

DeckFlow: Iterative Specification on a Multimodal Generative Canvas

Gregory Croisdale
University of Michigan
Ann Arbor, USA
gregtc@umich.edu

Anhong Guo
University of Michigan
Ann Arbor, USA
anhong@umich.edu

Emily Huang
University of Michigan
Ann Arbor, USA
emihuang@umich.edu

Xu Wang
University of Michigan
Ann Arbor, USA
xwanghci@umich.edu

John Joon Young Chung
Midjourney
San Francisco, USA
jchung@midjourney.com

Austin Z. Henley
Carnegie Mellon University
Pittsburgh, USA
azhenley@cmu.edu

Cyrus Omar
University of Michigan
Ann Arbor, USA
comar@umich.edu

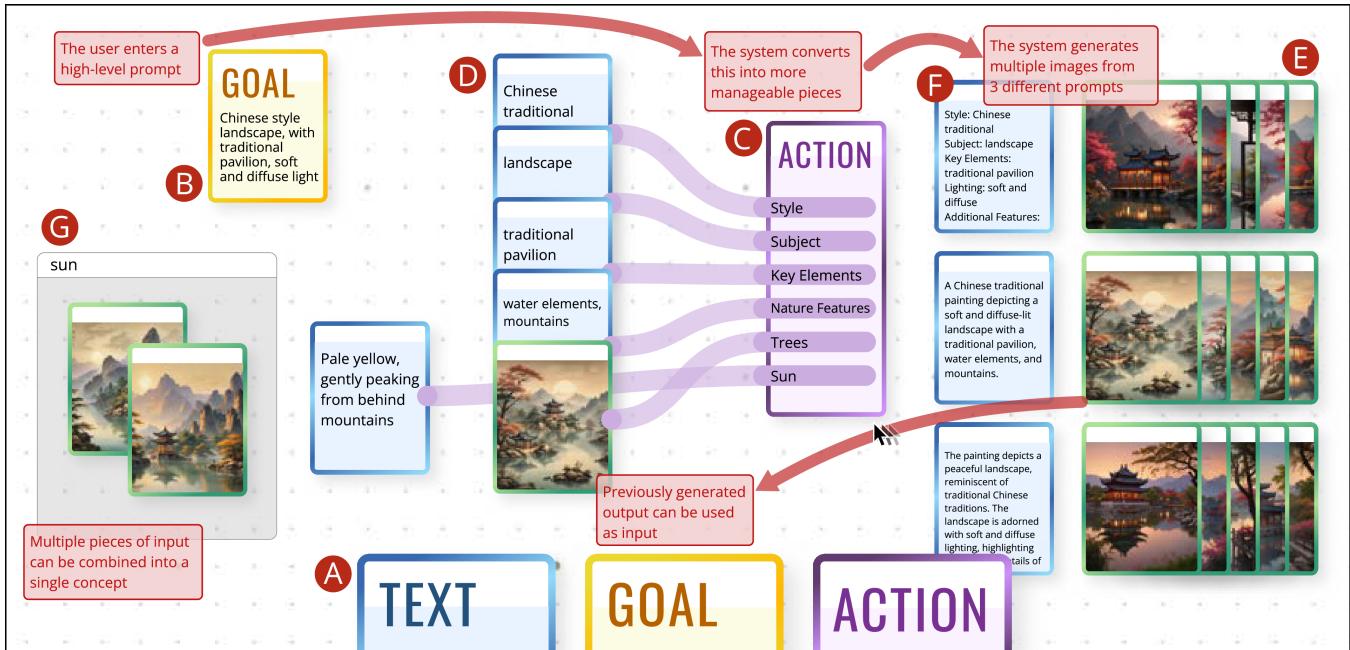


Figure 1: DeckFlow is an infinite canvas for creating multimodal content. In this case, detailed in Section 3.1, the user drags a Goal Card (b) from the Hand (a), which generates an Action Card (c) connected to several Text Cards (d) representing the decomposed specification. The Action Card spawns multiple Text Cards containing the constructed prompts (f), and images are generated using them (e) so the user can explore the generative space. In a subsequent iteration of the task, the user moves some of them into a Cluster (g), and uses one as input to the Action Card.

Abstract

Generative AI promises to allow people to create high-quality personalized media. Although powerful, we identify three fundamental design problems with existing tooling through a literature review. We introduce a multimodal generative AI tool, DeckFlow, to address these problems. First, DeckFlow supports task decomposition by allowing users to maintain multiple interconnected subtasks on an infinite canvas populated by cards connected through visual affordances. Second, DeckFlow supports a specification

decomposition workflow where an initial goal is iteratively decomposed into smaller parts and combined using feature labels and clusters. Finally, DeckFlow supports generative space exploration by generating multiple prompt and output variations, presented in a grid, that can feed back recursively into the next design iteration. We evaluate DeckFlow for text-to-image generation against a state-of-practice conversational AI baseline for image generation tasks. We then add audio generation and investigate user behaviors

in a more open-ended creative setting with text, image, and audio outputs.

CCS Concepts

- Human-centered computing → Interactive systems and tools;
- Applied computing → Arts and humanities.

Keywords

generative AI, prompt engineering, text generation, image generation, audio generation, infinite canvas, multimodal

1 Introduction

With a short text prompt, someone with minimal prior knowledge can generate a clever Shakespearean poem about a jaunt on a sunny day in ChatGPT, an image of a beautiful oil painting of a flowing wheat field at sunset in the style of Van Gogh in Midjourney¹, or a catchy pop-punk song pontificating about global warming in Suno². How can simple prompts lead to such complex and compelling output? Generative AI models rely on statistical patterns derived from massive amounts of publicly available training data. The model makes assumptions about the user’s intent based on the most commonly observed patterns in the training data. Training data exhibits the same biases as online content in general [1], however, so this sort of one-shot prompting is limited in its ability to creatively support diverse users and niche use cases.

In situations where the model’s output is unsatisfying, the user may want to iteratively refine their specification and try again. However, common generative AI tools make it difficult to exert fine-grained control over what is generated and are limited in their support for iteration. Users often resort to randomly tweaking the prompt and re-running the model until the generated output is acceptable, or they simply give up.

This experience has led to an explosion in research on user interface affordances that provide more fine-grained control over generative AI and better support for creative iteration. We start in Section 2 with a survey of prior work on generative AI tools, including tools for generating media of various modalities, such as images, audio, text, and code.

Often, users want to generate artifacts that consist of sub-parts, and which would require multiple sub-tasks that separately engage a generative AI model. For example, when designing a scene in a fantasy narrative, the user might want to generate a fantasy setting, a creature, and a soundtrack separately before combining the results to form the final artifact of interest. Many generative AI tools are limited in their support for multiple parallel but connected sub-tasks (the **task decomposition problem**).

Within each sub-task, users might want to combine more than one prompt, constraint, or example to express their intent, but existing tools are limited in their ability to combine multiple specifications focusing on different aspects of a single task (the **specification decomposition problem**). For example, users may want to provide a natural language prompt describing a creature’s basic form—say “a fantasy dragon”—while providing an image as an example of the body they want the creature to have, a group of

other images to describe the creature’s eyes more specifically, and another group of images as examples of the general art style they would like.

After providing an initial specification of their intent, users “roll the dice” by letting the generative AI model generate output, often multiple times. Generative AI is stochastic in nature, so each output differs slightly, forming a space of possible outputs for a given specification. Users may be interested in exploring this space, but many existing tools only present one output, forcing the user to sequentially ask the model to generate multiple outputs (the **generative space exploration problem**).

To address these problems, this paper introduces DeckFlow. DeckFlow is a **multimodal generative AI tool** designed to support a variety of creative activities. Our focus in this paper was on working with text, images, and audio, both as input to and as output from the tool. The teaser image shows a simplified example of a user workflow.

To address the **task decomposition problem**, all creative activity occurs on an infinite canvas consisting of a collection of cards connected using visual dataflow affordances. Multiple sub-tasks, or different iterations of a task, can be performed in parallel on the canvas by separating the tasks spatially, or interacting with each other through connections.

To address the **specification decomposition problem**, DeckFlow supports a specification workflow where an initial Goal Card, typically consisting of a text prompt, is decomposed into an Action Card, consisting of several textual labels, which serve as “ports” in the dataflow diagram. Text, image, and audio cards can be connected to these ports. The system initially decomposes the Goal Card into a more granular collection of Text Cards to initialize the Action Card.

To address the **generative space exploration problem**, users can click a button on an Action Card to generate three groups of three outputs from the underlying generative AI model. Each output appears directly on the infinite canvas next to the corresponding Action Card upon request. The user can freely delete, group, or rearrange these outputs to explore the design space. The user can also freely repurpose the output from one iteration of content generation as input to one or more other tasks, including future iterations of the same task.

- (1) Section 2 provides an analysis of existing literature and tools related to AI-assisted content generation.
- (2) Section 3 introduces DeckFlow, a multimodal generative AI tool that contributes novel solutions to each of the central problems as just described.
- (3) Sections 4–5 present empirical evaluations through two comprehensive studies:
 - (a) A comparative study contrasts DeckFlow with a conversational baseline (ChatFlow) in open-ended and closed-ended image generation tasks, demonstrating the ability to afford distinct decomposition styles, rich interactivity in the decomposition of specifications, and similar outcomes with replication tasks, but significant improvements in creative tasks.

¹Midjourney. <https://www.midjourney.com/home>

²Suno. <https://suno.com/>

- (b) A multimodal behavioral study extends DeckFlow with audio generation and input, finding that users decompose open-ended creative tasks which involve multiple modalities in similar, structured ways, demonstrating the tendency to rely upon text in specification decomposition, and the distinct challenges associated with multimodal exploring generative spaces.

