Speech Input and Output

https://www.twitch.tv/seebotschat

This week

- Introduction to Voice UIs (VUIs)
- Thinking about the design of VUIs
- Interesting examples
- Prototyping VUIs with DialogFlow
  - Leading in to Project 2

Housekeeping

- Extra time for Project 1
- Demo code for Project 1
- Two extra credit assignments posted

Understanding speech input

Why speech?

- Shifting gears a little bit from focusing on specific population groups, to a whole class of interaction
- Why?
  - Speech is a broad-ranging topic
  - Speech just works differently than other technologies

Big ideas

- Consuming speech is very different than reading on screen
- Navigating speech interfaces is very different than navigating GUIs
- Development tools are quite different too

Drawing from our everyday experiences

Let’s talk about voice interaction

1a. When has it worked well for you?
2a. When has it broken down?

1b. Can we figure out **what kinds of tasks are well-suited** for voice?
1c. **What kinds of tasks are NOT** well-suited for voice?
2b. Can we come up with a list of the **types of errors** that occur?

(and write them down)

Good experiences

- Querying datasets, wikipedia
- Hands free situations
- Discrete tasks that we know it can do (single step)
- Tasks with a clearly defined workflow / logic tree
- Search (esp. on input constrained devices)
- Social interactions / practice communications
-

Limitations

- Asymmetry - receiving a large amount of information is hard
- Unconstrained input —> low accuracy
- Lock in
- Inferring context
- Voice characteristics (age, accent, disability, speech rate)
- Cloud based / network dependencies
- Learning features
- Debugging

Discussion

- menti.com, 59 98 81
  - Q1: For what kinds of tasks can voice interaction be most successful?
  - Q2: What kinds of errors can happen with voice interaction?

When it works well (and why)

More benefits of speech

- Support completion of tasks using natural language, without knowing commands
- Commands can take many forms
  (“play music”, “play Kanye West”, “play track 3 from *Graduation*”)
- Accessible to users of different abilities
- Can interact in the background or at a distance

When it does not (and why)

More challenges of speech

- Accounting for many different entry paths
- Partial or incorrectly formed requests
- Noise and recognition errors
- Navigating output can be tricky

When to include voice

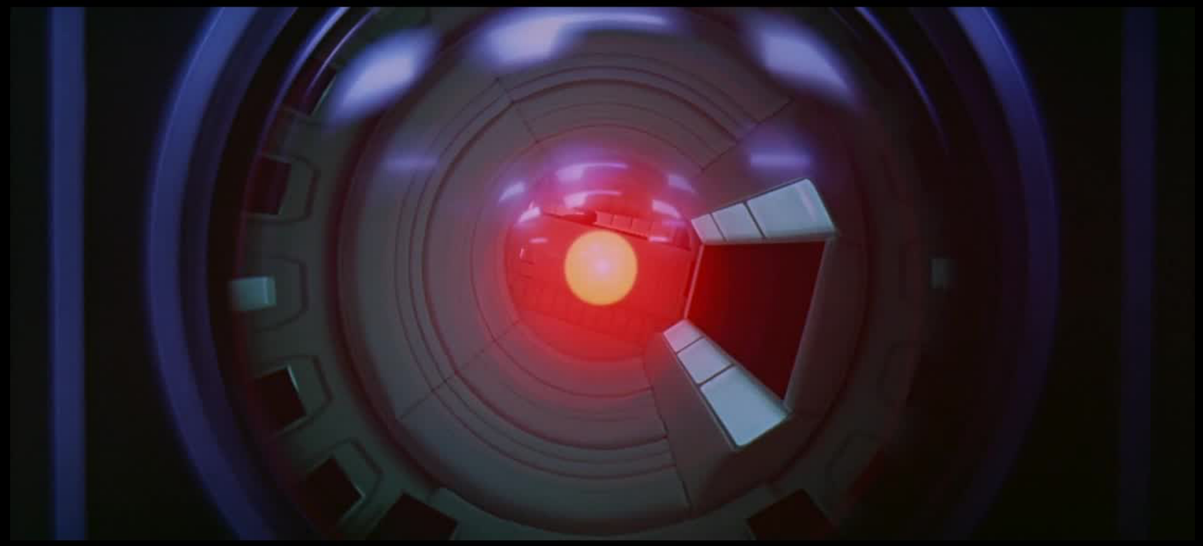
- There are certain contexts in which voice is the only (or obvious) solution
- But there are also certain kinds of tasks where voice is ideal (and others where it is not)

History of voice interaction

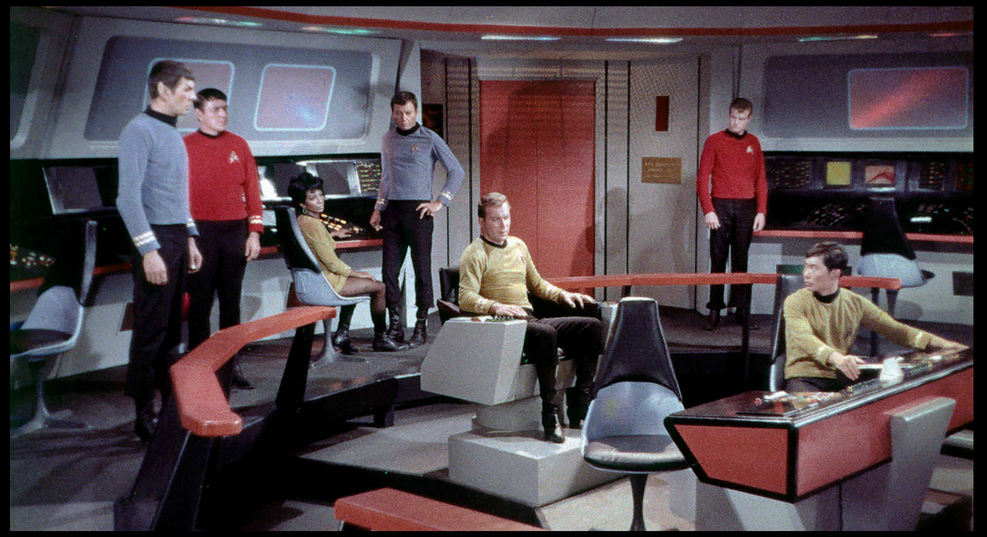
Examples from fiction

- This is an instance where we have had many fictional systems before real-world systems

Voice interfaces in fiction



HAL 9000, *2001: A Space Odyssey* (1968)



Star Trek (1966)

User expectations?

