Design Adjectives: A Framework for Interactive Model-Guided Exploration of Parameterized Design Spaces

Evan Shimizu
Carnegie Mellon Univeristy
Pittsburgh, PA
eshimizu@cs.cmu.edu

Matt Fisher
Adobe Research
San Francisco, CA
matfishe@adobe.com

Sylvain Paris
Adobe Research
Boston, MA
sparis@adobe.com

James McCann
Carnegie Mellon University
Pittsburgh, PA
jmcann@cs.cmu.edu

Kayvon Fatahalian
Stanford University
Stanford, CA
kayvonf@cs.stanford.edu

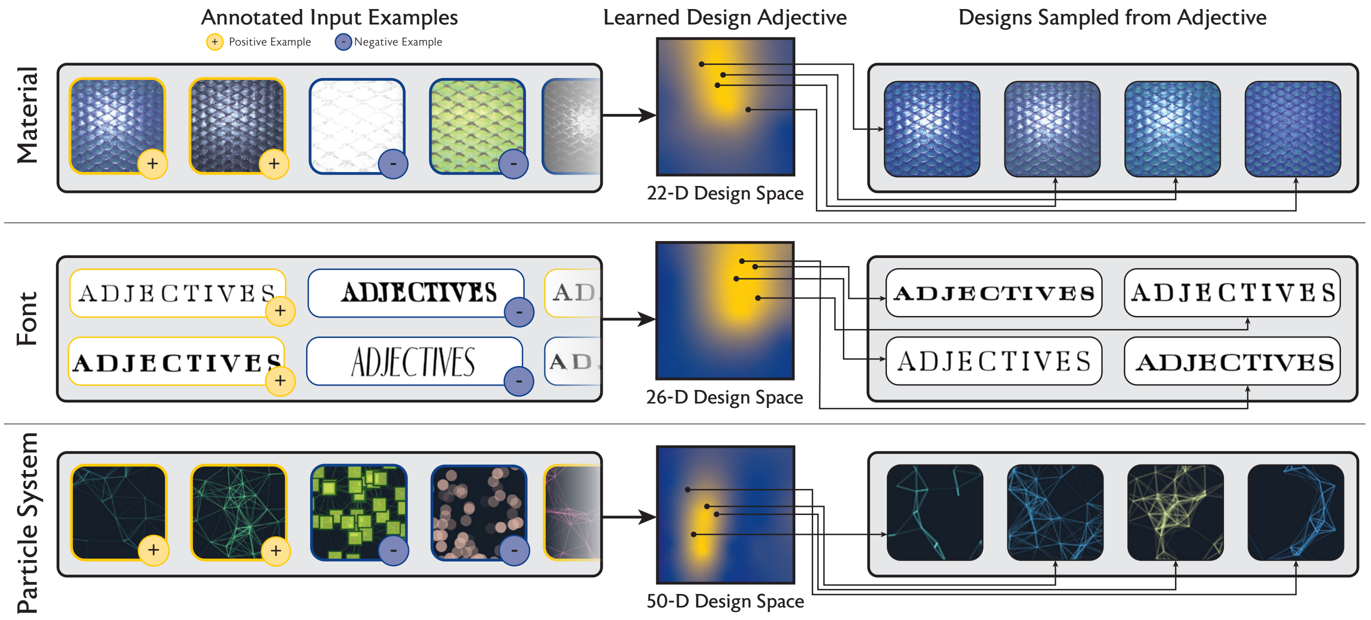


Figure 1. A design adjective is a learned model of a design concept, and can be used in the design adjectives framework to guide exploration of parameterized design spaces. From top to bottom, design adjectives defined in three domains (parametric materials, parametric fonts, particle systems). Each row depicts the process of creating a design adjective from a set of annotated examples in the design space (left), learning the adjective's design preference function through Gaussian process regression (center), visualized here with a 2D slice of the function values indicating scores with color (0 (blue) 1 (yellow)), and generating design suggestions by sampling the design adjective (right).

ABSTRACT

Many digital design tasks require a user to set a large number of parameters. Gallery-based interfaces provide a way to quickly evaluate examples and explore the space of potential designs, but require systems to predict which designs from a high-dimensional space are the right ones to present to the user. In this paper we present the *design adjectives framework* for building parameterized design tools in high dimensional design spaces. The framework allows users to create and edit *design adjectives*, machine-learned models of user intent, to guide exploration through high-dimensional design spaces. We provide a domain-agnostic implementation of the design adjectives framework based on Gaussian process regression, which is able to rapidly learn user intent from only a few examples. Learning and sampling of the design adjective occurs at interactive rates, making the system suitable for iterative design workflows. We demonstrate use of the design adjectives framework to create design tools for three domains: materials, fonts, and particle systems. We evaluate these tools in a user study showing that participants were able to easily explore the design space and find designs that they liked, and in professional case studies that demonstrate the framework's ability to support professional design concepting workflows.

Author Keywords
Design tools, creativity support, interactive interfaces, sampling methods

CCS Concepts
•Human-centered computing → Interactive systems and tools; •Computing methodologies → Graphics systems and interfaces; •Applied computing → *Media arts;*

INTRODUCTION

Many design tasks in digital applications take place in parameterized spaces, where an output design is determined by tens to hundreds of parameters. These design spaces occur commonly in domains such as parametric materials, parametric fonts, and particle systems for web applications, game engines, and visual effects (Figure 1). New design spaces arise often (e.g., each procedural material might have its own unique parameters), creating never seen before spaces for designers to explore. For example, the fish scale material created with Substance Designer [40] (Figure 1-top) has 22 parameters; the brick material (Figure 7), also created with Substance Designer, has 62; and the five Prototypo [31] font templates in Figure 13 share a set of parameters but produce significantly different visual outputs.

When performing a design task there is often no single "correct" solution. The criteria used to determine success is a combination of task constraints and a designer's personal preferences, both of which are often poorly defined at the start, and which become more definite over the duration of the design process [39]. As a result, designers typically engage in an iterative process where multiple designs are rapidly generated and evaluated against current objectives [15, 45, 21]. This problem-solving process occurs at all skill levels [16], and the ability to efficiently generate and refine designs is a critical part of effective design interfaces [36].

Unfortunately, most commercial tools [40, 41, 43, 3] only provide interfaces for modifying parameter values one-by-one with per-parameter sliders. Design-gallery based interfaces [23] provide a way to rapidly explore a range of design possibilities and evaluate examples, but to be effective in high dimensional spaces they must predict what designs are most useful to present to the user throughout the iterative design process. To address this problem, we present the *design adjectives framework* for building parameterized design tools for high dimensional design spaces. Central to the framework is a *design adjective*: a learned model of a user's design intent. The design adjectives framework provides support for interactively creating and sampling from design adjectives to populate gallery-based interfaces. We demonstrate that this combination facilitates interactive exploration of high-dimensional parameterized design spaces, and also provides opportunities to enhance the per-parameter slider interfaces present in state-of-the-art systems. Specifically we make the following contributions:

The design adjectives framework. We define a domain-agnostic framework for creating parameterized design tools that guide user exploration through high-dimensional design spaces using a learned model of user intent called a design adjective. The components of the framework are aligned with the conceptual progression of the design process: providing support for preliminary exploration of the design space, refinement of a design idea, and per-parameter fine tuning needed in professional design tools.

An implementation of design adjectives using Gaussian process regression [32]. This model requires only a small number of training examples to capture preference. It learns at interactive rates, allowing it to support interactive, iterative design workflows that include evolving user preferences. The implementation also provides user interface components inspired by prior work on design galleries and slider highlighting [23, 19] to support visual exploration of the design space guided by examples from the learned preference model.

An evaluation of design tools built using the design adjectives framework that demonstrates that our implementation supports exploratory design in the font design and material design domains. We find that users are more easily able to create designs that they like with the design adjectives framework, also and find that the interface supports professional design concepting workflows.

RELATED WORK

The design process is characterized by three distinct phases: preliminary design (creating an initial design), refinement (iterating on or exploring an existing feature), and detail (fine-tuning specific aspects of the design) [15]. The goal of the design adjectives framework is to facilitate user interaction with a design space in a manner that mirrors these steps: defining an initial model of user intent (the learned *design adjective*), refining that model by reacting to examples produced by an interactive sampling process, and fine-tuning using low-level parameter controls. Our solution involves a synthesis of prior work in creativity support tools, parameter manipulation tools, re-parameterization methods, and machine learning, which we describe here.

In parameterized design spaces, sampling-based design tools have been proposed as an exploration support tool. A possible approach is to exhaustively sample the space and present those samples to the user in a Design Gallery [23]. Exhaustive sampling of arbitrary design spaces becomes more difficult as the number of parameters increases. More efficient sampling methods use some form of human-provided seed data to locate regions of interest in the space [33, 22, 42], or objective functions, if they exist, to limit the scope of designs found in the space [24, 37]. Our generic implementation of design adjectives cannot rely on such objectives, as the domain itself is the one component we assume is variable. Such gallery-based tool

UIST'20, October 20–23, 2020, Minneapolis, MN, USA

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-6708-0/20/04…$15.00

DOI: https://doi.org/10.1145/3313831.XXXXXXX

allow users to pick through the provided samples, but refining designs in these frameworks can be difficult. Users can only explore designs that they can see in the gallery, and must use a different tool to perform adjustments.

The most direct way to adjust a parameterized design is to manually set a parameter value. These controls are typically displayed as sliders. Many methods have been developed to assist users with locating what visual changes a parameter is responsible for [4, 44, 35], and some methods overlay a preference function on the sliders to assist with finding generally good configurations [19]. It is difficult to create many alternatives when manipulating parameters one-by-one, and this class of interface is better suited for detail work instead of exploratory design. Therefore, we employ slider highlighting techniques similar to those described in Koyama et al. [19] primarily for the detail design phase.

One way to speed up the process of performing per-parameter design is to re-parameterize the design space to have fewer dimensions; ideally ones that are meaningful to designers. Early methods to do this in computer graphics used support vector machines trained on large data sets to create axes representing different traits [25]. Later methods adopted the term *semantic attributes* to describe this style of re-parameterization, using modern machine learning techniques to generate using human-understandable image labels [10, 11]. Semantic attributes have seen success in a variety of computer graphics domains [6, 29, 20, 48, 38, 8], presenting the learned attributes as sliders that could be manipulated as if they were normal parameters.

Semantic attributes model objective functions by using an expensive learning process on a large amount of annotated data. While working with these attributes, designers may want to personalize or adjust the attributes to be closer to their individual design goals. Methods exist for refining semantic attributes [30, 18], but such methods can only be used after an initial set of attributes has been defined. Additionally, these methods assume that the interests of designers are "regular" [5, 19], meaning that there exists a consensus about what the attribute means between designers. In creative domains, this assumption does not always hold true. Instead of pre-defining attributes across the space for later adjustment, a design adjective learns a model of intent from input data provided by a single user.

Methods for modeling an objective function given a small amount of input data have been studied in machine learning. This class of problems is referred to as few-shot, or zero-shot, learning. Of the numerous techniques that could be used to implement a low-data model of an objective function, we have chosen Gaussian process regression (GPR) [32] to implement a design adjective. This technique was selected due to prior work demonstrating success with computer graphics design tasks for material design [49] and shape grammars [7]. We build upon work in [49] to demonstrate GPR's ability to support design tasks in general parameterized design spaces.

