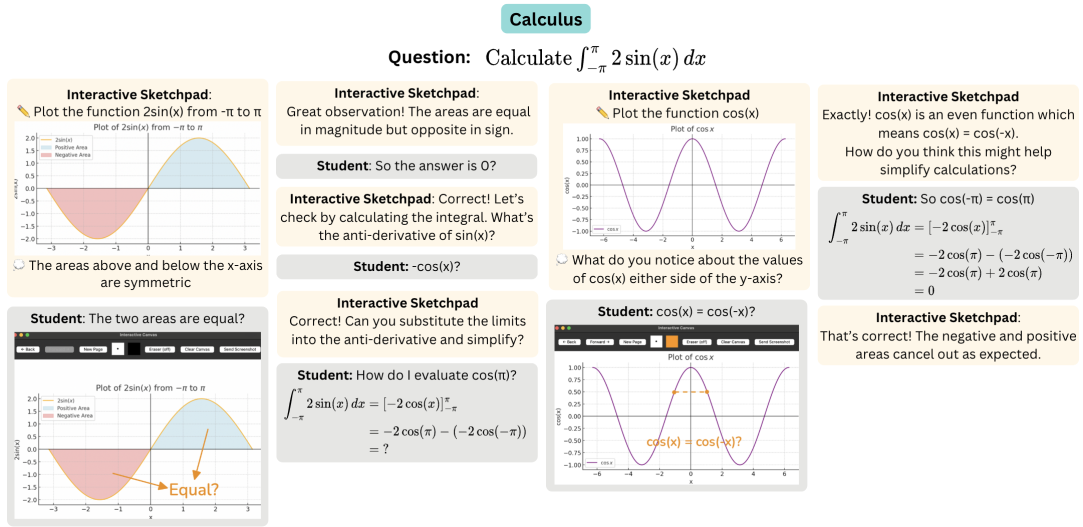
Interactive Sketchpad: An Interactive Multimodal System for Collaborative, Visual Problem-Solving

Jimin Lee[*]
jimin24@mit.edu
Massachusetts Institute of Technology
Cambridge, MA, USA

Steven-Shine Chen[*]
stevensh@mit.edu
Massachusetts Institute of Technology
Cambridge, MA, USA

Paul Pu Liang
ppliang@mit.edu
Massachusetts Institute of Technology
Cambridge, MA, USA



Question: Calculate $\int_{-\pi}^{\pi} 2\sin(x)\,dx$

Interactive Sketchpad:
✏️ Plot the function 2sin(x) from -π to π

💭 The areas above and below the x-axis are symmetric

Student: The two areas are equal?

Interactive Sketchpad:
Great observation! The areas are equal in magnitude but opposite in sign.

Student: So the answer is 0?

Interactive Sketchpad: Correct! Let's check by calculating the integral. What's the anti-derivative of sin(x)?

Student: -cos(x)?

Interactive Sketchpad
Correct! Can you substitute the limits into the anti-derivative and simplify?

Student: How do I evaluate cos(π)?

$$\int_{-\pi}^{\pi} 2\sin(x)\,dx = [-2\cos(x)]_{-\pi}^{\pi}$$
$$= -2\cos(\pi) - (-2\cos(-\pi))$$
$$= ?$$

Interactive Sketchpad
✏️ Plot the function cos(x)

💭 What do you notice about the values of cos(x) either side of the y-axis?

Student: cos(x) = cos(-x)?

Interactive Sketchpad
Exactly! cos(x) is an even function which means cos(x) = cos(-x).
How do you think this might help simplify calculations?

Student: So cos(-π) = cos(π)

$$\int_{-\pi}^{\pi} 2\sin(x)\,dx = [-2\cos(x)]_{-\pi}^{\pi}$$
$$= -2\cos(\pi) - (-2\cos(-\pi))$$
$$= -2\cos(\pi) + 2\cos(\pi)$$
$$= 0$$

Interactive Sketchpad:
That's correct! The negative and positive areas cancel out as expected.

Figure 1: Interactive Sketchpad enhances GPT-4o's ability to provide step-by-step, visual hints for problem-solving. Given a student query and problem statement, Interactive Sketchpad generates both textual hints and dynamic visual diagrams, allowing students to engage with the problem iteratively. Without Interactive Sketchpad, GPT-4o struggles to offer effective interactive guidance, frequently revealing the answer and not providing any visual aids, whereas Interactive Sketchpad enables a natural, multimodal learning experience that improves conceptual understanding.

Abstract

Humans have long relied on visual aids like sketches and diagrams to support reasoning and problem-solving. Visual tools, like auxiliary lines in geometry or graphs in calculus, are essential for understanding complex ideas. However, many tutoring systems remain text-based, providing feedback only through natural language. Leveraging recent advances in Large Multimodal Models (LMMs), this paper introduces Interactive Sketchpad, a tutoring system that combines language-based explanations with interactive visualizations to enhance learning. Built on a pre-trained LMM, Interactive Sketchpad is fine-tuned to provide step-by-step guidance in both text and visuals, enabling natural multimodal interaction with the student. Accurate and robust diagrams are generated by incorporating code execution into the reasoning process. User studies conducted on math problems such as geometry, calculus, and trigonometry demonstrate that Interactive Sketchpad leads to improved task comprehension, problem-solving accuracy, and engagement levels, highlighting its potential for transforming educational technologies. All code is available at: https://stevenshinechen.github.io/interactivesketchpad/.

CCS Concepts

• **Human-centered computing → Human computer interaction (HCI)**; **Interaction techniques**; • **Computing methodologies → Machine learning**; **Artificial intelligence**.

Keywords

AI for education, multimodal interaction, vision-language models

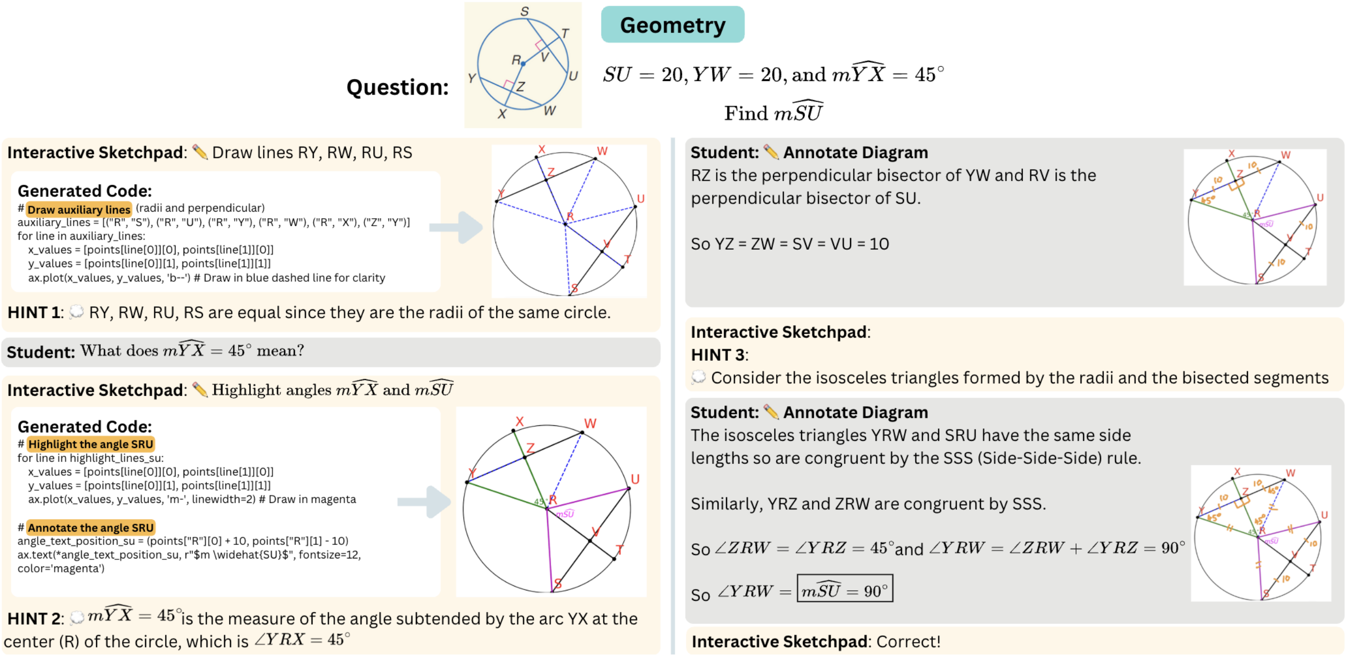
ACM Reference Format:
Jimin Lee, Steven-Shine Chen, and Paul Pu Liang. 2025. Interactive Sketchpad: An Interactive Multimodal System for Collaborative, Visual Problem-Solving. In . ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

---

[*] Equal Contribution.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, Washington, DC, USA
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX



Geometry

Question: $SU = 20, YW = 20,$ and $m\widehat{YX} = 45^\circ$

Find $m\widehat{SU}$

Interactive Sketchpad: ✏️ Draw lines RY, RW, RU, RS

Generated Code:

```
# Draw auxiliary lines (radii and perpendicular)
auxiliary_lines = [("R", "S"), ("R", "U"), ("R", "Y"), ("R", "W"), ("R", "X"), ("Z", "Y")]
for line in auxiliary_lines:
   x_values = [points[line[0]][0], points[line[1]][0]]
   y_values = [points[line[0]][1], points[line[1]][1]]
   ax.plot(x_values, y_values, 'b--') # Draw in blue dashed line for clarity
```

HINT 1: 💭 RY, RW, RU, RS are equal since they are the radii of the same circle.

