Canvas Color Mixer Programmer’s Guide

Team Darkest Error

Members

Dakota Stephens <djsh2z@umsl.edu>

Mindy Zheng <mzhfc@umsl.edu>

Nathan Pimentel <nathanjpimentel@umsl.edu>

Anthony Pardo <ajpcnc@umsl.edu>

Frankie Mccaa <fm6np@umsl.edu>

Thomas Citrowske <tjcnc2@umsl.edu>

Table of Contents

[Assumptions about the Programmer 3](#_Toc153727569)

[High Level Design 3](#_Toc153727570)

[More Detailed Design 4](#_Toc153727571)

[Installation Instructions 13](#_Toc153727572)

[Appendix A. Implementation Code 14](#_Toc153727573)

[Index.html: 14](#_Toc153727574)

[Usage.html: 29](#_Toc153727575)

[Run.html: 35](#_Toc153727576)

[Myjs.js: 51](#_Toc153727577)

[Home.js: 84](#_Toc153727578)

[Styles.css: 110](#_Toc153727579)

[Run.css: 110](#_Toc153727580)

[Appendix B. User Manual 111](#_Toc153727581)

[Appendix C. Test Plan 112](#_Toc153727582)

[Introduction 113](#_Toc153727583)

[Testing Plan 114](#_Toc153727584)

# Assumptions about the Programmer

To set up the system we are assuming that the administrator will be able to have and configure a web hosting platform for external access to a folder. In which, the files for this webpage will need to be placed. Additionally, we assume the admin has a basic understanding of how to configure the default landing page for the web host of choice.

We assume that the admin has access to download the webpage files and that the admin is able to upload and save the files into the directory described above.

We assume that the administrator has a basic understanding of how a website works in the context of HTML and JavaScript. The administrator must also a basic understanding of using GitHub to retrieve the webpage files from the repository: https://github.com/TheBuilderHero/TheBuilderHero.github.io.git.

That said, we assume that the administrator is familiar with:

* JavaScript
* HTML
* Web hosting
* GitHub

Note, hosting this is not something which these files handle. The hosting must be set up by the administrator.

# High Level Design

Here is the high-level design for the flow of information in and out of our website along with some backend considerations. All the HTML files use bootstrap JavaScript and CSS.

The HTML file index.html uses the JavaScript file home.js, the image file img.png, and the CSS file styles.css.

The HTML file usage.html uses an image file called using\_site.gif. The HTML file usage.html has no JavaScript has no custom JavaScript file needs.

The HTML file run.html uses the JavaScript file myjs.js and the CSS file run.css.

A diagram of a website

Description automatically generated

# More Detailed Design

For a more detailed understanding of how the program operates, we have included some pseudo code which describes the operation of the website from a higher-level standpoint:

# Initialize variables

checkAll = [0, 0, 0, 0] # Acts as a "key" for X-dim, Y-dim, repetitions, and x-values

optionsArray = [] # Declare an empty array to store options

# Function to handle music change

function change\_music(e):

e.preventDefault()

elm = e.target

audio = document.getElementById('backGroundAudio')

source = document.getElementById('audio')

source.src = elm.getAttribute('data-value')

audio.load() # Preload the audio without playing

audio.play() # Play the song right away

audio.stop()

# Event listener on window load to gather options from user input

window.addEventListener("load", function(event):

optionsArray.push(document.getElementById("option1").value)

optionsArray.push(document.getElementById("option2").value)

optionsArray.push(document.getElementById("option3").value)

optionsArray.push(document.getElementById("option4").value)

optionsArray.push(document.getElementById("option5").value)

optionsArray.push(document.getElementById("option6").value)

optionsArray.push(document.getElementById("option7").value)

optionsArray.push(document.getElementById("option8").value)

optionsArray.push(document.getElementById("option9").value)

# Function to confirm user input

function confirmInput():

# Implementation for confirming input

# Function to store old value on focus and change

function storeOldValue(ele):

document.previousColorSelected = ele.value

# Initialize global variables

globalListOfItems = [] # Ascending list of x-values

indValue = ""

# Function to add item to the list

function addItemToList():

# Implementation for adding an item to the list

# Function to remove item from the list

function removeItemFromList():

# Implementation for removing an item from the list

# Function to update other lists based on the selected element

function updateOtherLists(ele):

# Implementation for updating other lists based on the selected element

# Function to show inputs based on the selected independent variable

function show\_inputs\_dependent(ele):

# Implementation for showing inputs based on the selected independent variable

# Function to remove selected item from a select box

function removeSelected(selectBox, item):

# Implementation for removing selected item from a select box

# Function to remove all items excluding the selected ones from a select box

function removeAllExcludingSelected(selectBox):

# Implementation for removing all items excluding the selected ones from a select box

# Function to check if two arrays are equal

function arraysEqual(a, b):

# Implementation for checking if two arrays are equal

# Function to show a message

function showMessage(id):

id.classList.remove("hide")

id.classList.add("show")

# Function to hide a message

function hideMessage(id):

id.classList.remove("show")

id.classList.add("hide")

# Function to handle change in input values

function change():

# Implementation for handling change in input values

# Function to register event listeners

function register():

# Implementation for registering event listeners

# Function to check dimensions for Y\_X

function checkDimXY():

# Implementation for checking dimensions for Y\_X

# Function to check dimension Y

function checkDimY():

# Implementation for checking dimension Y

# Function to check repetitions

function checkRep():

# Implementation for checking repetitions

# Function to check the form before submission

function submitCheck():

# Implementation for checking the form before submission

# Function to set dimension Y\_X based on dimension Y

function setDimY\_X():

# Implementation for setting dimension Y\_X based on dimension Y

# Function to get a random integer

function getRandomInt(max):

# Implementation for getting a random integer

# Function for three-color picker

function three\_color\_picker(colorItems):

# Implementation for three-color picker

# Function to clear select list

function clearSelectList():

# Implementation for clearing select list

# Function to randomize inputs

function randomize\_inputs():

# Implementation for randomizing inputs

# HTML Document Structure

Start HTML document

Start HTML head

Include title tag with text "Painting Grid"

Include meta tags for character set and viewport

Include Bootstrap CSS and JavaScript libraries

Include local JavaScript and CSS files

End HTML head

Start HTML body with onload event "register()"

Start navigation bar with Bootstrap styling

Include brand logo and navigation links

End navigation bar

Start container with Bootstrap styling for the main content

Start row with content and sidebar

Start sidebar with music choices

Display music links

End sidebar

Start main content column

Start form for the color experiment with action "./run.html" and method "get" and onsubmit event "submitCheck()"

Include hidden input for global list of items

Display welcome message and quick run buttons

Display dependent and independent variables form

Display color information, stopping criteria, and song choice forms

Display submit and restart buttons

End form

Start grid container

End grid container

End main content column

Start sidebar with additional music choices

Display additional music links

End sidebar

End row

End container

Start footer with copyright information

End HTML body

End HTML document

1. Initialize HTML document with language set to "en".

2. Create head section:

a. Set document title to "Painting Grid".

b. Specify character set as UTF-8.

c. Set viewport for responsive design.

d. Include Bootstrap CSS from CDN.

e. Include jQuery and Bootstrap JS from CDN.

f. Include custom JavaScript from "./myjs.js".

g. Include another version of jQuery from CDN (version 3.7.1).

h. Define internal styles for the document.

3. Create body section:

a. Create a navigation bar using Bootstrap.

i. Include a collapsible button for small screens.

ii. Add a logo in the navigation bar.

iii. Include navigation links for "Home" and "Usage".

b. Create a fluid container for the main content.

i. Create a row with two columns: one empty column and one column for the main content.

ii. Inside the content column:

- Display a heading "Basic usage".

- Provide a brief description of the application.

- Display an image ("using\_site.gif").

- Provide information about the main independent value and its requirements.

- Explain how to input other information and personalize the experiment.

- Add a link to start the experiment.

- Display a grid container (empty in the pseudocode).

c. Create a fixed footer with copyright information.

i. Include a copyright notice with the current year and the website owner.

ii. (Optional) Include a link to a painting music playlist.

4. Close the HTML document.

1. Initialize HTML document with language set to "en".

2. Create head section:

a. Set document title to "Painting Grid".

b. Specify character set as UTF-8.

c. Set viewport for responsive design.

d. Include Bootstrap CSS, jQuery, Bootstrap JS, custom JavaScript, and another version of jQuery from CDNs.

e. Include YouTube iframe API for audio.

f. Include color mixing scripts from CDNs.

g. Define internal styles for the document.

3. Create body section:

a. Create a navigation bar using Bootstrap.

i. Include a collapsible button for small screens.

ii. Add a logo in the navigation bar.

iii. Include navigation links for "Home," "Usage," and "Experiment."

b. Create a fluid container for the main content.

i. Create a row with one column for the experiment details.

- Display a heading "Running Experiment!".

- Provide a warning not to reload the page.

- Display experiment controls (speed slider, stop/start buttons, extreme speed button).

ii. Create another row with three columns for result selection, final canvas, and calculated values.

- Display a canvas for the experiment.

- Provide options for terminating conditions, speed control, and buttons for start, stop, and extreme speed.

- Display checkboxes for selecting experiment values to be calculated.

- Show buttons for continuing, displaying graphs/tables, and completing the experiment.

c. Create a container for the result selection and final canvas.

- Display the experiment result selection, final canvas, and calculated values.

d. Create a container for the graph and table.

- Display a graph using CanvasJS and a table with the selected experiment values.

e. Create a container for additional buttons.

- Display buttons for making a new table/graph, abandoning the experiment, and quitting the program.

4. Create a fixed footer with copyright information and an optional YouTube audio player.

5. Close the HTML document.

# Installation Instructions

Before you can install and/or set up the website please make sure you have acquired a host from which you will be hosting your site externally. Note, this was originally designed to be hosted from GitHub Pages. Due to a restriction which GitHub Pages has we also setup the website to use the GET method for page submissions (that might be important for choice of host).

Steps to setup and/or install:

1. Once you have the host, you will need to download all the files in the docs folder from the repository: <https://github.com/TheBuilderHero/TheBuilderHero.github.io.git>
2. Place all files into the new web hosted directory (All files should be taken out of the docs folder and moved directly into the hosted directory).
3. Verify the following files exist in the directory: “index.html”, “run.html”, “usage.html”, “home.js”, “myjs.js”, “img.png”, “using\_site.gif”, “run.css”, and “styles.css”.
4. From there, make sure that the host default landing page is index.html (which is the home page).
5. Save these changes and navigate to the host’s provided URL to test.
6. If you are placed onto the home page, then you have successfully installed and set up the website. If not, then please reattempt the above instructions.

Please refer to the User Manual for a walkthrough of the operation of the website. Additionally, you can visit the usage tab directly on the website to also get information on how to operate the website.

# Appendix A. Implementation Code

## Index.html:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Painting Grid</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

<!--<script src="./check.js"></script>-->

<script src="./home.js"></script>

<link href="./styles.css" rel="stylesheet" type="text/css">

<style>

/\* Remove the navbar's default margin-bottom and rounded borders \*/

.navbar {

margin-bottom: 0;

border-radius: 0;

}

/\* Set height of the grid so .sidenav can be 100% (adjust as needed) \*/

.row.content {height: 450px}

/\* Set gray background color and 100% height \*/

.sidenav {

padding-top: 20px;

background-color: #f1f1f1;

height: 100%;

}

/\* Set black background color, white text and some padding \*/

footer {

background-color: #555;

color: white;

padding: 15px;

position: fixed;

left: 0;

bottom: 0;

width: 100%;

}

/\* On small screens, set height to 'auto' for sidenav and grid \*/

@media screen and (max-width: 767px) {

.sidenav {

height: auto;

padding: 15px;

}

.row.content {height:auto;}

}

</style>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto auto auto auto auto auto auto;

background-color: unset;

padding: 10px;

}

.grid-item {

display:grid;

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

grid-template-columns: repeat(auto-fit, minmax(1px, 1fr));

grid-template-rows: repeat(auto-fit, minmax(1px, 1fr));

}

.container {

display: grid;

grid-gap: 10px;

grid-template-columns: repeat(auto-fill, minmax(min(200px,100%), 1fr));

}

.child {

display: flex;

align-items: center;

justify-content: center;

padding: 5px;

border: 3px solid #a07;

}

.formItems{

text-align: center;

color: white;

}

.sidenav{

height: 190%;

}

#var-legend {

color: white;

}

body {

margin:0;

color: white;

background-color: #023f3f;

}

</style>

</head>

<body onload="register();">

<nav class="navbar navbar-inverse">

<div class="container-fluid">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar" style="color: white; background-color: blue;">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#"><img src="img.png" width="50px" height="50px" style="margin-top: -15px !important; /\* 50% of your logo width \*/"></a>

