History of GUIs, cont., and FLTK: Fast Light ToolKit

8 January 2009
CMPT370
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

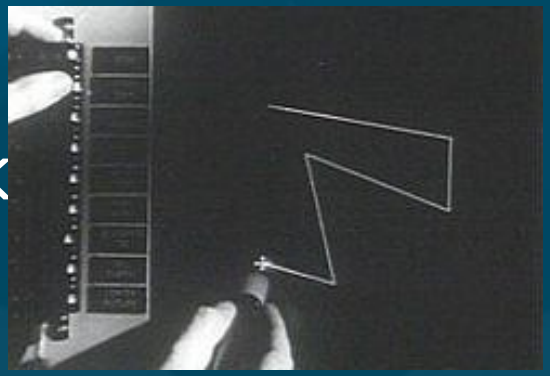
What's on for today

- An excerpted history of GUIs, cont.
  - Sutherland, Engelbart, PARC, Apple, MS
- GUI toolkits
- Events and callbacks
- FLTK and Fluid
  - BankInterest example
  - Template lab write-up

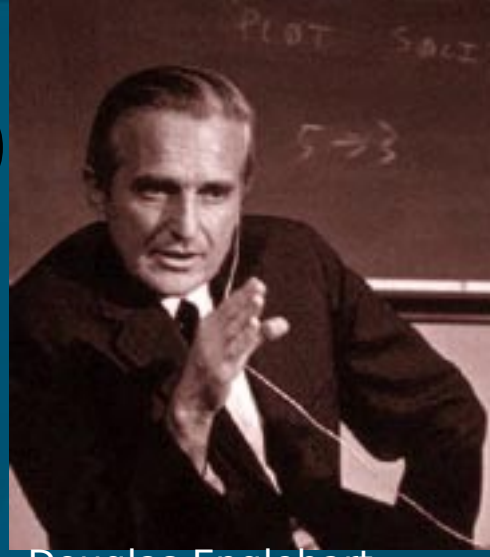
- Next time: UI/HCI design issues

Sutherland's SketchPad (1963)

- Ivan Sutherland
  Ph.D. thesis at MIT
- Used light pen to directly manipulate graphical objects on screen
- Pioneer of computer-aided drafting (CAD):
  - Draw "master" diagram once
  - Instantiate multiple copies, tweak (OO design)
  - Constraint-based system (e.g., keep two lines at fixed angle)



Engelbart's NLS demo (19



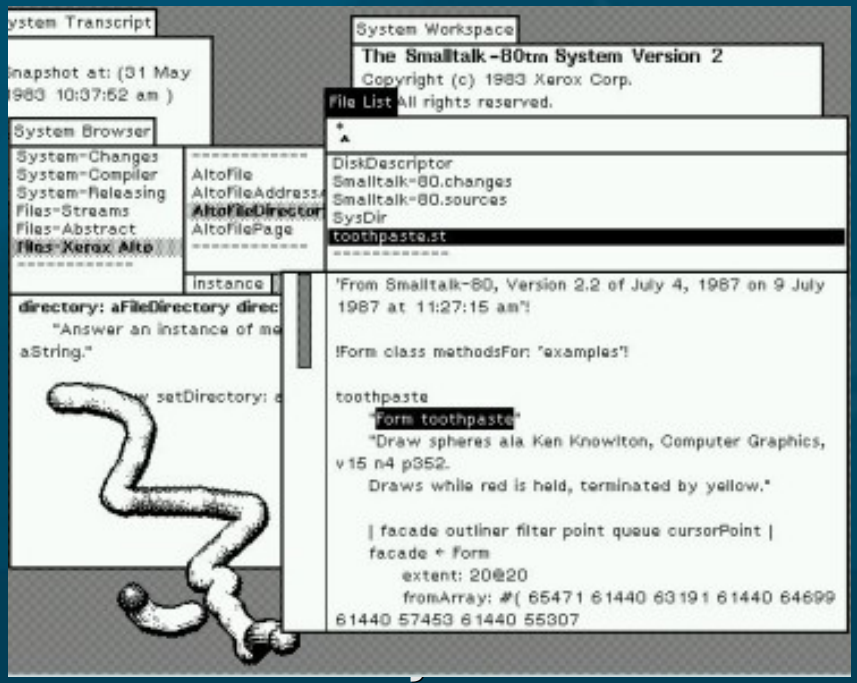
Douglas Englebart,
Stanford Research Inst.

