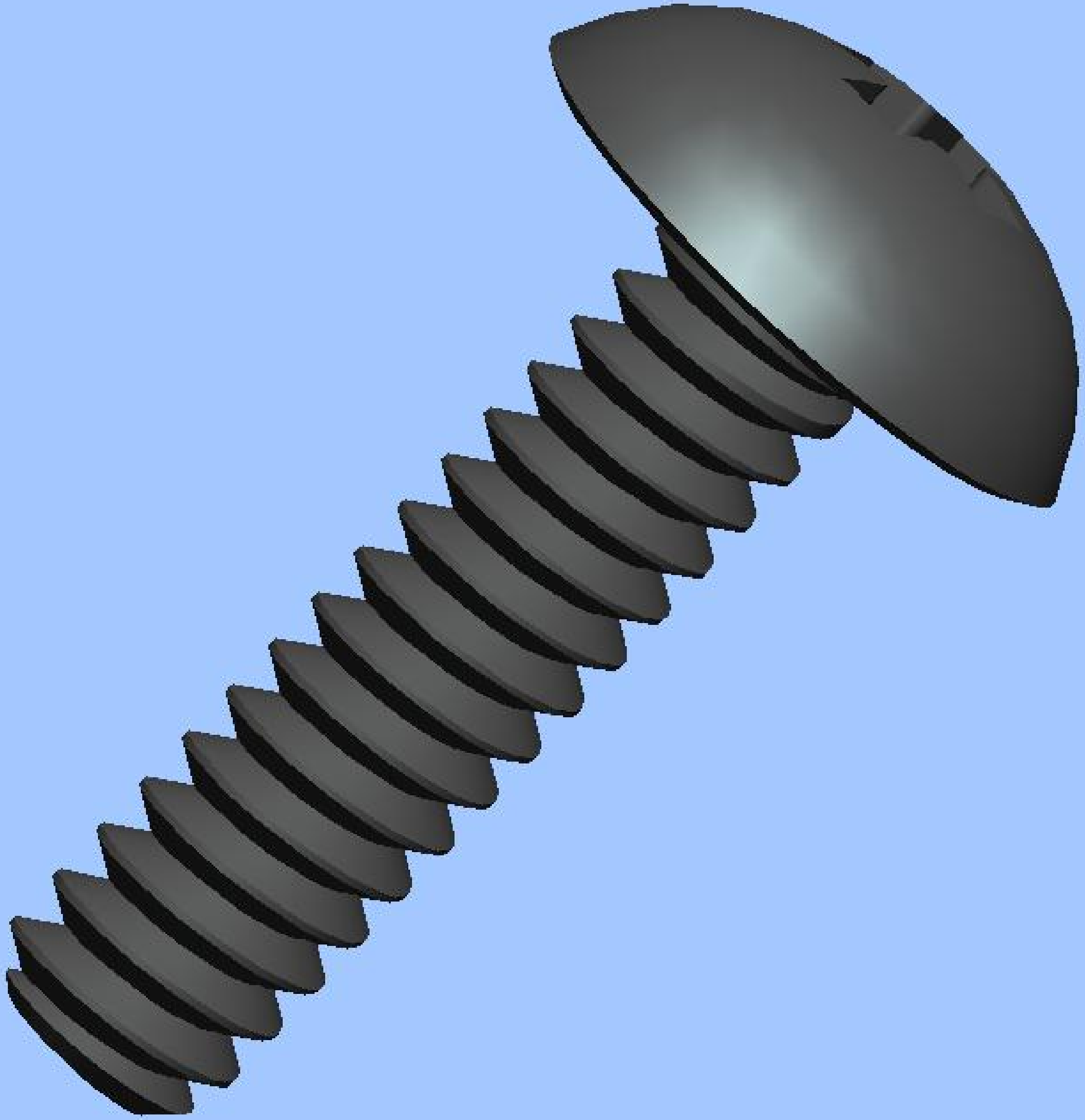
Software Tools

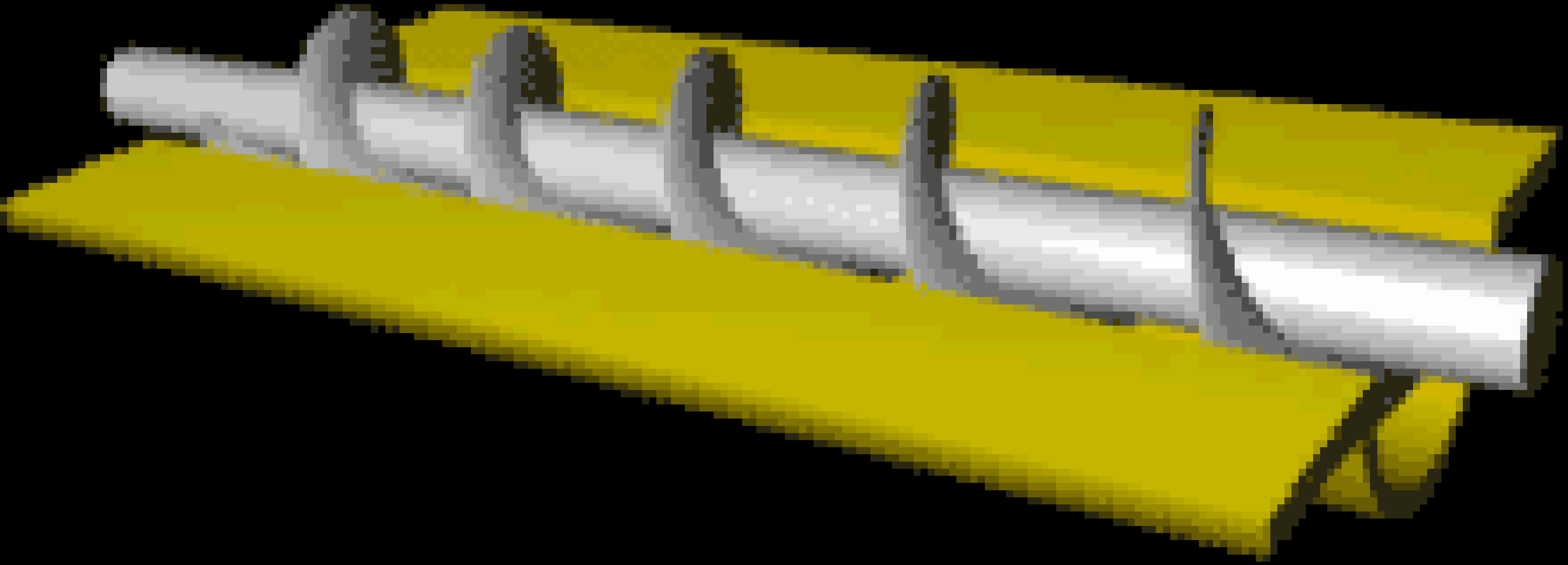
Scott Klemmer

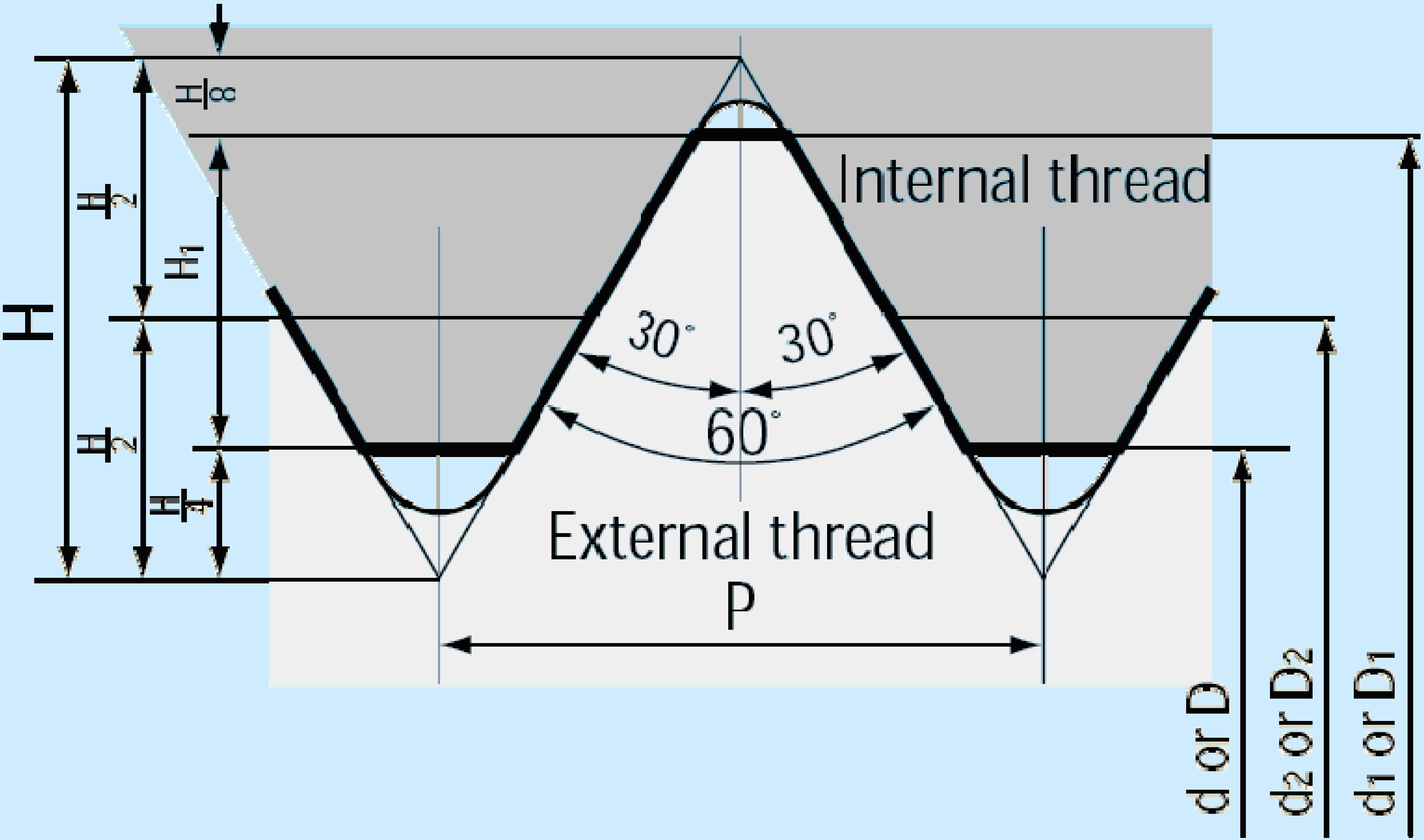
TAs: Marcello Bastea-Forte, Joel Brandt, Neil Patel, Leslie Wu, Mike Cammarano

04 December 2007

http://cs147.stanford.edu

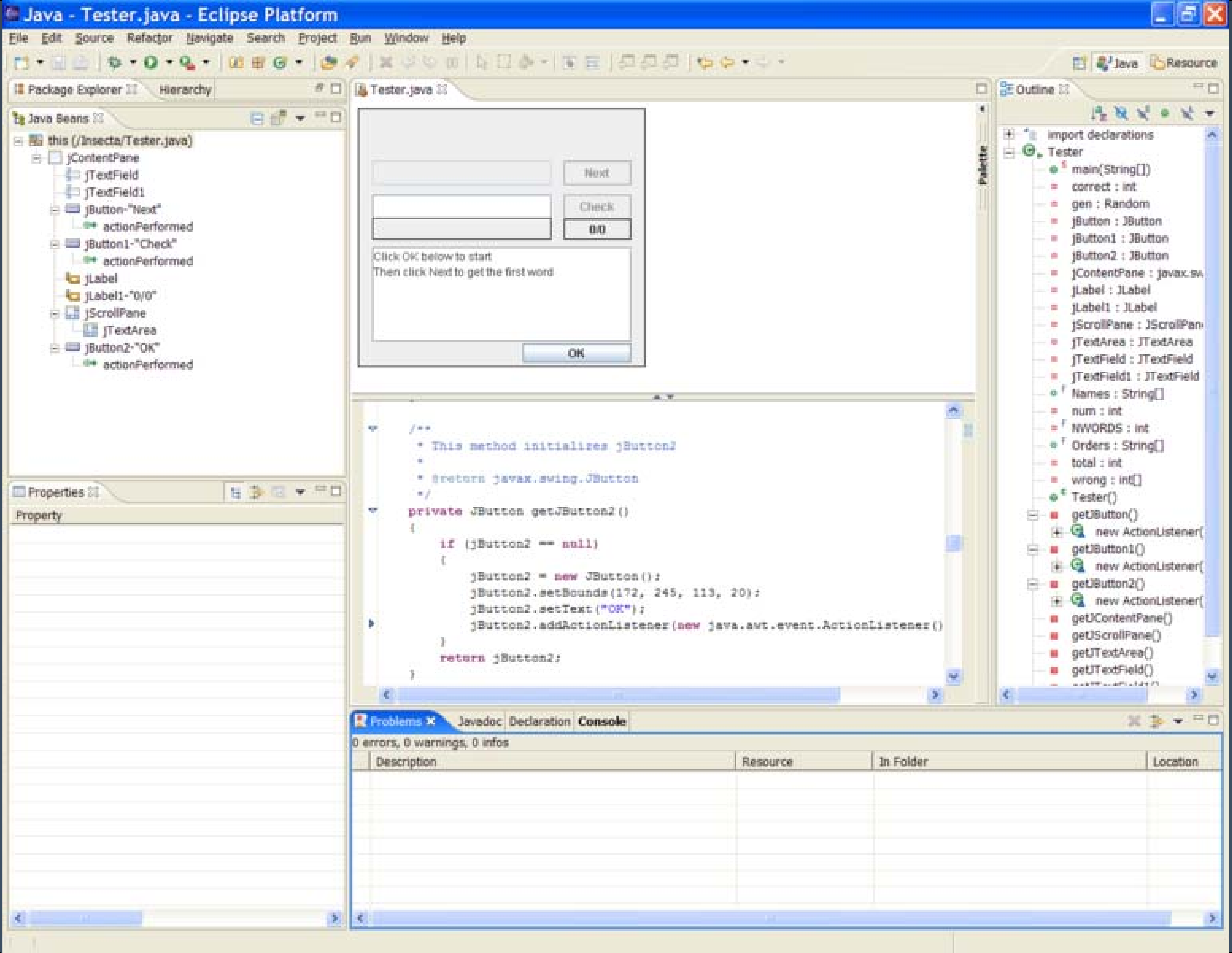





H/8
H/2
H₁
H
H/2
H/4
30°
30°
60°
Internal thread
External thread
P
d or D
d2 or D2
d1 or D1

Developers are People Too

Tools are Interfaces Too



Click OK below to start
Then click Next to get the first word

```
    /**
     * This method initializes jButton2
     *
     * @return javax.swing.JButton
     */
    private JButton getJButton2()
    {
        if (jButton2 == null)
        {
            jButton2 = new JButton();
            jButton2.setBounds(172, 245, 113, 20);
            jButton2.setText("OK");
            jButton2.addActionListener(new java.awt.event.ActionListener()
        }
        return jButton2;
    }
```

