Masters of Science in Software & Information Systems

To be developed and delivered in conjunction with Regis University, School for Professional Studies

Graphics Programming

December, 2005
# Table of Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Details</td>
<td>3</td>
</tr>
<tr>
<td>Course Outline</td>
<td>5</td>
</tr>
<tr>
<td>Workshop 1</td>
<td>6</td>
</tr>
<tr>
<td>Workshop 2</td>
<td>8</td>
</tr>
<tr>
<td>Workshop 3</td>
<td>10</td>
</tr>
<tr>
<td>Workshop 4</td>
<td>12</td>
</tr>
<tr>
<td>Workshop 5</td>
<td>14</td>
</tr>
<tr>
<td>Workshop 6</td>
<td>16</td>
</tr>
<tr>
<td>Workshop 7</td>
<td>18</td>
</tr>
<tr>
<td>Workshop 8</td>
<td>20</td>
</tr>
<tr>
<td>Appendices</td>
<td>21</td>
</tr>
</tbody>
</table>
Section 2 Module Details

Programme Title: M.Sc. in Software and Information Systems

Module Name: Graphics Programming

Course Description:
This module is intended for students who have experience in programming using C or C++, but who do not have any experience in real-time graphics programming. The module will provide students with a solid foundation in the key theoretical concepts of modern graphics programming, and an appreciation for the various compromises and trade-offs that are inherent in high-performance real-time graphics applications. As well as this emphasis on the theoretical basis for modern graphics applications, hands-on experience will be gained in two quite different graphics programming environments. This will provide practical examples of the theories covered by the module, as well as providing useful skills for the student. The two environments are: (i) a high-level descriptive graphics environment (Virtual Reality Modelling Language – VRML) and (ii) a general-purpose low level procedural graphics library (Open Graphics Library – OpenGL).

Prerequisites:
Successful completion of a C or C++ programming module.

Course Objectives:
At the end of the module, students are expected to be able to competently:

1. Discuss the evolution of computer graphics software and hardware systems
2. Assess and implement 3rd party 2D and 3D graphics libraries for C/C++
3. Use 3D coordinate systems, translations and transformations in order to describe complex 3D models
4. Develop complex interactive 3D graphics models and key-frame animations using VRML
5. Implement high-performance interactive 2D/3D graphics applications of intermediate complexity using OpenGL
6. Make appropriate use of the various high- and low-level techniques for 3D geometry description in VRML and OpenGL
7. Explain the trade-offs inherent in real-time graphics rendering, and the techniques that are used to simultaneously maximise both rendering performance and graphical realism.

**Required Texts:**

**Course Emphasis on Fundamental Theories and Applied Skills:**
In this module, students will receive a solid theoretical introduction to the art and science of computer graphics programming. The emphasis is on both the fundamental theories underpinning modern graphics programming and rendering, and also on the practicalities of working with high- and low-level graphics programming libraries. Upon successful completion, students will be capable of developing interactive graphics models and animations of reasonable size and complexity. They will also understand the underpinning theories and therefore be capable of moving towards an increased level of complexity.

**Course Assignments:**
Student assessment will take the form of:
- Weekly discussions, reports and/or practical assignments
- A Final Exam consisting of graphics development tasks and theoretical questions

**Grading:**

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Weight</th>
<th>Workshop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participation</td>
<td>20%</td>
<td>1-7</td>
</tr>
<tr>
<td>Assignments</td>
<td>45%</td>
<td>1-7</td>
</tr>
<tr>
<td>Final Exam</td>
<td>35%</td>
<td>8</td>
</tr>
<tr>
<td>Total</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>
## Course Outline