- NLS (oNLine System) innovations:
  - Mouse
  - Windowing system
  - Collaborative document editing with email, IM, and video conferencing
  - Hyperlinks
  - Chording keyboard

Xerox PARC in the 1970'

Smalltalk
on the Star

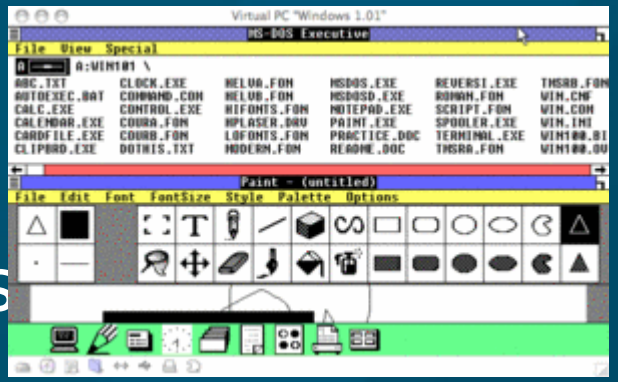
- Xerox Palo Alto:
  - Towards “paperless office”
  - Microcomputers: Alto (1973), Star (1981)
  - WIMP model: windows, icons, menus, pointer
  - Desktop
  - Smalltalk (1974):
    - Pure OO language
    - Integrated graphical development and runtime environment



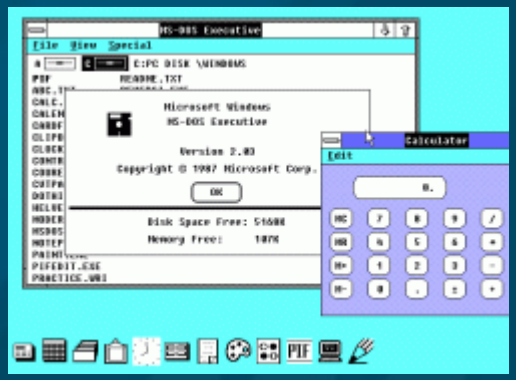
Apple in the 1980's

- Lisa (1983):
  - Drag-and-drop
  - Double-click to open/run
- Macintosh (1984):
  - Much cheaper ($2,495 vs. >$10k)
  - Accessible to the public
  - Mass-marketing ad campaign during SuperBowl and 1984 Olympics in L.A.

Microsoft Windows (1980's)



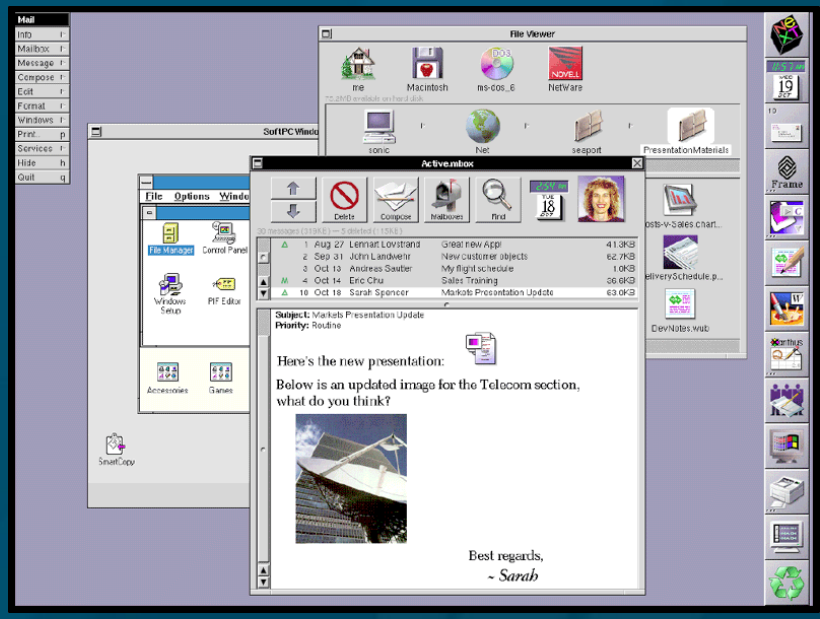
- Windows 1.0 (1985):
  - Mostly character-based graphics
  - Tiled windows
  - Popularity dwarfed by Mac



- Windows 2.0 (1987):
  - Overlapping windows
  - Apple sues MS over “look and feel” (loses)
- Windows 3.11 (1992), Win95:
  - Looks pretty; wildly popular

