ACT-Touch Reference Manual

Working Draft

Franklin P. Tamborello
Cogscent, LLC

Kristen K. Greene
National Institute of Standards and Technology

Preface

This document describes the manual request extensions to ACT-R 6 (hereafter, ACT-R) provided by ACT-Touch as well as the included virtual multitouch display device. As ACT-Touch is intended to serve as an extension to ACT-R, this document follows the formatting conventions of the ACT-R Reference Manual by Dan Bothell. A description of that notation is quoted verbatim from that document for the reader's convenience. The scope of this manual is restricted to the ACT-Touch distribution. This document is written with the assumption that the reader is familiar with ACT-R and programming in Lisp. Please refer to the ACT-R 6.0 Reference Manual for all other ACT-R issues.

Acknowledgments

- Mike Byrne, Rice University
  - Public release of library code upon which ACT-Touch depends
- Dan Bothell, Carnegie Mellon University
  - ACT-R technical support
- Ross Micheals, National Institute of Standards and Technology
  - Project guidance
- NIST
  - This work is funded by Measurement Science & Engineering grant 60NANB12D134 from the National Institute of Standards and Technology in support of their Biometric Web Services project (bws.nist.gov).

Introduction

ACT-Touch is a set of manual motor request extensions for ACT-R. Facility with programming models for ACT-R is assumed throughout this document. These manual request extensions constitute theoretical claims predicting motor preparation and execution times for certain manual gestures commonly used with multitouch computer displays. These manual request extensions follow ACT-R's theoretical claims about how cognition interfaces with action to output information from the human to the environment, which in turn originated with the EPIC architecture. This document is meant as a practical guide to using ACT-Touch; it does not focus on theoretical developments.

As ACT-Touch extends ACT-R's framework with additional manual movement vocabulary, many movement styles are analogous to extant ACT-R movement styles in the sense that a movement is composed of a certain set of features which specify the movement such as a hand, a finger, a direction and a distance. Consequently ACT-Touch's movement styles are subject to the same constraints and caveats as ACT-R's, e.g., finger positions that would be physically impossible for any physical human hand to attain are specifiable for a cognitive model, and so it is up to the modeler to consider such things.

Unlike ACT-R, all distances in ACT-Touch and its virtual multitouch display are specified in pixels. The virtual multitouch display measures 1,024 pixels wide by 768 pixels tall. The model's default positions for its hands are at either side of the display, approximately centered vertically.

ACT-Touch is implemented as Lisp code that is meant to load with ACT-R's software. ACT-Touch can be downloaded as a single archive from Cogscent, LLC's website, http://www.cogscent.com. The archive contains act-touch.lisp, which is the set of manual request extensions; support files implementing a demonstration ACT-R device to handle ACT-Touch's manual requests; a demonstration model; and this reference manual. Direct technical support inquiries regarding ACT-Touch to Frank Tamborello at frank.tamborello@cogscent.com.

Notations in the Documentation[1]

When describing the commands' syntax the following conventions will be used:

- items appearing in **bold** are to be entered verbatim
- items appearing in *italics* take user-supplied values
- items enclosed in {curly braces} are optional
- * indicates that any number of items may be supplied
- + indicates that one or more items may be supplied
- | indicates a choice between options which are enclosed in [square brackets]
- (parentheses) denote that the enclosed items are to be in a list
- a pair of items enclosed in <angle brackets> denote a cons cell with the first the car and the second the cdr
- -> indicates that calling the command on the left of the arrow will return the item to the right of the arrow
- **::=** indicates that the item on the left of that symbol is of the form given by the expression on the right

When examples are provided for the commands they are shown as if they have been evaluated at a Lisp prompt. The prompt that is shown prior to the command indicates additional information about the examples. There are three types of prompts that are used in the examples:

- A prompt with just the character '>' indicates that it is an individual example – independent of those preceding or following it.
- A prompt with a number followed by '>', for example 2> means that the example is part of a sequence of calls which were evaluated and the result depends on the preceding examples. For any given sequence of calls in an example the numbering will start at 1 and increase by 1 with each new example in the sequence.
- A prompt with the letter E preceding the '>', E>, indicates that this is an example which is either incorrect or was evaluated in a context where the call results in an error or warning. This is done to show examples of the warnings and errors that can occur.

