

Building Interactive Objects



Approaches



- Multi-object Composition
- Within single objects
 - Inheritance
 - Aggregation
- Not mutually exclusive

Inheritance



- Reuse/specialize base classes
 - e.g. Swing

Aggregation



- Similar approach to inheritance, but different mechanism
 - combine parts handling different tasks into (logically) individual objects

- connect to act as one object/unit

Composition Example: InterViews



- C++ and X
- 3 categories of graphical objects
 - Interactors, structured graphics and text
- They do not share a common base class
 - special container objects to compose one inside the other

Interactor objects



- Base class:
- Composition class:

Structured Graphics



- Base class:
- Composition class:

Structured Text



- Base class:
- Composition class:

Interactor composition



- TeX boxes and glue
 - Tiled composition with glue between

Composed Hierarchy Layout



- Computed in two passes:
 - bottom up
 - top down

Example: Figs 4 & 5



- Two level composition
 - vbox:
 - hbox at top:
 - hbox at bottom:

Another Example: Trays



- One interactor as background
 - positions another interactor over it (Fig 9)

Simpler Interactors



Structured Graphics Composition



Aggregation



- Separation of concerns
- Can do this with
 - (multiple) inheritance
 - multiple objs aggregated into 1 logical unit
- Example of aggregation
 - Smalltalk Model-View-Controller (MVC)

Detailed Example: Garnet



- Similar goals to other modern toolkits
- Does it in a different fashion
 - the "interactor model"

Parameters to Interactors



- Output object(s)
- Start-where region
- Running-where region
- Start, stop, and abort event descriptions

More Parameters to Interactors



- Feedback objects

- Actions

(Local) Dialog Control



- One very general parameterized machine

Simplified State Machine



Types of Interactors



- Six within framework
- Differ: feedback and output object(s)
- Do standardized things to them

Six Interactors



- Menu
- Move-Grow
- New-Point
- Angle
- Text
- Trace