</div>

<div class="collapse navbar-collapse" id="myNavbar">

<ul class="nav navbar-nav">

<li class="active"><a href="./index.html">Home</a></li>

<li><a href="./usage.html">Usage</a></li>

</ul>

<!-- Remove Login feature

<ul class="nav navbar-nav navbar-right">

<li><a href="#"><span class="glyphicon glyphicon-log-in"></span> Login</a></li>

</ul>

-->

</div>

</div>

</nav>

<div class="container-fluid text-center">

<div class="row content">

<div class="col-sm-2 sidenav" style="margin-bottom: 0px; background-color: unset;">

<p>Music Choices:</p>

<p><a href="https://www.youtube.com/watch?v=sRkzqLudJPE" target=”\_blank”>Larry Owens - The Joy Of Painting</a></p>

<p><a href="https://www.youtube.com/watch?v=oy2zDJPIgwc" target=”\_blank”>Mozart – Eine kleine Nachtmusik</a></p>

<p><a href="https://www.youtube.com/watch?v=DLi25fgVuRs" target=”\_blank”>Puccini – 'O mio babbino caro' from Gianni Schicchi</a></p>

<p><a href="https://youtube.com/playlist?list=PLerDLiFhFsuoFwuThFGOFCLLtThSREnRx&si=ye\_u-Zc4iDUVIXil" target=”\_blank”>Painting Music Playlist</a></p>

</div>

<div class="col-sm-8 text-left">

<!--<h1 style="text-align: center">Welcome</h1>

<p style="text-align: center">Let us draw a grid of squares which we will color later. Please input the required fields for the color experiment!</p>

<hr>-->

<form style="padding-top: 1%" action="./run.html" method="get" id="myForm" onsubmit="return submitCheck();">

<input type="hidden" id="globalListOfItemsInput" name="globalListOfItems" value="">

<div class="row-sm-12 formItems">

<fieldset>

<h1>Welcome Experiment Painting Website</h1>

<h3>To start you off, the Following Buttons give you an option to run the experiment very quick and easily</h3>

<div class="col-sm-3"></div>

<div class="col-sm-3">

<button id="submit1" type="submit" style="margin-bottom: 60px; color: white; background-color: blue;">RUN EXPERIMENT</button>

</div>

<div class="col-sm-3">

<input type="button" id="randomize" value="FILL RANDOM EXPERIMENT" style=" color: white; background-color: blue;" onclick="randomize\_inputs();">

</div>

<div class="col-sm-3"></div>

</fieldset>

</div>

<div class="row-sm-12 formItems">

<fieldset>

<legend id = "var-legend">Dependent and Independent Variables</legend>

<div class="col-sm-4 formItems">

<label style="font-size: medium;">Independent</label><br>

<label for="independent1" style="font-size: small;">D, a single dimension that is used for square canvases.</label><br>

<input type="radio" id="independent1" name="independent" value="1" onchange="show\_inputs\_dependent(this);" required><br>

<label for="independent2" style="font-size: small;">X, the x-dimension, with the y-dimension held constant.</label><br>

<input type="radio" id="independent2" name="independent" value="2" onchange="show\_inputs\_dependent(this);"><br>

<label for="independent3" style="font-size: small;">R, the number of repetitions in the experiment</label><br>

<input type="radio" id="independent3" name="independent" value="3" onchange="show\_inputs\_dependent(this);"><br><br>

</div>

<div class="col-sm-4 formItems">

<label style="font-size: medium">Independent Value</label><br>

<select id="dependentList" multiple size="3" style="width: 180px; color: black;">

</select><br><br>

<span id="depErr1" class=" hide error">Nothing entered</span>

<span id="depErr2" class="block hide error">Too few items: a minimum of 4</span>

<span id="depErr3" class=" hide error">Too many items: a maximum of 12</span>

<span id="depErr4" class=" hide error">Min of 0</span>

<span id="depErr5" class=" hide error">Max of 99</span>

<span id="depErr8" class=" hide error">Max of 40</span>

<span id="depErr6" class=" hide error">X-values must be greater than or equal to previous entered</span><br>

<!--<p id="depErr7" class="help-block error">Test this!</p>-->

<input type="number" id="numberInput" style="color: black;" required><br><br>

<script>

document.getElementById('numberInput').addEventListener('keypress', function(event) {

if (event.key === 'Enter' || event.keyCode == 13) {

addItemToList();

event.preventDefault();

}

});

// e.key is the modern way of detecting keys

// e.keyCode is deprecated (left here for for legacy browsers support)

// keyup is not compatible with Jquery select(), Keydown is.

</script>

<button type="button" id="addNumber" name="addNumber" style="color: white; background-color: blue;" onclick="addItemToList();">Add Number</button>

<button type="button" id="removeNumber" name="removeNumber" style="color: white; background-color: blue;" onclick="removeItemFromList();">Remove selected Number</button><br><br>

</div>

<div class="col-sm-4 formItems">

<label id="repetitions\_label" style="font-size: small; display: none;" for="repetitions">Repetitions Value</label>

<span id="repErr1" class="help-block hide error">Nothing entered for Repetitions</span>

<span id="repErr2" class="help-block hide error">Repetitions must be greater than 0</span>

<span id="repErr3" class="help-block hide error">Repetitions must be less than 99</span>

<input type="number" id="repetitions" name="repetitions" style="display: none; color: black;">

<br><br>

<label id="dim\_y\_label" for="dim\_y" onload="checkDimY();" style="font-size: small;display: none;">Dimension Y Value</label>

<span id="dimYErr1" class="help-block hide error">Nothing entered for Dimension Y</span>

<span id="dimYErr2" class="help-block hide error">Dimension Y must be greater than 0</span>

<span id="dimYErr3" class="help-block hide error">Dimension Y must be less than 40</span>

<input type="number" id="dim\_y" name="dim\_y" value="" style="display: none; color: black;">

<br><br>

<label id="dim\_Y\_X\_label" for="dim\_Y\_X" style="font-size: small;display: none;">Square Dimension Value</label>

<span id="dimY\_XErr1" class="help-block hide error">Nothing entered for Dimension</span>

<span id="dimY\_XErr2" class="help-block hide error">Dimension must be greater than 0</span>

<span id="dimY\_XErr3" class="help-block hide error">Dimension must be less than 40</span>

<input type="number" id="dim\_Y\_X" name="dim\_Y\_X" value="" style="display: none; color: black;" onchange="setDimY\_X();">

</div>

<!--<div class="col-sm-3 formItems">

<label style="font-size: medium">Dependent</label><br>

<label for="dependent1" style="font-size: small;">D, a single dimension that is used for square canvases.</label><br>

<input type="radio" id="dependent1" name="dependent" value="1" onchange="updateOtherLists\_Depend\_Independ(this);"required><br>

<label for="dependent2" style="font-size: small;">X, the x-dimension, with the y-dimension held constant.</label><br>

<input type="radio" id="dependent2" name="dependent" value="2" onchange="updateOtherLists\_Depend\_Independ(this);"><br>

<label for="dependent3" style="font-size: small;">R, the number of repetitions in the experiment</label><br>

<input type="radio" id="dependent3" name="dependent" value="3" onchange="updateOtherLists\_Depend\_Independ(this);"><br><br>

</div>-->

</fieldset>

</div>

<fieldset>

<div class="row-sm-12">

<!-- <div class="col-sm-6 formItems"> -->

<!-- </div> -->

<div class="col-sm-3 formItems">

<legend class="formItems">Color Info</legend>

<label for="color1">Color 1</label><br>

<select id="color1" name="color1" style="color: black;" onchange="updateOtherLists(this);" onfocus="storeOldValue(this);" required>

<option id="option0" disabled selected value> -- select an option -- </option>

<option id="option1" value="red">red</option>

<option id="option2" value="blue">blue</option>

<option id="option3" value="green">green</option>

<option id="option4" value="yellow">yellow</option>

<option id="option5" value="purple">purple</option>

<option id="option6" value="orange">orange</option>

<option id="option7" value="pink">pink</option>

<option id="option8" value="brown">brown</option>

<option id="option9" value="black">black</option>

</select><br><br>

<label for="color2">Color 2</label><br>

<select id="color2" name="color2" style="color: black;" onchange="updateOtherLists(this);" onfocus="storeOldValue(this);"required>

<option disabled selected value> -- select an option -- </option>

<option value="red">red</option>

<option value="blue">blue</option>

<option value="green">green</option>

<option value="yellow">yellow</option>

<option value="purple">purple</option>

<option value="orange">orange</option>

<option value="pink">pink</option>

<option value="brown">brown</option>

<option value="black">black</option>

</select><br><br>

<label for="color3">Color 3</label><br>

<select id="color3" name="color3" style="color: black;" onchange="updateOtherLists(this);" onfocus="storeOldValue(this);" required>

<option disabled selected value> -- select an option -- </option>

<option value="red">red</option>

<option value="blue">blue</option>

<option value="green">green</option>

<option value="yellow">yellow</option>

<option value="purple">purple</option>

<option value="orange">orange</option>

<option value="pink">pink</option>

<option value="brown">brown</option>

<option value="black">black</option>

</select><br><br>

</div>

<div class="col-sm-4">

<legend class="formItems">Stopping Criteria</legend>

<div class="formItems">

<label style="font-size: medium">When should the experiment to terminate?</label><br>

<label for="termItem1" style="font-size: small;">Terminate When: the last unpainted square is painted for the first time.</label><br><input type="radio" id="termItem1" name="termItem" value="1" required><br>

<label for="termItem2" style="font-size: small;">Terminate When: the first time any square gets its second paint blob.</label><br><input type="radio" id="termItem2" name="termItem" value="2"><br>

<label for="termItem3" style="font-size: small;">Terminate When: the first time any square gets its third paint blob.</label><br><input type="radio" id="termItem3" name="termItem" value="3"><br>

<!-- <label for="termItem4"> Termination Item 4</label><br><input type="radio" id="termItem4" name="termItem" value="4"><br>

<label for="termItem5"> Termination Item 5</label><br><input type="radio" id="termItem5" name="termItem" value="5"><br><br> -->

</div>

</div>

<div class="col-sm-5">

<legend class="formItems">Song Choice</legend>

<div class="formItems">

<label style="font-size: medium">What song do you want to play while you paint?</label><br>

<div class="formItems col-sm-6">

<label for="song1" style="font-size: small;">Larry Owens - The Joy Of Painting</label><br><input type="radio" id="song1" name="song" value="sRkzqLudJPE" checked required><br>

<label for="song2" style="font-size: small;">Mozart – Eine kleine Nachtmusik</label><br><input type="radio" id="song2" name="song" value="oy2zDJPIgwc"><br>

<label for="song3" style="font-size: small;">Puccini – 'O mio babbino caro' from Gianni Schicchi</label><br><input type="radio" id="song3" name="song" value="DLi25fgVuRs"><br>

</div>

<div class="formItems col-sm-6">

<label for="song4" style="font-size: small;">J.S. Bach – Toccata and Fugue in D minor</label><br><input type="radio" id="song4" name="song" value="z69C2Jqp42Q"><br>

<label for="song5" style="font-size: small;">Beethoven – Symphony No.5 in C minor (shortened)</label><br><input type="radio" id="song5" name="song" value="W2qW6fOtAMY"><br>

<label for="song6" style="font-size: small;">Johann Strauss II - The Blue Danube Waltz</label><br><input type="radio" id="song6" name="song" value="\_CTYymbbEL4"><br>

</div>

<!-- <label for="termItem4"> Termination Item 4</label><br><input type="radio" id="termItem4" name="termItem" value="4"><br>

<label for="termItem5"> Termination Item 5</label><br><input type="radio" id="termItem5" name="termItem" value="5"><br><br> -->

</div>

</div>

</div>

</fieldset>

<div class="formItems">

<div class="col-sm-4"></div>

<div class="col-sm-2">

<button id="submit" type="submit" style="margin-bottom: 60px;color: white; background-color: blue;">Run Experiment!</button>

</div>

<div class="col-sm-2">

<button id="reload" type="button" style="margin-bottom: 60px;color: white; background-color: blue;" onclick="location.reload();">Restart Page Inputs</button>

</div>

<div class="col-sm-4"></div>

</div>

</form>

<div class="grid-container">

</div>

</div>

<div class="col-sm-2 sidenav" style="margin-bottom: 0px; background-color: unset;">

<p>Music Choices:</p>

<p><a href="https://www.youtube.com/watch?v=z69C2Jqp42Q" target=”\_blank”>J.S. Bach – Toccata and Fugue in D minor</a></p>

<p><a href="https://www.youtube.com/watch?v=W2qW6fOtAMY" target=”\_blank”>Beethoven – Symphony No.5 in C minor (shortened)</a></p>

<p><a href="https://www.youtube.com/watch?v=\_CTYymbbEL4" target=”\_blank”>Johann Strauss II - The Blue Danube Waltz</a></p>

