Generative Design through Quality-Diversity Data Synthesis and Language Models

Adam Gaier
Autodesk Research
Bonn, Germany
adam.gaier@autodesk.com

James Stoddart
Autodesk Research
Atlanta, USA
james.stoddart@autodesk.com

Lorenzo Villaggi
Autodesk Research
New York, USA
lorenzo.villaggi@autodesk.com

Shyam Sudhakaran
Autodesk Research
San Francisco, USA
shyam.sudhakaran@autodesk.com

ABSTRACT

Two fundamental challenges face generative models in engineering applications: the acquisition of high-performing, diverse datasets, and the adherence to precise constraints in generated designs. We propose a novel approach combining optimization, constraint satisfaction, and language models to tackle these challenges in architectural design. Our method uses Quality-Diversity (QD) to generate a diverse, high-performing dataset. We then fine-tune a language model with this dataset to generate high-level designs. These designs are then refined into detailed, constraint-compliant layouts using the Wave Function Collapse algorithm. Our system demonstrates reliable adherence to textual guidance, enabling the generation of layouts with targeted architectural and performance features. Crucially, our results indicate that data synthesized through the evolutionary search of QD not only improves overall model performance but is essential for the model's ability to closely adhere to textual guidance. This improvement underscores the pivotal role evolutionary computation can play in creating the datasets key to training generative models for design. Web article at https://tilegpt.github.io

KEYWORDS

Quality-Diversity; MAP-Elites; Language Model

ACM Reference Format:
Adam Gaier, James Stoddart, Lorenzo Villaggi, and Shyam Sudhakaran. 2024. Generative Design through Quality-Diversity Data Synthesis and Language Models. In *Genetic and Evolutionary Computation Conference (GECCO '24), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3638529.3654138

1 INTRODUCTION

Generative Design (GD) in architecture represents a paradigm shift in the way designs are conceptualized and realized. It draws inspiration from natural evolution to explore vast design spaces to discover high-performing, innovative solutions [34]. At its core, GD involves a geometry generator that delineates a broad solution space, coupled with simulations and analytical methods for evaluating each design against a set of metrics. Metaheuristic search algorithms, such as genetic algorithms, navigate this space to identify optimal solutions [32]. This approach is versatile and scale-agnostic, making it applicable to a wide range of design problems and scales.

In Architecture, Engineering, and Construction (AEC), GD is most commonly used in the early stages of design [3]. This is when the potential to influence outcomes is highest, and the cost implications of design changes are minimal [36, 40]. GD has been successfully applied in numerous AEC projects, enabling practitioners to tackle complex challenges, balance conflicting objectives, and make informed decisions based on solid evidence [32, 33, 46].

But the development and deployment of GD methods requires a high level of technical expertise, which limits their scalability and accessibility. Not only that, the results of GD are large sets of complex solutions, requiring designers to spend as much effort on analysis as on creation. Worse still, typical GD workflows offer limited scope for interaction – making changes often necessitates rerunning the entire optimization process.

Large language models (LLMs), which have streamlined many tasks, could also be applied to design. LLMs fine-tuned on labeled segments of existing Mario Bros. levels are able to generate levels which reflect descriptive prompts (e.g., "few enemies," "many pipes") [42, 43]. Many tasks in architectural design, particularly in the conceptual phase, can be modeled at a similar level of abstraction as video game levels. Experimentation has already begun in AEC to take advantage of the same tile-based layouts and procedural content generation (PCG) techniques used in games. [24, 47].

Crucially, to adapt an LLM-based approach to design in this way it is necessary to have a large corpus of labeled data. Quality-Diversity (QD) [6, 37] approaches are able to generate large collection of solutions, ideal for use as training data. These high performing collections of span user-defined features, allowing users to define the design features to explore, and then generate a bespoke dataset that spans those features.

Designs generated by language model are created only through the learned statistical relationships, but in design it is necessary that constraints are followed. Rather than forcing the LLM to learn every constraint, we instead task it with creating a conceptual plan which is then handed to the Wave Function Collapse (WFC) algorithm [19], a PCG approach based on constraint satisfaction [23].

GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0494-9/24/07.
https://doi.org/10.1145/3638529.3654138

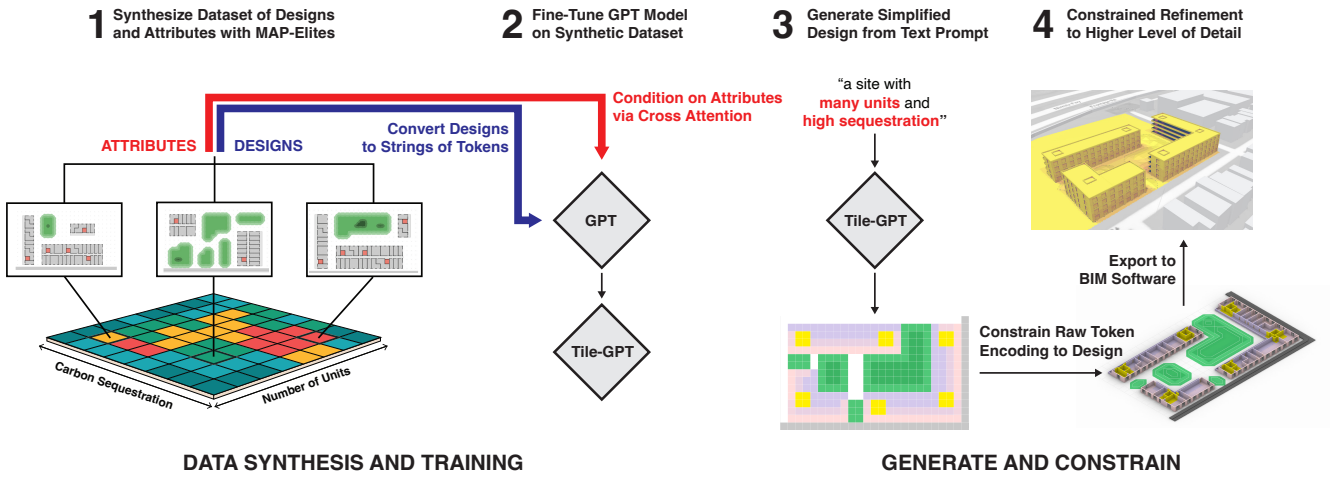

Figure 1: Algorithm flow of the proposed generative design approach, TileGPT. (1) A dataset of paired designs and attributes is generated with the MAP-Elites algorithm, which is used to (2) fine-tune a GPT model to produce designs with given attributes. (3) Given a natural language description a simplified design with the described attributes is generated by the GPT model, and (4) given to a constraint satisfaction algorithm, which refines it into a detailed site plan.