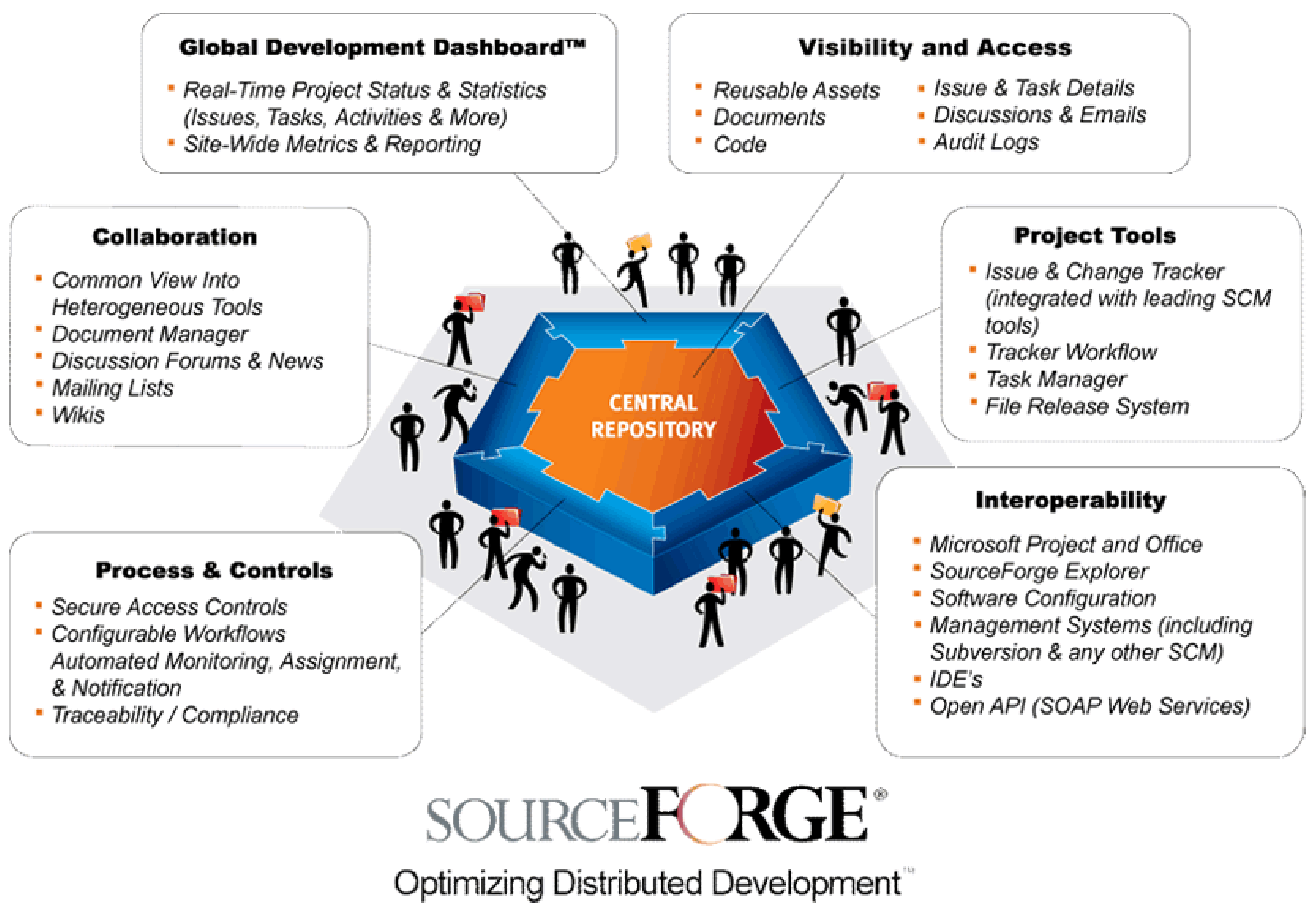
0 errors, 0 warnings, 0 infos

Example: refactoring support

- A code refactoring is any change to a computer program's code which improves its readability or simplifies its structure without changing its results.

Source: Wikipedia

And workflow support



Global Development Dashboard™

- Real-Time Project Status & Statistics (Issues, Tasks, Activities & More)
- Site-Wide Metrics & Reporting

Visibility and Access

- Reusable Assets
- Documents
- Code
- Issue & Task Details
- Discussions & Emails
- Audit Logs

Collaboration

- Common View Into Heterogeneous Tools
- Document Manager
- Discussion Forums & News
- Mailing Lists
- Wikis

Project Tools

- Issue & Change Tracker (integrated with leading SCM tools)
- Tracker Workflow
- Task Manager
- File Release System

Process & Controls

- Secure Access Controls
- Configurable Workflows Automated Monitoring, Assignment, & Notification
- Traceability / Compliance

Interoperability

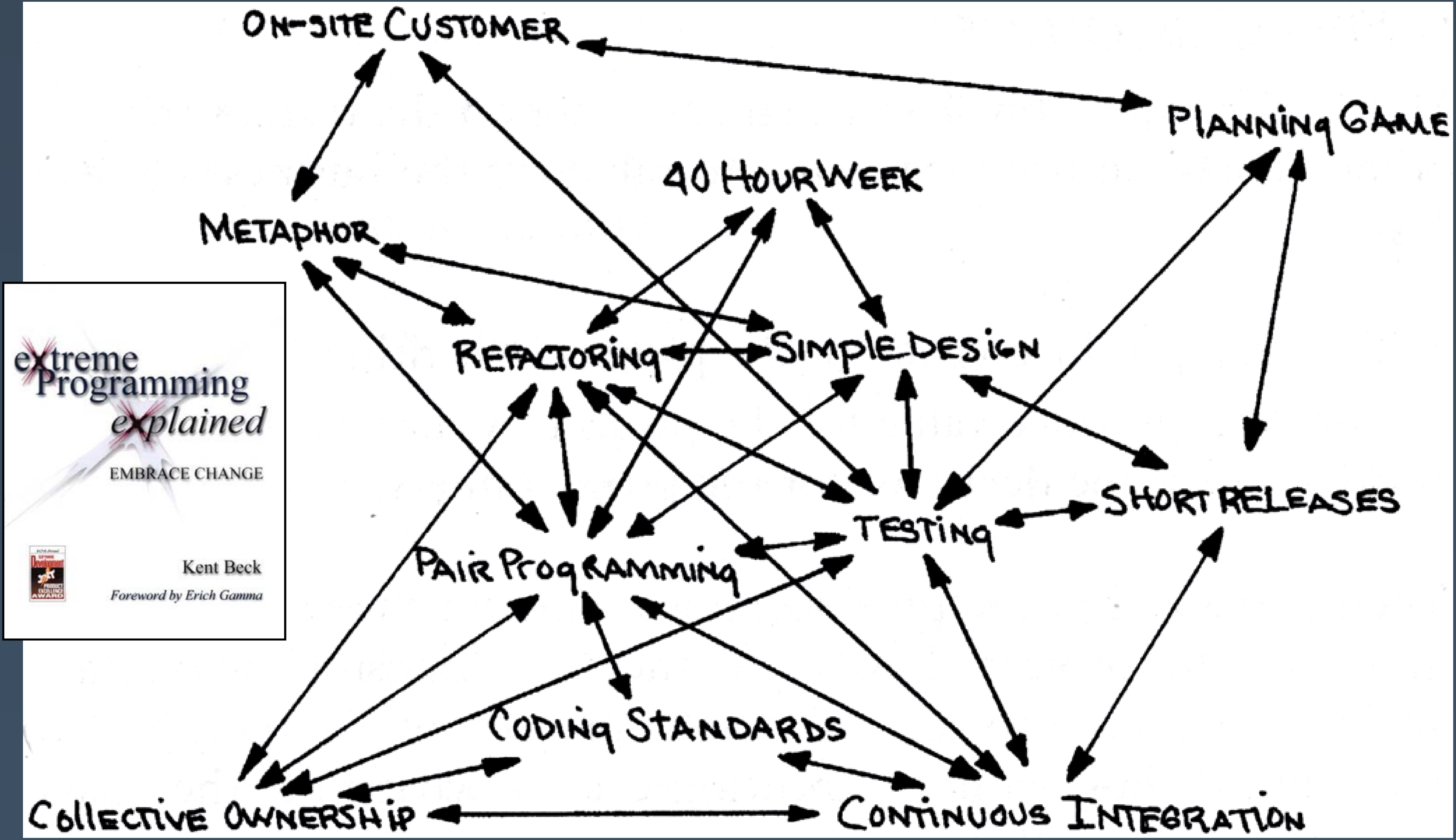
- Microsoft Project and Office
- SourceForge Explorer
- Software Configuration
- Management Systems (including Subversion & any other SCM)
- IDE's
- Open API (SOAP Web Services)

SOURCEFORGE®

Optimizing Distributed Development™

Source: Wikipedia, collab.net

Extreme Programming



Why use toolkits?

- Code reuse saves programmer time
  - 50% of code is for the GUI [Myers & Rosson, CHI ’92]
- Consistent look & feel across apps
- Easier to modify and iterate the UI
- Make UI development accessible to more people
  - Non-artists
  - Non-programmers???

What should tools do?

- Help **design** the interface given a specification of the tasks.
- Help **implement** the interface given a design.
- Help **evaluate** the interface after it is designed and propose improvements, or at least provide information to allow the designer to evaluate the interface.
- Create easy-to-use interfaces.
- Allow the designer to rapidly investigate different designs.
- Allow non-programmers to design and implement user interfaces.
- Provide portability across different machines and devices.
- Be easy to use themselves.

Toolkits

- A collection of widgets
  - Menus, scroll bars, text entry fields, buttons, etc.
- Toolkits help with programming
- Help maintain consistency among UIs
  - Key insight of Macintosh toolbox

è Path of least resistance translates into getting programmers to do the right thing

- Successful partially because address common, low-level features for all UIs

  è Address the useful & important aspects of UIs

Why Tools?

- **The quality of the interfaces will be higher.** This is because:
  - Designs can be rapidly prototyped and implemented, possibly even before the application code is written.
  - It is easier to incorporate changes discovered through user testing.
  - More effort can be expended on the tool than may be practical on any single user interface since the tool will be used with many different applications.
  - Different applications are more likely to have consistent user interfaces if they are created using the same user interface tool.
  - A UI tool will make it easier for a variety of specialists to be involved in designing the user interface.

Why Tools, cont.

- **The user interface code will be easier and more economical to create and maintain**. This is because:
  - There will be less code to write, because much is supplied by the tools.
  - There will be better modularization due to the separation of the user interface component from the application.
  - The level of expertise of the interface designers and implementers might be able to be lower, because the tools hide much of the complexities of the underlying system.
  - The reliability of the user interface may be higher, since the code for the user interface is created automatically from a higher level specification.
  - It may be easier to port an application to different hardware and software environments since the device dependencies are isolated in the user interface tool.

Success of Tools

- Today’s tools are highly successful
  - Window Managers, Toolkits, Interface Builders ubiquitous
  - Most software built using them
  - Are based on many years of HCI research

    Brad A. Myers. “A Brief History of Human Computer Interaction Technology.”
    ACM interactions. Vol. 5, no. 2, March, 1998. pp. 44-54.

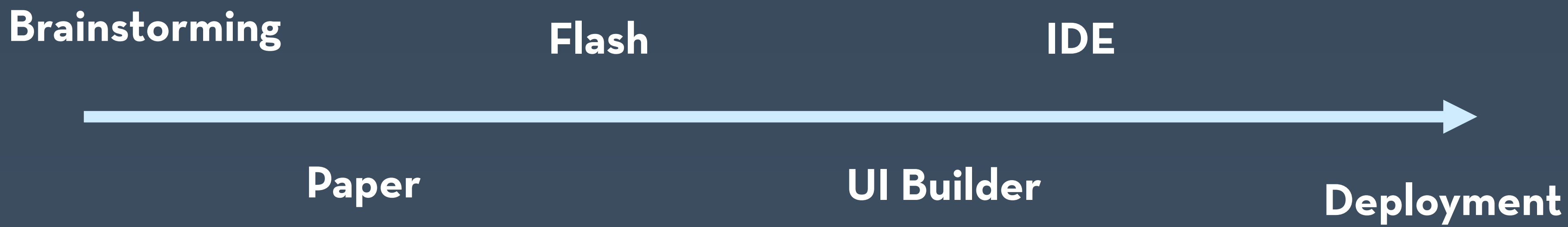
Application Types

- Each has own unique UI style, and implementation challenges
- Word processors
- Drawing programs
  - CAD/CAM
- Painting programs
- Hierarchy displays, like file browsers
- Mail readers
- Spreadsheets
- Forms processing
- WWW
- Interactive games
- Visualizations
- Automated-teller machines (ATM)
- Virtual Reality
- Multi-media
  - Video
  - Animation
- Controlling machinery

Metaphors

- Content metaphors
  - desktop
  - paper document
  - notebook with tabs
  - score sheet , stage with actors (Director)
  - accounting ledger (spreadsheet)
  - stereo (for all media players)
  - phone keypad
  - calculator
  - Web: "Shopping Carts"
  - Quicken: "CheckBook"
- Interaction metaphors = tools, agents: "electronic secretary“

A Software Design Timeline



Brainstorming

Paper

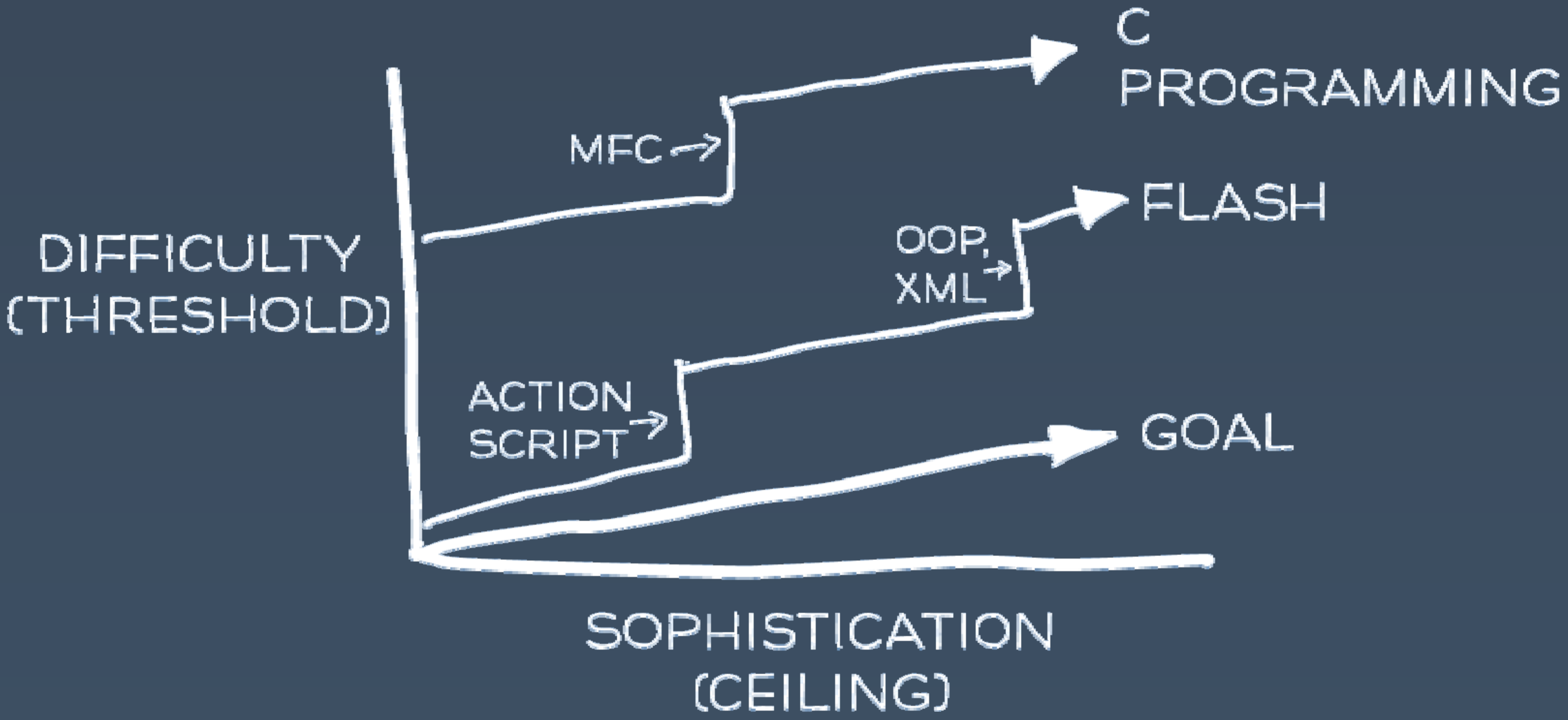
Flash

UI Builder

IDE

Deployment

Threshold and Ceiling



(after Myers)

