UCI University of California, Irvine

Informatics 134

Software User Interfaces
Spring 2023

Mark S. Baldwin
baldwinm@ics.uci.edu

4/09/2024

Agenda

Upcoming

Upcoming

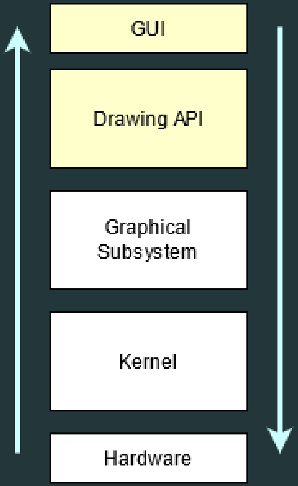
- Lecture today
- A0 Due tonight

Basic Structured Graphics

Basic Structured Graphics

Review the Requirements of a Graphical Program

- Manage what gets rendered to a screen
- Manage how it gets rendered
- Manage when it gets rendered



Basic Structured Graphics

Requirements of a Graphical Program

As programmers of graphical user interfaces, we primarily concern ourselves with *what* is rendered, rather than *how* or *when*.

Why?

Basic Structured Graphics

Requirements of a Graphical Program

The *how* and *when* are largely repeatable tasks that do not change across different user interfaces.

The *how* and *when* requirements, therefore, can be abstracted into reusable mechanisms that can support the *what* that programmers create.

This type of system is called ”Structured Graphics”

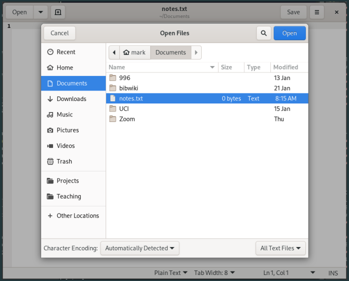
Basic Structured Graphics

Structured Graphics System

Encapsulate a primitive (rectangles, lines, images, icons, etc.)

Expose reusable code for rendering how and when.

Enable programmer to create the what (e.g., a button)



Name	Size	Type	Modified
996			13 Jan
bibwiki			21 Jan
notes.txt	0 bytes	Text	8:15 AM
UCI			15 Jan
Zoom			Thu

Character Encoding: Automatically Detected
All Text Files
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

Basic Structured Graphics

Advantages of Structured Graphics

- Less code, more reusable
- Encapsulation of common mechanisms enables automation of required actions like redraw and refresh
- Hierarchical model supports custom encapsulation as well

Basic Structured Graphics

Some Trade-offs

- Supporting reuse increases memory consumption
- Redraw and refresh can take more time
- Combined, can effect ’snappiness’ of UI

Though modern computing power negates most of these concerns

Basic Structured Graphics

Redraw and Refresh Operations

Operation depends on underlying algorithm (the how and when)

- One approach is to redraw every object every time a change occurs to any graphical object.
- Draws all objects in the hierarchy from back to front (from a display perspective), top down hierarchically

Basic Structured Graphics

Redraw and Refresh Operations

Operation depends on underlying algorithm (the how and when)

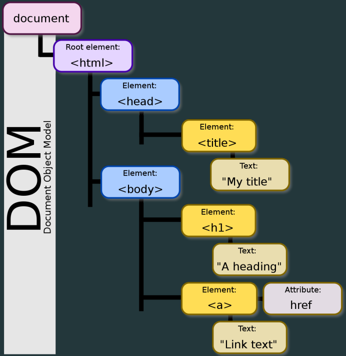
- Another approach is to only redraw the area of the display that has changed.
- Capture all objects that intersect the area to be redrawn, redraw from back to front.

Basic Structured Graphics

Structured Graphics System

The DOM used in web browsers is an example of structured graphics.

> Maintains a hierarchical list, or ”retained object model” of all graphical objects. Update the screen by editing objects in the list.



[Eriksson, 2022]

Basic Structured Graphics

The Hierarchical List

Graphical Primitives

- text, icons, and shapes

Aggregates

- collections of graphical objects
- “‘div“‘ or ’Groups’ in SVG
- Parent/child relationship

Basic Structured Graphics

The Hierarchical List

Hierarchies are built through aggregate object types and inheritance.

One Example: CSS.

```css
body {
    color: green;
}
.my-class-1 a {
    color: inherit;
}
.my-class-2 a {
    color: initial;
}
.my-class-3 a {
    color: unset;
}
```

```html
<ul>
    <li>Default <a href="#">link</a> color</li>
    <li class="my-class-1">Inherit the <a href="#">link</a> color</li>
    <li class="my-class-2">Reset the <a href="#">link</a> color</li>
    <li class="my-class-3">Unset the <a href="#">link</a> color</li>
</ul>
```

[Mozilla, 2021]

Basic Structured Graphics

The Hierarchical List

Issues and Design Considerations

- Complexity increases with features
- Which point in the hierarchy has responsibility for a given property
- Which hierarchy is responsible for themes? events propagation?
- Should objects change in appearance based on type or should each type be a new object?

Assignment 1: Roll Your Own Button

Let’s Dive In!

References

References i

Eriksson, B. (2022).
Document object model.

Mozilla (2021).
Cascade and inheritance.