

# Interaktionsprogrammering

## TDDC73

Anders Fröberg

[anders.froberg@liu.se](mailto:anders.froberg@liu.se)

# Outline

- Historic perspective
  - Where are we coming from
  - Why do things look like they do
- Course outline

# Before we start

- Both at LiU and via Teams
  - Designed and planned for at LiU
  - But will stream and record on Teams.
- Feedback from last year
  - Getting started guide to easy
  - Project easier than the lab

West of House 0/0

In the beginning there was a command line

ZORK I: The Great Underground Empire  
Infocom Interactive fiction - a fantasy

Entire 14 inch screen devoted to a single application

Copyright (c) 1981, 1982, 1983, 1984,  
1985, 1986 Infocom, Inc.

All rights reserved.  
ZORK is a registered trademark of  
Infocom, Inc.

Release 52 / Serial number 071125 /

Users operated advanced system through natural language like syntax

And it was all good (especially for developers)

You are standing in an open field west  
of a white house, with a boarded front  
door.

And then came the gui and it was all down hill from there (for the developers)

>fuck mailbox

I do not know the word "fuck."

>\_

# ..and then came the GUI



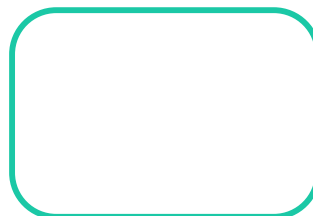
..and then came the GUI



..and then came the GUI

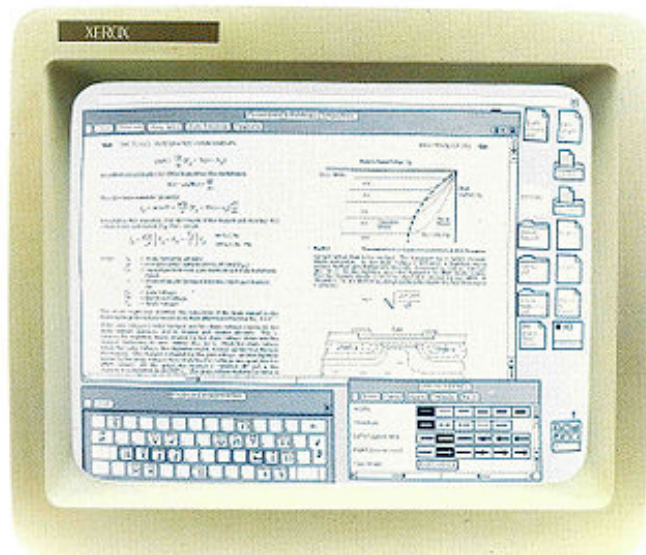


..and then came the GUI

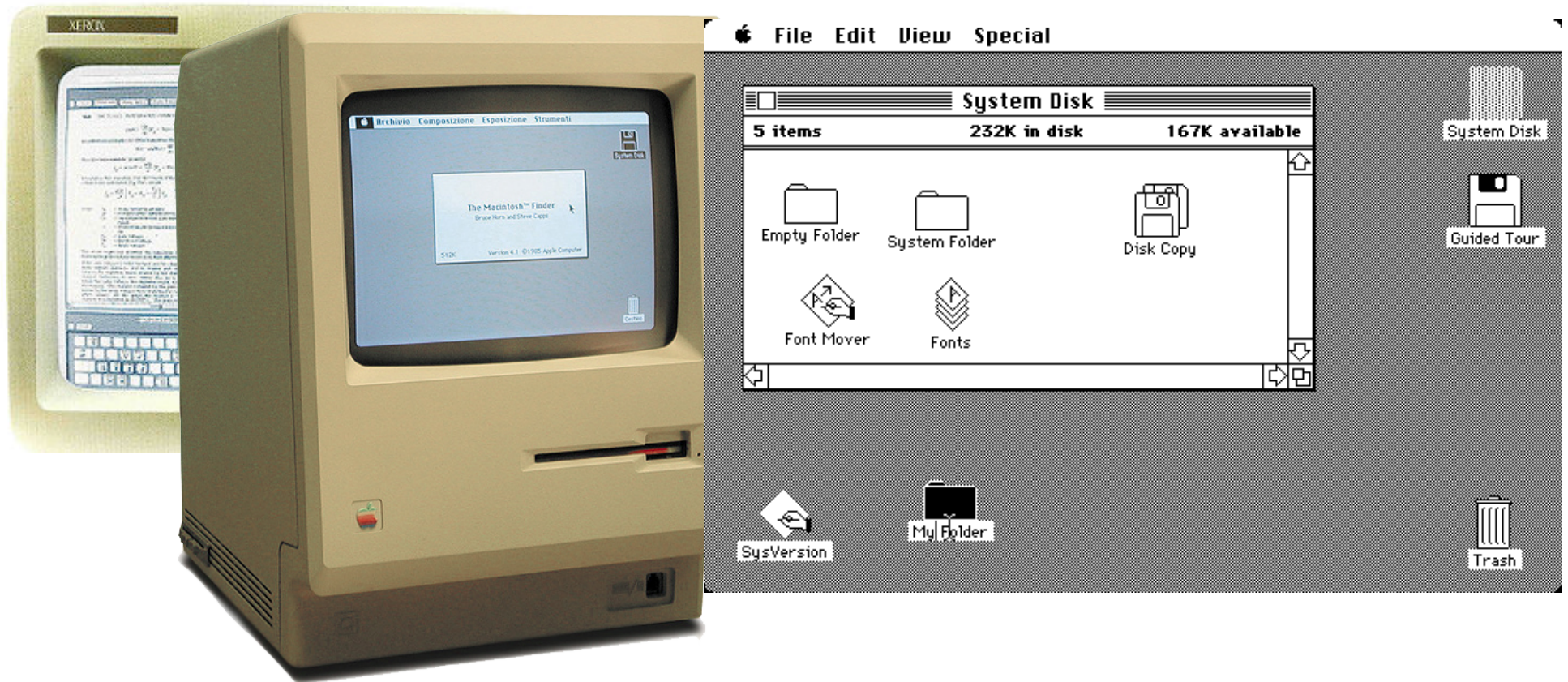




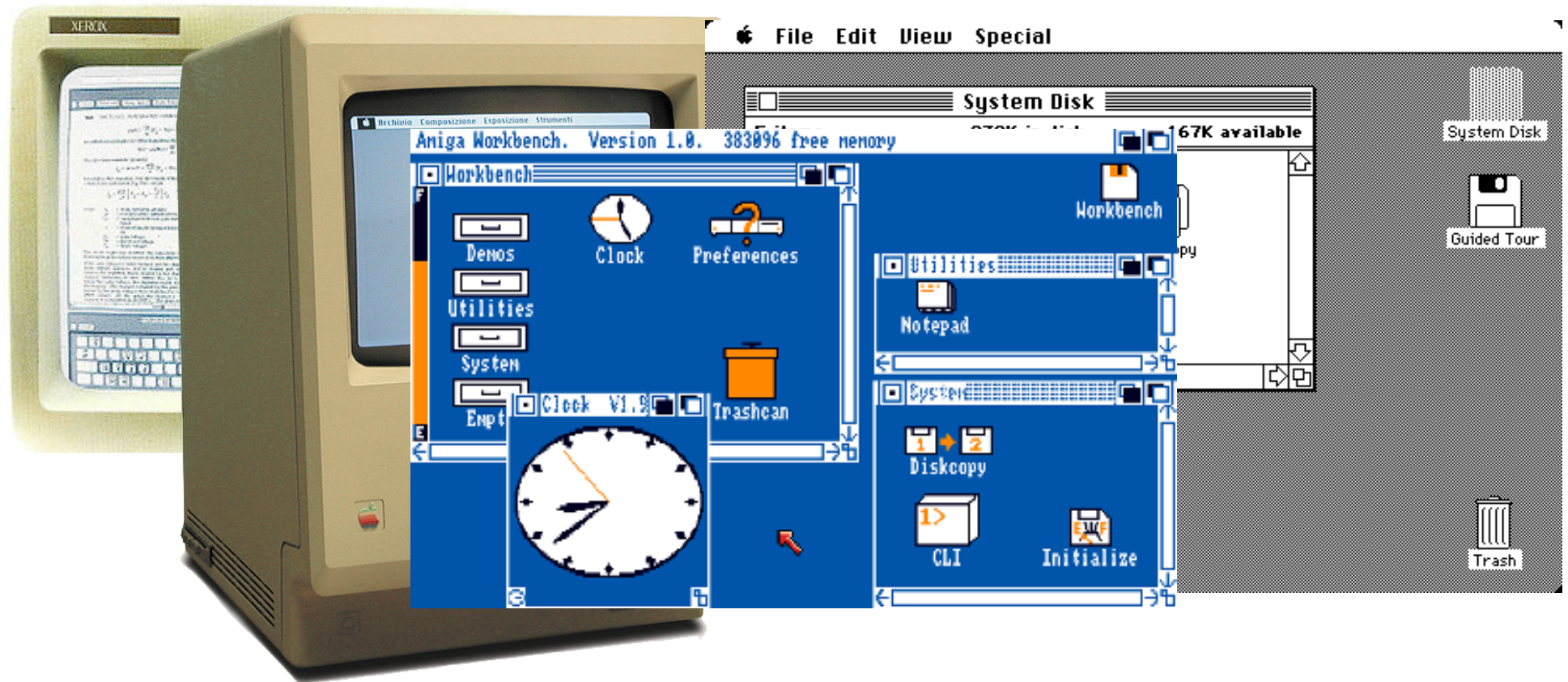
..and then came the GUI



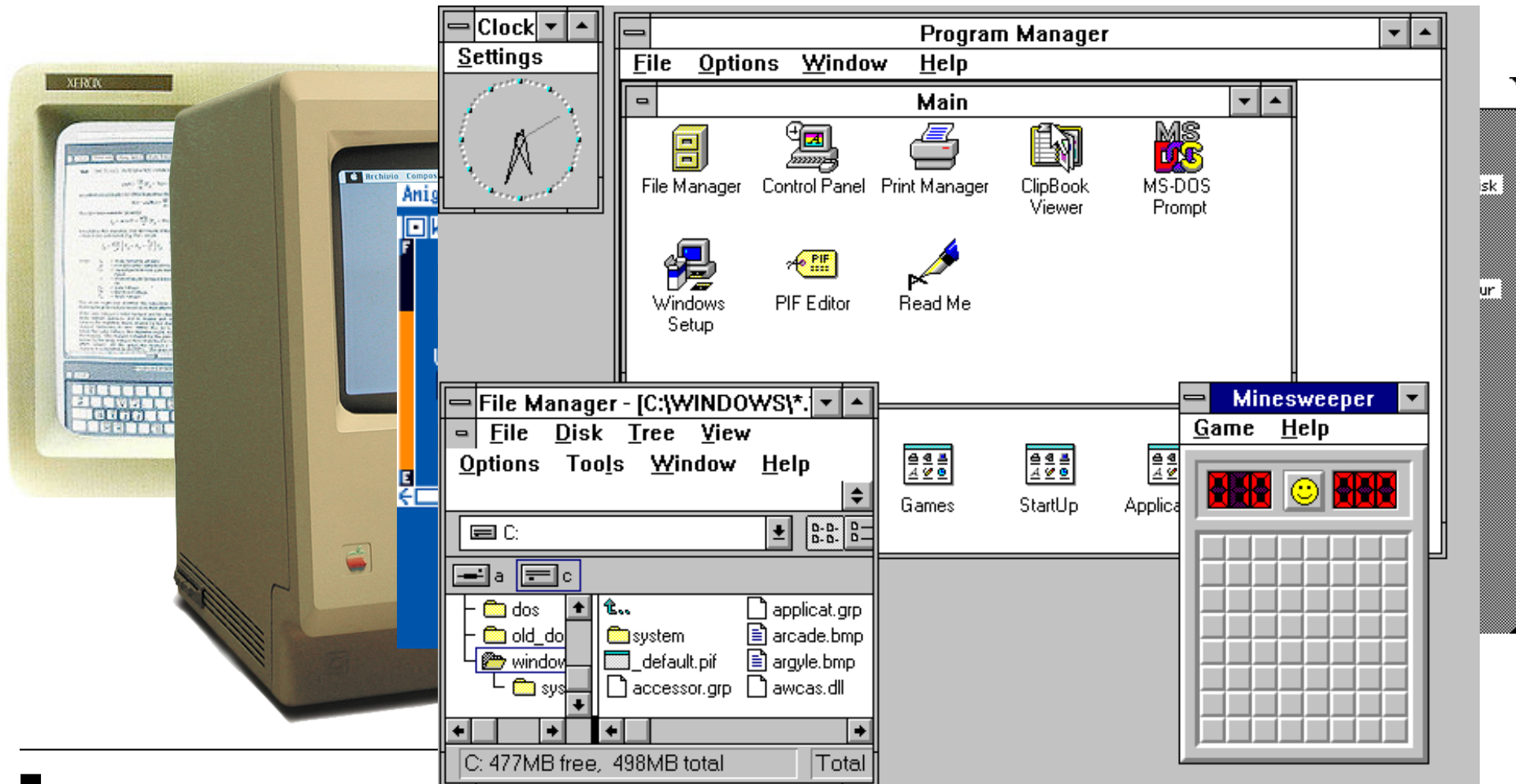
