Garnet and Amulet

Brad A. Myers

Human-Computer Interaction Institute

Carnegie Mellon University

Pittsburgh, PA 15213-3891

bam@cs.cmu.edu

Creating User Interface Software

Garnet

and

Amulet

Carnegie Mellon

Objectives

- Provide programming and interactive, demonstrational tools that support gestures, speech, 3D, animation, multiple-users, visualization, direct manipulation, and machine independence.
- Flexibility so useful to UI researchers, easy to use for students, and performance for general use.

Key Ideas

- Interactive tools increase productivity by factors of 10 to 50.
- Most of the user interface can be demonstrated by giving examples of desired graphics and behaviors.
- Knowledge is used to guide the design
- Flexible object & constraint system built in C++.

Milestones

- Initial release on Unix with full widgets
- Interactive tools for creating interfaces
- Port to Macintosh and MS Windows
- Support 3D, multiple views, multiple users
- Support visualization, maps
- Teaching version with simple interface

Garnet work

- **Macintosh Port (V 3.0) almost released**
- **Talisman for CSCW**

  Toolkit and Application Layer Interface for
  Synchronous Multiple-user Access over a Network

  - **MS Thesis of Nobuhisa Yoda**
  - **Single central machine drives multiple displays**
  - **Provides high level abstractions for floor control and for mapping abstract data to multiple views**
- **SILK**

  Sketching Interfaces Like Krazy

  - **PhD Thesis of James Landay, to be proposed Oct 28.**
  - **Interactive tool for graphic designers that allows initial designs to be sketched**
  - **Can interact with sketch: move slides, press buttons**
  - **Will transform design into actual widgets**
  - **Support for history, annotations, multiple versions, etc.**

Garnet work, cont.

- **Katie**
  - **PhD Thesis of Dave Kosbie; almost complete**
  - **Architecture for hierarchical events**
    - » **In addition to recording low-level mouse and keyboard events, records higher level events like “Delete Object” or “Change to Bold”.**
  - **Supports sophisticated Undo and programming-by-demonstration**

Amulet Overview

- **Decided to rename the C++ version since is different.**
- **Name is tentative**
- **Tentative acronym:**

  Automatic Manufacture of Usable and Learnable Editors and Toolkits
- **Design Status**
  - **Started work over the summer.**
  - **Have the object, constraint, and low level graphics layers finished for Unix.**
  - **Supports multiple C++ compilers**
  - **MasterMind already evaluating it**

Amulet Schedule

Oct'94	Initial release of object system to ISI/ GATech
Jan'95	Initial release of full toolkit for Unix, including widgets
Mar'95	Sophisticated multi-font text widget
Jun'95	Macintosh and Windows versions of toolkit
Sep'95	Teaching version and interface builder, support for CSCW
Jan'96	Support for multiple views, visualization, maps
Jun'96	Support for 3D, interactive tools for 3D
Sep'96	End-user customization tools; Demonstrational tools for visualization, maps, CSCW
Jun'97	Multiple constraint solvers
Sep'97	Demonstrational tools for end-user customization