<p style="margin-bottom: 14%;"><a href="https://youtube.com/playlist?list=PLerDLiFhFsuoFwuThFGOFCLLtThSREnRx&si=ye\_u-Zc4iDUVIXil" target=”\_blank”>Painting Music Playlist</a></p>

</div>

</div>

</div>

<footer class="container-fluid text-center">

<p style = "font-size: 14px;"> Copyright © 2023 Darkest Error </p>

</footer>

</body>

</html>

## Usage.html:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Painting Grid</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<script src="./myjs.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

<style>

/\* Remove the navbar's default margin-bottom and rounded borders \*/

.navbar {

margin-bottom: 0;

border-radius: 0;

}

/\* Set height of the grid so .sidenav can be 100% (adjust as needed) \*/

.row.content {height: 450px}

/\* Set gray background color and 100% height \*/

.sidenav {

padding-top: 20px;

background-color: #2f2f2fd4;

height: 100%;

}

/\* Set black background color, white text and some padding \*/

footer {

background-color: #555;

color: white;

padding: 15px;

position: fixed;

left: 0;

bottom: 0;

width: 100%;

}

/\* On small screens, set height to 'auto' for sidenav and grid \*/

@media screen and (max-width: 767px) {

.sidenav {

height: auto;

padding: 15px;

}

.row.content {height:auto;}

}

</style>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto auto auto auto auto auto auto;

background-color: unset;

padding: 10px;

}

.grid-item {

display:grid;

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

grid-template-columns: repeat(auto-fit, minmax(1px, 1fr));

grid-template-rows: repeat(auto-fit, minmax(1px, 1fr));

}

.container {

display: grid;

grid-gap: 10px;

grid-template-columns: repeat(auto-fill, minmax(min(200px,100%), 1fr));

}

.child {

display: flex;

align-items: center;

justify-content: center;

padding: 5px;

border: 3px solid #a07;

}

.sidenav{

height: 190%;

}

body {

margin:0;

color: white;

background-color: #023f3f;

}

</style>

</head>

<body>

<nav class="navbar navbar-inverse">

<div class="container-fluid">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#"><img src="img.png" width="50px" height="50px" style="margin-top: -15px !important; /\* 50% of your logo width \*/"></a>

</div>

<div class="collapse navbar-collapse" id="myNavbar">

<ul class="nav navbar-nav">

<li><a href="./index.html">Home</a></li>

<li class="active"><a href="./usage.html">Usage</a></li>

</ul>

<!-- Remove Login feature

<ul class="nav navbar-nav navbar-right">

<li><a href="#"><span class="glyphicon glyphicon-log-in"></span> Login</a></li>

</ul>

-->

</div>

</div>

</nav>

<div class="container-fluid text-center">

<div class="row content">

<div class="col-sm-2 text-left"></div>

<div class="col-sm-8 text-left" style="text-align: center;">

<h1>Basic usage</h1>

<p>You enter the inputs, and watch the magic happen as your experiments run!</p>

<hr style="margin-bottom: 5%;">

<img SRC="./using\_site.gif" style="margin-bottom: 5%;">

<h3>Start, Independent Value.</h3>

<p style="margin-bottom: 5%;">This is what makes each experiment unique. Then enter the Independent Values for the main Independent Value. You must enter a minimum of 4,

and there's a maximum of 12. Each of these list values must be larger than or equal to the previous value.

If the main Independent Value you chose is D or X, the maximum number you can enter is 40. If you choose R, you can enter up to 99.</p>

<h3>Now other Information</h3>

<p style="margin-bottom: 5%;">For those values that were not chosen for Independent they need to be fixed at a specific value.

Please fill out the fixed values which are either going to be Y value, Square Dimension value, and/or repetitions value.

With this do not forget to select your stopping criteria of choice.</p>

<h3>Now personalize your Experiment</h3>

<p style="margin-bottom: 5%;">Then, enter the three colors you want to drop onto your canvas and a song which you enjoy.</p>

<h3>Enjoy</h3>

<p style="margin-bottom: 5%;">Finally, sit back and enjoy watching the magic happen before your very eyes. The experiment you specified will be able to be watched in real-time or faster.</p>

<h1 style="margin-bottom: 14%;"><a href="./index.html">Let's get painting!</a></h1>

<div class="grid-container">

</div>

</div>

<!-- <div class="col-sm-2 sidenav" style="margin-bottom: 40px">

<div class="well">

<p>ADS</p>

</div>

<div class="well">

<p>ADS</p>

</div>

</div> -->

</div>

</div>

<div class="col-sm-2 text-left"></div>

<footer class="container-fluid text-center">

<p style = "font-size: 14px;"> Copyright © 2023 Darkest Error </p>

<!-- <p><a href="https://youtube.com/playlist?list=PLerDLiFhFsuoFwuThFGOFCLLtThSREnRx&si=ye\_u-Zc4iDUVIXil" target=”\_blank”>Painting Music Playlist</a></p> -->

</footer>

</body>

</html>

## Run.html:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Painting Grid</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<script src="./myjs.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>

<!--Audio-->

<script src="https://www.youtube.com/iframe\_api"></script>

<!--<script src="https://cdn.rawgit.com/labnol/files/master/yt.js" defer></script>-->

<!-- Color mixing -->

<script src="https://code.jquery.com/jquery-1.12.4.min.js"

integrity="sha384-nvAa0+6Qg9clwYCGGPpDQLVpLNn0fRaROjHqs13t4Ggj3Ez50XnGQqc/r8MhnRDZ"

crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-color/2.1.2/jquery.color.min.js"

integrity="sha384-Dmia8eQq5QAVU5VL95mpXzGE2n2oYxIv9aFwn+JAtQ3Y/4bWeHSjKUgYSJi/BovQ"

crossorigin="anonymous"></script>

<script src="https://scrtwpns.com/mixbox.js"></script>

<style>

/\* Remove the navbar's default margin-bottom and rounded borders \*/

.navbar {

margin-bottom: 0;

border-radius: 0;

}

/\* Set height of the grid so .sidenav can be 100% (adjust as needed) \*/

.row.content {height: 450px}

/\* Set gray background color and 100% height \*/

.sidenav {

padding-top: 20px;

background-color: unset;

height: 100%;

}

/\* Set black background color, white text and some padding \*/

footer {

background-color: #555;

color: white;

padding: 15px;

position: fixed;

left: 0;

bottom: 0;

width: 100%;

}

/\* On small screens, set height to 'auto' for sidenav and grid \*/

@media screen and (max-width: 767px) {

.sidenav {

height: auto;

padding: 15px;

background-color: unset;

}

.row.content {height:auto;}

}

</style>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto auto auto auto auto auto auto auto;

background-color: unset;

padding: 10px;

}

.grid-item {

display:grid;

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

grid-template-columns: repeat(auto-fit, minmax(1px, 1fr));

grid-template-rows: repeat(auto-fit, minmax(1px, 1fr));

}

.container {

display: grid;

grid-gap: 10px;

grid-template-columns: repeat(auto-fill, minmax(min(200px,100%), 1fr));

}

.child {

display: flex;

align-items: center;

justify-content: center;

padding: 5px;

border: 3px solid #a07;

}

.sidenav{

height: 190%;

background-color: unset;

}

#html-data-table {

border: 1px solid white; /\* Add white border around the table \*/

border-collapse: collapse;

width: 100%;

background-color: #32373A;

color: white;

}

/\* Style table header cells \*/

#html-data-table th {

border: 1px solid white; /\* Add border to header cells \*/

padding: 8px;

text-align: left;

}

/\* Style table cells \*/

#html-data-table td {

border: 1px solid white; /\* Add border to data cells \*/

padding: 8px;

}

#reduced-html-data-table {

border: 1px solid white; /\* Add white border around the table \*/

border-collapse: collapse;

width: 100%;

background-color: #32373A;

color: white;

}

/\* Style table header cells \*/

#reduced-html-data-table th {

border: 1px solid white; /\* Add border to header cells \*/

padding: 8px;

text-align: left;

}

/\* Style table cells \*/

#reduced-html-data-table td {

border: 1px solid white; /\* Add border to data cells \*/

padding: 8px;

}

body {

margin:0;

color: white;

background-color: #023f3f;

}

</style>

</head>

<body>

<nav class="navbar navbar-inverse">

<div class="container-fluid">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar" style="color: black;">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#"><img src="img.png" width="50px" height="50px" style="margin-top: -15px !important; /\* 50% of your logo width \*/"></a>

</div>

<div class="collapse navbar-collapse" id="myNavbar">

<ul class="nav navbar-nav">

<li><a href="./index.html">Home</a></li>

<li><a href="./usage.html">Usage</a></li>

<li class="active"><a href="#">Experiment</a></li>

</ul>

<!-- Remove Login feature

<ul class="nav navbar-nav navbar-right">

<li><a href="#"><span class="glyphicon glyphicon-log-in"></span> Login</a></li>

</ul>

-->

</div>

</div>

</nav>

<div class="container-fluid text-center">

<div class="row content">

<!-- <div class="col-sm-2 sidenav" style="margin-bottom: 40px;">

</div> -->

<div class="col-sm-12 text-left">

<h1 style="text-align: center;">Running Experiment!</h1>

<p style="text-align: center;">WARNING: Please do not reload the page, or you will be sent back to the previous screen.</p>

<hr>

</div>

<div class="col-sm-5" style="text-align: center" id="controls">

<h3 style="text-align: center;">Experiment Controls</h3>

<p id="t1" hidden>Terminating When: the last unpainted square is painted for the first time.</p>

<p id="t2" hidden>Terminating When: the first time any square gets its second paint blob.</p>

<p id="t3" hidden>Terminating When: the first time any square gets its third paint blob.</p>

<div class="slider" style="margin-bottom: 20px;">

<label for="speed">Speed Control<output id="showspeed" style="color: white;"></output></label>

<input type="range" min=".25" max="10" value="1" id="speed" name="speed" step=".25" list="speedsettings">

<datalist id="speedsettings">

<option>.25</option>

<option>.5</option>

<option>.75</option>

<option>1</option>

<option>1.25</option>

<option>1.5</option>

<option>1.75</option>

<option>2</option>

<option>2.25</option>

<option>2.5</option>

<option>2.75</option>

<option>3</option>

<option>3.25</option>

<option>3.5</option>

<option>3.75</option>

<option>4</option>

<option>4.25</option>

<option>4.5</option>

<option>4.75</option>

<option>5</option>

<option>5.25</option>

<option>5.5</option>

<option>5.75</option>

<option>6</option>

<option>6.25</option>

<option>6.5</option>

<option>6.75</option>

<option>7</option>

<option>7.25</option>

<option>7.5</option>

<option>7.75</option>

<option>8</option>

<option>8.25</option>

<option>8.5</option>

<option>8.75</option>

<option>9</option>

<option>9.25</option>

<option>9.5</option>

<option>9.75</option>

<option>10</option>

</datalist>

</div>

<!--

<label for="l\_x">X:</label>

<input type="number" id="l\_x" name="l\_x"><br><br>

<label for="l\_y">Y:</label>

<input type="number" id="l\_y" name="l\_y"><br><br>-->

<button type="button" onclick="pause();" style="color: black;">Stop</button>

<button type="button" onclick="unpause();" style="color: black;">Start</button><br><br>

<!--<button type="button" onclick="changeSizeStuff();">Change Size Stuff</button>-->

</div>

<div class="row col-sm-12" style="text-align: center" id="after\_run\_div1" hidden>

<h2 style="text-align: center;">Experiment Result Selection and Final Canvas</h2>

<hr>

</div>

<div class="row col-sm-6" style="text-align: center" id="after\_run\_div" hidden>

<h3 style="text-align: center;">Please select an experiment value to be Calculated</h3>

<div class="" style="text-align: center">

<!--<label style="font-size: medium; padding-top: 20px">Experiment Value</label><br>-->

<label for="A" style="font-size: small;">A: the number of paint drops put on the canvas before the painting halts</label><br><input type="checkbox" id="A" name="calc\_val" value="1"><br>

<label for="A1" style="font-size: small;">A1. The number of paint drops on the canvas of Color 1.</label><br><input type="checkbox" id="A1" name="calc\_val" value="2"><br>

<label for="A2" style="font-size: small;">A2. The number of paint drops on the canvas of Color 2.</label><br><input type="checkbox" id="A2" name="calc\_val" value="3"><br><br>

<label for="A3" style="font-size: small;">A3. The number of paint drops on the canvas of Color 3.</label><br><input type="checkbox" id="A3" name="calc\_val" value="4"><br><br>

<label for="B" style="font-size: small;">B: the maximum number of paint drops on any given square when the painting halts (that is, looking at all the squares, what is the largest number of paint drops that fell on one square?)</label><br><input type="checkbox" id="B" name="calc\_val" value="5"><br><br>

<label for="C" style="font-size: small;">C. the average number of paint drops over all the squares when the painting for this canvas halts</label><br><input type="checkbox" id="C" name="calc\_val" value="6" required><br><br>

