Evolution of Swarm User Interfaces

Björn Hartmann (EECS)
Sensor Swarms Can Engage Us Directly In Our World.
HCI Research in Sensor Swarms

**Swarm UIs**
User interfaces built using sensor swarms

**Swarm UI Design Tools**
Authoring tools that enable [developers|designers|hobbyists] to create swarm UIs
Swarm UIs
Swarm UIs
Swarm UIs

Fixed Controller, Fixed Scheme

Inventing a Control Scheme IS the Game
SLABS
Arrays of Pressure Sensitive Touch Pads
Wessel, CNMAT
Exploring the Space

- New fall 2012 graduate course “Interactive Device Design”
- EECS (Hartmann) + ME (Wright)
- Fabrication (3D printing), embedded electronics, wireless communication, machine learning for detecting actions
Swarm UI Design Tools
Audiences for Swarm Design Tools

User Interfaces are Frequently Created by **Interaction Designers** (not EECS PhDs)

Resurgence of DIY Movement: **Makers**

To make swarm applications ubiquitous, enable these groups to experiment!
Three Challenges

• Rapid Prototyping
• Dealing with Multiple Input Streams
• Real-time Responsiveness
Session Overview

• Valkyrie Savage (CS): Midas
  Fabricating Capacitive Touch Sensors

• Kenrick Kin (CS): Proton
  Multitouch Gestures as Regular Expressions

• John MacCallum (CNMAT): Stream Processing in OSC
Midas

Fabricating Custom Capacitive Touch Sensors for Prototyping Interactive Objects

Valkyrie Savage, Xiaohan Zhang, Björn Hartmann
University of California, Berkeley
Berkeley Institute of Design
2012
Prototyping the Swarm
How do we design these?
d.tools

[UIST06]
Now what?
Design
Touch Areas
SVG Layout File generated by MIDAS
Auto-generated instructions

6. Cut out the file

7. Remove the unnecessary background pieces from the sensors and tails, leaving the sensors undisturbed if possible

8. Using the transfer tape, put the newly-cut sensors onto your object

9. Attach one rainbow wire to each copper tail that leads to a button. Note that you should begin with the brown wire attaching to the top tail.

10. Load a vinyl sheet into the cutter

11. Download this file

12. Open your downloaded file in SignCutPro

13. Use the mirror tool at the bottom to flip the image

14. Cut out the file

15. Remove the positive parts of the cutout, i.e. the shapes

16. Transfer the mask onto your object so that your sensors show through the holes

17. In the Midas interface, program some interactions!
Record and Replay
1. Raw touch events over USB

2. Open WebSocket on startup

3. Processed Touch Events over Web Socket

4. Processed Touch Events delivered to JavaScript callbacks
Button  Discrete Slider  Continuous Slider  2D Pad (2-Layer)
$200
A continuous slider layout fabricated on the LPKF mill
Dimatix Material Printer, with conductive silver ink

Lexmark Inkjet, with Methode 9101 conductive ink
Making it relevant
QT1106 Capacitive Touch Controller

Atmel Microcontroller (USB connection faces away)

US Quarter
The legs of the powerful Tyrannosaurus Rex allowed it to run 40-70 km/h.
Proton: Multitouch Gestures as Regular Expressions

Kenrick Kin$^1,2$  Björn Hartmann$^1$  Tony DeRose$^2$  Maneesh Agrawala$^1$

$^1$University of California, Berkeley  $^2$Pixar Animation Studios
Event-Handling

mouseDown()  

mousemove()  

mouseUp()
Event-Handling

mouseDown() → mouseMove() → mouseUp()

touchesDown() → touchesMove() → touchesUp()

touchesDown() → touchesMove() → touchesUp()

