#include <SPI.h>

#define XAXIS 0

#define YAXIS 1

#define ZAXIS 2

#define POS\_X 0

#define NEG\_X 1

#define POS\_Z 2

#define NEG\_Z 3

#define POS\_Y 4

#define NEG\_Y 5

#define BUTTON\_PIN 8

#define RED\_LED 5

#define GREEN\_LED 7

#define TOTAL\_EFFECTS 8

#define RAIN 0

#define PLANE\_BOING 1

#define SEND\_VOXELS 2

#define WOOP\_WOOP 3

#define CUBE\_JUMP 4

#define GLOW 5

#define TEXT 6

#define LIT 7

#define RAIN\_TIME 260

#define PLANE\_BOING\_TIME 220

#define SEND\_VOXELS\_TIME 140

#define WOOP\_WOOP\_TIME 350

#define CUBE\_JUMP\_TIME 200

#define GLOW\_TIME 8

#define TEXT\_TIME 300

#define CLOCK\_TIME 500

uint8\_t characters[10][8] = {

{0x3C, 0x42, 0x42, 0x42, 0x42, 0x42, 0x42, 0x3C}, //0

{0x10, 0x18, 0x14, 0x10, 0x10, 0x10, 0x10, 0x3C}, //1

{0x3C, 0x42, 0x40, 0x40, 0x3C, 0x02, 0x02, 0x7E}, //2

{0x3C, 0x40, 0x40, 0x3C, 0x40, 0x40, 0x42, 0x3C}, //3

{0x22, 0x22, 0x22, 0x22, 0x7E, 0x20, 0x20, 0x20}, //4

{0x7E, 0x02, 0x02, 0x3E, 0x40, 0x40, 0x42, 0x3C}, //5

{0x3C, 0x02, 0x02, 0x3E, 0x42, 0x42, 0x42, 0x3C}, //6

{0x3C, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40}, //7

{0x3C, 0x42, 0x42, 0x3C, 0x42, 0x42, 0x42, 0x3C}, //8

{0x3C, 0x42, 0x42, 0x42, 0x3C, 0x40, 0x40, 0x3C}, //9

};

uint8\_t cube[8][8];

uint8\_t currentEffect;

uint16\_t timer;

uint64\_t randomTimer;

bool loading;

void setup() {

loading = true;

randomTimer = 0;

currentEffect = RAIN;

SPI.begin();

SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI\_MODE0));

pinMode(BUTTON\_PIN, INPUT\_PULLUP);

pinMode(RED\_LED, OUTPUT);

pinMode(GREEN\_LED, OUTPUT);

randomSeed(analogRead(0));

digitalWrite(GREEN\_LED, HIGH);

}

void loop() {

randomTimer++;

if (digitalRead(BUTTON\_PIN) == LOW) {

clearCube();

loading = true;

timer = 0;

currentEffect++;

if (currentEffect == TOTAL\_EFFECTS) {

currentEffect = 0;

}

randomSeed(randomTimer);

randomTimer = 0;

digitalWrite(RED\_LED, HIGH);

digitalWrite(GREEN\_LED, LOW);

delay(500);

digitalWrite(RED\_LED, LOW);

digitalWrite(GREEN\_LED, HIGH);

}

switch (currentEffect) {

case RAIN: rain(); break;

case PLANE\_BOING: planeBoing(); break;

case SEND\_VOXELS: sendVoxels(); break;

case WOOP\_WOOP: woopWoop(); break;

case CUBE\_JUMP: cubeJump(); break;

case GLOW: glow(); break;

case TEXT: text("0123456789", 10); break;

case LIT: lit(); break;

default: rain();

}

renderCube();

}

void renderCube() {

for (uint8\_t i = 0; i < 8; i++) {

digitalWrite(SS, LOW);

SPI.transfer(0x01 << i);

for (uint8\_t j = 0; j < 8; j++) {

SPI.transfer(cube[i][j]);

}

digitalWrite(SS, HIGH);

}

}

void rain() {

if (loading) {

clearCube();

loading = false;

}

timer++;

if (timer > RAIN\_TIME) {

timer = 0;

shift(NEG\_Y);

uint8\_t numDrops = random(0, 5);

for (uint8\_t i = 0; i < numDrops; i++) {

setVoxel(random(0, 8), 7, random(0, 8));

}

}

}

uint8\_t planePosition = 0;

uint8\_t planeDirection = 0;

bool looped = false;

void planeBoing() {

if (loading) {

clearCube();

uint8\_t axis = random(0, 3);

planePosition = random(0, 2) \* 7;

setPlane(axis, planePosition);

if (axis == XAXIS) {

if (planePosition == 0) {

planeDirection = POS\_X;

} else {

planeDirection = NEG\_X;

}

} else if (axis == YAXIS) {

if (planePosition == 0) {

planeDirection = POS\_Y;

} else {

planeDirection = NEG\_Y;

}

} else if (axis == ZAXIS) {

if (planePosition == 0) {

planeDirection = POS\_Z;

} else {

planeDirection = NEG\_Z;

