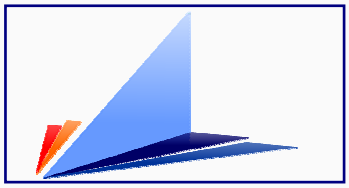
Teaching in English 06-07



Management Computing

Escuela Politécnica Superior

Subject: Object Oriented Programming

Lecturer: Estela Saquete Boro
Credits: 4,5
Dates: From September 2006 to January 2007

I. OBJECTIVES

- To introduce to the student in the philosophy of the Object oriented programming (OOP).
- To introduce a general knowledge about the main concepts related with OOP: encapsulation, messages passing, etc.
- To introduce the concept and use of the inheritance for the resolution of problems.
- To introduce the concept and use of the polymorphism for the resolution of problems.
- To identify the concepts explained in a real problem.
- To present a simple standard annotation of OO Design.
- To present different Programming languages related to the object-oriented programming.

II. SYLLABUS

1. Introduction to object oriented programming.
2. Object oriented programming foundations.
3. Inheritance.
4. Polymorphism.
5. Exception handling.

III. METHODOLOGY

Learning will be based on both lectures and practical exercises:

- Theoretical lessons will be given in order to obtain a broad overview of each topic.
- Practical lessons will consist in developing an application divi order to apply the main concepts of OOP.

IV. ASSESSMENT

- A practical work with 3 parts (60%).
- A written examination (40%).

.

V. SELECTED BIBLIOGRAPHY

- An Introduction To Object Oriented Programming 3rd Edition. T. Budd. Addison-Wesley.
- C++ how to program. H. M. Deitel, P. J. Deitel. Prentice Hall, Upper Saddle River (cop. 2003).
- C++ Primer. 3rd. Edition. S. B. Lippman. Addison-Wesley.
- Requirements Analysis and System Design. Developing Information Systems with UML. L.A. Maciaszek, Addison-Wesley.

Subject: Programming Tools

Lecturer: Jose Oncina Carratalá
Credits: 6
Web Site: (http://www.dlsi.ua.es/asignaturas/hp/)
Dates: From September 2006 to January 2007

I. OBJECTIVES

The purpose of this subject is to endow the Engineers in Computer Science with the necessary knowledge to develop, in a successful way, big applications, either because they are programmers, or because they carry out coordination tasks in a programmers group.

In professional programming environments there exists a "programming culture" due, among other things, to the use of a series of tools and a methodology of work that, usually, it is not know out of them and therefore, it is not used in the practice.

We can found software development companies whose operations are totally anarchic in this subject: minimal or none coordination among the programmers, deficient management of the software versions, not acquainted of the full capacities of the tools they use, etc.

In this subject we try to show all the necessary information to our students could with the aim of carrying out programming related tasks in the most appropriate and efficient way.

To attain it we raise the following objectives:

1. To know a whole series of tools for the management and treatment of text and binary files.
2. To show series of guidelines/procedures to keep in mind during the development of great scale applications and team-group tasks (identifiers nomenclature standards, calling application options standards, code source indent and comments standards, as well as, for every project, standards for the distribution of the code in tree directories).
3. To learn how to express dependences between files with "make" type tools. How to apply it to the management and development of applications. And to get expertise with tools for semi-automatic and portable creation of "makefiles".
4. To know in detail the use of the compiler, linker and debugger: the usual and the less known options (warnings activation, optimization, architecture selection, etc.).
5. How to find and eliminate mistakes due to bad use of the dynamic memory.
6. To learn how to create and to apply "tests" in order to verify the correct operation of an application in an automatic way.
7. To learn how to reduce the execution time of an application detecting which are its "bottle-necks" (profiling).
8. To learn how to use the formal verification in the practice (programming by contract).

9. To know how to “localize” (i18n) or to translate, in a simple way, the messages that an application presents to the user.
10. To learn how to use “version control” tools for the development of a software project.
11. To learn how to construct extensible applications (extension languages).
12. To know the different types of software licenses that the programmer could find using third party software packages.

Moreover, all the tools and packages used in the development of this subject are of the open source software kind, which will allow the student:

1. To obtain easily and with the appropriate documentation. All this without any cost.
2. To execute it personal computer working under Unix* or Windows* operating systems.
3. To compare it with commercial alternatives.

II. SYLLABUS

1. Basic tools.
- Use of an interpreter of commands.
- Specific tools for text files.
- Specific tools for binary files.
- Generic tools for files.
- Text Editors.
2. A bit of order...
- Writing code standards. Correct use of comments (automatic documentation generation from them).
- Standards for project code organization. How to distribute it in a directory tree.
3. Compilation, link and debugging.
- Compiler and linker.
- Debugging.
- Project management with “make”. Other similar tools: “ant”, “jam”.
- “Auto-tools”: “Autoconf”, “Automake”, “Libtool”.
4. Detection and correction of mistakes.
- Dynamical memory mistakes.
- Use of tests.
- Programming by contract. Creation of software free of mistakes (or at least, easily detectable).
5. Other tools.
- Code optimization (gprof, gcov).
- Localization (i18n) of applications. How to automate the translation to other languages.
- Version control.
- Extensible applications. Extension languages.
- IDEs - Integrated Environments of Development.

