MANCHESTER
1824
The University of Manchester

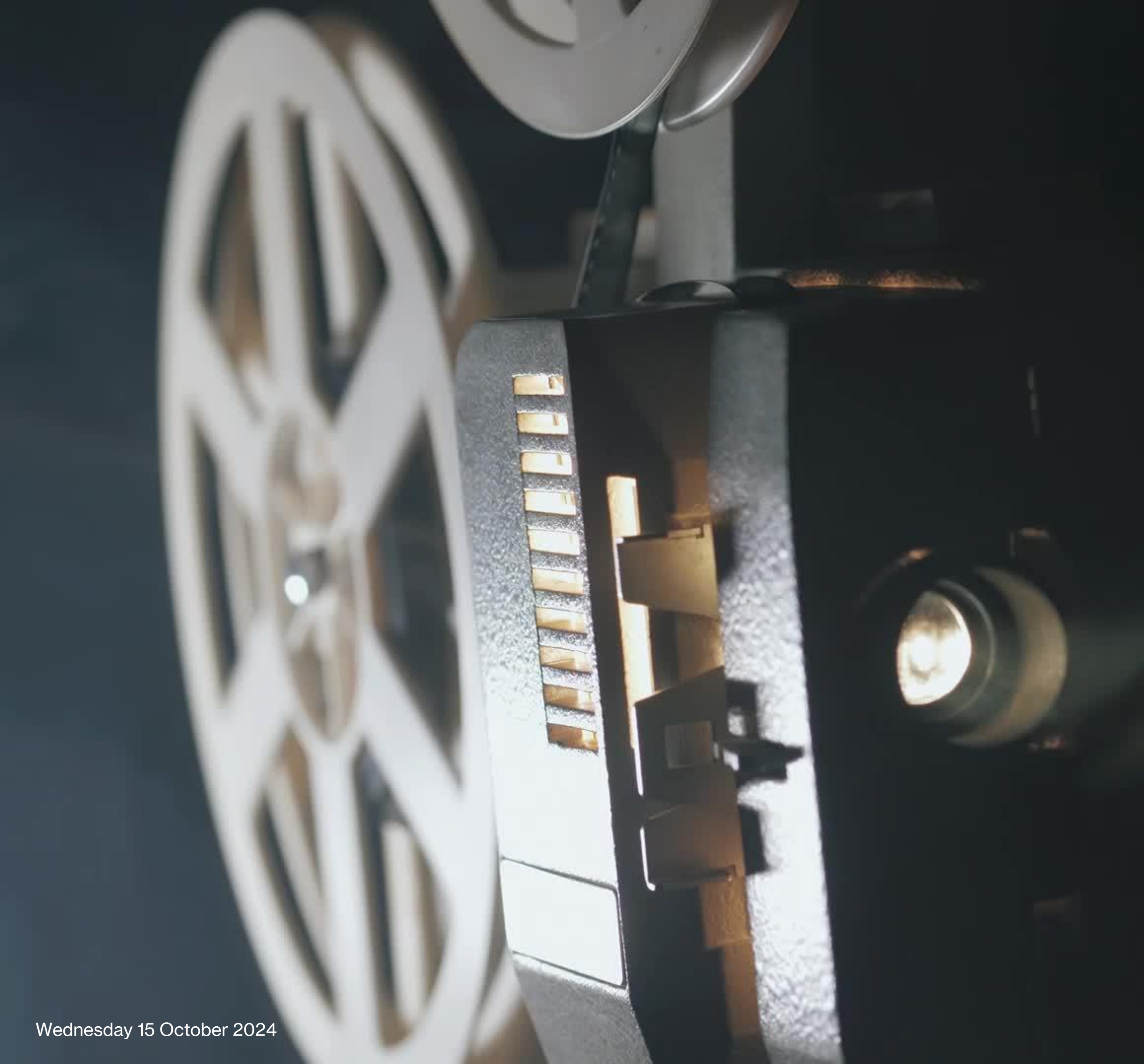
GEANT4
A SIMULATION TOOLKIT

Geant4 ASSOCIATES INTERNATIONAL
Experts in Radiation Simulation

Geant4 User Interface and Visualisation

How do we do that?

The future?



HOW DO WE DO THAT?

