Unit 1

Methods of Iterating week 3

MA GCD | Y1

Prae Sunsermsook

29 January 2026

The project’s inquiry

```
<!doctype html>
<html lang="en" itemscope itemtype='http://schema.org/CollectionPage'>
    <head>
        <meta name="generator" content="Hugo 0.46" />
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrin
k-to-fit=no">

        <meta property="description" content='Kids Ga[illegible]
e fun to play. Play exciting games for the ages of 2[illegible]
ree! A safe place to play the best free games! Games for boys and girls that can
be played on any device (mobile, tablets and computers).' />

        <meta name="author" content="Marcelo Canina">
        <link rel="icon" href="https://[illegible]
        <link rel="canonical" hre[illegible]

        <title>Play Fun, Free Games for Bo[illegible]
ds</title>

        <link href="https://kidsgames.world/css/style.css" rel="stylesheet" type=
"text/css">

        <link href="https://kidsgames.world/css/custom.css" rel="stylesheet" type
="text/css">

        <meta property="og:title" content="Play Fun[illegible]
s Online" />
<meta property="og:description" content="Kids Games has family games that are fun
 to play. Play exciting games for the ages of 2 to 6 years, online and for free!
A safe place to play the best free games! Games for boys and girls that can be pl
ayed on any device (mobile, tablets and computers)." />
<meta property="og:type" content="website" />
<meta property="og:url" content="https://kidsgames.world/" />


<meta property="og:updated_time" content="2017-11-19T16:06:44-03:00"/>
```

```
<!doctype html><html lang=en itemscope itemtype=http://schema.org/CollectionPage
<head><meta name=generator content="Hugo 0.46"><meta charset=utf-8><meta name=vi
wport content="width=device-width,initial-scale=1,shrink-to-fit=no"><meta proper
y=description content="Kids Games has family games that are fun to play. Play ex
iting games for the ages of 2 to 6 years, online and for free! A safe place to p
ay the best free games! Games for boys and girls that can be played on any devic
 (mobile, tablets and computers)."><meta name=author content="Marcelo Canina"><l
nk rel=icon href=https://kidsgames.world/favicon.ico><link rel=canonical href=ht
ps://kidsgames.world/><title>Play Fun, Free Games for Boys and Girls Online | Fr
[illegible] href=https://kidsgames.world/css/style.css rel=sty
esheet><link href=https://kidsgames.world/css/custom.css rel=stylesheet><meta pr
perty=og:title content="Play Fun, Free Games for Boys and Girls Online"><meta pr
perty=og:description content="Kids Games has family games that are fun to play.
lay exciting games for the ages of 2 to 6 years, online and for free! A safe pla
[illegible] nd girls that can be played on an
[illegible] property=og:type content=websit
[illegible] games.world/><meta property=og:updated
time content=2017-11-19T16:06:44-03:00><meta property=og:site_name content="Free
Games For Kids"><meta itemprop=name content="Play Fun, Free Games for Boys and G
[illegible] content="Kids Games has family games that
[illegible] r the ages of 2 to 6 years, online and fo
 free! A safe place to play the best free games! Games for boys and girls that c
n be played on any device (mobile, tablets and computers)."><meta name=twitter:c
rd content=summary><meta name=twitter:title content="Play Fun, Free Games for Bo
s and Girls Online"><meta name=twitter:description content="Kids Games has famil
 games that are fun to play. Play exciting games for the ages of 2 to 6 years, o
[illegible] ee! A safe place to play the best free games! Games for boys and
irls that can be played on any device (mobile, tablets and computers)."><meta na
[illegible] mesforkids><meta name=google-site-verification cont
[illegible] wjxzEd54eqqNFRK07U><meta name=msvalidate.01 content=
53BB779294406F41BD202E3E24FFE55><meta name=yandex-verification content=94dfdda77
c13d7b><script async src=//pagead2.googlesyndication.com/pagead/js/adsbygoogle.j
></script><script>(adsbygoogle=window.adsbygoogle||[]).push({google_ad_client:"c
-pub-1234567890",enable_page_level_ads:true});</script><link rel=alternate type=
pplication/rss+xml href=https://kidsgames.world/index.xml title="Free Games For
ids"><script>var doNotTrack=false;if(!doNotTrack){window.ga=window.ga||function(
{(ga.q=ga.q||[]).push(arguments)};ga.l=+new Date;ga('create','UA-31899481-6','au
o');ga('send','pageview');}</script><script async src=https://www.google-analyti
s.com/analytics.js></script></head><body><nav class="navbar navbar-expand-lg nav
ar-light bg-light"><a class=navbar-brand href=https://kidsgames.world/>🏠 Kids Ga
```

Interrogating the tension of

Code as a bilingual language for (human, `machine`);

Hack the tool by
using code to visualise itself

The promise of p5.js

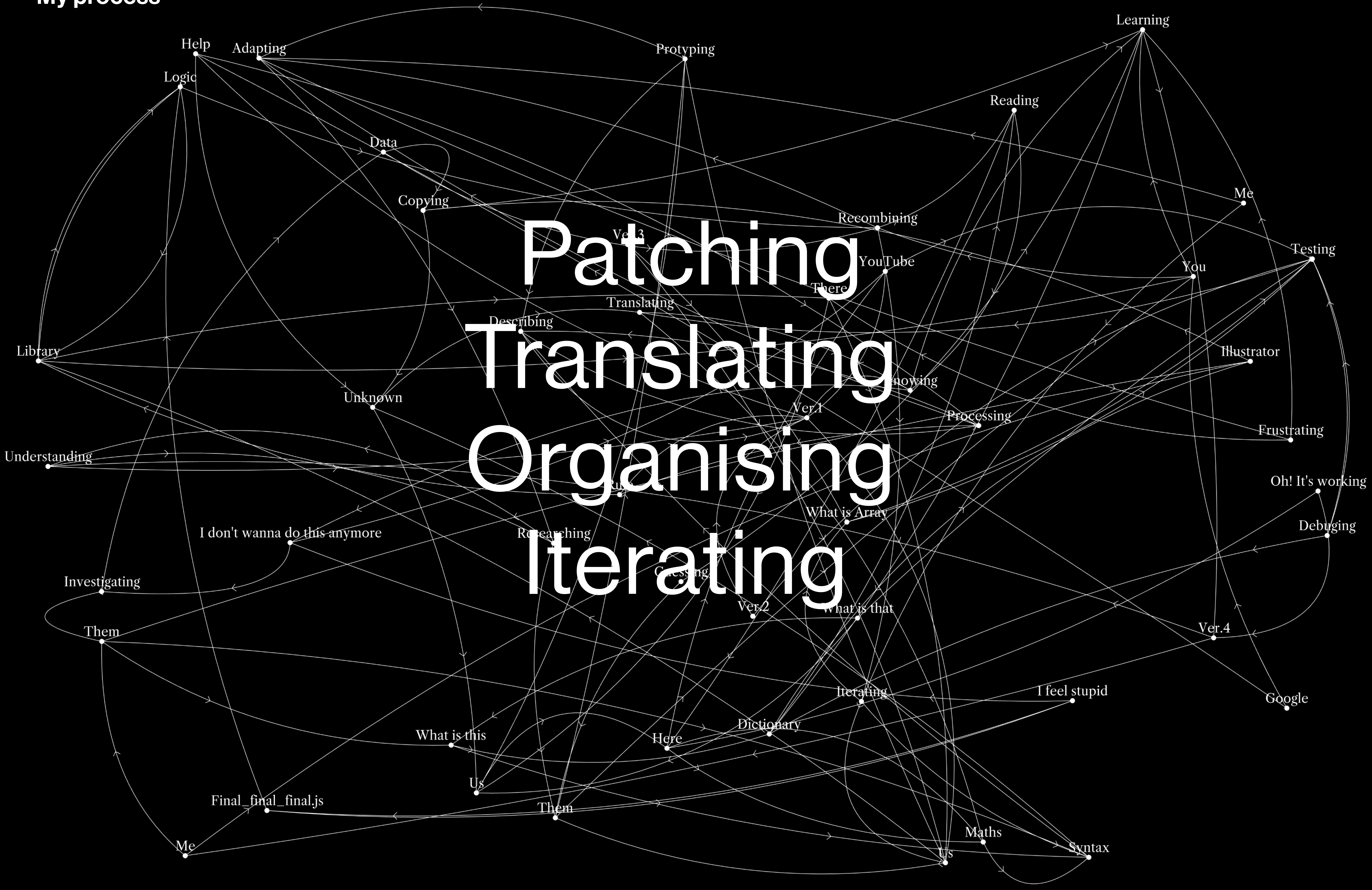
I revisited the landing page's brand message. It claims that p5.js is a friendly tool for learning to code. However, from my experience, I don’t feel that way at all.

p5.js
Reference
Tutorials
Examples
Contribute
Community
About
English
Accessibility

p5.js is a friendly tool for learning to code and make art. It is a free and open-source JavaScript library built by an inclusive, nurturing community. p5.js welcomes artists, designers, beginners, educators, and anyone else!

p5.js will not add any new features except those that increase access.

My process



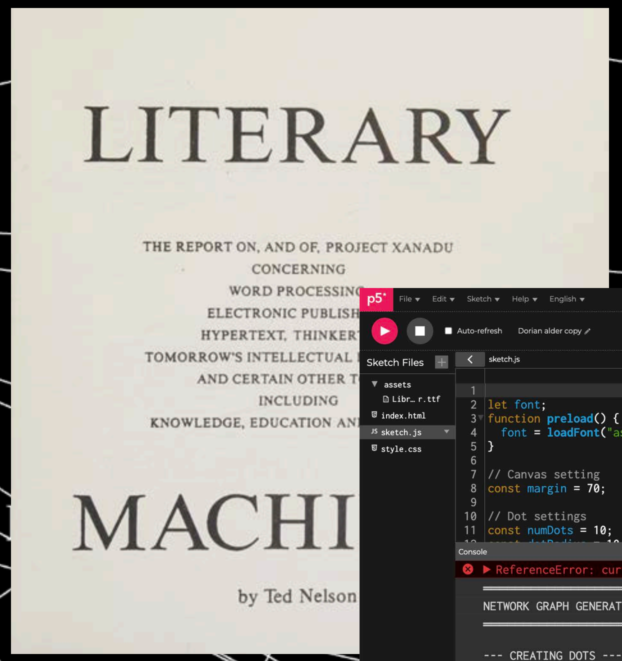
Patching
Translating
Organising
Iterating

My process

Help
Adapting
Logic
Protyping
Learning
Copying
Translating
Describing
There
Knowing
Testing
Library
Unknown
Frustrating
Understa
Quark syzygy
Chalk singularity
Complete neighborhood
Deserted psychology
Early chameleon
Bramble stranger
Calm cross
Road ground
Blue twister
Lake swamp
Oh! It's working
lon't wanna do this anymore
What is Array
What is that
Investigating
Iterating
I feel stupid
Them
Maths
Us
Syntax
Struggling

BLACK GOOEY UNIVERSE
American Artist

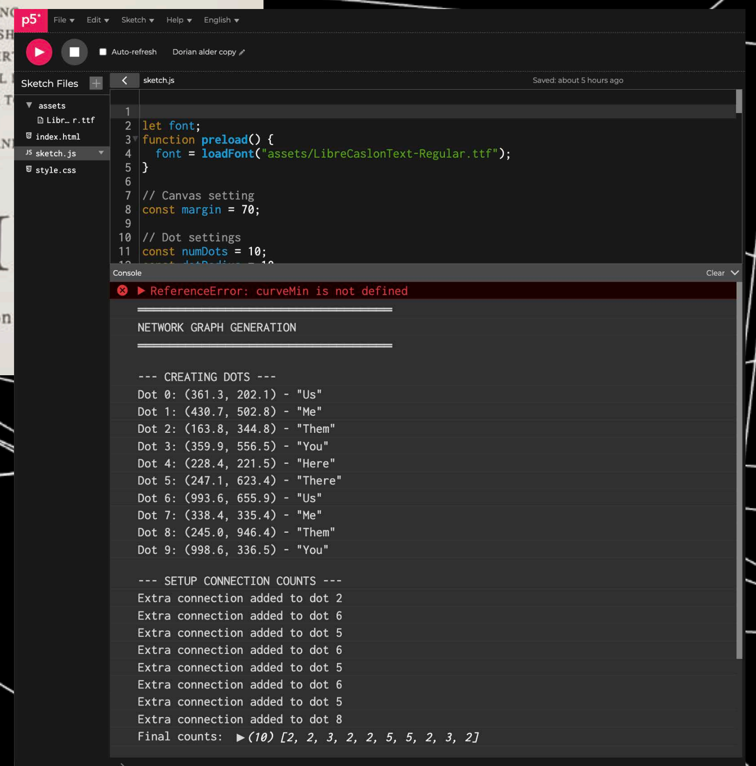
"This is called a graphical user interface—GUI or gooey—where they come up with these names. The battle to bring gooeys to PCs and make them more user-friendly took ten years and is a helluva story—that is what this program is about. It's also about how Bill Gates ended up master of the gooey universe and a gazillionaire. I never said it was a fairy story." – Triumph of the Nerds, PBS (1996)


LITERARY

THE REPORT ON, AND OF, PROJECT XANADU CONCERNING WORD PROCESSING ELECTRONIC PUBLISH HYPERTEXT, THINKER TOMORROW'S INTELLECTUAL AND CERTAIN OTHER T INCLUDING KNOWLEDGE, EDUCATION AN

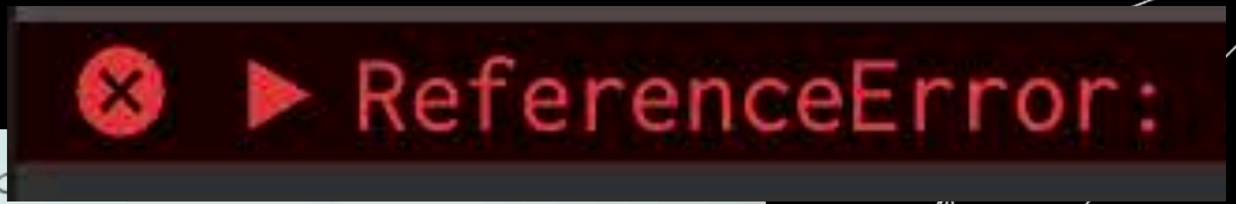
MACHI

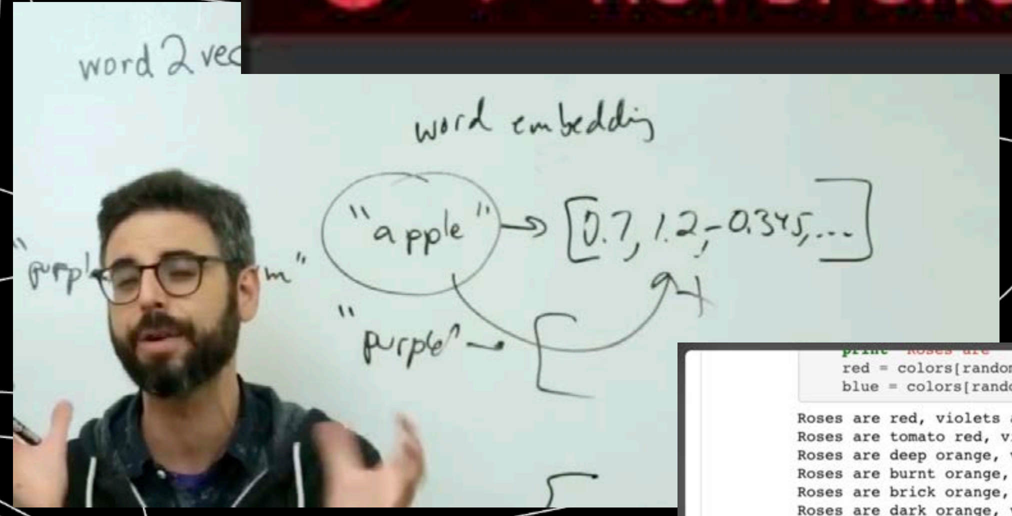
by Ted Nelson

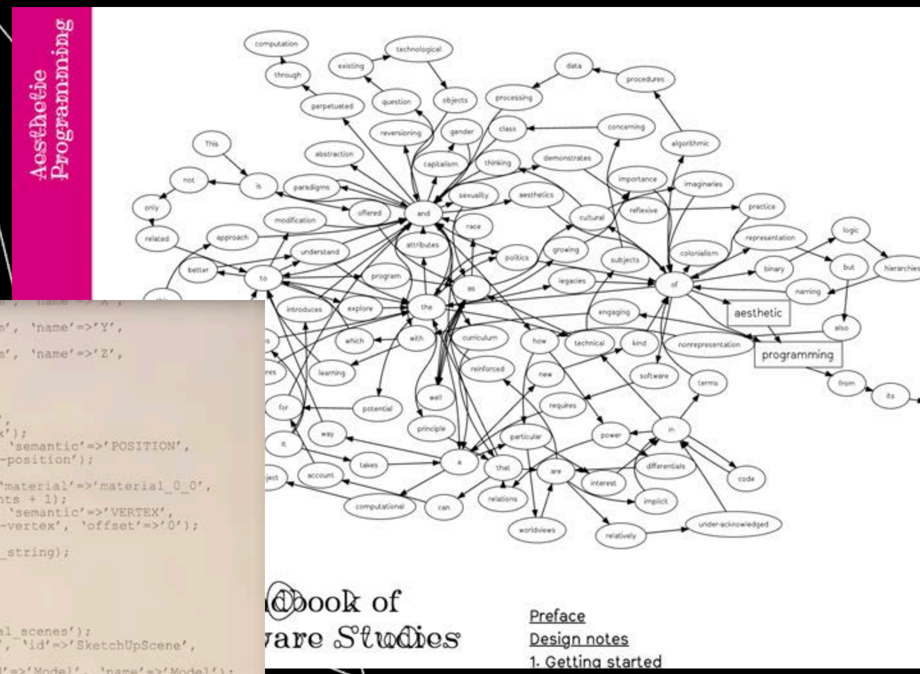


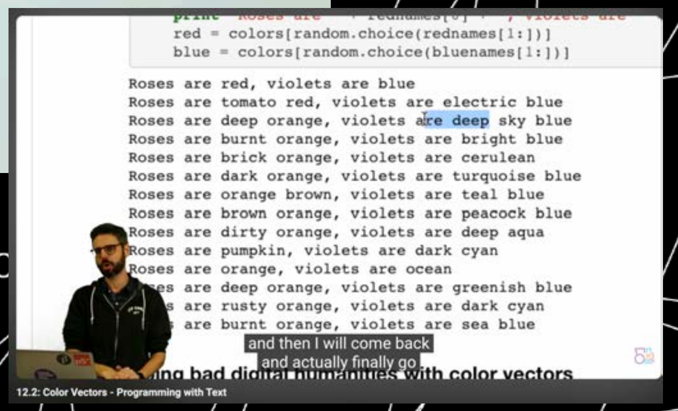
```
let font;
function preload() {
  font = loadFont("assets/LibreCaslonText-Regular.ttf");
}

// Canvas setting
const margin = 70;

// Dot settings
const numDots = 10;
```

```
▶ ReferenceError: curveMin is not defined
═══════════════════════════════════════
NETWORK GRAPH GENERATION
═══════════════════════════════════════

--- CREATING DOTS ---
Dot 0: (361.3, 202.1) - "Us"
Dot 1: (430.7, 502.8) - "Me"
Dot 2: (163.8, 344.8) - "Them"
Dot 3: (359.9, 556.5) - "You"
Dot 4: (228.4, 221.5) - "Here"
Dot 5: (247.1, 623.4) - "There"
Dot 6: (993.6, 655.9) - "Us"
Dot 7: (338.4, 335.4) - "Me"
Dot 8: (245.0, 946.4) - "Them"
Dot 9: (998.6, 336.5) - "You"

--- SETUP CONNECTION COUNTS ---
Extra connection added to dot 2
Extra connection added to dot 6
Extra connection added to dot 5
Extra connection added to dot 6
Extra connection added to dot 5
Extra connection added to dot 6
Extra connection added to dot 5
Extra connection added to dot 8
Final counts:  ▶(10) [2, 2, 3, 2, 2, 5, 5, 2, 3, 2]
```


```
✖ ▶ ReferenceError:
```





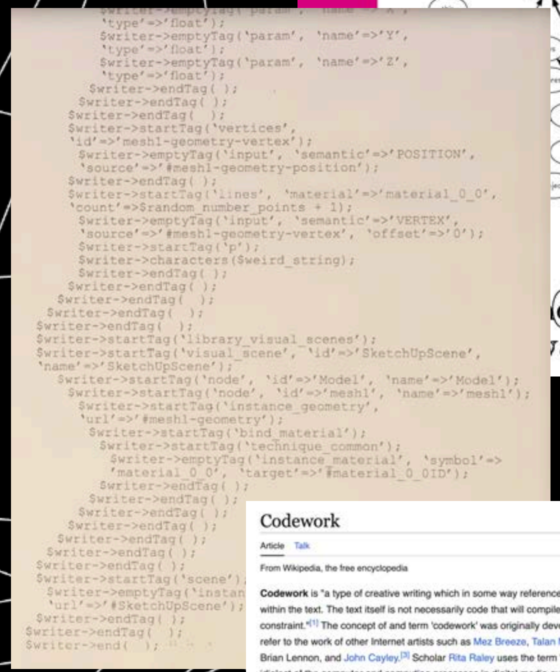

```
red = colors[random.choice(rednames[1:])]
blue = colors[random.choice(bluenames[1:])]
```

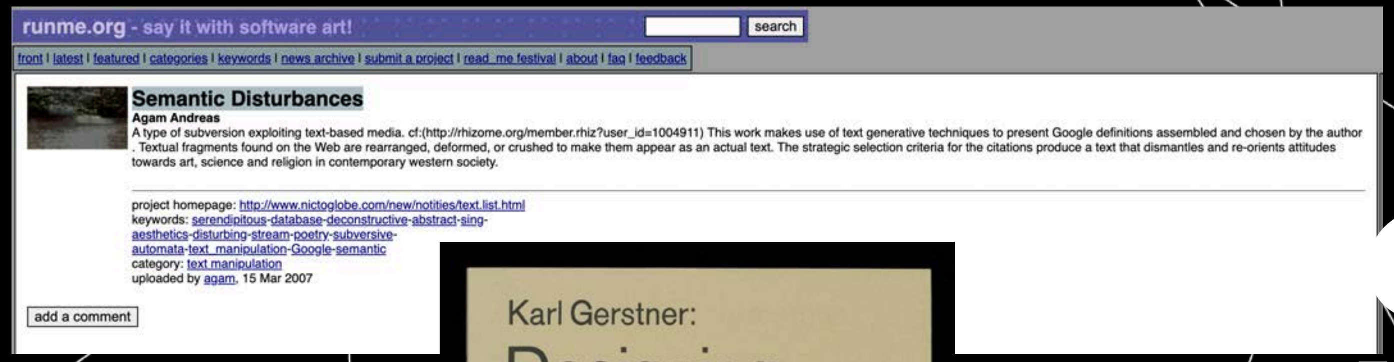
```
Roses are red, violets are blue
Roses are tomato red, violets are electric blue
Roses are deep orange, violets are deep sky blue
Roses are burnt orange, violets are bright blue
Roses are brick orange, violets are cerulean
Roses are dark orange, violets are turquoise blue
Roses are orange brown, violets are teal blue
Roses are brown orange, violets are peacock blue
Roses are dirty orange, violets are deep aqua
Roses are pumpkin, violets are dark cyan
Roses are orange, violets are ocean
Roses are deep orange, violets are greenish blue
Roses are rusty orange, violets are dark cyan
Roses are burnt orange, violets are sea blue
```

and then I will come back and actually finally go

doing bad digital humanities with color vectors

12.2: Color Vectors - Programming with Text


```
$writer->startTag('library_visual_scenes');
$writer->startTag('visual_scene', 'id'=>'SketchUpScene',
'name'=>'SketchUpScene');
$writer->startTag('node', 'id'=>'Model', 'name'=>'Model');
$writer->startTag('node', 'id'=>'mesh1', 'name'=>'mesh1');
$writer->startTag('instance_geometry',
'url'=>'#mesh1-geometry');
$writer->startTag('bind_material');
$writer->startTag('technique_common');
$writer->emptyTag('instance_material', 'symbol'=>
'material_0_0', 'target'=>'#material_0_0ID');
$writer->endTag( );
$writer->endTag( );
$writer->endTag( );
$writer->endTag( );
$writer->endTag( );
$writer->endTag( );
$writer->startTag('scene');
$writer->emptyTag('instance_visual_scene',
'url'=>'#SketchUpScene');
$writer->endTag( );
$writer->endTag( );
$writer->end( );
```