Interactive approaches for defining concepts and exploring a design space were proposed as possible interface paradigms early in the computer graphics literature [17], but these methods rely on the ability to generate and render design suggestions in real time in order to allow designers to quickly perform iteration cycles. At the time, the computational power available to these methods limited which domains they could be used on. Recent advances in computer hardware and algorithms have led to the development of techniques that enable real-time rendering of complex design domains [46, 47, 27, 34, 49], allowing them to be used in interactive machine learning. The design adjectives framework applies the iterative, trial-and-error, training paradigm presented by Amershi et al. [2, 1] in parameterized design spaces for creative applications.

DESIGN ADJECTIVES OVERVIEW

To understand how design adjectives are used to support preliminary design, refinement, and final detail, consider the following scenario: Jen, a material designer, seeks to create a shiny blue version of the fish scale material shown at the top of Figure 1.

Preliminary Design

The first thing Jen wants to do in the preliminary design phase is to see what material variations are possible in the 22-parameter design space. This requires the generation of a large number of diverse materials. Jen can use a bootstrapper provided by the framework to generate a set of random fish scale designs. She can then browse through a gallery of these designs to select and assign high scores to designs that best represent her vision of "shiny blue scales" and low scores to those that do not (Figure 1-top left). The examples provided to the adjective define Jen's initial model of intent.

Refinement

After surveying the design space and establishing an initial idea for what "shiny blue scales" means to her, Jen creates a number of variations of her initial designs, refining her design ideas as she does so. Jen generates designs by sampling from the "shiny blue scales" design adjective. Jen decides to look at designs that have similar scores to her current design. (Figure 1-top right). Jen sees a returned sample that is not as shiny as the others, and assigns it a low score to update the adjective. She then runs another sampling operation with the updated adjective. The interactive refinement loop created by Jen's repeated sampling and updating of the design adjective enables intent-driven exploration of the design space.

Detail Design

Towards the end of the design process, Jen wants to make a few final tweaks. To do so, Jen uses the individual parameter controls. The adjective that Jen created highlights which parameters had the most impact on her definition of "shiny blue scales," helping Jen determine which parameters to change for her final edits. Jen can make use of the framework's per-parameter tools at any time.

THE DESIGN ADJECTIVES FRAMEWORK

To enable workflowssuch as the one described above, the design adjectives framework consists of components that interface with design adjectives (Figure 2). The framework is designed to support guided exploration based on design adjectives learned from personal preferences, and facilitate access

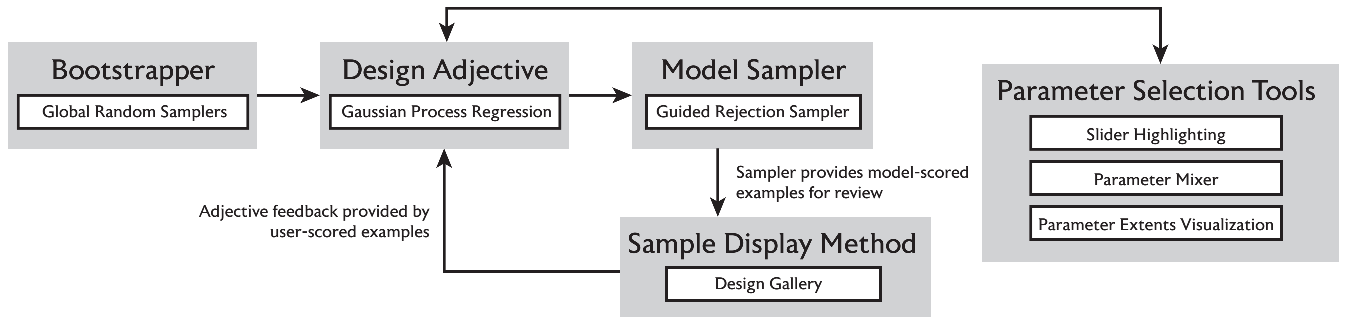

Bootstrapper
Global Random Samplers
Design Adjective
Gaussian Process Regression
Model Sampler
Guided Rejection Sampler
Parameter Selection Tools
Slider Highlighting
Parameter Mixer
Parameter Extents Visualization
Sampler provides model-scored examples for review
Adjective feedback provided by user-scored examples
Sample Display Method
Design Gallery

Figure 2. ***Overview of the Design Adjectives Framework*****. The framework consists of a set of components that allow users to explore a design space in a way that aligns with the design process itself. These components are shown in the shaded boxes, with our domain-agnostic implementation of the component shown in white boxes. The framework is centered around a design adjective, defined by scored examples, that can be used to generate new designs through a sampling method. Adjectives can be updated by adding new examples or modifying the scores of existing examples, and are initialized with the aid of a bootstrapper. Low-level parameters are always accessible. The model can be used to aid per-parameter operations, and per-parameter controls can help filter out parameters during adjective sampling.**

to low-level parameter tools. The interfaces specified by the framework support an interactive workflow, using real-time sampling to enable guided exploration, and always represents samples in the original parameter space to allow access to low-level parameters at all times.

In this section, we provide a specification for the design adjectives framework. We provide a domain-agnostic implementation of the full framework in the next section. Of course, other implementations of the framework could modify or replace these components based on the needs of a specific design domain.

Parameterized Design Space Definition

The design adjectives framework supports design domains consisting of a finite set of bounded, real-valued parameters. The domain also must provide a rendering function that, given a point in the design space, generates a visual preview of the corresponding design (preferably at interactive rates). The rendering function is required for enabling visual exploration, and parameter bounds are required for normalizing data passed to the design adjective. Our domain-agnostic implementation only requires a design domain definition for use in a new domain.

The Design Adjective

The design adjective represents user intent from user-scored input examples. The input to the adjective is assumed to be standard example-score pairings, with examples created in the design space that the user is working in. The output of the adjective is a function that assigns a real-valued preference score to all points in the design space. The adjective must be able to learn a user's intent given a small amount of initial data, and be trained and updated at interactive rates in order to support rapid iteration. The user should be able to incrementally update the adjective by providing new examples or editing existing example scores. The adjective should also provide the ability to operate on a user-specified subset of parameters within the design space.

Sampling from the Design Adjective

Designs are generated from the adjective with a sampler. Generating designs serves two purposes: to show the user things that they might like, based on the current adjective definition, and to give the user the chance to assess the accuracy of the adjective and adjust the definition by manually scoring the returned suggestions.

This sampler must be able to return results at interactive rates (one second or less) according to what the user wants to see relative to the current adjective (such as "more like this" or "similarly scored designs"). Note that this interactivity requirement can be met by streaming results into the UI, allowing users to explore designs as they get generated instead of waiting for the entire process to complete. To speed up this process, the adjective must provide its training data to the sampling method, allowing the sampler to potentially use this data as part of the sampling process. For example, our implementation of this sampler uses high-scored inputs to accelerate the sampling process. Results returned by the sampler must be arranged in the UI in a way that allows users to update the definition of an adjective. In our implementation, we present the results in a gallery view.

Bootstrapping the Adjective

The adjective must use in-domain examples as inputs. This means that when a new adjective is created, there are no existing examples to use, and users must find some way to provide initial examples and scores to the adjective. At minimum, this can be done with existing per-parameter controls, but better tools can be provided. Our implementation provides a uniform random sampler as its bootstrapper. We expect that interfaces developed for a specific domain will be able to provide better bootstrappers by using domain-specific design principles.

Per-Parameter Identification Tools

The design adjective can be used to provide additional support during the detail design phase. The preference function can be used to enable tools such as per-parameter highlighting [19] when performing detail design. All framework components operate on the original parameters, so other techniques for parameter identification [44, 35] can be easily added to any implementation of this framework.

IMPLEMENTING DESIGN ADJECTIVES

We present a domain-agnostic implementation of the design adjectives framework that can be used with any design domain that meets the framework specification. Domain-specific implementations of this framework gain immediate access to all domain-agnostic components, and can replace these components with domain-specific variations as desired.

Building Adjectives using Gaussian Process Regression

The design adjective takes as input a set of points in the design space with associated scores $\{(\mathbf{x}_1, f_1), (\mathbf{x}_2, f_2)\ldots\}$, where the $\mathbf{x}_i$ points are parameter vectors in the design space $\mathcal{D}$ and the f_i values are the scores between 0 and 1 assigned by the user to these points. We use this data to estimate the preference function $f(\mathbf{x})$ across the entire design space at interactive rates. Our implementation uses Gaussian process regression (GPR) to estimate this function. We chose GPR because it satisfies the requirements expected by a design adjective; it is able to estimate functions given a small amount of input data, and can be trained in real time. We perform the regression with GPyTorch [13]. GPR is a known technique and we refer to the book by Rasmussen and Williams [32] for a detailed review. For the sake of completeness, the rest of this section introduces the basic concepts required in our context and discusses how they apply to the design adjectives framework.

GPR uses a Gaussian process (GP) to estimate the value of an unknown function. A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution. Gaussian processes are completely specified by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$. In the case of a design adjective, the random variables are the preference scores f_i of the points $\mathbf{x}_i$ in the design space. An input pair $(\mathbf{x}_i, f_i)$ can be viewed as a point sample of the preference score at $\mathbf{x}_i$. A GP defines a distribution over functions, meaning that the value of the GP at a point $\mathbf{x}^*$ in the space is itself a Gaussian distribution. We use the mean of the GP as the estimate of the adjective score at an unobserved point $\mathbf{x}^*$. We do not make use of the variance in our implementation.

To define a GP for a given design adjective, we create the matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2 \ldots]$ (i.e., its columns are the points with a user-assigned score) and the vector $\mathbf{f} = [f_1, f_2 \ldots]^\mathsf{T}$ from the scores, and use a standard zero-mean function $m(\mathbf{x}) = 0$, indicating that every unobserved point is assumed to have a low adjective score. For the covariance function, we use the radial basis kernel:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^\mathsf{T}\,\Theta^{-2}\,(\mathbf{x}-\mathbf{x}')\right) \qquad (1)$$

where Θ is a diagonal matrix learned at training time that applies a scaling factor to each dimension of the design space. This function k can be used to construct a covariance vector $\mathbf{k}(\cdot,\cdot)$ with the covariance values between the column vectors of a matrix and a given vector, and a covariance matrix $\mathbf{K}(\cdot,\cdot)$ with the covariance values between all pairs of vector columns of two matrices. Using these quantities, the GP estimates the score at a new point $\mathbf{x}^*$ as follows.

$$f(\mathbf{x}^*) = \mathbf{k}(\mathbf{X}, \mathbf{x}^*)^\mathsf{T}\,\mathbf{K}(\mathbf{X}, \mathbf{X})^{-1}\,\mathbf{f} \qquad (2)$$

The GP defined above can actually be used as an estimator immediately, although it is not likely to be accurate, as the covariance function does not accurately represent the relationships between the different parameters. In order to improve the accuracy of the function, we optimize the GP by adjusting the scaling factors contained in the Θ matrix of the covariance function for each parameter (Eq. 1), a process called *automatic relevance determination* [32, ch. 5.1]. Intuitively, the scaling factors indicate how quickly the function varies along each dimension. In this instance, small factors indicate that the function value varies quickly, while large factors indicate that the function varies slowly. The value of the factor associated with a parameter is a proxy for how important it is to the overall function score. After completing this regression process, the GP can be used to more accurately estimate adjective scores for arbitrary points.

