Visual Lyrics: Automatic Generation of Animated Lyric Videos

Anonymous Author(s)

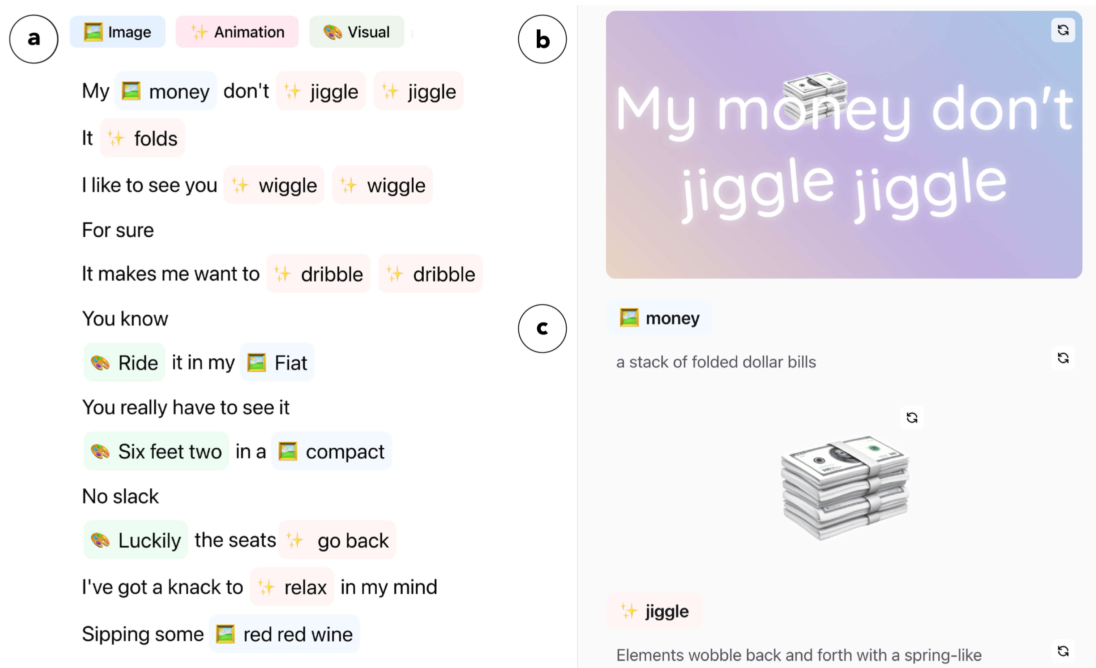


Figure 1: Visual Lyrics interface. The system analyzes a song's audio and language features to suggest words that can be highlighted with image, animation, or visual stylizations. On the Annotation Panel (left), suggestions appear as annotations over lyrics (a). The user can edit the annotations to steer the creative direction of the lyric video. The Generation Panel (right) displays generated animated scenes for each line of lyrics (b). The user can see intermediate LLM instructions for creating the images, animations, and visuals (c). The user can regenerate new instructions or edit them manually for finegrained control.

ABSTRACT

Animated lyric videos transform song lyrics into dynamic visual experiences, offering a powerful medium for artistic expression and audience engagement. However, creating these videos is challenging, requiring expertise in audio, typography, graphic design, and animation, making it inaccessible to novices. To address this challenge, we introduce Visual Lyrics, an automatic animated lyric video generation system with an intuitive text-driven interface for creative control. We examined existing lyric videos to distill a taxonomy and design guidelines, informing the design of Visual Lyrics. Our key insight is a multimodal music analysis pipeline based on the taxonomy and leveraging LLM's strong natural language understanding and code generation capabilities to synthesize creative and semantically meaningful animations. We collected a dataset of over 300 code-driven creative text animations to serve as inspiration for our LLM-driven pipeline, which we open source. In a user study, Visual Lyrics enabled novices to easily create high-quality animated lyric videos with high ratings of enjoyment, inspiration, and exploration.

UIST '25, September 28-October 1, 2025, Busan, Korea

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS CONCEPTS

• **Human-centered computing → Interactive systems and tools**; **Human computer interaction (HCI)**.

ACM Reference Format:
Anonymous Author(s). 2025. Visual Lyrics: Automatic Generation of Animated Lyric Videos. In *The 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25), September 28-October 1, 2025, Busan, Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

An animated lyric video transforms song lyrics into dynamic visual experiences, serving as a powerful medium for artistic expression and audience engagement. These videos have steadily grown in popularity, driven by music artists and content creators on platforms like YouTube and TikTok.

However, creating these animations remains a challenging endeavor requiring expertise in multiple domains, such as audio, typography, graphic design, and animation. Current approaches often involve manual animation in tools like After Effects [1], which is time-consuming (often spending hours to create a few seconds of animation) and requires significant technical skill to achieve quality results, making it inaccessible to many content creators.

Many creators attempt to streamline this process through templates or preset animations [3, 4], but these solutions often produce generic results that miss the opportunity to capture the semantics of the lyrics and emotions of the song. This is important because while basic karaoke-style videos provide functional text synchronization, there is an opportunity to create much richer and engaging videos that exhibit more creativity and aesthetic beauty.

Prior work has explored automated approaches for more constrained subtasks like overlaying static lyrics on videos [30] or animating static vector graphics [29, 39], but the challenge of automatically generating expressive animated lyrics for entire songs end-to-end remains largely underexplored. Our key insight is that LLMs have an increasingly strong code generation capability [17]. By leveraging LLMs' understanding of natural language and their code generation abilities, we can create creative and semantically meaningful animations that amplify musical lyrics.

