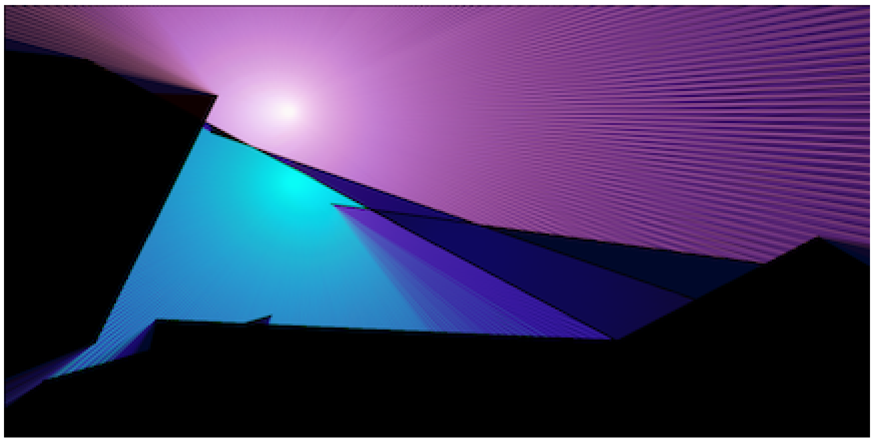
A portfolio for CODYRYANDESIGN P5JS examples

Sketch Gallery Project Resources Site Guide

Site Guide

Hello and thank you for visiting CODYRYANDESIGN’s website portfolio for showcasing interactive graphical sketches, written in P5JS, on the web! Below is a guide detailing the contents of this website and how to interact with it. Future expansions to this project will feature addition sketches, broader theme integration, and addition details on the vector-based physics system. For now, enjoy the site!



Introduction To This Site

A web-based portfolio of randomized generative art examples alongside exposed interface variables that provide the user with the means to experiment with and manipulate the state of each example. Through interacting with each piece the user will more-fully appreciate and understand the process of making computer-aided generative art, and enjoy creating something uniquely their own.

Though only an initial attempt, the goal of this site is to provide anyone with access to a modern internet browser the means to interact with, observe, and record their experiences with complex behavioral systems, such as flow fields, generative art, and vector mathematics, within a responsive graphical environment.

The graphical presentation and user input operations of this application will be updated and displayed in real-time to provide the user a greater sense of control and feedback from the program than one would attain from static images or diagrams describing these types of systems. In essence, this application is a real-time learning and testing playground for the user.

In addition to the challenge of incorporating interactive javascript content into a browser interface, this project was also my introduction into static-site generation and front-end toolsets which allow me to present this content in a clean and organized manner.

Site-Guide Content Listing

        - Jekyll
          A static-site generator.
            - Jekyll’s Tool-Set
              A full list of Jekyll’s tools.
            - Jekyll’s File Structure
              An explanation of Jekyll’s file organization.
    - Processing’s P5JS
      An explanation of a typical P5JS project’s file structure.
        - p5.js
        - p5.dom
        - quicksettings.js
        - p5.gui
        - Sketch Files:
            - sketch.js
            - particle.js
            - rocket.js
- Sketch Gallery: Home Page For All Sketches
    - Sketch Layout
      A description of a typical sketch page on this site.
    - Interactivity Options
        - GUI Controls
          A Description of GUI Interaction
        - The Canvas Window
          A DOM element that is embedded into the HTML code and renders the scene at 60 frames per-second.
            - Mouse Interaction With The Canvas
              A Description of Mouse Interaction
            - Reloading Randomized Conditions
              Instructions on resetting the canvas.
            - Capturing The Canvas
              Instructions on how to download snapshots of canvas renderings.
        - Sketch Description: A description of the general behavior of the sketch.
        - Sketch Manipulation Suggestions: A few examples of how the sketch might be interacted with.