The language model interprets the designers intent through natural language prompts, and generates designs at a high level, such as the placement of buildings and green spaces. These designs are then processed by WFC to generate the detailed layout of modules. This method ensures that the final design not only resonates with the input provided by the designer but also rigorously complies with the constraints of construction.

The outlined system, dubbed TileGPT, demonstrates:

- The integration of QD with PCG techniques to synthesize tailored labeled datasets of high performing solutions.
- The use of a fine-tuned LLM to interpret and implement design directives in natural language and apply them to a real-world generative design case.
- The application of constraint satisfaction to guarantee the validity of designs generated by an LLM.

This novel approach integrates QD, LLMs, and constraint satisfaction within the GD framework. This integration aims to enhance the accessibility of GD methods, reduce the technical barriers to their use, and provide more intuitive, interactive design manipulation capabilities through natural language inputs.

2 BACKGROUND

2.1 Wave Function Collapse

Wave Function Collapse (WFC) is a procedural content generation technique, popularized by Maxim Guman [19] for creating 2D and 3D content, in the form of a constraint satisfaction algorithm. It is similar to the Example-Based Model Synthesis [30] method and is adept at generating non-tiling, self-similar structured data based on sparse input examples.

The algorithm works through iterations of a single cell collapse — assignment to a single fixed state — and neighborhood propagation — where surrounding tiles are constrained to compatible patterns with the collapsed cell. Cells are collapsed in order of minimum entropy, measured as the certainty of a specific outcome from the weightings of potential states, precisely defined as:

$$\text{Shannon Entropy} = \log\left(\sum w_i\right) - \frac{\sum(w_i \times \log(w_i))}{\sum w_i} \quad (1)$$

where w_i represents the weight of each potential state for a cell. The weight reflects the likelihood or frequency of a particular state occurring based on the adjacency constraints and the neighbors.

The WFC methodology consists of a four-step process:

(1) *Pattern extraction*: Utilizing one or more self-similar input examples, WFC identifies cell adjacencies, which are used to form a domain of possible constrained states.
(2) *Initialization and pre-constraint*: The output is initialized with each cell represented as an array of potential states. Individual cells can be pre-constrained to a subset of these states. This is commonly used to enforce boundary conditions or enable controlled generation from an initial pattern.
(3) *Cell collapse*: The output cell with the lowest entropy value is selected for collapse. From the selected cell's possible states, a single, final state is chosen using a weighted random selection and all other potential states discarded. In the event more than one cell has the same lowest entropy value, the cell to collapse is chosen randomly from the candidates.
(4) *Propagation*: After a cell is collapsed, the solver iterates through the adjacent cells and removes all pattern states incompatible with collapsed cell.

The solver repeats steps 3 and 4 until the output is fully collapsed, with each cell assigned to a single state, or until a contradiction arises, indicating that the solver cannot satisfy all constraints.

2.2 Quality-Diversity

Quality-Diversity (QD) approaches, like MAP-Elites [5, 31], search for high-performing solutions which cover a range of user-defined features. Generating diversity along features rather than only objectives makes QD well suited to the needs of GD, as designers are often interested in other attributes beyond objectives [3]. QD has been applied in various design domains including aerodynamics [9, 20, 21], game design [1, 16, 17] and architecture [13, 14, 47].

QD produces a collection of solutions in a single run. The ability to produce numerous high-quality and varied solutions positions QD as an ideal tool for synthesizing datasets for machine learning. Collections of solutions generated through QD have been effectively used in creating surrogate models that predict performance [10, 25, 50], building generative models to aid optimization [12, 38] and exploration [11, 18, 35], creating conditioned reinforcement learning policies [8], and fine-tuning language models to produce virtual creature body plans [27]. In this work, we leverage the designs produced by MAP-Elites to fine-tune and condition a language model to produce designs based on text prompts.

2.3 Language Models

Large Language models (LLMs) are powerful and versatile, able to learn from massive datasets for sequence modelling tasks such as generating text [4], code [15, 29], and multimodal outputs such as images and robot states [7]. These models leverage attention mechanisms [45] to capture patterns in long term sequences. Pre-trained LLMs can be fine-tuned for diverse downstream sequence modelling tasks, reusing the models parameters as a starting point and adding an additional layer trained from scratch. These tasks are not limited to text, but can be generalized to other sequences, such as tile based layouts. Several works have explored this in the context of video games, including MarioGPT, a fine tuned LLM for Mario level generation [42, 43]. The authors showed that MarioGPT was able to generate coherent and playable levels whose layout could be guided by text – an approach we build upon in this work.

3 METHOD

The TileGPT system, described in detail below, proceeds as follows:

(1) Dataset Generation
  (a) A dataset is generated by using MAP-Elites to search the space of designs that can be generated by WFC.
  (b) Each design in the dataset is paired with a text label such as 'many units' or 'little carbon sequestration' based on the features of the design.
(2) Model Training
  (a) A GPT model is fine-tuned using this dataset of designs, adapting it to produce layouts one tile at a time.
  (b) Attribute labels are converted into numerical vectors via a text encoder and incorporated into the fine-tuning process through a cross attention layer.
(3) Layout Generation
  (a) The GPT model is provided with a natural language prompt corresponding to the desired attributes and generates a design at a low level of detail.
  (b) These rough layouts are refined by the WFC algorithm, ensuring local constraint satisfaction and validity.

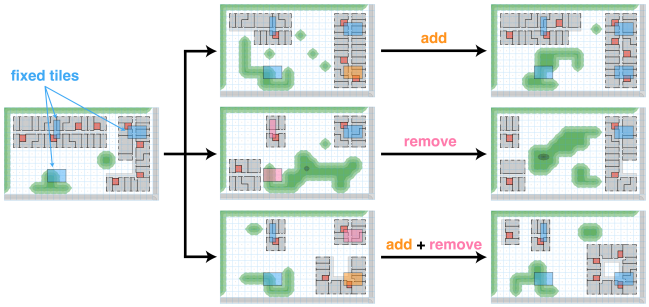

Figure 2: Mutation of a WFC genome. Fixed tiles are encoded into the genome, and set at the start of a WFC rollout, influencing the development of the final design.

3.1 Dataset Generation

Synthesizing Data with MAP-Elites. WFC stands out for its ability to generate a wide array of unique designs from minimal initial examples, a potential we leverage for the generation of synthetic datasets. By conducting numerous iterations or 'rollouts' of the WFC algorithm, a large volume of data can easily synthesized.

