Leibniz Universität Hannover

Mensch-Computer-Interaktion 2

Interaction Elements

Human-Computer Interaction Group

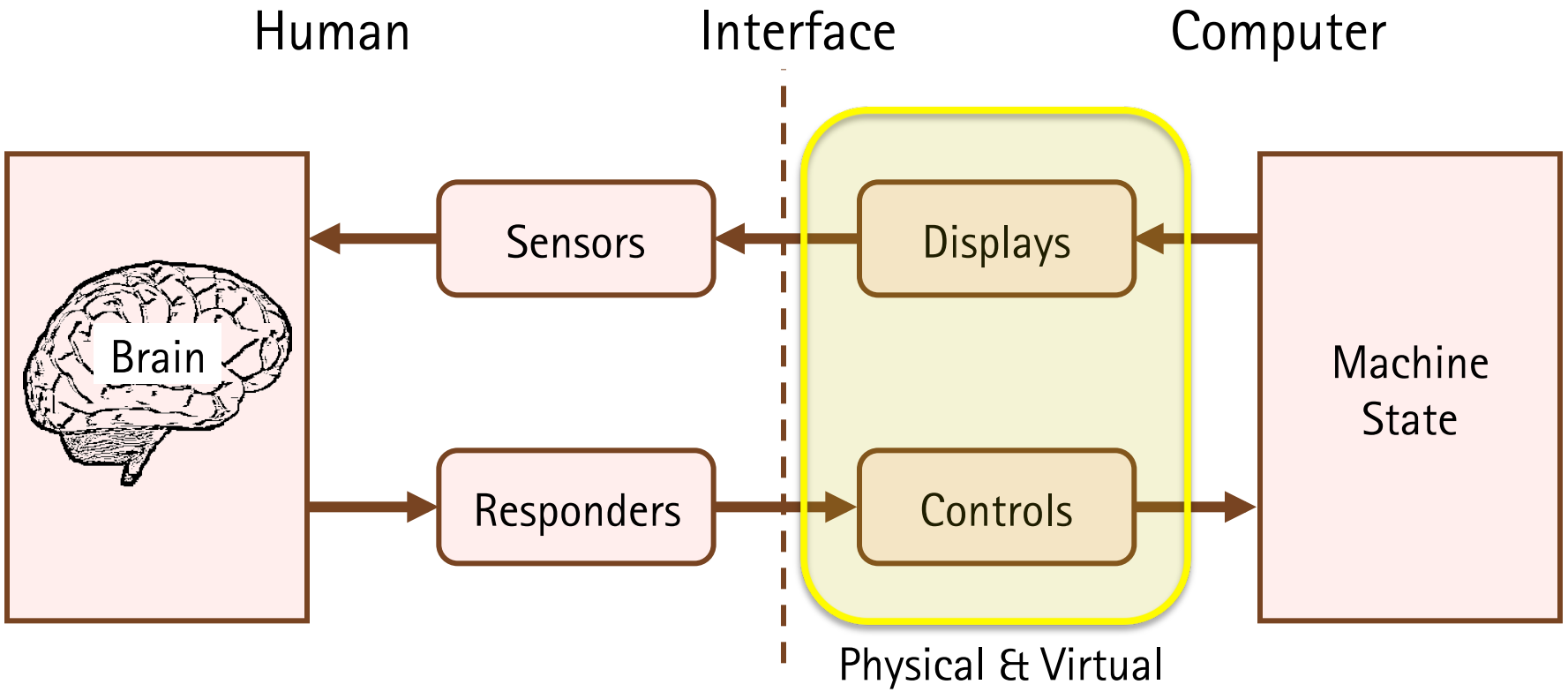
Prof. Dr. Michael Rohs
michael.rohs@hci.uni-hannover.de

Lectures

Session	Date	Topic	
1	6.4.	Introduction	
2	13.4.	Interaction elements	GUI toolkits, interaction techniques
3	20.4.	Event handling	
4	27.4.	Scene graphs	
5	4.5.	Interaction techniques	
	11.5.	no class (CHI)	
	18.5.	no class (spring break)	
6	25.5.	Experiments	design and analysis of experiments
7	1.6.	Data Analysis	
8	8.6.	Data Analysis	
9	15.6.	Visualization	
10	22.6.	Visualization	current topics beyond-desktop UIs
11	29.6.	Modeling interaction	
12	6.7.	Computer vision for interaction	
13	13.7.	Computer vision for interaction	

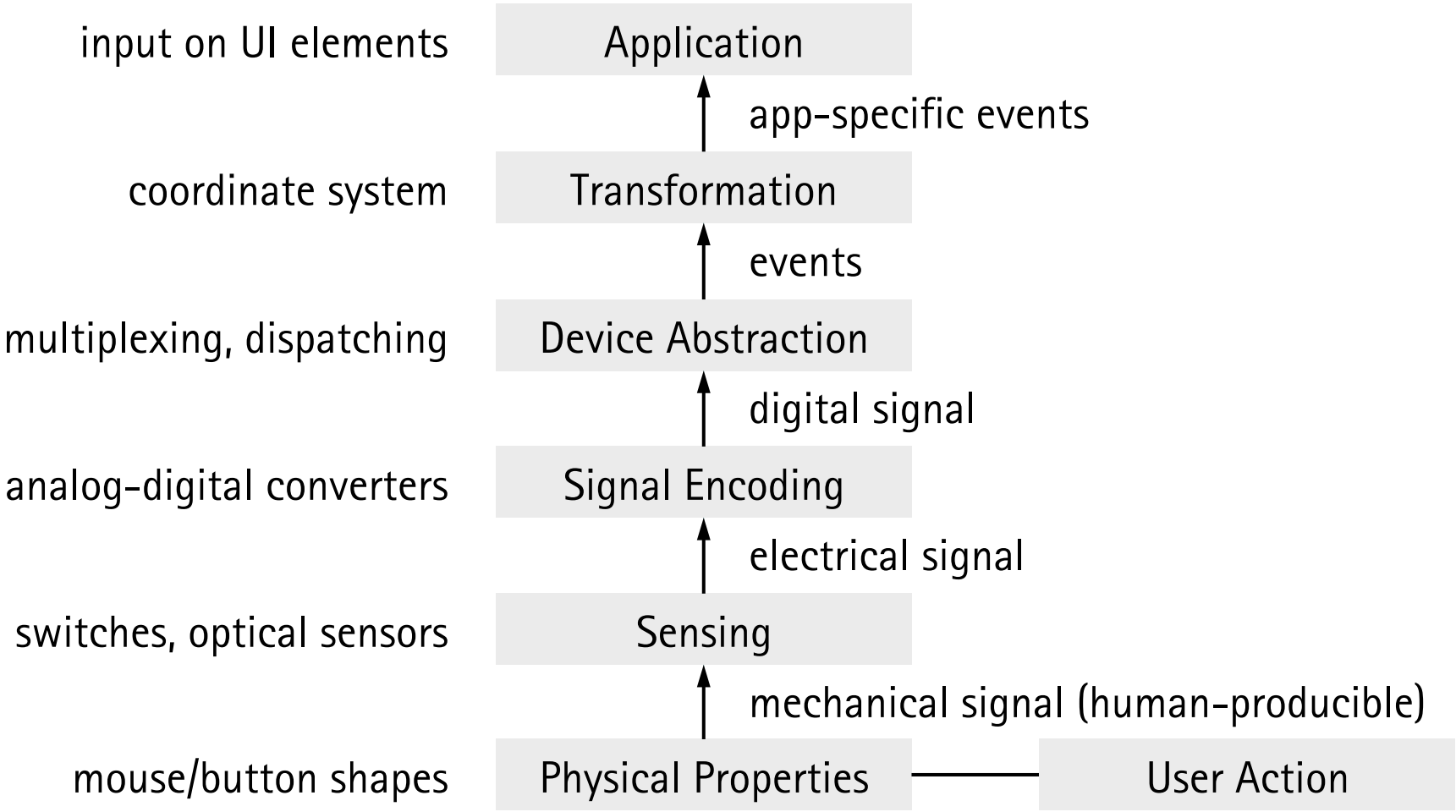
Klausur:
28.7.2016
8-11 Uhr
HG E214

Human Factors Model (revisited)



MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Layers of Input Processing



Interaction Elements

- Fit (physical) input/output devices to human effectors/sensors
  - Shape, behavior
- Link input devices to (virtual) interaction elements
  - Temporary coupling of physical input device to virtual tool
- Apply interaction elements to domain objects
  - Invoke function on domain object to change its state

- Interaction elements (widgets): Components of a user interface (UI) designed to manipulate the state of the UI and of domain objects

Interaction Elements

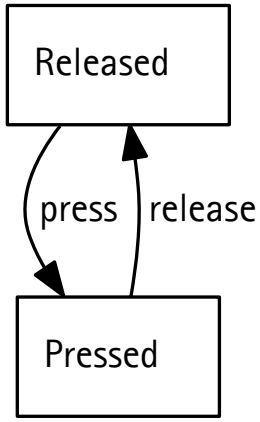
- Interaction elements can be studied at many levels and in different contexts
- Tasks in cognitive band of Newell's time scale of human action
  - Deliberate acts (≈100 ms), operations (≈1 s), unit tasks (≈10 s)
- Tasks in this range are well suited to empirical research
  - Duration of experiments, control of relevant factors

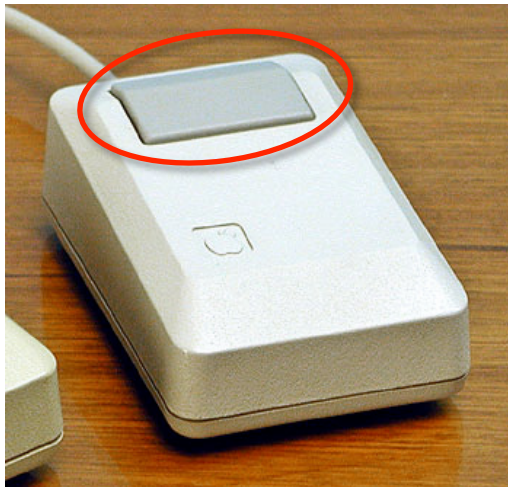
MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

INPUT DEVICES

Mouse Button

- Actions
  - Press, release
- States
  - Pressed, released
- State transition diagram

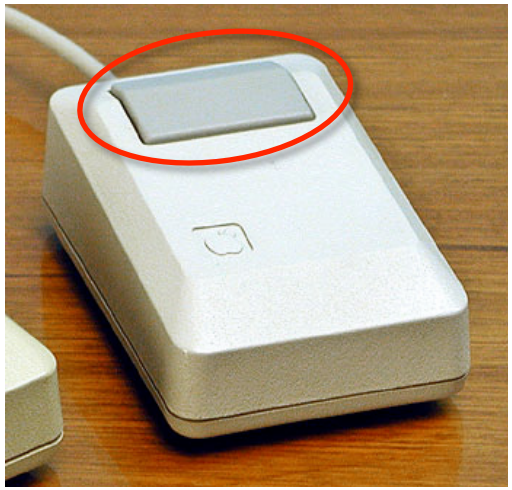




- State machine of mouse interacts with state machines of widgets

How to press the button

- Double click
  - click: press followed by release
  - Example: Start program, select word
- Long press
  - press, wait, feedback occurs, release
  - Example: Android, popup menu
- Click + Modifier-key?
  - Example: Mac mouse (button + Ctrl for context menu)
- Triple click
  - Example: Select line
- Double click, then long press?

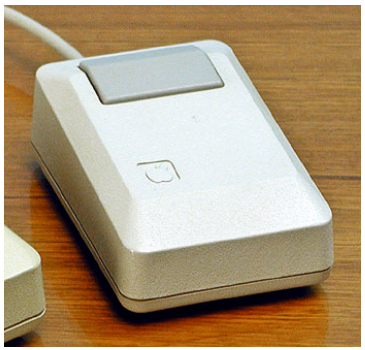


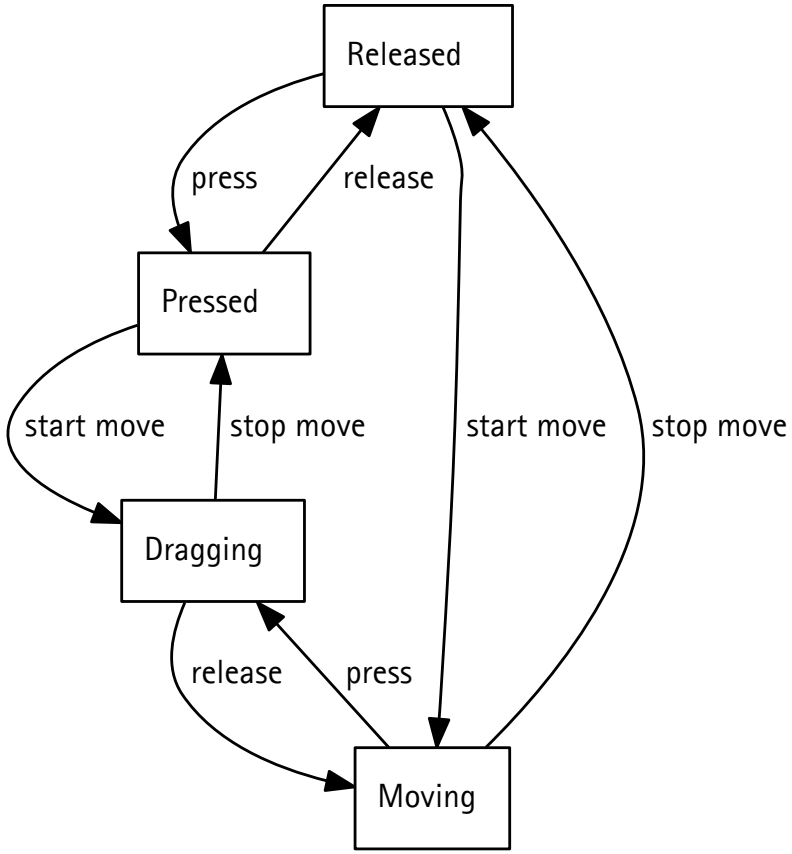
Feedback

- Physical button
  - Tactile feedback when pressed
  - Auditory feedback when pressed
  - Visual feedback on screen
- Vibration?
  - Could add vibration to mouse to indicate events
  - Crossing boundaries, events occurring, progress bar

Single-Button Mouse

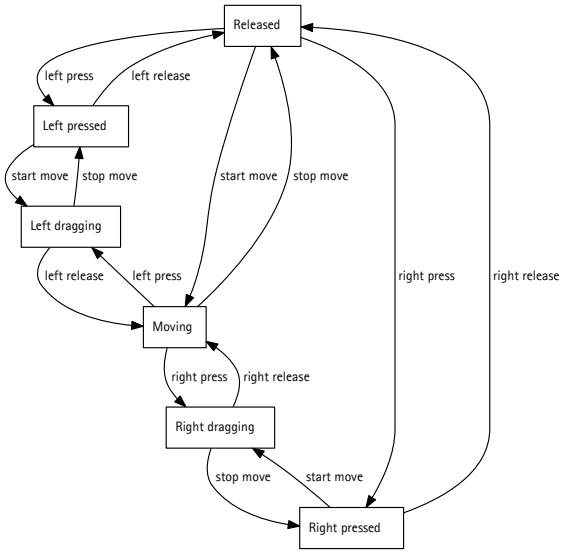
- Actions
  - Press, release, start move, stop move
- States
  - Pressed, released, moving, dragging
- State transition diagram

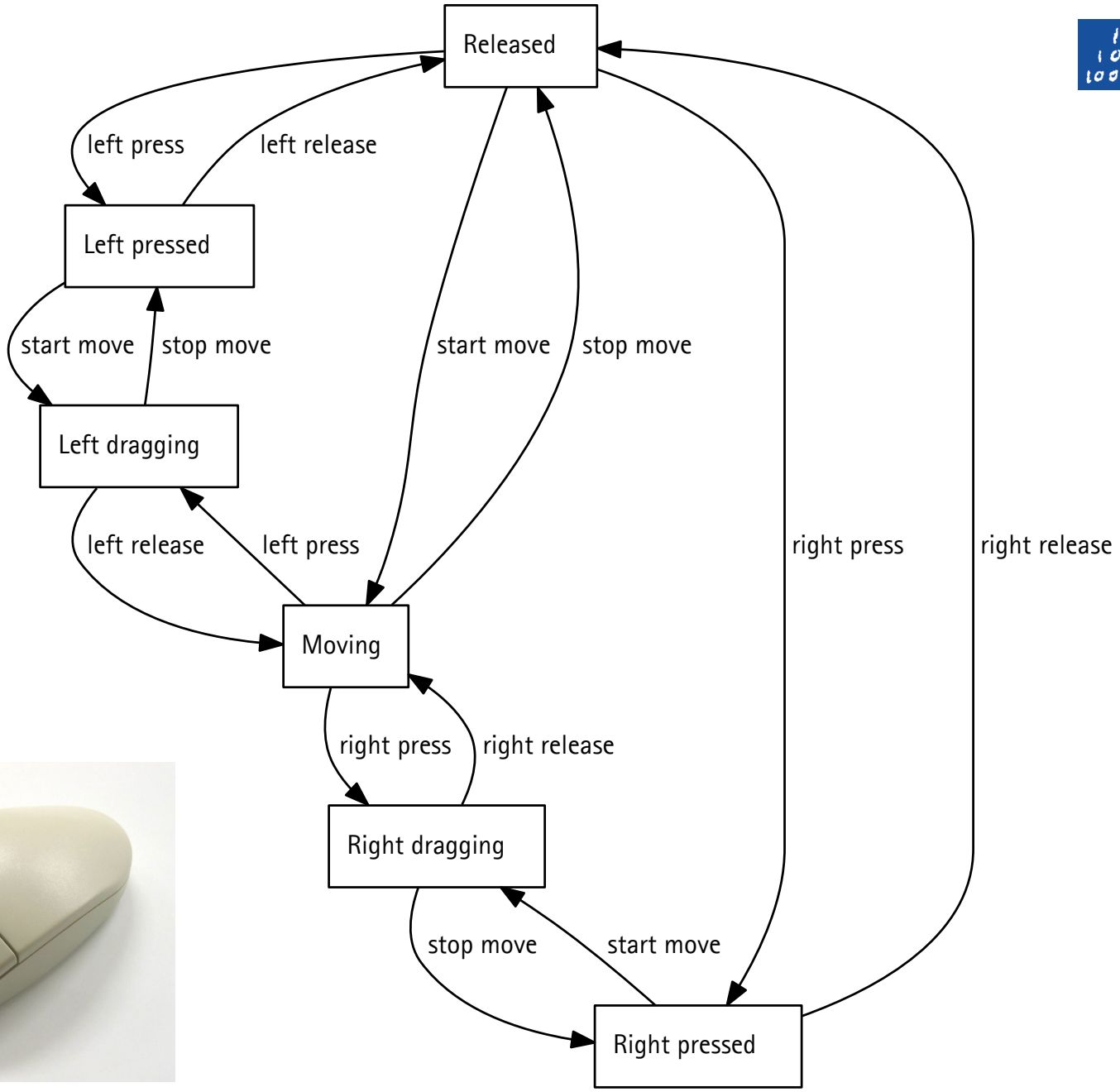




Two-Button Mouse

- 6 Actions
  - Left press, left release, right press, right release, start move, stop move
- 6 States
  - Left/right pressed, released, moving, left/right dragging
- State transition diagram

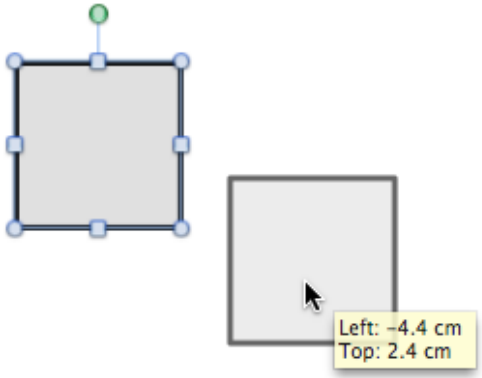



Released
left press
left release
Left pressed
start move
stop move
start move
stop move
Left dragging
left release
left press
right press
right release
Moving
right press
right release
Right dragging
stop move
start move
Right pressed

Right dragging used in practice?