Student: What does $m\widehat{YX} = 45^\circ$ mean?

Interactive Sketchpad: ✏️ Highlight angles $m\widehat{YX}$ and $m\widehat{SU}$

Generated Code:

```
# Highlight the angle SRU
for line in highlight_lines_su:
   x_values = [points[line[0]][0], points[line[1]][0]]
   y_values = [points[line[0]][1], points[line[1]][1]]
   ax.plot(x_values, y_values, 'm-', linewidth=2) # Draw in magenta

# Annotate the angle SRU
angle_text_position_su = (points["R"][0] + 10, points["R"][1] - 10)
ax.text(*angle_text_position_su, r"$m \widehat{SU}$", fontsize=12,
color='magenta')
```

HINT 2: 💭 $m\widehat{YX} = 45^\circ$ is the measure of the angle subtended by the arc YX at the center (R) of the circle, which is $\angle YRX = 45^\circ$

Student: ✏️ **Annotate Diagram**

RZ is the perpendicular bisector of YW and RV is the perpendicular bisector of SU.

So YZ = ZW = SV = VU = 10

Interactive Sketchpad:

HINT 3:

💭 Consider the isosceles triangles formed by the radii and the bisected segments

Student: ✏️ **Annotate Diagram**

The isosceles triangles YRW and SRU have the same side lengths so are congruent by the SSS (Side-Side-Side) rule.

Similarly, YRZ and ZRW are congruent by SSS.

So $\angle ZRW = \angle YRZ = 45^\circ$ and $\angle YRW = \angle ZRW + \angle YRZ = 90^\circ$

So $\angle YRW = \boxed{m\widehat{SU} = 90^\circ}$

Interactive Sketchpad: Correct!

Figure 2: Overview of INTERACTIVE SKETCHPAD: Given a multimodal question, INTERACTIVE SKETCHPAD generates a program to create a visual aid, then uses the visual aid as part of a hint to help the user solve the problem. The visual aid is sent to the interactive whiteboard which the user can write and draw on before sending the annotated diagram back to receive feedback or further help.

1 Introduction

Intelligent tutoring systems are revolutionizing education by democratizing access to personalized and effective learning experiences [32]. However, most educational tutoring systems are text-based, providing feedback only through natural language and limited visual support [33]. On the other hand, humans have long relied on visual aids like sketches and diagrams to support reasoning and problem-solving. Whether drawing auxiliary lines in geometry, diagrams for graph theory, or visualizing abstract concepts in math, visual tools are essential for breaking down and understanding complex ideas [4, 22, 26]. Studies show that students who engage with visual aids grasp ideas more effectively than those who rely solely on textual explanations [26]. Language-only systems are therefore unintuitive and often fall short when addressing tasks that require visual and spatial reasoning.

Today's large multimodal models (LMMs) provide a new opportunity to generate diagrams and sketches for tutoring systems automatically [10, 15, 20, 40]. Our paper prototypes such a system, called INTERACTIVE SKETCHPAD, to enable multimodal input and output interaction for problem-solving (see Figure 1). INTERACTIVE SKETCHPAD is built upon a pre-trained LMM and fine-tuned to provide step-by-step hints and explanations in both language and visual diagrams to tutor students. INTERACTIVE SKETCHPAD creates diagrams accurately and robustly by generating Python programs to output diagrams when executed. The diagram, along with corresponding textual hints, is sent to the student on their 'sketchpad' to be freely annotated or sketched on. This multimodal human-AI interaction framework enables the student to collaborate back and forth with the LMM on a shared interactive whiteboard, creating a natural and seamless interaction loop where both the human and the LMM can share text, images, and annotations.

We conduct several case studies with university students on a range of math tasks (including geometry and calculus), and find that INTERACTIVE SKETCHPAD leads to improved task comprehension, problem-solving accuracy, and engagement levels among students. Unlike traditional systems, this approach emphasizes a hands-on, learner-centered methodology that adapts to individual needs while leveraging the benefits of vision and language reasoning [26]. In addition to addressing the immediate gaps in educational technology, this research also contributes to the broader field of AI by exploring how multimodal systems can enhance human-computer interaction, leading to better AI systems that complement human expertise.

2 Related Work

Benefits of visualizations in learning. Visualizations play a crucial role in mathematics education, helping students understand abstract concepts, solve problems, and improve memory retention [1, 25], especially in visually oriented topics such as geometry and calculus [39]. Cognitive science research highlights that visualization can also reduce cognitive load by presenting data in a more accessible format [31] such as diagrams and images. Tools like graphing calculators and interactive whiteboards [18] have further demonstrated the benefits of incorporating visual elements

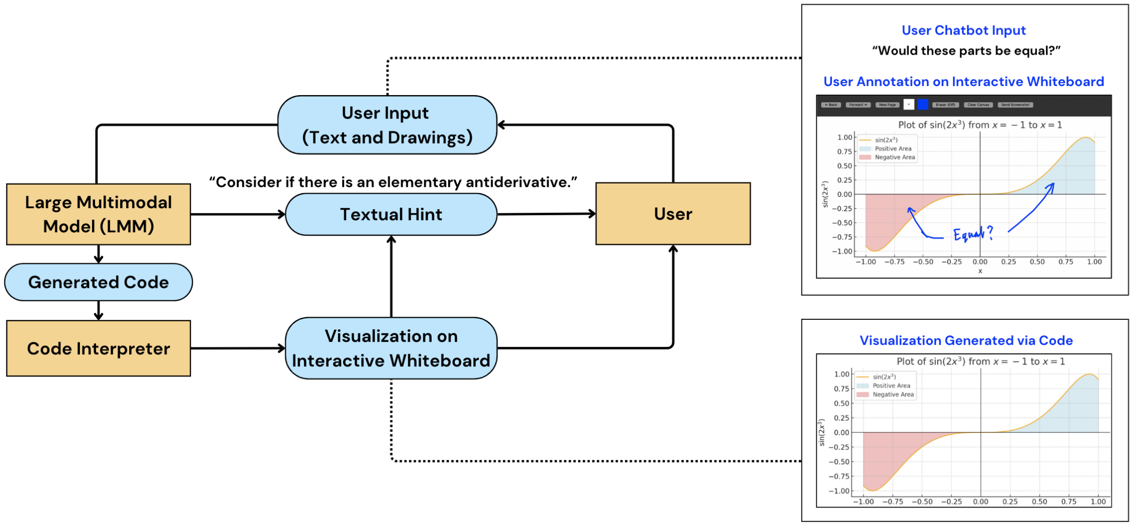

User Chatbot Input
"Would these parts be equal?"
User Annotation on Interactive Whiteboard
Plot of $\sin(2x^3)$ from $x = -1$ to $x = 1$
Equal?
User Input
(Text and Drawings)
"Consider if there is an elementary antiderivative."
Large Multimodal Model (LMM)
Textual Hint
User
Generated Code
Code Interpreter
Visualization on Interactive Whiteboard
Visualization Generated via Code
Plot of $\sin(2x^3)$ from $x = -1$ to $x = 1$

Figure 3: This flowchart demonstrates the interaction between user inputs, large multimodal models, and visualization components. The system processes user inputs, generates textual hints, interprets and executes code, and provides visualizations to guide problem-solving. The user can also annotate visualizations using the interactive whiteboard component.

into learning and have worked towards bridging the gap between theoretical constructs and practical understanding.

