

AR Design

Tue, August 11 (Week 8)

Design

As solving problems.

For software engineering: how to build an application that solves a problem.

For augmented reality: how to build an AR application that solves a problem.

Usually, having a framework for thinking helps a lot.

Model-view-controller

The dominant framework behind all applications built for users.

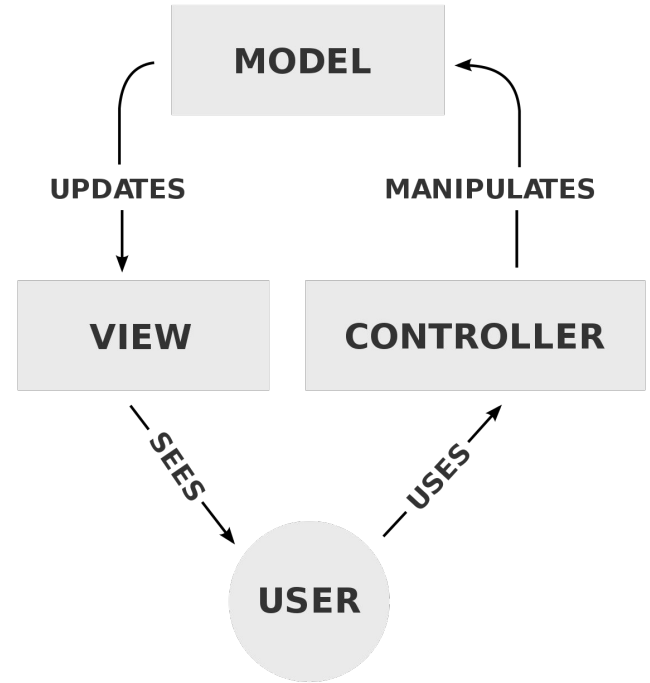
TODO app as an example:

Model: TODO tasks (data)

View: how to show the tasks

Controller: how to update the tasks

(Note: this is a very simplified version)



Model-view-controller

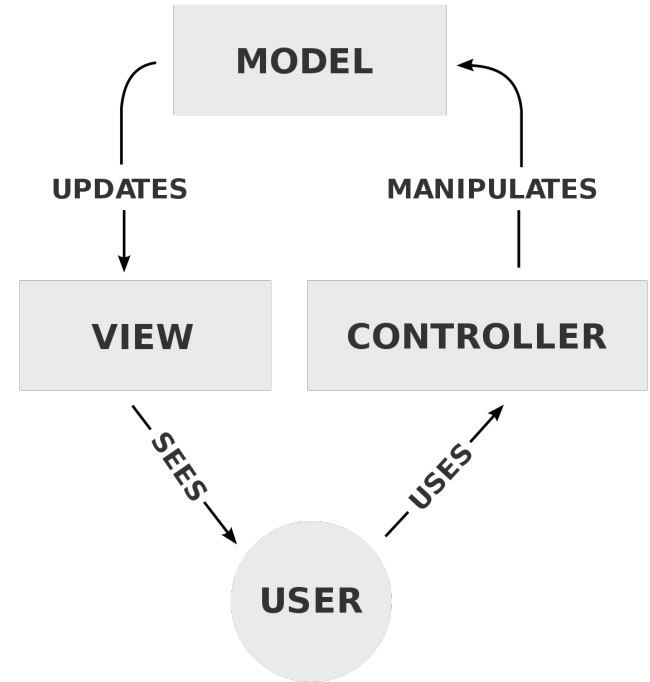
The advantage of MVC as a framework:

Allows splitting the software architecture into smaller pieces.

For example: if you separate the task from the button showing the task, you can write code for

1. storing the task in a database
2. updating the button based on the database

separately. Makes things easier.



Design Language

The set of terminologies you use to describe constructs.

Example: we used the word “button” in the previous slide. Surprisingly, we all now what it is.

There are names given to ideas and for 2D applications, there are implementations of them in code.