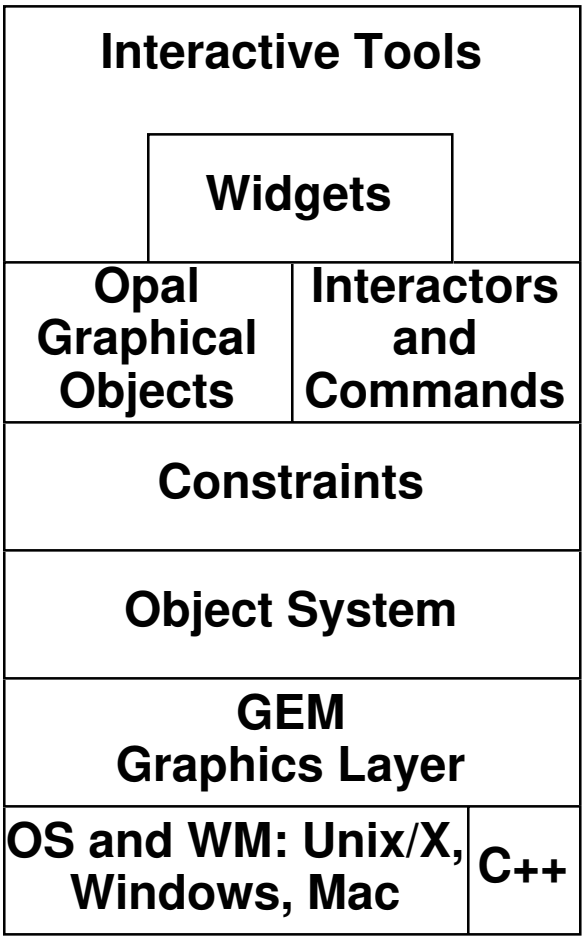
Goals

- **For researchers**
  - **Flexible constraint system** *** NEW ***
  - **Easy to explore new interactive techniques**
    - » **Speech recognition** *** NEW *** **, Gestures, etc.**
  - **Complete control over widgets:**
    - » **Requires reimplementing them**
    - » **No annoying layout rules**
    - » **Can build own widgets (Shneiderman)**
    - » **Can be made flexible for multiple users:**
      - **Multiple cursors in text widget**
      - **Enabling and disabling of widgets based on floor control for multiple users**
- **For students** *** NEW ***
  - **Easy to learn; useful for a sophomore Software Engineering course**
  - **Simple things are simple and complex things are possible**
- **For general developers**
  - **Sufficient performance, robustness and documentation**

Initial Target Applications

- **MasterMind**
- **Our own interactive tools**
- **WWW (Mosaic) Interface (WYSIWYG html editing)**
- **Multimedia**

Architecture Overview



Gem Level

- **Abstract interface to Window Manager graphics**
  - **All code written using Gem should port without change**
  - **Most programmers will not use Gem layer**
- **Supplies C++ objects for "Drawonable" (window), drawing styles, fonts, etc.**
- **Not a retained object layer: standard commands to draw on the window**

```
#include <Amulet/gem.h>
main ()
{
Am_Style *red = Am_Style::Create (1.0, 0.0, 0.0);  // r, g, b
Am_Style *green =  Am_Style::Create (0.0, 1.0, 0.0);
Am_Drawonable *root =
   Am_Drawonable::Get_Root_Drawonable();
Am_Drawonable *d = root->Create (800, 10, 300, 480); //
   new window

d->Draw_Rectangle(red, NULL, 10, 320, 100, 50); // red
   outline, no fill
d->Draw_Line(green, 20, 30, 100, 200);
d->Flush_Output ();
}
```

Gem Level, input handling

- **Object member functions for the various events**
- **Events parsed from strings like “LEFT_UP”, “CONTROL_A”, “DOUBLE_MIDDLE_DOWN”, or “META_SHIFT_F3”**

```
class Am_Input_Event_Handlers {
public:
    virtual void Map_Notify (Am_Drawonable *draw) = 0;
    virtual void Unmap_Notify (Am_Drawonable *draw) = 0;
    virtual void Reparent_Notify (Am_Drawonable *draw) = 0;
    virtual void Destroy_Notify (Am_Drawonable *draw) = 0;
    virtual void Configure_Notify (Am_Drawonable *draw,
        int left, int top, int width, int height) = 0;
    virtual void Exposure_Notify (Am_Drawonable *draw,
        int count, int left, int top, int width, int height) = 0;
     // next one used for keys, mouse buttons, mouse moved,
    // and enter-leave.
    virtual void Input_Event_Notify (Am_Drawonable *draw,
                                     Am_Input_Event *ev) = 0;
};
```

Object Level

- **Prototype-Instance Object System implemented in C++**
- **Slots (member variables) dynamically created & typed**
- **Full querying capabilities:**
  - **Names, types and contents of slots**
  - **Prototypes and instances**
- **Object layer also supports part-owner (aggregate or group and its parts) hierarchy**
- **Excellent preliminary performance numbers:**

on a Sun SPARC ELC:	Garnet v3.0 Allegro CL	Amulet C++	Ratio
Simple slot as	4.6 microsec	1.446	3.18
Inherited slot access	4.6	2.786	1.65
Valid formula	7.2	2.456	2.93

Constraint Level

- Eager, so more intuitive
- Can have multiple Amulet constraints on one slot
  - Allows a form of multi-way constraints, like Rendezvous
- Can have multiple constraint solvers
  - How they will interact is an open research issue
- Also, there is a flexible “demon” mechanism
  - trigger on slot set or slot value changed
  - used internally for graphics and constraint maintenance
  - also available to programmers