Despite its versatility, WFC-generated designs are not ideal samples, particularly when performance of the designs is a priority. In a domain like site design, common issues with WFC include inefficient utilization of space, which could be better employed for buildings or landscaping. Furthermore, there's a tendency for the attributes of the designs to converge towards average values, leading to a dataset that lacks extremes and, as a result, limits the scope of what models can generate. The sparsity of varied and compelling examples in the dataset restricts the model's ability to produce innovative designs or to respond appropriately to text prompts.

To overcome challenges related to the quality and uniformity in design generation, we use MAP-Elites for data synthesis. By adopting a diversity-based optimization strategy, we actively seek out high-quality solutions that encompass a broad spectrum of features. This method moves beyond simply sampling, ensuring the creation of a dataset that is both diverse and of high quality. The enriched dataset thus obtained is pivotal in training our model, enabling it to produce designs that are not only varied but also superior in quality. This refined approach significantly boosts the model's capability to generate diverse site layouts, enhancing the overall effectiveness of the design process.

Optimization of Designs with WFC. To use WFC in the optimization process, we must devise a way of effectively searching the space of WFC produced designs. The core strength of WFC is that it is capable of producing a large variety of solutions that follow a consistent style and set of constraints. This constrained expressivity makes it an appealing option for optimization, but searching the space of solutions through WFC is challenging. The variety in WFC comes from the chaotic elements of its generation process – small changes in the initial conditions or early choices have dramatic consequences for the final result.

The a common lever to guide WFC is to adjust the probability of each tile being chosen when a cell is 'collapsed'. Macro level differences are possible to induce in this way, but it is impossible

to replicate or preserve distinct tile patterns. Adjusting tile weights alone does not produce a suitable encoding for optimization. An encoding based on a tile weight genotype and fully collapsed tile phenotype is highly non-local [39] – a small change in the genotype produces a large and unexpected change in the phenotype, dooming any search algorithm to be little better than random.

We can consider the mapping of genotype to phenotype through the intermediary of WFC as a *probabilistic* encoding, where each genome maps to a distribution of phenotypes. To create an encoding which is more local, and so more amenable to search, we must narrow this distribution while also making it heritable.

At the start of the WFC algorithm we can fix a set of tiles, preserving a few existing parts of the parent design and allow the algorithm to generate the remainder. These fixed tiles can be included as part of the genome and passed on to child solutions. A genome composed of fixed tiles and tile weights is a more local encoding – children resemble parents, and small changes in genotype typically produce small changes in phenotype. The more tiles which are fixed the narrow the distribution of possible mappings from genotype to phenotype.

We can further instantiate individuals by including a random seed, ensuring that a given genome always produces the same phenotype. The resulting genotype is represented as a tuple comprising tile weights, fixed tiles, and a seed. It takes the form:

$$\text{Genotype} = (T_{\text{weight}}, T_{\text{fixed}}, \text{Seed})$$

Where, T_{weight} is a vector of tile weights, T_{fixed} is a list of tuples with each tuple representing a tile type and its position in the grid.

To search this space, we apply a mutation operator, which involves the following steps:

(1) The T_{weight} vector is modified by the addition of Gaussian noise, adjusting the weights either upward or downward.
(2) Tiles are added or removed from the T_{fixed} list.
(3) Seed is reset to a new random integer.

At each generation an equal number of individuals are chosen to have tiles removed and added. Tiles are chosen to be added or removed randomly, and the number added or removed drawn from a uniform distribution between 1 and 4 tiles.

Fixed tiles are added from the phenotype of the parent solution. Adding tiles in this way not only allows children to inherit the same structures, it ensures that the constellation of fixed tiles is a valid one – we know there must be at least one valid phenotype to be found by WFC with that set of tiles. The process of fixed tile mutation is illustrated in Figure 2.

The iterative adding and removing of tiles allows a search algorithm to purposefully search through the space of designs generated by WFC – designs which are guaranteed to follow the guidelines and requirements of the designer.

Dataset Preprocessing. The number of potential tiles, considering their rotations and reflections, can easily reach into the hundreds – and each tile comes with its own unique set of adjacency rules[1]. Training a GPT model to predict tokens at this level of granularity distracts from its central objective: facilitating global-level optimization and exploration.

[1] see Appendix C for the full set of 216 tiles in our application

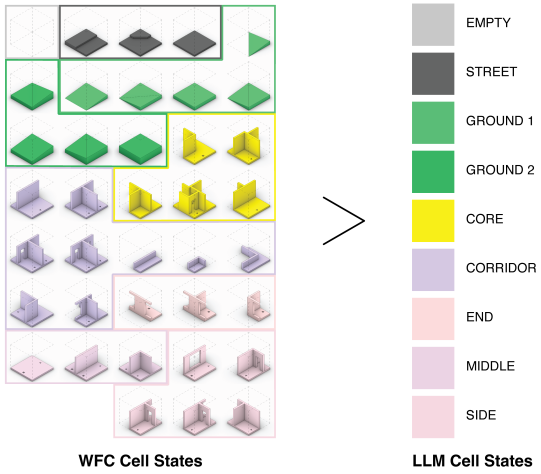

Figure 3: Possible WFC cell states and their simplifications for tokenization. Designs are evaluated using the WFC cell states, but generated using the reduced set of LLM cell states.

Our approach positions the GPT model as a strategic director in the design process. Its role is not to micromanage the minutiae of tile adjacencies but to guide overarching design decisions. This perspective aligns the model's strengths with the demands of high-level conceptual design, and steers clear of the intricacies of individual tile relationships. To streamline this process, we categorize the full tile set into a smaller set of distinct functional groups, as illustrated in Figure 3. This categorization substantially reduces the complexity the GPT model has to manage.

Designs are represented as a grid of tiles, but to convert these designs into tokens we transform each into one of these functional categories. Subsequently, each category is represented by a unique character (e.g., 'A', 'B', 'C'). We then flatten this grid of characters into a vector format to fit the standard sequence completion training paradigm of GPT models. Each site's features – defined by their coordinates in the MAP-Elites grid – are paired with their respective design. These are then translated into high-level natural language descriptions during training (e.g., "few/some/many parks").

