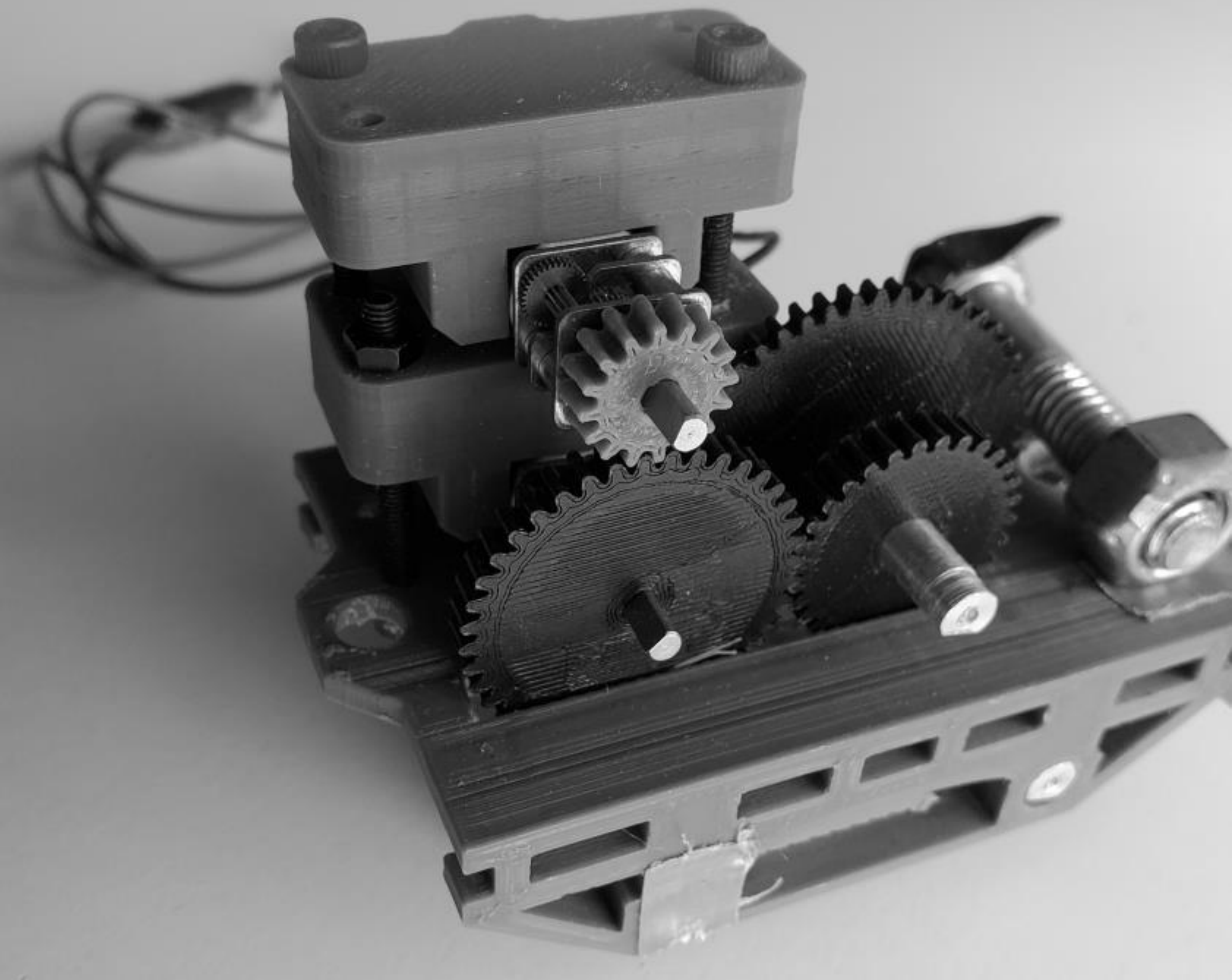
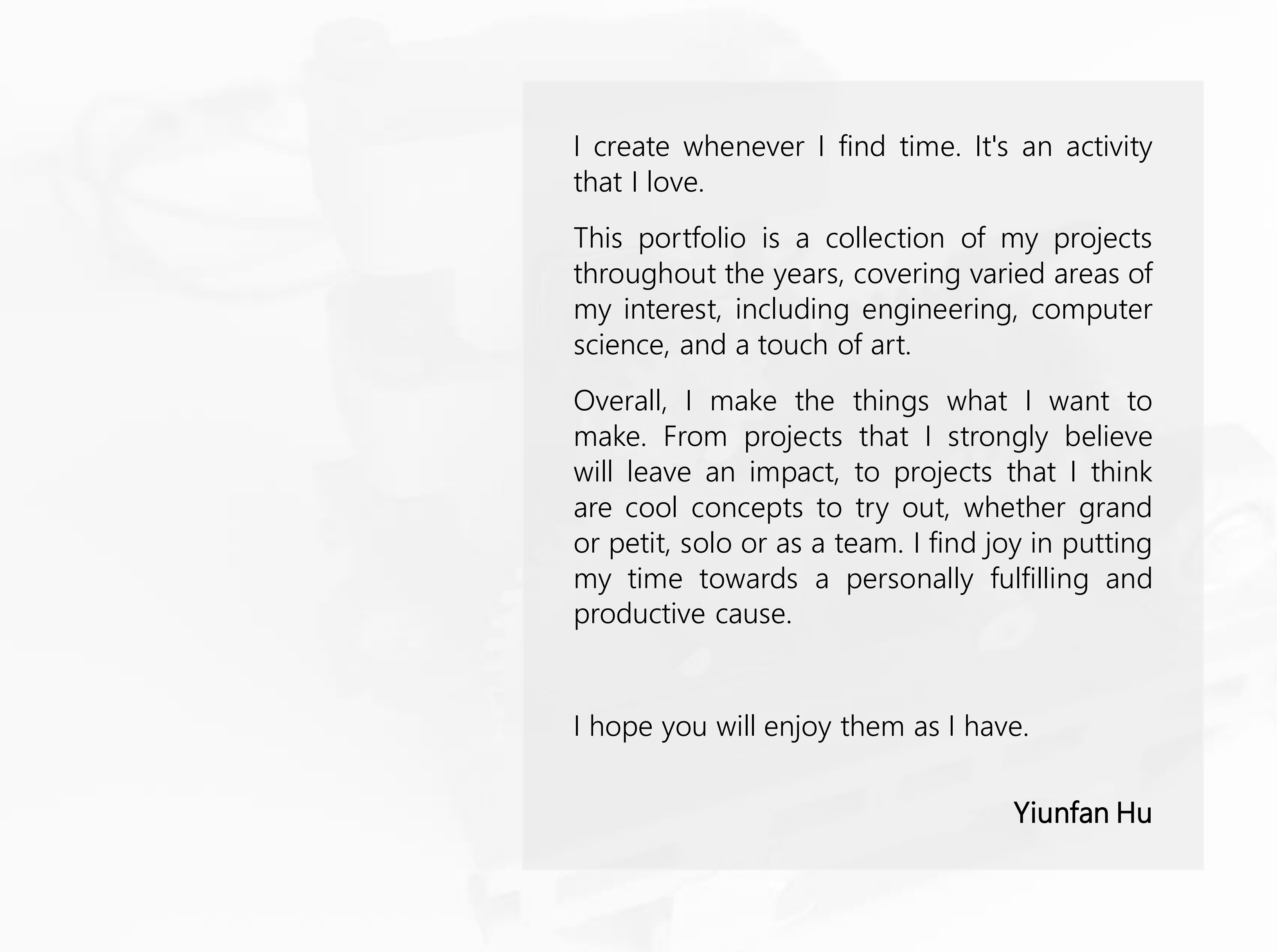
Maker Portfolio

Yiunfan Hu



Yiunfan Hu
yiunfanhu@gmail.com
https://croissantderp.github.io

Maker Portfolio



I create whenever I find time. It's an activity that I love.

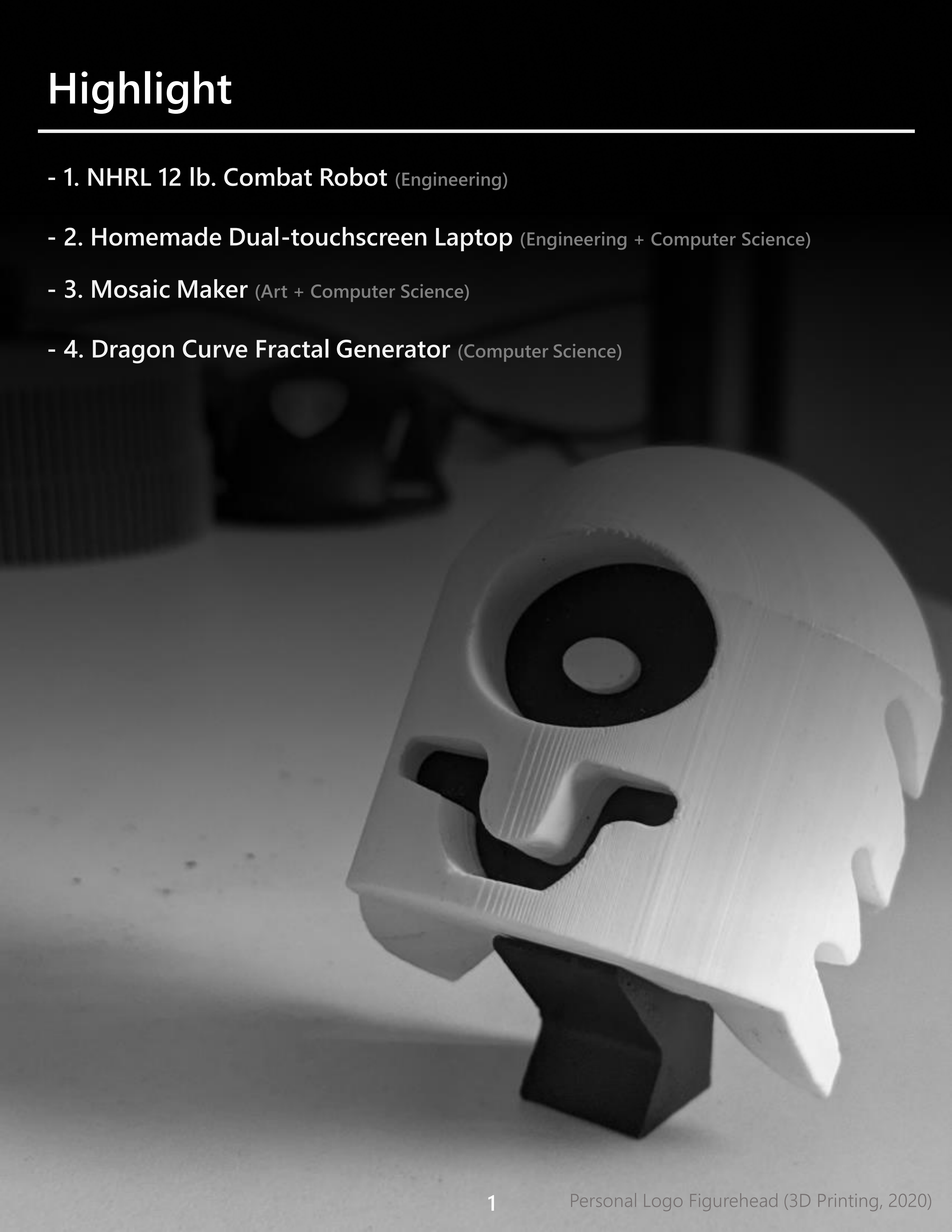
This portfolio is a collection of my projects throughout the years, covering varied areas of my interest, including engineering, computer science, and a touch of art.

Overall, I make the things what I want to make. From projects that I strongly believe will leave an impact, to projects that I think are cool concepts to try out, whether grand or petit, solo or as a team. I find joy in putting my time towards a personally fulfilling and productive cause.

I hope you will enjoy them as I have.

