Novel Widgets and Toolkits For Creating Them

- Toolkits for novel interaction techniques.
  - SwingStates
  - subArctic

- Toolkit for groupware applications.
  - MAUI

- All of these toolkits are Java libraries.
SwingStates

- SwingStates is a library that adds state machines to the Java Swing user interface toolkit.
- The state machines are used as a control structure for programming interaction.
  - Goal is to allow for programming advanced interaction and creating novel interaction techniques.

• SwingStates includes a Canvas widget.
  – Similar to the Tk canvas widget – GmlCanvas.
  – Holds a display list of shapes, including simple or arbitrary paths, text strings, images and some Swing widgets.
  – Each shape can have a geometric transform, a parent shape and a clipping shape.
  – Shapes can be tagged (similar to Tk).
State Machines

• A state machine consists of the following:
  – A set of states and a set of transitions labeled with events.
  – Each transition may have an associated guard and action.
  – Each state has a set of output transitions and a set of input transitions (may have an enter and leave action).
```java
StateMachine sm = new StateMachine() {
    SShape dragged = null;
    public State start = new State() {
        Transition dragOn =
            new PressOnShape(BUTTON1, "drag") {
                public void action() {
                    dragged = getShape();
                }
            };
    }
    public State drag = new State() {
        Transition drag = new Drag(BUTTON1, "drag") {
            public void action() {
                move(dragged);
            }
        };
        Transition dragOff =
            new Release(BUTTON1, "start") { };
    }
};
```
Attaching State Machines to UI Objects

• Attach directly to regular Swing widgets to extend or redefine their behavior.

• Attach directly to the canvas and to individual shapes on the canvas.

• Via Tags
  
  – *Extensional tags* – added to or removed from objects explicitly.
  
  – *Intentional tags* – specified using a predicate.
    
    • Example: all objects with a blue background
Extending Swing Button Behavior

- A crossing interface for Swing Buttons

```java
JStateMachine cross = new JStateMachine() {
    public State out = new State() {
        Transition enter = new EnterOnTag("javax.swing.JButton", "in") {
        }
    }
    public State in = new State() {
        Transition leave = new Leave("out") {
            public void action() {
                ((JButton)getComponent()).doClick();
            }
        }
    }
    // attach this state machine to the quit button of Fig. 2
    cross.attachJComponent(quitButton);
};
```

Applet
An Example Using Multiple Features

- Pie Menu example using the following elements
  - Canvas
  - Glasspane
  - Two state machines (communicating with each other)
  - Multiple widgets with tags

Applet
// ColorTag are designed to be added to each menu item.
1 class ColorTag extends CExtensionalTag {
2    Color color;
3    ... // constructor
4 }

// Color events are sent by smMenu to smWidget.
5 class ColorEvent extends VirtualEvent {
6    Color color;
7    ... // constructor
8 }

// The state machine that manages the pie menu
9 smMenu = new CStateMachine(canvas) {
10    public State menuOff = new State() {
11        Transition show = new Press(BUTTON1,"menuOn"){
12            public void action() {
13                showMenu(getPoint());
14             }
15         }
16    }
17    public State menuOn = new State() {
18        Transition command = new ReleaseOnTag
19            (ColorTag.class, BUTTON1, "menuOff") {
20            public void action() {
21                Color c = ((ColorTag)getTag()).color;
22                smWidgets.processEvent(new ColorEvent(c));
23                hideMenu();
24            }
25        }
26        Transition hide = new Release
27            (BUTTON1, "menuOff") {
28            public void action() {
29                hideMenu();
30            }
31        }
32    }
33    }
34
35    // "colorable" widgets must be attached to this state machine
36    smWidgets = new JStateMachine() {
37        JComponent picked;
38        public State noSelection = new State() {
39            Transition select = new PressOnComponent() {
40                public void action() {picked = getComponent();}
41            }
42            public State selection = new State() {
43                Transition deselect
44                    = new Event("cancel", "noSelection") {};
45                Transition color
46                    = new Event("color", "noSelection") {
47                    public void action() {
48                        ColorEvent e = (ColorEvent)getVirtualEvent();
49                        if (selected) picked.setBackground(e.color);
50                    }
51                }
52            }
53        }
54    };
Why use SwingStates?

- SwingStates lets you design and implement the interaction technique separate from the UI object.
- Once you have a state machine it can be attached to UI components in flexible ways.
- The canvas and tags borrowed from Tk make some of the drawing in Swing easier.
The subArctic toolkit was designed to “supply a powerful library of reusable interactive objects, and to make it easy to create new, unusual, and highly customized interactions”.

