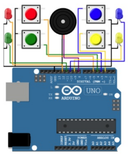
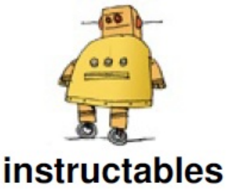
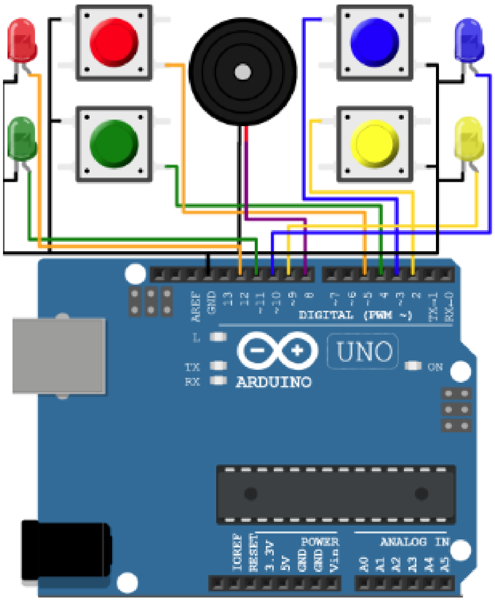


instructables Square Tiling WOKWI Online Arduino Simulato Instructions

Contents



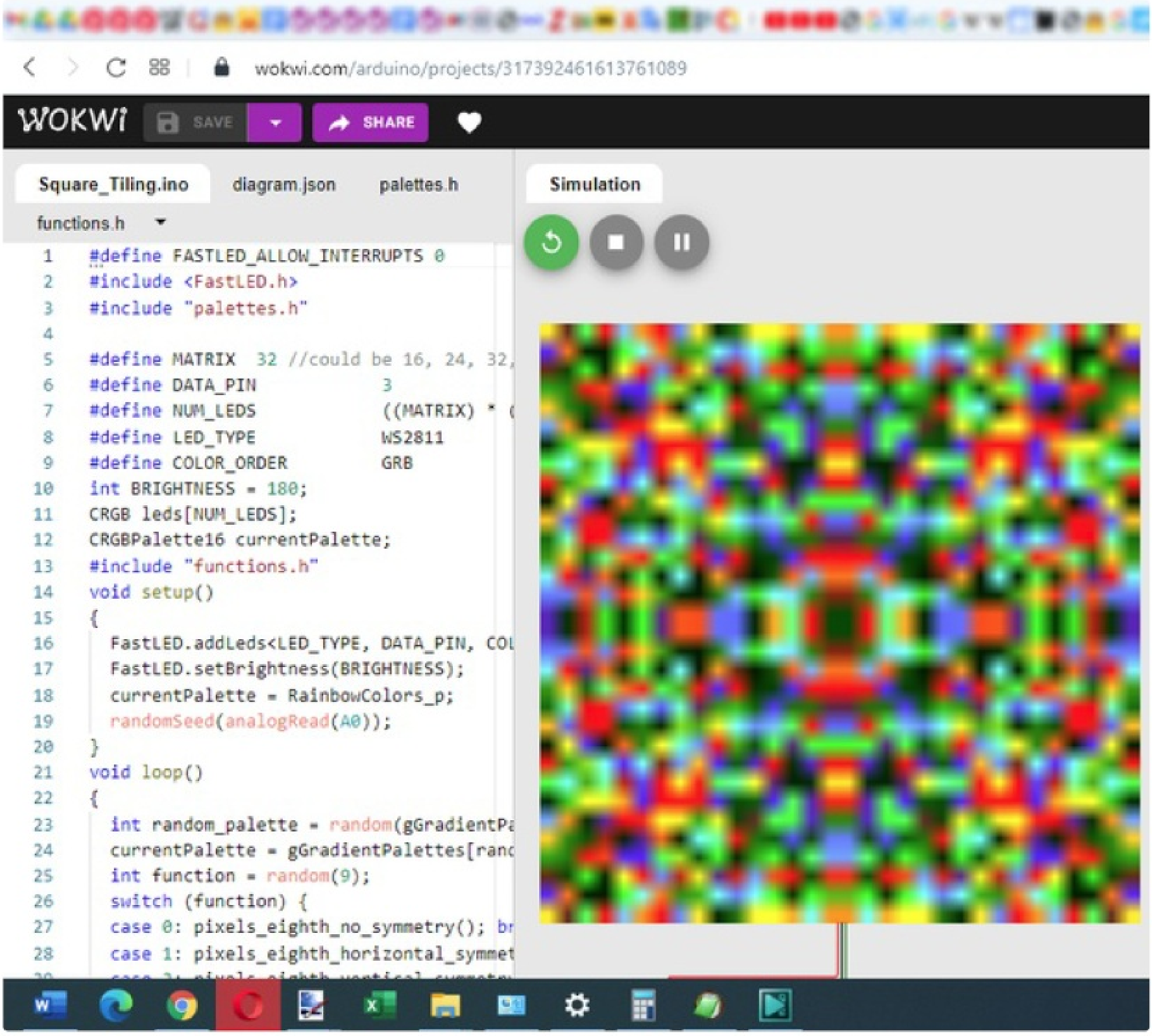
instructables Square Tiling WOKWI Online Arduino Simulato



