**Faculty of Computing, Engineering & Media (CEM)**

**Coursework Brief 2023/24**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Module name:** | | Quality Assurance | | | | | | | |
| **Module code:** | | IMAT3003 | | | | | | | |
| **Title of the Assessment:** | | Assignment 1: Quality Assurance & Testing with Labyrinth | | | | | | | |
| **This coursework item is:** ***(delete as appropriate)*** | | | | Summative | | |  | | |
| **This summative coursework will be marked anonymously: *(delete as appropriate)*** | | | | | |  | | | No |
| **The learning outcomes that are assessed by this coursework are:**   1. Comprehensively understand the standards of Quality Assurance, by implementing effective quality checks, and discerning and pre-empting quality issues before they arise. 2. Demonstrate established Quality Assurance considerations when developing content for cross-platform applications, including aspects, such as handling memory issues and how to disseminate this knowledge across a production pipeline schedule/team, many of whom will not be specialist in Quality Assurance methodologies. | | | | | | | | | |
| **This coursework is**: | | | Group | | | | |  | |
|  | | | | | | | | | |
| **This coursework constitutes** 100% **of the overall module mark.** | | | | | | | | | |
| **Date Set:** | Friday 3rd November 2023 (Week 5) | | | | | | | | |
| **Date & Time Due (the deadline):** | **Friday 15th December 2023 at 5:00pm (Week 11)** | | | | | | | | |
| **In accordance with the University** [**Assessment and Feedback Policy**](https://www.dmu.ac.uk/about-dmu/quality-management-and-policy/academic-quality/learning-teaching-assessment/assessment-feedback-policy.aspx), **your marked coursework and feedback will be available to you on:** | | | | | Friday 26th January 2024 | | | | |
| You should normally receive feedback on your coursework **no later than 15 University working days after the formal hand-in date,** provided that you have met the submission deadline  If for any reason this is not forthcoming by the due date your module leader will let you know why and when it can be expected. The Associate Professor Student Experience ([CEMstudentexperience@dmu.ac.uk](mailto:CEMstudentexperience@dmu.ac.uk)) should be informed of any issues relating to the return of marked coursework and feedback. | | | | | | | | | |
| **When completed you are required to submit your coursework via:**   * **Part 1: Test Cases (table) and Bug Reports. (Week 11)**   + All in one document in Word or PDF. Submitted via TurnItIn link on LearningZone. * **Part 2: Unity Project with New Features and Unit Tests. (Week 11)**   + One zipped folder, submitted via LearningZone Link. * **Part 3: Report. (Week 11)**   + 500 word approx. One document in Word or PDF. Submitted via TurnItIn link on LearningZone. * **Game demonstration (Week 15)** * **Peer-Review** documentation by **Monday 15th January**, **(week 16).**   + *See below for more details*   If you need any support or advice on completing this coursework please visit the Student Matters tab on the CEM LearningZone shell. | | | | | | | | | |
| **Late submission of coursework** **policy:**  Late submissions will be processed in accordance with current [University regulations](https://www.dmu.ac.uk/about-dmu/quality-management-and-policy/academic-quality/academic-regulations-assessment-boards/academic-regs-assessment-board-homepage.aspx).  ***Please check the regulations carefully to determine what late submission period is allowed for your programme.*** | | | | | | | | | |
| **Academic Offences and Bad Academic Practices:**  Please ensure you read the section entitled “Academic Offences and Bad Academic Practice” in the module handbook or the relevant sections in this link: [BaseCamp Link: Overview: Assessment and Good Academic Practices](https://eobject.dmu.ac.uk/Basecamp/content/) | | | | | | | | | |
| **Tasks to be undertaken:**  *See below for more details* | | | | | | | | | |
| **Deliverables to be submitted for assessment:**  *See below for more details* | | | | | | | | | |
| **How the work will be marked:**  *See below for more details and matrix* | | | | | | | | | |
| **Module leader/tutor name:** | | Nasar Hasshu, Salim Hasshu | | | | | | | |
| **Contact details:** | | [nasar.i.hasshu@dmu.ac.uk](mailto:nasar.i.hasshu@dmu.ac.uk) [salim.hasshu@dmu.ac.uk](mailto:salim.hasshu@dmu.ac.uk) | | | | | | | |

