Experimental Interactions 008 - Aurélien Tabard

How touching?
touch, multi-touch, gestures and a *bit* more

Some material and inspiration from N. Roussel's keynote:

Multitouch & interaction gestuelle : vraies et fausses bonnes idées - interaction.lille.inria.fr/~roussel/

Plan

Pippin?

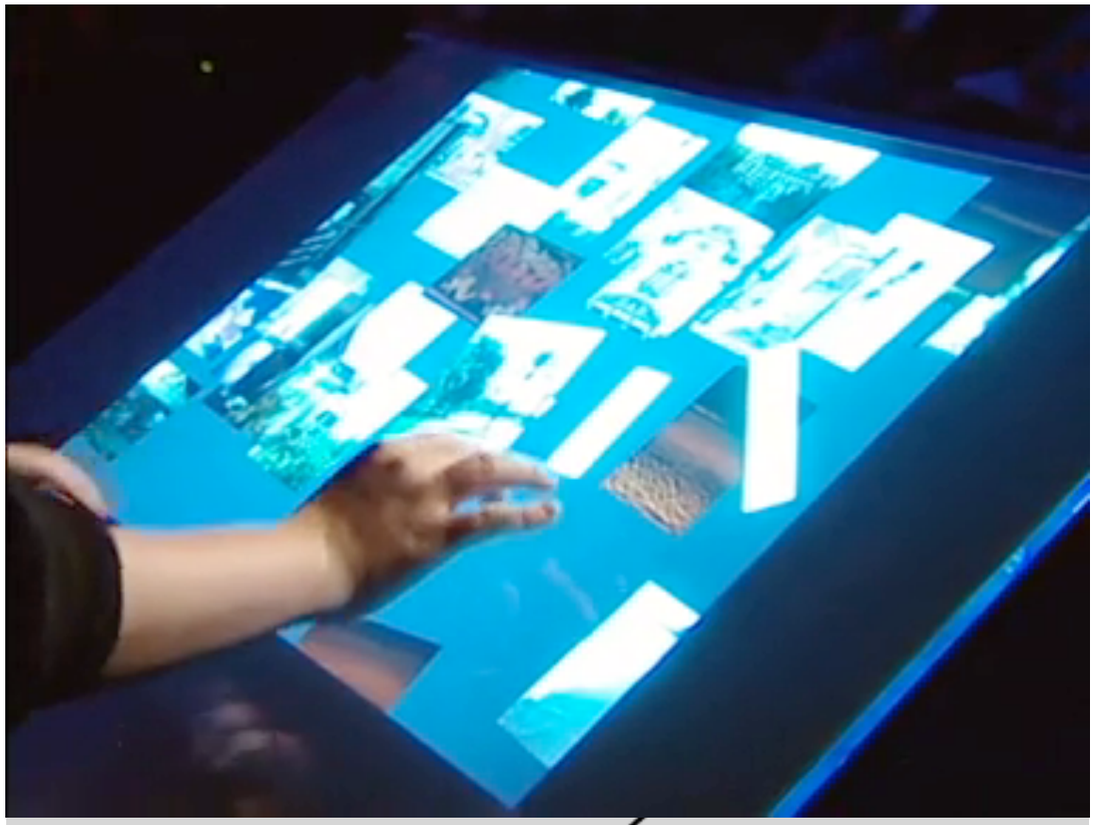
I talk.

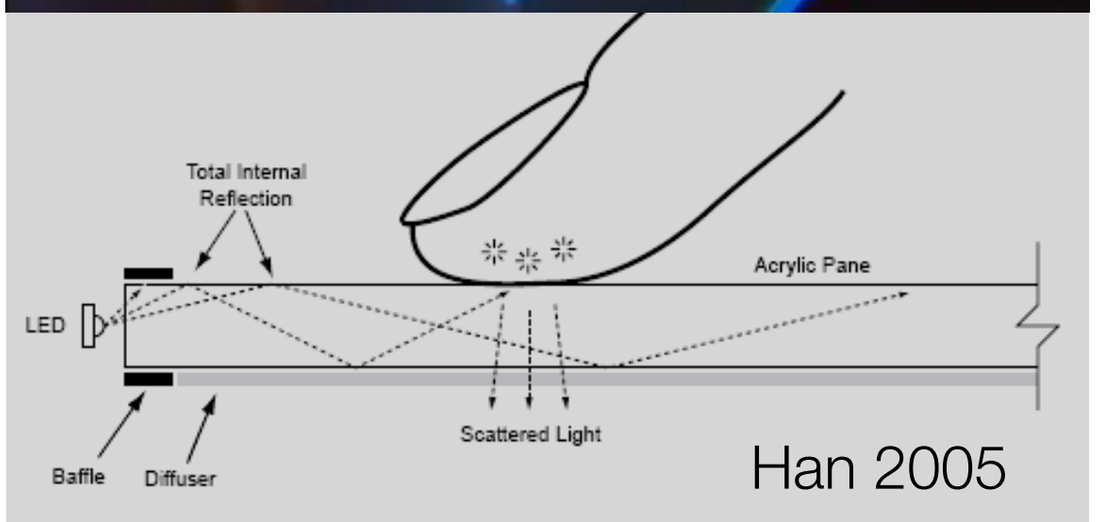
Pippin?

A tour of ITU’s multi-touch devices.

Your first memory of (multi)-touch?

Multi-touch





Han 2005

January 9, 2007

“The future is already here. It is just not uniformly distributed”

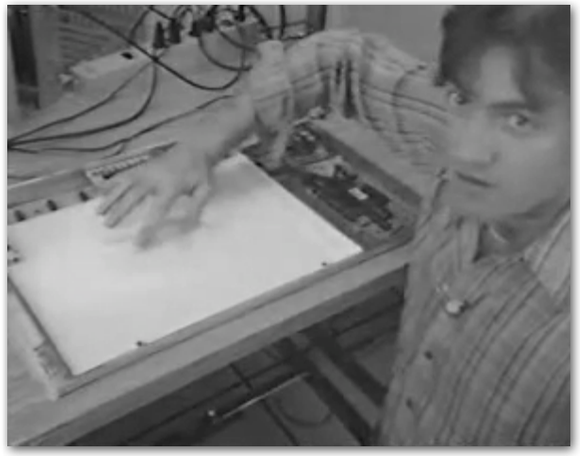
—William Gibson

The long nose of innovation

1972 : PLATO IV

1979 : Put that there

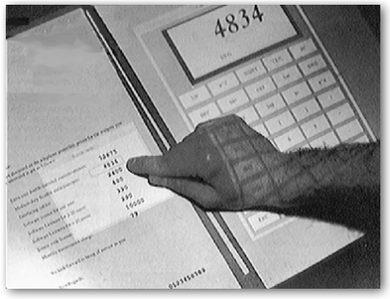
1983 : Videoplace



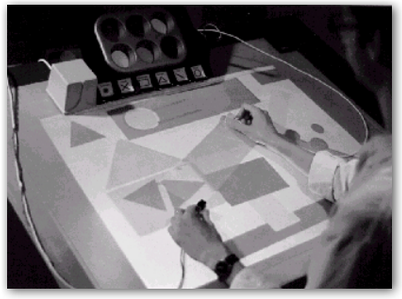
1985 : Multitouch tablet

(...) "new" technologies - like multi-touch - do not grow out of a vacuum. While marketing tends to like the "great invention" story, real innovation rarely works that way.

In short, the evolution of multi-touch is a text-book example of what I call "the long-nose of innovation".

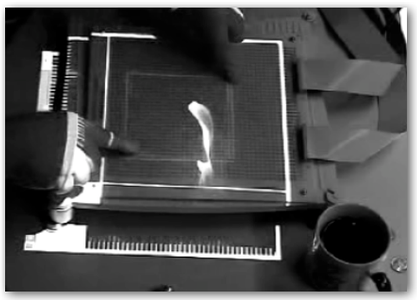


1991: Digital Desk



1991: Bricks

1999 : Augmented surfaces



2001 : DiamondTouch

2004 : DiamondSpin

2006 : DigiTable

Slide from N. Roussel

Why now?

1994 : Disclosure

1995 : Johnny Mnemonic

2002 : Minority report

2005 : The island

2008 : Quantum of solace

2008 : Iron man

Is touch limited to these devices?

Beyond mouse and keyboard

- Mobile devices
- Surface computing
- Augmented reality
- Ambient technologies
- Tangibles and Wearables

Interactive surfaces

NYU - Perceptive pixel (FTIR)

Shared display

.: Collaboration

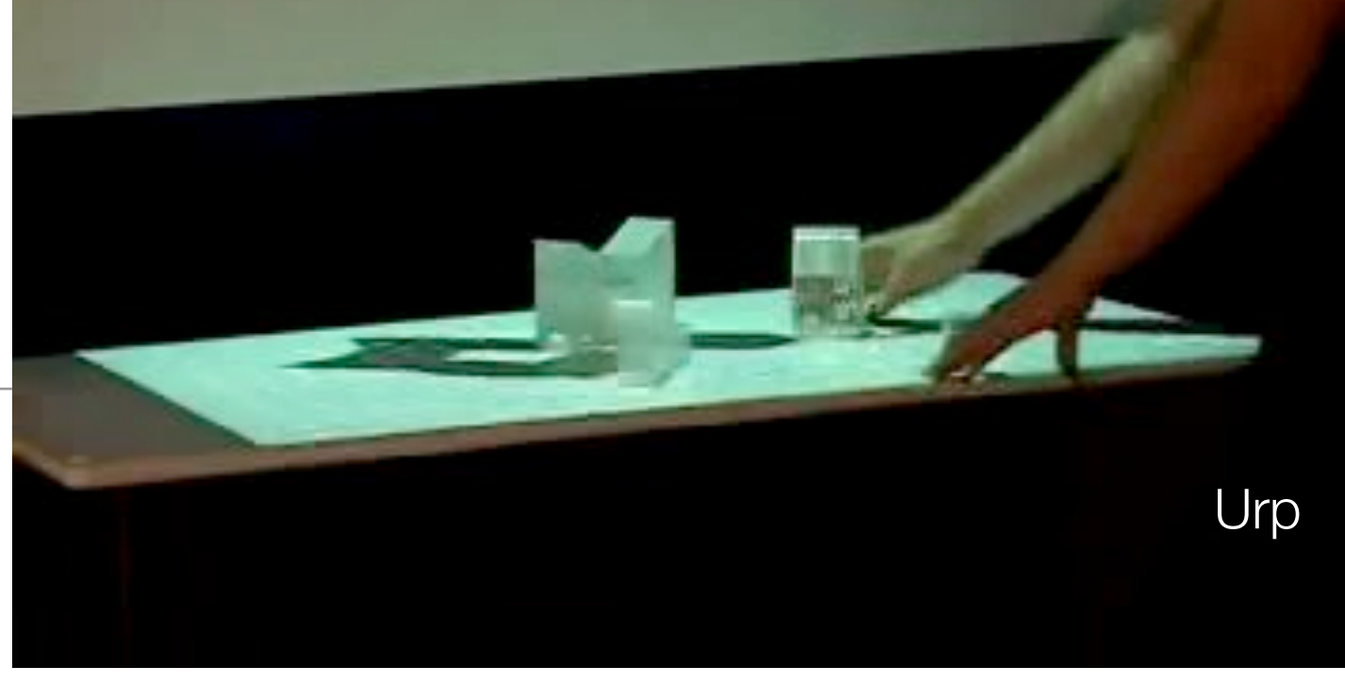
.: Space control

.: Input management

Focus (+ context?)

.: Lack of overview

Augmented reality



Urp

Mixing the digital and the physical

.: capture

.: consistency

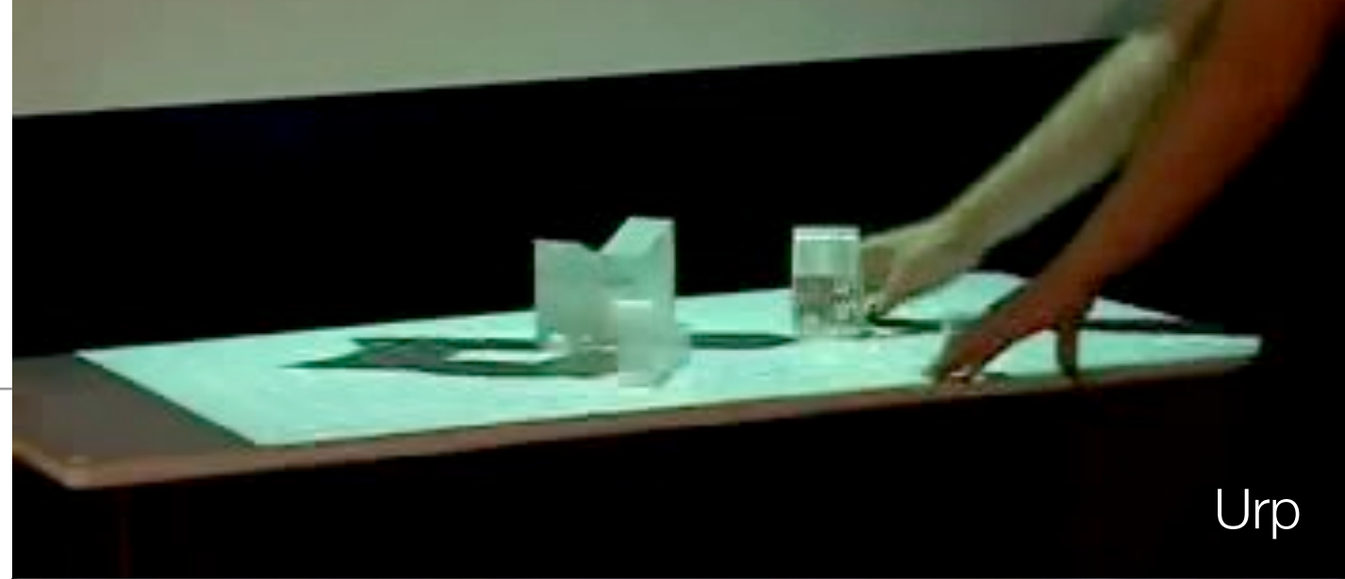
Wellner’s Digital Desk

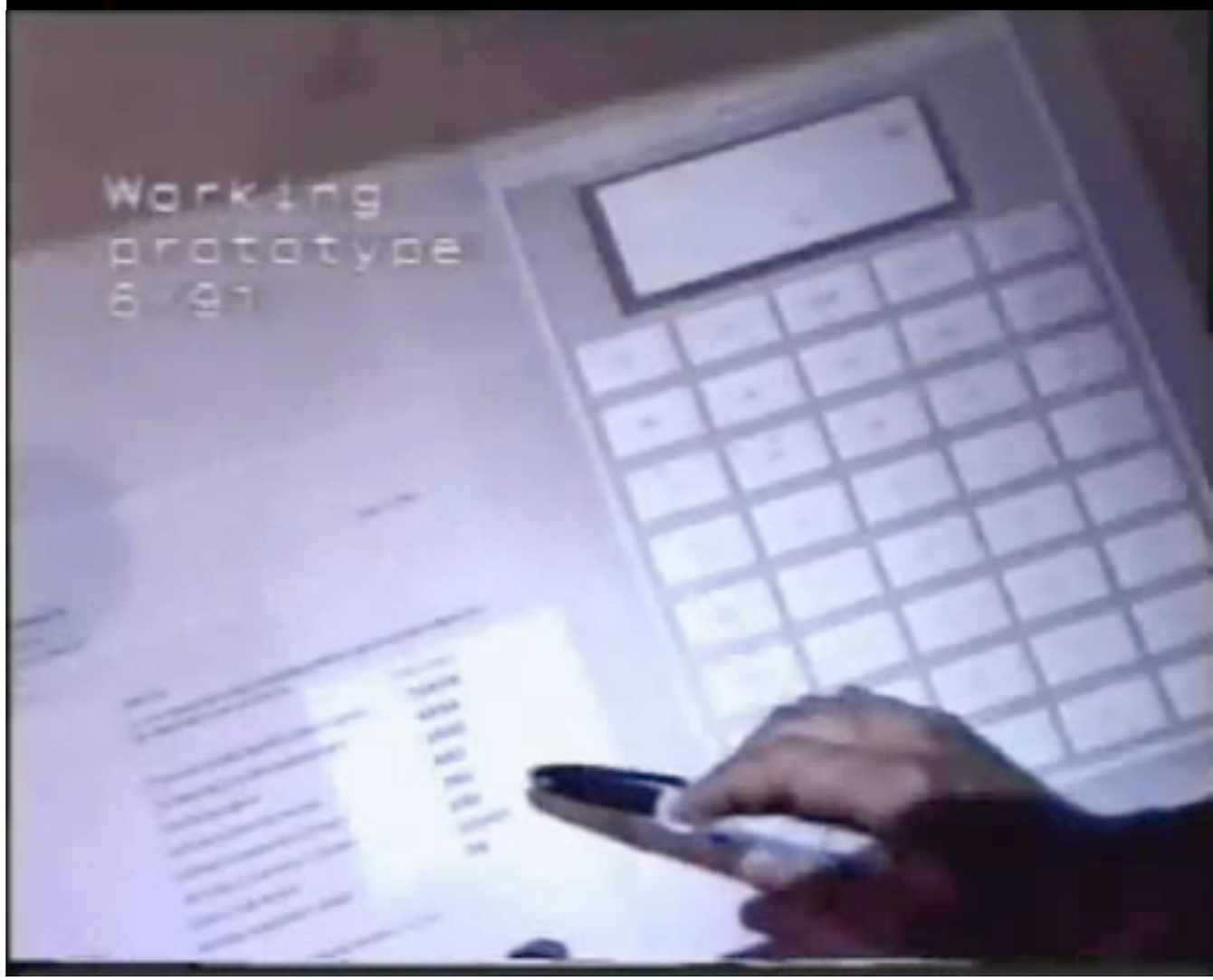
Augmented reality

Mixing the digital and the physical

.: capture

.: consistency





Wellner’s Digital Desk

Tangible interfaces

Ishii’s bottles

Radically new but always familiar

.: Everyday (no actuation)