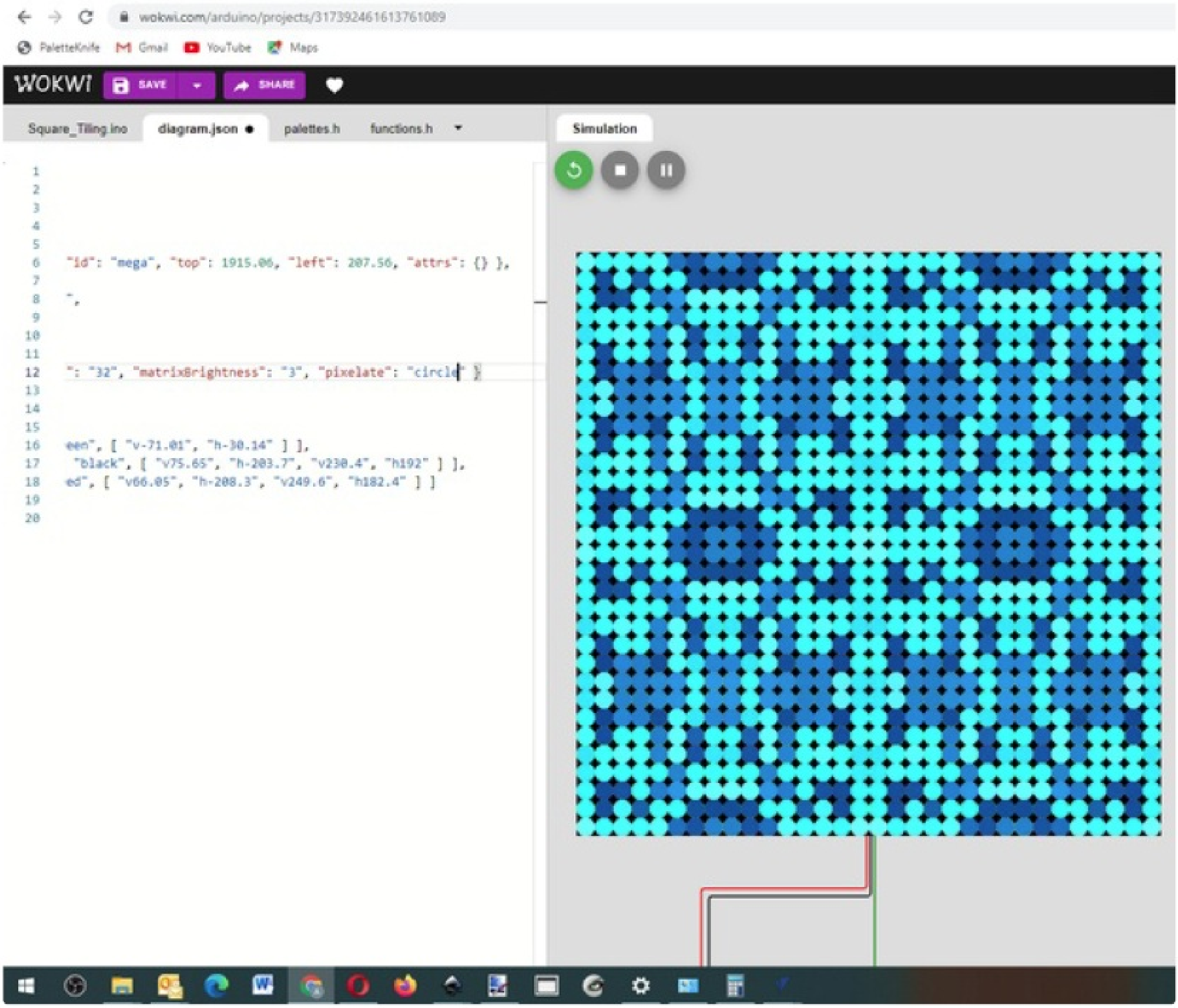
Square Tiling in WOKWI – the Online Arduino Simulator

by andrei.erdei A few days ago I published an article about tiling with the help of some right-angle triangles (Tetrakis Square Tiling With WS2812 LEDs) and I asked myself the question, I think somewhat justified, how would it look like built with the help of WS2812 LED matrices. There are very cheap 8×8 LED arrays, but 16×16 ones can also be found cheaply. Four such matrices could make an excellent display. But the practical realization, from scratch, of the whole ensemble would take quite a long time and honestly I would not put time and money in such a project before I know, at least roughly, what the result would look like. Luckily for me, and for many others, there are solutions. They are called simulators. So I would like to present to you the simulation of a generator of colored geometric figures, I think very attractive, and which are nothing more than a regular tiling application, more precisely regular square tiling. I used WOKWI, it was my first time using it, and in the end, it wasn't as hard as I expected.

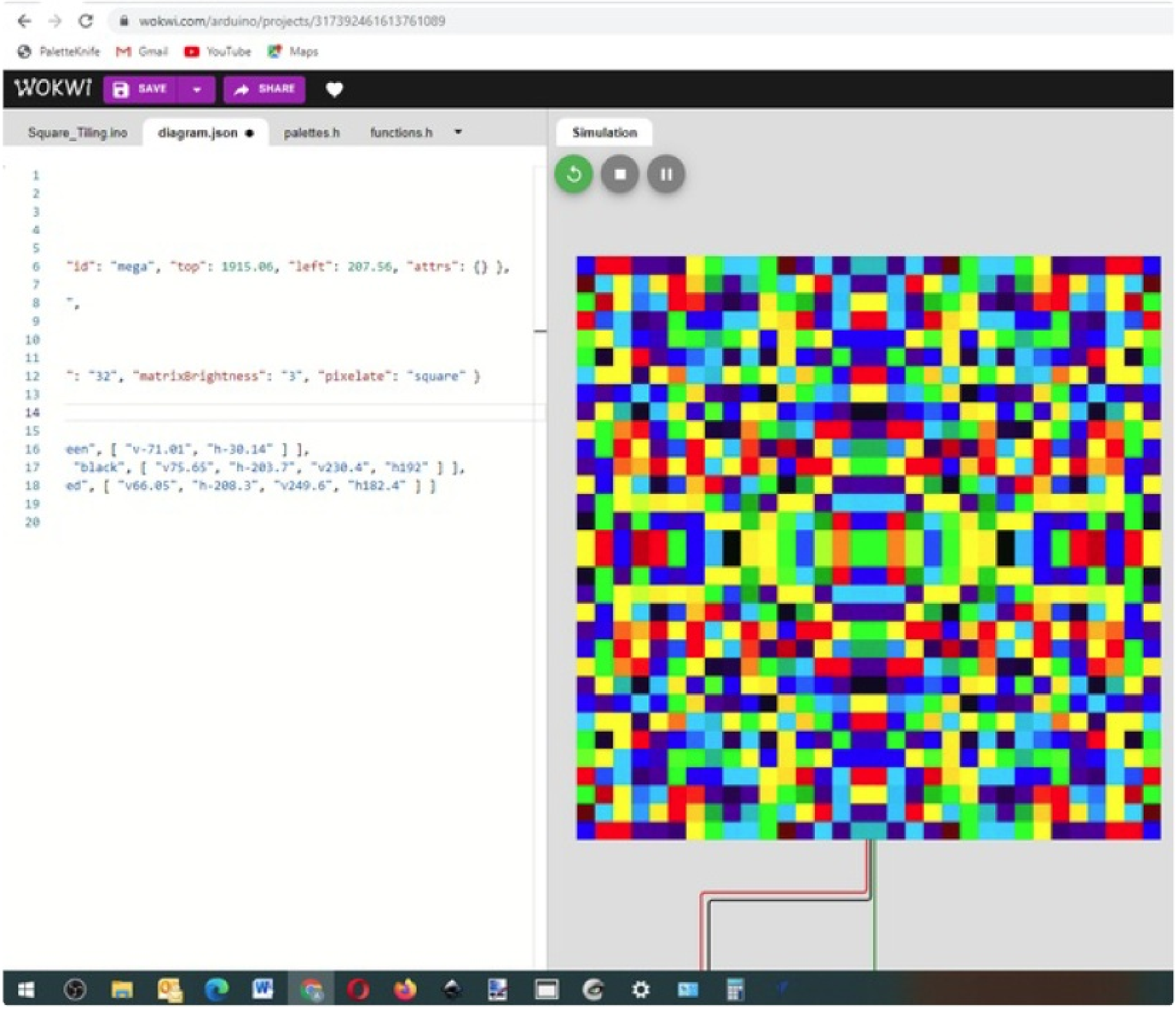
INSTALLATION INSTRUCTIONS



```
#define FASTLED_ALLOW_INTERRUPTS 0
#include <FastLED.h>
#include "palettes.h"

#define MATRIX  32 //could be 16, 24, 32,
#define DATA_PIN           3
#define NUM_LEDS           ((MATRIX) * (
#define LED_TYPE           WS2811
#define COLOR_ORDER        GRB
int BRIGHTNESS = 180;
CRGB leds[NUM_LEDS];
CRGBPalette16 currentPalette;
#include "functions.h"
void setup()
{
  FastLED.addLeds<LED_TYPE, DATA_PIN, COL
  FastLED.setBrightness(BRIGHTNESS);
  currentPalette = RainbowColors_p;
  randomSeed(analogRead(A0));
}
void loop()
{
  int random_palette = random(gGradientPa
  currentPalette = gGradientPalettes[ranc
  int function = random(9);
  switch (function) {
  case 0: pixels_eighth_no_symmetry(); br
  case 1: pixels_eighth_horizontal_symmet
  case 2: pixels_eighth_vertical_symmetry
```



