Анализ кода проекта

Файлы проекта

- **Texture.cpp** – реализация загрузки текстур из DDS и PNG.
- **DirectX11App.cpp** – основной файл приложения: инициализация окна, DirectX, создание ландшафта, шейдеров, рендеринг, постпроцессинг, интерфейс ImGui.
- **Light.h** – структура источника света и функция расчёта освещения по Фонгу.
- **Texture.h** – объявление структуры описания текстуры и прототипов функций.
- **framework.h**, **targetver.h**, **Resource.h** – вспомогательные заголовки Windows (не содержат логики).

Таблицы функций и методов

Texture.cpp

Функция	Назначение	Принцип работы
`DivUp(UINT32 a, UINT32 b)`	Целочисленное деление с округлением вверх.	Возвращает `(a + b - 1) / b`. Используется для расчёта количества блоков в сжатых текстурах.
`GetBytesPerBlock(const DXGI_FORMAT& fmt)`	Возвращает размер блока (в байтах) для сжатых форматов BC.	По формату возвращает 8 байт (BC1, BC4) или 16 байт (остальные). При неизвестном формате – assert.
`WCSToMBS(const std::wstring& wstr)`	Преобразование широкой строки в многобайтовую.	Использует `wcstombs_s`. В проекте не вызывается, но оставлена для совместимости.
`LoadDDS(const std::wstring& filepath, TextureDesc& desc, bool singleMip)`	Загрузка текстуры из DDS-файла.	Читает заголовок DDS, проверяет сигнатуру, определяет формат (DXT1/3/5), вычисляет размер данных, читает все mip-уровни в выделенный буфер. Поддерживает флаг `singleMip` для загрузки только первого уровня.
`LoadPNG(const std::wstring& filepath, TextureDesc& desc, bool singleMip)`	Загрузка PNG через WIC с генерацией mip-цепочки.	Создаёт фабрику WIC, декодер, получает кадр, конвертирует в 32-битный RGBA. Читает первый уровень, затем вручную генерирует последующие уровни усреднением 2x2. Все данные объединяются в один буфер.

DirectX11App.cpp

Глобальные структуры

Структура	Назначение	Поля
`GeomBuffer`	Константный буфер для геометрии объекта.	`m` – матрица модели, `normalM` – матрица нормалей, `shineSpeedTexIdNM` – параметры материала (блеск, индекс текстуры, наличие карты нормалей), `posAngle` – не используется.

Структура	Назначение	Поля
`SceneBuffer`	Константный буфер сцены (общие данные).	`vp` – матрица вида-проекции, `cameraPos` – позиция камеры, `lightInfo` – параметры освещения (количество источников, флаги), массив `lights[10]`, `ambientColor`.
`PostProcessBuffer`	Буфер для постпроцессинга.	`effectType` – тип эффекта (0..3), `padding[3]` – выравнивание.
`SmallSphereGeomBuffer`	Буфер для маленьких сфер (источников света).	`m` – матрица модели (трансляция в позицию источника), `color` – цвет источника.

Функции

Функция	Назначение	Принцип работы
`WinMain`	Точка входа.	Инициализирует COM, создаёт окно, DirectX, ImGui, загружает ресурсы, запускает главный цикл сообщений с вызовом `Render()` в простое.
`InitWindow`	Создание окна приложения.	Регистрирует класс окна, создаёт окно с заданными размерами, отображает его.
`WndProc`	Оконная процедура.	Обрабатывает сообщения: изменение размера, нажатия кнопок мыши/клавиш, колесо мыши. Передаёт события в ImGui.
`InitDirectX`	Инициализация Direct3D 11.	Создаёт устройство и цепочку обмена, настраивает задний буфер, буфер глубины, состояния растеризатора, глубины (reversed depth), смешивания.
`SetupBackBuffer`	Создание RTV заднего буфера и буфера глубины.	Получает задний буфер из цепочки обмена, создаёт для него RTV. Создаёт текстуру глубины D32_FLOAT и DSV.
`InitColorBuffer`	Создание промежуточного буфера цвета для постпроцессинга.	Создаёт текстуру, RTV и SRV для неё.
`InitBuffers`	Создание константного буфера сцены.	Буфер типа `D3D11_USAGE_DYNAMIC` для частого обновления.
`InitTerrain`	Генерация ландшафта из карты высот EXR.	Загружает EXR, строит сетку с шагом 8, создаёт вершины с позицией, текстурными координатами, вычисляет нормали и касательные усреднением по треугольникам. Создаёт вершинный, индексный и геометрический буферы.
`InitShaders`	Компиляция и создание основных шейдеров (для ландшафта).	Компилирует вершинный и пиксельный шейдеры из встроенных строк, создаёт шейдерные объекты и input layout.
`LoadTexture`	Загрузка основной текстуры (альбедо) из PNG.	Вызывает `LoadPNG`, проверяет поддержку формата, создаёт текстуру с mip-уровнями и SRV, создаёт сэмплер.

