ACT-Touch

ACT-R gets its Hands on a Multitouch Display

Frank Tamborello & Kristen Greene

Cogscent, LLC

National Institute of Standards and Technology
U.S. Department of Commerce

Welcome to the 21st century.

Now that it’s the future, we don’t always interact with our computers using keyboards and mice, and we’re not always sitting at a desk. Now we often have much smaller displays, virtual keyboards, direct manipulation of interface elements, and mobility. All are likely to have cognitive implications.

For example: To select text in the desktop environment, assuming hands start on the home row, move hand to the mouse, move mouse. In the touch screen screen, move hand—usually then a small menu will appear.

With this significant change in task environment comes the need for our cognitive modeling tools to reflect this development. ACT-Touch is intended to do just that. It consists of motor request extensions and a virtual multitouch display device.

How do we actually get ACT-R up away from the desk and transition from moving hands on keyboards and mice to moving and gesturing on a multitouch display?

Manual Request Extensions

- Based on same building blocks as other ACT-R movement styles
- EPIC's feature preparation & execution framework
  - Features
  - Preparation
  - Execution

Uses the architecture's extant motor feature elements, remixed to be appropriate for multitouch display gestures.

- Based on same building blocks as other ACT-R movement styles
- EPIC's feature preparation & execution framework
  - Features
  - Preparation
  - Execution

The architectural component to ACT-Touch is simply a library of manual motor request extensions as well as a few assumptions to accommodate the different task environment. For example, the hands are no longer at home row because there's no physical keyboard. Instead, hands begin model runs at sides just like how I'm holding this iPad.

Now instead of moving hands around a keyboard or causing a mouse to move, need to move hand across display surface: Move-cursor is replaced by Move-Hand-Touch

Example: Tap

- Like punch
  - Request mechanics
  - Features: hand & finger
- Unlike punch: Fitts from a z-index

Like ACT-R's punch movement style
  1.1. Request schedules an event to call the movement’s method. The movement’s method calls prepare-movement with an instance of the movement’s class.
  1.2. Prep time = (feat-prep-time module) * number of features

  2.1. Unlike punch: Exec time:
ACT-R no longer constrained to holding hands directly over keyboard, so does not use key closure parameter.
 Need movement from somewhere not actually on the display to on the display surface; may also want more variation in the future (e.g., movement start farther than 1” above display surface).

Multitouch Display Device

- Classes & methods to run a task & collect data
- Model's world is a list of virtual widgets

Need an appropriate device to interact with model

1024 x 768 px, landscape orientation

Model starts with hands at either side.

Same basic premise as the list device, discussed in slides distributed with ACT-R.

List of Virtual Widgets
A thing to encapsulate & bind visual and action properties of one spatial region of the task environment
Paired list of visual-location & visual-object chunks
Defines a bounded area for device to receive a gesture and do something with it.
Device method to return a widget that exists at the location of a gesture as specified by the widget's VL chunk.

Example: Tap Widget

- Visual chunks encapsulated with methods to make multitouch interactions work
- Device handler method
  - determines which widget tapped, if any
  - what to do about it

Walk through what the device does when the model outputs a tap.

The device has widgets. Widgets have names, a reference back to the device, a visual-location chunk, and a visual-object chunk.

1. device-handle-tap calls current-widget
2. current-widget checks the list of widgets of the device, returns the tapped widget if the tap occurred inside the area defined by the visual-location of the widget
3. device-handle-tap calls the device's state-checking method with the name of the widget, control passed to the device to do things like update the state of the task & record data

Conclusions:
Now we have a basic platform for modeling a variety of touchscreen gesture interactions.

Issues & Next Goals

- Validation
- EPIC's HFRT model may not be directly applicable at all times
- See-through hand

1.1. Some assumptions borne from directly adapting EPIC's model become kind of wacky in the touchscreen domain. E.G.

1.1.1. What's the target of a swipe? …or a pinch? Fitts' Law requires a target width.

1.1.2. Adding a feature to capture number of fingers used in a gesture adds preparation time that probably shouldn't be added, e.g., for swipe. But are two- and three-fingered swipes really different movement styles?

Eventually:
motor learning
manual fatigue (e.g., customs officer during 8hr shift)

Code Availability

- cogscent.com

ACT-Touch code is available to the public domain at cogscent.com

Acknowledgments

- Mike Byrne
- Dan Bothell
- Ross Michaels
- bws.nist.gov

1. Mentoring & library code
2. Technical Support
3. Project Guidance

This work is funded by a Measurement Science & Engineering grant from the National Institute of Standards and Technology for their biometric web services project. This project will allow biometric data collection to be freed from constraints imposed by the desktop computing environment.