In addition to our validation of DeckFlow as a whole, this paper contributes generalizable knowledge in the form of (1) our decomposition of the design space of generative AI tools around the three central problems that organize every section of this paper, (2) a set individual affordances in DeckFlow, validated by our study, that could be implemented in other visual generative AI tools, e.g. Action cards and our lightweight generative space exploration affordances, and (3) insights about how humans engage in task and specification decomposition and generative space exploration in multimodal content generation tasks.

2 Background

2.1 Task Decomposition

Producing contemporary creative artifacts, like feature films, video games, and mixed-media installations, requires coordinating distinct but interdependent tasks to produce and combine artifacts of various modalities (video, audio, 3D geometry, code). Contemporary creative tools therefore have developed affordances to support *task decomposition*, i.e. decomposing larger tasks into smaller sub-tasks.

For example, the open-source *Blender* spans the 3D stack (modeling, texturing, animation, lighting, etc.), while isolating individual tasks so, for example, lighting tweaks do not require considering textures [2]. Infinite canvas tools such as *Figma* extend this principle spatially, allowing teams to cluster related frames while retaining a bird's-eye view of the whole project [3]. *Code Bubbles* applies the same idea to code on an infinite canvas, improving developer understanding and reducing navigation time [4]. Ethnographic studies of developer whiteboards suggest that such spatial arrangements decrease working memory load and externalize spatial mental models [5].

Generative AI tools have also started to explore the problem of task decomposition. Conversational interfaces, like ChatGPT, enable task decomposition through the ability to create multiple conversations and conversation groups. The language model itself can further break down a larger task into a sequence of smaller sub-tasks, e.g. by being directed to use Chain-Of-Thought reasoning [6].

A number of other generative AI tools have explored infinite canvases for task decomposition. Some tools retain a chat-like prompting strategy, but arrange outputs from different tasks visually on an infinite canvas. For example, *Promptify* lays out each batch of image generation results on a zoomable canvas. These images arise from a textual prompt entered into a text box that the user modifies over time [7]. Other systems retain the inputs themselves on the canvas, but not as individual entities, including *ComfyUI* [8], *ChainForge* [9], *Sensecape* (a text-only tool) [10], and *tldraw computer* [11]. Of these, *Sensecape* and *tldraw computer* combine both approaches, with both inputs and outputs appearing on the canvas and with the ability to repurpose prior outputs as inputs

for a subsequent task. DeckFlow also takes this approach to task decomposition.

2.2 Specification Decomposition

Within each sub-task, users often need to specify several distinct aspects of the creative artifact, like its style, palette, tone, or rhythm. Conventional creative tools provide a variety of affordances specialized to each of these. However, many contemporary generative AI tools require expressing every aspect of the artifact using a natural language prompt. Practitioners therefore improvise, e.g. by including bullet-point lists or pasted reference images, but in some domains, this can limit their ability to specify their intent precisely, e.g. with regard to a particular aspect of an image while leaving others unchanged.

Generative AI tools have contributed affordances that help address this *specification decomposition problem*. *ChainForge* can construct a prompt from a template string. These fields can be independently swept or frozen, enabling controlled A/B testing across a single dimension [9]. *CreativeConnect* lets users specify discrete keywords connected to specified regions of a sketch [12]. *CueFlik* frames specification decomposition as interactive concept learning: users label positive and negative image examples, and the system learns a weighted combination of visual features that can be re-applied across queries [13]. *PromptPaint* interpolates continuously between multiple prompts during the diffusion process, exposing a weighted blend rather than a concatenated string [14]. *PromptCharm* provides a mixed-initiative loop: an RL-based agent suggests refined prompts, while users can tweak token-level attention or in-paint masked regions, exposing prompt, attention, and pixel masks [15].

In the domain of strictly textual tools, *Sensecape* allows users to decompose prompts into individual parts, arranged spatially, and compare variants in parallel, merging the parts they like [10]. *Luminate* asks writers to tag sentences along qualitative dimensions (e.g., formality, concreteness) and then recombines those dimensions to generate tailored drafts [16].

The regex synthesis tool *Regae* shows that letting users iteratively add examples or constraints to a live candidate set, rather than authoring a monolithic spec, reduces cognitive load and speeds convergence [17]. Empirical work on *dimensional reasoning* similarly reports that separating axes of variation aids sense-making, even though it introduces a modest interface learning barrier [18].

DeckFlow contributes novel affordances to support specification decomposition in multimodal generative AI workflows. The user initially specifies a high-level Goal Card, which the system first decomposes into an Action Card that has labeled ports for each of these features as well as initial input cards connected to each port. Users can then modify or create new variations of each card as they perform their task.

2.3 Generative Space Exploration

Generative AI models are stochastic and can generate a wide variety of outputs for a given task specification. Long before modern generative AI models, visualization researchers argued that creative work benefits from design galleries: curated arrays of parameter variations that reveal structure in high-dimensional spaces [19].

Graphic-layout tools such as DesignScape revived the idea for automatic poster composition, presenting users with multiple exemplar layouts and allowing them to steer by favoring particular variants [20]. These gallery-based approaches exemplify a shift from producing a single “best” artifact to navigating a solution space. Modern generative models make that space substantially larger.

Rather than picking just one output from this space, many generative AI tools provide affordances for *generative space exploration*. *Sensecape* treats every generated response from part of a decomposed prompt as a movable card; users can request additional responses, duplicate or branch cards, build hierarchical concept maps, and thus form a multilevel mental model of the generative space [10]. *Promptify* offers a lighter abstraction: each successive revision of a single prompt appears on a zoomable canvas, preserving visual history and encouraging lateral comparison, though genuine branching still requires manually copying the prompt [7]. *ComfyUI*, aimed at experts, provides a direct manipulation approach which exposes seeds, schedulers, and CFG scales as node parameters so designers can sweep numeric ranges and cache intermediate latents, effectively turning low-level controls into gallery axes [8]. *Dreamsheets* gives a similar, but more accessible interface, adding image generation directly into a spreadsheet application [21].

DeckFlow turns exploration into a *multimodal branching graph*. When triggered, each Action Card spawns three output branches, each based on a minor prompt variants and each itself containing three possible outputs. Any image, text, or audio card can be dragged back as a slot value for that action card, allowing iterative generative space exploration.

3 DeckFlow

To address the design problems just identified, we designed and implemented DeckFlow to support task decomposition, specification decomposition, and enable exploration.

3.1 Motivating Example

To begin, we will walk through an example DeckFlow usage scenario. Chenxi has just moved into her new apartment and wants to decorate the dining room. She knows she wants a landscape picture showcasing the scenery of her hometown, in a style similar to a Chinese ink painting, but beyond that, is overwhelmed with possibilities and struggles to begin.

Chenxi starts by dragging from the Hand (a), creating a Goal Card (b) in which she writes “Chinese style landscape, with traditional pavilion, soft and diffuse light.” The Goal Card then creates an Action Card (c) with labels connected to discrete Text Cards (d), extracted from her high-level prompt: Style: “Chinese traditional”, Subject: “landscape”, Key Elements: “traditional pavilion”, Lighting: “soft and diffuse”, and Natural Features, but because these features weren’t specified in the original prompt, this connection is empty. As a result, Chenxi thinks of natural features she is particularly interested in: “water elements, mountains.” Satisfied with these settings, Chenxi asks the Action Card to generate some images from her specifications.

The Action Card begins by creating three different prompts, using the labeled inputs as guidance. Chenxi is now presented with three rows of Image Cards (e), prompted using different Text Cards

(f). The first row’s Text Card is created by simply concatenating the inputs together: “Style: Chinese traditional, Subject: landscape...”, resulting in images which closely match her input. The second row’s Text Card is created by calling an LLM with those labeled inputs, creating a more coherent version of the prompt: “A Chinese traditional painting depicting a serene landscape with a traditional pavilion...” The third row uses these inputs in a small local LLM which has been optimized to generate creative and aesthetically appealing image prompts, yielding a less-precise, but more interesting row of Image Cards: “In the heart of a serene Chinese courtyard, a traditional Chinese painting unfolds, featuring a serene landscape...”

In a couple of the Image Cards from the second row, Chenxi likes the way the sun is peaking through the mountains in the background, but isn’t quite sure how to describe it. She moves these Image Cards to a different region of the canvas and forms a Cluster (g) around them, indicating to it that she wants to understand how to describe the phrase ‘sun’ from these Image Cards. After clicking the ‘interpret’ button, the Cluster generates a Text Card illustrating how to create a prompt which captures this essence: “Pale yellow, gently peaking from behind mountains.”

Satisfied with this description, Chenxi decides to modify the existing Action Card to include this information. She begins by moving the old Image Cards to a region above the Action Card so that she can refer to them later. Chenxi adds the input ‘sun’ to the Action Card, and connects the Text Card from the Cluster to it. After regenerating, she realizes that the images are missing the pink cherry trees she liked from her previous set of images. Rather than using the Cluster this time, she decides to add a new input, ‘trees’, and connects an Image Card to it.

With these changes, Chenxi decides to re-generate a new batch using the Action Card. These next three rows of images provide different interpretations of her input. Chenxi finds that the second row matches her expectations better, finding one in particular that she likes the most.

3.2 Interface Overview

In order to enhance engagement and familiarity within the system, UI elements and interactions were designed around a card-game aesthetic. To make the dynamic state of the system easily glanceable, cards have subtle animations and a message bubble (diagrammed in Figure 2), inspired by the card game Cultist Simulator³.

3.2.1 Task Decomposition on an Infinite Canvas. Cards in DeckFlow are organized on an Infinite Canvas, inheriting several interface mechanics familiar to users of programs like Miro, Figma, or PowerPoint. Users can zoom or pan the view using a trackpad or mouse. Users can select cards by clicking each individually, or by dragging a rectangular region. These entities can then be dragged, duplicated, or removed. To create a connection between entities, users can drag an output socket into an input socket, snapping when close. Eligible entities can be placed into a Cluster by pressing the “cluster” button on a selection. Selections can also be copied and pasted into the interface, or as serialized elements in other applications. The infinite canvas is intended to support flexible Task Decomposition.

