  - This is a dot (.) file you keep in your home directory (~/)
  - The format is:
    - First line: chosen default UI session: Qt, Xm, Win32, tcsh or csh
    - Subsequent lines: **<app> <session> [<vis>] [<window-size>]**

```
Qt  # Default session
#exampleB1 tcsh
#exampleB1 Qt TSG 1000x1000+0-0
#exampleB1 Qt Qt3D
#exampleB2a tcsh
exampleB2b Qt Qt3D
#exampleB2b tcsh
#exampleB3 tcsh
#mvexampleB4a Xm
exampleB5 Qt TSG
#test202 tcsh
#test202 Qt VTK
#userVisAction tcsh
```

- Note: Environment takes precedence.
- After all the above you can still change driver with **/vis/open [driver]**

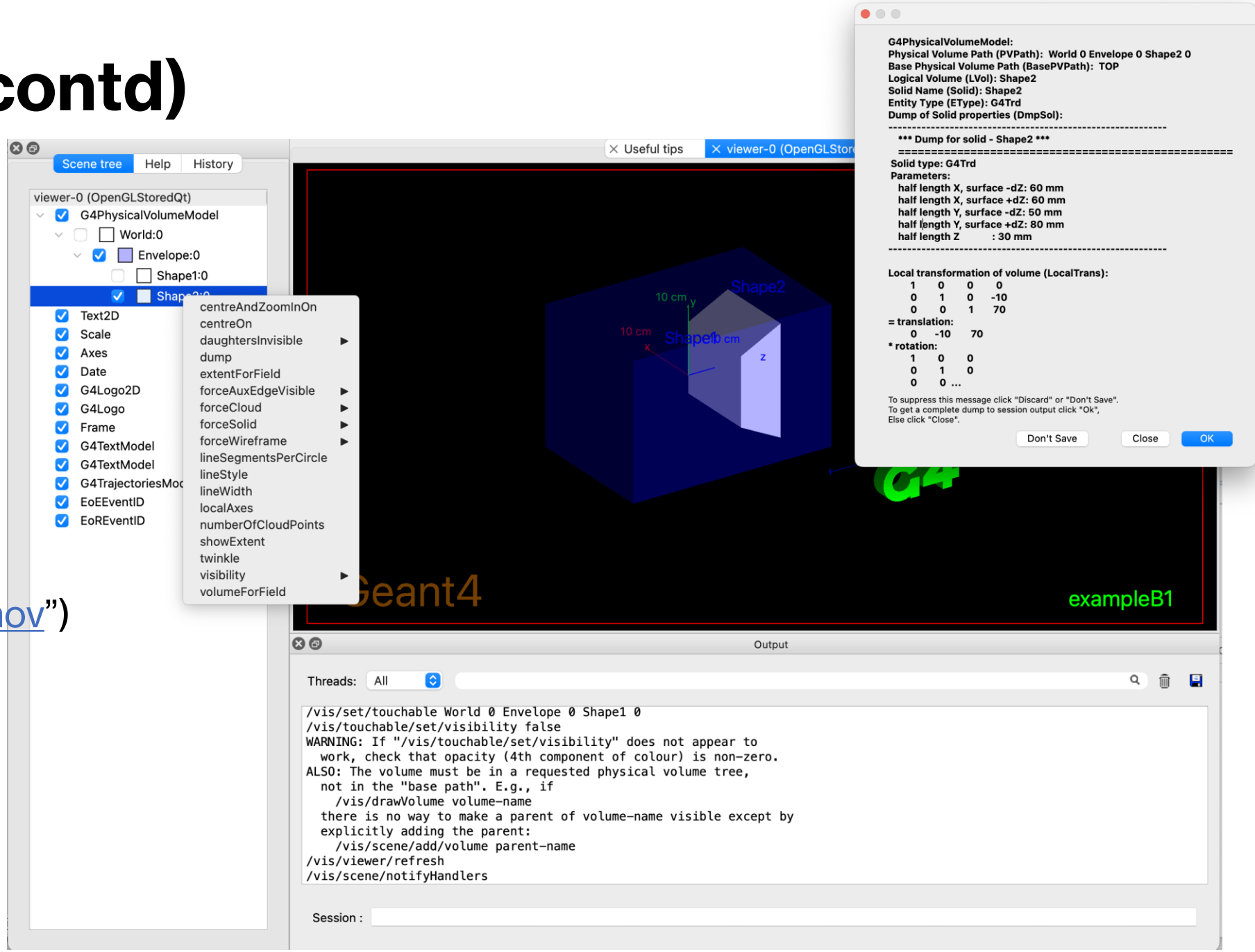
The Qt GUI

- Help
  - Shows selection of **/vis/scene/list**
    - Double-click to transfer to Session window
    - Then complete as required and hit return
- Note: commands are highlighted (if echoed with **/control/verbose 2**)
- Several single action buttons/icons along the top



