End-to-End Procedural Level Generation in Educational Games with Natural Language Instruction

Vikram Kumaran, Dan Carpenter, Jonathan Rowe, Bradford Mott and James Lester
North Carolina State University
{vkumara, dcarpen2, jprowe, bwmott, lester}@ncsu.edu

2023 IEEE Conference on Games (CoG) | 979-8-3503-2277-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/CoG57401.2023.10333195

***Abstract*—As the role of procedural content generation in mixed-initiative game design continues to grow, it is crucial to develop an end-to-end approach that enables non-technical designers to artfully guide content generation. Recent advances in large language models, such as GPT-4, are rapidly transforming the landscape of automated generation of text-based content. Large language models have significant potential for mixed-initiative procedural level generation by providing natural language interfaces for designers. This paper presents an end-to-end procedural level generation framework that interprets natural language descriptions of level design constraints and optimization objectives to facilitate the collaborative creation of game levels for a strategy game focused on environmental sustainability education. The framework enables designers to specify a problem domain, goal metrics, and target difficulty via natural language description. It then employs large language models for the semantic extraction of constraints and optimization targets to drive the generation of candidate levels. Generated game levels are evaluated via game-playing agents trained with deep reinforcement learning techniques to ensure the game levels meet the level designer's specifications. Manual evaluation by authors shows that the proposed framework can effectively transform designers' natural language descriptions into fully playable game levels that reflect their intended design objectives.**

***Index Terms*—Educational games, co-creativity, large language models, programmatic game-level generation.**

I. Introduction

Mixed-initiative procedural content generation (PCG) has long been an area of interest in game research because it combines human creativity with computational generation to create novel game experiences [1]–[3]. In addition to being useful in games for entertainment, PCG has also been explored for educational games [4]. As the scope of PCG expands to encompass a broader range of users like educators, students, and subject matter experts, there is a demand for interfaces that are appropriate for designers who may have limited technical expertise. A natural language interface would enable designers to intuitively provide subject matter guidance and instruct PCG about player experience. In the context of educational games, this can lead to the creation of engaging virtual worlds that can enhance students' critical thinking and other essential skills necessary for success in today's world [5], [6]. However, most of the content creators in this space are not experienced game designers. A crucial insight from educational game design is that input from subject matter experts is essential in developing a game with the necessary pedagogical impact [7]. An end-to-end framework that balances a programmatic content generator and a user-friendly natural language interface could facilitate seamless integration of input from non-technical level designers.

Large Language Models (LLM) pre-trained on vast amounts of human-generated text can quickly adapt to new tasks with limited examples, a process known as few-shot or zero-shot learning. LLMs such as GPT-4 can perform prediction tasks by filling unfilled slots in a prompt string using their internal word prediction probabilities. No domain-specific training is required, and they can adapt to new tasks with the appropriate instructions implanted in the prompting text [8]. These models have also successfully synthesized code with only simple instructions [9]. We leverage the power of these models to extract level designer choices expressed in natural language into structured constraints and optimization goals for our PCG game-level generator.

Our natural language based PCG framework is evaluated on a prototype game-based learning environment, FUTURE WORLDS, for children aged 9-12 to learn about environmental sustainability. The game engages players in enhancing a virtual biosphere, where success hinges on optimizing environmental sustainability metrics and learning through gameplay. Our end-to-end PCG framework enables level designers to generate game levels by inputting natural language instructions on the sustainability problem, goal metrics, and difficulty, which are then extracted using LLM-based semantic parsing as a zero-shot multi-class classification engine [9]. We use LLM's code generation capability to output classification labels as JSON objects. This generated data structure is the input to the game-level generator for generating candidate game levels. We use Deep RL (DRL) to solve game levels and determine their difficulty based on the RL engine's minimum required steps. The DRL selects the level that best matches designer instruction on difficulty. We estimate the variability and alignment of solutions by comparing player-chosen sustainability goals with the generated levels.

Our novel framework allows for the intuitive and customizable creation of game levels with designers controlling the problem domain, goal metrics, and difficulty levels to

979-8-3503-2277-4/23/$31.00 ©2023 IEEE

Authorized licensed use limited to: N.C. State University Libraries - Acquisitions & Discovery S. Downloaded on March 25,2024 at 14:35:49 UTC from IEEE Xplore. Restrictions apply.

tailor game levels to individual player needs, through natural language instructions.

II. Related Work

Mixed initiative co-creation of games addresses a broad range of creative collaboration with varying degrees of contribution from humans and program-based content generators [1]–[3]. Guzdial et al. [10] explore human-AI collaboration types and PCG agent styles. Many co-creation tools, such as Tanagra [11] and Sentient Sketchbook [3], are visual tools that work at the level of detailed level specification. Co-creation with natural language in a game-level generation has received little research beyond narrative authoring. According to Lai et al. [12], co-creation systems must offer designers control and respect their existing workflow. In the case of non-technical level designers using PCG-based co-creation, their comfort level is with their subject matter, and their current workflow uses natural language for communication.

