Designing for motor disabilities

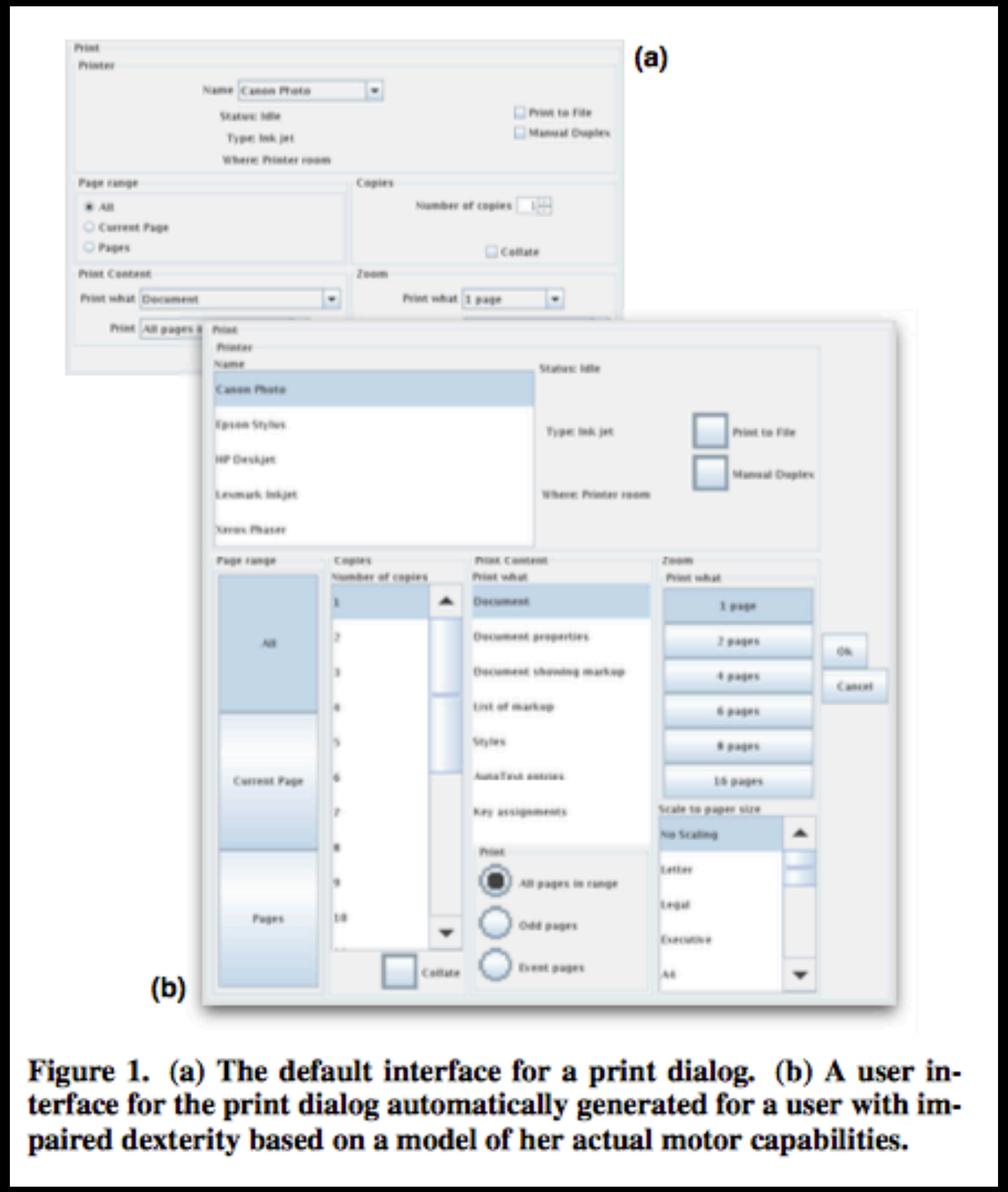


Figure 1. (a) The default interface for a print dialog. (b) A user interface for the print dialog automatically generated for a user with impaired dexterity based on a model of her actual motor capabilities.

This week

- DialogFlow tutorial
- Accessibility for motor impairments

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?

Designing for motor disabilities

Designing for motor disabilities

- Understanding types of motor difficulties
- Benchmark projects
- Strategies for addressing motor challenges

Understanding motor disability

- Health conditions:
  - Cerebral palsy, Parkinson’s Disease, ALS, traumatic brain injury
  - Temporary injuries (broken arm, extreme fatigue)
- Difficult to design for each condition (they vary greatly)
- Instead, focus on functional characteristics

Functional characteristics

- Reduced range of motion
- Inability to make smooth or continuous motions
- Inability to pose or gesture properly
- Tremor / weakness
- Limited fingers / hands / etc
- Fatigue / pain
- Can affect other aspects of interaction (e.g. **dysarthric speech**)
- Comorbidity with other disabilities (cognitive, vision, etc)

Common interaction challenges

- Reach
- Performing gestures (range of motion may vary)
- Bimanual interaction
- Fast or repetitive motions
- Intelligible speech (because of dysarthric speech)

Participation restrictions

- Tasks that require dexterity, reach, fast or repetitive motions
- May have difficulty juggling multiple tasks or items, retrieving items from the environment
- May have difficulty accessing spaces because they are not accessible to a wheelchair or other mobility device)

Representative users

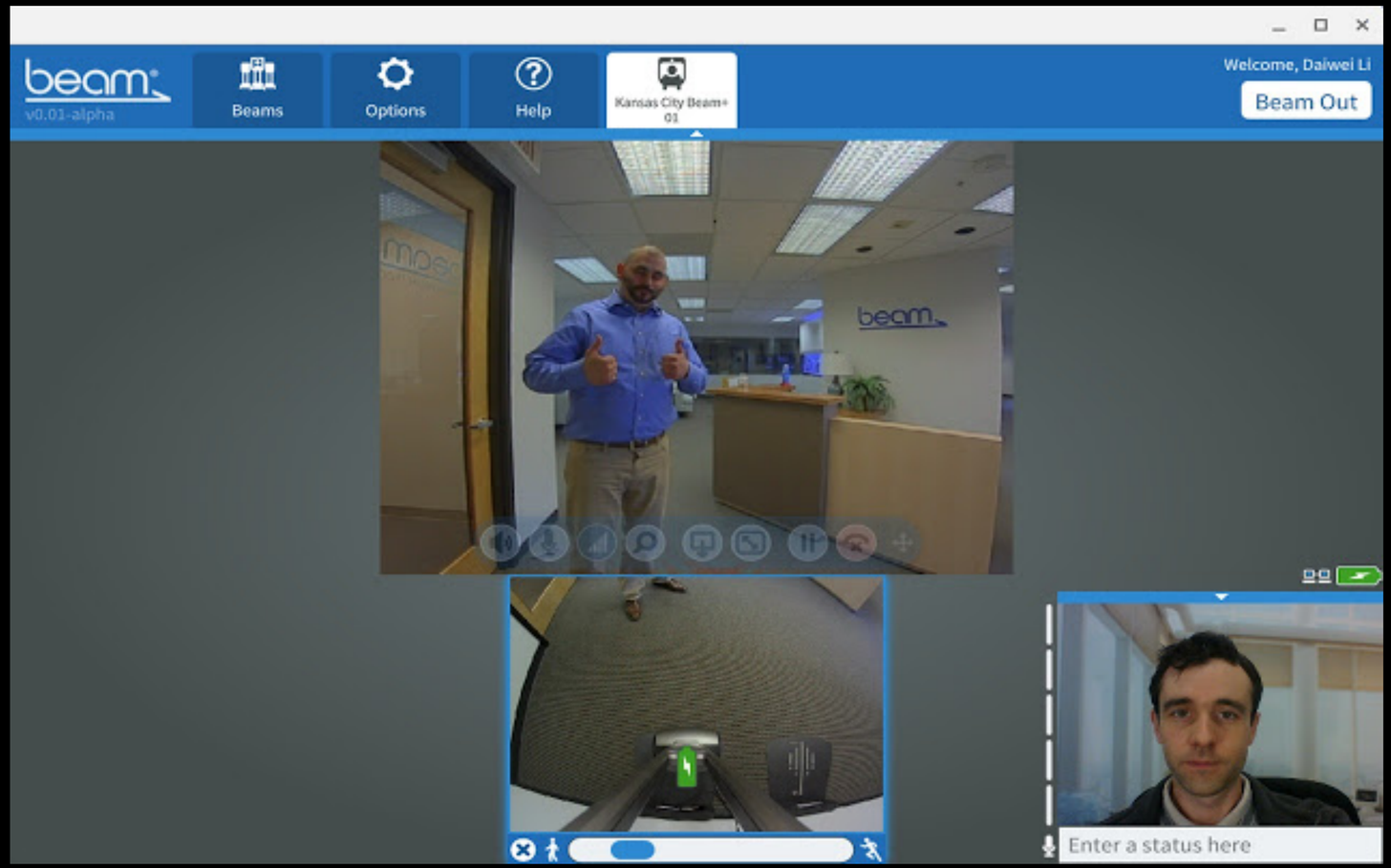
- Kavita Krishnaswamy (SMA)
- Steve Gleason (ALS)

