Abstractions for Programming

By

Prof. S. Shakya

Overview

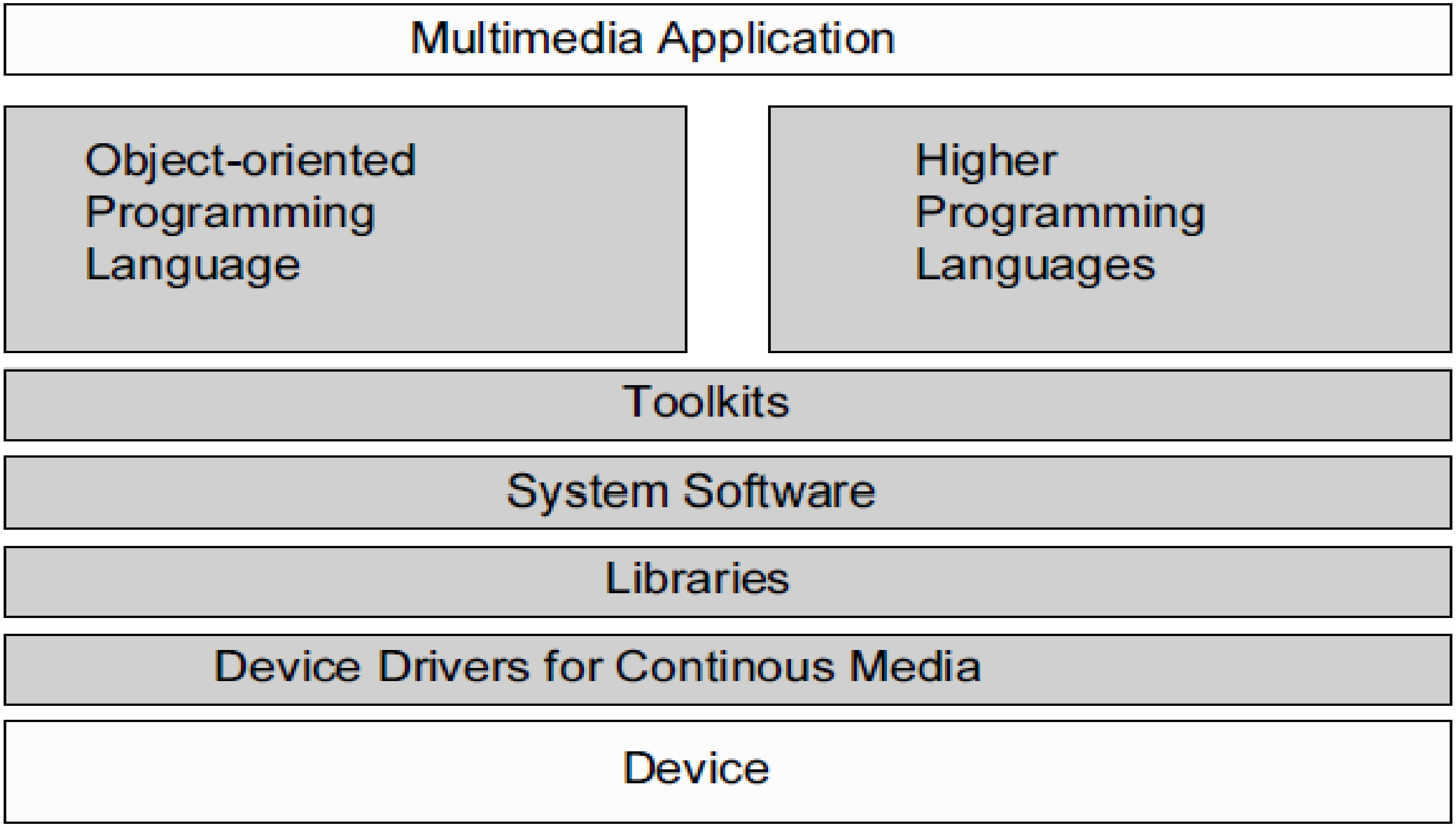
The state of the art of programming

- Most of the current commercially available multimedia applications are implemented in procedure-oriented programming languages
- Application code is still highly dependent on hardware
- Change of multimedia devices still often requires re-implementation
- Common operating system extensions try to attack these problems
- Different programming possibilities for accessing and representing multimedia data