In the description of some commands it will describe a parameter or return value as a generalized boolean. What that means is that the value is used to represent a truth value – either true/successful or false/failure. If the value is the symbol **nil** then it represents false and all other values represent true. When a generalized boolean is returned by one of the commands, one should not make any assumptions about the returned value for the true case. Sometimes the true value may look like it provides additional information, but if that is not specified in the command's description then it is not guaranteed to hold for all cases or across updates to the command.

---

[1] Quoted from the ACT-R 6.0 Reference Manual

Loading ACT-Touch

The most straight-forward way to load ACT-Touch is to simply place its files in ACT-R's user-loads folder before loading ACT-R. This will load ACT-Touch at the end of the ACT-R loading process. Load act-touch.lisp alone if you don't want to use the included virtual multitouch display device or demo model. However, note that whatever device you use must supply an index-z class slot (pixels at 72 ppi). Load misc-lib.lisp, virtual-experiment-window.lisp, and virtual-multitouch-device.lisp if you wish to use the included virtual multitouch display device. Act-touch-demo-model.lisp has an example model.

Manual Request Extensions to ACT-R

Isa tap
hand [***left*** **|** ***right*** **]**
finger [***index*** **|** ***middle*** **|** ***ring*** **|** ***pinkie*** **|** ***thumb*** **]**

This request will execute a tap action for the specified finger on the specified hand. This will result in the finger moving toward and momentarily contacting the surface of the multitouch display directly under the finger's current location. This is analogous to ACT-R's punch command. These are the actions which will be shown in the trace for a tap action indicating the request being received, the preparation of the features completing, the initiation time having passed, the actual contacting of the display surface which is currently under that finger (showing the global XY screen coordinates tapped by the finger on the virtual multitouch display), and the time to finish the execution of the action (returning the finger to its starting position where it is ready to act again):

```
     0.050   MOTOR                 TAP HAND RIGHT FINGER INDEX
…
     0.200   MOTOR                 PREPARATION-COMPLETE
…
     0.250   MOTOR                 INITIATION-COMPLETE
…
     0.713   MOTOR                 DEVICE-HANDLE-TAP #<MULTITOUCH-DISPLAY #x302001E3A23D>
#(500 300) RIGHT INDEX
…
     0.763   MOTOR                 FINISH-MOVEMENT
```

Isa peck-tap
hand [***left*** **|** ***right*** **]**
finger [***index*** **|** ***middle*** **|** ***ring*** **|** ***pinkie*** **|** ***thumb*** **]**
r *distance*
theta *direction*

Analogous to the peck style movement, but resulting in a tap gesture on the multitouch display device.

```
     0.050   MOTOR                 PECK-TAP HAND RIGHT FINGER INDEX R 100 THETA 1
…
     0.300   MOTOR                 PREPARATION-COMPLETE
…
     0.350   MOTOR                 INITIATION-COMPLETE
…
     1.312   MOTOR                 DEVICE-HANDLE-PECK-TAP #<MULTITOUCH-DISPLAY
#x302001D7751D> #(554 384)
Model peck-tapped (#(554 384)).
…
     1.362   MOTOR                 FINISH-MOVEMENT
```

Isa peck-recoil-tap
hand [***left*** **|** ***right*** **]**
finger [***index*** **|** ***middle*** **|** ***ring*** **|** ***pinkie*** **|** ***thumb*** **]**
r *distance*
theta *direction*

