London South Bank
University

Coding and Programming

AME-5-CPG

BA (Hons) Game Cultures

Faculty of Arts and Human Sciences
Department of Arts, Media and English

Semester 1, 2013–2014

Level 5

Table of Contents

1. MODULE DETAILS

Module Title:	Coding and Programming
Module Level:	5
Module Reference Number:	AME_5_CPG
Credit Value:	20 CAT points
Student Study Hours:	164
Contact Hours:	36
Pre-requisite Learning (If applicable):	n/a
Co-requisite Modules (If applicable):	n/a
Course(s):	Game Cultures
Year and Semester	Year 2, Semester 2
Module Coordinator:	Siobhán Thomas
UC Contact Details (Tel, Email, Room)	Telephone: 0207 815 5809 Email: thomass5@lsbu.ac.uk Room: B401, Borough Road
Teaching Team & Contact Details (If applicable):	Paul Sinnett
Subject Area:	Arts and Media
Summary of Assessment Method:	**CW1: Robot war A.I.** An A.I. script to defeat bots in a simple robot arena fighting game. Supported by weekly coding exercises and development log (task tracking reports) (worth 25% of the marks for the module.) Due week 12 of semester 1. **CW2: Small game** A simple video game program demonstrating an understanding of the programming techniques covered during the module. Supported by weekly coding exercises and development log (task tracking reports) (worth 75% of the marks for the module.) Due week 12 of semester 2.

2. SHORT DESCRIPTION

In Coding and Programming you will explore key concepts in programming, within the context of your own game project. The module introduces you to the Unity development environment (a commonly used engine in the game industry) and over the course of the module you will not only develop your understanding of the fundamentals of game programming, but further develop your understanding of the game production processes you learned in year 1. Each week you will learn about key programming concepts that you will be expected to apply towards the creation of your own small game. You'll also be expected to schedule and accurately track your programming tasks. Thus, the module will not only provide you with a conceptual understanding of programming, but will provide you with key project planning and management skill which you can then apply to next year's Advanced Game Project work.

3. AIMS OF THE MODULE

The aims of this module are to:

- Introduce students to programming computers
- Provide familiarity with an industry standard development environment
- Explore key concepts in program construction and testing
- Develop students' understanding of scheduling programming tasks

4. LEARNING OUTCOMES

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

4.1 Knowledge and Understanding

- Understand how computers work
- Read and interpret programs in multiple programming languages
- Understand build processes

4.2 Intellectual Skills

- List and compare programming styles
- Deconstruct a game design into component game objects
- Explain the purpose of common code construction techniques

4.3 Practical Skills

- Implement game designs through the application of consistent rules
- Create an original small scope video game
- Use industry standard game development and collaboration tools

4.4 Transferable Skills

- Construct programs in multiple programming languages
- Track your own programming productivity

5. ASSESSMENT OF THE MODULE

5.1 General Statement about the Philosophy of Teaching and Learning in Game Cultures Classes

Be prepared to receive and give constructive criticism. Part of presenting your work is being able to gracefully accept other people's ideas and critiques, and to assimilate this feedback into your work. You also need to be prepared to give criticism—to share your ideas about other's work—in a constructive way.

5.2 Assessment Overview

- CW1 Game Programming Task: An A.I. script to defeat bots in a simple robot arena fighting game. Supported by weekly coding exercises and task tracking reports (Worth 25% of the marks for the module.) Due week 12 of semester 1.
- CW3 Game: A simple video game program demonstrating an understanding of the programming techniques covered during the course. Supported by weekly coding exercises and task tracking reports. (Worth 75% of the marks for the module.) Due week 12 of semester 2.

5.3 Coursework 1: Game Programming Task

For the first component of the assessment for Coding and Programming, you must remix and share a robot A.I. arena game in the Scratch development environment and supply your own robot A.I. to participate in the game. This will be supported by the programs you create in the class activities in the first block.