.: Actuated

.: Ambient

Tangible interfaces

Ishii’s bottles

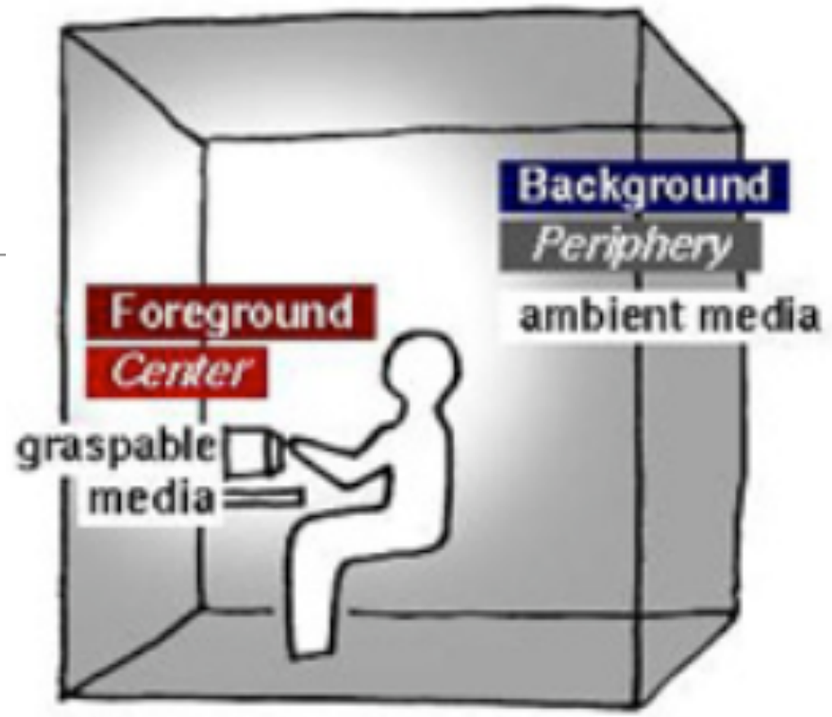
Radically new but always familiar

.: Everyday (no actuation)

.: Actuated

.: Ambient

Ambient technologies



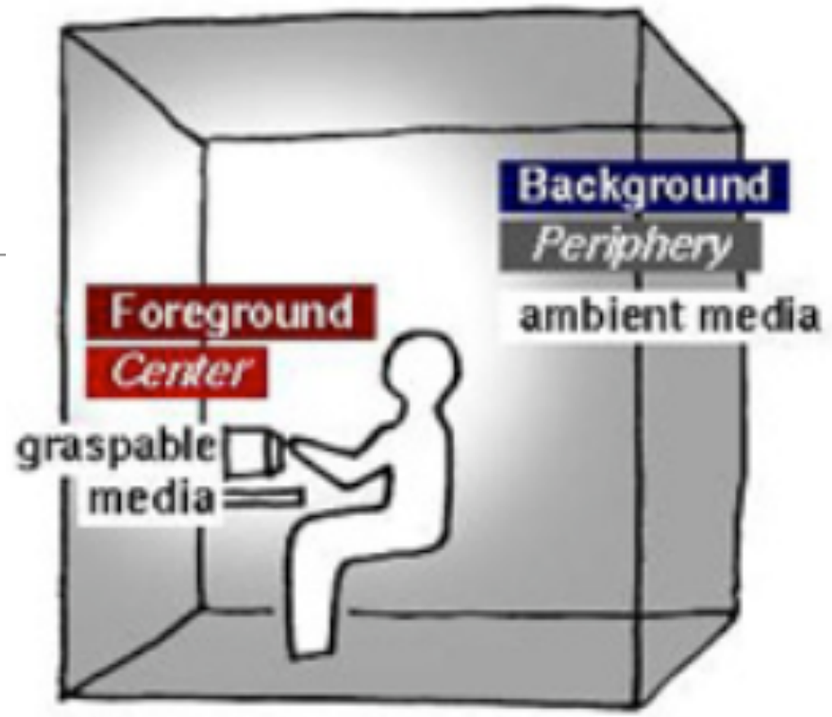
drawing: Hiroshi Ishii

Moving between

.:background and,

.:foreground of attention

Ambient technologies



drawing: Hiroshi Ishii

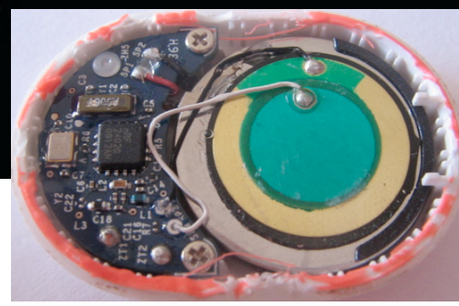
Moving between

.:background and,

.:foreground of attention

Wearables


CAMERA
COLOR MARKERS
PROJECTOR
MIRROR



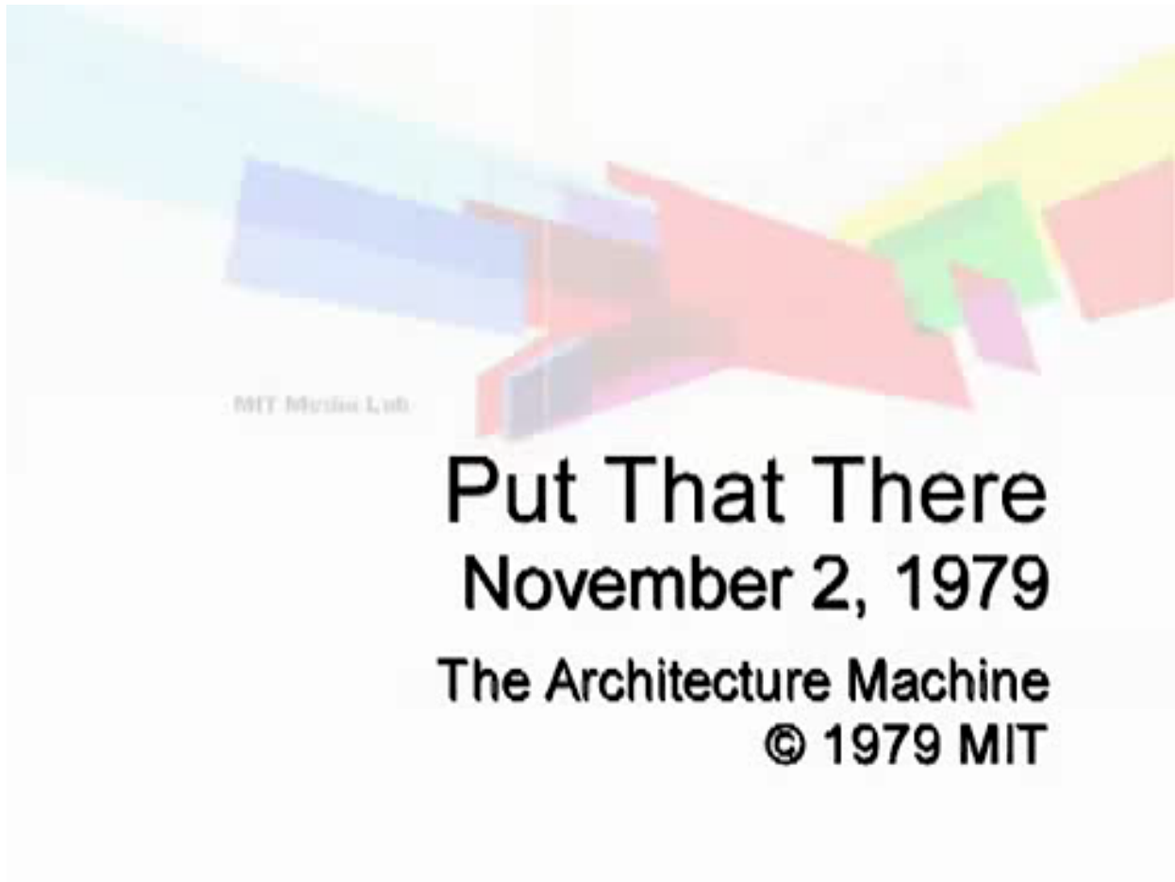

GARMIN
10:52 06
SAT 25
GARMIN
GARMIN

Multimodality

Definitions:

.: Multimodal generally refers to an interface that can accept input from two or more combined modes.

.: Multimedia generally refers to an interface that produces output in two or more modes.



Modalities

Input:

.: mouse

.: pen

.: speech

.: audio (non-speech)

.: tangible object manipulation

.: gaze, posture, body-tracking

Output

.: Visual displays

.: Haptics: Force Feedback

.: Audio

.: Smell

.: Taste

Motivations

.:Hands busy / eyes busy

.:Mutual disambiguation

.:Faster input

.:More “natural”

Anthropomorphism?

Input-Output loop

.:Modes: visual, auditory, haptic...

.:Scale: wearable, mobile, office, city, ambient...

Input-Output loop

.:Modes: visual, auditory, haptic;

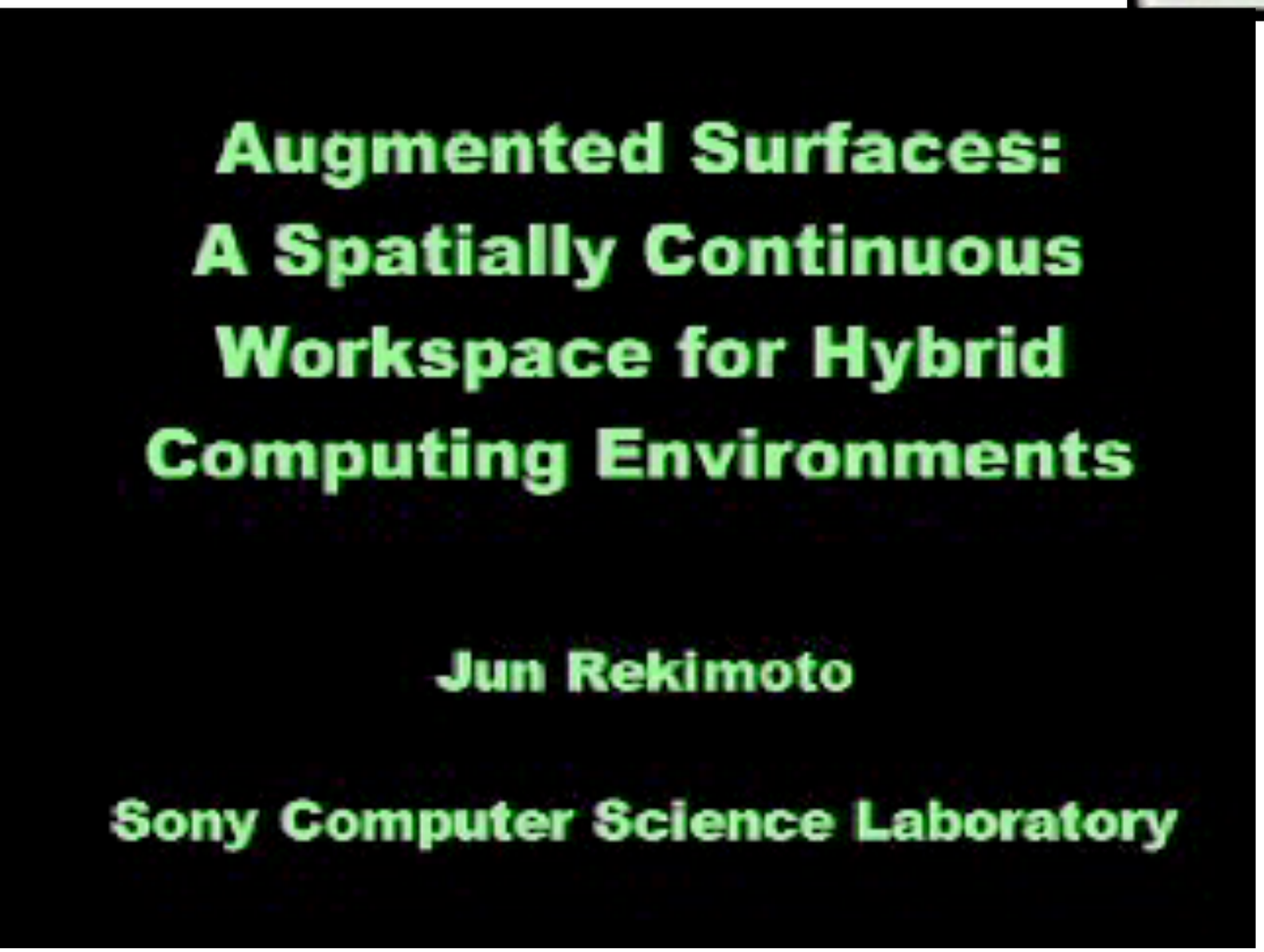
.:Scale: wearable, mobile, office, city, ambient;

How to move from one device to the next?

