University of Central Lancashire UCLan **Department of Computer Science**	**UCLan Coursework Assessment Brief**	2022-2023
	Module Title: Games Development 2	
	Module Code: CO3301	Level 6
	Game Entities	This assessment is worth 50% of the overall module mark

RELEASE DATES AND HAND IN DEADLINE

Assessment release date: 26th January 2023
Assessment deadline date and time: **Saturday 2nd April 2023, before 20:00**
Upload to Google Drive or similar and email a link to lsnoel@uclan.ac.uk.
Ensure the files are publicly accessible or it might be counted as late.

Please note that this is the final time you can submit – not the time to submit!
Your feedback and mark for this assessment will be provided 15 working days later.

THE BRIEF / INSTRUCTIONS

OVERVIEW

You are given a shell of a game consisting of three boats in an ocean scene. Basic entity and rendering classes are provided. To **pass the assignment (40%-45%)** you must:

- Implement simple entity behaviour based on a state machine: each boat tries to destroy the others with missiles.
  - The boat entities patrol and target. You will also create and move missile entities.
- Write a short report discussing your implementation and comparing it with an alternative approach

To get further marks **(up to 85%)** you must:

- Enhance the user interface
- Implement more complex game behaviour
- Use XML file import to set up the scene
- Extend the report to cover your additional implementation
- All details are below

Marks beyond this will be awarded for the use of more advanced techniques of your own choosing relevant to the module material. *Changes should be made to game logic and architecting. Marks are not offered for changes to the rendering systems.*

This is an individual project and no group work is permitted.

Starting Project Details

This assignment is concerned with high-level game architecture rather than lower-level details, so there is a working base project provided as a starting point. It contains a simple graphics engine, entity manager, messenger and game loop - similar to projects used in the labs. The project can be downloaded from the CO3301 blackboard page.

There are two class hierarchies for entities and entity templates. The entity templates are loaded and the entities instantiated in the Scene class constructor. An entity manager handles this process and provides helper functions to look up templates or entities by name / UID. An empty entity class for missiles is provided.

The game runs on a simple timed game loop. First the scene is rendered, and then each entity's Update function is called to perform entity behaviour. Initially the entity behaviour is some basic undirected movement.

Assignment Tasks – To Achieve a Pass

Basic Requirements

- Implement boat & missile behaviour:
  - First write the boat name next to each boat (*note: already done in sample code)*. Look at the Scene Render function
  - Next you need to support a new set of states and behaviours by adding new states and member variables to the Boat class (Boat.cpp/.h) and rewriting the Boat Update function. New message types will be needed in Messenger.cpp/.h. You should remove or comment out the example code to make way for this new behaviour:

  - A boat should have four states: Inactive, Patrol, Aim and Evade. All boats should start in the Inactive state
  - Inactive state:
    - The boat remains stationary
    - If a "Start" message is received, it enters the Patrol state
  - Patrol state:
    - The boat patrols between random points all around the game area. The boats must face the direction they are moving for a pass.
    - Boat A can “see” boat B if it is within 70 degrees of A’s facing direction and they are closer than 140 units from each other. In patrol state, check if any other boat can be seen and if so, enter Aim state
  - Aim state:
    - The boat stops moving while it counts down a timer (initialised at two seconds).
    - When the counter reaches zero the boat fires (creates) a missile at the other boat, then enters the Evade state
    - For a basic pass the missile will move straight towards the point where the other boat was when it was fired. So it the other boat continues moving, the missile will pass behind it.
    - For a basic pass, the boat’s gun does not need to face correctly
  - Evade state:
    - On entering the state, the boat selects a random nearby position within 80 units of its current location.
    - The boat moves at double speed towards this point.
    - When the boat reaches the point, it enters the Patrol state
  - If, at any time, a "Hit" message is received, the boat loses 20 HP
    - If the boat’s HP reaches zero or less, it is destroyed
    - A member variable for HP is already initialised for the boats
  - If, at any time, a "Stop" message is received, a boat should immediately enter the Inactive state
  - The user can press the keys '1' and '2' to start and stop the game:
    - Start game: A "Start" message is sent to all the boats
    - Stop game: A "Stop" message is sent to all boats

- Fill in the missile entity class with this simple behaviour:
  - A missile has no states. It travels in a fixed direction (at sea level for a basic pass) while it counts down a timer (initialised at four seconds)
  - When the countdown reaches zero, the missile is destroyed
  - If a missile passes close to any other boat (within a radius of 15), a “Hit” message is sent to that boat and the missile is destroyed

- ***Additional technical guidance is given at the top of the file Boat.cpp. Ensure you read this information before commencing. Also example code is given throughout Boat.cpp and Scene.cpp to show how existing features work. Ensure to read it.***

Discussion & Evaluation

- Also to reach the pass level, write a short report explaining your behaviour implementation and evaluating it against an alternative method:
    - Explain the processes used in your patrol and targeting implementation
    - Provide a state transition diagram for the boat entity behaviour, clearly noting the event that causes each transition. Support this diagram with a very brief discussion of the use of states and messages
    - Discuss how you would implement the same behaviour with a single 'SceneUpdate' function rather than separate 'EntityUpdate' functions. Contrast this with your implementation, considering the respective benefits and drawbacks

