Higher National Unit specification

General information for centres

Unit title: Games Development: Object Oriented Programming

Unit code: F86A 35

Unit purpose: This Unit is designed to enable candidates to develop a broad knowledge of the concepts, principles, and techniques of object oriented software development. Candidates will develop problem-solving and object oriented technical skills. Candidates will then be required to demonstrate their proficiency in these skills through the creation of object oriented software solutions to problems. This Unit focuses on object oriented programming concepts and creating graphical programs rather than console-based programs, enabling the candidates to create simple games.

On completion of the Unit the candidate should be able to:

1 Analyse a programming problem from a given brief and design an object oriented solution.
2 Investigate object oriented programming techniques and apply them to a design.
3 Implement a solution from an object oriented design using object oriented techniques.

Credit points and level: 3 HN credits at SCQF level 8: (24 SCQF credit points at SCQF level 8*)

**SCQF credit points are used to allocate credit to qualifications in the Scottish Credit and Qualifications Framework (SCQF). Each qualification in the Framework is allocated a number of SCQF credit points at an SCQF level. There are 12 SCQF levels, ranging from Access 1 to Doctorates.*

Recommended prior knowledge and skills: Access to this Unit will be at the discretion of the Centre. However, it is recommended that candidates should have achieved the Core Skill of *Problem Solving* at SCQF level 5 before taking this Unit. It is also strongly recommended that candidates are familiar with fundamental programming concepts at SCQF level 7. This may be demonstrated by possession of the HN Unit F8HC 34 *Structured Programming for Games.*

Alternatively, candidates may have considerable practical work experience and a portfolio of programs which demonstrate their competence with the techniques covered in the HN Unit F8HC 34 *Structured Programming for Games.*

Core Skills: There are opportunities to develop the Core Skill of *Problem Solving* at SCQF level 6 in this Unit, although there is no automatic certification of Core Skills or Core Skills components.

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.

General information for centres (cont)

Assessment: It is recommended that all Outcomes are integrated into one holistic assessment which takes the form of a project. This project should require the candidate to analyse a problem and design and implement an object oriented solution.

The assessment should be completed on an individual basis under open-book supervised conditions.

Assessors should ensure themselves of the authenticity of the candidate's evidence.

Higher National Unit specification: statement of standards

Unit title: Games Development: Object Oriented Programming

Unit code: F86A 35

The sections of the Unit stating the Outcomes, Knowledge and/or Skills, and Evidence Requirements are mandatory.

Please refer to *Knowledge and/or Skills for the Unit* and *Evidence Requirements for the Unit* after the Outcomes.

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. Candidates 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

Analyse a programming problem from a given brief and design an object oriented solution

Knowledge and/or Skills

- Analyse the problem from a given brief
- Identify the requirements from a given brief
- Develop a design solution for the specification using an object oriented approach
- Storyboarding

Evidence Requirements

Candidates will need to provide evidence to demonstrate their Knowledge and Skills by showing that they can analyse a programming problem from a brief and design an object oriented solution.

A candidate’s response can be judged to be satisfactory where the evidence produced shows the candidate is able to:

- produce a Use Case model.
- produce a class diagram which depicts an appropriate, object-oriented solution to the given problem and includes the functionality requested in the brief. The class diagram should use the correct UML notation and must include, at a minimum:
  - classes
  - attributes, including their access and data types
  - methods/behaviours, including their access and return types
  - relationships between classes, correctly indicating the type of relationship (association, aggregation, or inheritance) between classes
- produce a collection of storyboards which show the following:
  - the layout of each screen of the program.
  - the components to be used on each screen. All components should be accurately labelled with a name and type.
  - indication of fixed or relative positioning (co-ordinates or layout managers).
  - how user interaction affects what is shown on each screen, including any new screens/ message boxes which may appear due to user interaction.

