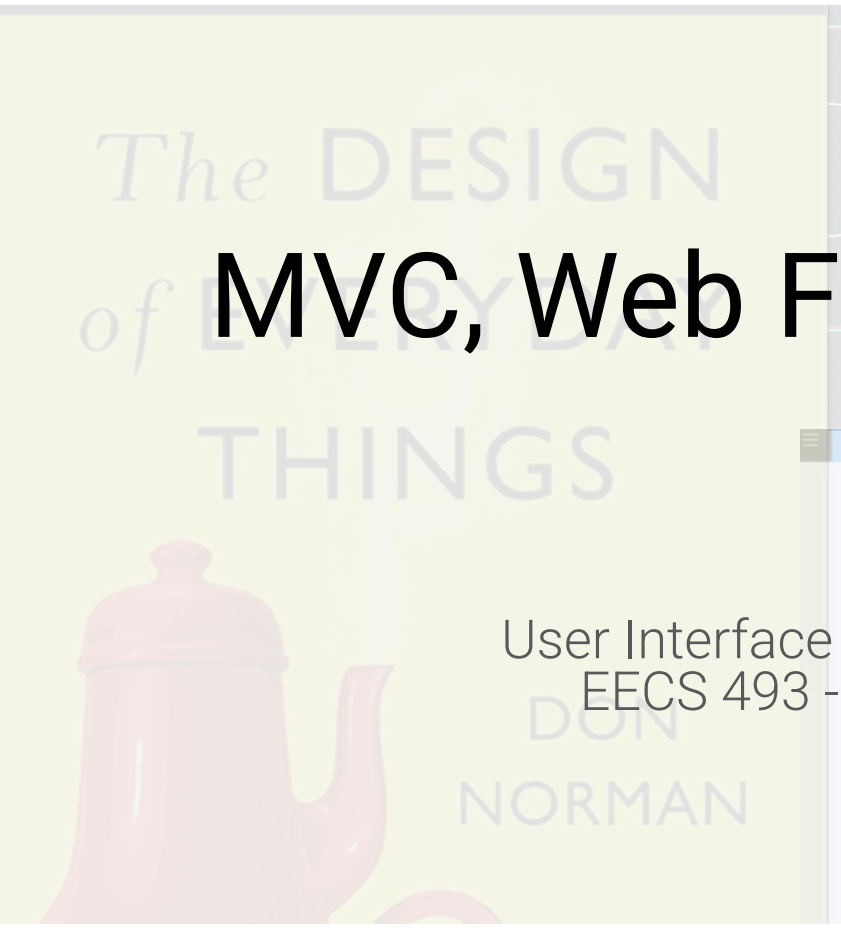
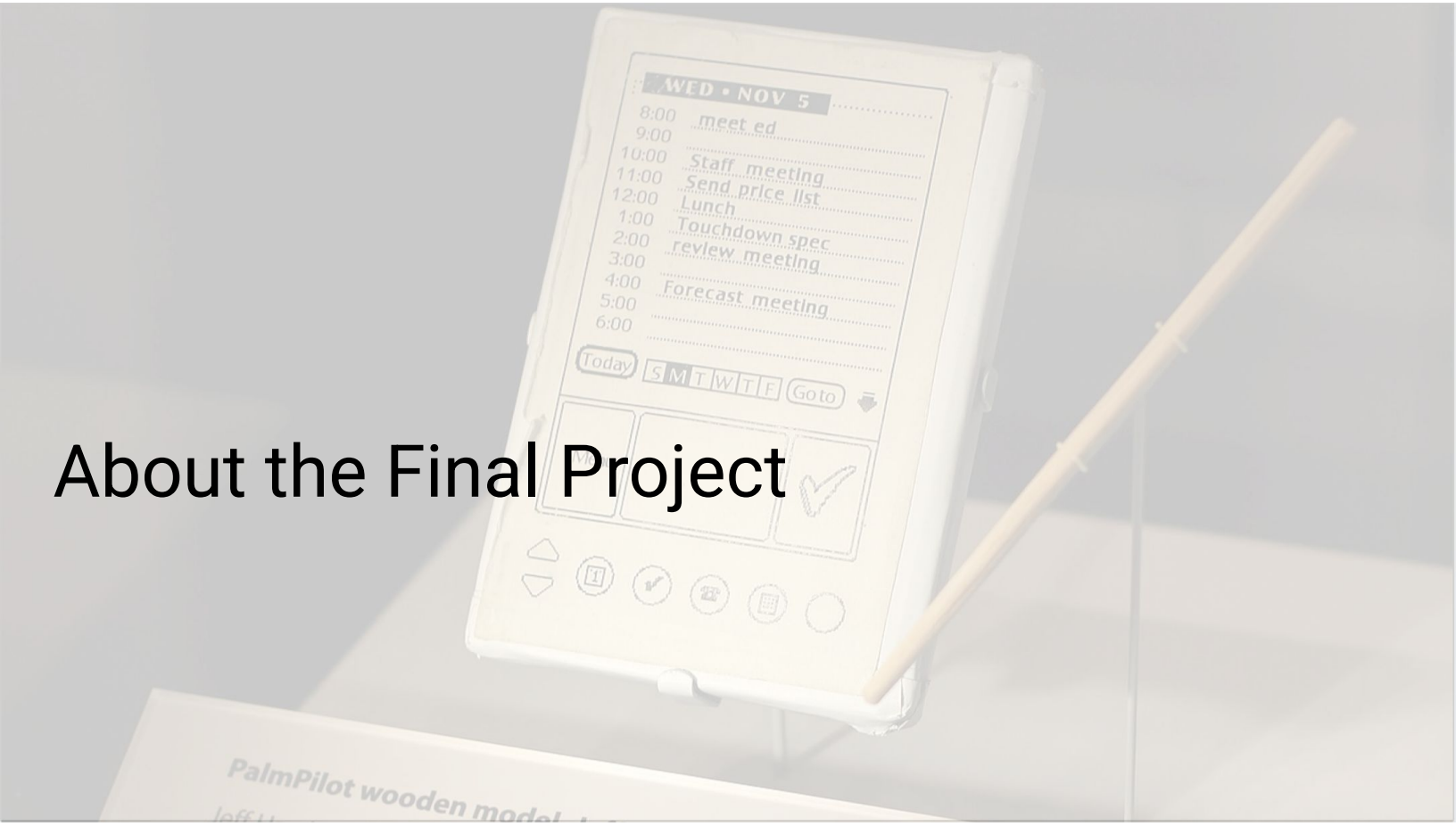
MVC, Web Frameworks



