Interpreting Multitouch Gestures

Michael Schuweiler

Division of Science and Mathematics
University of Minnesota, Morris

December 6th 2014
Senior Seminar Conference, Morris, MN

Interpreting Multitouch Gestures

1. Background
2. Gestures
3. Proton
   - Gesture Notation
   - Static Analysis Process
   - Architecture
4. Proton++
   - New Notation
   - User Study
5. Conclusion

Background

- Multitouch systems are readily available
- Phones, tablets, computers, etc
- Touches

Background

- Basic low level touches
  - Down
  - Move
  - Up
- Composing these touch events into sequences
- Most systems call these sequences gestures

Interpreting Multitouch Gestures

1. Background
2. **Gestures**
3. Proton
   - Gesture Notation
   - Static Analysis Process
   - Architecture
4. Proton++
   - New Notation
   - User Study
5. Conclusion

Gestures

- Definition: Functionality allowing a touchscreen, track pad, etc. to register multiple points of contact made on the surface simultaneously
- Tools currently available
  - Proton
  - Proton++
  - Others including: Gesture Studio, Gesture Coder, GeForMT, etc.

Interpreting Multitouch Gestures

1 Background

2 Gestures

3 Proton

- Gesture Notation
- Static Analysis Process
- Architecture

4 Proton++

- New Notation
- User Study

5 Conclusion

Proton

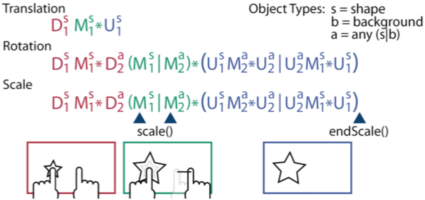
- Multitouch framework
- What is its main purpose?

Proton: Regular Expressions

- Proton uses regular expressions to denote gestures
- Kleene star * = zero or more (any)
- | = or

$$E^{O_{Type}}_{T_{ID}}$$

E ϵ {D,M,U}, O_{type}: object hit, T_{id}: touch ID



Translation

$D^s_1\ M^s_1{*}U^s_1$

Object Types: s = shape
b = background
a = any (s|b)

Rotation

$D^s_1\ M^s_1{*}D^a_2\ (M^s_1|M^a_2){*}(U^s_1M^a_2{*}U^a_2\,|\,U^a_2M^s_1{*}U^s_1)$

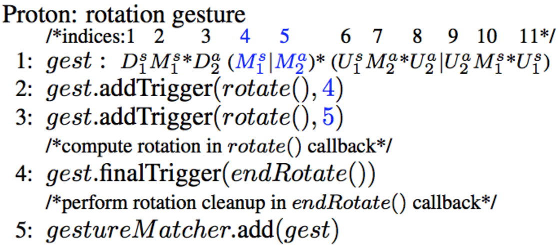
Scale

$D^s_1\ M^s_1{*}D^a_2\ (M^s_1|M^a_2){*}(U^s_1M^a_2{*}U^a_2\,|\,U^a_2M^s_1{*}U^s_1)$

Figure: Proton regular expressions

Proton: Regular Expressions

- Indices
- Triggers applied on 4, 5
- Callback functions



Proton: rotation gesture

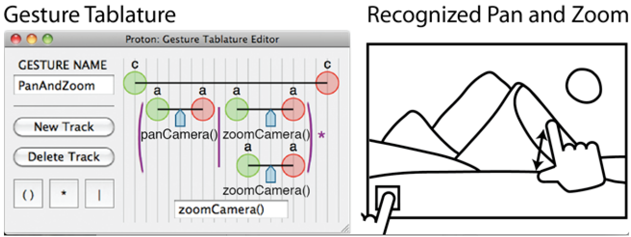
```
    /*indices:1   2     3    4    5       6   7    8    9   10   11*/
$gest : D_1^s M_1^s{*} D_2^a \, (M_1^s | M_2^a){*} \, (U_1^s M_2^a{*} U_2^a | U_2^a M_1^s{*} U_1^s)$
$gest$.addTrigger($rotate()$, 4)
$gest$.addTrigger($rotate()$, 5)
    /*compute rotation in $rotate()$ callback*/
$gest$.finalTrigger($endRotate()$)
    /*perform rotation cleanup in $endRotate()$ callback*/
$gestureMatcher$.add($gest$)
```

