**Специфікація вимог до програмного забезпечення**

**1. Вступ**

**1.1 Мета**

Метою розробки є створення гнучкої та високопродуктивної системи 3D-рендерингу, яка базується на абстрактному шарі інтерфейсу апаратного рендерингу (RHI). Система призначена для уніфікації роботи з кількома графічними API, зокрема з початковою підтримкою DirectX 12 та потенційною розширюваністю до інших API, таких як Vulkan. Основна ідея полягає в абстрагуванні низькорівневих деталей API, щоб дозволити розробникам зосередитися на високорівневій логіці створення складних 3D-сцен, таких як ігрові світи, архітектурні візуалізації чи симуляції. Система має підтримувати сучасні техніки рендерингу, зокрема фізично базований рендеринг (PBR), інстансований рендеринг для оптимізації продуктивності та обробку активів у стандартизованому форматі GLTF. Розробка спрямована на забезпечення високої якості зображення, стабільної частоти кадрів у реальному часі та можливості адаптації до майбутніх технологій і платформ, що робить її цінним інструментом для професіоналів у сфері комп’ютерної графіки.

**1.2 Область застосування**

Система орієнтована на використання в галузях, де потрібна висока продуктивність і гнучкість 3D-рендерингу, зокрема в розробці ігрових рушіїв, інструментах візуалізації та симуляційних програмах. Вона призначена для команд розробників, які шукають видайний компроміс між візульною якістю та швидкодією. Також система може застосовуватися в архітектурному дизайні для створення деталізованих 3D-моделей будівель із підтримкою PBR для точного відтворення текстур і світла, а також у симуляціях (наприклад, авіаційних чи автомобільних), де потрібна висока деталізація сцен. Система не включає графічний інтерфейс користувача, а натомість пропонує програмний API, призначений для інтеграції в існуючі рушії чи розробницькі середовища, що робить її придатною для професійних розробників із досвідом у роботі з низькорівневими API та C++.

**1.3 Визначення, скорочення та абревіатури**

* **3D:** Three-Dimensional (тривимірний).
* **API:** Application Programming Interface (програмний інтерфейс прикладного програмування).
* **ECS:** Entity-Component-System (система компонентів-об’єктів).
* **GLTF:** Graphics Language Transmission Format (формат передачі графічних даних).
* **GPU:** Graphics Processing Unit (графічний процесор).
* **PBR:** Physically-Based Rendering (фізично базований рендеринг).
* **RHI:** Rendering Hardware Interface (інтерфейс апаратного рендерингу).

**2. Загальний опис**

**2.1 Перспектива продукту**

Система 3D-рендерингу розробляється як модульна платформа, яка виступає проміжним шаром між високорівневими логіками розробників і низькорівневими деталями графічних API. Вона базується на абстрактному шарі RHI, який уніфікує операції рендерингу (наприклад, створення буферів, налаштування шейдерів, синхронізація з GPU) для різних API, починаючи з DirectX 12. Цей підхід дозволяє уникнути дублювання коду для кожного API та полегшує майбутнє розширення, наприклад, додавання підтримки Vulkan. Система інтегрує сучасні техніки, такі як PBR для реалістичного відтворення матеріалів і інстансований рендеринг для оптимізації великих сцен, а також підтримує завантаження активів у форматі GLTF із використанням бібліотеки DirectXTex для генерації міп-карт. Архітектура на основі ECS забезпечує гнучке управління сценами, що робить систему придатною для обробки складних 3D-середовищ із тисячами об’єктів.

