See it. Say it. Sorted: Agentic System for Compositional Diagram Generation

Hantao Zhang
Yale University
hantao.zhang@yale.edu

Jingyang Liu
Uniersiy of Edinburgh
jingyang__liu@outlook.com

Ed Li
Yale University
ed.li@yale.edu

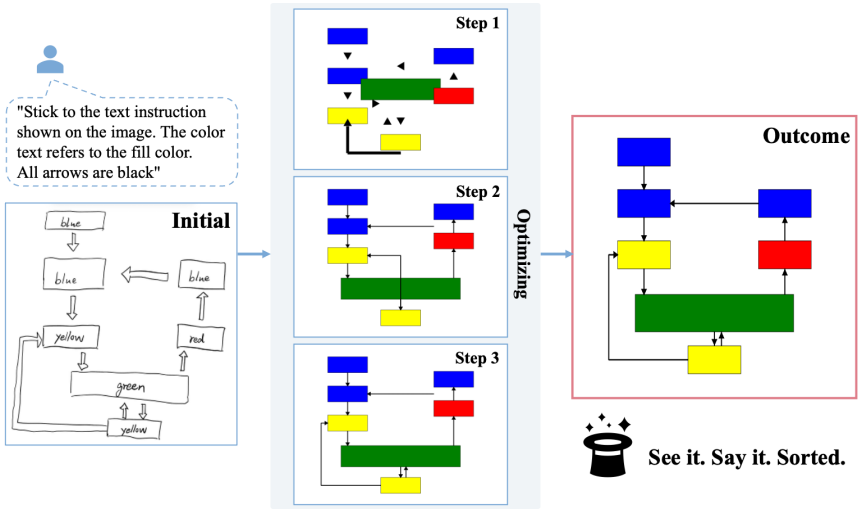


Figure 1: Given a sketch of a flow chart, our agent faithfully reconstructs the intended structure while strictly following the accompanying text instructions. The system operates in an iterative Critic–Candidates–Judge loop: a VLM critiques discrepancies between the sketch and the current diagram, multiple LLMs propose diverse SVG modifications, and a Judge VLM selects the best candidate. This training-free framework enables accurate, controllable, and editable diagram generation, moving beyond pixel-level synthesis toward structured programmatic outputs.

Abstract

We study *sketch-to-diagram* generation: converting rough hand sketches into precise, compositional diagrams. Diffusion models excel at photorealism but struggle with the spatial precision, alignment, and symbolic structure required for flowcharts. We introduce *See it. Say it. Sorted.*, a *training-free* agentic system that couples a Vision–Language Model (VLM) with Large Language Models (LLMs) to produce editable Scalable Vector Graphics (SVG) programs. The system runs an iterative loop in which a *Critic VLM* proposes a small set of qualitative, relational edits; multiple *candidate LLMs* synthesize SVG updates with diverse strategies (conservative→aggressive, alternative, focused); and a *Judge VLM* selects the best candidate, ensuring stable improvement. This design prioritizes qualitative reasoning over brittle numerical estimates, preserves global constraints (e.g., alignment, connectivity), and naturally supports human-in-the-loop corrections. On 10 sketches derived from flowcharts in published papers, our method more faithfully reconstructs layout and structure than

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Space in Vision, Language, Embodied AI.

two frontier closed-source image generation LLMs (GPT-5 and Gemini-2.5-Pro), accurately composing primitives (e.g., multi-headed arrows) without inserting unwanted text. Because outputs are programmatic SVGs, the approach is readily extensible to presentation tools (e.g., PowerPoint) via APIs and can be specialized with improved prompts and task-specific tools. The codebase is open-sourced at `https://github.com/hantaoZhangrichard/see_it_say_it_sorted.git`.

1 Introduction

Sketch-to-diagram generation transforms rough hand-drawn sketches into precise, compositional diagrams or flowcharts, akin to those made in tools like PowerPoint. This has broad applications in software engineering, education, and design, particularly in collaborative settings requiring rapid ideation and clear communication.

Diffusion-based generative models achieve high-fidelity image synthesis from text and sketches [9, 12, 17], but struggle with the compositional reasoning and spatial precision needed for structured diagrams. Their pixel-based diffusion process makes it difficult to maintain distinct shapes, lines, and alignments [10, 18], and they are not easily adaptable to varying diagram sizes or fine-grained user instructions.