Yiunfan Hu

In the Cover: Mobile GeoTrax Train (3D Printing, 2022)



Highlight

- 1. NHRL 12 lb. Combat Robot (Engineering)

- 2. Homemade Dual-touchscreen Laptop (Engineering + Computer Science)

- 3. Mosaic Maker (Art + Computer Science)

- 4. Dragon Curve Fractal Generator (Computer Science)

Personal Logo Figurehead (3D Printing, 2020)

NHRL 12 lb. Combat Robot (Engineering) [1/4] 2023 (17 years old)

YOB TROHS
12 lb. Combat Robot

A 12 lb. combat robot named YOB TROHS (SHORT BOY spelled backwards). Designed to compete in the National Havoc Robotics League (NHRL), built from scratch over 5 months.

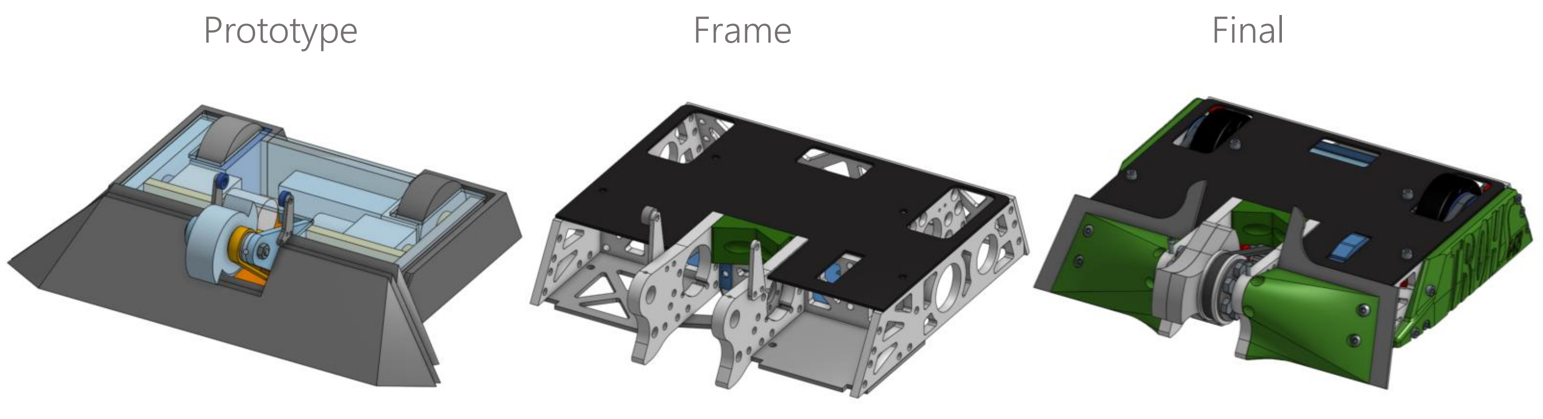
[Video Demo] [CAD File] [Source Code]

As a long-time BattleBots fan, I spearheaded this project and assembled a small team with friends from my school's Robotics team, to design and build a full 12 lb. combat robot from scratch in my house. As the team captain, I oversaw the project and was also in charge of robot design, and electronics. My teammates handled the majority of metal manufacturing and driving the robot.

Robot design: Via Onshape to take advantage of its collaborative features, viewable final design here.

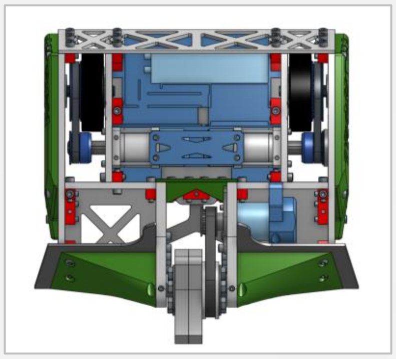
Leveraging the functionality, weight, cost, and the margin of error, I went with a two-wheel drive, vertical spinner, because of its robust and efficient design.

For space arrangement, as a robot is expensive and we were contributing all components out of pocket, the design is optimized for survivability. For this reason, the expensive drive modules, consisting of two BaneBots P6S gearboxes (16:1) with Cobra C-2808/30 Brushless Motor (Kv=1000), are placed in the very center of the robot. A ¼ inch belt with 3D-printed pulleys is positioned further back to drive the wheels. The space in the very back is thus freed up to house all the electronics in a single package.



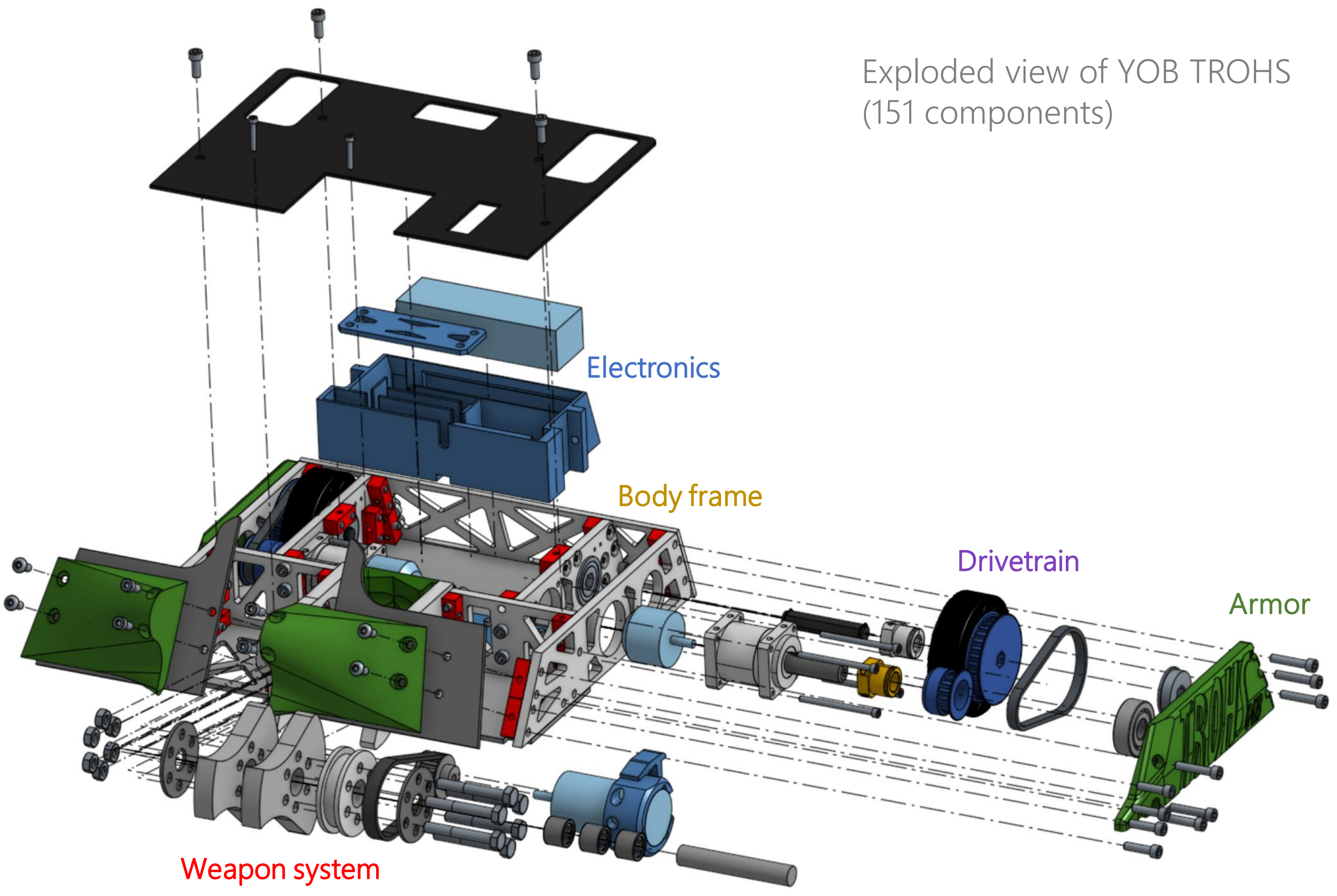
Models at different design stages, from prototyping, to outlining, and then finalization.

NHRL 12 lb. Combat Robot (Engineering) [2/4]



At major revisions, I reached out to members of the NHRL community for feedback and communicated to the team the potential modifications. The robot was given the name "YOB TROHS" after someone compared the design as a short version of another robot named YOB GNOL.

The robot includes total 151 components at the final design, comprising the body frame, armor, weapon system, drivetrain, and electronics. All these components from different systems went through the integration check before the physical build, virtually assembled into one robot without conflict in space assignment.

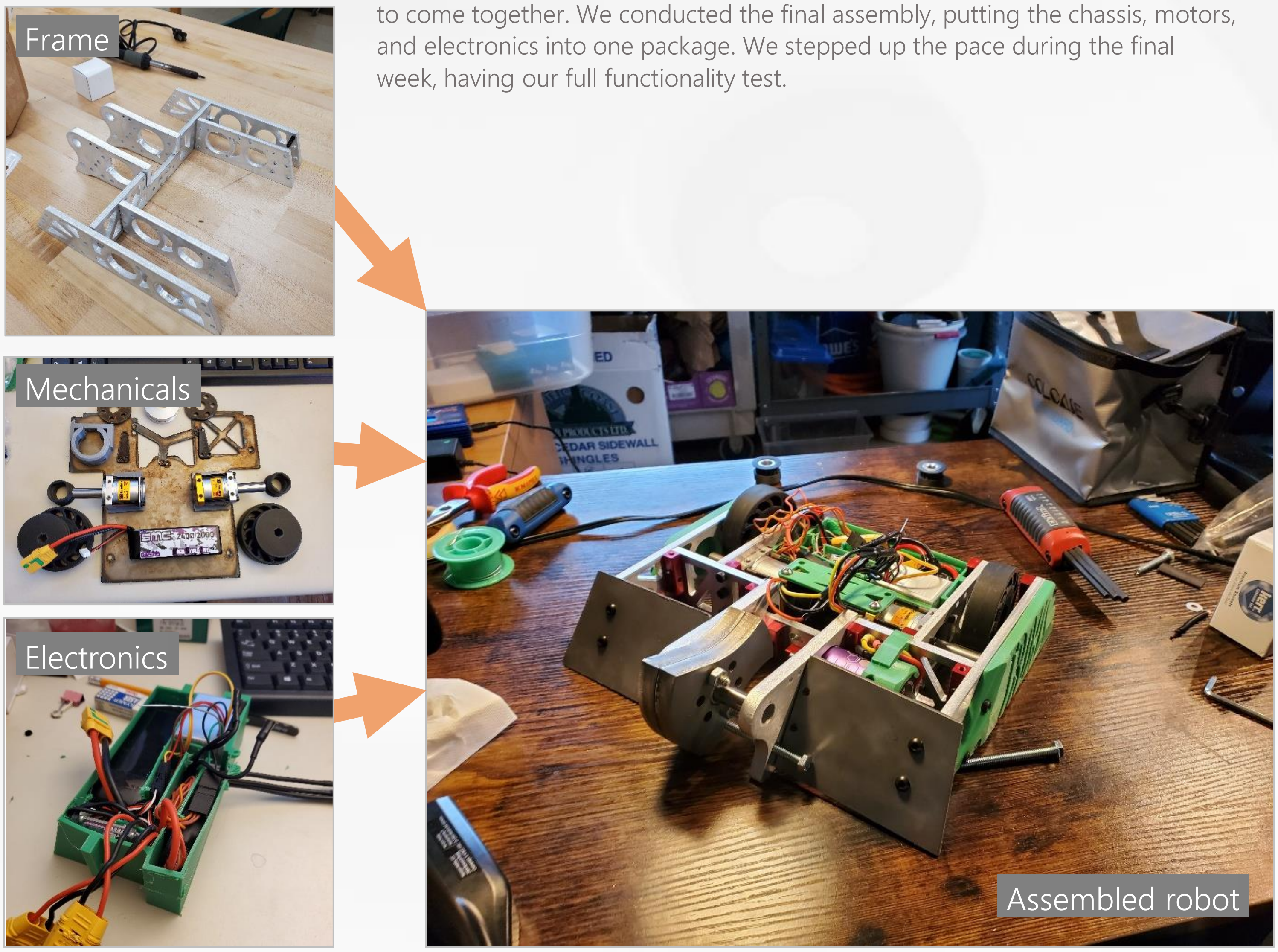


NHRL 12 lb. Combat Robot (Engineering) [3/4]

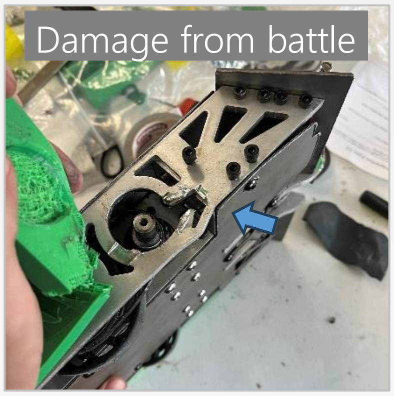
Robot building: The components were intended to be home-made as much as possible. I worked on printing everything we needed, utilizing exotic filaments with special properties, such as Carbon Fiber for its stiffness and strength and TPU for its flexibility and impact-absorbing properties. To work with metals, my team and I used the school machine shop. Complex parts were made on the lathe and mill, and I operated the plasma-cutter to make the armor.

