The IGUANA Interactive Graphics Toolkit with Examples from CMS and D0

George Alverson, Ianna Osborne, Lucas Taylor, Lassi Tuura
(Northeastern University, Boston, USA)

Abstract
IGUANA (Interactive Graphics for User ANAlysis) is a C++ toolkit for developing graphical user interfaces and high performance 2-D and 3-D graphics applications, such as data browsers and detector and event visualisation programs.

The IGUANA strategy is to use freely available software (e.g. Qt, SoQt, OpenInventor, OpenGL, HEPVis) and package and extend it to provide a general-purpose and experiment-independent toolkit.

We describe the evaluation and choices of publicly available GUI/graphics software and the additional functionality currently provided by IGUANA. We demonstrate the use of IGUANA with several applications built for CMS and D0.

Keywords: visualisation, graphics toolkit, detector, event display

1 Introduction
The IGUANA (Interactive Graphics for User Analysis) software project [1] was initiated in mid-1998 to meet the interactive graphics needs of CMS. It is however not specific to CMS and parts have been co-developed and used by members of the D0 and L3 experiments. From the outset, the main IGUANA focus has been interactive detector and event visualisation software. Although early IGUANA prototypes included some interactive analysis modules these are not considered a direct responsibility of IGUANA and further developments have ceased [2]. It is assumed that this functionality will be provided by other tools, such as JAS, Hippodraw, Lizard, OpenScientist, or ROOT. Significant effort has, however, been expended on the IGUANA architecture to ensure that a wide spectrum of external tools can inter-operate smoothly. This aspect of IGUANA is described in the architecture talk by Lassi Tuura [3].

This paper describes: the IGUANA strategy and basic design; the specific technology choices; the extensions we have developed (as demonstrated by a detailed description of the event browsers developed for D0 and CMS from the IGUANA toolkit). It ends with a short description of future extensions.

2 IGUANA Strategy and Design
The reasons for needing interactive detector and event display software are well-accepted. The need for a generic toolkit for this task (rather than a few dedicated programs as in, e.g., the LEP-era) is perhaps not so obvious, especially since it takes more effort to develop generic tools. There are three main motivations for developing IGUANA as a generic toolkit:

1) the need to deploy visualisation programs for a variety of tasks:
   - event display of simulated or real data with the full detector;
   - event display of various and changing test-beam configurations;
   - detector geometry de-bugging tools (closely linked to GEANT3 and/or GEANT4);
   - all of the above for more than one experiment;

2) the need to guarantee high-quality supported software for many years, which is facilitated by a system of loosely-coupled modules which are testable and easy to replace as they become obsolete;
3) the need to maximally leverage developer effort, both from developers using IGUANA as well as those developing public-domain or commercial software which IGUANA can exploit. Arising from the choice to exploit a number of independent toolkits is the need for an open and coherent architecture. The main architectural units of IGUANA, which are currently implemented in C++, are:

- a thin portability and utilities layer;
- a small kernel that manages...
- ...a number of plug-in modules, of the following types:
  - application personalities;
  - a session together with extensions forming the shared application state;
  - user interface components: sites and browsers;
  - representation converter methods to map between the experiment objects and the various browsers (in fact these are just special extensions);
- external software imported into IGUANA for convenience of building and distribution;
- external software which remains outside IGUANA.

The architecture is described in detail elsewhere [3]. The following sections expand on the choices made for the external software and describe how IGUANA has extended the base functionality to provide HEP-tailored modules for interactive detector and event visualisation.

On a prosaic note, IGUANA is written in C++ with a small amount of F77/C code solely for interfacing to GEANT3. It can in principle run on all flavours of Unix and Windows NT/2000, although development is primarily RedHat Linux / Sun Solaris for CMS, standard Fermi Linux / IRIX for D0 (with KAI compiler), with some development also on Windows NT. Details of the download, build, and installation procedures are available online [1]. For CMS and D0 IGUANA is built using the SCRAM and ctbuild build tools respectively; it could also be made available as header/source files and pre-built libraries if required.

3 External Software Choices

The desiderata for choosing an external component are: it should meet the functional requirements [2]; it should expose a clean and consistent API, preferably based on a de-facto or de-jure standard; it should be stable, robust, well supported and well-documented; it should run on (at least) all of the platforms of the experiment; the total cost of ownership, including licenses and both external and in-house support, should be affordable.

