明治大学大学院

２０２４年度　修　士　論　文

論文題名　PP-Checker: 大規模言語モデルとの協調による
プログラミング教育円滑化手法と実践

先端数理科学研究科　先端メディアサイエンス専攻

指導教員名　中村　聡史

本人氏名　関口　祐豊

2024年度 修士学位請求論文

# PP-Checker: 大規模言語モデルとの協調による プログラミング教育円滑化手法と実践

明治大学大学院先端数理科学研究科
先端メディアサイエンス専攻

関口祐豊

Master's Thesis

# PP-Checker: A Method and Practice for Facilitating Programming Education through Collaboration with Large Language Models

Frontier Media Science Program,

Graduate School of Advanced Mathematical Sciences,
Meiji University

Yuto Sekiguchi

# 概要

プログラミング教育は，アルゴリズム的思考や問題解決能力の育成において重要な役割を果たすが，学生と教員・TA の人数に偏りがある場合，課題の迅速なフィードバックと採点の負担が課題となっている．従来の自動採点システムは，テストケースに基づく固定的な評価を行うものが一般的であり，動的要素や創造的な解答が含まれるクリエイティブコーディングに特化したプログラミング課題では，十分に対応できなかったり，テストケースの作成が面倒であったりするという問題があった．

本研究では，これらの課題を解決するために，大規模言語モデル（LLM）を活用した自動採点システム「PP-Checker」を提案する．PP-Checker は LLM を利用することで，正確性，即時性，曖昧性の 3 つの観点を重視した採点手法を導入した．

2024 年度の春学期および秋学期前半における実験運用の結果, 16 回の講義で PP-Checker を用いた 13,035 回の提出があり，PP-Checker 導入後，学生の再提出までの平均時間が 2023 年度の約 30.9 分から約 12.3 分に短縮された．これは，学生が LLM によるフィードバックを受けて自発的に修正を行う機会が増加したことにもつながっていると考えられる．さらに，プロンプト変更機能を導入することで，採点精度は平均 3.7%向上し，特定の課題では初期精度 33.3%が 72.9%まで向上するなど，LLM を活用したプロンプト設計の有用性を確認した．

今後は，運用コストの最適化とさらなる採点精度向上のため，プロンプトのトークン数削減やキャッシュ機構の導入を進める．また，視覚的な課題の採点精度向上を目的として実行画面の画像をプロンプトに取り込む方法や，学生が自らプロンプトを設計し採点精度を競うプロンプトコンペティションの導入も検討する．さらに，他分野への PP-Checker の適用を進めることで，LLM による学習支援システムとしての幅広い利用可能性を探求する．数年規模での効果検証を通じて，学生の学習促進や TA の業務効率化への貢献を定量的に示し，新たな学習支援システムとして成長させることを目指す．

# Abstract

Programming education plays a crucial role in cultivating algorithmic thinking and problem-solving skills. However, when there is a disparity between the number of students and faculty or teaching assistants (TAs), providing immediate feedback and handling grading workload becomes a significant challenge. Traditional automated grading systems commonly rely on static assessments based on predefined test cases. These systems, however, may be inadequate for programming assignments in creative coding which often involve dynamic elements and creative responses. This inadequacy can lead to insufficient support for students and make test case creation burdensome for educators.

This paper proposes an automated grading system, "PP-Checker," which leverages large language models (LLMs) to address above challenges. By integrating LLMs, PP-Checker introduces a grading method that prioritizes accuracy, immediacy, and flexibility.

In a pilot deployment during the spring and early autumn semesters of 2024, PP-Checker handled 13,035 submissions across 16 lectures. Following the introduction of PP-Checker, the average time for students to resubmit assignments decreased significantly, from approximately 30.9 minutes in 2023 to about 12.3 minutes. This suggests an increased willingness among students to make improvements based on feedback provided by LLMs. Furthermore, the effectiveness of prompt engineering in grading was confirmed by the fact that implementation of prompt adjustment functionality enhanced grading accuracy by an average of 3.7%, with initial accuracy rising from an initial 33.3% to 72.9% for specific assignments.

In future work, we aim to optimize operational costs and further improve grading accuracy by reducing the token usage in prompts and implementing caching mechanisms. Additionally, to enhance the accuracy of grading for visual assignments, we intend to incorporate screenshots of execution results into prompts. We also plan to introduce a "Prompt Competition" feature, allowing students to design their own prompts to compete in grading accuracy.

Finally, by extending PP-Checker's application to domains beyond programming education, we will explore its broader potential as an LLM-powered learning support system. Through multi-year evaluations, we will quantitatively demonstrate PP-Checker's effectiveness in promoting student learning and improving TA efficiency, and we aim to develop PP-Checker into a new educational support system.

# 目 次

# 第1章　はじめに

## 1.1　プログラミング教育の現状と課題

プログラミング教育は，探究力やアルゴリズム的思考，論理的思考力を養うのに重要な役割を果たしており，国内外の教育機関において幅広く導入が進んでいる [48,84,90]．しかし，大学における CS（Computer Science）1 や CS2 といった入門的なプログラミング教育では，多数の学生を対象とする大規模な講義形態が主流であり，効果的な教育支援と学生サポートの両立が課題となっている [49,99]．

こうした大規模教育環境では，数十人から数百人規模の学生が同時に学習を進めるため，TA（Teaching Assistant）が配置され，主に課題採点と学生からの質問対応を行う役割を担っているが [98]，学生数に対してサポート体制が不足している場合，十分な教育効果を発揮できない．例えば，著者が所属する明治大学総合数理学部先端メディアサイエンス学科では，毎年 8〜10 名程度の TA と 4 名の教員が協力し，120 名以上の学生が受講する必修のプログラミング演習を担当しており，講義や課題提示，質問対応や課題採点などを行っている．しかし，学生・TA・教員の人数比に偏りがある場合，サポート体制が不十分となり，講義運営が円滑に進まないことがある．

## 1.2　人数比が課題採点業務に与える影響

プログラミング演習講義では，学生が授業内で提示された複数の課題に限られた時間内で取り組むことが一般的である．著者が所属する学科のプログラミング演習講義（100 分 2 コマ）でも，毎講義 4〜6 問の課題を提示し，学生は与えられた課題に取り組んでいる．こうした課題に対し，迅速な採点とフィードバックの繰り返しが学生の学習意欲の向上につながることが知られている [14,97]．

しかし，TA や教員の人数に対して学生の人数が多い場合，図 1 のような業務の流れとなり，TA と教員は随時発生する質問対応を行いながら，膨大な課題の採点を並行して進める必要がある．随時発生する質問対応を行いつつ課題の採点を行うのは容易ではなく，全学生の課題を迅速に採点できない [10,46,47,71]．こうした採点の遅延やフィードバッ

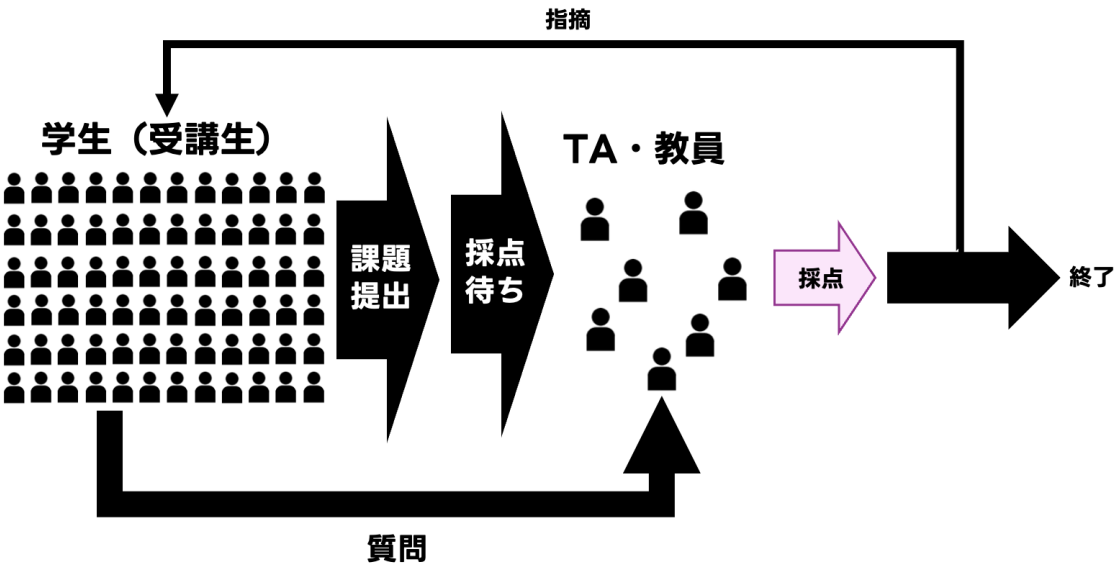



図 1: 課題採点の主な流れ．採点が遅れると採点待ちの学生が増えたり，質問対応へのリソースが割けなかったりするという問題点がある [10,71].

クの不足は，学生の理解度や学習意欲に悪影響を及ぼし，教育効果を低下させるリスクがある．

## 1.3 従来の自動採点手法の限界

自動採点は課題採点業務を効率化する方法の一つであり，フィードバックの迅速化だけでなく，学生の動機付けやスキル向上にも寄与することが知られている [19]．実際，既存の自動採点システムには，定義されたテストケースや短冊型問題 [88] に対応する手法 [44, 58, 67, 89, 95]，構文木を用いた自動採点手法 [91, 94] が数多く存在する．しかし，講義における課題の準備において，テストケースの設計は多くの時間と労力を要するため，教員にとって大きな負担となる．また，Processing [70] のような動的要素を含むクリエイティブコーディングを主としたプログラミング言語では，実行結果の視覚的な評価が求められるため，従来のテストケースベースの採点手法では対応が困難である．動的な出力を正確に評価するためには，表示結果に対する柔軟かつ高度な採点基準が必要となり，これまでの自動採点手法では対応しきれない課題が存在する．これらの背景から，動的要素を含むプログラムに対応可能な新たな自動採点システムの確立は，新規性と実用性の双方で高い価値を持つと考えられる．

## 1.4 大規模言語モデル（LLM）の利用

OpenAI社のGPT [11,61,63,68,69]やGoogle社のGemini [17], Anthropic社のClaude [5]など大規模言語モデル（LLM）の登場は，従来の自動採点手法が抱える課題を解決する一つの方法として期待されており [28, 34, 36, 43, 76, 82]，プログラミング演習講義支援の研究において，アルゴリズムやデータ構造に関する課題に対し，正確なフィードバックを生成することを目的とした研究が進められている [6, 56, 78, 92].

しかし，これらの研究では，課題に対する正確な解答やエラー修正に焦点が当てられており，直接的な解答を学生に提供することで，学生の自発的な問題解決能力を阻害するリスクが指摘されている．特に，ChatGPT [59] をはじめとする直接的な回答を示してしまう LLM を搭載した AI ツールは，教育機関での使用に関して懸念が高まっている [9, 16, 66, 73].

そのため，CodeAid [52] や ChotGPT [96]，GPT4HINTS-GPT3.5VAL [81] などの直接的な回答を示さずにフィードバックを提供するシステムが実運用されている．しかし，Processing のようなクリエイティブコーディング向けの言語では，LLM が幻覚（ハルシネーション） [29] を起こすリスクが高いことが指摘されており [76]，これら既存の手法やシステムが対応しきれていないという課題がある.

## 1.5 研究目的

本研究では，1.1〜1.4 節で論じた課題を踏まえ，プログラミング教育の円滑化を目的とする．本研究において「円滑化」とは，教育プロセスにおける学生，TA，教員間の障壁をなくし，学生がスムーズに学習を進められる環境を整えることを指す．これは単なる効率化にとどまらず，採点・フィードバック・修正を一つの流れとし，学習の循環を途切れさせないことや学生，TA，教員間の適切なコミュニケーションやサポートなども含む.

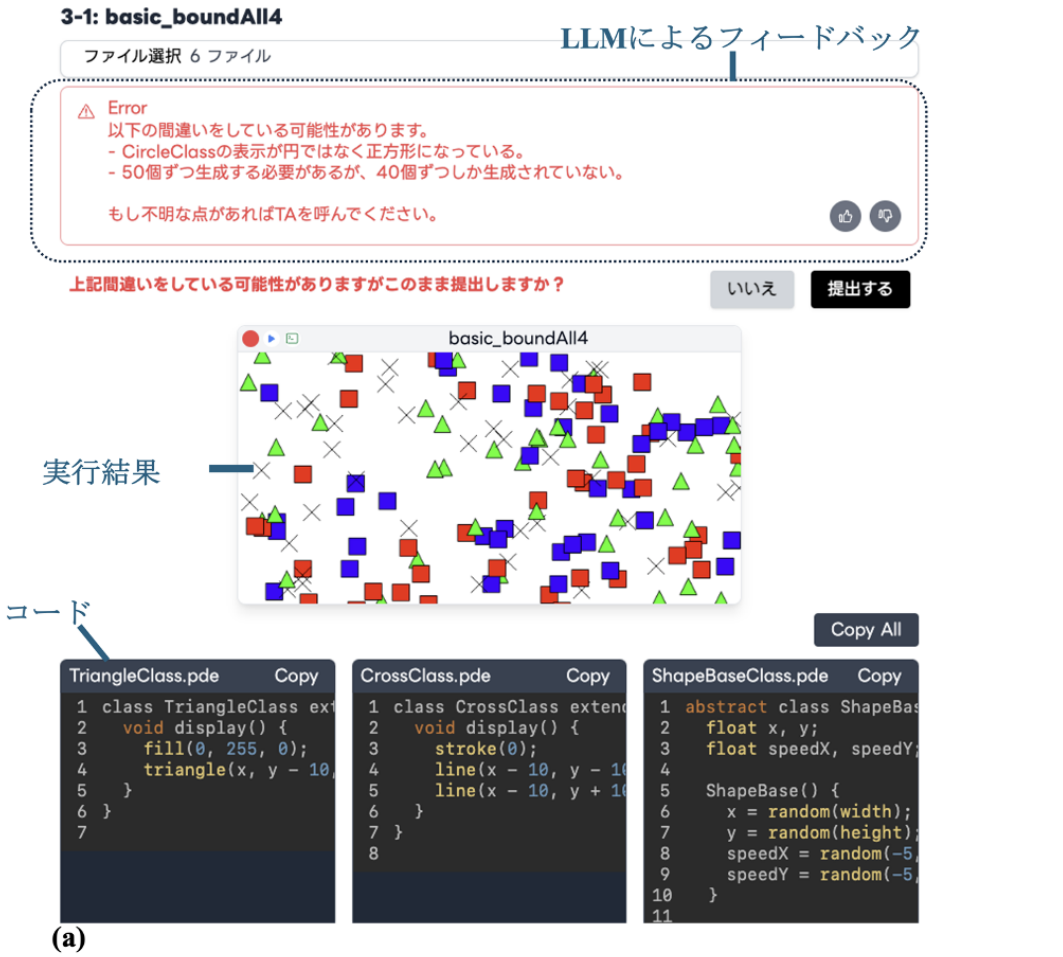
この円滑化の一環として，採点業務の効率化が重要な要素となる．そこで本稿では，プログラミング教育における課題採点の負担およびフィードバックの遅れが学習成果に与える影響に焦点を当て，これらの課題に対する解決策を検討する．具体的には，LLM の活用がインタラクティブなプログラミング環境における採点プロセスを効率化し，教育効果を向上させる可能性について検証する.