<table>
<thead>
<tr>
<th>Workshop</th>
<th>Text Reading</th>
<th>Topics</th>
<th>Assessments to be completed</th>
</tr>
</thead>
</table>
| 1: 2D Graphics Theory | | • Digital images and file formats  
• 2D computer graphics - theoretical foundations  
• Overview of modern 2D graphics APIs  
• Hardware accelerated graphics adapters | • Class discussion on 2D graphics applications  
• Research into popular 2D graphics libraries |
| 2: 2D Graphics Programming | OpenGL Book:  
• 1.1 – 1.6  
• 2.1 – 2.17  
• 3.1 – 3.7 | • Windows programming and input/output  
• The Windows GDI graphics library  
• OpenGL introduction and 2D graphics | • Development of an interactive 2D OpenGL program |
| 3: 3D Coordinate Systems, Projections and Geometrical Primitives | OpenGL Book:  
• 4.1 – 4.4, 4.8 | • 3D Coordinate systems  
• Planar projections  
• VRML introduction and graphical primitives  
• OpenGL cameras, objects, projections | • Research report on types of planar projection |
| 4: 3D Translation, Rotation, and Scaling | OpenGL Book:  
• 5.1 – 5.9 | 3D Translation, rotation, and scaling:  
• theoretically  
• in VRML  
• in OpenGL | • Development of an interactive 3D OpenGL program  
• Development of a VRML model using primitives |
| 5: Complex 3D Geometry | OpenGL Book:  
• 4.5 – 4.7 | • Hidden line/surface removal algorithms  
• Rendering efficiency and spatial partitioning  
• Complex geometrical modelling techniques in VRML  
• Hidden surface removal in OpenGL  
• Arbitrary facet modelling and stock objects in OpenGL | • Research and discussion on hidden line/surface removal algorithms  
• Development of a VRML model using extrusion |
| 6: Animation and Interactivity, Shading, Lighting and Materials | OpenGL Book:  
• 6.1 – 6.9 | • Events, animation, and interactivity in VRML  
• Surface shading algorithms  
• Lights and Materials in VRML and OpenGL | • Research report on Binary Space Partitioning  
• Development of a VRML animation |
| 7: Textures | OpenGL Book:  
• 7.1 – 7.4  
• 8.1 – 8.6, 8.8 | • Texture/bump/normal mapping  
• Pixel/vertex shaders  
• Texture mapping in VRML  
• Bitmaps and texture mapping in OpenGL  
• Backgrounds, billboards and fog | • Research report and discussion on either Games Engines or 3D Modelling Software  
• Development of a textured animation using either VRML or OpenGL |
| 8: Module Review and Final Exam | | • Module Review | • Final exam |
Section 3  Workshop Syllabus

Workshop One (2D Graphics Theory):

- Readings to be completed before Workshop One:
  - This Course Handbook

- List of topics to be covered:
  - Digital images and file formats
  - 2D computer graphics - theoretical foundations
    - Translation, rotation, scaling, and the use of matrices
    - Viewports and windows
    - Point-inside-polygon algorithms
    - Antialiasing
  - Overview of modern 2D graphics APIs
    - Definitions
    - Tools
  - Hardware accelerated graphics adapters

- Course Objectives for Workshop One
Upon completion of this workshop, students are expected to be able to:
  - Discuss the storage of compressed/uncompressed digital images
  - Identify the common attributes in modern digital image file formats
  - Define the terms ‘raster’ and ‘vector’ in the context of 2-dimensional (2D) computer graphics
  - Perform translation, rotation and scaling operations on 2D co-ordinates using matrices
  - Describe the 2D graphics concepts of viewports, windows, and line clipping
  - Discuss point-inside polygon algorithms, and identify the appropriateness of each to different circumstances
  - Explain the raster concept of antialiasing and discuss the algorithms used
  - Identify and explain the various functions that are provided by modern 2D graphics application programming interfaces (APIs)
  - Explain, from a theoretical perspective, the advances made by modern hardware accelerated graphics adapters

- Suggested Activities/Progress Checks/Review Questions:
  - The facilitator and students introduce themselves
  - The facilitator explains course objectives, grading criteria, assignments and administrative matters
  - The facilitator reviews the concepts of the reading material
- The student reads this workshop’s presentation slides
- The class discusses the historical and modern applications of 2D computer graphics
- The facilitator reviews the workshop’s assignment questions
- Students work on the workshop’s assignments
- The class discusses the strengths and weaknesses of some popular graphics programs of which they have some knowledge

