

CISC 839

Software Engineering of Usable Computing Systems

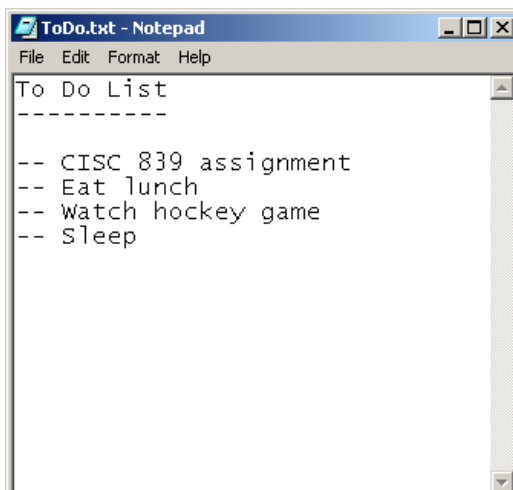
T.C. Nicholas Graham

<http://www.cs.queensu.ca/~graham/cisc836>

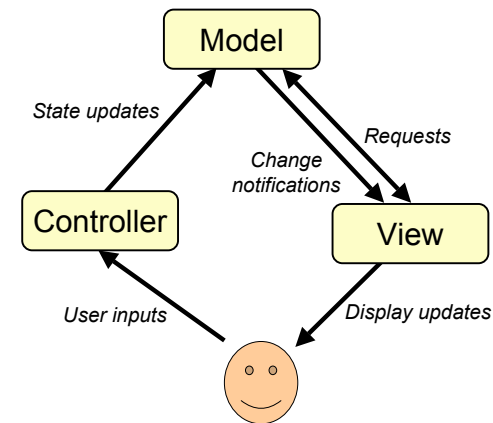
Higher-Level Architectures for Interactive Systems

- ⌘ Callback-based architectures are error-prone
- ⌘ Higher-level architectures structure user interface more closely to its design
 - ☒ Model-View-Controller [Krasner and Pope]
 - ☒ PAC [Coutaz et al.]
 - ☒ PAC-Amodeus [Nigay]

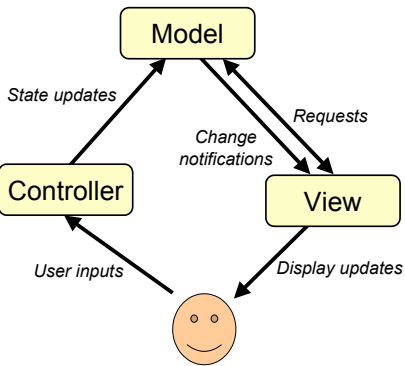
Example: Notepad



MVC Architecture



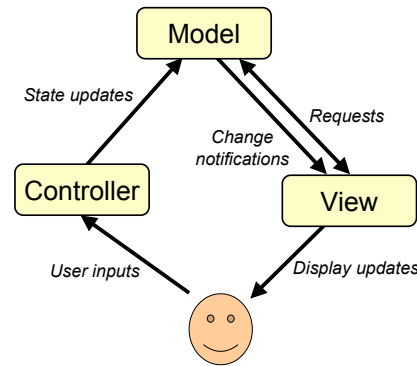
Model-View-Controller



⌘ Model

- ☒ Represents underlying state of application
- ☒ Presents an abstract representation of the user interface's contents
- ☒ Notepad: Model is a string representing the content of the text editor

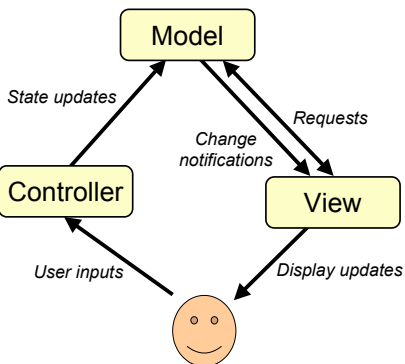
Model-View-Controller



⌘ View

- ☒ Responsible for updating the display
- ☒ Model notifies View of changes to Model
- ☒ View makes requests to Model for new Model state, updates display

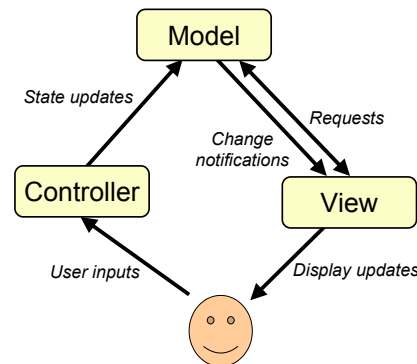
Model-View-Controller



⌘ View

- ☒ Notepad: View notified when document changes
 - ☒ e.g., new text entered, text deleted, ...
- ☒ View requests updated text from Model, modifies the text on the display

Model-View-Controller



⌘ Controller

- ☒ Responsible for processing user inputs
 - ☒ I.e., mouse and keyboard events
- ☒ Updates the state of the Model
- ☒ Notepad: Controller processes key clicks, sends modifications to the Model
 - ☒ New text, delete key, etc.

