Parallel Session 3B
Visualization and User Interface

Geant4 Collaboration Workshop
Sevilla, 24 September 2013

MT Mode latest tag

Visualization done by Master thread, based on event keeping.

Run Duration

Geometry

Decorations

Transient

Trajectories & Hits

~~User Drawn~~

Can draw all events of run together:
vis/scene/endOfEventAction accumulate

But nothing seen until end of run

MT Mode Release 10 plan 1

Add new thread we would call the "Vis Management Thread."

Run Duration

Transient

Geometry

Decorations

Trajectories & Hits

~~User Drawn~~

Can draw each event separately:
vis/scene/endOfEventAction refresh

Can draw all events of run together:
vis/scene/endOfEventAction accumulate

Events seen as soon as any are ready

MT Mode Release 10 plan 2

John's experiments last two days draw GL directly from worker threads.
If this works, MT could have the same full functionality as Sequential.
Other vis drivers should not be a problem.

Run Duration

Transient

Geometry

Decorations

Trajectories & Hits

User Drawn

Can draw each event separately:
vis/scene/endOfEventAction refresh

Can draw all events of run together:
vis/scene/endOfEventAction accumulate

Events seen as soon as any are ready

Harmonizing verbosity between GUI and Vis, but not only !

- A quick grep give :
  - About 315 times the word « WARNING » into a G4cout
  - About 70 times the word « ERROR » into a G4cerr
  - About 210 time the word « ERROR » into a G4cout (160 in vis)
  - About 28 times the word « WARNING » into a G4cerr (0 in vis)

- More serious :
  - 381 std::cerr() (12 in vis, 5 in UI)
  - 348 std::cout() (15 in vis, 5 in UI)
  - ….6 abort() statements outside G4Exception
  - ….22 std::exit()

- **Will review all these and migrate to G4Exception after release 10**

Removing all getenv statements from Geant4

- In the « real life », applications could be run from anywhere without any terminal by double-click ☺
  => In this case, how to deal with environment variables ?

- Creation of a configuration file?
  - All environment variables could be set inside

- **Ben is working on these ideas. See his talk elsewhere.**

Qt5 and Geant4

- Qt5 is available since December 2012
- From code point of view, no major update to do, quite all is backward compatible
- From cmake point, we have to deal with two Qt version and not the same library hierarchy

- **Ben may not have time for release 10.**
  Laurent may work on it. Anyway not critical for release 10.

GL with no graphics card

Useful for automated tests

Or for any user who wants GL graphics from Batch

- Run visualisation in a frame buffer instead of inside a window, thanks to XVFB library (X With Frame Buffer)
- At cmake configuration step, set XVFB_EXECUTABLE
- Get rid of "/vis/open OGL<xxx>" and use "/vis/open OL" instead
- Use /vis/ogl/printEPS into your .mac file
- Add in CMakeList.txt :

```
find_package(Xvfb QUIET)

if(XVFB_FOUND)
   message(STATUS "G4 TESTS: found Xvfb --> run test202")
   GEANT4_ADD_TEST(test202 COMMAND xvfb.sh ${CMAKE_CURRENT_BINARY_DIR}/test202 runo.mac
ENVIRONMENT ${GEANT4_TEST_ENVIRONMENT} TIMEOUT 2000)
endif()
```

