Testing Creative and Comprehensive Capabilities of Generative AI Through Game Design Evaluation

Chris Chang

Received August 25, 2024
Accepted December 20, 2024
Electronic access December 31, 2024

Despite generative artificial intelligence's(AI) proficiency in manual tasks and skill acquisition, concerns arise over its lack of creativity in content creation, which may hinder its application in fields such as game development. However, the ability of models, such as OpenAI's large language model ChatGPT-4, to emulate abstract qualities, such as moral judgment, suggests that generative AI could potentially understand abstract values and generate appropriate content for specified difficulty, audience, and genre. Thus, this paper investigated the creative and comprehensive capacity of generative AI and evaluated its potential as a game designer. To explore this, the puzzle game Bloxorz (Roll the Block) was used for its simple controls and high difficulty potential. By prompting GPT4 to generate levels of varying difficulty with different amounts of training data, the diversity of its designs and the accuracy of their difficulty were observed. The AI model demonstrated an increase in comprehension of difficulty with more extensive training data, but a lack of change in its creative incapabilities for game design. The results underscore the significance of larger datasets for training abstract values in AI, but also highlight that creative diversity for game design in AI may not simply be solved with larger data. Future work should explore effective training methods to enhance the model's creativity and address observed limitations in generations.

Introduction

Despite its rapidly increasing adeptness at manual labor and a number of skills, generative AI has demonstrated a lack of creativity and variation when it comes to generating new content. For instance, AI generated ideas in literature tended to be very similar and exhibited low collective diversity of novel content[1]. As boredom is structured through consumption of simple repetitive content, its inability to creatively generate content throws into question its capability as a quality content creator[2]. This barrier is what mainly prevents generative AI from stepping into fields such as game development. However, generative AI has displayed the ability to learn and emulate abstract qualities, including moral judgment. When trained with various scenarios rating the morality of a character, GPT4 demonstrated moral ratings of a 0.93 correlation to actual human ratings[3]. GPT4's demonstration of morality shows promise in generative AI's capability to understand abstract values and apply them. In the field of game development, this ability can allow generative AI to create contents of specific difficulty, emotions, or ethics, which can be highly useful when generating game designs of specific difficulty and audiences. As DeepMind did with AlphaZero, video games are often used in order to evaluate AI performance. DeepMind has stated that the game of chess represented the pinnacle of AI research over several decades, where state-of-the-art programs are based on powerful engines that search many millions of positions and leverage handcrafted expertise[4]. Similarly to how video game performance can provide insight into AI efficiency, the paper aimed to utilize game design to evaluate its creative and comprehensive potential. Thus, this paper investigated generative AI's creative capacity and comprehensive ability regarding abstract values, specifically difficulty, and evaluated its potential as a game designer. The chosen game was the famous puzzle game Bloxorz(A.K.A. Roll the Block) for its simplistic player controls and high difficulty ceiling. By prompting GPT4 with varying difficulty and different sizes of training data, the results demonstrated repetitiveness or variation in its designs, providing insight into its creativity and understanding of difficulty. Our contributions give intuition into the potential for creativity and general comprehensive ability in generative AI and whether these abilities can be further trained through large datasets. Information about such capabilities can illustrate generative AI's potentials to automate game development in the future.

Literature Review

Technical Background

Models such as GPT4 are foundation models that have capability to learn broadly applicable skills from large, diverse datasets, and subsequently adapt them to downstream tasks. Task specification can be broken down into a process that breaks

down human-provided task description into a quantitative metric that measures AI's completion and progress[5]. Specification can accept a variety of description modalities, such as goal states, natural language, pairwise or ranking comparisons, and feedback. Utilizing this, the prompt uses specification of the goal and game mechanisms and compares example levels to accurately deliver information. Furthermore, to process inputs, GPT utilizes a transformer model, which uses the self-attention mechanism to weigh the relative importance of words. This transformer model uses multiple layers of attention and feed-forward networks, which help in processing complex patterns in the text[6]. This allows GPT to interpret our lengthy and complex prompts. Reasoning and search is a critical theme for this study, as GPT needs to be able to reason a solution for its generations in order to ensure they are playable. In early stages, symbolic approaches were most widely used, but they quickly proved to be inefficient due to the required engineering effort to formalize heuristics[7]. Recently, data-driven methods using neural networks have shown proficiency by exploiting statistical structures and learning useful heuristics. For instance, in the game Go, which has long been viewed to be the most challenging of classic games for artificial intelligence due to its vast search space and the difficulty of assessing board positions and moves, AlphaGo has defeated the human European Go champion in games of Go by 5 to 0, using deep neural networks and tree search[8]. Similarly, Bloxorz solutions can be quickly evaluated by search algorithms, such as A* and Best First Search, that utilizes appropriate heuristics. Using such algorithms and neural networks, foundation models can model unlimited possible designs, and generate candidates suitable to the difficulty, making its generativity potentially more capable than human iterations.

Potential Regarding Game Development

The skills that generative AI currently possess already exhibit sufficient potential to automate manual aspects of game development. GPT3 displayed adeptness in programming, with an overall success rate of 71.875% across 128 code generation and debugging problems on Leetcode[9]. This result indicates potential to generate and debug mass code, which is largely significant for products such as video games that involve large amounts of scripts and code. Furthermore, only 38.7% of humans were unable to distinguish real photos from Midjourney-V5 generated ones, indicating its potential for automating art generation[10]. Automating art production allows for significant reduction in production time and cost, easing workload on developers. Most game industry professionals also exhibited an openness to the adoption and utilization of text-to-image generative (TTIG) AI; 14 Finnish game industry professionals expressed keenness to embrace a sustainable adoption of the technology in interviews[11]. However, the ethics of automating game development with AI remain questionable. Many artists fear AI art generators to replace their jobs, and concerns about copyright remain a legal and moral dilemma[11]. Yet these tools also significantly reduce development cost and speed, lowering barriers for independent indie game developers and smaller studios. Suitable regulations and further discussion are necessary for healthy integration of such tools. Nonetheless, generative AI is developing at an incredibly fast speed, foreshadowing its potential to replicate or even overtake human performance in the near future. For example, Google Scholar returns approximately 34,000 results for papers regarding generative AI predating 2020[12]. On the other hand, there are about 45,100 articles post-2024, indicating a double in the growth of the field over the span of 4 years. Its exponential growth rate implies that its current skills will only grow with time. Furthermore, generative AI has demonstrated potential in aspects beyond simple manual labor, displaying capability in emulating human-like judgements and sentiments when given appropriate training. For instance, OpenAI's ChatGPT-3.5 demonstrated judgments with correlation larger than 0.93 between human judgments when asked to evaluate the morality of persons in various scenarios, given 16 scenarios for training[3]. The result highlights generative AI's potential to comprehend abstract concepts and values, which can be used to generate level designs of suitable difficulty and automate quality level generation. In combination with its pre-existing programming and artistic capabilities, adeptness in design imply potential to vastly automate and easen game development. As of current, most machine learning and generative AI tools implemented in the game design industry aim to assist developers, specifically in artwork, rather than automating or taking the bulk of designing. For instance, Sketchar, a Generative AI tool that allows designers to prototype game characters and generate images based on conceptual input, provides visual outcomes that enhance communication and feedback with illustrators[13]. In PROJECT NOX, a game developed using AI tools, game designers used Midjourney to quickly generate character art and sped up the development processes significantly[14]. The study highlights "the emergence of a new era in which designers and artists who possess the skills to create effective and creative prompts, and who have access to super assistants such as Midjourney, Stable Diffusion, and Novel AI creative, can become a 'super game designer'." However, despite the numerous examples of utilizing generative AI to prototype character design, artwork, and storyboards, there is yet to be a study directly testing or utilizing AI's independent level design capacities.