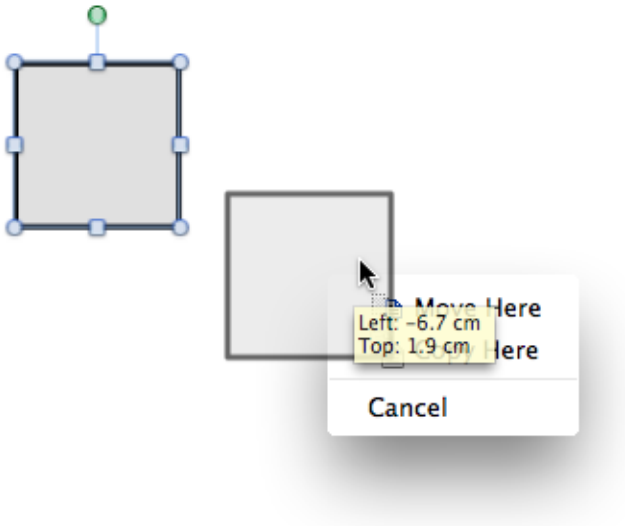
Example: PowerPoint

Left drag:



Move

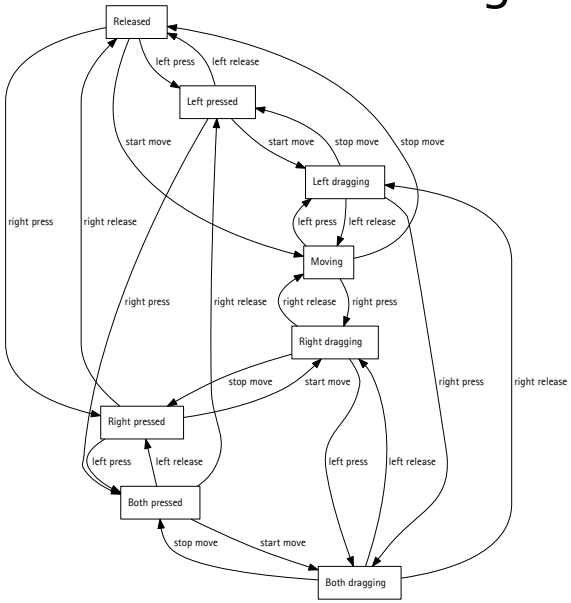
Right drag:

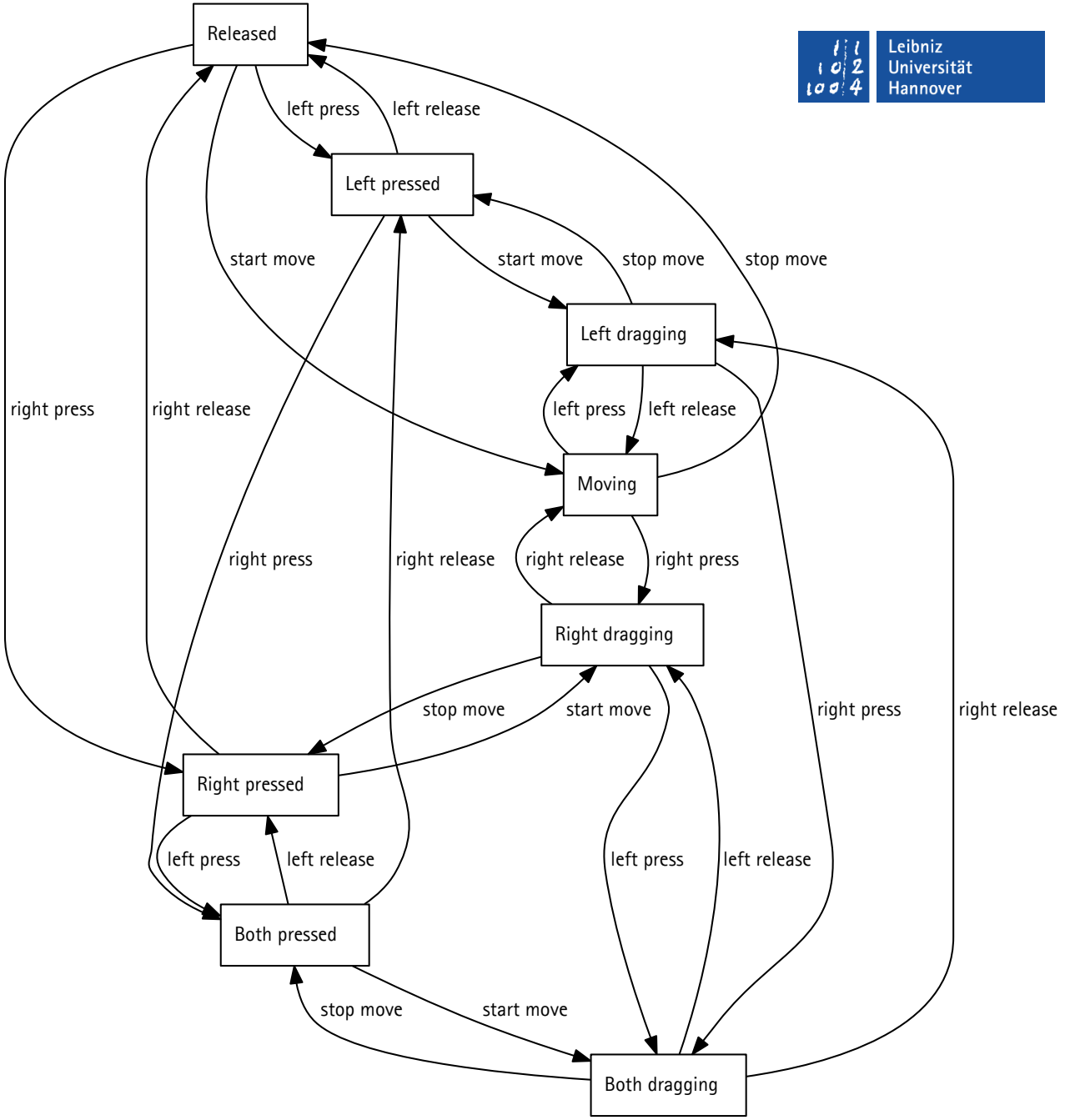


Move, Copy, or Cancel

Two-Button Mouse

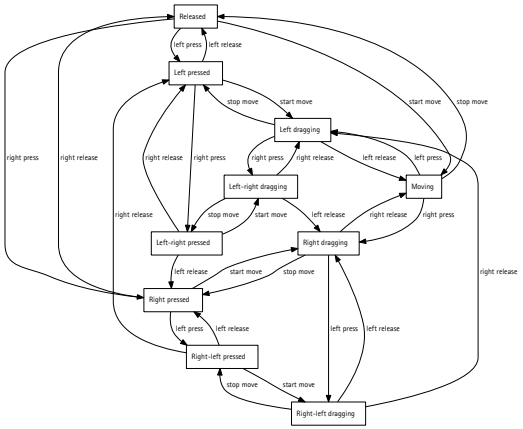
- 6 Actions
  - Left press, left release, right press, right release, start move, stop move
- 8 States
  - Left/right/both pressed, released, moving, left/right/both dragging
- State transition diagram

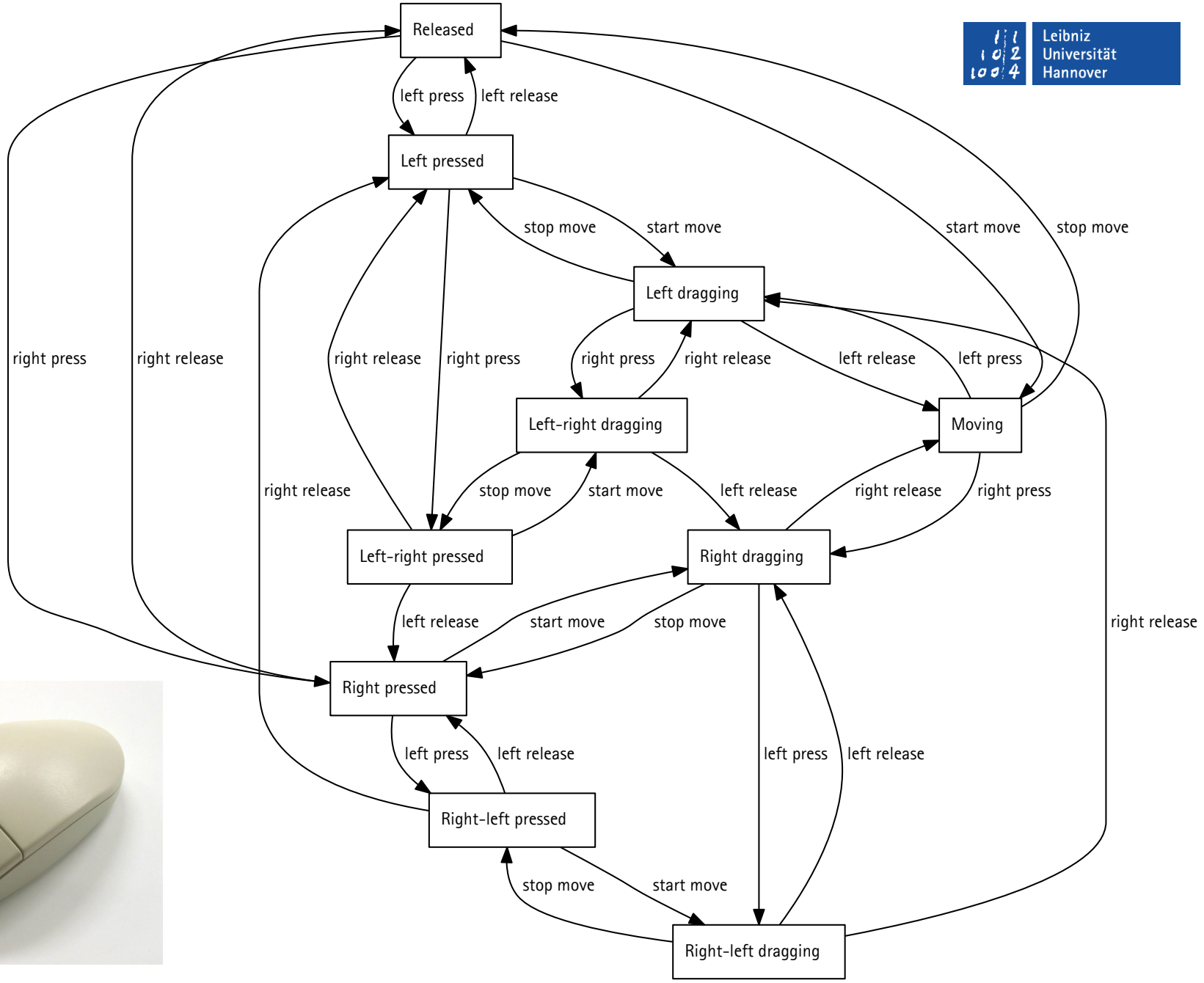



Released
left press
left release
Left pressed
start move
start move
stop move
stop move
Left dragging
right press
right release
left press
left release
Moving
right press
right release
right release
right press
Right dragging
stop move
start move
right press
right release
Right pressed
left press
left release
left press
left release
Both pressed
stop move
start move
Both dragging

Two-Button Mouse

- Distinguish between left-then-right and right-then-left pressed
- 6 Actions
  - Left press, left release, right press, right release, start move, stop move
- 10 States
  - Left/right/left-right/right-left pressed, released, moving, left/right/left-right/right-left dragging
- State transition diagram




Released
left press
left release
Left pressed
stop move
start move
start move
stop move
Left dragging
right press
right release
right release
right press
right press
right release
left release
left press
Left-right dragging
Moving
right release
stop move
start move
left release
right release
right press
Left-right pressed
Right dragging
left release
start move
stop move
right release
Right pressed
left press
left release
left press
left release
Right-left pressed
stop move
start move
Right-left dragging

Class Discussion

- Distinction between
  - Press left-then-right
  - Press right-then-left
- What could this be used for?

Hard Controls, Soft Controls

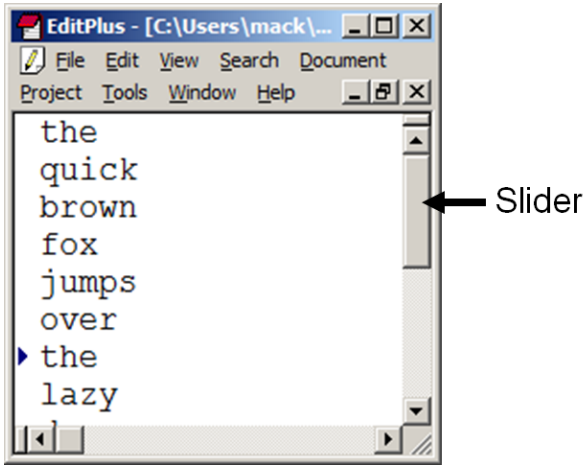
- In the past, controls were physical, single-purpose devices → hard controls
- Today's graphical displays change state
- Interfaces created in software → soft controls
- Soft controls rendered on a display
- Distinction blurred between soft controls and displays
- Consider controls to format a slide

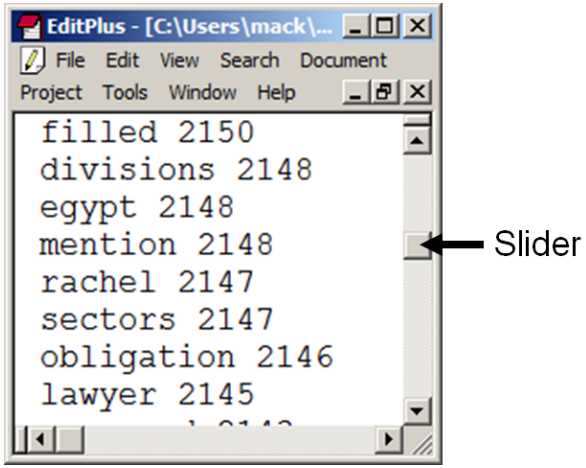
Controls are also displays!

MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Scrollbar Slider

- Example of a soft control (control + display)
- As a control
  - Moved to change view in document
- As a display
  - Position reveals view location in document
  - Size reveals view size relative to entire document

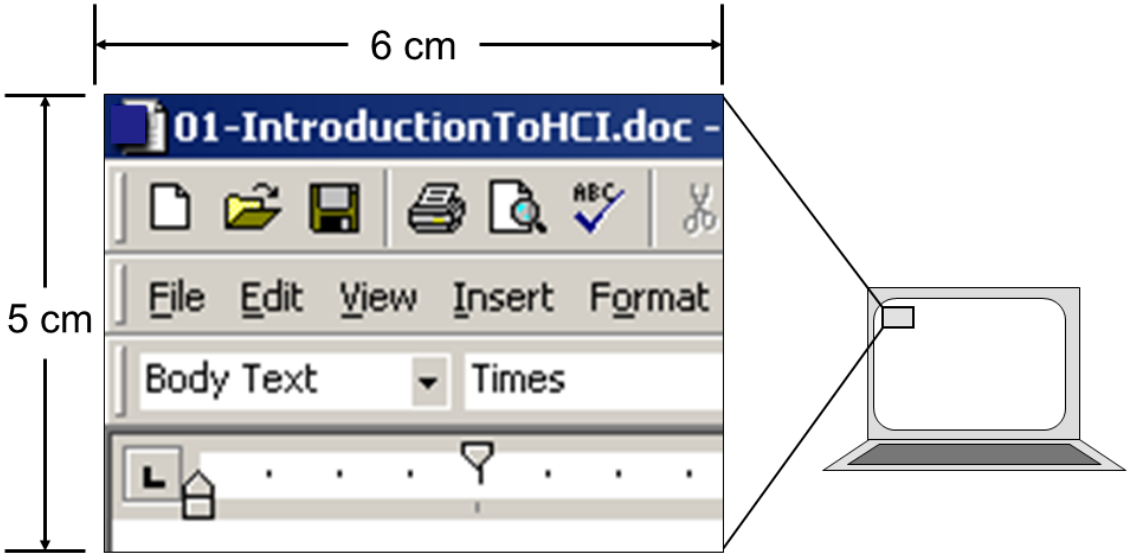




MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

GUI Malleability

- Below is a 30 cm^2 view into a GUI
- >20 soft controls (or are they displays?)



- Click a button and this space may morph into a different set of soft controls/displays

MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Instrumental Interaction (Beaudouin-Lafon)

- Interaction instruments mediate between users and objects of the application domain (domain objects)
  - Like conventional tools mediate between people and the physical world

- Interaction instruments translate the physical actions of the user into commands on the application object

Beaudouin-Lafon. Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. CHI 2000.

