University of Central Lancashire UCLan	# UCLan Coursework Assessment Brief		2021/2022
School of Psychology and Computer Science	Module Title: Games Development 1	Module Code: CO2301	Level 5
	Multiplayer Assignment	This assessment is worth 30% of the overall module mark	

THE BRIEF/INSTRUCTIONS

You are to create a multiplayer enabled assault course prototype (Think of the Indie game 'Fall Guys'). You should only construct the course using placeholder assets (e.g. BSP Brushes) to enable you to focus on the programming. Your course should implement the following requirements/features:

40% Requirements:

1. Assault Course blocked out using BSP containing at least 4 obstacles.
2. Contains a C++ 'Spinning Bar' Actor/obstacle and movement is replicated (doesn't need to be simulated) See Appendix A1 for further details.
3. Game allows a client game instance to connect to another game instance over a local network (game instances can be launched separately from the command line - evidenced by video demo).
4. Code controlling authoritative rotation of the spinning bar should only execute on the server.
5. Players are reset back onto the course if they fall or are knocked off the course (can be reset back at the start of the level).
6. Players are based on a C++ Character class and movement is replicated (can use built in replication provided by the CharacterMovementComponent).

50% Requirements:

7. Game contains a 'Fixed Door' Actor/obstacle that replicates physics from server to clients. See Appendix A2 for further details.
8. When players fall off the map, they are reset back to a point close to the obstacle that knocked them off the course.
9. Spinning Bar actor rotation is simulated locally on client game instances and uses a replicated property to set rotation speed. (if rotation speed is changed on the server, it should replicate to allow the client game instances to simulate with the new rotation speed).
10. Player is able to 'dive' by clicking the right mouse button, launching the character forward (Remote Procedure Call to server, calling LaunchCharacter).
11. Course contains a revolving doors (or floor) obstacle with replicated physics (constrained to rotate only on one axis) See Appendix A3 for further details.
12. Level starts with flyby of the course using a 'Level Sequence' (camera position replicated to clients from server, player movement is disabled during flyby)

60% Requirements:

13. Players are able to 'Push' each other using the Characters LaunchCharacter method (Triggered via a Remote Procedure Call from client to the server).
14. 'Push' functionality should only launch opposing players who are in front of the player and are within a range of 100cm (check should be implemented using a raycast/line trace)
15. Contains a C++ finish zone actor that transports all players to an end level using Server Travel when 3 players have entered the zone (should use a trigger to detect entering finish zone). See Appendix A4
16. Finish zone actor only detects OnOverlap events on the server
17. Game contains a Game Instance class and players are able to host and join games from inside the game (can use console commands)
18. Game uses a client side remote procedure call to play a sound only on the local client when the locally controlled player crosses the line (sound is not played when other connected players cross the line)

70% Requirements:

19. Contains a C++ sliding door Actor/obstacle that is replicated only once when the client logs into the level. It is still simulated locally on the client using Multicast remote procedure calls to update local properties when appropriate. (so it still runs smoothly with a NetUpdateFrequency of 1 replication per second). See Appendix A5 for further details.
20. Course level has a custom C++ GameState and C++ Game Mode class that keeps track of the number of connected players in the level and the number of players who have crossed the finish line.
21. Game contains a holding level where players are placed until the target number of players have joined the game (4). (Level should have it's own GameMode that keeps track of logged in players by overriding AGameMode::PostLogin).
22. When holding level is full (at least 4 players), players are moved together to the assault course using ServerTravel.

23. Finish Zone actor contains two UNiagaraComponents that are activated on all clients when the first player enters the zone (you should use a multicast remote procedure call to do this).

LEARNING OUTCOMES ASSESSED

3.	Apply a systematic approach to games development from specification to implementation.
4.	Develop software using game-specific tools and environments

ASSESSMENT CRITERIA

for 40% you must Implement:
All of the 40% Criteria
Code is reasonably commented

for 50% you must implement (in addition to the above):
At least 5 50% Requirements
Appropriate Coding style/naming conventions

for 60% you must implement (in addition to the above):
At least 5 60% Requirements

for 70% you must implement (in addition to the above):
At least 4 70% Requirements

For 85%+ you must implement (in addition to the above):
<u>ALL</u> 40,50,60% and 70% Requirements.
For a high mark in the 85%+ level we will be looking to see you build on the topics covered in the lectures, demonstrate self directed study and implement functionality that extends beyond the lecture materials. Some possible examples are:

a. Players are able to host/join games from within the game through UI Widgets (without using the console)
b. The number of people who have crossed the line could be visible to all clients through a UI Widget
c. Appropriate cheat protection / client input validation implemented.
d. Players knocked to the ground by the spinning bar.

