GIMP-Turbo開発計画

2025年3月24日

21:14

概要

URL：[chatGPT 要件定義キャンバス](https://chatgpt.com/canvas/shared/67e16e86b04881919883f4600ac74977)

# 目的

画像品質向上に特化したソフトウェアの開発

# 経緯

画像生成AIを利用した画像・イラスト生成を行っている中で、画像の加工を行うためには一旦画像の高画質化が必要となっていた。ただし、高画質化には手間のかかるソフトウェアや、料金のかかる手順ばかりであるため、簡単で費用のかからない自分用のソフトウェア開発を実施することを決断するに至った。

# プロジェクト方針

1. 「まずは完成させろ」が最優先課題
2. 再現性の高い手順で開発
3. 明確かつ詳細なドキュメントが存在する手順を選択

要件定義

# 開発要件

1. 画像処理機能を持つコンソールアプリの開発
2. ユーザーインプットおよびアウトプット専用のGUIアプリの開発
3. コンソールアプリとGUIアプリの統合

# 技術要件

1. 再現性の高いライブラリ（作業手順が明確）
2. 実装が容易なライブラリ（作業ミスが起こりづらい）
3. 処理速度が期待できるライブラリ（性能要件１）
4. 拡張機能が期待できるライブラリ（性能要件２）

# 機能要件

* コンソールアプリ
  + 画像解析
  + 画像品質向上
* GUI機能
  + 画像取り込み
  + 画像保存
  + 画像表示
  + プレビュー表示
  + パラメータ調整

詳細設計

## コンソール設計

* ノイズ解析

機能：取り込んだ画像のノイズを解析し、ノイズ除去に適切なパラメータを取得

* ノイズ除去

機能：ノイズ解析で取得したパラメータに従ってノイズを除去

* アップスケーリング

機能：画質の低下を可能な限り抑止しながら解像度を向上

* 画像解析

機能：画質を解析し、画質向上に必要なパラメータを取得

* 画質向上

機能：画像解析で取得したパラメータに従ってノイズを除去

* 解像度復元

機能：アップスケーリング前の画像サイズに復元(縮小)

* パラメータ引き渡し

機能：ノイズ解析や画像解析で取得したパラメータをGUIの初期設定にするため引き渡す

* プレビュー画像引き渡し

機能：パラメータに従って画像処理を行った画像をプレビュー画像としてGUIに引き渡す

# 

# GUI設計

## 画像表示

* 取り込み画像表示

機能：ユーザー操作で取り込んだ画像を表示

表示場所：画面左上に300×300で配置（画像直下中央に「取り込み画像」のラベル表示）

* プレビュー画像

機能：パラメータ通りに処理されたプレビュー画像を表示

表示場所：画面右上に300×300で配置（画像直下中央に「プレビュー画像」のラベル表示）

## インジケータ

機能：プレビュー画像の生成進捗を示す機能

表示場所：取り込み画像とプレビュー画像の下に配置

## パラメータ調整

機能：画像処理の各パラメータを調整するスライダー

表示場所：インジケータ下に配置

スライダー種類

* ノイズ除去
* アップスケーリング
* 画像品質向上（細分化するかどうか要検討）

## ボタン

* 画像取り込み

機能：ダイアログボックスで処理対象画像を取り込む

表示場所：各スライダー下の左側（取り込み画像中央）に配置

* 画像保存

機能：プレビュー表示中の画像を名前を付けて保存

表示場所：各スライダー下の右側（プレビュー画像中央）に配置

* キャンセル

機能：画面を閉じる

表示場所：画面右下端

# コンポーネント設計

## 主要ライブラリ

画像解析ライブラリ:

画像処理ライブラリ:

GUIライブラリ:

## 依存関係ライブラリ

# 画像処理設計

* ノイズ解析方式
* ノイズ除去方式
* アップスケーリング方式
* 画質解析方式
* 画質向上方式

開発環境

# 基盤

WSL + ubuntu + docker （安定した環境での開発を優先するため）

dockerイメージ：ubuntu

# バージョン管理

Git + GitHUB

# フォルダ構成（Windows）

bash

コピーする編集する

D:/dev/GIMP-Turbo/

│── docker/ # Docker 環境

│── product/ # GitHub リポジトリ (ルート)