Multimodal interaction and visualizations. With the advancement of LMMs, researchers have explored the use of code generation to solve visual reasoning tasks. VisProg [9] generates pseudocode interpreted as a 'visual program', enabling modular, compositional visual reasoning using in-context learning. ViperGPT [30] uses a similar approach but generates Python code instead of a domain-specific language, making it more flexible. Visual Sketchpad [10] creates programs that generate intermediate diagrams like drawing auxiliary lines in geometry which are used in a visual chain of thought to enhance performance on a range of mathematical and computer vision tasks. Recent work has also focused on improving models' reasoning capabilities. Chain-of-Thought (CoT) [35], enhances large language models' reasoning abilities by guiding them to generate intermediate reasoning steps before arriving at a final answer. This concept has been generalized to incorporate vision with visual chain-of-thought reasoning, which involves step-by-step reasoning across modalities [23, 27]. However, these approaches are designed to facilitate LMM visual reasoning rather than interacting with and helping humans in education.

AI for education. Recent advances in NLP and LMMs have demonstrated significant potential in automating education [8, 12], with the ability to help students in a wide range of subjects including math [21, 29], programming [2, 11, 38] and language [36, 37], and in assisting humans with tutoring [34]. However, while LMMs can serve as effective educational tools, excessive reliance on them for direct answer generation has been shown to negatively impact student learning outcomes [3, 19]. This issue can be mitigated by moderating LMM responses to provide hints rather than complete answers [13], thereby encouraging active learning. Furthermore, incorporating both textual and visual representations of student work, rather than relying on text alone, has been found to enable more effective feedback mechanisms [14].

3 INTERACTIVE SKETCHPAD

INTERACTIVE SKETCHPAD enhances problem-solving and tutoring by incorporating a multimodal interaction loop, allowing humans to interact with LMMs using both text and images. We show the overall interactive usage in Figure 2 and its detailed architecture in Figure 3. The key components of INTERACTIVE SKETCHPAD include:

1. **Problem analysis**: The system evaluates the problem to determine whether a visualization would be beneficial.
2. **Visual generation**: If a visualization is deemed helpful, the system generates a Python program to create an appropriate visualization.
3. **Hint generation**: Then a textual hint is generated which references the visualization if one was created.
4. **Interactive whiteboard**: The hint and visualization appear in the chatbot interface as usual. However, if a visualization is generated, it is also displayed on the user's interactive whiteboard, allowing direct annotation and interaction. The annotated visualization along with additional text or images can be sent directly from the whiteboard back to the LMM which processes the new input and continues the iterative reasoning cycle.

Together, this loop fosters continuous human-AI collaboration, where the model provides interactive assistance in both language and vision to enhance concept understanding. We now explain each of these four steps in detail.

1) Problem Analysis. We prompt a pre-trained GPT-4o model [20] regarding when visualizations may be helpful based on the problem and user query (see examples in Appendix **??**).

2) Visualization Generation. To draw accurate diagrams, we generate Python programs using the LMM which, when executed,

Model	Maxflow	Isomorphism	Connectivity	Convexity	Parity
GPT-4o [20]	25.0	50.8	96.1	87.2	84.4
Visual Sketchpad [10]	66.3	65.3	98.4	94.9	94.7
INTERACTIVE SKETCHPAD (ours)	100.0	75.0	99.2	96.5	95.6
Improvement	**+33.7**	**+9.7**	**+0.8**	**+1.6**	**+0.9**

Table 1: Accuracy scores on graph algorithms and mathematical functions. INTERACTIVE SKETCHPAD outperforms Visual Sketchpad and other large multimodal model baselines by using code execution for calculations to solve tasks.

render the visualizations, similar to recent works such as Visual Sketchpad [10], VisProg [9] and ViperGPT [30]. For the LMM, we use GPT-4o [20] for multimodal reasoning and use the OpenAI Code Interpreter tool to run and execute the code. To draw the visualizations, we generate Python code that uses common Python libraries, primarily `matplotlib`, for plotting. The Code Interpreter then executes the code, and if it fails to run, the LMM will iterate on the code by attempting to run different code until it successfully generates a runnable program. Once the program is successfully executed, the resulting image file is displayed to the user.

3) Hint Generation. We prompt a GPT-4o model to give subtle hints rather than revealing the answer and to generate visualizations for the hint using code. To enable good hints and appropriate visual hints, we collected human demonstrations of good hints interleaved with images in an example conversation between a student and a tutor, using several samples of math and coding problems. We then fine-tuned the model with this data (prompt found in Appendix **??**). In subsequent iterations, the model uses the conversation history, a screenshot of the user's interactive whiteboard, and the user's query to generate hints. Finally, we also provide the model access to the OpenAI Code Interpreter to perform mathematical calculations by executing code to reduce calculation errors, similar to Chen et al. [5]. For example, the model will perform calculations such as numerical integration for calculus questions, or arithmetic for systems of equations to provide more accurate hints and check student answers (see examples in Appendix C).

4) Interactive Whiteboard. The interactive whiteboard component lets users intuitively collaborate with the LMM using textual and visual information. Users can take snapshots of their whiteboard to ask questions with their sketches, these snapshots are then sent to the chatbot server. The LMM would automatically be prompted with the screenshot and additional text or images provided by the user. Once a new visualization is generated, it is sent to the interactive whiteboard server, which automatically presents it to the user to continue drawing and interacting with the system.

4 Experiments

4.1 Research Questions

We aim to answer the following research questions through our experiments.

- **RQ0:** How accurate is INTERACTIVE SKETCHPAD at solving the problems of interest?
- **RQ1:** How effective is the proposed interactive system in improving users' understanding of concepts compared to text-only language model systems?
- **RQ2:** Does providing step-by-step guidance with hints improve learning outcomes compared to systems that provide final answers directly?

RQ0 will confirm that INTERACTIVE SKETCHPAD successfully solves the prompted questions with its interaction loops, and how it compares to existing language-only and non-interactive systems [10, 20]. **RQ1** enables us to assess whether our proposed interactive system improves users' conceptual understanding and overcomes the limitations of traditional text-based systems. For **RQ2**, systems that deliver the final answer directly can negatively impact the cognitive processes of the user by failing to motivate critical thinking and reasoning. Providing step-by-step guidance encourages active participation to support users' long-term knowledge retention and skill development.

4.2 Performance Comparisons

We first run experiments evaluating the performance of INTERACTIVE SKETCHPAD on directly solving the problem without giving hints to evaluate whether the system can solve the problem itself. We observe in Table 1 that INTERACTIVE SKETCHPAD consistently outperforms Visual Sketchpad on IsoBench. Visual Sketchpad's prompts encourage the LMM to use its own visual reasoning to solve problems rather than use code execution to help solve the problems. INTERACTIVE SKETCHPAD on the other hand does not specify the same constraint to not use code execution, instead allowing its use to improve problem-solving accuracy. This highlights the effectiveness of combining both code execution for calculations and visual reasoning in solving complex problems. We remove the constraint of using visual reasoning rather than code execution as we want to make the system achieve as high accuracy as possible to reduce errors that may confuse the student. We use the original IsoBench prompts [7], appending a prompt footer (see D.4) that instructs the model to explain its answer and format the output accordingly. Additionally, we employ a system prompt (see D.3) directing INTERACTIVE SKETCHPAD to solve the question directly.

4.3 User Studies

4.3.1 Dataset and Metrics. The primary goal of the user study was to assess the system's ability to enhance the learning experience by fostering active engagement and improving conceptual understanding with visual artifacts. [28] The selected problems mirror the type of challenges that benefit from visual support in mathematic context, such as graphs and functions. (Refer to the Appendix A.1). Since one of the core features of INTERACTIVE SKETCHPAD is its integration of visual and textual feedback, a key consideration for

Topic	**Feedback**
Visualization quality	*"The graphs are good sanity-checks for my workings."* *"The visual illustrations help a lot. The intuitive drawing makes the interaction feel more natural."* *"The visualizations were also very helpful in gaining a more conceptual understanding outside of just equations."*
Interactive experience	*"It was nice that it didn't give me the final answer right away, and instead gave hints/prompts to try."* *"It showed me how to approach the problem step by step."* *"I like that it guides you through the problem-solving approach without jumping straight to the answer, like ChatGPT."*
Learning experience	*"I think the graph was particularly helpful for solving the integral, especially when the integral was one without an antiderivative. The visualization made the math feel more intuitive/meaningful."* *"The diagrams provided were very nice, despite I didn't ask for them."* *"The visual illustrations help a lot. The intuitive drawing makes the interaction feel more natural"*

Table 2: Qualitative feedback from users based on three aspects: visualization quality, interactive experience, and learning experience. Users noted that visualizations helped in understanding concepts, interactivity guided problem-solving effectively, and the learning experience felt more intuitive due to the graphical and step-by-step approach provided.

