GXLib ― プロジェクト指令書

> **この文書は、本プロジェクトに携わる AI（Claude）が作業の方向性を見失わないための最上位ドキュメントである。 実装判断に迷った時、スコープが曖昧な時、優先度を決める時、必ずこの文書の「第1章 なぜ作るのか」に立ち返ること。**

■ 第1章：なぜ作るのか ― 本プロジェクトの存在意義

1.1 解決すべき問題（これが全ての出発点）

DXライブラリは日本のゲーム開発教育・ホビーシーンで広く使われているが、**以下の根本的な欠陥**を抱えている：

1. **DirectX 11 止まり** ― DirectX 12 に非対応で、モダン GPU の性能を活かせない
2. **見た目がショボい・安っぽい** ― PBR もHDR も標準では使えず、ビジュアル品質に天井がある
3. **ポストエフェクトが全て自前実装** ― Bloom 1つ入れるだけで大量のコードが必要で、クソめんどくさい
4. **描画レイヤー管理がない** ― 描画順を制御する標準的な仕組みがなく、煩雑なコードを強いられる
5. **GUIシステムが存在しない** ― メニューやHUDを作る標準手段がない

これらの問題を「根本から」解決する。パッチではなく、ゼロから作り直す。

1.2 本プロジェクトの正体

DXライブラリの完全上位互換フレームワークを、DirectX 12 ベースでゼロから構築する。

「上位互換」とは：

- DXライブラリにある機能は **全て** 備える（一つ残らず）
- DXライブラリに **ない** 機能を大量に追加する（後述）
- DXライブラリの使いやすさは維持しつつ、表現力を桁違いに引き上げる

1.3 絶対に達成すべきゴール（完成条件）

以下の5条件を「全て」満たした時のみ、このプロジェクトは完成とする：

#	完成条件	妥協不可の理由
G1	DXライブラリの全APIカテゴリを網羅	「上位互換」を名乗る以上、抜けがあってはならない
G2	ポストエフェクトパイプラインが標準搭載され、設定ファイルでON/OFF可能	これがないと「ショボい見た目」問題が解決しない
G3	描画レイヤーシステムが動作し、レイヤー単位でブレンド・エフェクトが可能	DXLibの最大の不満の一つを解消するため
G4	XMLベースのGUIシステムでメニュー・HUD・エディタUIが構築可能	GUIなしのフレームワークは片手落ち
G5	サンプルプロジェクト群（2Dゲーム、3Dゲーム、GUIデモ）が動作する	動くサンプルがなければ使い物にならない

1.4 作業中に常に意識すべき設計理念

実装のあらゆる判断は、以下の理念に照らして行うこと：

1. **DXライブラリ互換** — 既存DXLibユーザーが違和感なく移行できるAPI設計
2. **モダンレンダリング** — PBR、HDR、ポストエフェクトは「標準」。オプションではなくデフォルト
3. **レイヤードアーキテクチャ** — 低レベルAPI（D3D12直接操作）と高レベルAPI（DXLib風の簡単操作）の両方を提供
4. **宣言的GUI** — XMLとCSS-likeスタイルシートでUIを記述。C++コードでレイアウトを書かせない
5. **拡張性** — プラグインやカスタムシェーダーで自由に拡張できる構造

■ 第2章：何を作るのか ― 技術仕様

2.1 技術スタック

項目	選定	選定理由
グラフィックスAPI	DirectX 12 (D3D12)	DXLibがDX11止まりという問題の直接的解答
シェーダー言語	HLSL (Shader Model 6.x)	DX12ネイティブ、最新機能が使える

シェーダーコンパイラ	DXC (DirectX Shader Compiler)	SM6.x対応の公式コンパイラ
ビルドシステム	CMake + vcpkg	業界標準、依存管理が楽
言語	C++20	コンセプト、ranges、コルーチン等を活用
オーディオ	XAudio2 / WASAPI	Windows標準、低遅延
入力	XInput + Raw Input + DirectInput(後方互換)	全入力デバイスカバー
物理	内製2D + Jolt Physics(3D、オプション)	軽量2D物理は自前、3Dは実績あるライブラリ
GUI記述	XML + CSS-like スタイルシート	Web開発者にも馴染みやすい宣言的UI
スクリプト	Lua（オプション）	軽量で組み込みやすい
テスト	Google Test + GPUベース回帰テスト	品質保証