Kavita Krishnaswamy

- PhD student in computer science at UMBC
- Has spinal muscular atrophy
- Has movement in head and face, very limited movement in one hand
- Often participates via Skype or a telepresence robot

Using the Beam

- Costs about $10,000 USD
- Operated via the browser
- Controls for driving the robot, logging in
- Requires wifi (so it doesn’t work in an elevator!)



Steve Gleason

- Former pro football player
- Diagnosed with ALS at 34
- Cannot speak
- Limited to eye gaze and some eyebrow movement

Learning about people with motor impairments

- How do we do it?

Learning about people with motor impairments

- Collaborate with community orgs
- Simulation?
- Using assistive technology
  - But how devices are used varies
- Learn from online demonstrations

Participation

all tech considered TECH, CULTURE AND CONNECTION

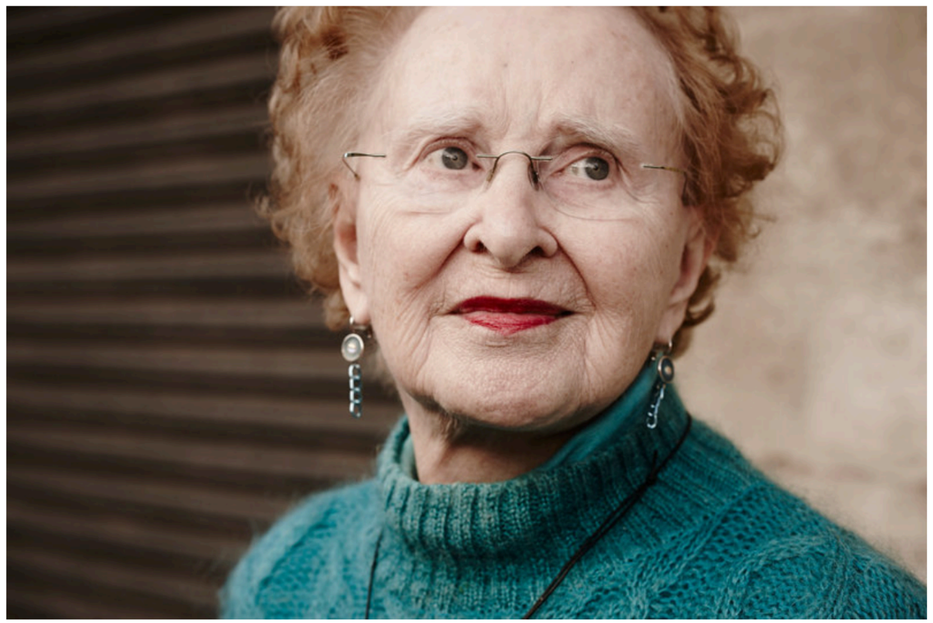
INNOVATION

At 90, She's Designing Tech For Aging Boomers

4:36

+ Queue
Download
Embed
Transcript

January 19, 2015 · 2:32 PM ET
Heard on All Things Considered



LAURA SYDELL

Barbara Beskind, 90, is a designer at IDEO who works with engineers on products that improve the quality

Participation

- Does the need for representative users on design teams leave others out?
  - Should only older adults design for older adults?

- No! There is always a need for experts in technology and design to translate work from the team, evaluate results

Simulation

‘Aging suit’ helps develop cars for older drivers

Automakers’ engineers simulate arthritis, weight gain, impaired eyesight

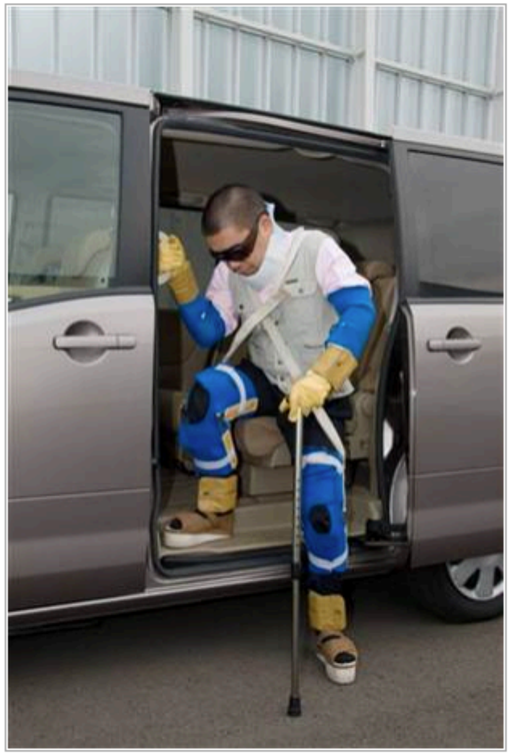
FRANKLIN, Tenn. — Nissan calls it an "aging suit," a cumbersome, strap-on outfit that gives young auto designers the feel of driving with a bulging belly, arthritic joints and shaky balance.

The suit — including goggles that distort color and mimic the effects of cataracts — is used to simulate the physical effects of aging as designers work to make future vehicles safer and more comfortable.

Drivers 65 and older, while not as accident-prone as the youngest drivers, are 16 percent more likely than adult drivers ages 25 to 64 to cause an accident, according to a 2007 report by the Rand Institute for Civil Justice.

Ford Motor Co. also has used what it calls a "Third Age" suit to simulate aging and now uses a virtual reality lab to evaluate vehicle ergonomics and clarity of drivers' views. At General Motors Corp., researchers are working on a high-tech windshield designed to enhance a driver's view.

Joseph Coughlin, director of the Massachusetts Institute of Technology AgeLab and the U.S. Department of Transportation's New England Transportation Center, said automakers have a "long way to go."



AP

Engineers at the Nissan Technology Center outside Tokyo have been using their aging suit to research dashboard switch locations and angles, interior seating and exterior views.

User-generated content

- People often share their experiences online
- YouTube/blogs, online communities for a certain condition, help or tech support forums…

Analyzing User-Generated YouTube Videos to Understand Touchscreen Use by People with Motor Impairments

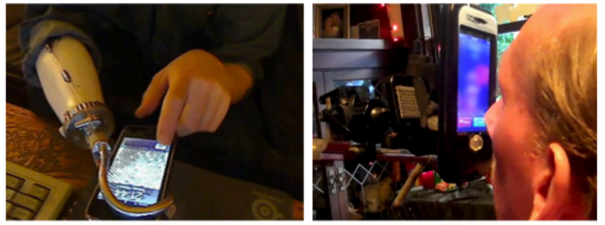
Lisa Anthony
UMBC Information Systems
1000 Hilltop Circle
Baltimore MD 21250 USA
lanthony@umbc.edu

YooJin Kim
College of Information Studies
University of Maryland
College Park MD 20742 USA
ykim0710@umd.edu

Leah Findlater
College of Information Studies
University of Maryland
College Park MD 20742 USA
leahkf@umd.edu

ABSTRACT
Most work on the usability of touchscreen interaction for people with motor impairments has focused on lab studies with relatively few participants and small cross-sections of the population. To develop a richer characterization of use, we turned to a previously untapped source of data: YouTube videos. We collected and analyzed 187 non-commercial videos uploaded to YouTube that depicted a person with a physical disability interacting with a mainstream mobile touchscreen device. We coded the videos along a range of dimensions to characterize the interaction, the challenges encountered, and the adaptations being adopted in daily use. To complement the video data, we also invited the video uploaders to complete a survey on their ongoing use of touchscreen technology. Our findings show that, while many people with motor impairments find these devices empowering, accessibility issues still exist. In addition to providing implications for more accessible touchscreen design, we reflect on the application of user-generated content to study user interface design.

