Таблица функций и методов в порядке их появления в коде (task_7)

№	Функция/Метод	Назначение и принцип работы	Ссылка на лекцию
1	WinMain	Точка входа в приложение. Инициализирует окно, DirectX, ресурсы и запускает главный цикл.	Лекция 2, стр. 2–4
2	InitWindow	Создаёт и регистрирует окно с заданными размерами и стилем.	Лекция 2, стр. 5–6
3	InitDirectX	Инициализирует DirectX: создаёт устройство, контекст, своп-цепь, растеризатор, буфер глубины D32_FLOAT, состояния глубины и blend states.	Лекция 2, стр. 7–15; Лекция 8, стр. 5–20
4	D3D11CreateDeviceAndSwapChain	Создаёт логическое устройство GPU (ID3D11Device) и своп-цепь для вывода в окно.	Лекция 2, стр. 13–15
5	SetupBackBuffer	Создаёт render target view для back buffer’а своп-цепи и буфер глубины D32_FLOAT.	Лекция 2, стр. 28–29; Лекция 8, стр. 5–6
6	InitCube	Создаёт геометрию куба (вершинный и индексный буферы), компилирует шейдеры, создаёт input layout. Создаёт вершины с нормалями и касательными векторами.	Лекция 3, стр. 5–12; Лекция 6, стр. 11–13 Лекция 9, стр. 13, 25
7	D3DCompile	Компилирует шейдер из исходного кода в байт-код DXBC.	Лекция 3, стр. 17–22
8	CreateInputLayout	Описывает структуру вершинного буфера для передачи в вершинный шейдер.	Лекция 3, стр. 26–28
9	InitBuffers	Создаёт константные буферы для матриц модели (m) и вида, а также для второго куба. Константные буферы расширены для матрицы нормалей и shine. Создает константные буферы для инстансинга: буфер для всех инстансов (m_pGeomBufferInst) и буфер для видимых индексов (m_pGeomBufferInstVis).	Лекция 5, стр. 6–8 Лекция 9, стр. 18-20 Лекция 10, стр. 4, 31-33
11	LoadDDS	Читает DDS-файл, извлекает заголовок, формат, данные и мип-уровни.	Лекция 6, стр. 20–24
12	InitSkybox	Создаёт сферу для skybox, загружает cubemap-текстуру, компилирует шейдеры с reversed depth.	Лекция 6, стр. 34–42; Лекция 8, стр. 31–32
13	CreateSphere	Генерирует вершины и индексы для сферы методом параметризации.	Лекция 6, стр. 34
14	InitTransparentObjects	Инициализирует прозрачные	Лекция 8, стр.

№	Функция/Метод	Назначение и принцип работы	Ссылка на лекцию
		прямоугольники: геометрию, шейдеры, константные буферы, bounding boxes.	15–30
15	WndProc	Обрабатывает сообщения окна: изменение размера, вращение камеры, колесо мыши.	Лекция 2, стр. 2–4
16	ResizeSwapChain	Изменяет размер back buffer'а и буфера глубины при изменении окна.	Лекция 2, стр. 28
17	UpdateCamera	Обновляет матрицу вида и проекции (reversed depth), записывает в буфер. Расширен для передачи информации об освещении в буфер.	Лекция 5, стр. 21–24; Лекция 8, стр. 12–14 Лекция 9, стр. 12
18	Render	Основной цикл рендеринга: очистка, настройка конвейера, отрисовка непрозрачных объектов, skybox, прозрачных объектов. Добавлен рендеринг источников света и обновлён шейдерный пайплайн. использует DrawIndexedInstanced для отрисовки всех видимых инстансов за один вызов. Вызывает CullBoxes для отсечения.	Лекция 3, стр. 52–54; Лекция 6, стр. 28; Лекция 8, стр. 33 Лекция 9, стр. 11-12 Лекция 10, стр. 3, 34
19	RenderTransparentObjects	Рендерит прозрачные объекты с сортировкой от дальнего к ближнему, использованием blending и без записи глубины.	Лекция 8, стр. 25–30
20	IASetVertexBuffers / IASetIndexBuffer	Настраивают входной ассемблер: привязывают вершинный и индексный буферы.	Лекция 3, стр. 32
21	VSSetShader / PSSetShader	Устанавливают вершинный и пиксельный шейдеры в конвейер.	Лекция 3, стр. 23–24
22	PSSetShaderResources / PSSetSamplers	Передают текстуры и сэмплеры в пиксельный шейдер.	Лекция 6, стр. 27–28
23	VSSetConstantBuffers	Привязывают константные буферы к вершинному шейдеру.	Лекция 5, стр. 6
24	OMSetBlendState	Устанавливает состояние смешивания для прозрачных объектов.	Лекция 8, стр. 15–20
25	OMSetDepthStencilState	Устанавливает состояние глубины (reversed depth) для разных типов объектов.	Лекция 8, стр. 12–14, 27
26	DrawIndexed	Запускает отрисовку геометрии по индексам.	Лекция 3, стр. 54
27	UpdateSubresource	Копирует данные из CPU в GPU-буфер (например, матрицу модели).	Лекция 5, стр. 9–10
28	Map / Unmap	Отображают GPU-буфер в CPU-память для записи (динамический буфер сцены).	Лекция 5, стр. 43–45
29	Cleanup	Освобождает все созданные DirectX-ресурсы.	—
30	CreateSphere (вспом.)	Генерация геометрии сферы для skybox.	Лекция 6, стр. 34