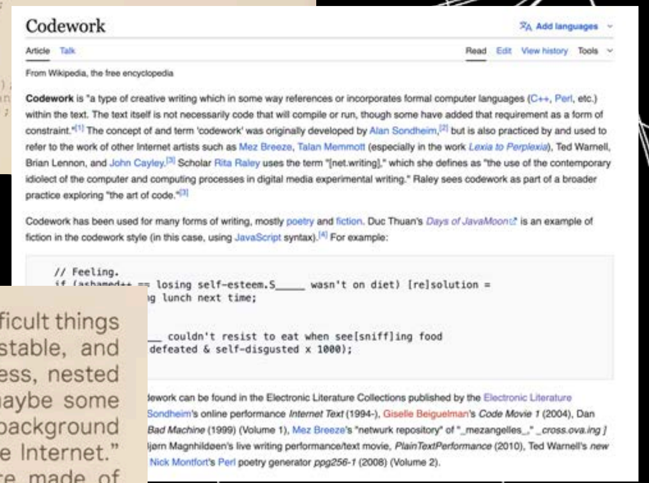

runme.org - say it with software art!

front | latest | featured | categories | keywords | news archive | submit a project | read_me festival | about | faq | feedback

Semantic Disturbances
Agam Andreas
A type of subversion exploiting text-based media. cf:(http://rhizome.org/member.rhiz?user_id=1004911) This work makes use of text generative techniques to present Google definitions assembled and chosen by the author . Textual fragments found on the Web are rearranged, deformed, or crushed to make them appear as an actual text. The strategic selection criteria for the citations produce a text that dismantles and re-orients attitudes towards art, science and religion in contemporary western society.

project homepage: http://www.nictoglobe.com/new/notities/text.list.html
keywords: serendipitous-database-deconstructive-abstract-sing-aesthetics-disturbing-stream-poetry-subversive-automata-text_manipulation-Google-semantic
category: text manipulation
uploaded by agam, 15 Mar 2007

add a comment


Codework

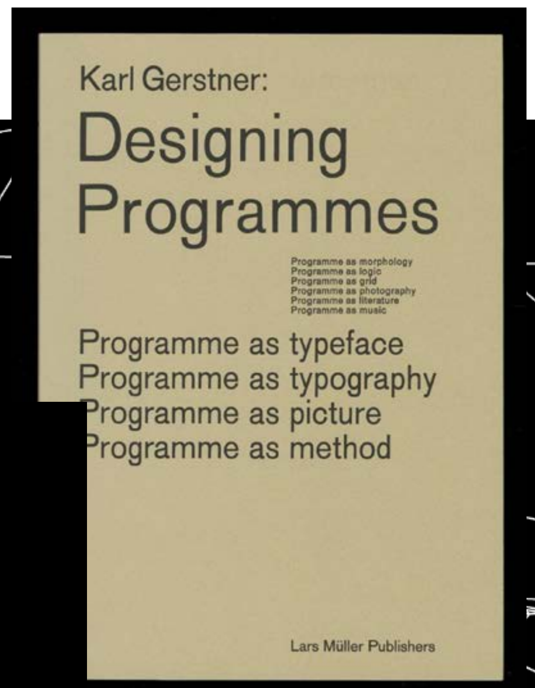
From Wikipedia, the free encyclopedia

Codework is "a type of creative writing which in some way references or incorporates formal computer languages (C++, Perl, etc.) within the text. The text itself is not necessarily code that will compile or run, though some have added that requirement as a form of constraint."[1] The concept of and term 'codework' was originally developed by Alan Sondheim,[2] but is also practiced by and used to refer to the work of other Internet artists such as Mez Breeze, Talan Memmott (especially in the work *Lexia to Perplexia*), Ted Warnell, Brian Lennon, and John Cayley.[3] Scholar Rita Raley uses the term "[net.writing]," which she defines as "the use of the contemporary idiolect of the computer and computing processes in digital media experimental writing." Raley sees codework as part of a broader practice exploring "the art of code."[3]

Codework has been used for many forms of writing, mostly poetry and fiction. Duc Thuan's *Days of JavaMoon* is an example of fiction in the codework style (in this case, using JavaScript syntax).[4] For example:

```
// Feeling.
if (ashamed++ == losing self-esteem.S____ wasn't on diet) [re]solution =
                 ng lunch next time;

                ____ couldn't resist to eat when see[sniff]ing food
                 defeated & self-disgusted x 1000);
```

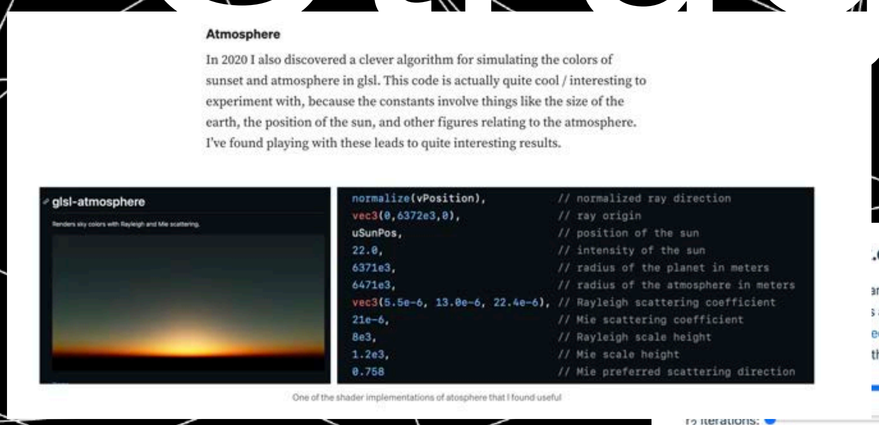
…ework can be found in the Electronic Literature Collections published by the Electronic Literature … Sondheim's online performance *Internet Text* (1994-), Giselle Beiguelman's *Code Movie 1* (2004), Dan … *Bad Machine* (1999) (Volume 1), Mez Breeze's "netwurk repository" of "_mezangelles_," _cross.ova.ing] … Bjørn Magnhildøen's live writing performance/text movie, *PlainTextPerformance* (2010), Ted Warnell's *new* … Nick Montfort's Perl poetry generator *ppg256-1* (2008) (Volume 2).


Karl Gerstner:
Designing Programmes

Programme as morphology
Programme as logic
Programme as grid
Programme as photography
Programme as literature
Programme as music

Programme as typeface
Programme as typography
Programme as picture
Programme as method

Lars Müller Publishers

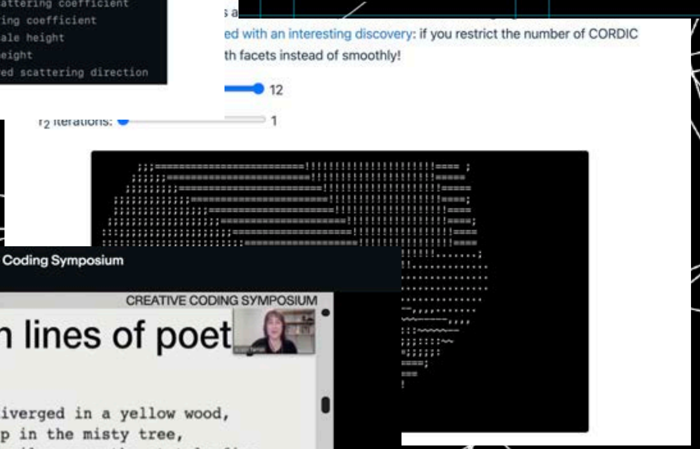

Atmosphere

In 2020 I also discovered a clever algorithm for simulating the colors of sunset and atmosphere in glsl. This code is actually quite cool / interesting to experiment with, because the constants involve things like the size of the earth, the position of the sun, and other figures relating to the atmosphere. I've found playing with these leads to quite interesting results.

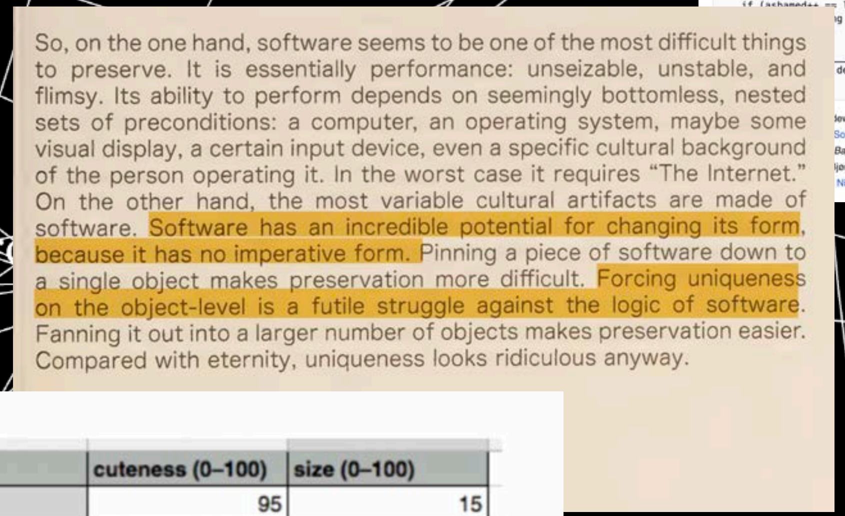
glsl-atmosphere
Renders sky colors with Rayleigh and Mie scattering.

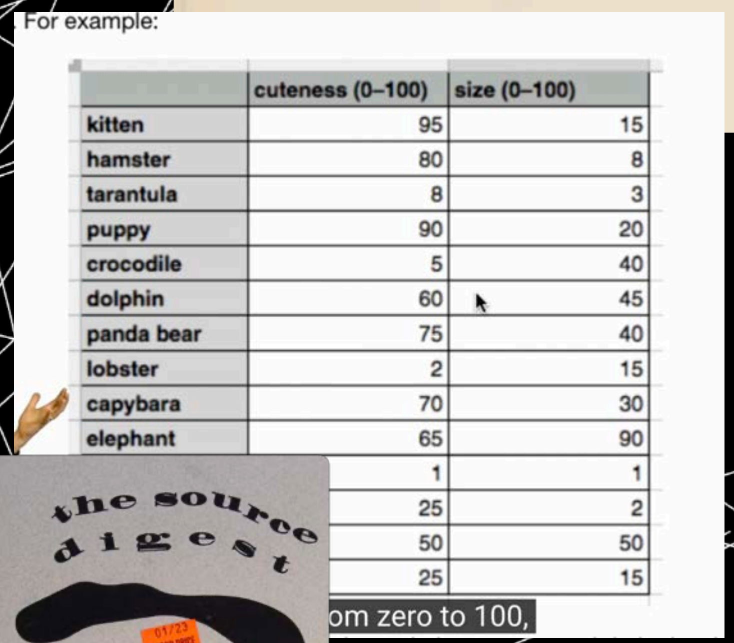
```
normalize(vPosition),                 // normalized ray direction
vec3(0,6372e3,0),                     // ray origin
uSunPos,                              // position of the sun
22.0,                                 // intensity of the sun
6371e3,                               // radius of the planet in meters
6471e3,                               // radius of the atmosphere in meters
vec3(5.5e-6, 13.0e-6, 22.4e-6),       // Rayleigh scattering coefficient
21e-6,                                // Mie scattering coefficient
8e3,                                  // Rayleigh scale height
1.2e3,                                // Mie scale height
0.758                                 // Mie preferred scattering direction
```

One of the shader implementations of atosphere that I found useful


…ed with an interesting discovery: if you restrict the number of CORDIC …th facets instead of smoothly!

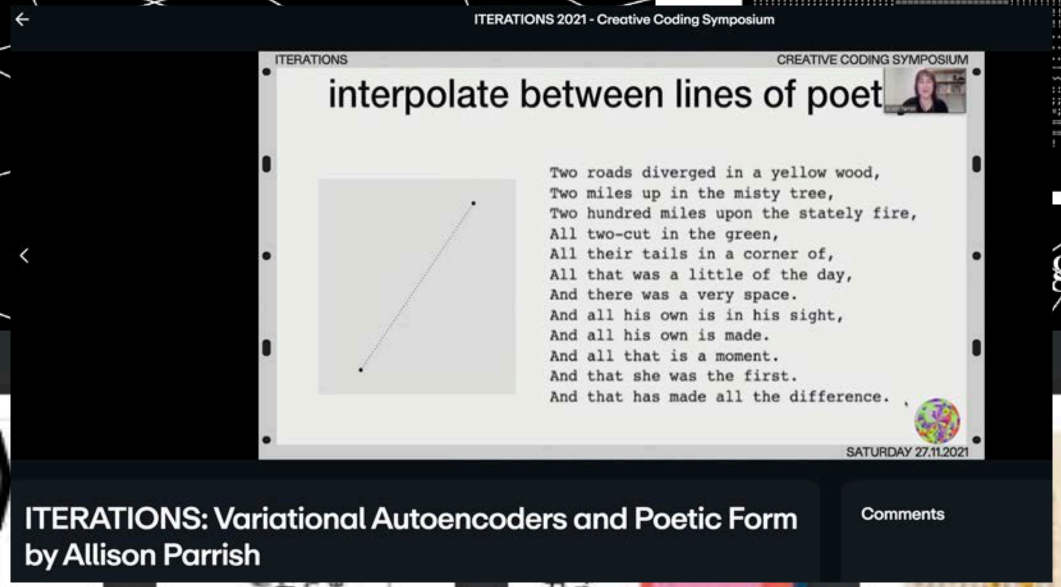
r₂ iterations: 1


So, on the one hand, software seems to be one of the most difficult things to preserve. It is essentially performance: unseizable, unstable, and flimsy. Its ability to perform depends on seemingly bottomless, nested sets of preconditions: a computer, an operating system, maybe some visual display, a certain input device, even a specific cultural background of the person operating it. In the worst case it requires "The Internet." On the other hand, the most variable cultural artifacts are made of software. Software has an incredible potential for changing its form, because it has no imperative form. Pinning a piece of software down to a single object makes preservation more difficult. Forcing uniqueness on the object-level is a futile struggle against the logic of software. Fanning it out into a larger number of objects makes preservation easier. Compared with eternity, uniqueness looks ridiculous anyway.


For example:

	cuteness (0–100)	size (0–100)
kitten	95	15
hamster	80	8
tarantula	8	3
puppy	90	20
crocodile	5	40
dolphin	60	45
panda bear	75	40
lobster	2	15
capybara	70	30
elephant	65	90
[illegible]	1	1
[illegible]	25	2
[illegible]	50	50
[illegible]	25	15

…om zero to 100,


ITERATIONS 2021 - Creative Coding Symposium

ITERATIONS CREATIVE CODING SYMPOSIUM

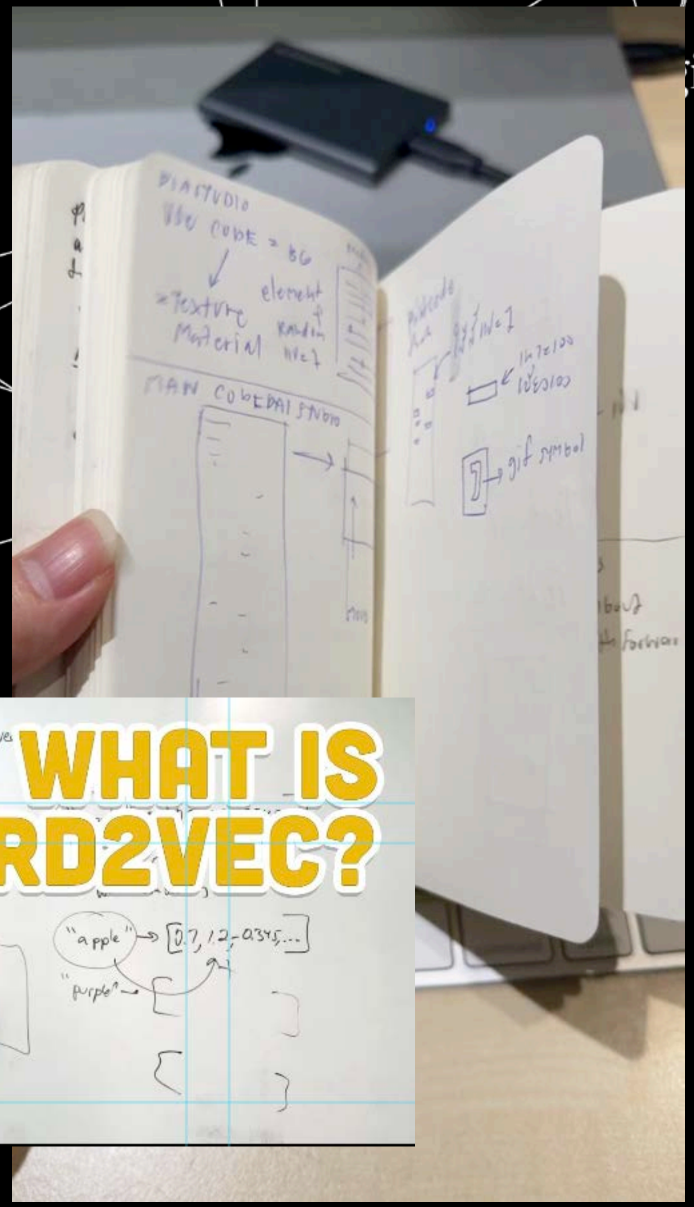
interpolate between lines of poet

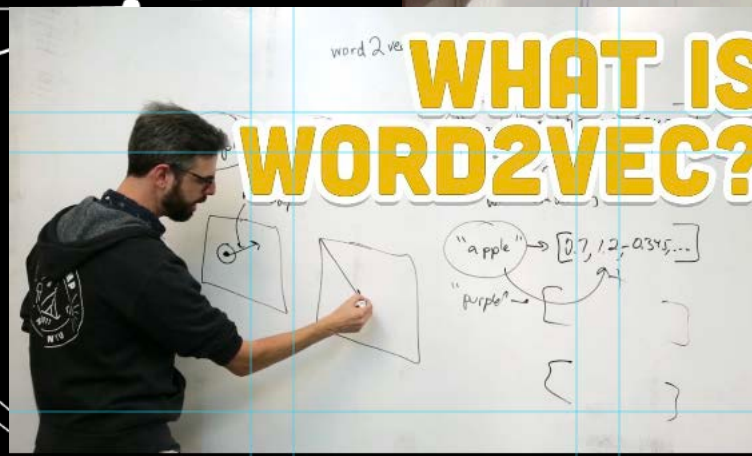
```
Two roads diverged in a yellow wood,
Two miles up in the misty tree,
Two hundred miles upon the stately fire,
All two-cut in the green,
All their tails in a corner of,
All that was a little of the day,
And there was a very space.
And all his own is in his sight,
And all his own is made.
And all that is a moment.
And that she was the first.
And that has made all the difference.
```

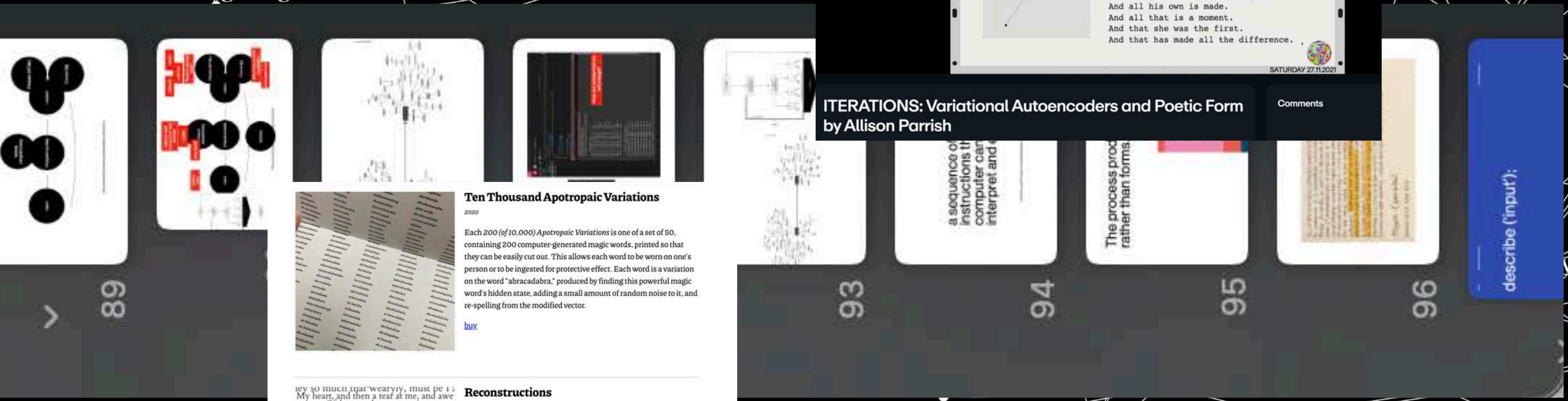
SATURDAY 27.11.2021

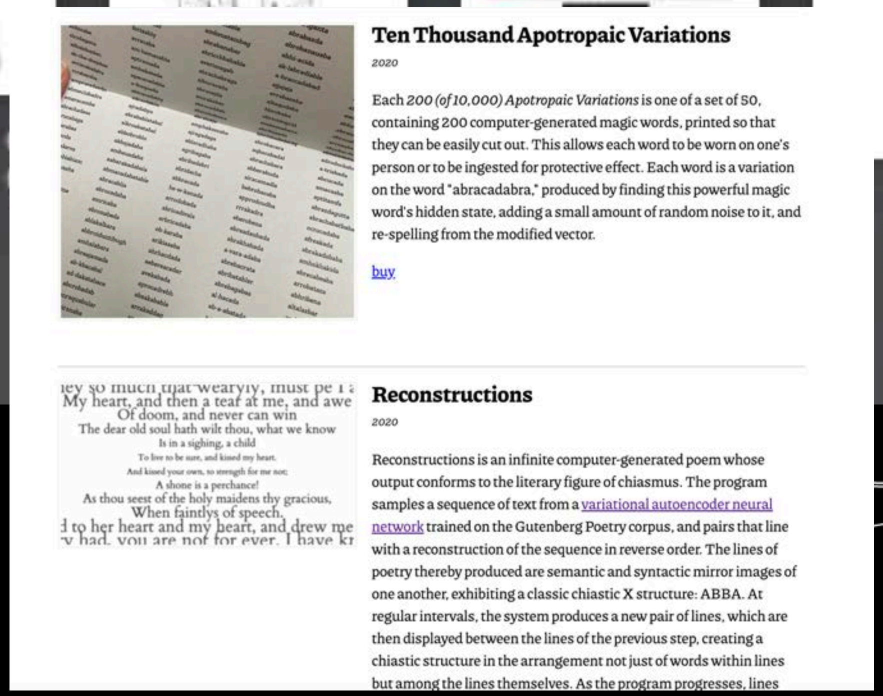
ITERATIONS: Variational Autoencoders and Poetic Form by Allison Parrish

Comments








Ten Thousand Apotropaic Variations
2020

Each *200 (of 10,000) Apotropaic Variations* is one of a set of 50, containing 200 computer-generated magic words, printed so that they can be easily cut out. This allows each word to be worn on one's person or to be ingested for protective effect. Each word is a variation on the word "abracadabra," produced by finding this powerful magic word's hidden state, adding a small amount of random noise to it, and re-spelling from the modified vector.

buy

Reconstructions
2020

Reconstructions is an infinite computer-generated poem whose output conforms to the literary figure of chiasmus. The program samples a sequence of text from a variational autoencoder neural network trained on the Gutenberg Poetry corpus, and pairs that line with a reconstruction of the sequence in reverse order. The lines of poetry thereby produced are semantic and syntactic mirror images of one another, exhibiting a classic chiastic X structure: ABBA. At regular intervals, the system produces a new pair of lines, which are then displayed between the lines of the previous step, creating a chiastic structure in the arrangement not just of words within lines but among the lines themselves. As the program progresses, lines

My positioning

The Superficial Coder

Last week, I mentioned that my entire process felt like patching and translating everything, and this week I am still struggling even more. I focus too much on research, and I am overloaded with content and ideas. More importantly, even though I have an idea, trying to code one that works is even more challenging.