In this paper, we present Visual Lyrics, an end-to-end pipeline for automatically generating animated lyric videos. We begin by examining existing lyric videos to distill a taxonomy of common stylization effects in lyric videos and establish three design guidelines that inform the development of our system. Given a song, Visual Lyrics analyzes it to identify language and audio features based on our taxonomy, and generates matching code-driven animations using HTML, CSS, and JavaScript. Visual Lyrics breaks down the complex task of animated lyric video creation into three stages: *Planning*, which determines which words to add stylization effects to and what types of effects to use; *Generation*, which involves conceptualizing the overall theme, creating image assets, designing static layouts, and animating those layouts; and *Validation*, which implements feedback loops to ensure that each stage of the generation process produces high-quality results. To enhance the generation pipeline, we collected a dataset of code-driven creative text animations for retrieval-augmented generation, which we open-source. In a user study, Visual Lyrics enabled novice users to create high-quality animated lyric videos with high self-reported ratings of enjoyment, inspiration, and exploration.

This research thus contributes:

- A **taxonomy** of stylization effects in animated lyric videos.
- **Visual Lyrics**, an end-to-end animated lyric video generation system with an intuitive text-driven interface for creative control.
- A **dataset** of 306 code-driven, creative text animations.
- A **user study** demonstrating the utility of Visual Lyrics for novice creators and informing the development of future animated lyric creation tools.

2 RELATED WORK

This work draws on prior research in automatic music video generation, kinetic typography, and generative animation.

2.1 Automatic Music Video Generation

Researchers have explored the automatic generation of videos to accompany music, enhancing the listening experience through adding a visual component. Many works focus on adding images based on the lyrics. MusicStory [34] and Cai et al. [7] extracted salient words (e.g., nouns) and queried online image repositories. To establish visual coherence, Shin et al. [35] aligned visual content with the song's emotional tones. In this work, we create a visually coherent concept for the entire lyric video, though going beyond static images by also styling and animating the text displayed on screen.

Another thread of research in this space addresses the technical challenge of aligning lyrics with audio. Fujihara et al. [13] developed an automatic lyrics-to-audio synchronization system and Goto et al. [14] created the Songle platform for crowdsourcing lyric alignment. Recently, Ma et al. [30] introduced one of the first end-to-end pipelines to automatically convert music videos (without lyrics) into lyric videos. Most closely related to our work is TextAlive by Kato et al. [20], which is among the first tools to offer interactive authoring of lyric videos, animate text in sync with music. Kato and Goto's Lyric App [19] further provided environments for crafting lyric-driven visuals. Both systems primarily rely on manual authoring by the user and have a relatively high learning curve for novice users.

Our work builds on Kato et al.'s efforts with a focus on developing a fully automatic pipeline to support novice creators. We leverage the strong natural language understanding and code generation capabilities of LLMs, enabling users to describe animations in natural language and automatically synthesize flexible and creative animations beyond predefined motion algorithms through code.

2.2 Kinetic Typography

Kinetic typography is a motion graphics technique where text is animated to convey emotion, narrative, and emphasis beyond static words. Early HCI researchers recognized its expressive power and began developing tools to support authoring it. ActiveText [24] is one of the first systems for authoring dynamic text, demonstrating how text motion can enhance communication. Lee et al. developed the Kinetic Typography Engine [23], which brought film-like visual expressiveness to text. Its follow-up work, Kinedit [12], enabled animators to apply presets for text motion in order to convey affect in messages. However, these early systems were largely manual, requiring designers to handcraft animations.

With the increasing popularity of video content, commercial tools like Adobe Express [4] and Canva Magic Animate [3] offer limited preset effects for animating content. Recent works by Liwenhan et al., including Creating Emordle [40] and Wakey-Wakey [41], have explored automated methods for animating words based on design heuristics and by mimicking character motions, taking into account the emotional qualities of words.

Building on previous insights from kinetic typography research, our work draws on the expressive power of words to convey narrative and emotional affect through an automated pipeline specifically designed for lyric videos. Our approach considers both the audio and language channels of music to generate animated kinetic typography.

2.3 Generative Animation

Beyond text, our work connects to the broader field of generative animation, which involves methods for producing moving visuals from high-level inputs (e.g., sketching, text, or code). Early sketch-based systems like K-Sketch [9] and Draco [21] allowed users to sketch motion paths for objects, enabling algorithmic approaches that bring static illustrations to life using kinetic textures and user-guided input.

Artists and researchers have long experimented with using code to generate visuals of unique styles. For example, Processing [33] in the 2000s has enabled creative custom animations through programming. The generated animations are highly flexible, as they can include any arbitrary visual or animation effect defined by rules or code. However, they typically lack automatic planning, requiring users to manually create generative rules. Being unable to interpret songs or lyrics, early music visualizers or demo-scene animators often reacted mainly to audio amplitude or beats, less on lyrical content or higher-level music structures. The challenge for our work is to combine the flexibility of code-driven animation generation with an automated understanding of a song's key features, using code to generate animations that follow the semantics of lyrics and vocals to create meaningful visual experiences.

Over the past year, several works have shown that LLMs are capable of generating code for rendering animations, as seen in works like Keyframer [39] and LogoMotion [29]. However, these explorations focus on more constrained tasks, such as animating static vector graphics (Keyframer) and logos (LogoMotion). In this work, we explore generating visually cohesive sequences of expressive kinetic text effects, images, and animations for entire songs, end-to-end.

3 DESIGN GOALS

We follow the methodology by Agrawala et al. [5] to identify guidelines by examining animated lyric video tutorials and existing examples of animated lyric videos. Our analysis included 20 tutorials, featuring those from tool creators such as Adobe (the developer of After Effects), as well as from various artists. We examined the top 100 animated lyric videos sourced from YouTube using keywords like "animated lyric video," "kinetic typography video," and "motion lyric video." Researchers manually filtered out videos of low quality. From this analysis, we distilled three design goals to inform the development of Visual Lyrics. These guiding design goals include analyzing both the language and audio channels of the music, supporting a wide range of stylizations, and maintaining the readability of the text.

