Introduction to Geant4 Visualisation and User Interface

John Allison

Geant4 Users Workshop, SLAC, 18th February 2002

THE UNIVERSITY *of* MANCHESTER

Contents

Introduction - 1

- Variety of user requirements
  - quick drawing of detector components and tracks
  - high quality output for journals
  - special effects for demonstration
  - camera control for debugging geometry
  - detection of geometry overlaps
  - picking of information about drawn objects
  - run control and parameter setting

Introduction - 2

- Design solution
  - Multiple systems
  - Minimal abstract interfaces
    - for visualisation
      - for geometry use only (`G4VGraphicsScene`)
      - for general use (`G4VVisManager`)
    - for user interaction (`G4UIsession`)
- Avoid re-invention
- Exploit added value of systems

Introduction - 3

- Visualisation and UI sessions are "plug-ins" – they *use* Geant4
- Commands are interactivley accessible if a UI session is started
- Users drawing code must be protected by a run-time test...

```
...
G4VVisManager* pVisMan = G4VVisManager::GetConcreteInstance();
if (pVisMan) {
  ...
  pVisMan->Draw(...)
...
```

Introduction - 4

- One can interactively add visualisable objects – such as detetector components, axes, scale – to a scene and use interactive commands to draw them
- One may write one's own code to draw in a valid viewer at any time
  - lines, markers (squares, circles), text, detector components and shapes
- Trajectories and hits have their own `Draw` methods which may be implemented; trajectories and hits may be added to a scene and will be drawn at the end of each event, accumulated or refreshed.

Running Geant4 without vis and UI

- From novice example 1...

```
main() {
  G4RunManager* runManager = new G4RunManager;
  runManager->SetUserInitialization(new ExN01DetectorConstruction);
  runManager->SetUserInitialization(new ExN01PhysicsList);
  runManager->SetUserAction(new ExN01PrimaryGeneratorAction);
  runManager->Initialize();
  runManager->BeamOn(1000);
  delete runManager;
}
```

- ...or instead of `runManager->BeamOn(1000)`

```
...
G4UImanager::GetUIpointer()->ApplyCommand("/run/beamOn 1000");
...
```

- ...or execute a command file...

```
...
UI->ApplyCommand("/control/execute mymacro.g4m");
...
```

Instantiating a Geant4 (G)UI session

- Gains interactive access to built-in commands

```
...
  G4UIsession* session = new G4UIterminal;
  session->SessionStart();
  delete session;
...
```

- Available UI sessions

		environment	resources
Very dumb terminal	`G4UIterminal`	`G4UI_USE_TERMINAL`	
Smart terminal	`G4UIterminal(new G4UItcsh)`	`G4UI_USE_TCSH`	Unix
Smart window	`G4UIGAG`	`G4UI_USE_GAG`	Java
Motif window	`G4UIXm`	`G4UI_USE_XM`	Motif & Xt
Athena window	`G4UIXAW`	`G4UI_USE_XAW`	Athena & Xt
OPACS window	`G4UIWo`	`G4UI_USE_WO`	OPACS
Windows	`G4UIWin32`	`G4UI_USE_WIN32`	Windows

Note: environment assumes use of standard installation; sets corresponding cpp macro.

Instantiating a Geant4 visualisation manager (and drivers) - 1

- The Geant4 visualisation manager
  - supports many commands
  - is supplied with a wide range of graphics drivers, selectable by environment variables if external libraries or packages are required
- The installer (your system manager) has to build the drivers (set `G4VIS_BUILD_...` variables)
- The user has to instantiate the manager *and* drivers (set `G4VIS_USE_...` variables)

Instantiating a Geant4 visualisation manager (and drivers) - 2

- Implement `RegisterGraphicsSystems()`, e.g...

```
class ExN02VisManager: public G4VisManager {
public:
  ExN02VisManager ();
private:
  void RegisterGraphicsSystems ();
};
```

- Sample implementation on Slide 13.)

Instantiating a Geant4 visualisation manager (and drivers) - 3

- Instantiate and initialise...

```
...
#ifdef G4VIS_USE
#include "ExN02VisManager.hh"
#endif
...
int main(int argc,char** argv) {
...
#ifdef G4VIS_USE
  G4VisManager* visManager = new ExN02VisManager;
  visManager->Initialize();
#endif
...
#ifdef G4VIS_USE
  delete visManager;
#endif
...
```

Note: Assumes use of standard installation which sets cpp macro `G4VIS_USE` if environment variable `G4VIS_NONE` is not set.

Available visualisation drivers - 1

- Needing no external libraries (produce files, normally built by default)

		resources
Technical quality	`G4DAWNFILE`	DAWN (to view)
Generic HepRep	`G4HepRepFile`	WIRED (to view)
Ray tracer	`G4RayTracer`	JPEG viewer
VRML	`G4VRMLnFile, n = 1,2`	VRML browser

Note: cpp macros `G4VIS_USE_DAWNFILE`, `G4VIS_USE_HEPREPFILE`, `G4VIS_USE_RAYTRACER` and `G4VIS_USE_VRMLFILE` are set by default. Assumes use of standard installation.

