#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_RESET    -1

#define SCREEN\_ADDRESS 0x3C

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);

// 您的位图数据

const unsigned char bitmap[] PROGMEM = {

  0x3C, 0x00, 0x03, 0xF0, 0x00, 0x04, 0x08, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

  0xE0, 0x00, 0x0E, 0x1E, 0x00, 0x08, 0x08, 0x10, 0x00, 0x00, 0x04, 0x00, 0x04, 0x00, 0x00, 0xC0,

  0x80, 0x00, 0xF8, 0x03, 0xC0, 0x08, 0x08, 0x10, 0x00, 0x00, 0x04, 0x00, 0x0C, 0x00, 0x00, 0xC0,

  0x00, 0x03, 0x80, 0x30, 0x70, 0x08, 0x08, 0x10, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0xC0,

  0x00, 0x1E, 0x00, 0x60, 0x11, 0x08, 0x18, 0x10, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x03, 0xF0,

  0x00, 0xF0, 0x00, 0xC0, 0x10, 0x08, 0x60, 0x10, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x07, 0xF0,

  0x0F, 0x80, 0x00, 0x80, 0x10, 0x09, 0x80, 0x10, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0,

  0x38, 0x00, 0xF8, 0x80, 0x10, 0x0E, 0x00, 0x70, 0x00, 0x00, 0x00, 0x07, 0xFF, 0x00, 0x00, 0xC0,

  0x60, 0x01, 0x88, 0x7E, 0x10, 0x08, 0x01, 0xC0, 0x00, 0x00, 0x00, 0x0F, 0xFF, 0x80, 0x00, 0x80,

  0x00, 0x03, 0x08, 0x03, 0x10, 0x00, 0x0F, 0x40, 0x00, 0x00, 0x00, 0x1C, 0x01, 0xC0, 0x10, 0x00,

  0x00, 0x42, 0x08, 0x01, 0x10, 0x00, 0xF8, 0x40, 0x00, 0x00, 0x00, 0x38, 0x00, 0xE0, 0x10, 0x00,

  0x00, 0xC2, 0x18, 0x23, 0x10, 0x03, 0x88, 0x40, 0x00, 0x00, 0x00, 0x70, 0x00, 0x70, 0x10, 0x00,

  0x00, 0x82, 0x71, 0x1E, 0x10, 0x1F, 0x08, 0x40, 0x00, 0x00, 0x00, 0xE0, 0x00, 0x38, 0x30, 0x00,

  0x00, 0x83, 0xC3, 0x00, 0x10, 0x71, 0x09, 0xC0, 0x02, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x00, 0x00,

  0x21, 0x81, 0x86, 0x00, 0x13, 0xC1, 0x0F, 0xC0, 0x07, 0x00, 0x03, 0x80, 0x00, 0x00, 0x00, 0x00,

  0x31, 0x00, 0xFC, 0x00, 0x1E, 0x41, 0xFE, 0x40, 0x02, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00,

  0x01, 0x00, 0x00, 0x00, 0x78, 0x43, 0xF8, 0x40, 0x00, 0x00, 0x03, 0x18, 0x00, 0x00, 0x00, 0x00,

  0x09, 0x81, 0x00, 0x00, 0xF0, 0x47, 0x08, 0x40, 0x00, 0x00, 0x03, 0x0C, 0x00, 0x00, 0x0E, 0x00,

  0x08, 0x83, 0x00, 0x03, 0x90, 0x7F, 0x08, 0x40, 0x00, 0x00, 0x03, 0x00, 0x01, 0x00, 0x11, 0x00,

  0x0C, 0xC6, 0x00, 0x1F, 0x10, 0xF1, 0x08, 0x40, 0x00, 0x00, 0x03, 0x00, 0x03, 0x80, 0x24, 0x80,

  0x06, 0x7C, 0x00, 0x7A, 0x13, 0xC1, 0x08, 0x40, 0x00, 0x00, 0x43, 0x00, 0x01, 0x00, 0x2C, 0x80,

  0x02, 0x00, 0x1F, 0xC6, 0x1E, 0x41, 0x08, 0x40, 0x00, 0x00, 0x63, 0x00, 0x00, 0x00, 0x28, 0x80,