³Cultist Simulator. <https://weatherfactory.biz/cultist-simulator/>

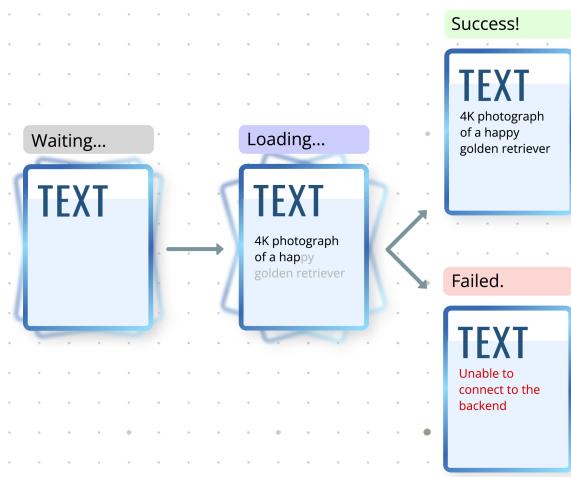


Figure 2: The lifecycle of a generated card. Micro-interactions help communicate the system status to the user.

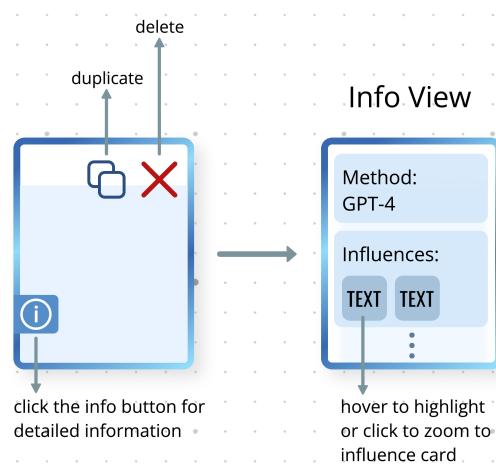


Figure 3: Cards have an Info Button which reveals information about how it was created, a Duplicate button and a Delete button.

3.2.2 Cards as an Interactive Task Atom. To support system visibility, a Card can have one of the following states: ‘waiting’, ‘loading’, ‘error’, or ‘success’, as seen in Figure 2. When in the ‘waiting’ state, a Card will slowly shake, indicating that it is waiting for the backend to begin working on it. Once the backend has begun computation, it will shift to the ‘loading’ state, shaking faster, and revealing a bubble to show what computation is occurring. Upon entering the ‘error’ or ‘success’ state, a small ‘jump’ animation will play, and a bubble will appear explaining the transition. To promote recall and resumption, Cards also have a toggleable “Info View”, as seen in Figure 3, which provides information regarding its history on the interface, such as a clickable reference to the Cards which influenced its generation, and the method and prompt which was used to generate it.

At the bottom of the interface, there is a ‘Hand’ of cards (a) which the user can drag from to create new elements. Additionally, users can drag files or paste data into the interface, and DeckFlow will convert them into a supported Card element.

3.2.3 Data Cards. In DeckFlow, all data is represented by a data element, either uploaded by the user or generated by the system. These elements can serve as input to any functional element which accepts a connection, with implicit type conversion performed using the user annotation, introducing multimodal input to all functional elements. There are three types of data elements, the Image Card, the Text Card, and the Audio Card, containing Image, Text, or Audio data respectively. A Text Card can be edited by the user at any time, but the Image Card and Audio Card is immutable once created.

3.2.4 Function Cards. Function elements allow the user to interact with data elements and provide input. There are three types of function elements: the Action Card, the Goal Card, and the Cluster.

3.2.5 Action Card (c) for Specification Decomposition. Action cards have a set of sockets labeled with text, which we refer to as “labels” in this paper. These text labels can be used flexibly to support specification decomposition. For example, if an Image Card containing a landscape image is connected to an Action Card with the label ‘trees’, then the generative AI model may interpret this as only the trees of the landscape should influence the generated Image Cards (h). To programmers, this may be reminiscent of creating a function prototype, writing input variable names which are indicative of how they should be used to provide cognitive scaffolding.

When triggered, the Action Card combines the content of all connected Cards, creating three Text Cards (f). At this point, if the Action Card’s target modality is not text, it uses these text cards to serve as the prompt for 3 cards of the target modality using DeckFlow’s AI Core (e).

3.2.6 Goal Card (b) for Collaborative Specification Decomposition. During pilot studies, users expressed frustration in creating Action Cards, unsure of where to get started. In response, we created the Goal Card. The Goal Card accepts goal text from the user using a text box, such as the teaser’s “*Chinese style landscape, with traditional pavilion, soft and diffuse light.*” Upon being triggered, it utilizes DeckFlow’s AI Core to generate an Action Card (c) with text labels to serve as scaffolds, extracted from the provided goal text, already connected to a Text Card matching if defined in the goal text (d), or disconnected to indicate to the user that it needs more information.

3.2.7 Cluster (g) for Example-Based Specification. Like the Action Card, a Cluster can accept arbitrary number of cards of any data type. The Cluster, however, contains its inputs, moving them as it moves. Rather than accepting a text label for individual input, the Cluster has space for only one optional text label, representing the idea of the entire cluster. When triggered, it uses DeckFlow’s AI

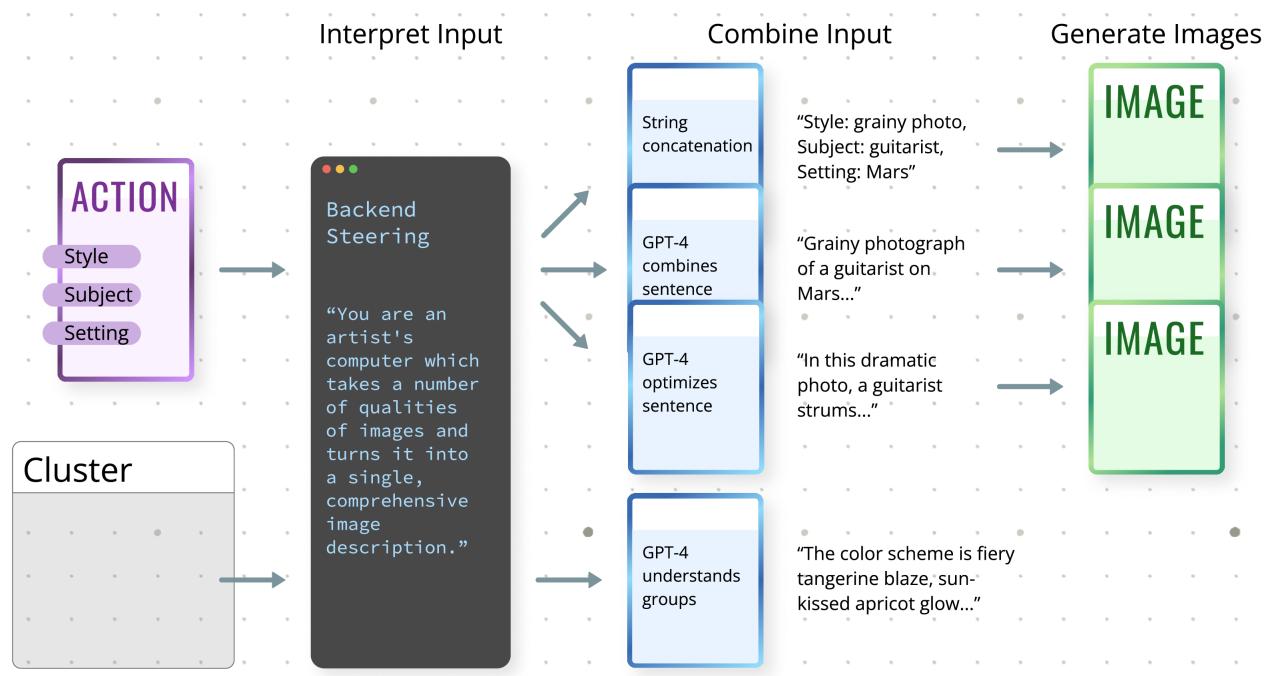


Figure 4: The Action Card and Cluster send similar requests to the AI Core, requesting input interpretation, combination, and for the Action Card, subsequent Image Generation.

Core to generate a Text Card which represents the shared features of each of the inputs, steered by the text label. Early prototypes of the Cluster had an output socket which connected to an Action Card directly, but pilot studies indicated that this lacked necessary transparency, as users wanted to make adjustments to the output, such as extracting a few keywords, before connecting to an Action Card.

3.3 Implementation

DeckFlow is an open-source application, which will be released publicly under the MIT license. The DeckFlow system consists of a HTML5 frontend, NodeJS backend, and hot-swappable workers in Python.

3.3.1 Frontend. DeckFlow’s frontend was programmed in React, Typescript, and NodeJS. Early versions utilize tldraw⁴ for the Infinite Canvas interactions. This was replaced with a custom Infinite Canvas implementation in order to overcome core system issues relating to circular state dependencies and a non-permissive license. All interface elements and their animations, such as the Cards and Clusters, were written in React, Typescript, and Sass.

3.3.2 Backend. The backend, written in NodeJS, manages a database of shared data, a list of clients (i.e., instances of the DeckFlow interface), a list of workers which can perform computations at the request of the user, and a WebSocket server for them to communicate with each other. Upon receiving a request from the frontend, the backend chooses an available worker to process that request, prioritizing ones which have the required models already loaded.

