```c
/**
 * Copyright (c) 2020 Raspberry Pi (Trading) Ltd.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include <stdio.h>
#include <stdlib.h>

#include "pico/stdlib.h"
#include "hardware/pio.h"
#include "hardware/clocks.h"
#include "ws2812.pio.h"

#define IS_RGBW true
#define NUM_PIXELS 150

#ifdef PICO_DEFAULT_WS2812_PIN
#define WS2812_PIN PICO_DEFAULT_WS2812_PIN
#else
// default to pin 2 if the board doesn't have a default WS2812 pin defined
#define WS2812_PIN 2
#endif

static inline void put_pixel(uint32_t pixel_grb) {
    pio_sm_put_blocking(pio0, 0, pixel_grb << 8u);
}

static inline uint32_t urgb_u32(uint8_t r, uint8_t g, uint8_t b) {
    return
            ((uint32_t) (r) << 8) |
            ((uint32_t) (g) << 16) |
            (uint32_t) (b);
}

void pattern_snakes(uint len, uint t) {
    for (uint i = 0; i < len; ++i) {
        uint x = (i + (t >> 1)) % 64;
        if (x < 10)
            put_pixel(urgb_u32(0xff, 0, 0));
        else if (x >= 15 && x < 25)
            put_pixel(urgb_u32(0, 0xff, 0));
        else if (x >= 30 && x < 40)
            put_pixel(urgb_u32(0, 0, 0xff));
        else
            put_pixel(0);
    }
}

void pattern_random(uint len, uint t) {
    if (t % 8)
        return;
    for (int i = 0; i < len; ++i)
        put_pixel(rand());
}

void pattern_sparkle(uint len, uint t) {
    if (t % 8)
        return;
    for (int i = 0; i < len; ++i)
        put_pixel(rand() % 16 ? 0 : 0xffffffff);
}

void pattern_greys(uint len, uint t) {
    int max = 100; // let's not draw too much current!
    t %= max;
    for (int i = 0; i < len; ++i) {
        put_pixel(t * 0x10101);
        if (++t >= max) t = 0;
    }
}
```

(24) pio instance "pio0" put the given number left-shifted 8 bits to the TX FIFO of the "state machine 0"

(25) if t is not multiples of 8, just return

(26) if t is , then randomly set the pixel

```c

typedef void (*pattern)(uint len, uint t);
const struct {
    pattern pat;
    const char *name;
} pattern_table[] = {
        {pattern_snakes,  "Snakes!"},
        {pattern_random,  "Random data"},
        {pattern_sparkle, "Sparkles"},
        {pattern_greys,   "Greys"},
};

int main() {
    //set_sys_clock_48();
    stdio_init_all(); (1) initialize all of the present standard stdio types
    printf("WS2812 Smoke Test, using pin %d", WS2812_PIN); (2) print the operating pin #

    // todo get free sm
    PIO pio = pio0; (3) choose pio 0 as the pio instance.
    int sm = 0; (4) initialize the state machine
    uint offset = pio_add_program(pio, &ws2812_program); (5) load the program, returns the starting location of the program in the memory.

    ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW); (6) initialize the program with { pio instance / state machine / program location / operating pin # / freq./baud rate. / choice of RGB (threshold in bits to shift one out of OSR)

    int t = 0;
    while (1) {
        int pat = rand() % count_of(pattern_table); (20) randomly choose a pattern
        int dir = (rand() >> 30) & 1 ? 1 : -1; (21) randomly generate a direction, represented by ±1
        puts(pattern_table[pat].name); (22) print the chosen pattern's name.
        puts(dir == 1 ? "(forward)" : "(backward)"); (23) print the direction
        for (int i = 0; i < 1000; ++i) {
            pattern_table[pat].pat(NUM_PIXELS, t); (24) set the light by the chosen pattern.
            sleep_ms(10); (18) sleep for 10ms
            t += dir; (19) t ++
        }
    }
}

```