Note*: If you are unable to complete a lower level requirement, but are able to complete a higher level requirement in it's place, you are recommended to do so, as we <u>may</u> be able to take this into account when marking your work.*

PREPARATION FOR THE ASSESSMENT
Before completing the assessment it will help if you are familiar with, and have completed the weekly programming tasks on Blackboard. These will be a good resource to refer to when programming your final submission. The exercises are designed to equip you with the tools and techniques required to complete the assignment.

RELEASE DATES AND HAND IN DEADLINE
Assessment Release date: **Friday 28/02/2022** Assessment Deadline Date and time: Friday **29/04/2022**

Please note that this is the <u>latest</u> time you can submit – not <u>the</u> time to submit!
Your feedback and mark for this assessment will be provided on **20/05/22**

SUBMISSION DETAILS
Submission uses the same process as assignment 1. A video guide to assignment submission can be found under the 'Assignments' section of the module space on Blackboard.

1. Before submission make sure you have made a backup of your project.
2. Delete the following folders from your project (These can all be recreated by UE4 and Visual Studio):
   - Intermediate
   - Saved
   - .vs
   - Binaries
3. <u>Add the completed assignment coversheet into your project folder. (complete the self assessment)</u>
4. Zip the remaining files in your project into a single .zip file and submit via the 'Project Files' link on Blackboard.

5. You must record a video demo of your game of between 5 and 10 minutes in length showcasing how it has met the requirements you have implemented. This should be submitted via the 'submit video demo' link on Blackboard. A youtube link is acceptable if the video file size is too large.
6. You should submit a game build as a separate .zip file to the link provided on Blackboard
7. You should submit a .zip file containing only your source files. Also add the completed assignment coversheet to this file.

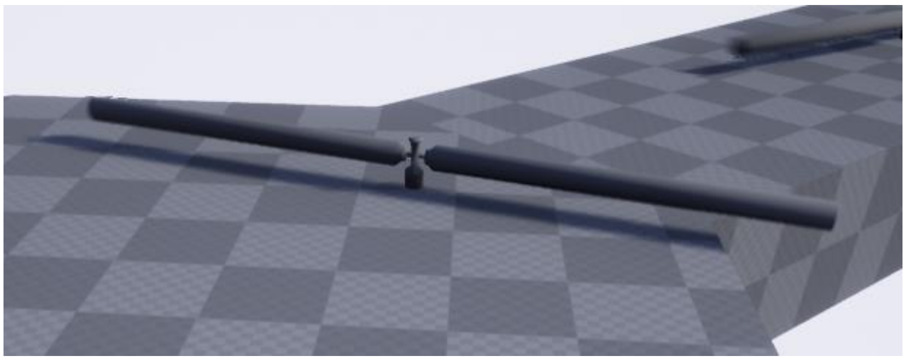
HELP AND SUPPORT

- Academic Support for this module can be gained by posting to the 'Assignment Queries' channel on the modules Microsoft Teams Space. You may also email your tutors for support, or to request a teams meeting, but it is likely you will get a quicker response through the MS Teams channel.
- For support with using library resources, please contact Bob Frost, RSFrost@uclan.ac.uk or 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 complete a Disclosure Form. The Inclusive Support team will then contact 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 <i>, The Student Information and 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 Information and 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.