You will be assessed on your usage and understanding of the programming concepts covered in the classes.

You are also required to keep track of the tasks you complete as part of this module by maintaining a development log.

These requirements will be fully explained in week 1.

Due: No later than 4.00 pm, Week 12 of semester 1, Friday December 13, 2013, to the Faculty Office in Borough Road, you must submit a DVD/ CD containing the items listed below. At the same time you must upload a digital copy of what you have submitted to the CPG folder on the Game Studio server.

- A clearly labelled disc (AME_5_CPG_CW1_20132014_StudentIDNumber_SurnameFirstname). No printed labels and no CD cases/ sleeves.
- A read me file outlining what is contained on the CD/DVD, any technical instructions and links for playing your games.
- A folder containing exercises from each of the weeks.
- A folder containing your development log
- A folder containing copies of any material that may have inspired you (examples of code, videos, images, etc.)

Weekly Coding Exercises

Getting your head around programming requires you to program every week. The weekly coding exercises are, thus, requirements of the module and are assessed as either pass or fail. Each exercise expands on the concepts covered in class. You must complete ALL of the exercises by each week's due date to pass the module. Exercises must be shared each week on your Scratch account.

5.4 Coursework 2: Game

For the second component of the assessment for Coding and Programming, you must create a small game program in Unity. If you wish to include program code from outside sources in your game code, you must attribute your sources in the comments. This project will also be supported by any programs you create in the class activities in the second block.

You are required to keep track of the tasks you complete as part of this module by maintaining a development log.

Your code must:

1. be free of syntax errors
2. be free of warnings (or explain why the warning is benign)
3. have minimal duplication
4. use descriptive variable and function names
5. be neatly and consistently formatted
6. have high cohesion and loose coupling
7. be usefully commented

You are also expected to manage the schedule and log the development of your own game. These documents must be uploaded each week to version control.

Due: No later than 4.00 pm, Week 12 of semester 2, Friday 9th May 2014, to the Faculty Office in Borough Road, you must submit a DVD/ CD containing the items listed below. At the same time you must upload a digital copy of what you have submitted to the CPG folder on the Game Studio server.

- A clearly labelled disc (AME_5_CPG_CW2_20132014_StudentIDNumber_SurnameFirstname). No printed labels and no CD cases/ sleeves.
- A read me file outlining what is contained on the CD/DVD, any technical instructions and links for playing your games.
- A playable game built for Windows .EXE and Mac .APP. The game must have clearly written, useful comments, and conform to industry standard programming conventions. Within the code anything not written by you must be clearly commented indicating its origins and hyperlinks to its source, if applicable. Failure to comment code not written by you will result in your work receiving an automatic fail due to plagiarism.
- A folder containing the complete source for the game.
- A folder containing each of the exercises from each of the weeks.
- A folder containing your development log
- A folder containing copies of any material that may have inspired you (examples of code, videos, images, etc.)

Successful completion of this module is heavily dependent upon full attendance and punctuality. Students who do not regularly attend or who are consistently late are liable to fail this module and may endanger their chances of progressing beyond this point of the degree programme.

Generic Assessment Criteria for Production Work

Percentage	Quality	Classification
70% - 100%	**Excellent**	**1st**
60% - 69%	**Very good**	**2:1**
50% - 59%	**Good**	**2:2**
40% - 49%	**Weak**	**3rd**
39% or below	**Poor**	**Fail**

1st
PRODUCTION
Outstanding work in all respects: you need to show evidence that your intellectually challenging idea has been translated effectively into the appropriate medium, demonstrating not only technical proficiency but considerable originality in your approach. Your well-presented, labeled production will demonstrate a significant element of fusion in terms of form and content. The work will show consistent attention to detail, and be striking in terms of its conceptual innovation and manipulation of technical processes.