}

}

timer = 0;

looped = false;

loading = false;

}

timer++;

if (timer > PLANE\_BOING\_TIME) {

timer = 0;

shift(planeDirection);

if (planeDirection % 2 == 0) {

planePosition++;

if (planePosition == 7) {

if (looped) {

loading = true;

} else {

planeDirection++;

looped = true;

}

}

} else {

planePosition--;

if (planePosition == 0) {

if (looped) {

loading = true;

} else {

planeDirection--;

looped = true;

}

}

}

}

}

uint8\_t selX = 0;

uint8\_t selY = 0;

uint8\_t selZ = 0;

uint8\_t sendDirection = 0;

bool sending = false;

void sendVoxels() {

if (loading) {

clearCube();

for (uint8\_t x = 0; x < 8; x++) {

for (uint8\_t z = 0; z < 8; z++) {

setVoxel(x, random(0, 2) \* 7, z);

}

}

loading = false;

}

timer++;

if (timer > SEND\_VOXELS\_TIME) {

timer = 0;

if (!sending) {

selX = random(0, 8);

selZ = random(0, 8);

if (getVoxel(selX, 0, selZ)) {

selY = 0;

sendDirection = POS\_Y;

} else if (getVoxel(selX, 7, selZ)) {

selY = 7;

sendDirection = NEG\_Y;

}

sending = true;

} else {

if (sendDirection == POS\_Y) {

selY++;

setVoxel(selX, selY, selZ);

clearVoxel(selX, selY - 1, selZ);

if (selY == 7) {

sending = false;

}

} else {

selY--;

setVoxel(selX, selY, selZ);

clearVoxel(selX, selY + 1, selZ);

if (selY == 0) {

sending = false;

}

}

}

}

}

uint8\_t cubeSize = 0;

bool cubeExpanding = true;

void woopWoop() {

if (loading) {

clearCube();

cubeSize = 2;

cubeExpanding = true;

loading = false;

}

timer++;

if (timer > WOOP\_WOOP\_TIME) {

timer = 0;

if (cubeExpanding) {

cubeSize += 2;

if (cubeSize == 8) {

cubeExpanding = false;

}

} else {

cubeSize -= 2;

if (cubeSize == 2) {

cubeExpanding = true;

}

}

clearCube();

drawCube(4 - cubeSize / 2, 4 - cubeSize / 2, 4 - cubeSize / 2, cubeSize);

}

}

uint8\_t xPos;

uint8\_t yPos;

uint8\_t zPos;

void cubeJump() {

if (loading) {

clearCube();

xPos = random(0, 2) \* 7;

yPos = random(0, 2) \* 7;

zPos = random(0, 2) \* 7;

cubeSize = 8;

cubeExpanding = false;

loading = false;

}

timer++;

if (timer > CUBE\_JUMP\_TIME) {

timer = 0;

clearCube();

if (xPos == 0 && yPos == 0 && zPos == 0) {

drawCube(xPos, yPos, zPos, cubeSize);

} else if (xPos == 7 && yPos == 7 && zPos == 7) {

drawCube(xPos + 1 - cubeSize, yPos + 1 - cubeSize, zPos + 1 - cubeSize, cubeSize);

} else if (xPos == 7 && yPos == 0 && zPos == 0) {

drawCube(xPos + 1 - cubeSize, yPos, zPos, cubeSize);

} else if (xPos == 0 && yPos == 7 && zPos == 0) {

drawCube(xPos, yPos + 1 - cubeSize, zPos, cubeSize);

} else if (xPos == 0 && yPos == 0 && zPos == 7) {

drawCube(xPos, yPos, zPos + 1 - cubeSize, cubeSize);

} else if (xPos == 7 && yPos == 7 && zPos == 0) {

drawCube(xPos + 1 - cubeSize, yPos + 1 - cubeSize, zPos, cubeSize);

} else if (xPos == 0 && yPos == 7 && zPos == 7) {

drawCube(xPos, yPos + 1 - cubeSize, zPos + 1 - cubeSize, cubeSize);

} else if (xPos == 7 && yPos == 0 && zPos == 7) {

drawCube(xPos + 1 - cubeSize, yPos, zPos + 1 - cubeSize, cubeSize);

}

if (cubeExpanding) {

cubeSize++;

if (cubeSize == 8) {

cubeExpanding = false;

xPos = random(0, 2) \* 7;

yPos = random(0, 2) \* 7;

zPos = random(0, 2) \* 7;

}

} else {

cubeSize--;

if (cubeSize == 1) {

cubeExpanding = true;

}

}

}

}

bool glowing;

uint16\_t glowCount = 0;

void glow() {

if (loading) {

clearCube();

glowCount = 0;

glowing = true;

loading = false;

}

timer++;

if (timer > GLOW\_TIME) {

timer = 0;

if (glowing) {

if (glowCount < 448) {

do {

selX = random(0, 8);

selY = random(0, 8);

selZ = random(0, 8);

} while (getVoxel(selX, selY, selZ));

setVoxel(selX, selY, selZ);

glowCount++;

