MINI AMUSEMENT PARK: A ROAD MAP TO MODELING BUSINESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce Mini Amusement Park (MAP), an amusement park simulator designed to highlight the gap between AI and humans in long-horizon planning tasks. Unlike prior benchmarks, it integrates objective optimization, stochastic transitions, and multi-modal reasoning while resembling a real-world task. We provide environment design, human baselines, and system evaluations, establishing Mini Amusement Park as a challenging testbed for AI.

1 WORK IN PROGRESS

We note that the work outlined in this paper in currently in progress and will be updated in the near future.

2 INTRODUCTION

AI has achieved, and in some cases surpassed, human performance on narrow, well-defined tasks such as board games (Silver et al., 2017), competitive programming (Li et al., 2022), and professional exams including the Bar (Katz et al., 2023) and MCAT (Liu et al., 2024). Yet when faced with the long-horizon, stochastic nature of every day human tasks, for which humans heavily rely on their model of the world (Kuperwajs et al., 2025), AI falls short. This reflects a core limitation: prevailing benchmarks capture narrow competence but miss the reasoning, adaptability, and robustness required for real-world utility. We introduce Mini Amusement Park (MAP) to address this limitation.

Prior benchmarks fall into three categories. First, a set of works, including Crafter (Hafner, 2022) and DiscoveryWorld (Jansen et al., 2024), introduce long-term planning challenges through dependency trees. However, these trees simplify relationships between objects int prerequisites, whereas real-world interactions often involve richer, multi-step dependencies. A second line of work focuses on abstract measures of fluid intelligence. ARC-AGI and ARC-AGI2 (Chollet, 2019; Chollet et al., 2025) evaluate a system's ability for few-shot generalization. While they reveal striking human–AI performance gaps (>70%), their synthetic grid-based puzzles bear little resemblance to real-world tasks. Finally, applied benchmarks expect agents to complete practical tasks. OSWorld evaluates agents on computer tasks (Xie et al., 2024), and VendingBench evaluates an agents ability to operate a vending machine (Backlund and Petersson, 2025). Notably, OSWorld, and all the mentioned environments other than VendingBench, evaluate binary task satisfaction using deterministic transition functions rather than optimizing objectives through uncertainty.

These designs overlook two critical challenges. (i) **Stochasticity**, where any given action can lead to multiple different outcomes. This drastically increases the difficulty of world modeling as there is a distribution of possible outcomes, and no hypothesis can be tested using a single example. (ii) **Optimization**, where *how* a task is completed is important. Real-world problem-solving requires reasoning about efficiency, opportunity costs, and trade-offs, i.e., modeling a continuous measure of success rather than just binary success or failure. While VendingBench (Backlund and Petersson, 2025) does include these challenges, it lacks many desirable qualities found in the other benchmarks such as testing an agent's multi-modal reasoning and ability to learn. Further LLMs already outperform its minimal human baseline.

To address these limitations, we introduce MAP, an amusement park simulator. Like DiscoveryWorld, it requires long-horizon planning, but with more complex object interactions. Like ARC-AGI2, it

demands efficient generalization and multi-modal reasoning, while remaining human-dominated. Like VendingBench, it emphasizes optimization under uncertainty and is grounded in a realistic pursuit: designing, operating, and expanding a business. Crucially, this interweaving of challenges makes MAP a comprehensive testbed for advancing adaptive, robust world models and AI systems.

Specifically, our contributions are: (1) The design and release of MAP. (2) Human performance baselines that provide a meaningful reference point for AI progress. (3) An evaluation of state-of-the-art AI systems, revealing gaps in world modeling, planning, and multi-modal reasoning.

3 RELATED WORK

We outline benchmarks that provide both a well-defined environment and human baselines along five axes: (i) inclusion of an unsaturated human baseline, (ii) task satisfaction vs. objective optimization, (iii) stochastic vs. deterministic transition function, (iv) resemblance to real-world domains, and (v) benefits from multi-modal reasoning. We also consider whether tasks involve long-horizon planning[1] and whether environments are intended for zero-shot assessment or agent learning.

	Core Properties					Secondary Properties		Features		
Environment	Unsaturated	Optimization	Stochastic	Resembles Real Task	Multi-Modal Reasoning	Long-Horizon Planning	Requires Learning	Online Leaderboard	Variable Difficulty	Playable Online
ARC-AGI2	✓	✗	✗	✗	✓	✗	✓	✓	✓	✓
Crafter	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗
IVRE	✓	✗	✗	✗	✓	✗	✓	✗	✗	✗
DiscoveryWorld	✓	✗	✗	✓	✓	✓	✓	✗	✓	✗
WebShop	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗
WebArena	✓	✗	✗	✓	✓	✓	✗	✓	✗	✗
OSWorld	✓	✗	✗	✓	✓	✓	✗	✓	✗	✗
ALE	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗
VendingBench	✗	✓	✓	✓	✗	✓	✗	✓	✗	✗
MAP	✓	✓	✓	✓	✓	✓	✓			

Table 1: Comparison of environment properties and features across benchmarks that report human performance. Green checkmarks indicate presence; red Xs indicate absence.

Task Satisfaction. Most benchmarks that report human baselines focus on satisfiability in deterministic environments. Notably, most human tasks involve stochasticity, and success falls in on a continuum rather than a binary success or failure. Nonetheless, these benchmarks do emphasize from multi-modal reasoning, which is important in many human endeavours.