Figure: Regular expression triggers callbacks

Proton: Gesture Tablature

- Guitar tablature
- $\triangle$ = panCamera()
- $\triangle$ = zoomCamera()

$$D_1^c M_1^c * (D_2^a (M_1^c | M_{2_\triangle}^a) * U_2^a M_1^c *$$
$$| D_2^a (M_1^c | M_2^a) * D_3^a (M_1^c | M_{2_\triangle}^a | M_{3_\triangle}^a) * (U_3^a (M_1^c | M_2^a) *$$
$$U_2^a M_1^c * | U_2^a (M_1^c | M_3^a) * U_3^a M_1^c *)) * U_1^c$$



GESTURE NAME

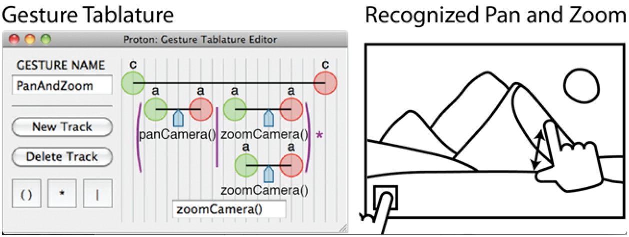
PanAndZoom

Figure: Proton gesture tablature creation tool

Proton: Gesture Tablature

- Guitar tablature
- $\triangle$ = panCamera()
- $\triangle$ = zoomCamera()

$$D_1^c M_1^c * (D_2^a(M_1^c|M_{2_\triangle}^a) * U_2^a M_1^c * |D_2^a(M_1^c|M_2^a) * D_3^a(M_1^c|M_{2_\triangle}^a|M_{3_\triangle}^a) * (U_3^a(M_1^c|M_2^a) * U_2^a M_1^c * |U_2^a(M_1^c|M_3^a) * U_3^a M_1^c *)) * U_1^c$$



Gesture Tablature

Recognized Pan and Zoom

GESTURE NAME: PanAndZoom

Figure: Proton gesture tablature creation tool

Proton: Gesture Tablature

- Guitar tablature
- $\triangle$ = panCamera()
- $\triangle$ = zoomCamera()

$$D_1^c M_1^c * (D_2^a(M_1^c|M_{2_\triangle}^a) * U_2^a M_1^c *$$
$$|D_2^a(M_1^c|M_2^a) * D_3^a(M_1^c|M_{2_\triangle}^a|M_{3_\triangle}^a) * (U_3^a(M_1^c|M_2^a) * U_2^a M_1^c *$$
$$|U_2^a(M_1^c|M_3^a) * U_3^a M_1^c *)) * U_1^c$$

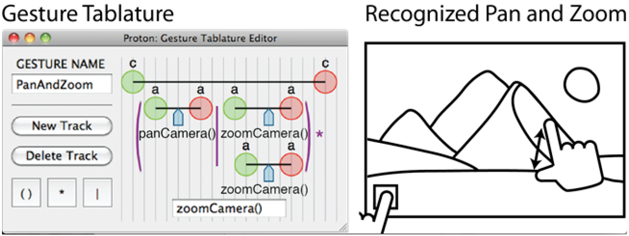
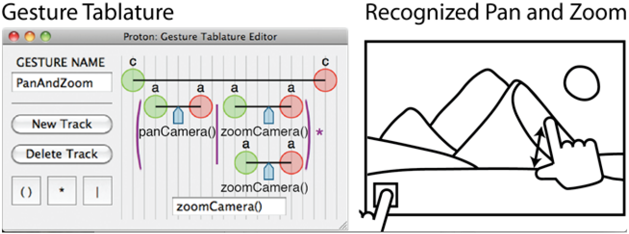


Figure: Proton gesture tablature creation tool

Proton: Gesture Tablature

- Guitar tablature
- $\triangle$ = panCamera()
- $\triangle$ = zoomCamera()

$$D^c_1 M^c_1 * (D^a_2(M^c_1|M^a_{2_\triangle}) * U^a_2 M^c_1 *$$
$$|D^a_2(M^c_1|M^a_2) * D^a_3(M^c_1|M^a_{2_\triangle}|M^a_{3_\triangle}) * (U^a_3(M^c_1|M^a_2) * U^a_2 M^c_1 *$$
$$|U^a_2(M^c_1|M^a_3) * U^a_3 M^c_1 *)) * U^c_1$$



