Introduction to Geant4 Visualisation and User Interface

John Allison

Geant4 Users Workshop, SLAC, 18th February 2002
Contents

- Introduction
- Running Geant4 without visualisation and UI
- Instantiating a Geant4 (G)UI session
- Instantiating a Geant4 visualisation manager (and drivers)
- Available visualisation drivers
- Visualisation commands
- Visualisation attributes
- Writing drawing code
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
Visualisation and UI sessions are “plug-ins” – they use Geant4

Commands are interactively accessible if a UI session is started

Users drawing code must be protected by a run-time test...

```c
...  
G4VVisManager* pVisMan = G4VVisManager::GetConcreteInstance();
if (pVisMan) {
    ...
    pVisMan->Draw(...)
    ...
```
Introduction - 4

- One can interactively add visualisable objects – such as detector 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...

```c
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)`

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

...or execute a command file...

```c
...
    UI->ApplyCommand("/control/execute mymacro.g4m");
...
```
Instantiating a Geant4 (G)UI session

- Gains interactive access to built-in commands

```
...  
G4UISession* session = new G4UItcsh;
session->SessionStart();
delete session;
...
```

- Available UI sessions

<table>
<thead>
<tr>
<th>Environment</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>G4UITcsh</td>
<td>Unix</td>
</tr>
<tr>
<td>G4UIGAG</td>
<td>Java</td>
</tr>
<tr>
<td>G4UIXm</td>
<td>Motif &amp; Xt</td>
</tr>
<tr>
<td>G4UIXAW</td>
<td>Athena &amp; Xt</td>
</tr>
<tr>
<td>G4UIWo</td>
<td>OPACS</td>
</tr>
<tr>
<td>G4UIWin32</td>
<td>Windows</td>
</tr>
</tbody>
</table>

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 ExN2VisManager: public G4VisManager {
      public:
          ExN2VisManager ();
      private:
          void RegisterGraphicsSystems ();
  };
  ```

- Sample implementation on Slide 13.)
Instantiating a Geant4 visualisation manager (and drivers) - 3

Instantiate and initialise...

```cpp
...  
#define G4VIS_USE
#include "ExNO2VisManager.hh"
#endif
...  
int main(int argc,char** argv) {
  ...  
#define G4VIS_USE
  G4VisManager* visManager = new ExNO2VisManager;
  visManager->Initialize();
#define 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)

- Technical quality
- Generic HepRep
- Ray tracer
- VRML

<table>
<thead>
<tr>
<th>Driver</th>
<th>G4DAWNLnFile, n = 1,2</th>
<th>DAWN (to view)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic HepRep</td>
<td>G4HepRepFile</td>
<td>WIRED (to view)</td>
</tr>
<tr>
<td>Ray tracer</td>
<td>G4RayTracer</td>
<td>JPEG viewer</td>
</tr>
<tr>
<td>VRML</td>
<td>G4VRMLnFile, n = 1,2</td>
<td>VRML browser</td>
</tr>
</tbody>
</table>

Note: cpp macros G4VIS_USE_DAWNLFILE, G4VIS_USE_HEPREPFILE, G4VIS_USE_RAYTRACER and G4VIS_USE_VRMLFILE are set by default. Assumes use of standard installation.

Needing external libraries and environment

<table>
<thead>
<tr>
<th>Library</th>
<th>G4G4Wo, G4xo</th>
<th>G4VIS_USE.OPACS</th>
<th>OPACS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPACS</td>
<td>G4OpenGL,...*</td>
<td>G4VIS_USE.OPENGL,...*</td>
<td>OpenGL</td>
</tr>
<tr>
<td>OpenGL</td>
<td>G4OpenInventor,...*</td>
<td>G4VIS_USE.OI,...*</td>
<td>Open Inventor</td>
</tr>
</tbody>
</table>

* 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
  - Technical quality: G4FukuiRenderer
  - VRML: G4VRMLn, n = 1,2
  - environment: G4VIS_USE_DAWN
  - resources: DAWN
  - environment: G4VIS_USE_VRML
  - resources: VRML browser
  
  Note: environment assumes use of standard installation; sets corresponding cpp macro.

- Tree representation of the geometry model
  - Ascii output: G4ASCII Tree
  - Java tree: G4GATree
  
  Note: cpp macros G4VIS_USE_ASCII_TREE and G4VIS_USE_GAG_TREE are set by default. Assumes use of standard installation.
Available visualisation drivers - 3

- RegisterGraphicsSystems() implementation...

```cpp
def ExNO2VisManager::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 G4OpenGLImmediateX);
        RegisterGraphicsSystem (new G4OpenGLStoredX);
    #endif
```
RegisterGraphicsSystem (new G4OpenGLImmediateWin32);
RegisterGraphicsSystem (new G4OpenGLStoredWin32);
#endif
#else
  RegisterGraphicsSystem (new G4OpenGLImmediateXm);
  RegisterGraphicsSystem (new G4OpenGLStoredXm);
#endif
#else
  RegisterGraphicsSystem (new G4openInventorX);
#endif
#else
  RegisterGraphicsSystem (new G4openInventorWin32);
#endif
#else
  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.