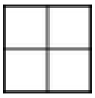
```





"id": "mega", "top": 1915.06, "left": 207.56, "attrs": {} },

",



": "32", "matrixBrightness": "3", "pixelate": "circle" }



een", [ "v-71.01", "h-30.14" ] ],
 "black", [ "v75.65", "h-203.7", "v230.4", "h192" ] ],
ed", [ "v66.05", "h-208.3", "v249.6", "h182.4" ] ]


```



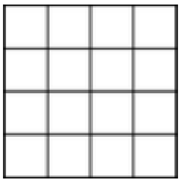
```





"id": "mega", "top": 1915.06, "left": 207.56, "attrs": {} },

",




": "32", "matrixBrightness": "3", "pixelate": "square" }



een", [ "v-71.01", "h-30.14" ] ],
 "black", [ "v75.65", "h-203.7", "v230.4", "h192" ] ],
ed", [ "v66.05", "h-208.3", "v249.6", "h182.4" ] ]


```

Concept

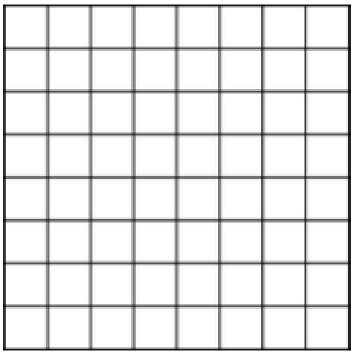
The idea I started from was very similar to the one in the “ Tetrakis Square Tiling With WS2812 LEDs ” project, except that instead of pieces of LED strips I used square LED matrices of different sizes but with the same number of LEDs horizontally and vertically to ease the programming. Also, another value that I considered is the “cell”. This is the group of LEDs that I will reflect horizontally and vertically in the LED array to generate symmetrical figures. The minimum cell would be a group of 4 LEDs, 2 rows and 2 columns.



The next cell for mirroring would result by doubling the number of LEDs horizontally and vertically, ie 4×4 LEDs (16 in total)



and finally, the third cell is obtained by again doubling, resulting 8×8 LEDs (ie 64).



This last cell would represent half the horizontal and vertical dimension of the LED matrix that we use, ie 16×16 LEDs. The following mirroring functions and default display types are shown:

- 2×2 cell without mirroring;
- 2×2 cell mirroring horizontally;
- 2×2 cell mirroring vertically;
- 2×2 cell mirroring horizontally and vertically;
- 4×4 cell without mirroring;
- 4×4 cell mirroring horizontally;
- 4×4 cell mirroring vertically;
- 4×4 cell mirroring horizontally and vertically;
- 8×8 cell mirroring horizontally and vertically;

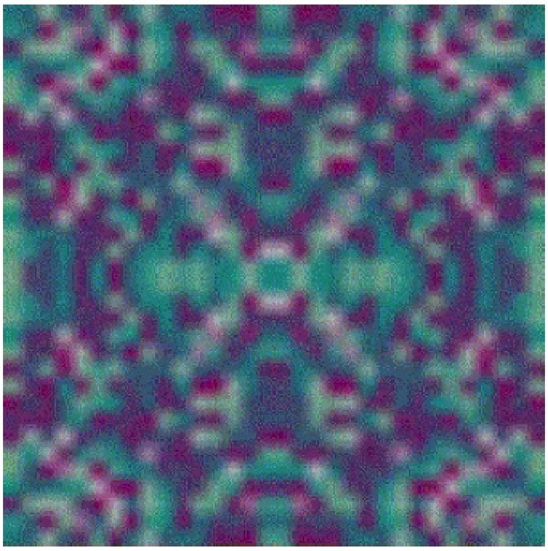
So a total of 9 functions
Following the same rules (taking into account the base cell) we can have the following dimensions for the LED matrix:

- 24×24 – ie cells with 3×3, 6×6, 12×12 LEDs
- 32×32 – that is 4×4, 8×8, 16×16
- 40×40 – that is 5×5, 10×10, 20×20
- 48×48 – that is 6×6, 12×12, 24×24

More than 48×48 (the next matrix is 56×56) doesn't work in the Wokwi simulator (maybe not enough memory? I don't know…)