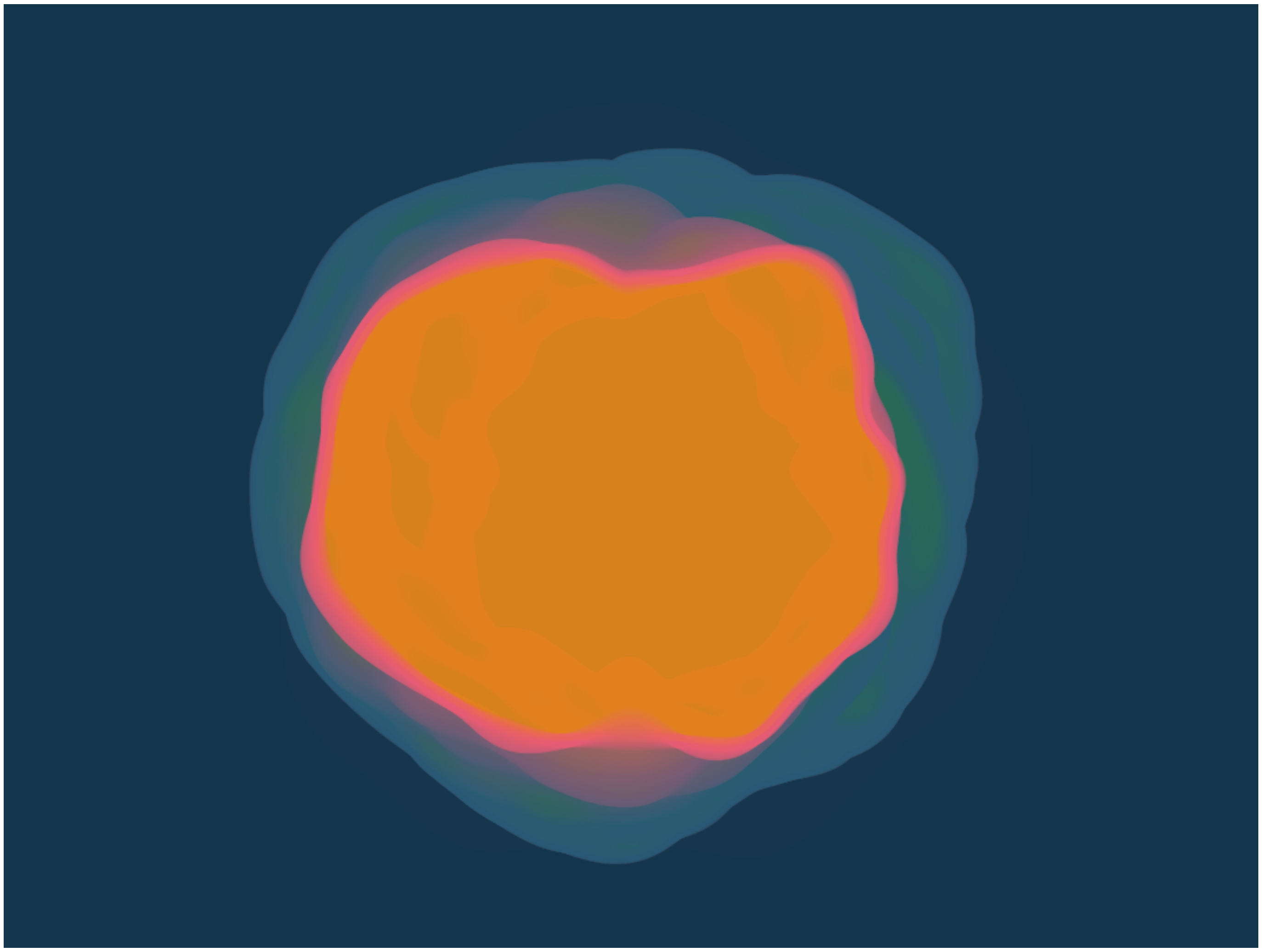
    - GUI Variable Descriptions: A listing of interactable sketch variables accessed from within the GUI panel.
    - Mouse Interaction Descriptions: A listing of interactive input through the mouse.
- Sketches

  A listing of available sketches.

- Back End Services

  A brief listing of minimal back-end resources required for this project.

    - Amazon Web Services: Amazon web services are used to host this site utilizing:
        - S3 Cloud Storage
        - Linux Environment
        - Apache Web server
    - Version Control
        - GitHub
        - GitHub Pages



System Overview:

This project is a full-stack development, utilizing a cloud-hosted linux environment on AWS S3, an Apache web server, version-control through GitHub, a static-site generator, Jekyll, and sketch renderings built in The Processing Foundation's P5JS with additional features added through community library extensions.

This project utilizes a number of libraries and toolsets across multiple languages. A complete list of resources used for this project can be found here.

- ## Static Site Generation:

  - ### What Is A Static Site?

    A static site is one or more HTML documents stored on a server and delivered to the User when they visit the website. Static site documents are presented to the User in a mostly unaltered format from how they are stored within the server. Because static-site files do not need to be created or changed on the fly, static-sites often load faster and react more quickly to User navigation needs.

