GA-CLIP Engine (遺伝アルゴリズム × CLIP評価)

talklog.txtの会話をベースに作成した、遺伝アルゴリズムで抽象画像を進化させ、CLIPで自動評価するシステムです。 FastAPIへの拡張を想定したモジュール構成になっています。

特徴

- **遺伝子表現**: 半透明の図形（長方形/楕円）のリストとして画像を表現
- **レンダリング**: Pillowでアルファ合成により画像生成
- **CLIP評価**: transformersの`openai/clip-vit-base-patch32`でテキストプロンプトとの類似度を計算
- **遺伝アルゴリズム**: トーナメント選択、一様/一点交叉、パラメータ・構造突然変異
- **拡張性**: Engineクラスで統合、FastAPI化が容易

インストール

```
# 仮想環境作成
python -m venv venv
source venv/bin/activate  # Windowsは venv\Scripts\activate

# 依存関係インストール
pip install -r requirements.txt
```

使い方

CLIPなしで動作確認（ダミースコア）

```
python ga_clip_engine.py --dummy --out runs/test --generations 5
```

CLIPで評価して進化

```
python ga_clip_engine.py \
  --out runs/demo \
  --prompts "aesthetic curves" "smooth gradients" \
  --pop-size 24 \
  --genes-min 80 \
  --genes-max 200 \
  --generations 20 \
  --elitism 2 \
  --tournament 4 \
  --mutation-rate 0.15
```

パラメータ説明

- `--out`: 出力ディレクトリ
- `--prompts`: 評価用テキストプロンプト（複数指定可）
- `--model`: CLIPモデル名（デフォルト: openai/clip-vit-base-patch32）
- `--pop-size`: 個体数
- `--genes-min/max`: 図形数の最小/最大
- `--generations`: 世代数
- `--elitism`: エリート保存数
- `--tournament`: トーナメント選択のサイズ
- `--mutation-rate`: 突然変異率
- `--crossover`: 交叉方式（uniform/one_point）

出力

実行すると指定ディレクトリに以下が保存されます：

```
runs/demo/
├── gen_000/           # 各世代のトップ画像とメトリクス
│   ├── rank_01_score_0.xxxx.png
│   ├── rank_01.json   # 遺伝子情報
│   └── metrics.json   # 世代統計
├── gen_001/
├── ...
└── best/              # 最高スコアの個体
    ├── best_score_0.xxxx.png
    └── best.json
```

FastAPI拡張例

```python
from fastapi import FastAPI
from ga_clip_engine import Engine, GAConfig, RenderConfig

app = FastAPI()
engine = Engine(GAConfig(), RenderConfig(), prompts=["aesthetic art"])

@app.post("/prompts")
def set_prompts(prompts: list[str]):
    engine.set_prompts(prompts)
    return {"status": "ok", "prompts": prompts}

@app.post("/step")
def evolve_one_generation():
    scored = engine.step()
    return {
        "generation": engine.generation,
        "best_score": float(max(s for _, s in scored)),
        "mean_score": float(sum(s for _, s in scored) / len(scored))
    }
```

```
@app.get("/population/{index}")
def get_individual_image(index: int):
    if 0 <= index < len(engine.population):
        img = engine.renderer.render(engine.population[index])
        # Return as base64 or save to temp file
        return {"index": index, "genes":
len(engine.population[index].genes)}
    return {"error": "Index out of range"}
```

プロンプト例

- `"aesthetic curves"` - 美的な曲線
- `"vibrant colors"` - 鮮やかな色彩
- `"minimalist composition"` - ミニマリストな構成
- `"abstract art"` - 抽象芸術
- `"geometric patterns"` - 幾何学パターン
- `"soft gradients"` - 柔らかなグラデーション

注意事項

- GPUがあれば`torch`がCUDA版になっているか確認してください
- 初回実行時はCLIPモデルのダウンロードに時間がかかります