Module 05

User Interfaces

CS 106 Winter 2019

UI is a big topic

GBDA 103: User Experience Design

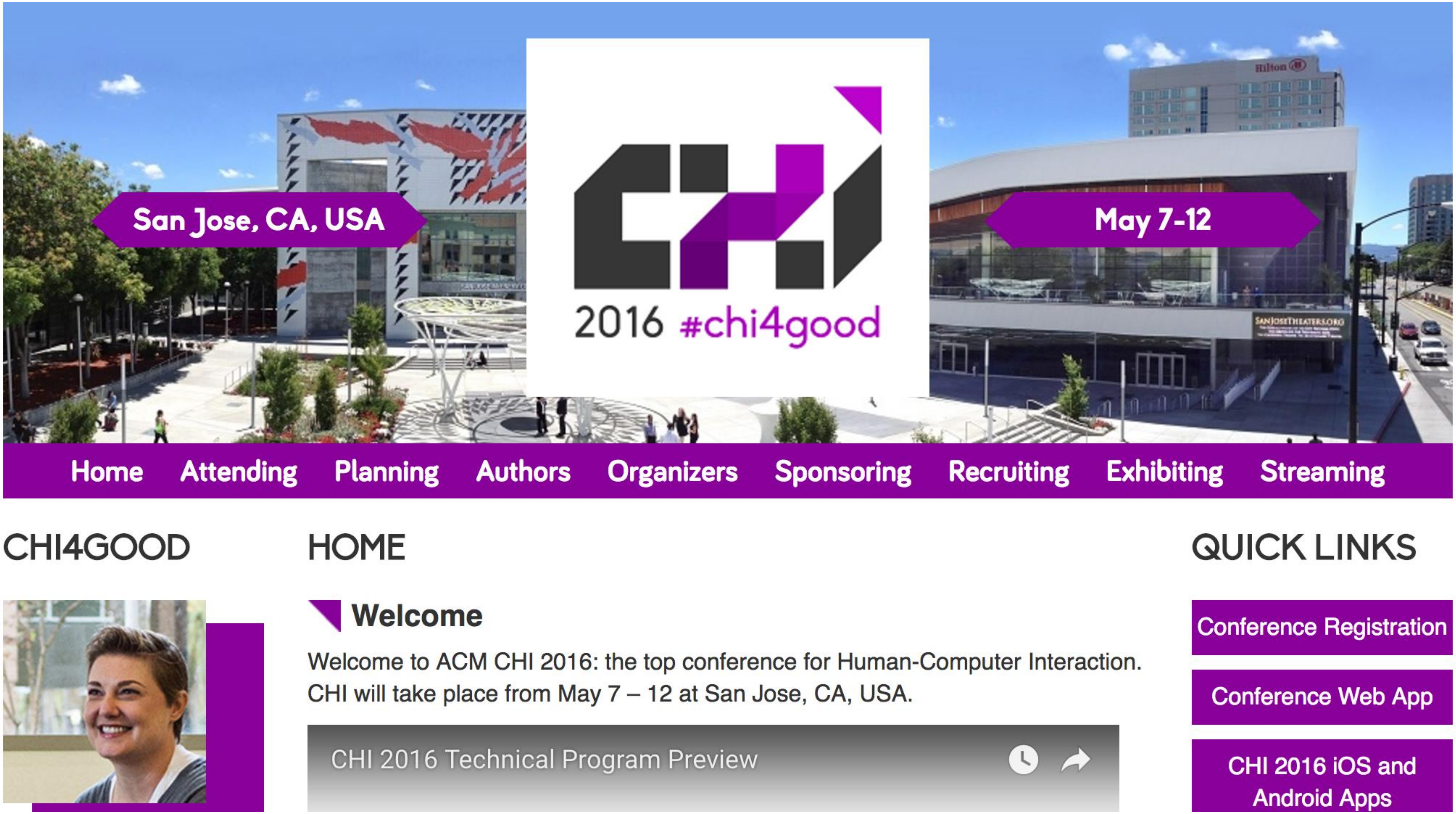
UI is a big topic

GBDA 103: User Experience Design

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

MSCI 343: Human-Computer Interaction

DAC 309: User Experience Design



CHI4GOOD

HOME

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.

QUICK LINKS

MONDAY 14:30 PM - 15:50 PM

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 Lindtner

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

210D Course: C01

Research Methods for Child Computer Interaction (2/2)

Janet C Read, Shuli Gilutz

210H Course: C03

Designing with the Mind in Mind: The Psychological Basis for UI Design Guidelines (2/2)

Jeff A Johnson

210C Course: C05

Introduction To Human Computer Interaction (2/2)

Jonathan Lazar, Simone D J Barbosa

210G Course: C08

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 Vlachokyriakos, Patrick Olivier

114 Case Studies: Tools for Workers

Chair: Pernille Bjorn

Untethered Workspaces: A Zones Concept Towards Supporting Operator Movements in Control Rooms

Veronika Domova, 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

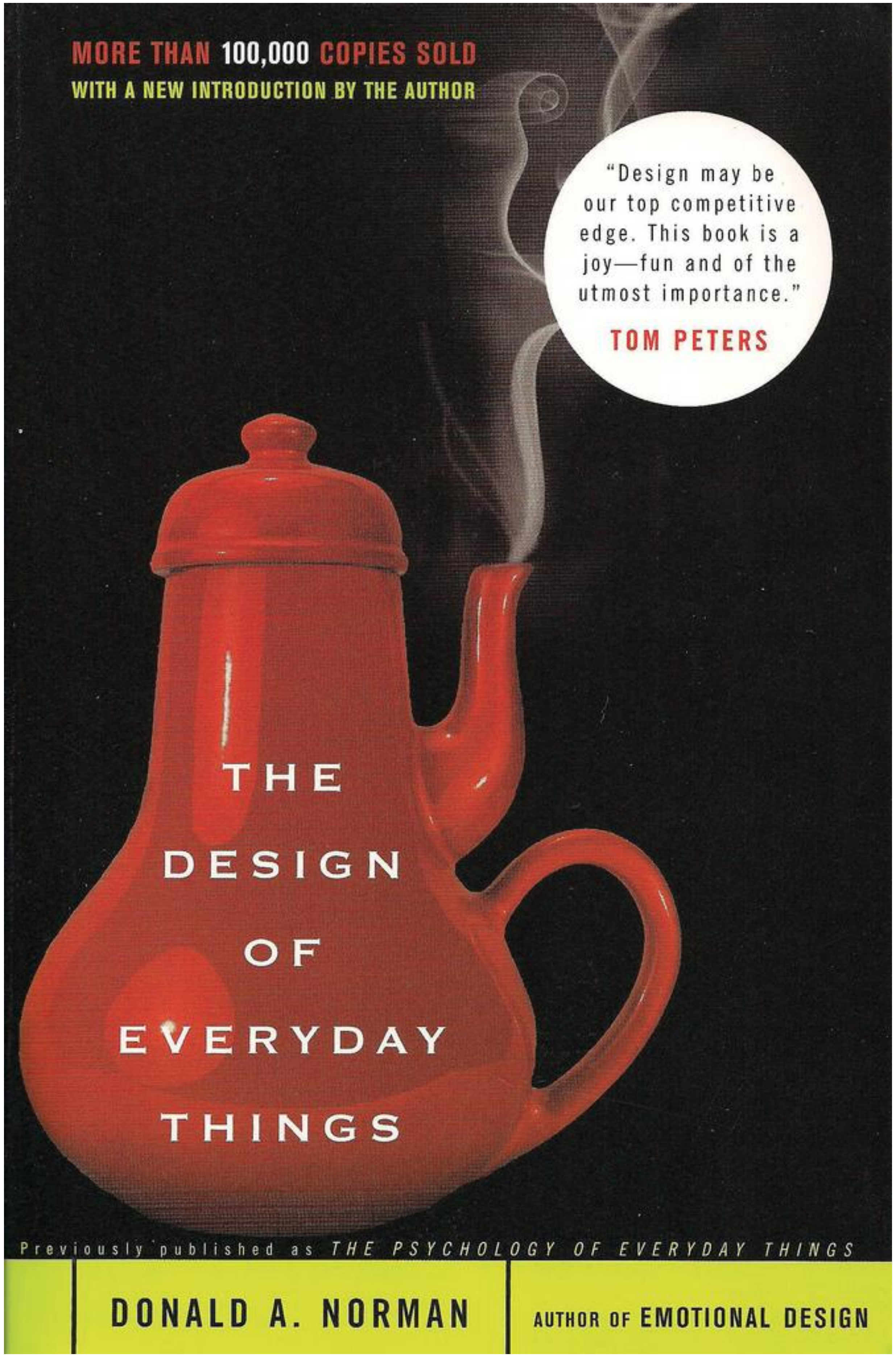
Designing the Alarm Management User Experience for Patient Monitoring

Sharoda A Paul, Alexander K Carroll, Stephen Treacy

LL21C Papers: Computer Supported Parenting

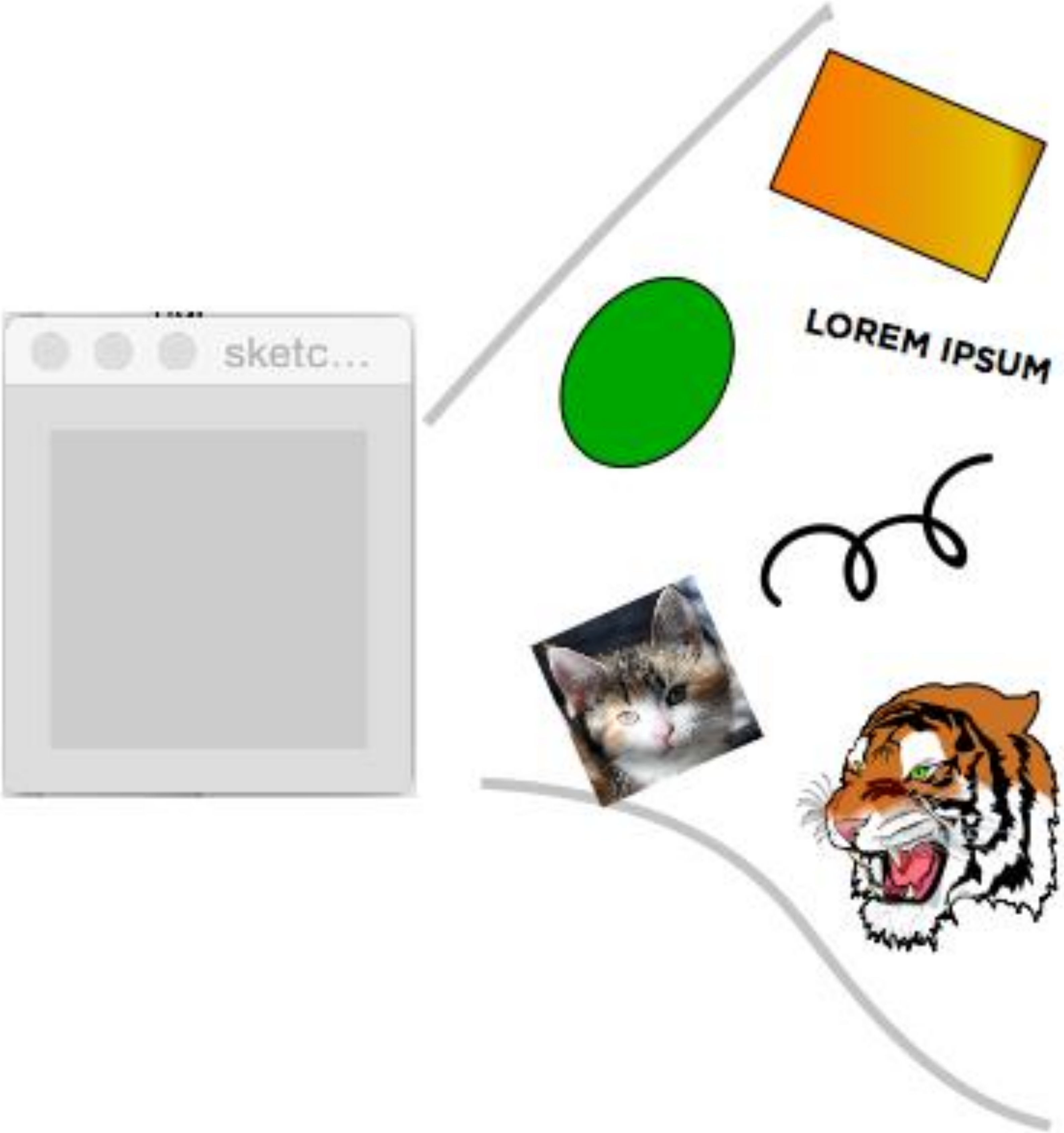
• Cash Withdrawals
• Deposits
• Bill Payments
• Transfers
• Balances
• Account Activities
• Bankbook Updates
CIBC

