Can GPT-4 Trace Rays?

T. H. Feng[1], B. C. Wünsche[1], P. Denny[1], A. Luxton-Reilly[1], and S. Hooper[1]

[1]University of Auckland, New Zealand

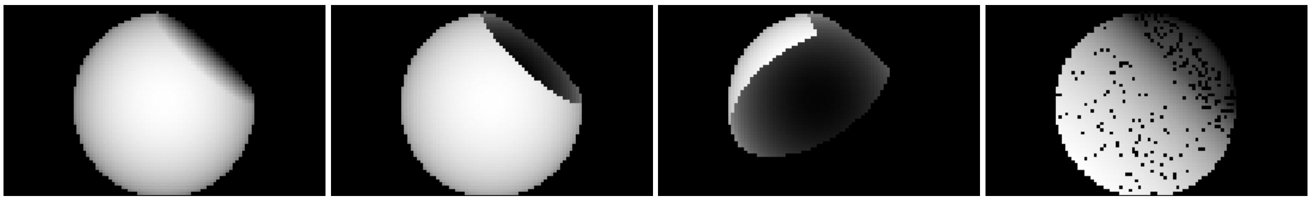


Figure 1: *The output image rendered from the correct answer to a programming question computing the intersection between a ray and a solid sphere sliced by a plane (1st left), and three output images rendered from three incorrect attempts of solving said question by GPT-4 (2nd, 3rd, 4th left).*

Abstract

Ray Tracing is a fundamental concept often taught in introductory Computer Graphics courses, and Ray-Object Intersection questions are frequently used as practice for students, as they leverage various skills essential to learning Ray Tracing or Computer Graphics in general, such as geometry and spatial reasoning. Although these questions are useful in teaching practices, they may take some time and effort to produce, as the production procedure can be quite complex and requires careful verification and review. From the recent advancements in Artificial Intelligence, the possibility of automated or assisted exercise generation has emerged. Such applications are unexplored in Ray Tracing education, and if such applications are viable in this area, then it may significantly improve the productivity and efficiency of Computer Graphics instructors. Additionally, Ray Tracing is quite different to the mostly text-based tasks that LLMs have been observed to perform well on, hence it is unclear whether they can cope with these added complexities of Ray Tracing questions, such as visual processing and 3D geometry. Hence we ran some experiments to evaluate the usefulness of leveraging GPT-4 for assistance when creating exercises related to Ray Tracing, more specifically Ray-Object Intersection questions, and we found that an impressive 67% of its generated questions can be used in assessments verbatim, but only 42% of generated model solutions were correct.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Ray tracing; Natural language generation;*** • ***Applied computing*** $\rightarrow$ ***Education;***