Instrumental Interaction (Beaudouin-Lafon)

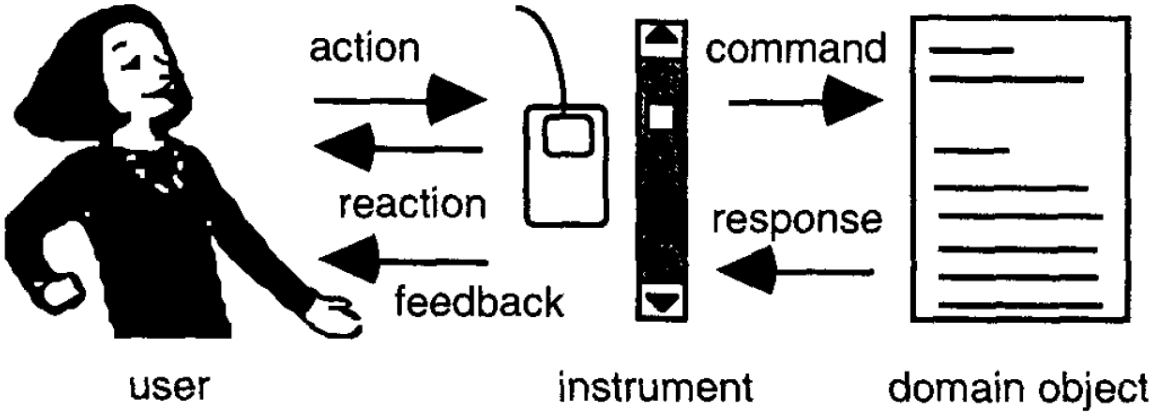


Figure 1: Interaction instrument mediating the interaction between a user and a domain object

- Instruments composed of a physical part and a logical part
- Explicit activation of instruments in conventional GUIs
  - Example: Mouse can be associated with different logical parts, like scrollbars and menus

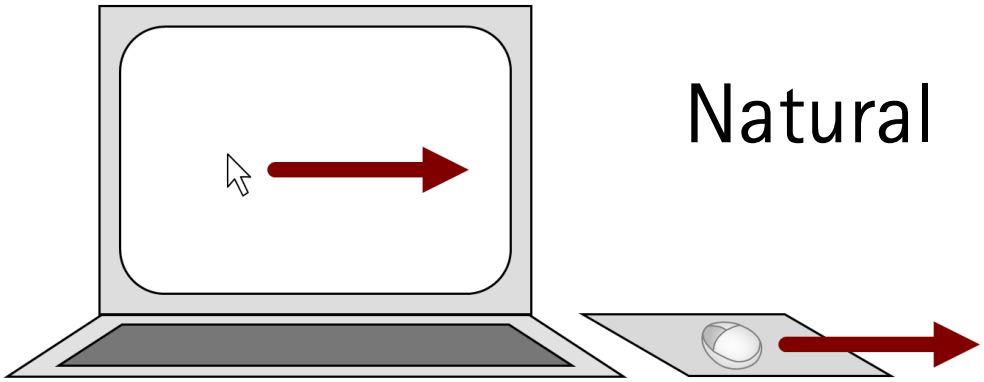
Beaudouin-Lafon. Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. CHI 2000.

Control-Display Relationships

- Relationship between operation of a control and the effect created on a display
  - Also called mappings
- Spatial relationship
  - Movement direction of hand/finger → movement direction of interaction element
- Dynamic relationship
  - Movement speed of hand/finger → movement speed of interaction element

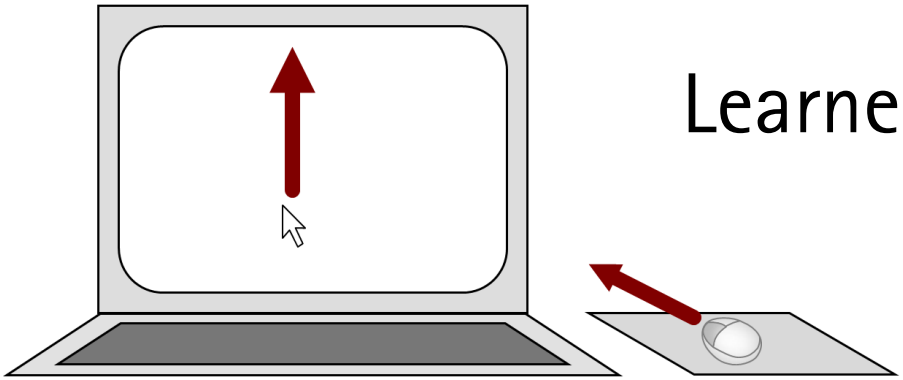
MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Spatial Relationships



Natural

Spatial congruence
Control: right
Display: right



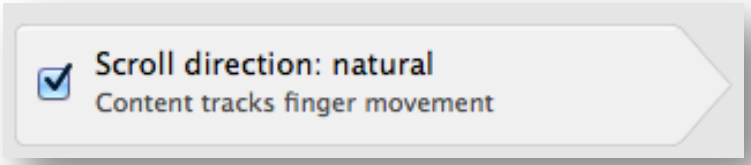
Learned

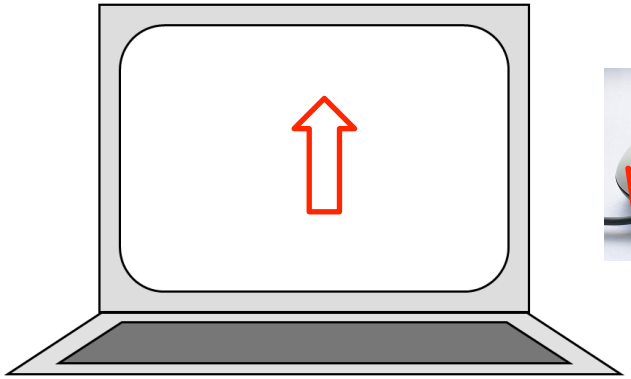
Spatial transformation
Control: forward
Display: up

MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

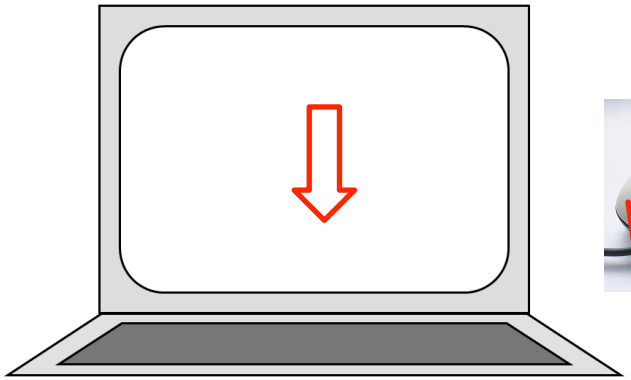
Learned Spatial Relationships

- Wheel forward: content up



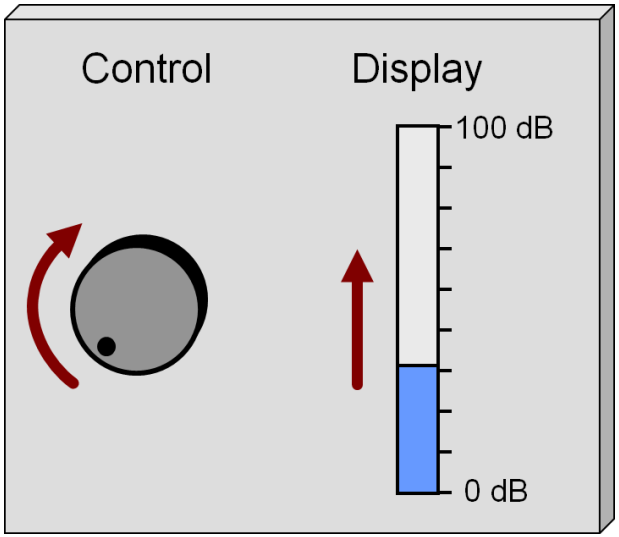


- Wheel forward: content down



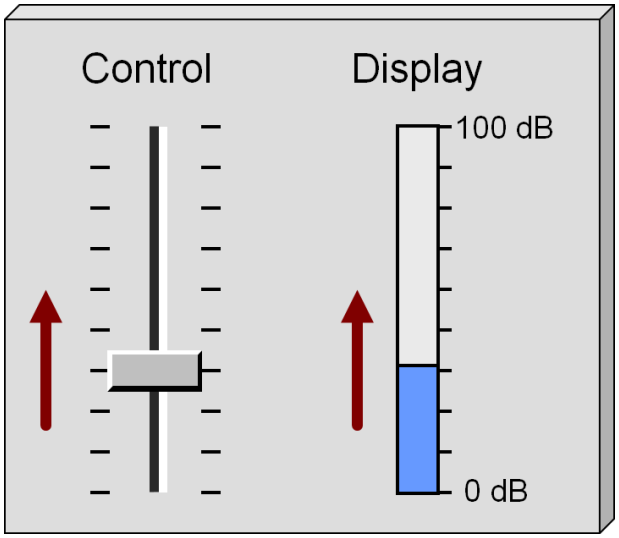
- Learned spatial relationships can quickly be relearned

Learned relationship



DOF	Control	Display
x		
y		+
z		
θ_x		
θ_y		
θ_z	+	

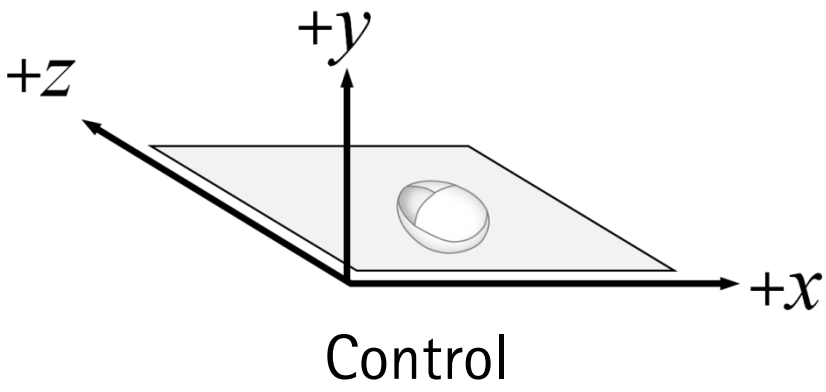
Natural relationship



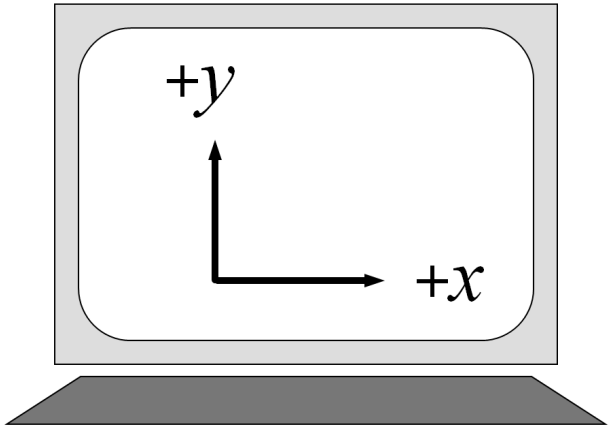
DOF	Control	Display
x		
y	+	+
z		
θ_x		
θ_y		
θ_z		

MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Axis Labeling



Control

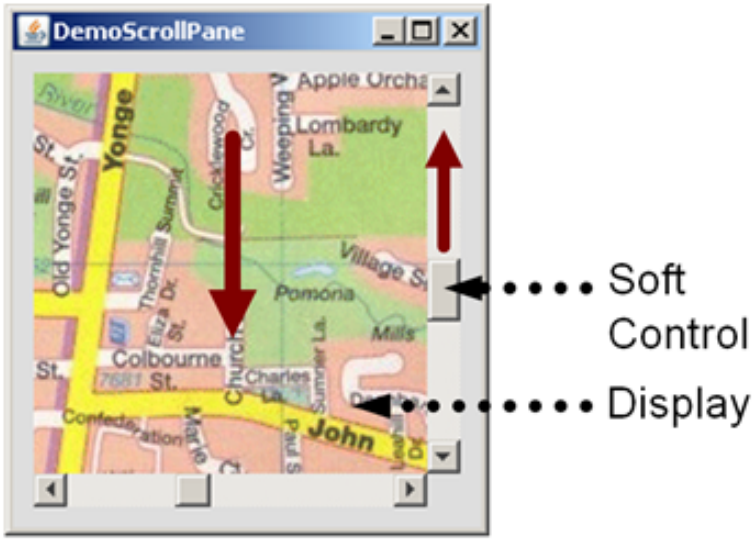


Display

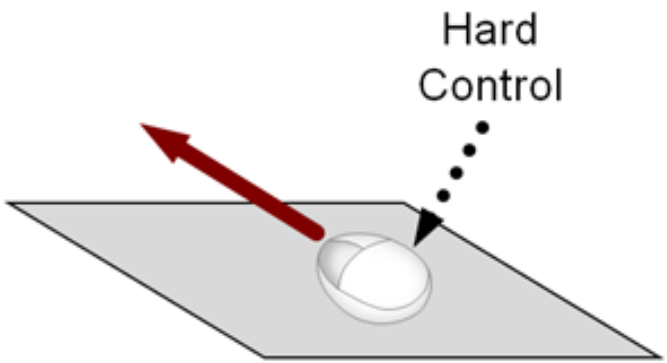
Axis	Control (mouse)	Display (cursor)
x	+	+
y		+
z	+	

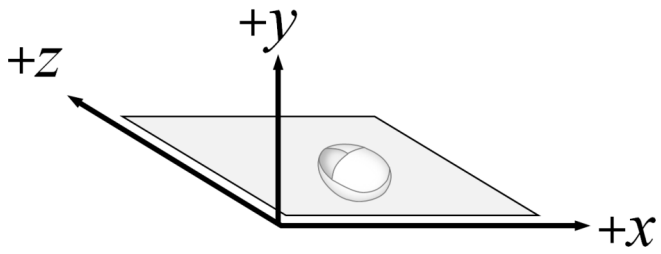
MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Indirection through Interaction Element



DOF	Hard Control	Soft Control	Display
x			
y		+	-
z	+		
θ_x			
θ_y			
θ_z			

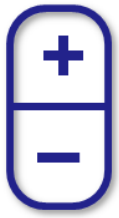




MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

3D in Interactive Systems

- Usually interaction devices supports subset of the 6 DOF
  - Degrees of freedom (DOF): Number of independent parameters
- Spatial transformations must be learned
- Example: Google StreetView

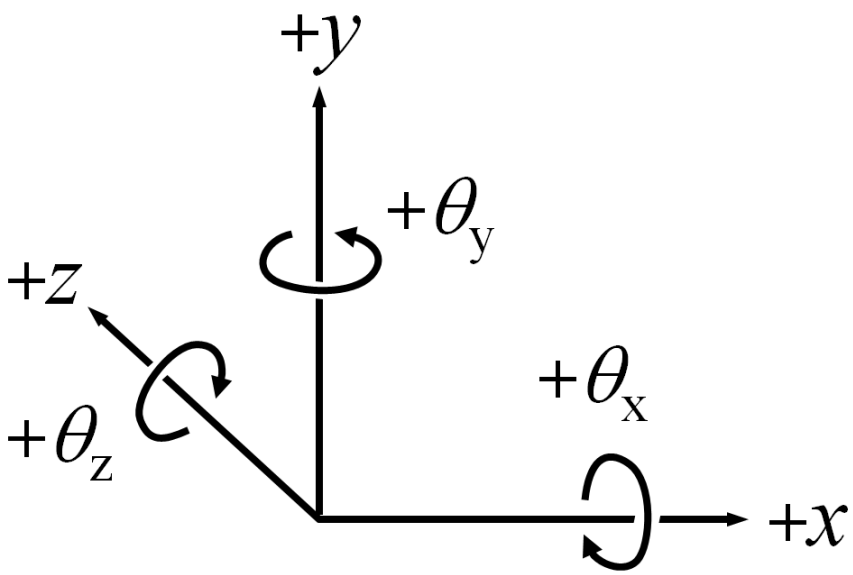


MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Panning in Google StreetView

- Example: Google StreetView, panning with the mouse
- Spatial transformations

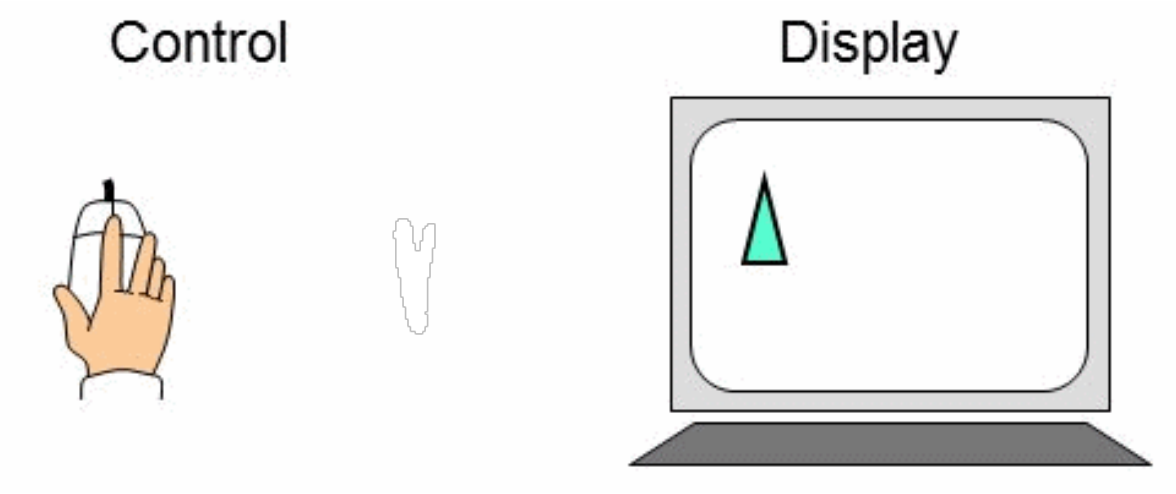
DOF	Control	Display
x	+	
y		
z	+	
θx		+
θy		-
θz		



MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Modes and Degrees of Freedom

- If control DOF < display DOF, modes are necessary to fully access the display DOF
  - Remember modes? (HCI 1)
- Consider a mouse (2 DOF) and a desktop display
- x-y control (no problem):

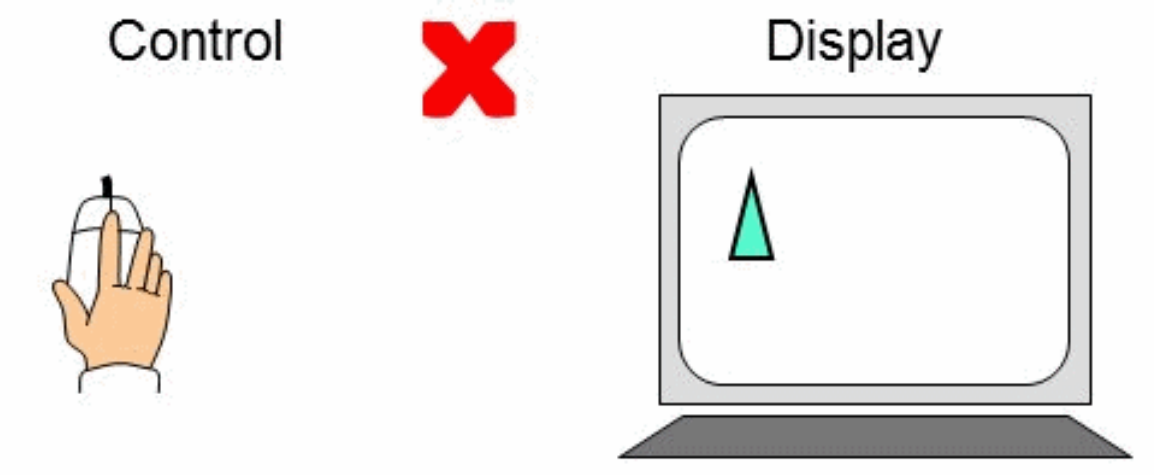


but...

MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Modes and Degrees of Freedom

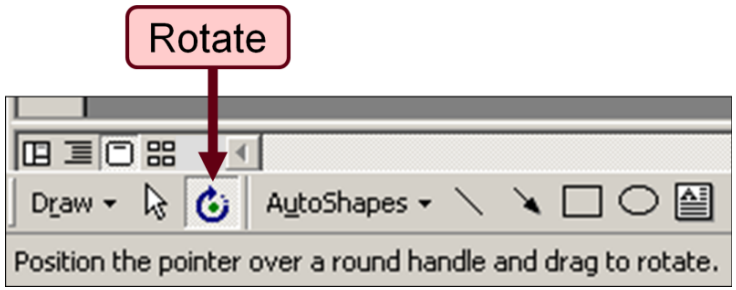
Rotation is a problem:

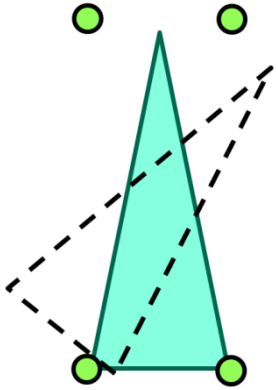


MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

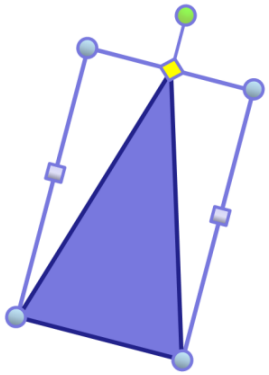
Rotate Mode

- The solution: Rotate mode
- Two approaches
  - Separate rotate mode:





  - Embedded rotate mode:



Could be avoided with...

MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

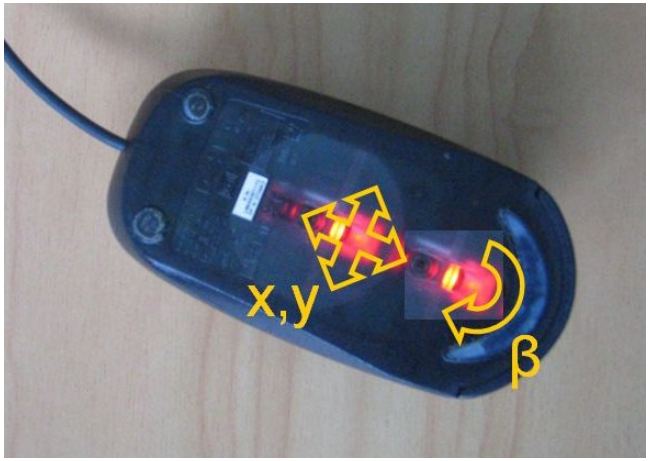
3 DOF Mouse

- Lots of research:

1

2

3



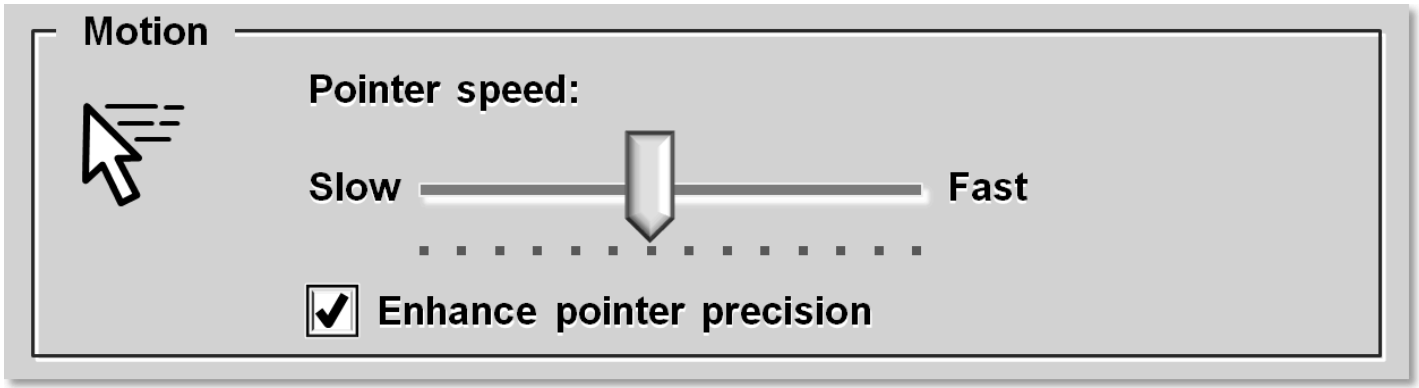
- But no commercial products (yet)

[1] Almeida, Cubaud. Supporting 3D window manipulation with a yawing mouse. Proc. NordiCHI 2006.
[2] MacKenzie, Soukoreff, Pal. A two-ball mouse affords three degrees of freedom. Proc. CHI 1997.
[3] Hannagan, Regenbrecht. TwistMouse for simultaneous translation and rotation. Tech Report. HCI Group. University of Otago, New Zealand, 2008.

MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Control-Display (CD) Gain

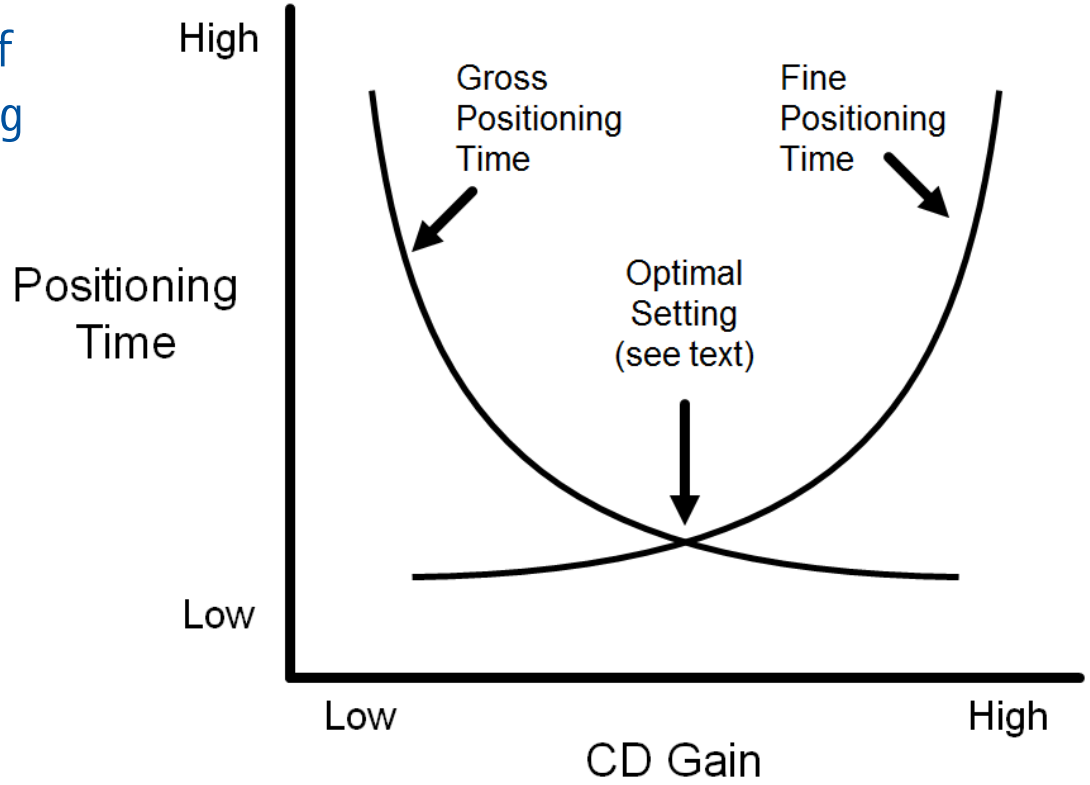
- Quantifies the amount of display movement for a given amount of controller movement
  - Example: 2 cm controller movement → 4 cm display movement
- Sometimes specified as a ratio (C:D ratio)
- For non-linear gains, the term transfer function is used
- Typical control panel to adjust CD gain:



MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

CD Gain and User Performance

- Tricky to adjust CD gain to optimize user performance
- Issues
  - Speed accuracy trade-off (what reduces positioning time tends to increase errors)
  - Opposing relationship between gross and fine positioning times



MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Latency

- Latency (aka lag or delay) is the delay between an input action and the corresponding response on a display
- Often negligible
  - Example: Key presses, character appearance
- Noticeable in some settings
  - Remote manipulation (e.g. robot control)
  - Virtual reality (VR)

MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Virtual Reality (VR) Controllers

- 6 DOF controllers common in VR and other 3D environments
- Considerable processing requirements
- Lag often an issue
- E.g., Polhemus G4™
- Lag specified as <10 ms (very low)

MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

Control: Absolute vs. Relative, Position vs. Velocity

- Property sensed
  - Absolute position (touchscreen)
  - Relative displacement (mouse)
  - Force (joystick)

- Property controlled
  - Position (of cursor/object): position control
  - Velocity (of cursor/object): rate control

MacKenzie: Human-Computer Interaction - An Empirical Research Perspective.

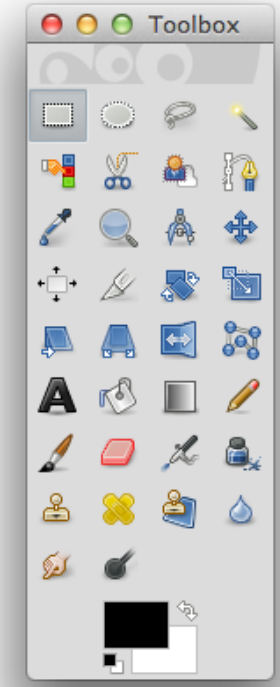
INTERACTION ELEMENTS

Widgets (aka Controls)

- Widgets (aka Controls) are user interface elements
  - Examples: Buttons, check boxes, edit fields, tooltips
- Basic building blocks for creating graphical user interfaces
  - Generic, reusable, directly manipulable
  - Have a well-defined, consistent behavior across applications
  - Offer familiar affordances to the user
  - Important to get the details right
- Organized into widget toolkits
  - Need to play together well

Widgets (aka Controls)

- Widgets are tools to manipulate the content
  - Each widget solves a specific interaction problem
- Verb-object relationship
  - Widgets (tools) are the verb
  - Content (data) are the objects
  - Example: Pick a color (tool, verb) from an image pixel (data, object)

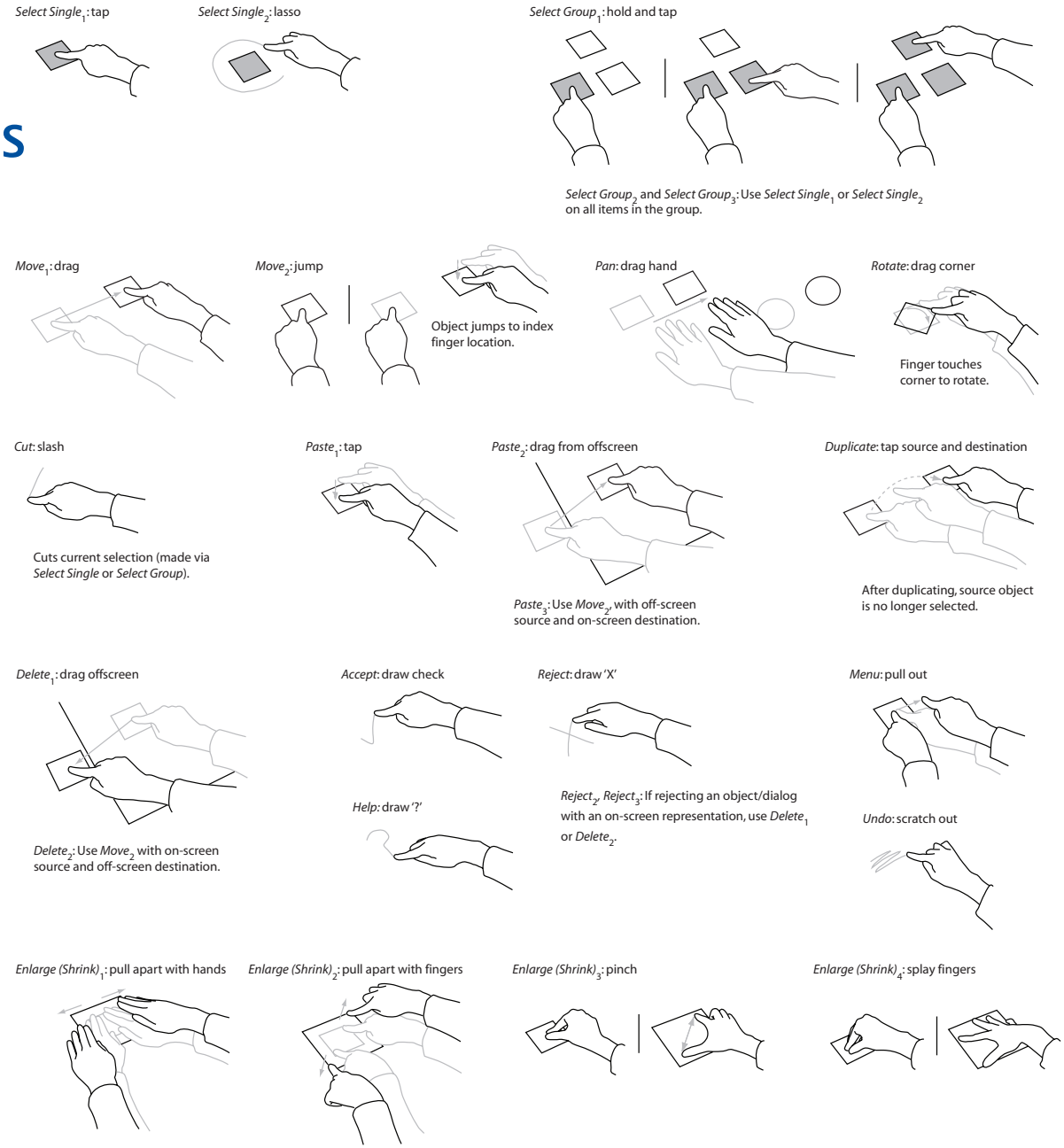


“No Chrome”

- Widgets are often in conflict with the actual content
  - Widgets need screen space
  - Widgets lead to visual clutter
- Multitouch interfaces (“natural user interfaces”, NUIs) try to avoid widgets (“no chrome”)
  - Multitouch allows encoding verbs in gestures
  - Example: Pinch-to-zoom
- “No chrome” philosophy
  - The interface elements should get out of the way to let users focus directly on the content itself
  - Works well for media consumption interfaces (immersion)
  - Works well for experts (focus on task, no visual distractions)

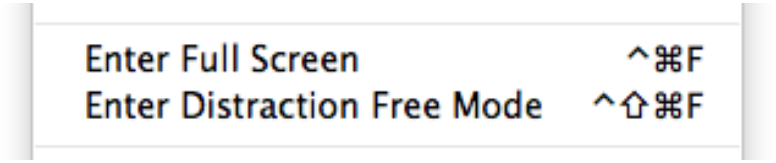
Multitouch Gestures

- No widgets
- Rich set of gestures instead
- Natural, but
  - Discoverability
  - Learnability
  - Memorizability
  - Gesture recognition errors

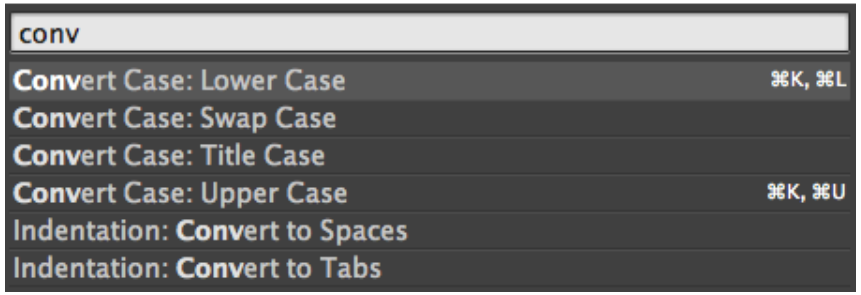


“No Chrome” Example: Sublime Text Editor

- “Distraction free” mode for full concentration on text editing
  - “You and your text. Nothing else.”