さらに，学生が提出したプログラムの迅速なフィードバックと高精度な採点を実現するため，人間と LLM が協働し，曖昧性を許容した半自動採点システム「PP-Checker（図 2）」を提案する．PP-Checker は，学生が提出したプログラムに対して LLM を用いて迅速な

フィードバックを提供することで，学生の自発的な修正と再提出を促進し，採点業務の効率化を目指す．PP-Checker が着目する即時性，採点精度，曖昧性の 3 つのアプローチに関しては 3 章で詳細に述べる．また，本システムをプログラミング演習講義で運用し，その有用性を検証する．具体的には，2024 年度春学期のプログラミング演習 I で 12 回，秋学期のプログラミング演習 II で 4 回，実際に運用し，学生の再提出までの時間や採点精度を評価することで，プログラミング教育における新たな自動採点手法の実現を目指す．

## 1.6 本稿の構成

本稿は，本章を含む全 8 章から構成される．まず本章で，現状のプログラミング教育の課題と LLM を利用した自動採点の可能性について述べた．これ以降，2 章では本研究の関連研究について述べる． 3 章では，人間と LLM の協調によってインタラクティブなプログラミング言語に対応し，課題採点業務の効率化を目的とした曖昧性を許容する自動採点システムである PP-Checker を提案する． 4 章では，PP-Checker の運用中に直面した課題を解決するための改良点について述べる． 5 章では，講義での運用の結果を定量的に評価し，学生，TA，教員それぞれの観点から定性的な分析も行う． 6 章では， 5 章の結果を踏まえた考察を学生，TA，教員，採点精度の観点から述べる． 7 章では，本研究の制約と今後の展望について述べ，最後に 8 章で本稿のまとめを行う．



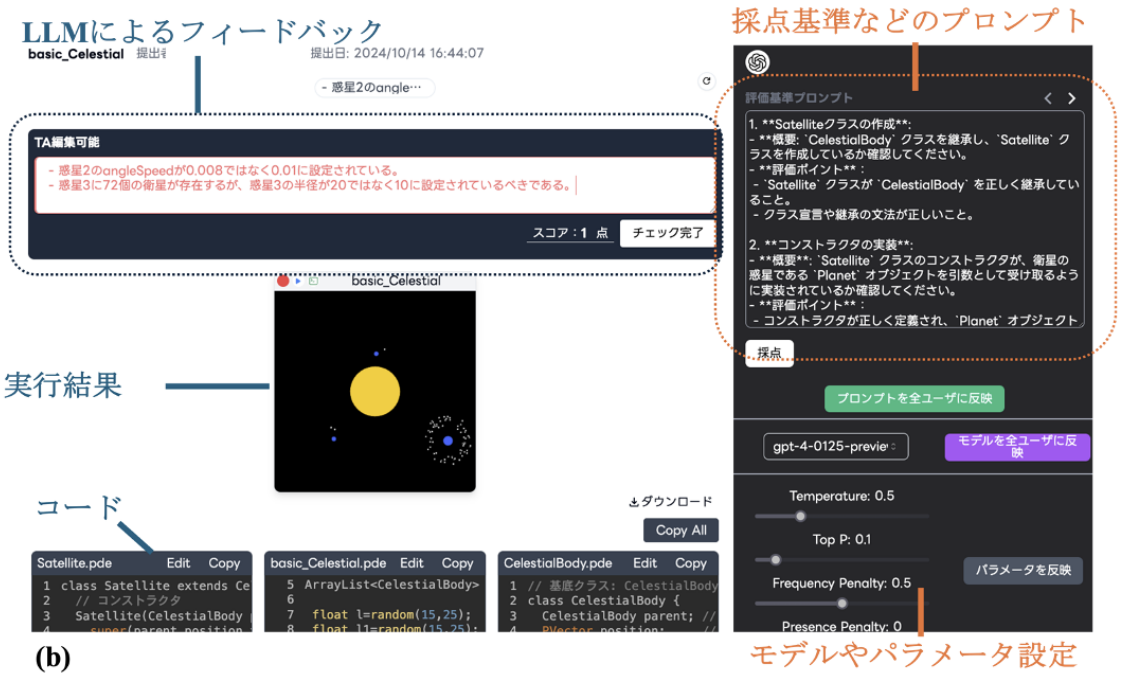




図 2: PP-Checker の 2 つの主要な画面．LLM による自動採点機能を備えた課題提出画面（a），手動チェック画面（b）．

# 第2章　関連研究

## 2.1　クリエイティブコーディングの教育的利点

ゲームやアニメーション [7,55]，音楽 [2,8,26,38] を利用したプログラミング支援手法は数多く提案されており，これらの手法は初学者の学生の関心を引き，学習のモチベーションを維持するために重要なきっかけを提供することが知られている [21,39]．特に，アニメーションやデジタルアートなど芸術的な表現やインタラクティブなデザインを目的とし，プログラミングを創作の手段として活用するアプローチのことを「クリエイティブコーディング [51,65]」という．クリエイティブコーディングの環境は，最初期には John Maeda による Design by Numbers [50] に端を発し，その後，Processing [70] や p5.js [1] といったツールなどが開発され，現在でも Coding Train [75] のような教育メディアを含め，幅広く利用されている [12,22,30,54,85]．

Greenberg ら [20] は，Processing が Java と比較して学習者のモチベーションを高めることを明らかにした．Salga ら [72] は Processing.js を提案し，教育やアートの分野での広範な利用可能性を示した．Terroso ら [79] は，非プログラマ向けに p5.js を用いた講義において，インタラクティブなプログラムが学生の関心や評価を高めることを明らかにした．また，特にプログラミング教育に興味を抱いていない学生にとっても，創造的または芸術的な課題が関心を高めると考えられている [53]．著者の学科の講義であるプログラミング演習Ⅰ・Ⅱにおいても，学生の関心を高めるため，錯視を取り入れた視覚的なプログラムの作成や，統計量の可視化，インタラクティブなゲームの作成といったように課題へのモチベーションを高める工夫を行っている[1]．

本研究では，クリエイティブコーディングのような創造的なプログラミング環境における採点業務に着目し，従来のテストケースに基づく採点方式では十分に評価できない点を補うために，LLM を活用し，曖昧性を許容する柔軟な自動採点手法の有用性を確かめる．

[1] 課題の詳細については，プログラミング演習Ⅰ・Ⅱのウェブサイト（https://lecture.nkmr.io）を参照．サイトでは，課題のサンプルコード，視覚的なプログラムや統計量の可視化，インタラクティブなゲーム作成などの課題内容に関する資料，および講義資料などのリソースがある．

## 2.2 LLM を活用したプログラミング支援

クリエイティブコーディングに対応した自動採点を実現するうえで，OpenAI 社の GPT [61, 63] や Google 社の Gemini [17]，Anthropic 社の Claude [5] などの大規模言語モデル（LLM）の活用が重要な役割を果たしている．従来の自動採点システムは固定されたテストケースに基づいてプログラムの正確性を評価することが多かったが，クリエイティブコーディングでは，表現やインタラクションが多様であり，従来の採点手法ではその創造性や動的な要素を正確に評価することが難しかった．しかし，LLM を活用することで，クリエイティブな要素を含むプログラムでも，柔軟かつ精度の高い採点を期待できる．さらに，LLM によるフィードバックは，従来のテストベースのフィードバックを補完する形で効果的であることが示されているが [23]，誤情報の提供やフィードバックの正確性には改善の余地があり [15, 32, 74]，この点を考慮することが求められている．

LLM がプログラミング学習に利益をもたらす可能性が示されている一方で，教育におけるこれらのシステムの不適切な利用に対する懸念も高まっている [41, 42, 62, 77, 87]．Kazemitabaar ら [40] は，AI コードジェネレータである CodeX [13] が初学者のプログラミング学習においてコードの完成率やスコアを向上させることを示したが，同時にツール依存を避け，スキル向上のための LLM の適切な使用が重要であると指摘していた．Jonsson ら [33] は，AI がプログラミング教育に有効である一方，予測不可能な振る舞いが学習体験に与える影響に注意が必要であることを示唆した．Pankiewicz ら [64] や Hellas ら [24]，Prather ら [28] は，プログラミング課題の達成において LLM によるフィードバックが有用であると示しつつ，LLM のフィードバックに依存するリスクを課題としてあげている．また，Xiao ら [86] は，LLM が生成するフィードバックが具体的なコード例を提示することで学習効果を高めることができると述べている一方で，AI 生成コードへの過度な依存が生じるリスクを指摘した．Abolnejadian ら [4] は，GPT [61, 63] を用いた教材カスタマイズが学生の理解度を向上させることを示しつつ，誤情報の訂正には教師の役割が不可欠であるとした．Kabir ら [37] も同様に，LLM の包括性と明瞭さを評価しつつ，人間と AI の協力の重要性を強調した．Lau ら [45] は，初学者向けのプログラミングコースの講師との対話で，学生が LLM に頼って答えを得ようとすることでコードを理解できなくなることを示した．さらに，NotebookGPT [18] という Jupyter Notebook 内で GPT から効果的なフィードバックを得られるシステムも開発されており，LLM への過度な依存によって学習者が自ら解決策を考える機会が減少する可能性が示された．KOGI [93] は，Google Colab 上で動作するクラウドベースの演習環境に LLM を統合した学習支援シス

テムであり，その有用性が示されている．しかし，NotebookGPT と同様に，LLM への過度な依存による学習意欲の低下といった課題が残されていると考えられる．

このように，LLM の活用はプログラミング教育において学習者の理解度やスキルの向上に寄与する一方で，依存のリスクや学習意欲に対する対策が課題となっている．この課題を踏まえた研究もいくつか行われている．Hou ら [25] は，プログラミング初心者向けに LLM を活用した CodeTailor を提案した．CodeTailor は，学生が誤ったコードを提出した際，そのコードを基にパーソナライズされた Persons パズルを生成することで，生成されたコードに依存することなく，正しいコード構造を学べるようサポートした．また，LLM を活用した模擬学生を用いることで，初学者の TA がリスクの小さい環境で多様な学生ペルソナに応じた指導スキルを磨く場を実現した GPTeach [35] のように，プログラミング教育の講義運営を支援している研究もある．さらに，CodeAid [52] や ChotGPT [96] のように，直接的な回答を提示せずにエラーを解決するため LLM を活用したシステム開発の研究も進展している．

これらの研究を踏まえ，PP-Checker は人間と LLM の協調による自動採点を実現し，課題採点業務の負担を軽減しつつ，迅速かつ適切なフィードバックを提供することを目指している．また，先行研究が，LLM を活用したプログラミング教育のサポートやエラー解決に焦点を当てている一方で，著者が提案する PP-Checker はクリエイティブコーディングにおける自動採点に着目しているという点で異なる．先行研究で明らかになった知見は，PP-Checker の設計において重要な指針となっている．また，PP-Checker は先行研究の成果を踏まえ，それを超える実用性を備えたシステムであると考えている．

# 第3章　PP-Checker

学生や TA，教員の人数比の偏りにより，TA や教員が全学生の質問対応をしつつ課題採点を行うことは負荷が高い．また，学生は迅速なフィードバックを求めており [14,97]，こうした即時性のニーズに応えることがプログラミング教育の質向上において重要な要素である．一方，Processing [70] などのインタラクティブなプログラミング言語は動的要素を含むことから，従来の採点手法では正確な評価が困難であるため，採点精度および曖昧性を考慮することが重要である．そこで，システムの実現にあたり，即時性，採点精度，曖昧性の観点に着目した仕組みについて次節で述べる．

## 3.1　提案手法

従来の手動採点ではフィードバックが遅れることが多く，学生の学習意欲や効果に悪影響を及ぼす可能性がある．そこで，LLM を活用して自動採点を行うことにより，フィードバックの即時性を大幅に向上させることを目指す．具体的には，この仕組みにより，学生は TA の採点を待たずに自身のコードの潜在的な問題点を早期に発見し，再提出することが可能になると考える．

先述の通り，自動採点では複雑なテストケースを作成し，提出されたコードが期待通り動作するかを評価する必要がある．しかし，視覚的・インタラクティブなプログラム課題においてはテストケースの準備に多くの時間と労力を要する．まず，グラフィカルな出力やインタラクティブな要素に対するテストケースの生成は，ユーザインタラクションや視覚的評価のシミュレーションを要求し，多大な時間とリソースを消費する．さらに，ユーザ入力の順序やタイミングの予測不可能性，および一部のインタラクティブ要素の非決定的な挙動により，あらゆる実行経路を事前に把握することはほぼ不可能である．したがって，従来のプログラミング課題とは異なり，期待される出力を明確に定義し，自動採点を行うことは困難である．

そこで，LLM とプロンプトを用いた採点手法を導入することで，テストケースの作成を不要とする．しかし，LLM の出力は完全ではなく，採点精度の向上にはプロンプト設計が重要であるため [100]，TA や教員が LLM のプロンプトをリアルタイムに変更できる

機能も導入する．具体的には，採点中にプロンプトを修正すると，更新されたプロンプトに基づいた採点が未採点の課題に即座に反映される仕組みを構築する．この機能により，教員や TA は，講義開始前に完璧なプロンプトを準備することができなかった場合でも，採点の精度を講義中にリアルタイムで調整することが可能になる．

LLM による自動採点は当然判定ミスも多くなると考えられる．そこで，学生に対して LLM の指摘を受け入れるか否かを選択する機会を提供する．これにより，学生は LLM の指摘を参考にしつつ，自らの判断を信頼して学習を進めることが可能になる．このアプローチは，LLM などシステムへの過度な依存を避け，自立性と判断力の強化にもつながることが期待される．

さらに LLM を用いた指摘において，具体的な正答を示してしまうと学生の問題解決能力を高めることができない．そこで，具体的な正答を示さないように LLM の出力結果を工夫することで，学生が自ら考えを深め，問題解決能力を高めること [80] を目指す．

## 3.2 システム概要

提案手法を講義において実運用することを目指し，課題提出，実行，採点，コード管理を一元化した Web アプリケーションとして実装を行った．本システムにより TA や教員は複数のツールを使い分けることなく，一つのプラットフォーム上で全ての課題採点業務を行うことができるため，業務の効率化が期待される．PP-Checker は，課題一覧画面，課題提出（自動採点）画面，提出物一覧画面，手動チェック画面の 4 つの主要な画面から構成されている．

### 3.2.1 課題一覧画面

課題一覧画面（図 3）は，PP-Checker のホーム画面であり，学生は各課題の進捗状況，提出状況，フィードバックの有無，過去に提出したコードを確認することが可能である．さらに，各課題項目をクリックすることで課題提出画面に移動し，コードの提出を行うことができる．これにより，学生は課題管理が容易になり，学習の効率が向上することが期待される．


図 3: 課題一覧画面

### 3.2.2 課題提出（自動採点）画面（学生用）

課題提出画面（図 2a）では，学生が自身のプログラムコードを提出し，LLM が採点を行う．具体的には，学生はコードを含むフォルダをアップロードすることで，提出前にその場で LLM による自動採点を受けることができる．LLM はアップロードされたコードを即座に採点し，誤りの可能性がある箇所を指摘するフィードバックを生成する．このフィードバックをもとに学生は自身のコードの潜在的な問題点を TA に提出する前に把握し，コードの修正や再提出が可能となる．また，この画面には，学生が提出したコードやその実行結果も表示され，コードが期待通りに動作しているかを視覚的に再確認することが可能である．このインタフェースによって，学習効果の向上とフィードバックの即時性を両立させる教育支援ツールとしての役割を強化している．

### 3.2.3 提出物一覧画面

提出物一覧画面（図 5）は，TA や教員が提出された課題を一覧で確認できるインタフェースである．この画面には，各課題の提出状況，採点の進捗，フィードバックの有無，どの

