How Human Factors Drove the Design and Implementation of the Virtual Windtunnel

Steve Bryson
NASA Ames Research Center
http://science.nas.nasa.gov/~bryson
http://science.nas.nasa.gov/Software/VWT

Outline

- Introduction
- Design
  - The Application Task
  - Primary Challenges
  - Initial Choices
- Implementation
  - Software Design
  - Run-time Architecture
  - Interface Design

Introduction

- **Virtual Windtunnel: Virtual reality for the visualization of CFD simulations**
  - **volumes of vector and scalar data on complex meshes**
  - **pre-computed files**
  - **variety of visualization techniques**
    - ∞streamlines
    - ∞isosurfaces
    - ∞contour planes
    - ∞etc...

The Application Task

- Exploration of spatially complex simulated phenomena in 3D volume
- Different users looking for different things
  - vortical structure
  - pressure distribution
  - overall sense of flow
- Ease of use with many capabilities

Primary Challenges

- High performance
  - computationally intensive
- Versatile interface
- Very large amounts of data (> 200 GB)
- Extensibility to new interfaces and capabilities
- Conservative user community
- Distributed operation

Initial Choices
(not an historical account)

- Support a variety of direct manipulation interfaces
  - any visual display with sufficient resolution/comfort
    - ∞disqualifies head-mounts
  - any user input technology
- Place all control within environment
- Decouple computation and interaction
- All data resident in memory
- Use an object-oriented approach, allows easy addition

Initial Choices: User Control

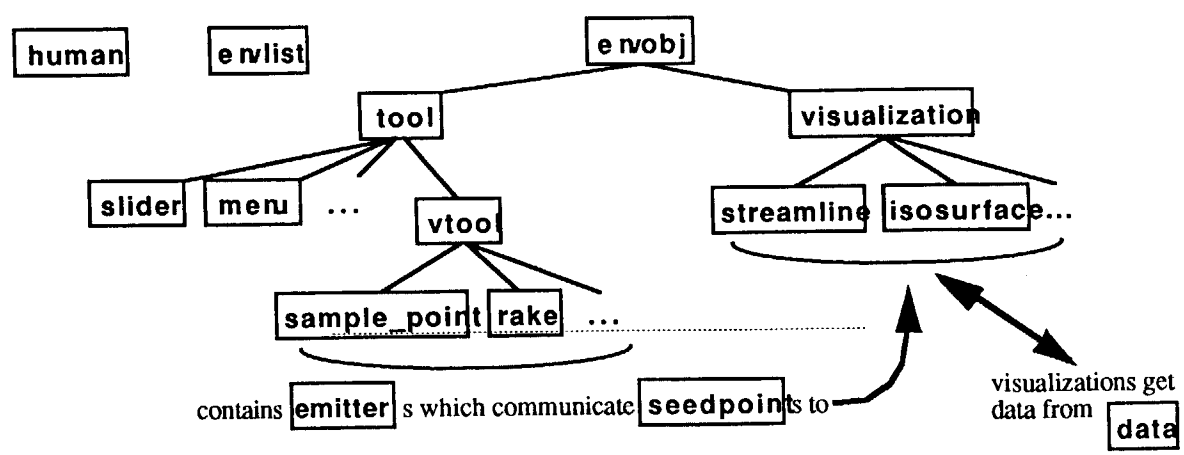
- Based on direct manipulation
- Restrict to two active gestures
  - grab and point
- Reject "direct manipulation everywhere"
  - very crowded scenes
  - many visualizations move
- Use "visualization control tools"
  - groups visualizations in a natural way
  - metaphors from real wind tunnels

Implementation: Software Design

- Implemented in C++ and openGL
- Strong encapsulation
  - each object manages all aspects of itself
  - specified high-level interface between objects
- User interacts with *tool* objects
  - some tools control visualizations (vtools)
  - some tools control environment
- Visualizations access *data* object(s)

Class Hierarchy

- Tools and visualizations are *environment objects*
  - managed by *environment list* object
- Human(s) and data are special global objects

