Higher National unit specification

General information

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

Unit code: HH3L 35

Superclass: CC

Publication date: November 2016

Source: Scottish Qualifications Authority

Version: 01

Unit purpose

The purpose of this unit is to develop the learner's knowledge and skills in writing mathematical algorithms using a high level language.

The unit is intended for learners who possess some knowledge of mathematics and coding, who wish to further develop these abilities in a software development context. It seeks to deliver the key mathematical knowledge and skills required to implement common mathematical techniques used within software development.

This is a **non-specialist** unit that can be used within a range of Computing awards. It is particularly well suited to learners who require to apply numerical methods in programming contexts, such as learners who wish to develop computer games or systems software.

Outcomes

On successful completion of the unit the learner will be able to:

1 Implement algorithms that perform algebraic operations.
2 Implement algorithms that perform geometric operations.
3 Implement algorithms that perform calculus operations.

Credit points and level

1 Higher National unit credit at SCQF level 8: (8 SCQF credit points at SCQF level 8)

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

Recommended entry to the unit

Access to this unit is at the discretion of the centre. However, it is recommended that learners should have completed the unit *Computer Programming: Applied Mathematics* (SCQF level 7) and a suitable programming unit, such as *Structured Programming for Games*. Alternatively, other units or experience in programming and mathematics may be suitable.

Core Skills

Opportunities to develop aspects of Core Skills are highlighted in the Support Notes for this unit specification.

There is no automatic certification of Core Skills or Core Skill components in this unit.

Context for delivery

If this unit is delivered as part of a Group Award, it is recommended that it should be taught and assessed within the subject area of the Group Award to which it contributes.

The Assessment Support Pack (ASP) for this unit provides assessment and marking guidelines that exemplify the national standard for achievement. It is a valid, reliable and practicable assessment. Centres wishing to develop their own assessments should refer to the ASP to ensure a comparable standard. A list of existing ASPs is available to download from SQA's website **(http://www.sqa.org.uk/sqa/46233.2769.html)**.

Equality and inclusion

This unit specification has been designed to ensure that there are no unnecessary barriers to learning or assessment. The individual needs of learners should be taken into account when planning learning experiences, selecting assessment methods or considering alternative evidence.

Further advice can be found on our website **www.sqa.org.uk/assessmentarrangements**.

Higher National unit specification: Statement of standards

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

Acceptable performance in this unit will be the satisfactory achievement of the standards set out in this part of the unit specification. All sections of the statement of standards are mandatory and cannot be altered without reference to SQA.

Where evidence for Outcomes is assessed on a sample basis, the whole of the content listed in the Knowledge and/or Skills section must be taught and available for assessment. Learners should not know in advance the items on which they will be assessed and different items should be sampled on each assessment occasion.

Outcome 1

Implement algorithms that perform algebraic operations.

Knowledge and/or Skills

- Polynomials
- Interpolation
- Linear algebra: 2D and 3D vectors
- Linear algebra: matrices
- Algebraic algorithm construction

Outcome 2

Implement algorithms that perform geometric operations.

Knowledge and/or Skills

- Co-ordinate systems
- 3D geometrical points and shapes
- Interactions between geometric objects
- Transformations
- Performing transformations using matrixes
- Normals
- Geometric algorithm construction

Outcome 3

Implement algorithms that perform calculus operations.

Knowledge and/or Skills

- Differentiation of standard functions
- Integration of standard functions
- Evaluation of definite integrals
- Numerical methods for Ordinary Differential Equations (ODEs)
- Calculus algorithm construction

Higher National unit specification: Statement of standards (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

Evidence Requirements for this unit

Candidates will need to provide evidence to demonstrate their Knowledge and/or Skills across all Outcomes. Evidence is normally required for all of the Knowledge and Skills in every Outcome.

The Evidence Requirements for this unit will take the form of evidence of **practical competence**. This evidence will comprise **one or more** computer programs that demonstrate the candidate's ability to write code that covers **all** of the defined Knowledge and/or Skills.

The evidence may consist of a large number of program segments that collectively demonstrate all of the Knowledge and/or Skills or a small number of larger programs that collectively demonstrate all of the Knowledge and/or Skills.

