Campus Universitari de la Mediterrània

Aibo Programming

An introduction to R-CODE and OPEN-R

IX-2004

by Ricardo A. Téllez

General Index

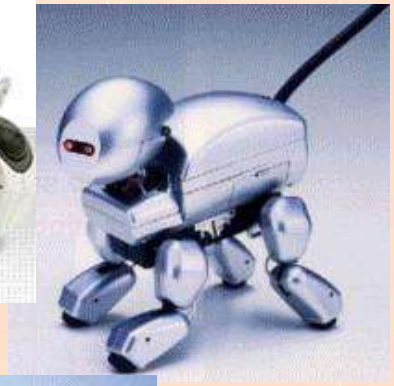
- *Short introduction to the Aibo robot*
- *Setting the environment up*
- *Introduction to R-CODE*
- *Introduction to OPEN-R*

Aibo Programming

SHORT INTRODUCTION TO THE AIBO ROBOT

Aibo Description

- *It is a robot dog created by Sony*
- *Fully programmable*
- *Several models already:*
  - *Mutant*
  - *ERS-110*
  - *ERS-210*
  - *ERS-220*
  - *ERS-7*



Aibo Description

- *For the ERS-7:*
  - *It has 18 DOF*
  - *It has several sensors:*
    - *Paw sensors (4)*
    - *Distance sensors (3)*
    - *Touch sensors (4)*
    - *Color camera (1)*
    - *Stereo micro (2)*
    - *Accelerometers (3)*

Aibo Description

- *Aibo programs are stored into memory sticks (MS)*
- *MS are plugged into Aibo to run the program*
- *You can produce any type of controller program for Aibo: neural controller, behavior based, etc...*

Aibo Description

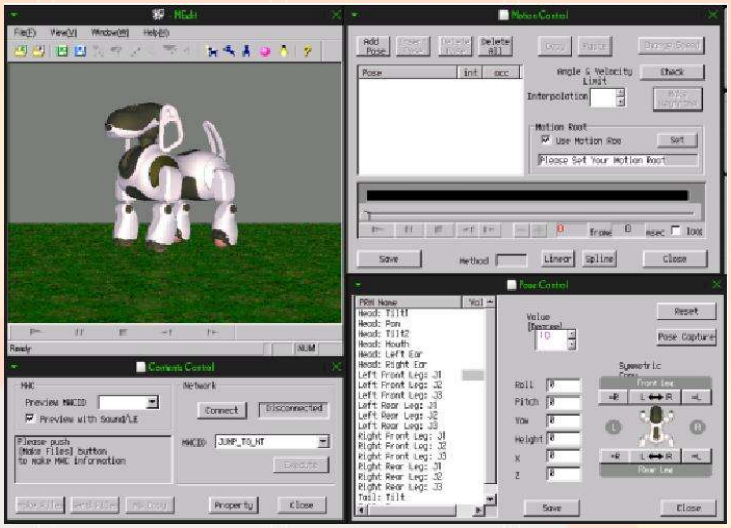
The programming environment

- *R-CODE*
  - *It is a scripting language*
  - *Easy to use and to generate behaviors*
  - *No compilation required*
  - *Complete control of the robot is not possible*
- *OPEN-R*
  - *It is a C++ Software Development Kit*
  - *Difficult to understand and to generate control architectures*
  - *C++ compilation required*
  - *Allows total control of the robot*

Aibo Description

Additional tools (released by Sony)

- *Remote Framework*
  - *Visual C++ program that runs on a PC*
  - *The program connects with Aibo*
  - *Remote control the robot*
- *Motion Editor (MEdit)*
  - *Easy creation of motions for Aibo*



Aibo Programming

SETTING THE ENVIRONMENT UP

- Installing the OPEN-R SDK on the PC (done)

- Installing the memory stick reader/writer (MS R/W)

- Installing the base system on a memory stick (MS)

- Setting up the wireless network

- Compiling a sample program

- Setting the FTP server

Setting the Environment Up

Installing the OPEN-R SDK on the PC (done)

- *For this course, this task has been done by the sysadmin*
- *Almost automatic*
- *More info in the* Aibo Quickstart Manual *and the Sony's* Installation Guide

Installing the MS reader/writer

- *Most of work done by the sysadmin*
- *Plug the MS R/W*
- *Insert a MS on it*
- *Type on a console:*
  - *mount /mnt/usb*
  - *cd /mnt/usb*

Setting the Environment Up

Installing the base system on a MS

- *Select the type of environment (Basic, Wlan or Wconsole)*
- *Select the memory protection type (memprot, nomemprot)*
- *Copy the resulting OPEN-R directory to the memory stick (cp -r OPEN-R /mnt/usb)*

Setting the Environment Up

Setting up the wireless network: configuring a wireless environment with Access Point (AP)

- *Configuration of the AP done by sysadmin*
- *Configuration of the PC done by sysadmin*
- *Configuring the Aibo wireless card:*
  - *Modify the OPEN-R\SYSTEM\CONF\WLANDFLT.TXT file of the MS with following data:*

HOSTNAME: AIBO
ETHER_IP: 147.83.60.20x
ETHER_NETMASK: 255.255.255.0
IP_GATEWAY:147.83.60.200
ESSID: ESAII-EPSEVG

WEPENABLE: 1
WEPKEY: *ESAIIEPSEVG*
APMODE: 2 (auto-mode)
CHANNEL: 3

Setting the Environment Up

Compiling a sample program: the HelloWord

- *Go to the HelloWord program directory:*
  - *cd sample_programs/common/HelloWord*
- *Compile the program*
  - *make ; make install*
- *Transfer generated code to the MS*
  - *cp -r sample_programs/common/HelloWord/MS/OPEN-R*
- *Insert the MS on Aibo and switch it on*
- *Telnet to the robot to see the result*
  - *telnet 147.83.60.20x 59000*