- Each function has a keyboard shortcut
- Function access through search interface
  - Supports discoverability
- Transient views
  - Chrome only while invoking a function



Widgets Categories (Cooper)

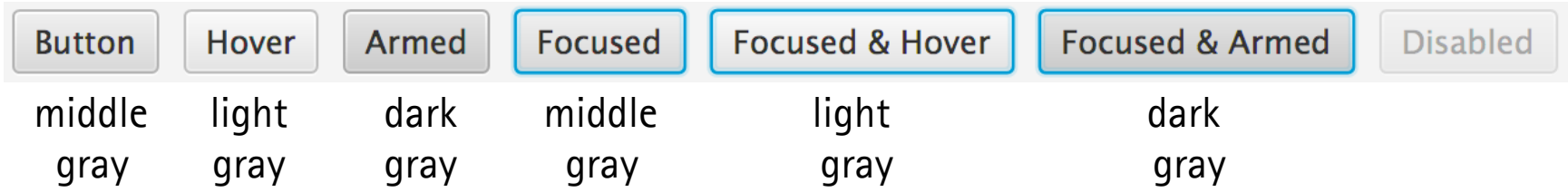
- Imperative widgets
  - Initiate a function
  - Example: buttons
- Selection widgets
  - Allow the user to select some option or data
  - Example: check boxes
- Entry widgets
  - Enable the input of data
  - Examples: sliders (bounded), text edit fields (unbounded)
- Display widgets
  - Manipulation and appearance of data
  - Examples: scroll bar, split pane

Widget Focus and Hover

- Focus
  - At most one widget has the input focus, i.e., receives input events
  - Widgets on the screen have a focus order
  - Setting the focus
    - Mouse press sets focus on specific widget
    - Tab shifts focus to next widget
    - Shift+Tab shifts focus to previous widget
- Hover
  - Widgets respond to mouse cursor movement above them

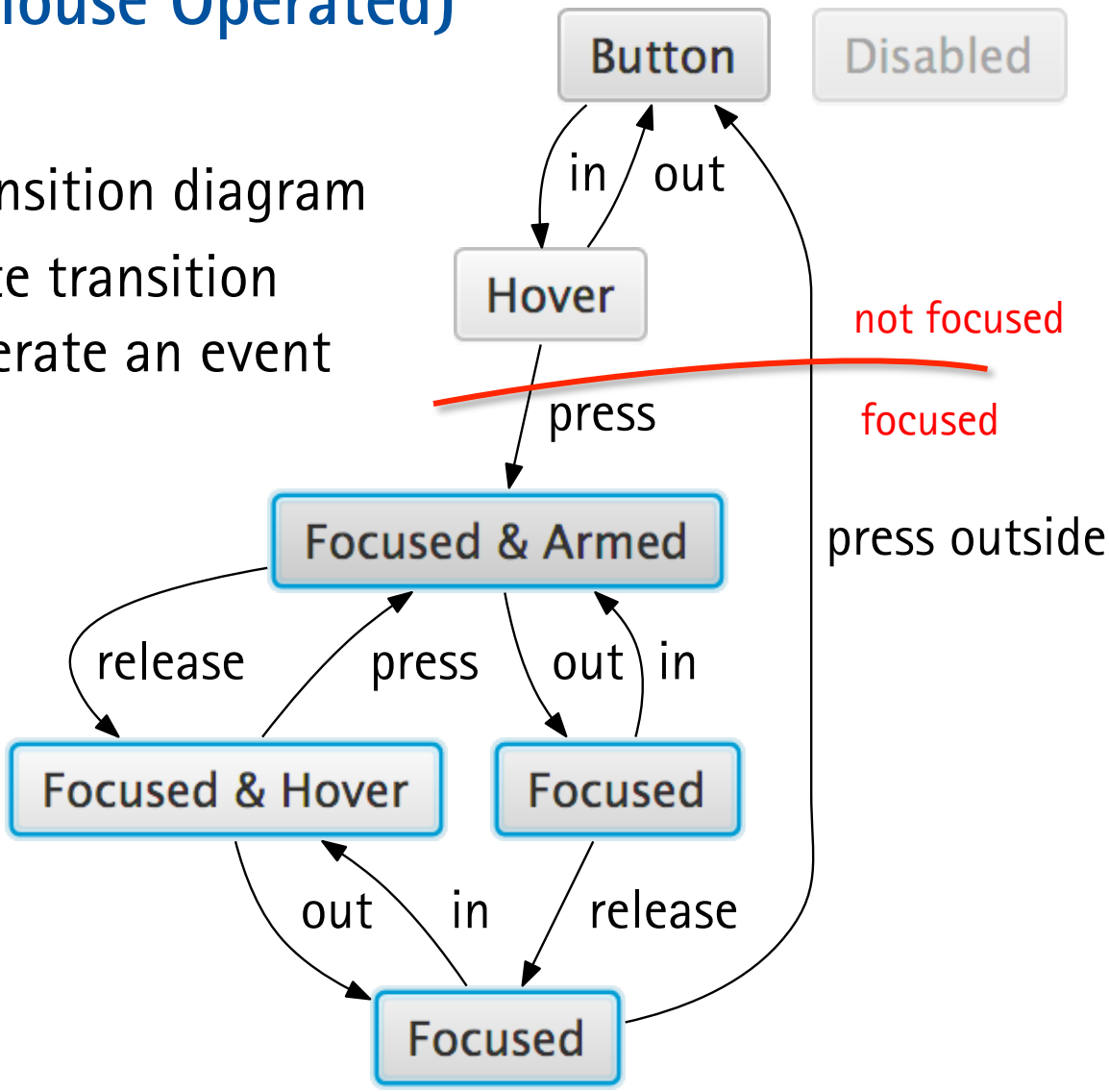
Button

- Actions
  - Mouse: move in, move out, press, release
  - Keyboard: Tab, Shift+Tab, Space
- States
  - Neutral, Hover, Focused & Armed, Hover & Focused, Focused
  - Disabled



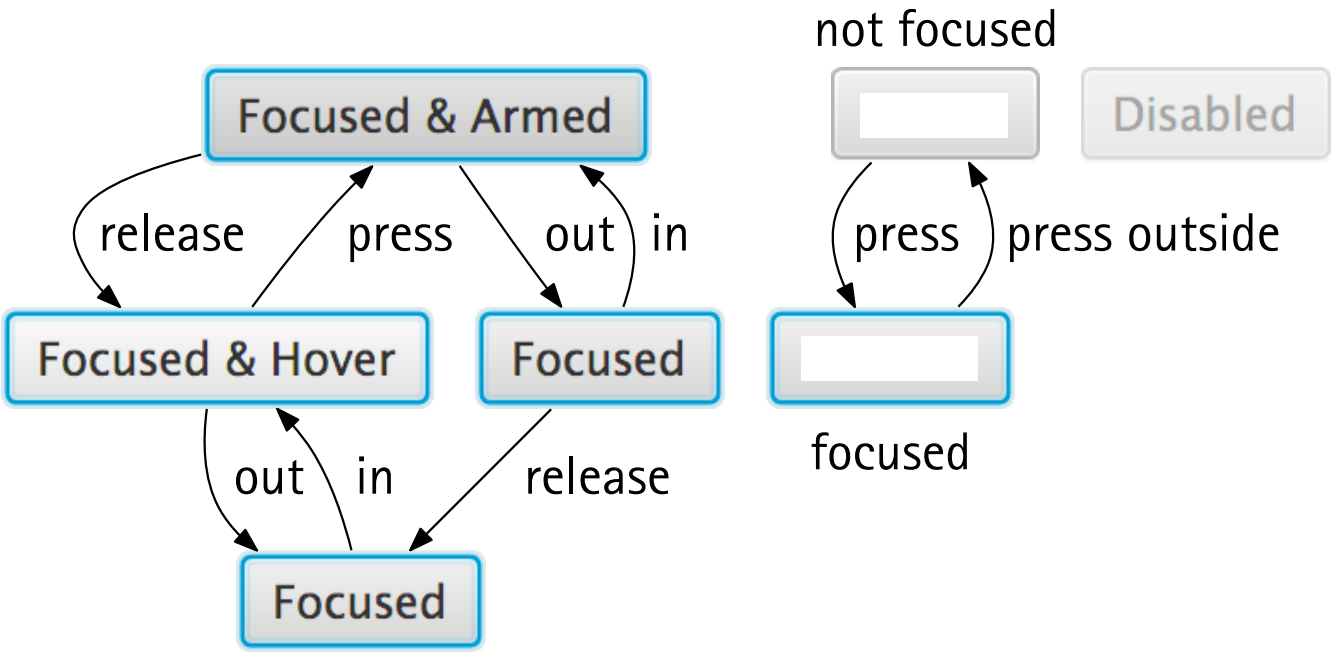
Button (Mouse Operated)

- State transition diagram
- Each state transition may generate an event



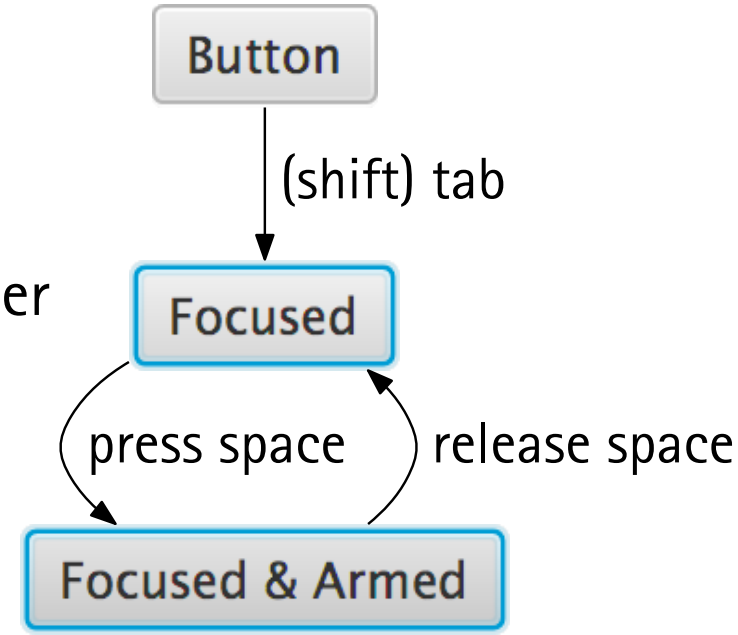
Button (Mouse Operated)

- Separate diagrams for focus <-> no-focus
  - Action press always results in focus
  - A press outside always results in no-focus



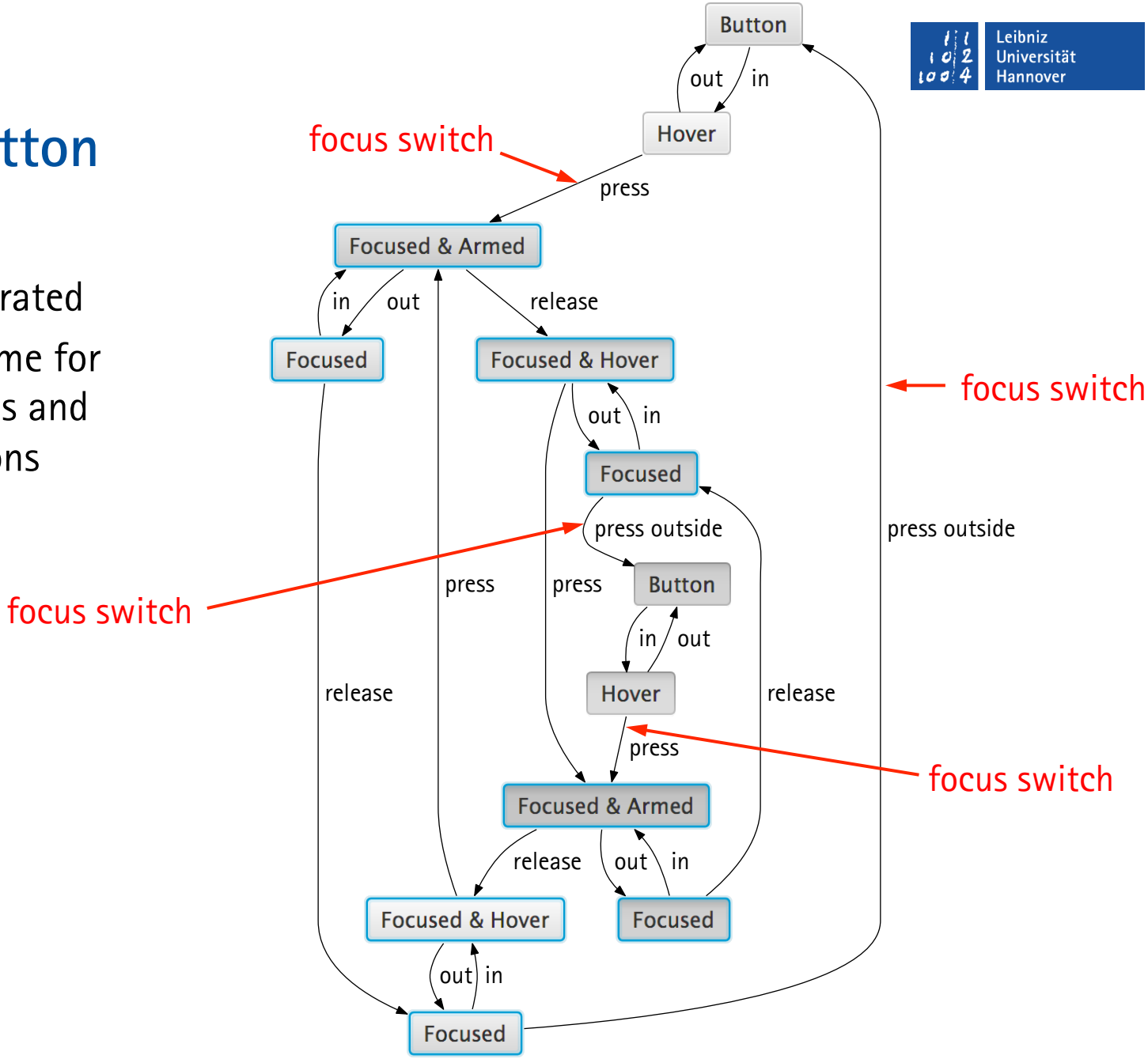
Button (Keyboard Operated)

- Operating widgets via keyboard
  - May prevent change of input device
- Homing time: Moving from one input device to another
  - Here: Keyboard to mouse
- Widgets on a page have a focus order
- Tab-key shifts focus
  - Next element: tab
  - Previous element: shift + tab
- Space activates focused button
  - Press and release space



