

SPECTRAL

CREATED BY LENA PANG

New Media Design +
Applied Mathematics
Capstone Project

Rochester Institute of Technology
Spring 2025
Independent Study

Project Summary

PROMPT

Combine math and design knowledge to research, visualize, and apply a topic in mathematics.

TIMELINE

1 semester (15 weeks)

CONCEPT

Create an interactive experience to demonstrate and visualize linear algebra concepts including linear transformations, eigenvectors, and eigenvalues.

DESIGNER INTRODUCTION

As an entirely self-directed, never-done-before project, the freedom and versatility was both a curse and a blessing. I had no reference point, no box to think outside of—but that meant that I had limitless options. I could do anything.

The inspiration for my final concept was born out of my passion for math, design, and teaching, and I hope it will be a helpful resource to make linear algebra more tangible, or, if nothing else, a cool tool to play around with.

SKILLS

UI/UX, web design, HTML, CSS, JavaScript, linear algebra (eigendecomposition)

Table of Contents

04 Ideation

- 05 Brainstorming
- 06 Inspiration
- 07 Concept

08 Content

- 09 Information Dilemmas
- 10 Defining a Voice
- 11 Segmentation
- 12 Visual Development

13 Code

- 14 The Beginnings: Processing
- 16 Transitioning to JavaScript
- 17 JavaScript Libraries
- 18 Programming Matrix Algebra

19 Final Design

- 20 Splash Screen
- 21 Menu
- 23 Main Pages
- 27 Subtle Animations

29 Style Guide

- 30 Color
- 31 Typography
- 32 Assets

33 Remarks

- 34 User Feedback
- 35 Designer Reflection

IDEATION

inspiration + brainstorming

Brainstorming

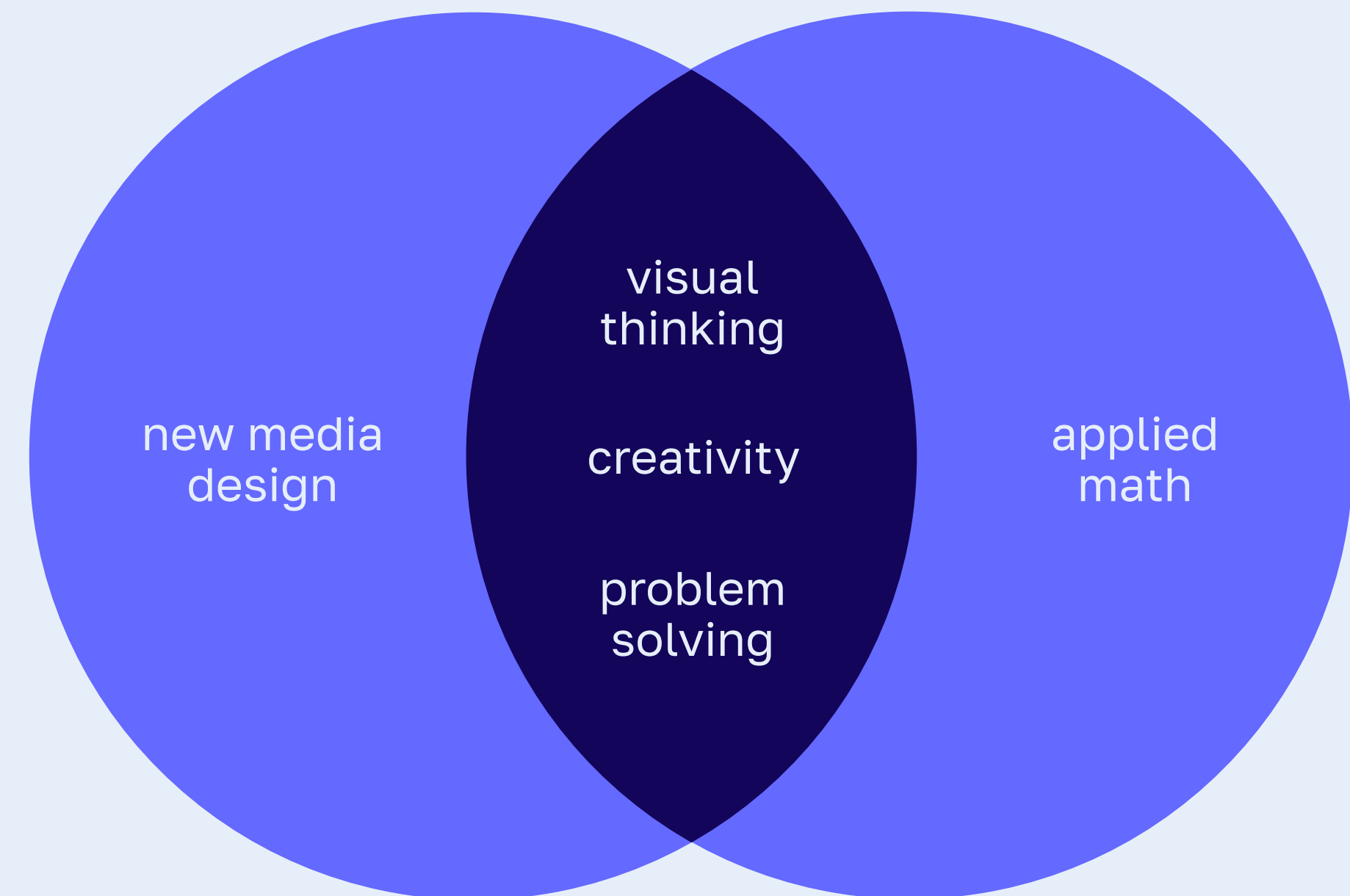
My capstone was the first of its kind at RIT, so I had little to no baseline or reference to work with: my design capstone had to have a bit of design. My math capstone had to have a bit of math. My math-design capstone would be... anything with a bit of both?

The world was my oyster.

My original capstone pitch had a bunch of visual math topics I had interest in researching:

- fractals
- paper folding
- animation interpolation
- cymatics
- chaotic attractors
- eigensystems

I began delving into each topic, but nothing really spoke to me.





Inspiration

I finally struck lightning in my numerical linear algebra class, when my teacher was talking about spectral radius. He asked if we could picture it, and he said it was impossible. It was a trick question.

Right? Well. Here we are.

(Okay, it's not exaaactly a counter example, but it's close.)

The Concept

When linear algebra is taught in schools, there is little emphasis on the visual aspect. Words like eigenvectors, eigenvalues, null space—they're assigned equations, but for most people, that means basically nothing.

My goal was to **communicate linear algebra concepts through interactive visualizations** that are intuitive and memorable, helping the user put physical, dimensional meaning to the numbers and variables they learn in class.

I also have personal investment helping people with linear algebra: it's one of my favorite subjects, and comes somewhat naturally to me because I can visualize it.

I enjoy helping others better understand the beauty and elegance of the subject, and I've tutored friends on several occasions.

CONTENT

layout + interactions + segmentation

Information Dilemmas

While I did not intend for the site to be thorough enough to constitute a standalone course, I did want to introduce some of the basics to help the user to grasp exactly what they are interacting with.

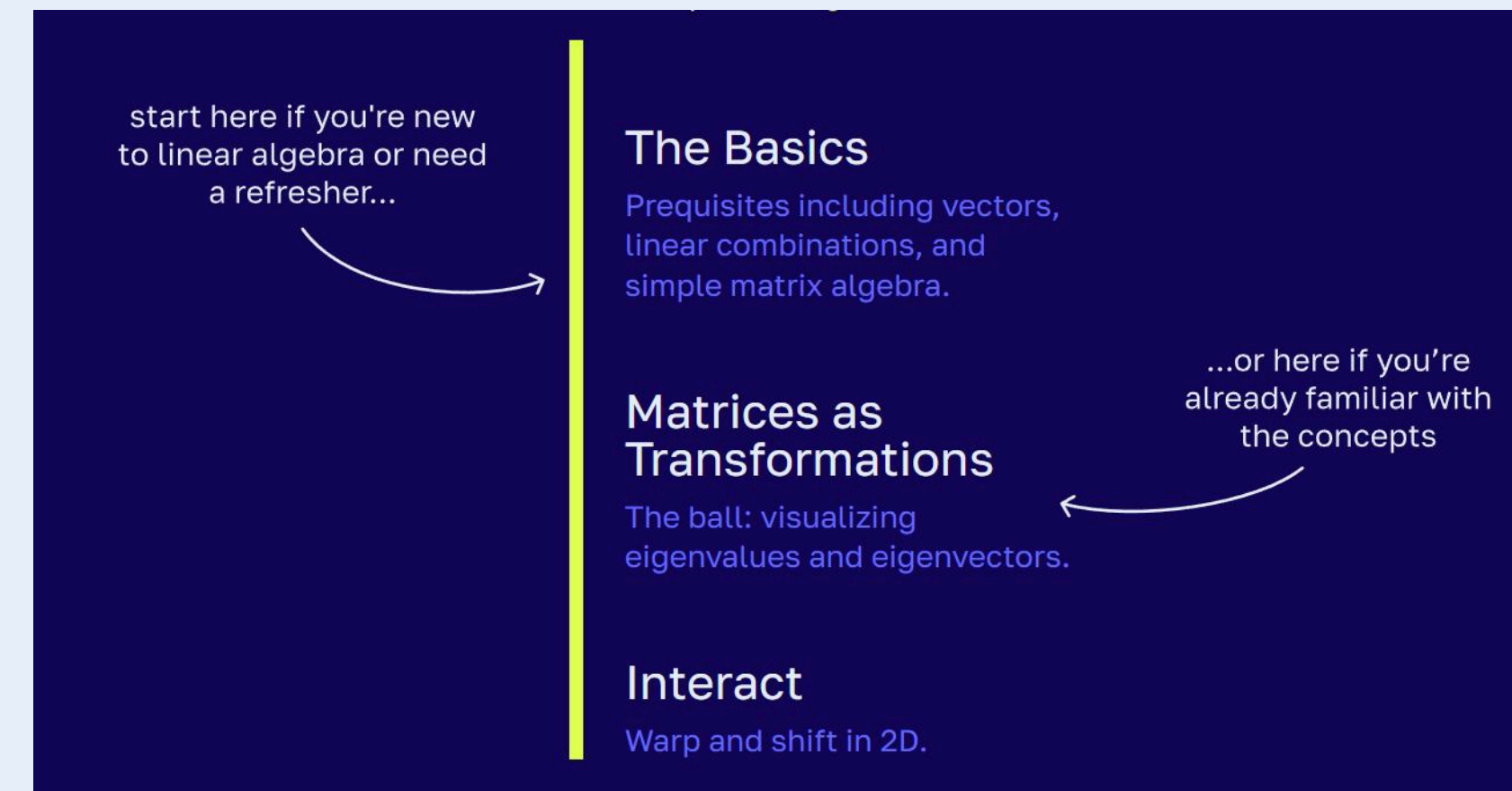
But that meant that I had to essentially make write and design my own linear algebra crash course. This raised many questions.