Meanwhile, Vision–Language Models (VLMs) and Large Language Models (LLMs) have shown strong *2D spatial reasoning* [1, 14, 15]. Agentic VLM+LLM systems have been applied to visual-feedback tasks like 3D graphics editing [4, 6], part assembly [16], and scene generation [5], with the VLM acting as critic and the LLM generating code. Other works train transformers to produce graphics formats like Scalable Vector Graphics (SVG) and Constructive Solid Geometry (CSG) [2, 8, 13], though these lack the generalization of frontier VLMs.

We introduce a *training-free* VLM+LLM agentic system, *See it. Say it. Sorted.*, which interprets sketches and text instructions to produce editable SVG diagrams. Compared to diffusion methods, our approach offers:

- **Adaptability:** Handles varying sizes and complexities without retraining.
- **Human-in-the-loop:** Allows iterative refinement at any stage.
- **Interpretability:** Outputs object-based, modifiable SVG code.

These properties make our approach highly extensible to real-world graphics design environments, such as PowerPoint or Google Slides, by integrating with their APIs. As a proof-of-concept, our work demonstrates how an agentic system can serve as a foundation for compositional graphics design—moving beyond pixel-level synthesis toward structured, controllable, and user-centric workflows. The framework can be further specialized through refined prompt design and the integration of pre-built, task-specific tools, enabling tailored solutions for a wide range of application domains.

2 See it. Say it. Sorted.

We present our agentic system for sketch-to-diagram generation and evaluate its effectiveness on SVG graphics generation. We only equip our agent with minimal set of primitive shapes and color palette, details shown in Appendix A.2. Given this limitation, our agent is still able to faithfully reconstruct the sketch image. Our results are compared against other image generation methods using 10 sketches derived from flowcharts in real published papers.

Initial program. The *Critic VLM* examines the target sketch and describes it in terms of available primitives, including approximate positions, sizes, and colors. This qualitative description is passed to an LLM, which generates an initial SVG expression as the starting point for refinement.

See it. Say it. At each optimization step, the Critic VLM compares the target and current images, then: 1. Gives a high-level scene description; 2. Identifies 1–3 key discrepancies; 3. Suggests targeted modifications.

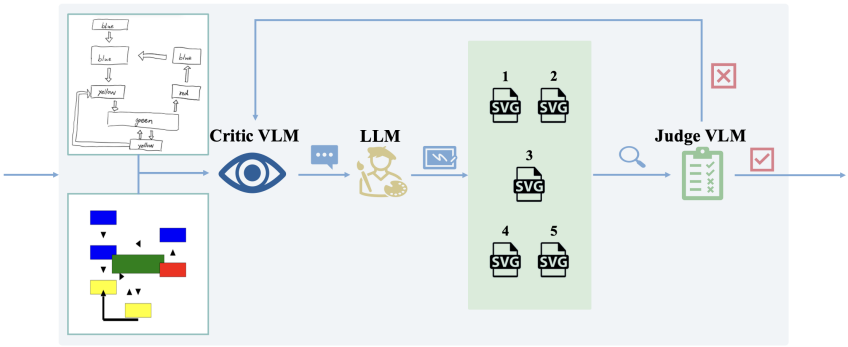


Figure 2: Pipeline of *See it. Say it. Sorted.* for one optimization step: The Critic VLM compares the target sketch and the current image and identifies few small modifications, passing to LLM. Given VLM's instructions, LLM generates several candidates that balances exploration-exploitation trade-off. These candidates are rendered and evaluated together with the current image by the Judge VLM. Judge VLM decides which one best reconstruct the sketch. If the current image is chosen, then the optimization is reverted and Critic VLM receives feedback of failed modifications. If one of the candidates is chosen then proceed to next optimization step.

This prompting strategy follows two design principles:

(i) Limiting modifications per step. Restricting the VLM to only a few changes per iteration helps stabilize the optimization process. Without this constraint, VLMs often produce large sets of suggestions, mixing correct and incorrect ones. Applying all changes at once can cause the generated image to oscillate between suboptimal states.

(ii) Emphasizing qualitative over quantitative descriptions. Rather than requesting exact coordinates or dimensions, we instruct the VLM to express relative spatial and size relationships, referencing other primitives or the canvas as anchors (e.g., *"The width should be about half the canvas; the size should be doubled; the blue rectangle should just touch the red circle on its left."*). This choice is motivated by:

- VLMs are more reliable at describing qualitative relations than at estimating precise numerical values [3].
- The generated image evolves over time, with primitives moving, being added, or removed, making a fixed external ID mapping unreliable.
- Qualitative references provide richer contextual cues, enabling the LLM to identify the intended primitive more accurately.
- This approach gives the LLM greater flexibility in exploring the solution space, avoiding overly rigid adherence to potentially inaccurate quantitative instructions.