Should you need any further information or advice please email [cemadvicecentre@dmu.ac.uk](mailto:cemadvicecentre@dmu.ac.uk)

**Assignment 1: Quality Assurance & Testing with Labyrinth**

For this assignment, you will be working on the game provided for you and developing and testing on it. You need to download Labyrinth game from LearningZone. There are **Three** parts to the assessment. First part you need create Test Cases in Excel and check if they pass, if they fail document them using bug reports. The second part is the *Implementation* and testing *of* *New Features* using *Unit Tests* for existing and new features. Finally, there is a **report** submission.

****

Once you have downloaded Labyrinth, have a play. The game is designed for PC and has 3 levels. Use Up, Down, Left, Right / W, S, A, D keys to tilt platform. The input keys should not be inverted. The goal of the game is tilt platform so that the ball reaches the goal.

Please ensure you are using **Unity version 2022.3.4f1** to ensure compatibility.

**Part 1: Initial Functional & Ad-hoc Testing.**

This assignment is about Quality Assurance and Testing. Plan and write test cases for the game based on typical requirements of Labyrinth. This can be in an Excel Document (template provided in lab). An example of test cases can be; Platform Movement (left right up and down), Ball Movement, Goal trigger, etc. You must log the outcome of these tests.

There are bugs in this game. You must create bug reports for the bugs that you find in the game through ad-hoc testing. These bugs can be: Input Issues, Collider Issues, Visual Issues, etc. There are **five** initial bugs. Proceed to fix these bugs and document them. You may want to run the test cases again if any have failed due to these bugs.

Once you are happy that all the written test cases pass and bugs fixed, you can move onto the next section to add new features and unit testing.

**Part 2: New Features & Unit Testing.**

For this part your task is to implement new features into this game using Unit Tests to confirm they are working as expected. You may need to create new Scripts for features you create.

There are three Scripts already created **Ball.cs**, **GameController.cs** and **Tilt.cs** which you may modify to enable your new features. **Tilt.cs** script needs be modified to allow function calls for each movement (TiltHorizontal, TiltVertical) so that movement can be called in Unit Tests.

You must implement the following feature sets and choose at least 3 suitable **Unit Tests** for each feature. You **DO NOT** need to create bug reports or Test Cases for this section.

**Feature Set 1:**

* 1. **Lives System**

You game needs to have a lives system. Lives should be displayed on a UI showing how many are remaining. Player dying or falling off should reduce a life by one. When player runs out of lives player needs to start from **level one again**.

* 1. **Game over**

When player runs out of lives a Game Over UI needs to be displayed. This menu should have buttons for player to restart game or quit.

* 1. **Static Enemy**

There should be something on the platform that the player should avoid. If the ball collides with an enemy, it should call Game Over UI. This enemy does not move.

**Feature Set 2:**

* 1. **Moving Enemies**

Create enemies that the player must avoid. If the enemy is not avoided, then the player must be affected in a negative way.

* 1. **Changing Colour Mechanic**

Implement a feature where the changing colour of ball or end point is part of the gameplay. For example, to complete the level only a green ball can enter a green end point. Use your imagination on how to make colours into this game mechanic. Think about using collectables, destructible and enemies to enhance gameplay. Add extra levels if required.

For every feature in the game, including the initial features, you must implement Unit Tests. Each feature must have at least 3 Unit Tests to ensure behaviour is as expected. On this Labyrinth game you must setup up Unit Tests for both **EditMode** and **PlayMode** to test different aspects of the game. Once you have set up you should have **EditModeScript.cs** and **PlayModeScript.cs** to implement your Unit Tests.

**Part 3: Report.**

Write a report (approximately 500 words) outlining what features were implemented and how Unit Tests were used to ensure correct behaviour. Talk about what tests you implemented and what it does to reduce the risk of bugs arising in the future. Also, what steps were taken to ensure efficient code when implementing your features and test.