PRODUCTION ANALYSIS
Your Production Analysis needs to be articulate, logically structured and well-presented, including a contents page and page numbering. There should be no grammatical, spelling or typographical errors. The Analysis covers each aspect of the remit's criteria in substantial detail according to the Module Guide remit, with appropriate referencing from a wide range of sources, supported by an appropriately constructed bibliography and any other necessary ancillary evidence, such as newspaper research or flow charts. The Analysis needs to offer consistently insightful and considered scrutiny of both the production process and the finished piece, and deal with wider theoretical issues in a considered and informed manner.

2:1
PRODUCTION
This will be a well-presented, conscientious piece of work, featuring a strong central idea. While it may attempt to explore an intellectually challenging concept the piece may be undermined by some minor errors in terms of technical processes; alternatively while perhaps you demonstrate considerable technical proficiency in terms of how you express the idea in the chosen medium,

the idea itself may be somewhat derivative or lacking in certain aspects of its development. This nonetheless stands out as strong work.

PRODUCTION ANALYSIS
This is a well-referenced, reasonably well-written, well-structured piece of analytical work, demonstrating only minor gaps in knowledge, but dealing with all aspects of the Production Analysis remit, though perhaps not always in quite sufficient depth. Wider theoretical issues are addressed, though they may not be fully explored or integrated into scrutiny of your own production work, and the production processes which led to it.

2:2
PRODUCTION
This is a well-presented piece of work displaying a good central idea, which may be somewhat unoriginal or not fully pursued in terms of intellectual coherence. While it demonstrates technical competence it may contain a significant number of errors, or represent little in the way of creative use of the medium. The production may well be unambitious in its approach in terms of either form or content or both; or it may have been unrealistically ambitious.

PRODUCTION ANALYSIS
A satisfactory piece of work which is clearly written showing a good understanding of the topic. However, the essay may be largely descriptive, or rather generalised in places, or lack sufficient analysis or argument. All or some of the Production Analysis remit's criteria may be covered, though not in sufficient depth. It may be poorly written in terms of grammar, spelling, sentence construction or paragraphing. It may need restructuring or be poorly presented.

<u>3rd</u>
PRODUCTION
This represents a poor expression of the remit in all areas: the central idea will be unoriginal, and/or poorly thought-through, and the execution will demonstrate both a lack of basic understanding and application in achieving basic technical competencies. This work stands out because of its overall poor quality.

PRODUCTION ANALYSIS
A totally descriptive essay, lacking in all areas demanded by the Production Analysis remit. The expression may be poor, with spelling mistakes, weak grammar and a lack of paragraphing. The essay may lack a clear introduction, conclusion or overall structure. The presentation is poor.

<u>Fail</u>
PRODUCTION
This has failed to address all or most elements expressed in the remit: the central idea may be weak and/or unoriginal and little or no attempt will have been made to achieve basic technical competencies in the appropriate medium. Presentation may also be poor.

PRODUCTION ANALYSIS
The work answers very few or none of the Production Analysis remit's criteria. It is badly structured, poorly written and poorly presented. It is purely descriptive and lacks details for analysis. There is little evidence of planning or of understanding the module objectives or assessment criteria. Work will have to be resubmitted to gain a maximum of 40%.

6. FEEDBACK

You will receive prompt feedback on the development of your work during weekly practice sessions, one-to-one tutorials, presentations and, in some cases, by email. Prompt feedback is often verbal feedback and part of the way you learn to develop a critical practice. It is an essential element of your academic studies on a practice-led course so that your own work benefits from staff experience. This ongoing prompt feedback is known as formative feedback. You need to take an active role in this process and come to practice sessions and tutorials with work completed so that you can get the most effective feedback to help improve your work. During these sessions, you are expected to ask your tutors for feedback rather than only waiting for your tutor to offer you feedback. In other words, acquiring feedback is a process largely directed by you.

It is useful to form study groups with other students to assist in peer support and learning. You should also seek external feedback from professionals in the industry when and wherever you can.