<button style="margin-bottom: 16%; color: black;" onclick="graph\_table\_show();">CONTINUE</button>

</div>

</div>

<div class="col-sm-4" style="text-align: left;" id="update\_canvas">

<canvas id="canvas" role="presentation"></canvas>

</div>

<div class="row col-sm-12" id="graph\_table" style="margin-bottom: 100px;" hidden>

<div class="col-sm-6" style="text-align: center">

<script>

window.onload = function () {

var chart = new CanvasJS.Chart("chartContainer", {

animationEnabled: true,

zoomEnabled: true,

theme: "dark2",

title: {

text: "Growth in Internet Users Globally"

},

axisX: {

title: "Year",

valueFormatString: "####",

interval: 1

},

axisY: {

logarithmic: true, //change it to false

title: "Internet Users (Log)",

titleFontColor: "#6D78AD",

lineColor: "#6D78AD",

gridThickness: 0,

lineThickness: 1,

labelFormatter: addSymbols

},

axisY2: {

title: "Internet Users",

titleFontColor: "#51CDA0",

logarithmic: false, //change it to true

lineColor: "#51CDA0",

gridThickness: 0,

lineThickness: 1,

labelFormatter: addSymbols

},

legend: {

verticalAlign: "top",

fontSize: 16,

dockInsidePlotArea: true

},

data: [{

type: "line",

xValueFormatString: "####",

showInLegend: true,

name: "Log Scale",

dataPoints: [

{ x: 1, y: 1 },

{ x: 2, y: 2 },

{ x: 2, y: 2 },

{ x: 2, y: 2 },

{ x: 2, y: 2 },

]

},

{

type: "line",

xValueFormatString: "####",

axisYType: "secondary",

showInLegend: true,

name: "Linear Scale",

dataPoints: [

{ x: 1, y: 1 },

{ x: 2, y: 2 },

{ x: 2, y: 2 },

{ x: 2, y: 2 },

{ x: 2, y: 2 },

{ x: 2, y: 2 },

]

}]

});

chart.render();

function addSymbols(e) {

var suffixes = ["", "K", "M", "B", "T"];

var order = Math.max(Math.floor(Math.log(Math.abs(e.value)) / Math.log(1000)), 0);

if(order > suffixes.length - 1)

order = suffixes.length - 1;

var suffix = suffixes[order];

return CanvasJS.formatNumber(e.value / Math.pow(1000, order), "#,##0.##") + suffix;

}

}

</script>

<div id="chartContainer" style="height: 370px; width: 100%;"></div>

<script src="https://cdn.canvasjs.com/canvasjs.min.js"></script>

</div>

<div class="col-sm-6" style="text-align: center">

<table id="html-data-table">

<tr>

<th>Independent Variable</th>

<th>X-Dim</th>

<th>Y-Dim</th>

<th>Colors</th>

<th>Stopping Criteria</th>

<th>Repetitions</th>

<th>A Min</th>

<th>A Max</th>

<th>A Avg</th>

<th>B Max</th>

<th>C Avg</th>

</tr>

<tr>

<td>2</td>

<td>2</td>

<td>3</td>

<td>Red, Green, Blue</td>

<td>2nd Paint Blob</td>

<td>4</td>

<td>34</td>

<td>32</td>

<td>54</td>

<td>43</td>

<td>39</td>

</tr>

</table>

</div>

<div class="col-sm-6" style="text-align: center; margin-top: 30px;">

<table id="reduced-html-data-table">

<tr>

<th>Independent Variable</th>

<th>X-Dim</th>

<th>Y-Dim</th>

<th>Colors</th>

</tr>

<tr>

<td>2</td>

<td>2</td>

<td>3</td>

<td>Red, Green, Blue</td>

</tr>

</table>

</div>

<div id="data-table">

</div>

</div>

<div class="row col-sm-12" id="graph-container" style="margin-bottom: 16%;" hidden>

<div>

<button class="btn btn-success" onclick="show\_option()" style="color: black;">Make a new table/graph</button>

<button class="btn btn-warning" onclick="newExperiment()" style="color: black;">Abandon this experiment</button>

<button class="btn btn-danger" onclick="quitProgram()" style="color: black;">Quit the program</button>

</div>

</div>

<footer class="container-fluid text-center">

<p style = "font-size: 14px;"> Copyright © 2023 Darkest Error </p>

<div data-autoplay="1" data-loop="1" id="youtube-audio" hidden="hidden"></div>

</footer>

</body>

</html>

## Myjs.js:

///define global variables:

window.defaultWidthOfCanvas = 400;

window.defaultHeightOfCanvas = 400;

window.maxXSquares = defaultWidthOfCanvas/10;

window.maxYSquares = defaultHeightOfCanvas/10;

window.arrOfColors = new Array(maxXSquares); //max array size. //Note: first number is the amount of times it's been painted, other three number are RGB.

window.tempArrOfColors = new Array(maxXSquares); //same as above just used for each experiment.

window.colorRGBMap = {

red: 'rgb(255,0,0)',

blue: 'rgb(0,0,255)',

green: 'rgb(0,255,0)',

yellow: 'rgb(255,255,0)',

purple: 'rgb(128,0,128)',

orange: 'rgb(255,165,0)',

pink: 'rgb(255,192,203)',

brown: 'rgb(165,42,42)',

black: 'rgb(0,0,0)',

};

window.stage\_index = 0;

window.color1\_drops = [0];

window.color2\_drops = [0];

window.color3\_drops = [0];

window.totalPaintDrops = [0];

window.maxPaintDrops = [0];

window.averagePaintDrops = [0];

window.data\_points\_max = 60;

document.y\_title = ["",""];

document.table\_title = ["",""];

//used for size of grid:

window.x\_value = 0;

window.y\_value = 0;

var a = 0; // the total number of paint drops put on the canvas before the stopping criterion stops the painting.

var a1 = 0; // The number of paint drops on the canvas of Color 1.

var a2 = 0; // The number of paint drops on the canvas of Color 2.

var a3 = 0; // The number of paint drops on the canvas of Color 3.

var b = 0; // the maximum number of paint drops on any given square when the painting halts (that is, looking at all the squares,

//what is the largest number of paint drops that fell on one square?)

var c = 0; // the average number of paint drops over all the squares when the painting halts

var rands = {randNum1: 0, randNum2: 0};

var stopId; // id for setinterval

var computations = {

a: 0,

a1: 0,

a2: 0,

a3: 0,

b: 0,

c: 0

}

var s;

var termItem;

let data = window.performance.getEntriesByType("navigation")[0].type;

//force back to screen to fill in info if they try to reload or anything:

if ("navigate" != data){

window.location.href = './index.html';

}

window.addEventListener("load", (event) => {

// initialize canvas size:

setCanvasSize(defaultWidthOfCanvas, defaultHeightOfCanvas);

//setup the array:

for (let i = 0; arrOfColors.length > i; i++) { //ignoring 0 index

let arrayOfY = new Array(maxYSquares);

for (let i2 = 0; arrOfColors.length > i2; i2++) {

arrayOfY[i2] = [0,0,0,0];

}

arrOfColors[i] = arrayOfY;

}

//At this point array of colors should have all the values initialized to 0.

//Load all values:

//https://sentry.io/answers/how-to-get-values-from-urls-in-javascript/

const searchParams = new URLSearchParams(window.location.search);

window.searchParams = searchParams;

for (const param of searchParams) {

console.log("Data:",param);

}

//var dim\_x = searchParams.get('dim\_x');

//document.getElementById("l\_x").value = dim\_x;

//var dim\_y = searchParams.get('dim\_y');

//document.getElementById("l\_y").value = dim\_y;

window.colorOptions = [searchParams.get('color1'), searchParams.get('color2'), searchParams.get('color3')];

//<div data-video="" data-autoplay="1" data-loop="1" id="backGroundAudio" hidden="hidden"></div>

const audioItem = document.getElementById("youtube-audio");

//audioItem.setAttribute("data-autoplay","1");

//audioItem.setAttribute("data-loop","1");

//audioItem.setAttribute("hidden","hidden");

audioItem.setAttribute('data-video', searchParams.get('song'));

termItem = searchParams.get('termItem');

//console.log(searchParams.get('song'));

//Play SONG:

var e = document.getElementById("youtube-audio")

, t = document.createElement("img");

t.setAttribute("id", "youtube-icon"),

t.style.cssText = "cursor:pointer;cursor:hand",

e.appendChild(t);

var a = document.createElement("div");

a.setAttribute("id", "youtube-player"),

e.appendChild(a);

var o = function(e) {

var a = e ? "IDzX9gL.png" : "quyUPXN.png";

t.setAttribute("src", "https://i.imgur.com/" + a)

};

e.onclick = function() {

r.getPlayerState() === YT.PlayerState.PLAYING || r.getPlayerState() === YT.PlayerState.BUFFERING ? (r.pauseVideo(),

o(!1)) : (r.playVideo(),

o(!0))

};

var r = new YT.Player("youtube-player",{

height: "0",

width: "0",

videoId: e.dataset.video,

playerVars: {

autoplay: "1", //e.dataset.autoplay

loop: "1"//e.dataset.loop

},

events: {

onReady: function(e) {

r.setPlaybackQuality("small"),

o(r.getPlayerState() !== YT.PlayerState.CUED)

},

onStateChange: function(e) {

e.data === YT.PlayerState.ENDED && o(!1)

}

}

})

if(window.searchParams.get('termItem') == 1){

document.getElementById("t1").hidden = false;

} else if(window.searchParams.get('termItem') == 2){

document.getElementById("t2").hidden = false;

} else if(window.searchParams.get('termItem') == 3){

document.getElementById("t3").hidden = false;

}

window.current\_repititions = 1;

window.endExperiment = false;

window.isPaused = false;

audioItem.volume = .10;

window.current\_pos = 0;

startPainting(searchParams);

});

window.addEventListener("DOMContentLoaded", (event) => {

const display\_value = document.getElementById("showspeed");

const speed = document.getElementById("speed");

window.delay = 1000/speed.value;

speed.addEventListener("input", (event) => {

display\_value.textContent = speed.value;

display\_value.textContent = event.target.value;

});

speed.addEventListener("change", (event) => {

console.log(window.performance.now());

console.log(1000/speed.value, "ms");

window.delay = 1000/speed.value;

});

});

function setCanvasSize(width, height) {

let canvas = document.getElementById("canvas");

canvas.width = width;

canvas.height = height;

}

function drawInitialShape(x,y) {

const canvas = document.getElementById("canvas")

const ctx = document.getElementById("canvas").getContext("2d");

ctx.clearRect(0, 0, canvas.width, canvas.height); //clear canvas

//Make block size based on largest length choice:

window.canvasBlockSize = Math.floor((defaultWidthOfCanvas/x <= defaultWidthOfCanvas/y) ? defaultWidthOfCanvas/x : defaultWidthOfCanvas/y);

if(x>=1 && x<= maxXSquares && y>=1 && y <= maxYSquares){

for (let i = 0; i < y; i++) {

for (let j = 0; j < x; j++) {

ctx.strokeStyle = 'rgb(0,0,0)';

ctx.beginPath();

ctx.rect(1+canvasBlockSize\*j,1+canvasBlockSize\*i,canvasBlockSize-2,canvasBlockSize-2);

ctx.stroke();

}

}

} else {

//output error message

}

}

function draw() {

resetTempArrayOfColors();

x = window.x\_value;

y = window.y\_value;

const canvas = document.getElementById("canvas")

const ctx = document.getElementById("canvas").getContext("2d");

//Make block size based on largest length choice:

window.canvasBlockSize = Math.floor((defaultWidthOfCanvas/x <= defaultWidthOfCanvas/y) ? defaultWidthOfCanvas/x : defaultWidthOfCanvas/y);

if(x>=1 && x<= maxXSquares && y>=1 && y <= maxYSquares){

ctx.clearRect(0, 0, canvas.width, canvas.height); //clear canvas

for (let i = 0; i < y; i++) {

for (let j = 0; j < x; j++) {

ctx.strokeStyle = 'rgb(0,0,0)';

ctx.beginPath();

ctx.rect(1+canvasBlockSize\*j,1+canvasBlockSize\*i,canvasBlockSize-2,canvasBlockSize-2);

ctx.stroke();

}

}

} else {

//output error message

}

}

function fillXY(x,y,colorChoice){

const ctx = document.getElementById("canvas").getContext("2d");

if (x >= 1 && x <= maxXSquares && y >= 1 && y <= maxYSquares) {

var colorRGB = colorRGBMap[colorChoice];

//console.log("Painting an element");

//console.log("Color choice is: " + colorChoice);

if (window.tempArrOfColors[x-1][y-1][0] > 0) {

colorRGB = mixColors(colorRGB, window.tempArrOfColors[x-1][y-1][1], window.tempArrOfColors[x-1][y-1][2], window.tempArrOfColors[x-1][y-1][3]);

console.log("Painted an element for the", window.tempArrOfColors[x-1][y-1][0] + 1, "time");

