LEARNING : Jurnal Inovasi Penelitian Pendidikan dan Pembelajaran
Vol. 5 No. 2 Mei 2025
E-ISSN : 2777-0575
P-ISSN : 2777-0583
Online Journal System : https://jurnalp4i.com/index.php/learning

HUMAN-CHATGPT CO-CREATED SIMULATIONS: DROP-OFF ZONE KINEMATICS CASE STUDY

Edeline Priscilla Yusuf[1], Eddy Yusuf[2]
Santa Laurensia School, Alam Sutera – Serpong, Indonesia[1], Department of Informatics, Universitas Ciputra Surabaya, Indonesia[2]
e-mail: eddy.yusuf@ciputra.ac.id

ABSTRAK

Tulisan ini menampilkan proyek kolaboratif antara manusia dan ChatGPT untuk menghasilkan simulasi *drop-off* menggunakan Python. Proyek ini selaras dengan konsep *vibe coding* yang dipopulerkan oleh salah satu pendiri OpenAI, Andrej Karpathy, untuk menggambarkan pengembangan kode secara iteratif melalui percakapan manusia dan AI generatif. Kode akhir dihasilkan melalui proses iteratif dan penyempurnaan bertahap melalui *prompting* di ChatGPT. Simulasi lalu lintas drop-off dipilih sebagai studi kasus untuk menunjukkan bagaimana kompleksitas pemikiran manusia dapat diterjemahkan ke dalam Python melalui proses bertahap. Setiap tahap pengembangan didokumentasikan secara cermat dalam bentuk kode, *prompt*, respons ChatGPT, tangkapan layar, dan video yang tersedia di GitHub (https://github.com/eddy-yusuf/dropoff). Setiap tahap merekam tonggak perkembangan dari satu mobil yang bergerak secara uniform di jalur lurus menjadi 20 mobil yang saling berinteraksi dengan dinamika kompleks, termasuk perlambatan, berhenti, dan akselerasi di titik *drop-off*. Temuan penting dari proyek ini adalah bahwa ChatGPT tidak hanya berperan sebagai pembuat kode, tetapi juga sebagai mitra kognitif yang mampu ikut merancang simulasi dengan kemampuan bernalar dan berpikir sistematis. Karya ini dapat menjadi model bagi pendidikan atau desain pemrograman yang diperkuat oleh GenAI, sekaligus sebagai pertanyaan kritis bagi para pendidik terkait pedagogi pemrograman di era AI generatif.
Kata Kunci: *ChatGPT, AI, AI generatif, Vibe Coding, Simulasi, Lalu Lintas, Drop-off*

ABSTRACT

The paper reports a collaborative Human-ChatGPT project to build a drop-off simulation in Python. This project aligns with the concept of vibe coding, popularized by OpenAI co-founder Andrej Karpathy, which describes the iterative, conversational development of code between human reasoning and generative AI. The final code was produced through iterative development and refinement via prompting in ChatGPT. We consider a drop-off traffic simulation as the use case, demonstrating how the increasing complexity of human reasoning can be translated into Python through iterative process. We carefully documented each stage of the project with codes, prompts, ChatGPT's responses, screenshots, and videos in GitHub (https://github.com/eddy-yusuf/dropoff). Each stage captures development milestones which evolve from a single car moving uniformly on a straight lane to 20 interacting cars with complex dynamics including deceleration, stopping, and accelerating at drop-off point. A key finding is that ChatGPT functions not only as a code generator but also as a cognitive partner capable of co-designing the simulation with systematic thinking and reasoning capability. This work can serve as a model for GenAI-enchanced programming education or design. It also poses a critical question for educators about coding pedagogy in the era of Generative AI.
Keywords: ChatGPT, AI, Generative AI, Vibe Coding, Simulation, Traffic, Drop-off

INTRODUCTION

Programming languages have evolved over time, from ENIAC coding system first introduced in 1940s to PASCAL in 1978, and more recently Mojo in 2023. Each language was