Author Keywords
Touchscreen; motor impairments; physical disabilities; assistive technology; YouTube; iPad; iPhone.



(a) (b)

Figure 1. Examples of unconventional touchscreen use being adopted by people with physical disabilities: (a) a user with a hand prosthesis demonstrates unlocking; (b) nose input with an iPhone.

particularly problematic for people with physical disabilities. Research on touchscreen interface design for users with physical disabilities has been largely limited to lab studies with relatively few participants [3,6,10,16,31], or to small interview studies [20]. Moreover, even less attention has been paid to subpopulations such as children.

To develop a richer characterization of how people with physical disabilities are adopting touchscreen devices, we turned to a previously untapped source of data: YouTube videos. We collected and analyzed 187 non-commercial

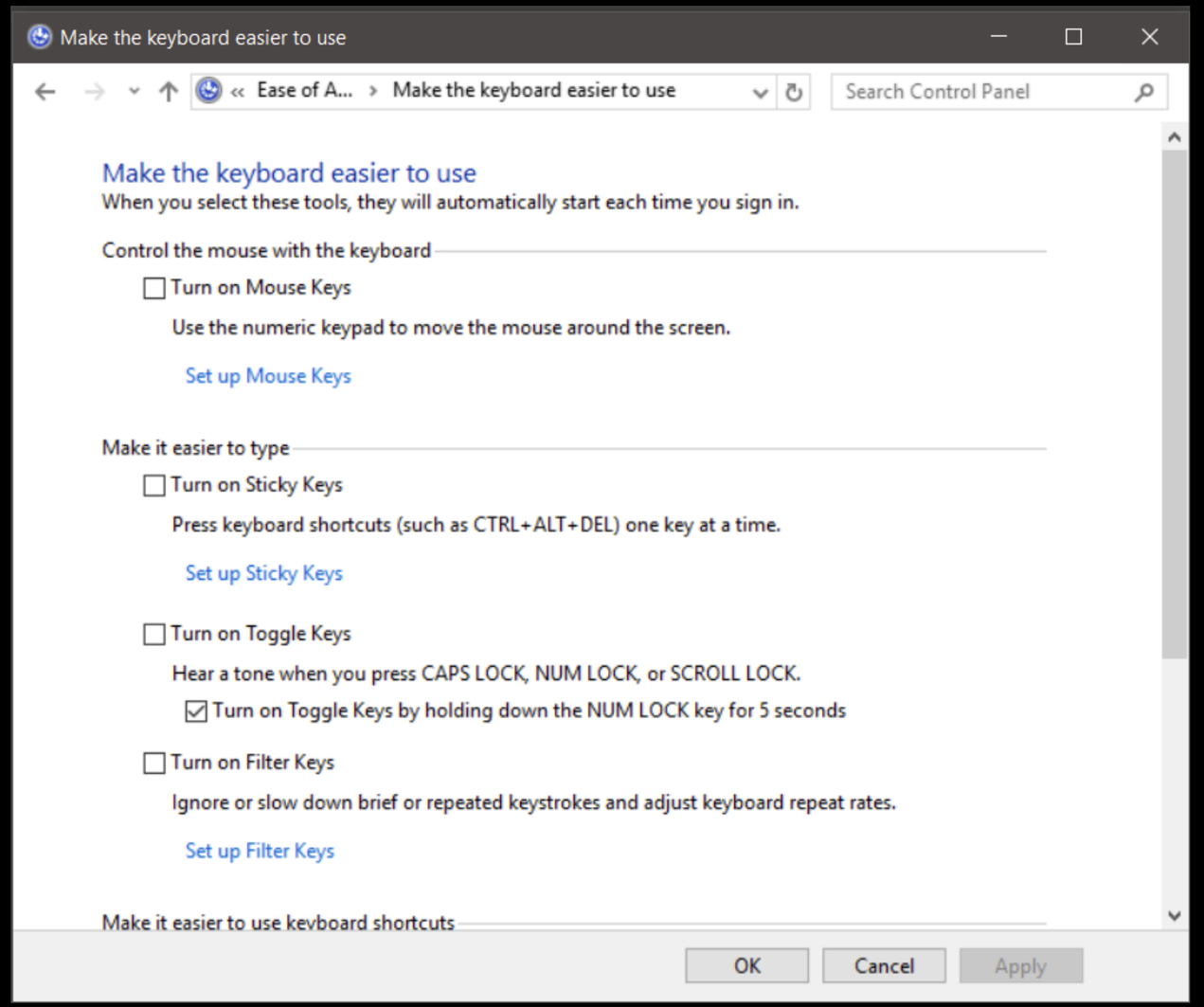
Assistive technology for motor disabilities

Improving computer access: strategies

- Replace complex interactions with simple interactions
- Use hardware that helps prevent errors
- Use software to detect and reduce errors
- Alter existing interfaces to be more accessible

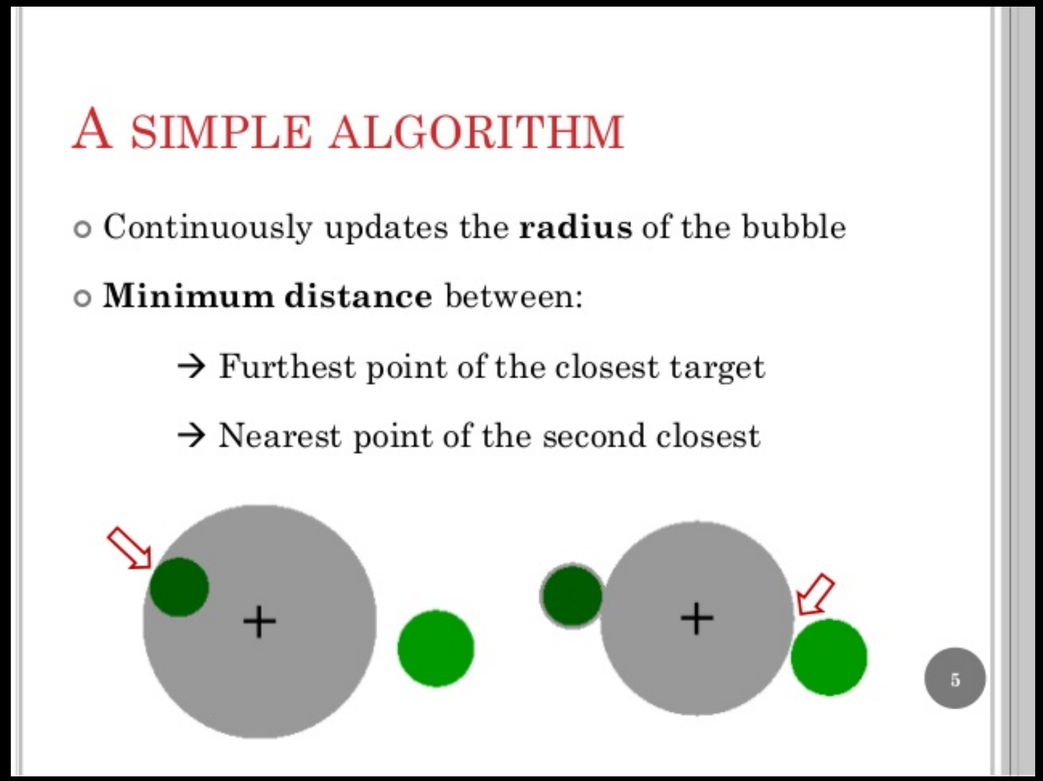
Assistive technologies: software

- **Mouse keys** - use keys to move mouse cursor
- **Sticky keys** - make shift, alt, control **modal** - press once to toggle mode
- **Filter keys** - remove repeated keys if the key is held down for too long



Assistive pointing

- Seems like the Bubble Cursor and similar techniques would help
- So why don’t we use them?