Adjectives may be defined on a subset of parameters in the design space. This subset is referred to as the adjective's *affected parameters*. Using GPR to implement affected parameters is straightforward: the input parameter vectors $\mathbf{x}_i$ defined on the full design space are projected onto the affected parameter subspace before being provided to the GPR. Our implementation automatically detects affected parameters from the input, assuming that a parameter is affected if it has different values in any two input points. Users can override the set of affected parameters at any time.

Generating Designs with a Guided Rejection Sampler

In order to use the Gaussian process model of an adjective in the design adjectives framework, we need to provide a sampling method. To return samples as quickly as possible, we introduce a guided rejection sampler on the design adjective (Alg. 1), accepting samples that meet a user-specified criteria described later in this section, and are different enough from already accepted samples according to a L^2 distance threshold.

Conceptually, Algorithm 1 attempts to guide itself towards good samples by starting with highly-rated examples provided by the user (Step #1), and randomizing a subset of the parameters of those examples (Step #2). The sampler should be returning novel samples, not just recycling what the designer has already input, so we do not use a hill climbing (or similar) step in the sampler to avoid returning the best already labeled samples. In our implementation, p_0 is set equal to half of the length of the design space parameter vector, $p_{\text{floor}} = 3$, $maxIter = 10000$, $r = 10$, and $count = 20$. All five settings can be modified by the users, although none of them saw the need for it in our experiments.

Acceptance Criteria

Users have different intents at different stages of their exploration. We express these intents in the sampling algorithm by changing the rejection sampler's acceptance criteria (the `Accept`$(\mathbf{x}, f(\mathbf{x}))$ function in Algorithm 1). We provide four criteria in this implementation (Figure 3).

Algorithm 1: Guided Rejection Sampler

```
input    adjective function and input examples: $f(\cdot)$, $\mathcal{X}$
         set of affected parameters: $\mathcal{P}$
         current design: $\mathbf{x}_0$
         settings of the fine randomization: $p_0$, $p_{\text{floor}}$, $r$
1        max number of iterations before bailing out: $maxIter$
         number of samples to generate: $count$
         acceptance criteria function: $Accept(\mathbf{x}, f(\mathbf{x}))$
output   set of samples: $\mathcal{S}$
2  // Initialization
3  $\mathcal{S} \leftarrow \{\}$    $p \leftarrow p_0$    $i, p_{\text{count}} \leftarrow 0$
4  $\mathcal{X}_{\text{good}} \leftarrow \{\mathbf{x}_i \in \mathcal{X}, \text{s.t. } f(\mathbf{x}_i) > 0.5\}$ // Select examples
     with a score above 50%.
5  // Main loop of the sampler
6  while $(|\mathcal{S}| < count)$ and $(i < maxIter)$ do
7      // Step #1: Coarse Randomization
         Randomly select a good example.
8      $\mathbf{x} \leftarrow GetRandomElement(\mathcal{X}_{\text{good}})$
9      // Step #2: Fine Randomization
         Randomly perturb a random parameter subset.
10     $\mathcal{I}_{\text{random}} \leftarrow SelectRandomIndices\ (\mathcal{P}, p)$ // Subset of
         affected indices to randomize
11     for $index \in \mathcal{I}_{\text{random}}$ do
12         $\mathbf{x}[index] \leftarrow UniformRnd(0, 1)$ // Use a uniform
             distribution to randomize the selected indices.
13     // If a parameter is not affected by the adjective, set it
         to the value of the current design.
14     for $index \notin \mathcal{P}$ do
15         $\mathbf{x}[index] \leftarrow \mathbf{x}_0[index]$
16     // If this is a valid sample...
17     if $Accept(\mathbf{x}, f(\mathbf{x}))$ then
18         $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{x}\}$ // add it to sample set...
19         $p \leftarrow p_0$ // and reset the fine randomization.
20     else
21         $p_{\text{count}} \leftarrow p_{\text{count}} + 1$
22         // If the algorithm failed too many times to find a
             valid sample...
23         if $p_{\text{count}} > r$ then
24             $p_{\text{count}} \leftarrow 0$ // reset the counter...
25             $p \leftarrow \max(p - 1, p_{\text{floor}})$ // and weaken the fine
                 randomization without going below a
                 minimum.
26     $i \leftarrow i + 1$
```

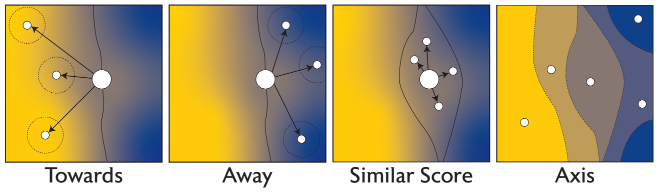


Figure 3. ***Sampling Mode Visualizations*. The four sampling modes described in the "Acceptance Criteria" section visualized on a 2D design space. The adjective score is represented by color, with yellow indicating high scores and blue indicating low scores. The starting point for each sampling method is indicated with the large circle, and the sample points represented with smaller circles. Dotted lines around the sample points in Towards and Away represents the difference threshold criteria.**

Towards generates samples with a higher adjective score, which is useful, for instance, when searching for the final design. New designs must score better than the current design.

Away generates samples with a lower adjective score, e.g., to refine the definition of the adjective. New designs must score worse than the current design. This sampling mode is used to get out of local maxima, moving the current design away from known good samples in order to show more variety.

Similar Score picks variants with similar "amount of the adjective" to the current design, e.g., to help users explore the design space without converging to a high-score sample. New designs must have an adjective score within $\pm 10\%$ of the current design's score. Samples returned by this method are not necessarily visually similar to the current design, as the similarity metric is the adjective score not parameter vector distance.

Axis visualizes samples regularly spaced on the adjective scale, which is useful to assess the range of the effect captured by the adjective and quickly navigate through it. All new designs are accepted at first, but an additional de-duplication step is added where designs that score within $\pm 2.5\%$ of a previously accepted design are rejected.

Bootstrapping Design Adjectives

Adjectives are defined by labeled examples, and a newly created adjective has no examples to start with. In order to initialize, or bootstrap, the adjective, examples must be provided by the user. To assist users in this task, the design adjectives framework requires a bootstrapping tool. We provide two random samplers for generating preliminary examples in a domain-agnostic scenario.

The *uniform randomizer* draws samples from a uniform random distribution. The *jitter* sampler applies a random delta to each parameter in the current parameter vector. These functions can either operate on all parameters in the design space, or use a subset of parameters selected by the user. We find that these samplers are sufficient for working in generic design spaces.

Per-Parameter Tools

As the designer reaches the end of the design process, small tweaks to the design are performed using individual parameter

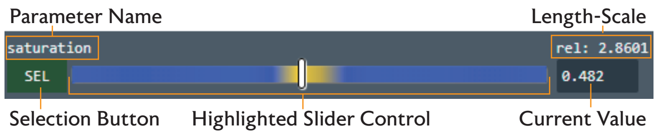


Figure 4. ***Slider Controls*****. Labeled components of a single parameter control element in our adjectives interface (shown in Figure 7). The slider highlighting indicates how the current adjective score would change if the parameter were moved to the corresponding location. Yellow highlighting indicates a higher score, blue indicates a lower score.**

controls. The framework requires that the output of sampling an adjective is a design representable in the original parameter space, so all existing per-parameter tools can be used in this platform. We provide a set of tools for per-parameter manipulation that we found to be most useful for users working in general parameterized spaces. This is a non-exhaustive list of per-parameter tools, and we expect that domain-specific implementations of this platform will add additional tools specific to their domain.

Highlighted Parameter Sliders

We color-code the sliders to show the user how the adjective score would change if the parameter value were modified (Figure 4). This technique helps users navigate complex design spaces [19]. To color-code the sliders, we evaluate the adjective at individual parameter values across its full range, sampled in increments of 5%, while holding all other parameters constant. Our interface supports both absolute and relative color coding of of sliders. Absolute color coding assumes the adjective function values fall within the $[0,1]$ range and assign colors accordingly, while relative color coding assigns colors relative to the minimum and maximum values observed in the parameter sweep over all parameters. Running the color-coding operation is almost instant in our implementation, allowing users to perform optimization steps themselves, if they desire, during the fine-tuning phase.

Parameter Selection

Adjectives do not need to affect every parameter in the design space. Designers can select a subset of parameters to work with while defining adjectives and running the bootstrapping tools. Users may wish to do this to reduce the dimensionality of the search space; focusing only on changing parameters that are likely to lead to a design of interest. Expert designers may not need help identifying which parameters are relevant, but novice users may have trouble determining what a parameter does.

To assist users in finding relevant parameters, we provide a visualization of the *parameter extents*. Clicking on the name of a parameter toggles this visualization, which displays thumbnails of designs that span the full range of the selected parameter's values while all other parameters are held constant (Figure 5). This visualization is a type of side view [44], providing assistance with locating the effect of a single parameter so that the user can determine if the parameter should be included or excluded in the next operation.

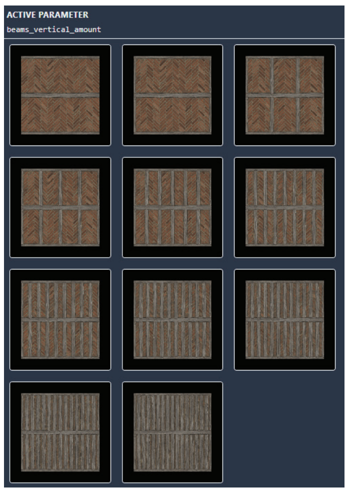


Figure 5. ***Parameter Extents*****. Displaying thumbnails for values of "beam_vertical_amount" between 0 and 1.**

Parameter Mixer

The *mixer* utility takes two parameter vectors and creates a new vector by randomly combining parameters from the input vectors, similar to a crossover operation in a genetic algorithm (Figure 6). Designers can set a bias value to select parameters more frequently from one design over the other.

The mixer returns a gallery of results, each with a different set of parameters selected for combination. This allows designers to pick out elements that they like in a design, without needing to know which exact parameters are responsible for creating that element. The mixer does not require an adjective to operate. As an example, the mixer could be used to combine a blue brick design with a large brick design to create a large blue brick design, without the designer needing to know what parameters control those properties.