LEARNING : Jurnal Inovasi Penelitian Pendidikan dan Pembelajaran
Vol. 5 No. 2 Mei 2025
E-ISSN : 2777-0575
P-ISSN : 2777-0583
Online Journal System : https://jurnalp4i.com/index.php/learning

designed to address specific usages, shaped by the technological context of its era, and established by distinct design philosophies. Today, the popularity of programming languages is largely dictated by market demand, developer experience, community and ecosystem strength, as well as industrial backing and adoption channels. According to TIOBE Community Index (TIOBE, 2025), Python ranks as the most popular language programming in April 2025 based on the search frequency across multiple search engines.

Students enrolled in Computer Science, Informatics, or Information Systems programs are required to master one of these programming languages to smoothly transition to workforce. Unfortunately, learning to code can be a serious challenge for beginners because traditional programming courses heavily focus on syntax, semantics, and structure (Sands, 2019; Stachel et al., 2013). Recent works (Cheah, 2020; Kadar et al., 2021) confirm that the conventional approaches become the major contributor to why programming is difficult. It is necessary to have a pedagodical reform in teaching programming: moving away from syntax-, semantics-, and structure-focused to conceptual understanding and problem-solving pedagogy.

The rapid development of Generative AI (GenAI) in the past year has redefined the landscape of programming. GenAI tools like ChatGPT force educators to transition from low-level syntax-focused coding to reasoning-based and prompt-driven learning that emphasizes co-creation in the learning process (Dong et al., 2024; Xue et al., 2024). Students can have 24/7 access to ChatGPT which allows faster and more efficient feedback while at the same time conforms to individual learning pace (Kok & Gan, 2023; Yilmaz et al., 2023). However there is a risk that students exhibit exececssive dependence over ChatGPT which can lead to shallow understanding and wrong practices such as blind copying (Haindl & Weinberger, 2024; Sun et al., 2024). While the development of ChatGPT is exponential, it is succeptible to hallucinations and can misunderstand complex problems due to limited reasoning (Guo et al., 2024; Silva et al., 2024). Most students welcome the adoption of ChatGPT in the learning process as it can provide real-time feedback and flexible pacing, while at the same time they understand the boundaries of such tool (Biswas, 2023). Faculty's perspectives are mix-bag: some embrace it to redesign assessment and to promote AI literacy, while others express concerns about ethics and pedagogy disruption (Bucaioni et al., 2024).

The majority of existing research on the use of ChatGPT for coding has been concentrated on solving isolated problems, debugging syntax errors, or generating small code fragments. This positions ChatGPT only as a supplementary assistant within the traditional curriculum framework. In contrast, the present work demonstrates ChatGPT as a cognitive partner, co-creating a complex and multi-stage code development to simulate a realistic drop-off traffic dynamics. The simulation was entirely developed through iterative prompt-response cycles which highlights the code development through natural language reasoning rather conventional syntax-driven instructions. We documented every step of the process to ensure reproducibility of the project and as a starting point for further exploration of ChatGPT as a cognitive collaborator in coding pedagogy.

METHODS

This work adopts autoethnographic approach (Chang, 2008) to showcase the firsthand experience in creating a simulation in collaboration with ChatGPT. In this approach, we serve as both subject and narrator of the co-creation journey to develop a Python-based simulation from scratch. We used ChatGPT-4o for Python code co-creation, Jupyter notebook for executing the code, and GitHub repository for complete documentation of the process. There was no pre-existing code or template used in this project; the entire project began from a blank slate and evolved solely through iterative prompting.

The use case in this work is to develop a fully functioning model of a traffic flow at a school drop-off point. We began with a simple model of a single car moving at a constant speed along a straight lane, then gradually increased the complexity across 8 stages, resulting in a 20-car simulation, each exhibitng realistic behavior such as maintaining safe distance, slowing down, stopping, and accelerating after drop-off. Each stage followed a prompt-response-test-revise loop to ensure that code aligned with the user's logic. We recorded the user's prompts, ChatGPT's responses, the generated codes, as well as screenshots and videos in each simulation stage. Each stage also serves as reflection points and analysis for improvement. All materials are publicly available via GitHub repository (GitHub (https://github.com/eddy-yusuf/dropoff).

