Diceplay: A Modular Canvas for Physical Image Composition

MILIN KODNONGBUA*, University of Washington, USA
ZIHAN JACK ZHANG*, SHISHI XIAO*, VIVIAN LI, and HEATHER ROBERTSON, Brown University, USA
RULIN CHEN, Beijing Normal-Hong Kong Baptist University, China and BNBU Centre for Computational Culture and Heritage, China
DAVID LAIDLAW and ADRIANA SCHULZ, Brown University, USA

Fig. 1. **Diceplay**, a modular physical canvas for abstract visual composition. Diverse abstract designs automatically generated by our optimization framework are physically realized using a low-power, LED-enabled dice display, which can be reconfigured through changes in dice orientation and dynamically updated via programmable color.

We present Diceplay, a modular physical display for abstract visual composition built from a grid of identical dice. Each die has six faces with distinct geometric primitives, and images emerge through the placement and orientation of the dice. While this medium enables reusable and reconfigurable physical imagery, it poses a challenging design problem: images must be expressed through discrete, extremely low-resolution abstractions, making manual authoring difficult. To address this challenge, we introduce a computational design system that automatically generates Diceplay configurations from text prompts. Our key technical contribution is a grammar-based formulation that relaxes this discrete design space into a smooth optimization landscape, enabling gradient-based optimization using score distillation sampling. We show that our approach consistently produces meaningful abstractions for this medium, whereas state-of-the-art smoothing techniques fail in this extremely challenging regime. We demonstrate our method across a range of prompts and fabricated examples, showing how computationally generated abstractions can be realized as physical visual artifacts.

CCS Concepts: • **Computing methodologies** → **Shape modeling**.

Additional Key Words and Phrases: optimization, fabrication, shape grammar

ACM Reference Format:
Milin Kodnongbua, Zihan Jack Zhang, Shishi Xiao, Vivian Li, Heather Robertson, Rulin Chen, David Laidlaw, and Adriana Schulz. 2026. Diceplay: A Modular Canvas for Physical Image Composition. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3799902.3811206

*Equal Contribution.

Authors' Contact Information: Milin Kodnongbua, milink@cs.washington.edu, University of Washington, Seattle, USA; Zihan Jack Zhang, zihan_zhang4@brown.edu; Shishi Xiao, shishi_xiao@brown.edu; Vivian Li, vivian_li1@brown.edu; Heather Robertson, heather_robertson@brown.edu, Brown University, Providence, USA; Rulin Chen, rulinchen@bnbu.edu.cn, Beijing Normal-Hong Kong Baptist University, Zhuhai, China and BNBU Centre for Computational Culture and Heritage, Zhuhai, China; David Laidlaw, david_laidlaw@brown.edu; Adriana Schulz, adriana_schulz@brown.edu, Brown University, Providence, USA.

SIGGRAPH Conference Papers '26, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2554-8/2026/07
https://doi.org/10.1145/3799902.3811206

1 Introduction

Across art and design, creators often rely on abstraction to communicate ideas, evoke meaning, and invite interpretation without relying on photorealistic detail. Many visual languages deliberately operate with constrained vocabularies, where limited palettes of simple geometric primitives accentuate form, composition, and identity.

One way artists have worked within these abstract visual languages is through modular composition, in which images emerge from arrangements of repeated visual primitives on a grid. This strategy appears in geometric art traditions such as Bauhaus (see 2). Although each unit is simple, their composition supports a wide range of outcomes, highlighting how expressiveness can arise from severe constraints.

Fig. 2. **Bauhaus Style.** Image by Vectorportal.com.

In this work, we propose a modular physical canvas inspired by abstract, grid-based visual languages. Because such compositions are built from a small set of elements whose appearance depends on their orientation, we realize the canvas as a collection of identical dice. We call this system *Diceplay*, a display made of dice (Figure 3). Each die is identical, and each of its six faces contains a distinct geometric pattern inspired by Bauhaus-style abstract art. While these abstract primitives are drawn from existing artistic explorations of 2D modular design [Online-Schule für Gestaltung 2024], to the best of our knowledge, Diceplay is the first system to realize this visual language as a physical display composed of dice.

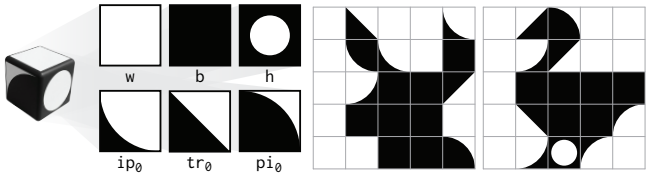


Fig. 3. **Diceplay Tiles.** (Left) The six faces of a die, each with a tile pattern, which we denote using lowercase letters. Tiles `ip`, `tr`, and `pi`[1] admit four rotations, which we denote using subscripts. There are 15 tiles in total. (Right) Example Diceplay configurations created by our method for text prompts "*deer*" and "*duck*".

Each die can be fabricated using low-cost 3D printing and placed into a grid-like frame that defines the display, which we keep intentionally small to allow easy manual reconfiguration (Figure 1). The frame contains programmable LEDs aligned with each cell, while the dice themselves remain passive elements that can be flipped and reinserted to change the visible pattern. This design allows each cell to be updated both through reorienting the dice and changing the LED colors. Together, these properties enable the same set of components to be reused to create different compositions over time, supporting experimentation as well as applications such as educational activities and ambient or glanceable visual displays.

Despite the simplicity and flexibility of the Diceplay medium, designing effective imagery for it is challenging. Images are formed at extremely low resolution using a highly restricted visual vocabulary, requiring recognizable content to be conveyed through carefully abstracted features. Achieving such abstractions is difficult for users, particularly when working directly with individual dice: small local changes, such as flipping a single die, can lead to abrupt and global changes in appearance.

[1]The sector pattern looks like a pie, and is thus named `pi`, the inverse sector is thus named by inverting the letters – `ip`

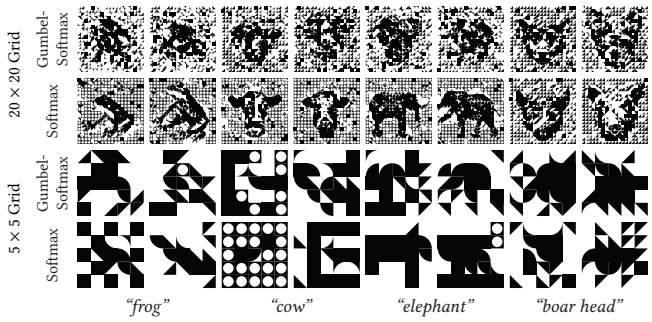


Fig. 4. **Curse of Low-Resolution.** Failure cases of Softmax and Gumbel-Softmax relaxations when applied to Diceplay optimization with an SDS loss. Due to the extremely coarse grid resolution, both methods become trapped in noisy local minima and fail to produce recognizable shapes.

