ASSESSMENT AND INTERNAL VERIFICATION FRONT SHEET (Individual Criteria)

(Note : This version is to be used only for assignments uploaded via Classter)

Course Title	**BSc Year 2 - Multimedia Software Development**		**Lecturer Name & Surname**	**Owen Sacco**	
Unit Number & Title	ITCGR-506-2003 - Programming for Computer Graphics				
Assignment Number, Title / Type	Procedural Content Generation in Games – Home Assignment				
Date Set	18/12/2020	**Deadline Date**	05/02/2021		
Student Name	George Nicholas Sammut	**ID Number**	307501L	**Class / Group**	MSD 6.2A

Assessment Criteria	Maximum Mark
AA1: Produce and develop code that generates meshes	7
AA2: Produce and develop code that generates shapes such as cubes, planes and pyramids	7
SE1: Generate meshes and shapes that can be generated through code in games	10
AA3: Produce and develop code for generating trees and vegetation	7
AA4: Produce and develop code for generating natural elements (for example but not limited to: erosion, rain, wind, clouds, fog, sky etc.)	7
SE2: Generate random terrains	10
AA5: Produce and develop code that generates random levels	7
SE3: Generate random levels in games	10
Total Mark	65

Notes to Students:

- This assignment brief has been approved and released by the Internal Verifier through Classter.
- Assessment marks and feedback by the lecturer will be available online via Classter (Http://mcast.classter.com) following release by the Internal Verifier
- Students submitting their assignment on Moodle/Unicheck will be requested to confirm online the following statements:

Student's declaration prior to handing-in of assignment

- ❖ I certify that the work submitted for this assignment is my own and that I have read and understood the respective Plagiarism Policy

Student's declaration on assessment special arrangements

- ❖ I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit.
- ❖ I declare that I refused the special support offered by the Institute.

Assignment: Procedural Content Generation in Games

Home Assignment

Assignment Guidelines

Read the following instructions carefully:

- The assignment coversheet should be the first sheet in your assignment. Moreover, the coversheet should be fully completed with all the necessary details.
- You are required to use Unity and C# for this assignment. All tasks should be developed through code and not created through any designer tools unless instructed otherwise.
- All text/code must be properly referenced. In the absence of proper referencing, the assignment will be regarded as plagiarised.
- **Copying is strictly prohibited and will be penalised** in line with the College's disciplinary procedures.
- The deadline to submit all deliverables is **05/02/2021**
- You need to submit all your assignment deliverables (as explained in the assignment brief) via the links on Moodle.
- You need to commit all your C# scripts and project files to a Git repository. It is important to commit your code in stages to show your progress whilst working on this assignment. Include clear comments and notes to explain key concepts in your code.
- You are required to record a (video) demonstration of all your deliverables and upload your recording to any streaming service of your choice such as Microsoft Stream or YouTube. Ensure that you set the correct permissions and share it only with me. Kindly limit the recording to not more than 20 minutes.

Task 1 – Mesh Generation

Task 1.1 – [AA1] Produce and develop code that generates meshes – 7 marks

In this task, you are requested to programmatically develop a helper class that procedurally generates meshes. This helper class should generate meshes out of triangles. You are to write clean code and optimise the code in the most efficient manner. Once the mesh generator is complete, you are requested to test your code by programmatically develop a script that generates triangles using the helper class mesh generator and display these triangles in Unity. Your script should be able to assign materials to the triangles programmatically.

Task 1.2 - [AA2] Produce and develop code that generates shapes such as cubes, planes and pyramids – 7 marks

In this task, you are requested to programmatically develop primitives by using the mesh generator developed in Task 1.1. You are requested to at least develop a cube, a plane and a pyramid within the same scene. Your code should assign materials to the primitives programmatically.

