

# Process 1

Write code in IDLE

Terminal & IDLE



Python code

```
CAPTCHAs Code.py (3.14.2)
import math
import random
from PIL import Image, ImageDraw, ImageFont

def generate_warped_text(text, output_path="output.png"):
    # 1. 基础设置
    width = 600
    height = 200
    background_color = (255, 255, 255) # 白色背景

    # 创建画布
    img = Image.new('RGB', (width, height), color=background_color)
    draw = ImageDraw.Draw(img)

    # 加载字体 (Mac 自带的 Arial 字体路径)
    try:
        font = ImageFont.truetype("/System/Library/Fonts/Supplemental/Arial.ttf")
    except:
        font = ImageFont.load_default()

    # 2. 随机颜色绘制文字 (模仿你图片里的多色效果)
    colors = [(200, 50, 50), (50, 150, 50), (50, 50, 200), (150, 150, 50)]

    # 计算文字居中位置
    text_width = draw.textlength(text, font=font)
    start_x = (width - text_width) / 2
    start_y = (height - 80) / 2

    # 逐个字符绘制, 方便给每个字上色
    current_x = start_x
    for char in text:
        draw.text((current_x, start_y), char, font=font, fill=random.choice(colors))
        current_x += draw.textlength(char, font=font)

    # 3. 核心步骤: 扭曲变形 (Sine Wave Distortion)
    # 创建一个新图像用于存放扭曲后的像素
    warped_img = Image.new('RGB', (width, height), background_color)
    pixels_in = img.load()
    pixels_out = warped_img.load()

    amplitude = 15 # 扭曲的振幅 (波浪的高度)
    frequency = 0.05 # 扭曲的频率 (波浪的密集度)

    for x in range(width):
        for y in range(height):
            # 计算 y 轴的偏移量: $y' = y + A \cdot \sin(2\pi \cdot f \cdot x)$
            offset_y = int(amplitude * math.sin(2 * math.pi * frequency * x))
            new_y = y + offset_y

            # 确保像素点在画布范围内
            if 0 <= new_y < height:
                pixels_out[x, new_y] = pixels_in[x, y]

    # 4. 保存结果
    warped_img.save(output_path)
    print(f"图片已生成并保存至: {output_path}")

# 调用函数
if __name__ == "__main__":
    user_input = input("请输入你想转换的文字: ")
    generate_warped_text(user_input)
```