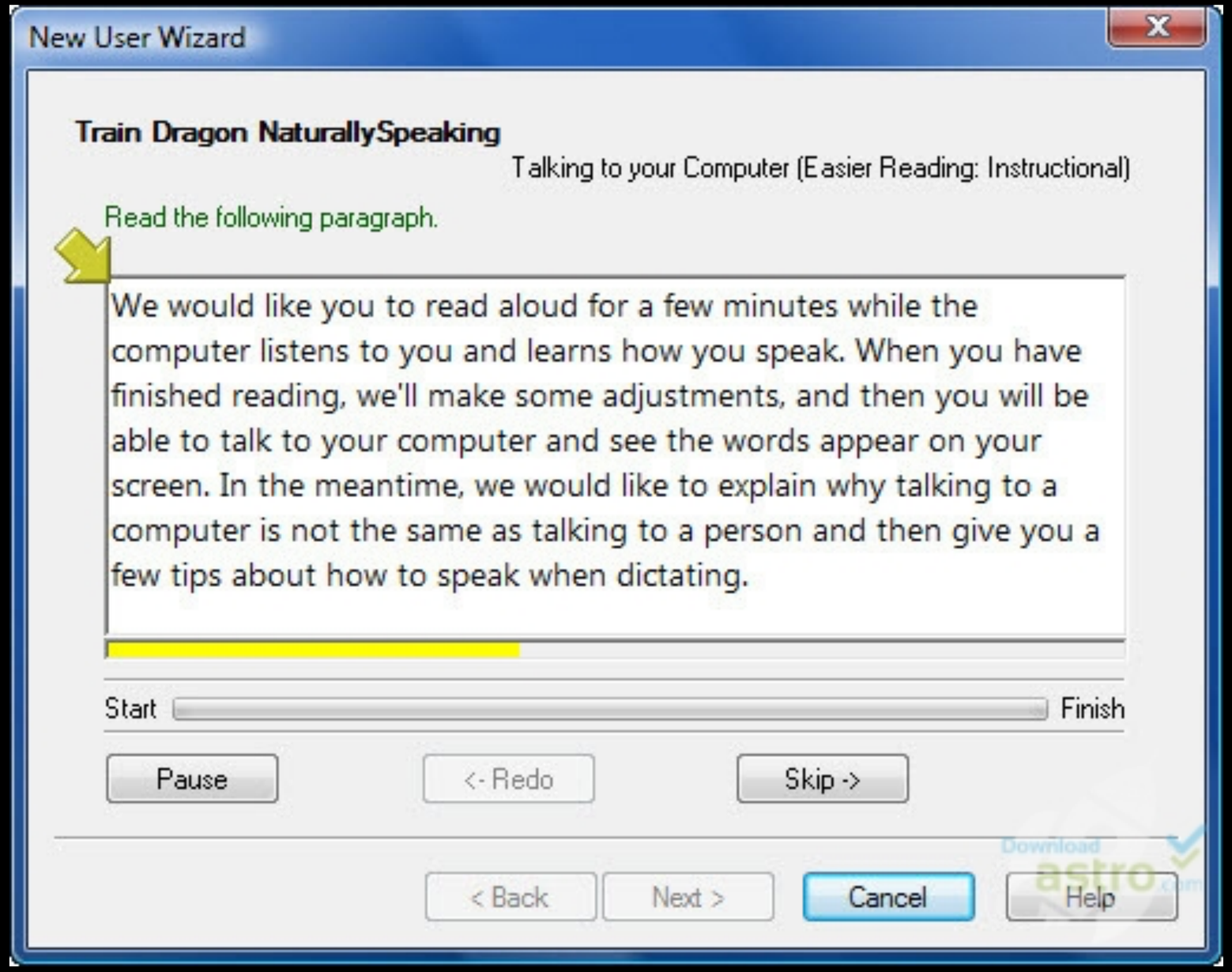
Some user expectations

- Can speak in a natural voice, at normal rate
- Can phrase commands in various ways and via natural language
- Can assume context from questions
- If information is missing, can ask follow-up
- Response is instant (or at least very fast)

A brief history of (real) voice UIs

- 1970s/80s - Interactive Voice Response systems (“press 3 for voicemail…”)
- 1990s/2000s - first commercial speech recognition systems (e.g., Dragon NaturallySpeaking)
  - Require significant per-user training
- 2010s - Siri and voice agents, **speaker-independent** speech recognition

