

Extending the Web3D: Design of Conventional GUI Libraries in X3D

Ivan Sopin

Network-Enabled WorkSpaces Laboratory
Armstrong Atlantic State University

Web3D 2010

Outline

- Introduction
- Related Work
- Design
- Implementation
- Demo
- Future Work

Outline

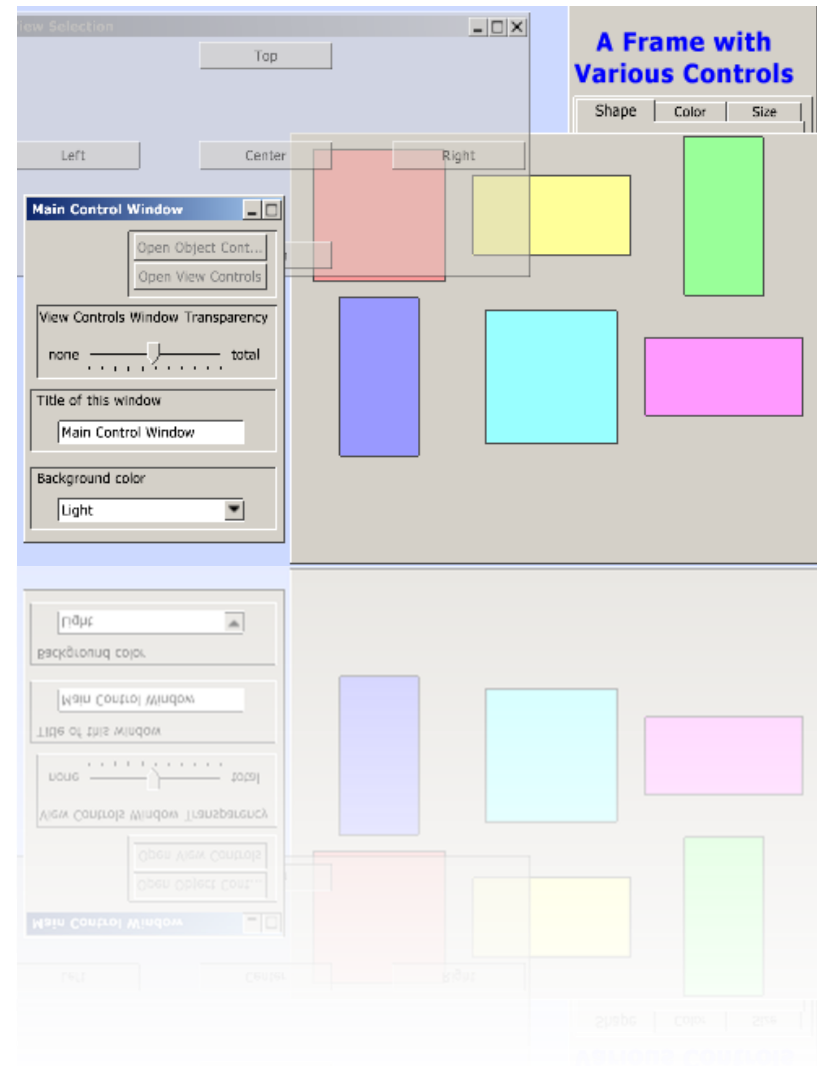
- **Introduction**
- Related Work
- Design
- Implementation
- Demo
- Future Work

Introduction: Motivation

- Web3D technologies, and X3D in particular, are becoming popular.
- X3D has no tools for creating conventional user interfaces (UIs).
- Content authors are responsible for their own UI implementations.
- Ad-hoc solutions are usually very limited and limiting.

Introduction: Proposal

- Library for creating functional and visually appealing interfaces: X3DUI
- Traditional MS-Windows-like UI elements
- GUI design very similar to Java or Visual C++