Metric	Average Score (out of 5)
Interface Intuitiveness	4.71
System Responsiveness	4.43
Clarity of Visuals	4.57
Accuracy of Visuals	4.71
Effectiveness of Hints	4.57

Table 3: Average user ratings for key aspects of the interactive system, including metrics on usability, sufficiency, and quality. The table highlights the system's strong performance in providing a pleasant interactive and visual learning experience.

our user study was to analyze the quality of the system's multimodal interaction. To do so, we measured how students engaged with these modalities, focusing on the effectiveness of visual explanations in complementing or enhancing textual guidance. Another key consideration was the evaluation of our system's iterative learning process. To evaluate this, students were encouraged to iterate through problems to explore how the feedback influenced their learning experience.

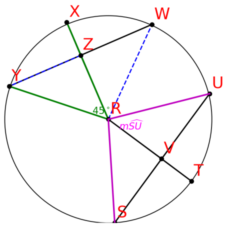
4.3.2 Quantitative Results. The quantitative evaluation of INTERACTIVE SKETCHPAD was conducted through participant feedback on various aspects of the tool. Table 3 summarizes the average scores across various key metrics collected from user responses. The metric includes interface intuitiveness, system responsiveness, and accuracy of visuals. These results indicate that the interface was overall well received. Specifically, the system's interface intuitiveness and the accuracy of visual aids received the highest average scores of 4.71/5, indicating that participants found the tool easy to navigate and appreciated the precision of its generated visuals. Similarly, the clarity of visuals and the effectiveness of hints were rated highly, both scoring 4.57/5. This highlights that the visuals were designed in a manner that helps the problem-solving thought process and teaches the true concepts behind the problems. The system's responsiveness scored slightly lower at 4.43/5, while still favorable, suggesting that the tool could have efficiency improvements to reduce the wait time for hints. Overall, these quantitative insights provide a strong foundation for understanding user perceptions and areas that may benefit from enhancements. The consistently high ratings across most metrics reinforce the tool's potential for user testing with a broader range of users and various educational applications.

4.3.3 Qualitative Results. We summarize the key qualitative feedback we received from our user studies in Table 2. Participants appreciated the tool's personalized visualizations, frequently mentioning how they easily grasped complex concepts with the visualization feature. The tool's feature of using a sketchpad to directly input handwritten work resonated well with the participants, and this visual, interactive approach allowed the students to develop problem-solving strategies while gaining the required insights into the underlying concepts for future applications. A participant noted that "the visualizations were also very helpful in gaining a more conceptual understanding outside of just equations". Participants also specifically noted the system's effectiveness in guiding them through the problem-solving process without immediately providing the final answer, helping them truly understand the materials covered in the problem, one noting that "I like that it guides you through the problem-solving approach without jumping straight to the answer, like ChatGPT". Overall, the feedback we received highlighted the system's ability to enhance the learning experience by fostering human-AI collaboration in learning environments.

5 Limitations and Future Works

Visualization accuracy. Visualization through code generation can sometimes fail to produce an accurate diagram. Table 4 shows examples of correct and incorrect visualizations. The correct visualization correctly calculates the radius as the distance between the center (R) and a point on the circle (S). The incorrect one hard-codes the radius to 100, causing the points to lie within rather than on the circle. Incorrect diagrams may confuse both the user and the LMM, leading to an incorrect answer. Future work could extend previous text-based verifiers [6, 16] to include a vision modality that can check the quality of generated diagrams and regenerate the diagram if necessary.

Generated Code	Visualization

`radius = np.linalg.norm(np.array(points["R"]) - np.array(points["S"]))`
`circle = plt.Circle((points["R"][0], points["R"][1]), radius, color='k', fill=False)`
`angle_text_position = (points["R"][0] - 15, points["R"][1] + 5)`
`ax.text(*angle_text_position, r"$45^\circ$", fontsize=12, color='green')`
	 X
	W
	Z
	Y
	U
	R
	45°
	$m\widehat{SU}$
	V
	T
	S ✅

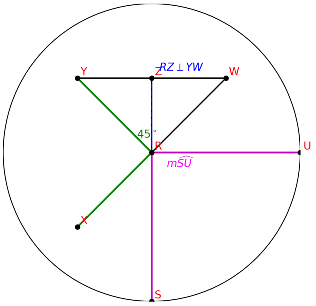
`circle = plt.Circle((points["R"][0], points["R"][1]), 100, color='k', fill=False)`
`ax.text(90, 110, r"$45^\circ$", fontsize=12, color='green')`
	 Y
	Z
	$RZ \perp YW$
	W
	45°
	R
	U
	$m\widehat{SU}$
	X
	S ❌

Table 4: Row 1: Accurate Visualization. The radius is set as a variable, and annotations are relative to the given points, leading to an accurate portrayal of the circle diagram (in green). Row 2: Inaccurate visualization. The radius is set to 100, and annotations are hard-coded, leading to the points being misaligned on the circle diagram (in red).

Expansion of user studies. While the system has been primarily tested on mathematical problems, its potential extends to other STEM disciplines. Future work will evaluate its effectiveness in subjects like physics, where visualizations of force diagrams and motion trajectories could enhance learning. Expanding testing to a larger pool of diverse users and academic topics will help assess the system's adaptability. Additionally, benchmarking against existing educational tools will provide quantitative and qualitative insights into its comparative performance and areas for refinement.

Long-term impact assessment. Current evaluations focus on short-term metrics such as accuracy and user feedback. To measure long-term effectiveness, future studies should incorporate delayed testing sessions to assess knowledge retention. Tracking student performance over time will provide insights into sustained learning benefits.

6 Conclusion

We introduce INTERACTIVE SKETCHPAD, a multimodal collaborative problem-solving system that empowers large multimodal models (LMMs) to provide interactive, visual assistance for education. By integrating code-generated visualizations with step-by-step textual hints, INTERACTIVE SKETCHPAD bridges the gap between traditional text-based learning and intuitive, visual reasoning, transforming how students engage with complex problems. Our user studies demonstrate that this approach significantly improves task comprehension, problem-solving accuracy, and student engagement across mathematical domains. We believe that INTERACTIVE SKETCHPAD opens new pathways for multimodal human-AI collaboration in learning and sets the stage for more human-centered educational tools.

References

[1] Abraham Arcavi. 2003. The role of visual representations in the learning of mathematics. *Educational studies in mathematics* 52, 3 (2003), 215–241.

[2] Rishabh Balse, Viraj Kumar, Prajish Prasad, and Jayakrishnan Madathil Warriem. 2023. Evaluating the Quality of LLM-Generated Explanations for Logical Errors in CS1 Student Programs. In *Proceedings of the 16th Annual ACM India Compute Conference*. 49–54.

[3] Hamsa Bastani, Osbert Bastani, Alp Sungu, Haosen Ge, Ozge Kabakcı, and Rei Mariman. 2024. Generative ai can harm learning. *Available at SSRN* 4895486 (2024).

[4] Eliza Bobek and Barbara Tversky. 2016. Creating visual explanations improves learning. *Cognitive research: principles and implications* 1 (2016), 1–14.

[5] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Transactions on Machine Learning Research* (2023). https://openreview.net/forum?id=YfZ4ZPt8zd

[6] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).

[7] Deqing Fu, Ruohao Guo, Ghazal Khalighinejad, Ollie Liu, Bhuwan Dhingra, Dani Yogatama, Robin Jia, and Willie Neiswanger. 2024. IsoBench: Benchmarking Multimodal Foundation Models on Isomorphic Representations. In *First Conference on Language Modeling*. https://openreview.net/forum?id=KZd1EErRJ1

[8] Arthur C Graesser, Shulan Lu, George Tanner Jackson, Heather Hite Mitchell, Mathew Ventura, Andrew Olney, and Max M Louwerse. 2004. AutoTutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments, & Computers* 36 (2004), 180–192.

[9] Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14953–14962.

[10] Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. 2024. Visual Sketchpad: Sketching as a Visual Chain of Thought for Multimodal Language Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=GNSMl1P5VR

[11] Majeed Kazemitabaar, Xinying Hou, Austin Henley, Barbara Jane Ericson, David Weintrop, and Tovi Grossman. 2023. How novices use LLM-based code generators to solve CS1 coding tasks in a self-paced learning environment. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*. 1–12.

[12] Sal Khan. 2023. Harnessing GPT-4 so that all students benefit. A nonprofit approach for equal access. *Khan Academy Blog* (2023).