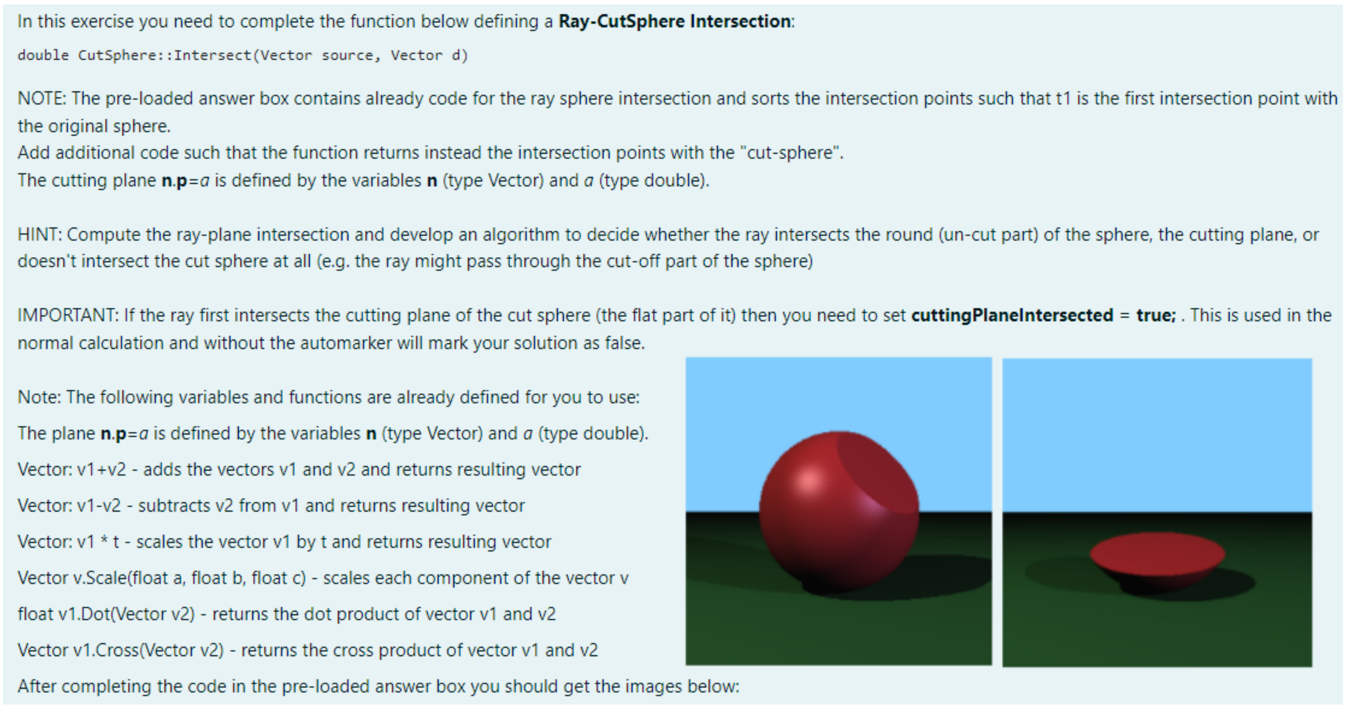
1. Introduction

Ray Tracing is an integral component of introductory CG (Computer Graphics) courses [BWF17]. Although Ray Tracing is one of the most fundamental topics in CG, understanding its related concepts requires a combination of skills, such as spatial reasoning and linear algebra. Students may find it challenging to grasp the theories under Ray Tracing, and they may need extensive practice to understand the details fully [SWLR19].

One prominent exercise for learning Ray Tracing is performing ROI (Ray-Object Intersection) computations. Such exercises usually provide students with the description of a ray (starting point, direction) and an object (shape, coordinates, orientation, etc.), and the students are expected to write a function to determine whether and where the ray intersects the object. An example of such exercises is shown in Figure 2. Many CG courses offered in top universities for Computer Science also incorporate similar exercises in their assessments [MIT12, Sta22, UCB23]. Through engaging in these exercises, students learn the mechanics of a camera in a scene, the underlying mathematics and geometry of various surfaces, and familiarise themselves with the components of a realistic rendered image, hence gaining a better understanding of various fundamental principles of Ray Tracing.

One of the best ways of learning programming concepts is through repeated practice [DLRTH11], but this is not easy for instructors to accomplish with Ray Tracing, due to the complexity of these programming exercises and the effort required to produce

DOI: 10.2312/eged.20241003



In this exercise you need to complete the function below defining a **Ray-CutSphere Intersection**:

```
double CutSphere::Intersect(Vector source, Vector d)
```

NOTE: The pre-loaded answer box contains already code for the ray sphere intersection and sorts the intersection points such that t1 is the first intersection point with the original sphere.
Add additional code such that the function returns instead the intersection points with the "cut-sphere".
The cutting plane **n.p**=*a* is defined by the variables **n** (type Vector) and *a* (type double).

HINT: Compute the ray-plane intersection and develop an algorithm to decide whether the ray intersects the round (un-cut part) of the sphere, the cutting plane, or doesn't intersect the cut sphere at all (e.g. the ray might pass through the cut-off part of the sphere)

IMPORTANT: If the ray first intersects the cutting plane of the cut sphere (the flat part of it) then you need to set **cuttingPlaneIntersected** = **true;** . This is used in the normal calculation and without the automarker will mark your solution as false.

Note: The following variables and functions are already defined for you to use:

The plane **n.p**=*a* is defined by the variables **n** (type Vector) and *a* (type double).

Vector: v1+v2 - adds the vectors v1 and v2 and returns resulting vector

Vector: v1-v2 - subtracts v2 from v1 and returns resulting vector

Vector: v1 * t - scales the vector v1 by t and returns resulting vector

Vector v.Scale(float a, float b, float c) - scales each component of the vector v

float v1.Dot(Vector v2) - returns the dot product of vector v1 and v2

Vector v1.Cross(Vector v2) - returns the cross product of vector v1 and v2

After completing the code in the pre-loaded answer box you should get the images below:

Figure 2: *An example of a challenging ROI exercise used in an introductory CG course*

such exercises. For example, creating questions for an automatic assessment tool, such as CodeRunner [LH16], requires the following steps.

1. Choose an object to be ray traced in the question. Different objects have different geometric complexities, hence their corresponding ROI solutions are of different difficulty levels. The object must meet the complexity requirements for an undergraduate Computer Science student (i.e., not too basic and not too complex).
2. Write a coherent paragraph describing the problem, including a description of the chosen object. The instructor must also ensure that all required information is provided in the question and that it is rigorous and clear of potentially confusing statements.
3. Write a model solution and several test cases of different camera positions and ray directions, then verify the model solution by manually determining whether the output image is plausible, i.e., corresponds with the object and conditions given.

From the rapid growth and popularisation of LLMs (Large Language Models), methods and tools using LLMs to aid teaching have been developed [LSSD23, HAMD23], and automated generation of exercises for students became a possibility [SDHL22]. Although automated exercise generation has been experimented with many subjects and topics, as far as we are aware, the possibility of this is unexplored in Ray Tracing. If LLMs are capable of generating exercise questions related to Ray Tracing, then this may potentially save CG instructors a lot of time and effort. This also opens the doors to more readily available Ray Tracing exercises for students.

Leveraging LLMs in Ray Tracing is also of particular interest from a research perspective, since it combines moderately complex mathematics, with code, textural descriptions, and visual output, hence it may be a challenging task for LLMs and this may yield different results from most previous publications which considered more homogeneous topics, such as introductory programming.

We chose GPT-4 [Ope23] as the model to evaluate since it is the most readily available and popular, also past research has shown its great capabilities in other areas of academia [NKM*23, KBGA23, AQ23]. We ran a series of experiments using GPT-4 to write ROI programming questions and also their model solutions, analysed the results, and then discussed the feasibility of using GPT-4 to alleviate the effort in writing these questions.

In summary, we aim to answer the following research question.