Setting the Environment Up

Setting the FTP server

- *Compile the FTP program*
  - *cd sample_programs/common/TinyFTPD ; make install*
- *Install the generated object on the MS*
  - *cp TinyFTPD/MS/OPEN-R/MW/OBJS/TINYFTPD.BIN /mnt/usb/OPEN-R/MW/OBJS/*
- *Install the password file*
  - *cp TinyFTPD/MS/OPEN-R/MW/CONF/PASSWD /mnt/usb/OPEN-R/MW/CONF*
- *Add line /OPEN-R/MW/OBJS/TINYFTPD.BIN to /OPEN-R/MW/ CONF/OBJECT.CFG*

Aibo Programming

INTRODUCTION TO R-CODE

Introduction to R-CODE

- *It is a scripting language similar to Basic*
- *Allows programming complicated things with a few commands*
- *An R-Code program is a text file*
- *Can be created in any operating system*

- *Example:*

```
:START
 CALL:1001
DO
 WAIT:1
 IF:AU_Voice:=:1:THEN
  WAIT:1
  SWITCH:AU_Voice_ID
  CASE:1:CALL:1003
  CASE:6:CALL:1005
  CASE:ELSE:CALL:1007
  CALL:1001
 ENDIF
 WAIT:1000
```

Introduction to R-CODE

You can easily do:

- *Put Aibo in SIT, STAND and SLEEP positions*
- *Make Aibo walk, turn around, move head, track ball*
- *Make Aibo find the ball, AIBOne and faces*
- *Make Aibo recognise verbal commands (53)*
- *Make Aibo execute contents (motions, LED, WAVs)*
- *Use your own motions, LEDs and WAVs*
- *Acquire distances to objects*

Introduction to R-CODE

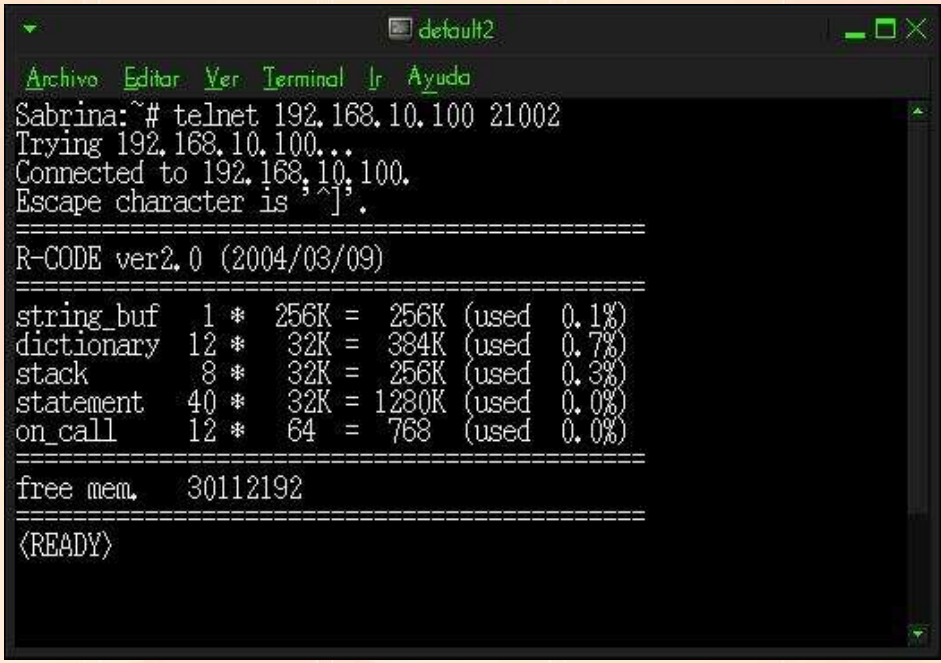
Running a R-Code program

- *Prepare the memory stick with R-Code*
  - *Copy Redist7/Eng/OPEN-R directory to empty MS*
- *Set the wireless network*
  - *Configure the WLANCONF.TXT file*
  - *Delete file /OPEN-R/APP/DATA/P/OWNER.TXT*
  - *Create file /OPEN-R/APP/PC/AMS/NOAUTH.CFG*
- *Copy your R-Code program with name R-CODE.R to /OPEN-R/APP/PC/AMS/*
- *Switch on Aibo*

Introduction to R-CODE

Using the console

- *Telnet to Aibo at port 21002*
  - *telnet Aibo_IP 21002*
- *Send commands using the console*
  - *Ex: PLAY:ACTION:SIT*
- *Use EDIT, END and RUN to send and execute a new program*



- *Use @DISS command to close connection*

Introduction to R-CODE

General considerations

- *R-Code programs are scripts (text files)*
- *Commands are words separated by colons*
- *Ex: PLAY:ACTION:TURN:90*
- *R-Code is case sensitive. Use lower case for user defined vars*
- *Only ASCII characters and underscores*
- *Use of 32 bits integers*

Introduction to R-CODE

- *To produce an R-Code program you use:*
  - *commands, relational operators, system variables and actions*
- *You can also use:*
  - *Aibo recognised words, sounds and tones*

Introduction to R-CODE

R-Code commands

- *They implement different functions like in a Basic program*
- *Each line is a command*
- *They can be sent individually through the console*
- *Examples:*
  - *ADD, FOR, IF, LET, WAIT, GO, CALL, RETURN*