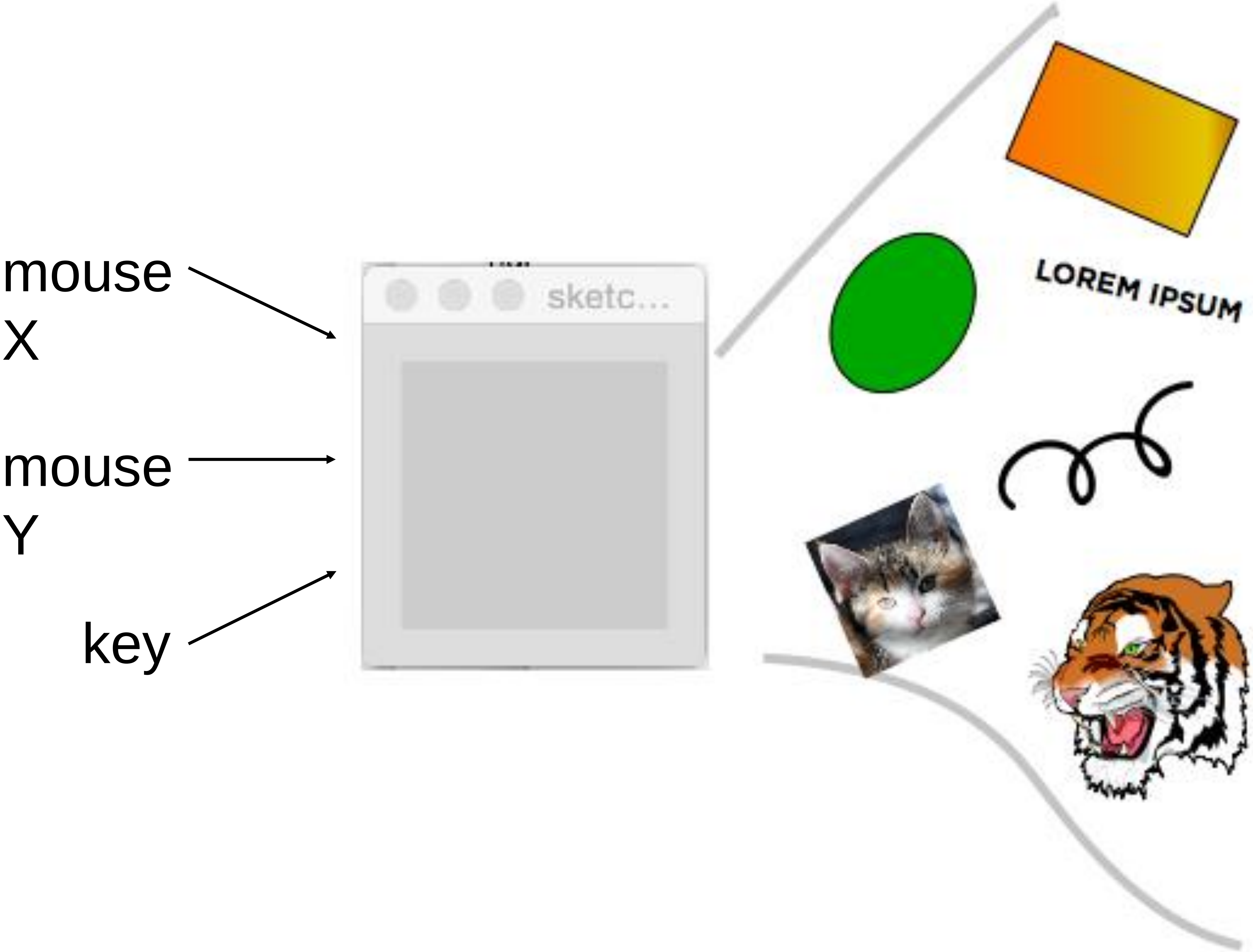
3G
9:41 AM
Monday
7
Messages
Calendar
Photos
Camera
YouTube
Stocks
Maps
280
73°
Weather
Notes
Utilities
iTunes
App Store
Settings
Phone
Mail
Safari
iPod


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?


sketc...
LOREM IPSUM



mouse X

mouse Y

key

Topics

Model-view-controller paradigm

Direct manipulation

User interface toolkits

Building interfaces with ControlP5

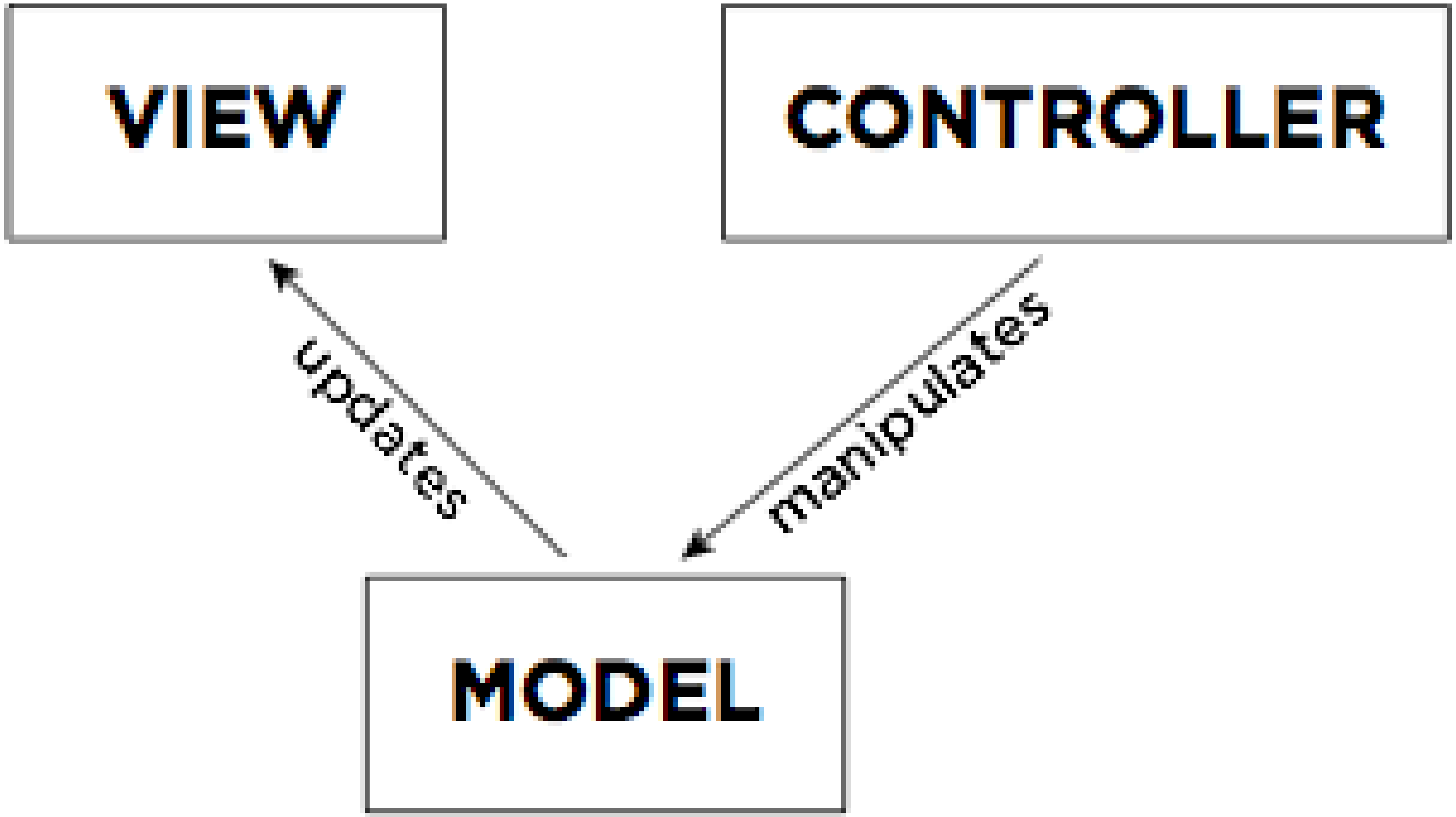
Model-View-Controller (MVC)

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.