2.2 命名規則（プロジェクト全体で統一）

対象	規則	例
名前空間	`GX::`	`GX::Graphics`, `GX::Audio`
クラス	PascalCase	`SpriteBatch`, `RenderLayer`
メソッド	PascalCase	`DrawSprite()`, `LoadTexture()`
メンバ変数	`m_camelCase`	`m_width`, `m_renderTarget`
定数	`k_PascalCase`	`k_MaxLayers`, `k_DefaultFOV`
ファイル名	PascalCase.h/.cpp	`SpriteBatch.h`, `SpriteBatch.cpp`

■ 第3章：DXライブラリ全機能マッピング ー 漏れなく網羅するための一覧

> この表がチェックリストである。全行に対応実装が完了して初めてゴールG1を達成する。

3.1 システム系

DXLib機能	DXLib関数例	GXLib対応方針
ウィンドウ管理	`DxLib_Init`, `SetGraphMode`, `ChangeWindowMode`	Win32ウィンドウ + DXGI SwapChain
メッセージ処理	`ProcessMessage`	内部メッセージループ + イベントコールバック
フルスクリーン切替	`ChangeWindowMode`	DXGIフルスクリーン + ボーダーレス
解像度変更	`SetGraphMode`	動的スワップチェーンリサイズ
DPI対応	（なし）	**【新規】** Per-Monitor DPI V2
マルチウィンドウ	（なし）	**【新規】** マルチウィンドウレンダリング
タイマー	`GetNowCount`, `GetNowHiPerformanceCount`	`QueryPerformanceCounter` ベース
FPS制御	`SetWaitVSyncFlag`	VSync + フレームレートリミッター
ログ出力	`printfDx`, `ErrorLogAdd`	構造化ログ（レベル別、ファイル出力）

3.2 描画系 ― 2D

DXLib機能	DXLib関数例	GXLib対応方針
画像読込・描画	`LoadGraph`, `DrawGraph`	テクスチャ管理 + スプライトバッチ
画像分割読込	`LoadDivGraph`	スプライトシート + アトラスパッカー
画像回転拡大	`DrawRotaGraph`, `DrawExtendGraph`	Transform2D指定描画
画像ブレンド	`SetDrawBlendMode`	ブレンドステート（加算、乗算、α等）

画像輝度設定	`SetDrawBright`	シェーダーパラメータ / カラーマスク
図形描画	`DrawLine`, `DrawBox`, `DrawCircle`, `DrawTriangle`	プリミティブバッチレンダラー
アンチエイリアス図形	`DrawLineAA`, `DrawCircleAA`	MSAA + シェーダーベースAA
ピクセル操作	`GetPixelSoftImage`, `DrawPixelSoftImage`	CPU側ソフトイメージ + Readback
グラフィックフィルタ	`GraphFilter`（モノ、ガウス、明度等）	**ポストエフェクトパイプラインで代替**
描画先変更	`SetDrawScreen`, `MakeScreen`	RenderTargetシステム
描画レイヤー	（なし）	**【新規・重要】** レイヤースタック（Z-order、ブレンド、エフェクト）
スプライトアニメーション	（なし）	**【新規】** アニメーションコントローラー

3.3 描画系 — 3D

DXLib機能	DXLib関数例	GXLib対応方針
3Dモデル読込	`MV1LoadModel`	glTF / FBX / OBJローダー
モデル描画	`MV1DrawModel`	メッシュレンダラー + インスタンシング
モデルアニメーション	`MV1AttachAnim`, `MV1SetAttachAnimTime`	スケルタルアニメーション + ブレンドツリー
モデル衝突判定	`MV1CollCheck_Sphere`, `MV1CollCheck_Line`	メッシュコリジョン
カメラ制御	`SetCameraPositionAndTarget_UpVecY`	Cameraコンポーネント（Perspective / Ortho）
ライティング	`SetLightDirection`, `SetLightDifColor`	**【新規】PBRライティング**（Directional, Point, Spot, Area）
		【新規】PBRマテリアル（Albedo,

マテリアル	`MV1SetMaterialDifColor`	Normal, Metallic, Roughness, AO）
シャドウマップ	`SetShadowMapDrawArea`	**【新規】CSM** + PCF/VSM
フォグ	`SetFogEnable`, `SetFogColor`	ボリューメトリックフォグ / 距離フォグ
3D図形	`DrawSphere3D`, `DrawCone3D`	プリミティブメッシュ生成
Zバッファ	`SetUseZBuffer3D`	深度ステート制御
環境マップ	（なし）	**【新規】** キューブマップ / IBL反射
PBR	（なし）	**【新規】** Cook-Torrance BRDF
スカイボックス	（なし）	**【新規】** HDRスカイボックス / プロシージャル空
地形	（なし）	**【新規】** ハイトマップ地形 + LOD