ARC-AGI2 (Chollet et al., 2025) tests fluid intelligence through few-shot generalization in synthetic grid puzzles, with humans strongly outperforming LLMs. Similarly, IVRE (Xu et al., 2023) tests abstract causal reasoning via a Blicket-inspired setup (Gopnik et al., 2001). Notably neither resembles real-world tasks. Crafter (Hafner, 2022) and DiscoveryWorld (Jansen et al., 2024) investigates long-term planning through dependency trees, but fall short of modeling complex long-horizon inter-object relationships. Lastly, there has also been a surge of benchmarks revolving around real computer-based tasks. WebShop (Yao et al., 2022) requires agents to navigate, customize, and purchase items in simulated e-commerce websites; WebArena (Zhou et al., 2023) introduces long-horizon tasks involving website interaction; and OSWorld (Xie et al., 2024) comprises 369 real-world computer tasks ranging from file system manipulation to formatting spreadsheets.

Objective Optimization. Fewer benchmarks target optimization. The Atari Learning Environment (ALE) (Bellemare et al., 2013) has long served as a canonical human benchmark in AI. While impactful historically, it is now saturated. Moreover, it is deterministic and removed from real-world domains. Most related to our work is VendingBench (Backlund and Petersson, 2025), a vending machine simulation with an objective to maximize profit over hundreds of days with stochastic customers. However, its human baselines are minimal—i.e., a single participant— and are already surpassed by LLMs. Moreover, it was designed as a zero-shot probe rather than requiring agents to learn and adapt — a core hallmark of human intelligence (Tenenbaum et al., 2011). Lastly, it also does not require any multi-modal reasoning, which humans frequently require to solve complex tasks.

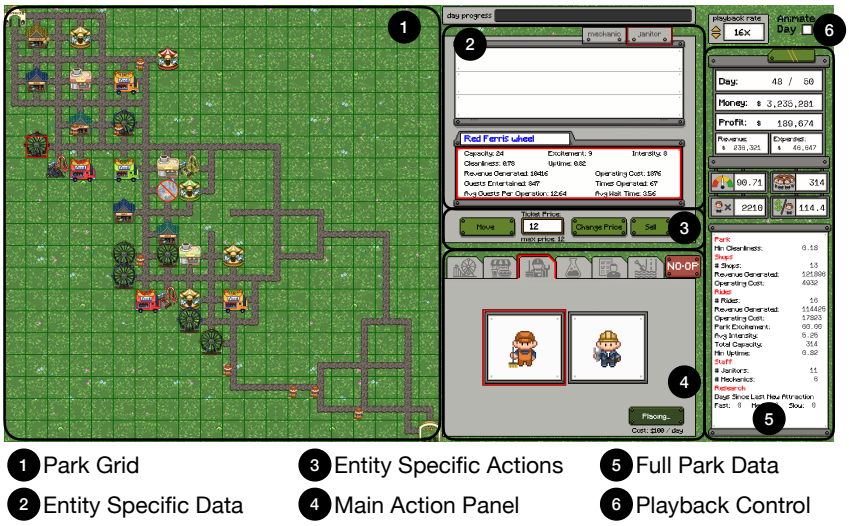

Figure 1: The GUI view of MAP.

4 METHOD

Mini Amusement Park (MAP) is an amusement park simulator designed as a comparative human-AI benchmark which evaluates the ability to optimize profit through long-horizon stochastic planning. In MAP, a player takes the role of a theme park manager. At the start of each day, the manager performs an action to modify their park such as building new rides or shops, hiring staff, or setting a research agenda. Guests then spend the day visiting and interacting with the park. While guests act according to general patterns—e.g., going to a food stall when hungry—their specific decisions—e.g., which food stall they go to—are sampled from a distribution. The park features complex dynamics based not only on the manager's most recent action, but the overall design of the park. MAP currently provides 10 varied starting layouts.

4.1 PARK COMPONENTS

We now outline the six primary components of the game. A full description of each component and their interactions can be found in the game's official documentation (cf. Section D)

The Park. The park is a 20x20 grid. It has an entrance, an exit, and a path connecting the two.

Rides. Rides are attractions that drive the park's capacity and rating. In turn, capacity and park rating determine how many guests visit the park. Rides have three subtypes: carousels, ferris wheels, and roller coasters. Each subtype has their own benefits and drawbacks. Rides are defined by the following attributes: *capacity*, *excitement*, *intensity*, *ticket price*, and *operating cost*.

Shops. Shops are attraction that handle the needs of guests. Shops also have three subtypes. Food shops satiate hunger, drink shops quench thirst, and specialty shops provide a variety of unique benefits. Each shop has an *item price*, *operating cost*, and shop specific attributes.

Subclasses. Each attraction subtype has four tiered subclasses: yellow, blue, green, and red. Yellow attractions are basic and affordable, while red attractions provide significant value at high costs.

Staff. There are two types of staff. Janitors clean dirty tiles, while mechanics repair broken rides. Cleanliness and ride uptime impact park rating; both are critical to a park's success.

Guests. Guests arrive with a varying amount of money, hunger, thirst, and happiness. Throughout the day, guests sample what to do based on their status. They visit rides to increase happiness, food shops to decrease hunger, and drink shops to decrease thirst. Happy, full, and hydrated guests stay in the park longer. Unhappy guests litter, which decreases cleanliness.

[1] $\geq$ 25 steps or explicitly stated

4.2 Observation & Action Space

To enable both human and AI play, MAP provides (i) a *GUI* of the park (ii) a *text-based* representation using JSON objects (via Pydantic). Both observation spaces includes stats about the park's current state (e.g. money, profit, park rating...), as well as detailed information about each employee and attraction. The GUI is depicted in Fig. 1, and an example of the text-based observation is in Section B.