Overview of different abstraction levels

- Libraries
- System software
- Toolkits
- Higher Programming languages
- Object-oriented approaches

Abstraction for Programming


Multimedia Application
Object-oriented Programming Language
Higher Programming Languages
Toolkits
System Software
Libraries
Device Drivers for Continous Media
Device

Abstraction Levels of the Programming of Multimedia Systems

Abstractions from Multimedia Hardware

Strong hardware dependency may cause problems with:

- Portability
- Reusability
- Coding efficiency

Abstraction Levels

- Common operating system extensions try to solve this problem
- Different programming possibilities for accessing and representing multimedia data

Libraries

Processing of continuous media based on functions embedded in libraries

- Libraries differ in their degree of abstraction

Libraries - OpenGL

2D and 3D graphics API developed by Silicon Graphics

- Basic idea: “write applications once, deploy across many platforms”:

✓ PCs

✓ Workstations

✓ Super Computers

- Benefits:

✓ Stable

✓ Reliable and Portable

✓ Evolving

✓ Scalable (Features like Zoom, Rectangle handling ...)

✓ Well documented and easy to use

- Integrated with:

✓ Windows 95/NT/2000/XP

✓ UNIX X Window System

System Software

❑ Device access becomes part of the operating system:

❑ Data as *time capsules (file extensions)*

- Each Logical Data Unit (LDU) carries in its time capsule its data type, actual
- value and valid life span
- Useful concept for video, where each frame has a valid life span of 40ms (rate
- of read access during a normal presentation)
- Presentation rate is changed for VCR (Video Cassette Recorder) functions like
- fast forward, slow forward or fast rewind by

  - Changing the presentation life span of a LDU

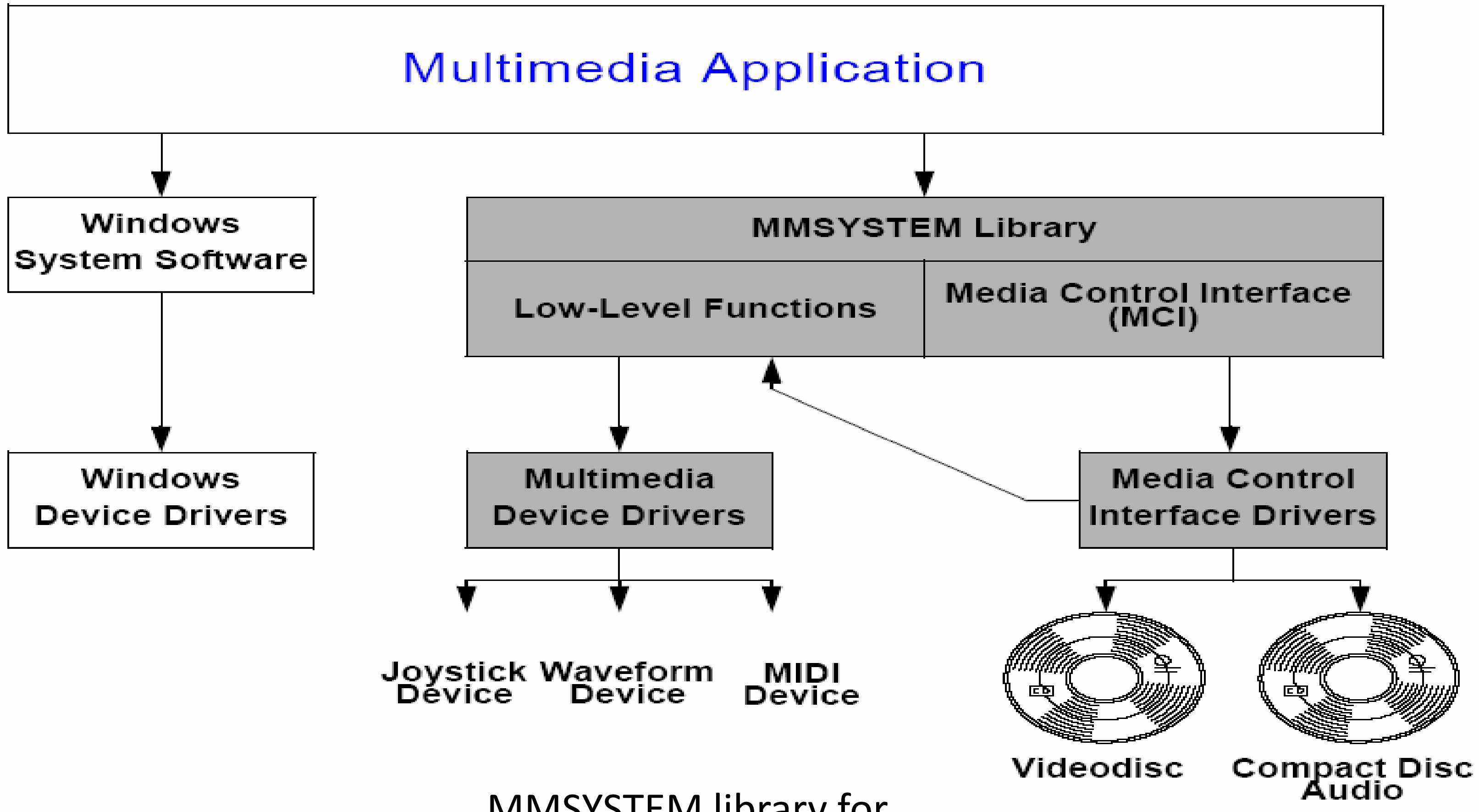
  -Skipping of LDUs or repetition of LDUs

