IN4MATX 133: User Interface Software

Lecture 11:
Separation in Angular &
Modeling human performance

Announcements

- After today's lecture, you should have everything you need to complete A3
- Get started on it! There's some good discussion on Slack already

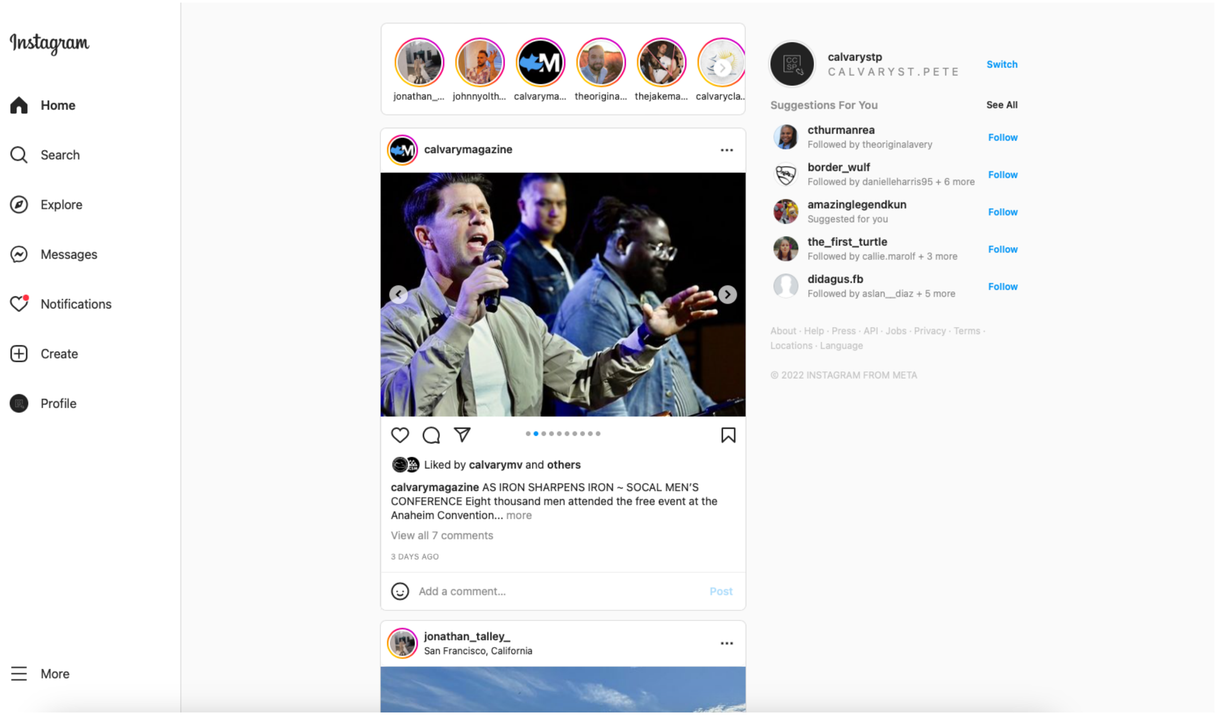
Today's goals

By the end of today, you should be able to…

- Differentiate and explain the roles of Angular components, modules, and services
- Implement a service in Angular
- Navigate Angular's file structure
- Describe the major components of Fitts's Law
- Explain how Fitts's Law impacts how interfaces should be designed
- Describe approaches for correcting systematic errors in touch performance

A “large” client interface

- Hundreds of pages and ways to navigate between pages
- Repeated UI elements (status updates)
  - Angular implements these as *components*
- Different content, links, etc. displayed for each person



A “large” client interface

- Loading lots of libraries can be slow and expensive
- So Angular supports importing the components you need

“Standalone” components

- You can import a component within another component
- New in Angular 19 (2024), previously there were modules to essentially “wrap” components

Root component

app.component.ts

Import HeaderComponent and FooterComponent

```
import { Component } from '@angular/core';
import { FooterComponent } from ‘.footer/footer.component’;
import { HeaderComponent } from ‘.header/header.component’;

@Component({
imports: [HeaderComponent, FooterComponent],
selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

Allows this component to use HeaderComponent and FooterComponent

Angular routing

- Angular maintains a list of “routes”
  - Routing: defines what URIs to send to what endpoints
  - For Angular, defines what URIs to send to what components

Angular routing

app-routing.module.ts (or app.routes.ts in newer Angular)

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { ArtistPageComponent } from './pages/artist-page/artist-page.component';
import { TrackPageComponent } from './pages/track-page/track-page.component';
import { AlbumPageComponent } from './pages/album-page/album-page.component';
import { HomePageComponent } from './pages/home-page/home-page.component';

const routes: Routes = [
  { path: 'artist/:id', component: ArtistPageComponent},
  { path: 'track/:id', component: TrackPageComponent},
  { path: 'album/:id', component: AlbumPageComponent},
  { path: '', component: HomePageComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Listens for any endpoint `artist/:id`

`id` can be retrieved in `album-page.component.ts`

Retrieving route in a component

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-album-page',
  templateUrl: './album-page.component.html',
  styleUrls: ['./album-page.component.css']
})
export class AlbumPageComponent implements OnInit {

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
    var albumId = this.route.snapshot.paramMap.get('id');
  }

}
```

"Injecting a service"

Retrieve the id from the URI

Angular services

- Anything not associated with a specific view should be turned into a *service*
  - e.g., getting data from an API, parsing URIs for routing information
- Helps keep components lightweight
- Services can then be *injected* into a component (importing)
- To inject, import the service and retrieve it as a parameter in the constructor
- `ng generate service [name]`

Angular services

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-album-page',
  templateUrl: './album-page.component.html',
  styleUrls: ['./album-page.component.css']
})
export class AlbumPageComponent implements OnInit {

  constructor(private route: ActivatedRoute) { }

  ngOnInit() {
    var albumId = this.route.snapshot.paramMap.get('id');
  }
}
```

Importing a service

Injecting it

Service can be referenced later

Angular services

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class SpotifyService {
 baseUrl:string = 'http://localhost:8888';

  constructor(private http:HttpClient) { }

  private sendRequestToExpress(endpoint:string) {
  }
}
```

Defined as injectable

Services can inject other services!

What module(s) can use this service

HttpClient injected

Import a custom service

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { SpotifyService } from '../../services/spotify.service';

@Component({
  selector: 'app-album-page',
  templateUrl: './album-page.component.html',
  styleUrls: ['./album-page.component.css']
})
export class AlbumPageComponent implements OnInit {

  constructor(private route: ActivatedRoute,
private spotifyService:SpotifyService) { }
```

Import service via file structure

Inject it like any other service

Angular classes

- Plain-old classes can also be made in Angular
  - Any processing or munging you need to do, for example

- `ng generate class [name]`

```
export class Dataparser {
  public constructor() {
    console.log('Hello, world!');
  }
}
```

Import a class

```
import { Component, OnInit, Input } from '@angular/core';
import { Dataparser } from '../dataparser';

@Component({
  selector: 'app-day',
  templateUrl: './day.component.html',
  styleUrls: ['./day.component.css']
})
export class DayComponent implements OnInit {
  @Input() today:string;

  days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];

  constructor() {
    var data = new Dataparser();
  }

  ngOnInit() {
  }
}
```

Import class via file structure

Instantiate it like any other class

Import a library

- Since Angular is in TypeScript, it can use any JavaScript or TypeScript library
- Install as normal with npm: `npm install [packagename]`
  - If you want TypeScript typings, don't forget to install `@types/[packagename]`

Import a library

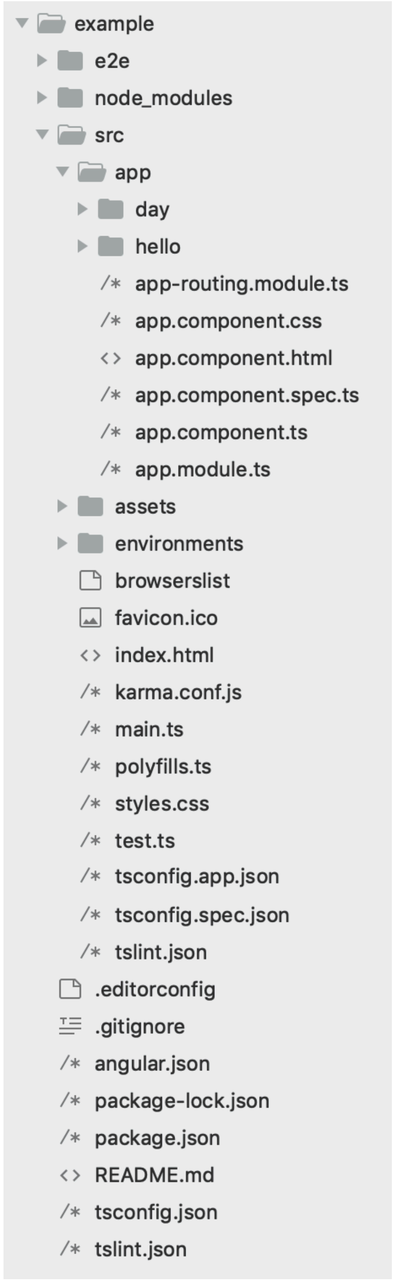
```
import chroma from 'chroma-js';