Sorted. Given the VLM's feedback, multiple LLMs generate candidate SVGs in JSON format following the grammar specified in A.2 using distinct strategies (e.g., conservative, moderate, aggressive, alternative, focused) to balance exploration and exploitation. LLMs are instructed to respect the VLM's general description to maintain global constraints such as alignment and connectivity. These candidates are then rendered into images, passing to the Judge VLM.

The *Judge VLM* evaluates the candidates and current image, selecting the one that best matches the sketch. If no candidate improves the result, the step is reverted and the Critic VLM revises its suggestions. This loop of critique, diverse modification, and selection enables stable convergence toward the target diagram.

3 Experiment Results

We evaluated our agent on 10 sketch images inspired by flowcharts from real published papers. Each sketch was accompanied by a single text instruction: *"Stick to the text instruction shown on the image.*

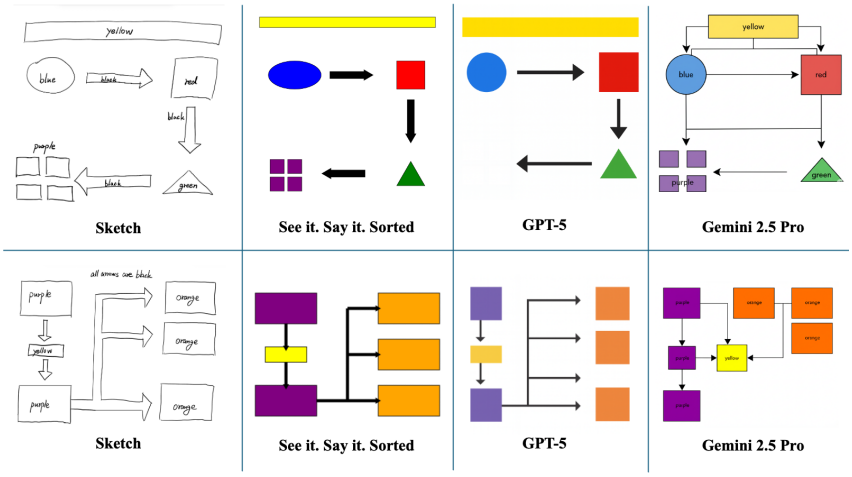


Figure 3: Our agent faithfully generates flow charts based on the sketch within 3 optimization steps, significantly outperforming GPT-5 and Gemini-2.5-Pro at preserving the structure and characteristics of the diagram. For the complete comparison for all 10 tasks, see Appendix 1.

The color text refers to the fill color. Do not include any text in the final diagram.". In all experiments, we used Gemini-2.5-Pro as Critic VLM and Judge VLM and Gemini-2.5-Flash as the LLM.

Figure 3 compares our system against two frontier closed-source image generation models: GPT-5 and Gemini-2.5-Pro. Our agent consistently reconstructs the structure depicted in the sketch, maintaining precise horizontal and vertical alignments between blocks. Notably, it can use fundamental primitives (e.g., rectangles, triangles) to compose arrows, including complex multi-headed arrows with correct orientations.

In contrast, GPT-5 often misses key structural details—for example, omitting an entire cluster of purple blocks (top-row example) or producing an incorrect number of arrowheads (bottom-row example). Gemini-2.5-Pro exhibits more severe failures, frequently generating layouts that diverge from the sketch. Even when explicitly instructed that the text in the sketch only indicates fill colors and should not appear in the final diagram, Gemini repeatedly inserts text into blocks.

4 Discussion

We presented *See it. Say it. Sorted.*, a *training-free* VLM+LLM agentic system that transforms hand-drawn sketches into structured, editable SVG diagrams through an iterative Critic–Candidates–Judge loop. By emphasizing qualitative, relational feedback rather than brittle numerical estimates, the method preserves global structural constraints such as alignment and connectivity while enabling precise local refinements, outperforming frontier image LLMs in faithfully reconstructing compositional layouts. Because outputs are programmatic, the system is inherently extensible to real-world graphics design environments such as PowerPoint or Google Slides via APIs, and can incorporate additional computer graphics utilities—e.g., auto-layout, snapping, arrow libraries, style transfer—to expand aesthetic control while maintaining semantic correctness. As VLMs and LLMs continue to advance, the same architecture will naturally improve without retraining, offering increasing accuracy and efficiency. In current experiments, the primary bottleneck lies in the Critic VLM, which occasionally produces imprecise or inappropriate suggestions that can lead to worse candidates; this is mitigated by the Judge VLM, which filters regressions and ensures optimization stability. Beyond this application, VLM-based judging [19] represents a promising approach for broader agentic systems and reinforcement learning in visual domains, but systematic evaluations of judge reliability—covering calibration, robustness, and domain generalization—are needed. Looking ahead, the qualitative-critique $\rightarrow$ multi-strategy synthesis $\rightarrow$ judging loop underlying our method could extend naturally to 3D tasks such as part assembly, scene layout [11], and CAD editing [7], leveraging recent advances in 3D spatial reasoning.