Training speaker-dependent voice recognition

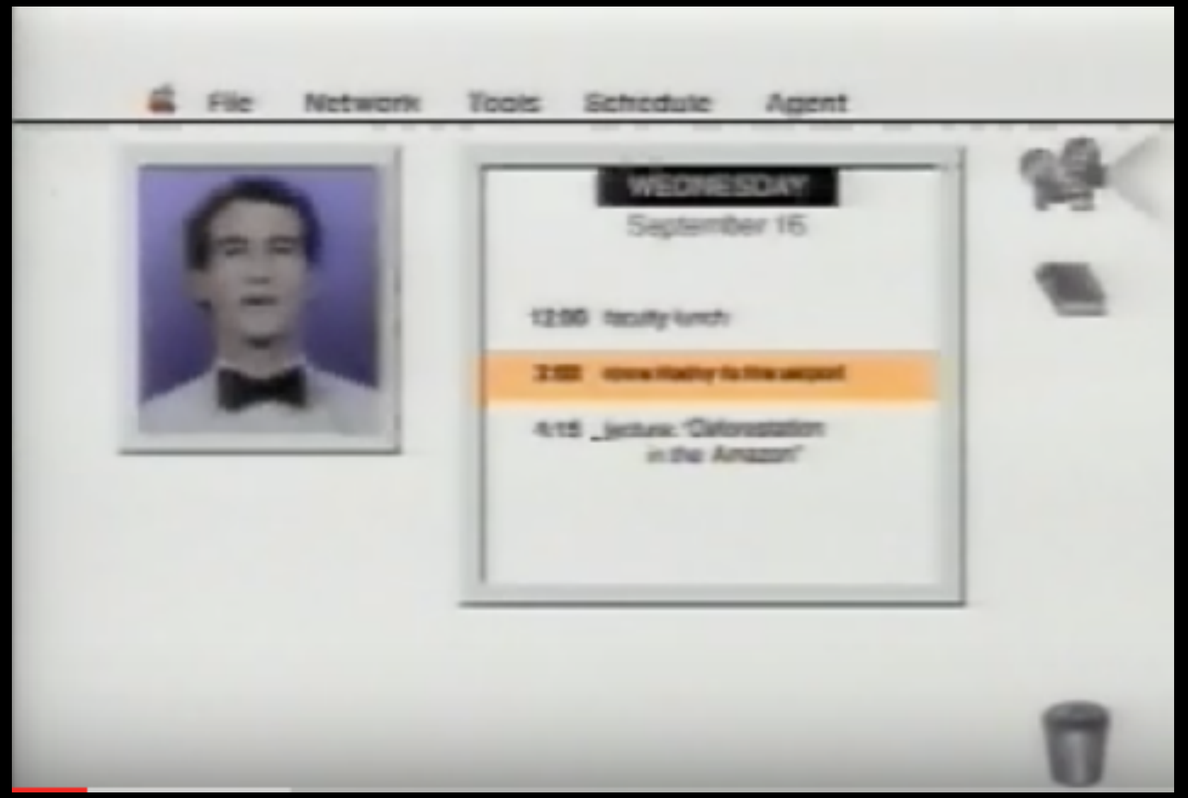


Research prototypes

- “Put that There”
- Knowledge Navigator

Put That There (1979)

Knowledge Navigator (1987)



What features did you notice?

Features in KN

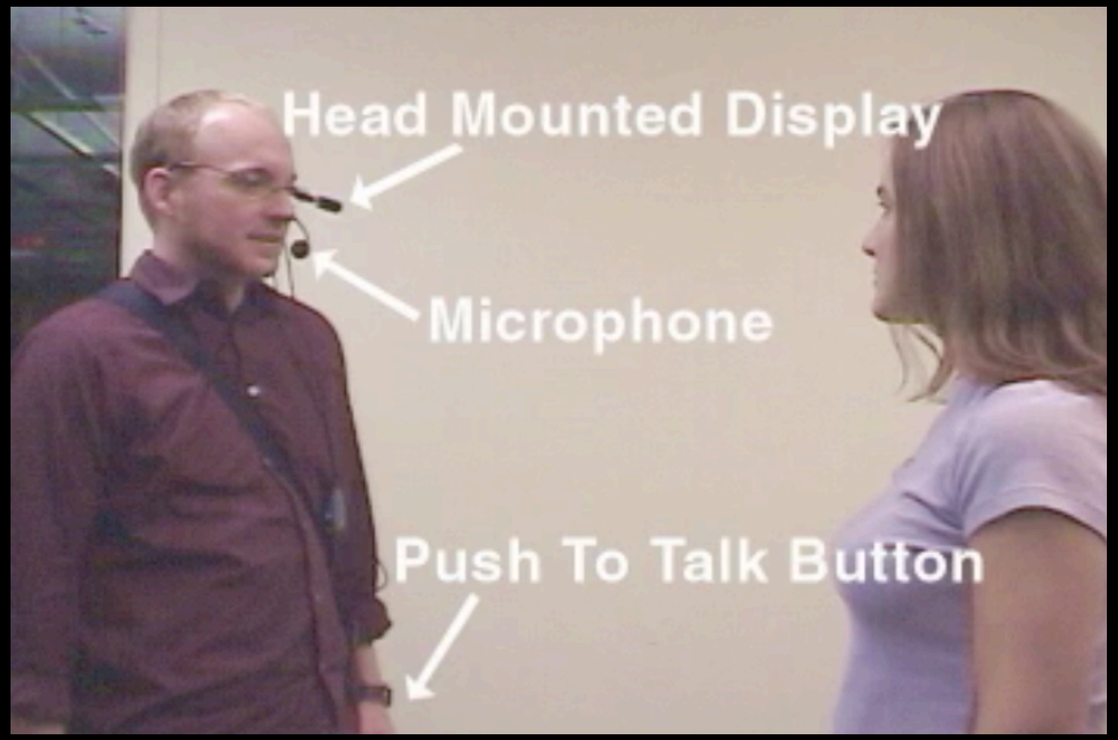
- Context (space, conversation)
- Personal assistant
- Human conversation (no delimiter)
- Summary questions
- Local variables (Jill)
- Errors, partial knowledge

Considering use context

- We should not just accept any speech recognizer, but instead tailor it to the context
- Customizing for the context
  - Kitchen vs. operating room?
  - Home vs. mobile/wearable?