- How much information should I include? What is necessary and what's not?
 - Casual vs formal definitions?
 - Graphics?
 - Derivations?
 - Proofs?
 - Examples? If so, how many?
- How should I group and divide the information?
 - How many pages?
 - How to segment content within each page?

Defining a Voice

One of my first courses of action was defining a voice: a persona that speaks to the user through definitions and explanations.

I ultimately decided on a tone that is casual, direct, even sardonic at times, talking informally to—almost with—the user. The resulting effect is more approachable than, say, a textbook.



The **identity matrix** (0 everywhere except 1s on the diagonal) does nothing to the vector. Useless.

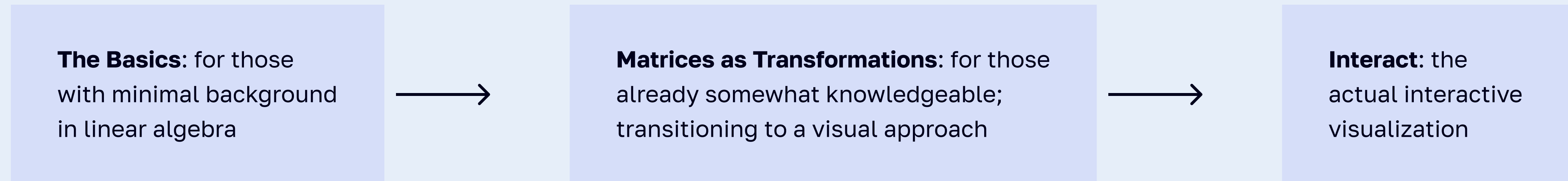
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Now try moving the dot around.

By scaling the 1s, we can manipulate how the matrix transforms our vector.

Segmentation

I ultimately divided my content by level of knowledge:

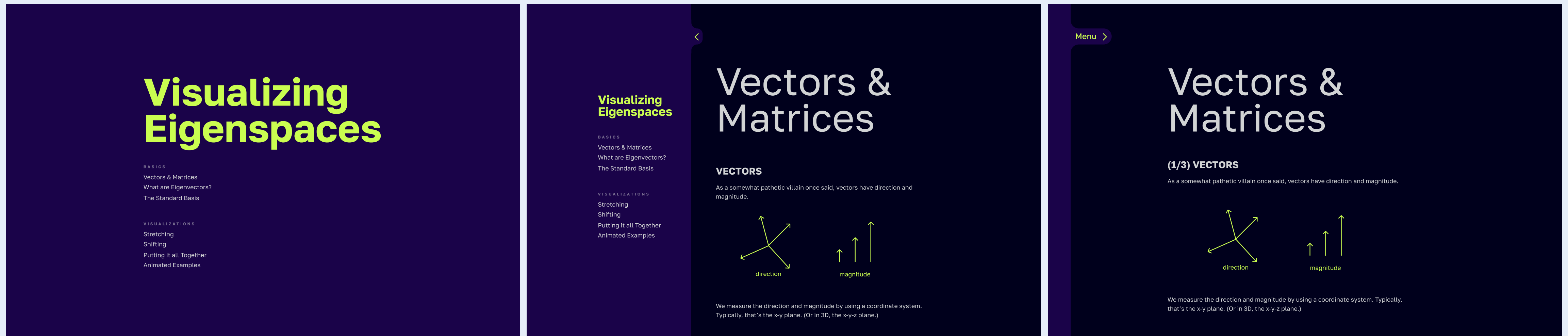


Within each page, there are sections based on vocabulary or topics. Most of the math is casual (no proofs), and the more overwhelming and thorough explanations are hidden in dropdown boxes if the user wants to learn more in-depth. This keeps each page relatively clean and concise.

Visual Development

As you can tell, my layout stayed somewhat consistent, but my content (title, segmentation, text, graphics) changed drastically. Much of the layout changed to fit this change in body copy.

I modified my color palette to increase readability, changing the background to a pale blue-grey, and made my colors more saturated for more a more vivid visual identity.



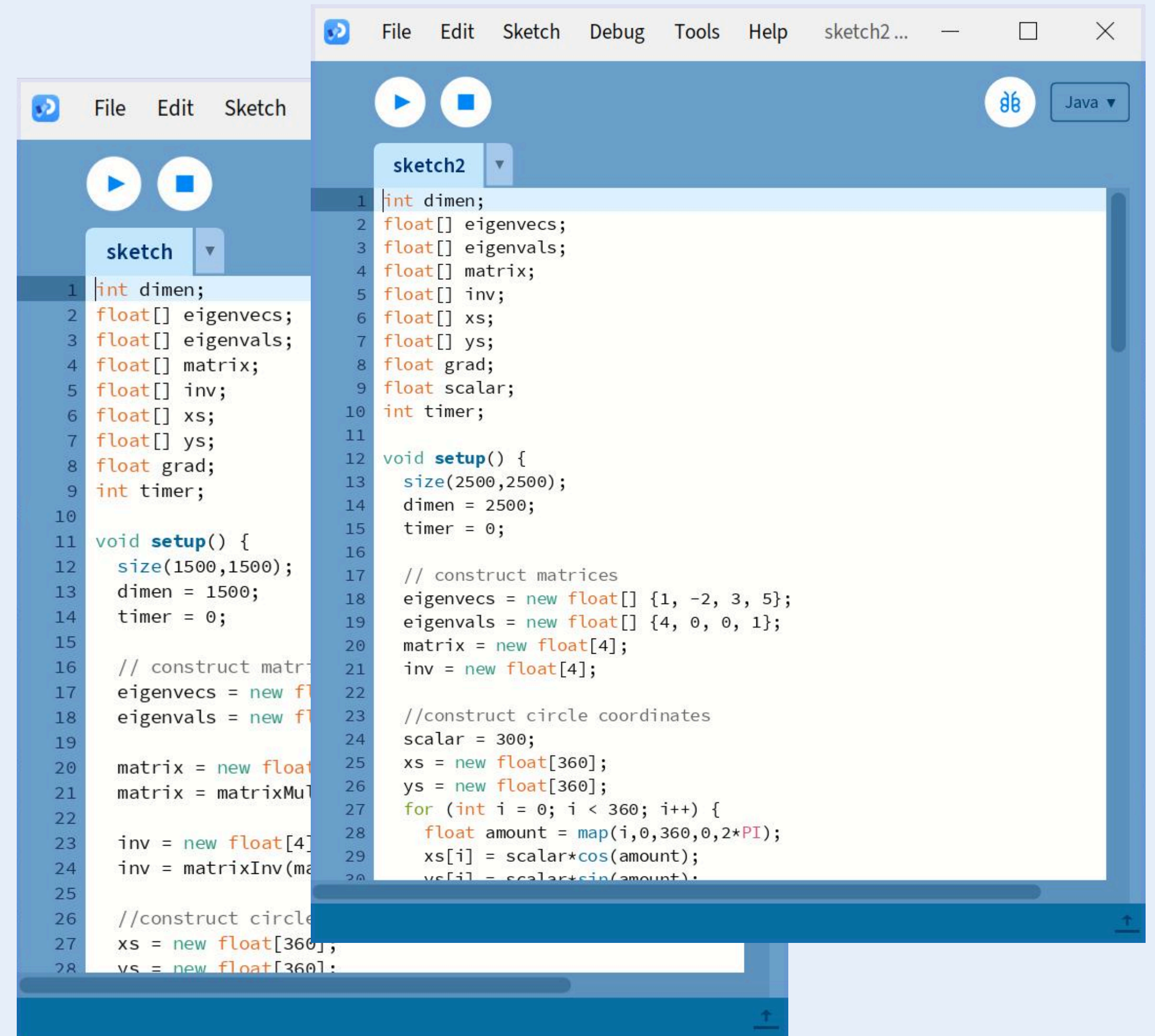
CODE

processing + javascript + web development

The Beginnings: Processing

My first explorations for the interactive page were written in Processing—this was a whole conundrum in itself, what with needing to declare variables and functions by fixed type, specifying everything.

But the hassle, combined with my unwillingness to do more work than I have to, resulted in some clever use of custom classes and functions to make calculations and visualizations more efficient.