Actions are formatted as a `Python` function, where the function name defines the type of action to be performed and the function parameters mapping to action parameters. Available actions include *placing*, *moving*, *changing the price* of, and *selling* attractions; *hiring*, *firing*, and *moving* staff; *setting research speed and topics*; and *waiting* (i.e., no-op). An additional action, *surveying guest*, is provided to allow park managers to learn about information that is otherwise private to guests, such as the reason a guest left the park. A full description of all actions can be found in the documentation.

5 Benchmarking

We first benchmark the ability for current systems to model the dynamics of MAP. We then compare human performance to the performance pf ReAct (Yao et al., 2023), SPRING (Wu et al., 2023), and WALL-E (Zhou et al., 2024) on MAP. For both SPRING and WALL-E, small adaptations were required to make them work on MAP. We outline these changes, as well a full description of our evaluation protocol in Section A.

Model				
TBD	$TBD_{\pm TBD}$	$TBD_{\pm TBD}$	$TBD_{\pm TBD}$	$TBD_{\pm TBD}$
TBD	$TBD_{\pm TBD}$	$TBD_{\pm TBD}$	$TBD_{\pm TBD}$	$TBD_{\pm TBD}$

Table 2: World modeling ability of various systems.

Model	Profit ($)	Time Taken (s)
Human	$TBD_{\pm TBD}$	$TBD_{\pm TBD}$
ReAct	12943.7 ± 25233.4	14.4s ± 5.2
SPRING	75162.3 ± 93732.5	161.4s ± 38.9
WALLE	10229.1 ± 30099.3	217.7s ± 55.8

Table 3: Performance comparison across different models on MAP

Table 2 shows that current methods to learn world model struggle to learn the complex dynamics of MAP, highlighting the challenge and opportunity for advancement that the environment provides for world modeling research. Table 3 highlights how humans are still the gold standard when it comes to planning in a stochastic world.

6 Discussion

Modeling MAP's World. There are number of factors that make MAP an ideal challenge for world models. First, it requires modeling stochastic transitions. This prevents world models from learning simple rules to mimics the transition function, which is the basis for many current WMs (Zhou et al., 2024; Tang et al., 2024). Instead it requires learning what the distribution of outputs may be, which more closely resembles how the real world operates. Second, the world model is forced to capture the spatial relationships in the park to provide accurate predictions. Evaluating the spatial reasoning of current world models is lacking from existing benchmarks, providing an ideal opportunity for new research directions. Third, the long horizons found in MAP require a world model that is robust to compounding errors (Zhou et al., 2025). This provides an interesting potential to explore hierarchical world models and planning, an avenue that has garnered attention in recent years (LeCun, 2022).

Future Work We are actively developing MAP to improve its use as an ongoing testbed for world modeling and human-AI benchmarking. In terms of the underlying environment, we are developing three additional mechanics: research, terrain tiles, and terraforming. Their implementations will be phased into two new difficulty settings that will also look at longer time horizons (100 and 250

days respectively). We will also define more specific resource settings to allow a wider range of researchers to evaluate their systems in fair ways. These will include true-zero (only the action space is provided), zero-shot (only the game manual and actions space is provided), few-shot (the above + a specific number of in-game transitions to learn from), and unlimited (any and all information can be used). For each combination of these settings, we will run robust human evaluation and a wider range of SotA models. Lastly, we are building a website with a live leaderboard (for both models and human performance), and the ability to play the game online to enable easier adoption by researchers and facilitate standardized evaluation.

REFERENCES

David Silver, Julian Schrittwieser, Karen Simonyan, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Yujia Li, David Choi, Junyoung Chung, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo David Arredondo. Gpt-4 passes the bar exam. *SSRN*, 2023.

Mingxin Liu, Tsuyoshi Okuhara, XinYi Chang, Ritsuko Shirabe, Yuriko Nishiie, Hiroko Okada, and Takahiro Kiuchi. Performance of chatgpt across different versions in medical licensing examinations worldwide: systematic review and meta-analysis. *Journal of medical Internet research*, 26:e60807, 2024.

Ionatan Kuperwajs, Evan M. Russek, Marcelo G. Mattar, Wei Ji Ma, and Thomas L. Griffiths. Looking deeper into the algorithms underlying human planning. *Trends in Cognitive Sciences*, 2025. ISSN 1364-6613. doi:https://doi.org/10.1016/j.tics.2025.06.006. URL https://www.sciencedirect.com/science/article/pii/S1364661325001524.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1W0z96MFEoH.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. Discoveryworld: A virtual environment for developing and evaluating automated scientific discovery agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=cDYqckEt6d.

François Chollet. On the measure of intelligence, 2019. URL https://arxiv.org/abs/1911.01547.

Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arc-agi-2: A new challenge for frontier ai reasoning systems, 2025. URL https://arxiv.org/abs/2505.11831.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=tN61DTr4Ed.

Axel Backlund and Lukas Petersson. Vending-bench: A benchmark for long-term coherence of autonomous agents, 2025. URL https://arxiv.org/abs/2502.15840.

Manjie Xu, Guangyuan Jiang, Wei Liang, Chi Zhang, and Yixin Zhu. Interactive visual reasoning under uncertainty. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 42409–42432. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/844f722dbbcb27933ff5baf58a1f00c8-Paper-Datasets_and_Benchmarks.pdf.

Alison Gopnik, David M. Sobel, Laura E. Schulz, and Clark Glymour. Causal learning mechanisms in very young children: Two-, three-, and four-year-olds infer causal relations from patterns of variation and covariation. *Developmental Psychology*, 37(5):620–629, 2001.