№	Функция/Метод	Назначение и принцип работы	Ссылка на лекцию
31	GetBytesPerBlock / DivUp	Вспомогательные функции для работы с DXT-сжатыми текстурами.	Лекция 6, стр. 17–19
32	LoadNormalMap()	Загружает карту нормалей из DDS файла, создаёт текстуру и SRV	Лекция 9, стр. 21-27
33	InitSmallSpheres()	Инициализирует маленькие сферы для визуализации источников света	Лекция 9, стр. 11
34	RenderSmallSpheres()	Рендерит маленькие сферы, представляющие источники света	Лекция 9, стр. 11
35	CalculateLighting() (в шейдере)	Функция в пиксельном шейдере, реализующая модель Фонга	Лекция 9, стр. 14-16
36	CalculateColor() (в Light.h)	CPU-реализация освещения по Фонгу для справки	Лекция 9, стр. 14-16
37	ImGui окно управления освещением	Позволяет управлять источниками света, картами нормалей и т.д.	Лекция 9, стр. 11
38	InitGeomInstance	Инициализация параметров одного инстанса: позиция, блеск, ID текстуры, наличие карты нормалей, AABB	Лекция 10, стр. 24-25
39	CullBoxes	Основная функция frustum culling: строит пирамиду видимости, проверяет каждый инстанс на попадание, обновляет буфер видимых индексов	Лекция 10, стр. 23-30
40	BuildPlane	Строит уравнение плоскости по 4 точкам. Используется для построения плоскостей пирамиды видимости	Лекция 10, стр. 29
41	IsBoxInside	Проверяет, находится ли AABB внутри пирамиды видимости (frustum)	Лекция 10, стр. 30
42	LoadTextureArray	Загружает массив текстур (Brick.dds и Kitty.dds) для использования в инстансинге	Лекция 10, стр. 12-15
45	Шейдеры (обновлено)	Вершинный и пиксельный шейдеры изменены для работы с инстансингом: используют SV_InstanceID для доступа к данным каждого инстанса и буфер видимых индексов	Лекция 10, стр. 5-8, 16-18, 32-33

Сравнение кодов: что нового добавилось в task_7 (по сравнению с task_6)

Компонент	task_6 (старый)	task_7 (новый)	Что изменилось
Рендеринг множества объектов	Два куба, два вызова `DrawIndexed`	Множество кубов (до 100) с одним вызовом `DrawIndexedInstanced`	Использование инстансинга для уменьшения вызовов отрисовки
Константные буферы для инстансов	Отдельные буферы для каждого куба	Один буфер для всех инстансов (`m_pGeomBufferInst`) и буфер видимых индексов (`m_pGeomBufferInstVis`)	Структура данных для инстансинга и отсечения
Frustum Culling	Нет	Реализовано отсечение по пирамиде видимости (функции `CullBoxes`, `BuildPlane`, `IsBoxInside`)	Оптимизация: рисуются только видимые объекты
Управление инстансами через ImGui	Нет	Добавлено окно "Instances control" для управления количеством инстансов и отсечением	Возможность динамически добавлять/удалять инстансы, включать/выключать отсечение
Шейдеры	Шейдеры для двух кубов с освещением и картами нормалей	Шейдеры переписаны для инстансинга: использование `SV_InstanceID` и буфера видимых индексов	Поддержка множества инстансов с разными параметрами (текстура, блеск, нормали)
Массив текстур	Загрузка одной текстуры (Brick.dds)	Загрузка массива текстур (Brick.dds и Kitty.dds) и выбор текстуры по индексу в шейдере	Разные инстансы могут использовать разные текстуры из массива
AABB (Axis Aligned Bounding Box)	Нет	Добавлены AABB для каждого инстанса, используемые для отсечения	Упрощенная проверка на видимость
Динамическое добавление/удаление инстансов	Нет	Кнопки в ImGui для добавления/удаления инстансов, сброса к двум	Интерактивное управление количеством объектов