I also worked on the electronics and code. I soldered all the connectors and joints inside the robot. To translate the controller inputs into our motors, I used an Arduino Nano microcontroller and wrote code. The steering and throttle PWM inputs from the RC receiver are converted into PWM outputs for the left and right drive motor speed controllers.

As the target event approached, our combat robot YOB TROHS was beginning to come together. We conducted the final assembly, putting the chassis, motors, and electronics into one package. We stepped up the pace during the final week, having our full functionality test.



NHRL 12 lb. Combat Robot (Engineering) [4/4]

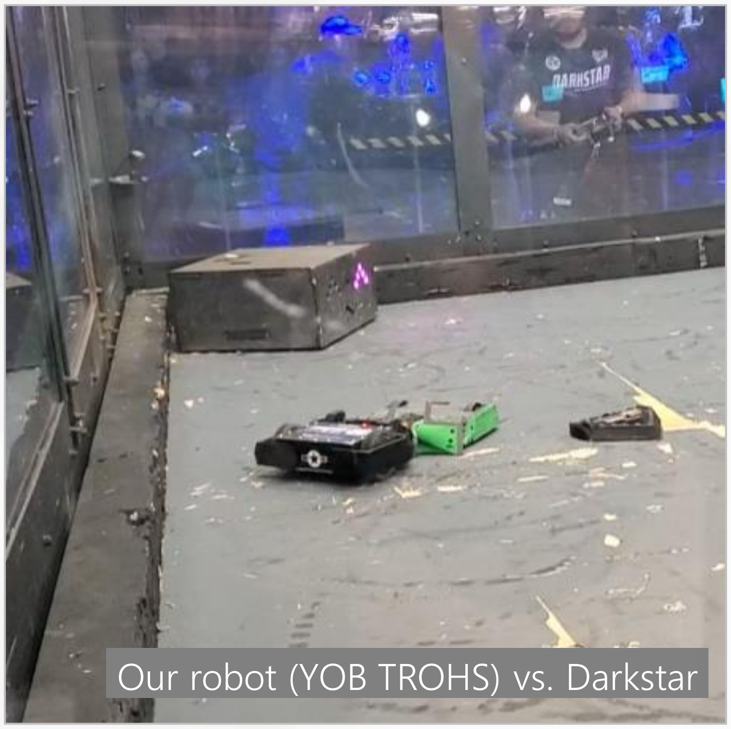

Damage from battle

Robot competition: The event was in September 2023, at Norwalk, CT. I traveled with the robot and arrived one day prior to the event, finalizing the robot before the mandatory safety inspection on the game day. YOB TROHS passed safety at a weight of 11.98 lbs., barely making the 12 lb. cut-off.

Many last-minute surprises and fixes awaited us. Despite all parts fully functioning in our tests at home, during one of our tests on-site, the weapon hit its own pulley and exploded. We suspected it was due to vibration loosening the screw holding it in, but we were forced to go into our first fight without a weapon. Our first match was Whomper, a terrifying horizontal spinner. The results was not good, Whomper had bent one of the frame sides, bending the gearbox axle underneath.

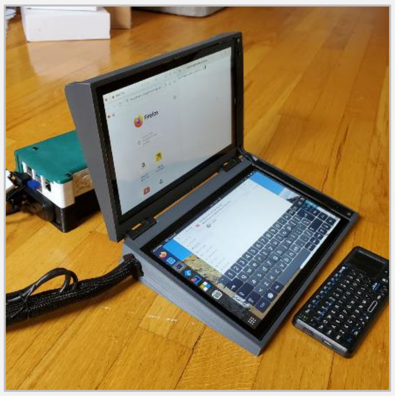
The builder community is a very friendly environment, and we got tips from many of the other competitors after the match. As we have never seen the cage floor before, we slightly overestimated how much grip we would get on the wooden floors. I was told this is a common rookie situation and recommended us to employ 4-wheel drive and cleats in the future. To repair our robot, I took the bent parts to the competitor machine shop and bent most of the components back into a usable shape with a press.

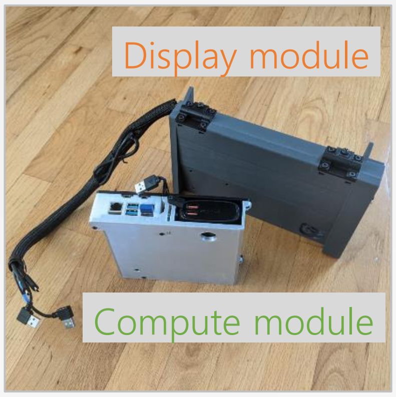
The second match was against Darkstar. The newly-repaired YOB TROHS went the whole 3 minutes and managed to engage Darkstar properly.



Our robot (YOB TROHS) vs. Darkstar

Homemade Dual-touchscreen Laptop (Engineering + Computer Science) [1/3] 2022 (16 years old)





A Raspberry Pi powered Laptop. Designed with a fully 3D printable case and modular electronics.

[Video Demo] [STL Files] [Source Code]

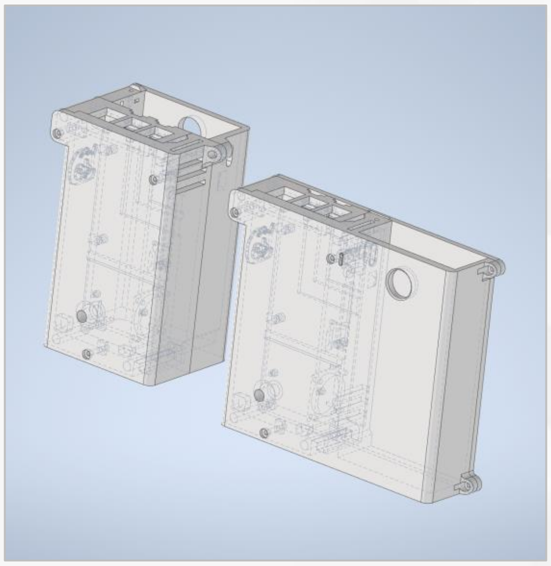
I was intrigued to create a computer device that is truly customized to my needs. It would have to be small, lightweight, fit inside my school bag, plus a large screen area to work on. With lack of variety in pre-built laptops, and as a challenge, I made my own, a self-built laptop.

Self-built laptops are fairly rare. This is mostly because compact computer parts are rarely sold, especially not in a form meant for DIY hobbyists. Thus, as a base to work on, the Raspberry Pi stood out as the best option available, being small, efficient, and reasonably powerful.

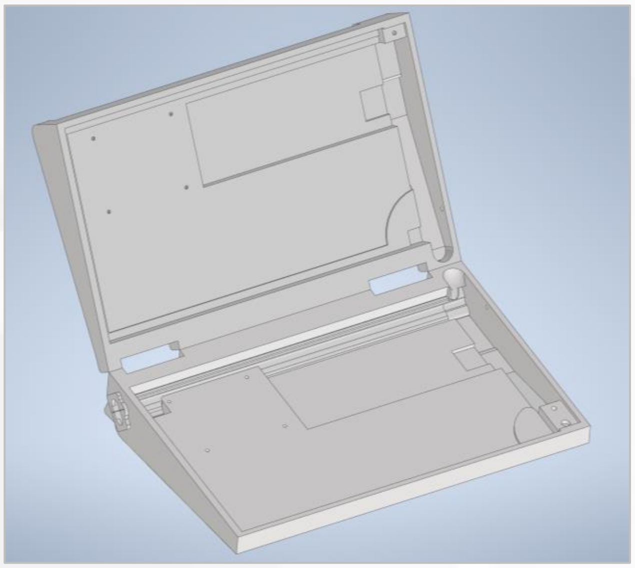
I composed my laptop with two modules. One is the compute module including a Raspberry Pi 4 powered by a 40000mAh battery and a breadboard for electronics. The other is the display module including two 10-inch touchscreens and an optional remote keyboard.

I designed and 3D printed the cases of both modules in a compact fashion. The design process was significantly more rigorous than it appears. Space efficiency was key. Every part had to fit perfectly using minimum space.

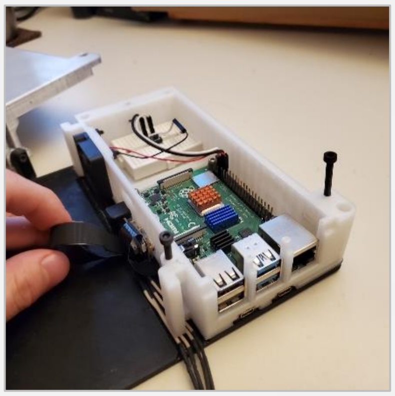
Compute module

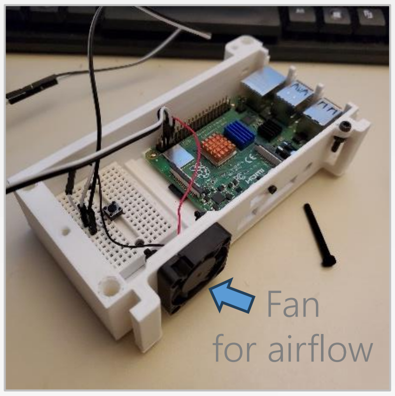


Display module



Homemade Dual-touchscreen Laptop (Engineering + Computer Science) [2/3]