To what extent can GPT-4 assist instructors with creating Ray Tracing questions, especially ROI programming questions and model solutions, for assessment purposes?

2. Related Work

2.1. Ray Tracing Education

Ray Tracing is an illumination modelling technique that aims to generate photorealistic images at the cost of computation complexity [Whi79, Gla89]. Ray Tracing is widely used in various applications, such as static Computer-Generated Imagery and visual effects for film and television [IKSZ03, PBD*10, Ped19]. Since it is one of the most important concepts in CG, Ray Tracing is a common topic to be taught in any CG course [BWF17]. Furthermore, Shene achieved success in teaching a CG course with a focus on Ray Tracing, as it involves many fundamental concepts in CG [She02]. Therefore, optimising the pedagogy for Ray Tracing can be immensely beneficial for CG instructors around the world.

Several tools have been developed to assist instructors in teaching Ray Tracing. One such tool is *CodeRunner* [LH16], an online educational platform that allows instructors to upload programming questions for students to attempt and receive immediate feedback. CodeRunner contains numerous Ray Tracing programming questions with test cases for students to develop and test their code, it also provides a sandbox for students to design scenes to be ray traced, but at this time students cannot define new customisable objects and their corresponding Ray-Object Intersections.

Two other educational tools for Ray Tracing are *GraphicsMentor* [NS02] and *Rayground* [VGV*20]. *GraphicsMentor* is an application that allows students to view a scene containing various objects, and interact with the scene by changing the parameters of the objects, camera, and light sources via a GUI. This tool allows students to more easily visualise and understand Ray Tracing concepts, such as surfaces, the geometry of light rays, and illumination models. *Rayground* is an application similar to *GraphicsMentor*, except that it is web-based for easier access, and the parameters are changed by modifying the source code shown alongside the rendered scene. One of the advantages that *Rayground* has over *GraphicsMentor* is that it allows students to directly experiment with the source code, and hence become more familiar with the programming environment used in CG applications.

There are many more educational tools for Ray Tracing [Rus99, GFE*12, dlHvWFK22], and such tools are helpful for beginners trying to grasp concepts, but hands-on exercises are more helpful for gaining a deeper understanding. As far as we know, no tools that assist the creation of such exercises exist, and this is a potential avenue for research and development in CG.

2.2. LLMs in Education

LLMs have obtained impressive performance in various fields of academia, such as medicine [NKM*23], law [KBGA23], and numerous other subjects [AQ23], this has led to many opportunities across a wide range of disciplines [AAAA*23, TZSZ23, YH23]. In the context of Computing Education, the effect of LLMs is even more prominent [DPB*24]. Many studies have found that LLMs are capable of comfortably passing CS1/CS2 assessments [FADB*22, DKG23, FADLR*23, SAA*23], and instructors have leveraged the capabilities of LLMs to produce some revolutionary teaching tools in Computing Education. Sarsa et al. demonstrated that OpenAI Codex is capable of generating programming exercises and code explanations that are of satisfactory standards [SDHL22]. Liffiton et al. and Hicke et al. have shown the possibility of using LLMs to provide students with personalised support in Computing Education and beyond [LSSD23,HAMD23].

However, limited research regarding LLMs has been conducted in CG, and the possibility of advancements in CG from LLMs is still largely unexplored. Feng et al. evaluated the performance of GPT-4 in CG assessments and found that GPT-4 answered 42.1% of the questions correctly, and only 36.4% of questions related to Ray Tracing [FDW*24]. They also found that GPT-4 could not consistently reach the performance of an average student.

3. Methods

3.1. Overview

There are several possible avenues where the assistance of LLMs can relieve manual effort when writing Ray Tracing exercises. We chose two tasks in the procedure and attempted to use GPT-4 to automate the tasks. From the performance of GPT-4 in these tasks, we can estimate the degree of usefulness that GPT-4 provides in assisting with writing Ray Tracing exercises, and furthermore estimate the capabilities of GPT-4 in other tasks related to Ray Tracing.

Firstly, we asked GPT-4 to write ROI programming questions given some pre-chosen objects and evaluated the coherence and rigour of the output questions. Secondly, we asked GPT-4 to write model solutions for some ROI programming questions and evaluated the correctness of the solutions. Further details are given below in the following subsections.

We accessed GPT-4 via the OpenAI API. The temperature of GPT-4 indicates the degree of randomness of its responses, where 1 is the default value, 0 is completely deterministic, and 2 is the maximum allowed randomness. We used the temperature 0.75 for the responses in this study, as it was observed to achieve the best performance across a range of temperatures in a similar study [PSKD23]. The System Message used in this study is as follows:

"You are an instructor for an introductory Computer Graphics course in a university. You are currently writing Ray-Object Intersection exercises for an upcoming assessment related to Ray Tracing."

3.2. Problem Design

After the instructor has chosen an appropriate object, there is still a significant amount of work involved in formulating an appropriate question. The instructor has to ensure that the question is coherent and clear, and the description of the object given uniquely corresponds to a chosen object, i.e., one set of unique parameters describes a unique object.

This is not a trivial process as different objects may contain different parameters in their descriptions, and even the same parameters may have different meanings for different objects. For example, a sphere can be uniquely described using its centre and radius, whereas a cylinder can be uniquely described using the centre and radius of its base, its height, and its axis. Hence an instructor must be attentive and thoughtful when writing object descriptions to avoid discrepancies between solutions derived from different interpretations.