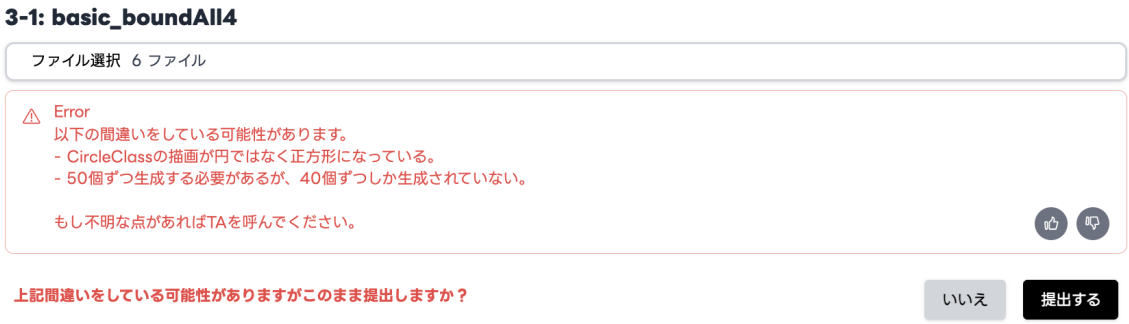



図 4: LLM によるフィードバックの例


図 5: 提出物一覧画面

TA がどの課題をチェックしているかがリアルタイムで表示される．また，各提出物の課題名をクリックすることで，手動チェック画面に移動し，LLM が行った初期採点結果を TA や教員が最終確認・修正できる仕組みとなっている．

### 3.2.4 手動チェック画面（TA・教員用）

手動チェック画面（図 2b）は，TA や教員が LLM による採点結果を確認し，必要に応じて補足や修正を行うことができる．画面左側には，LLM が生成したフィードバックを確認・修正できるインタフェースがあり，その下部には学生のコードと実行結果が表示される（図 6）．これにより，TA は学生のコードの動作を視覚的に確認しつつ，課題採点を行うことができる．また，画面上部には，他の学生の課題に対するフィードバックコメントを選択できるインタフェースが設置されており，これを再利用することで，入力


図 6: 手動チェック画面左側．LLM によるフィードバック確認およびコード動作確認，コード閲覧インタフェース．


図 7: フィードバックコメント選択インタフェース

の手間を省き，効率的な課題採点業務をサポートする（図 7）．画面右側では，LLM のプロンプトやモデルの変更，パラメータの調整を行うことができ，リアルタイムで採点精度を高めることが可能である（図 8）．また，プロンプトは TA や教員が容易に変更できるように，課題の満たすべき条件を箇条書き形式で記述可能とし，TA や教員が柔軟に調整できる設計とした．

## 3.3 実装

PP-Checker は，フロントエンドに Next.js（TypeScript），バックエンドに FastAPI，データベースに MySQL を用いて実装した．LLM は OpenAI 社の gpt-4-turbo-preview，gpt-4o-mini，gpt-4o，o1-mini，o1-preview の 5 つのモデルを選択可能で，リアルタイム

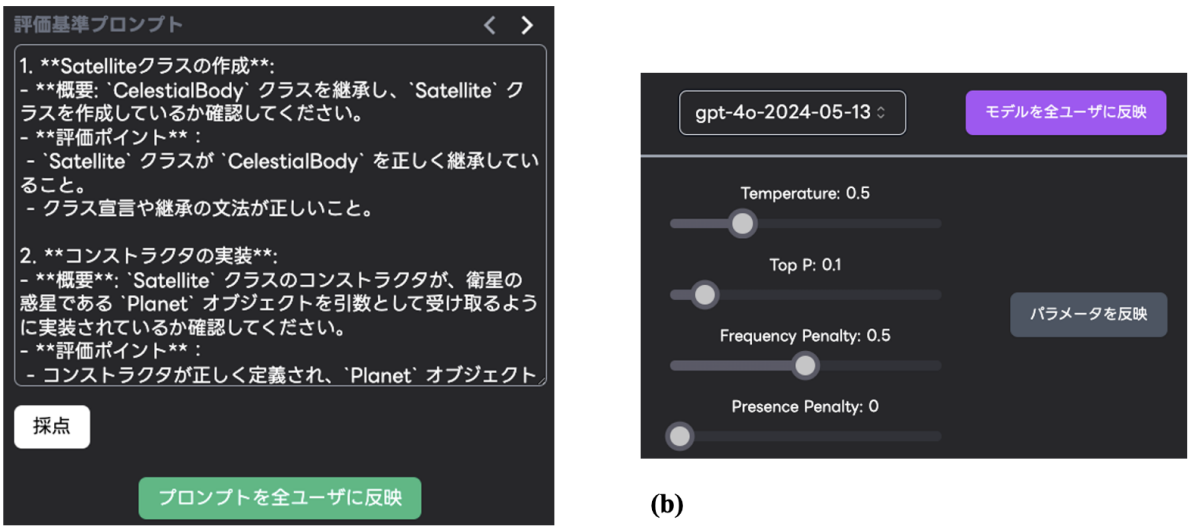



図 8: プロンプト変更インタフェース (a)．モデル，パラメータ変更インタフェース (b)．

通信に Socket.io を利用しており，学生と TA の迅速なやり取りを可能にしている．

Processing [70] のような動的要素を含むクリエイティブコーディングを主としたプログラミング言語は，静的なコードよりも複雑な評価が求められる．そこで，Processing.js [72] を用いて実行画面を描画し，コードの自動採点と視覚的な確認を可能にしており，マウスやキーボードの操作や学生ごとに異なるデザインを含む課題にも対応可能で，Processing.js で未対応の関数（`circle` や `enum` など）は自作関数で補完している．また，JavaScript と Processing 間の挙動の違いによるエラーを最小限に抑える設計も施している．なお，学生や TA の識別には，大学発行のメールアドレスでの認証を使用している．

# 第4章　運用から見えた問題とシステム改良

## 4.1　システム改良の重要性

システムの運用中に発生する問題は，特に教育支援ツールにおいては避けられないものである．教育現場では多様な環境や状況下で使用されるため，システムが直面する問題も多岐にわたる．こうした問題を単に発見するだけでなく，それに迅速かつ適切に対応することが，システムの信頼性やユーザ満足度を向上させる要因として極めて重要である [57]．また，教育支援システムにおける改善は，単なるバグ修正やパフォーマンスの向上に留まらず，学習効果の最大化や教育者，学習者双方にとっての使いやすさ向上も求められる．本章では，PP-Checker の運用を通じてどのように改善を進めてきたのかを述べる．具体的な運用の結果や評価については 5 章で示すが，まずは運用過程において直面した過程や，それに対する試行錯誤について述べる．

## 4.2　再実行の実装

プログラミング演習の講義において，乱数を用いる課題は，単回の実行結果のみから正確な手動評価を行うことが困難なケースがあった．その一例として，以下に示す課題（basic_Dice）がある．

課題例（**basic_Dice**）:

- 400x300 のウインドウを作成し，そのウインドウ内でマウスのボタンを押すたびに，A さんが 1〜6 の目が均等に出る 6 面ダイス，B さんが 1〜4 の目が均等に出る 4 面ダイスを振った結果を示すとともに，どちらが勝ったか（または引き分けたのか）を `print` や `println` を用いて「標準出力するプログラム」を作成せよ．
- 値の大きい方が勝ち，同じなら引き分けとする．

このような課題では，結果が実行ごとに変わるため，1 回の実行のみでは得られる情報

が限定的である．そこで，実行結果の表示エリアに再実行ボタンを設けることで，同一プログラムを容易に複数回実行できる仕組みを導入した．この改善により，TA はコードの挙動を反復的に確認し，ランダム性による変動を考慮したうえで，公平かつ安定した採点を行うことが可能となった．

## 4.3　複数ファイル提出の対応

プログラミング演習 II では，クラス設計を含む課題があり，複数のファイルに分割されたコードの正確な採点が求められる．具体的な課題（basic_boundAll4）は，その一例である．

**課題例（basic_boundAll4）**：

- `ObjectBase` クラスを継承し，赤色の○が動き回る `CircleClass`，青色の□が動き回る `SquareClass`，黒色の×が動き回る `CrossClass`，緑色の△が動き回る `TriangleClass` を作成せよ．
- またこれを利用して 50 個の○と，50 個の□と，50 個の×と，50 個の△が動き回るプログラムを作成せよ．
- ただし，その速度は x，y 方向それぞれ-5〜5 の実数値とし，○と□は壁で跳ね返り，×と△は跳ね返らずに反対側から出てくるようにせよ．
- なお，ArrayList を利用して実現せよ．

こうした要件に対応するため，PP-Checker ではフォルダ単位での提出を可能にし，複数ファイルから構成される提出物を一覧表示で管理および確認できるよう改良した．また，提出されたフォルダ内に含まれる Processing 拡張子（.pde）のファイル群を自動的に結合し，ひとつのプログラムとして実行および自動採点を行う仕組みを構築した．この機能拡張により，クラス定義やモジュール分割が求められる複雑な課題に対しても，シームレスな実行と自動採点が可能となり，PP-Checker の仕組みをより高度な課題にも適応することができるようになった．こうした改良は，複雑なコード構成を持つ学生の提出物においても，正確かつ効率的な評価が実現され，教育支援システムとしての信頼性の向上につながる．

## 4.4　マルチメディアへの対応

プログラミング演習 I の終盤では，画像ファイルや音声ファイルなどのマルチメディアリソースを扱う課題が提示される．以下に示す課題（basic_BoundSound）は，その一例である．

**課題例（basic_BoundSound）：**

- 背景画像がセットされた画面内を 2 つのキャラクタ（同じキャラクタでも別のキャラクタでもよい）が移動し，壁に当たると跳ね返るプログラムを作成せよ．なお，壁に当った時に効果音が鳴るようにせよ（効果音は同じでも別でも良い）．
- 速度は壁との衝突が視認可能なレベルとせよ．
- なお，キャラクタの中心座標の都合で微妙に壁にめり込んでも良い．

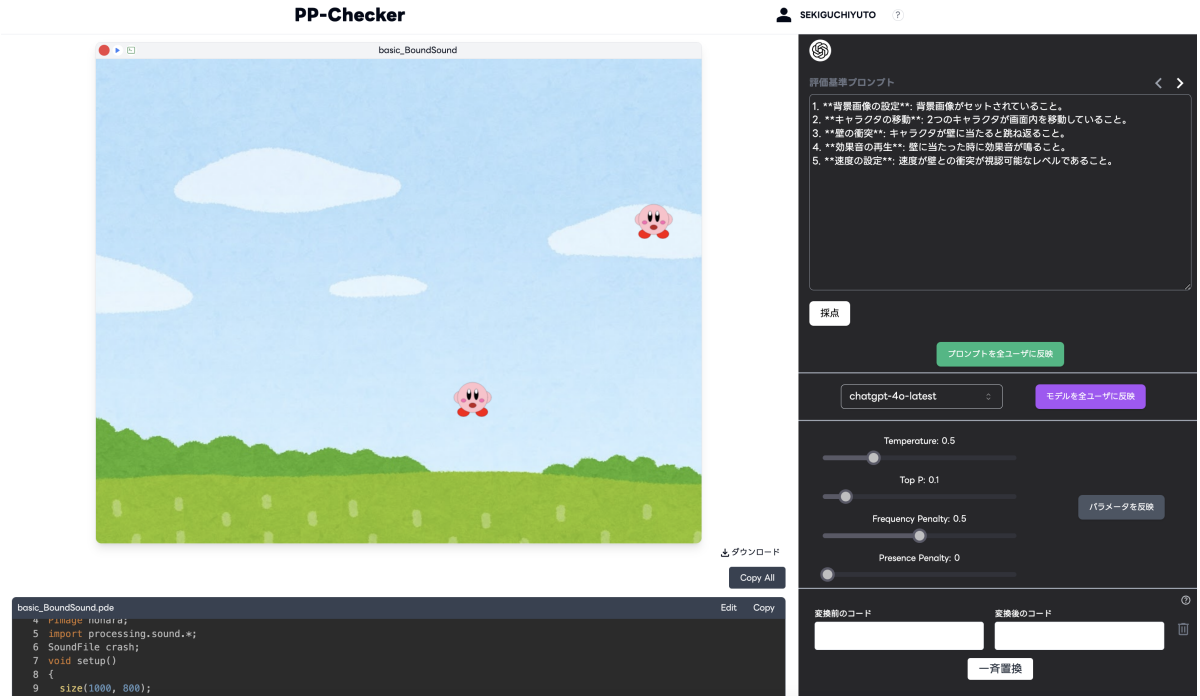



図 9: マルチメディアを含む課題の PP-Checker 上での実行画面

この要件に対応するため，PP-Checker は画像や音声ファイルといったマルチメディアを含む提出物にも対応できるようにシステムの機能を拡張した（図 9）．具体的には，提出フォルダ内に格納されたマルチメディアファイルをサーバ上に保存したうえで，プログ

ラム内で参照されるローカルパスをサーバ上のリソースパスに自動変換することで，ファイルパスの不一致による実行エラーを防ぎ，正しい実行を可能にしている．また，音声ファイル再生の実現にあたり，各音声ファイルを React の DOM 内に `audio` タグとして配置し，`playSound` や `stopSound` といった関数を新規定義し，それらを通じて動的に制御する工夫を施している．`playSound` 関数では再生開始時に既存再生を停止し，必ず音声冒頭から出力するよう設計するとともに，`stopSound` 関数は一時停止と再生位置のリセットを実行することで，繰り返し再生がスムーズに行えるよう配慮している．この機能拡張により，マルチメディアを内包する課題に対しても実行時エラーの発生を最小化しつつ，TA や教員による評価が円滑かつ正確に行える基盤が構築された．

## 4.5　標準出力を活用したプロンプト設計

初期段階の PP-Checker は，提出されたコードの内容のみを自動採点の評価指標としており，実行結果である標準出力が評価基準に含まれていなかった．このため，標準出力が重要な役割を果たす課題においても，標準出力の結果を無視して自動採点が行われていた．そこで，PP-Checker を改良し，標準出力結果をプロンプトへ組み込み，コード構造だけでなく出力内容も LLM による採点時に考慮できるような仕組みを構築した．具体的には，Processing.js コンソールへと出力される標準出力（`print`/`println`）の結果を取得することで実現した．これにより，コードの構造だけでなく，プログラムの出力結果も LLM が採点時に考慮するようになり，採点精度の向上が期待される．

特に，以下に示すような出力結果を重視する課題（advanced_CalcPI）においては，より精緻な採点が期待でき，自動採点の精度が大幅に向上すると考えられる．

**課題例（advanced_CalcPI）**：

- フーリエ級数展開を用いると，以下の式を使って円周率の近似値を求めることができる．

$$\sum_{k=0}^{N} \frac{(-1)^k}{2k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots = \frac{\pi}{4}$$

- この N をいくつまで指定するかというのを引数とし，上記の式に 4 を掛けた値を返り値として，`double` 型で返値を返す `calcPI` を作成せよ．
- 関数の定義：`double calcPI(int N);`

- また，`calcPI` の引数を 10, 100, 1000, 10000, 100000, 1000000, 10000000 とした時の結果（円周率の近似値）を標準出力せよ．

標準出力情報を組み込むことが採点精度に及ぼす具体的な影響については，5.3 節で述べる．

# 第5章　プログラミング演習講義での運用と分析

## 5.1　運用形態

PP-Checker を，明治大学総合数理学部先端メディアサイエンス学科 1 年次対象の必修科目であるプログラミング演習 I（100 分 2 コマ，2024 年 4 月 15 日から 7 月 22 日までの期間）およびプログラミング演習 II の前半（100 分 2 コマ，2024 年 9 月 23 日から 10 月 14 日までの期間）の 16 回分の講義（計 3,200 分）で運用した．履修者は学部 1 年生および再履修者で構成され，プログラミング演習 I は 123 名，プログラミング演習 II は 130 名が受講した．TA はそれぞれ著者を含む大学院生 10 名，教員は 4 名であった．