To support users in designing for this display, we propose a computational design system that automatically generates abstractions from high-level instructions. We focus on text prompts, which are well suited to specifying abstract concepts, and optimize designs using score distillation sampling (SDS) loss, which has been shown to work well for matching images to text prompts. A fundamental challenge, however, is that the Diceplay design space is purely discrete, making it poorly suited to standard gradient-based optimization. While prior work has successfully combined SDS with continuous relaxations such as Gumbel-Softmax [Binninger and Sorkine-Hornung 2024], these approaches perform poorly in our setting due to the extremely low resolution (Figure 4).

We address this challenge with two key contributions. First, we identify sets of visually similar dice faces and introduce continuous parameters that interpolate between the boundaries of patterns within each set, rather than relying on alpha blending (Figure 7 (a) and (c)). Second, we represent the design space as a shape grammar whose production rules permit only continuous transitions within each cell and coordinated transitions across neighboring cells. This construction shapes the optimization landscape to more closely resemble that of vector graphics, encouraging coherent abstractions. Finally, we employ Stochastic Rewrite Descent (SRD) [Kodnongbua et al. 2025], a gradient-based optimization method for mixed continuous–discrete grammars, to search for Diceplay configurations that best match the input text prompt. We use this optimization to generate black-and-white Diceplay configurations from text prompts and complete the end-to-end system with a simple colorization scheme and an interactive editing interface grounded in the same grammar abstraction, along with a low-cost fabrication method using 3D printing and programmable LEDs (total cost is roughly $50).

We demonstrate the capabilities of Diceplay through a set of fabricated examples that illustrate the range of designs and applications supported by the system. We evaluate our grammar-based optimization across diverse user inputs, showing that it reliably produces recognizable and visually compelling results despite the highly discrete and constrained design space. Comparisons against standard smoothing strategies used in prior work reveal that these methods break down in Diceplay's extremely coarse regime, while

our grammar-based formulation enables effective search and optimization in practice.

2 Related Work

Fabricable Abstractions. Substantial work in computational design and fabrication represents complex shapes using abstract or constrained representations. These approaches transform designs into structured forms that better align with fabrication processes, enabling faster production, or easier assembly. Examples include systems that optimize designs to better match the constraints of a specific fabrication process, such as 3D printing [Chen et al. 2025, 2022], laser cutting [Chen et al. 2013; Cignoni et al. 2014; Hildebrand et al. 2012], 2D bending [Miguel et al. 2016], or CNC milling [Fanni et al. 2018; Muntoni et al. 2018]. Others approximate a target shape using a set of predefined building blocks, such as LEGO bricks [Ge et al. 2024b,a; Luo et al. 2015; Xu et al. 2021; Zhou et al. 2023], Zometool construction set [Zimmer and Kobbelt 2014; Zimmer et al. 2014], SL blocks [Shih 2016], and interlocking voxels [Zhang and Balkcom 2016; Zhang et al. 2021]. Please refer to the comprehensive survey [Wang et al. 2021] for a more detailed discussion of these techniques. While these methods constrain designs to match fabrication requirements, they typically aim to approximate a target shape as closely as possible, rather than to reinterpret a high-level idea within a deliberately abstract, fabricable representation. Closest to our work is a recent wire art system that generates a fabricable continuous wire shape to convey a visual concept [Tojo et al. 2024]. Diceplay builds on this general idea of abstract fabrication, but targets an abstraction problem with stricter constraints.

Commercial toys that encourage children to build abstract shapes from geometric primitives (Figure 5) also inspire our work. We complement our digital display with 3D-printed dice that children can use to recreate generated designs. To our knowledge, we are the first to propose dice as primitives for abstract shape representation and the first computational approach for generating such designs.

Fig. 5. **Toys.** Examples of commercial abstract shape toys [LiKee nd; LovesTown nd; Pozale nd]. Each toy uses a distinct set of geometric primitives; for each, we show one example card alongside the corresponding assembled shape.

Image Abstractions. Image abstraction methods represent images using simplified primitives while preserving visual or semantic intent. Image-driven approaches focus on maintaining perceptual similarity to the input by optimizing continuous vector representations under differentiable rendering [Berio et al. 2025; Li et al. 2020; Ma et al. 2022], while vision-language methods use CLIP for semantic alignment despite visual simplification [Frans et al. 2022; Mathur et al. 2025; Vinker et al. 2022]. Recent approaches further incorporate diffusion-based priors via score distillation [Jain et al. 2023; Wang et al. 2025; Xing et al. 2024]. Across these methods, abstraction is performed in a high-resolution, continuous design space where primitives can be smoothly adjusted. Even methods exploring discrete primitives typically realize abstraction through high-resolution continuous representations [Binninger and Sorkine-Hornung 2024; Misra et al. 2025]. In contrast, Diceplay operates on a strictly discrete, low-resolution grid where abstraction emerges from coarse combinatorial structure.

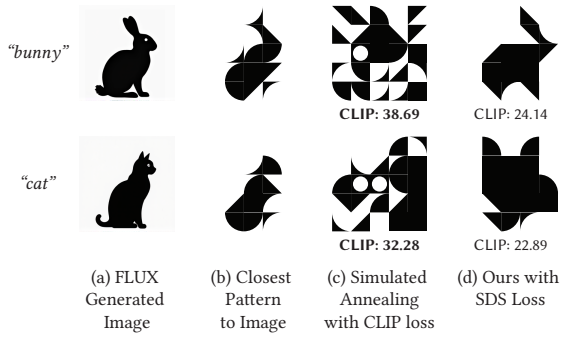


Fig. 6. **Choice of Objective Functions.** Given prompts "*bunny*" and "*cat*", matching the tiles as close as possible (b) to the FLUX.dev-1 generated images (a) produces results which are not recognizable from the prompts. Using simulated annealing with CLIP loss (c) produces designs with higher CLIP similarity than our method (d), yet these designs are noisy and suboptimal, showing that CLIP is unreliable in our problem setting.

Continuous relaxations. Continuous relaxations enable gradient-based optimization over discrete design spaces. For example, Gumbel-based relaxations have been used for optimizing discrete color palettes in embroidery [Binninger and Sorkine-Hornung 2024]. However, these techniques require sufficient spatial resolution; in Diceplay, each cell represents a large semantic unit, so standard relaxations fail to provide useful gradients. This motivates our grammar-based approach, which introduces smoothness at the level of discrete transitions rather than cell-level relaxations.