**2.2 Функції продукту**

* **Ініціалізація рендерингового контексту:** Система налаштовує контекст рендерингу через RHI, забезпечуючи базову інфраструктуру для роботи з DirectX 12, включаючи створення командних буферів і налаштування конвеєра.
* **Завантаження GLTF-активів:** Модуль AssetsLoader парсить GLTF-файли (моделі, текстури, матеріали) за допомогою бібліотеки tinygltf, інтегруючи їх у сцену з автоматичною генерацією міп-карт через DirectXTex для оптимізації продуктивності.
* **Фізично базований рендеринг (PBR):** Реалізовано форвардний конвеєр PBR із підтримкою фізичних властивостей матеріалів (металізованість, шорсткість, нормалі), що забезпечує реалістичне відтворення світла.
* **Інстансований рендеринг:** Система використовує єдиний виклик DrawInstanced для рендерингу тисяч однакових об’єктів, оптимізуючи навантаження на CPU і GPU через буфери трансформацій.
* **Управління сценами через ECS-архітектуру:** Використання бібліотеки entt для створення сутностей із компонентами (Transform, InstancedStaticMesh, MaterialParameters), що дозволяє ефективно керувати великими сценами з ієрархічними зв’язками.

**2.3 Характеристики користувачів**

Система орієнтована на професійних розробників із досвідом у програмуванні на C++ та роботі з графічними API. Користувачі повинні володіти знаннями про структуру 3D-даних, шейдери (HLSL) та принципи рендерингу. Основна аудиторія включає розробників ігрових рушіїв, які потребують високої продуктивності, та спеціалістів із візуалізації, які працюють над деталізованими 3D-моделями. Система не призначена для початківців через відсутність графічного інтерфейсу та складність інтеграції.

**2.4 Обмеження**

* **Платформна залежність:** Початкова підтримка лише Windows із DirectX 12, що обмежує негайну кросплатформність (потенційна підтримка Vulkan у майбутньому).
* **Технологічні обмеження:** Використання C++, DirectXTex і tinygltf вимагає сумісності з цими бібліотеками та їх оновленнями.
* **Апаратні вимоги:** Система розрахована на GPU з підтримкою DirectX 12 (асинхронні обчислення, шейдерна модель 6.0+), що виключає застаріле обладнання.
* **Відсутність UI:** Система не включає графічний інтерфейс, вимагаючи інтеграції через програмний API.

**2.5 Припущення та залежності**

* **Апаратне забезпечення:** Передбачається наявність сучасного GPU з підтримкою DirectX 12 та достатнім обсягом відеопам’яті для PBR і інстансованого рендерингу.
* **Користувацькі навички:** Користувачі мають знати C++ і принципи роботи з графічними API.
* **Залежності:** Система залежить від бібліотек DirectXTex (для обробки текстур) і tinygltf (для GLTF), а також від стабільної роботи DirectX 12 у Windows 10+.
* **Умови експлуатації:** Тестування та використання проводяться в контрольованих умовах із середнім і високим рівнем апаратних ресурсів.

**3. Специфічні вимоги**

**3.1 Функціональні вимоги**

* **Ініціалізація рендерингового контексту:** Система має забезпечувати ініціалізацію рендерингового контексту через абстрактний шар RHI, включаючи створення командних буферів, налаштування конвеєра рендерингу та синхронізацію з GPU для роботи з DirectX 12. Цей процес має бути модульним, дозволяючи майбутнє розширення для інших API, таких як Vulkan.
* **Завантаження GLTF-активів:** Модуль AssetsLoader має підтримувати завантаження 3D-активів у форматі GLTF, включаючи геометрію, текстури, матеріали та анімації (на майбутнє). Використовуючи бібліотеку tinygltf, система парсить файли та інтегрує їх у сцену, а бібліотека DirectXTex генерує міп-карти для оптимізації текстур, зменшуючи артефакти та навантаження на GPU.
* **Фізично базований рендеринг (PBR):** Реалізовано форвардний конвеєр PBR із підтримкою фізичних властивостей матеріалів, таких як металізованість, шорсткість, нормалі та альбедо. Система використовує шейдери HLSL для обчислення BRDF (Bidirectional Reflectance Distribution Function), забезпечуючи реалістичне відтворення світла та матеріалів у реальному часі.
* **Інстансований рендеринг:** Система підтримує інстансований рендеринг через єдиний виклик DrawInstanced, передаючи масиви трансформацій у вершинні буфери. Це дозволяє ефективно рендерити тисячі однакових об’єктів (наприклад, дерева чи будівлі) за один цикл, оптимізуючи продуктивність.
* **Управління матеріалами:** Модуль MaterialsManager забезпечує динамічне налаштування матеріалів, включаючи прив’язку текстур (базові, нормальні, шорсткості) до шейдерів, управління константними буферами для PBR-параметрів та підтримку masked opaque матеріалів для прозорих об’єктів.