Функция	Назначение	Принцип работы
`LoadNormalMap`	Загрузка карты нормалей из DDS.	Вызывает `LoadDDS`, создаёт текстуру с учётом сжатого формата, создаёт SRV.
`LoadDetailTexture`	Загрузка текстуры деталей из PNG.	Аналогично `LoadTexture`, создаёт SRV.
`InitSmallSpheres`	Создание геометрии и шейдеров для визуализации источников света.	Генерирует сферу (8x8 сегментов), создаёт вершинный/индексный буферы, компилирует простые шейдеры (только закраска цветом), создаёт константные буферы для каждой сферы.
`InitPostProcess`	Создание шейдеров и буфера для постпроцессинга.	Компилирует вершинный шейдер (рисует треугольник на весь экран) и пиксельный шейдер с эффектами (сепия, холодный тон, ночное зрение). Создаёт динамический константный буфер для выбора эффекта.
`CreateSphere`	Заполнение массивов вершин и индексов для сферы.	Генерирует сферические координаты для заданного количества сегментов, заполняет индексы для треугольников.
`UpdateCamera`	Обновление матрицы вида и проекции, запись в буфер сцены.	Вычисляет позицию камеры по сферическим координатам, строит матрицу вида с вращающимся up-вектором. Для reversed depth создаёт проекцию с переставленными near/far. Заполняет `SceneBuffer` и обновляет его.
`UpdatePostProcessBuffer`	Обновление буфера постпроцессинга.	Отображает текущий эффект в константный буфер через Map/Unmap.
`RenderPostProcess`	Применение постпроцессинга.	Переключает цель рендера на задний буфер, устанавливает шейдеры постпроцессинга, привязывает промежуточную текстуру как ресурс, рисует треугольник.
`ResizeSwapChain`	Обработка изменения размера окна.	Освобождает старые ресурсы, изменяет размер цепочки обмена, пересоздаёт задний буфер и буфер цвета.
`RenderSmallSpheres`	Рендеринг сфер для каждого источника света.	Для каждого активного источника устанавливает матрицу трансляции и цвет, обновляет соответствующий константный буфер и рисует сферу.
`Render`	Основной рендер каждого кадра.	Очищает промежуточный буфер цвета и глубину, обрабатывает перемещение камеры по клавишам, обновляет буферы, рисует ландшафт, затем маленькие сферы, применяет постпроцессинг, рендерит ImGui, вызывает Present.
`Cleanup`	Освобождение всех ресурсов.	Последовательно вызывает `SAFE_RELEASE` для всех COM-объектов, завершает ImGui и COM.

Light.h

Элемент	Назначение	Принцип работы
`struct Light`	Представление	Содержит позицию (`XMFLOAT4`) и цвет (`XMFLOAT4`).

Элемент	Назначение	Принцип работы
	источника света.	
`CalculateColor`	Вычисление цвета пикселя по модели Фонга.	Принимает цвет объекта, нормаль, позицию, камеру, массив источников. Если `showNormals` – возвращает нормаль как цвет. Иначе: суммирует ambient, для каждого источника – диффузную и (если `shine>0`) зеркальную составляющие с учётом квадратичного затухания.

