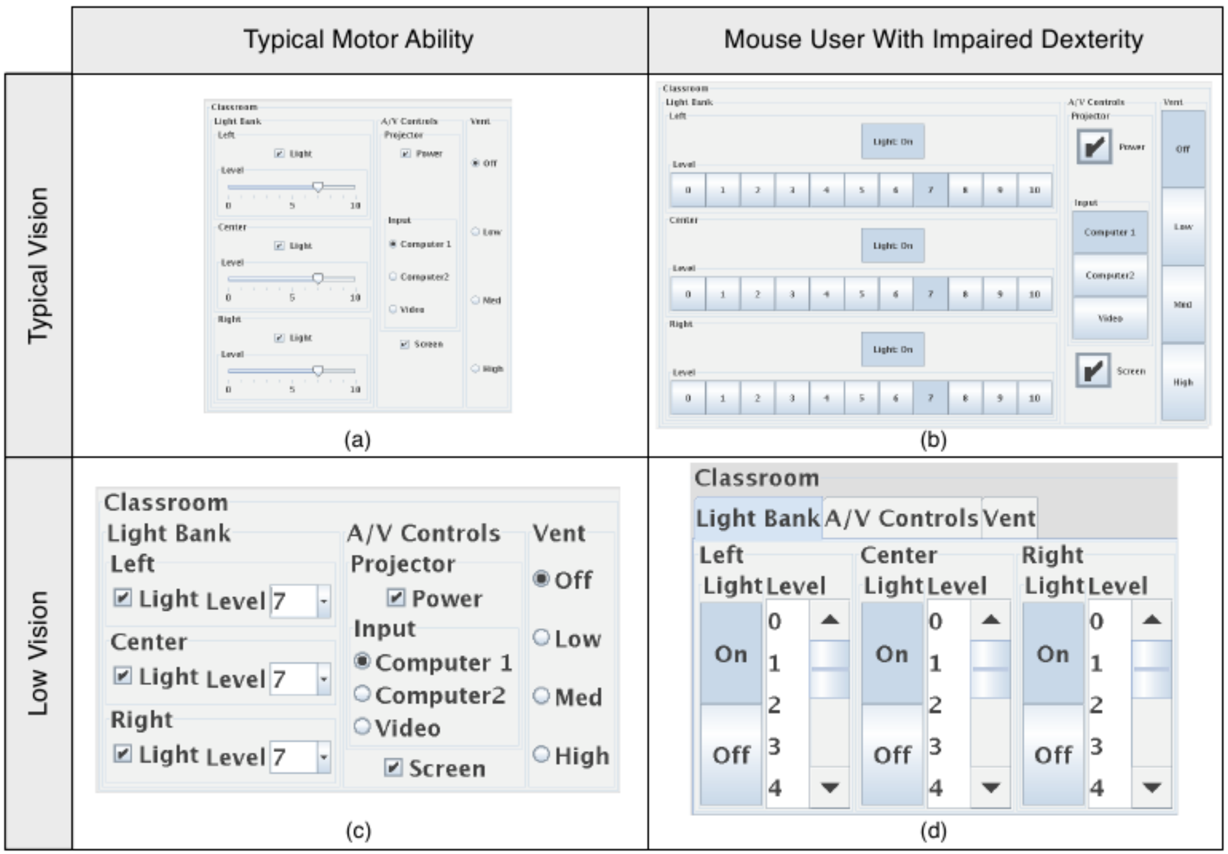
Automatically Generating User Interfaces Adapted to Users' Motor And Vision Capabilities

Krzysztof Z. Gajos, Jacob O. Wobbrock, Daniel S. Weld (University of Washington)

Presented By Kalan MacRow and Stephen Ramage

Motivation

GUI design favours able-bodied users, assistive technologies are naive and poorly maintained: we can do better.



Fitts' Law

Time T required to move to a target area is a function of the distance to the target, and the width of the target

- $T = a + b * ID$
- $ID = lg(D / W + 1)$

Supple

- Describes a UI with a hierarchical functional specification, S_f
- Searches the space of possible UIs, given S_f
- Branch & Bound to select rendering with minimum cost

Design Objective (1)

- "Simple and fast to setup, use, configure and maintain"
- Focus on motor and visual impairments
- Generate UIs that are legible and that can rearrange their contents to fit on the user's screen

Design Objective (2)

- Strike a balance among UI elements: complexity, type, difficulty
- For the visually impaired, provide intelligent improvements, not just enlargement
- Serve people with a *combination* of motor and vision impairments

Supple++

- Model users' motor capabilities in a one-time performance test
- Use this model to personalize UI generation for individuals
- Extend Supple with Expected Movement Time (EMT) based cost function