touchesDown() → touchesMove() → touchesUp()
Touch Event Sequences

- number of touches
- the sequence of touch downs
- what each touch hits
- the sequence of touch ups
Kinect
Current Gesture Implementation

```cpp
_state = GesturePossible;

touchesDown(Array *touches, Event *event)
    if(event->allTouches()->count() == 1)
        if(touches[0]->target() != 'n')
            _state = GestureFailed;
    else if(event->allTouches()->count() == 2)
        if(touches[0]->target() != 'm')
            _state = GestureFailed;
    else
        _state = GestureFailed;

touchesMove(Array *touches, Event *event)
    for(i = 0; i < touches->count(); i++)
        if(touches[i]->touchId() == 0 && touches[i]->target() != 'n')
            _state = GestureFailed;
        else if(touches[i]->touchId() == 1 && touches[i]->target() != 'm')
            _state = GestureFailed;

touchesUp(Array *touches, Event *event)
    if(touches[0]->touchId() == 0)
        if(event->allTouches()->count() == 1 && touches[0]->target() == 'n')
            connectNodes();
        else
            _state = GestureFailed;
    else
        if(event->allTouches()->count() == 1 || touches[0]->target() != 'm')
            _state = GestureFailed;
```
Managing Large Gesture Sets

[Wobbrock '09]
Managing Large Gesture Sets

Conflict Detection and Resolution

[Wobbrock '09]
Proton
\[
gesture = D_1^n M_{i}^n * D_2^m (M_{i}^n \| M_2^m) * U_2^m M_{i}^n * U_1^n
\]

\[
gesture.\text{finalTrigger}(\text{connectNodes}());
\]

\[
gestureMatcher.add(\text{gesture});
\]
Declarative Specification

gesture = $D^n_1 M^n_1 * D^m_2 (M^n_1 | M^m_2) * U^m_2 M^n_1 * U^n_1$

gesture.finalTrigger(connectNodes());
gestureMatcher.add(gesture);

Declarative Specification
gesture = $D^n_1 M^n_1 * D^m_2 (M^n_1 | M^m_2) * U^m_2 M^n_1 * U^n_1$

gesture.finalTrigger(connectNodes());
gestureMatcher.add(gesture);

Declarative Specification

Recognition Code Generation
gesture = $D^n_1 M^n_1 * D^m_2 (M^n_1 | M^m_2) * U^m_2 M^n_1 * U^n_1$

gesture.finalTrigger(connectNodes());
gestureMatcher.add(gesture);

Declarative Specification

Recognition Code Generation

Conflict Detection
\[ D_i^c M_i^c \star (D_2^a (M_i^c | M_2^a) \star U_2^a M_i^c \star D_2^a (M_i^c | M_2^a) \star D_3^a (M_i^c | M_2^a | M_3^a) \star \ldots \]

panCamera()

zoomCamera()

\[(U_3^a (M_i^c | M_2^a) \star U_2^a M_i^c \star U_2^a (M_i^c | M_3^a) \star U_3^a M_i^c \star) \star U_i^c\]
Camera()

pan Camera()

zoomCamera()
Proton Touch Event
Touch Event Symbol

\[ \mathbf{E} \quad \text{ObjectType} \]

\[ \text{TouchID} \]

\[ E \in \{D, M, U\} \]
Touch Event Symbol

E

ObjectType

TouchID
Touch Event Symbol

D

ObjectType

TouchID
Touch Event Symbol

ObjectType

TouchID
Touch Event Symbol

E  ObjectType

E  TouchID
Touch Event Symbol

E

_ObjectType_
Touch Event Symbol
Touch Event Symbol

E

ObjectType

3
Touch Event Symbol

E ObjectType

TouchID
Touch Event Symbol

\[ E_s = \text{star} \]

TouchID
Touch Event Symbol

\[ E_b = \text{background} \]

TouchID
Touch Event Symbol

$D^s_1$

Touch Down with 1st Touch on Star
Touch Event Symbol

\[ M^b_2 \]

Touch Move with 2nd Touch on Background
Translation Gesture

\[ D^s_i M^s_i \ast U^s_i \]

translate()
Translation Gesture

\[
D^s_i M^s_i \ast U^s_i
\]