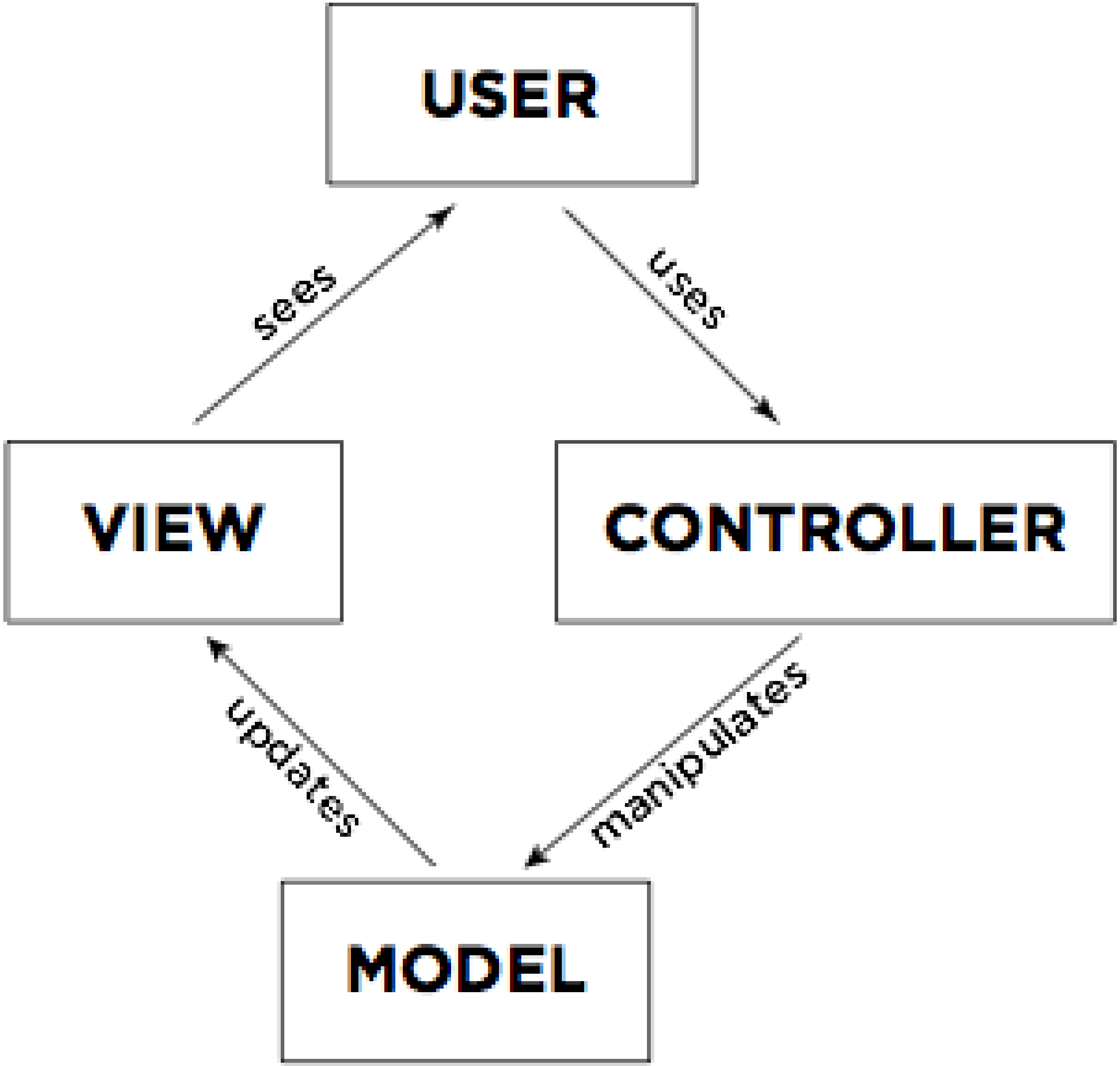
VIEW

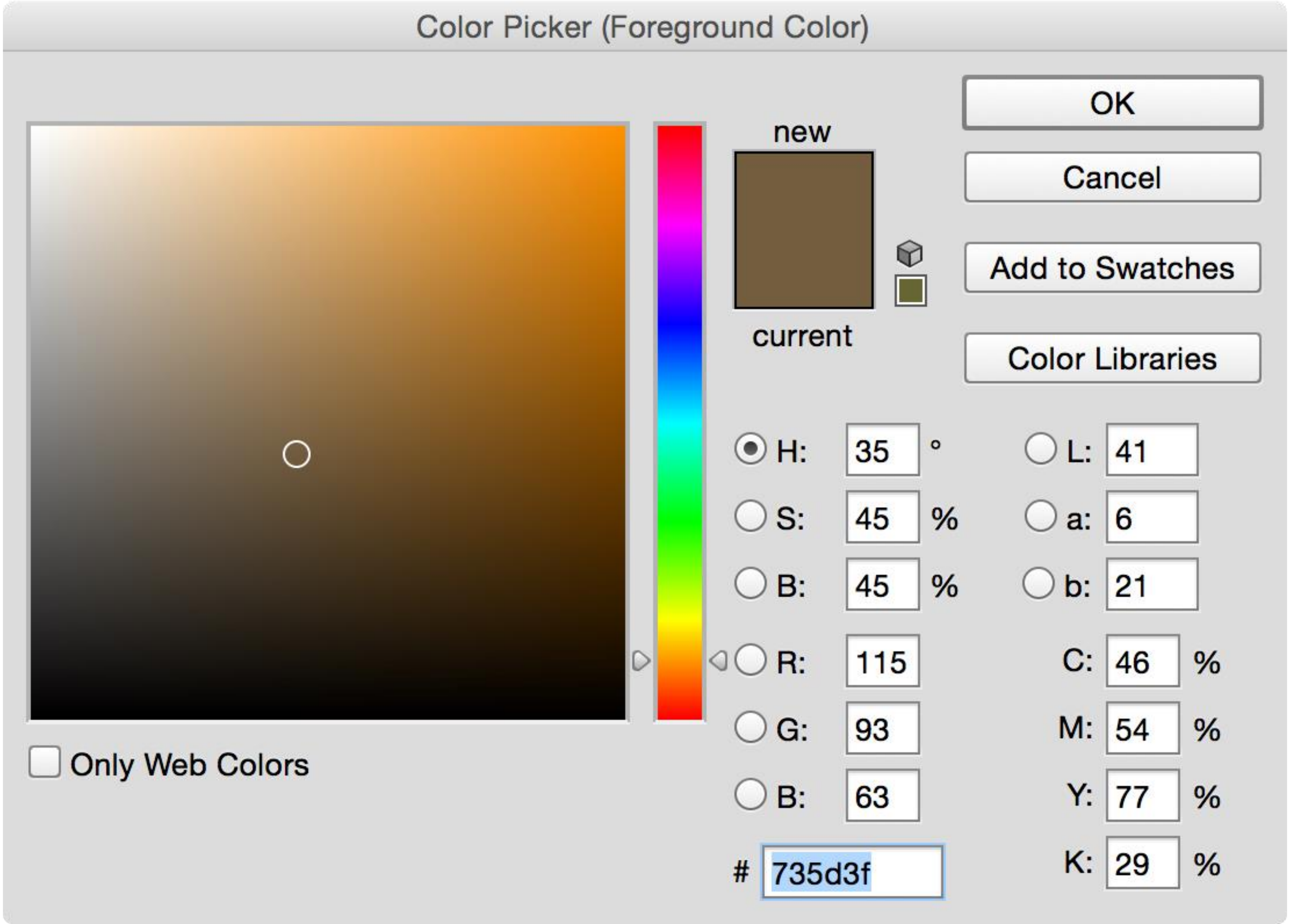
CONTROLLER

updates

manipulates

MODEL

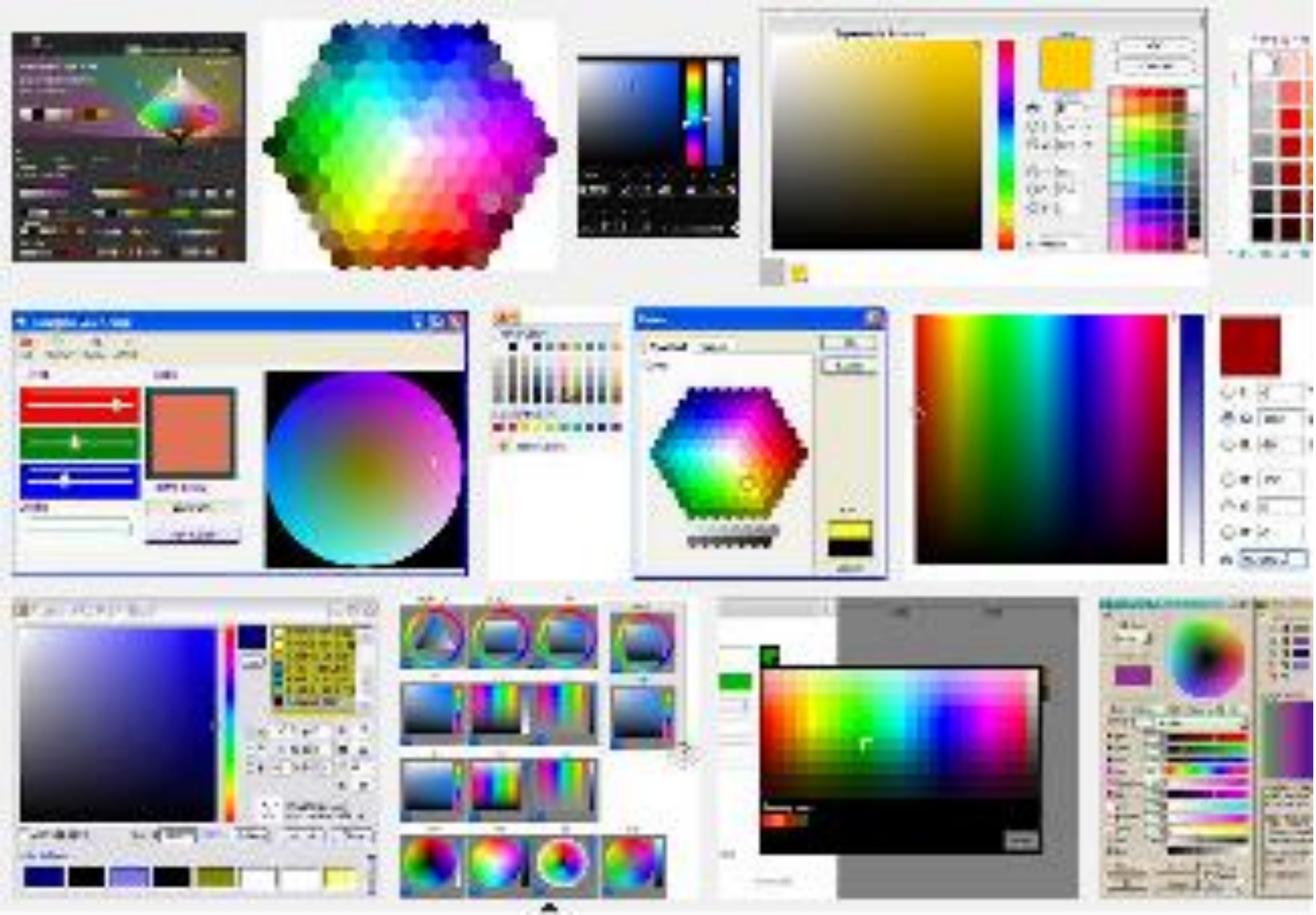

USER
sees
uses
VIEW
CONTROLLER
updates
manipulates
MODEL



Color Picker (Foreground Color)

● H: 35 °
○ S: 45 %
○ B: 45 %
○ R: 115
○ G: 93
○ B: 63
○ L: 41
○ a: 6
○ b: 21
C: 46 %
M: 54 %
Y: 77 %
K: 29 %
735d3f

