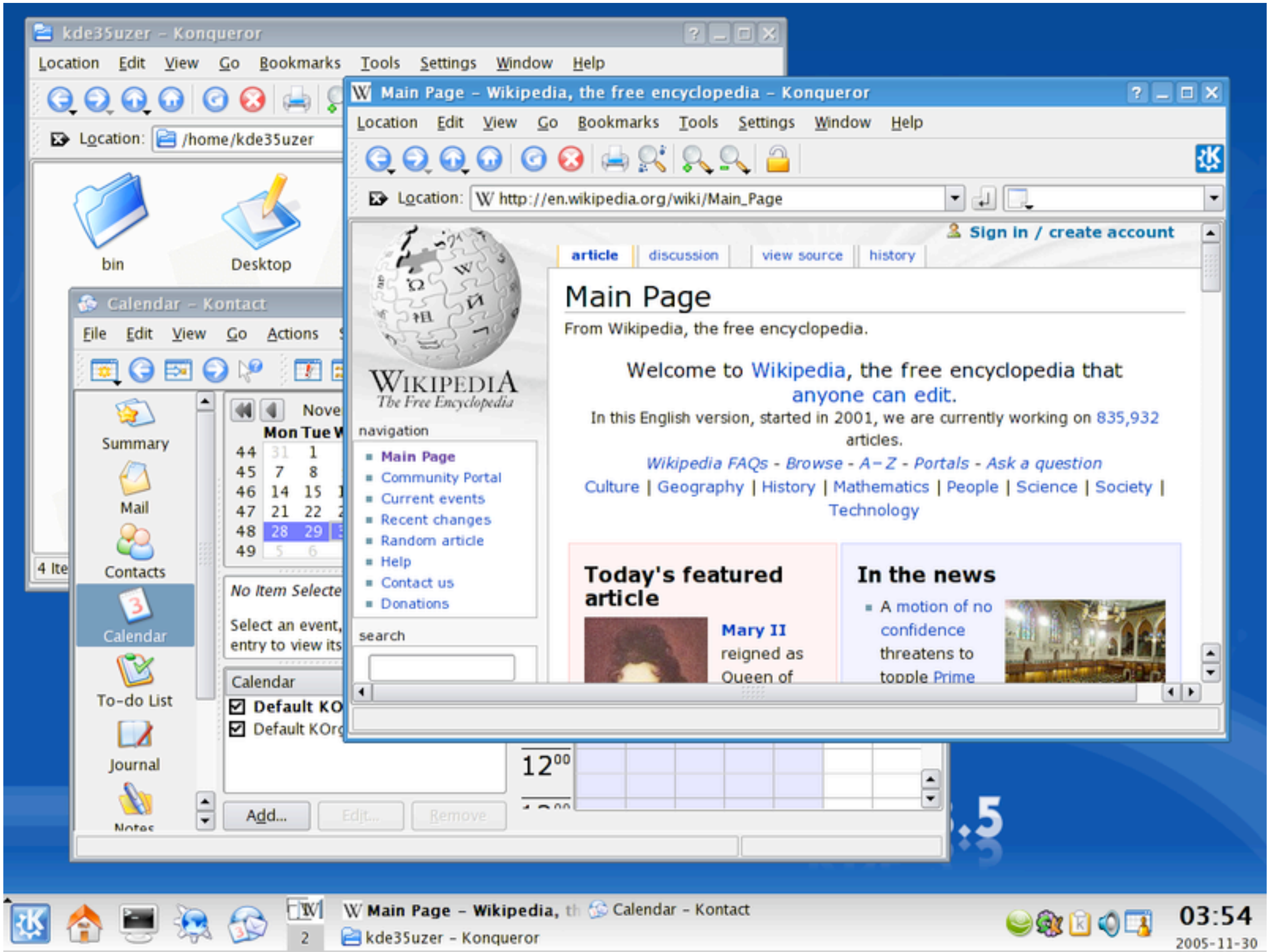
# CS 349
# Graphic Abstractions

Byron Weber Becker
Spring 2009

Slides mostly by Michael Terry

University of Waterloo



/home/kde35uzer

http://en.wikipedia.org/wiki/Main_Page

| | | | |
|---|---|---|---|
| 44 | 31 | 1 | |
| 45 | 7 | 8 | |
| 46 | 14 | 15 | 1 |
| 47 | 21 | 22 | 2 |
| 48 | 28 | 29 | 3 |
| 49 | 5 | 6 | |

*No Item Selecte*

Select an event,
entry to view its

☑ Default KO
☑ Default KOrg

# Main Page

From Wikipedia, the free encyclopedia.