Training

- Collect motor performance data from participants
- Pointing tasks based on ISO 9241-9

Code	Device used	Health condition
ET01	Eye tracker (ERICA) (click by dwelling)	
HM01	Head Mouse (click w th right fist using a switch)	spinal cord injury (incomplete tetraplegia)
M03	Mouse	muscular dystrophy (impaired dexterity)
M04	Mouse	
TB01	Trackball (Kensington Expert Mouse)	spinal cord injury (incomplete tetraplegia)
TP01	Trackpad (Apple MacBoook Pro)	
VJ01	Vocal Joystick [14]	
VJ02	Vocal Joystick [14]	

Table 1: List of participants

Inadequacy of Fitt's Law

- ET01 (Eye Tracker)
  - Distance to target only marginally affected performance
- HM01 (Head Mouse)
  - Performance degraded sharply for distances larger than 650px
- TB01 (Trackball)
  - Performance improved very slowly for small targets
- Fitts' law says grow widgets without bound
- Empirically poor fit

Pointing Performance Model

1. Find the best set of features to include in the model
2. Train a regression model that is linear in the selected features

Features

Participants

	D	1/W	W	log(D)	log(W)	ID	1
ET01		x	x				
HM01		x	x			x	
M03		x				x	x
M04	x			x		x	x
TB01	x		x	x			
TP01		x	x			x	
VJ01	x			x		x	x
VJ02	x	x		x			

Optimizing the UI (Supple)

- Two components

  M: How good of a match widget is to the metaphor

  N: Cost of navigation

- Cost of a trace *T* on an interface $R(S_f)$ is the sum of the match *M* and navigational cost *N* of each node

- Minimize cost ($)

Optimizing the UI (Supple++)

$$
\begin{aligned}
\$(R(\mathcal{S}_f, s), \mathcal{T}) &= EMT(R(\mathcal{S}_f, s), \mathcal{T}) \\
&= EMT_{nav}(R(\mathcal{S}_f, s), \mathcal{T}) \\
&\quad + \sum_{n \in \mathcal{S}_f} EMT_{manip}(R(n, s), \mathcal{T})
\end{aligned}
$$

- A more complex cost function based on EMT and minimum target size, *s*

Computing EMT_{manip}

- Many widgets can be operated in different ways depending on the data being controlled
- **ListBox**: might need to scroll, scrolling can be done in various ways: *click, drag, etc*.
- Assign a uniform probability to selectable values, compute expected cost
- $EMT_{manip} = \min(EMT_{manip}$ for each method$)$

Bounding *EMT*$_{nav}$

- Need size bound to use branch & bound
- For a leaf *n* compute the minimum bounding rectangle for compatible widgets
- Propagate lower-bound dimensions up: a layout is at least as wide as the sum of its children

Bounding *EMT*$_{nav}$

- Can compute the shortest possible distance between any pair of elements in a layout
- Lower-bound the time to move from A to B using the shortest distance and largest target size for widgets compatible with B
- Update estimates every time an assignment is made, or undone via backtracking

Low Vision

- Users directly control visual cue size, as in a web browser: 8 discrete zoom levels

- Reflowing the UI to increase/decrease zoom level should be fluid

- Solution: augment the cost function with a penalty to renderings that don't resemble the original (using a distance function)

Computational Cost

- Between 3.6 seconds and 20.6 minutes to compute personalized UIs
- EMT_{nav} estimation reduced runtime from hours to seconds!
- Performance is acceptable, caching can improve the situation

Results

- Personalized UIs allowed participants to complete tasks in 20% less time than the baseline interface

- 50% of participants were fastest with a personalized UI

- 60% of participants rated a personalized UI as easiest to use

Limitations

- Underestimated the time to manipulate list widgets
- Did not take into account visual verification time
- Users impressions did not always align with personalized UI

Future Work

- Extend the motor performance model to better predict list selection times

- Explicitly model the cost of recovering from errors (misplaced clicks, etc)

- Broaden diversity of motor differences represented

- Evaluate the system's ability to adapt to combination impairments

Questions

- Users didn't always prefer the UI that gave them the best performance. How could we include preferences in the model?
- How could the system accommodate changes in the user's abilities?
- What other features might be included in the motor performance model?
- How might changing the pointing semantics help: eg, "snap" to widgets?
- Could some form of SLS perform better in this domain?

Questions 2

- Baseline UIs seem bad, why didn't they compare to baselines optimized using Fitts' law?
- How could this be integrated with existing GUI environments and OSs?
- Is a one-time motor performance test enough to accurately model a user's ability?
- Why use B&B, would IDA* or something else be better?
- Were there enough participants and enough variety for the results to be meaningful?

Thank you!