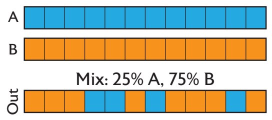
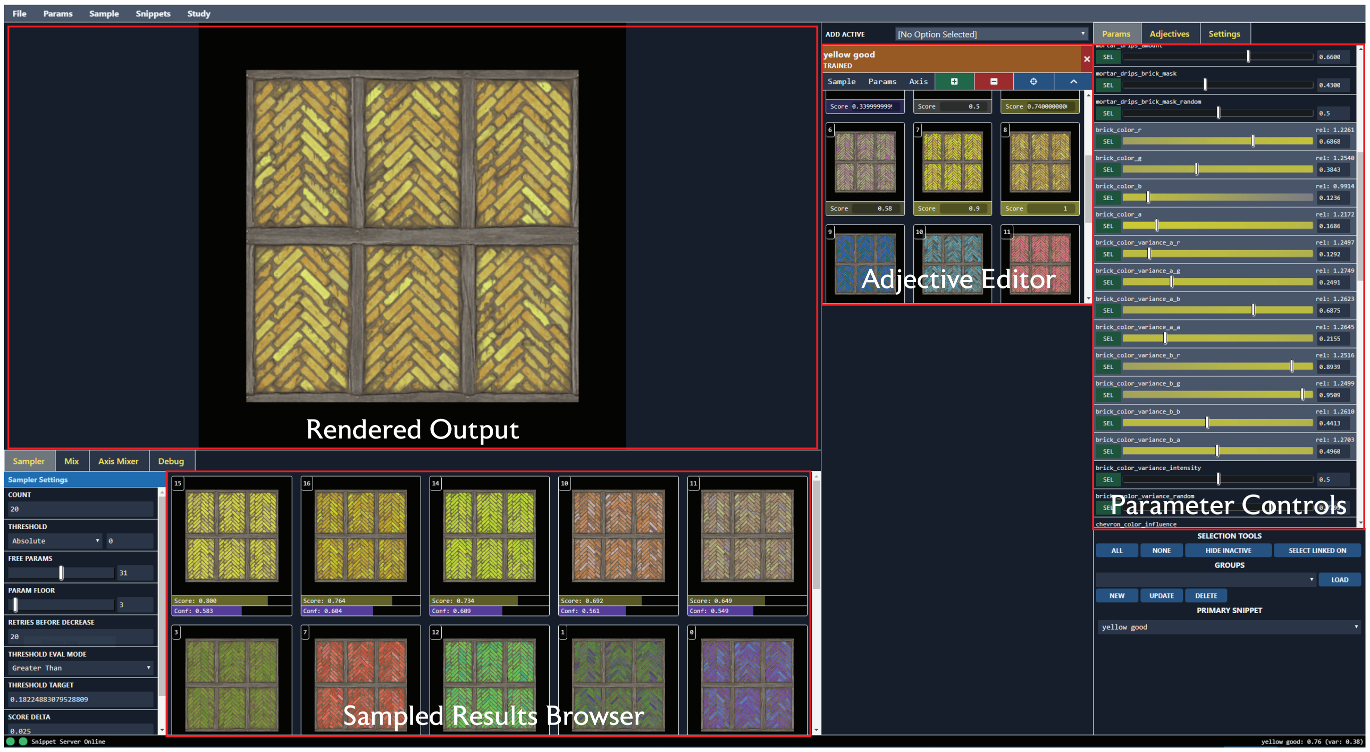


Figure 6. ***Mixer*****. One run of the mixer combining designs A and B, with a bias towards design B.**

Technical Implementation Details

We implement the design adjectives framework using a generic client-server interface. The server runs the training of the adjective model using GPyTorch [13], and the client renders designs and provides the requisite UI for interacting with the adjectives. The server does not have knowledge of any specific design domain, operating on the parameters assuming they are bounded within $[0,1]$. The client performs the normalization step before sending data to the server, and discrete parameters are made continuous by mapping the enumerated options to equal-sized intervals within $[0,1]$. This is a common way of performing this type of transformation, but does create visual discontinuities along the discrete parameter's range.

The server's adjective model training process favors training speed over accuracy, and the number of optimization steps taken is tuned such that the training completes in under a second (400 iterations on our hardware). Training a GPR is known to scale poorly as the size of the training set increases,



COUNT: 20
THRESHOLD: Absolute 0
FREE PARAMS: 31
PARAM FLOOR: 3
RETRIES BEFORE DECREASE: 20
THRESHOLD EVAL MODE: Greater Than
THRESHOLD TARGET: 0.18224883079528809
SCORE DELTA: 0.025

mortar_drips_brick_mask		0.4300
mortar_drips_brick_mask_random		0.5
brick_color_r	rel: 1.2261	0.6868
brick_color_g	rel: 1.2540	0.3843
brick_color_b	rel: 0.9914	0.1236
brick_color_a	rel: 1.2172	0.1686
brick_color_variance_a_r	rel: 1.2497	0.1292
brick_color_variance_a_g	rel: 1.2749	0.2491
brick_color_variance_a_b	rel: 1.2623	0.6875
brick_color_variance_a_a	rel: 1.2645	0.2155
brick_color_variance_b_r	rel: 1.2516	0.8939
brick_color_variance_b_g	rel: 1.2499	0.9509
brick_color_variance_b_b	rel: 1.2610	0.4413
brick_color_variance_b_a	rel: 1.2703	0.4960
brick_color_variance_intensity		0.5

PRIMARY SNIPPET: yellow good

yellow good: 0.76 (var: 0.38)

Figure 7. ***Design Adjectives User Interface*****. A screenshot of the design adjectives interface used in the evaluation of the system. The current design is rendered in the top-left. Design suggestions returned from sampling an adjective are shown in a gallery at the bottom, and hovering over a thumbnail renders the design in the main view. The adjective definition is viewable and editable in the right-center, with per-parameter controls always available on the right.**

typically running in $O(n^3)$ time. In our context, n is the number of input points, i.e., $n = |\mathcal{X}|$. Most adjectives defined in our implementation are well under the size at which this becomes a problem, however sparse Gaussian process regression algorithms can be used if needed [13]. Splitting the capabilities of the server and client in this way made it possible for us to implement three design domains (fonts, materials, particle systems) in the framework using the same generic software primitives.

The user interface (Figure 7) is written in Javascript using Electron [9] to provide a desktop application wrapper. New design domains can be added to the interface by implementing a driver interface, requiring the domain to provide a rendering function for full-quality renders and thumbnail previews, and requiring a list of parameters that can be modified in the domain. Rendering is typically performed on an HTML canvas element. Implementing the three domains shown in this paper in the interface took a few hours of development time each, with the variance coming primarily from how easily results could be rendered in the Electron-based UI. With enough development time, the server could be re-used in plugins developed for existing parameterized design interfaces. Client and server source code for our implementation can be found at https://github.com/ebshimizu/DesignAdjectives.

EVALUATION

To determine how effectively our implementation of the design adjectives framework supports exploratory design, we perform two studies: a study with intermediate users evaluating the tool in a series of short design tasks, and a case study evaluating the tool's ability to support expert workflows.

In the user study, we are primarily interested in determining the extent to which the design adjective models are perceived to accurately represent a design concept. We also investigate the extent to which the system is perceived by users to support exploration, and which parts of the toolkit are viewed as the most useful. We detail the process and results in the "User Study" section and find that our implementation is able to represent per-user design concepts, and that it supports exploratory design better than existing interfaces.

In the expert case study, we investigated the extent to which the design adjectives framework can support an expert-level design process. The tool was given to a professional graphic designer, who used it to create a series of fonts. The designer found that adjectives enabled them to perform design concepting easily and quickly. In an additional informal evaluation, an expert material designer spent a day with the tool to generate a set of brick material variations, and we detail the results of this design session later in this section.

User Study

In the user study, participants performed three design tasks: two using our implementation of the design adjectives framework, and one using a baseline sliders configuration. We collected user feedback via Likert scale questions measuring the extent to which users felt that each configuration helped

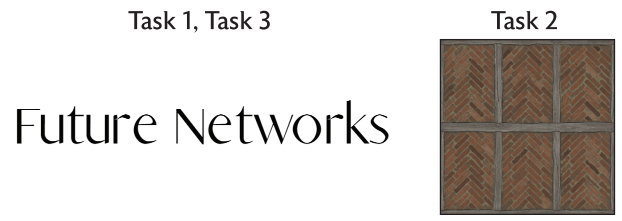


Figure 8. ***User Study Initial Configurations*****. The initial designs displayed to users performing the tasks in the user study. The font for Tasks 1 and 3 is shown on the left, and the material for Task 2 is shown on the right. The particle system configuration is omitted here because it is not used in an evaluated task, but is shown in the video accompanying this paper.**

them with the design task. Overall, we found that participants were able to use adjectives to easily explore the design space, and find solutions that satisfied the design task. Participants preferred to use the adjectives configuration over the baseline sliders-only configuration for exploratory tasks.

Participants
10 adult participants were recruited for this voluntary study. No compensation was given. All participants had at least used a computer graphics design tool (e.g. Autodesk Maya, Unity3D, Adobe Photoshop, etc.) before. Participants generally rated themselves as skilled users of design tools, but not daily users of these tools, with a median self-reported skill of 3 out of 5 on a scale ranging from "uses design tools once every few months" (1) to "uses design tools daily" (5). Users were not specifically knowledgeable about particle system design, material design, or font design.

Interface Configurations
Participants used the two following configurations.

Design Adjectives This configuration allows users to use our implementation of the design adjectives framework (Figure 7). When performing a task with this configuration, users were limited to using one self-defined adjective, mirroring the workflow described in the "Design Adjectives Overview" section. No pre-existing adjectives were provided.

Sliders This configuration only allows users to use the sliders, a baseline similar to existing state-of-the-art computer graphics tool interfaces. This configuration used the same interface shown in Figure 7 with all design adjectives-related functionality disabled.

Tasks
We used one design domain to teach users how to use the design adjectives configuration, and two design domains to evaluate the design adjectives configuration. Each participant in this study performed the same tasks in the same order, performing two tasks with the design adjectives configuration and then repeating the first task using the sliders configuration. By running tasks in this order, we can investigate to whether or not the design adjectives configuration helped users understand how the low-level parameters affect the design space.

Participants were first given a 20 minute tutorial, where they were taught how to use the tools present in the design adjectives configuration using a particle system. They were then given a series of three 10-minute design tasks where they were instructed to create a font or material according to a description of the design goal. Participants were allowed to finish their design tweaks at the end of the 10 minute time period, if they felt that it was necessary.

Participants answered a series of Likert-scale survey questions between each task, and the interface recorded a log of actions taken during each task. At the end of the study, participants answered three forced-choice questions asking them to choose one of the two configurations for use in general exploration tasks, directed exploration tasks, and detail design tasks. They were also allowed to give written feedback at the end of the study. The study took approximately 60 minutes to complete.

Particle System (50 parameters): Tutorial. A particle system simulation run by particles.js [14], an open-source library. This domain is only used for the tutorial task. The tutorial consisted of a fully user-driven exploration task, where participants were instructed to find a particle system that was interesting to them using the adjectives configuration while being taught about the interface capabilities.

Font (26 parameters): Tasks 1 and 3. Participants were asked to create a "futuristic" font for a made-up networking conference titled "Future Networks." Participants designed their font using the characters contained in the conference title. The parameters were provided by the Prototypo [31] font design system, using the "Elzevir" template as the base (Figure 8-left). Task 1 allowed users to use the design adjectives configuration, while Task 3 repeated the design task with the sliders configuration. Under these conditions, we expected that the repeat task would be easier, as users had previously worked in the design space.

Material (62 parameters): Task 2. Participants were asked to make the selected brick texture look more "weathered, cracked, old, or broken." The material was created using Substance3D [40] and rendered using the physically-based shader in the three.js library [28] (Figure 8-right). This configuration was used in Task 2.

User Study Results
We draw four main conclusions from analyzing the results of the user study:

- Adjectives support exploratory design,
- Users prefer to use the adjectives configuration for exploratory design tasks,
- Adjectives are the most commonly used component of the adjectives configuration, and
- Adjectives are used for directed exploration in particular.

