

http://www.kom.e-technik.tu-darmstadt.de  
http://www.tk.informatik.tu-darmstadt.de  
© R. Steinmetz, M. Mühlhäuser

# Multimedia-Systems: Programming

Prof. Dr.-Ing. **Ralf Steinmetz**

Prof. Dr. **Max Mühlhäuser**

**MM:** TU Darmstadt - Darmstadt University of Technology,  
Dept. of of Computer Science

TK - Telecooperation, Tel.+49 6151 16-3709,

Alexanderstr. 6, D-64283 Darmstadt, Germany, [max@informatik.tu-darmstadt.de](mailto:max@informatik.tu-darmstadt.de) Fax. +49 6151 16-3052

**RS:** TU Darmstadt - Darmstadt University of Technology,

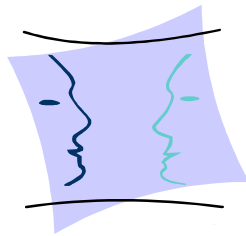
Dept. of Electrical Engineering and Information Technology, Dept. of Computer Science

KOM - Industrial Process and System Communications, Tel.+49 6151 166151,

Merckstr. 25, D-64283 Darmstadt, Germany, [Ralf.Steinmetz@KOM.tu-darmstadt.de](mailto:Ralf.Steinmetz@KOM.tu-darmstadt.de) Fax. +49 6151 166152

GMD -German National Research Center for Information Technology

httc - Hessian Telemedia Technology Competence-Center e.V.