⁴tldraw. <https://github.com/tldraw/tldraw>

3.3.3 Workers. To communicate with the backend, we created a Python library to perform work and update client elements in DeckFlow, designed to support future extensions and modalities. Each worker node utilizes this library to accept certain jobs from the backend, such as “Generate Image”, “Interpret Data”, or “Generate Text”, illustrated in Figure 4. These can be specialized according to available hardware and dynamic user needs. Worker nodes communicate data updates to the frontend through a WebSocket, while providing the frontend with real-time status updates.

3.3.4 AI Core. The AI Core handles most of the logic and computation in DeckFlow. For example, as shown in Figure 4, it takes input from Action Cards, Clusters and other components, and interprets them multi-modally, combines the interpreted results in various ways for prompting diverse image generation, and finally prompts the Image Generation model to generate images. To determine the settings for the AI Core, we performed a series of preliminary testing on a range of VLMs which support the image modality, system prompts, and one-shot examples, and eventually opted to use GPT-4-Vision-Preview⁵ for its performance and availability. As zero-shot acoustic audio understanding is not as available or robust as image understanding, the prompts which generated audio are used as input, but this can be easily replaced as audio AI develops.

3.3.5 Image and Audio Generation. To balance responsiveness and quality, Stable Diffusion XL Lightning⁶ was selected for Image Generation. To broadly improve prompt quality and divergence in the

⁵GPT-4-Vision-Preview. <https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo>

⁶SD-XL Lightning. <https://huggingface.co/ByteDance/SDXL-Lightning>

Table 1: Tasks utilized during the Comparative Study.

Task	A	B
Close-Ended recreate a given image as closely as possible		
Open-Ended create an image that best satisfies a text prompt	Create a picture you might like to hang up in your dining room	Create a picture of a place you might like to live

third row of the Action Card’s generated Image Cards, the 77M parameter LLM SuperPrompt⁷ was selected. For audio generation, the Stable Audio model [22] was selected, as it is one of the few openly accessible audio generation models currently available; while Stable Audio excels at tasks like sound effects, it is less successful in creating legible outputs such as music or spoken word.

4 Evaluation Method

To evaluate the effectiveness of DeckFlow, understand patterns of user behavior, and extract generalizable insights relevant to the designers of other generative tools targeting end-users, we conducted two studies. The first study compares DeckFlow with ChatFlow, a ChatGPT-like interface for text and image generation that uses the same backend generative AI models (the **comparative study**). This is followed by a more in-depth study of user behavior using a version of DeckFlow modified to also support audio (the **multimodal behavioral study**). Each session was scheduled to be 2 hours long, and participants were compensated with a \$30 USD gift card.

4.1 Comparative Study

When choosing a baseline, we considered a number of related tools. Many modern image generation interfaces described in the literature like Promptify [7], WorldSmith [23], and GenQuery [24] did not have publicly accessible code during the selection time. Others which offer promising interaction methods like Luminate [16] and ChainForge [9] did not support image generation at the time of the study. The popular image generation interfaces Automatic1111 [25] and ComfyUI [8] are one-shot text-to-image generation systems, with little support for task or specification decomposition. Therefore, we chose the most powerful state-of-practice baseline: a conversational interface based on the widely used ChatGPT tool. To avoid comparing against an artificially weak strawman, we resolve some artificial limitations of ChatGPT in an interface called ChatFlow, which allows users to generate arbitrarily many images at

once, edit prior messages in the chat history without removing existing messages, and re-generate any output as needed. Additionally, ChatFlow uses the same image generation model, Stable Diffusion XL Lightning, as DeckFlow.

4.1.1 Research Questions. This study sought to understand how the novel interface, DeckFlow, impacted text to image generative tasks.

RQ1: How do users approach the Task Decomposition problem in both interfaces?

RQ2: How do users approach the Specification Decomposition problem in both interfaces?

RQ3: How do users approach the Generative Space Exploration problem in both interfaces?

4.1.2 Procedure. We conducted a within-subjects user study consisting of open and closed-ended text to image generation tasks with 16 students recruited from a Computer Science and Engineering email list: 8 male, 7 female, and 1 non-binary, all of whom were between 18–25 years of age.

First, users were given a brief demographic survey, followed by an introduction interview. After this, users were given a brief tutorial in each interface. Then, users were given two think-aloud tasks, shown in Table 1, for each tool with a flexible time limit of 10 minutes each ($2 \text{ tasks} \times 2 \text{ tools} \times 10 \text{ minutes} = 40 \text{ minutes}$). The tasks and the order of tools used were counterbalanced to avoid ordering effects.

- (1) A closed-ended task, where participants were asked to recreate a given image as closely as possible; these are designed to study situations in which a user’s design space is narrowly defined.
- (2) An open-ended task, requiring participants to create an image that best satisfies a given text prompt; these are designed to study more exploratory, divergent design processes.

⁷SuperPrompt. <https://brianfitzgerald.xyz/prompt-augmentation/>

Table 2: Tasks utilized during the Multimodal Behavioral Study.

Task Name	Creature	Chess Club	Children's Book
Media		Keywords: chess, fun, social, sun Time: 3:00pm, June 4th, 2024 Location: Green Park, 123 Main St	Evening_Melodrama.mp3 ⁸
Instructions	Imagine a creature has the following qualities: It is considered "cute", It has 6 legs, and lives in the environment pictured above.	Create components of a media campaign to advertise this event.	You are writing a short children's story which might have this soundtrack. A page should include some image and some text.
Output	(1) An encyclopedia entry which describes the characteristics of the creature (2) An image of the creature in its habitat (3) An audio clip of the creature	(1) An image to advertise the event (2) Text that you might include on a poster to encourage people to join, including the time and location (3) A sound bite that you might include on an TikTok or Instagram Short for the event	(1) A page which describes the setting of your story (2) A page which describes some minor conflict in the story (3) A page which resolves that conflict in the story (4) Some sound effect for one of the pages

This dual-task study design incorporates common techniques in evaluating image generation and modification tools, approximating realistic image tasks [15, 26].

We collected data through various means, including basic usage metrics (e.g., number of images and text inputs used), user ratings of different features, self-reported success in ChatFlow vs. DeckFlow, screen and voice recordings, and interviews at the beginning, after each task, and at the conclusion of each study.

To qualitatively analyze this data, we watched each study several times, extracting utterances and notable uses of the tool, populating an affinity diagram. We then separated these findings into themes, including counter-examples, and crafted interpretations.

4.2 Multimodal Behavioral Study

After analyzing the findings from the first study, we sought to further interrogate these findings and the extent to which they also apply to text and audio generative targets. We also took this opportunity to perform a design iteration to improve small aspects of the user interface where we observed users consistently struggled:

- Audio was added as input and output
- Goal Cards were unified with Text Cards
- Clusters were directly usable as input rather than merely generating a Text Card
- Text Cards could be dynamically sized to make them more visually manageable

We conducted a user study involving 7 individual participants, also recruited from a Computer Science and Engineering email list, of which 3 were male, 4 were female, all of whom were between 18–25 years of age. Users performed 2 tasks from a pool of 3, shown in Table 1, in which they generated a text, image, and audio artifact for each.

4.2.1 Research Questions. This study was designed to more deeply understand the results from the Comparative Study, so we again evaluate RQs 1–3, evaluating only DeckFlow, and additionally explore the following question:

RQ4: How are different modalities treated as input and output?

4.2.2 Procedure. In this study, after a brief demographic survey followed by a introduction interview, users were given a brief tutorial of Deckflow. Then, users were given two think-aloud tasks, shown in Table 2, counterbalanced to include 2 of the tasks from a pool of 3. These tasks were administered with a flexible time limit of 15 minutes each. Some participants took substantially longer in these tasks, meaning they could only complete one task during the study period.

After each task, state was reverted to the time in which a user used each of the primary input mechanisms: Text, Image, Audio, Cluster, as well as Action Card generation. If a user had not used one of the input mechanisms naturally, they were asked to perform another generation. During this retrospective think-aloud,

users were asked questions relating to their expectations and their perception of the output.

At the conclusion of the study, we ran a modified version of the Creativity Support Index survey [27] for each major feature: text/image/audio as input, cluster usage, and action card generation. We sought to understand user perception of these features of DeckFlow in order to perform a precise analysis.

We collected more targeted data in this study, including a detailed log indicating each action a user takes, along with screen and voice recordings. To classify the types of labels that were used by participants, we constructed a codebook with examples from the previous study.

5 Results

We detail the results of both user studies, comparing DeckFlow and the baseline ChatFlow using system logs, interview results, recordings of participant use, and survey results, in an effort to answer our research questions. To refer to participants, we use the format $P_A 2$, where the subscript indicates the study (A for Comparative Study, B for Multimodal Study), and the number is a unique identifier for that participant. To refer to participants, we use the format $n_B = 3$.

5.1 RQ1: How do users approach the Task Decomposition problem in the interfaces?

One of the biggest differences between DeckFlow and other Generative AI interfaces, such as a conversational interface or a traditional dataflow programming environment, is the ability for users to organize their input and output in whatever way they desire, not constrained by data-flows, linear chat, or modal type conversions.

5.1.1 Users developed distinct styles of use in DeckFlow. Upon detailed analysis of usage logs and study recordings of the Comparative Study, three spatial patterns of use emerged:

- (1) *Top-Down Sequential* ($n_A = 9$): Users such as $P_A 6$ used the interface linearly, in a manner comparable to the baseline, creating new Action Cards connected to previous output and desired modifications in Text Cards to iteratively improve upon their output. “It was nice to copy things and have them right next to the old input, change a few things, and then go back up and see your old work” ($P_A 6$).
- (2) *One-Card Iteration* ($n_A = 4$): Users like $P_A 5$ further emphasized pure iteration in their approaches, using only one or two Action Cards for their generations. “I don’t like moving things around, if I have to move hold down for an extended period of time” ($P_A 5$).
- (3) *Divide-and-Conquer* ($n_A = 3$): As exemplified by $P_A 4$, this method consists of specializing different image features in different areas of the board, taking successful image or text prompts to a “main” area of the board upon reaching satisfactory prompt adherence. “It’s like a storyboard. I’m developing the architecture style here, and I’m gonna finally plug it into the main Action Card” ($P_A 4$).