Adjectives Support Exploratory Design
Participants felt that the design adjectives framework helped them with exploratory design, in that it was easy to explore the different design options in the design space, explore variations of their current design adjective, and the adjectives generated designs that were consistent with their expectations of the adjective (Figure 9). While most participants agreed that

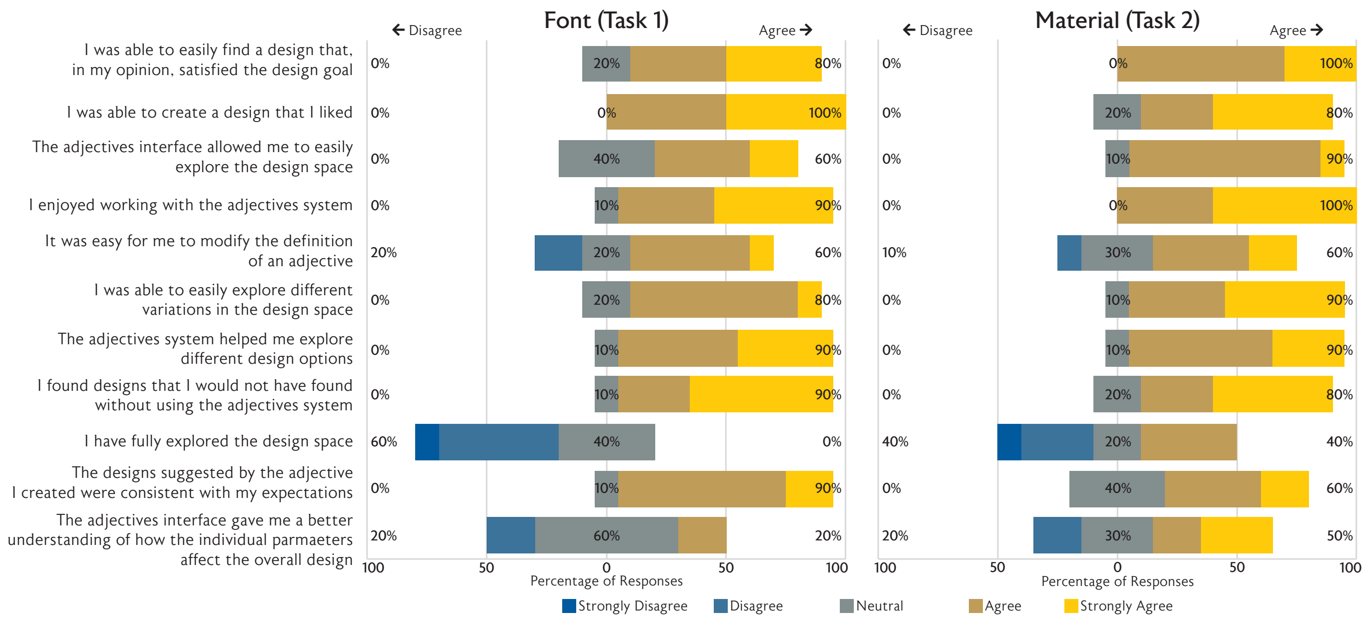


Figure 9. ***Adjectives Interface Survey Results*. Likert plots of the responses to the survey questions about the adjectives interface used in the font design task (task 1, left) and the material design (task 2, right). Responses are generally consistent between tasks using the adjectives configuration.**

the adjective definition was easy to modify, a few disagreed, possibly indicating that the initial examples provided to the adjective held too much weight. Participants used an average of 23 examples (13 positive, 10 negative) to define the "futuristic" adjective in Task 1, and 21 examples (11 positive, 10 negative) to define the "weathered" adjective in Task 2. In contrast to prior work [26, 12], participants expressed no difficulty in locating negative examples. This is likely due to the preliminary random sampler outputting a large number of clearly irrelevant samples that can easily be marked as negative at the start of the process.

Participants agreed that the adjectives configuration allowed them to find designs that they might not have found otherwise. This suggests that the design suggestion sampling method for adjectives is effective at providing suggestions that are different enough from the input examples. Another source of this variation comes from the global samplers, however, participants indicated that they found the adjective sampling to be more useful (median score of 5) to them than the general random sampling operations (median score of 4) (Table 1). Analysis of the interface logs shows that users ran 2 global sampling operations on average and 5 adjective sampling operations.

Participants felt that they were more easily able to explore different variations in the design space with the design adjectives configuration when compared to the baseline sliders configuration. Plotting the designs explored in the space during Tasks 1 and 3 with a PCA projection along the first two components shows that users did indeed end up exploring many more dissimilar designs while using the design adjectives interface (Figure 10). Some participants noted that some of the designs found while using the adjectives configuration were interesting but orthogonal to their current design goal and wanted to use multiple adjectives to keep track of those designs. This capability exists in the interface, but was restricted for this study.

Participants felt that adjectives configuration did not particularly help them understand how individual parameters affect the design space, however only half of the participants in the study ended up using the per-parameter controls (Table 1). Participants were still successful in creating a design that they liked with the adjectives configuration, indicating that the adjectives configuration can model user preferences accurately even without the user understanding how individual parameters affect the design.

The Adjectives Configuration is Preferred for Exploratory Design Tasks

Participants appeared to be more engaged with the adjectives configuration, reporting that they liked working with the adjectives configuration much more than the sliders configuration, while also spending less time with the sliders configuration task. Participants took 9.5 minutes to complete the font design task on average with the adjectives configuration and 6.3 minutes on average with the sliders configuration. The shorter completion time and the survey results in Figure 11 suggest that while users were able to complete all tasks in the study, they appeared to be more engaged with the adjectives configuration, to the point of voluntarily spending as much time as possible with the adjectives configuration. Some users expressed a strong dislike for the sliders configuration in written feedback, mentioning that "using the sliders was not an enjoyable experience. While I felt I did not have enough time with the adjectives interface, I could not wait to be done with the sliders task."

Participants were generally able to complete the task to their satisfaction, although a higher percentage of participants were dissatisfied with their final results in Task 3. Their dislike of

	Task 1			Task 2		
	Median Rating	% Used	Avg. Use Time	Median Rating	% Used	Avg. Use Time
Adjective Sampler	5	100 %	322 s	5	100 %	259 s
Global Samplers	4	100 %	125 s	4	90 %	148 s
Parameter Highlighting	4	50 %	53 s	4	50 %	152 s
Parameter Selection	4	40 %	45 s	5	80 %	51 s
Parameter Extents	4.5	20 %	115 s	5	60 %	105 s
Mixer	5	50 %	97 s	4	50 %	94 s

Table 1. ***Component Usefulness Scores*****. Summary of survey feedback regarding the usefulness of each adjectives interface component. Participants were asked to rate how useful each component was on a five-point scale. The rating column reports the median score from these questions, the "% Used" column reports the percentage of participants that used the component at least once during the trial, and the "Avg. Use Time" column reports the average amount of time spent using the component in seconds (see Figure 12 for when each tool was used). Ratings and times are only taken into account if the participant used the component (validated by the trial's action log).**

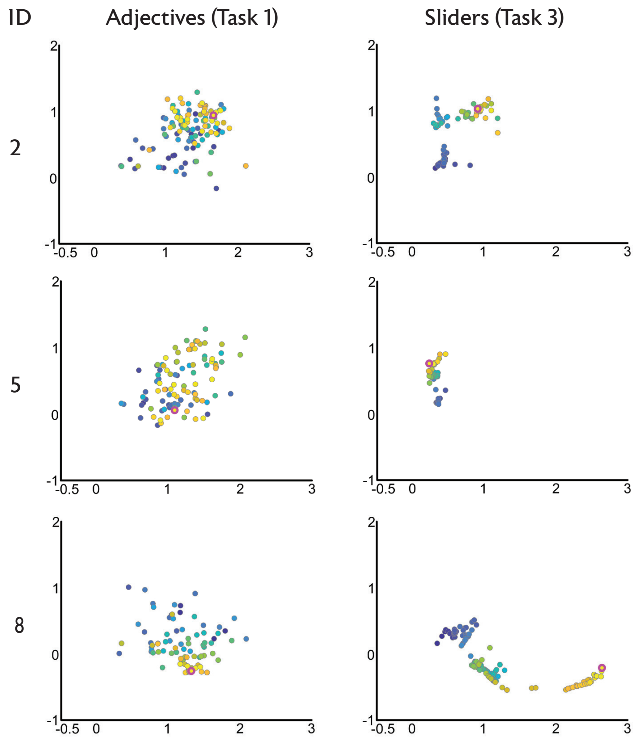


Figure 10. ***2D PCA Projections of User Design Sessions*****. Users tended to explore a more diverse set of designs while using the design adjectives configuration. These plots represent designs seen by participant IDs 2, 5, and 8 during design sessions for Tasks 1 and 3 projected into 2D along the first two PCA components. Color indicates when a design was seen during the process, with blue indicating earliest and yellow indicating latest. The pink outlined point is the final design chosen by the user. The same embedding is used for all plots.**

the sliders configuration was specific to its use for exploratory design tasks, noting that while the sliders limited the number of variations they felt they could explore, they did prefer to use the sliders in the detail design phase. One participant explained that "the slider interface was nice for fine control (somewhat slow), while adjectives was kind of fast. I think there could be a nice workflow going between these two configurations."

When asked to pick one interface to use for each part of the design phase in a forced choice comparison, users again indicated a strong preference for using the adjectives configuration over the sliders configuration for exploratory tasks, with all participants stating that they would use it for general exploration ($p \ll 0.01$), and all but one participant stating that they would use it for directed exploration ($p < 0.05$). For performing detail-oriented design, a majority of participants (7 out of 10) indicated that the sliders interface was more appropriate for small, low-level design changes, with only 3 out of 10 users preferring to use the adjectives configuration for that phase of the design process. This last result is only weakly significant ($p > 0.05$).

Adjectives are the Most Commonly Used Component

To better understand which parts of the interface contributed the most to the participant's evaluation of the adjectives configuration in Figure 9, each participant rated how useful they felt each component was, and we compare this data with how frequently each tool was used in the interface logs. Table 1 summarizes the results of this analysis, listing the per-task median score, the percentage of users who used the the component at least once during a task, and the average amount of time spent using the tool was during each task. We find that the adjective sampling was perceived as central to the adjectives configuration, as every participant used the adjective samplers at least once and rated it highly, while other framework components were seen as useful optional utilities. The bootstrapping tools (global random and jitter samplers) were also frequently used, but seen as less useful than the adjective samplers. The other components were used less frequently than both samplers, but were viewed favorably by the participants that did use those features.

The per-parameter features were rated highly, but not used as frequently as the sampling components. Many of these tools provide additional control over individual sliders, and some participants may have perceived these as unnecessary, as the adjectives alone were sufficient to accomplish their design goals. The usage rate for the per-parameter features (highlighting, selection, and extents) increased between Task 1 and Task 2. Since Task 2 contained more parameters than Task 1, many participants chose to perform operations that limited the number of active parameters (parameter selection). These tools, selection and viewing extents in particular, may be more useful as the number of parameters increases.