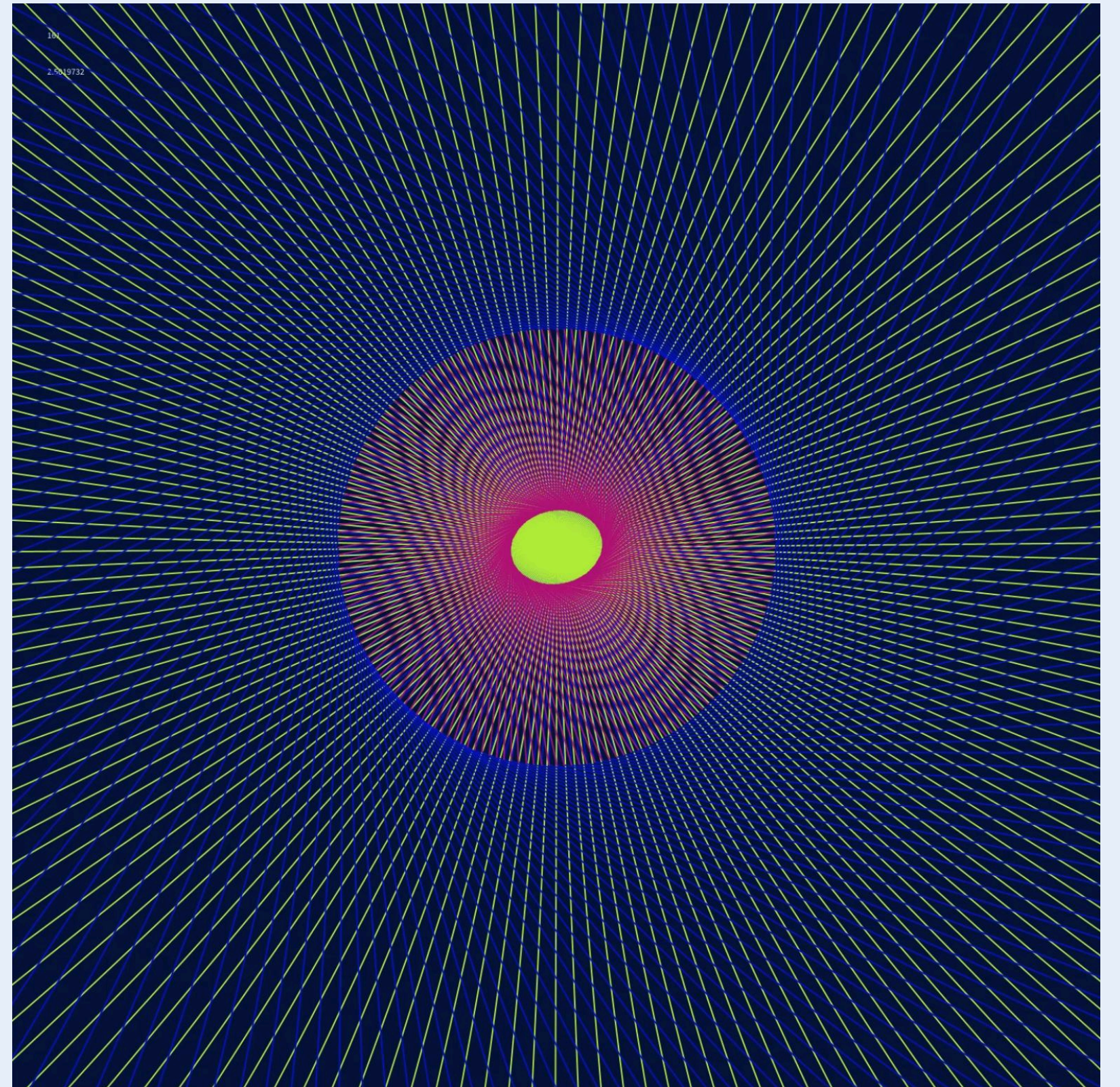
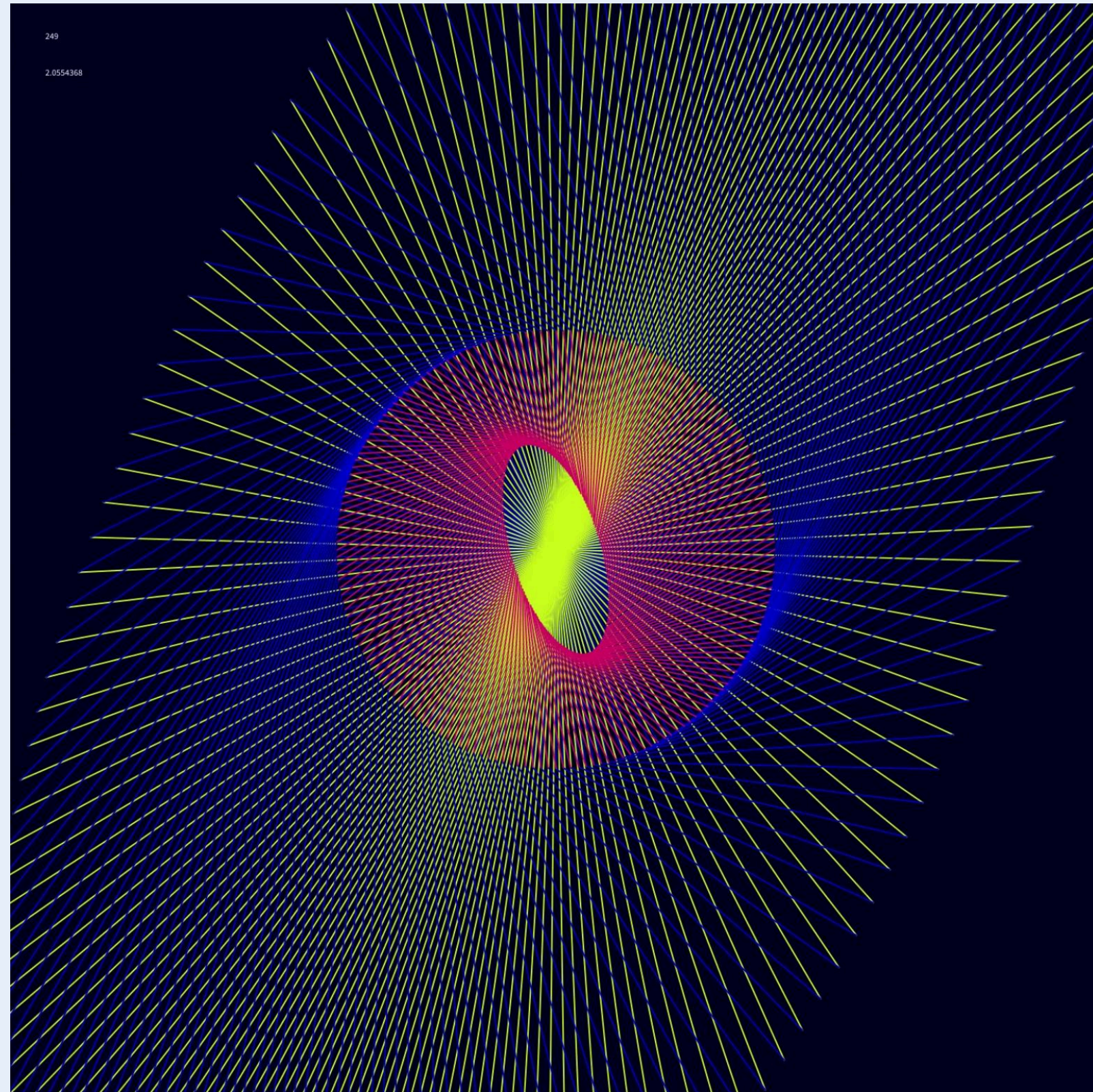
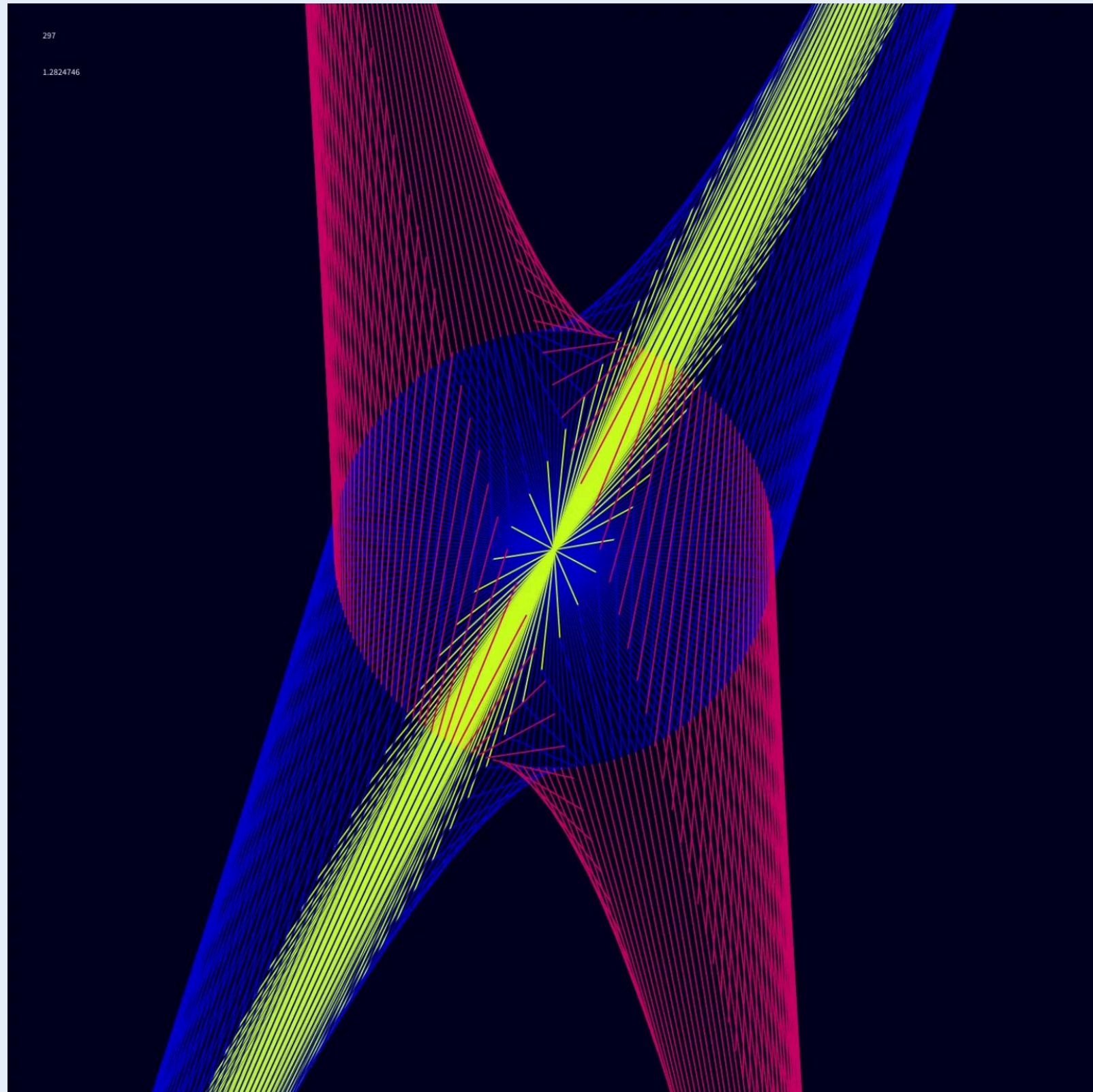


