#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

};

// 粒子结构

struct Particle {

float x, y;

float vx, vy;

bool active;

};

#define MAX\_PARTICLES 50

Particle particles[MAX\_PARTICLES];

// 全局变量

unsigned long effectStartTime = 0;

int currentEffect = 0;

int scrollX = 0;

float angle = 0;

float scale = 1.0;

int waveOffset = 0;

void setup() {

Serial.begin(115200);

Wire.begin(6, 7); // SDA=GPIO6, SCL=GPIO7

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

Serial.println(F("SSD1306初始化失败"));

for(;;);

}

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(SSD1306\_WHITE);

display.setCursor(0, 0);

display.println(F("Multi-Effect Demo"));

display.println(F("Starting..."));

display.display();

delay(2000);

effectStartTime = millis();

Serial.println("多效果演示开始！");

initParticles();

}

void loop() {

unsigned long currentTime = millis();

unsigned long elapsed = currentTime - effectStartTime;

// 每6秒切换一次效果

if (elapsed >= 6000) {

currentEffect = (currentEffect + 1) % 5;

effectStartTime = currentTime;

elapsed = 0;

// 重置变量

scrollX = 0;

angle = 0;

scale = 1.0;

waveOffset = 0;

Serial.print("切换到效果: ");

Serial.println(currentEffect);

if (currentEffect == 3) {

initParticles(); // 重新初始化粒子

}

}

// 根据当前效果执行对应动画

switch(currentEffect) {

case 0:

effectA\_ScrollBreathing(elapsed); // 水平滚动+呼吸灯

break;

case 1:

effectB\_ScaleRotateFade(elapsed); // 缩放+旋转+淡入淡出

break;

case 2:

effectC\_WaveScroll(elapsed); // 波浪+滚动

break;

case 3:

effectD\_ParticleEffect(elapsed); // 粒子聚合/离散

break;

case 4:

effectE\_RandomMix(elapsed); // 随机混合效果

break;

}

}

// ========== 效果A: 水平滚动 + 呼吸灯 ==========

void effectA\_ScrollBreathing(unsigned long elapsed) {

display.clearDisplay();

// 水平滚动

display.drawBitmap(scrollX, 0, bitmap, 128, 64, SSD1306\_WHITE);

scrollX -= 2;

if (scrollX < -128) scrollX = 128;

// 呼吸灯效果（使用正弦波）

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B: 缩放 + 旋转 + 淡入淡出 ==========

void effectB\_ScaleRotateFade(unsigned long elapsed) {

display.clearDisplay();

// 计算缩放比例（0.5-1.5之间变化）

scale = 1.0 + 0.5 \* sin(elapsed / 800.0);

// 计算旋转角度

angle += 0.05;

if (angle > 6.28) angle = 0;

// 简化：只做缩放效果（完整旋转需要复杂的矩阵运算）

int w = 128 \* scale;

int h = 64 \* scale;

int x = (128 - w) / 2;

int y = (64 - h) / 2;

// 绘制缩放后的图像（使用简单采样）

for (int dy = 0; dy < 64; dy++) {

for (int dx = 0; dx < 128; dx++) {

int sx = (dx - 64) / scale + 64;

int sy = (dy - 32) / scale + 32;

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

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

int bitIndex = 7 - (sx % 8);

if (byteIndex < 1024 && (pgm\_read\_byte(&bitmap[byteIndex]) & (1 << bitIndex))) {

display.drawPixel(dx, dy, SSD1306\_WHITE);

}

}

}

}

// 淡入淡出

int brightness = (sin(elapsed / 600.0) \* 127.5) + 127.5;

display.ssd1306\_command(SSD1306\_SETCONTRAST);

display.ssd1306\_command(brightness);

display.display();

delay(40);

}

// ========== 效果C: 波浪效果 + 滚动 ==========

void effectC\_WaveScroll(unsigned long elapsed) {

display.clearDisplay();

waveOffset += 2;

scrollX -= 1;

if (scrollX < -128) scrollX = 128;

// 绘制带波浪效果的图像

for (int y = 0; y < 64; y++) {

// 计算波浪偏移

int waveShift = sin((y + waveOffset) \* 0.1) \* 8;

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

int srcX = (x - scrollX - waveShift + 256) % 128;

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

int bitIndex = 7 - (srcX % 8);

if (byteIndex < 1024 && (pgm\_read\_byte(&bitmap[byteIndex]) & (1 << bitIndex))) {

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

}

}

}

display.display();

delay(30);

}

// ========== 效果D: 粒子聚合/离散 ==========

void initParticles() {

// 从图像中提取粒子位置

int particleCount = 0;

for (int y = 0; y < 64 && particleCount < MAX\_PARTICLES; y += 4) {

for (int x = 0; x < 128 && particleCount < MAX\_PARTICLES; x += 4) {

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

int bitIndex = 7 - (x % 8);

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

particles[particleCount].x = x;

particles[particleCount].y = y;

particles[particleCount].vx = (random(-20, 20)) / 10.0;

particles[particleCount].vy = (random(-20, 20)) / 10.0;

particles[particleCount].active = true;

particleCount++;

}

}

}

// 填充剩余粒子

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

particles[i].active = false;

}

}

void effectD\_ParticleEffect(unsigned long elapsed) {

display.clearDisplay();

// 前3秒离散，后3秒聚合

bool isGathering = (elapsed % 6000) > 3000;

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

if (!particles[i].active) continue;

if (isGathering) {

// 聚合：粒子向原始位置移动

float targetX = (i % 32) \* 4;

float targetY = (i / 32) \* 16;

particles[i].vx = (targetX - particles[i].x) \* 0.05;

particles[i].vy = (targetY - particles[i].y) \* 0.05;

}

// 更新位置

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

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

// 边界检查（离散时）

if (!isGathering) {

if (particles[i].x < 0 || particles[i].x > 127) particles[i].vx \*= -1;

if (particles[i].y < 0 || particles[i].y > 63) particles[i].vy \*= -1;

particles[i].x = constrain(particles[i].x, 0, 127);

particles[i].y = constrain(particles[i].y, 0, 63);

}

// 绘制粒子

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

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

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

}

display.display();

delay(30);

}

// ========== 效果E: 随机混合效果 ==========

void effectE\_RandomMix(unsigned long elapsed) {

// 每2秒切换一个子效果

int subEffect = (elapsed / 2000) % 3;

switch(subEffect) {

case 0: // 闪烁

if ((elapsed / 300) % 2 == 0) {

display.clearDisplay();

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

display.display();

} else {

display.clearDisplay();

display.display();

}

delay(50);

break;

case 1: // 快速滚动

display.clearDisplay();

scrollX -= 5;

if (scrollX < -128) scrollX = 128;

display.drawBitmap(scrollX, 0, bitmap, 128, 64, SSD1306\_WHITE);

display.display();

delay(20);

break;

case 2: // 百叶窗效果

display.clearDisplay();

int lineOffset = (elapsed / 50) % 8;

for (int y = lineOffset; y < 64; y += 8) {

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

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

int bitIndex = 7 - (x % 8);

if (byteIndex < 1024 && (pgm\_read\_byte(&bitmap[byteIndex]) & (1 << bitIndex))) {

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

}

}

}

display.display();

delay(30);

break;

}

}