UI is a big topic

GBDA 103: User Experience Design
UI is a big topic

GBDA 103: User Experience Design

CS 349: User Interfaces
CS 449: Human-Computer Interaction

MSCI 343: Human-Computer Interaction

DAC 309: User Experience Design
Welcome

Welcome to ACM CHI 2016: the top conference for Human-Computer Interaction. CHI will take place from May 7 – 12 at San Jose, CA, USA.

CHI 2016 Technical Program Preview
220A  Panel: User Experience (UX) in India
User Experience (UX) in India - ‘We are Not Like This Only’ - We are World Class and Much More!
Apala Lahiri Chavan, Girish Prabhu, Sarit Arora, Janaki Kumar, Sudhindra V

220B  alt.chi: Critical Theory and Pedagogy
Chair: Silvia Lindner
The User Experience in Zen and the Art of Motorcycle Maintenance
Simon Harper
Meaning Reconstruction as an Approach to Analyze Critical Dimensions of HCI Research
Colin M Gray, Austin L Toombs, Christian McKay
Critical Realist HCI
Christopher Frauenberger
Making the Case for an Existential Perspective in HCI Research on Mortality and Death
Victor Kaptelinin

Faceless Interaction - A Conceptual Examination of the Notion of Interface: Past, Present, and Future
Lars-Erik Janlert, Erik Stolterman
Five Provocations for Ethical HCI Research
Barry Brown, Alexandra Weilenmann, Donald McMillan, Airi Lampinen
Acting with Technology: Rehearsing for Mixed-Media Live Performances
Louise Barkhuus, Chiara Rossitto

112  SIG: Refugees and HCI
Refugees and HCI SIG: The Role of HCI in Responding To the Refugee Crisis
Reem Talhouk, Syed Ishtiaque Ahmed, Volker Wulf, Clara Crivellaro, Vasilis Viachokryiakos, Patrick Olivier

114  Case Studies: Tools for Workers
Chair: Permille Bjørn
Untethered Workspaces: A Zones Concept Towards Supporting Operator Movements in Control Rooms
Veronika Dormova, Saad Azhar, Maria Ralph, Jonas Brönmark
From Two CSCW Frameworks to User Requirements Definition for a Retail Planning Collaborative Software
Grégory Petit, Justin Soles
Interactive Colormapping: Enabling Multiple Data Range and Detailed Views of Ocean Salinity
Francesca Samsel, Sebastian Klaassen, Mark Petersen, Terece L Turton, Greg D Abram, David H Rogers, James Ahrens
Designing the Alarm Management User Experience for Patient Monitoring
Sharoda A Paul, Alexander K Carroll, Stephen Treacy

LL21C  Papers: Computer Supported Parenting
Chair: Tarmo Haukka
THE DESIGN OF EVERYDAY THINGS

Donald A. Norman

Previously published as THE PSYCHOLOGY OF EVERYDAY THINGS

Author of EMOTIONAL DESIGN

More than 100,000 copies sold

With a new introduction by the author

"Design may be our top competitive edge. This book is a joy—fun and of the utmost importance."

TOM PETERS
How do programmers think about user interfaces?

What tools and techniques do they use to create user interfaces?
Topics

Model-view-controller paradigm

Direct manipulation

User interface toolkits

Building interfaces with ControlP5
Model-View-Controller (MVC)

A standard *paradigm* for describing the components of an interactive program.
**Model**: the underlying object or data being manipulated by the program.

**View**: the means by which the model is communicated to the user.

**Controller**: the means by which the user is able to manipulate the model.
VIEW updates MODEL
CONTROLLER manipulates MODEL
color the_colour;

void setup()
{
  size( 200, 200 );
}

void draw()
{
  background( the_colour );
}

void mouseMoved()
{
  float r = map( mouseX, 0, width, 0, 255 );
  float g = map( mouseY, 0, height, 0, 255 );
  the_colour = color( r, g, 0 );
}
Direct Manipulation

The controller is coupled to the view (or equal to the view)

Interaction is continuous and incremental.
Hit Testing

Every on-screen element that can be manipulated needs a *hit test*—a way to determine if the mouse is over that element.
ellipse( cx, cy, 2 * rad, 2 * rad );

Draw a circle with centre (cx, cy) and radius rad.
ellipse( cx, cy, 2 * rad, 2 * rad );

if ( dist( mouseX, mouseY, cx, cy ) <= rad ) {
  ...