3.1 Taxonomy

From our analysis of existing animated lyric videos, we developed a taxonomy of different types of words where editors commonly add special stylizations to the lyrics (see Table 1). We categorized them into three types: Image, Visual, and Animation. Image refers to instances where editors add an additional supporting graphic to the video, such as identifying visually-concrete objects or metaphors that can be associated with objects. Visual involves editors applying font stylizations to the words, such as stylized font choice, font size, and font color. This is often used for words related to size, color, emotional qualities, or depending on the energy of the vocals (sung particularly loudly or quietly). Animation refers to instances where editors animate the word itself, such as words related to motion or words sung with special vocal attributes like upwards or downwards pitch shift, word elongation, and vibrato. From our observations, we identified that some stylizations are based on the text (language features), while others are based on the vocals (audio features).

3.2 Design Goal 1: Analyze Audio and Language

A creative animated lyric video should identify interesting opportunities to add special stylizations to words. As illustrated in Table 1, these opportunities can arise from either the lyrical aspects (language features) or the vocal elements (audio features) of the song. Current animated lyric authoring tools predominantly focus on the language aspect (see Section 2.2). For instance, some tools identify visually concrete words [34] or specific words with emotive attributes [35]. In this work, we build on our identified language and audio features from our taxonomy to develop a multimodal analysis pipeline (see Section 4.2).

3.3 Design Goal 2: Support Diverse Stylizations

From reviewing past videos, we observed that the implementation of creative stylization effects can span a broad range of techniques. These include altering the visual appearance of words in various ways, applying custom animations, or creating new supporting images (Table 1). Current tools primarily support preset effects or focus on a single type of stylization, such as matching images (Section 2.1), animating text (Section 2.2), or editing the visual attributes of text [38]. In this work, we harness the rich flexibility of code (e.g., CSS, JavaScript) to synthesize a diverse variety of stylizations (see Section 4.3). To increase the quality of the code-implemented stylization effects, we sourced a dataset of over 300 code-driven text animation effects (see Section 4.3.4).

3.4 Design Goal 3: Maintain Readability

While stylizing words with creative and expressive visuals and animations are appealing, ensuring the legibility of the text remains crucial, and achieving a balance between the two is essential. Current tools largely overlook this aspect, requiring users to manually identify and correct readability errors. In this work, we implement validations to automatically detect potential readability issues at various stages of the generation process (see Section 4.4).

4 VISUAL LYRICS

We begin by illustrating how a user might use Visual Lyrics through an example. Following this, we describe the technical implementation of Visual Lyrics, which consists of three primary components:

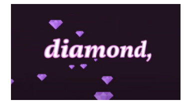
Word Type	Modality	Description	Common Effect	Example
Visual	Language	Visually-concrete objects, such as `diamond` or `heart`.	Image	
Metaphor	Language	Abstract concepts corresponding to metaphorical objects, such as airplane for `farewell` or steel for `strength`.	Image	
Size	Language	Size-related words, such as `big` or `tiny`.	Font size	
Color	Language	Color-related words, such as `gold` or `red`.	Font color	
Emotion	Language	Words with emotional attributes, such as `sweet` or `sad`.	Font family or color	
Energy	Audio	Words sung by the singer with a louder/smaller volume.	Font size	
Motion	Language	Motion-related words, such as `shake` or `bounce`.	Semantic animations	
Pitch shift	Audio	Words sung by the singer with a pitch shift upwards/downwards.	Vertical movement or trail animations	
Elongation	Audio	Words sung by the singer with an elongated emphasis.	Stretched or repeated animations	
Vibrato	Audio	Words sung by the singer with a vibrato.	Pulsing or distortion animations	

Table 1: Taxonomy of Word Stylizations in Animated Lyric Videos. We categorize ten different types of word stylizations across language and audio modalities, showing how the semantic properties of words and vocals can be visually represented through different properties (Image , Visual , and Animation). Each category includes a description of the word type, the visual effect that is commonly used, and an example usage.

Planning, Generation, and Validation. Planning involves preprocessing the music and analyzing multimodal aspects of the vocals and lyrics to extract relevant features. Generation includes conceptualizing the overall theme, creating image assets, generating static layouts, and animating these layouts. Validation encompasses quality checks of outputs across the different stages of the generation pipeline. To enhance the code generation output, we collected a dataset of 306 creative text animation code snippets for retrieval augmented generation, which we open source.

4.1 System Walkthrough

Taylor is a content creator who wants to create an animated lyric video for her friend's song: "Jiggle Jiggle" [18].

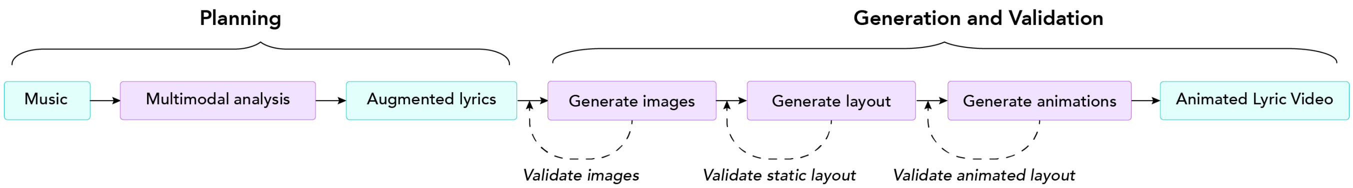


Figure 2: System Overview. The pipeline consists of *Planning* (music preprocessing and multimodal analysis to produce augmented lyrics), *Generation* (conceptualizing overall theme, creating image assets, designing static layouts, and animating layouts), and *Validation* (feedback loops for quality checks at different stages of Generation) to produce the final animated lyric video.

4.1.1 Annotating Lyrics. To begin, Taylor opens the Visual Lyrics interface and sees the *Annotation Panel* on the left, where the song's lyrics are transcribed line-by-line (Figure 1a). She notices that the lyrics are automatically annotated with three types of annotations: Image, Animation, and Visual. These annotations are generated by the system after analyzing both the song's language and audio features.