□ Only Web Colors



```
color the_colour;
```

Model

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

View

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

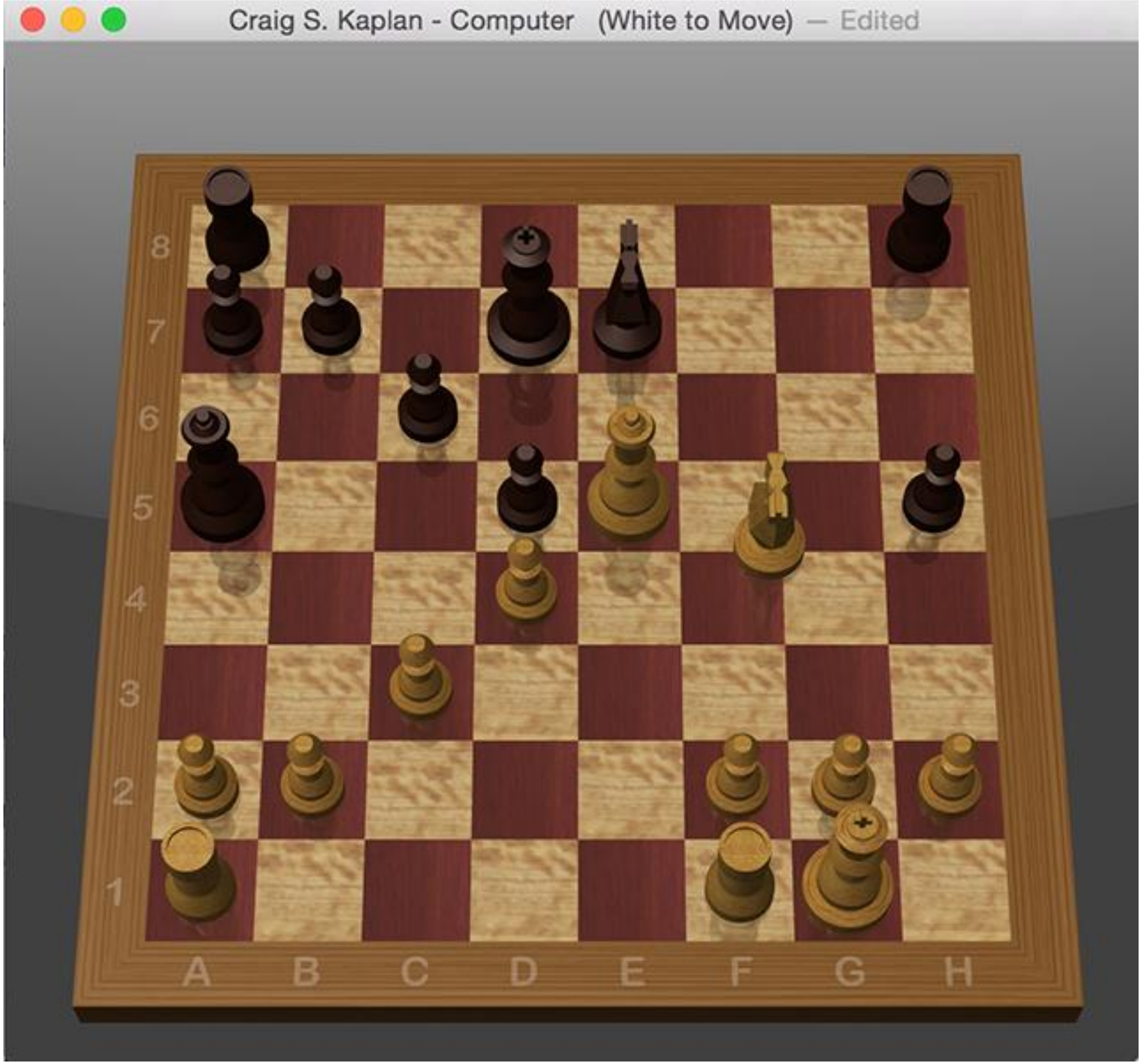
Controller

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

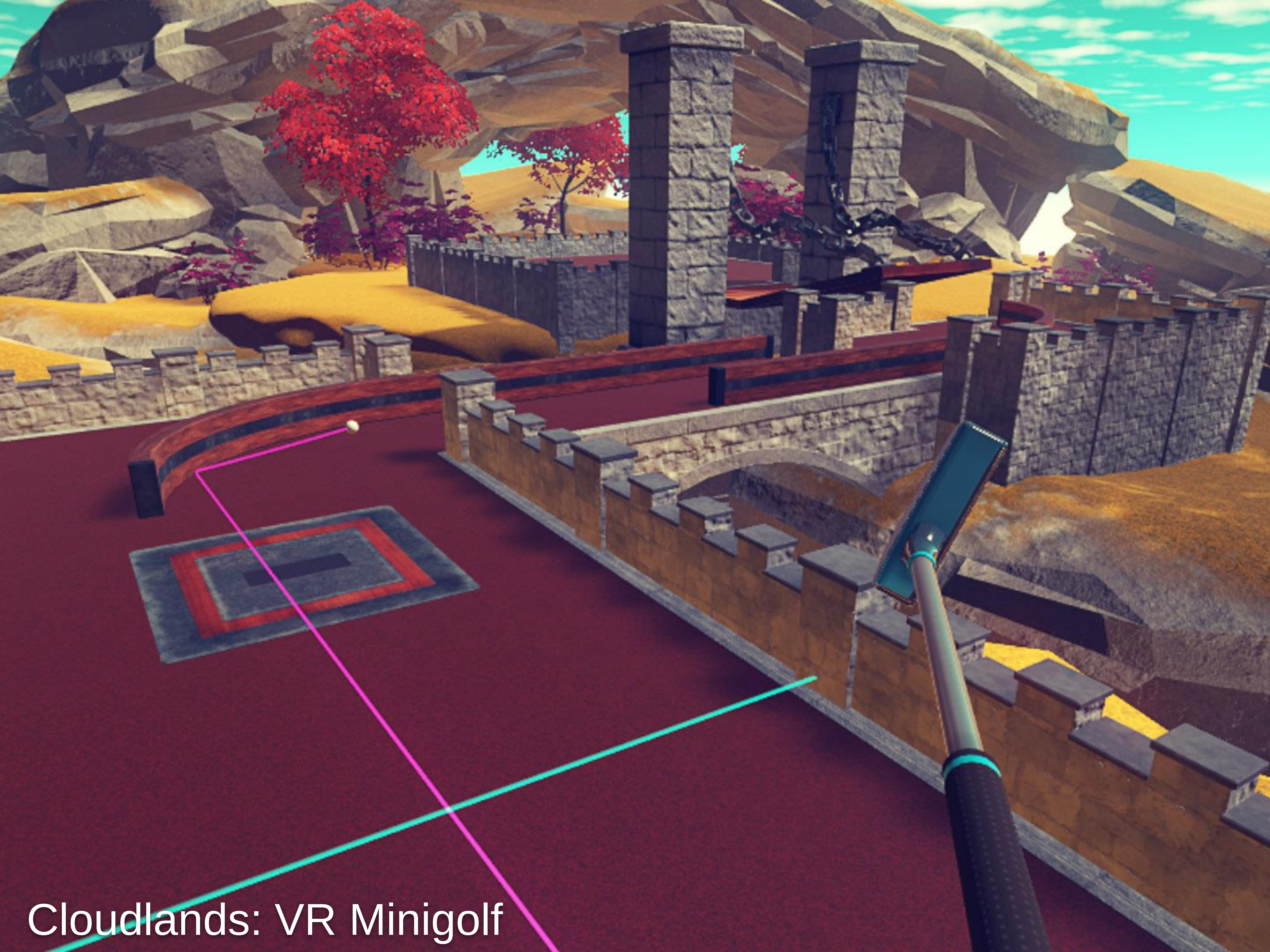
Direct Manipulation

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

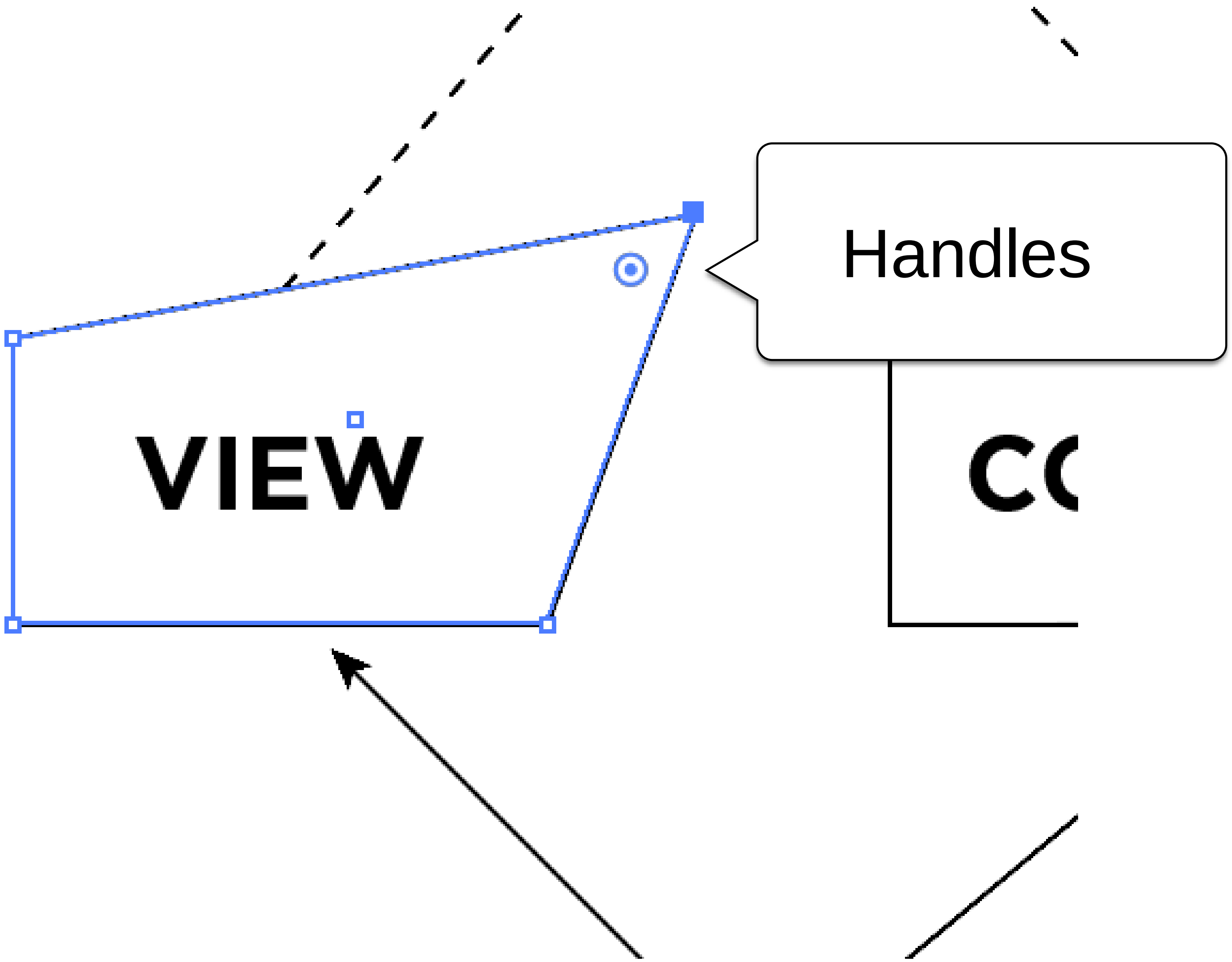
Interaction is continuous and incremental.



Craig S. Kaplan - Computer (White to Move) — Edited



Cloudlands: VR Minigolf



Handles

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

```
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

```
boolean active;
```

Are we dragging the circle?

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

Hit test

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

Controller updates the model

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

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.

```
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;
  }
}
```

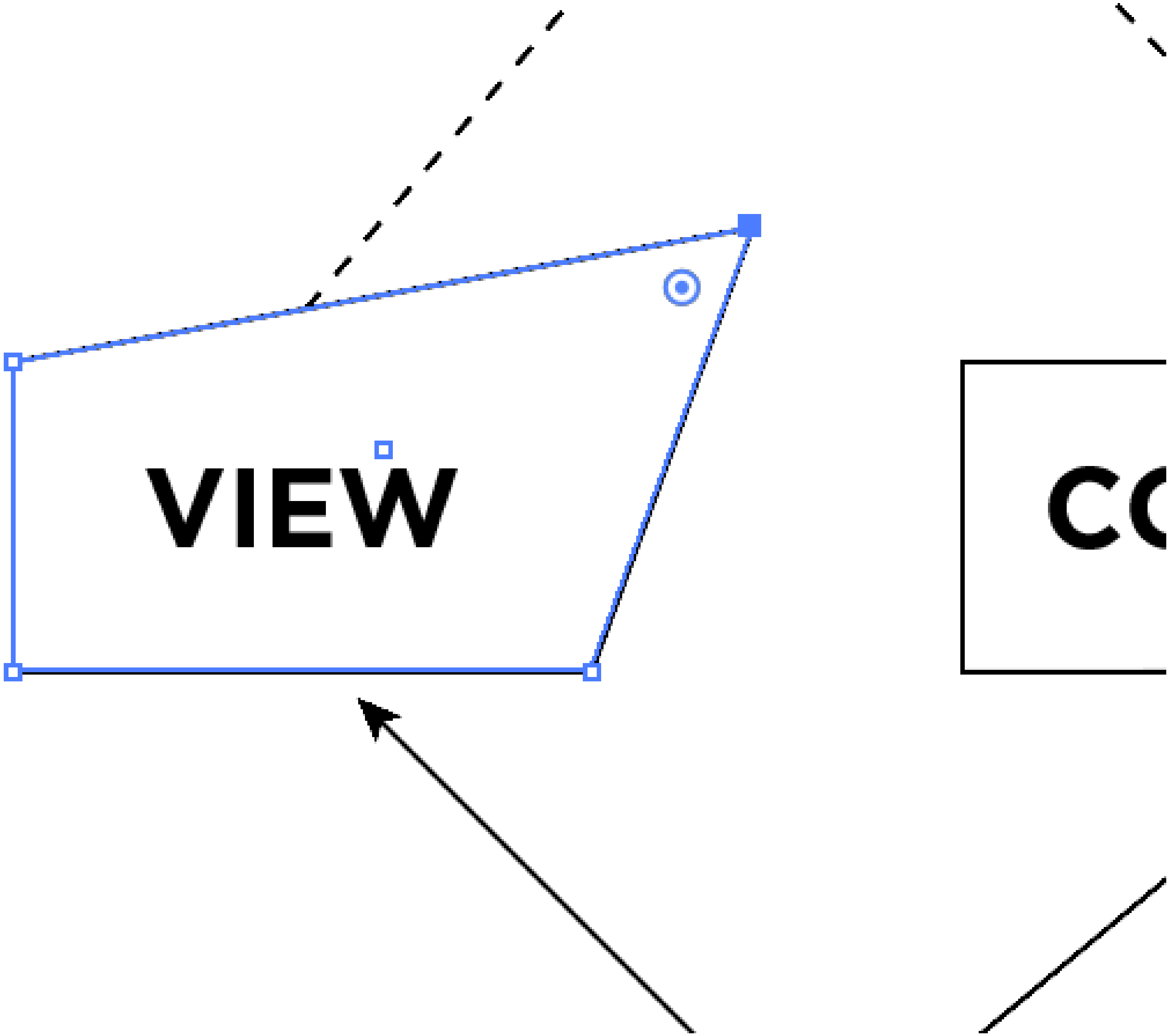
If we have an interface with multiple elements, we need a way to keep track of which one was hit.

```
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;
  }
}
```

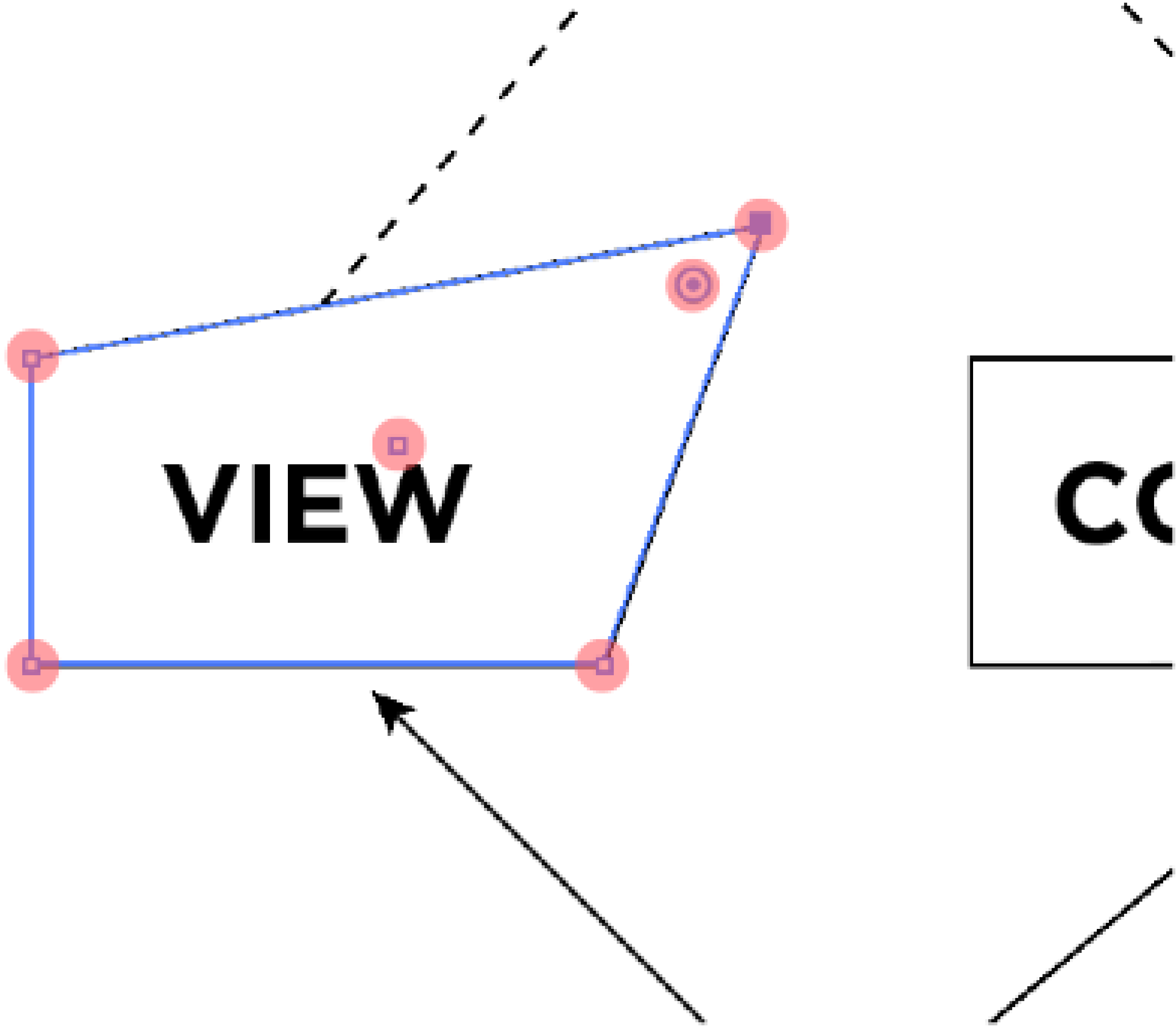
If we have an interface with multiple elements, we need a way to keep track of which one was hit.