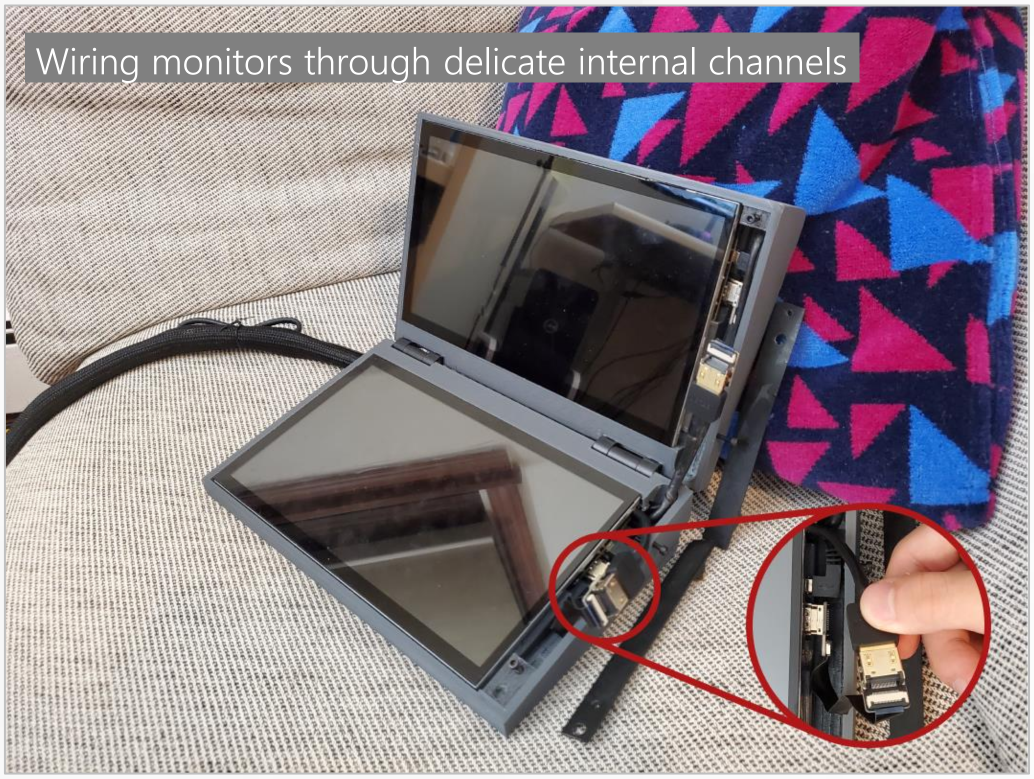




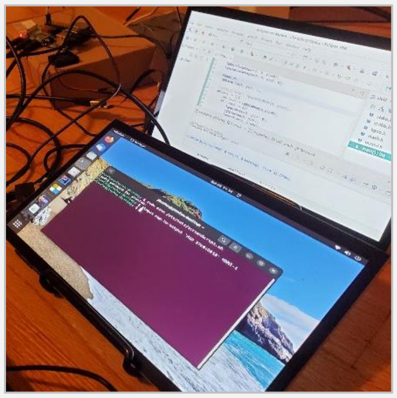
The case for the compute module was designed to be able to fit in my school bag. Every port and opening had to be on the top side to make it easy to use and unsusceptible to damage from what might be in my bag with it. This led to two challenges: airflow and wiring.

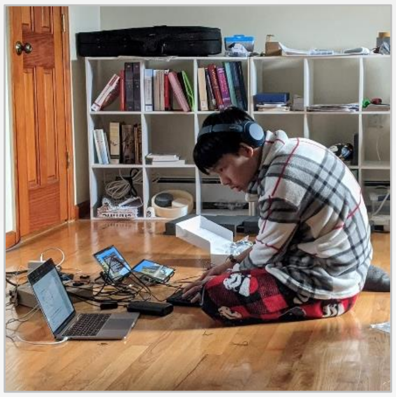
To adequately cool off the Raspberry Pi, I needed a fan and a way pushing air over the board. I settled on a ventilation shaft with a small computer fan pushing air into the main chamber. To attach the monitor and power cables to the Raspberry Pi, I needed to move the ports from the side to the top. I eventually solved this using FPC cables extending the port enough to reach the top side without expanding the profile at all.

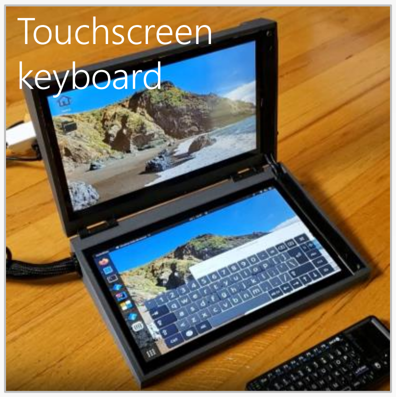
The shell of the display module was the most challenging to make and assemble, with the margin of error being extremely small. The sensitive touchscreens needed to be delicately pressed and secured into the shell. The wiring for both monitors also needed to be run through an internal channel and carefully linked up. To save the most space, I designed the monitors to sit at a slight angle inside the shell, with cables running through the channels on the right side. This also had the benefit of giving the screens a slight tilt towards the user.

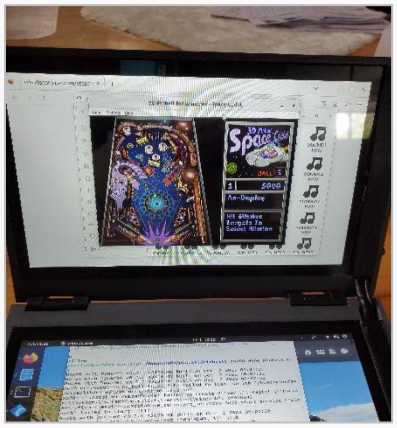


Homemade Dual-touchscreen Laptop (Engineering + Computer Science) [3/3]









Running iconic Microsoft 3D Pinball: Space Cadet

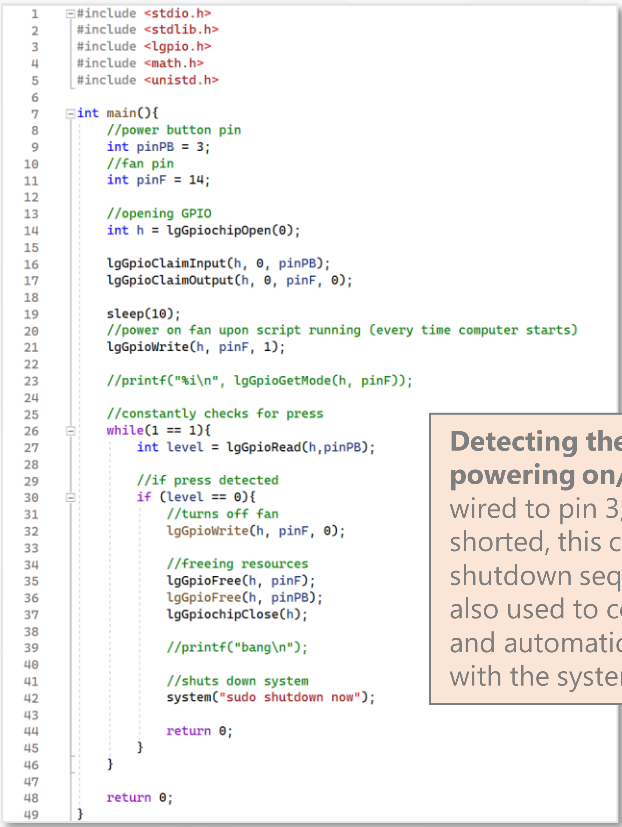
On the software front, I chose Ubuntu Linux, as it is fully capable of browsing the internet and supporting most development tools that a daily driver would need. When required, I can also emulate Windows applications using Wine and Box86.

With my unique hardware configuration, it took significant work to get Ubuntu fully functional. One of these problems was mapping the touchscreen input onto the correct screen and I spent many days and nights working on this one problem.

Taking advantage of the Raspberry Pi's structure, I designed my laptop as an expandable system which can be easily used as a component of future electronics projects. Through controlling the GPIO pins using the C programming language, I could interact with electronics I connected up to the computer. I used this functionality to install a power button and an automatic cooling fan.

While my home-made laptop is not outright comparable to an expensive commercial product, it only cost around $250 at the time and is much more useful being designed around my daily use-case. My design reached a great balance between economics, functionality, and my personal needs.

Laptop-Power-Script/PBScript.c | Line 1-49 (with C)

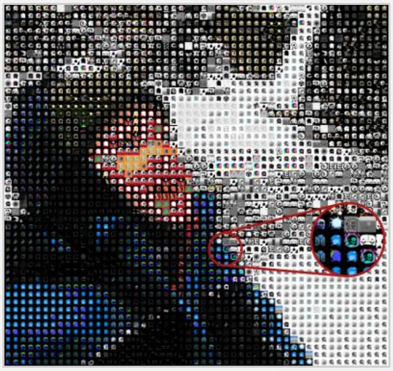


```c
#include <stdio.h>
#include <stdlib.h>
#include <lgpio.h>
#include <math.h>
#include <unistd.h>

int main(){
    //power button pin
    int pinPB = 3;
    //fan pin
    int pinF = 14;

    //opening GPIO
    int h = lgGpiochipOpen(0);

    lgGpioClaimInput(h, 0, pinPB);
    lgGpioClaimOutput(h, 0, pinF, 0);

    sleep(10);
    //power on fan upon script running (every time computer starts)
    lgGpioWrite(h, pinF, 1);

    //printf("%i\n", lgGpioGetMode(h, pinF));

    //constantly checks for press
    while(1 == 1){
        int level = lgGpioRead(h,pinPB);

        //if press detected
        if (level == 0){
            //turns off fan
            lgGpioWrite(h, pinF, 0);

            //freeing resources
            lgGpioFree(h, pinF);
            lgGpioFree(h, pinPB);
            lgGpiochipClose(h);

            //printf("bang\n");

            //shuts down system
            system("sudo shutdown now");

            return 0;
        }
    }

    return 0;
}
```

Detecting the power button and powering on/off. Using a button wired to pin 3, whenever that pin is shorted, this code activates a shutdown sequence. Another pin is also used to control a cooling fan, and automatically turns it on and off with the system.

Mosaic Maker (Art + Computer Science) [1/3]

2020 (14 years old)



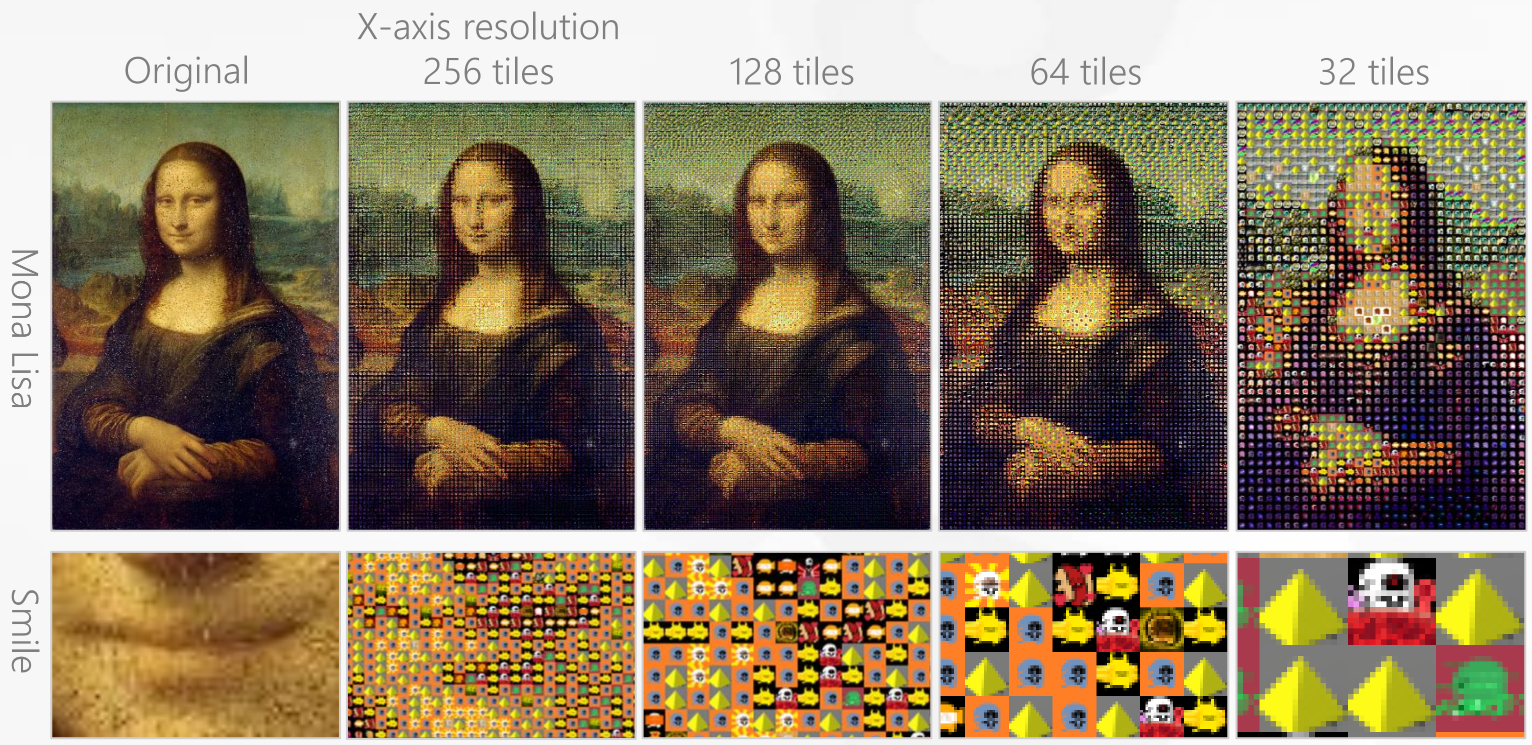
A customizable mosaic maker written with C#. Designed to convert any photos and videos into mosaic art composed of smaller image tiles.