A word with an Image annotation suggests that she could add an image to the animated lyric video to visualize the word. For example, "`car`" → generate a car image. A word with an Animation annotation suggests that she could animate the word itself to emphasize it. For example, "`jiggle`" → apply a "jiggling" animation. A word with a Visual annotation suggests that she could apply visual stylizations to the word's font attributes. For example, "`red`" → change the word's color to red.

Taylor then customizes the annotations to tailor to her creative preferences by adding or removing annotations for different words.

4.1.2 Generating Animations. On the right, Taylor sees the *Generation Panel*. She notices that the panel is divided into sections, with each section corresponding to a line of the lyrics. Each section contains the animated scenes generated for each line of lyrics (Figure 1b) and also includes generated instructions on how the system implements the stylization effects for each annotated word in the scene (Figure 1c).

For example, Taylor notices that for the word "`money`", which she has annotated with an Image annotation, the system has created an image generation prompt: "`a single dollar bill`" and has also generated an image below it. Instead of a single dollar bill, Taylor wants a larger pile of money, so she manually edits the textbox to read "`a huge stack of dollar bills`" and regenerates the image by clicking the image's regenerate button.

In addition, Taylor notices that for the word "`jiggle`", which she has annotated with an Animation annotation, the system has suggested an LLM-generated instruction for implementing it. However, she doesn't like it very much. She wants to try something different and clicks on the instruction's regenerate button. She ends up liking the suggestion: "`hop randomly in place as if on a hot surface`".

Taylor reviews the various scenes corresponding to different lyric lines to finetune the stylizations according to her preferences. She does this by either regenerating instructions and images (akin to pulling a slot machine) or manually editing (to exert her own creative input). She frequently switches between interface panels, sometimes returning to the Annotation Panel to modify the annotations, which are then reflected in the Generation Panel for more precise edits. Additionally, she plays the entire video to evaluate how the animated results look all together and synchronized with the music. Here is an example of what Taylor's final animated lyric video could look like: Taylor's version.

4.2 Planning

We first separate vocal and non-vocals in the music using the Spleeter model [15]. We then transcribe the lyrics on the isolated vocal track using WhisperX [6]. This transcription serves as the foundation for two parallel annotation processes: audio annotations and language annotations.

4.2.1 Audio Annotations. For audio annotations, we analyze the audio characteristics of both the entire song and each word.

At the song level, we compute the average beats per minute (BPM) and the average energy level. We determine BPM using onset detection with a Butterworth low-pass filter [37] to reduce noise, then apply peak analysis to identify beats. We determine the song's average energy by computing the Root Mean Square (RMS) energy with overlapping windows of 2048 samples and a hop length of 512 samples [31].

At the word level, we identify four categories of special words based on their audio properties:

- High/low energy words: Words that have an RMS energy above/below a threshold.
- Upward/downward pitch-shifted words: Words with large upward/download shifts in pitch. We first identify the fundamental frequency with the YIN algorithm [10] using windows of 1024 samples and a hop length of 256 samples, then detect words with upwards/downward pitch shifts between start and end of words above a threshold.
- Elongated words: Words with long sustained energies. We first compute the RMS energy in windows of 512 samples with 128 sample overlap, then identify continuous sequences of windows with an energy above a threshold that spans over 30% of the word's duration.
- Vibrato words: Words with oscillating frequencies. We first count pitch oscillations, then check if the oscillation frequency is in the range of 4Hz to 8Hz (i.e., typical vocal vibrato frequency [11]).

Figure 3: Example results for different types of annotations. Top row shows Image annotations with generated supporting images (`fiat` with car image, `for sure` with thumbs up image, `no slack` with an image of a pair of shoes). Middle row shows Animation annotations with dynamic text animations (`jiggle jiggle` jiggling, `back` spinning backwards, `it folds` being folded). Bottom row shows Visual annotations with creative typography (`red red` in red color, `six feet two` in tall compact font, `relax` with a faded color gradient). More animated examples here.

For each identified special word, we map it to a visual or animation effect (Figure 1a). Specifically, high/low energy words are emphasized with big/small text (visual word), pitch-shifted words are emphasized with growing/shrinking animation (animation word), elongated words are emphasized with stretching animation (animation word), and vibrato words are emphasized with oscillating growing and shrinking animation (animation word).

4.2.2 Language Annotations. For language annotations, we use LLMs to identify three categories of special words from the lyrics (Table 1):

- Image words: Visually-concrete words that can be visualized as physical objects (e.g., "sun" or "flower") or metaphorical concepts that can be visualized (e.g., steel for "strength" or rose for "love").
- Animation words: Words related to motion (e.g., "jump" or "spin" or objects strongly associated with movement (e.g., "waves" or "arrow").
- Visual words: Words that can be enhanced through font attributes, including color (e.g., "blue" or "dark"), size (e.g., "big" or "tiny"), and emotional qualities that can be conveyed through font choice or color (e.g., "happy" or "elegant").

We call the annotated lyric transcript the "augmented lyrics". For each identified special word in the augmented lyrics, the LLM then generates either a description on how to implement the creative effect in HTML, CSS, and JavaScript (for animation and visual words) or generates an image generation prompt that can be used by a text-to-image model to generate a supporting image (for image words) (Figure 1c).

4.3 Generation

Creating an animated lyric video consists of many components, including conceptualizing an overall theme, creating image assets, organizing text and images into layouts, and adding dynamic animations to each element. In Visual Lyrics, we create a separate "agent" for each task, including the Creator Director, the Illustrator, the Layout Designer, and the Animator.

