Module 33 Computer Graphics

Module title	Computer Graphics
Module NFQ level (only if an NFQ level can be demonstrated)	8
Module number/reference	BSCH-CG
Parent programme(s)	Bachelor of Science (Honours) in Computing Science
Stage of parent programme	Award stage
Semester (semester1/semester2 if applicable)	Semester 2
Module credit units (FET/HET/ECTS)	ECTS
Module credit number of units	5
List the teaching and learning modes	Direct, Blended
Entry requirements (statement of knowledge, skill and competence)	Learners must have achieved programme entry requirements.
Pre-requisite module titles	BSCH-FC, BSCH-LA, BSCH-NO
Co-requisite module titles	None
Is this a capstone module? (Yes or No)	No
Specification of the qualifications (academic, pedagogical and professional/occupational) and experience required of staff (staff includes workplace personnel who are responsible for learners such as apprentices, trainees and learners in clinical placements)	Qualified to as least a Bachelor of Science (Honours) level in Computer Science or equivalent and with a Certificate in Training and Education (30 ECTS at level 9 on the NFQ) or equivalent.
Maximum number of learners per centre (or instance of the module)	60
Duration of the module	One academic semester, 12 weeks teaching
Average (over the duration of the module) of the contact hours per week	3
Module-specific physical resources and support required per centre (or instance of the module)	One class room with capacity for 60 learners along with one computer lab with capacity for 25 learners for each group of 25 learners

Analysis of required learning effort		
	Minimum ratio teacher / learner	Hours
Effort while in contact with staff		
Classroom and demonstrations	1:60	18
Monitoring and small-group teaching	1:25	18
Other (specify)		
Independent Learning		
Directed e-learning		
Independent Learning		50
Other hours (worksheets and assignments)		39
Work-based learning – learning effort		
Total Effort		125

Allocation of marks (within the module)					
	Continuous assessment	Supervised project	Proctored practical examination	Proctored written examination	Total
Percentage contribution	50%			50%	100%

Module aims and objectives

The aim of this module is to teach the theoretical and practical underpinnings of modern 3D computer graphics APIs and applications. Learners are introduced to the two main families of rendering (Rasterization and Ray Tracing). However, this module focusses mainly on Rasterization as this is what the majority of graphics hardware is built for. Learners study the mathematics behind taking a collection of polygonal 3D triangular meshes and transforming them into an image that is viewable on a 2D screen.

The objectives of this module are to give learners the skills to create applications and render images using a modern Rasterization API (e.g. Vulkan, DirectX 12) through the practical application of transform and lighting techniques, texturing and animation. We also explain to learners how to take these applications and rework them to take advantage of advanced 3D rendering hardware such as stereoscopic screens and VR headsets.

Minimum intended module learning outcomes

On successful completion of this module, the learner will be able to:

1. Explain the rationale and difference between the two main families of graphics rendering techniques: Rasterization and Ray-Tracing.
2. Explain the steps in the fixed function rendering pipeline, and the changes made by modern programmable rendering pipelines.

3. Write 3D applications using an appropriate graphics API.
4. Mathematically describe object and viewing transformations.
5. Explain and demonstrate Transform and Lighting of lines and polygons using both Gouraud Shading and Phong Shading.
6. Explain and demonstrate basic 2D texture mapping onto a 3D surface
7. Explain and demonstrate basic animation using a matrix stack.
8. Describe how 3D applications can be modified/restructured to take advantage of stereoscopic screens and VR headsets.

Rationale for inclusion of the module in the programme and its contribution to the overall MIPLOs

The module enables learners to interact with and take advantage of GPU hardware available to them to generate 3D images. This can be used for game development or visualisation of data. Also by learning to interact with GPU based hardware this primes the learner to go on and learn about General Purpose GPU programming which can be used to heavily accelerate data parallel problems.

Appendix 1 of the programme document maps MIPLOs to the modules through which they are delivered.

Information provided to learners about the module

Learners receive a programme handbook to include module descriptor, module learning outcomes (MIMLO), class plan, assignment briefs, assessment strategy, and reading materials.

Module content, organisation and structure

Overview and introduction:

- Introduction to computer graphics
- Differences between 2D and 3D rendering
- Introduction to GPU hardware and difference in architecture and computation model.