Syntax

- **No preprocessor now, so syntax a little verbose:**

```
#include <Amulet/formula.h>
#define SHADOW 502 // Pick an arbitrary number for the slot name

Define_Formula (int, ownerwidth)  // define a formula function named
                                  // owner width which returns an int
   {  return self->GV_Owner ()->GV (WIDTH)  -  10;  }

void main ()
{
  Am_Object* a = Am_Group->Create ()
   ->Set (LEFT, 0)
   ->Set (TOP, 5)
   ->Add_Part(SHADOW, Am_Rectangle->Create()
                    ->Set(LEFT, 10)
                    ->Set(TOP, 10)
                    ->Set(WIDTH, Am_Formula::Create (ownerwidth));

int value = d->Get (WIDTH);
Am_Main_Window->Add_Part(a);  //display it in the default window
Am_Main_Event_Loop();  //handle input events
}
```

Opal (Graphics) Level

- Graphical objects for the standard types:
  - Rectangle, Circle, Text, Polylines, etc.
- Hides Gem graphics layer
- Add objects to windows or groups
- Automatic refresh if change values
- Structural inheritance, so can change prototypes
- Soon will have multi-font, multi-line text

Interactors

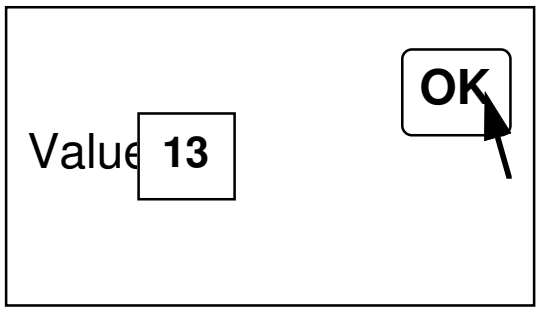
- Create Interactor objects to handle input
- Hides Gem input layer
- Different interactor types for different classes of input
  - select-object, move/grow-object, new-object, rotate-object, etc.
  - parameterized for flexibility
- Fewer parameters than Garnet version
- Instead, flexibility achieved with functions
- Final-function (result) is a Command Object

Command Objects

- **Instead of executing a callback, interactors and widgets create a "Command Object" and execute its "Do" method**
- **Slots of command object:**
  - **DO**
  - **UNDO**
  - **DO_AFTER_UNDO (redo)**
  - **SHORT_HELP**
  - **LONG_HELP**
  - **ACTIVE (enabled)**
  - **etc.**

Command Objects, cont.

- **Based on MacApp, but hiearchical like Katie**



— For example:

User Clicks in OK button

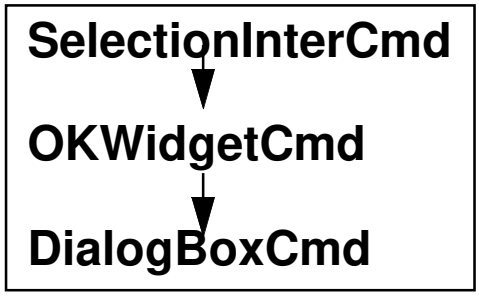
Selection-Interactor associated with that button

Selection-Interactor's command object DO method

OK Button widgets's command object DO method

Dialog box's command object DO method

History:



Command Objects, cont.

- **Advantages:**
  - **Undo**
  - **Help**
  - **Easy place to put constraints to determine enabled (greyed out)**
  - **Standard operations supplied in an easy way**
    - » **Cut, Copy, Paste, Create-Object, Delete-object, To-Top, etc.**
    - » **Easier to customize**
  - **Makes transcripting easier**
    - » **Like for Marquise**
    - » **End-User programming by demonstration**
    - » **For use by models**
    - » **Automatic analysis**
  - **Will help with CSCW**
    - » **Executing the command on multiple screens**

Future

- **Widgets**
  - **Will be written by creating new objects that call Gem, rather than as composites, for efficiency**
- **Interactive language**
  - **Needed for interactive tools, and to make development faster**
  - **Important issue: which one to choose, or create new one?**
  - **tcl language not generally liked**
  - **Probably Scheme, Python, Interpreted subset of C or C++**
- **Interactive tools**
  - **Interface builder will be early**
    - » **Will be designed with students in mind**
- **Support for Macintosh and Windows**
  - **Would prefer to do Macintosh first since have in-house expertise**
- **Support for advanced interaction techniques**
  - **3D, multiple users, multiple views, etc.**

Conclusions

- **Very excited about potential of new system.**
- **Bring the innovations and proven effectiveness of Garnet to a wider audience.**
- **Improvements based on experience.**
- **Extensions will open up new research avenues.**