Our work builds on Design for Descent [Kodnongbua et al. 2025], which introduces gradient-based optimization for mixed-continuous-discrete shape grammars. Since Diceplay's space is fully discrete, we define an augmented grammar with smooth parameters combined with a discreteness-enforcing cost. This encourages solutions to remain close to valid discrete configurations and effectively turns the grammar into a relaxation suitable for gradient-based optimization.

3 Computational Design of Diceplay

Diceplay *configurations* are formed by arranging dice in a grid of N rows and M columns. We also define *cells* as the fixed spatial locations within this grid, *tiles* as the chosen geometric patterns displayed on the dice, *boundaries* as the boundary between black and white regions, and *cell edges* as the four edges of a cell. Each die displays one of 15 canonical geometric tiles (Figure 3), which we denote using lowercase letters:

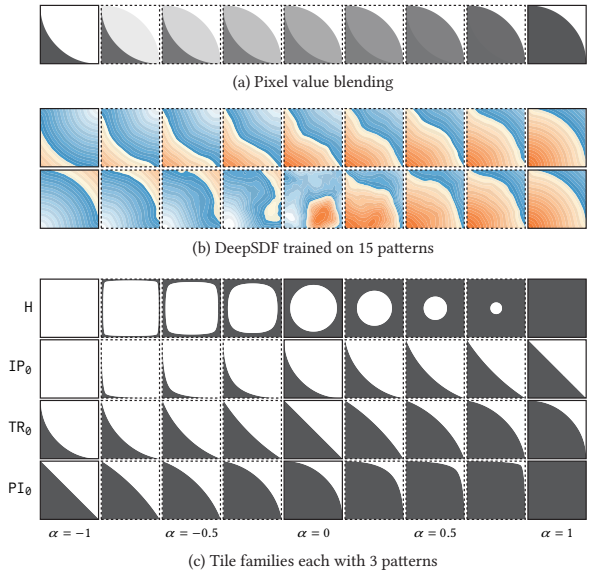
$$T \coloneqq \{\mathtt{w}, \mathtt{b}, \mathtt{h}, \mathtt{ip}_\mathtt{k}, \mathtt{tr}_\mathtt{k}, \mathtt{pi}_\mathtt{k}\},\ \mathtt{k} \in \{0, 1, 2, 3\}$$

where k denotes the rotations. Let $\mathcal{A}$ denote the $N \times M$ grid of cells, a Diceplay configuration is a mapping $C : \mathcal{A} \to T$ that assigns each cell a tile. Unless otherwise stated, we use $N = M = 5$. Given a text prompt, we seek a configuration C that minimizes the semantic distance to the user-provided prompt.

The problem presents two fundamental challenges: (1) *Objective is difficult to define.* The natural approach would be to generate a reference image from the prompt and minimize a pixel-wise loss. However, a configuration C that minimizes pixel error may fail to capture salient features that make an abstraction recognizable (Figure 6 (b)). An alternative is to optimize for CLIP similarity with the text embedding, but CLIP scores are often misaligned with human perception of abstraction quality (Figure 6 (c)). We also refer to Figure 15 where we plot CLIP scores vs. the number of votes from our user study (Sec. 5). Finetuning existing image generation models is also intractable due to the lack of training images composed with our dice tiles. (2) *Discrete search space.* Optimizing C is a combinatorial optimization problem. Even for a modest grid size of $N = M = 5$, the search space contains $|T|^{NM} = 15^{25} \approx 2.5 \times 10^{29}$ possible configurations. Furthermore, typical solvers for combinatorial optimization are not applicable here due to the difficulty of finding suitable objectives.

To address these challenges, we propose to use Score Distillation Sampling (SDS) loss [Poole et al. 2022], which provides gradients towards the provided text prompt, and we relax the Diceplay design space to a continuous counterpart which allows for efficient gradient-based optimization. Prior work such as SD-πXL [Binninger and Sorkine-Hornung 2024] employ soft-blending through the Gumbel-softmax trick. However, we show that these approaches fail at Diceplay's extremely low resolution (Figure 4), where each cell represents a semantically significant visual unit and soft blending of tile boundaries produces incoherent interpolations (Figure 7 (a)).

Our key insight is that effective interpolation must operate on tile boundaries rather than pixel values. One might attempt this by training a DeepSDF [Park et al. 2019] model to interpolate between the signed distance fields of all 15 tiles. As shown in Figure 7 (b), this produces more geometrical interpolations compared to pixel value blending; however, this approach lacks global coherence: each grid cell interpolates independently, ignoring the boundaries it shares with cell neighbors, and the interpolation is not perfectly smooth. Inspired by recent work on designing shape grammars for gradient-based optimization [Kodnongbua et al. 2025], we address both problems, incoherent interpolation and independent cells, by constructing a parametric shape grammar. Our grammar restricts which tiles may be interpolated at each cell and encodes dependencies between adjacent cells through its production rules, enabling smooth and locally changing traversal of Diceplay configurations.



(a) Pixel value blending

(b) DeepSDF trained on 15 patterns

(c) Tile families each with 3 patterns

Fig. 7. **Tile Interpolation.** Prior methods for tile interpolation are suboptimal. (a) Prior softmax-based methods use pixel value blending which ignores the geometry of the tiles, (b) DeepSDF interpolations leverage geometry but the interpolation is not perfectly smooth for intermediate values, (c) Our method restricts the subset of tiles that are interpolated and creates smooth homotopies between tile patterns.

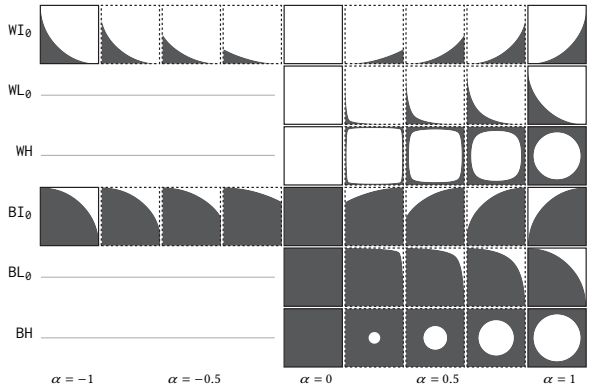


Fig. 8. **Interpolations for Transitional Tile Families.** The transitional tile families handle interpolations between tile families B and W in a boundary-aware manner.