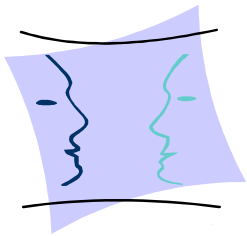
Contents

Contents



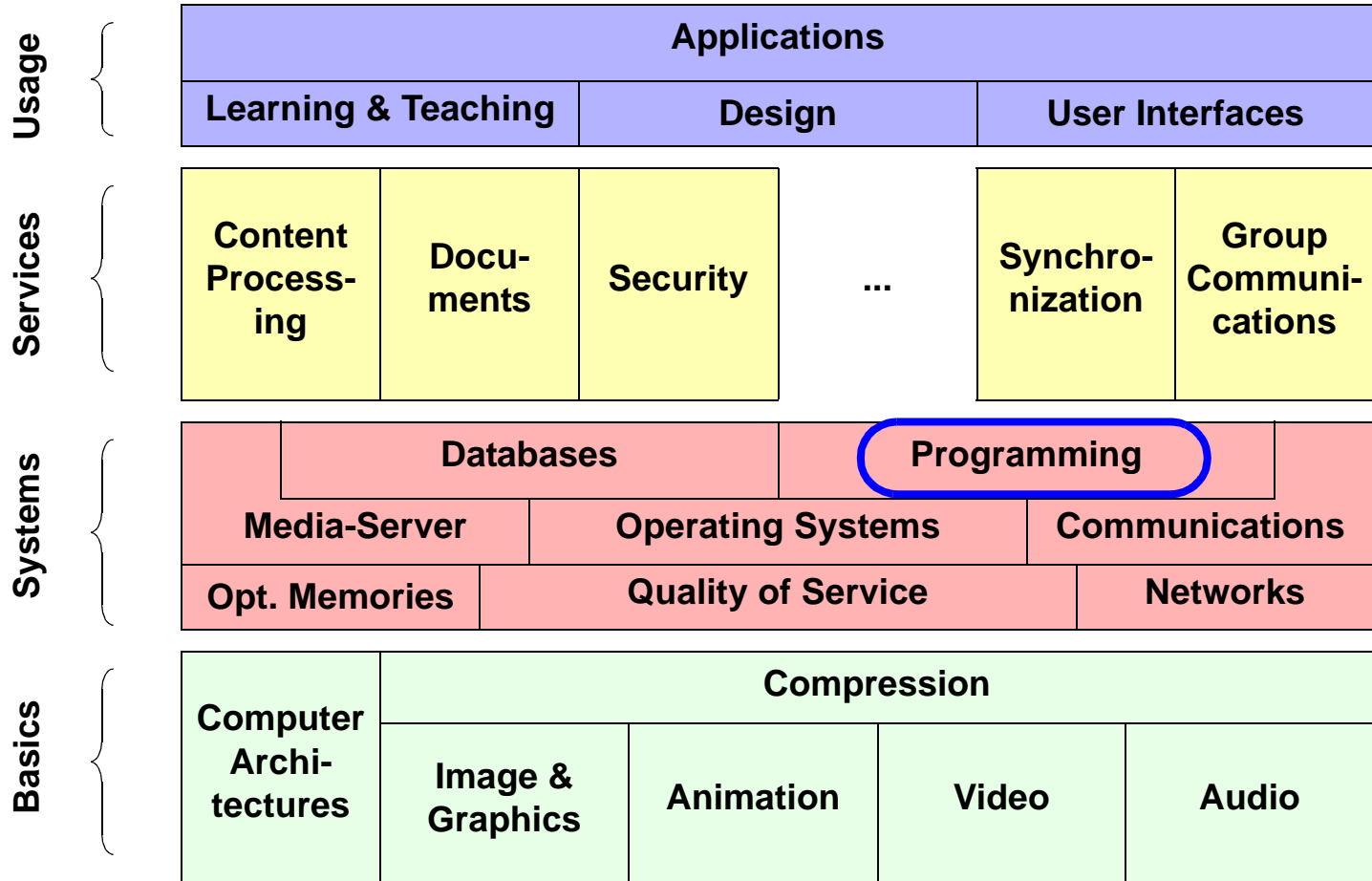
# Scope

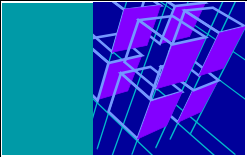
<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
 © R. Steinmetz, M. Mülhäußer