- G4VisManager and messengers
- Handling the vis sub-thread
- The GUI scene tree
- G4PhysicalVolumeModel
- Trajectory modeling and filtering
- Vis Attributes Modifiers
- Meshes
- Interpolating views

G4VisManager and messengers

- 21,000 lines of code!!
- Lots of checks, code for printing information, loads list of colours (**/vis/list** to print), verbosity (**/vis/verbose [verbosity]**)
- The vis manager registers over 150 messengers – mostly one per command
  - Some **set** commands share a messenger
- **Draw** methods – are passed on to scene handler
- Intercepts state changes – **Begin/End Of Run/Event**
- Deals with a change of geometry
- For multiple views of the same scene, attempts to keep up to date (**/vis/scene/notifyHandlers**)
- Keeps events – or rather, asks the run manager to keep events until start of next run
  - 100 events by default – may be changed or suppressed
  - The user may independently keep events
    - **G4EventManager::GetEventManager()->KeepTheCurrentEvent()**
  - They may be viewed one by one (**/vis/reviewKeptEvents**)
  - Transients (e.g., trajectories) are not cleared from the view until start of next run
- Handles multithreading by starting a special vis sub-thread at start of run
  - Drivers have to handle timing issues
  - Events (e.g., trajectories) are drawn from this sub-thread in multithreaded mode
    - Not all drivers can handle this; for some, you see events only at end of run
- The user has to instantiate driver factories – **G4VisExecutve**, a sub-class of **G4VisManager**
  - Register the required graphics driver factories (and trajectory and filtering model factories)
    - A factory is an object that knows how to instantiate the *actual* driver – e.g., its scene handler and viewer
  - This is because only the user knows what graphics libraries are available
  - **G4VisManager** must be blind to specific graphic systems and libraries

```
Simple graded message scheme - digit or string (1st character defines):
  0) quiet,         // Nothing is printed.
  1) startup,       // Startup and endup messages are printed...
  2) errors,        // ...and errors...
  3) warnings,      // ...and warnings...
  4) confirmations, // ...and confirming messages...
  5) parameters,    // ...and parameters of scenes and views...
  6) all            // ...and everything available.
```