```
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;
    }
  }
}
```

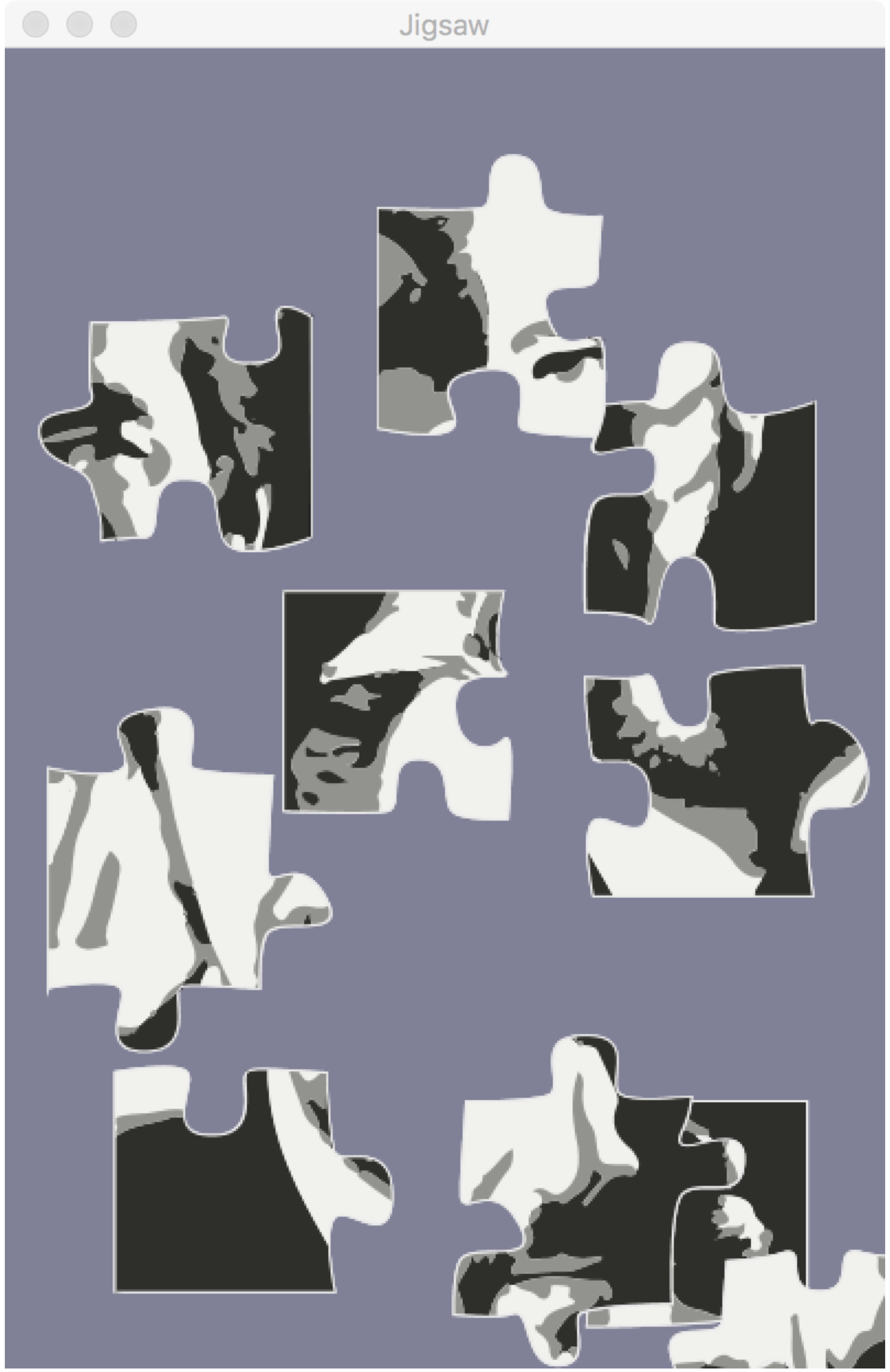
Small handles



Small handles

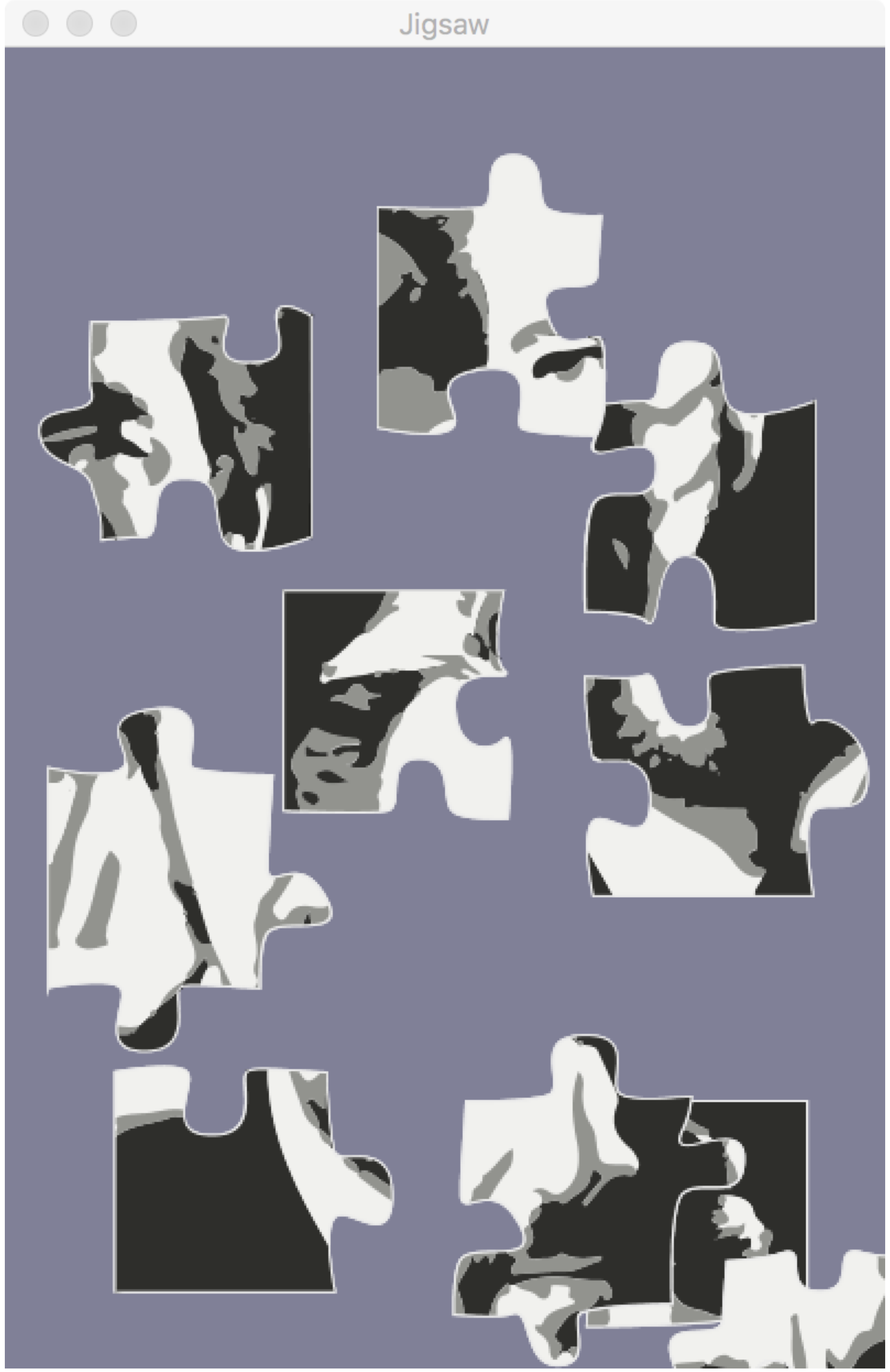


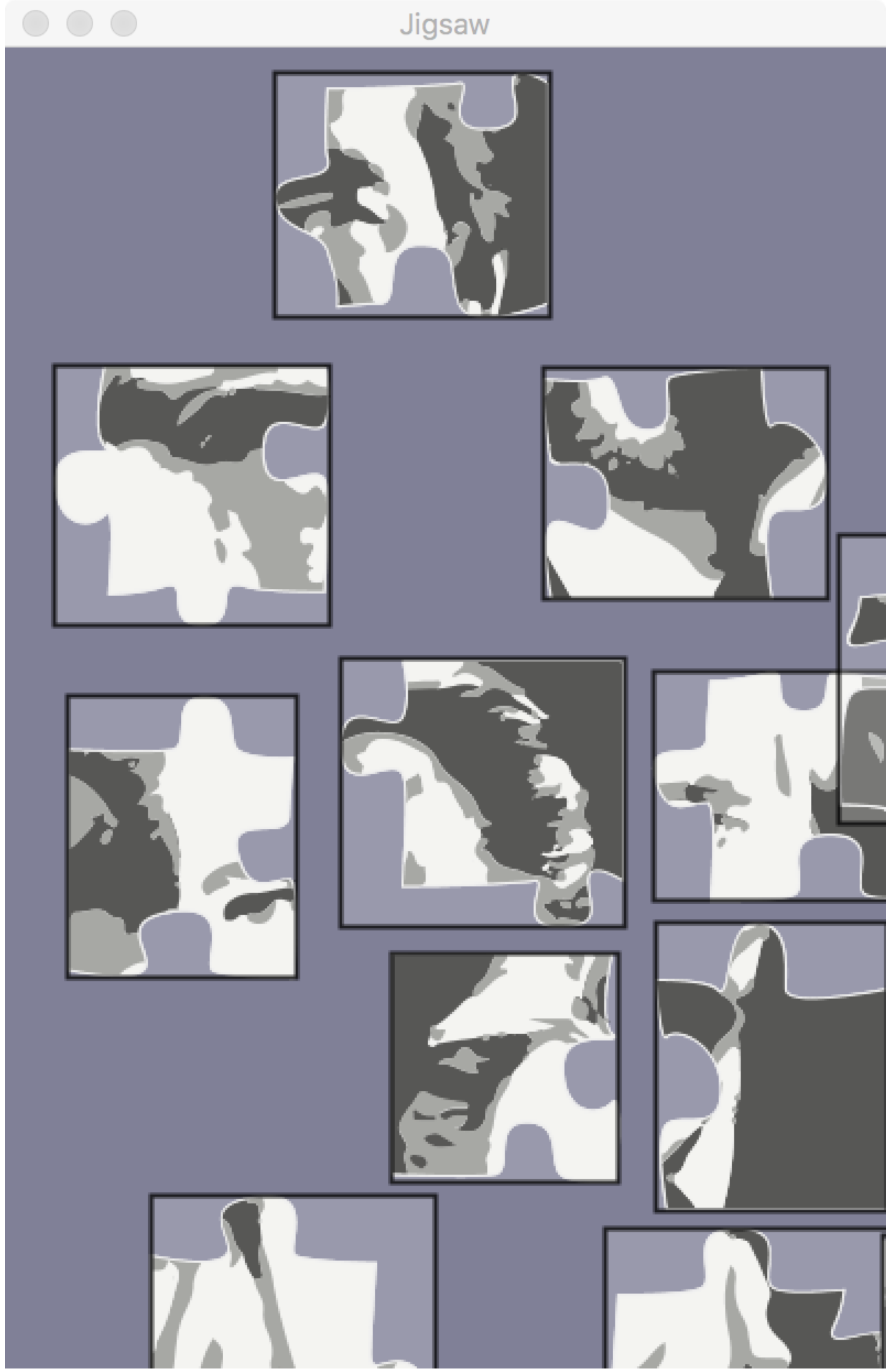
Complex shapes



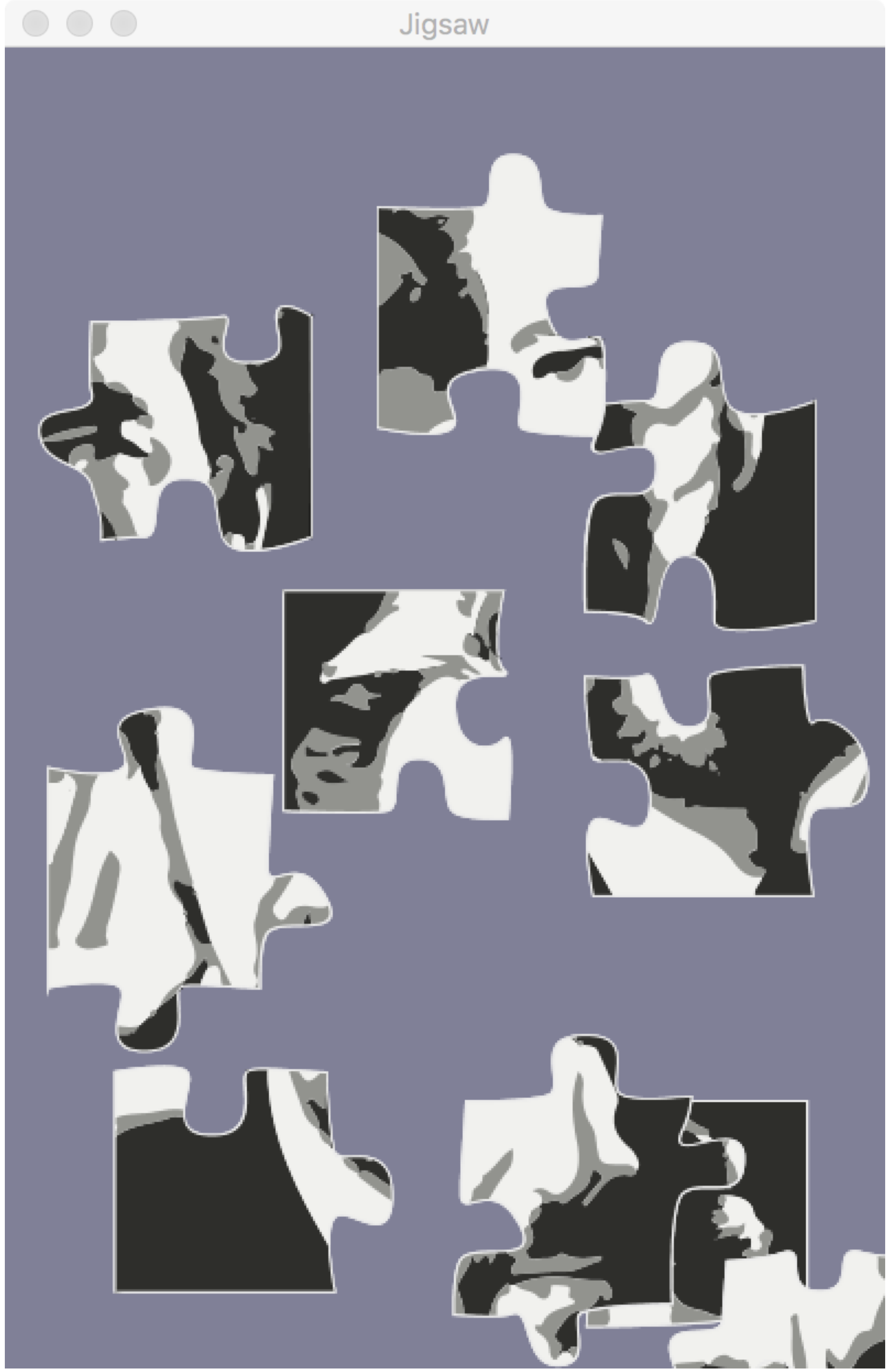
How can we hit test a shape with a complicated boundary?

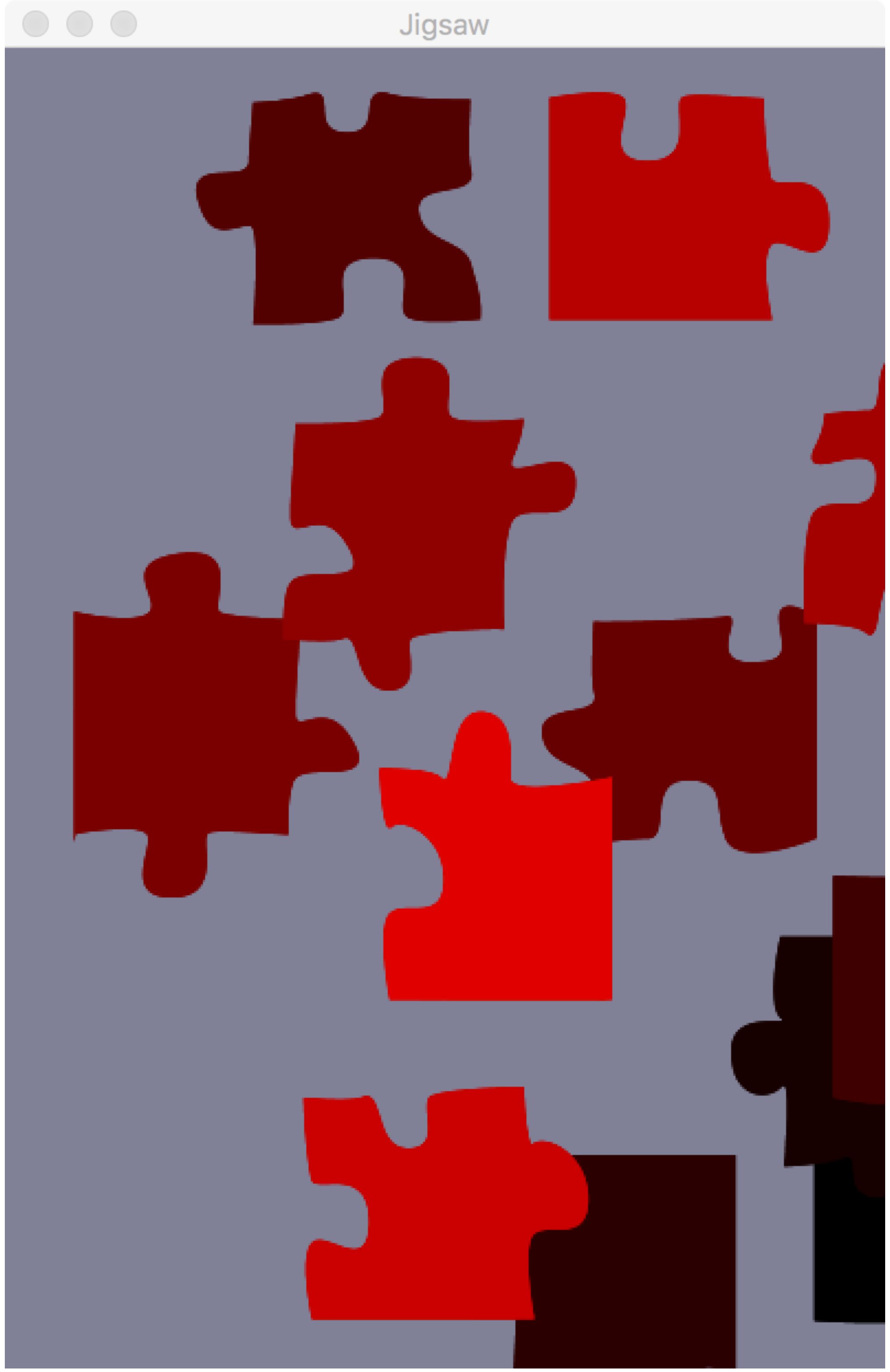
Proxy geometry





Selection buffer





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.

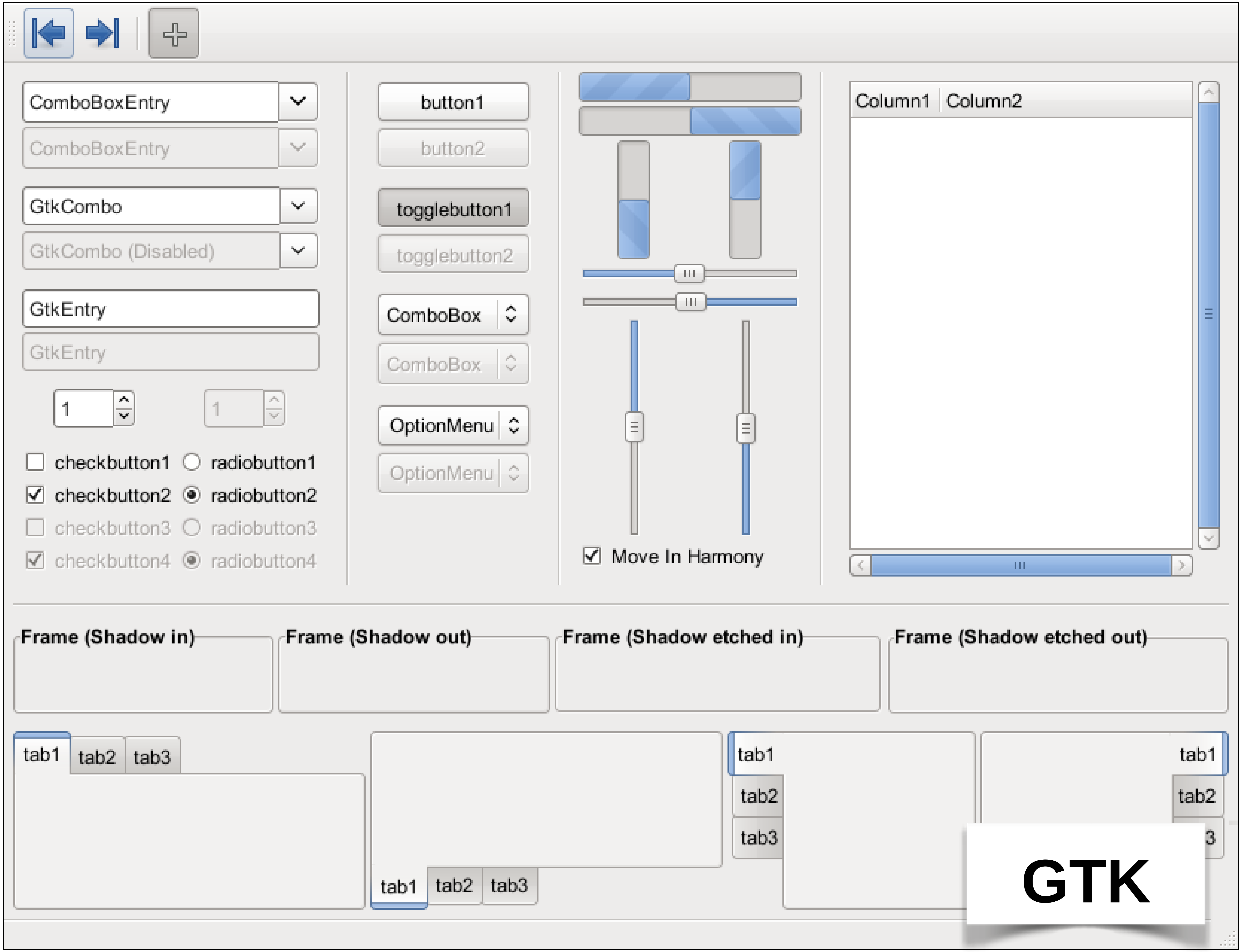
Make hit test region usable, regardless of how it’s drawn.

Toolkits

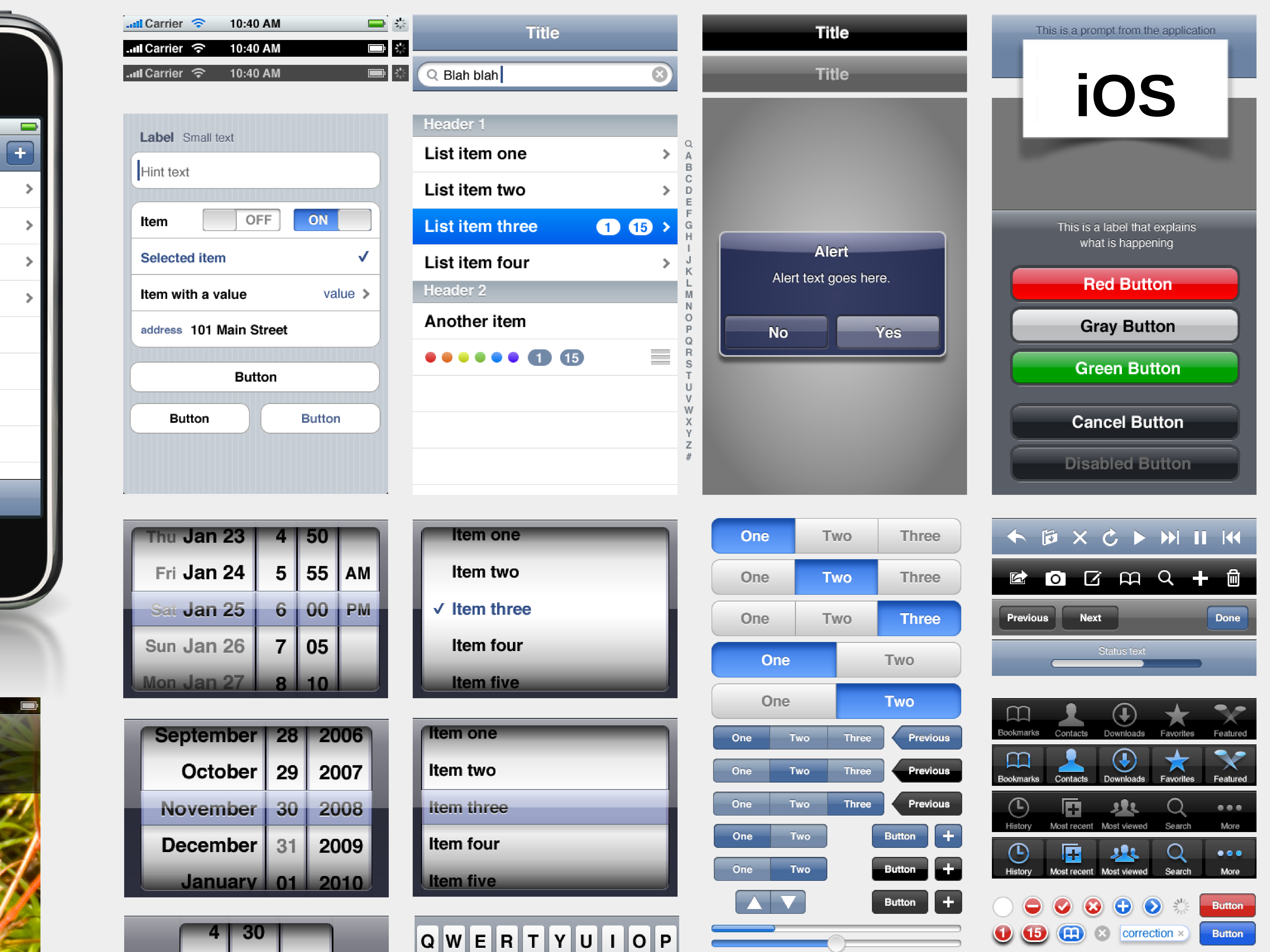
Some interactions are so canonical that it makes sense to invent standardized “widgets” to handle them.

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

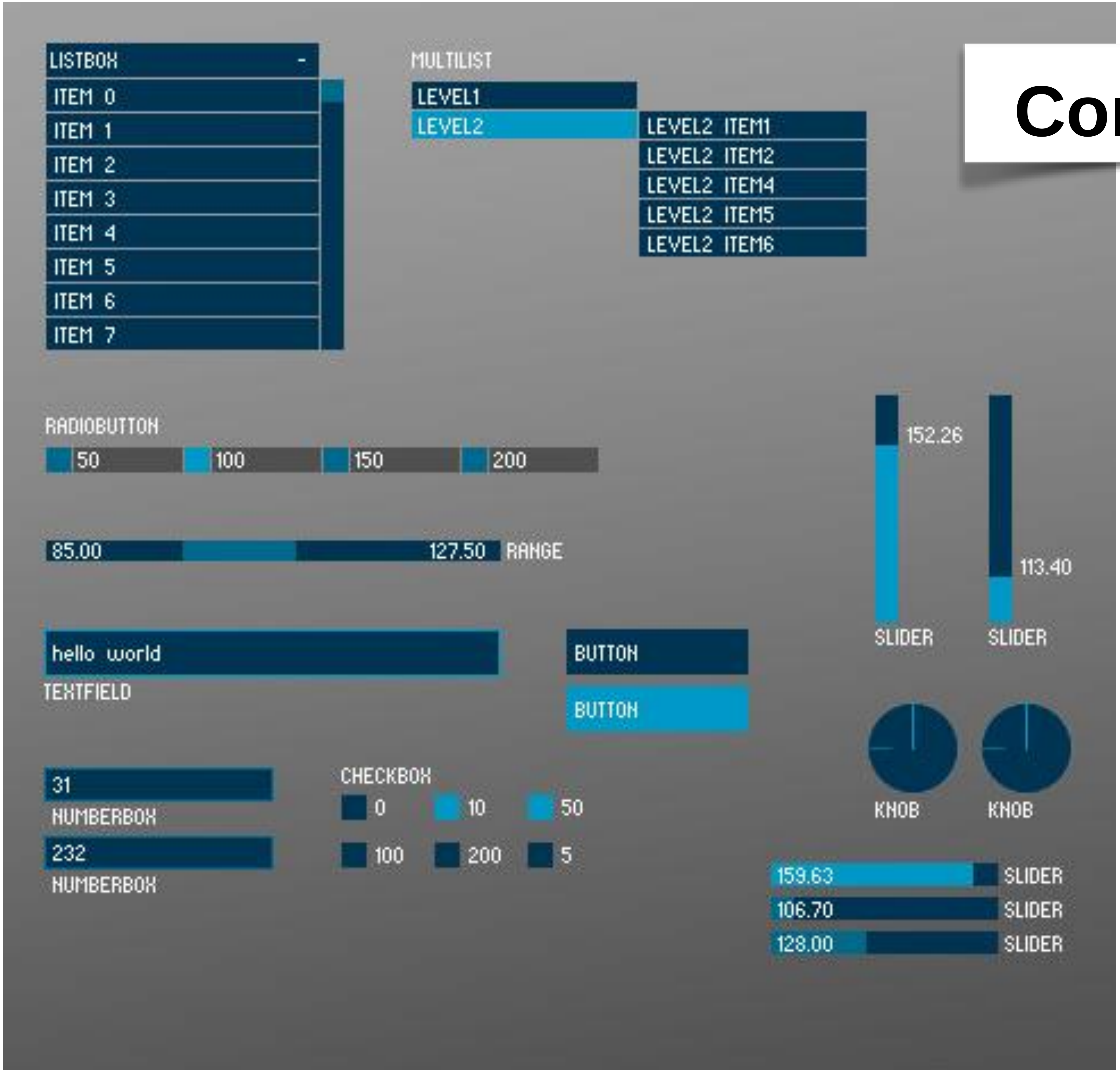
Classes and objects are perfect for this!


ComboBoxEntry
ComboBoxEntry
GtkCombo
GtkCombo (Disabled)
GtkEntry
GtkEntry
1
1
□ checkbutton1
☑ checkbutton2
□ checkbutton3
☑ checkbutton4
○ radiobutton1
● radiobutton2
○ radiobutton3
● radiobutton4
button1
button2
togglebutton1
togglebutton2
ComboBox
ComboBox
OptionMenu
OptionMenu
☑ Move In Harmony
Column1
Column2
Frame (Shadow in)
Frame (Shadow out)
Frame (Shadow etched in)
Frame (Shadow etched out)
tab1
tab2
tab3
tab1
tab2
tab3
tab1
tab2
tab3
tab1
tab2
3

GTK


Carrier 10:40 AM
Carrier 10:40 AM
Carrier 10:40 AM
Title
Blah blah
Title
Title
This is a prompt from the application
iOS
Label Small text
Hint text
Item OFF ON