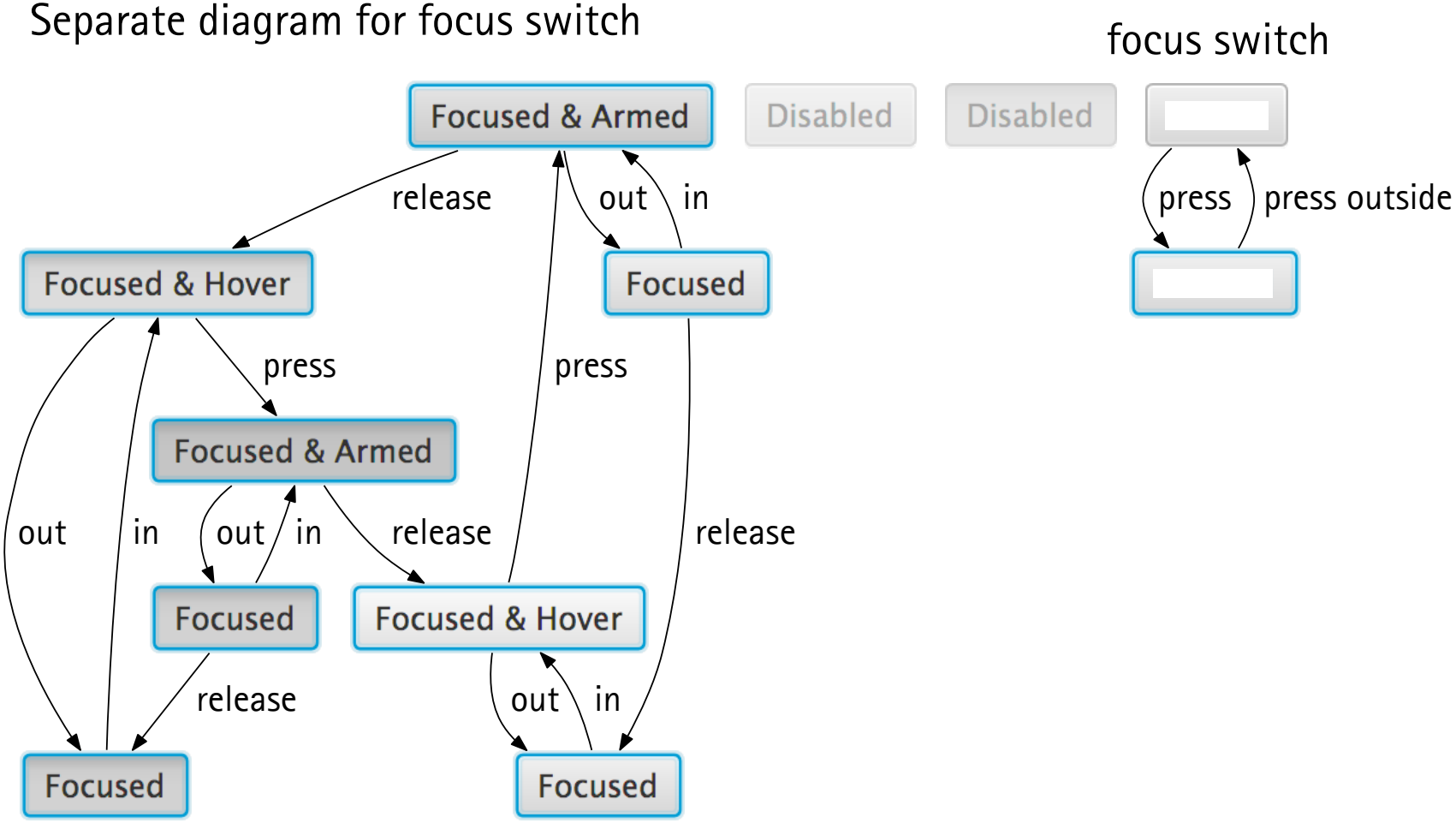
Toggle Button

- Mouse operated
- Same scheme for check boxes and radio buttons

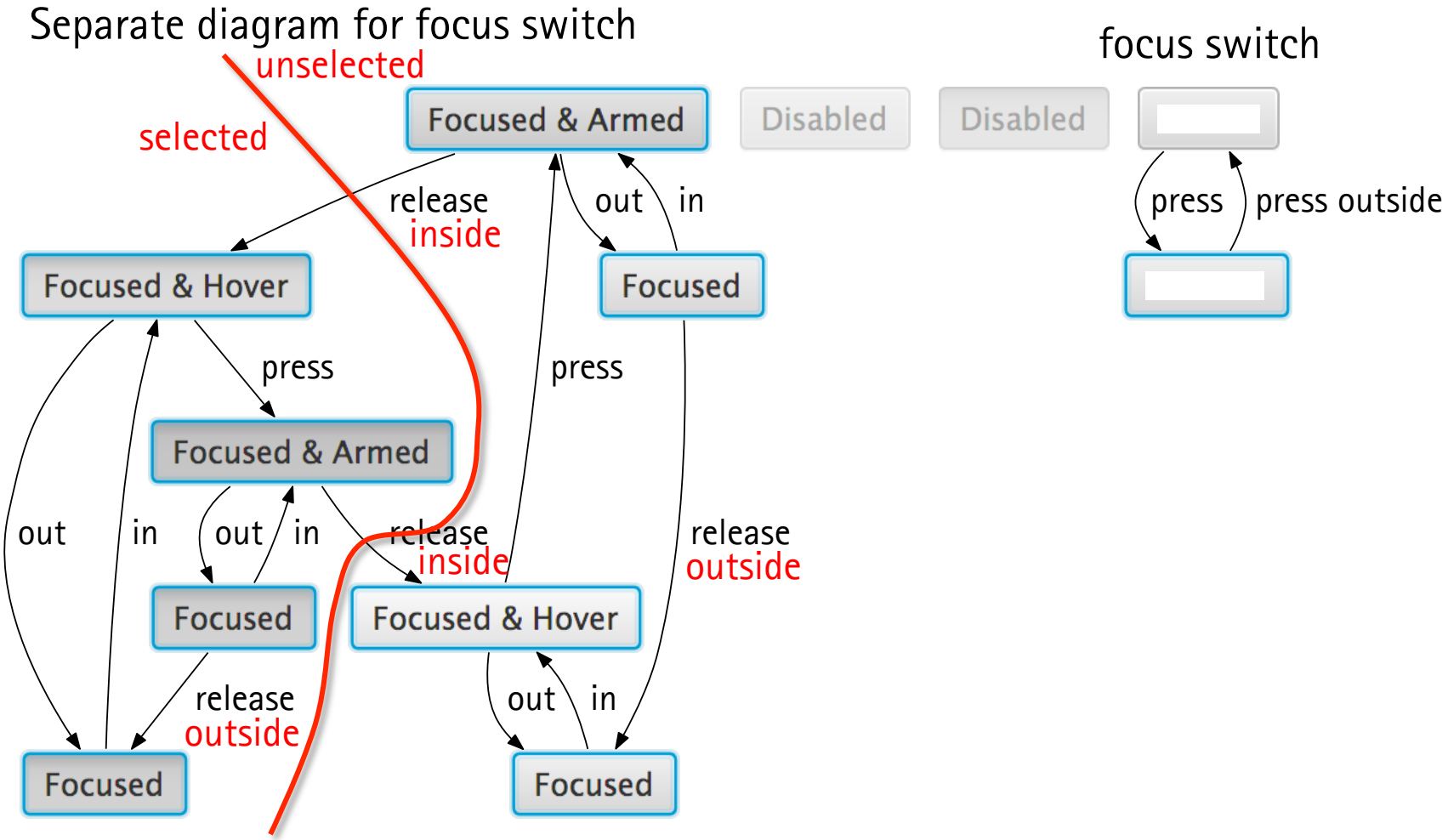


Toggle Button

Separate diagram for focus switch

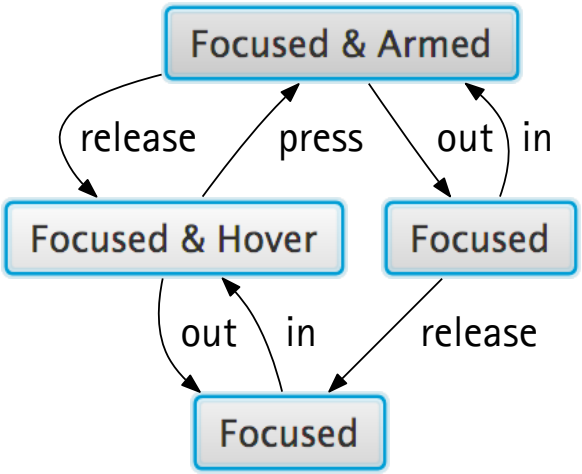


Toggle Button

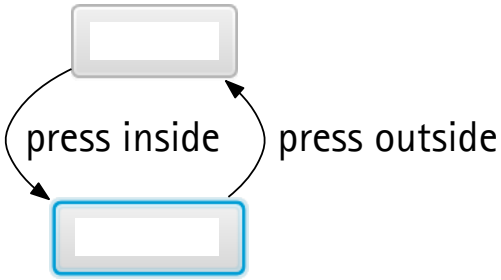


Toggle Button

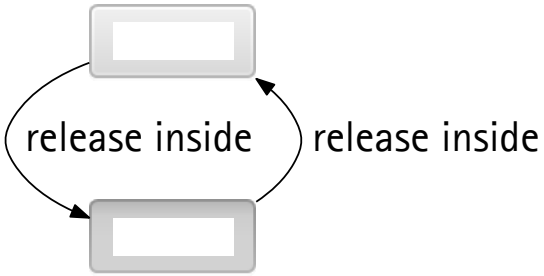
- Separate diagrams for focus switch and selection switch



focus switch



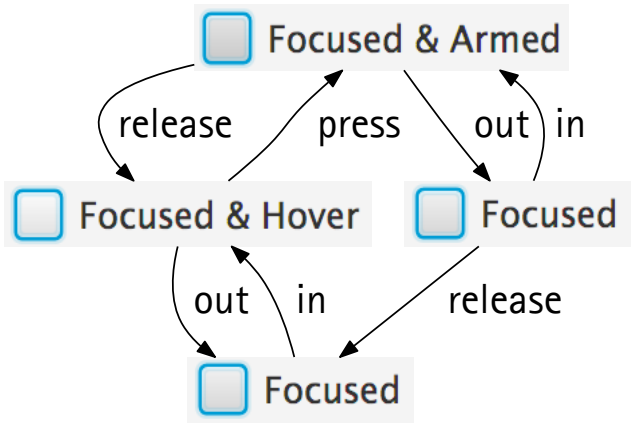
selection switch



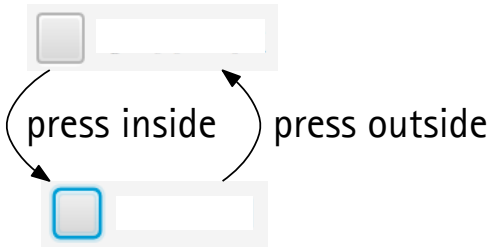
- Left diagram is identical to simple button diagram
- Separation of diagrams leads to simpler description

Check Box

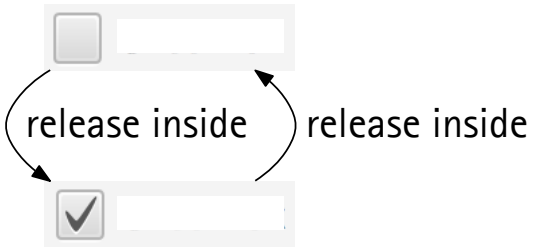
- Diagram structure identical to toggle button diagram



focus switch

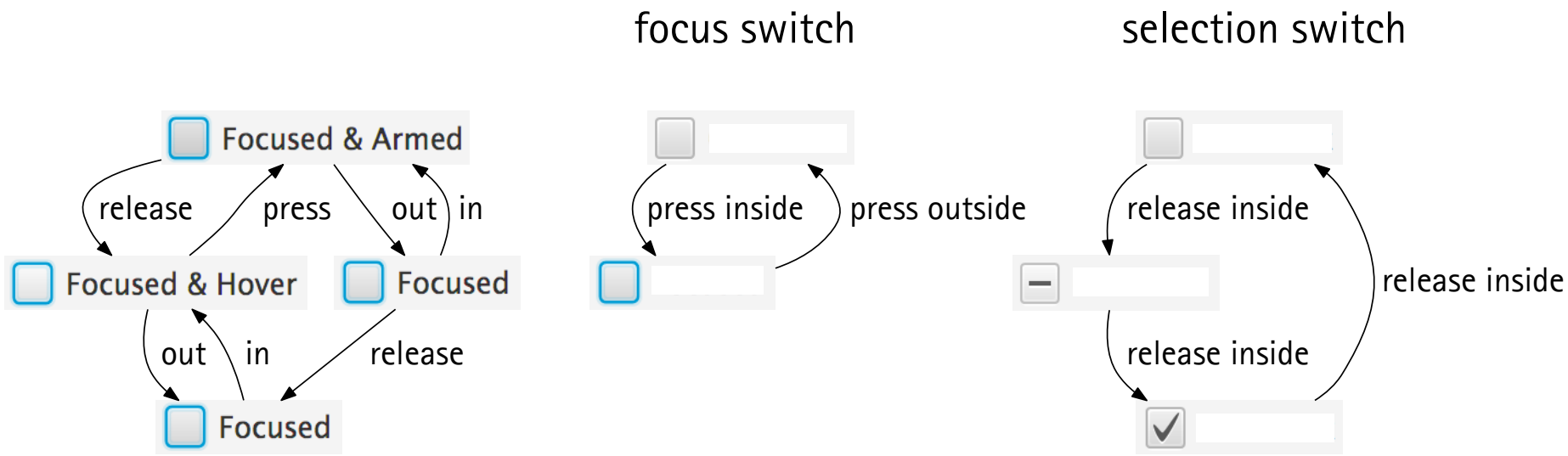


selection switch



Check Box

- Trivial to add another state (indeterminate)

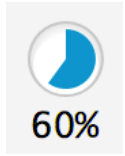


Widgets Categories (JavaFX)

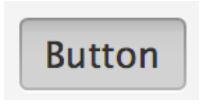
- Naming and status widgets
  - Feedback and visibility
- Imperative widgets
  - Triggering an action
- Selection widgets
  - Changing a status
- Entry widgets
  - Data input
- Collections of items
  - Structured data
- Layout and navigation widgets
  - Hierarchies of widgets

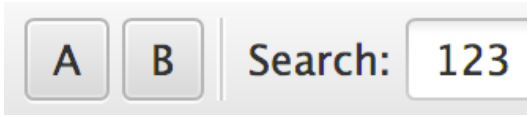
Widgets Categories (JavaFX)

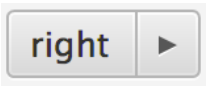
- Naming and status widgets (feedback and visibility)
  - Label, tooltip
  - Progress bar, progress indicator
- Imperative widgets (triggering an action)
  - Button, toggle button, tool bar
  - Menu, menu bar, context menu
  - Menu button, split menu button
- Selection widgets (changing a status)
  - Check box, radio button, choice box, combo box (not editable)
  - Color picker, date picker
  - File chooser

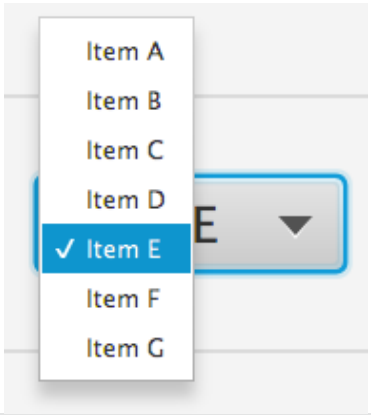


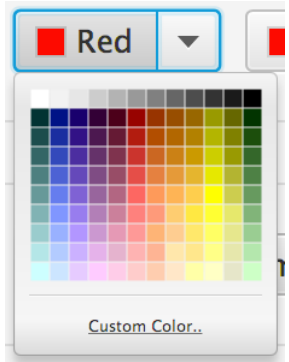
(Widgets from JavaFX)





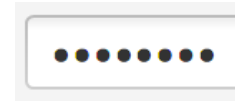


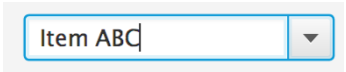


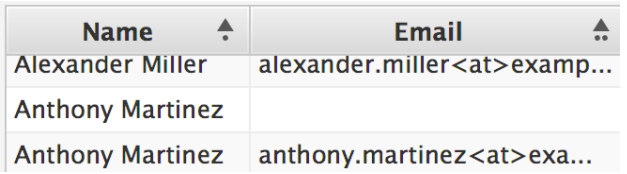


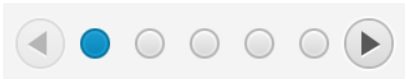
Widgets Categories (JavaFX)

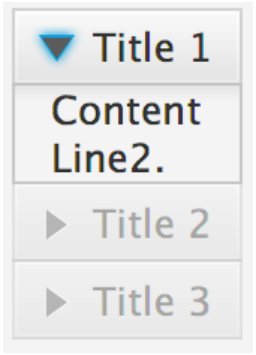
- Entry widgets (data input)
  - Text field, password field, text area, HTML editor
  - Combo box (editable)
  - Scroll bar, slider
- Collections of items (structured data)
  - List view, table view, tree view, tree table view
  - (Charts)
- Layout and navigation widgets
  - Scroll pane, split pane
  - Tab, tab pane
  - Titled pane, accordion
  - Pagination
  - Hyperlink

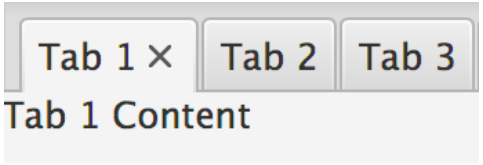








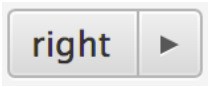




(Widgets from JavaFX)

Orthogonality

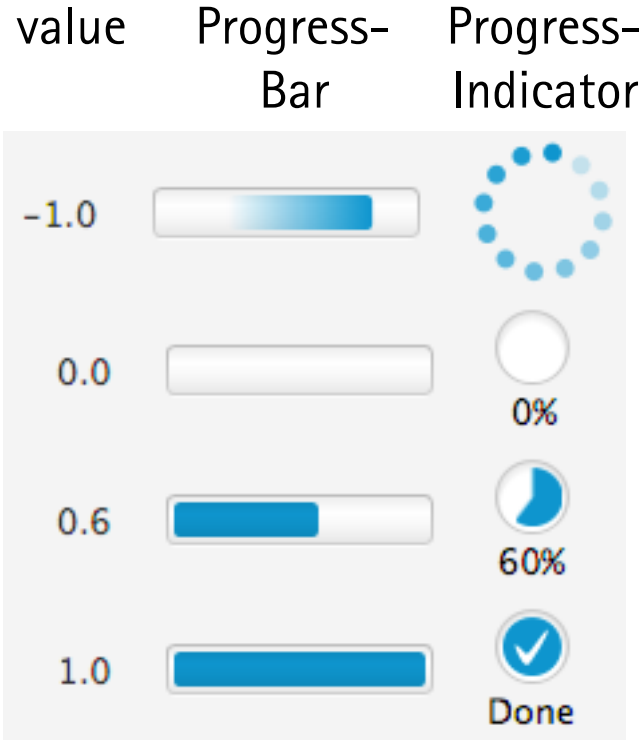
- Ideally: Widgets can be freely combined with each other
- Richness from simple, well-defined components
- Examples
  - Text field in a table view for editing a cell
  - Split menu button within a toolbar



- Layout and navigation widgets can be nested
- Example
  - Text area within a scroll pane
    within a split pane
    within a tab pane
    within a pane...

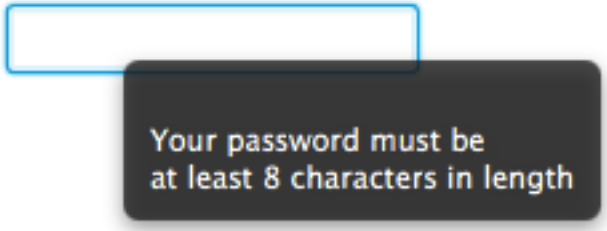
Feedback on Progress

- Value indicates state of progress
  - value < 0.0: indeterminate mode
  - value = 0.0: start
  - value = 0.6: 60% complete
  - value = 1.0: completion
- Shows progress on time-consuming operation (use if duration ≥2s)
- If possible, avoid indeterminate mode, show time to completion, show how much work has been done
- Provide cancel button

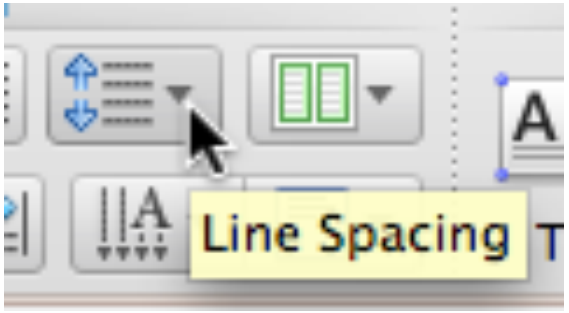


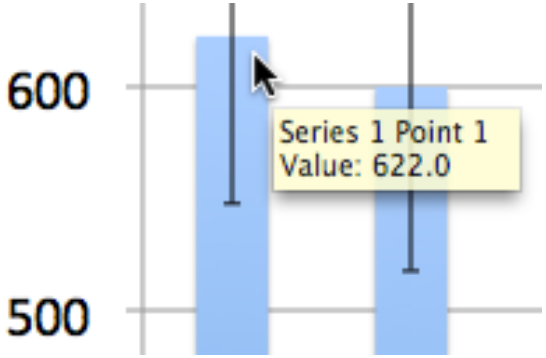
http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/progress.htm

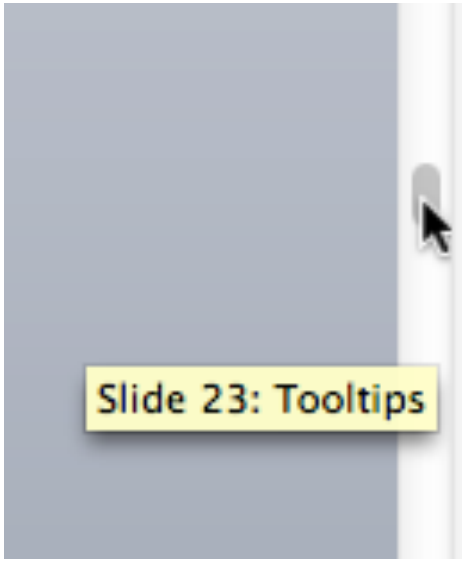
Tooltips



- Tooltip can be attached to any widget
  - Appears after delay when hovered over
  - Shows additional information
- Important component of multi-level help, crucial for icons!
- Can serve as "You are here" indicators on a scrollbar, visual attention
- Can describe item under mouse cursor
  - Data value in a graph
  - POI on a map

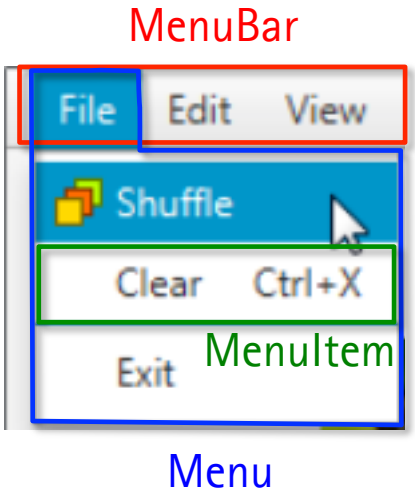






http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/tooltip.htm

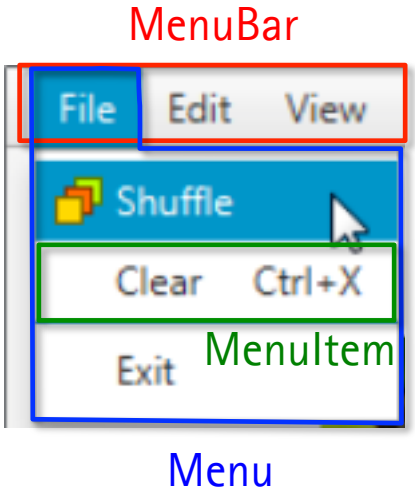
Menus



- Lists of items, grouped into categories
  - Follow platform conventions
  - Affinity diagramming for categories
- Menu bar is always visible
  - Typically at top of window or screen
- Menus and menu items are hidden before and after usage
  - Space-efficient but slow access
  - Serves as an index of the application functionality
  - Duplicates items on toolbars and in context menus
- Show keyboard shortcuts for faster access
  - Allows user to become more efficient (expert user)

http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/menu_controls.htm

Menus (in JavaFX)



- MenuBar: horizontal bar of menus
- Menu: main menu or sub-menu
- MenuItem: a single actionable option
- RadioButtonItem: mutually exclusive selection
- CheckMenuItem: option that can be selected/unselected
- SeparatorMenuItem: horizontal line for separating items
- Menu items with keyboard accelerators

```
MenuItem clear = new MenuItem("Clear");
clear.setAccelerator(KeyCombination.keyCombination("Ctrl+X"));
```