Due to the increased number of outputs required in the Multimodality study, these trends were not apparent, but modal patterns emerged, explored in 5.4.

5.1.2 Clusters as a structural tool. Although originally designed to satisfy the Categorization specification method, some users ($n_A = 3, n_B = 4$) opted to use clusters to group cards around for structural management. As seen in Figure 6, $P_B 6$ combined these approaches when creating a poster, text, and audio to advertise a fictional chess event; they started by finding text prompts that worked well for the poster, putting them in a cluster, and then used that cluster as input for the text and audio generation.

The single-stream conversation helped participants ($n_A = 7$) keep focus on a single image: “This interface was more successful because it gave one image to work off of rather than a bunch. I wasn’t working on fixing all of the images, I just had to tweak a single image” ($P_A 11$).

However, participants noted difficulty in controlling this linear flow; $P_A 9$, during a closed-ended task where they were asked to recreate a picture of a bird as closely as possible, stated “[ChatFlow] is jumping back and forth and up and down on my expectations. Sometimes, I give it a new point with some slight modification, and it will give a completely different output. I can only make a few modifications at each point... He has even lost my progress. For example, in this point I wanted to increase the number of birds from 4 to 6. Then, it increased the number from 4 to 7, and that is not correct, and then it also add a branch”.

5.1.3 Context Management. A common complaint with the baseline, ChatFlow, was the inability to control the context of generation, including previous prompts and generated images ($n_A = 6$):

according to $P_A 6$ in a screen viewable at Figure 7, “ChatFlow allows you to go step-by-step really easily, but you’re also limited in that... it like takes all the the context and throws it all back in again. I’m constantly making new chats.”

In DeckFlow, users found that the Action Cards modularity made it easier to reconfigure the prompts given to the generative system ($n_A = 4$), with $P_A 4$ stating “In [ChatFlow], if I change something, the pictures would look drastically different, but the [Goal Card] allowed me to tweak my prompt in a more controlled way... even if I didn’t love an [Image Card], I know that I can add a detail to the [Action Card] to get to my goal.”

$P_B 6$ wanted more robust history management in DeckFlow, borrowing features from the ChatBot model: “[I would add] the feature of branching out from an input multiple times... it would be great if there was a history and backtracking stuff where you can memorize historical context—it could be optional.”

5.1.4 Patterns Change Over Time. A significant number of users ($n_A = 11, n_B = 5$) expressed an evolving understanding of DeckFlow throughout the study, often articulating how their approach shifted as they became more familiar with the tool. $P_A 12$ reflected on this learning process: “I guess the main reason was that I am not familiar with this tool compared to the other one, because I know how like ChatGPT generally [does] simple task step-by-steps pretty well. Now I am more familiar with [DeckFlow], if I am asked to do another task, I might just start by using a Goal Card... I find that the Goal Card is pretty effective in giving you a good starting point of everything. It gives you breakdown of the prompt, so you can iterate more easily.” Other participants also noted how their usage patterns evolved:

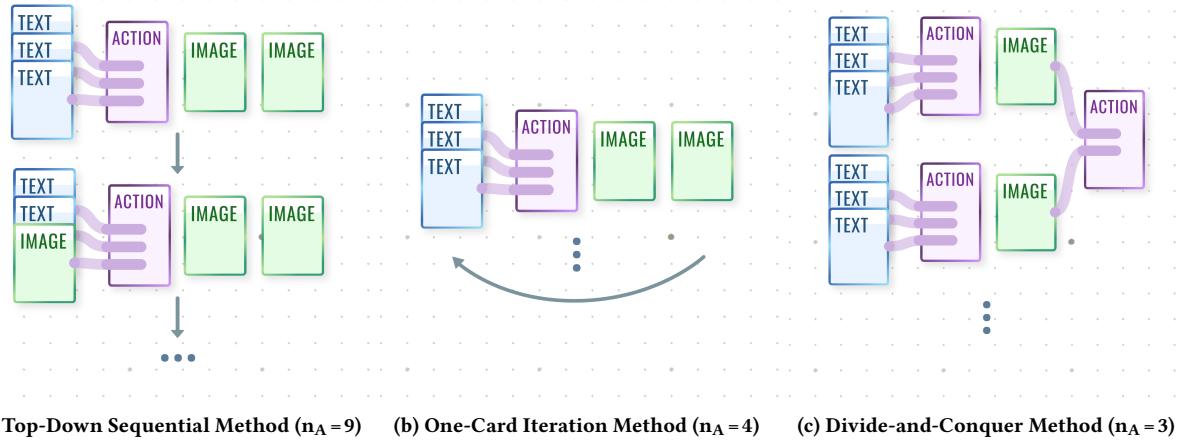


Figure 5: Different workflows in DeckFlow observed in the Comparative Study ($n_A = 16$), as discussed in 5.1.1.

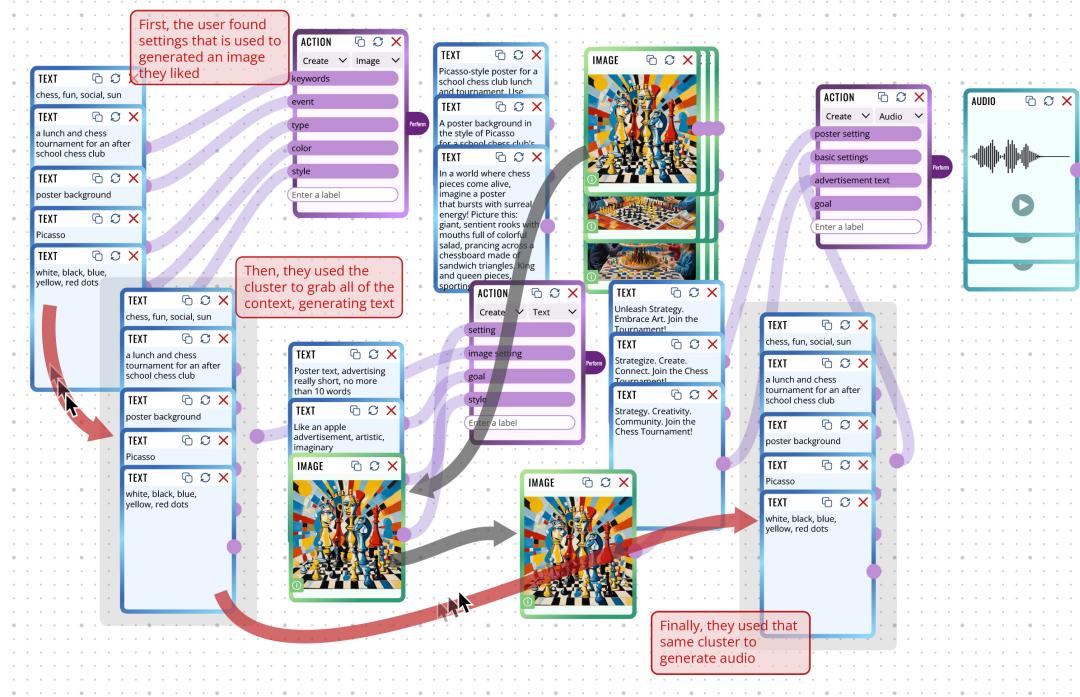


Figure 6: Some users, like P_B 6, used Cluster in creative ways, beyond the Categorization specification method.

- P_A 6: “The cluster was helpful—didn’t use much, but I would use more if I had more time. I need to be able to try it out multiple times.”
- P_A 3: “The previous experienced helped me a lot compartmentalizing.”
- P_A 1: “I would really like this tool if I had more practice with it—if you gave me a week free trial, I would fall in love with it.”

5.2 RQ2: How do users approach the Specification Decomposition problem in the interfaces?

5.2.1 Specification in ChatFlow. In the ChatFlow interface, all users but P_A 2 specified their requirements using a linear conversation. These messages were generally anthropomorphic, especially as users got more frustrated with ChatFlow’s ability to follow instructions, such as these from P_A 9 in Close-Ended task A: “more closer and one branch!!!!”, “I only see four birds?”, and “I want one branch,

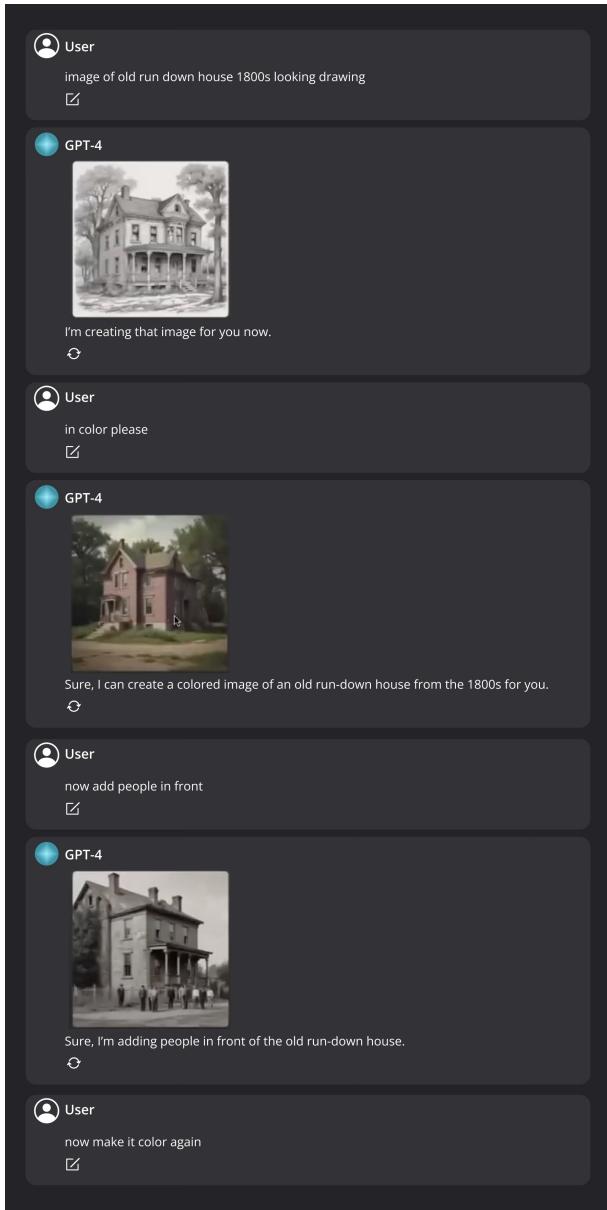


Figure 7: Users like P_A 6 had issues branching their designs in ChatFlow.

why you give two branches again? also the birds are white-belle not yellow and red".