Where design meets reality

- The bubble cursor is **target-aware**
- Requires that the application knows where each target is; must be able to control either the pointer or change the UI layout
- Many interesting accessibility systems cannot easily be implemented in the real-world due to OS limitations

Can we help pointing?

- Identify possible errors using heuristic methods or machine learning
- Example: Steady Clicks (Trewin et al., ASSETS ’06)
  - Some users consistently slip after clicking the mouse button
  - Detect if the cursor moves between mouse down and mouse up; if so, put it back at the mouse down location
  - Doesn’t require knowledge of the underlying UI

Physical adaptations

- Keyguards
- Switch control
- Head mouse
- Eye trackers (more next week)
- Physical pointers / styli
- Using other body parts / poses

Keyguard

- Physically constrains movements between keys
- Drawbacks: expensive, fragile, difficult to transport, can’t use your buddy’s device

An “invisible keyguard”

- Detect error-like movements and correct them

An Invisible Keyguard

Shari Trewin
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown, NY 10598 USA
+1 914 784 7616
trewin@us.ibm.com

ABSTRACT
Overlap errors, in which two keys are pressed down at once, are a common typing error for people with motor disabilities. Keyguards are a commonly suggested means to reduce overlap errors. However, they are also unpopular with many users. We present an alternative to the keyguard, a software filter which targets overlap errors. Basic, keystroke timing-based, and language-based techniques for identifying and correcting overlap errors are described. Their performance is compared using a corpus of typing data recorded by keyboard users with motor disabilities. The best filter performance was obtained by using keystroke timing characteristics to identify and filter out extra characters. Accuracy of error identification was dependent on the typing style of the user. The filter accurately corrected 80% of the overlap errors presented. Combining the identification and correction techniques gave a 50-75% reduction in errors for the three study participants with the highest error rates.

Keywords
Keyguard, keyboard, accessibility, OverlapKeys, typing errors, motor disabilities

Display a menu

tremor, or disruption of proprioceptive or visual feedback mechanisms.

For users who have high overlap error rates, a keyguard may be recommended as a way to suppress these errors. However, keyguards are often unpopular with users.

A new accessibility utility – *OverlapKeys* – is proposed. OverlapKeys acts as a software keyguard in the sense that it suppresses overlap errors, but is not subject to the drawbacks that make physical keyguards unpopular.

This paper explores the potential effectiveness of software filtering of overlap errors, and compares a number of possible implementations of OverlapKeys. A quantitative comparison of these options, based on recorded typing data, is made. This leads to specific recommendations for an implementation of OverlapKeys, and a method for assessing the expected benefit to be gained by using the filter for a particular user.

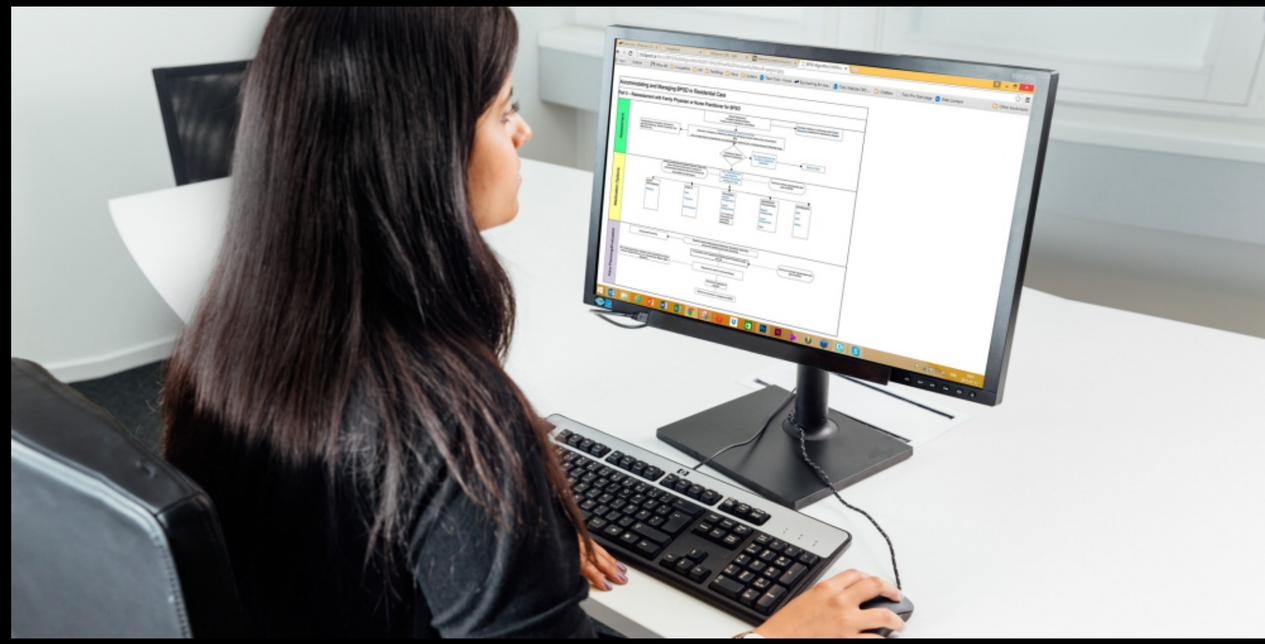
BACKGROUND
Keyboard users who make frequent overlap errors have a number of options for improving the accuracy of their

Head mice

- Can use a reflective dot on the head or track face
- May use infrared cam (with an IR-reflective dot) or webcam
- Interaction challenges?
  - Speed vs. accuracy tradeoff (gain)
  - Midas touch problem
  - Field of vision

Eye tracker

- Infrared camera tracks movement of pupils
- Useful when head mouse is not possible
- Similar challenges to head mouse, but may be even less accurate



Physical pointers

- Can be attached to various parts of the body
- Can support different end effectors
- May require capacitance, pressure, etc

Alternative poses / body parts

- May be unable to form certain hand shapes
- Can be a problem for multi-finger and whole-hand gestures
- Range of motion may be very different

Using other body parts

- May have limited range of motion, strength, visibility
- Other concerns depending on the body part (e.g. can’t see when you’re pointing with your nose)

Wednesday

- Switch control
- Cool research demos
- Developing few-button interfaces

Switch control

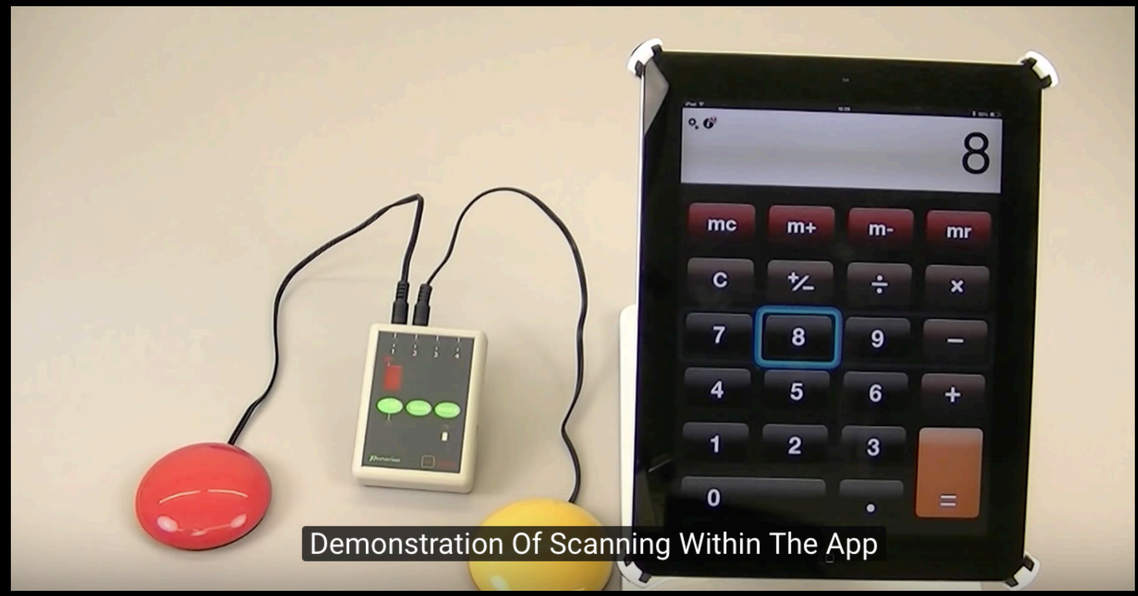
- These are really just buttons + software to control traditional user interfaces
- May be placed in various places (near hands, head, etc.)
- Multiple actuation methods depending on type
  (e.g. firm or light pressure, proximity)

Designing switch interfaces

- What to do with only N keys?
- 5 keys: 4 directions + select
- 3 keys: previous, next + select
- 2 keys: next + select
- 1 key: ????