3.2 Language Model Training

A causal language model is finetuned to learns "next tile prediction", analogous to the "next token prediction" objective which most causal language models are optimized for – the model learns to generate a design by predicting a single tile based on a sequence of previous tiles. Previous work has shown that by finetuning LLMs for tile generation they can generate new playable levels in Sokoban[44] and Mario Bros[42]. Similar to [42], we choose a distilled version of GPT2 (DistilGPT2) [41] as our base LLM to finetune, with additional cross attention weights used for prompt conditioning. To incorporate these prompts, we utilize a frozen text encoder (BART) [28] to embed the prompts as a vector of floats. These vectors are averaged and used in the cross attention weights in combination with the encoded tile sequence. All previous tiles are used as context for predicting the next tile. The architecture is illustrated in Figure 5.

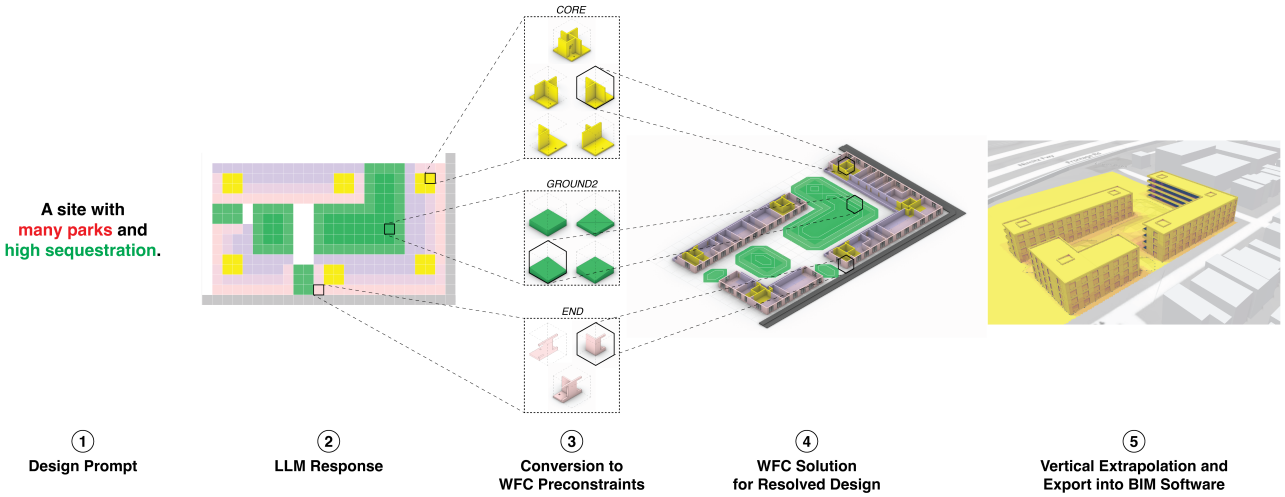

Figure 4: Layout Generation in TileGPT. (1) A site description is provided to the model, which (2) produces a high level layout. (3) This layout is converted into preconstraints for the WFC algorithm, which (4) generates detailed geometry. The 2D geometry can be then be extruded (5) into a form suitable for use with commercial design software.

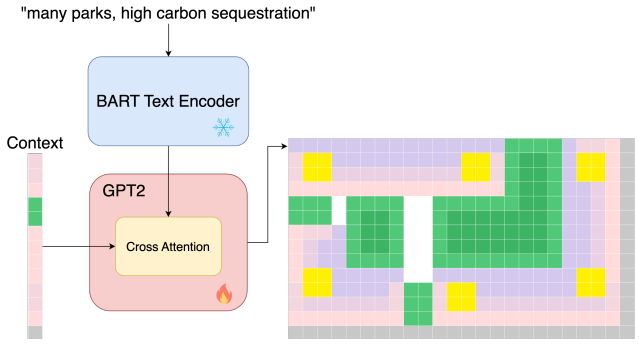

Figure 5: TileGPT architecture. Text prompts are encoded through a frozen text encoder and are combined with previous tiles in GPT2's cross attention mechanism.

Because we use DistilGPT2, the model in TileGPT is relatively small and utilizes only 96 million trainable parameters. This allows for training efficiently on a single GPU. We train TileGPT for 500,000 steps, sampling 16 random designs uniformly at each training iteration and optimize weights using Adam optimizer [26].

3.3 Layout Generation

To use the model for design generation, we follow the a series of steps, as depicted in Figure 4. In this integrated process, the GPT model lays the foundation for the overarching design based on natural language prompts, while WFC ensures its practical feasibility and completeness.

Step 1: Design Initiation via Prompt. The process begins with the input of a design prompt. This prompt incorporates the natural language parameters our model has been trained on. The system inserts randomly sampled prompts for those not provided. These prompts are converted to a vector and used as a constant input to the cross-attention layer – laying the groundwork for the subsequent design generation.

Step 2: LLM-Driven Site Design Formation. Following the initial prompt, the GPT model, steered by the textual input, engages in an iterative process of selecting tiles from a simplified set of categories. These selections form a high-level blueprint, outlining the fundamental structure of the design.

Step 3: Translation to Permissible Tile Sets. The basic tile types delineated by the GPT model are then transformed into a set of allowable tiles. For instance, a 'building core' might be represented in every possible orientation. This step refines the blueprint, preparing it for more detailed procedural generation.

Step 4: Detailed Design Completion through WFC. The refined blueprint is subsequently transferred to the WFC. WFC selects from a comprehensive tile-set to add intricate details, from orientations to placement of windows and interior walls.

Step 5: Finalizing a Valid Design. Upon completion of WFC, we obtain a single, valid design. This design is not only complete in its structure but also readily transferable to Building Information Modeling (BIM) software for detailed editing and analysis.

Importantly, this procedure is not rigid. Users have the flexibility to modify the design iteratively. For example, a portion of the site can be erased and re-generated by inputting an alternative text prompt, directing the system to refill the area with a design that incorporates specific desired features. This iterative capability enhances the adaptability and user-interactivity of our design generation process.