  0x03, 0x00, 0x3C, 0x8A, 0x70, 0x41, 0x09, 0xE0, 0x10, 0x00, 0x23, 0x00, 0x00, 0x00, 0x20, 0x80,

  0x00, 0x01, 0xE1, 0x57, 0xD0, 0x41, 0x0F, 0x80, 0x10, 0x00, 0x03, 0x00, 0x00, 0x00, 0x20, 0x80,

  0x00, 0x0F, 0x22, 0x25, 0x10, 0x41, 0x3C, 0x00, 0x28, 0x00, 0x03, 0x1F, 0xFF, 0x00, 0x11, 0x00,

  0x00, 0xF8, 0x54, 0x3E, 0x10, 0x43, 0xE0, 0x00, 0x00, 0x00, 0x03, 0x3F, 0xFF, 0x83, 0x0A, 0x00,

  0x07, 0x88, 0x88, 0xE4, 0x10, 0x4F, 0x00, 0x00, 0x00, 0x00, 0x03, 0x70, 0x01, 0xC3, 0x04, 0x00,

  0x3C, 0x15, 0x07, 0x04, 0x10, 0x5B, 0x00, 0x80, 0x00, 0x00, 0x03, 0xE0, 0x00, 0xE0, 0x0E, 0x00,

  0xFA, 0x22, 0x1E, 0x04, 0x10, 0xF1, 0x80, 0x80, 0x00, 0x08, 0x03, 0xC0, 0x00, 0x70, 0x04, 0x08,

  0x89, 0x40, 0xF2, 0x04, 0x17, 0xB0, 0xC1, 0x40, 0x00, 0x0C, 0x03, 0x86, 0x0C, 0x38, 0x04, 0x18,

  0x08, 0x8F, 0x83, 0x04, 0x3C, 0x18, 0x60, 0x00, 0x00, 0x08, 0x03, 0x0F, 0x1E, 0x18, 0x04, 0x10,

  0x08, 0x78, 0x83, 0x04, 0xE6, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x03, 0x19, 0xB3, 0x18, 0x06, 0x00,

  0x0F, 0xE0, 0xC2, 0x07, 0x83, 0x06, 0x18, 0x00, 0x00, 0x00, 0x03, 0x18, 0xE3, 0x18, 0x02, 0x00,

  0x0C, 0x21, 0xC3, 0x7C, 0x01, 0x83, 0x0C, 0x00, 0x00, 0x00, 0x03, 0x18, 0x43, 0x18, 0x02, 0x00,

  0x18, 0x21, 0x87, 0xC0, 0x00, 0xC1, 0x86, 0x00, 0x00, 0x00, 0x03, 0x18, 0x0B, 0x18, 0x01, 0x00,

  0x10, 0x60, 0x8F, 0x80, 0x10, 0x60, 0xC3, 0x00, 0x01, 0x00, 0x03, 0x18, 0x1B, 0x18, 0x01, 0x00,

  0x10, 0x60, 0xFB, 0x80, 0x10, 0x30, 0x61, 0x80, 0x01, 0x00, 0x03, 0x18, 0x13, 0x18, 0x01, 0x00,

  0x50, 0x21, 0xC1, 0xC0, 0x28, 0x18, 0x30, 0xC0, 0x02, 0x80, 0x03, 0x18, 0x03, 0x18, 0x01, 0x00,

  0x70, 0x3E, 0x01, 0xC0, 0x00, 0x0C, 0x18, 0x71, 0xC4, 0x40, 0x03, 0x0C, 0x06, 0x18, 0x80, 0x00,

  0x70, 0xF0, 0x00, 0xE0, 0x00, 0x06, 0x0C, 0x3B, 0x98, 0x30, 0x03, 0x06, 0x0C, 0x19, 0x81, 0x00,

  0x77, 0x38, 0x00, 0xF0, 0x00, 0x03, 0x06, 0x1E, 0x04, 0x40, 0x23, 0x03, 0x18, 0x18, 0x80, 0x00,

  0x3C, 0x38, 0x03, 0xC0, 0x00, 0x41, 0x83, 0x78, 0x02, 0x80, 0x23, 0x81, 0xB0, 0x38, 0x01, 0x00,