export class Dataparser {

  constructor() {
    console.log(chroma('royalblue')); //'#4169e1'
  }
}
```

Note: different syntax

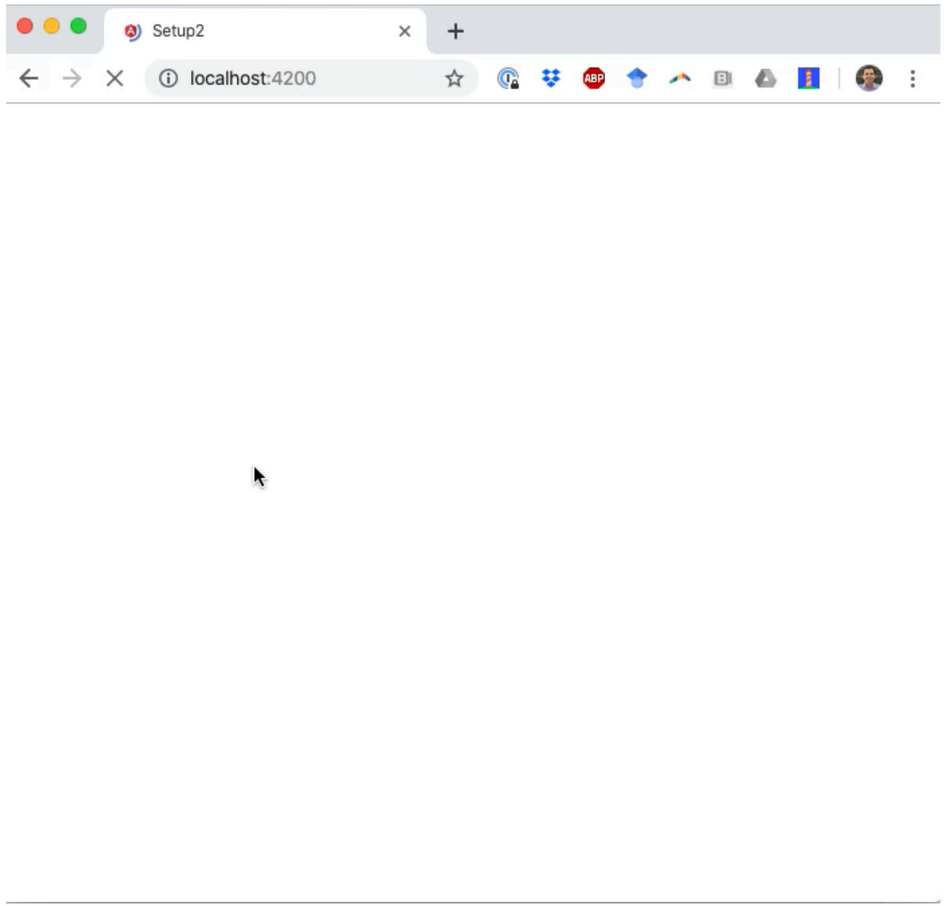
Can now be referenced

Angular's file structure

- Angular projects generate a *lot* of files
  - There are about 75 in the starter code for A3
- Most are boilerplate



Rolling Dice



Switching topics:
Modeling performance

Question

Which button would be faster to click on?

A. A

B. B

C. Roughly equal

D.

E.

Which button would be faster to click on?

A 0%

B 0%

Roughly equal 0%

Question

Which button would be faster to click on?

- A A
- B B
- C Roughly equal
- D
- E

Fitts’s Law (1954)

- Models time to acquire targets in aimed movement
  - Reaching for control in a cockpit
  - Moving across a dashboard
  - Pulling defective items from a conveyor belt
  - Clicking on icons using a mouse

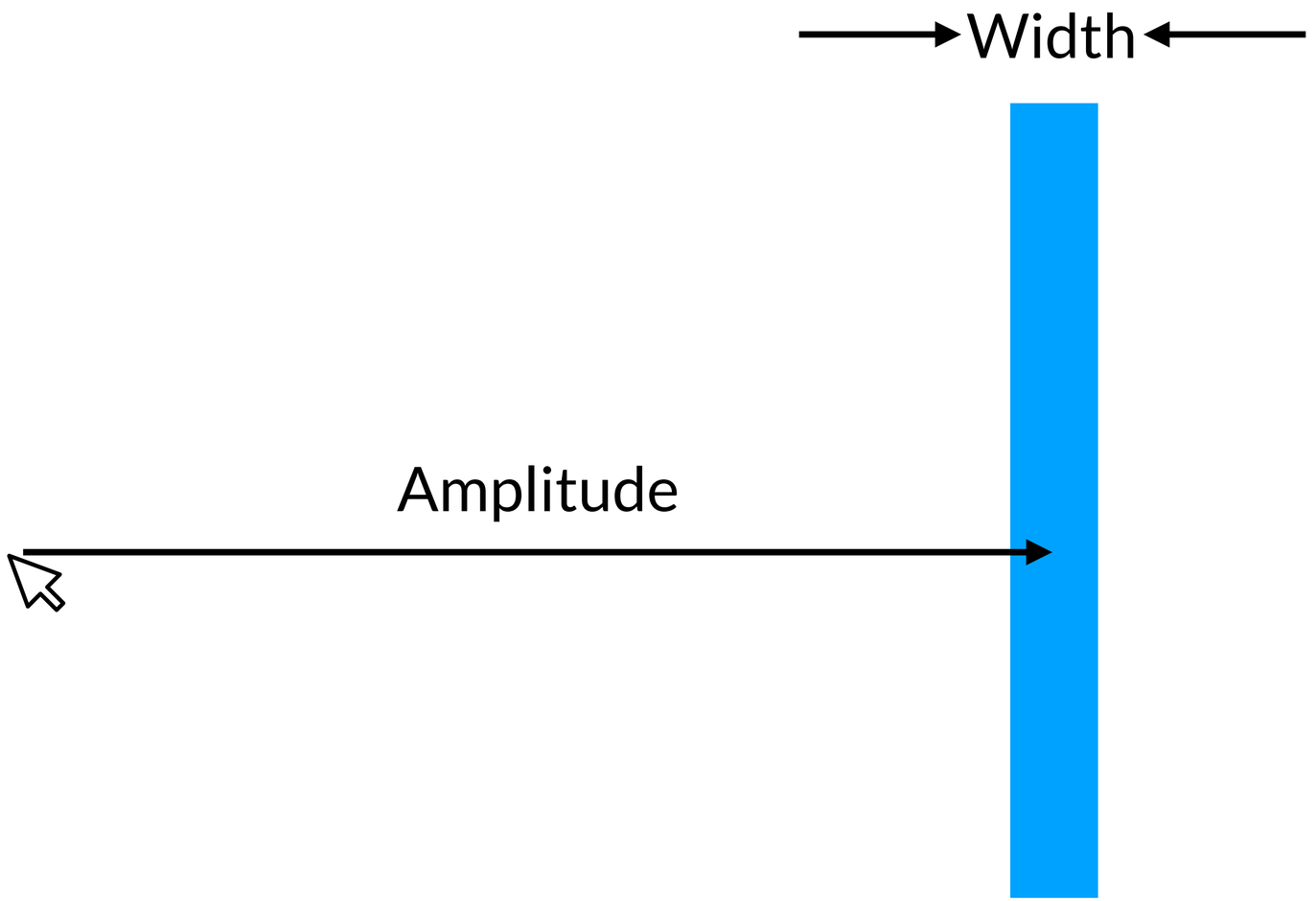
https://en.wikipedia.org/wiki/Fitts%27s_law

Fitts’s Law (1954)

- Very powerful, widely used
  - Holds for many circumstances (e.g., under water)
  - Allows for comparison among different experiments
  - Used both to measure and predict

https://en.wikipedia.org/wiki/Fitts%27s_law

Point-select task



Fitts’s Law

- MT = a + b log2(A / W + 1)
  - What kind of equation does this look like?

Fitts's Law

- MT = a + b log2(A / W + 1)
  - What kind of equation does this look like?
- y = mx + b
- MT = a + bx, where x = log2(A / W + 1)