The image shows two overlapping screenshots of the Processing IDE. The top window, titled 'sketch', shows the following code:

```
1 int dimen;
2 float[] eigenvecs;
3 float[] eigenvals;
4 float[] matrix;
5 float[] inv;
6 float[] xs;
7 float[] ys;
8 float grad;
9 int timer;
10
11 void setup() {
12   size(1500,1500);
13   dimen = 1500;
14   timer = 0;
15
16   // construct matrices
17   eigenvecs = new float[4];
18   eigenvals = new float[4];
19
20   matrix = new float[4][4];
21   matrix = matrixMul(matrix, eigenvecs, eigenvals);
22
23   inv = new float[4][4];
24   inv = matrixInv(matrix, eigenvecs, eigenvals);
25
26   //construct circle coordinates
27   xs = new float[360];
28   ys = new float[360];
```

The bottom window, titled 'sketch2', shows the following code:

```
1 int dimen;
2 float[] eigenvecs;
3 float[] eigenvals;
4 float[] matrix;
5 float[] inv;
6 float[] xs;
7 float[] ys;
8 float grad;
9 float scalar;
10 int timer;
11
12 void setup() {
13   size(2500,2500);
14   dimen = 2500;
15   timer = 0;
16
17   // construct matrices
18   eigenvecs = new float[4] {1, -2, 3, 5};
19   eigenvals = new float[4] {4, 0, 0, 1};
20   matrix = new float[4][4];
21   inv = new float[4][4];
22
23   //construct circle coordinates
24   scalar = 300;
25   xs = new float[360];
26   ys = new float[360];
27   for (int i = 0; i < 360; i++) {
28     float amount = map(i,0,360,0,2*PI);
29     xs[i] = scalar*cos(amount);
30     ys[i] = scalar*sin(amount);
```

animated matrix transformations in processing

Transitioning to JavaScript

Moving from locally-hosted processing to my website proved difficult in the computational power and speed (or lack thereof) of JavaScript.

However, it was much easier than working with Processing—my variables can change types now!

I learned a lot about using event listeners, class lists, and DOM queries to keep interactions fluid and cohesive. I could also utilize a variety of libraries to enhance my user interface.

```
248 // MAIN FUNCTIONS
249 p.recalc = function() {
250   newCoords = new Coords();
251   invCoords = new Coords();
252
253   // calc scalars and normalize
254   eigenvals[0] = norm([eig1x/scalar,eig1y/scalar]);
255   eigenvals[3] = norm([eig2x/scalar,eig2y/scalar]);
256   eigenvecs = [eig1x/scalar/eigenvals[0],eig2x/scalar/eigenvals[3],eig1y/scalar/eigenvals[0],eig2y/scalar/eigenvals[3]];
257
258   myMatrix = matrixMult(eigenvecs,matrixMult(eigenvals,matrixInv(eigenvecs)));
259   inv = matrixInv(myMatrix);
260
261   // add transformed to coords
262   for (i = 0; i < count; i += 2) {
263     newCoords.x[i] = myMatrix[0]*unitCircle.x[i] + myMatrix[1]*unitCircle.y[i];
264     newCoords.y[i] = myMatrix[2]*unitCircle.x[i] + myMatrix[3]*unitCircle.y[i];
265     invCoords.x[i] = inv[0]*unitCircle.x[i] + inv[1]*unitCircle.y[i];
266     invCoords.y[i] = inv[2]*unitCircle.x[i] + inv[3]*unitCircle.y[i];
267   }
268
269   var updateLines = document.getElementById("numLines");
270   updateLines.innerHTML = "# of lines: "+String(count);
271
272   var updateMat = document.getElementById("intMat");
273   updateMat.innerHTML = "$$\\small{\\begin{bmatrix}}"+String(Math.round(myMatrix[0]*100))+String(Math.round(myMatrix[1]*100))+String(Math.round(myMatrix[2]*100))+String(Math.round(myMatrix[3]*100))+String("\\end{bmatrix}}";
274   MathJax.typesetPromise([updateMat]).then(() => {});
275
276   var updateEigs1 = document.getElementById("intEigs1");
277   updateEigs1.innerHTML = "$$\\scriptsize{\\lambda_1}"+String(Math.round(eigenvals[0]*100))+String("\\lambda_2}"+String(Math.round(eigenvals[3]*100))+String("\\end{matrix}}";
278   MathJax.typesetPromise([updateEigs1]).then(() => {});
279
280   var updateEigs2 = document.getElementById("intEigs2");
281   updateEigs2.innerHTML = "$$\\scriptsize{\\lambda_1}"+String(Math.round(eigenvals[0]*100))+String("\\lambda_2}"+String(Math.round(eigenvals[3]*100))+String("\\end{matrix}}";
282   MathJax.typesetPromise([updateEigs2]).then(() => {});
283
284   p.drawing();
285 }
286
287 p.drawing = function () {
288   p.background(0,0,8);
289   timer = 0;
290
291   // BLUE ORIGINAL
292   if (varOrig.querySelector(".checkLine").checked) {
293     p.stroke(101,106,255);
294     for (i = 0; i < count; i += 2) {
295       vector(unitCircle.x[i],unitCircle.y[i]); // original
296       vector(newCoords.x[i],newCoords.y[i],unitCircle.x[i],unitCircle.y[i]); // to c
297     }
298   }
299   if (varOrig.querySelector(".checkCir").checked) {
300     p.stroke(101,106,255);
301     p.ellipse(0,0,2*scalar,2*scalar,p.min(count0,50)); // ellipse
302   }
303
304   // GREEN NEW
305   if (varNew.querySelector(".checkLine").checked) {
306     p.stroke(222,255,35);
307     p.push();
308     p.fill(222,255,35);
309     for (i = 0; i < count/2; i += 1) {
310       vector(newCoords.x[i],newCoords.y[i]);
311       vector(-1*newCoords.x[i],-1*newCoords.y[i]); // lines
```

JavaScript Libraries

The logo for p5.js, featuring the text "p5.js" in a red, sans-serif font.

p5.js

- visualizations with canvas instancing
- WEB3GL format
 - faster rendering/performance
 - easier coordinate mapping (origin is in center of canvas vs top left)

The logo for LottieFiles, featuring a teal square icon with a white curved line and the text "LottieFiles" in a teal, sans-serif font.

LottieFiles

- make Figma animations into playable Lottie JSON files (or DotLottie)
- infinitely scalable
- relatively small file size

The logo for MathJax, featuring the text "MathJax" in a green, sans-serif font.

MathJax.js

- display LaTeX (standard math typesetting system) in HTML pages
- customizable formatting
- combined with p5 and innerHTML to live update calculations to match interactions

Programming Matrix Algebra

The problem: JavaScript is pretty slow when you ask it to do many calculations at once.

And with 3 matrix calculations and over 400 vector coordinates updating every single frame, that was a lot to process. My first few iterations had my PC whirring and my background YouTube videos glitching out and crashing.

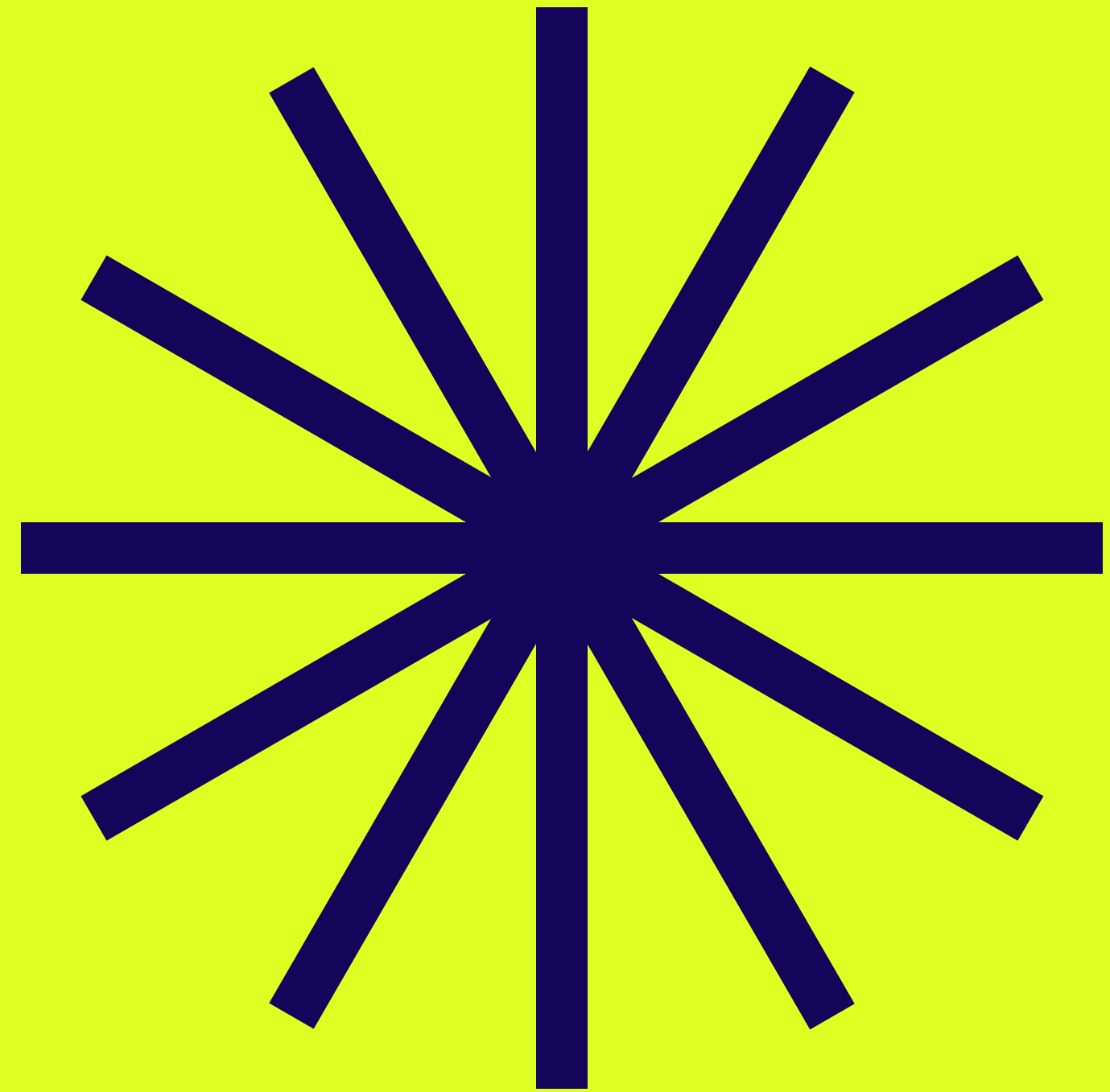


The solution: I reevaluated exactly what I wanted out of my code, and that was: coordinates.