Creative Framework

To evaluate generative AI's potential in game development and design, an evaluation of its creative ability and potential is necessary. For this, a creative framework is required. Existing

frameworks define "divergent thinking as the basis of creativity," and regard "the synthetic skill to see problems in new ways and to escape the bounds of conventional thinking" to be critical to creative skill[15]. In the case of the study, more divergent and distinguishable designs can be considered more creative. Inversely, a lack of design variance can be seen as uncreative.

Level Evaluation

Level design can be evaluated in various ways. The aspects of level design are typically split into two separate elements: usability and playability. Usability is defined as the degree to which the video game can be learned, used and is attractive to the player, while playability depends on the product's gameplay and interaction[16]. Usability can be measured through model-driven video game development, comparing elements, such as learnability, ease of use, technical accessibility, etc, through measurable variables, such as steps required for navigation. However, in the study's objective of assessing game design, only playability was evaluated. One way the difficulty of a level can be evaluated by using relative algorithm performance profiles. The performance difference, measured as score or win-rate, between generally better and worse game-playing algorithms is on average higher for well-designed games, as a game insensitive to skill is likely to be poor[17]. Playability can also be measured through heuristic sets, which contain standards for aspects including gameplay, control, and storytelling, that can be used for playtesters to give feedback on[18]. Specific standards from these sets can be utilized to create player evaluation forms to further assess difficulty and design on generated levels. User evaluation for game levels can also be completed through Player Experience of Need Satisfaction (PENS)[19]. Using a 7-point Likert scale, a PENS survey evaluates competence (difficulty), autonomy, relatedness (connection to others), presence, and intuitive control. Observation of the user's playthrough can also yield significant information about the player's honest opinion of the difficulty. Combining both to analyze player experiences and post-gameplay surveys can allow accurate perceptions of the difficulty.

1 Methods

Game Selection

The famous puzzle game Bloxorz was chosen for its simple mechanics but complex game design. Bloxorz is a puzzle game where players control a 1 by 1 by 2 block, flipping it onto its sides with the arrow keys (see Figure 1).

The goal of the game is for the player to fall into the 1 by 1 square hole at the end of each stage. If any part of the block extends over the edge, the block falls off the map and the level restarts, as seen in Figure 2. The complexity of the

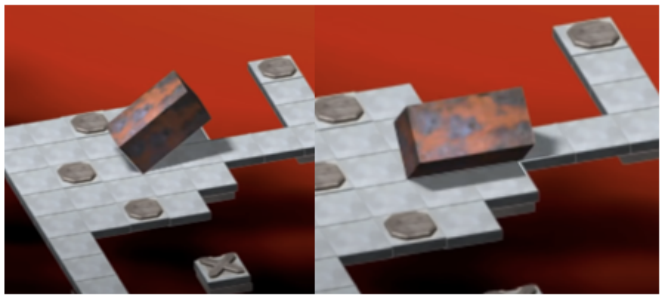


Fig. 1 Flipping the Block Onto Various Sides

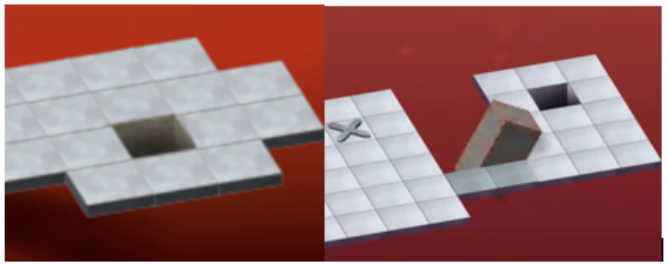


Fig. 2 End Goal(Left), Player Falling Off the Map(Right)

game comes from the fact that the goal requires the player to stand vertically on the hole, forcing the player to reposition themselves to be able to flip into the hole. For instance, in Figure 3, the player is forced to reposition themselves, as simply laying on top of the hole by flipping down would not complete the level. The game has additional features, such as buttons

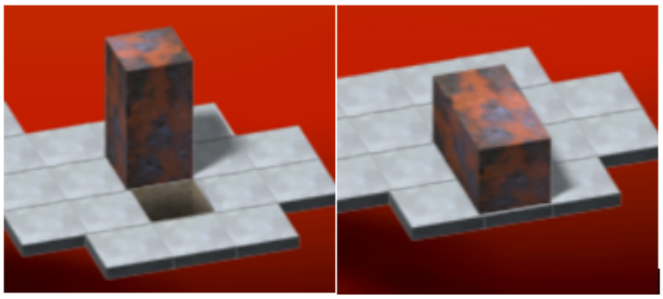


Fig. 3 Situation Where Repositioning is Required to Complete Level

that activate bridges when stood on, tiles that break when the player stands vertically on them, buttons that split the player into two, etc. These additional features, along with complex map designs, allow for incredibly difficult levels. It was determined that its high difficulty ceiling would allow GPT freedom in design, while its simplistic mechanics would be easy for it to understand. For the purpose of the study, GPT was only allowed to create levels without additional features in order to not make the prompt overly long and overload GPT with information. The Bloxorz solver used in the study also did not allow custom features to be used in level generation.

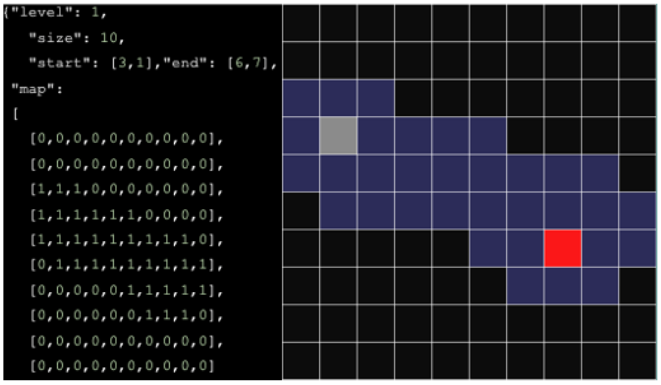


Fig. 4 (Left): JSON Representation of Bloxorz Level 1 (Right): Figure 4's JSON File Read in the Program

Prompting

Model Selection. OpenAI models, such as GPT-4 and GPT-4 Turbo, were selected for testing due to their state-of-the-art performance and public accessibility. Comparative evaluations like MMLU (multitask accuracy), GPQA (Graduate-Level Google-Proof Q&A), and MATH (mathematical problem solving) between other models had the following results:

- Claude 3.5 Sonnet: 88.3 (MMLU), 59.4 (GPQA), 71.1 (MATH)
- Gemini 1.5 Pro: 81.9 (MMLU), 58.5 (MATH) (GPQA scores not reported)
- GPT-4 Turbo: 86.7 (MMLU), 49.3 (GPQA), 73.4 (MATH)[20]

Although the three models exhibit similar performances, the advanced Claude and Gemini are closed behind paywalls, leading to the more publicly available ChatGPT being used for testing.