- Assignment(s) for Workshop One:
  - Participate in a class discussion on the historical and modern applications of 2D computer graphics
  - Identify two popular 2D graphics APIs, and review the functions that they provide. Discuss the relative strengths and weaknesses of the two APIs that you have chosen. Examples: Java2D; DirectDraw; Windows GDI.
Workshop Two (2D Graphics Programming)

- Readings/activities to be completed before Workshop Two:
  - The assignments, exercises and discussions of Workshop One
  - **OpenGL: A Primer:**
    - Sections 1.1 – 1.6
    - Sections 2.1 – 2.17
    - Sections 3.1 – 3.7

- List of topics to be covered:
  - The essentials of Windows programming
  - Writing simple 2D graphics applications using the Windows Graphics Device Interface (GDI) API
  - Compiling and executing OpenGL projects using Visual C++
  - The OpenGL graphics API:
    - Overview
    - Event loops & callbacks
  - The OpenGL Utility Toolkit (GLUT)
    - 2D graphics in OpenGL
      - Primitive shapes
      - Colours
      - Co-ordinate systems & transformations
      - Text
    - Interaction/animation
      - Animation callbacks
      - Double buffering
  - Keyboard and mouse input using GLUT

- Course Objectives for Workshop Two
  Upon completion of this workshop, students are expected to be able to:
  - Write simple graphical programs running in Windows, using C++ and the Windows GDI.
  - Set up Visual C++ projects that use 3rd party libraries
  - Discuss the main features of the OpenGL API and the GLUT toolkit
  - Explain the flow of control differences between procedural and event-driven programming
  - Define the terms ‘event loop’ and ‘callback function’
  - Write simple, 2-dimensional graphical programs using OpenGL and GLUT, that include primitive shapes, colours, text, and animation, and that process input from the keyboard and mouse.
  - Discuss the need for, and implementations of, the graphics rendering approach of ‘multiple-buffering’
Suggested Activities/Progress Checks/Review Questions:
- The facilitator reviews the concepts of the week’s reading material
- The student reads the workshop’s presentation slides
- The student reads the workshop’s designated book chapters/sections
- The facilitator reviews the workshop’s assignment questions
- Students work on the workshop’s assignments
- The class discusses the process of setting up OpenGL with their C++ compiler and any difficulties encountered in getting simple OpenGL programs compiled and executed

Self-Review Exercises for Workshop Two:
- Set up your C++ compiler with the OpenGL and GLUT libraries; compile and run program 2.1 from the OpenGL textbook.

Assignment(s) for Workshop Two:
- Write an OpenGL program that receives keyboard and/or mouse input to control simple 2D shapes. For example, the shapes could follow the mouse around the screen, or the latest numerical key to be pressed could control the number of shapes being displayed. You can define the exact parameters that are controlled, the main requirement being to illustrate that you can set up a user-controlled 2D animation using OpenGL/GLUT.
Workshop Three (3D Coordinate Systems, Projections and Geometrical Primitives)

- Readings/activities to be completed before Workshop Three:
  - The assignments, exercises and discussions of Workshop Two
  - OpenGL: A Primer:
    - Sections 4.1 – 4.4, 4.8

- List of topics to be covered:
  - 3D Coordinate Systems
  - Planar Projections
  - Virtual Reality Modelling Language (VRML):
    - Introduction & terminology
    - Primitive geometrical shapes & groups
    - 3D text and fonts
  - OpenGL:
    - Cameras & objects
    - Projections

- Course Objectives for Workshop Three
  Upon completion of this workshop, students are expected to be able to:
  - Define the term “3D coordinate system”
  - Define the term “planar projection” and differentiate between the various types of planar projection that are used
  - Discuss the relative merits of procedural graphics APIs, such as OpenGL, and descriptive graphics languages, such as VRML
  - Define the components of a VRML .wrl file
  - Write simple VRML files that use nodes and fields, and that display as expected in VRML viewing programs/browser plug-ins
  - Construct VRML primitive shapes with customised appearances and materials
  - Construct VRML text using the various font display options
  - Write simple OpenGL programs that display wireframe 3D objects
  - Position the camera in OpenGL programs
  - Write OpenGL programs that render using perspective projection