6. Types of software licenses.

III. METHODOLOGY

Learning will be based on both lectures and practical exercises:

- Theoretical lessons will be given in order to obtain a broad overview of each topic.
- Practical computer lessons will consist in exercises with different tools.

IV. ASSESSMENT

The evaluation consists of two parts (reports):

1. An individual obligatory exercise of auto-evaluation in which the pupil should demonstrate his capacity to overcome the examination. The assessment of this exercise will provide the student with and orienting mark. It is indispensable to fulfil this exercise to sit in the following one.
2. An individual practical examination in the laboratory that will give the qualification in the subject. To this one the student can bring with her/his all the auxiliary documentation that he/she considers opportune.

V. SELECTED BIBLIOGRAPHY

No book covers all the tools studied in this subject. The main information sources are the manual pages and the documentation that comes with each tool. Please, see the web page: http://www.dlsi.ua.es/asignaturas/hp/bibliografia.html for further details.

Subject: Computer Architecture

Lecturer: José García Rodríguez
Credits: 7,5
Dates: Second semester

I. OBJECTIVES

The global objective of the subject is to introduce the basic and general concepts on the structure, organization and operation of the computers. More specifically:

- To identify and to understand the internal organisation of a computer, being structured this one in its functional units.
- To analyze the operation of the different functional units from the computer.
- To identify the functional elements as parts in the execution of an instruction.
- To determine how must be connected these elements to manage the information transference.
- To know the main characteristics the different storage systems that is possible to find in the computer establishing a suitable classification.
- To understand the functions that the input/output system makes and how they are carried out.
- To approximate the concept of architecture and analysis of the performance.
- To understand the segmentation as a technique to improve the performance.
- To make programs in assembly languages.
- To design and modify the structure of a simple computer.

II. SYLLABUS

0. Presentation (1 session).
1. Introduction to the computer organisation (2 sessions).
2. Control unit (7 sessions).
3. Arithmetic-logic Unit (4 sessions).
4. Memory management (6 sessions).
5. Input/output (4 sessions).
6. Buses (2 sessions).
7. Introduction to the computer architecture. Parameters of performance (4 sessions).

III. METHODOLOGY

Learning will be based on both lectures and practical exercises:

- Theoretical lessons will be given in order to obtain a broad overview of each topic.
- Practical lessons will consist in exercises in the use of a simulator to programming with assembler code.

IV. ASSESSMENT

- Some practical exercises and final practical exam (30%).
- A written examination (70%).

V. SELECTED BIBLIOGRAPHY

- Grediaga, M. L. Rico, A. Soriano y A. Párraga. Computer Organization, Servicio de Publicaciones de la Universidad de Alicante, 1999.
- Soriano, A. Gredaga, J. García y F. J. Mora. Computer Organization*: Solved Problems*, Servicio de Publicaciones de la Universidad de Alicante, 2002.
- J.L: Hennessy y D.A. Patterson. Computer Architecture a quantitative approach, McGraawHill, 1996.
- D. A. Patterson y J. L. Hennesy. Computer Organization and Design: the hardware/software interface, Morgan Kauffman, 1997.
- W. Stallings. Computer Organization and Architecture, Prentice Hall, 2003.

Subject:Semi-structured Information Systems (XML Information Systems)

Lecturer: Pedro Pastor Seva
Credits: 6
Web Site :
Dates: From February 2007 to May 2007

I. OBJECTIVES

- To introduce the XML meta-language and XML Information Systems.
- To introduce the concept of XML databases and how to use it in the web.
- To introduce XML style-sheets and programming.

II. SYLLABUS

1. Introduction to XML and XML Information Systems.
2. Description DTD language: validating XML.
3. Description of XML-Schemas language: XML Database concepts.
4. XML publishing systems.
5. XSLT language for XML transforms.

III. METHODOLOGY

Learning will be based on both lectures and practical exercises:

- Theoretical lessons will be given in order to obtain a broad overview of each topic.
- Practical lessons will consist in exercises in the different system developing aspects of XML, and using different XML technologies.

IV. ASSESSMENT

- A practical exercise to run in the web (70%).
- A written examination (30%).

V. SELECTED BIBLIOGRAPHY

- XML, The Annotated Specification. Bob DuCharme. Prentice Hall 1999.
- XSLT Programmer's Reference (2nd Edition). Michael Kay. Wrox 2001.
- Professional XML Schemas. Jon DUckett et alt. Wrox 2001
- SAX-2. David Brownell. OReilly, 2002.
- XPath, XLink, XPointer. A Practical Guide to Web Hyperlinking. Erik Wilde. Addison-Wesley 2003.
- XSLT 2.0, Programmer's Reference (3rd Edition). Michael Kay. Wrox, 2004.

Subject: Autonomous Robots

Lecturer: Miguel Ángel Cazorla Quevedo, Otto Colomina Pardo
Credits: 6
Web Site:
Dates: From February 2007 to May 2007

I. OBJECTIVES

- Knowing the areas where a robot is used.
- Applying algorithms to solve robotics tasks:
  - Path planning.
  - Mapping.
  - Localization.
  - Behaviors.
- Knowing to use several sensors: cameras, laser, sonar.