\text{translate()}

![Diagram showing a translation gesture with a star and a finger touching it.](image)
Translation Gesture

\[
\begin{align*}
D^s_l & M^s_l \ast U^s_l \\
\text{translate()} \\
\end{align*}
\]
Translation Gesture

\[ D_i^s \text{ } M_i^s \star U_i^s \]

\text{translate()}

Diagram: A hand gestures to a star symbolized with a star shape, labeled "s" and "b."
Translation Gesture

$D_i^s M_i^s U_i^s$

translate()
Rotation Gesture

\[ D_s^1 M^s_1 \ast D^b_2 (M^s_1 | M^b_2) \ast (U^s_1 M^b_2 \ast U^b_2 | U^b_2 M^s_1 \ast U^s_1) \]
Rotation Gesture

\[ D_1^s M_1^s D_2^b (M_1^s M_2^b) \ast (U_1^s M_2^b U_2^b \mid U_2^b M_1^s U_1^s) \]

rotate()
Rotation Gesture

\[ D_1^s M_1^s D_2^b (M_1^s | M_2^b)^* (U_1^s M_2^b U_2^b | U_2^b M_1^s U_1^s) \]
Rotation Gesture

\[ D_1^s M_1^s D_2^b (M_1^s \mid M_2^b)^* (U_1^s M_2^b * U_2^b \mid U_2^b M_1^s * U_1^s) \]

rotate()
Rotation Gesture

\[ D_1^s M_1^s D_2^b (M_1^s \mid M_2^b) \ast (U_1^s M_2^b \ast U_2^b \mid U_2^b M_1^s \ast U_1^s) \]

rotate()
Stream Generation & Gesture Matching
Stream Generator
Stream Generator

\[ D_i^S \]

Touch Down with 1st Touch on Star
Stream Generator

Touch Move with 1st Touch on Star
Stream Generator

Touch Up with 1st Touch on Star
Gesture Matcher

Translation: $D_1^s M_1^s * U_1^s$

$\text{translate()}$

Rotation: $D_1^s M_1^s * D_2^b (M_1^s || M_2^b)^* ((U_1^s M_2^b U_2^b) | (U_2^b M_1^s U_1^s))$

$\text{rotate()}$
Gesture Matcher

Translation

$$D_1^s M_1^s U_1^s$$

$$\text{translate}()$$

Rotation

$$D_1^s M_1^s D_2^b (M_1^s | M_2^b)^* ((U_1^s M_2^b U_2^b) | (U_2^b M_1^s U_1^s))$$

$$\text{rotate}()$$
Gesture Matcher

Translation: \( D_1^s M_1^s U_1^s \)

Rotation: \( D_1^s M_1^s D_2^b (M_1^s | M_2^b)^* (U_1^s M_2^b U_2^b | U_2^b M_1^s U_1^s) \)
Gesture Matcher

Translation

\[ D_1^S M_1^S U_1^S \]

\[ \text{translate()} \]

Rotation

\[ D_1^S M_1^S D_2^b (M_1^s | M_2^b)^* \left( (U_1^s M_2^b U_2^b) | (U_2^b M_1^s U_1^s) \right) \]

\[ \text{rotate()} \]
Gesture Matcher

\[ D_s^1 M_s^1 M_s^1 D_b^2 \]

Translation: \[ D_s^1 M_s^1 U_s^1 \]
\[ \text{translate()} \]

Rotation: \[ D_s^1 M_s^1 D_b^2 (M_s^1 | M_s^2)^* ((U_s^1 M_s^b U_s^b) | (U_s^b M_s^1 U_s^s)) \]
\[ \text{rotate()} \]
Gesture Matcher

Translation

\[ D^s_1 M^s_1 M^s_1 D^b_2 \]

\[ \text{translate()} \]