Benefits of MVC

⌘ Separation of concerns

- ☑ Input and output are completely independent
 - ☑ Implies changes in input mechanisms don't affect output; changes in output format don't affect input

⌘ No callbacks

MVC Deployments

⌘ Smalltalk-80

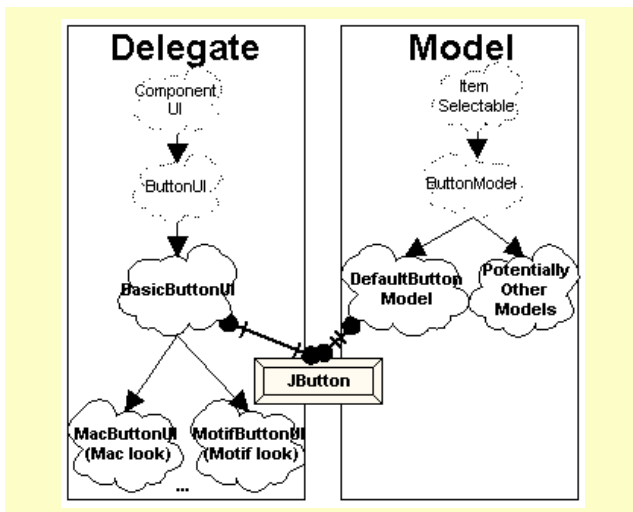
⌘ Java Swing

- ☑ Model/Delegate

⌘ Microsoft Foundation Classes

- ☑ MFC Document-View architecture

Java Swing Model/Delegate



Source: <http://developer.java.sun.com/developer/onlineTraining/GUI/Swing2/shortcourse.html#JFCMVC>

Java Swing Model/Delegate

⌘ Model = MVC Model

⌘ Delegate = MVC View + Controller

MFC Document/View

⌘ Document = MVC Model

⌘ View = MVC View + Controller

☒ View uses event handlers for input

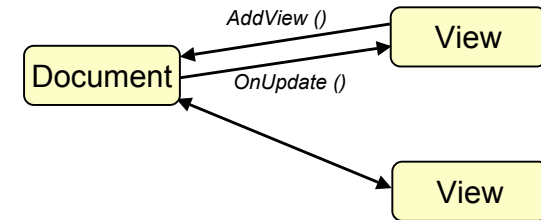
☒ See examples from week 6

☒ Standard classes:

☒ CView = base class for View

☒ CDocument = base class for Document

MFC Document/View



Document Operations

AddView	Attaches a view to the document.
GetDocTemplate	Returns a pointer to the document template that describes the type of the document.
GetFirstViewPosition	Returns the position of the first in the list of views; used to begin iteration.
GetNextView	Iterates through the list of views associated with the document.
GetPathName	Returns the path of the document's data file.
GetTitle	Returns the document's title.
IsModified	Indicates whether the document has been modified since it was last saved.
RemoveView	Detaches a view from the document.
SetModifiedFlag	Sets a flag indicating that you have modified the document since it was last saved.
SetPathName	Sets the path of the data file used by the document.
SetTitle	Sets the document's title.
UpdateAllViews	Notifies all views that document has been modified.

CView Operations

GetDocument	Returns the document associated with the view.
OnDraw	Called to render an image of the document for screen display, printing, or print preview. Implementation required.
OnUpdate	Called to notify a view that its document has been modified.

The PAC Architecture

The PAC Architecture Style

- ⌘ Supporting development of user interfaces
- ⌘ *Conceptual* architecture that can be mapped to multiple *concrete* architectures
- ⌘ Agent-based
 - ☑ Program decomposed into a set of agents, each responsible for some part of the task of interacting with the user

Example: PAC Architecture for a Set of Buttons

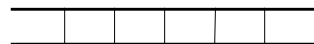
- ⌘ Assume an interface has a set of buttons controlling navigation

- ☑ "Back" means return to previous page in history, "Forward" means go forward to last page we moved back from (if there is one)



Button

List of pages visited



Current page

- "Back" decrements current page counter
- "Forward" increments current page counter
- If current page is front of list, "Back" button is disabled
- If current page is end of list, "Forward" button is disabled
- Going to a new page enqueues that page after the current

Starting with Just a Button...