- Suggested Activities/Progress Checks/Review Questions:
  - The student reads the workshop’s presentation slides
  - The student reads the workshop’s designated book chapters/sections
  - The facilitator leads a discussion of the workshop’s text and presentation slides
The class discusses planar projection operations and their uses in 3D graphics applications
The facilitator reviews the workshop’s assignment questions
The class discusses and works on the self-review questions
Students work on the workshop’s assignments

Self-Review Exercises for Workshop Three:
Obtain and install a VRML viewing program or Browser plug-in. Type in/copy-and-paste some of the simple VRML examples from the presentation slides.

Assignment(s) for Workshop Three:
Review the various types of planar projection used in 3D graphics applications, and identify the appropriateness of each to typical 3D graphics applications such as games and engineering design (CAD)
VRML is often seen as an “idealised” standard for specifying 3D models, yet it is lacking in implementation details. Research and discuss the VRML consortium’s process: how were the details agreed upon, what was the process, and what are the implications of such an “implementation agnostic” approach?
Workshop Four (3D Translation, Rotation and Scaling)

➢ Readings/activities to be completed before Workshop Four:
  ➢ The assignments, exercises and discussions of Workshop Three
  ➢ OpenGL: A Primer:
    • Sections 5.1 – 5.9

➢ List of topics to be covered:
  ➢ Theoretical foundations of 3-Dimensional (3D) computer graphics:
    • Translation, rotation, and scaling using matrices
  ➢ VRML:
    • Translation, rotation, and scaling using the Transform node
  ➢ OpenGL:
    • Translation, rotation, and scaling using matrix operations

➢ Course Objectives for Workshop Four
Upon completion of this workshop, students are expected to be able to:
  ➢ Calculate translation, rotation and scaling operations on 3D co-
    ordinates using matrices.
  ➢ Define rotations around arbitrary 3D axes.
  ➢ Define the term ‘nested coordinate system’ and apply VRML
    translation, rotation and scaling nodes on multiple-level nested
    systems.
  ➢ Use matrices in OpenGL to apply translation, rotation and scaling
    on rendered 3D objects.
  ➢ Use nested coordinate systems in OpenGL in order to manage
    complexity while defining 3D animations.

➢ Suggested Activities/Progress Checks/Review Questions:
  ➢ The student reads the workshop’s presentation slides
  ➢ The student reads the workshop’s designated book
    chapters/sections
  ➢ The facilitator leads a discussion of the workshop’s text and
    presentation slides
  ➢ The facilitator reviews the workshop’s assignment questions
  ➢ The facilitator explains expectations and grading criteria for the
    VRML assignments that will be worked upon over the next several
    weeks
  ➢ The class discuss and work on the self-review questions
  ➢ Students work on the workshop’s assignments
  ➢ The members of the class post their VRML models on the forum,
    and discuss the sequence of activities that they went through in
    developing them.
Assignment(s) for Workshop Four:

- Using your 2D OpenGL program developed during workshop 2 as a starting point, create a program that displays GLUT primitive shapes and allows the user to interact with them using the mouse and keyboard.
- Develop a VRML model of a suitable everyday object - one that can reasonably be defined as a set of translated and rotated primitive shapes, such as a table or a chair.
Workshop Five (Complex 3D Geometry):

➢ Readings/activities to be completed before Workshop Five:
   ➢ The assignments, exercises and discussions of Workshop Four
   ➢ OpenGL: A Primer:
     • Sections 4.5 – 4.7

➢ List of topics to be covered:
   ➢ Hidden line/surface removal algorithms
   ➢ Rendering efficiency & spatial partitioning
   ➢ VRML:
     • Complex geometry: points, lines, faces, elevation grids, extrusion
   ➢ OpenGL:
     • Vertex arrays
     • Hidden surface removal
     • Quadrics and GLUT Objects