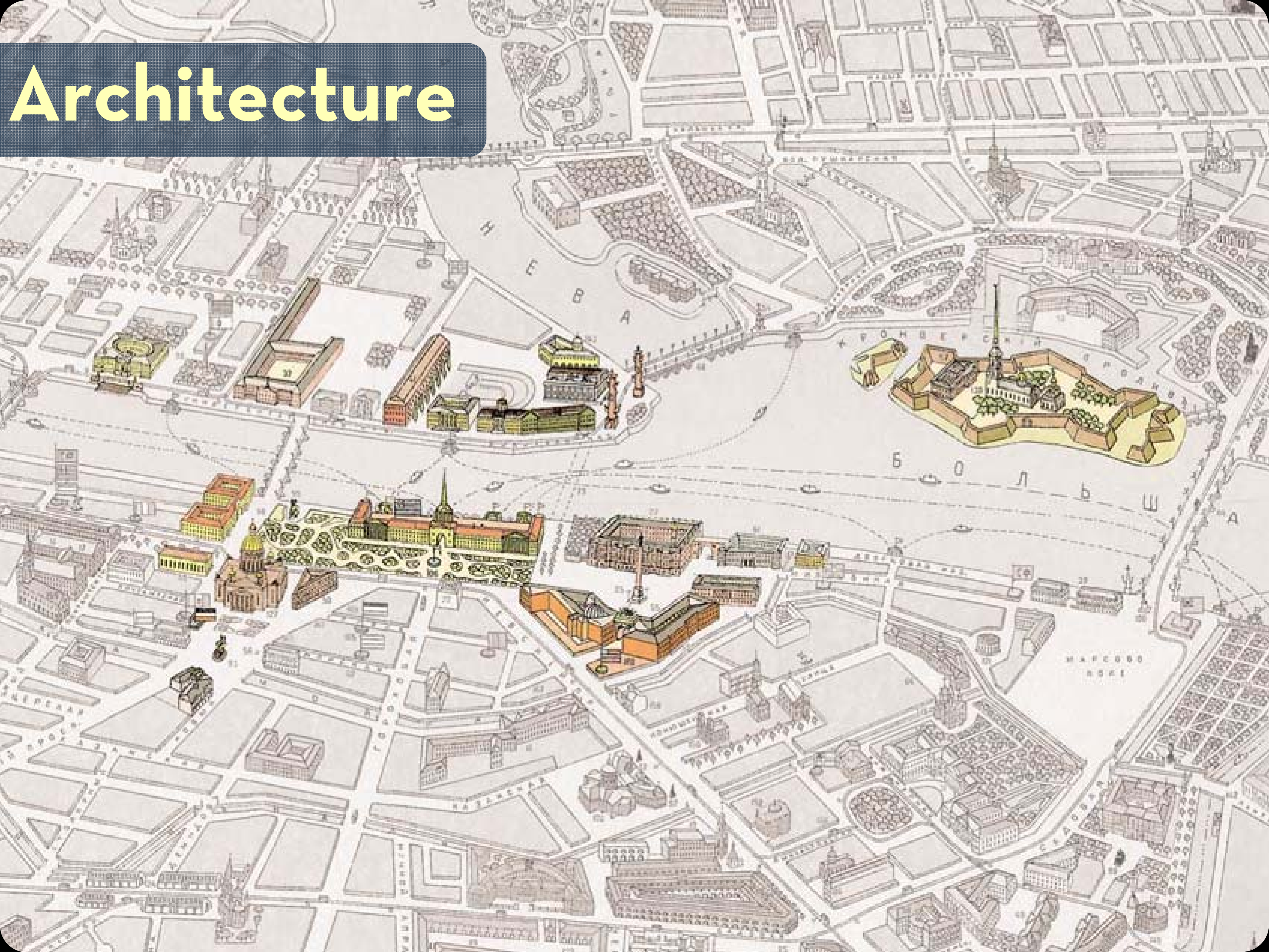
Discussion of Themes

- è Address the useful & important aspects of UIs
  - Narrower tools have been more successful than ones that try to do “everything”
  - Do one thing well
- è Threshold / Ceiling
  - Research systems often aim for high ceiling
  - Successful systems seem to instead aim for a low threshold
  - Impossible to have both?

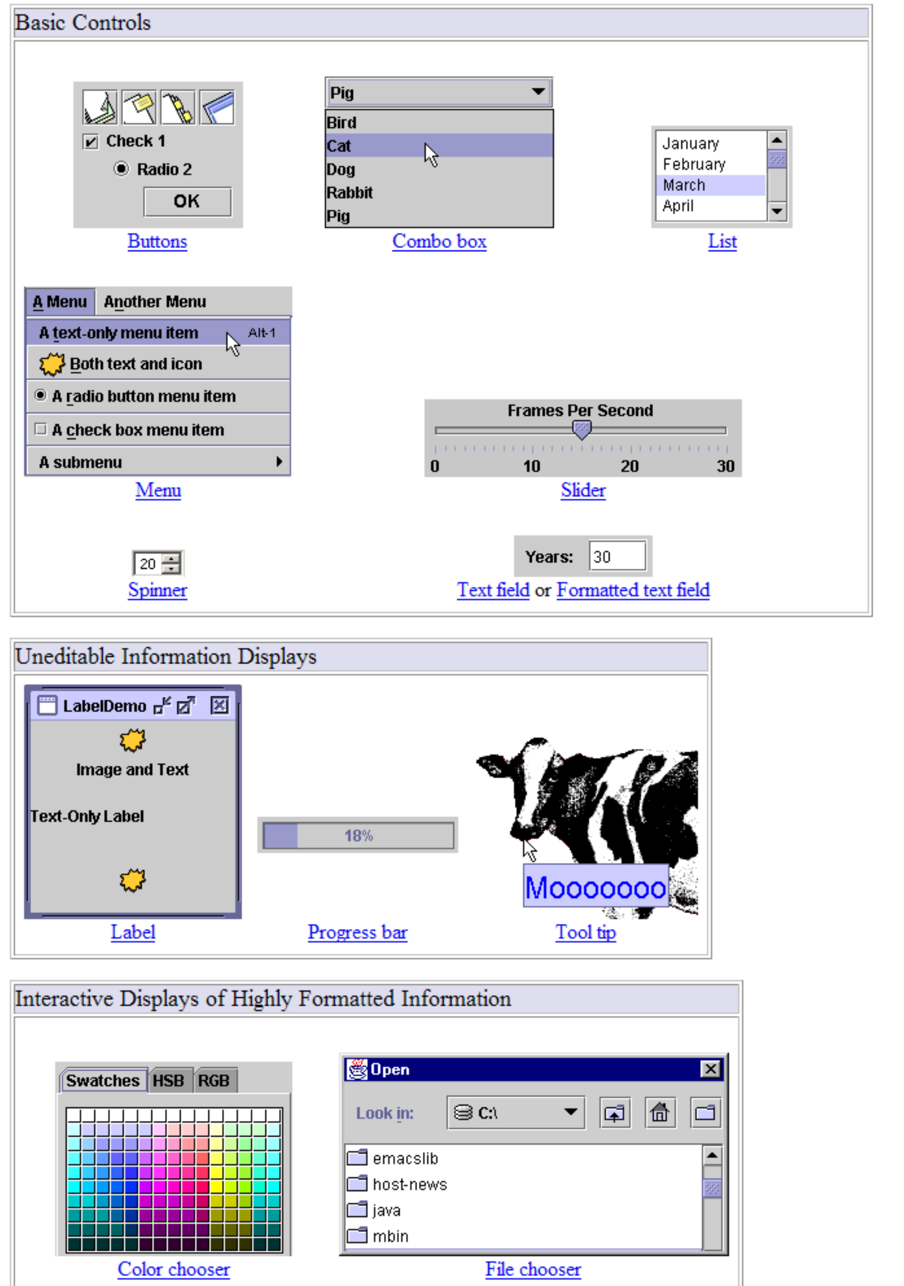
Library

TREES
HERBAL MEDICINE
FLORA BRITANNICA
RICHARD MABEY
Trees of Southern Africa
THE HEALING FOREST
HERBS
WILDFLOWERS
TREES & SHRUBS
PLANTS AND FLOWERS

Architecture



Library and Architecture



Discussion of Themes, cont.

- è Path of Least Resistance
  - Tools *should* guide implementers into better user interfaces
  - Goal for the future: do this more?
- è Predictability
  - Programmers do not seem willing to release control
  - Especially when system may do sub-optimal things
- è Moving Targets
  - Long stability of Macintosh Desktop paradigm has enabled maturing of tools

Window Managers

- Multiple (tiled) windows in research systems of 1960's: NLS, etc.
- Overlapping introduced in Alan Kay's thesis (1969)
- Smalltalk, 1974 at Xerox PARC
- Successful because multiple windows help users manage scarce resources:
  - Screen space and input devices
  - Attention of users
  - Affordances for reminding and finding other work

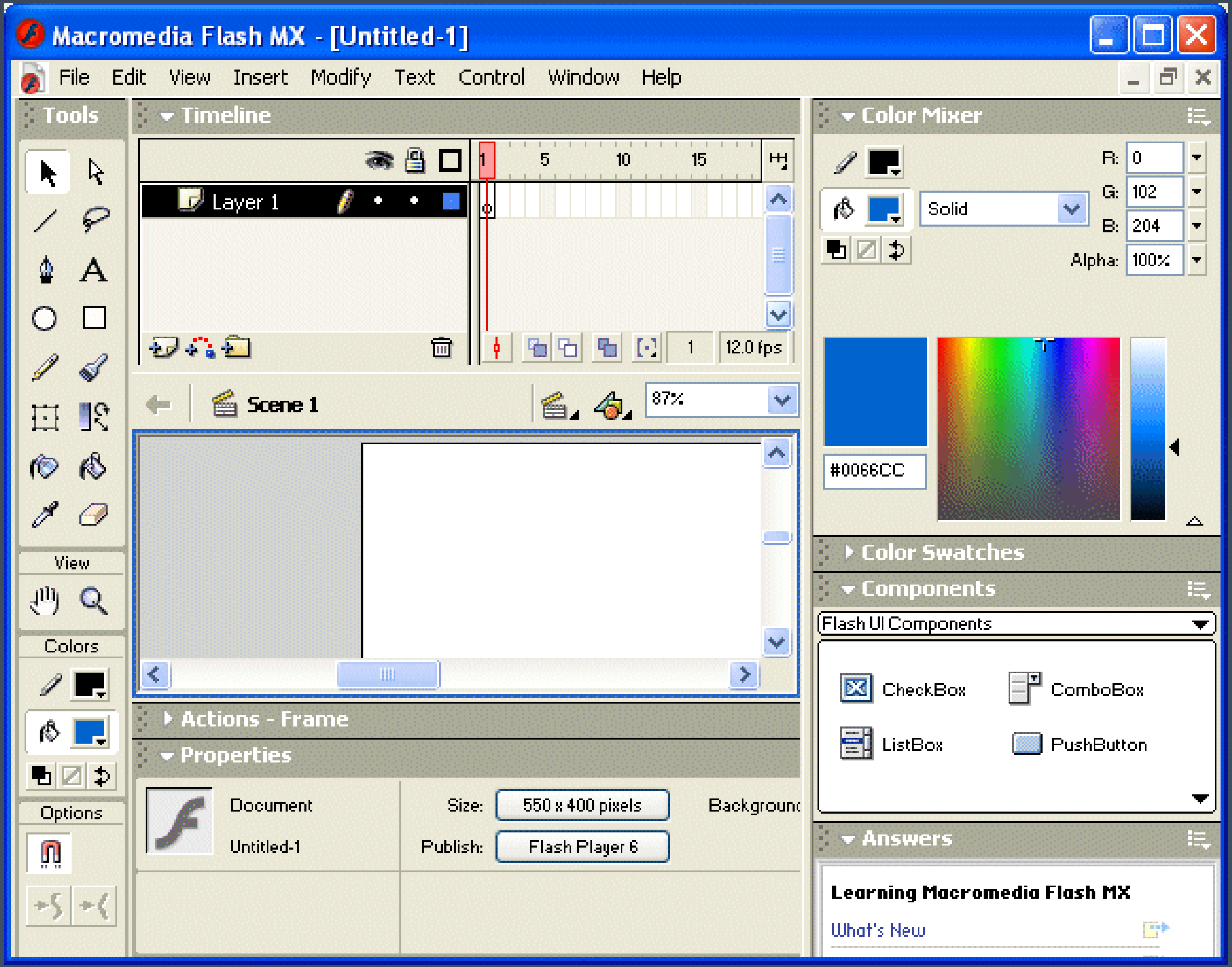
Event Languages

- Create programs by writing event handlers
- Many UIMSs used this style
  - Univ. of Alberta (1985), Sassafras (1986), etc.
- Now used by HyperCard, Visual Basic, Lingo, etc.
  - Toolkits with call-backs or action methods are related
- Advantages:
  - Natural for GUIs since generate discrete events
  - Flow of control in user’s hands rather than programmer’s
    - Discourages moded UIs

Graphical Interactive Tools

- Create parts of user interface by laying out widgets with a mouse
  - Examples: Menulay (1983), Trillium (1986), Jean-Marie Hullot from INRIA to NeXT
  - Now: Interface Builders, Visual Basic's layout editor, resource editors, "constructors"
- Advantages:
  - Graphical parts done in an appropriate, graphical way
    - èAddress the useful & important aspects of UIs
  - Accessible to non-programmers
    - èLow threshold