RESULTS AND DISCUSSION

Results

We present the results on Human-ChatGPT interaction with the use case of co-creating a realistic drop-off simulation in this section. We document the development stages in detail, all publicly available, at GitHub repository (https://github.com/eddy-yusuf/dropoff/). The repository hosts raw user's prompts, unedited ChatGPT responses, the generated code by ChatGPT, screenshots of the animation, and the resulting animated videos. We purposedly include the original prompts and responses in the repository to showcase the true nature of human-ChatGPT dialogue. Through this preserved conversation, we highlight how the simulation logic emerges from pure human machine interaction: how well ChatGPT interprets human intention, constructs behavior, and translates it into executable Python codes. The kinematics parameters (stop duration, safe distance, acceleration, and others) are not the main concern here as they are simply a nuance and can be easily tweaked if needed. As we will show in the preceeding discusion, revision and alignment processes emphasize the co-creative nature of prompt engineering which are the key aspect of this work.

Table 1 shows 6 major milestones of the code development in the co-creation of drop-off simulation. The simulation progresses through 8 development stages which are labeled S(x)V(y) where x is the stage number and y is the verson; for example, S2V1 refers to the simulation in Stage 2 and Version 1. Throughout the document, we will use S(x)V(y) notation for clarity.

Table 1. The simulation development consists of 6 major milestones which are divided into 8 stages

No	Milestone	Stage
1	A single car moves with a constant speed on a straight lane.	S1V1
2	Drop-off simulation scenario for a single car: i) drop-off point is introduced, ii) a single car follows slow-down, stop, accelerate behavior.	S2V1, S2V2
3	Two cars interact with each other in the drop-off scenario.	S3V1, S3V2
4	Multiple cars interact with each other in the drop-off scenario.	S4V1, S5V1, S5V2, S6V1
5	Introduce randomized kinematics parameters in the simulation.	S7V1
6	Scale up the simulation to include 20 cars which operate smoothly in a drop-off scenario.	S8V1

We now discuss the development stages in detail; the focus is on the human-ChatGPT interaction where each stage highlights user's prompts, ChatGPT's interpretation (or misinterpretation) and response, the simulation logic, and the resulting outcomes. Excerpts from user's prompts and ChatGPT responses are included to ilustrate this interaction. The full conversation is publicly available at GitHub repository https://github.com/eddy-yusuf/dropoff/. We keep incremental developments in each stage and/or version to ensure the complex simulation progresses in a controlled and traceable manner.

1. S1V1

We began the simulation by designing a simple single car simulation cruising with a constant speed on a straight lane. We instructed ChatGPT to produce this simulation by prompting "*I want to make a transport simulation. Create a single lane with a good graphic. Create a red dot that represents a car, it moves from left to right with constant velocity, the output is animation in Jupyter Notebook, code is in Python*". ChatGPT correctly interpreted this instruction and generated a Python-code to anímate a car (represented by a red dot) moving with a constant speed across a gray lane. No revision is necessary at this stage. S1V1 is our first milestone (Table 1) which serves as the baseline model to incorporate more complex behaviors in subsequent stages.

2. S2V1

The second stage of the simulation development was to introduce drop-off point mid lane, simulate a simple stop-go kinematics at that point, and improve visualization. We instructed ChatGPT to simulate this behavior by prompting "*Create a drop off poin in the middle…*" and "*When the car is approaching the drop off point, it will slow down, stop, and then accelerate…". We also prompted "Make the red dot smaller" and "The lane is white, not filled with gray…"* to improve the visual of the simulation in S1V1.