Output



# Process 2

## Write code in IDLE

```
第二档.py (3.14.2)

import math
import os
from PIL import Image, ImageDraw, ImageFont

# =====
# 配置区域
# =====
CANVAS_WIDTH = 1000
CANVAS_HEIGHT = 1500
BACKGROUND_COLOR = (255, 255, 255)
MARGIN = 80
FONT_SIZE = 70

# 颜色设置
COLORS = [(200, 30, 30), (30, 150, 30), (30, 30, 180)]

# 扭曲参数: 低频率保证可读性
WAVE_AMPLITUDE = 25
WAVE_FREQUENCY = 0.002

def get_best_font():
    """为 Mac 系统寻找支持最全的字体"""
    font_paths = [
        "/System/Library/Fonts/PingFang.ttc",      # 苹方: 支持中文和各种符号
        "/System/Library/Fonts/Supplemental/Arial Unicode.ttf", # Unicode 全支持
        "/System/Library/Fonts/STHeiti Light.ttc",  # 华文细黑
        "/Library/Fonts/Arial Unicode.ttf"
    ]
    for path in font_paths:
        if os.path.exists(path):
            return ImageFont.truetype(path, FONT_SIZE)
    return ImageFont.load_default()

def wrap_text_to_lines(text, font, max_width):
    """
    改进的换行逻辑:
    直接按字符遍历, 确保像“、”这样的符号紧跟文字, 不会因为 split() 而丢失
    """
    lines = []
    current_line = ""

    for char in text:
        test_line = current_line + char
        # 使用 getlength 获取精确的渲染宽度
        if font.getlength(test_line) <= max_width:
            current_line = test_line
        else:
            if current_line:
                lines.append(current_line)
                current_line = char
            if current_line:
                lines.append(current_line)
            return lines

def apply_gentle_wave(image):
    width, height = image.size
    new_img = Image.new('RGB', (width, height), BACKGROUND_COLOR)
    pixels_in = image.load()
    pixels_out = new_img.load()

    for x in range(width):
        # 预计算每一列的偏移量, 提高效率
        offset_y = int(WAVE_AMPLITUDE * math.sin(2 * math.pi * WAVE_FREQUENCY * x))
        for y in range(height):
            source_y = y - offset_y
            if 0 <= source_y < height:
                pixels_out[x, y] = pixels_in[x, source_y]
            else:
                pixels_out[x, y] = BACKGROUND_COLOR

Ln: 89 Col: 23
```

```
第二档.py (3.14.2)

for x in range(width):
    # 预计算每一列的偏移量, 提高效率
    offset_y = int(WAVE_AMPLITUDE * math.sin(2 * math.pi * WAVE_FREQUENCY * x))
    for y in range(height):
        source_y = y - offset_y
        if 0 <= source_y < height:
            pixels_out[x, y] = pixels_in[x, source_y]
        else:
            pixels_out[x, y] = BACKGROUND_COLOR
    return new_img

def create_poster(text, output_filename="symbol_poster.png"):
    print("正在加载字体并排版...")
    font = get_best_font()

    # 1. 创建基础画布
    base_img = Image.new('RGB', (CANVAS_WIDTH, CANVAS_HEIGHT), BACKGROUND_COLOR)
    draw = ImageDraw.Draw(base_img)

    # 2. 文本自动换行
    printable_width = CANVAS_WIDTH - (MARGIN * 2)
    lines = wrap_text_to_lines(text, font, printable_width)

    # 计算垂直起始位置
    line_spacing = 30
    total_text_height = len(lines) * (FONT_SIZE + line_spacing)
    start_y = (CANVAS_HEIGHT - total_text_height) // 2

    current_y = start_y
    color_index = 0

    # 3. 逐行绘制
    for line in lines:
        line_width = font.getlength(line)
        current_x = (CANVAS_WIDTH - line_width) // 2

        # 为了模仿海报的彩色效果, 我们按字符或短语着色
        for char in line:
            current_color = COLORS[color_index % len(COLORS)]
            draw.text((current_x, current_y), char, font=font, fill=current_color)
            # 更新 X 坐标: 获取单个字符宽度
            current_x += font.getlength(char)
            # 只有遇到非空格字符才切换颜色, 保持视觉节奏
            if not char.isspace():
                color_index += 1

        current_y += FONT_SIZE + line_spacing

    print("正在进行波浪变形处理...")
    final_img = apply_gentle_wave(base_img)

    final_img.save(output_filename)
    print(f"生成成功! 文件保存在: {os.path.abspath(output_filename)}")
    os.system(f"open {output_filename}")

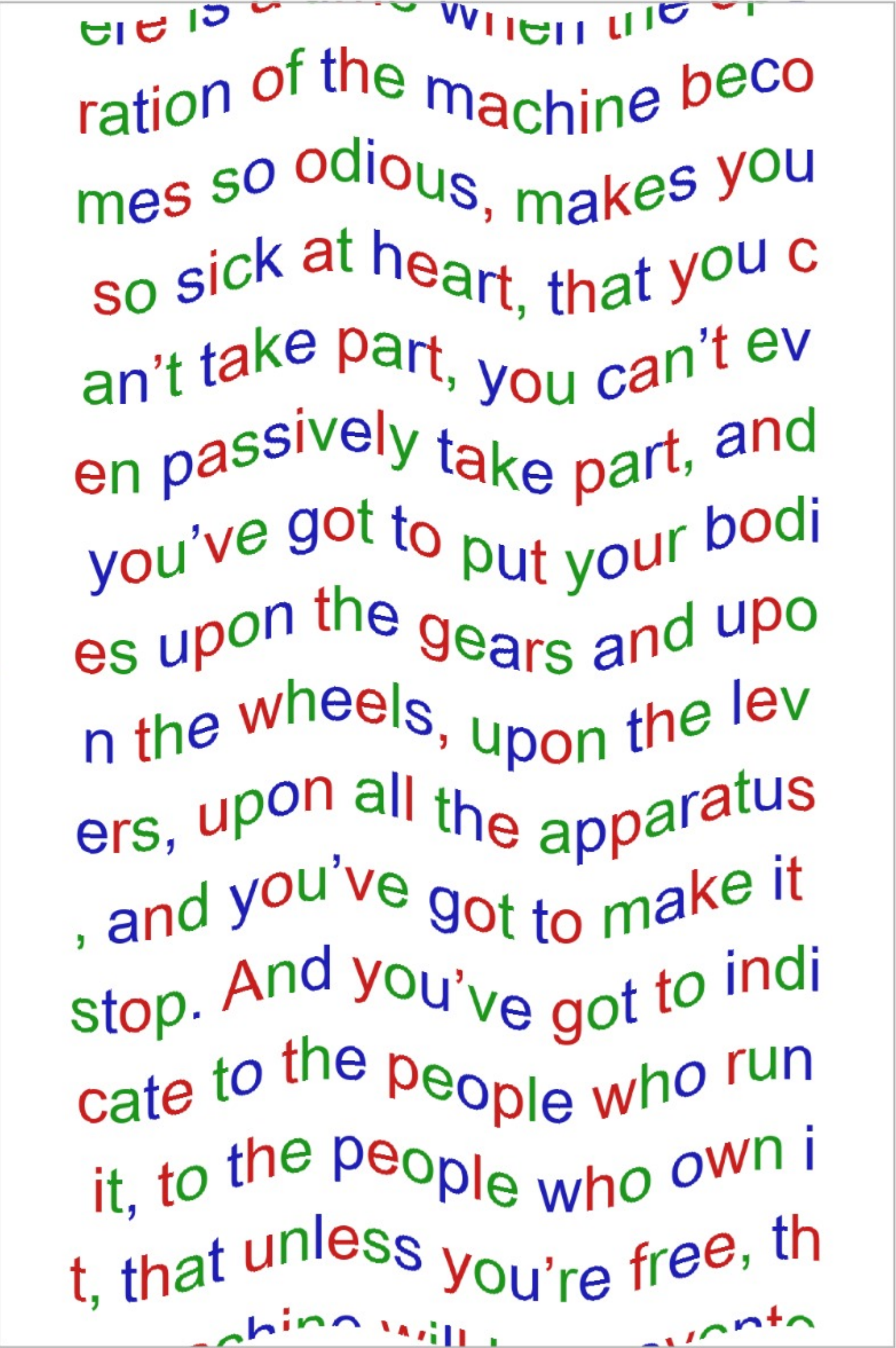
if __name__ == "__main__":
    # 测试文段, 包含你要求的 “” 和 “”
    test_text = "We are human beings!" 他说: '这是一个带有符号的测试'。符号验证: ? ! @#

    print("---- 符号支持版生成器 ----")
    user_input = input("请输入带符号的文字 (回车使用测试样例文段): ")

    if not user_input.strip():
        text_to_process = test_text
    else:
        text_to_process = user_input

    create_poster(text_to_process)

Ln: 127 Col: 8
```