Interactive Prototypes



Document

Untitled-1

Size: 550 x 400 pixels

Publish: Flash Player 6

Backgrounc

R: 0

G: 102

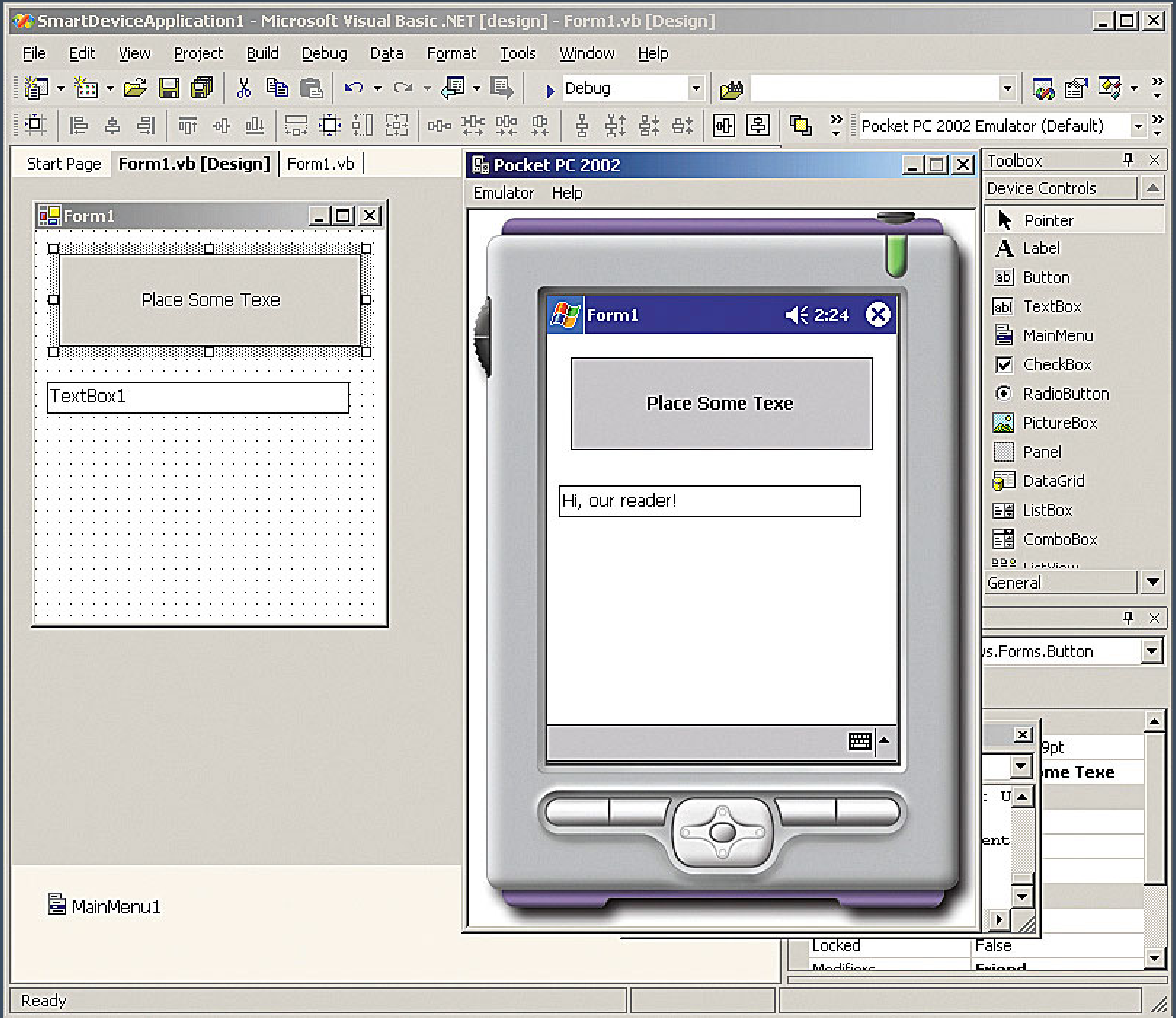
B: 204

Alpha: 100%

Solid

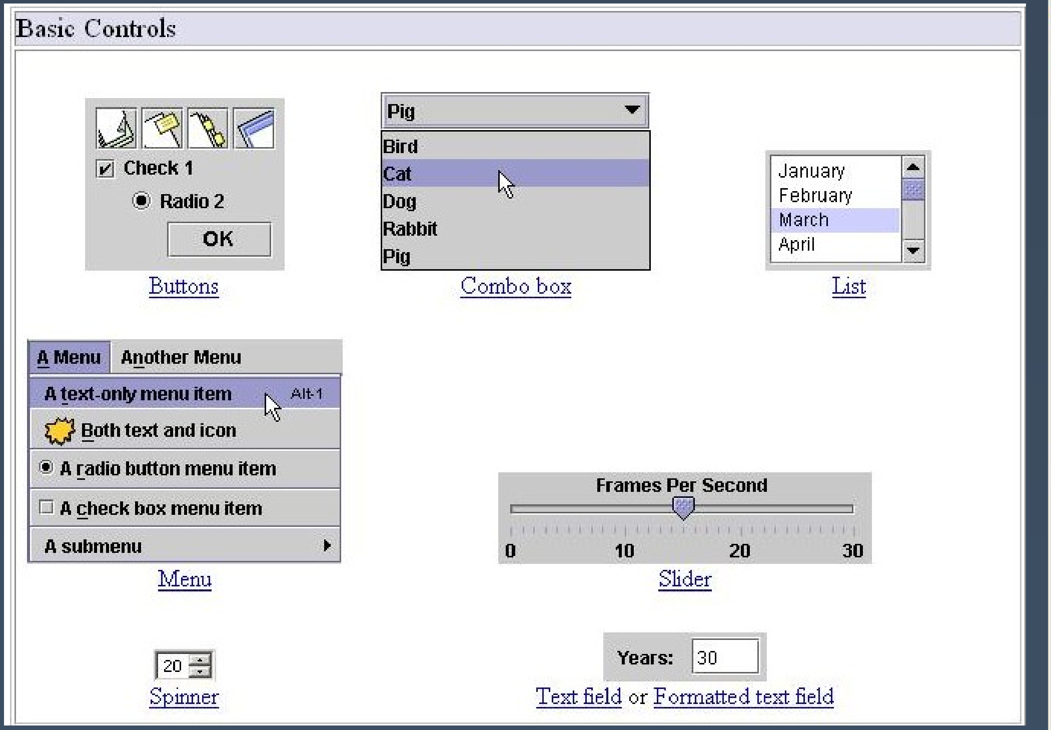
#0066CC

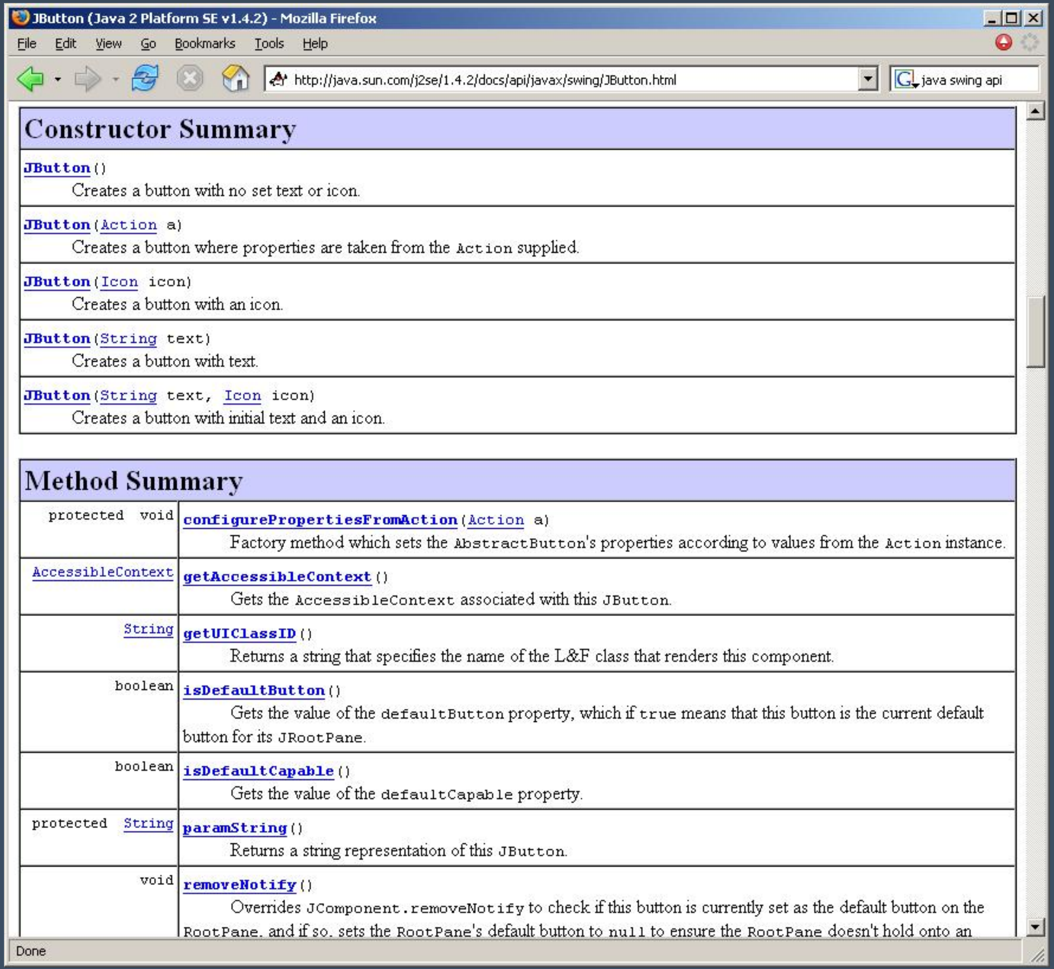
UI Builders


SmartDeviceApplication1 - Microsoft Visual Basic .NET [design] - Form1.vb [Design]
File
Edit
View
Project
Build
Debug
Data
Format
Tools
Window
Help
Debug
Pocket PC 2002 Emulator (Default)
Start Page
Form1.vb [Design]
Form1.vb
Form1
Place Some Texe
TextBox1
MainMenu1
Pocket PC 2002
Emulator
Help
Form1
2:24
Place Some Texe
Hi, our reader!
Toolbox
Device Controls
Pointer
Label
Button
TextBox
MainMenu
CheckBox
RadioButton
PictureBox
Panel
DataGrid
ListBox
ComboBox
General
s.Forms.Button
9pt
me Texe
Locked
False
Ready

Example: Java Swing

- GUI toolkit with a widget set and an API




JButton (Java 2 Platform SE v1.4.2) - Mozilla Firefox
File
Edit
View
Go
Bookmarks
Tools
Help
http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JButton.html
java swing api
Done

Constructor Summary

`JButton()`
Creates a button with no set text or icon.

`JButton(Action a)`
Creates a button where properties are taken from the `Action` supplied.

`JButton(Icon icon)`
Creates a button with an icon.

`JButton(String text)`
Creates a button with text.

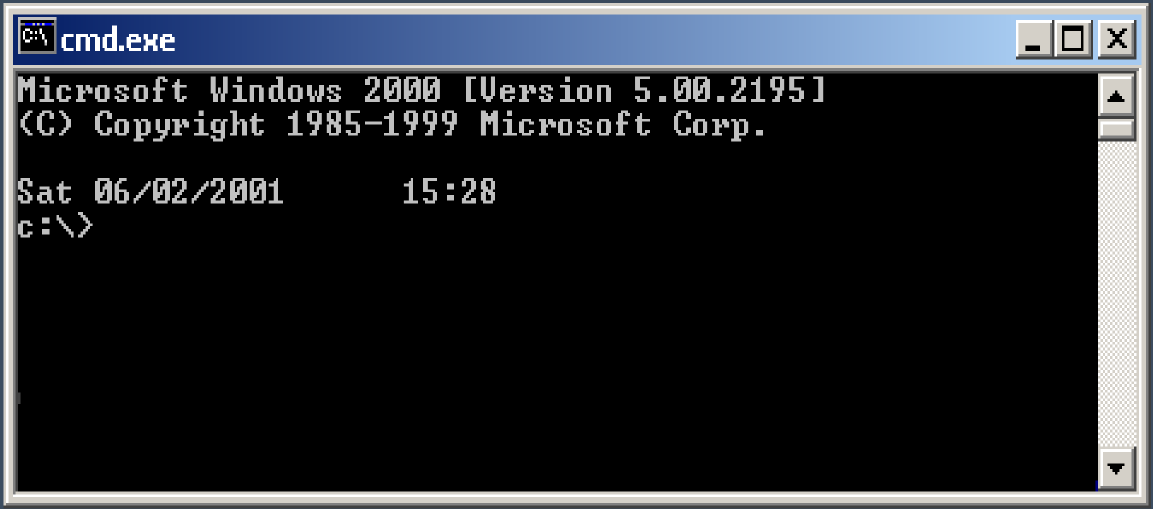
`JButton(String text, Icon icon)`
Creates a button with initial text and an icon.

Method Summary

`protected void`	`configurePropertiesFromAction(Action a)` Factory method which sets the `AbstractButton`'s properties according to values from the `Action` instance.
`AccessibleContext`	`getAccessibleContext()` Gets the `AccessibleContext` associated with this `JButton`.
`String`	`getUIClassID()` Returns a string that specifies the name of the L&F class that renders this component.
`boolean`	`isDefaultButton()` Gets the value of the `defaultButton` property, which if `true` means that this button is the current default button for its `JRootPane`.
`boolean`	`isDefaultCapable()` Gets the value of the `defaultCapable` property.
`protected String`	`paramString()` Returns a string representation of this `JButton`.
`void`	`removeNotify()` Overrides `JComponent.removeNotify` to check if this button is currently set as the default button on the `RootPane`, and if so, sets the `RootPane`'s default button to `null` to ensure the `RootPane` doesn't hold onto an

Sequential Programs

- Program takes control, prompts for input
  - command-line prompts (DOS, UNIX)

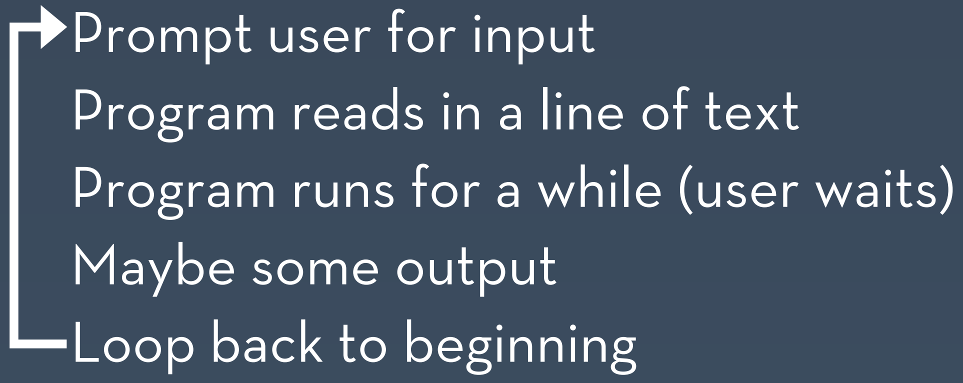


```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

Sat 06/02/2001      15:28
c:\>
```

- The user waits on the program
  - program tells user it’s ready for more input
  - user enters more input

Sequential Programs (cont.)

- General Flow



Prompt user for input

Program reads in a line of text

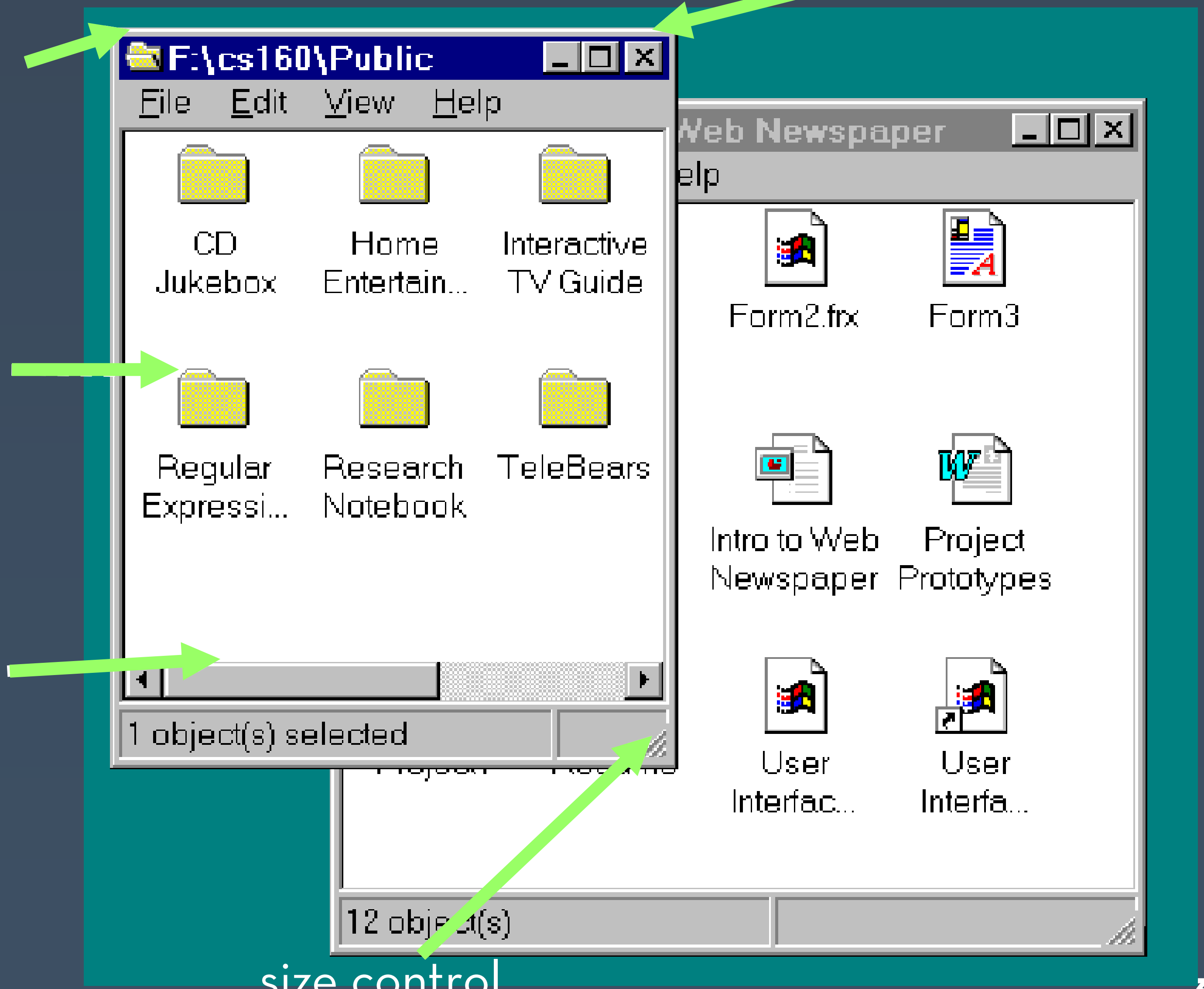
Program runs for a while (user waits)