Analogous to the peck-recoil style movement, but resulting in a tap gesture on the multitouch display device.

```
     0.050   MOTOR                  PECK-RECOIL-TAP HAND RIGHT FINGER INDEX R 100 THETA 1
…
     0.300   MOTOR                  PREPARATION-COMPLETE
…
     0.350   MOTOR                  INITIATION-COMPLETE
…
     1.312   MOTOR                  DEVICE-HANDLE-PECK-RECOIL-TAP #<MULTITOUCH-DISPLAY
#x302001BB2F3D> #(1078 384)
Model peck-recoil-tapped (#(1078 384)).
…
     2.325   MOTOR                  FINISH-MOVEMENT
```

Isa tap-hold
hand [***left*** **|** ***right*** **]**
finger [***index*** **|** ***middle*** **|** ***ring*** **|** ***pinkie*** **|** ***thumb*** **]**

This movement style results in the model tapping and holding the specified finger on the surface of the multitouch display device until the model requests the tap-release movement style.

```
     0.050   MOTOR                  TAP-HOLD HAND RIGHT FINGER INDEX
…
     0.200   MOTOR                  PREPARATION-COMPLETE
…
     0.250   MOTOR                  INITIATION-COMPLETE
…
     0.713   MOTOR                  DEVICE-HANDLE-TAP-HOLD #<MULTITOUCH-DISPLAY
#x302001C5984D> RIGHT INDEX
Model tap-held (RIGHT INDEX).
…
     0.763   MOTOR                  FINISH-MOVEMENT
```

Isa tap-release
hand [***left*** **|** ***right*** **]**
finger [***index*** **|** ***middle*** **|** ***ring*** **|** ***pinkie*** **|** ***thumb*** **]**

When the model is already holding a finger against the surface of the multitouch display device (e.g., with a tap-hold), this movement style will release the finger from the display and return it to its default distance from the display at the current X, Y coordinates.

```
     0.050   MOTOR                  TAP-RELEASE HAND RIGHT FINGER INDEX
…
     0.200   MOTOR                  PREPARATION-COMPLETE
…
     0.250   MOTOR                  INITIATION-COMPLETE
…
     0.350   MOTOR                  DEVICE-HANDLE-TAP-RELEASE #<MULTITOUCH-DISPLAY
#x302001DD1A6D> RIGHT INDEX
Model tap-released (RIGHT INDEX).
…
     0.400   MOTOR                  FINISH-MOVEMENT
```

If the model does not already have a finger on the display surface (index-z = 0), this error will result:

```
#|Warning: Finger must already be held against the surface
                     of the multitouch display. |#
```

Isa tap-drag-release
hand [***left*** **|** ***right*** **]**
finger [***index*** **|** ***middle*** **|** ***ring*** **|** ***pinkie*** **|** ***thumb*** **]**
r *distance*
theta *direction*

The model will tap-hold the display surface under its finger, move its finger the specified distance and direction without breaking contact with the display surface, and then release the finger from the display.

```
     0.050   MOTOR                  TAP-DRAG-RELEASE HAND RIGHT FINGER INDEX R 100 THETA 1
…
     0.300   MOTOR                  PREPARATION-COMPLETE
…
     0.350   MOTOR                  INITIATION-COMPLETE
…
     1.312   MOTOR                  DEVICE-HANDLE-TAP-DRAG-RELEASE #<MULTITOUCH-DISPLAY
#x302001E92F3D> RIGHT INDEX 100 1
Model tap-drag-released (RIGHT INDEX 100 1).
…
     1.826   MOTOR                  FINISH-MOVEMENT
```

Isa swipe
hand [***left*** **|** ***right*** **]**
finger [***index*** **|** ***middle*** **|** ***ring*** **|** ***pinkie*** **|** ***thumb*** **]**
r *distance*
theta *direction*
num-fngrs *integer*

The model moves the specified number of fingers, starting with the specified finger and incrementing from index to pinkie and thumb, onto the display and then moves the specified distance and direction, then releases the fingers from the display. Num-fngrs defaults to 1.

```
     0.050   MOTOR                  SWIPE HAND RIGHT FINGER INDEX R 100 THETA 1 NUM-FNGRS 3
…
     0.350   MOTOR                  PREPARATION-COMPLETE
…
     0.400   MOTOR                  INITIATION-COMPLETE
…
     1.362   MOTOR                  DEVICE-HANDLE-SWIPE #<MULTITOUCH-DISPLAY #x302001E585ED>
#(500 300) #(554 384)
…
     2.375   MOTOR                  FINISH-MOVEMENT
```