We chose 10 objects suitable for ROI exercises of varying difficulties, and a set of evaluation criteria to assess the quality of the responses. We then used a series of prompts to query GPT-4 to write 10 programming questions using each of the objects. The prompts used are in the following template ("[OBJECT]" is a placeholder for the name of the object used):

"Please write a Ray-Object Intersection programming question

using a(n) [OBJECT]. Please use clear wording and avoid discrepancies in the solutions. Please ensure that any descriptions of the object correspond to the object uniquely."

We then evaluated the coherence and completeness of all 100 questions using the rubric below.

- Coherent: The response aligns with the request, i.e., the response is in the form of an ROI assessment question and the object used is the same as specified in the request. The question is unambiguous, easy to understand, and poses no contradictions.
- Unique: The response is coherent (as defined above). The object description uniquely defines the object, and the question does not lead to multiple solutions due to different interpretations.
- Complete: The response is coherent and unique (as defined above). The question is ready to be used in an assessment without modifications, i.e., free of any errors.

3.3. Problem Solving

After designing an ROI programming question, the instructor has to write a model solution for the question, and this may require some time and effort due to the visual processing and code writing needed to solve an ROI question. If GPT-4 is capable of solving such questions, then the process of writing model solutions may be automated. This may also serve as a benchmark for the performance of GPT-4 in ROI programming questions.

Reusing the 10 chosen objects from Subsection 3.2, we wrote a viable ROI programming question for each object (10 programming questions in total). Then we asked GPT-4 to provide a model solution for each of the questions.

We chose Python as the programming language to be used, as GPT-4 is the most commonly used in Python out of all programming languages. GPT-4 was given 10 independent attempts to solve each question. The attempts were then evaluated for correctness by output comparison using several test cases written by us (i.e., comparing the output distance matrix from the correct solution written by us and the output distance matrix from the attempt by GPT-4). The prompts used are in the following template ("[OBJECT]" and "[DESCRIPTION]" are placeholders for the name of the object used, and a description of the object):

Please provide a model solution in Python for the following Ray-Object Intersection programming question. Please only include the function definition and no other text in the response.

In this exercise you need to write a function computing the intersection between a ray and a(n) [OBJECT] in a 3D scene.

The ray is defined by its origin O and direction D, and it is represented as R(t) = O + tD, where t is a scalar indicating the distance of a point from the ray origin. The function takes two parameters: 'O' for ray origin and 'D' for ray direction. They are both tuples of 3 floats in the form of (x, y, z). The function returns a float 't'. If an intersection exists, then 't' is the distance from the ray origin to the closest intersection. If no intersections exist, then 't' is -1 indicating the lack of intersections.

[DESCRIPTION]

You may assume that the ray always starts outside the object, and points where the ray grazes the object at one single point are considered intersections.

An example of an object description is shown below.

"The object is a sphere cut by a plane. The sphere is centered at (0, 0, 0) with radius 1. The plane is x + y = 1. The smaller portion of the sphere is removed. The sphere is solid, hence the intersections between the ray and the cross-section at the cut need to be considered."

All code (including objects, object descriptions, and model solutions of ROI programming questions) used in this study can be found in the link provided in Section 7.

4. Results

The results of the experiments described in Section 3 are shown in Table 1 and Table 2. Please note that the percentages overlap in Table 1, they are also listed in decreasing order since completeness implies uniqueness and uniqueness implies coherence. But this is not the case for Table 2, where the proportions are mutually exclusive and add up to 100%.

5. Discussion

5.1. Problem Design

GPT-4 produced an impressive rate of 84% for writing coherent ROI programming questions without contradictions according to the given prompts, most of which contained descriptions that uniquely describe objects. Although some questions consisted of minor errors, such as inaccurate facts and test cases, 67% of all questions were up to usable standards, and could be used directly in an assessment. Most minor errors could also be recognised and corrected with minimal human input, such as changing the output of a test case from True to False, hence the overall performance of GPT-4 on the task of designing an ROI programming question was remarkably satisfactory.

Other than the observations described in Table 1, there are some general characteristics exhibited by GPT-4 in its responses. The ROI programming questions written by GPT-4 tend to be much longer and more detailed than those written by instructors, but they can sometimes include incorrect facts. They also tend to include hints to guide students to the right line of thinking, but the hints may sometimes give away too much information that the questions are made too easy, such as providing the entire pseudocode. Some generated questions can also contain test cases despite not being explicitly asked, but they are prone to be incorrect (the expected output is incorrect). There is not much variety to the test cases, as rays tend to be aimed directly at the centre of the object in most generated test cases. Although the test cases do not contain much diversity, the generated questions can be much more diverse even prompted using the same object. For the same object, GPT-4 was observed to generate questions set not only in 3D but in 2D too. The number of parameters for the function was also variable, as some questions only take parameters associated with the ray whereas the object is fixed, and some questions take parameters associated with both the ray and the object, where the object can be moved according to the input values.

