UOW MALAYSIA

PART OF THE UNIVERSITY
OF WOLLONGONG AUSTRALIA
GLOBAL NETWORK

Bachelor of Game Development (Hons)

Game Engine Architecture and Design

XBGT3124N

Prepared by Mohamad Faris Zakwan
Semester September 2024

SCHOOL OF COMPUTING & CREATIVE MEDIA

ASSIGNMENT 2

Course Title : Game Engine Architecture & Design
Course Code : XBGT3124N
Course Lecturer : Mohamad Faris Zakwan
: faris.z@uow.edu.my

BRIEF

> ⚠ Important ⚠
>
> This course's assignments are continuous in nature.
> Use your Assignment 1 project as the base for Assignment 2.

Integrate to your game engine project:

- OCM, including Scene Management.
- Rendering System.
- Resource Management.

Then, you need to:

1. Add TransformComponent to replace the embedded transformation in GameObject.
2. Make some improvements to the OCM to make its usage easier for game developers.
3. Make a showcase.
4. **Documentation describing:**
    - **Your improvements to the OCM.**
    - **What the showcase is demonstrating.**

REQUIREMENTS

OCM & Scene Management

Integration
Integrate the OCM and Scene management systems from the tutorial to your game engine project.

Improvements
Additional improvements could be made to the OCM system, **for example (non-exhaustive)**:

1. Currently, game developers are required to manually add each created GameObject to GameObjectCollection. This process can be automated.

2. Currently, a component can be added to a GameObject multiple times. While there are valid use-cases of this, some components **MUST NOT** be allowed to be attached multiple times, e.g., TransformComponent.

3. **(With TransformComponent)** For other components to refer to the GameObject's transformation data, it needs to obtain the reference using `gameObject.GetComponent<TransformComponent>()`.

   In Unity game engine, game developers could simply refer to Transform component using `gameObject.transform`, which is much easier.

4. GameObjectCollection uses `std::vector` internally to hold GameObjects. This is problematic, as game developers can accidentally add the same GameObject twice, making the GameObject's lifecycle called twice per update loop.

   Provide safety mechanism to ensure each unique GameObject is only added once.

5. Currently, the OCM does not support nested GameObject i.e. GameObject as a child of another GameObject. Add this functionality.

   Note: This requires understanding of hierarchical transformation from XBGT2064.

6. There is no way to find GameObject by name. Searching GameObject by name is an expensive process if done every update frame but has a valid use-case.

   Add a way for game engine users to find GameObject by name.

You are to **provide documentation describing the work done** to improve *your* OCM.
- For each work, describe the problem, the proposed solution, and the approach to reach the solution.
- Explain in brief and concise sentences.

Transform Component

The OCM from tutorial embeds transformation data in the GameObject class. This does not conform to OCM. Create a TransformComponent class to handle transformation.

The TransformComponent needs to:

- Have Position (XY), rotation (Z) and scaling (XY)
- The values settable and gettable using setXXX() and getXXX()
- **MUST** be added to GameObject by default on the GameObject's creation.

You are also required to **remove ALL codes pertaining to the transformation data** in GameObject class.

Rendering System

Integrate the rendering system (from tutorial) to the game engine project.

Then, add Texture rendering functionality.

Resource Management

Include proper Resource Management. All loadable resources **MUST** make use of this system.

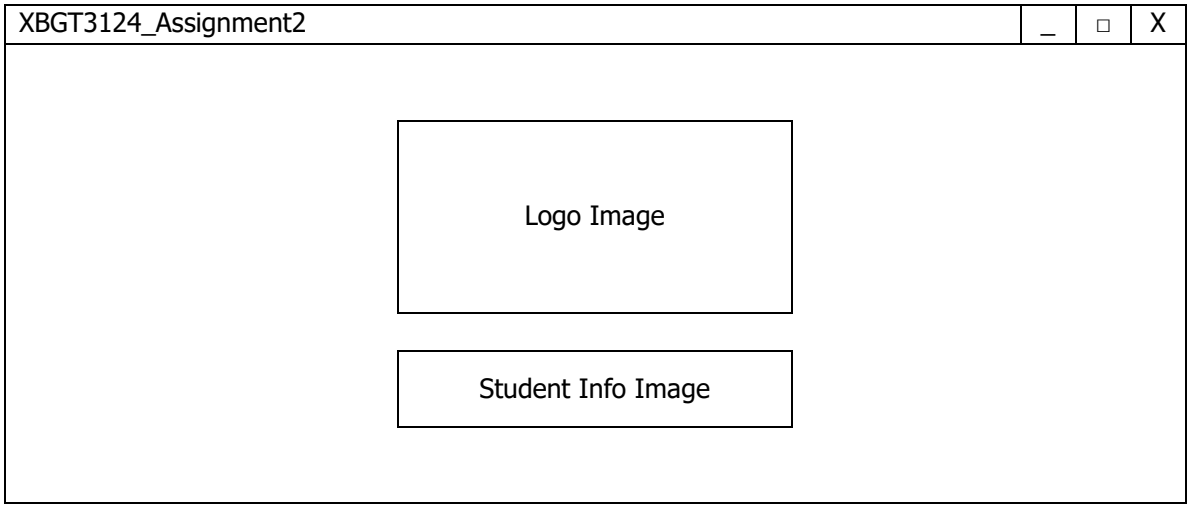
IMPORTANT:
INCLUDING RESOURCE MANAGEMENT BUT NOT USING IT MEANS YOU GET 0 MARK FOR THIS PART.