# Process 3

## Write code in IDLE

```
CAPTCHAs Copy 3.0 (3.14.2)

import math
import os
import random
from PIL import Image, ImageDraw, ImageFont

# =====
# 配置区域
# =====
# 固定宽度，高度将动态计算
CANVAS_WIDTH = 1000
BACKGROUND_COLOR = (255, 255, 255)
MARGIN = 80 # 四周留白
FONT_SIZE = 65 # 字体大小
LINE_SPACING = 35 # 行间距

# 颜色池：红、绿、蓝
COLORS = [
    (200, 30, 30), # Red
    (30, 150, 30), # Green
    (30, 30, 180) # Blue
]

# --- 核心修改：强力扭曲参数（参考图6风格） ---
# 振幅：控制波浪的“高度”，越大扭得越厉害
WAVE_AMPLITUDE = 35
# 频率：控制波浪的“密集度”。
# 从 0.002 改为 0.03，波浪变密了很多，形成了剧烈的扭曲效果。
WAVE_FREQUENCY = 0.03

# 默认长文本 (Mario Savio 演讲词)
DEFAULT_TEXT = "We are human beings! There is a time when the operation of the m

# =====
# 工具函数
# =====

def get_best_font():
    """加载对符号支持较好的字体"""
    font_paths = [
        "/System/Library/Fonts/PingFang.ttc",
        "/System/Library/Fonts/Supplemental/Arial Unicode.ttf",
        "/Library/Fonts/Arial Unicode.ttf"
    ]
    for path in font_paths:
        if os.path.exists(path):
            return ImageFont.truetype(path, FONT_SIZE)
    return ImageFont.load_default()

def wrap_text_by_word(text, font, max_width):
    """
    改进的换行函数：按单词分割，保证不把单词切断
    """
    words = text.split()
    lines = []
    current_line = []
    current_width = 0
    space_width = font.getlength(" ")

    for word in words:
        word_width = font.getlength(word)
        # 如果当前行加上新词和空格会超宽，就换行
        if current_width + word_width <= max_width:
            current_line.append(word)
            current_width += word_width + space_width
        else:
            lines.append(" ".join(current_line))
            current_line = [word]
            current_width = word_width + space_width

    if current_line:
        lines.append(" ".join(current_line))
    return lines

def apply_strong_wave(image):
```