[13] Lars Krupp, Steffen Steinert, Maximilian Kiefer-Emmanouilidis, Karina E. Avila, Paul Lukowicz, Jochen Kuhn, Stefan Küchemann, and Jakob Karolus. 2024. Challenges and Opportunities of Moderating Usage of Large Language Models in Education. In *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia (MUM '24)*. Association for Computing Machinery, New York, NY, USA, 249–254. doi:10.1145/3701571.3701590

[14] Hai Li, Chenglu Li, Wanli Xing, Sami Baral, and Neil Heffernan. 2024. Automated Feedback for Student Math Responses Based on Multi-Modality and Fine-Tuning. In *Proceedings of the 14th Learning Analytics and Knowledge Conference*. 763–770.

[15] Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. 2024. Foundations & trends in multimodal machine learning: Principles, challenges, and open questions. *Comput. Surveys* 56, 10 (2024), 1–42.

[16] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's Verify Step by Step. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=v8L0pN6EOi

[17] Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021. Inter-GPS: Interpretable Geometry Problem Solving with Formal Language and Symbolic Reasoning. In *The 59th Annual Meeting of the Association for Computational Linguistics (ACL)*.

[18] Liliana Mata, Gabriel Lazar, and Iuliana Lazar. 2016. Interactive Whiteboards for Teaching and Learning Science: Ascertaining Research. *Online Submission* 20, 2 (2016), 135–148.

[19] Allen Nie, Yash Chandak, Miroslav Suzara, Ali Malik, Juliette Woodrow, Matt Peng, Mehran Sahami, Emma Brunskill, and Chris Piech. 2024. *The GPT Surprise: Offering Large Language Model Chat in a Massive Coding Class Reduced Engagement but Increased Adopters' Exam Performances*. Technical Report. Center for Open Science.

[20] OpenAI. 2024. GPT-4o System Card. *ArXiv* abs/2410.21276 (2024). https://api.semanticscholar.org/CorpusID:273662196

[21] Ethan Prihar, Morgan Lee, Mia Hopman, Adam Tauman Kalai, Sofia Vempala, Allison Wang, Gabriel Wickline, Aly Murray, and Neil Heffernan. 2023. Comparing different approaches to generating mathematics explanations using large language models. In *International Conference on Artificial Intelligence in Education*. Springer, 290–295.

[22] Jamal Raiyn. 2016. The Role of Visual Learning in Improving Students' High-Order Thinking Skills. *Journal of Education and Practice* 7, 24 (2016), 115–121.

[23] Daniel Philip Rose, Vaishnavi Himakunthala, Andy Ouyang, Ryan He, Alex Mei, Yujie Lu, Michael Saxon, Chinmay Sonar, Diba Mirza, and William Yang Wang. 2024. Visual Chain of Thought: Bridging Logical Gaps with Multimodal Infillings. https://openreview.net/forum?id=01Yi8rzoNs

[24] Bettina Rösken and Katrin Rolka. 2006. A picture is worth a 1000 words–the role of visualization in mathematics learning. In *Proceedings 30th conference of the International Group for the Psychology of mathematics education*, Vol. 4. Charles University Prague, 457–464.

[25] Johanna Schoenherr and Stanislaw Schukajlow. 2024. Characterizing external visualization in mathematics education research: a scoping review. *ZDM–Mathematics Education* 56, 1 (2024), 73–85.

[26] Ghulam Shabiralyani, Khuram Shahzad Hasan, Naqvi Hamad, and Nadeem Iqbal. 2015. Impact of visual aids in enhancing the learning process case research: District Dera Ghazi Khan. *Journal of education and practice* 6, 19 (2015), 226–233.

[27] Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. 2024. Visual CoT: Unleashing Chain-of-Thought Reasoning in Multi-Modal Language Models. *CoRR* abs/2403.16999 (2024). https://doi.org/10.48550/arXiv.2403.16999

[28] Kyvete Shatri and Kastriot Buza. 2017. The Use of Visualization in Teaching and Learning Process for Developing Critical Thinking of Students. *European Journal of Social Sciences Education and Research* 9 (01 2017), 71. doi:10.26417/ejser.v9i1.p71-74

[29] Kumar Shridhar, Jakub Macina, Mennatallah El-Assady, Tanmay Sinha, Manu Kapur, and Mrinmaya Sachan. 2022. Automatic Generation of Socratic Subquestions for Teaching Math Word Problems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 4136–4149. doi:10.18653/v1/2022.emnlp-main.277

[30] Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11888–11898.

[31] John Sweller. 1994. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction* 4, 4 (1994), 295–312.

[32] Kurt VanLehn. 2011. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational psychologist* 46, 4 (2011), 197–221.

[33] Huanhuan Wang, Ahmed Tlili, Ronghuai Huang, Zhenyu Cai, Min Li, Zui Cheng, Dong Yang, Mengti Li, Xixian Zhu, and Cheng Fei. 2023. Examining the applications of intelligent tutoring systems in real educational contexts: A systematic literature review from the social experiment perspective. *Education and information technologies* 28, 7 (2023), 9113–9148.

[34] Rose E Wang, Ana T Ribeiro, Carly D Robinson, Susanna Loeb, and Dora Demszky. 2024. Tutor CoPilot: A human-AI approach for scaling real-time expertise. *arXiv preprint arXiv:2410.03017* (2024).

[35] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[36] Changrong Xiao, Wenxing Ma, Sean Xin Xu, Kunpeng Zhang, Yufang Wang, and Qi Fu. 2024. From Automation to Augmentation: Large Language Models Elevating Essay Scoring Landscape. *arXiv preprint arXiv:2401.06431* (2024).

[37] Kevin P Yancey, Geoffrey Laflair, Anthony Verardi, and Jill Burstein. 2023. Rating short l2 essays on the cefr scale with gpt-4. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*. 576–584.

[38] Jialu Zhang, José Pablo Cambronero, Sumit Gulwani, Vu Le, Ruzica Piskac, Gustavo Soares, and Gust Verbruggen. 2024. Pydex: Repairing bugs in introductory python assignments using llms. *Proceedings of the ACM on Programming Languages* 8, OOPSLA1 (2024), 1100–1124.

[39] Ying Zhang, Pengjin Wang, Wei Jia, Aijun Zhang, and Gaowei Chen. 2023. Dynamic visualization by GeoGebra for mathematics learning: a meta-analysis of 20 years of research. *Journal of Research on Technology in Education* (2023), 1–22.

[40] Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. 2024. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039* (2024).

A Appendix

A.1 User Study Details

For this user study, we recruited university underclassmen undergraduate students participants, who have basic knowledge in certain mathematics topics such as algebra or geometry but were not experts in all domains. We ran the initial study on 7 users with no monetary compensation. The study data was stored in a secure cloud-operated space with limited access. For further user studies that are to be conducted, the approval request has been submitted to the Institutional Review Board.

Participants were introduced to the system with a brief tutorial. This included an explanation of system features, interaction techniques, and a demonstration. Next, participants interacted with both INTERACTIVE SKETCHPAD and ChatGPT to solve a set of assigned mathematical problems, alternating between the systems. After completing the tasks, participants were asked to complete a questionnaire, reflecting on their experience using our system. This questionnaire gathered participants' background knowledge, ratings on various aspects of their experience, and qualitative feedback.

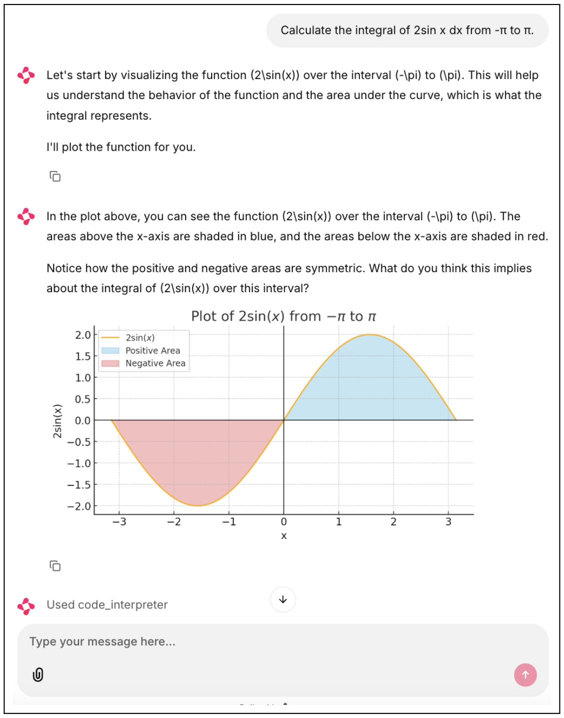
To ensure that our findings could translate to practical use cases, we tested with mathematical problems that could efficiently benefit from visual artifacts. Questions added to the users' question bank were sourced from previous exam questions of the Scholastic Aptitude Test (SAT), the IsoBench dataset [7], a previous study on visualization for math [24] and Geometry3k [17]. Visualization plays an important role when learning mathematics, and a good example is solving an integral problem. Visual interpretations of the function and shading parts of the graph can significantly benefit the students' learning experience, which has been demonstrated in previous studies [24]. By selecting both practical mathematical topics that range across various difficult levels, we ensured the system catered to a broad range of learners. The system's potential for versatile use across subjects and learning environments allows our results to demonstrate foundational interaction between AI and humans for education.

