01. 技術スタック詳解

Matcap Maker v3 は、パフォーマンスと開発効率のバランスを考慮し、以下のモダンな技術スタックで構成されています。

Core Technologies

カテゴリ	技術	バージョン	選定理由と役割
Language	**Python**	3.10+	エコシステムの広さと開発速度。特に `type hinting` を活用した堅牢なコードベース構築が可能。
GUI Framework	**PySide6 (Qt6)**	6.x	`QOpenGLWidget` による強力なOpenGL統合。ネイティブアプリケーションとしての操作性とクロスプラットフォーム対応。
Graphics API	**PyOpenGL**	3.1.x	生のOpenGLコマンド（`glUseProgram`, `glBindFramebuffer` 等）を制御し、独自のレンダリングパイプラインを構築するため。
Math / Array	**NumPy**	1.2x	エクスポート時の画像処理（パディング）における高速な配列演算。将来的な画像解析への拡張性。
Image I/O	**Pillow**	9.x	画像の読み込み・保存、フォーマット変換のデファクトスタンダード。
Styling	**qt-material**	-	モダンなMaterial Designテーマの適用。CSSライクなスタイルシート管理。

なぜ Python + OpenGL なのか？

通常、リアルタイムグラフィックスアプリケーションは C++ で書かれることが多いですが、本ツールでは以下の理由から Python を採用しました：

1. **プロトタイピングからプロダクトへのシームレスな移行**:
   - 数学的なアルゴリズム（パディング処理やノイズ生成ロジック）を試行錯誤する際、Pythonのインタプリタ言語としての特性が有利に働きました。
2. **GPU依存のパイプライン**:
   - 重たい処理（合成、フィル、ライティング計算）はすべて **Fragment Shader (GLSL)** 上で行われます。
   - CPU側のPythonコードは「API呼び出しのオーケストレーション」を担当するだけであり、Pythonの実行速度（GILなど）はボトルネックになりにくいアーキテクチャです。
3. **拡張性**:
   - 将来的にユーザーがPythonスクリプトでカスタムレイヤーを追加できる「プラグインシステム」を構想しており、ホスト言語がPythonであることは大きな利点です。

GUI統合: QOpenGLWidget

Qtフレームワークの `QOpenGLWidget` は、OSネイティブのウィンドウシステムとOpenGLコンテキストの間にある複雑な処理（コンテキスト作成、スワップチェーン管理、高DPI対応）を隠蔽してくれます。 これにより、開発者は `initializeGL()`, `paintGL()`, `resizeGL()` という3つのメソッドを実装するだけで、Qtのウィジェット階層の中に3Dビューポートを埋め込むことができます。