Ln: 8 Col: 22

```
CAPTCHAs Copy 3.0 (3.14.2)

def apply_strong_wave(image):
    """应用强力的正弦波扭曲"""
    width, height = image.size
    new_img = Image.new('RGB', (width, height), BACKGROUND_COLOR)
    pixels_in = image.load()
    pixels_out = new_img.load()

    for x in range(width):
        # 计算当前列的垂直偏移量
        offset_y = int(WAVE_AMPLITUDE * math.sin(2 * math.pi * WAVE_FREQUENCY *
            for y in range(height):
                source_y = y - offset_y
                # 边界检查，防止越界
                if 0 <= source_y < height:
                    pixels_out[x, y] = pixels_in[x, source_y]
                else:
                    pixels_out[x, y] = BACKGROUND_COLOR
    return new_img

# =====
# 主逻辑
# =====

def create_final_poster(text, output_filename="final_poster.png"):
    print("1. 正在进行排版计算...")
    font = get_best_font()
    printable_width = CANVAS_WIDTH - (MARGIN * 2)

    # --- 解决问题3：动态计算画布高度 ---
    lines = wrap_text_by_word(text, font, printable_width)
    total_lines = len(lines)
    # 计算所需的总高度 = 行数 * (字体高 + 行间距)
    required_text_height = total_lines * (FONT_SIZE + LINE_SPACING)
    # 最终画布高度 = 文本总高度 + 上下边距
    dynamic_canvas_height = required_text_height + (MARGIN * 2)

