UCI University of California, Irvine

Informatics 134

Software User Interfaces
Spring 2021

Mark S. Baldwin
baldwinm@ics.uci.edu

4/13/2021

Agenda

1. Next Class
2. Building User Interfaces
3. Assignment 3: Custom Graphical Toolkit

Next Class

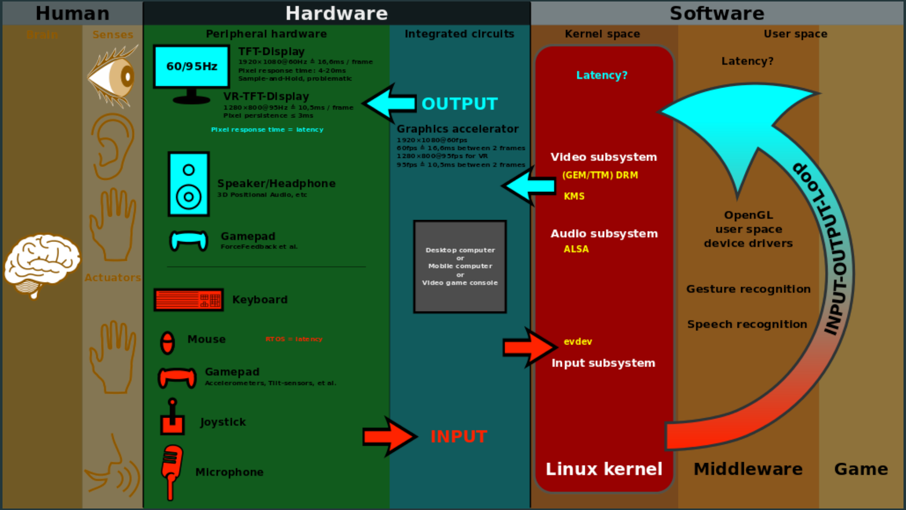
Thursday

- Team Evaluations
- Keep working on T2 (DUE 4/26)
- Get started on A3 (DUE 5/10)

Building User Interfaces

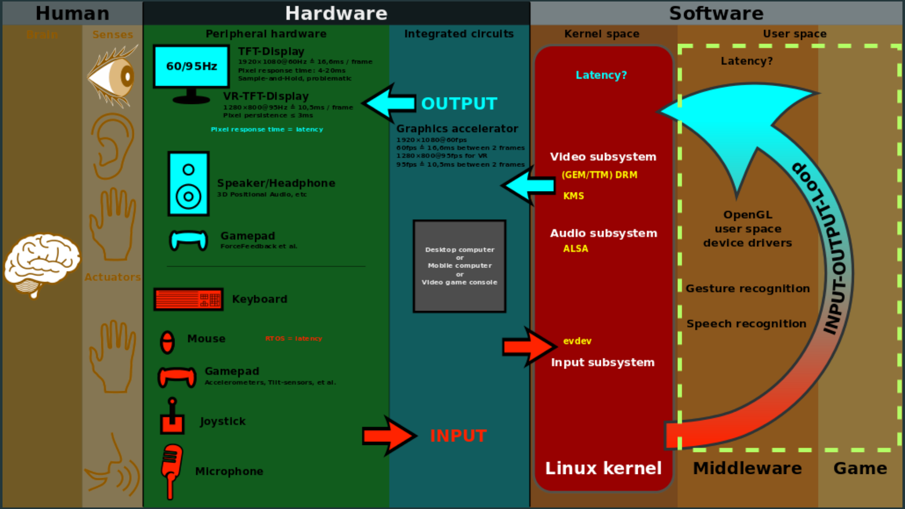
Building User Interfaces

Building User Interfaces



[Wikipedia, 2021]

Building User Interfaces

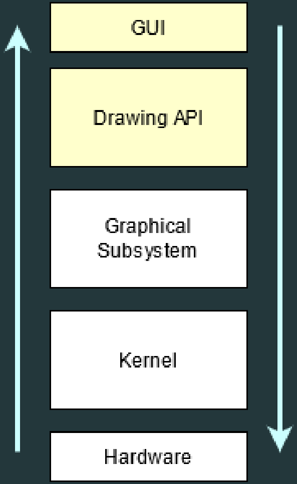


[Wikipedia, 2021]