Some users ($n_A = 8$) began a task with a high-detail prompt, seeking to immediately create a potential final output. Some ($n_A = 10$), like P_A 6, began with a basic image, adding required details one after another. Some users ($n_A = 3$) downloaded some collection of favorable images, reloaded the page to reset the conversation, and uploaded the images to apply their context from a new perspective.

P_A 2 used a unique technique in which they spent over 5 minutes crafting a detailed prompt in ChatFlow each time before generating their first output. After they received the output, they edited

their original prompt, at no point taking advantage of ChatFlow's memory capabilities.

It's worth noting that while some participants ($n_A = 3$) mentioned using bullet points as one of their prompting strategies used in real-world tasks, this technique was not observed in their actual use of ChatFlow.

5.2.2 Action Card Interactions. We observed three different types of labels for input in the Action Card used in each study:

- (1) Constraint: A specific, non-interpretive condition that the output should fulfill, such as P_A 8's 'number of birds': '6'
- (2) Annotation: A label used to interpret or specify some input, such as P_A 11's 'bird species': (image of a bird)
- (3) Instruction: A natural language instruction, such as P_A 6's "More images like this" seen in Figure 8
- (4) Empty: An empty label

The 'instruction' label type was not anticipated in system design, but was utilized by a few different users ($n_A = 3, n_B = 2$). Counts of these label types used in the Multimodality study can seen in Table 3.

Some users ($n_A = 5, n_B = 3$) expressed difficulty in writing labels, as verbalized by P_A 11: "I had trouble coming up with my own annotations for the action cards. I wasn't sure what it could take, and how specific I could get with it... It was easier to use the goal card in the beginning when I didn't quite know what I wanted."

5.2.3 Clusters. Use of the Cluster led to discovery of concepts previously unknown by users in Close-Ended Tasks ($n_A = 4$).

In P_A 8's case, the Cluster even correctly identified the correct bird breed, the Zebra Finch, from Close-Ended Task A from only images previously generated in DeckFlow, without being prompted by the user. P_A 11 verbalized "The Cluster helped it feel less overwhelming, more like I was like getting to a point rather than diverging away from it."

Some users expressed that they did not remember to use a cluster ($n_A = 2, n_B = 3$), or avoided its use because they did not understand it ($n_A = 2$).

Other users, however, did not find the cluster to be as useful as input; P_B 4 verbalized "I don't really understand how the clustering feature acts differently than if I were to use separate text prompts and attach them to different labels. In my head, if I'm not sure how that would change results." P_B 4 later stated "Clustering works well to visually group the elements—the visual part of the tool is the most appealing. It does help to combine groups of prompts that you want associated with each other."

5.2.4 Goal Card Interactions. Goal Cards was popular among participants, primarily utilized to break down high-level prompts into more manageable components. P_A 11 articulated this benefit, stating, "The goal card helped a lot with breaking down large prompts into specific parameters, and I have the opportunity to choose which ones I want to implement."

Some users employed Goal Cards as a task decomposition strategy, giving the system context for the entire task, not just the subtask stage. For instance, P_A 11 used a Goal Card to pose the question, "Show me a place people want to live in". Similarly, P_B 7 utilized a Goal Card with the label "A child story with a twisted

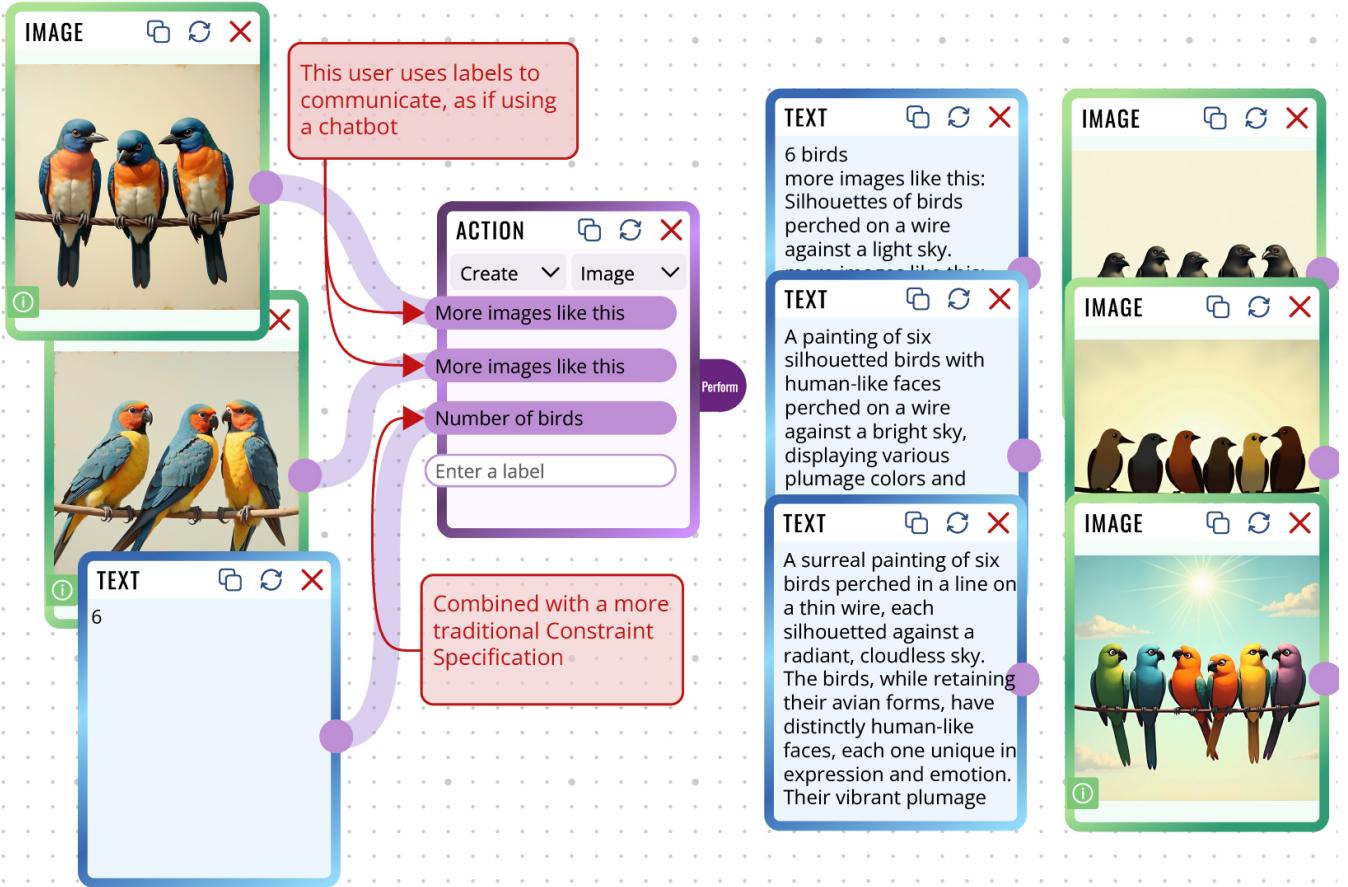


Figure 8: P_A 6 communicating the DeckFlow using ChatBot-like instructions in Hard Task B

Table 3: Label categories and input modalities attached to an Action Card each time a modality was generated.

Modality	Type of Label				Type of Input			
	Annotation	Constraint	Instruction	Empty	Text	Image	Audio	
Text	13	7	1	7	25	3	0	
Image	47	6	0	11	61	2	1	
Audio	17	3	1	6	20	6	1	

storyline, explained with three images and text description” for the Children’s Story task.

During the comparative study, many users ($n_A = 4$) expressed a desire to provide multimodal input to the Goal Card. However, after this feature was implemented in the subsequent study, participants consistently used only text as input when creating an Action Card.

5.3 RQ3: How do users approach the Generative Space Exploration problem in the interfaces?

5.3.1 *Similar performance in closed-ended tasks.* In the rating scale results from the Comparative Study (Figure 9), there was no clear favorite for closed-ended tasks; P_A 11 stated that “I prefer ChatFlow for [closed-ended tasks] because it’s more suited for sentence prompts”, but P_A 9 disagreed, saying “In the hard task, I definitely prefer DeckFlow because it gives me several options, and gives me an outline of what GPT expects”. In the survey, DeckFlow was more closely resembled the user’s target image, but were evenly satisfied with the final image.