Isa pinch
hand [***left*** **|** ***right*** **]**
finger [***index*** **|** ***middle*** **|** ***ring*** **|** ***pinkie*** **]**
start-width *integer*
end-width *integer*

The model moves the specified finger and thumb onto the surface of the display, then moves them together or apart by the difference between the specified start- and end-widths, then releases them from the display. Start- and end-widths are in pixels.

```
     0.050   MOTOR                  PINCH HAND RIGHT FINGER INDEX START-WIDTH 200 END-WIDTH 0
…
     0.300   MOTOR                  PREPARATION-COMPLETE
…
     0.350   MOTOR                  INITIATION-COMPLETE
…
     0.942   MOTOR                  DEVICE-HANDLE-PINCH #<MULTITOUCH-DISPLAY #x302001E7E39D>
#(500 300) RIGHT INDEX 200 0
…
     1.455   MOTOR                  FINISH-MOVEMENT
```

Isa rotate
hand [***left*** **|** ***right*** **]**
finger [***index*** **|** ***middle*** **|** ***ring*** **|** ***pinkie*** **]**
rotation *direction*

This movement request results in a movement of the specified finger and the thumb to the display, moves them rotationally by the specified direction (in radians), then releases them.

```
     0.050   MOTOR                 ROTATE HAND RIGHT FINGER INDEX ROTATION 1
…
     0.250   MOTOR                 PREPARATION-COMPLETE
…
     0.300   MOTOR                 INITIATION-COMPLETE
…
     1.153   MOTOR                 DEVICE-HANDLE-ROTATE #<MULTITOUCH-DISPLAY #x302001DBB72D>
#(500 300) 36
…
     1.666   MOTOR                 FINISH-MOVEMENT
```

Isa move-hand-touch
[**object** *object* | **loc** *location*]

Analogous to move-cursor, this request will result in a ply-style movement of the model's right hand. That ply will move the index finger to either the object (which must be a chunk which is a subtype of visual-object) or location (which must be a chunk which is a subtype of visual-location). The trace indicates the distance and direction moved.

```
     0.100   MOTOR                 MOVE-HAND-TOUCH OBJECT NIL LOC VISUAL-LOCATION0-0-0
…
     0.300   MOTOR                 PREPARATION-COMPLETE
…
     0.350   MOTOR                 INITIATION-COMPLETE
…
     0.502   MOTOR                 MOVE-A-HAND RIGHT 544.76416 -2.6188035
…
     0.552   MOTOR                 FINISH-MOVEMENT
```

Virtual Multitouch Display Device

The virtual multitouch display device included with ACT-Touch is a basic ACT-R device built to present a virtual visual environment to an ACT-R model and receive ACT-R's touch gesture motor movements. It includes some code that is specific to the demonstration model and one class slot that is important for ACT-Touch functionality. The index-z slot of the multitouch-display class is used for the ACT-Touch manual request extensions as a measure of distance from the model's index finger to the multitouch display surface, in pixels at 72 ppi. Whatever device is used with ACT-Touch, index-z must be supplied as a slot of the device's object class.

This section provides only superficial coverage regarding the mechanics of the device. See the ACT-R documentation for details regarding how to build your own device for ACT-R. This section includes some details about ACT-Touch's device interface methods as well as some discussion about how to modify ACT-Touch's included multitouch device. We assume familiarity with Common Lisp's object system, so object-oriented programming in Lisp will not be discussed here.