You will also receive, in addition to this prompt formative feedback, final written summative feedback and a grade once your work has been submitted for assessment, normally at the end of the module, and after your work has been assessed and double marked. The Course Director, or your module tutor, will email you to alert you when this feedback will be distributed, after the completion of the module.

Final marks are not confirmed until after the External Examiner's visit and the meeting of the Award and Progression Board in the summer. You can also arrange to see the module tutor or Course Director for further clarification and feedback if necessary once the summative feedback has been distributed.

7. INTRODUCTION TO STUDYING THE MODULE

7.1 Overview of the Main Content

- Building games from the ground up
- Foundation in required logic and mathematics
- Development of programming languages
- Code formatting
- Structure and factoring
- Programming terminology and concepts
- Use of game engines

7.2 Overview of Types of Classes

- Laboratory-based contextual sessions
- Laboratory-based practice sessions

7.3 Importance of Student Self-Managed Learning Time

As with every practice-module on the BA (Hons) Game Cultures course, you are expected to work independently outside of class hours on every aspect of the project. For group projects this will necessarily mean meeting fellow students at mutually convenient times; for individual projects this will mean committing time to work in the University laboratories or, if you have the option, working at home on your project.

7.4 Employability

This module introduces students to a highly sought after job role in the game industry: that of the programmer. After completing the module students will understand programming fundamentals and be able to write code that can be understood and used by members of professional game development teams. While students may not wish to pursue a career as a programmer, the module will give students knowledge and lexicon to effectively collaborate with programmers.

8. THE PROGRAMME OF TEACHING, LEARNING AND ASSESSMENT

Block One Wks 1-12 semester 1	**Block Two Wks 1-12 semester 2**
Introduction to programming: learning programming techniques in Scratch.	Transferring programming skills into the Unity 3D development environment.

8.1 Semester 1

Week 1: Introduction, Tuesday 24th September 2013

A brief overview of the course.

This introduction will cover:
- the structure of the course
- overview of the weekly lectures
- what to expect in terms of workload
- doing your own research
- an introduction to the Scratch environment

Activities:
- list values important to you
- sign up with Scratch
- hello world (your first program)
- "you are awesome" (your real first program)

Week 2: Game programming in practice, Tuesday 1st October 2013

This lecture will cover the challenges you'll face as you develop your game programming skills.

We'll examine the two main enemy types that programmers typically face:
- the adversary
- the imposter

And some techniques for defeating them:
- unit testing
- pair programming
- self-deprecation jar
- rubber duck debugging

Activities:
- find the imposter
- self-testing games
- pass your own test

Week 3: Adventure game logic, Tuesday 8th October 2013

This lecture will cover how games keep track of our state within the game. We will also experiment with the atoms of game logic: AND, OR, and NOT.

Here we will look at:

- conditional expressions
- the value of an expression
- combining logical expressions
- named data

Activities:

- doors
- keys
- adventure game

Week 4: The game loop, Tuesday 15th October 2013

Here we take a look at how game programs can quickly turn into spaghetti, and how structure and loops can make your game programs smaller, and easier to maintain and modify.

Uses for loops in game programming:

- arranging
- waiting
- avoiding duplication
- repeating things
- route finding and other algorithms

Activities:

- drawing shapes
- laying out a grid

Week 5: Structuring your game data, Tuesday 22nd October 2013

This week we'll take a look at the main data structures and their typical uses in game programs; also the scope of variables and how keep things modular.

Topics:

- scope: local, instance, global, static
- arrays
- lists
- trees
- The Matrix (2 dimensional arrays)

Activities:

- laying out an array of enemies
- guess the animal game

Week 6: Problem solving, Tuesday 29th October 2013

Knowing the syntax of a programming language inside out doesn't get you very far. Game programming work is mostly solving problems. This week we'll look at some general tips for problem solving.

