A Novel Approach to Designing an Event Display

Chris Jones
Paul Avery
University of Florida

www.phys.ufl.edu/~hepvis/spectator.html
Overview

- Goals
- User Centric Design
- Design Method
  - Design
  - Prototype
  - User Testing
- Use
- Conclusion
Goals

- Easy to use
  - user centric design
- Extensible
  - object oriented language
  - modular
- Maintainable
- Run on X terminal
  - advanced features may require special hardware
- Provide many different views of data
  - not just 3D
User Centric Design

• Idea: Concentrate on how users interact with the system

• Reference:
  – “The Design of Everyday Things” Donald Norman
    • Explains how people interact with objects
    • Philosophy: “If it’s hard to use, it’s poorly designed”

• One Element of User Centric Design: Conceptual Model
  – User’s mental model of the system
  – Formed by the interface presented to the user
  – Good Conceptual Model
    • Builds on concepts the user already understands
    • Matches the actual internal model of the system
Objects in an Event

- Use objects familiar to physicists
- Group objects into categories based on
  - similar purpose
  - same life expectancy in program
- Object categories we use
  - Detector Components
  - Response
  - Reconstruction
  - Monte Carlo
  - Analysis
  - User Defined
Objects Categories

- **Detector Components**
  - Physical and logical detector parts
    - drift chamber endplate

- **Response**
  - Response of the detector to a particle
    - drift chamber hit

- **Reconstruction**
  - Pattern recognition on response items
    - track
Object Categories (continued)

- Monte Carlo
  - Objects created by simulation
    - pion
- Analysis
  - Objects found in an analysis
    - reconstructed B
- User Defined
  - Annotations added by user
    - labels, pointers, etc.
Object Categories

Objects in an Event

Links between categories

drift chamber

position

hits

fit from

track

inferred by

pion

user

?
Interacting With Objects

- Physicists want to interact with objects in different ways
  - change display properties
    - e.g. color and visible/invisible
  - create new objects
    - e.g. make a Kaon from a Track

- Consistent method of interacting
  - “Pick, then Act”

- Once user knows how to perform one action on one type of object, she knows how to perform any action on any type of object
  - e.g. pick track1 and make it green
  - e.g. pick the entire detector and make it invisible
System Model

• **Purpose**
  – Need an overall system model to keep implementation consistent
  – User’s mental model should match the system model

• **Overview**
  – Entity
  – Model
  – View
System Model

- **Entity**
  - Physical object within the program
    - track

- **Model**
  - Abstract representation of the Entities
    - 2D model of track
    - hierarchical listing of tracks
  - One model can support several representations
    - path of particle through detector
    - momentum vector at production point
System Model (continued)

• View
  – Image based on a Model
    • curve showing x-y projection of particle’s path
  – Allows picking of any visible object
Design Schematic

framework components

GUI
CLI
USER INTERFACE

Info
Model

Hierarchy
Model

2D
Model

3D
Model

DataInput

Spectator

CLEO Libraries

OpenGL

OpenInventor
Design Overview

• **Spectator**: Holds event Objects (Detector, Response, etc.) and all Actions on those objects, and provides feedback to models when objects change.

• **DataInput**: Reads event info (detector geometry, response, etc.) and creates event objects which are stored in Spectator. Only module with CLEO specific code.

• **Models**: Implement methods to draw the event objects.

• **User Interface**: Provides user access to Actions in Spectator and new model views.
Prototype

• **Entities**
  – **Detector Components**
    • Composite, Tube, PolyCone, Polyhedron
  – **Reconstruction**
    • Track, Shower
  – **Monte Carlo**
    • Monte Carlo Particle

• **Models**
  – **2D Projection Model**
  – Hierarchical List
  – Information (e.g. Table)
Prototype (continued)

• Actions
  – Make Entity visible or invisible
  – Change color of Entity

• Help System
  – Web based
  – Uses standard web browsers
Third Party Tools

- X Designer: GUI builder
- CLHEP: matrices and vectors
- OpenGL: 2D graphics
  - MESA: OpenGL API for X windows
- XrT: Motif widget set
  - Table: nice widget for making tables
  - Gear: pop up help, hierarchy, tabs, etc.
- Xpm: icons
- FWF: hierarchy widget (slow to initialize)
- Netscape and Mosaic: used to read help pages
Prototype: Hierarchy View

- Displays a Category as a hierarchical list
- User decides how deep the hierarchy should be displayed
- Can pick
  - single Entity
  - Entity and its children
Prototype: 2D View

Projects 3D object onto front side top

Cursor can pick zoom center
(Cursor’s shape shows mode)
Prototype: 2D Representations

Path

Point

Vector

Labels: Can label Entity with any of its data
E.g.: showers: reference ID tracks: momentum magnitude
Prototype: Information View

Add/Remove columns
Choose column’s content
Sorts rows using sort ‘widget’
Shows pick status

<table>
<thead>
<tr>
<th>Entity</th>
<th>Category</th>
<th>View</th>
<th>Column</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track</td>
<td>Reconstruction</td>
<td>Information View</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>momentum magnitude</td>
<td>track1 1.89604</td>
</tr>
<tr>
<td>charge</td>
<td>track3 0.285967</td>
</tr>
<tr>
<td>charge</td>
<td>track4 0.228552</td>
</tr>
<tr>
<td>charge</td>
<td>track5 0.180275</td>
</tr>
<tr>
<td>charge</td>
<td>track2 0.884364</td>
</tr>
<tr>
<td>charge</td>
<td>track6 0.748271</td>
</tr>
<tr>
<td>charge</td>
<td>track9 0.315218</td>
</tr>
</tbody>
</table>
Prototype: Picking

Pick object in one View, highlighted in all Views
Prototype: Action Feedback

No Entity Picked:
Apply button not active

Entity Picked:
show property of Entity
Apply button active
Prototype: Help System

- All windows have a help button or menu item
- When user asks for help
  - starts a web browser if none are running
  - sets browser to appropriate help page

Help Preference Window

Choose browser

Read web pages
- from Florida over the internet
- locally
Usability Testing

• Watch physicists try to use the prototype
  – Start by letting them explore
  – Ask them to perform simple actions
  – Constantly ask them how they think the system works to determine their conceptual model

• Lessons we have learned
  – Users are constrained by what they have done before
    • E.g. old CLEO event display had one graphics window. No one opens extra windows in prototype.
  – Users seldom explore menu items
    • Exception: Users look in File menu to open new files
  – Users seldom use help
Use

• The core of Cleo3D (Spectator) is used in several different places
  – stand alone event display (Cleo3D)
  – event display that works in the CLEO III reconstruction/analysis framework
  – geometry viewer for recon/analysis framework
  – in the near future a stand alone geometry constructor
Use: Reconstruction/Analysis

- CLEO III reconstruction/analysis job
  - data sources write to a data container (Frame)
  - sequence of independent algorithms (Processors) which read from/write to a Frame
  - data sinks read from a Frame

- Cleo3D is a Processor

- Example: reconstruction
  - use Tracker and Cleo3D
  - step through events and view tracks
  - modify tracking parameters and reprocess an event

- Example: analysis
  - use analysis Processor and Cleo3D
  - Cleo3D displays decay chain
Conclusion

• **Present**
  - User Centric Design is a worthwhile approach
  - Several iterations of design and user scrutiny
  - Developed prototype that is useful for CLEO II

• **Future**
  - 3D Views
  - Refine the 2D View
  - Hits
  - Command line interface with scripting
  - Allow user to navigate links between objects, 
    e.g. select hits on a track
  - Advanced selection capability
    e.g. pick all tracks with momentum < 1 GeV