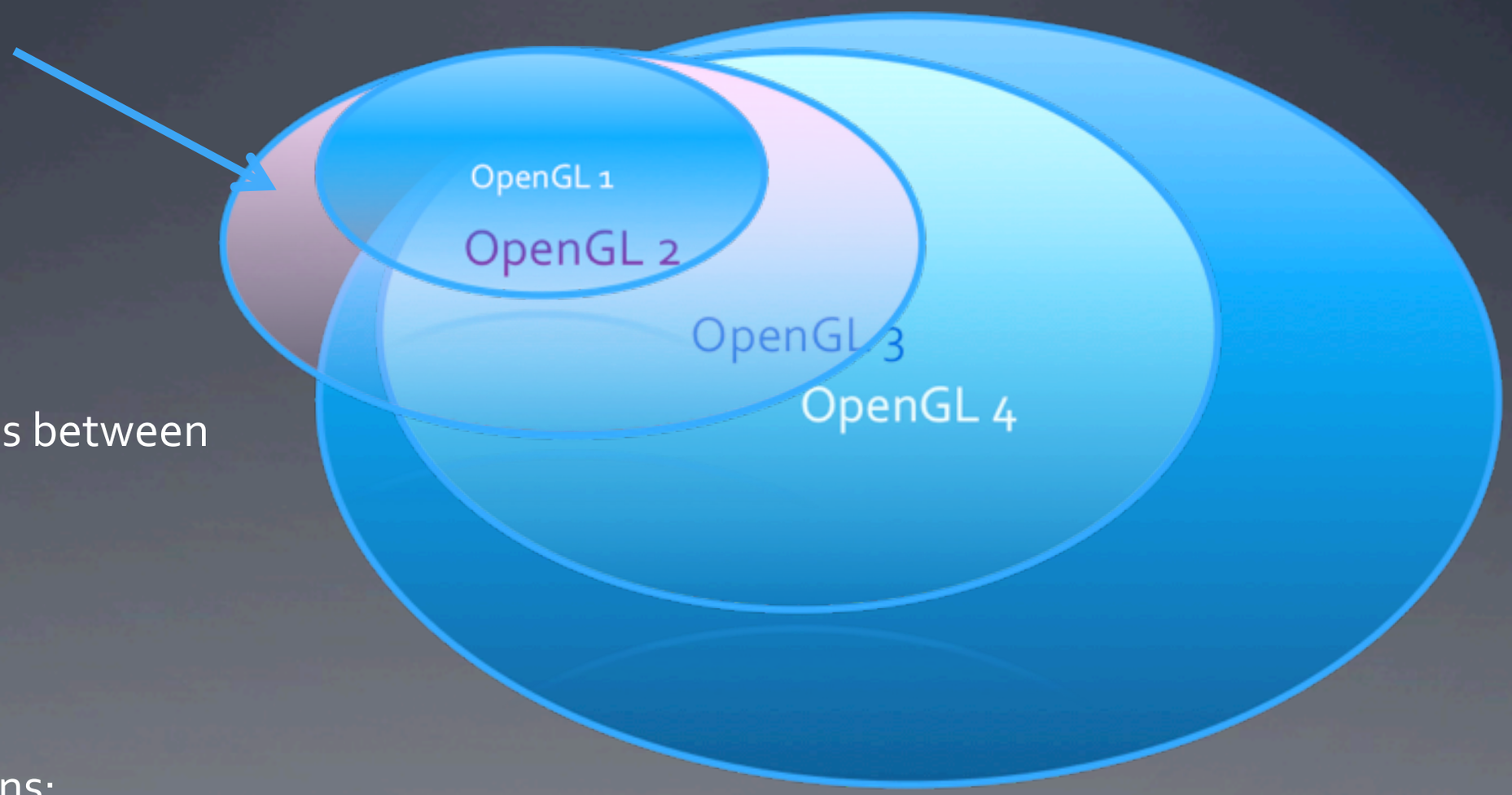
Testing visualisation

- Compare generated pictures against a reference file folder
- Since eps files are text, can just diff them
- Added Charged Geantino to test202 so can even test track and hit drawing
- Will add a field so can also test auxiliary points

OpenGL migration

Risk that Mac OSX Maverick might not support OpenGL2.
Would require a near complete rewrite of our OGL drivers.

Deprecated Function :
- glBegin(...)
- glEnd()
- glCallList(...)
- glColor*
- glMaterial*
- glVertexPointer