3.4 ポストエフェクト（ゴールG2に直結 ― 全て新規）

エフェクト	実装方式
Bloom	Dual Kawase / ガウシアン ダウンサンプリング
Tonemapping	ACES / Reinhard / Uncharted 2 / AgX
HDR	浮動小数テクスチャ (R16G16B16A16_FLOAT)
SSAO	GTAO / HBAO
被写界深度 (DoF)	Bokeh DoF（六角形/円形）
モーションブラー	Per-Object / Cameraベース
カラーグレーディング	3D LUT
FXAA / TAA	FXAA 3.11 / TAA
ビネット	シェーダーベース
色収差	シェーダーベース
スクリーンスペースリフレクション	SSR (Hi-Z trace)

ボリューメトリックライト	レイマーチング
輪郭線（トゥーン）	Sobel / 法線・深度エッジ検出

3.5 シェーダー

DXLib機能	DXLib関数例	GXLib対応方針
頂点シェーダー	`LoadVertexShader`	HLSL SM6.xコンパイル + ホットリロード
ピクセルシェーダー	`LoadPixelShader`	同上
定数バッファ	`SetShaderConstantReg`	CBV自動バインド
シェーダー描画	`SetUseVertexShader`	マテリアルシステム統合
コンピュートシェーダー	（なし）	**【新規】** CSパイプライン
シェーダーホットリロード	（なし）	**【新規】** ファイル監視による即時反映
シェーダーバリアント	（なし）	**【新規】** プリプロセッサ定義による分岐管理

3.6 サウンド

DXLib機能	DXLib関数例	GXLib対応方針
サウンド読込・再生	`LoadSoundMem`, `PlaySoundMem`	XAudio2ベース
BGMストリーミング	`PlayMusic`	ストリーミングデコード
音量制御	`ChangeVolumeSoundMem`	デシベルベース音量制御
パン制御	`ChangePanSoundMem`	ステレオパン
再生速度	`SetFrequencySoundMem`	ピッチシフト
3Dサウンド	`Set3DPositionSoundMem`	**3D空間音響**（HRTFオプション）
対応フォーマット	WAV, OGG, MP3	WAV, OGG, MP3, FLAC, OPUS

サウンドミキサー	（なし）	**【新規】** バス・エフェクトチェーン
リアルタイムエフェクト	（なし）	**【新規】** リバーブ、EQ、コンプレッサー

3.7 入力

DXLib機能	DXLib関数例	GXLib対応方針
キーボード	`CheckHitKey`, `GetHitKeyStateAll`	Raw Input
マウス	`GetMousePoint`, `GetMouseInput`	Raw Input + カーソル管理
マウスホイール	`GetMouseWheelRotVol`	WM_MOUSEWHEEL
ジョイパッド	`GetJoypadInputState`, `GetJoypadAnalogInput`	XInput + DirectInputフォールバック
ジョイパッド振動	`StartJoypadVibration`	XInputバイブレーション
タッチ	`GetTouchInputNum`	WM_TOUCH / WM_POINTER
入力マッピング	（なし）	**【新規】** アクションマップ（設定ファイルで再マップ可能）
入力バッファリング	（なし）	**【新規】** 格闘ゲーム向けコマンドバッファ
デッドゾーン設定	（なし）	**【新規】** アナログスティック デッドゾーン

3.8 文字描画

DXLib機能	DXLib関数例	GXLib対応方針
文字列描画	`DrawString`, `DrawFormatString`	**SDFフォントレンダリング**
フォント作成	`CreateFontToHandle`	DirectWrite + フォントアトラス
文字列幅取得	`GetDrawStringWidth`	テキスト計測API

文字コード	`SetUseCharCodeFormat`	UTF-8 / UTF-16
リッチテキスト	（なし）	**【新規】** カラー、サイズ混在テキスト
テキストレイアウト	（なし）	**【新規】** ワードラップ、行間、カーニング
ビットマップフォント	（なし）	**【新規】** BMFont形式対応

3.9 ネットワーク

DXLib機能	DXLib関数例	GXLib対応方針
TCP接続	`ConnectNetWork`, `NetWorkSend`	Winsock2非同期TCP
UDP	`MakeUDPSocket`, `NetWorkSendUDP`	Winsock2 UDP
HTTP	`GetHTTP`, `GetHTTPRes`	WinHTTP / libcurl
WebSocket	（なし）	**【新規】** WebSocketクライアント
非同期IO	（なし）	**【新規】** IOCPベース非同期