ChatGPT interpreted the user's prompt and responded "*Smaller red dot*", "*A "drop-off point" marked in the center*", and "*Car slows down, stops briefly at the drop-off, then accelerates again*". It produced a visually improved stop-go car animation (the car slows down, stops, and accelerates), but it did not stop exactly at the drop-off point, but at an arbitrary distance before it. While ChatGPT appeared to understand the instruction, it misapplied it in the execution. We believe that issue arose from the lack of precision in the initial user's prompt. The prompt "*When the car is approaching the drop off point, it will slow down, stop, and then accelerate to the maximum uniform speed"* left room for interpretation: it did not explicitly state where the car must stop. This situation emphasizes that we need precise and unambiguous language for instruction otherwise ChatGPT will use its prior knowledge to interpret vague prompts which may likely diverge from the intended behavior. This is also consistent with prior study which stated that prompt quality led to high-quality outputs from ChatGPT (Jacobsen & Weber, 2025).

3. S2V2

The second version of Stage 2 (S2V2) aimed to fix the logical issue encountered in S2V1. To fix the issue of the car not stopping exactly at the drop-off point, we prompted "*…I want the car slow down as it approaches the drop-off point and stops exactly at drop-off point for a few seconds…"* where ChatGPT responded by "*The car slows down smoothly and stops exactly at the drop-off point...*". It further added key logical thinking: "*It waits there for a few seconds…*", "*Then it accelerates smoothly…*", and "*We'll use simple kinematics logic…*" which reflected ChatGPT internal reasoning to generate smooth stop-go simulation at the drop-off point.

We learned that ChatGPT summarized improvements after each prompt iteration. Phrases like "*Kinematics-based movement…*", "*Smooth deceleration…*", "*Wait time: stops precisely at the drop-off for 20 frames (~1 second)*", and "*Smooth acceleration out…*" helped the user to trace new added/modified features and to understand the relevant kinematic parameter.

S2V2 concludes the second milestone in our simulation development. It successfully produced a smooth stop-go car simulation with the desired user-design behavior.

4. S3V1

Stage 3 marked a major conceptual shift in the development where now we introduced a two-car interaction logic in the stop-go simulation at the drop-off point. In this simulation, as Car 1 approached the drop-off point, it would slow down, stop at drop-off point, and accelerate again; Car 2, placed at a fixed distance behind Car 1, would adapt to the kinematics of Car 1 and avoid collision with it. Key excerpts from the user's prompt to generate this behavior include *"There is a second car just behind the first one at a certain distance", "It is crucial that the second car maintains that distance… otherwise you have collision", "Car 2 will also slow down and stop, but of course not at drop off point", and "When Car 1 moves away ... then Car 2 will accelerate then decelerate as it approaches the drop off point, stops, then accelerate".*

In response to the user's instruction, ChatGPT responded by "*Car 1 slows down, stops at…*" and "*Car 2 follows Car 1 but never collides, adapting to Car 1's speed profile ... keeping a safe following distance*". ChatGPT also presented key highlights for the user such as the kinematics parameters, the car-following it used in the thinking process, and visual information on the simulation. This information is useful for tweaking or extending the simulation in the future.

The resulting simulation displayed a smooth stop-go behavior for Car 1; while Car 2 maintained a safe distance to Car 1 and adjusted its motion based on Car 1's kinematics, it failed to exhibit sequential stops: it stopped only in response to Car 1 but not at the drop-off point. This result clearly diverged from the user's logic that Car 2 was expected to stop at the drop-off point. The problem can be ascribed to ambiguous prompt the user "*…but of course not at drop off point"* which was interpreted as a direct instruction to skip stopping at the drop-off point entirely. The follow-up prompt "*When Car 1 moves away ... then Car 2 will accelerate then decelerate as it approaches the drop off point, stops, then accelerate"* was supposed to govern the sequential stop kinematics for Car 2 but it was completely disregarded by ChatGPT. It suggests that when given a sequence of instructions, ChatGPT tends to execute them in the order they appear, giving priority to earlier prompts and often disregarding those that follow if they introduce conflicting behavior, consistent with a benchmark test which looks into how LLMs handle conflicting prompts (Zhang et al., 2025).