Welcome to Wikipedia, the free encyclopedia that anyone can edit.

In this English version, started in 2001, we are currently working on 835,932 articles.

*Wikipedia FAQs - Browse - A–Z - Portals - Ask a question*

Culture | Geography | History | Mathematics | People | Science | Society | Technology

## Today's featured article

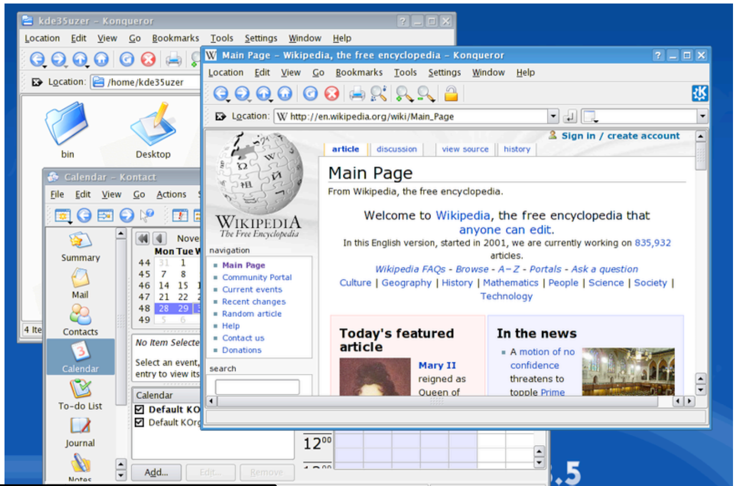
**Mary II** reigned as Queen of

## In the news

- A motion of no confidence threatens to topple Prime

Screenshot from http://en.wikipedia.org/

# Windowing Hierarchy



Main Page

From Wikipedia, the free encyclopedia.

Welcome to Wikipedia, the free encyclopedia that anyone can edit.

In this English version, started in 2001, we are currently working on 835,932 articles.

Wikipedia FAQs - Browse - A–Z - Portals - Ask a question

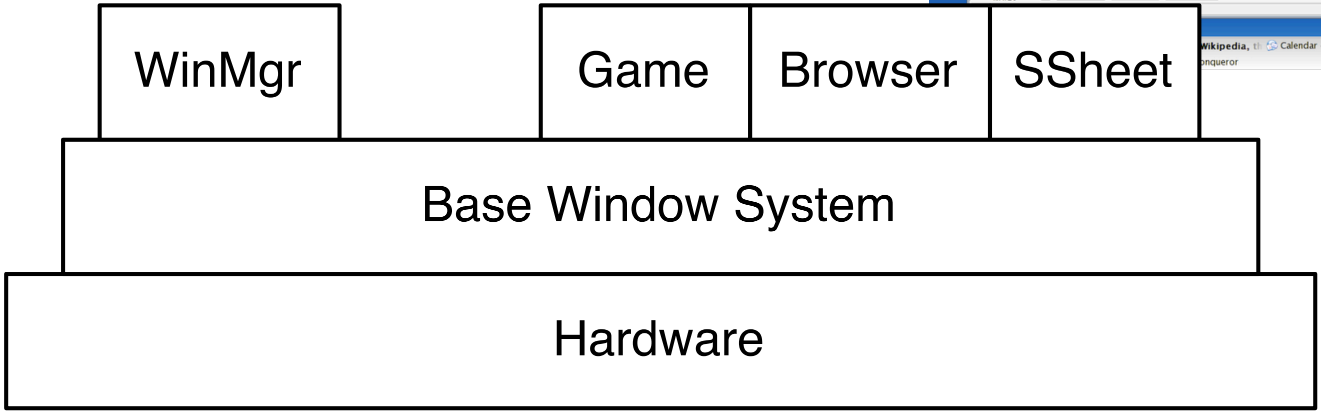
Culture | Geography | History | Mathematics | People | Science | Society | Technology