Dual purpose speech

- Taking advantage of mobile context & display



Designing voice UIs

Some notes

- A good way to learn this is to look at translating from a GUI to a Voice UI
  - What do we get “for free” visually?
  - How can we simplify interactions?
  - How to create a fundamental set of actions?
- Dev tools still in an early stage
  - Opportunity to do better

Why is speech challenging?

adapted from Schnelle and Lyardet

- Speech is **one-dimensional**
- Speech is **transient**
- Speech is **invisible**
- Speech is **asymmetric**
- Flexibility vs. accuracy

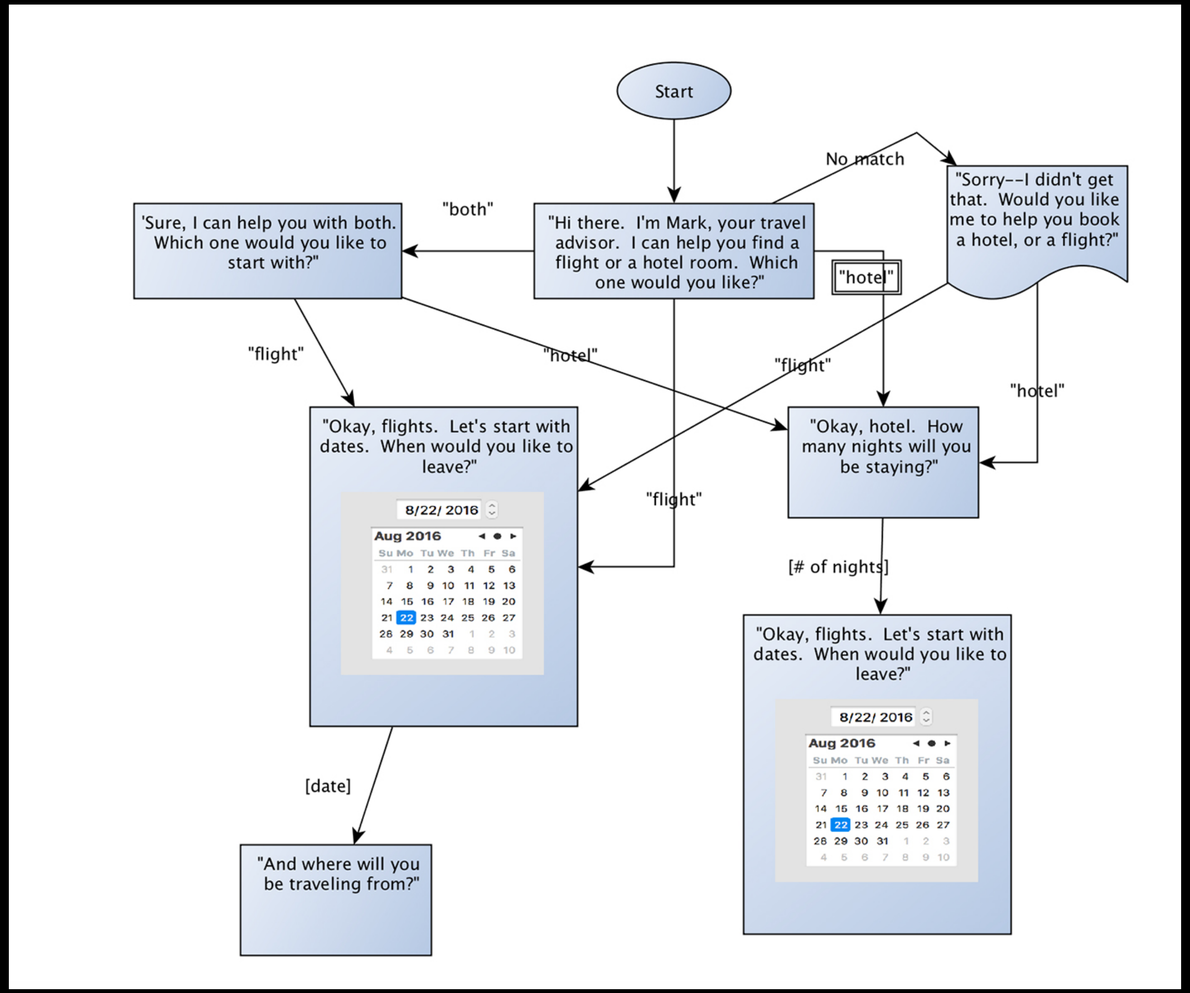
Why is speech challenging?

adapted from Schnelle and Lyardet

- **One-dimensional.** Not easy to skim speech, have to listen through it
- **Transient.** Spoken information requires short- or long-term memory
- **Invisible.** Possible actions are not clear
- **Asymmetric.** Faster to speak than type; slower to listen than read
- **Flexibility vs. accuracy.** Classic HCI tradeoff!

How to design voice UIs

- Most current systems involve some sort of **pattern matching** and **entity extraction**
- Pattern: “turn on the <name_of_light>”
- Dialog tree: model possible inputs and conversation paths



Wizard of Oz prototyping

- Person acts out what the computer would do
- Can use this as a prototyping tool, **but we have to follow the rules we ourselves set**
- Is our interaction logic robust enough for the real world?

Let’s try it

- Let’s model the dialog tree for a smart thermostat
- Assume we have working speech recognition, entity extraction
- We’ll design it together