User Interface Development
EECS 493 - Winter 25

Goals for today:

1. Notes about the final project
2. The benefit of the “Model-View-Controller” paradigm
3. MVC examples and frameworks
4. Differences between frameworks, libraries and tools



About the Final Project

Mismatch between need and your idea

- If your idea is disconnected from the user need
- If your idea is too general
- That'll lead to shallow and boring designs

- **Instead:** focus on one specific goal, dive deep, show the step-by-step user interaction to achieve that goal

Example 1: consultation with professional

- **Example need:** students need to connect with professional to get advice
- **Not-so-good idea:** an app that users can connect with professionals
  - ***Not detailed enough!***
  - Existing solutions exist, why do they not work?
  - How do you recruit the professionals?
  - Availability is a big challenge, how do you address that?

Example 2: resume revision

- **Example need:** students need feedback on their resumes
- **Not-so-good idea:** an app that users upload resume, and suddenly AI made it better
  - ***Not detailed enough!***
  - How is it different from directly prompting ChatGPT?
  - What UI interactions will you present to the user?

Example 2: resume revision

- **Example need:** students need feedback on their resumes
- **Dive deeper:**
  - If AI gives suggestions, how is the user going to accept, reject, or make changes based on the suggestions?
  - If user is confused by the suggestion, how are they going to request more information from the AI?

Example 2: resume revision

- **What you need to do:**
  - The final project needs to layout all the specific interactions.
  - Use a real example resume, get real output from ChatGPT
  - Include all screens that the feedback is visible to the users, and specify the users' actions based on the feedback
  - Maybe users request specific sections for AI to focus on
  - Maybe users can ask questions for AI to help with
  - How do the users specify their intent on the UI?