    print(f" - 文本共 {total_lines} 行")
    print(f" - 动态计算画布高度为: {dynamic_canvas_height} 像素")

    # 创建基础画布
    base_img = Image.new('RGB', (CANVAS_WIDTH, dynamic_canvas_height), BACKGROUN
    draw = ImageDraw.Draw(base_img)

    # 开始绘制
    current_y = MARGIN # 从顶部边距开始画

    print("2. 正在逐词绘制并随机上色...")
    for line in lines:
        # 计算整行的宽度，用于居中
        line_width = font.getlength(line)
        current_x = (CANVAS_WIDTH - line_width) // 2

        # --- 解决问题2：按单词随机上色 ---
        words_in_line = line.split()
        for word in words_in_line:
            # 为当前单词随机选一个颜色
            word_color = random.choice(COLORS)

            draw.text((current_x, current_y), word, font=font, fill=word_color)

            # 更新下一个词的 X 坐标（加上单词宽和空格宽）
            current_x += font.getlength(word + " ")

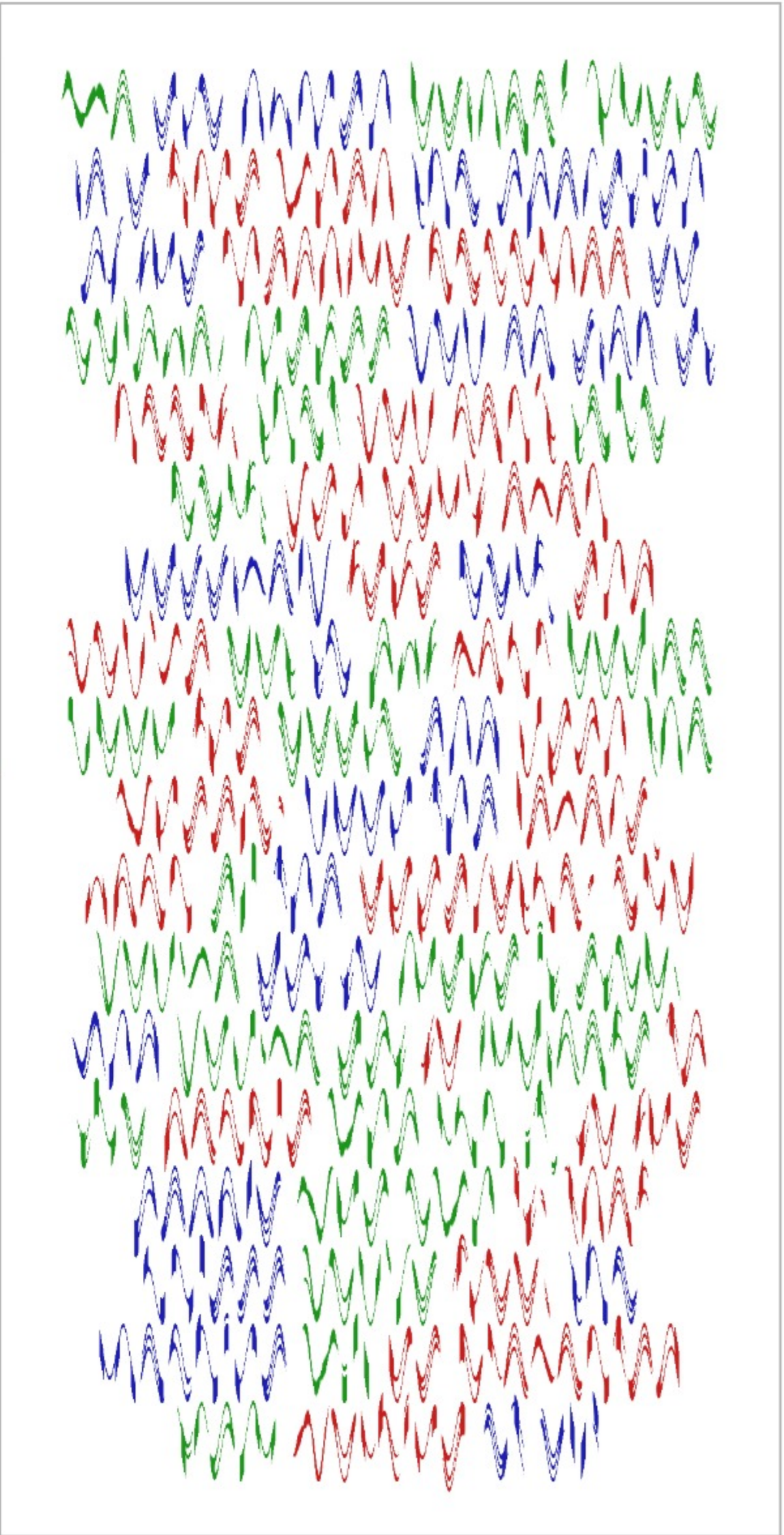
        # 更新下一行的 Y 坐标
        current_y += FONT_SIZE + LINE_SPACING

    print("3. 正在应用强力波浪扭曲（请稍候）...")
    # --- 解决问题1：应用强力扭曲 ---
    final_img = apply_strong_wave(base_img)

    final_img.save(output_filename)
    print(f"\n成功！图片已完美生成并保存为: {output_filename}")
    os.system(f"open {output_filename}")

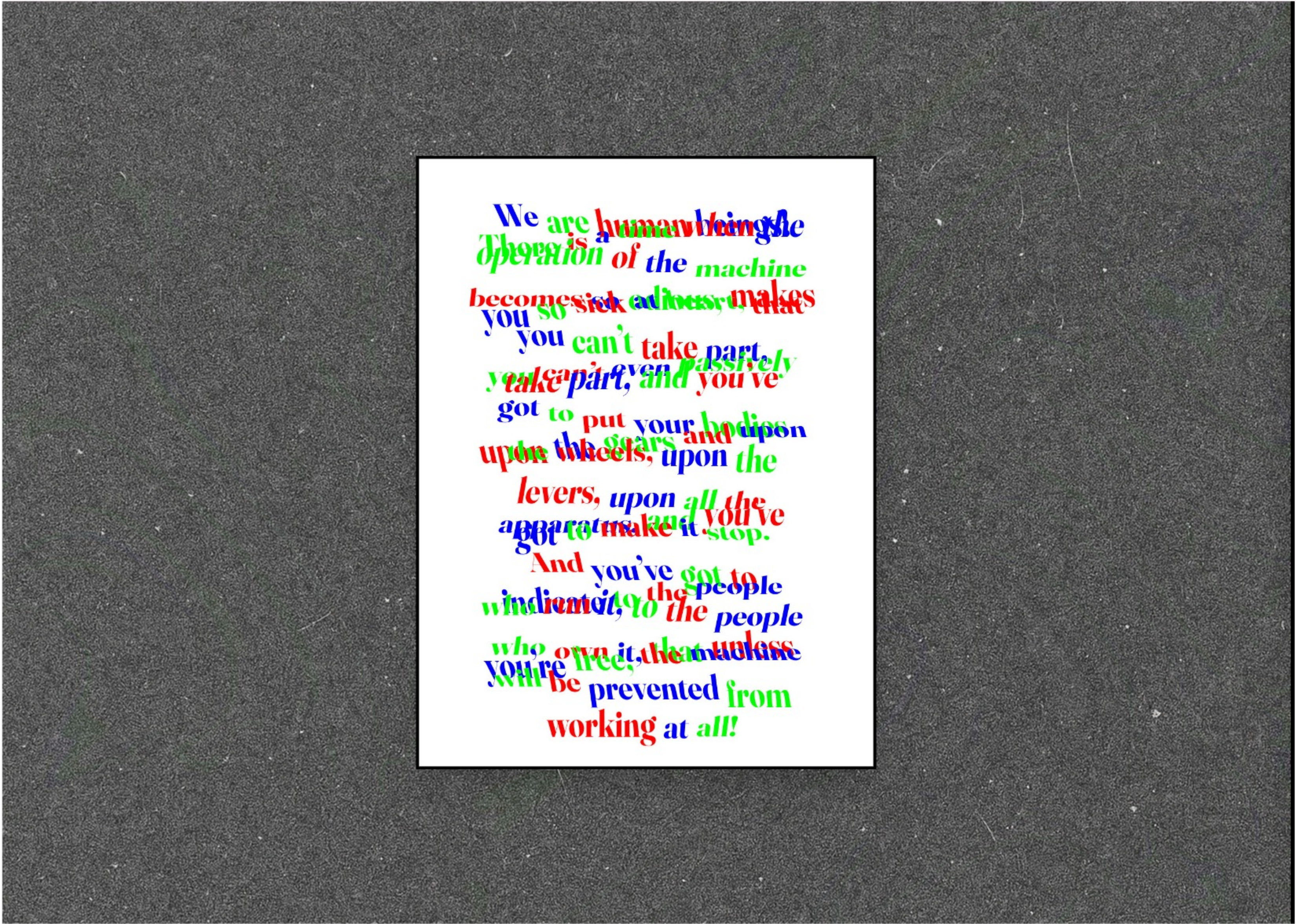
if __name__ == "__main__":
```