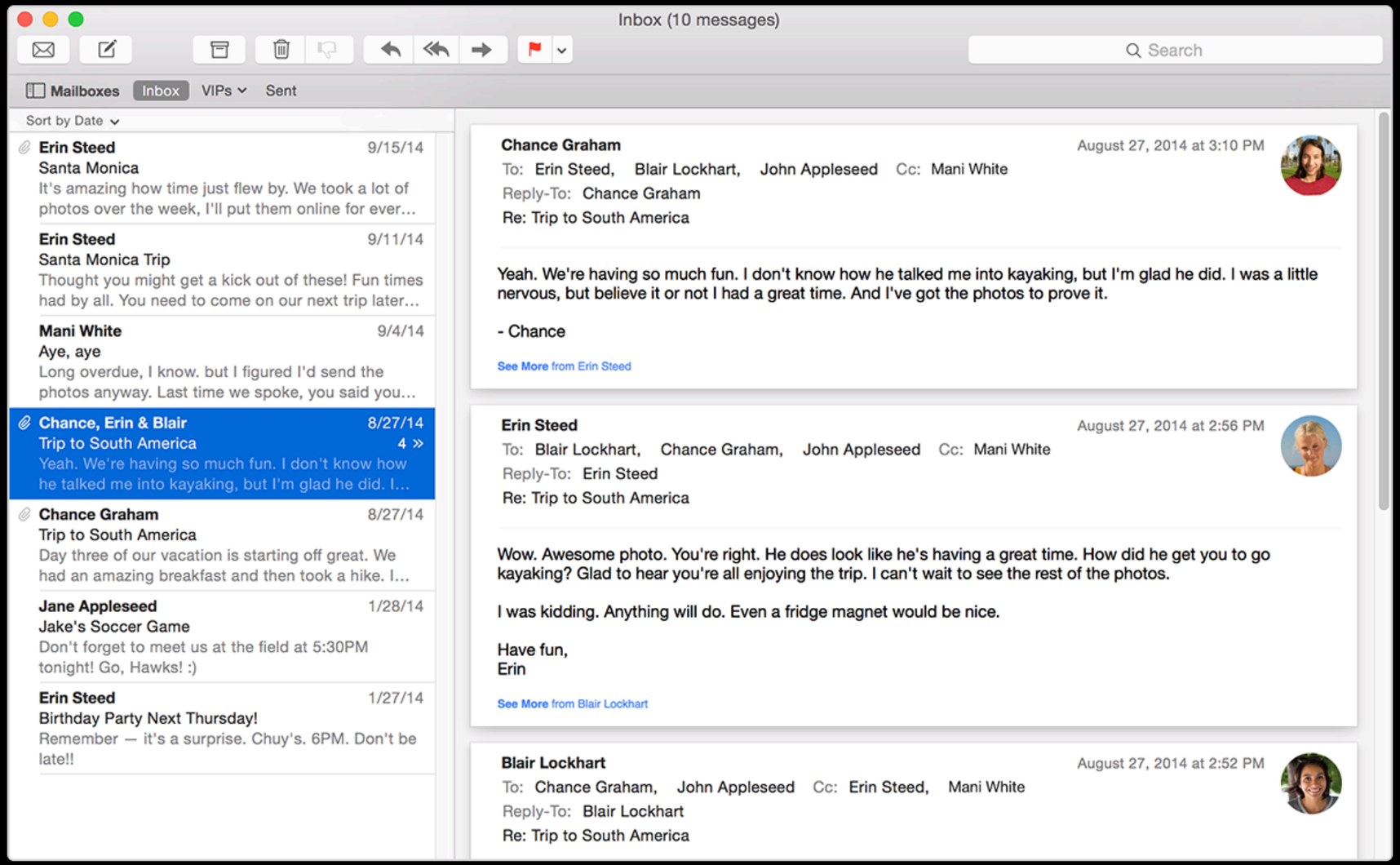
Example WOz fail

- “What time is the **<meeting_name>**?”
- What inputs will break this?

Designing “natural” voice UIs

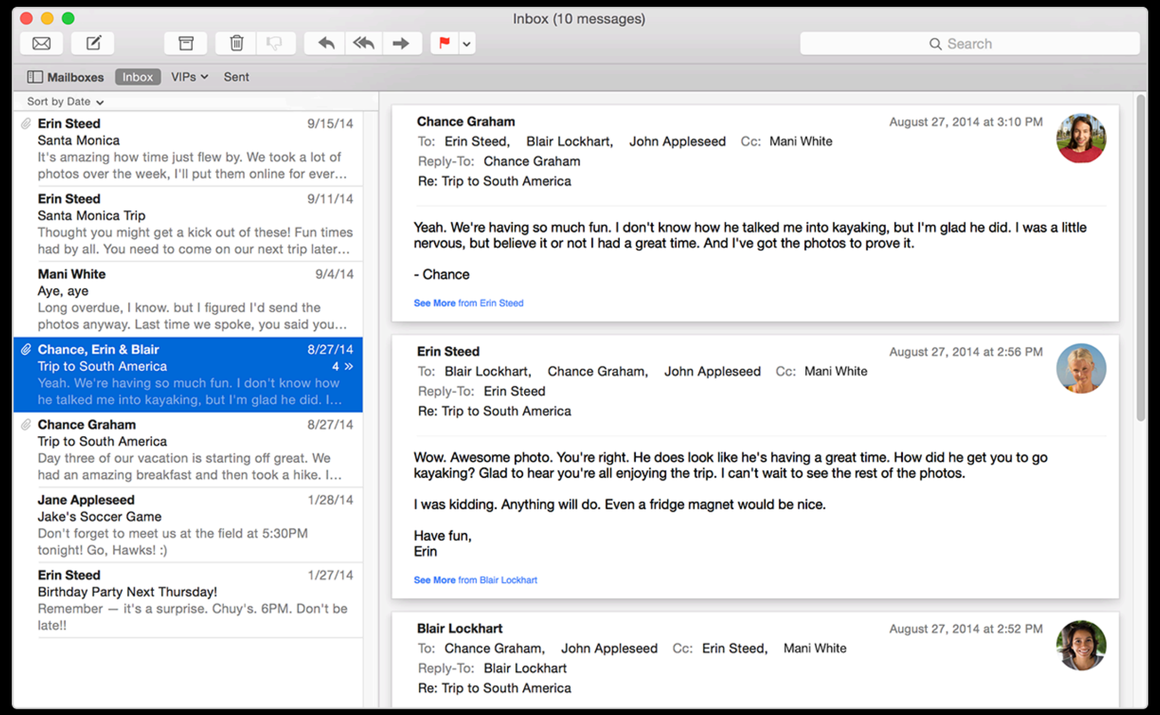
1. Identify use cases, *intents*
2. Identify use cases that we get “for free” with a graphical user interface, and forgot about
3. Work out the details of each intent (including edge cases)
4. Design output (verbosity, controls to navigate output)
5. Test and iterate