講義では，教員が冒頭の数十分間でスライドを用いて説明を行い，その後，4〜6 つの課題が提示される．課題は難易度に応じて 2 種類に分かれ，基本課題は講義時間内に，難易度の高い発展課題（または宿題）は，次回授業開始時までに提出する必要がある．これらの課題の提出先を PP-Checker にすることで運用を行った．運用時に使用したプロンプトについては付録 A に示す．

以降，本章では，講義での運用の結果を定量的に評価し，学生，TA，教員それぞれの観点から分析を行う．

## 5.2　運用結果と分析

運用期間中，基本課題 46 問と発展課題 17 問，宿題 9 問の計 72 問の課題を実施し，PP-Checker を通じて合計 13,035 回の提出が行われた．

PP-Checker の導入による迅速なフィードバックの効果を定量的に評価するため，学生がフィードバックを受けてから再提出までの時間的変化や提出回数について講義内に採点業務を行う必要がある基本課題に絞って分析を行った．分析の結果，基本課題 8,591 回の提出のうち，1,799 回（約 20.9%）は TA 採点前に LLM によるフィードバックを受けた時点で，学生が自主的に課題を取り下げてコードの改善を行っていることがわかった．図 10 に，PP-Checker を利用した際の再提出までの時間の分布を箱ひげ図で示す．TA による採点後に再提出された件数は 634 件であったのに対し，LLM によるフィードバック

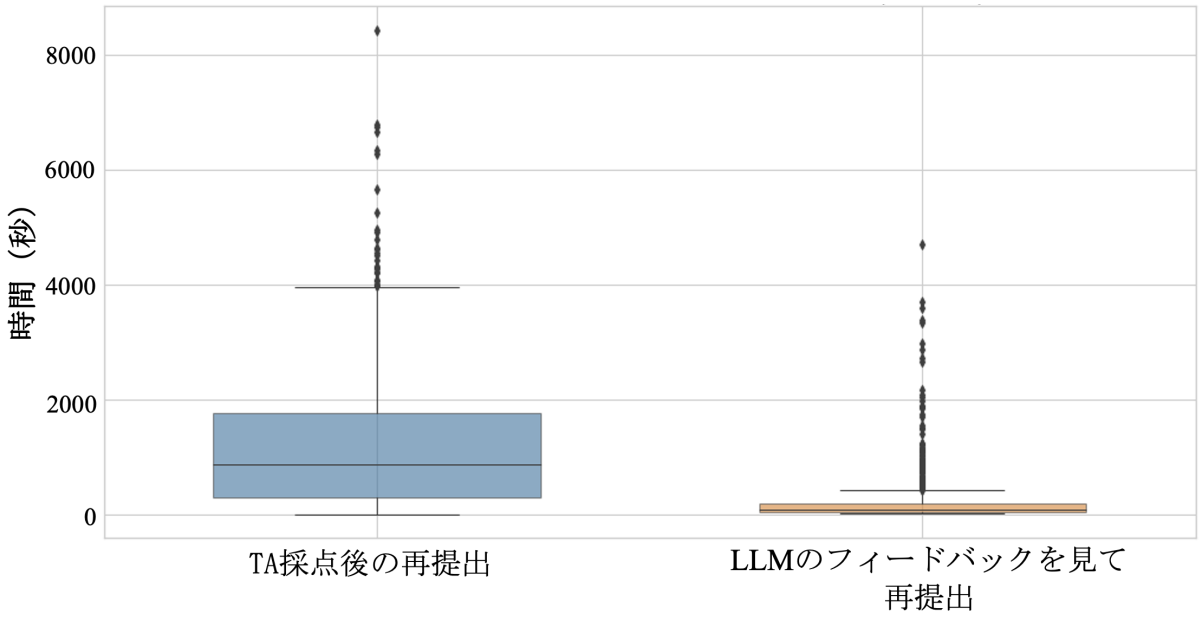



図 10: PP-Checker 上における再提出までの時間の比較

を受けた後，自主的に修正し TA に再提出された件数は 886 件であり，学生が早期フィードバックを活用していることが明確となった．再提出までの平均時間は，TA 採点後の再提出が約 20.3 分であったのに対し，TA 採点前に LLM のフィードバックを受けて修正した場合は約 3.6 分であり，LLM による即時フィードバックが学生の迅速な修正に寄与していることが示された．

基本課題のうち，2023 年度と 2024 年度のプログラミング演習 I で変数について教授する回において，学生が最初に取り組む 1 問目の課題に同一の内容のものがあったため，この課題について PP-Checker を使用した 2024 年度の結果（123 人）と PP-Checker を使用する前の 2023 年度の結果（115 人）を比較分析した．従来の採点手法では，学生は Google Drive 上の指定フォルダに提出物アップロードし，Google Form を使用し，学年，組，番号，名前，提出する課題の詳細を申請する必要があった．このプロセスは，提出や再提出を行うたびに繰り返す必要があった．また，TA や教員は，Google Drive と同期した Processing で作成された専用アプリケーションを使用して，学生と課題を選択した後，Processing エディタ上で提出物を開き，コードを手動で実行・確認する．その後，要件を満たしているか評価し，点数とフィードバックコメントを Google スプレッドシートに手動で入力する必要があった．

従来の採点手法と PP-Checker を比較した結果を図 11 に示す．図 11 より，学生がコードの誤りに気付き，再提出するまでの平均時間は，2023 年度の約 30.9 分に対し，2024 年度は約 12.3 分と大幅に短縮されたことが明らかになった．また，再提出回数に関しては，

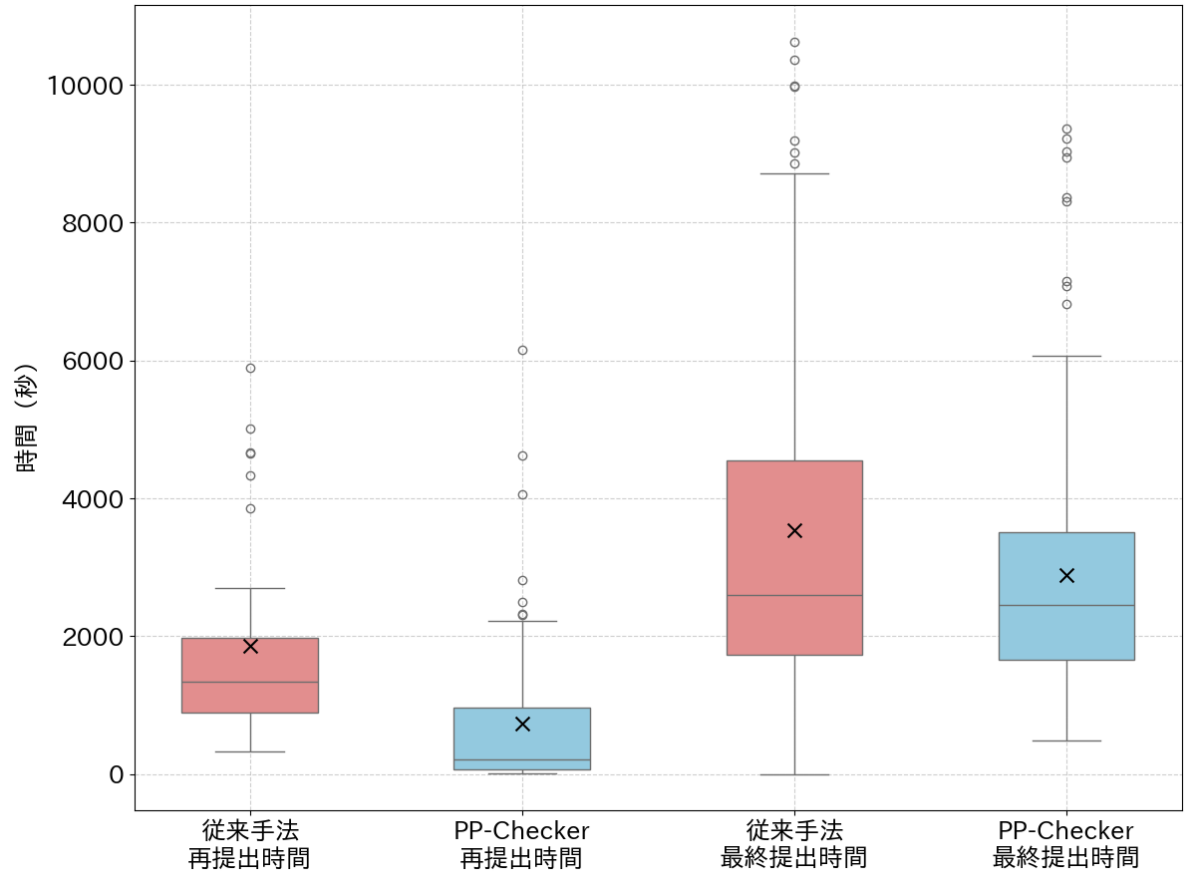



図 11: PP-Checker 導入前後における課題達成時間の比較

PP-Checker を利用した 2024 年度の学生の提出回数は 2023 年度より増加したものの，課題が提示されてから最終的な提出物を提出するまでの平均時間は，2023 年度の約 58.9 分に対し 2024 年度は約 48.2 分と短縮されたことがわかった．

また，プロンプト変更機能の使用状況について，プロンプトの変更履歴の収集を開始したプログラミング演習 I 第 5 回講義以降の基本課題 36 問について分析を行った結果，合計 99 回のプロンプト変更が観察された．プロンプト変更前後の LLM の採点正解率を比較したところ，変更前が 61.9%（標準偏差：13.5），変更後は 65.6%（標準偏差：14.2）と 3.7%の向上がみられた．課題ごとの分析では，プロンプト変更の効果に顕著な変動があることが確認された．具体的には，初期精度が 60%未満のプロンプトに変更を加えた場合，90%近くのケースで精度が向上することが確認された．最も改善した課題では，初期精度 33.3%のプロンプトが変更後に 72.9%まで向上し，約 40%の精度向上が確認された．一方，初期精度が 80%以上あった課題では，プロンプト変更後にその精度が低下するケースが大半であった．さらに，標準出力の結果が重要視されるような動的でない課題においても，採点精度が約 54.2%と低い結果を示した課題が存在した．

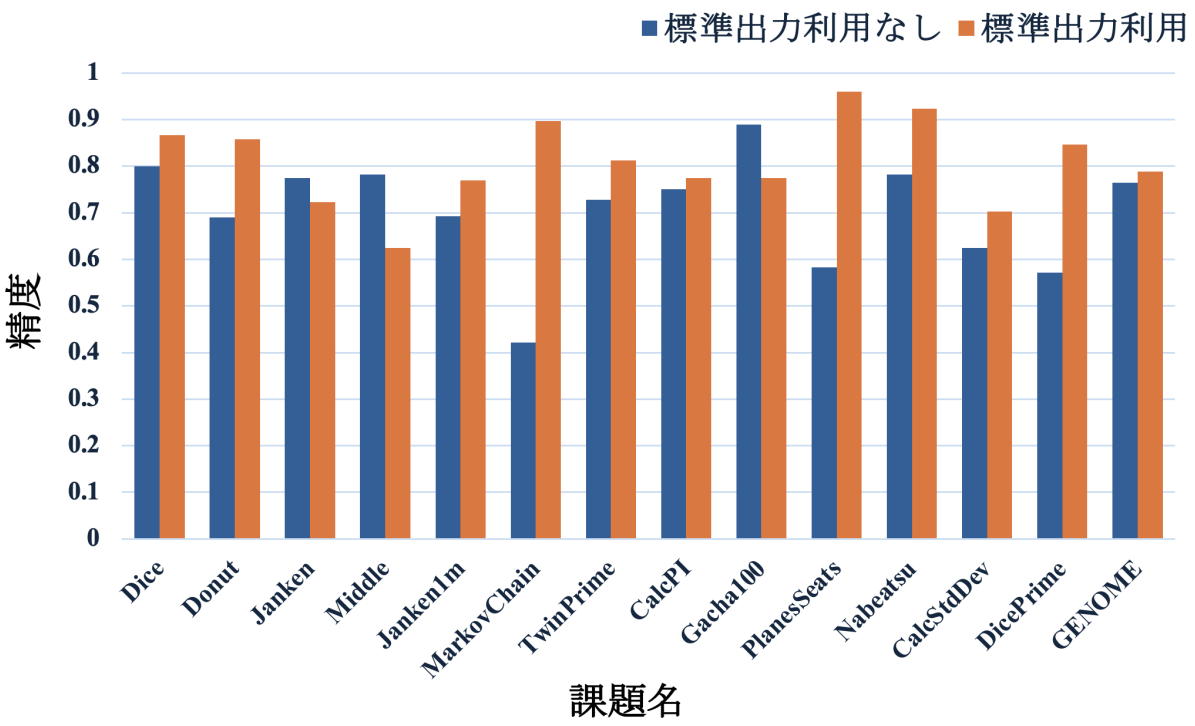



図 12: プロンプトへの標準出力反映有無による採点精度の比較

## 5.3　標準出力を用いた判定手法の検証

4.5 節で述べた通り，標準出力を含む課題において，提出されたコードだけでなく，標準出力の結果もプロンプトに反映させることで，自動採点の精度向上が期待できるためシステムを途中で改良した．その結果，実際の運用においては，標準出力以外の要素に対する一部の指摘に誤りが見られたものの，標準出力が誤っている提出物については正確に指摘できているケースが多く確認された．

さらに，標準出力を活用したプロンプト設計が採点精度に与える影響を評価するため，標準出力を含む課題に対して，標準出力を用いた判定手法導入前後の課題および提出物に対する採点精度を比較した結果を図 12 に示す．この図から，プロンプトに標準出力の結果を反映させた場合，14 個の標準出力を含む課題のうち 11 個の課題において精度が向上していることがわかった．また，プロンプトに標準出力の結果を反映しない場合の採点精度の平均は約 70.4%（標準偏差：11.5）であった一方，標準出力の結果を反映した場合の採点精度の平均は約 80.8%（標準偏差：8.8）となり，採点精度は 10.4%向上した．

表 1: 学生による LLM フィードバックでのコード誤り早期発見に関する時期別主観評価結果

| 時期 | フィードバックの有用性評価（人） | | | | |
|---|---|---|---|---|---|
| | 0〜25% | 25〜50% | 50〜75% | 75〜100% | 指摘は受けていない |
| 初回講義 | 11 | 12 | 22 | 26 | 43 |
| 学期前半（4〜5 月） | 7 | 33 | 49 | 19 | 6 |
| 学期後半（6〜7 月） | 7 | 40 | 50 | 15 | 2 |

表 2: フィードバックの例

| No. | フィードバック内容 |
|---|---|
| 1 | 直径 5 ピクセルの円の数が 64 個ではない。 |
| 2 | 素数 39 が含まれている。 |
| 3 | 双子素数の総数が出力されていない。 |
| 4 | キャラクタの表示順序がマウスに近いキャラクタが手前に表示されるように実装されていない。 |
| 5 | 各アイテムの累計取得数をカウントし、最も多いアイテムの数を標準出力しているが、最も多いアイテムの ID が出力されていない。 |
| 6 | 2 人目以降の着席のシミュレートにおいて、i が 264 未満であるため、264 番目の座席が考慮されていない。 |
| 7 | コマの反転が正しく動作していない。クリックしたコマが白から黒、または黒から白に変わるべきだが、現在のコードではその動作が不完全である。 |
| 8 | 関数 drawMinimumRect が作成されていない。 |

## 5.4　SUS と学生アンケートの結果

2024 年 7 月 15 日に，PP-Checker のユーザビリティと信頼性に関する実証的評価を行うため，System Usability Scale（SUS）[31] を用いた PP-Checker のユーザビリティと信頼性を評価するアンケート調査を実施し，114 名の学生から有効回答を得た．また，生成 AI に関する信頼性の変化に関して調査した．