Maybe some output

Loop back to beginning

- But how do you model the many actions a user can take?
  - for example, a word processor?
  - printing, editing, inserting, whenever user wants
  - sequential doesn't work as well for graphical and for highly-interactive apps

Example Interactions



close box

title bar

folder

scroll bar

size control

Modern GUI Systems

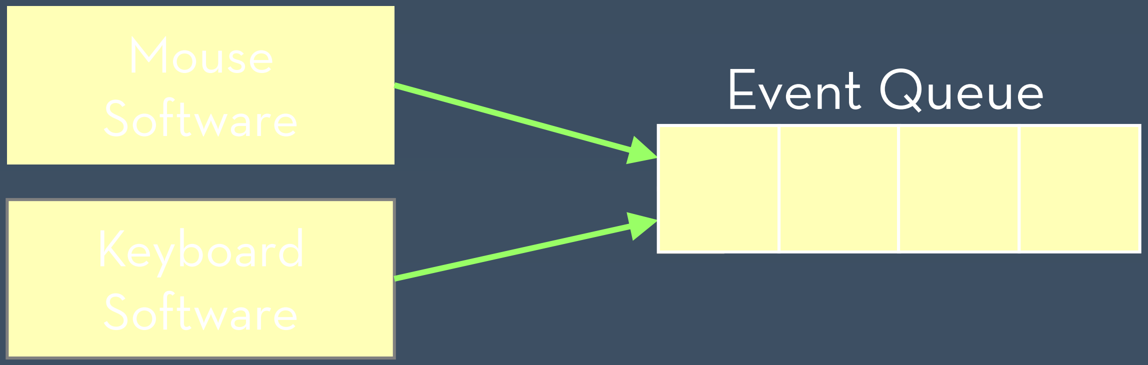
- Three concepts:
  - Event-driven programming
  - Widgets
  - Interactor Tree
- Describes how most GUIs work
  - Closest to Java
  - But similar to Windows, Mac, Palm Pilot

Event-Driven Programming

- Instead of the user waiting on program, program waits on the user
- All communication from user to computer is done via “events”
  - “mouse button went down”
  - “item is being dragged”
  - “keyboard button was hit”
- Events have:
  - type of event
  - mouse position or character key + modifiers
  - the window the event is directed to

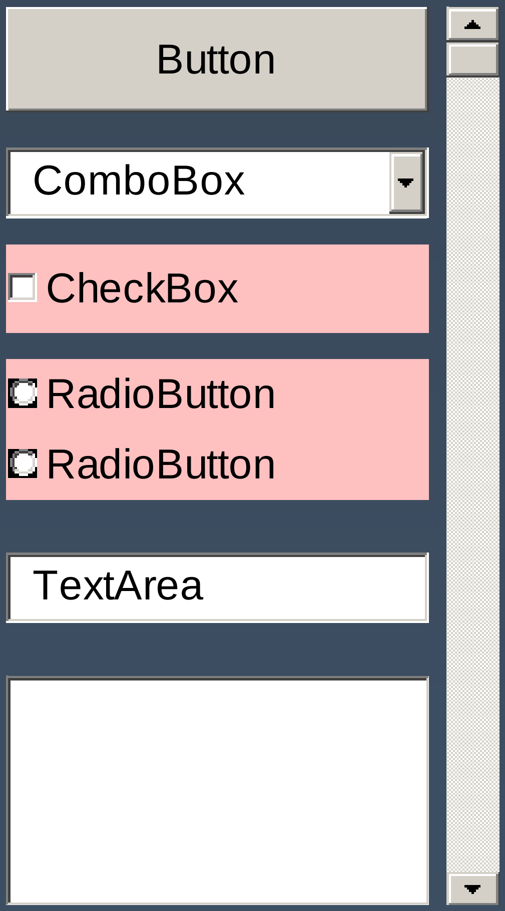
Event-Driven Programming

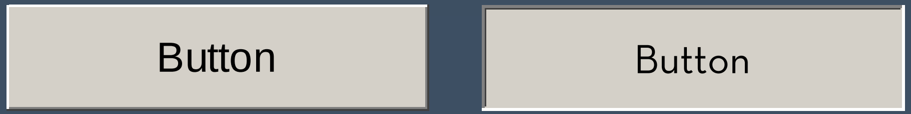
- All generated events go to a single *event queue*
    - provided by operating system
    - ensures that events are handled in the order they occurred
    - hides specifics of input from apps



Widgets

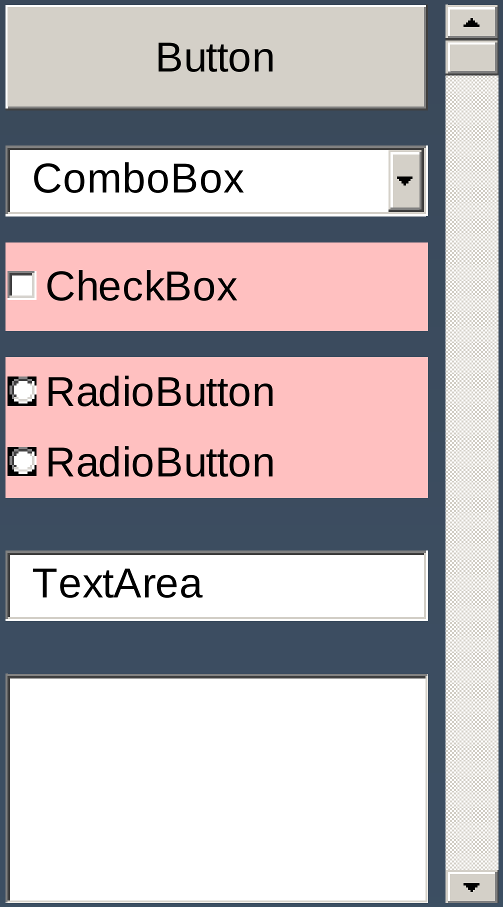
- Reusable interactive objects
- Handle certain events
  - widgets say what events they are interested in
  - event queue sends events to the “right” widget
- Update appearance
  - e.g. button up / button down



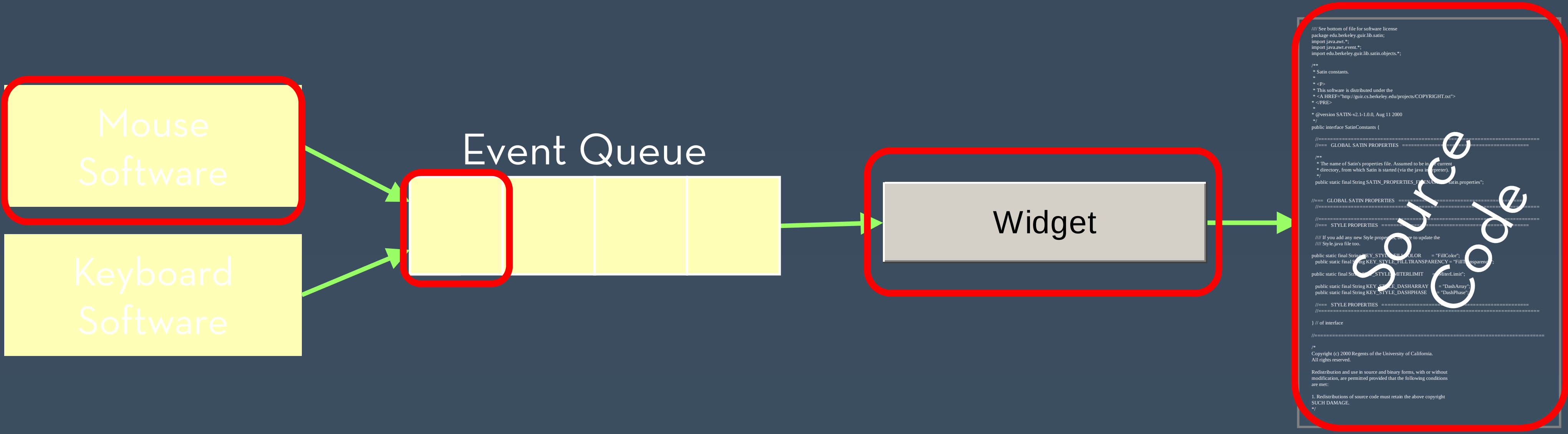


Widgets (cont.)

- Generate some new events
  - “button pressed”
  - “window closing”
  - “text changed”
- But these events are sent to interested listeners instead
  - custom code goes there



Widgets (cont.)

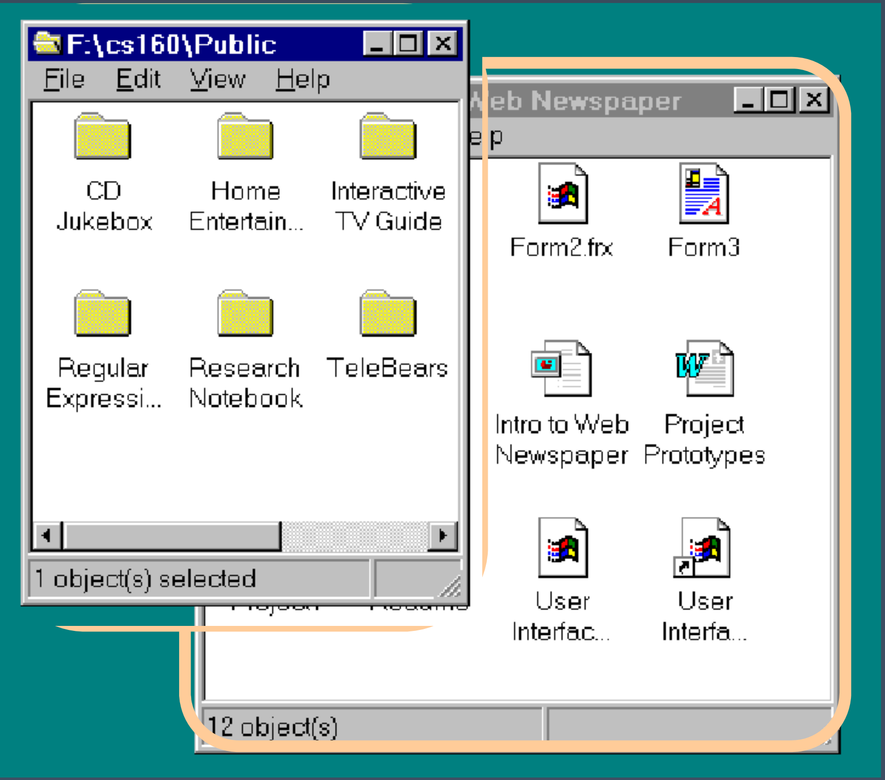


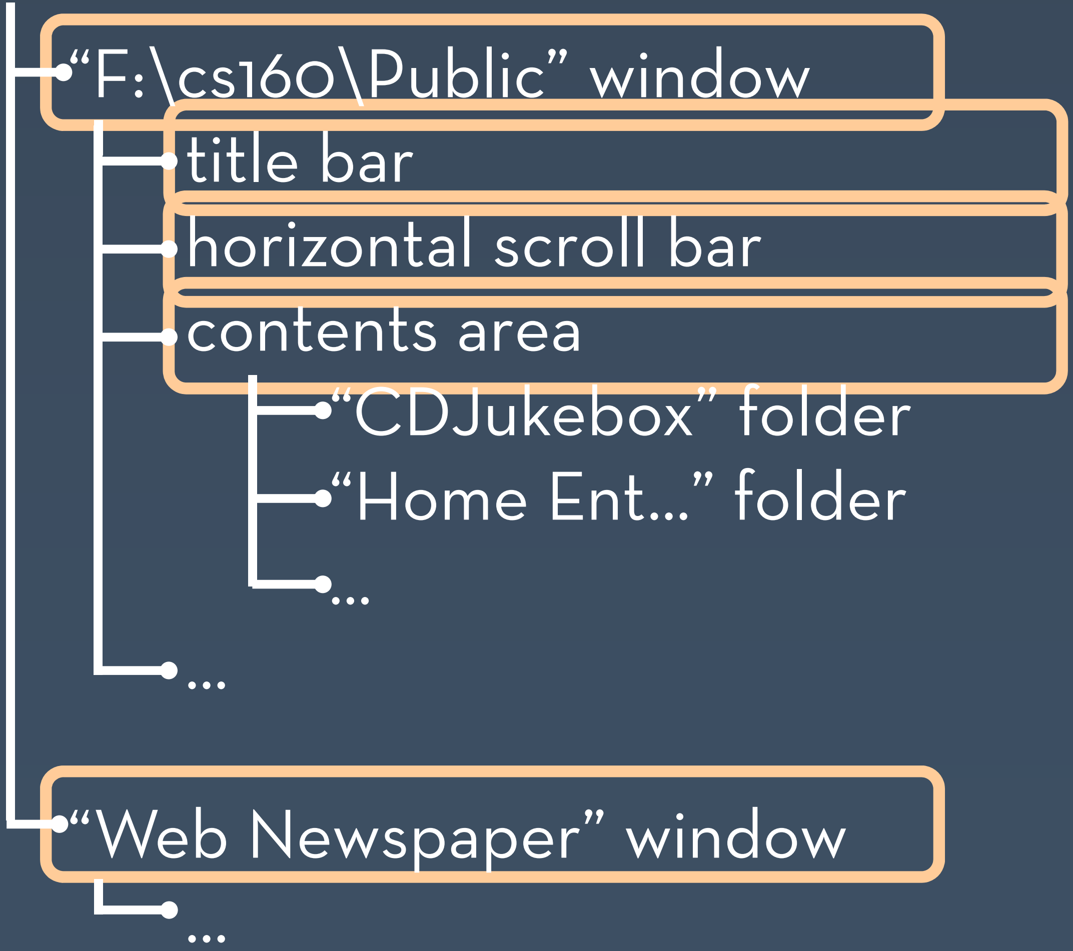
```
//// See bottom of file for software license
package edu.berkeley.guir.lib.satin;
import java.awt.*;
import java.awt.event.*;
import edu.berkeley.guir.lib.satin.objects.*;

/**
 * Satin constants.
 *
 * <P>
 * This software is distributed under the
 * <A HREF="http://guir.cs.berkeley.edu/projects/COPYRIGHT.txt">
* </PRE>
 *
* @version SATIN-v2.1-1.0.0, Aug 11 2000
 */
public interface SatinConstants {

   //==========================================================================
   //===   GLOBAL SATIN PROPERTIES   ==========================================

   /**
    * The name of Satin's properties file. Assumed to be in the current
    * directory, from which Satin is started (via the java interpreter).
    */
   public static final String SATIN_PROPERTIES_FILENAME = "satin.properties";


//===   GLOBAL SATIN PROPERTIES   ==========================================
   //==========================================================================

   //==========================================================================
   //===   STYLE PROPERTIES   =================================================

   //// If you add any new Style properties, be sure to update the
   //// Style.java file too.

public static final String KEY_STYLE_FILLCOLOR        = "FillColor";
   public static final String KEY_STYLE_FILLTRANSPARENCY = "FillTransparency";

public static final String KEY_STYLE_MITERLIMIT       = "MiterLimit";

   public static final String KEY_STYLE_DASHARRAY      = "DashArray";
   public static final String KEY_STYLE_DASHPHASE      = "DashPhase";

   //===   STYLE PROPERTIES   =================================================
   //==========================================================================

} // of interface

//============================================================================

/*
Copyright (c) 2000 Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
SUCH DAMAGE.
*/
```