3.1 Tile Families

We introduce the concept of *tile families*. Instead of interpolating between all tiles simultaneously as in DeepSDF, we define a set of *one-parameter families* $\mathcal{T}$ that interpolates between restricted subsets of at most three compatible tiles from T (Figure 7 (c)). Formally, each family $\tau \in \mathcal{T}$ is a mapping $\tau : \Theta_\tau \to S$, where the domain Θ_τ is $[-1, 1]$ or $[0, 1]$ depending on the family τ and S is the shape space of tiles. For a given parameter $\alpha \in \Theta_\tau$, $\tau(\alpha)$ is a render of a specific tile, and varying α produces smooth, boundary-preserving interpolations between tiles. We denote these sets of one-parameter families using uppercase letters:

$$\mathcal{T} = \{\mathtt{W}, \mathtt{B}, \mathtt{H}, \mathtt{WH}, \mathtt{BH}, \mathtt{IP}_\mathtt{k}, \mathtt{TR}_\mathtt{k}, \mathtt{PI}_\mathtt{k}, \mathtt{WL}_\mathtt{k}, \mathtt{BL}_\mathtt{k}, \mathtt{WI}_\mathtt{k}, \mathtt{BI}_\mathtt{k}\},$$

where , $\mathtt{k} \in \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}\}$ again represents the rotations. We design each family so that evaluating at $\alpha = 0$ recovers the corresponding original tile from T. That is, for each family denoted with an

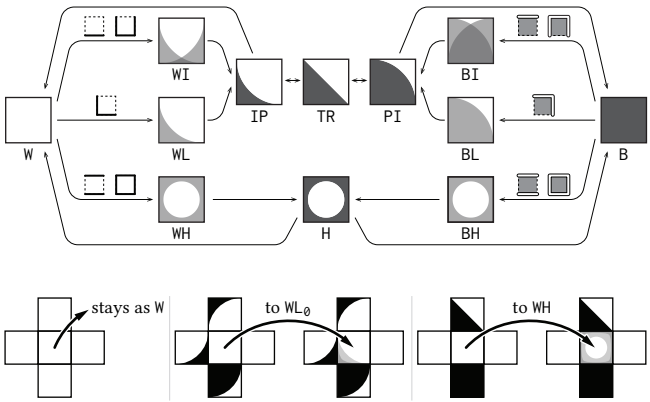


Fig. 9. **Boundary-aware Production Rules.** (Top) For families B and W, their production rules depend on whether its four neighboring tiles has black or white edges along its four sides (these edge conditions are displayed over the arrows). (Bottom) A concrete example of how a tile's four neighbors influence the production rules applicable to it.

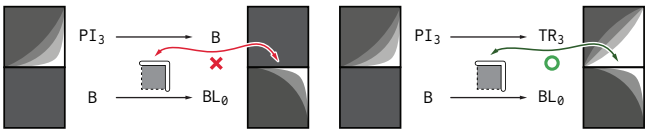


Fig. 10. **Conflicting Production Rules.** (Left) The two production rules cannot be applied simultaneously since the bottom rule is conditional on the LHS having two white edges, which is violated after the top rule (changing their shared edge from white to black). (Right) Compatible production rules can be applied simultaneously.

uppercase letter, its lowercase counterpart is recovered at zero:

$$\mathtt{TR}_\emptyset(0) = \mathtt{tr}_\emptyset, \quad \mathtt{PI}_\emptyset(0) = \mathtt{pi}_\emptyset, \quad \mathtt{H}(0) = \mathtt{h}$$

As α varies toward -1 or 1, the family smoothly interpolates toward neighboring tiles. For example, the $\mathtt{TR}_0$ family spans three original tiles:

$$\mathtt{TR}_\emptyset(-1) = \mathtt{ip}_\emptyset, \quad \mathtt{TR}_\emptyset(0) = \mathtt{tr}_\emptyset, \quad \mathtt{TR}_\emptyset(1) = \mathtt{pi}_\emptyset$$

This construction ensures that when optimization converges with α at integer boundaries, the result maps back to a valid discrete configuration in T.

3.2 Diceplay Grammar

We define the Diceplay grammar as a parametric shape grammar $\mathcal{G} = (\mathcal{T}, P, \mathcal{S}, \Theta)$. Here, $\mathcal{T}$ is the alphabet of parametric tile families, $\Theta \subset \mathbb{R}$ is the parameter domain, $\mathcal{S}$ is the space of tile patterns, and P is the set of production rules.

The production rules P govern transitions between tile families. When the parameter α of a family τ reaches a boundary value (within a fixed threshold $\delta = 0.15$), a production rule rewrites the current family to a neighboring family and resets α to a boundary value to maintain visual continuity. For example, we have the production rule with conditional expression: $\mathtt{TR}_\emptyset(\alpha) : (\alpha > 1 - \delta) \rightarrow \mathtt{PI}_\emptyset(0)$. This rule fires when α exceeds $1 - \delta$, transitioning from the $\mathtt{TR}_\emptyset$ family to the $\mathtt{PI}_\emptyset$ family. Visually, both $\mathtt{TR}_\emptyset(1)$ and $\mathtt{PI}_\emptyset(0)$ render the same tile $\mathtt{pi}_\emptyset$, so the transition is seamless.

For families $\mathtt{IP}_k$, $\mathtt{TR}_k$, $\mathtt{PI}_k$, and H, production rules depend only on the local parameter α, not on neighboring cells. This is because the "insideness" along cell edges remains constant throughout the interpolation. For instance, the left and bottom edges of $\mathtt{TR}_0$ are always considered inside (black).

Nevertheless, the solid families W and B require special treatment because valid transitions depend on adjacent cells. A white cell surrounded by white cells should remain white, whereas a white cell adjacent to a black region could transition to form a boundary. To handle this, $\mathcal{T}$ contains the transition families $\mathtt{WL}_k$, $\mathtt{BL}_k$, $\mathtt{WI}_k$, $\mathtt{BI}_k$, WH, and BH (Figure 8) that become available based on the $2^4 = 16$ edge conditions of the four neighboring cells. For example, a production rule $\mathtt{W}(\cdot) \rightarrow \mathtt{WL}_\emptyset(0)$ becomes available when the left neighbor has a black right edge and the bottom neighbor has a black top edge (Figure 9 (bottom, middle)). For black cells, the condition depends on the neighboring edges being white. All conditions are depicted in Figure 9 (top) on the arrows originating from W and B. This mechanism ensures that the grammar only permits transitions that maintain consistent boundaries across the grid.

We summarize all tile families and production rules (omitting rotations) in Figure 9 (top). In total, the alphabet $\mathcal{T}$ contains 33 tile families. Our boundary-aware production rules are designed to bias topology preserving changes, while all families are reachable through these production rules and parameter updates, we do not guarantee that all valid *configurations* are reachable. It is possible to change the topology, but it typically requires longer trajectories to reach a certain configuration than in (Gumbel-)softmax settings where logits of each tile can be directly modified. For the full grammar definition and production rules, refer to Appendix A.

