This pad builds on [[9Bn87r74CA/rev.0]], created by

This pad builds on [[hacklab-scanlineanimator/rev.6]], created by [unnamed author] & rambo

```
#include "FastLED.h"
// How many leds in your strip?
#define NUM_LEDS 60
// For led chips like Neopixels, which have a data line, ground, and power, you just
// need to define DATA_PIN.  For led chipsets that are SPI based (four wires - data, clock,
// ground, and power), like the LPD8806, define both DATA_PIN and CLOCK_PIN
#define DATA_PIN 11
#define CLOCK_PIN 10
// Define the array of leds
CRGB leds[NUM_LEDS];
CRGB c;
int start = 0;
void setup() {
  FastLED.addLeds<NEOPIXEL, DATA_PIN>(leds, NUM_LEDS);

}
void loop() {
  wave(0);
  wave(2);
  wave(2);
  wave(0);
  wave(3);
  wave(4);
  wave(20);
}
void wave(int additionalIntensity) {

  CRGB topColor = randomColor(150 + additionalIntensity, 50, 100, 20, 40 + additionalIntensity * 2, 40);
  CRGB midColor = randomColor(145, 30, 160, 40, 90, 40);
  CRGB botColor = randomColor(135, 50, 200, 20, 30, 30);
  CRGB mid2Color = randomColor(125, 40, 160, 40, 40 + additionalIntensity, 30);
  int overallDelay = random8(0, 100) + additionalIntensity * 3;
  int botDelay = random8(30, 60) + overallDelay;
  int midDelay = random8(20, 40) + overallDelay;
  int topDelay = random8(5, 20)  + overallDelay;
  int mid2Delay = random8(7, 30) + overallDelay;

  int steps = 64;
  int steps2 = 40;
  scrollSection(botDelay, steps, botColor);
  scrollSection(midDelay, steps, midColor);
  scrollSection(topDelay, steps, topColor);
  scrollSection(mid2Delay, steps, mid2Color);
}
CRGB randomColor(int hue, int hueVar, int sat, int satVar, int val, int valVar) {
  int h = randomVar(hue, hueVar);
  int s = randomVar(sat, satVar);
  int v = randomVar(val, valVar);
  return CHSV(h, s, v);
```

```
}
int randomVar(int value, int variation) {
  //int result = value - variation / 2 + (random16(variation) + random16(variation)) / 2;
  int result = value - variation / 2 + random(variation);
  if (result <= 0) return 0;
  else if (result >= 255) return 255;
  else return result;
}
void scrollSection(int stepDelay, int steps, CRGB toColor) {
  CRGB fromColor = leds[0];
  CRGB color;

  int stepSize = 256 / steps;

  for(int i = 0; i <= 256; i += stepSize) {
    color.r = mix(i, fromColor.r, toColor.r);
    color.g = mix(i, fromColor.g, toColor.g);
    color.b = mix(i, fromColor.b, toColor.b);

    scrollPixel(color);
    FastLED.show();

    delay(stepDelay);
  }
}
uint8_t mix(uint16_t t, uint8_t from, uint8_t to) {
  return uint8_t( (((to * t) + ((256 - t) * from) ) >> 8) );
}
void scrollPixel(CRGB color) {

  for(int i = NUM_LEDS; i > 0; i--) {

    leds[i].r = leds[i - 1].r;
    leds[i].g = leds[i - 1].g;
    leds[i].b = leds[i - 1].b;
  }
  leds[0].r = color.r;
  leds[0].g = color.g;
  leds[0].b = color.b;

}
```