Higher National Unit specification: statement of standards (cont)

Unit title: Games Development: Object Oriented Programming

Assessment Guidelines

See Outcome 3 Assessment Guidelines

Outcome 2

Investigate object oriented programming techniques and apply them to a design

Knowledge and/or Skills

- Object oriented concepts and terms
- Object oriented programming techniques
- Objects and classes
- Attributes and methods
- Parameter passing
- Abstraction, encapsulation and information-hiding
- Inheritance
- Polymorphism
- Association
- Aggregation
- Coupling
- Cohesion

Evidence Requirements

Candidates will need to provide evidence to demonstrate their Knowledge and Skills by showing that they can investigate object oriented programming techniques and apply them appropriately to a design.

A candidate's response can be judged to be satisfactory where the evidence produced shows the candidate has successfully investigated and applied appropriate object oriented programming techniques.

This will be evidenced by the development of a program which demonstrates the following:

- abstraction, encapsulation and information hiding used where appropriate
- inheritance used only if appropriate to the solution
- Polymorphism used only if appropriate to the solution
- all class-wide variables are private to prevent content coupling
- class-wide variables are kept to a minimum to ensure a minimum of common coupling
- data coupling is used (using parameter passing) in preference to content or common coupling
- program does not contain a lot of unnecessary data coupling
- classes are highly cohesive

Assessment Guidelines

See Outcome 3 Assessment Guidelines

Unit title: Games Development: Object Oriented Programming

Outcome 3

Implement a solution from an object oriented design using object oriented techniques

Knowledge and/or Skills

- Declaring and initialising variables
- Using operators.
- Implementing control structures.
- Defining data structures.
- Accessing and manipulating data structures.
- Using parameter passing.
- Creating Classes.
- Creating instances of classes
- Creating relationships between classes.
- Creating Constructor methods.
- Overloading methods
- Use of exceptions
- Use of standard object libraries
- Use of pre-defined interface components.
- Implementing Event-driven interaction.
- Documenting code.

Evidence Requirements

Candidates will need to provide evidence to demonstrate their Knowledge and Skills by showing that they can implement a working solution from an object oriented design.

A candidate’s response can be judged to be satisfactory where the evidence produced shows the candidate is able to:

- implement a working solution which meets the requirements of the given brief and matches the Class Diagram produced for Outcome 1
- declare and initialise variables
- correctly use arithmetic and/or logical operators
- implement at least one control structure
- implement at least two data structures
- create a minimum of four classes, which contain attributes, methods, and a constructor method
- create a minimum of three objects from the classes, with appropriate initial attribute values set through the Constructor methods
- implement at least one overloaded method (this may be the constructor)
- link the classes appropriately through association, aggregation, or inheritance relationships
- pass parameters correctly both within and between objects
- define the access type (public, private, or protected) for methods, attributes and classes, and the access modifiers chosen are appropriate
- make use of pre-defined Classes and/or Methods from the standard object library
- make use of pre-defined interface components

Unit title: Games Development: Object Oriented Programming

- ♦ implement at least three Event-driven components
- ♦ appropriately handle errors with exceptions or pre-validation
- ♦ implement code commented appropriately throughout

Assessment of this Outcome should be conducted under open-book supervised conditions. Assessors must assure themselves of the authenticity of each candidate's submission.

Assessment Guidelines

This Outcome should be assessed holistically along with Outcomes 1 and 2 in the form of a single project. It is recommended that the project has a minimum of two deadlines: one for the program design and one for the implemented solution, to prevent candidates wasting time implementing an unsuitable design.

It is recommended that candidates are given the option to choose from a selection of projects. Suitable projects would include 2D puzzle games such as Sokoban or Tetris-style games. 2D or 3D scrolling adventure games, or arcade-style games are also suitable projects. You may also allow the candidate to suggest their own project, as long as it will be sufficiently complex to meet all Evidence Requirements, whilst not being too complex to implement in the time allocated. The project specifications should allow the candidates enough flexibility to customise their program. The project should not be too regimented in terms of physical appearance and sound features, but must be precise in its operational requirements. For example, the appearance of the interface should not be specified. However, the manner in which the user of the program must be able to interact with it, such as Keyboard input, and the functionality of the program, must be clear.

