

AIイラスト用 シマビルダー

Ultimate Prompt Generation Tool

クイックスタート（最短で使う）

まずはJSファイルの読み込みが完了し、画面上の「Finalizing」表示が消えるまで待つ。

基本使用フロー：

1. カテゴリーからワードを選択
2. 「生成」を押す
3. 必要に応じて強調／弱化で微調整
4. 「コピー」で生成結果をAIに貼り付け

ボタン機能：

- ・生成：チェックしたワードを生成
- ・コピー：生成ワードをコピー
- ・リセット：すべてのチェックを外し初期化
- ・確認：チェック項目を検索（環境により動作差あり）
- ・履歴：過去に生成したワードを呼び出す

強調・弱化の使い方：

（）強調 {} 強調 [] 弱化

カーソルを合わせたワードに適用する。

強調度選択：

0.7 / 0.8 / 0.9 / なし / 1.1 / 1.2 / 1.3 / 1.4

例：(beautiful girl:1.3)

※強度選択は（）のみ対応。



Chapter 1 構造化プロンプトの理論

1-1 プロンプトは文章ではなく「構造体」である
多くの人はプロンプトを文章として書く。

beautiful girl, masterpiece, ultra high resolution, cinematic lighting...

一見、問題はない。

しかし実際には、これは文章ではなく指示の集合である。

AIは文脈を読む存在ではあるが、同時に重み・配置・関係性にも強く影響を受ける。

つまりプロンプトとは、単語の意味 単語の強度 (weight) 単語の位置 他要素との競合
これらが絡み合った「構造体」である。

構造を理解せずに単語を増やすと、やがて出力は崩壊する。

それはAIが悪いのではない。

設計が曖昧だからである。

1-2 レイヤー構造という考え方 プロンプトを安定させる最初の鍵は、
レイヤー分解である。

本書では、プロンプトを以下の層で捉える。

- ① 基盤層 (Quality Layer) 画質、レンダリング、物理表現など 全体の描写品質を決める層。
- ② 主体層 (Subject Layer) 人物、キャラクター、主役となる存在。
- ③ 構造層 (Structure Layer) 衣装、ポーズ、装備など 視覚的な形状を決める要素。
- ④ 演出層 (Effect Layer) 光、粒子、質感、エフェクトなど 雰囲気を増幅する要素。
- ⑤ 環境層 (World Layer) 背景、場所、時間帯、世界観。

