ABSTRACTIONS FOR PROGRAMMING

Outline

- Abstractions Levels
- Libraries
- System Software
- Toolkits
- Higher Programming Languages
- Object Oriented approaches

Lecturer: Mukesh Prasad Chaudhary

ABSTRACTIONS FOR PROGRAMMING

Programming in the past & present

- In the past, Multimedia applications are implemented in structural language i.e. based on procedure like C.
- Multimedia specific functions were called, and respectively controlled, through hardware specific libraries or device drivers.

Now / Present scenario

- Well defined abstractions in higher programming languages can be found in the form of data types (e.g. float, int in C), which hides the actual hardware configuration.
- In research, OOP approaches to the programming of multimedia systems and especially applications have been used.
- Multimedia objects help for fast integration in the environment; are used,
- Libraries, system software, toolkits etc. are emerging.

ABSTRACTION LEVELS

- Abstraction levels in programming define different approaches with a varying degree of detail for representing, accessing and manipulating data.

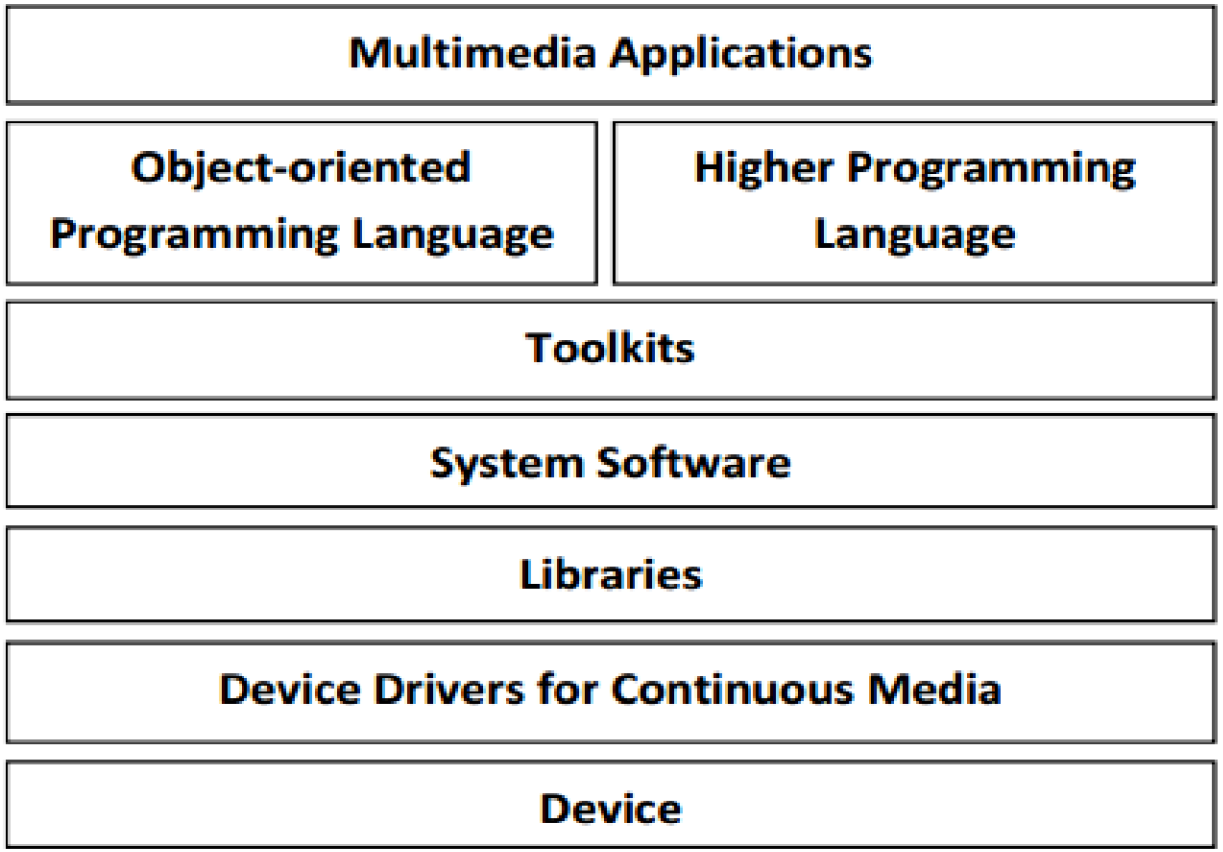


Figure: Abstraction levels of the programming of multimedia system

ABSTRACTION LEVELS Contd..

