



Carnac

2D Graphics Development Environment

Carnac is a powerful set of libraries to facilitate the development of sophisticated 2D graphical applications in C++/Qt, Java, or C#/.NET languages.

Carnac greatly simplifies the task of building interactive graphics displays required in scientific and business applications, by providing an infrastructure that handles the complexity of graphics programming, including hardcopy output, object selection and editing, performance optimization, and layout of complex plots.

Carnac is ideal for replacing components in legacy applications and can be combined with other visualization packages. It serves as an excellent migration path from platform specific code to cross platform code that meets high performance needs. Using Carnac, developers reap substantial benefits such as rapid development, code reuse, and lower maintenance costs.

Unique Advantages

Ease of Use

Carnac provides an intuitive graphics environment where users can model and organize their data in terms of shapes, attributes, and layers. A model can be attached to a view for visualization or printing. Selection strategies and manipulators are provided for easy selection and editing of shapes inside a view.

High Performance

Carnac is designed with performance as a priority, and every component is tailored to optimally drive the native graphics pipeline; therefore, most applications written using Carnac will be faster than applications written using native graphics directly.

Extensible

An underlying toolkit should not limit the capabilities of advanced users, so Carnac has been designed to be extensible. Programmers can create their own graphics entities and attributes. For instance, it is easy for developers to create custom shapes, provide a special editing handler for shapes, or customize the axis object to generate their own labels.

FEATURES

- Chart library for stunning charts with minimum effort
- Extensive library of primitive shapes
- Different shape attributes can be set per target device
- Extensive support for selection, scaling, reshaping, and rotation
- Variety of scene implementations to handle differing data cases such as caching or generating shapes on-the-fly
- Support for view layering, filtering, and buffering, including embedded views
- Thread safe - can implement rendering as a separate thread
- Support for binary and ASCII serialization of scenes and palettes
- Online documentation in PDF format for easy access
- Hardcopy - PostScript support; CGM available
- C++ version supports Qt4