Interactor Tree

- Decompose interactive objects into a tree

Display Screen

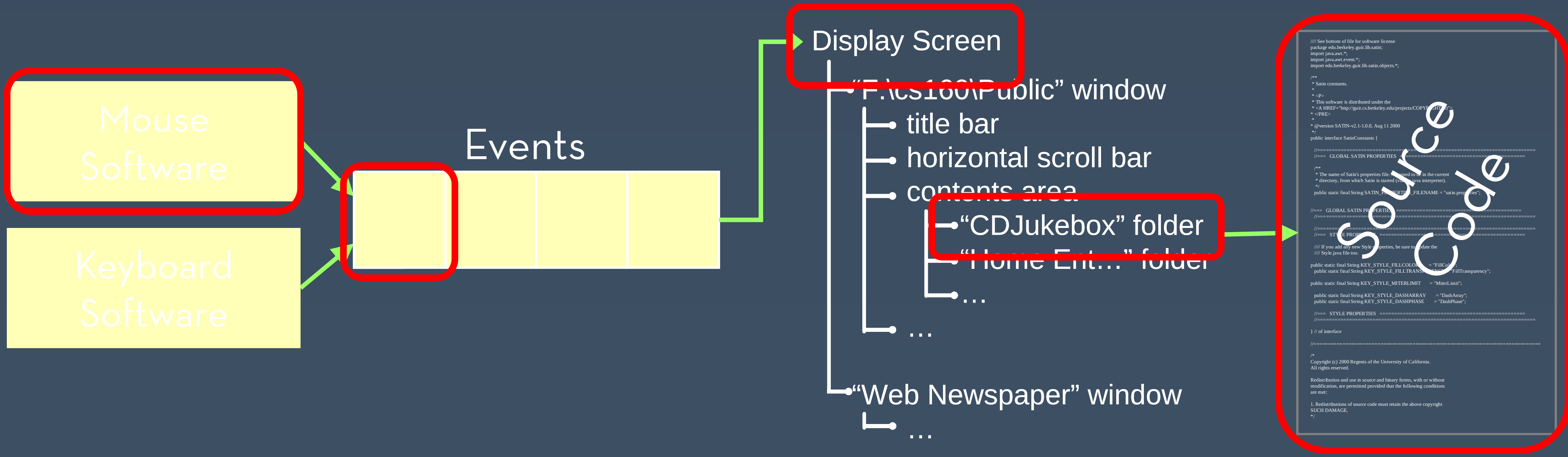




“F:\cs160\Public” window
title bar
horizontal scroll bar
contents area
“CDJukebox” folder
“Home Ent…” folder
…
…
“Web Newspaper” window
…

Main Event Loop

```
while (app is running) {
        get next event
        send event to right widget
}
```



```
//// See bottom of file for software license
package edu.berkeley.guir.lib.satin;
import java.awt.*;
import java.awt.event.*;
import edu.berkeley.guir.lib.satin.objects.*;

/**
 * Satin constants.
 *
 * <P>
 * This software is distributed under the
 * <A HREF="http://guir.cs.berkeley.edu/projects/COPYRIGHT.txt">
* </PRE>
 *
* @version SATIN-v2.1-1.0.0, Aug 11 2000
 */
public interface SatinConstants {

  //===========================================================================
  //===   GLOBAL SATIN PROPERTIES   =========================================

  /**
   * The name of Satin's properties file. Assumed to be in the current
   * directory, from which Satin is started (via the java interpreter).
   */
  public static final String SATIN_PROPERTIES_FILENAME = "satin.properties";


//===   GLOBAL SATIN PROPERTIES   =========================================
  //===========================================================================

  //===========================================================================
  //===   STYLE PROPERTIES   ==================================================

  //// If you add any new Style properties, be sure to update the
  //// Style.java file too.

public static final String KEY_STYLE_FILLCOLOR        = "FillColor";
  public static final String KEY_STYLE_FILLTRANSPARENCY = "FillTransparency";

public static final String KEY_STYLE_MITERLIMIT       = "MiterLimit";

  public static final String KEY_STYLE_DASHARRAY        = "DashArray";
  public static final String KEY_STYLE_DASHPHASE        = "DashPhase";

  //===   STYLE PROPERTIES   ==================================================
  //===========================================================================

} // of interface

//=============================================================================

/*
Copyright (c) 2000 Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
SUCH DAMAGE.
*/
```

What this means for design

- Harder to use non-standard widgets
  - have to buy or create your own, ex. pie menus
- Easy to re-arrange widgets and layout of app, but hard to change behavior (i.e. the code)
  - provides some support, not a lot
  - stresses importance of getting features right first
- Harder to do things beyond mouse and keyboard
  - speech and sketching harder
- Harder to do multi-user multi-device apps

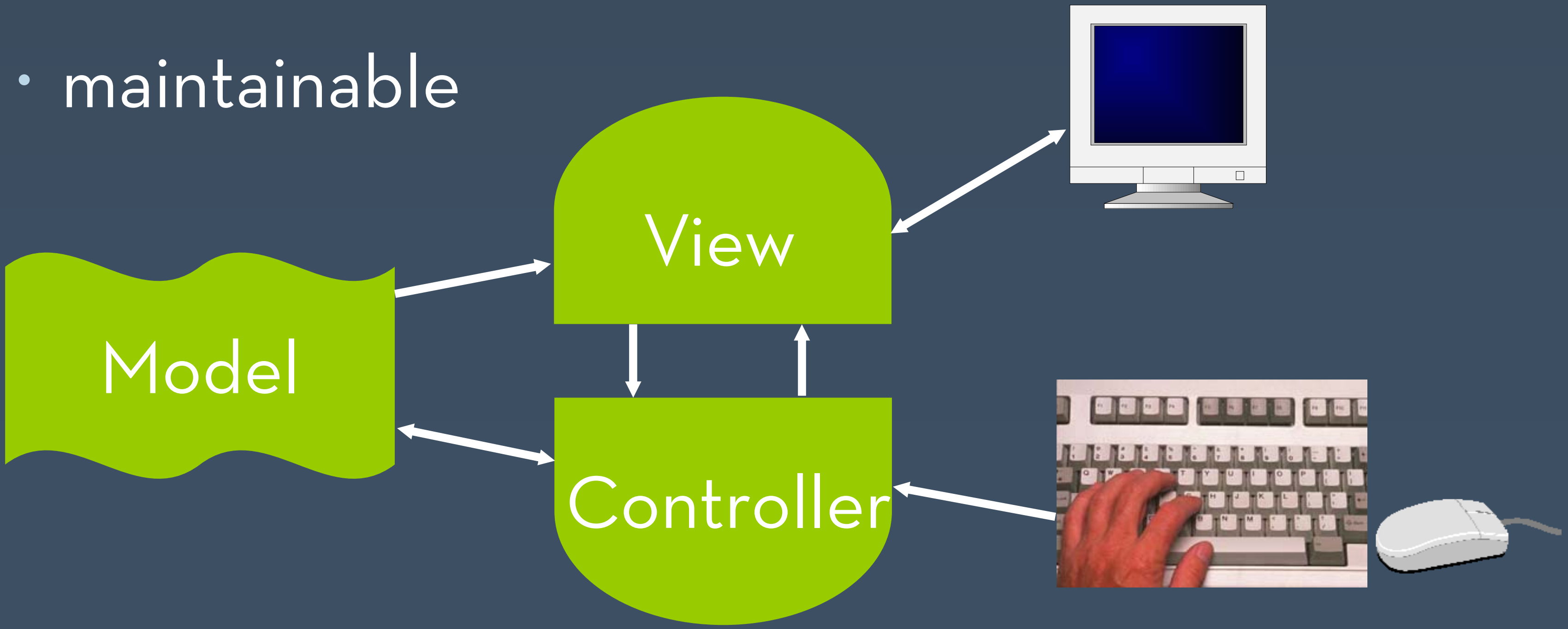
Scripting Languages

- First GUIs used interpreted languages
  - Smalltalk, InterLisp
    - Rapid development, supports prototyping

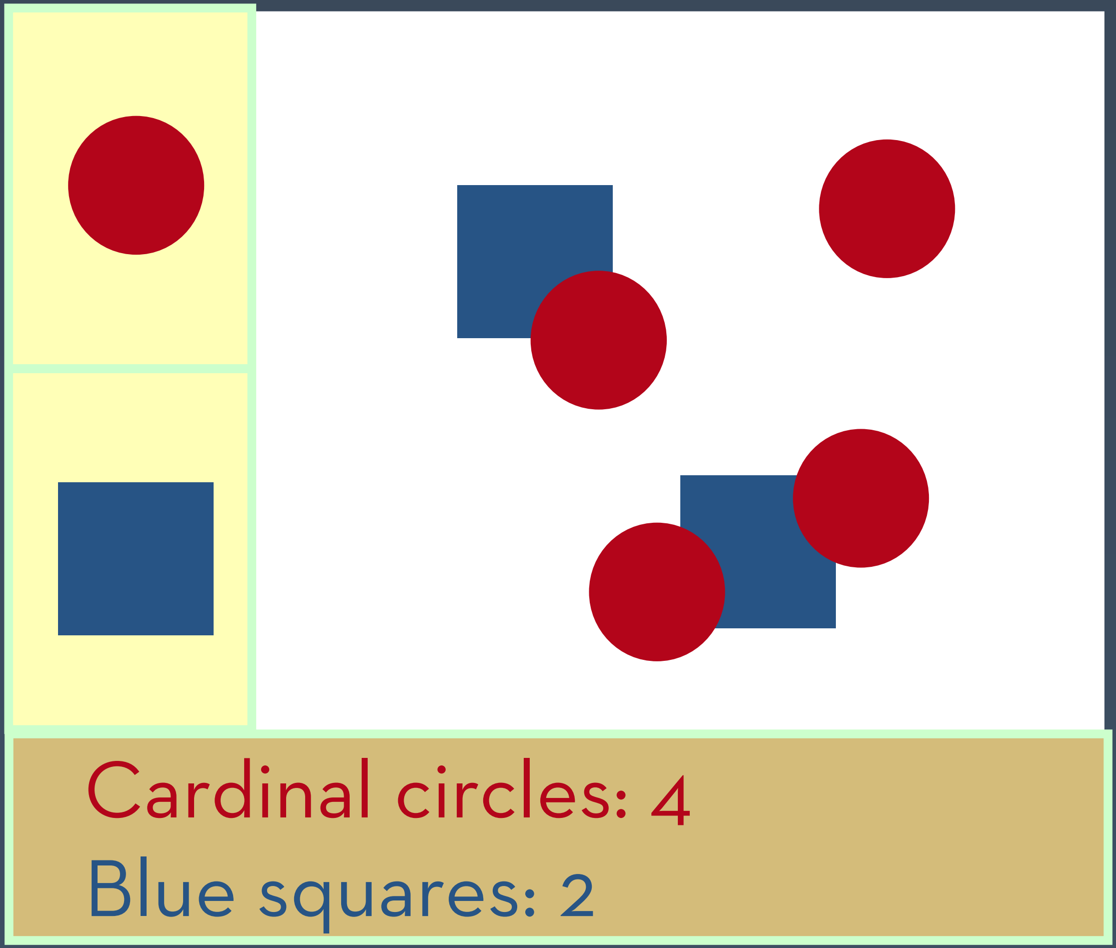
    èLow threshold
- Then C and C++ became popular
- Now, bringing back advantages in scripting languages
  - tcl/tk, Python, perl
  - Visual Basic, Javascript
- But language ***must*** contain general-purpose control structures

Model-View-Controller

- Architecture for interactive apps
  - introduced by Smalltalk developers at PARC
- Partitions application in a way that is
  - scalable
  - maintainable