}

var colorObj = $.Color(colorRGB);

arrOfColors[x-1][y-1][0] += 1; //keep track of total paints

window.tempArrOfColors[x-1][y-1][0] += 1;

window.tempArrOfColors[x-1][y-1][1] = colorObj.red();

window.tempArrOfColors[x-1][y-1][2] = colorObj.green();

window.tempArrOfColors[x-1][y-1][3] = colorObj.blue();

ctx.fillStyle = colorRGB;

ctx.strokeStyle = 'rgb(0,0,0)';

ctx.fillRect((2+canvasBlockSize \* x)-canvasBlockSize, (2+canvasBlockSize \* y)-canvasBlockSize, canvasBlockSize-4, canvasBlockSize-4);

switch (parseInt(termItem)) {

case 1: // Terminate when last unpainted square is painted for the first time

console.log("R V: ", window.repititions, "AND", window.current\_repititions);

if (checkAllElementsPainted()) {

clearInterval(stopId);

clearTimeout(stopId);

console.log("Stopped1");

window.fullStop = true;

window.current\_repititions += 1;

}

if(window.repititions < window.current\_repititions){

clearInterval(stopId);

clearTimeout(stopId);

window.current\_repititions = 1;

console.log("REPETITION STOP");

saveAllValues();

window.color1\_drops.push(0);

window.color2\_drops.push(0);

window.color3\_drops.push(0);

window.totalPaintDrops.push(0);

window.maxPaintDrops.push(0);

window.averagePaintDrops.push(0);

window.stage\_index++;

window.fullStop = true;

window.current\_pos += 1;

console.log("POS:",window.current\_pos, " vs ", window.max\_pos);

if(window.current\_pos != window.max\_pos){

startPainting(window.searchParams);

} else {

window.endExperiment = true;

}

}

if(window.current\_pos != window.max\_pos && checkAllElementsPainted()) startPainting(window.searchParams);

break;

case 2: // Terminate when a square is painted for the second time

console.log("R V: ", window.repititions, "AND", window.current\_repititions);

if (window.tempArrOfColors[x-1][y-1][0] > 1) {

console.log("Stopped2");

window.fullStop = true;

clearInterval(stopId);

clearTimeout(stopId);

window.current\_repititions += 1;

}

if(window.repititions < window.current\_repititions){

clearInterval(stopId);

clearTimeout(stopId);

window.current\_repititions = 1;

console.log("REPETITION STOP");

saveAllValues();

window.color1\_drops.push(0);

window.color2\_drops.push(0);

window.color3\_drops.push(0);

window.totalPaintDrops.push(0);

window.maxPaintDrops.push(0);

window.averagePaintDrops.push(0);

window.stage\_index++;

window.fullStop = true;

window.current\_pos += 1;

console.log("POS:",window.current\_pos, " vs ", window.max\_pos);

if(window.current\_pos != window.max\_pos){

startPainting(window.searchParams);

} else {

window.endExperiment = true;

}

}

if(window.current\_pos != window.max\_pos && window.tempArrOfColors[x-1][y-1][0] > 1) {

startPainting(window.searchParams)

}

break;

case 3: // Terminate when a square is painted for the third time

console.log("R V: ", window.repititions, "AND", window.current\_repititions);

if (window.tempArrOfColors[x-1][y-1][0] > 2) {

console.log("Stopped3");

window.fullStop = true;

clearInterval(stopId);

clearTimeout(stopId);

window.current\_repititions += 1;

}

if(window.repititions < window.current\_repititions){

clearInterval(stopId);

clearTimeout(stopId);

window.current\_repititions = 1;

console.log("REPETITION STOP");

saveAllValues();

window.color1\_drops.push(0);

window.color2\_drops.push(0);

window.color3\_drops.push(0);

window.totalPaintDrops.push(0);

window.maxPaintDrops.push(0);

window.averagePaintDrops.push(0);

window.stage\_index++;

window.fullStop = true;

window.current\_pos += 1;

console.log("POS:",window.current\_pos, " vs ", window.max\_pos);

if(window.current\_pos != window.max\_pos){

startPainting(window.searchParams);

} else {

window.endExperiment = true;

}

}

if(window.current\_pos != window.max\_pos && window.tempArrOfColors[x-1][y-1][0] > 2) {

startPainting(window.searchParams);

}

break;

default:

console.log("Invalid termItem:", termItem);

}

}

}

function fillRandomCellWithRandomColor(){

rands.randNum1 = Math.floor(Math.random()\*window.x\_value+1); // from 1 to x

rands.randNum2 = Math.floor(Math.random()\*window.y\_value+1); // from 1 to y

var randomColor = Math.floor(Math.random()\*3); // from 0 to 2

if(randomColor == 0) window.color1\_drops[window.stage\_index]++;

if(randomColor == 1) window.color2\_drops[window.stage\_index]++;

if(randomColor == 2) window.color3\_drops[window.stage\_index]++;

var colorChoice = colorOptions[randomColor];

fillXY(rands.randNum1, rands.randNum2, colorChoice);

}

function mixColors(color1, color2R, color2G, color2B) {

color2 = 'rgb(' + color2R + ',' + color2G + ',' + color2B + ')';

var mixingRatio = 0.5;

var resultColor = mixbox.lerp(color1, color2, mixingRatio);

return resultColor;

}

function changeSizeStuff() {

const can = document.getElementById('canvas');

const update\_div = document.getElementById('update\_canvas');

const controls = document.getElementById('controls');

const radio\_choice = document.getElementById('after\_run\_div');

const radio\_choice\_heading = document.getElementById('after\_run\_div1');

can.style.width = '300px';

can.style.height = '300px';

update\_div.style.textAlign = 'center';

update\_div.classList.remove('col-sm-4');

update\_div.classList.add('col-sm-6');

controls.hidden = true;

radio\_choice.hidden = false;

radio\_choice\_heading.hidden = false;

}

function paintOne() {

//stopId = setInterval(fillRandomCellWithRandomColor, 1000);

fillRandomCellWithRandomColor();

console.log("Interval started");

}

function paintMany() {

for (var i = 0; i < repititions; i++) {

paintOne();

}

}

function checkPaint(){

if(window.timeStamp + window.delay <= window.performance.now()) {

//console.log(window.timeStamp + window.delay, " VS ", window.performance.now());

//console.log("TIME STAMP HIT!");

window.timeStamp = window.performance.now();

paintOne();

}

clearInterval(stopId);

console.log(!window.fullStop);

if(!window.fullStop && (window.current\_pos != window.max\_pos)){

stopId = setInterval(checkPaint, 25);

} else if (window.current\_pos == window.max\_pos) {

console.log("COMPLETED EXPERIMENT!");

changeSizeStuff();

}

}

function startPainting(searchParams) {

console.log("CALL TO START PAINT!");

let independent\_value = searchParams.get('independent');

if(searchParams.get('independent') == 1){

let DIMS = searchParams.get('globalListOfItems').split(',');

window.max\_pos = DIMS.length;

const dim\_XY = DIMS[window.current\_pos];

console.log("Running", dim\_XY);

window.x\_value = dim\_XY;

window.y\_value = dim\_XY;

window.fullStop = false;

if(!window.endExperiment) draw();

window.repititions = searchParams.get('repetitions');

window.timeStamp = window.performance.now();

checkPaint();

//stopId = setInterval(checkPaint, 25);

} else if(searchParams.get('independent') == 2){

let DIMXS = searchParams.get('globalListOfItems').split(',');

window.max\_pos = DIMXS.length;

const dim\_X = DIMXS[window.current\_pos];

console.log("Running", dim\_X);

window.fullStop = false;

let dim\_y = searchParams.get('dim\_y');

window.x\_value = dim\_X;

window.y\_value = dim\_y;

if(!window.endExperiment) draw();

window.repititions = searchParams.get('repetitions');

window.timeStamp = window.performance.now();

checkPaint();

//stopId = setInterval(checkPaint, 25);

} else if(searchParams.get('independent') == 3){

let REPS = searchParams.get('globalListOfItems').split(',');

window.max\_pos = REPS.length;

const rep\_value = REPS[window.current\_pos];

console.log("Running", rep\_value);

window.repititions = rep\_value;

console.log([searchParams.get('globalListOfItems').split(',')]);

console.log(Number(rep\_value));

window.fullStop = false;

let dim\_XY = searchParams.get('dim\_Y\_X');

window.x\_value = dim\_XY;

window.y\_value = dim\_XY;

if(!window.endExperiment) draw();

window.timeStamp = window.performance.now();

checkPaint();

//stopId = setInterval(checkPaint, 25);

} else {

console.log("INVALID INDEPENDENT VALUE: ", independent\_value);

}

}

function checkAllElementsPainted() {

var xMax = window.x\_value;

var yMax = window.y\_value;

for (let iterX = 0; iterX < xMax; iterX++) {

for (let iterY = 0; iterY < yMax; iterY++) {

if (window.tempArrOfColors[iterX][iterY][0] == 0) {

//console.log("window.tempArrOfColors[x=" + iterX + "][y=" + iterY + "][0] == 0, is not painted");

return false;

}

}

}

return true;

}

function pause(){

clearInterval(stopId);

window.isPaused = true;

}

function unpause(){

if(window.isPaused){

window.timeStamp = window.performance.now();

window.isPaused = false;

checkPaint();

}

}

function resetTempArrayOfColors(){

//setup the array:

for (let i = 0; window.tempArrOfColors.length > i; i++) { //ignoring 0 index

let arrayOfY = new Array(maxYSquares);

for (let i2 = 0; window.tempArrOfColors.length > i2; i2++) {

arrayOfY[i2] = [0,0,0,0];

}

window.tempArrOfColors[i] = arrayOfY;

}

//At this point array of colors should have all the values initialized to 0.

}

function graph\_table\_show(){

if(validate\_selection()) {

document.getElementById('canvas').hidden = true;

document.getElementById('update\_canvas').hidden = true;

document.getElementById('after\_run\_div').hidden = true;

document.getElementById('after\_run\_div1').hidden = true;

document.getElementById('graph\_table').hidden = false;

document.getElementById('graph-container').hidden = false;

set\_graph\_values();

produce\_graph();

} else {

alert("Please only select One or Two and no more than that for results!");

}

}

let independentVariable;

let dependentVariables;

function startExperiment() {

// Code for setting up and running the experiment

// Populate the experiment results in the table

showTable();

}

function showTable() {

// Code for displaying the experiment table

//document.getElementById("table-container").hidden = false;

}

function showGraph() {

// Code for generating and displaying the graph

//document.getElementById("table-container").hidden = true;

//document.getElementById("graph-container").hidden = false;;

}

function newTable() {

// Code for creating a new table/graph from the current experimental data

// This might involve recalculating or updating values

showTable();

}

function newExperiment() {

// Code for abandoning the current experiment and starting a new one

window.location.href = "index.html";

}

function quitProgram() {

// Code for quitting the program

alert("Thanks for using the program!");

// Redirect to the opening page of the website

window.location.href = "usage.html";

}

function show\_option(){

//document.getElementById("table-container").hidden = false;

//document.getElementById("graph-container").hidden = false;

document.getElementById('canvas').hidden = false;

document.getElementById('update\_canvas').hidden = false;

document.getElementById('after\_run\_div').hidden = false;

document.getElementById('after\_run\_div1').hidden = false;

document.getElementById('graph\_table').hidden = true;

document.getElementById('graph-container').hidden = true;

}

function produce\_graph() {

if(document.y\_title[0] != "" && document.y\_title[1] != ""){

let new\_string = document.y\_title[0] + " and " + document.y\_title[1];

var chart = new CanvasJS.Chart("chartContainer", {

animationEnabled: true,

zoomEnabled: true,

theme: "dark2",

title: {

text: document.table\_title,

fontSize: 20

},

axisX: {

title: document.x\_title,

titleFontColor: "#6D78AD",

lineColor: "#6D78AD",

minimum: 0,

//maximum: window.data\_points\_max+10,

//valueFormatString: "##",

interval: 10

},

axisY: {

//logarithmic: true, //change it to false

title: new\_string,

titleFontColor: "#6D78AD",

lineColor: "#6D78AD",

//gridThickness: 0,

//lineThickness: 1,

//labelFormatter: addSymbols

},

legend: {

verticalAlign: "top",

fontSize: 16,

dockInsidePlotArea: true

},

data: [{

type: "line",

//xValueFormatString: "##",

showInLegend: true,

name: document.y\_title[0],

dataPoints: window.data\_points1

},

{

type: "line",

//xValueFormatString: "####",

//axisYType: "secondary",

showInLegend: true,

name: document.y\_title[1],

dataPoints: window.data\_points2

}

]

});

chart.render();

