VA 345 final: Motion Typography Video

The motivation behind my project is rooted in the theme of "Typography in Motion," with a particular emphasis on **pixelization** as a design and conceptual motif. Inspired by the interplay between grids and typography, the project highlights how letters and words can be broken down into their pixel-like components, mirroring the fundamental building blocks of digital screens. The narrative, "MOTION is in letters, types are moving," conveys the fluidity and adaptability of typography in the digital age. Using a monochromatic black-and-white palette, the video underscores the purity of structure and motion, drawing attention to the intricate grid-based transformations that bring the text to life.

This emphasis on grids not only pays homage to the digital medium but also offers a fresh perspective on how grids can serve as both a structural and aesthetic element in experimental typography.

Collection containing the code of the frames used in this project:

https://editor.p5js.org/denizavukat/collections/4_bXlNzPL

Sliding Text:

The animation uses a **p5.Graphics buffer (pg)** to render text offscreen, allowing independent manipulation of the text, such as slicing and distortion, before integrating it into the main canvas. A **grid-based tiling** system divides the canvas into a grid of tiles, calculated based on `tilesX` and `tilesY`. Each tile corresponds to a segment of the buffer, enabling precise control over localized effects.
Wave and mosaic effects are achieved using sine waves (`sin()`), which introduce dynamic distortions by varying based on the frame count and tile position, creating smooth and continuous motion. **Dynamic source coordinates** leverage the `copy()` function to extract distorted portions of the buffer dynamically, with values determined by the wave effect and time progression.
To enhance visual complexity, **dual effects** are applied by defining and rendering two sets of source and destination coordinates simultaneously.
Finally, **animation control** is managed through `frameCount` and sine functions, producing smooth temporal transitions and enabling visually compelling localized distortions across the grid.

Grid Entering Motion:

This code generates a dynamic animation where grid points inside letters move and wave over time, accompanied by a perspective grid background. It starts by generating a **grid of points** for each letter in the text using the `font.textToPoints()` function, which provides the outline of letters as points. Each point is placed in a grid, and the animation moves these points to the left, wrapping them back to the right when they leave the screen. A **waving effect** is applied to the points using trigonometric functions like `sin()` and `cos()`.

The `isInsideLetter()` check the points inside the letter outlines using **ray-casting algorithm:** It is a computational technique used to determine whether a point lies inside a polygon. It works by casting an imaginary "ray" from the point in a specific direction and counting how many times the ray intersects the edges of the polygon. The key principle is that:

- If the ray intersects the polygon's edges an odd number of times, the point is inside the polygon.
- If the ray intersects an even number of times, the point is outside the polygon determines whether a point lies inside a letter shape, enabling accurate grid point placement.

The **grid background** adds depth and visual interest, scaling and tilting dynamically for a 3D effect.

“LETTERS” Scattering Dots:

This animation dynamically transitions dots from random positions to form the word "LETTERS" while incorporating Perlin noise for a natural and flowing effect. The target positions for the dots are generated using `font.textToPoints()`, which converts each letter into a series of coordinates. Initially, the dots are placed at random positions and animated towards their respective target points using linear interpolation (`lerp`). To enhance visual appeal, Perlin noise is applied to each dot's movement, adding a subtle wave-like effect during the transition. The animation progress is managed with a variable (`animationProgress`) that increments over time, ensuring a smooth transition. Once the animation is complete, it pauses for two seconds before resetting the dots to random positions and restarting the process. Overall, the combination of procedural noise, interpolation, and dynamic resetting creates an engaging and continuous text animation.
This scene used inversely in video editing.

Grid of Types Turning Each Other:

This program creates a dynamic animation that morphs between random letters across a grid. Each grid cell displays a letter transitioning between two randomly selected fonts. The morphing effect is achieved by generating points for the outlines of both

letters using the textToPoints method and interpolating between them using linear interpolation (lerp). The interpolation progress is tracked and reversed when it reaches the limits (0 or 1), creating a seamless back-and-forth animation. To ensure smooth morphing, a normalization function ensures both point arrays have equal lengths by duplicating the last point if necessary.
The program divides the canvas into a grid, calculates the dimensions of each cell, and assigns a random letter pair to each cell. Each cell morphs independently, resulting in a visually engaging and synchronized animation. By combining multiple fonts and random letters, the animation achieves variety and complexity, making it both dynamic and visually captivating.

“are” Waves:

This animation is made in After Effects, using kinetic typography tools.