..and then came the GUI



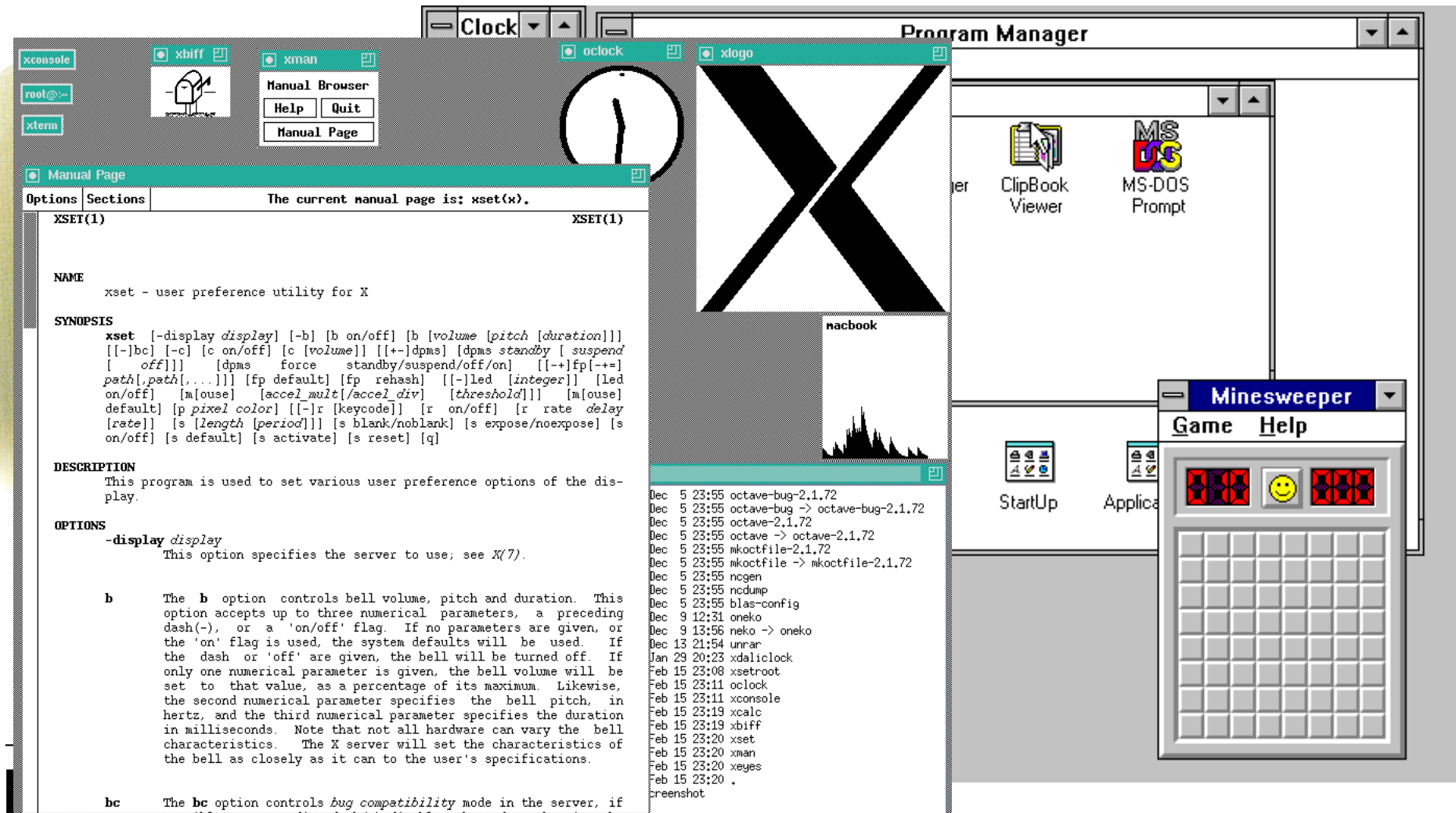
..and then came the GUI



..and then came the GUI

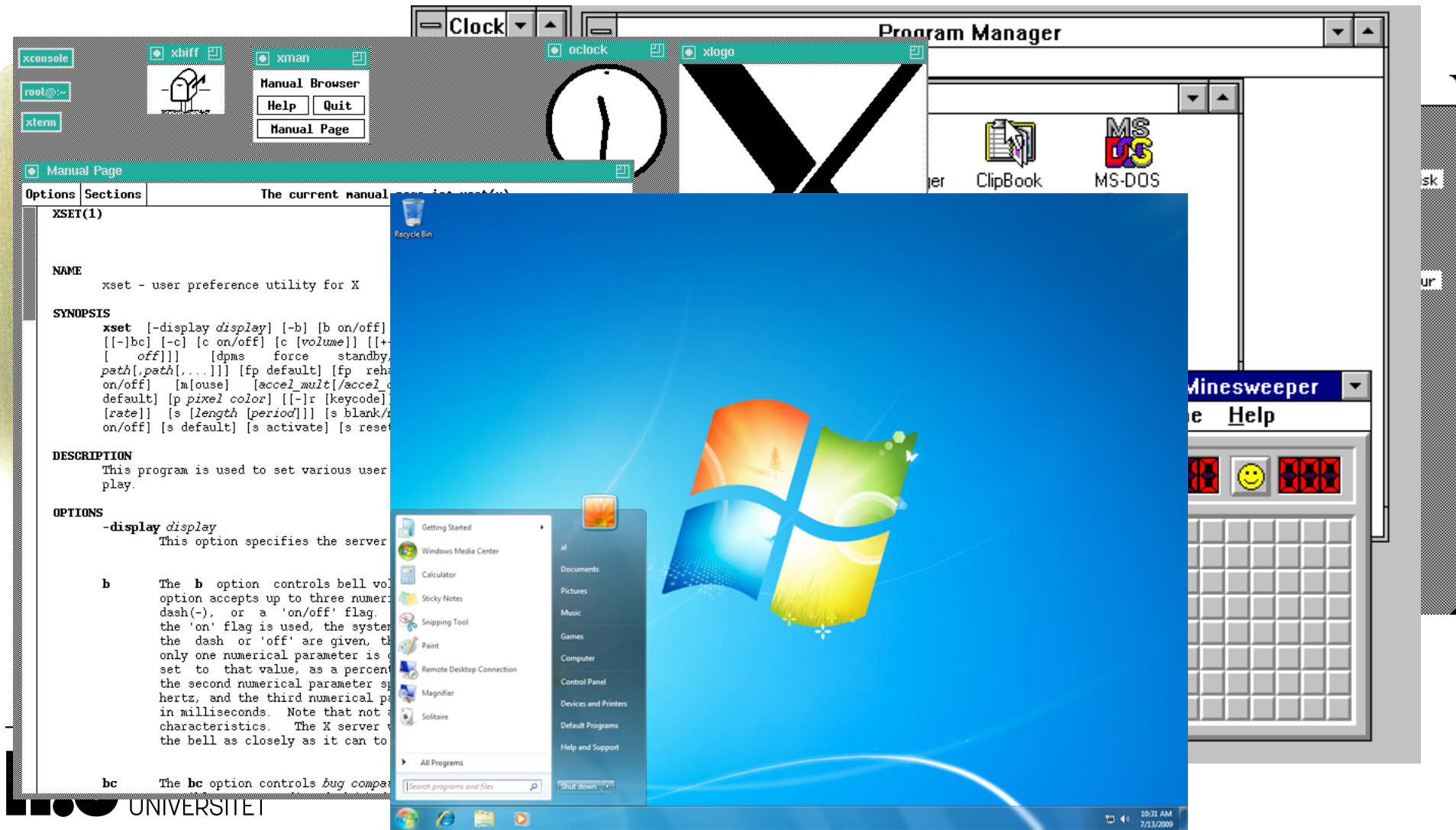


# ..and then came the GUI

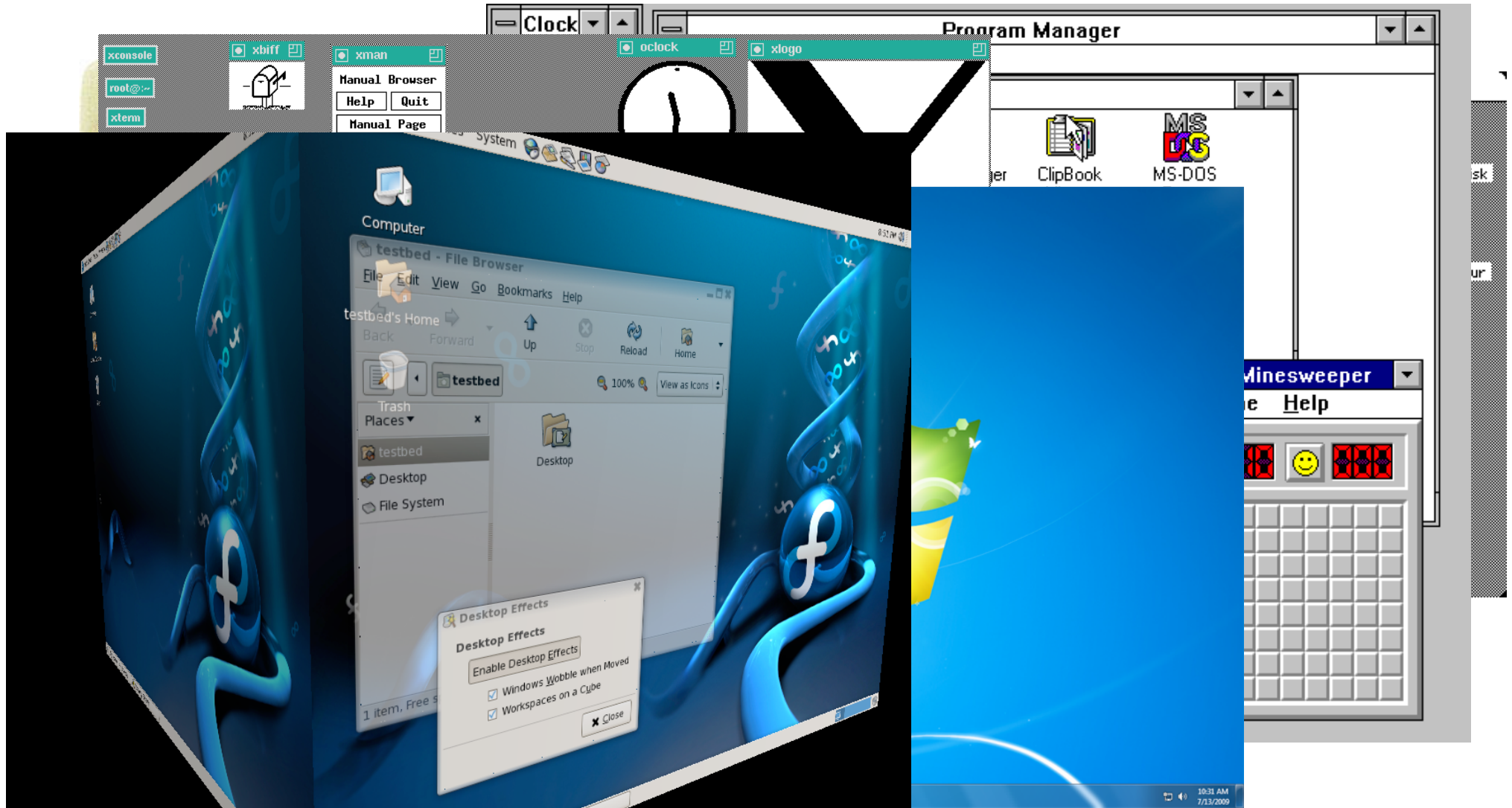


sk  
ur

# ..and then came the GUI



..and then came the GUI









# Command line ->GUI

- New problem arises for developers
- Complex interaction structure
  - Elaborate graphics
  - Multiple interaction mechanisms
  - Non-linear command structure