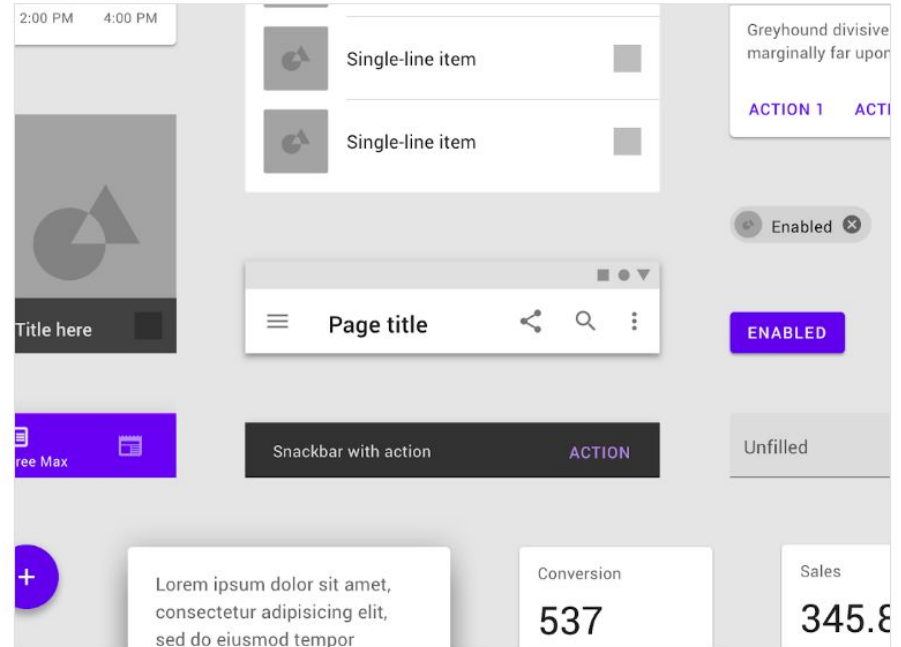
Example: Material Design

Google introduced Material Design in 2014.

While they build mobile and web applications, though the code behind them are written in different programming languages, they are based on the same design language.

Components

Material Components are interactive building blocks for creating a user interface, and include a [built-in states system](#) to communicate focus, selection, activation, error, hover, press, drag, and disabled states. Component libraries are available for [Android](#), [iOS](#), [Flutter](#), and [the web](#).



AR vs. Others (PCs or Phones)

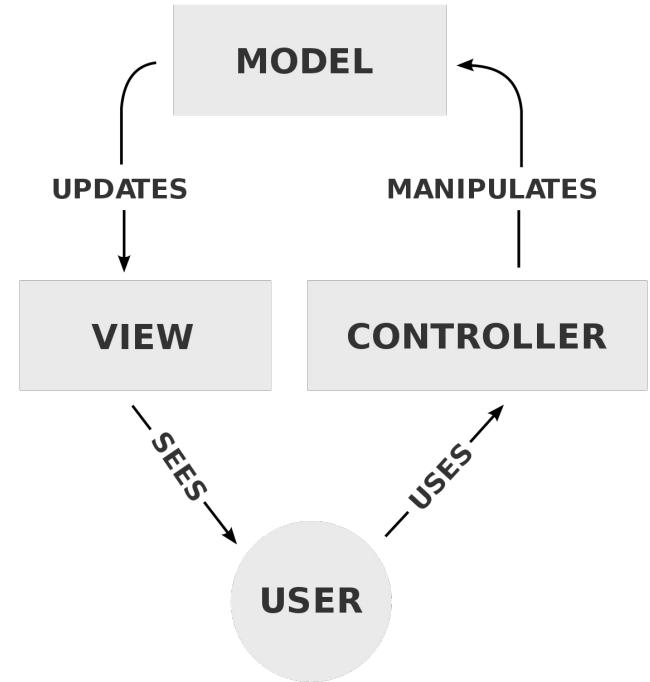
Model: almost the same

View: very different

Controller: very different (even PCs and Phones are different from each other)

View should be based on spatial components.

Controller is no longer based on keyboards, mouses, or touch screens.



Memory Lingering in Our Heads

People who work on AR applications have worked on 2D applications and cannot stop thinking in the 2D way.


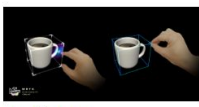
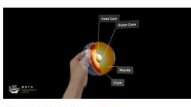






Comparison:

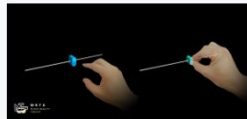
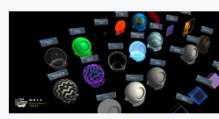

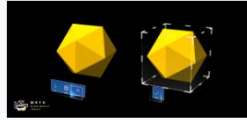



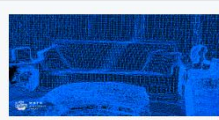

For 2D applications, people built the design language from scratch.

For AR applications, people are building it based on the ones for 2D.

Example: Mixed Reality Toolkit

UX building blocks

 <p>Button</p> <p>A button control which supports various input methods, including HoloLens 2's articulated hand</p>	 <p>Bounding Box</p> <p>Standard UI for manipulating objects in 3D space</p>	 <p>Object Manipulator</p> <p>Script for manipulating objects with one or two hands</p>
 <p>Slate</p> <p>2D style plane which supports scrolling with articulated hand input</p>	 <p>System Keyboard</p> <p>Example script of using the system keyboard in Unity</p>	 <p>Interactable</p> <p>A script for making objects interactable with visual states and theme support</p>
 <p>Solver</p> <p>Various object positioning behaviors such as tag-along, body-lock, constant view size and surface magnetism</p>	 <p>Object Collection</p> <p>Script for laying out an array of objects in a three-dimensional shape</p>	 <p>Tooltip</p> <p>Annotation UI with a flexible anchor/pivot system, which can be used for labeling motion controllers and objects</p>