Formatting Level Designs. First, a JSON representation of a Bloxorz level was created so that GPT could read and generate new levels in the same format. The JSON file uses a square 2D list to represent the map grid, where 0's are empty and 1's have tiles. The file includes a level number, the level size(number of rows), start and end positions on the grid. This JSON file is fed into a python based top-down Bloxorz simulator, a tool developed by Fábio Oliveira, which then creates a level matching the map list[21]. For instance, Figure 5 is the simulation version of Figure 4, where the gray tile is the player position, blue tiles are the map, and the red tile is the end position.

This JSON format could potentially be scaled to include more complex mechanics by using other numbers to represent various game elements, such as using '2' to represent a button in game. This extension would help include larger datasets for testing in the future and also be used to represent games other than Bloxorz, allowing flexible game generation.

Introducing the Game. The movement mechanics were then introduced to GPT by breaking down the game mechanics into different states. By specifying the states that the block can be in and how the direction of movement changes states and the number of tiles required, GPT was able to more easily comprehend the mechanics. There are three states the block can be in: Standing Up, Lying Flat Horizontally, Lying Flat Vertically. Moving in a direction in a specific state results can change states, and the number of tiles required for the move to be valid depends on the current state. This was specified by detailing how movements in each state affect position and block states. For instance:

- **Block States**

1. Standing Up: The block occupies 1 tile.
2. Lying Flat Horizontally: The block occupies 2 tiles side-by-side horizontally.
3. Lying Flat Vertically: The block occupies 2 tiles side-by-side vertically.

- **Movement Details**

1. Standing Up (1 tile)

a. Move Up: The block will change its state to lying down vertically and take up the tiles a row above and row two steps above the starting position. Both tiles need to be 1's.

b. Move Down: The block will change its state to lying down vertically and take up the tiles a row below and row two steps below the starting position. Both tiles need to be 1's.

c. Move Left: The block will change its state to lying down horizontally and take up the tiles a column left and column two steps left of the starting position. Both tiles need to be 1's.

d. Move Right: The block will change its state to lying down horizontally and take up the tiles a column right and column two steps right of the starting position. Both tiles need to be 1's.

The other two states were also detailed in similar manner. Directly addressing previous mistakes in the prompt was also effective in preventing generation failures. For example, the line "After you move off of a tile, make sure that you keep the tile a 1. You often change the value of the tile to 0 after the block moves off of it. Do not alter the original map" was included after the error persisted. After providing it the necessary information about the movement mechanics, GPT's understanding of the mechanics was thoroughly tested by prompting it with a 5x5 grid and various movement directions:

now demonstrate that you correctly understand this by moving around in this 2d list:
[[0, 0, 0, 0, 0, 0],
[0, 1, 1, 1, 1, 0],
[0, 1, 1, 1, 1, 0],
[0, 1, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 0]]

After verifying that its understanding was correct, the goal of the game was introduced. Emphasis on the fact that the player must end in a Standing Up status on the end position was necessary in order to prevent misinterpretation of the goal: "...The game is not completed if the block is just Laying Flat Horizontally or Vertically with only a part of the block occupying the end tile."

Feeding Datasets and Prompting Level Generation. Afterward, various pairs of levels from the Bloxorz game were introduced. As GPT was limited to design levels with only tiles, only levels from Bloxorz that did not use additional features were selected: level 1, 3, and 6. The paper refers to these levels sourced from the actual game as 'True Levels,' in contrast to generated levels. GPT's comprehensive ability in relation to data size was tested through one-shot or few-shot cases, alternating the levels given. "Shot cases" refer to the number of training data used, where a one-shot case equates to one training level inputted. The test cases were as follows:

1. One-shot: a. Level 1 provided, b. Level 3 provided, c. Level 6 provided,

2. Two-shot: a. Levels 1 & 3 provided, b. Levels 1 & 6 provided, c. Levels 3 & 6 provided,

3. Three-shot: All three levels provided

Having provided example levels, GPT was prompted to generate levels of varying difficulty. Prompts specified either to generate a new design for a level given, or to generate a level in between or higher than given levels. For example, in a two-shot case, level 3 and 6 were provided with a prompt to generate a level 4, and, in a three-shot case, GPT was prompted to generate a level more difficult than all given levels, level 9. The number of moves required for each level were also provided in order to give GPT a better sense of the scaling difficulty: "this is level 6 of the game for reference of difficulty and completability. It can be completed in 29 moves. . . Noting the scaling difficulty, create a new level design of LV6 that would have the same difficulty." There were no restraints other than level difficulty, allowing GPT to randomize start and end positions, as well as obstacles and paths.

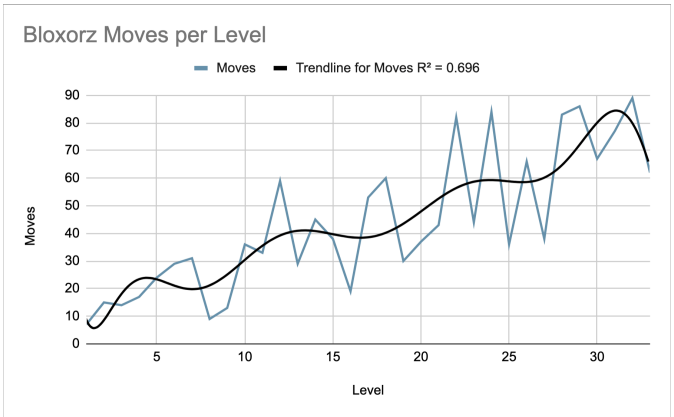


Fig. 5 Required Number of Moves vs. Bloxorz Level Number.

Level Evaluation

The generated levels were first tested by search algorithms to verify its playability. Once playability was ensured, various methods were used to evaluate its difficulty.

Number of required moves. Bloxorz levels have a clear increasing trend in the number of moves required as the levels increase in difficulty, making it a good estimator of difficulty. Other metrics, such as map size, distance between start and end positions, number of available paths, etc, did not have a clear positive relationship with difficulty, making them difficult to use.