Object	Coherent	Unique	Complete	Selected Errors
Sphere	10	10	7	Two questions contained some minor mistakes in the generated test cases and one question contained an incorrect formula.
Cube	10	10	8	One question used the term "t-interval" incorrectly and one question incorrectly stated the number of parameters for the function.
Square	9	7	7	Two responses were not unique since there was no mention of the aligned axis (which implies possible rotation around the centre).
Cylinder	9	9	7	One question stated that the cylinder is infinitely long on the y-axis but the function takes a "height" parameter for the cylinder.
Two Spheres	7	7	7	One question simply asked the student to perform a Ray-Sphere Intersection twice and give two separate solutions.
Disc	10	10	8	One question asked the student to return a "special value" if there were no intersections but did not specify the value.
Quad with Circular Hole	6	6	4	Two questions were set in 2D, which makes the circular hole useless as light in 2D must pass through the quad to intersect through the hole. One question made the circular hole opaque.
Plane with Triangular Hole	9	9	8	One question stated that the plane does not intersect the triangle. One question used the term "homogeneous coordinates" incorrectly.
Ellipsoid	7	7	5	Two questions described the ellipsoid using an equation and implied that it was centred at (0, 0, 0), but the functions take "centre" as a parameter. Some questions also included mistakes in the test cases.
Cut Sphere	7	6	6	One question implied that the centre of the sphere was (0, 0, 0), but the function takes "centre" as a parameter. One question described a sphere with a cylindrical hole instead of a sphere sliced by a plane.
Total	**84%**	**81%**	**67%**	

Table 1: *The counts (out of 10) of ROI programming questions using various objects written by GPT-4 that fit the criteria outlined in Subsection 3.2, and a selection of errors that GPT-4 made in these questions.*

Given that GPT-4 was capable of generating logical and complete ROI programming questions given a large variety of different objects with a high success rate, we can deduce that GPT-4 can be quite helpful for instructors when writing these questions. Although GPT-4's responses are generally imperfect, they serve as an excellent starting point for writing such questions. Instructors can save a lot of time by modifying and adding to GPT-4's responses instead of starting from scratch. GPT-4's responses are also often detailed so it can remind the instructor of something they may have missed and excluded from the question previously. It is important to manually ensure the quality of questions generated by GPT-4 instead of automating the generation process since the responses can be flawed (e.g., may include incorrect facts), so they should be reviewed before they are used.

5.2. Problem Solving

As opposed to the impressive results achieved by GPT-4 for writing ROI programming questions, GPT-4 did not achieve such high performance for solving ROI programming questions, only achieving a success rate of 42%. The drop in performance may be due to the more intensive technicality and visual processing required to perform an ROI computation than to merely describe one.

GPT-4 achieved a high success rate with spheres, squares, and discs (circles), which are frequently used objects in Ray Tracing. However, GPT-4 failed to solve any questions using cylinders or cut spheres, which are the only objects used with both curved faces and flat faces. From inspecting the output images, we can infer that GPT-4 only considered the curved faces of the objects, and failed to account for the flat faces in all of its attempts for cylinders and cut spheres. For other objects, GPT-4 only answered the questions correctly in less than half of its attempts due to various errors.

As shown in Table 2, a common error that GPT-4 made is not removing intersections that are located behind the camera. Many of GPT-4's attempts consisted of representing the ray and the object using parametric equations, and solving the equations simultaneously to compute the value of "t". However, the solutions of these systems of equations may include negative values, and a negative value for "t" means that the ray has to travel backwards from the camera to reach the intersection, which does not occur in Ray Tracing applications, hence such intersections should be discarded. GPT-4 failed to take this into consideration in many of its attempts, which caused this error to occur frequently. Some other common errors include the use of incorrect algorithms, not considering the special case of dividing by zero, incorrect return values, etc.

Since GPT-4 only achieved 42% in Solving ROI programming questions and could not solve 2 of the 10 questions, it may not be a reliable source for model solutions for ROI programming questions. It can serve as a starting point for instructors, but it may not be as easy to identify and amend errors in code, as the underlying algorithm needs to be understood first, hence this approach can be counterproductive, as it may take more effort to correct the generated code than to start from scratch.

Object	Correct	Incorrect	Error	Selected Errors
Sphere	9	1	0	One attempt failed a test case where the object was behind the camera but was still shown, i.e., negative solutions were not discarded.
Cube	4	1	5	Many errors were caused due to zero division errors, indicating that the special case of dividing by zero was not considered. The incorrect attempt was due to an undiscarded negative solution.
Square	7	3	0	All incorrect attempts were due to undiscarded negative solutions.
Cylinder	0	9	1	No attempts considered the top and bottom faces of the cylinder, hence they all failed the "bird's-eye view" test cases (Figure 3 top-left). Some undiscarded negative solutions. The one error was due to a timeout.
Two Spheres	4	3	3	One attempt placed one of the spheres in front of the other despite the spheres being at the same distance (Figure 3 top-right). Some attempts returned NaN as the value for "t". Some undiscarded negative solutions.
Disc	8	1	1	The incorrect attempt was due to an undiscarded negative solution. The attempt that produced an error was due to an incorrect return type.
Quad with Circular Hole	2	4	4	Two attempts produced incorrect dimensions for the quad, one of which also failed to produce the circular hole. One attempt generated an arbitrary shape that does not resemble the requested object (Figure 3 middle-left). One attempt returned no intersections for the entire area.
Plane with Triangular Hole	4	5	1	There were attempts with misaligned planes and misplaced triangles (Figure 3 middle-right). Some undiscarded negative solutions.
Ellipsoid	4	3	3	Some attempts produced ellipsoids of incorrect dimensions and locations. One attempt produced a sphere. One error was produced by the incorrect initialisation of an array (GPT-4 initialised the radii array as [2, 2, 3, 4] instead of the correct [2, 3, 4].)
Cut Sphere	0	10	0	Many different causes of errors. Some spheres were hollow, some were sliced twice, some attempts removed the larger portion of the sphere, and some output images contained arbitrary patterns of black dots due to floating point inaccuracies (Figure 1 and Figure 3 bottom).
Total	**42%**	**40%**	**18%**	

Table 2: *The counts (out of 10) of correct, incorrect, and erroneous solutions from GPT-4's attempts at solving ROI programming questions, and a selection of errors that GPT-4 made in these attempts.*

5.3. Limitations

We would like to acknowledge several limitations of this study.