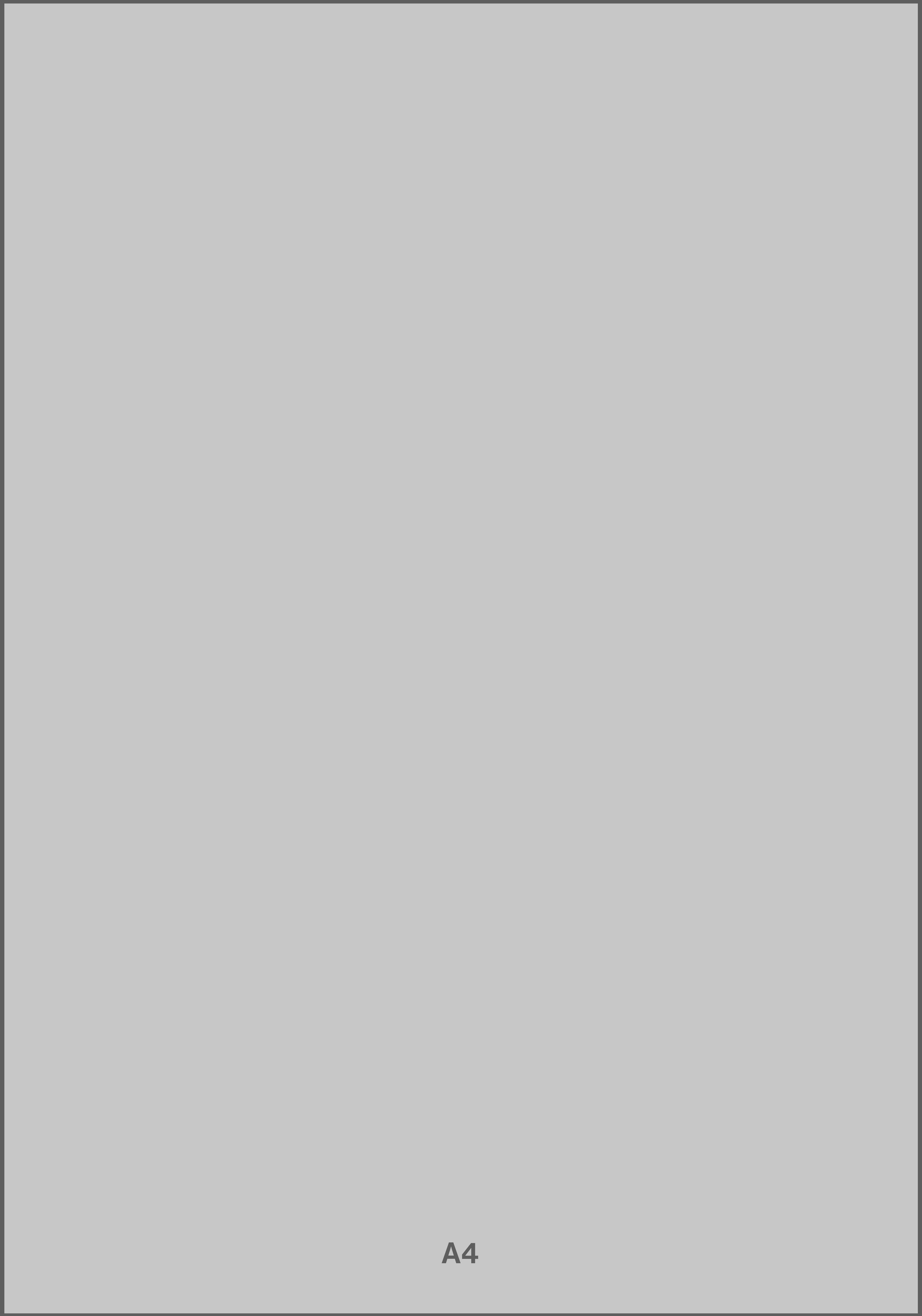
I need to remind myself that I am a superficial coder. I also agree that, given my abilities, I should explore the tool in “Default Mode" to remove creative distractions and focus on the materiality of the existing code in the p5.js library. I use the lens of Adhocism to explore the primordial stage of coding.

To start altering the context of the code operating in a digital format, I create my canvas in A4 size, mirroring the physical format to modify its usual appearance.

Mode of approach

default();

Focusing on the materiality of the code that already exists in p5.js



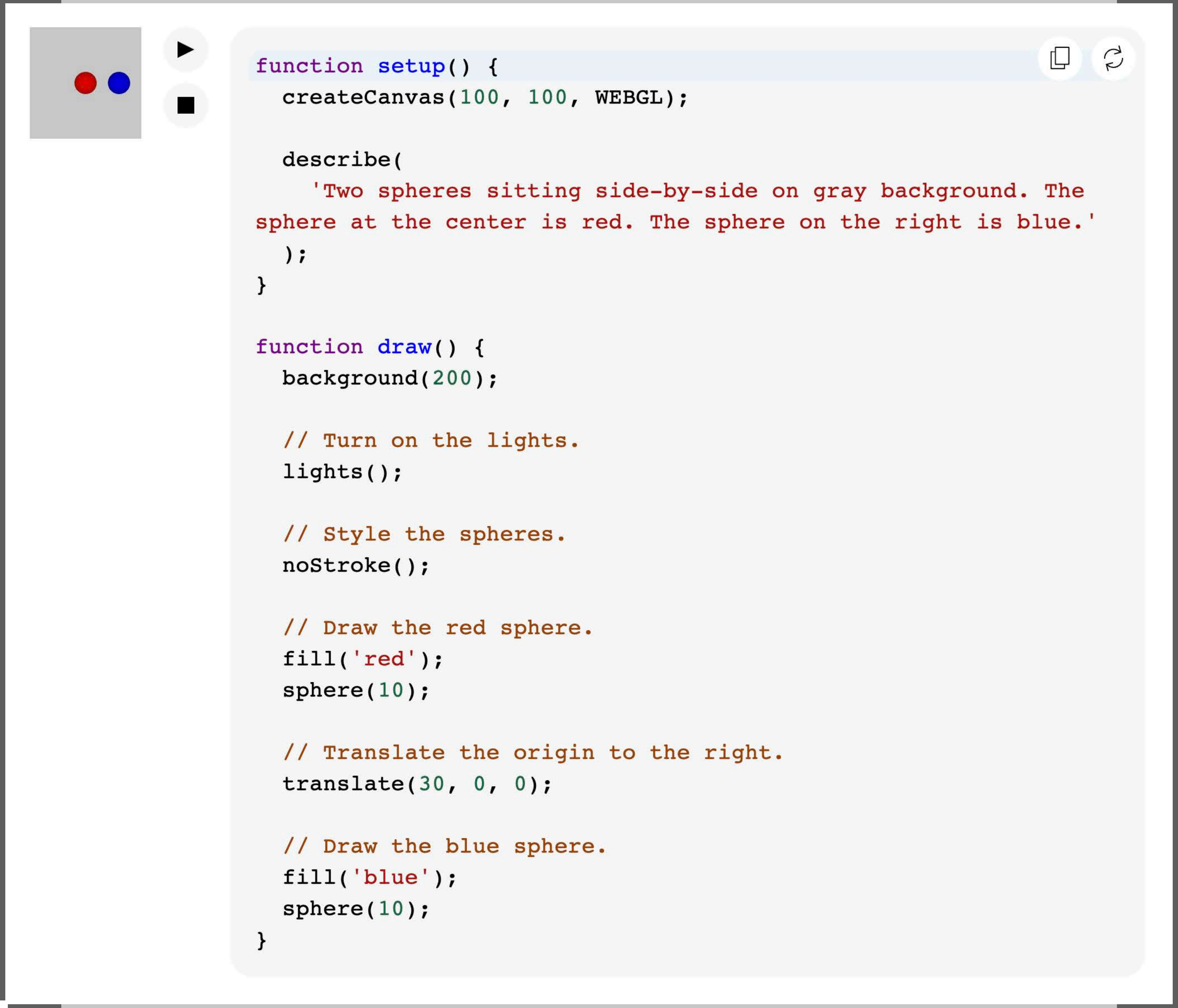
The Iteration Set

01

Using the ‘translate()’ function name as literal text

Subvert:
Library and function

Translate() in p5.js means to translate the coordinate system.



```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    'Two spheres sitting side-by-side on gray background. The
sphere at the center is red. The sphere on the right is blue.'
  );
}

function draw() {
  background(200);

  // Turn on the lights.
  lights();

  // Style the spheres.
  noStroke();

  // Draw the red sphere.
  fill('red');
  sphere(10);

  // Translate the origin to the right.
  translate(30, 0, 0);

  // Draw the blue sphere.
  fill('blue');
  sphere(10);
}
```

Syntax

```
translate(x, y, [z])
```

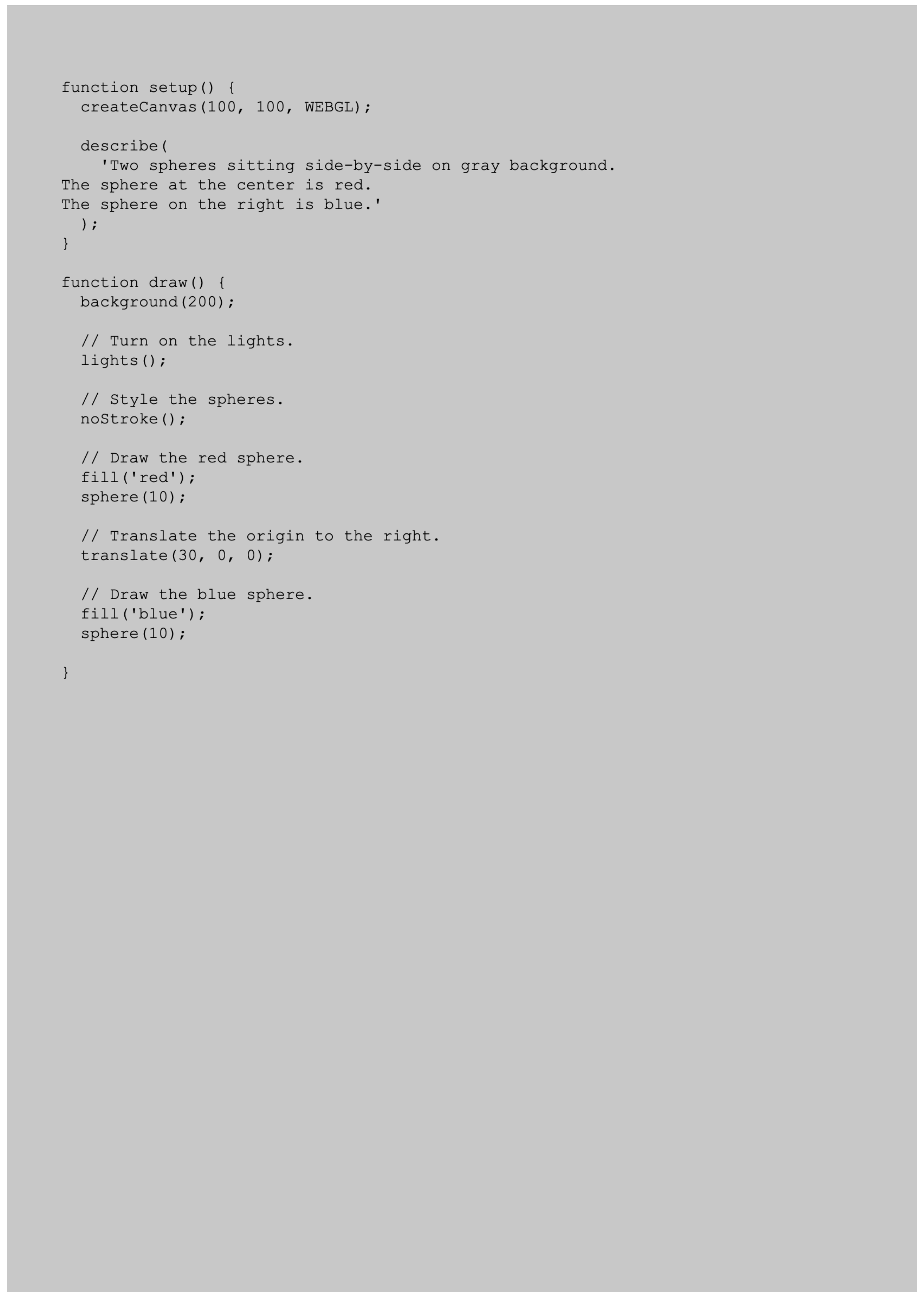
The Iteration Set 01-1: Default

```
function setup() {
createCanvas(100, 100, WEBGL);

describe(
'Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue.'
);
}

function draw() {
background(200);

// Turn on the lights.
lights();

// Style the spheres.
noStroke();

// Draw the red sphere.
fill('red');
sphere(10);

// Translate the origin to the right.
translate(30, 0, 0);

// Draw the blue sphere.
fill('blue');
sphere(10);

}
```

I used the translate() function and its example as my reference, assuming that the function and its library are either non-existent or non-functional. I created code that displays this example on the preview canvas and connects the Google Translate API with p5.js, allowing me to translate it into another language. This approach reflects my perspective and that of other users who may not be fluent in English or familiar with coding, which has made my learning process more difficult. I kept the default design provided by the tool.

```
function setup() {
createCanvas(100, 100, WEBGL);

describe(
'Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue.'
);
}

function draw() {
background(200);

// Turn on the lights.
lights();

// Style the spheres.
noStroke();

// Draw the red sphere.
fill('red');
sphere(10);

// Translate the origin to the right.
translate(30, 0, 0);

// Draw the blue sphere.
fill('blue');
sphere(10);

}
```

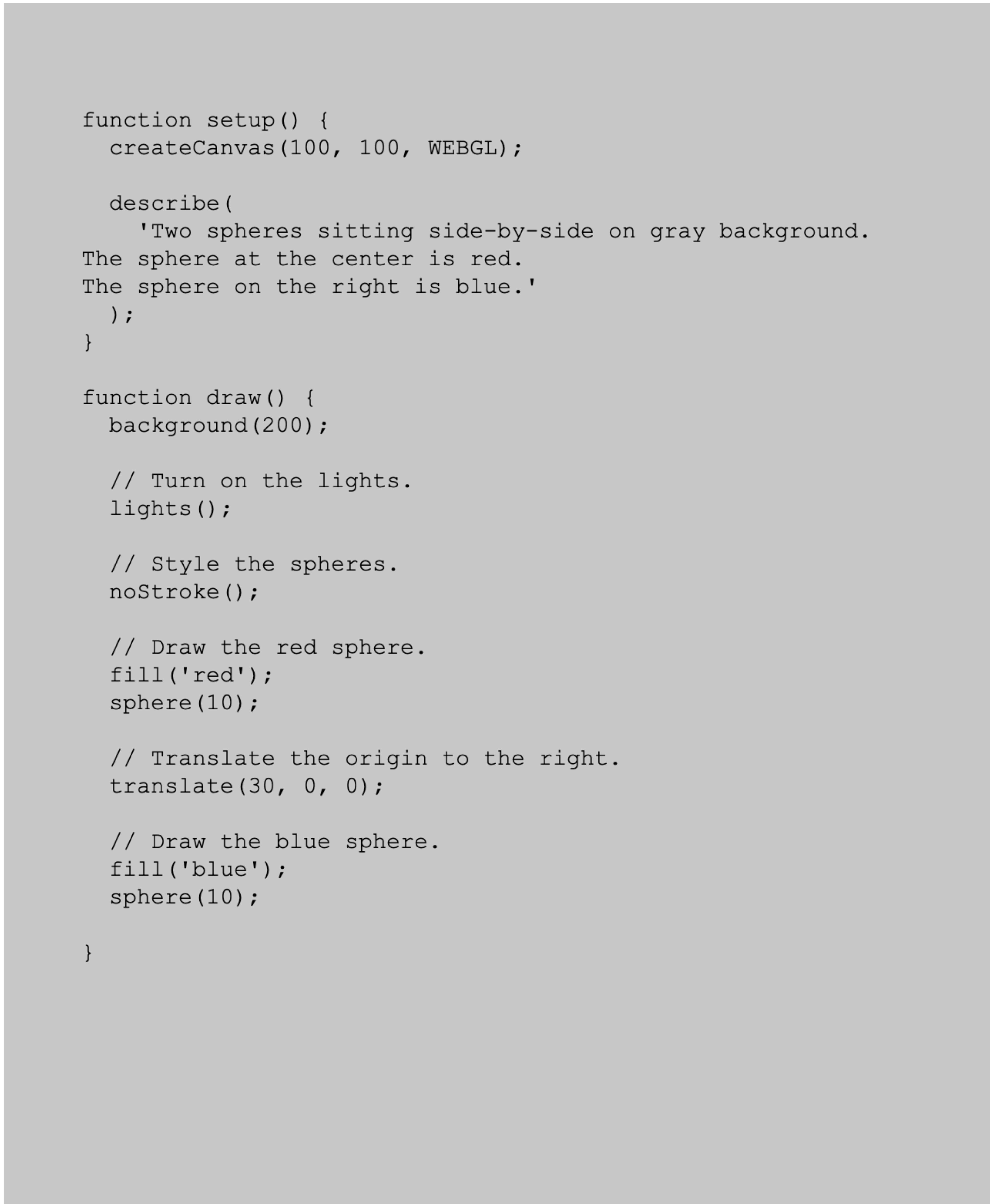
The Iteration Set 01-2



```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    'Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue.'
  );
}

function draw() {
  background(200);

  // Turn on the lights.
  lights();

  // Style the spheres.
  noStroke();

  // Draw the red sphere.
  fill('red');
  sphere(10);

  // Translate the origin to the right.
  translate(30, 0, 0);

  // Draw the blue sphere.
  fill('blue');
  sphere(10);

}
```

In the second version, I refined the layout by adjusting the margins and leading, and set the font to the same one the library uses: Courier New. It is a monospaced font essential for maintaining code consistency.

Font: Courier New / Monospaced Font



```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    'Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue.'
  );
}

function draw() {
  background(200);

  // Turn on the lights.
  lights();

  // Style the spheres.
  noStroke();

  // Draw the red sphere.
  fill('red');
  sphere(10);

  // Translate the origin to the right.
  translate(30, 0, 0);

  // Draw the blue sphere.
  fill('blue');
  sphere(10);

}
```

The Iteration Set 01-3

```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    'Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue.'
  );
}

function draw() {
  background(200);

  // Turn on the lights.
  lights();

  // Style the spheres.
  noStroke();

  // Draw the red sphere.
  fill('red');
  sphere(10);

  // Translate the origin to the right.
  translate(30, 0, 0);

  // Draw the blue sphere.
  fill('blue');
  sphere(10);

}
```

After translating it into another language, I had to change the font to Noto Sans to ensure more consistent typography across languages, as it supports nearly all languages worldwide.

Font: Noto Sans / Universal Font

```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    'Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue.'
  );
}

function draw() {
  background(200);

  // Turn on the lights.
  lights();

  // Style the spheres.
  noStroke();

  // Draw the red sphere.
  fill('red');
  sphere(10);

  // Translate the origin to the right.
  translate(30, 0, 0);

  // Draw the blue sphere.
  fill('blue');
  sphere(10);

}
```

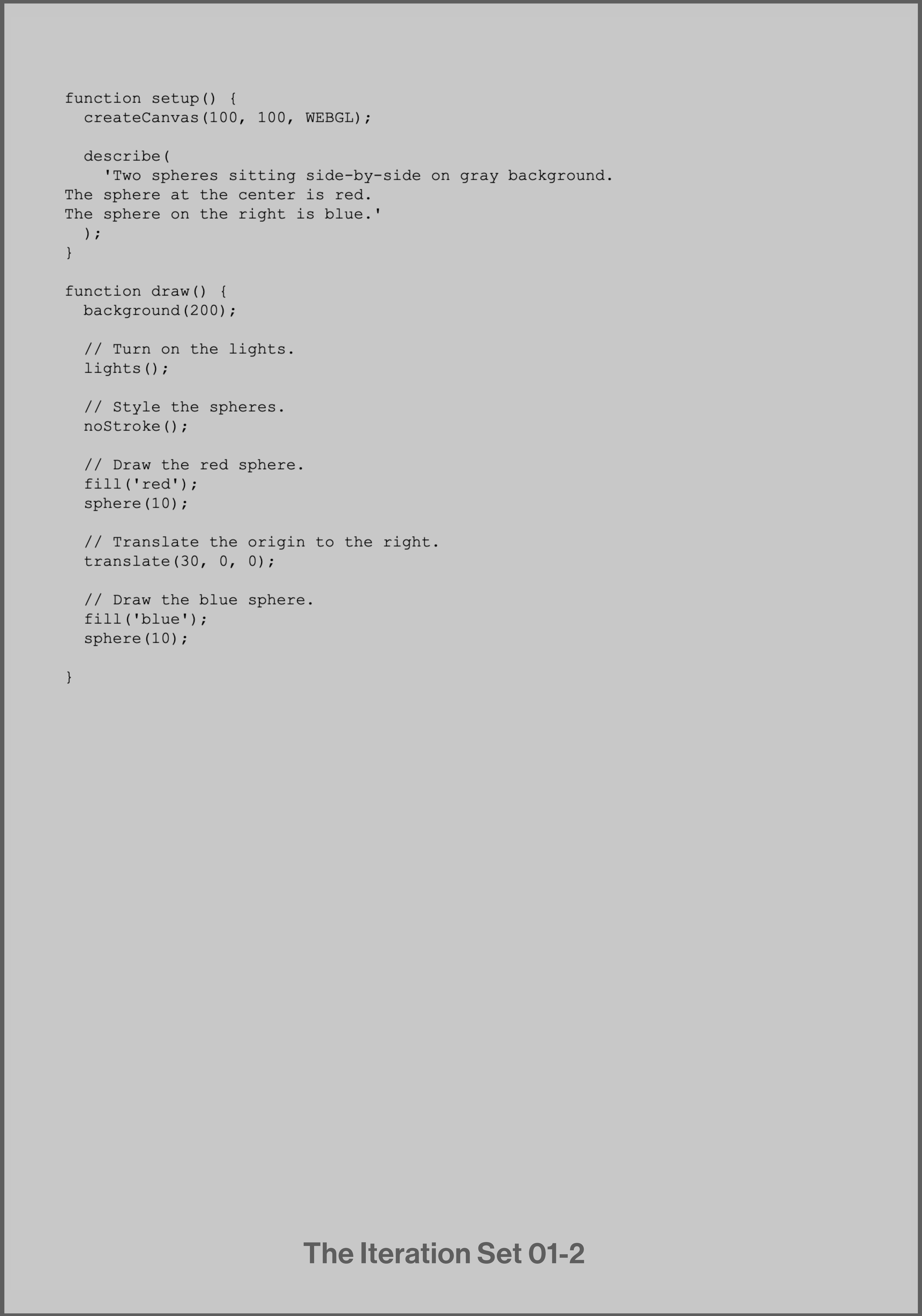
The Iteration Set

02

Deconstructing the syntax elements of code

Subvert: Syntax

I removed symbols, variables, and so on to show individual types of text functions or letters and investigate how complex the syntax for beginners has to be, as these are all essential and cannot be lost; otherwise, the code will error.



```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    'Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue.'
  );
}

function draw() {
  background(200);

  // Turn on the lights.
  lights();

  // Style the spheres.
  noStroke();

  // Draw the red sphere.
  fill('red');
  sphere(10);

  // Translate the origin to the right.
  translate(30, 0, 0);

  // Draw the blue sphere.
  fill('blue');
  sphere(10);

}
```

The Iteration Set 01-2

The Iteration Set 02-1 / 02-2

```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    'Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue.'
  );
}

function draw() {
  background(200);

  // Turn on the lights.
  lights();

  // Style the spheres.
  noStroke();

  // Draw the red sphere.
  fill('red');
  sphere(10);

  // Translate the origin to the right.
  translate(30, 0, 0);

  // Draw the blue sphere.
  fill('blue');
  sphere(10);

}
```

```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    ''
  );
}

function draw() {
  background(200);


  lights();


  noStroke();


  fill('red');
  sphere(10);


  translate(30, 0, 0);


  fill('blue');
  sphere(10);

}
```

The Iteration Set 02-3

Translate synonym

```
function setup
  create Canvas

  describe




function draw
  background


  lights


  no Stroke


  fill red
  sphere


  translate


  fill blue
  sphere
```

Another function I added to my code is the Datamuse synonym API, so every time it refreshes, the words are replaced with synonyms until the text becomes nonsensical.