Adjectives are Used for Directed Exploration

To validate the claim that adjectives are primarily used for the exploratory phase of the design process, we plot the times at which participants used each interface component in Tasks

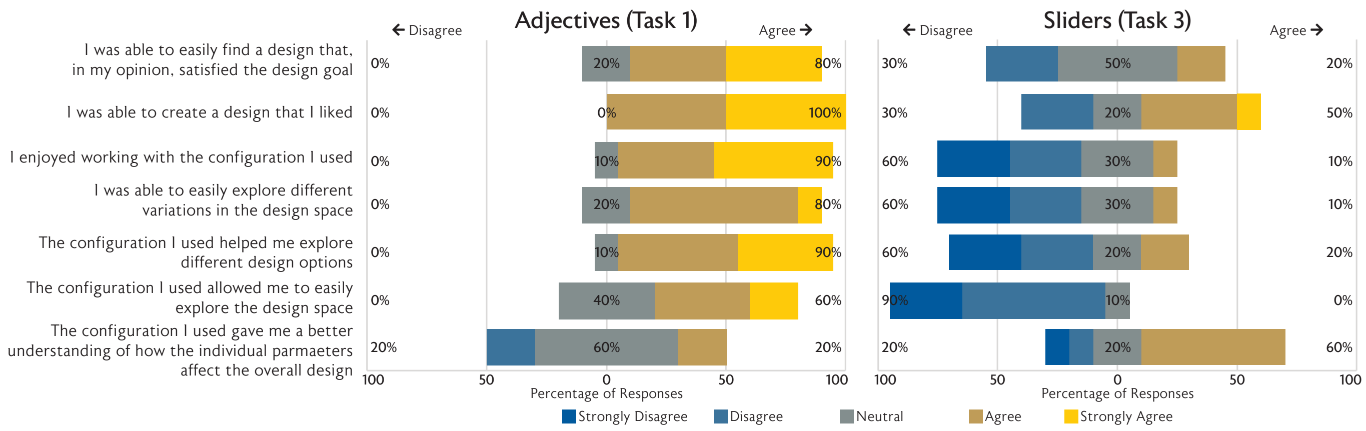


Figure 11. ***Adjectives vs. Sliders Interface Survey Results*****. Likert plots of the responses to the survey questions about the adjectives interface used in Tasks 1 and 3. Participants generally preferred to use the adjectives interface, despite Task 3 being a repeat of Task 1 with the sliders interface configuration.**

1 and 2, and for how long, in Figure 12. Each segment of a timeline marks when a major interface component has been invoked, and its width indicates the time until the next major component is used. The major components are the adjective sampler operations, global samplers (random and jitter), parameter selection, the mixer, and individual parameter changes (sliders). With one exception (user 4 in both tasks), the global samplers are all used first, followed by the adjective samplers, with a few selection operations sometimes happening before adjective use.

In both tasks, the activity traces reveal a design process that is closely aligned with the three conceptual phases of the design. Users generally performed preliminary design by using the global samplers, refined their design idea by using the adjective sampling methods, and often performed detail work with the individual parameter controls, extents view, or mixer components at the end. In general, we see usage of the design adjectives tools where we expected; comprising the bulk of the design process, used to refine and explore preliminary concepts. Participants used per-parameter controls less frequently in Task 1, possibly due to most parameter configurations outputting valid fonts. In contrast, Task 2's material had a larger possibility of outputting invalid designs, requiring users to use the parameter selection, extents, and direct controls more frequently.

Speed is Important

Speed is critical in interactive design applications, and the adjectives configuration is no exception. While the sampling operation for adjectives completed within a second, the prototype interface required a few second to render the returned design suggestions. A few users expressed frustration with this small delay, commenting that "The biggest thing that threw me off is the rendering time/lag," and that "it takes a while to render and thus its harder for me to see the differences between similar things." Interfaces implementing a design adjectives-style framework should ensure that the training and sampling of adjectives, and rendering of results, completes in real-time.

Professional Case Study

We asked a professional graphic designer to create a set of fonts using our implementation of the design adjectives framework. Many of this designer's projects involve the selection and editing of a font for a logo or package design. Their typical workflow involves the selection of a base font, followed by the generation of a number of design-specific adjustments to the base font, which are then presented to the client and further refined based on that feedback. The interface used to perform these adjustments is typically a glyph editor, where the font characters are edited directly.

The designer was given the design adjectives interface, five Prototypo [31] font templates (Antique, Elzevir, Fell, Grotesk, and Spectral), and a 15-minute tutorial. The font templates contained 26 parameters controlling different font properties, without requiring the use of a glyph editor. While the full Prototypo interface offered glyph-level control in addition to parameter-level control, re-implementing a glyph editor in our interface was out of scope for this project. They used the interface unsupervised over the course of one week and created the 15 fonts in Figure 13. When working with the templates, the designer set out to create distinct fonts within each template. Each font took an estimated 16 minutes on average to create, as reported by the designer, and are considered by the designer to be usable in professional work with a few glyph-specific tweaks.

Overall, the designer enjoyed working with the tool, specifically noting that the tool was especially effective at providing a framework for quick design concepting. Their primary workflow when using the tool consisted of creating a new adjective, then adding one positive (high rated) example, and one negative (low rated) example to an adjective. The designer would then perform a "towards" sampling operation, update the current design, and repeat the addition of one positive and one negative example to the adjective. The designer would repeat this refinement cycle until sampling "towards" stopped returning results. This process converged in three to four cycles and "almost always produced an adjective that was good enough to start detailed refinement." In the instances when this method was not used, the designer bootstrapped the adjectives by generating a set of random samples, and then used the mixer

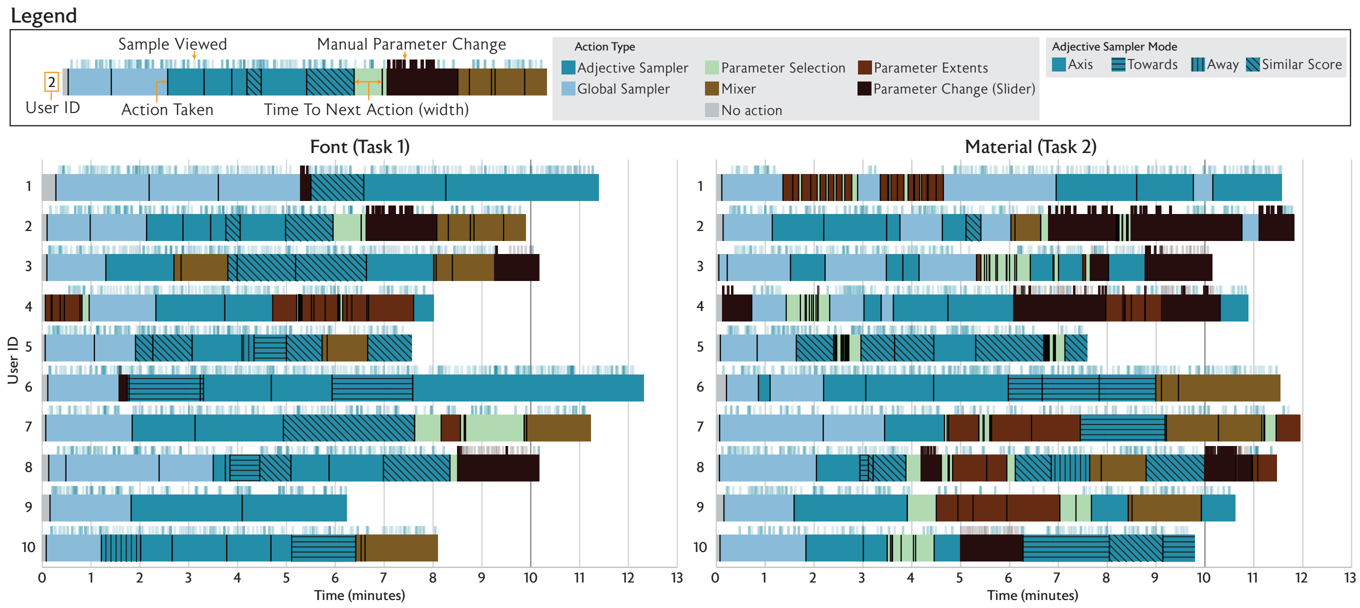


Figure 12. ***User Activity Trace by Interaction Type*****. Plot detailing when each user performing tasks 1 and 2 used an interface element belonging to one of the six component categories in the implementation. Each row of the plot displays one participant's actions over the course of the task. Each segment in a row represents when the participant activated an interface component, its width represents the time until they activated another component, and the color indicates the action type. The time between actions is typically when users inspect the results returned by invoking the interface action. The marks on top of each trace indicate when the user previewed a new design from a set of samples (blue) or directly manipulated the design with a slider (brown). The adjective sampling mode is indicated by the hatching pattern on the block.**

tool to create examples. The designer noted that the random samples would often have features that would be considered unreasonable by most design standards, such as distorted serifs, and that the implementation may want to impose some stricter bounds for the parameters in this space.

Analysis of the interface's recorded logs verify the designer's assertions. The logs show that the professional designer typically followed a usage pattern similar to that of the intermediate designers in Figure 12. They first bootstrapped the adjectives using the global sampler, ran the adjective sampler a few times, and finished designs with the mixer and per-parameter tools. This professional liked using the mixer, and the log shows frequent use of that tool at all parts of the design process.

One of the limitations of the tools in the designer's view was the lack of a sideboard, a panel in the interface that can store designs unrelated to any current adjectives for later use. To get around this limitation, the designer created a "sideboard" adjective, where the scores did not matter, to store these designs. The designer expressed some frustration with the tool's parameter selection capabilities, mentioning that it was difficult to find the controls for preventing a parameter value from changing during sampling. While this capability exists in the implementation, it was not presented clearly enough for the designer to make effective use of it. Fonts created in this interface could be exported to existing glyph editing tools for final adjustments, after the design concepts have been created. Parametric fonts such as those used in this case study already provide a high-level interface for modifying all of the font glyphs at once, without resorting to a low-level glyph editor. Even in this context, the designer felt that the adjectives framework helped them locate interesting variations much more quickly than directly manipulating parameters.

Material Design Session

A professional material designer spent a day experimenting with our implementation, creating and adjective and twenty representative variations of that adjective for a brick texture in Figure 14. The adjective (left unnamed by the designer) used for this design session included 160 labeled examples. Figure 14 includes visualizations of the axis and towards sampling acceptance criteria, and details an professional's thought process regarding how a design gets refined by using sampling results combined with the mixer interface component. High resolution images of the created design variations and the professional's process diagrams can be found in the supplemental material.

DISCUSSION

In this paper, we introduced design adjectives, models of user intent defined through interactive machine learning, and introduced a software framework that uses design adjectives to build interfaces that enable interactive guided exploration in parameterized design spaces. We found that when using tools built on top of this framework, users were more easily able to create designs that satisfied their personal objectives. We hope that the framework, and our domain-agnostic implementation of its components, serves as an inspiration and baseline for the creation of new component implementations and new design tools that can react in real time to the ever changing preferences of designers as they solve ill-structured design problems.