For example, do I really need to calculate the angle of an eigenvector? I made a workaround with just coordinates. Do I need to recalculate all variables, or just some of them? I separated my calculations and minimized the function calls.

FINAL DESIGNS

cordeliart.com/spectral



SPECTRAL

Explore the world of matrices.

SPECTRAL

created by Lena Pang

start here if you're new
to linear algebra or need
a refresher...



The Basics

Prerequisites including vectors,
linear combinations, and
simple matrix algebra

Matrices as Transformations

The ball: visualizing
eigenvalues and eigenvectors.

...or here if you're
already familiar with
the concepts



Interact

The FUN part! Warp in 2D.

SPECTRAL

created by Lena Pang

The Basics

Prerequisites including vectors, linear combinations, and simple matrix algebra

Matrices as Transformations

The ball: visualizing eigenvalues and eigenvectors.

Interact

The FUN part! Warp in 2D.



Matrices as Transformations

II. Now consider: matrix warps ball.

...s in a certain direction. Let's look at some examples in 2D.

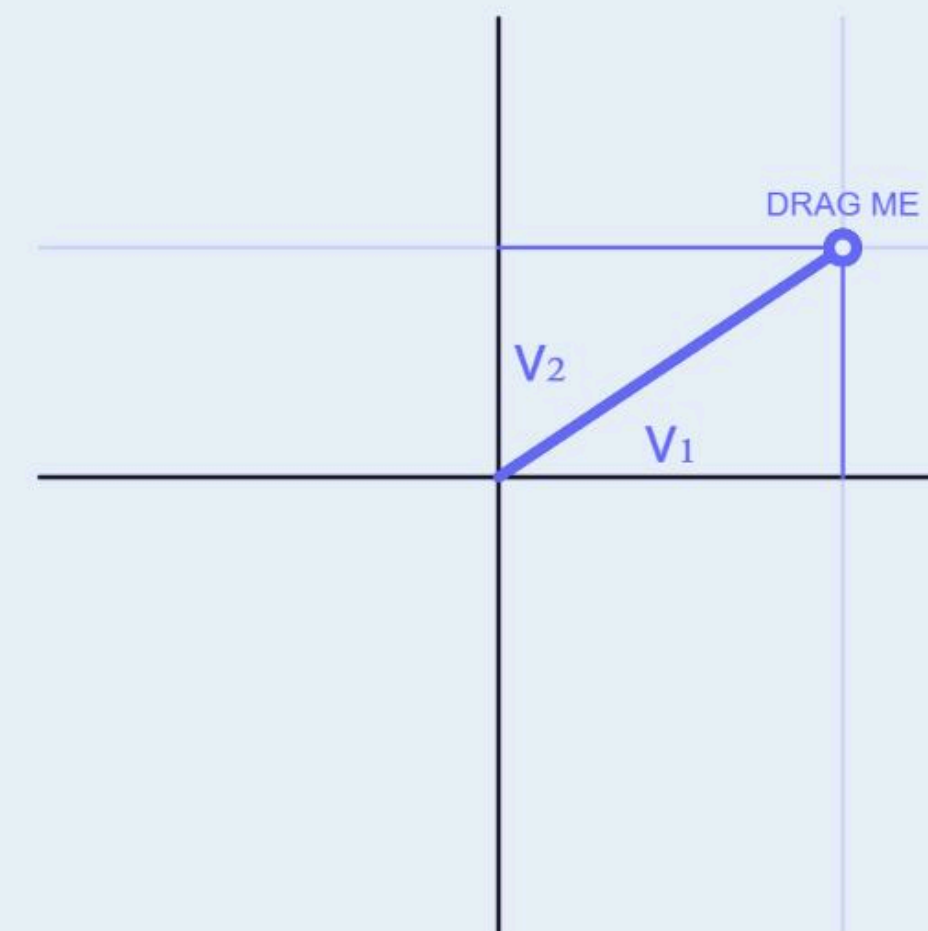
One Vector

... everywhere except 1s on the diagonal. ...g to the vector. Useless.

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

...t around.

...an manipulate how the matrix



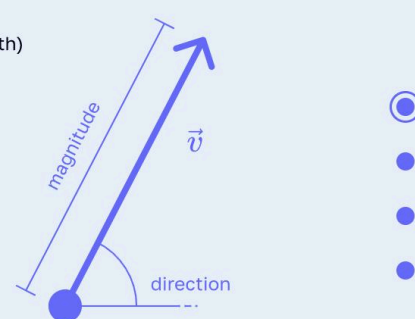


The Basics

Vectors, linear combinations, span, and bare-bones matrix algebra. Scroll to begin.

Vectors

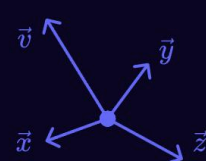
vector – a thing with magnitude (length) and direction (...well, direction)



Linear Combinations

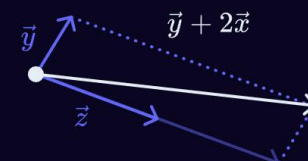
If you have a collection of vectors, you call them a **set**. The only restriction here is no duplicates.

$$\{\vec{v}, \vec{x}, \vec{y}, \vec{z}\}$$

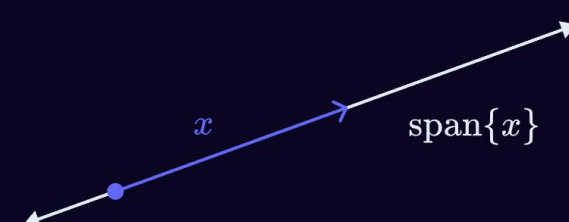


Another important term is a **linear combination**: a vector produced scaling and/or adding vectors.

$$c_1\vec{x} + c_2\vec{y} + \dots + c_n\vec{z}$$



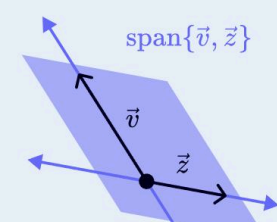
The **span** of a set is ALL possible linear combinations of its vectors. As an example, the span of a singular vector is a line (scaling up and down to infinity).



So what about the span of two vectors? A plane? Not always. That depends on a little something called linear independence.

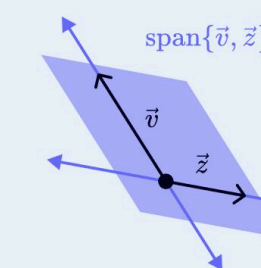
Linear Independence

These two vectors are **linearly independent**, so the span is a plane...

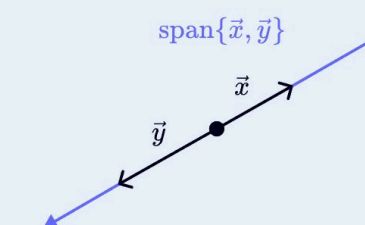


Linear Independence

These two vectors are **linearly independent**, so the span is a plane...



...but these two are not. They are **linearly dependent** because \vec{y} is a linear combination of \vec{x} (specifically, \vec{y} is a scaled copy of \vec{x} , and vice versa).



So what exactly is linear independence? Basically, vectors in a set are **not redundant**.

There are a few ways to think about it:

- the number of vectors matches the dimension of the span (in the first example, 2 vectors making a 2-D plane = linearly independent)
- each vector brings a new direction to the set (in the second example, \vec{y} does not bring a new direction to \vec{x} = NOT linearly independent)

The super formal definition is that a linear combination of the vectors equals zero is if and only if **ALL** the coefficients are zero.

$$0 = c_1\vec{x} + c_2\vec{y} + \dots + c_n\vec{z}$$

Matrices

A **matrix** is a grid with m rows and n columns of **elements** (not components), labeled by their row, then their column.

$$A_{m \times n} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & & \\ \vdots & & \ddots & \\ a_{m,1} & & & a_{m,n} \end{bmatrix}$$

For our sake, a matrix is a **function** that acts on a vector. Input a vector, output a vector. For our sake, the calculations are not important.

> Want to learn the calculations anyways?

Linearity

The important thing, though, is that matrices are **linear operations!** Hence, the name linear algebra. That means vector addition and scalar multiplication can be moved through the operator:

$$A(\vec{x} + \vec{y}) = A\vec{x} + A\vec{y}$$

$$A(k\vec{x}) = kA\vec{x}$$

Yippee! This is a super useful property that doesn't happen very often. For example, the easy function $f(x) = x^2$ is not linear, because $f(2x) \neq 2f(x)$.

And that's it for the basics!

Matrices as Transformations →



Matrices as Transformations

Consider: a ball. Now consider: matrix warps ball. Let me explain.

Matrices stretch vectors in a certain direction. Let's look at some examples in 2D.

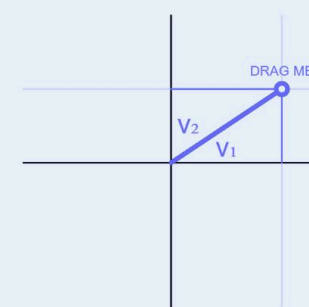
Working with One Vector

The **identity matrix** (0 everywhere except 1s on the diagonal) does nothing to the vector. Useless.

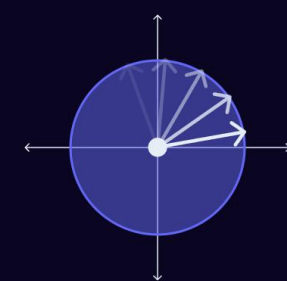
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Now try moving the dot around.

By scaling the 1s, we can manipulate how the matrix transforms our vector.



> Show me the calculations!



The Ball: a New Perspective

AKA, the unit circle. This includes all the vectors with magnitude 1 in every single direction (in 2D).

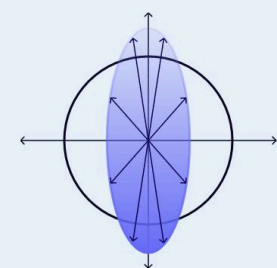
Working with Everything (in 2D)

We've been working mostly with one vector made of two components.