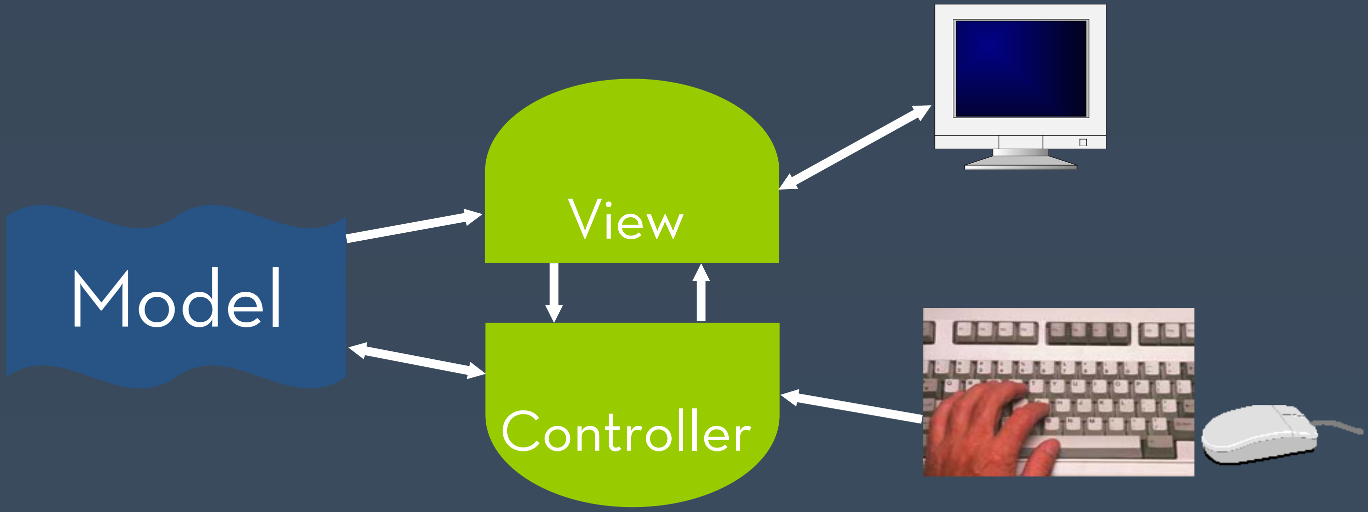
Example Application



Cardinal circles: 4

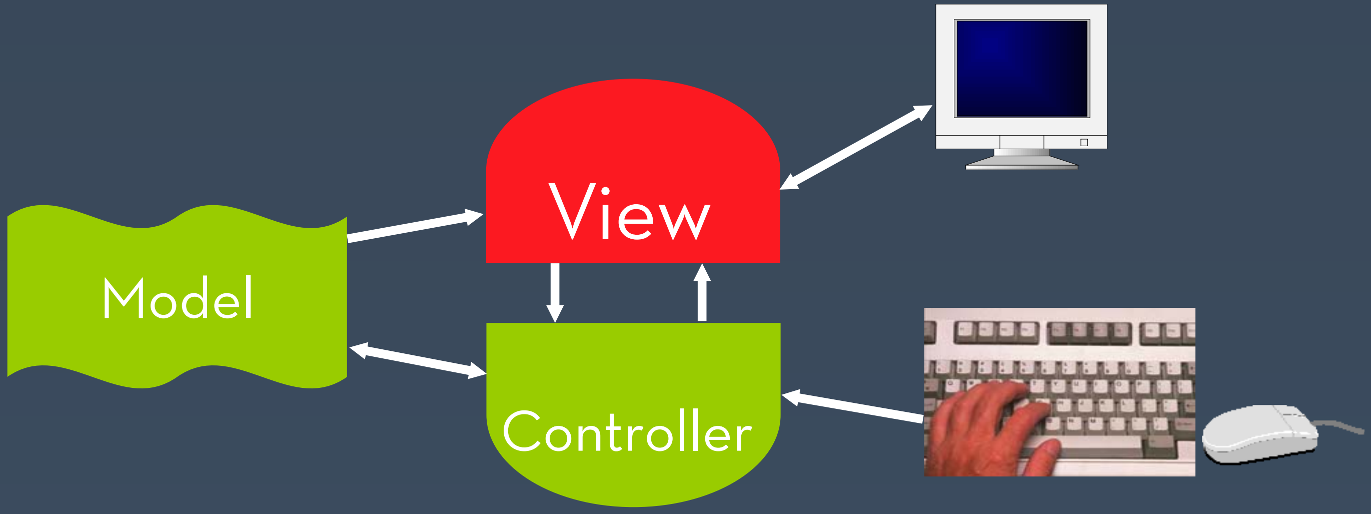
Blue squares: 2

Model



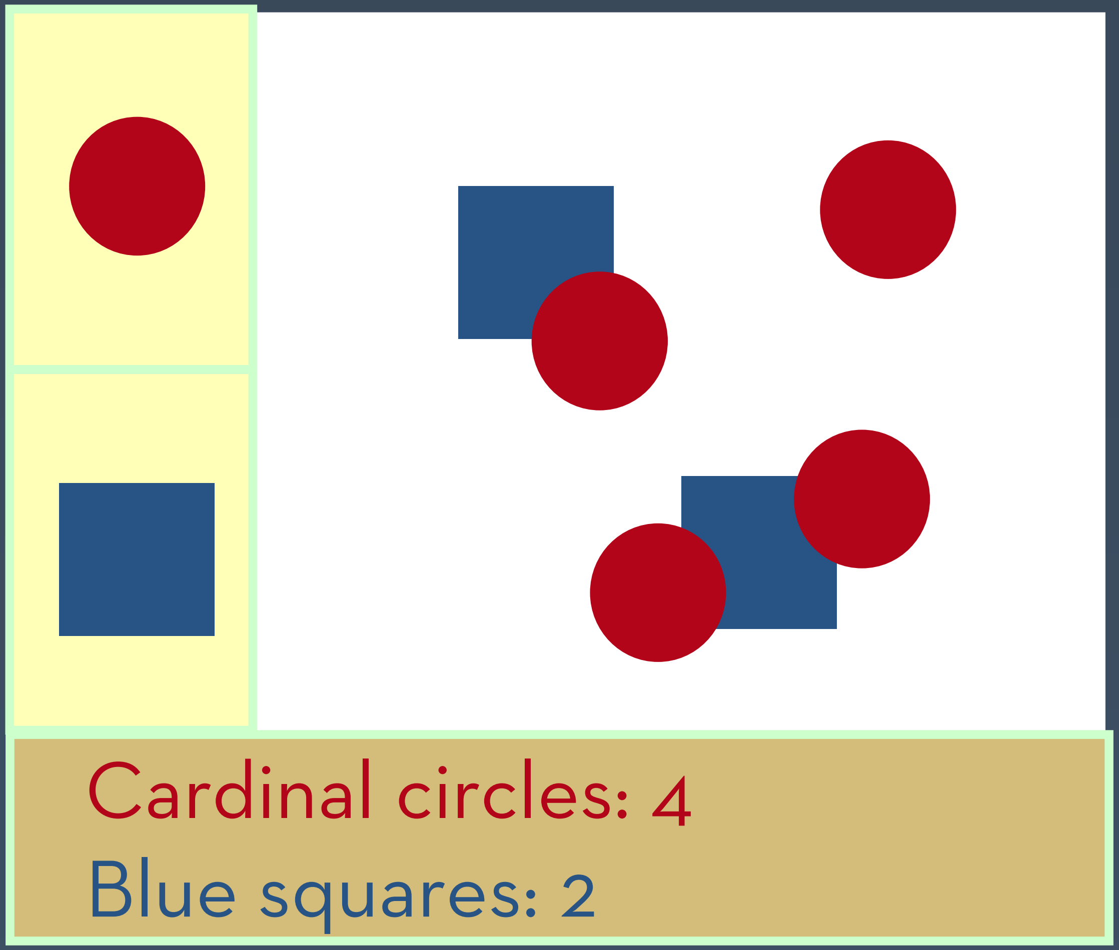
- Information the app is trying to manipulate
- Representation of real world objects
  - circuit for a CAD program
    - logic gates and wires connecting them
  - shapes in a drawing program
    - geometry and color

View



- Implements a visual display of the model
- May have multiple views
  - e.g., shape view and numerical view

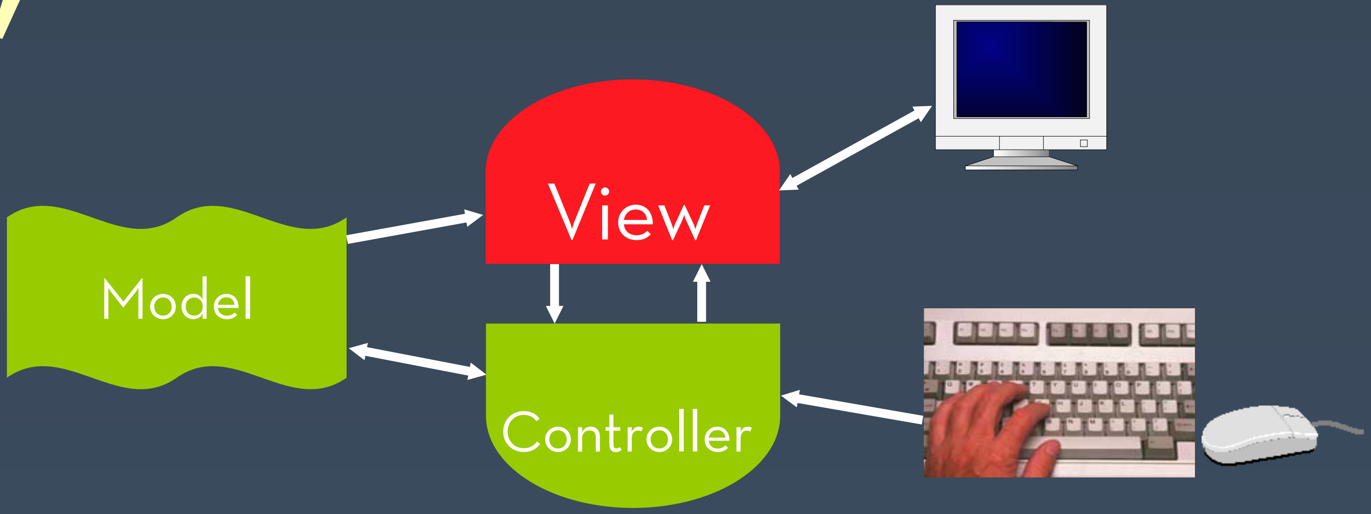
Multiple Views



Cardinal circles: 4

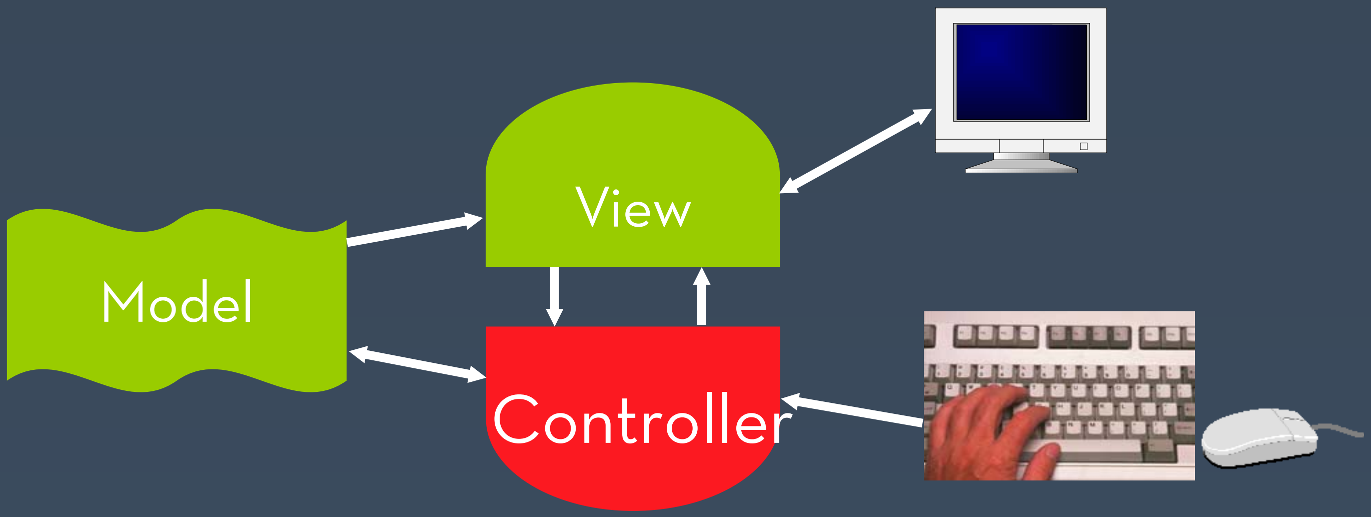
Blue squares: 2

View



- Implements a visual display of the model
- May have multiple views
  - e.g., shape view and numerical view
- Any time the model is changed, each view must be notified so that it can change *later*
  - e.g., adding a new shape

Controller



- Receives all input events from the user
- Decides what they mean and what to do
  - communicates with view to determine which objects are being manipulated (e.g., selection)
  - calls model methods to make changes on objects
    - model makes change and notifies views to update

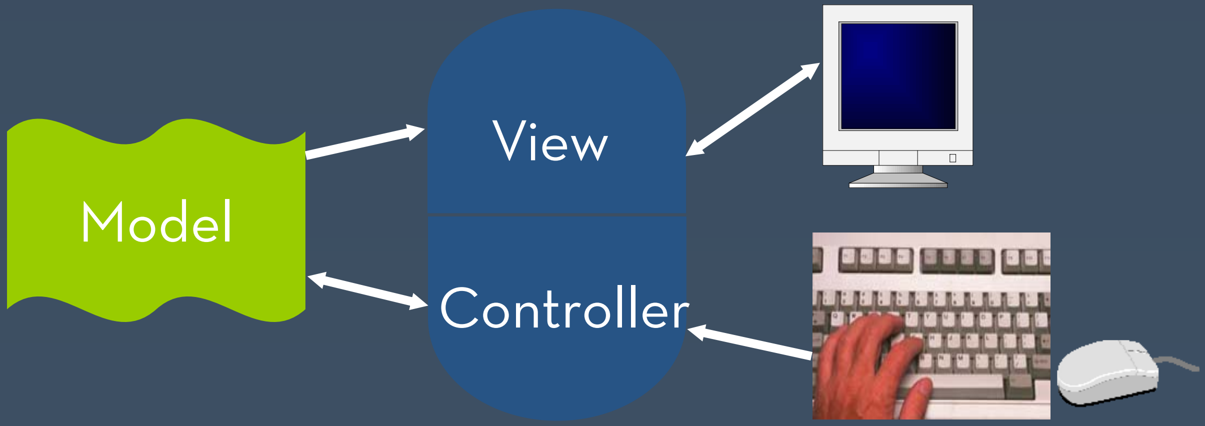
View/Controller Relationship

"pattern of behavior in response to user events (controller issues) is independent of visual geometry (view issues)"

- Controller must contact view to interpret what user events mean (e.g., selection)

Combining View & Controller

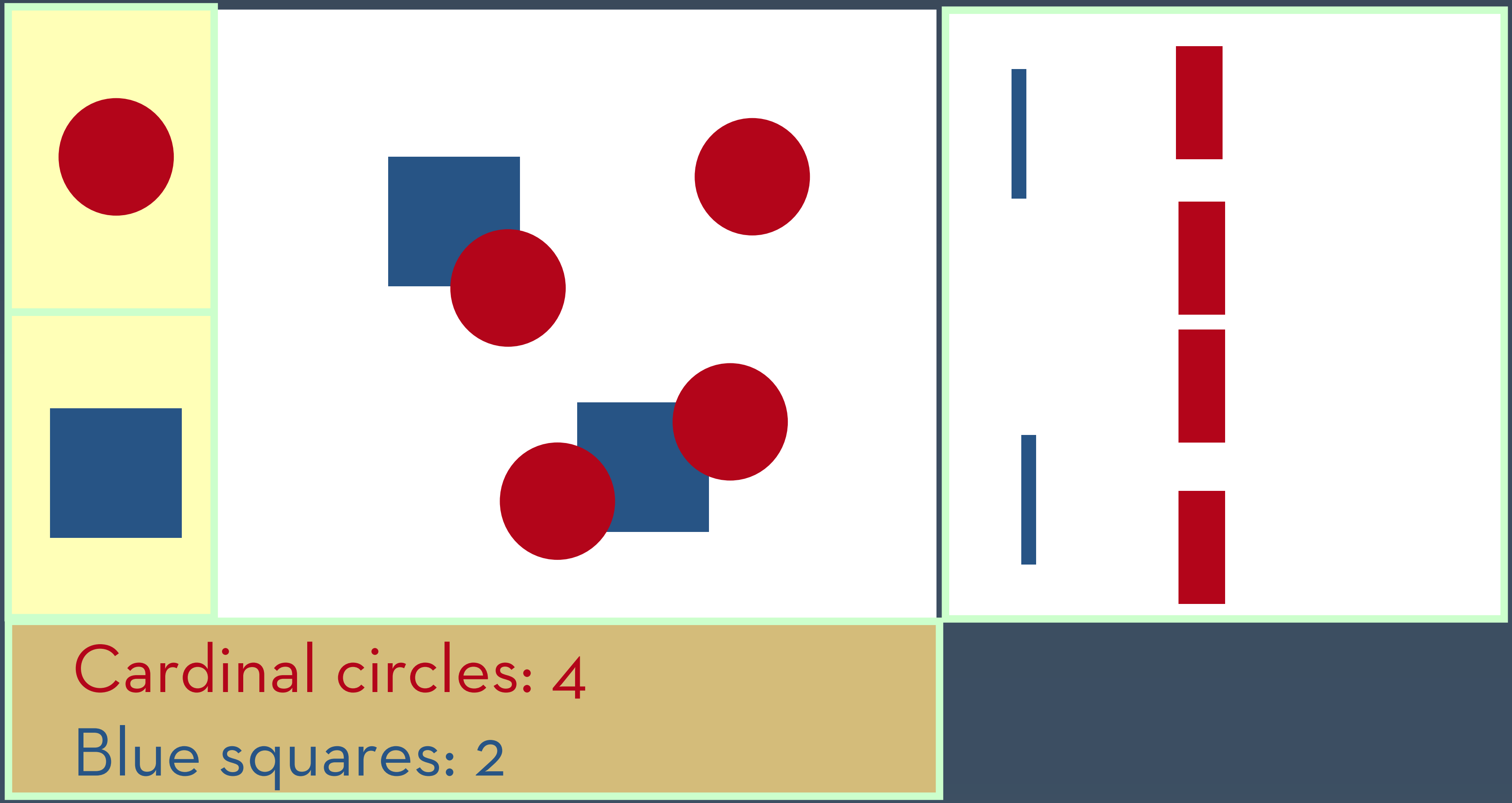
- View and controller are tightly intertwined
  - lots of communication between the two
- Almost always occur in pairs
  - i.e., for each view, need a separate controller
- Many architectures combine into a single class



Why MVC?

- Combining MVC into one class or using global variables will not scale
    - model may have more than one view
        - each is different and needs update when model changes
- Separation eases maintenance
    - easy to add a new view later
    - new model info may be needed, but old views still work
    - can change a view later, e.g., draw shapes in 3-d (recall, view handles selection)

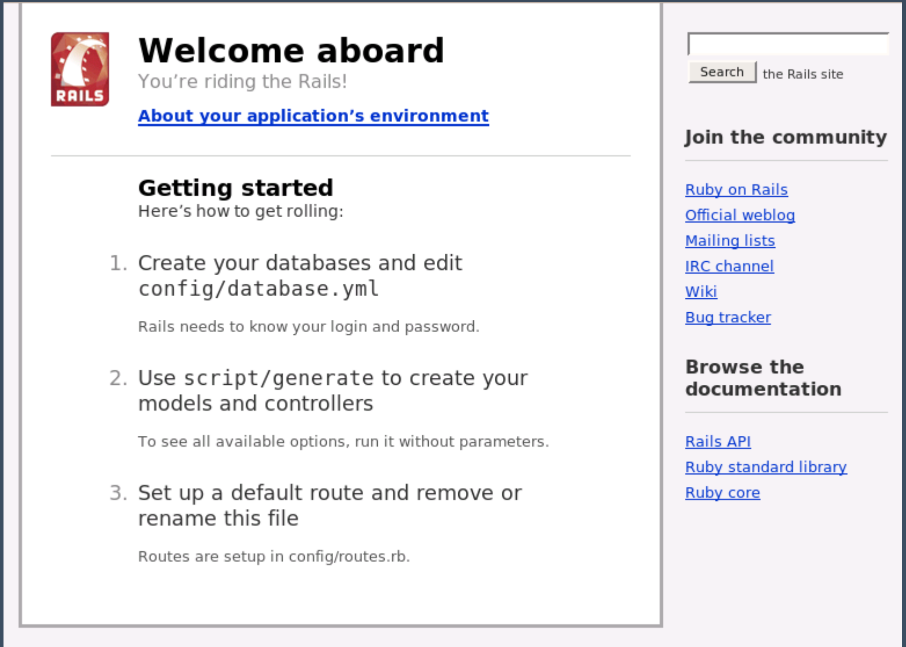
Adding Views Later



Cardinal circles: 4

Blue squares: 2

Example Frameworks : Ruby on Rails



Welcome aboard

You're riding the Rails!