Task 1.3 – [SE1] Generate meshes and shapes that can be generated through code in games – 10 marks

In this task, you are requested to programmatically develop a single level, single player maze game out of primitives by using the mesh builder in Task 1.1 and the primitives developed in Task 1.2. The maze should be constructed programmatically out of the primitives developed in Task 1.2 and the single player controller can be any primitive constructed in Task 1.2. The maze should have a random starting point location and a random finishing point location. The game ends either when the player quits or when the player reaches the finishing point marker. The starting point marker and the finish point marker could be represented by any primitive developed in Task 1.2. All materials should be assigned to the objects programmatically. It is important that the player controller, the starting point marker and the finish point marker stand out and clearly distinguishable from the maze walls.

Task 2 – Terrain Generation

Task 2.1 - [AA3] Produce and develop code for generating trees and vegetation – 7 marks

In this task, you are requested to first develop a random landscape programmatically with different heights as mountains along the landscape. You are then requested to programmatically add vegetation and trees at random locations along the landscape.

Task 2.2 - [AA4] Produce and develop code for generating natural elements – 7 marks

In this task, you are requested to programmatically generate random natural elements (such as but not limited to water, river, rain, wind, clouds, fog, sky etc.) to the landscape developed in Task 2.1. You are requested to programmatically generate at least 3 natural elements of your choice at random locations along the landscape.

Task 2.3 - [SE2] Generate random terrains – 10 marks

In this task, you are requested to add final touches (such as more mountains, more trees and/or vegetation, and more natural elements) to your landscape developed in Task 2.1 and Task 2.2. Programmatically add a random path to your landscape and turn your landscape into an exploration game. Programmatically add a player controller that could navigate and explore the terrain. The player controller could be a primitive developed in Task 1. The game ends when the player quits the game. The player should start at any random position within the landscape.

Task 3 – Level Generation

Task 3.1 - [AA5] Produce and develop code that generates random levels – 7 marks

In this task, you are requested to programmatically generate levels consisting of randomly generated car racing tracks. Each level should contain a different randomly generated car racing track. Develop at least 3 levels – i.e. 3 different programmatically randomly generated car racing tracks. Each racing track should contain a race track barrier on each side (edge) of the track, a middle white road (line) marker, and a road on each side of the middle white road (line) marker. All materials should be assigned to the objects programmatically.

Task 3.2 - [SE3] Generate random levels in games – 10 marks

In this task, you are requested to turn the generated levels developed in Task 3.1 into a playable car racing game. You are requested to programmatically add a player controller to the game. The player controller can be any car controller freely provided by the Unity Asset store. A random start position should be programmatically added to each track and the player controller should be positioned at this starting position. A level ends when the player races one lap until the player controller reaches once again the start position. Once a level ends, the next level loads. The game finishes once all 3 levels have been played (or when the player quits the game). You are not required to add any opponents within the game.

Grading Scheme

Task	Assessment Criteria	Marks Awarded	Task Marks
1.1	• Mesh generator – 4 marks • Triangle generator – 2 marks • Assign materials programmatically – 1 mark		7
1.2	• Cube generator – 2 marks • Plane generator – 2 marks • Pyramid generator – 2 marks • Assign materials programmatically – 1 mark		7
1.3	• Maze generator – 5 marks • Player controller – 2 marks • Starting point and finish point markers – 2 marks • Assign materials programmatically – 1 mark		10
2.1	• Landscape generator – 2 marks • Vegetation generator – 2 marks • Tree generator – 2 marks • Randomly locating vegetation and trees along the landscape – 1 mark		7
2.2	• 3 natural elements generator (2 marks each) – 6 marks • Randomly locating natural elements along the landscape – 1 mark		7
2.3	• Finishing touches – 4 marks • Random path generator – 3 marks • Exploration game – 1 mark • Player controller – 2 marks		10
3.1	• 3 randomly generated car racing tracks (2 marks each) – 6 marks		7

	• Assign materials programmatically – 1 mark		
3.2	• Level generator – 3 marks • Car controller – 2 marks • Random start position of car controller – 2 marks • Level management (transitioning from one level to another) – 3 marks		10
Total			65