Assignment Tasks – For Further Marks

User Interface Enhancements

- Add further information next to the boat's name:
    - Show the boat's current HP and current state
    - Show the number of missiles the boat has fired
    - Use the '0' key to toggle between this extended information and just the name
- Support camera picking with the mouse. In the first instance allow the user to click on a boat to make it instantly switch to evade state. For further marks support other behaviour, e.g. click on a boat to select it (displaying its selection status in some way), and then click on a point to send it there. Document your picking UI so the tutor can properly credit it
- Provide chase cameras for each boat, selectable with the number keys

Advanced Game Behaviour

- Utilise the existing entity template data to provide individuality to each boat type (max speed, t turn speed etc.). More credit for more elements used
- Add more interesting boat destruction behaviour
- Have the missiles follow a parabola rather than flying at sea level:
    - Give the missile an initial y velocity that is affected by gravity each frame.
    - The x and z velocities will determine the direction of movement.
    - The choice of y velocity will affect where the missile strikes. Extra credit for doing the correct mathematics to target the other boat correctly when using a parabola and for getting the gun to point in the right direction.
    - The timeout for the missile can be removed if this feature is added.
- Use "leading" for targeting the other boats:
    - The missiles will target a point just far enough *in front* of the other boat, that the missile will strike accurately if the other boat continues on its current course.
    - This feature can be added with or without the implementation of the parabola feature
- There are icebergs in the ocean. Move three into the main game area and use line-of-sight calculations so the boats cannot see each other if there is an iceberg between them
    - Use an axis-aligned cuboid as the bounding volume for the icebergs
- Have two teams of three boats rather than three individual boats. They do not shoot at teammates. Additionally:
    - When a boat is hit, have it send a "Help" message to its teammates, which has a 50% chance of putting them into Aim state. Some flexibility in game mechanic will be allowed here
- Implement an ammunition game mechanic. Each boat starts with 10 missiles. Add a loading station at the edge of the game space from which new missiles can be picked up when they run out.
    - You will need to add new entities, states, and maybe messages to support this behaviour. Extra credit given for updating FSM effectively to deal with the situation.

XML Scene Setup

- The initial entity and scenery locations are set up in code (the Scene class constructor). Rework this function so that it initialises the scene elements from data in an XML file
    - The XML element naming and structure is up to you
    - The XML parsing code will need to be adapted to deal with the different entity types
- Aim for getting all data from XML, such as turn speeds, or loading station location

SUBMISSION DETAILS

Deliverables

See notes at the top of this document for submission method.

You must provide:

- Executable program of your solution
- All the source and project files required to build the executable
- Discussion / evaluation report
- All files should be sensibly named and in working order

GRADING CRITERIA

Marks will be awarded based on the learning outcomes highlighted above.

To obtain **40-45%** and a **pass,** you must:

1. Implement the **basic requirements** described above
    - Marks are awarded based on the clarity and robustness of the solution
2. Write a report containing the **discussion and evaluation** as described above
    - Marks will be awarded based on the articulation and depth of analysis presented in the document

Up to a further **7%** will be awarded for:

- **User interface enhancements**, as described above

Up to a further **26%** will be awarded for:

- **Advanced entity behaviour**, as described above
    - Marks are based on the challenge of each task and the quality of the given solution

Up to a further 7**%** will be awarded for:

- **XML scene setup**, as described above

Further marks beyond this will be awarded for:

- **Addition of advanced work** relevant to the module material to extend the project. In particular credit will be given for a more complex set of states and behaviours, or for embedding the application in a Windows-based GUI and adding tools to manipulate the game settings. You **must** document any such additional developments or they may not gain credit

A proportion of all the implementation marks will be given over to the readability and maintainability of your source code. Higher marks are reserved for solutions that have been tweaked and polished to achieve better realism and interest.

Note: Once you have satisfied the basic requirements, it is not necessary to address the remaining tasks in the order given. You may attempt any of the additional tasks in any order you wish.

LEARNING OUTCOMES

- Critically evaluate approaches towards games development from design to implementation.
- Analyse and evaluate game-specific algorithms in terms of their theoretical underpinnings.
- Design and implement game architectures from the ground up.

HELP AND SUPPORT

- Support will be provided via Microsoft Teams and email. You will also have the opportunity to ask questions during lectures / labs. You may request a one to one meeting with a tutor during their office hours (as published on Starfish).
- For support with using library resources, please contact our subject librarian subjectlibrarians@uclan.ac.uk. You will find links to lots of useful resources in the My Library tab on Blackboard.
- If you have not yet made the university aware of any disability, specific learning difficulty, long-term health or mental health condition, please let us know. The Inclusive Support team will then contact you to discuss reasonable adjustments and support relating to any disability. For more information, visit the Inclusive Support site.
- To access mental health and wellbeing support, please complete our online referral form. Alternatively, you can email wellbeing@uclan.ac.uk, call 01772 893020 or visit our UCLan Wellbeing Service pages for more information.
- If you have any other query or require further support you can contact The Student Support Centre. Speak with us for advice on accessing all the University services as well as the Library services. Whatever your query, our expert staff will be able to help and support you. For more information, how to contact us and our opening hours visit Student Support Centre.
- If you have any valid mitigating circumstances that mean you cannot meet an assessment submission deadline and you wish to request an extension, you will need to apply online prior to the deadline.

Version: 2
Updated 01/09/2022