ART AND ETHEREUM: DECENTRALIZED MEDIA DISTRIBUTION

Jeff Edmonds, B.S. in Computer Engineering 2022, in collaboration with Kristi Arnold, Ph.D.
Joseph T. Wunderlich, Ph.D.

ETOWN
ELIZABETHTOWN COLLEGE

OPPORTUNITY

Public blockchains, like Ethereum, offer securitized peer-to-peer distribution of data. By developing autonomous art production software and connecting it to the blockchain via Web3 and IPFS, this project explores a decentralized approach to distributing digital artworks.

GENERATIVE SOFTWARE

```
// global variables
let img;
let backg;
let backsel = backgs[Math.floor(Math.random() * backgs.length)];
let sel = [5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20];
let num = sel[Math.floor(Math.random() * sel.length)];;
let forms = new Array(num);

// load images
function preload() {
for (let i = 0; i < forms.length; i++)
   forms[i] = loadImage(random(imgs));
 backg = loadImage(backsel);
}
// print individual form
function printForm(
,  img,
,  x = random(0, windowWidth),
,  y = random(0, windowHeight),
,  w = random(10, windowWidth),
,  h = random(10, windowHeight),
,  angle = random(0, 360),
,  trans = random(5, 250))
 {
   push();
   translate(x,y);
   rotate(radians(angle));
   tint(255,trans);
   image(img,0,0,w,h);
   pop();
 }
// polar lemniscate
function polarLem() {
 push();
 translate(random(0, windowWidth), random(0, windowHeight));
 stroke(255);
 noFill();
 let i = 0;
 let w = random(10, windowWidth);
 let h = random(10, windowHeight);
 let angle = random(0, 360);
 let trans = random(5, 250);
 let spc = random(0.1, 1);
 let b = random(10,windowHeight/2);
 let len = random(0.1,TWO_PI);
 beginShape();
 for (let a = 0; a < len; a += spc) {
   let r = b*(1-cos(a)*sin(3*a));
   let x = r*cos(a);
   let y = r*sin(a);
   printPolar(img = forms[i], x, y, w, h, angle, trans);
 }
 endShape(CLOSE);
 pop();
```

Good to know: Ethereum is moving to proof-of-stake and offers L2 protocols. Both significantly decrease its environmental impact.

Figure 1: Selection of code from the generative software written in JavaScript with p5.js, including randomized image file selection, constrained random parameters, and one of many versions of a discrete polar curve

SMART CONTRACT

Smart contracts are deployed on the Ethereum blockchain. This contract offers basic functionality and creates a record of artwork data including an ID, the blockchain account that initiated the new artwork, and the date of creation. The contract also has functions to retrieve artwork data and the current artwork count.

```
pragma solidity ^0.5.0;

contract Art {
  uint public printCount = 0;

  struct Print {
    uint id;
    address acct;
    uint date;
  }

  // create data structure to map key/value
  mapping(uint => Print) public prints;

  // create new instance of print struct and increment print count
  function createPrint() public {
    printCount++;
    prints[printCount-1] = Print(printCount, msg.sender, now);
  }

  // retrieve print data
  function getPrint(uint idx) public view returns (uint id, address acct, uint date) {
    // copy the data into memory
    Print memory p = prints[idx];

    // break the struct's members out into a tuple
    return (p.id, p.acct, p.date);
  }

  // retrieve current print count
  function getPrintCount() public view returns (uint) {
    return printCount;
  }
```

Figure 2: Smart contract written in Solidity

```
const Art = artifacts.require("Art");

contract("Art", (accounts) => {
  let art0;

  before(async () => {
      art0 = await Art.deployed();
  });

  describe("creating a print and retrieving data", async () => {
    before("getting a print", async () => {
      await art0.createPrint();
    });
    it("can get print id and account", async () => {
      const print = await art0.getPrint(0);
      assert.equal(print[0], 1, "print id is correct");
      assert.equal(print[1], accounts[0], "print account is correct");
                                                          );
    });
  });

  describe("creating a print and returning print count", async () => {
    before("getting a print", async () => {
      await art0.createPrint();
    });
    it("can get the current print count", async () => {
      const count = await art0.getPrintCount();
      assert.equal(count, 2, "print count is correct");
    });
```

Figure 3: Integration testing written in JavaScript

Integration testing is done with the Truffle development framework using a local instance of a blockchain, secured from the main Ethereum network.

PROTOTYPE ARTWORK

The generative software uses a set of images and backgrounds created by the artist. Combining predetermined forms with constrained randomness creates a myriad of novel compositions.

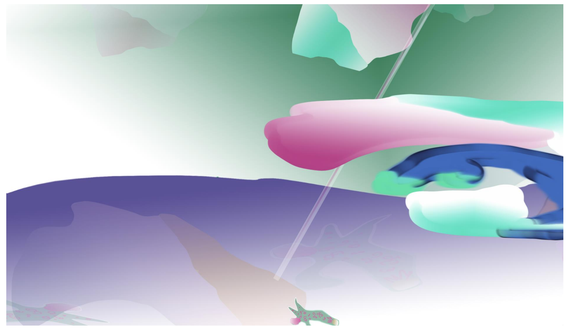

Figure 4: Prototype, 2022

Figure 5: Prototype, 2022

MINING HARDWARE

Cryptocurrencies are ambitious projects that seek alternatives to centrally controlled fiat currencies. A small mining rig was created for use in the project, rather than trade US dollars for ETH using a fiat on-ramp solution. ETH is used to generate transactions on the blockchain.

Figure 6: Xilinx Varium C1100 Blockchain Accelerator FPGA Card

IMPACT

Decentralized, peer-to-peer distribution can have an impact on many different types of markets, including financial services and cryptocurrencies, ticketing systems, and digital media such as video games, music, and art.

WEBSITE

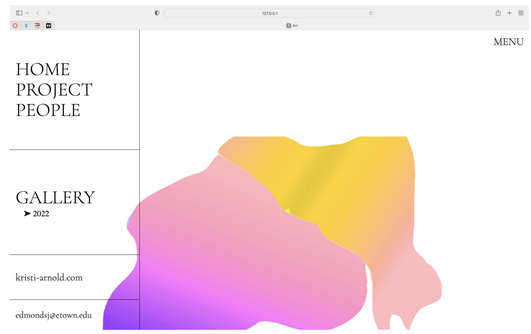

Figure 7: Main menu written in HTML, CSS, and JavaScript