Now, we're going to think of every vector as a linear combination of the components to see how a matrix stretches **all** vectors.

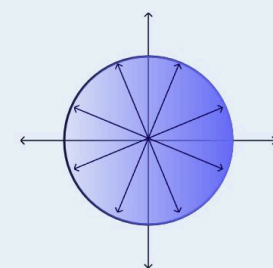
Scaling

$$\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}$$



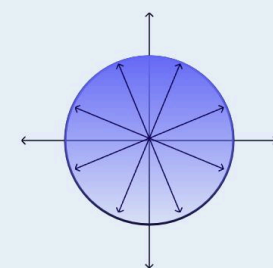
Rotating

$$\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$



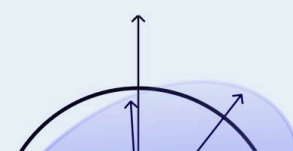
Reflecting

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$



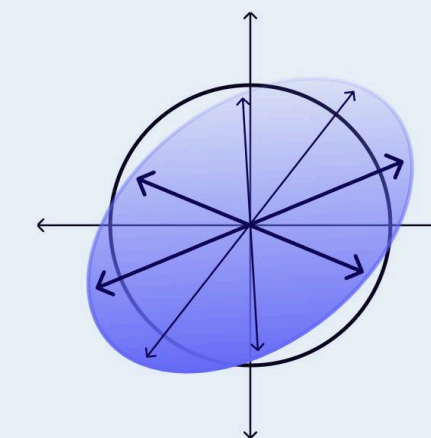
Eigenvectors & Eigenvalues

Of course, we can combine all of these into one. Take a look at this matrix transformation:



Eigenvectors & Eigenvalues

Of course, we can combine all of these into one. Take a look at this matrix transformation:



You might notice that the bold vectors stay pointing in the same direction, even after they are transformed. They're only scaled up and down.

These are called **eigenvectors** \vec{u} . The amount they're scaled is called an **eigenvalue** λ . Because the matrix just scales \vec{u} by λ , we can say:

$$A\vec{u} = \lambda\vec{u}$$

The **spectral radius** of a matrix is the largest eigenvalue magnitude (absolute value).

> Exercise: What are the eigenvectors in the examples above?

Every vector v can be written as a linear combination of the eigenvectors. In other words, we can think of each eigenvector as our "components."

That means we can write the transformation of every vector $A\vec{v}$ as a linear combination of the transformed eigenvectors. **This is what allows us to warp the unit circle in the way we did above.**

$$\vec{v} = c_1\vec{u}_1 + c_2\vec{u}_2 + \dots + c_n\vec{u}_n$$

$$A\vec{v} = A(c_1\vec{u}_1 + c_2\vec{u}_2 + \dots + c_n\vec{u}_n) = c_1A\vec{u}_1 + c_2A\vec{u}_2 + \dots + c_nA\vec{u}_n$$

Eigendecomposition

Each matrix can be decomposed into 3 matrices using its eigenvectors and eigenvalues:

$$A_{n \times n} = QDQ^T = \begin{bmatrix} \uparrow & \uparrow & \dots & \uparrow \\ \vec{u}_1 & \vec{u}_2 & \dots & \vec{u}_n \\ \downarrow & \downarrow & \dots & \downarrow \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & \\ \vdots & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix} \begin{bmatrix} \leftarrow \vec{u}_1 \rightarrow \\ \leftarrow \vec{u}_2 \rightarrow \\ \vdots \\ \leftarrow \vec{u}_n \rightarrow \end{bmatrix}$$

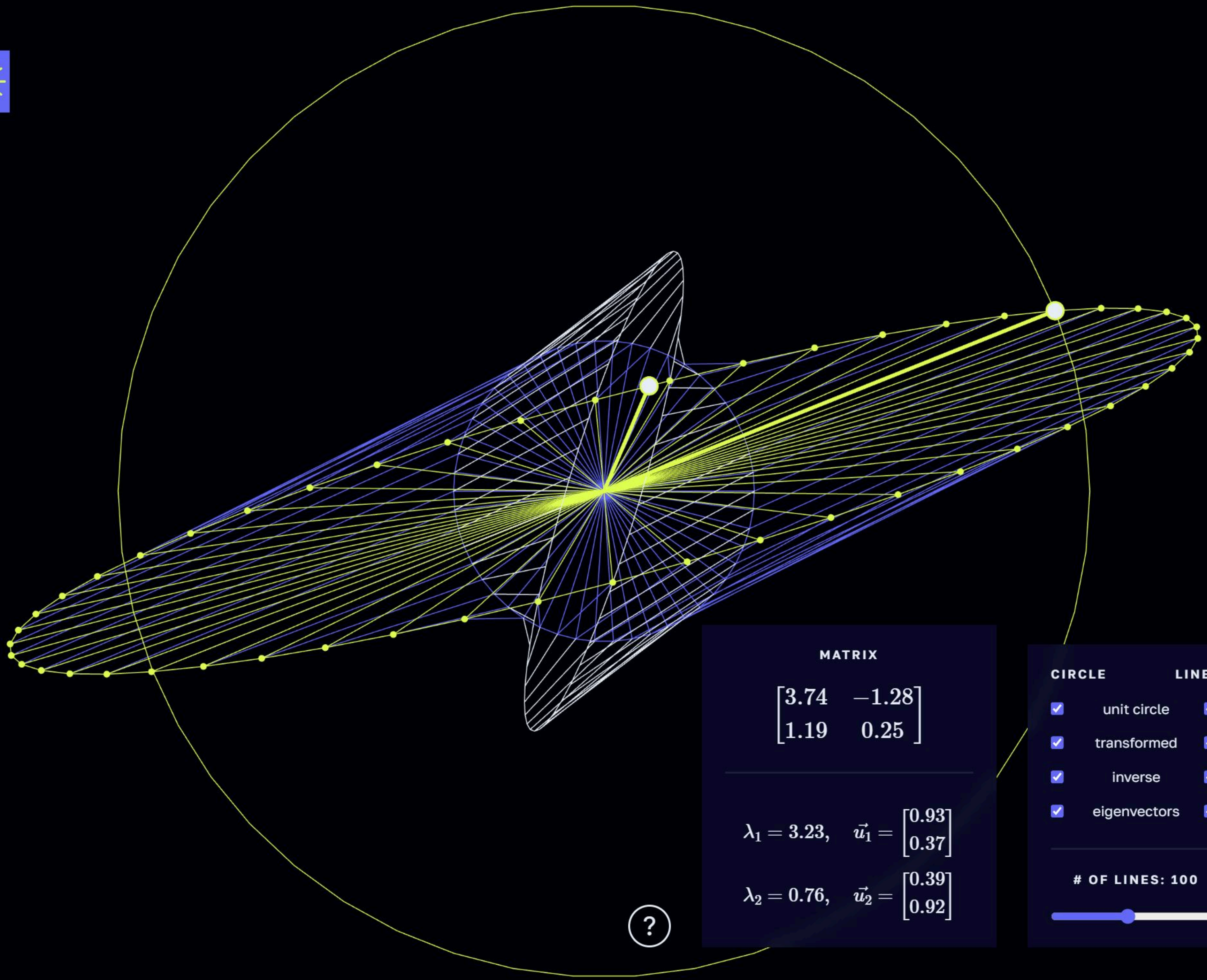
D is a matrix with the eigenvalues along the diagonal and zeroes everywhere else.

Q is a matrix with the normalized eigenvectors as its columns. (Q^T is the transpose of Q , meaning flipped along the diagonal.)

This decomposition is how the vectors and ellipses on the next page are calculated.

Now, those are big words, complicated matrices, and weird symbols. So I've created an interactive experience where you can click and drag eigenvectors, and see how it changes the matrix in real time.

Interact in 2D →



MATRIX

$$\begin{bmatrix} 3.74 & -1.28 \\ 1.19 & 0.25 \end{bmatrix}$$

$$\lambda_1 = 3.23, \quad \vec{u}_1 = \begin{bmatrix} 0.93 \\ 0.37 \end{bmatrix}$$

$$\lambda_2 = 0.76, \quad \vec{u}_2 = \begin{bmatrix} 0.39 \\ 0.92 \end{bmatrix}$$

CIRCLE

- unit circle
- transformed
- inverse
- eigenvectors

LINES

-
-
-
-

OF LINES: 100





What am I looking at?

This is the ball, but fancier:

green lines =
transformed vectors

blue lines = connect to
unit circle vectors

white lines = inverse
(applied to unit circle)

Drag around the eigenvectors to manipulate the matrix—you'll notice that the eigenvectors will always be parallel to the blue unit circle lines.

