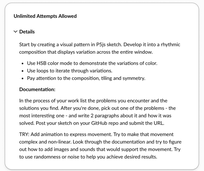
Inspo Gathering!


Unlimited Attempts Allowed

Details

Start by creating a visual pattern in P5js sketch. Develop it into a rhythmic composition that displays variation across the entire window.

- Use HSB color mode to demonstrate the variations of color.
- Use loops to iterate through variations.
- Pay attention to the composition, tiling and symmetry.

Documentation:

In the process of your work list the problems you encounter and the solutions you find. After you're done, pick out one of the problems - the most interesting one - and write 2 paragraphs about it and how it was solved. Post your sketch on your GitHub repo and submit the URL.

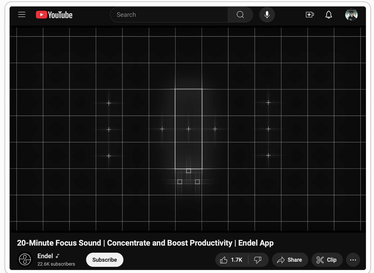
TRY: Add animation to express movement. Try to make that movement complex and non-linear. Look through the documentation and try to figure out how to add images and sounds that would support the movement. Try to use randomness or noise to help you achieve desired results.

Code location:

https://editor.p5js.org/mreplan_mdes/sketches/zH5-tdno9

Gathering Inspiration for this Project!

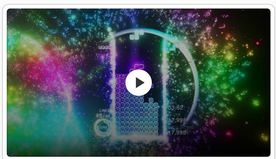
Inspo, Productivity Apps: Endel App used for focus periods accompanied by sound and geometric pattern visuals.



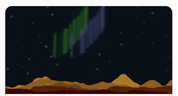
Inspo, 2D Motion Graphics, Cubesato!: color choice, movement to music.

2D motion graphic
YouTube

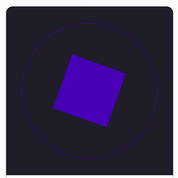
Inspo, Visual & Sound Design indicating intensity: Tetris Effect video game dynamically adjusts effects based on player performance.



10 Minutes of TETRIS EFFECT Music and Gameplay
YouTube



Aurora
A meditative space where you can look outside of a train window, observe...
drawaurora.com



Creative Coding: Build a Relaxing animation with P5.js
Coding is no longer just about logic and problem-solving; it's also a...
medium.com

Interactive Focus Timer — Erika Noma
erikanoma.com

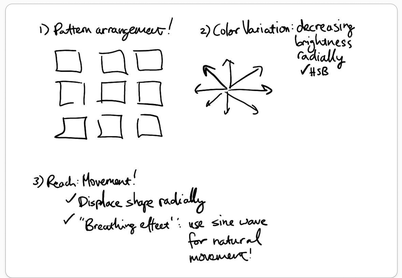
Focus! I'm watching you!
The final project for Creative Coding Fall 2021
medium.com

"By combining creativity with code, you can create impactful visuals that not only challenge your mind, but also relax your senses."

Code It Up

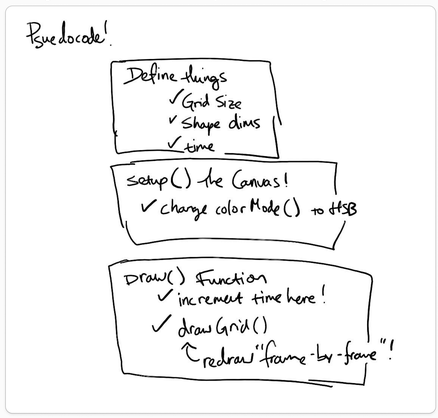
Code it... but first, literally sketch it:

I like to first draw out the concept with the use of my tablet and sketching app. **My goal is to create an array of squares that vary in brightness radially** (less color on the outer tiers) and later on, implement movement that emulates gentle breathing!



Okay! Psuedocode it Up!

I then like to psuedocode it, just identifying the main "blocks" of code helps me break down the effort into manageable chunks here and keeps the vision straight!



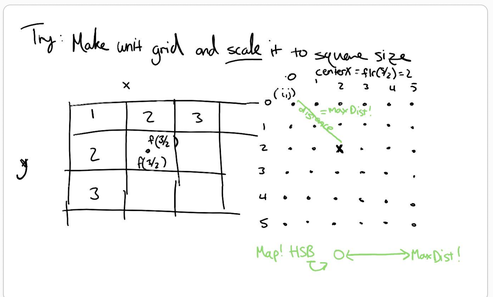
Most Interesting Challenge: Using a Unit Grid

So I really appreciated the grid approach taught in class, but I challenged myself to think of a more parameterized/alternative way to generate the repeated shapes. Further, t**his is what I primarily did in previous data visualization work with MatLab... so I wanted to see how feasible it was to apply my MatLab approach to P5js!**

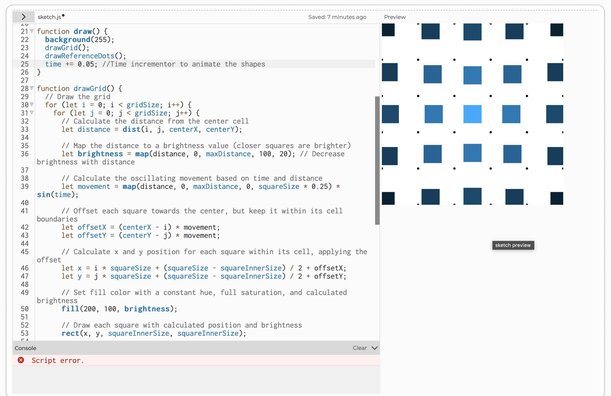
I created a draw function drawGrid to generate the "Unit Grid" by looping through i and j iterator variables. From there, the placement of each **rectangle corner coordinate would be simply a multiple of the i and j iterator, the unit grid position!**

Reflecting on this approach when this implemented... I like this! It helps reinforce me to approach repeated patterns in a mathematical way and it also makes this very scalable. I'm thinking of how this sketch can actively expand based on the user's webpage viewport size!

Drawing the coordinate system helped A LOT when coding it up, below is a sketch where I was tracking relative distance from the center (the square that stays static) and calculating the distance from an example (i,j) square. This distance will then be updated by a sin() function to vary the position of each i/jth square over time!



Overall, when in doubt, draw it out! ...and applying my MatLab experience, walking through the code in each iteration and understanding how each variable changes after each iteration helps a lot here!



Other Challenges I noted!

Calculating Euclidian Distance From Center: Okay, at first I spent a lot of personal CPU thinking cycles on using pythagorean theorem to calculate and track the distance for each unit grid point ... a quick documentation search yielded the distance() function and that saved the day!

Understanding how I can use sin() to vary movement: now that I can determine the distance from the center, I can use this, multiplied by sin(time) to oscillate and generate an "offsetX" and "offsetY". One interesting problem I noticed is the value I set my time+= incrementor. I set it simply to 1 but the oscillation was ridiculously fast...since sin() is in radians, an input of 1 and increments of 1 jumps through the sin() wave quite rapidly! Further, by incrementing smaller, you travel up and down the sin() slower, and appreciate the gentle curves! Currently, I decide to set it to 0.05.

map() helped create a way to vary color: the relative distance from center helped dictate the amount of brightness I wanted each square to have, just as I sketched in the beginning! By using map() I can map the ith/jth distance from center to the relative brightness compared to the central square!