Topics:

- don't give up
- do take breaks
- solve a different problem
- research
- spiking
- low hanging fruit
- black box abstraction
- divide and conquer
- recursion

Activities:

- solve the puzzle

Week 7: Estimation, Tuesday 5th November 2013

Estimation is an important skill in all areas of game development. Here we'll look at techniques for producing estimates and where estimates are useful:

- abstraction
- tracking time
- keeping a devlog
- recording performance
- processing in parallel
- the critical path
- lies, damned lies, and schedules

Activities:

- make a devlog
- planning poker

Week 8: Cooperative game programming, Tuesday 12th November 2013

Most programmers work in teams. This week we'll look at technology and techniques to make teamwork easier.

Topics:

- source control
- committing changes
- merging files
- abstract interfaces
- library code

Activities:

- registration with Google Code
- game programming to an interface

Week 9: Big numbers, Tuesday 19th November 2013

Our brains are not particularly good at grasping the kinds of scale and progression we commonly encounter in game development. This week we'll look at techniques and terminology to help us manage these big numbers.

Topics:
- constants
- logs
- linear growth
- square (and triangle) growth
- exponential growth
- geometric growth

Activities:
- guess the progression
- permutations

Week 10: Object-oriented game programming, Tuesday 26th November 2013

This week we'll look at an alternative way to design game programs. We'll imagine a game as the interactions of many small and individual parts rather than a big single list of instructions.

Topics:
- object-oriented programming
- tell don't ask
- parallel programming
- continuations

Activities:
- decompose a classic game

Week 11: Randomness, Tuesday 3rd December 2013

Game designs often require some form of randomisation. However, it's not actually possible to create a truly random number in software. But we can create pseudo-random numbers.

Topics:
- pseudo random numbers
- seeding
- procedural data
- random shuffle

Activities:
- write the perfect song shuffle
- procedural generation

Week 12: A.I., Tuesday 10th December 2013

Good A.I. can bring a game world to life. This week, we look at A.I. and it's applications within gaming

Topics:
- maze solving
- route finding
- simulated annealing

Activities:
- SHALL WE PLAY A GAME?
- find your best match

Assessment game project: Due Friday, December 13th, 4 p.m.

Your final robot A.I. program and work log must be submitted before the deadline.

8.2 Semester 2

Week 1: Unity, Tuesday 28th January 2014

This week we'll transfer our programming experience over to the Unity development environment.

Topics:
- syntactic sugar
- Unity Script
- using the debugger

Activities:
- porting games from Scratch

Week 2: 3D, Tuesday 4th February 2014

This week we'll revise the 3D mathematics that you'll need to successfully write scripts for 3D gameplay in Unity.

Topics:
- vectors
- splines
- interpolation
- matrices
- quaternions

Activities:
- make an orrery
- splines from first principles

Week 3: Alternative counting systems, Tuesday 11th February 2014

Some game programming problems are best solved through clever manipulation of numbers. This week, we'll look at some unusual number systems.

Topics:
- binary
- bumfit
- hexadecimal vs. octal
- why powers of 2?
- why start with zero?

Activities:
- traditional counting sheep

Week 4: Binary encoding, Tuesday 18th February 2014

All of your game data, including your program code, are encoded in binary. This week we'll look at different data file formats and how they all end up as ones and zeros.

Topics:
- information theory
- bitmaps
- logic
- binary encoding of: text, programs, audio, and video

Activities:
- the memory trick
- painting with numbers

Week 5: Sorting and searching, Tuesday 25th February 2014

Here we'll continue to expand our problem solving toolset by researching the different kinds of search and sort algorithms computer science has to offer.

Topics:
- uses in games
- good choices
- optimisations
- how they look
- how they sound

Activities:
- pick an algorithm
- sorting students

Week 6: Performance, Tuesday 4th March 2014

Game programmers are frequently limited by processor speed and memory considerations; both in terms of runtime footprint and executable image size.