  - x is called the Index of Difficulty (ID)
  - As "A" goes up, ID goes up
  - As "W" goes up, ID goes down

Movement Time (MT)

- MT = a + b log2(A / W + 1)
- Time, in seconds, to acquire the target (e.g., click on the button)

Index of Difficulty (ID)

- log2(A / W + 1)
  - Fitts's Law claims that the time to acquire a target increases linearly with the log of the ratio of the movement distance or amplitude (A) to target width (W)

Index of Difficulty (ID)

- log2(A / W + 1)
  - Fitts's Law claims that the time to acquire a target increases linearly with the log of the ratio of the movement distance or amplitude (A) to target width (W)
- Why is it significant that it is a ratio?
  - Units of A and W don't matter
  - Allows comparison across experiments

Index of Difficulty (ID)

- log2(A / W + 1)
  - Fitts's Law claims that the time to acquire a target increases linearly with the log of the ratio of the movement distance or amplitude (A) to target width (W)
- ID units typically in "bits"
  - Because of association with information capacity and somewhat arbitrary use of base-2 logarithm

Index of Performance (IP)

- MT = a + b log2(A / W + 1)
  - b is slope
- 1/b is called Index of Performance (IP)
  - If MT is in seconds, IP is in bits/second
- Also called “throughput” or “bandwidth”
- **a and b depend on the input device**

Question 

Will a mouse or a touchpad have lower movement time (MT)?

A. A mouse

B. A touchpad

C. Roughly equal