This will be a series of programming tasks, each which of will directly or indirectly apply Knowledge and Skills from one or more Outcomes. Where knowledge and skills are applied indirectly (for example, in using differentiation to derive a formula to use in interpolation) this should be evidenced by appropriate documentation, for example using either suitably detailed comments, written reports, or a combination of both.

The evidence for all practical requirements may be wholly or partly produced under controlled conditions. When evidence is produced in uncontrolled or loosely controlled conditions it must be authenticated. The Guide to Assessment provides further advice on methods of authentication. Evidence of authentication must be provided when any of the evidence is generated under loosely controlled conditions.

There are no time limitations on the production of evidence. The evidence may be produced at any time during the life of the unit. Candidates may use reference materials when undertaking practical assessment.

The Guidelines on Approaches to Assessment (see the Support Notes section of this specification) provides specific examples of instruments of assessment.

Higher National unit Support Notes

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

Unit Support Notes are offered as guidance and are not mandatory.

While the exact time allocated to this unit is at the discretion of the centre, the notional design length is 40 hours.

Guidance on the content and context for this unit

Unit purpose

This unit is intended to improve learners' understanding of how to implement mathematical techniques in a games programming context.

The programming environment should be a high level language, and the learner is not necessarily expected to write code that performs the mathematical operations themselves, but rather to apply existing mathematical operations in order to solve programming problems. For example, students should not be expected to write their own code to calculate a matrix determinant; but it would be reasonable to ask learners to make use of an existing 'invert' function (which uses the determinant), in order to perform 'picking' — determining which 3d object was clicked on in a 2d projection.

Most widely-used languages have appropriate function libraries available, such as C++, C#, Python, and Java. The unit is aimed at learners who already have an intermediate understanding of mathematics and programming. They should not need to study programming concepts such as control structures, data structures, or objects.

Unit Outcomes

The three Outcomes should cover commonly used techniques in the three areas, focusing on mathematical functions and their practical applications as far as is feasible. The following outlines suggested topics for delivery within those areas.

Algebraic operations

This Outcome covers a variety of common algebraic techniques employed in software development. The knowledge and skills learned here are partially intended to provide a basis for further programming techniques. As such, not all of this knowledge is directly applicable in a realistic scenario on their own, but instead form a fundamental part of many later solutions.

Higher National unit Support Notes (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

- Polynomials:
  - Linear functions ($y = mx + c$)
    - First equation of motion: $v_f = v_i + at$
    - Contextualised examples, eg:
      Hit Points in FFV at level 21: HP = 320 + 10 * Stamina
  - Quadratic functions ($y = ax^2 + bx + c$, $a \neq 0$)
    - Second equation of motion: $x_f = x_0 + v_it + 1/2at^2$
    - Contextualised examples, eg:
      XP required for first 10 levels in WoW: XP = 40x2 + 360x
  - Cubic functions ($y = ax^3 + bx^2 + cx + d$, $a \neq 0$)
    - Typical smoothing function: Cubic Hermite spline with unit interval (0-1) and zero slope at both boundaries x0 and x1.
      ie: $f(t) = -2t^3 + 3t^2$
    - Contextualised examples, eg:
      XP required for levels 11-27 in WoW: XP = -.4x3 + 40.4x2 + 396x
  - Nth-degree Polynomials ($y = a_n.x^n + a_{n-1}.x^{n-1} \ldots a_2.x^2 + a_1.x + a_0$)
- Interpolation:
  - Linear interpolation
  - Cosine interpolation
  - Cubic interpolation
  - Basic concepts of other types: Slerp, catmull-rom, Hermite, Bezier, etc
- Linear algebra: 2d and 3d vectors:
  - Using vectors in 3d space
  - Vector arithmetic:
    - Vector addition
    - Vector subtraction
    - Scalar multiply
    - Dot product
    - Cross product
  - Normalisation
- Linear algebra: matrices:
  - Matrix dimensions and special types:
    - column matrix
    - row matrix
    - identity matrix
    - zero matrix
  - Vectors as column matrices
  - Matrix operations
    - Matrix addition
    - Matrix subtraction
    - Scalar multiply
    - Matrix multiplication
    - Matrix inversion
    - Matrix transpose
  - Matrix determinants

Higher National unit Support Notes (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

Geometric operations

This Outcome expands on knowledge of 2d geometry into a 3d space, as well as introducing applications for matrices in the manipulation of shapes and vertexes in a game world, and applications of vectors in normal mapping for textures.