One such challenging area for PCG is educational games, as the content must be innovative, solvable, and aligned with learning objectives. Limited progress has been made in this area, with most studies using pre-existing human-authored content or guidelines [13], [14] . Hooshyar et al. [4] proposed a data-driven approach using a genetic algorithm and support vector machines to automatically generate tailored educational game content. Prior research on PCG in educational games has focused on teaching computer programming. These range from template-based puzzles for programming [13] and training GANs to generate programming challenges [15] or using graph grammars to create challenges for parallel programming [16]. However, automating content generation that aligns with educational goals in other domains like STEM education remains challenging. Our approach involves a game model with a customizable library of game blocks to assemble game levels that align with specific design intentions, as demonstrated in the Future Worlds game, addressing sustainability aspects like river conservation, renewable energy, responsible farming, and eco-friendly urban development.

Our natural language based PCG framework addresses the challenge of an intuitive interface for designers by using pretrained LLMs to convert natural language instructions into game-level constraints and optimization goals for PCG. Pretraining LLMs on a large amount of human-generated text enables them to perform diverse language tasks without retraining. Sometimes, providing only a small set of examples as part of the prompt is necessary [17], [18]. Researchers have fine-tuned these models to respond to human instructions and generate content according to constraints specified in the prompt [19]. These models can output their semantically parsed natural language text as structured data in multiple programming languages [20]. Our research uses these pretrained LLMs to parse level designer preferences into PCG constraints and optimization goals.

We have developed an end-to-end natural-language-based PCG framework to address the content and intuitive interface challenge of educational procedural level generation. Our framework separates educational content creation from game-level design. The level designer is responsible for deciding which aspects of the lesson content the generated level will address and determining the level's challenge. To make our framework highly user-friendly, we allow the level designer to provide their input in free-form natural language text.

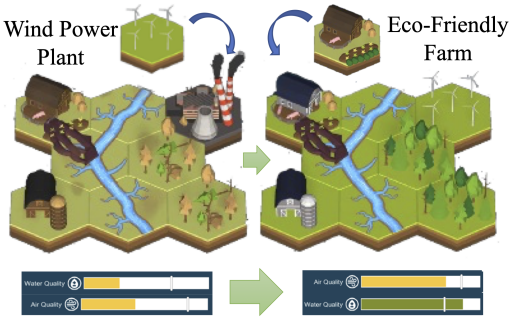

Fig. 1. Gameplay in Future Worlds replacing "Coal Plant" with "Wind Power Plant", and "Polluted Farm" with an "Eco-Friendly Farm".

III. Sustainability-Based Strategy Game

Future Worlds is a 3D virtual game focused on environmental sustainability that enables players to explore the impacts of alternative decisions on water, energy, and food. Players can modify the simulated environment by clicking on hexagonal tiles and experimenting with modifications to improve the sustainability of the environment [21]. In Future Worlds, players can test their hypotheses about various environmental tradeoffs and examine relationships between different environmental choices while accessing visual elements that impart information about ecological features, such as forests, power plants, industrial farms, and rivers. The science content in the game targets learners aged 9-12, and prior research shows its effectiveness in improving learner knowledge and engagement [22].

The game has a hexagonal grid map featuring over 35 environmental factors as tiles, including cities with energy-efficient homes, poorly managed farms, nuclear power plants, polluted rivers, and many more. Each factor affects six components of the simulation that capture the quality of the environment (e.g., air quality, water quality, and biodiversity) as well as the resources available to people living in the environment (e.g., housing availability, food availability, and energy sufficiency). These components were determined by environmental experts and remain constant across levels, allowing players to understand the consistent implications of each factor. For example, building farms would increase food availability, but that comes at the cost of space for building residences, reducing housing availability. On the other hand, building suburban homes reduce the land available for farming.

Each environmental factor in the game-level grid contributes to a numerical score representing environmental quality and resources. Players must modify the environmental factors on the map to achieve specific goals for certain simulation

Authorized licensed use limited to: N.C. State University Libraries - Acquisitions & Discovery S. Downloaded on March 25,2024 at 14:35:49 UTC from IEEE Xplore. Restrictions apply.