D.

E.

Will a mouse or a touchpad have lower movement time (MT)?

A mouse 0%

A touchpad 0%

Roughly equal 0%

[Fitts’s law demo]

http://www.yorku.ca/mack/FittsLawSoftware/

“Beating” Fitts’s law

- It is the law, right?
  - MT = a + b log2(A/W + 1)
- So how can we reduce movement time?
  - Reduce amplitude (A)
  - Increase width (W)

“Beating” Fitts’s law

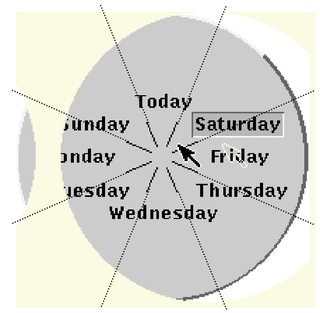
- Put targets closer together
- Make targets bigger
- Make cursor bigger
- Make impenetrable edges

Question

menus

(A) Pop-up Linear Menu

(B) Pop-up Pie Menu



Which menu will be faster on average?

A. A

B. B

C. Roughly equal

D.

E.

Which menu will be faster on average?

(A) Pop-up Linear Menu

(B) Pop-up Pie Menu

A 0%

B 0%

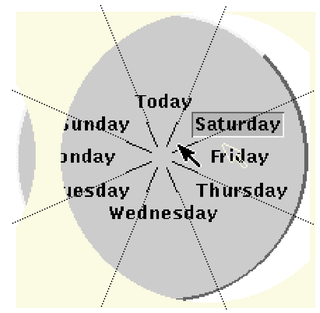
Roughly equal 0%

Question

menus

(A) Pop-up Linear Menu

(B) Pop-up Pie Menu



Which menu will be faster on average?

A. A

B. B

C. Roughly equal

D.

E.

Fitts’s Law in windowing

- Windows 95: missed by a pixel
- Windows XP: good to the end
- Corners and edges make great targets
  - Do not have to move precisely to trigger them
  - They have “infinite” width

Fitts’s Law in other domains

- How would Fitts’s Law apply to using touch input on a phone?
  - Shorter distances (smaller screen)
- All things being equal, movement times *should* be lower
  - Shorter distances, faster to move your finger than a mouse

Fitts’s Law in other domains