Other GUI environments

- GEM (Digital Research) for Atari (1985)
- Amiga Workbench (1985)
- NeXTstep (Steve Jobs) (1988)
  - Pretty, but CPU-intensive
- OS/2 (IBM) (1988):
  - competed with Windows
- Unix X10 (1984), X11 (1987)
  - Network transparency (Xwin32)
  - Multiple libraries on top: Athena, Motif/CDE, OpenLook, KDE/Qt, Gnome/gtk, FLTK



NeXTstep

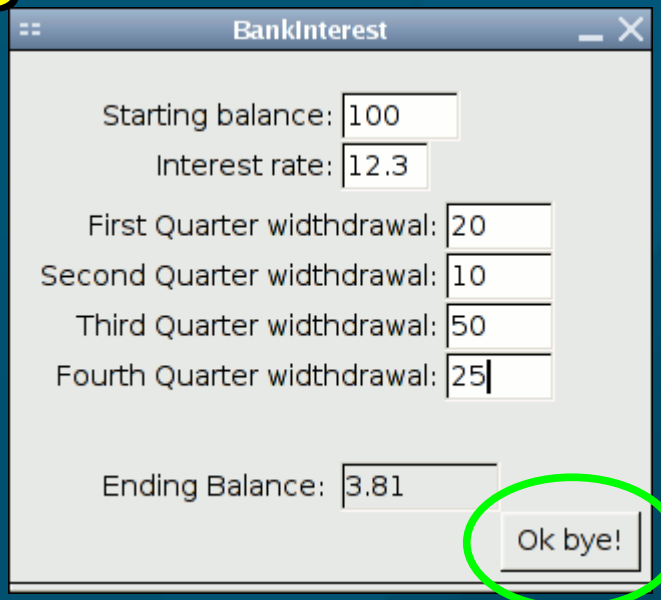
OS environment vs. toolkit

- In the past, the only GUI was what was provided by the operating system
- Now, we can write programs that link to various GUI toolkits:
  - Libraries that provide a way to build a GUI program
  - Menus/windows that look just like Windows:
    - Link with MFC or Visual Basic or .NET
  - Other options: FLTK, Qt, wxWidgets, gtk, ...
    - Cross-platform: can run on Linux, Mac, etc.

Compiling with GUI toolkits

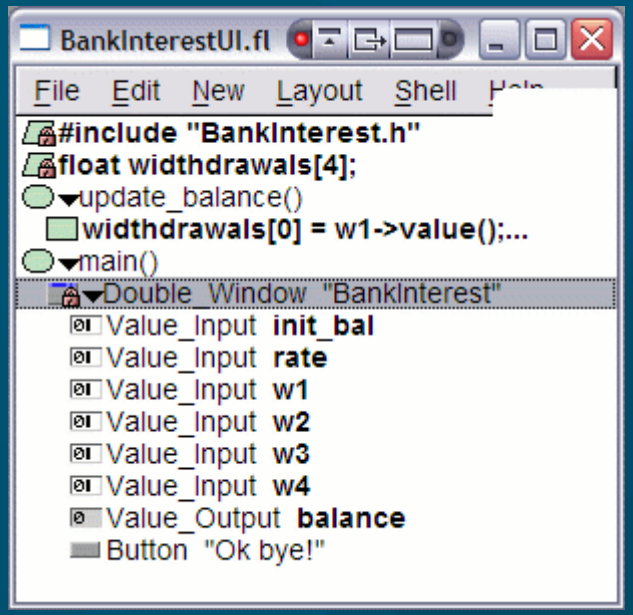
- Libraries provide GUI components as objects
  - Windows (Fl_Window), menus, tabs, etc.
  - Widgets: buttons, textboxes (Fl_Input), sliders, scrollbars, dials, etc.
- Link your program with the toolkit library
  - Static linking: libfltk.a
    - Needed objects are bundled into the executable
  - Dynamic linking: libfltk.dll.a / libfltk.so
    - Need separate shared library
  - FLTK-1 libraries: fltk, fltk_gl, fltk_images, fltk_forms