Raytracing and Rasterization:

- Raytracing: aims and objectives
- Rasterization: aims and objectives
- Comparison of approaches
- Use cases of both approaches

Fixed function pipeline and modern programmable pipelines:

- Layout of the fixed function pipeline and purpose of each stage
- Modifications made by modern programming pipelines
- Changes in the 3D programming model due to pipeline changes
- How pipeline maps to hardware

Object and Viewing transformations:

- Coordinate systems: object, model, view, clip, NDCS, DCS
- How the coordinate systems transform a 3D scene into a 2D image
- Translate, scale, rotate operations
- Orthographic and Perspective projections

Transform and Lighting:

- The physics of light: specular, diffuse, emission and ambient.
- How light is approximated by the shading models
- Illumination and shading
- Flat, Gourad, and Phong shading models
- Barycentric coordinates

Texturing:

- Texture mapping as an optimisation to reduce geometric detail
- Basic principles of texture mapping.
- Mapping 2D textures to 3D polygonal meshes
- 2D textures and texture warping
- Mipmaps
- Cube mapping
- 3D textures

Animation:

- Introduction to animation
- The matrix stack and its relationship with animation
- Skeletal animation with the matrix stack
- Effect of push, pop and transform operations.

Stereoscopic vision and advanced rendering hardware:

- Differences between 3D and 2D displays
- Stereoscopic screens (passive and active)
- VR displays
- Changes required to rendering to take advantage of such displays

Module teaching and learning (including formative assessment) strategy

The module is taught as a combination of Lecture and Lab sessions. The lecture sessions discuss and explain to learners the principles and challenges involved in building 3D applications. They are introduced to both main rendering families before focusing heavily on Rasterization. Learners are brought through the mathematics and concepts of: Coordinate Systems, Transform and Lighting, Texturing, and Animation necessary to generate such applications. In the lab sessions learners get the opportunity to interact and develop computer graphics applications through an appropriate API to develop these skills.

Assessment is split into two. In terms of continuous assessment there are three major assignments that first test the learner's ability to make a working computer graphics application with Transform and Lighting before moving onto more challenging applications that involve texturing and animation. Finally, there is an end of semester exam that tests the learners understanding of the theoretical material.

Timetabling, learner effort and credit

The module is timetabled as one 1.5-hour lecture and one 1.5-hour lab per week.

The number of 5 ECTS credits assigned to this module is our assessment of the amount of learner effort required. Continuous assessment spreads the learner effort to focus on the major components of computer graphics development.

There are 36 contact hours made up of 12 lectures delivered over 12 weeks with classes taking place in a classroom. There are also 12 lab sessions delivered over 12 weeks taking place in a fully equipped computer lab. The learner will need 50 hours of independent effort to further develop the skills and knowledge gained through the contact hours. An additional 39 hours are set aside for learners to work on worksheets and assignments that must be completed for the module.

The team believes that 125 hours of learner effort are required by learners to achieve the MIMLOs and justify the award of 5 ECTS credits at this stage of the programme.

Work-based learning and practice-placement

There is no work based learning or practice placement involved in the module.

E-learning
The college VLE is used to disseminate notes, advice, and online resources to support the learners. The learners are also given access to Lynda.com as a resource for reference.

Module physical resource requirements
Requirements are for a classroom for 60 learners equipped with a projector, and a 25 seater computer lab for practical sessions with access to a Computer Graphics SDK and IDE (for example Vulkan or DirectX 12 but this may change should better techonologies arise).

Reading lists and other information resources
Recommended Reading
Hugues, J. F. (2014) *Computer Graphics: Principles and Practice*. Upper Saddle River: Addison-Wesley.

Sellers, G. and Kessenich, J. (2016) *Vulkan Programming Guide: The Official Guide to Learning Vulkan*. Boston: Addison-Wesley.

Secondary Reading:
Sellers, G., Jr, R. S. W. and Haemel, N. (2015) *OpenGL Superbible: Comprehensive Tutorial and Reference*. New York: Addison-Wesley.

Lengyel, E. (2012) *Mathematics for 3D Game Programming and Computer Graphic*. Boston: Cengage Learning

Specifications for module staffing requirements
For each instance of the module, one lecturer qualified to at least Bachelor of Science (Honours) in Computer Science or equivalent, and with a Certificate in Training and Education (30 ECTS at level 9 on the NFQ) or equivalent.. Industry experience would be a benefit but is not a requirement.

Learners also benefit from the support of the programme director, programme administrator, learner representative and the Student Union and Counselling Service.

Module Assessment Strategy
The assignments constitute the overall grade achieved, and are based on each individual learner’s work. The continuous assessments provide for ongoing feedback to the learner and relates to the module curriculum.

