04. UIとOpenGLの同期メカニズム

GUIアプリケーションの中にレンダリングループを組み込む際、最も重要なのは **「イベントループをブロックしないこと」** と **「コンテキストの管理」** です。

シグナル・スロット駆動の描画更新

ゲームエンジンのような「毎フレーム更新（Game Loop）」ではなく、省電力かつ効率的な **「デマンド駆動（Demand-driven）」** 方式を採用しています。

1. **State Change**: ユーザーがUIスライダーを動かすと、モデルの値が更新されます。
2. **Request Update**: `PreviewWidget` に対して `update()` が呼び出されます。これは即座に描画を行う命令ではなく、「次のV-Syncまたはアイドル時に描画が必要である」というフラグを立てる操作です。
3. **Paint Event**: Qtのイベントループが `paintGL()` メソッドを呼び出します。ここで初めて重い描画処理が走ります。

これにより、ユーザーが1秒間に100回スライダーを動かしたとしても、描画回数はモニタのリフレッシュレート（60fps等）に制限され、UIのフリーズを防ぎます。

コンテキスト共有とリソース管理

`QOpenGLWidget` は内部的に独自のFBOを持っています。 `initializeGL` で初期化されたOpenGLリソース（VBO, Shader Program, TextureID）は、そのウィジェットのコンテキストに関連付けられます。

本ツールでは、`Engine` クラスがこれらのリソースを一元管理していますが、 **「Engineは特定のWidgetに依存しない」** ように設計されています。 `Engine.initialize()` や `Engine.render()` は、呼び出し元の `PreviewWidget` がコンテキストをアクティブにしている（`makeCurrent()` されている）ことを前提に動作します。

これにより、将来的に「プレビュー用ウィジェット」と「エクスポート用（オフスクリーン）コンテキスト」を分ける必要が出てきた場合でも、同じ `Engine` インスタンス（または同じロジック）を再利用できる設計になっています。

比較表示 (Geometry Switching)

プレビューには「球体」と「比較（球体 x2）」のモードがあります。 これを実現するために、シーン全体の構造を変えるのではなく、 **「描画する頂点バッファの中身を入れ替える」** という軽量なアプローチを取っています。

- **Standard**: 球体メッシュ（中心座標 0,0）
- **Comparison**:
  - 左球体（x offset -0.5, UV調整あり）
  - 右球体（x offset +0.5, UV調整あり）

`GeometryEngine` クラスがこれらの頂点配列を生成し、VBOへ転送します。シェーダー側は「単に渡された頂点を描画するだけ」なので、モードによる分岐を持つ必要がありません。