} else {

var chart = new CanvasJS.Chart("chartContainer", {

animationEnabled: true,

zoomEnabled: true,

theme: "dark2",

title: {

text: document.table\_title,

fontSize: 20

},

axisX: {

title: document.x\_title,

titleFontColor: "#6D78AD",

lineColor: "#6D78AD",

minimum: 0,

//maximum: window.data\_points\_max+10,

//valueFormatString: "##",

interval: 10

},

axisY: {

//logarithmic: true, //change it to false

title: document.y\_title[0],

titleFontColor: "#6D78AD",

lineColor: "#6D78AD",

//gridThickness: 0,

//lineThickness: 1,

//labelFormatter: addSymbols

}/\*,

legend: {

verticalAlign: "top",

fontSize: 16,

dockInsidePlotArea: true

}\*/,

data: [{

type: "line",

//xValueFormatString: "##",

//showInLegend: true,

//name: "Experiment Change",

dataPoints: window.data\_points1

},

{

type: "line",

//xValueFormatString: "####",

//axisYType: "secondary",

//showInLegend: true,

//name: "Linear Scale",

dataPoints: window.data\_points2

}

]

});

chart.render();

}

function addSymbols(e) {

var suffixes = ["", "K", "M", "B", "T"];

var order = Math.max(Math.floor(Math.log(Math.abs(e.value)) / Math.log(1000)), 0);

if(order > suffixes.length - 1)

order = suffixes.length - 1;

var suffix = suffixes[order];

return CanvasJS.formatNumber(e.value / Math.pow(1000, order), "#,##0.##") + suffix;

}

}

function set\_graph\_values(){

let calc\_value = [0,0];

var ele = document.getElementsByName('calc\_val');

let index\_temp\_checked = 0;

for (let i = 0; i < ele.length; i++) {

if (ele[i].checked) {

calc\_value[index\_temp\_checked++] = ele[i].value;

}

}

console.log("calc\_value (dependent)",calc\_value);

if(calc\_value[0] == 1){

document.table\_title[0] = "A: The number of paint drops put on the canvas before the painting halted";

document.y\_title[0] = "Total Paint Drops";

} else if(calc\_value[0] == 2){

document.table\_title[0] = "A1. The number of paint drops on the canvas of Color 1";

document.y\_title[0] = "Total Color 1 Paint Drops";

} else if(calc\_value[0] == 3){

document.table\_title[0] = "A2. The number of paint drops on the canvas of Color 2";

document.y\_title[0] = "Total Color 2 Paint Drops";

} else if(calc\_value[0] == 4){

document.table\_title[0] = "A3. The number of paint drops on the canvas of Color 3";

document.y\_title[0] = "Total Color 3 Paint Drops";

} else if(calc\_value[0] == 5){

document.table\_title[0] = "B: the maximum number of paint drops on any given square when the painting halted";

document.y\_title[0] = "Max Paint Drops (single square)";

} else if(calc\_value[0] == 6){

document.table\_title[0] = "C. the average number of paint drops over all the squares when the painting for this canvas halted";

document.y\_title[0] = "Average Paint Drops";

}

if(calc\_value[1] == 1){

document.table\_title[1] = "A: The number of paint drops put on the canvas before the painting halted";

document.y\_title[1] = "Total Paint Drops";

} else if(calc\_value[1] == 2){

document.table\_title[1] = "A1. The number of paint drops on the canvas of Color 1";

document.y\_title[1] = "Total Color 1 Paint Drops";

} else if(calc\_value[1] == 3){

document.table\_title[1] = "A2. The number of paint drops on the canvas of Color 2";

document.y\_title[1] = "Total Color 2 Paint Drops";

} else if(calc\_value[1] == 4){

document.table\_title[1] = "A3. The number of paint drops on the canvas of Color 3";

document.y\_title[1] = "Total Color 3 Paint Drops";

} else if(calc\_value[1] == 5){

document.table\_title[1] = "B: the maximum number of paint drops on any given square when the painting halted";

document.y\_title[1] = "Max Paint Drops (single square)";

} else if(calc\_value[1] == 6){

document.table\_title[1] = "C. the average number of paint drops over all the squares when the painting for this canvas halted";

document.y\_title[1] = "Average Paint Drops";

}

console.log("independent option",searchParams.get('independent'));

window.graph\_array = Array();

let independent\_value = searchParams.get('independent');

window.data\_points1 = Array();

window.data\_points2 = Array();

if(searchParams.get('independent') == 1){

let DIMS = searchParams.get('globalListOfItems').split(',');

window.max\_pos = DIMS.length;

const dim\_XY = DIMS[window.current\_pos];

document.x\_title = "Square Dimension (x,y)";

for(let i = 0; i < DIMS.length; i++){

let max = 0;

if(DIMS[i] > max) window.data\_points\_max = DIMS[i];

let values1 = {

x: DIMS[i],

y: getyvalue(calc\_value[0],i)

}

window.data\_points1.push(values1);

if(calc\_value[1] > 0) {

let values2 = {

x: DIMS[i],

y: getyvalue(calc\_value[1], i)

}

window.data\_points2.push(values2);

}

}

//window.data\_points = [{ x: 1994, y: 25437639 }, { x: 1995, y: 44866595 }, { x: 1996, y: 77583866 }];

/\*

console.log("Running", dim\_XY);

window.x\_value = dim\_XY;

window.y\_value = dim\_XY;

window.fullStop = false;

if(!window.endExperiment) draw();

window.repititions = searchParams.get('repetitions');

window.timeStamp = window.performance.now();

checkPaint();

\*/

//stopId = setInterval(checkPaint, 25);

} else if(searchParams.get('independent') == 2){

let DIMXS = searchParams.get('globalListOfItems').split(',');

window.max\_pos = DIMXS.length;

const dim\_X = DIMXS[window.current\_pos];

document.x\_title = "Dimension X";

for(let i = 0; i < DIMXS.length; i++){

let max = 0;

if(DIMXS[i] > max) window.data\_points\_max = DIMXS[i];

let values1 = {

x: DIMXS[i],

y: getyvalue(calc\_value[0],i)

}

window.data\_points1.push(values1);

if(calc\_value[1] > 0) {

let values2 = {

x: DIMXS[i],

y: getyvalue(calc\_value[1], i)

}

window.data\_points2.push(values2);

}

}

/\*

console.log("Running", dim\_X);

window.fullStop = false;

let dim\_y = searchParams.get('dim\_y');

window.x\_value = dim\_X;

window.y\_value = dim\_y;

if(!window.endExperiment) draw();

window.repititions = searchParams.get('repetitions');

window.timeStamp = window.performance.now();

checkPaint();\*/

//stopId = setInterval(checkPaint, 25);

} else if(searchParams.get('independent') == 3){

let REPS = searchParams.get('globalListOfItems').split(',');

window.max\_pos = REPS.length;

const rep\_value = REPS[window.current\_pos];

document.x\_title = "Repetitions";

for(let i = 0; i < REPS.length; i++){

let max = 0;

if(REPS[i] > max) window.data\_points\_max = REPS[i];

let values1 = {

x: REPS[i],

y: getyvalue(calc\_value,i)

}

window.data\_points1.push(values1);

if(calc\_value[1] > 0) {

let values2 = {

x: REPS[i],

y: getyvalue(calc\_value[1], i)

}

window.data\_points2.push(values2);

}

}

/\*

console.log("Running", rep\_value);

window.repititions = rep\_value;

console.log([searchParams.get('globalListOfItems').split(',')]);

console.log(Number(rep\_value));

window.fullStop = false;

let dim\_XY = searchParams.get('dim\_Y\_X');

window.x\_value = dim\_XY;

window.y\_value = dim\_XY;

if(!window.endExperiment) draw();

window.timeStamp = window.performance.now();

checkPaint();\*/

//stopId = setInterval(checkPaint, 25);

} else {

console.log("INVALID INDEPENDENT VALUE: ", independent\_value);

}

}

function getyvalue(calc\_value,rep){

console.log(calc\_value);

if(calc\_value == 1){ //total point drops in a given run.

return window.totalPaintDrops[rep];

} else if(calc\_value == 2){

return window.color1\_drops[rep];

} else if(calc\_value == 3){

return window.color2\_drops[rep];

} else if(calc\_value == 4){

return window.color3\_drops[rep];

} else if(calc\_value == 5){

return window.maxPaintDrops[rep];

} else if(calc\_value == 6){

return window.averagePaintDrops[rep];

}

}

//This needs to be fixed:

function getTotalPaintDrops(){

let total = 0;

for (let iterX = 0; iterX < maxXSquares; iterX++) {

for (let iterY = 0; iterY < maxYSquares; iterY++) {

total += arrOfColors[iterX][iterY][0];

}

}

return total;

}

function getMaxPaintDrops(){

let max = 0;

for (let iterX = 0; iterX < maxXSquares; iterX++) {

for (let iterY = 0; iterY < maxYSquares; iterY++) {

if(max < arrOfColors[iterX][iterY][0]) max = arrOfColors[iterX][iterY][0];

}

}

return max;

}

function getAveragePaintDrops(){

console.log("total paint drops",window.totalPaintDrops[window.stage\_index]);

console.log("grid",(window.x\_value \* window.y\_value));

return window.totalPaintDrops[window.stage\_index] / (window.y\_value \* window.x\_value);

}

function saveAllValues(){

window.totalPaintDrops[window.stage\_index] = getTotalPaintDrops();

window.maxPaintDrops[window.stage\_index] = getMaxPaintDrops();

window.averagePaintDrops[window.stage\_index] = getAveragePaintDrops();

}

function validate\_selection(){

var ele = document.getElementsByName('calc\_val');

let index\_temp\_checked = 0;

for (let i = 0; i < ele.length; i++) {

if (ele[i].checked) {

if(++index\_temp\_checked > 2) return false;

}

}

return true;

}

## Home.js:

var checkAll = [0, 0, 0, 0]; //Acts as a "key" for X-dim, Y-dim, repetitions, and x-values

//declare empty array to put the options in

let optionsArray = new Array();

function change\_music(e) {

e.preventDefault();

let elm = e.target;

//console.log(elm);

var audio = document.getElementById('backGroundAudio');

var source = document.getElementById('audio');

source.src = elm.getAttribute('data-value');

audio.load(); //call this to just preload the audio without playing

audio.play(); //call this to play the song right away

audio.stop();

};

window.addEventListener("load", (event) => {

//get options from user input and push to array

optionsArray.push(document.getElementById("option1").value);

optionsArray.push(document.getElementById("option2").value);

optionsArray.push(document.getElementById("option3").value);

optionsArray.push(document.getElementById("option4").value);

optionsArray.push(document.getElementById("option5").value);

optionsArray.push(document.getElementById("option6").value);

optionsArray.push(document.getElementById("option7").value);

optionsArray.push(document.getElementById("option8").value);

optionsArray.push(document.getElementById("option9").value);

//console.log(optionsArray);

});

function confirmInput(){

//let dimX = document.getElementById("dim\_x");

//let dimY = document.getElementById("dim\_y");

}

function storeOldValue(ele) {

// Store the current value on focus and on change

document.previousColorSelected = ele.value;

//console.log("Value stored = ", ele.value);

}

let globalListOfItems = []; //Ascending list of x-values

let indValue = "";

//Only checks globalListOfItems when button is clicked

function addItemToList(){

let listOfItems = document.getElementById("dependentList");

let input = document.getElementById("numberInput");

globalListOfItems = globalListOfItems.filter(item => item !== undefined && item !== null && item !== ""); //Deletes null inputs in globalListOfItems array

let depErr1 = document.getElementById("depErr1"); //Nothing entered

let depErr4 = document.getElementById("depErr4"); //Input is too small

let depErr5 = document.getElementById("depErr5"); //Input is too big

let depErr8 = document.getElementById("depErr8"); //Input is too big for dimensions

let depErr6 = document.getElementById("depErr6"); //Input smaller than previous input

let tooBigErr;

let inputValueAsNumber = parseInt(input.value);

let maxNum = 40;

if(indValue === "1" || indValue === "2") {

maxNum = 40;

tooBigErr = document.getElementById("depErr8"); //Input is too big 40

}

else {

maxNum = 99;

tooBigErr = document.getElementById("depErr5"); //Input is too big 99

}

//If input is correct, push to globalListOfItems, else sho an error message

if (globalListOfItems.length === 0) {

if(input.value !== "" && inputValueAsNumber >= 0 && inputValueAsNumber <= maxNum) {

listOfItems.add(new Option(input.value, input.value), undefined);

globalListOfItems.push(input.value);

}

else {

if(input.value === "") {

showMessage(depErr1);

}

else {

hideMessage(depErr1);

}

if(inputValueAsNumber < 1) {

showMessage(depErr4);

}

else {

hideMessage(depErr4);

}

if(inputValueAsNumber > maxNum) {

showMessage(tooBigErr);

}

else {

hideMessage(tooBigErr);

}

}

}

else if (globalListOfItems.length >= 1) {

if(input.value !== "" && (inputValueAsNumber >= parseInt(globalListOfItems[globalListOfItems.length - 1])) && inputValueAsNumber >= 0 && inputValueAsNumber <= maxNum) {

listOfItems.add(new Option(input.value, input.value), undefined);

globalListOfItems.push(input.value);

if (globalListOfItems.length > 4 || globalListOfItems.length < 12) {

//checkAll[0] = 1;

change();

}

}

else {

//checkAll[0] = 0;

change();

if(input.value === "") {

showMessage(depErr1);

}

else {

hideMessage(depErr1);

}

if (inputValueAsNumber < parseFloat(globalListOfItems[globalListOfItems.length - 1])){

showMessage(depErr6);

//document.getElementById("depErr7").textContent = globalListOfItems; //For Testing!