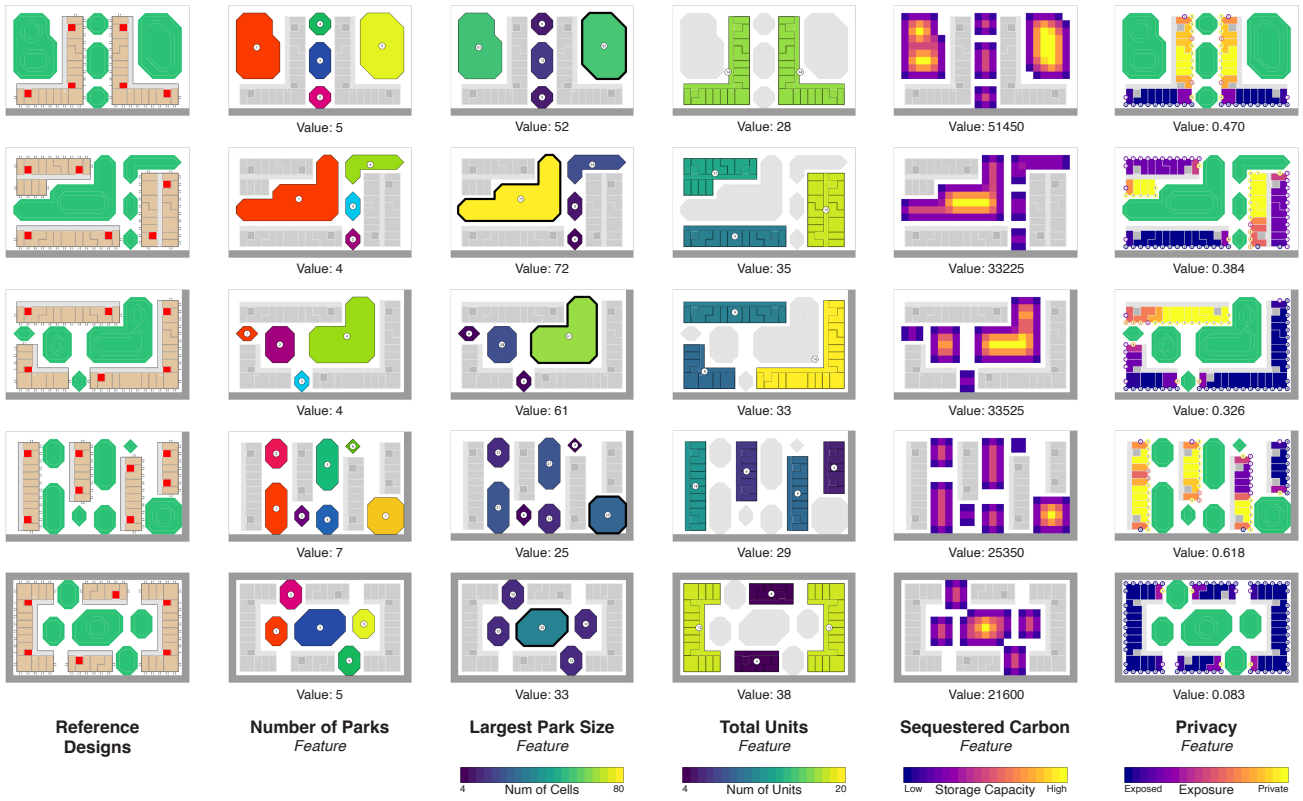

Figure 6: Features explored with MAP-Elites. Layouts which span these features are generated to form a dataset for training.

4 EXPERIMENTS

4.1 Setup

We test our system in a real-world design scenario: the design of apartment complex layouts for prefabricated housing. As part of an applied research collaboration with the modular construction company FactoryOS[2], we derived our modules from their real-world catalog of prefabricated apartment units and worked together to test the WFC algorithm for early stage design.

Adjacency rules for our WFC algorithm are derived from a small set of manually created reference designs (see Appendix B). Each generated site layout consists of a 25x15 grid, totaling 375 tiles. These tiles represent various elements: livable building component modules, utility elements like corridors and cores, more or less intensive landscaping such as trees or lawn, and unused spaces and streets. Site borders are fixed, surrounded by street or landscaping.

Sites are evaluated on five metrics: number of parks, largest park size, total units, sequestered carbon, and privacy, each illustrated in Figure 6. A site's performance is gauged by the proportion of non-empty tiles. Each site is labeled with a text prompt that mirrors these features, divided into low, medium, and high values, for a total of 243 (3^5) possible text labels. For clarity we will refer to these metrics as 'features' and an instance of these features as an attribute (high privacy vs. low privacy).

[2]https://factoryos.com/

4.2 Results

Experiment Objectives and Methodology. Experiments are designed to evaluate our system, a language model fine-tuned on a synthetic dataset, in generating designs that are then refined to meet specific constraints and criteria. We focus on two key aspects:

(1) **Validity:** Does our system reliably produce valid designs that can be transformed into complete layouts by WFC?
(2) **Fidelity:** How well do designs align with the given prompts?

Where a layout is 'complete' if it is filled with a set of tiles that obey all adjacency constraints, and a design is considered to 'align' with the prompt if the attribute value is in the ranges defined during training for each text prompt (see Figure 7 for demarcations). We evaluate our model with the following exhaustive approach:

- We prompt the model to produce 100 designs for every combination of prompts, amounting to 243 prompts.
- Validity is measured by the WFC solver's ability to generate a complete layout from each design.
- Fidelity is measured for each valid layout, with achieving fidelity when the attributes match those specified in the prompt, each attribute evaluated separately.

We investigate the impact of employing a QD approach in the generation of the synthetic dataset. Two datasets are used to produce models, one generated with MAP-Elites and the other by sampling WFC, each with a dataset contains a total of 50,000 designs each.