[Video Demo] [Source Code]

I am interested to create an application that could make mosaic art composed of smaller tiles, with the flexibility to employ any set of images as the source of tiles. This was an effort to expand the usages of my image collection named "Geckoimages", which includes >1000 variants of my personal logo, dubbed "Gecko".

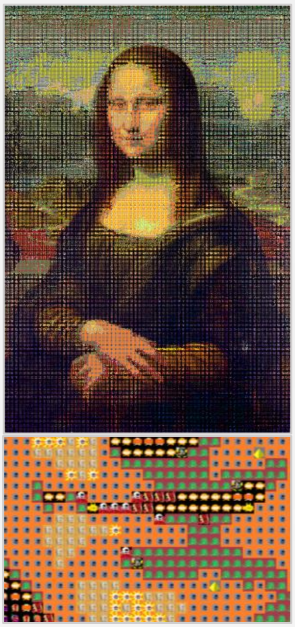
The conversion process takes two steps. First, down-sample the photo to a user-specified resolution. Then second, replace each pixel by an image tile that has the closest average RGB color value to the original, found by calculating the closest overall distance in 3 axes of red, green, and blue.

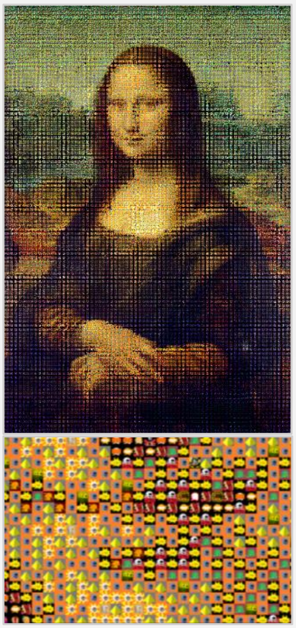
I also introduced batch processing and multithreading to be able to process videos. The code is written in C# and built upon my experience and interest in image processing.



Output with different specified resolutions

Mosaic Maker (Art + Computer Science) [2/3]

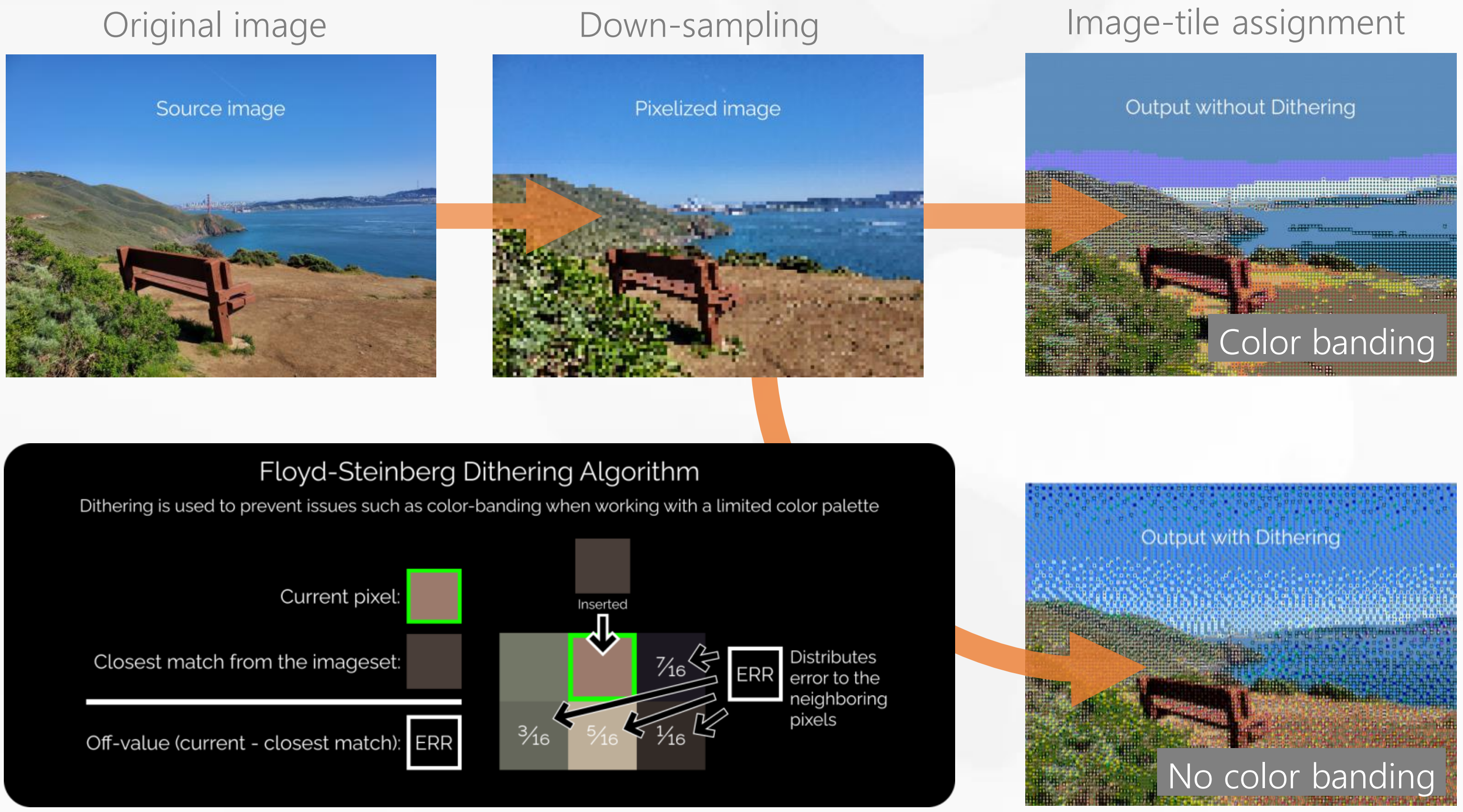

Color banding


Fixed

A potential processing issue is color banding. It occurs when high color count is forced into a smaller color palette. For example, a gradient made of 20 colors, when forced into a 2-color palette, becomes 2 bands. I then applied a Floyd-Steinberg Dithering algorithm to tackle this issue.

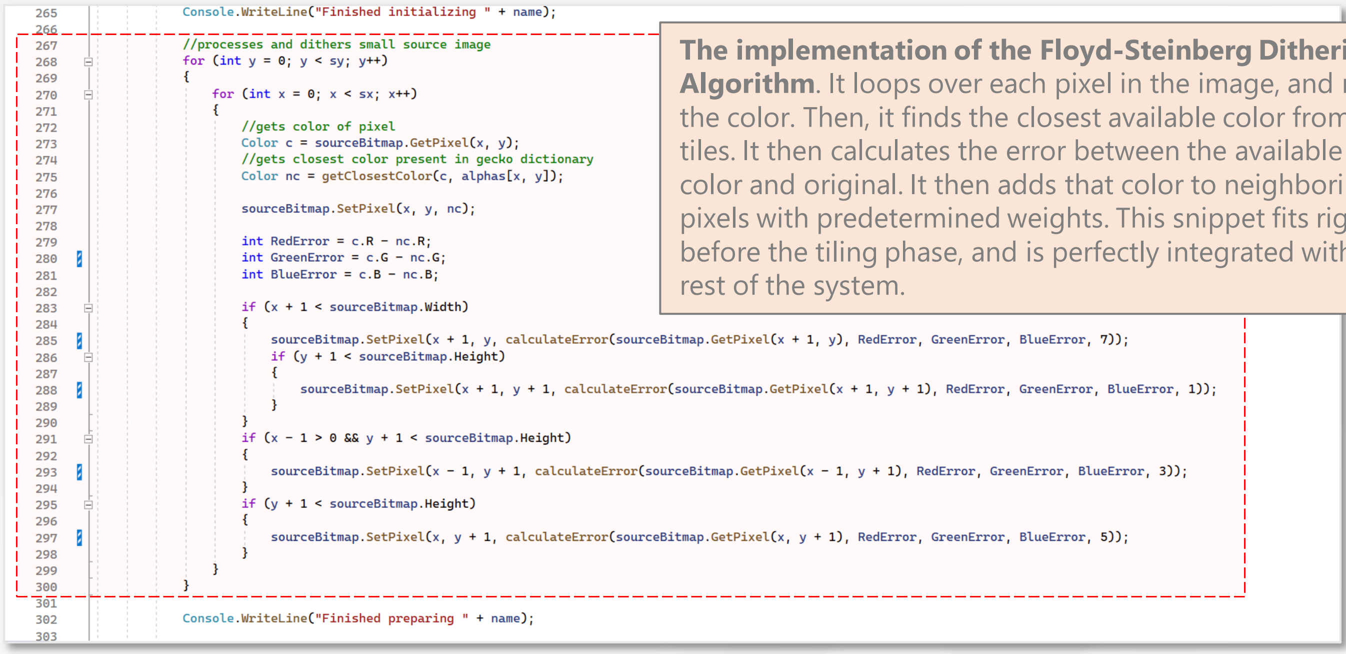
Dithering in graphics is a technique using a combination of colors placed next to each other to create an illusion of an intended color. This helps eliminate color banding by smoothing the transition between two colors. It also increases the accuracy of the colors in the output.

Floyd-Steinberg dithering works by scanning through an image pixel by pixel. On each pixel, the difference between the original color and the closest available color is added to the neighboring pixels in a predetermined pattern. Thus, instead of the error being lost with each pixel, it is preserved in other pixels. I inserted Floyd-Steinberg Dithering as a step before replacing the pixels. And the problem was neatly fixed.