Rotation

\[ D^s_1 M^s_1 D^b_2 (M^s_1 | M^b_2)^* ((U^s_1 M^b_2 U^b_2) | (U^b_2 M^s_1 U^s_1)) \]

\[ \text{rotate()} \]
Gesture Matcher

Translation

Rotation
Gesture Matcher

\[
D_1^s M_1^s M_1^s D_2^b M_1^b M_2^b M_1^b M_2^b M_1^b M_2^b U_1^s M_2^b U_2^b
\]

Translation

\[
D_1^s M_1^s U_1^s \quad \text{translate()}
\]

Rotation

\[
D_1^s M_1^s D_2^b (M_1^s M_2^b)^* ((U_1^s M_2^b U_2^b) | (U_2^b M_1^s U_1^s)) \quad \text{rotate()}
\]
Gesture Matcher

Rotation

\[ D_1^s M_1^s D_2^b (M_1^s | M_2^b)^* ((U_1^s M_2^b U_2^b) | (U_2^b M_1^s U_1^s)) \]
Gesture Tablature
Gesture Tablature
Gesture Tablature

touch down
Gesture Tablature

touch down

touch move
Gesture Tablature

object type → s

- touch down
- touch move
- touch up
Gesture Tablature

\[
\text{S} \quad \text{S}
\]

\[
\begin{array}{c}
\text{S} \\
\text{S}
\end{array}
\]
Gesture Tablature
Gesture Tablature

s
b

s
b
Gesture Tablature
Gesture Tablature

rotate()

s
---
s
b
---
b
Gesture Tablature

\[
D_s^1 M_s \ast D_2^b (M_s^1 \mid M_2^b) \ast (U_s^1 M_2^b \ast U_2^b \mid U_2^b M_s^1 \ast U_s^1)
\]
Gesture Tablature

deleteShape()
Gesture Tablature

d d

(s s)

(deleteShape()) *
Tablature Editor
Applications
2D Shape Manipulation
Sketching
EdgeWrite

[Wobbrock '03]
User Study

Time (s) to Understand Gesture

- Tablature: 20 s
- Expression: 40 s (4.7x slower than tablature)
- iOS: 140 s (4.7x slower than tablature)
Customizing Touch Event Symbols

Area, Orientation, Hand ID, User ID

Direction

“L” Gesture
Multiple Touch Streams

User 1: $D^s_1M^s_1M^s_1D^a_2M^s_1M^a_2M^s_1M^a_2M^s_1\ldots$

User 2: $D^s_1M^s_1M^s_1M^s_1M^s_1M^s_1M^s_1M^s_1M^s_1M^s_1\ldots$
Kinect

Multiuser
Kinect

Joint

Direction
Kinect

S HandLeft
M HandLeft
F HandLeft

NextSlide()
Sensors

E

Attributes

Sensor
Conclusion

- Declarative specification
- Automatic gesture matching
- Conflict resolution
- Graphical notation
Open Sound Control Data
Stream Processing

John MacCallum
john@cnmat.berkeley.edu

Center for New Music and Audio Technologies
Open Sound Control (OSC)

What is it?

“A communication protocol that allows musical instruments, computers and other multimedia devices to share music performance data in realtime over a network…” --Wikipedia
Open Sound Control (OSC)

What is good about it?

**Availability:** widespread adoption by the community

**Lightweight:** easy to understand and efficient

**Optimized for modern technology:** high speed networks and high resolution data formats

**Optimized for musical requirements:** temporal semantics
Open Sound Control (OSC)

Who uses it?

Many multimedia software environments (Max/MSP, Ardour, Digital Performer, ...)

Many hardware controllers and platforms (MakingThings Make Controller Kit, IRCAM Ethersense, numerous iPhone and Android apps, ...)

SwarmLab 5/18/2012
Open Sound Control (OSC)

What does it look like?

Message:

/synth/1/frequency    440.0

URL-style address    Data
Open Sound Control (OSC)

What does it look like?