Shunyu Yao, Howard Chen, John Yang, and Karthik R Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=R9KnuFlvnU.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Joshua B. Tenenbaum, Charles Kemp, Thomas L. Griffiths, and Noah D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285, 2011. doi:10.1126/science.1192788. URL https://www.science.org/doi/abs/10.1126/science.1192788.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yue Wu, So Yeon Min, Shrimai Prabhumoye, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li. SPRING: Studying papers and reasoning to play games. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=jU9qiRMDtR.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. Wall-e: World alignment by rule learning improves world model-based llm agents. *arXiv preprint arXiv:2410.07484*, 2024.

Hao Tang, Darren Yan Key, and Kevin Ellis. Worldcoder, a model-based LLM agent: Building world models by writing code and interacting with the environment. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=QGJSXMhVaL.

Guangyao Zhou, Sivaramakrishnan Swaminathan, Rajkumar Vasudeva Raju, J Swaroop Guntupalli, Wolfgang Lehrach, Joseph Ortiz, Antoine Dedieu, Miguel Lazaro-Gredilla, and Kevin Patrick Murphy. Diffusion model predictive control. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=pvtgffHtJm.

Yann LeCun. A path towards autonomous machine intelligence. 2022. URL https://openreview.net/pdf?id=BZ5a1r-kVsf.

A Implementation Details

We divide the 10 starting layouts into 7 train/val layouts, and 3 testing layouts. For methods that require a training dataset (WALLE and WorldCoder), we used a combination of Monte Carlo Tree Search (MCTS) and a heuristic-based policy to generate 12 trajectories per training layout. This resulted in 84 training trajectories. The mean profit of these training trajectories was @ian TODO. Additionally, all methods had full access to the games documentation, which can be found in Section D.

We evaluated all our agents on 12 trajectories per testing layout. We reported the mean and standard deviation for the profit earned across these 36 trajectories. Fo

To evaluate world models, we ...

To evaluate agents playing the game.

A.1 HUMAN EVALUATION

A.2 AI SYSTEM EVALUATION

For building a world model for MAP we adapt Walle 1.0 (Zhou et al., 2024). We modify all the original prompts to incorporate our new state and action specifications. Both the state and action specifications are represented using JSON objects. We further modify the prompt for predicting the next state given the action to incorporate and provide and examples of state transitions. For few shot examples, we associate transitions from the training data with an embedding of the action string, then at inference time we retrieve the top K (5) transitions based on the current action. For rule generation we follow (Zhou et al., 2024), however unlike in Walle 1.0 we do not use different outputs for rules predicting valid and invalid actions. All rules will predict True if the action is valid and false if the action is invalid. For learning the rules, we use randomly sampled trajectories with an error rate of 0.2, however since random trajectories often result in short trajectories due to bankruptcy, we follow the following process: First, we sample a set of trajectories by searching the space for high rewards, next we select a random position in the trajectory, finally we rollout using the random agent with error rate of 0.2. We believe this would result in something akin to having a small set of suboptimal trajectories. For game playing experiments, and after conferring with the authors of WallE to best align with their set up, we combined the WallE world model with a ReACT (Yao et al., 2023) agent using a Model Predicitive Control mechanism (MPC). For each step, we first sample multiple initial actions using ReACT, then we rollout using a single action react and WALLE for k (4) steps. We then compute the rewards at the end of each rollout and take action with the highest scoring rollout.

A.2.1 REACT AND SPRING IMPLEMENTATION DETAILS

We implement a standard ReAct (Yao et al., 2023) baseline; i.e., conditioning on the history of past actions and observation (i.e. game states) before thinking and generating the next action. Our ReAct prompt was modified from LangChain[2], and we limited the history to the past 5 states and actions. We include the game manual in the system prompt alongside details of the game difficulty, length, and objective – i.e., maximizing profits.

For our ReAct model predictive control experiments (i.e., best of n rollouts guided by a world model), we create a variant of our ReAct that prompts the model to generate several actions. We did this instead of sampling several times from standard ReAct as we found that ReAct produced a low diversity of actions. This variant prompt is only used to predict the first action in the rollout. Each action is paired with an independent thought, which is then used to initialize the history of a standard ReAct agent from that point onwards.

In addition to ReAct, we modified SPRING (Wu et al., 2023) to our setting. SPRING is a form of inference-time scaling for game-playing that shapes chain-of-thought via a manually constructed directed acyclic graph (DAG) of questions. Each node corresponds to a question that the LLM must answer – the LLM is provided with its answers to the parent nodes, as well as the current state, the past state, and the game manual. We found the author's DAG of 9 questions, see Table 1 and Figure 3 of Wu et al. (2023), were fairly generic and worked well despite being designed for the Crafter environment (Hafner, 2022). The only changes we made to their questions, q1 through q9, were to rephrase q1 and q3 to be about attractions and staff rather than objects ("List the attractions and staff in the current observation, answer what purpose they serve and any requirements." and "For each entity, are the requirements met? If an entity has no requirements, then they are met."), and directly providing the last action to q3 instead of prompting the LLM to extract this information (i.e., removing q2 and instead providing q4 with the ground-truth last action taken). The removal of q2 was done purely to reduce LLM inference costs. The original SPRING also included a summarization phase that converted the Crafter paper, i.e. Hafner (2022), into Crafter game documentation. We experimented with summarizing our game documentation, but found this degraded performance; generally summarization failed to shorten the documentation, removed critical information or introduced hallucinations. This is unsurprising given that MAPS has a concise, dense, human-written documentation, as opposed to a scientific paper containing significant tangential material, as well as irrelevant references, and figures.

[2] https://python.langchain.com/api_reference/_modules/langchain/agents/react/agent.html

A.2.2 World Coder Implementation Details

TODO: Copy from the NeSy paper after it's been revised based on any feedback.

