21 September 1995

Andrew Consortium
School of Computer Science
Carnegie Mellon

Extensible Widgets for Andrew

Fred Hansen
Director, Andrew Consortium

Sneaking up on Widgets

Every issue of The Andrew View
has mentioned widgets
The need for
Plans to implement
WidgetMentionsInNewsletter

The time has come

Taught Software Engineering this past winter
Helped my thinking

Now are implementing
Widget Driver
described in latest AndrewView
Andrew View, Summer 1995

Initial Widgets

PushButton - pushbutton
ToggleButton - on/off
RadioButton - one of several is on
TextResponse - a small text entry area
Label - simple text label button
List - choose from list of strings; perhaps with scrollbar

Design

Widgets described in widget resource files

Executable code
- in C++
- in an extension language
  - Ness
  - HotJava
  - Rexx
  - tcl

Image is a figure
- Code may modify it

Naming:
- classes start with 'A'
  - capitalized words
- file names start with 'a'
  - all lower-case

Resources give Attribute values
Attribute values stored in Slots

Files in X **resource** format
    Not yet dependent on X

Each resource defines an **attribute** for a widget

Attribute values reside in **slots**
    attached to an AWidget dataobject
    (or a derived class)

ASlot objects store values

Slots are implemented as ASlot objects

ASlot is a base class

ASlotInt, ASlotReal, ASlotFunction, ASlotPtr, ASlotString, ASlotATK
soon: ASlotDataObj and ASlotFigure

A typical .H file will declare an attribute thus:

something.H

```
class something : AWidget {
    . . .
    ASlotInt height;
    . . .
}
```

Dataobject handles its slots

Give a slot a name

something.C

```
something::something() {
    . . .
    Baptize(height, "height");
    . . .
}
```

Initialize slot from
resource file
datastream

```
something::Read (...) {
    . . .
    // get widget class name from stream
    FetchInitialResources(widgetclassname);
    . . .
    // may later find a value for
'height'
    //  in the data stream itself
}
```

Slots behave like C++ values

other.C:

```
AWidget *obj = . . .;
int size = obj->height - 3;
obj->height = 2 * size;
```

OR if 'height' is private

something.H:

```
inline int GetHeight()
    { return height;}
inline void SetHeight(int v)
    { height = v; }
```

other.C:

```
AWidget *obj = . . .;
int size = obj->GetHeight() - 3;
obj->SetHeight(2 * size);
```

Adding a slot

Add flameColor slot to an object

other.C:

```
{
    something *s = ...;
    . . .
    slotname(flameColor);  //macro makes
            //  flameColor into a
slotname
    . . .
    s->Set(flameColor, "bright red");
        // Set adds the slot if absent
    . . .
    char *fcol = s->Get(flameColor);
}
```

AWidget dataobject

Dataobject plus

Stores arrays of ASlots
local (via Baptize)
added (via Set)

Reads attribute values from resource files

Supports access to attributes via extension language

AWidgetView

AWidgetView is a view
gets parameters from predefined slots in AWidget

E. g., AWidgetView::DesiredSize returns
desiredWidth and desiredHeight

User input calls functions, if defined
mouse{Left,Right}{Down,Move,Up}

AWidgetView image

Widget image
    screenImage slot is a figure

AWidgetView::FullUpdate
    first calls fullUpdate slot, if any
    then fetches and displays screenImage

AWidgetView::Update
    first calls update slot, if any
    then fetches screenImage
        redraws if replaced
        calls its Update otherwise

Summary

General widget mechanism

Focus for providing common functionality

Easier implementation

Accessible from both C++ and extension languages

Later, a widget construction editor