Natural Language Instruction	LLM Parsed PCG Instruction
I'm looking for a fun video game that focuses on taking care of water ecosystems and being responsible for habitats. The game should show us why it's important to keep our rivers and their surroundings healthy. It should also teach us how to balance aquatic life and human activities. Additionally, the game should remind us to protect our lands to keep them productive without using up all the resources. It should be challenging but still enjoyable for both beginners and experienced players.	{"difficulty":"medium", "size":"large", "metrics":["Water Quality", "Biodiversity"], "problem":["river pollution", "land management"]}

TABLE I
LEVEL DESIGNER INPUTS AND TRANSLATION INTO STRUCTURED INFORMATION USING LLMS

components. The available options for changes are defined in a transformation matrix. For instance, players can transform a river into a river with a hydropower plant, but they cannot convert a city into a forest. The visual representation of the virtual environment reflects the choices made. A key challenge in the game arises from the tradeoffs between simulation components, as improving one factor may harm another. Players need to understand the interactions between environmental factors to succeed. The game is designed as a multiobjective optimization problem, a versatile approach applicable to various domains. Fig. 1 illustrates gameplay steps that enhance air and water quality. In FUTURE WORLDS, level composition involves various initial environmental factors in a grid, a transformation matrix for factor interchangeability, selected simulation components outlining objective criteria, target values for each component, a move limit for level completion, and a problem description. Designing levels involves verifying that each goal state aligns with environmental sustainability concepts. PCG systems can support designers in creating levels by translating pedagogical intents into potential new levels.

IV. END-TO-END LEVEL GENERATOR FRAMEWORK

Fig. 2 illustrates our end-to-end game level generator that interprets natural language instructions to create game levels. The generator is comprised of four main components. The NL Instruction Parser module acts as an interpreter for the dialogue between the designer and the system, collecting key details such as the game's sustainability topic, targeted metrics, and desired level complexity. The Game Level Generator module then leverages these parsed instructions to formulate various potential game levels, using the appropriate tiles to meet the intended learning objectives and simulation targets. Following this, the DRL Level Evaluator module applies deep reinforcement learning to assess and estimate the difficulty of each prospective game level. Finally, the Playable Level Generation module translates this into a 3D game level in the FUTURE WORLDS Unity game. The following section will delve deeper into these modules.

A. NL Instruction Parser

LLM's zero-shot and few-shot capabilities form the basis for our natural language instruction parser [17], [18]. Earlier language models, such as GPT-2 and BERT, required fine-tuning to a specific problem requiring extensive training data. Recent LLM model iterations have achieved human-level natural language understanding without task-specific training, often requiring only a few examples of text prompts, with zero-shot matching few-shot with the right prompt [20]. Reynolds et al. [20] outline various meta-prompting strategies, employing a multi-step process where the language model generates suitable prompts for the given problem. Our NL instruction parser uses a QA-style prompt model for problem categorization, simulation target metric, difficulty, and size [8]. For instance, a sample prompt template is "given that 'metrics' can only include $< MetricList >$ and given a text $< X >$, list the metrics referred to in the text." Next, we direct the LLM to transform the prior prompt results into JSON objects.

Our natural language based PCG framework uses OpenAI's GPT-3.5 engine [17] for natural language understanding in deciphering level designer instructions. Our framework isn't exclusive to GPT-3.5; other instruction-tuned LLMs could also be used. LLM-based natural language interfaces provide more flexibility than menu-based ones in understanding the designer's intent, especially with vague inputs, and allow the framework to improve without changing the interface.

We identify a 4-tuple (P, M, S, D) as essential for level generation by PCG for FUTURE WORLDS game: P signifies target sustainability issues, M denotes simulation goal metrics, S corresponds to level size, and D indicates difficulty. Table 1 shows an example designer input text and the corresponding level generation parameter output. As noted above, we employ a multi-stage prompting technique. Initially, we extract the four-tuple values, followed by mapping and constraining parameters to game-compatible values.

B. Game Level Generator

In FUTURE WORLDS, a game level consists of a certain number of environmental factors where each factor can be one of 35 different possible tiles. Based on the inputs provided by the level designer in the previous section, we calculate a distribution of weights for the various likely environmental factors. The level designer instruction 4-tuple (P, M, S, D) discussed in the previous section identifies the problem domains that drive the subset of tiles favored in constructing the game level.

- We initialize all environmental factors to have the same weights.
- We add a fixed value to the weight of the factors in the same category as the given P. Suppose the input P stipulates that the game level should focus on affordable housing or farming. In this case, the game level should

Authorized licensed use limited to: N.C. State University Libraries - Acquisitions & Discovery S. Downloaded on March 25,2024 at 14:35:49 UTC from IEEE Xplore. Restrictions apply.