Milestone 3

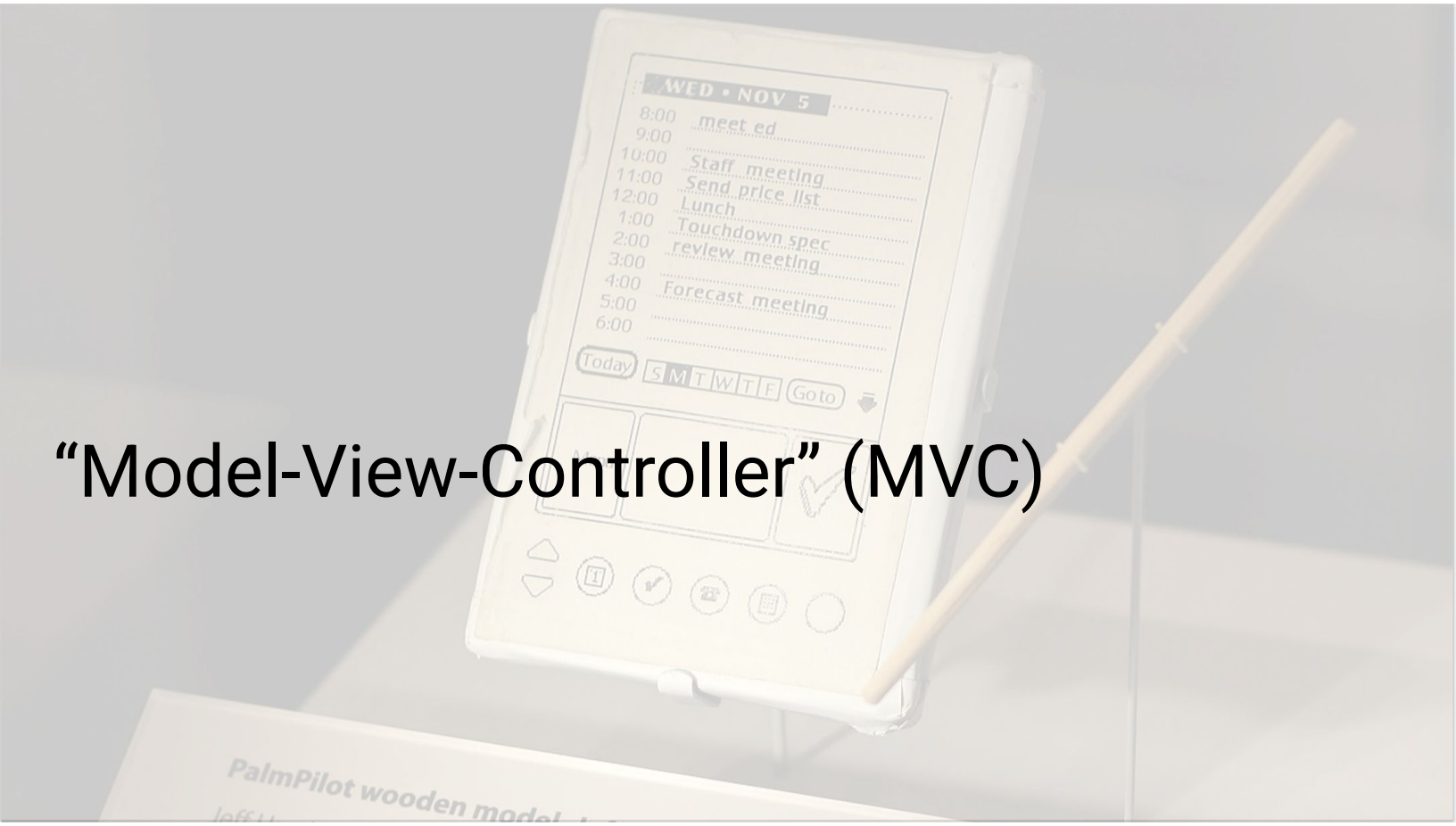
- **What you need to do:**
  - **Identify 1 goal**: e.g., specific enough such as “improving resume”, not as general as “find a job”
  - **Identify 2 tasks**: both supporting the same goal, each with a detailed sequence of subtasks – more on next slide
  - **Each subtask should be like**: “modifying an AI-generated revision” or “specifying which section I want AI to review”
  - **No obvious tasks**: e.g., Account Management

Tasks and subtasks

- As an example for using Gmail to send an email, you can imagine including in this task:
  - *compose an email, add a recipient, add someone in bcc, compose the email and mentioned an attachment, click send, notifies that subject line is empty, user add subject, notifies that didn’t attach any document, user attach, then successfully send, quickly realize an issue, undo, then add final touch, send, then check in Sent one more time.*
  - see, there’s a lot more than 8 for a workflow that **includes various edge cases and how the system should handle them**.

Your Figma prototype

- Build on other people's work! Use Figma templates!
- Deal with user input:
  - If you are not innovating on how to input text/numbers, I wouldn't spend the effort on creating components for that. You can give users specific info to put in (they have to be realistic and make sense for the behaviors later), and users just tap on a keyboard and the value is placed into the field.
  - Same for other types of user interactions
- Don't add a subtask just for the sake of it.



"Model-View-Controller" (MVC)

UI Development

(Once we have the design…)

How do we efficiently structure the development of UIs?

Some examples

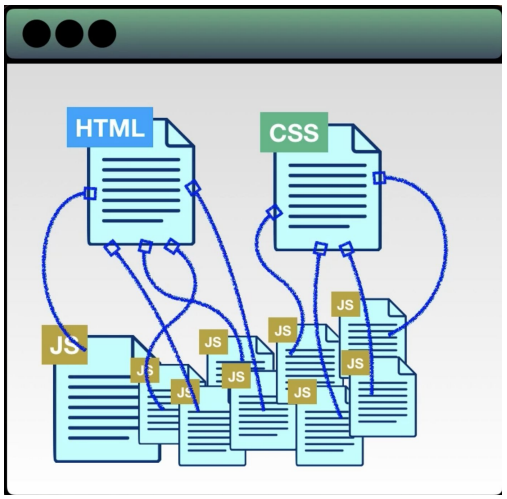
Think about HTML, CSS, Javascript

Think about web responsiveness

Think about Vue reactivity

Think about components in Figma

…



Abstractions to help

- Design patterns
  - “general, reusable solution to a commonly occurring problem” (Wikipedia)

- Separation of concerns
  - Reduce a complex problem space into manageable, separate components. The overall goal is to establish a well-organized system. (adapted from Simplicable)