Selected item ✓
Item with a value value
address 101 Main Street
Button
Button
Button
Header 1
List item one
List item two
List item three 1 15
List item four
Header 2
Another item
1 15
Q A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #
Alert
Alert text goes here.
No
Yes
This is a label that explains what is happening
Red Button
Gray Button
Green Button
Cancel Button
Disabled Button
Thu Jan 23 4 50
Fri Jan 24 5 55 AM
Sat Jan 25 6 00 PM
Sun Jan 26 7 05
Mon Jan 27 8 10
Item one
Item two
✓ Item three
Item four
Item five
One Two Three
One Two Three
One Two Three
One Two
One Two
Previous Next Done
Status text
September 28 2006
October 29 2007
November 30 2008
December 31 2009
January 01 2010
Item one
Item two
Item three
Item four
Item five
One Two Three Previous
One Two Three Previous
One Two Three Previous
One Two Button +
One Two Button +
Button +
Bookmarks Contacts Downloads Favorites Featured
Bookmarks Contacts Downloads Favorites Featured
History Most recent Most viewed Search More
History Most recent Most viewed Search More
Button
1 15 correction ×
Button
4 30
Q W E R T Y U I O P

ControlP5


LISTBOX
-
ITEM 0
ITEM 1
ITEM 2
ITEM 3
ITEM 4
ITEM 5
ITEM 6
ITEM 7
MULTILIST
LEVEL1
LEVEL2
LEVEL2 ITEM1
LEVEL2 ITEM2
LEVEL2 ITEM4
LEVEL2 ITEM5
LEVEL2 ITEM6
RADIOBUTTON
50
100
150
200
85.00
127.50
RANGE
hello world
TEXTFIELD
BUTTON
BUTTON
31
NUMBERBOX
232
NUMBERBOX
CHECKBOX
0
10
50
100
200
5
152.26
113.40
SLIDER
SLIDER
KNOB
KNOB
159.63
SLIDER
106.70
SLIDER
128.00
SLIDER

Minimal ControlP5