調査の結果，PP-Checker の SUS スコアは 76.4 であった．標準偏差は 12.2 であり，学生間の評価には一定のばらつきがみられたものの，平均スコア（68.0）を大きく上回っており，システムのユーザビリティが高く評価された．この結果は，Bangor らの SUS 評価基準 [3] に基づいても高評価であることがわかる．

実験期間ごとに，PP-Checker 上での LLM フィードバックがコード誤りの早期発見に与えた影響について，学生へのアンケート結果を表 1 に示す．表 1 より，講義の進行に伴い，LLM から指摘を受ける学生の人数が増加していることがわかる．また，学期後半においても 57%以上の学生が提供されたフィードバックの半分以上を有用と評価していることがわかった．

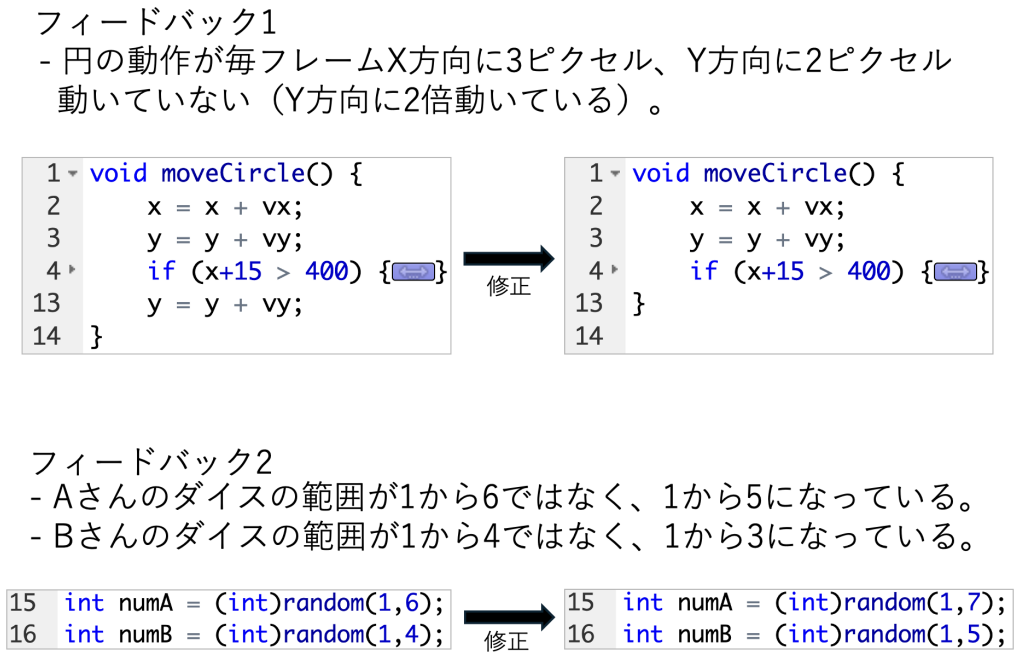



図 13: 参考になった LLM のフィードバックと修正の実例

運用中に LLM が実際に行ったフィードバックの一部を表 2 に示す．表 2 にはフィードバックの一部のみを記載しているが，記載している通り，ほとんどのフィードバックにおいて直接的な回答を示すものはなく，曖昧性が保たれた表現が用いられていた．参考になったフィードバックには，「円の動作が毎フレーム X 方向に 3 ピクセル、Y 方向に 2 ピクセル動いていない（Y 方向に 2 倍動いている）。」や「A さんのダイスの範囲が 1 から 6 ではなく、1 から 5 になっている。」といった具体的な誤りの箇所を含むフィードバックがあげられた（図 13）．クリックするたびにじゃんけんの結果を標準出力する課題では，「judgeJanken 関数内の条件で、パーとグーの勝敗判定が間違っている。」といった，複数回操作しないと気づきにくい誤りを指摘するフィードバックも有用と評価された．さらに，「枠線が削除されていない」といった，一見しただけでは見落としがちな問題を指摘するフィードバックも高く評価された．

一方で，参考にならないと評価されたのは，正確性に欠けるフィードバックや課題を解く上でプログラムの挙動に影響しない指摘であった．例えば，「標準偏差の計算において、平均値を毎回再計算しているため、効率が悪い。」といった課題の評価に影響しない指摘は，参考にならなかったと評価された．

表 3 は，PP-Checker の使用による ChatGPT や Claude などの生成 AI に対する信頼度の変化を調査した結果を示している．この表から，学生の生成 AI に対する信頼の平均値

表 3: 生成 AI のチャット機能における対話内容の信頼性の時期別主観評価結果

| 時期 | 生成 AI への信頼性評価（人数） | | | | 平均 |
|---|---|---|---|---|---|
| | 全く信頼していない: -2 | あまり信頼していない: -1 | 少し信頼している: 1 | 非常に信頼している: 2 | |
| 初回講義 | 9 | 28 | 61 | 16 | 0.41 |
| 学期前半（4〜5 月） | 2 | 33 | 67 | 12 | 0.47 |
| 学期後半（6〜7 月） | 1 | 37 | 67 | 9 | 0.40 |

表 4: TA アンケート結果

| 質問項目 | 評価値の分布 | | | | | 平均 |
|---|---|---|---|---|---|---|
| | -2 | -1 | 0 | 1 | 2 | |
| PP-Checker は全体的に使いやすいと思いましたか？ | 0 | 0 | 0 | 2 | 7 | 1.78 |
| PP-Checker をこれからも利用したいと思いますか？ | 0 | 0 | 0 | 1 | 8 | 1.89 |
| PP-Checker 導入前後で、課題採点業務の作業効率は向上しましたか？ | 0 | 0 | 0 | 0 | 6 | 2.00 |

には大きな変化が見られないことがわかった．しかし，PP-Checker の使用頻度が増すにつれ，生成 AI を極端に信頼しない学生および極端に信頼する学生の数が減少する傾向にあった．

## 5.5　TA アンケートの結果

プログラミング演習 I における PP-Checker の運用終了後，著者を除く 9 名の TA を対象にアンケート調査を実施した．アンケートは，PP-Checker の使用感や作業効率に関する定量評価項目（表 4）と，質的データ収集のためのインタビュー項目で構成した．定量評価は 5 段階のリッカート尺度（-2：全くそう思わない〜2：非常にそう思う）を用いた．PP-Checker 導入前後の比較は，導入前の TA 経験を持つ 6 名に限定して実施した．

表 4 に示される評価項目において，全体的な使いやすさに対する評価（平均 1.78）では，システムの直感的な操作性と安定性が高く評価され，日々の業務負担を軽減する効果が認められた．今後の継続利用意欲（平均 1.89）も高い結果であることから，TA の役割における PP-Checker の重要性が確認できた．採点業務の作業効率向上（平均 2.00）においても非常に高い評価が得られており，特に手動での採点やフィードバック提供に比べ，迅速かつ精度の高いフィードバックが可能であることが利点として指摘された．

インタビューでは，LLM との協調や PP-Checker 導入による TA 業務の変化に焦点を当てた．結果として，「採点作業の効率化により，採点にかけていた時間を質問対応に回せるようになった」や「GPT のフィードバックがあることで基礎的な質問が減少した」，「プ

ロンプトの変更により採点精度が向上した」といったポジティブな意見が多く得られた．

一方で，課題内容による LLM の精度の差に関する意見や「プロンプトの変更が改善に繋がるとは限らないうえ，時間に制約がある TA の業務内で行うのは難しいと感じた」といったネガティブな意見も一部みられた．

## 5.6 教員アンケートの結果

プログラミング演習 I における PP-Checker の運用終了後，採点やプロンプトの変更などを行っていた 2 名の教員を対象にアンケート調査を実施した．アンケートは，PP-Checker の利点や改善点，TA や学生に感じた変化などを調査するための項目で構成した．調査の結果，PP-Checker の利点として，人手で確認する際に注目すべき点がわかりやすくなる点や LLM のフィードバックによって学生が誤りに気づいて修正する周期が早くなったという意見が得られた．一方，段階的な減点条件を定めたい場合のプロンプト作成が難しかったという意見が得られた．

教員が TA に感じた変化として，以前は多かった氏名や課題名の記入ミスに対する指摘をしなくて済むようになったため，ストレスが減少しているように感じたという意見が得られた．また，不備を指摘することが機械であることで，TA や教員側だけでなく学生側のストレスも下がっているという意見が得られた．しかし，自発的な興味が確立していない初学者に対する LLM の依存を危惧するような意見もあった．

PP-Checker を今後も運用したいかという質問では，2 名ともに非常にそう思うと肯定的な回答をした．

# 第6章　考察

## 6.1　学生に与える影響に関する考察

SUS スコアが 76.4 を記録し，平均スコアを大きく上回っていることから，PP-Checker は学生に対して高いユーザビリティと信頼性を提供できるシステムであることが示された．PP-Checker の導入により，学生が誤りを素早く認識・修正でき，学習サイクルが加速したと考えられる．実際，LLM のフィードバックにより短い時間での再提出が確認でき，PP-Checker を導入した 2024 年度は，学生が誤りを認識し，再提出するまでの平均時間が 2023 年度の約 30.9 分から約 12.3 分に短縮されていた．この結果は，PP-Checker が学生の迅速な自己修正と学習効率の向上に寄与していることを示している．また，LLM からのフィードバックを受けて自ら修正を行うプロセスは，単にプログラミングスキルの向上だけでなく，問題解決能力全般の改善にもつながると考えられる．

PP-Checker の利用が生成 AI に対する信頼度に及ぼす影響については，全体的な信頼度の変化は見られなかったものの，使用頻度の増加に伴い極端な評価が減少する傾向が確認された．この結果は，学生が PP-Checker を通して生成 AI をより実用的かつ現実的なツールとして捉えるようになっていることを示唆している．特に，PP-Checker 使用前のような「完全に信頼する」または「全く信頼しない」といった極端な評価が減少し，生成 AI が便利なツールであると理解されつつも，万能ではなく，あくまで補助的なツールとして利用すべきだという認識が深まったことを示していると考えられる．今後は，本研究で着目しているフィードバックの曖昧性がもたらす効果について検証を行っていく予定である．

## 6.2　TA に与える影響に関する考察

実証実験の結果から，PP-Checker が TA と教員の作業効率向上およびストレス軽減に有効であることが示された．TA アンケートの結果から，PP-Checker は課題採点の効率化を実現し，TA が学生への質問対応に時間を充てられるようになった点が特に評価されていると考えられる．従来の採点プロセスでは，TA が各課題ファイルを個別に開き，コード

を手動で確認する必要があり，時間と労力を大幅に要していたが，PP-Checker の導入によりこのプロセスが自動化され，採点業務の負担が軽減されたと考えられる．また，LLM による即時フィードバックにより，学生は初歩的なミスを自ら修正し，より完成度の高い状態で提出を行うようになった．その結果，TA が同じような間違いを含む提出物を繰り返し確認する手間が減少したと考えられる．さらに，LLM によるフィードバックが基礎的な質問の減少につながったことも，TA の業務効率向上に寄与したと考えられる．これにより，TA が学生に対してより質の高い支援を提供できる環境が整ったといえる．

しかし，TA からは LLM フィードバック精度に課題内容によるばらつきがある点や，プロンプトの変更が必ずしも採点精度の向上に直結しない点に対する懸念も示された．特に，時間制約のある TA の業務の中でプロンプトの修正を行うことが難しいと感じている点は，今後の改善に向けた重要な意見であると考えている．

## 6.3 教員に与える影響に関する考察

教員アンケートの結果からも，PP-Checker によって学生はより迅速にフィードバックを得ることができ，再提出までのプロセスが効率化され，学習効果が高まったことが示された．また，教員や TA にとっては人為的な確認作業の負担が軽減され，学生の記入漏れや不備の指摘が減少したことで，ストレスが軽減された点も大きな成果であると考えられる．実際に，プログラミング演習 II はプログラミング演習 I に比べて扱うトピックの難易度が高いため，TA および教員に対する質問も多く，また課題のチェックには非常に手間がかかる．そのため，2013 年度から 2023 年度まではプログラミング演習 II の基本課題について講義内に何度か採点を試行したものの，結局講義内に採点が終わらないことが判明し，また質問対応などへの負担も大きかったため，採点を行っていなかった．今回，PP-Checker の導入とプログラミング演習 I における運用実績から，プログラミング演習 II においても講義内の採点が可能であると判断し，実際に基本課題の採点についても運用したところ，講義内で迅速な採点およびフィードバックが可能となった．その結果，採点効率が上昇し，プログラミング演習 II において教授するクラスに関する理解も高まったと考えられる．

一方，教員アンケートの他項目では，学生が LLM の依存に対する懸念も見受けられた．特に，学習初期段階の学生が自発的な興味を持たないまま，LLM のフィードバックに過度に依存する可能性は，教育支援における LLM の利用に関する研究で重要な課題 [27,40,83] として挙げられる．本研究では，学生の生成 AI に対する信頼度に及ぼす影響を調査する

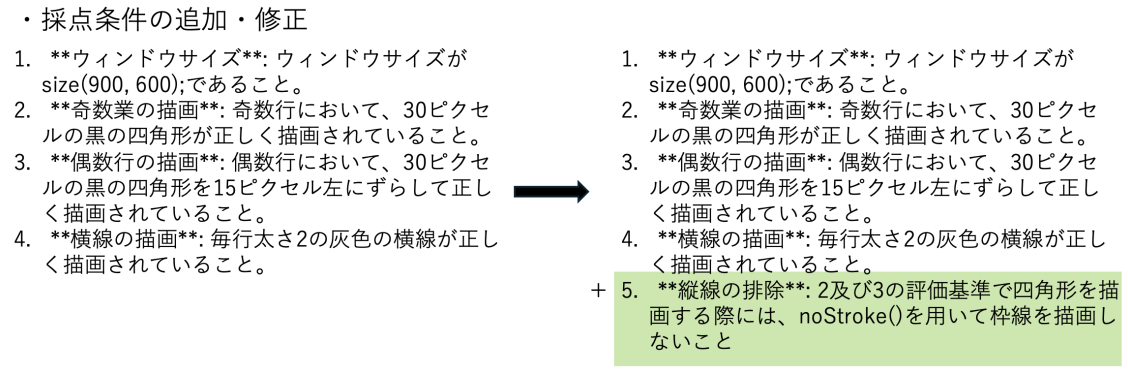



図 14: 採点条件の追加を行っているプロンプト修正の例

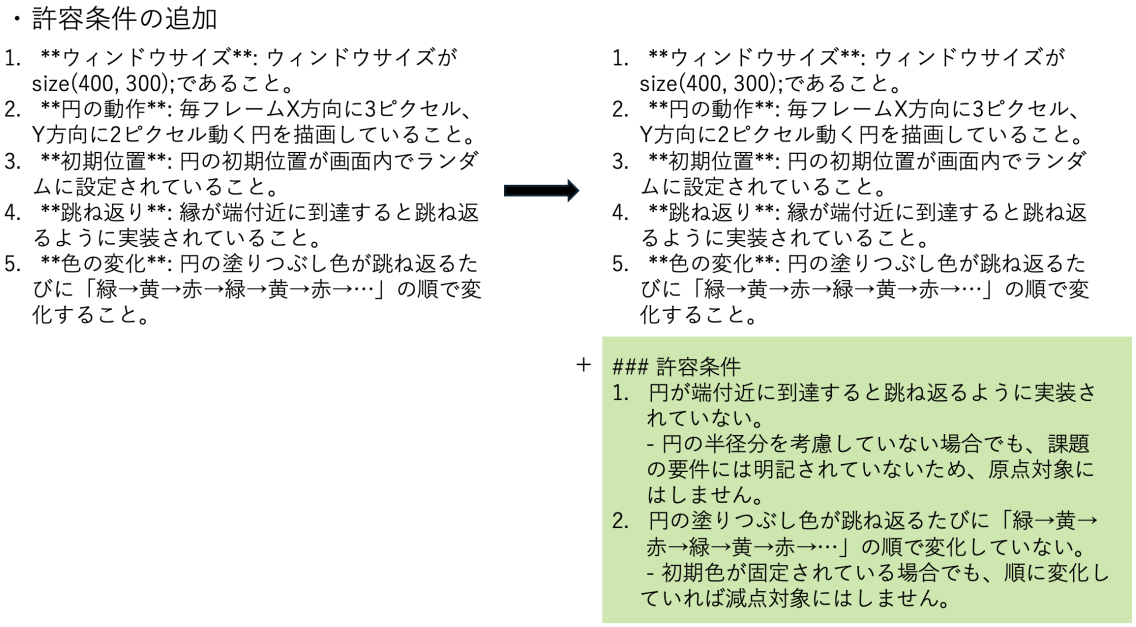



図 15: 許容条件の追加を行っているプロンプト修正の例

（表 3）とともに，曖昧性に着目することで LLM への過度な依存を防ぐ試みを行っているが，今後の LLM の発展や運用に伴い，LLM によるフィードバック内容がどのように学生の理解や修正に影響を与えたかを定量的に評価し，依存を防ぐ手法の構築が必要になると考えられる．