4.3.1 Conceptualizing Overall Theme. The Creative Director agent establishes a theme specification for the entire animated lyric video to ensure a consistent visual style and animation pace across animated scenes. The Creative Director uses an LLM to take in the computed song-level audio features (BPM and average energy level) and the complete song lyrics as input and generates an overall mood description, color scheme (using HEX values), typography (using

Google Fonts), animation style description, and background style description.

4.3.2 Creating Image Assets. The Illustrator agent generates images for words marked with image annotations. The Illustrator generates images using the FLUX.1 Schnell text-to-image model [22] with Low-Rank Adaptation (LoRA) finetuning [16] on Apple emoji designs. The FLUX.1 Schnell model is capable of generating high-quality images with fast performance using only 4 steps. The LoRA finetune allows FLUX.1 Schnell to generate emoji-style designs suitable for animated lyric videos with minimal prompt engineering. The Illustrator then removes the backgrounds of the generated images using ViTMatte [42].

4.3.3 Designing Static Layouts. The Layout Designer agent generates static layouts for each line of the lyrics (i.e., each line is a scene in the animated lyric video). Given the augmented lyrics, the theme specification, and the generated image assets, the Layout Designer uses an LLM to generate a layout with HTML/CSS code, the Layout Designer uses an LLM to generate a layout with HTML and CSS code. It is worth noting that we chose to generate code instead of asking the LLM to compose a layout using bounding-box coordinates [27]. In our early implementations, we found that LLMs have limited ability to generate correct numeric values for positioning, which often results in layouts with misalignment and overlap issues. Instead, HTML/CSS's relative positioning and built-in responsive layout system proved to be more robust. In addition, we can use code to apply complex animation effects that involve depth and physics properties to these layouts (Section 4.3.5).

4.3.4 Dataset. Inspired by prior research in retrieval augmented generation [26], we enhance the quality of the LLM-generated code-driven text animations providing the LLM with a collection of high-quality examples, handcrafted by designers, to serve as inspiration. We collected 306 text animation code snippets from CodePen [2], an online community for sharing code snippets. These snippets were sourced from public "pens" and were selected based on their implementation using HTML, CSS, and JavaScript. The selection process was manually curated by the researchers. We searched public pens using keywords such as "text effects", "text animation", and "CSS text." Overall, the collected code snippets are diverse and encompass a wide range of custom animations and visual stylizations (see Figure 4). Some of the most frequently appearing keywords in their titles include "shadow", "3D", "glitch", "neon", and "gradient". Among collected code snippets, some focus more on visual stylizations of static text, such as neon glow retro style text, Lego-like 3D text, and metallic texture text. Others focus on the animation of texts, such as text with liquid physics-like behavior, disappearing text mimicking smoke, and text that appears to be written with handwriting. The full dataset can be viewed here.

4.3.5 Adding Animations. The Animator agent adds animations to the static elements and generates subtle animated background elements. Given a static layout, the augmented lyrics, and the theme specification, the Animator uses an LLM to add animation effects using HTML, CSS, and JavaScript code. In addition, the Animator generates subtle decorative elements for the background, such as animated gradients, 3D particles, and geometric shapes.

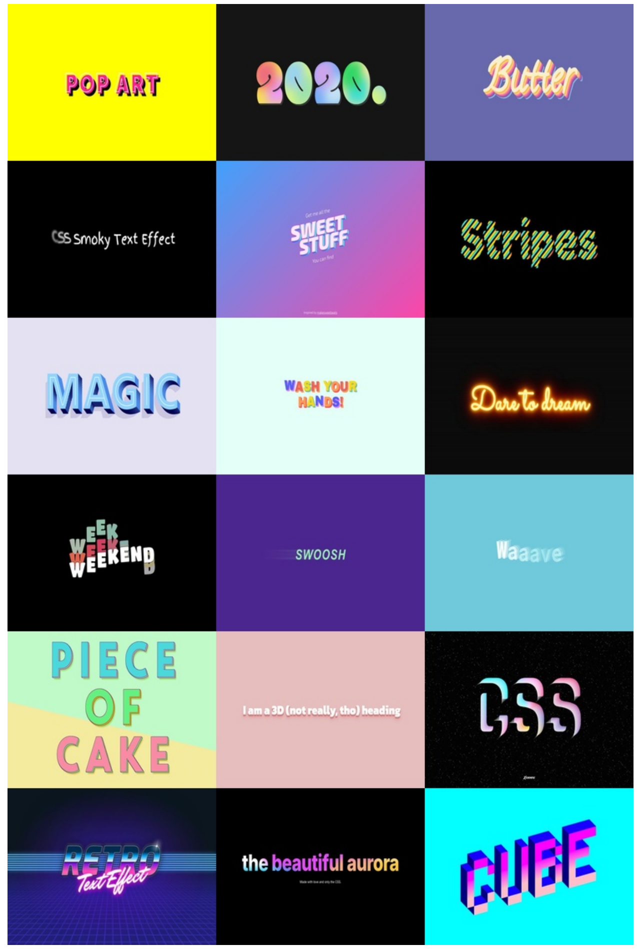


Figure 4: Examples from our dataset of 306 creative text animation code snippets collected from CodePen. See full dataset here.

4.4 Validation

Throughout the Generation pipeline, we validate the outputs of each agent with the Creative Director agent with feedback loops.

4.4.1 Illustrator Validation. The Creative Director first uses LLaVA [28], a state-of-the-art captioning model, to caption the images. The Creative Director, then embeds both the original image generation prompt and the LLaVA-generated caption using CLIP [32] and computes their cosine similarities. CLIP encodes text into semantic embeddings. If the similarity is below a threshold, the Illustrator is asked to regenerate a new image.

4.4.2 Layout Designer Validation. The Creative Director validates the static layouts through heuristics that check for the following constraints:

- All elements must be within visible bounds.
- All text must be readable (via OCR [36]) and not occluded by other elements.