Model-View-Controller

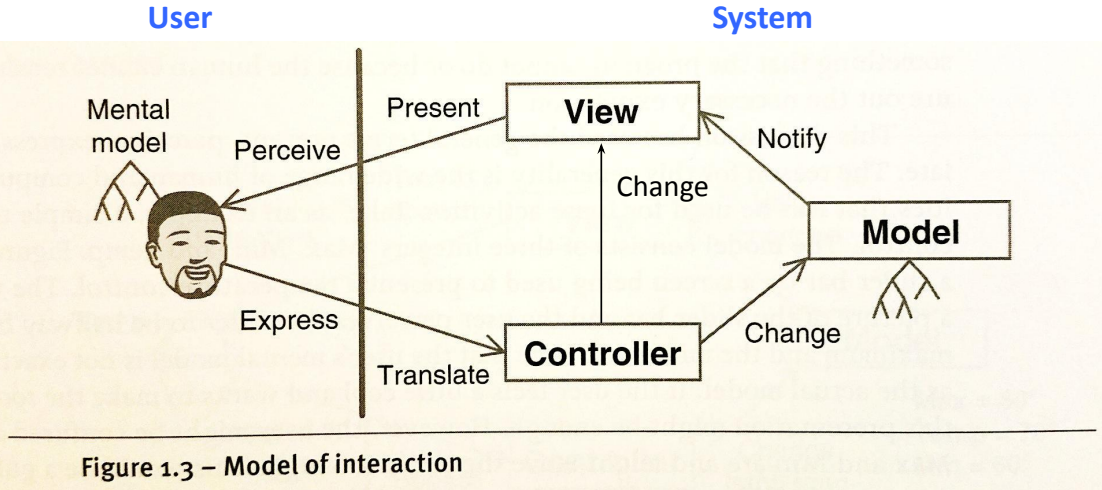


Figure 1.3 – Model of interaction

Model-View-Controller

- **Model**: system state
  - This is the core, underlying representation of the DOMAIN object
  - Think of it as the datasource, e.g., JSON, a database

- **View**: what users see
  - Gets values from the model, and generates output to the user
  - This might be a spreadsheet view, a pie chart, a scatterplot

- **Controller**: how stuff gets changed
  - Takes in user input (e.g., keystrokes), generates changes in the model and view
  - Figures out mouse events (and other events) happen, then calls the right thing

Model-View-Controller

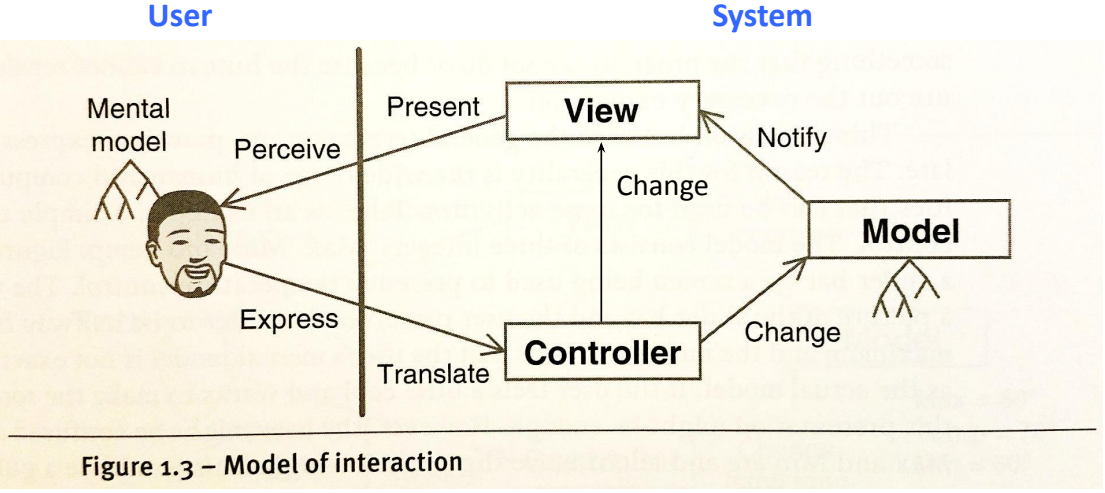
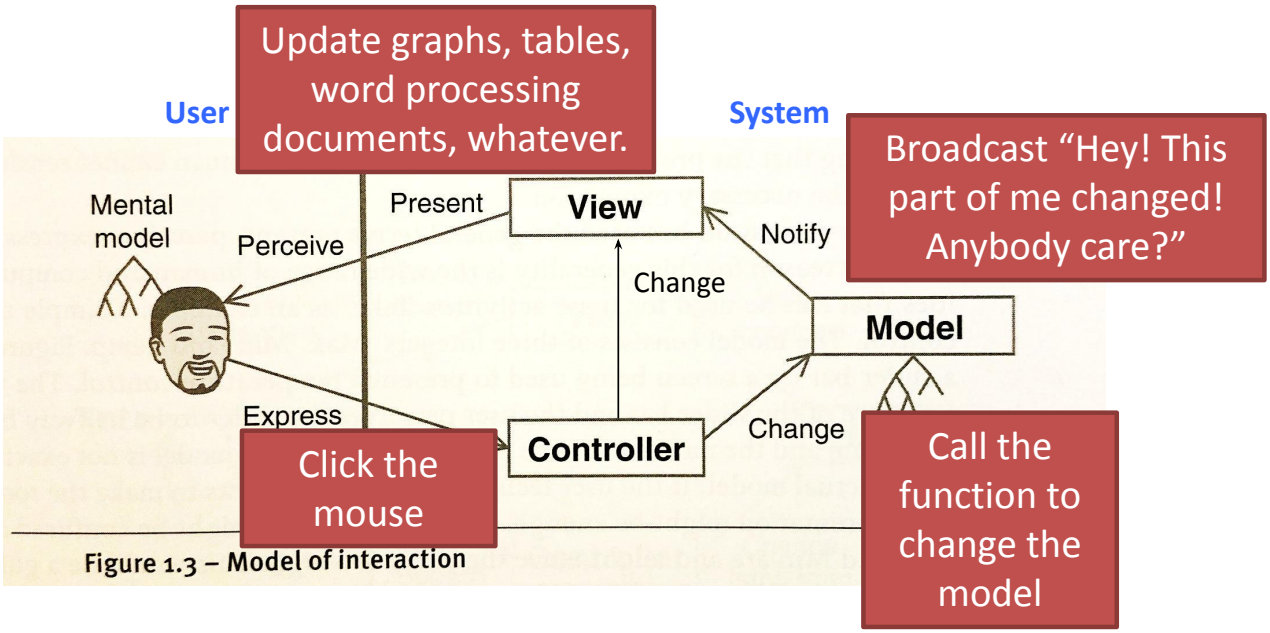