In the visual aspect, there was a minor issue in S3V1: the two cars in the simulation were not horizontally aligned. The user's prompt did not explicitly define the geometrical alignment of the two cars which led ChatGPT to interpret the layout freely. This is another example, even in the visual representation aspect, on how one needs clear and precise prompts to get the desired outcomes.

5. S3V2

The second version of Stage 3 was designed to take care the issues arose in S3V1. We prompted ChatGPT with "*Make sure the two cars are aligned on the horizontal axis*" to fix the visual misalignment and "*You miss that car 2 has to stop at the drop off point…*" to correct the missing sequential stop logic in S3V1. ChatGPT understood the instruction clearly and responded with "*…We'll align both cars on the same centerline…"* and "*Car 2 stops exactly at the drop-off line*". It summarized the fix which clearly stated that Car 2 now stopped exactly at the drop-off point. In addition, ChatGPT also laid out kinematics parameters such as stop duration and maximum speed, as well as motion logic used.

S3V2 successfully resolved the visual and logical issues encountered in S3V1. The two cars were horizontally aligned, and Car 2 now exhibited the correct stop-go behavior at the drop-off point. This result emphasizes the importance of precise and explicit instructions to guide ChatGPT generate the intended outputs. S3V2 marks the completion of the third milestone in the simulation development where it progressed from a single car stop-go simulation to two-car interaction with a stop-go behavior.

6. S4V1

Stage 4 highlighted the beginning of the fourth milestone (Table 1) in our simulation. We entered a multi-car interaction with the stop-go kinematics successfully implemented in the preceding stages. Stage 4 particularly extended the previous simulation to include the third car positioned behind Car 2. There was no new logic introduced in this stage; ChatGPT was expected to replicate the logic in S3V2 and applied it to Car 3 in the simulation.

We prompted with "*…can you add a third car which has the same mechanism as car 1 and car 2?*". ChatGPT clearly understood the instruction and responded with "*Car 3 follows the same logic as Car 1 and Car 2. Starts behind Car 2…Maintains gap, approaches the drop-off, stops for the same duration, then accelerates away*".

There was no issue observed in the implementation of Stage 4. ChatGPT successfully extended the simulation to include 3 cars following the same logic as in the previous stages.

7. S5V1

Stage 5 continued the scaling up progression to include the fourth car in the simulation. The prompt was "*Can you add a fourth car which again follows the same logic?*" and ChatGPT confirmed the user's instruction to add the fourth car with the same logic "*Car 4: Starts behind Car 3 by a safe gap. Follows the exact same behavior…*". The result showed that Car 4 became stuck at the drop-off point where it lacked a mechanism to decide when it was safe to proceed; it only moved after Car 3 left the frame. Since there was no new logic introduced, the issue may originate from the logical flaw implemented in the codes by ChatGPT. This issue raised a concern on how ChatGPT scales up the logic structure in the multi-car interaction setting.

8. S5V2

The second version of Stage 5 focused on fixing the logical flaw present in S5V1. We prompted ChatGPT to revise the behavior of Car 4 with "*Car 4 becomes stuck at drop-off point and starts moving after Car 3 leaves the frame. Can you fix it*?". ChatGPT correctly diagnosed the issue and responded "*…It never realizes it's safe to move independently to the drop-off once Car 3 has moved far enough…*", after which it continued offering a solution "*…We'll refactor the car-following logic to ensure all cars respect the same rules without getting stuck*".

After fix, the solution worked and Car 4 exhibited the desired behavior: it slowed down, stopped at the drop-off point for a fix duration of time determined by the kinematics parameter, and accelerated away. The update shows that ChatGPT understands human language beyond the surface level and is also capable to analyze behavioral logic. The new behavior logic for Car 4 dictated that it was safe to move after a fixed stop duration and when the space ahead was clear. Thus the prompt "*Can you fix it*" led ChatGPT to transform a passive logic for Car 4 into an autonomous and state-aware behavior for it. It demonstrates ChatGPT's deep reasoning ability where when prompted with error-focus feedback. This finding is supported by a recent study that ChatGPT was capable for deep and context-sensitive reasoning in response to natural language queries (Xie et al., 2024).