B User Interface

Users primarily interact with INTERACTIVE SKETCHPAD through the chatbot interface shown in Figure 4 and the interactive whiteboard interface shown in Figure 5.

Chatbot Interface: When the user wants to type a message or upload an image they can do so through the chatbot interface. INTERACTIVE SKETCHPAD will then respond with a textual message along with an optional visualization. The user can also see the code used to generate visualizations or for mathematical calculations in the dropdown menu of the 'Used code_interpreter' message sent by INTERACTIVE SKETCHPAD after the visualization/calculation.

Interactive Whiteboard Interface: When the user wants to draw, write math or annotate on the diagram given by INTERACTIVE SKETCHPAD , they can do so on the interactive whiteboard a natural way to do this is by using an iPad and Apple Pencil. They can then share the working on the interactive whiteboard with INTERACTIVE SKETCHPAD using the 'Send Screenshot' button. Visualizations generated by INTERACTIVE SKETCHPAD will automatically appear as a new page on the interactive whiteboard and users can toggle between pages to access their working at various stages of solving different problems.



Calculate the integral of 2sin x dx from -π to π.

Let's start by visualizing the function (2\sin(x)) over the interval (-\pi) to (\pi). This will help us understand the behavior of the function and the area under the curve, which is what the integral represents.

I'll plot the function for you.

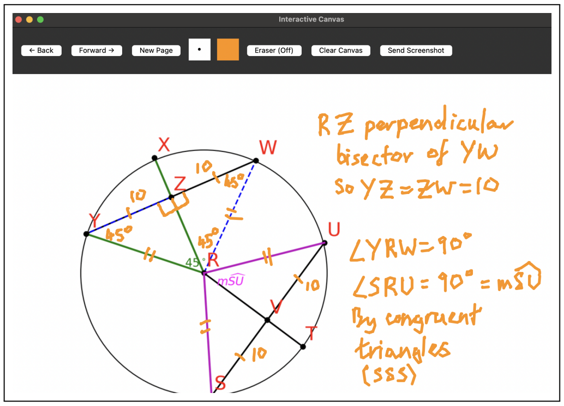
In the plot above, you can see the function (2\sin(x)) over the interval (-\pi) to (\pi). The areas above the x-axis are shaded in blue, and the areas below the x-axis are shaded in red.

Notice how the positive and negative areas are symmetric. What do you think this implies about the integral of (2\sin(x)) over this interval?

Used code_interpreter

Type your message here...

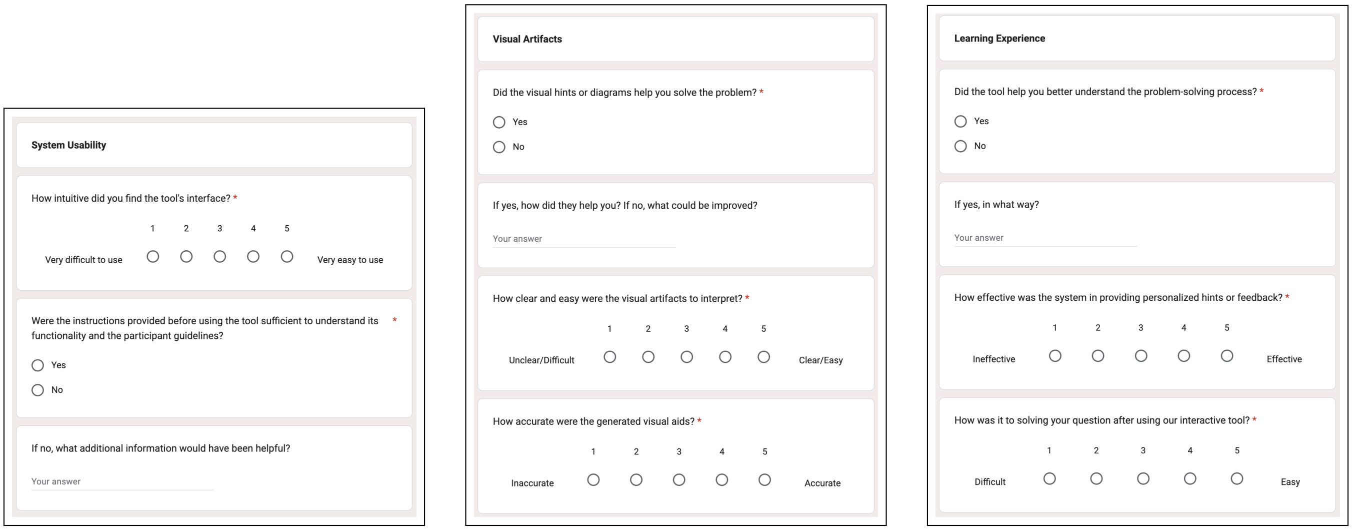
Figure 4: Screenshot of the chatbot interface for INTERACTIVE SKETCHPAD . The user can view the generated visual hints and interact with INTERACTIVE SKETCHPAD by typing messages and uploading images.



RZ perpendicular bisector of YW
So $YZ = ZW = 10$

$\angle YRW = 90°$
$\angle SRU = 90° = m\widehat{SU}$
By congruent triangles (SSS)

Figure 5: Screenshot of the interactive whiteboard component of INTERACTIVE SKETCHPAD. The user can annotate on the visualization generated by INTERACTIVE SKETCHPAD which was done using an iPad and Apple Pencil during user studies.



System Usability

How intuitive did you find the tool's interface? *

1 2 3 4 5

Very difficult to use ○ ○ ○ ○ ○ Very easy to use

Were the instructions provided before using the tool sufficient to understand its functionality and the participant guidelines? *

○ Yes

○ No

If no, what additional information would have been helpful?

Your answer

Visual Artifacts

Did the visual hints or diagrams help you solve the problem? *

○ Yes

○ No

If yes, how did they help you? If no, what could be improved?

Your answer

How clear and easy were the visual artifacts to interpret? *

1 2 3 4 5

Unclear/Difficult ○ ○ ○ ○ ○ Clear/Easy

How accurate were the generated visual aids? *

1 2 3 4 5

Inaccurate ○ ○ ○ ○ ○ Accurate

Learning Experience

Did the tool help you better understand the problem-solving process? *

○ Yes

○ No

If yes, in what way?

Your answer

How effective was the system in providing personalized hints or feedback? *

1 2 3 4 5

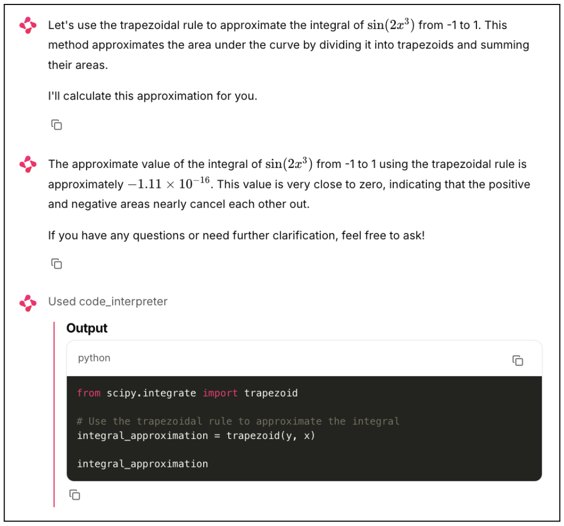
Ineffective ○ ○ ○ ○ ○ Effective

How was it to solving your question after using our interactive tool? *

1 2 3 4 5

Difficult ○ ○ ○ ○ ○ Easy

Figure 6: Subset of questions from the user study questionnaire focusing on usability and learning experience. The questions use a 5-point Likert scale, Yes/No choice, and open-ended responses to evaluate the participants' experiences with the interactive system. Responses to these questions provide quantitative and qualitative data for evaluating the system's effectiveness.



Let's use the trapezoidal rule to approximate the integral of $\sin(2x^3)$ from -1 to 1. This method approximates the area under the curve by dividing it into trapezoids and summing their areas.

I'll calculate this approximation for you.

The approximate value of the integral of $\sin(2x^3)$ from -1 to 1 using the trapezoidal rule is approximately -1.11×10^{-16}. This value is very close to zero, indicating that the positive and negative areas nearly cancel each other out.