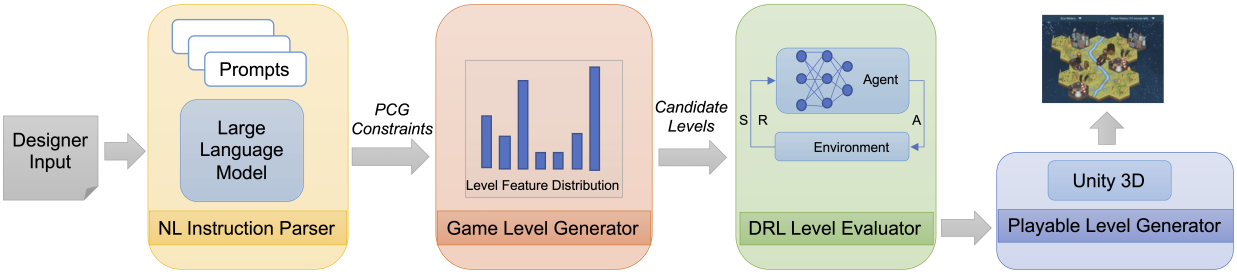

Fig. 2. End-to-end level generator architecture.

prioritize farm, city, or suburban factors more frequently in the grid.
- Based on the provided simulation goal metric M, we rank all factors according to their contribution to the metric. Factors with lower contributions receive higher weights.
- The factors are then sampled based on the updated weights.
- The parsed level designer's instructions S decides the grid size.
- The deep reinforcement learning-based level evaluator, discussed in the next section, assesses the solvability and difficulty of solving the game level to match the level designer instruction D.

The output from the stage includes a list of environmental factors (forest, organic pig farm, hydropower plant, etc.), the transitions allowed for each factor, a set of simulation metrics (Air Quality, Biodiversity, etc.), and the minimum goal values the game board needs for the simulation metrics to solve the game level.

C. DRL Level Evaluator

Game-level solvability and solution difficulty are determined by setting it up as a deep reinforcement learning (DRL) problem to solve the game-level grid. One could also use methods like planning and graph-search for this task without losing generality. We use the OpenAI implementation of the Proximal Policy Optimization (PPO) algorithm for our DRL implementation [23]. PPO implements a policy gradient method that updates the objective function represented by the neural network in mini-batches after multiple sample steps. This method outperforms other policy gradient methods on multiple benchmarks like Atari games and robot locomotion.

The game-level generator generates multiple candidate levels constrained by the 4-tuple (P, M, S, D) attributes. We define the reinforcement problem as follows:

- *Transition Matrix (State)*: A matrix of allowed/blocked transitions from one environmental factor to another.
- *Score Matrix (State)*: The score values for each of the six simulation components for each of the allowed environmental factors. The score values are relative values aggregated to get the score for the grid.
- *Target Score (State)*: Corresponds to goal values for a subset of simulation components, for example: {Air Quality : 3, Biodiversity: 4}.
- *Observation Space (State)*: A set of environmental factors in a given game-level grid.
- *Action Space*: A multi-discrete action space with a 3-tuple (factor category, optimization metric, improve/degrade). The factor category can be one of seven values: forest, farm, land, housing, river, energy, and city. The simulation goal metric can be one of six values: air quality, water quality, biodiversity, food availability, housing availability, and energy sufficiency.
- *Rewards*: Each factor change has a penalty of -1. Each step also gets a Z-scaled difference between current and simulation goal scores if the goal has not been reached. The agent earns 50 points, and the game ends when the target is achieved.

The primary task for players involves determining the appropriate sequence of tile switches to achieve the target score based on the designated goal metrics. Due to constraints imposed by the transition matrix, certain tile transitions are restricted. The problem's difficulty level is characterized by the minimum number of tile modifications required to reach a solution. The DRL model runs on each candidate game level generated to identify the set of solvable game levels with appropriate difficulty, as measured by the minimum number of tile changes.

D. Playable Level Generation

FUTURE WORLDS is a Unity engine-based 3D virtual environment. FUTURE WORLDS was designed using an iterative process involving interdisciplinary collaboration among developers, educators, and subject matter experts in game design, elementary science education, and environmental disciplines. The current implementation of the game contains environmental factor tiles visualized as hexagonal tiles in the game, with associated images and educational content consisting of visual and text-based information about the environmental factor's characteristics and its effect on environmental sustainability. The game engine picks up the game-level details using a data-driven framework. The input to the module consists of the following: A set of environmental factor tiles, a transition matrix detailing all permitted transitions, a tile layout grid, a problem description text generated using the LLM based on the level designer's input parameters, and lastly, the goal values for simulation metrics that will be monitored and assessed throughout gameplay.

Authorized licensed use limited to: N.C. State University Libraries - Acquisitions & Discovery S. Downloaded on March 25,2024 at 14:35:49 UTC from IEEE Xplore. Restrictions apply.

Fig. 3. Example game level progression: (a) The initial puzzle level. (b) Details about a single tile and possible transitions. (c) Simulation metrics value and gap to target for the game level grid.

Gameplay is through clicking and changing individual tiles based on the provided transition options and the information available for each option. Fig. 3 shows a version of the initial problem game level, information dialog, and solution progress for the game level.