} else if (glowCount < 512) {

lightCube();

glowCount++;

} else {

glowing = false;

glowCount = 0;

}

} else {

if (glowCount < 448) {

do {

selX = random(0, 8);

selY = random(0, 8);

selZ = random(0, 8);

} while (!getVoxel(selX, selY, selZ));

clearVoxel(selX, selY, selZ);

glowCount++;

} else {

clearCube();

glowing = true;

glowCount = 0;

}

}

}

}

uint8\_t charCounter = 0;

uint8\_t charPosition = 0;

void text(char string[], uint8\_t len) {

if (loading) {

clearCube();

charPosition = -1;

charCounter = 0;

loading = false;

}

timer++;

if (timer > TEXT\_TIME) {

timer = 0;

shift(NEG\_Z);

charPosition++;

if (charPosition == 7) {

charCounter++;

if (charCounter > len - 1) {

charCounter = 0;

}

charPosition = 0;

}

if (charPosition == 0) {

for (uint8\_t i = 0; i < 8; i++) {

cube[i][0] = characters[string[charCounter] - '0'][i];

}

}

}

}

void lit() {

if (loading) {

clearCube();

for(uint8\_t i=0; i<8; i++) {

for(uint8\_t j=0; j<8; j++) {

cube[i][j] = 0xFF;

}

}

loading = false;

}

}

void setVoxel(uint8\_t x, uint8\_t y, uint8\_t z) {

cube[7 - y][7 - z] |= (0x01 << x);

}

void clearVoxel(uint8\_t x, uint8\_t y, uint8\_t z) {

cube[7 - y][7 - z] ^= (0x01 << x);

}

bool getVoxel(uint8\_t x, uint8\_t y, uint8\_t z) {

return (cube[7 - y][7 - z] & (0x01 << x)) == (0x01 << x);

}

void setPlane(uint8\_t axis, uint8\_t i) {

for (uint8\_t j = 0; j < 8; j++) {

for (uint8\_t k = 0; k < 8; k++) {

if (axis == XAXIS) {

setVoxel(i, j, k);

} else if (axis == YAXIS) {

setVoxel(j, i, k);

} else if (axis == ZAXIS) {

setVoxel(j, k, i);

}

}

}

}

void shift(uint8\_t dir) {

if (dir == POS\_X) {

for (uint8\_t y = 0; y < 8; y++) {

for (uint8\_t z = 0; z < 8; z++) {

cube[y][z] = cube[y][z] << 1;

}

}

} else if (dir == NEG\_X) {

for (uint8\_t y = 0; y < 8; y++) {

for (uint8\_t z = 0; z < 8; z++) {

cube[y][z] = cube[y][z] >> 1;

}

}

} else if (dir == POS\_Y) {

for (uint8\_t y = 1; y < 8; y++) {

for (uint8\_t z = 0; z < 8; z++) {

cube[y - 1][z] = cube[y][z];

}

}

for (uint8\_t i = 0; i < 8; i++) {

cube[7][i] = 0;

}

} else if (dir == NEG\_Y) {

for (uint8\_t y = 7; y > 0; y--) {

for (uint8\_t z = 0; z < 8; z++) {

cube[y][z] = cube[y - 1][z];

}

}

for (uint8\_t i = 0; i < 8; i++) {

cube[0][i] = 0;

}

} else if (dir == POS\_Z) {

for (uint8\_t y = 0; y < 8; y++) {

for (uint8\_t z = 1; z < 8; z++) {

cube[y][z - 1] = cube[y][z];

}

}

for (uint8\_t i = 0; i < 8; i++) {

cube[i][7] = 0;

}

} else if (dir == NEG\_Z) {

for (uint8\_t y = 0; y < 8; y++) {

for (uint8\_t z = 7; z > 0; z--) {

cube[y][z] = cube[y][z - 1];

}

}

for (uint8\_t i = 0; i < 8; i++) {

cube[i][0] = 0;

}

}

}

void drawCube(uint8\_t x, uint8\_t y, uint8\_t z, uint8\_t s) {

for (uint8\_t i = 0; i < s; i++) {

setVoxel(x, y + i, z);

setVoxel(x + i, y, z);

setVoxel(x, y, z + i);

setVoxel(x + s - 1, y + i, z + s - 1);

setVoxel(x + i, y + s - 1, z + s - 1);

setVoxel(x + s - 1, y + s - 1, z + i);

setVoxel(x + s - 1, y + i, z);

setVoxel(x, y + i, z + s - 1);

setVoxel(x + i, y + s - 1, z);

setVoxel(x + i, y, z + s - 1);

setVoxel(x + s - 1, y, z + i);

setVoxel(x, y + s - 1, z + i);

}

}

void lightCube() {

for (uint8\_t i = 0; i < 8; i++) {

for (uint8\_t j = 0; j < 8; j++) {

cube[i][j] = 0xFF;

}

}

}

void clearCube() {

for (uint8\_t i = 0; i < 8; i++) {

for (uint8\_t j = 0; j < 8; j++) {

cube[i][j] = 0;

}

}

}