The Iteration Set

03

Overlaying the iterations of ‘set 01 and 02’ using Photoshop

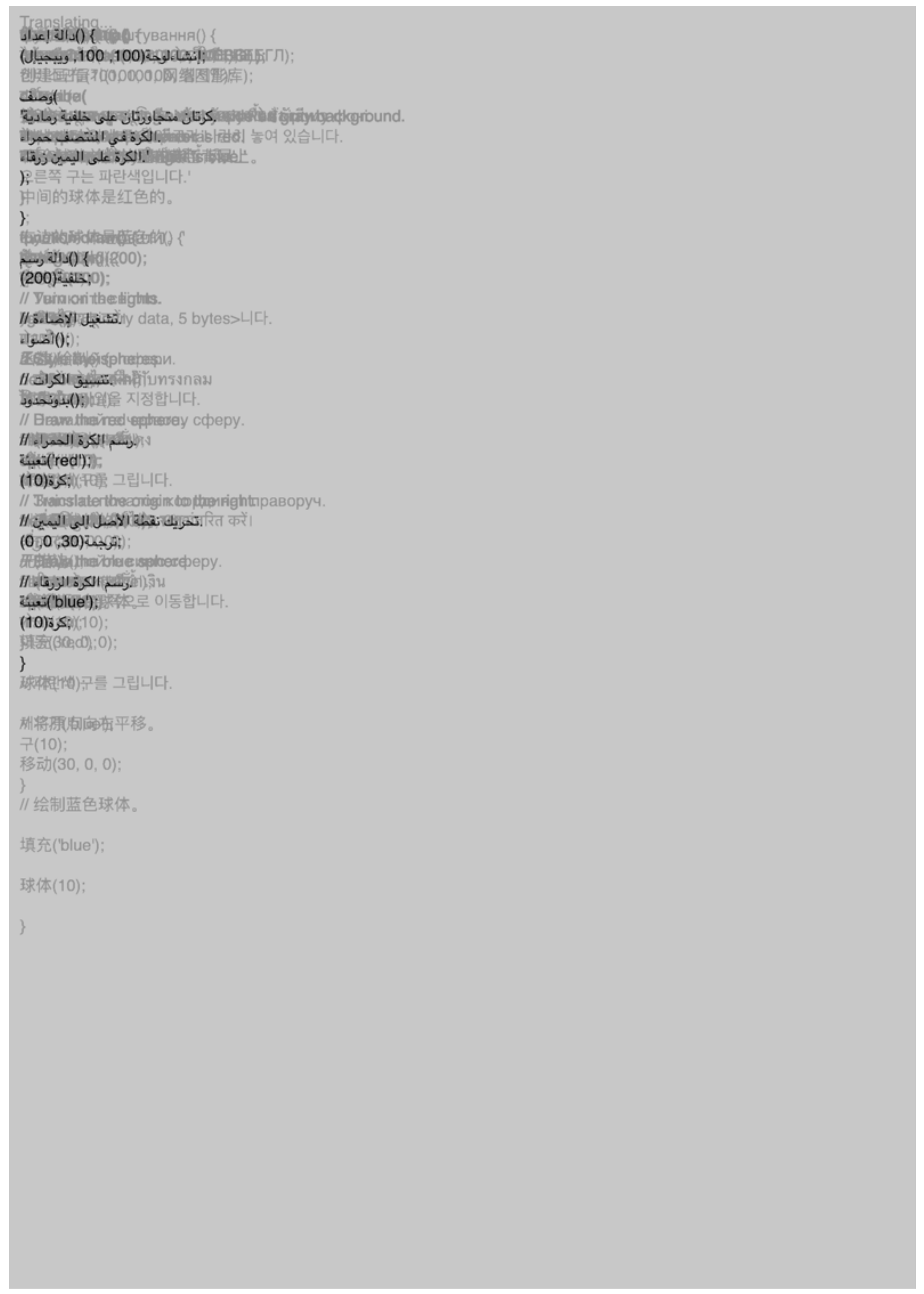
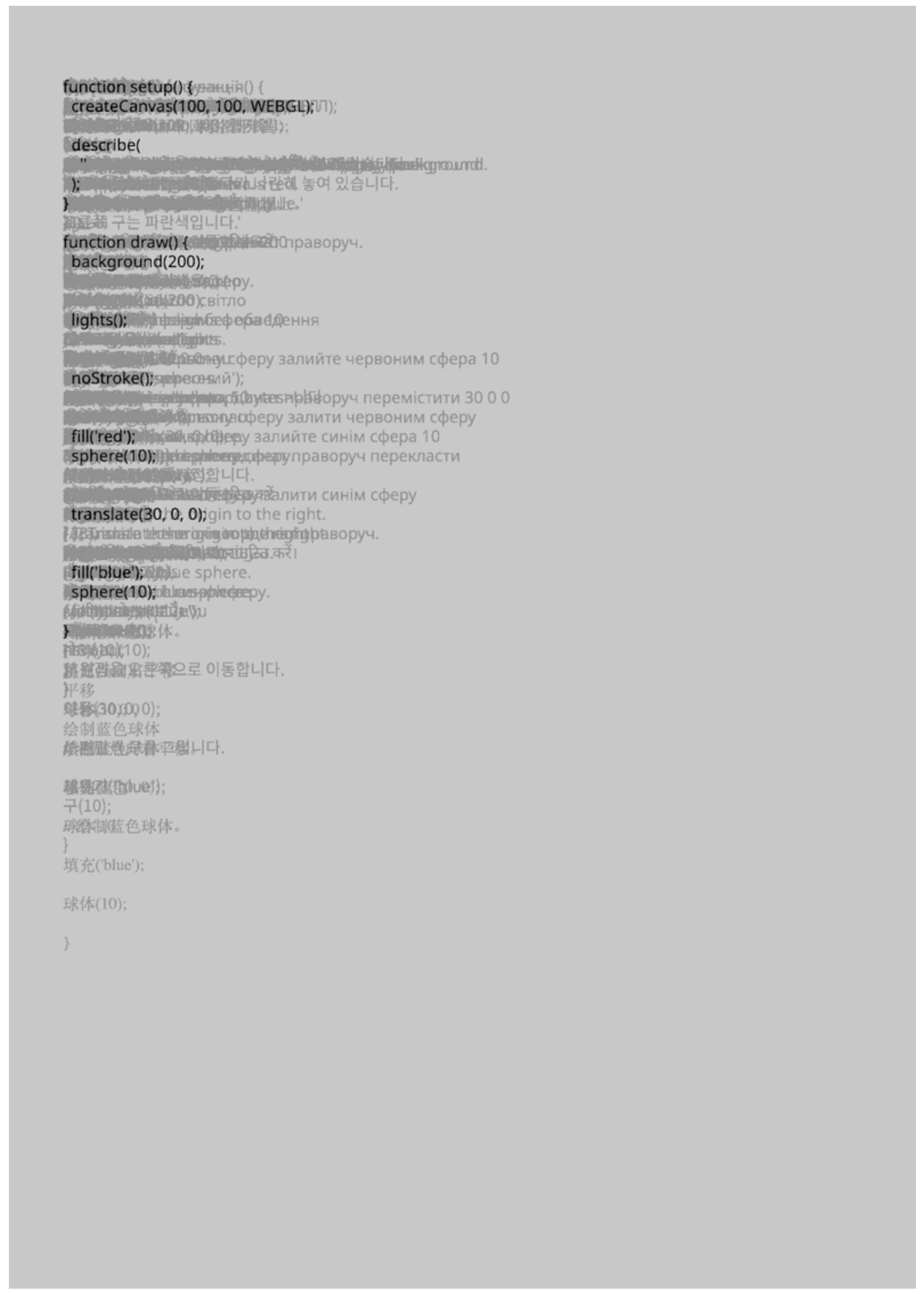
Subvert:
Code

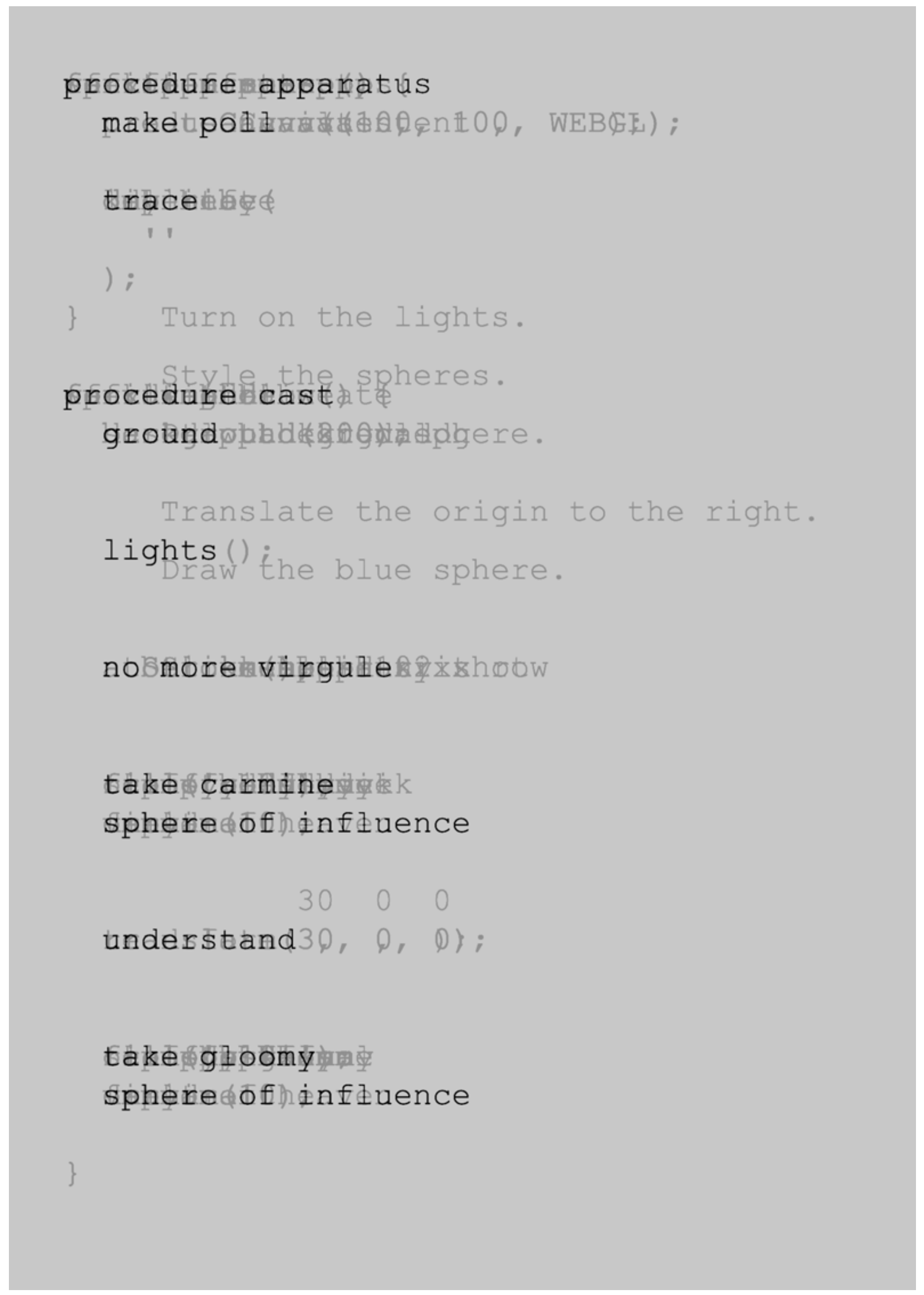
After completing those iterations, I overlaid them in Photoshop using three blending modes: Pin Light, Darken, and Difference. The resulting effects are interesting, as I aim to evoke the sensation of being lost in translation. Sometimes the meaning of words becomes blurred, or some words even pixelate when languages blend, much like how a beginner coder often needs to revisit the library to recheck the meaning of their code.

```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    'Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue.'
  );
}

function draw() {
  background(200);

  // Turn on the lights.
  lights();

  // Style the spheres.
  noStroke();

  // Draw the red sphere.
  fill('red');
  sphere(10);

  // Translate the origin to the right.
  translate(30, 0, 0);

  // Draw the blue sphere.
  fill('blue');
  sphere(10);

}
```