これを意識せずに混在させると、要素同士が競合する。

例えば、強い光演出 強い質感指定 強い世界観指定 これらを同時に最大強度で入れると、
主役が埋もれる。

それは“盛りすぎ”ではなく、レイヤー干渉である。

1-3 強度 (weight) の力学 (1.2) や (1.4) のような数値指定は、
単なる「強くする記号」ではない。

それは優先度の宣言である。

強度が上がるほど、他要素を押しのける 競合時に優位に立つ 画面の支配力を増す
しかし、強度を全体にかければどうなるか。

答えは単純である。

優先順位が消える。

全員が叫べば、誰の声も通らない。

強度は「全体を強くする装置」ではなく、「どれを主役にするか決める装置」である。

1-4 出力順序の支配力 AIは左から右へ処理される傾向がある。

完全に線形ではないが、前方の情報は基礎として扱われやすい。

そのため、基盤 (Quality) は前方に 主体は早い段階で確定 演出は後段で補強
この順序設計が安定を生む。

順序を無視すると、エフェクトが主体を上書き 世界観が人物を飲み込む 質感が崩壊する
といった現象が起きる。

偶然成功することもある。

しかしそれは再現性を持たない。

1-5 過積載崩壊理論 プロンプトは長いほど良いわけではない。

むしろ一定を超えると、要素同士が干渉 意図が分散 モデルが平均化 という状態になる。

これを本書では 過積載崩壊と呼ぶ。

兆候はこうだ。

画質は高いが印象が弱い どこを見せたいのか分からない 毎回微妙に違う結果になる

対策は単純である。

レイヤーごとに整理し、主役を一つに決める。

1-6 再現性を生む組み方 再現性とは、「同じ構造で組めば、近い結果が出る」という状態である。

そのために必要なのは：

レイヤー分解 強度の意図的配分 出力順序の固定 不要要素の削減 偶然の成功を繰り返すのではなく、構造で成功を再現する。

これがプロンプトエンジニアリングである。

Chapter 2 プロンプトビルダーの全体構造

2-1 カテゴリー設計思想 このビルダーは、単語を並べるための箱ではない。

レイヤー構造を強制的に整理するための装置である。

カテゴリー分けは見た目の整理ではない。

思考の整理である。

主な設計思想は次の通り。

- Quality（基盤層）画質、レンダリング、物理演算、光学処理など出力全体の描写基盤を決定する層。

ここが不安定だと、上に何を積んでも崩れる。

- Character / Subject（主体層）誰が主役なのかを明確にする層。

AIは曖昧さを許容するが、主役の曖昧さは画面の弱体化に直結する。

- Outfit / Structure（構造層）形状・装備・衣装・ポーズなど視覚的な骨格を決める層。

主体を具体化する役割を持つ。

- Effects / Texture（演出層）光、粒子、質感、特殊効果。

ここは“増幅装置”である。

主役が確定していない状態で増幅すると、何もかもが強くなり、結局弱くなる。

- World / Environment（環境層）背景、場所、時間帯、空気感。

世界観は強力である。

だからこそ最後に設計する。

カテゴリー分けは、AIのためではなく、人間のためにある。

人間の思考が整理されれば、出力も整理される。

2-2 プロンプト生成ロジック

ビルダーは単にチェックボックスを結合しているわけではない。

設計思想はこうだ。

カテゴリー単位で要素を選択 出力順序に従って整列 必要に応じて break を挿入
最終プロンプトを構造順で出力 順序が固定されていることで、再現性が生まれる。

毎回違う順序で出力される仕組みであれば、安定はしない。

2-3 break / multiple の設計思想

- break の役割 break は単なる改行ではない。

AIに対する「区切り」の宣言である。

基盤と主体を分ける 主体と演出を分ける 構造の塊を整理する これにより、干渉を緩和する。

- multiple の役割 multiple は要素の重複指定を許容する仕組み。

しかし、乱用すれば強度競合を引き起こす。

multiple は 同種要素を積むための装置 強度を分散させる手段 として使うべきであり、
“全部盛りボタン”ではない。

2-4 なぜ棚分けするのか 棚分けには3つの目的がある。

思考の分解 優先順位の明確化 拡張性の確保 プロンプトは増やすほど管理が難しくなる。

棚分けをしない場合、何がどこにあるか分からない 重複が発生する 競合が増える
構造化とは、管理可能にすることでもある。

2-5 設計と自由度のバランス ビルダーは自由を奪うための装置ではない。

むしろ、自由を制御可能にするための装置である。

完全自由は混乱を生む。

完全固定は創造性を殺す。

このビルダーは、レイヤーを固定し 内部で自由を許容する というバランスを取っている。

構造を守りながら、内容を遊ぶ。

それが最も安定する。

Chapter 3 基本操作ガイド

3-1 起動と初期画面の理解 ビルダーを起動すると、複数のカテゴリーが並んだ画面が表示される。

ここで最初にやるべきことは、何かを選ぶことではない。

全体を見ることである。

どのカテゴリーが存在し、どの順序で並んでいるか。

この並び順は偶然ではない。

前章で述べたレイヤー構造が、そのままUIに反映されている。

つまり、画面そのものが構造図である。

焦ってチェックを入れ始める前に、一度全体を俯瞰する。

（この一呼吸で、出力の安定度は変わる。）

3-2 カテゴリー選択の基本原則 基本ルールは単純である。

下からではなく、上から組む。

推奨手順 Quality（基盤）を決める 主体（Character / Subject）を確定する 構造（Outfit / Pose）を追加 演出（Effects / Texture）を補強 環境（World）で締める