❑ Data as *streams*

  - a stream denotes the continuous flow of audio and video data between a source and a sink

  - Prior to the flow the stream is established equivalent to the setup of a connection in a networked environment

System Software: Windows *Media Control Interface (MCI):*

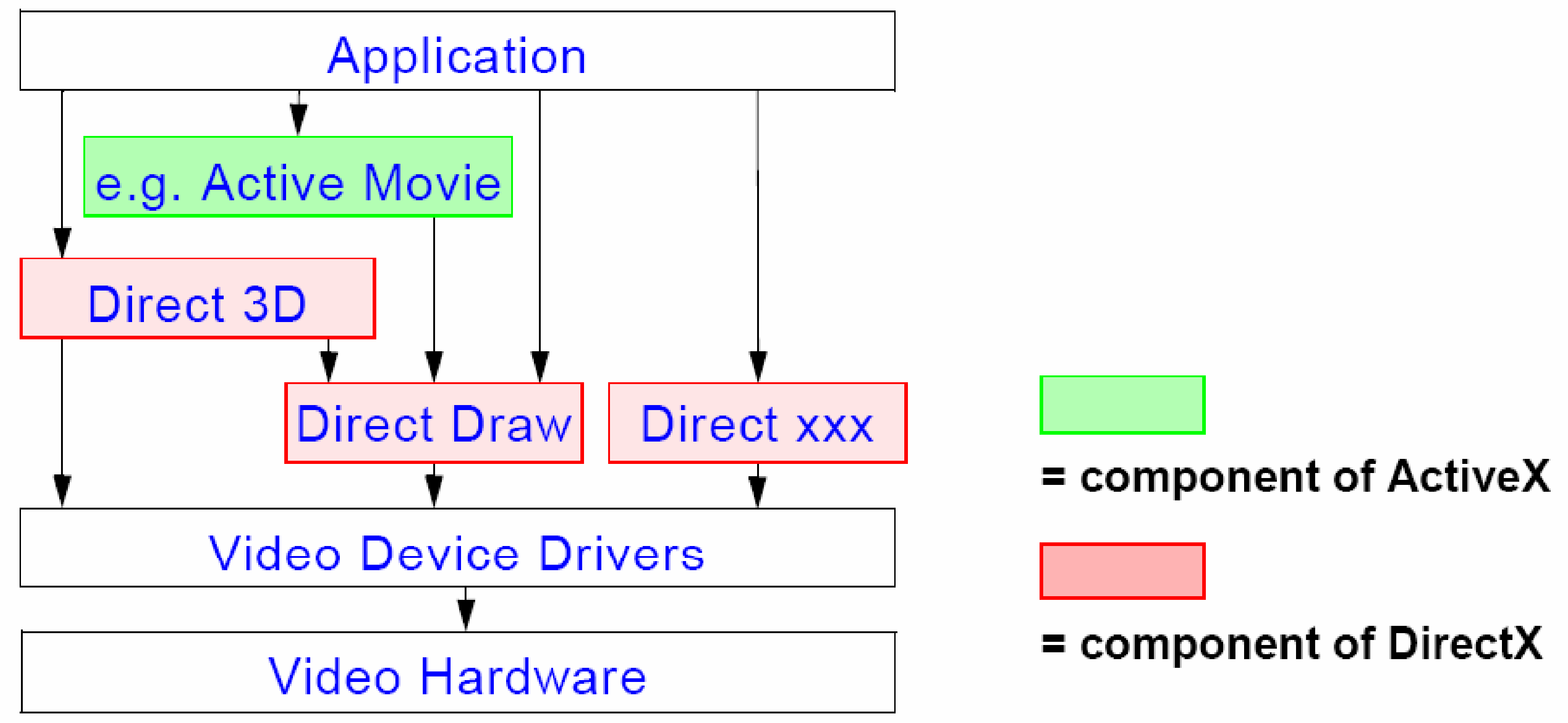


MMSYSTEM library for extensibility and device independence

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

System Software - DirectX



Application

e.g. Active Movie

Direct 3D

Direct Draw

Direct xxx

Video Device Drivers

Video Hardware

🟩 = component of ActiveX

🟥 = component of DirectX

System Software - DirectX

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

✓ Synchronization

Toolkits

Simpler approach than the system software interface from the users point of view are toolkits:

- Abstract from the actual physical layer
- Allow a uniform interface for communication with all different devices of continuous media
- Introduce the client-server paradigm
- Can be embedded into programming languages or object-oriented environments

Higher Programming Languages

Media as data types:

- Definition of appropriate data types (e.g. for video and audio)
- Smallest unit can be a LDU
- Example of merging a text and a motion picture:

Higher Programming Languages

Media as files:

- instead of considering continuous media as data types they can be considered as files:

```
file_h1 = open(MICROPHONE_1,...)
file_h2 = open(MICROPHONE_2,...)
file_h3 = open(SPEAKER, ...)
...
read(file_h1)
read(file_h2)
mix(file_h3, file_h1, file_h2)
activate(file_h1, file_h2, file_h3)
...
deactivate(file_h1, file_h2, file_h3)
...
rc1 = close(file_h1)
rc2 = close(file_h2)
rc3 = close(file_h3)
```

Programming Language Requirements

- The high-level language should support parallel processing, because the processing of continuous data is
- controlled by the language through pure asynchronous instructions
- an integral part of a program through the identification of media

Different processes must be able to communicate through an inter-process communication mechanism, which must be able to:

- Understand a priori and/or implicitly specified time requirements (QoS parameters or extracted from the data type)
- Transmit the continuous data according to the requirements
- Initiate the processing of the received continuous process on time

Object-Oriented Approaches

Basic ideas of object-oriented programming is data encapsulation in connection with class and object definitions

✓ Abstract Type Definition (definition of data types through abstract interfaces)

✓ Class (implementation of a abstract data type)

✓ Object (instance of a class)

Other important properties of object-oriented systems are:

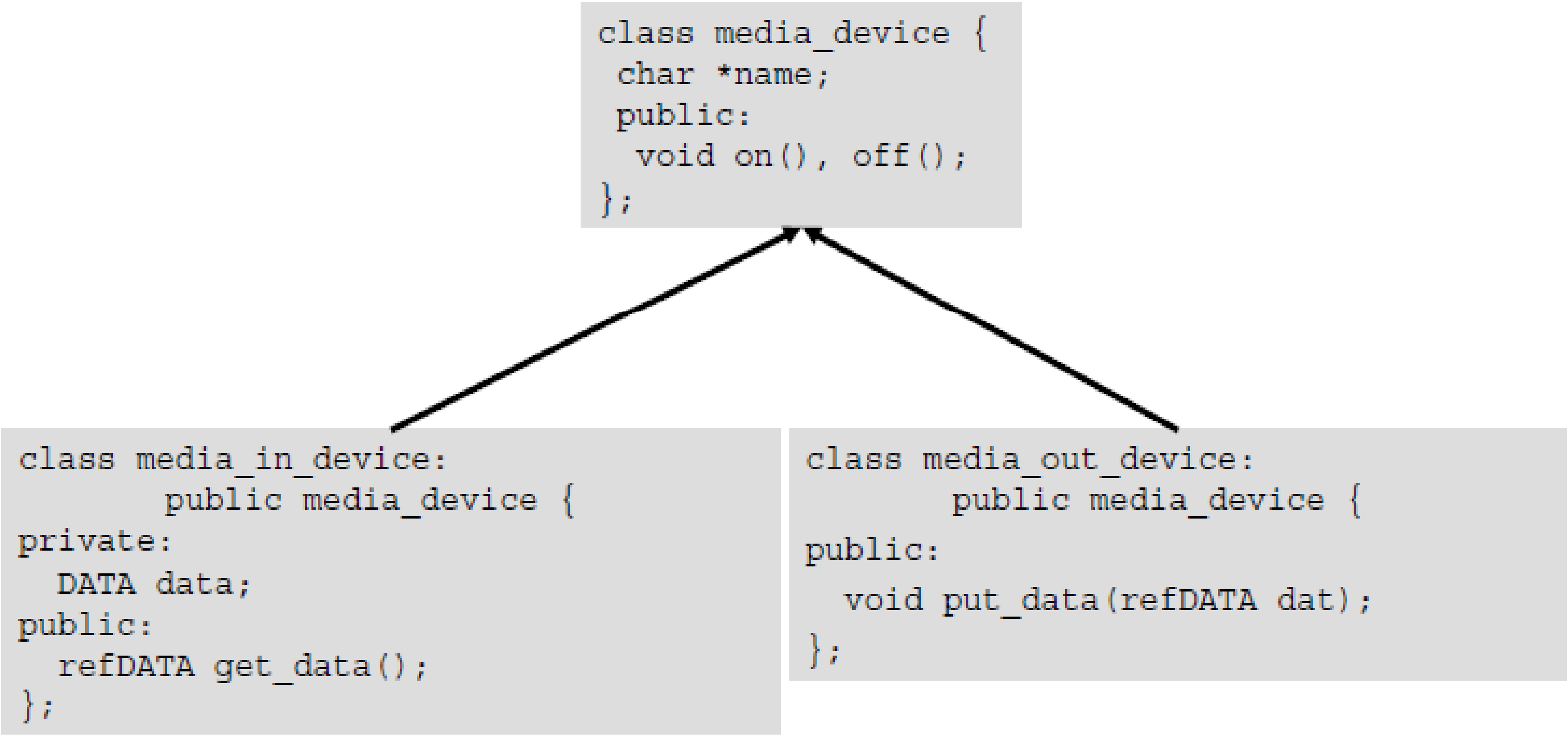
✓ Inheritance

✓ Polymorphism

Object-Oriented Approaches

- Devices as classes: devices are assigned to objects which represent their behavior and interface

Devices as classes



```
class media_device {
 char *name;
 public:
  void on(), off();
};
```

```
class media_in_device:
       public media_device {
private:
  DATA data;
public:
  refDATA get_data();
};
```

```
class media_out_device:
       public media_device {
public:
  void put_data(refDATA dat);
};
```

Object-Oriented Approaches

Processing units as classes:

- Three main objects:

✓ Source objects

✓ Destination objects

✓ Combined source-destination objects allows the creation of data flow paths through connection of objects

- Multimedia object

✓ Basic Multimedia Classes (BMCs) / Basic Multimedia Objects (BMOs)