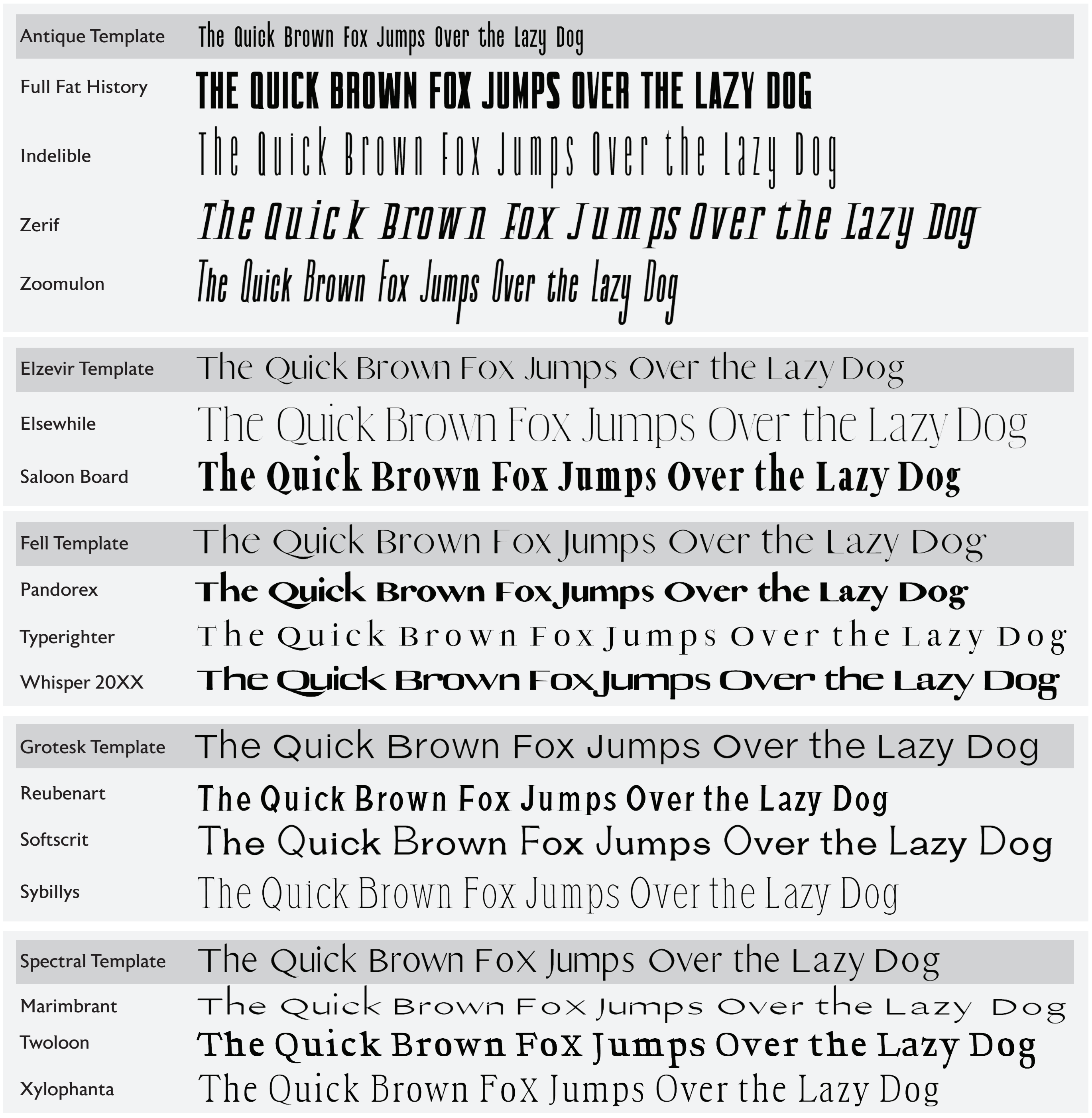

Antique Template — The Quick Brown Fox Jumps Over the Lazy Dog
Full Fat History — THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
Indelible — The Quick Brown Fox Jumps Over the Lazy Dog
Zerif — The Quick Brown Fox Jumps Over the Lazy Dog
Zoomulon — The Quick Brown Fox Jumps Over the Lazy Dog
Elzevir Template — The Quick Brown Fox Jumps Over the Lazy Dog
Elsewhile — The Quick Brown Fox Jumps Over the Lazy Dog
Saloon Board — The Quick Brown Fox Jumps Over the Lazy Dog
Fell Template — The Quick Brown Fox Jumps Over the Lazy Dog
Pandorex — The Quick Brown Fox Jumps Over the Lazy Dog
Typerighter — The Quick Brown Fox Jumps Over the Lazy Dog
Whisper 20XX — The Quick Brown Fox Jumps Over the Lazy Dog
Grotesk Template — The Quick Brown Fox Jumps Over the Lazy Dog
Reubenart — The Quick Brown Fox Jumps Over the Lazy Dog
Softscrit — The Quick Brown Fox Jumps Over the Lazy Dog
Sybillys — The Quick Brown Fox Jumps Over the Lazy Dog
Spectral Template — The Quick Brown Fox Jumps Over the Lazy Dog
Marimbrant — The Quick Brown Fox Jumps Over the Lazy Dog
Twoloon — The Quick Brown Fox Jumps Over the Lazy Dog
Xylophanta — The Quick Brown Fox Jumps Over the Lazy Dog

Figure 13. ***Professionally Created Fonts*****. 15 fonts created by a professional graphic designer using the design adjectives interface. Fonts are grouped by which Prototypo template was used to create them, with the designer chosen font name displayed on the left. The template font is shown at the top of each group.**

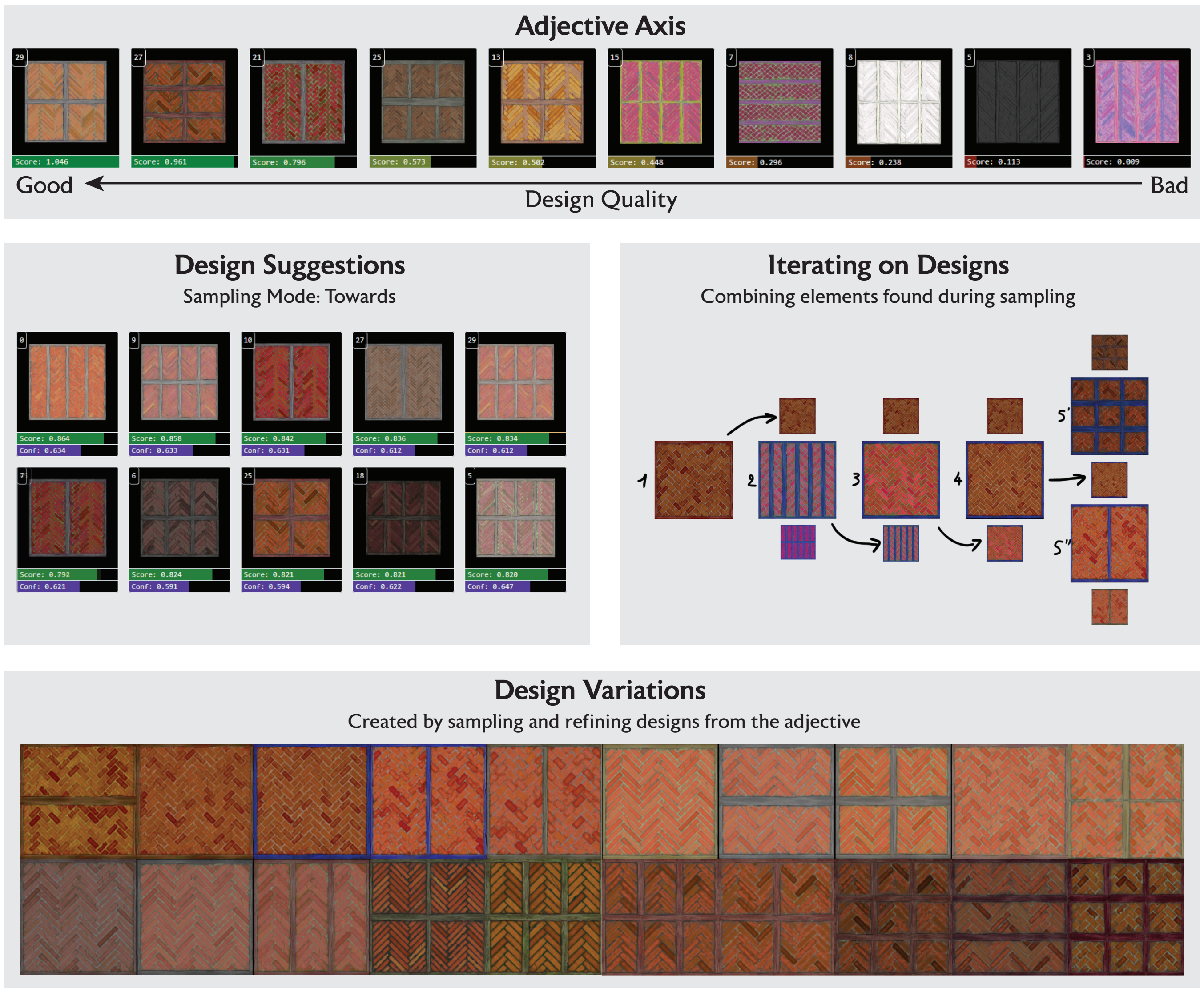


Figure 14. ***Professionally Created Materials*****. A visualization of a professional design session with the adjectives interface. The adjective definition is visualized by displaying the results of sampling using the Axis acceptance criteria (top). Samples generated by using the Towards criteria produce designs that are highly rated by the adjective (middle left). These designs can be iterated on by mixing together pairs of sampled designs (middle-right), where the large design is created from a combination of the two smaller designs on the top and bottom. A set of 20 designs created from this adjective is shown on the bottom.**

Limitations

Our implementation of a design adjective sampler requires at least one positive example in order for the guided sampler to run, and creating this initial positive example can be difficult. The global randomization bootstrapper works well enough for the design spaces used in the evaluation, but may struggle to provide valid examples in spaces where only a small number of parameter configurations produce a satisfactory initial design. The GPR formulation of adjectives permits use of more complex priors to help shape the adjective function, but creating these priors may require expert knowledge about a specific design space. For example, it may be possible to create classifiers that mark obviously bad designs in a space (e.g. for a material: overexposed, low contrast, etc.) that can be used by a bootstrapper to present only designs that pass this filter.

We chose to focus on workflows involving a single design adjective in order to demonstrate that adjectives can represent design concepts defined by one user. This is an important baseline to establish before adjectives are used as the basis for more complex interfaces. It is possible that a user might create a library of adjectives and wish to use multiple adjectives at the same time during a design task. Understanding such use cases was outside the scope of this paper.

Adjectives are only valid for the specific design domain they are learned on. For example, if an adjective is defined on a brick material created with Substance3D, the adjective cannot be used on a brick material that has a different parameterization. Since the design adjectives framework only operates on the low-level parameters, domain transfer of intent cannot be easily implemented in the current framework. It may be possible to create transferrable design adjectives if the adjective has knowledge of the visual properties of the design. This type of understanding may also enable better per-parameter identification tools, as the framework would have a way to recognize what parameters affect specific visual properties.

Future Work

We believe that the design adjectives framework opens exciting avenues for future work. Our approach could be applied to local editing which would enable a new take on the creation of complex structured artifacts, e.g., to manipulate the materials of a large 3D scene. The framework could be extended to a collaborative environment where multiple users who each have their own collections of adjectives could combine their adjectives into a library, creating opportunities to share information and combine adjectives. Finally, we also see a lot of potential between our framework and other design metaphors like sketching and image-based modeling, that could be used to create better bootstrappers for a design adjective. We hope that providing an implementation of this framework will make it easier to create and experiment with these future interfaces.

REFERENCES

[1] Saleema Amershi. 2012. *Designing for effective end-user interaction with machine learning*. Ph.D. Dissertation. University of Washington.