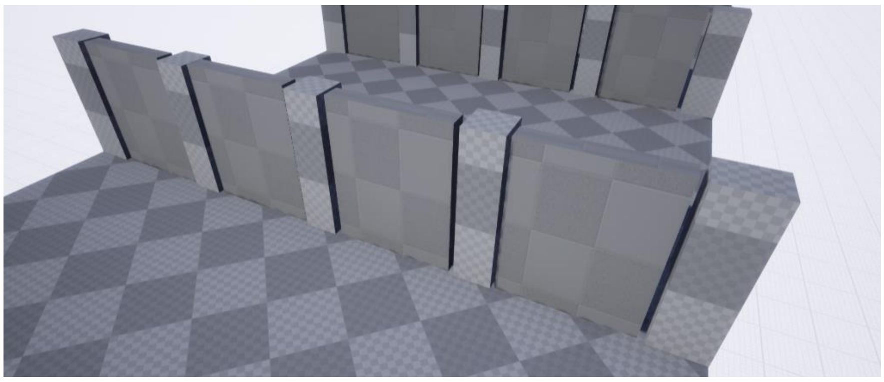
Disclaimer: The information provided in this assessment brief is correct at time of publication. In the unlikely event that any changes are deemed necessary, they will be communicated clearly via e-mail and a new version of this assessment brief will be circulated.	Version: 2

Appendix – Obstacle Descriptions

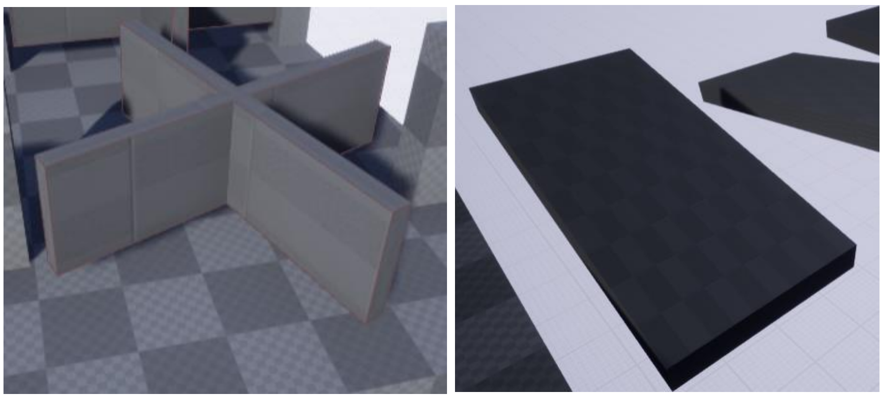
A1 - A ‘Spinning Bar’ Actor is a Static Mesh actor rotates in a single direction at a given rotation rate. The actor can be rotated using the actors AddActorWorldRotation method. (should be implemented in C++)



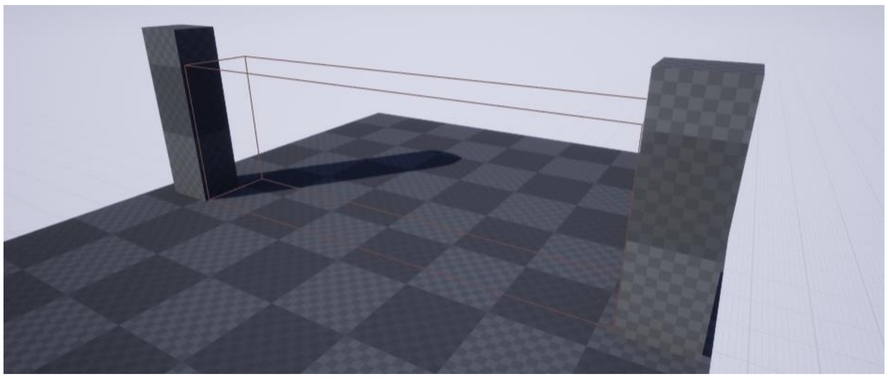
A2 - A ‘Fixed Door’ Actor is a Static Mesh actor that is shaped like a door. They should simulate physics and have the result replicate to connected clients. They can be knocked over by the player running through them. (You do not need to implement these in C++).



A3 – Revolving Door/Floor actors are static mesh actors that simulate physics but are constrained to rotated only on one axis. They can be moved by the player and the result should replicate to connected clients. (You do not need to implement these in C++



A4 - A Finish Zone is an actor that contains a box trigger to keep track of the players who have crossed the finish line (Should be implemented in C++)



A5 – A Sliding Door Actor is a static mesh actor that slides up and down through the floor. It should slide at a fixed speed. When the door reaches the top and bottom it should wait for a random amount of time (between 1 and 4 seconds) before sliding back in the opposite direction. Hint: You can you the FMath::VInterpContrantTo to slide the doors up and down in the actors Tick function. The doors should be simulated on the client and control over when the doors should be sliding should be decided by the server and sent to clients via a multicast remote procedure call. (Should be implemented in C++).