- ⌘ Buttons have
 - ☑ A *label* (e.g. "Back", "Forward")
 - ☑ A *state* (disabled, enabled)
- ⌘ Buttons have local behaviour
 - ☑ "Rollover" - become highlighted when mouse moves over them
 - ☑ Become depressed following mouse down
 - ☑ Become released following mouse up



Enabled button



Disabled button



"Rolled over" button



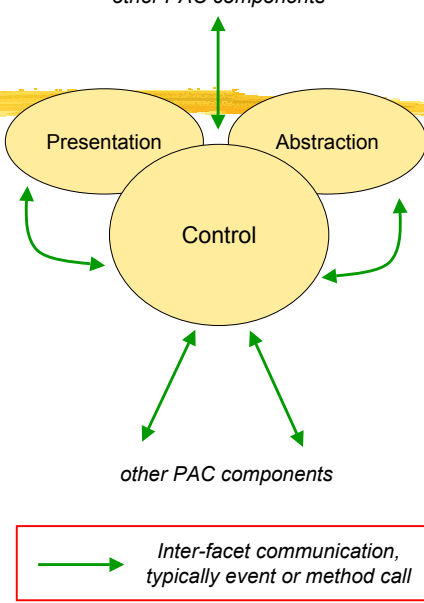
Depressed button
(Couldn't make accurate picture of this one)

Button Abstraction

- ⌘ A button's state is therefore
 - ☒ Label (String value)
 - ☒ Is Enabled? (Boolean value)
 - ☒ Is "rolled over"? (Boolean value)
 - ☒ Is depressed? (Boolean value)
- ⌘ Therefore, the state of a button is captured in three Boolean values and a string value
- ⌘ In PAC, this state is called the button's *Abstraction*

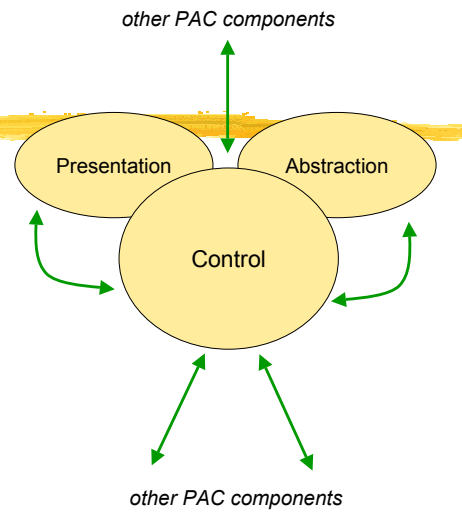
PAC Components

- ⌘ PAC architectures constructed from a set of components
- ⌘ Each component composed of three *facets*
 - ☒ *Presentation*: handles user input and redraws the display
 - ☒ *Abstraction*: underlying state of interactive entity
 - ☒ *Control*: mediates communication between Presentation, Abstraction and other PAC components



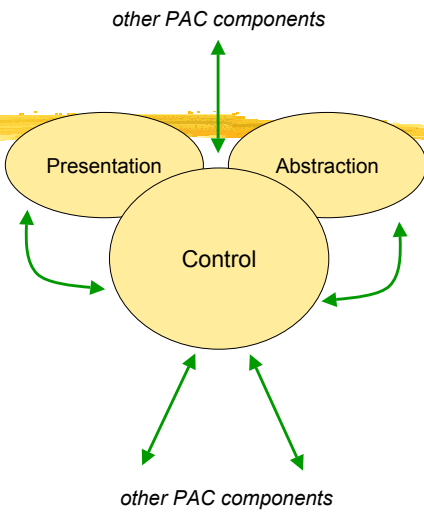
PAC Button

- ⌘ Abstraction
 - ☒ Java object containing methods to set/get
 - ☒ Button label
 - ☒ Enabled/disabled state
 - ☒ Rolled-over state
 - ☒ Depressed/released state



PAC Button

- ⌘ Presentation
 - ☒ Accepts mouse input
 - ☒ mouse down
 - If button enabled, depresses button
 - ☒ mouse up
 - If button depressed, releases button
 - ☒ mouse enter
 - If active, puts button in rollover state
 - ☒ mouse leave
 - If rolled over, removes rollover state