Topics:

- O notation
- synchronous performance
- asynchronous performance
- critical path revisited

Activities:

- find the bottleneck
- data packing

Week 7: Parallel programming, Tuesday 11th March 2014

As the number of processors on gaming devices increases, game programmers are increasingly required to run code in parallel.

Topics:

- multi-threaded programming
- parallel programming
- pure functions
- functional programming languages
- advantages of avoiding side effects

Activities:

- threading in Unity
- fun with shaders

Week 8: Transistors, Tuesday 18th March 2014

It's a curious feature of games programming and programming in general, that many of the principles apply in some form at every level of abstraction in game interactions right down to the individual transistors. This week we'll take a bottom up look at the hardware that runs our game programs.

Topics:

- how transistors work
- alternative technologies
- logical operations revisited
- adding with logic

Activities:

- complete the circuit
- the game of life

Week 9: Paradigm shifts, Tuesday 25th March 2014

Computer science is riddled with dichotomies and unresolved issues. This week we'll explore the dichotomies that game programmers commonly face.

Topics:

- Turing complete vs. language wars
- spaces vs. tabs
- K&R vs. Allman
- camelCase vs. under_score
- garbage collection vs. pointers
- duck vs. static types
- object-oriented vs. data-oriented
- functional vs. procedural
- declarative vs. imperative
- compiled vs. interpreted

Activities:

- write a style manual

Week 10: Memory, Tuesday 1st April 2014

This week we'll examine how computer memory works.

Topics:

- flip-flops
- clocks
- accessing memory
- types of memory

Activities:

- make a flip-flop
- cache friendly data

Week 11: The limits of programming, Tuesday 28th April 2014

When working near the limits of video game technology, it's useful to know where the edges are.

Topics:

- Quine programs
- the halting problem
- travelling salesman problem

Activities:

- write a Quine program
- solve the travelling salesman's problem

Week 12: The history of game development, Tuesday 6th May 2014

This week we'll look at the development of games from Ralph Baer's brown box console through to modern day consoles and into the future of game devices.

Topics:
- hardwired consoles
- machine code
- assembly language
- C and 3rd generation languages
- middleware and Unity
- the future...

Activities:
- CPU simulator

Assessment game project: Due Friday, 9th May 2014, 4 p.m.

Your final Unity game project and work log must be submitted before the deadline.

9. STUDENT EVALUATION

This module's student evaluation will occur in week 12 of semester 2 and will be an opportunity for you to provide feedback about the module in a formal way. We encourage students, however, to provide feedback regarding teaching and learning on an ongoing basis throughout the semester. In the Game Cultures Course we take excellence in teaching very seriously. You can play a pivotal role in helping us achieve teaching excellence by providing us with feedback.

If you have suggestions, questions or concerns about the module please discuss them with your module lecturers or tutors. If you have any suggestions, questions or concerns about the Course in general please contact the Game Cultures Course Director, Siobhan Thomas via email thomass5@lsbu.ac.uk.

10. LEARNING RESOURCES

10.1 Core Materials

Games specific

- Gregory, J. (2009). *Game engine architecture*. Wellesley, Mass.: A K Peters
- Dawson. M. (2011). *Beginning C++ through game programming.* Boston, Mass.: Cengage Learning

General programming

- Nisan, N., & Schocken, S. (2008). *Elements of Computing Systems: Building a Modern Computer from First Principles.* MIT Press.
- Abelson, H. (1985). *Structure and interpretation of computer programs*. Cambridge Mass: MIT Press.
  This book is freely available on the web in HTML:
  http://mitpress.mit.edu/sicp/full-text/book/book.html
  and PDF:
  http://www.scribd.com/doc/15556326/Structure-and-Interpretation-of-Computer-Programs-SICP
  and with videos of the lectures by the authors:
  http://groups.csail.mit.edu/mac/classes/6.001/abelson-sussman-lectures/
- Kernighan, B., & Ritchie, D. (1988). *The C Programming Language (2nd ed.).* Prentice Hall.
- Stroustrup, B. (2000). *The C++ Programming Language (3rd ed.).* Addison Wesley.
- Kernighan, B. (1999). *The practice of programming*. Reading, MA: Addison-Wesley.

