1**. Improve Texture Management (Data Structure Enhancement)**

**Current Issue:**

* m\_textureIDs is a fixed-size array of 16 elements.
* Texture search methods (FindTextureID, FindTextureSlot) use linear search, which is inefficient as the number of textures grows.

**Enhancement Plan:**

* Replace m\_textureIDs array with a std::unordered\_map<std::string, GLuint> to store textures by tag.
* This will allow O(1) average lookup for texture IDs and slots instead of O(n) linear search.
* Update all texture-related methods (CreateGLTexture, FindTextureID, FindTextureSlot, BindGLTextures) to use the map.

**Skills Highlighted:**

* Knowledge of hash tables and algorithmic efficiency.
* Improves search operations in rendering.

**2. Optimize Material Lookup**

**Current Issue:**

* Materials are stored in std::vector<OBJECT\_MATERIAL> and searched linearly in FindMaterial.

**Enhancement Plan:**

* Store materials in std::unordered\_map<std::string, OBJECT\_MATERIAL> keyed by material tag.
* Update DefineObjectMaterials to populate the map.
* FindMaterial becomes a simple map.find() call.

**Skills Highlighted:**

* Use of map data structure for fast retrieval.
* Demonstrates understanding of key-value pair storage for real-time rendering**.**

**3. Refactor Repetitive Loops into Helper Functions**

**Current Issue:**

* Rendering multiple objects (placemats, cups, table legs) uses repeated nested loops.
* Hard-coded scale, rotation, and positions in RenderScene.

**Enhancement Plan:**

* Create helper methods like:
  + void RenderRepeatedObjects(glm::vec3 scale, glm::vec3 startPos, glm::vec3 step, int countX, int countZ, std::string materialTag, std::string textureTag, std::function<void()> drawFunc);
* This reduces code repetition, improves maintainability, and clearly separates rendering logic from transformation setup.

**Skills Highlighted:**

* Modular design, code refactoring, and algorithmic thinking.
* Shows ability to handle nested loops and object transformations efficiently.

**4. Improve Memory Management**

**Current Issue:**

* DestroyGLTextures() calls glGenTextures() instead of glDeleteTextures().
* Could cause memory leaks in OpenGL.

**Enhancement Plan:**

* Correct DestroyGLTextures():

void SceneManager::DestroyGLTextures() {

for (auto& tex : m\_textureMap) {

glDeleteTextures(1, &tex.second);

}

m\_textureMap.clear();

}

* Reinforces proper resource management.

**Skills Highlighted:**

* Safe memory and resource handling in graphics programming.

**5. Narrative for ePortfolio**

Once you implement these enhancements, your narrative can highlight:

* Selection of artifact: A complex 3D scene renderer showing real-world application of algorithms and data structures.
* Skills showcased: Use of hash tables for texture/material management, modular rendering functions, and efficient search algorithms.
* Improvement reflection: Enhanced code structure, faster lookups, fewer bugs, and better memory safety.
* Course outcomes: Algorithm efficiency (Outcome 3), software engineering practices (Outcome 4), secure resource management (Outcome 5).