  - ### What Is A Dynamic Site?

    A dynamic site is, in many ways, the opposite of a static site. Instead of delivering content to the User as it is stored on the server, content in generated, based on User needs and interactions, by a server-side scripting language (like PHP) in real-time.

  - ### Static v Dynamic Sites

    For more information on how static and dynamic sites differ, check out this handy presentation by Nilclass.com outlining the key differences between the two implementations.

    - Take-aways:
      - Static Sites Are:
        - Flexible
        - Dependable
        - Cheap
        - Very efficient
        - Incredibly fast
        - Best-fit for content that rarely changes over time
        - Easily transferrable between hosting services
        - More Secure
      - Dynamic Sites Are:

        - Pages are created through run-time server-side scripting
        - Can utilize popular and familiar web-tools, like Wordpress.
        - Slower to load and interact with, compared to static sites

- ## What Is A Static Site Generator?

  One of the biggest complaints when it comes to managing a static site is page-management and updates. In earlier implementations of static sites, before dynamic sites were around, there was constant maintenance and updating of static website pages when changes were required. If you wanted to make the same change to every page on your site, you would have to manually edit the HTML of each page.
  In order to address this issue, many tools have been created that allow for many static site pages to be generated quickly and easily through a templating process. The output from a static-site generator is ultimately just static HTML files and assets, and can easily be transferred to a host service. For this website, I decided to use a Ruby-based SSG called Jekyll to quickly build site assets from a minimal code-base.

  - ### Jekyll

    - Jekyll's Toolset:
      - HTML5
        The most-recent implementation of the Hyper-text Markup Language standard for communicating between computers on the internet.
      - CSS
        A declarative styling language for specifying how HTML content should be rendered on the page.
      - SCSS
        Sometimes referred to as SASS, SCSS extends the capabilities of CSS.
      - Liquid
        A Ruby-native templating language created by Shopify and used as part of Jekyll's theming tool-set.
      - Markdown
        A lightweight HTML markup language that uses plain text conversion to make the process of writing for the web easier. The entirety of this website overview is written in Markdown and converted to HTML on build.
      - kramdown
        A Markdown converter written in Ruby.
      - FrontMatter

A meta-data parser for page-specific variables that stores data in a YAML or JSON format. Jekyll uses Frontmatter to detect what content for a page should be displayed, how pages might be ordered in a list, and how the layout of a page should be formatted, among other potential uses.

- Rogue
  A quick and simple sytax-highlighting tool for embedded code snippets displayed on web pages.

  It can take something like this:

```
<div id="sketch-holder">
  <script type="text/javascript" src="sketch/sketch.js"></
</div>
```

  and transform it into this:

```
  <div id="sketch-holder">
    <script type="text/javascript" src="sketch/sketch.js">
  </div>
```

- minima Theme
  The base theme assets for building Jekyll's default site. This site utilizes an extension of the minima theme by Benjamin Habert to include support for P5JS sketches.

- Jekyll's File Structure:
  - ## _config.yml

    Settings and configuration preferences for site-wide setup.

  - ## _layouts

    This folder contains a template for specifying the layout of the gallery page.

  - ## _includes

    This folder contains additional layout snippets used for repeating implementations of page headers, footers, gallery image positioning, and included dependencies.

  - ## _data

    This folder contains the list of dependencies any sketch can utilize as needed. This is especially useful for multiple sketches that utilize the same dependencies.

  - ## _projects

    Each sub-folder within the projects folder contains a P5JS sketch, it's associated .js helper files, and a Markdown document that builds to HTML. See the P5JS Project File

Structure section for more details.

- ## Site Pages

  The individual sections of the site.

  - ### gallery.md:

    The “home” page of the site. This is populated with gallery links to available sketches on build.

  - ### introduction.md:

    A page dedicated to introducing and explaining the functionalities and goals of this project. You are currently viewing this page.

  - ### resources.md:

    A page listing all relevant resources, references, and inspiration utilized in the creation of this website project.

- ## _site

  When Jekyll builds the site, this folder is populated with all the static files needed to run the site. Here is the general file structure of this folder

  - _site\assets
    This folder contains CSS styling information, as well as default alternatives to page resources.

  - _site\introduction
    This folder contains the HTML for the “introduction” page.

  - _site\resources
    This folder contains the HTML for the “resources” page.

  - _site\index.html
    The HTML for the “home” page.

- ## assets

  This folder contains default assets and the SCSS styling for the minima theme.