Texture.h

Элемент	Назначение	Принцип работы
`struct TextureDesc`	Описание загруженной текстуры.	Хранит `pitch`, `mipmapsCount`, `fmt`, `width`, `height`, `pData` (указатель на данные).
Прототипы `LoadDDS`, `LoadPNG`, `GetBytesPerBlock`, `DivUp`, `WCSToMBS`	Объявления функций из Texture.cpp.	_

Детальный конспект (описание ключевых блоков)

1. Инициализация приложения (WinMain)

- **Что делает:** Запускает приложение: создаёт окно, инициализирует DirectX, ImGui, загружает ресурсы, входит в цикл сообщений.
- **Как работает:**
  1. `CoInitializeEx` – инициализация COM для WIC.
  2. `InitWindow` – создание окна.
  3. `InitDirectX` – создание устройства, цепочки обмена, состояний.
  4. Инициализация ImGui (контекст, бэкенды Win32/DX11).
  5. Последовательный вызов функций инициализации: шейдеры, ландшафт, буферы, текстуры, карты нормалей, детали, маленькие сферы, постпроцессинг.
  6. Установка начальных параметров освещения.
  7. Главный цикл: если нет сообщений – вызывается `Render`.

2. Инициализация DirectX (InitDirectX)

- **Что делает:** Создаёт устройство, цепочку обмена, необходимые состояния.
- **Как работает:**
  - Заполняет `DXGI_SWAP_CHAIN_DESC` (формат `R8G8B8A8`, двойная буферизация, FLIP_DISCARD).
  - Пытается создать устройство с аппаратным драйвером, при неудаче – WARP.
  - Вызывает `SetupBackBuffer` и `InitColorBuffer` для создания RTV и буфера глубины.
  - Создаёт состояние растеризатора (без отсечения граней).
  - Создаёт состояние глубины для **reversed depth** (`D3D11_COMPARISON_GREATER`).
  - Создаёт состояние смешивания (отключено).

3. Загрузка текстур (LoadTexture, LoadNormalMap, LoadDetailTexture)

- **Что делает:** Загружает изображения из файлов и создаёт текстуры Direct3D.
- **Как работает:**

- Для PNG используется `LoadPNG` из Texture.cpp: через WIC декодирует в RGBA, генерирует mip-цепочку усреднением.
- Для DDS (карта нормалей) используется `LoadDDS`, поддерживающая сжатые форматы BC.
- Проверяется поддержка формата через `CheckFormatSupport`.
- Создаётся `ID3D11Texture2D` с массивом субресурсов (по одному на каждый mip-уровень).
- Создаётся шейдерное представление (`SRV`).
- Для основной текстуры создаётся также сэмплер с анизотропной фильтрацией.

4. Генерация ландшафта (InitTerrain)

- **Что делает:** Строит сетку ландшафта на основе карты высот в формате EXR.
- **Как работает:**
  - Загружает EXR с помощью библиотеки tinyexr.
  - Уменьшает разрешение сетки, беря каждый 8-й пиксель (для производительности), ограничивая до 256x256.
  - Для каждой вершины вычисляет позицию: x,z равномерно по ширине/глубине, y = значение высоты * масштаб.
  - Текстурные координаты: `(i/(size-1), j/(size-1))`.
  - Генерирует индексы для двух треугольников на ячейку.
  - Вычисляет нормали и касательные: для каждого треугольника вычисляется нормаль и касательная, затем усредняются по вершинам и нормализуются.
  - Создаёт вершинный, индексный и константный буферы.

5. Освещение и карты нормалей

- **Что делает:** Реализует модель освещения Фонга с несколькими источниками и поддержкой карт нормалей.
- **Как работает:**
  - В `SceneBuffer` передаются позиции/цвета источников и флаги.
  - В пиксельном шейдере:
    - Если `showNormals` включён, возвращается нормаль как цвет (отладка).
    - Иначе: начинается с ambient.
    - Для каждого источника: направление света, затухание `1/(dist²)`, диффузная составляющая.
    - Если есть блеск (`shine`), вычисляется зеркальная составляющая (отражённый вектор).
  - Если `useNormalMaps` и у объекта есть карта нормалей, то нормаль из карты преобразуется в касательное пространство через TBN-матрицу.

6. Постпроцессинг

- **Что делает:** Применяет один из цветовых эффектов к отрендеренной сцене.
- **Как работает:**
  - Сцена рендерится в промежуточную текстуру `m_pColorBuffer`.
  - Затем `RenderPostProcess` переключает цель на задний буфер, устанавливает специальный вершинный шейдер (рисует треугольник на весь экран) и пиксельный шейдер с эффектами.
  - Пиксельный шейдер читает цвет из промежуточной текстуры и в зависимости от `effectType` применяет матрицу сепии, холодный тон или ночное зрение.
  - Константный буфер постпроцессинга обновляется через `UpdatePostProcessBuffer`.