  Hit test for the same circle.
ellipse( cx, cy, 2 * rad, 2 * rad );

if ( dist( mouseX, mouseY, cx, cy ) <= rad ) {
    ...
}

rect( ax, ay, w, w );

if ( (mouseX >= ax) && (mouseX <= (ax+w))
    && (mouseY >= ay) && (mouseY <= (ay+w)) ) {
    ...
}
Direct Manipulation and classes

```java
class InteractiveThingy {
    // Fields (i.e., part of the Model)

    void drawSelf() {
        // Draw this object in the sketch (View)
    }

    boolean hitTest( float x, float y ) {
        // Is point (x,y) inside this object? (Controller)
    }
}
```
Handling events

Are we dragging the circle?

```java
boolean active;

void mousePressed()
{
  float d = dist(cx, cy, mouseX, mouseY);
  if (d < rad)
  {
    active = true;
  }
}

void mouseDragged()
{
  if (active)
  {
    cx = mouseX;
    cy = mouseY;
  }
}

void mouseReleased()
{
  active = false;
}
```

Hit test

Controller updates the model
Handling events

boolean active;

void mousePressed()
{
    float d = dist(cx, cy, mouseX, mouseY);
    if (d < rad)
    {
        active = true;
    }
}

void mouseDragged()
{
    if (active)
    {
        cx += mouseX - pmouseX;
        cy += mouseY - pmouseY;
    }
}

void mouseReleased()
{
    active = false;
}
If we have an interface with multiple elements, we need a way to keep track of which one was hit.

```java
boolean circle_active = false;
boolean square_active = false;

void draw()
{
    drawCircle( ... );
    drawSquare( ... );
}

void mousePressed()
{
    circle_active = false;
    square_active = false;

    if( hitTestCircle( ... ) ) {
        circle_active = true;
    } else if( hitTestSquare( ... ) ) {
        square_active = true;
    }
}
```
If we have an interface with multiple elements, we need a way to keep track of which one was hit.

```java
boolean circle_active = false;
boolean square_active = false;

void draw()
{
    drawSquare(...);
    drawCircle(...);
}

void mousePressed()
{
    circle_active = false;
    square_active = false;

    if( hitTestCircle(...) ) {
        circle_active = true;
    } else if( hitTestSquare(...) ) {
        square_active = true;
    }
}
```
If we have an interface with multiple elements, we need a way to keep track of which one was hit.

```java
Circle[] some_circles;
int active = -1;

void draw()
{
    for (int idx = 0; idx < some_circles.length; ++idx)
    {
        drawCircle(some_circles[idx]);
    }
}

void mousePressed()
{
    active = -1;
    for (int idx = some_circles.length - 1; idx >= 0; --idx)
    {
        if (hitTest(some_circles[idx]))
        {
            active = idx;
            return;
        }
    }
}
```
Small handles

VIEW

co
Small handles
Complex shapes

How can we hit test a shape with a complicated boundary?
Proxy geometry
Selection buffer
Direct manipulation notes

Shift objects using mouseX-pmouseX and mouseY-pmouseY, don’t “teleport” them.

Draw objects from back-to-front, hit test them from front-to-back.

Make hit test region usable, regardless of how it’s drawn.
Toolkits

Some interactions are so canonical that it makes sense to invent standardized “widgets” to handle them.

Perform an action: **Button**
Set a continuous value: **Slider**
Enter text: **Text field**

Classes and objects are perfect for this!
GTK
Minimal ControlP5

```java
import controlP5.*;
ControlP5 ui;
void setup()
{
    size( 500, 500 );
    ui = new ControlP5( this );
}
```

- **Import directive**
- **Global “factory object”**
- **Initialize the library, “build the factory”**
Minimal ControlP5

```java
import controlP5.*;

ControlP5 ui;

void setup()
{
  size( 500, 500 );
  ui = new ControlP5( this );
  ui.addButton( "Hello!" );
}
```

Add a widget
void setup()
{
    size( 500, 500 );

    ui = new ControlP5( this );

    Button hello = ui.addButton( "Hello!" );
    hello.setPosition( 200, 200 );
    hello.setSize( 120, 60 );
}
void setup()
{
    size( 500, 500 );

    ui = new ControlP5( this );

    Button hello = ui.addButton( "Hello!" );
    hello.setPosition( 200, 200 );
    hello.setSize( 120, 60 );
}

Hold on to an object that represents the button
```java
void setup()
{
    size( 500, 500 );

    ui = new ControlP5( this );

    Button hello = ui.addButton( "Hello!" );
    hello.setPosition( 200, 200 );
    hello.setSize( 120, 60 );
}

Set some of the button’s properties
```
void setup()
{
  size( 500, 500 );

  ui = new ControlP5( this );

