Module 05
User Interfaces
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
<table>
<thead>
<tr>
<th>Room</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>220A</td>
<td>Panel: User Experience (UX) in India &lt;br&gt; User Experience (UX) in India - 'We are Not Like This Only' - We are World Class and Much More! &lt;br&gt; Apala Lahiri Chavan, Girish Prabhu, Sarit Arora, Janaki Kumar, Sudhindra V</td>
</tr>
<tr>
<td>220B</td>
<td>altchi: Critical Theory and Pedagogy &lt;br&gt; Chair: Silvia Lindner &lt;br&gt; The User Experience in Zen and the Art of Motorcycle Maintenance &lt;br&gt; Simon Harper &lt;br&gt; Meaning Reconstruction as an Approach to Analyze Critical Dimensions of HCI Research &lt;br&gt; Colin M Gray, Austin L Toombs, Christian McKay &lt;br&gt; Critical Realist HCI &lt;br&gt; Christopher Frauenberger &lt;br&gt; Making the Case for an Existential Perspective in HCI Research on Mortality and Death &lt;br&gt; Victor Kaptelinin</td>
</tr>
<tr>
<td>210D</td>
<td>Course: C01 &lt;br&gt; Research Methods for Child Computer Interaction (2/2) &lt;br&gt; Janet C Read, Shuli Gilutz</td>
</tr>
<tr>
<td>210H</td>
<td>Course: C03 &lt;br&gt; Designing with the Mind in Mind: The Psychological Basis for UI Design Guidelines (2/2) &lt;br&gt; Jeff A Johnson</td>
</tr>
<tr>
<td>210C</td>
<td>Course: C05 &lt;br&gt; Introduction To Human Computer Interaction (2/2) &lt;br&gt; Jonathan Lazar, Simone D J Barbosa</td>
</tr>
<tr>
<td>210G</td>
<td>Course: C08 &lt;br&gt;</td>
</tr>
<tr>
<td>112</td>
<td>SIG: Refugees and HCI &lt;br&gt; Refugees and HCI SIG: The Role of HCI in Responding To the Refugee Crisis &lt;br&gt; Reem Tahhouk, Syed Ishtiaque Ahmed, Volker Wulf, Clara Crivellaro, Vasilis Viachokyiakos, Patrick Olivier</td>
</tr>
<tr>
<td>114</td>
<td>Case Studies: Tools for Workers &lt;br&gt; Chair: Pernille Bjørn &lt;br&gt; Untethered Workspaces: A Zones Concept Towards Supporting Operator Movements in Control Rooms &lt;br&gt; Veronika Domova, Saad Azhar, Maria Ralph, Jonas Brönmark &lt;br&gt; From Two CSCW Frameworks to User Requirements Definition for a Retail Planning Collaborative Software &lt;br&gt; Grégory Petit, Justin Sales &lt;br&gt; Interactive Colormapping: Enabling Multiple Data Range and Detailed Views of Ocean Salinity &lt;br&gt; Francesca Samsel, Sebastian Klaassen, Mark Petersen, Terece L Turton, Greg D Abram, David H Rogers, James Ahrens &lt;br&gt; Designing the Alarm Management User Experience for Patient Monitoring &lt;br&gt; Sharoda A Paul, Alexander K Carroll, Stephen Treacy</td>
</tr>
<tr>
<td>LL21C</td>
<td>Papers: Computer Supported Parenting &lt;br&gt; Chair: Fatima Rana</td>
</tr>
</tbody>
</table>
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

THE DESIGN OF EVERYDAY THINGS

Previously published as THE PSYCHOLOGY OF EVERYDAY THINGS

DONALD A. NORMAN

AUTHOR OF EMOTIONAL DESIGN
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.
USER

VIEW

CONTROLLER

MODEL

sees 
uses 
updates 
manipulates
```cpp
color the_colour;

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

define 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 );
}
```

**Model**
```
color the_colour;
```

**View**
```
void draw()
{
    background( the_colour );
}
```

**Controller**
```
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.
Handles
Hit Testing

Every on-screen element that can be manipulated needs a *hit test*—a function that determines 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)) ) {
    ...
}
Handling events

boolean active;  
Are we dragging the circle?

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;
}
Handling events

```java
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.

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
Small handles
Complex shapes

How can we hit test a shape with a complicated boundary?
Proxy geometry
Selection buffer
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!
<table>
<thead>
<tr>
<th>Label</th>
<th>Small text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hint text</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>OFF</td>
</tr>
<tr>
<td>Selected item</td>
<td></td>
</tr>
<tr>
<td>Item with a value</td>
<td>value</td>
</tr>
<tr>
<td>address</td>
<td>101 Main Street</td>
</tr>
<tr>
<td>Button</td>
<td></td>
</tr>
</tbody>
</table>

**Header 1**
- List item one
- List item two
- List Item three
- List item four

**Header 2**
- Another item

**Alert**
- Alert text goes here

- No
- Yes

**Buttons**
- Red Button
- Gray Button
- Green Button
- Cancel Button
- Disabled Button

**Date Picker**
- Thu Jan 23 4 50 AM
- Fri Jan 24 5 55 AM
- Sat Jan 25 6 00 PM
- Sun Jan 26 7 05
- Mon Jan 27 8 10

**Calendar**
- September 28 2006
- October 29 2007
- November 30 2008
- December 31 2009
- January 01 2010

**Keyboard**
- Q W E R T Y U I O P

**iOS**
- This is a prompt from the application
Minimal ControlP5

```java
import controlP5.*;  // Import directive
ControlP5 ui;        // Global "factory object"

void setup()
{
  size( 500, 500 );
  ui = new ControlP5( this );
}
```

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!" );
}

void draw()
{
}
```

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
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");
  }
}
controlP5
A GUI (graphical user interface) library for processing.

Download
Download controlP5 version 2.2.5 release 07/30/2015
controlP5.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
controlP5 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 controlP5 here. Or file an issue on github

Some projects using controlP5
decode
copp15 identity
generative gestaltung
predray
fractallables
typestar
3D SuperShape

Installation
Unzip and put the extracted controlP5 folder into the libraries folder of your processing sketches. Reference and examples are included in the controlP5 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).

controllers

controllers/ControlP5accordion

controllers/ControlP5button

controllers/ControlP5canvas

controllers/ControlP5chart
Direct manipulation and a toolkit can co-exist in one interface.
Kinect
Eye tracking

Eye gentle for the most sensitive skin.

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