The Iteration Set 01 / 02

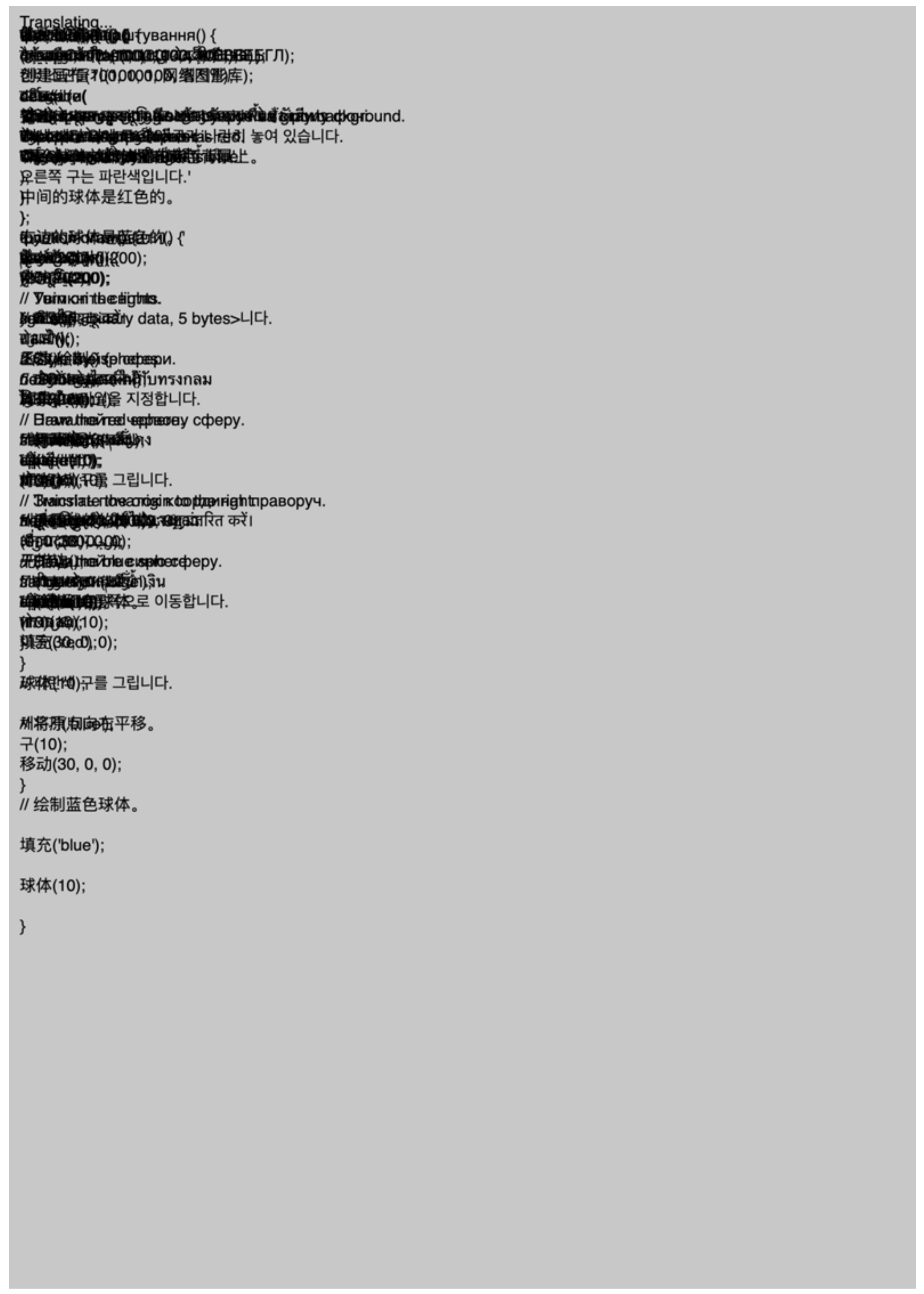
The Iteration Set 03-1

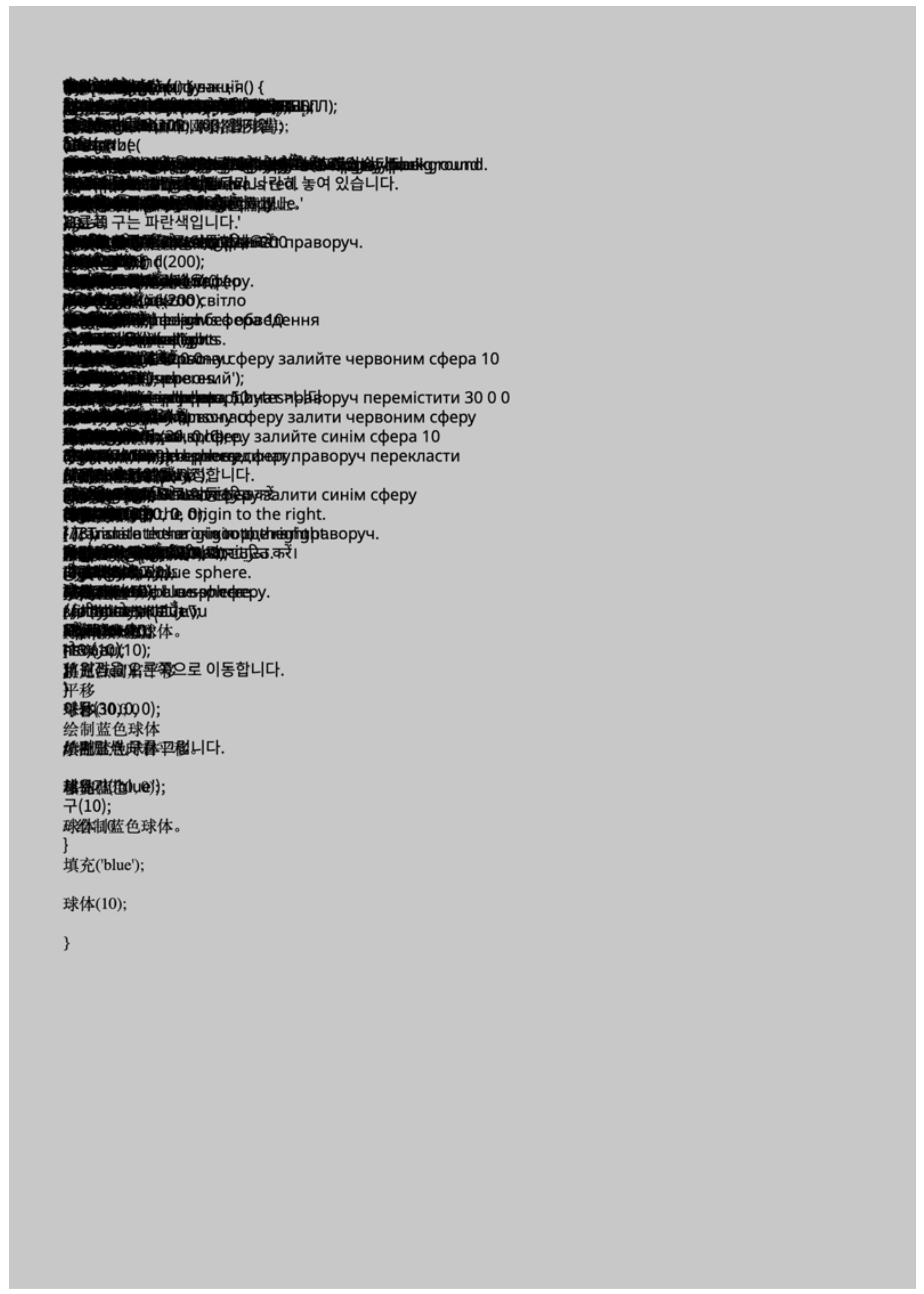



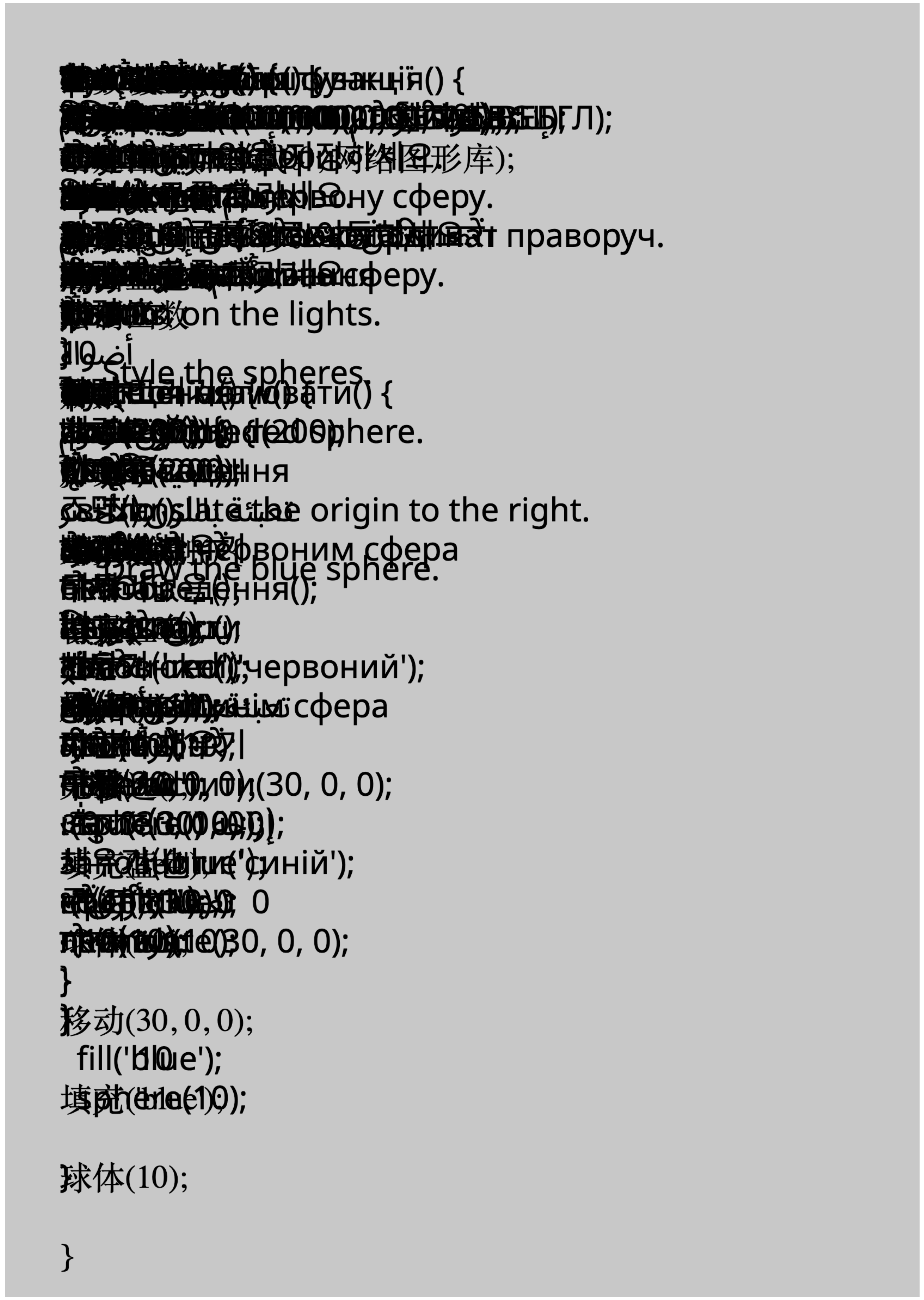


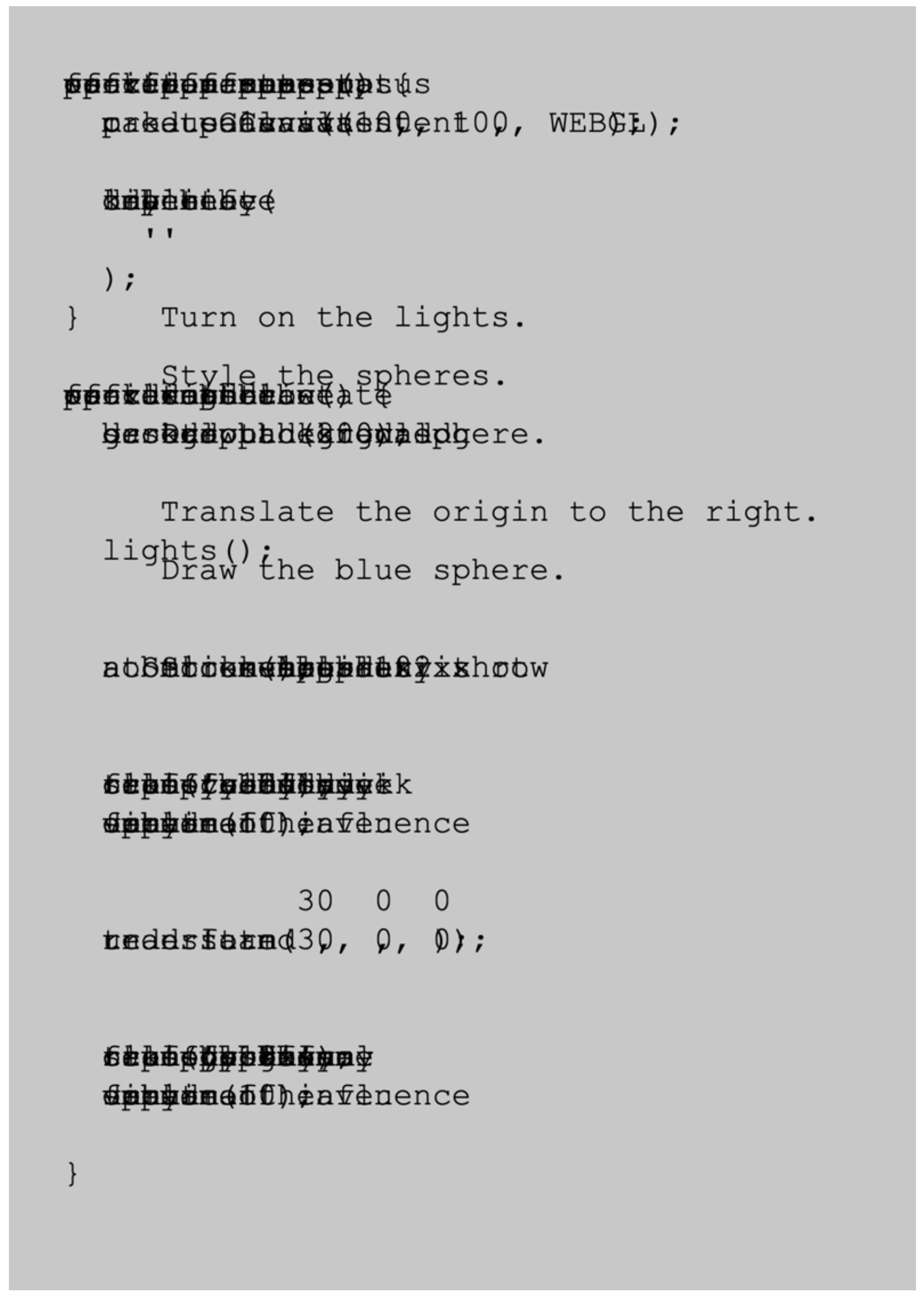
Blending Mode:
Pin Light

The Iteration Set 03-2



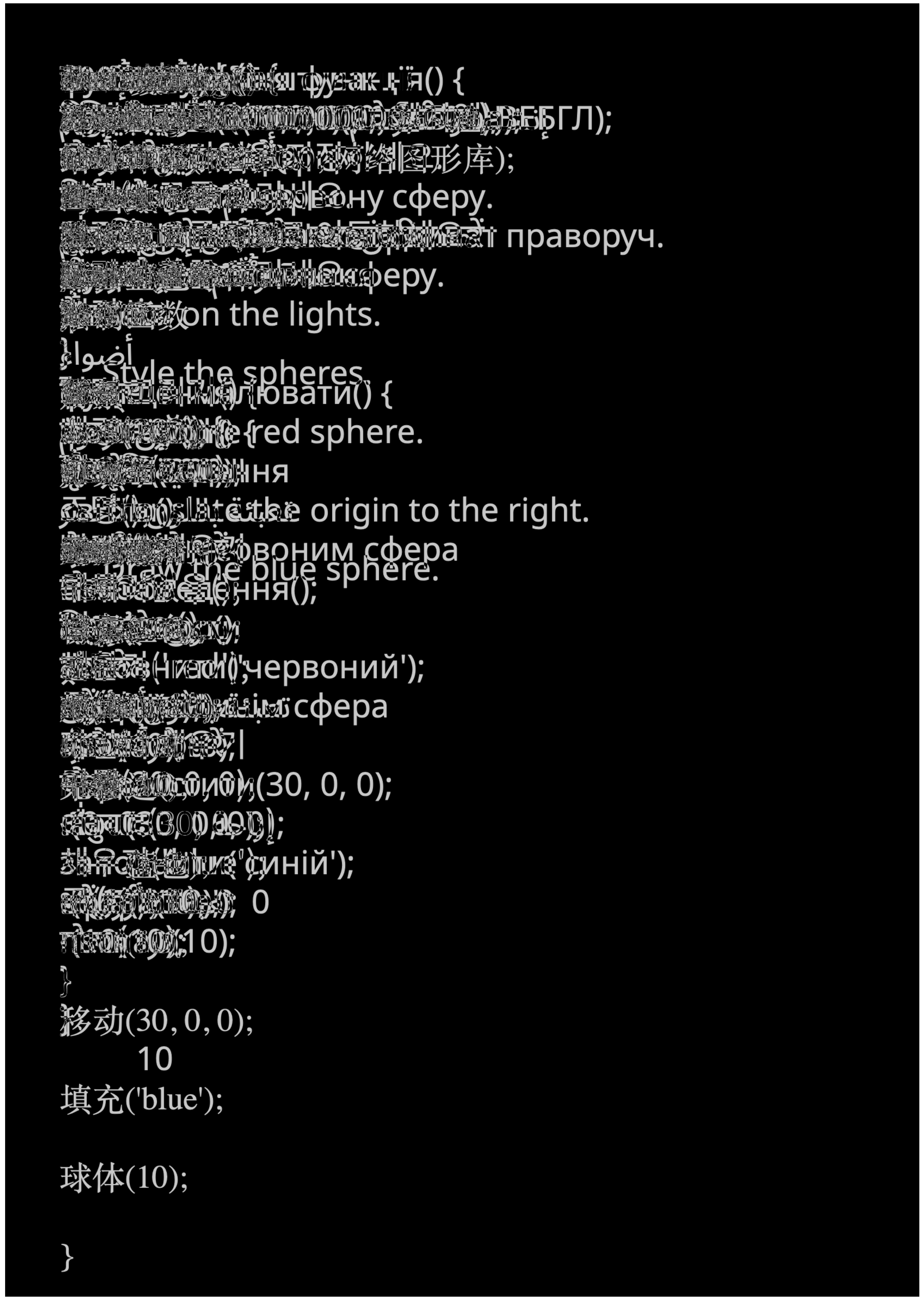


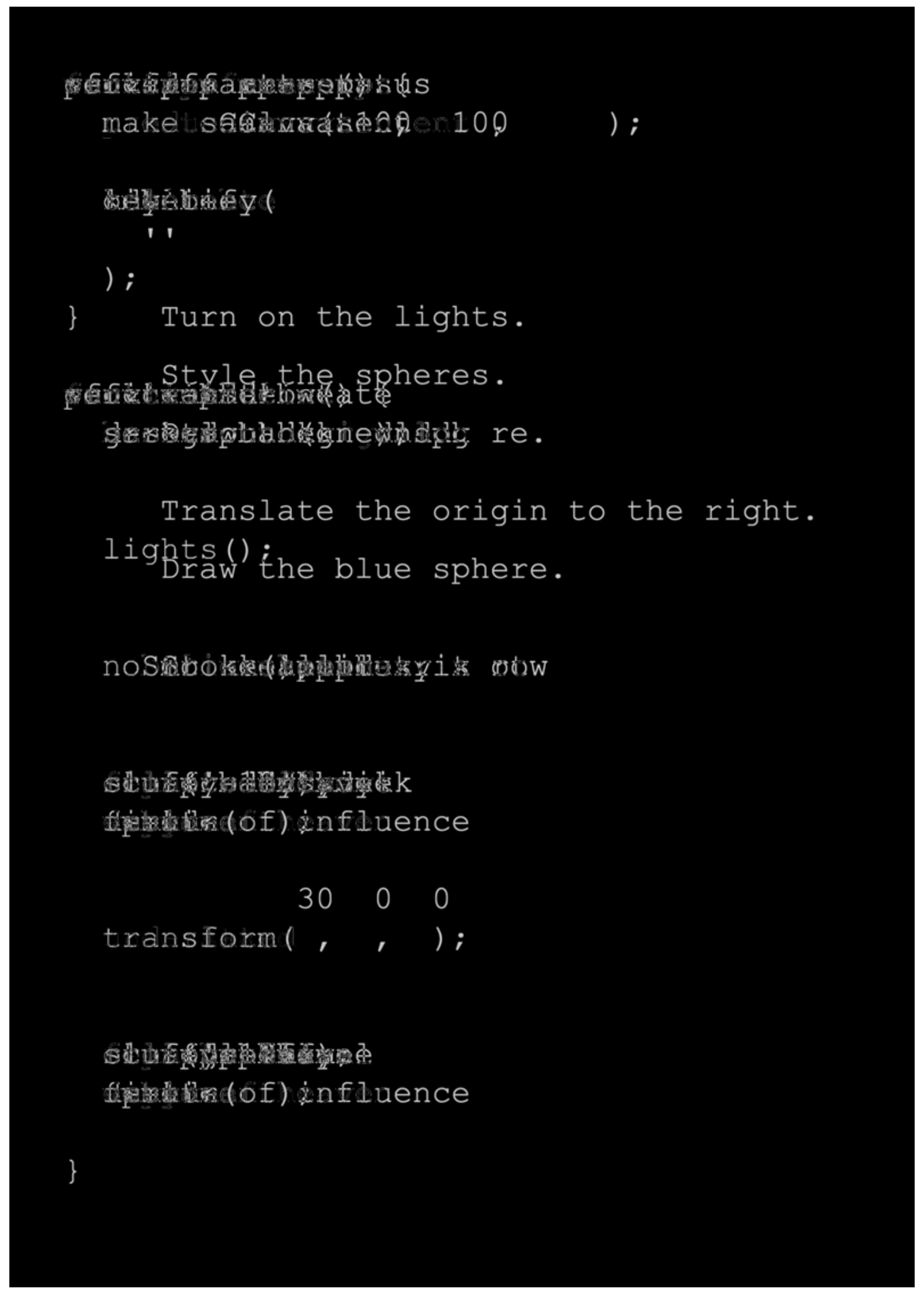




Blending Mode:
Darken

The Iteration Set 03-3



"Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue."


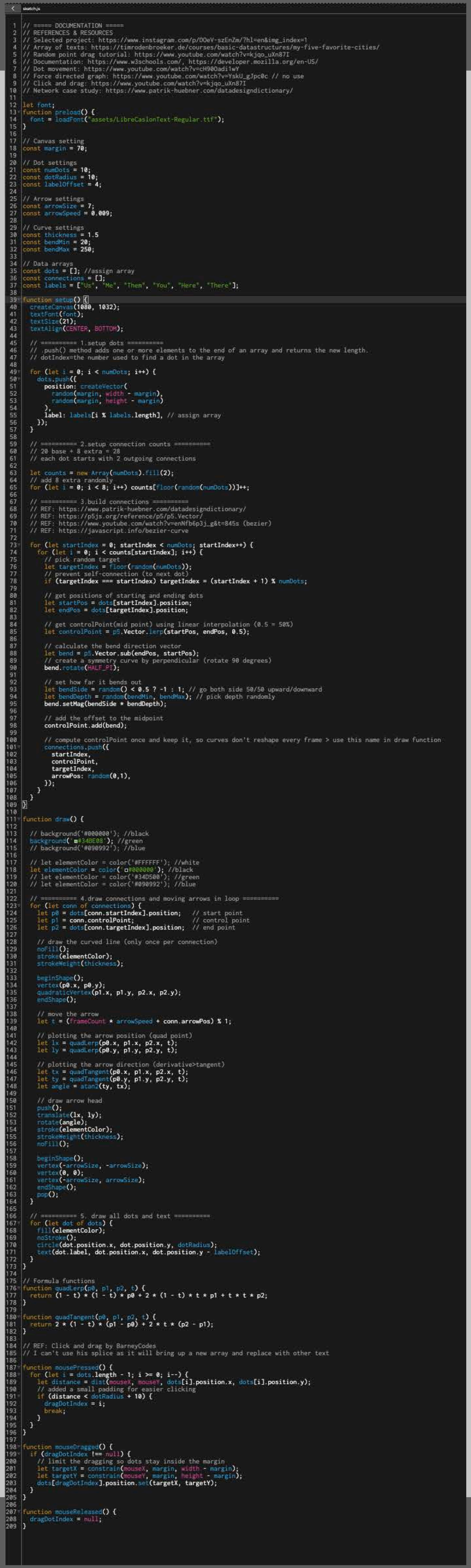
Blending Mode:
Difference

The Iteration Set

04

Over-compressing the code from the replicated project

Subvert: Code



sketch.js

```javascript
// ===== DOCUMENTATION =====
// REFERENCES & RESOURCES
// Selected project: https://www.instagram.com/p/DOeV-szEnZm/?hl=en&img_index=1
// Array of texts: https://timrodenbroeker.de/courses/basic-datastructures/my-five-favorite-cities/
// Random point drag tutorial: https://www.youtube.com/watch?v=kjqo_uXn87I
// Documentation: https://www.w3schools.com/, https://developer.mozilla.org/en-US/
// Dot movement: https://www.youtube.com/watch?v=cH90Oadi1wY
// Force directed graph: https://www.youtube.com/watch?v=YskU_gJpc0c // no use
// Click and drag: https://www.youtube.com/watch?v=kjqo_uXn87I
// Network case study: https://www.patrik-huebner.com/datadesigndictionary/

let font;
function preload() {
  font = loadFont("assets/LibreCaslonText-Regular.ttf");
}

// Canvas setting
const margin = 70;

// Dot settings
const numDots = 10;
const dotRadius = 10;
const labelOffset = 4;

// Arrow settings
const arrowSize = 7;
const arrowSpeed = 0.009;

// Curve settings
const thickness = 1.5
const bendMin = 20;
const bendMax = 250;

// Data arrays
const dots = []; //assign array
const connections = [];
const labels = ["Us", "Me", "Them", "You", "Here", "There"];

function setup() {
  createCanvas(1080, 1032);
  textFont(font);
  textSize(21);
  textAlign(CENTER, BOTTOM);

  // ========== 1.setup dots ==========
  // .push() method adds one or more elements to the end of an array and returns the new length.
  // dotIndex=the number used to find a dot in the array

  for (let i = 0; i < numDots; i++) {
    dots.push({
      position: createVector(
        random(margin, width - margin),
        random(margin, height - margin)
      ),
      label: labels[i % labels.length], // assign array
    });
  }

  // ========== 2.setup connection counts ==========
  // 20 base + 8 extra = 28
  // each dot starts with 2 outgoing connections

  let counts = new Array(numDots).fill(2);
  // add 8 extra randomly
  for (let i = 0; i < 8; i++) counts[floor(random(numDots))]++;

  // ========== 3.build connections ==========
  // REF: https://www.patrik-huebner.com/datadesigndictionary/
  // REF: https://p5js.org/reference/p5/p5.Vector/
  // REF: https://www.youtube.com/watch?v=enNfb6p3j_g&t=845s (bezier)
  // REF: https://javascript.info/bezier-curve

  for (let startIndex = 0; startIndex < numDots; startIndex++) {
    for (let i = 0; i < counts[startIndex]; i++) {
      // pick random target
      let targetIndex = floor(random(numDots));
      // prevent self-connection (to next dot)
      if (targetIndex === startIndex) targetIndex = (startIndex + 1) % numDots;

      // get positions of starting and ending dots
      let startPos = dots[startIndex].position;
      let endPos = dots[targetIndex].position;

      // get controlPoint(mid point) using linear interpolation (0.5 = 50%)
      let controlPoint = p5.Vector.lerp(startPos, endPos, 0.5);

      // calculate the bend direction vector
      let bend = p5.Vector.sub(endPos, startPos);
      // create a symmetry curve by perpendicular (rotate 90 degrees)
      bend.rotate(HALF_PI);

      // set how far it bends out
      let bendSide = random() < 0.5 ? -1 : 1; // go both side 50/50 upward/downward
      let bendDepth = random(bendMin, bendMax); // pick depth randomly
      bend.setMag(bendSide * bendDepth);

      // add the offset to the midpoint
      controlPoint.add(bend);

      // compute controlPoint once and keep it, so curves don't reshape every frame > use this name in draw function
      connections.push({
        startIndex,
        controlPoint,
        targetIndex,
        arrowPos: random(0,1),
      });
    }
  }
}

function draw() {

  // background('#000000'); //black
  background('#34BE08'); //green
  // background('#090992'); //blue

  // let elementColor = color('#FFFFFF'); //white
  let elementColor = color('#000000'); //black
  // let elementColor = color('#34D500'); //green
  // let elementColor = color('#090992'); //blue

  // ========== 4.draw connections and moving arrows in loop ==========
  for (let conn of connections) {
    let p0 = dots[conn.startIndex].position;   // start point
    let p1 = conn.controlPoint;                // control point
    let p2 = dots[conn.targetIndex].position;  // end point

    // draw the curved line (only once per connection)
    noFill();
    stroke(elementColor);
    strokeWeight(thickness);

    beginShape();
    vertex(p0.x, p0.y);
    quadraticVertex(p1.x, p1.y, p2.x, p2.y);
    endShape();

    // move the arrow
    let t = (frameCount * arrowSpeed + conn.arrowPos) % 1;

    // plotting the arrow position (quad point)
    let lx = quadLerp(p0.x, p1.x, p2.x, t);
    let ly = quadLerp(p0.y, p1.y, p2.y, t);

    // plotting the arrow direction (derivative>tangent)
    let tx = quadTangent(p0.x, p1.x, p2.x, t);
    let ty = quadTangent(p0.y, p1.y, p2.y, t);
    let angle = atan2(ty, tx);

    // draw arrow head
    push();
    translate(lx, ly);
    rotate(angle);
    stroke(elementColor);
    strokeWeight(thickness);
    noFill();

    beginShape();
    vertex(-arrowSize, -arrowSize);
    vertex(0, 0);
    vertex(-arrowSize, arrowSize);
    endShape();
    pop();
  }

  // ========== 5. draw all dots and text ==========
  for (let dot of dots) {
    fill(elementColor);
    noStroke();
    circle(dot.position.x, dot.position.y, dotRadius);
    text(dot.label, dot.position.x, dot.position.y - labelOffset);
  }
}

// Formula functions
function quadLerp(p0, p1, p2, t) {
  return (1 - t) * (1 - t) * p0 + 2 * (1 - t) * t * p1 + t * t * p2;
}

function quadTangent(p0, p1, p2, t) {
  return 2 * (1 - t) * (p1 - p0) + 2 * t * (p2 - p1);
}

// REF: Click and drag by BarneyCodes
// I can't use his splice as it will bring up a new array and replace with other text

function mousePressed() {
  for (let i = dots.length - 1; i >= 0; i--) {
    let distance = dist(mouseX, mouseY, dots[i].position.x, dots[i].position.y);
    // added a small padding for easier clicking
    if (distance < dotRadius + 10) {
      dragDotIndex = i;
      break;
    }
  }
}

function mouseDragged() {
  if (dragDotIndex !== null) {
    // limit the dragging so dots stay inside the margin
    let targetX = constrain(mouseX, margin, width - margin);
    let targetY = constrain(mouseY, margin, height - margin);
    dots[dragDotIndex].position.set(targetX, targetY);
  }
}

function mouseReleased() {
  dragDotIndex = null;
}
```

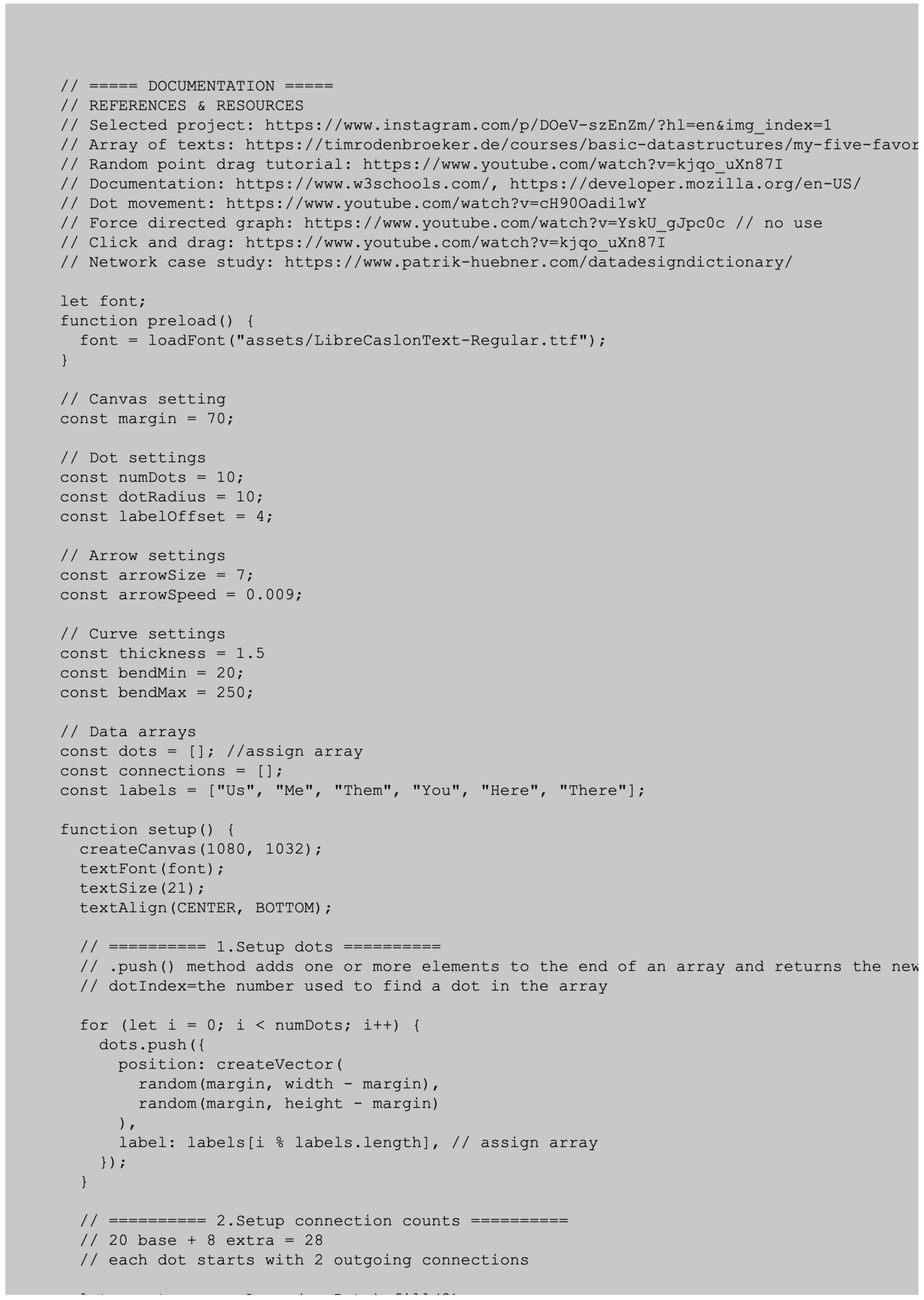
I displayed the code from the replicated project to compare a more advanced version with the simpler first-iteration syntax, translate().

The Iteration Set 04-1

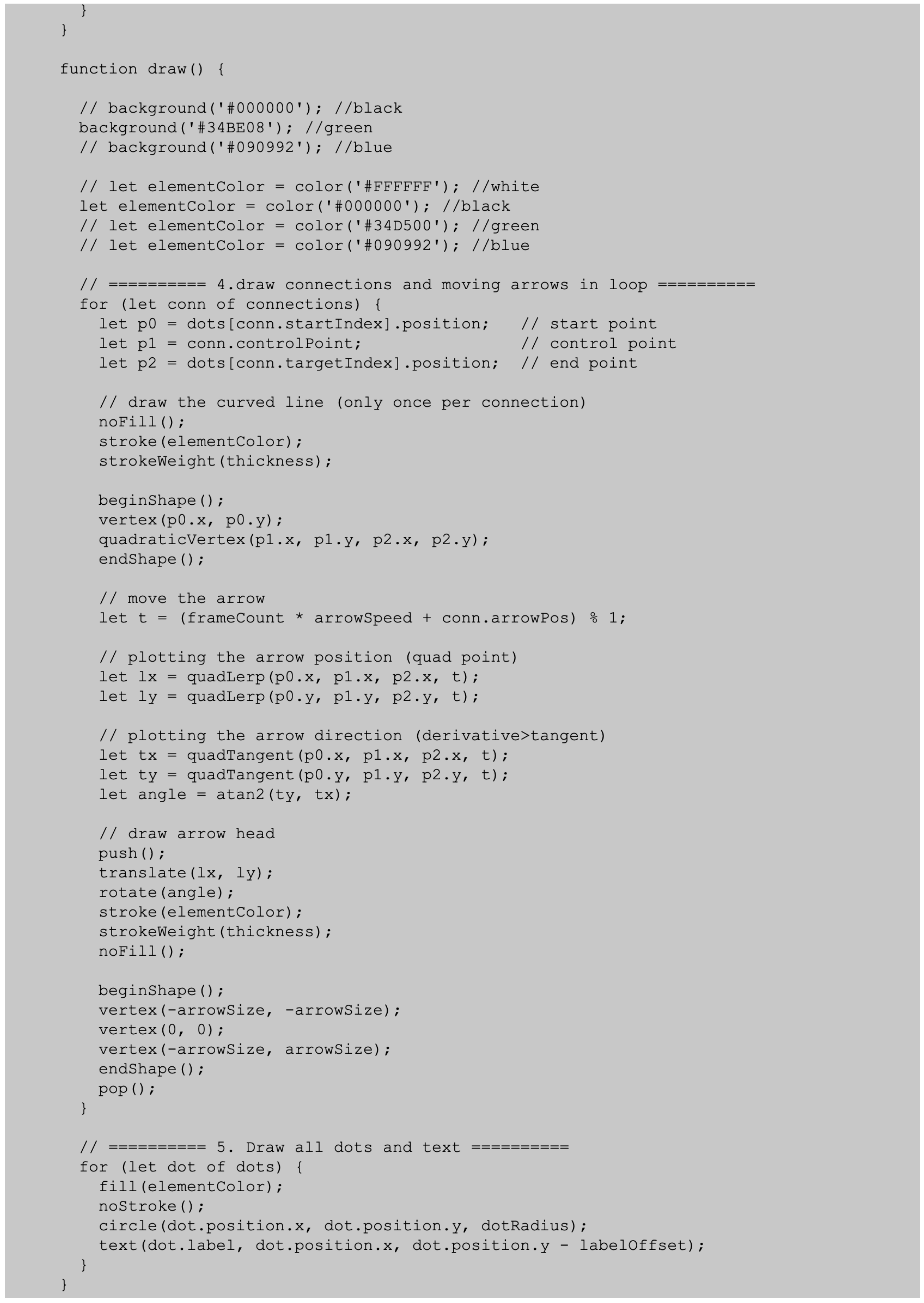
Readable Mode

I displayed the code from the replicated project to compare a more advanced version with the simpler first-iteration syntax, translate(). This code format is longer and looks more advanced than the example from translate(), but it still reads clearly because of the spacing and comments that explain each line.