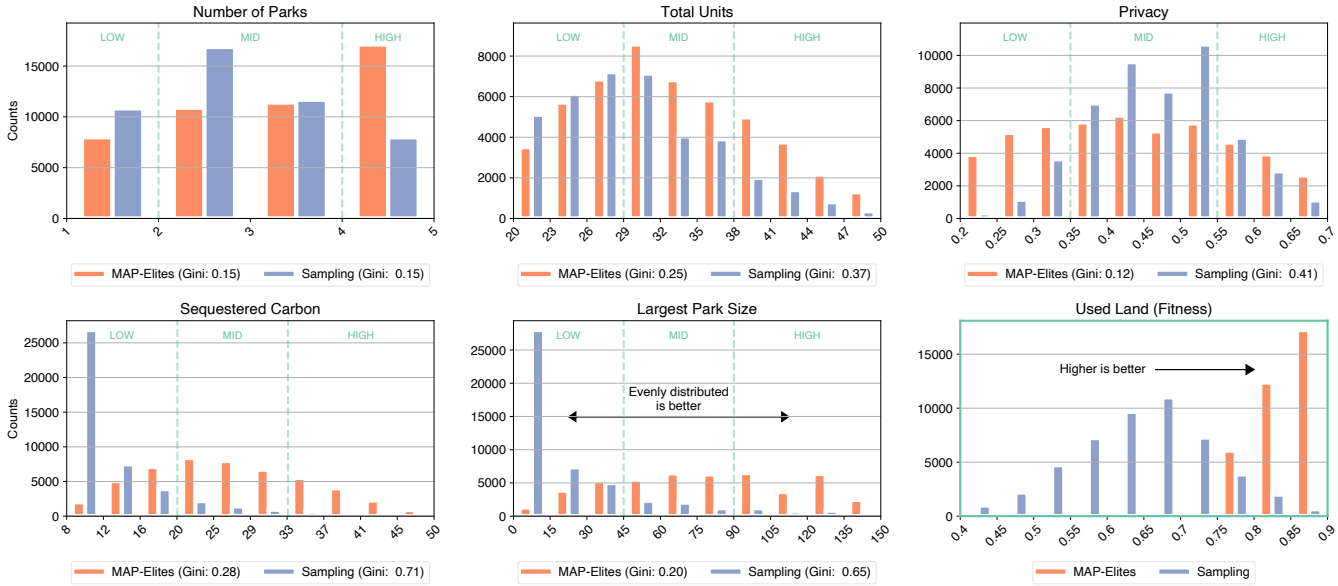

Figure 7: Distribution of feature and performance values of in datasets of designs generated with MAP-Elites and Sampling. Gini coefficient of number of samples in each bin is provided to aid interpretation of distributions. Demarcation of the qualitative labels used to train the model (e.g. low, mid, high number of units) show in green.

Comparative Analysis of Datasets. It is informative to first examine the differences in the datasets generated by sampling and by MAP-Elites. Analyzing the composition of these datasets provides a clearer understanding of the differences in the resulting models.

A key aspect of producing expressive models is ensuring a diverse range of features in the dataset. Ideally, this would manifest as a uniform distribution across all features. While a completely filled MAP-Elites archive would produce this ideal scenario, in practice there are inherent trade-offs in features, and not every combination can be produced, so creating some imbalance in unavoidable.

The distribution of feature values in the designs of each dataset is shown in Figure 7. To underline the difference in uniformity, we also calculate the Gini coefficient[3], a measure of inequality, of the number of samples in each bin.

This analysis reveals that MAP-Elites produces a far more uniformly distributed dataset compared with sampling. More than half of all samples generated by sampling WFC are in the lower tenth of sequestered carbon and large parks – randomly generated designs rarely yield large parks, which are crucial for substantial carbon sequestration. Random sampling simply cannot reliably cover the extremes of some feature distributions.

In addition we examine the distribution of performance values (Figure 7, bottom right). The datasets generated by sampling alone tend to follow a normal distribution around a low mean. In contrast, MAP-Elites actively seeks out high-performing designs. This distinction underscores the effectiveness of targeted search methods like MAP-Elites in creating datasets that not only span a broad feature range but also include high-performance design options, which are less likely to emerge through random generation.

[3]A Gini coefficient of 0 indicates perfect uniformity, while 1 indicates all samples concentrated in a single bin

Model Performance. The performance of each model, including the differences between them, is shown in Figure 8. The model trained with MAP-Elites synthesized dataset demonstrates a higher level of fidelity to the design prompts across nearly all categories. Though the category of 'total units' shows comparatively weaker performance, this can be attributed to the model's limited control over this aspect; while it can outline the building design, the actual generation of walls—and consequently the number of units—is determined by the WFC solver and randomness of the seed.

The model trained on the dataset generated by WFC exhibits uneven performance, mirroring the inconsistencies in its training dataset. The model struggles to generate designs with high carbon sequestration, large parks, or low privacy solutions, all of which are underrepresented in the sampled dataset. For attributes with abundant data, such as low carbon sequestration or number of parks, the model performs well. That the sampled dataset is lower-performing, with a lot of empty tiles, translates into fewer and smaller parks, and fewer units. This alone may be enough to bias the generation toward these attributes, regardless of the prompt.

The validity of designs generated by the WFC-trained model is lower across all categories, particularly in the 'high' level categories where the fidelity is also lacking. This trend can be attributed to the model's limited exposure to the cross-attention signal of rarer prompts in the WFC dataset, leading to challenges in handling less predictable inputs and consequently producing invalid designs.

The results underscore that the caliber and variety of the training data are key to successful model training. In particular this emphasizes the superiority of QD methods in creating rich and varied datasets, proving their effectiveness for sophisticated, real-world design problems where random sampling is not sufficient.

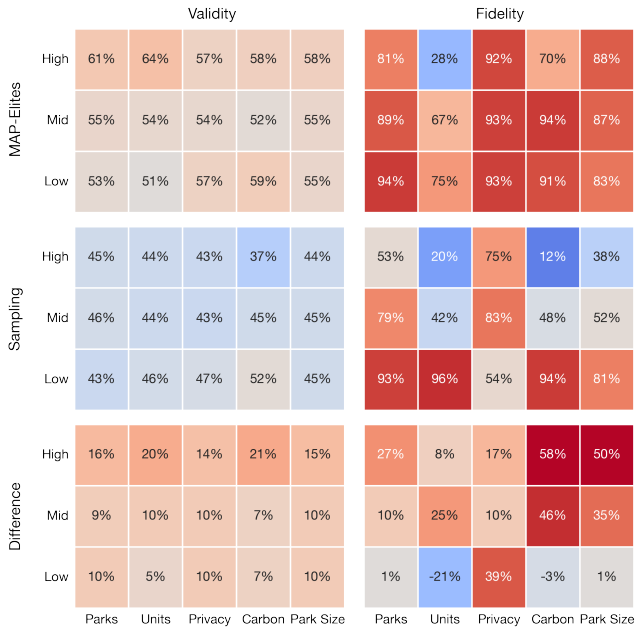

Figure 8: Model performance when trained on a MAP-Elites synthesized dataset vs. one obtained by sampling. Each cell represents the mean of a single prompt (e.g. "High number of parks") in combination with every other prompt (varied levels of units, privacy, carbon, park size). *Validity*: how often a design with this prompted feature generates a valid design. *Fidelity*: how many valid solutions follow the prompt.

5 DISCUSSION

This work introduces a novel approach to generative design, addressing the challenges of data availability, ease of use, and constraint compliance. Our method combines optimization techniques, constraint satisfaction mechanisms, and the generative capabilities of language models to remedy stubborn difficulties intrinsic to GD.

Building on existing generative design methods, our approach transforms their main weakness—the overwhelming volume of results—into a key advantage. Instead of requiring users to sift through thousands of generative design outcomes, these results become raw material to train a model to help them explore the possibilities of design. This integration allows users direct access to the exploratory benefits of evolutionary AI and the precision of constraint-satisfying symbolic AI, all through the user-friendly interface of a generative AI language model.