Building User Interfaces

User Interfaces from an Architectural Level

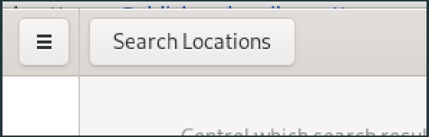
- GUIs rely on many different units of code to function
- Data propagates between these units to represent state and interaction
- Each unit is responsible for making decisions on how to handle a particular operation



Building User Interfaces

The Button Example

What are some observations that we can make about its functionality?



Building User Interfaces

The Button Example

- Clickable
- Can visually change in response to interaction
- Executes a command

Building User Interfaces

The Button Example

In computer science, these observations can be represented by a state chart and implemented through a state machine.

Building User Interfaces

The Button Example

In computer science, these observations can be represented by a state chart and implemented through a state machine.

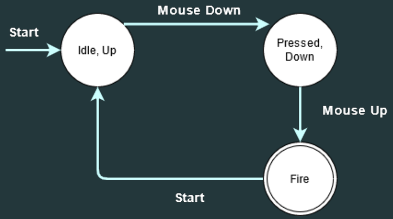
- Clickable
- Can visually change in response to interaction
- Display data
- Can execute a command

Button State Chart

Current State	Transition	Present State
Idle	Mouse Down	Pressed
Pressed	Mouse Up	Execute
Execute	Mouse Up	Idle

Button State Chart

The simple button example represented using a state chart diagram



Building User Interfaces

The Button Example

Although this simple button example could work, most buttons (and other widgets) are typically far more complex.

What are some other states we might need to support in a fully featured button?

Tiny widgets filled with tiny state machines

Let’s revisit our earlier observations…

- Clickable
- Can visually change in response to interaction
- Display data
- Can execute a command

Can you think of any architectures that might bring these widgets together?

Building User Interfaces

Tiny Widgets...MVC

The Model-View-Controller paradigm is the dominant way to represent groups of widgets.

Controller	Clickable
View	Can visually change in response to interaction
Model	Display data
Controller	Can execute a command

Building User Interfaces

MVC Refresher

Model

Stores data to be presented by the GUI

View

Visualizes the data stored in the model

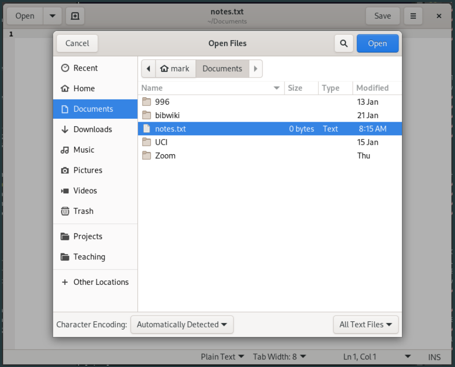
Controller

Handles user input, model data, and updates

Building User Interfaces

Model...View...Controller

How would we represent the GUI pictured here using an MVC architecture



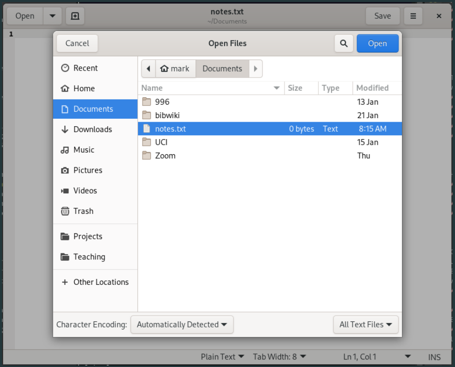
Name	Size	Type	Modified
996			13 Jan
bibwiki			21 Jan
notes.txt	0 bytes	Text	8:15 AM
UCI			15 Jan
Zoom			Thu

Character Encoding: Automatically Detected

All Text Files

Building User Interfaces

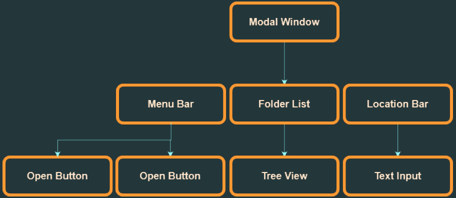
But there's something else interesting about this GUI...