3.10 ファイル・アーカイブ

DXLib機能	DXLib関数例	GXLib対応方針
ファイル読み書き	`FileRead_open`, `FileRead_gets`	std::filesystem + 非同期IO
DXAアーカイブ	`SetDXArchiveExtension`	**カスタムアーカイブ**（AES暗号化対応）
メモリ上読込	`CreateGraphFromMem`	メモリストリーム対応
アセットホットリロード	（なし）	**【新規】** ファイル監視による即時反映
非同期アセット読込	（なし）	**【新規】** バックグラウンドロード + プログレス

3.11 算術・衝突判定

DXLib機		

能	DXLib関数例	GXLib対応方針
ベクトル演算	`VGet`, `VAdd`, `VCross`	SIMD最適化数学ライブラリ（DirectXMathラップ）
行列演算	`MGetIdent`, `MMult`	Matrix4x4クラス
衝突判定	`HitCheck_Sphere_Sphere`, `HitCheck_Line_Triangle`	衝突判定ユーティリティ
2D物理	（なし）	**【新規】** 簡易2D物理エンジン
空間分割	（なし）	**【新規】** Quadtree / Octree / BVH

3.12 動画・マスク

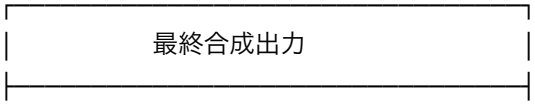
DXLib機能	DXLib関数例	GXLib対応方針
動画再生	`PlayMovie`, `OpenMovieToGraph`	Media Foundationデコード→テクスチャ
動画フレーム取得	`SeekMovieToGraph`, `TellMovieToGraph`	フレーム単位シーク
マスク描画	`CreateMaskScreen`, `DrawMask`	ステンシルバッファ + マスクテクスチャ
マスク図形	`DrawFillMask`, `DrawCircleMask`	ステンシル描画

■ 第4章：新規システム詳細設計

> **この章は、DXライブラリに「ない」ものを定義する。ここがこのフレームワークの差別化ポイントであり、存在意義である。「第1章の問題」を解決するのがこの章の設計。**

4.1 描画レイヤーシステム（ゴールG3に直結）

なぜ必要か: DXライブラリには描画順を管理する仕組みがなく、 `DrawGraph` の呼び出し順でしか制御できない。これはゲームが複雑になると地獄になる。



```
│  Layer: UI (Z: 1000)            │ ← ポストエフェクト適用外
│  Layer: HUD (Z: 900)            │ ← ポストエフェクト適用外
│  Layer: PostFX                  │ ← ポストエフェクトパイプライン
│  Layer: Particles (Z: 500)      │
│  Layer: Characters (Z: 400)     │
│  Layer: World (Z: 100)          │ ← PBRレンダリング
│  Layer: Background (Z: 0)       │
│  Layer: Skybox (Z: -1000)       │
└─────────────────────────────────┘
```

レイヤーが持つ機能:

- 個別のRenderTargetを保持
- レイヤー単位のブレンドモード設定（通常、加算、乗算、スクリーン...）
- レイヤー単位の不透明度制御
- レイヤー単位のポストエフェクト適用可否
- レイヤーのカメラ独立設定（パララックススクロール等）
- レイヤーのソートモード設定（Z-sort, Y-sort, 挿入順）

4.2 ポストエフェクトパイプライン（ゴールG2に直結）

なぜ必要か: DXライブラリで見た目がショボくなる最大の原因。Bloomすら自前で書くのはクソめんどくさい。これを設定ファイル1つで制御できるようにする。

```
Scene → [HDR RenderTarget]
          ↓
     ┌─ SSAO ──────────┐
     ├─ SSR ───────────┤
     ├─ MotionBlur ────┤
     ├─ DoF ───────────┤    ← 設定ファイルで ON/OFF、順序変更可能
     ├─ Bloom ─────────┤
     ├─ ColorGrading ──┤
     ├─ Tonemapping ───┤
     ├─ FXAA/TAA ──────┤
     ├─ Vignette ──────┤
     └─ ChromaticAberr ┘
          ↓
     [LDR BackBuffer] → Present
```