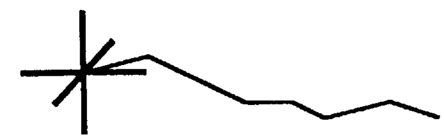


Visualizations

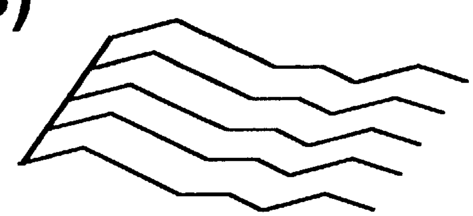
- Vector visualization
  - streamlines, etc
- Scalar visualization
  - (local) isosurfaces
  - cutting/contour planes
  - color-mapped objects
  - grid planes
- Vector and Scalar visualization
  - numerical values

Visualization control tools (vtools)

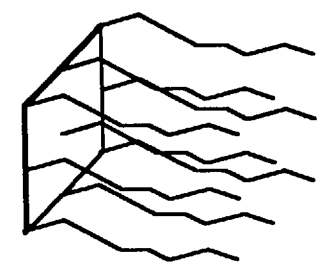
- Vtools contain emitters of visualizations
  - emitter can emit up to five visualization types
- Sample point (single emitter)



- Rake (line of emitters)



- Plane (2D array of emitters)
  - single emitter for cutting/contour plane



Non-Direct-Manipulation Visualizations

- Global Visualizations
  - controlled via menu and sliders
- Pre-computed graphics data
  - allows much larger data sets to be non-interactively viewed
    - ∞e.g. results of batch visualizations

Other interface tools

- Pop-up menus for command selection
  - “point in space” paradigm
  - hierarchical
  - acts on current vtool
  - depends on current vtool
- Sliders to control parameters of environment/current vtool
  - 1D, 2D, 3D

Extensibility: Adding a Visualization

- fill in *compute()* and *draw()* routines according to template
  - interaction with vtools happens "automatically": managed at higher level of class hierarchy
  - use of some obtuse calls required, examples in template
- developer only needs to know how to compute and draw visualization
  - uses local data supplied by vtools
  - does not need to understand rest of VWT

Extensibility: Adding a vtool

- fill in *find()*, *grab()*, and *draw()* routines according to template
  - find() and grab() use human object as input
  - developer determines what “find” and “grab” mean
  - requires somewhat deeper knowledge of vwt
- Success
  - several visualizations have been added
    - ∞by colleagues and summer students