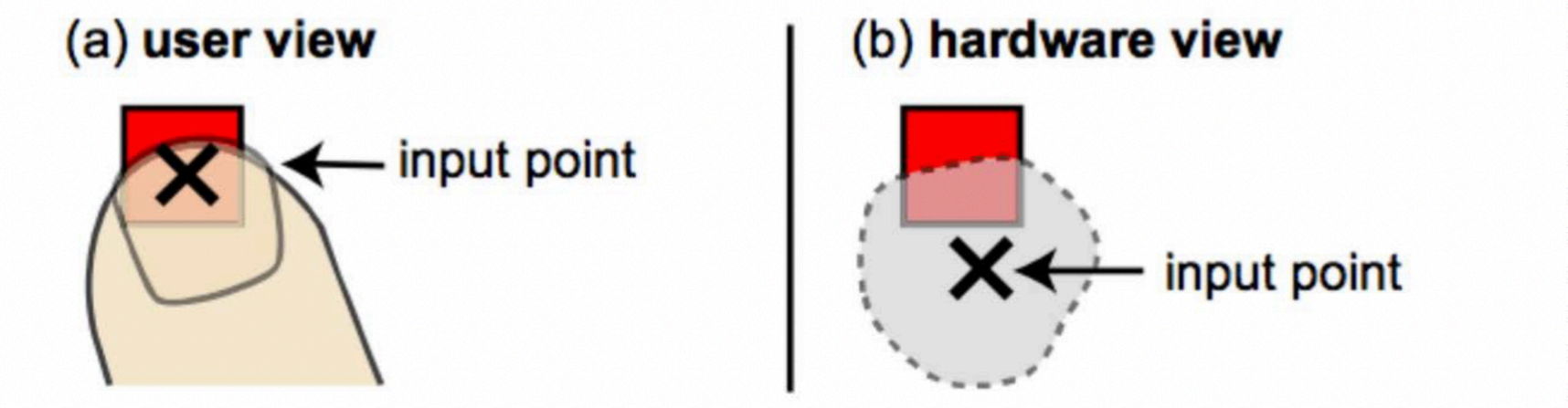
- But in practice, touchscreens on mobile tend not to be much faster
  - Buttons are smaller
  - People tend to be slower near the edges of touchscreens

Modeling input

Modeling mouse position

- Mouse pointer is relatively small
- We model it via X, Y position on the screen
- See whether that X, Y overlaps with a button, for example
  - Targets are usually large enough that “exact” position does not matter

Modeling touch position



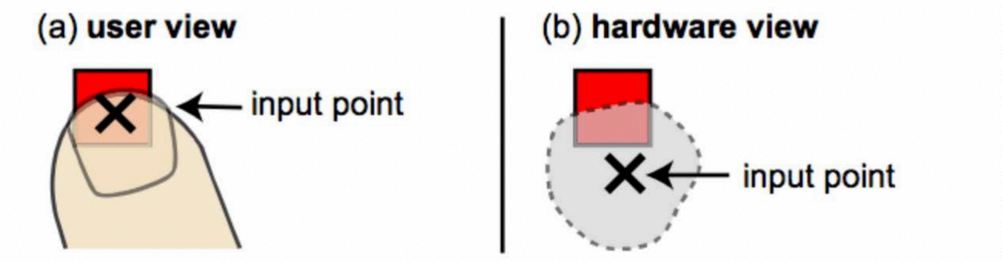
Daniel Vogel and Patrick Baudisch. 2007. Shift: a technique for operating pen-based interfaces using touch. CHI 2007. https://doi.org/10.1145/1240624.1240727

Modeling touch position

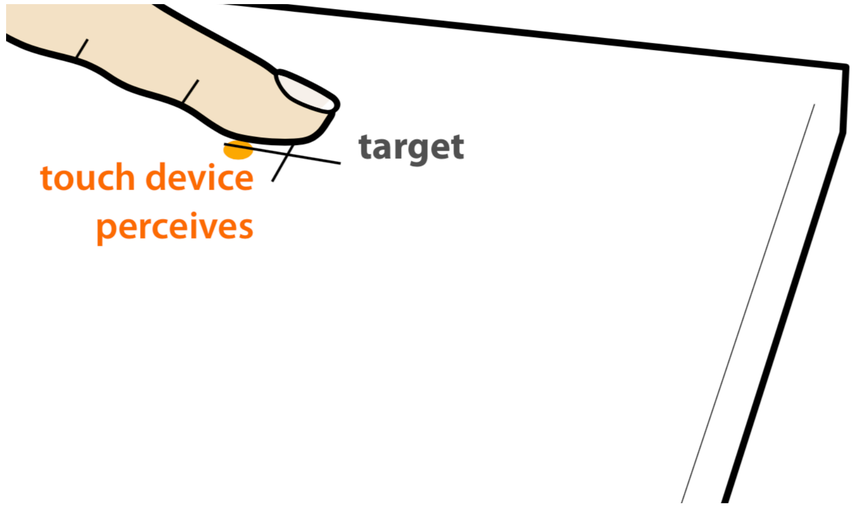
- One interpretation of the problem:
  our fingers are fat
  - We should use tiny styluses to make our our selection more accurate
- Another interpretation:
  our model of touch position is inaccurate
  - We should make our model better

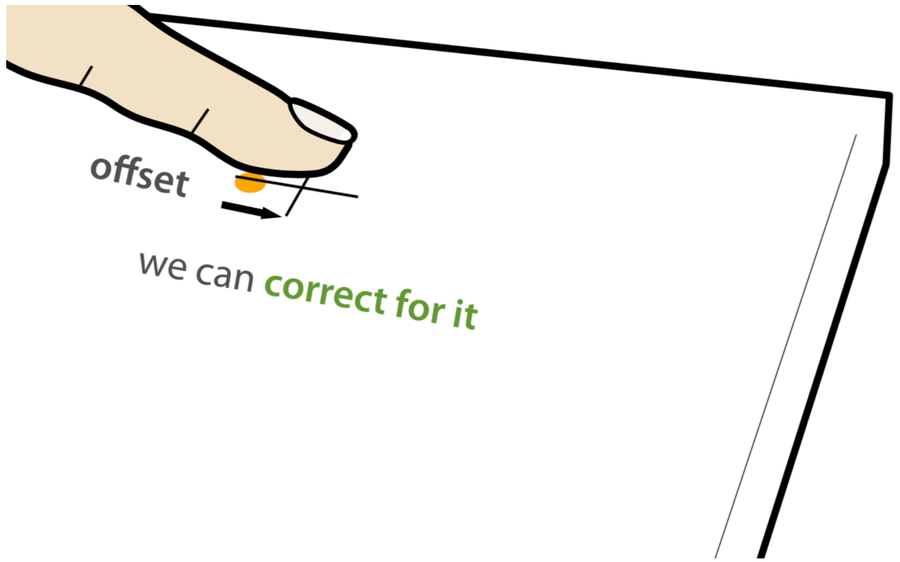
Modeling touch position

- How can we improve our model?
- Make the hardware view more closely match the user view



Modeling touch position

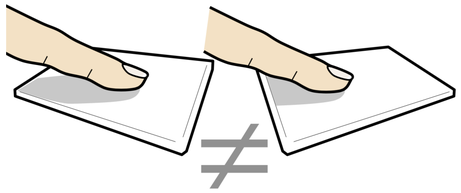




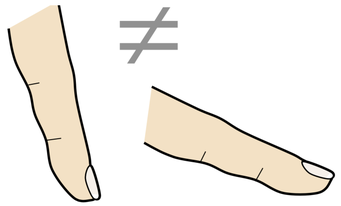
Christian Holz and Patrick Baudisch. 2011. Understanding touch. CHI 2011. https://doi.org/10.1145/1978942.1979308

Modeling touch position

- Hypothesis: yaw, pitch, and roll all impact touch position