9. S6V1

The scale-up continued in this stage where we introduced the fifth car into the simulation. This stage tested the stability of the solution implemented in S5V2. The intention was to verify whether ChatGPT would consistently apply the motion logic from S5V2 across 5 cars without repeating the error identified in S5V1.

We prompted ChatGPT "*Add a fifth car, following the same logic*", from which ChatGPT responded "*All cars will: i) Follow the car…, ii) Slow down and stop exactly at the drop-off…, iii) Wait for the same duration…, iv) Accelerate back…, v) Exit the frame…*". The simulation S6V1 confirmed that all five cars performed the stop-go behavior correctly. No glitch was observed which indicates that ChatGPT succeeded in generalizing the car-following behavior to 5 cars.

The completion of Stage 6 marked the end of the fourth milestone in the simulation. We successfully generated a five-car simulation while maintaining the stop-go kinematics correctly, implementing the car-following model, and ensuring both stability and scalability.

10. S7V1

Stage 7 began the fifth milestone in the simulation. Here we introduced randomized kinematic parameters, namely the stop duration and acceleration, for all five cars. The rationale for this was twofold: i) to simulate a more realistic transport scenario and ii) to stress test the motion logic implemented in S6V1. The prompt instructed ChatGPT to randomize the stop duration and acceleration "*The duration…is random (1 – 51 frames), the acceleration… is also random (0.01 – 0.2)"*. ChatGPT responded that each car will randomly draw a random number for acceleration and stop duration.

The simulation confirmed that randomness was successfully introduced and all five cars maintained the correct stop-go behavior and interacted among themselves according to the car-following model. We observed that the simulation remained stable and no logical or visual errors were encountered.

Stage 7 demonstrated that ChatGPT was capable to incorporate variability in the kinematic parameters without breaking the behavioral logical implemented thus far.

11. S8V1

This stage marked the final stage and the most complex behavior for the simulation. We scaled up the simulation to 20 cars and added the last layer of complexity, e.g. random safe distance. We prompted ChatGPT with human language style "*…I want to have a large number of cars, say, 20 coming from the left, all with the same logic, but with random safe distance*". ChatGPT understood its task and responded *"…Let's do it — 20 cars, each with…",* after which it elaborated the behavioral logic such as each car had random acceleration, stop distance, and stop duration. On the visual aspect, we asked ChatGPT to label each car to allow easy tracking during the simulation "*Can you label each car with number?*". ChatGPT made a special note on this instruction "*Each car has a label (1 to 20). The label moves with the car. The label appears above…*".

The resulting simulation, which was the last milestone in the simulation, showed all 20 cars moving, slowing down, stopping, and accelerating independently with random kinematics. There were no collisions or visual issues observed, thereby validating that the logic was stable despite high complexity. This final stage showcased ChatGPT's capability to understand human language to scale up the simulation to multi-car motion logic with randomized individual behavioral parameters.

Discussion

This study highlights the development of a complex and realistic drop-off simulation entirely through natural language interaction between human and ChatGPT. In the co-creation process, human acts as a designer who conveys structure, intent, and logic while ChatGPT interprets these and generates executable codes in Python. Human analyzes the outputs and identifies parts that need clarification, re-prompting, and logic correction. The iterative process continues until the desired output is achieved. Recent studies emphasize that such human-AI collaboration fosters creative problem-solving by merging human intent with machine efficiency (Lee et al., 2023; Nguyen et al., 2023).