Today's featured article

Mary II reigned as Queen of

In the news

A motion of no confidence threatens to topple Prime



WinMgr

Game

Browser

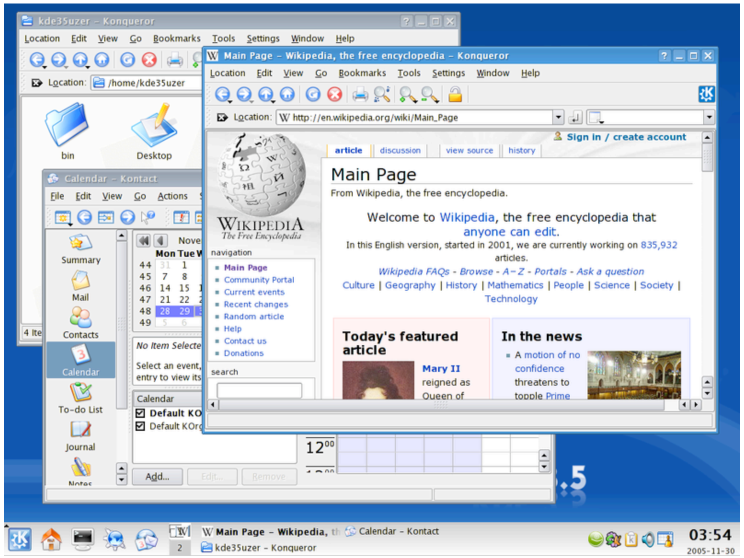
SSheet

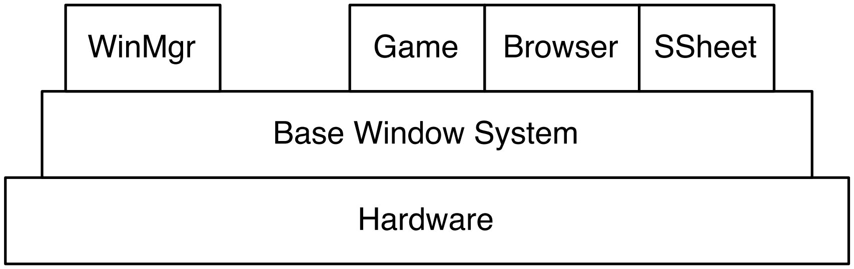
Base Window System

Hardware

# Base Window System

- Lowest level abstraction for windowing system
- Provides routines for creating, destroying, managing windows
- Routes input to correct window
- Ensures only one application changing frame buffer (video memory) at a time
  - Is one reason why you see only single-threaded / non-thread-safe GUI architectures




WinMgr

Game

Browser

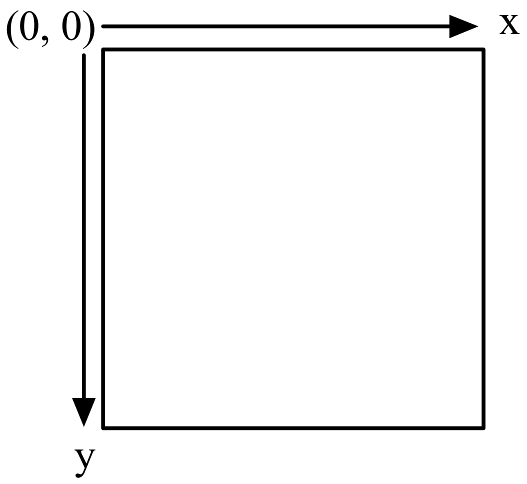
SSheet

Base Window System

Hardware

# Base Window System

- Creates canvas abstraction for applications
  - Applications shielded from details of frame buffer, visibility of window, other application windows
- Each window has its own coordinate system
  - Orientation varies
  - BWS transforms between coordinate systems
  - Each window does not need to worry where it is on screen, always assumes its top-left is (0,0)
- Provides basic graphics routines for drawing