- ## Gemfile

  A Ruby file containing a list of gems needed for the project to execute properly.

- ## gemfile.lock

  A container for housing all gems specified in the Gemfile.

- ## Processing’s P5JS:

  - ### P5JS Main Libraries and Extensions

    - #### p5.js

      A Javascript library, based off the original Java library Processing, made for artists, designers and creative coders with a robust toolset for drawing in 2D and 3D on the web.

    - #### p5.dom

      A library extension for P5JS that allows for easy DOM element manipulation through the browser.

    - #### quicksettings.js

      A JavaScript library designed to make GUI panel creation fast and easy.

    - #### p5.gui

      A P5JS-compatible library extension of quicksettings that allows for quick, clean GUI panel generation in a P5JS sketch.

    - #### Sketch Files

      The main file for executing sketch processes is sketch.js. This is a custom file unique to every sketch. Below is an example of a simple sketch.js file, along with it’s helper classes: particle.js, and rocket.js. This example borrows from the Fireworks! Sketch

      - ##### sketch.js

```
//Fireworks!

//Create a gloabl fireworks var
//for holding all active fireworks
var fireworks = [];

//Declare global GUI vars
//These vars can be referenced
//anywhere in the sketch
var backgroundC = '#090017';
var backgroundA = 0.06;
var backgroundAMin = 0;
var backgroundAMax = 1;
var backgroundAStep = .01;

var rocketSpawnChance = .05;
var rocketSpawnChanceMin = .01;
var rocketSpawnChanceMax = 1;
var rocketSpawnChanceStep = .01;

var rocketSize = 1;
var rocketSizeMin = 0.1;
var rocketSizeMax = 5;
var rocketSizeStep = .01;

var particleSize = 3;
var particleSizeMin = 0;
var particleSizeMax = 20;
var particleSizeStep = .1;

var explosionSize = 30;
var explosionSizeMin = 1;
var explosionSizeMax = 100;
var explosionSizeStep = 1;

var particleDecay = .013;
var particleDecayMin = 0.001;
var particleDecayMax = .03;
var particleDecayStep = .001;

var gravityAmount = 0.19;
var gravityAmountMin = 0;
var gravityAmountMax = .8;
var gravityAmountStep = .01;

var windAmount = 0;
var windAmountMin = -.1;
var windAmountMax = .1;
var windAmountStep = .001;

var particleVelocity = 0.85;
var particleVelocityMin = 0;
var particleVelocityMax = 2;
var particleVelocityStep = .01;

//Create the gui panel
var gui;
//and toggle it's visibility
var visible = true;
```

```
//The setup function is only run once
function setup() {
  //Create a canvas object whose
  //dimensions are based off the browser
  //window size
  canvas = createCanvas(window.innerWidth/2, window.innerHeig
  canvas.parent('sketch-holder');
  // colorMode(HSB, 360, 100, 100, 1)
  //Draw all colors using
  //Hue, Saturation, Brightness, and Alpha
  colorMode(HSB);

  //Title the GUI panel
  gui = createGui('HSV GUI');
  //and add in all global GUI vars
  gui.addGlobals(
    'backgroundC',
    'backgroundA',
    'rocketSpawnChance',
    'rocketSize',
    'particleSize',
    'explosionSize',
    'particleDecay',
    'gravityAmount',
    'windAmount',
    'particleVelocity');
  //Initially hide the GUI from view
  gui.hide();
}

//The draw loop attempts to execute
//60 frames every second
function draw() {
  //Create a gravitational force
  //vector based off the gravityAmount
  gravity = createVector(0, gravityAmount);
  //Create a gravitational force
  //vector based off the windAmount
  wind = createVector(windAmount, 0);
  //Take in background color-picker
  //value as a color
  bColor = color(backgroundC)
  //Render the background according to the
  //hue, saturation, brightness of bColor
  //Also assign alpha based off backgroundA
  background(hue(bColor), saturation(bColor), brightness(bCol
  //Change probability of a new rocket being
  //spawned based off rocketChance value
  if(random(1) < rocketSpawnChance) {
    //Push the new rocket object into the
    //rockets array
    fireworks.push(new Firework());
  }
  //Working backwards through the rockets array
  for(var i = fireworks.length - 1; i >= 0; i--) {
    //update every rocket's properties
    fireworks[i].update();
    //and show it on the screen
    fireworks[i].show();
    //When a rocket has finished activities
    if(fireworks[i].done)
      //Remove it from the rockets array
      fireworks.splice(i, 1);
  }
  //Listen for user key presses
  detectKeyPress();

}

//Handle user key presses
function detectKeyPress() {
  //If the user presses the 'p' key
  if(key == 'p') {
    //Take a snapshot of the canvas
    save('fireworks.png');
    //nullify key value to prevent multiple downloads on subs
    key = null;
  }
    //If the user presses the 's' key
  if(key == 's') {
    //Toggle the visibility of GUI panel(s)
    if(visible) {
      gui.show();
    }
    else {
      gui.hide();
    }
    visible = !visible;
    //nullify key value to prevent multiple downloads on subs
    key = null;
  }
}
```