Детальный конспект (task_7)

Концепция	Реализация в коде	Как работает	Теория
Instancing	`DrawIndexedInstanced` в функции `Render` и шейдеры с `SV_InstanceID`	Один вызов отрисовки рисует множество одинаковых объектов с разными параметрами (матрицы, текстуры и т.д.)	Лекция 10, стр. 2-3
Массив текстур	`LoadTextureArray` загружает две текстуры в массив, в шейдере выборка по индексу: `colorTexture.Sample(..., float3(uv, texIndex))`	Разные инстансы могут использовать разные текстуры из одного массива	Лекция 10, стр. 11-15
Константный буфер для инстансов	`m_pGeomBufferInst` - буфер, содержащий данные для всех инстансов (матрицы, нормали, параметры)	Шейдер обращается к этому буферу по индексу инстанса (полученному через `SV_InstanceID`)	Лекция 10, стр. 4-8
Frustum Culling	Функции `CullBoxes`, `BuildPlane`, `IsBoxInside` вычисляют видимые инстансы и записывают их индексы в `m_pGeomBufferInstVis`	На CPU отсекаются инстансы, не попадающие в поле зрения, и в шейдер передается только список видимых	Лекция 10, стр. 23-30
AABB (Axis Aligned Bounding Box)	Для каждого инстанса при инициализации вычисляется AABB (куб 1x1x1) и сохраняется в `m_geomBBs`	Упрощенная геометрия для быстрой проверки на пересечение с пирамидой видимости	Лекция 10, стр. 24-25
Буфер видимых индексов	`m_pGeomBufferInstVis` - буфер, содержащий индексы видимых инстансов	В шейдере по `SV_InstanceID` получаем индекс из этого буфера, затем по нему данные из `m_pGeomBufferInst`	Лекция 10, стр. 31-33
Универсальный шейдер с развилкой	В пиксельном шейдере проверка `geomBuffer[idx].shineSpeedTexIdNM.w` для определения, использовать ли карту нормалей	Динамическое ветвление в шейдере в зависимости от параметров инстанса	Лекция 10, стр. 20-21
Управление инстансами через ImGui	Окно "Instances control" с кнопками "Add Instance", "Remove Instance", "Reset to 2" и чекбоксом "Frustum Culling"	Позволяет интерактивно управлять количеством инстансов и включать/выключать отсечение	Лекция 10, стр. 35 (задание)

Ключевые концепции для защиты (task_7)

Концепция	Что добавилось в task_7	Как реализовано	Теория
Instancing	Один вызов `DrawIndexedInstanced` вместо многих `DrawIndexed`	Массив константных буферов + `SV_InstanceID` в шейдере	Лекция 10, стр. 2-8
Массив текстур	Несколько текстур в одном ресурсе	`Texture2DArray` + выборка по третьей координате	Лекция 10, стр. 11-15
Frustum Culling	Отсечение невидимых объектов	Построение пирамиды	Лекция 10,

Концепция	Что добавилось в task_7	Как реализовано	Теория
	на CPU	видимости + проверка AABB	стр. 23-30
AABB (Axis Aligned Bounding Box)	Простой параллелепипед для каждого объекта	Две точки (min и max) для быстрой проверки	Лекция 10, стр. 24-25
Буфер видимых индексов	Второй константный буфер для передачи видимых индексов	`GeomBufferInstVis` + индексация в шейдере	Лекция 10, стр. 31-33
Динамическое управление инстансами	Добавление/удаление объектов через UI	ImGui интерфейс + обновление буферов	Лекция 10, стр. 35

Итог: **task_7 добавляет**

1. **Instancing** - оптимизацию рендеринга множества одинаковых объектов
2. **Массив текстур** - возможность использования разных текстур для разных инстансов
3. **Frustum Culling** - отсечение объектов вне поля зрения для повышения производительности
4. **AABB** - упрощенные bounding box для быстрой проверки видимости
5. **Интерактивное управление** - динамическое добавление/удаление инстансов через ImGui

Основное достижение: Значительное повышение производительности при рендеринге множества объектов за счет сокращения вызовов отрисовки и отсечения невидимых объектов, что соответствует профессиональным подходам в игровой графике.