## 6.4 採点精度に関する考察

### 6.4.1 プロンプトのリアルタイム修正

リアルタイムでプロンプトを修正することで，LLM のフィードバックがより的確になり，学生がコードの誤りに迅速に気づいて修正する機会を与えることができたと考えら

れる．具体的には，プロンプト変更前の採点精度が 61.9%であったのに対し，変更後は 65.6%に改善し，約 3.7%の向上が確認された．この結果は，プロンプトのリアルタイムな柔軟性が自動採点の精度に直接的な影響を与えることを示している．TA のアンケートでは，一部の TA が時間制約のある TA の業務の中でプロンプトの修正を行うことが難しいと感じていたものの，全 16 回の講義で 99 回のプロンプト変更が実際に行われており，プロンプトの修正は可能であることが示唆される．しかし，TA が時間制約の中でプロンプトを頻繁に修正するのが難しいという課題は依然として存在するため，今後は効率的にプロンプトを変更できるインターフェースの開発も検討する．

プロンプト修正の主な例として，「**クリック時の再描画**: 画面をクリックするたびに **a, b, c, d の値が 1〜20 の間でランダムに決定され**、曲線が再描画されていること。」という条件の太字部分のような未定義パラメータの具体化，出力形式の明確化，数式の具体的な指定といった変更が挙げられる．また，図 14 に示すような講義開始時には不足していた採点条件の追加や修正，図 15 のような明示的な例外規定の追加などのプロンプト修正も行われていた．これらの修正により，プロンプトの曖昧さを授業の進行に伴い軽減し，LLM の反復的な誤評価を抑制することで，教育効果の向上につながる可能性が考えられる．

図 16 は，ある基本課題における時間経過に伴う提出状況，採点精度，点数の推移を示している．この結果では最初の 20 分は精度が 30%程度と低いが，30 分までに 2 回プロンプトの修正がなされ 70%近くまで向上している．このプロンプトの修正は，最初の 20 分程度で提出された課題の自動採点およびその結果を無視して提出されたものにおける自動採点の結果を参考に行われている．

プロンプトがまだ不十分な段階で精度が低い時に課題を提出する学生は基本的にプログラミング能力が高く，また自信があると考えられるため，精度が低くてもそのフィードバックを無視して提出できるため，大きな問題はない．一方で，中盤に課題を提出する，プログラミングにおける自信があまりないと考えられる学生は精度が低い場合にその影響を受けてしまう可能性がある．つまり，今回のように採点精度は時間とともに徐々に向上することが望ましく，どの課題においても一定の精度向上を保証できるような採点手法の構築が重要であると考えられる．

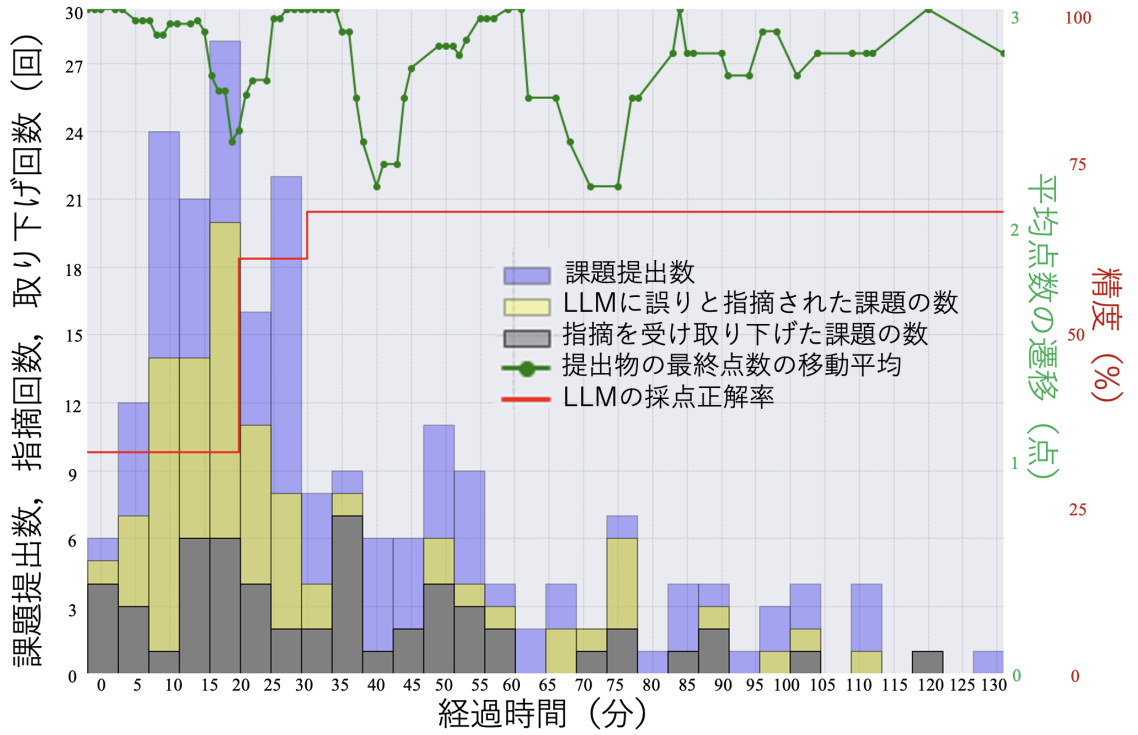



図 16: ある課題における提出数（青棒），LLM の指摘数（黄棒），課題取り下げ回数（灰棒），課題点数の移動平均（緑），LLM フィードバックの正解率（赤）の時間推移．PP-Checker は，プログラミングに自信のない学生が課題を提出する中盤以降において，高い採点精度を保つことで初学者に対する有用性があると考えられる．

### 6.4.2 標準出力を考慮したプロンプト

標準出力がある課題において標準出力の結果をプロンプトに反映させることで、自動採点の精度が約 10.4%向上したことが確認された．これにより，単にコードの構文だけでなく，プログラムの実行結果を評価に組み込むことが有効であることが示された．

一方，すべての課題で精度が向上したわけではなく，標準出力を反映しても精度が低下した課題もあった．特に，ボタン操作などのインタラクティブな要素と標準出力が組み合わさる課題で顕著であり，動的なインタラクションを含む場合はプロンプト設計の工夫が求められる．今後，標準出力に加え，実行画面の画像をプロンプトに含めた採点手法の検討も進める予定である．

### 6.4.3 採点精度向上の可能性

プロンプトのリアルタイム修正や標準出力を考慮する手法により，多くの課題で採点精度が向上したものの，依然として精度が不十分な課題や，期待された効果が見られなかった課題も少なからず存在した．

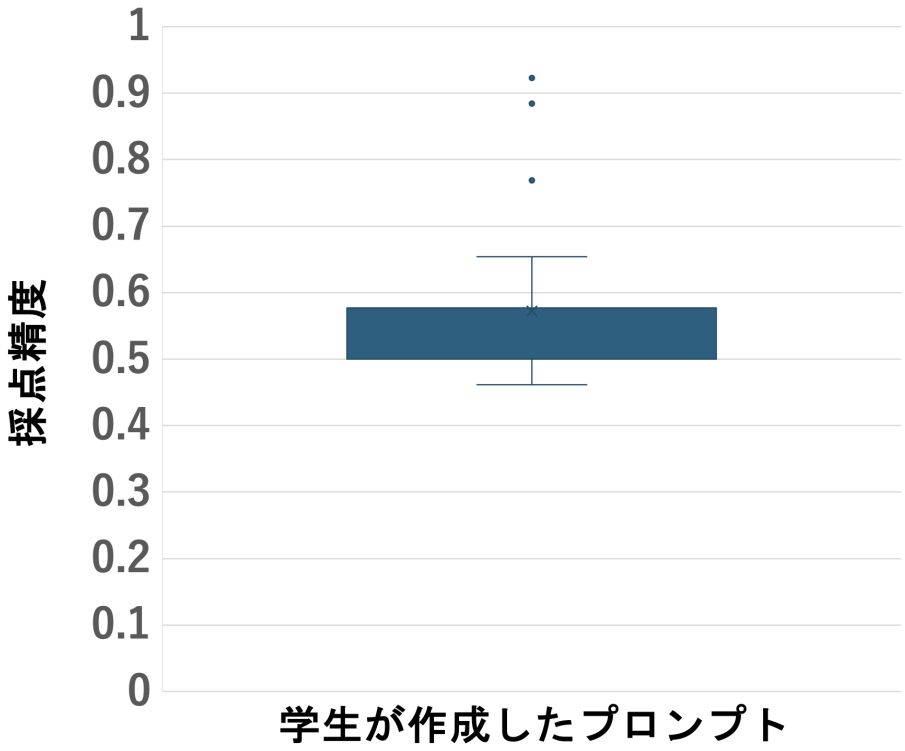



図 17: 学生が作成したプロンプトの採点精度分布

そこで，実際に出題された課題のうち採点精度が特に低かった 3 つの課題を対象に，大学生および大学院生 27 名の学生に各々独立にプロンプトを作成してもらった．その結果，運用時に採点精度が 62.5%であった課題において，学生の作成したプロンプトの採点精度の平均が 57.0%，標準偏差が 13.7%であったが，ある学生が作成したプロンプトによって 92.3%にまで精度が向上するケースが確認された（図 17）．この採点精度が高いプロンプトを採用することで授業内の採点精度を飛躍的に高めることができる．この結果を基に，今後は課題を早期に解き終えた学生同士でプロンプト設計のコンペティションを行い，多様な視点や柔軟な発想をプロンプトに反映していくような仕組みを導入することで，授業の進捗に合わせてプロンプトの質全体を向上させる取り組みを進める予定である．このアプローチにより，LLM を活用した自動採点の精度向上がさらに期待できる．

LLM の技術革新により採点精度向上が見込まれるため，2024 年 9 月 12 日に OpenAI 社がより高度な推論能力を持つモデルとして発表した OpenAI o1-preview [60] を利用して精度を検証した．しかし，実際に一部の課題を o1 モデルで採点した結果，従来のモデルと比べて採点精度に大きな変化は見られなかった．これにより，現在のプロンプト設計をそのまま適用しただけでは，o1 モデルの能力を十分に引き出せない可能性が示唆された．

また，o1 モデルを使用するにはコスト面も課題となる．現在の PP-Checker で o1 モデ

ルを使用すると，1 つの課題を採点する際に約 3,800 トークンを消費しており，これを講義で頻繁に利用するには非常に高額になる．例えば，1 リクエストのコストが 0.2 ドルだとすると，100 人の学生が 1 日に 10 回利用した場合，1 日で 200 ドル，1 学期（PP-Checker の使用は 12 回）では 2,400 ドルに達する．実際に，分析過程において著者は，1,894 件の課題に対する採点で合計 329 ドルもの料金を消費した．こうしたコストを考えると，教育現場での継続的な利用は 2024 年現在では難しいと考えられる．しかし，将来的に o1 モデルの利用コストが低下すれば，エラーを起こした学生の思考過程を再現するような出力や，教員や TA が採点時に注目しているポイントの提示など，より高度なフィードバックに活用できる可能性が期待される．

# 第7章　本研究の制約と展望

## 7.1　運用コストについて

PP-Checker の運用において，LLM を活用した自動採点には，生成・処理されるトークン数に伴うコストが大きく，これはシステム全体の持続可能性に影響を与える可能性がある．PP-Checker は全ての提出物に対して LLM によるリアルタイムでの自動採点を行うため，多人数の講義や高度な LLM モデルの使用は運用コストの増加につながる．このため，LLM 利用の際のコスト削減と採点の精度維持を両立するための手法も必要となる．

コストの最適化のためには，プロンプトのトークン数を最小限に抑えることや，提出内容の類似度に応じたキャッシュ機構を導入することが考えられる．具体的には，同様の課題に対して頻出する回答パターンを事前に蓄積し，これらに該当する回答には過去のフィードバックを再利用することで，計算資源の消費を大幅に削減することが期待される．このようにして，コスト削減と採点精度の向上を同時に達成し，教育現場での継続的な運用を支える体制が構築できると考えられる．

## 7.2　採点精度について

LLM を活用した採点システムである PP-Checker においては，特に動的要素や視覚的表現を含むプログラムに対する評価の精度が課題である．ランダム性や動的挙動を伴う課題では，LLM の出力に正確性や一貫性が欠け，教員や TA がフィードバックの確認・修正を行う必要が生じることがある．こうした不一致や誤認識を防ぐには，プロンプト設計におけるさらなる工夫が求められる．

今後，LLM の採点精度向上には，プロンプト設計の工夫が重要な要素となる．6 章で述べたように，実行画面の画像をプロンプトに取り込むことや課題を解き終えた学生自身によるプロンプト設計，将来的な LLM モデルの精度向上が採点精度向上につながることが期待される．また，TA や教員に対してリアルタイムで現在のプロンプトの採点精度を提示することにより，プロンプトのリアルタイム調整を通じて採点精度の向上を図ることも可能であると考えられる．

## 7.3 数年規模での効果検証と長期的な成長可能性

PP-Checker の効果を最大限に引き出すためには，システムが利用者の行動や成果促進に与える影響を数年規模で検証することが重要である．リアルタイムフィードバックの精度向上や学生の理解促進にどのように貢献するかについての定量的評価を行うことで，PP-Checker の有効性を長期的に示すことができると考えられる．

こうした評価を通じて，PP-Checker が学生の自主的な学習や TA の採点業務効率化に有益であることを確認し，教育者と学生の双方にとって価値あるシステムとして成長させるための改善を重ねていく．PP-Checker はプログラミング教育支援の可能性をさらに開拓し，長期にわたる実用性を持つシステムとしての成長を目指す．

## 7.4 適用範囲の拡大と教育支援システムとしての発展

2025 年 1 月現在，PP-Checker は主にクリエイティブコーディングのプログラミング課題に対応しているが，Web 開発や PDF 等のレポート，書類といった他の分野にも展開することで，様々な現場での利用がさらに拡大できる可能性がある．このような適用範囲の拡大により，幅広いユーザ層や多様な内容に対応できる仕組みとして，PP-Checker における手法の利便性と実用性がより一層高まることが期待される．異なる分野や業務環境にも対応するためには，PP-Checker の柔軟性を向上させることが求められる．特に，Web 開発における動作チェックや，書類・レポート内容の整合性確認といった多様なタスクに最適化したフィードバックを提供できるようなカスタマイズ性を持たせることで，より幅広いニーズに応えることが可能となる．このように，PP-Checker はプログラミング教育の分野のみにとどまることなく，多様な業界での導入が進むことが期待できる．