- **Device:** A separate component in a computer that is used for processing continuous media. It is directly accessible to every component and application.
- **Library:** Simplest abstraction level, which includes the necessary functions for controlling the corresponding hardware with specific device access operations.
- **Device Drivers:** It is used for bounding the multimedia devices.
- **System Software:** It does the processing of continuous data. So, for this several properties are required like-schedulers (can be monotonic scheduler or earliest deadline-first scheduler).
- **Higher procedural programming language:** Language used to implement multimedia applications contains abstractions of multimedia data.
- **Object-oriented programming language:** Provides the application with a class hierarchy for the manipulation of multimedia.

LIBRARIES

- **Libraries** contain the set of functions used for processing the continuous media.
- **Libraries** are provided together with the corresponding hardware. Some libraries can be considered as **extensions of the GUI**, whereas other libraries consist of control instructions passed as control blocks to the corresponding drivers.
- **Libraries** are very useful at the operating system level. Since, there isn't any sufficient support of OS for continuous data and no integration into the programming environment exists, so there will always be a variety of interfaces and hence, a set of different libraries.

System Software

- Instead of implementing access to multimedia devices through individual libraries, the device access can become parts of the OS. E.g. Nemo system.
- The nemo system consists of the nemo trusted **supervisor call running** in the supervisor mode and 3 domains running in user mode: system, device drivers and application.
- **System processes** implement the majority of the services provided by the OS. **Devices processes** are similar to system process, but are attached to device interrupt stubs which execute in supervisor mode. **Application processes** contains user programs.
- **Processes** interact with each other via the system abstraction – IPC (Inter Process Communication). IPC is implemented using low-level system abstractions events and, if required, shared memory.

System Software Contd..

Data as time capsule:

- **Time capsules** are the special abstraction related to the file systems. These **files extensions** serve as storage, modification and access for continuous media.
- Each logical data unit (LDU) carries in its time capsule, in addition to its data types and actual value, its valid life span. This concept is used widely in video than in audio.

Data as streams:

- A stream denotes the continuous flow of audio and video data. A stream is established between **source** and **sink** before the flow.
- Operation on a stream can be performed such as play, fast forward, rewind and stop.
- In Microsoft windows, a media control interface (MCI) provides the interface for processing multimedia data. It allows the access to continuous media streams and their corresponding devices.

Toolkits

- Toolkits are used for *controlling the audio and video data processing* in a programming environment.
- Toolkit *hides the process structures*. It represents interfaces at the system software level. **Toolkits are used to:**
  - Abstract from the actual physical layer.
  - Allow a uniform interface for communication with all different devices of continuous media
  - Introduce the client-server paradigm
  - To hide process structures.
- **Toolkits** should represent interfaces at the system software level. In this case, it is possible to embed them into the programming languages or object-oriented environment.

Higher Programming Language

- In the higher programming languages, the processing of continuous media data is influenced by a group of similar constructed functions.
- These calls are mostly hardware and driver independent. The programs in a high level language (HLL) either directly access multimedia data structures, or communicate directly with, the active processes in the real-time environment.
- The processing devices are controlled through corresponding device drivers.
- Media can be considered differently inside a programming language.
  - Media as types
  - Media as files
  - Media as processes
  - Programming language requirements
    - Inter-process communication mechanism
    - Language

Higher Programming Language Contd..

Media as types:

- One of the alternatives to programming in an **HLL** with libraries is the concept of media as types, here, the data types for **video and audio** are defined. In the case of **text**, **character** is the type.
- A program can address such characters through functions and sometimes directly through operations. They can be copied, compared with other characters, deleted, created, read from a file or stored. The smallest unit can be the **LDU**.

Media as files:

- Another possibility of programming continuous data is the consideration of continuous media streams as files instead of data types.
- Read and write functions are based on continuous data behavior. Continuous data are often played from source of non- persistent data.
- A microphone and camera are examples of such sources. The most device units are handled at their interfaces to the application as files.
- Using this kind of programming of continuous data, the number and functionality of the operations with continuous data is limited. This approach can be seen as the programming of data streams.

Higher Programming Language Contd..

Media as types: Example

E.g. Programming expression used in OCCAM-2.

a, b REAL;

ldu.left1, ldu.left2, ldu.left-mixed AUDIO_LDU;

……

WHILE

COBEGIN

PROCESS_1

Input (micro1, ldu.left1)

PROCESS_2

Input(micro2,ldu.left2)

ldu.left_mixed=a*ldu.left1+b*ldu.left2;

……

END WHILE

Media as files: Example

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)

This example describes the merging of two audio files.

Higher Programming Language Contd..

Media as Process:

- The processing of continuous data contains a time-dependency because the life span of a process equals to the life span of a connections between sources and destinations.
- A connection can exist locally, as well as remotely. The processing can be done either once or continuously, meaning that during the entire transmission of continuous data:
  - The loudness is determined by a device driver call. The driver loads a certain storage content which is used by the running process controlling an audio board.
  - If the main processor passes the audio data further from a file to the communication system, then the loudness can be changed. Thus, the compression and coding must be considered.