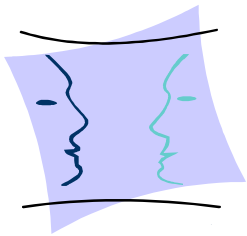
Contents

Contents





<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



Contents

Contents

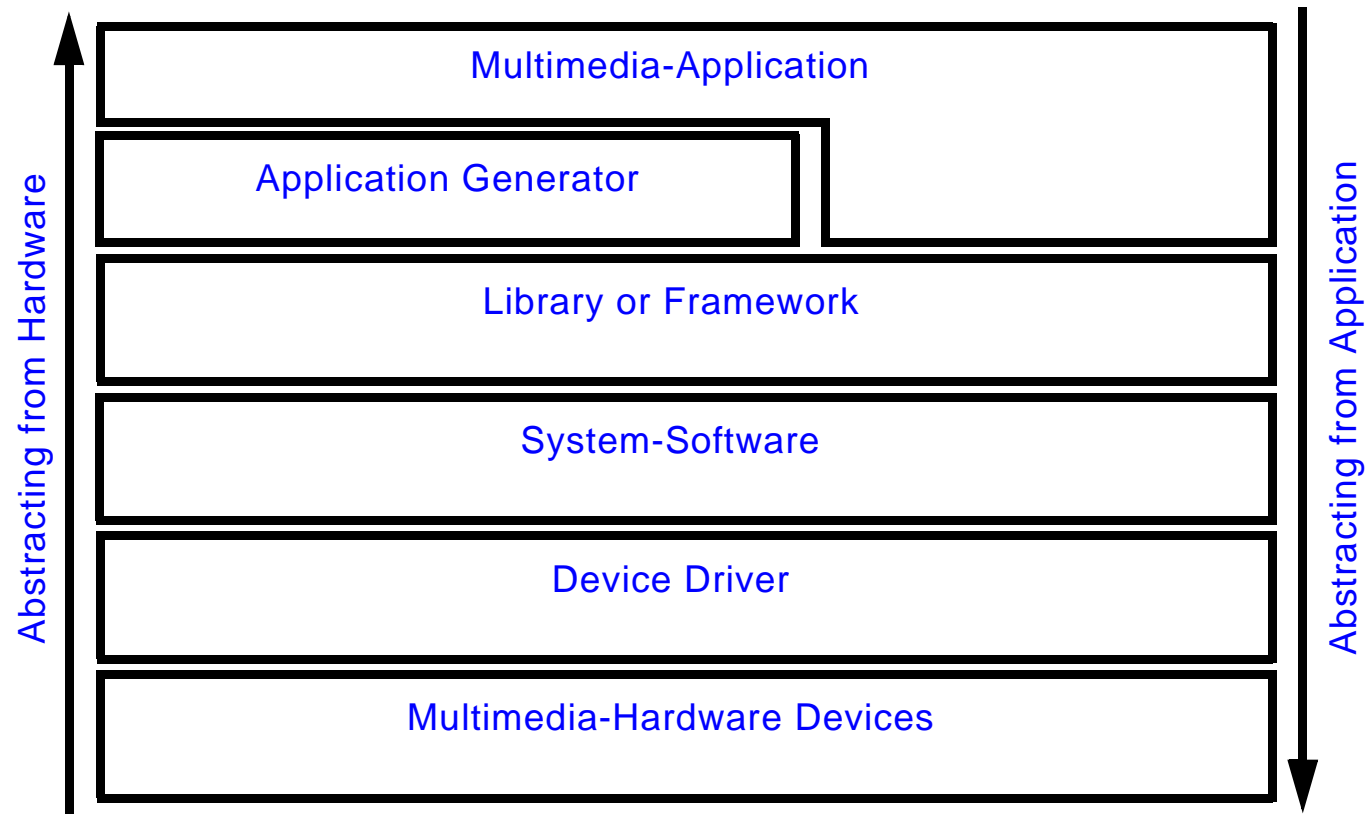


# Contents

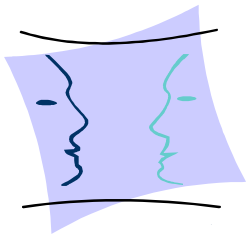
---

- 1. Programming - Abstracting from MM-Hardware**
  - Device Drivers**
  - Devices**
  - System Software - MCI**
  - System Software - DirectX**
  - Libraries - Open GL**
  - Java Media Framework (JMF)**
  - Application Generators and Visual Programming**
- 2. Programming Languages**
  - Requirements**
  - Object Orientation - Basics**
  - Java**
- 3. Distributed Computing**
  - Initial Situation and Requirements**
  - CORBA**
  - CORBA and Java**

# 1. Programming - Abstracting from MM-Hardware



<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



Contents

Contents



# Device Drivers

## At hardware level

- **sequence of (very) specific instructions**
- **IO / DMA addresses determined by individual hardware**

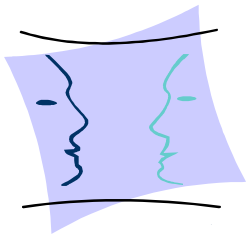
```
static unsigned long read_timer(void)
{
```

```
    /* example taken from Linux kernel code - i386 architecture */
    unsigned long t, flags;
    int i;
    __save_flags(flags);
    __cli();
    t = jiffies * 11932;
    outb_p(0, 0x43);
    i = inb_p(0x40);
    i |= inb(0x40) << 8;
    __restore_flags(flags);
    return (t - i);
```

```
}
```

## Strong hardware dependency may cause problems with:

- **portability**
- **reusability**
- **coding efficiency**



# Devices

## Try to:

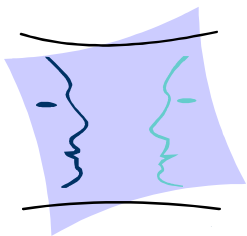
- **identify classes of similar devices**
- **treat them via a generic interface as part of the operating system**
  - block devices (e.g. /dev/hda)
  - character devices (e.g. /dev/sound)

## Abstraction - File

- **system calls:**
  - open, close
  - read, write
  - (seek)
  - fcntl, ioctl - for all further functionality
- **example:**
  - `micro_in = open (“/dev/audio”, O_RDONLY);`
  - `samples = read (micro_in, buffer, 1024);`
  - ...

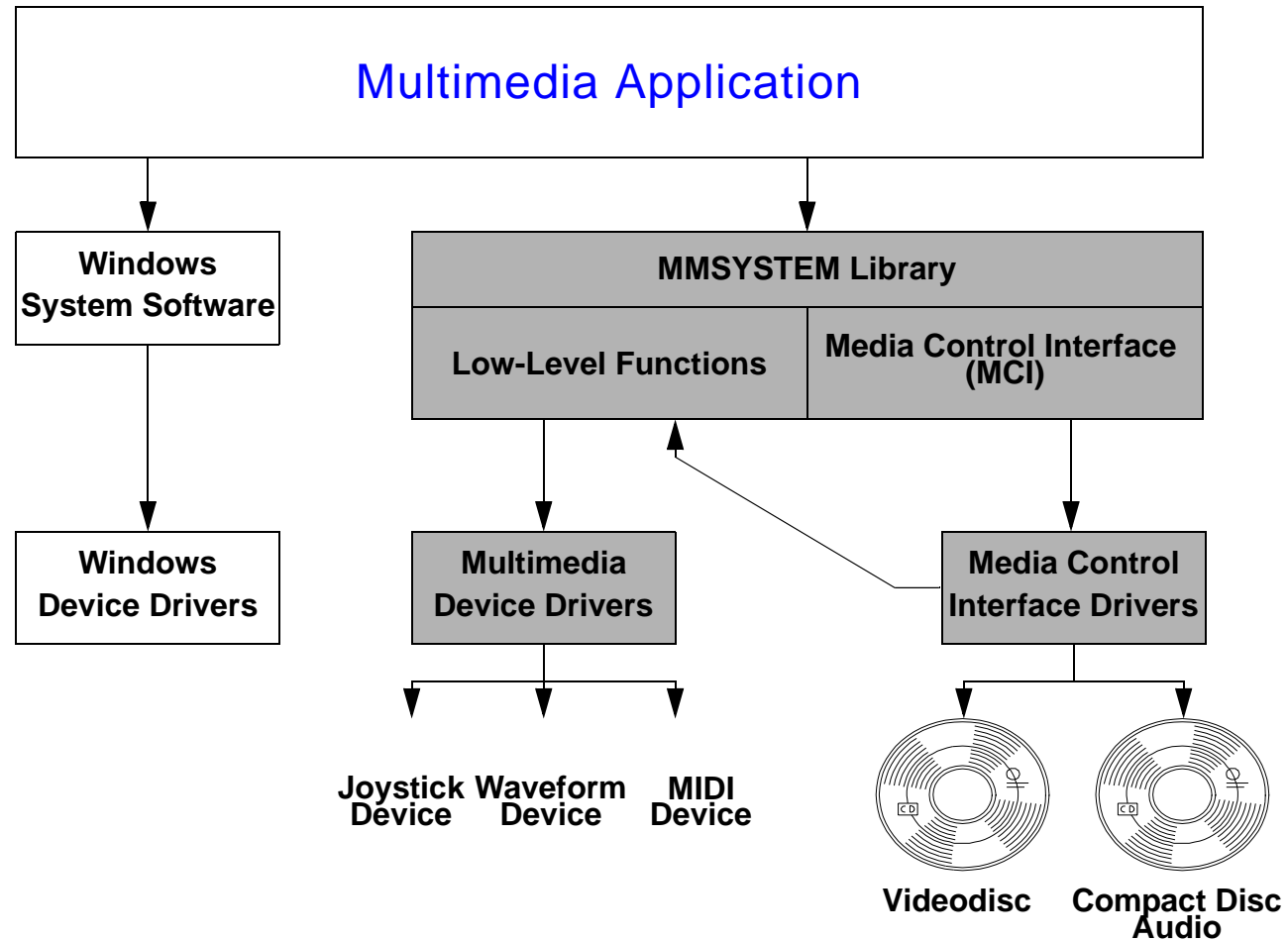
## other mechanisms:

- **memory mapping**
- **I/O or DMA mapping**



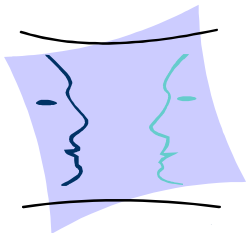
# System Software - MCI

## OS/2 and Windows *Media Control Interface (MCI)*



- **MMSYSTEM lib. for extensibility and device independence**

<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



Contents

Contents



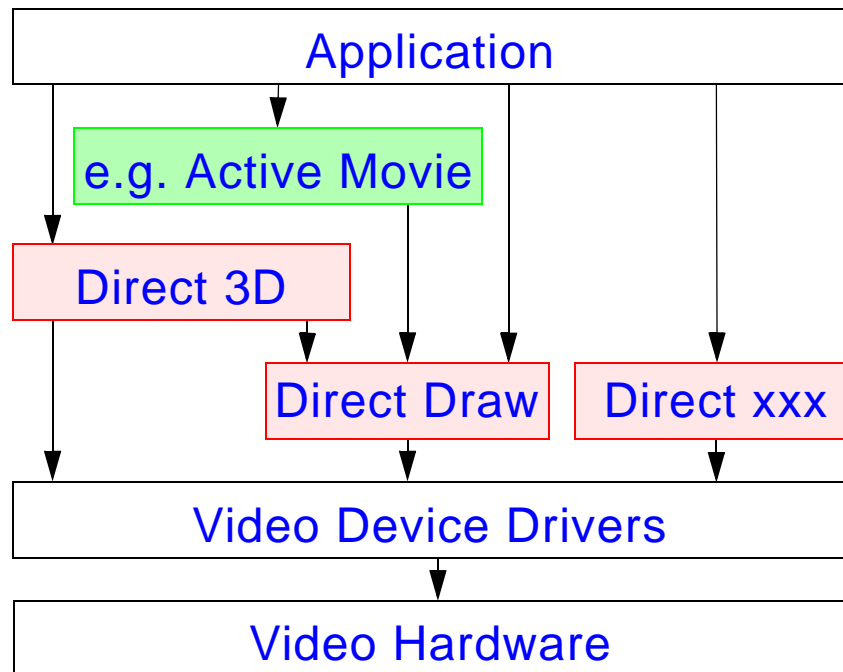
# System Software - DirectX

Low-level APIs and libraries for high-performance applications

- Especially games - formerly known as the "Game SDK"

Direct access to hardware services

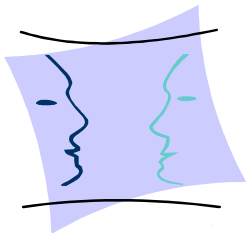
- e.g. audio & video cards, hardware accelerators
- "DirectX" = "direct access"
- strong relationship / interaction with ActiveX / DCOM



