**Programming 2**

**4CC511**

Assessment Brief

Dr Patrick Merritt

Programming 2 4CC511

Contents

[Module Leader 3](#_Toc131061219)

[Key dates and details 3](#_Toc131061220)

[Description of the assessment 3](#_Toc131061221)

[Assessment Content 3](#_Toc131061222)

[Mandatory Exercises (20%) 4](#_Toc131061223)

[In-Class Test (40%) 4](#_Toc131061224)

[Application Development (40%) 4](#_Toc131061225)

[Defining an application 4](#_Toc131061226)

[Assessment Content 5](#_Toc131061227)

[Advice to help you… 6](#_Toc131061228)

[Submission requirements 6](#_Toc131061229)

[Fit to submit form 7](#_Toc131061230)

[Anonymous Marking 7](#_Toc131061231)

[Assessment Rubric 8](#_Toc131061232)

[Feature checklist 12](#_Toc131061233)

# Module Leader

Mr Patrick Merritt

[p.merritt@derby.ac.uk](mailto:p.merritt@derby.ac.uk)

Office Hours:- 12 – 2pm Mondays MS 308

# Key dates and details

|  |  |
| --- | --- |
| **Assessment Type:** | Individual  Portfolio |
| **Assessment weighting:** | 100% |
| **Word count/Length** | N/A |
| **Learning Outcomes:** | 1. Design, develop, and test computer applications of moderate complexity, using a professional approach  2. Demonstrate the ability to implement software that meets the requirements described in a specifications document |
| **Submission Method:** | Blackboard Assignment & Computer-Based Test |
| **Submission Date:** | 12 noon UK time, 22/05/2023 |
| **Provisional Feedback Release Date:** | 16:00 UK time, 12/06/2023 |

# Description of the assessment

Students are required to complete a series of programming exercises submitted as a portfolio of work.

# Assessment Content

Programming 2 is an assessment portfolio made up of 3 parts

* Mandatory weekly exercises are set in sessions 1 - 4 Deadline is session 8. (20% of portfolio).
* One in-class test taken in the lab session 8. (40% of portfolio).
* One application developed in line with the specification outlined in this document. (40% of portfolio).

You **MUST** make a submission for all parts of the assessment, failure to complete 1 or more parts will result in a non-submission being recorded.

## Mandatory Exercises (20%)

4 weeks of mandatory exercises will be set in each lab session in sessions 1,2,3 and 4. Students are expected to complete all these exercises, with a maximum of 5% of the overall portfolio grade being awarded for each completed section. Partial completion of a week's exercises may yield a partial credit being awarded at the lab instructor's discretion.

## In-Class Test (40%)

An in-class computer-based test will be sat in Lab session 8 and will test students' knowledge of core C# programming concepts. These will be computer-based tests that will be sat under exam conditions.

## Application Development (40%)

You will develop an application using the Unity game engine suitable for demonstrating your ability to implement a working application using the C# programming language. The finished application should be considered something that could be considered useable and serve a function. You must also perform various forms of testing to validate the functionality of your application and its suitability for the purpose it was designed.

# Defining an application

An application, in this case, is defined as a reasonably complex piece of Software written using the Unity engine that a reasonable individual could deem to serve a purpose and be suitable for inclusion in a portfolio of work.

* A purpose, in this scenario, means someone could reasonably want to use it (for example).
  + A video game.
  + An inventory management system
  + An application visualising the process of investigating the contents of a hard drive.
* Art assets and sound will primarily be sourced from external packs.
  + You do not need to develop these assets yourself; however, you must be able to demonstrate an appropriate licence for the assets you use.
* The application developed should open and close in a manner expected for an application of the type developed.

# Assessment Content

Your task is to develop an application demonstrating your abilities with the C# programming language. You must use the Unity game engine for this purpose.

You will be assessed on the following factors.

* The correct usage of elements of the C# programming language used in the solution.
* The complexity of the application.
* Your ability to follow correct HCI principles in your application.
* Your ability to produce readable code solutions following an appropriate style guide.
* Your ability to organise the folder structure in the Unity engine in an appropriate manner.

**PLEASE MAKE SURE YOU:**

* **READ THE RUBRIC AT THE END OF THIS DOCUMENT TO UNDERSTAND HOW MARKS ARE ALLOCATED**
* **USE THE FEATURE CHECKLIST AT THE END OF THIS DOCUMENT**

The complexity and functionality of the techniques used will significantly impact the grade awarded for these components. Features implemented must serve a purpose in the application, and no marks will be awarded for features with no obvious purpose.

You do not have to design an original application (but may do so if you wish, but no marks will be awarded for this); however, you should ensure that the application you choose can demonstrate the components needed for this module. If you are unsure about this, please ask your lecturer for help.

The application you produce will be demonstrated in a viva with your lecturer on the week of the submission date, and slots for demonstrations will be organised with your lecturer closer to the time. If you are unable to attend the viva in person (such as due to remote working), you need to contact your lecturer to make alternative arrangements.