Outline

- Introduction
- **Related Work**
- Design
- Implementation
- Demo
- Future Work

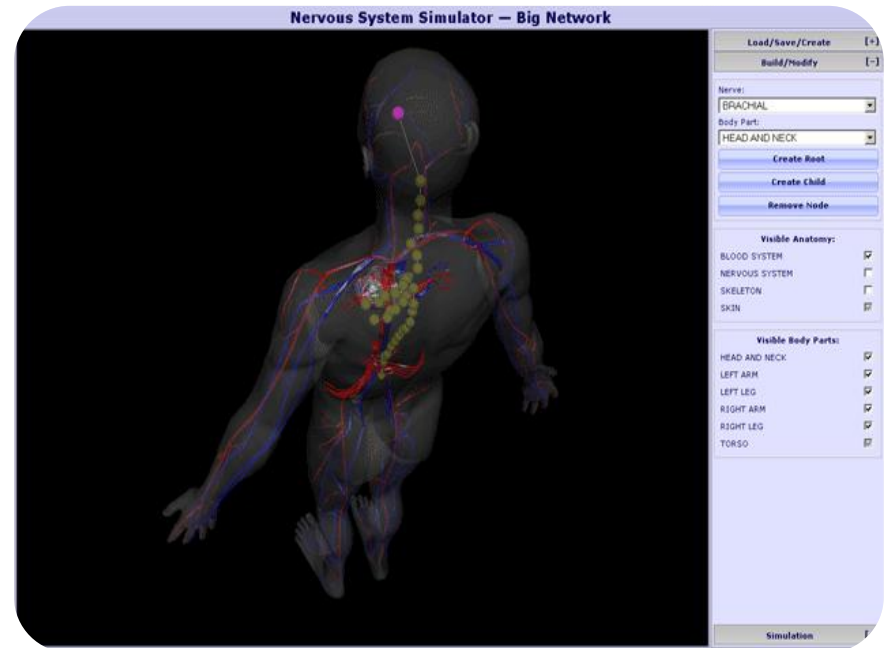
Related Work: X3D Interfaces

- Traditional way: interconnect geometric nodes via scripting
- Usually not so functional and not so appealing
- A single button might be hundreds of lines of code.
- Common problems:
 - inability to respond dynamically
 - poor rendering quality
 - unnatural integration of 2D and 3D



Related Work: HTML Interfaces

- X3D plug-ins provide APIs for runtime access to the scene via JavaScript.
- Advantages:
 - interactivity
 - accessibility
 - compatibility with Web technologies
- Disadvantages:
 - browser-only
 - high computation costs
 - delays and jitter
 - a subset of traditional GUI
 - no stereo mode
 - nothing above 3D scene



Related Work: Other Approaches

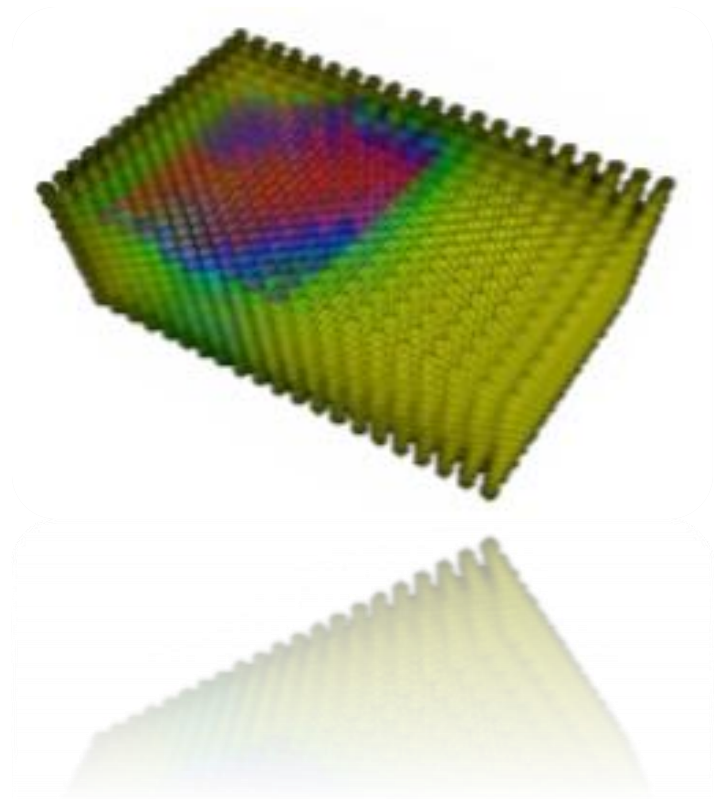
- New UI definition language to universalize the interface generation
- Architecture requires a special interpreter.
- Code is usually precompiled.
- Integration with other technologies is difficult.

Outline

- Introduction
- Related Work
- **Design**
- Implementation
- Demo
- Future Work

Design: Dimensionality

- 2D nature of content delivery has remained more or less unchanged.
- Information arranged in 2D is distributed and interpreted more easily.
- 2D or 2.5D implementations are often more suitable for GUIs.

