```
// Nokia 5110 LCD-Display (SIZE_HxSIZE_V Bildpunkte)
// TSJWang
// February 15, 2017
// Conway's game of life using Arduino Nano and a Nokia 5110 Display
// a button on reset for initialization and a button on pin 2 for lighting.
/**modifying this code so it can do wrap world**/

#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>

#define LCD_WIDTH 84
#define LCD_HEIGHT 48
#define LCD_HEIGHT8 (LCD_HEIGHT >> 3)

#define map(i, j) (i + LCD_WIDTH*j)
#define map_x(i, j, k) (i)
#define map_y(i, j, k) (j*8+k)
#define map_i(x, y) (x)
#define map_j(x, y) (y/8)
#define map_k(x, y) (y%8)

#define for_i for (int i = 0; i < LCD_WIDTH; i++)
#define for_j for (int j = 0; j < LCD_HEIGHT8; j++)
#define for_k for (int k = 0; k < 8; k++)
#define for_n for (int n = 0; n < 8; n++)

#define for_x for (int x = 0; x < LCD_WIDTH; x++)
#define for_y for (int y = 0; y < LCD_HEIGHT; y++)
#define for_y_a for (int y_a = y - 1; y_a <= y + 1; y_a++)
#define for_x_a for (int x_a = x - 1; x_a <= x + 1; x_a++)

#define bit_read(matrix, i, j, k) bitRead(matrix[map(i, j)], k)
#define bit_set(matrix, i, j, k) bitSet(matrix[map(i, j)], k)
#define bit_clear(matrix, i, j, k) bitClear(matrix[map(i, j)], k)

#define bit_set_xy(matrix, x, y) bit_set(matrix, map_i(x, y),  map_j(x, y), map_k(x, y))
#define bit_read_xy(matrix, x, y) bit_read(matrix, map_i(x, y),  map_j(x, y), map_k(x, y))
#define bit_clear_xy(matrix, x, y) bit_clear(matrix, map_i(x, y),  map_j(x, y), map_k(x, y));

#define pixel_set(i, j, k) display.drawPixel(i, j*8+k, BLACK)
#define pixel_unset(i, j, k) display.drawPixel(i, j*8+k, WHITE)
#define pixel_map(i, j, k) display.drawPixel(i, j*8+k, bit_read(matrix, i, j, k) == 1 ? BLACK :
WHITE)

#define between(n, a, b) ((n >= a) && (n <= b))

#define NORTH 0
#define NORTHEAST 1
#define EAST 2
#define SOUTHEAST 3
#define SOUTH 4
#define SOUTHWEST 5
#define WEST 6
```

```
#define NORTHWEST 7

#define RAND_RANGE 255

// Software SPI (slower updates, more flexible pin options):
// pin 13 - Serial clock out (SCLK)
// pin 11 - Serial data out (DIN)
// pin 4 - Data/Command select (D/C)
// pin 3 - LCD chip select (CS)
// pin 2 - LCD reset (RST)
Adafruit_PCD8544 display = Adafruit_PCD8544(8, 7, 5, 4, 3);
const int lite = 6;  //control for this interrupt pin will be 2
const int pot = A7;

// interrupt stuff
const byte interruptPin = 2;
volatile byte state = LOW;

// debouncing values
long debouncing_time = 200; //Debouncing Time in Milliseconds
volatile unsigned long last_micros;

unsigned char matrix[LCD_WIDTH * LCD_HEIGHT8] = {0};
unsigned char matrix_new[LCD_WIDTH * LCD_HEIGHT8] = {0};
unsigned char toggle = 0;

void initialize_matrix() {
  for_x {
    for_y {
      char rand = random(RAND_RANGE);
      if (between(rand, 0, RAND_RANGE / 2)) {
        bit_set_xy(matrix, x, y);
      }
    }
  }
}

// implemented using dubaiss' "neighbours XXX" algorithm
void evolve_matrix() {

  // new state
  unsigned char neighbour[8] = {0};
  unsigned char byte_cell;
  unsigned char byte_cell_new;
  unsigned char byte_cell_count;
  // calculate new state