The three criteria to assess the quality of generated ROI programming questions require manual judgment and grading, which is a subjective process, as every instructor may have different perceptions of an adequate question, and whether a question satisfies a criterion. Therefore, we chose the criteria based on some specific essential qualities of every adequate ROI programming question, making it easier to judge whether a question fits the criteria. We also tried our best to be unbiased when grading the questions. It is impossible to objectively grade the quality of questions, and our scores only serve as rough estimates for reference. Additionally, we compiled a selection of objective errors and characteristics to provide the reader with more context and perspective.

The 10 attempts by GPT-4 to solve each ROI programming question were generated independently. However, in practice, when programmers receive incorrect outputs, they tend to revisit their code and make modifications, but we have deprived GPT-4 of opportunities to be informed of the errors and modify its solutions accordingly. Providing GPT-4 with the chance to reassess and modify its solutions (i.e., chain-of-thought prompting) may potentially yield a better representation of its problem-solving capabilities.

The technology of LLMs is growing rapidly, which means that the performance of GPT-4, or other LLMs, in the tasks outlined in this study may continue to improve, and the results obtained in this paper may become outdated soon. Fortunately, the experiments used in this paper can mostly be automated, and it would not be difficult to rerun these experiments to detect any improvements in the future.

6. Conclusion and Future Work

In this study, we ran experiments to evaluate the usefulness of GPT-4 when assisting with the writing of ROI programming questions, which is a common exercise for students learning Ray Tracing, a fundamental concept in CG. The results indicate that GPT-4 may be helpful when writing ROI programming questions, as an impressive 67% of its generated questions can be used in assessments verbatim. However, it performed more poorly for writing model solutions, and an accuracy of only 42% indicates that it may not be as applicable for writing model solutions for ROI questions. Due to the low accuracy of GPT-4 in solving ROI questions, students