**3.2 Нефункціональні вимоги**

* **Вимоги до продуктивності:** Система має підтримувати стабільну частоту кадрів (не менше 60 FPS) при рендерингу сцен із до 10,000 інстансованих об’єктів на GPU середнього рівня (наприклад, NVIDIA GTX 1650). Час завантаження GLTF-файлів не повинен перевищувати 2 секунди для активів розміром до 100 MB.
* **Масштабованість:** Архітектура дозволяє додавати підтримку нових API (наприклад, Vulkan) та технік рендерингу (наприклад, трасування променів) без значних змін у базовому коді, завдяки модульності RHI та ECS.
* **Сумісність:** Система сумісна з операційною системою Windows 10+ та GPU з підтримкою DirectX 12 (Shader Model 6.0+), що забезпечує роботу на сучасних ігрових та робочих станціях.

**3.3 Обмеження проєктування**

* Обмеження пов’язані з початковою орієнтацією на DirectX 12, що вимагає адаптації для інших API.
* Використання C++ і специфічних бібліотек (DirectXTex, tinygltf) обмежує гнучкість зміни технологічного стеку.
* Відсутність підтримки застарілого обладнання через вимоги до GPU.

**3.4 Атрибути якості**

* **Зручність використання:** API системи спроектоване для розробників із досвідом, надаючи чітку документацію та приклади інтеграції.
* **Надійність:** Система витримує безперервну роботу протягом 24 годин із динамічними змінами сцени без збоїв чи витоків пам’яті.
* **Підтримуваність:** Модульна структура дозволяє легко оновлювати компоненти (наприклад, RHI чи AssetsLoader) без впливу на інші модулі.

**3.5 Інші вимоги**

* **Безпека:** Належне керування пам’яттю GPU, уникнення переповнення буферів і використання синхронізації для запобігання гонок даних.
* **Захист:** Дані активів захищаються від некоректного доступу через ізоляцію модулів обробки.

**4. Перевірка та валідація**

**4.1 Плани тестування**

* **Функціональне тестування:** Перевірка ініціалізації RHI, завантаження GLTF-активів (коректність геометрії, текстур, матеріалів), роботи PBR та інстансованого рендерингу.
* **Тестування продуктивності:** Оцінка FPS у сценах із різною кількістю об’єктів на GPU, вимірювання часу завантаження GLTF-файлів.
* **Тестування стабільності:** Тривале виконання (24 години) із динамічними змінами (додавання/видалення об’єктів, зміна освітлення) для виявлення збоїв чи витоків пам’яті.
* **Інструменти:** Використання DirectX Debug Layer для аналізу помилок API, власні скрипти для автоматизації тестів.

**4.2 Критерії приймання**

* Система вважається прийнятою, якщо:
  + Успішно ініціалізується рендеринговий контекст і завантажуються GLTF-активи без помилок.
  + PBR забезпечує реалістичне відтворення матеріалів із правильним освітленням (перевірка візуально та через порівняння з еталонними сценами).
  + Інстансований рендеринг обробляє 10,000 об’єктів із FPS не нижче 60 на цільовому GPU.
  + Система працює стабільно протягом 24 годин без збоїв.
  + Час завантаження GLTF-файлу (100 MB) не перевищує 2 секунди.