Administrative Information

Unit code:	F86A 35
Unit title:	Games Development: Object Oriented Programming
Superclass category:	CB
Original date of publication:	August 2009
Version:	02

History of changes:

Version	Description of change	Date
02	References to F8HC 34 *Structured Programming for Games* added where applicable.	22/07/10

Source: SQA

SQA acknowledges the valuable contribution that Scotland's colleges have made to the development of Higher National qualifications.

Higher National Unit specification: support notes

Unit title: Games Development: Object Oriented Programming

This part of the Unit specification is offered as guidance. The support notes are not mandatory.

While the exact time allocated to this Unit is at the discretion of the centre, the notional design length is 120 hours.

Guidance on the content and context for this Unit

This Unit is intended as a three credit introduction to object oriented programming. Candidates will acquire knowledge of the concepts and principles of object oriented software development. Candidates will then be required to implement object oriented programming skills through the creation of object oriented solutions to problems.

In contrast to existing object oriented programming Units, this Unit is not an introductory programming Unit. It is an introduction to programming in an **object oriented** manner. However, this Unit assumes prior knowledge of programming fundamentals. In addition, there is much less of a focus on documentation and testing than in other programming Units. The design and documentation process for Computer Games projects differs from mainstream software development projects, and a focus on documentation would not be useful for Games students. This Unit also incorporates some aspects of event driven programming techniques. Object oriented and event driven programming are not mutually exclusive, and almost all object oriented languages can be used in an event driven way. A program can both be structured in an object oriented manner and enable user interaction through an event driven interface. An understanding of basic event driven techniques is therefore required for designing programs which have a graphical user interface. This is an important part of Games Programming.

The intention is for this Unit to act as a follow-on to the HN Unit F8HC 34 *Structured Programming for Games*. Candidates should already be familiar with the basic programming building blocks, allowing this Unit to focus on object oriented programming concepts such as classes and inheritance. The Unit will also focus on creating graphical programs rather than console-based programs, enabling the candidates to create simple games. The teaching and learning approaches may include problem-based learning, teamwork, and potentially an element of competition (eg ‘solve in an hour’ code challenges) to engage candidates and encourage self-directed study.

This Unit forms part of an HN Computer Games Group Award program and should be delivered within the context of the Group Award. It would be suitable for candidates who are proposing to follow a career in games design or development. It is not suitable as a stand-alone Unit, but could be taken after successful completion of the HN Unit F8HC 34 *Structured Programming for Games*, or fit into the second year of the HND Computing (Software Development) structure.

As a non-introductory programming Unit, the context and examples used can be relatively complex in terms of the scope and programming fundamentals required. However, they should be relatively simple in terms of object orientation. Using examples of relatively simple games (for example noughts and crosses, battleships, pacman, pong, space invaders) would hopefully make the learning process more enjoyable and meaningful for games students, and would permit the candidate to concentrate more on the programming techniques involved, rather than trying to understand the initial problem.

Unit title: Games Development: Object Oriented Programming

By the end of the Unit, the candidate should have achieved a good foundation in the skills required for developing reliable, robust and efficient object oriented program designs for simple computer games.

It is likely that the Unit will be delivered in the second year of a full-time HN Computer Games Development Group Award.

The Unit is capable of being delivered on its own, but could also be delivered in conjunction with either of the HN Units DH35 34: *Computing: Planning* or DH3G 34: *Systems Development: Object Oriented Design (Introduction).*

Guidance on the delivery and assessment of this Unit

Outcome 1

Demonstrate the ability to analyse a programming problem and design an object oriented solution to the specified problem.

Delivery

Developing the ability to analyse a project brief and design a suitable solution is a skill that is acquired through practice rather than one that can be explicitly taught. This Outcome should be embedded throughout the Unit, and will be acquired through continual practice of the problem-solving and program design process at each stage of Unit.