Using a 10th degree polynomial regression of the number of moves required for each true Bloxorz level, their difficulties were evaluated by calculating their residuals from the regression model (see Figure 5). Compared with other regression models(linear, logarithmic, exponential, lower degree polynomials), polynomial regression had the highest R^2 value of 0.696, leading to its usage to estimate expected move count per level. Taking random error into consideration, the generations were considered to have achieved expected difficulty when the absolute values of their residuals were less than the Mean Absolute Error (MAE)The slope for the linear regression is 2.07, and the MAE from levels 1 through 10 is 6.83. As target difficulties of the generations only ranged from 1 to 10, MAE was only calculated from levels 1 through 10 to reduce skewness. The reason for using MAE instead of standard deviation or MSE was due to MSE and standard deviation's weakness to outliers. The dataset had a fairly high number of outliers, as there are some levels that introduce new elements to the player, taking a dip in difficulty and move count, despite its high level number. Various pathfinding algorithms, such as Depth-first Search, Breadth-First Search, Uniform cost Search, Best first Search, and A*, were used to find the smallest number of moves required for each level, as these are the most commonly used graph search algorithms for solving puzzle games and were also built into the

Bloxorz solver utilized in this research.

Algorithm Performance Profiles. More complex levels were further tested by utilizing various search algorithms to calculate the average pathfinding time for each level. By comparing the pathfinding time for more complex levels, it is reasonable to hypothesize that levels with longer solving times are generally more complex than others. Depth-First Search was excluded due to the skewness in average solve time that the brute force nature of DFS caused. The previously mentioned relative algorithm performance profiling can also be used to evaluate level design, but it is notable that they can also produce results that do not accurately reflect level designs, as these search algorithms were not optimized for Bloxorz and can have inconsistent performances.

Results

GPT4o VS GPT4

GPT4o, a newer model of GPT4, displayed a relative incapability in comprehending the game mechanics. In all cases, where GPT4o was given one-shot, two-shot, or three-shot prompts, they all produced unplayable levels. When prompted to provide a solution, GPT4o generated solutions only possible for a 1x1x1 cube, implying its misinterpretation of the game mechanics, despite continuous iterations of the dimensions. In other cases, 4o displayed correct understanding of the control mechanics, but failed to remember that the block cannot move off the map, generating impossible solutions and levels. On the other hand, GPT4 was able to comprehend the same prompts and generate completable levels. Thus, GPT4 was utilized for the test cases. Both models were capable of correcting their errors when pointed out.

One-shot cases for GPT4

Neither GPT4 nor GPT4o were capable of generating levels on a zero-shot case, where they were provided no example levels. In a one-shot case, GPT4 tended to generate levels very similar to the designs of the provided level, regardless of how difficult their target level was. For instance, when prompted to generate level 4, 6, and 10, given level 1, they produced near identical designs, despite being prompted with higher difficulty (See Figure 6). The only key difference was an increase in the map sizes as the prompted difficulties increased.

This pattern followed for other one-shot cases. When provided with level three, generated levels tended to model off its relatively round design (See Figure 7).

The solution costs for each generated level 4 was 4, 10, and 16, respectively, indicating a slight increase in complexity, but not complex enough to hit the expected number of moves. For level 6, the solution costs were 6, and 14. The average residuals

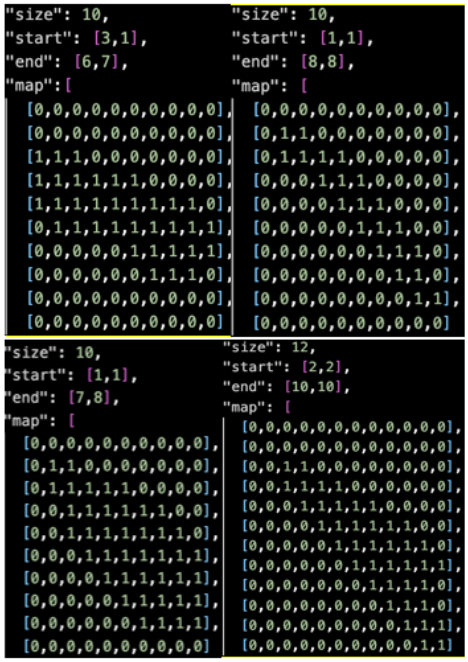


Fig. 6 True Level 1(Top-Left) in Comparison to Generated Level 4(Top-Right), Generated Level 6(Bottom-Right), and Generated Level 10(Bottom-Left) [22]

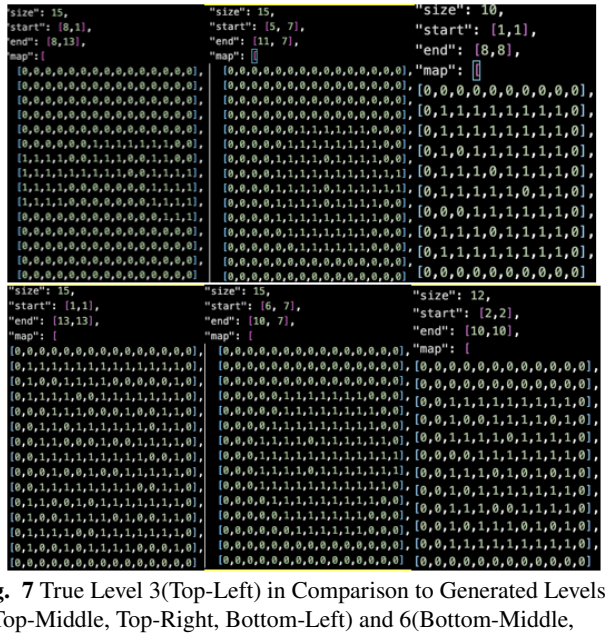


Fig. 7 True Level 3(Top-Left) in Comparison to Generated Levels 4(Top-Middle, Top-Right, Bottom-Left) and 6(Bottom-Middle, Bottom-Right)

from the number of required moves regression model for each level was -13.36, and -9.08 moves, both much greater than the MAE, and the standard deviations were 4.89 and 4. One-shot cases given level 6 had significant difficulty creating playable levels. When prompted to create a solution, it would generate unplayable solutions that caused players to move off the map. When pointed out, GPT attempted to correct them generating very simplistic designs, similar to Figure 6.

Two-shot cases

There were three different two-shot cases where the true levels were paired up differently and provided.