  0xF0, 0x1C, 0x0E, 0x00, 0x40, 0x40, 0xC1, 0xF0, 0x01, 0x00, 0x21, 0xC0, 0xE0, 0x70, 0x14, 0x60,

  0x80, 0x0C, 0x7C, 0x00, 0x40, 0xA0, 0x67, 0xC0, 0x01, 0x00, 0x60, 0xE0, 0x40, 0xE0, 0x01, 0x00,

  0x00, 0x0F, 0xE0, 0x00, 0xA0, 0x00, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x70, 0x01, 0xC0, 0x00, 0x00,

  0x00, 0x07, 0x80, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x3F, 0xFF, 0x88, 0x02, 0x80,

  0x00, 0x06, 0x00, 0x00, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x1F, 0xFF, 0x0C, 0x08, 0x40,

  0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,

  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

  0x00, 0x00, 0x00, 0x03, 0x80, 0xFE, 0x10, 0x18, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x80,

  0x0C, 0x0F, 0x1F, 0x87, 0xE0, 0xFE, 0x18, 0x31, 0x00, 0xF0, 0x3F, 0x8F, 0xE2, 0x20, 0x03, 0x04,

  0x0C, 0x1D, 0x1F, 0x8C, 0x60, 0xC6, 0x0C, 0x23, 0x81, 0x08, 0x24, 0x91, 0x02, 0x20, 0x1C, 0x04,

  0x00, 0x30, 0x18, 0x08, 0x30, 0xC0, 0x04, 0x65, 0x42, 0x00, 0x2E, 0xA1, 0x02, 0xFC, 0x20, 0x04,

  0x0C, 0x30, 0x18, 0x00, 0x30, 0xC0, 0x06, 0xCD, 0x62, 0x00, 0x24, 0x8F, 0xC3, 0x24, 0x22, 0x04,

  0x0C, 0x60, 0x18, 0x00, 0x30, 0xFC, 0x03, 0x95, 0x52, 0x00, 0x2E, 0x89, 0x02, 0xA8, 0x22, 0x04,

  0x0C, 0x60, 0x1F, 0x03, 0xF0, 0x7E, 0x01, 0x9F, 0xF2, 0xF0, 0x20, 0x89, 0x02, 0x30, 0x22, 0x04,

  0x0C, 0x60, 0x1F, 0x06, 0x60, 0x06, 0x03, 0x08, 0x23, 0x08, 0x24, 0xBF, 0xE3, 0xFF, 0x1F, 0xC4,

  0x0C, 0x70, 0x18, 0x0C, 0x30, 0x06, 0x06, 0x0F, 0x22, 0x08, 0x2A, 0x81, 0x0A, 0x20, 0x02, 0x00,

  0x0C, 0x30, 0x18, 0x0C, 0x30, 0xC6, 0x0C, 0x09, 0x22, 0x08, 0x4A, 0x81, 0x12, 0x20, 0x0A, 0x84,

  0x0C, 0x1D, 0x1F, 0x8E, 0x70, 0xFE, 0x18, 0x0D, 0x22, 0x08, 0x44, 0x81, 0x02, 0x50, 0x12, 0x40,

  0x0C, 0x0F, 0x1F, 0x87, 0xEC, 0xFE, 0x30, 0x09, 0x21, 0x10, 0x80, 0x81, 0x02, 0x48, 0x22, 0x30,

  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0x1F, 0xF0, 0xE1, 0x01, 0x81, 0x02, 0x87, 0x06, 0x00,

  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 优化后的粒子系统（减少数量）

struct Particle {

  int x, y;

  int vx, vy;

  int targetX, targetY;

};

#define MAX\_PARTICLES 40  // 从80减少到40

Particle particles[MAX\_PARTICLES];

unsigned long effectStartTime = 0;

int currentEffect = 0;

// 数字雨变量

#define RAIN\_COLS 16  // 从20减少到16

int8\_t rainY[RAIN\_COLS];

int8\_t rainSpeed[RAIN\_COLS];

void setup() {

  Serial.begin(115200);

  Wire.begin(6, 7);

  if(!display.begin(SSD1306\_SWITCHCAPVCC, SCREEN\_ADDRESS)) {

    Serial.println(F("SSD1306失败"));

    for(;;);