- particle.js

```
//A simple class for creating vector-based objects
//that can be influenced by other vector forces.
function Particle(x,y, velocity) {
//Initialize the particle object with
//an x,y position
  this.pos = createVector(x,y);
  //and optional velocity
  this.vel = velocity || createVector(0,0);
  //Also give the object an acceleration vector
  this.acc = createVector(0,0);
  //Keep track of how long the particle has existed for
  this.lifespan;
  //Keeps track of the size of the particle object
  this.size;

//A function for handling forces that act on the object
  this.applyForce = function(force) {
    //Pass in a vector force and
    //add it to the particles acceleration
    this.acc.add(force);
  }

//Update particle object properties during every loop
  this.update = function() {
    //Add the acceleration to the velocity
    this.vel.add(this.acc);
    //Add the velocity to the object's position
    this.pos.add(this.vel);
    //Reset acceleration information for the next frame
    this.acc.mult(0);
  }

}
```

- rocket.js

```
function Firework() {
  //Inherit Particle class properties
  //and initialize the particle object
  //at a randomized position below the canvas
  this.firework = new Particle(random(width), height, createV
  //If the mouse is interacting with the sketch
  //determine the rocket's color by the mouse's
  //y coordinate
  this.color = mouseIsPressed ? map(mouseY, 0, height, 0, 360
  //Keep track of whether the rocket has
  //exploded or not
  this.exploded = false;
  //The rocket is done when its children
  //particles have all been removed from
  //the scene
  this.done = false;
  //Create an empty particle array for holding
  //explosion particles
  this.particles = [];

  //Update firework properties every frame,
  //keeping track of applied vector forces,
  //and explosion conditions
  this.update = function() {
    if(!this.exploded) {
      //Only apply gravity to the rocket if
      //it hasn't yet exploded
      this.firework.applyForce(gravity);
      //Update the rocket particle's
      //position based on active forces
      this.firework.update();
      //When the rocket's velocity matches
      //the gravitational force acting on it
      if(this.firework.vel.y >= 0)
        //EXPLODE!
        this.explode();
    }
    //For every child particle created
    //when the rocket explodes
    for(var i = this.particles.length-1; i >= 0 ; i--) {
      //Apply gravity to the child-particle
      this.particles[i].applyForce(gravity);
      //Apply wind to the child-particle
      this.particles[i].applyForce(wind);
```

```
        //Update child-particle properties
        this.particles[i].update();
        //If the child-particle's lifespan
        //runs out
        if(this.particles[i].lifespan <= 0) {
          //Remove that child particle from the array
          this.particles.splice(i, 1);
        }
        else {
          //Otherwise, decrement the child-particle's
          //lifespan by a random amount of the particleDecay
          //value
          this.particles[i].lifespan -= random(particleDecay);
        }
      }
      //When the rocket has exploded and all
      //child-particles have died
      if(this.exploded && this.particles.length == 0)
        //The rocket can be removed from the scene
        this.done = true;
    }

    //A function for generating a random amount
    //of child-particles when the rocket explodes
    this.explode = function() {
      //Determine how many child-particles
      //the rocket will have
      var numParticles = random(50, 300);
      //For every child-particle
      for(let i = 0; i < numParticles; i++) {
        //Assign the child-particle a random
        //position based off the location of
        //the rocket's explosion
        var p = new Particle(this.firework.pos.x, this.firework
        //Give the child-particle a lifespan of 1
        p.lifespan = 1;
        //Randomly assign the size of the child-particle
        p.size = random(particleSize);
        //Calculate the velocity of a child-particle
        //based off the explosionSize value
        p.vel.mult(random(explosionSize));
        //Push the fully-built child-particle to
        //the rocket's array
        this.particles.push(p);
      }
      //The rocket has now exploded
      this.exploded = true;
    }

    //Display the rocket particle
    //as it soars into the sky
    this.show = function() {
      if(!this.exploded) {
        //As long as it hasn't yet exploded
        stroke(this.color, 100, 100, 1);
        strokeWeight(rocketSize * this.firework.vel.mag()/2);
        //Render the rocket as a point at it's
        //current position
        point(this.firework.pos.x, this.firework.pos.y);
      }
      else
        //Otherwise, draw the child-particles instead
        for(var i = 0; i < this.particles.length; i++) {
          //Add an additional fall amount to particles after th
          this.particles[i].vel.mult(particleVelocity);
          //Color the particles based off the initial firework
          this.particles[i].color = this.firework.color + rando
          stroke(this.color + random(-20, 20), 100, 100, this.p
          strokeWeight(this.particles[i].size*this.particles[i]
          //Render the child-particle as a point
          point(this.particles[i].pos.x, this.particles[i].pos.
        }
    }

}
```