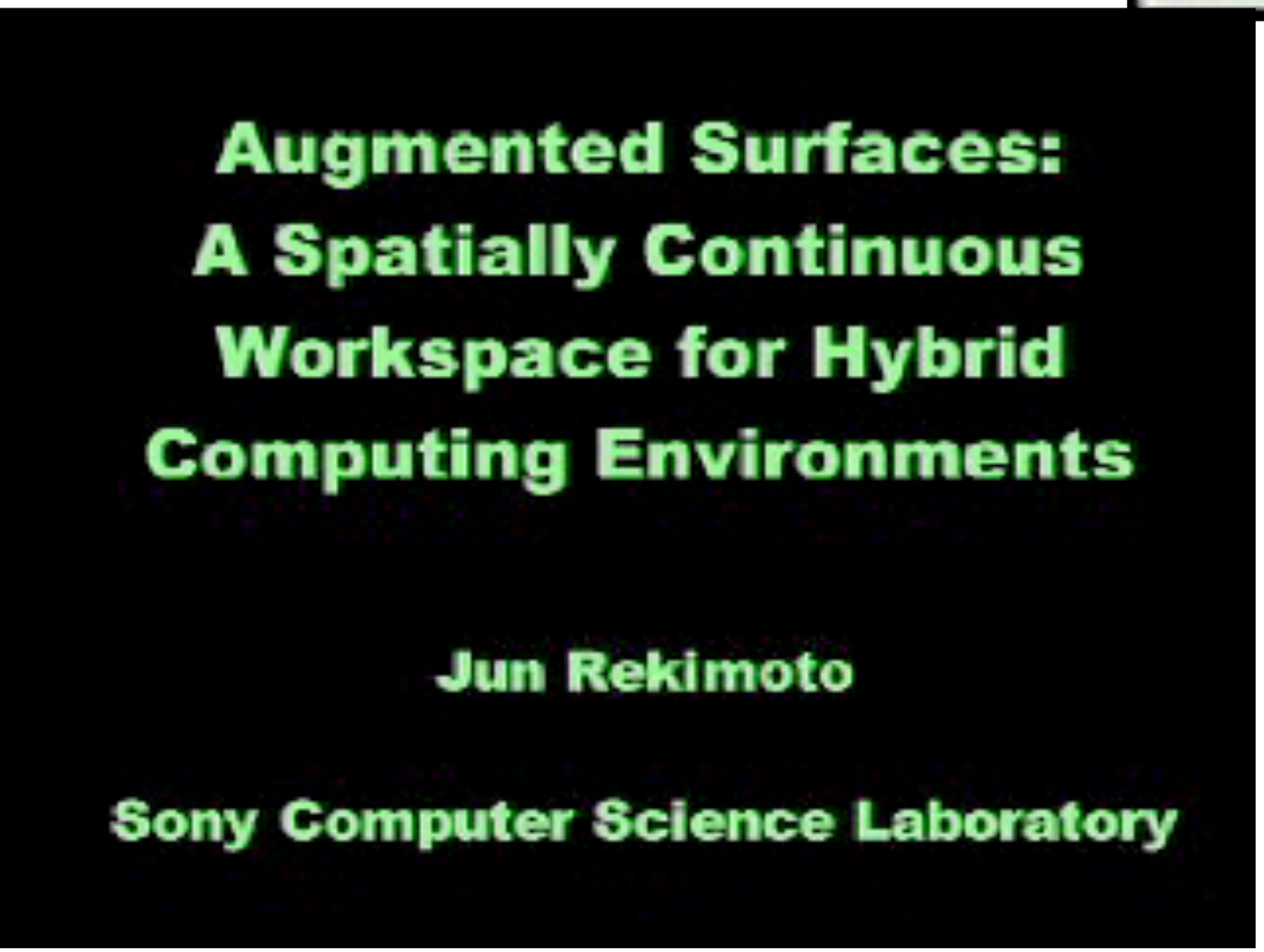
ex: Augmented surfaces
Recombinant computing
Plasticity

Integration

Integration



Integration



Beyond mouse and keyboard

Mobile devices

Augmented reality

Surface computing

Ambient technologies

Tangibles and Wearables

Beyond mouse and keyboard

Mobile devices

Augmented reality

Surface computing

Ambient technologies

Tangibles and Wearables

Natural interaction?*

Natural User Interfaces?

Dennis Wixon | UX Week 2008

John Underkoffler

http://oblong.com/article/085zBpRSY9JeLv2z.html

You adapt the gestural language from the Luminous Room work. ***You train the actors to use this language.*** *They become adept, even though it is partly an exercise in mime.* ***The production will shoot the actors performing gestural tasks in front of an enormous transparent screen, but the screen will be blank, a prop****. Graphics will be composited onto the screen in post-production. You understand that for a sense of causality to emerge the actors must be able to clearly visualize the effects of their gestural work. You assemble a training video showing this.*

When the time comes to shoot, the director explains what sequence of analysis should occur in each scene. You translate this into the gestural language. You explain what graphical elements the actors would be seeing on different parts of the screen, how they are manipulating those elements. You make sure that a detailed account of all this is subsequently available to the editor and the visual effects people. Continuity of the original intent is critical. The cameras roll.

The movie appears in 2002. ***The scenes of gestural computation show something apparently real.***

On intuitive and natural

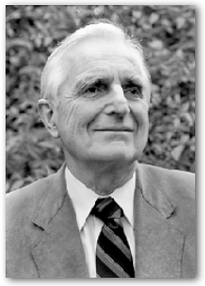
Jef Raskin in *The humane interface* (p. 150)

Many interface requirements specify that the resulting product be intuitive, or natural. However, there is no human faculty of intuition, as the word is ordinarily meant; that is, knowledge acquired without prior exposure to the concept, without having to go through a learning process, and without having to use rational thought*. When an expert uses what we commonly call his intuition to make a judgement, with a speed and accuracy that most people would find beyond them, we find that he has based his judgement on his experience and knowledge. Often, experts have learned to use methods and techniques that nonexperts do not know. Task experts often use cues of which others are not aware or that they do not understand. Expertise, unlike intuition, is real.*

When users say that an interface is intuitive, they mean that it operates just like some other software or method with which they are familiar. *Sometimes, the word is used to mean habitual, as in "The editing tools become increasingly intuitive over time." Or, it can mean already learned, as was said of a new aircraft navigation device: "Like anything, it can be learned, but it would take a lot of experience to do it intuitively" (Collins 1994).*

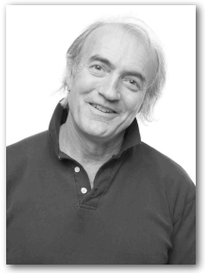
Another word that I try to avoid in discussing interfaces is natural. Like intuitive, it is usually not defined. An interface feature is natural, in common parlance, if it operates in such a way that a human needs no instruction. This typically means that there is some common human activity that is similar to the way the feature works. However, it is difficult to pin down what is meant by similar. *Similarities or analogies can occur in many ways. Certainly, that the cursor moves left when a mouse is pushed to the left and right when the mouse is pushed to the right is natural. Here, the term natural equates to very easily learned. Although it may be impossible to quantify naturalness, it is not too difficult to quantify the learning time.*

On learning



Douglas Engelbart in Thierry Bardini's *Bootstrapping* (p. 28)

When interactive computing in the early 1970s was starting to get popular, and they [researchers from the AI community] start writing proposals to NSF and to DARPA, ***they said*** *well, what we assume is that* ***the computer ought to adapt to the human*** *[…]* ***and not require the human to change or learn anything. And that was just so just soantithetical to me. It's sort of like making everything to look like a clay tablet so you don't have to learn to use paper.***



Bill Buxton on the *power law of practice*

Don't waste people skills. They're really expensive to acquire and we're already too busy. (...) ***One of the key things is whenever possible to not force you to learn something new but to do the design in a way that exploits the skills you already have****. (...)* ***Now there are some places*** *(...)* ***where if the value is there, it's worth learning something new.***

Implications on interaction

Making Sense of Sensing Systems

.: When I address a system, how does it know I am addressing it?

.: When I ask a system to do something how do I know it is attending?

.: When I issue a command (such as save, execute or delete), how does the system know what it relates to?

.: How do I know the system understands my command and is correctly executing my intended action?

.: How do I recover from mistakes?

see Bellotti et al., CHI 2002

How do you cope with break-downs?

How do you cope with break-downs?

Seamless / seamful

From M. Weiser, UIST'94
Building Invisible Interfaces

> Is a seamless building one in which you never notice as you move from place to place?
>
> Making everything the same is easy;
>
> Hard is letting everything be itself, with other things
>
> **Goal: seamful systems, with beautiful seams**

see Chalmers, M. and Maccoll, I. (2003)
Seamful and seamless design in ubiquitous computing

Novel Interactions

Let’s get technical

.: architectures

.: computer vision

.: tangibles

.: gestures

Implications for UI *inputs* handling

Implications for UI *inputs* handling

.:Event-based

Implications for UI *inputs* handling

.:Event-based

.:Focus on widgets (or interactive elements)
Tendency to manage presentation + behavior

Implications for UI *inputs* handling

.:Event-based

.:Focus on widgets (or interactive elements)
Tendency to manage presentation + behavior

.:State machines

Implications for UI *inputs* handling

.:Event-based

.:Focus on widgets (or interactive elements)
Tendency to manage presentation + behavior

.:State machines

.:Focus on interactions
Ability to handle multiple interactions simultaneously.

Implications for UI *inputs* handling

.:Event-based

.:Focus on widgets (or interactive elements)
Tendency to manage presentation + behavior

.:State machines

.:Focus on interactions
Ability to handle multiple interactions simultaneously.

.:Reactive / Flow-based

Implications for UI *inputs* handling

.:Event-based

.:Focus on widgets (or interactive elements)
Tendency to manage presentation + behavior

.:State machines

.:Focus on interactions
Ability to handle multiple interactions simultaneously.

.:Reactive / Flow-based

.:Cascading reactive devices, input-reconfiguration.
Real-time settings

Architecture example I

Event oriented

.: most GUI

.: The Event Heap from the iRoom [Joahnson & Fox, 2002]

.: Paper toolkit [Yeh et al, UIST'08]

Paper Toolkit

Architecture example I

```
1 public class SimplePaperApp {
2   private static int numStrokes = 0;
3   public static void main(String[] args) {
4      Application app = new Application();
5      Sheet sheet = app.createSheet(8.5, 11);
6      Region inkReg = sheet.createRegion(1, 1, 4, 4);
7      Region tapReg = sheet.createRegion(1, 6, 1, 1);
8      Device remote = app.createRemoteDevice();
9      inkReg.addEventHandler(
10        new InkHandler() {
11          public void handleInkStroke(InkEvent e) {
12             numStrokes++;
13         }});
14     tapReg.addEventHandler(
15       new ClickAdapter() {
16         public void clicked(PenEvent e) {
17           remote.displayMessage(numStrokes);
18         }});
19     app.run(); // starts event dispatch loop
20   }
21 }
```

Architecture example II

State machines

.: Instrumental interaction and state machines

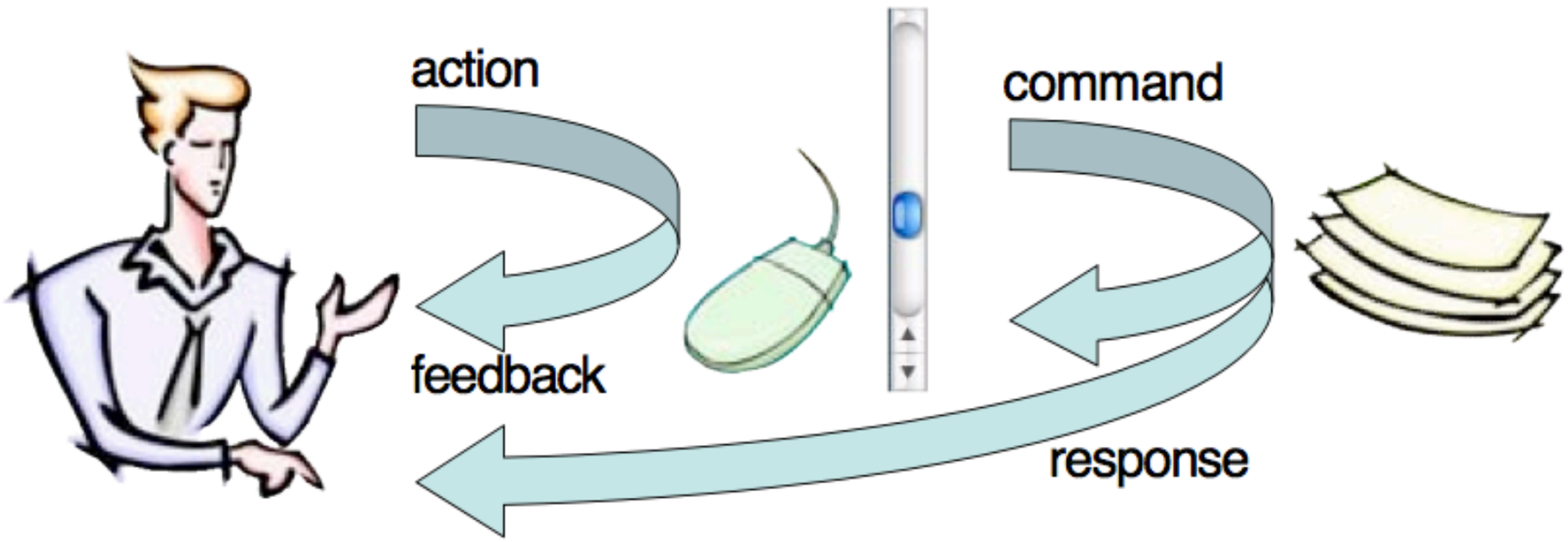
.: State machines in QT 4.6

.: VIGO: Instrumental Interaction in Multi-Surface Environments

Instrumental interaction

Definition:

“An instrument is a mediator or two-way transducer between the user and domain objects. The user acts on the instrument,which transforms the user’s actions into commands affecting relevant target domain objects.”



Instrumental interaction

3 design principles:

.: Reification,

.: Polymorphism,

.: Reuse.

```
segment s;    // global variable
bool drawing; // global variable
void HandleButtonPress (point p) {
   s.p1 = s.p2 = p; drawing = false;
}
void HandleMouseMove (point p) {
   if (! drawing && distance (s.p1, p) > 3) {
      drawing = true; s.p2 = p; Draw (s);
   } else {
      Erase (s); s.p2 = p; Draw (s);
   }
}
void HandleButtonRelease () {
   if (drawing) {
      Erase (s);
      AppNewSegment (s); // notify application
   }
}
```