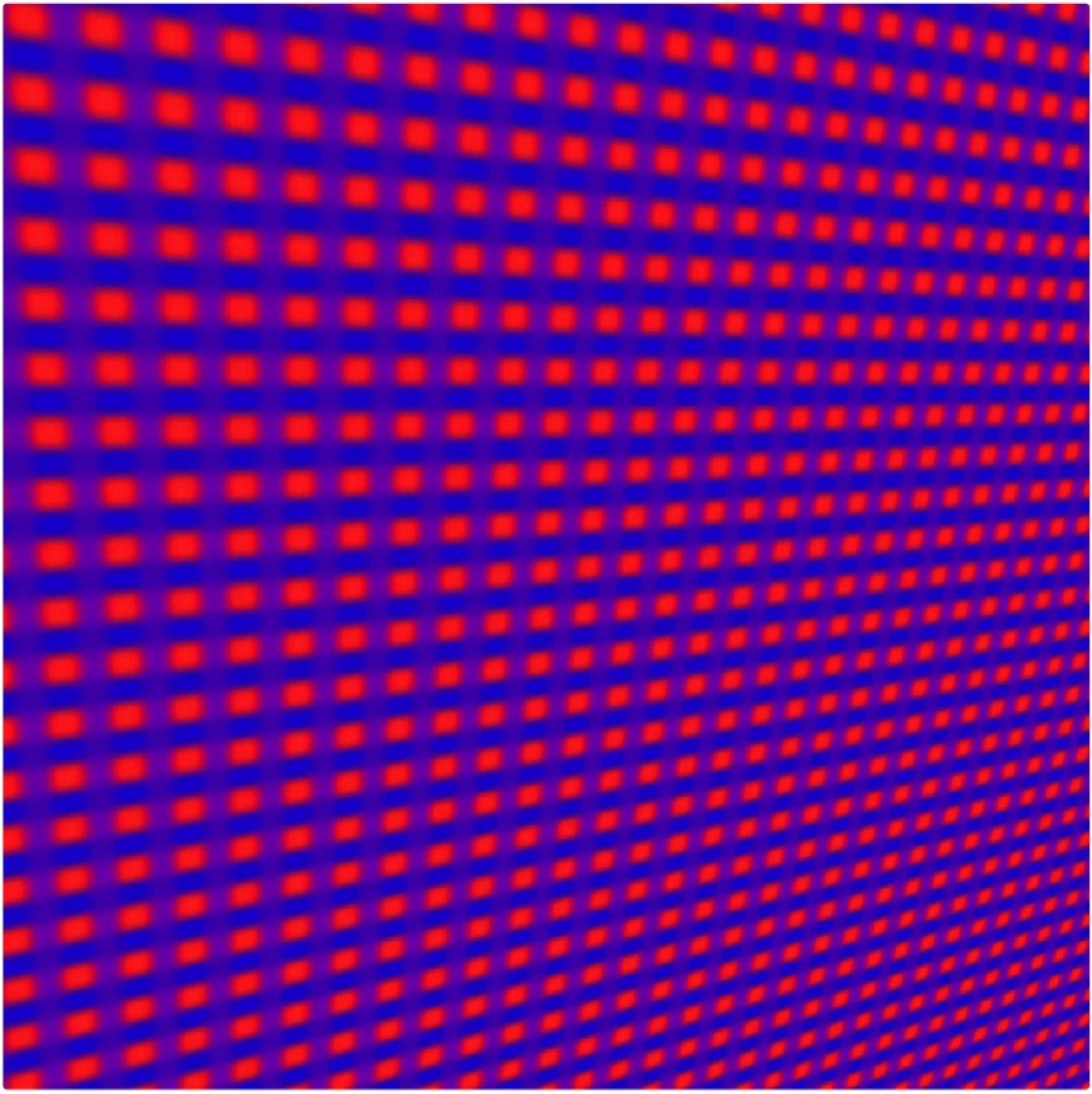
Execution

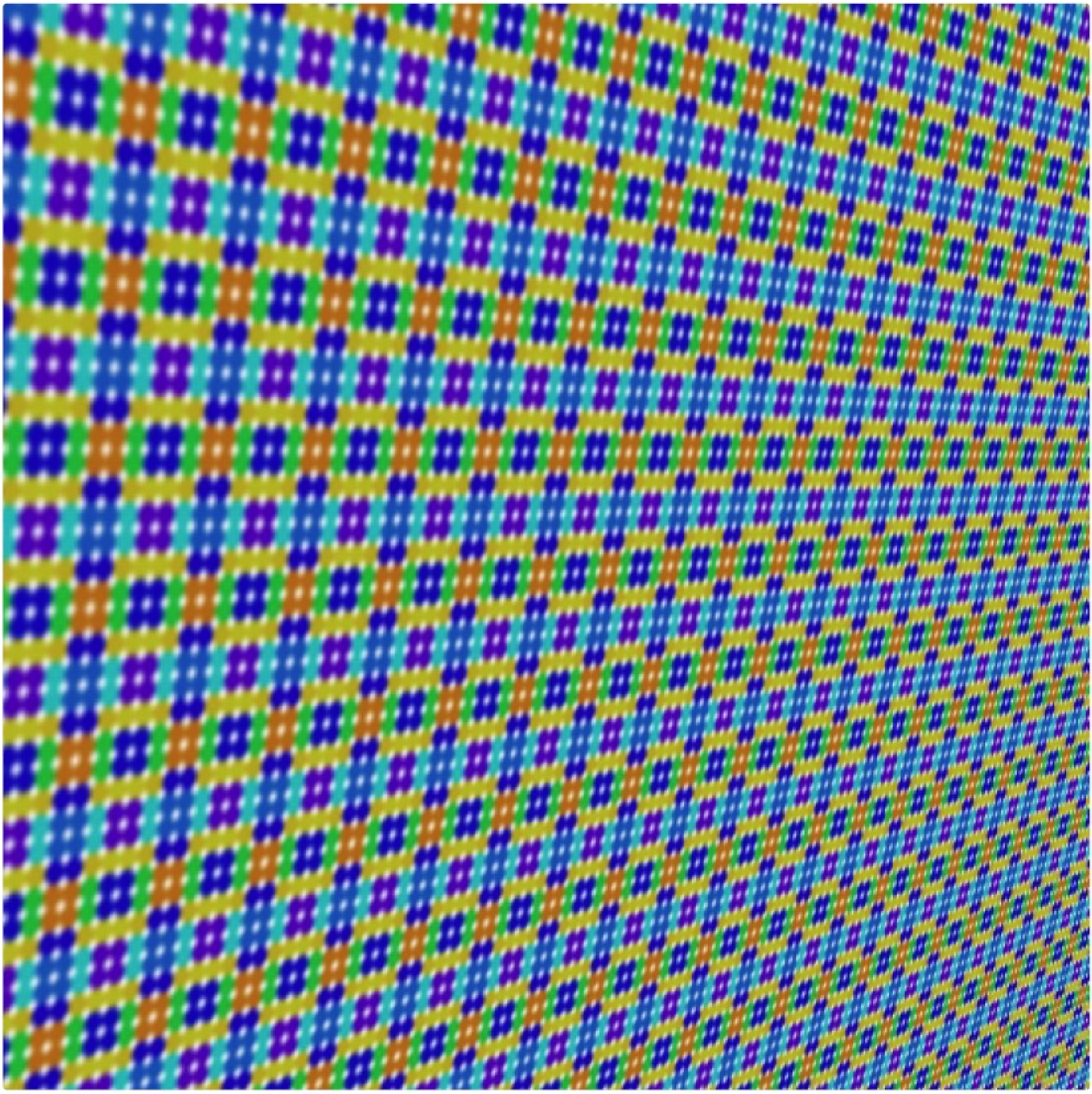
I signed in to the WOKWI site with my gmail account and opened a simulation example from the FastLED library examples – LEDFace. I saved a copy of this project to my projects in my new WOKWI account (top left menu "Save – Save a copy") I modified the "diagram.json" file, ie I deleted the three buttons. I renamed the ino file I added two files: palette.h and functions.h When running the simulation I can change the size of the LED array in the ino file, ie by changing the value of the MATRIX variable. I can also change the "pixelate" attribute of the "woke-neo pixel-canvas" component (try "", "circle", "square" to see how the simulation changes visually). I would like to point out here that I wanted to use a "woke-__alpha__-diffuser" component that I found in the " Fire Clock" project, to make the LED light diffusion as natural as possible but unfortunately, it did not work for me. In fact, the documentation at WOKWI is a bit sparse and quite unclear, however it is a great simulator and I really enjoyed working with it. I already had the source code from my project and adapting the code to square matrices was not diffcult at all and the fact that WOKWI works with the code that might be used in the future in the physical realization of the project is very helpful. And the result, as you can see in the gif below, is great!