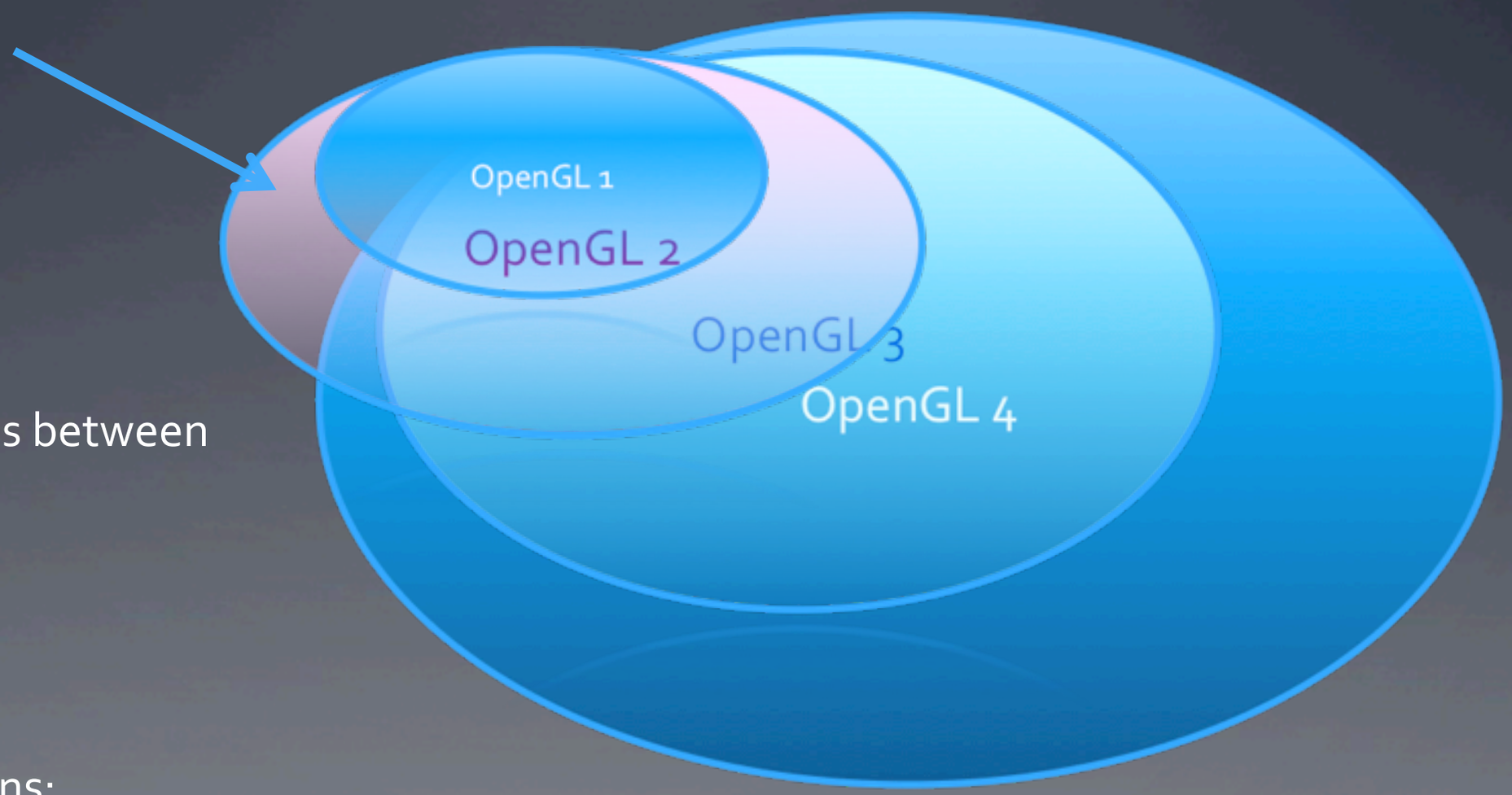
Also anything that goes between glBegin and glEnd:
- glVertex*
- glNormal*
- glTexCoord*
- glMultiTexCoord*

All the matrix operations:
- glRotate*, glTranslate*, glScale*, glMatrixMode(), glLoadIdentity(), glPushMatrix() glPopMatrix(), glFrustum(), gluPerspective(...), gluLookAt(..)

OpenGL migration

Breaking news! Yes, Maverick still supports OpenGL 2 (and even 1)

Deprecated Function :
- glBegin(...)
- glEnd()
- glCallList(...)
- glColor*
- glMaterial*
- glVertexPointer



Also anything that goes between glBegin and glEnd:
- glVertex*
- glNormal*
- glTexCoord*
- glMultiTexCoord*

All the matrix operations:
- glRotate*, glTranslate*, glScale*, glMatrixMode(), glLoadIdentity(), glPushMatrix() glPopMatrix(), glFrustum(), gluPerspective(...), gluLookAt(..)