Gesture Tablature

Recognized Pan and Zoom

Proton: Gesture Tablature Editor

GESTURE NAME

PanAndZoom

Figure: Proton gesture tablature creation tool

Proton: Gesture Tablature

- Guitar tablature
- $\color{blue}\triangle$ = panCamera()
- $\color{magenta}\triangle$ = zoomCamera()

$$D_1^c M_1^c * (D_2^a(M_1^c|M_{2_{\color{blue}\triangle}}^a) * U_2^a M_1^c *$$
$$|D_2^a(M_1^c|M_2^a) * D_3^a(M_1^c|M_{2_{\color{magenta}\triangle}}^a|M_{3_{\color{magenta}\triangle}}^a) * (U_3^a(M_1^c|M_2^a) * U_2^a M_1^c *$$
$$|{\color{red}U_2^a(M_1^c|M_3^a) * U_3^a M_1^c *)) * U_1^c}$$

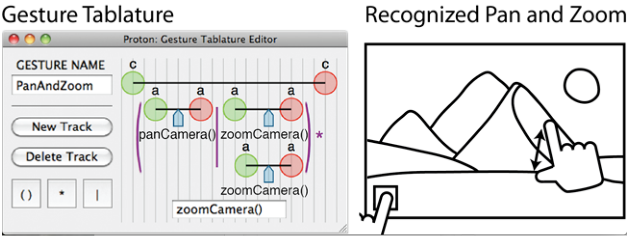


Figure: Proton gesture tablature creation tool

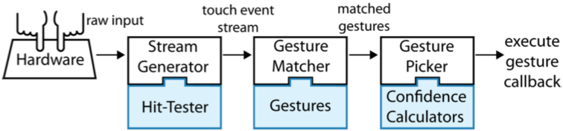
Static Analysis of Gesture Conflicts

- Regular expressions that have the same prefixes
- Ambiguity resulting from this
- Caught at compile time for developers
- Helpful tool, possible through the use of regular expressions

Proton: Development to User Input

- Tablature supports construction of gestures
- Gesture sets
- Proton then uses these gesture sets to go from user input to associated callback functions
- Possible through Proton's architecture

Proton: Architecture



- Three main components
  - Stream generator
  - Gesture matcher
  - Gesture picker

Proton: Architecture

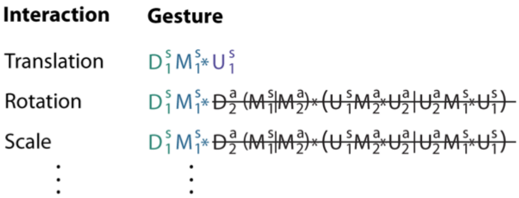
Stream Generator: Hit-Tester

- Used to determine the O_{type} hit
- Developers must include all objects possibly being hit in a scene

Proton: Architecture

Gesture Matcher: Gestures

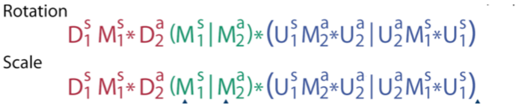
- Developers must create a gesture set with all possible gestures
- Matches incoming stream with gesture set



Interaction	Gesture
Translation	$D_1^s M_1^s{*} U_1^s$
Rotation	$D_1^s M_1^s{*}$ ~~$D_2^a (M_1^s \mid M_2^a){*} (U_1^s M_2^a{*} U_2^a \mid U_2^a M_1^s{*} U_1^s)$~~
Scale	$D_1^s M_1^s{*}$ ~~$D_2^a (M_1^s \mid M_2^a){*} (U_1^s M_2^a{*} U_2^a \mid U_2^a M_1^s{*} U_1^s)$~~
⋮	⋮

Proton: Architecture

Gesture Picker: Confidence Calculators



Rotation

$$D_1^s M_1^s {*} D_2^a (M_1^s | M_2^a) {*} (U_1^s M_2^a {*} U_2^a | U_2^a M_1^s {*} U_1^s)$$

Scale

$$D_1^s M_1^s {*} D_2^a (M_1^s | M_2^a) {*} (U_1^s M_2^a {*} U_2^a | U_2^a M_1^s {*} U_1^s)$$

Figure: Proton expressions for rotate and scale

- Confidence calculator applies a confidence score between 0 and 1 to each expression
- Highest score is returned and executed
- Determined from attributes associated with these expressions
- Are the touches in scale moving away or towards each other?