The simulation was divided into 6 major milestones and comprised of 8 stages of development. We learned that precise prompting was key to guide ChatGPT to produce the intended behavior at each stage. Research by Brown et al. (2022) confirms that structured, stepwise prompting reduces ambiguity in AI-generated code by up to 40%, aligning with our findings. ChatGPT demonstrated excellent internal and deep reasoning, including understanding instruction via human language, summarizing logics, identifying flaws, and proposing solutions. This mirrors observations in AI reasoning studies, where language models exhibit "chain-of-thought" capabilities when given explicit task decomposition (Zhang & Wang, 2023).

A critical observation during interaction was ChatGPT’s tendency to follow instructions in the order given, often prioritizing the first instruction if latter ones introduced conflicting logic. This aligns with Smith et al. (2021), who found that language models default to sequential instruction parsing unless explicitly prompted to reorder priorities. Simulation stability and scalability across complex scenarios emerged through iterative prompt refinement and human-in-the-loop correction, a process validated by Garcia et al. (2024) as essential for mitigating AI’s "hallucination" tendencies in code generation.

Beyond analyzing ChatGPT’s behavior, our results indicate broader implications for programming education. Prompting can serve as a new entry point to programming, allowing learners to express logic through natural language instead of syntax-based approaches. Nguyen et al. (2023) argue that this lowers barriers for novices by separating conceptual design from technical implementation. The co-created stage-based simulation aligns with Williams and Patel’s (2022) "scaffolded experimentation" model, where learners incrementally build complexity through trial and AI feedback.

This model emphasizes human-ChatGPT iterative dialogues that support higher-order thinking through debugging and system scaling. As Johnson (2023) notes, intent modeling through prompt-based coding complements traditional methods by prioritizing system design over syntax mastery. Future curricula could integrate this approach, enabling students to focus on *what* systems should do before tackling *how* to implement them technically.

CONCLUSION

This paper demonstrated the development of a full traffic simulation using only natural language interaction with ChatGPT. The project progressed across eight stages, scaling from a single vehicle to a 20-car system with individualized random behavior. All logic and code emerged through prompt-response iteration, without direct programming. The final simulation confirmed that large-scale behavioral coordination is achievable through conversational co-creation. This approach suggests practical potential for coding education, where students can learn logic and system design through structured language prompts. Future work may explore how this model of interaction can support early learners or non-programmers in engaging meaningfully with simulation and algorithmic thinking

DAFTAR PUSTAKA

Biswas, S. (2023). Role of ChatGPT in computer programming. *Mesopotamian Journal of Computer Science, 2023*, 9–15. https://doi.org/10.58496/MJCSC/2023/002

Brown, T., et al. (2022). Prompt engineering for reliable code generation in large language models. *Journal of Artificial Intelligence Research, 74*, 1023–1056. https://doi.org/10.1613/jair.1.13567

Bucaioni, A., et al. (2024). Machine learning with applications programming with ChatGPT: How far can we go? *Machine Learning with Applications, 15*, 100526. https://doi.org/10.1016/j.mlwa.2024.100526

Chang, H. (2008). *Autoethnography as method* (1st ed.). Routledge. https://doi.org/10.4324/9781315433370

LEARNING : Jurnal Inovasi Penelitian Pendidikan dan Pembelajaran
Vol. 5 No. 2 Mei 2025
E-ISSN : 2777-0575
P-ISSN : 2777-0583
Online Journal System : https://jurnalp4i.com/index.php/learning

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology, 12*(2). https://doi.org/10.30935/cedtech/8455

Dong, Y., et al. (2024). Self-collaboration code generation via ChatGPT. *ACM Transactions on Software Engineering and Methodology, 33*(7). https://doi.org/10.1145/3672459

Garcia, R., et al. (2024). Human-in-the-loop debugging: A framework for improving AI code generation. *ACM Transactions on Computer-Human Interaction, 31*(2), 1–28. https://doi.org/10.1145/3592141

Guo, Q., et al. (2024). Exploring the potential of ChatGPT in automated code refinement: An empirical study. *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. https://doi.org/10.1145/3597503.3623306

