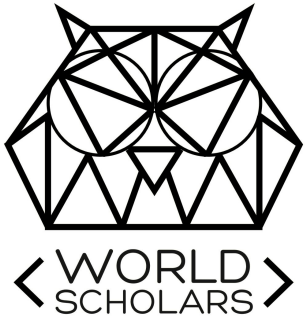


App Design

Mark Santolucito,
Yale CS PhD Candidate

Overview

- Definition of an “app” (not a-p-p)

- Design Process overview

- Prototype your own app
- Present plan

Types of Apps
(aka programs)

Mobile & Desktop Apps

- Platform specific
- Can be built with platform agnostic tool
- Touch (mobile) & Mouse/Keyboard (desktop)

Web Apps

- Works on all platforms
- Must be accessed through browser
- Support both Touch & Mouse/Keyboard input

The parts of an App

- Front End
  - Interface
  - Graphics
  - Art
- Back End
  - Databases
  - Computation
  - User Accounts

Front-end Basics

HTML, CSS, and javascript

- Core language of the web, fast to learn (codeacdemy.com)

Image editing

- Photoshop/GIMP, Illustrator, Wacom tablet

WYSIWYG (what you see is what you get)

- Very easy to use, but limited customization
- Wordpress, Dreamweaver, Squarespace

Back-end Basics

Databases and CRUD Operations

- Create, Read, Update, Delete

Local vs Remote Computation

- fast = locally, slow = remotely

Endpoints and the API

- Easy way to connect any front-end to key actions

Databases

MySQL, FireBase, MongoDB, Flat file storage

Store data on a seperate machine than your code!

“A backup a day keeps price-gouging data recovery services away”

Remote Computation

ssh, Amazon Web services (AWS), Digital Ocean

Run your apps on a machine “in the cloud”

Easy to restart and recover from failure

Endpoints and the API

Facebook API, Github API, Goverment APIs

- API demos

Key Issues

Technical
- MVP Feasability
- Long term goals
- Security

Buisness
- Market size
- Competition
- Risk/De-risking

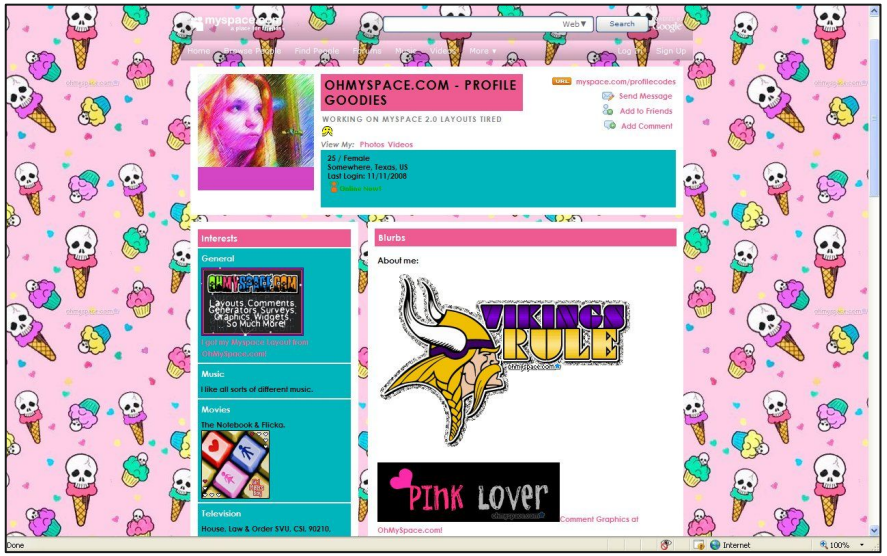
Human
- Team dynamic
- Passion
- Availability

Design Principles

- ***Simplicity***
- Uniformity
- Minimalist
- Navigation
- Helpful

Design Principles

- Simplicity
- ***Uniformity***
- Minimalist
- Navigation
- Helpful



Design Principles

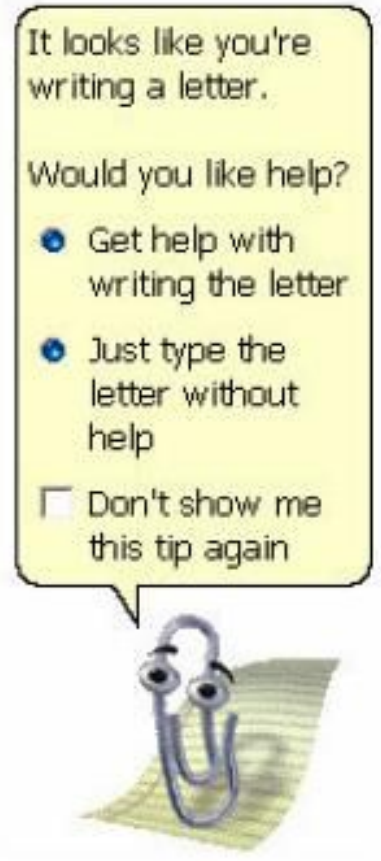
- Simplicity
- Uniformity
- ***Minimalist***
- Navigation
- Helpful