V. Evaluation

We evaluate our end-to-end natural language based PCG framework with four metrics: solvability, difficulty, variability, and design intention capture. In the following subsections, we describe the measures and their significance

A. Solvability

Solvability measures the success rate of the DRL level evaluator. Solvability, as we define it, is a metric that evaluates both the ability of the level generator to create solvable levels and the DRL engine's ability to solve the problem. The solvability score might be lower even if the level generator generates candidates that a human could solve but the DRL engine cannot. We explore the solvability of the candidate game level space by exploring various dimensions of simulation metrics. We assess solvability across a range of goal values for these metrics under three distinct correlation scenarios: uncorrelated metrics (Example: Air Quality and Energy Sufficiency), positively correlated metrics (Example: Biodiversity and Water Quality) and negatively correlated metrics (Food Availability and Housing Availability). A positive correlation is when improving one metric also improves the other without changing the game tiles. A negative correlation is when improving one metric decreases the score of the other. Uncorrelated means improving one metric has no impact on the other's score. We divide the range of possible values for metric pairs into bins and generate ten games for each target value bin to evaluate the number of solvable games within each bin.

B. Difficulty

The DRL engine provides a game-level solution, and we evaluate the difficulty as the minimum number of steps required to reach the solution. We create a distribution of difficulty values from a range of solvable candidate levels to evaluate the difficulty metric. In the results section, we explore the difficulty distribution across various problem (P) and metric (M) choices by level designers. We consider difficulty to be a controllable feature of our generator if we can produce candidate levels with diverse difficulty levels, spanning from solutions needing only a single tile modification to those requiring adjustments to all tiles. We use the solvable game levels generated to evaluate solvability to evaluate the difficulty distribution.

C. Variability

Variability of the generated game levels is a metric related to novelty. The distance between two candidate game levels is calculated as the minimum number of distinctly different tiles between the two levels. This distance or corresponding similarity metric is used to quantify variability. To calculate distance, we ignore the transition matrix restrictions used for the game-level solution as we are only interested in the variety of the candidates and not in their solvability. We examine the distribution of distances and corresponding Vendi scores [24]. The Vendi Score measures diversity in a sample set given a similarity function. Similar to difficulty and solvability, we look at the distribution of variability for various simulation metric correlation scenarios. Variability is a controllable metric if we can generate candidate game levels that are typically different from one another on at least half of their tiles. We also look at the expressive range [25], [26] comparing the number of distinct tiles in each generated game level and the corresponding difficulty of the level, for solvable levels. We generate 100 game levels of size 16 tiles for three sustainability problem-goal metric pairs and calculate inter-level distances by counting distinct tiles. Given the definition of distance, the range for distance is 0 to 32 for a 16-tile grid. We independently calculate the distance distribution for the level solution set and the generated problems set as separate distributions.

D. Pedagogical Relevance

We evaluate the levels generated for their alignment to the designer's intention expressed through their instruction. It is crucial to acknowledge that provided goal simulation metrics may permit multiple solutions per level, which may not all align with intended learning outcomes. We analyze the final solution's selected tiles to determine if they address the

Authorized licensed use limited to: N.C. State University Libraries - Acquisitions & Discovery S. Downloaded on March 25,2024 at 14:35:49 UTC from IEEE Xplore. Restrictions apply.

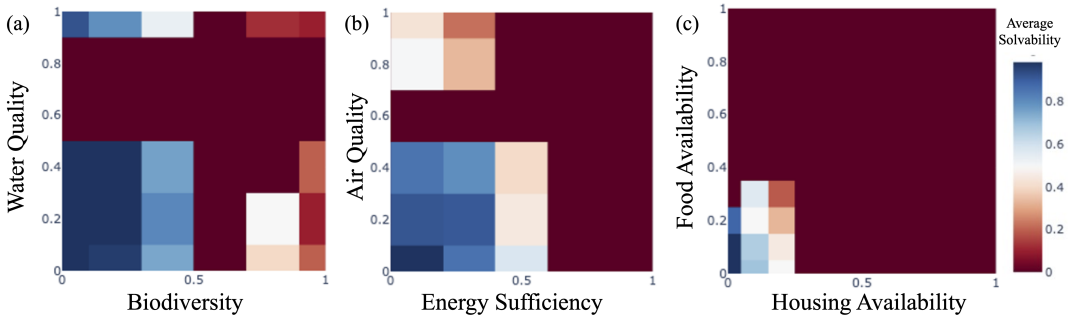

Fig. 4. Solvability by metric value heatmap: (a) Solvability of correlated simulation metrics. (b) Solvability of uncorrelated simulation metrics. (c) Solvability of negatively correlated simulation metrics.