- Provides standard reusable components.
- Allows input to be handled in flexible ways.

http://www.cc.gatech.edu/gvu/ui/sub_arctic/
Dispatch Agents

- Each agent handles a certain type of event.
  - Text editing input
  - Various types of dragging
  - Pressing, clicking, or double-clicking

- A simple drag agent would implement the following state controller.
Dispatch Agents (cont.)

- Dispatch agents use an input protocol to communicate with components.
- The component would then have to implement the Java interface that defines the protocol.
- Example of a simple drag agent protocol.

```java
public interface simple_draggable extends focusable {
    public boolean drag_start(...);
    public boolean drag_feedback(...);
    public boolean drag_end(...);
}
```
Picking

- SubArctic gives the programmer control over what gets picked under the cursor position.
- Provides an explicit list (*pick collector*) and lets the components decide how to fill in the list.
- Picking is performed by a top-down recursive traversal of the component tree.
A shadow drag container example for modifying picking order.

- Any widget placed in the container will be dragged if any one of the widgets in the container is picked.
- The container's pick() method creates an empty pick collector and passes it to the child objects.
- When the container gets the list back it adds itself to the original pick collector followed by all the children in the list.
Draggind Dispatch Agents
Provided by subArctic

- Conventional Dragging
- Constrained Motion
  - Uses a drag filter to limit the motion to an area.
- In/out dragging
  - Similar to the crossing interface seen in SwingStates example
- Semantic snap-dragging
  - Selected targets act as gravity wells for object being dragged. The target must pass a semantic test before the snap to target occurs.
Other subArctic Features

- Recording input
  - Ability to record input events to see the stream of input produced by the user.
    - Low level recording of user input.
  - Ability to record the input protocol and the method within the protocol used along with the object it is sent to.
    - Recording the semantics of user input.
- Animation support allowing for various effects.
  - Slow-in/slow-out motion
  - Squash and stretch
The MAUI Toolkit

- Multi-user Awareness UI toolkit
- Goal is to help synchronous and distributed collaborators maintain awareness of other people in the group.
- Extends many Java Swing widgets and adds some groupware specific widgets to provide feedthrough.

- http://hci.usask.ca/research/past/maui.shtml
MAUI Toolkit Features

- Supports both single-state and multi-state versions of most Java Swing widgets.
- Provides run-time customization that can be controlled either by the user or the application.
- Includes black-box network and session components that enable prototyping and testing.
- Packages components as Java Beans, which allows integration with standard IDEs.
Extended Java Swing Components

GButtons

- GButtons only have shared state form
GMenus have only shared version
- Transparent Representation
- Summary Representation
Extended Java Swing Components

GComboBoxes

- GComboBoxes have only shared versions
  - Transparent Representation
  - Summary Representation
Extended Java Swing Components

GSliders

- GSliders have two versions
  - Multi-User
  - Shared
GTextFields use a combination of shared and multi-user properties.

- Text is shared
- Selections in text are multi-user
Extended Java Swing Components
GSscrollPanes

- GSscrollPanes have two versions
  - Multi-User
  - Shared
Extended Java Swing Components

GTabbedPanes

- GTabbedPanes have two versions
  - Multi-User
  - Shared
Extended Java Swing Components
GLists

- GLists have two versions
  - Multi-User
  - Shared
MAUI Groupware Widgets

Telepointers

- Participant names can be added to pointer.
- Telepointers can leave fading trails.
  - Helps provide gesture information
- Three styles available
  - Arrow
  - Block
  - Icon
MAUI Groupware Widgets
Participant List

- Shows the names and colors of all connected users.
- Does not listen to standard AWT events. Only listens for connection and disconnection events.
- Can be either top-level dialog or a panel inside a frame.
MAUI Event Model

- Captures incoming AWT events from the local user.
- Generates MAUI-specific awareness events for distribution.
- Handles awareness events that have arrived from another machine.
Capturing AWT and Swing User Events

- MAUI components include two types of listeners.
  - ActionListener – a widget specific listener to handle action events.
  - BasicAwarenessAdapter – a MAUI adapter to handle intention events.
Creating and Distributing Awareness Events

- A MAUI event called a GControlEvent is created to send awareness information.
- The GControlEvent extracts information from original AWT event making it smaller and more suitable for sending over the network.
Handling Awareness Events at the Remote Component

- Components receive remote events from their GController by implementing the GControlListener interface.
MAUI Toolkit in the JBuilder IDE
Customization Dialogs

Figure 7. Customization dialogs for a GFrame (left, design-time) and for a GButton (right, run-time).
MAUI Limitations

- MAUI uses the glass pane so developers cannot use their own glass pane.
- Applications are limited to using the frame class as main window.
- Only TCP-based communication is provided and late entrants are not updated.
- Messages are processed on a first-come first-served basis.
References

