Software Development with C#/921CSPTST1K13/2020S

Matthias Steinletzberger | K11771438

Truchet

A multi-scale Truchet tile pattern generator, based on a paper by Christopher Carlson.

Links:

christophercarlson.com MULTI-SCALE TRUCHET PATTERNS

Bridges 2018 paper: Multi-Scale Truchet Patterns

Code Overview

A brief overview over all namespaces, classes, structs, and enums.

```
namespace Truchet
    class Program
             contains the main function: argument error handling,
             instantiating Random and Tileset,
             generating the Image

    internal struct Parameters
             handles the arguments given via CLI,
             sets error code in case of bad arguments

namespace Truchet.Perlin
    class NoiseMap
             generates a 2D double array, filled with perlin noise
             exists because of perlin noise octaves,
             setting frequency and amplitude for the noise function

    class Perlin
             generates the actual perlin noise

    struct Vec2
             simple 2D vector implementation,
             written because I wanted to show overloading operators
```

```
namespace Truchet.Tiles
    abstract class Palette
            abstract class for the palettes available, holds 2 System.Drawing.Brush
            holds a static List<Palette> with all available palettes,
            which is initialized through a static constructor

        class SolidColorPalette : Palette
            inherits from Palette
            used for solid color tilesets

        [UNUSED] class LinearGradientPalette : Palette
            not available in v1.0,
            since Gradients didn't work very well with the concept

    abstract class Tile
            abstract class, holding X, Y,
            the subdivision Level,
            abstract function isContainer

        class ContainerTile : Tile
            holds four (smaller) Tile-type objects
            isContainer() returns true

        class GraphicTile : Tile
            holds a reference to a tileset image used for painting the final image
            isContainer() returns false

    class Tileset
            holds all different tiles for all subdivision levels
            also has an [UNUSED] lookup table for tile connections (not implemented)

    enum Direction
            simple enum for North, West, South, East
            uses binary flags
            not used because tile connections were not implemented

    enum TileType
            enum for hodling all different tiletypes
            numbered from 0 to 4, bitshift by 4 to make use of the Direction enum.
```

Process of Generating an Image

RNG

`System.Random` is used as the random number generator. An instance with a seed (can either be left blank or given as an argument) is distributed to all objects who need to generate random numbers.

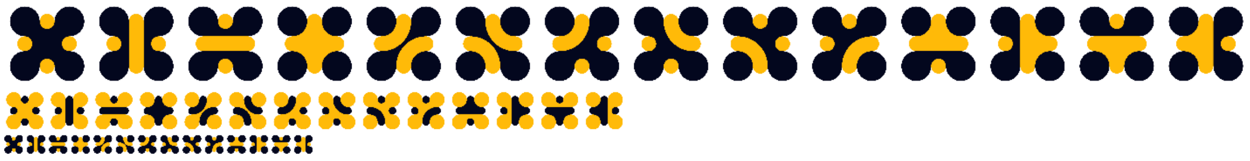
This way, a picture will always be generated the same way with the same seed, given that it has the same parameters. So, for example, the same image can be generated with multiple color palettes.

Tileset Generation

A `Tileset` is generated for each of the 14 different tiles. Tiles are only generated once and are reused for every time they are used.

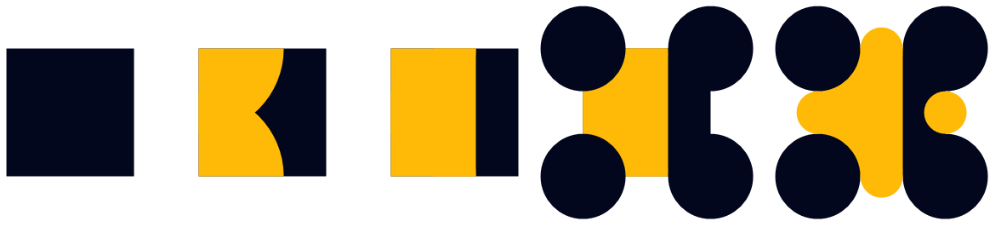
When I started out, I actually would only create the unique tiles once, then clone and rotate them, but this would create pixel-level imperfections, so I resorted to generating each tile separately.

The tiles are generated for each subdivision level, colors swapping at each new level.



Example of one tile (T West) being generated:

The first step and the last two steps are reused for every tile, the steps in between (can be none to 3 more steps) are different for each tile.



Noise Generation

The noise function is a Perlin noise function that I adapted for C#.

I wanted to use Perlin noise in my project because I was interested in understanding how it worked, and I felt that I would add a nice, organic touch to the generator. The result turned out great. I wrote a debug option that generates a picture to represent the values of the 2D noise array.

Generating the 2D Tile Array

A 2D array of `Tile`s is created with the dimensions specified through the CLI. If the pseudorandom method is used, there is a random chance that a `Tile` will become a `ContainerTile`.

If the Perlin noise is used, the decision whether or not a `Tile` becomes a `ContainerTile` that holds sub-tiles is made depending on the noise level at the tile position.

if a tile does not become a container tile, a random `ImageTile` is generated and subsequently assigned the `Image` from the `Tileset`.