Goal Metric Pair	Initial Level		Solved Level	
	Dist	*Vendi*	*Dist*	*Vendi*
EnergySufficiency - AirQuality	15.1±2	22.9	10.4±3	36.3
Housing - FoodAvailability	15.4±3	16.3	13.9±4	28.0
Biodiversity - WaterQuality	16.1±3	19.7	13.4±4	34.1

TABLE II
DIVERSITY IN INITIAL AND SOLVED GAME LEVELS

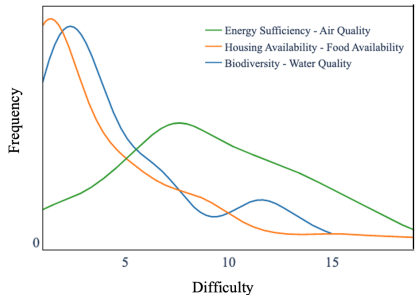

Fig. 5. Game level difficulty distribution for correlated (Biodiversity - Water Quality), uncorrelated (Energy Sufficiency - Air Quality) and negatively correlated (Housing - Food Availability) metric pairs.

problem (P) specified by the designer. The design intention alignment metric involves a manual comparison for end-to-end solution validation. We manually provide a set of instruction text and expected solution environmental factor tiles, then run our framework to obtain game levels and corresponding DRL-generated solutions. The design intention objective is met if the tile change from problem to solution includes the pre-specified environmental factors. A greater correspondence between instruction and solution signifies a higher level of alignment.

VI. Results and Discussion

The relationship between level designer instructions and the generated games is examined, serving as an implicit measure of control offered to the game-level designer. While the instructions specify the overall problem domain and simulation metrics, the game-level generation controls the minimum goal value each simulation metric must reach to complete the game level. Below we discuss the relationship between the problem domain, simulation metrics, and metric goal values and their impact on the generated game-level solvability, difficulty, and variance. All the charts in the following sections are for a game level with 16 tiles.

A. Solvability

Our study explores three scenarios by taking a problem domain pair, such as urban planning and responsible farming, and examining positive, negative, or uncorrelated goal metric pairs. We display our findings in Fig. 4, which shows a solvability heatmap for these three situations. Each cell on the heatmap represents the solvability of a game level. The X and Y axis represents the metric values to reach solvability. Blue cells indicate higher solvability, while red squares indicate lower solvability. For Fig. 4a, we analyze the metric pair "Water Quality" and "Biodiversity." These simulation metrics are positively correlated. This relationship between the simulation metrics allows higher goal values (top right corner) to be solvable (not dark red). Fig. 4b corresponds to the metric pair "Air Quality" and "Energy Sufficiency," which are uncorrelated simulation metrics in the game. The heatmap displays a solvability zone in the lower left portion, corresponding to simulation metrics' mid to lower goal values. In Fig. 4c, we analyze "Food Availability" and "Housing Availability," which are negatively correlated simulation metrics. It is difficult to achieve high values for both of them on a game board, resulting in the smallest solvability zone of the three scenarios. This zone only allows for low goal values for the simulation metrics. Our study shows that the PCG can control solvability when provided with game-level design instructions by leveraging the relationship between metric correlation and metric target values, as demonstrated in the solvability heatmaps of our three scenarios.

B. Difficulty

In our paper, we use the number of steps the DRL-level evaluator takes to solve a level as a measure of difficulty. Difficulty, like solvability, depends on the goal values of the simulation metrics and their correlation. Fig. 5 displays the difficulty distribution (measured as the number of game tile

Authorized licensed use limited to: N.C. State University Libraries - Acquisitions & Discovery S. Downloaded on March 25,2024 at 14:35:49 UTC from IEEE Xplore. Restrictions apply.

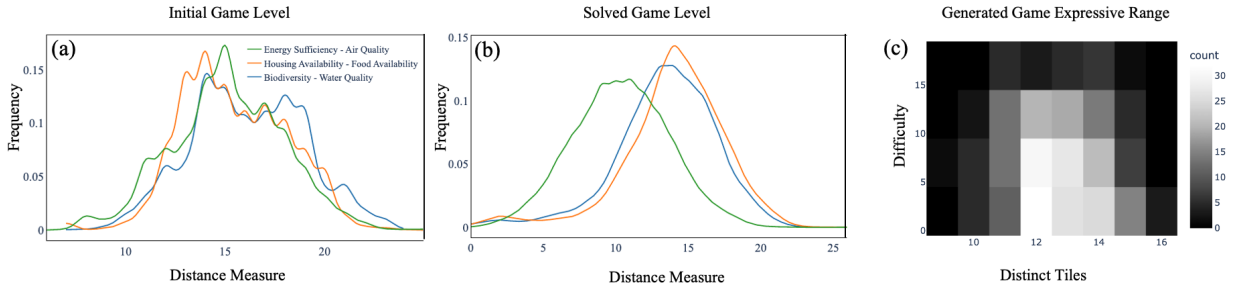

Fig. 6. Game level variability shown as: (a) Initial game level tiles distance distribution (b) Solved game level tiles distance distribution. (c) Expressive range for various difficulty and distinct game level tile counts.