Interpreting Multitouch Gestures

1 Background

2 Gestures

3 Proton

- Gesture Notation
- Static Analysis Process
- Architecture

4 Proton++

- New Notation
- User Study

5 Conclusion

Proton++

- Extended framework built on Proton
- Uses much of the same processes
- 5 new touch attributes
  - Direction
  - Pinch
  - Touch area
  - Finger orientation
  - Screen location

Proton++: Whats new

Proton++ has very important upgrades

- These new attributes allow for more dynamic regular expressions
- Stream generator splits into multiple streams
- Allows for multiple user applications

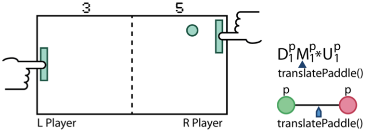
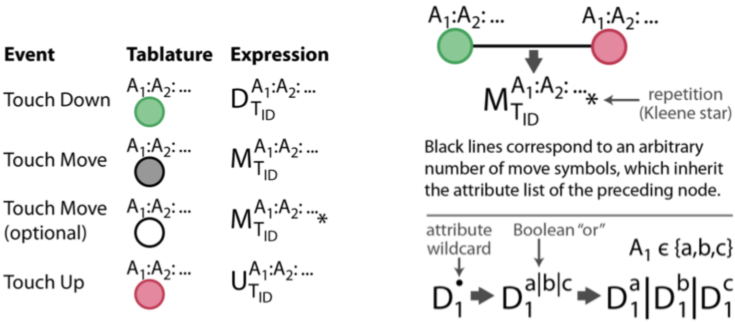


Figure: Pong application where Proton++ splits stream generator to allow for two gesture matchers to run at the same time

Proton++: New Notation

New form: $E_{T_{ID}}^{A_1:A_2:A_3\dots}$

- $E \in \{D, M, U\}$
- A_n are the attributes linked to the touch
- T_{ID} Touch-id



Event	Tablature	Expression
Touch Down	A_1:A_2: ... 🟢	$D_{T_{ID}}^{A_1:A_2:\dots}$
Touch Move	A_1:A_2: ... ●	$M_{T_{ID}}^{A_1:A_2:\dots}$
Touch Move (optional)	A_1:A_2: ... ○	$M_{T_{ID}}^{A_1:A_2:\dots *}$
Touch Up	A_1:A_2: ... 🔴	$U_{T_{ID}}^{A_1:A_2:\dots}$

Black lines correspond to an arbitrary number of move symbols, which inherit the attribute list of the preceding node.

attribute wildcard
Boolean "or"
$A_1 \in \{a,b,c\}$
$D_1^{\bullet} \rightarrow D_1^{a|b|c} \rightarrow D_1^a|D_1^b|D_1^c$

Figure: Proton++ syntax for tablature and regular expression

Examples of Proton++

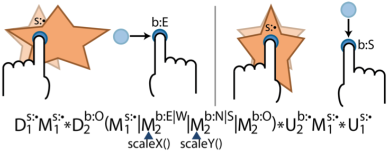

s:•
b:E
s:•
b:S
$D_1^{s:\bullet} M_1^{s:\bullet} * D_2^{b:O} (M_1^{s:\bullet} | M_2^{b:E|W} | M_2^{b:N|S} | M_2^{b:O}) * U_2^{b:\bullet} M_1^{s:\bullet} * U_1^{s:\bullet}$
scaleX()
scaleY()

Figure: Proton++ regular expression for scale gesture

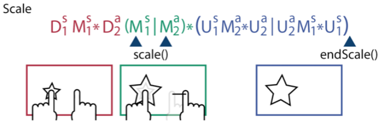
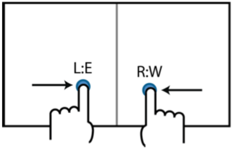

Scale
$D_1^s M_1^s * D_2^a (M_1^s | M_2^a) * (U_1^s M_2^a * U_2^a | U_2^a M_1^s * U_1^s)$
scale()
endScale()

Figure: Proton scale regular expression

Examples of Proton++



$$D_1^{L:O}M_1^{L:O}{*}D_2^{R:O}(M_1^{L:O}|M_2^{R:O}){*}\ldots$$
$$(M_1^{L:E}(M_1^{L:E}|M_2^{R:O}){*}M_2^{R:W}|M_2^{R:W}(M_1^{L:O}|M_2^{R:W}){*}M_1^{L:E})\ldots$$
$$(M_1^{L:O|E}|M_2^{R:O|W}){*}(U_1^{L:O|E}M_2^{R:O|W}{*}U_2^{R:O|W}|U_2^{R:O|W}M_1^{L:O|E}{*}U_1^{L:O|E})$$

executeMenuCommand()

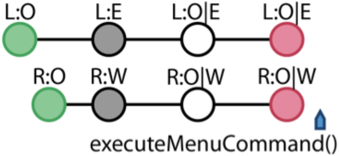


Figure: Figure at the top is the regular expression for opening a menu with two hands, and bottom is that regular expression in tablature form. Possible through the use of direction.