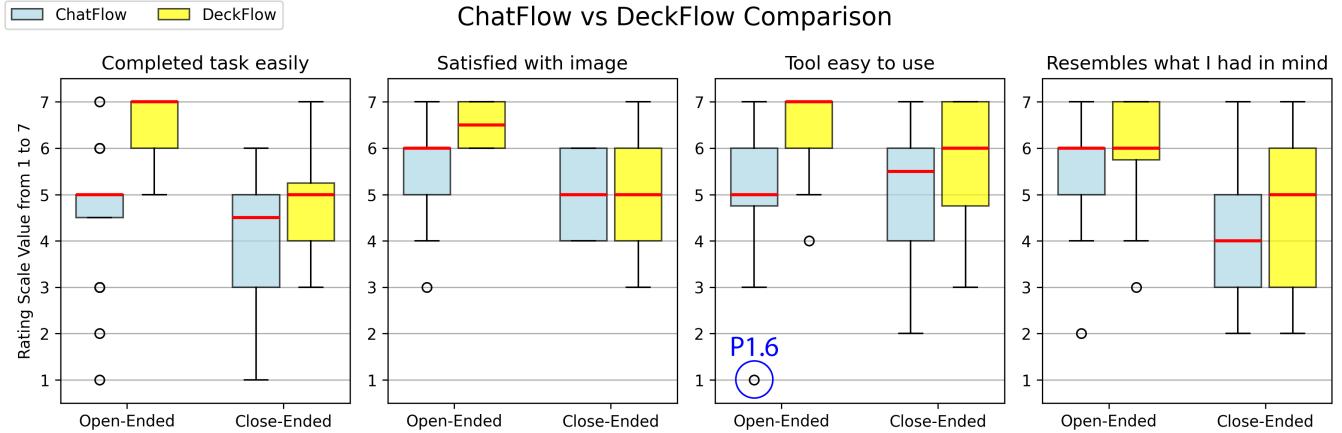


Figure 9: Rating scale results for post-task analysis in the Comparative Study (16 open-ended, 16 close-ended). P_A 6 rated ChatFlow very poorly in ease of use for the open-ended task.

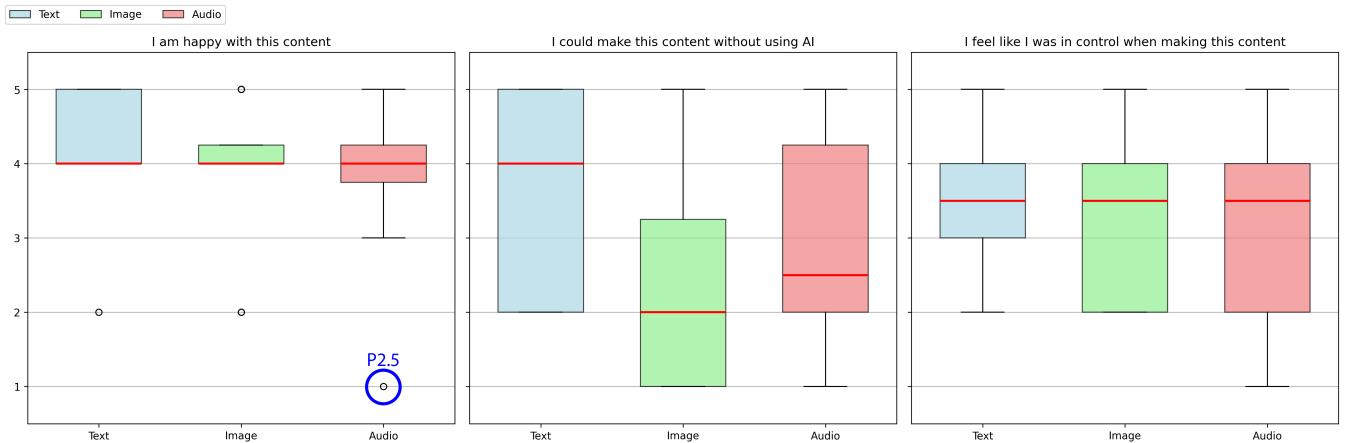


Figure 10: User evaluations of their outputs, (15 total) during the post-task interview. P_B 5 was unhappy with the generated audio content.

5.3.2 DeckFlow was universally preferred in open-ended tasks. As seen in Figure 9, participants found DeckFlow preferable in both outcome and usability in open-ended tasks, but similar in outcome. Notably, P_A 6 gave ChatFlow a 1 out of 7 when asked about ease of use for their open-ended task, stating “[ChatFlow] made more sense if I wanted to do one specific change to an image; something creative would be way easier in DeckFlow—you had such a wide array of images you got back so fast, and the prompt generation was really phenomenal. In ChatFlow, you had to type out the prompt, or ask the model to type it out”.

5.3.3 The role of divergence. The ability of an interface to diverge from past output was noted by many users ($n_A = 8, n_B = 4$). P_B 7 indicated that DeckFlow’s design supports this: “I think at least the prompt part—generating image or audio—the prompts are there. I don’t need to write them on my own. The creation process is a very slow process if I do it by myself. For DeckFlow, I see more possibilities more quickly. Although the image doesn’t live up to my

expectation, it helps me know what I want, because I know what I don’t want. Overall, I think it’s great for my creative process.” Because each of the generated rows utilize different prompting methods, some users ($n_A = 5$) picked up on these differences, especially the less adherent but more creative third row: “For the third row—it’s a higher quality picture, but it doesn’t really reflect the prompt “Also, there’s one thing that I have noticed is that since there are 3 roles every time, and I barely find any image from the last row relevant to what I want, although it is like a higher quality picture” (P_A 12).

Divergence was not always helpful, as stated by P_A 11 during their Close-Ended task: “There was more variety in these images, and I think that worked to the detriment to me trying to do the task. I was looking for something very specific, and it didn’t really quite hold.”

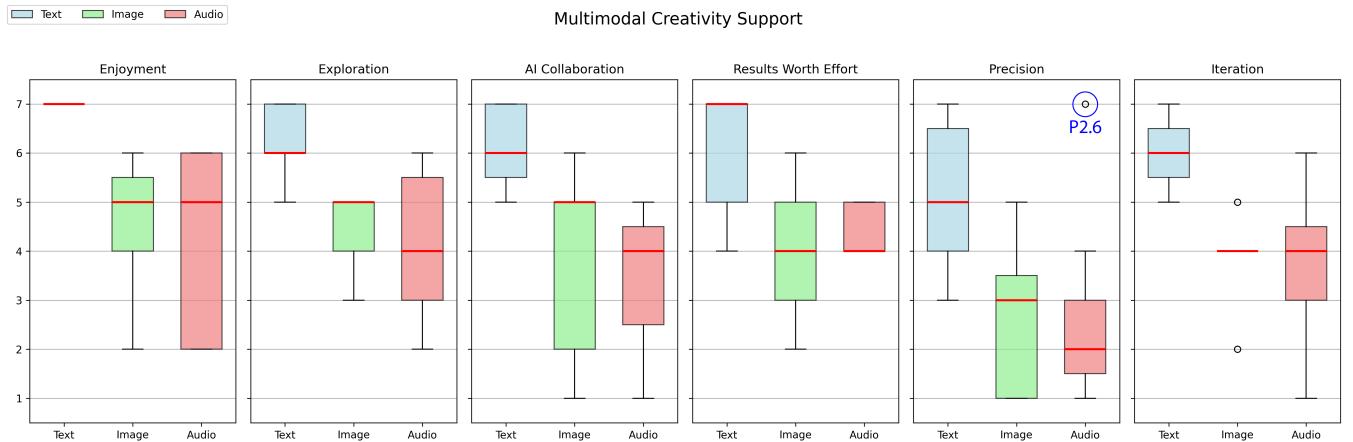


Figure 11: Adapted Creativity Support Index across modalities in the Multimodality Study $n_B = 7$.

5.4 RQ4: How are different modalities treated as input and output?

5.4.1 Text dominates as specification input. Users preferred using text to specify their intent, using it for 89.5% of their Action Card inputs, and rating text much higher than images or audio for each of the modified Creativity Support Index questions [27]: “Sometimes, I’m not sure to what extent do the audio or images affect the output, but I can always resort to text as the input, and it’s very clear to me” (P_B 3).

Some users ($n_B = 3$) mentioned that it felt like “Google Translate” or the game “telephone” due to the intermediate context always being revealed through text and the model’s gaps in modal understanding: “I think they all go through text; it reminds me of the old Google Translate, how if you translate to different languages, especially obscure ones—Haitian Creole for example—it would go through English, and you would see the English words that they couldn’t translate. *The image and audio cannot directly translate to each other, but they get described by text first, which is more efficient since you don’t need a separate model to communicate between that audio and the image.* At least ‘telephone game’ results were all pretty. It was positive for entertainment since you get slightly different things, but negative for trying to make something precise” (P_B 1).

One notable exception is P_B 6, who gave Audio as Input a 7 out of 7 when asked about its Precision, stating “It’s always a problem to get everything in my mind to the model. No matter with text, image, or audio—this is also a problem when I try to convey my thoughts to a real person... I can never express myself entirely—being able to express myself in different ways helped me. Like to say, if someone doesn’t know Picasso, I can at least show them an image.”

5.4.2 Text was the least interrogated output. Despite being the dominant choice for input, text only accounted for 16.8% of the output generated. Additionally, of the *final text outputs* in the Multimodality Study, 6/12 ended in incomplete sentences due to the model reaching its token limit, such as P_B 5’s chosen text output for the Creature task: “..., and small invertebrates. The creature often [sic]”.

5.4.3 Most generations were images. In the Multimodality Study, image generation accounted for 61.0% of all generations, consistently among each task. Participants began most tasks (8/15) with image generation, such as P_B 6, whose creative use of the Cluster was discussed in 5.1.2.