= component of ActiveX



= component of DirectX



# DirectX (cont.)

---

## Components:

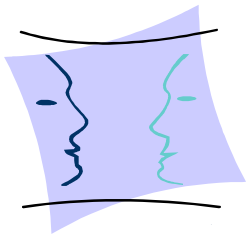
- **DirectDraw** - 2 dimensional graphics capabilities
- **Direct3D** - extensively functional 3D graphics programming API
- **DirectSound** - (3D) sound, mixing and playback of multiple streams
- **DirectPlay** - for network multiplayer game development
- **DirectInput** - input from various peripherals, e.g. joysticks, data gloves

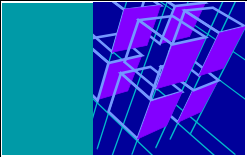
## Implementation Strategy:

- **Hardware Abstraction Layer (HAL)**
- **Hardware Emulation Layer (HEL)**
- **Media Layer (for aggregated “high level” functionality)**
  - animations
  - media streaming
  - synchronisation

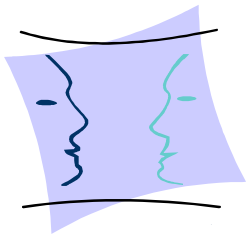
## Other System Software Examples

- Apple QuickDraw
- QuickTime





<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



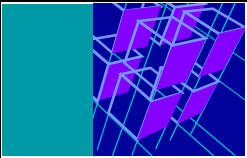
Contents

Contents

# Libraries - Open GL

- **2D and 3D graphics API developed by Silicon Graphics**
- **basic idea: “write applications once, deploy across many platforms”**
  - PCs
  - Workstations
  - Super Computers
- **Benefits:**
  - Industry Standard
  - Stable
  - Reliable and Portable
  - Evolving
  - Scalable (HW features like Z-buffering, Zoom, Rectangle handling ...)
  - Well documented and easy to use
- **integrated with:**
  - Windows NT
  - Windows 95
  - UNIX X Window System

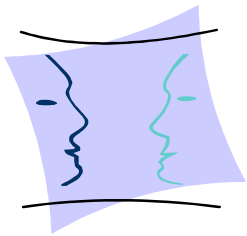




# Frameworks

---

<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



Contents

Contents

## Additional level of abstraction:

- **Combination of different media**
- **New data types**
  - e.g. **mixed.video := orig.video + subtitle**
  - granularity problematic (pixel, frame, image group, sequence, video, ...)
- **Reuse of Modules:**
  - Synchronization
  - Scaling
  - (De)compression
  - File formats
  - etc.

## Application frame given

- **Generic problem solution**

## Creation of specialized application components

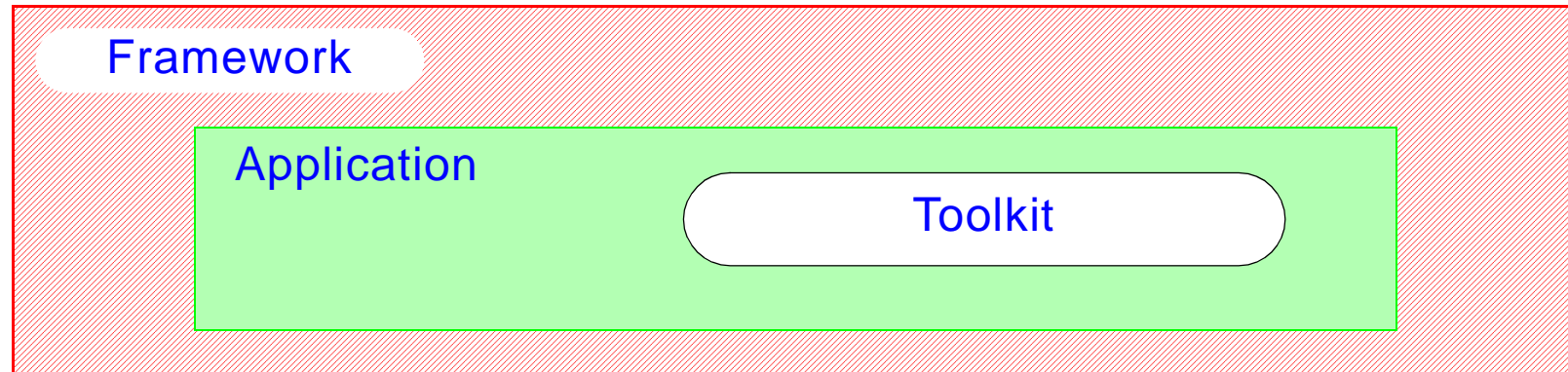
- **Media containers, media processors**



# Frameworks (cont.)

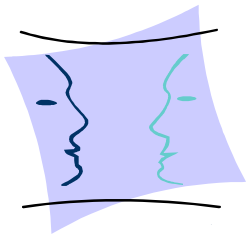
## Framework controls execution

- E.g. `myApplicationComponent.init(); myApplication.start();`



## Examples:

- **Java Media Framework (JMF)**
- **further information on Applets as another example in lecture on documents**