No.	Description	MIMLOs	Weighting
1	Practical Assignment that will get the learner to make a basic 3D application that will place a group of objects in the view and render them moving (basic translation and rotation) along with transform and lighting techniques.	3,5	12.5%
2	Practical Assignment that will build on the first assignment and will get learners to apply 2D textures to 3D objects (e.g. applying a map of earth to a sphere)	3,5,6	17.5%
3	Practical Assignment that will get the learner to build a complex object for animation (e.g. a man out of cylinders and spheres) and animate basic movement through the use of a matrix stack	3,5,6,7	20%
4	Written exam that tests the theoretical aspects of the module	1,2,4-8	50%

All repeat work is capped at 40%.

Sample assessment materials

Note: All assignment briefs are subject to change in order to maintain current content.

Assignment 01: Building a basic 3D application and graphics rendering pipeline

Introduction:

In this assignment you will be tasked with building a basic 3D rendering pipeline to display and manipulate objects. The provided Vulkan wrapper code will handle the majority of setup work needed for the GPU. You will need to construct and enable the appropriate parts of the pipeline you need with the help of this code. You will at a minimum need access to a vertex and fragment shader, depth buffering, vertex upload, command buffer generation, and to setup a perspective viewing projection.

Your task is to setup the render pipeline to render a number of objects (cubes, and spheres), setup a light source, setup basic shaders to do phong lighting with gouraud shading, apply materials to the objects, and to rotate the objects by using object transformation and the matrix stack.

Notes:

This assignment is designed in such a way that you will be required to finish the previous steps in order to progress onto the next one. The reason for this is that it takes a significant amount of code (even with the aid of the helper classes) to get a pipeline setup and rendering anything.

Remember you will only be able to verify that this is working correctly by visual observation you will not be able to write software tests to check that the generated images are correct.

Those of you running an Apple machine of any description please remember to Bootcamp your machine and boot into Windows as Mac OSX does not support the Vulkan API.

Tasks:

01) Generate a rendering pipeline that has access to a vertex shader, fragment shader, and depth buffering, set the screen clear colour to be black.
02) Enable a perspective projection on your pipeline. Also write a basic vertex shader that will transform vertices with the model view matrix. You should write a basic fragment shader that will always output the colour white for any fragment.
03) At (0, 0, -5) draw a square directly facing the camera to verify that steps 01, and 02 are correct. You should see a white square on a black background.
04) After verifying step 03 is working correctly construct the geometry for a cube and a sphere (mathematics provided in appendix) and store in GPU memory. You should render two spheres in the middle of your scene and two cubes one either side of the spheres. Give each object a basic colour (red, green, blue, yellow) to distinguish them.

Modify your vertex and fragment shaders to output the colour associated with each vertex.
05) generate and position a light source at (1000, 0 , 1000) such that it is above and behind the camera. It should have full light for all components (specular, diffuse, and ambient), modify your vertex shader to do the lighting calculations for each vertex. If you have not passed normal data as part of your geometry do so now. Modify your vertex data to include materials for full while specular, full diffuse of the basic object colour and 20% ambient white light.
06) Add code that will automatically rotate your cubes and spheres in response to the current framerate.

Assignment 02: Building a graphics rendering pipeline that is capable of using textures.

Introduction:

In this assignment you will be tasked with building an improved version of the 3D rendering pipeline that you constructed in the previous assignment. In this assignment you will be tasked with applying textures to a sphere and cube to imitate the look of more complex geometry. You will be using the Vulkan wrapper code that you received for the previous assignment to assist you.

Your task is to setup the render pipeline to render two objects with textures. One will be a cube with a texture applied to it to make it look like a companion cube (see the Portal games for reference) and the other will be a sphere that will have a texture of the map of earth applied to it. You will need to add additional caluclations into your fragment shader to mix the calculated lighting colours in with the information provided in the texture. You are required to use 16x anisotropic filtering for this.

Notes:

This assignment is designed in such a way that you will be required to finish the previous steps in order to progress onto the next one. The reason for this is that it takes a significant amount of code (even with the aid of the helper classes) to get a pipeline setup and rendering anything.

Remember you will only be able to verify that this is working correctly by visual observation you will not be able to write software tests to check that the generated images are correct.

Those of you running an Apple machine of any description please remember to Bootcamp your machine and boot into Windows as Mac OSX does not support the Vulkan API.

