Department of Computing and Mathematics

Manchester Metropolitan University

Assessment Cover Sheet 2025/26

Module code and title	6G5Z0026 Games Programming
Assessment set by	Misbahu Zubair
Assessment ID	1CWK100
Assessment weighting	100%
Assessment title	Programming Assignment
Type	Individual
Hand-in deadline	See Moodle
Hand-in format and mechanism	Submit a zipped folder containing your Visual Studio project and a completed implementation checklist.

Learning outcomes being assessed

LO1: Use the advanced features of an object-oriented programming language to express games programming solutions in modern, idiomatic code.

LO2: Use a range of strategies and tools to debug code.

Note: it is your responsibility to make sure that your work is complete and available for marking by the deadline. Make sure that you have followed the submission instructions carefully, and your work is submitted in the correct format, using the correct hand-in mechanism (e.g., Moodle upload). If submitting via Moodle, you are advised to check your work after upload, to make sure it has uploaded properly. If submitting via OneDrive, ensure that your tutors have access to the work. **Do not alter your work after the deadline**. You should make at least one full backup copy of your work.

Penalties for late submission

The timeliness of submissions is strictly monitored and enforced.

All coursework has a late submission window of 7 calendar days, but any work submitted within the late window will be capped at 40%, unless you have an agreed extension. Work submitted after the 7-day late window will be capped at zero unless you have an agreed extension. See 'Assessment Mitigation' below for further information on extensions.

Please note that individual tutors are unable to grant any extensions to assessments.

Assessment Mitigation

If there is a valid reason why you are unable to submit your assessment by the deadline you may apply for Assessment Mitigation. There are two types of mitigation you can apply for via the module area on Moodle (in the 'Assessments' block on the right-hand side of the page):

- **Non-evidenced extension**: does **not** require you to submit evidence. It allows you to add a **short** extension to a deadline. This is not available for event-based assessments such as in-class tests, presentations, interviews, etc. You can apply for this extension during the assessment weeks, and the request must be made **before** the submission deadline. For this assessment, the non-evidenced extension is 2 days.
- **Evidenced extension**: requires you to provide independent evidence of a situation which has impacted you. Allows you to apply for a longer extension and is available for event-based assessment such as in-class test, presentations, interviews, etc. For event-based assessments, the normal outcome is that the assessment will be deferred to the summer reassessment period.

Further information about Assessment Mitigation is available on the dedicated Assessments page.

Personal Learning Plans (PLP)

If you have a Personal Learning Plan (PLP) which states you can negotiate an extended deadline, submit an evidenced extension request on the Moodle site for the module.

Plagiarism

Plagiarism is the unacknowledged representation of another person's work, or use of their ideas, as one's own. Manchester Metropolitan University takes care to detect plagiarism, employs plagiarism detection software, and imposes severe penalties, as outlined in the Student Code of Conduct and Academic integrity and misconduct. Poor referencing or submitting the wrong assignment may still be treated as plagiarism. If in doubt, seek advice from your tutor.

As part of a plagiarism check, you may be asked to attend a meeting with the Module Leader, or another member of the module delivery team, where you will be asked to explain your work (e.g. explain the code in a programming assignment). If you are called to one of these meetings, it is very important that you attend.

Use of generative AI

The use of generative AI is permitted in this assessment, so long as it is used in accordance with the instructions provided in the 'Are you allowed to use AI in assessments?' section of the AI Literacy Rise Study Pack. All submitted work must be your own original content.

If you are unable to upload your work to Moodle

If you have problems submitting your work through Moodle, you can raise a ticket with the Assessment Management Team using the Assist Portal. This must be done **before the published deadline**, else your work will be logged as a late submission. Alternatively, you can save your work into a single zip folder then upload the zip folder to your university OneDrive and submit a Word document to Moodle which includes a link to the folder. **It is your responsibility to make sure you share the OneDrive folder with the Module Leader, or it will not be possible to mark your work.**

Assessment Regulations

For further information see the Undergraduate Assessment Regulations on the Assessments and Results information pages

Formative feedback:	*Students can ask for feedback on work in progress in weekly labs, and week 11's lab session will focus on assignment support.*
Summative feedback:	*Individual written feedback on Moodle, with a breakdown of marks from the marking scheme.*

Department of Computing and Mathematics

Manchester Metropolitan University

1. Introduction

You will be given some code and a range of tasks which can be carried out with the code. The tasks are divided into two sections A and B, you are expected to complete all tasks in Section A and complete any 2 tasks in Section B. The tasks are all independent, so there are no dependencies which will block you from attempting a task if another is not completed.

2. Submission

Your submission should be a zipped file which includes a copy of your Visual Studio project folder and a completed Implementation Checklist. The file should be named using the format **[Firstname Initial][Lastname Initial]_[Student Number}_GP.zip**, for example, **MZ_22551542_GP.zip**. This should be uploaded to the assessment inbox on Moodle. Alternatively, you can upload it to your university OneDrive and submit a Word document to Moodle which includes a link to the file. **It is your responsibility to make sure you share the OneDrive folder with the Module Leader, or it will not be possible to mark your work.**

You should also include your name and student number as comments in the main.cpp file of the project.

3. Scenario

You're working as a programmer for a small indie team, and you've been handed a rough prototype of a space shooter game, put together by another programmer for experimentation (you can download this prototype from the Assessment Information & Submission section on Moodle). You've been given two sets of tasks: the first set has a high priority and needs to be fully completed (maximum of **50 marks**), while the second set has a lower priority, and you're expected to complete any 2 out of the 4 tasks from that list (maximum of **40 marks**). Ensure your code follows best programming practices, with a focus on readability, maintainability, and extensibility (worth up to **10 marks**).