Close

the matrix (calculated
with eigendecomposition)

its eigenvalues
and normalized
eigenvectors

MATRIX

$$\begin{bmatrix} 3.74 & -1.28 \\ 1.19 & 0.25 \end{bmatrix}$$

$$\lambda_1 = 3.23, \quad \vec{u}_1 = \begin{bmatrix} 0.93 \\ 0.37 \end{bmatrix}$$

$$\lambda_2 = 0.76, \quad \vec{u}_2 = \begin{bmatrix} 0.39 \\ 0.92 \end{bmatrix}$$

customize what you see

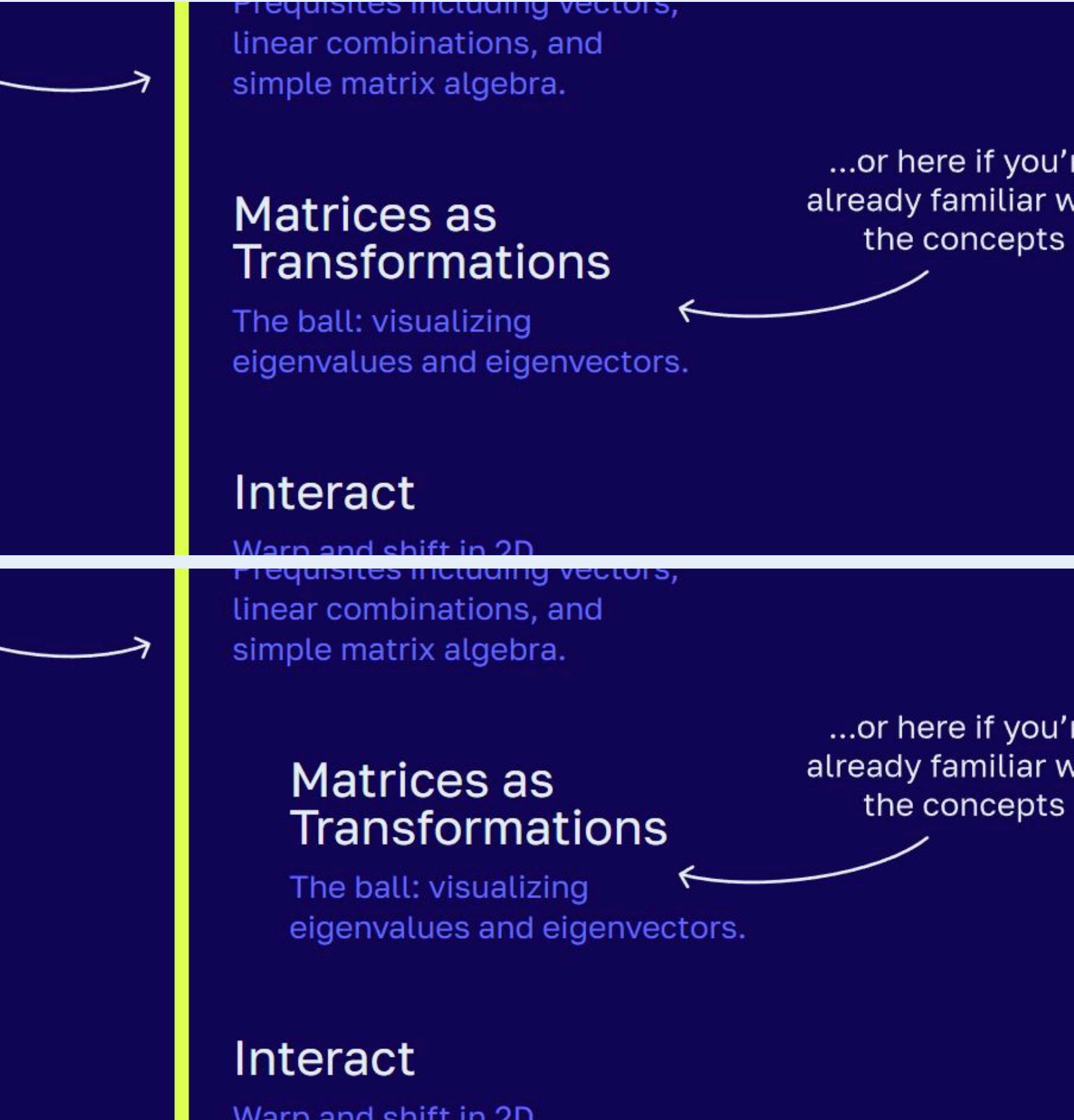
CIRCLE LINES

- | | | |
|-------------------------------------|--------------|-------------------------------------|
| <input checked="" type="checkbox"/> | unit circle | <input checked="" type="checkbox"/> |
| <input checked="" type="checkbox"/> | transformed | <input checked="" type="checkbox"/> |
| <input checked="" type="checkbox"/> | inverse | <input checked="" type="checkbox"/> |
| <input checked="" type="checkbox"/> | eigenvectors | <input checked="" type="checkbox"/> |

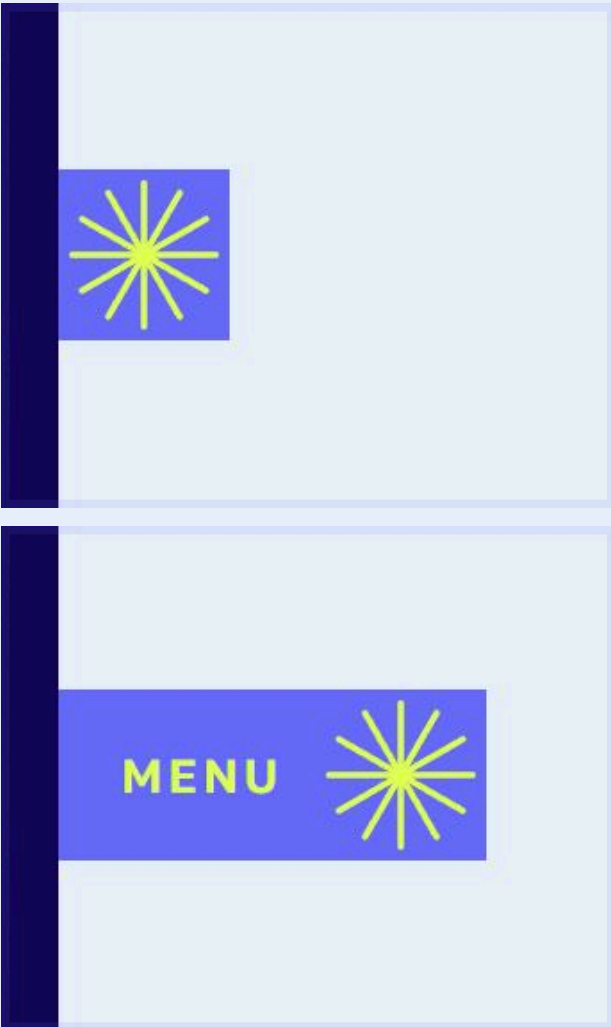
OF LINES: 100

Subtle Animations

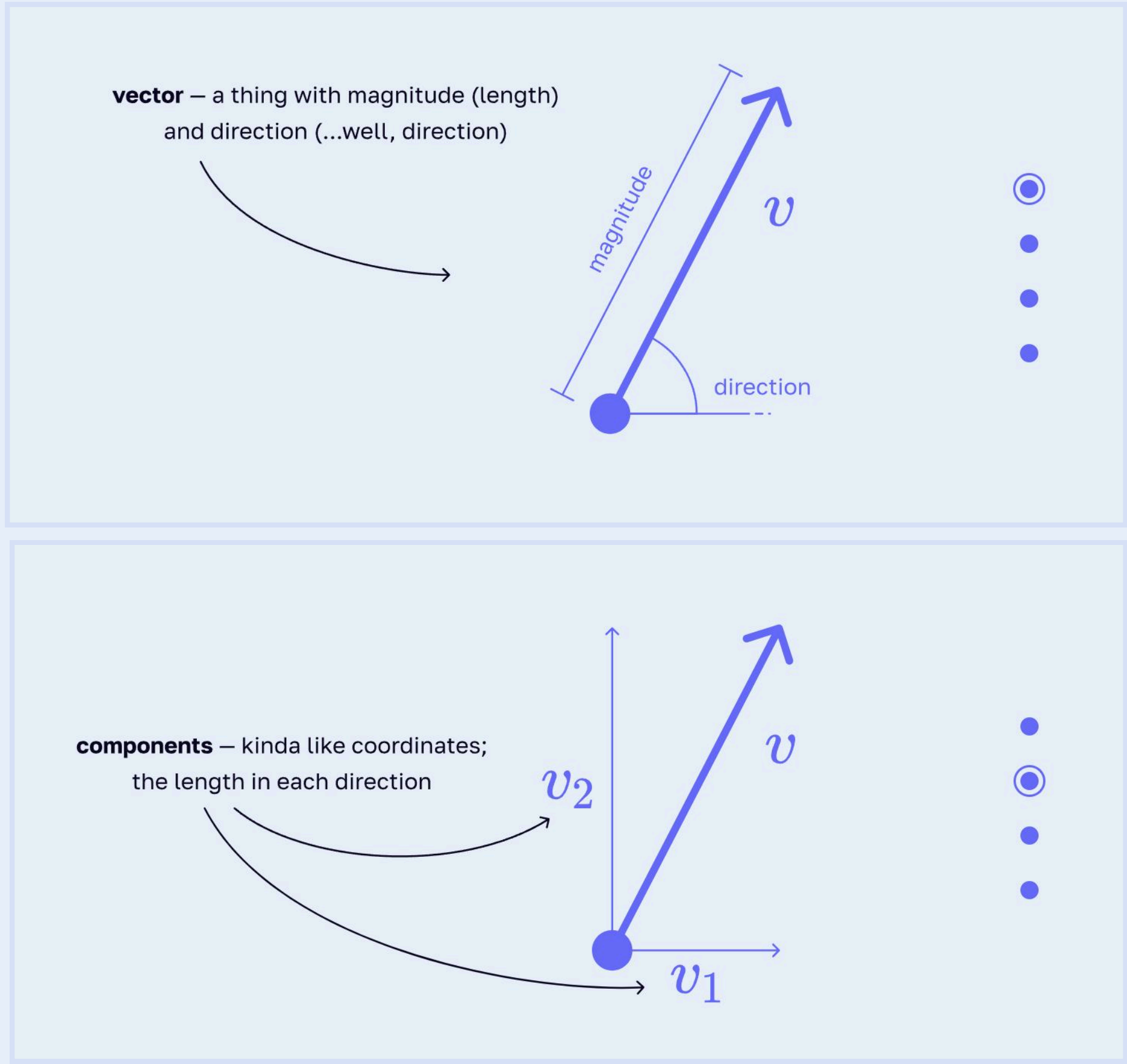
Navigation Hover



Menu Tab Hover



Dot Scrolling



Subtle Animations

Dropdown

> Want to learn the calculations anyways?

Dropdown Hover

> Want to learn the calculations anyways?

Dropdown Extended

> Want to learn the calculations anyways?

First off, the height of your vector **must** match the width of your matrix. Otherwise everything breaks.

The process for each row:

1. multiply the first A element by the first \vec{x} component
2. multiply the second A element by the second \vec{x} component
3. multiply the third A element by the third \vec{x} component
4. ...you get the idea
5. then add each row up and you're done!

So you put in a vector of size n , and output a vector of size m .

$$A\vec{x} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & & \\ \vdots & & \ddots & \\ a_{m,1} & & & a_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n \\ a_{2,1}x_1 + a_{2,2}x_2 + \dots \\ \vdots \\ a_{m,1}x_1 + \dots + a_{m,n}x_n \end{bmatrix}$$

Need a more concrete example? Here you go!