- Co-ordinate systems:
  - Right-handed
  - Left-handed
- 3d geometrical points and shapes:
  - Points and vertexes
  - Lines/rays/line segments
  - Spheres
  - Axis-aligned Bounding Boxes
  - Oriented Bounding Boxes
  - Polygons, planes, and shared vertexes
  - Other shapes: cylinder, capsule, frustum
- Interactions between geometric objects:
  - Intersection
  - Containment
  - Point-in-poly and point-on-surface
  - Separating Axis Theorem
  - Minkowski sums
- Transformations:
  - Translation
  - Rotation about a co-ordinate axis
  - Rotation about an arbitrary axis
  - Scale
  - Reflection
  - TRS Matrices aka. 'World matrix'
  - Projection
- Performing transformations using matrixes
- Normals:
  - Surface normals
  - Vertex normals
  - Normal maps
  - The fast inverse square root function

Higher National unit Support Notes (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

Calculus operations

This Outcome focuses on integration and differentiation of functions, and their application in game development, such as in physics engines.

- Differentiation of standard functions:
  - Power rule: Derivative of x^n
    - Special case: derivative of line x
  - Multiplication by constant: Derivative of ax^n
    - Special cases: derivative of constant c, derivative of ax
  - Derivative of trigonometric functions: asinx and acosx
- Integration of standard functions:
  - Power rule: Integral of x^n
    - Special case: integral of x
  - Multiplication by constant: Integral of ax^n
    - Special cases: integral of constant c, integral of ax
  - Integration of trigonometric functions: asinx and acosx
- Evaluation of definite integrals
- Numerical methods for Ordinary Differential Equations (ODEs):
  - Local error
  - Global error
  - Euler's method
  - Verlet
  - Runge-Kutta 4th order

Contextualisation

The problems presented to the learner to solve should be in the wider context of their studies, with abstract theory being minimised. For example, the cross product of a vector should be explained and used in a context such as computing surface normals in order to apply lighting effects to a 3d mesh.

Progression and Links

This unit provides underpinning knowledge for learners who wish to pursue higher education or a career involving mathematics in programming, such as a game development or systems programming environment.

It is suitable for learners who intend to progress onto courses which continue to use mathematics in programming, such as a university course in Computer Game Development, or other courses in the field of Computer Science.

The unit focuses on vocational skill in applying mathematical techniques in programming environments, and preparing learners for a career in software development where mathematics is often used. As such the theoretical aspects of mathematics such as formal proofs should be kept to a minimum when delivering this unit.

Higher National unit Support Notes (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

This unit currently has no special recognition by professional bodies, and there are no National Occupational Standards specific to this area at this time.

Guidance on approaches to delivery of this unit

It is suggested that the material be delivered in the order of the Outcomes and criteria. This is due to the fact that some areas of mathematics are interconnected, and learning in certain areas can be useful or even necessary before attempting later techniques.

However, that does not necessarily mean that the material need be assessed in that same order. For example, the cross product can easily be assessed later when calculating surface normals.

It is recommended that the unit be delivered using varied approaches to teaching and learning where possible. However due to the nature of the content and the wide range of material covered, opportunities for group activities may be limited and might only be feasible for formative work. Emphasis should be placed on practical tasks, particularly coding where possible.

The three Outcomes will likely share an approximately similar proportion of the learning and teaching time for this unit.

An example of one possible scheme for assessment of this unit is as follows, contextualised for delivery as part of a Computer Game Development Group Award:

The practical evidence could comprise a portfolio of projects containing code to solve a variety of programming problems using mathematical techniques. The portfolio would likely be based on code outlines provided by the tutor, which the learner would have to apply mathematical techniques to in order to achieve the stated goals.

- Use linear interpolation to move an object in a game world on a per-frame basis. (**Interpolation, polynomials**).
- Use a matrix to translate two 2d objects in order to perform pixel-perfect collision detection. (**Matrix algebra, vectors, transformations using matrixes**).
- Use matrix algebra to prepare 3d objects for rendering. (**Matrix algebra, 3d geometry, co-ordinate systems, transformations using matrixes**).
- Check for Intersection between two axis-aligned bounding boxes in 3d space (**3d geometry, Interaction between geometric objects**).
- For a given polygon surface, calculate a surface normal, and use it to apply a reflection transformation on an incoming ray of light to determine specular lighting. (**Normals, vectors, 3d geometry, matrix transformations**).
- Differentiate a cubic function and use the result to implement cubic interpolation for smoothing movement of game objects. (**Differentiation, polynomials, interpolation**).
- Implement physics for game objects by solving antiderivatives. (**Integration, polynomials**).