Provided Levels 3 & 6. When provided with levels 3 & 6, and prompted to design level 4, GPT generated very simple designs. The designs were much simpler than level 3, despite displaying awareness of the difficulty scale: "Given the examples of level 3 and level 6, it can be inferred that the difficulty for Level 4 should increase in complexity compared to Level 3 and be somewhat less complex than Level 6."[1] Start position was [1, 1], and the end position was [9, 8]. on a size 10 grid. The shortest solution for each generation was 11, 14, and 13 moves, with an average residual of -10.75 and standard deviation of 1.24. (See Figure 8)

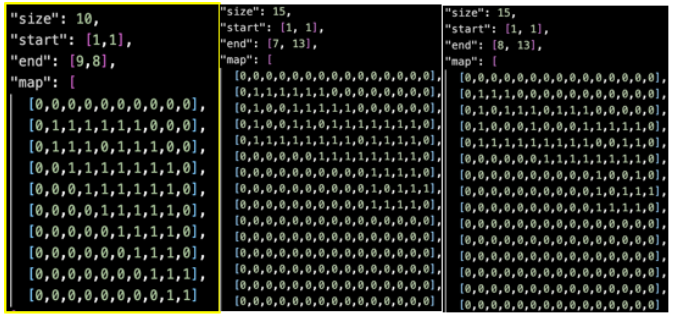


Fig. 8 Level 4 Generations Given Level 3 & 6

On the other hand, when prompted to design a level with the same difficulty as 6, it generated a level that took on a similar shape as Figure 8, but was relatively more complex, with a much larger grid. Start position was [1, 1], and the end position was [10, 10]. The shortest solution for each generation was 13, 15, and 15 moves, with an average residual of -6.75 moves under the expected move count and a standard deviation of 1.15. (See Figure 9)

Provided Levels 1 & 6. When provided with levels 1 & 6 and prompted to design level 4, GPT designed a level very similar to the previous cases of level 4 generations (see Figure 8, Figure 10). Start position was [1, 1], and the end position was [8, 8]. The shortest solution for each generation had a move count of 10, 9, and 10, which has an average residual of -13.75 moves and a standard deviation of 0.57.

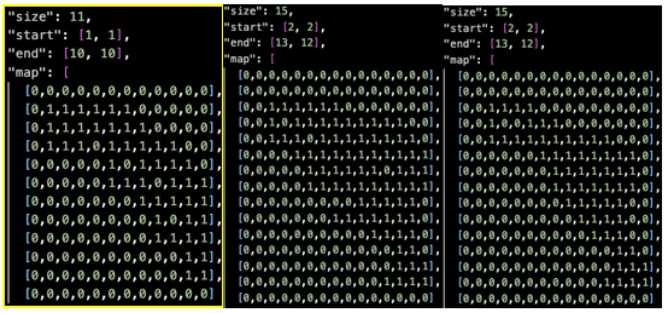


Fig. 9 Level 6 Generation Given Level 3 6

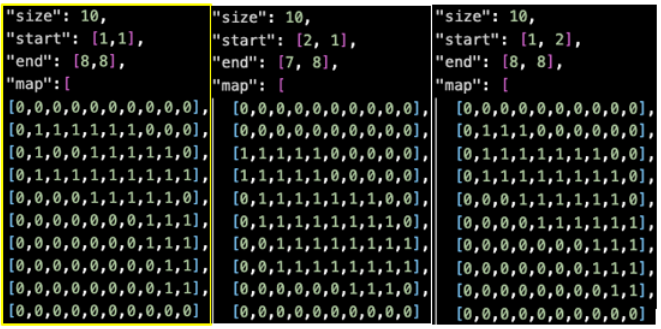


Fig. 10 Level 4 Generations Given Levels 1 & 6

When prompted to generate level 6, it also generated a level with a shape very similar to Figure 10. The key difference was its increase in size and slight increase in complexity through 0's scattered about the map. Start position was [1, 1], and the end position was [8, 9]. The required number of moves was 13, 12, and 15 for Figure 11, with an average residual of -8.58 moves and a residual standard deviation of 1.52.

Provided Levels 1 & 3. When provided with levels 1 and 3, and prompted to generate level 6, it generated a level that was of significant difficulty, with 0's planted in places to restrict player movement and false paths to confuse them. The generated levels had required move counts of 11, 12, and 16, respectively, with an average residual of -8.08 moves and a standard deviation of 2.64 (see Figure 12).

