MODULE NAME:	MODULE CODE:
GAME DEVELOPMENT 3B	GADE7322

ASSESSMENT TYPE: POE (PAPER)

TOTAL MARK ALLOCATION: 300 MARKS

TOTAL HOURS: A MINIMUM OF 45 HOURS IS RECOMMENDED TO COMPLETE THIS ASSESSMENT

By submitting this assignment, you acknowledge that you have read and understood all the rules as per the terms in the registration contract, in particular the assignment and assessment rules in The IIE Assessment Strategy and Policy (IIE009), the intellectual integrity and plagiarism rules in the Intellectual Integrity Policy (IIE023), as well as any rules and regulations published in the student portal.

INSTRUCTIONS:

1. ***Make a copy of your assignment before submitting it.***
2. *Assignments must be typed unless otherwise specified.*
3. *All work must be adequately and correctly referenced.*
4. *Begin each section on a new page.*
5. *Follow all instructions on the PoE cover sheet.*
6. *This assignment requires you to either work in a* ***group of two*** *members (to practice pair programming), or to work individually.*

BACKGROUND

For this PoE, you will be creating a tower defence game. Tower defence is a sub-genre of strategy games where the goal is to obstruct enemy waves from reaching exits or the player's territories by placing defensive structures along enemy paths.

You will be creating a tower defence with multiple pathways leading towards a central tower. Enemies will endlessly spawn and move towards and attack your tower. Players will have the ability to place defensive structures to stop the enemies from progressing towards and destroying the player's tower. If the central tower is destroyed, it is game over.

You will be required to use procedural generation techniques to complete various aspects of this brief. It is important to have a good understanding of the content covered in class, but you will also be required to do your own research.

In this assessment, you have the option to work in pairs (strictly two members), or to work individually. If you are working in pairs, the idea is not to divide up work, but to practice pair programming.

Pair Programming is a technique where two developers work together on the same task. In this one person writes the code (driver) and the other person reviews each line and provides feedback (navigator). This approach differs from traditional solo programming and offers various advantages (https://dev.to/documatic/pair-programming-best-practices-and-tools-154j).

This PoE will be structured as follows:

- **Part 1**
  In Part 1, you will be procedurally generating the terrain your game will take place on. The terrain will change every time you start a new game. You will also implement the base mechanics. By the end of this part, you should be able to play a very basic version of the game.

- **Part 2**

  In this part, you will be elaborating on all aspects of Part 1 by adding more enemy and defender types. You will also be required to create a more sophisticated procedural enemy spawner that scales difficulty based on player skill level and play style.

- **Part 3**

  For the final part of the POE, you will allow the player to upgrade defenders and the tower and add custom shaders and visual effects. You will also be adding a procedurally generated feature of your own design to your game.

This POE intentionally leaves some flexibility in terms of how you implement certain features of the game. It is important that these features are appropriately complex. You are therefore encouraged to check in with your navigator in class before each hand-in to make sure your project matches the required module standard.

Your final mark is calculated and captured as follows:

- Part 1 – 25%
- Part 2 – 30%
- POE – 35%
- ICE – 10%

INSTRUCTIONS

PLEASE NOTE: ANY COPYING OF CODE FROM ANOTHER STUDENT, AN ONLINE RESOURCE, A TUORIAL, A TEXBOOK, OR ANY OTHER SOURCE THAT IS NOT YOUR OWN WORK COUNTS AS PLAGIARISM.

You must work from a GitHub repository and include your lecturer as a collaborator. You will include the link to the repository when you submit each Part of the PoE.

PART 1 — PROCEDURAL TERRAIN GENERATION (Marks: 100)

For the first part of this POE, you will be generating the terrain for your game. This terrain must be procedurally generated, meaning it should be created with code and vary every time a new game starts.

Additionally, you must introduce one type of enemy that continuously spawns, moves towards, and attacks your base (tower) and defenders. You will also add one type of defender to protect your base. When your tower is destroyed, it is game over.

1. Procedurally generating your terrain and placing your Tower [20]

In class, you would have been introduced to various algorithms for procedurally generating the content. Using that knowledge and doing your own research, you must generate a 3D terrain on which your game will take place.