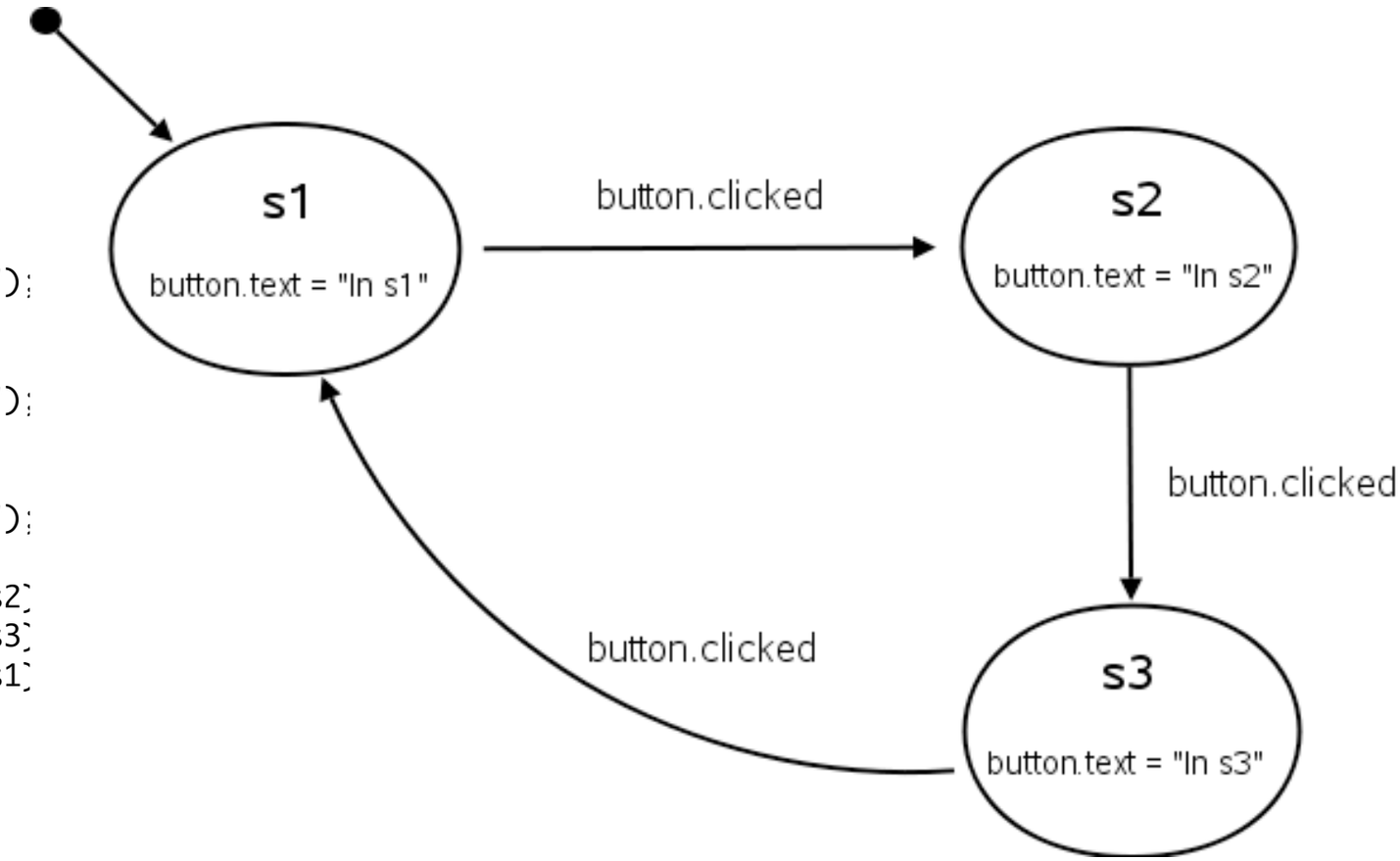
Figure 8. Event-driven programming of rubber-band interaction (event-loop omitted).

```
interaction RubberBand {
   segment s;
   state start {
      when ButtonPress(p) {
         s.p1 = s.p2 = p;
      } -> waitMove
   }
   state waitMove {
      when MouseMove(p)
        && distance(s.p1, p) > 3 {
         s.p2 = p; Draw(s);
      } -> track
      when ButtonRelease -> start
   }
   state track {
      when MouseMove(p) {
         Erase(s); s.p2 = p; Draw(s);
      }
      when ButtonRelease {
         Erase(s); App.NewSegment(s);
      } -> start
   }
}
```

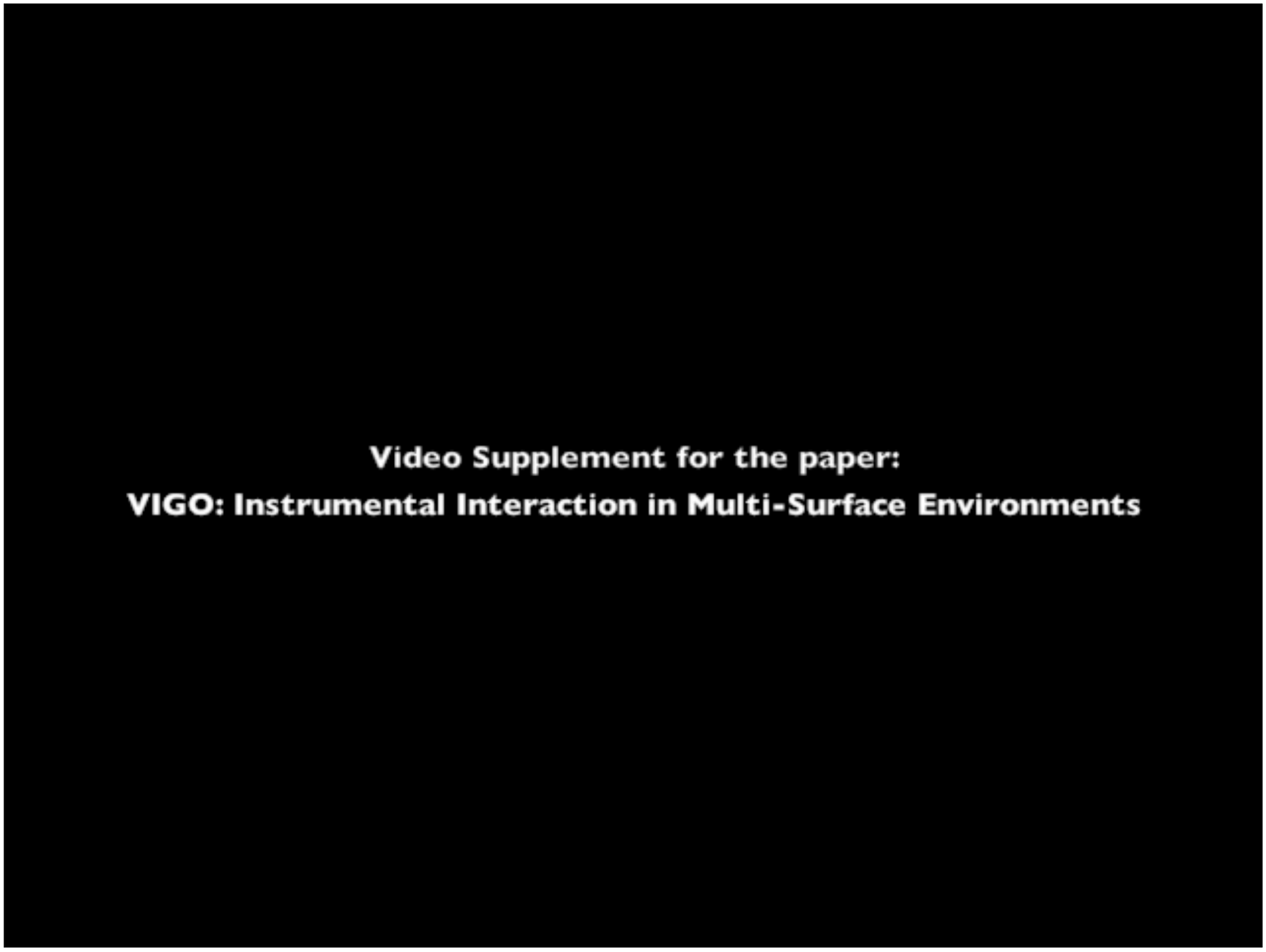
Figure 9. Interaction machine for rubber-band.

State machines in QT

```
int main(int argc, char **argv)
{
    QApplication app(argc, argv);
    QPushButton button;

    QStateMachine machine;

    QState *s1 = new QState(machine.rootState());
    s1->setPropertyOnEntry(&button, "text", "In s1");

    QState *s2 = new QState(machine.rootState());
    s2->setPropertyOnEntry(&button, "text", "In s2");

    QState *s3 = new QState(machine.rootState());
    s3->setPropertyOnEntry(&button, "text", "In s3");

    s1->addTransition(&button, SIGNAL(clicked()), s2);
    s2->addTransition(&button, SIGNAL(clicked()), s3);
    s3->addTransition(&button, SIGNAL(clicked()), s1);

    button.resize(200, 200);
    button.show();

    machine.setInitialState(s1);
    machine.start();

    return app.exec();
}
```



VIGO


Video Supplement for the paper:
VIGO: Instrumental Interaction in Multi-Surface Environments

Architecture example II

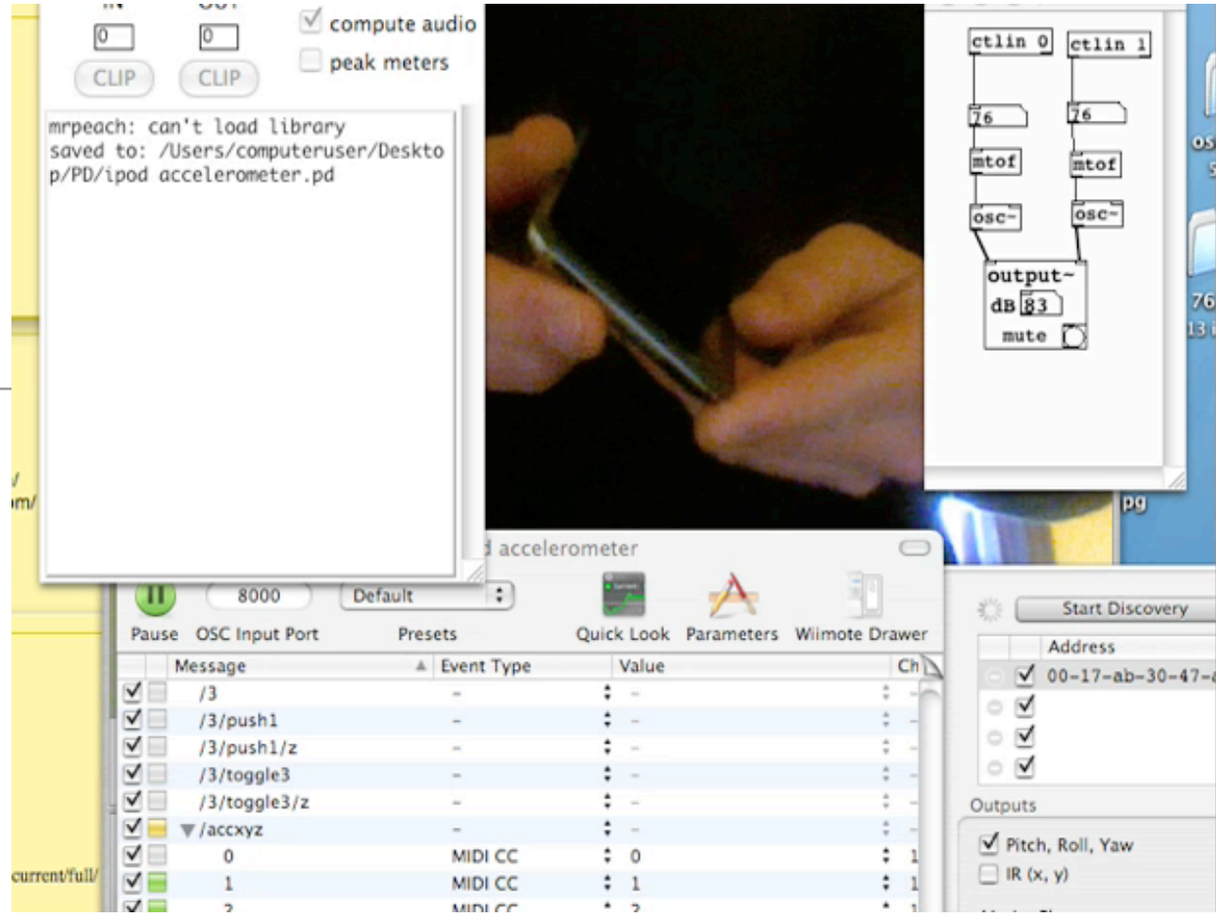
Pick and drop

```
@state # defining state “start”
def start(self):
    # transition on button press to state ”picking”
    @transition(event=ButtonDown, to=self.picking)
    def action(event)
        pass # no action

    @transition(event=Pick, to=self.picking)
    def action(event)
        self.picked = event.picked # record picked object
```

```
@state # defining state “picking”
def picking(self):
    @transition(event=Select, guard=self.picktest,
to=self.dropping)
    def action(event):
        self.picked = objectHandler.getObj(event.element)
        eventHandler.fireEvent(Pick(self.picked)) # fire
Pick event

    @transition(event=Pick, to=self.dropping)
    def action(event):
        self.picked = event.picked

    @transition(event=ButtonUp, to=self.start)
        def action(event):
        pass
```

```
@state # defining state “dropping”
def dropping(self):
    @transition(event=Select, guard=self.picktest,
to=self.picking)
    def action(event):
        if self.picked is not None:
            <insert picked object into selected object>
            eventHandler.fireEvent(Drop(self.picked)) #
fire Drop event

    @transition(event=Drop, to=self.picking)
        def action(event):
            pass

@state # defining state “picked”
def picked(self):
    @transition(event=ButtonDown, to=self.dropping)
    def action(event):
        pass

    @transition(event=Drop, to=self.start)
        pass
```