R-Code operators

- *= Equals*
- *== is equal to*
- *<> not equal to*
- *< less than*
- *> greater than*
- *&& AND*
- *|| OR*

Introduction to R-CODE

System variables

- *Describe the status of the robot*
- *Can be checked or set to act consequently*
- *Examples:*
  - *Face, Pink_Ball, Pink_Ball_D, AU_voice, Distance, Head_ON*

Aibo actions

- *Actions can be played by Aibo with command PLAY:ACTION or PLAY:MWCID*
- *Examples:*
  - *SIT, LIE, KICK, TURN, SEARCH, TRACK_HEAD*

Introduction to R-CODE

Recognised words

- *Use the AU_Voice variable to detect recognition*
- *Use the AU_Voice_ID variable to identify the word said*

Debugging

- *Use the console to debug your programs (EDIT,END and RUN)*
- *Use VDUMP to display var names:*
  - *VDUMP:<var name>*
- *Use PRINT to display comments:*
  - *PRINT:<format>:<vars>*

Aibo Programming

INTRODUCTION TO OPEN-R

Introduction to OPEN-R

OPEN-R program

A set of OPEN-R objects running concurrently that communicate between each other.

Objects are like **PROCESSES** in Aibo's computer

Objects inherit from the base class **OObject**

Objects are composed of a set of **internal states**

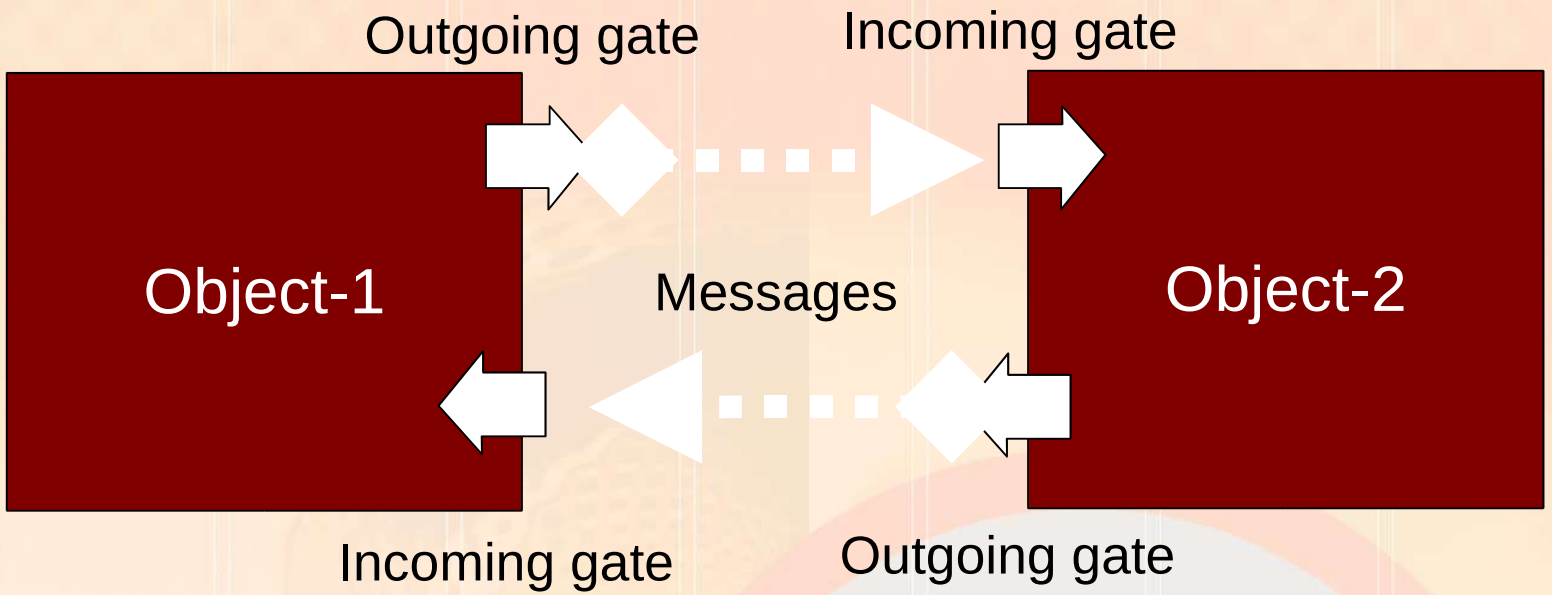
They must have defined virtual functions **DoInit**, **DoStart**, **DoStop** and **DoDestroy**

Ex: HelloWord

(change HelloWord to print a *bye* message)