Bundle (of messages):

```
#bundle <timestamp> [
  /synth/1/frequency  440.0
  /synth/1/gain       1.0
  ...
]
```
Identify and Source

Situate with respect to the performer (note OS/X specific HID numbers, use o.io.gametrak for PC)

Neutralize with respect to vendor numerical encoding (0-4095, 12-bit)

Calibrate to a familiar norm: degrees and meters

0.io.hid "Game-Trak V1.3"

0.route /15 /16 /17 /18 /19 /20

0.pak /left/raw/x 0 /left/raw/y 0 /left/raw/z 0 /right/raw/x 0 /right/raw/y 0 /right/raw/z 0

0.expr /left/x = scale(/left/raw/x, 0., 4095., -1., 1.); /right/x = scale(/right/raw/x, 0., 4095., -1., 1.)

0.expr /left/y = scale(/left/raw/y, 0., 4095., -1., 1.); /right/y = scale(/right/raw/y, 0., 4095., -1., 1.)

0.expr /left/z = 1 - /left/raw/z / 4095. ; /right/z = 1 - /right/raw/z / 4095.

0.expr /degrees/left/x = 30. * /left/x ; /degrees/left/y = 30. * /left/y

0.expr /degrees/right/x = 30. * /right/x ; /degrees/right/y = 30. * /right/y

0.expr /meters/left/z = 2. * /left/z ; /meters/right/z = 2. * /right/z
Identify and Source
o.io.hid "Game-Trak V1.3"
/15 1089
/16 892
/17 4056
/18 963
/19 321
/20 4056

Situate with respect to the performer
(note OS/X specific HID numbers, use o.io.gametrak for PC)
o.route /15 /16 /17 /18 /19 /20
o.pak /left/raw/x 0 /left/raw/y 0 /left/raw/z 0 /right/raw/x 0 /right/raw/y 0 /right/raw/z 0
/
/left/raw/x 1089
/left/raw/y 892
/left/raw/z 4056
/right/raw/x 963
/right/raw/y 321
/right/raw/z 4056

Neutralize with respect to vendor numerical encoding
(0-4095, 12-bit)
o.expr /left/x = scale(/left/raw/x, 0., 4095., -1., 1.);
/right/x = scale(/right/raw/x, 0., 4095., -1., 1.)
o.expr /left/y = scale(/left/raw/y, 0., 4095., -1., 1.);
/right/y = scale(/right/raw/y, 0., 4095., -1., 1.)
o.expr /left/z = 1 - /left/raw/z / 4095.;
/right/z = 1 - /right/raw/z / 4095.

Calibrate to a familiar norm: degrees and meters
o.expr /degrees/left/x = 30. * /left/x; /degrees/left/y = 30. * /left/y
o.expr /degrees/right/x = 30. * /right/x; /degrees/right/y = 30. * /right/y
o.expr /meters/left/z = 2. * /left/z; /meters/right/z = 2. * /right/z
Identify and Source

\text{O.IO.HID "Game-Trak V1.3"}

\text{\begin{verbatim}
/15 1089
/16 892
/17 4056
/18 963
/19 321
/20 4056
\end{verbatim}}

Situate with respect to the performer (note OS/X specific HID numbers, use \text{O.IO.Gametrak} for PC)

\text{\begin{verbatim}
o.route /15 /16 /17 /18 /19 /20
o.pak /left/raw/x 0 /left/raw/y 0 /left/raw/z 0 /right/raw/x 0 /right/raw/y 0 /right/raw/z 0
\end{verbatim}}

Neutralize with respect to vendor numerical encoding (0-4095, 12-bit)

\text{\begin{verbatim}
o.expr /left/x = scale(/left/raw/x, 0., 4095., -1., 1.); /right/x = scale(/right/raw/x, 0., 4095., -1., 1.)
o.expr /left/y = scale(/left/raw/y, 0., 4095., -1., 1.); /right/y = scale(/right/raw/y, 0., 4095., -1., 1.)
o.expr /left/z = 1 - /left/raw/z / 4095.; /right/z = 1 - /right/raw/z / 4095.
\end{verbatim}}