3.3 Optimization

Under the grammar $\mathcal{G}$, a *relaxed* Diceplay configuration is a mapping $\tilde{\mathcal{C}} : \mathcal{A} \rightarrow \mathcal{T} \times \Theta$ which assigns to each cell a tuple (τ, α). This configuration determines the Diceplay design by rendering the tiles $\tau(\alpha) \in \mathcal{S}$ at each grid cell. We optimize $\tilde{\mathcal{C}}$ with SDS loss and Stochastic Rewrite Descent (SRD) [Kodnongbua et al. 2025]. We also introduce an L1 regularization term on the tile parameters α that is gradually annealed during the optimization, promoting sparsity and pushing parameters toward integer boundaries. To obtain the solution to the original problem, we round α's to the nearest integer and use the corresponding tile in T for each grid cell.

To make this pipeline differentiable with respect to Θ, we realize the space of tile patterns $\mathcal{S}$ using Signed Distance Fields (SDFs). Specifically, the parametric families τ are implemented as quadratic Bézier curves where the control points are interpolated as a function of α. We then analytically compute the SDFs and employ soft rasterization to generate the image (details in Appendix B).

We note that SRD is a gradient-based optimizer designed for mixed discrete-continuous shape grammars; it alternates between continuous gradient descent steps to update α and discrete rewrite steps to update τ. In the discrete step, it samples possible production rules under the current configuration, computes the "improvement"

for each rule, and applies all "improving" rewrites that are non-conflicting, resulting in much faster optimization than applying a single rule at each step. In our case, the improvements are computed using a difference in the denoising score where the continuous optimization uses the SDS gradients directly. We refer to the original paper for further details.

We now define non-conflicting production rules in our context, which is crucial for faster optimization and better boundary coherence of the shape. The key invariant is that the edges used to satisfy the constraints need to remain the same after all the rules are applied. For example, in Figure 10 (left), the top and bottom production rules are incompatible because the constraint that the bottom edge of the top cell should remain white is violated after the top cell changes to `B` where as in (right), the edge remains white. To find the subset of non-conflicting rules, we sort the rules by their "improvements" and greedily accept rules if applying them does not conflict with all the constraints so far.

4 Diceplay Fabrication

The physical Diceplay canvas comprises two main elements: a set of 25 identical cubic dice displaying the geometric design primitives shown in Figure 2 and a tray with 25 inserts, arranged in a five-by-five grid, to hold these dice in place. Both elements are easily 3D-printable – in particular, we use white PLA for negative space and translucent PETG for positive space (indicated by white and black, respectively, in our figures). This allows us to integrate color into the display simply by lining the tray with a strip of programmable RGB LEDs. Here, it is essential to note that primitives `b` and `pi`, when placed against an LED, will prevent the passing of light. Thus, each tray insert requires at minimum two LEDs placed opposite each other, and the primitives must be arranged on the dice such that primitives `b` and `pi` oppose primitives `w` and `ip`.

5 Results

We evaluate our approach on 33 text prompts, manually selected around common themes from children's abstract visual games and puzzles. All results use a fixed 5×5 Diceplay grid and are optimized on an NVIDIA A100 GPU. We note that all results and experiments should be interpreted cautiously because the optimization is stochastic and variation across runs can also influence the outcomes. Additional results for 8×8 and 10×10 grid are in the supplemental.

Text Prompt-Based Optimization. We generate Diceplay designs directly from text prompts using our grammar-based optimization, starting from a fixed initial configuration in which the nine inner cells are black and the remaining cells are white. Because the optimization is stochastic, we run each prompt nine times and manually select the best result (Figure 14); all runs are provided in the supplemental material. Figure 11 shows representative results for a subset of prompts. Despite the extremely coarse resolution of the 5×5 grid, our method produces configurations that align well with the intended semantics of the text prompts.

We implement a baseline method where we use CLIP scores with the text embedding of the prompt and use simulated annealing (SA) to find optimal configurations (details in Appendix E). We observe that the subject is recognizable for some prompts, e.g. "*dog*", "*seahorse*", "*turkey*", and "*whale*" (Figure 14 (SA + CLIP)). However, the resulting configurations are noisy and do not show coherent boundary of the subject.

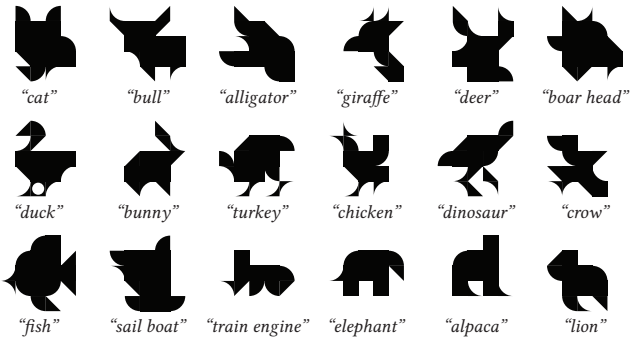


Fig. 11. **Representative Diceplay Designs.** We generate directly from text prompts using our grammar-based optimization on a 5×5 grid. Each result is selected from nine stochastic runs per prompt.

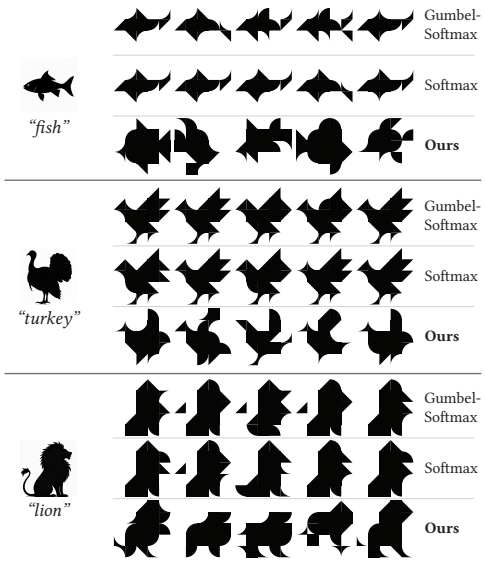


Fig. 12. **Image Initialization**. Using the same image for initialization, our method is able to explore more variations of the design conditioned on the image, while softmax-based methods do not deviate meaningfully from the initialization.

To compare against standard continuous relaxations, we implement baseline methods based on Softmax and Gumbel-Softmax. In these approaches, each Diceplay cell is parameterized by logits over the discrete set of dice patterns. We optionally add Gumbel noise, apply softmax, and render soft blending of tiles. The final discrete configuration is obtained by taking the most probable tile per cell. As shown in Figure 4, these methods work in high resolution but fail to converge to recognizable shapes and instead become trapped in noisy local minima in the 5×5 Diceplay grid. Our grammar-based formulation avoids this issue by defining a smoother search space over discrete configurations, enabling effective optimization.