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

Building User Interfaces

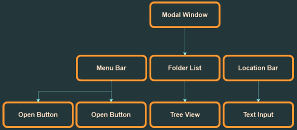
But there's something else interesting about this GUI...

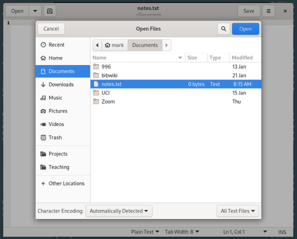


Building User Interfaces

GUIs are structured hierarchically

- Some widgets can contain other widgets
- Container widgets are not always visible
- Hierarchical composition supports layout and communication between widgets



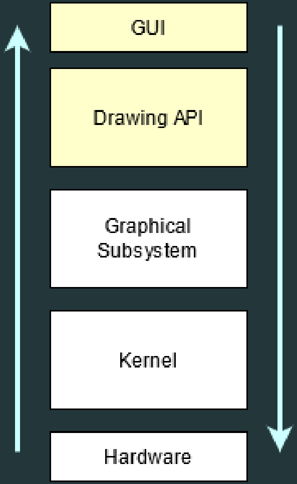


Name	Size	Type	Modified
996			13 Jan
bibwiki			21 Jan
notes.txt	0 bytes	Text	8:15 AM
UCI			15 Jan
Zoom			Thu

Building User Interfaces

Hierarchical Composition

- Layout managers
- Event handling and propagation

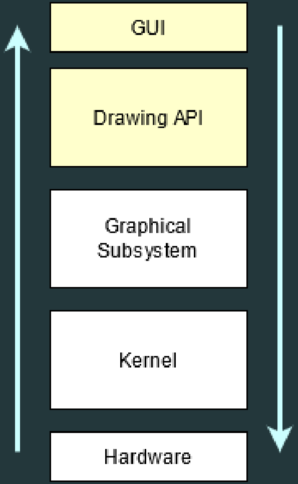


Assignment 3: Custom Graphical Toolkit

A3: Custom Graphical Toolkit

Build your own graphical toolkit

- Ground up using ’low-level’ graphics primitives
- Work in the browser
- Build on event propagation model of the DOM



A3: Custom Graphical Toolkit

Two Options:

Drawing

- Objects retain specification after draw
- Objects can be moved, changed, and scaled
- Better suited for user interfaces, high resolution
- In browser: Scalable Vector Graphics (SVG)

Painting

- Objects become pixels after draw
- Editable through change
- Better suited for games/graphics, low or fixed resolution
- In browser: Canvas

A3: Custom Graphical Toolkit

We will be using SVG

- SVG.js (https://svgjs.com)
- No dependencies–fast
- Choose your own editor or IDE

A Quick Sample

```html
<html lang="en">
<head>
  <meta charset="utf-8">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/svg.js/3.0.16/svg.min.js" ...></script>
  <script src="./toolkit.js" type="module"></script>
</head>
<body>
</body>
</html>
```

```javascript
SVG.on(document, 'DOMContentLoaded', function(){
    var btn = new Button;

    btn.onclick(function(event){
        console.log("clicked")
    })
});
```

A3: Custom Graphical Toolkit

You will create the following widgets:

- Button (use mine, customize)
- Check Box
- Radio Button
- Text Box
- Scroll Bar
- Progress Bar
- Custom (your choice)

A3: Custom Graphical Toolkit

You will be responsible for all of the following:

- Decide what functionality users of your widgets should be able to access.
- Apply a custom theme across all of your widgets
- Create a state chart for each widget
- Create a small GUI program that makes use of all of your widgets
- Write help documents

A3: Custom Graphical Toolkit

Getting started:

- Full assignment will be release after class
- Start looking at the SVG.js documentation
- Start working on your state charts
- We will be covering more over the next few lectures

References

References i

Wikipedia (2021).
Graphical user interface.