Example: an email voice client



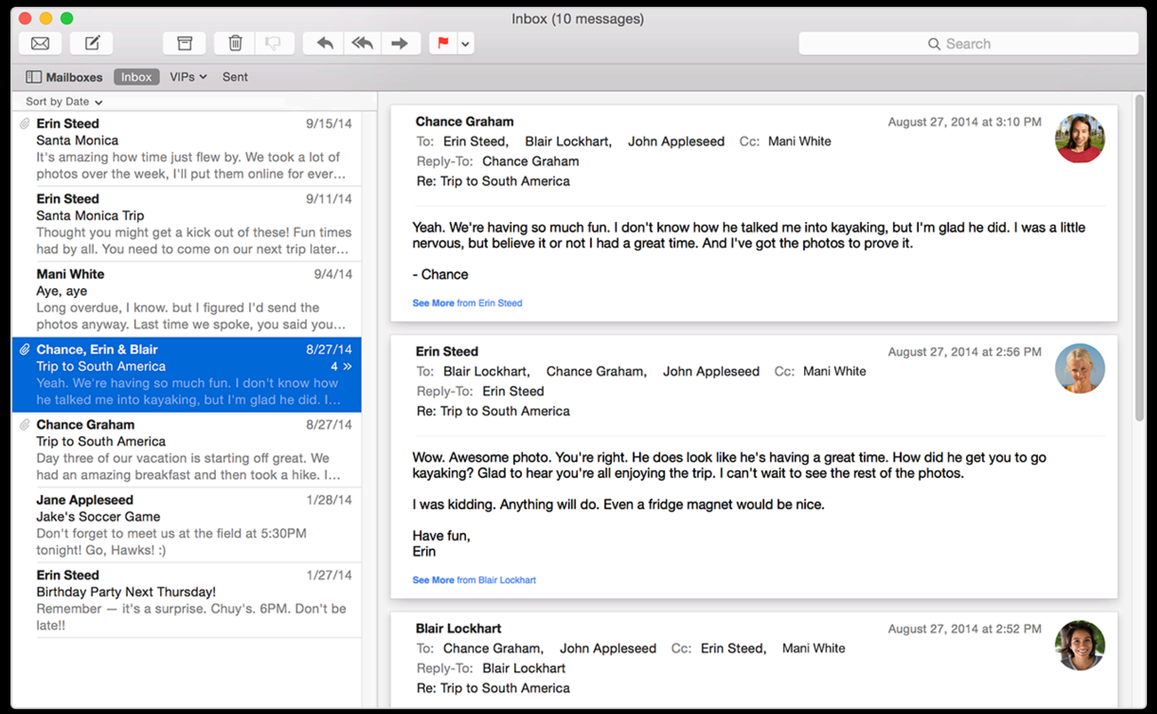
1. Identify intents

- Open/read
- Compose
- Navigate
- Delete
- Reply



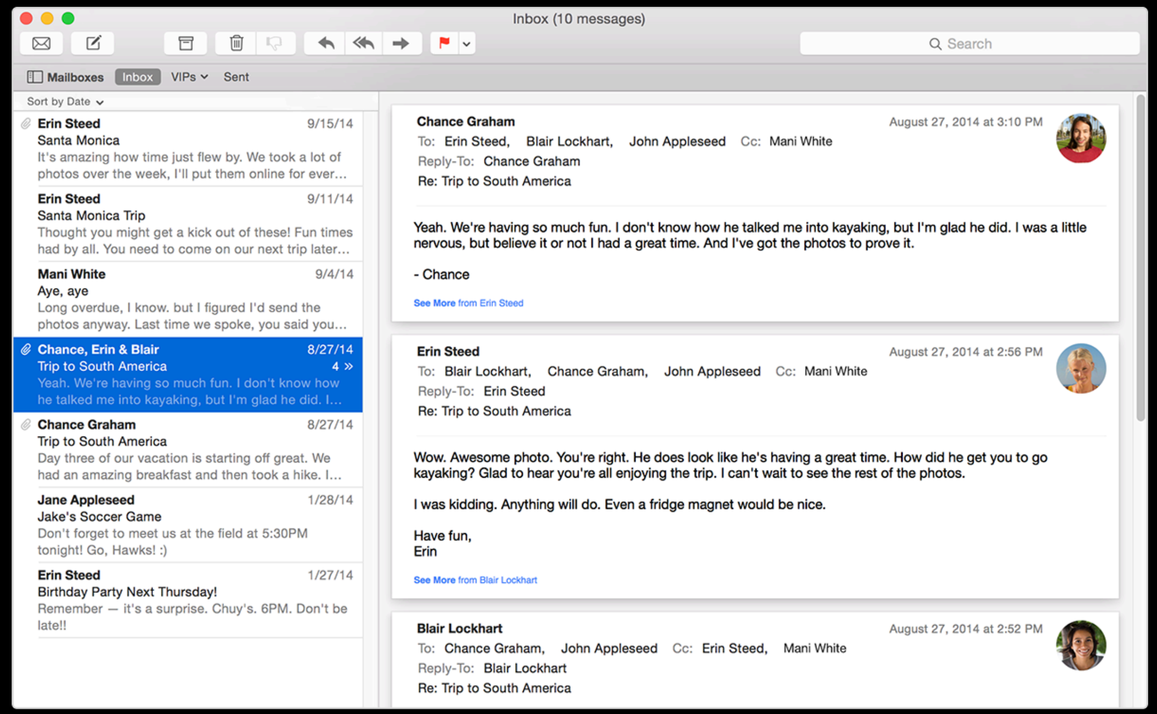
2. Identify visual tasks

- “Do I have an email from grandma?”
- “Delete all emails from grandma”