いきなりエフェクトを盛ると、後から主体が埋もれる。

順番には理由がある。

料理と同じだ。

最初にスパイスを全投入しない。

3-3 プロンプトの生成とコピー 必要な要素を選択したら、生成ボタンを押す。

出力されるテキストは、レイヤー順 適切な区切り 指定された強度 で整形されている。

ここで重要なのは、生成後にすぐ使わないこと。

一度、全体を読み返す。

主役は明確か？ 強度は偏っていないか？ 同系統の指定が重複していないか？

ビルダーは補助装置である。

最終確認は人間が行う。

3-4 リセットと再構築 出力が崩れた場合、やみくもに単語を足さない。

一度リセットする。

再構築の基本は、Qualityのみ 主体のみ追加 徐々に積む 原因特定は、積み上げ方式が最も早い。

全部入れたまま修正しようとすると、問題が見えない。

3-5 初心者がやりがちな誤り

■ 誤り1：

強度の全体最大化 (1.3) (1.4) を連打すると安心するが、優先順位が消える。

強度は「選ぶ」ためにある。

■ 誤り2：

エフェクト先行型 光や粒子は魅力的だが、主体が曖昧な状態で入れると画面が散る。

■ 誤り3：

成功プロンプトの無改造流用 他人の成功例をそのまま使うと、再現性は低い。

なぜ成功したのか理解していないからである。

■ 誤り4：

足し算思考 うまくいかない → 単語を足す これは最も多い。

実際には、うまくいかない → 構造を整理する が正解である。

3-6 操作は簡単、設計は奥深い このビルダーの操作自体は難しくない。

チェックして、生成するだけである。

しかし、何を選び、何を選ばないか。

そこに設計思想がある。

ツールは単純。

思考は深い。

その順番を守れば、出力は安定する。

Chapter 4 クオリティ設計論

4-1 「masterpiece」の本質 “ masterpiece ” は魔法の言葉ではない。

それは 出力傾向を高品質側へバイアスする宣言である。

しかし、多くの人はこれを万能強化ワードとして扱う。

結果どうなるか。

masterpiece best quality ultra high resolution high detail 8k UHD hyper detailed

これらを大量に積む。

すると何が起こるか。

品質は上がる可能性がある。

だが、主役がぼやけることもある。

理由は単純だ。

Quality層が強くなりすぎると、主体層よりも基盤層が支配的になる。

つまり、「何を描くか」より「どれだけ綺麗に描くか」が優先される。

品質は重要だが、主役ではない。

4-2 レンダリングワードの階層構造

レンダリング関連ワードは、大きく分けて三層ある。

■ 描写密度層 detailed intricate details highly detailed ultra detailed 描写量を増やす。

■ 物理演算層 physically based rendering global illumination subsurface scattering ambient occlusion 光や材質の計算精度を上げる。

■ エンジン指定層 unreal engine octane render lumen reflections nanite geometry 特定のレンダリング傾向を誘導する。