The Qt GUI (contd)

- Scene tree
  - Click to make (in)visible, or change colour
  - Right-click to get other actions (see video "Scene tree centreAndZoomInOn.mov")
- Plus, of course, you get an interactive graphical window



Adding your own action buttons or icons

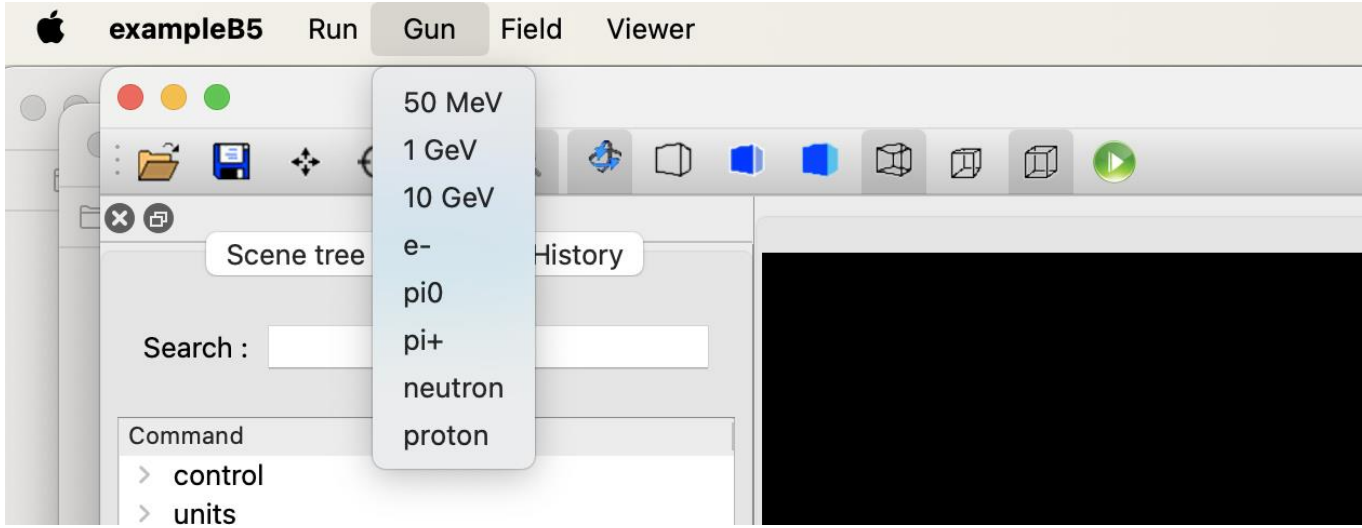
- The Qt GUI has several built-in buttons

  - You can remove, then add back what you want
    - **/gui/defaultIcons false**
    - **/gui/addIcon "Move" move**
    - See **examples/basic/B5/icons.mac**

```
UImanager->ApplyCommand("/control/execute init_vis.mac");
if (ui->IsGUI()) {
  UImanager->ApplyCommand("/control/execute gui.mac");
}
```

- You can add your own **gui.mac** (applies also to the the Xm GUI)
  - See basic examples B2, B4 and B5

```
/gui/addMenu gun Gun
/gui/addButton gun "50 MeV"  "/gun/energy 50 MeV"
/gui/addButton gun "1 GeV"   "/gun/energy 1 GeV"
/gui/addButton gun "10 GeV"  "/gun/energy 10 GeV"
```



Back-up slides

A PrimaryGeneratorAction

- Uses **G4GeneralParticleSource**
  - Instantiated in **PrimaryGeneratorAction**
    - Itself instantiated in **ActionInitialization**

```
#ifndef PrimaryGeneratorAction_hh
#define PrimaryGeneratorAction_hh 1
#include "G4VUserPrimaryGeneratorAction.hh"
class G4GeneralParticleSource;
class PrimaryGeneratorAction :
  public G4VUserPrimaryGeneratorAction {
public:
  PrimaryGeneratorAction();
  ~PrimaryGeneratorAction();
  void GeneratePrimaries(G4Event*);
private:
  G4GeneralParticleSource* fpParticleGun;
};
#endif
```

```
#include "PrimaryGeneratorAction.hh"
#include "G4GeneralParticleSource.hh"
PrimaryGeneratorAction::PrimaryGeneratorAction()
{fpParticleGun  = new G4GeneralParticleSource;}
PrimaryGeneratorAction::~PrimaryGeneratorAction()
{delete fpParticleGun;}
void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{fpParticleGun->GeneratePrimaryVertex(anEvent);}
```