It must meet the following minimum requirements:

- The mesh for the terrain must be generated at runtime
- The terrain must be different every time you start a new game
- The mesh being generated must have multiple pathways that the enemies will walk on towards a central point in the terrain where your tower will be placed
- There must be at least three (3) pathways that lead to the tower
- The tower must have a set amount of health, and the game is over when it is depleted and
- The tower must attack enemies automatically.

2. Adding and placing defenders [20]

The player must have the ability to place defenders on the terrain. This gives the player the opportunity to be strategic in defending the tower against enemy attacks.

For this part of the POE, you will also be creating your first defender type. Two additional defender types will be created in Part 2.

	It is up to you to decide how your defenders behave, but it is important to create a balanced experience for the player. The following minimum requirements must be met: • There must be a set number of predetermined locations that defenders can be placed. These locations are based on the generated terrain • You should not be able to place defenders in the path that leads towards the tower. This should be kept clear for enemies • Defenders attack automatically • Defenders can also be attacked by enemies and should have a set amount of health	
3.	**Creating and spawning enemies**	[20]
	Enemies must spawn at set locations on the map (based on how it was generated) and move down the path towards the tower. For this part of the POE, you will create your first enemy type. In Part 2, two additional enemy types will be added and procedural enemy waves that will create a more balanced experience as the player becomes stronger. For now, your enemy spawner must meet all the following requirements: • Spawn locations must be based on the generated terrain • Enemies must spawn from a spawn location at a set interval • Enemies must gradually move towards the tower and have the ability to attack the tower and defenders. It is up to you to decide what kind of abilities your enemies have and • Enemies have a set amount of health and can be destroyed by defenders and the tower.	
4.	**Complete game loop**	[10]
	You must add any necessary mechanics to create a complete playable game loop. Examples of things to consider: • Does the player generate resources that allow them to buy new defenders for defeating enemies?	

- Do you have a separate mechanic where the player mines for resources?
- What happens when the player has depleted resources and can’t buy new defenders?

It is up to you to decide how you wish to complete the game loop, as long as the game ends when the tower’s health bar is reduced to 0 and you create an experience that feels balanced.

5. User Interface and User Experience [10]

Having completed the game loop, it is important to ensure that all the necessary UI elements are present to create a positive experience for the player. It should be clear to the player what is going on at all times.

Examples of things to consider in terms of the UI:

- Is it clear to the player how much health each defender, enemy, and tower has left?
- Is it clear when the game is over?
- Can I see how much resources I have left to add new defenders?
- Can I see the cost to add a new defender?
- Is it clear when a defender, enemy, or the tower is being damaged?
- Can I pause the game and restart it at any point? Etc.

6. Suitable Complexity [10]

Your project will be assessed on how well it meets the required level of complexity for this module.

You get some flexibility in choosing how you implement parts of this project, but this means:

- Your project should reflect an appropriate level of difficulty and proficiency.
- Your solutions should show innovative thinking and effective problem-solving strategies.
- Your work should present original ideas or approaches, displaying creativity in tackling the project’s challenges.

7. Presentation Video [10]

Finally, create a presentation video to demonstrate your project.

These are the minimum requirements, and your video must demonstrate the following:

- The game you have created so far
- focus on demonstrating specific requirements of the brief
- Discuss the most relevant code you have written for this part of the PoE to demonstrate how you solved specific issues.

The video should be roughly 5 minutes long, but no longer than 6 minutes.

Submission

You must submit the following to Arc:

- A link to the GitHub repository that contains your project files
- Your video presentation.

Assessment Sheet (Marking Rubric) – Part 1

MODULE NAME:	**MODULE CODE:**
GAME DEVELOPMENT 3B	**GADE7322**

STUDENT NAME:	
STUDENT NUMBER:	