Although there is no way to easily teach this skill, there are a variety of helpful techniques which can be taught. The object oriented software development methodology should be introduced, if it is not being covered in another Unit on the course. Candidates should be introduced to the requirements elicitation process, and given examples of the process of identifying and clarifying requirements. Candidates should learn some UML diagrams. It would be useful to introduce candidates to Use Case diagrams which help to identify the main functions that a program is required to perform. Candidates must also be taught how to produce Class Diagrams, which show the static structure of the program architecture. Natural Language Analysis is another technique which candidates should be taught as this enables them to identify possible attributes, classes, objects, behaviours, and possible valid attribute values from any problem statement. It would also be beneficial to introduce candidates to Case-Based Reasoning. This is the technique of looking back at previous problems and their solutions and using similar cases as the basis for the design of a solution to the new problem. For example, candidates could be shown the problem of writing a noughts and crosses game, and as a class the solution to this problem could be designed and implemented. Candidates could then be asked to create a Connect 4 game, using the noughts and crosses game as a basis for their design due to the many similarities between the two games.

For user interface design, Storyboarding should be taught as a design technique. Candidates will also need to be taught about standard user interface components such as command buttons, textboxes, labels, radio buttons, checkboxes, and scrollbars, and the suitable purposes for these components. An introduction to Human Computer Interaction and designing effective user interfaces would be beneficial.

Unit title: Games Development: Object Oriented Programming

Assessment

This Outcome will be assessed along with Outcomes 2 and 3 in a single project. The development of a Class Diagram which details the program architecture for the proposed solution will provide evidence for this Outcome. This Outcome also provides evidence for the Core Skill of Problem Solving at SCQF level 6. The process of producing a class diagram from an initial problem statement or project brief provides indirect evidence of problem solving. To capture direct evidence it would be useful to require candidates to keep a wiki discussing how they approached the problem, what techniques they used, what pitfalls they encountered and how they made progress. The wiki could include drafts of the design in various stages, showing what changed at each stage and any designs which were rejected outright or modified, and the reasons for the changes. The wiki would be useful in showing the process, in addition to the final, polished solution. Direct evidence could be captured by other means, but a wiki has the advantage of providing a revision history, which other tools such as blogs do not. Assessment of this Outcome should be conducted under open-book conditions, and may be unsupervised. Assessors must assure themselves of the authenticity of each candidate's submission.

Outcome 2

Demonstrate a broad understanding of object oriented programming techniques.

Delivery

This Outcome covers the main theoretical object oriented concepts which candidates must understand. It should be taught in conjunction with Outcome 3. Outcomes 2 and 3 have been separated to prevent over-assessing candidates as to design and implement a program which includes every object oriented concept is onerous, and also makes it harder to identify suitable projects. However, the object oriented programming techniques required for Outcome 3 are simply a subset of those required for Outcome 2.

The following topics should be covered:

- Objects and classes
- Attributes and methods
- Parameter passing
- Abstraction, encapsulation and information-hiding
- Inheritance
- Polymorphism
- Association
- Aggregation
- Coupling
- Cohesion

Unit title: Games Development: Object Oriented Programming

It is recommended that theory and practice are not separated but are intertwined as much as possible. Candidates should be given several practical exercises for each topic as programming is a skill which is learnt and improved upon through practice. It is recommended that the practical exercises increase in complexity throughout the course of the Unit as new techniques are learnt. The practice of providing examples which show the new technique or skill in isolation is useful for initial understanding of how a technique may be implemented, but is not useful for the problem solving and program design process. Candidates should begin learning how to incorporate new techniques into larger programs from an early stage in the Unit so that there is suitable opportunity for integration of concepts.

Assessment

This Outcome will be assessed along with Outcomes 1 and 3 in a single project. The development of a program which contains a sampled selection of object oriented programming techniques will provide evidence of an understanding of the concepts covered in this learning Outcome.

Outcome 3

Implement a solution from design using object oriented techniques.

Delivery

Outcome 3 covers the practical programming concepts and techniques which candidates must be able to implement in an object oriented program. This Outcome should be taught in conjunction with Outcome 3.This Unit assumes knowledge of fundamental programming concepts, but it would be advisable to briefly refresh these basic topics (variables, operators, iteration, selection, arrays) and show how to implement these using the syntax of the object oriented programming language you have chosen to use. It is recommended that no more than 8 hours should be allocated to these topics, leaving the vast majority of the time to be allocated to object oriented programming techniques.

