Image Generation / DALL·E

Design prompts for editing specific regions of existing images while preserving context — targeted modification without regenerating everything.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Photo Editing, Image Repair, Expansion
Updated: May 2026
Why This Prompt Exists
DALL·E's inpainting and outpainting let you edit specific parts of an image without regenerating everything. Most users don't use these features because they don't know how to prompt for them effectively.

You get:

- edits that don't match the original style (inconsistent lighting)
- inpainting that creates artifacts at region boundaries
- outpainting that repeats patterns unnaturally
- no strategy for complex multi-step edits
- wasted edits because the prompt doesn't specify context

But inpainting/outpainting can be systematic:

- region specification: what part of the image to edit
- context description: what surrounds the edited area
- style matching: instructions to preserve original aesthetic
- seamless blending: prompts that avoid hard edges
- iterative refinement: edit, review, edit again

Without strategy, edits look like patches.

This prompt designs effective inpainting and outpainting workflows.

The Prompt

```
Assume the role of a DALL·E editing specialist who designs
inpaint/outpaint prompts.

Your task is to create prompts for targeted image editing.

Generate:

1. EDIT TYPE CLASSIFICATION
   - Edit type: [INPAINTING (replace region) / OUTPAINTING (expand
canvas) / REMOVAL (delete object)]
   - Region to edit: [describe location, size, shape]
   - Original image context: [describe surrounding elements]

2. STYLE PRESERVATION REQUIREMENTS
   - Lighting must match: [describe existing lighting]
   - Color palette must match: [describe dominant colors]
   - Texture must match: [smooth/rough/patterned]
   - Perspective must match: [camera angle, depth]

3. INPAINTING PROMPT TEMPLATE

**For replacing an object:**
`[surrounding context description] with [new object] in place of [old
object], matching the [lighting/color/texture] of the surrounding
area, seamless blend, no visible edges`
```

For removing an object:
`[surrounding context description] without [object to remove], filled with [appropriate background], seamless, natural`

For outpainting (expanding canvas):
`Extend this image to the [direction]. Add [new content] that continues the [style/color/lighting] of the original image. Seamless transition, no visible seam.`

4. REGION-SPECIFIC PROMPTS BY TASK

Task	Prompt Pattern	Critical Instructions
Remove person from crowd	"crowd of people without the person in the center"	"fill with background people"
Add object to scene	"room with a [new object] on the table"	"match the room's lighting"
Expand sky	"extend sky upward, add more clouds"	"match cloud style and color"
Fix damaged area	"repair the [damaged region] with [correct content]"	"blend with surrounding area"

5. ITERATIVE EDITING WORKFLOW
   - Step 1: Generate base image
   - Step 2: Identify region to edit (mask or description)
   - Step 3: Generate edit with context preservation prompt
   - Step 4: Review for seam artifacts
   - Step 5: If artifacts present, regenerate with "seamless blend" instruction

- Step 6: Repeat for additional regions

6. COMMON FAILURE PATTERNS AND FIXES

Failure	Cause	Fix
Visible seam	Missing context in prompt	Add "seamless blend, no visible edge"
Style mismatch	No style preservation instruction	Add "match original lighting and color"
Repeated pattern	Outpainting over-extrapolates	Add "vary the pattern, natural continuation"
Blurry edge	Low resolution edit	Edit at original resolution, use HD quality

INPUTS:

Source image description:
[DESCRIBE THE ORIGINAL IMAGE]

Edit type:
[REPLACE / REMOVE / ADD / EXPAND / REPAIR]

Region to edit:
[E.G., "center of the image, approximately 20% of width"]

What to put there (for replace/add):
[E.G., "a red bicycle" or "more sky and clouds"]

```
RULES:
- Always describe surrounding context (prevents mismatched edits)
- Add "seamless blend" for edge transitions (reduces artifacts)
- Match lighting explicitly ("match the warm sunset lighting")
- Edit at original resolution (downscaling loses detail)
- Test on a small region first before large edits
- Multiple small edits are better than one large edit
- DALL·E 2's inpainting is more reliable than DALL·E 3's (currently)
```

How To Use It

- Always describe surrounding context — prevents mismatched lighting and style.
- Add “seamless blend” for edge transitions — dramatically reduces visible seams.
- Match lighting explicitly — “match the warm sunset lighting of the surrounding area.”
- Edit at original resolution — downscaling loses detail needed for blending.
- Test on a small region first before attempting large edits.
- Multiple small edits are better than one large edit — easier to control.
- DALL·E 2’s inpainting is currently more reliable than DALL·E 3’s.

Example Input

Source image description:
“Beach at sunset, orange and purple sky, ocean waves, empty sand in foreground”

Edit type:
“ADD”

Region to edit:
“Foreground sand, center-left”

What to put there:
“A wooden beach chair and a small cooler”

Why It Works

Most users think inpainting means “select area, type prompt” — and get bad results because they don’t tell the model to preserve surrounding context.

This framework improves outcomes by forcing:

- edit type classification (replace, remove, add, expand, repair)
- style preservation requirements (lighting, color, texture, perspective)
- task-specific prompt patterns (proven templates for each edit type)
- iterative workflow (edit, review, refine)
- failure pattern recognition (what went wrong and how to fix it)

Failure modes this prevents:

- Visible seams at edit boundaries (no “seamless blend” instruction)
- Style mismatch (new object doesn’t match lighting/color)
- Repeated patterns in outpainting (over-extrapolation)
- Blurry edges (low resolution edits)

This improves on: Basic “select and edit” approach. Context-preserving prompts produce seamless edits.

Related to: DE-05 (Style Consistency) for matching across edits; DE-02 (Text) for adding text to edited regions.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also DALL·E vs. Midjourney Translator