Failure to attend the viva or make an alternative arrangement before the viva starts will result in a grade of zero for assessed component 3 of this coursework. You must ensure the application you have created is demonstratable on the lectern in the demonstration lab before your viva.

Demonstrations which fail to work and cannot be fixed within 1 hour of the viva starting will not be graded.

# Advice to help you…

Your lecturer in lab sessions will serve as a technical advisor and as a QA tester. Use them to maximise your performance.

* You should be turning up with a build each week to discuss with the lecturer.
* You should try out each other’s applications.
* You should get other people outside the course to try your application.

Doing this will gain you valuable feedback you can document and respond to improve the quality of your application.

I strongly advise you to create a design document. If you bring this to the lecturer in the lab session, they can advise you on the scope and feasibility of the application you propose.

The application does not need to be unique. It is perfectly acceptable to attempt to clone an existing game and reskin it with your own theme. If you are stuck for ideas, come talk to the lecturer.

Pick an application you want to make, you will find working on it easier.

Start early, don’t leave it to the last second…

And remember the golden rule…

IF IN DOUBT, ASK THE LECTURER!

# Submission requirements

* One zipped folder containing the following to blackboard
  + All source files used in the creation of this game.
  + Your test logs in a word document.
  + The completed fit-to-submit checklist.
  + A readme file containing the following information.
    - The version number of Unity used to produce the game.
    - Brief instructions for how to run the application.
    - A brief description of the application.
    - A list of all controls for the application.

You must submit the project as a single zip file. Files submitted in other formats will be treated as non-submission and will not be graded. Zip files should use the following naming format, **<StudentNumber> Game Behaviour Assignment One.zip**

# Fit to submit form

|  |  |
| --- | --- |
| Submission requirement | Included |
| One folder containing all source files for the application you have produced. | yes |
| One word document containing your testing logs. | yes |
| One readme file with the appropriate information. | Yes |
| One completed copy of this document. | Yes |
| Above files are contained in a zip folder with the appropriate naming format. | Yes |

# Anonymous Marking

You must submit your work using your **student number** to identify yourself, not your name. You must not use your name in the text of the work at any point. When you submit your work in Turnitin you must submit your student number within the assignment document and in the *Submission title* field in Turnitin.   
  
Assessment Regulations

The [University's regulations, policies and procedures](https://www.derby.ac.uk/about/academic-regulations/) for students define the framework within which teaching and assessment are conducted. Please make sure you are familiar with these regulations, policies and procedures.

# Assessment Rubric

|  |  |  |  |
| --- | --- | --- | --- |
| Grade Boundary | Application (70%) | Style & Organisation (15%) | Application Complexity and Quality (15%) |
| 80%+ (Professional) | Application scores 80 points or more from the feature checklist. | The code style is of a professional standard with no errors, clearly following a style guide.  The organisation of the project is sufficient so that the assessor can easily find files of their choice without problems.  File and asset names are appropriate. | The application presented is of professional quality to be considered ready for deployment to a wider audience.  The application is considered to be significantly above the normally expected complexity for a level 4 student.  The application has **NO** usability issues and the student can demonstrate the inclusion of accessibility features or design choices to make the application accessible for disabled users. |
| 70% - 79% (Excellent) | Application scores 70 points or more from the feature checklist. | The code style is of an excellent standard to be considered a workable quality by the assessor. Commenting is provided consistently throughout the application where appropriate to aid others in understanding the program without being verbose or excessive. A style guide is followed with a minimal number of errors.  The organisation of the project is sufficient so that the student can easily find files of their choice without problems and an outside user should be able to follow the logic of the structure with minimal issues.  File and asset names are appropriate. | The application presented is of Excellent quality and suitable for inclusion in a portfolio of work highlighting the student’s skills.  The application is considered to be of the normally expected complexity for a level 4 student, with one or more features showing potential for implementing advanced features.  The application has **NO** usability issues, and the student can produce a robust testing plan to validate this. |
| 69% - 60% (Very Good) | Application scores 60 points or more from the feature checklist. | The code style is of a very good standard with the majority of the code being of a workable standard but with significant issues in places. Commenting is provided consistently throughout the application where appropriate to aid others in understanding the program but is verbose or excessive. A style guide is followed but a few areas of the code are inconsistent.  The organisation of the project is sufficient so that the student can find files, but an outside user may have trouble following their logic.  File and asset names are appropriate. | The application presented is of very good quality and suitable for inclusion in a portfolio of work highlighting the student’s skills.  The application is considered to be of the normally expected complexity for a level 4 student.  The application has only minor usability issues, which cause frustration but do not affect the performance of the application. |
| 59% - 50% (Good) | Application scores 50 points or more from the feature checklist. | The code style is of a good standard, with some of the code being of a workable standard in places, but significant issues exist in many areas of the project. Commenting is provided inconsistently throughout the. A style guide is not followed but a few areas of the code can be considered consistent.  The organisation of the project is attempted but is either inconsistent or illogical, an outside user will have trouble following their logic.  File and asset names are mostly appropriate. | The application presented is of good quality and, with minor additional work based on feedback, could be considered of acceptable quality for inclusion in a portfolio of the student’s work.  The application is considered to be of the normally expected complexity for a level 4 student.  The application has usability issues, which impact the performance of the application to the point it becomes difficult to use. |
| 49% -40% (Satisfactory) | Application scores 40 points or more from the feature checklist. | The code style is satisfactory as to produce a working program; however, working with the code would be problematic for anyone unfamiliar with the code base. A style guide is clearly not followed.  The organisation of the project is not attempted to any meaningful level.  File and asset names are inappropriate. | The application presented requires major work to be considered of acceptable quality for inclusion in a portfolio of the student’s work.  The application is below the normally expected complexity for a level 4 student.  The application has major useability issues such that only someone with working knowledge of the program could operate it. |
| 39% - 35% (Unsatisfactory) | Application scores 35 points or more from the feature checklist. | The code style and/or project organisation are unsatisfactory to the point that it is almost impossible to identify how the application functions. | The application is well below the standard of expected complexity for a level 4 student.  The program is not an application and more a series of technical demonstrations. |
| 0 > 34% (Fail) | The application demonstrates less than 35 points from the feature checklist. | Little of merit was demonstrated or the demonstration fails to run. | Little of merit was demonstrated or the demonstration fails to run. |