II. SYLLABUS

1. Introduction.
2. Coordinate systems and linear transformation.
3. Geometrical and motion models.
4. Sensors.
5. Robot Vision.
6. Obstacle avoidance.
7. Mapping and localization.
8. Robotics applications.

III. METHODOLOGY

Learning will be based on practical exercises and several theoretical classes.

IV. ASSESSMENT

Several practical assessment.

V. SELECTED BIBLIOGRAPHY

- P. J. McKerrow. Introduction to Robotics. Addison-Wesley, 1991.
- R. Arkin. Behavior Based Robotics. The MIT Press, 1998.
- D. Dudek and M. Jenkin. Computational Principles of Mobile Robotics. Cambridge University Press, 2000.
- J.Latombe. Robot Motion Planning. Kluwer Academic Publisher, 1991.
- R. Murphy. Introduction to AI Robotics. The MIT Press, 2000.
- E.Trucco and A.Verri. Introductory Techniques for 3-D Computer Vision. Prentice Hall, 1998.

Subject: Computer Vision

Lecturer: Francisco Escolano Ruiz
Credits: 6
Web Site: (http://www.rvg.ua.es/moodle)
Dates: Second Semester

I. OBJECTIVES

- Understand modern (state-of-the-art) Computer Vision algorithms and their limitations and applications.
- Implement Computer Vision algorithms using JavaVis.

II. SYLLABUS

1. Introduction to Robot Vision.
2. Region Segmentation. Grouping, Textures.
3. Active Contours and Deformable Templates.
4. Object recognition. PCA-based, Constellation methods.
5. Motion estimation. Optic flow methods vs Feature-based methods.
6. Tracking algorithms. Kalman Filter and Particle Filters.
7. Stereo vision and Camera calibration.

III. METHODOLOGY

Learning will be based on both lectures and practical exercises:

- Theoretical lessons will be given in order to obtain a broad overview of each topic.
- Practical lessons will consist of introducing the JavaVis Computer Vision Library http://javavis.sourceforge.net/ and explaining the implementation of several algorithms, previously to implement the algorithm chosen by the student.

IV. ASSESSMENT

- A practical exercise addressed to implement a given algorithm. Students may choose between the algorithms explained in the classroom (segmentation, tracking, PCA-recognition, and so on). Finally, they must report experimental results and conclusions (80%).
- A short report on one of the topics explained in theory including web searching on related papers (20%).

V. SELECTED BIBLIOGRAPHY

- ✓ E. Trucco & A. Verry. Introductory Techniques for Computer Vision. Prentice Hall. 1998.
- ✓ Blake & Yuille. Active Vision. MIT Press. 1992.
- ✓ Specific papers describing the algorithms and their applications R. Murphy. Introduction to AI Robotics. The MIT Press, 2000.

Subject: Advanced Computer Graphics and Animation

Lecturer: Rafael Molina Carmona
Credits: 6
Web Site:
Dates: Second Semester

I. OBJECTIVES

- Reflection about the context of Computer Graphics, from the analysis of their problems and techniques.
- Contact with CG literature.
- Use of CG terminology.
- Identification of development languages and tools for CG.
- Understanding, knowledge, analysis and application of CG methods to solve problems.
- Understanding of advanced rendering methods.
- Understanding of the Aliasing problem and how to improve the display using antialiasing methods.
- Knowledge of how to apply textures on synthetic objects.
- Relation of rendering and time, to generate computer animations.

II. SYLLABUS

1. Introduction to advanced CG.
2. Lighting in CG.
3. Ray tracing.
4. Radiosity.
5. Introduction to Aliasing problem.
6. Textures.
7. Introduction to animation.
8. Animation of articulated structures.
9. Animation of soft objects.
10. Procedural animation.

III. METHODOLOGY

Learning will be based on both lectures and practical exercises:

- Theoretical lessons will be given in order to obtain a broad overview of each topic.
- Practical lessons will consist in exercises in a programming language and using several CG applications.

IV. ASSESSMENT

- Practical exercises about rendering and animation (40%).
- A collaborative project (30%).
- A written examination (30%).

V. SELECTED BIBLIOGRAPHY

- "Advanced Animation and Rendering Techniques“, A. Watt, M. Watt, ACM Press. Addison-Wesley Publishing Company, 1993.
- "Computer Graphics with OpenGL “, D. Hearn, M.P. Baker., Prentice Hall, 2003.
- “Real-time rendering”, T. Akenine-Möller, E. Haines. Ed. AK Peters, 2002.
- “The art of 3D computer animation and effects”. I.V. Kerlow. Ed. Wiley, 2004.
- “An introduction to Ray Tracing”. Ed. Andrew S. Glassner, 2002.
- “Radiosity and Global Illumination”. F. X. Sillion, C. Puech. Ed. Morgan Kauffman, 1994.
- “3D Animation: From Models to Movies”. A. Watkins. Ed. Charles River Media, 2001.