# 第8章　結論

本研究では，プログラミング教育における課題採点業務の効率化を目指した曖昧性のある自動採点システム PP-Checker を提案し，その有用性について検証を行った．PP-Checker は，学生の多様な解答を LLM により柔軟に評価し，即時的なフィードバックを提供することを特徴としている．

16 回の講義における実運用の結果，PP-Checker は学生，TA，教員のいずれからも高い評価を得た．特に，LLM を用いた早期フィードバックによる再提出の迅速化や課題の修正機会の提供が学習プロセスの加速に寄与することが示された．さらに，曖昧性のある採点システムは学生の創造的な解法を促し，学習意欲の向上にも寄与すると考えられる．

今後は，学生がプロンプト設計に参加する手法の検討や LLM の高度なモデルの適用とそのコストや精度のバランスを考慮した運用方法の検討を行う．また，実行画面の画像をプロンプトに含める手法についても検討を行う．

本研究の成果は，LLM と人間がいかに協力し，互いの強みを引き出し合うかを考える上での新しい指針も示している．PP-Checker はプログラミング教育における支援ツールとしての可能性を広げ，教育現場での採点支援，学習意欲向上，柔軟なフィードバック提供を通じて，学習者と教育者の双方にとって価値あるシステムとなることを目指す．LLM の活用が進む中で，本研究の成果が，プログラミング教育支援システムのさらなる発展と進化に貢献することを期待する．

# 謝辞

本研究を行うにあたり，ご協力いただいた中村先生をはじめとする研究室の方々に感謝申し上げます．また，家族や友人といった周りの方々のご支援のおかげで充実した学生生活を送ることができました．ありがとうございました．

学部3年から4年間にわたり研究指導してくださった中村聡史先生に深く感謝いたします．先生のご指導のおかげで国内発表をはじめ，国際発表や査読付きの国内発表を行うことができ，充実した研究室生活となりました．また，学部3年生から4年間，学生生活を共にしてくださった同期に感謝申し上げます．加えて，論文添削をはじめとした研究活動に積極的に協力してくださった先輩方，後輩の皆様にも感謝を申し上げます．日頃から，上下関係の壁を越えて関わってくださり，本当にありがとうございました．そして，ここまで応援してくれた家族のおかげで楽しく，充実した学生生活を送ることができました．本当にありがとうございました．

最後に，ご支援いただいた全ての皆様に感謝いたします．ありがとうございました．今後も何卒よろしくお願いいたします．

# 付録

## A　使用プロンプト

### System プロンプト

あなたは Processing 言語のコードを評価する TA です。以下の基準に基づいて、提出されたコードを正確に評価してください。また、コードが適切に動作していれば、コードの整理度合いは問いません。

**課題説明**: 後述する Processing のコードは、特定の条件を満たすプログラムです。コードは正確に動作し、指定された要件を満たす必要があります。全ての条件を満たしている可能性もあります。

**評価基準**:

1. **Satellite クラスの作成**:

- **概要**: ‘CelestialBody‘ クラスを継承し、‘Satellite‘ クラスを作成しているか確認してください。
- **評価ポイント**：
- ‘Satellite‘ クラスが ‘CelestialBody‘ を正しく継承していること。
- クラス宣言や継承の文法が正しいこと。

2. **コンストラクタの実装**:

- **概要**: ‘Satellite‘ クラスのコンストラクタが、衛星の惑星である ‘Planet‘ オブジェクトを引数として受け取るように実装されているか確認してください。
- **評価ポイント**：
- コンストラクタが正しく定義され、‘Planet‘ オブジェクトを引数として受け取っていること。
- 引数として受け取った ‘Planet‘ オブジェクトを適切にクラス内で使用していること。

3. **衛星のパラメータ設定**:

- **概要**: コンストラクタ内で、‘distance‘、‘angleSpeed‘、‘bodyColor‘、半径 が指

定通りに設定されているか確認してください。
- **評価ポイント**：
- ‘distance‘ が 30 以上 50 未満の値でランダムに設定されていること。
- ‘angleSpeed‘ が 0.02 以上 0.08 未満の値でランダムに設定されていること。
- ‘bodyColor‘ が白色に設定されていること。
- 半径が 2 に設定されていること。

4. **衛星の動作と描画**:
- **概要**: 衛星が惑星の周りを正しく公転し、描画されているか確認してください。
- **評価ポイント**：
- 衛星が親惑星の周囲を指定されたパラメータに基づいて公転していること。
- 描画位置が正しく計算され、衛星がスムーズに動いていること。

5. **太陽系の構築**:
- **概要**: 指定されたパラメータに基づいて、太陽、惑星、衛星が正しく配置されているか確認してください。
- **評価ポイント**：
- 中央に ‘Sun‘ オブジェクトが配置されていること。
- **惑星 1**:
- ‘distance‘ が 150、半径が 10、‘angleSpeed‘ が 0.01 の惑星が存在すること。
- この惑星に 1 つの衛星が存在すること。
- **惑星 2**:
- ‘distance‘ が 250、半径が 10、‘angleSpeed‘ が 0.008 の惑星が存在すること。
- この惑星に 2 つの衛星が存在すること。
- **惑星 3**:
- ‘distance‘ が 350、半径が 20、‘angleSpeed‘ が 0.006 の惑星が存在すること。
- この惑星に 72 個の衛星が存在すること。

**出力形式**: ・**満たされていない理由**のみをリストアップしてください。**評価基準そのものの記載は不要です**。**評価基準が満たされた場合は、その基準については記載しないでください**。・報告の最後に、**満たされていない評価基準の個

数を「合計点数: X」として記載**してください。ただし、満たされていない基準が 3 つ以上ある場合は「合計点数: 3」と記載してください。満たされている基準についての点数は含めないでください。・全ての基準を満たしている場合は、「合計点数: 0」とだけ出力してください。・誤った情報の提供は厳禁です。・修正方法や改善策は記載しないでください。・具体的なコードの記載は一切しないでください。

出力例:

- Hello ではなく Hey となっている。
- 縦幅が 300 である。

合計点数: 2

別の出力例:

合計点数: 0

## User プロンプト

以下のコードを評価してください。

コード:

（提出されたコードの文字列）

# 参考文献


[1] p5.js. `https://p5js.org/` (2024/8/31 確認).

[2] Tweakable: an online programming environment for audio and video, 2022. Accessed 2024/11/02.

[3] Aaron Bangor, Philip T. Kortum, and James T. Miller. An empirical evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[4] M. Abolnejadian, S. Alipour, and K. Taeb. Leveraging ChatGPT for Adaptive Learning through Personalized Prompt-based Instruction: A CS1 Education Case Study. *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems(CHI EA'24)*, (521):1–8, 2024.

[5] Anthropic. The Claude 3 Model Family: Opus, Sonnet, Haiku. *Anthropic Blog*, pp. 1–42, 2024.

[6] R. Balse, B. Valaboju, S. Singhal, J. M. Warriem, and P. Prasad. Investigating the Potential of GPT-3 in Providing Feedback for Programming Assessments. *In Proceedings of the Conference on Innovation and Technology in Computer Science Education (ITiCSE'23)*, pp. 292–298, 2023.

[7] J. D. Bayliss and S. Strout. Games as a "flavor" of CS1. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, p. 500–504, New York, NY, USA, 2006. Association for Computing Machinery.

[8] R. E. Beck, J. Burg, J. M. Heines, and B. Manaris. Computing and music: a spectrum of sound. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, p. 7–8, New York, NY, USA, 2011. Association for Computing Machinery.

[9] B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prather, and E. A. Santos. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, p. 500–506, New York, NY, USA, 2023. Association for Computing Machinery.

[10] D. Boud and E. Molloy. *Feedback in Higher and Professional Education*. Routledge, 2012.

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language Models are Few-Shot Learners. *In Advances in Neural Information Processing Systems*, pp. 1–75, 2020.

[12] C. Burgess, D. Lockton, M. Albert, and D. Cardoso Llach. Stamper: An Artboard-Oriented Creative Coding Environment. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, p. 1–9, New York, NY, USA, 2020. Association for Computing Machinery.

[13] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating Large Language Models Trained on Code. *arXiv:2107.03374*, 2021.

[14] J. Clune, V. Ramamurthy, R. Martins, and U. A. Acar. Program Equivalence for Assisted Grading of Functional Programs. *ACM Programming Languages*, 4(171):1–29, 2020.

[15] P. Denny, V. Kumar, and N. Giacaman. 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*, p. 1136–1142, New York, NY, USA, 2023. Association for Computing Machinery.

[16] Enkelejda Kasneci and Kathrin Seßler and Stefan Küchemann and Maria Bannert and Daryna Dementieva and Frank Fischer and Urs Gasser and Georg Groh and Stephan Günnemann and Eyke Hüllermeier, et al. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, 2023.

[17] GeminiTeamGoogle. Gemini: A Family of Highly Capable Multimodal Models. *arxiv:2312.11805*, pp. 1–90, 2024.

[18] S. George and P. Dewan. NotebookGPT – Facilitating and Monitoring Explicit Lightweight Student GPT Help Requests During Programming Exercises. *In Companion Proceedings of the 29th International Conference on Intelligent User Interface (IUI'2024 Companion)*, pp. 62–65, 2024.

[19] A. Gordillo. Effect of an Instructor-Centered Tool for Automatic Assessment of Programming Assignments on Students' Perceptions and Performance. *Sustainability*, 11(20):1–24, 2019.

[20] I. Greenberg, D. Kumar, and D. Xu. Creative coding and visual portfolios for CS1. *In Proceedings of the 43rd ACM technical symposium on Computer Science Education (SIGCSE'12)*, pp. 247–252, 2012.

[21] M. Guzdial. Does contextualized computing education help? *ACM Inroads*, 1(4):4–6, Dec. 2010.

[22] M. Guzdial and A. Forte. Design process for a non-majors computing course. *SIGCSE Bull.*, 37(1):361–365, Feb. 2005.

[23] Hagit Gabbay and Anat Cohen. Combining LLM-Generated and Test-Based Feedback in a MOOC for Programming. *In Proceedings of the Eleventh ACM Conference on Learning @ Scale (L@S'24)*, pp. 177–187, 2024.

[24] A. Hellas, J. Leinonen, S. Sarsa, C. Koutcheme, L. Kujanpää, and J. Sorva. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1 (ICER'23)*, pp. 93–105, 2023.

[25] X. Hou, Z. Wu, X. Wang, and B. J. Ericson. Codetailor: LLM-powered personalized Parsons puzzles for engaging support while learning programming. In *Proceedings of the Eleventh ACM Conference on Learning@ Scale (L@S'24)*, pp. 51–62, 2024.

[26] Hundredrabbits. Orca, 2021. Accessed 2024/11/02.

[27] M. Jakesch, A. Bhat, D. Buschek, L. Zalmanson, and M. Naaman. Co-Writing with Opinionated Language Models Affects Users' Views. *In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI'23)*, (111):1–15, 2023.

[28] James Prather and Brent N. Reeves and Paul Denny and Brett A. Becker and Juho Leinonen and Andrew Luxton-Reilly and Garrett Powell and James Finnie-Ansley and Eddie Antonio Santos. "It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *ACM Trans. Comput.-Hum. Interact.*, 31(1), Nov. 2023.

[29] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of Hallucination in Natural Language Generation. *In ACM Computing Surveys*, 55(12):1–38, 2023.

[30] P. Jiang. Positional Control in Node-Based Programming. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2023. Association for Computing Machinery.

[31] B. John. SUS: A 'Quick and Dirty' Usability Scale. *Usability Evaluation In Industry*, p. 207–212, 1996.

[32] M. Jonsson and J. Tholander. Cracking the code: Co-coding with AI in creative programming education. In *Proceedings of the 14th Conference on Creativity and Cognition*, p. 5–14, New York, NY, USA, 2022. Association for Computing Machinery.

[33] M. Jonsson and J. Tholander. Cracking the code: Co-coding with AI in creative programming education. *In Proceedings of the 14th Conference on Creativity and Cognition (CC'22)*, pp. 5–14, 2022.

[34] Juho Leinonen and Arto Hellas and Sami Sarsa and Brent Reeves and Paul Denny and James Prather and Brett A. Becker. Using Large Language Models to Enhance Programming Error Messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, p. 563–569, New York, NY, USA, 2023. Association for Computing Machinery.

[35] Julia M. Markel, Steven G. Opferman, James A. Landay, and Chris Piech. Gpteach: Interactive TA Training with GPT-based Students. In *Proceedings of the Tenth ACM Conference on Learning @ Scale (L@S'23)*, pp. 226–236, 2023.

[36] B. Jury, A. Lorusso, J. Leinonen, P. Denny, and A. Luxton-Reilly. Evaluating LLM-generated Worked Examples in an Introductory Programming Course. In *Proceedings of the 26th Australasian Computing Education Conference*, p. 77–86, New York, NY, USA, 2024. Association for Computing Machinery.

[37] S. Kabir, D. N. Udo-Imeh, B. Kou, and T. Zhang. Is Stack Overflow Obsolete? An Empirical Study of the Characteristics of ChatGPT Answers to Stack Overflow Questions. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'24)*, pp. 1–17, 2024.

[38] J. Kato and M. Goto. Lyric App Framework: A Web-based Framework for Developing Interactive Lyric-driven Musical Applications. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2023. Association for Computing Machinery.

[39] J. S. Kay. Contextualized approaches to introductory computer science: the key to making computer science relevant or simply bait and switch? In *Proceedings of*

*the 42nd ACM Technical Symposium on Computer Science Education*, p. 177–182, New York, NY, USA, 2011. Association for Computing Machinery.

[40] M. Kazemitabaar, J. Chow, C. K. T. Ma, B. J. Ericson, D. Weintrop, and T. Grossman. Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'23)*, (455):1–23, 2023.

[41] M. Kazemitabaar, X. Hou, A. Henley, B. J. Ericson, D. Weintrop, and T. Grossman. 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*, New York, NY, USA, 2024. Association for Computing Machinery.

[42] M. Khalil and E. Er. Will ChatGPT get you caught? Rethinking of Plagiarism Detection, 2023.

[43] C. Koutcheme and A. Hellas. Propagating Large Language Models Programming Feedback. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, p. 366–370, New York, NY, USA, 2024. Association for Computing Machinery.

[44] S. Krusche and A. Seitz. ArTEMiS: An Automatic Assessment Management System for Interactive Learning. *SIGCSE'18*, pp. 284–289, 2018.

[45] S. Lau and P. Guo. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1*, p. 106–121, New York, NY, USA, 2023. Association for Computing Machinery.

[46] Lawrence M. Fisher. Booming enrollments. *Communications of the ACM*, 59(7):17–18, 2016.

[47] Linda J. Sax and Kathleen J. Lehman and Christina Zavala. Examining the Enrollment Growth: Non-CS Majors in CS1 Courses. In *Proceedings of the 2017*

*ACM SIGCSE Technical Symposium on Computer Science Education*, p. 513–518, New York, NY, USA, 2017. Association for Computing Machinery.

[48] Linda Mannila and Valentina Dagiene and Barbara Demo and Natasa Grgurina and Claudio Mirolo and Lennart Rolandsson and Amber Settle. Computational Thinking in K-9 Education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, p. 1–29, New York, NY, USA, 2014. Association for Computing Machinery.