PART 1				
Marking Criteria	***Poor***	***Developing***	***Good***	***Excellent***
Procedurally generating the terrain and placing the tower **[20 Marks]**	None or one of the requirements were met.	One or two of the requirements were met.	Three or four of the requirements were met.	• Terrain is procedurally generated. And changes every time the game starts. • There are 3 or more distinct pathways for enemies to walk on. • The tower is placed where pathways meet. • Tower has health. • Tower automatically attacks enemies
	0 – 4 Marks	**5 – 9 Marks**	**10 – 15 Marks**	**16 – 20 Marks**
Adding and placing defenders **[20 Marks]**	None or one of the requirements were met.	One or two of the requirements were met.	Three or four of the requirements were met.	• There is a set number of locations to place defenders. • Defenders cannot be placed in pathways. • Defenders attack automatically. • Defenders can also be attacked and destroyed by enemies • Defenders have a set amount of health.
	0 – 4 Marks	**5 – 9 Marks**	**10 – 15 Marks**	**16 – 20 Marks**

Creating and spawning enemies **[20 Marks]**	None or one of the requirements were met.	One or two of the requirements were met.	Three or four of the requirements were met.	• Spawn locations are based on the generated terrain. • Enemies must spawn from a spawn location at a set interval. • Enemies gradually move towards the tower • Enemies can attack the tower and defenders. • Enemies have a set amount of health and can be destroyed by defenders and the tower.
	0 – 4 Marks	**5 – 9 Marks**	**10 – 15 Marks**	**16 – 20 Marks**

Complete the game loop **[10 Marks]**	The game loop is not complete. The game does not run, or the player can never reach a game over state.	Game loop is complete, but it is done in a way that does not result in an enjoyable experience. The game has quite a few bugs.	Game loop is complete. The mechanics implemented are fun and interesting to interact with. There are some minor bugs.	Game loop is complete, and unique and interesting mechanics are used to provide an enjoyable experience.
	0 – 2 Marks	**3 – 4 Marks**	**5 – 7 Marks**	**8 – 10 Marks**
User Interface **[10 Marks]**	Very little effort was put into the creation of the UI. Resulting in a confusing experience for the player.	Some effort was put into the UI, but there is a heavy reliance on text instead of UI elements to communicate with the player. For example, enemies, defenders, and towers don't have health bars, but text indicating their health.	Considerable effort was put into UI, but a few obvious UI features are missing. For example, it is not clear when an enemy is hit by a defender, because the enemy mesh does not flash on impact, etc.	The UI feels complete. Attention was paid to small details, resulting in an extremely positive player experience.
	0 – 2 Marks	**3 – 4 Marks**	**5 – 7 Marks**	**8 – 10 Marks**
Suitable Complexity **[10 Marks]**	Project reflects a level of difficulty and proficiency that is far below the module standard.	Project reflects a level of difficulty and proficiency that requires some work to meet the module standard.	Project reflects a level of difficulty and proficiency that meets the module standard.	Project reflects a level of difficulty and proficiency that exceeds the module standard.
	0 – 2 Marks	**3 – 4 Marks**	**5 – 7 Marks**	**8 – 10 Marks**

Presentation **[10 Marks]**	No presentation, or presentation does not have audio.	Presentation is just a demo. Students neglects to discuss the code implementation of the most relevant aspects of the project.	Presentation demonstrates most of the specific requirements of the brief. Code implementation of some of the most relevant aspects of the project are discussed.	Presentation demonstrates all specific requirements of the brief. Code implementation of the most relevant aspects of the project is discussed in full.
	0 – 2 Marks	**3 – 4 Marks**	**5 – 7 Marks**	**8 – 10 Marks**

END OF PART 1

[TOTAL MARKS: 100]

PART 2 — PROCEDURAL ENEMY WAVES ______________________________ (Marks: 100)

For the second part of the PoE, will be focusing on creating a system to procedurally generate waves of enemies. You will also add two new enemy and defender types.

1. Additional enemy types [20]

Create and integrate two additional enemy types.

They must meet these minimal requirements:

- The two new enemy types must have distinct behaviours from each other and the existing enemy type.
- All enemies must be visually distinctive.
- As in Part 1, they must follow a path in the terrain and move towards the tower.
- They must also attack the tower and the defenders.
  - Enemies have health bars and can be destroyed by defenders and the tower.

2. Additional defender types [20]

Create and integrate two additional defender types.

The following minimum requirements must be met:

- The two new defender types must have distinct behaviour from each other and the existing defender type
- All defenders must be visually distinct
- All defenders attack automatically
  - Defenders can also be attacked by enemies and should have a set amount of health.