  }

  display.clearDisplay();

  display.setTextSize(2);

  display.setTextColor(SSD1306\_WHITE);

  display.setCursor(5, 15);

  display.println(F("SMOOTH"));

  display.setCursor(10, 35);

  display.println(F("EFFECTS"));

  display.display();

  delay(2000);

  effectStartTime = millis();

  initParticles();

  initRain();

  Serial.println("流畅版本启动！");

}

void loop() {

  unsigned long elapsed = millis() - effectStartTime;

  if (elapsed >= 6000) {

    currentEffect = (currentEffect + 1) % 5;

    effectStartTime = millis();

    elapsed = 0;

    if (currentEffect == 2) initParticles();

    if (currentEffect == 0) initRain();

    Serial.print("效果: ");

    Serial.println(currentEffect + 1);

  }

  switch(currentEffect) {

    case 0:

      effect1\_MatrixRain(elapsed);

      break;

    case 1:

      effect2\_RotateWarp(elapsed);

      break;

    case 2:

      effect3\_ParticleBurst(elapsed);

      break;

    case 3:

      effect4\_WaveBreathing(elapsed);

      break;

    case 4:

      effect5\_ZoomPulse(elapsed);

      break;

  }

}

// ============ 效果1: 矩阵数字雨（优化版）============

void initRain() {

  for (int i = 0; i < RAIN\_COLS; i++) {

    rainY[i] = random(-64, 0);

    rainSpeed[i] = random(2, 5);

  }

}

void effect1\_MatrixRain(unsigned long elapsed) {

  display.clearDisplay();

  // 绘制简化的数字雨

  for (int i = 0; i < RAIN\_COLS; i++) {

    int x = i \* 8;

    // 只绘制3条拖尾（而不是8条）

    for (int j = 0; j < 3; j++) {

      int y = rainY[i] - j \* 10;

      if (y >= 0 && y < 64) {

        display.setCursor(x, y);

        display.setTextSize(1);

        display.write('0' + (i + j) % 10);

      }

    }

    rainY[i] += rainSpeed[i];

    if (rainY[i] > 70) rainY[i] = -10;

  }

  // 逐渐显示图像（每帧只绘制部分）

  int linesToDraw = (elapsed / 100) % 64;

  for (int y = 0; y <= linesToDraw; y += 2) {  // 每2行绘制一次

    for (int x = 0; x < 128; x++) {

      int byteIndex = y \* 16 + x / 8;

      int bitIndex = 7 - (x % 8);

      if (pgm\_read\_byte(&bitmap[byteIndex]) & (1 << bitIndex)) {

        display.drawPixel(x, y, SSD1306\_WHITE);

      }

    }

  }

  display.display();

  delay(50);

}

// ============ 效果2: 旋转扭曲（简化版）============

void effect2\_RotateWarp(unsigned long elapsed) {

  display.clearDisplay();

  float angle = elapsed / 2000.0;

  // 简化：只做旋转，不做扭曲

  for (int y = 0; y < 64; y += 1) {  // 全像素

    for (int x = 0; x < 128; x += 1) {

      float dx = x - 64;

      float dy = y - 32;

      // 简单旋转

      float dist = sqrt(dx\*dx + dy\*dy);

      float a = atan2(dy, dx) + angle;

      int srcX = 64 + (int)(cos(a) \* dist);

      int srcY = 32 + (int)(sin(a) \* dist);

      if (srcX >= 0 && srcX < 128 && srcY >= 0 && srcY < 64) {

        int byteIndex = srcY \* 16 + srcX / 8;

        int bitIndex = 7 - (srcX % 8);

        if (pgm\_read\_byte(&bitmap[byteIndex]) & (1 << bitIndex)) {

          display.drawPixel(x, y, SSD1306\_WHITE);

        }

      }

    }

  }

  display.display();

  delay(40);

}

// ============ 效果3: 粒子爆发（优化版）============

void initParticles() {

  int count = 0;

  // 降低采样密度，从每3像素采样改为每5像素

  for (int y = 0; y < 64 && count < MAX\_PARTICLES; y += 5) {

    for (int x = 0; x < 128 && count < MAX\_PARTICLES; x += 5) {

      int byteIndex = y \* 16 + x / 8;

      int bitIndex = 7 - (x % 8);

      if (pgm\_read\_byte(&bitmap[byteIndex]) & (1 << bitIndex)) {

        particles[count].targetX = x;

        particles[count].targetY = y;

        particles[count].x = 64;

        particles[count].y = 32;