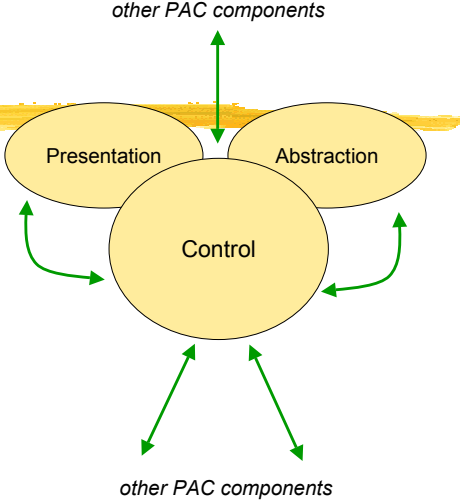


PAC Button

⌘ Presentation

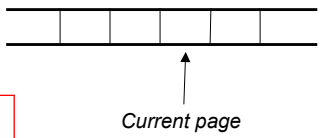
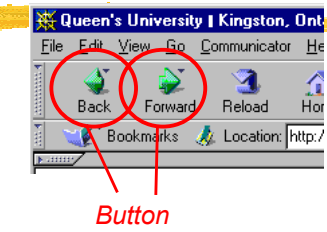
- ⌘ Responsible for redrawing button
 - ⌘ When changes in enabling, depressed/released, rolled over, redraws with appropriate animation, GIF image

Question: Why doesn't input part of Presentation simply handle redrawing?



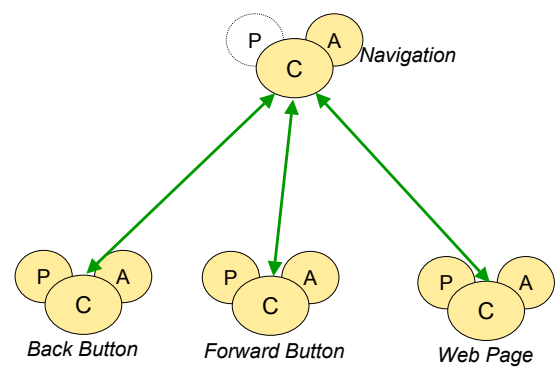
Building an Architecture with PAC Components

- ⌘ Require a PAC component for each button
- ⌘ Require a component for navigation
- ⌘ Require components for the rest of the browser, such as the display of the web page



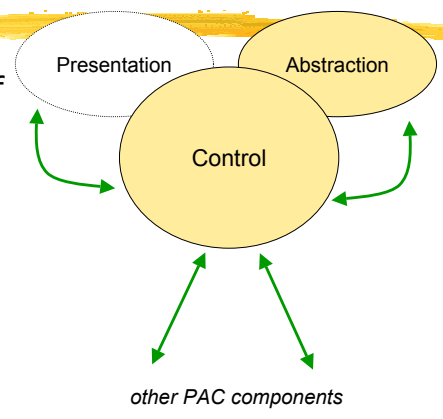
- "Back" decrements current page counter
- "Forward" increments current page counter
- If current page is front of list, "Back" button is disabled
- If current page is end of list, "Forward button is disabled
- Going to a new page enqueues that page after the current page, sets current page to that page