***Remember!***

***Components/Assets/Code of the project which are you marked on (based on the mark scheme) must be your own!***

**Marks:**

You will receive an individual grade within the context of you team. Your submission mark will be broken down based on the mark scheme.

**A screenshot of a screen

Description automatically generated**Your submission will be marked as is. The submission mark will then be scaled to achieve your individual mark. You will be required to allocate 100 points (%) between your other team members based on the amount you believe they contributed to the submission; this will be done confidentially. **YOU DO NOT MARK YOURSELF**. These peer evaluation scores will then be used to allocate final marks to individuals, unless the tutor has reason to believe peer evaluations have not been done confidentially or the process has not been wholly above board in any way. In this case the tutor will allocate marks.

*An example of mark allocation:*

On the left is an example of how each individual will complete the Peer Evaluation spreadsheet. A total of 100 points (%) is divided between the other members of the team. Below is an example of how the scale is calculated of a submission worth 67%.

A table with numbers and text

Description automatically generated

Individual marks will not exceed +10 marks for the team score. For example, Paul’s team members constantly gave Paul slightly higher contribution marks (across), so is scaled up. If the team project was scored at 67, and the matrix used to calculate the individual mark for Paul was given as 80%. The individual mark would be capped at 77% (max +10 submission). However, there is no minimum cap. This means no contribution can still lead to a failure.

**Deliverables & Deadlines.**

You are required to submit Three parts for this assignment:

* **Part 1: Test Cases (table) and Bug Reports (Week 11).**
  + All in one document in Word or PDF. Submitted via TurnItIn link on LearningZone.
  + Deadline **Friday 15th December 2023 at 5:00pm**.
* **Part 2: Unity Project with New Features and Unit Tests (Week 11).**
  + One zipped folder, submitted via LearningZone Link.
  + Deadline **Friday 15th December 2023 at 5:00pm**.
* **Part 3: Report. (Week 11)**
  + 500 word approx. One document in Word or PDF. Submitted via TurnItIn link on LearningZone.
  + Deadline **Friday 15th December 2023 at 5:00pm**.
* **Game demonstration (Week 15)**
  + Demonstrate your game in week 15 lab timetable **Friday 12th January 2024**.
* **Peer-Review** documentation by **Monday 15th January 2024**, **(week 16).**