Architecture example III



Reactive - flow

.: Max Msp / Pure data

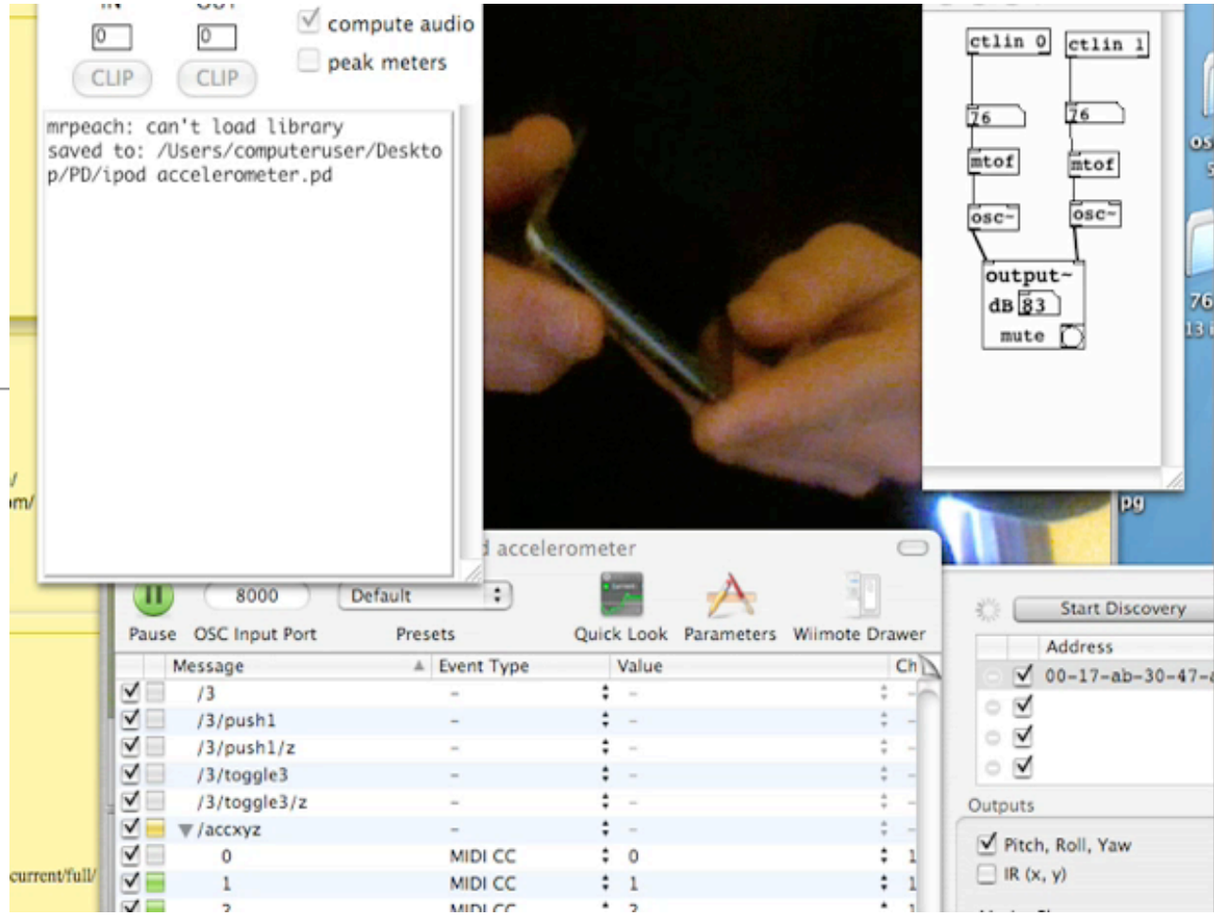
.: ICON

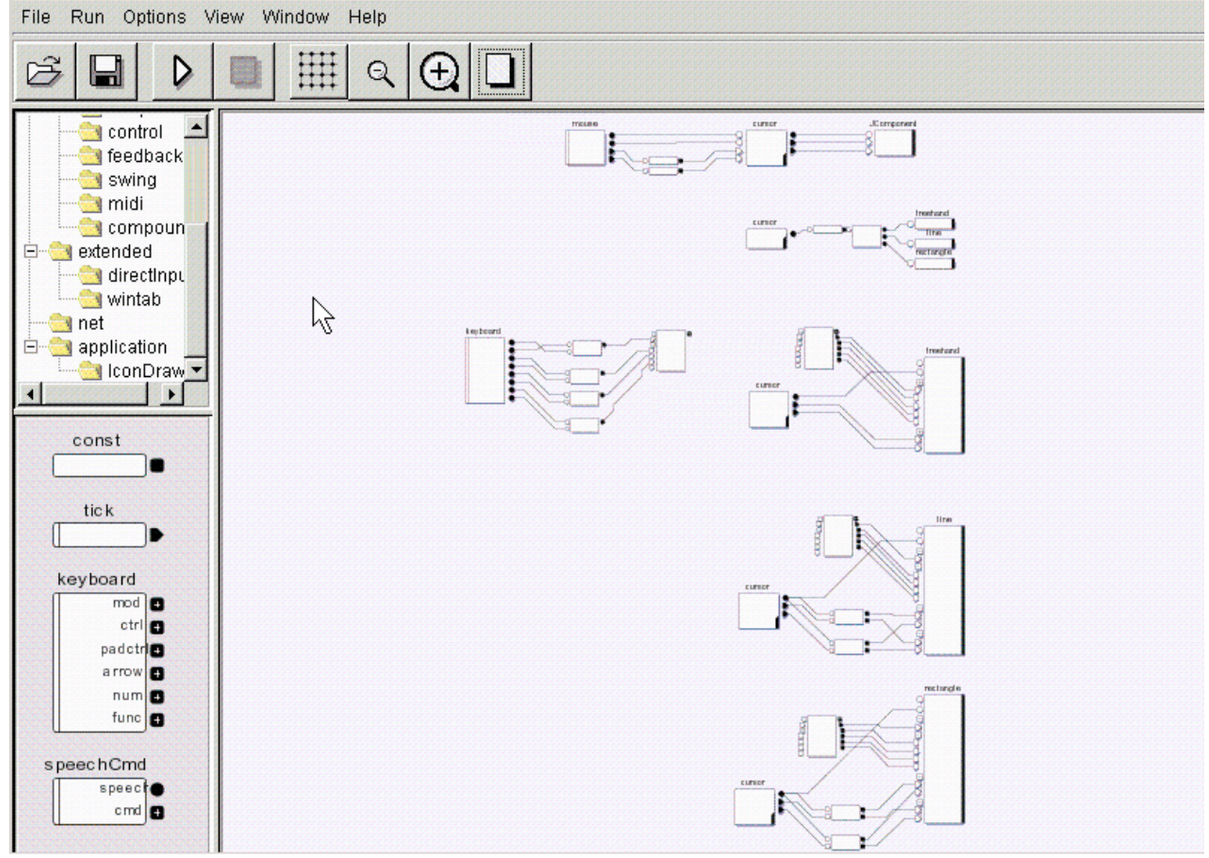
Architecture example III

Reactive - flow

.: Max Msp / Pure data

.: ICON



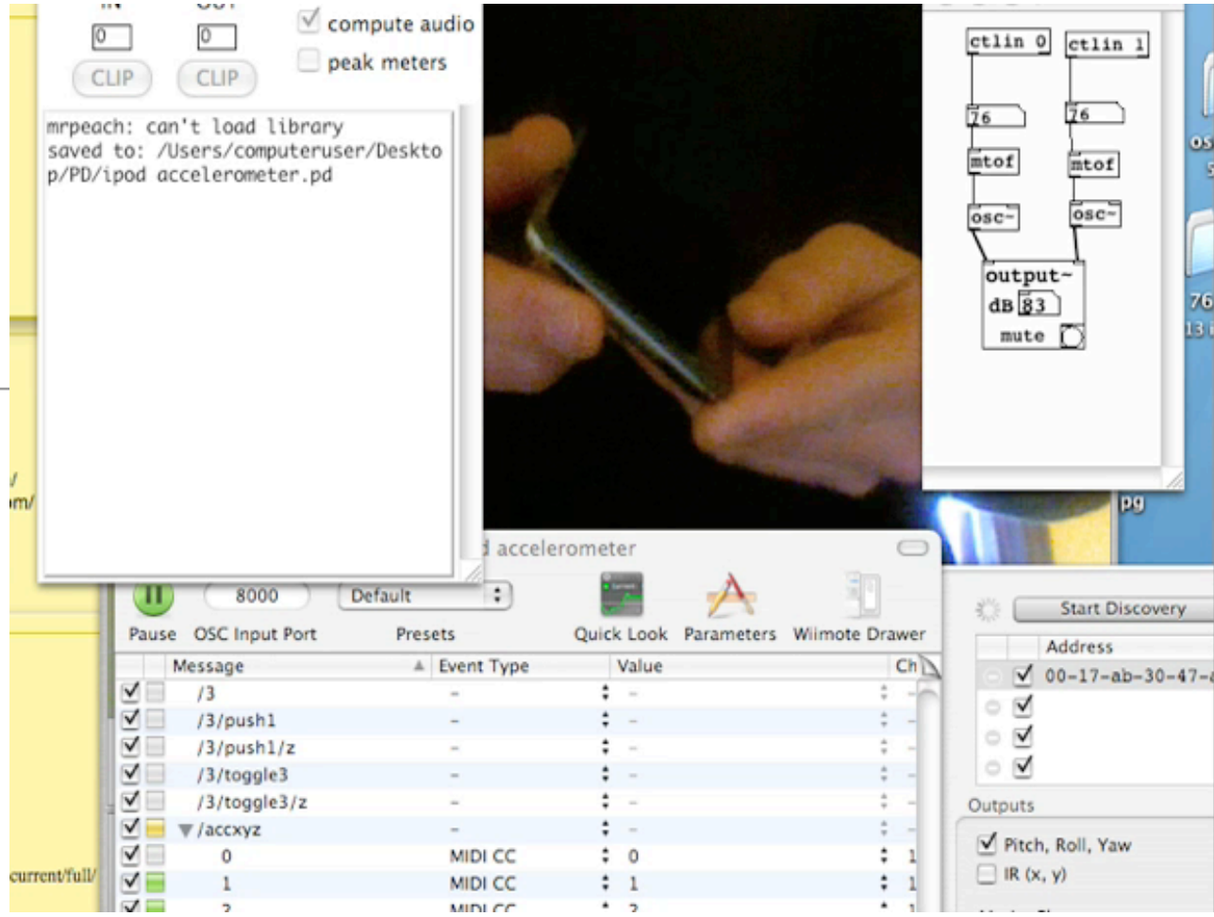


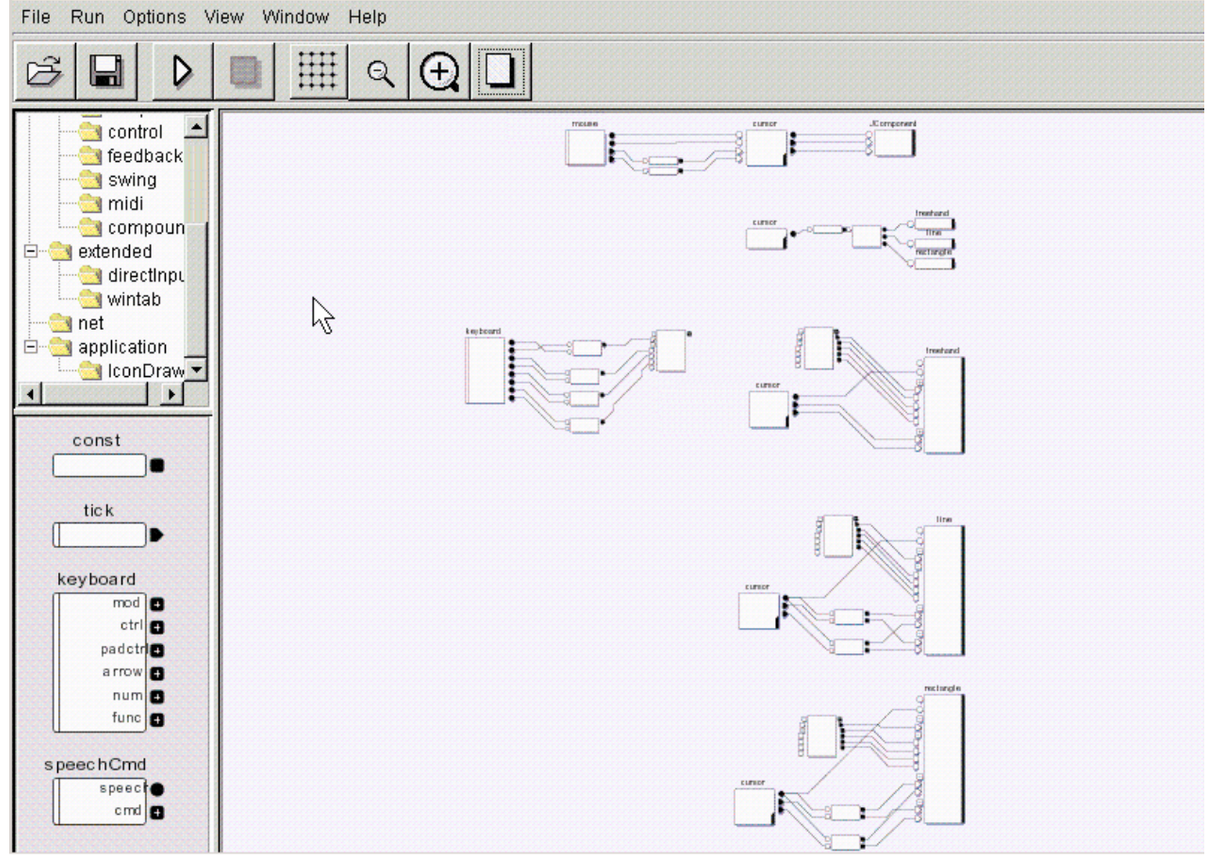
Architecture example III

Reactive - flow

.: Max Msp / Pure data

.: ICON





Using computer vision

Myron Krueger,

"Videoplace", 1970

Current applications:

.: Augmented reality

.: Multitouch tracking

.: *In the air* gestures

Using computer vision

Myron Krueger,

"Videoplace", 1970

Current applications:

.: Augmented reality

.: Multitouch tracking

.: *In the air* gestures

Using computer vision

Myron Krueger,

"Videoplace", 1970

Current applications:

.: Augmented reality

.: Multitouch tracking

.: *In the air* gestures

Vision-based UIs: “Verbs”

.::Detection and Tracking

.::Capturing content

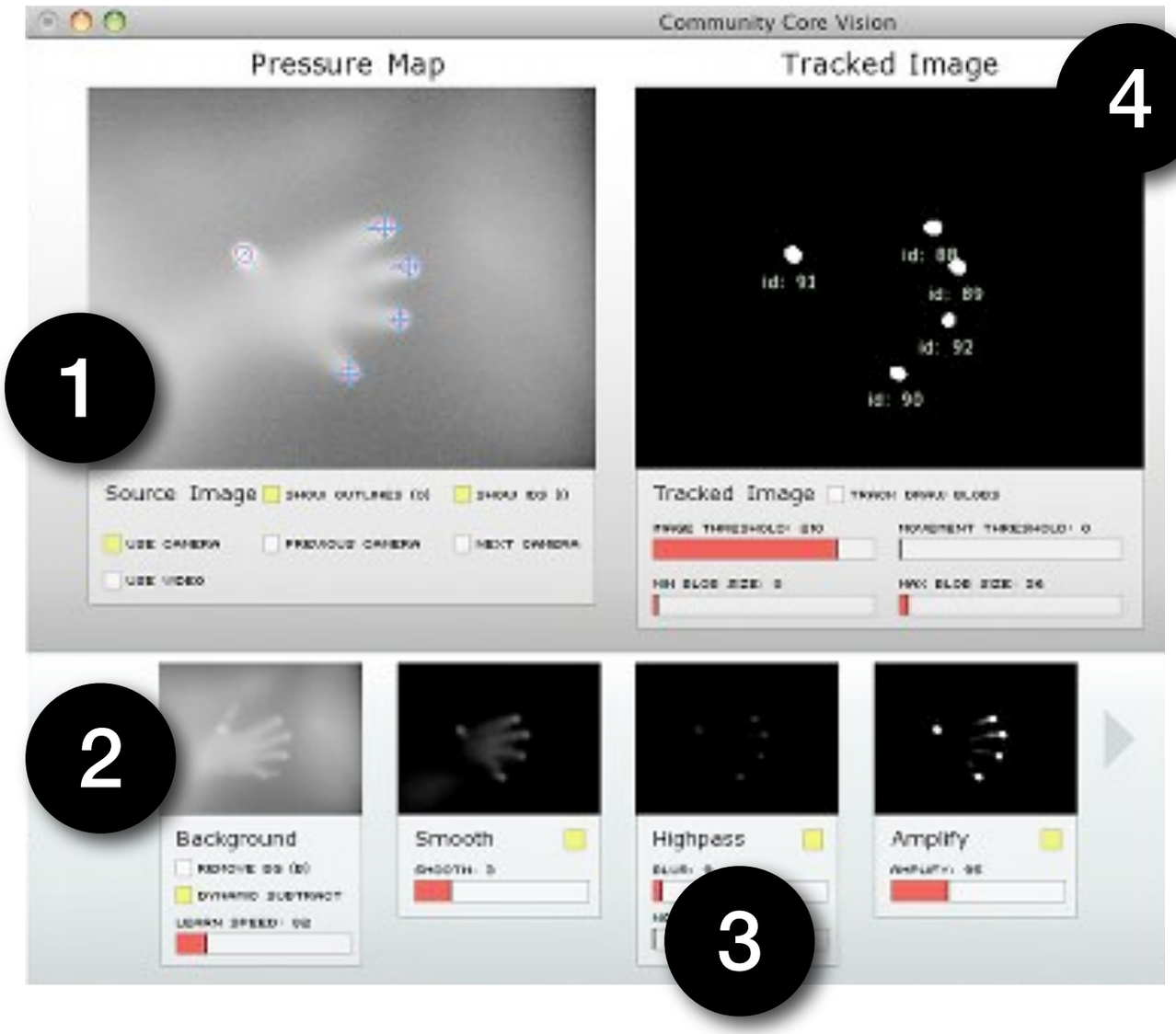
.::Recognition

Vision-based UIs: “Nouns”

- .:People
  - .:Bodies
  - .:Faces
  - .:Hands
- .:Documents
- .:Objects

Basic image processing

1. Capture
2. Background subtraction
3. Processing / Filtering
4. Recognition
5. Tracking



Toolkits

.:Generic:

.:Intel's OpenCV
C API for highly optimized image processing (threshold, dilate, optical flow, …)
http://www.intel.com/research/mrl/research/opencv

.:OpenCV ports for Processing (Java), Flash...

.:For surfaces

.: reacTIVision

.: dtouch

.: Community Core Vision (CCV)

Next?

Touch is moving away from vision

.: capacitive

.: chips doing pixel level vision

.: ...

Touch and feedback?