Ln: 48 Col: 0



# Process 4

Write code in IDLE



## ***DISTORT***

Variable Font & Acid Graphics Generator

### T TEXT INPUT

indicate to the people  
who run it, to the people  
who own it, that unless  
you're free, the machine  
will be prevented from  
working at all!

DISTORTION (66%)

FONT WEIGHT (710)

FONT SIZE (41PX)

LINE HEIGHT (0.8)

### ↻ COLOR CYCLE

Color 1



Color 2



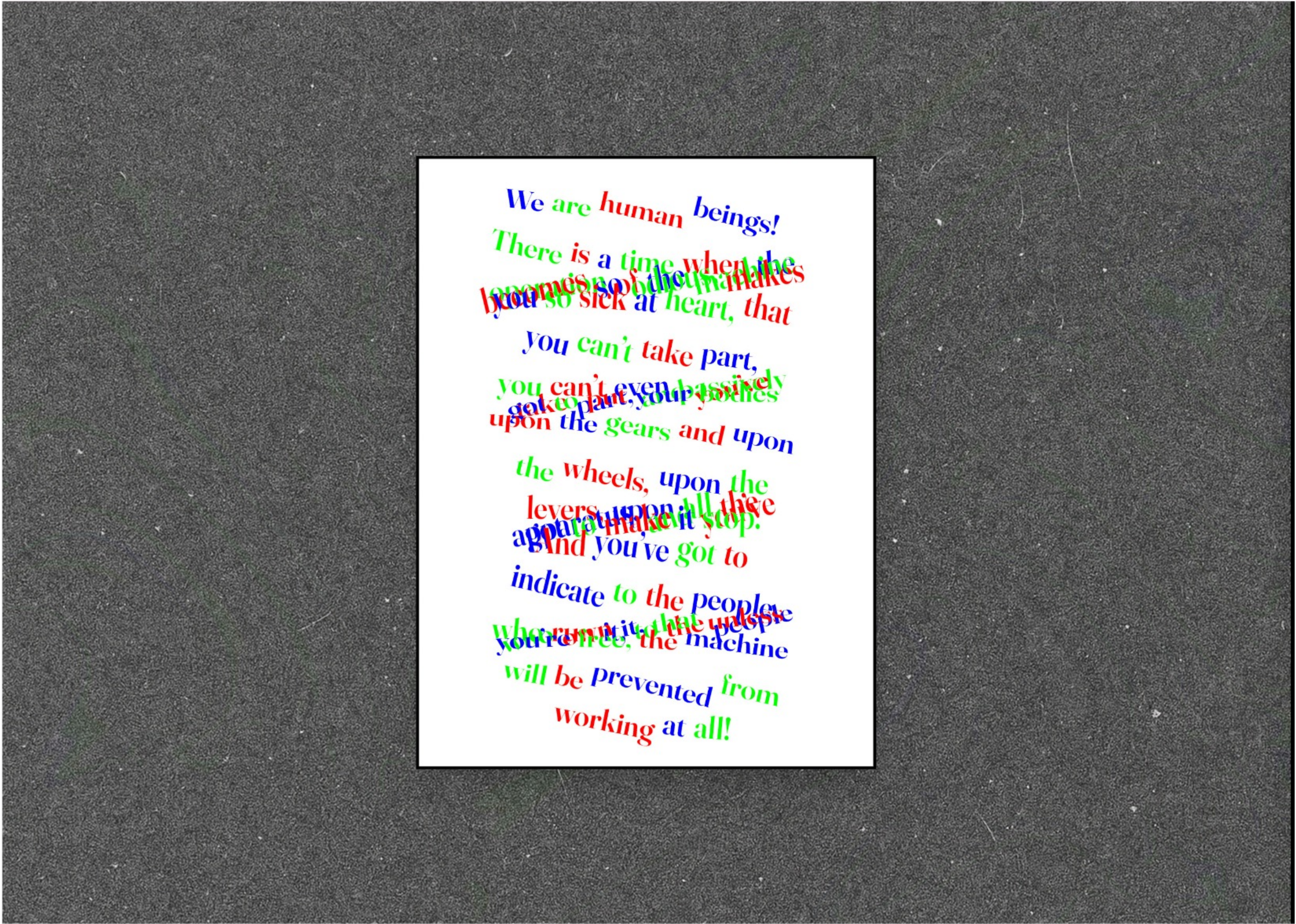
Color 3



↓ EXPORT PNG (3:4)

