Manchester Metropolitan University

Department of Computing and Mathematics

ASSESSMENT COVER SHEET 2024/25

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 copy of your Visual Studio project folder and a completed Implementation Checklist. Submission is online, via Moodle See the Submission section in this document for more details.

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 self-certification 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 assessments 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.

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 Misconduct Policy. 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 send your work to the Assessment Management Team using the Contingency Submission Form. Assessment Management will then forward your work to the appropriate person for marking. If you use this submission method, your work must be sent **before the published deadline**, or it 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.*

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 3 out of the 4 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 access this prototype by downloading **Assessment_Starter_Project.zip** from the Assessment Information & Submission section on Moodle). Your task is to continue developing the project. You've been given two sets of tasks: the first set has a high priority and needs to be fully completed, while the second set has a lower priority, and you're expected to complete any 3 out of the 4 tasks from that list.

Section A: Complete all tasks below

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. There is a memory leak in **ExplodedParticles** class because **particle** pointers in the **particles** vector that have lived past their lifetime are not properly deleted. Update the class to fix this issue.
[5 Marks]

iii. All the fields and methods of the **GameManager** class are currently public, however, not all members of the class need to be public. Update the class to apply the right access controls.
[5 Marks]

iv. The draw method in the **GameManager** class has 4 index-based **for-loops** that iterate over 4 vector containers. Refactor this code to use **iterators** instead.
[10 Marks]

v. The **GameManager** class has two methods with similar functionality, **resolveSlowEnemyCollisionsWithPlayer** and **resolveFastEnemyCollisionsWithPlayer.** Refactor the code to use a single template function instead.
[10 Marks]

vi. Add a round-shot feature to the **Player** class. On pressing the **LShift** key, the player should shoot out 36 evenly spaced-out lasers in a circular pattern. The round-shot should have a cool-down time of 20 seconds.
[10 Marks]

vii. When an enemy is hit, the **resolveEnemyCollisionsWithLasers** method in **GameManager.h** creates an instance of **ExplodedParticles**. However, all instances are currently positioned at a fixed point, regardless of where the enemy was hit. Fix this by refactoring the **ExplodedParticles** class constructor to accept a hit enemy's position as a parameter and then use it to position instances.
[10 Marks]

Section B: Complete three of the four tasks below

i. The **Player** class contains conditional code for shooting lasers executed when the **Space key** is pressed. Right now, there are four possible laser shooting options, but this is likely to increase when more power-up options are added. Refactor this code using the state pattern before it gets any more complex.
[15 Marks]

ii. Refactor the **Laser** class by extracting its initialization and movement logic into separate components using the component design pattern.
[15 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.
[15 Marks]

iv. Create a new enemy class that is the same as the **SlowEnemy** except for two differences, it should use the **enemy_type_three.png** sprite and its **rotate()** method should rotate it at the same speed as its movement speed. Using Polymorphism, add pointers to two new instances of the new enemy class to the **slowEnemies** vector in **GameManger.h** when the player reaches level 3.
[15 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:

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

ii. 5 marks for fixing the memory leak.

iii. 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.

iv. 2.5 marks for every correct conversion.

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

vi. 3 marks for implementing the round shot, 3 for implementing the cool-down feature, and 4 for utilizing appropriate OOP practices.

vii. 3 marks for correctly updating the constructor, 3 for utilizing appropriate OOP practices, and 4 for correctly updating the instantiation code.

Section B:

i.

a. Appropriate state classes are defined for each laser shooting option. [3 Marks]
b. Correct Implementation of logic [3 Marks]
c. Dynamic switching between states (6 Marks)
d. Code clarity and extensibility [3 Marks]

ii.

a. Extract functionalities into individual component classes. [6 marks]
b. All components are fully decoupled from the Laser class. [6 Marks]
c. Code clarity and extensibility [3 Marks]

iii.

a. Only one instance of SoundManager is allowed throughout the game. [3 Marks]
b. Idiomatic techniques are used to enforce the singleton pattern. [3 Marks]
c. Sound playback functionality is implemented appropriately. [3 Marks]
d. Appropriate sound is played every time an enemy is hit. [3 Marks]
e. Code clarity and extensibility. [3 Marks]

iv.

a. A new enemy class is defined with the specified features. [3 Marks]

b. Polymorphism and Inheritance are used appropriately within the class. [3 Marks]

c. GameManager uses polymorphism to manage SlowEnemy and its derived class. [6 Marks]

d. Code clarity and extensibility. [3 Marks]

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?

Primarily, support will be available in the timetabled lectures and labs, so your attendance at these is vital. 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
Office Hours - Tuesday 10.00 - 12.00 and Wednesday 11.00 - 12.00
01612471526
misbahu.zubair@mmu.ac.uk

You can (and should) join the Manchester Met Games Discord using the following invite link:
https://discord.gg/RFpNWND