Showcase

Your showcase **MUST** demonstrate all requirements in action, with some MANDATORY requirements.

Mandatory Requirements

1. On application launch: run SplashScene for three **(3)** seconds. SplashScene MUST contain:
    a. A custom logo (max image size 256x256)
    b. Your name and student ID (as another image below it)
    c. See diagram below for example layout.
2. Load ShowcaseScene after SplashScene sequence.
3. If [Space] key is pressed while SplashScene is running, skip to ShowcaseScene.



In ShowcaseScene, demonstrate your OCM in action, such as:

1. Spawning GameObject with component.
2. Destroying components and GameObject.
3. Using Input system to manipulate GameObject and components, especially TransformComponent.
4. Component spawning GameObject with a Component that destroys that GameObject after a delay.

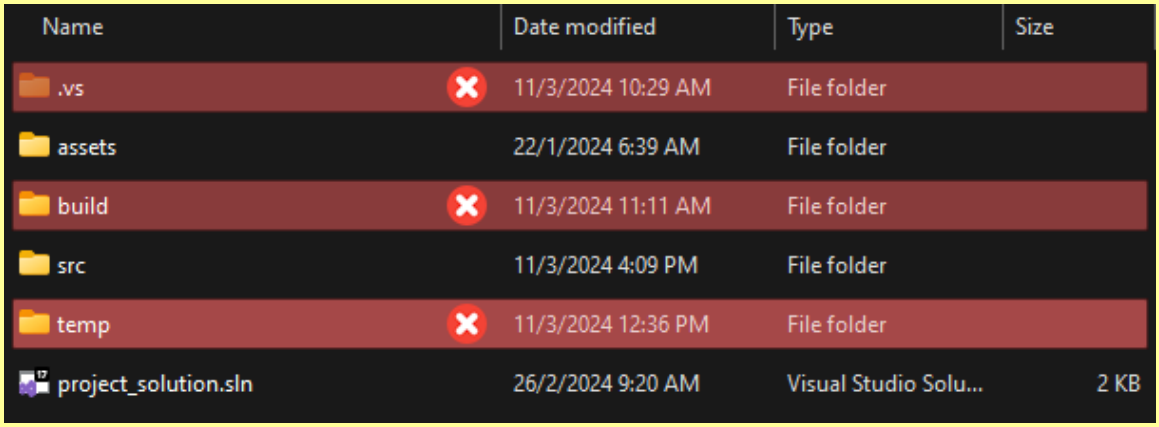
Notes

- For showcase, you are required to create some custom components and scenes.
    - You may provide more than one Showcase scene, just ensure that you provide a way to switch between the scenes!
- **Custom scenes CAN have logic!** e.g., you can put logic, such as input, in `on_update()`, and rendering in `on_render()`.
- Having visualization is highly encouraged, e.g., seeing a coloured rectangle appearing and disappearing (due to GameObject/Component spawning and removal) demonstrates better than logging to console.

SUBMISSION REQUIREMENTS

⚠ **Do not include the highlighted folders in your submission!** ⚠

By default, .vs folder is hidden in Windows Explorer. Enable Hidden items to see it.



Delete the highlighted folders before zipping

Zip Filename Format: *StudentID*_XBGT3124_A2
e.g., 0120123_XBGT3124_A2.zip

Your zip file MUST contain:

1. project/ folder **(REMEMBER TO DELETE THE HIGHLIGHTED FOLDERS!)**
2. deps/ folder
3. Readme or documentation about your project.

Each student is given access to an individual OneDrive folder for assignment submissions.

- Only the student and lecturer-in-charge will be able to access the contents inside the folder.
- The same folder is used for all coursework submissions.

Penalty applies:

- **If you submit past the deadline: -20% of marks per day late.**
- **If you update your submission past the deadline without receiving permission from the lecturer beforehand.**

ASSESSMENT CRITERIA

Coursework marks allocated for this assignment is **30%**, of which:

- OCM & Scene Management : 12%
  - Integration : 2%
  - Improvements : 10%
- Transform Component : 3%
- Rendering System : 7%
- Resource Management : 3%
- Showcase : 5%

No submission or non-working submission is assessed as 0% of the allocated marks.