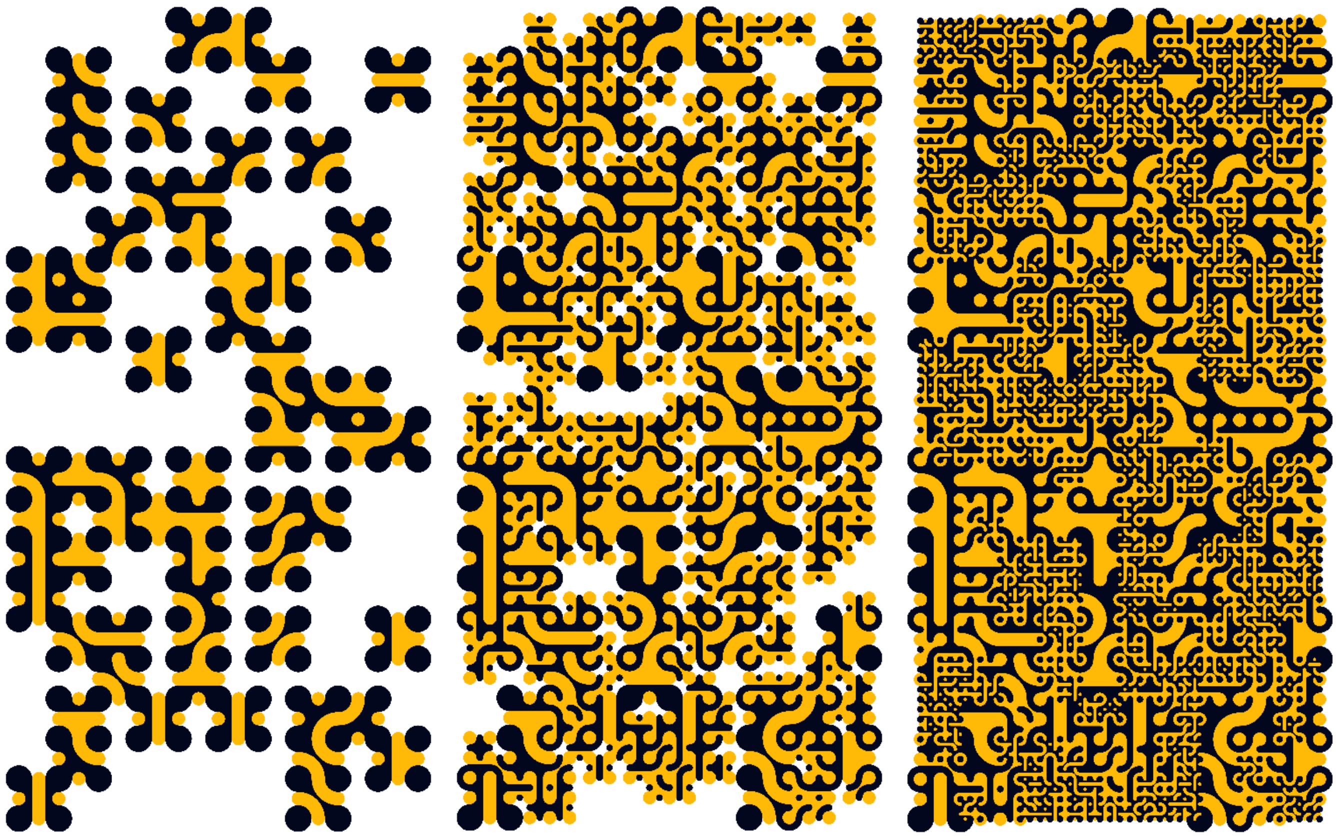
Generating the Image

All items in the 2D Tile Array are put into a queue. For each subdivision level, the queue is processed:

- Every `GraphicTile` in the queue is painted onto the final image
- Every `ContainerTile` has their sub-tiles put into a new queue

This process is then repeated for the new queue until all subdivision levels have been painted.

The reason why the image is generated level by level from biggest to smallest tile is that otherwise the bigger tiles would overlap the smaller ones, since tiles overlap when they are drawn.



Program Arguments

```
Standard functionality via CLI.
Syntax: truchet.exe [-h] [-d] [-r] [-p] [-b]
                    [--Palette id] [-l count] [-s seed]
                    [-rc count] [-cc count] [-ts size]

Options:
   -h               Displays this help screen.
   -d               Generates additional debug images. (default: off)
   -r               Sets generating method to random. (default: off)
   -p               Sets generating method to perlin noise.(default: on)
   -b               Turns on border cropping. (default: off)
   --Palette id     Specifies a palette. (default: Monochrome)
   -l count         Specifies the number of subdivision levels. (default: 3)
   -s seed          Specifies a seed. (default: random seed)
   -rc count        Specifies the amount of rows. (default: 10)
   -cc count        Specifies the amount of columns. (default: 10)
   -ts size         Specifies the tile size. (default: 300)

The following palettes are available:
   0: Monochrome
   1: Sapphire
   2: Imperial
   3: Deep
   4: Apricot
   5: Xiketic
   6: Canary
   7: Meadow
```

More Sample Images