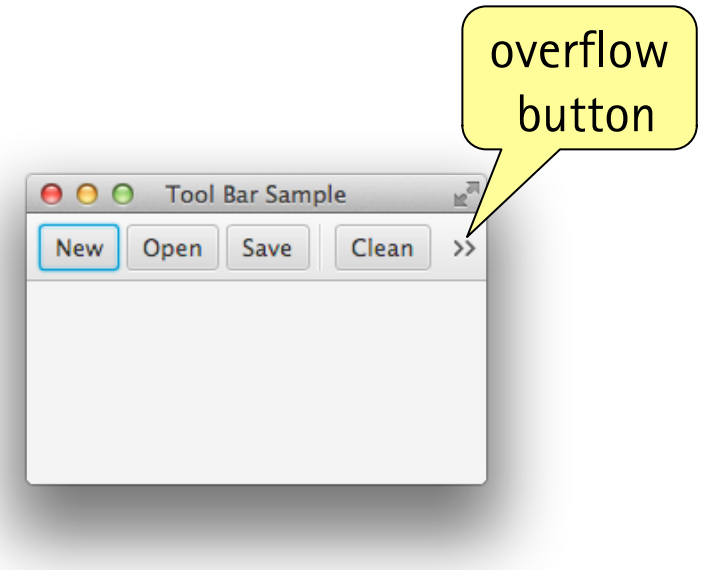
http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/menu_controls.htm

Context Menus (aka Pop-Up Menus)

- Allows associating actions to a data object
- Example
  - Copy an image to the clipboard
- Invisible unless invoked
  - Typically right-clicked
- Often faster than travelling to main menu or tool bar

http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/menu_controls.htm

Tool Bars

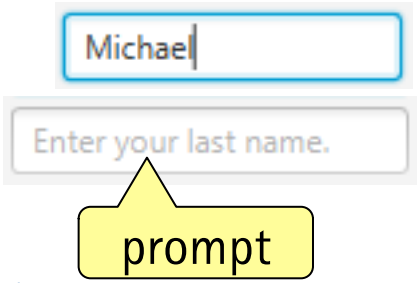


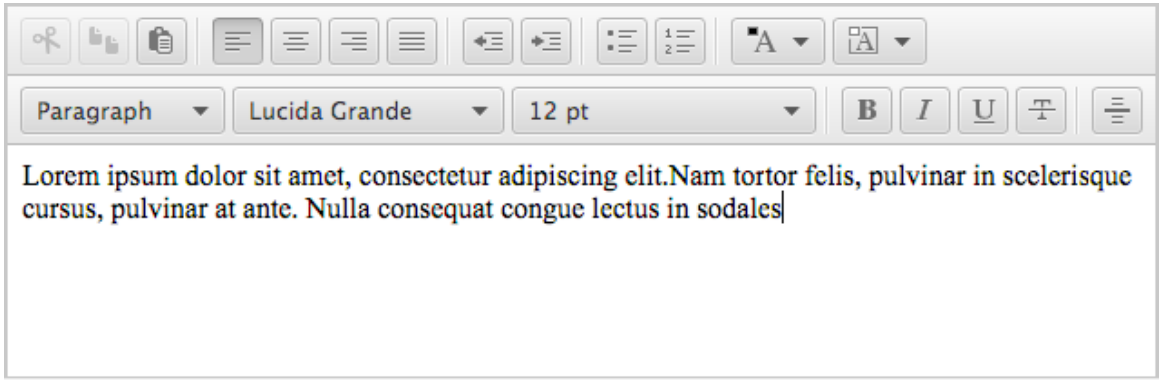
- Horizontal or vertical bar
- Can contain different widgets
  - Button, text entry, combo box, etc.
- Fast access to common functions
  - Put these in main menu as well
- Icons allow for dense packing of buttons
  - Common functions → meaning of icons will be learned
  - Yet, provide tooltips (with keyboard shortcuts)

http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/ToolBar.html

Text Fields and Text Areas (in JavaFX)

- Text fields for entering a single line of text
- Prompts inform the user what to enter
  - Preferable to tooltips
  - More space efficient than labels, but disappear on entry
- Text areas allow entering multiple lines of text
- HTML Editor allows entering multiple lines of formatted text
  - Rendering HTML content with WebView widget



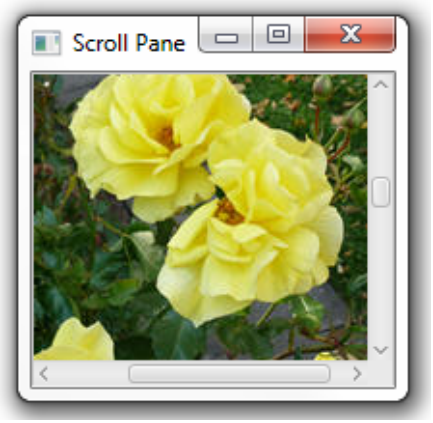


http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/text-field.htm

Layout and Navigation Widgets: ScrollPane

- Laying out and navigating large content
- Scroll pane

```
Image roses = new Image(
        getClass().getResourceAsStream("roses.jpg"));
ScrollPane sp = new ScrollPane();
sp.setContent(new ImageView(roses));
```

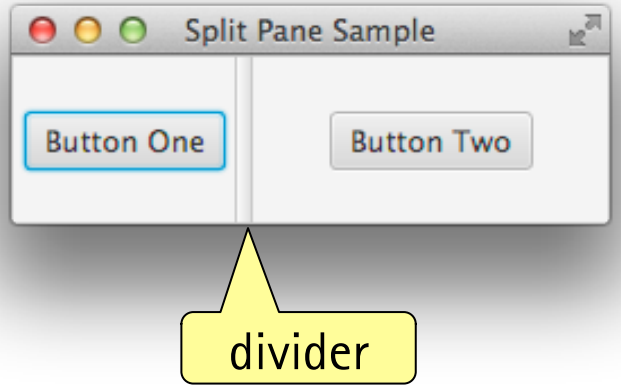


http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/scrollpane.htm

Layout and Navigation Widgets: SplitPane

- Split pane has 2+ sides, separated by a draggable divider
- Elements need to be placed inside a layout container
- Example

```
SplitPane sp = new SplitPane();
StackPane sp1 = new StackPane();
sp1.getChildren().add(new Button("Button One"));
StackPane sp2 = new StackPane();
sp2.getChildren().add(new Button("Button Two"));
sp.getItems().addAll(sp1, sp2);
sp.setOrientation(Orientation.HORIZONTAL);
sp.setDividerPositions(0.4, 0.7);
```

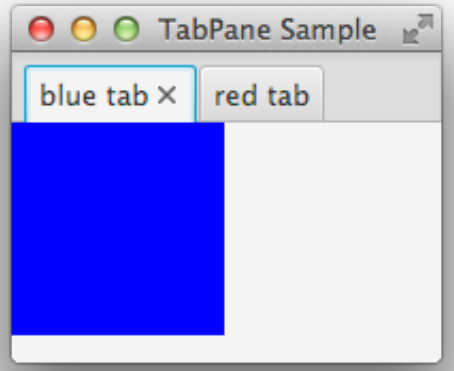


http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/SplitPane.html

Layout and Navigation Widgets: TabPane

- Switching between multiple tabs
  - Space-efficient, easy to switch between categories or views
- Example

```
TabPane tabPane = new TabPane();
tabPane.setSide(Side.TOP);
Tab tab1 = new Tab();
tab1.setText("blue tab");
tab1.setContent(new Rectangle(100, 100, Color.BLUE));
Tab tab2 = new Tab();
tab2.setText("red tab");
tab2.setContent(new Rectangle(100, 100, Color.RED));
tabPane.getTabs().addAll(tab1, tab2);
```

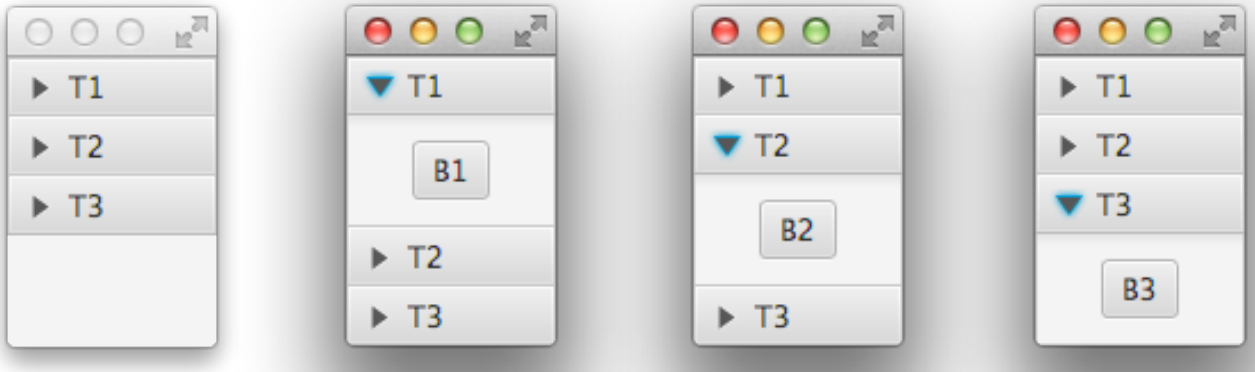


http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TabPane.html

Layout and Navigation Widgets: Accordion

- Accordion: Group of TitledPanes, only one may be open at once
- Example

```
Accordion accordion = new Accordion();
TitledPane t1 = new TitledPane("T1", new Button("B1"));
TitledPane t2 = new TitledPane("T2", new Button("B2"));
TitledPane t3 = new TitledPane("T3", new Button("B3"));
accordion.getPanes().addAll(t1, t2, t3);
```



http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Accordion.html

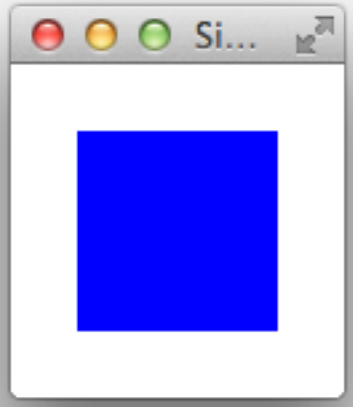
Layout and Navigation Widgets: Pagination

- Pagination for navigation between pages of coherent data

- Page factory callback is invoked when user switches pages
  - Example shows Lambda expression (Java 8)
- Example

```
Pagination pagination = new Pagination(10, 0);
pagination.setPageFactory((Integer page) -> {
        Color c = new Color(0.1 * page, 0, 0, 1);
        return new Rectangle(100, 100, c);
});
```

JavaFX Hello World



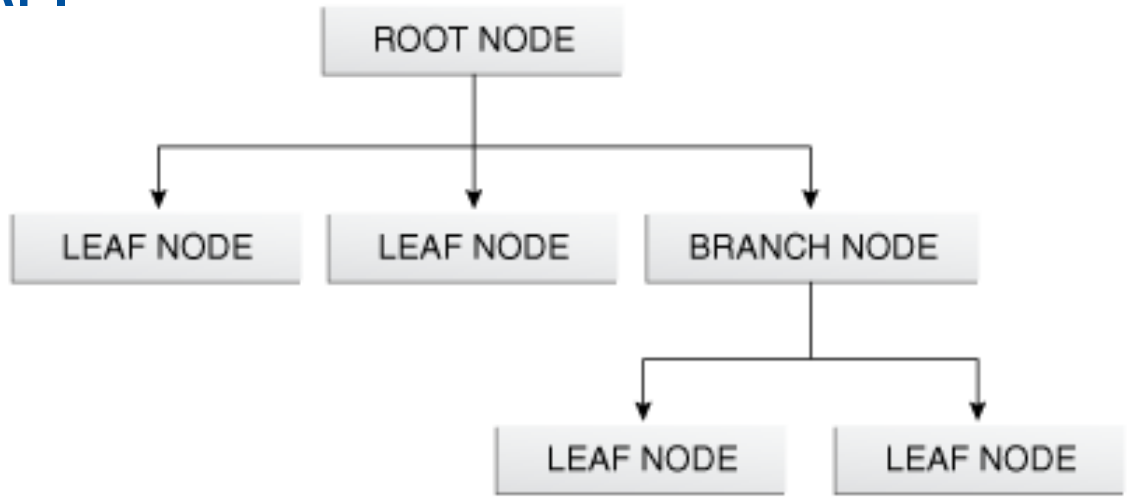

```
public class SimpleSceneGraph extends Application {
    public void start(Stage stage) {
        Rectangle r = new Rectangle(25, 25, 75, 75);
        r.setFill(Color.BLUE);
        Pane root = new Pane(); // root node
        root.getChildren().add(r); // rectangle child of root
        Scene scene = new Scene(root, 125, 125); // top-level container
        stage.setTitle("Simple Scene Graph"); // window title
        stage.setScene(scene); // stage metaphor
        stage.show();
    }
                                    public static void main(String[] args) {
                                        launch(args);
                                    }
                                }
```

Scene Graphs

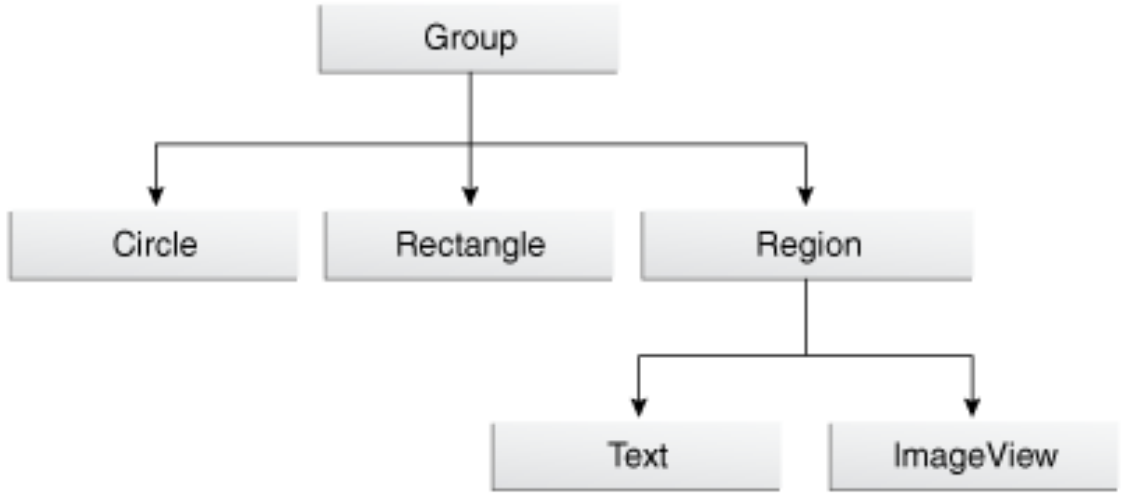
- Scene graph: Set of trees of graphical objects (nodes)
  - Retained mode API: System maintains model of all graphical objects
- Commonly used in video games and 3D graphics
- Allows post-hoc transformations of the structure
  - Size changes
  - Structure changes
  - Animation
  - Effects, etc.
- Creating graphics by modifying the scene graph
- System manages details of graphics rendering
  - Efficiency
  - Less application code

JavaFX Scene Graph API

- Scene graph contains branch nodes and leaf nodes



- Example scene graph



http://docs.oracle.com/javase/8/javafx/scene-graph-tutorial/scenegraph.htm

JavaFX Class Hierarchy of Scene Graph Nodes

- Node
  - Parent
    - Group (groups its children, modification to group is applied to children)
    - Region
      - Control (base class for widgets)
      - Pane (base class for layout panes)
  - ImageView
  - Shape
    - Rectangle, Circle, Line, Text, etc.
  - Shape3D
    - Box, Cylinder, Sphere, MeshView
  - Canvas (has GraphicsContext for immediate mode API)

JavaFX Class Hierarchy of Scene Graph Nodes

- Region (rectangular by default, has background and border)
  - Control (base class for widgets)
    - Accordion, ChoiceBox, ..., TreeView
  - Pane (base class for layout panes)
    - BorderPane
    - HBox, VBox
    - StackPane
    - GridPane
    - FlowPane
    - TextFlow
    - TilePane
    - AnchorPane

Layout

- UI layout: Position and size of UI elements
- Manually laying out widgets is tedious
- Layout panes implement the arrangement of UI elements
- When a window is resized, layout panes reposition and scale the UI elements they contain according to
  - their layout strategy
    - row, column, grid, flow, anchors, etc.
  - the size constraints of the widgets
    - preferred size, min/max size, alignment
    - not all nodes are resizable (Text, Group, shapes)

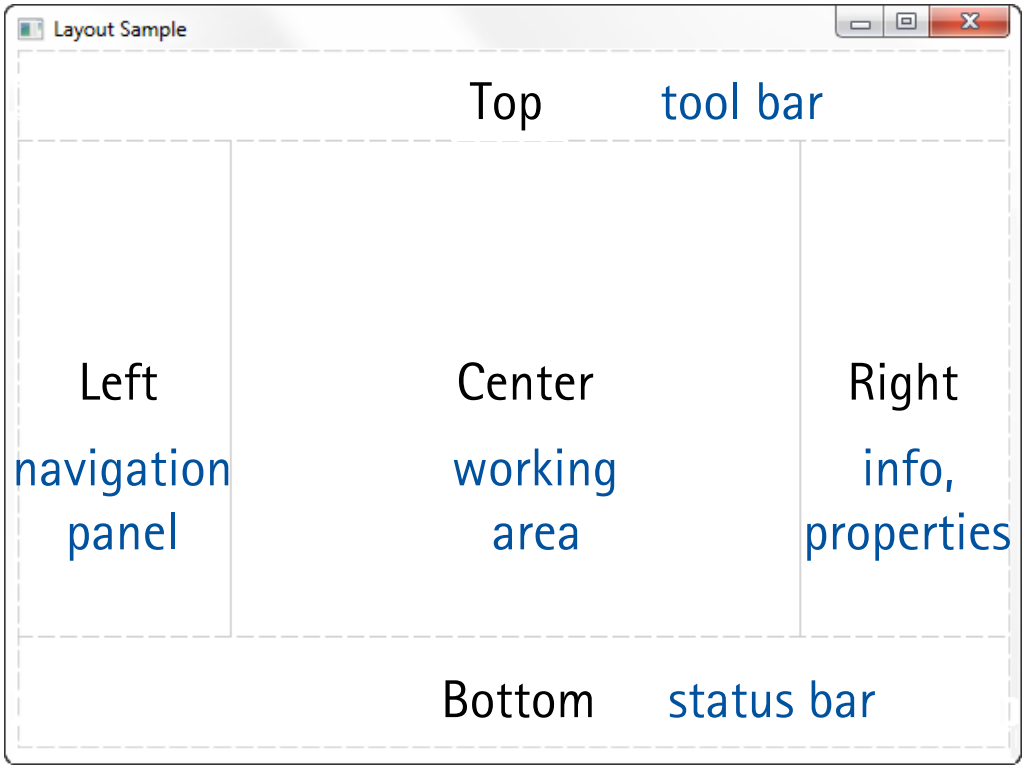
Resizing Range of Nodes

- Preferred width
  - Ideal width (within its min..max range)
- Minimum width
  - Parents should not resize to a smaller width
- Maximum width
  - Parents should not resize to a larger width
- Content-bias of a node
  - Horizontal: height depends on width
  - Vertical: width depends on height
  - None: no content-bias

