

CSE 341, Spring 2005

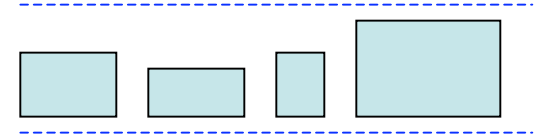
Lecture 27

More OO design

2 Pass Layout Algorithm

- Pass 1:

- walk along widget list adding up widths until you find that the next widget is too wide to fit



- while you're at it, find their max height

- Pass 2:

- display those widgets on baseline = previous baseline + max height

} ignored in later slides

“Classic” C

```
struct Widget {width, height, type, ...type-specific fields...}
Widget[1000] data;
x = 0; y = 0;
for(i = 0; i < n; i++) {
    if(x+data[i].width > linewidth) {x = 0; y += maxh; maxh = 0}
    switch ( data[i].type ) {
        1: drawstring(x,y,data[i].string,...); break;
        2: drawradio(x,y,data[i].radiolabels,...); break;
        3: ...
    }
    x += data[i].width; maxh = max(maxh,data[i].height)
}
```

Better C

Separate Layout & Drawing

```
struct Widget {width, height, type, ...type-specific fields...}
Widget[1000] data;
x = 0; y = 0;
for(i = 0; i < n; i++) {
    if(x+data[i].width > linewidth) {x = 0; y += maxh; maxh = 0}
    draw_widget(x,y,data[i]);
    x += data[i].width; maxh = max(maxh,data[i].height)
}
void draw_widget(x,y,theWidget) {
    switch ( theWidget.type ) {
        1: drawstring(x,y,theWidget.string,...); break;
        2: drawradio(x,y,theWidget.radiolabels,...); break;
        3: ...
    }
}
```

Java

```
abstract class Widget {
    int: width, height
    abstract void draw(x,y)
}
class strWidget extends Widget {
    String: string...
    void draw(x,y) {
        drawstr(x,y,string,...)
    }
}
class radioWidget extends Widget {
    String: radiolabels...
    void draw(x,y) {
        drawradio(x,y,radiolabels,...)
    }
}
...
```

```
class Page {
    Widget[] data; // initialized somehow
    void layout() {
        x = 0; y = 0; maxh = 0;
        for(i = 0; i < data.length; i++) {
            if(x+data[i].width > linewidth) {
                x = 0;
                y += maxh;
                maxh = 0
            }
            data[i].draw(x,y);
            x += data[i].width;
            maxh = max(maxh,data[i].height);
        }
    }
}
```

Smalltalk

- except for syntax, a lot like Java

Scheme

- A widget is stored as a list, with type, width, height, ... in fixed positions
- A page is a list of widgets
- Main algorithm is pretty similar to above, except recursion (or mapcar) used to iterate over list
- Code somewhat opaque since widget fields often accessed as “(caddr widgetlist)”, e.g.

ML

- A lot like Scheme, but ML data structs make field access more transparent
- Case/pattern match handles different widget types
- Iteration via “fold”, using another data struct to “accumulate” info about current x, y to decide whether next widget fits on a line

Pro/Con of OO design (here)

- *Algorithm* recognizably the same in all four languages, despite, e.g., loops vs recursion vs fold.
- + OO Localizes/groups/encapsulates info
 - + Main does layout alg, largely widget-independent
 - + Widget holds generic widget-essence
 - + Subclasses hold widget-specific stuff
- + OO probably better for code reuse/extension
- OO somewhat verbose
- Re Scheme: typelessness is a 2-edged sword, & lack of named data struct fields probably hurts (the “caddr” problem)

Change Orders

- Format control
 - Fonts, sizes, colors, ...
- Layout control
 - Justification, recursive subregions, tables...
- New widgets
 - Sliders, dials, pull-downs, .png, .jpg, ...
- Windows/Mac/Linux ports...

Solution 1: Duplicate Lots

```
abstract class MacWidget {
```

```
    MacWidget[] data;
```

```
    abstract class WinWidget {
```

```
        int: width, height
```

```
        abstract void draw(x,y)
```

```
    }
```

```
    class strWidget extends Widget {
```

```
        String: string...
```

```
        void draw(x,y) {
```

```
            Windrawstr(x,y,string,...)
```

```
    }}
```

```
    class radioWidget extends Widget {
```

```
        String: radiolabels...
```

```
        void draw(x,y) {
```

```
            Windrawradio(x,y,radiolabels,...)
```

```
    }}
```

```
    ...
```

```
    WinWidget[] data;
```

```
    x = 0; y = 0;
```

```
    for(i = 0; i < data.length; i++) {
```

```
        if(x+data[i].width > linewidth) {
```

```
            x = 0;
```

```
            y += maxh;
```

```
            maxh = 0
```

```
        }
```

```
        data[i].draw(x,y)
```

```
        x += data[i].width;
```

```
        maxh =
```

```
            max(maxh,data[i].height)
```

```
    }
```

Windows

Mac

Solution 2: Flags & Tests

Pass or set global platform flag and test it everywhere

```
abstract class Widget {  
    int: width, height  
    abstract void draw(x,y)  
}
```

```
class strWidget extends Widget {  
    String: string...  
    void draw(x,y) {  
        drawstr(x,y,string,...)  
    }  
}
```

```
class radioWidget extends Widget {  
    String: radiolabels...  
    void draw(x,y) {  
        drawradio(x,y,radiolabels,...)  
    }  
}
```

...

```
Widget[] data;  
x = 0; y = 0;  
for(i = 0; i < data.length; i++) {  
    if(x+data[i].width > linewidth) {
```

```
switch (platform) {  
    case microsoft: winDrawStr(...); break;  
    case mac: quartzDrawText(...); break;  
    case Xmotif: motifStrShow(...); break;  
}
```

```
        x += data[i].width;  
        maxh = data[i].height)
```

```
switch (platform) {  
    case microsoft: ...  
    case mac: ...;  
    case Xmotif: ...  
}
```

Solution 3: The Object Factory

```
abstract class FrameFactory {
  abstract Frame makeFrame()
}
class WinFrameFactory extends FrameFactory {
  Frame makeFrame() { return new WinFrame
} // ditto MacFrameFactory
abstract class Frame {
  Widget[] data;
  void add(Widget: aWidget) { data[i]=aWidget}
  abstract drawstr(x,y); ...
}
class WinFrame extends Frame {
  void drawstr(...) {winDrawStr(...)}
}
class MacFrame extends Frame {
  void drawstr(...) {quartzDrawText(...)}
```

```
abstract class Widget {
  int: width, height
  abstract void draw(x,y)
}
class strWidget extends Widget {
  String: string...
  void draw(x,y) { drawstr(x,y,string,...)
}
class radioWidget extends Widget {
  String: radiolabels...
  void draw(x,y) { drawradio(x,y,radiolabels,...)
}
...
```

```
switch (platform) {
  case msft: g = new WinFrameFactory;
    break
  case mac: g = new MacFrameFactory;
    break;
  ...
}
theFrame = g.makeFrame();
```

```
class Page {
  Widget[] data; // initialized somehow
  void layout() {
    x = 0; y = 0; maxh = 0;
    for(i = 0; i < data.length; i++) {
      if(x+data[i].width > linewidth) {
        x = 0;
        y += maxh;
        maxh = 0
      }
      data[i].draw(x,y);
      x += data[i].width;
      maxh = max(maxh,data[i].height);
    }
  }
}
```

Unchanged