# Feature checklist

|  |  |  |
| --- | --- | --- |
| **C1** | Collections and arrays are used appropriately. | / 10 |
| **C1.1** | 1 List used appropriately. | / 2 |
| **C1.2** | 1 Dictionary used appropriately. | / 2 |
| **C1.3** | 1 Additional collection used appropriately. | / 3 |
| **C1.4** | 1 Array used appropriately. | / 3 |
| **C2** | Conditions and loops are used correctly. | / 10 |
| **C2.1** | Condition created and used in an appropriate manner. | / 2 |
| **C2.2** | Logical operator used correctly. | / 2 |
| **C2.3** | While loop used appropriately. | / 2 |
| **C2.4** | For loop used appropriately. | / 2 |
| **C2.5** | Foreach used appropriately. | / 2 |

|  |  |  |
| --- | --- | --- |
| **C3** | Basic Classes. | / 20 |
| **C3.1** | Class created that inherits from monobehaviour. | / 2 |
| **C3.2** | Access modifiers set appropriately for a single class. | / 4 |
| **C3.3** | Property created and used effectively. | / 3 |
| **C3.4** | Public method created and used effectively. | / 2 |
| **C3.5** | Private method created and used effectively. | / 2 |
| **C3.6** | Constructor created and used effectively. | / 3 |
| **C3.7** | Set only Property created and used appropriately. | / 2 |
| **C3.8** | Get only property created and used appropriately. | / 2 |

|  |  |  |
| --- | --- | --- |
| **U4** | Advanced Classes. | / 15 |
| **U4.1** | Class created that inherits from a class **other** than monobehaviour. | / 3 |
| **U4.2** | Class implements an interface appropriately. | / 3 |
| **U4.3** | Abstract class created and used effectively. | / 3 |
| **U4.4** | Protected keyword used appropriately. | / 3 |
| **U4.5** | Static member used appropriately. | / 3 |
| **U.4.6** | Static class used appropriately. | / 3 |
| **U5** | Polymorphism. | / 10 |
| **U5.1** | Runtime polymorphism utilised effectively. | / 4 |
| **U5.2** | Compile time polymorphism utilised effectively. | / 4 |
| **U5.3** | Both Runtime and Compile time polymorphism used in the same class and clear explanation given as to the difference. | / 2 |

|  |  |  |
| --- | --- | --- |
| **H6** | Design principles. | / 15 |
| **H6.1** | Visibility of system status. | / 3 |
| **H6.2** | Match between the system and the real world. | / 3 |
| **H6.3** | User control and freedom. | / 3 |
| **H6.4** | Consistency and standards. | / 3 |
| **H6.5** | Error prevention. | / 3 |
| **H6.6** | Recognition not Recall. | / 3 |
| **H6.7** | Flexibility and efficiency of use. | / 3 |
| **H6.8** | Aesthetic and minimalist design. | / 3 |
| **H6.9** | Help users identify and recover from errors. | / 3 |
| **H6.10** | Help and Documentation. | / 3 |
| **R7.0** | Advanced Interfaces | / 10 |
| **R7.1** | Use the IEnumerable interface in a meaningful way. | / 5 |
| **R7.2** | Implement ICollection interface or a derivative of it in a meaningful way. | / 5 |
| **R.8** | Innovation | / 10 |
| **R8.1** | Additional marks for innovative solutions at the lecturers discretion. | / 10 |