B Observation Example

C Estimating Size of State & Action Space

Being conservative, we estimate that the effective state space is in excess of 190 trillion unique reachable states. We also estimate that the number of valid actions a competent human player can reasonably encounter in a given turn is at least 177 actions by turn 30. The number of valid actions varies over the game, but is lower bounded by 1 and upper bounded by 867.

We perform our estimate of the state space's size as follows:

Given that the entrance is always on one edge of the map and the exit is always at the opposite edge, the shortest possible path is 18 tiles long, thus providing $2 \times 18 = 36$ possible locations for buildings. Note that in practice our paths are longer. In principle a layout could have the path completely against the edge of the park (in which case there would be only 18 valid placements), however none of our layouts are so designed. As there are 12 rides and 12 shops, and a tile adjacent to a path must be a ride, a shop or empty, there are therefore 25 possible values a tile may have. Therefore there are at least $25^{36} \approx 2.1 \times 10^{50}$ possible building placements.

However, some of these may be unreachable within the 50-timestep limit due to budget reasons. Being very conservative, we estimate that a competent human player's profit per turn consistently exceeds the maximum building cost by turn 30. This would leave 20 turns placing any building; we will conservatively assume that 2 turns are required to place a building at a given location (deleting an existing building and placing a new building). Playing the game manually, we confirmed this is possible (i.e., playing until turn 30 and then only constructing red roller coasters or making space to build red roller coasters). Therefore there are at least 10 new buildings of any type and subtype constructed in any valid location, and so $25^{10} \approx 9 \times 10^{13}$ unique changes that can be made to the initial state at turn 30. We therefore very conservatively estimate the effective number of layouts to be $25^{10} \approx 9 \times 10^{13}$.

In addition to the number of building states, each building and path has a continuous cleanliness $c \in [0, 1]$ making the state space technically infinite. Again being conservative, we will assume it can be discretized to 2 levels (this assumption is incorrect, but we shall proceed) – then this would increase the state space by a further factor of 2^{b+p} where $b, p \in \mathbb{N}$ are the current number of buildings and paths respectively. Many of these states will not however be reachable (e.g., alternating clean and dirty paths are unlikely due to how janitors behave). For this reason we will very conservatively assume a binary state (either the park is clean or dirty), leading us to $2 \times 25^{10} \approx 1.9 \times 10^{14}$.

States can also contain several janitors or mechanics, which can be positioned on any path or building – thought doing so requires an action and therefore removes an opportunity for placing a building. As these states will be orders of magnitude less varied than building placement, we will ignore this in our estimates.

Ignoring the diversity in states reachable by timestep 30 (as these could only be partially overwritten in the last 20 steps of the game), we can very conservatively lower bound the number of reachable states at 1.9×10^{14}, or 190 trillion states. Increasing the time horizon, all building placements are possible and thus the number of reachable states is orders of magnitude above the number of possible building layouts, which alone is 2.1×10^{50} possibilities (211 quindecilion states).

Strictly speaking, the branching factor is fixed throughout the course of the game. Invalid actions are interpreted as a no-op command. In practice the branching factor of *valid* actions is much lower, particularly when the budget is low or space is constrained. If we use the conservative estimate of 36 possible building locations from above, then the upper bound on the branching factor of valid actions is therefore 24 building constructions in any of 36 positions, (in which case we cannot sell a building as they are all sold), hiring a janitor, hiring a mechanic, or skipping the turn, for a total of $24 \times 36 + 2 + 1 = 867$ valid actions. The lower bound is 1 (waiting), if player has no budget and no buildings to sell. By turn 30 a competent human player will be able to afford any

building, and (being conservative) must have at least $36 - 30 = 6$ unoccupied spots. In this case, the branching factor would be 6 spots for building, 30 buildings for sale, hiring, or waiting, for a total of: $24 \times 6 + 30 + 2 + 1 = 177$ actions.

D GAME DOCUMENTATION

The following pages include the games documentation.

Game Mechanics

The purpose of the game is to maximize a theme park's profit given a starting budget and timeframe. As the CEO of the park, you perform one action at the start of the day. The park then opens and guests interact with the park for a full day, which consists of 500 ticks.

There are three difficulty modes to the game ("easy", "medium", and "hard").

- Easy: All attractions are available from the beginning. Water tiles are disabled. Short time horizon (50 days by default)
- Medium: Only yellow attractions are available from the beginning, other attractions must be researched. Layouts can include water tiles. Medium time horizon (100 by default)
- Hard: Only yellow attractions are available from the beginning, other attractions must be researched. Layouts can include water tiles and terraforming actions (i.e. adding/removing paths/water) are enabled. Long time horizon (250 by default)

There are 7 primary components to the game. We provide a high level overview here, and a detailed description of each follows.

- The Park. This is defined by a square grid (defaults to 20x20) and contains all other components. The theme park has an entrance and an exit, which are connected by a path.
- Terrain. There are three kinds of terrain: Paths, Water, and Empty.
- Rides. Rides are one of two types of attractions you can place in your theme park. They are the core of the theme park, and are what draw guests in and keep them happy. There are three subtypes of rides: Carousels, Ferris Wheels, and Roller Coasters. Each has four subclasses: yellow, blue, green, and red.
- Shops. Shops are the second type of attractions you can place in your theme park. They allow you to cater your guests' needs. There are three subtypes of shops: Drink shops, Food shops, and Specialty shops. Drink shops alleviate the thirst of guests. Food shops alleviate the hunger of guests. Specialty shops provide a range of utilities. There similarly four subclasses for each subtype: yellow, blue, green, and red.
- Staff. Staff can be hired to maintain your park. There are two types of staff: janitors and mechanics. Janitors keep your park clean, while mechanics can repair rides that need maintenance.
- Guests. This is who you build the park for! Guests enter your park to enjoy the attractions you've built.
- Research. Early on, you only know how to build certain rides. Investing in research will allow you to build more subclasses of rides and shops as time goes on.