```
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

```
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

```
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 );
}
```

```
{
 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!" );
}
```

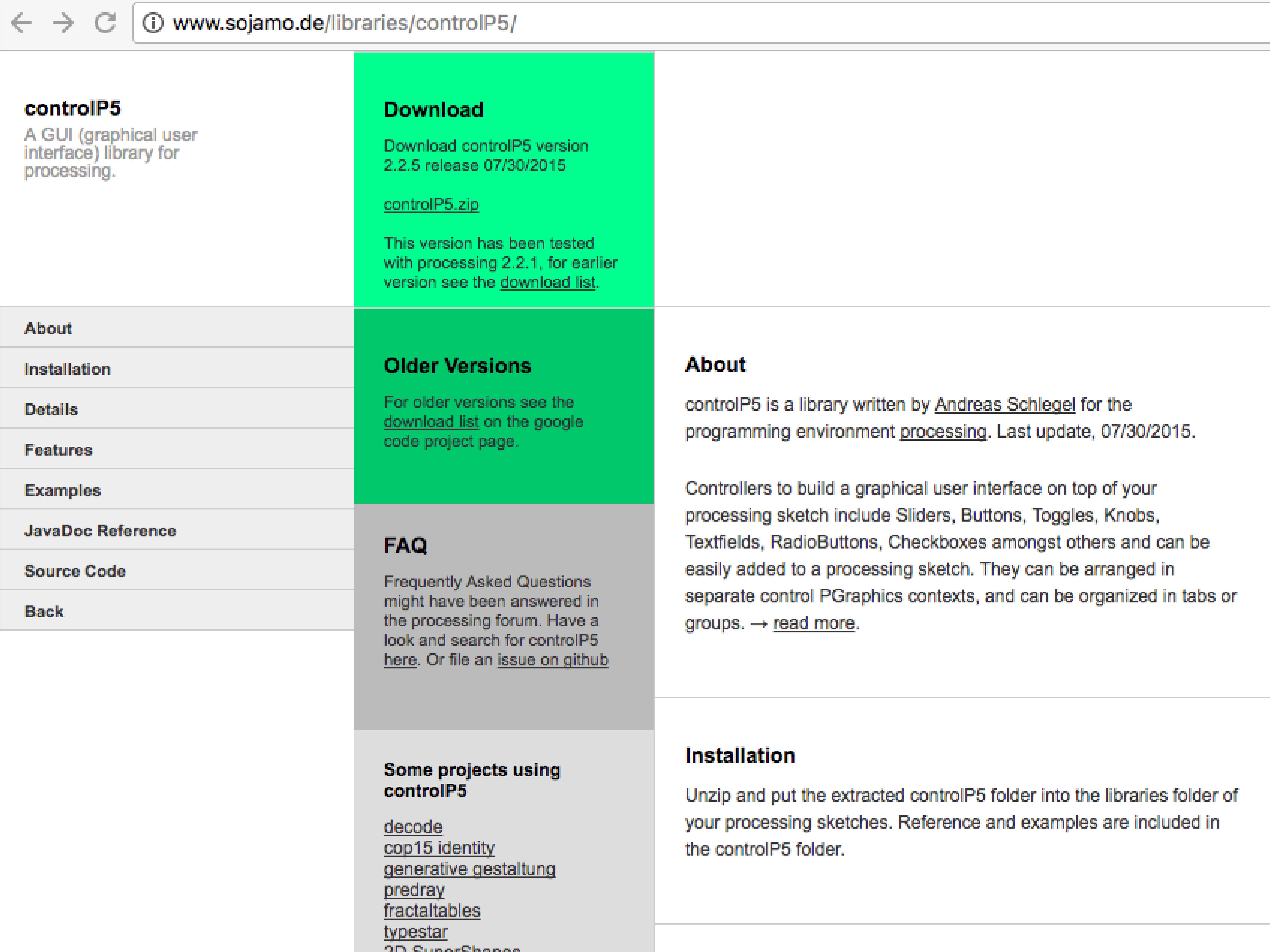
```
void setup()
{
  size( 500, 500 );

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

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

Information about the event

```
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" );
  }
}
```



www.sojamo.de/libraries/controlP5/

controlP5

A GUI (graphical user interface) library for processing.

- About
- Installation
- Details
- Features
- Examples
- JavaDoc Reference
- Source Code
- Back

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.

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
cop15 identity
generative gestaltung
predray
fractaltables
typestar
2D SuperShapes

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.