Original image
Source image
Down-sampling
Pixelized image
Image-tile assignment
Output without Dithering
Color banding
Floyd-Steinberg Dithering Algorithm
Dithering is used to prevent issues such as color-banding when working with a limited color palette
Current pixel:
Closest match from the imageset:
Off-value (current - closest match): ERR
Inserted
7⁄16
3⁄16
5⁄16
1⁄16
ERR
Distributes error to the neighboring pixels
Output with Dithering
No color banding

Mosaic Maker (Art + Computer Science) [3/3]

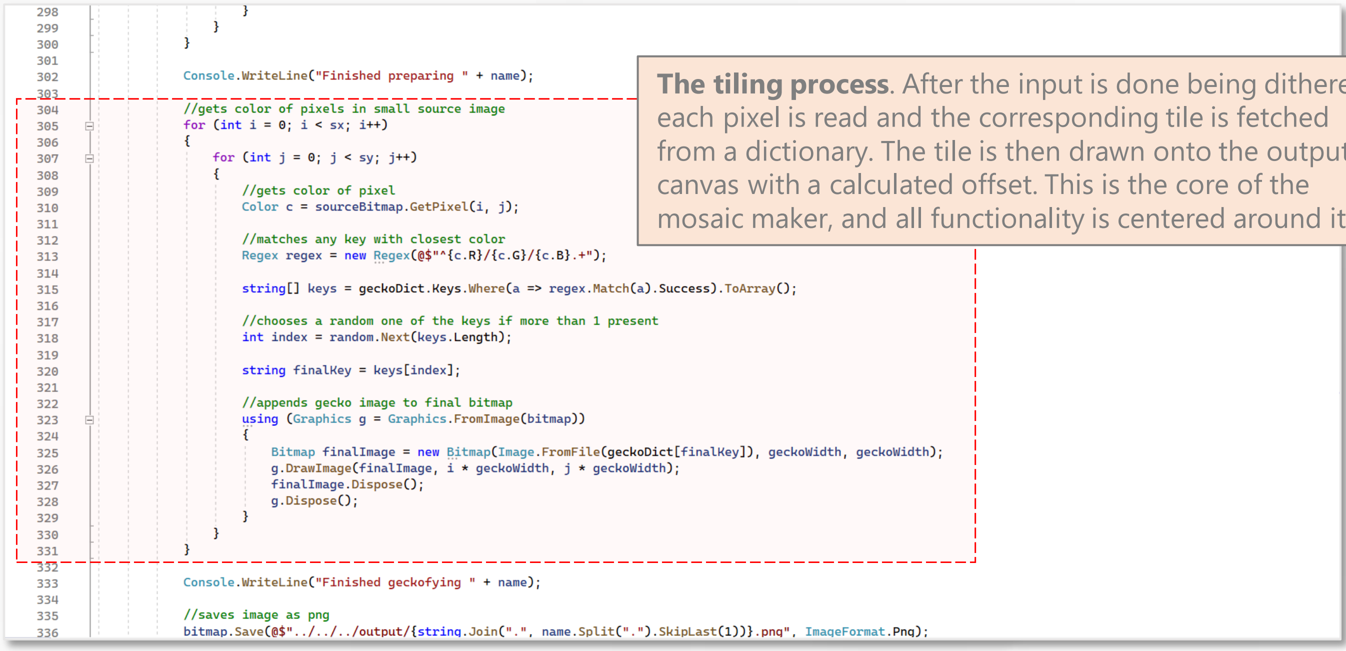
Geckoinator9000/Program.cs | Line 267-300 (with C#)



```csharp
265            Console.WriteLine("Finished initializing " + name);
266
267            //processes and dithers small source image
268            for (int y = 0; y < sy; y++)
269            {
270                for (int x = 0; x < sx; x++)
271                {
272                    //gets color of pixel
273                    Color c = sourceBitmap.GetPixel(x, y);
274                    //gets closest color present in gecko dictionary
275                    Color nc = getClosestColor(c, alphas[x, y]);
276
277                    sourceBitmap.SetPixel(x, y, nc);
278
279                    int RedError = c.R - nc.R;
280                    int GreenError = c.G - nc.G;
281                    int BlueError = c.B - nc.B;
282
283                    if (x + 1 < sourceBitmap.Width)
284                    {
285                        sourceBitmap.SetPixel(x + 1, y, calculateError(sourceBitmap.GetPixel(x + 1, y), RedError, GreenError, BlueError, 7));
286                        if (y + 1 < sourceBitmap.Height)
287                        {
288                            sourceBitmap.SetPixel(x + 1, y + 1, calculateError(sourceBitmap.GetPixel(x + 1, y + 1), RedError, GreenError, BlueError, 1));
289                        }
290                    }
291                    if (x - 1 > 0 && y + 1 < sourceBitmap.Height)
292                    {
293                        sourceBitmap.SetPixel(x - 1, y + 1, calculateError(sourceBitmap.GetPixel(x - 1, y + 1), RedError, GreenError, BlueError, 3));
294                    }
295                    if (y + 1 < sourceBitmap.Height)
296                    {
297                        sourceBitmap.SetPixel(x, y + 1, calculateError(sourceBitmap.GetPixel(x, y + 1), RedError, GreenError, BlueError, 5));
298                    }
299                }
300            }
301
302            Console.WriteLine("Finished preparing " + name);
303
```

The implementation of the Floyd-Steinberg Dithering Algorithm. It loops over each pixel in the image, and reads the color. Then, it finds the closest available color from the tiles. It then calculates the error between the available color and original. It then adds that color to neighboring pixels with predetermined weights. This snippet fits right before the tiling phase, and is perfectly integrated with the rest of the system.

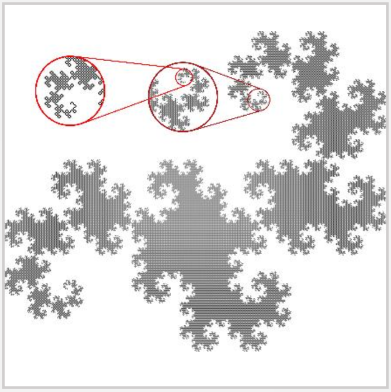
Geckoinator9000/Program.cs | Line 304-331 (with C#)



```csharp
298                    }
299                }
300            }
301
302            Console.WriteLine("Finished preparing " + name);
303
304            //gets color of pixels in small source image
305            for (int i = 0; i < sx; i++)
306            {
307                for (int j = 0; j < sy; j++)
308                {
309                    //gets color of pixel
310                    Color c = sourceBitmap.GetPixel(i, j);
311
312                    //matches any key with closest color
313                    Regex regex = new Regex(@$"^{c.R}/{c.G}/{c.B}.+");
314
315                    string[] keys = geckoDict.Keys.Where(a => regex.Match(a).Success).ToArray();
316
317                    //chooses a random one of the keys if more than 1 present
318                    int index = random.Next(keys.Length);
319
320                    string finalKey = keys[index];
321
322                    //appends gecko image to final bitmap
323                    using (Graphics g = Graphics.FromImage(bitmap))
324                    {
325                        Bitmap finalImage = new Bitmap(Image.FromFile(geckoDict[finalKey]), geckoWidth, geckoWidth);
326                        g.DrawImage(finalImage, i * geckoWidth, j * geckoWidth);
327                        finalImage.Dispose();
328                        g.Dispose();
329                    }
330                }
331            }
332
333            Console.WriteLine("Finished geckofying " + name);
334
335            //saves image as png
336            bitmap.Save(@$"../../../output/{string.Join(".", name.Split(".").SkipLast(1))}.png", ImageFormat.Png);
```

The tiling process. After the input is done being dithered, each pixel is read and the corresponding tile is fetched from a dictionary. The tile is then drawn onto the output canvas with a calculated offset. This is the core of the mosaic maker, and all functionality is centered around it.

Dragon Curve Fractal Generator (Computer Science) [1/3] 2022 (16 years old)



A dragon curve fractal generator written with C and C#. Designed to generate dragon curves with specified iteration numbers.

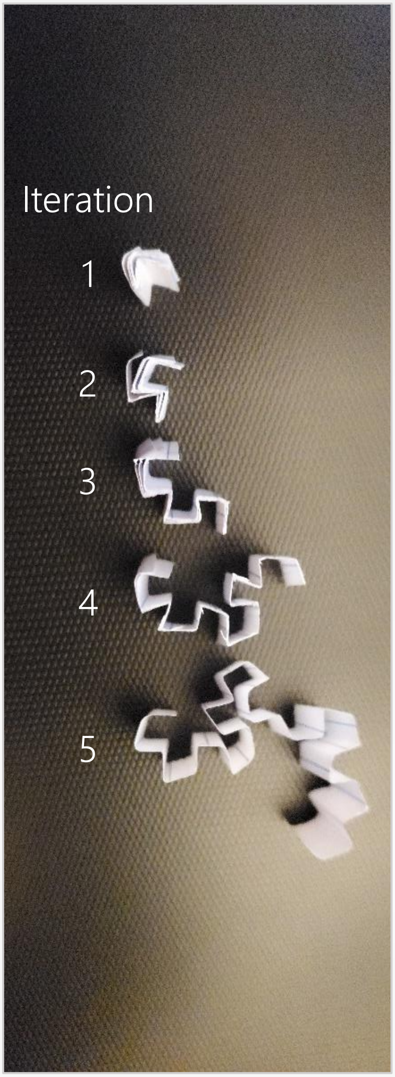
[Video Demo] [Source Code]

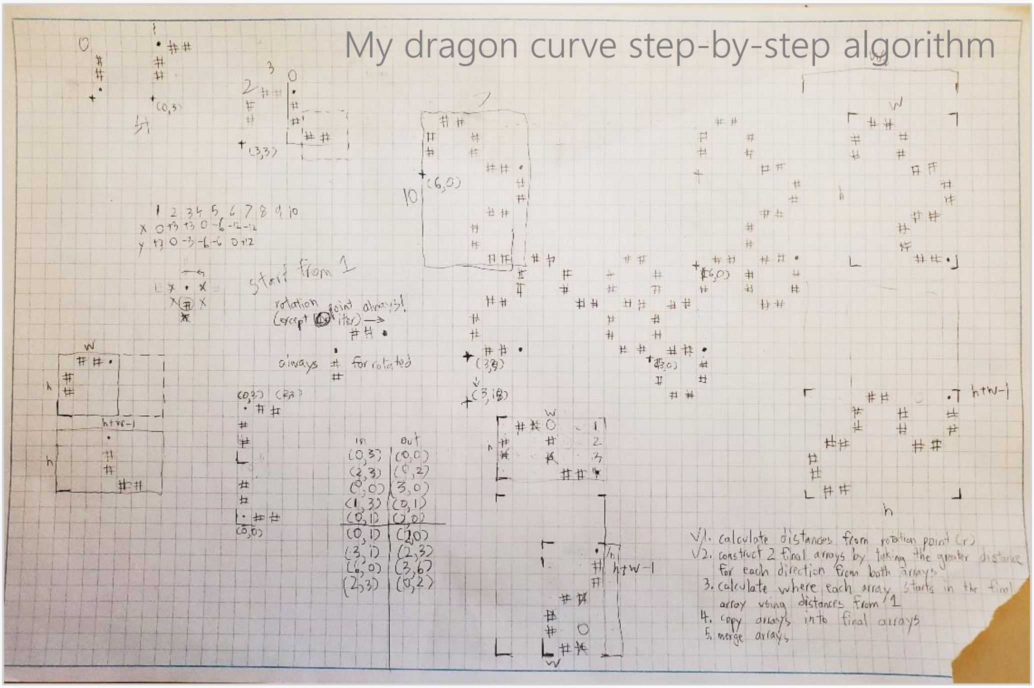
After I was introduced to the concept of recursion in CS50x, I became intrigued to use my new knowledge of recursion to make fractals. After a pilot run making a simple Sierpinski's Triangle, I aimed for the dragon curve to recreate using C.

One of the classic and most intuitive ways to produce a dragon curve is to repeatedly fold a piece of paper in half in the same direction, then unfold it at right angles. It was this idea that inspired my approach.

To emulate unfolding a piece of paper into a right angle, I start with a line. A copy of that line is created, rotated 90 degrees, then attached to the original at a rotation point. Taking the output, the process can be repeated on it, and so on... This is the core concept this project is built on.

To convert my idea into code, I drew out my thinking onto paper. I noticed some patterns in the iterations when I drew them out, and came up with a step-by-step algorithm to get the job done.



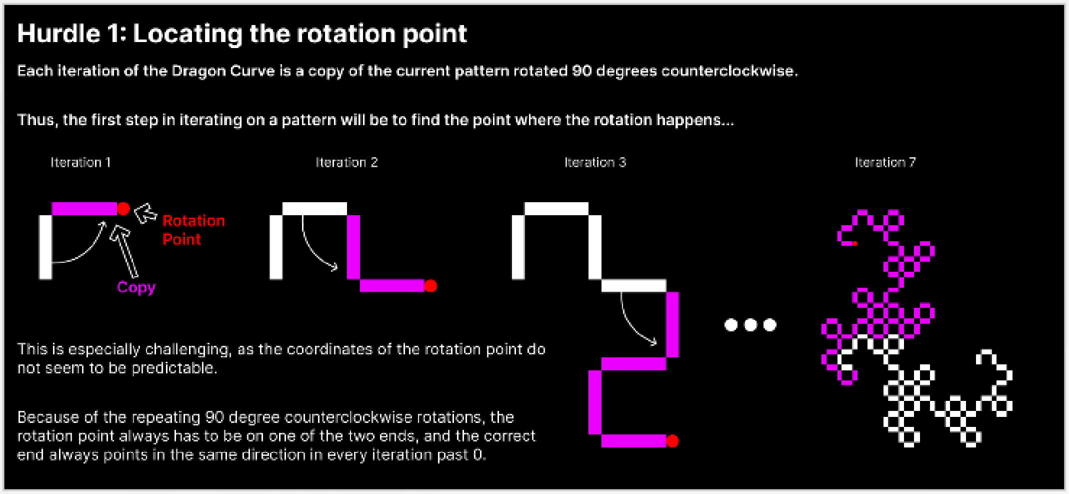


My dragon curve step-by-step algorithm

√1. calculate distances from rotation point (r)
√2. construct 2 final arrays by taking the greater distance for each direction from both arrays
3. calculate where each array starts in the final array using distances from 1
4. copy arrays into final arrays
5. merge arrays

Dragon Curve Fractal Generator (Computer Science) [2/3]

The resulting algorithm in the final code consists of three major hurdles.