Proton++: User Study

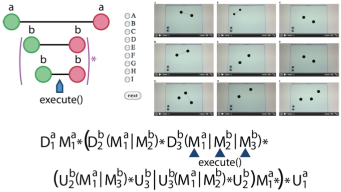
- Recruited 12 experienced programmers
- Checked for correctness and time of understanding
- Two parts
  - Part 1: Basic gestures
  - Part 2: Trajectory

Proton++: User Study Part 1

Part 1:

- Different gesture representations affect participant's understanding of basic gestures
- 3 different representations
  - Tablature
  - Regular expressions
  - iOS event-handling

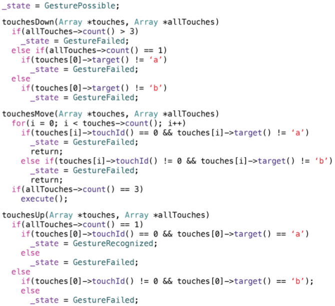
Proton++: User Study Part 1



$$D^a_1\, M^a_1{*}\left(D^b_2\,(M^a_1|M^b_2){*}\,D^b_3(\underset{\blacktriangle}{M^a_1}|\underset{\blacktriangle}{M^b_2}|\underset{\blacktriangle}{M^b_3}){*}\right.$$

$$\left.(U^b_2(M^a_1|M^b_3){*}U^b_3\,|\,U^b_3(M^a_1|M^b_2){*}U^b_2)M^a_1{*}\right){*}\,U^a_1$$

Figure: Gesture Tablature and Regular Expression



```
_state = GesturePossible;

touchesDown(Array *touches, Array *allTouches)
  if(allTouches->count() > 3)
    _state = GestureFailed;
  else if(allTouches->count() == 1)
    if(touches[0]->target() != ‘a’)
      _state = GestureFailed;
  else
    if(touches[0]->target() != ‘b’)
      _state = GestureFailed;

touchesMove(Array *touches, Array *allTouches)
  for(i = 0; i < touches->count(); i++)
    if(touches[i]->touchId() == 0 && touches[i]->target() != ‘a’)
      _state = GestureFailed;
      return;
    else if(touches[i]->touchId() != 0 && touches[i]->target() != ‘b’)
      _state = GestureFailed;
      return;
  if(allTouches->count() == 3)
    execute();

touchesUp(Array *touches, Array *allTouches)
  if(allTouches->count() == 1)
    if(touches[0]->touchId() == 0 && touches[0]->target() == ‘a’)
      _state = GestureRecognized;
    else
      _state = GestureFailed;
  else
    if(touches[0]->touchId() != 0 && touches[0]->target() == ‘b’);
    else
      _state = GestureFailed;
```

Figure: iOS Event-handling

Proton++: User Study Part 1

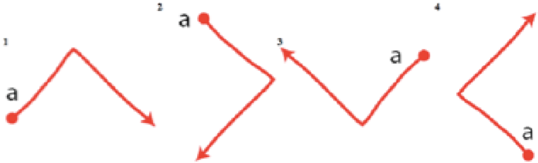
Results:

- Tablature recognition: 23.50s
- Regular expression recognition: 49.25s
- iOS event-handling recognition: 110.99s

Proton++: User Study Part 2

Part 2:

- Identify gestures that use both hit-target attributes and the direction attribute for specifying trajectory
- Presented a gesture, and set of four images
- Asked to identify corresponding gesture to correct image



Proton++: User Study Part 2

Results:

- Gesture Tablature: 17.82s
- Regular expression: 35.49s
- iOS event-handling: 75.29s

1 Background

2 Gestures

3 Proton

- Gesture Notation
- Static Analysis Process
- Architecture

4 Proton++

- New Notation
- User Study

5 Conclusion

Conclusion

- Background and gestures
- Proton
  - Proton regular expressions
  - Proton gesture tablature
  - Architecture
- Proton++
  - New attributes
  - User study

Questions

Bibliography

- K. Kin, B. Hartmann, T. DeRose, and M. Agrawala. Proton++: A customizable declarative multitouch framework. In Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology, UIST 12, pages 477486, New York, NY, USA, 2012. ACM.
- K. Kin, B. Hartmann, T. DeRose, and M. Agrawala. Proton: Multitouch gestures as regular expressions. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 12, pages 28852894, New York, NY, USA, 2012. ACM.