The virtual multitouch display is a slightly more sophisticated version of the list device presented in extending-actr.pdf distributed with ACT-R. Like the list device, it uses a paired list of visual-location and visual-object chunks as ACT-R's visual world. Unlike the list device, it uses objects of a virtual-widget class to encapsulate those chunks with data used by the experiment code to control the state of the simulated task environment and perform some action according to the appropriate device handler methods. Virtual-multitouch-device.lisp also contains all the device handler methods for what the experiment code should do when the model outputs each of ACT-Touch's manual request extension types. Familiarity with CLOS and ACT-R device programming are helpful for adapting or replacing the virtual multitouch display device. Both topics are covered in-depth elsewhere, namely in ANSI Common Lisp by Paul Graham and the ACT-R Reference Manual, respectively. However, this section provides basic information about ACT-Touch's virtual multitouch display device that will help you to modify it.

Device Classes

multitouch-display (procedure-window)

visual-world
index-z
widgets
:state-vec #(:TAP1 :TAP2 :TAP3 :SWIPE :PINCH :REVERSE-PINCH :ROTATE-CW :ROTATE-CCW)

The multitouch-display class is a child of the procedure-window class. It adds slots visual-world, index-z, and widgets. Upon initialization it assigns a default value to the state-vec slot that it inherits from procedure-window. That default value is a vector of keywords, each representing a discrete state of the task environment—first step is :TAP1, second step is :TAP2, etc. These keywords correspond to the nick-names that you will give to each of the virtual-widgets in the ACT-R device. Change this state-vec to change the steps of the task.

virtual-widget ()
nick-name
vwindow
vis-loc
vis-obj
action-type

The virtual-widget class is a parent class to act as a container for the visual-location and visual-object chunks that comprise a model's visual environment. It also is to receive motor actions from the model so that the device can interact with the model. It is subclassed for each style of movement (e.g., tap, swipe). The nick-name slot takes a keyword with which the experiment software may refer to that widget (e.g., :tap1). Vwindow refers to the virtual-window within which the virtual-widget appears (i.e., the virtual-multitouch-device). Vis-loc and vis-obj are the visual chunks that ACT-R will access when it constructs its visicon and the visual-location's associated visual-object. The action-type specifies the movement style the virtual-widget is to receive (e.g., tap).

Device Methods & Functions

initialize-instance :after ((wind multitouch-display) &key)

This method sets up the multitouch-display device with its visual-location and visual-object chunks, which it uses to construct the widgets which will constitute the model's simulated world. The first subexpression within the let expression defines the visual-location chunks. The slots, such as screen-x and screen-y, determine where the widgets are placed within the ACT-R device. Change the value of the visual-location slots here to change where widgets are located and what they look like. The second subexpression defines some textual labels to appear within the device. The third subexpression defines the visual-object chunks that ACT-R's device methods will pair with each of the visual-location chunks defined for the device. The device methods (described below) determine how when ACT-R moves visual-attention to a visual-location, ACT-R gets the appropriate visual-object chunk. In this third subexpression, multitouch-display's initialize-instance after method defines these visual-object chunks.

The fourth subexpression defines the widgets that comprise the ACT-R device. There are different types of widgets, each of which correspond to the various gestures of the multitouch command vocabulary (e.g., tap, pinch). Change the first argument to change what type of widget, and thus which command, to use. This should be a symbol naming a virtual-widget subclass, such as *'tap-widget.* Each widget takes the visual-location and visual-object chunks defined in the first and third subexpressions and assigns them to widgets, interface objects to receive model input and perform some task-relevant function. These widgets take the visual-location and visual-object chunks according to the position indicated by the second argument of the *nth* expressions—e.g., (nth 0 vis-locs) refers to the first visual-location chunk. Nick-names assigned here are to correspond to states of the multitouch-display, steps of the model's task (e.g., :TAP1). This concludes the let expression.

Next, multitouch-display's initialize-instance after method sets the multitouch-display's visual-world slot to be a paired list of the visual-location and visual-object chunks and its widgets slot to be the

widgets just defined. After that the method calls some model setup model functions that should be familiar to any ACT-R modelers, such as install-device and proc-display.