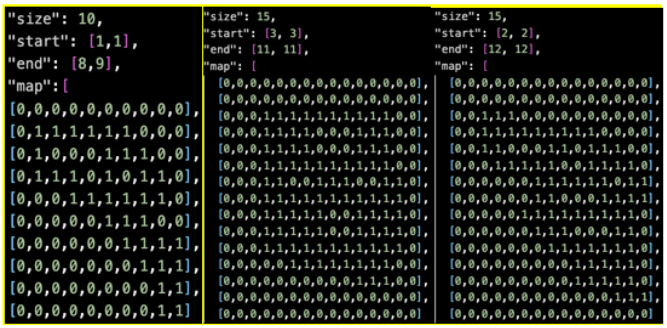


Fig. 11 Level 6 Generations Given Levels 1 & 6

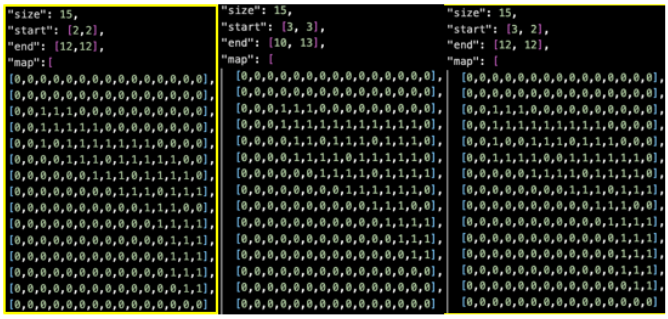


Fig. 12 Generated Level 6's Given Levels 1 & 3

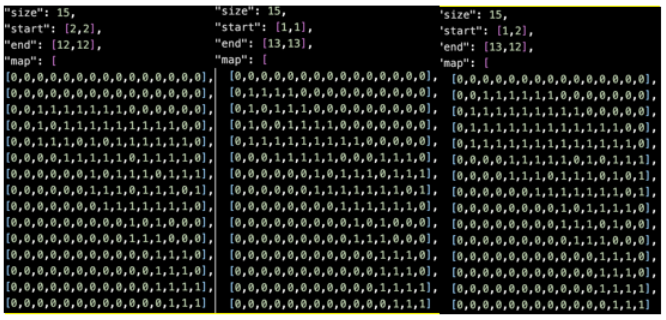


Fig. 14 Level 9 Generations Given All True levels

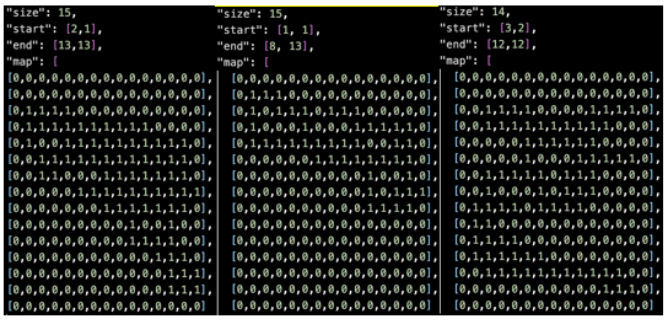


Fig. 13 Level 6 Generations Given All True levels

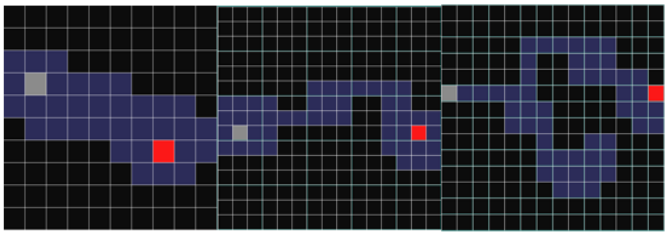


Fig. 15 Visualized Maps of True Levels 1, 3 & 6 For Comparison

Three-shot cases

When given all three levels and prompted to create a new level 6 of the same difficulty, GPT generated much more complex level designs than previous. The start position was [2, 1], and the end position was [13, 13], on a size 15 grid. The solution cost for each generation was 17, 34, and 17, with an average residual of 1.58, first case with a positive residual, and a standard deviation of 8.01 (see Figure 13).

When prompted to generate level 9, it generated the most complex level by far. GPT added more complexity by scattering more 0's across the map, attempting to restrict the player's movement further (see Figure 14). It claimed that "the map will have narrow paths, dead ends, and strategic use of space that forces the player to carefully manipulate the block's orientation and position" in order to fulfill the "significant increase" in complexity level 9 should have in comparison to previous levels. The start position was [2, 2], and the end position was [12, 12], on a size 15 grid. The number of required moves for each generation was 22, 32, and 16 which has an average residual of -7.06 moves and standard deviation of 5.72. (See Figure 14) It is also noted that, despite the relatively low move count solution for generation 3, the level was much more complex than lower shot-case generations, utilizing tight and multiple pathways to confuse the player.

It should be noted that algorithm performance may not be reflective of actual difficulty, as some map designs tend to be more susceptible to longer search times due to the large number of possible moves, while its perceived difficulty may not be as high (See Figure 16).

While the results of the two-shot cases were more steady with a low standard deviation, other cases were more erratic and had wide ranges of results. (See Table 2) Due to the large standard deviation in one-shot cases, the difference between one-shot and two-shot cases may not be as statistically significant, so comparisons between the two should be taken with caution. On the other hand, the residual averages of three-shot cases did not fall into the confidence intervals of the two-shot cases, and vice-versa, indicating that the difference between the two-shot and three-shot cases are significant enough to take note of.

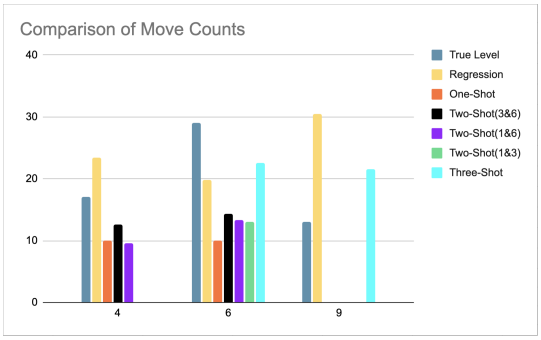


Fig. 16 Comparison of Move Count Between Categories

Table 1: Algorithm Performance Profiles

Level Solved / Algorithms	BFS	Uniform Cost Search	Best First Search	A*	Avg.
Figure 8: Level 4 / Two-shot(3&6)	1.666	2.794	0.42	1.05	1.4825
Figure 9: Level 6 / Two-shot(3&6)	1.293	1.977	0.687	0.971	1.232
Figure 10: Level 4 / Two-shot(1&6)	1.561	2.937	1.091	1.009	1.6495
Figure 11: Level 6 / Two-shot(1&6)	0.278	0.415	0.317	0.317	0.3317
Figure 12: Level 6 / Two-shot(1&3)	1.198	2.233	0.803	1.75	1.496
Figure 13: Level 6 / Three-shot	3.969	8.842	1.552	1.972	4.0837
Figure 14: Level 9 / Three-shot	2.664	4.689	2.077	1.963	2.8482

Residuals Summary

Table 2: Table of Residuals and Solution Costs For Each Generation

Level / Shot Case	Residual	Average Solution Cost	Standard Deviation
Figure 7: Level 4 / One-shot	-13.36	10	4.89
Figure 7: Level 6 / One-shot	-9.08	10	4
Figure 8: Level 4 / Two-shot(3&6)	-10.75	12.6	1.24
Figure 9: Level 6 / Two-shot(3&6)	-6.75	14.3	1.15
Figure 10: Level 4 / Two-shot(1&6)	-13.75	9.6	0.57
Figure 11: Level 6 / Two-shot(1&6)	-8.58	13.3	1.52
Figure 12: Level 6 / Two-shot(1&3)	-8.08	13	2.64
Figure 13: Level 6 / Three-shot	1.58	22.6	8.01
Figure 14: Level 9 / Three-shot	-7.06	21.5	5.72

Discussion

Although GPT4 at its base model did not have an accurate comprehension of difficulty, a relatively accurate sense of difficulty can be trained through larger data. Though not linear, the comprehension level relatively increased with larger inputs from the dataset, shining light onto generative AI's potential to comprehend abstract values with large data. However, GPT4 demonstrated creative limitations in its designs across all shot cases, generating levels of very similar design and varying the details, rather than creating new designs. These results highlight that larger datasets do not necessarily increase the creativity of generative AI and that a difficult method of training needs to be investigated for improvement in this aspect.

In general, as shot counts increased, there were decreases in residuals and increases in solve times, indicating an increase in difficulty accuracy in generations. For instance, residuals for two-shot cases were approximately half of one-shot cases, and three-shot cases saw a further decrease in residual average (See Table 2). Average solve times going from one to two-shot cases increased from 1.4825ms to 1.6495ms, and there was also a clear jump in solve time average going from two-shot to three-shot cases (See Table 1). Higher shot generations utilized more obstacles and movement limitations as their primary methods of increasing difficulty. Three-shot levels especially did this well, producing the lowest residual average, which was well under the MAE, and thus hitting the expected difficulty range. However, most levels took upon very similar shapes, with a general trend of starting in the top-left corner and progressing diagonally. For instance, Figures 11 and 12, which were given the different shot sizes, closely resemble each other's diagonal progression and shape. Despite an increase in difficulty, this repetition can result in repetitive gameplay. The similarity across many generations, regardless of shot size, indicates that creative limitations are not trainable through larger data, unlike comprehension.