References

[1] Mu Cai, Zeyi Huang, Yuheng Li, Utkarsh Ojha, Haohan Wang, and Yong Lee. An investigation on llms' visual understanding ability using svg for image-text bridging. URL https://openaccess.thecvf.com/content/WACV2025/papers/Cai_An_Investigation_on_LLMs_Visual_Understanding_Ability_using_SVG_for_WACV_2025_paper.pdf.

[2] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation, 2020. URL https://arxiv.org/abs/2007.11301.

[3] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities, 2024. URL https://arxiv.org/abs/2401.12168.

[4] Yunqi Gu, Ian Huang, Jihyeon Je, Guandao Yang, and Leonidas Guibas. Blendergym: Benchmarking foundational model systems for graphics editing, 2025. URL https://arxiv.org/abs/2504.01786.

[5] Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scene as blender code, 2024. URL https://arxiv.org/abs/2403.01248.

[6] Ian Huang, Guandao Yang, and Leonidas Guibas. Blenderalchemy: Editing 3d graphics with vision-language models, 2024. URL https://arxiv.org/abs/2404.17672.

[7] Benjamin Jones, Dalton Hildreth, Duowen Chen, Ilya Baran, Vladimir G. Kim, and Adriana Schulz. Automate. *ACM Transactions on Graphics*, 40:1–18, 12 2021. doi: 10.1145/3478513.3480562.

[8] Shreyas Kapur, Erik Jenner, and Stuart Russell. Diffusion on syntax trees for program synthesis, 2024. URL https://arxiv.org/abs/2405.20519.

[9] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. URL https://openaccess.thecvf.com/content/CVPR2023/papers/Kawar_Imagic_Text-Based_Real_Image_Editing_With_Diffusion_Models_CVPR_2023_paper.pdf.

[10] Subhadeep Koley, Bhunia Ayan Kumar, Aneeshan Sain, Pinaki Nath Chowdhury, Tao Xiang, and Yi-Zhe Song. Picture that sketch: Photorealistic image generation from abstract sketches, 2023. URL https://arxiv.org/abs/2303.11162.

[11] Parker Liu, Chenxin Li, Zhengxin Li, Yipeng Wu, Wuyang Li, Zhiqin Yang, Zhenyuan Zhang, Yunlong Lin, Sirui Han, and Brandon Y Feng. Ir3d-bench: Evaluating vision-language model scene understanding as agentic inverse rendering, 2025. URL https://arxiv.org/abs/2506.23329.

[12] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv:2302.08453 [cs]*, 03 2023. URL https://arxiv.org/abs/2302.08453.

[13] Ronghuan Wu, Wanchao Su, and Jing Liao. Chat2svg: Vector graphics generation with large language models and image diffusion models. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Wu_Chat2SVG_Vector_Graphics_Generation_with_Large_Language_Models_and_Image_CVPR_2025_paper.pdf.

[14] Wenshan Wu, Shaoguang Mao, Yadong Zhang, Yan Xia, Li Dong, Lei Cui, and Furu Wei. Visualization-of-thought elicits spatial reasoning in large language models, 04 2024. URL https://arxiv.org/abs/2404.03622.

[15] Yutaro Yamada, Yihan Bao, Andrew K Lampinen, Jungo Kasai, and Ilker Yildirim. Evaluating spatial understanding of large language models, 2023. URL https://arxiv.org/abs/2310.14540.

[16] Yutaro Yamada, Khyathi Chandu, Yuchen Lin, Jack Hessel, Ilker Yildirim, and Yejin Choi. L3go: Language agents with chain-of-3d-thoughts for generating unconventional objects, 2024. URL https://arxiv.org/abs/2402.09052.