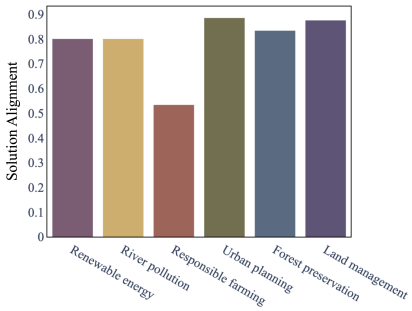

Fig. 7. Alignment of solution to the design intention provided to the natural language based PCG framework.

changes required for a solution) for various combinations of simulation goal metrics. We observe that the negatively correlated metric pair (Housing Availability and Food Availability) has less variety in game-level difficulty than the uncorrelated metric pair (Energy Sufficiency and Air Quality), which has a broader distribution of possible difficulty. This difference is likely due to the limited range of solvable games for the negatively correlated metric pair, resulting in fewer possible levels. Additionally, having smaller goal values for simulation metrics makes reaching the simulation goal value likely with only a few tile changes.

Uncorrelated goal metrics allow for more tile selection options when solving the problem. For example, changing Air Quality tiles does not affect Energy Sufficiency and vice versa. This lack of correlation increases possible game levels and solution combinations. We can adjust metric goal values within set limits through PCG to manage game difficulty. The relationship between these goals influences difficulty distribution. Even with a given choice of goal metrics, we can still develop games with varied difficulty levels (Fig. 5), ensuring PCG aligns with designer intentions.

C. Variability

Fig. 6 displays the distance distributions for the initial (Fig. 6a) and solved (Fig. 6b) game levels, between 100 generated levels as described in the previous section. Initial levels exhibit minimal dependence on goal metric choice and controlling difficulty, and solvability does not affect variability. However, solved levels depend on goal metric choice, which is expected due to tile-type constraints imposed by the goal metric. We note that the initial game level average distance is around 16, which corresponds to at least half the tiles on average being different between the generated game levels, which shows that we can produce good variability in generated levels. Looking at the expressive range in terms of difficulty and distinct tiles in a level (Fig. 6c), we notice a good coverage of the expressive range, albeit biased towards lower difficulty.

Utilizing the Vendi score, we assess the diversity of generated initial and solved game levels (Table 2). A notable trend emerges between average distance and Vendi score; as one moves from the initial to the solved level, the average distance decreases while the Vendi score increases. The trend can be attributed to the method of calculation for distance and similarity. Unlike the distance measure, the similarity measure is independent of the set size. The analysis shows that no loss of variability is observed in solvable game-level generation across different problem domains or goal values of simulation metrics.

D. Pedagogical Relevance

The authors created 25 examples of designer instructions and corresponding tiles we expect in the solution. For instance, if the designer talks about renewable energy, we expect to see solar or wind energy power plant in the solution grid. We assign our input into sustainability categories such as responsible farming, forest preservation, and other related areas. The comparatively low alignment for "responsible farming" could be attributed to the distribution of sustainability metrics across tiles, suggesting that other tiles might handle the same effect intended by responsible farming. For example, one instruction could be, "I want a game that's fun and engaging for kids, but still educational about sustainable fishing and the importance of protecting our oceans and marine life." This example would correspond to the topic of river pollution and would expect to optimize for biodiversity and have the tile, "River with riparian buffers" in the solution. Fig. 7 shows the alignment of the solutions for each of the categories. We demonstrate that for all categories, the alignment is greater than 50%, and the average alignment is 79%. Fig. 7 demonstrates that our end-to-end game-level generation can capture text-based instruction and reflect their intended design objectives in the generated game level.

Authorized licensed use limited to: N.C. State University Libraries - Acquisitions & Discovery S. Downloaded on March 25,2024 at 14:35:49 UTC from IEEE Xplore. Restrictions apply.

VII. Conclusion

We have introduced an end-to-end PCG framework that enables designers to generate levels in a strategy game with natural language instruction. By providing an end-to-end framework that combines natural language interfaces with semantic parsing and deep reinforcement learning techniques, we enable non-technical designers to quickly create game levels that address specific goals related to environmental sustainability. Our evaluation results show that the proposed framework successfully transforms designers' natural language descriptions into fully playable game levels that adhere to the intended design objectives. This work contributes to co-creativity in game design by offering an accessible and intuitive interface for a diverse population, including educators, students, and domain experts, to participate in level design. The developed framework can be applied to various educational game domains, allowing game-level customization to meet the needs of individual level creators.