Introduction to OPEN-R

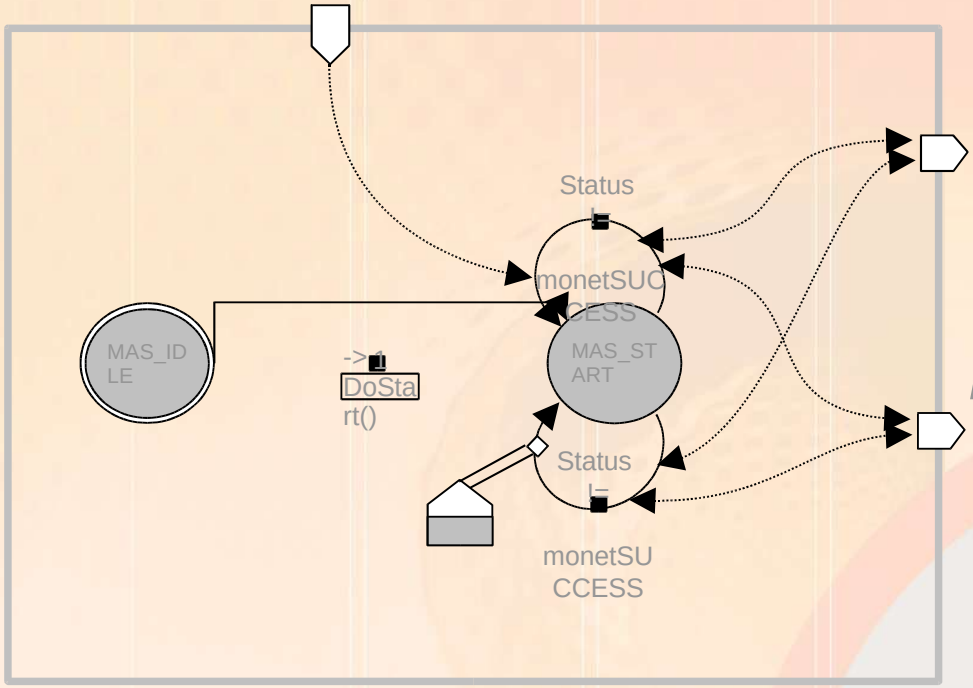
Objects communicate through **GATES** by using **MESSAGE** passing (allows coordination)



Gates are unidirectional. Two gates required for bidirectional communication

Introduction to OPEN-R

*Objects are composed of **internal states**. Transitions between states are started by reception of messages from other objects (event oriented programming)*



*The sender of the message is called the **subject**. The receiver is called the **observer***

Messages can be of any type of data.

*The **Assert_Ready** (AR) message indicates readiness*

Introduction to OPEN-R

How to implement an object (ex:ObjectComm)

- *By inheriting from the base class OObject*
- *Create the virtual functions DoInit, DoStart, DoStop and DoDestroy*
- *Define the states of the object*
- *Create the constructor*
- *Define the connections with other objects (stub.cfg file)*
- *Create the class required procedures to send, receive and process messages*

Introduction to OPEN-R

***DoInit** procedure*

- *Called when object loaded in memory*
- *Sets up gates and registers observers and subjects of the object*
- *Use OPEN-R macros to do the job*

```
OStatus
SampleObserver::DoInit(const
OSystemEvent& event)
{
NEW_ALL_SUBJECT_AND_OBSERVER;
REGISTER_ALL_ENTRY;

SET_ALL_READY_AND_NOTIFY_ENTRY
;
return oSUCCESS;
}
```

Introduction to OPEN-R

***DoStart** procedure*

- *Called when DoInit finished in all objects*
- *Sends AR message to all observers*
- *May change from IDLE to another state*
- *Use OPEN-R macros to do the job*

```
OStatus
SampleObserver::DoStart(const
OSystemEvent& event)
{
ENABLE_ALL_SUBJECT;

ASSERT_READY_TO_ALL_OBSERVER;
    return oSUCCESS;
}
```

Introduction to OPEN-R

***DoStop** procedure*

- *Called at shutdown of the system*
- *Sends DR message to all observers*
- *Changes to IDLE state*
- *Use OPEN-R macros to do the job*

```
OStatus
SampleObserver::DoStop(const
OSystemEvent& event)
{
DISABLE_ALL_SUBJECT;

DEASSERT_READY_TO_ALL_
OBSERVER;
    return oSUCCESS;
}
```

Introduction to OPEN-R

***DoDestroy** procedure*

- *Called after DoStop finished in all objects*
- *Deletes all objects*
- *Use OPEN-R macros to do the job*

```
OStatus
SampleObserver::DoDestroy(const
OSystemEvent& event)
{

DELETE_ALL_SUBJECT_AND_
OBSERVER;
    return oSUCCESS;
}
```

Introduction to OPEN-R

The stub.cfg file defines the gates of the object (one file per object)

ObjectName : SampleObserver
NumOfOSubject : 1
NumOfOObserver : 1
Service : "SampleObserver.DummySubject.DoNotConnect.S", null, null
Service : "SampleObserver.ReceiveString.char.O", null, Notify()

The connect.cfg file defines how objects inter-connect (one file per program)

SampleSubject.SendString.char.S SampleObserver.ReceiveString.char.O

OBJECT.CFG file contains objects to be executed

/MS/OPEN-R/MW/OBJS/POWERMON.BIN
/MS/OPEN-R/MW/OBJS/SUBJECT.BIN
/MS/OPEN-R/MW/OBJS/OBSERVER.BIN

add FTP!

