#include <FastLED.h>

/\*\* BASIC CONFIGURATION \*\*/

#define NUM\_LEDS 60

#define LED\_PIN 6

#define ANALOG\_READ 0

#define MIC\_LOW 0.0

#define MIC\_HIGH 737.0

#define AVGLEN 5

#define LONG\_SECTOR 50

#define HIGH 3

#define NORMAL 2

#define MSECS 30 \* 1000

#define CYCLES MSECS / DELAY

#define DEV\_THRESH 0.8

#define DELAY 1

float fscale(float, float, float, float, float, float);

void insert(int, int \*, int);

int compute\_average(int \*, int);

void visualize\_music();

void wave\_effect();

void fire\_effect();

int curshow = NUM\_LEDS;

int mode = 0;

int songmode = NORMAL;

unsigned long song\_avg;

int iter = 0;

float fade\_scale = 1.2;

CRGB leds[NUM\_LEDS];

int avgs[AVGLEN] = {-1};

int long\_avg[LONG\_SECTOR] = {-1};

struct time\_keeping {

  unsigned long times\_start;

  short times;

};

struct color {

  int r;

  int g;

  int b;

};

struct time\_keeping high;

struct color Color;

void setup() {

  Serial.begin(9600);

  FastLED.addLeds<NEOPIXEL, LED\_PIN>(leds, NUM\_LEDS);

  FastLED.clear();

  FastLED.show();

  for (int i = 0; i < AVGLEN; i++) insert(250, avgs, AVGLEN);

  high.times = 0;

  high.times\_start = millis();

  Color = {0, 0, 1};

}

void loop() {

  switch (mode) {

    case 0:

      visualize\_music();

      break;

    case 1:

      wave\_effect();

      break;

    case 2:

      fire\_effect();

      break;

  }

  delay(DELAY);

}

void check\_high(int avg) {

  if (avg > (song\_avg / iter \* 1.1)) {

    if (high.times != 0 && millis() - high.times\_start > 200.0) {

      high.times = 0;

      songmode = NORMAL;

    } else {

      high.times\_start = millis();

      high.times++;

    }

  }

  if (high.times > 30 && millis() - high.times\_start < 50.0)

    songmode = HIGH;

  else if (millis() - high.times\_start > 200) {

    high.times = 0;

    songmode = NORMAL;

  }

}

void visualize\_music() {

  int sensor\_value = analogRead(ANALOG\_READ);

  if (sensor\_value == 0) return;

  int mapped = (int)fscale(MIC\_LOW, MIC\_HIGH, MIC\_LOW, MIC\_HIGH, sensor\_value, 2.0);

  int avg = compute\_average(avgs, AVGLEN);

  if (((avg - mapped) > avg \* DEV\_THRESH)) return;

  insert(mapped, avgs, AVGLEN);

  insert(avg, long\_avg, LONG\_SECTOR);

  song\_avg += avg;

  iter++;

  if (iter > CYCLES) {

    song\_avg = song\_avg / iter;

    iter = 1;

  }

  int longavg = compute\_average(long\_avg, LONG\_SECTOR);

  check\_high(longavg);

  fade\_scale = (songmode == HIGH) ? 3 : 2;

  Color = (songmode == HIGH) ? color{5, 3, -1} : color{-1, 2, 1};

  curshow = fscale(MIC\_LOW, MIC\_HIGH, 0.0, (float)NUM\_LEDS, (float)avg, -1);

  for (int i = 0; i < NUM\_LEDS; i++) {

    if (i < curshow) {

      leds[i].r = constrain(leds[i].r + Color.r, 0, 255);

      leds[i].g = constrain(leds[i].g + Color.g, 0, 255);

      leds[i].b = constrain(leds[i].b + Color.b, 0, 255);

    } else {

      leds[i].fadeToBlackBy(fade\_scale);

    }

  }

  FastLED.show();

}

void wave\_effect() {

  static uint8\_t hue = 0;

  for (int i = 0; i < NUM\_LEDS; i++) {

    leds[i] = CHSV(hue + (i \* 10), 255, 255);

  }

  hue++;

  FastLED.show();

  delay(30);

}

void fire\_effect() {

  for (int i = 0; i < NUM\_LEDS; i++) {

    leds[i] = CRGB(random8(200, 255), random8(50, 150), 0);

  }

  FastLED.show();

  delay(20);

}

int compute\_average(int \*avgs, int len) {

  int sum = 0;

  for (int i = 0; i < len; i++) sum += avgs[i];

  return sum / len;

}

void insert(int val, int \*avgs, int len) {

  for (int i = 1; i < len; i++) avgs[i - 1] = avgs[i];

  avgs[len - 1] = val;

}

float fscale(float originalMin, float originalMax, float newBegin, float newEnd, float inputValue, float curve) {

  float OriginalRange = originalMax - originalMin;

  float NewRange = newEnd - newBegin;

  float zeroRefCurVal = inputValue - originalMin;

  float normalizedCurVal = zeroRefCurVal / OriginalRange;

  float rangedValue = pow(normalizedCurVal, pow(10, curve \* -.1)) \* NewRange + newBegin;

  return rangedValue;

}