Events and callbacks



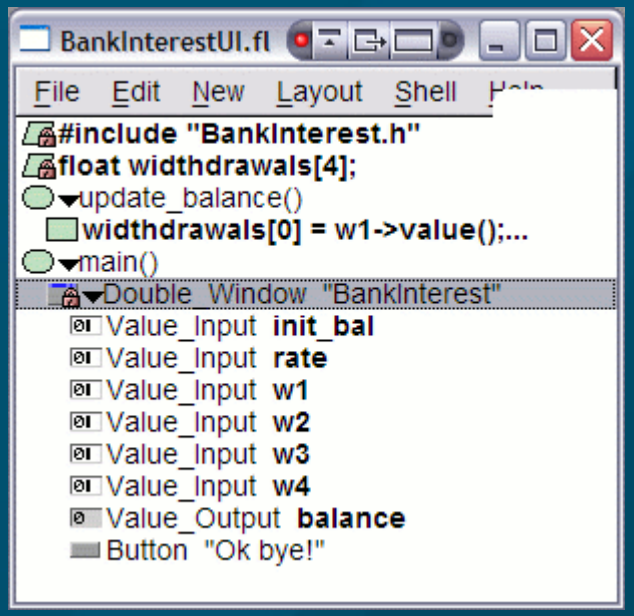
- An event is a user action:
  - Click a button
  - Fill in a text box
  - Press a key
  - Move the mouse
- A callback is a procedure invoked by an event:
  - Close the window when user clicks "Ok bye!"
  - Draw a circle where the user clicks the mouse

Using Fluid

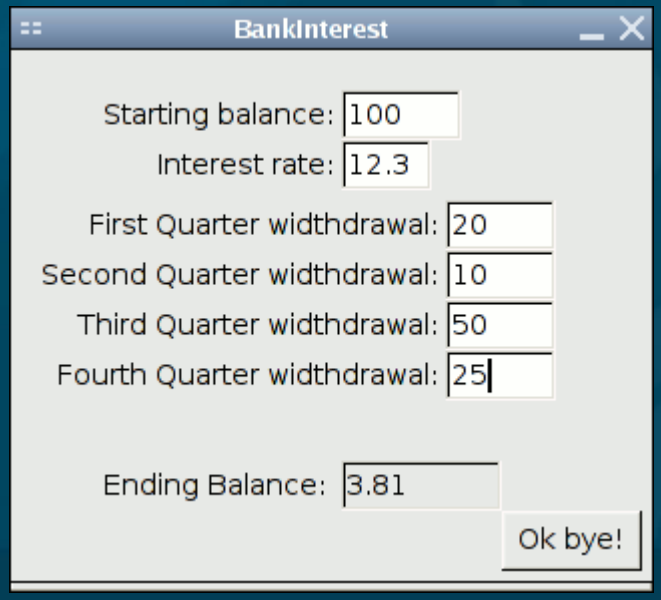


- Fluid is FLTK's interactive GUI designer
  - Drag and drop widgets
  - Write code blocks / callbacks
- Saves *.fl Fluid files; exports *.cxx/*.h code
- Compile and link this code into your program
- It is possible to write a whole program in Fluid
- But better to separate GUI from main program logic: form vs. function
  - Akin to HTML/CSS vs. PHP/ASP/JS

FLTK example: BankInterest



- BankInterestUI:
  - Just the user interface
  - Get values from the widgets
  - Minimal program logic
    - But I did choose to put main() here
- BankInterest:
  - Main program functionality
  - Provides functions invoked by UI callbacks
    - calc_balance()



Lab write-ups (see BankInterest)

- Purpose: a sentence or two
- Suitability: do we even need a computer to do this? Or a GUI? Or carmel?
- Restatement: given, todo, result
- Libraries: libfltk, GL, etc. (w/versions if possible)
- Program Design
- User Manual
- Test Cases
- Bugs / Limitations / future work
- Screenshots

Carmel

- Next topic (next Thu): parallel programming
- We'll be using carmel's “8” processors
  - OpenMP under gcc4
  - Linux command-line (ssh/PuTTY); no GUI
- If you don't have a carmel login yet, Dave Friesen can get you one for this class
- We'll get to this next week, but you can get started early if you like:
  - Download nbody and runtest to carmel
  - './runtest' and watch the results

TODO

- Brush up on your C++
  - Links at bottom of our IDE policy sheet
- Lab0 due next Tues 14Jan
  - FLTK orientation, tutorials
  - No write-up needed
  - Upload ZIP/tarball to myCourses by midnight
- Lab1 due Tue 21Jan
  - Design + implement your own FLTK program
  - Should be “cool” and somewhat “useful”