Figure 1.3 – Model of interaction

- User side of the equation parallels this…
  - Mental model: user state **[model]**
  - Actions (input): what the system sees **[view]**
  - Perceptions: updates user model **[controller]**

Model-View-Controller



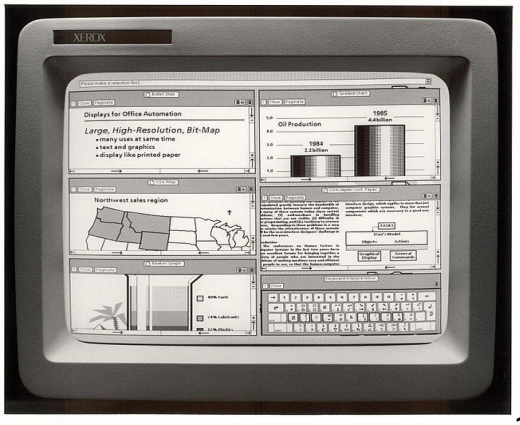
Tracking a User's Interaction

- System displays a prompt for a username (a View).
- User clicks in the prompt.
  - Controller: There's a mouse click! Username View, take it!
  - Username View: I've got focus!
- User types keys:
  - Controller: These are your keys
  - Username View: I've got those letters. Model, please update.
  - Username Model: I've got those letters. The model has now changed! Everyone who cares, you are notified and please update!

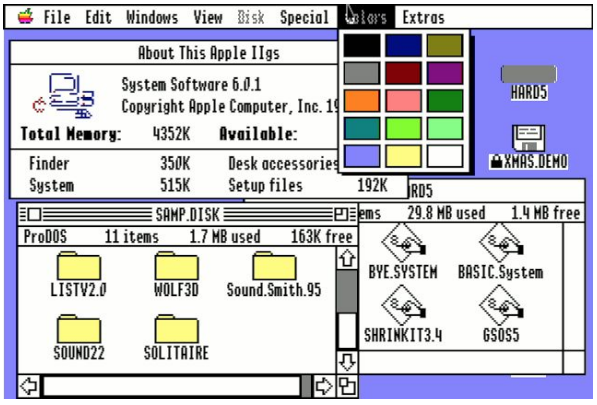
Model-View-Controller

- Many variants of MVC
- The critical pieces that cross all of versions MVC:
  a. The view is separated from the domain logic and the model so that **the same view component can be used in many applications**
  b. The View and Model are de-coupled
     - The view does not talk directly to a database
     - The View and Model communicate through notifications or other mechanisms

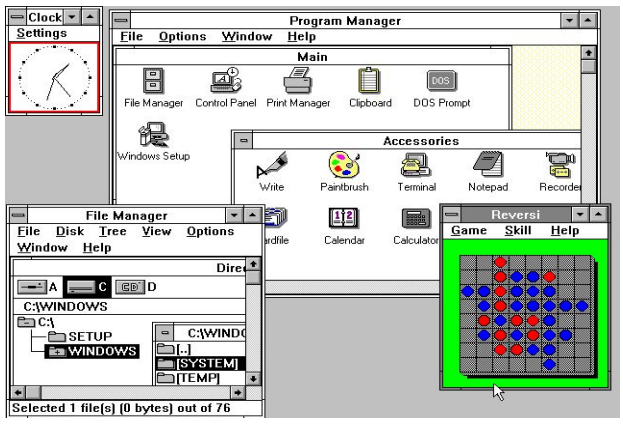
Windowed UIs



Xerox Star



MS Windows (3.0)