[2] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the People: The Role of Humans in Interactive Machine Learning. *AI Magazine* 35, 4 (Dec. 2014), 105. DOI: http://dx.doi.org/10.1609/aimag.v35i4.2513

[3] Autodesk Inc. 2020. Autodesk Maya. (2020). https://www.autodesk.com/products/maya/overview

[4] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. 1993. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques - SIGGRAPH '93*. ACM Press, 73–80. DOI: http://dx.doi.org/10.1145/166117.166126

[5] Eric Brochu, Tyson Brochu, and Nando de Freitas. 2010. A Bayesian Interactive Optimization Approach to Procedural Animation Design. (2010), 10.

[6] Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas Funkhouser. 2013. Attribit: content creation with semantic attributes. In *Proceedings of the 26th annual ACM symposium on User interface software and technology - UIST '13*. ACM Press, St. Andrews, Scotland, United Kingdom, 193–202. DOI: http://dx.doi.org/10.1145/2501988.2502008

[7] Minh Dang, Stefan Lienhard, Duygu Ceylan, Boris Neubert, Peter Wonka, and Mark Pauly. 2015. Interactive Design of Probability Density Functions for Shape Grammars. *ACM Trans. Graph.* 34, 6 (Oct. 2015), 206:1–206:13. DOI: http://dx.doi.org/10.1145/2816795.2818069

[8] Ruta Desai, Fraser Anderson, Justin Matejka, Stelian Coros, James McCann, George Fitzmaurice, and Tovi Grossman. 2019. Geppetto: Enabling Semantic Design of Expressive Robot Behaviors. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, 369:1–369:14. DOI: http://dx.doi.org/10.1145/3290605.3300599 event-place: Glasgow, Scotland Uk.

[9] Electron. 2020. Electron. (2020). https://www.electronjs.org/

[10] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. 2009. Describing objects by their attributes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 1778–1785. DOI: http://dx.doi.org/10.1109/CVPR.2009.5206772 ISSN: 1063-6919.

[11] Vittorio Ferrari and Andrew Zisserman. 2008. Learning Visual Attributes. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (Eds.). Curran Associates, Inc., 433–440. http://papers.nips.cc/paper/3217-learning-visual-attributes.pdf

[12] Martin R. Frank and James D. Foley. 1993. Model-based user interface design by example and by interview. In *Proceedings of the 6th annual ACM symposium on User interface software and technology (UIST '93)*. Association for Computing Machinery, Atlanta, Georgia, USA, 129–137. DOI: http://dx.doi.org/10.1145/168642.168655

[13] Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. 2018. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. *arXiv:1809.11165 [cs, stat]* (Sept. 2018). http://arxiv.org/abs/1809.11165 arXiv: 1809.11165.

[14] Vincent Garreau. 2019. particles.js. (Oct. 2019). https://github.com/ebshimizu/particles.js original-date: 2019-10-01T18:30:48Z.

[15] Vinod Goel and Peter Pirolli. 1992. The structure of Design Problem Spaces. *Cognitive Science* 16, 3 (1992), 395–429. DOI: http://dx.doi.org/10.1207/s15516709cog1603_3

[16] William B. Kerr and Fabio Pellacini. 2010. Toward evaluating material design interface paradigms for novice users. *ACM Transactions on Graphics* 29, 4 (July 2010), 1. DOI: http://dx.doi.org/10.1145/1778765.1778772

[17] Sandeep Kochhar. 1990. A prototype system for design automation via the browsing paradigm. In *Proceedings of Graphics Interface '90*, Vol. Halifax. 156–166. DOI: http://dx.doi.org/10.20380/gi1990.19

[18] Adriana Kovashka, Devi Parikh, and Kristen Grauman. 2015. WhittleSearch: Interactive Image Search with Relative Attribute Feedback. *International Journal of Computer Vision* 115, 2 (Nov. 2015), 185–210. DOI: http://dx.doi.org/10.1007/s11263-015-0814-0 arXiv: 1505.04141.

[19] Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2014. Crowd-powered parameter analysis for visual design exploration. In *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*. ACM Press, Honolulu, Hawaii, USA, 65–74. DOI: http://dx.doi.org/10.1145/2642918.2647386

[20] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. 2014. Transient attributes for high-level understanding and editing of outdoor scenes. In *ACM Transactions on Graphics*, Vol. 33. 1–11. DOI: http://dx.doi.org/10.1145/2601097.2601101

[21] Bryan Lawson. 2006. *How Designers Think: The Design Process Demystified*. Elsevier/Architectural.

[22] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. Designing with interactive example galleries. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, Atlanta, Georgia, USA, 2257. DOI:http://dx.doi.org/10.1145/1753326.1753667

[23] J. Marks, W. Ruml, K. Ryall, J. Seims, S. Shieber, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, and H. Pfister. 1997. Design galleries: a general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97*. ACM Press, 389–400. DOI: http://dx.doi.org/10.1145/258734.258887

[24] Justin Matejka, Michael Glueck, Erin Bradner, Ali Hashemi, Tovi Grossman, and George Fitzmaurice. 2018. Dream Lens: Exploration and Visualization of Large-Scale Generative Design Datasets. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, 369:1–369:12. DOI: http://dx.doi.org/10.1145/3173574.3173943 event-place: Montreal QC, Canada.

[25] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. 2003. A Data-driven Reflectance Model. In *ACM SIGGRAPH 2003 Papers (SIGGRAPH '03)*. ACM, New York, NY, USA, 759–769. DOI: http://dx.doi.org/10.1145/1201775.882343 event-place: San Diego, California.

[26] Richard G. McDaniel and Brad A. Myers. 1998. Building applications using only demonstration. In *Proceedings of the 3rd international conference on Intelligent user interfaces (IUI '98)*. Association for Computing Machinery, San Francisco, California, USA, 109–116. DOI: http://dx.doi.org/10.1145/268389.268409

[27] Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. 2015. Interactive Design of 3D-printable Robotic Creatures. *ACM Trans. Graph.* 34, 6 (Oct. 2015), 216:1–216:9. DOI: http://dx.doi.org/10.1145/2816795.2818137

[28] Mr.doob. 2019. three.js. (Nov. 2019). https://github.com/mrdoob/three.js original-date: 2010-03-23T18:58:01Z.

[29] Peter O'Donovan, Jānis Lībeks, Aseem Agarwala, and Aaron Hertzmann. 2014. Exploratory font selection using crowdsourced attributes. *ACM Transactions on Graphics* 33, 4 (July 2014), 1–9. DOI: http://dx.doi.org/10.1145/2601097.2601110

[30] Devi Parikh and Kristen Grauman. 2011. Relative attributes. In *2011 International Conference on Computer Vision*. 503–510. DOI: http://dx.doi.org/10.1109/ICCV.2011.6126281 ISSN: 1550-5499.

[31] Prototypo. 2019. Prototypo. (2019). https://www.prototypo.io/

[32] Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian processes for machine learning*. MIT Press, Cambridge, Mass. OCLC: ocm61285753.

[33] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. 2011. d.tour: style-based exploration of design example galleries. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, Santa Barbara, California, USA, 165. DOI: http://dx.doi.org/10.1145/2047196.2047216

[34] Adriana Schulz, Jie Xu, Bo Zhu, Changxi Zheng, Eitan Grinspun, and Wojciech Matusik. 2017. Interactive Design Space Exploration and Optimization for CAD Models. *ACM Trans. Graph.* 36, 4 (July 2017), 157:1–157:14. DOI: http://dx.doi.org/10.1145/3072959.3073688

[35] Evan Shimizu, Matt Fisher, Sylvain Paris, and Kayvon Fatahalian. 2019. Finding Layers Using Hover Visualizations. *Proceedings of Graphics Interface 2019* Kingston (2019), 28–31 May 2019. DOI: http://dx.doi.org/10.20380/gi2019.16

[36] Ben Shneiderman, Gerhard Fischer, Mary Czerwinski, Mitch Resnick, Brad Myers, Linda Candy, Ernest Edmonds, Mike Eisenberg, Elisa Giaccardi, Tom Hewett, Pamela Jennings, Bill Kules, Kumiyo Nakakoji, Jay Nunamaker, Randy Pausch, Ted Selker, Elisabeth Sylvan, and Michael Terry. 2005. Creativity Support Tools: Report From a U.S. National Science Foundation Sponsored Workshop. (Sept. 2005). http://www.cs.umd.edu/hcil/CST/report.html

[37] Maria Shugrina, Ariel Shamir, and Wojciech Matusik. 2015. Fab forms: customizable objects for fabrication with validity and geometry caching. *ACM Transactions on Graphics* 34, 4 (July 2015), 100:1–100:12. DOI: http://dx.doi.org/10.1145/2766994

[38] Leonid Sigal, Moshe Mahler, Spencer Diaz, Kyna McIntosh, Elizabeth Carter, Timothy Richards, and Jessica Hodgins. 2015. A perceptual control space for garment simulation. *ACM Transactions on Graphics* 34, 4 (July 2015), 117:1–117:10. DOI: http://dx.doi.org/10.1145/2766971

[39] Herbert A. Simon. 1973. The structure of ill structured problems. *Artificial Intelligence* 4, 3-4 (1973), 181–201. DOI:http://dx.doi.org/10.1016/0004-3702(73)90011-8

[40] Substance3D. 2019. Substance Designer. (Jan. 2019). https://www..substance3d.com

[41] Adobe Systems. 2019. Adobe Photoshop Lightroom CC. (2019). https://www.adobe.com/products/photoshop-lightroom.html

[42] Jerry O. Talton, Daniel Gibson, Lingfeng Yang, Pat Hanrahan, and Vladlen Koltun. 2009. Exploratory modeling with collaborative design spaces. *ACM Transactions on Graphics* 28, 5 (Dec. 2009), 1. DOI: http://dx.doi.org/10.1145/1618452.1618513

[43] Unity Technologies. 2019. Unity3D. (2019). https://unity3d.com/

[44] Michael Terry and Elizabeth D. Mynatt. 2002. Side views: persistent, on-demand previews for open-ended tasks. In *Proceedings of the 15th annual ACM symposium on User interface software and technology - UIST '02*. ACM Press, Paris, France, 71. DOI: http://dx.doi.org/10.1145/571985.571996

[45] David G. Ullman, Thomas G. Dietterich, and Larry A. Stauffer. 1988. A model of the mechanical design process based on empirical data. *AI EDAM* 2, 1 (Feb. 1988), 33–52. DOI: http://dx.doi.org/10.1017/S0890060400000536

[46] Nobuyuki Umetani, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. 2011. Sensitive Couture for Interactive Garment Modeling and Editing. In *ACM SIGGRAPH 2011 Papers (SIGGRAPH '11)*. ACM, New York, NY, USA, 90:1–90:12. DOI: http://dx.doi.org/10.1145/1964921.1964985 event-place: Vancouver, British Columbia, Canada.

[47] Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. 2014. Pteromys: Interactive Design and Optimization of Free-formed Free-flight Model Airplanes. *ACM Trans. Graph.* 33, 4 (July 2014), 65:1–65:10. DOI: http://dx.doi.org/10.1145/2601097.2601129

[48] Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K. Hodgins, and Levent Burak Kara. 2015. Semantic shape editing using deformation handles. *ACM Transactions on Graphics* 34, 4 (July 2015), 86:1–86:12. DOI: http://dx.doi.org/10.1145/2766908

[49] Károly Zsolnai-Fehér, Peter Wonka, and Michael Wimmer. 2018. Gaussian material synthesis. *ACM Transactions on Graphics* 37, 4 (July 2018), 1–14. DOI: http://dx.doi.org/10.1145/3197517.3201307