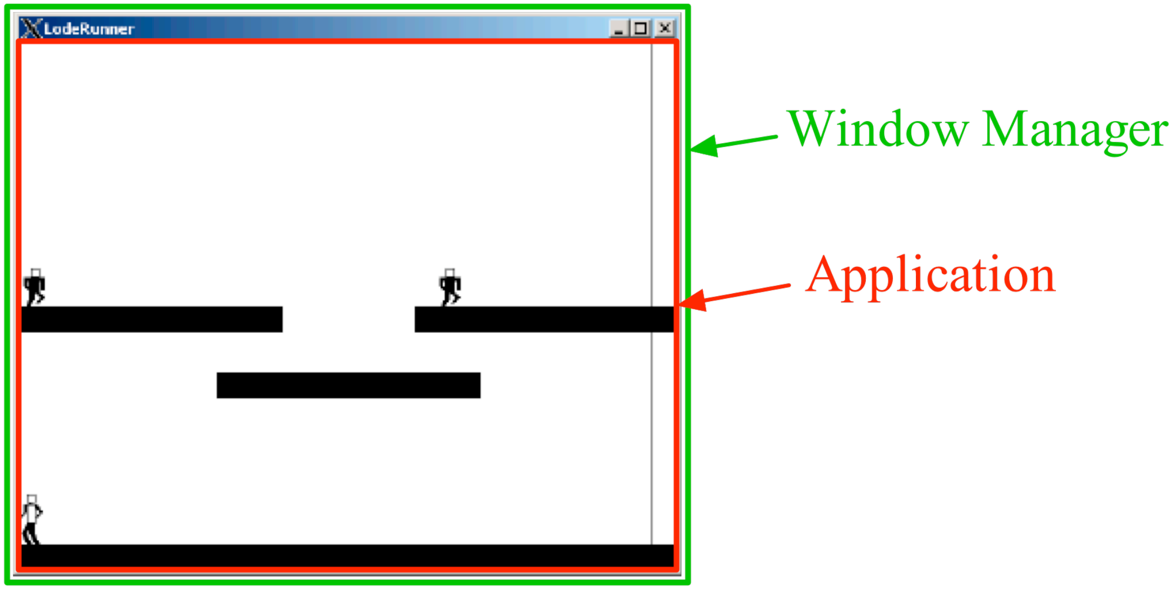


# Window Manager

- Window Manager provides conceptually different functionality
  - Layered on top of Base Window System
  - Provides interactive components for windows (menus, close box, resize capabilities)
  - Creates the “look and feel” of each window