設定ファイル例 (post_effects.json):

```
{
```

```
  "postEffects": {
    "bloom": { "enabled": true, "threshold": 1.0, "intensity": 0.8, "radius": 4 }
    "tonemapping": { "enabled": true, "operator": "ACES", "exposure": 1.2 },
    "ssao": { "enabled": true, "radius": 0.5, "bias": 0.025, "samples": 32 },
    "dof": { "enabled": false },
    "fxaa": { "enabled": true }
  }
}
```

4.3 XML-GUIシステム（ゴールG4に直結）

なぜ必要か: DXライブラリにはGUIがないため、メニュー画面1つ作るにも大量のC++コードが必要。XMLで宣言的に書けるようにすることで、UIの構築速度を10倍にする。

XMLレイアウト定義

```
<!-- ui/main_menu.xml -->
<Window id="mainMenu" width="100%" height="100%">
  <Panel id="centerPanel" layout="vertical" align="center" valign="center"
         background="#00000080" padding="20" cornerRadius="8">

    <Text id="title" text="My Game" fontSize="48" fontFamily="GameFont"
          color="#FFFFFF" shadow="2,2,#000000" />

    <Spacer height="40" />

    <Button id="btnStart" width="300" height="60" text="ゲーム開始"
            class="menuButton" onClick="onStartGame" />
    <Button id="btnOption" width="300" height="60" text="オプション"
            class="menuButton" onClick="onOpenOptions" />
    <Button id="btnExit" width="300" height="60" text="終了"
            class="menuButton" onClick="onExit" />
  </Panel>
</Window>
```

CSS-likeスタイルシート

```
/* ui/styles/menu.gss */
.menuButton {
    background: linear-gradient(#4A90D9, #357ABD);
    color: #FFFFFF;
    fontSize: 24;
    fontFamily: "GameFont";
    cornerRadius: 6;
```

```
    border: 2px solid #2A5A8E;
    transition: background 0.2s;
}
.menuButton:hover {
    background: linear-gradient(#5BA0E9, #4590DD);
    transform: scale(1.05);
}
.menuButton:pressed {
    background: linear-gradient(#2A5A8E, #1A4A7E);
    transform: scale(0.95);
}
```

C++バインド

```
auto ui = GX::GUI::Load("ui/main_menu.xml", "ui/styles/menu.gss");
ui->Bind("onStartGame", [&]() { scene.TransitionTo<GameScene>(); });
ui->Bind("onOpenOptions", [&]() { ui->Show("optionsPanel"); });
ui->Bind("onExit", [&]() { GX::System::Exit(); });

// 動的操作
ui->Find<GX::GUI::Text>("title")->SetText("Updated Title");
ui->Find<GX::GUI::Button>("btnStart")->SetEnabled(false);
```

GUIウィジェット一覧（全て実装すること）

Window, Panel, Text, Button, Image, TextInput, Slider, CheckBox, RadioGroup/Radio, DropDown, ListView, ScrollView, ProgressBar, TabView, Dialog, Canvas, Spacer