7. Управление камерой

- **Что делает:** Позволяет пользователю вращать камеру вокруг точки интереса (правая кнопка мыши) и перемещать точку интереса (клавиши WASD+QE).
- **Как работает:**
  - Камера представлена сферическими координатами: радиус `r`, угол возвышения `theta`, азимут `phi`, точка интереса `poi`.
  - В `WndProc` при движении мыши с зажатой правой кнопкой изменяются `theta` и `phi`.
  - Колесо мыши меняет `r`.
  - В `Render` при нажатых клавишах вычисляются векторы forward (от камеры к poi), up (вращающийся) и right, и точка интереса смещается в соответствующем направлении.
  - В `UpdateCamera` по текущим сферическим координатам вычисляется позиция камеры, строится матрица вида с корректным up-вектором, матрица проекции с reversed depth.

8. Визуализация источников света (маленькие сферы)

- **Что делает:** Рисует маленькие сферы в позициях источников света для наглядности.
- **Как работает:**
  - Создаётся низкополигональная сфера (8 сегментов) с помощью `CreateSphere`.
  - Вершинный буфер содержит только позиции.
  - Для каждого источника отдельный константный буфер обновляется матрицей трансляции (в позицию источника) и цветом.
  - Шейдеры просто передают цвет без освещения.

9. Интерфейс ImGui

- **Что делает:** Предоставляет панели для управления освещением и постпроцессингом.
- **Как работает:**
  - Инициализируется после создания устройства.
  - В `Render` после отрисовки сцены создаётся новый кадр ImGui, добавляются элементы управления (checkboxes, слайдеры, кнопки, списки).
  - Параметры привязаны к глобальным переменным (`m_lightCount`, `m_lights[]`, `m_postProcessEffect` и т.д.).
  - Рендерится поверх сцены.

10. Очистка ресурсов (Cleanup)

- **Что делает:** Освобождает все созданные COM-объекты и завершает библиотеки.
- **Как работает:** Последовательный вызов `SAFE_RELEASE` для всех указателей, завершение ImGui, вызов `CoUninitialize`.

Общее описание работы программы

Проект представляет собой приложение на DirectX 11, визуализирующее трёхмерный ландшафт с динамическим освещением и постобработкой. Ландшафт генерируется на основе карты высот в формате EXR, текстурируется с использованием карт нормалей и текстуры деталей. Реализована модель освещения Фонга с возможностью добавления/удаления источников света через интерфейс ImGui. Камера управляется мышью (вращение) и клавишами (перемещение). Присутствует постпроцессинг (сепия, холодный тон, ночное зрение), который применяется к финальному изображению. Используется техника reversed depth для повышения точности глубины.

Основные этапы работы:

1. Инициализация окна и DirectX.
2. Загрузка ресурсов: карта высот, текстуры, карты нормалей.
3. Создание геометрии ландшафта с вычислением нормалей и касательных.
4. Компиляция шейдеров (основные, для маленьких сфер, для постпроцессинга).
5. Главный цикл: обработка ввода, обновление камеры, рендеринг сцены в промежуточный буфер, применение постэффекта, вывод на экран.
6. Интерфейс ImGui позволяет в реальном времени менять параметры освещения и выбирать постэффект.

Ключевые особенности реализации:

- **Reversed depth:** буфер глубины очищается значением 0, функция сравнения `GREATER`, матрица проекции строится с переставленными near/far. Это улучшает распределение точности в перспективе.
- **Карты нормалей:** нормали из текстуры преобразуются в касательное пространство с помощью TBN-матрицы, построенной по касательным и нормалям вершин.
- **Многомипные текстуры:** для PNG генерируются вручную усреднением 2x2, для DDS загружаются все уровни из файла.
- **Постпроцессинг через один треугольник:** используется вершинный шейдер, который по `SV_VertexID` генерирует три вершины, покрывающие весь экран, что эффективнее прямоугольника.
- **Динамическое управление источниками:** можно добавлять до 10 источников, менять их позиции и цвета через ImGui.