- ## Sketch Gallery

  - ### Sketch Layout

    Each sketch page on this site follows a similar layout as described below.

    - **Sketch Title**
      The title of the sketch.

    - **Sketch Description**
      A brief introduction to the general behaviors of the sketch.

- ## Interactivity Options

  This section guides the viewer through a sketches interactivity options.

  - ### GUI Controls

    Manipulating the sketch through GUI variable manipulation.

    - #### Toggling GUI Visibility

      GUI panel visibility can be toggled by pressing the `s` key

    - #### Moving GUI Panels Within The Window

      GUI panels can be moved around the window by dragging the title bar with the mouse.

    - #### Collapsing GUI Panels

      GUI panels can be collapsed into a discrete form by double-clicking the title of the panel.

    - #### Manipulating Sketch Variables

      There are a variety of ways the user can interact with a particular sketch. Often, the user will have access to setting variable values within a slider range, enabling or disabling sketch conditions, or changes color preferences within a color-picker interface.

- ## The Canvas Window

  The canvas window sits in the center of the page and renders (ideally) at 60 frames per second. It is the only part of the sketch page that is interacted with.

  - ### Mouse Interactions With The Canvas

    Some sketches offer the option to influence particle behavior by clicking or dragging the mouse within the canvas window. If the mouse is outside the canvas window, it will perform normally for changing GUI panel values and navigating the site.

  - ### Reloading Randomized Conditions

    Many of the sketch environments are randomly determined when the sketch is loaded. If you would like to randomize the sketch environment, simply reload the page at any time by pressing `CMD+R`.

  - ### Capturing The Canvas

    Still image screenshots of the entire canvas window can be taken at any time. Simply press `p` and a screenshot of the canvas window will be downloaded to your computer.

- Sketch Manipulation Suggestions:

  Sketch-specific suggestions on how to manipulate GUI variables and interact with the canvas. These suggestions merely guide the viewer towards particular experiences with the sketch, and are only a fraction of possible sketch alterations.

- GUI Variable Descriptions

  A listing of the available GUI controls and brief descriptions of each variables effect on the sketch.

- Sketches

  A listing of available sketches. (Identical listing to the content-links in the Sketch Gallery.)

  - 2D Ray-Casting
    A light and shadow simulation utilizing a 2D ray-casting technique.
  - Flowfield
    Swarms of particles are propelled across a field of vector forces.
  - Fireworks!
    A simulation of fireworks shooting into the sky under the effects of gravitational and wind forces.
  - Perlin Noise Blob
    An oozing blob is drawn using Perlin noise coordinates.
  - Lissajous Curve Table
    A visualization of harmonic-resonance.
  - With more to come!

- Back End Services

  This project made use of a number of cloud services including: content hosting on Amazon Web Service's S3 cloud storage, working within a Linux environment, running an Apache web server, and utilizing GitHub for basic version-control.

  - Amazon Web Services

    - S3 Cloud Storage

      Inexpensive cloud hosting of static content files.

    - Linux Environment

      Open-source and widely supported as the predominant server-side OS platform.

    - Apache Web Server

An open-source HTTP web server.

- Version Control
  - GitHub

    Basic version control.

  - GitHub Pages (coming soon)

    Implementing GitHub Pages will allow for seamless hosting of project files and generation of a live website on a single platform.

return to top of page
view sketch gallery

A portfolio for CODYRYANDESIGN P5JS examples

A portfolio for CODYRYANDESIGN P5JS examples
codyryandesign@gmail.com

codyryandrake
codyryandesign

A website for deploying portfolio examples of P5JS (Processing) sketches on the web!