(analogously for node height)

BorderPane

- Five regions for nodes
- Unused regions disappear
- Typical arrangement
  - Top: Tool bar
  - Bottom: Status bar
  - Left: Navigation panel
  - Right: Info, properties
  - Center: Working area



- If window larger than space needed (for preferred sizes)
  - Top, bottom, left, right: Large enough to wrap content
  - Center: Gets extra space

http://docs.oracle.com/javase/8/javafx/layout-tutorial/builtin_layouts.htm

HBox Pane

- HBox: A row of nodes
- Padding: Distance between edges of HBox pane and nodes
- Spacing: Distance between nodes

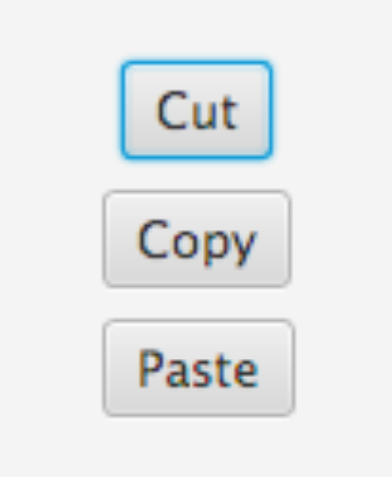
```
HBox hbox = new HBox();
hbox.setPadding(new Insets(15, 12, 15, 12)); // distance top, right, bottom, left
hbox.setSpacing(10); // distance between nodes
Button buttonCurrent = new Button("Current");
buttonCurrent.setPrefSize(100, 20); // ideal size
Button buttonProjected = new Button("Projected");
buttonProjected.setPrefSize(100, 20); // ideal size
hbox.getChildren().addAll(buttonCurrent, buttonProjected);
```

http://docs.oracle.com/javase/8/javafx/layout-tutorial/builtin_layouts.htm

VBox Pane

- VBox: A column of nodes
- Example

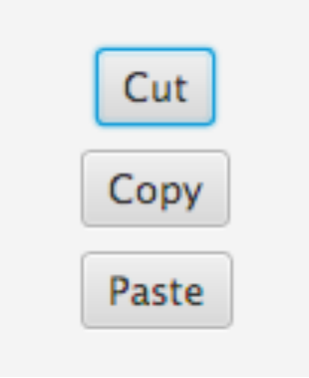
```
VBox vbox = new VBox(8); // spacing = 8
Button b1 = new Button("Cut");
Button b2 = new Button("Copy");
Button b3 = new Button("Paste");
vbox.getChildren().addAll(b1, b2, b3);
vbox.setAlignment(Pos.CENTER);
```



http://docs.oracle.com/javase/8/javafx/layout-tutorial/builtin_layouts.htm

Resizing Range of a VBox

	width	height
minimum	left inset + largest of the children's min widths + right inset	top inset + sum of the children's min heights + bottom inset + sum of spacing between children
preferred	left inset + largest of the children's pref widths + right inset	top inset + sum of the children's pref heights + bottom inset + sum of spacing between children
maximum	Double.MAX_VALUE	Double.MAX_VALUE

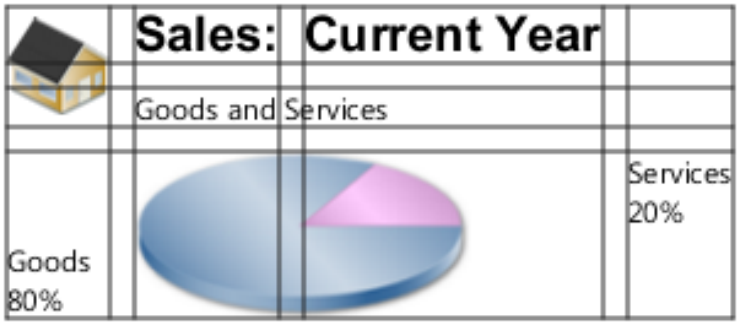


StackPane

- Children stacked on top of each other
- From first child at bottom to last child at top
- Children will be resized to fill content area
- If not sizable to content area, will be centered by default
- Example
  - Text on top of rectangle with gradient:

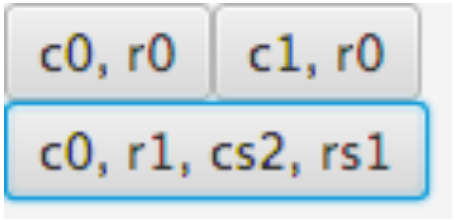
http://docs.oracle.com/javase/8/javafx/layout-tutorial/builtin_layouts.htm

GridPane



http://docs.oracle.com/javase/8/javafx/layout-tutorial/builtin_layouts.htm

- Flexible grid of rows and columns
- Grid will automatically expand as new children are added
- Child may be placed at any row/column
- Child may span multiple rows/columns
- Children may overlap
- GridPanes are sized to fit their content (pref. size)
- Example



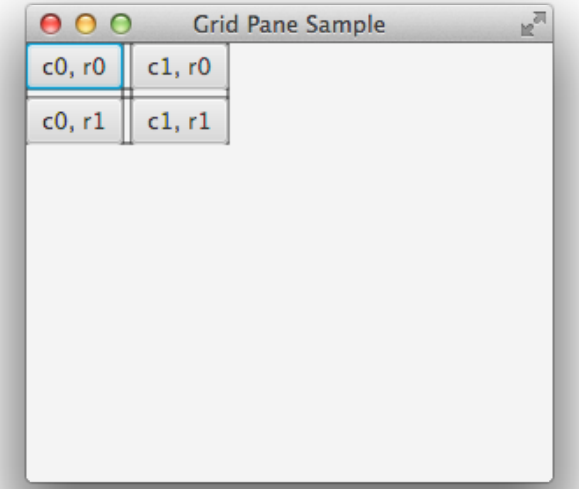

```
GridPane gp = new GridPane();
gp.add(new Button("c0, r0"), 0, 0);  // column=0, row=0
gp.add(new Button("c1, r0"), 1, 0);  // column=1, row=0
gp.add(new Button("c0, r1, cs2, rs1"), 0, 1, 2, 1); // columnspan=2, rowspan=1
```

Layout Constraints of Nodes within a Grid

- column index, row index
- column span, row span
- horizontal alignment (left, center, right)
- vertical alignment (top, center, bottom, baseline)
- horizontal/vertical grow priority (always, never, sometimes)
  - If the grid has more (or less) horiz./vert. space available than needed, priority determines how that space is distributed among grid nodes
- fillWidth, fillHeight
  - Determines whether the node is expanded beyond its preferred size when the grid column/row has extra space
- margin (space around a node)

Layout Constraints of Grid Nodes: Example

```
GridPane gp = new GridPane();
gp.setHgap(5); // horizontal gap between nodes
gp.setVgap(5); // vertical gap between nodes
gp.setGridLinesVisible(true); // invisible by default
gp.add(new Button("c0, r0"), 0, 0);
gp.add(new Button("c1, r0"), 1, 0);
gp.add(new Button("c0, r1"), 0, 1);
gp.add(new Button("c1, r1"), 1, 1);
```



default layout constraints

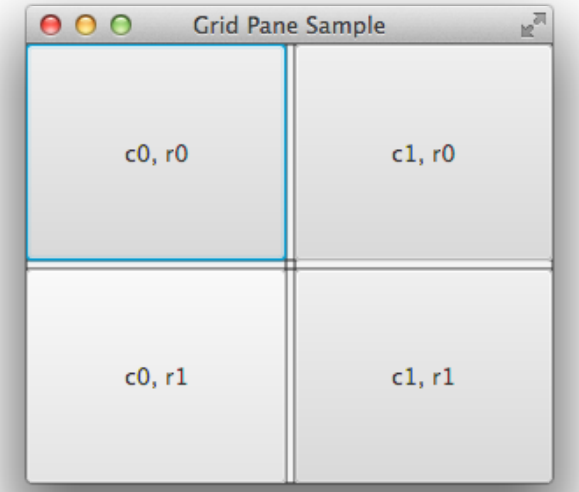
Layout Constraints of Grid Nodes: Example

```
...
for (Node node : gp.getChildren()) {
    if (node instanceof Button) {
        Button b = (Button) node;
        b.setMaxSize(Double.MAX_VALUE, Double.MAX_VALUE);
        GridPane.setHgrow(b, Priority.ALWAYS);
        GridPane.setVgrow(b, Priority.ALWAYS);
    }
}
```

buttons normally don't expand beyond their preferred size, (i.e. pref = max)

refers to growth of grid cell

grid cells may have different priorities on how much available space the may take



GridPane Column and Row Constraints

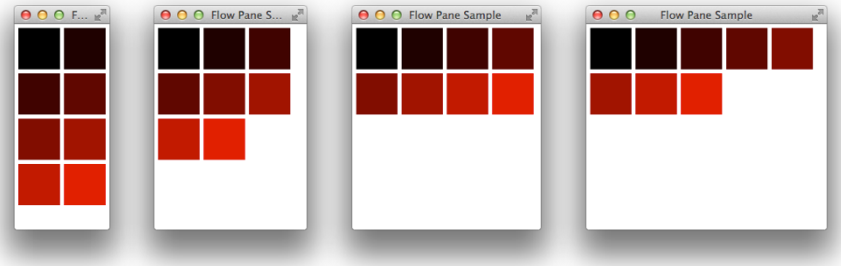
- ColumnConstraints and RowConstraints
  for specifying constraints for a row / column
  - Node constraints (except hgrow/vgrow) have priority over these
- Example: first column gets extra width, second has fixed width

```
ColumnConstraints cc1 = new ColumnConstraints(
        /*minWidth*/100, /*prefWidth*/100, /*maxWidth*/Double.MAX_VALUE,
        /*hgrow*/Priority.ALWAYS,
        /*halign*/HPos.LEFT,
        /*fillWidth*/true);
ColumnConstraints cc2 = new ColumnConstraints(
        /*fixed width*/100);
gridPane.getColumnConstraints().addAll(cc1, cc2);
```

FlowPane

- Nodes flow horizontally (rows) or vertically (columns) and wrap at the pane boundary
  - Similar to line wrap in a text editor
- Example

```
FlowPane flow = new FlowPane();
flow.setPadding(new Insets(5, 5, 5, 5));
flow.setVgap(5); flow.setHgap(5);
Rectangle[] rs = new Rectangle[8];
for (int i = 0; i < rs.length; i++) {
        Color c = new Color((double) i / rs.length, 0, 0, 1);
        rs[i] = new Rectangle(50, 50, c);
        flow.getChildren().add(rs[i]);
}
```

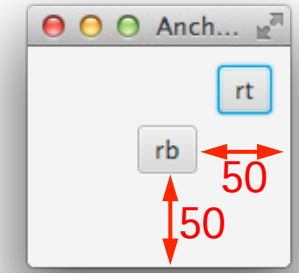


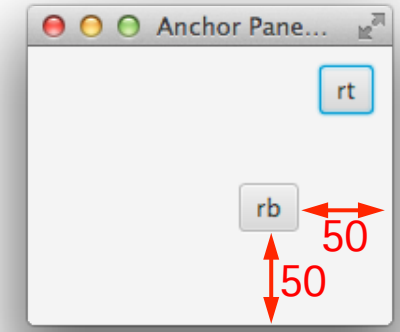
see also: TextFlow, TilePane

AnchorPane

- Anchor edges of child nodes at offset from pane's edges
- Anchor constraints on child nodes
  - topAnchor: distance from pane's top to child's top edge
  - left-, bottom-, rightAnchor analogously
- Example

```
AnchorPane anchorpane = new AnchorPane();
Button rt = new Button("rt");
Button rb = new Button("rb");
AnchorPane.setRightAnchor(rt, 10);
AnchorPane.setTopAnchor(rt, 10);
AnchorPane.setRightAnchor(rb, 50);
AnchorPane.setBottomAnchor(rb, 50);
anchorpane.getChildren().addAll(rt, rb);
```

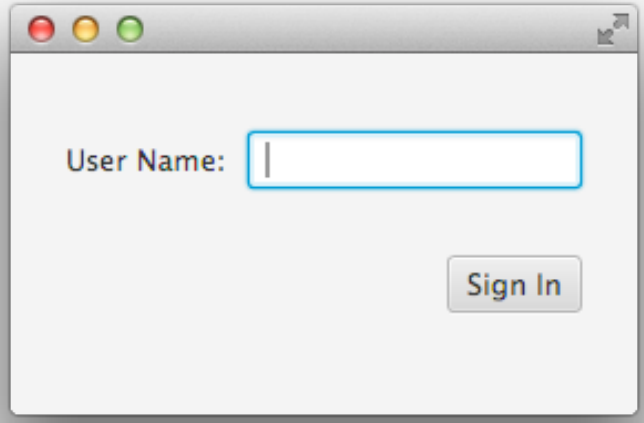




Declarative Layout Definition with FXML

- Allows for constructing scene graphs declaratively
- XML structure corresponds to UI structure
- More clearly shows UI structure than procedural code
- For details, see http://docs.oracle.com/javase/8/javafx/api/javafx/fxml/doc-files/introduction_to_fxml.html

FXML Example: MyLayout.fxml




```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.geometry.Insets?>
...
<GridPane alignment="center" hgap="10" vgap="10" fx:controller="sample.MyController"...>
    <padding><Insets top="25" right="25" bottom="10" left="25"/></padding>
    <Label text="User Name:" .../>
    <TextField .../>
    <HBox ...>
        <Button text="Sign In" onAction="#handleButtonAction"/>
    </HBox>
    <Text fx:id="message"...>
</GridPane>
```

MVC controller class

property

child node

link button to event handler in Java

set text from Java

FXML Example: Application Class

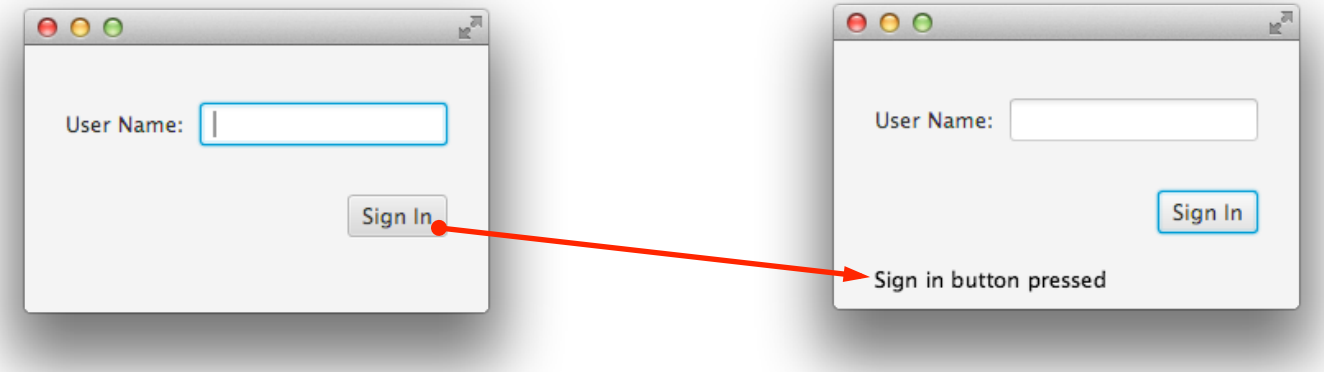
```
public class FXMLSample extends Application {
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(
            getClass().getResource("MyLayout.fxml"));
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

FXML Example: Controller Class

```
public class MyController {

    public Text message;

    public void handleButtonAction(ActionEvent event) {
        message.setText("Sign in button pressed");
    }

}
```

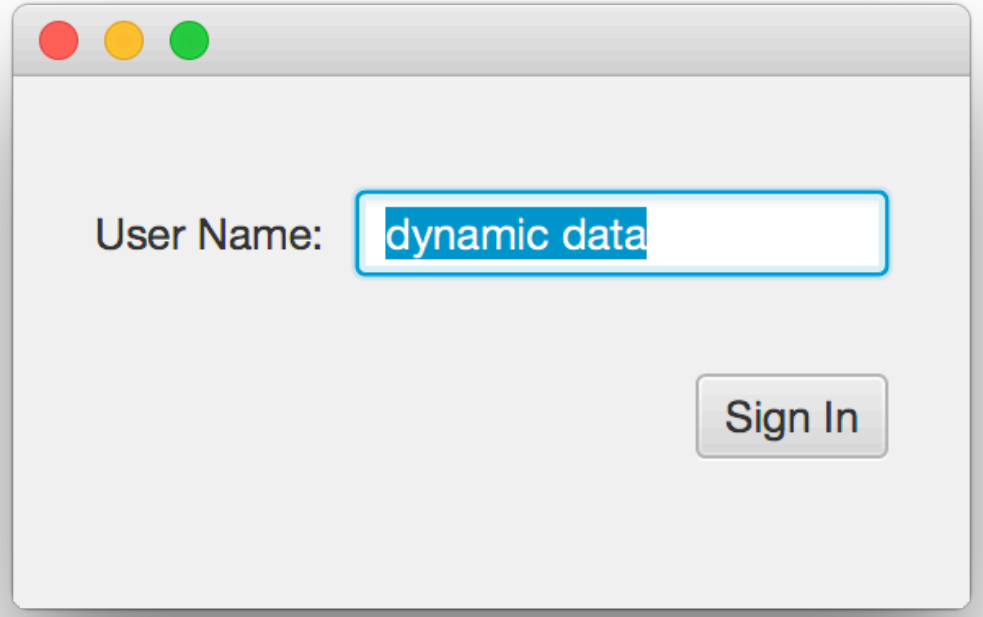
<Text fx:id="message".../>

<Button ... onAction="#handleButtonAction"/>



Initializing the Controller with Dynamic Content

```
<GridPane xmlns:fx="http://javafx.com/fxml" ...>
    ...
    <TextField fx:id="field" GridPane.columnIndex="1" GridPane.rowIndex="1"/>
    ...
</GridPane>
```



Initializing the Controller with Dynamic Content

- Controller class

```
public class MyController {
    public TextField field; // in XML file: fx:id="field"
    private String fieldInit;
    public MyController(String fi) {
        fieldInit = fi;
    }
    public void initialize() { // will be called to initialize XML elements
        field.setText(fieldInit);
    }
}
```

Initializing the Controller with Dynamic Content

- Application class

```
public class FXMLSample extends Application {
    public void start(Stage stage) throws Exception {
        URL location = getClass().getResource("MyLayout.fxml");
        FXMLLoader loader = new FXMLLoader(location);
        loader.setController(new MyController("dynamic data"));
        Pane root = (Pane) loader.load();
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }
    ...
}
```

GUI Editor: JavaFX Scene Builder