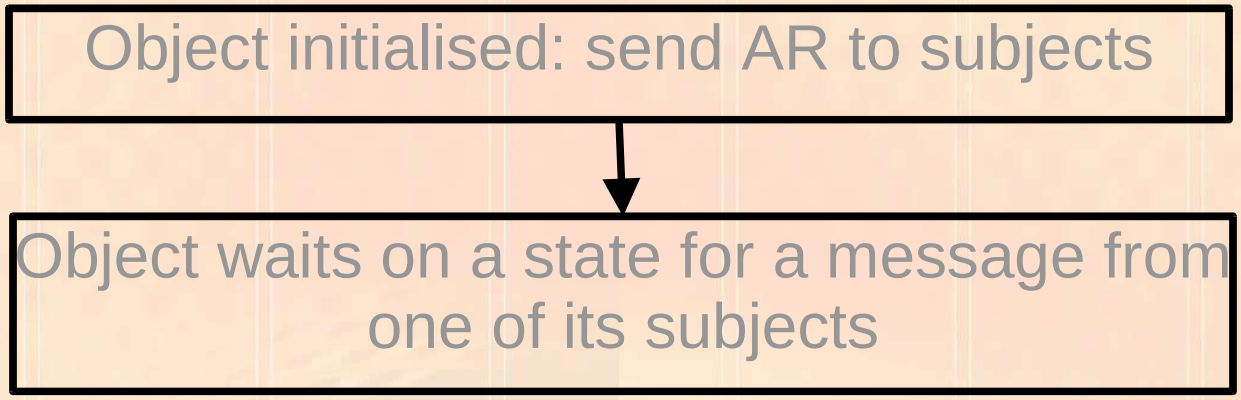
Introduction to OPEN-R

An object's life

Object initialised: send AR to subjects

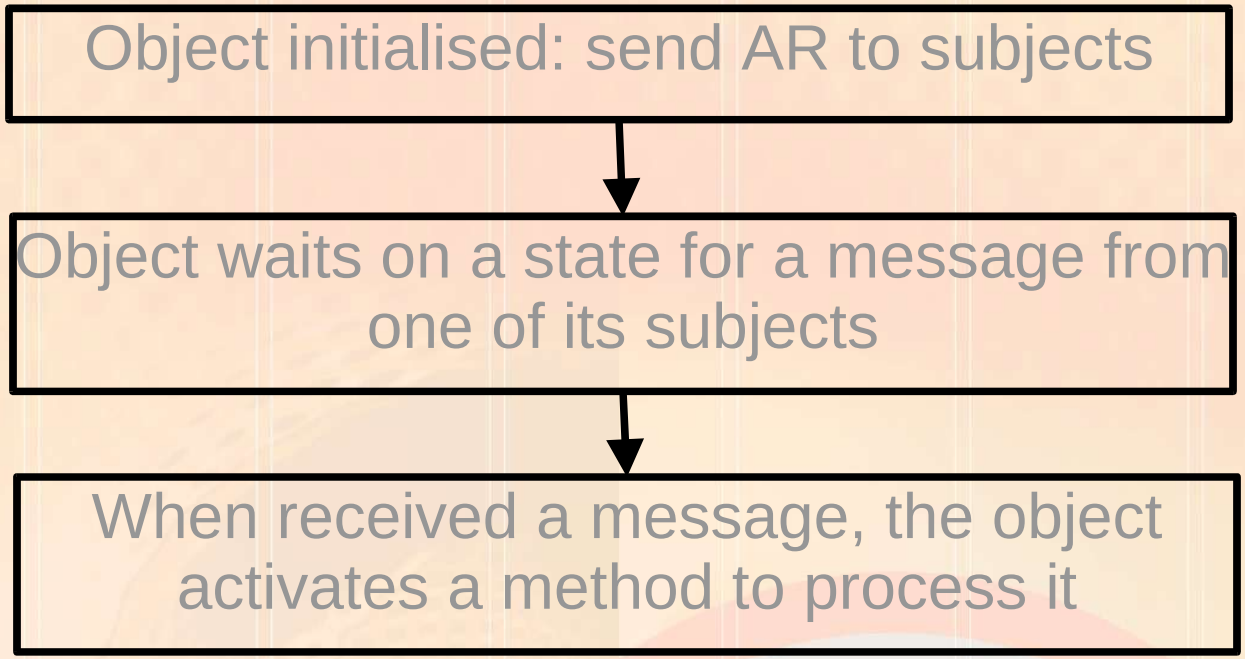
Introduction to OPEN-R

An object's life



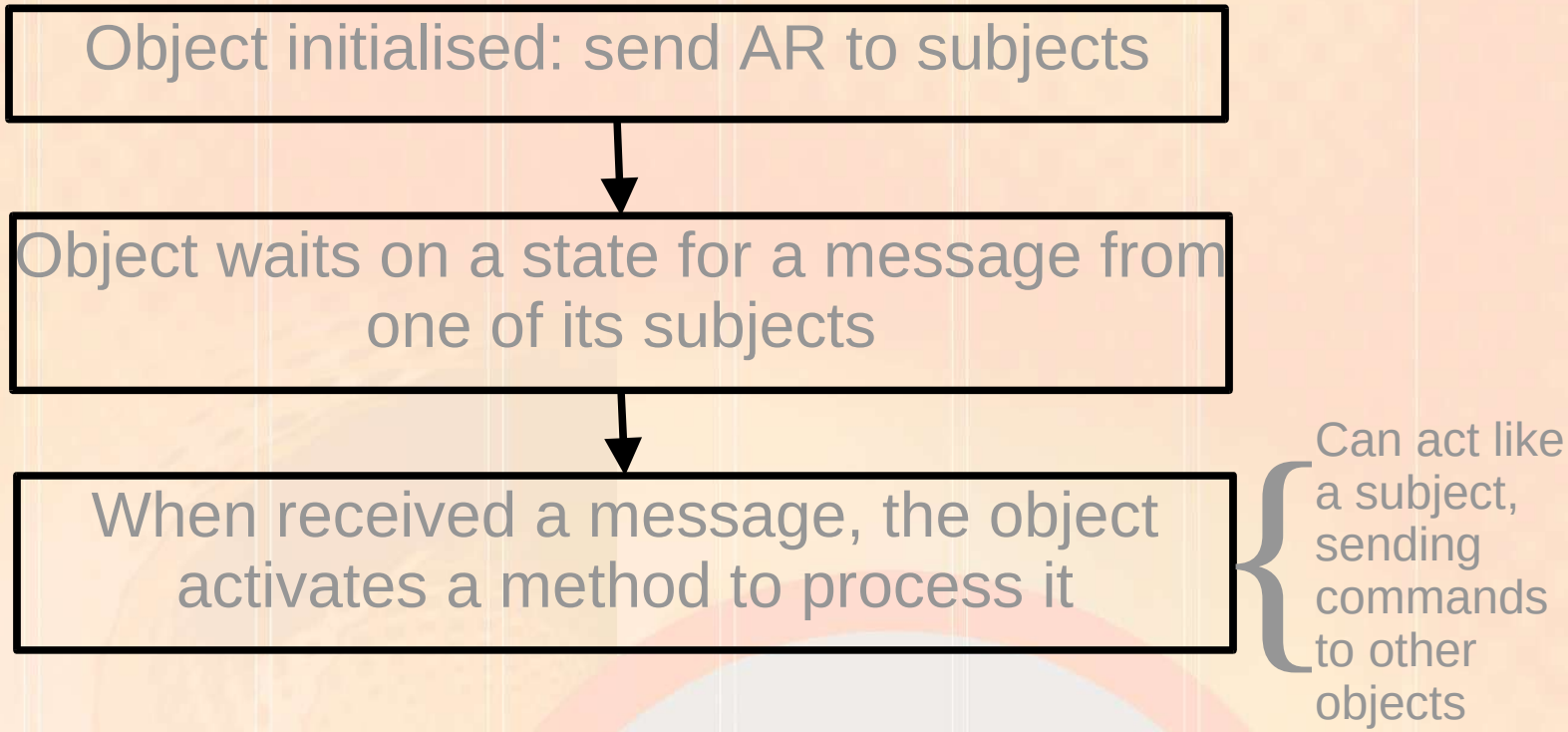
Introduction to OPEN-R

An object's life