A core feature of your park is your park rating. This rating depends on several factors in your park and is a driving force in how many guests will come visit your park.

Terrain

- Empty tiles: A blank tile on which something can be built.
- Path tiles: The tiles used by guests to move around your park. All attractions must be placed on an empty tile that is adjacent to a path tile.
- Water tiles: The excitement of any ride adjacent to a water tile is increased by 1, however it nothing can be built on it.

In hard mode, terrain can be terraformed. Paths can be built ($1000) and removed ($2500). Water tiles can similarly be built ($5000) and removed ($10000)

Rides

Rides are the core of your theme park. They drive guest attraction and guest happiness. Rides have several key attributes:

- Capacity. How many guests can fit on your ride at a single time. The cummulative capacity of your park is also a key factor in how many guests can visit your park.
- Excitement. How exciting a ride is. Higher excitement scores lead to greater guest happiness. A high excitement score is also crucial to your park rating.
- Intensity. How intense the ride is. Keeping your average intensity balanced (i.e., as close to 5 as possible) will create a park that caters to a wide range of guests, and will help increase your park rating.
- Ticket price. How much money guests have to spend to ride the ride. You are here to make a profit after all. Note that if a guest does not have enough money to pay the ticket price, they will be rejected by the ride, which will decrease their happiness.
- Cost per operation. How much it costs each time you operate the ride.

Rides have no fixed costs, their operating costs depends solely on the number of times it is operated each day.
Rides only operate if guests have boarded, and wait for a few turns after the first guest boards to see if more guests will join. This wait is longer for rides with a larger capacity, but decreases as more guests board the ride. A full ride will always operate.
Rides will sometimes breakdown after operating. While broken down, it is out of service and will not accept guests. Having broken rides will negatively impact your park rating, so it is wise to hire mechanics to fix broken rides.

There are three subtypes of rides: Carousels, Ferris Wheels, and Roller Coasters. Each subtype has distinct characteristics:

- Carousels a cheap to build, operate, and they rarely breakdown. However, they provide limited other benefits.
- Ferris Wheels are an intermediate ride option. They are the ride subtype with the highest capacities.
- Roller Coasters are an expensive, but high-value ride. They boast the highest excitement and intensity scores, but breakdown more than the other types of rides.

Each ride also has four possible subclasses. In medium and hard mode, you only start with the yellow version of each ride, and have to perform research to unlock other subclasses of rides. Here is a general outline of subclasses

- Yellow rides are starter rides. Cheap to build and operate, but otherwise unremarkable.
- Blue rides provide an immediate step up. Often with higher excitement, intensity, capacity, and maximum allowable ticket prices.
- Green rides are the subsclass that provides the highest capacity, but have lower excitement and worse intensity values.
- Red rides are thrill rides. High excitement, high intensity, and have the highest possible ticket prices, but they prone to breaking down.

For the exact parameters of each ride, see All Rides

Shops

Shops are necessary to adequately cater to guest needs. Shops have several key attributes:

- Operating costs: how much it cost to stock the shop each day. Unlike rides, shops cost a fixed amount per day.
- Item price: the price guests pay for the item being sold.

Similar to rides, there are three subtypes of shops:

- Drink. Drink shops sell drinks that quench the thirst of guests.
- Food. Food shops sell food that satiate the hunger of guests.
- Specialty. Specialty shops provide a range of services based on their subclass. See All Shops for a description of each. Notably, guests will never seek out

specialty shops, they will only visit specialty shops if they walk by them.

Again similar to rides, shops have subclasses:

- Yellow food and drink shops are starter shops. They provide a basic version of their product.
- Blue food and drink shops sell an improved (more hunger satiating / thirst quenching) version of their respective product.
- Green food and drink shops provide multiple benefits instead of a single benefit.
- Red drink shops sell coffee which caffeinate guests. Red food shops are luxury food items, providing highly satiating food that also boost happiness.

For exact parameters of each shops, see All Shops

Staff

There are two types of staff: a janitor and a mechanic. Each staff has a salary: $25 for mechanics and $100 for janitors.

Janitors will roam the park, generally moving toward dirtier areas. When on dirty tiles, janitors will clean them.

Mechanics will move toward rides that are broken down, with a preference towards nearer broken rides. When they reach a tile containing a ride that needs maintenance, mechanics will perform repairs until the ride is operational again. Rides will require 0.05% of their total building costs to fully repair once broken, and the speed of repair is proportional to this cost.

Multiple staff can occupy the same tile. Multiple of the same kind of staff on a tile will multiplicatively increase the speed at which the tile is cleaned / repaired.

Guests

Guests are what the park is made for. The amount of guests that visit your theme park is based on the park's rating and the overall capacity of the park. *Capacity* determines both how many guests can be in the park at any given time, as well as how many potential guests consider visiting the park. Capacity is entirely determined by the cummulative capacity of the current rides in the park. **NOTE:** Since only rides increase capacity, a park with no rides will receive no visitors. We aren't making a food hall.

Park Rating determines the likelihood that a potential guest decides to enter the park and become a real guest. New guests cannot enter the park if the park is currently at capacity. Park rating can be increased by increasing the total excitement of the park, and by having guests leave your park happy. Park rating will drop if the park is dirty, rides have low uptimes (i.e., are out of service for portions of the day), or if the average intensity of rides is too high or too low.

Each guests that visits your park will bring with them some amount of money. If they run out of money, they will leave your park. Each guests also has a finite amount of energy. Every step they take in the park will decrease this energy. If they run out of energy, they will leave your park.