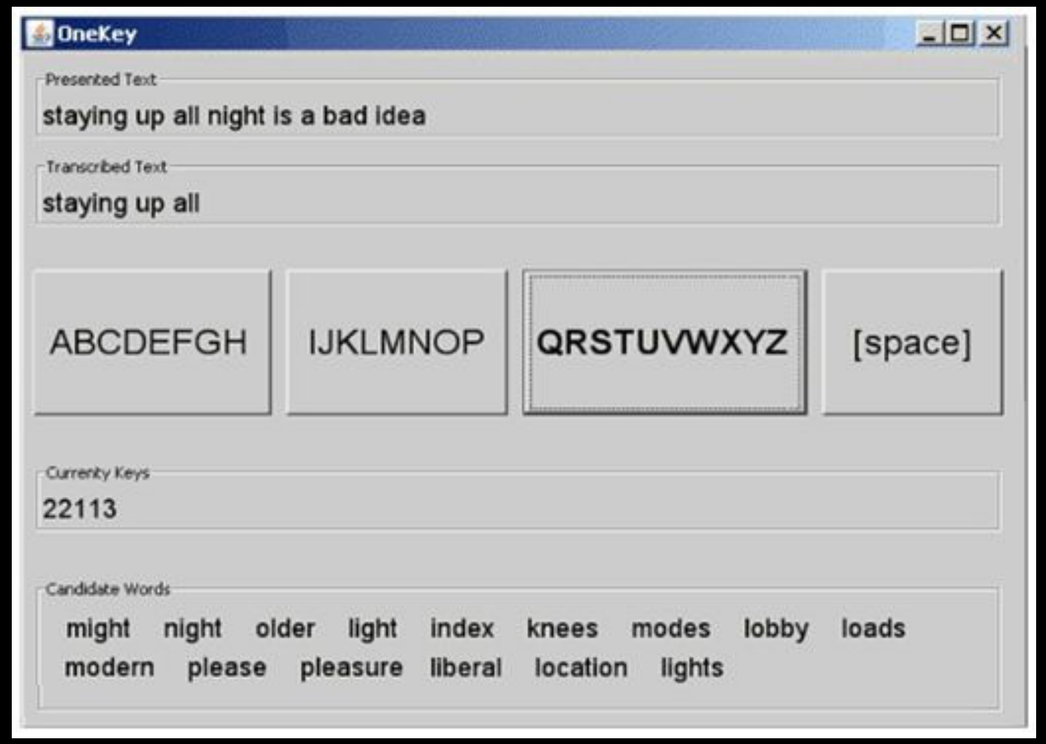
Single switch access

- Use a **scanning** user interface
- Cursor moves between controls at a timed interval
- Press to select the current item
- May pop-up a menu if multiple input methods are available (e.g. left vs right mouse button, gestures)



Scanning Ambiguous Keyboards

- Use language model to predict what the user is writing (and give them backup choices)
- Trade-off between number of keys and quality of guess
- Can formally optimize number of buttons for the user / # of switch controls



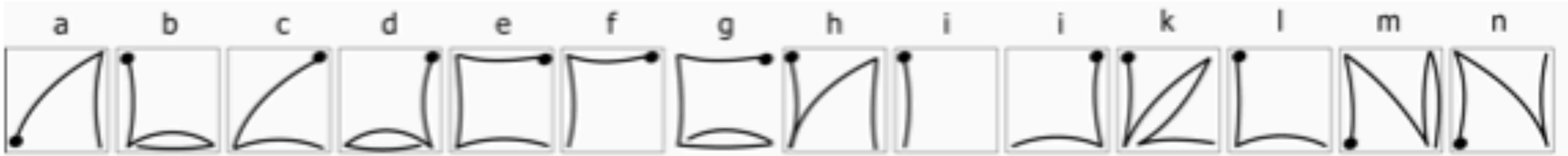
AT examples from research

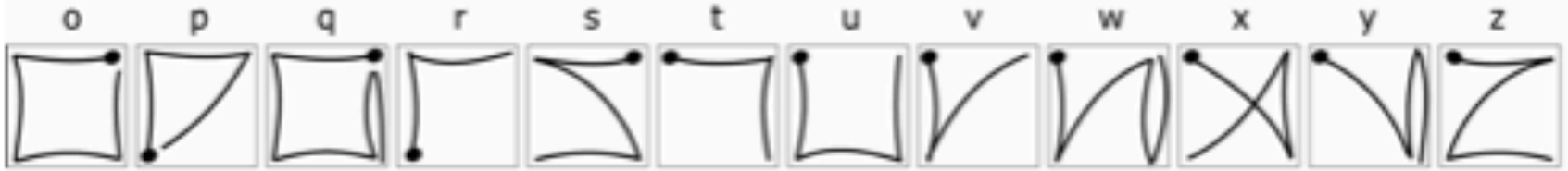
- EdgeWrite
- TrueKeys
- Gest Rest
- Smart Touch
- SUPPLE++
- VoiceDraw
- 3D printed prosthetics and grips

EdgeWrite [Wobbrock et al. 2003]

- Stylus-based text entry for people with motor difficulties
- Uses physical template over writing area to steady hand
- Uses modified stroke alphabet to reduce requirement for straight lines
- 4-corner recognizer supported on many devices (stylus, trackball, joystick, touch screen)

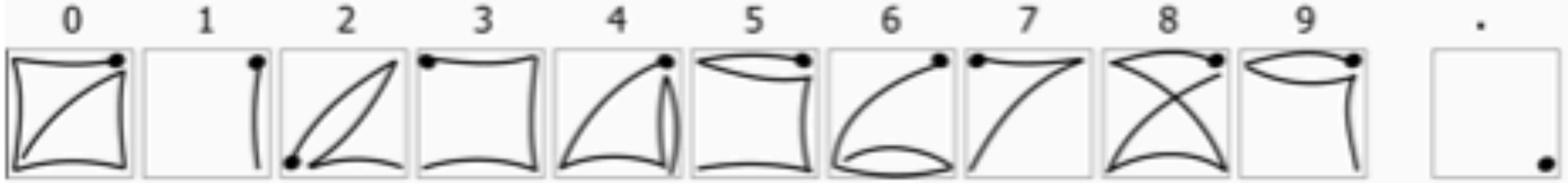
Primary EdgeWrite Character Forms

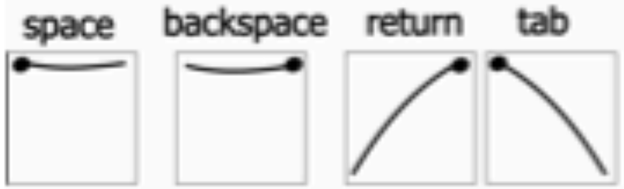




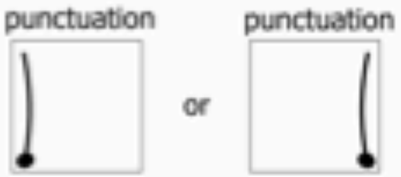
capitalize

To capitalize, make the letter as usual, then finish in the upper-left corner.