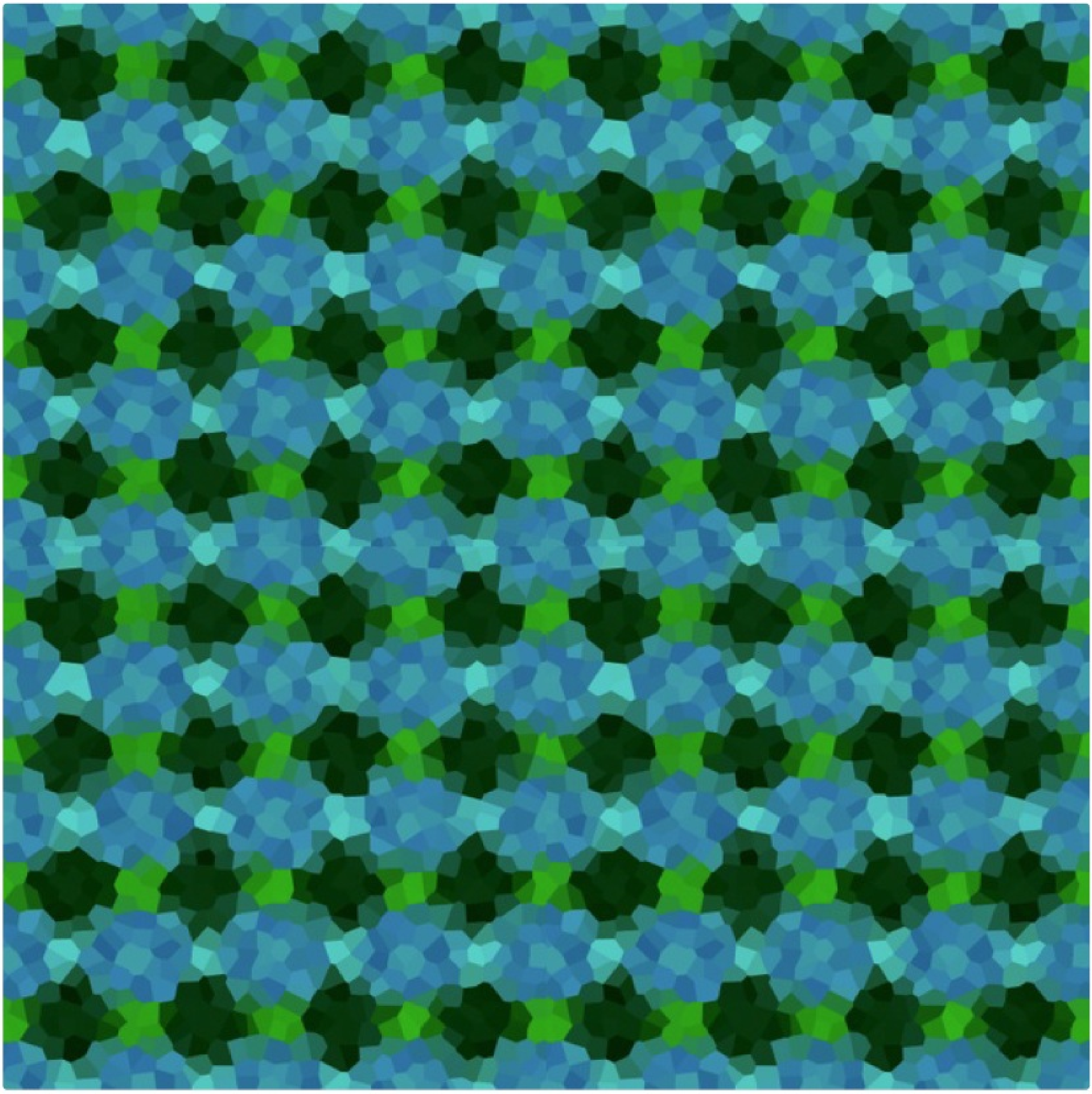


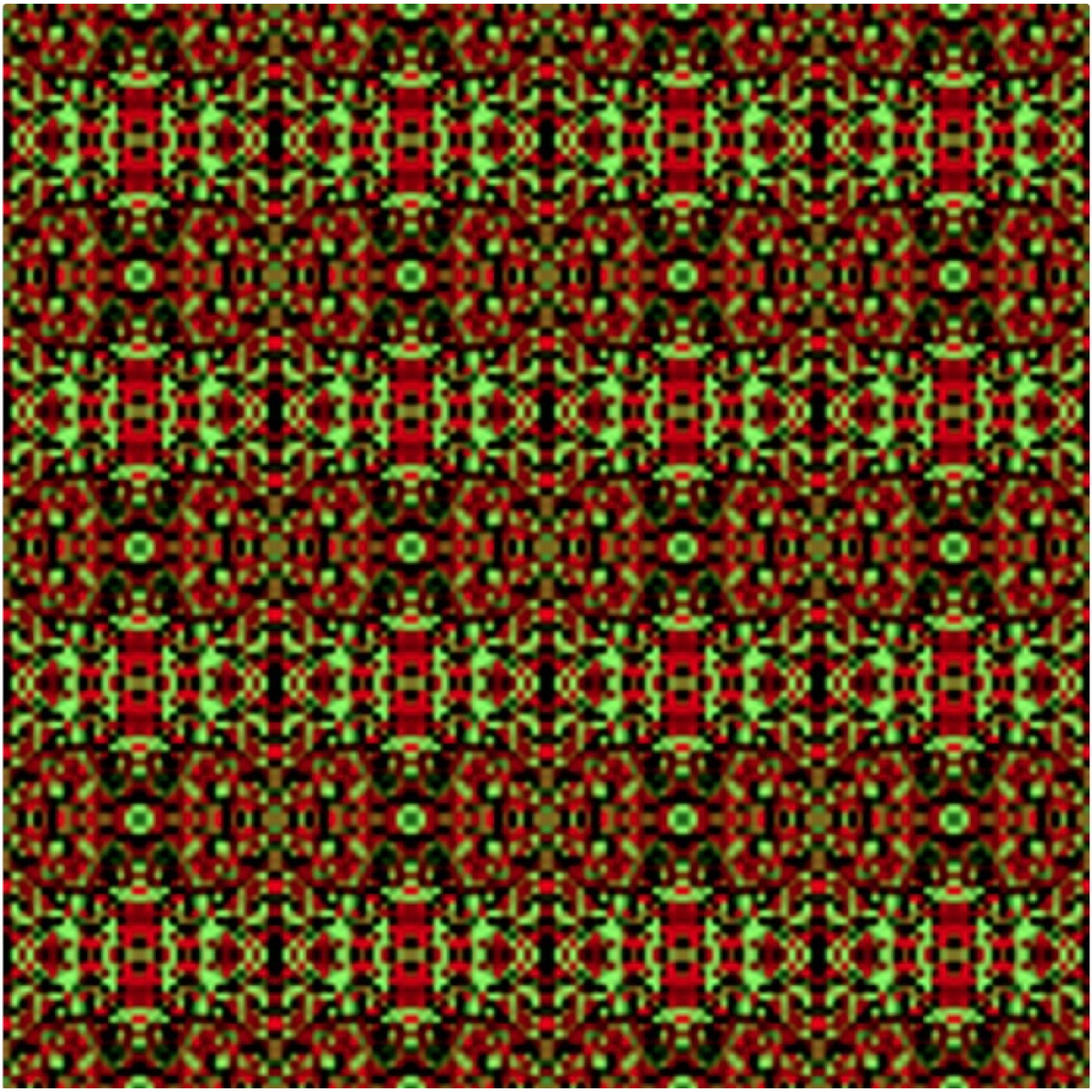
An Unusual Usage

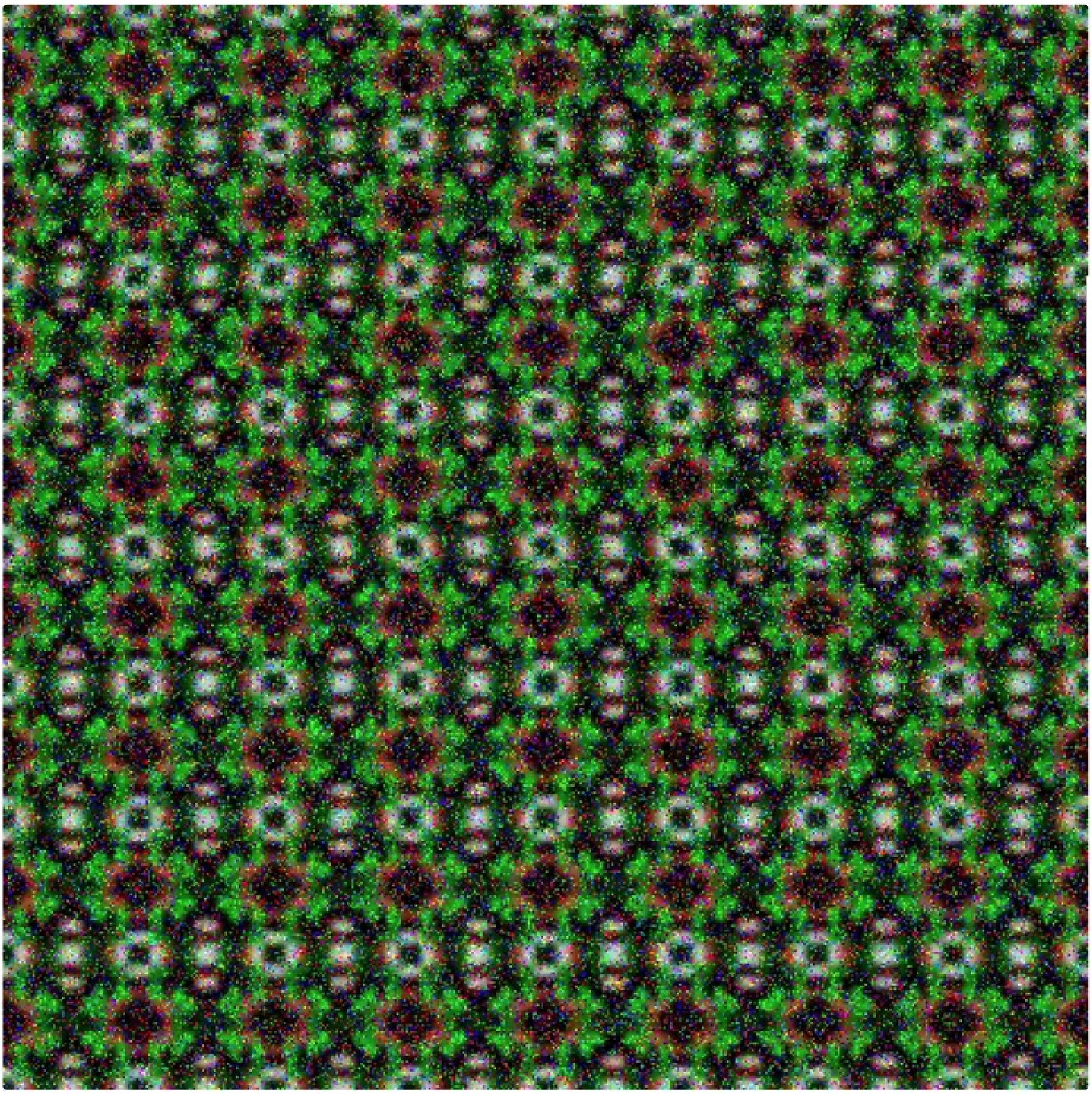
Seeing the results from the gif above, it occurred to me there might be a way to use the generated images from it. So I simply paused the simulation on an interesting pattern and with the help of paint.net, a freeware image processing program and applying some simple transformations and effects, I got interesting (and original) textures. You can see some of them attached above.