  - Additionally, for each person, finger size/shape and mental model impact touch position



Yaw: angle of touch device



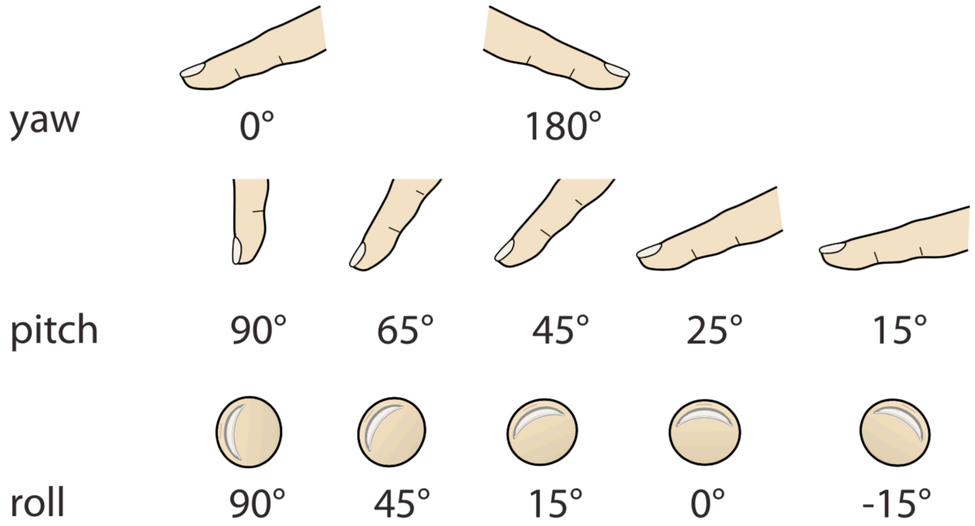
Pitch: angle of finger

Roll: rotation of finger

Christian Holz and Patrick Baudisch. 2011. Understanding touch. CHI 2011. https://doi.org/10.1145/1978942.1979308

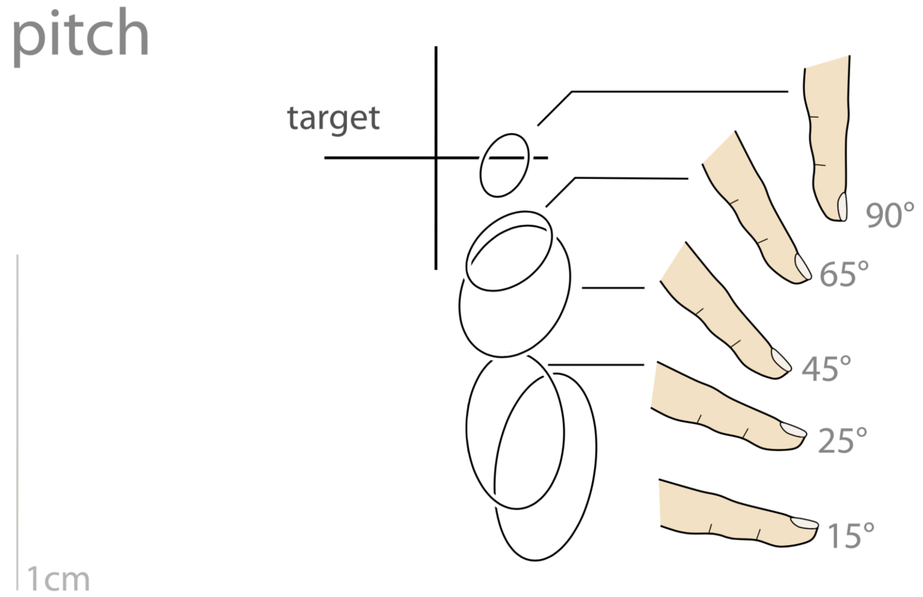
Modeling touch position

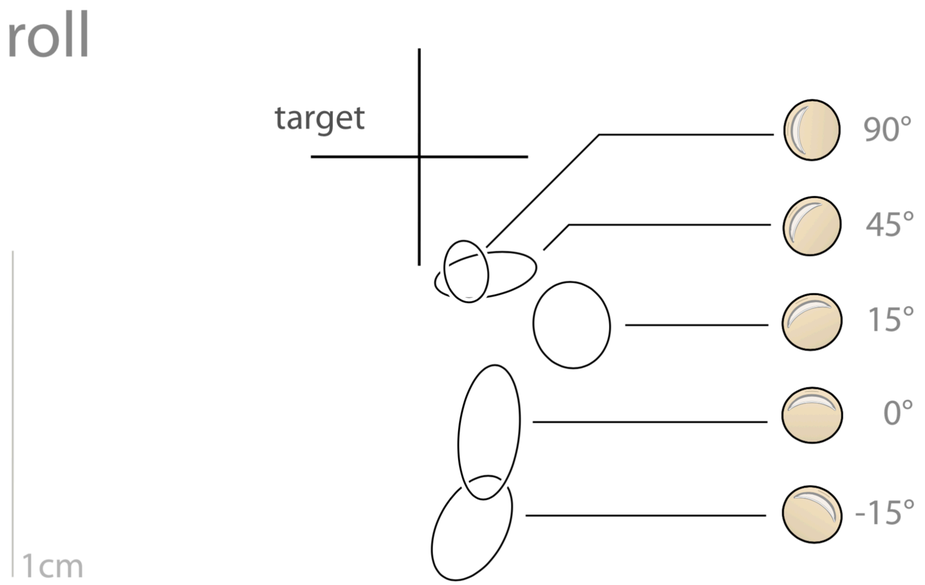
- Ran a study
  - 12 participants touched 600 points each
  - Varied yaw, pitch, and roll



Christian Holz and Patrick Baudisch. 2011. Understanding touch. CHI 2011. https://doi.org/10.1145/1978942.1979308

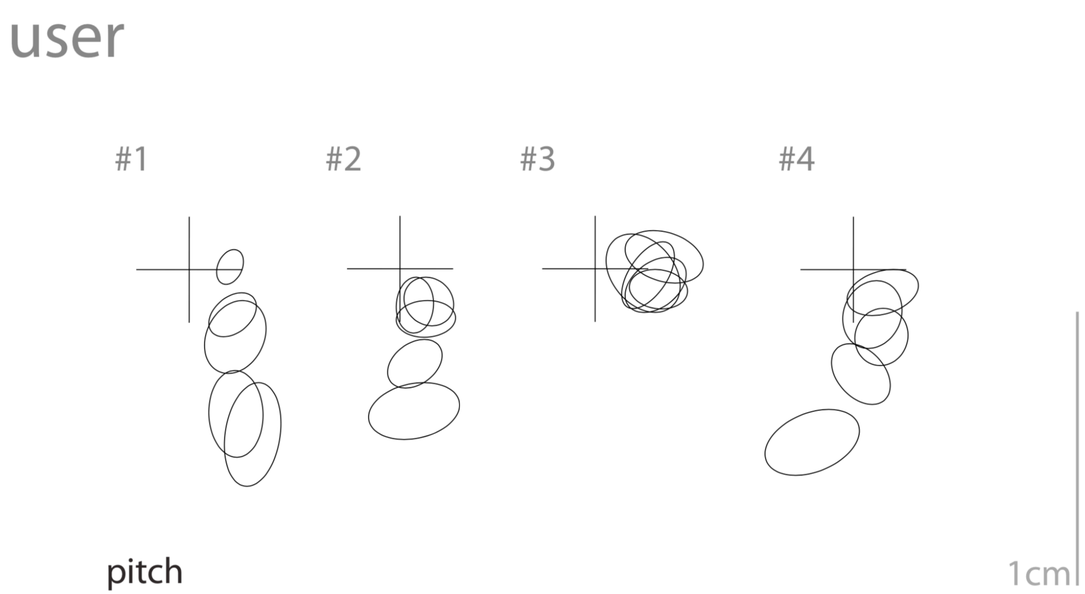
Modeling touch position





Christian Holz and Patrick Baudisch. 2011. Understanding touch. CHI 2011. https://doi.org/10.1145/1978942.1979308

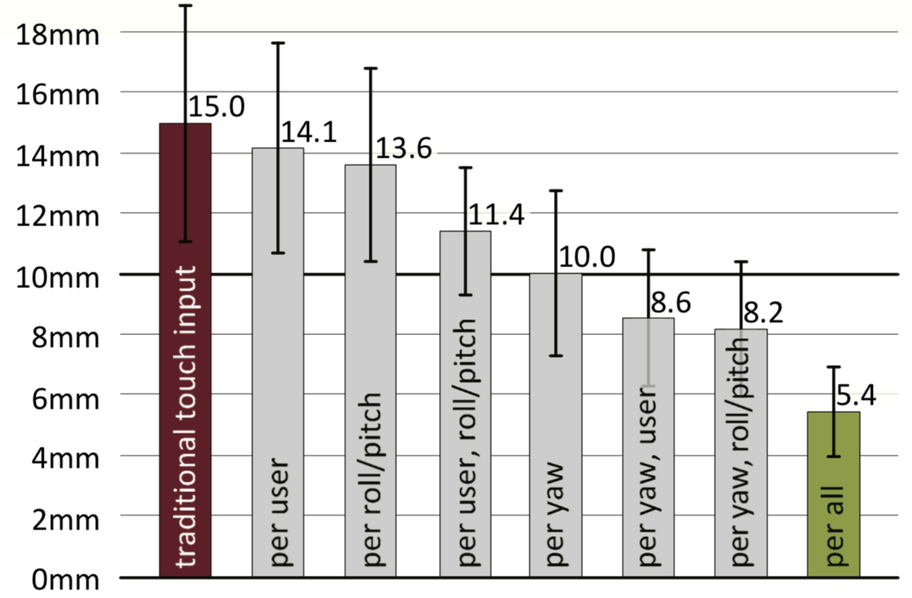
Modeling touch position



Christian Holz and Patrick Baudisch. 2011. Understanding touch. CHI 2011. https://doi.org/10.1145/1978942.1979308

Modeling touch position

minimum button size



Improving the model means that buttons can be **3x** smaller and not be any harder to click

Christian Holz and Patrick Baudisch. 2011. Understanding touch. CHI 2011. https://doi.org/10.1145/1978942.1979308

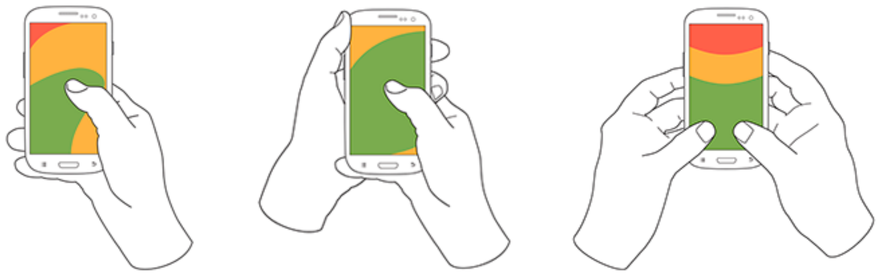
Modeling touch input

- Study was *very* controlled
  - Participant sat in a chair, the screen was on a desk
- How about the other ways that people use their phones?