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 0 \\ 4 & -1 \end{bmatrix} \quad \vec{x} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

$$A\vec{x} = \begin{bmatrix} 1 & 3 \\ 2 & 0 \\ 4 & -1 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 1(5) + 3(6) \\ 2(5) + 0(6) \\ 4(5) + -1(6) \end{bmatrix} = \begin{bmatrix} 23 \\ 10 \\ 14 \end{bmatrix}$$

STYLE GUIDE

color + typography + assets

Color

R	G	B
231	239	249



222	255	35
-----	-----	----



101	106	255
-----	-----	-----



19	6	90
----	---	----



0	0	29
---	---	----

Typography

Heading 1 | 1rem

Heading 2 | 4rem

Heading 3 | 4rem

Heading 4 | 4rem

HEADING 5 | 0.5rem

Paragraph | 1.1rem

navigation title

1.7rem
Golos Text 400

page title

6rem
Golos Text 900

page subtitle

1.7rem
Golos Text 700

section

1.7rem
Golos Text 400

categories

0.8rem
Golos Text 600
2px letter spacing

body copy

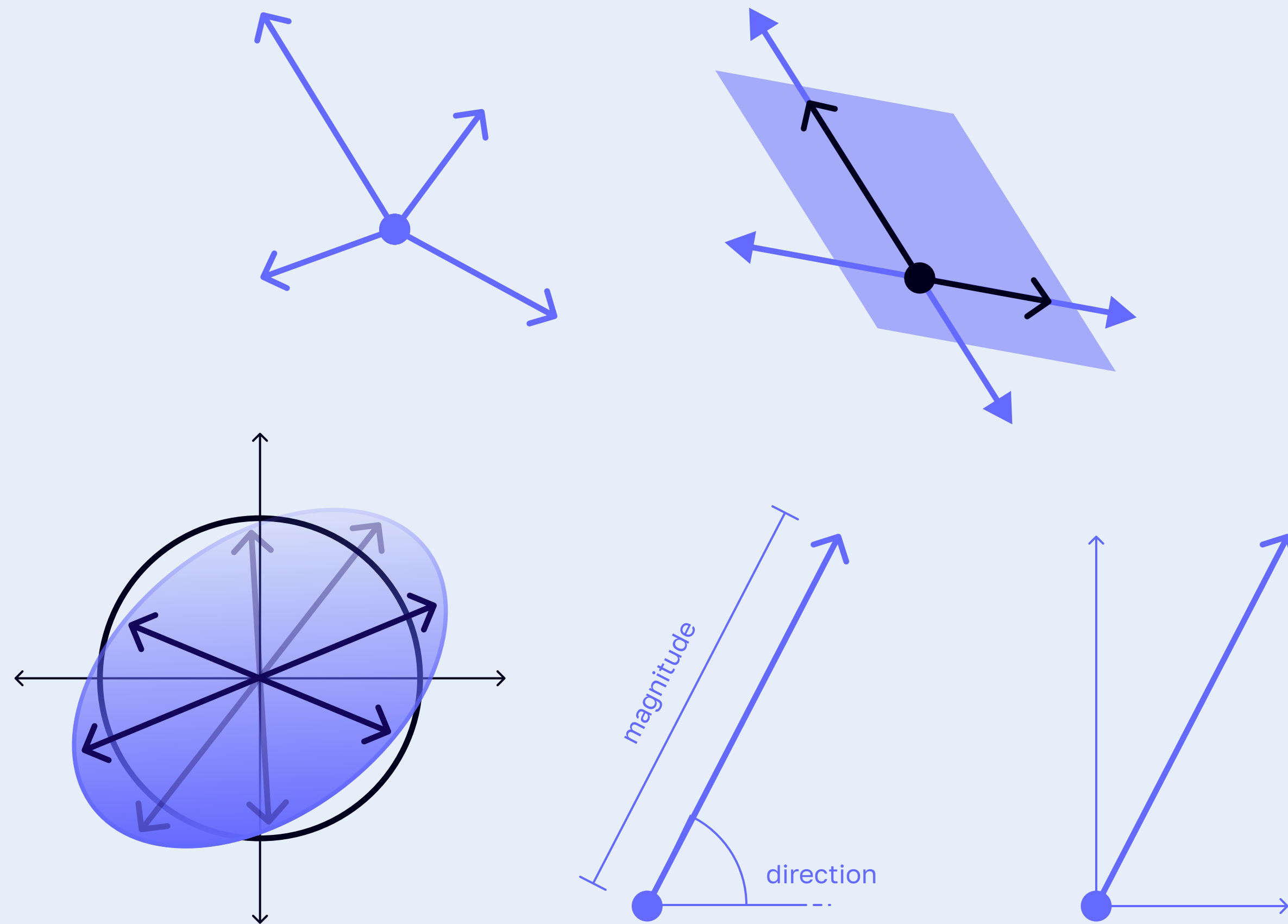
1.1rem
Golos Text 400
(Span 700)

Assets

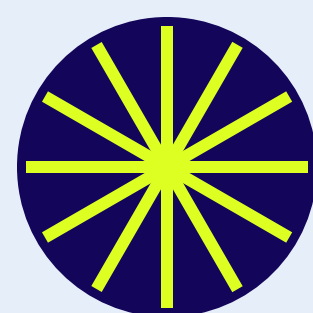
OG Image



Graphics



Favicon



REMARKS

user feedback + deisgner reflection

User Feedback

I sent the website out to my family, professors, friends, and friends of friends. (There is a surprising number of math nerds in my close network.)

Common feedback from the non-math users was that they didn't know what was going on (and couldn't give much critique on the content itself), but it looked cool and the interaction was fun to play with. Many people liked the splash screen and the overall aesthetic style.

Feedback from math users was more helpful. Most commented that it moved a bit quickly, which was fair, considering it wasn't meant to be an independent course.

Some revisions I made from the critiques included adding arrow notation over the vector variables to differentiate between vectors and components, and specifying that "complex eigenvectors" refers to complex numbers, not just a complicated eigenvector, haha.

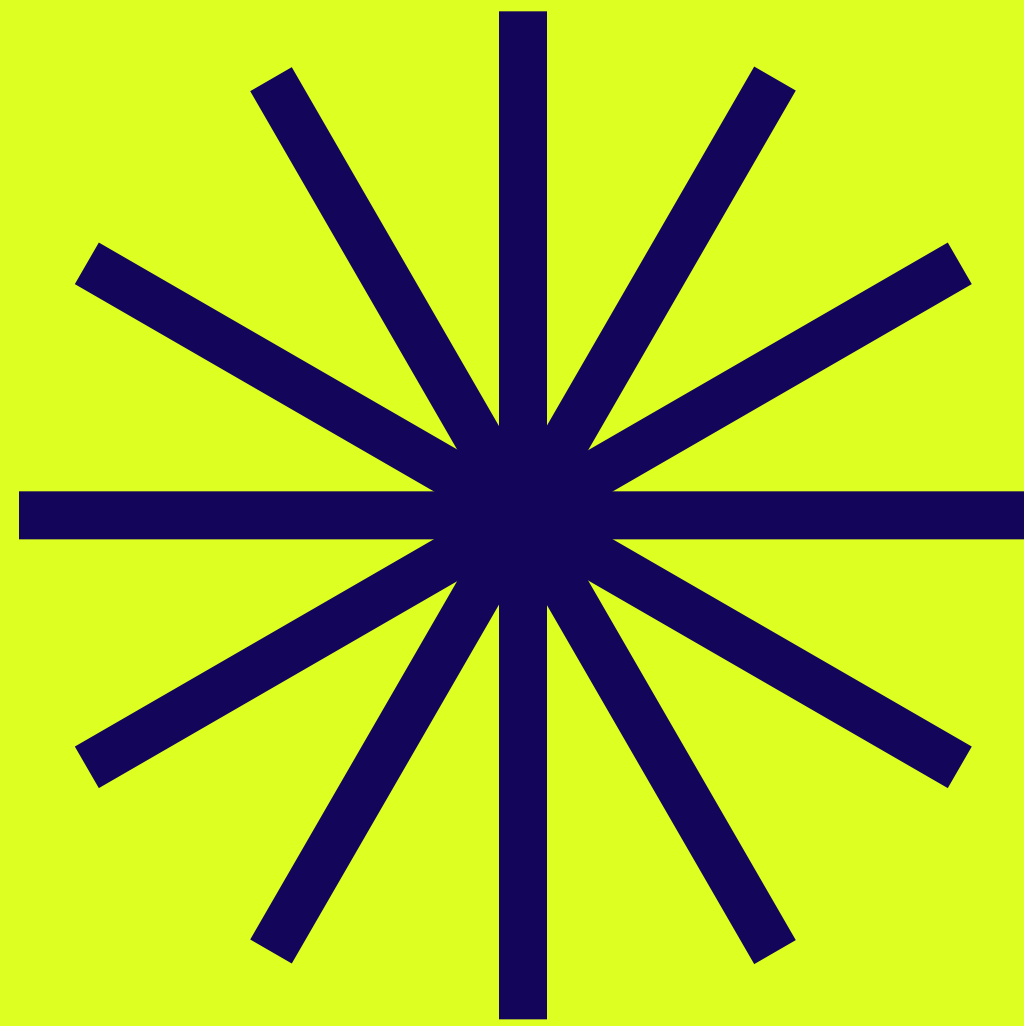
I also received a suggestion that the blue lines (original) and white lines (inverse) were overwhelming, so I disabled the inverse lines upon first launch, keeping the option to turn them on. However, I decided to keep the blue lines on launch because they serve an important purpose: showing how the eigenvectors distort compared to all the other vectors.

Designer Reflection

This project was a fun experience. I enjoy teaching, so making a crash course was fun in itself—and the designing, of course, was an adventure of its own.

My first designs were definitely rough, but once I got the content going, the UI almost built itself around the interactivity and content. While some of my reactive code could be reduced, I am satisfied with the overall final product. The site feels cohesive, original, and appealing from even a purely aesthetic viewpoint.

If I continue to work on this project, I will add more to the crash course aspect, and possibly make a 3D version of the interactive portion. (Currently, my issue with that is I am unsure how drag controls will work in a 3D environment on a 2D screen. Rendering could also be an issue for some browsers/computers.)



SPECTRAL

Thanks for reading!