- Implement physics for game objects by using numerical methods such as Euler's method, Verlet, or Runge-Kutta 4th order integration. (**Numerical methods, polynomials**).
- A space vehicle is approaching cruising speed, and is easing off on the thrusters. Calculate the fuel usage of the vehicle, based on a formula showing the decrease in force applied over time. (**evaluation of definite integrals**).

Guidance on approaches to assessment of this unit

Evidence can be generated using different types of assessment. The following are suggestions only. There may be other methods that would be more suitable to candidates.

Centres are reminded that prior verification of centre-devised assessments would help to ensure that the national standard is being met. Where candidates experience a range of assessment methods, this helps them to develop different skills that should be transferable to work or further and higher education.

All of the Evidence Requirements must be met in a practical manner, but this does not necessarily mean that every Evidence Requirement must correspond to a specific function in the code, since some mathematical techniques may be used as the basis for others. Where evidence for a given requirement is not directly visible in the practical task, the assessor should ensure that suitable evidence is generated to meet that requirement, for example by requiring candidates to write documentation which explains the mathematical derivation of their functions.

It may well be possible to integrate assessment of certain criteria in this unit along with other Outcomes in this unit, or as part of other units. For example, if the candidates produce a solution to a programming problem as part of an object oriented programming unit, which includes numerical integration in it, the assessor may accept that solution as part of the folio of practical applications for this unit.

It is recommended that the practical tasks be undertaken in supervised conditions, to help ensure authenticity of candidates' submissions. If the centre allows candidates to work on programming problems unsupervised (eg at home, or in a student-led study area) then assessors should take additional steps to ensure the authenticity of the submission, such as asking the candidate to explain parts of the code verbally and how the code solves the problem.

Opportunities for e-assessment

E-assessment may be appropriate for some assessments in this unit. By e-assessment we mean assessment which is supported by Information and Communication Technology (ICT), such as e-testing or the use of e-portfolios or social software. Centres which wish to use e-assessment must ensure that the national standard is applied to all learner evidence and that conditions of assessment as specified in the Evidence Requirements are met, regardless of the mode of gathering evidence. The most up-to-date guidance on the use of e-assessment to support SQA's qualifications is available at **www.sqa.org.uk/e-assessment**.

Higher National unit Support Notes (cont)

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

Opportunities for developing Core and other essential skills

There is likely to be numerous opportunities to develop elements of the Core Skills of *Numeracy* in this unit, and possibly some elements of *Problem Solving*, although there is no automatic certification of Core Skills or Core Skills components.

History of changes to unit

Version	Description of change	Date

General information for learners

Unit title: Computer Programming: Applied Mathematics (SCQF level 8)

This section will help you decide whether this is the unit for you by explaining what the unit is about, what you should know or be able to do before you start, what you will need to do during the unit and opportunities for further learning and employment.

This is an intermediate unit using mathematical techniques and algorithms in programming. It is assumed that you have some understanding of programming and some pre-calculus level maths knowledge. It introduces concepts of mathematics as used in programming games and applications, and is suited to a number of computing courses, but may also be studied on its own by learners wishing to improve their programming and mathematics skills.

You will be presented with a variety of programming problems which you will be required to solve using the mathematical functions available. These solutions will form a portfolio of work relating to game development. You may be required to document some of the underlying mathematical calculations you performed in order to arrive at certain functions.

The solutions will cover areas such as:

- Forms of algebra common to software development: such as polynomial formulas, interpolation for smooth movement of objects, and the use of vectors and matrices for handling positions and movement of objects.
- Geometry of objects: such as 3d shapes, collision detection, transforming objects in world space, and calculating reflection of light from object surfaces.
- Fundamental applied calculus for programming: differentiation and integration, and numerical methods used to integrate physics equations.

While solving these problems, you will develop elements of Core Skills in the areas of *Numeracy* and *Problem Solving*.