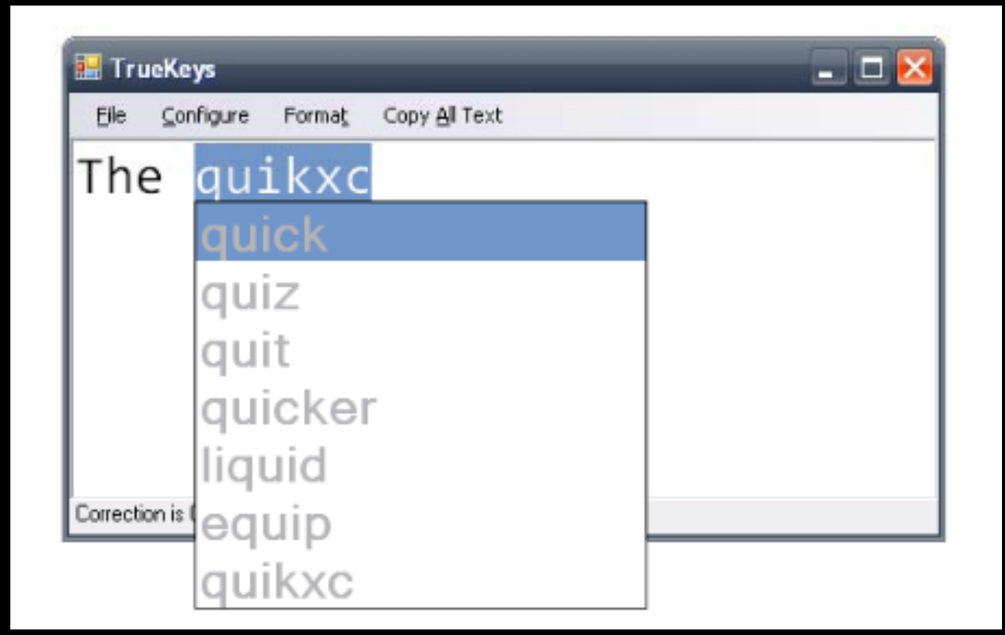


Punctuation Mode:
Stroke up on either side



TrueKeys [Kane et al. 2008]

- Custom spelling corrector for people with motor difficulties
- Recognizes common errors for that specific user
- Let the user type normally; make the system sort it out



Smart Touch [Mott et al. 2016]

- Recognize alternative hand placements on the touch screen
- Allows the user to train a touch point, even if they physically can’t make one

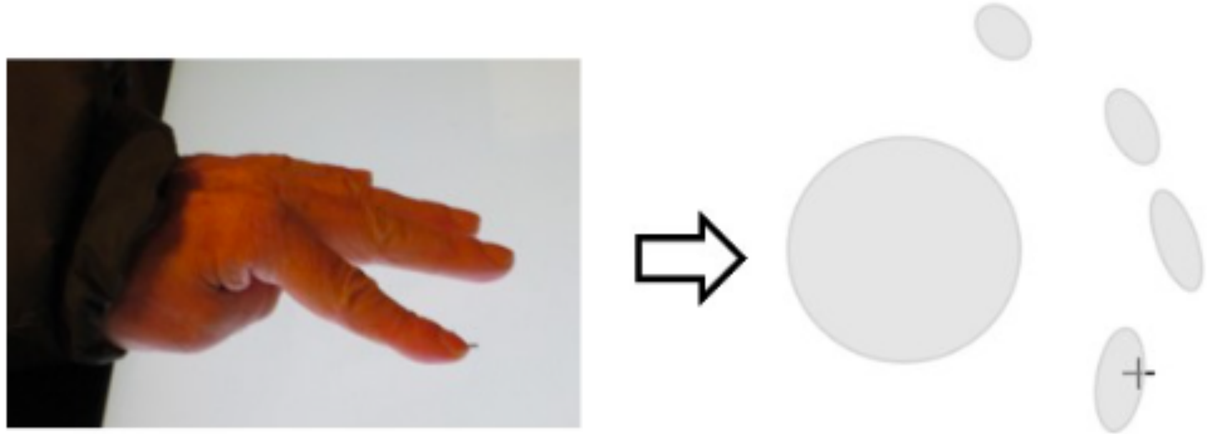


Figure 1. Touching with multiple fingers or various parts of the hand (left) creates various contact regions (right). Current touch screens are not designed to accommodate this kind of touch input when the user's goal is to activate just a single (x,y) point.

Goal crossing [Wobbrock and Gajos 2007]

- Pointing at and clicking a target can be difficult for some users
- Instead, set a goal that the user can drive the cursor through

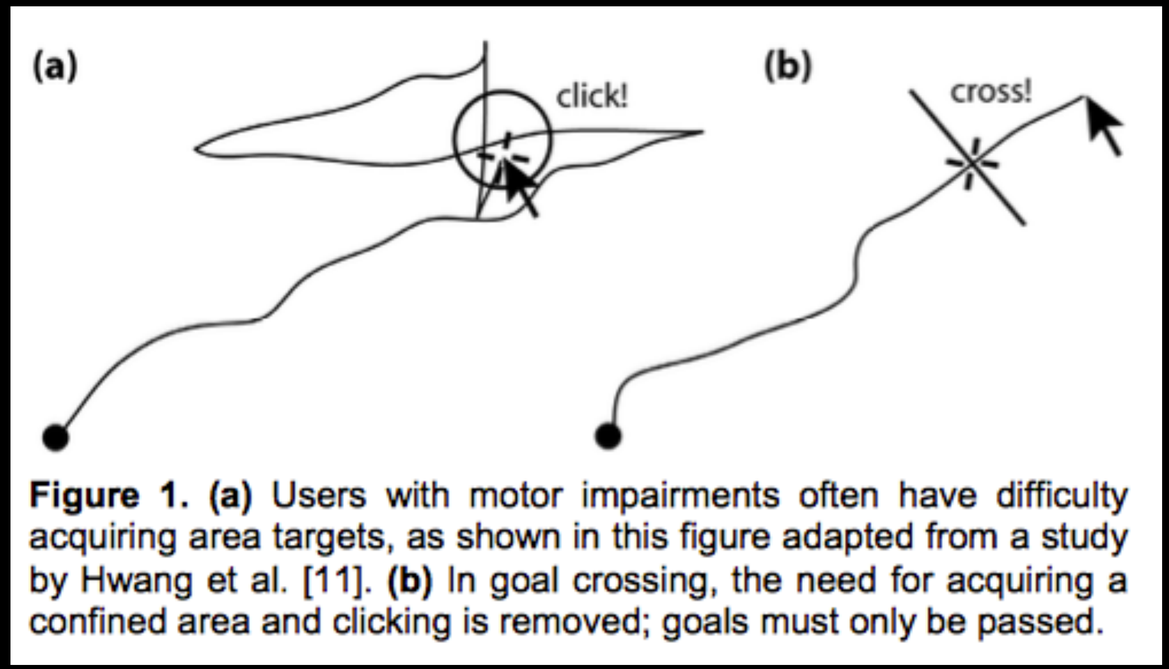


Figure 1. (a) Users with motor impairments often have difficulty acquiring area targets, as shown in this figure adapted from a study by Hwang et al. [11]. **(b)** In goal crossing, the need for acquiring a confined area and clicking is removed; goals must only be passed.

How to design goal crossing interfaces?

- Difficult to adapt to existing UIs
- Targets are line segments, not rectangles!