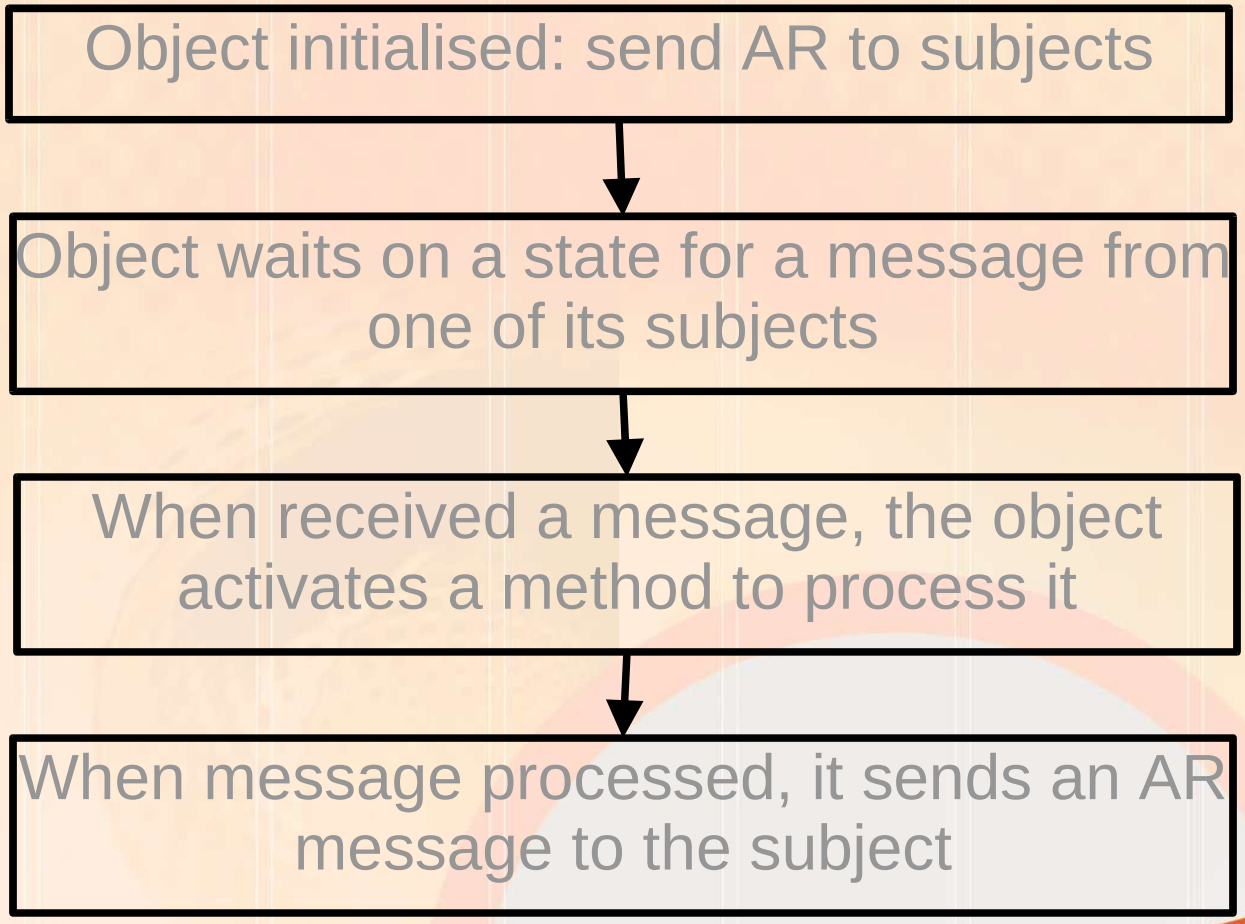
Introduction to OPEN-R

An object's life



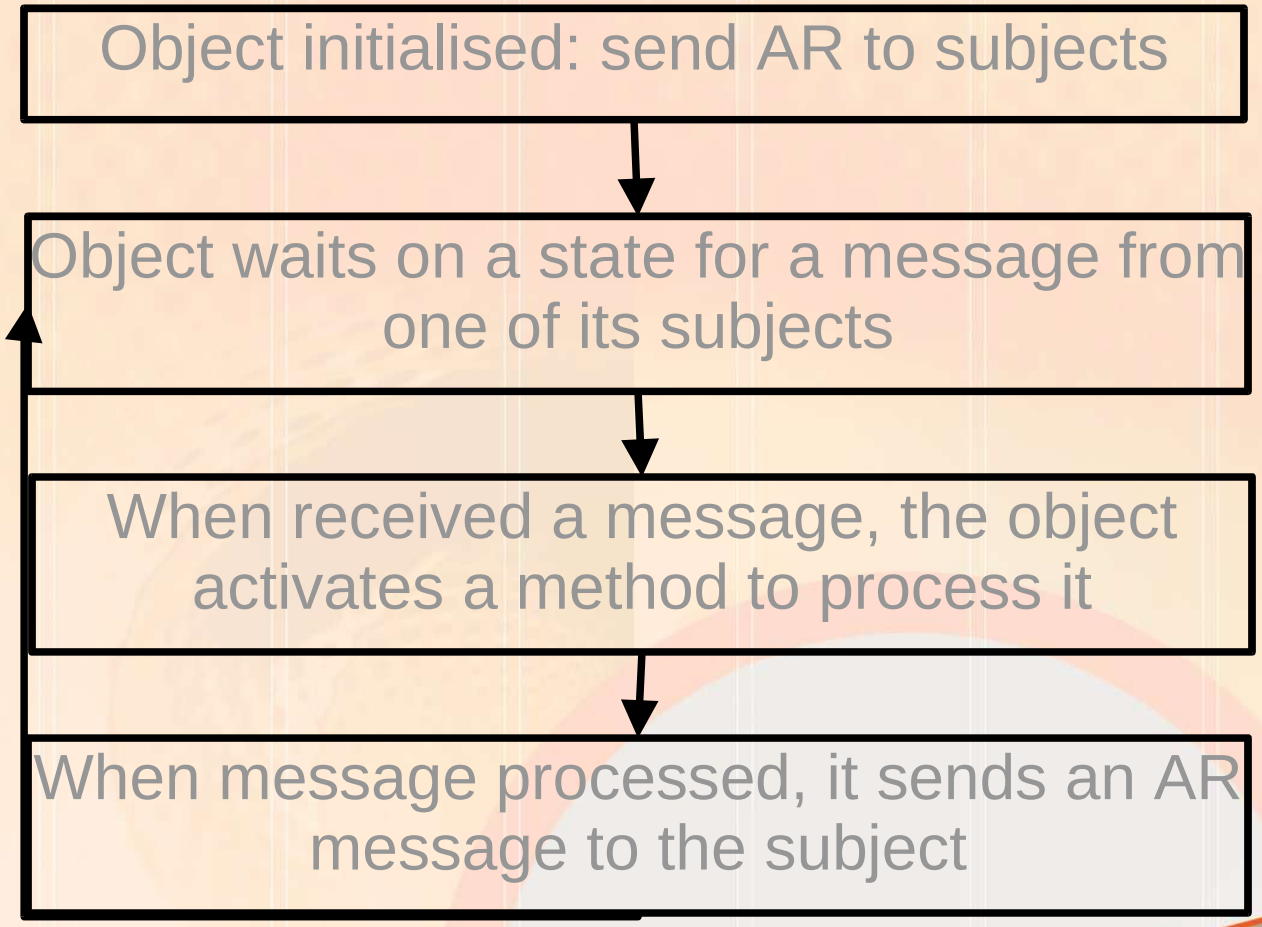
Introduction to OPEN-R

An object's life



Introduction to OPEN-R

An object's life



Introduction to OPEN-R

Two special objects:

- *OVirtualRobotComm*