# Solutions

- Iterative methods
- Software Development Kits (SDKs)
  - Swing
  - Motif
  - .Net Forms
  - React React-Native
  - Flutter
  - Vue
  - Svelte
- GUI builders
- Specialized GUI languages

## Some stats

- Already in '94 almost all Unix contained a GUI
- Half to code is GUI code
- Time spend equal all other parts together
- Benefit Tools
  - Reduces code by 83%
  - Time spend reduced by a factor 4 (Building GUI)

# So what is this UI thing

## Wikipedia

The user interface (also known as human computer interface or man-machine interface (MMI)) is the aggregate of means by which people—the [users](#)—[interact](#) with the [system](#)—a particular [machine](#), device, [computer program](#) or other complex [tool](#). The user interface provides means of:

- [Input](#), allowing the users to manipulate a system
- [Output](#), allowing the system to indicate the effects of the users' manipulation.

## Wiktionary

Noun

[user interface](#) (plural [user interfaces](#))

1. ([countable](#)) The part of a [software application](#) that a [user](#) sees and interacts with.

# GUI for programmers

- Two main components
  - User(s)
  - System/Program/Application/Hardware
- Two pipes of information
  - Input from the User(s)
  - Output to the User(s)

# GUI for programmers

- Two main components
  - User(s)
  - System/Program/Application/Hardware
- Two pipes of information
  - Input from the User(s)
  - Output to the User(s)

Focus



# What constitutes a (G)UI

- The worlds smallest GUI

The screenshot shows an IDE window titled "Jetty Server Control" with several panels:

- Connections:**
  - Total number of connections: 3
  - Maximum number of open connections: 1
  - Maximum connection duration: 65695
  - Average connection duration: 64925
- Requests:**
  - Total number of requests: 29
  - Current number of active requests: 0
  - Max number of concurrent requests: 1
  - Max request duration: 340
- requestsDurationAve:** A line graph showing average request duration over time (13:10 to 13:28).
- FreeMemory:** A line graph showing free memory usage over time (13:10 to 13:28).
- Server information:**
  - JBoss Version: WonderLand
  - Host address: 10.0.0.7
  - Operating system: x86 Windows XP (S.1)
  - BuildID: 200305041533
  - Hostname: pink
  - Number of processors: 1
- Runtime information:**
  - Active Threads: 75
  - Free memory: 24470456
  - Max memory: 66650 Mb
- MBean Inspector (Java):**
  - Object Name: jboss.system:type=Server
  - MBean Class: org.jboss.system.server.ServerImpl





# A GUI for a programmer

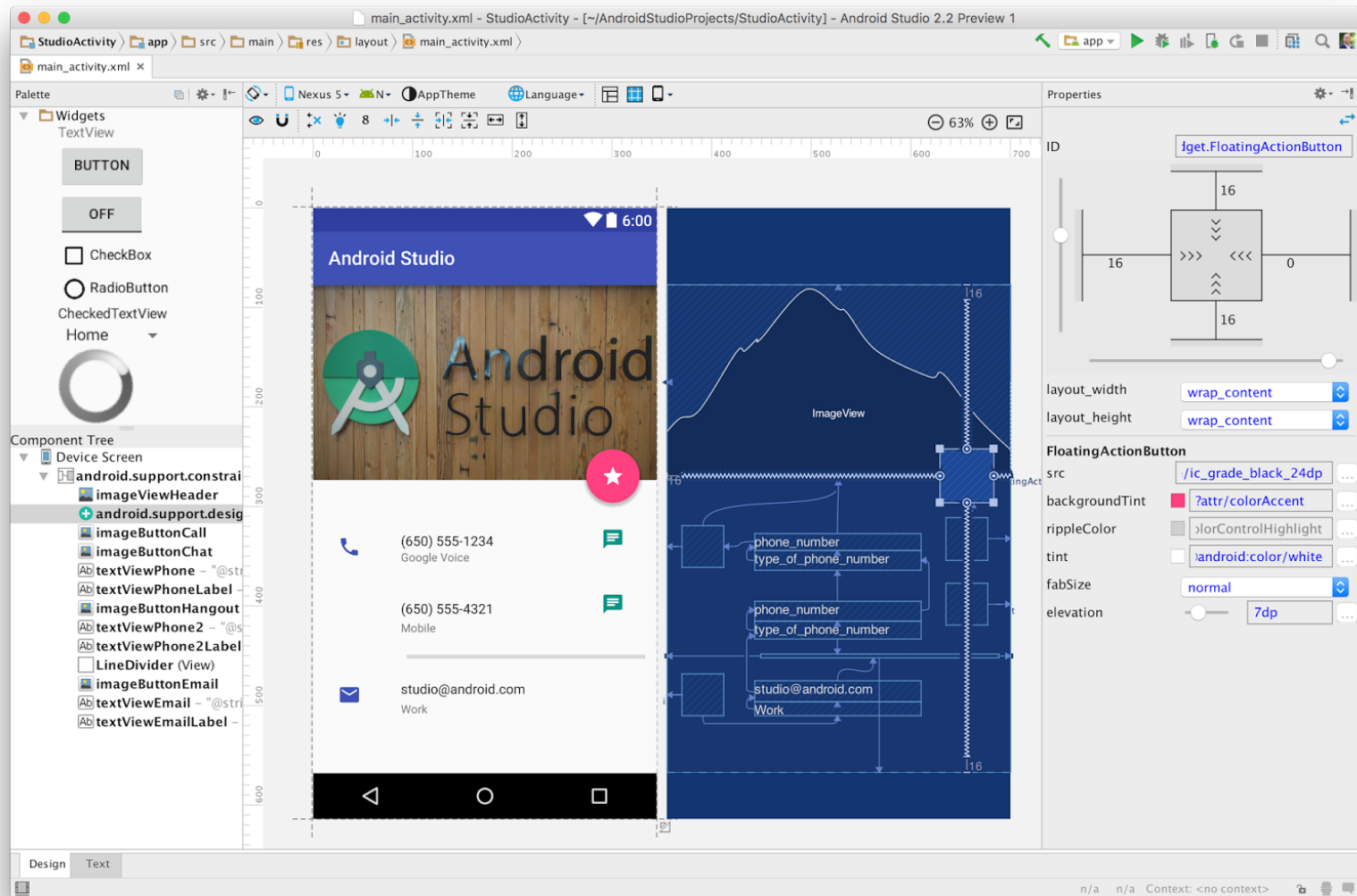
```
public void  
paint(Graphics g){  
    g.moveTo(100,100);  
    g.setColor("ffffff");  
    g.drawLineTo(200,300);  
}
```