---

■ 第5章：アーキテクチャ ー ディレクトリ構成

> **この構成に従ってファイルを配置すること。勝手にディレクトリ構造を変えない。**

```
GXLib/
├── Core/                    # コアシステム
│   ├── Application.h/cpp    # アプリケーションライフサイクル
│   ├── Window.h/cpp         # ウィンドウ管理
│   ├── Timer.h/cpp          # 高精度タイマー
│   ├── Logger.h/cpp         # ログシステム
│   ├── Memory/              # メモリ管理（Pool, Stack, Linear）
│   ├── Event/               # イベントシステム（EventBus, Delegate）
│   └── Config/              # 設定管理（JSON/INI）
│
```

```
├── Graphics/                  # 描画エンジン
│   ├── Device/                # D3D12デバイス管理
│   │   ├── GraphicsDevice.h   # デバイス初期化・管理
│   │   ├── SwapChain.h        # スワップチェーン
│   │   ├── CommandQueue.h     # コマンドキュー
│   │   ├── CommandList.h      # コマンドリスト
│   │   ├── DescriptorHeap.h   # デスクリプタヒープ管理
│   │   ├── Fence.h            # GPU同期
│   │   └── GPUResource.h      # リソース基底クラス
│   ├── Pipeline/              # パイプライン管理
│   │   ├── PipelineState.h    # PSO管理
│   │   ├── RootSignature.h    # ルートシグネチャ
│   │   ├── Shader.h           # シェーダーコンパイル・管理
│   │   └── ShaderLibrary.h    # シェーダーライブラリ + ホットリロード
│   ├── Resource/              # GPUリソース
│   │   ├── Texture.h          # テクスチャ
│   │   ├── Buffer.h           # 頂点/インデックス/定数バッファ
│   │   ├── RenderTarget.h     # レンダーターゲット
│   │   ├── DepthBuffer.h      # 深度バッファ
│   │   └── SamplerState.h     # サンプラー
│   ├── 2D/                    # 2D描画
│   │   ├── SpriteBatch.h      # スプライトバッチ
│   │   ├── SpriteSheet.h      # スプライトシート
│   │   ├── PrimitiveBatch.h   # 図形バッチ
│   │   ├── TextRenderer.h     # テキスト描画(SDF)
│   │   ├── FontManager.h      # フォント管理
│   │   └── Animation2D.h      # スプライトアニメーション
│   ├── 3D/                    # 3D描画
│   │   ├── Mesh.h / Model.h / SkeletalAnim.h
│   │   ├── Camera.h / Light.h / Material.h
│   │   ├── ShadowMap.h / Skybox.h / Terrain.h
│   │   └── Primitive3D.h
│   ├── Layer/                 # 描画レイヤー
│   │   ├── RenderLayer.h
│   │   ├── LayerStack.h
│   │   └── LayerCompositor.h
│   ├── PostFX/                # ポストエフェクト
│   │   ├── PostEffectPipeline.h
│   │   ├── PostEffect.h       # 基底クラス
│   │   ├── Bloom.h / Tonemapping.h / SSAO.h / DoF.h
│   │   ├── MotionBlur.h / ColorGrading.h / FXAA.h / TAA.h
│   │   ├── Vignette.h / ChromaticAberration.h
│   │   ├── SSR.h / VolumetricLight.h / OutlineEffect.h
│   │   └── （各.cppは対応する.hと同名）
│   └── Renderer.h             # レンダラー統合
│
├── Audio/                     # オーディオ
```

```
│   ├── AudioDevice.h / Sound.h / SoundPlayer.h
│   ├── MusicPlayer.h / AudioMixer.h
│   ├── Audio3D.h / AudioEffect.h
│   └── （各.cpp）
│
├── Input/                    # 入力
│   ├── Keyboard.h / Mouse.h / Gamepad.h / Touch.h
│   ├── InputManager.h / ActionMap.h
│   └── （各.cpp）
│
├── GUI/                      # GUIシステム
│   ├── GUIManager.h          # 統合管理
│   ├── GUIParser.h           # XMLパーサー
│   ├── GUIStyleSheet.h       # スタイルシートエンジン
│   ├── GUIRenderer.h         # GUI描画
│   ├── GUILayout.h           # レイアウトエンジン（Flexbox-like）
│   ├── GUIAnimation.h        # UIアニメーション
│   ├── GUIEvent.h            # UIイベント伝搬
│   └── Widgets/              # ウィジェット群
│       ├── Widget.h（基底） / Panel.h / Button.h / Text.h
│       ├── TextInput.h / Image.h / Slider.h / CheckBox.h
│       ├── RadioButton.h / DropDown.h / ListView.h
│       ├── ScrollView.h / ProgressBar.h / TabView.h
│       ├── Dialog.h / Canvas.h
│       └── （各.cpp）
│
├── IO/                       # ファイル・ネットワーク
│   ├── FileSystem.h / Archive.h / AsyncLoader.h
│   ├── Network/ (TCPSocket, UDPSocket, HTTPClient, WebSocket)
│   └── Serialization.h
│
├── Math/                     # 数学ライブラリ
│   ├── Vector2.h / Vector3.h / Vector4.h
│   ├── Matrix4x4.h / Quaternion.h / Color.h
│   ├── MathUtil.h / Random.h
│   └── Collision/ (AABB, Sphere, Ray, Frustum, CollisionUtil)
│
├── Physics/                  # 物理（オプション）
│   ├── Physics2D.h / PhysicsWorld.h
│   └── （各.cpp）
│
├── Movie/                    # 動画再生
│   └── MoviePlayer.h / MoviePlayer.cpp
│
├── Compat/                   # DXライブラリ互換レイヤー
│   ├── DxLibCompat.h         # DXLib関数名互換API
│   └── DxLibTypes.h          # 型変換
```

```
│
└── Utility/                    # ユーティリティ
    ├── StringUtil.h / PathUtil.h / Hash.h
    ├── ThreadPool.h / Profiler.h
    └── （各.cpp）
```