3. Planning: Procedural enemy waves (Individual) [20]

In Part 1, the enemies moved towards the tower and were simply spawned at spawn locations at set intervals.

1. You are now required to create a more sophisticated enemy spawning system. Your procedural enemy spawning system must take game balance into consideration and provide an appropriate challenge for the player.

2. Each team member must write their OWN INDIVIDUAL document of 400-500 words that provides a detailed explanation of the strategy for spawning enemies. Consider the following:
    - How will difficulty be scaled to provide a consistent challenge?
    - How will the spawning adapt to the player’s skill level or play style?
    - How will the spawn locations be determined?
    - How will you determine when to spawn which enemy type?

    Use images and diagrams to support your explanation.

4. Implementation: Procedural enemy waves [20]

Now that you have planned your procedural enemy spawning strategy, it is time to implement it.

You will be assessed on the following:

- How closely your implementation matches what you described in your document.
- How well the game adapts to the player’s skill level and play style.
- How consistently the game challenges the player.

5.	**Suitable Complexity**	[10]
	Your project will be assessed on how well it meets the required level of complexity for this module. You get some flexibility in choosing how you implement parts of this project, but this means: • Your project should reflect an appropriate level of difficulty and proficiency. • Your solutions should show innovative thinking and effective problem-solving strategies. • Your work should present original ideas or approaches, displaying creativity in tackling the project's challenges.	
6.	**Presentation Video**	[10]
	Finally, create a presentation video to demonstrate your project. You must demonstrate the following: • The game you have created so far • Focus on demonstrating specific requirements of the brief • Discuss the most relevant code you have written for this part of the POE to demonstrate how you solved specific issues. The video should be roughly 5 minutes long, but no longer than 6 minutes.	
Submission		
You must submit the following to Arc: • a link to the GitHub repository that contains your project files • your procedural enemy waves planning document • your video presentation.		

Assessment Sheet (Marking Rubric) – Part 2

MODULE NAME:	**MODULE CODE:**
GAME DEVELOPMENT 3B	**GADE7322**

STUDENT NAME:	
STUDENT NUMBER:	

PART 2				
Marking Criteria	***Poor***	***Developing***	***Good***	***Excellent***
Additional enemy types **[20 Marks]**	None or one of the requirements were met.	One or two of the requirements were met.	Three or four of the requirements were met.	• All enemy types have distinct behaviour. • All enemies are visually distinct. • New enemies follow a pathway towards the tower. • New enemies attack the tower and the defenders. • New enemies have health bars and can be destroyed by defenders and the tower.
	0 – 4 Marks	**5 – 9 Marks**	**10 – 15 Marks**	**16 – 20 Marks**
Adding and placing defenders **[20 Marks]**	None or one of the requirements were met.	One or two of the requirements were met.	Three or four of the requirements were met.	• All defender types have distinct behaviour. • All defenders are visually distinct. • All defenders attack automatically. • Defenders can also be attacked by enemies • Defenders have a set amount of health.
	0 – 4 Marks	**5 – 9 Marks**	**10 – 15 Marks**	**16 – 20 Marks**

Planning: Procedural enemy waves (INDIVIDUAL) **[20 Marks]**	No or minimal planning of the requirements and/or the document is incomplete or does not include explanations of the questions or diagrams are missing.	Planning is not detailed and has not considered all of the requirements. The document does not answer all of the questions	Planning is consistent but not documented in detail.	• Planning of procedural enemy waves considers providing a consistent challenge to the player. • Planning also takes into account scaling difficulty to adapt to different player skills or play styles. • Planning is detailed and complete. • Diagrams and images are used to effectively communicate ideas.
	0 – 4 Marks	**5 – 9 Marks**	**10 – 15 Marks**	**16 – 20 Marks**