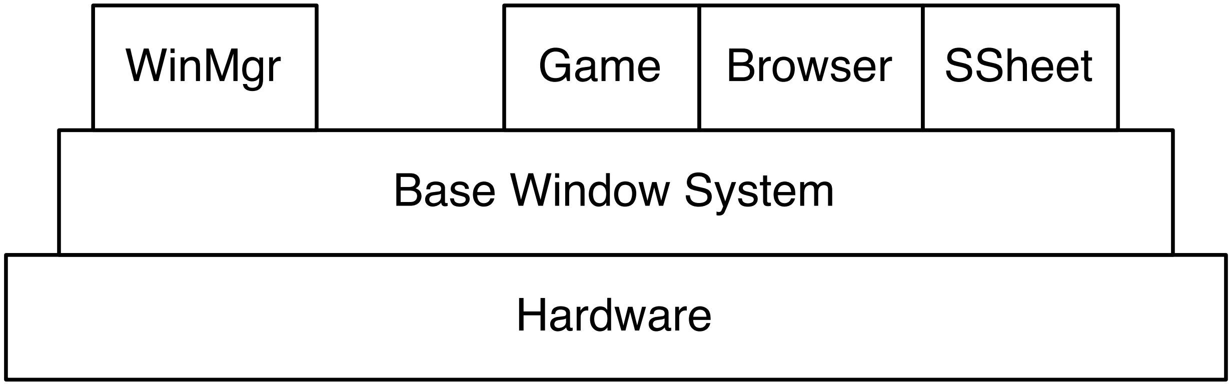
# Window Manager

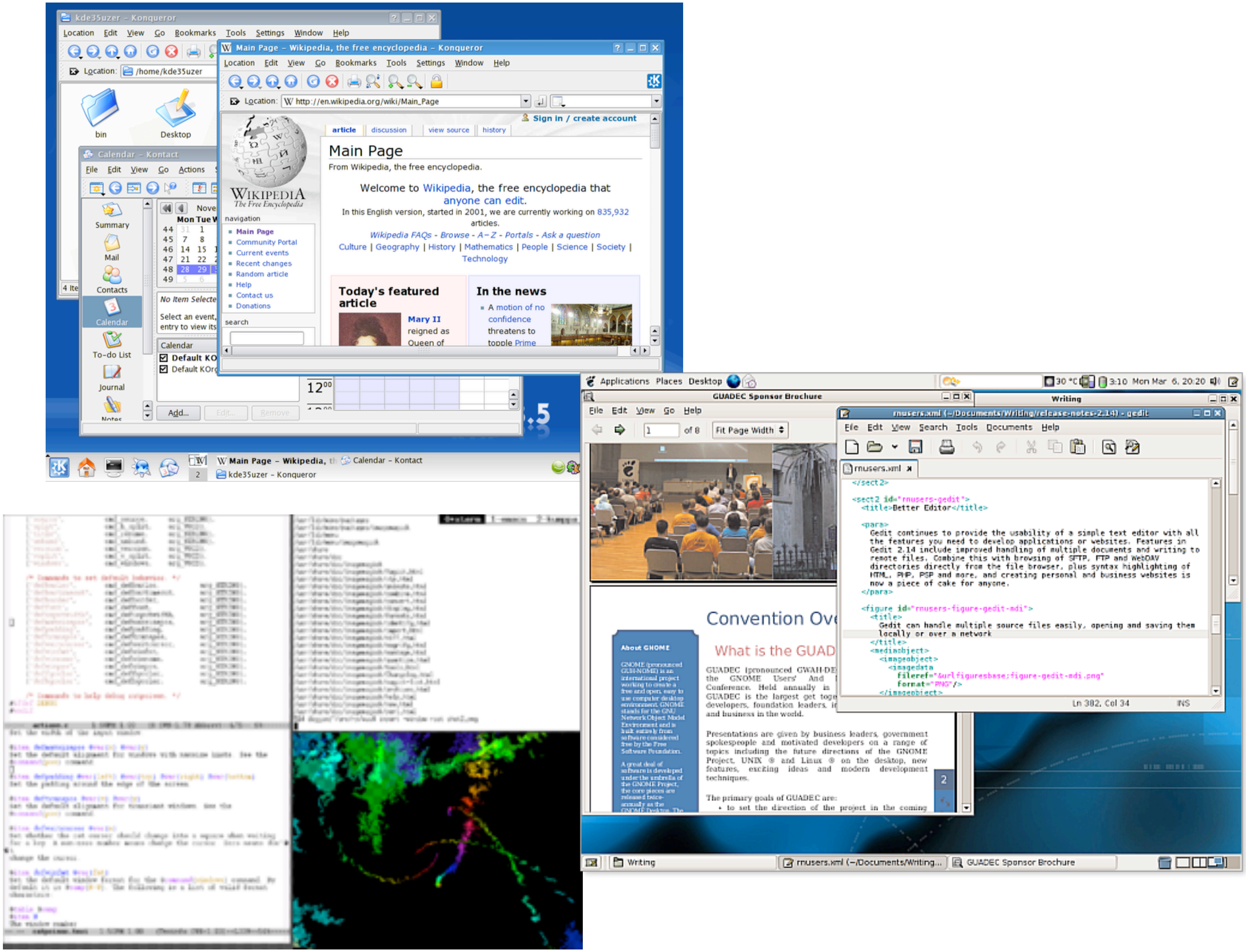
- Frame vs. content area (actual canvas)