The following topics should be covered:

- Defining data structures
- Accessing and manipulating data structures
- Using parameter passing
- Creating classes
- Creating instances of classes
- Creating relationships between classes
- Creating constructor methods
- Overloading methods
- Use of exceptions
- Use of standard object libraries
- Use of pre-defined interface components
- Implementing Event-driven interaction
- Documenting code

Unit title: Games Development: Object Oriented Programming

The implementation language chosen is not specified. At the time of writing, the most appropriate languages to use would be Java or C#. Both these languages are widely used in industry and have a similar, C-type syntax. C++ would be less suitable because it provides far less support for creating user interfaces. Java is a good teaching language with a large object library, and the creation of user interfaces is supported through awt and swing. C# allows for the creation of interfaces through simple drag and drop, and the XNA development kit can be used in association with C# for the development of fairly complex computer games.

The particular technical skills taught for event driven programming will be dependent upon the implementation language chosen. For example, if Java is used then candidates will need to be taught about the awt and swing packages, whereas if C# is used the candidates could be taught how to set up user interfaces through code, or be introduced to the toolbox of user interface components and their various properties.

Assessment

This Outcome will be assessed along with Outcomes 1 and 2 in a single project. It is recommended that the project has a minimum of two deadlines: one for the program design and one for the implemented solution, to prevent candidates wasting time implementing an unsuitable design.

It is recommended that candidates are given the option to choose from a selection of projects. Suitable projects would include 2D puzzle games such as Sokoban or Tetris-style games. 2D or 3D scrolling adventure games, or arcade-style games are also suitable projects. You may also allow the candidate to suggest their own project, as long as it will be sufficiently complex to meet all Evidence Requirements, whilst not being too complex to implement in the time allocated. The project specifications should allow the candidates enough flexibility to customise their program. The project should not be too regimented in terms of physical appearance and sound features, but must be precise in its operational requirements. For example, the appearance of the interface should not be specified. However, the manner in which the user of the program must be able to interact with it, such as keyboard input, and the functionality of the program, must be clear.

Opportunities for developing Core Skills

Candidates meet the problem-solving Core Skill through identifying how to design the software solution based on a problem statement which does not provide detailed instructions on how to design or structure a solution. The candidates could be required to write a short report detailing their problem-solving process, ie how they got from the problem statement to the class diagram, in order to fulfil this requirement.

Open learning

If this Unit is delivered by open or distance learning methods, additional planning and resources may be required for candidate support, assessment and quality assurance. A combination of new and traditional authentication tools may have to be devised for assessment and re-assessment purposes.

Higher National Unit specification: support notes (cont)

Unit title: Games Development: Object Oriented Programming

Disabled candidates and/or those with additional support needs

The additional support needs of individual candidates should be taken into account when planning learning experiences, selecting assessment instruments, or considering whether any reasonable adjustments may be required. Further advice can be found on our website **www.sqa.org.uk/assessmentarrangements**

General information for candidates

Unit title: Games Development: Object Oriented Programming

This Unit is designed to cover the object oriented programming skills required for a career in computer games. It is a non-introductory Unit and assumes prior knowledge and proficiency in basic programming concepts and techniques. The HN Unit F8HC 34 *Structured Programming for Games* is a recommended pre-requisite for this Unit.

In this Unit you will acquire knowledge of the concepts, principles, and techniques of object oriented software development necessary to enable you to design and develop object oriented software including computer games. You will also acquire the skills necessary to develop graphical user interfaces for object oriented programs.

This will involve the following areas of learning:

- ♦ Using problem-solving techniques and an object oriented design methodology, you will design an object oriented solution from a given project brief.
- ♦ Using the features of an object oriented programming language, you will implement a software solution based on your design. Your understanding and grasp of object oriented concepts and programming techniques will be reinforced throughout with practical exercises.
- ♦ Using storyboarding and event driven components, you will design and implement a graphical user interface for your object oriented program.