Guests also have hunger, thirst, and happiness levels. Hungry guests will seek out food shops. Thirsty guests will seek out drink shops. Unhappy guests will seek out rides. If a guest's hunger or thirst levels get too high, it will negatively impact their happiness. If guests become too unhappy, too thirsty, or too hungry, they will leave your park.

If a guest has all their needs met, they will target any attraction (except specialty shops, see below). If guests pass by another attraction on their way to their target, they may visit that attraction. Guests like novelty and will favor attractions they have visited less frequently. Guests will also favor attractions that are nearby, but they will never visit the same attraction twice in a row. If a guest visits an attraction that they cannot afford or that is currently broken down, their happiness will be affected. **NOTE:** Guests will never seek out specialty shops. They will only visit specialty shops if they pass by them on their way to another attraction.

Guests sometimes litter. This decreases the cleanliness of the path or attraction they are currently on. Unhappy guests are more likely to litter. If guests visit dirty tiles, it negatively impacts their happiness. If a guest visits a ride that is too dirty, they may choose to go elsewhere which will negatively impact their happiness.

In hard mode, guests may have preferences, meaning they will only interact with a subset of attractions. If a guests visits a ride that does not match their preference, it will negatively impact their happiness. Providing guests with information through an information booth (blue specialty shop) will allow them to know ahead of time which attractions match their preference.

NOTE: You can learn more about guests by surveying them through the SurveyGuest action. This provides information about why the guest left the park, what the guest preferences are, and more. You can choose how many guests to survey (up to a maximum of 25) for a price of $1000 per guest.

Research

In easy mode, all attractions are avaiable from the start and research is disabled. This section is only relevant to medium and hard mode.

At the start of the game, you only the yellow subclass of each attraction is available. To learn how to build more subclasses, you have to invest in research. To do this, you have to set your research, which includes setting how fast you want to perform research and the topic(s) of research (where the topics are attraction subtypes). Once set, research will be performed each day according to your settings. This persists until you change the research settings, until you run out of funds, or you successfully research all possible subclasses for the specified research topics. In the latter two cases, the research speed will be set back to "none". Research will always unlock new subclasses in the following order: blue, green, red. Once a new attraction has been unlocked, research will continue on the next topic in your list of topics.

NOTE: if, for example, you want to unlock the red roller coaster as fast as possible, you will want to set the research speed to "fast" and set your research topics to ONLY ["roller coaster"].

Performing research faster requires more money. Below are the research speeds and their rates:

- "none": 0$/day; research is halted at this speed.
- "slow": 2000$/day.
- "medium": 10000$/day.
- "fast": 50000$/day

The default setting is a research speed of "none" and all attraction subtypes selected as topics.

All Rides

Carousels

Yellow

- Building Cost: 250
- Cost per Operation: 1
- Capacity: 7
- Max Ticket Price: 3
- Excitement: 1
- Intensity: 1
- Breakdown Rate: 0.001

Blue

- Building Cost: 1250
- Cost per Operation: 2
- Capacity: 14
- Max Ticket Price: 6
- Excitement: 4
- Intensity: 5
- Breakdown Rate: 0.002

Green

- Building Cost: 7500
- Cost per Operation: 16
- Capacity: 26
- Max Ticket Price: 7
- Excitement: 2
- Intensity: 4
- Breakdown Rate: 0.003

Red

- Building Cost: 15000
- Cost per Operation: 12
- Capacity: 18
- Max Ticket Price: 11
- Excitement: 7
- Intensity: 5
- Breakdown Rate: 0.005

Ferris Wheels

Yellow

- Building Cost: 500
- Cost per Operation: 6
- Capacity: 10
- Max Ticket Price: 4
- Excitement: 3
- Intensity: 2
- Breakdown Rate: 0.006

Blue

- Building Cost: 3750
- Cost per Operation: 8
- Capacity: 22
- Max Ticket Price: 5
- Excitement: 6
- Intensity: 3
- Breakdown Rate: 0.009

Green

- Building Cost: 37500
- Cost per Operation: 40
- Capacity: 42
- Max Ticket Price: 9
- Excitement: 5
- Intensity: 7
- Breakdown Rate: 0.023

Red

- Building Cost: 55000
- Cost per Operation: 28
- Capacity: 24
- Max Ticket Price: 12
- Excitement: 9
- Intensity: 8
- Breakdown Rate: 0.018

Roller Coasters

Yellow

- Building Cost: 1000
- Cost per Operation: 10
- Capacity: 5
- Max Ticket Price: 15

- Excitement: 4
- Intensity: 6
- Breakdown Rate: 0.01

Blue

- Building Cost: 10000
- Cost per Operation: 25
- Capacity: 8
- Max Ticket Price: 20
- Excitement: 8
- Intensity: 8
- Breakdown Rate: 0.02

Green

- Building Cost: 30000
- Cost per Operation: 40
- Capacity: 16
- Max Ticket Price: 18
- Excitement: 6
- Intensity: 9
- Breakdown Rate: 0.03

Red

- Building Cost: 75000
- Cost per Operation: 50
- Capacity: 10
- Max Ticket Price: 50
- Excitement: 10
- Intensity: 10
- Breakdown Rate: 0.033

All Shops

Drink Shops

Yellow

- Building Cost: 100
- Operating Cost: 16
- Max Item Price: 2
- Thirst Reduction: 0.5

Blue

- Building Cost: 1750
- Operating Cost: 75
- Max Item Price: 6
- Thirst Reduction: 0.9

Green

- Building Cost: 17500
- Operating Cost: 280
- Max Item Price: 10
- Thirst Reduction: 0.8
- Happiness Boost: 0.4

In addition to quenching thirst, green drink shops additionally provide a boost to guest happiness.