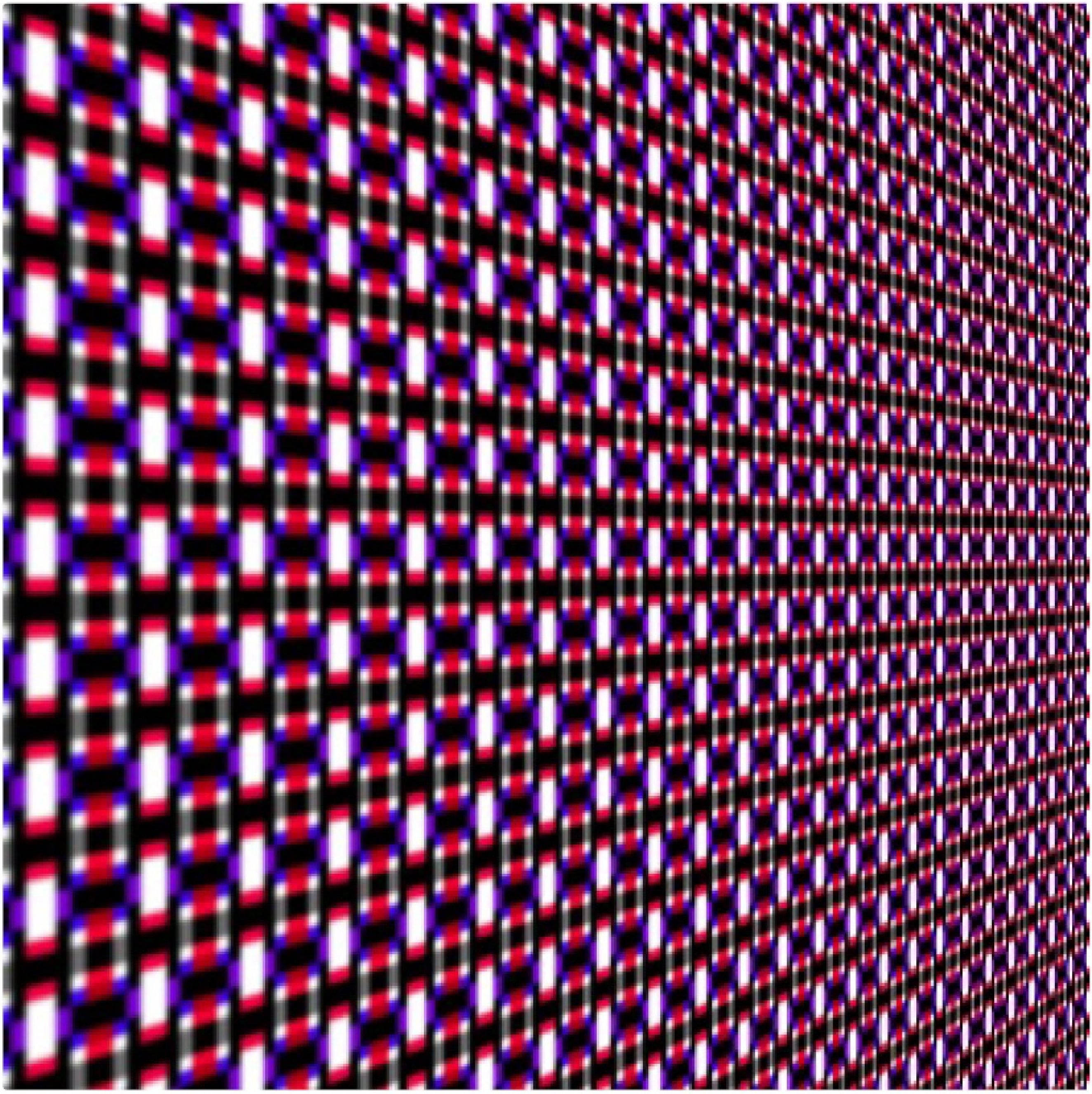


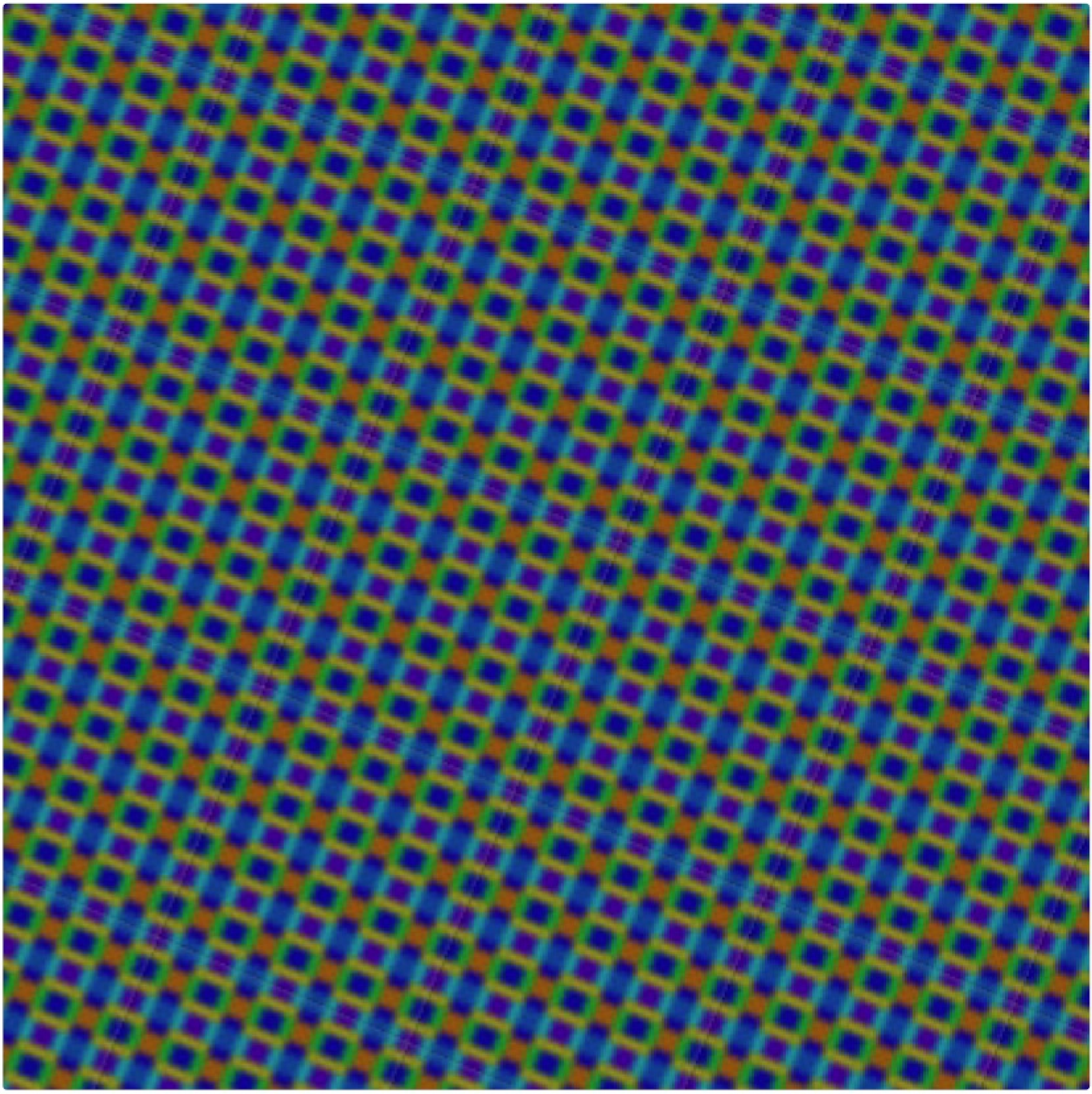


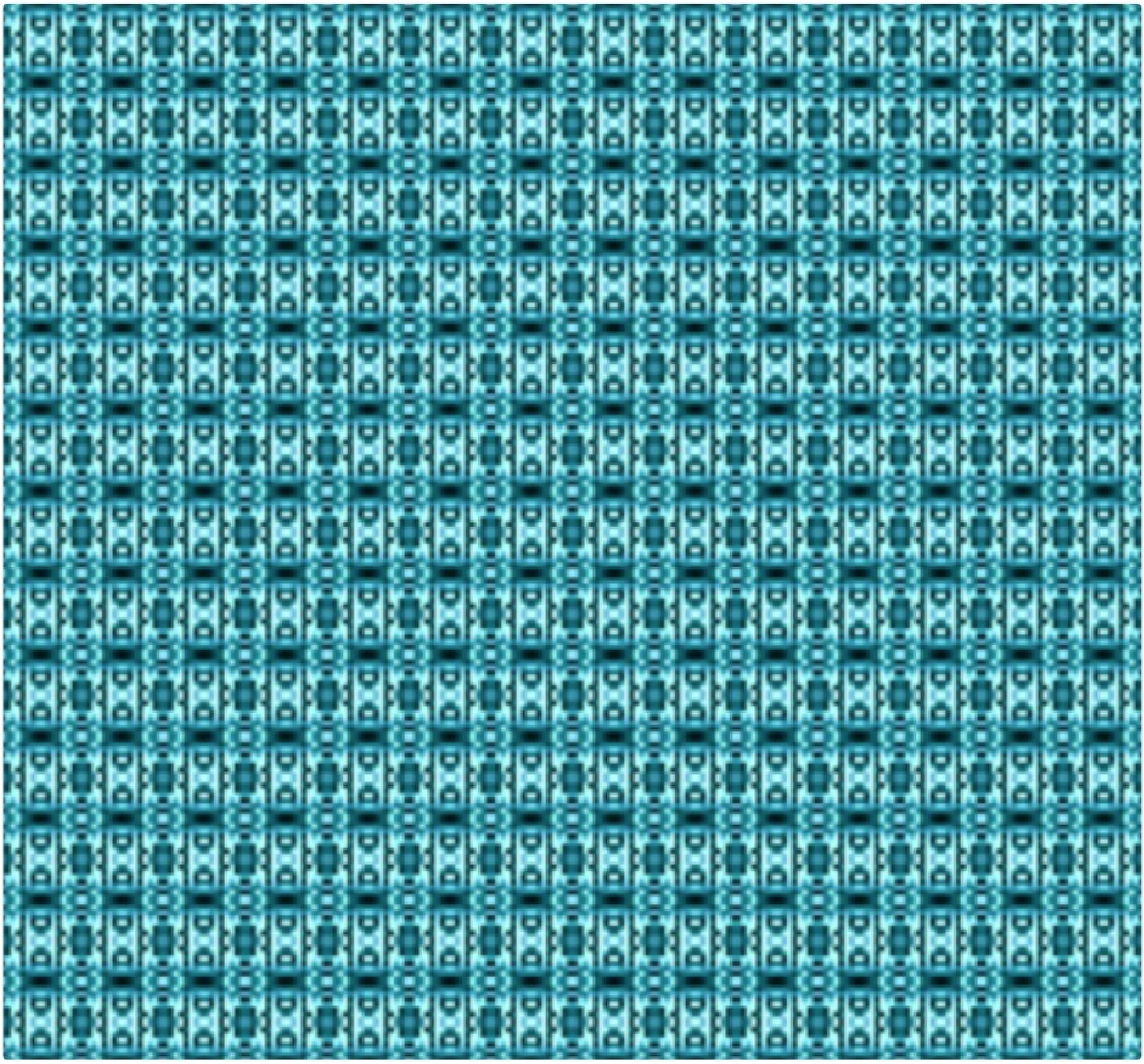


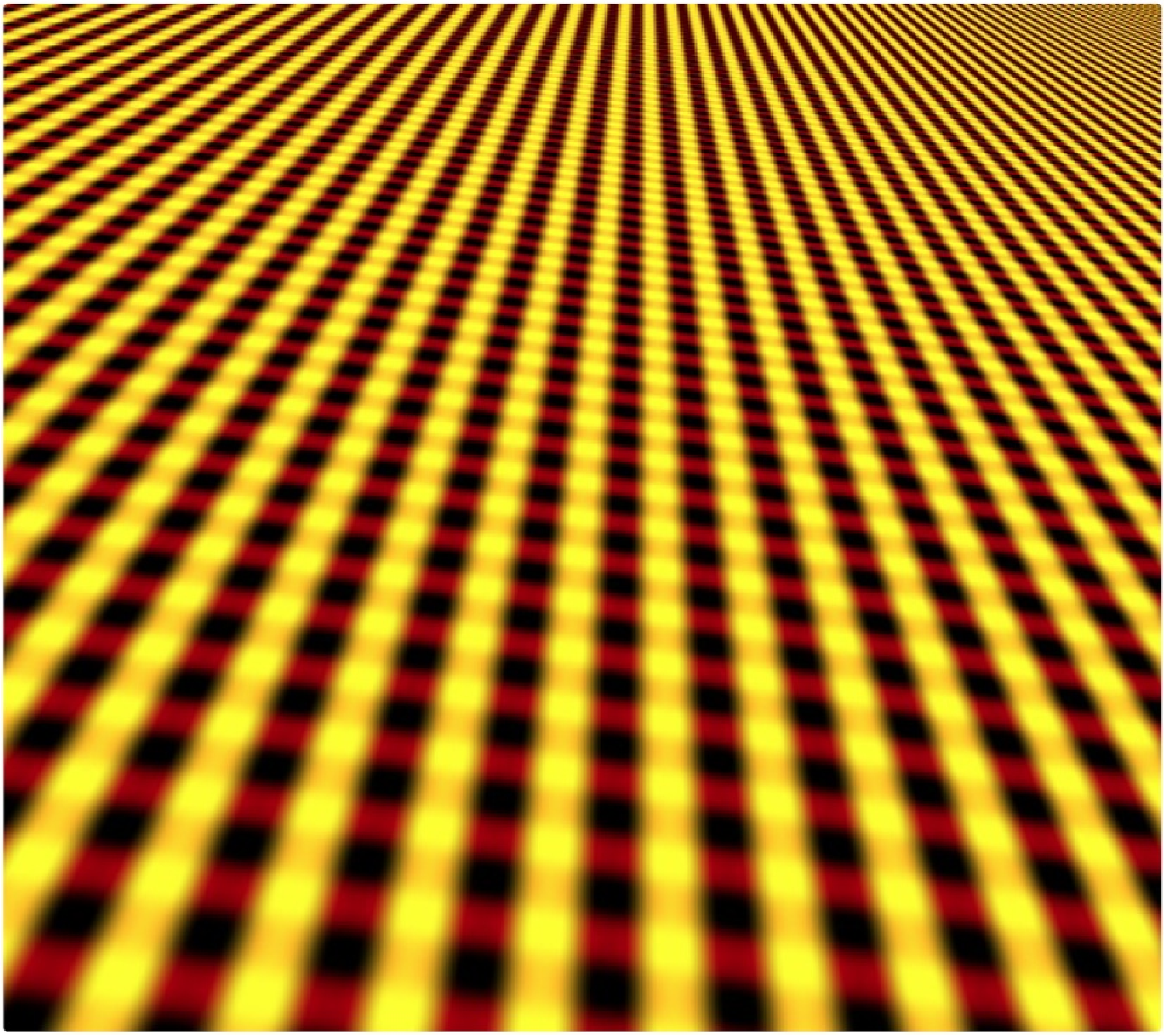












Square Tiling in WOKWI – the Online Arduino Simulator

Instead of Conclusions

Of course something is missing! I have to tell you the most important part of the article   Here is the link to the simulation on wokwi.com https://wokwi.com/arduino/projects/317392461613761089 And finally I look forward to your comments and your feedback.

Documents / Resources

	instructables Square Tiling WOKWI Online Arduino Simulato [pdf] Instructions Square Tiling WOKWI Online Arduino Simulato, Square Tiling, WOKWI Online Arduino Simulato, Online Arduino Simulato, Arduino Simulato

References

- **paint.net**

- Wokwi - Online Arduino and ESP32 Simulator
- Yours for the making - Instructables
- Andrei.erdei's Profile - Instructables
- Square Tiling in WOKWI - the Online Arduino Simulator : 4 Steps (with Pictures) - Instructables
- Square tiling - Wikipedia
- Wokwi - Online Arduino and ESP32 Simulator
- Clock Segment Fire - Wokwi Arduino and ESP32 Simulator
- Square Tiling Simulation - Wokwi Arduino and ESP32 Simulator
- Tetrakis Square Tiling With WS2812 LEDs : 5 Steps (with Pictures) - Instructables