# Process 5

Using the website to write code



## ***DISTORT***

v2.0 // Wave & Motion



### T INPUT

And you've got to  
indicate to the people  
who run it, to the people  
who own it, that unless  
you're free, the machine

### ☐ WAVE DISTORTION

Amplitude 28

Frequency 0.30

Speed 0.17

### T TYPOGRAPHY

Size  Weight

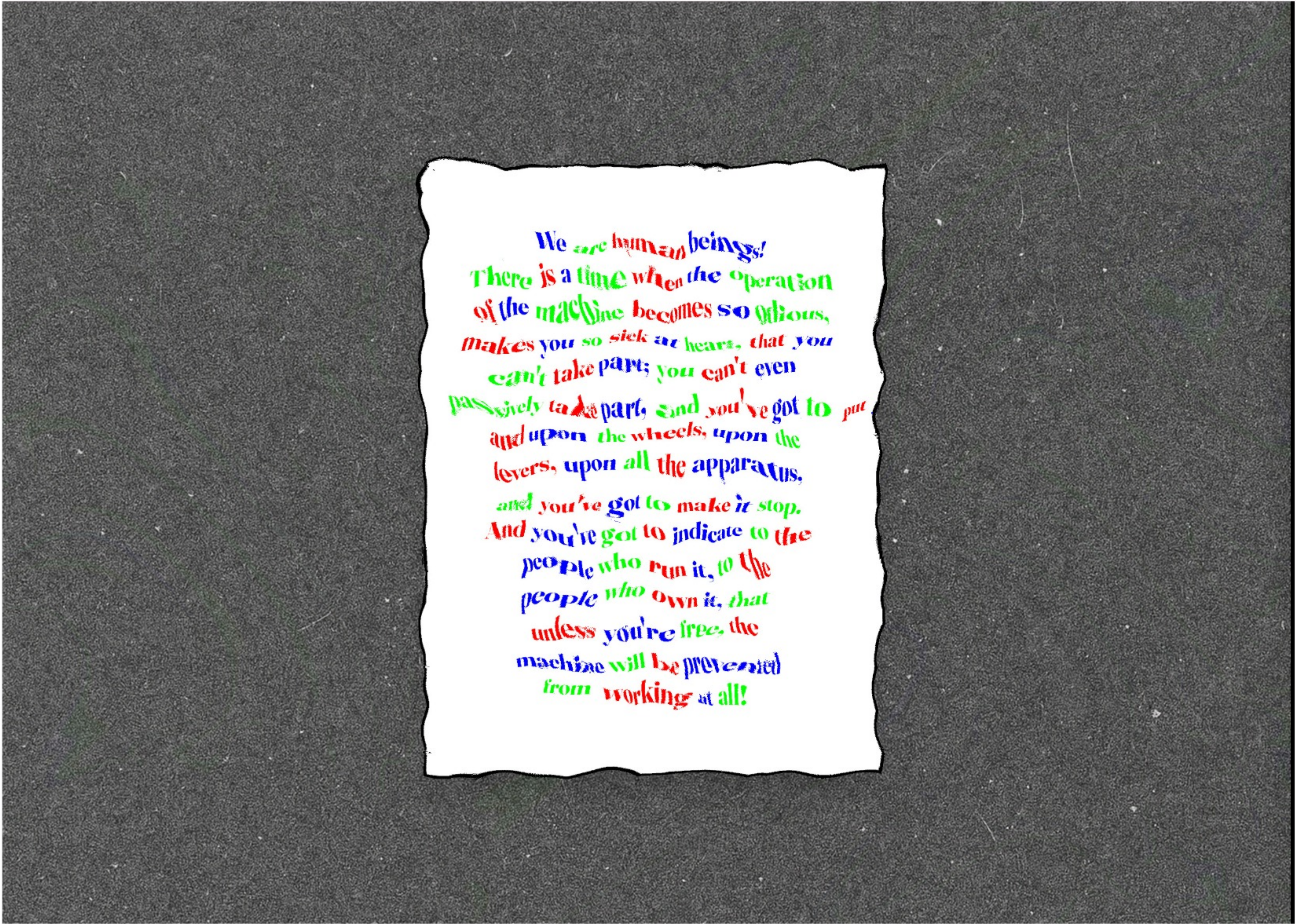
Line Height

### ⊕ COLORS

Portrait (3:4)

# Process 6

Using the website to write code



## ***DISTORT***

v3.0 // Liquid SVG Filters



### T INPUT

and upon the wheels, upon the  
levers, upon all the apparatus,  
and you've got to make it stop.  
And you've got to indicate to the  
people who run it, to the  
people who own it, that