Christian Holz and Patrick Baudisch. 2011. Understanding touch. CHI 2011. https://doi.org/10.1145/1978942.1979308

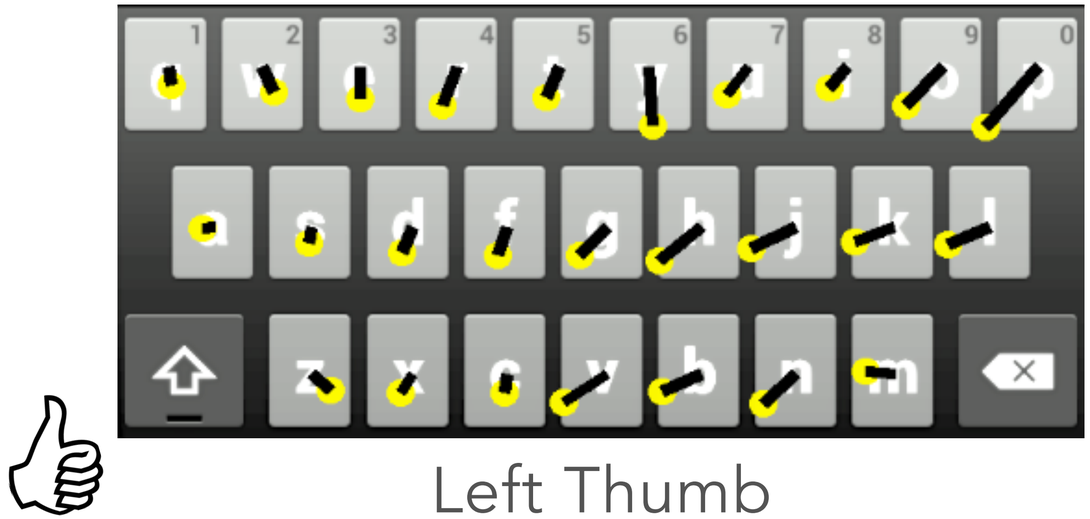
Modeling phone grip

- People grip their phones in different ways
- Grip changes with phone size, hand size
  - Situational changes (e.g., walking, holding something)
- Can we detect phone grip and update our model?



http://static.googleusercontent.com/media/www.google.com/en//intl/ALL_ALL/think/multiscreen/pdf/multi-screen-moblie-whitepaper_research-studies.pdf

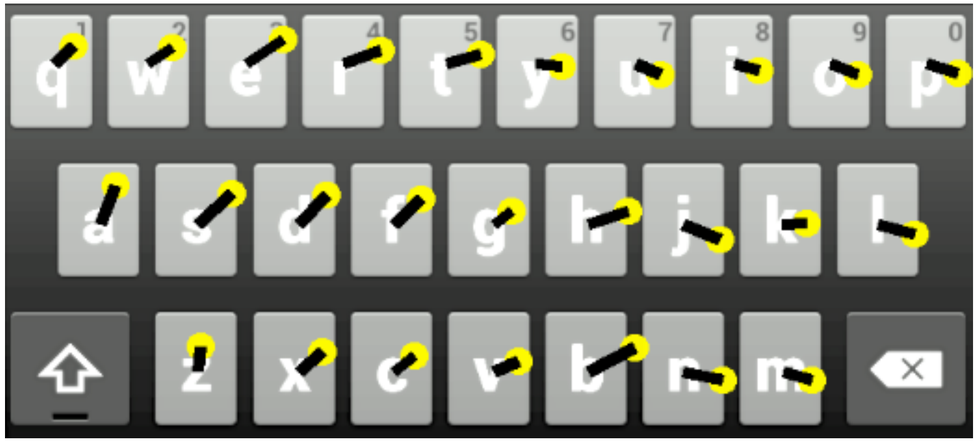
Modeling phone grip



Left Thumb

Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

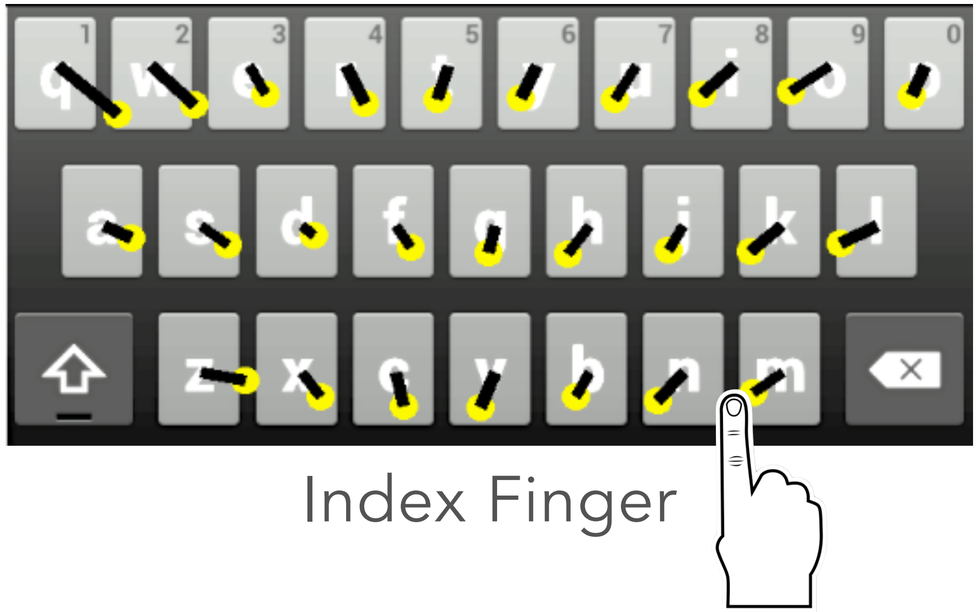
Modeling phone grip



Right Thumb

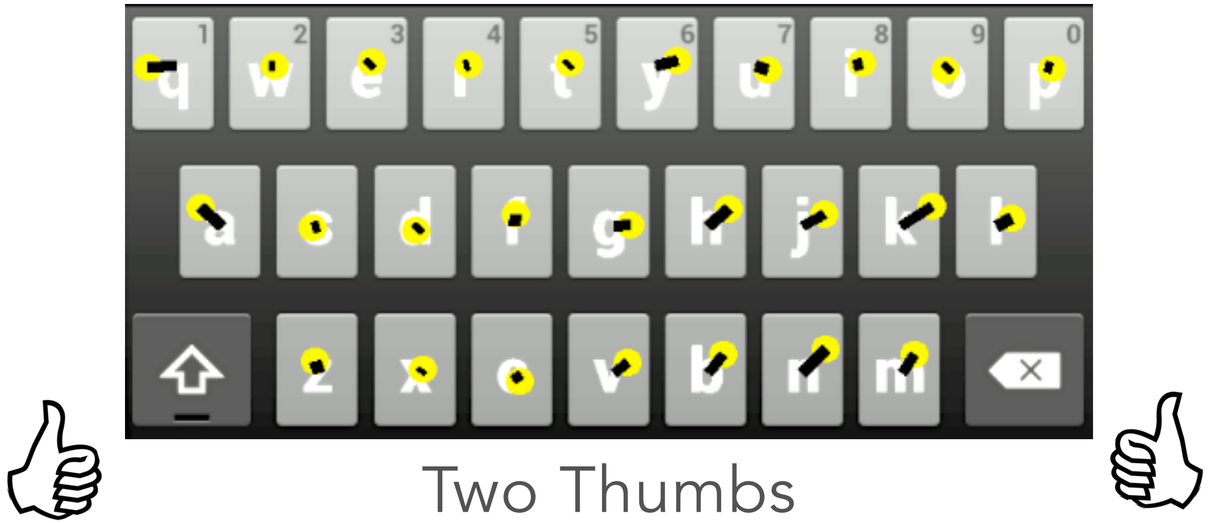
Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

Modeling phone grip



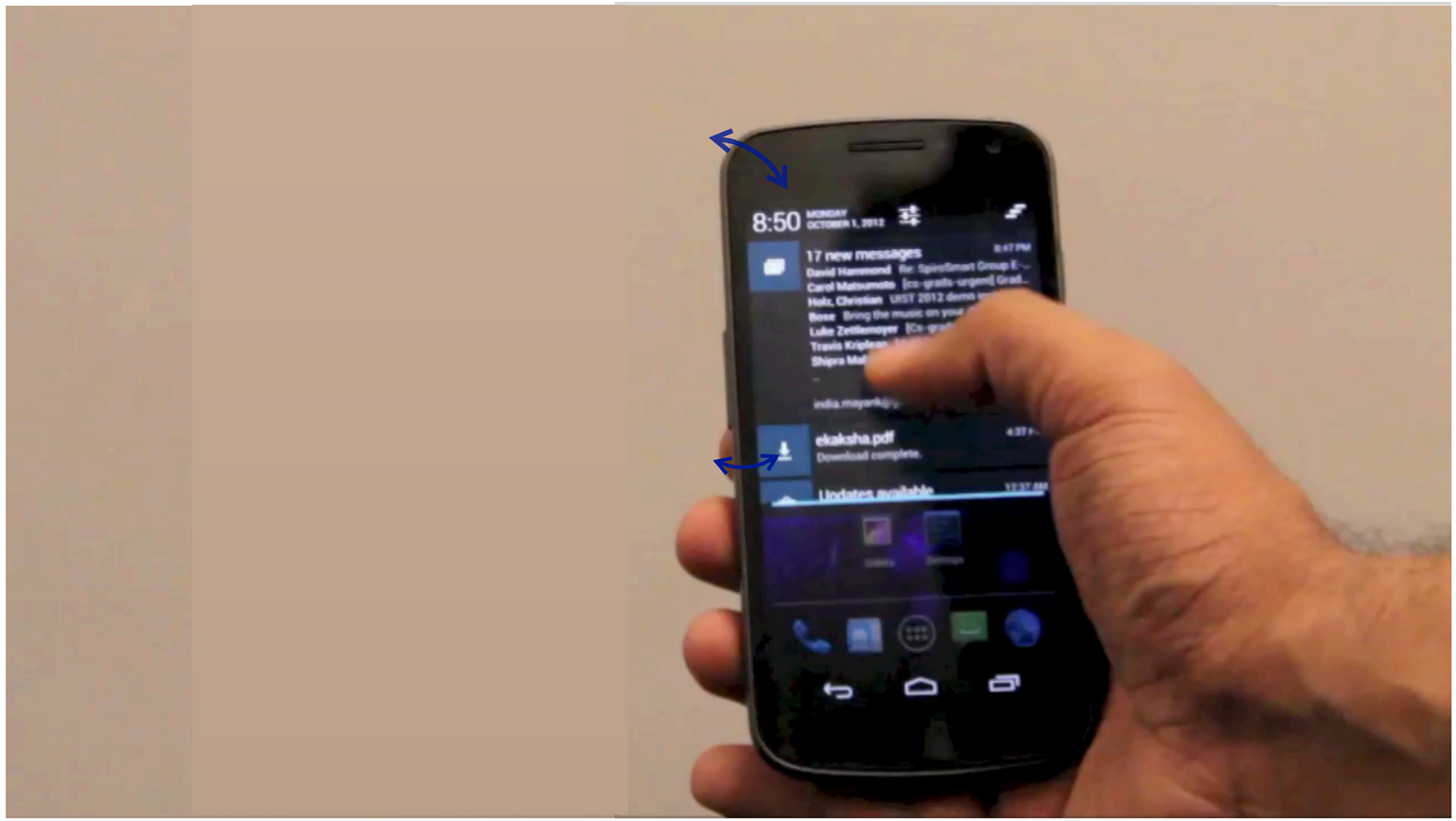
Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

Modeling phone grip



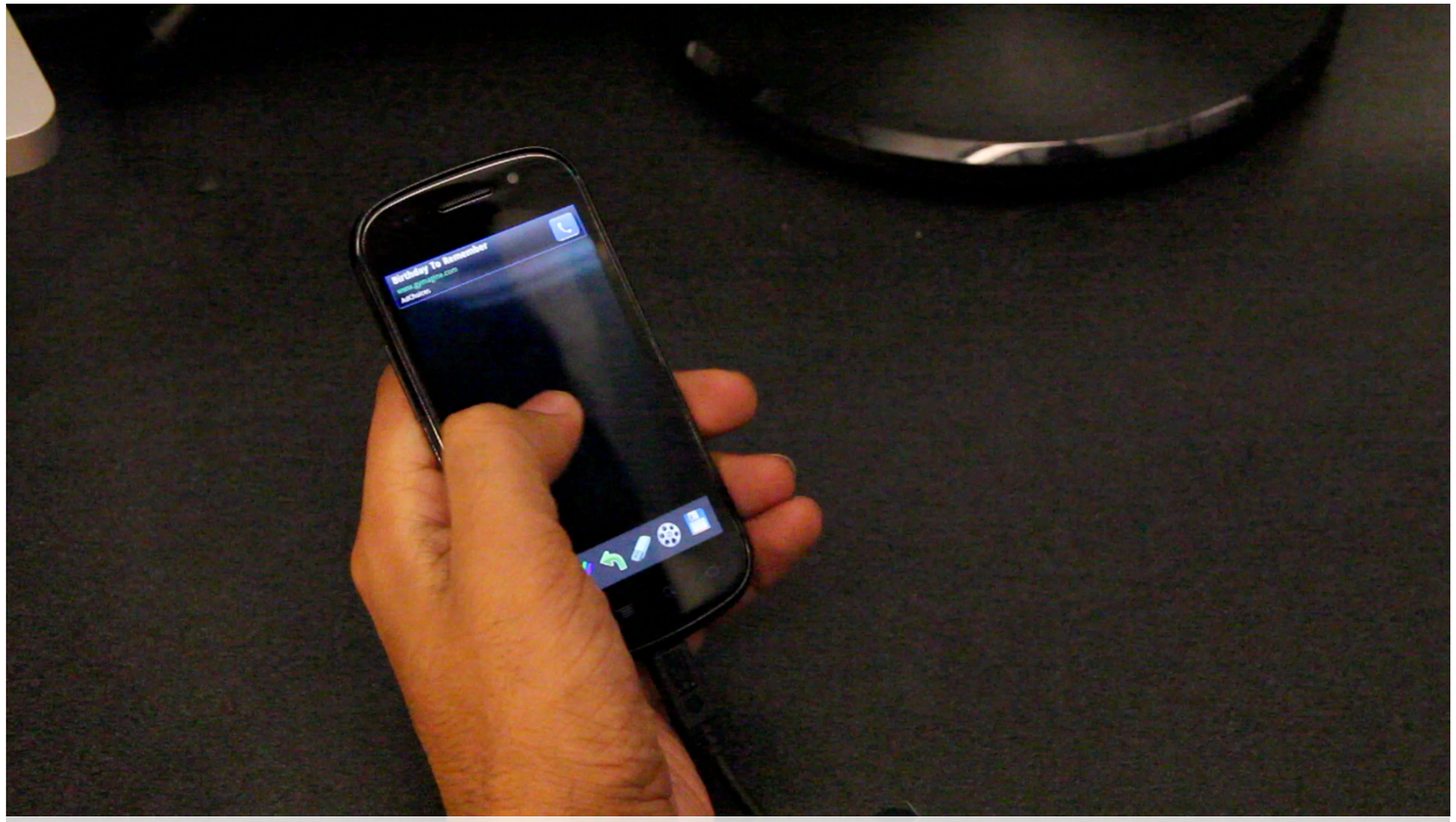
Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

Detecting phone grip with sensors



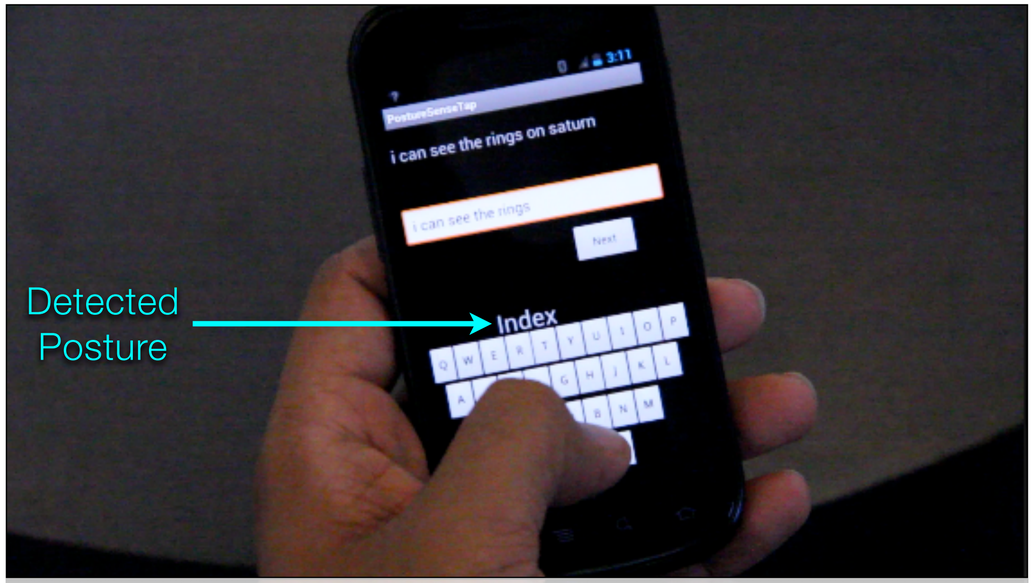
Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

Detecting phone grip with sensors



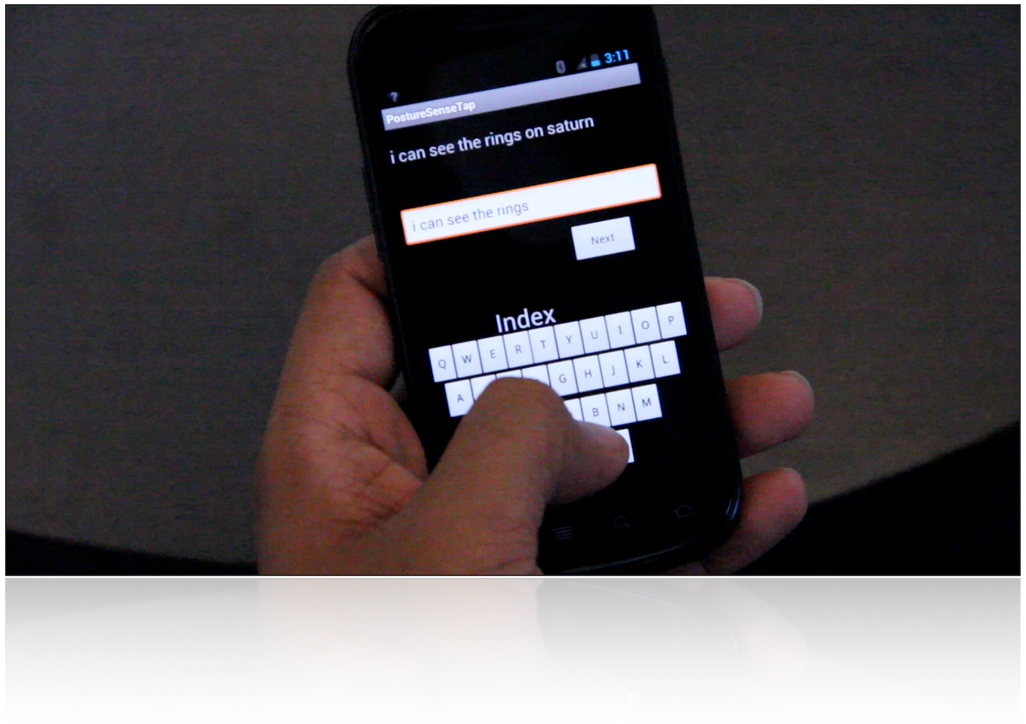
Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

Detecting phone grip with sensors



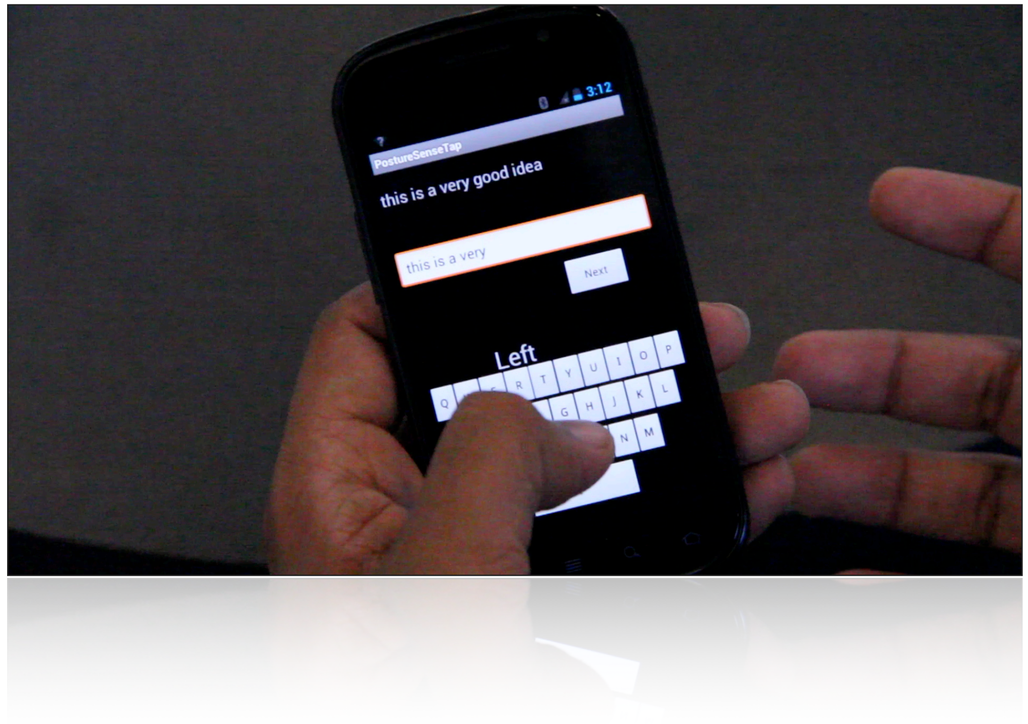
Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

