Observer/command 模式

Command 模式

对象（vtkRender）发送一个命令,然后执行观察者相应的动作

Void StartEvent() -> ExecuteStart()

Void EndEvent() -> ExecuteEnd()

若有很多命令，同时有相应的动作，如何统一一个接口？

Void InvokeEvent(Event) -> Execute()

命令映射，从而使每个具体命令有唯一标识符（命令的名字作为主键）,用带命令标识符参数的函数统一接口InvokeEvent(Event),对于被调用的对象依次发起的每个命令(如StartEvent, EndEvent, ProcessEvent)，在客户端注册的观察者集合中查找匹配的命令，并执行客户端的动作

动作通过接口继承统一接口Execute().

Observer模式

客户端注册命令和相应的动作（事件和相应的回调函数作为一个观察者），从而可以在对象运行时，了解对象的状态。（多个观察者接注册顺序添加到对象的变量列表中如vtkRender的vtkObservers，从而有机会被通知到干活）

应用程序运行时，会依次发起客户端注册的事件，从而触发客户端的回调函数.(一对多)

（另一个例子是界面语言切换，发起事件，从而每个注册的界面切换语言）



VTK 使用心得

VTK source directory Structure

InfoVis

-----classes for information visualization

Views

-----classes for viewing data including filters, visualization, interaction and selection

Common

-----core classes

Filtering

-----classes related to data processing in the visualization pipeline

GenericFiltering

-----an adaptor framework to interface VTK to external simulation packages

GeoVis

-----view, sources and other objects useful in terrain visualization

Graphics

-----filters that process 3D data

GUISupport

-----classes for using VTK with the MFC and Qt user interface packages

Hybrid

-----complex classes that depend on classes in multiple other directories

Imaging

-----image processing filters

IO

-----classes for reading and writing data

Parallel

-----classes used to render

Utilities

-----supporting software like expat, png, jpeg, tiff and zlib

VolumeRendering

-----classes used for volume rendering

Widgets

-----3D widget classes

Wrapping

-----support for Tcl, Python, and Java wrapping.

Examples

-----examples, grouped by topic

CMake

-----configuration files for cross-platform building

Application software: ParaView

Except creating VTK application using the Tcl ( in this case, pre-compiled binaries may be available for the windows platform), you will have to compile and link the source code to produce libraries and executables.

VTK = visualization pipeline + rendering engine

Visualization pipeline is used to acquire or create data, process that data, and either write the results to a file or pass the results to the rendering engine for display

Rendering engine is responsible for creating a visual representation of the data

Actors: serves to group rendering attributes such as surface properties(e.g., ambient, diffuse, and specular color), representation(e.g., surface or wireframe), texture maps, and a geometric definition(a mapper)

Mappers: geometric definition using analytic primitives such as points, lines, polygons and triangle strips, the mapper terminates the visualization pipeline and serves as the bridge between the visualization subsystem and the graphics subsystem

Coordinate systems:

Display: x-y pixel values in the rendering window, the original is the lower-left corner

View: x-y-z(-1,1) values in camera coordinates(z is depth)

Operation performed on image data in VTK: image processing + geometry extraction + direct rendering