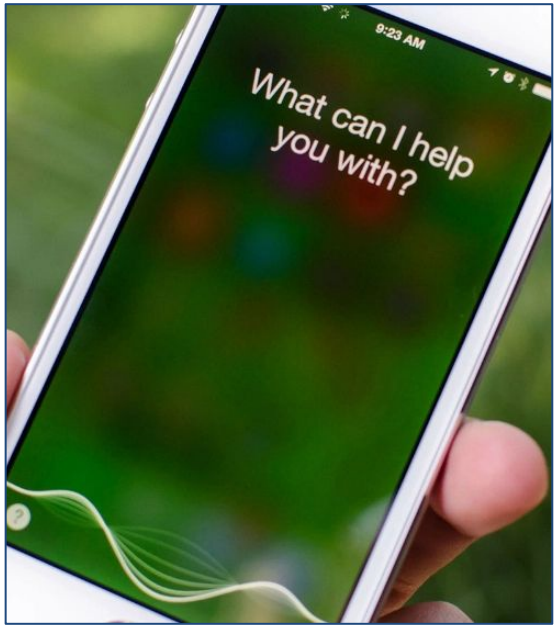
Design Principles

- Simplicity
- Uniformity
- Minimalist
- ***Navigation***
- Helpful

Design Principles

- Simplicity
- Uniformity
- Minimalist
- Navigation
- ***Helpful***





Clippy, RIP 2001-2007
Predecessor to Siri

Design Questions

- **User Classification**
  - Who are the users?
  - Regularity of use, Sophistication, Age group?
- **Task Analysis for Processes**
  - What are the required tasks
  - Break down into a flowcharts
- **Security and Access Control**
  - Establish requirements, priorities

Design Process

- *Step 0 - Market reserach*
- **Step 1** - User reserach and classification

- **Step 2** - Transform **Step 1** into specifications

- **Step 3a** - Design of software requirements
- **Step 3b** - Web interface prototyping and testing
- **Step 3c** - go back to **Step 2**

Design Process

- **Step 4** - Design core components of application (Version 0.1)

- **Step 5** - Usability testing
- **Step 6** - Implement more components
- **Step 7** - Go back to **Step 5**

Design Process

- **Step 8** – Release Party! (V 1.0)
- **Step 9** - Immediate bug fixes (V 1.0.1)
- **Step 10** - Ongoing maintenance (V 23.2.9)

Summary

- Mobile/Desktop/Web apps
- Technical, Buisness, and Human challenges
- Design Process requires
  - Reserach
  - Iteration
  - Patience

Try it out

- **Step 1** (10 min) - User classification
- **Step 2** (10 min) - Transform user descriptions into well-structured, specifications
- **Step 3** (10 min) - Paper prototype
- **Present** (5 min to prep) + (4 min/group)