Section A: Complete all tasks below (50 Marks Max)

i. The **WindowManager** class is defined in the **WindowManager.h** file. Refactor the code so that **WindowManager.h** contains the class definition and WindowManager.cpp contains the implementation.
[5 Marks]

ii. Refactor the **Projectile** class to **std::unique_ptr** for managing trail objects instead of raw pointers, ensuring proper memory ownership and automatic cleanup of trail particles.
[5 Marks]

iii. Add a destructor to the **GameManager** class to properly release dynamically allocated memory and clean up resources when the objects are destroyed.
[5 Marks]

iv. All the fields and methods of the **Trail** class are currently declared public. Refactor the class to use the most appropriate access controls (public, private, or protected) for all members.
[5 Marks]

v. Refactor the **Projectile** class to ensure const correctness.
[5 Marks]

vi. The **GameManager** class utilises 4 different methods with similar functionality to check for collisions: **checkCollisionBetweenBaseEnemyAndPlayer**, **checkCollisionBetweenSmartEnemyAndPlayer**, **checkCollisionBetweenProjectileAndBaseEnemy**, and **checkCollisionBetweenProjectileAndSmartEnemy.** Refactor the code to use a single function template instead, name it **CheckCollisionBetweenTwoObjects**.
[10 Marks]

vii. Create a base **Enemy** class that all enemy types can inherit from. It should define a common interface and functionality that specific enemies can then extend. This base class should then be inherited by the **BaseEnemy** class and the **SmartEnemy** class.
[15 Marks]

Section B: Complete two of the four tasks below (40 Marks Max)

i. The **Player** class contains code that handles 3 key player ship behaviour: Movement, Shooting, and Shielding. Refactor this class using the component pattern so that these 3 behaviours can be added to the ship as components; and so adding new behavioural components in the future will require minimal changes to the class.
[20 Marks]

ii. SmartEnemy behaviour changes depending on whether it has an active shield. This is currently handled using if-else logic. Refactor this using the **State Pattern** so that shielded and unshielded behaviours are encapsulated in separate states, making it easier to extend the behaviour in the future.
[20 Marks]

iii. Create a singleton class called **SoundManager** for loading and playing sounds in the game. To test that it works, use it to play the hit.ogg sound whenever an enemy is hit by a projectile.
[20 Marks]

iv. Refactor the **RoomManager** so that each enemy type is spawned by a dedicated instance of an EnemyFactory class. Each factory should be associated with a rectangular spawn area where it spawns new enemies at 3-second intervals until there are no more enemies to spawn.
[20 Marks]

4. Marking Scheme

The marking scheme below outlines the criteria for allocating marks that will be used to assess the quality and completeness of submitted code.

Section A – 50 Marks

i. 0.5 marks per method correctly transferred to the .cpp file.

ii. 5 marks for appropriately using unique pointers.

iii. 5 marks for correctly implementing a destructor.

iv. 2.5 marks for appropriately applying member public access controls, 2.5 marks for appropriately applying member private access controls, and -0.5 marks for every member with inappropriate access control applied up to a maximum of 5 marks.

v. 1 mark for each correct use of const up to a maximum of 5 marks.

vi. 10 marks for correct implementation of the function template.

vii. 5 marks for appropriately declaring the abstract class, 5 marks each for refactoring existing enemy classes.

Section B – 40 Marks

i.

a. Extract functionalities into individual component classes. [10 marks]
b. All components are fully decoupled from the Player class. [10 Marks]

ii.

a. Appropriate state classes are defined for each behaviour. [10 Marks]
b. Dynamic switching between states. [10 Marks]

iii.

a. Only one instance of SoundManager is allowed throughout the game. [5 Marks]
b. Sound playback functionality is implemented appropriately. [10 Marks]
c. Appropriate sound is played every time an enemy is hit. [5 Marks]

iv.

a. Appropriate factory class is defined [5 Marks]
b. Spawner classes are appropriately defined [5 Marks]
c. Spawners are used appropriately in RoomManager [10 Marks]

Code Quality – 10 Marks

Up to 10 marks will be awarded for using good programming practices to write code that is easy to understand, maintain and extend.

5. Support for the Assessment

Help! I don't know where to begin or what to do!

Don't panic! Any assessment can seem daunting at first, especially if it's been a while since you did one or if the concepts of the module are relatively new to you. Your module tutor(s) are always happy to answer any questions that you may have and to talk you through the assessment in more detail.

You may find it helpful to consider the tasks of the assessment one at a time and come up with some initial thoughts and ideas of how you plan to respond to each. This can then be expanded upon to give you a clearer idea of what you need to do. You can share these thoughts and ideas with your tutor(s) at any time during the scheduled teaching activities or during their office hours.

Opportunities for Formative Feedback

As part of the weekly lectures and lab sessions, the module tutor(s) will be able to answer any questions and provide support and guidance over the duration of the module. In the final week of teaching, there will be time specifically dedicated to providing formative feedback on your progress in the assessment.

You are also encouraged to talk regularly with the other students in your class about your plans and progress on the assessment. Sharing of ideas and experiences is a great way to learn from each other and to get a fresh perspective on the work that you are doing.

Final Feedback

You will receive written feedback on your work within 20 working days of submission, in the form of a commented assessment grid identical to those included below, with a short comment on each column, and a general comment covering your piece of work.

There will also be general feedback offered to all students studying the module.

When, where and how can I get support from the module tutor?

Support will be available in timetabled sessions. You are also encouraged to make use of your tutor's office hours for discussion and support. The contact details for your tutor are:

Dr Misbahu Zubair
DB 3.26 | misbahu.zubair@mmu.ac.uk
Office Hours - Mondays 12.00 - 14.00 and Tuesdays 14.00 - 15.00
You can (and should) join the Manchester Met Games Discord using the following invite link:
https://discord.gg/RFpNWND