Tasks:

01) Generate a rendering pipeline that has access to a vertex shader, fragment shader, depth buffering, textures, and set the screen clear colour to black.
02) Enable a perspective projection on your pipeline. Take a copy of the vertex and fragment shaders provided (implementation of Assignment 01 shaders) and enable them on the pipeline. Add in a light source at (1000, 1000, 1000) will full white light for specular, diffuse and ambient.
03) position a cube and a sphere in your scene with full white materials for specular, and diffuse components, there should be a white 20% ambient component and no emission on the sphere, while the cube should also have the same ambient component but a 30% pink emission component added to it.
04) upload the provided companion cube and map of earth textures to the GPU.
05) apply the companion cube texture to all 6 faces of the cube.
06) apply the map of earth texture to the sphere using latitude and longitude corrdinates to map a 2D point on the texture to a 3D point on the surface (mathematics are in the appendix)
07) write code to rotate your cube and sphere in relation to the current framerate

Assignment 03: Building a graphics application capable of doing animation

Introduction:

In this assignment you will be tasked with building an application that will render a very basic man running in a circle. He will be composed of spheres and cylinders (code is provided for generating these objects). This assignment will show you how to use a matrix stack and object transformations to render, position, and orient complex objects with a skeleton like structure.

Your task is to first setup and render the geometry of the man by using the matrix stack, transformation matricies, and objects provided. Once this is done you will progressively add animation to the man to do a basic running motion. First you will make the entire structure move in a circle before adding animiation to the legs, and finally the arms. For the purposes of the assignment you are only required to animate movement on the shoulders and hips of the man. Knee, elbow, hand, and foot animation is not required.

Notes:

This assignment is designed in such a way that you will be required to finish the previous steps in order to progress onto the next one. The reason for this is that it takes a significant amount of code (even with the aid of the helper classes) to get a pipeline setup and rendering anything.

Remember you will only be able to verify that this is working correctly by visual observation you will not be able to write software tests to check that the generated images are correct.

Those of you running an Apple machine of any description please remember to Bootcamp your machine and boot into Windows as Mac OSX does not support the Vulkan API.

Tasks:

01) Take the provided pipeline, set it up and ensure it is working.
02) Using the matrix stack and the objects provided render the basic structure of the man, there should be cylinders to represent the main body, neck, upper arms, fore arms, thighs, and shins. Spheres should be used to represent the shoulders, elbows, hands, hips, knees, and feet. Cylinders should be done in a green material and spheres should be done in a red material.

To render the structure start with the main body and work your way out to each extremity using the matrix stack.

03) Animate the movement of the body around the centre point of the scene. The man should be constantly moving in a circle and in view the whole time. Do this relative to the frame rate.
04) Animate the movement of the legs from the hip. The leg should complete a full forward or backward movement in one quarter of a circle.
05) animate the movement of the arms from the shoulder. The arm should complete a full forward or backward movement in one quarter of a circle
06) add in controls to speed up or slow down the animation rate of the hips and arms based on key presses from the keyboard.

GRIFFITH COLLEGE DUBLIN

QUALITY AND QUALIFICATIONS IRELAND

EXAMINATION

COMPUTER GRAPHICS

Lecturer(s):
External Examiner(s):

Date: XXXXXXX **Time: XXXXXX**

THIS PAPER CONSISTS OF FIVE QUESTIONS
FOUR QUESTIONS TO BE ATTEMPTED
ALL QUESTIONS CARRY EQUAL MARKS

THE USE OF NON-PROGRAMMABLE CALCULATORS IS PERMITTED DURING THIS EXAMINATION

QUESTION 1

(a) List the main stages of a modern graphics pipeline in the correct sequence.

(7 marks)

(b) For each stage indicate the principal function of that stage.

(14 marks)

(c) In relation to the graphics pipeline, what is a shader?

(4 marks)

Total (25 marks)

QUESTION 2

(a) Give the generalised form of the homogenous 3 × 3 2D matrix to translate by (t_x, t_y)

(5 marks)

(b) Give the generalised form of the homogenous 3 × 3 2D matrix to rotate by an angle θ around the origin.

(5 marks)

(c) Explain what is a Scaling Transform in 3d space. Use diagrams to help.

(5 marks)

(d) Define cross product for any two vectors in 3d space

(10 marks)

Total (25 marks)

QUESTION 3

(a) Explain the Gouraud shading model.

(10 marks)

(b) Define the dot product of 2 vectors.

(10 marks)

(c) If two vectors are orthogonal, what is the value of their dot product?

(5 marks)

Total (25 marks)

QUESTION 4

(a) Give a brief description of :

 (i) Directional light source,

 (ii) Area light source,

(6 marks)

(b) What is the difference between local and global illumination?

(9 marks)

(c) Explain how the constant illumination model works.

(10 marks)

Total (25 marks)

QUESTION 5

(a) How does the ray tracing algorithm differ from the conventional rasterization approach, including diagrams with your answer?

(10 marks)

(b) Outline some of the of the most commonly used methods to speed up the raytracing process.

(10 marks)

(c) Explain the concept of MIP mapping

(5 marks)

Total (25 marks)