Detecting phone grip with sensors



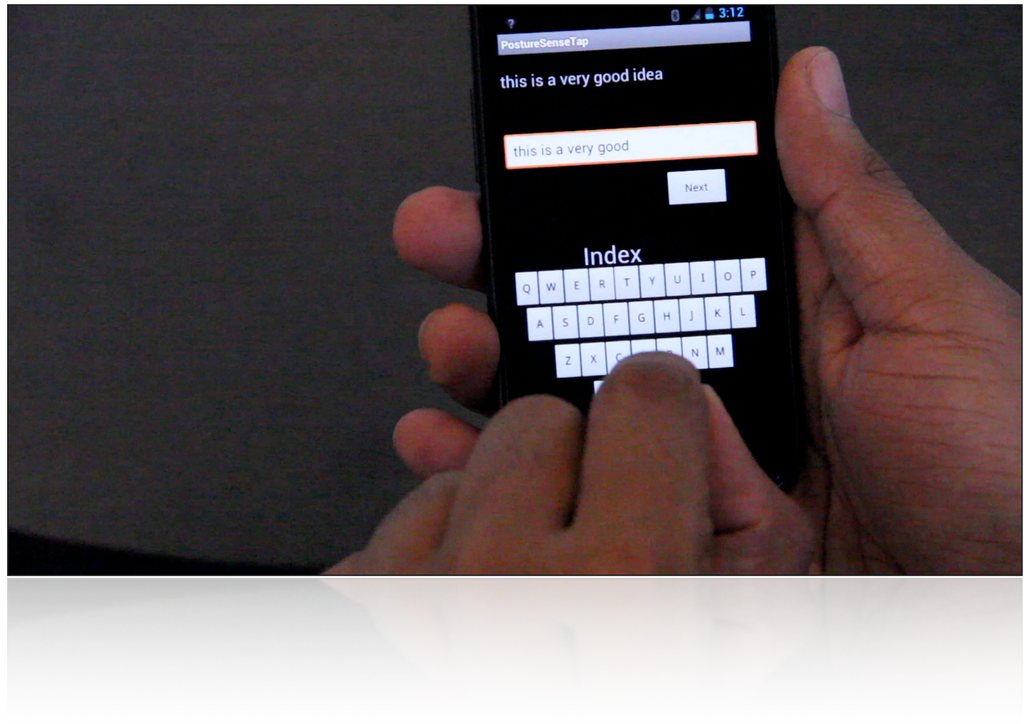
Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

Detecting phone grip with sensors



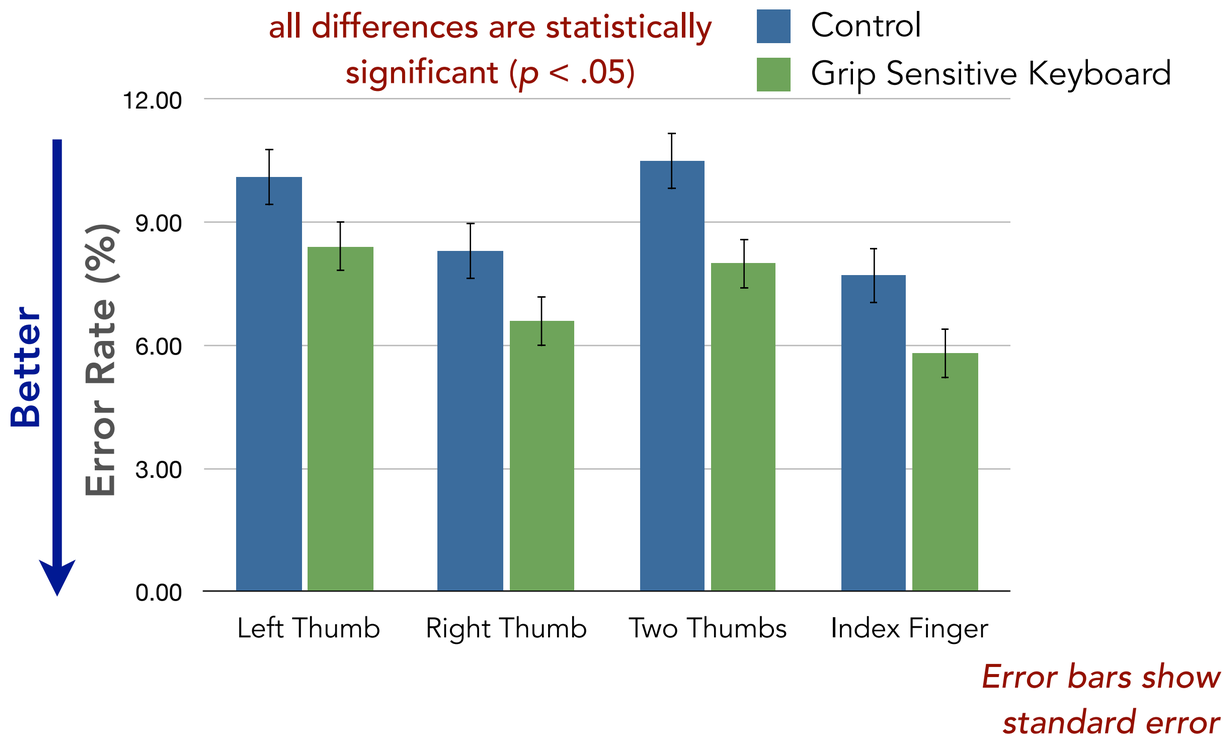
Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

Detecting phone grip with sensors



Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

Detecting phone grip with sensors



Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. CHI 2013. https://doi.org/10.1145/2470654.2481386

Summary

- Modeling helps us measure and predict whether a tool or approach is beneficial for a task
- Fitts's law models time taken to click on a target
  - Demonstrates that larger, nearer buttons reduce time taken
- Improved models lead to higher accuracy
  - Adjust for finger angle and rotation rather than assuming that a user intends to touch with the center of their finger
  - Infer grip using phone sensors to improve typing accuracy

Today's goals

By the end of today, you should be able to…

- Differentiate and explain the roles of Angular components, modules, and services
- Implement a service in Angular
- Navigate Angular's file structure
- Describe the major components of Fitts's Law
- Explain how Fitts's Law impacts how interfaces should be designed
- Describe approaches for correcting systematic errors in touch performance

IN4MATX 133: User Interface Software

Lecture 11:
Separation in Angular &
Modeling human performance