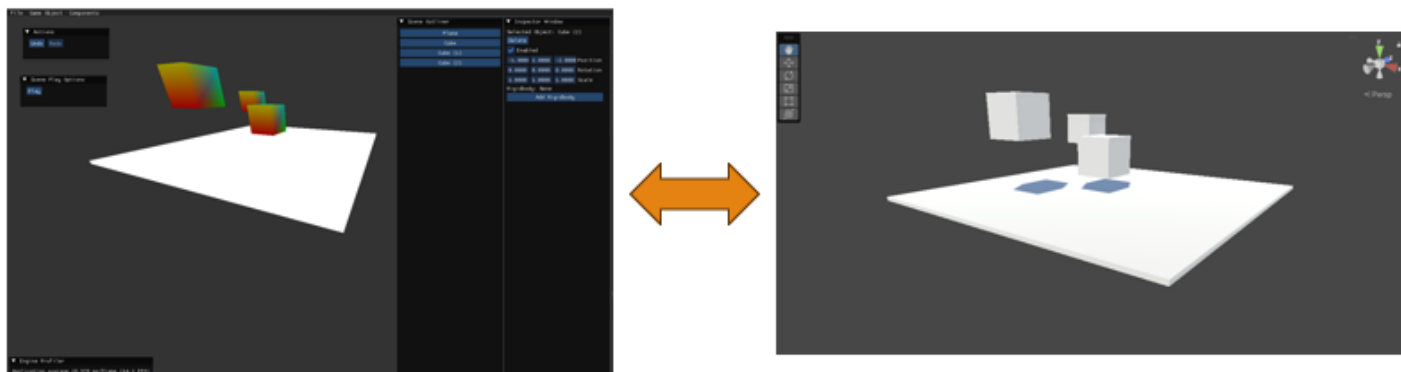


**[100 pts] General Instructions:** For the final exam, you are required to create parsers that translate your scene editor level, into a Unity scene and vice-versa.



For this hands-on final challenge, the following must be accomplished:

1. Your GDENG03 scene editor must have a way to save your level using a standardized text format. Let this be your **.level** file.
2. Create editor scripts for Unity where the goal of the script is to extract information from objects in the .level file, such as their transforms, object primitive type, and rigid body information, then recreate it within the Unity scene. From now on, these Unity editor script(s) will be referred to as the **l2u-parser** (.level to .unity parser).
  - o Editor Scripting: <https://learn.unity.com/tutorial/editor-scripting>
  - o You may also refer to <https://docs.unity3d.com/Manual/TextSceneFormat.html> for understanding how Unity .scene files are saved.
  - o This approach is taken as creating your own .scene file from scratch is **not recommended**.
  - o Objects should be instantiated during edit mode.
3. Finally, your GDENG03 scene editor must be able to parse Unity **.scene** files and load the objects as well as their transforms and (some) of the rigid body information. From now on, this scene editor script(s) will be referred to as the **u2l-parser** (.unity to .level parser).

### Checklist of Requirements

Your parser must have ALL the following features implemented properly.

Requirement	Description
Primitives supported	The following game object primitives must be supported: <b>plane and cube</b>
Saving/Loading in editor	The scene editor should have a method for saving your level as a text file. It must also have an option for loading existing level files.
Parser and editor scripts must be backwards compatible	Any changes that were performed in Unity, can be saved by the user. Hence, loading the updated file to your scene editor should reflect the new changes.
Can support multiple objects	Your scene editor should support loading of many objects. Scene hierarchy and object parenting not required.
Rigid body support	Objects with rigid body components in your scene editor, should be interpreted properly in Unity, attaching/enabling rigid bodies when required, and setting its respective properties (if present).

## Test Cases

Test cases will be provided in the Canvas assignment page.

## Submission Details

You are required to submit the following:

- SOURCE – A GDrive or a GitHub link that contains your source code. Add a README.txt (or README.md for GitHub) that has your name and instructions how to run your program. Also indicate the entry class file, where the main function is located. **Include this as a comment to your submission.**
  - Include the necessary editor scripts and dependencies by grouping them in the ff directory:
    - (root)/Unity
  - Include Academic Honesty Agreement form declaration. See Appendix A.
- Video evidence (mp4) for each test cases, compressed in a zip file.
  - Name each of the video recording as follows: test\_[#].mp4
  - Ensure that the video file size isn't too large (e.g., 1gb for a 15 second video). Refer to the following software for optimizing / compressing the videos: <https://handbrake.fr/>

## APPENDIX A: Academic Honesty Form

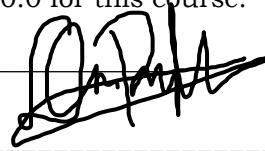
=====Copy and paste the following section below and sign the form. Save in PDF file=====

### ACADEMIC HONESTY AGREEMENT

I am answering this exam myself and to the best of my ability, without any assistance from other persons, materials, or resources that I am not allowed to access during the exam period. I declare that the software is fully written by me and without any assistance from my peers. I am fully aware and hereby agree to the clause that violation of this agreement is considered cheating and will result in a 0.0 for this course.

David Rex C. Mayuga

**Signature over Printed Name**



=====END=====