Nightly Rgbit Squad

@isheldon with a bit of support from @ilyarsoftware

https://github.com/eternalflow/nightly-rgbit-squad.git

Minimal well structured smart contract system, which will **craft generative colorful 100% random squad** of digital creatures called “rgbits”.

Each **Nightly Rgbit Squad** is originated from 256 bits of random material which you need to pay for.

Thus, each family of rgbits is 100% born on-chain. A squad needs exactly 1 block to self-wrap in form of NFT.

Finally, the most important fact about **Rgbit Squad** is that they are 24/7 ready to be observed as SVG pics.

In no time, at no cost. For**ever**.

Tech Flow.

1. I compile and deploy the system with **compile.sh, test.sh**
2. I send **mint**, or others send **buy** message to **NftRoot** contract.
3. Anyone run **asSVG** get-method on **Data Contract** of their choice
4. Fun.

Summary

- 100% onchain-born and ever-living pixel art pieces
- Pretty fun abstract model (pic.1)
- Introducing on-chain SVG export get-method (pic.2)
- Took a single day to implement, run and render.

30 lines (23 sloc) | 773 Bytes Raw Blame

```
pragma ton-solidity ^0.43.0;


struct NightlyRgbitSquad {
    uint16 area;
    uint16 atomSize;
    uint16 hCount;
    uint16 vCount;
    uint32[] rgbits;
}

abstract contract Nightly {
    uint8 constant NIGHTLY_RGBATOM = 0x000055;

    uint16 constant ATOM_SIZE = 10;
    uint16 constant SIZE_H = 16;
    uint16 constant SIZE_V = 16;
    uint16 constant AREA = SIZE_H * SIZE_V;



    function art(uint256 material) public pure returns (NightlyRgbitSquad nft) {
        for (uint i = 0; i < 256; i++) {
            uint32 nightlyUnit = uint32((material >> i) ^ NIGHTLY_RGBATOM);
            bool existence = ((material >> i) & 1) > 0;
            uint32 rgbit = uint32(existence ? nightlyUnit : NIGHTLY_RGBATOM);
            nft.rgbits.push(rgbit);
        }
    }
}
```

libraries > NightlyArt.sol

```
pragma ton-solidity >= 0.43.0;

import "../abstract/Model.sol";

library NightlyArt {

    function renderBlock(
        uint32 rgbit, uint idx, uint16 atomSize, uint16 hCount, uint16 vCount
    ) public pure returns (string svgRect)
    {
        uint x = idx % hCount * atomSize;
        uint y = idx / vCount * atomSize;
        svgRect = format(
            "<rect x='{}' y='{}' fill='rgba({},{},{},{})' width='{}' height='{}' />",
            x, y,
            (rgbit >> 16) & 0xf, (rgbit >> 8) & 0xf, rgbit & 0xf, 1,
            atomSize, atomSize);
    }

    function render(NightlyRgbitSquad mdl) public pure returns (string svg) {
        string open = "<svg xmlns='http://www.w3.org/2000/svg'><g>";
        string rects;
        for (uint i = 0; i < mdl.area; i++) {
            rects.append(
                renderBlock(mdl.rgbits[i], i, mdl.atomSize, mdl.hCount, mdl.vCount)
            );
        }
        string close = "</g></svg>";
        svg = open + rects + close;
    }
}
```