│ ├── Step1/ # 各開発ステップのプロジェクト

│ ├── Step2/

│ ├── Step3/

│ ├── Step4/

│ ├── Step5/

│ ├── docs/ # ドキュメント

│ ├── README.md

│ ├── LICENSE

│ └── .gitignore

# Docker と Windows の ディレクトリ 一覧

|  |  |  |
| --- | --- | --- |
| **目的** | **Windows 側** | **Docker 側** |
| Windows側ホームフォルダ | D:/dev/GIMP-Turbo/ |  |
| ソースコードの共有(GitHUBリポジトリルート) | D:/dev/GIMP-Turbo/product/ | /app/product/ |
| プロジェクトStep1共有フォルダ | D:/dev/GIMP-Turbo/product/Step1 | /app/product/Step1 |
| プロジェクトStep2共有フォルダ | D:/dev/GIMP-Turbo/product/Step2 | /app/product/Step2 |
| プロジェクトStep3共有フォルダ | D:/dev/GIMP-Turbo/product/Step3 | /app/product/Step3 |
| プロジェクトStep4共有フォルダ | D:/dev/GIMP-Turbo/product/Step4 | /app/product/Step4 |
| プロジェクトStep5共有フォルダ | D:/dev/GIMP-Turbo/product/Step5 | /app/product/Step5 |
| プロジェクトStep6共有フォルダ | D:/dev/GIMP-Turbo/product/Step6 | /app/product/Step6 |
| プロジェクトStep7共有フォルダ | D:/dev/GIMP-Turbo/product/Step7 | /app/product/Step7 |
| ドキュメント格納ディレクトリ | D:/dev/GIMP-Turbo/product/docs | /app/product/docs |
|  | D:/dev/GIMP-Turbo/product/README.md | /app/product/README.md |
|  | D:/dev/GIMP-Turbo/product/.gitignore | /app/product/.gitignore |
| Windows側Dockerフォルダ | D:/dev/GIMP-Turbo/docker/ |  |
| ビルド成果物の共有 | D:/dev/GIMP-Turbo/docker/build/ | /app/build/ |
| テスト用画像の共有 | D:/dev/GIMP-Turbo/docker/test\_data/ | /app/test\_data/ |
| 設定ファイルの共有 | D:/dev/GIMP-Turbo/docker/config/ | /app/config/ |
| SSH 鍵の共有（GitHub 連携用） | /mnt/wsl/Ubuntu/home/your-user/.ssh/ | /root/.ssh/ |

URL：[chatGPT マイルストーン キャンバス](https://chatgpt.com/canvas/shared/67e16e86b04881919883f4600ac74977)

# マイルストーン（Step 0～5）

## Step 0: 環境構築

* 開発環境の構築
* Step1～Step4で必要なコンポーネントを導入。
* 画像処理やGUIアプリの依存関係ライブラリもここでインストール。

## Step 1: コンソールアプリの基本動作構築

* **画像の入力と出力**のみを実装。
* 入力パスや出力パスは**ハードコーディング**。
* **エラー処理は最小限**（例: ログ出力のみ）で後回し。

## Step 2: 右クリックの「送る」メニューとの簡易連携

* PowerShellスクリプトを使用し、SJISからUTF-8に変換してアプリにパスを渡す。

***スクリプト例:***

powershell

コピーする編集する

param([string]$filePath)

$utf8Path = [System.Text.Encoding]::UTF8.GetString(

[System.Text.Encoding]::GetEncoding("Shift\_JIS").GetBytes($filePath)

)

wsl ./my\_console\_app "$utf8Path"

pause

## Step 3: 簡易画像処理

* 画像処理ライブラリを使用し、固定パラメータで画像処理を実施

## Step 4: 自動解析と画像処理の初期実装

* **画像処理ライブラリ**を使用し、画像の解析を実施。
* 解析結果に基づいて画像の最適化を実施。

## Step 5: GUI 開発

* 画像処理コンソールとの統合が前提のGUIを開発

## Step 6: コンソールとGUI の連携

* **コンソールとGUIを統合し、スタティックリンクとして構成**:

## Step 7: ソフトウェアのブラッシュアップ

* **動作の安定性向上**
* 不安定な箇所の改善。
* **デザイン・UIの改善**
* 使いやすさを考慮した改良。
* **処理の最適化**
* 高速化やより正確なアルゴリズムの導入。