Our current system was built on simple tile representations, and while many layout problems in architecture can be encoded in this way, it is an obvious limitation to the technique's versatility. Alternative tokenization schemes would enable the generation of different geometries, and many such approaches are already gaining traction for manufacturing design [22, 48, 49]

The conditioning of the model on features is currently based on linear ranges of user-defined features; however, future implementations could utilize non-linear regions or integrate more descriptive natural language labels for more intuitive exploration. Approaches like Quality-Diversity with AI Feedback [2], especially combined with multimodal models which could automatically label site plans with more qualitative attributes, could further enhance the system's capability for generating intuitive and meaningful design features.

Although not explicitly evaluated in this paper, the system is designed to be interactive. Users can modify specific areas of a site layout according to their prompts, enabling high-level exploration and alteration of site plans Such prompt-guided changes can act as high-level mutation operators, as shown in MarioGPT [42], offering a novel avenue for interactive and dynamic design modification.

Beyond the specifics of the presented system, this work represents a broader approach for applying generative models in engineering and architecture. This approach rests on three pillars: diversity-based optimization for generating high-quality datasets, the use of large models for generation and interaction, and constraint satisfaction algorithms that take the final step in generation to ensuring the valid designs. By weaving a generative model into the fabric of the design process, we mitigate the need for extensive post-hoc analysis typically associated with generative design. Instead, we pave a path for purposeful exploration, allowing for both controlled directives and serendipitous design outcomes.

REFERENCES

[1] Alberto Alvarez, Steve Dahlskog, Jose Font, and Julian Togelius. 2019. Empowering quality diversity in dungeon design with interactive constrained map-elites. In *2019 IEEE Conference on Games (CoG)*. IEEE, 1–8.

[2] Herbie Bradley, Andrew Dai, Hannah Teufel, Jenny Zhang, Koen Oostermeijer, Marco Bellagente, Jeff Clune, Kenneth Stanley, Grégory Schott, and Joel Lehman. 2023. Quality-Diversity through AI Feedback. *arXiv preprint arXiv:2310.13032* (2023).

[3] Erin Bradner, Francesco Iorio, Mark Davis, et al. 2014. Parameters tell the design story: ideation and abstraction in design optimization. In *Proceedings of the symposium on simulation for architecture & urban design*, Vol. 26. Citeseer, 1–8.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[5] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature* 521, 7553 (2015), 503–507.

[6] Antoine Cully and Yiannis Demiris. 2017. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation* 22, 2 (2017), 245–259.

[7] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378* (2023).

[8] Maxence Faldor, Félix Chalumeau, Manon Flageat, and Antoine Cully. 2023. MAP-Elites with Descriptor-Conditioned Gradients and Archive Distillation into a Single Policy. *arXiv preprint arXiv:2303.03832* (2023).

[9] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2017. Aerodynamic design exploration through surrogate-assisted illumination. In *18th AIAA/ISSMO multidisciplinary analysis and optimization conference*. 3330.

[10] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2018. Data-efficient design exploration through surrogate-assisted illumination. *Evolutionary computation* 26, 3 (2018), 381–410.

[11] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2019. Are quality diversity algorithms better at generating stepping stones than objective-based search?. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 115–116.

[12] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2020. Discovering representations for black-box optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 103–111.

[13] Adam Gaier, James Stoddart, Lorenzo Villaggi, and Peter J Bentley. 2022. T-DominO: Exploring Multiple Criteria with Quality-Diversity and the Tournament Dominance Objective. In *International Conference on Parallel Problem Solving from Nature*. Springer, 263–277.

[14] Theodoros Galanos, Antonios Liapis, Georgios N Yannakakis, and Reinhard Koenig. 2021. ARCH-Elites: Quality-diversity for urban design. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 313–314.

[15] GitHub. 2021. GitHub Copilot: Your AI pair programmer. https://github.com/features/copilot. Accessed: 2024-01-30.

[16] Miguel González-Duque, Rasmus Berg Palm, David Ha, and Sebastian Risi. 2020. Finding game levels with the right difficulty in a few trials through intelligent trial-and-error. In *2020 IEEE Conference on Games (CoG)*. IEEE, 503–510.

[17] Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. 2019. Procedural content generation through quality diversity. In *2019 IEEE Conference on Games (CoG)*. IEEE, 1–8.

[18] Luca Grillotti and Antoine Cully. 2022. Unsupervised Behavior Discovery With Quality-Diversity Optimization. *IEEE Transactions on Evolutionary Computation* 26, 6 (2022), 1539–1552.

[19] M. Gumin. 2016. Wave Function Collapse. https://github.com/mxgmn/WaveFunctionCollapse. Accessed: March 25, 2023.

[20] Alexander Hagg, Alexander Asteroth, and Thomas Bäck. 2018. Prototype discovery using quality-diversity. In *International Conference on Parallel Problem Solving from Nature*. Springer, 500–511.

[21] Alexander Hagg, Dominik Wilde, Alexander Asteroth, and Thomas Bäck. 2020. Designing air flow with surrogate-assisted phenotypic niching. In *International Conference on Parallel Problem Solving from Nature*. Springer, 140–153.

[22] Pradeep Kumar Jayaraman, Joseph George Lambourne, Nishkrit Desai, Karl Willis, Aditya Sanghi, and Nigel JW Morris. 2022. SolidGen: An Autoregressive Model for Direct B-rep Synthesis. *Transactions on Machine Learning Research* (2022).

[23] Isaac Karth and Adam M Smith. 2017. WaveFunctionCollapse is constraint solving in the wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*. 1–10.

[24] N. Kaylor, L. Villaggi, and D. Zhao. 2023. From Prototype to Platform: Delivering New Design Capabilities on Autodesk Forma. In *Autodesk University*. Las Vegas.

[25] Paul Kent, Adam Gaier, Jean-Baptiste Mouret, and Juergen Branke. 2023. BOP-Elites, a Bayesian Optimisation Approach to Quality Diversity Search with Black-Box descriptor functions. *arXiv preprint arXiv:2307.09326* (2023).

[26] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[27] Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. 2023. Evolution through large models. In *Handbook of Evolutionary Machine Learning*. Springer, 331–366.

[28] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv:1910.13461 [cs.CL]

[29] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. StarCoder: may the source be with you! *arXiv preprint arXiv:2305.06161* (2023).

[30] P. Merrell. 2007. Example-Based Model Synthesis. In *Symposium on Interactive 3D Graphics (i3D)*.

[31] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).