TeslaTouch

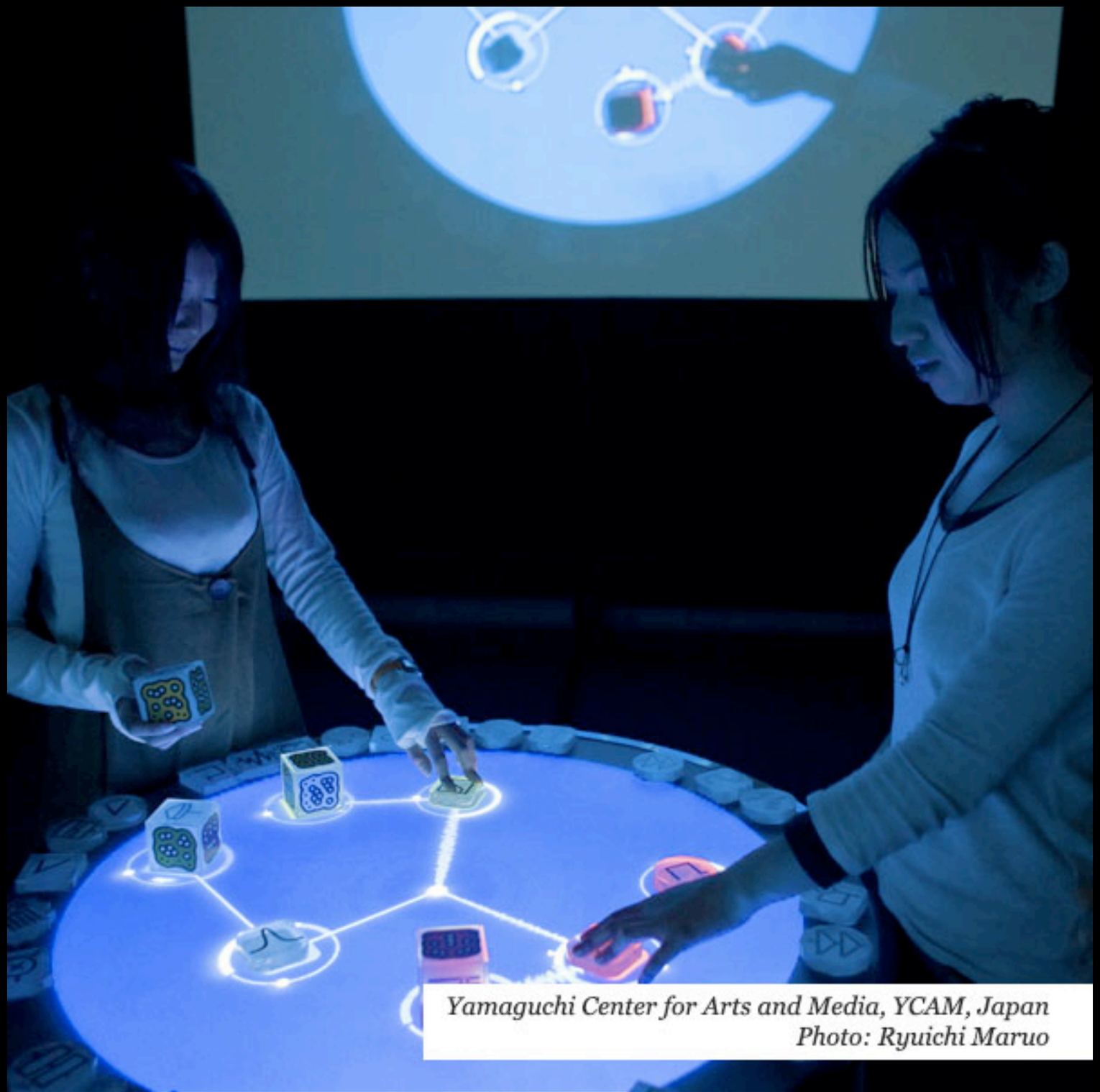
A Tactile Texture Display

Touch and Tangibles

.::Vision

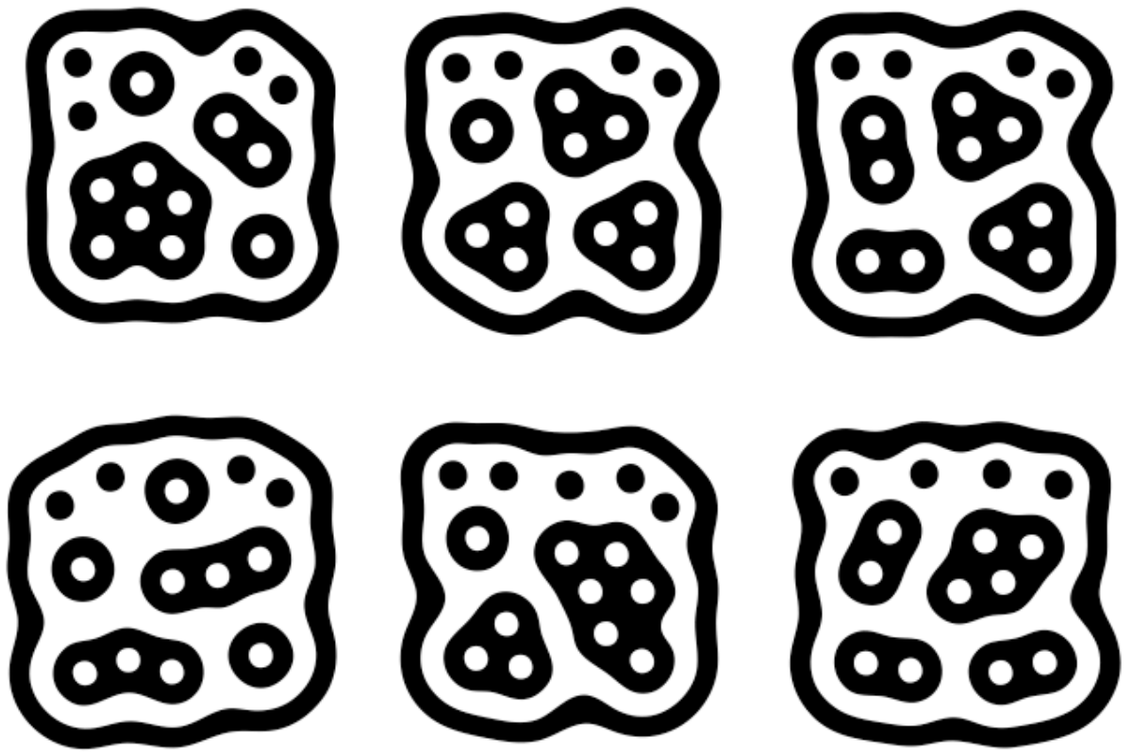
.::Capacitive

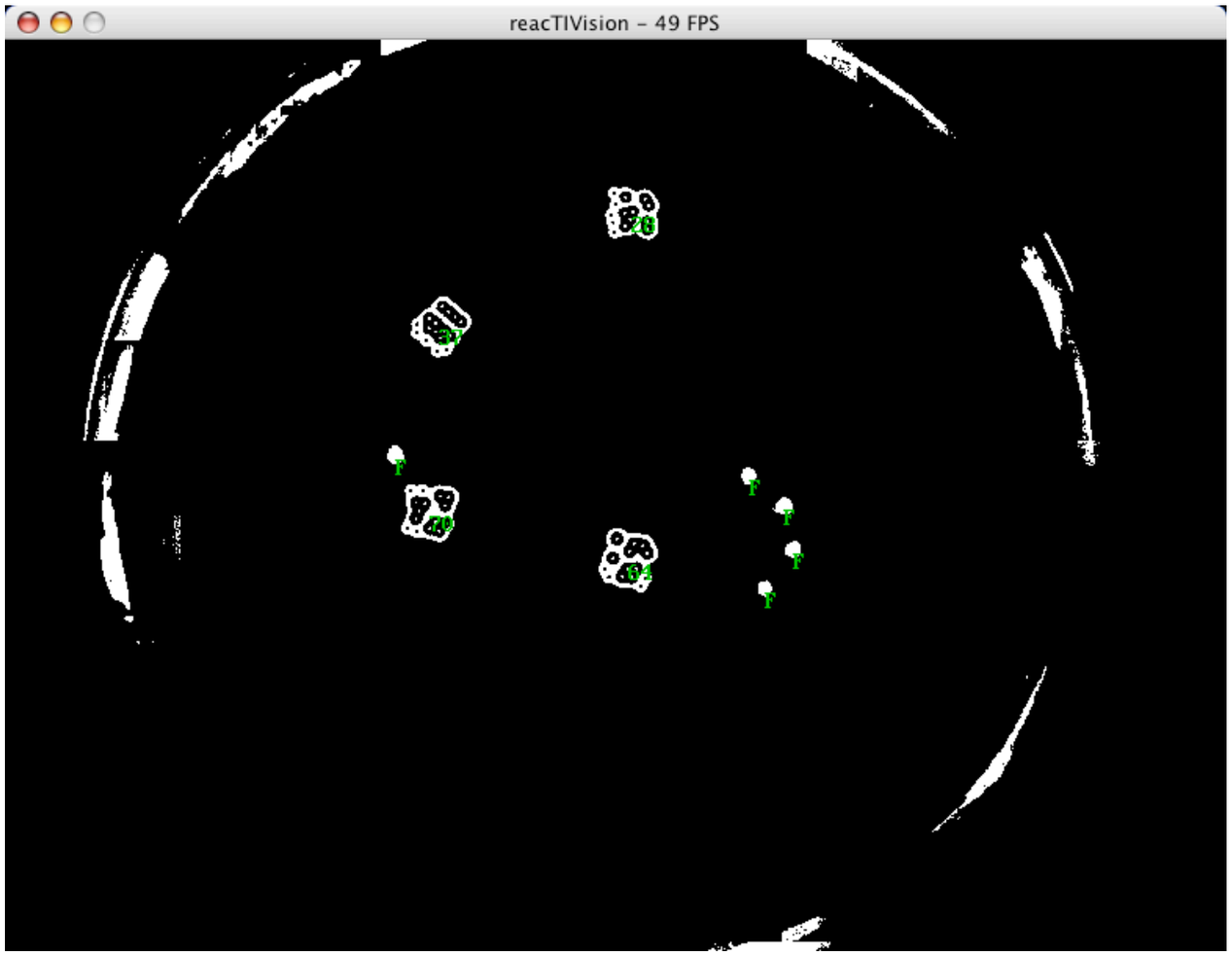
.::Electronics



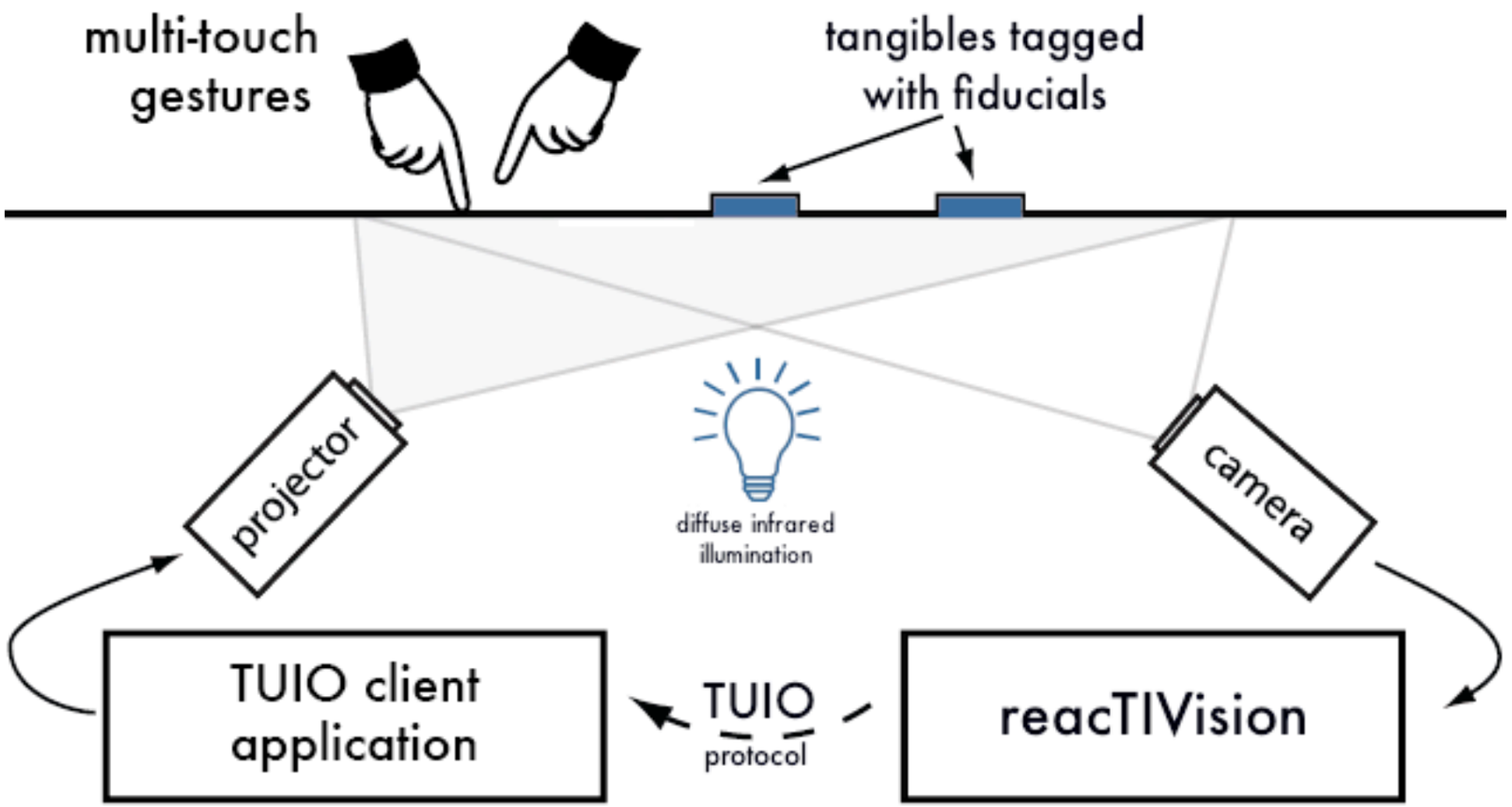
Yamaguchi Center for Arts and Media, YCAM, Japan
Photo: Ryuichi Maruo

