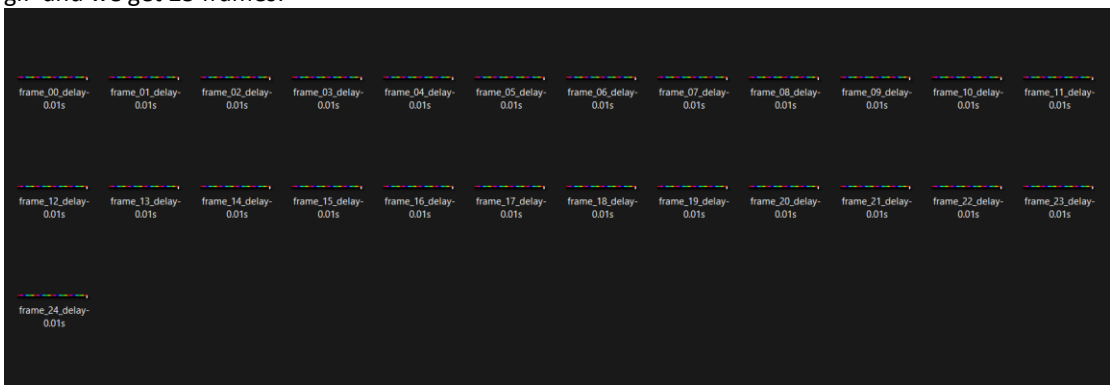
1. We get a gif file, which is only 12 px in height, no visual hints,So we extract the frames of the gif and we get 25 frames:



2. On some analysis and searching on google, we find that these little diagrams are npiet programs,so decoding it with the online compiler https://www.bertnase.de/npiet/npiet-execute.php

3. On executing each of the program, we get 0s and 1s strings.On stacking them on top of one other, we get

```
111111101000011101111111
100000101001100010100000l
101110101100101010111101
101110101110011110111101
101110100110101110111101
100000101110010000l000001
111111101010101010111111
000000000011101000000000
110011100010001100010111l
110101010001001100001l010
010010100111011111l001100
110010011100101100010011O
010110100000001101100l111
101010011011010110001001O
000011100001110110111l100
001011010000001011111O110
110100110001100011111l100
000000001111000110001000O
111111100101100010101000O
100000101001101010001111O
101110101110000011111111O
101110100011001101110011l
101110100011000010100101O
100000101001000001111111O
111111101011000101100011l
```

4. This looks like a 25*25 qr code in the form of 0s and 1s

5. On constructing a qr code using this script

```
        import numpy as np
from PIL import Image
import sys

def decode_25x25(txt_in, out_img):
    # Read file manually to handle both one-line and multi-line cases
    with open(txt_in, "r") as f:
        lines = [line.strip() for line in f if line.strip()]
```

```
    # If file is a single long line of 625 chars
    if len(lines) == 1 and len(lines[0]) == 625:
        data = list(lines[0])
        grid = np.array(data, dtype=np.uint8).reshape(25, 25)
    else:
        # Otherwise assume 25 lines each with 25 digits
        grid = np.array([[int(c) for c in line] for line in lines], dtype=np.uint8)
```

```
    h, w = grid.shape
    if h != 25 or w != 25:
        print(f"[!] Expected 25x25 grid but got {h}x{w}")
        sys.exit(1)
    print(f"[+] Loaded grid: {h}x{w}")
```

```
    # Convert 1 = black, 0 = white
    img = Image.fromarray(((1 - grid) * 255).astype(np.uint8), mode="L")
```

```
    # Add white border for scannability
    border = 4
    bordered = Image.new("L", (w + 2 * border, h + 2 * border), 255)
    bordered.paste(img, (border, border))
```

```
    # Scale up
    scale = 10
    final_img = bordered.resize(
        (bordered.width * scale, bordered.height * scale), Image.NEAREST
    )
    final_img.save(out_img)
    print(f"[+] Reconstructed QR saved as {out_img}")
```

```
if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: python 25qr_decoder.py <in_txt> <out_png>")
        sys.exit(1)
    decode_25x25(sys.argv[1], sys.argv[2])
```

6.Using this we get a qr code, which leads us to a drive link which has the flag.txt