  for_i {
    for_j {

      byte_cell = matrix[map(i, j)];

      /* * get each bit's neighbour one byte at a time * */
```

```
/* east and west */
neighbour[WEST] = 0b00000000;
if (i > 0) {
  neighbour[WEST] = matrix[map((i - 1), j)];
}

neighbour[EAST] = 0b00000000;
if (i < (LCD_WIDTH - 1)) {
  neighbour[EAST] = matrix[map((i + 1), j)];
}

/* north */
neighbour[NORTH] = 0b00000000;
neighbour[NORTHEAST] = 0b00000000;
neighbour[NORTHWEST] = 0b00000000;
if (j > 0) {
  neighbour[NORTH] = matrix[map(i, j - 1)];
  if (i > 0) {
    neighbour[NORTHWEST] = matrix[map(i - 1, j - 1)];
  }
  if (i < (LCD_WIDTH - 1)) {
    neighbour[NORTHEAST] = matrix[map(i + 1, j - 1)];
  }
}
neighbour[NORTH] = (byte_cell << 1) | ((neighbour[NORTH] & 0b10000000) >> 7);
neighbour[NORTHEAST] = (neighbour[EAST] << 1) | ((neighbour[NORTHEAST] & 0b1000000) >> 7);
neighbour[NORTHWEST] = (neighbour[WEST] << 1) | ((neighbour[NORTHWEST] & 0b1000000) >> 7);

/* south */
neighbour[SOUTH] = 0b00000000;
neighbour[SOUTHEAST] = 0b00000000;
neighbour[SOUTHWEST] = 0b00000000;
if (j < (LCD_HEIGHT8 - 1)) {
  neighbour[SOUTH] = matrix[map(i, j + 1)];
  if (i > 0) {
    neighbour[SOUTHWEST] = matrix[map(i - 1, j + 1)];
  }
  if (i < (LCD_WIDTH - 1)) {
    neighbour[SOUTHEAST] = matrix[map(i + 1, j + 1)];
  }
}
neighbour[SOUTH] = ((neighbour[SOUTH] & 0b00000001) << 7) | (byte_cell >> 1);
neighbour[SOUTHEAST] = ((neighbour[SOUTHEAST] & 0b00000001) << 7) | (neighbour[EAST] >> 1);
neighbour[SOUTHWEST] = ((neighbour[SOUTHWEST] & 0b00000001) << 7) | (neighbour[WEST] >> 1);

/* calculate each bit of next gen */
byte_cell_new = 0b00000000;

for_k {
  byte_cell_count = 0;
  for_n {
    byte_cell_count += (neighbour[n] & 0b00000001);
  }
```

```
        byte_cell_new >>= 1;
        if ((byte_cell_count == 3) || ((byte_cell_count == 2) && (byte_cell & 0b00000001))) {
          byte_cell_new |= 0b10000000;
        }

        for_n {
          neighbour[n] >>= 1;
        }
        byte_cell >>= 1;
      }

      matrix_new[map(i, j)] = byte_cell_new;
    }
  }

  // apply new state
  for_i {
    for_j {
      matrix[map(i, j)] = matrix_new[map(i, j)];
    }
  }

}

void update_display() {

  uint8_t color;
  for_i {
    for_j {
      for_k {
        pixel_map(i, j, k);
      }
    }
  }
  display.display();
}

void setup()   {
  pinMode(lite, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  pinMode(pot, INPUT);
  attachInterrupt(digitalPinToInterrupt(interruptPin), changelite, RISING);
  // random seed
  randomSeed(analogRead(0));
  // initialize display
  display.begin();
  // set contrast
  display.setContrast(56);
  // clears the screen and buffer
  display.clearDisplay();
  // initialize matrix
  initialize_matrix();
```

```
}


void loop() {
  update_display();
  evolve_matrix();
  delay(analogRead(pot));
}

/*******************This function controls lighting*********************/
void changelite(){
  if((long)(micros() - last_micros) >= debouncing_time * 1000) {
    state = !state;
    digitalWrite(lite, state);
    last_micros = micros();
  }
}
```