これらを無差別に混ぜると、方向性が曖昧になる。

例：

anime style photorealistic octane render cel-shaded これは物理的にも思想的にも競合する。

混ぜることは可能だが、意図がなければ破綻する。

4-3 PBR / UE / Octane の使い分け

■ PBR (Physically Based Rendering) 物理整合性を重視する。

質感が安定 金属や布の表現が自然 過剰な誇張が抑えられる 安定化装置として優秀。

■ Unreal Engine 系 ダイナミックな光演出。

コントラストが強い 立体感が出やすい ゲーム的臨場感 主役を強調したいときに有効。

■ Octane 系 反射と艶の誇張。

ハイライトが強い 光沢表現が派手 印象は強いが干渉も強い 演出層として扱うのが安全。

重要なのは、全部を最大で入れないこと。

目的に応じて選ぶ。

4-4 光の演出理論 光は画面を支配する。

しかし光には階層がある。

■ 基本光 soft lighting natural lighting 安定の土台。

■ 強調光 cinematic lighting dramatic lighting rim light 主役を浮かせる。

■ 増幅光 volumetric lighting god rays glow effects 雰囲気を作る。

増幅光を基盤より強くすると、画面は“光の塊”になる。

光は足すものではなく、設計するものである。

4-5 質感と物理的整合性 shiny skin, glossy hair, metallic reflections など。

質感は強力だ。

しかし質感は素材指定であり、主役ではない。

肌を輝かせすぎれば人形になる。

金属を強調しすぎれば現実感が消える。

質感は主役を支える。

質感が主役になると、画面は広告写真になる。

それが目的なら問題ない。

違うなら調整が必要だ。

4-6 高密度化と安定化の両立 高密度化は魅力的である。

だが安定化と両立しなければ、再現性は失われる。

両立の基本原則：

Quality層は固定テンプレを持つ 主役は常に強度上位 演出は最後に足す 不要な競合ワードを削除
品質を上げるのではない。

設計を整える。

これがクオリティ設計論の核心である。

Chapter 5 実戦テンプレート集 注意：

以下のテンプレートは“完成品”ではない。
構造の例である。目的に応じて調整すること。

5-1 安定量産型テンプレート 目的：

SNS投稿、連続シリーズ、安定した高品質出力。

- 構造方針 Qualityは固定 主体は明確 演出は控えめ 環境は軽く指定
- テンプレ例（構造順） コードをコピーする (masterpiece:1.2), (best quality:1.2), (high resolution), (physically based rendering), (global illumination), (soft natural lighting), break, 1girl, detailed face, clear eyes, balanced composition, break, simple elegant outfit, natural pose, break, subtle rim light, gentle depth of field, break, clean background, soft atmosphere
- 解説 Qualityはやや控えめ（1.2前後） 主役が常に画面中心 エフェクトは“補助”
これは崩れにくい。

5-2 一枚特化ハイエンド型 目的：

コンテスト、販売素材、ポスター用途。

- 構造方針 Quality強め 主体を強度上位に 光演出を戦略的に使用 世界観は明確に
- テンプレ例 コードをコピーする (masterpiece:1.3), (ultra high resolution:1.3), (intricate details:1.3), (physically based rendering), (lumen reflections), (volumetric lighting), break, (1girl:1.3), detailed eyes, expressive face, dynamic pose, break, ornate fantasy armor, flowing fabric, structured silhouette, break, cinematic lighting, rim light, glowing particles, break, epic battlefield at sunset, dramatic sky
- 解説 主体に(1.3)を付与し、Qualityと競合させない 演出層は後段で加える 世界観は明確に指定
ここでは“主役の勝利”を優先する。

5-3 世界観特化型 目的：

シリーズ物、物語性の強化。

- 構造方針 環境層をやや強め 主役との競合に注意 光と世界観を連動させる
- テンプレ例 コードをコピーする (masterpiece:1.2), (best quality), (physically based rendering), break, 1boy, calm expression, detailed costume, break, traditional scholar robe, layered fabric, break, soft volumetric light, floating dust particles, break, ancient library illuminated by candlelight, towering bookshelves
- 解説 世界観が主役を支える構図 光源と背景を一致させる 演出は“空気”を作る
世界観が主役を飲み込まないようにする。

5-4 演出増幅型（質感 × 環境） 目的：

印象を強く残すビジュアル。

- 構造方針 質感を明確に選択 光と整合させる 過積載を避ける
- テンプレ例 コードをコピーする (masterpiece:1.2), (high detail), (physically based rendering), break, 1girl, confident expression, break, sleek black latex suit, reflective surface, break, strong rim light, specular highlights, controlled bloom, break, dark futuristic city at night, neon reflections
- 解説 質感 (latex) を主軸に 光をそれに合わせる 世界観も連動
整合性があれば、派手でも崩れない。

5-5 ブランドシリーズ設計型 目的：

キャラクター資産化、統一感の維持。

■ 構造方針 Quality固定テンプレを持つ 主体要素を固定 世界観のみ変化させる

■ ベーステンプレ コードをコピーする (masterpiece:1.2), (best quality:1.2), (physically based rendering), (soft cinematic lighting), break, original character, signature hairstyle, signature color palette, break, core outfit elements, recognizable silhouette, break, subtle depth of field, consistent highlight style, break, [environment variation here]