➢ Course Objectives for Workshop Five:
Upon completion of this workshop, students are expected to be able to:
   ➢ Discuss theoretically the various hidden line/surface removal algorithms, in terms of their effectiveness and efficiency
   ➢ Construct arbitrary 3D points, lines and faces using VRML
   ➢ Create 3D surfaces in VRML using elevation grids
   ➢ Define the components of a 3D extrusion operation
   ➢ Define complex shapes in VRML using extrusion
   ➢ Recognise the types of 3D shape that suit modelling in VRML using primitives, extrusions, elevation grids, and arbitrary face sets
   ➢ Write OpenGL programs that build arbitrary 3D shapes using vertices and vertex arrays
   ➢ Implement hidden surface removal in OpenGL
   ➢ Make use of Quadric primitives and stock GLUT objects in OpenGL programs

➢ Suggested Activities/Progress Checks/Review Questions:
   ➢ The student reads the workshop’s presentation slides
   ➢ The student reads the workshop’s designated book chapters/sections
   ➢ The facilitator leads a discussion of the workshop’s text and presentation slides
   ➢ The facilitator reviews the workshop’s assignment questions
   ➢ The class discuss and work on the self-review questions
   ➢ The class conducts the discussion assignment
   ➢ Students undertake the workshop’s development assignment
The members of the class post their VRML models on the forum, and discuss the challenges that they posed and the learnings achieved.

Assignment(s) for Workshop Five:
- Research, and discuss with the class, the various hidden line/surface removal algorithms, in terms of their effectiveness and efficiency. Identify the requirements of different types of graphics applications when assessing these algorithms.
- Develop a VRML model of an everyday object using extrusion. Submit your code along with diagrams indicating the measurements taken of your object and its cross-section and spine.
Workshop Six (Animation and Interactivity, Shading, Lighting and Materials):

- Readings/activities to be completed before Workshop Six:
  - The assignments, exercises and discussions of Workshop Five
  - OpenGL: A Primer:
    - Sections 6.1 – 6.9

- List of topics to be covered:
  - VRML animation & interactivity:
    - The VRML event model
    - Animation: timers, interpolators and wiring
    - Interactivity: sensors
  - VRML lighting and materials
  - Surface Shading Algorithms
  - OpenGL Lights & materials:
    - Light sources
    - Reflections
    - Materials
    - Working with normals
    - Transparency

- Course Objectives for Workshop Six:
  Upon completion of this workshop, students are expected to be able to:
  - Discuss the various algorithms that are used in real-time 3D graphics applications for efficiently culling polygons that do not need to be rendered.
  - Explain the situations in which these various algorithms are most appropriate.
  - Explain the concept of key-frame animation, and apply it using VRML
  - Implement animation of various scalar and vector fields in VRML, using the appropriate interpolator nodes
  - Write interactive VRML worlds that respond to actions from the mouse, and that respond to proximity of the user’s camera
  - Discuss the evolution of shading and rendering algorithms for real-time and non-real-time graphics applications
  - Construct VRML worlds that use the realism techniques of lighting and materials. Make appropriate use of these techniques in order to simulate different types of surface materials and lighting situations.
  - Appreciate the reasons for the incomplete nature of the lighting model in languages such as VRML, and discuss the various techniques that are used to circumvent this
Explain the following terms as they apply to 3D graphics: surface normal, transparency, alpha transparency
Add lights to OpenGL applications, and control the shading algorithm used during the rendering process

Suggested Activities/Progress Checks/Review Questions:
The student reads the workshop’s presentation slides
The student reads the workshop’s designated book chapters/sections
The facilitator leads a discussion of the workshop’s text and presentation slides
The facilitator reviews the workshop’s assignment questions
The class discuss and work on the report assignment
The class undertakes the workshop’s development assignment
The members of the class post their VRML models on the forum, discuss the sequence of activities that they went through in developing their VRML model, and reflect on how (or if) their approach has changed in the last 3 weeks.
The class undertakes a discussion of the question “do graphics programmers also need to be artists?”