**Please refer to the marks scheme Below**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Mark Scheme** | **0 - 29%** | **30 - 39%** | **40 - 49%** | **50 - 59%** | **60-69%** | **70 – 89%** | **90 - 100%** |
| **Part 1 (15)** |  | | | | | | |
| ***Test Cases***  ***(5)*** | No or limited test cases. | Some attempt at test cases, but clearly not well understood. | Some test cases, but very limited. | Reasonable attempt at test cases, but covers only the basics. | Good range of test cases, with a good level of detail and range. | Very good range of test cases, covering game requirements. | Excellent range of test cases. |
| ***Bug Reports***  ***(5)*** | No or limited bug reports. | Some attempt made at reporting bug, but clearly not well understood. | Limited reporting of bugs, but has provided at least 1 bug report documentation. | Reasonable attempt at reporting bugs, but missing details. At least 3 bug reports. | Good level of detail in the bug reports. At least 3 reports to a high standard. | High level of detail and standard for bug reports. At least 4. | Excellent bug reporting. At least 5. |
| ***Bug Fixes***  ***(5)*** | No attempt to fix bugs, and/or bugs still remain. | Some attempt at fixing bugs, but has been unsuccessful. | Limited attempt at fixing bugs, at least 1 fixed. | Reasonable attempt at fixing bugs. At least 2 fixed. | Good attempt at fixing bugs. At least 3 fixed. | Very good attempt at fixing bugs. At least 4 fixed. | All bugs fixed. |
| **Part 2 (65)** |  | | | | | | |
| *Feature Set 1 (15)* |
| ***Lives system***  ***(5)*** | No or very limited implementation. | Some attempt made but clearly unfinished or misunderstood. | Limited evidence of lives system being implemented. | Appropriate functionality to lives system implemented. | Functioning lives system with UI showing total lives. | Fully developed lives system with UI showing lives points. |  |
| ***Game over***  ***(5)*** | No or very limited implementation. | Some attempt made but clearly unfinished or misunderstood. | Limited evidence of any Game Over UI or mechanic being implemented. | Basic functionality to finished lives implemented. Game Over UI not present or very basic. | Functioning timer with UI showing when it's Game over when lives are run out. | Fully developed Game Over system with UI showing with functioning Restart and quit buttons.  . |
| ***Static enemy***  ***(5)*** | No or very limited implementation. | Some attempt made but clearly unfinished or misunderstood. | Limited evidence of any enemy being implemented. | Basic functionality to enemy implemented. | Reasonable functionality to enemy implemented. | Robust enemy with comprehensive behaviour. | Innovation, going above and beyond the specification |
|  | **0 - 29%** | **30 - 39%** | **40 - 49%** | **50 - 59%** | **60-69%** | **70 – 89%** | **90 - 100%** |
| *Feature Set 2 (50)* |  | | | | | | |
| ***Tilt Script Modification for testing.***  ***(10)*** | No or very limited implementation. | Some attempt made but clearly unfinished or misunderstood. | Limited evidence of any changes being implemented. | Appropriate changes made to Tilt script. | Functioning Tilt mechanism which is separated into different functions for each movement. | Fully sound Tilt mechanism which is separated into different functions for each movement. |  |
| ***Moving Enemies***  ***(5)*** | No or very limited implementation. | Some attempt made but clearly unfinished or misunderstood. | Limited evidence of any moving enemy being implemented. At least 1 somewhat working. | Appropriate functionality to moving enemy implemented. At least 1 fully working. | Functioning Moving enemy with showing Game over UI when hit. At least 2 working. | Fully functioning moving enemies system showing exceptional understanding. At least 2 to a high standard and complexity. |  |
| ***Changing Colour Mechanic***  ***(15)*** | No or very limited implementation. | Some attempt made but clearly unfinished or misunderstood. Changing colour mechanics showing very limited functionality. | Limited evidence of changing colour mechanic being implemented. At least 1 somewhat feature working. | Appropriate and functional changing colour mechanic implemented. At least 2 features working. | Functioning changing colour mechanic with UI enhancing gameplay. At least 2 features implemented to a high standard. | Fully functioning advanced changing colour mechanic showing exceptional understanding. At least 3 colour features implemented to a high standard and enemies used to enhance gameplay. |  |
| ***Unit Tests & C#* Code**  **(20)** | No use of Unit Tests, no use of *C#* Code, | Minor attempt was made to follow Unit Tests and the Code examples in Labs, no comments, and/or partially incomplete code. | Some attempt was made to follow the Unit Tests and Code examples in Labs, no comments, and/or partially incomplete code. | A reasonable use of Unit Testing and reasonable *C#* Code within Unity, but not much understanding outside what was done in Labs, and/or limited commenting. | Overall, a good implementation of Unit Tests and use of *C#* Code within Unity, attempts at innovation, but minor consistency issues. | Excellent implementation of Unit Testing and use of *C#* Code within Unity, with an effective use of comments. Overall high-level of professionalism, with some innovation. | Innovation, going above and beyond the specification |
| **Part 3 (20)** |  | | | | | | |
| **Report/Demonstration**  **(20)** | No or very poor report or demonstration. | No clear vision or incorporation of the report or demonstration.. | A minor attempt however lacks descriptions, critical analysis of implantation and structure. Poor grammar and/or expression problems. | A reasonable attempt at a report with some descriptions, critical analysis of implantation and structure. Some grammar and/or expression problems | Overall, a good attempt at a report with good descriptions, critical analysis of implantation and structure but minor grammar and/or expression problems. | Excellent report with excellent descriptions, critical analysis of implantation and structure with excellent use of grammar, formatting, and clarity of expression. Clear understanding of implementation. | |