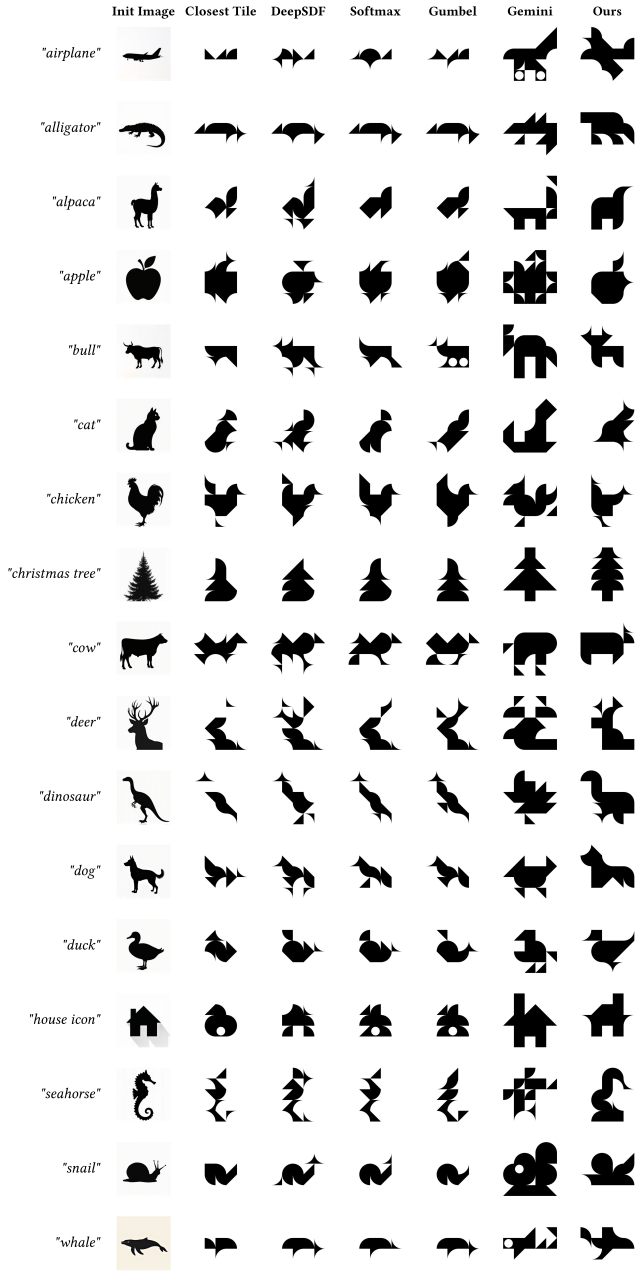


Fig. 13. **Image Initialization Comparison (Part 1).** For each method and prompt, we show the best result among nine runs; all runs are included in the supplemental material.

Image initialization. We further compare our method against alternatives with image-based initialization (Figure 12). In this setting, we first generate an image from the text prompt using the FLUX.1-dev model and optimize to match the image as initialization. Our method produces variations that are similar to the image while retaining canonical features of the objects. For example, the "*turkeys*" and the "*lions*" are facing the same way as the images but have different poses. Our method can deviate from the image for better abstractions (e.g. "*fish*"). Baseline methods based on (Gumbel-)softmax benefit from image initialization; they produce recognizable objects with a clean background. However, these approaches highly depend on the quality of the initial image because they deviate only weakly from the initialization (Figure 12).

Image initialization can also be used to guide large language model (LLM)–based approaches by prompting the models to output SVGs representing a Diceplay configuration. We experimented with several LLMs and found that Gemini (flash-3-preview) performed best. We prompt the model to explicitly reason about abstraction by first identifying salient features and their placement. As shown in Figure 13 (Gemini), this approach can succeed in some cases but remains unreliable overall.

Boundary Interpolation. We further validate our choice of interpolating the boundaries as opposed to pixel value blending by training a DeepSDF model that takes in x, y coordinates and a one-hot encoding of the 15 patterns and predicts an SDF value. We only supervise on the 15 patterns and rely on the smoothness of neural networks for interpolation. We optimize the logits and apply softmax before feeding to the DeepSDF model. The results, shown in Figure 14 (DeepSDF), are qualitatively better than the (Gumbel-)softmax baselines that use pixel value interpolation. Noticeably, we can see clear contour of the objects, for example the "*alpaca*", "*chicken*", "*duck*", and "*stanford bunny*". Results for most other prompts show recognizable objects but with noisy background. This is because the cells are optimized independently. Our method encodes dependencies across neighboring cells in the grammar, encouraging the optimization to focus on modifying the existing black regions.

Colorization. After optimizing for the black-and-white tile patterns, we perform colorization with Gemini (flash-3-preview), where the VLM is given the SVG code of the optimized tile and the corresponding PNG image and is prompted to alter only the color of each tile. Despite the model's unreliability with generating the tiles themselves (previous section), it is able to recognize the abstract features of the tile designs from our system and apply appropriate colors to the Diceplay design (Figure 14 (Ours + Color)).

User Study. We conduct a perceptual user study comparing our method against simulated annealing (SA) with CLIP scores and also Gumbel-softmax baselines, for both 5×5 and 8×8 resolutions. (an example task is shown in Figure 20 in supplemental). For each prompt, participants were asked to select three images out of 27 candidates (9 runs $\times$ 3 methods) that best represent the prompt. Across 32 participants, the selections were distributed as 73%, 18%, 9%, for our method, SA, and Gumbel-softmax, respectively. Additionally, for 5×5 resolution, 91% of responses included at least one image generated by our method among the three selected images, compared to 33% for SA and 22% for Gumbel-softmax. The detailed user voting breakdown is in Figure 16, and a selection of the most voted results for the SA baseline is shown in Figure 21 in supplemental. We further analyzed whether CLIP scores correlate with human votes via OLS regression, and we found no meaningful relationship

Fig. 14. **Text-conditioned Generation Comparison.** For each method and prompt, we show the best result among nine runs; all runs are included in the supplemental material.

between the two ($R^2 = 0.013$), indicating the CLIP similarity is not a driver for voting behavior (Figure 15).