Assignment(s) for Workshop Six:
Research and write a report on the algorithm called Binary Space Partitioning (BSP). The report should clearly explain how BSP trees are constructed, and should discuss the situations when the BSP approach is, and isn’t, appropriate.
Construct a VRML animation of an appropriate real-world object, with interactivity if you wish. Examples: a cupboard whose doors and drawers open/close when the user clicks them; an animated grandfather clock.
Workshop Seven (Textures):

- Readings/activities to be completed before Workshop Seven:
  - The assignments, exercises and discussions of Workshop Six
  - OpenGL: A Primer:
    - Sections 7.1 – 7.4
    - Sections 8.1 – 8.6, 8.8

- List of topics to be covered:
  - Texture/bump/normal mapping
  - Pixel/vertex shaders
  - VRML: Adding Realism: texture mapping
  - OpenGL Textures & texture mapping
    - Displaying bitmaps
    - Reading/writing pixels
    - Textures, texels, texture co-ordinates
    - Applying textures to surfaces
    - Mipmaps
  - VRML: Backgrounds, billboards, and fog

- Course Objectives for Workshop Seven:
  Upon completion of this workshop, students are expected to be able to:
  - Explain the need for, and discuss the technical details of, the various ‘mapping’ techniques that are used in realtime 3D graphics applications – texture mapping and mip mapping, bump mapping, normal mapping
  - Apply texture maps to VRML objects
  - Explain and discuss the terms: pixels, texels, texture coordinates, bitmap, bit block transfer, pixel shaders, vertex shaders
  - Write OpenGL programs that operate on the level of the individual pixel
  - Write OpenGL programs that implement texture mapping and mip mapping
  - Construct VRML worlds that use the rendering efficiency techniques of backgrounds and fog
  - Understand the trade-offs that are inherent in real-time 3D graphics applications, and discuss the numerous techniques that are used to circumvent these

- Suggested Activities/Progress Checks/Review Questions:
  - The student reads the workshop’s presentation slides
  - The student reads the workshop’s designated book chapters/sections
The facilitator leads a discussion of the workshop’s text and presentation slides
The facilitator reviews the workshop’s assignment questions
The class researches and discusses the Games Engine/3D Modelling Tools assignment
Students undertake the workshop’s development assignment

Assignment(s) for Workshop Seven:
With the recent growth in the computer games industry, many so-called “Games Engines” have become available. Research and discuss the features that are provided by one of the popular games engines – you should discuss the graphics features provided as well as other features, such as networking, that are needed by modern games. Examples: Quake, Unreal, Half Life, Torque, and Dark Basic.

Or
Software tools such as “3D Studio Max” and “Maya” are used extensively in the games, movie and multimedia industries for constructing 3D models – it would not be efficient to develop these models using direct scripting/coding as we have been doing. Research and discuss the features provided by a leading 3D modelling tool.

Obtain from the Internet appropriate texture bitmaps for planets from the Solar System, and construct an animation, using either OpenGL or VRML, which illustrates the motion of the planets around the sun. You may be able to implement a non-linear scale, or alternatively disregard real-world scale altogether.
Workshop Eight (Final Exam):

- Readings/activities to be completed before Workshop Eight:
  - The assignments, exercises and discussions of Workshop Seven

- List of topics to be covered:
  - Review of workshops 1 - 7

- Suggested Activities/Progress Checks/Review Questions:
  - Conclude all course lecture topics and conduct a final Q&A session
  - Complete the final exam
  - Students will complete Course Evaluation Forms

- Self-Review Exercises for Workshop Eight:
  - None

- Assignment(s) for Workshop Eight:
  - Final Exam
Section 4  Appendices

(a) Grading Criteria for Written Submissions
A standard template will be devised for this section, however if writers are using a grading template for similar courses, please include here for discussion with Faculty group.

(b) Evaluation of Oral/Group Presentations (if applicable)
A standard template will be devised for this section, however if writers are using a grading template for similar courses, please include here for discussion with Faculty group.

(c) References

(d) Information Sources
http://www.web3d.org/
http://www.octaga.com/
http://www.opengl.org/
http://www.wiley.com/legacy/compbooks/vrml2sbk/cover/cover.htm

(e) Guidelines for software piracy, plagiarism, using sources in academic works
Standard response will be provided in each module handbook.