# BWS vs. Window Managers

- X separates Base Window System from Window Manager
  - Enables many alternative “look and feels” for windowing system (e.g., KDE, GNOME, fvwm…)
  - One of the keys to its lasting power: Can continue to grow by changing the Window Manager layer
- Each a separate process





Main Page

From Wikipedia, the free encyclopedia.

Welcome to Wikipedia, the free encyclopedia that anyone can edit.

In this English version, started in 2001, we are currently working on 835,932 articles.

Wikipedia FAQs - Browse - A–Z - Portals - Ask a question

Culture | Geography | History | Mathematics | People | Science | Society | Technology

Today's featured article

Mary II reigned as Queen of

In the news

- A motion of no confidence threatens to topple Prime

Convention Ove

About GNOME

GNOME (pronounced GUH-NOME) is an international project working to create a free and open, easy to use computer desktop environment. GNOME stands for the GNU Network Object Model Environment and is built entirely from software considered free by the Free Software Foundation.

A great deal of software is developed under the umbrella of the GNOME Project, the core pieces are released twice-annually as the GNOME Desktop. The

What is the GUAD

GUADEC [pronounced GWAH-DE the GNOME Users' And Conference. Held annually in GUADEC is the largest get toge developers, foundation leaders, i and business in the world.

Presentations are given by business leaders, government spokespeople and motivated developers on a range of topics including the future directions of the GNOME Project, UNIX ® and Linux ® on the desktop, new features, exciting ideas and modern development techniques.

The primary goals of GUADEC are:

- to set the direction of the project in the coming

```
</sect2>

<sect2 id="rnusers-gedit">
  <title>Better Editor</title>

  <para>
    Gedit continues to provide the usability of a simple text editor with all
    the features you need to develop applications or websites. Features in
    Gedit 2.14 include improved handling of multiple documents and writing to
    remote files. Combine this with browsing of SFTP, FTP and WebDAV
    directories directly from the file browser, plus syntax highlighting of
    HTML, PHP, PSP and more, and creating personal and business websites is
    now a piece of cake for anyone.
  </para>

  <figure id="rnusers-figure-gedit-mdi">
    <title>
      Gedit can handle multiple source files easily, opening and saving them
      locally or over a network
    </title>
    <mediaobject>
      <imageobject>
        <imagedata
          fileref="&urlfiguresbase;figure-gedit-mdi.png"
          format="PNG"/>
      </imageobject>
```

Screenshots from http://en.wikipedia.org/

# BWS vs. Window Managers

- Macintosh, Windows bundle Base Window System and Window Manager together
  - *Very difficult* for 3rd party to achieve alternative look and feel
- Trade-offs in approaches?
  - Look and feel…
  - Window management possibilities…
  - Input possibilities…

# Windows and Components

- Window
  - The high-level unit managed by window manager
  - Has canvas
  - Contains windows/components
- Component
  - Individual elements within window user sees, interacts with
  - Label, slider, text component, etc.
- Components are either:
  - Actual windows themselves (e.g., X, MS Windows)
  - Objects implemented by the GUI toolkit (Java)
- You will be building a *lightweight* component toolkit for assignment

# Canvas

- Every system provides a *Canvas*-like abstraction
- The method by which one draws in the window
  - Canvas represents window's content area
- Canvas more than a "surface"
  - A "surface" *and* a set of routines for manipulating that surface
  - DrawLine(), DrawRectangle(), DrawString()…
- Graphics context (state)
  - State representing parameters for future drawing operations
  - Foreground color, line width, font…
  - Clip