[17] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 02 2023. URL https://arxiv.org/abs/2302.05543.

[18] Tianyu Zhang, Xiaoxuan Xie, Xusheng Du, and Haoran Xie. Sketch-guided scene image generation, 2024. URL https://arxiv.org/abs/2407.06469.

[19] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 07 2023. URL https://arxiv.org/abs/2306.05685.

A Appendix

A.1 Comparison with Frontier Models

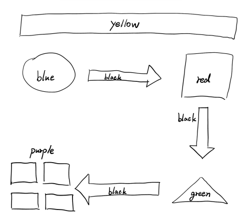
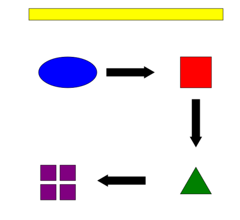
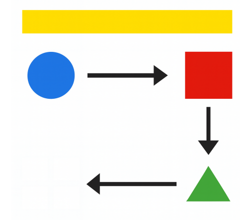
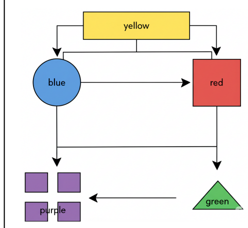
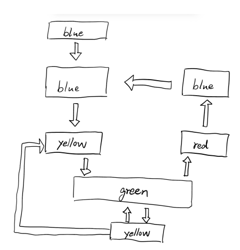
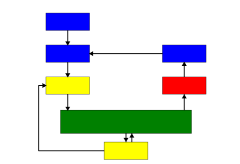
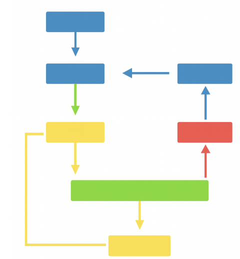
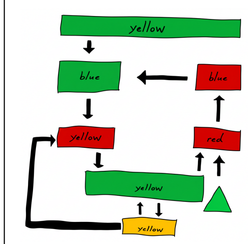
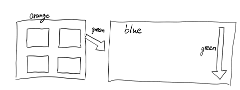
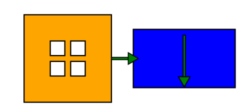
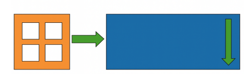
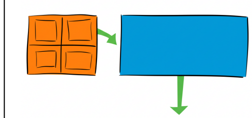
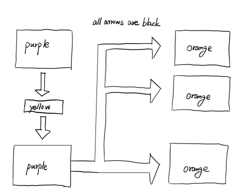
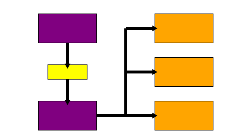
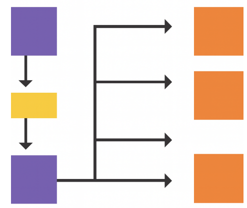
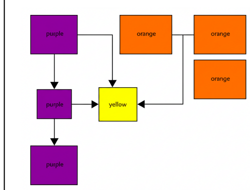
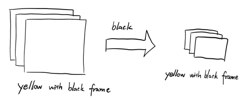
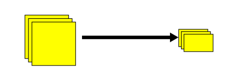
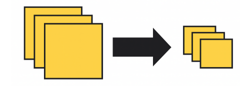
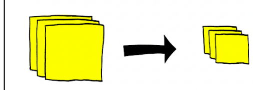
Sketch	See it. Say it. Sorted.	GPT-5	Gemini-2.5-Pro
			
			
			
			
			