Implementation: Procedural enemy waves **[20 Marks]**	None the requirements were met.	One of the requirements were met.	Two of the requirements were met.	• Procedural enemy waves provide a consistent challenge to the player. • Procedural enemy waves adapt to player skill level and play styles. • Implementation closely matches what was described in planning document.
	0 – 4 Marks	**5 – 9 Marks**	**10 – 15 Marks**	**16 – 20 Marks**
Suitable Complexity **[10 Marks]**	Project reflects a level of difficulty and proficiency that is far below the module standard.	Project reflects a level of difficulty and proficiency that requires some work to meet the module standard.	Project reflects a level of difficulty and proficiency that meets the module standard.	Project reflects a level of difficulty and proficiency that exceeds the module standard.
	0 – 2 Marks	**3 – 4 Marks**	**5 – 7 Marks**	**8 – 10 Marks**
Presentation **[10 Marks]**	No presentation, or the presentation does not have audio.	The presentation is just a demo. Students neglect to discuss the code implementation of the most relevant aspects of the project.	The presentation demonstrates most of the specific requirements of the brief. The code implementation of some of the most relevant aspects of the project is discussed.	The presentation demonstrates all specific requirements of the brief. The code implementation of the most relevant aspects of the project is discussed in full.
	0 – 2 Marks	**3 – 4 Marks**	**5 – 7 Marks**	**8 – 10 Marks**

END OF PART 2

[TOTAL MARKS: 100]

PART 3 — PROCEDURAL GENERATION, SHADERS AND VISUAL EFFECTS ___ (Marks: 100)

For the final part of this PoE, you will be adding defenders and tower upgrades. You will also be adding custom shaders and visual effects to your game.

Finally, you will be adding your own custom procedural generation feature to the game.

1. Upgradable defenders and tower [20]

You should be able to upgrade your defenders and tower.

The following minimal requirements must be met:

- Each defender and the tower must have at least two upgrades each
- An upgrade must increase the health of the defender or tower it is being applied to
- You can also increase other stats like attack speed or defence
- The upgrades must change the visual appearance of the defender or tower
- The game must still feel balanced and present a challenge to the player. You might need to make some adjustments to your procedural enemy wave spawning to take these upgrades into consideration.

Examples of things to consider:

- Does the player use resources to buy the upgrades?
  - Is this an upgrade that is applied to all subsequent defenders added to the game, or can it only be applied to a single defender for a once-off upgrade?

2. Adding custom Shaders [10]

You must now add at least two custom shaders to your project.

Your shaders must meet the following minimum requirements:

- At least one of the shaders must displace vertices of the mesh it is associated with
- At least one of the shaders must change the base colour of the associated material in some way
- These shaders should add to the overall experience in a positive way

- You are allowed to use a graph-based editor in your game engine to create these effects.

3. Adding custom Visual Effects (VFX) [10]

You must now add at least one custom particle system and screen space visual effect to your game.

Your visual effects must meet the following minimum requirements:

- You must have one particle system-based visual effect
- You must have one screen space post-processing effect
- Both these visual effects must be custom-made
- Both these effects must add to the overall experience in a positive way
  - You are allowed to use a graph-based editor in your game engine to create these effects.

4. Planning: Custom procedural generation feature [20]

You are required to create a custom, procedurally generated feature in your game. This could be anything that you think would make your final game feel more complete and add something meaningful to the player's experience.

Examples of ideas to get you started:

- Add a procedurally generated soundtrack to your game
- Create a procedurally generated narrative for your game that changes every time you start a new game
- Add a procedurally generated enemy to the game that changes appearance and stats every time it is spawned.

> **IMPORTANT**: Get feedback on your planning from your lecturer before you start implementing your idea. Your lecturer will be able to provide some guidance on how to implement the feature, but also ensure it meets the module standard.

Individually write a document of 400-500 words that provides a detailed explanation of the procedurally generated feature you are planning to implement. Use drawings and diagrams to support your planning.

5.	**Implementing: Custom procedural generation feature**	[20]
	Now that you have planned your custom procedural generation feature, it is time to implement it into your game. You will be assessed on the following: • How closely your implementation matches what you described in your document • How well it adds to the overall experience of the game • How well it integrates into the game and functions overall.	
6.	**Suitable Complexity**	[10]
	Your project will be assessed on how well it meets the required level of complexity for this module. You get some flexibility in choosing how you implement parts of this project, but this means: • Your project should reflect an appropriate level of difficulty and proficiency • Your solutions should show innovative thinking and effective problem-solving strategies • Your work should present original ideas or approaches, displaying creativity in tackling the project’s challenges.	
7.	**Presentation Video**	[10]
	Finally, create a presentation video to demonstrate your project. You must demonstrate the following: • The complete game you have created in this PoE • The specific requirements of the final part of this PoE • Discuss the most relevant code you have written for this part of the PoE to demonstrate how you solved specific issues. The video should be roughly 5 minutes long, but no longer than 6 minutes.	