Fabricated Results. Figure 1 shows our LED-enabled display under different dice configurations and lighting conditions. We also fabricated a smaller version of the dice and a simple frame without LEDs,

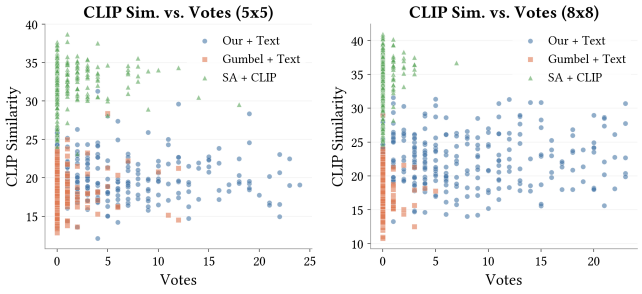


Fig. 15. **CLIP Similarity and User Preferences.** The data from our perceptual user study suggests that there is no meaningful correlation between CLIP scores and user preferences expressed in terms of the number of votes.

Fig. 16. **User Study Results.** (Left) Percentage of votes received overall for each method. (Right) Percentage of each method being selected at least once for each prompt. Note that on the right identically colored columns do not sum to 100%.

Fig. 17. **Physical dice.** Three geometric designs assembled by a 4-year-old child using the fabricated dice puzzle. Each design was created by matching dice orientations to target shape cards generated by our method.

and demonstrate its use as a geometric puzzle for children when paired with cards describing target shapes generated by our method. Figure 17 shows three designs that were successfully assembled by a 4-year-old child in 4–8 minutes (see supplemental video).

Limitations. While our method performs well in practice, it has several limitations. In some cases, when a high-quality image initialization is available, Gumbel-Softmax can produce stronger results than our approach. This occurs because our method intentionally explores beyond the image, whereas (Gumbel-)softmax remains more tightly coupled to it. This effect is particularly noticeable when the target image occupies only a portion of the grid, as our method tends to utilize the available space more fully. As shown in Figure 13, (Gumbel-)softmax yields visually compelling results for examples such as the "*snail*" and "*seahorse*". Our outputs in these cases often reflect less common but still reasonable abstractions of these shapes, rather than closely matching their canonical appearances.

Another limitation is that our method is less effective for shapes that are already highly geometric and admit an obvious tiling, for example "*house*" and "*christmas tree*". In these cases, our method often fails to converge to the expected structure. We observe similar behavior using (Gumbel-)softmax baselines. For such examples, direct LLM-based prompting can produce stronger results, likely because the desired structure is already well aligned with a simple geometric decomposition.

Finally, the optimization framework is computationally intensive. Despite being faster than exhaustive search, the SDS-based optimization requires hundreds of diffusion model evaluations, taking approximately 15-20 minutes per design on an A100. Real-time interactivity—where a user moves a physical die and the system suggests completions instantly—would require distilling the optimization process into a feed-forward network or a faster search policy.

6 Conclusion and Future work

We have presented Diceplay, a system for computational design of abstract imagery on a modular physical canvas. Our central technical contribution is showing that highly discrete, combinatorial design spaces can be made amenable to gradient-based optimization through careful grammar design.

On the application side, there are several directions for extending Diceplay. One natural extension is to apply this method to other sets of geometric patterns that are clear and interpolable. Since our grammar was manually designed for this specific set of primitives, doing so would require extending these ideas to new sets manually or through automated methods. We could also allow multiple types of dice with different sets of primitives would expand the visual vocabulary of the display. This would require the optimization framework to reason about uniqueness and allocation constraints, but could enable richer compositions.

Future work could also consider co-designing the display for specific applications. For curated installations or ambient displays with a known set of target compositions, it may be beneficial to jointly optimize the dice primitives and the intended content. While trivial for very small target sets, this becomes more interesting when balancing a larger number of desired compositions against a limited number of physical primitives.

Another possible extension is to explore three-dimensional compositions. As an alternative fabrication approach, we experimented with adding an array of magnets to all faces of the dice, allowing them to connect in arbitrary configurations. This supports volumetric assembly and suggests the possibility of sparse 3D structures in which both dice placement and orientation are optimized, subject to constraints such as connectivity. Extending Diceplay to this setting would introduce new challenges, including reasoning about multiple viewpoints and evaluating 3D abstractions.

Finally, our results suggest that Diceplay may function as an educational toy for young children. More broadly, pattern recognition and pattern construction play an important role in cognitive development and are widely used in educational tools for both children and adults, including those designed to support cognitive health.

Exploring these applications more systematically is a promising direction for future work.

On the technical side, our results highlight a simple but important principle: the structure of a design representation fundamentally shapes the optimization landscape. In highly discrete settings like Diceplay, grammar-based optimization offers an alternative to standard soft-blending approaches by explicitly defining search paths that respect the geometry of the design space. This perspective builds on ideas introduced in Design for Descent [Kodnongbua et al. 2025], which describes principles for grammar design that make design spaces more amenable to gradient-based optimization. Prior work, however, focuses on grammars with mixed discrete–continuous structure. In contrast, our setting is fundamentally discrete, and the grammar we design for Diceplay is *not* one that merely describes the design space, but one that *intentionally relaxes* it by introducing continuous parameters that do not exist in the original domain and penalizing deviations to encourage convergence back to valid discrete configurations.

More broadly, this suggests a general strategy for grammar-based optimization: starting from an existing grammar, constructing principled relaxations that enable optimization, and then mapping optimized solutions back to the original domain. Such relaxation techniques are widely used in optimization and have a long history in geometry processing, where continuous relaxations are employed to make combinatorial problems tractable. While Diceplay serves as a concrete case study of applying these ideas in a grammar-based design setting, an important direction for future work is to explore more systematic approaches to this process—for example, methods for automatically relaxing a given grammar to make it amenable to optimization, along with principled ways of projecting optimized results back to valid discrete designs. Developing such techniques could help broaden the applicability of grammar-based optimization to a wider range of design problems.

Acknowledgments

This work was supported by NSF 2537422, a Sloan Research Fellowship, an Amazon Research gift, and in part by BNBU Research Grant with No. of UICR0700147-26. Claude (Anthropic) and Gemini (Google) were used to provide support during parts of the code development process. ChatGPT (OpenAI) and Gemini (Google) were used to assist with writing tasks such as grammar checks, wording, and clarity revisions. All suggested changes were reviewed and included only after author evaluation and revision.

References

Daniel Berio, Michael Stroh, Sylvain Calinon, Frederic Fol Leymarie, Oliver Deussen, and Ariel Shamir. 2025. Neural Image abstraction using long smoothing B-splines. *ACM Transactions on Graphics (TOG)* 44, 6 (2025), 1–11.

Alexandre Binninger and Olga Sorkine-Hornung. 2024. SD-πXL: Generating Low-Resolution Quantized Imagery via Score Distillation. In *SIGGRAPH Asia 2024 Conference Papers*. 1–12.