PAC Architecture

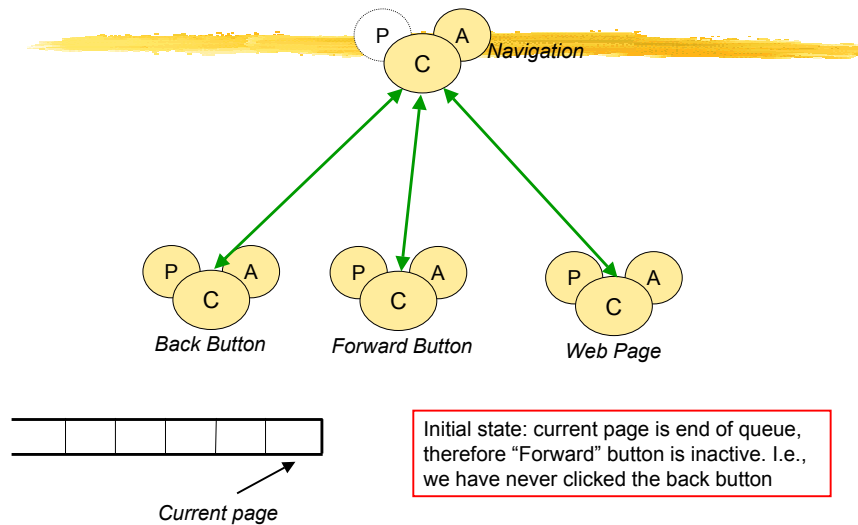


Navigation Component

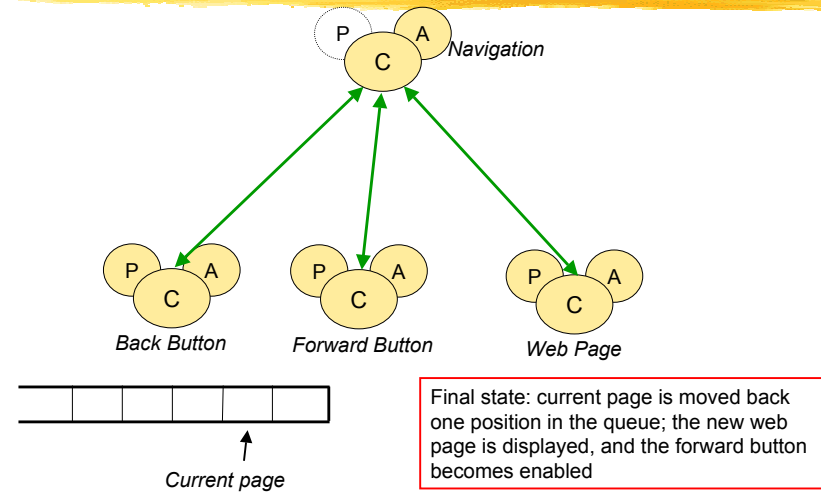
- ⌘ Abstraction contains queue of pages, current page pointer
- ⌘ If there are pages before current page, enables back button
- ⌘ If there are pages after current page, enables forward button
- ⌘ Actions from buttons manipulate current page
- ⌘ No Presentation facet required



Scenario: Go Back One Page

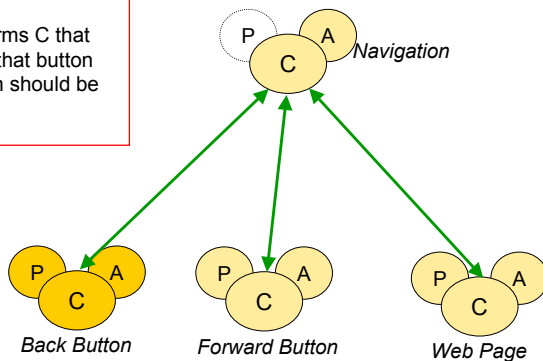


Scenario: Go Back One Page



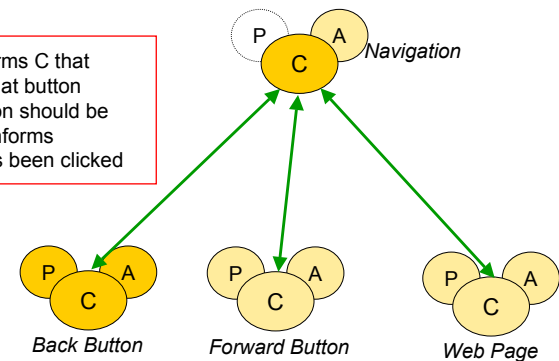
Scenario: Go Back One Page

1. User clicks mouse down; P informs C that mouse clicked down; C informs A that button depressed; C informs P that button should be redrawn; P redraws button



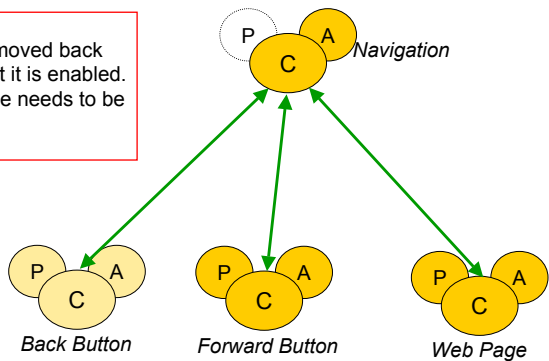
Scenario: Go Back One Page

2. User releases mouse; P informs C that mouse released; C informs A that button released; C informs P that button should be redrawn; P redraws button; C informs Navigation that Back button has been clicked



Scenario: Go Back One Page

3. C informs A that current page moved back one. C informs forward button that it is enabled. C informs web page that new page needs to be drawn.