In charge of implementing the access to sensors, actuators and camera

- *OVirtualAudioRobotComm*

In charge of implementing the audio interaction with the robot

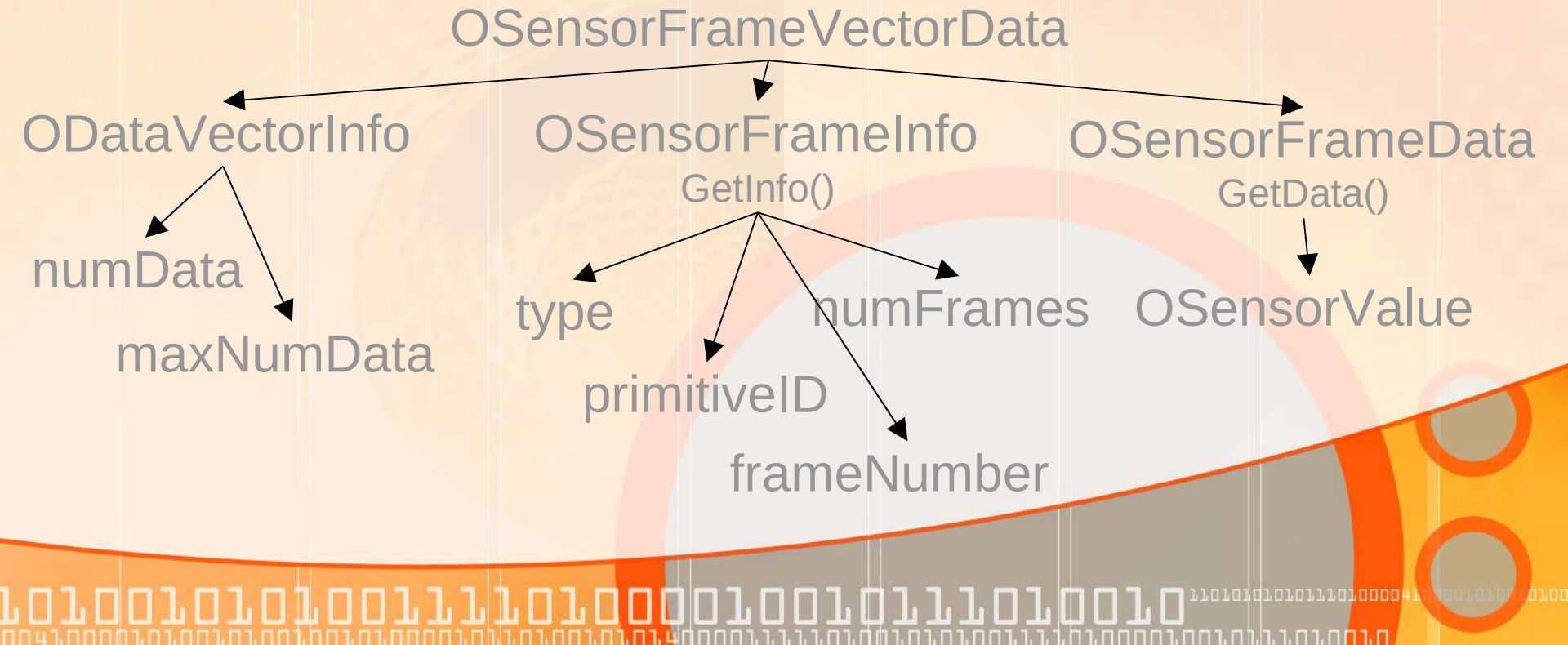
Programmer's objects must communicate with them in order to obtain sensors and audio values, and to send commands to actuators