DUE DATE: 12 NOVEMBER 2024 11:59:59PM

ASSESSMENT RUBRIC

CRITERIA	MARKS					
OCM	**10**	**8**	**6**	**4**	**2**	**0**
Integration (2)					Completed OCM tutorial and integrated the module correctly.	One of the following: • No submission. • Does not meet minimum requirements.
Improvements (10)	Includes five (5) improvements to the OCM. Documented the improvements correctly.	Includes four (4) improvements to the OCM. Documented the improvements correctly.	Includes three (3) improvements to the OCM. Documented the improvements correctly.	Includes two (2) improvements to the OCM. Documented the improvements correctly.	Includes one (1) improvement to the OCM. Documented the improvement correctly.	
TransformComp	**-**	**-**	**3**	**2**	**1**	
Implementation (3)			Transform component able to provide values: • Translation • Rotation • Scale • Transform Matrix Transform component added to GameObject by default on creation.	Transform component able to provide values: • Translation • Rotation • Scale Transform component added to GameObject by default on creation.	Transform component able to provide values: • Translation • Rotation • Scale	
Renderer	**-**	**4**	**3**	**2**	**1**	
Implementation (3)			Completed all the Renderer tutorial and integrated the module correctly.	Completed most of the Renderer tutorial and integrated the module correctly.		
Texture (4)		Five (5) of following: • Construction allows custom configuration. • Disposed properly on texture deletion. • Users can obtain width and height of the texture. • Can alter texture wrap mode and filter mode AFTER texture construction. • One loading function supports BOTH RGB and RGBA image.	Three (3) of following: • Construction allows custom configuration. • Disposed properly on texture deletion. • Users can obtain width and height of the texture. • Can alter texture wrap mode and filter mode AFTER texture construction. • One loading function supports BOTH RGB and RGBA image.	Two (2) of following: • Construction allows custom configuration. • Disposed properly on texture deletion. • Users can obtain width and height of the texture. • Can alter texture wrap mode and filter mode AFTER texture construction. • One loading function supports BOTH RGB and RGBA image.	One (1) of following: • Construction allows custom configuration. • Disposed properly on texture deletion. • Users can obtain width and height of the texture. • Can alter texture wrap mode and filter mode AFTER texture construction. • One loading function supports BOTH RGB and RGBA image. •	

Res. Manage	-	-	3	2	1	0
Implementation (3)			Upgraded Resource Management system to make use of C++ templates.	Completed the Resource Management tutorial and integrated the module correctly.		
Showcase	**5**	**4**	**3**	**2**	**1**	
Content (5)	Five (5) the following: • GameObject and Component creations BEFORE gameplay. • GameObject and Component addition DURING gameplay. • GameObject and Component deletion DURING gameplay. • Proper clean up BEFORE scene changes, or BEFORE program terminates. • Visuals AND console logs used to assist demonstration.	Four (4) the following: • GameObject and Component creations BEFORE gameplay. • GameObject and Component addition DURING gameplay. • GameObject and Component deletion DURING gameplay. • Proper clean up BEFORE scene changes, or BEFORE program terminates. • Visuals AND console logs used to assist demonstration.	Three (3) the following: • GameObject and Component creations BEFORE gameplay. • GameObject and Component addition DURING gameplay. • GameObject and Component deletion DURING gameplay. • Proper clean up BEFORE scene changes, or BEFORE program terminates. • Visuals AND console logs used to assist demonstration.	Two (2) the following: • GameObject and Component creations BEFORE gameplay. • GameObject and Component addition DURING gameplay. • GameObject and Component deletion DURING gameplay. • Proper clean up BEFORE scene changes, or BEFORE program terminates. • Visuals AND console logs used to assist demonstration.	One (1) the following: • GameObject and Component creations BEFORE gameplay. • GameObject and Component addition DURING gameplay. • GameObject and Component deletion DURING gameplay. • Proper clean up BEFORE scene changes, or BEFORE program terminates. • Visuals AND console logs used to assist demonstration.	One of the following: • No submission. • Does not meet minimum requirements.
PENALTIES	**OCM**	**TransformComp**	**Namespace**	**Res. Manage**	**Showcase**	**Late**
Infractions	Documentation missing or incomplete. (-1 each, max -5)	For not removing transformation code from GameObject (-1) **Commenting out DOES NOT COUNT as removal. Remove means remove.**	Objects that are part of game engine are not namespaced (-1) Objects that are NOT part of game engine are included in the game engine namespace (-1) Not changing namespace from MyGameEngine to something else (-1)	Resources not utilizing this object at all (-3)	Missing or incomplete documentation (-2) Mandatory requirements not met. (-1)	-20% assessed marks per day late. How days are counted: x = hours late Days = ceil(x / 24) e.g. 5 minutes late: x = 5/60 = 0.0833 Days = ceil(0.0833/24) = ceil(0.003472) = 1 day 27 hours late: x = 27 Days = ceil(27/24) = ceil(1.125) = 2 days