\text{\begin{verbatim}
/15 1089
/16 892
/17 4056
/18 963
/19 321
/20 4056
/o.expr /degrees/left/x = 30.* /left/x ; /degrees/left/y = 30.* /left/y
\end{verbatim}}
Neutralize with respect to vendor numerical encoding (0-4095, 12-bit)

Calibrate to a familiar norm: degrees and meters
p o io hid "Game-Trak V1.3"

/left/raw/x 1089
/left/raw/y 892
/left/raw/z 4056
/right/raw/x 963
/right/raw/y 321
/right/raw/z 4056
/exp/neutralize/x "left/x = scale(left/raw/x, 0., 4095., -1., 1.); right/x = scale(right/raw/x, 0., 4095., -1., 1.)"
/exp/neutralize/y "left/y = scale(left/raw/y, 0., 4095., -1., 1.); right/y = scale(right/raw/y, 0., 4095., -1., 1.)"
/exp/neutralize/z "left/z = 1 - left/raw/z / 4095; right/z = 1 - right/raw/z / 4095"
/exp/calibrate/degrees/x "degrees/left/x = 30. * left/x; degrees/right/x = 30. * right/x"
/exp/calibrate/degrees/y "degrees/left/y = 30. * left/y; degrees/right/y = 30. * right/y"
/exp/calibrate/meters/z "meters/left/z = 2. * left/z; meters/right/z = 2. * right/z"
p.o.io.hid "Game-Trak V1.3"

/expr/neutralize/x "/left/x = scale(/left/raw/x, 0., 4095., -1., 1.); /right/x = scale(/right/raw/x, 0., 4095., -1., 1.)"
/expr/neutralize/y "/left/y = scale(/left/raw/y, 0., 4095., -1., 1.); /right/y = scale(/right/raw/y, 0., 4095., -1., 1.)"
/expr/neutralize/z "/left/z = 1 - /left/raw/z / 4095.; /right/z = 1 - /right/raw/z / 4095."

/expr/calibrate/degrees/x "/degrees/left/x = 30. * /left/x; /degrees/right/x = 30. * /right/x"
/expr/calibrate/degrees/y "/degrees/left/y = 30. * /left/y; /degrees/right/y = 30. * /right/y"
/expr/calibrate/degrees/z "/degrees/left/z = 2. * /left/z ; /degrees/right/z = 2. * /right/z"

o.expr eval(/expr/neutralize/x); eval(/expr/neutralize/y); eval(/expr/neutralize/z)

o.expr eval(/expr/calibrate/degrees/x); eval(/expr/calibrate/degrees/y); eval(/expr/calibrate/degrees/z)

/expr/neutralize/x "/left/x = scale(/left/raw/x, 0., 4095., -1., 1.); /right/x = scale(/right/raw/x, 0., 4095., -1., 1.)"
/expr/neutralize/y "/left/y = scale(/left/raw/y, 0., 4095., -1., 1.); /right/y = scale(/right/raw/y, 0., 4095., -1., 1.)"
/expr/neutralize/z "/left/z = 1 - /left/raw/z / 4095.; /right/z = 1 - /right/raw/z / 4095."

/expr/calibrate/degrees/x "/degrees/left/x = 30. * /left/x; /degrees/right/x = 30. * /right/x"
/expr/calibrate/degrees/y "/degrees/left/y = 30. * /left/y; /degrees/right/y = 30. * /right/y"
/expr/calibrate/degrees/z "/degrees/left/z = 2. * /left/z ; /degrees/right/z = 2. * /right/z"

/expr/meters/x -0.468132
/expr/meters/y -0.52967
/expr/meters/z -0.564347
/expr/meters/x -0.843223
/expr/meters/z 0.00952381
/expr/meters/z 0.00952381
/expr/degrees/360 -15.8901
/expr/degrees/360 -16.9304
/expr/degrees/360 -25.2967
/expr/meters/z 0.0190476
/expr/meters/z 0.0190476
Open Sound Control (OSC)

http://cnmat.berkeley.edu

http://opensoundcontrol.org