}

else if (inputValueAsNumber >= parseFloat(globalListOfItems[globalListOfItems.length - 1])) {

hideMessage(depErr6);

}

if(inputValueAsNumber < 1) {

showMessage(depErr4);

}

else {

hideMessage(depErr4);

}

if(inputValueAsNumber > maxNum) {

showMessage(tooBigErr);

}

else {

hideMessage(tooBigErr);

}

}

}

//hideMessage(depErr5);

//hideMessage(depErr8);

document.getElementById("globalListOfItemsInput").value = globalListOfItems.join(",");

input.value = ""; //finish by resetting value of input feild.

}

function removeItemFromList(){

let listOfItems = document.getElementById("dependentList");

let selectedOptions = $('option:selected', listOfItems);

// Remove selected items from globalListofItems

for (let i = 0; i < selectedOptions.length; i++) {

let removedValue = selectedOptions[i].value;

let indexToRemove = globalListOfItems.indexOf(removedValue);

if (indexToRemove !== -1) {

globalListOfItems.splice(indexToRemove, 1);

}

}

// Remove selected items from dependentList

selectedOptions.remove();

change();

document.getElementById("globalListOfItemsInput").value = globalListOfItems.join(",");

//Checks the length of globalListOfItems when items are deleted

if (globalListOfItems.length < 3 || globalListOfItems.length > 12) {

checkAll[0] = 0;

input.required = true;

} else {

checkAll[0] = 1;

input.required = false;

}

}

function updateOtherLists(ele){ //passed in the element that called the function

let selectBox1 = document.getElementById("color1");

let selectBox2 = document.getElementById("color2");

let selectBox3 = document.getElementById("color3");

if(selectBox1.id === ele.id){

let lastValue = document.previousColorSelected;

removeSelected(selectBox2, selectBox1.value);

removeSelected(selectBox3, selectBox1.value);

if(document.previousColorSelected != ""){

selectBox2.append(new Option(document.previousColorSelected, document.previousColorSelected), undefined);

selectBox3.append(new Option(document.previousColorSelected, document.previousColorSelected), undefined);

}

}

if(selectBox2.id === ele.id){

removeSelected(selectBox1, selectBox2.value);

removeSelected(selectBox3, selectBox2.value);

if(document.previousColorSelected != ""){

selectBox1.append(new Option(document.previousColorSelected, document.previousColorSelected), undefined);

selectBox3.append(new Option(document.previousColorSelected, document.previousColorSelected), undefined);

}

}

if(selectBox3.id === ele.id){

removeSelected(selectBox1, selectBox3.value);

removeSelected(selectBox2, selectBox3.value);

if(document.previousColorSelected != ""){

selectBox1.append(new Option(document.previousColorSelected, document.previousColorSelected), undefined);

selectBox2.append(new Option(document.previousColorSelected, document.previousColorSelected), undefined);

}

}

}

function show\_inputs\_dependent(ele){

let independ1 = document.getElementById("independent1");

let independ2 = document.getElementById("independent2");

let independ3 = document.getElementById("independent3");

const r\_label = document.getElementById('repetitions\_label');

const r = document.getElementById('repetitions');

const dim\_y = document.getElementById('dim\_y');

const dim\_y\_label = document.getElementById('dim\_y\_label');

const dim\_x = document.getElementById('dim\_x');

const dim\_x\_label = document.getElementById('dim\_x\_label');

const dim\_xy = document.getElementById('dim\_Y\_X');

const dim\_xy\_label = document.getElementById('dim\_Y\_X\_label');

let depErr1 = document.getElementById("depErr1"); //Nothing entered

let depErr4 = document.getElementById("depErr4"); //Input is too small

let depErr5 = document.getElementById("depErr5"); //Input is too big

let depErr8 = document.getElementById("depErr8"); //Input is too big for dimensions

let depErr6 = document.getElementById("depErr6"); //Input smaller than previous input

var ind = document.querySelector('input[name="independent"]:checked').value;

let listOfItems = document.getElementById("dependentList");

hideMessage(depErr1);

hideMessage(depErr4);

hideMessage(depErr5);

hideMessage(depErr8);

hideMessage(depErr6);

indValue = ind;

globalListOfItems = [];

listOfItems.innerHTML = "";

if (indValue === "1") {

checkAll[1] = 0;

checkAll[2] = 1;

checkAll[3] = 1;

hideMessage(document.getElementById("dimYErr1"));

hideMessage(document.getElementById("dimYErr2"));

hideMessage(document.getElementById("dimYErr3"));

hideMessage(document.getElementById("dimY\_XErr1"));

hideMessage(document.getElementById("dimY\_XErr1"));

hideMessage(document.getElementById("dimY\_XErr1"));

}

if (indValue === "2") {

checkAll[1] = 0;

checkAll[2] = 0;

checkAll[3] = 1;

hideMessage(document.getElementById("dimY\_XErr1"));

hideMessage(document.getElementById("dimY\_XErr1"));

hideMessage(document.getElementById("dimY\_XErr1"));

}

if (indValue === "3") {

checkAll[1] = 1;

checkAll[2] = 1;

checkAll[3] = 0;

hideMessage(document.getElementById("dimYErr1"));

hideMessage(document.getElementById("dimYErr2"));

hideMessage(document.getElementById("dimYErr3"));

hideMessage(document.getElementById("repErr1"));

hideMessage(document.getElementById("repErr2"));

hideMessage(document.getElementById("repErr3"));

}

//let err = document.getElementById("depErr7");

//err.innerHTML = checkAll;

if(independ1.id === ele.id){

r.required = true;

dim\_xy.required = false;

dim\_xy.value = "";

dim\_y.required = false;

dim\_y.value = "";

r\_label.style.display = 'flex';

r.style.display = 'flex';

dim\_xy.style.display = 'none';

dim\_xy\_label.style.display = 'none';

dim\_y.style.display = 'none';

dim\_y\_label.style.display = 'none';

}

if(independ2.id === ele.id){

r.required = true;

dim\_y.required = true;

dim\_xy.required = false;

dim\_xy.value = "";

r\_label.style.display = 'flex';

r.style.display = 'flex';

dim\_y.style.display = 'flex';

dim\_y\_label.style.display = 'flex';

dim\_xy.style.display = 'none';

dim\_xy\_label.style.display = 'none';

}

if(independ3.id === ele.id){

r.required = false;

r.value = "";

dim\_y.required = false;

dim\_y.value = "";

dim\_xy.required = true;

dim\_xy.style.display = 'flex';

dim\_xy\_label.style.display = 'flex';

r.style.display = 'none';

r\_label.style.display = 'none';

dim\_y.style.display = 'none';

dim\_y\_label.style.display = 'none';

}

}

function removeSelected(selectBox,item) {

for (let i = 0; selectBox.options.length > i; i++) {

if(selectBox.options[i].value === item) selectBox.remove(i);

}

}

function removeAllExcludingSelected(selectBox) {

for (let i = 0; selectBox.options.length > i; i++) {

if(!selectBox.options[i].selected) selectBox.remove(i);

}

}

function arraysEqual(a, b) {

if (a === b) return true;

if (a === null || b === null) return false;

if (a.length !== b.length) return false;

for (let i = 0; i < a.length; ++i) {

if (a[i] !== b[i]) return false;

}

return true;

}

function showMessage(id) {

id.classList.remove("hide");

id.classList.add("show");

}

function hideMessage(id) {

id.classList.remove("show");

id.classList.add("hide");

}

function change() {

//let err = document.getElementById("depErr7");

//err.innerHTML = checkAll;

let depErr2 = document.getElementById("depErr2"); //Too few items

let depErr3 = document.getElementById("depErr3"); //Too many items

let depErr6 = document.getElementById("depErr6"); //Input smaller than previous input

let input = document.getElementById("numberInput");

let inputValueAsNumber = parseInt(input.value);

if (inputValueAsNumber >= parseInt(globalListOfItems[globalListOfItems.length - 1])){

hideMessage(depErr6);

input.required = false;

}

if(globalListOfItems.length < 4) {

showMessage(depErr2);

input.required = true;

}

else {

hideMessage(depErr2);

input.required = false;

}

if(globalListOfItems.length > 12) {

showMessage(depErr3);

input.required = true;

}

else {

hideMessage(depErr3);

input.required = false;

}

let checkThis = [1, 1, 1, 1];

if (globalListOfItems.length < 3 || globalListOfItems.length > 12) {

checkAll[0] = 0;

input.required = true;

} else {

checkAll[0] = 1;

input.required = false;

}

//document.getElementById("depErr7").textContent = checkAll;

console.log(checkAll);

if(arraysEqual(checkAll, checkThis)) {

document.getElementById("submit").disabled = false;

return true;

}

//else {

//document.getElementById("submit").disabled = true;

// return false;

//}

return false;

}

function register()

{

//document.getElementById("submit").disabled = false;

//document.getElementById("dim\_y").oninput = checkDimY;

//document.getElementById("repetitions").oninput = checkRep;

//document.getElementById("dim\_Y\_X").oninput = checkDimXY;

document.getElementById("repetitions").addEventListener("input", checkRep);

document.getElementById("dim\_y").addEventListener("input", checkDimY);

document.getElementById("dim\_Y\_X").addEventListener("input", checkDimXY);

document.getElementById("myForm").oninput = change;

}

function checkDimXY() {

let input = document.getElementById("dim\_Y\_X");

let err1 = document.getElementById("dimY\_XErr1");

let err2 = document.getElementById("dimY\_XErr2");

let err3 = document.getElementById("dimY\_XErr3");

if (input) {

if (input.value === "") {

showMessage(err1);

} else {

hideMessage(err1);

}

if (input.value < 1) {

showMessage(err2);

} else {

hideMessage(err2);

}

if (input.value > 40) {

showMessage(err3);

} else {

hideMessage(err3);

}

if (input.value > 0 && input.value < 40) {

checkAll[3] = 1;

} else {

checkAll[3] = 0;

}

}

}

function checkDimY() {

let input = document.getElementById("dim\_y");

let err1 = document.getElementById("dimYErr1");

let err2 = document.getElementById("dimYErr2");

let err3 = document.getElementById("dimYErr3");

if (input) {

if (input.value === "") {

showMessage(err1);

} else {

hideMessage(err1);

}

if (input.value < 1) {

showMessage(err2);

} else {

hideMessage(err2);

}

if (input.value > 40) {

showMessage(err3);

} else {

hideMessage(err3);

}

if (input.value > 0 && input.value < 40) {

checkAll[2] = 1;

} else {

checkAll[2] = 0;

}

}

}

function checkRep() {

let input = document.getElementById("repetitions");

let err1 = document.getElementById("repErr1");

let err2 = document.getElementById("repErr2");

let err3 = document.getElementById("repErr3");

if (input) {

if (input.value === "") {

showMessage(err1);

} else {

hideMessage(err1);

}

if (input.value < 1) {

showMessage(err2);

} else {

hideMessage(err2);

}

if (input.value > 99) {

showMessage(err3);

} else {

hideMessage(err3);

}

if (input.value > 0 && input.value !== "" && input.value < 99) {

checkAll[1] = 1;

} else {

checkAll[1] = 0;

}

}

}

//button is always active only deactivate when all values are not correct.

function submitCheck() {

if(indValue === "1")

{

checkRep();

}

if(indValue === "2")

{

checkDimY();

checkRep();

}

if(indValue === "3")

{

checkDimXY();

}

//checkDimX();

//checkDimY();

//BYPASSING DIMX AND DIMY:

//checkAll[0] = 1;

//checkAll[1] = 1;

//checkRep();

let did\_enter\_numbers\_list = change();

let checkThis = [1, 1, 1, 1];

console.log(did\_enter\_numbers\_list);

if(arraysEqual(checkAll, checkThis) && did\_enter\_numbers\_list) {

console.log(did\_enter\_numbers\_list);

return true;

}

// do what you want

return false;

}

function setDimY\_X(){

let input\_y = document.getElementById("dim\_y");

let input\_x = document.getElementById("dim\_x");

let input\_xy = document.getElementById("dim\_Y\_X");

input\_y.value = input\_xy.value;

input\_x.value = input\_xy.value;