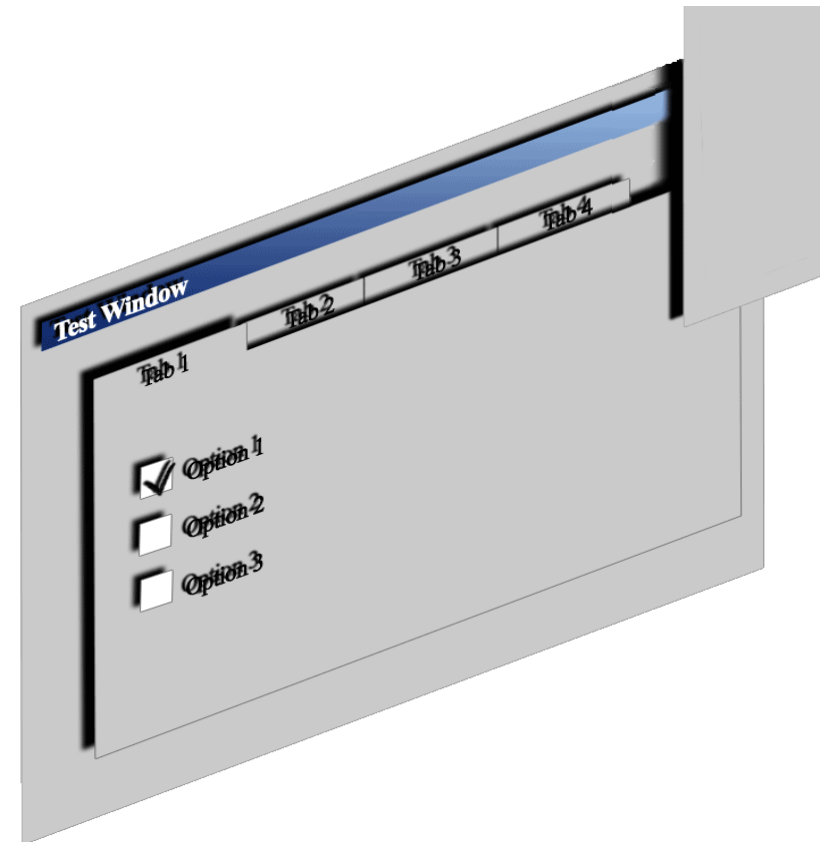


Design: Presentation in X3D

- Incorporation of 2D and 3D is non-trivial.
- For HUD-like behavior routing of a `ProximitySensor` to a `Transform` node is used.
- Issues of the traditional approach:
 - the geometry might flicker or “shake”
 - HUD-objects are penetratable
 - stereo mode is not supported
- `Layer3D` node does not have the described problems.

Design: Presentation in X3D (cont.)

- Management of 2D layers in the shared z-plane is challenging.
- Surface interpenetration is avoidable with separation along the z-axis.
- 2D items are rendered in the order they are displayed.



Design: Presentation in X3D (cont.)

- Visual GUI layering is done with `OrderedGroup` extension node.
- Depth-separation technique due to issue with sensors
- To avoid occlusion, transparency could be combined with “hideable” or closeable windows.
- Text-driven interfaces are informative and at times essential.
- `USE_TEXTURE` flag helps avoid tessellation-related problems

Outline

- Introduction
- Related Work
- Design
- **Implementation**
- Demo
- Future Work

Implementation: Structure of X3DUI

- 27 prototypes in 4 categories: *system*, *visual*, *group*, and *layout*
- The *system* category has components that organize the functioning of all other widgets.
- All prototypes with visual representation are in the *visual* category.
- The *group* category holds prototypes that manage behavior of several nodes in one group.
- Layout prototypes are combined in the *layout* group.