- Needing external libraries and environment

		environment	resources
OPACS	`G4G4Wo, G4Xo`	`G4VIS_USE_OPACS`	OPACS
OpenGL	`G4OpenGL...*`	`G4VIS_USE_OPENGL...*`	OpenGL
Open Inventor	`G4OpenInventor...*`	`G4VIS_USE_OI..*`	Open Inventor

* OpenGL is available for X windows, Motif and (soon) Windows, with and without display lists.
Open Inventor is available for X and (soon) Windows.
Note: environment assumes use of standard installation; sets corresponding cpp macro.

Available visualisation drivers - 2

- Communicate by socket or pipe

		environment	resources
Technical quality	`G4FukuiRenderer`	`G4VIS_USE_DAWN`	DAWN
VRML	`G4VRMLn, n = 1,2`	`G4VIS_USE_VRML`	VRML browser

Note: environment assumes use of standard installation; sets corresponding cpp macro.

- Tree representation of the geometry model

		resources
Ascii output	`G4ASCIITree`	
Java tree	`G4GAGTree`	Java

Note: cpp macros `G4VIS_USE_ASCIITREE` and `G4VIS_USE_GAGTREE`are set by default. Assumes use of standard installation.

Available visualisation drivers - 3

- `RegisterGraphicsSystems()` implementation...

```
void ExN02VisManager::RegisterGraphicsSystems () {
  // Graphics Systems not needing external packages or libraries...
  RegisterGraphicsSystem (new G4ASCIITree);
  RegisterGraphicsSystem (new G4DAWNFILE);
  RegisterGraphicsSystem (new G4GAGTree);
  RegisterGraphicsSystem (new G4HepRepFile);
  RegisterGraphicsSystem (new G4RayTracer);
  RegisterGraphicsSystem (new G4VRML1File);
  RegisterGraphicsSystem (new G4VRML2File);
  // Graphics systems needing external packages or libraries...
#ifdef G4VIS_USE_DAWN
  RegisterGraphicsSystem (new G4FukuiRenderer);
#endif
#ifdef G4VIS_USE_OPACS
  RegisterGraphicsSystem (new G4Wo);
  RegisterGraphicsSystem (new G4Xo);
#endif
#ifdef G4VIS_USE_OPENGLX
  RegisterGraphicsSystem (new G4OpenGLImmediateX);
  RegisterGraphicsSystem (new G4OpenGLStoredX);
#endif
#ifdef G4VIS_USE_OPENGLWIN32
```

```
  RegisterGraphicsSystem (new G4OpenGLImmediateWin32);
  RegisterGraphicsSystem (new G4OpenGLStoredWin32);
#endif
#ifdef G4VIS_USE_OPENGLXM
  RegisterGraphicsSystem (new G4OpenGLImmediateXm);
  RegisterGraphicsSystem (new G4OpenGLStoredXm);
#endif
#ifdef G4VIS_USE_OIX
  RegisterGraphicsSystem (new G4OpenInventorX);
#endif
#ifdef G4VIS_USE_OIWIN32
  RegisterGraphicsSystem (new G4OpenInventorWin32);
#endif
#ifdef G4VIS_USE_VRML
  RegisterGraphicsSystem (new G4VRML1);
  RegisterGraphicsSystem (new G4VRML2);
#endif
  if (fVerbose > 0) {
    G4cout <<
      "\nYou have successfully chosen to use the following graphics systems."
         << G4endl;
    PrintAvailableGraphicsSystems ();
  }
}
```

Visualisation commands - 1

- Important concepts...
  - scene, scene handler, viewer
- Create scene...

```
/vis/scene/create
/vis/scene/add/volume
/vis/scene/add/axes
```

- Create scene handler...

```
/vis/sceneHandler/create OGLIX
```

- Create viewer and draw...

```
/vis/viewer/create
/vis/viewer/set/viewpointThetaPhi 30 30
/vis/viewer/flush  (refresh + update)
```

- Using compound commands...

```
/vis/open OGLSX
/vis/drawVolume
```

Visualisation commands - 2

- Adding trajectories/hits...

```
/tracking/storeTrajectory 1
/vis/scene/add/trajectories
/vis/scene/endOfEventAction accumulate  (or refresh each event)
/run/beamOn 10
```

- Full list in

  `geant4/source/visualization/README.built_in_commands`

- Or use interactive help
- Look at novice example 3 for a range of scenarios

Visualisation attributes

- See

  `http://wwwinfo.cern.ch/asd/geant4/milestones/training/docs/unit2/index.html`,

  Visualization and (G)UI, Section 3.

Writing drawing code

- See

  `http://wwwinfo.cern.ch/asd/geant4/milestones/training/docs/unit2/index.html`,

  Visualization and (G)UI, Section 4.