Hurdle 1: Locating the rotation point

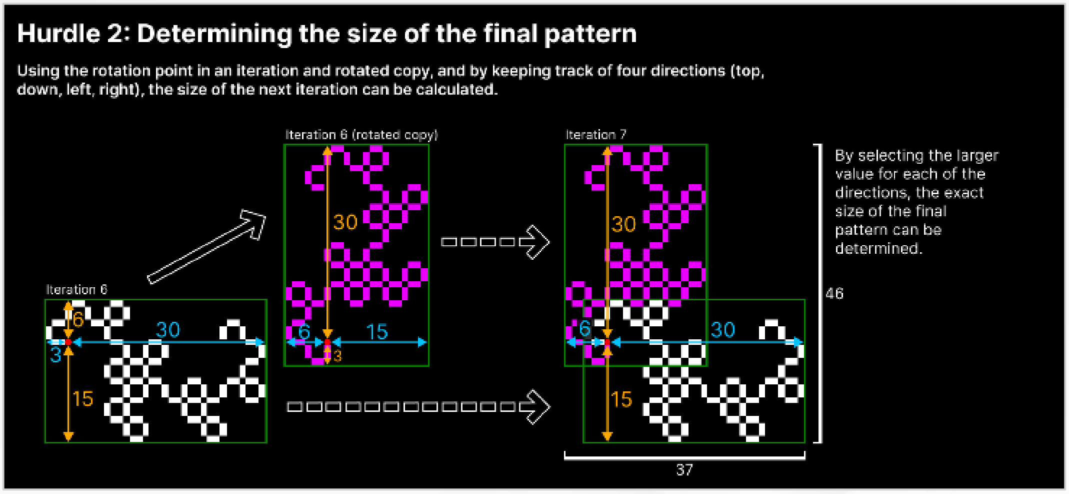
Each iteration of the Dragon Curve is a copy of the current pattern rotated 90 degrees counterclockwise.

Thus, the first step in iterating on a pattern will be to find the point where the rotation happens...

This is especially challenging, as the coordinates of the rotation point do not seem to be predictable.

Because of the repeating 90 degree counterclockwise rotations, the rotation point always has to be on one of the two ends, and the correct end always points in the same direction in every iteration past 0.

First, the location of the rotation point is needed to know where to merge the pieces together.

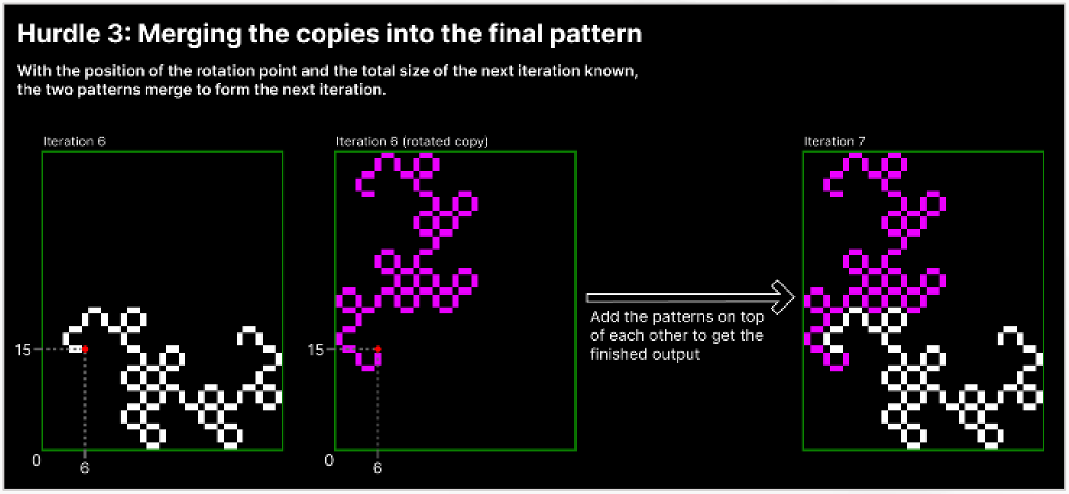


Hurdle 2: Determining the size of the final pattern

Using the rotation point in an iteration and rotated copy, and by keeping track of four directions (top, down, left, right), the size of the next iteration can be calculated.

By selecting the larger value for each of the directions, the exact size of the final pattern can be determined.

Next, the size of the final output is needed to create a container large enough to house it.



Hurdle 3: Merging the copies into the final pattern

With the position of the rotation point and the total size of the next iteration known, the two patterns merge to form the next iteration.

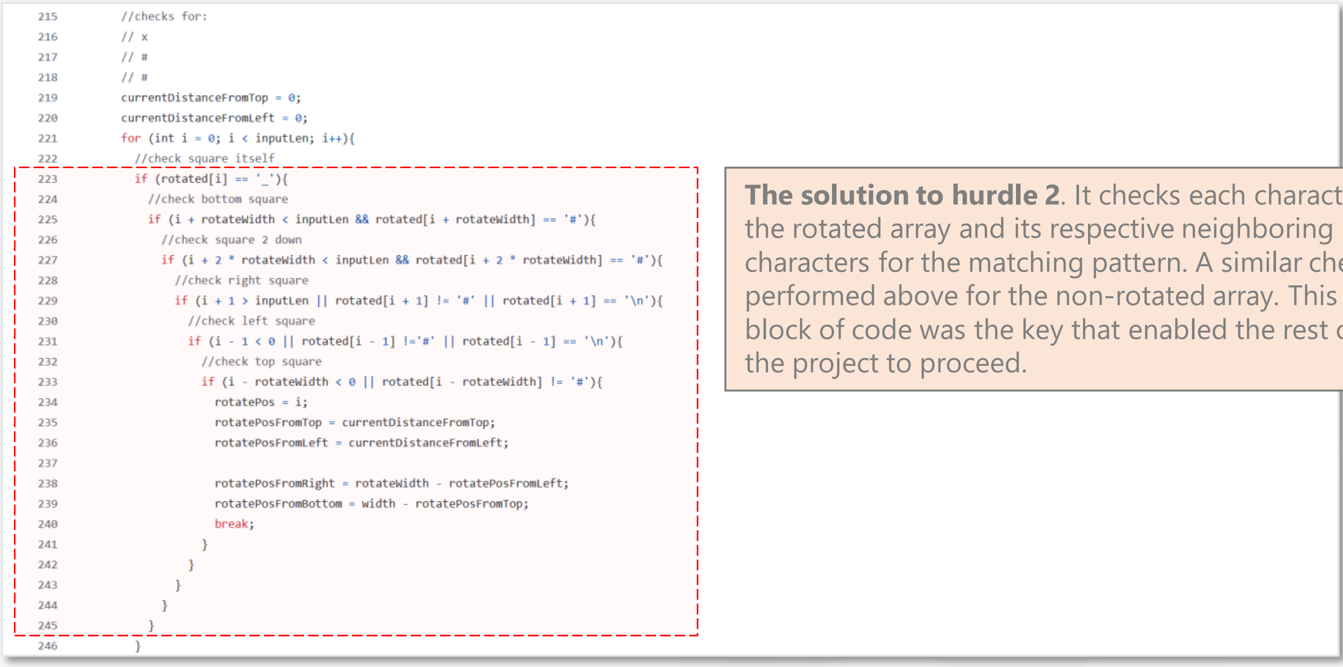
Add the patterns on top of each other to get the finished output

With both pieces of information, the original and copy can be joined together to create the output.

As the final step, I made a C# script that converted the ASCII outputs of the C code into pixels in an image which greatly improved their viewability.

Dragon Curve Fractal Generator (Computer Science) [3/3]

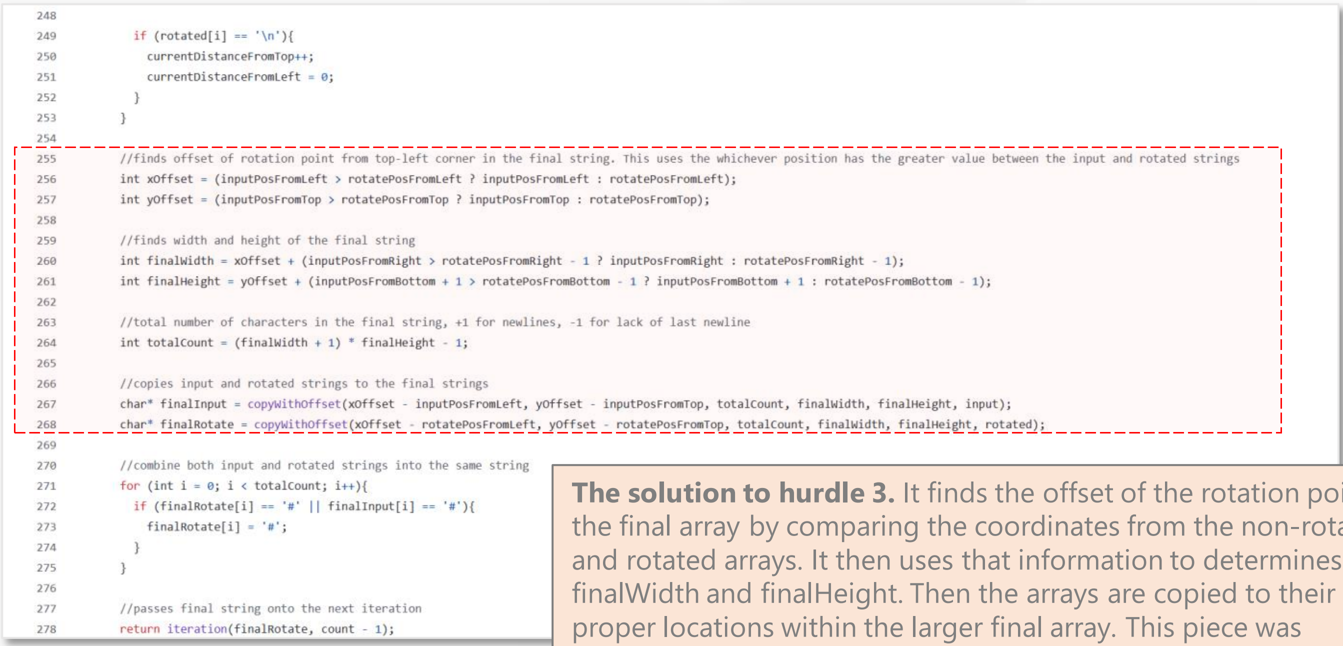
C-experiments/dragon/dragon.c | Line 223-245 (with C)



```c
215        //checks for:
216        // x
217        // #
218        // #
219        currentDistanceFromTop = 0;
220        currentDistanceFromLeft = 0;
221        for (int i = 0; i < inputLen; i++){
222          //check square itself
223          if (rotated[i] == '_'){
224            //check bottom square
225            if (i + rotateWidth < inputLen && rotated[i + rotateWidth] == '#'){
226              //check square 2 down
227              if (i + 2 * rotateWidth < inputLen && rotated[i + 2 * rotateWidth] == '#'){
228                //check right square
229                if (i + 1 > inputLen || rotated[i + 1] != '#' || rotated[i + 1] == '\n'){
230                  //check left square
231                  if (i - 1 < 0 || rotated[i - 1] !='#' || rotated[i - 1] == '\n'){
232                    //check top square
233                    if (i - rotateWidth < 0 || rotated[i - rotateWidth] != '#'){
234                      rotatePos = i;
235                      rotatePosFromTop = currentDistanceFromTop;
236                      rotatePosFromLeft = currentDistanceFromLeft;
237
238                      rotatePosFromRight = rotateWidth - rotatePosFromLeft;
239                      rotatePosFromBottom = width - rotatePosFromTop;
240                      break;
241                    }
242                  }
243                }
244              }
245            }
246          }
```