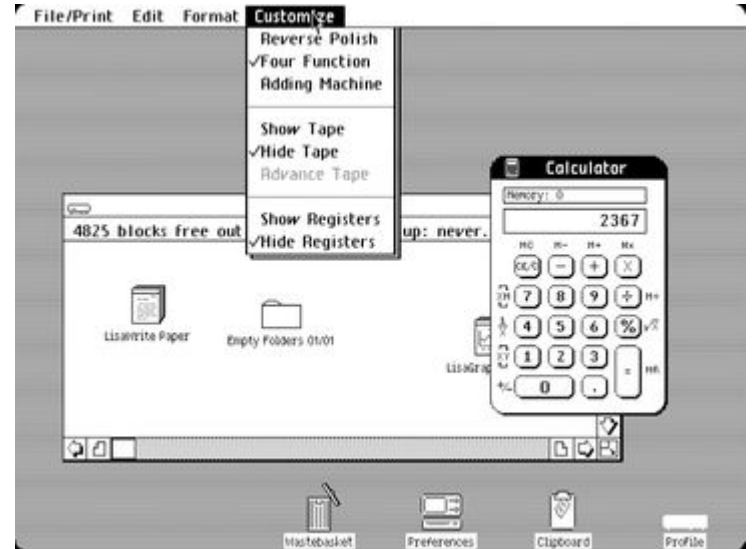
 <p>Slider</p> <p>Slider UI for adjusting values supporting direct hand tracking interaction</p>	 <p>MRTK Standard Shader</p> <p>MRTK's Standard shader supports various Fluent design elements with performance</p>	 <p>Hand Menu</p> <p>Hand-locked UI for quick access, using the Hand Constraint Solver</p>
 <p>App Bar</p> <p>UI for Bounding Box's manual activation</p>	 <p>Pointers</p> <p>Learn about various types of pointers</p>	 <p>Fingertip Visualization</p> <p>Visual affordance on the fingertip which improves the confidence for the direct interaction</p>
 <p>Near Menu</p> <p>Floating menu UI for the near interactions</p>	 <p>Spatial Awareness</p> <p>Make your holographic objects interact with the physical environments</p>	 <p>Voice Command / Dictation</p> <p>Scripts and examples for integrating speech input</p>

Many are coming from 2D applications.

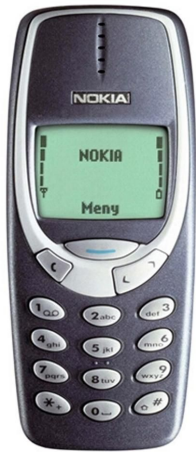
WIMP

Windows, Icons, Menus, Pointer

Invented in Xerox PARC in 1973,
popularized by Apple in 1984.



Phones



To Inherit or Not to Inherit?

Smartphones inherited WIMP except for the mouse and pointer, by interpreting touches as mouse clicks.

Should we do the same thing for augmented reality? In that case we do:

Windows floating in air, icons for applications that pop up those windows, pointing using our hands or hand controllers.

How to Control AR Applications

One Example Question: Should we use eye gaze as a pointer?

Advantage: we can point at things very quickly.

Disadvantage: often we do not want to point at it but just look at it.

How to Control AR Applications

Another Example Question: Should we ask people to touch a button or point the button and make a gesture?

Advantage of touching: it is very simple to understand how to use the buttons.

Disadvantage: users have to move to the buttons to click them.

How to Control AR Applications

Holding arm in air is hard. Perhaps, it might be better to have floating screens, sometimes attached on walls. (There is a reason why mice are so popular without floating in air.)



Design Guidelines

There are attempts to provide a design guideline.

Examples from Microsoft and Google:



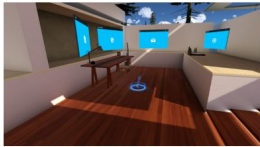
Comfort



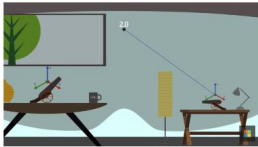
Holographic frame



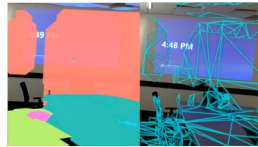
Types of mixed reality apps



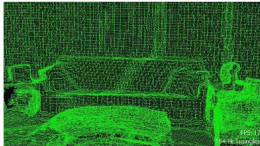
App model



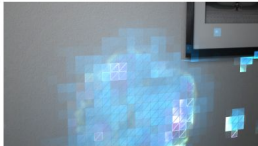
Coordinate systems



Scene understanding



Spatial mapping



Room scan visualization



Spatial anchors

Augmented Reality Design Guidelines

Introduction