Courier New



```
// ===== DOCUMENTATION =====
// REFERENCES & RESOURCES
// Selected project: https://www.instagram.com/p/DOeV-szEnZm/?hl=en&img_index=1
// Array of texts: https://timrodenbroeker.de/courses/basic-datastructures/my-five-favor
// Random point drag tutorial: https://www.youtube.com/watch?v=kjqo_uXn87I
// Documentation: https://www.w3schools.com/, https://developer.mozilla.org/en-US/
// Dot movement: https://www.youtube.com/watch?v=cH90Oadi1wY
// Force directed graph: https://www.youtube.com/watch?v=YskU_gJpc0c // no use
// Click and drag: https://www.youtube.com/watch?v=kjqo_uXn87I
// Network case study: https://www.patrik-huebner.com/datadesigndictionary/

let font;
function preload() {
  font = loadFont("assets/LibreCaslonText-Regular.ttf");
}

// Canvas setting
const margin = 70;

// Dot settings
const numDots = 10;
const dotRadius = 10;
const labelOffset = 4;

// Arrow settings
const arrowSize = 7;
const arrowSpeed = 0.009;

// Curve settings
const thickness = 1.5
const bendMin = 20;
const bendMax = 250;

// Data arrays
const dots = []; //assign array
const connections = [];
const labels = ["Us", "Me", "Them", "You", "Here", "There"];

function setup() {
  createCanvas(1080, 1032);
  textFont(font);
  textSize(21);
  textAlign(CENTER, BOTTOM);

  // ========== 1.Setup dots ==========
  // .push() method adds one or more elements to the end of an array and returns the new
  // dotIndex=the number used to find a dot in the array

  for (let i = 0; i < numDots; i++) {
    dots.push({
      position: createVector(
        random(margin, width - margin),
        random(margin, height - margin)
      ),
      label: labels[i % labels.length], // assign array
    });
  }

  // ========== 2.Setup connection counts ==========
  // 20 base + 8 extra = 28
  // each dot starts with 2 outgoing connections
```



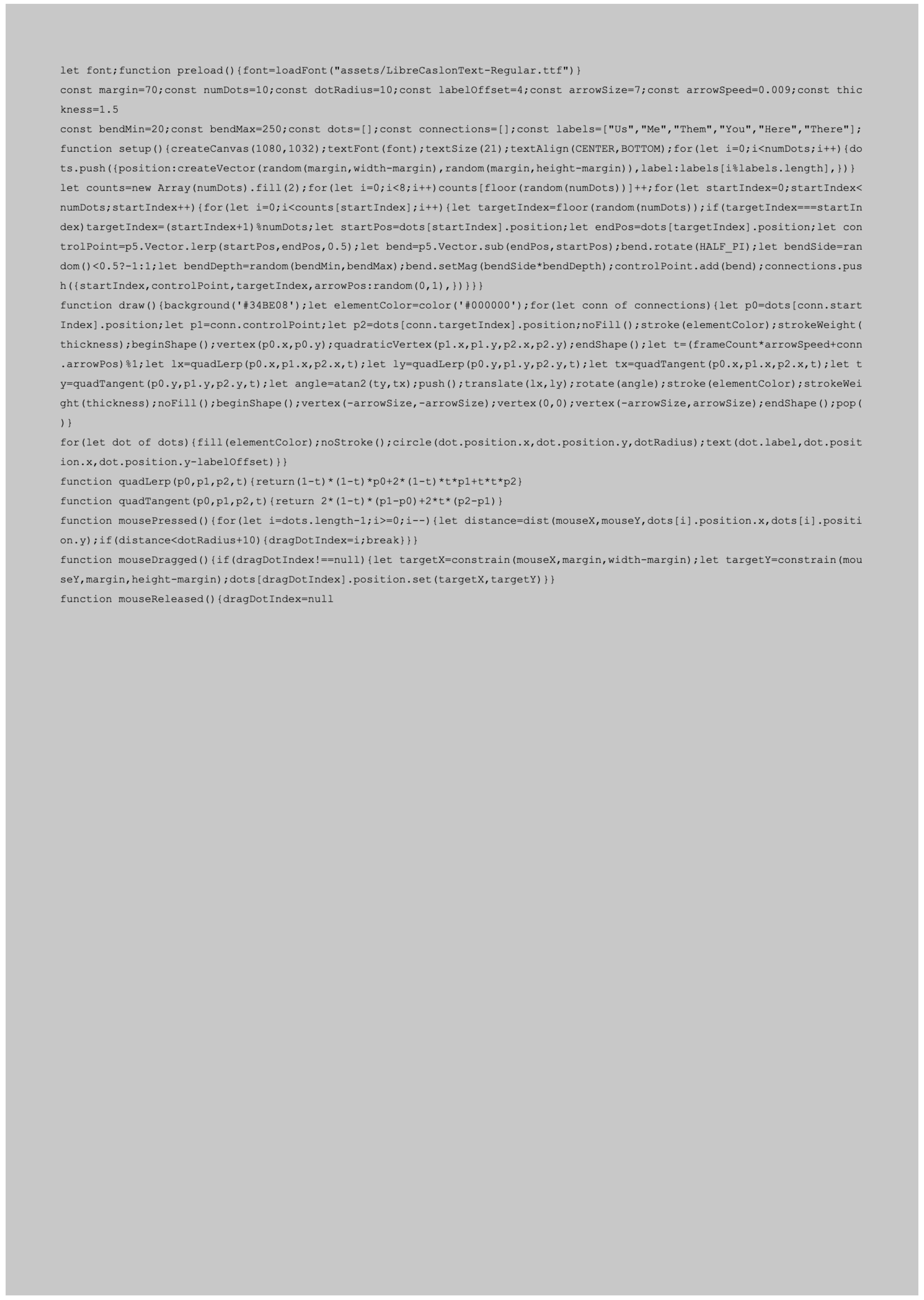
```
    }
}

function draw() {

  // background('#000000'); //black
  background('#34BE08'); //green
  // background('#090992'); //blue

  // let elementColor = color('#FFFFFF'); //white
  let elementColor = color('#000000'); //black
  // let elementColor = color('#34D500'); //green
  // let elementColor = color('#090992'); //blue

  // ========== 4.draw connections and moving arrows in loop ==========
  for (let conn of connections) {
    let p0 = dots[conn.startIndex].position;   // start point
    let p1 = conn.controlPoint;                // control point
    let p2 = dots[conn.targetIndex].position;  // end point

    // draw the curved line (only once per connection)
    noFill();
    stroke(elementColor);
    strokeWeight(thickness);

    beginShape();
    vertex(p0.x, p0.y);
    quadraticVertex(p1.x, p1.y, p2.x, p2.y);
    endShape();

    // move the arrow
    let t = (frameCount * arrowSpeed + conn.arrowPos) % 1;

    // plotting the arrow position (quad point)
    let lx = quadLerp(p0.x, p1.x, p2.x, t);
    let ly = quadLerp(p0.y, p1.y, p2.y, t);

    // plotting the arrow direction (derivative>tangent)
    let tx = quadTangent(p0.x, p1.x, p2.x, t);
    let ty = quadTangent(p0.y, p1.y, p2.y, t);
    let angle = atan2(ty, tx);

    // draw arrow head
    push();
    translate(lx, ly);
    rotate(angle);
    stroke(elementColor);
    strokeWeight(thickness);
    noFill();

    beginShape();
    vertex(-arrowSize, -arrowSize);
    vertex(0, 0);
    vertex(-arrowSize, arrowSize);
    endShape();
    pop();
  }

  // ========== 5. Draw all dots and text ==========
  for (let dot of dots) {
    fill(elementColor);
    noStroke();
    circle(dot.position.x, dot.position.y, dotRadius);
    text(dot.label, dot.position.x, dot.position.y - labelOffset);
  }
}
```

The Iteration Set 04-2 / 04-3

Minified and Compressed Mode

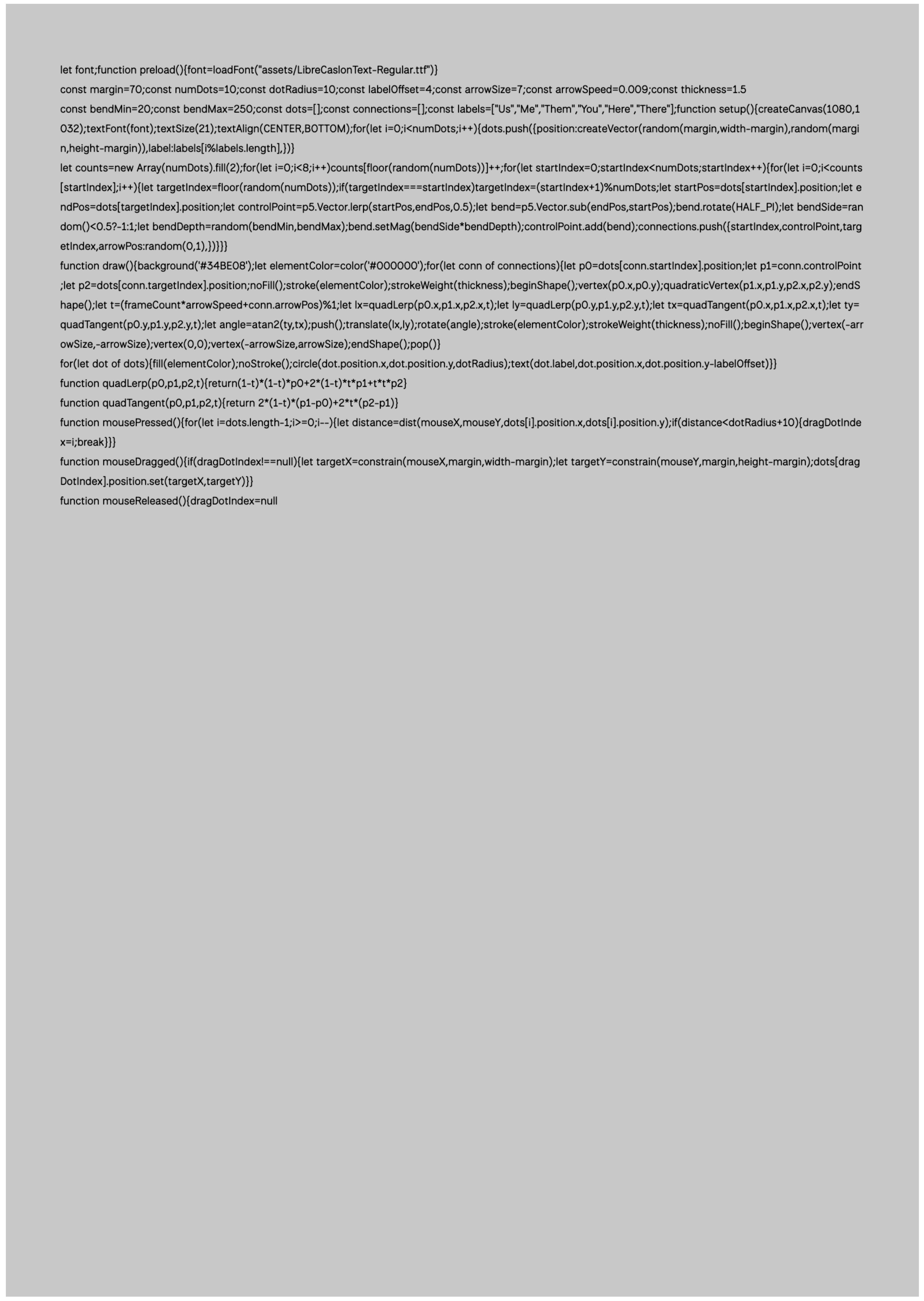
The next iteration is a minified version of the readable one to reduce data size and enhance processing speed. Resizing the font without adjusting the leading resulted in an interesting effect: it looked like a visual representation of a beginner's mindset, with a dense wall of technical language that made it harder for beginners to understand.

Courier New



```
let font;function preload(){font=loadFont("assets/LibreCaslonText-Regular.ttf")}
const margin=70;const numDots=10;const dotRadius=10;const labelOffset=4;const arrowSize=7;const arrowSpeed=0.009;const thic
kness=1.5
const bendMin=20;const bendMax=250;const dots=[];const connections=[];const labels=["Us","Me","Them","You","Here","There"];
function setup(){createCanvas(1080,1032);textFont(font);textSize(21);textAlign(CENTER,BOTTOM);for(let i=0;i<numDots;i++){do
ts.push({position:createVector(random(margin,width-margin),random(margin,height-margin)),label:labels[i%labels.length],})}
let counts=new Array(numDots).fill(2);for(let i=0;i<8;i++)counts[floor(random(numDots))]++;for(let startIndex=0;startIndex<
numDots;startIndex++){for(let i=0;i<counts[startIndex];i++){let targetIndex=floor(random(numDots));if(targetIndex===startIn
dex)targetIndex=(startIndex+1)%numDots;let startPos=dots[startIndex].position;let endPos=dots[targetIndex].position;let con
trolPoint=p5.Vector.lerp(startPos,endPos,0.5);let bend=p5.Vector.sub(endPos,startPos);bend.rotate(HALF_PI);let bendSide=ran
dom()<0.5?-1:1;let bendDepth=random(bendMin,bendMax);bend.setMag(bendSide*bendDepth);controlPoint.add(bend);connections.pus
h({startIndex,controlPoint,targetIndex,arrowPos:random(0,1),})}}}
function draw(){background('#34BE08');let elementColor=color('#000000');for(let conn of connections){let p0=dots[conn.start
Index].position;let p1=conn.controlPoint;let p2=dots[conn.targetIndex].position;noFill();stroke(elementColor);strokeWeight(
thickness);beginShape();vertex(p0.x,p0.y);quadraticVertex(p1.x,p1.y,p2.x,p2.y);endShape();let t=(frameCount*arrowSpeed+conn
.arrowPos)%1;let lx=quadLerp(p0.x,p1.x,p2.x,t);let ly=quadLerp(p0.y,p1.y,p2.y,t);let tx=quadTangent(p0.x,p1.x,p2.x,t);let t
y=quadTangent(p0.y,p1.y,p2.y,t);let angle=atan2(ty,tx);push();translate(lx,ly);rotate(angle);stroke(elementColor);strokeWei
ght(thickness);noFill();beginShape();vertex(-arrowSize,-arrowSize);vertex(0,0);vertex(-arrowSize,arrowSize);endShape();pop(
)}
for(let dot of dots){fill(elementColor);noStroke();circle(dot.position.x,dot.position.y,dotRadius);text(dot.label,dot.posit
ion.x,dot.position.y-labelOffset)}}
function quadLerp(p0,p1,p2,t){return(1-t)*(1-t)*p0+2*(1-t)*t*p1+t*t*p2}
function quadTangent(p0,p1,p2,t){return 2*(1-t)*(p1-p0)+2*t*(p2-p1)}
function mousePressed(){for(let i=dots.length-1;i>=0;i--){let distance=dist(mouseX,mouseY,dots[i].position.x,dots[i].positi
on.y);if(distance<dotRadius+10){dragDotIndex=i;break}}}
function mouseDragged(){if(dragDotIndex!==null){let targetX=constrain(mouseX,margin,width-margin);let targetY=constrain(mou
seY,margin,height-margin);dots[dragDotIndex].position.set(targetX,targetY)}}
function mouseReleased(){dragDotIndex=null
```

National Park
(p5.js's web font)



```
let font;function preload(){font=loadFont("assets/LibreCaslonText-Regular.ttf")}
const margin=70;const numDots=10;const dotRadius=10;const labelOffset=4;const arrowSize=7;const arrowSpeed=0.009;const thickness=1.5
const bendMin=20;const bendMax=250;const dots=[];const connections=[];const labels=["Us","Me","Them","You","Here","There"];function setup(){createCanvas(1080,1
032);textFont(font);textSize(21);textAlign(CENTER,BOTTOM);for(let i=0;i<numDots;i++){dots.push({position:createVector(random(margin,width-margin),random(margi
n,height-margin)),label:labels[i%labels.length],})}
let counts=new Array(numDots).fill(2);for(let i=0;i<8;i++)counts[floor(random(numDots))]++;for(let startIndex=0;startIndex<numDots;startIndex++){for(let i=0;i<counts
[startIndex];i++){let targetIndex=floor(random(numDots));if(targetIndex===startIndex)targetIndex=(startIndex+1)%numDots;let startPos=dots[startIndex].position;let e
ndPos=dots[targetIndex].position;let controlPoint=p5.Vector.lerp(startPos,endPos,0.5);let bend=p5.Vector.sub(endPos,startPos);bend.rotate(HALF_PI);let bendSide=ran
dom()<0.5?-1:1;let bendDepth=random(bendMin,bendMax);bend.setMag(bendSide*bendDepth);controlPoint.add(bend);connections.push({startIndex,controlPoint,targ
etIndex,arrowPos:random(0,1),})}}}
function draw(){background('#34BE08');let elementColor=color('#000000');for(let conn of connections){let p0=dots[conn.startIndex].position;let p1=conn.controlPoint
;let p2=dots[conn.targetIndex].position;noFill();stroke(elementColor);strokeWeight(thickness);beginShape();vertex(p0.x,p0.y);quadraticVertex(p1.x,p1.y,p2.x,p2.y);endS
hape();let t=(frameCount*arrowSpeed+conn.arrowPos)%1;let lx=quadLerp(p0.x,p1.x,p2.x,t);let ly=quadLerp(p0.y,p1.y,p2.y,t);let tx=quadTangent(p0.x,p1.x,p2.x,t);let ty=
quadTangent(p0.y,p1.y,p2.y,t);let angle=atan2(ty,tx);push();translate(lx,ly);rotate(angle);stroke(elementColor);strokeWeight(thickness);noFill();beginShape();vertex(-arr
owSize,-arrowSize);vertex(0,0);vertex(-arrowSize,arrowSize);endShape();pop()}
for(let dot of dots){fill(elementColor);noStroke();circle(dot.position.x,dot.position.y,dotRadius);text(dot.label,dot.position.x,dot.position.y-labelOffset)}}
function quadLerp(p0,p1,p2,t){return(1-t)*(1-t)*p0+2*(1-t)*t*p1+t*t*p2}
function quadTangent(p0,p1,p2,t){return 2*(1-t)*(p1-p0)+2*t*(p2-p1)}
function mousePressed(){for(let i=dots.length-1;i>=0;i--){let distance=dist(mouseX,mouseY,dots[i].position.x,dots[i].position.y);if(distance<dotRadius+10){dragDotInde
x=i;break}}}
function mouseDragged(){if(dragDotIndex!==null){let targetX=constrain(mouseX,margin,width-margin);let targetY=constrain(mouseY,margin,height-margin);dots[drag
DotIndex].position.set(targetX,targetY)}}
function mouseReleased(){dragDotIndex=null
```

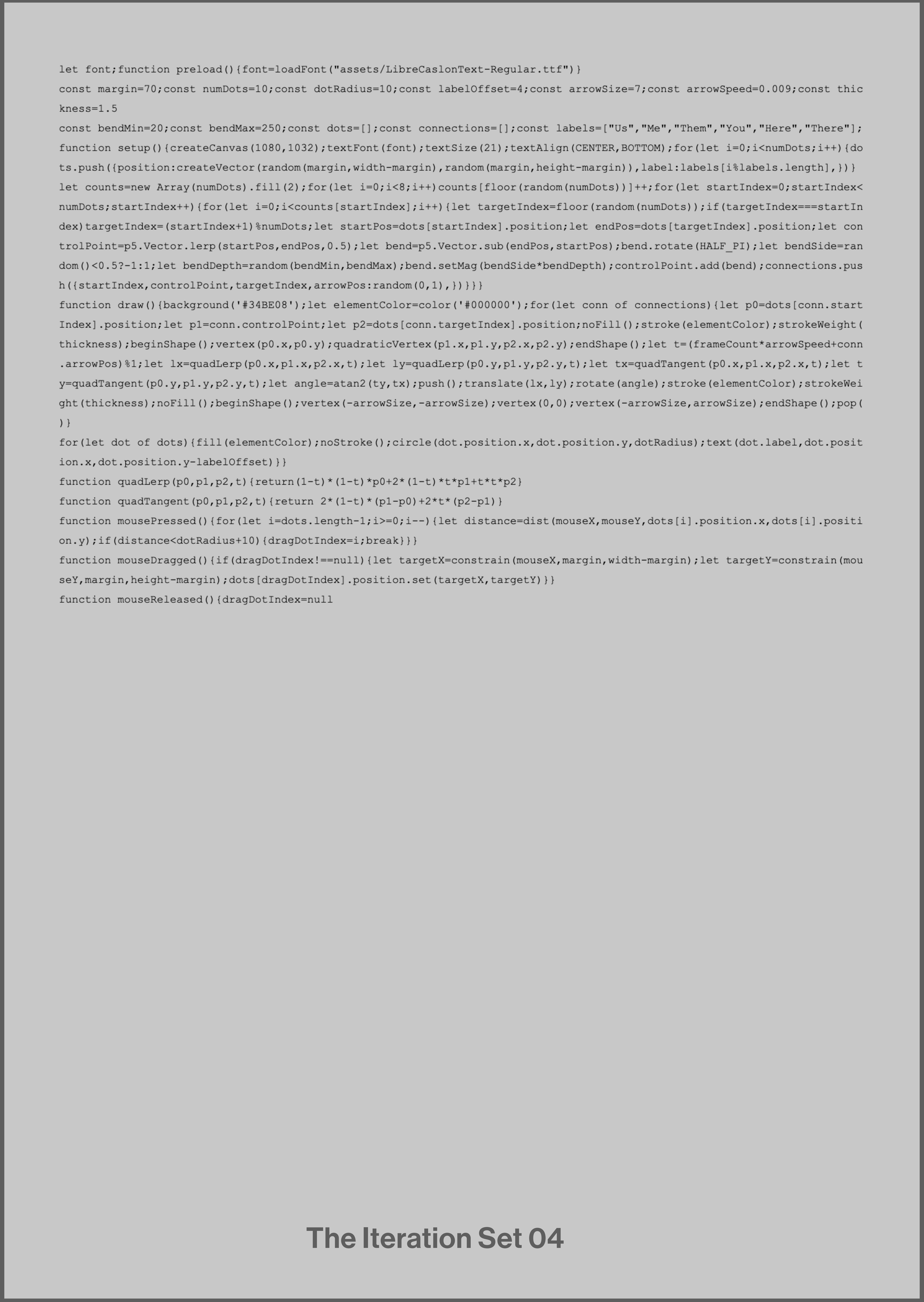
The Iteration Set

05

Overlaying
the iterations of
‘set 04’ using
Photoshop

Subvert:
Code

I used the same rules again with set 3, which is overlaying, and the result feels more chaotic and starting to resemble the primordial state. The more I overlay everything, the noisier it becomes. Code that used to be code now appears more like an ancient artefact or stone inscription. Eventually, it turns into a black page and noise.