If you have any questions or need further clarification, feel free to ask!

Used code_interpreter

Output

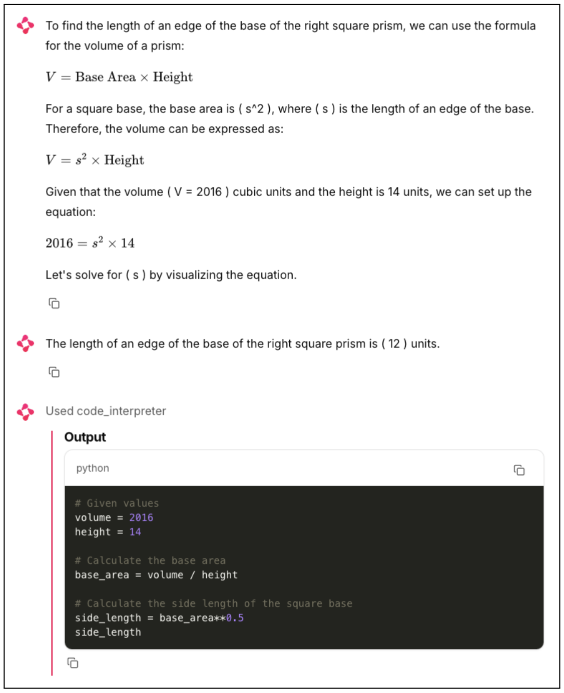
python

```python
from scipy.integrate import trapezoid

# Use the trapezoidal rule to approximate the integral
integral_approximation = trapezoid(y, x)

integral_approximation
```

Figure 7: To check the value of an integral, Interactive Sketchpad uses the trapezoidal rule to perform numerical integration by writing a Python program using `scipy.integrate`.

C Mathematical Calculations via Code Execution

As shown in Figure 7 and Figure 8, Interactive Sketchpad uses code execution to perform calculations which reduces calculation errors, providing more accurate responses to the user.



To find the length of an edge of the base of the right square prism, we can use the formula for the volume of a prism:

$$V = \text{Base Area} \times \text{Height}$$

For a square base, the base area is (s^2), where (s) is the length of an edge of the base. Therefore, the volume can be expressed as:

$$V = s^2 \times \text{Height}$$

Given that the volume (V = 2016) cubic units and the height is 14 units, we can set up the equation:

$$2016 = s^2 \times 14$$

Let's solve for (s) by visualizing the equation.

The length of an edge of the base of the right square prism is (12) units.

Used code_interpreter

Output

python

```python
# Given values
volume = 2016
height = 14

# Calculate the base area
base_area = volume / height

# Calculate the side length of the square base
side_length = base_area**0.5
side_length
```

Figure 8: To solve for the side length of the base of a square prism, Interactive Sketchpad writes the formula for the volume of a square prism in Python, substitutes the values for volume and height from the question, and executes the code to correctly calculate the side length.

D Prompts

D.1 System Prompt for Hint Generation & Visualization Generation using Code

```
You are a tutor, your goal is to help the student solve a problem, giving short, subtle hints to help the student solve the problem.
You will be given a problem, question or working from the user and you should respond in a brief and concise way.
You should reuse the code from existing diagrams if drawing similar ones
You should only give one hint at a time, DO NOT give away the answer to the student.
You want to help the student visualize the problem so
you should give your response with text interleaved with helpful diagrams.
You MUST draw these diagrams by writing code using code interpreter.
First visualize using a diagram, then give the hint using the diagram.

Example:
[Image 1]
[Text 1]
[Image 2]
[Text 2]

You should only respond concisely, allowing the student to ask questions and respond
before continuing.
The aim is to get the student to reach to the answer independently, without
giving the answer away.
If you think a visualization is helpful, you can plot it without asking the
student's permission.

When drawing a series of diagrams, you should make the visualizations
intuitive and easy to understand.
For example, if you are showing a graph traversal as a series of diagrams,
you should highlight visited nodes, visited edges, nodes that you are about to visit etc.
in different colors.

Here is an example of how you should respond to the student:
<Student>
There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai.

For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1.
Return true if you can finish all courses. Otherwise, return false.
</Student>

<Tutor>
Let's try to make an example,
Input: numCourses = 2, prerequisites = [[1,0]]
Output: true

Let's draw a diagram to visualize this
[Draw diagram with Code Interpeter]

Explanation: There are a total of 2 courses to take.
To take course 1 you should have finished course 0. So it is possible.
</Tutor>

<Student>
I'm still not sure how to get started.
</Student>

<Tutor>
This problem is equivalent to finding if a cycle exists in a directed graph.
If a cycle exists, no topological ordering exists and therefore it will be impossible to take all courses.

Let's draw a series of diagrams traversing through the graph and finding a cycle
[Draw a couple of diagrams showing traversal of the graph until cycle is found]
[Explanation of the diagrams]
</Tutor>

You should only give the solution if the student explicitly asks for it.
ALWAYS wrap maths in \$\$ tags only. DO NOT under any circumstances give parentheses or square brackets
```