■ 第6章：実装フェーズ ー 順序と各フェーズの完了条件

> **必ずフェーズ順に実装すること。前のフェーズが完了条件を満たさないまま次に進んではならない。 各フェーズの「成果物」が動作確認できて初めて次に進む。**

Phase 0: D3D12基盤構築

完了条件: 色付き三角形がウィンドウに描画される

- CMakeプロジェクト構成 + vcpkg依存管理
- Win32ウィンドウ作成・メッセージループ
- D3D12デバイス初期化
  - Factory, Device, CommandQueue作成
  - SwapChain作成（ダブルバッファリング）
  - DescriptorHeap管理クラス
  - Fence による CPU-GPU同期
  - CommandAllocator / CommandList管理
- パイプラインステート基盤
  - RootSignatureビルダー
  - PSOビルダー
  - DXCによるシェーダーコンパイル
- 三角形描画（Hello Triangle）
- フレームタイミング・FPS制御
- ログシステム

Phase 1: 2D描画エンジン

完了条件: DXLibの2Dサンプルが全て再現可能

- テクスチャ管理（WIC/stb_image、キャッシュ、ミップマップ）
- スプライトバッチ（DrawGraph, DrawRotaGraph, DrawRectGraph, DrawExtendGraph, DrawModiGraph相当 + ブレンド + カラー）
- スプライトシート・アトラス（LoadDivGraph相当 + アニメーション再生）
- プリミティブバッチ（全図形 + AA対応）
- RenderTarget（MakeScreen / SetDrawScreen / GetDrawScreenGraph相当）
- ソフトイメージ（CPU側ピクセル操作 + GPU転送）
- カメラ2D

Phase 2: テキスト・入力・サウンド

完了条件: 音が鳴り、操作可能な2Dゲームが作れる

- **テキスト:** DirectWrite → SDFフォントアトラス → SDFレンダリング、DrawString/DrawFormatString相当、フォントハンドル、文字幅取得、リッチテキスト、BMFont
- **入力:** キーボード(Raw Input)、マウス(Raw Input)、ゲームパッド(XInput+DInput)、アクションマッピング
- **サウンド:** XAudio2初期化、WAV/OGG/MP3デコード、再生制御、音量/パン/速度、BGMストリーミング、3Dサウンド、ミキサー、エフェクト

Phase 3: 3D描画エンジン

完了条件: PBR litシーンが描画でき、モデルがアニメーションする

- モデルローダー（glTF 2.0 必須、FBX/OBJオプション）
- PBRレンダリング（Cook-Torrance BRDF、マテリアルシステム、ライティング、IBL）
- シャドウ（CSM + PCF、ポイントライト/スポットライト シャドウ）
- カメラ（Perspective/Ortho、FPS/TPS/Free）
- スカイボックス（キューブマップ + HDR + プロシージャル）

- フォグ、3Dプリミティブ、インスタンシング、LOD、地形

Phase 4: ポストエフェクトパイプライン

完了条件: 全エフェクトがON/OFFでき、ビジュアルが劇的に向上する。JSON設定ファイルで制御可能。

- PostEffectPipelineフレームワーク（チェーン管理、中間RT自動管理、JSON設定）
- Bloom, Tonemapping（ACES+自動露出）, SSAO(GTAO), DoF, MotionBlur
- ColorGrading(3D LUT), FXAA, TAA
- Vignette, ChromaticAberration, SSR, VolumetricLight, OutlineEffect

---

Phase 5: 描画レイヤーシステム

完了条件: 背景→ゲーム→UIが独立レイヤーで管理され合成される

- RenderLayerクラス（個別RT、Z-order、ブレンド、不透明度、カメラ）
- LayerStack管理（追加/削除/並替/グループ）
- LayerCompositor（合成シェーダー、レイヤー単位PostFX制御、マスクレイヤー）
- ステンシルマスクシステム（DXLibのマスク機能再現）

Phase 6: XML-GUIシステム

完了条件: XMLとCSSでメニュー・HUDが構築でき、C++でイベント処理可能

- パーサー（XML + CSS-likeスタイルシート → ウィジェットツリー構築）
- レイアウトエンジン（ボックスモデル、Flexbox、Grid、Absolute、%/px/auto）
- 描画（角丸矩形、ボーダー、影、グラデーション、9-slice）
- インタラクション（ヒットテスト、イベント伝搬、フォーカス、ゲームパッドナビ、D&D）
- アニメーション（プロパティアニメ、Easing、transition、ページ遷移）
- 全ウィジェット実装（Widget一覧参照）
- C++バインディング（イベントバインド、動的操作、データバインディング）