Table 1: Task 1-5

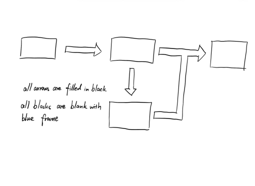
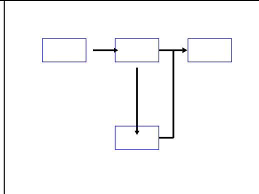
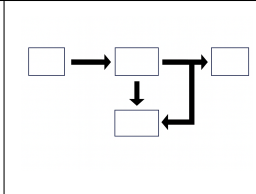
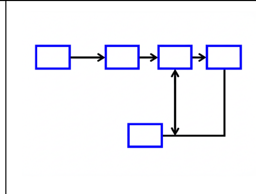
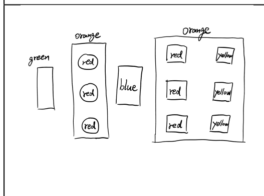
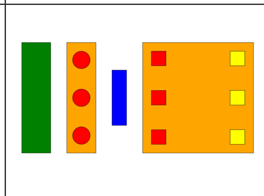
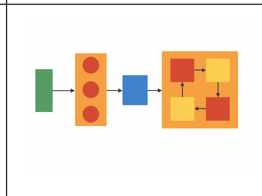
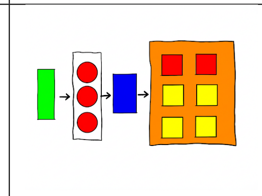
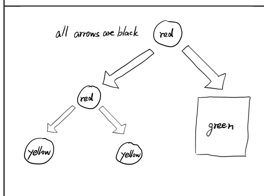
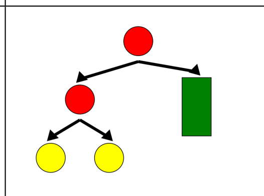
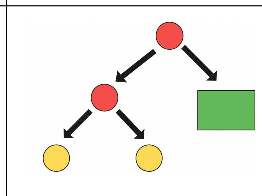
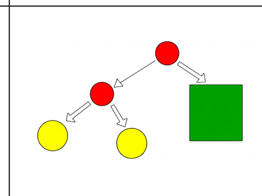
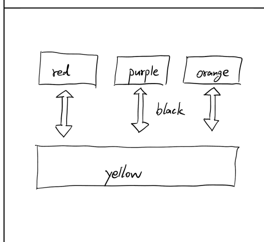
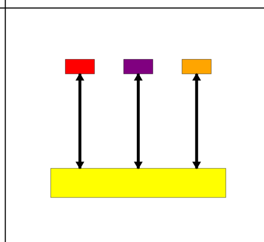
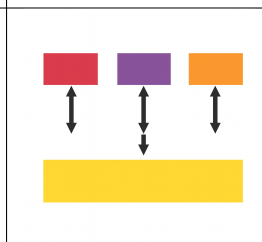
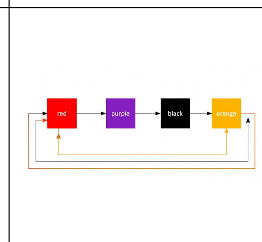
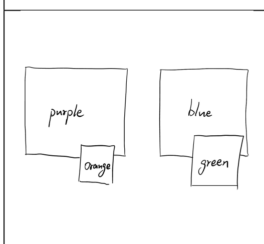
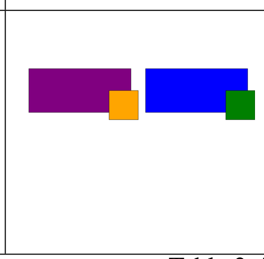
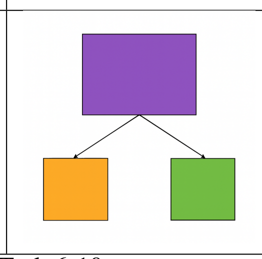
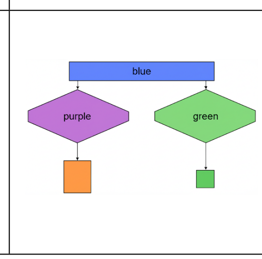
Sketch	See it. Say it. Sorted.	GPT-5	Gemini-2.5-Pro
			
			
			
			
			

Table 2: Task 6-10

A.2 SVG Grammar

Listing 1: SVG grammar

```
Canvas
  size: {canvas_width} x {canvas_height}   # pixels (width x height)
  coords: origin (0,0) top-left; +x right; +y down
  positioning: (x,y) = center of shape

Types
  <type> := "circle" | "rectangle" | "ellipse" | "triangle"

Colors (lowercase only)
  <color> := "red"|"green"|"blue"|"yellow"|"purple"|"orange"|"black"|"white"|"none"
  # "none" disables fill or stroke respectively

Shape Object
  {
    "shape_type":   <type>,                              # required
    "x":            <num>=0,  "y": <num>=0,              # center position (px)
    "scale_x":      <num>=1,  "scale_y": <num>=1,        # width/height; for circle/
    ellipse: diameters
    "fill_color":   <color>="none",                      # fill color (use "none" for
    no fill)
    "stroke_color": <color>="black",                     # stroke color (use "none" for
     no stroke)
    "stroke_width": <num>=1,                             # stroke thickness (px)
    "rotation":     <num>=0                              # degrees, clockwise; triangle
     0 degree points up
  }

Output (JSON)
  {
    "shapes": [ <Shape Object>, <Shape Object>, ... ]
  }
```