---

Spans from low level to very  
high level

```
val intent = Intent()  
intent.type = "image/*"  
intent.action = Intent.ACTION_GET_CONTENT  
startActivityForResult(  
    Intent.createChooser(intent, "Select Picture"), PICK_IMAGE);
```

# ...and even visual



# GUI for the User

The screenshot displays the JBoss Server Control GUI, which provides a comprehensive view of the server's operational status. The interface is divided into several key sections:

- Exploring Root:** A tree view on the left showing the server's architecture, including MBean Servers, Views, JBoss 3.2.1, and various modules like EJB, JCA, and Web.
- Jetty Server Control:** A central panel showing real-time statistics:
  - Connections:** Total number of connections: 3, Maximum number of open connections: 1, Maximum connection duration: 65695, Average connection duration: 64925.
  - Requests:** Total number of requests: 29, Current number of active requests: 0, Max number of concurrent requests: 1, Max request duration: 340.
  - requestsDurationAve:** A line graph showing the average duration of requests over time, with a sharp spike at 13:18.
- JBoss Server:** A panel providing server information and runtime metrics:
  - Server information:** JBoss Version: WonderLand, BuildID: 200305041533, Host address: 10.0.0.7, Operating system: x86 Windows XP (5.1).
  - Runtime information:** Active Threads: 75, Active Threadgroups: 6, Total memory: 63963 Mb, Free memory: 24470456.
  - FreeMemory:** A line graph showing the amount of free memory over time, with a significant drop at 13:18.
- Code Editor:** A window showing Python code for monitoring and visualization, including the creation of charts and tabbed panels.
- MBean Inspector (Java):** A panel displaying detailed information about the selected MBean, including its name, class, and a table of attributes.

Name	Value	Type
BuildID	200305041533	java.lang.String
BuildDate	May 4 2003	java.lang.String
Version	3.2.1(200305041533)	java.lang.String
StartDate	Mon Nov 17 13:13:11 CET 2...	java.util.Date
VersionName	WonderLand	java.lang.String
BuildNumber	200305041533	java.lang.String

Name	Parameters	Return type	Invoke
exit	int,	void	Invoke
exit	void	void	Invoke
halt	int,	void	Invoke
traceMethodCalls	java.lang.Boolean,	void	Invoke

# And GUI for You in this course

```
protected void onDraw(Canvas canvas) {  
    super.onDraw(canvas);  
    // Draw the shadow  
    canvas.drawOval( mShadowBounds, mShadowPaint );  
  
    canvas.drawText(mData.get(mCurrentItem).mLabel, mTextX, mTextY, mTextPaint);  
}
```

Android/Java

Spans from low  
level to very high  
level

---

```
val intent = Intent()  
intent.type = "image/*"  
intent.action = Intent.ACTION_GET_CONTENT  
startActivityForResult(  
    Intent.createChooser(intent, "Select Picture"), PICK_IMAGE);
```

Android/Kotlin

# How do we get from

What user wants and needs (user/system requirements)



The screenshot displays the JBoss IDE interface. The main window shows the 'Jetty Server Control' window, which provides real-time performance metrics for the Jetty server. The 'Connections' section shows 3 total connections, with a maximum of 1 open connection. The 'Requests' section shows 29 total requests, with 0 active requests. A line graph titled 'requestsDurationAve' shows the average request duration over time, with a peak around 13:18. The 'Free Memory' graph shows memory usage over time, with a peak around 13:18. The 'JBoss Server' window shows server information, including the version (WonderLand 200305041533), host address (10.0.0.7), and operating system (x86 Windows XP (5.1)). The 'Runtime information' section shows 75 active threads, 6 active thread groups, 63963 Mb total memory, and 24470456 Mb free memory. The 'MBean Inspector (Java)' window shows the MBean information for the server, including the object name, MBean class, and attributes.

Name	Value	Type
buildID	200305041533	java.lang.String
buildDate	May 4 2003	java.lang.String
Version	3.2.1(200305041533)	java.lang.String
StartDate	Mon Nov 17 13:13:11 CET 2...	java.util.Date
VersionName	WonderLand	java.lang.String
buildNumber	200305041533	java.lang.String

Name	Parameters	Return type	Invoke
exit	int,	void	Invoke
exit	void	void	Invoke
halt	int,	void	Invoke
traceMethodCalls	java.lang.Boolean,	void	Invoke

# How do we get from

```
protected void onDraw(Canvas canvas) {
```

```
    super.onDraw(canvas);
```

```
    // Draw the shadow
```

```
    canvas.drawOval( mShadowBounds, mShadowPaint );
```

```
    canvas.drawText(mData.get(mCurrentItem).mLabel, mTextX, mTextY, mTextPaint);
```

The screenshot displays an IDE with several windows. The 'Jetty Server Control' window shows a 'requestsDurationAve' graph with a red line showing a sharp spike at 13:18. The 'JBoss Server' window shows runtime information for a server named 'pink'.