GPT-4o vs. GPT-4

In finding a reason for GPT-4o's inferior performance, both models are close-sourced and are not open to the public, so direct structural comparison is difficult in this case. Despite OpenAI's claims that GPT-4o "matches GPT-4 Turbo performance on text in English and code…while also being much faster and 50% cheaper in the API," previously conducted evaluations of the two models back up the study results[23]. On evaluations on MMLU(multitasking accuracy), GPQA (Graduate-Level Google-Proof Q&A), MATH(mathematical problem solving), and HumanEval(coding accuracy), GPT-4o had a scores of 85.7(MMLU), 46.0(GPQA), 68.5(MATH), and 90.2(HumanEval) as of November 20th, while GPT-4 Turbo performed scores of 86.7(MMLU), 49.3(GPQA), 73.4(MATH), and 88.2(HumanEval)[20]. These findings indicate that while GPT-4o is superior in programming skills, its problem solving and general thinking skills fall behind GPT-4, giving reasoning for GPT-4o's relative failures in testing.

Zero-Shot Case

GPT-4 failed to produce successful level designs on zero-shot cases. This result is most likely due to the fact that ChatGPT models have generalized capabilities for public use, and thus would most likely not have innate game design capabilities. This is further reaffirmed in testing cases, where GPT-4 produced incorrect information about Bloxorz move mechanics upon questioning, indicating that GPT models most likely lack sufficient understanding of the game to have successful level designs without training.

One-Shot Case

GPT displayed a very basic understanding of difficulty and complexity in one-shot cases by increasing map sizes in attempts to make levels more difficult (See Figures 6 and 7). As a result, the number of required moves did increase, but not significantly due to a lack of change in design complexity. This suggests that, without training, GPT assumes that simply increasing the map size, instead of increasing design complexity, would suffice in increasing difficulty, demonstrating inaccurate understanding at lower shot cases. However, the difficulty of the levels were much lower than the expected level number. The residuals for Figure 7 were over double the MAE, indicating that it was considerably under the expected difficulty. These results infer that when given little to no data, GPT cannot accurately determine the difficulty of a level, concluding that comprehension of difficulty has to be trained and not reasoned off base model. Furthermore, GPT was unable to vary its designs and rather heavily modeled its levels off the data. GPT's generations closely resembled the shapes of the true levels that were given to them, with only simple variations in size, length and height of the maps (See Figures 6 and 7). It can be inferred that GPT heavily relies on its training data and is relatively incapable of creative generations without large data sizes.

Two-Shot Cases

Designs in two-shot cases displayed a closer adherence to the expected difficulty and an increase in complexity, indicating that comprehension of difficulty increases with more samples. However, as average residuals were still larger than the MAE, the levels were still easier than expected, implying that training data needs to be of significant size to achieve accurate comprehension. The pair of levels provided did not significantly influence difficulty comprehension, as residuals were similar between the numerous two-shot cases. Each level 4 and level 6 generation had a residual less than 1 from their respective averages, indicating similar difficulty levels for both generations (See Table 2). The lack of change in difficulty highlights how the dataset itself does not significantly alter its level of difficulty comprehension, as long as the data is consistent.

Three-shot Cases

GPT performed the best when provided all three levels, generating levels with significant complexity and difficulty matching the expectations. The generations had the largest average solve time of 4.0837ms. It was the first case where the generations had a positive residual, indicating it generated levels beyond expected difficulty. GPT also displayed comprehension of difficulty scaling beyond given data, accurately producing a level with the highest move count of its generations when prompted with level 9. It included many more 0's carefully placed across the map to create false paths to confuse the player and extremely limit player movement (See Figures 13 and 14). Although because of this, the number of total possible moves decreased, and, as a result, the average solve time was lower, from a complexity standpoint GPT was successful in its generation.

In summary, providing larger amounts of sample levels allowed GPT to generate levels of which difficulty were more adherent to the difficulty scale. Thus, it can be inferred that GPT does not have an innate sense of difficulty, but it can be trained and emulated to an extent using large datasets.

However, unlike comprehension, larger datasets were still not able to creatively train GPT to be able to generate new designs. Generations still took on similar shapes and trends in progression despite increases in shot cases (See Figures 13 and 14). The lack of improvement in creativity across all three cases imply that larger datasets do not necessarily guarantee diverse content generation, and that separate training may be necessary for this element.

Prompt Development

Various prompting methods were experimented throughout the study. The first method tested was textual visualization of the movements, with textual graphs indicating current block positions and how movement direction changes them:

> X marks the tiles that the block is taking up. 1s are the tiles the block can move onto.
> [X, 1, 1]
> [1, 1, 1]
> Block starts vertically on tile [0, 0]
> [1, X, X]
> [1, 1, 1]
> The block flips left onto its longer side, thus taking up two tiles.
> [1, 1, 1]
> [1, X, X]
> The block flips downward, onto a different long side, thus it needs two tiles below it to flip down.

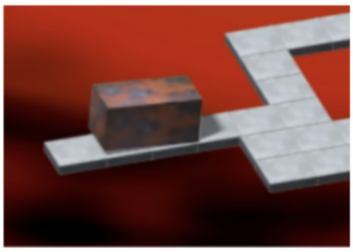
This method was met with consistent misunderstandings of block movements, particularly producing errors such as thinking the block is a cube (despite continuous iterations of its dimensions), or thinking the block does not move off the initial position and instead flips onto the same position. This was attempted to be amended by utilizing actual visuals rather than text. By using images and descriptions of the movement, the prompt aimed to amend its misunderstandings of the movement through visual learning (See Figure 15).

"Lying Flat (Horizontal or Vertical)

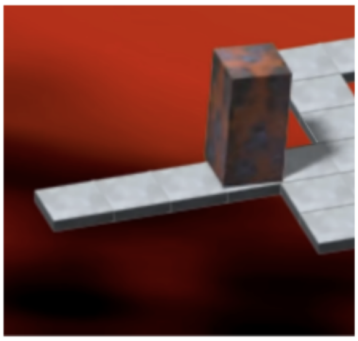
- Occupies 2 tiles.

For instance, look at the images.

The first image is the spawn state of the block.



Then it moves right to flip onto its side, as in image 2.



Then it moves right again to stand back up, as in image three.

Fig. 17 Prompt Excerpt Utilizing Visuals

- Movement:
  Up/Down: The block will flip along its length, occupying 2 tiles in a vertical line.
  Requires 2 tiles in the direction of movement.
  Left/Right: The block will flip along its width, occupying 2 tiles in a horizontal line.
  Requires 2 tiles in the direction of movement."

Although most of its description was inconsistent and inaccurate, taking this structure as a basis, the movement mechanics were detailed through different block states and how movement changes the states to one another. Its understanding was further reinforced through a testing of its comprehension on a random grid, prompted with random directions of movement. This method produced consistently accurate understandings of the movement. The continuous need for adjustment to the prompting indicates that GPT-4 is sensitive to prompts and weak or misleading prompts will produce undesirable results. Although minor adjustments, such as phrasing or grammar, are insignificant, the structure of the prompt and methods of introducing complex subjects sway the performances heavily. It is important to explain games rules in a clear and logistic manner, without usages of vague and metaphoric wording, when attempting to reproduce results.