Vision

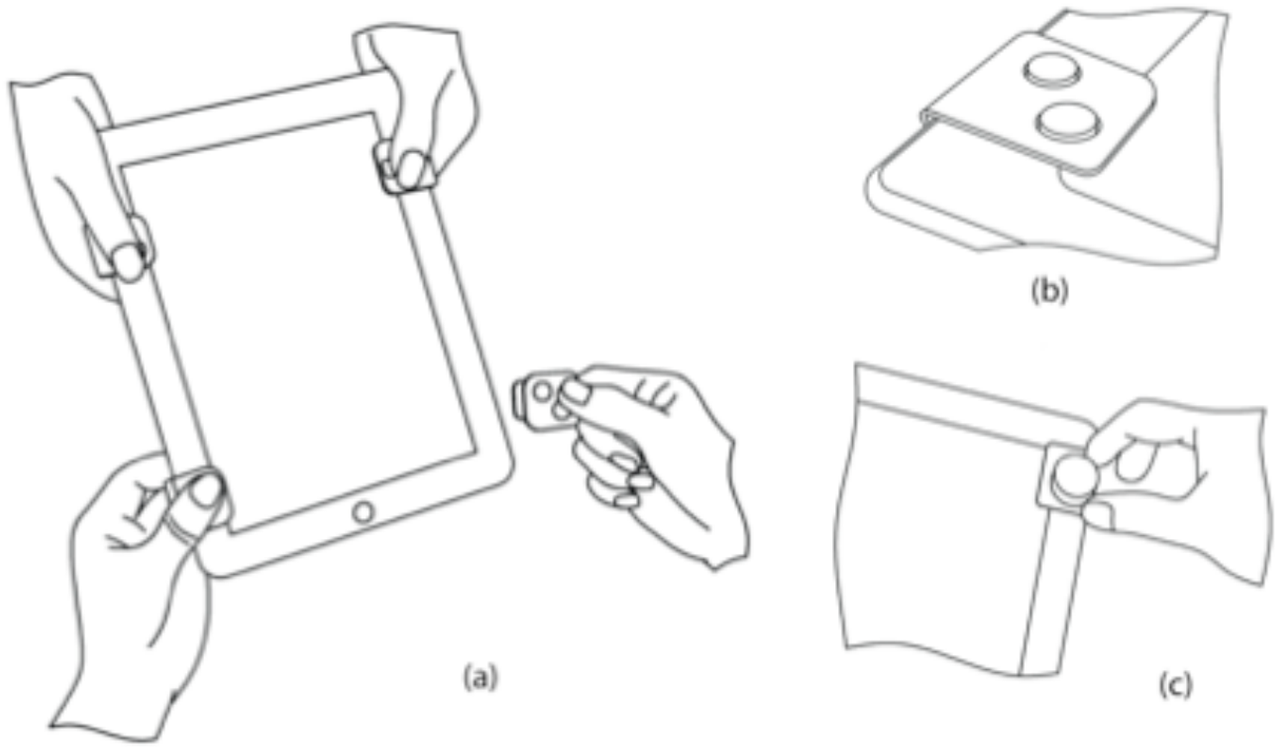


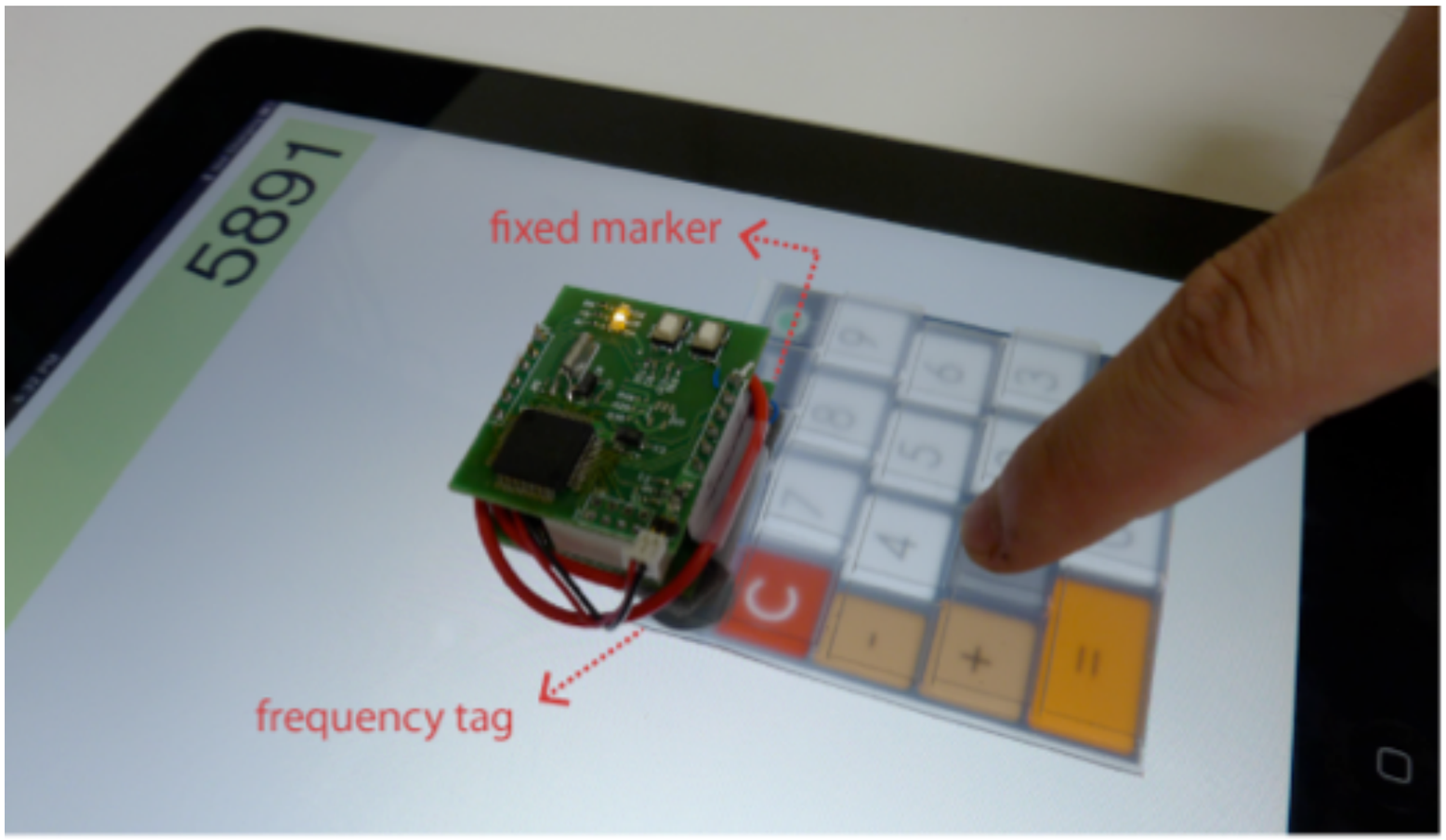


Reactivision



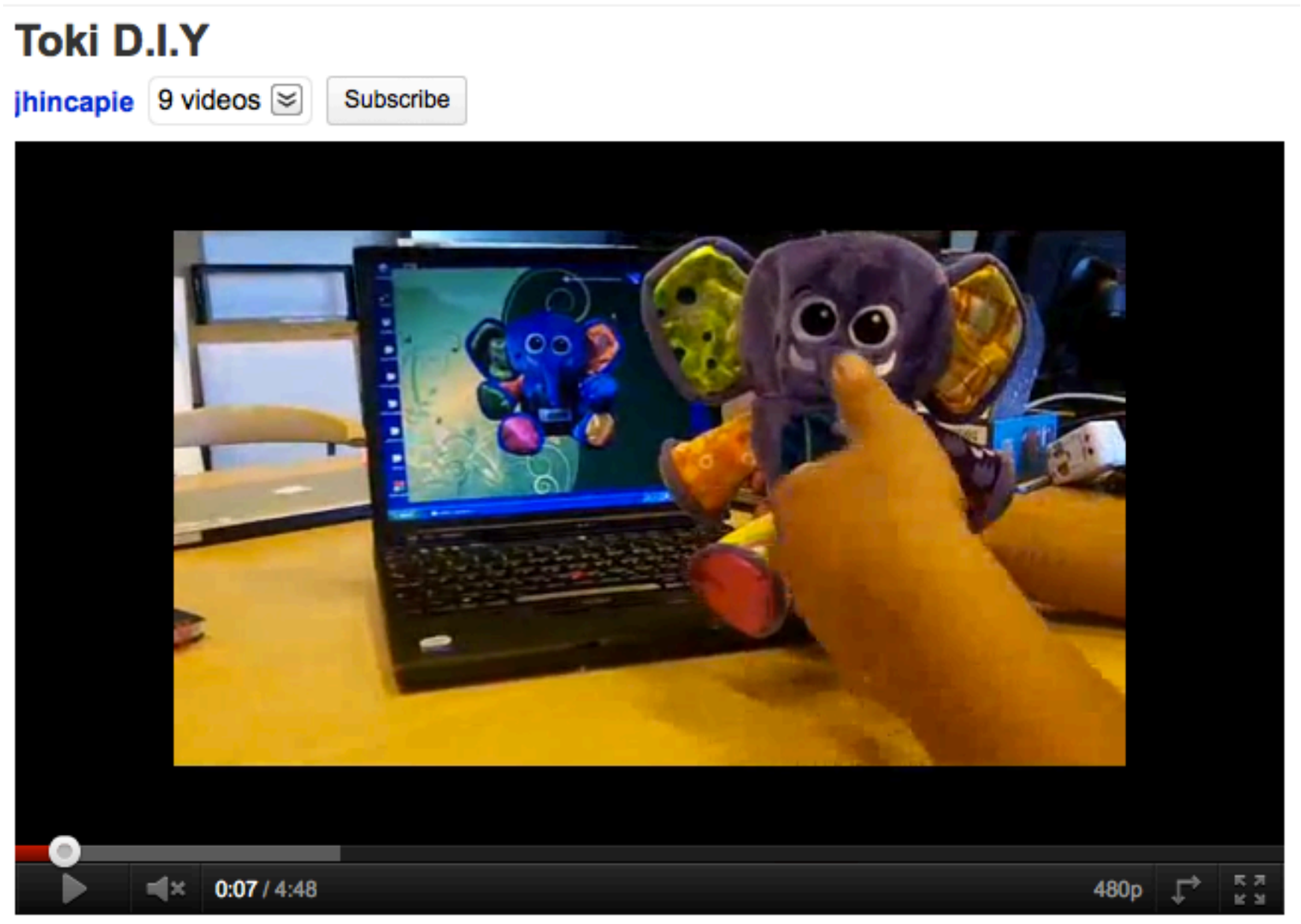
Capacitive sensing





Neng-Hao Yu et al. (2011) TUIC: Enabling Tangible Interaction on Capacitive Multi-touch Displays - CHI 2011

Touch and tangibles



http://itu.dk/people/mortenq/loki/index.html

Gestures

Gestures - for computers (and us)

Acquisition, description

Characterization

identify salient features

Segmentation and interpretation
which command, which parameters?

Execute command

Feedback

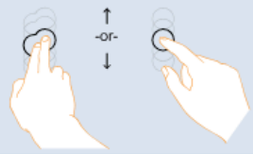
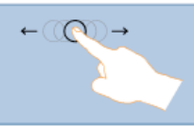
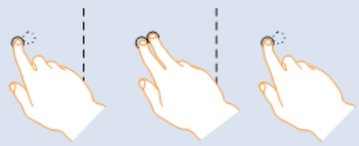
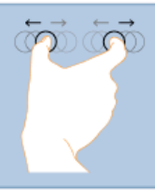
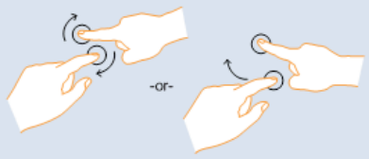
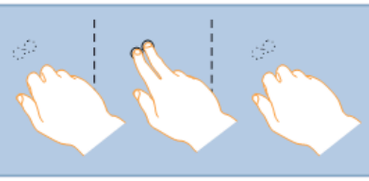
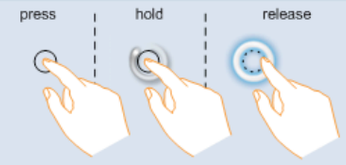
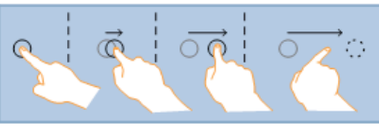
Meaning of gestures

(Cadoz, 1994)

Semiotic: communicate information

Ergotic: create and manipulate artifacts through physical actions

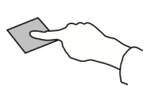
Epistemic: acquire information, through tactile or haptic exploration

GESTURE	WINDOWS USAGE	GESTURE ACTION	ACTION (○= finger down | ◌= finger up)	Single Contact	Multi Contact
Tap / Double Tap	Click / Double Click			✦	✦
Panning with Inertia	Scrolling	Drag 1 or 2 fingers up and down			✦
Selection / Drag (left to right with one finger)	Mouse Drag / Selection	Drag one finger left / right		✦	✦
Press and Tap	Right-click	Press on target and tap using a second finger			✦
Zoom	Zoom (defaults to CTRL key + Scroll wheel)	Move two fingers apart / toward each other			✦
Rotate	No system default unless handled by Application (using WM_GESTURE API)	Move two fingers in opposing directions -or- Use one finger to pivot around another			✦
Two-Finger Tap	N/A – Exposed through Gesture API, used by Application discretion.	Tap two fingers at the same time (where the target is the midpoint between the fingers)			✦
Press and Hold	Right-click	Press, wait for blue ring animation to complete, then release		✦	✦
Flicks	Default: Pan up/ Pan Down/ Back, and Forward	Make quick drag gestures in the desired direction		✦	✦

Windows Touch Gestures Overview

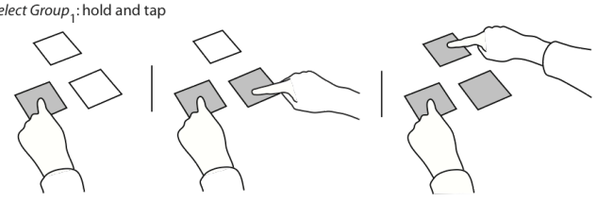
Vocabulary?

Select Single$_1$: tap



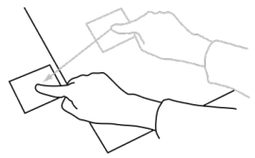
Select Single$_2$: lasso

Select Group$_1$: hold and tap



Select Group$_2$ and *Select Group*$_3$: Use *Select Single*$_1$ or *Select Single*$_2$ on all items in the group.

Delete$_1$: drag offscreen



Delete$_2$: Use *Move*$_2$ with on-screen source and off-screen destination.

Accept: draw check

Help: draw '?'

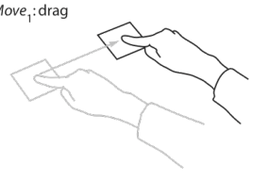
Reject: draw 'X'

Reject$_2$, *Reject*$_3$: If rejecting an object/dialog with an on-screen representation, use *Delete*$_1$ or *Delete*$_2$.

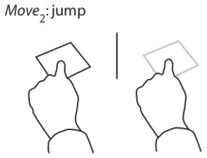
Menu: pull out

Undo: scratch out

Move$_1$: drag

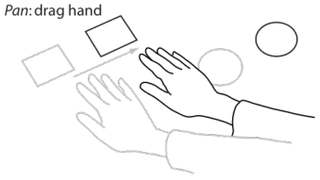


Move$_2$: jump

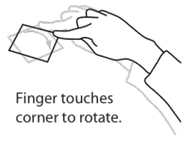


Object jumps to index finger location.

Pan: drag hand



Rotate: drag corner

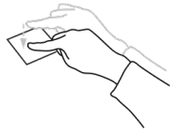


Finger touches corner to rotate.

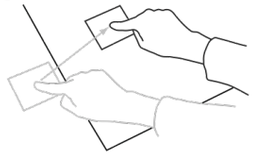
Cut: slash

Cuts current selection (made via *Select Single* or *Select Group*).

Paste$_1$: tap

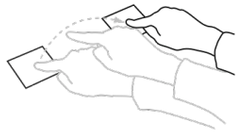


Paste$_2$: drag from offscreen



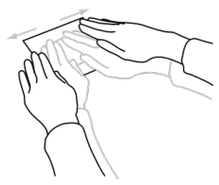
Paste$_3$: Use *Move*$_2$ with off-screen source and on-screen destination.

Duplicate: tap source and destination

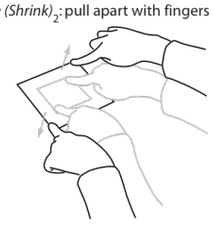


After duplicating, source object is no longer selected.