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

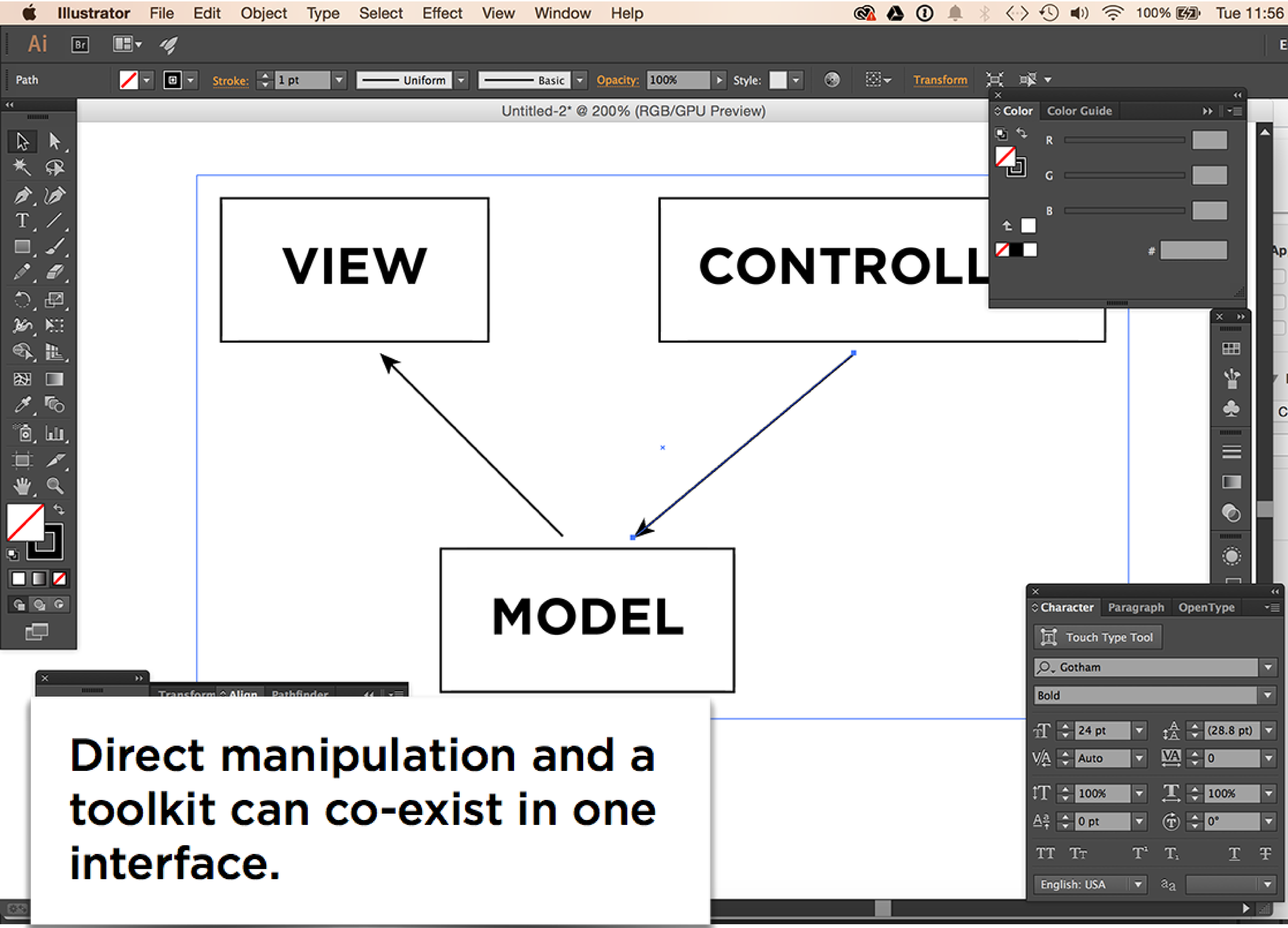
controllers/ControlP5accordion

controllers/ControlP5bang

controllers/ControlP5button

controllers/ControlP5canvas

controllers/ControlP5chart



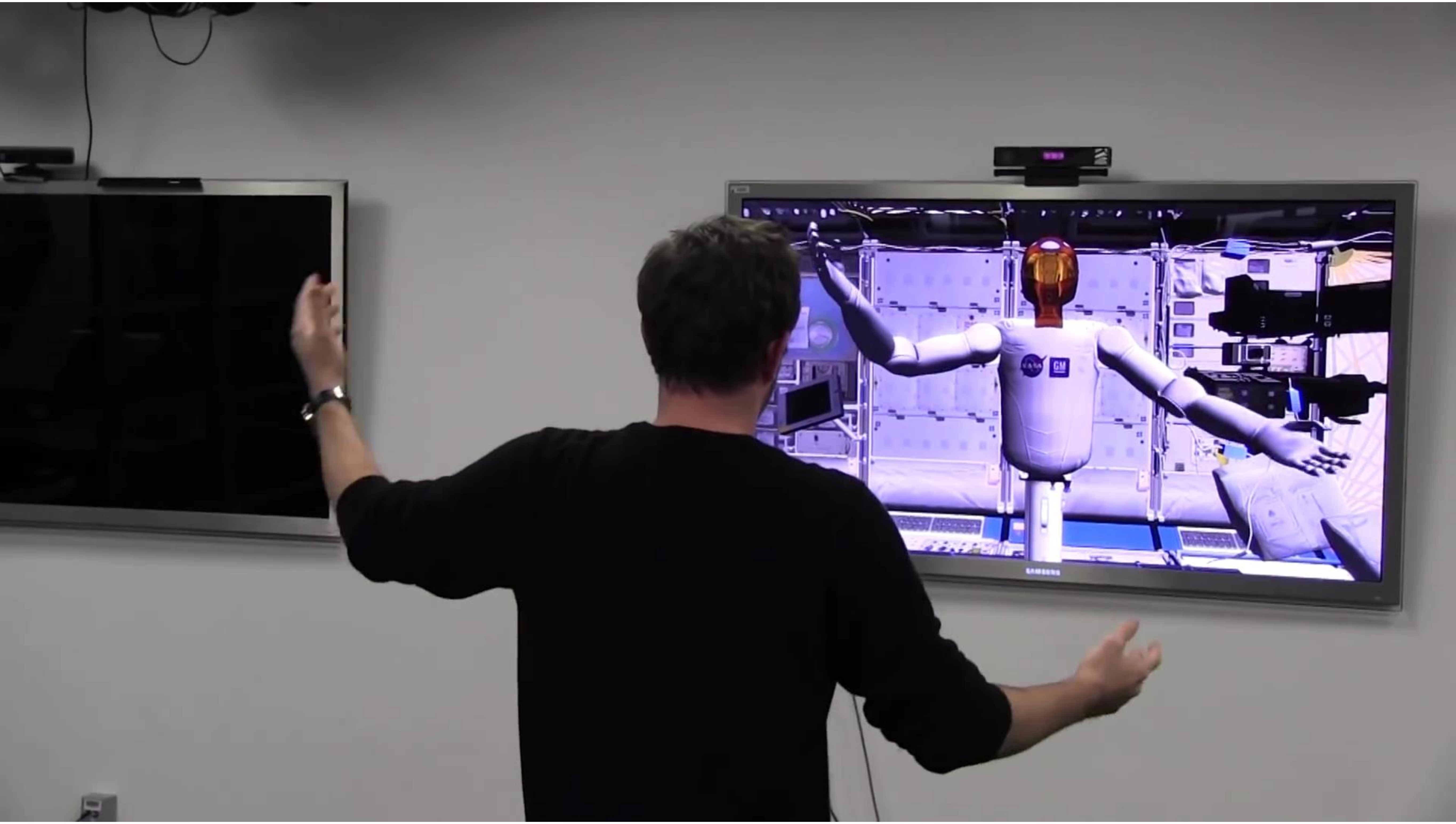
Path | Stroke: 1 pt | Uniform | Basic | Opacity: 100% | Style:

Untitled-2* @ 200% (RGB/GPU Preview)

Font: Gotham
Style: Bold
Font size: 24 pt
Leading: (28.8 pt)
Kerning: Auto
Tracking: 0
Vertical scale: 100%
Horizontal scale: 100%
Baseline shift: 0 pt
Character rotation: 0°
Language: English: USA

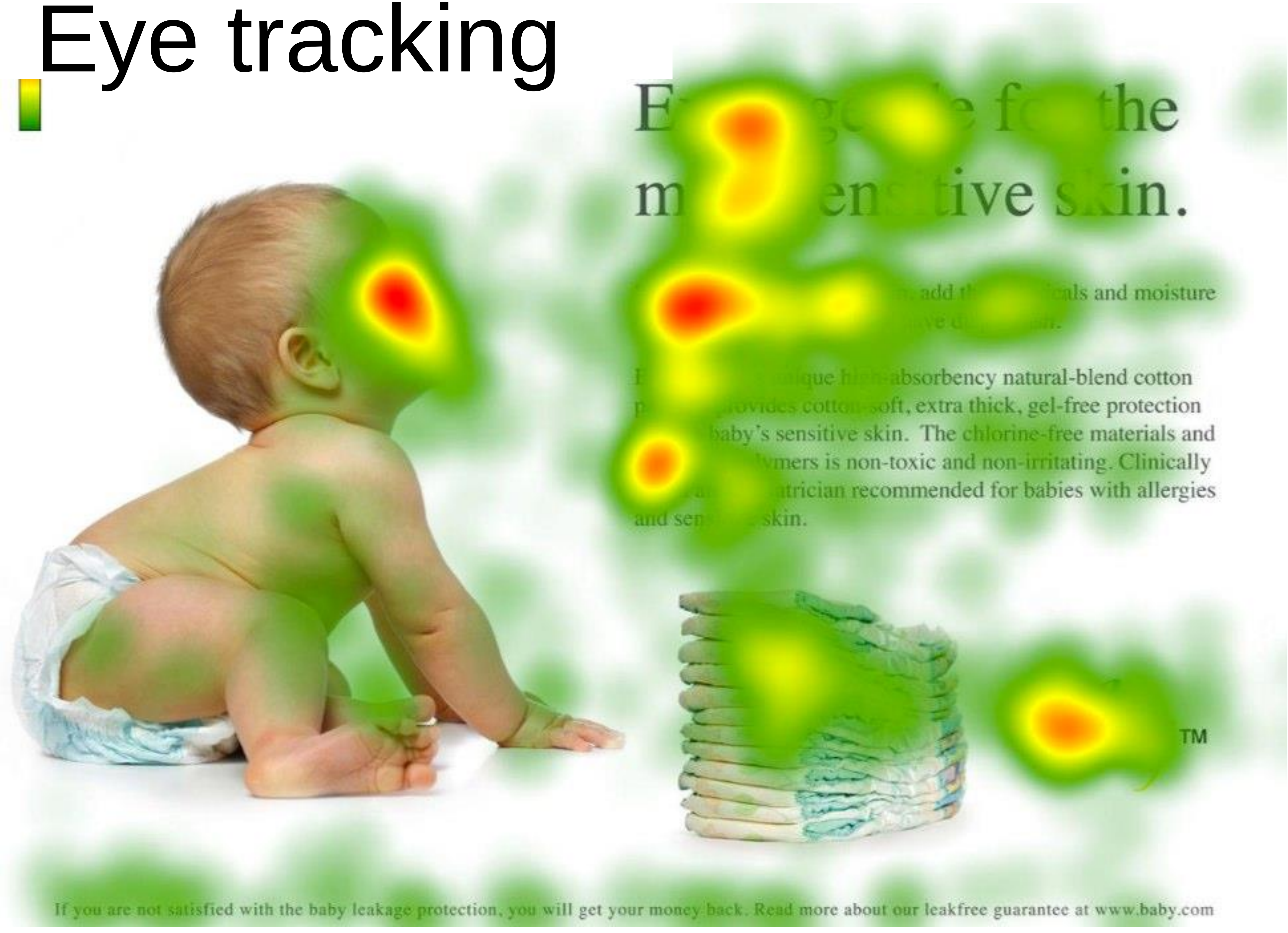
Direct manipulation and a toolkit can co-exist in one interface.

Kinect

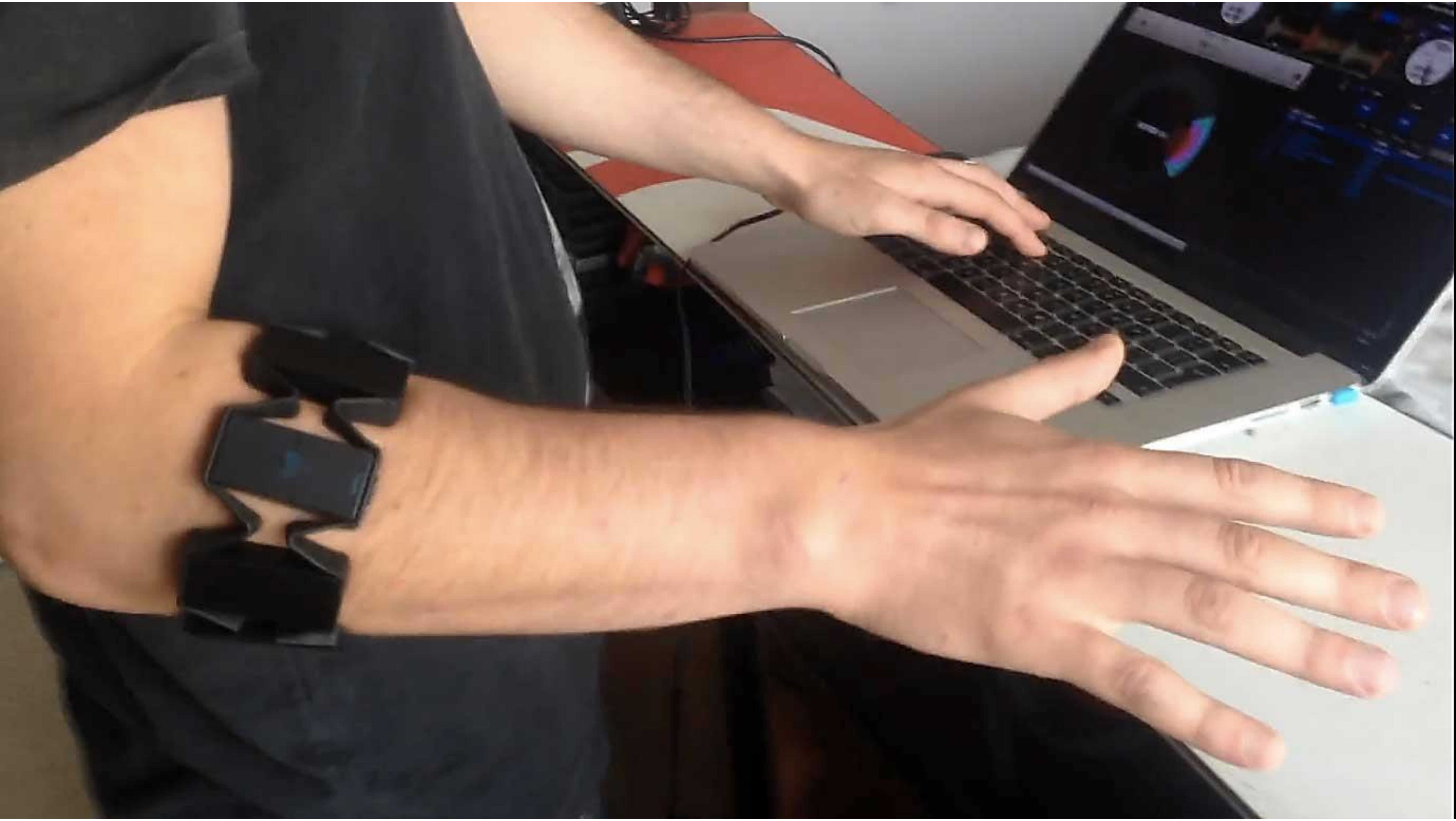


NASA / Jet propulsion laboratory

Eye tracking


E ge e f the
m ens tive s in.
add th icals and moisture
ve d
ique h absorbency natural-blend cotton
p rovides cotton-soft, extra thick, gel-free protection
baby's sensitive skin. The chlorine-free materials and
ymers is non-toxic and non-irritating. Clinically
trician recommended for babies with allergies
and sen skin.
TM
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



Thalmic Labs

On-world interfaces