[32] Danil Nagy, Damon Lau, John Locke, Jim Stoddart, Lorenzo Villaggi, Ray Wang, Dale Zhao, and David Benjamin. 2017. Project Discover: An application of generative design for architectural space planning. In *Proceedings of the Symposium on Simulation for Architecture and Urban Design*. Society for Computer Simulation International, 7.

[33] D Nagy, L Villaggi, and D Benjamin. 2018. Generative urban design: Integration of financial and energy design goals in a generative design workflow for residential neighborhood layout. In *Symposium on Simulation for Architecture and Urban Design*.

[34] Danil Nagy, Lorenzo Villaggi, Dale Zhao, and David Benjamin. 2017. Beyond heuristics: a novel design space model for generative space planning in architecture. (2017).

[35] Giuseppe Paolo, Alban Laflaquiere, Alexandre Coninx, and Stephane Doncieux. 2020. Unsupervised learning and exploration of reachable outcome space. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2379–2385.

[36] B. C. Paulson. 1976. Designing to Reduce Construction Costs. *Journal of the Construction Division* 102, 4 (1976), 587–592.

[37] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI* 3 (2016), 40.

[38] Nemanja Rakicevic, Antoine Cully, and Petar Kormushev. 2021. Policy manifold search: Exploring the manifold hypothesis for diversity-based neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 901–909.

[39] Franz Rothlauf and Franz Rothlauf. 2006. *Representations for genetic and evolutionary algorithms*. Springer.

[40] The Construction Users Roundtable. 2004. *Collaboration, Integrated Information, and the Project Lifecycle in Building Design, Construction and Operation*. Technical Report WP-1202. Introduction of the "MacLeamy Curve".

[41] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108 [cs.CL]

[42] S Sudhakaran, M González-Duque, C Glanois, M Freiberger, E Najarro, and S Risi. [n. d.]. MarioGPT: open-ended text2level generation through large language models (2023). *arXiv preprint arxiv:2302.05981* ([n. d.]).

[43] Shyam Sudhakaran, Miguel González-Duque, Claire Glanois, Matthias Freiberger, Elias Najarro, and Sebastian Risi. 2023. Prompt-guided level generation. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. 179–182.

[44] Graham Todd, Sam Earle, Muhammad Umair Nasir, Michael Cerny Green, and Julian Togelius. 2023. Level Generation Through Large Language Models. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*. 1–8.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. arXiv:1706.03762 [cs.CL]

[46] L. Villaggi, D. Nagy, and J. Stoddart. 2018. Generative Design for Architectural Space Planning. In *Autodesk University*. Las Vegas.

[47] Lorenzo Villaggi, James Stoddart, and Adam Gaier. 2022. Harnessing Game-Inspired Content Creation for Intuitive Generative Design and Optimization. In *Design Modelling Symposium Berlin*. Springer, 149–160.

[48] Xiang Xu, Pradeep Kumar Jayaraman, Joseph G Lambourne, Karl DD Willis, and Yasutaka Furukawa. 2023. Hierarchical neural coding for controllable cad model generation. *arXiv preprint arXiv:2307.00149* (2023).

[49] Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. 2022. SkexGen: Autoregressive generation of CAD construction sequences with disentangled codebooks. *arXiv preprint arXiv:2207.04632* (2022).

[50] Yulun Zhang, Matthew C Fontaine, Amy K Hoover, and Stefanos Nikolaidis. 2022. Deep surrogate assisted map-elites for automated hearthstone deckbuilding. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 158–167.

APPENDIX

A SOURCE CODE

Source code can be found linked from the project's permanent home: https://tilegpt.github.io

B WFC BASE DESIGNS

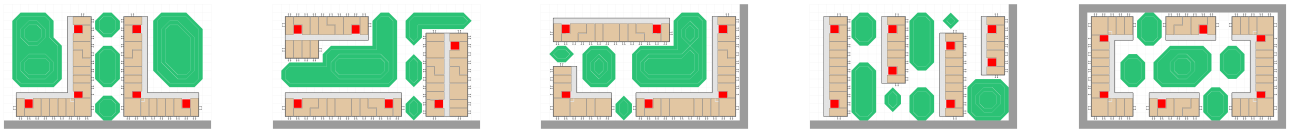

Table 1: Base designs used by Wave Function Collapse. WFC learns a set of adjacency rules from a small set of examples. In this work, which generated 50,000 different samples for two different datasets, the generator was based off the adjacency rules learned from just the above five layouts.

C FULL TILE TO TILE TO CHARACTER ENCODING

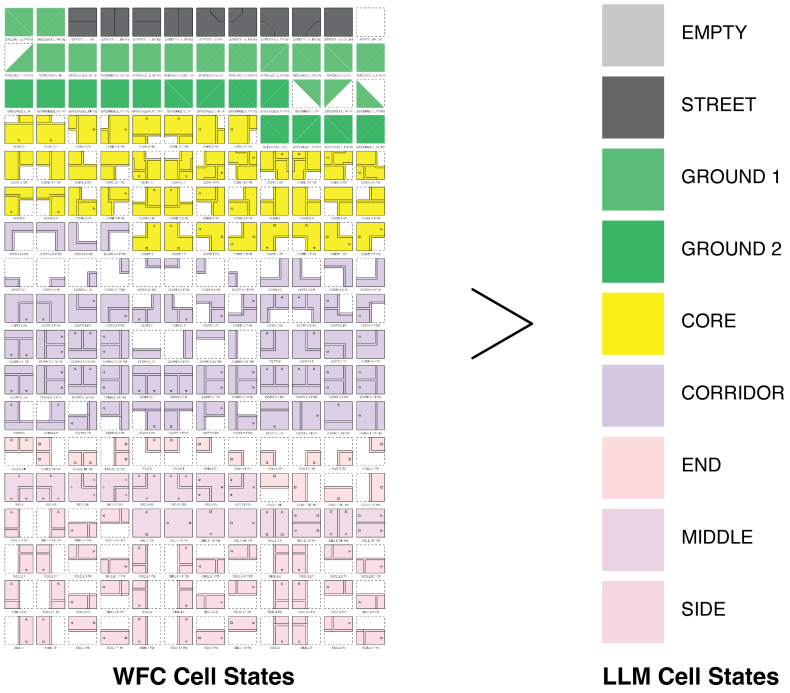

Figure 9: Complete set of possible tile states used in WFC. 216 tile states in total, including rotations and reflections, are derived from the WFC samples. When fixed during evolution, or fed to the language mode as tokens, these states are simplified to their functional category on the right.