**KINETIC TYPOGRAPHY CREATOR MODULE**

|  |  |  |
| --- | --- | --- |
| **Author** | **Creation Date** | **Last Edit** |
| Omer Sharon | 30/03/2025 | 30/03/2025 |

**Overall Architecture Design**

The script is organized into **five modular components**, each with a dedicated responsibility:

1. **config.py**  
   Stores **global constants** and configuration variables such as:
   * Video dimensions, font settings, color presets
   * Paths for input/output directories
   * Frame rate and default audio track path
2. **bpm\_detector.py**  
   Handles **BPM detection** using the provided audio file:
   * Returns both the estimated **tempo (BPM)** and a **list of beat timestamps**
   * Enables precise timing alignment between the music rhythm and text animations
3. **effects\_config.py**  
   Centralized configuration for all supported text effects:
   * Defines animation durations, initial scaling, rotation angles, overshoot values
   * Contains a SPECIAL\_BEAT\_SKIP\_VALUES map that declares how many beats each effect "skips" to simulate pauses or transitions
   * Supports easy customization and scalability of effect parameters
4. **typographic\_effects.py**  
   Implements **visual effects logic** for individual tokens:
   * Includes side-slide, pop-in, rotation, and any future effects
   * Accepts MoviePy text clips and applies transformations using position, rotation, resizing, or fade transitions
5. **get\_content.py**  
   Reserved for future **GPT integration**:
   * Will dynamically generate text to match available beats
   * Must support a **target word count** that adapts to available beat count minus the number of special character skips
   * Will allow the entire script to be run in one command without manual input

**Script Execution Flow**

Main script --> CLI args --> Text Preprocessing --> Beat Analysis --> Token Timing Assignment --> Video Clip Generation --> Final Composition + Export

**1. Input Parsing & CLI**

* Accepts raw input text and optional flags (--uppercase)
* Text is sanitized via regex to only keep valid characters, including special effect triggers

**2. Tokenization & Special Character Analysis**

* Input text is split into words (“tokens”)
* Each token is analyzed for:
  + Special characters (. \* & ! ^ etc).
  + Effects to apply (pop\_in, side\_slide, rotate, etc.)
  + Dynamic beat skip value using SPECIAL\_BEAT\_SKIP\_VALUES
* Final token object includes:  
  text, skip\_beats, text\_color, bg\_color, effect\_flags

**3. Beat Detection**

* BPM is calculated from the audio file
* A list of **beat timestamps** is returned (real-time, in seconds)
* Beat times are adjusted with WORD\_SPEED\_FACTOR to allow faster/slower word delivery

**4. Token Timing Calculation**

* For each token:
  + start\_time = beat\_times[current\_index]
  + duration = beat\_times[current\_index + skip\_beats + 1] - start\_time
* This ensures accurate placement of words **in rhythm** with the music
* current\_index is advanced by 1 + skip\_beats after each token

**5. Clip Creation & Effects Assignment**

* Each token becomes a TextClip with:
  + Custom font, colors, size, start time, and duration
  + Effect functions (if applicable) applied dynamically:
    - E.g. apply\_pop\_in\_effect(), apply\_rotation\_effect(), etc.
* Each effect is animated over the **first beat interval**, and the token remains on screen for any additional beat skips (creating the illusion of pauses)

**6. Video Composition**

* All text clips are combined via CompositeVideoClip
* Video duration is automatically set to the last clip’s start + duration

**7. Audio Synchronization**

* Background audio is trimmed or looped to match video duration
* Final audio is added to the video and exported

**Designed for Scalability**

✅ Supports unlimited special characters and effects  
✅ Easy to add new effects (just extend typographic\_effects.py and update SPECIAL\_BEAT\_SKIP\_VALUES)  
✅ Dynamic timing system aligns text with real beat flow  
✅ Designed to plug in GPT-generated content that respects total beat count  
✅ Will eventually support full auto-generation from a single audio file

**Formula Awareness & Content Planning**

To ensure GPT content fits within the detected beat list:

# Total number of beats available from the audio

total\_beats = len(beat\_times)

# Total skip beats added by the special characters in the generated content

total\_skip\_beats = sum(SPECIAL\_BEAT\_SKIP\_VALUES[ch] for ch in content if ch in SPECIAL\_BEAT\_SKIP\_VALUES)

# Number of actual words to generate

words\_needed = total\_beats - total\_skip\_beats

This formula can be passed as a constraint in your GPT prompt to dynamically generate perfectly timed content.

**Final Summary**

Your system is already robust, modular, and beat-aware — but more importantly, it is **designed for real-time audiovisual synchronization**. With minimal overhead, you're ready to add:

* GPT-powered content control
* More dynamic effects (e.g. blur, bounce, fade-out)
* Theme and typography presets
* Support for emojis, colored keywords, or even image overlays