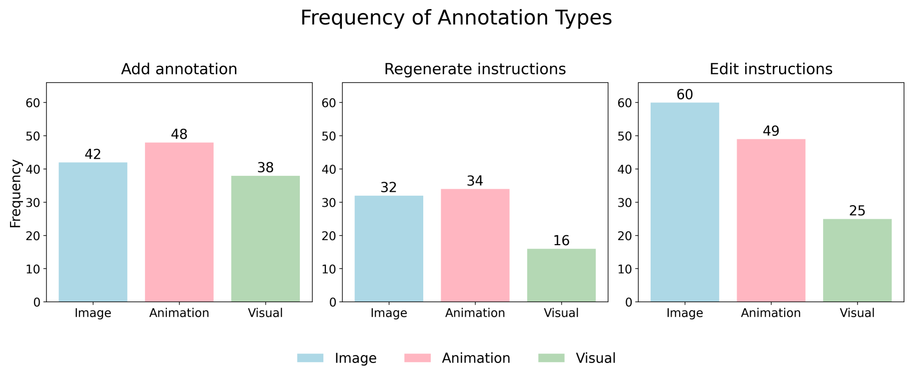


Figure 5: Usage Frequency of Annotation Types. All three types of annotations were added onto the Annotation Panel relatively equally. Users generally regenerated and manually edited Image and Animation annotations more.

- Images should be appropriately sized (maximum 80% of container height).

If the constraints are not satisfied, the Layout Designer is asked to correct the layout.

4.4.3 Animator Validation. The Creative Director validates the animated layouts through heuristics that check for the following constraints:

- All elements must be within visible bounds.
- There should not be any non-renderable animation code.
- All text must be readable during the animations (via OCR [36]).
- Images should have subtle animations (not static).

If the constraints are not satisfied, the Animator is asked to correct the animations

After the Creative Director approves the final animated layouts, we obtain a sequence of animated lyrics that follows a cohesive theme. Overall, Visual Lyrics generates creative animated lyric videos with complementary images, word stylization, and dynamic animation, driven by both audio and lyrical analysis, while maintaining a consistent visual concept. Figure 3 show some examples created with Visual Lyrics (see animated examples here).

5 USER STUDY

We conducted a user study to understand how Visual Lyrics could support novice creators in making animated lyric videos, its potential to be integrated into their personal workflows, and identify improvement areas.

5.1 Participants

We invited ten participants (P1-P10, 7 female and 3 male, aged 18 to 38) to participate in a one-hour user study. Participants were recruited through postings on Slack channels at our institution and by word-of-mouth. They had no prior exposure to the Visual Lyrics system or concept before the study. The participants were novice creators familiar with watching animated lyric videos (self-rated familiarity μ=4.00, σ=1.05 on a 5-point Likert scale from 1=low familiarity to 5=high familiarity) but less familiar with creating them (self-rated familiarity μ=1.80, σ=1.48). During the study, participants accessed Visual Lyrics through a web browser, shared their screens, and verbally explained their actions and thoughts (think-aloud).

5.2 Measures

We asked participants to complete questionnaires to capture their perceptions of creativity and usability while using Visual Lyrics. We assessed creativity using the Creativity Support Index (CSI) [8], which measures enjoyment, inspiration, exploration, expressiveness, immersiveness, and effort/reward trade-off. Usability was assessed using the System Usability Scale (SUS) [25], which evaluates perceived confidence, ease of learning, quality of integration between different components, ease of use, and likelihood of frequent use. Additionally, we asked participants to rate the satisfaction of their overall usage experience and the quality of the final results they created. All questionnaire items were rated on a 5-point Likert scale (5=strongly agree, 1=strongly disagree). Furthermore, we logged user interaction data, including when participants added or removed annotations, regenerated stylization instructions, and manually edited stylization instructions.

5.3 Procedure

5.3.1 Introduction (10 minutes). Participants provided informed consent, completed a background questionnaire, and then received an introduction to Visual Lyrics, as described in Section 4.

5.3.2 Reproduction Task (15 minutes). Participants were asked to create an animated lyric video for the song "Jiggle Jiggle," as described in Section 4.1. They were given a brief that guided them through the various components of the system to create the video.

5.3.3 Free Creation Task (20 minutes). Participants were asked to freely explore Visual Lyrics and create an animated lyric video. They could use their own song or choose from a selection of twelve songs encompassing various artists and genres, such as pop, R&B, K-pop, and rap.

5.3.4 Post-Study (15 minutes). Participants completed questionnaires that assessed their perceived sense of creativity, usability,

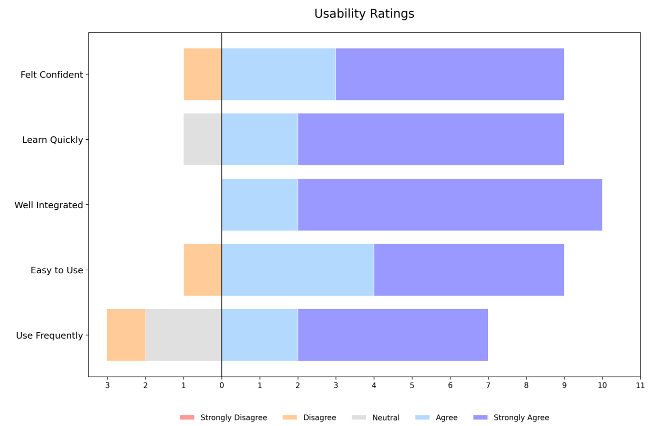


Figure 6: Usability ratings measured with SUS [25].