Phase 7: ファイル・ネットワーク・動画

完了条件: 暗号化アーカイブからのアセット読込、HTTP通信、動画再生が動作

- ファイルシステム抽象化（物理/アーカイブ透過アクセス、マウントポイント）
- カスタムアーカイブ（パッキングツール、AES-256暗号化、LZ4/zstd圧縮）
- 非同期アセットローダー + ホットリロード
- TCP/UDP/HTTP/WebSocket
- Media Foundation動画デコード → テクスチャ出力

Phase 8: 数学・物理・衝突判定

完了条件: DXLibの全数学・衝突判定関数 + 空間分割 + 簡易物理が動作

- 数学ライブラリ（DirectXMathラッパー: Vector, Matrix, Quaternion, Color, MathUtil, Random）
- 衝突判定（2D: AABB, Circle, Polygon, Line / 3D: AABB, Sphere, Ray, Frustum, OBB + Sweep）
- 空間分割（Quadtree, Octree, BVH）
- 簡易2D物理（リジッドボディ、コリジョンレスポンス、トリガー）
- Jolt Physics統合（3D、オプション）

Phase 9: DXLib互換レイヤー + シェーダーホットリロード

完了条件: `#include "DxLibCompat.h"` で既存コードが動作する目処が立つ

- DxLibCompat.h（全パブリック関数ラッパー、型変換、定数マッピング）
- シェーダーホットリロード（ファイル監視→DXC再コンパイル→PSO再生成+エラーオーバーレイ）
- シェーダーバリアント管理（#define分岐、キャッシュ）
- 移行ガイドドキュメント

Phase 10: 最適化・品質・ドキュメント

完了条件: ドキュメント完備、全サンプル動作、プロファイラ動作

- メモリアロケータ、リソースバリア最適化、マルチスレッドCmdList、GPUプロファイラ
- テクスチャストリーミング、描画コールバッチング最適化
- D3D12 Debug Layer統合、GPU回帰テスト、Google Test、メモリリーク検出
- APIリファレンス（Doxygen）
- チュートリアル（Getting Started → 2D → 3D → GUI）
- **サンプルプロジェクト（ゴールG5）:**
  - 2Dシューティング
  - 2Dプラットフォーマー
  - 3Dウォークスルー
  - GUIメニューデモ
  - ポストエフェクトショーケース
- DXLib移行ガイド

■ 第7章：技術的リスクと対策

リスク	対策
D3D12の複雑さ（バリア、ヒープ）	Phase 0で基盤を固める。D3D12MA活用
シェーダーコンパイル時間	DXC + キャッシュ + ホットリロードを早期実装
glTF/FBXのエッジケース	cgltfを基本、Assimpはフォールバック
GUIレイアウト複雑化	Flexboxのみ先行、Gridは後回し
パフォーマンスボトルネック	GPUプロファイラをPhase 0から組み込み
スコープクリープ	フェーズ毎に「最小限動くもの」を作る

■ 第8章：作業上の注意事項（CLIで作業するAIへ）

8.1 方向性を見失った時

1. **第1章に戻れ。** 全ての判断基準はそこにある。
2. 「これはDXLibの上位互換に必要か？」と自問しろ。答えがYesなら進め、Noならスコープ外。
3. 5つのゴール（G1〜G5）のどれに貢献するか明確にしろ。どれにも貢献しないならやるな。

8.2 実装判断の優先順位

1. **正しく動く** > パフォーマンス > コードの美しさ
2. 各フェーズの完了条件を満たすことが最優先
3. 過度な抽象化をしない。DXLibの「使いやすさ」が設計理念の筆頭にあることを忘れるな
4. 「後でやる」を恐れるな。フェーズ分けはそのためにある

8.3 やってはいけないこと

- フェーズを飛ばすこと
- ディレクトリ構成を勝手に変えること
- 命名規則を逸脱すること
- 完了条件を満たさずに次のフェーズに進むこと
- DXLib互換を無視してオレオレAPIだけで進めること
- ポストエフェクト・レイヤー・GUIのいずれかを「後回しにしすぎる」こと（これらが存在意義）

8.4 このドキュメントの更新

フェーズ完了時に、以下をこのドキュメントに追記すること：

- 完了日
- 実際にかかった期間
- フェーズ中に発生した変更点や学び
- 次フェーズへの申し送り事項

> **最後に繰り返す。このプロジェクトの核心は「DXライブラリは見た目がショボいし機能が足りない。DirectX 12で完全上位互換を作る」である。この一文を常に頭に置いて作業せよ。**