# Java Media Framework (JMF)

support for platform-neutral synchronized playback of multimedia data

- **local**
- **via network**
  - pull protocols - e.g. HTTP
  - push protocols - e.g. RTP

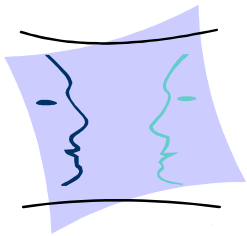
easy integration into platforms native environment and core packages

supported media formats:

- **MPEG-1**
- **MPEG-2**
- **QuickTime**
- **AVI**
- **WAV**
- **AU**
- **MIDI**

extensible:

- **new Data Sources (e.g. for FTP or Video on Demand (VOD) protocols)**
- **new Players**



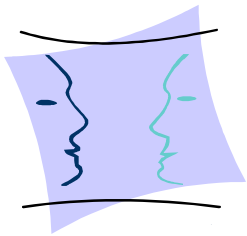
# Java Media Framework (cont.)

---

## Basic features:

- **state model**
- **communication and control via base API (extend abstract base class)**
- **event model**
- **may be embedded in Beans (Component) model**
- **set of controls is player defined**
- **synchronisation support (one player may supervise another)**

<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



Contents

Contents



# Application Generators and Visual Programming

## Goal:

- **Support for Authoring and Generation of Multimedia Applications on an intuitive basis**

## Categories (maybe mixed):

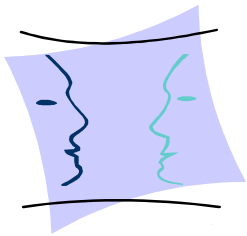
- **script-based**
- **icon-based**
- **timeline-based**

Approach may be used for visual programming

## Example: Macromedia Director

- **describe parts of a presentation and arrange them on a time axis**
- **Scripting Language LINGO**
- **may**
  - generate standalone application
  - export several formats
    - HTML
    - Java
    - Shockwave

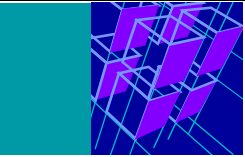
<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



Contents

Contents





## 2. Programming Languages

### Requirements

#### Generally should support:

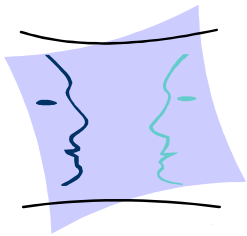
- **Code Reuse**
- **Extensability**
- **Maintainability**
- **Robustness**

#### Especially for Multimedia Applications:

- **Efficient Handling of Large Amount of Data / Efficiency**
- **Realtime Support**
- **Synchronisation Support**

#### Conclusion:

- **we can! use modern Object Oriented Approaches**
- **inherit**
- **extend and**
- **reuse**
  - rather than develop from scratch or copy, paste and fix



# Object Orientation - Basics

---

## Termini:

- **Object / Class**
- **Instance**
- **Interface**

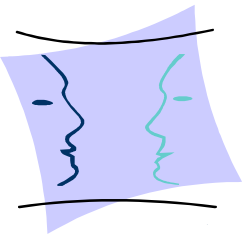
## Basic OO features:

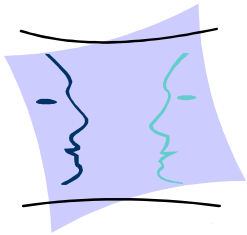
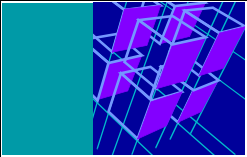
- **Inheritance**
- **Polymorphism**
- **Dynamic (late) Binding**
- **Communication using Messages**

support differs in various languages (e.g. C++)

## Those together fulfill the real OO goals:

- **Abstraction**
- **Encapsulation**
- **Comprehensiveness**
- **Reusability**





# Java

**High-level programming language (Sun, initially for settop boxes - project OAK)**

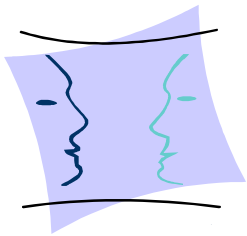
- **Object-oriented**
  - C++-like (easy to learn), but also strong similarities to smalltalk
- **Type safety**
  - promoted by compiler and runtime environment (exceptions)
  - but not completely forced
- **Inherent language support for Multi-Threading**
- **No pointer mechanisms**
- **Garbage collection instead of user driven memory allocation**
- **No system dependent features**
  - data types have fixed sizes
- **Security mechanisms**
  - restricted language features
  - various checks at compile as well as on run-time
- **Platform-independence by means of bytecode**
  - allows for Applet-Code loadable via network
- **comes with a powerful class library**