Limitations

GPT had difficulty retaining memory of the game mechanism, often leading to generations of unplayable levels. Despite displaying correct understanding of the movement when tested, once asked to generate a level, it would forget basic rules of the movement half-way through generating the solutions. The most common mistake was that it would forget that no part of the block can be on a 0, creating levels with solutions that go off the map. This forced many of the generations to be discarded and to be reprompted. The errors occurred for the one-shot case of level 6 is hypothesized to have occurred due to the level provided being too complex for GPT to base its generations off of, creating levels that prioritized complexity over playability. When prompted with feedback, GPT recognized this and generated simplistic designs similar to Figure 6, indicating that it opted for simplicity once it saw the errors of their over-complex designs. Furthermore, solve times were often not well reflective of the difficulties of the levels. The search algorithms used are often susceptible to certain designs, regardless of their true difficulty. For instance, the maps in Figure 7 have almost no constraints on player movement, being a large connected map. This allows for a large number of possible movements, causing BFS to take much longer than others, despite the low difficulty, and skewing the results (See Figure 16).

Other algorithms also appear to have weaknesses that skew search times. A*, Best First Search, and Uniform Cost Search

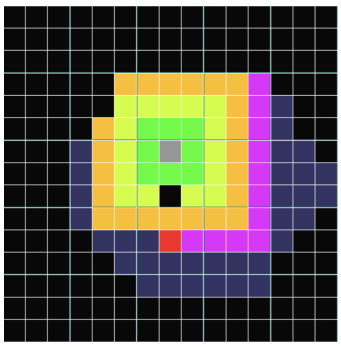


Fig. 18 Visualization of BFS Search in Figure 7; Sweep Progresses From the Player(Gray) to Green, Yellow, Orange, Purple and Stops Once It Reaches End Goal(Red)

seem to be particularly weak at repositioning themselves to vertically land on the end goal. This is because repositioning often requires moving away from the goal in order to put the block in the state needed to land on the end position, which is contradictory to the cost heuristics they tend to follow. For instance, in Figure 10, the level requires positioning the block before approaching the goal in order to be able to complete it, which causes search times to be longer than expected for such algorithms. This makes it appear as if Figure 10 is more difficult than Figure 11, despite Figure 11's larger complexity (See Figure 11). For this reason algorithm performances were used cautiously, and statistically significant times, such as the large spike in average time for three-shot cases, were utilized. The specific requirements for the levels selected as training data also enforce a limitation on dataset size, potentially influencing the model to bias its generations off the small amount of training data.

Additionally, generative models still face the hardships of balancing difficulty and enjoyability and engagement, an issue even human developers continue to face. While a relatively accurate understanding of difficulty, and even enjoyability, could be trained through large datasets, their potential to understand this balance and generate both difficult and engaging levels remain difficult to foresee.

Conclusion

In conclusion, this study provides evidence of generative AI's potential in the field of game development, particularly in its comprehensive abilities, but highlights its lack of creativity. Through the evaluation of GPT-4's performance in level generation for the game Bloxorz, it was observed that the AI model demonstrates an increasing comprehension of difficulty with more extensive training data, but there was no observed increase in creative diversity in its generations. The study underscores the importance of larger datasets in training AI for abstract values, especially in smaller data regimes, but also highlights that fostering creative diversity in AI-generated game designs may require other methods of training. As seen with AI's creative limits in literature, this creative barrier may hold across other creative applications as well, but testing in other fields would be required to confirm this suspicion [1].These findings suggest that while generative AI shows promise in automating aspects of game development, further advancements are necessary to overcome its current creative constraints and leverage its full potential in the field. Future work should explore effective training methods to enhance the model's creativity and address limitations observed in generations.

Acknowledgments

The author expresses gratitude towards Anna Bair for her assistance in formulating the study.

References

1 A. R. Doshi and O. P. Hauser, *Science Advances*, 2024, **10**, year.

2 P. J. Geiwitz, *Journal of Personality and Social Psychology*, 1966, **3**, 592–600.

3 D. Dillon, N. Tandon, Y. Gu and K. Gray, *Trends in Cognitive Sciences*, 2023, **27**, 597–600.

4 D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan and D. Hassabis, *arXiv.org*, 2017.

5 R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, C. for Research on Foundation Models (CRFM), S. I. for Human-Centered Artificial Intelligence (HAI), S. University and P. Liang, *On the opportunities and risks of foundation models*, 2022, https://arxiv.org/pdf/2108.07258.pdf.

6 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser and I. Polosukhin, Advances in Neural Information Processing Systems, 2017.

7 S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson, 4th edn, 2020.

8 D. Silver, A. Huang *et al.*, *Nature*, 2016, **529**, 484–489.

9 A. F. Sakib, H. S. Khan and R. A. H. M. Karim, *Extending the frontier of ChatGPT: code generation and debugging*, 2023, https://arxiv.org/pdf/2307.08260.

10 Z. Lu, D. Huang, L. Bai, J. Qu, C. Wu, X. Liu and W. Ouyang, Advances in Neural Information Processing Systems, 2023.

11 V. Vimpari, A. Kultima, P. Hämäläinen and C. Guckelsberger, *Proceedings of the ACM on Human-Computer Interaction*, 2023, **7 (CHI PLAY)**, 131–164.

12 *Google Scholar*, Retrieved August 24, 2024, from https://scholar.google.com.

13 L. Ling *et al.*, *Proceedings of the ACM on Human-Computer Interaction*, 2024, **8 (CHI PLAY)**, 337.

14 J. Lee and J. S. Gero, *Proceedings of the 24th International Society for Design and Development in Education (ISDDE) Conference*, 2023.

15 R. J. Sternberg, *Creativity Research Journal*, 2006, **18**, 87–98.

16 A. Fernandez, E. Insfran, S. Abrahão, J. Carsí and E. Montero, Lecture Notes in Computer Science, 2012, pp. 307–314.

17 T. S. Nielsen, G. A. B. Barros, J. Togelius and M. J. Nelson, Lecture Notes in Computer Science, 2015, pp. 369–380.

18 F. Manzoni *et al.*, ICEIS, 2020, pp. 499–510.

19 S. Kirginas, *Software Engineering for Games in Serious Contexts*, Springer, Cham, 2023, pp. 19–42.

20 OpenAI, *GitHub - Openai/Simple-evals*, 2024, `https://github.com/openai/simple-evals?tab=readme-ov-file`.

21 Erroler, *GitHub - Erroler/Bloxorz: Bloxorz Is a Challenging Puzzle Game in Which You Must Move a Two-story Block Into the Objective Square Hole by Rolling It Around*, 2020, `https://github.com/Erroler/Bloxorz/tree/master`, Course Assignment.

22 OpenAI, *ChatGPT (August 8 version)*, 2024, `https://chat.openai.com/chat`, Large language model.

23 *Hello GPT-4o*, 2024, `https://openai.com/index/hello-gpt-4o`.