**Jetty Server Control - Connections:**

Total number of connections:	3	Maximum number of open connections:	1
Maximum connection duration:	65695	Average connection duration:	64925

**Jetty Server Control - Requests:**

Total number of requests:	29	Current number of active requests:	0
Max number of concurrent requests:	1	Max request duration:	340

**JBoss Server - Server information:**

JBoss Version:	WonderLand	BuildID:	200305041533
Host address:	10.0.0.7	Hostname:	pink
Operating system:	x86 Windows XP (5.1)	Number of processors:	1

**JBoss Server - Runtime information:**

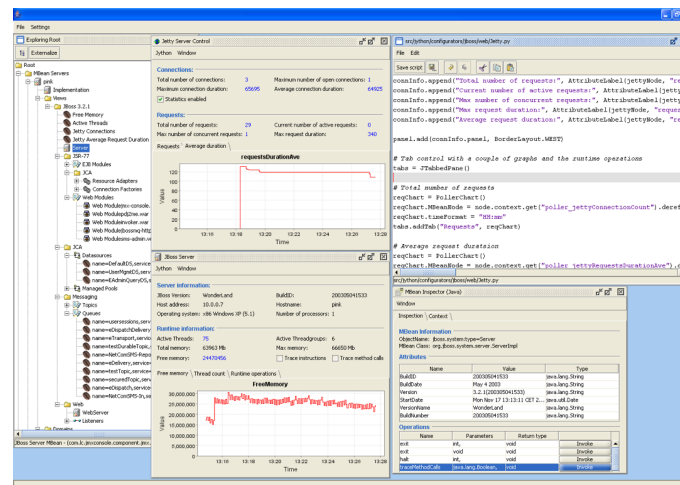
Active Threads:	75	Active Threadgroups:	6
Total memory:	63953 Mb	Max memory:	66650 Mb
Free memory:	24470456		

**JBoss Server - FreeMemory graph:**

Time	Free Memory (Mb)
13:16	~25000
13:18	~25000
13:20	~25000
13:22	~25000
13:24	~25000
13:26	~25000
13:28	~25000

# Solution

SDKs  
for (gui-)code:  
How to build it



Interaction Patterns  
for user requirements:  
What to build

# Now how do you get there

- Welcome to this course
- More interaction programming
- This is a programming course (G2 level)
  - Focus on writing code
  - Focus on writing for others
- You will learn from working not listening



**People generally remember...  
(learning activities)**

10% of what they read

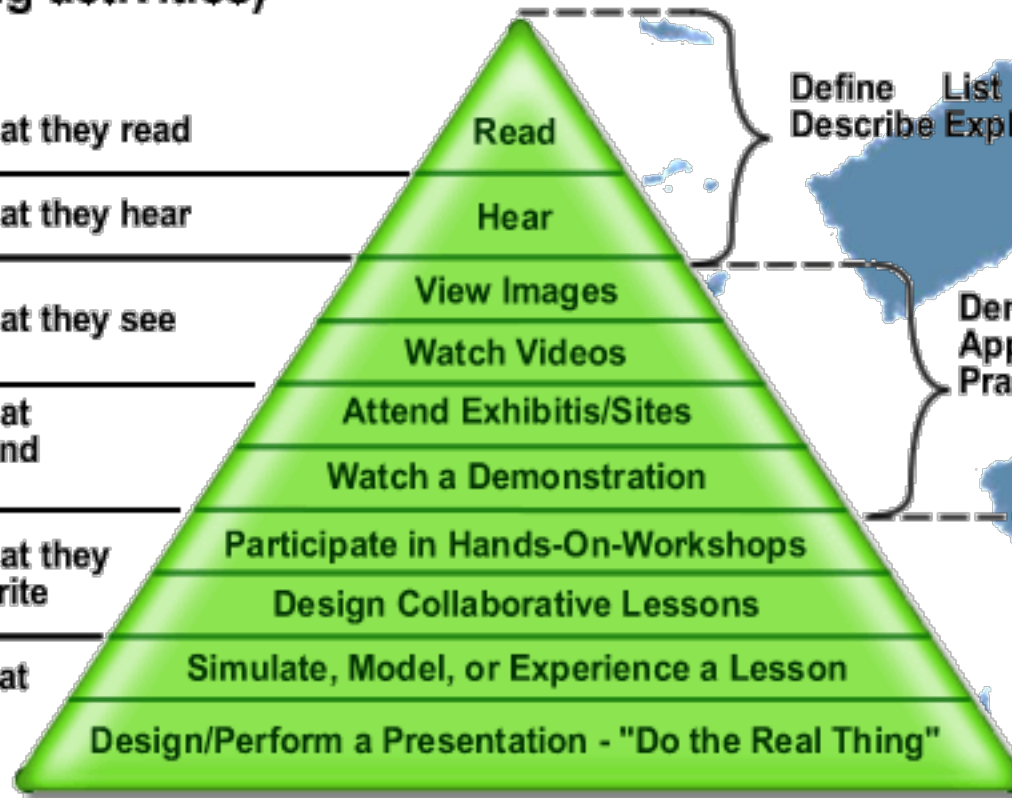
20% of what they hear

30% of what they see

50% of what they see and hear

70% of what they say and write

90% of what they do.



**People are able to...  
(learning outcomes)**

**Define** **List**  
**Describe** **Explain**

**Demonstrate**  
**Apply**  
**Practice**

**Analyze**  
**Define**  
**Create**  
**Evaluate**

# Course outline - Your work

- 3 Labs
  - Component placement (Four frameworks)
  - User interaction and component coupling
  - Communication with the rest of the world
- Mini project
  - Your own mini Library
    - Implement 2 (at least) high-level interaction patterns in a Library.
    - You may need to develop supporting components

# Course outline - our work

- Introduction to Course (right now)
- Two ui lectures
  - Closely related to the labs
- One lecture on adjacent topics
  - Testing
  - Framework/Library design

# Labs

- Cover basic GUI-programming
  - Component placement
  - Component Interaction
  - Component-System/network interaction

# Mini library

- A Library with of interaction patters (two of them)
- A example app using your Library

# Grading for Course

- 3 Labs and mini project
- 4
  - grade 3 + Testing or Getting started tutorial
- 5
  - grade 3 + Testing and Getting started tutorial

# Examination

- To pass the course you need to
  - Pass the labs (assistants)
  - Pass the mini project (assistants)
  - Pass the oral examination (Anders)

# Feedback from last year

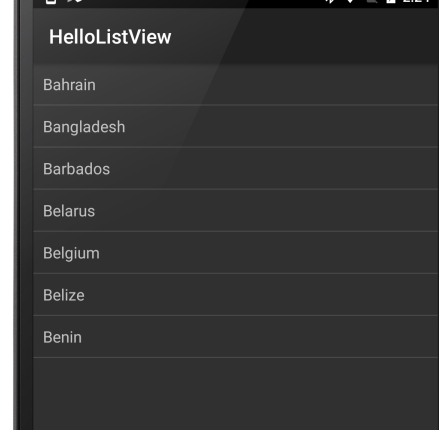
- Fun
  - But time consuming
- Lab 3 was a bit to open
  - What is “trending” on GitHub
- Project was/is easier than the labs