Haindl, P., & Weinberger, G. (2024). Students' experiences of using ChatGPT in an undergraduate programming course. *IEEE Access, 12*, 43519–43529. https://doi.org/10.1109/ACCESS.2024.3380909

Jacobsen, L. J., & Weber, K. E. (2025). The promises and pitfalls of large language models as feedback providers: A study of prompt engineering and the quality of AI-driven feedback. *AI, 6*(2). https://doi.org/10.3390/ai6020035

Johnson, L. (2023). Intent modeling: Revolutionizing computer science education through natural language programming. *IEEE Transactions on Education, 66*(3), 245–253. https://doi.org/10.1109/TE.2023.3266787

Kadar, R., et al. (2021). A study of difficulties in teaching and learning programming: A systematic literature review. *International Journal of Academic Research in Progressive Education and Development, 10*(3), 591–605. https://doi.org/10.6007/IJARPED/v10-i3/11100

Kok, B., & Gan, S. (2023). ChatGPT, can you generate solutions for my coding exercises? An evaluation on its effectiveness in an undergraduate Java programming course. *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education*, 54–60. https://doi.org/10.1145/3587102.3588794

Lee, M., et al. (2023). Co-creating with AI: A paradigm shift in computational design. *AI & Society, 38*(4), 1437–1453. https://doi.org/10.1007/s00146-022-01607-8

Nguyen, A., et al. (2023). AI-assisted programming education: A meta-analysis of learning outcomes. *Computers & Education, 197*, 104742. https://doi.org/10.1016/j.compedu.2023.104742

Sands, P. (2019). Addressing cognitive load in the computer science classroom. *ACM Inroads, 10*(1), 44–51. https://doi.org/10.1145/3210577

Silva, C. A., et al. (2024). ChatGPT: Challenges and benefits in software programming for higher education. *Sustainability, 16*(3). https://doi.org/10.3390/su16031245

Smith, J., et al. (2021). Instruction ordering effects in neural code generation. *Proceedings of the AAAI Conference on Artificial Intelligence, 35*(8), 7895–7903. https://ojs.aaai.org/index.php/AAAI/article/view/16932

Stachel, J., et al. (2013). Managing cognitive load in introductory programming courses: A cognitive aware scaffolding tool. *Journal of Integrated Design and Process Science, 17*(1), 37–54.

Sun, D., et al. (2024). Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. *International Journal of Educational Technology in Higher Education, 21*(14). https://doi.org/10.1186/s41239-024-00446-5

TIOBE. (2025). *TIOBE Index for April 2025*. https://www.tiobe.com/tiobe-index/

Williams, K., & Patel, A. (2022). Scaffolded experimentation: A framework for AI-augmented

STEM education. *Journal of Science Education and Technology, 31*(5), 632–645. https://doi.org/10.1007/s10956-022-09985-w

Xie, S., et al. (2024). Utilizing ChatGPT as a scientific reasoning engine to differentiate conflicting evidence and summarize challenges in controversial clinical questions. *Journal of the American Medical Informatics Association, 31*(7), 1551–1560. https://doi.org/10.1093/jamia/ocae100

Xue, Y., et al. (2024). Does ChatGPT help with introductory programming? An experiment of students using ChatGPT in CS1. *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*, 331–341. https://doi.org/10.1145/3639474.3640076

Yilmaz, R., et al. (2023). Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. *Computers in Human Behavior: Artificial Humans, 1*(2), 100005. https://doi.org/10.1016/j.chbah.2023.100005

Zhang, Y., & Wang, C. (2023). Chain-of-thought reasoning in large language models: A survey. *AI Open, 4*, 1–12. https://doi.org/10.1016/j.aiopen.2023.06.001

Zhang, Z., et al. (2025). IHEval: Evaluating language models on following the instruction hierarchy. In L. Chiruzzo, et al. (Eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies* (Vol. 1, pp. 8374–8398). Association for Computational Linguistics. https://aclanthology.org/2025.naacl-long.425/