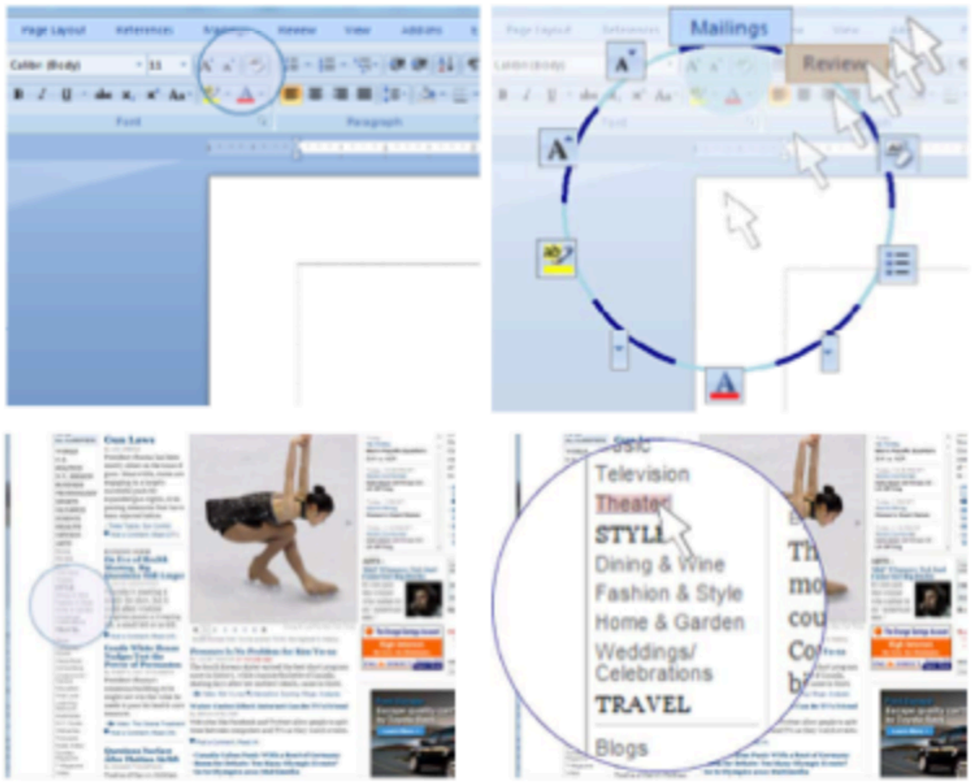
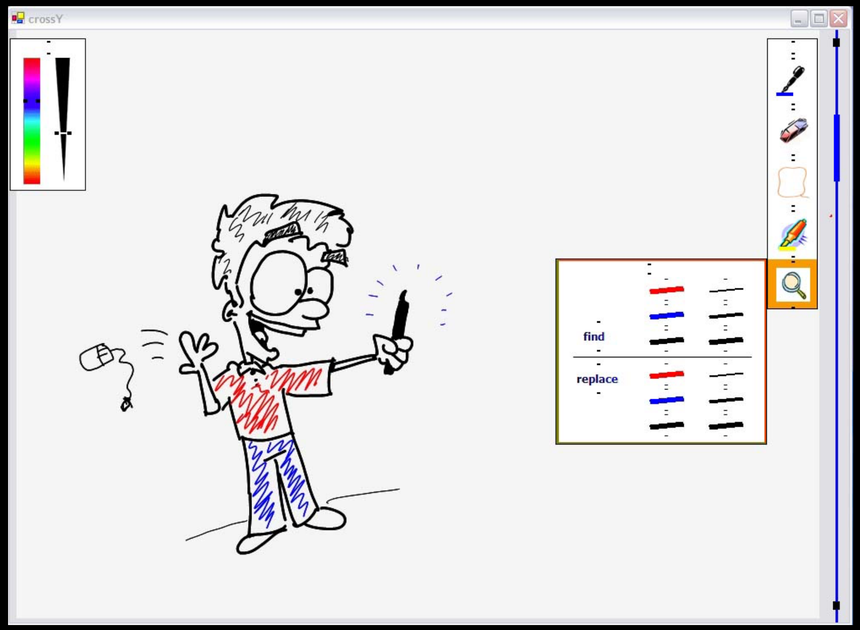


Figure 1. Two enhanced area cursors. Click-and-Cross: an area cursor (top-left) transforms covered targets into crossing arcs (top-right). Visual-Motor-Magnifier: an area cursor (bottom-left) expands visual and motor space for point-and-click selection (bottom-right).

CrossY [Apitz and Giumbretiere 2004]

- Not designed for users with disabilities, but for pen tablet users
- All interactions involve crossing without clicking



SUPPLE++ [Gajos et al. 2008]

- Use performance to tailor user interface
- Adjust button size, spacing, etc.
- Disability agnostic; based entirely on performance

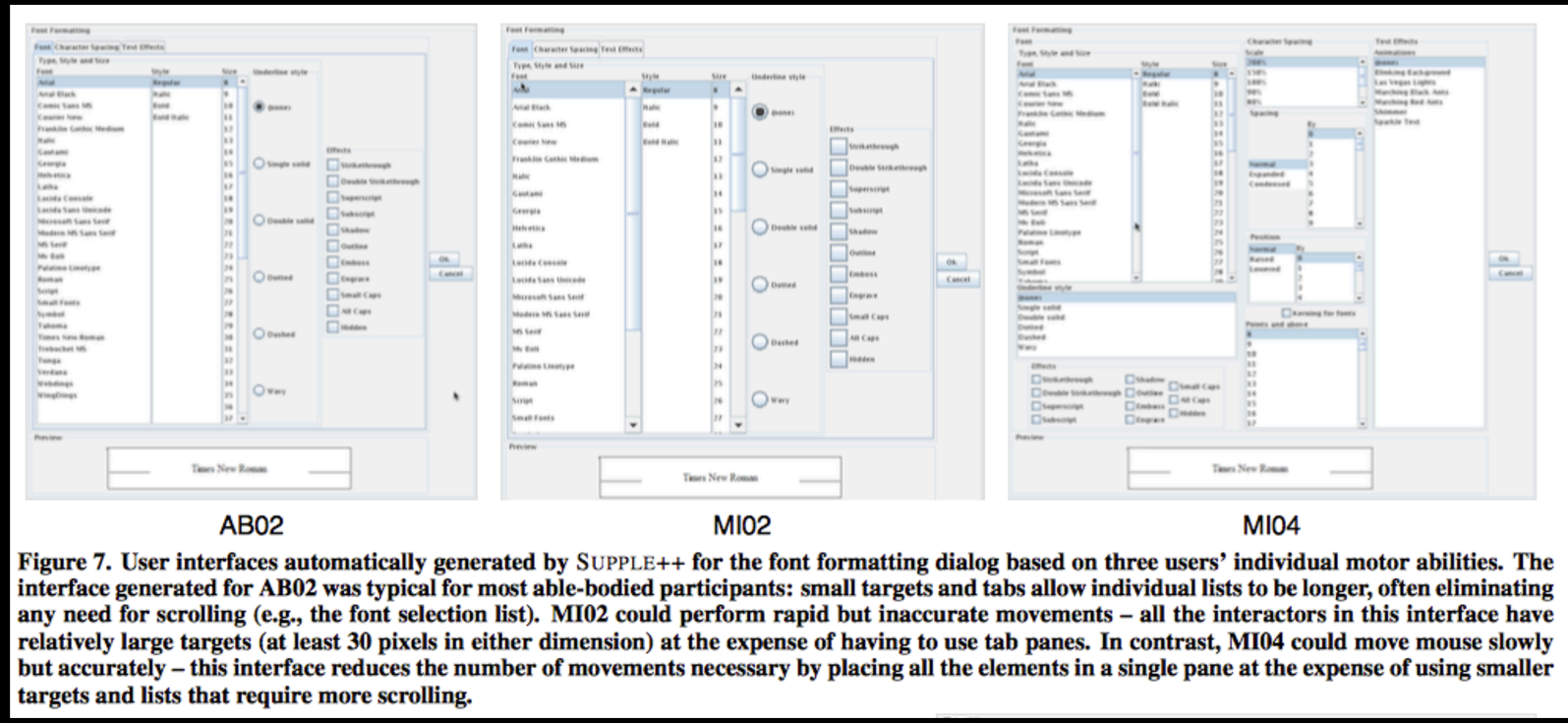
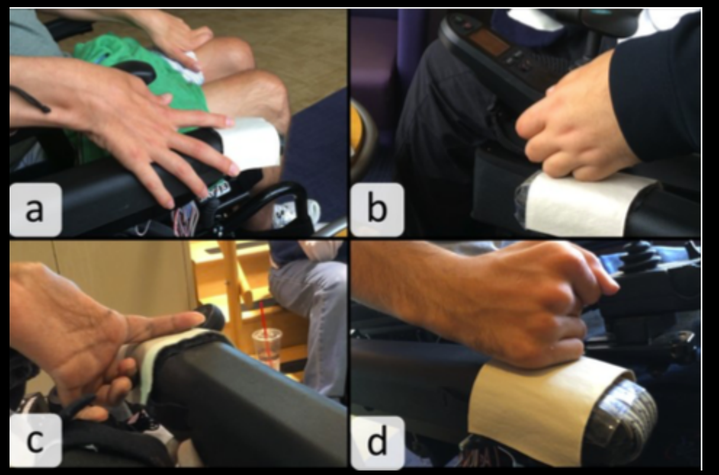
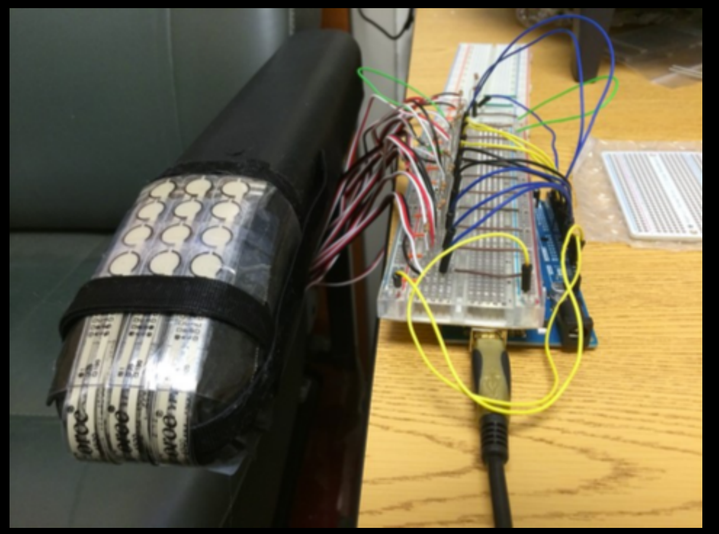