# Canvas

- Division of entities not standardized
- Examples:
  - X: Display + XLib routines + GC
  - Java: Component + Graphics/Graphics2D object
  - Mac OS X's Cocoa: NSView + NSGraphicsContext
  - Windows: Device Context + Graphics Device Interface (GDI)
- Java rolls a lot into Graphics/Graphics2D object
  - Graphics context (foreground color, font, clip…)
  - Drawing routines
  - *Only* way to manipulate canvas's graphics

# Graphics Abstraction

- Abstraction of drawing routines useful to create *device independence*
  - In theory, same drawing routines, regardless of where output rendered (e.g., CRT, LED display, printer)
  - Don’t need to know hardware capabilities, just draw
- But devices *do* matter, in some cases
  - Output to screen vs. printer

# Graphics Abstraction Issues

- Rendering to printer rather than display
  - Papers have “hard” edges, so content can’t go on endlessly
  - Pagination issues with printing
  - Resolution a big deal with printing
    - Huge differences in DPI (dots per inch) between display devices, printing devices
    - 72 DPI vs. 300 DPI
  - So same drawing routines for screen and printer desirable, but not exactly possible
- Color models also an issue
  - RGB vs. CMYK

# The Clip

- Need to ensure each window/canvas only draws in its own area
- Need to optimize drawing routines
- *Clip* provides this functionality

# The Clip

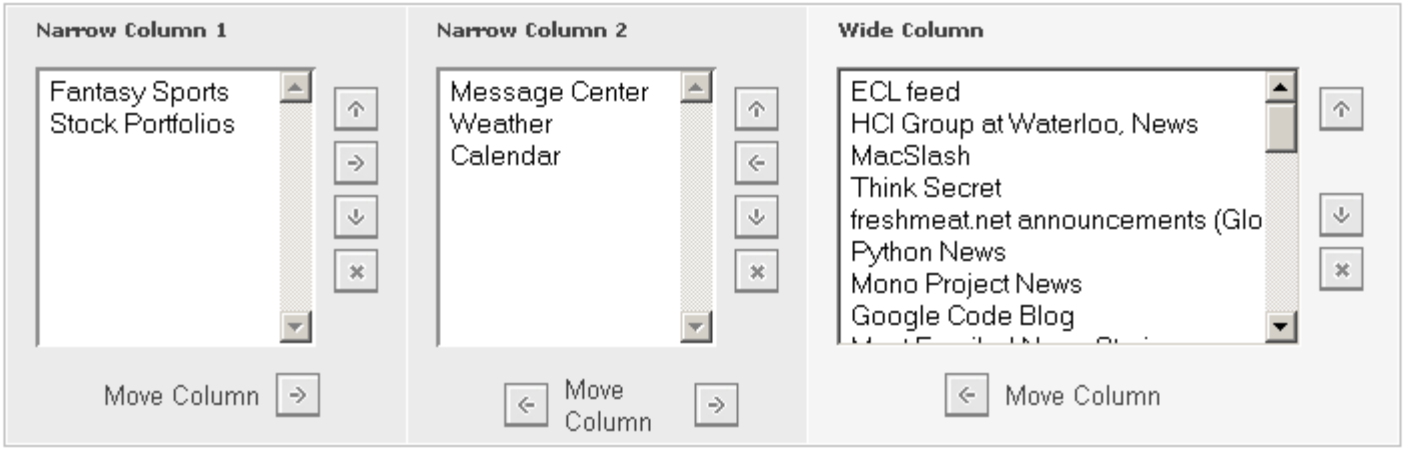
- A (potentially) arbitrary region that defines where drawing operations will / will not have effect
- Part of the GC (graphics context) in X
- Part of Graphics/Graphics2D in Java
- Manipulable by programmer, but can never draw outside of your own window

```
    XRectangle      clip_rect;
    clip_rect.x = 0;
    clip_rect.y = 20;
    clip_rect.width = 30;
    clip_rect.height = 40;

    while (1)                       // event loop
    {   XEvent event;
        XNextEvent( display, &event );
        switch( event.type ) {
            case KeyPress:
                clip_rect.x += 10;
                XSetClipRectangles(display, gc, 0, 0, &clip_rect, 1, Unsorted);
                repaint(display, window, gc);
                break;
             …
        }}}

void repaint(Display* display, Window window, GC gc)
{   XClearWindow( display, window );
    XDrawString(display, window, gc, 30, 50, "String test", strlen("String test"));
    XDrawLine(display, window, gc, 30, 50, 200, 50);
    XFlush(display);
}
```