Modern Windows

Lecture 18 Survey 1: MVC

www.yellkey.com/camera ⇒ https://forms.gle/36aC5c9L4H3AREuv8

Survey 1: MVC

Consider the command line in a UNIX terminal

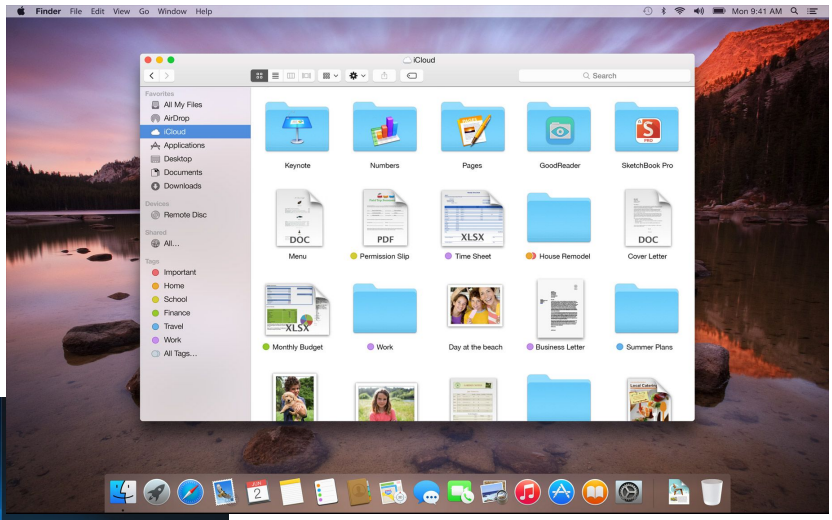
```
rohit@hogwartsCastle:~/Documents
Ψ cd /usr
rohit@hogwartsCastle:/usr
Ψ ls
bin           lib           libexec       local         sbin          share         standalone
rohit@hogwartsCastle:/usr
Ψ
```

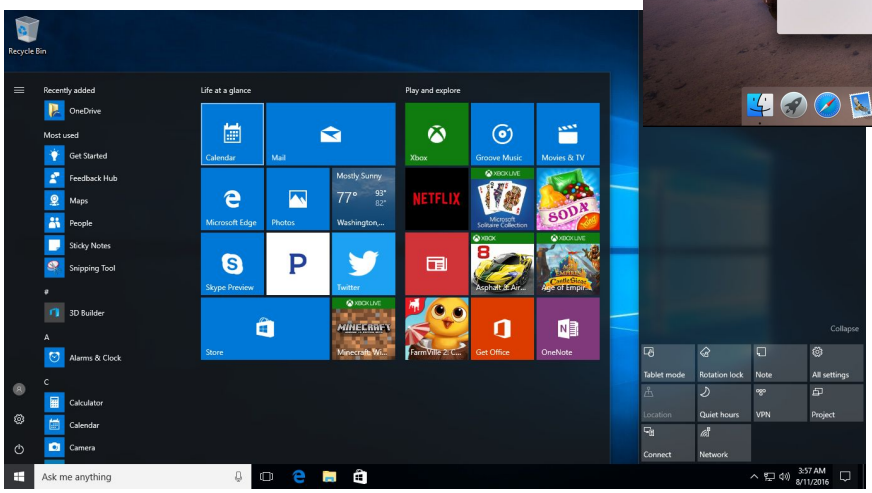
Given what you know about normal terminal interfaces, which of the below is likely true? *

- [ ] The terminal's controller is reading keyboard events
- [ ] The terminal's model is a tree of files
- [ ] The terminal's controller is tracking touch events
- [ ] The terminal's view is a text area with both output and input

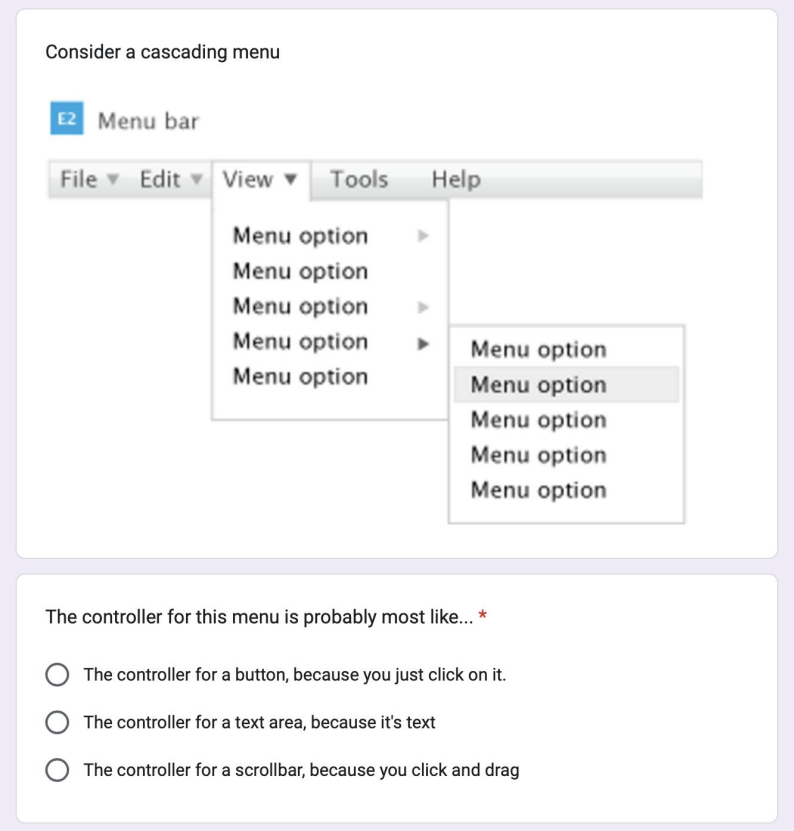
What changed?


```
rohit@hogwartsCastle:~/Documents
Ψ cd /usr
rohit@hogwartsCastle:/usr
Ψ ls
bin           lib           libexec       local         sbin          share         standalone
rohit@hogwartsCastle:/usr
Ψ
```