They act like a normal OPEN-R object

Introduction to OPEN-R

*Sensor information is obtained from the **Sensor** gate of OVirtualRobotComm*

*Data obtained is a structure of type **OSensorFrameVectorData***



Introduction to OPEN-R

- *Two points to clarify*
  - *Data from sensors is obtained in **frames***
  - *Any sensor and actuator has its own **primitive** to access to it.*

"PRM:/a1-Sensor:a1", // ACCELEROMETER Y

But OSensorFrameVector uses primitive's ID

See SensorObserver7 example

Introduction to OPEN-R

To obtain a sensor value:

- Get the primitive of the sensor

  "PRM:/a1-Sensor:a1"

- Get the primitive ID with OPENR::OpenPrimitive()

```
result = OPENR::OpenPrimitive(ERS7_SENSOR_LOCATOR[i], &sensorID);
```

- Compare ID with the one given by OSensorFrameInfo and obtain its index

```
OSensorFrameInfo* info = sensorVec->GetInfo(j);
        if (info->primitiveID == sensorID) {
```

- Store index in user array

```
ers7idx[i] = j;
```

.....continue ->

Introduction to OPEN-R

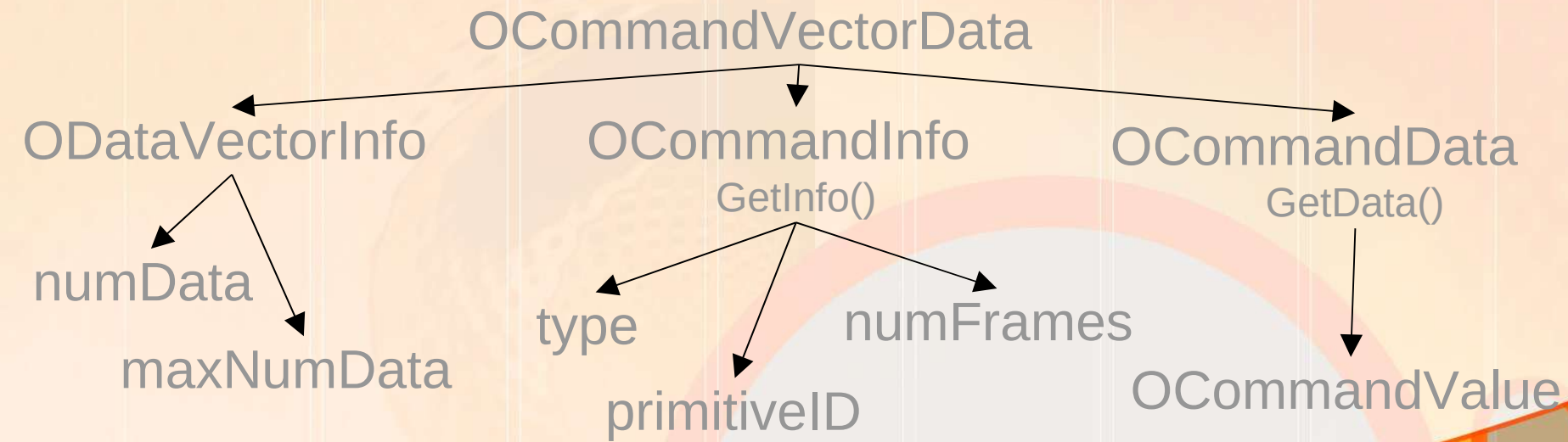
- *Use the index with OSensorFrameData to access sensor value*

```
OSensorFrameData* data = sensorVec->GetData(index);
  OSYSPRINT(("[%2d] val    %d %d %d %d\n",
          index,
          data->frame[0].value, data->frame[1].value,
          data->frame[2].value, data->frame[3].value));
```

Introduction to OPEN-R

*Commands to actuators are sent through the **Effector** gate of OVirtualRobotComm*

*Data sent is a structure of type **OCommandVectorData***



Introduction to OPEN-R

Steps to send a command

- *Initialization*
- *Setting joint gains*
- *Calibrating joints*
- *Using shared memory region*
- *Setting the joint value*