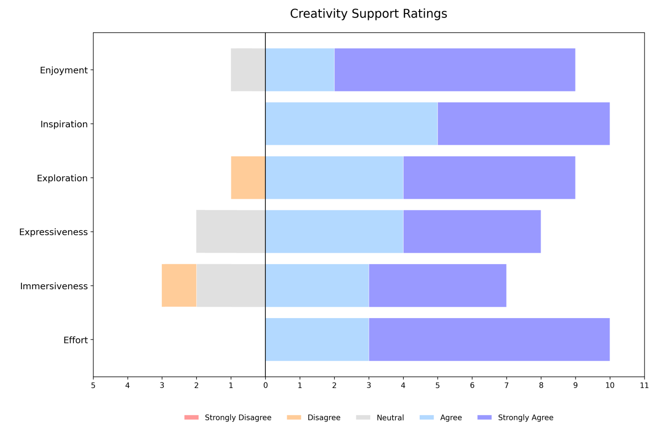


Figure 7: Creativity support ratings measured with CSI [8].

overall usage experience, and the quality of their final creation (see Section 5.2). Additionally, participants completed a free-response questionnaire asking about their overall experience of using Visual Lyrics, whether they could see Visual Lyrics being integrated into their personal workflows, and areas for improving the system.

5.4 Results, Discussion, and Future Work

All participants completed the reproduction and free creation tasks. Participants were generally satisfied with the overall usage experience (μ=4.40, σ=0.52, 5-point Likert Scale) and with the final animations they created (μ=4.50, σ=0.71). Users suggested areas for future improvements. Example user creations can be viewed here.

5.4.1 Helping Novice Users Create Quality Animations. Participants generally reported high ratings for usability measured with the System Usability Scale, including the quality of integration between the tool's different components (μ=4.80, σ=0.42, 5-point Likert Scale), ease of learning (μ=4.60, σ=0.70), and ease of use (μ=4.30, σ=0.95). **Overall, participants expressed that they were able to create high quality animations with little manual effort**: "I spent maybe a minute deciding [which] words to highlight in the chorus… and the tool was good at getting creative animations that probably would have taken me hours in CapCut (P2)".

Participants commented that the dual interface design (Annotation Panel and Generation Panel) supported a natural workflow: "[I could use] the left panel for choosing what to emphasize and the right panel for refining how those emphasizations looked (P8)". In particular, participants commented how the tool helped streamline the typically highly technical and fragmented process: "The technical barrier to entry for animation is usually so high… I don't have to worry about finding compatible fonts, designing graphics, or how to keyframe specific motion effects (P2)". For improvement, P3 suggested allowing users to edit the automatically generated theme specifications. Similarly, P9 wished to be able to edit the color theme selections.

The validation mechanisms were valuable for the novice participants *(Design Goal 3)*. P10 observed that "the built-in validation [was] like having an expert designer looking over my shoulder", when the system adjusted animations to prevent the text from being off-screen. P2 noted, "When I added too many image elements to one video, [the tool automatically] adjusted their sizes and positioning to make sure that the text remained the main focus". For future extension, P5 suggested giving users control over the placement of different elements in the animated scene, as well as the ability to specify image movement, such as by sketching a motion path.

5.4.2 Supporting Flexible Prototyping of Diverse Stylizations. Participants generally reported high ratings for creativity measured with the Creativity Support Index, including enjoyment (μ=4.60, σ=0.70), inspiration (μ=4.50, σ=0.53), exploration (μ=4.30, σ=0.95), and effort/reward tradeoff (μ=4.70, σ=0.48). **Overall, participants expressed that the tool helped them quickly explore a diverse range of animations and often inspired them with new ideas** *(Design Goal 2)*: "I was able to switch between [annotation] types for the same word and quickly visualize completely different results… fire as a flame image, as a red-orangish gradient text style, as a flickering animation effect… I found myself deliberately experimenting just to see the possibilities of each approach (P7)". To enable greater flexibility, P2 suggested the ability to apply multiple types of annotations to the same word for future work. Figure 5 shows a relatively equal distribution among the different added annotation types, which may suggest that participants found value in all types of stylizations, rather than relying solely on one type.

Participants were generally more satisfied with the automatically generated visual stylizations, resulting in fewer regenerations and less manual editing (Figure 5). We observed that participants most commonly edited image instructions to add specificity, such as changing "`gold coin`" to "`a shimmering stack of gold coins`". On the other hand, they primarily made animation edits to modify intensity, such as changing "`bouncing up and down`" to "`bouncing gently up and down`". Our interaction logs show the tool supporting different working styles among participants. Some participants focused on first annotating the lyrics, then diving into fine-grained editing (P2, P6, P8). Other participants had regular alternations between annotating and editing (P5, P7, P10).

Participants felt that the automatically suggested annotations were helpful in "overcoming a blank canvas (P10)": "I like how the lyric is auto-scanned and words are annotated already so there's some sort of example to start from (P3)". In particular, the audio-based annotation suggestions were appreciated by participants, who noted that they led to unexpected discoveries *(Design Goal 1)*, such as the "oscillating animation of the word `'heart'` matching the oscillating voice of the singer [vibrato] (P9)". P5 commented that "this first layer of suggestions was helpful for scaffolding... allowing [them] to begin the creation process confidently".

6 CONCLUSION

This research presents Visual Lyrics, an end-to-end system for automatic animated lyric video generation. Visual Lyrics adopts a multimodal song analysis pipeline designed around a taxonomy of stylization effects and leverages LLMs' strong code generation capabilities to create dynamic and semantically matching animations. Feedback from novice creators using Visual Lyrics demonstrated that the tool helped them create high-quality animations with low manual effort, and they were able to use it to quickly explore a diverse range of animations, often being inspired with new ideas.