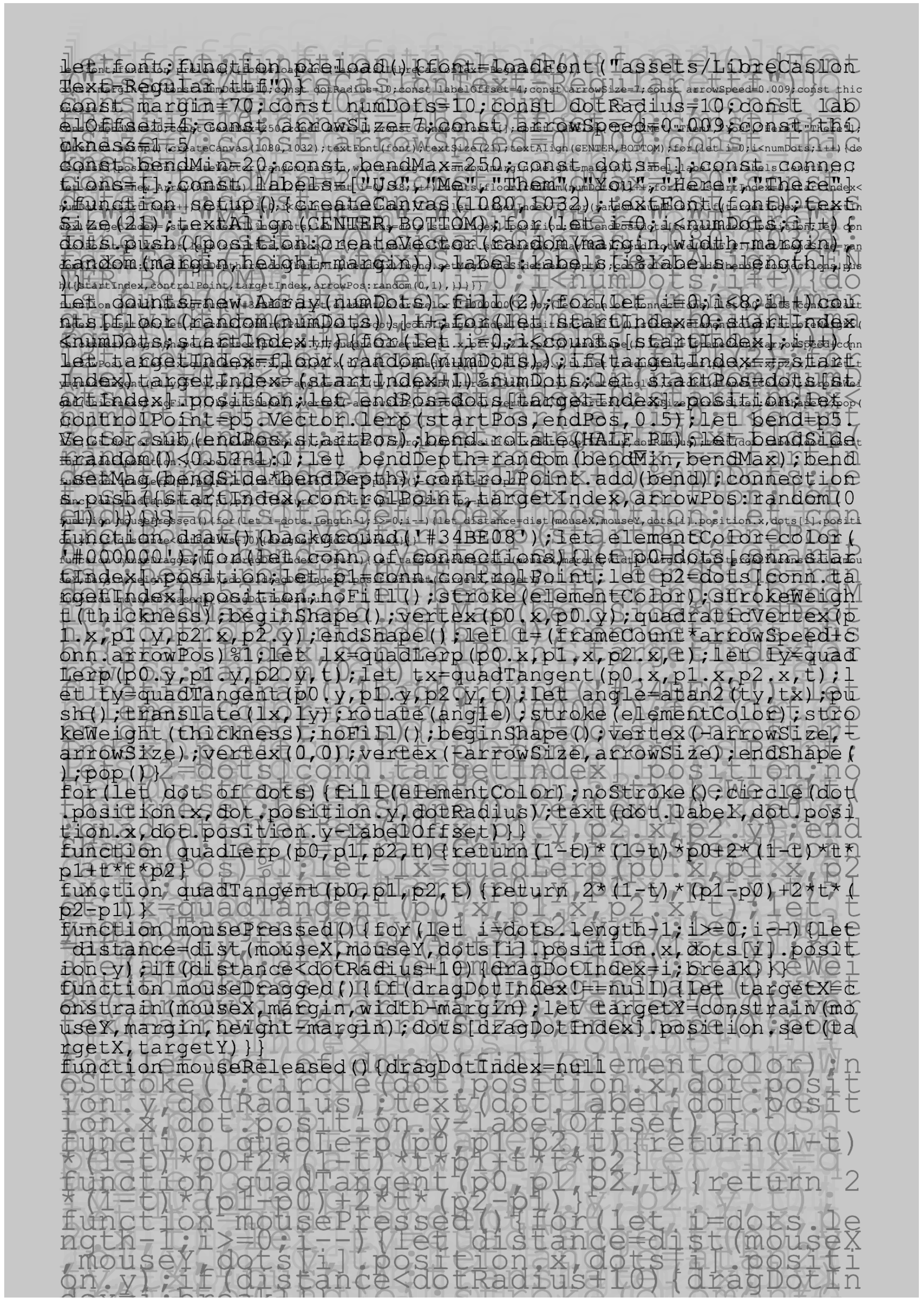
```
let font;function preload(){font=loadFont("assets/LibreCaslonText-Regular.ttf")}
const margin=70;const numDots=10;const dotRadius=10;const labelOffset=4;const arrowSize=7;const arrowSpeed=0.009;const thic
kness=1.5
const bendMin=20;const bendMax=250;const dots=[];const connections=[];const labels=["Us","Me","Them","You","Here","There"];
function setup(){createCanvas(1080,1032);textFont(font);textSize(21);textAlign(CENTER,BOTTOM);for(let i=0;i<numDots;i++){do
ts.push({position:createVector(random(margin,width-margin),random(margin,height-margin)),label:labels[i%labels.length],})}
let counts=new Array(numDots).fill(2);for(let i=0;i<8;i++)counts[floor(random(numDots))]++;for(let startIndex=0;startIndex<
numDots;startIndex++){for(let i=0;i<counts[startIndex];i++){let targetIndex=floor(random(numDots));if(targetIndex===startIn
dex)targetIndex=(startIndex+1)%numDots;let startPos=dots[startIndex].position;let endPos=dots[targetIndex].position;let con
trolPoint=p5.Vector.lerp(startPos,endPos,0.5);let bend=p5.Vector.sub(endPos,startPos);bend.rotate(HALF_PI);let bendSide=ran
dom()<0.5?-1:1;let bendDepth=random(bendMin,bendMax);bend.setMag(bendSide*bendDepth);controlPoint.add(bend);connections.pus
h({startIndex,controlPoint,targetIndex,arrowPos:random(0,1),})}}}
function draw(){background('#34BE08');let elementColor=color('#000000');for(let conn of connections){let p0=dots[conn.start
Index].position;let p1=conn.controlPoint;let p2=dots[conn.targetIndex].position;noFill();stroke(elementColor);strokeWeight(
thickness);beginShape();vertex(p0.x,p0.y);quadraticVertex(p1.x,p1.y,p2.x,p2.y);endShape();let t=(frameCount*arrowSpeed+conn
.arrowPos)%1;let lx=quadLerp(p0.x,p1.x,p2.x,t);let ly=quadLerp(p0.y,p1.y,p2.y,t);let tx=quadTangent(p0.x,p1.x,p2.x,t);let t
y=quadTangent(p0.y,p1.y,p2.y,t);let angle=atan2(ty,tx);push();translate(lx,ly);rotate(angle);stroke(elementColor);strokeWei
ght(thickness);noFill();beginShape();vertex(-arrowSize,-arrowSize);vertex(0,0);vertex(-arrowSize,arrowSize);endShape();pop(
)}
for(let dot of dots){fill(elementColor);noStroke();circle(dot.position.x,dot.position.y,dotRadius);text(dot.label,dot.posit
ion.x,dot.position.y-labelOffset)}}
function quadLerp(p0,p1,p2,t){return(1-t)*(1-t)*p0+2*(1-t)*t*p1+t*t*p2}
function quadTangent(p0,p1,p2,t){return 2*(1-t)*(p1-p0)+2*t*(p2-p1)}
function mousePressed(){for(let i=dots.length-1;i>=0;i--){let distance=dist(mouseX,mouseY,dots[i].position.x,dots[i].positi
on.y);if(distance<dotRadius+10){dragDotIndex=i;break}}}
function mouseDragged(){if(dragDotIndex!==null){let targetX=constrain(mouseX,margin,width-margin);let targetY=constrain(mou
seY,margin,height-margin);dots[dragDotIndex].position.set(targetX,targetY)}}
function mouseReleased(){dragDotIndex=null
```

The Iteration Set 04

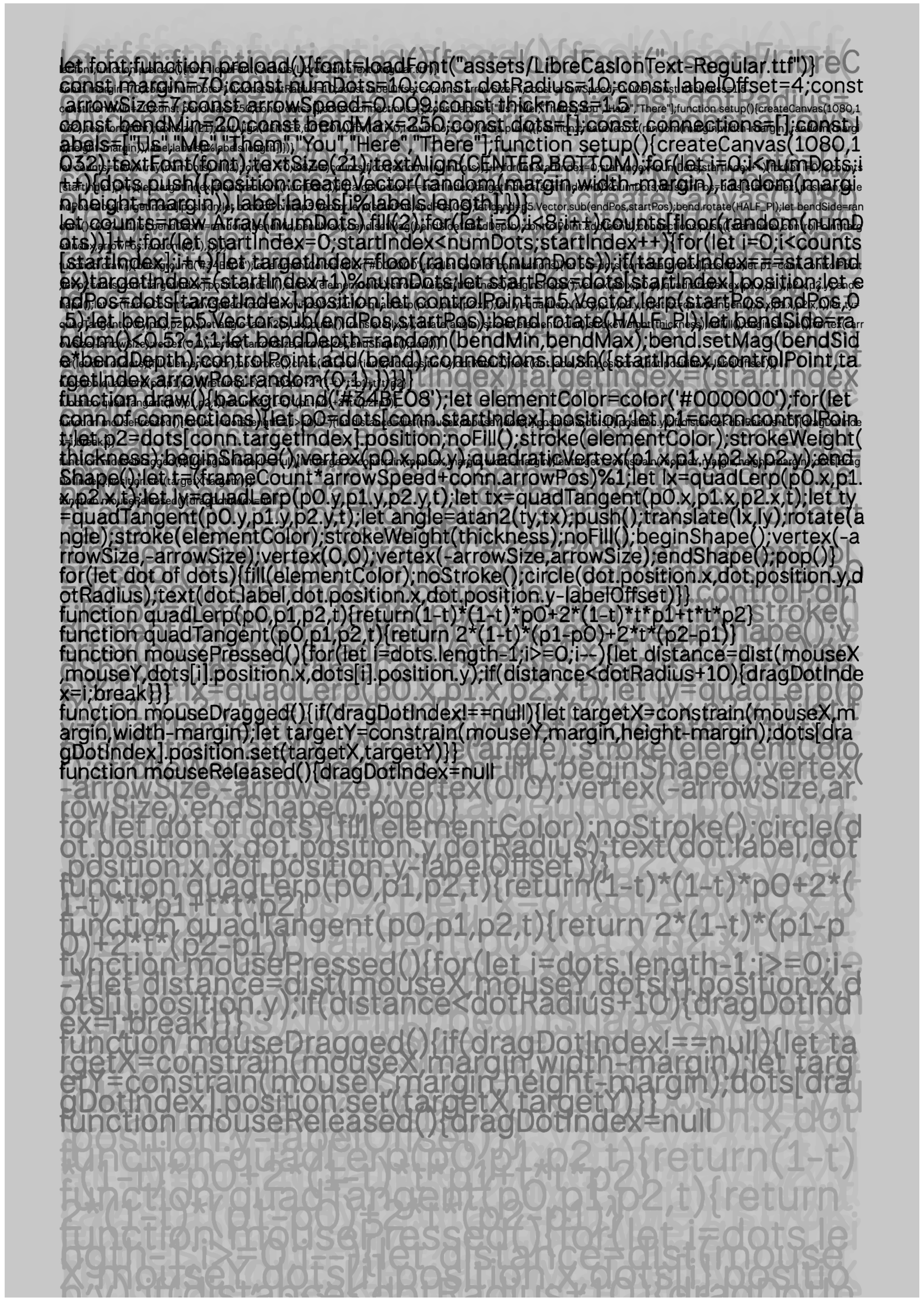
The Iteration Set 05

Blending Mode: Pin Light

Courier New



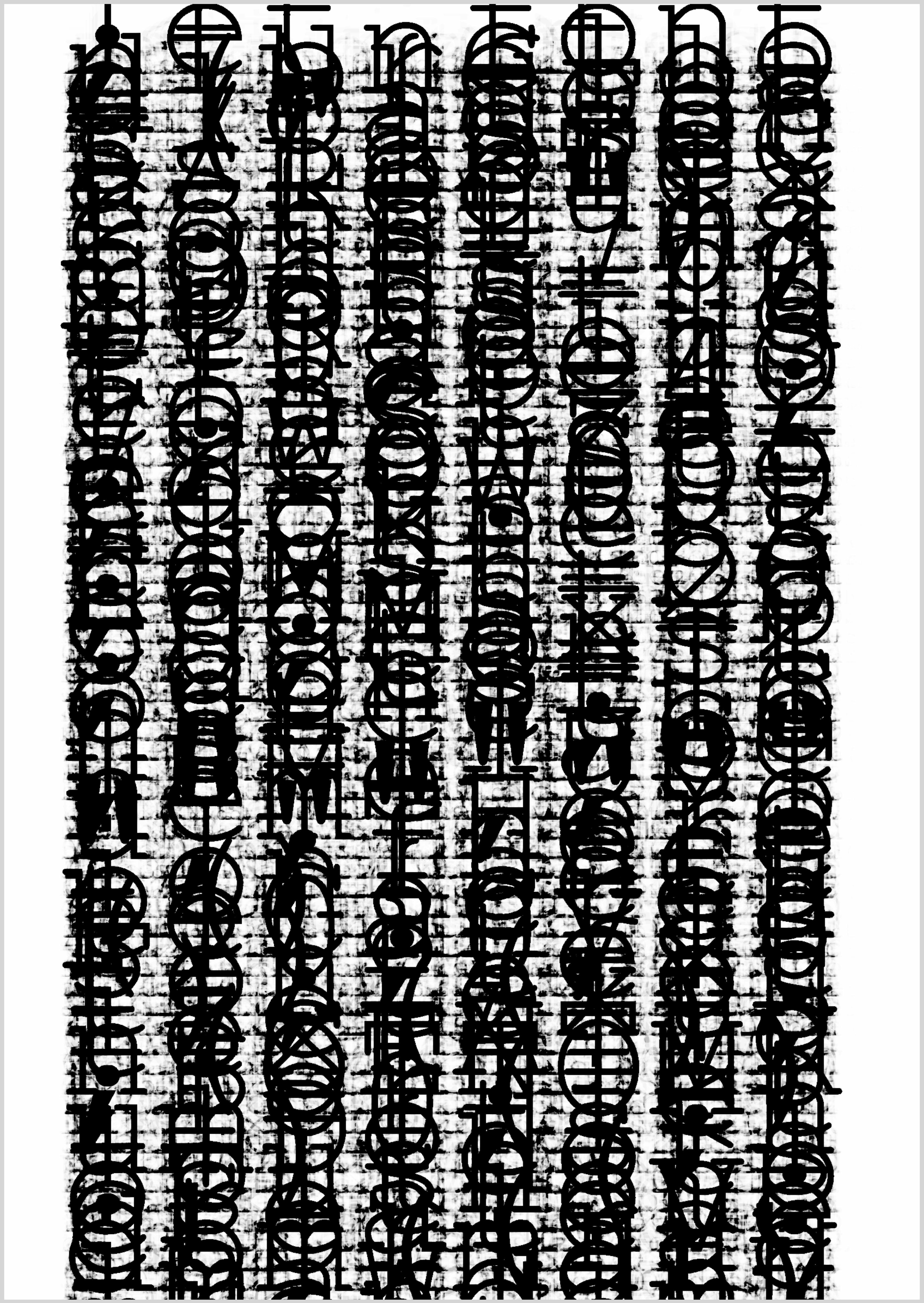
National Park (p5.js's web font)



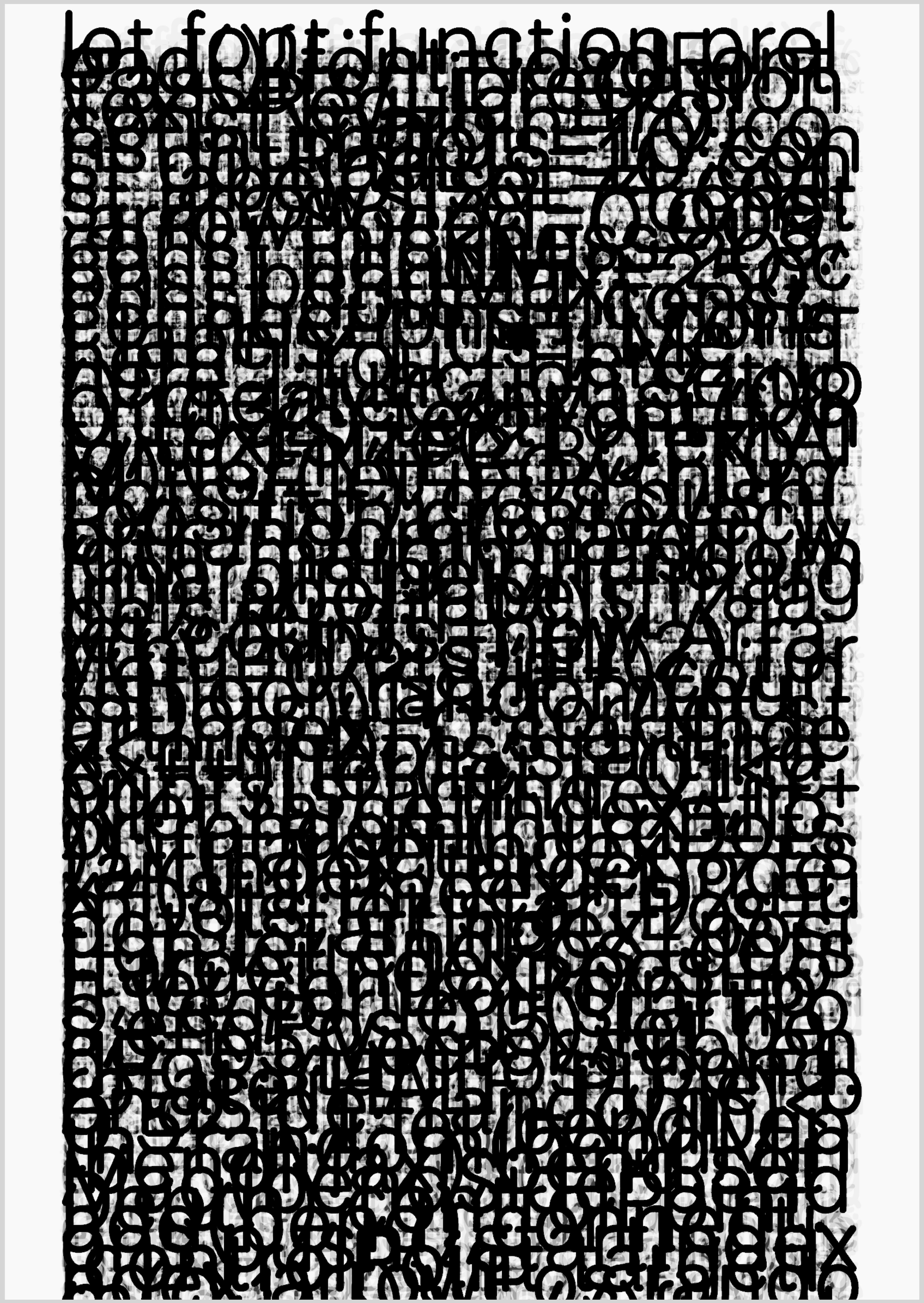
The Iteration Set 05

Blending Mode: Pin Light

Courier New



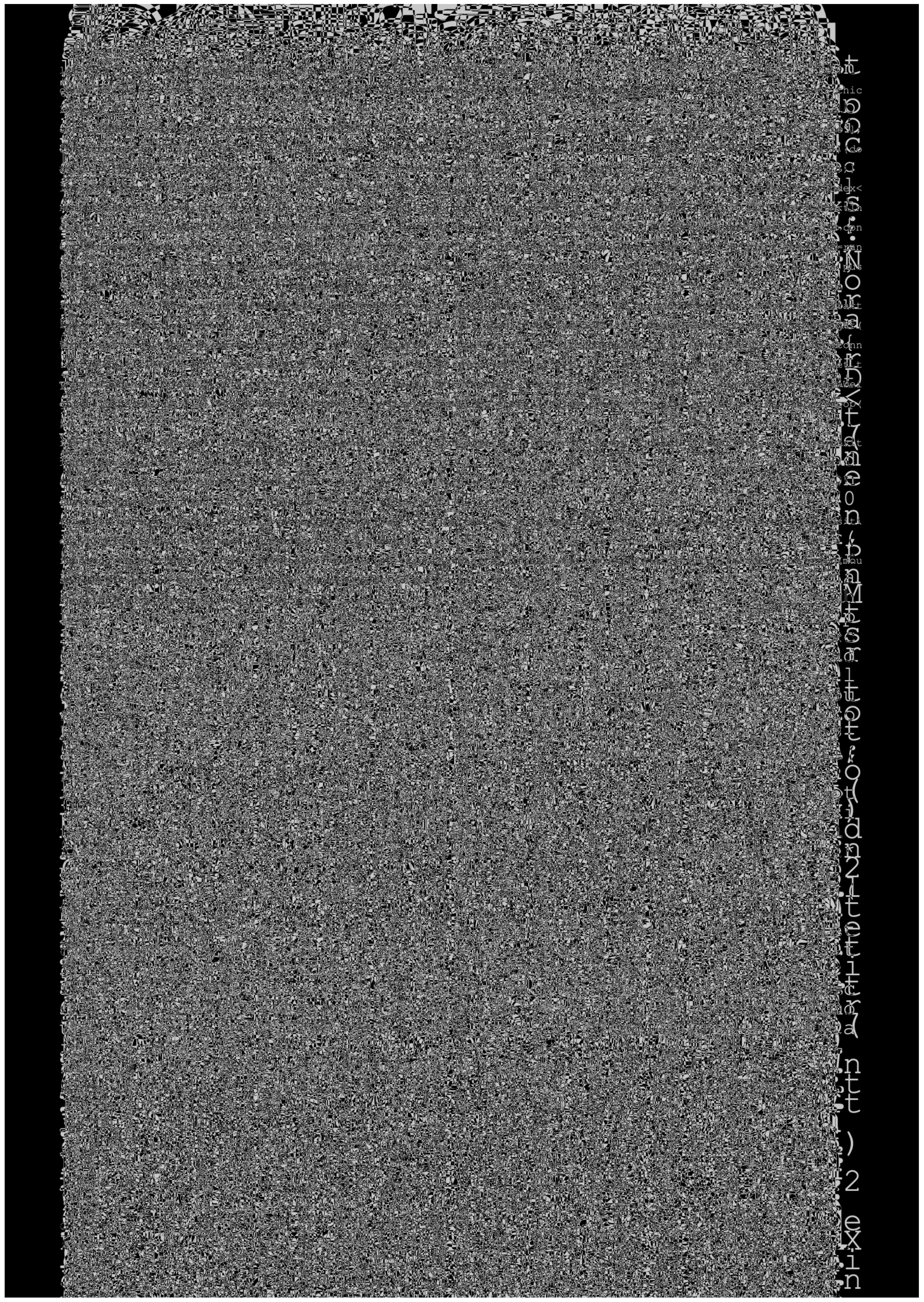
National Park (p5.js's web font)