Red

- Building Cost: 48000
- Operating Cost: 600
- Max Item Price: 15
- Thirst Reduction: 0.4
- Energy Boost: 50
- Caffeinated Steps: 50

Red drinks caffeinate guests, which boosts a guest's energy and allows them to move twice as fast

Food Shops

Yellow

- Building Cost: 150
- Operating Cost: 28
- Max Item Price: 4
- Hunger Reduction: 0.4

Blue

- Building Cost: 3000
- Operating Cost: 150
- Max Item Price: 10

- Hunger Reduction: 0.8

Green

- Building Cost: 32000
- Operating Cost: 500
- Max Item Price: 18
- Hunger Reduction: 0.6
- Thirst Reduction: 0.6

Green food shops both satiate hunger and quench thirst.

Red

- Building Cost: 60000
- Operating Cost: 1000
- Max Item Price: 25
- Hunger Reduction: 0.9
- Happiness Boost: 0.5

Red food shops sell luxury food. This greatly satiates hunger and increases happiness

Specialty Shops

NOTE: Guests will not target specialty shops, and will only visit specialty shops if the walk adjacent to one.

Yellow (Souvenir Shop)

- Building Cost: 250
- Operating Cost: 80
- Max Item Price: 12
- Happiness Boost: 0.3

These provide a happiness boost to guests that buy them the first time, but this happiness boost diminishes with each subsequent souvenir purchased.

Blue (Information Booth)

- Building Cost: 10000
- Operating Cost: 200
- Max Item Price: 5

These provide information to guests about the rides in the park and ensure that guests only visit rides that fall within their budget and preferences. These do not provide guests about information related to the cleanliness or operation (i.e., if an attraction is out of service) of an attraction.

Green (ATM)

- Building Cost: 50000
- Operating Cost: 500
- Max Item Price: 2
- Money Withdrawal: 50

ATMs allow guests to withdraw more money. The amount of money withdrawn decreases exponentially with every subsequent withdrawal, to a minimum of 5

Red (Billboard)

- Building Cost: 10000
- Operating Cost: 0
- Max Item Price: 0
- Thirst Boost: 0.5
- Hunger Boost: 0.5

Billboards make guests more hungry and thirsty, will reset the visit count of attractions (meaning that guests are more likely to revisit attractions), and, if the guest has less than $20, will set the guest's target to an ATM.

Action Space

Actions must be written as python function calls with keyword arguments. These action functions must be in the following format:

```
action_name(param_1=<param1_value>, param_2=<param1_value>, ... )
```

For example:

```
place_attraction(x=5, y=7, type='ride', subtype='carousel', subclass='red', price=8)
```

Would specify placing a red carousel with a ticket price of $8 at location (5, 7).

The full list of available actions, including the action names, parameters, and a description of what they do is below.

All Actions

place_attraction
Description: Place an attraction (ride or shop) in the theme park
Parameters:

- x: The x position of the ride
- y: The y position of the ride
- type: The type of attraction. Must be one of ride or shop
- subtype: The subtype of the ride or shop to be placed.For rides, must be one of carousel, ferris wheel, or roller coaster. For shops, must be one of drink, food, or specialty
- subclass: The specific instance of the ride or shop to be placed.Must be one of yellow, blue, green, red, or custom (custom is only available for rides)
- price: The price of the ticket or item

move_attraction
Description: Move an attraction (ride or shop) in the theme park
Parameters:

- type: The type of attraction. Must be one of ride or shop
- x: The current x position of the attraction
- y: The current y position of the attraction
- new_x: The new x position
- new_y: The new y position

sell_attraction
Description: Sell an attraction (ride or shop)
Parameters:

- type: The type of attraction. Must be one of ride or shop
- x: The x position of the attraction
- y: The y position of the attraction

change_price
Description: Change the price for an attraction
Parameters:

- type: The type of attraction. Must be one of ride or shop
- x: The x position of the attraction
- y: The y position of the attraction
- price: The new price

hire_staff
Description: Hire an employee of a particular type and place them in the theme park
Parameters:

- type: The type of employee. Must be janitor or mechanic
- x: The x position of the employee
- y: The y position of the employee

fire_staff
Description: Fire an employee
Parameters:

- type: The type of employee. Must be janitor or mechanic
- x: The x position of the employee
- y: The y position of the employee

move_staff
Description: Change the position of an employee
Parameters:

- type: The type of employee. Must be janitor or mechanic
- x: The current x position of the employee
- y: The current y position of the employee
- new_x: The new x position of the employee
- new_y: The new y_position of the employee

set_research
Description: Set the research speed and topic(s) for the theme park. Only available in medium or hard difficulty.
Parameters:

- research_speed: The research speed. Must be one of none, slow, medium, or fast
- research_topics: The topics(s) of research. Must be a subset of ['carousel', 'ferris_wheel', 'roller_coaster', 'drink', 'food', 'specialty']

survey_guests
Description: Retrieve a sample of information from guests
Parameters:

- num_guests: The number of guests to survey

add_path
Description: Add a path tile to the theme park. Only available in hard difficulty.
Parameters:

- x: The x position of the path tile
- y: The y position of the path tile

remove_path
Description: Remove a path tile from the theme park. Only available in hard difficulty.
Parameters:

- x: The x position of the path tile
- y: The y position of the path tile

add_water
Description: Add a water tile to the theme park. Only available in hard difficulty.
Parameters:

- x: The x position of the water tile
- y: The y position of the water tile

remove_water
Description: Remove a water tile from the theme park. Only available in hard difficulty.
Parameters:

- x: The x position of the water tile
- y: The y position of the water tile

wait
Description: runs the day without any new action
Parameters:
No parameters