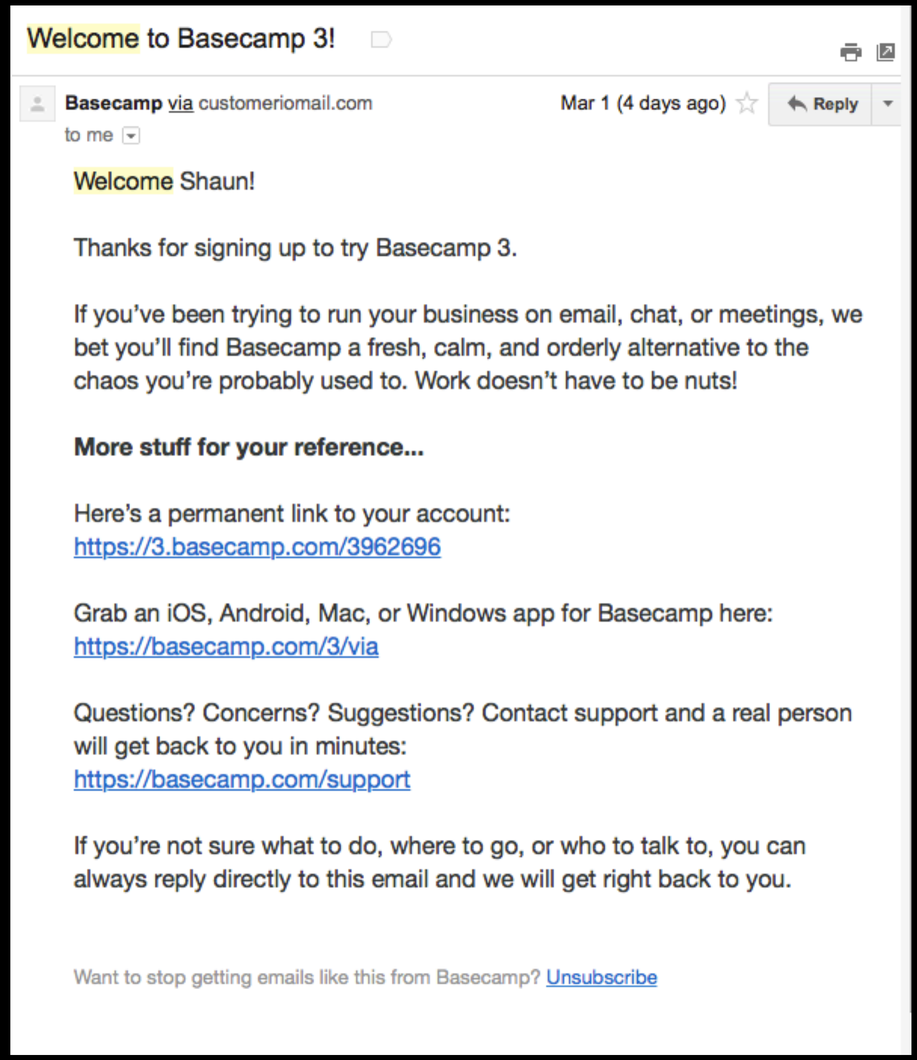
3. Work out logic for each intent

- For now, let’s design “compose”



4. Design output

- What to read?
- What to ignore?
- What navigation to provide?



Welcome to Basecamp 3!

Basecamp via customeriomail.com
to me

Mar 1 (4 days ago) Reply

Welcome Shaun!

Thanks for signing up to try Basecamp 3.

If you've been trying to run your business on email, chat, or meetings, we bet you'll find Basecamp a fresh, calm, and orderly alternative to the chaos you're probably used to. Work doesn't have to be nuts!

More stuff for your reference...

Here's a permanent link to your account:
https://3.basecamp.com/3962696

Grab an iOS, Android, Mac, or Windows app for Basecamp here:
https://basecamp.com/3/via

Questions? Concerns? Suggestions? Contact support and a real person will get back to you in minutes:
https://basecamp.com/support

If you're not sure what to do, where to go, or who to talk to, you can always reply directly to this email and we will get right back to you.

Want to stop getting emails like this from Basecamp? Unsubscribe

What we didn’t cover

- Advanced composing options (reply all, forward)
- Address book lookup
- Formatting
- Organizing messages into tags and folders
- …

Design guidelines

- Questions may be formulated in a variety of ways
  - Can use dialogs to fill in missing variables
- User must remember what she just heard (short-term), must remember what the system can do (long-term)
- Provide clear feedback:
  - System is listening, system has heard your command, system has performed an action
- Balance verbosity with clarity about what the system is expecting
- Provide help and guidance

It can get complicated…

- Using pronouns (and keeping them across queries)
  - “What is the capitol of Germany?”
  - “How many people live there?”
- Bilingual queries
  - “Siri, play Los Lobos”
- Domain-specific language changes meaning
  - “Play *Nevermind*”

Some unsolved problems

- Expressivity of synthesized speech
- Identifying speakers (although this is improving)
- Understanding user’s context
- User privacy

More on voice UIs

Today

- Accessibility of voice interfaces
- Intro to DialogFlow
- Begin Project 2

Accessibility of speech UIs

- Can provide a usable interface for people with various disabilities
- Opportunities to automate physical tasks

"Accessibility Came by Accident": Use of Voice-Controlled Intelligent Personal Assistants by People with Disabilities

Alisha Pradhan[1], Kanika Mehta[1], Leah Findlater[2]

[1]College of Information Studies
University of Maryland, College Park
alisha93@terpmail.umd.edu, mkanika@umd.edu

[2]Human Centered Design and Engineering
University of Washington
leahkf@uw.edu

ABSTRACT

From an accessibility perspective, voice-controlled, home-based intelligent personal assistants (IPAs) have the potential to greatly expand speech interaction beyond dictation and screen reader output. To examine the accessibility of off-the-shelf IPAs (*e.g.*, Amazon Echo) and to understand how users with disabilities are making use of these devices, we conducted two exploratory studies. The first, broader study is a content analysis of 346 Amazon Echo reviews that include users with disabilities, while the second study more specifically focuses on users with visual impairments, through interviews with 16 current users of home-based IPAs. Findings show that, although some accessibility challenges exist, users with a range of disabilities are using the Amazon Echo, including for unexpected cases such as speech therapy and support for caregivers. Richer voice-based applications and solutions to support discoverability would be particularly useful to users with visual impairments. These findings should inform future work on accessible voice-based IPAs.