Example

PROCESSS cont_Process_a;

……

On_message_do

Set_Volume……..

Set_loudness…….

……..

…….

[main]

pid=create(cont_process_a)

send(pid, set_volume,3)

send(pid, set_loudness)

……

Here, the process cont_process_a implements a set of actions which apply to a continuous data

stream.

Higher Programming Language Contd..

Programming language requirements

- HLL should support parallel processing. Number of processes must be known as compile time. Process should be defined dynamically at run-time.

Inter Process communication mechanism

- The IPC mechanism must be able to transmit the audio and video in a timely fashion because these media have a limited life span. The IPC must be able to:-
  - Understand a prior and/or implicitly specified time requirements.
  - Transmit the continuous data according to the requirements.
  - Initiate the processing of the received continuous process on time.

Language

- A simple language should be developed for the purpose of simplicity.
- A partial language replacement is also quite difficult because cooperation between the real-time environment and the remaining programs requires semantic changes in the programming languages. The IPC must be designed and implemented in real-time, the current IPC can be omitted.
- An example of such language is OCCAM-2, ADA, parallel C-variant for transputer etc.

Object-Oriented Approaches

Object oriented approach is used to reduce the complexity in the software development. The basic idea behind object-oriented programming is data encapsulation & inheritance, in connection with class and object definition.

Abstract type definition

- The definition of data types through abstract interfaces is called abstract type definitions. Abstract type definition is understood as an interface specification without a knowledge and implementation of internal algorithms. Data abstraction hides the used algorithms.

Class

- The implementation of abstract data types is done through classes. A class is a collection of objects of similar type. Class is user defined data type and bahves like a built-in type of a programming language. A class is data type having data member and member functions. The syntax for class is as follows:

```
class class_name {
        Private: //data member and member functions.
        Public:  // data member and member functions
        Protected: //data member and member functions.
}
```

Object-Oriented Approaches Contd..

Object

- It is an instance of the class.
- All objects, derived from the same class include the same operations as an interface to the outside world.
- Object includes a set of operations, which are called **methods**.
- **Object communicates** among each other through the exchange of messages.

Inheritance

- **Inheritance** is the process by which the objects of one class acquire the properties of objects of another class.
- It helps to share common characteristics with the class from which it is derived, **for example**: The bird 'Robin' is a part of class 'flying bird' which is again a part of class 'bird'.
- The **concept of inheritance** provides the idea of **reusability**. This means that we can add additional features to an existing class without modifying it.

Object-Oriented Approaches Contd..

Polymorphism

- **Polymorphism** is the ability to make more than one form. **Any operation** may show different behaviors in different instances. For example: consider the operation of addition. For two numbers the operation will generate sum but if operands are strings, then the operation will produce third string by concatenation.

Application specific metaphors as classes

- An **application specific class** hierarchy introduces abstractions specifically designed for a particular application. Thus it is not necessary to consider other class hierarchies.
- This approach leads to a number of different class hierarchies.
- Unfortunately, this is currently the most used solution, which has led to different kinds of class hierarches. Although, for similar applications, similar class hierarches can be implemented.

Object-Oriented Approaches Contd..

Application-generic Metaphors as classes

- Another approach is to combine similar functionalities of all applications. These properties of functions can be defined and implemented as classes for all applications.
- An application is defined only through a binding of this class. For example: basic functions or functional units can create classes. In theory this approach sounds easy to follow. In practice we have not yet a very successful generic application classes, because they only work well for a very restricted set of application.

Devices as Classes

- The devices are assigned to objects which represent their behavior and interface. Methods with similar semantics which interact with different devices, should be defined in a device-independent manner.
- The concept of device as class hierarchies provides is not supported in this hierarchy and must be provided through other components; multiple inheritance is often needed.

Object-Oriented Approaches Contd..

Processing units as classes

- This abstraction consists of **source objects**, **destination objects**, and **combined source and destination objects** which perform intermediate processing of continuous data.
- With this approach, a kind of "logo" system is created which allows for the creation of a data flow path through a connection of objects. The outputs of objects are connected with inputs of other objects, either directly or through channel objects.

Media as Classes

- **The media class hierarchy** defines a hierarchical relation for different media. Different class hierarches are better suited for different applications.
- A specific property of all states during their life spans. Data transfer of continuous media is performed as long as the corresponding connection is active.
- A connection can be either a connection for local data transfer between source and destination, or a connection for remote data transfer. Besides, the class hierarchy, the main attributes need to be considered for different classes.

Object-Oriented Approaches Contd..