Going forward…

Mark Santolucito
mark.santolucito@yale.edu
marksantolucito.com/gydo

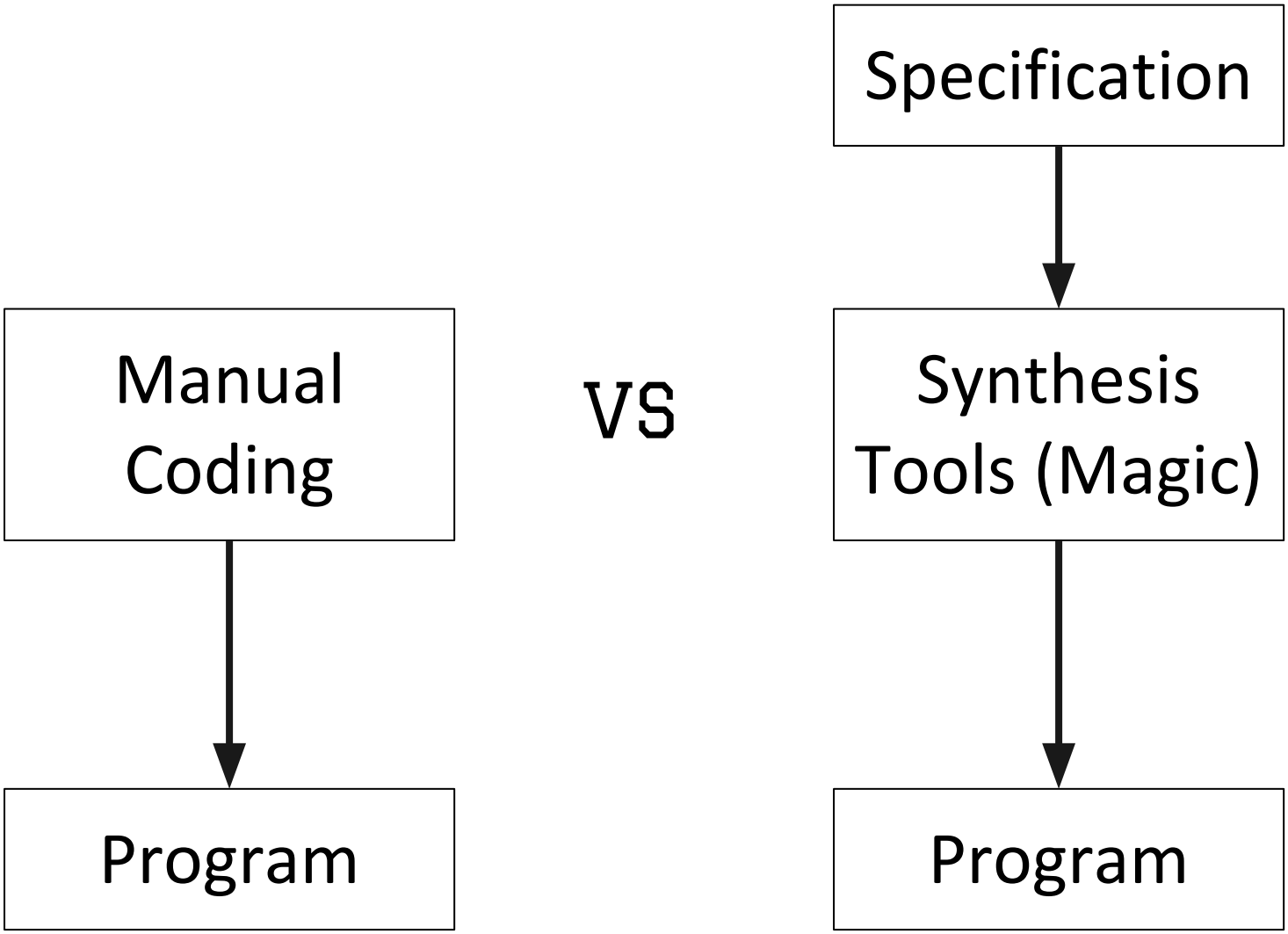
Bubble.io

Unity

Android SDK

Program Synthesis from TSL specifications

Program Synthesis



Program Synthesis

Programming
by Example

2 -> 4, 4 -> 8 ..... def fxn(x):
x * 2

2 -> 4, 3 -> 6, ... def fxn2(x):
4 -> 7, 5 -> 8 if (x < 4): x*2
else x+3

Program Synthesis

Logical Specifications

$\forall x . x*2 = out$ ..... `def fxn(x): x * 2`

$\forall x .$
$x * 2 = out \vee$
$x + 3 = out$ .....

```
def fxn2(x):
    if (x < 4): x*2
    else x+3
```

Logic Refresher

x	x is true
$\neg x$	x is not true
$x \vee y$	either x or y or both x and y are true
$x \wedge y$	both x and y are true
$x \Rightarrow y$	if x is true, then y is also true
$\forall x . x$	*for all* possible values of x, x is true
$\exists x . x$	there *exists* some value of x, such that x is true

Temporal Logic

x	x is true right now
◯ x	x will be true in the next step
$\mathcal{A}$ x	x is *always* (▢) true from now on
$\mathcal{E}$ x	x will *eventually* (◇) be true
x $\mathcal{U}$ y	x will be true *until* y is true

Temporal Logic for Apps

$\mathcal{A}$ (user opens app $\Rightarrow$ update news feed)

clicks upload a profile picture x $\Rightarrow$

$\mathcal{E}$ (profile picture = x)

display login page $\mathcal{U}$ user is logged in