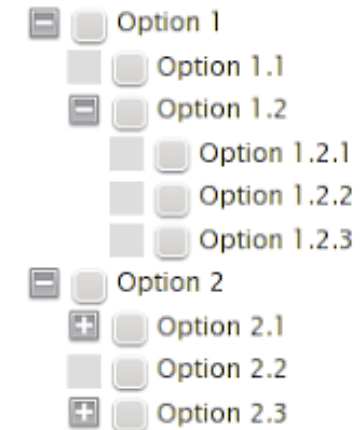
# GUI SDKs

- GUI SDK contains
  - Runtime -a mapping between Abstract UI and native OS/SDK components/primitives
  - Widgets - Buttons, Menus , Text Field etc
  - Widget Messages - interoperability among widgets
  - Messaging structures for in-/output - Messages
  - Rule sets/widget constraints - for controlling layout

# UI -Widgets

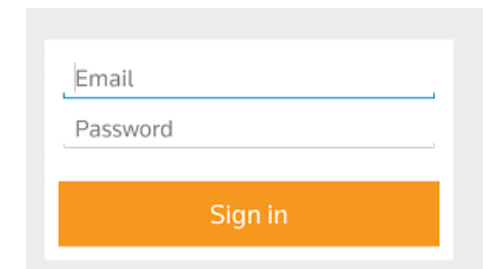


## Tree Widget Test



Select up to 3 options.

- Components
  - A set of visual reusable objects
  - Buttons, Tree, Table, Text-field
    - Android : View, Button , List, EditText, TextView
- Containers
  - Placeholders for Components and Containers
    - Usually allows for nested Containers
    - Windows, Panels, Menus, Toolbars
    - Android: ViewGroup, Layouts



# Widget Messages

- Messages from widgets to your code
  - Information about a change in state
  - Messages of user generated actions
    - Mouse movement, keyboard action
    - Touch , Sensors
- Widgets has a set of Messages (zero or more) you choose which you care about

# Widget Messages

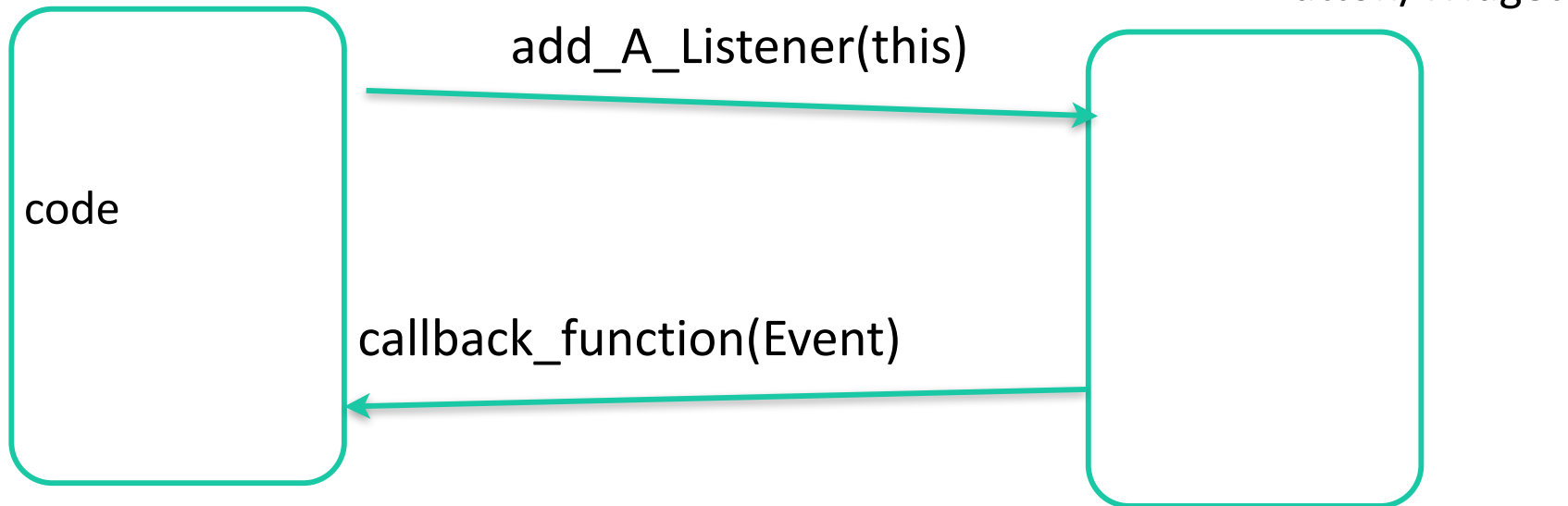
- Messages passed through callback functions
- You registers yourself with component using listeners
  - android : `setOnClickListener`
- Receive notification
  - `onClick(View v)`

# Listeners



Your code

Button/Widget



# Messages for in- and output

- GUI runs in a separate thread
  - Keep this thread clean only use for GUI related issues
- SDK provides mechanism for delivering info that can be received and processed in that thread

# Threads

- Problem: UI not Thread safe
  - GUI code only in the GUI Event thread
- Solution: Place code on cue
  - Causes UI thread to run a given runnable when it can on the UI Event thread
- Use post any time you want to modify a UI object outside of a listener method

# Android : {View}.post

```
mImageView.post(new Runnable() {  
    public void run() {  
        mImageView.setImageBitmap(bitmap);  
    }  
});
```

*// OR AsyncTask*

```
SwingUtilities.invokeLater(  
    new Runnable(){  
        public void run(){  
            outputArea.append(messageToDisplay);  
        }  
    }  
);
```

---



# Rule sets and widget constraints

- Controlling widgets
  - placement, size
  - relation to other widgets
  - reaction to external changes
    - Window size change
    - Device orientation, size, resolution

# Placing components

- Layouts in Android
  - Constraint, Motion Linear- Vertical/Horizontal (Grid , Table)
- Layouts in React Native
  - Flexbox
- Layouts in Flutter
  - Layout widgets
    - Single-Child widget(s)
    - Multi-Child widget(s)

# Conclusion

- What I hope you take with you
  - This is a programming course you learn by doing
  - Tools aids developers - so learn them
  - Some brief hints on how to develop today's GUI application.

- Questions