}

function getRandomInt(max) {

return Math.floor(Math.random() \* max)+1;

}

function three\_color\_picker(colorItems) {

//function for checking for empty user input boxes

function checkBoxLength (stringLength) {

return stringLength.length != "";

}

//function to remove empty user input boxes

let options = optionsArray;

let valueChoice = new Array();

for (let element in colorItems) {

//randomise number to be used to select array index position

let optionsRan = 0;

let i = 0;

do {

optionsRan = Math.floor(Math.random() \* options.length)+1;

if(i++ > 1000) return;

} while (valueChoice.includes(optionsRan));

valueChoice.push(optionsRan);

//alert the user to the option that the function has selected

//console.log("Value selected", optionsRan);

//console.log("element: ", element);

if (optionsRan === 0) {

document.getElementById(element).selectedIndex = 0;

} else if (optionsRan === 1) {

document.getElementById(element).selectedIndex = 1;

} else if (optionsRan === 2) {

document.getElementById(element).selectedIndex = 2;

} else if (optionsRan === 3) {

document.getElementById(element).selectedIndex = 3;

} else if (optionsRan === 4) {

document.getElementById(element).selectedIndex = 4;

} else if (optionsRan === 5) {

document.getElementById(element).selectedIndex = 5;

} else if (optionsRan === 6) {

document.getElementById(element).selectedIndex = 6;

} else if (optionsRan === 7) {

document.getElementById(element).selectedIndex = 7;

} else if (optionsRan === 8) {

document.getElementById(element).selectedIndex = 8;

} else if (optionsRan === 9) {

document.getElementById(element).selectedIndex = 9;

} else {

alert("D'oh!");

}

}

}

function clearSelectList(){

let listOfItems = document.getElementById("dependentList");

//clear select box:

let i, L = listOfItems.options.length - 1;

for(i = L; i >= 0; i--) {

listOfItems.remove(i);

}

globalListOfItems = []; //reset to null.

}

function randomize\_inputs(){

//clear all list options

clearSelectList();

//set random value for independent:

let independ1 = document.getElementById("independent1");

let independ2 = document.getElementById("independent2");

let independ3 = document.getElementById("independent3");

switch (getRandomInt(3)){

case 1:{

independ1.checked = true;

show\_inputs\_dependent(independ1);

independ2.checked = false;

independ3.checked = false;

let reps = document.getElementById("repetitions");

reps.value = getRandomInt(99);

//numbers in list:

let input = document.getElementById("numberInput");

let dim\_value\_list = getRandomInt(5);

input.value = dim\_value\_list;

addItemToList();

input.value = dim\_value\_list\*2;

addItemToList();

input.value = dim\_value\_list\*2\*2;

addItemToList();

input.value = dim\_value\_list\*2\*2\*2;

addItemToList();

input.value = "";

break;

}

case 2:{

independ2.checked = true;

show\_inputs\_dependent(independ2);

independ1.checked = false;

independ3.checked = false;

//set repetitions

let reps = document.getElementById("repetitions");

reps.value = getRandomInt(99);

//numbers in list:

let input = document.getElementById("numberInput");

let dim\_value\_list = getRandomInt(5);

input.value = dim\_value\_list;

addItemToList();

input.value = dim\_value\_list\*2;

addItemToList();

input.value = dim\_value\_list\*2\*2;

addItemToList();

input.value = dim\_value\_list\*2\*2\*2;

addItemToList();

input.value = "";

//set random value for dimension:

let input\_y = document.getElementById("dim\_y");

input\_y.value = getRandomInt(40);

break;

}

case 3:{

independ3.checked = true;

show\_inputs\_dependent(independ3);

independ2.checked = false;

independ1.checked = false;

//numbers in list (repetitions):

let input = document.getElementById("numberInput");

let rep\_value\_list = getRandomInt(10);

input.value = rep\_value\_list;

addItemToList();

input.value = rep\_value\_list\*2;

addItemToList();

input.value = rep\_value\_list\*2\*2;

addItemToList();

input.value = rep\_value\_list\*2\*2\*2;

addItemToList();

input.value = "";

//set random value for dimension:

let input\_xy = document.getElementById("dim\_Y\_X");

input\_xy.value = getRandomInt(40);

break;

}

}

//\*

//color choice:

let color1 = document.getElementById("color1");

let color2 = document.getElementById("color2");

let color3 = document.getElementById("color3");

three\_color\_picker({color1,color2,color3});

//\*/

//select termination option:

let termItem1 = document.getElementById("termItem1");

let termItem2 = document.getElementById("termItem2");

let termItem3 = document.getElementById("termItem3");

let termChoice = getRandomInt(3);

if(termChoice === 1){

termItem1.checked = true;

} else if(termChoice === 2){

termItem2.checked = true;

} else {

termItem3.checked = true;

}

//select music option:

let musicItem1 = document.getElementById("song1");

let musicItem2 = document.getElementById("song2");

let musicItem3 = document.getElementById("song3");

let musicItem4 = document.getElementById("song4");

let musicItem5 = document.getElementById("song5");

let musicItem6 = document.getElementById("song6");

let musicChoice = getRandomInt(6);

if(musicChoice === 1){

musicItem1.checked = true;

} else if(musicChoice === 2){

musicItem2.checked = true;

} else if(musicChoice === 3){

musicItem3.checked = true;

} else if(musicChoice === 4){

musicItem4.checked = true;

} else if(musicChoice === 5){

musicItem5.checked = true;

}else {

musicItem6.checked = true;

}

}

## Styles.css:

.error {  
 color: red;  
}  
  
.neutral {  
 background-color: white;  
}  
  
.false {  
 background-color: pink;  
}  
  
.true {  
 background-color: lightgreen;  
}

## Run.css:

body {

background: white;

}

.slider {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%,-50%);

width: 500px;

height: 60px;

padding: 30px;

padding-left: 40px;

background: #fcfcfc;

border-radius: 20px;

display: flex;

align-items: center;

box-shadow: 0px 15px 40px #7E6D5766;

}

.slider label {

font-size: 24px;

font-weight: 400;

font-family: Open Sans;

padding-left: 10px;

color: black;

}

.slider input[type="range"] {

width: 210px;

height: 2px;

background: black;

border: none;

outline: none;

}

# Appendix B. User Manual

When the user first enters the website, they should see two buttons at the top. The first button (Run Experiment) should be pressed after filling out the form on the rest of the page. The second button (Fill Random Experiment) can be pressed to fill the form out to run the experiment with random values.

If the user doesn’t want random values, they must fill out the form to run the experiment with their chosen variable values. To fill out the form, first select a variable for the independent variable. Then input at least 4 values into the list in the middle of the form. To fill the list, the user must input a number in the field below the list and press the *Add Number* button. Then after filling this list with independent variable values, the list for dependent variable values will appear to the right. This field must be filled out also.

After filling those fields, the colors, stopping criteria, and song choice can be filled out below. When all fields are filled out, the user should press the *Run Experiment* button.

The next screen will be the running experiment. During the run of the experiment the speed can be sped-up or slowed down using the slider control on the left.

After the experiment is finished the variable to be calculated is shown at the bottom. The user should choose one of these variables to calculate and be shown on the graph. After choosing this variable the user should press the *Calculate and Display Chosen Value* button which will display a graph.

# Appendix C. Test Plan

Canvas Color Mixer Test Plan

Team Darkest Error

Members:

Dakota Stephens <djsh2z@umsl.edu>

Mindy Zheng <mzhfc@umsl.edu>

Nathan Pimentel <nathanjpimentel@umsl.edu>

Anthony Pardo <ajpcnc@umsl.edu>

Frankie Mccaa <fm6np@umsl.edu>

Thomas Citrowske <tjcnc2@umsl.edu>

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Changes** | **Revised By** |
| 1.0 | 11/17/2023 | Initial Draft | Mindy Zheng |
| 2.0 | 12/9/2023 | Testing rounds | Mindy Zheng |
| 3.0 | 12/13/2023 | Simple Test | Dakota Stephens |

## Introduction

Our project is an immersive interactive website designed to engage middle-school girls in the diverse fields of STEM. Our objective is to provide an enjoyable, educational, and hands-on way to explore fundamental concepts like randomness, statistics, and coding through a simulated, random paint-dropping process on a virtual canvas.

The user will answer multiple prompts, specifying the experiment’s input parameters such as: the dimensions of the canvas(X and Y), their color choices, and the stopping criterion for the experiment. The website will provide clear instructions for the user, as well as limits for each input to smoothly guide the user through the simulation.

Once the user sets the parameters, the simulation will begin. Users will also be given an option to adjust the speed of the simulation as it progresses. Upon completion, the users will proceed to next phase of the website - the experimental section.

Here, the user will be prompted again to select an independent variable from three option: D (representing a single number for square canvases), X (with Y held constant), or R (the number of experiment repetitions), along with its respective input values and other fixed variables. Error handling will be implemented at every input prompt to ensure appropriate values are being selected.

During this simulation, several values will be calculated: A (the total number of paint drops of each color - A1, A2, A3), B (the maximum number of paint drops on any square), and C (the average number of paint drops over all squares). After running the experiments, a comprehensive table of all the calculated values will be displayed, giving the user a detailed overview of the simulation and experiment results.

This website aims to inspire and spark an interest within young girls as they begin to explore the different fields of STEM. By integrating a unique blend of art and mathematics, we hope to foster a deeper understanding and curiosity of randomness, computational thinking, and statistical skills.

## Testing Plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item No.** | **Feature Tested** | **Test Plan/Expected Behavior** | **Comments** | **Test Date** | **Result (P/F)** | **Tested By** |
| 2.1 | User input functionality | Test the user’s ability to input the parameters for the painting simulation - dimensions, color choices, stopping criterion; it should accept valid inputs and display appropriate error messages | Does not display appropriate and detailed error messages. No errors were displayed when wrong inputs were given. Dimension 1 - 40. Repetitions 1 - 99. | 11/16/23 | F | Dakota Stephens |
| 2.2 | Animation functionality | Test the animation’s ability to explain the purpose of the website. The animation should provide the user with a visualization and inform the user. | Absent as of now | 11/17/23 | F | Mindy Zheng |
| 2.3 | Simulation Functionality | Test the painting simulation with various parameters. The simulation should run correctly and reflect the chosen parameters | Created grid works/visual canvas works, reflects the chosen parameters of single dimension independent [1,2,3,4], repetition value of 2, colors (blue, black, green), and term setting of first time any square painted twice. | 11/17/23 | T | Mindy Zheng |
| 2.4 | Experiment functionality | Test the experiments portion of the website, including choosing an independent variable and running the experiment. | Not completed as of yet | 11/17/23 | F | Mindy Zheng |
| 2.5 | Detailed data functionality | Test the ability to generate a table and graph that accurately represents the experiment’s data | Not completed as of yet | 11/17/23 | F | Mindy Zheng |
| 2.6 | Random Fill Experiment | Test the ability to randomly fill values on open fields and run experiment | Random values fill successfully but was not able to run the experiment when clicking the run experiment button. The values were filled as follows: Independent was single dimension [3,6,9,12], repetition value of 34, colors (yellow, black, blue), and term setting of first time any square painted for the third time. | 12/6/23 | F | Mindy Zheng |
| 2.7 | Random Fill Experiment | Test the ability to randomly fill values on input fields and run | When pressing the “Fill Random Experiment” button it filled all values within range and worked thereafter when I clicked run experiment. Functioning correctly. The values were filled as follows: Independent was the number of repetitions [2,4,6,8], Square Dimension value of 21, colors (black, blue, purple), and term setting of the first time any square painted for the third time. | 12/9/23 | T | Mindy Zheng |
| 2.8 | User input Functionality | Test the user’s ability to input the parameters for the painting simulation - dimensions, color choices, stopping criterion; it should accept valid inputs and display appropriate error messages. | Correctly displays errors and accepts inputs. Attempted to put values greater than 99 and less than 1 (which is not allowed) in both independent field and fixed input fields. Also attempted the same with dimension for both fixed and independent. Note, this value range is from 1 to 40.  I determined that no error was being displayed when entering the value 0. | 12/9/23 | F | Mindy Zheng |
| 2.9 | Speed Functionality | Test user’s choice to adjust speed when running simulation | Speed adjustment works change value from 1 to 10. Then from 10 to 0.25 | 12/9/23 | T | Mindy Zheng |
| 3.1 | Animation functionality | Test the animation’s ability to explain the purpose of the website. The animation should provide the user with a visualization and inform the user. | After navigating to the website. Change pages by clicking the usage tab. Once there I watched the animation for web page playout. It was informative and worked correctly. No further information needed. | 12/13/23 | T | Dakota Stephens |

\*Testing will be conducted by Mindy Zheng on Windows 11 and Dakota Stephens on Windows 10, with each feature being evaluated in its respective environment.