✓ Compound Multimedia Classes (CMCs) / Compound Multimedia Objects (CMO), which are compound of BMCs / BMOs and other CMCs/CMOs

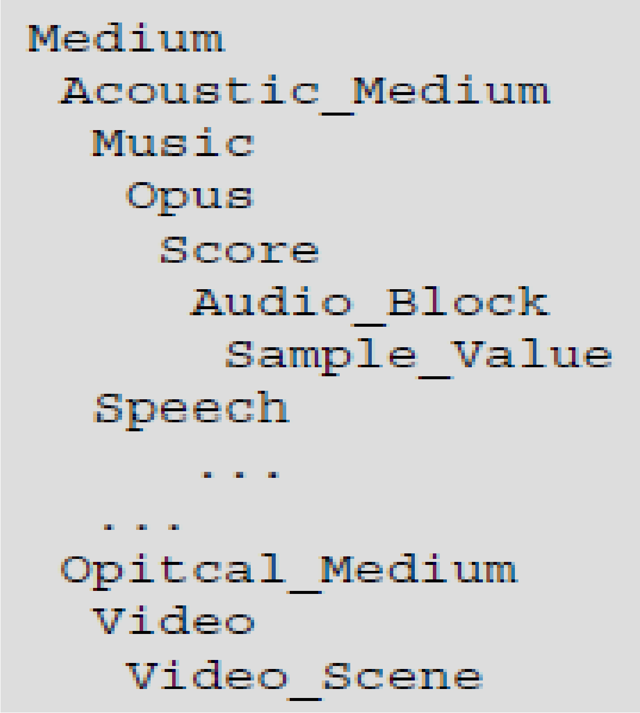
✓ BMOs and CMOs can be distributed over different computer nodes

Object-Oriented Approaches

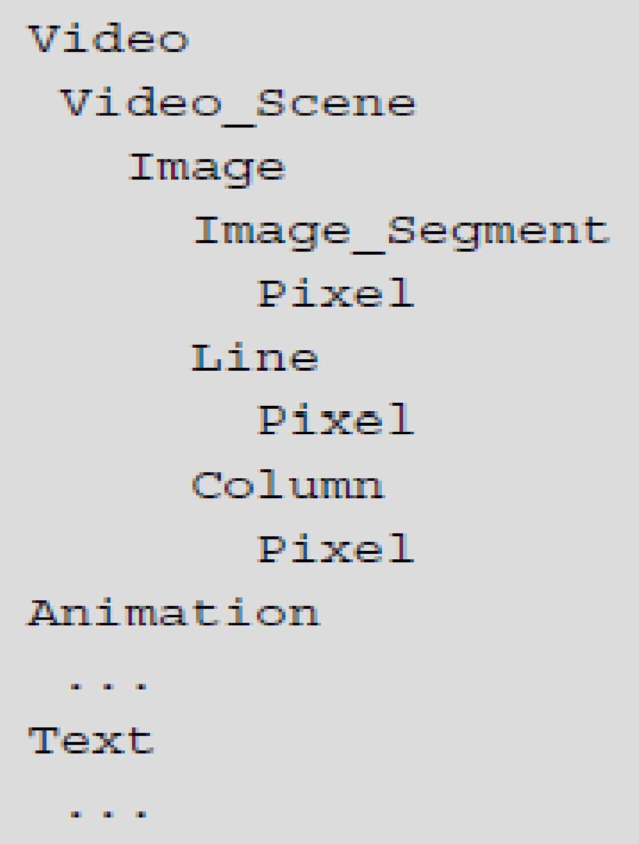
Media as classes:

- Media Class Hierarchies define hierarchical relations for different media
- Different class hierarchies are better suited for different applications

Object-Oriented Approaches-Media as Class



```
Medium
 Acoustic_Medium
  Music
   Opus
    Score
     Audio_Block
      Sample_Value
  Speech
     ...
  ...
 Opitcal_Medium
  Video
   Video_Scene
```



```
Video
 Video_Scene
   Image
     Image_Segment
       Pixel
     Line
       Pixel
     Column
       Pixel
Animation
 ...
Text
 ...
```