5.4.4 Strong reactions to Audio. Despite only averaging 2 bulk generations per task, some participants ($n_B = 3$) had strong reactions to generated audio. For example, as seen in Figure 10, P_B 5 was unhappy with their generated audio content which sounded vaguely like giggling and crackles: “That was nightmarish... I’m not gonna hold it against the model. I think if it were a better model, this would be closer to what I want.” P_B 7, when generating audio for the Creature task, indicated they were somewhat fearful of listening to the audio, worried that it would be “creepy” due to the image used as input. However, after one generation, they stated “*I think the sound is what makes a creature imaginative most.* The visual part, although it may look like a different creature than what exists on earth, maybe I can imagine it from the descriptions, but the audio, I totally cannot expect. In daily life, we learn a lot from what we see, but not a lot from what we hear. I think audio is the most important part to make this creature like an alien. The audio makes me feel like the creature exists.”

6 Discussion

6.1 Supporting Task Decomposition Through Interactivity

DeckFlow’s infinite canvas addresses the task decomposition problem by enabling users to organize work according to their preferences. This flexibility fostered diverse usage patterns (5.1.1), from top-down sequential to divide-and-conquer approaches, while encouraging exploration (5.3.3). Components like Clusters were adapted beyond their intended purpose for canvas organization (5.1.2), highlighting users’ need for structural management. Future interfaces could enhance task decomposition through attention heatmaps from PromptCharm [15], or spatial dimensions for parameter and prompt exploration from Automatic1111 [25] and Dreamsheets [21].

Widespread AI tools, however, lack many of the features required to support these interactions. ChatGPT [28] and Cursor⁹, for example, lack robust and glanceable incremental version control and usage flexibility. As features like the Model Context Protocol¹⁰ become more widely integrated and new modalities enable more complicated Human-AI co-creation, it is vital that interfaces that support flexibility and glanceability develop alongside these technical accomplishments.

6.2 Conversational Expectations

Though DeckFlow addressed specification decomposition through structured components, users frequently approached it conversationally, potentially revealing ingrained mental models from interfaces like ChatGPT. Users created instruction-style labels in Action Cards (5.2.2) rather than using the intended annotation types, and used Goal Cards as if they were beginning a conversation (5.2.4). Combined with the frustration of DeckFlow lacking a centralized memory model (5.1.3), it seems that user expectations of natural language have begun to include features found in Chat interfaces. Future interface designers must either accommodate these conversational patterns, or provide clearer scaffolding for alternative interaction models.

6.3 Multimodal Inputs and Generative Space Exploration

DeckFlow's approach to generative space exploration through multimodal inputs showed mixed results. While supporting iterative refinement (5.2), it sometimes failed to maintain consistency in direct iteration (5.3.1), and created disconnection between modalities (5.4.1). Users strongly preferred text for specification (89.5% of inputs) but spent most time with images (61% of generations), and showed strongest emotional responses to audio (5.4.4).

These findings suggests that modalities can serve different roles in generation: text for precise specification, images for quickly understood output, and audio for emotional engagement. Future work could enhance exploration by giving direct control, such as attention masking [15] or model adaptation [29, 30].

6.4 Threats to Validity

Our lab studies had several limitations: short tasks (10–15 min), a homogeneous participant pool (18–25 year-old CS/EE students), unfamiliar equipment, awareness of the researchers' roles, and asymmetric editing capabilities across modalities. The ChatGPT-like baseline represents just one possible comparison point in a rapidly evolving landscape. These factors limit generalizability to other tools, populations, and real-world creative workflows.

7 Conclusion

We have presented DeckFlow, a novel interface for iterative human-AI co-design in generative content creation. DeckFlow addresses the **task decomposition problem** through an infinite canvas for parallel subtasks, the **specification decomposition problem** via

Goal Cards that break into labeled Action Cards supporting multimodal inputs, and the **generative space exploration problem** by presenting structured output groups that represent different creative directions directly on the canvas.

Our evaluations demonstrated DeckFlow supports diverse workflows and improves outcomes for open-ended creative tasks. Users developed distinct patterns, had a strong preference for text-based specification despite multimodal capabilities. While our conversational baseline and DeckFlow performed similarly for closed-ended tasks, participants preferred DeckFlow for exploratory creation. The study revealed unexpected behaviors, including conversational expectations and strong emotional responses to audio generation.

As generative AI evolves, interfaces like DeckFlow will be crucial in empowering users while maintaining creative control. Future work could extend the specification design space, investigate long-term impacts on creative processes, and develop tools supporting diverse needs for content creation.

References

- [1] M. Zhou, V. Abhishek, T. Derdenger, J. Kim, and K. Srinivasan, "Bias in generative ai," 2024.
- [2] Blender Foundation, "Blender features," blender.org.
- [3] "Figma design features," [Figma](https://www.figma.com/design-tools/).
- [4] A. Bragdon, R. Zelezniak, S. P. Reiss, S. Karumuri, W. Cheung, J. Kaplan, C. Coleman, F. Adeputra, and J. J. LaViola Jr, "Code bubbles: a working set-based interface for code understanding and maintenance," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2503–2512, 2010.
- [5] M. Cherubini, G. Venolia, R. DeLine, and A. J. Ko, "Let's go to the whiteboard: how and why software developers use drawings," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, (New York, NY, USA), p. 557–566, Association for Computing Machinery, 2007.
- [6] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, P. S. Dulepet, S. Vidyadhara, D. Ki, S. Agrawal, C. Pham, G. Kroiz, F. Li, H. Tao, A. Srivastava, H. D. Costa, S. Gupta, M. L. Rogers, I. Gonçalves, G. Sarli, I. Galynker, D. Peskoff, M. Carpuat, J. White, S. Anadkat, A. Hoyle, and P. Resnik, "The prompt report: A systematic survey of prompting techniques," 2024.
- [7] S. Brade, B. Wang, M. Sousa, S. Oore, and T. Grossman, "Promptify: Text-to-image generation through interactive prompt exploration with large language models," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, (New York, NY, USA), Association for Computing Machinery, 2023.
- [8] Comfy Org, "Comfyui," 2024.
- [9] I. Arawjo, C. Swoopes, P. Vaithilingam, M. Wattberg, and E. Glassman, "Chainforge: A visual toolkit for prompt engineering and llm hypothesis testing," 2023.
- [10] S. Suh, B. Min, S. Palani, and H. Xia, "Sensecape: Enabling multilevel exploration and sensemaking with large language models," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, p. 1–18, ACM, Oct. 2023.
- [11] "Gemini Powers tldraw's "Natural Language Computing" Experience," [Google AI for Developers](https://ai.google.com/).
- [12] D. Choi, S. Hong, J. Park, J. J. Y. Chung, and J. Kim, "Creativeconnect: Supporting reference recombination for graphic design ideation with generative ai," *arXiv preprint arXiv:2312.11949*, 2023.
- [13] J. Fogarty, D. Tan, A. Kapoor, and S. Winder, "Cueflink: interactive concept learning in image search," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, (New York, NY, USA), p. 29–38, Association for Computing Machinery, 2008.
- [14] J. J. Y. Chung and E. Adar, "Promptpaint: Steering text-to-image generation through paint medium-like interactions," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, (New York, NY, USA), Association for Computing Machinery, 2023.
- [15] Z. Wang, Y. Huang, D. Song, L. Ma, and T. Zhang, "Promptcharm: Text-to-image generation through multi-modal prompting and refinement," *arXiv preprint arXiv:2403.04014*, 2024.
- [16] S. Suh, M. Chen, B. Min, T. J.-J. Li, and H. Xia, "Structured generation and exploration of design space with large language models for human-ai co-creation," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2024.

⁹Cursor. <https://www.cursor.com/en>

¹⁰Model Context Protocol. <https://modelcontextprotocol.io/introduction>

- [17] T. Zhang, L. Lowmanstone, X. Wang, and E. L. Glassman, "Interactive program synthesis by augmented examples," in *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, (New York, NY, USA), p. 627–648, Association for Computing Machinery, 2020.
- [18] S. MacNeil, J. Okerlund, and C. Latulipe, "Dimensional reasoning and research design spaces," in *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, C&C '17, (New York, NY, USA), p. 367–379, Association for Computing Machinery, 2017.
- [19] J. Marks, B. Andelman, P. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, and W. Ruml, "Design galleries: A general approach to setting parameters for computer graphics and animation," in *SIGGRAPH*, pp. 389–400, 1997.
- [20] P. O'Donovan, A. Agarwala, and A. Hertzmann, "Designscape: Design with interactive layout suggestions," in *CHI*, pp. 1221–1224, 2015.
- [21] S. G. Almeda, J. Zamfirescu-Pereira, K. W. Kim, P. Mani Rathnam, and B. Hartmann, "Prompting for discovery: Flexible sense-making for ai art-making with dreamsheets," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, (New York, NY, USA), Association for Computing Machinery, 2024.
- [22] Z. Evans, J. D. Parker, C. Carr, Z. Zukowski, J. Taylor, and J. Pons, "Stable audio open," 2024.
- [23] H. Dang, F. Brady, G. Fitzmaurice, and F. Anderson, "Worldsmith: Iterative and expressive prompting for world building with a generative ai," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–17, 2023.
- [24] K. Son, D. Choi, T. S. Kim, Y.-H. Kim, and J. Kim, "Genquery: Supporting expressive visual search with generative models," 2023.
- [25] AUTOMATIC1111, "Stable Diffusion Web UI," Aug. 2022.
- [26] N. Evirgen and X. A. Chen, "Ganzilla: User-driven direction discovery in generative adversarial networks," in *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST '22, (New York, NY, USA), Association for Computing Machinery, 2022.
- [27] E. Cherry and C. Latulipe, "Quantifying the creativity support of digital tools through the creativity support index," *ACM Trans. Comput.-Hum. Interact.*, vol. 21, jun 2014.
- [28] Nov. 2022.
- [29] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or, "An image is worth one word: Personalizing text-to-image generation using textual inversion," 2022.
- [30] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.