### ☐ LIQUID DISTORTION

Intensity (Scale) 33  
Roughness (Freq) 0.009  
Boil Speed 0.50

### T TYPOGRAPHY

Size Weight  
Line Height

### ⊕ COLORS

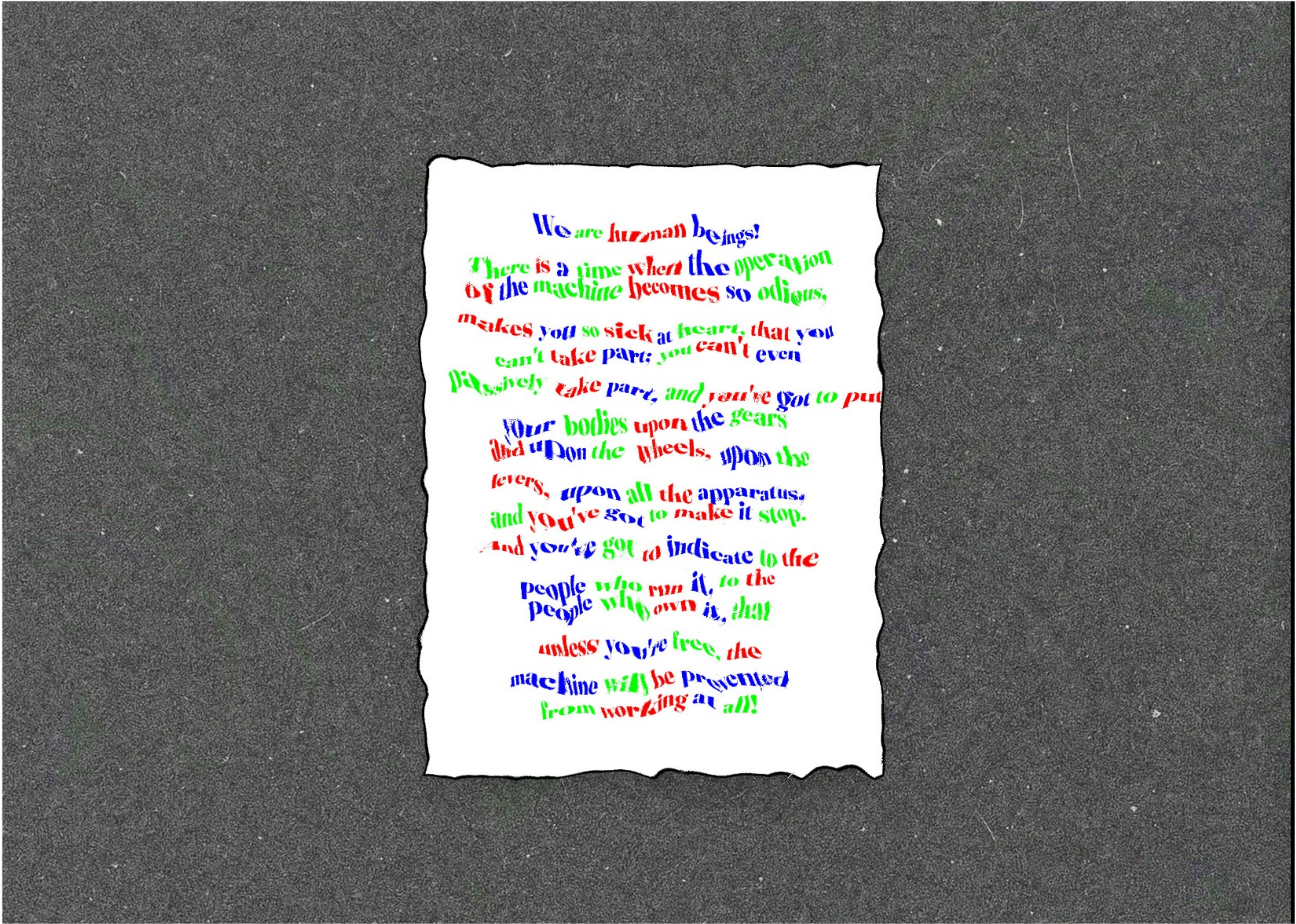
Color selection area with three color swatches: blue, green, and red.

Portrait (3:4) ▶

⬇ PNG ⬆ GIF

# Process 7

Using the website to write code



## ***DISTORT***

v4.0 // Hybrid Engine



### T INPUT

We are human beings!  
There is a time when the operation  
of the machine becomes so odious,  
makes you so sick at heart, that you  
can't take part; you can't even

### ≡ HYBRID DISTORTION

|                 |      |
|-----------------|------|
| Wave Amplitude  | 8    |
| Wave Frequency  | 0.40 |
| Melt Intensity  | 31   |
| Animation Speed | 0.07 |

### T TYPOGRAPHY

|             |        |
|-------------|--------|
| Size        | Weight |
| Line Height |        |

### 🎨 COLORS

|  |  |  |
|--|--|--|
|  |  |  |
|--|--|--|

Portrait (3:4)

PNG

# Output

Using the website to write code

Original



My copy