IGUANA requires a number of toolkits for the graphical user interface and the low- and high-level 2D and 3D graphics, as well as “glue packages” to enable them to work together, and “extension packages” for generic items such as viewers and HEP-graphics classes. Figure 1 shows the main graphics components of IGUANA, including the main external toolkits.

For the low-level performant graphics rendering OpenGL[4] was chosen since it is prevalent and highly-performant. It is supported by all major vendors and is available free as the Mesa library. OpenGL is, however, a fairly low-level API therefore IGUANA chose to access it through a higher-level API graphics library. We chose OpenInventor[5, 6] but also considered HOOPS and VTF and rejected the non-OO PHIGS which we were previously using[7]. OpenInventor became open-source this year adding to its attractiveness. Some generic high-level HEP-specific graphics extensions are exploited from the HEPVis[8] library of Open Inventor representations of physics objects.

From the outset we decided not to develop our own graphical user interface (GUI) toolkit. Rather, we assessed a number of existing toolkits [9] to replace the non-OO Motif we were previously using, and ultimately we chose Qt[10] from Trolltech (other promising
candidates included: OpenAmulet, Coral, Fltk, FOX, GLUI, gtk, V, and WxWindows. Qt is free for Unix systems and recently has become free for Windows systems as well. Since HEPVis 1999, Qt seems to be becoming increasingly popular in HEP.

Associated to Qt are two “glue” packages for the inter-operability of OpenGL, OpenInventor, and Qt which we exploit: SoQt[11] and QGL. The former provides a basic OpenInventor / Qt viewer component which we extend for use in detector and event visualisation. It is important to note that although we have not built an adaptor interface on top of these toolkits, due to our careful separation of functionality it has not hitherto been too onerous a job to swap between toolkits with similar functionality.

The IGUANA toolkit builds on these packages to provide additional functionality to meet HEP visualisation needs, as shown on Figure 1 and as described in the next sections.

4 Extending the Components

To highlight the use of the IGUANA visualisation toolkit, we describe two applications in current use, ORCAvis and D0Scan. We concentrate on those aspects which are provided solely by IGUANA and not by the component libraries. Figure 2 shows a screen grab of D0Scan. ORCAvis is quite similar, the differences lying in the unseen experiment specific details such as integration with the framework and geometry access.

Object visibility; picking Control of the visibility of the individual representations in the rendered scene is provided by tree-like control structures. Several instances of interfaces for these structures can be seen in the figure (e.g., see item number 3). They are linked with the object they control so that upon a user query (through picking in a 3D view window, item
1) the representation is highlighted (in this case, a muon resulting from a $Z^0$ decay) as is its control object in the tree widget. In general this action will also result in additional detail text (item 4). This will not necessarily be the case for a saved scene read back in (such as that shown in the second 3D view window, item 2).

There is an additional method of controlling visibility within the scene: to apply a threshold to some internal quantity such as an ADC level or momentum. The threshold slider widget provides this functionality (see item 5).

**Scene control** To manipulate the scene as a whole, three additional additional widgets have been provided. User-defined viewpoints can be created within the scene using the viewpoint widget (item 7). Regions of the scene can be excluded from view using either the slicer widget (item 8) or by creating an arbitrary set of clipping planes (not shown).

**Printing** A print widget has been added that allows printing the scene in any of the formats available from the underlying Open Inventor library (item 6).

**Additional features** Controls to snap the viewpoint to one of the x, y, or z axes were added to the original viewer, as were tooltips (as in the message “Invert co-ords through origin”, next to the push button which implements that function). There is also a memory feature which saves the status of the widgets until the next invocation of the application.

5 Further extensions and enhancements

Although the existing toolkit is fully functional and performant, there are areas where additional enhancements can occur. The visualisation toolkit is strictly three dimensional; the layering, grouping, and vector graphics output features of most 2D toolkits are missing. The integration with GEANT4 needs to be improved so that geometry debugging tools are easier to create and use. Both of these areas are currently under development, in at least one case by individuals outside the IGUANA team. We feel this indicates that our original intent, that the IGUANA toolkit be both easily used and readily extended, has been realized.

References

[3] A Coherent and Non-Invasive Open Analysis Architecture and Framework with Applications in CMS (talk number 3-039)
[8] HEPVis http://www-pat.fnal.gov/graphics/HEPVis/www/
[10] Qt http://www.trolltech.com