**Background: What is a Tangible User Interface (TUI)?**

TUIs use physical objects to represent digital data. They often take the form of objects with sensors that can be placed and moved on a surface. Certain movements, defined by the programmer, cause actions to be performed on this data. These systems aim to break away from the traditional computer screen and bring reality based interaction into interfaces. For example, a tagged glass could be placed in a specified area on a restaurant table to automatically send a message for a drink refill to your waiter or waitress.

**Example TUI: Urban Planning**

Wind Tool: Place on surface; point in the direction of wind. Projector above surface then draws wind flow patterns. This tool would make no sense with no buildings placed, so at least one building must be placed for it to work.

Sun Tool: Place on surface; move clock arms to set a time. Projector above surface then updates shadows of buildings to reflect this time.

Buildings: Place on surface. The surface detects them and wind, shadows and any other effects are drawn by a projector above the surface.

**Project Aims**

Currently, TUI systems are difficult to build; each developer has their own method. The design process would be greatly enhanced if there was an easy way to specify, test, and prototype them. Thus our aim is to create a software solution for designing TUIs. This was made possible by the creation of a new language, **TUIML** (TUI Model Language). TUIML was created here at Tufts. It is a visual language, much like UML. A combination of **three main diagrams** completely specify any TUI, listed below.

1. **TAC Diagram**
2. **Dialogue Diagram**
3. **Interaction Diagram**

**Why are TUIs Important?**

TUIs allow for not only human-computer interaction, but human-human interaction. On left, the **Reactable** TUI is a musical instrument, but can be considered a stage show and is fun to use, especially with multiple people.

TUIs are the future of interfaces and facilitate a new way of thinking.

**A Quick Lesson On TUIML**

**TAC: Tokens and Constraints**

**Tokens:** Physical objects that represent digital data.

**Constraints:** Physical objects that provide context to tokens manipulation by constraining the ways in which tokens can be manipulated.

**Example TAC Diagram:**

**Wind Tool**

**Building Model:** The house itself represents a building. The "n" represents multiplicity, here meaning 1 to n buildings must be on the surface for this relationship to hold. The circle means it represents digital data.

Many of these TAC relationships exist in a system, thus all TACs are combined into a **TAC palette**. An example is shown to the left.

**Dialogue Diagram**

**Surface:** The rectangle represents the surface, implying that both buildings and the wind tool must be placed on the surface to function.

**Wind Tool:** The star represents the wind tool constraint. It has an implied multiplicity of 1, meaning only 1 can be on the surface at a time. The center square means the wind tool is a computational function.

**Interaction Diagram**

The rectangle represents transitions. Various symbols are used for different conditions.

- Rounded rectangles represent TACs.
- Arrows leaving the transition are preconditions needed for the transition.
- Arrows exiting transition are post-conditions resulting from the transition.
- Hexagons represent continuous manipulations, also making use of bi-directional arrows.
- Ellipses represent digital data.

- The recycler symbol indicates the diagram returns to its initial state.

**Our Work: TUIML Visual Editor**

The software is currently being developed as a Plug-In for Eclipse, a popular IDE. This gives the advantage of being cross platform, as well as being simple to distribute and use.

The ability to easily create all three main diagrams is the key focus of the software. At right is a concept screen shot, showing a Dialog Diagram. Users simply pick a shape to draw from a palette and then click and drag to draw it. Double clicking existing items will allow the user to edit their names and attributes.

**Future Work**

**Future ideas:**

- The ability to freely sketch the diagrams, instead of selecting from a palette.
- The ability to prototype; creating a virtual testing ground for the fully specified TUI.
- TUIML also has built in support for XML, thus the diagrams could be saved as XML, making it possible to edit them outside of the software.

There is much room for addition onto this project, as it is not meant to be a complete, commercial solution. It is hoped that it will complement the newly created language (TUIML) and propel it forward as the standard language of TUIs.

**References:**


TUIML Diagrams and TUI pictures provided courtesy of Orit Shaer.