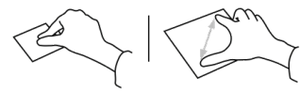
Enlarge (Shrink)$_1$: pull apart with hands



Enlarge (Shrink)$_2$: pull apart with fingers

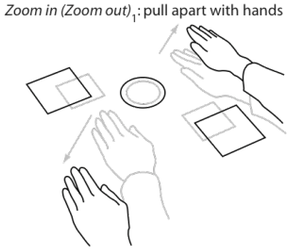


Enlarge (Shrink)$_3$: pinch



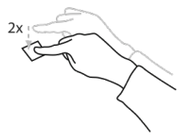
Enlarge (Shrink)$_4$: splay fingers

Zoom in (Zoom out)$_1$: pull apart with hands



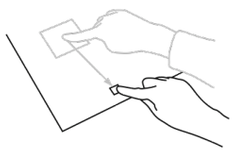
Zoom in (Zoom out)$_{2\text{-}4}$: Use *Enlarge (Shrink)*$_{2\text{-}4}$, performed on background.

Open$_1$: double tap



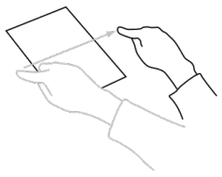
Open$_{2\text{-}5}$: Use *Enlarge*$_{1\text{-}4}$, atop an "openable" object.

Minimize$_1$: drag to bottom of surface



Minimize$_2$: Use *Move*$_2$ to move object to the bottom of the surface (as defined by user's seating position).

Next (Previous): draw line across object



J. O. Wobbrock, M. Ringel Morris and A. D. Wilson. "User-defined gestures for surface computing". In *Proceedings of ACM CHI'09*, pages 1083–1092.

$1 recognizer

.:Unistrokes

.:Invariance

.: rotation

.: scale

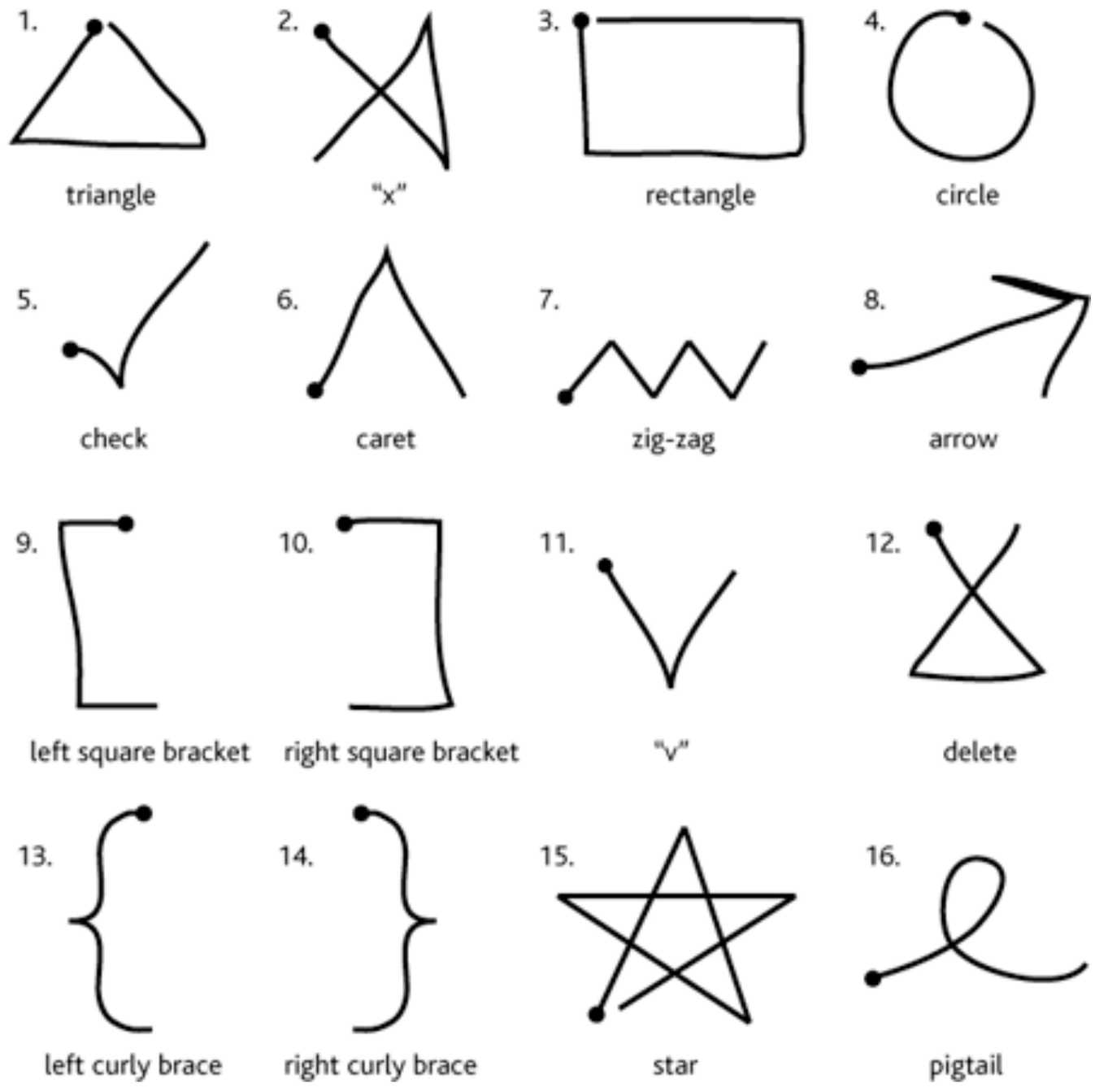
.: position

.:Processing:

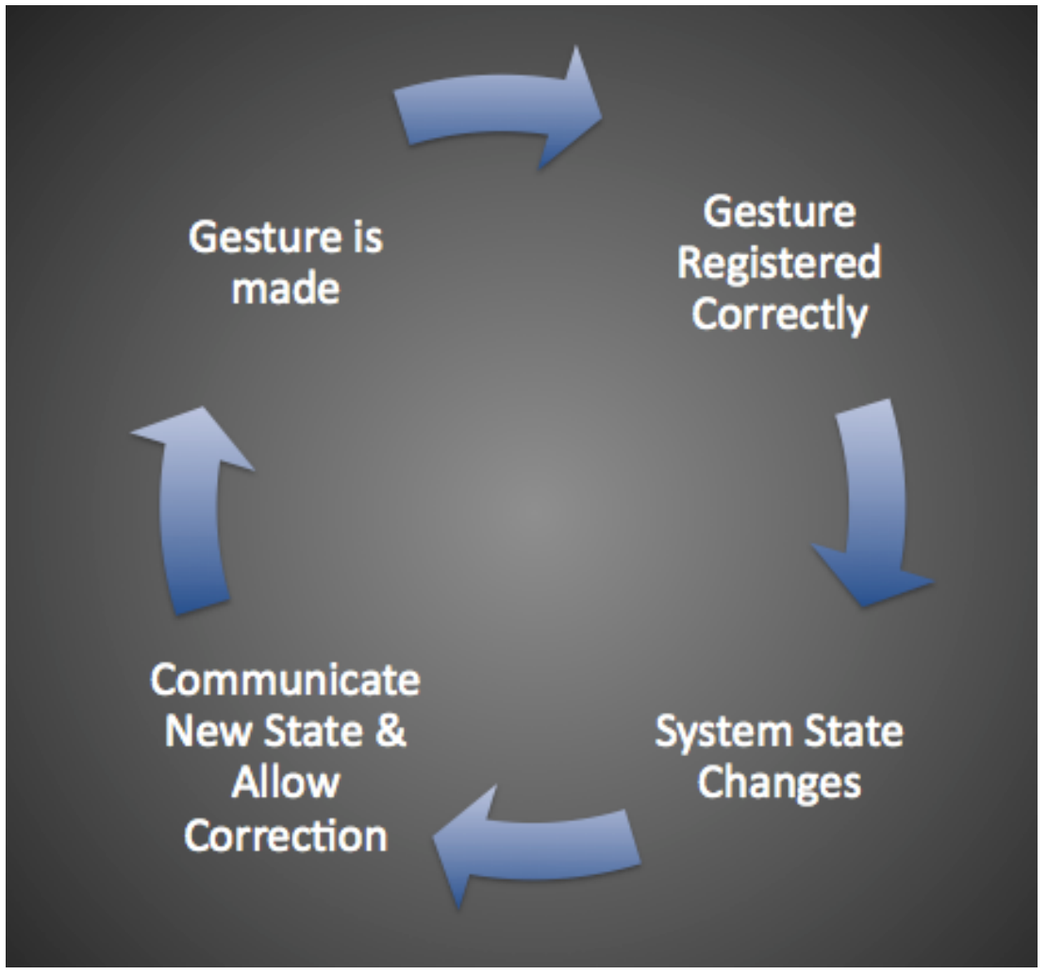
.:Create Template

.:Resampling

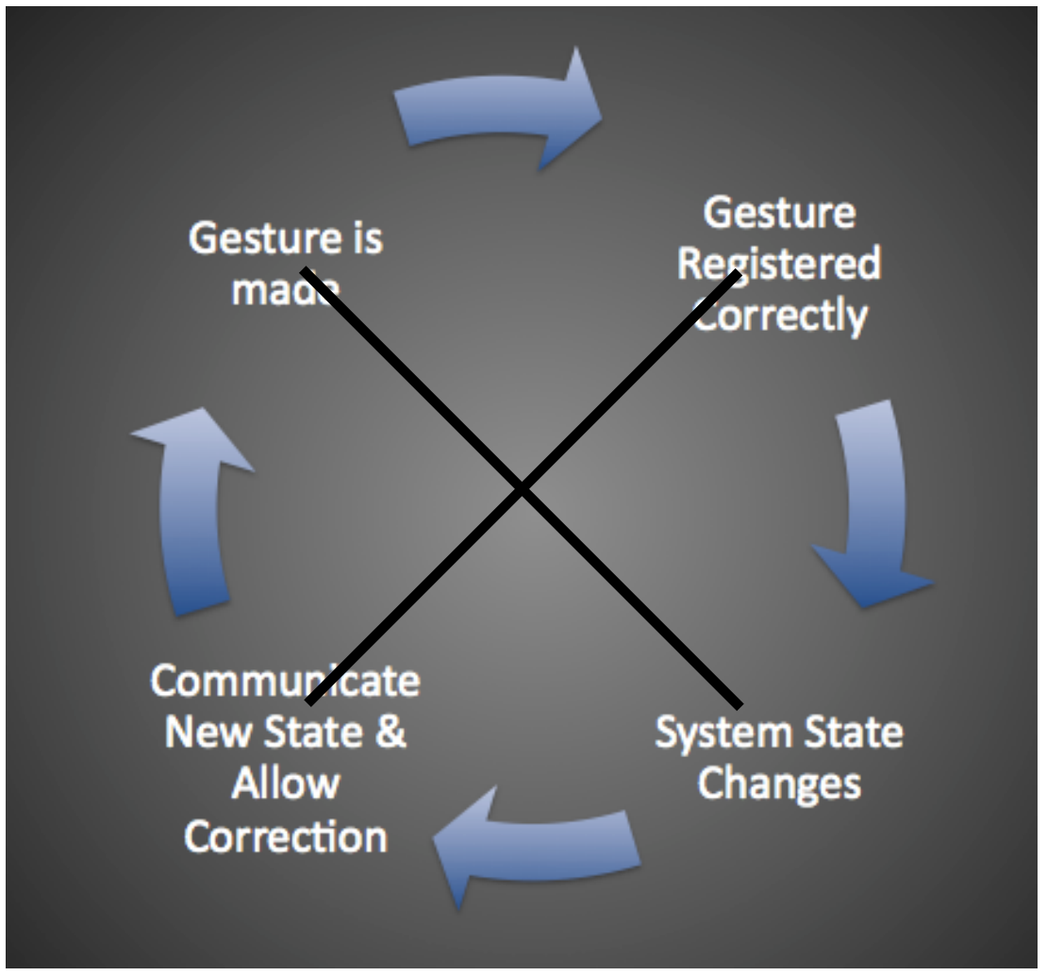
.:Compare input to invariant template(s)

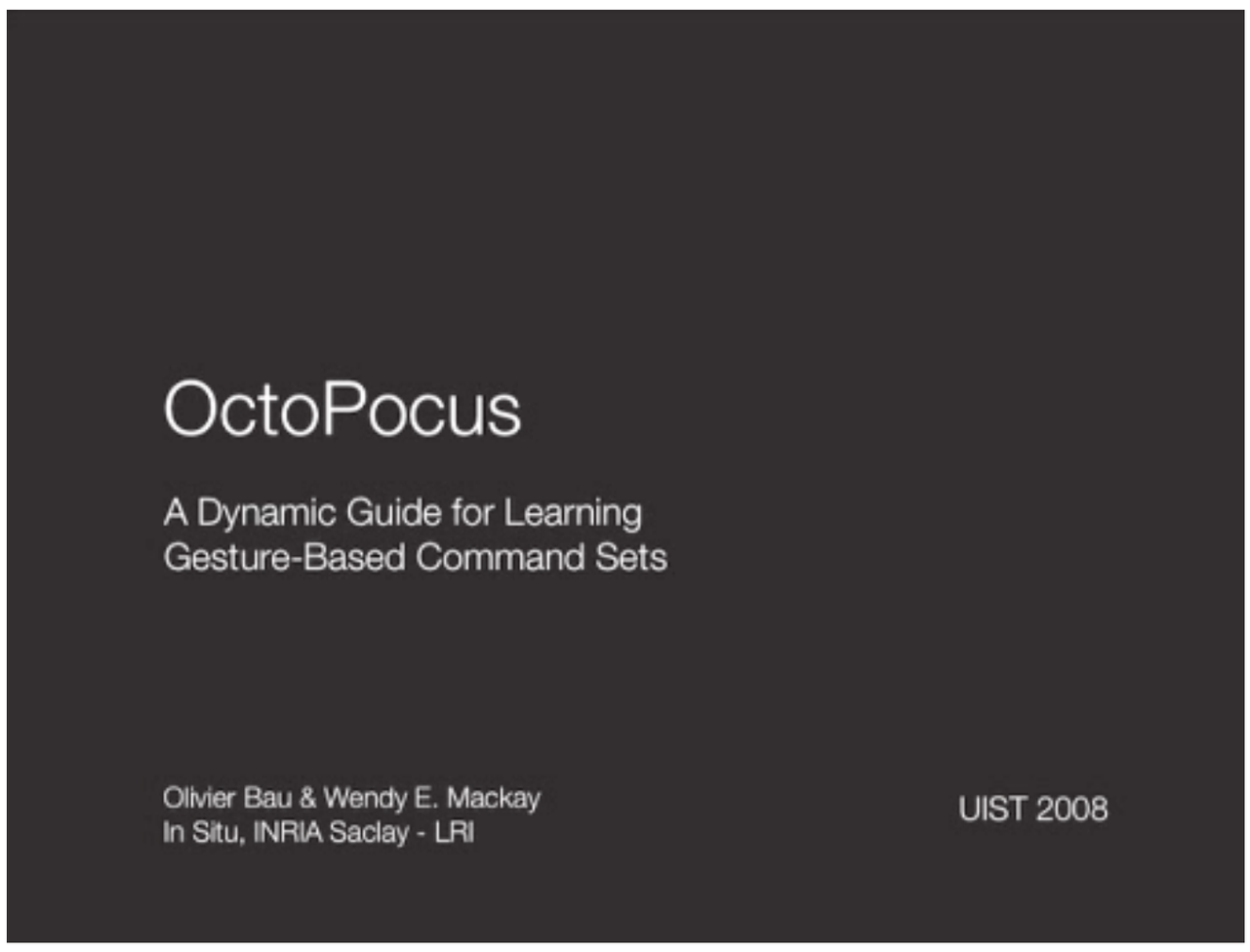


Feedback and gestures



Feedback and gestures





Wrap up

Computing by the inch, foot, & yard

Away from the PC

Continuity

Attention, considerations for calm technologies and Ubicomp systems

Technicalities

Applications

eLabBench

eLabBench

1. Digitization

2. Exploration
& Engineering

Tangible integration

Mediated tabletop interaction