  Button hello = ui.addButton( "Hello!" )
    .setPosition( 200, 200 );
  .setSize( 120, 60 );
}

```java
class Point {
    float x;
    float y;

    Point( float xIn, float yIn ) {
        x = xIn;
        y = yIn;
    }

    Point setX( float xIn ) {
        x = xIn;
        return this;
    }
}
```
void setup()
{
    size( 500, 500 );

    ui = new ControlP5( this );
    ui.setFont( createFont( "Gotham-Bold", 24 ) );

    Button hello = ui.addButton( "Hello!" )
        .setPosition( 200, 200 );
    .setSize( 120, 60 );
}
Handling UI events

How do we discover when a button was pressed, and what can we do when that happens?

ControlP5 defines a new hook, controlEvent().
void setup()
{
    size( 500, 500 );

    ui = new ControlP5( this );
    ui.setFont( createFont( "Gotham-Bold", 24 ) );

    Button hello = ui.addButton( "Hello!" )
        .setPosition( 200, 200 );
        .setSize( 120, 60 );
}

void controlEvent( ControlEvent ce )
{
    println( "Something happened!" );
}
void setup()
{
    size( 500, 500 );

    ui = new ControlP5( this );
    ui.setFont( createFont( "Gotham-Bold", 24 ) );

    Button hello = ui.addButton( "Hello!" )
        .setPosition( 200, 200 )
        .setSize( 120, 60 );
}

Name of the hook

void controlEvent( ControlEvent ce )
{
    println( "Something happened!" );
}
void setup()
{
    size( 500, 500 );

    ui = new ControlP5( this );
    ui.setFont( createFont( "Gotham-Bold", 24 ) );

    Button hello = ui.addButton( "Hello!" )
        .setPosition( 200, 200 );
        .setSize( 120, 60 );
}

void controlEvent(ControlEvent ce)
{
    println( "Something happened!" );
}
import controlP5.*;

ControlP5 ui;

Button b1;
Button b2;

void setup()
{
  size( 500, 500 );
  ui = new ControlP5( this );
  b1 = ui.addButton( "One" );
  b2 = ui.addButton( "Two" );
}

void draw()
{
}

void controlEvent( ControlEvent ce )
{
  if( ce.isFrom( b1 ) ) {
    println( "One" );
  } else if( ce.isFrom( b2 ) ) {
    println( "Two" );
  }
}
controlIP5
A GUI (graphical user interface) library for processing.

Download
Download controlIP5 version 2.2.5 release 07/30/2015
controlIP5.zip

This version has been tested with processing 2.2.1, for earlier version see the download list.

Older Versions
For older versions see the download list on the google code project page.

About
controlIP5 is a library written by Andreas Schlegel for the programming environment processing. Last update, 07/30/2015.

Controllers to build a graphical user interface on top of your processing sketch include Sliders, Buttons, Toggles, Knobs, Textfields, RadioButtons, Checkboxes amongst others and can be easily added to a processing sketch. They can be arranged in separate control PGraphics contexts, and can be organized in tabs or groups. → read more.

FAQ
Frequently Asked Questions might have been answered in the processing forum. Have a look and search for controlIP5 here. Or file an issue on github

Some projects using controlIP5
decode
cop15 identity
generative gestaltung
predray
fractaltables
typestar
2D SuperShapes

Installation
Unzip and put the extracted controlIP5 folder into the libraries folder of your processing sketches. Reference and examples are included in the controlIP5 folder.
Examples

Find a list of examples in the current distribution of controlP5, or have a look by following the links below. If you want to share any examples, please let me know (andi at sojamo dot de).

<table>
<thead>
<tr>
<th>controllers</th>
</tr>
</thead>
<tbody>
<tr>
<td>controllers/ControlP5accordion</td>
</tr>
<tr>
<td>controllers/ControlP5bang</td>
</tr>
<tr>
<td>controllers/ControlP5button</td>
</tr>
<tr>
<td>controllers/ControlP5canvas</td>
</tr>
<tr>
<td>controllers/ControlP5chart</td>
</tr>
</tbody>
</table>
Direct manipulation and a toolkit can co-exist in one interface.
Eye tracking

If you are not satisfied with the baby leakage protection, you will get your money back. Read more about our leakfree guarantee at www.baby.com
Myo Armband
On-world interfaces