and output beyond the traditional confines of text dictation and screen reader software. A person with limited mobility, for example, can control their home's lighting or door locks by voice, while a blind user can ask for the time or weather. Due to their relatively recent introduction, however, researchers have only begun to understand how these devices are being used by the general population (*e.g.*, [16,32]), much less by users with disabilities. As such, our focus is to address exploratory questions such as: To what extent are off-the-shelf IPAs, which were not necessarily designed with accessibility in mind, accessible? How are people with disabilities making use of them? What design opportunities do these devices offer to further support everyday activities for users with disabilities?

To answer these questions, we conducted two studies. The first study broadly examined use of IPAs by people with disabilities, by collecting and analyzing online customer reviews of the Amazon Echo, a popular IPA, and its offshoots, the Echo Dot and Tap. We identified 346 reviews that described use of the device by a person with a

Los Angeles, Feb 25, 2019

Cedars-Sinai Taps Alexa for Smart Hospital Room Pilot

Cedars-Sinai patient John Gooch said his voice activated assistant is "super cool." He uses it to listen to music and contact his nurses. Photo by Cedars-Sinai.

A pilot program underway in more than 100 patient rooms at Cedars-Sinai is allowing patients to use an Alexa-powered platform known as Aiva to interact hands-free with nurses and control their entertainment. Aiva is the world's first patient-centered voice assistant platform for hospitals.

In the pilot project, patient rooms are equipped with Amazon Echos and patients simply tell the device what they need. For example, patients can turn their TV off and on and change channels by giving verbal commands like, "Alexa, change the channel to ESPN." A patient who needs assistance getting out of bed might say, "Alexa, tell my nurse I need to get up to use the restroom."

Accessibility challenges

- Lack of feedback (audio and/or visual)
- Discoverability
- Issues with speech quality
  - Dysarthric speech (stroke, cerebral palsy, others)

Voice agents vs. screen readers

- How are voice agents different than screen readers?

Voice agents vs. screen readers

	Voice Agents	Screen readers
Users	Untrained	Trained
Input	Speech	Keyboard
Visual UI as fallback?	Maybe	Probably not
Interactivity	Low	High
Output	Unstructured	Structured
Speech rate	Low	High
Voice	Natural	Optimized for efficiency

Shared challenges: voice agents & screen readers

- Difficulty of presenting concepts (especially visual concepts) as speech
- Difficulty in providing efficient and navigable audio output
- Difficult to understand how to best convert visual interactions to audio

Prototyping in Dialogflow

What is Dialogflow?

- Web service from Google that supports the creation of conversational agents/bots
- Provides Natural Language Processing (NLP) features for identifying the user's intent, understanding multiple variations of a command
- Supports deployment to Facebook Messenger, Google Assistant, Alexa

Dialogflow interaction model

- In Dialogflow, you can model your conversation structure
- Dialogflow uses **webhooks** to send the user's intent to your web app
  (what the user intended, not what they said)
- Your app then returns a result to Dialogflow
- Dialogflow reads the appropriate response to the user via their preferred method

Why use Dialogflow?

- NLP is hard; this provides robust support for conversational interaction
- Your code only needs to focus on the actions taken by the bot, not what users say
- Easy deployment to different platforms
- Very fast way to build prototypes, that can later be used for the real apps

What we will do

- Build conversation structure using Dialogflow **intents**
- Prototype state using **contexts**
- Deploy our bot to the web and a voice agent

Dialogflow basics

- Intents, parameters, entities, contexts

Intents

- Possible responses to a user request
  - One (and only one) intent is always triggered
  - Chosen based on matching of the request syntax, and **context**
  - May include followup requests if some parameters are missing
- Generates a response (chosen from a list of possible responses)

Parameters

- Represent data in user requests and responses
- Each intent specifies required and optional parameters
  - An intent can prompt for missing parameters
- Can refer to parameters inside our responses
  “Hello $name”
- By default, parameters aren’t saved between requests - we need **contexts** for that

Entities

- These are essentially variable types
- Some built in (@sys.number, @sys.any)
- But we can specify our own
  - Example: majors (“computer science”, “applied math”, …)

Contexts

- Represent state of the current conversation
- Last for a set number of interactions (unless we cancel or overwrite them)
- **Output context:** created after an intent is executed; has a developer-specified lifespan
- **Input context:** Determines what state is needed for an intent to fire
- Think about these as boolean variables

Variables within contexts

- We can also use contexts to pass parameters across intents
- Each parameter value is stored in the output context
- Can reference them later using #contextName.parameterName

What is difficult about Dialogflow?

- We have to rely on the built-in intent matching and entity extraction
  - But we can add more training data, and test
- Can’t include conditional behaviors in intents
  - Instead, we can add **input contexts** into our intents
  - Intent only fires if the matching input context exists

Conditional behaviors

- The best way to add conditional behaviors is to use input contexts
- Example: lightbulb bot
  - “Turn on” intent sets the output context **light-on**
  - “Turn off” intent sets the output context **light-off** (and cancels light-on)
  - “Is the light on?” is then two intents: one requires light-on and one requires light-off

Examples

- LightbulbBot - using contexts to track state; conditional intents
- FanBot - entities; storing variables across requests using contexts

What we'll do

- Prototype a smart appliance
- 45 minutes development; then Wizard of Oz testing
  - Or use Google Assistant app

Testing

- Write down the name of your appliance on the board
- For test users: tell them what the appliance is, but not what the commands are
  - Did you include the right intents?
  - Did you include enough test phrases for each intent?
  - Does your app correctly track state?