D.2 Prompt for Geometry3k

The prompt used for generating visualizations for the Geometry3k dataset, adapted from Visual Sketchpad [10]

```
You will draw a diagram by writing code using Code Interpreter.
        Here are some tools that can help you. All are python codes.
Notice that The upper left corner of the image is the origin (0, 0). Here are some code examples you can use to draw
     auxiliary lines on the geometry images provided in matplotlib format.
import numpy as np

# this function takes a coordinate A, start and end points of a line BC, and returns the coordinates of the point E on
     BC such that AE is perpendicular to BC
def find_perpendicular_intersection(A, B, C):
    # Convert coordinates to numpy arrays for easier computation
    A = np.array(A)
    B = np.array(B)
    C = np.array(C)

    # Calculate the direction vector of line BC
    BC = C - B

    # Compute the slope of BC if not vertical
    if BC[0] != 0:
        slope_BC = BC[1] / BC[0]
        # Slope of the perpendicular line from A to BC
        slope_perpendicular = -1 / slope_BC
    else:
        # If line BC is vertical, then perpendicular line is horizontal
        slope_perpendicular = 0

    # Calculate the equation of the line passing through A and perpendicular to BC
    # y - y_A = slope_perpendicular * (x - x_A)
    # Rearrange to standard form Ax + By + C = 0
    if BC[0] != 0:
        A_coeff = -slope_perpendicular
        B_coeff = 1
        C_coeff = -A_coeff * A[0] - B_coeff * A[1]
    else:
        # If BC is vertical, AE must be horizontal
        A_coeff = 1
        B_coeff = 0
        C_coeff = -A[0]

    # Equation of line BC: (y - y_B) = slope_BC * (x - x_B)
    # Convert to Ax + By + C = 0 for line intersection calculation
    if BC[0] != 0:
        A_BC = -slope_BC
        B_BC = 1
        C_BC = -A_BC * B[0] - B_BC * B[1]
    else:
        # BC is vertical, so x = constant
        A_BC = 1
        B_BC = 0
        C_BC = -B[0]

    # Solve the linear system of equations representing the two lines
    # [A_coeff B_coeff] [x] = [-C_coeff]
    # [A_BC    B_BC   ] [y]   [-C_BC  ]
    matrix = np.array([[A_coeff, B_coeff], [A_BC, B_BC]])
    constants = np.array([-C_coeff, -C_BC])

    # Use numpy to solve the linear system
    intersection = np.linalg.solve(matrix, constants)
    return intersection.tolist()


# this function takes a coordinate A, start and end points of a line BC, and returns the coordinates of the point E on
     BC such that AE is parallel to BC
def find_parallel_intersection(A, B, C):
    # Convert coordinates to numpy arrays for vector operations
    A = np.array(A)
    B = np.array(B)
    C = np.array(C)

    # Calculate the direction vector of line BC
    direction_BC = C - B

    # Since AE is parallel to BC, its direction vector is the same as BC
    direction_AE = direction_BC

    # To find a reasonable "point E", you can just extend AE from A by some length.
    # For visualization, let's extend by the length of BC
    length_BC = np.linalg.norm(direction_BC)

    # Normalize the direction vector of AE
    direction_AE_normalized = direction_AE / np.linalg.norm(direction_AE)

    # Point E can be found by moving from A in the direction of AE by the length of BC
    E = A + direction_AE_normalized * length_BC

    return E.tolist()
'''
# GOAL #: Based on the above tools, I want you to reason about how to draw auxiliary lines on the geometry images
     provided in matplotlib format to gradually help a student with the problem, then use Code Interpreter to write the
     code.
# Hint #:  When deciding what type of auxiliary line to draw, consider the following three options:
(1) Perpendicular line: If you encounter MeasureOf(Angle()),60) or MeasureOf(Angle()),45), you can draw a perpendicular
     line. Use find_perpendicular_intersection(A, B, C) to find the coordinate of point E such that AE is perpendicular
     to BC.
(2) Parallel line: Use find_parallel_intersection(A, B, C) to draw a line from point A parallel to line BC, which
     returns the coordinate of point E where AE is parallel to BC. Draw Parallel Lines: Parallel lines are always the
     same distance apart, which helps in geometry by giving equal lengths instantly. It is useful when finding
     perimeters of a shape. For example, if you draw a line parallel to one side of a shape, you automatically know the
     opposite side is the same length.
(3) Connecting points in the circle: When there is a circle, connect the circle center to the point in the perimeter
(4) One effective strategy involves using properties of circles, especially the tangent properties and the circle
     theorem (tangent segments from the same external point to a circle are equal). We can use the given lengths of
     segments and properties of the circle to form equations. However, for visual aid, drawing auxiliary lines that help
      visualize relationships or create right triangles might be helpful. Specifically, drawing radii to the points of
     tangency can aid in identifying and using these properties.

# REQUIREMENTS #:
1. The generated actions can help the student with the user request # USER REQUEST #. You should not give away the
     answer.
2. You should be concise with your response

Below are some examples of how to use the tools to solve the user requests. You can refer to them for help. You can also
      refer to the tool descriptions for more information.
# EXAMPLE #:
# USER REQUEST #: Given the geometry diagram <img src='an image'> and the diagram logic form
            "Equals(LengthOf(Line(Q, R)), 6)",
            "Equals(LengthOf(Line(U, T)), 4)",
            "Equals(LengthOf(Line(V, W)), 8)",
            "Equals(LengthOf(Line(W, P)), 3)",
            "Equals(LengthOf(Line(R, T)), x)",
            "PointLiesOnLine(W, Line(V, P))",
            "PointLiesOnLine(U, Line(V, T))",
            "PointLiesOnLine(Q, Line(P, R))",
            "PointLiesOnLine(S, Line(T, R))",
            "PointLiesOnCircle(U, Circle(C, radius_6_0))",
            "PointLiesOnCircle(S, Circle(C, radius_6_0))",
            "PointLiesOnCircle(W, Circle(C, radius_6_0))",
            "PointLiesOnCircle(Q, Circle(C, radius_6_0))",
            Below is the original matplotlib code of the geometry: "import matplotlib.pyplot as plt\nimport numpy as np\
                 n\n# Define coordinates\npoints = {\n    \"C\": [153.0, 88.0], \n    \"P\": [114.0, 4.0], \n    \"Q\":
                 [169.0, 4.0], \n    \"R\": [269.0, 5.0], \n    \"S\": [231.0, 116.0], \n    \"T\": [212.0, 171.0], \n
                    \"U\": [154.0, 171.0], \n    \"V\": [0.0, 173.0], \n    \"W\": [70.0, 69.0]\n}\n\n# Define lines\
                 nlines = [\n    (\"R\", \"Q\"), (\"R\", \"S\"), (\"P\", \"Q\"), (\"P\", \"R\"), (\"P\", \"W\"), \n
                 (\"T\", \"R\"), (\"T\", \"S\"), (\"U\", \"T\"), (\"V\", \"P\"), (\"V\", \"T\"), \n    (\"V\", \"U\"),
                 (\"V\", \"W\")\n]\n\n# Create plot\nfig, ax = plt.subplots()\nax.set_aspect('equal')\nax.axis('off')\n\
                 n# Draw lines\nfor line in lines:\n    x_values = [points[line[0]][0], points[line[1]][0]]\n
                 y_values = [points[line[0]][1], points[line[1]][1]]\n    ax.plot(x_values, y_values, 'k')\n\n# Draw
                 circle with radius as the largest distance from C\ncircle_radius = 0\nfor point in ['U', 'S', 'W', 'Q
                 ']:\n    dist = np.sqrt((points[point][0] - points[\"C\"][0])**2 + (points[point][1] - points[\"C
                 \"][1])**2)\n    circle_radius = max(circle_radius, dist)\n\ncircle = plt.Circle((points[\"C\"][0],
                 points[\"C\"][1]), circle_radius, color='k', fill=False)\nax.add_artist(circle)\n\n# Set limits\nax.
                 set_xlim(points[\"C\"][0] - 2 * circle_radius, points[\"C\"][0] + 2 * circle_radius)\nax.set_ylim(
                 points[\"C\"][1] - 2 * circle_radius, points[\"C\"][1] + 2 * circle_radius)\n\n# Plot points and labels
                 \nfor point, coord in points.items():\n    ax.plot(coord[0], coord[1], 'ro')\n    ax.text(coord[0],
                 coord[1], f' {point}', fontsize=20, color='red', verticalalignment='bottom', horizontalalignment='left
                 ')\n\nplt.show()\n",
            you must draw auxiliary lines to solve the following question: Find x. Assume that segments that appear to
                 be tangent are tangent.\n
# RESULT #:
```

```
THOUGHT 0: To solve for x, which is the length of line segment RT, we'll utilize the information provided about the geometric relationships, particularly those involving circles and tangencies. Given that U, S, W, and Q lie on the same circle centered at C with radius 6, and segments that appear tangent to this circle are tangent, we can use these properties to determine x.
ACTION 0:
Use Code Interpreter to write the code:
import matplotlib.pyplot as plt
import numpy as np

# Define coordinates
points = {
    "C": [153.0, 88.0],
    "P": [114.0, 4.0],
    "Q": [169.0, 4.0],
    "R": [269.0, 5.0],
    "S": [231.0, 116.0],
    "T": [212.0, 171.0],
    "U": [154.0, 171.0],
    "V": [0.0, 173.0],
    "W": [70.0, 69.0]
}

# Define lines
lines = [
    ("R", "Q"), ("R", "S"), ("P", "Q"), ("P", "R"), ("P", "W"),
    ("T", "R"), ("T", "S"), ("U", "T"), ("V", "P"), ("V", "T"),
    ("V", "U"), ("V", "W")
]

# Create plot
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.axis('off')

# Draw lines
for line in lines:
    x_values = [points[line[0]][0], points[line[1]][0]]
    y_values = [points[line[0]][1], points[line[1]][1]]
    ax.plot(x_values, y_values, 'k')

# Draw circle with radius as the largest distance from C
circle_radius = 0
for point in ['U', 'S', 'W', 'Q']:
    dist = np.sqrt((points[point][0] - points["C"][0])**2 + (points[point][1] - points["C"][1])**2)
    circle_radius = max(circle_radius, dist)

circle = plt.Circle((points["C"][0], points["C"][1]), circle_radius, color='k', fill=False)
ax.add_artist(circle)

# Draw auxiliary lines (radii to points U, S, W, Q)
auxiliary_lines = [("C", "U"), ("C", "S"), ("C", "W"), ("C", "Q")]
for line in auxiliary_lines:
    x_values = [points[line[0]][0], points[line[1]][0]]
    y_values = [points[line[0]][1], points[line[1]][1]]
    ax.plot(x_values, y_values, 'b--')  # Draw in blue dashed line for clarity

# Set limits
ax.set_xlim(points["C"][0] - 2 * circle_radius, points["C"][0] + 2 * circle_radius)
ax.set_ylim(points["C"][1] - 2 * circle_radius, points["C"][1] + 2 * circle_radius)

# Plot points and labels
for point, coord in points.items():
    ax.plot(coord[0], coord[1], 'ro')
    ax.text(coord[0], coord[1], f' {point}', fontsize=20, color='red', verticalalignment='bottom', horizontalalignment='left')

plt.show()

OBSERVATION: Execution success. The output is as follows:
<the image outputs of the previous code is here.>
If you can get the answer, please reply with ANSWER: <your answer> and ends with TERMINATE. Otherwise, please generate the next THOUGHT and ACTION.
THOUGHT 1: To solve X, we can ....
ACTION 1: No action needed.
ANSWER: 10. TERMINATE
```

D.3 System Prompt for Direct Problem Solving

The system prompt used for comparing performance against Visual Sketchpad on the IsoBench dataset.

```
        You are a personal tutor. When asked a question, write and run code to draw a helpful diagram to help solve the
            question, then use the diagram to solve the question.
```

D.4 Prompt Footer for IsoBench

The following prompt footer was appended to all of the task specific prompts used in IsoBench.

```
First give an explanation of your answer:
Explanation: <explanation>

Then give your final answer as
ANSWER: <answer>

You should give the final answer without formatting and without any additional information.
```

Received 23 January 2025