# Interactor Tree



Fantasy Sports
Stock Portfolios

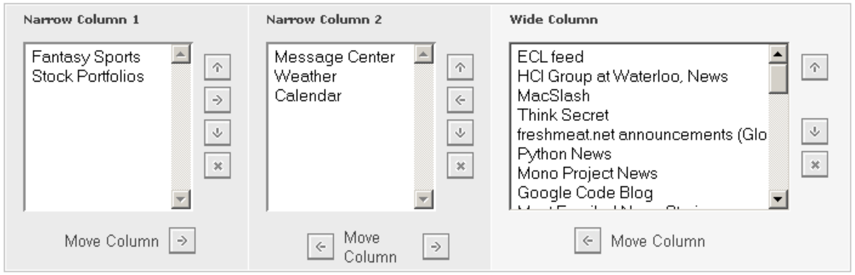
Message Center
Weather
Calendar

ECL feed
HCI Group at Waterloo, News
MacSlash
Think Secret
freshmeat.net announcements (Glo
Python News
Mono Project News
Google Code Blog

- Components contained within components
  - *Containment hierarchy* or *interactor tree*

# Interactor Tree

- Window
  - First Panel
    - Narrow Column 1 Label
    - List Box
    - Up arrow
    - Right arrow
    - …
  - Second Panel
    - Narrow Column 2 Label
    - List Box
    - ...
  - Third Panel...



Fantasy Sports
Stock Portfolios

Message Center
Weather
Calendar

ECL feed
HCI Group at Waterloo, News
MacSlash
Think Secret
freshmeat.net announcements (Glo
Python News
Mono Project News
Google Code Blog

# Interactor Tree

- Interactor tree represents a hierarchy of containers
  - Components contained within components
- Containment hierarchy helps decide where to target events
- Hierarchy also used for *drawing* components
  - Child components “draw” within parent components
  - Child’s painting is clipped to parent component’s bounds; parent likely restricts child even further

# Coordinate Systems

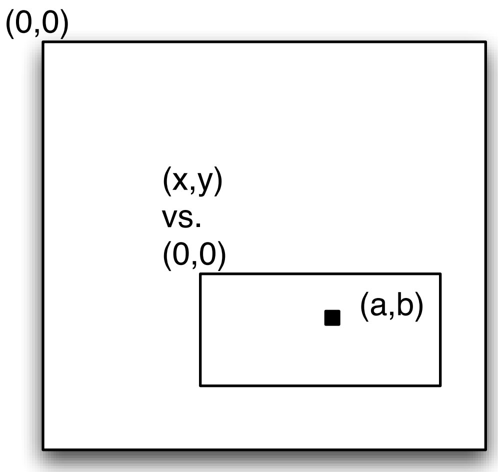
- Component’s location and bounds are represented in coordinate system of parent component
- But…
- …Drawing within component is assumed to be relative to component’s top-left corner
  - Top-left corner is always (0, 0)
- Has important consequences for delivering event information to components, drawing into components…

# Drawing in Windows/Components

- Need elegant way to set up drawing routines so they are always relative to top-left corner of component
- In X, MS Windows, where a window == a component, base window system automatically sets up coordinate system so drawing routines are relative to component
- For lightweight architectures (Java’s Swing, your assignment), coordinate system needs to be explicitly set

# Drawing in Components

- Proper coordinate system set up in an *affine transform*
- Affine transform a 3x3 matrix representing translations, scales, rotations of coordinate system
  - (Demo)
- In Java, proper transform already set up in Graphics2D when you are asked to paint yourself in the component
- In your architecture, will need to set it up in Graphics
- Key method: Translate()


(0,0)
(x,y)
vs.
(0,0)
(a,b)