Desai Chen, Pitchaya Sitthi-amorn, Justin T. Lan, and Wojciech Matusik. 2013. Computing and Fabricating Multiplanar Models. *Computer Graphics Forum* 32, 2pt3 (2013), 305–315.

Rulin Chen, Xuyang Ma, Praveer Tewari, Chi-Wing Fu, and Peng Song. 2025. Inverse Tiling of 2D Finite Domains. In *ACM SIGGRAPH Asia 2025 Conference Papers*.

Rulin Chen, Ziqi Wang, Peng Song, and Bernd Bickel. 2022. Computational Design of High-level Interlocking Puzzles. *ACM Transactions on Graphics (SIGGRAPH)* 41, 4 (2022), 150:1 – 150:15.

Paolo Cignoni, Nico Pietroni, Luigi Malomo, and Roberto Scopigno. 2014. Field-aligned mesh joinery. *ACM Transactions on Graphics* 33, 1 (2014), 1–12.

Filippo A. Fanni, Gianmarco Cherchi, Alessandro Muntoni, Alessandro Tola, and Riccardo Scateni. 2018. Fabrication Oriented Shape Decomposition using Polycube Mapping. *Computers Graphics* 77 (2018), 183–193.

Kevin Frans, Lisa Soros, and Olaf Witkowski. 2022. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *Advances in Neural Information Processing Systems* 35 (2022), 5207–5218.

Jiahao Ge, Mingjun Zhou, Wenrui Bao, Hao Xu, and Chi-Wing Fu. 2024b. Creating LEGO figurines from single images. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–16.

Jiahao Ge, Mingjun Zhou, and Chi-wing Fu. 2024a. Learn to create simple LEGO micro buildings. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–13.

Kristian Hildebrand, Bernd Bickel, and Marc Alexa. 2012. crdbrd: Shape Fabrication by Sliding Planar Slices. *Computer Graphics Forum* 31, 2pt3 (2012), 583–592.

Ajay Jain, Amber Xie, and Pieter Abbeel. 2023. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1911–1920.

Milin Kodnongbua, Zihan Zhang, Nicholas Sharp, and Adriana Schulz. 2025. Design for Descent: What Makes a Shape Grammar Easy to Optimize?. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers (SA Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 172, 11 pages. doi:10.1145/3757377.3764004

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. 2020. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.

LiKee. n.d.. LiKee Open Ended Wooden Shape Puzzles. https://www.amazon.com/dp/B07QZB8PLW. Accessed: January 2026.

LovesTown. n.d.. LovesTown 209 PCS Shapes Wooden Pattern Blocks. https://www.amazon.com/dp/B08C37JMP2. Accessed: January 2026.

Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. 2015. Legolization: Optimizing lego designs. *ACM Transactions on Graphics (ToG)* 34, 6 (2015), 1–12.

Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. 2022. Towards Layer-wise Image Vectorization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Nityanand Mathur, Shyam Marjit, Abhra Chaudhuri, and Anjan Dutta. 2025. CLIP-Draw++: Text-to-Sketch Synthesis with Simple Primitives. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 6247–6256.

Eder Miguel, Mathias Lepoutre, and Bernd Bickel. 2016. Computational design of stable planar-rod structures. *ACM Transactions on Graphics (SIGGRAPH)* 35, 4 (2016), 1–11.

Vihaan Misra, Peter Schaldenbrand, and Jean Oh. 2025. ShapeShift: Towards text-to-shape arrangement synthesis with content-aware geometric constraints. *arXiv preprint arXiv:2503.14720* (2025).

Alessandro Muntoni, Marco Livesu, Riccardo Scateni, Alla Sheffer, and Daniele Panozzo. 2018. Axis-Aligned Height-Field Block Decomposition of 3D Shapes. *ACM Transactions on Graphics* 37, 5 (2018).

Online-Schule für Gestaltung. 2024. 2D Modular Design. https://www.pinterest.com/pin/591871576061711785/. Accessed: 2026-01-19.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. arXiv:1901.05103 [cs.CV] https://arxiv.org/abs/1901.05103

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2022. DreamFusion: Text-to-3D using 2D Diffusion. arXiv:2209.14988 [cs.CV] https://arxiv.org/abs/2209.14988

Pozale. n.d.. Pozale Pattern Blocks and Boards with 120 Pieces. https://www.amazon.com/dp/B0D3HW2QSG. Accessed: January 2026.

Shen-Guan Shih. 2016. On the Hierarchical Construction of SL Blocks. In *Proc. of Advances in Architectural Geometry*. 124–136.

Kenji Tojo, Ariel Shamir, Bernd Bickel, and Nobuyuki Umetani. 2024. Fabricable 3d wire art. In *ACM SIGGRAPH 2024 Conference Papers*. 1–11.

Yael Vinker, Ehsan Pajouheshgar, Jessica Y Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. 2022. Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–11.

Zhenyu Wang, Jianxi Huang, Zhida Sun, Yuanhao Gong, Daniel Cohen-Or, and Min Lu. 2025. Layered image vectorization via semantic simplification. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 7728–7738.

Ziqi Wang, Peng Song, and Mark Pauly. 2021. State of the art on computational design of assemblies with rigid parts. In *Computer graphics forum*, Vol. 40. Wiley Online Library, 633–657.

Ximing Xing, Juncheng Hu, Jing Zhang, Dong Xu, and Qian Yu. 2024. SVGFusion: Scalable Text-to-SVG Generation via Vector Space Diffusion. *arXiv preprint arXiv:2412.10437* (2024).

Hao Xu, Ka-Hei Hui, Chi-Wing Fu, and Hao Zhang. 2021. Computational LEGO technic design. *ACM Trans. Graph.* 38, 6, Article 196 (Aug. 2021), 14 pages. doi:10.1145/3355089.3356504

Yinan Zhang and Devin Balkcom. 2016. Interlocking Structure Assembly with Voxels. In *Proc. of IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. 2173–2180.

Yinan Zhang, Yotto Koga, and Devin Balkcom. 2021. Interlocking Block Assembly With Robots. *IEEE Transactions on Automation Science and Engineering* 18, 3 (2021), 902–916.

Mingjun Zhou, Jiahao Ge, Hao Xu, and Chi-Wing Fu. 2023. Computational Design of LEGO® Sketch Art. *ACM Trans. Graph.* 42, 6, Article 201 (Dec. 2023), 15 pages. doi:10.1145/3618306

Henrik Zimmer and Leif Kobbelt. 2014. Zometool Rationalization of Freeform Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1461–1473.

Henrik Zimmer, Florent Lafarge, Pierre Alliez, and Leif Kobbelt. 2014. Zometool shape approximation. *Graphical Models* 76, 5 (2014), 390–401.