        // 使用整数运算

        int angle = random(0, 360);

        particles[count].vx = (cos(angle \* 0.01745) \* 3);

        particles[count].vy = (sin(angle \* 0.01745) \* 3);

        count++;

      }

    }

  }

}

void effect3\_ParticleBurst(unsigned long elapsed) {

  display.clearDisplay();

  bool isExploding = (elapsed % 6000) < 2000;

  bool isGathering = (elapsed % 6000) >= 2000 && (elapsed % 6000) < 4500;

  for (int i = 0; i < MAX\_PARTICLES; i++) {

    if (isExploding) {

      particles[i].x += particles[i].vx;

      particles[i].y += particles[i].vy;

      // 整数阻力

      particles[i].vx = (particles[i].vx \* 95) / 100;

      particles[i].vy = (particles[i].vy \* 95) / 100;

    } else if (isGathering) {

      int dx = particles[i].targetX - particles[i].x;

      int dy = particles[i].targetY - particles[i].y;

      particles[i].vx = dx / 10;

      particles[i].vy = dy / 10;

      particles[i].x += particles[i].vx;

      particles[i].y += particles[i].vy;

    } else {

      particles[i].x = particles[i].targetX;

      particles[i].y = particles[i].targetY;

    }

    // 绘制粒子

    if (particles[i].x >= 0 && particles[i].x < 128 &&

        particles[i].y >= 0 && particles[i].y < 64) {

      display.drawPixel(particles[i].x, particles[i].y, SSD1306\_WHITE);

      display.drawPixel(particles[i].x + 1, particles[i].y, SSD1306\_WHITE);

      display.drawPixel(particles[i].x, particles[i].y + 1, SSD1306\_WHITE);

      display.drawPixel(particles[i].x + 1, particles[i].y + 1, SSD1306\_WHITE);

    }

  }

  display.display();

  delay(30);

}

// ============ 效果4: 波浪呼吸============

void effect4\_WaveBreathing(unsigned long elapsed) {

  display.clearDisplay();

  int waveOffset = elapsed / 20;

  // 波浪效果 - 每2行处理

  for (int y = 0; y < 64; y += 1) {

    int wave = sin((y + waveOffset) \* 0.15) \* 6;

    for (int x = 0; x < 128; x++) {

      int srcX = (x - wave + 128) % 128;

      int byteIndex = y \* 16 + srcX / 8;

      int bitIndex = 7 - (srcX % 8);

      if (pgm\_read\_byte(&bitmap[byteIndex]) & (1 << bitIndex)) {

        display.drawPixel(x, y, SSD1306\_WHITE);

      }

    }

  }

  // 呼吸灯

  int brightness = (sin(elapsed / 500.0) \* 127.5) + 127.5;

  display.ssd1306\_command(SSD1306\_SETCONTRAST);

  display.ssd1306\_command(brightness);

  display.display();

  delay(30);

}

// ============ 效果5: 缩放脉冲============

void effect5\_ZoomPulse(unsigned long elapsed) {

  display.clearDisplay();

  // 缩放系数

  float scale = 1.0 + 0.3 \* sin(elapsed / 800.0);

  // 简化的缩放 - 降低采样率

  for (int y = 0; y < 64; y++) {

    for (int x = 0; x < 128; x++) {

      int sx = (x - 64) / scale + 64;

      int sy = (y - 32) / scale + 32;

      if (sx >= 0 && sx < 128 && sy >= 0 && sy < 64) {

        int byteIndex = sy \* 16 + sx / 8;

        int bitIndex = 7 - (sx % 8);

        if (pgm\_read\_byte(&bitmap[byteIndex]) & (1 << bitIndex)) {

          display.drawPixel(x, y, SSD1306\_WHITE);

        }

      }

    }

  }

  // 边框脉冲效果

  if ((elapsed / 100) % 2 == 0) {

    display.drawRect(0, 0, 128, 64, SSD1306\_WHITE);

  }

  display.display();

  delay(40);

}