Submission
You must submit the following to Arc: • A link to the GitHub repository that contains your project files • Your custom procedural generation feature planning document • Your video presentation.

Assessment Sheet (Marking Rubric) – (POE)

MODULE NAME:	MODULE CODE:
GAME DEVELOPMENT 3B	GADE7322

STUDENT NAME:	
STUDENT NUMBER:	

PART 3				
Marking Criteria	***Poor***	***Developing***	***Good***	***Excellent***
Upgradable defenders and tower **[20 Marks]**	None or 1 of the requirements were met.	Two of the requirements were met.	Three of the requirements were met.	• Defenders and the tower have at least 2 upgrades each. • Upgrades increase health. • Upgrades change the visual appearance of defenders and towers. • The game still feels balanced and presents a challenge to the player.
	0 – 4 Marks	**5 – 9 Marks**	**10 – 15 Marks**	**16 – 20 Marks**
Adding custom shaders **[10 Marks]**	None the requirements were met.	One of the requirements were met.	Two of the requirements were met.	• At least one of the shaders displaces vertices of the mesh it is associated with. • At least one of the shaders changes the base colour of the associated material in some way. • The shaders add to the overall experience in a positive way.
	0 – 2 Marks	**3 – 4 Marks**	**5 – 7 Marks**	**8 – 10 Marks**

Adding custom visual effects. **[10 Marks]**	None or one of the requirements were met.	Two of the requirements were met.	Three of the requirements were met.	• A custom particle system-based visual effect is present in the game. • A custom screen space post-processing effect is present in-game. • Both these visual effects are custom made. • Both these effects add to the overall experience in a positive way.
	0 – 2 Marks	**3 – 4 Marks**	**5 – 7 Marks**	**8 – 10 Marks**

Planning: Custom procedural generation feature (INDIVIDUAL) **[20 Marks]**	No or minimal planning of the requirements and/or the document is incomplete or does not include explanations of the questions or diagrams are missing.	Planning is not detailed and has not considered all of the requirements. The document does not answer all of the questions	Planning is consistent but not documented in detail.	• Planning of custom procedural generation feature is informed by research. • Planning of feature shows creative use of procedural generation techniques. • Planning is detailed and complete. • Diagrams and images are used to effectively communicate ideas.
	0 – 4 Marks	**5 – 9 Marks**	**10 – 15 Marks**	**16 – 20 Marks**
Implementation: Custom procedural generation feature **[20 Marks]**	None or one of the requirements were met.	Two of the requirements were met.	Three of the requirements were met.	• Custom procedural generation adds to the overall experience of the game in a positive way. • Implementation closely matches what was described in the planning document. • The feature is integrated into the game and doesn't feel like a stand-alone feature. • The feature works well and is bug free.
	0 – 4 Marks	**5 – 9 Marks**	**10 – 15 Marks**	**16 – 20 Marks**

Suitable Complexity **[10 Marks]**	Project reflects a level of difficulty and proficiency that is far below the module standard.	Project reflects a level of difficulty and proficiency that requires some work to meet the module standard.	Project reflects a level of difficulty and proficiency that meets the module standard.	Project reflects a level of difficulty and proficiency that exceeds the module standard. Full marks will be awarded for going above and beyond the minimal requirements.
	0 – 2 Marks	**3 – 4 Marks**	**5 – 7 Marks**	**8 – 10 Marks**
Presentation **[10 Marks]**	No presentation, or presentation does not have audio.	Presentation is just a demo. Students neglect to discuss the code implementation of the most relevant aspects of the project.	Presentation demonstrates most of the specific requirements of the brief. The code implementation of some of the most relevant aspects of the project is discussed.	Presentation demonstrates all specific requirements of the brief. The code implementation of the most relevant aspects of the project is discussed in full.
	0 – 2 Marks	**3 – 4 Marks**	**5 – 7 Marks**	**8 – 10 Marks**

END OF POE (PART 3)

[TOTAL MARKS: 100]