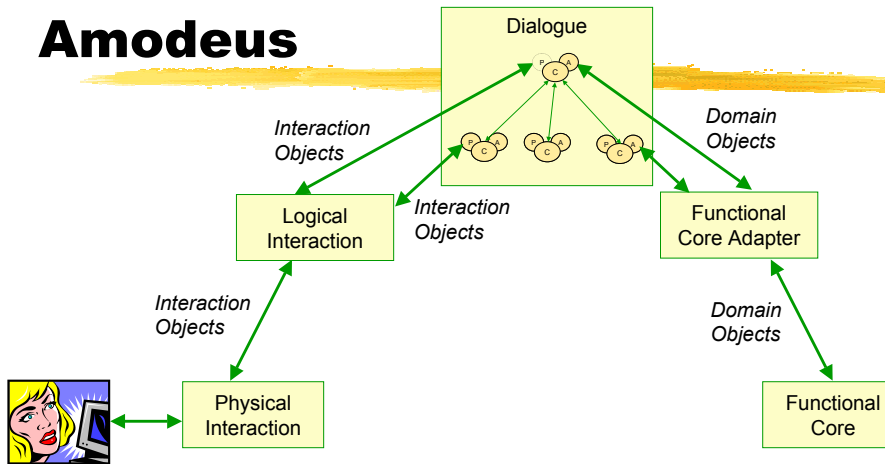


PAC-Amodeus

PAC Amodeus

- ⌘ Integrate ARCH and PAC
 - ⌘ Dialogue component implemented by PAC agents
 - ⌘ Presentation facets interact with Logical Interaction component
 - ⌘ Abstraction facets interact with Functional Core Adapter

PAC-Amodeus



Constraint-Based Architectures

Examples

⌘ Garnet

⌘ Myers 89

⌘ RendezVous

⌘ Hill 92

⌘ CIP

⌘ Borning, Freeman-Benson '92

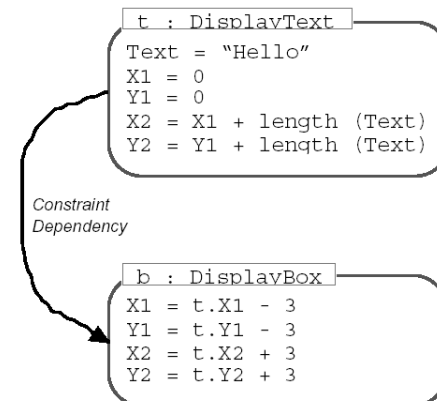
Constraints

⌘ Constraints allow the values of one object to depend on the values of fields of other objects

⌘ Changes in field values propagated via constraints

⌘ No need for callbacks or notifications

Display Objects and Text



Output:

Hello

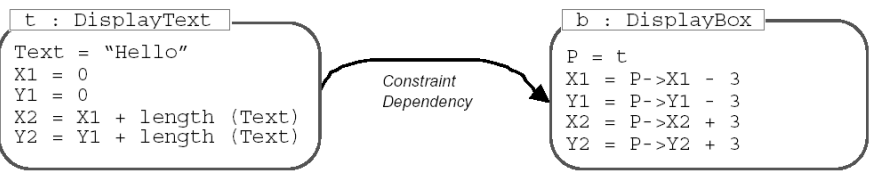
Modify:

t.Text := "Guten Tag"

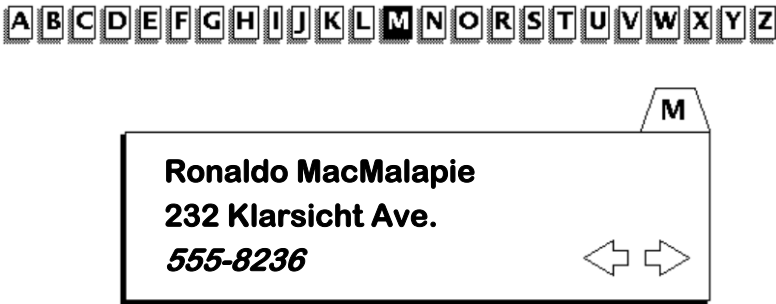
New Output:

Guten Tag

Pointer Variables



Example Interface



Constraints in this Interface

- ✂ Clicking on a letter button changes the current card, changes the letter in the tab
- ✂ Editing the last name may change the highlighted letter button, letter in the tab