Survey 1: MVC



MVC Frameworks

MVC Everywhere

- MVC is the most important ***design pattern*** in UI work
- It fits
  a. Web Development with Frameworks
  b. iOS Dev with UIKit
  c. Android Dev with Android Architecture Components
  d. Desktop App Dev with Java Swing
  e. Game Dev with Unity Engine

MVC Everywhere

- What examples/analogies can you come up with?

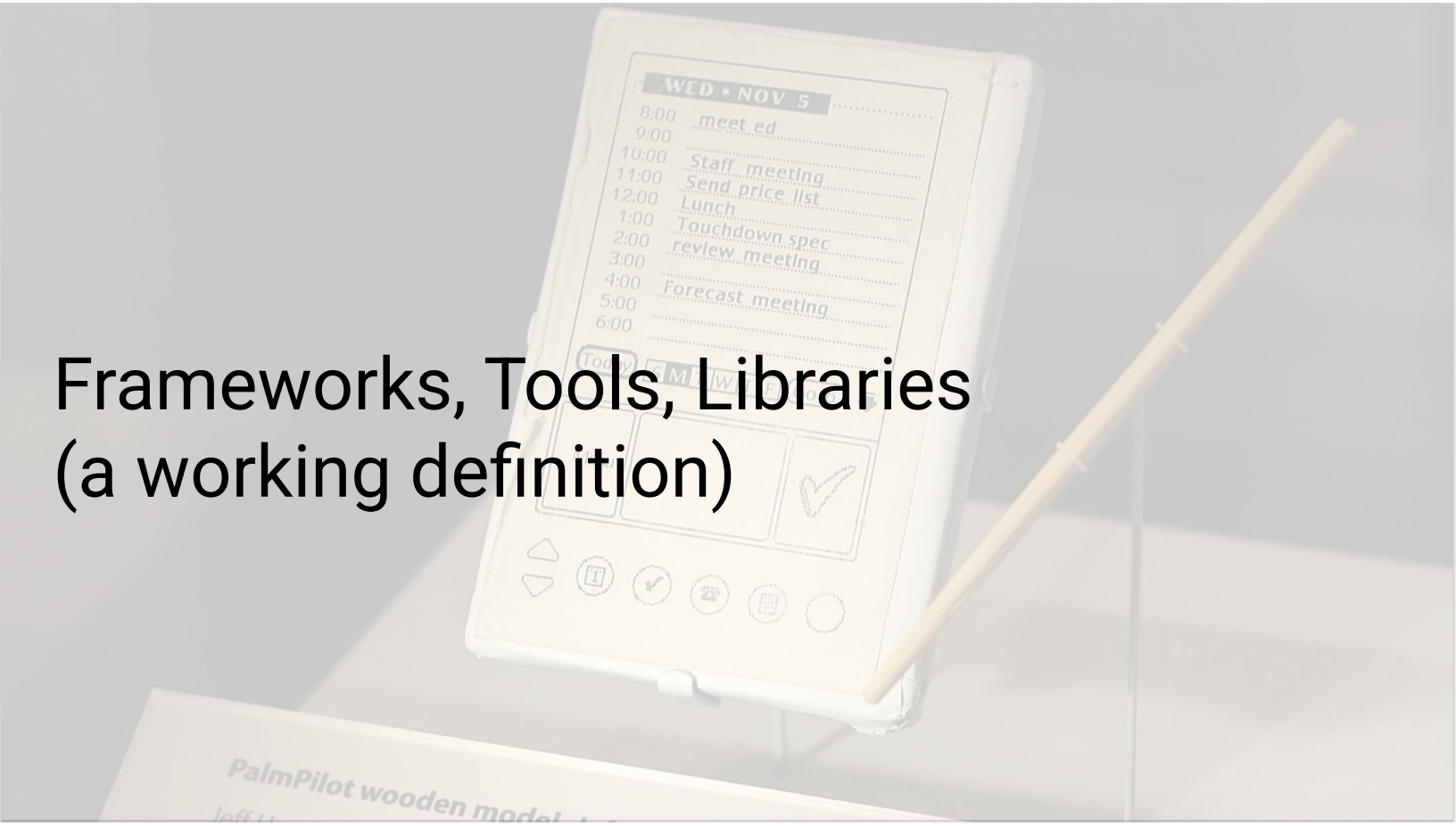
MVC has some variants

- Besides traditional MVC…
- Model-View-ViewModel (MVVM)
- Model-View-Presenter (MVP)
- Component-based MVC
- Middleware-based MVC

- **Commonality**: separation of concerns, modularity, reusability

MVC <> web frameworks

Web Framework	MVC Support
Ruby on Rails	Yes
Django	Yes
Vue.js	Partial (Component-based)
React	Partial (Component-based)
Angular	Partial (MVVM)
Express.js	Partial (Middleware-based)



Frameworks, Tools, Libraries (a working definition)

We have used many things in this course

- HTML, CSS, Javascript, jQuery, Bootstrap, Vue
- Visual Studio Code, Sublime, Chrome DevTools, Figma
- Human-Centered Design
- Semi-structured interviews, Affinity Diagram, Speed dating, Heuristic Evaluation, Think-Aloud…

What are Tools?