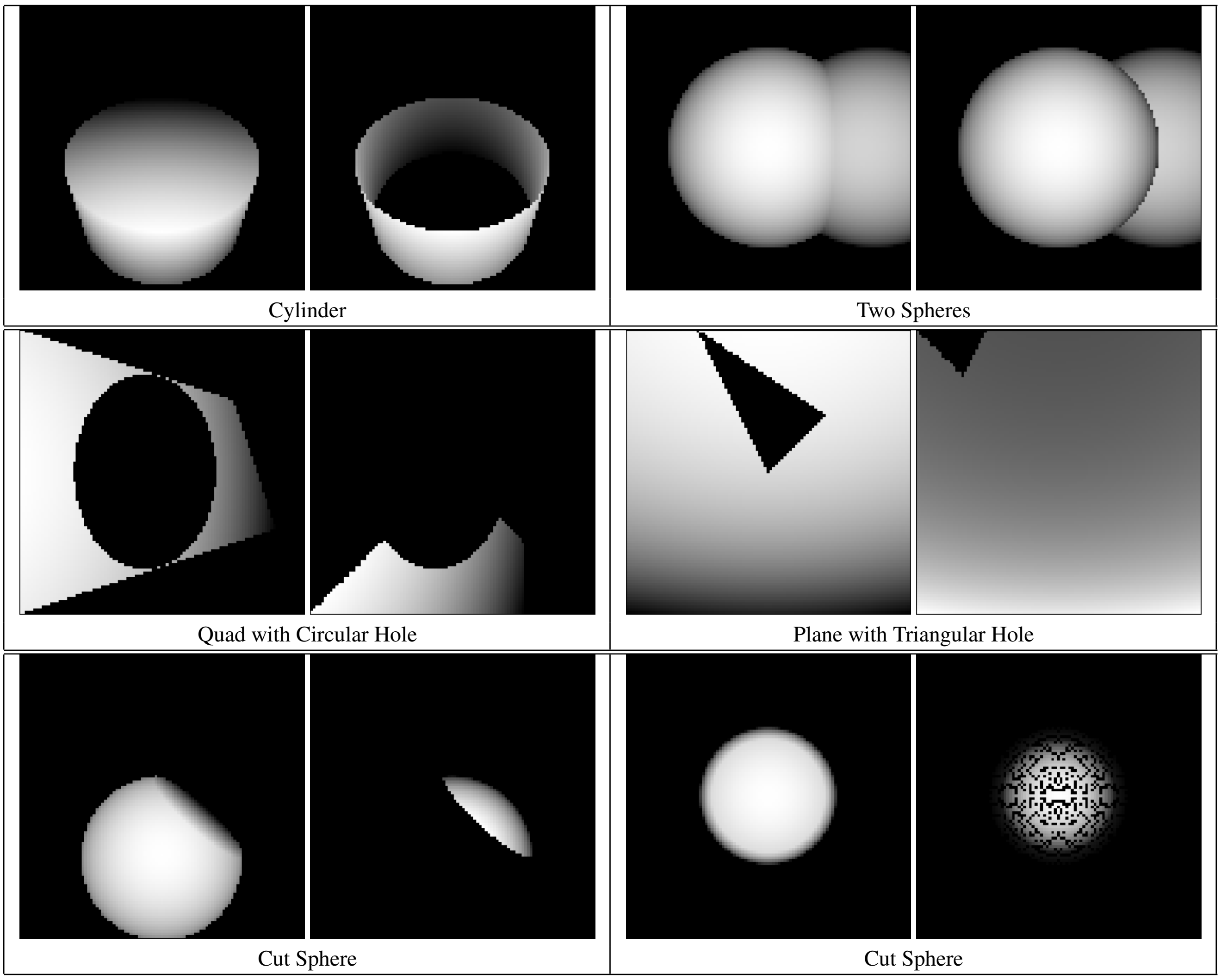


Figure 3: *A selection of output images rendered using ROI functions. Within each cell of the table, the left image shows the correct output image for an ROI programming question from its model solution written by us, and the right image shows the output image produced from an incorrect solution written by GPT-4 for the same question, with the associated object labelled below. More details of these errors can be found in Table 2.* ***Please note that the colour of a pixel denotes depth and not brightness****, i.e., the darker a pixel is, the further it is from the camera (ray origin), and vice versa. A completely dark pixel indicates the lack of intersections.*

should be warned that the solutions LLMs provide for similar questions may often be incorrect (results from this study can be used as examples), and the best form of help they can receive is still through trusted sources, such as instructors and textbooks.

The templates for the prompts used in this study may also serve as starting points for prompts used to generate similar material, and they may be modified to fit any specific requirements of the user. For example, a possible method to reduce minor errors for "Problem Design" is to include "Do not generate test cases." in the prompts used.

There are several possible avenues for future work. Although Python is the language that is the most prominently used with GPT-4, C++ is the most used language in the domain of CG, hence the training data of GPT-4 related to CG may contain more C++ code than Python code. Evaluating and comparing the performance of GPT-4 in both languages in tasks related to CG may provide a wider perspective on its capabilities in CG. Additionally, as mentioned in Subsection 5.3, using chain-of-thought prompting may further improve the performance of GPT-4, and the results from this study may be used as a benchmark for comparison if similar experiments are to be run in the future.

7. Resources

All code and generated content used in this study can be found under this GitHub repository: `https://github.com/TFPlusPlus/GPT-Ray-Tracing`

References

[AAAA*23] ABD-ALRAZAQ A., ALSAAD R., ALHUWAIL D., AHMED A., HEALY P. M., LATIFI S., AZIZ S., DAMSEH R., ALRAZAK S. A., SHEIKH J., ET AL.: Large language models in medical education: Opportunities, challenges, and future directions. *JMIR Medical Education 9*, 1 (2023), e48291. 3

[AQ23] AI4SCIENCE M. R., QUANTUM M. A.: The impact of large language models on scientific discovery: a preliminary study using gpt-4. *arXiv preprint arXiv:2311.07361* (2023). 2, 3

[BWF17] BALREIRA D. G., WALTER M., FELLNER D. W.: What we are teaching in Introduction to Computer Graphics. In *EG 2017 - Education Papers* (2017), Bourdin J.-J., Shesh A., (Eds.), The Eurographics Association. doi:10.2312/eged.20171019. 1, 2

[DKG23] DENNY P., KUMAR V., GIACAMAN N.: Conversing with copilot: Exploring prompt engineering for solving cs1 problems using natural language. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (2023), pp. 1136–1142. 3

[dlHvWFK22] DE LA HOUSSAYE W. A. V., VAN WEZEL C. S., FREY S., KOSINKA J.: Virtual ray tracer. In *Eurographics 2022-education papers*. The Eurographics Association, 2022. 3

[DLRTH11] DENNY P., LUXTON-REILLY A., TEMPERO E., HENDRICKX J.: Codewrite: supporting student-driven practice of java. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (2011), pp. 471–476. 1

[DPB*24] DENNY P., PRATHER J., BECKER B. A., FINNIE-ANSLEY J., HELLAS A., LEINONEN J., LUXTON-REILLY A., REEVES B. N., SANTOS E. A., SARSA S.: Computing education in the era of generative ai. *Commun. ACM 67*, 2 (jan 2024), 56–67. URL: https://doi.org/10.1145/3624720, doi:10.1145/3624720. 3

[FADB*22] FINNIE-ANSLEY J., DENNY P., BECKER B. A., LUXTON-REILLY A., PRATHER J.: The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference* (2022), pp. 10–19. 3

[FADLR*23] FINNIE-ANSLEY J., DENNY P., LUXTON-REILLY A., SANTOS E. A., PRATHER J., BECKER B. A.: My ai wants to know if this will be on the exam: Testing openai's codex on cs2 programming exercises. In *Proceedings of the 25th Australasian Computing Education Conference* (2023), pp. 97–104. 3

[FDW*24] FENG T. H., DENNY P., WUENSCHE B., LUXTON-REILLY A., HOOPER S.: More than meets the AI: Evaluating the performance of gpt-4 on computer graphics assessment questions. In *Proceedings of the 26th Australasian Computing Education Conference* (2024), pp. 182–191. 3