In Future Worlds, the mutually exclusive effects of environmental factors on goal metrics enable straightforward weighted selection for level generation. As the framework is expanded to other game genres, exploring diverse strategies for generating levels is essential. Promising directions for future work include expanding the genres of games that the PCG framework can support and extending the PCG framework to enable designers to refine game levels continuously. It will also be important to explore LLM-based methods that enable designers to engage in mixed-initiative PCG that supports increasingly expressive co-creation processes.

Acknowledgements

This work is supported by the National Science Foundation under award DRL-2112635. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] G. Lai, F. F. Leymarie, and W. Latham, "On mixed-initiative content creation for video games," *IEEE Transactions on Games*, vol. 14, no. 4, pp. 543–557, 2022.

[2] N. Shaker, J. Togelius, M. J. Nelson, A. Liapis, G. Smith, and N. Shaker, "Mixed-initiative content creation," *Procedural content generation in games*, pp. 195–214, 2016.

[3] G. N. Yannakakis, A. Liapis, and C. Alexopoulos, "Mixed-initiative co-creativity," 2014.

[4] D. Hooshyar, M. Yousefi, M. Wang, and H. Lim, "A data-driven procedural-content-generation approach for educational games," *Journal of Computer Assisted Learning*, vol. 34, no. 6, pp. 731–739, 2018.

[5] D. B. Clark, E. E. Tanner-Smith, and S. S. Killingsworth, "Digital games, design, and learning: A systematic review and meta-analysis," *Review of educational research*, vol. 86, no. 1, pp. 79–122, 2016.

[6] J. P. Gee, "What video games have to teach us about learning and literacy," *Computers in entertainment (CIE)*, vol. 1, no. 1, pp. 20–20, 2003.

[7] K. Isbister, M. Flanagan, and C. Hash, "Designing games for learning: insights from conversations with designers," in *Proceedings of the sigchi conference on human factors in computing systems*, 2010, pp. 2041–2044.

[8] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[9] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[10] M. Guzdial, N. Liao, J. Chen, S.-Y. Chen, S. Shah, V. Shah, J. Reno, G. Smith, and M. O. Riedl, "Friend, collaborator, student, manager: How design of an ai-driven game level editor affects creators," in *Proceedings of the 2019 CHI conference on human factors in computing systems*, 2019, pp. 1–13.

[11] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: Reactive planning and constraint solving for mixed-initiative level design," *IEEE Transactions on computational intelligence and AI in games*, vol. 3, no. 3, pp. 201–215, 2011.

[12] G. Lai, W. Latham, and F. F. Leymarie, "Towards friendly mixed initiative procedural content generation: Three pillars of industry," in *Proceedings of the 15th International Conference on the Foundations of Digital Games*, 2020, pp. 1–4.

[13] Y. Dong and T. Barnes, "Evaluation of a template-based puzzle generator for an educational programming game," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 2017, pp. 1–4.

[14] L. Rodrigues, R. P. Bonidia, and J. D. Brancher, "A math educacional computer game using procedural content generation," in *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, vol. 28, no. 1, 2017, p. 756.

[15] K. Park, B. W. Mott, W. Min, K. E. Boyer, E. N. Wiebe, and J. C. Lester, "Generating educational game levels with multistep deep convolutional generative adversarial networks," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–8.

[16] J. Valls-Vargas, J. Zhu, and S. Ontañón, "Graph grammar-based controllable generation of puzzles for a learning game about parallel programming," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 2017, pp. 1–10.

[17] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[18] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[19] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *arXiv preprint arXiv:2203.02155*, 2022.

[20] L. Reynolds and K. McDonell, "Prompt programming for large language models: Beyond the few-shot paradigm," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–7.

[21] J. P. Rowe, E. V. Lobene, B. W. Mott, and J. C. Lester, "Play in the museum: Design and development of a game-based learning exhibit for informal science education," *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS)*, vol. 9, no. 3, pp. 96–113, 2017.

[22] A. K. Vail, J. F. Grafsgaard, K. E. Boyer, E. N. Wiebe, and J. C. Lester, "Gender differences in facial expressions of affect during learning," in *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, 2016, pp. 65–73.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] D. Friedman and A. B. Dieng, "The vendi score: A diversity evaluation metric for machine learning," *arXiv preprint arXiv:2210.02410*, 2022.

[25] G. Smith and J. Whitehead, "Analyzing the expressive range of a level generator," in *Proceedings of the 2010 workshop on procedural content generation in games*, 2010, pp. 1–7.

[26] A. Summerville, "Expanding expressive range: Evaluation methodologies for procedural content generation," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 14, no. 1, 2018, pp. 116–122.

Authorized licensed use limited to: N.C. State University Libraries - Acquisitions & Discovery S. Downloaded on March 25,2024 at 14:35:49 UTC from IEEE Xplore. Restrictions apply.