REFERENCES

[1] 2025. Adobe After Effects. https://www.adobe.com/products/aftereffects.html
[2] 2025. The best place to build, test, and discover front-end code. https://codepen.io/
[3] 2025. Bring your designs to life with Magic Animate. https://www.canva.com/pro/animator/
[4] 2025. Design Made Easy - Adobe Express. https://www.adobe.com/express/
[5] Maneesh Agrawala, Wilmot Li, and Floraine Berthouzoz. 2011. Design principles for visual communication. *Commun. ACM* 54, 4 (2011), 60–69.
[6] Max Bain, Jaesung Huh, Tengda Han, and Andrew Zisserman. 2023. Whisperx: Time-accurate speech transcription of long-form audio. *arXiv preprint arXiv:2303.00747* (2023).
[7] Rui Cai, Lei Zhang, Feng Jing, Wei Lai, and Wei-Ying Ma. 2007. Automated music video generation using web image resource. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, Vol. 2. IEEE, II–737.
[8] Erin Cherry and Celine Latulipe. 2014. Quantifying the creativity support of digital tools through the creativity support index. *ACM Transactions on Computer-Human Interaction (TOCHI)* 21, 4 (2014), 1–25.
[9] Richard C Davis, Brien Colwell, and James A Landay. 2008. K-sketch: a'kinetic'sketch pad for novice animators. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 413–422.
[10] Alain De Cheveigné and Hideki Kawahara. 2002. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America* 111, 4 (2002), 1917–1930.
[11] Fitton Music. 2025. Vibrato. https://www.fittonmusic.com/writing/noise/filtering/vibrato.html.
[12] Jodi Forlizzi, Johnny Lee, and Scott Hudson. 2003. The kinedit system: affective messages using dynamic texts. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 377–384.
[13] Hiromasa Fujihara, Masataka Goto, Jun Ogata, and Hiroshi G Okuno. 2011. LyricSynchronizer: Automatic synchronization system between musical audio signals and lyrics. *IEEE Journal of Selected Topics in Signal Processing* 5, 6 (2011), 1252–1261.
[14] Masataka Goto, Kazuyoshi Yoshii, Hiromasa Fujihara, Matthias Mauch, and Tomoyasu Nakano. 2011. Songle: A Web Service for Active Music Listening Improved by User Contributions.. In *ISMIR*. Citeseer, 311–316.
[15] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. 2020. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software* 5, 50 (2020), 2154.
[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).
[17] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515* (2024).
[18] Duke & Jones and Louis Theroux. 2022. Jiggle Jiggle. https://genius.com/Duke-and-jones-and-louis-theroux-jiggle-jiggle-lyrics.
[19] Jun Kato and Masataka Goto. 2023. Lyric app framework: A web-based framework for developing interactive lyric-driven musical applications. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–18.
[20] Jun Kato, Tomoyasu Nakano, and Masataka Goto. 2015. TextAlive: Integrated design environment for kinetic typography. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 3403–3412.
[21] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014. Draco: bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 351–360.
[22] Black Forest Labs. 2024. FLUX. https://github.com/black-forest-labs/flux.
[23] Johnny C Lee, Jodi Forlizzi, and Scott E Hudson. 2002. The kinetic typography engine: an extensible system for animating expressive text. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*. 81–90.
[24] Jason E Lewis and Alex Weyers. 1999. ActiveText: a method for creating dynamic and interactive texts. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*. 131–140.
[25] James R Lewis. 2018. The system usability scale: past, present, and future. *International Journal of Human–Computer Interaction* 34, 7 (2018), 577–590.
[26] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.
[27] Jiawei Lin, Jiaqi Guo, Shizhao Sun, Zijiang Yang, Jian-Guang Lou, and Dongmei Zhang. 2023. Layoutprompter: awaken the design ability of large language models. *Advances in Neural Information Processing Systems* 36 (2023), 43852–43879.
[28] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 26296–26306.
[29] Vivian Liu, Rubaiat Habib Kazi, Li-Yi Wei, Matthew Fisher, Timothy Langlois, Seth Walker, and Lydia Chilton. 2024. LogoMotion: Visually Grounded Code Generation for Content-Aware Animation. *arXiv preprint arXiv:2405.07065* (2024).
[30] Jiaju Ma, Anyi Rao, Li-Yi Wei, Rubaiat Habib Kazi, Hijung Valentina Shin, and Maneesh Agrawala. 2023. Automated Conversion of Music Videos into Lyric Videos. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–11.
[31] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and music signal analysis in python. *SciPy* 2015 (2015), 18–24.
[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.
[33] Casey Reas and Ben Fry. 2006. Processing: programming for the media arts. *Ai & Society* 20 (2006), 526–538.
[34] David A Shamma, Bryan Pardo, and Kristian J Hammond. 2005. Musicstory: a personalized music video creator. In *Proceedings of the 13th annual ACM International Conference on Multimedia*. 563–566.
[35] Ki-Ho Shin, Hye-Rin Kim, and In-Kwon Lee. 2016. Automated music video generation using emotion synchronization. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 002594–002597.
[36] Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, Vol. 2. IEEE, 629–633.
[37] Soundation. 2025. Butterworth Filter - The low pass filter. https://soundation.com/audio-effects/butterworth-filter.
[38] Maham Tanveer, Yizhi Wang, Ali Mahdavi-Amiri, and Hao Zhang. 2023. Ds-fusion: Artistic typography via discriminated and stylized diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 374–384.
[39] Tiffany Tseng, Ruijia Cheng, and Jeffrey Nichols. 2024. Keyframer: Empowering animation design using large language models. *arXiv preprint arXiv:2402.06071* (2024).
[40] Liwenhan Xie, Xinhuan Shu, Jeon Cheol Su, Yun Wang, Siming Chen, and Huamin Qu. 2023. Creating emordle: Animating word cloud for emotion expression. *IEEE Transactions on Visualization and Computer Graphics* (2023).
[41] Liwenhan Xie, Zhaoyu Zhou, Kerun Yu, Yun Wang, Huamin Qu, and Siming Chen. 2023. Wakey-wakey: Animate text by mimicking characters in a gif. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–14.
[42] Jingfeng Yao, Xinggang Wang, Shusheng Yang, and Baoyuan Wang. 2024. Vitmatte: Boosting image matting with pre-trained plain vision transformers. *Information Fusion* 103 (2024), 102091.

Figure 8

Figure 9