- Tools are software programs or utilities that help you solve a problem
- E.g., Visual Studio Code, Sublime, Chrome DevTools, Figma
- Often are **language-agnostic**!

What are Libraries?

- A collection of functions that solve a class of problems. Libraries typically have commonalities in approach, call-style, etc.
- Examples: jQuery, D3.js for data viz
- Often are **language-specific**!

What are Frameworks?

- A framework scaffolds the creation of a tool or application. It has initial connections and definable “slots” that developers can use to accomplish some task.
- Gives you a structure.

- Example: Vue, Bootstrap, React

- Often are **language-specific**!

http://stackoverflow.com/questions/3057526/framework-vs-toolkit-vs-library

What’s the Difference?

*“The most important difference, and in fact the defining difference between a library and a framework is **Inversion of Control**.*

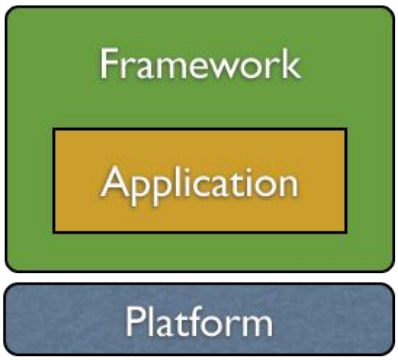
...

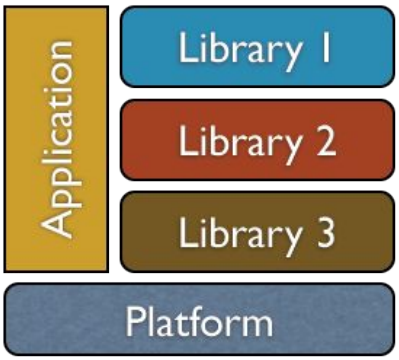
it means that when you call a library, you are in control. But with a framework, the control is inverted: the framework calls you.This is pretty much the definition of a framework. If it doesn't have Inversion of Control, it's not a framework.”

http://stackoverflow.com/questions/3057526/framework-vs-toolkit-vs-library

Examples

- jQuery is a library
- Vue, React, Bootstrap are frameworks
- The Chrome debugger is a tool





http://tom.lokhorst.eu/2010/09/why-libraries-are-better-than-frameworks

L18 Survey 2: Frameworks, Tools, Libraries

www.yellkey.com/nearly ⇒ https://forms.gle/vWbJrw9ApdrNZkX27

Survey 2: Frameworks, Tools, Libraries

A _____ is language-specific AND defines how you structure your program. In a sense, it calls your code -- you fill in slots and provide components. *

- ○ Framework
- ○ Tool
- ○ Library

Survey 2: Frameworks, Tools, Libraries

A _____ is language-specific but not how/what you program. It provides functionality that the developer can use. *

- ○ Framework
- ○ Tool
- ○ Library

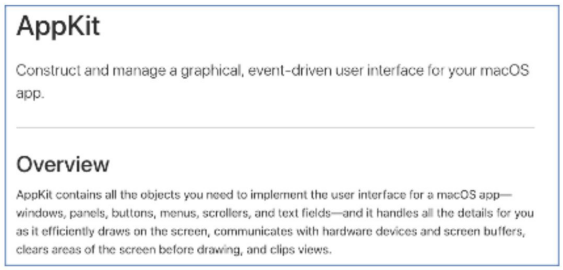
Survey 2: Frameworks, Tools, Libraries

A _____ is relatively language agnostic. It doesn't care how you program *

- ◯ Framework
- ◯ Tool
- ◯ Library

Survey 2: Frameworks, Tools, Libraries

Apple's AppKit

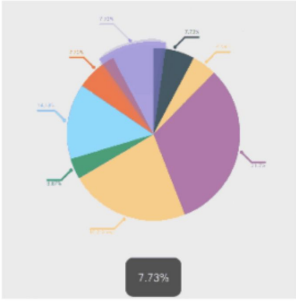


When developers are building for Mac OS X, you can use AppKit, which is a complete application...that doesn't do anything. You fill in pieces and override some default functionality to make your application. AppKit is... *

- ◯ A Framework
- ◯ A Library
- ◯ A Tool

Survey 2: Frameworks, Tools, Libraries

AnimatedPieView provides support for displaying pie and ring charts in Android applications.



AnimatedPieView is... *

- ◯ A Framework
- ◯ A Library
- ◯ A Tool

Goals for today:

1. Important notes about the final project
2. The benefit of the “Model-View-Controller” paradigm
3. MVC examples and frameworks
4. Differences between frameworks, libraries and tools