About your application's environment

Getting started

Here's how to get rolling:

1. Create your databases and edit `config/database.yml`

   Rails needs to know your login and password.

2. Use `script/generate` to create your models and controllers

   To see all available options, run it without parameters.

3. Set up a default route and remove or rename this file

   Routes are setup in config/routes.rb.

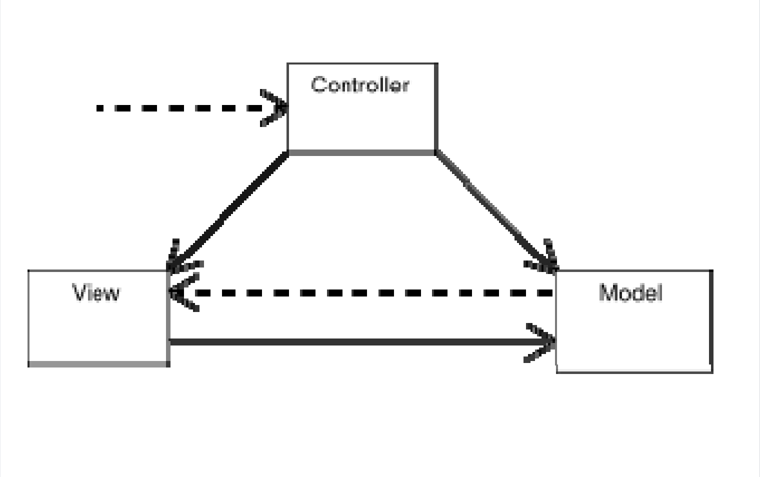
Join the community

- Ruby on Rails
- Official weblog
- Mailing lists
- IRC channel
- Wiki
- Bug tracker

Browse the documentation

- Rails API
- Ruby standard library
- Ruby core

Ruby on Rails



MVC

Source: Wikipedia

Example Frameworks : Ruby on Rails

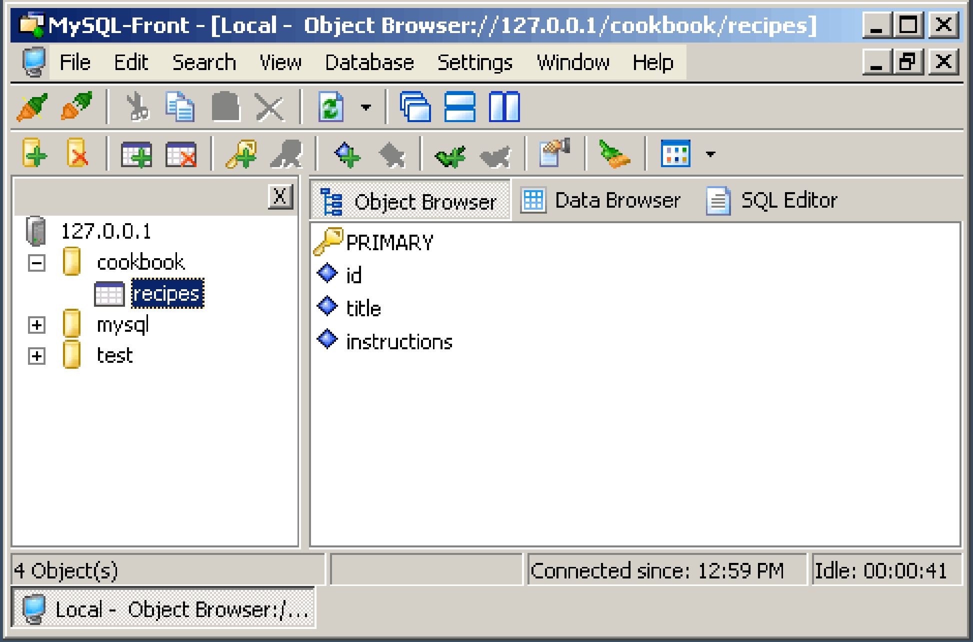
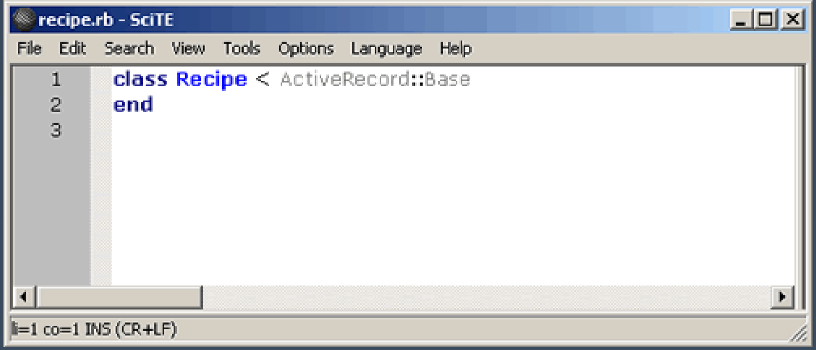

MySQL-Front - [Local - Object Browser://127.0.0.1/cookbook/recipes]
File Edit Search View Database Settings Window Help
127.0.0.1
cookbook
recipes
mysql
test
Object Browser
Data Browser
SQL Editor
PRIMARY
id
title
instructions
4 Object(s)
Connected since: 12:59 PM
Idle: 00:00:41
Local - Object Browser:/...

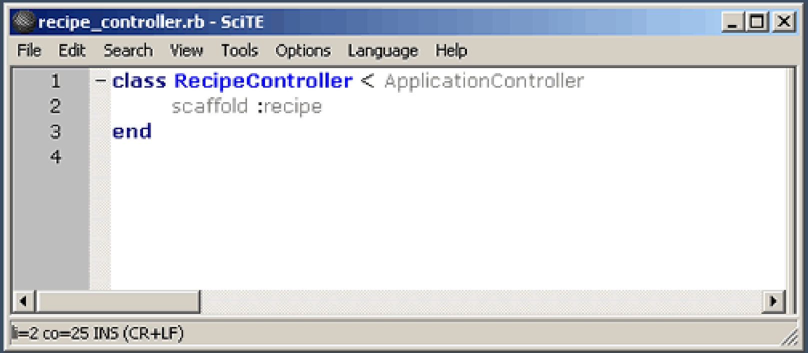
Figure 27. The modified recipe table [in MySQL – the Model]



```
class Recipe < ActiveRecord::Base
end
```

li=1 co=1 INS (CR+LF)

Figure 29. The contents of recipe.rb


recipe_controller.rb - SciTE
File Edit Search View Tools Options Language Help

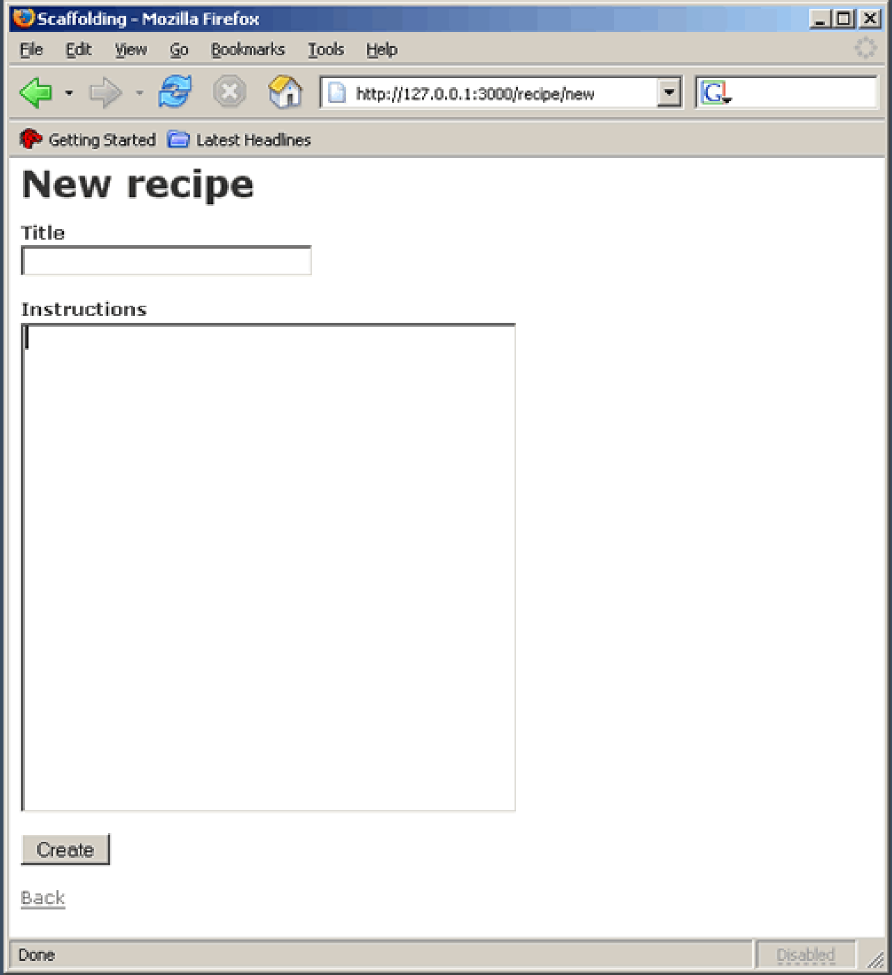
```
class RecipeController < ApplicationController
        scaffold :recipe
end
```

li=2 co=25 INS (CR+LF)

Figure 31. One line of code in RecipeController [the Controller]

Source: http://www.onlamp.com/pub/a/onlamp/2005/01/20/rails.html?page=3

Example Frameworks : Ruby on Rails



Scaffolding - Mozilla Firefox

http://127.0.0.1:3000/recipe/new

New recipe

Title

Instructions

Create

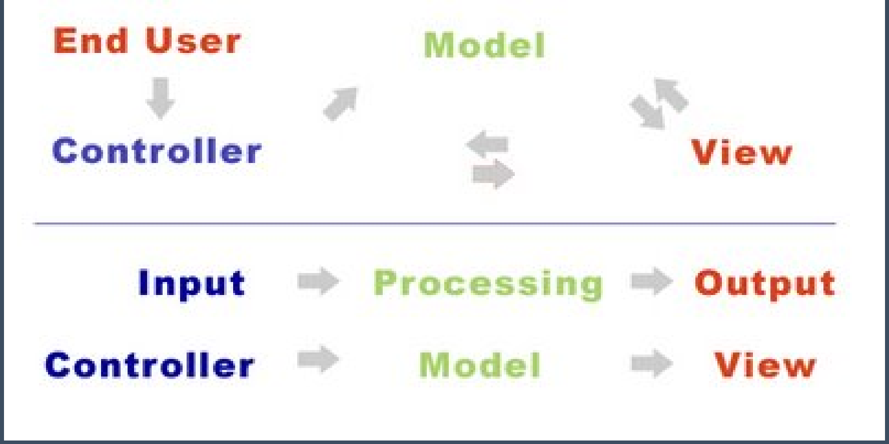
Back

Done

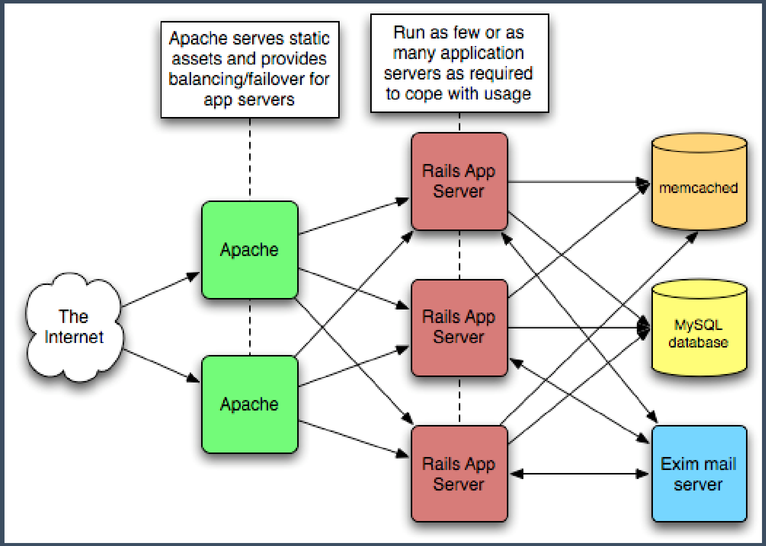
Figure 30. Creating a new recipe page [the View]

Source: http://www.onlamp.com/pub/a/onlamp/2005/01/20/rails.html?page=3

Implementing different time / different place systems



Model View Controller


Apache serves static assets and provides balancing/failover for app servers
Run as few or as many application servers as required to cope with usage
The Internet
Apache
Apache
Rails App Server
Rails App Server
Rails App Server
memcached
MySQL database
Exim mail server

Ruby on Rails

Source: http://railsruby.blogspot.com/2006/03/mvc-model-view-controller-architecture.html, http://www.rubaidh.com/services/hosting/system-architecture

Recap: What are Interface Toolkits?

- Goal: make it easier to develop user interfaces by providing application developers with reusable components that accomplish common input and output needs
- Toolkits have a well-planned architecture and API & provide a library

Drawbacks

- Can be limiting – developers are likely to make the kinds of UIs that the toolkit makes easy
- Traditional GUI toolkits are problematic for non-WIMP user interfaces such as:
  - Groupware
  - Physical UIs

Evaluating Toolkits

- Ease of use
  - A toolkit's API is a user interface, too! [Klemmer et al., 2004] evaluated the API of Papier-Mache
- Depth, Breadth, and Extensibility
- Systems issues
  - Speed
  - Portability

Current Research Challenges

- Complex design space
  - e.g., Do we have to update the toolkit every time someone creates a new sensor or actuator?
- Ambiguous input
  - Speech, gestures, computer vision, etc. aren't recognized as accurately as mouse clicks. Should the toolkit handle the recognition?

Summary

- I/O Toolkits provide reusable interface components to simplify UI development
- Toolkit trap: it's tempting to only make UIs that the toolkit makes easy, instead of making what's best for a specific app
- Toolkit types:
  - WIMP (Garnet, Swing, Motif, etc)
  - Speciality (Phidgets, iStuff, Papier-Mache, DiamondSpin, GroupKit, Peripheral Displays Toolkit, etc)

The Future of Design Tools

Supporting…

- Fieldwork
- Prototyping
- Collaboration
- Usability testing
- and emerging interface styles, such as
  - mobile
  - recognition-based UIs (speech, pens, vision)
  - information appliances
  - multiple devices

Announcements

- Experimental Participation
  - Everyone must have at least 1.5 units on CHIME
  - For those with less than 4 units on CHIME:
    - Either conduct a study of your prototype
    - Or participate in a study of someone else's
    - When you've done this, email ___.
- Midterm's have been upcurved
- Final Projects Presentations on 12/13 @7pm
  - Two parts: 1-minute madness, poster presentation

Further Reading

Books and courses on Building UIs

- *Introduction to User Interface Software.* Dan Olsen Jr. Morgan Kaufmann Publishers, 1998.
- Courses with notes online:
  - Carnegie Mellon University
    http://www.cc.gatech.edu/classes/AY2001/cs4470_fall/
  - Georgia Institute of Technology
    http://www.cs.cmu.edu/~hudson/teaching/05-631/