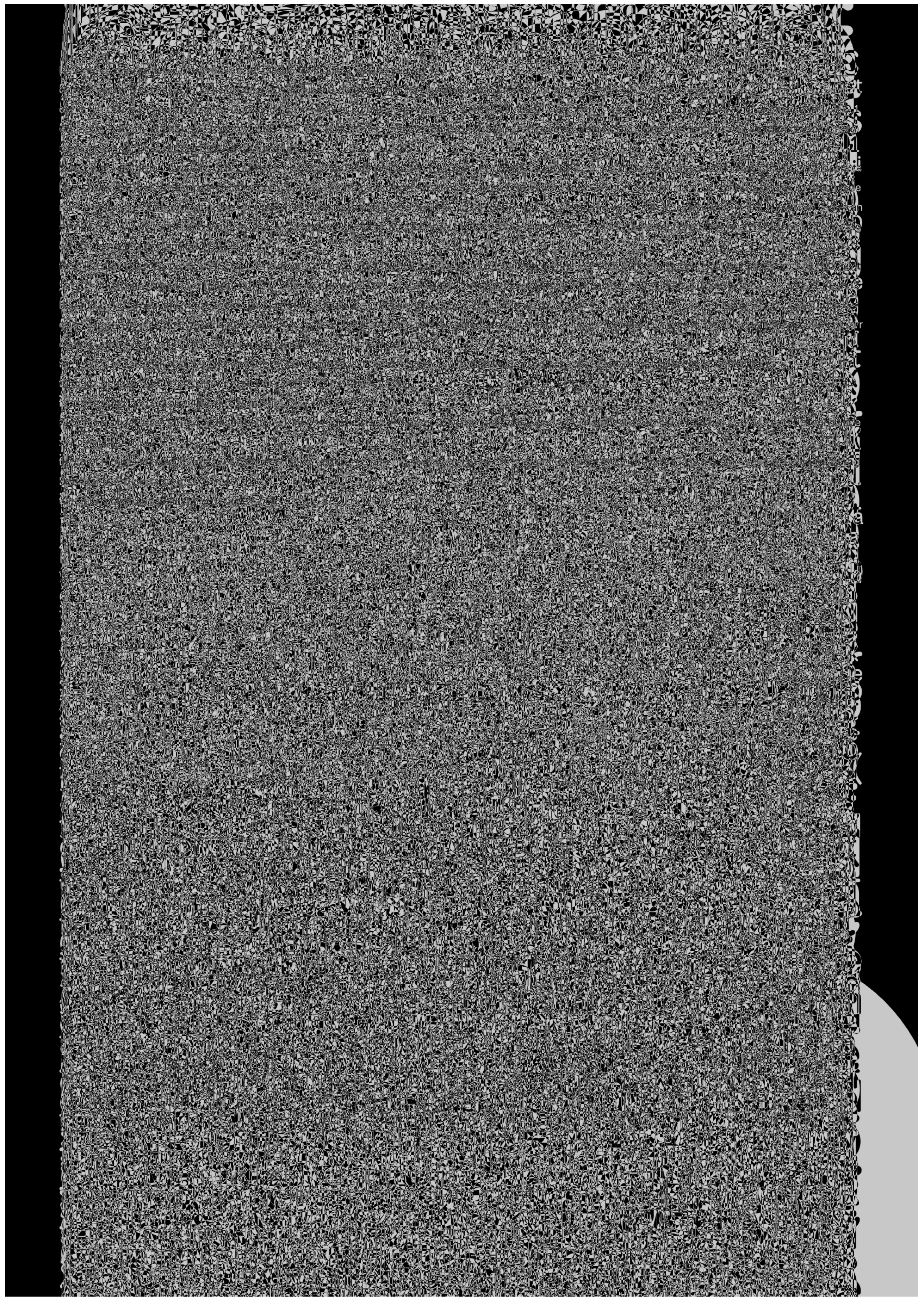
The Iteration Set 05

Blending Mode: Difference

Courier New



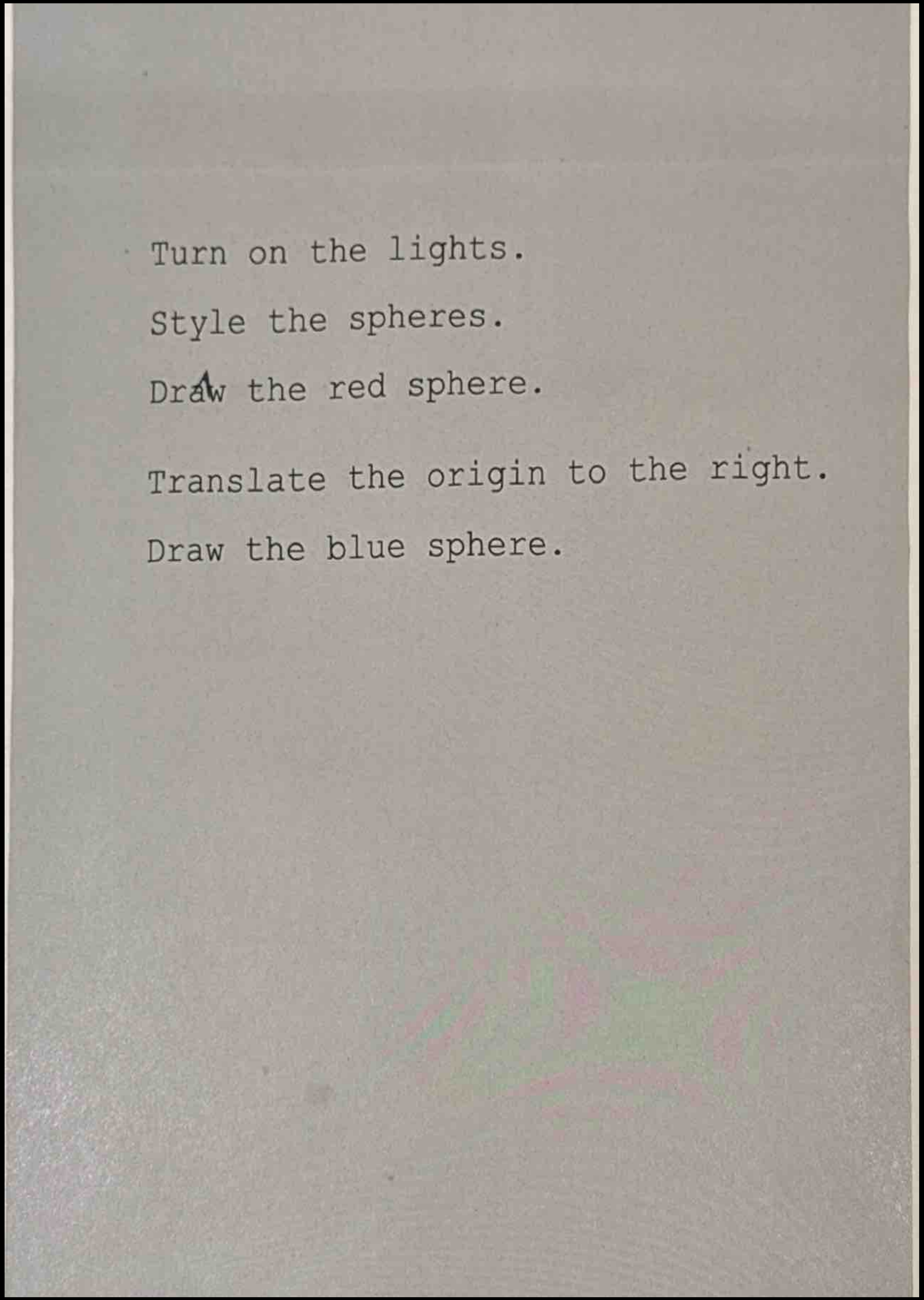
National Park (p5.js's web font)



The Iteration Set 00

At this point, I felt like I wanted to draw rather than use the computer anymore, so I drew following the instructions from my own deconstructed code. One of the earliest forms of human communication and record-keeping is drawing.

The instruction of deconstructed code



Turn on the lights.

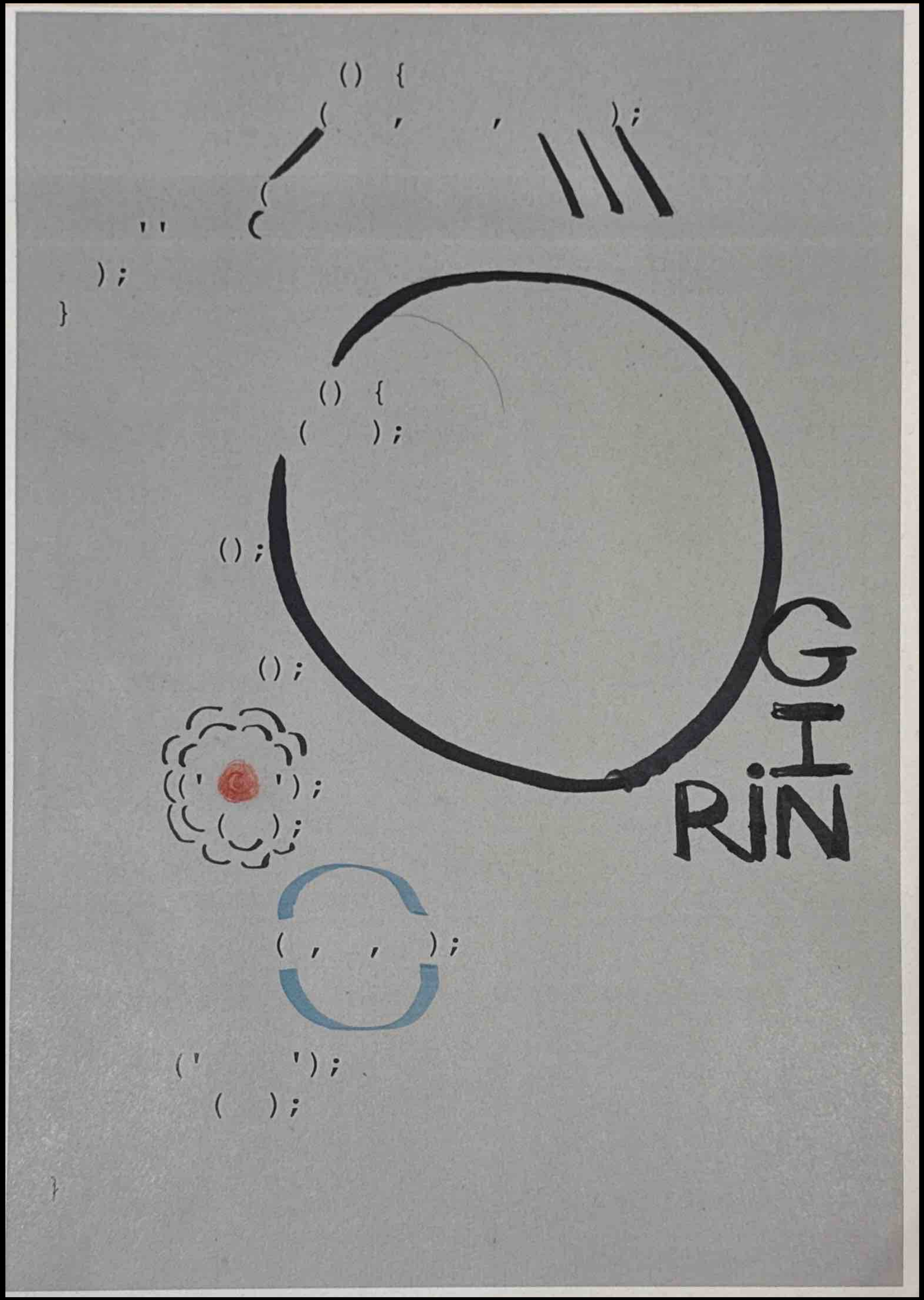
Style the spheres.

Draw the red sphere.

Translate the origin to the right.

Draw the blue sphere.

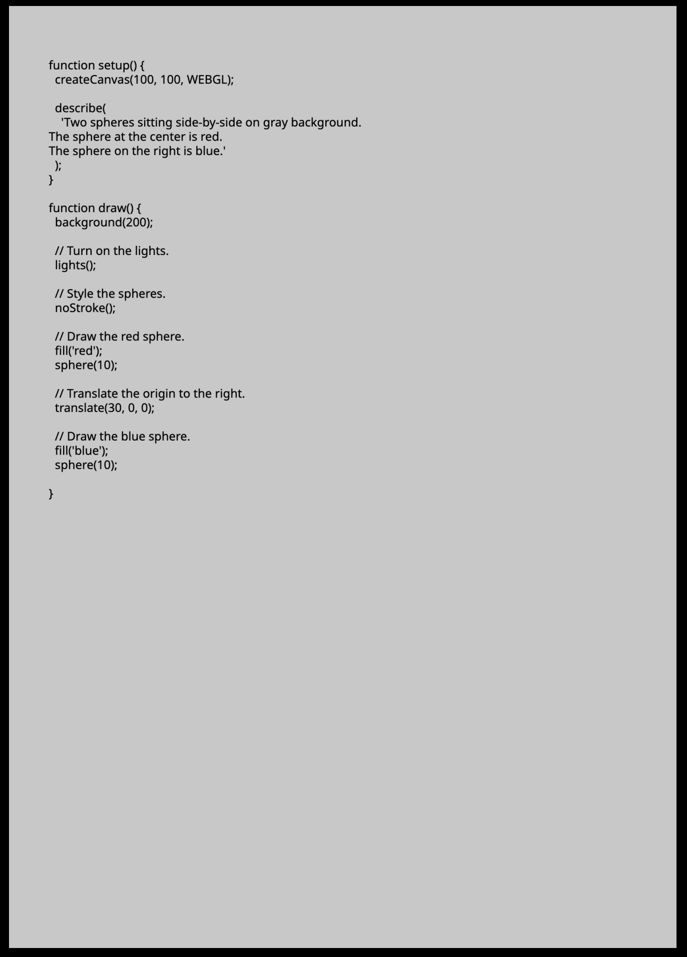
My drawing



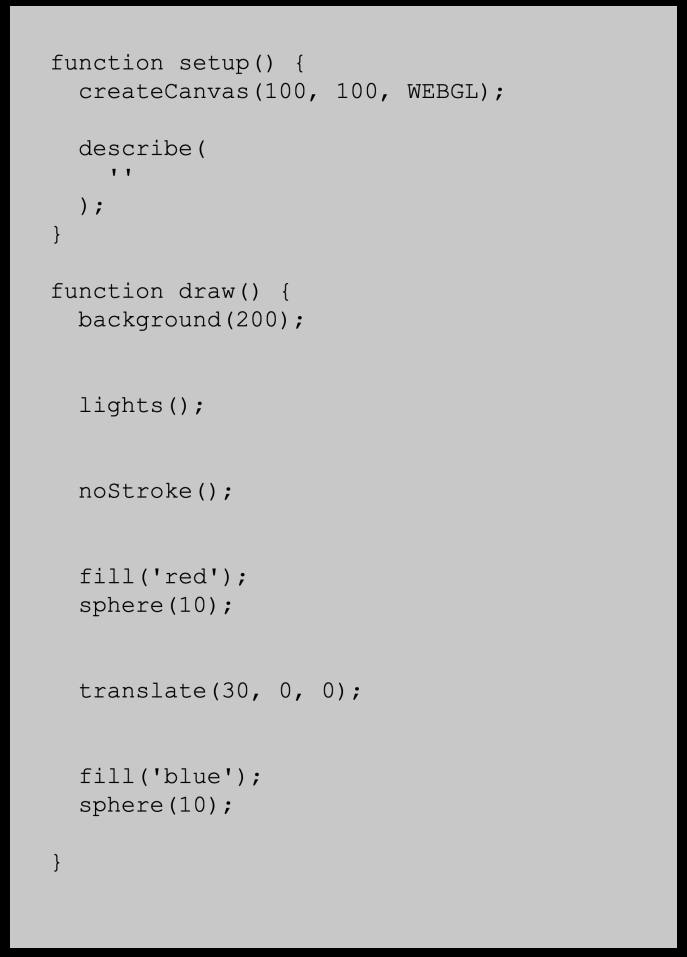
The Default Stage of Code

The Primordial Stage of Code

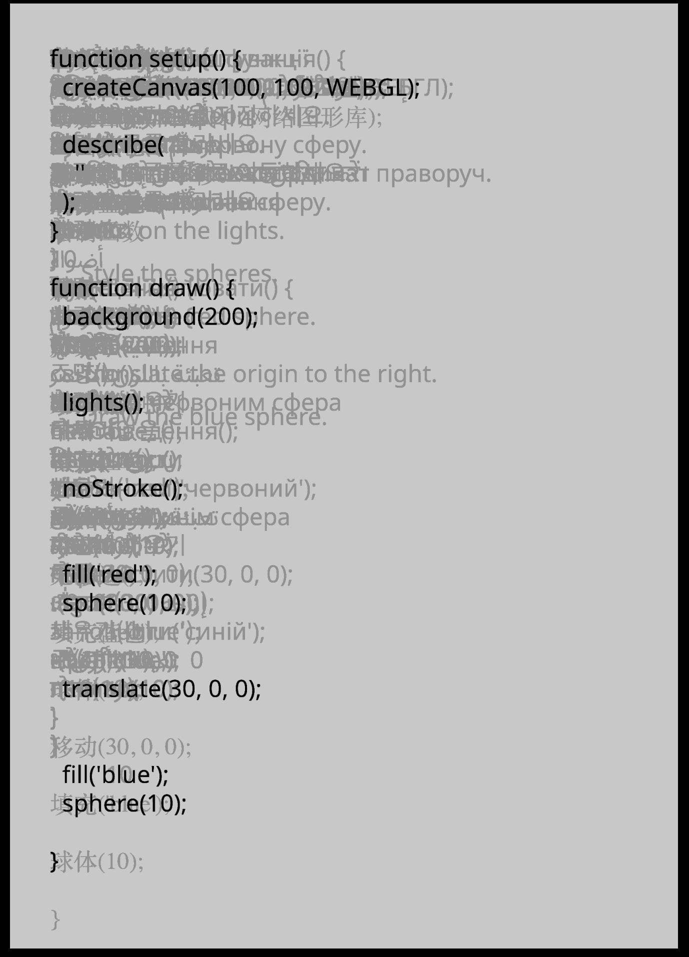
Organised ◄┈┈┈┈┈┈┈┈┈┈┈┈┈┈┈► Chaotic



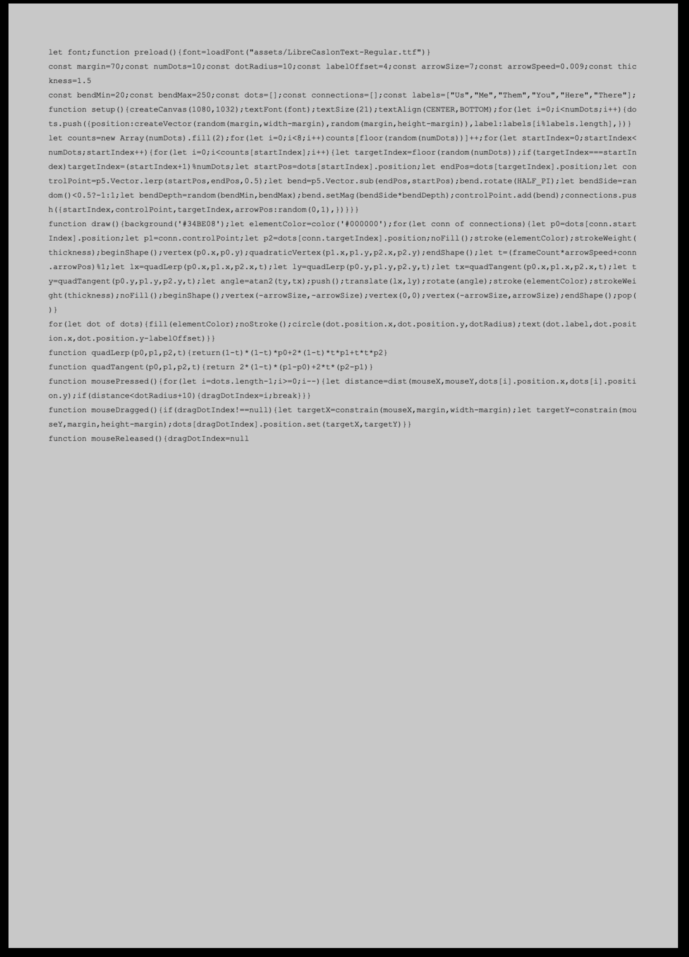
```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    'Two spheres sitting side-by-side on gray background.
The sphere at the center is red.
The sphere on the right is blue.'
  );
}

function draw() {
  background(200);

  // Turn on the lights.
  lights();

  // Style the spheres.
  noStroke();

  // Draw the red sphere.
  fill('red');
  sphere(10);

  // Translate the origin to the right.
  translate(30, 0, 0);

  // Draw the blue sphere.
  fill('blue');
  sphere(10);

}
```

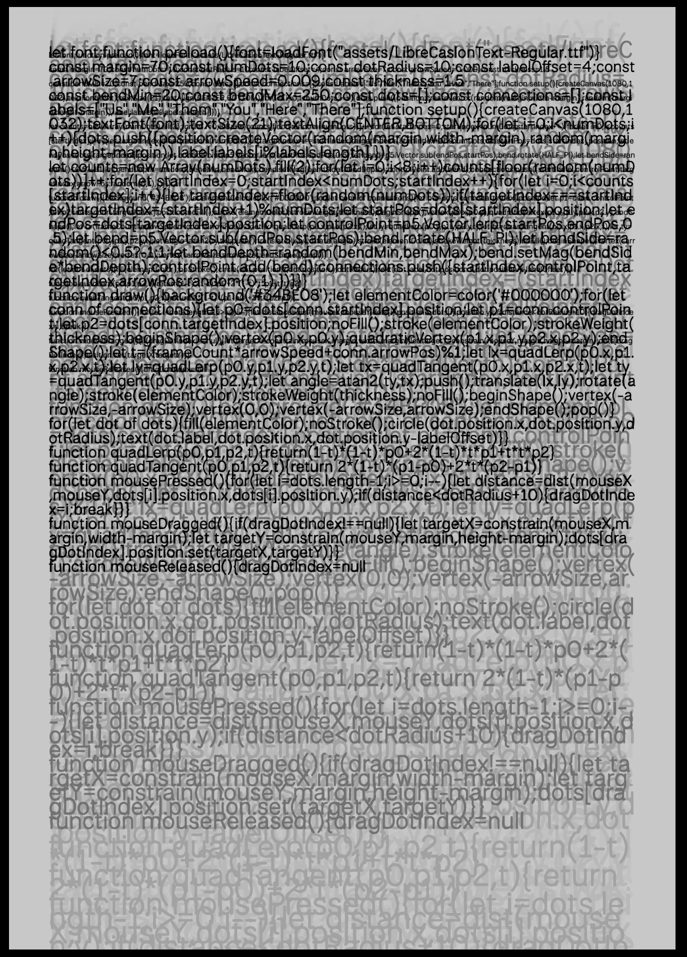


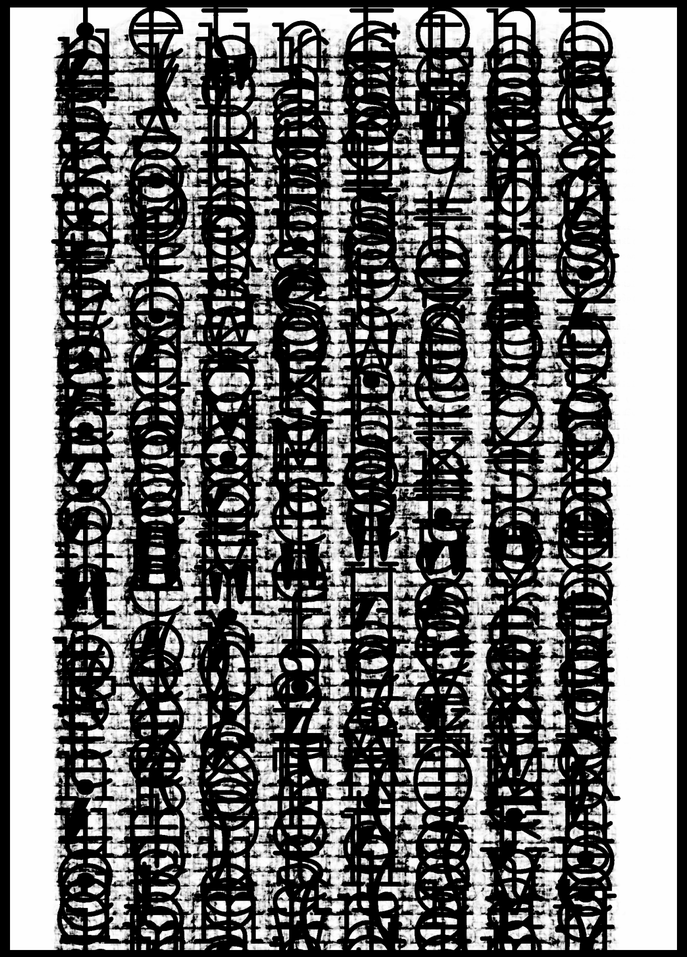
```
function setup() {
  createCanvas(100, 100, WEBGL);

  describe(
    ''
  );
}

function draw() {
  background(200);


  lights();


  noStroke();


  fill('red');
  sphere(10);


  translate(30, 0, 0);


  fill('blue');
  sphere(10);

}
```

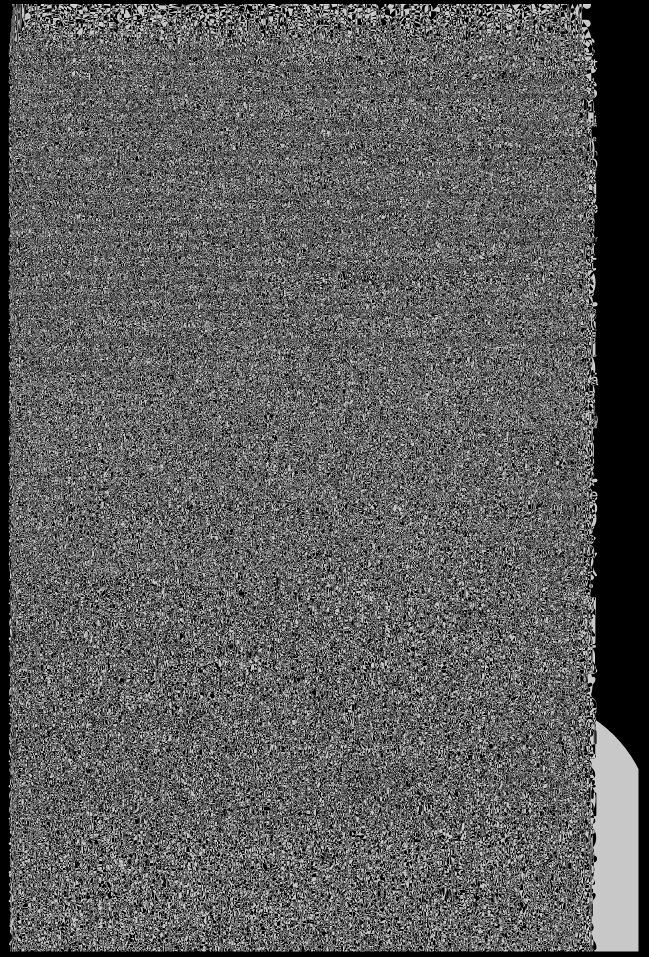


```
function setup() {
  createCanvas(100, 100, WEBGL);
  describe(
    ''
  );
}

function draw() {
  background(200);

  lights();

  noStroke();

  fill('red');
  sphere(10);

  translate(30, 0, 0);
}
  fill('blue');
  sphere(10);

}
```









Ultimately, I realised that the 'friendliness' is just a thin surface over complex, rigid code. Interestingly, what began as a simple, default approach developed into something with a more organic, ancient feel. As humans, we can’t be perfect with machines. I initially aimed for a perfect idea and possibility, but I became overwhelmed by this language. The way human intuition conflicts with this rigid, linear coding system creates a tension I want to explore further.