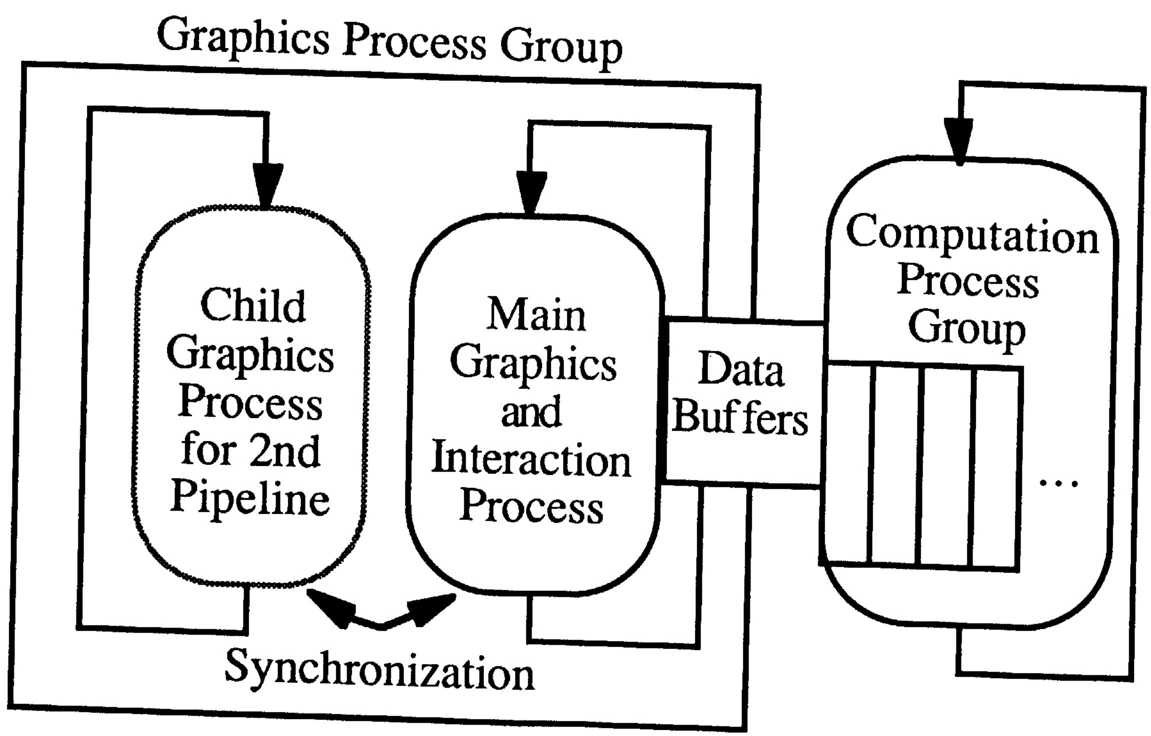
Environment Control

- **Control commands come from several sources**
  - **menus, sliders, startup scripts, keyboard**
  - **future: voice control, text input**
- ***Command*** **and** ***Actuator*** **classes**
  - **Command encapsulates action of command**
    - ∞API is text and value based
  - **Actuator is a subclass of tool which contains a command**
  - **add a command and assign it to an actuator**

Implementation: Run-Time Architecture

- Interaction > 10 fps, < 0.1 sec latency
  - both display and control
- Computation > 2 fps < 0.5 sec latency
  - determined by user
- Implies that interaction and computation take place in two asynchronous process groups
  - issues of communication and synchronization

Overall Run-Time Architecture



Computation Process Group

- Parallelize across "non-parallel" visualizations
  - streamlines, iterative isosurfaces, etc.
  - no attempt to load-balance
- Allow "internally parallel" visualizations
  - global isosurfaces
- So there is a parallel and non-parallel phase

Graphics Process Group

- Multi-processed to support multiple graphics pipelines
- Includes sampling of I/O devices
- Includes setting of human states
  - current graphics transformations
  - location of human “parts”
  - current gesture(s)
- Processing of user input

Computation-Graphics Communication

- **Many demands**
  - **efficiency, simultaneity, reversible time...**
    - ∞e.g. time stops, one visualization moved
- **Handle on an object-by-object basis**
  - **communication buffers for each object**
  - **use various senses of time to choose buffer**
- **Provide API to hide this complexity**
- **Converts gracefully to distributed operation**

Visualization Computation

- User specifies the desired computational frame rate
  - does not effect display/interaction frame rate (one hopes)
- VWT distributes specified frame time over visualizations
  - non-trivial
- Each visualization determines how to do best job in given time
  - time-critical computing

Implementation: Display

- 3D presence very important
- Immersion less important
  - users often relate to simulation as model
    - ∞immersive workbench
    - ∞"fishtank" VR display
- Display quality very important
- Comfort very important
- Work at many scales
  - build into navigation paradigm

Implementation: User control

- Support both 3D and 2D manipulation
  - trackers and mice
  - *find()* and *grab()* in tools come in both 2D and 3D versions
- Maintain the same interface for all devices
  - menus in environment even when in workstation/mouse mode
- Workstation/mouse mode popular
  - fast, cheap, and in control

Near Future

- Remove data-in-memory limitation through very fast access from disk
  - requires files in special format
  - active research area in our group
- Problem when there are a large number of widgets
  - explore methods of grouping

Unresolved Issues: Evaluation

- Large System Evaluation is very difficult
  - can test pieces, but this does not inform how pieces work together
  - can test whole system, but don't have sense of what in particular is good/bad
  - problem is one of high dimensionality
- User feedback is best