For catalogue of different ways to factor programs:

- Fowler, M. (2007). *Refactoring: Improving the design of existing code* (1st ed.). Boston: Addison-Wesley.

To point out the pitfalls of programming in C++ consult the Scott Meyers series:

- Meyers, S. (2005). *Effective C++: 55 specific ways to improve your programs and designs* (3rd ed.). Upper Saddle River, NJ: Addison-Wesley.
- Meyers, S. (1996). *More effective C++: 35 new ways to improve your programs and designs.* Reading Mass.: Addison-Wesley.
- Meyers, S. (2001). *Effective STL: 50 specific ways to improve your use of the standard template library*. Boston: Addison-Wesley.

For a look at current and future directions for the C++ language:

- Alexandrescu, A. (2007). *Modern C++design: Generic programming and design patterns applied* (15th ed.). Munich: Addison-Wesley.

11. NOTES

11.1 Notes

LEVEL 5: Coding and Programming (AME-5-CPG) Coursework 1 2013-2014
Lecturer: Paul Sinnett
STUDENT NAME:

A.I., Supported by Weekly Coding Exercises (Weighting 25%)	1st Excellent 70-100%	2.1 Very good 60-69%	2.2 Good 50-59%	3rd Weak 40-49%	Fail Poor 39%-0%
Weekly exercises					
Week 1					
Week 2					
Week 3					
Week 4					
Week 5					
Week 6					
Week 7					
Week 8					
Week 9					
Week 10					
Week 11					
Week 12					
Robot A.I. project					
Scratch project shared correctly					
Task tracking log					
Free of errors					
Free of bugs					
Minimal duplication of code					
Descriptive naming					
Tidy blocks					
High cohesion					
Loose coupling					
Useful comments					
Demonstration of programming concepts					
Removal of redundant blocks					

<u>Lecturer's comments</u>

I recommended that you visit the Learning and Development Centre for additional study skills support	Yes or no
I have read the student's Support Arrangements Form and have marked the attached work in accordance with the University's DDS Marking Policy.	Yes or n/a

Final mark:	**Date:**	**Signature:**

The mark or grade indicated on this form is provisional until it has been considered and approved or modified by the appropriate Subject Area Examination Board.

LEVEL 5: Coding and Programming (AME-5-CPG) Coursework 2 2013-2014
Lecturer: Paul Sinnett
STUDENT NAME:

Small game, supported by weekly exercises and development log (Weighting 75%)	1st Excellent 70-100%	2.1 Very good 60-69%	2.2 Good 50-59%	3rd Weak 40-49%	Fail Poor 39%-0%
Weekly exercises					
Week 1					
Week 2					
Week 3					
Week 4					
Week 5					
Week 6					
Week 7					
Week 8					
Week 9					
Week 10					
Week 11					
Week 12					
Game project					
Unity project correctly configured					
All source files included					
All assets included					
Task tracking log					
Free of syntax errors					
Free of build warnings					
Free of bugs					
Minimal duplication of code					
Descriptive variable and function names					
Neat and consistent formatting					
High cohesion					
Loose coupling					
Useful comments					
Demonstration of programming concepts					
Removal of redundant code					

Lecturer's comments

I recommended that you visit the Learning and Development Centre for additional study skills support	Yes or no
I have read the student's Support Arrangements Form and have marked the attached work in accordance with the University's DDS Marking Policy.	Yes or n/a

Final mark:	**Date:**	**Signature:**

The mark or grade indicated on this form is provisional until it has been considered and approved or modified by the appropriate Subject Area Examination Board.