Figure 7. User interfaces automatically generated by SUPPLE++ for the font formatting dialog based on three users' individual motor abilities. The interface generated for AB02 was typical for most able-bodied participants: small targets and tabs allow individual lists to be longer, often eliminating any need for scrolling (e.g., the font selection list). MI02 could perform rapid but inaccurate movements – all the interactors in this interface have relatively large targets (at least 30 pixels in either dimension) at the expense of having to use tab panes. In contrast, MI04 could move mouse slowly but accurately – this interface reduces the number of movements necessary by placing all the elements in a single pane at the expense of using smaller targets and lists that require more scrolling.

Gest Rest [Carrington et al. 2014]

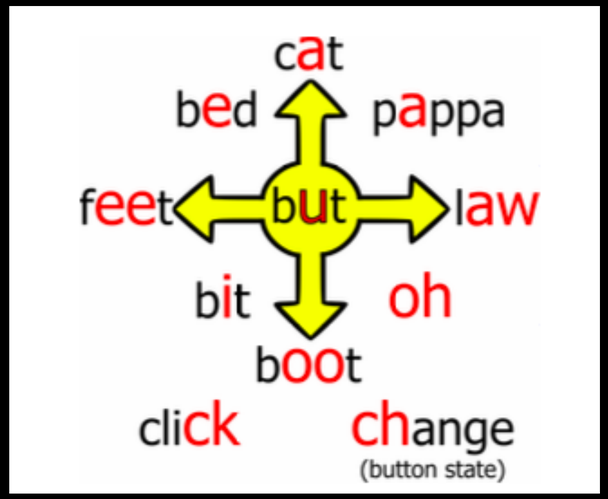
- Integrate input into wheelchair itself
- Array of pressure sensors in a textile cover
- Use pressure-based gestures to support wide range of interactions

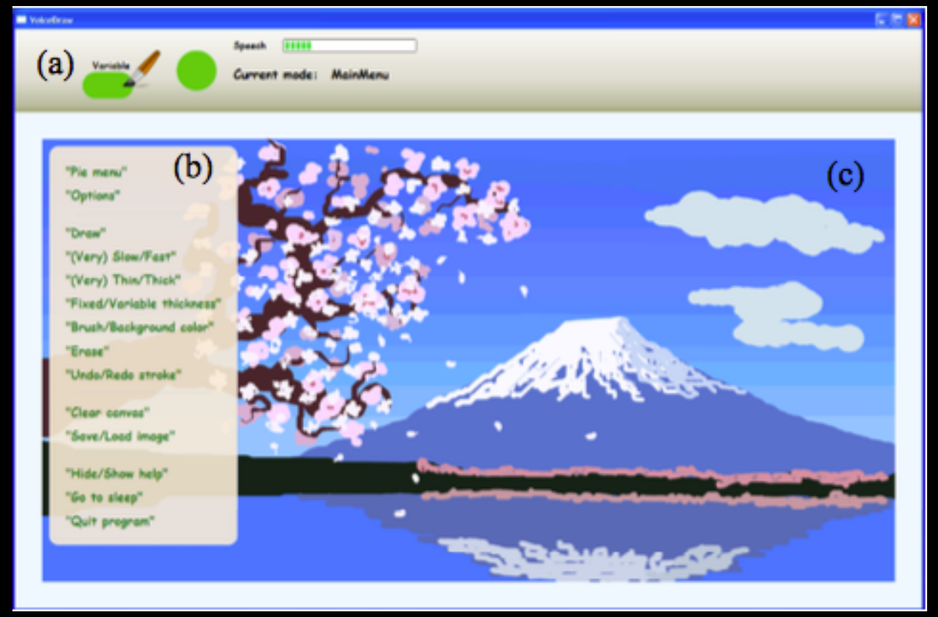




VoiceDraw [Harada et al. 2007]

- Voice based interface that uses non-speech vowel sounds
- Use voice as an expressive input mode (use available variables - tone, volume)





Tongue-drive wheelchair

- Track with metallic tongue stud, smart retainer, or ultrasonic waves
- Extremely useful for paralyzed people

Brain Controlled Interfaces (BCIs)

- Typically use EEG headsets (other methods are much more invasive)
- Limited range of input - typically one or very few signals possible
- Can be combined with a scanning-based UI

NeuroPhone [Campbell et al. 2010]

- EEG-based mobile phone dialer
- Relatively limited signal from brain without going inside the skull
- So, uses scanning interface

3D printing for motor disabilities

- 3D printed prosthetics from e-NABLE project
- Customized 3D printed pen grip

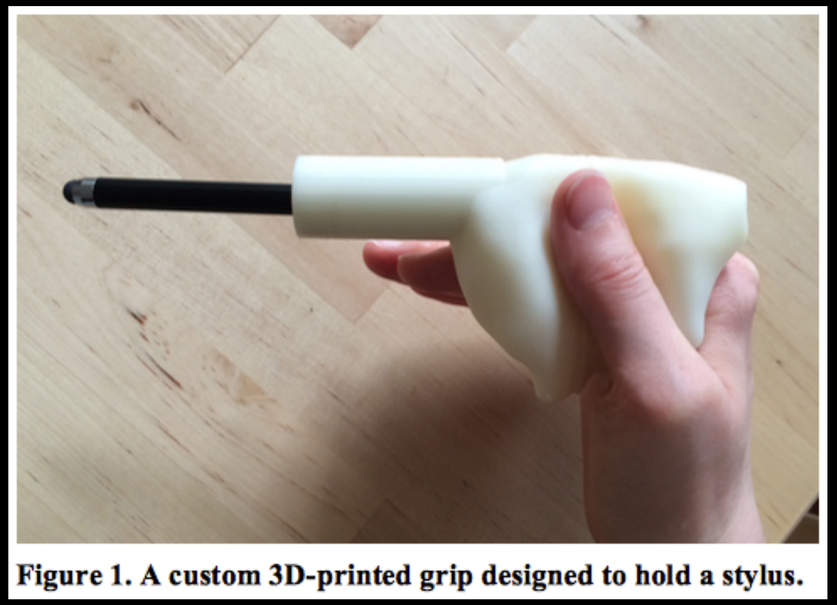


Figure 1. A custom 3D-printed grip designed to hold a stylus.

Takeaways

- Consider alternative uses for existing devices
- Identify the user’s ability, range of motion (even from unexpected places)
- Design gesture or command set to match