[49] Lisa Yan and Nick McKeown and Chris Piech. The PyramidSnapshot Challenge: Understanding student process from visual output of programs. *In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE'19)*, pp. 119–125, 2019.

[50] J. Maeda. *Design By Numbers*. MIT Press, Cambridge, MA, 2001.

[51] J. Maeda. *Creative Code: Aesthetics + Computation*. Thames Hudson, 2004.

[52] Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Zachary Henley, Paul Denny, Michelle Craig, and Tovi Grossman. CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs. *In Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI'24)*, (650):1–20, 2024.

[53] M. Malita and E. Schuster. From drawing to coding: teaching programming with processing. *J. Comput. Sci. Coll.*, 35(8):245–246, Apr. 2020.

[54] A. M. Mcnutt, A. Outkine, and R. Chugh. A Study of Editor Features in a Creative Coding Classroom. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2023. Association for Computing Machinery.

[55] B. Moskal, D. Lurie, and S. Cooper. Evaluating the effectiveness of a new instructional approach. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, p. 75–79, New York, NY, USA, 2004. Association for Computing Machinery.

[56] H. Nguyen and V. Allan. Using GPT-4 to Provide Tiered, Formative Code Feedback. *In Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE'24)*, pp. 958–964, 2024.

[57] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, p. 249–256, New York, NY, USA, 1990. Association for Computing Machinery.

[58] Nischal Ashok Kumar and Andrew Lan. Using Large Language Models for Student-Code Guided Test Case Generation in Computer Science Education, 2024.

[59] OpenAI. Introducing ChatGPT. `https://openai.com/blog/chatgpt` (2024/10/28 確認).

[60] OpenAI. Introducing OpenAI o1-preview. `https://openai.com/index/introducing-openai-o1-preview`（2024/10/28 確認）.

[61] OpenAI. GPT-4 Technical Report. *OpenAI Blog*, pp. 1–100, 2023.

[62] J. A. Oravec. Artificial intelligence implications for academic cheating: Expanding the dimensions of responsible human-AI collaboration with ChatGPT. *Journal of Interactive Learning Research*, 34(2):213–237, 2023.

[63] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. *In Advances in Neural Information Processing Systems*, pp. 1–15, 2022.

[64] M. Pankiewicz and R. S. Baker. Large Language Models (GPT) for automating feedback on programming assignments. *arXiv preprint arXiv:2307.00150*, 2023.

[65] K. Peppler and Y. Kafai. Creative Coding: Programming for Personal Expression. In *Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL)*, pp. 30–7, 2009.

[66] J. Prather, P. Denny, J. Leinonen, B. A. Becker, I. Albluwi, M. Craig, H. Keuning, N. Kiesler, T. Kohn, A. Luxton-Reilly, S. MacNeil, A. Petersen, R. Pettit, B. N.

Reeves, and J. Savelka. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*, p. 108–159, New York, NY, USA, 2023. Association for Computing Machinery.

[67] R. A. P. Queirós and J. P. Leal. PETCHA: a Programming Exercises Teaching Assistant. *ACM Innovation and Technology in Computer Science Education (ITiCSE'12)*, pp. 192–197, 2012.

[68] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving Language Understanding by Generative Pre-Training. *OpenAI Blog*, pp. 1–12, 2018.

[69] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, pp. 1–24, 2019.

[70] C. Reas and B. Fry. Processing: a learning environment for creating interactive Web graphics. *In ACM SIGGRAPH 2003 Web Graphics(SIGGRAPH'03)*, 2003.

[71] Rishabh Singh and Sumit Gulwani and Armando Solar-Lezama. Automated feedback generation for introductory programming assignments. *In Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'13)*, pp. 15–26, 2013.

[72] A. Salga, D. Hodgin, A. Sobiepanek, S. Downe, M. Medel, and C. Leung. Processing.js: sketching with <canvas >. *In ACM SIGGRAPH 2011 Talks (SIGGRAPH'11)*, (15):1, 2011.

[73] Sam Lau and Philip J. Guo. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. *In Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1 (ICER'23)*, pp. 106–121, 2023.

[74] S. Sarsa, P. Denny, A. Hellas, and J. Leinonen. 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*, p. 27–43, New York, NY, USA, 2022. Association for Computing Machinery.

[75] D. Shifman. Coding Train, 2021.

[76] A. Singla. Evaluating ChatGPT and GPT-4 for Visual Programming. *In Proceedings of the 2023 ACM Conference on International Computing Education Research (ICER'23)*, 2:14–15, 2023.

[77] A. Smolansky, A. Cram, C. Raduescu, S. Zeivots, E. Huber, and R. F. Kizilcec. Educator and Student Perspectives on the Impact of Generative AI on Assessments in Higher Education. In *Proceedings of the Tenth ACM Conference on Learning @ Scale*, p. 378–382, New York, NY, USA, 2023. Association for Computing Machinery.

[78] Stephen MacNeil and Andrew Tran and Arto Hellas and Joanne Kim and Sami Sarsa and Paul Denny and Seth Bernstein and Juho Leinonen. Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, p. 931–937, New York, NY, USA, 2023. Association for Computing Machinery.

[79] T. Terroso and M. Pinto. Programming for Non-Programmers: An Approach Using Creative Coding in Higher Education. *In Third International Computer Programming Education Conference (ICPEC'22)*, 102(13):1–8, 2022.

[80] Tricia J. Ngoon and C. Ailie Fraser and Ariel S. Weingarten and Mira Dontcheva and Scott Klemmer. Interactive Guidance Techniques for Improving Creative Feedback. *In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI'18)*, (55):1–11, 2018.

[81] Tung Phung and Victor-Alexandru Pădurean and Anjali Singh and Christopher Brooks and José Cambronero and Sumit Gulwani and Adish Singla and Gustavo Soares. Automating Human Tutor-Style Programming Feedback: Leveraging GPT-4 Tutor Model for Hint Generation and GPT-3.5 Student Model for Hint Validation. In *Proceedings of the 14th Learning Analytics and Knowledge*

*Conference*, p. 12–23, New York, NY, USA, 2024. Association for Computing Machinery.

[82] Tung Phung and Victor-Alexandru Pădurean and José Cambronero and Sumit Gulwani and Tobias Kohn and Rupak Majumdar and Adish Singla and Gustavo Soares. Generative AI for Programming Education: Benchmarking ChatGPT, GPT-4, and Human Tutors. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 2*, p. 41–42, New York, NY, USA, 2023. Association for Computing Machinery.

[83] X. Wang, H. Kim, S. Rahman, K. Mitra, and Z. Miao. Human-LLM Collaborative Annotation Through Effective Verification of LLM Labels. *In Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI'24)*, (303):1–21, 2024.

[84] J. M. Wing. Computational Thinking. *Communications of the ACM*, 49(1):33–35, 2008.

[85] Z. J. Wood, P. Muhl, and K. Hicks. Computational Art: Introducing High School Students to Computing via Art. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, p. 261–266, New York, NY, USA, 2016. Association for Computing Machinery.

[86] R. Xiao, X. Hou, and J. Stamper. Exploring How Multiple Levels of GPT-Generated Programming Hints Support or Disappoint Novices. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pp. 1–10, 2024.

[87] Yang, Stephanie and Zhao, Hanzhang and Xu, Yudian and Brennan, Karen and Schneider, Bertrand. Debugging with an AI Tutor: Investigating Novice Help-seeking Behaviors and Perceived Learning. In *Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1*, p. 84–94, New York, NY, USA, 2024. Association for Computing Machinery.

[88] Yasuichi Nakayama and Yasushi Kuno and Hiroyasu Kakuda. Split-Paper Testing: A Novel Approach to Evaluate Programming Performance. *Journal of Information Processing*, 28:733–743, 2020.

[89] 久野靖. 短冊型問題を用いたプログラミング学習アドバイスツール. 情報処理学会 Programming Symposium 報告集, pp. 129–139, 2020.

[90] 山本利一, 本郷健, 本村猛能, 永井克昇. 初等中等教育におけるプログラミング教育の教育的意義の考察. 教育情報研究, 32(2):3–12, 2016.

[91] 漆原宏丞, 本多佑希, 岸本有生, 兼宗進. 抽象構文木を利用したプログラミング理解度採点の試み. 情報処理学会 研究報告コンピュータと教育（CE）, 2021-CE-162(16):1–6, 2021.

[92] 若谷彰良, 前田利之. 生成 AI を用いた C 言語プログラミング学習のための助言システムの試作. 甲南大学紀要 知能情報学編, 16(2):7–16, 2024.

[93] 小原有以, 佐藤美唯, 倉光君郎. KOGI: ChatGPT を Colab に統合したプログラミング演習支援. 情報処理学会 情報教育シンポジウム論文集, pp. 141–148, 2023.

[94] 小川弘迪, 小林亜樹. プログラミング課題の自動採点に向けた構文木上のカーネル法による類似度関数の提案. 情報処理学会 第 80 回全国大会講演論文集, 2018(1):661–662, 2018.

[95] 新田章太, 小西俊司, 竹内郁雄. 複数言語に対応しやすいオンラインプログラミング学習・試験システム track. 情報処理学会 情報教育シンポジウム論文集, pp. 114–121, 2019.

[96] 森陽菜, 松澤芳昭. ChatGPT を利用したプログラミング教育支援システム「ChotGPT」の提案と評価. 情報処理学会 研究報告コンピュータと教育（CE), 2024-CE-174(7):1–8, 2024.

[97] 石原浩一, 泰山裕. フィードバックと振り返りが学習者の認知欲求に及ぼす影響の検討. 日本教育工学会論文誌, 44(1):105–113, 2020.

[98] 文部科学省. 中央教育審議会 大学分科会 制度部会（第 22 回（第 3 期第 7 回））議事録・配付資料 ［資料 2 － 1］ ティーチング・アシスタント（TA）について. https://www.mext.go.jp/b_menu/shingi/chukyo/chukyo4/003/gijiroku/07011713/001/002.htm (2024/10/28 確認).

[99] 又吉康綱, 中村聡史. askTA: 消極性を考慮したオンライン演習講義支援システム. コンピュータソフトウェア, 39(1):55–71, 2022.

[100] 尹子旗, 王昊, 堀尾海斗, 河原大輔, 関根聡. プロンプトの丁寧さと大規模言語モデルの性能の関係検証. 言語処理学会 第 30 回年次大会, pp. 1–6, 2024.

# 研究業績

[1] 野中滉介, 関口祐豊, 小松原達哉, 桑原樹蘭, 中村聡史. コミクエ: 新刊読書時に前巻までの流れを想起可能とするクイズ共有手法の提案. 第 6 回 コミック工学研究会, pp.63-68, 2021.

[2] 関口祐豊, 植木里帆, 横山幸大, 中村聡史. 三択の選択肢の色の組み合わせが選択行動に及ぼす影響. 情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI）, Vol.2021-HCI-195, No.32, pp.1-8, 2021.

[3] 守安真也, 佐竹雪乃, 関口祐豊, 小林稔. TRing：休憩によるやる気低下を防ぐ行動支援システムの提案. 情報処理学会 研究報告グループウェアとネットワークサービス（GN）, Vol.2022-GN-116, No.22, pp.1-7, 2022.

[4] 木下裕一朗, 関口祐豊, 植木里帆, 横山幸大, 中村聡史. 選択インタフェースにおけるアイテムの遅延表示が選択に及ぼす影響. 情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI）, Vol.2022-HCI-200, No.27, pp.1-8, 2022.

[5] 小林沙利, 植木里帆, 関口祐豊, 中村聡史, 掛晃幸, 石丸築. デジタルペンの筆圧による濃淡表現の有無が筆算の正答率に及ぼす影響. HCG シンポジウム 2022, No.C-5-5, pp.1-8, 2022.

[6] 関口祐豊, 植木里帆, 中村聡史. PP-Undo: 筆圧の制御により付与されたストロークの確信度に基づく Undo/Redo 手法の提案, 研究報告ヒューマンコンピュータインタラクション（HCI）, Vol.2023-HCI-201, No.15, pp.1-8, 2023.

[7] 木下裕一朗, 関口祐豊, 植木里帆, 横山幸大, 中村聡史. 選択肢の時間差表示が選択行動に及ぼす影響. 信学技報 ヒューマンコミュニケーション基礎研究会（HCS）, 2023.

[8] 田中佑芽, 関口祐豊, 櫻井翼, 小松原達哉, 中村聡史. コミクエ: 漫画の内容のクイズ作成が既読巻の想起に与える影響. 情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI）, Vol.2023-HCI-204, No.1, pp.1-8, 2023.

[9] Yuto Sekiguchi, Riho Ueki, Kouta Yokoyama, Satoshi Nakamura. Does the Average Color Influence Selection?. International Conference on Human-Computer Interaction (HCII 2023), Vol.LNCS, volume 14012, pp.485-496, 2023.

[10] Yuichiro Kinoshita, Yuto Sekiguchi, Riho Ueki, Kouta Yokoyama, Satoshi Nakamura. Do People Tend to Select a Delayed Item?. International Conference on Human-Computer Interaction (HCII 2023), Vol.LNCS, volume 14012, pp.397-407, 2023.

[11] 関口祐豊, 中村聡史. PP-Undo: 筆圧を軸とした Undo 機能のためのストローク群に対する一括筆圧情報付与手法の提案とその評価. 情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI）, Vol.2023-HCI-204, No.14, pp.1-8, 2023.

[12] 関口祐豊, 中村聡史. PP-Undo+: 筆圧を軸とした手書き編集手法. 第 31 回インタラクティブシステムとソフトウェアに関するワークショップ, 八ヶ岳, 2023.

[13] 櫻井翼, 田中佑芽, 関口祐豊, 中村聡史. 漫画の振り返りを支援するクイズとその答えからのシーン推定. HCG シンポジウム 2023, No.B-4-3, 2023.

[14] 田中佑芽, 関口祐豊, 櫻井翼, 中村聡史. コミクエ: 他者が作成したクイズが漫画の既読巻の想起に及ぼす影響. 情報処理学会 研究報告エンタテインメントコンピューティング（EC）, Vol.2024-EC-71, No.36, pp.1-7, 2024.

[15] Yume Tanaka, Yuto Sekiguchi, Tsubasa Sakurai, Satoshi Nakamura. ComiQA: A Comic Question-Answer Sharing System that Helps Users to Recollect the Content of Previous Volumes. 28th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2024), 2024.

[16] Tsubasa Sakurai, Yume Tanaka, Yuto Sekiguchi, Satoshi Nakamura. Manga Scene Estimation by Quiz and Answer. 28th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2024), 2024.

[17] 関口祐豊, 中村聡史. PP-Checker: プログラミング教育における大規模言語モデルと協調した曖昧性のある自動採点システム. 第 32 回インタラクティブシステムとソフトウェアに関するワークショップ（WISS 2024）.

[18] 瀬崎夕陽, 関口祐豊, 中村聡史. 微細な筆圧変化を用いたシート判別手法の提案. 情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI）, Vol.2025-HCI-211, No.19, pp.1-8, 2025.

[19] 能宗巧, 瀬崎夕陽, 小林沙利, 関口祐豊, 中村聡史, 近藤葉乃香, 梅澤侑己, 橋本忠樹. 書き心地の改善に向けたペン先の摩擦が筆記のブレに及ぼす影響. 情報処理学会 研究報告ヒューマンコンピュータインタラクション（HCI）, Vol.2025-HCI-211, No.15, pp.1-8, 2025.

[20] 木下裕一朗, 関口祐豊, 中村聡史. 選択肢表示のズレが選択行動に及ぼす影響. 情報処理学会論文誌, Vol.66, No.2, pp.1-9, 2025.

[21] 関口祐豊, 中村聡史. PP-Checker 1.5: LLM との協働により標準出力を含むプログラムを評価する半自動採点システム. インタラクション 2025 論文集, to appear, 2025.