Visualization WG

REMIND

the **goal** of this 1st workshop is to identify:

- the **scope** of the WG activity, its boundaries
- the **questions/challenges** to be answered/addressed by the WG *in the next 6 months*...
- ...with a view of **5-10 years** from now
- if you cite products (like SW packages), do that only as **examples of the functionalities** they provide. In 5-10 years they might disappear and other tools may appear...

WORK PLAN & PRACTICAL CONSIDERATIONS

- Plan future meetings / workshops
- Define and distribute homework
- Identify possible white-papers and ask people/ groups to contribute on specific topics

Riccardo Maria BIANCHI *(Pittsburgh)*

Short report on the WG survey

Short report on the WG survey

- responses submitted so far: 8 complete (out of 20)
- Experiments/groups: ALICE, ATLAS, Belle II, CMS, GSI
- You can find all the responses in the **CVS file** attached to the agenda

We need visualisation for

(quoted directly from one of the responses...it may be from Jeremi)

1. **Detector and reconstruction experts:**
   - check of the position of hits in the detector,
   - compare tracks reconstructed from different detectors with global tracks,
   - to visualise and better understand noice in the detector,
   - to cross-check correctness of reconstruction algorithms.

2. **Physicists - to visualise key aspects of the analysis:**
   - select an event with two back-to-back jets to show what contributes to near-side peak and away-side ridge in the angular correlations analysis,
   - show how particles' shower created in cosmic ray collision with matter near the detectors looks like,
   - visualise decay of some heavy particle to show the principle of cascade reconstruction.

3. **Outreach:**
   - we need to produce beautiful pictures which not necessarily carry a lot of useful scientific information and may not be 100% technically correct (for example reconstruction may not be very accurate due to the lack of precise calibration), but look nice for the general audience and draw their attention to HEP.

The requirements:

- visualisation of all data types (raw data, reconstructed clusters, calorimeter towers, tracks, tracklets, V0s, kinks, cascades...),
- easy navigation between events,
- possibility to find specific event (for example high multiplicity, of with given particle reconstructed etc.),
- geometry visualisation,
- manipulation of the scene (zooming, rotating, moving),
- 2D projections of the 3D scene,
- tracks selection (highlighting single track and getting detailed information about its properties like PID, momentum, reconstruction details),
- possibility to show/hide different parts of visualisation,
- easy change of the appearance (color, line widths, line styles, transparency...),
- automatic, single-click screenshots production (with auto-layout, experiments' logo embedding, printing basic information about the event),
- online mode operation,
- online access for wide audience via website,
- multiplatform (linux/mac for physicists and experts, web-based + mobile platforms for outreach),
- probably many more which I don't recall at the moment.

How our software fulfilled them:

- most of them are fulfilled, but we have 2 separate applications, one for experts, the other one for outreach.

Requirements on event display tools

- User **requirements** about needed features in an event display tool often come from all **Physics, Offline Software** and **Upgrade** groups

- In three main use-cases:

  1. Detector and reconstruction experts

  2. Physicists - to visualize key aspects of the analysis

  3. Outreach

- quote: *“most of them are fulfilled, but we have **2 separate applications**, one for experts, the other one for outreach”*

Desktop graphics libraries

- Packages used: ROOT GL & Eve, Total Event Display, plain OpenGL, Coin3D, hand-coded libs

- Other: Open Inventor, Performer, Open Scene Graph, Ogre3D, Qt3D, Unreal Engine, Unity, Autocad

- quote: *“I don't think there is one that would cover all aspects of what we're discussing here, i.e., a) physics analysis / reco algorithm* ***debugging tools*** *that are really CAD platforms and; b)* ***pretty / immersive*** *rendering of extended surroundings of the detector.”*

Desktop graphical libraries - Issues

ROOT:

- *“sometimes difficult to **control what's being drawn** and what transformations are applied to the scene”*
- *“**animations** [and graphics effects] are difficult to implement”*

Coin3D:

- *“Coin handles **transparency** very poorly, and seems to have **glitches** with large complex events. Rendering boolean shapes is difficult. **It is old**.”*
- Coin is **not maintained** anymore

Unreal Engine:

- very much designed for **games**. Hard/impossible to use with Qt and not obvious how to compile against it without using their build system.

Qt3d:

- no examiner view yet. Quite a lot of **missing functionality** (e.g. instancing).