# Java - Functionality

## Rich set of Predefined Classes

- **java.awt**
  - *abstract window toolkit*
  - hides basic mechanisms -> makes programs portable
  - functions e.g.:
    - access to underlying window system (native look and feel)
    - reaction upon events like keystrokes or mouse clicks
    - drawing of lines / figures / buttons / scrollbars
    - image manipulation
- **java.applet**
  - basic framework for applets
  - e.g. get and display WWW pages, play audio clips
  - more infos in lecture on Documents / WWW
- **Networking (Server and Multicast support)**
- **Display of various media: text, graphics, animation, audio**
  - ====> Java Media Frame (JMF)
  - ====> Java Telephony API (JTAPI)
  - ...

<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



Contents

Contents



# Java - Resume

---

## What about performance?

- Runtime Optimization (Just in Time Compiling JIT)
- Compilation to machine specific target code
- Mechanisms to integrate native Code (Java Native Interface - JNI)
- Hardware implementations (PicoJava Processor)

## also important driving innovation in development

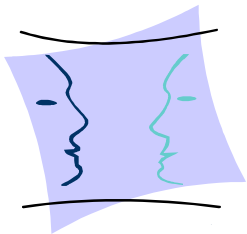
- **from retrieval-only WWW**
- **towards an “Object Web”**

## Java is:

- **not just a another programming language**
- **but a large step towards the vision**

**“The network is the computer !”**

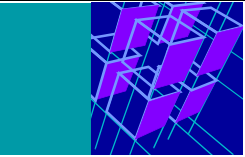
<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



Contents

Contents

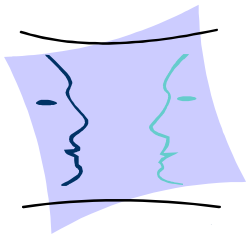




# 3. Distributed Computing

## Initial Situation and Requirements

<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



Contents

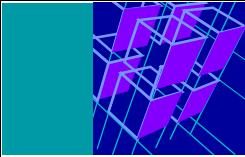
Contents



- **Heterogeneity**
  - Hardware
  - Operating Systems
  - Network
  - Programming Language
- **General strong need for:**
  - Access Transparency
  - Location Transparency
  - Migration Transparency
  - Replication Transparency
  - Concurrency Transparency
  - Failure Transparency
  - Performance Transparency
  - Scalability Transparency

### Note:

- **Multimedia Requirements (Performance, QoS) may be in conflict with some of those**



# Middleware

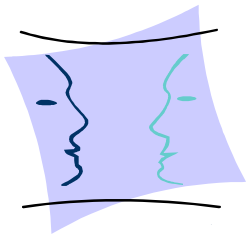
---

## Classification:

- **Message Oriented Middleware**
  - IBM MQ, DEC Message Queue, NCR Top End ...
- **Transaction Processing Middleware**
  - IBM CICS, BEA Tuxedo
- **Object Oriented Middleware**
  - Open Management Group (OMG) CORBA
  - Microsoft DCOM
  - Java / RMI

**Different approaches also converging**

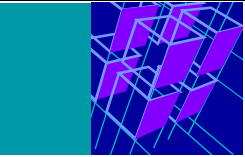
<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer



Contents

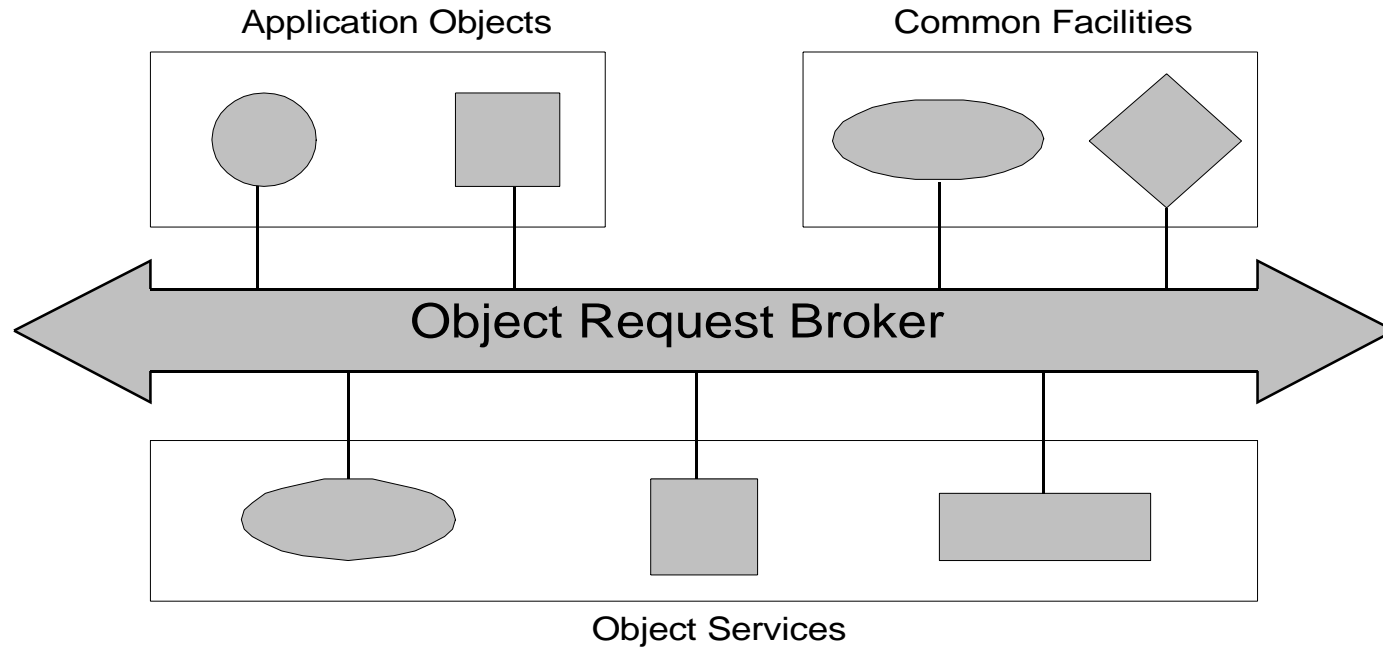
Contents



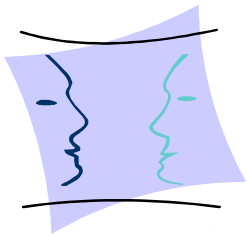


# CORBA

- extends Remote Procedure Call (RPC) model by means of Object Oriented Middleware Concept



<http://www.kom.e-technik.tu-darmstadt.de>  
<http://www.tk.informatik.tu-darmstadt.de>  
© R. Steinmetz, M. Mülhäußer

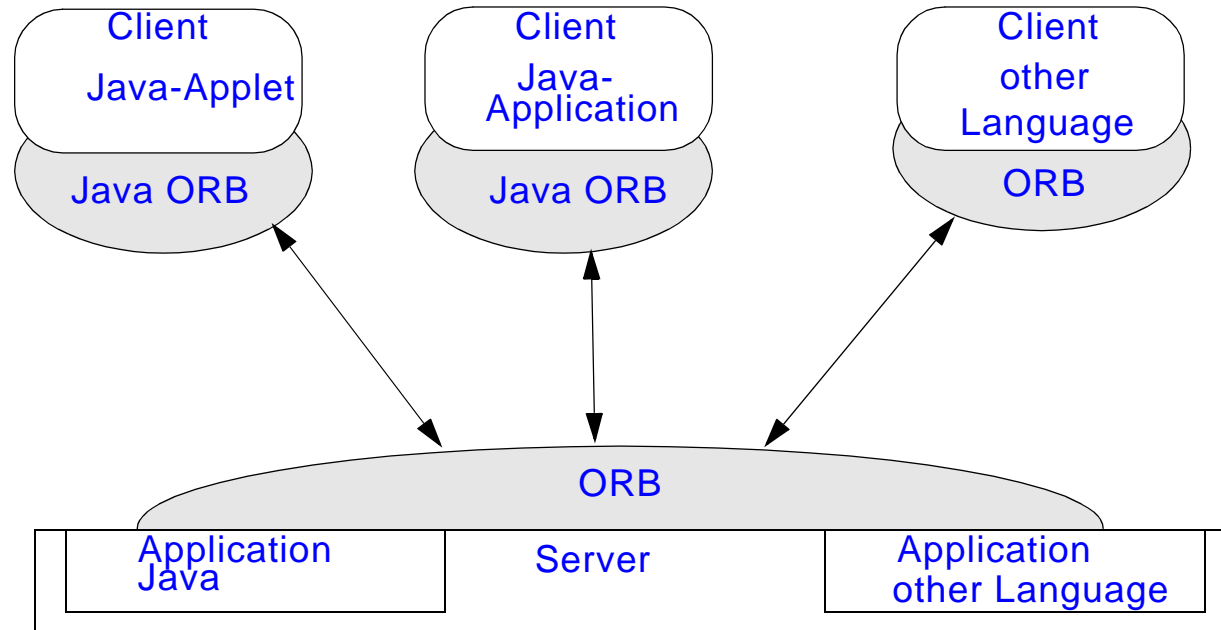


Contents

Contents



# CORBA and Java



## Implementation

- **100 percent pure Java ORBs loaded from filesystem or via network**
- **starting with JDK1.2, CORBA is a integral part**

## For Java-only Distributed Systems

- **consider using Remote Method Invocation (RMI)**