■ 解説 主体の識別性を固定 光と質感も固定 背景だけ変化させる
これでシリーズとしての統一感が出る。

5-6 テンプレートは型であって完成形ではない 重要なのは、
テンプレートを“コピペ”することではない。

レイヤー順で思考すること。

テンプレートは補助輪である。

外せるようになることが目標である。

**Chapter 6 修正版 開発者向けではなく、
“高度な使いこなし” 向けに変える。**

Chapter 6 高度な活用戦略

6-1 テンプレート固定戦略 安定した出力を得る最も強力な方法は、Quality層を固定テンプレート化すること。

例：

品質 光 レンダリング基盤 ここを毎回変えない。

変えるのは、主体 世界観 演出 固定部分を持つことで、シリーズとしての統一感が生まれる。

6-2 出力崩壊時の復旧プロセス 崩れた時にやってはいけないこと：

単語を足す。

正しい手順：

Qualityのみで出力確認 主体を追加 演出を一つずつ戻す この“積み上げ再構築”が最短ルート。

6-3 優先順位設計の応用 高度な設計では、主体を1.3 Qualityを1.2 演出は無印といった強度の階層化を行う。

これにより、主役が常に勝つ。

強度は“全体強化”ではなく、優先順位の宣言である。

6-4 シリーズ設計の方法 ブランド化を狙う場合、主体のコア要素を固定 カラーパレットを固定 光の傾向を固定 背景のみ変更 この設計で資産化が進む。

ランダム性ではなく、一貫性が価値を生む。

6-5 ツール依存を超える思考 このビルダーは装置である。

だが重要なのは、レイヤー思考そのものだ。

AIモデルが変わっても、レイヤー分解 強度管理 順序設計 は通用する。

ツールに依存しない思考を身につけること。

それが最終的な目的である。

これなら、内部構造は開示しない 価値は保つ 上級者も満足する バランスが取れている。

Chapter 7 トラブルシューティング

出力が不安定なとき、原因は“AIの気まぐれ”ではない。

多くの場合、構造の問題である。

7-1 思った通りに出力されない

- 症状 主役が弱い 意図したポーズにならない 雰囲気が違う
- 原因の可能性 主体層が弱い 世界観や演出が強すぎる 強度が均一化している
- 対処法 主体に強度を付与（例：

1.3）演出層を一時的に削除 breakで層を整理 まずは“引き算”。

7-2 画面がごちゃごちゃする

- 症状 何を見せたいか分からない 情報量が多いが印象が弱い
- 原因 過積載崩壊。

光が多すぎる エフェクトが重複 世界観と主体が競合

- 対処 演出層を半分削る 主役の強度を再設定 背景を簡素化 整理すると、むしろ印象は強くなる。

7-3 品質は高いのに“刺さらない”

- 症状 綺麗だが印象に残らない SNSで反応が弱い
- 原因 Quality層が主役を支配している。

“高画質”が主役になっている。

- 対処 主体を1段階強くする 演出の方向性を一つに絞る テーマを明確にする 品質は土台であって、主役ではない。

7-4 出力が毎回バラつく

- 症状 同じ構成でも結果が安定しない
- 原因 強度設計が曖昧 要素が多すぎる 主体の定義が抽象的
- 対処 Qualityテンプレを固定 主体定義を具体化 不要要素を削除 再現性は“減らす”ことで生まれる。

7-5 強度を上げてても改善しない

- 症状 (1.4)にしても変化が弱い
- 原因 強度競合。

全員が1.4なら意味がない。

- 対処 主役だけ上げる 他を下げる 層を整理する 強度は“差”があって初めて機能する。

7-6 崩れたときの最短復旧手順 迷ったらこれ。

Qualityのみで出力 主体を追加 構造を追加 演出を最後に戻す これで原因はほぼ特定できる。

7-7 AIは敵ではない 出力が崩れたとき、AIを疑いたくなる。

だが多くの場合、構造が曖昧である。

AIは曖昧さを平均化する。

設計が明確なら、出力も明確になる。

Final Chapter 構造化思考の未来

F-1 ツールは変わる。構造は残る。

AIモデルは進化し続ける。

今日の最適解は、半年後には古くなるかもしれない。

しかし変わらないものがある。

レイヤー分解 強度管理 順序設計 優先順位の明確化 これらはモデルに依存しない。

構造を理解していれば、新しいAIにも適応できる。

ツールを覚えるのではない。

思考を覚える。

F-2 再現性は資産になる 偶然の成功は、消える。

再現可能な成功は、積み上がる。

構造化されたプロンプトは、シリーズ化できる ブランド化できる 他者に説明できる 修正できる
それは単なるテキストではない。

設計資産である。

F-3 差が生まれる場所 多くの人は、単語を増やす 強度を上げる 流行タグを追う ここで止まる。

しかし差が生まれるのは、何を削るか どこを固定するか 何を主役にするか という設計部分である。

構造化思考は、派手ではない。

だが確実に差を生む。

F-4 このビルダーの位置づけ このビルダーは、創造性を制限するものではない。

むしろ、創造性を安定させる装置である。

自由を奪うのではなく、自由を制御可能にする。

そのために、レイヤーを分け、順序を整え、強度を管理している。

使いこなすほど、設計思考が自然に身につく。

それが最終目的である。

F-5 最後に プロンプトは文章ではない。

設計である。

設計は、再現できる。

再現できるものは、積み上がる。

積み上がるものは、資産になる。

その第一歩として、この構造を使いこなしてほしい。