Desktop graphical libraries - Desiderata

- *“**simple and consistent API** for drawing objects of different types, easy and efficient **animations** of the properties of those objects, clear and working way to save **screenshots**.”*

- *“Ability to **interact with GUI** (e.g. Qt) easily. Good **rendering** ability (e.g. transparency). **VR support**. 'Examiner' style view. **Export to standard 3D formats** e.g. wrl, 3ds. Unencumbered legally. Can link to it to **make standalone apps**.”*

Web-based graphics libraries

- Packages: D3, JavaScript, ROOT JS
- Other: crossfilter
- quotes:
  - *“Visualization of histograms and physics-related distributions. They work really well [but]* ***They can use a lot of resources on the client****”*
  - *“Use in* ***educational*** *programs like masterclasses”*
  - *“WebGL in a browser can allow for creation of high-quality graphics.* ***A browser-based display can reach a large number of users****. Only a modern browser is required. A reliable Internet connection for some users cannot be assumed in some cases; with an application using HTML, CSS, and JavaScript* ***one can run offline and online****.”*
  - *“****limited size of models*** *[but…] no additional viewers […]* ***no installation****, platform independent, click-and-go”*
  - *“Use cases:* ***online analysis monitoring****, […] access to binary* ***ROOT files from the browser****. […] Possibility to* ***extend with user code****.”*
  - *“small details which are* ***different for different browsers*** *or graphical cards”*

GUI libraries

Packages used:

- *Desktop*: Qt, ROOT UI, GSI QtRoot, own package
- *Web*: HTML5, bootstrap.js, jQuery.js

Other packages: FLTK (but old), GEANT4 (clunky)

GUI - Issues

ROOT:

- *“Building a GUI with ROOT's GUI builder is not very convenient, usually it takes a lot of time to take macros generated by the builder and embed them in the compiled code, changing all the objects' names etc. ROOT's GUI is also ugly, but this may change with upcoming ROOT 6”*

- *“numbers assigned to buttons in ROOT are useless”*

Qt + ROOT:

- *“I have long experience of integration Qt-based UI with ROOT graphics - with so called GSI QtRoot interface. Since a while* ***there are no good solutions for MacOS*** *- old X11-based Qt does not longer supported on Mac. Moreover, there are* ***increasing problems with ROOT6 and Qt4/5 integration*** *- due to custom version of LLVM in ROOT6. Probably,* ***on the long run such solution will not work at all****.”*

GUI - desiderata

- smart **auto-layout** features
- **automatic naming** matching given conventions
- Xcode interface builder given as an example of a good GUI builder
- *“**Portability**. Plain analysis code normally runs on any reasonable platform, but not a GUI. There are always a lot of pain to port existing code on new platform - even if underlying software (like Qt) supports it.”*

Math libraries

- Packages used: CLHEP, Eigen, ROOT + own, three.js
- Issues: CLHEP slower compared to Eigen.
- Advantages: CLHEP have **HEP-targeted functionalities** which Eigen and other libraries don’t have

Geometry libraries

Packages/formats used:

- DDD + ROOT TGeo for visualization, GeoModel, GEANT4, Collada, JSON (exported from the experiment framework)

Other:

- EGS4 Geometry package (see below)
- Google *SketchUp,* used to convert the CMS simulation geometry to different 3D formats

Quotes:

- *“EGS4 geometry is entirely outside the EGS4 package and must be built from scratch by end-user”*

Considerations about future tools - I

(quoted directly from one of the responses...it may be from Jeremi)

- I think that more and more focus will be put on outreach activities, which in turn are linked with new and exciting technologies. For me this means that **our visualisation software should be easily portable to any new device which may be launched in the future** (like now we have more and more VR devices, including helmets for smartphones).

- We should be able to prepare application for those new devices without too much effort. Because of that **I would choose some widely used and supported solution**, like Unity, so that there is a serious company working on the interface between their software and new devices and our only job is to download a new version of the libraries, update the application, deploy and test our software on the new device, **rather than implementing some low-level code**. This is also true for let's say 3D TV - this should be one option in our library/editor, rather than spending weeks on our own implementation.

- Finally, I think **we should be opened for new ways of data visualisation, completely different from what we already know**. I've been collaborating with artists willing to visualise our collisions and they have ideas **which are very far from "represent data as 3D objects and draw them on the screen"**. To make this possible **we need a way to export our data to widely-used formats, as well as the geometry**. Those formats should be human-readable, with wide support and possible to import in majority of existing 3D tools and programming languages. Like that **we keep opened for unknown** and do not limit possibilities of data visualisation to what we have in our minds today.

Considerations about future tools - II

(quoted directly from one of the responses...this may be from Tom)

Augmented and virtual reality:

- For the latter there is a developing **WebVR standard**
- iSpy WebGL supports "primitive" VR in the mobile browser using **Google Cardboard**
- **Game engines** like Unity **are a possible way forward**

Support for touch-enabled devices:

- **WebGL** (at least via three.js) supports touch events on the browser **on phones, tablets, and touch screens**
- Again, Unity can work for this as well to **support desktop and mobiles**

References *(to be completed...)*

Tools:

- https://alice-daq.web.cern.ch/products/events-and-geometry-visualisation
- https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookFireworks
- https://www.belle2.org/
- http://atlantis.web.cern.ch/atlantis/
- https://twiki.cern.ch/twiki/bin/viewauth/Atlas/PersintWiki
- https://atlas-tracer.web.cern.ch/
- https://atlas-vp1.web.cern.ch/

Libraries:

- https://bitbucket.org/Coin3D/coin/wiki/Home
- https://d3js.org/
- https://root.cern.ch/js/
- https://www.khronos.org/webgl/
- https://threejs.org/
- https://github.com/square/crossfilter