[GFE*12] GRIBBLE C., FISHER J., EBY D., QUIGLEY E., LUDWIG G.: Ray tracing visualization toolkit. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2012), pp. 71–78. 3

[Gla89] GLASSNER A. S.: *An introduction to ray tracing*. Morgan Kaufmann, 1989. 2

[HAMD23] HICKE Y., AGARWAL A., MA Q., DENNY P.: AI-TA: Towards an intelligent question-answer teaching assistant using open-source llms. *arXiv preprint arXiv:2311.02775* (2023). 2, 3

[IKSZ03] IONES A., KRUPKIN A., SBERT M., ZHUKOV S.: Fast, realistic lighting for video games. *IEEE computer graphics and applications 23*, 3 (2003), 54–64. 2

[KBGA23] KATZ D. M., BOMMARITO M. J., GAO S., ARREDONDO P.: Gpt-4 passes the bar exam. *Available at SSRN 4389233* (2023). 2, 3

[LH16] LOBB R., HARLOW J.: Coderunner: A tool for assessing computer programming skills. *ACM Inroads 7*, 1 (2016), 47–51. 2, 3

[LSSD23] LIFFITON M., SHEESE B. E., SAVELKA J., DENNY P.: Codehelp: Using large language models with guardrails for scalable support in programming classes. In *Proc. Koli Calling Conf. on Comp Ed Research* (New York, NY, USA, 2023), Koli Calling '23, ACM. URL: https://doi.org/10.1145/3631802.3631830. 2, 3

[MIT12] MITOPENCOURSEWARE: 2011 | Computer Graphics | Electrical Engineering and Computer Science | MIT OpenCourseWare. https://ocw.mit.edu/courses/6-837-computer-graphics-fall-2012/resources/mit6_837f12_2011_final/, 2012. [Accessed 31-12-2023]. 1

[NKM*23] NORI H., KING N., MCKINNEY S. M., CARIGNAN D., HORVITZ E.: Capabilities of gpt-4 on medical challenge problems. *arXiv preprint arXiv:2303.13375* (2023). 2, 3

[NS02] NIKOLIC D., SHENE C.-K.: Graphicsmentor: A tool for learning graphics fundamentals. *ACM SIGCSE Bulletin 34*, 1 (2002), 242–246. 3

[Ope23] OPENAI: GPT-4. https://openai.com/gpt-4, 2023. [Accessed 16-12-2023]. 2

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., ET AL.: Optix: a general purpose ray tracing engine. *Acm transactions on graphics (tog) 29*, 4 (2010), 1–13. 2

[Ped19] PEDDIE J.: *Ray tracing: a tool for all*. Springer, 2019. 2

[PSKD23] PURSNANI V., SERMET Y., KURT M., DEMIR I.: Performance of chatgpt on the us fundamentals of engineering exam: Comprehensive assessment of proficiency and potential implications for professional environmental engineering practice. *Computers and Education: Artificial Intelligence* (2023), 100183. 3

[Rus99] RUSSELL J. A.: An interactive web-based ray tracing visualization tool. *Undergraduate Honors Program Senior Thesis. Department of Computer Science, University of Washington 2* (1999). 3

[SAA*23] SAVELKA J., AGARWAL A., AN M., BOGART C., SAKR M.: Thrilled by your progress! large language models (gpt-4) no longer struggle to pass assessments in higher education programming courses. *arXiv preprint arXiv:2306.10073* (2023). 3

[SDHL22] SARSA S., DENNY P., HELLAS A., LEINONEN J.: Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1* (2022), pp. 27–43. 2, 3

[She02] SHENE C.-K.: Raytracing as a tool for learning computer graphics. In *32nd Annual Frontiers in Education* (2002), vol. 3, IEEE, pp. S4G–S4G. 3

[Sta22] STANFORD: Stanford CS248: Interactive Computer Graphics Participation Exercise 4. https://gfxcourses.stanford.edu/cs248/winter22content/static/pdfs/exercise4.pdf, 2022. [Accessed 31-12-2023]. 1

[SWLR19] SUSELO T., WÜNSCHE B. C., LUXTON-REILLY A.: Technologies and tools to support teaching and learning computer graphics: A literature review. In *Proceedings of the Twenty-First Australasian Computing Education Conference (ACE 2019)* (Sydney, NSW, Australia, 2019), ACM, pp. 96–105. doi:10.1145/3286960.3286972. 1

[TZSZ23] TU X., ZOU J., SU W. J., ZHANG L.: What should data science education do with large language models? *arXiv preprint arXiv:2307.02792* (2023). 3

[UCB23] UCBERKELEY: UC Berkeley CS184/284a Part 1: Ray Generation and Scene Intersection. https://cs184.eecs.berkeley.edu/sp23/docs/proj3-1-part-1, 2023. [Accessed 31-12-2023]. 1

[VGV*20] VITSAS N., GKARAVELIS A., VASILAKIS A.-A., VARDIS K., PAPAIOANNOU G.: Rayground: An online educational tool for ray tracing. In *Eurographics (Education Papers)* (2020), pp. 1–8. 3

[Whi79] WHITTED T.: An improved illumination model for shaded display. In *Proceedings of the 6th annual conference on Computer graphics and interactive techniques* (1979), p. 14. 2

[YH23] YEADON W., HARDY T.: The impact of ai in physics education: A comprehensive review from gcse to university levels. *arXiv preprint arXiv:2309.05163* (2023). 3