Implementation: Core Components

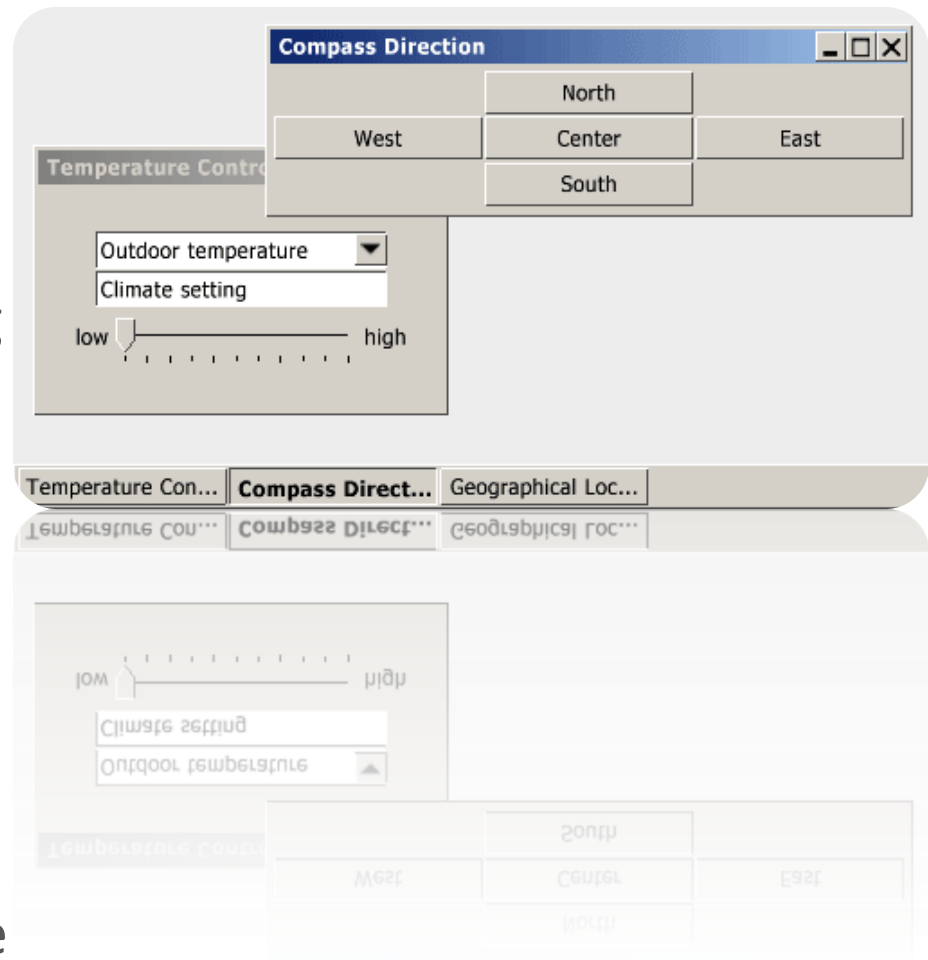
- Core components are `Display` and `Settings`.
- `Display` manages the operation of the entire interface.
- `Display` is a singleton, and all interface elements are its children.
- `Settings` contains configurations that define the “look-and-feel.”
- Recursive invocation of overridden functions
- Customized themes

Implementation: Visual Components

- `TextButton` is a rectangular button with a text label.
- `TextToggleButton` is a `ToggleButton` with text.
- `ControlButton` is accessory to the `Frame` prototype; four types: `MINIMIZE`, `MAXIMIZE`, `NORMALIZE`, and `CLOSE`.
- `CheckBox` and `RadioButton` for choice selection
- `RadioButtonGroup` for single-choice selection

Implementation: Visual Components (cont.)

- `Label` prototype for in-scene text management
- `TextField` generates a rectangular field for viewing and editing a string of characters.
- `ComboBox` is a drop-down choice list.
- `HorizontalSlider` for selecting continuous or discrete values from a range



Implementation: Container Components

- `Panel` is a basic layout-enabled grouping container.
- `TabPanel` displays the content of the active tab and labeled headers of inactive tabs.
- `Frame` is a complex and multifunctional component:
 - header with the title and a set of control buttons
 - may be floating or docked
 - may be resizable or static
- `TaskBar` is important in virtual desktop environments.

Implementation: Layouts

- Absolute positioning becomes extinct in the modern GUIs.
- X3DUI supports four layouts: `BorderLayout`, `BoxLayout`, `GridLayout`, and `FlowLayout`.
- The actual arrangement of components is performed by `LayoutManager`.
- `BorderLayout` places children in up to five areas: NORTH, SOUTH, WEST, EAST, and CENTER; the unused space is allotted to the CENTER area.
- `BoxLayout` positions components in a single row or column.
- `GridLayout` allocates components to individual cells of a grid with the specified number of rows and columns.
- `FlowLayout` is the default layout; it puts elements in rows on space availability basis.

Implementation: Deployment

- Prototypes stored in individual files, in one of the four directories.
- Additional directory with graphical files
- Total size of graphics is <4 KB; the source code measures ~430 KB.
- Automated tool generates a single minified file of ~280 KB.
- Loading time is shorter because of fewer file-system requests.
- Image folder location is an attribute in `Display` prototype.

Outline

- Introduction
- Related Work
- Design
- Implementation
- **Demo**
- Future Work

Outline

- Introduction
- Related Work
- Design
- Implementation
- Demo
- **Future Work**

Future Work

- Reduction of recurring code patterns and development of small task-specific prototypes
- Adjustment of the logic to recognize other X3D players and serve only player-supported content
- Implementation of new widgets: text area control, toolbars, file and context menus, dialogs, vertical and horizontal scrolls, lists, icons, etc.
- Build an X3DUI development suite to translate GUI written in an object-oriented language into X3D code?

Contact Information

Please email your questions and comments to

ivansopin@gmail.com

For more information on NEWS Laboratory, visit

<http://news.felixlup.info>

Questions?