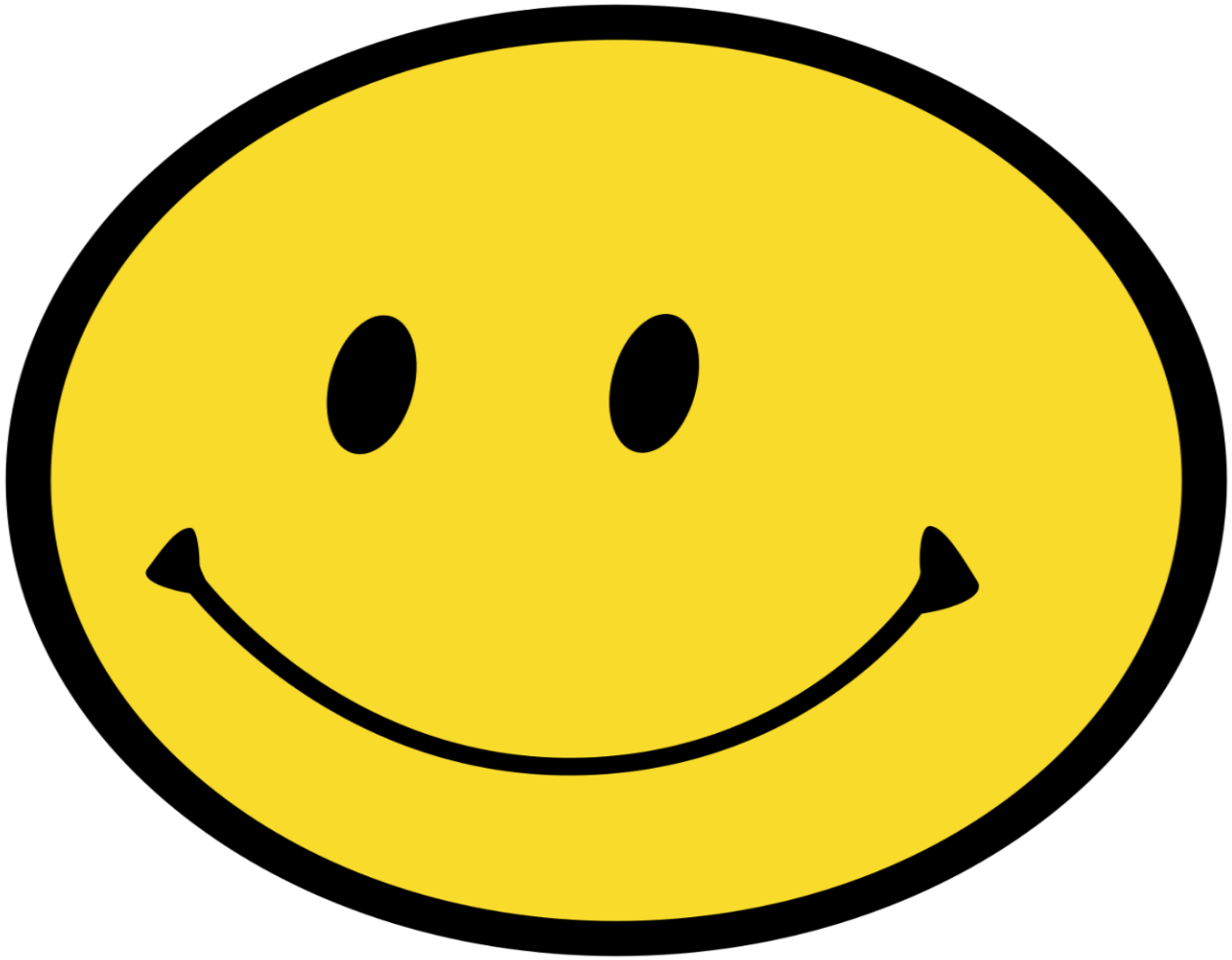
Communication-specific Metaphors as Classes

- These approaches consider **objects in a distributed environment** through an explicit specification of classes and objects tied to a communication system.
- **The information**, presentation object which is later used for presentation of information. Information objects can be converted to transport objects for transmission purposes.
- Information is often processed differently. It depends on whether the information should be presented, transmitted or stored.
- With storage objects, it is necessary to consider the different storage formats.
- Relevant formats are the coding and compression formats, format of interleaved data streams and formats such as CD-ROM ISO9660.

Object-Oriented Approaches Contd..

Distribution of BMOs and CMOs

- The channel object with the methods create connection and delete connection supports the possibility to manage connections of several media together.
- A CMO can consist of objects which are distributed over different computer nodes.
- At the moment when the CMO lexicon is active, a text of the text object is displayed.
- The CMO animation is a composition of audio and video objects.
- Destination and combined source-destination objects can also be specified as BMPs and CMOs. They may represent:
  - Output device such as windows, monitors or speakers.
  - Files of internal secondary storage devices or external storage devices.
  - Processing units of continuous media with input and output parts.



Thank you !!!