Handling the vis sub-thread

The GUI scene tree

G4PhysicalVolumeModel

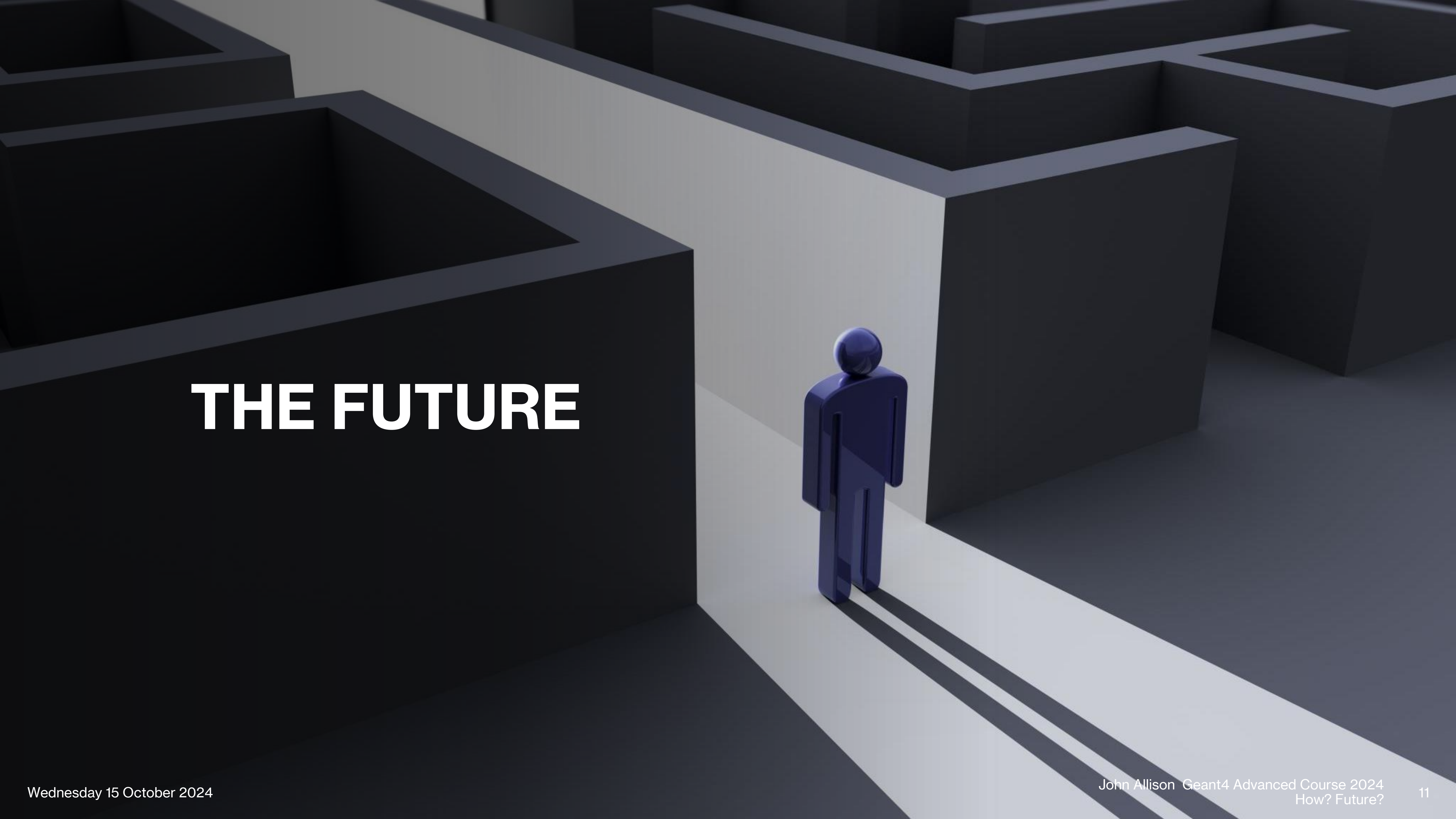
Trajectory modeling and filtering

Vis Attribute Modifiers (VAMs)

Interpolating views

Meshes

- During scanning of the geometry tree, **G4PhysicalVolumeModel** looks for possible meshes and instantiates a temporary **G4Mesh**
  - **G4Mesh** identifies any **G4VNestedParameterisation** in the geometry tree
  - Further descent of the tree is curtailed and the **G4Mesh** is passed to the scene handler
- The concrete scene handler is given a chance to do its own thing, but usually calls **G4VSceneHandler::StandardSpecialMeshRendering**
- For dots, we simply draw one dot at random within each elemental volume (box or tetrahedron)
  - Very fast and gives a nice see-through cloud
- For surfaces, we make use of some really smart algorithms developed by Evgueni Tcherniaev
  - Contiguous surfaces of a given material/colour are eliminated
    - For example, in **ICRP145: Heart**: 15525 tetrahedra (62100 faces) are reduced to 9538 external facets (15%) that are used to make a **G4TessellatedSolid**
    - In all, in **ICRP145**, 8233413 tetrahedra (32933652 faces) are reduced to 4807770 facets (14%)



THE FUTURE

“Prediction is difficult, especially of the future” – unattributed quote!

- The Qt GUI will continue to be our “flagship” UI
- So far, OpenGL has been our “flagship” graphics driver
  - Some features, notably time-slicing (drawing by time), is available *only* with OpenGL
    - Time slicing uses display lists, an OpenGL-1 deprecated feature
  - Cutting is somewhat crude (uses clipping planes)
- ToolsSG parallels OpenGL in almost all respects
  - Quite light and zippy
  - As well as OpenGL-ES, it has a ZB driver (with its own z-buffer), including offscreen
    - Memory-based (non-GPU), thus cannot compete for performance, but OK
    - Requires only the ability to draw or dump a pixel map
  - Has plotting
- VTK: now fully released, and being used
  - Very high performance
  - Interfaces to ParaView (and FreeCAD?)
  - Nice cutters and slicers
  - Can it do zoom-in-on and twinkling?
  - Can it do time-slicing?
- These – and other – drivers will continue to be offered as a choice for users

Maintenance, development and innovation

- This is a never ending task

Thank you