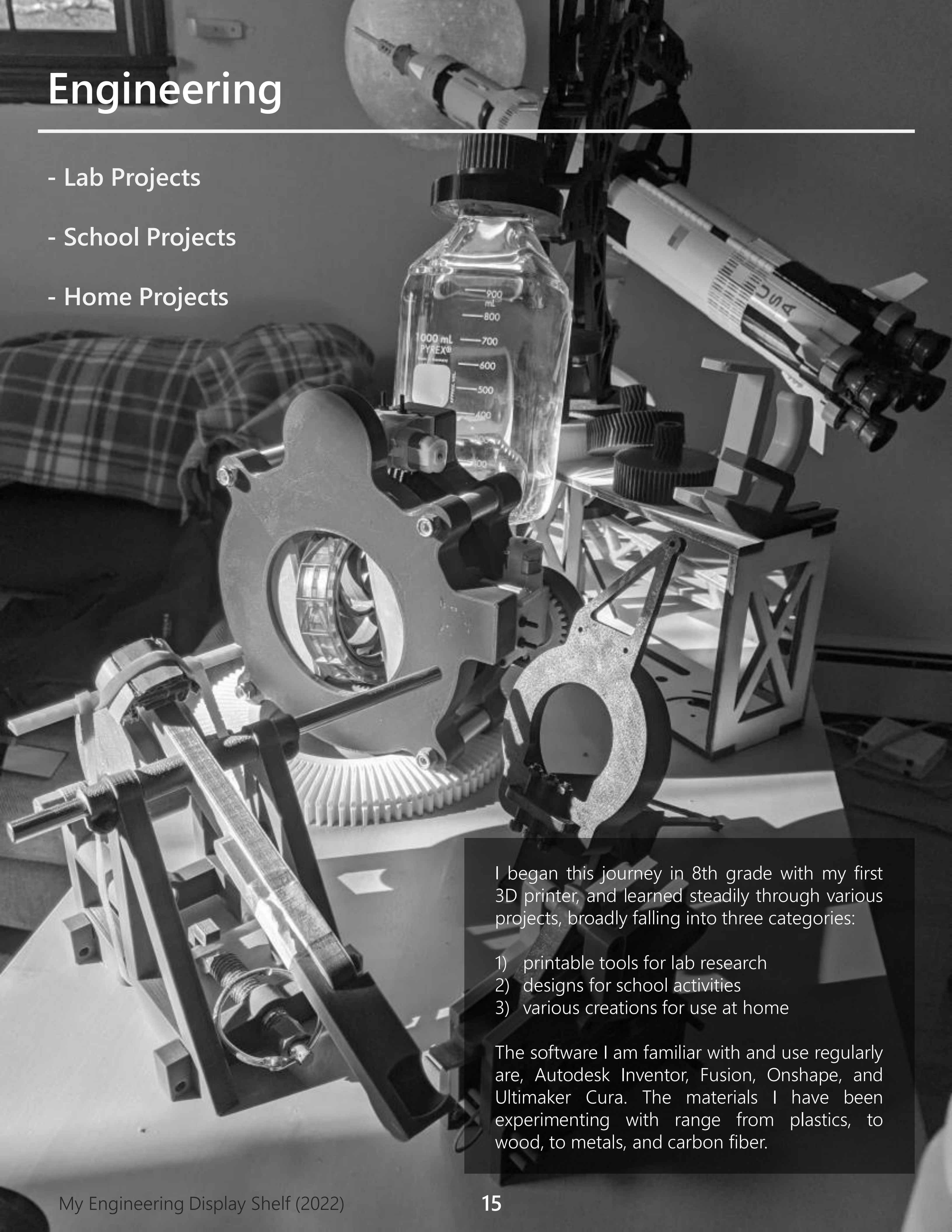
The solution to hurdle 2. It checks each character in the rotated array and its respective neighboring characters for the matching pattern. A similar check is performed above for the non-rotated array. This block of code was the key that enabled the rest of the project to proceed.

C-experiments/dragon/dragon.c | Line 255-268 (with C)



```c
248
249          if (rotated[i] == '\n'){
250            currentDistanceFromTop++;
251            currentDistanceFromLeft = 0;
252          }
253        }
254
255        //finds offset of rotation point from top-left corner in the final string. This uses the whichever position has the greater value between the input and rotated strings
256        int xOffset = (inputPosFromLeft > rotatePosFromLeft ? inputPosFromLeft : rotatePosFromLeft);
257        int yOffset = (inputPosFromTop > rotatePosFromTop ? inputPosFromTop : rotatePosFromTop);
258
259        //finds width and height of the final string
260        int finalWidth = xOffset + (inputPosFromRight > rotatePosFromRight - 1 ? inputPosFromRight : rotatePosFromRight - 1);
261        int finalHeight = yOffset + (inputPosFromBottom + 1 > rotatePosFromBottom - 1 ? inputPosFromBottom + 1 : rotatePosFromBottom - 1);
262
263        //total number of characters in the final string, +1 for newlines, -1 for lack of last newline
264        int totalCount = (finalWidth + 1) * finalHeight - 1;
265
266        //copies input and rotated strings to the final strings
267        char* finalInput = copyWithOffset(xOffset - inputPosFromLeft, yOffset - inputPosFromTop, totalCount, finalWidth, finalHeight, input);
268        char* finalRotate = copyWithOffset(xOffset - rotatePosFromLeft, yOffset - rotatePosFromTop, totalCount, finalWidth, finalHeight, rotated);
269
270        //combine both input and rotated strings into the same string
271        for (int i = 0; i < totalCount; i++){
272          if (finalRotate[i] == '#' || finalInput[i] == '#'){
273            finalRotate[i] = '#';
274          }
275        }
276
277        //passes final string onto the next iteration
278        return iteration(finalRotate, count - 1);
```

The solution to hurdle 3. It finds the offset of the rotation point in the final array by comparing the coordinates from the non-rotated and rotated arrays. It then uses that information to determines finalWidth and finalHeight. Then the arrays are copied to their proper locations within the larger final array. This piece was logically the most complex part of the program.



Engineering

- Lab Projects

- School Projects

- Home Projects

I began this journey in 8th grade with my first 3D printer, and learned steadily through various projects, broadly falling into three categories:

1) printable tools for lab research
2) designs for school activities
3) various creations for use at home

The software I am familiar with and use regularly are, Autodesk Inventor, Fusion, Onshape, and Ultimaker Cura. The materials I have been experimenting with range from plastics, to wood, to metals, and carbon fiber.

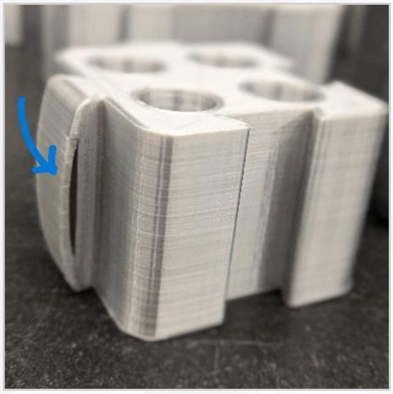
My Engineering Display Shelf (2022)

Engineering | Lab Projects

3D printable alternatives to often overpriced lab equipment. In the spirit of the Open Design Movement and Frugal Engineering, this project (as a whole, rather than individual designs) is to make research accessible for the general public and customizable for specialists. I reach these two targets via my dad's lab page under the alias "Bod Owens".

This project has been an outlet for my engineering creativity and a window for social impact. *I receive requests from researchers, develop specific solutions, and then release it as a free resource*. Important to note, due to the nature of this project, I practice a distinct design philosophy here by *prioritizing simplicity and printability over aesthetics and complexity*, to ensure the design could be robustly printed with even low-end FDM printers using PLA in potential target users' hand.

Spring Dovetail Blocks 2022 (16 years old)



A modularized tube rack set based on my novel joint design, the Spring Dovetail. Designed to be freely assembled by need and disassembled with ease.

[Video Demo] [STL Files]

Targeted Problem: 3D-printable objects are difficult to be properly connected with each other. Natural variation exists in 3D printing process, causing a critical issue when connecting two objects with joints is required. This also limits the application of modularized design, as the joints would be mostly too loose (cannot engage) or too tight (cannot disengage; imagine separating two tightly connect Lego bricks).

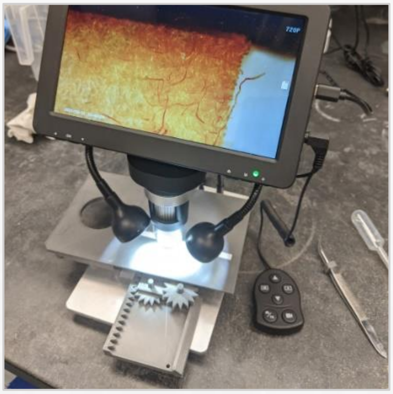
Designed Solution: I designed a new universal joinery for modularization of 3D-printed items, by adding my own touch on the classic dovetail joint of woodworking. A key curved spring layer is integrated into the junction to buffer printing errors and prevent overly tight/loose engagement.

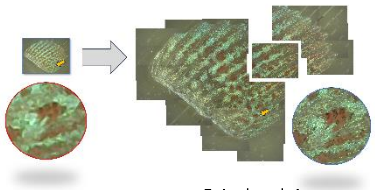
A good application of my Spring Dovetail design is for a system of modularized tube rack sets. Existing commercial products consist of a single molded part with large footprint. I thus designed modularized racks which can connect together via Spring Dovetail. The spring tension is fine-tuned to be just right so the block units could be connected and disconnected without much hassle and still stay together. This feature also allows the blocks to serve as a flexible base system for modularization, providing fixed distance and angles between components in a complex system.

The original editable files are also shared with the public under Creative Commons Licensing. Not only they are used by many, but also some derived versions are customized by the community, including integration into a commercial product made by a startup company.

Scope Gliding Table

2022 (16 years old)





A gliding table that can be used to reposition samples under microscopes. Designed to provides a precise and convenient way to move samples on a microscope stage.

[Video Demo] [STL Files]

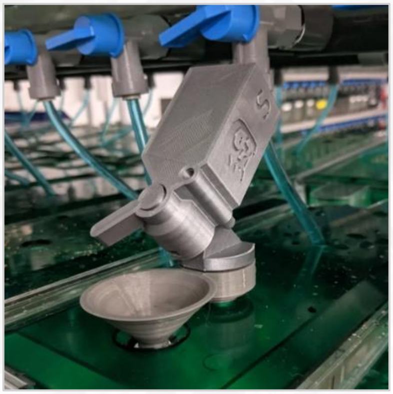
Targeted Problem: Consumer-grade digital microscopes do not have gliding table to move samples in X-Y axis. A gliding table is fundamental for research-grade microscopes, but rarely seen in consumer grade microscopes such as the ones for elementary school classrooms.

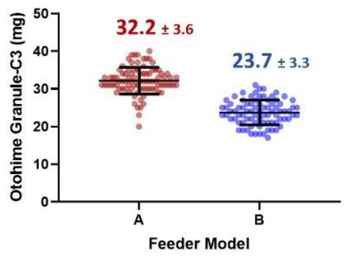
Designed Solution: I designed a 3D printable microscope gliding table. It includes an optional extension table for transmitted light scopes. The design was optimized to minimize friction by using sharp guide rails. This ensures both minimal contact and shaking. A pair of knobs are included to ease the pinion operation.

The gliding table enables a $100 camera scope from Amazon to match expensive research counterparts by making & stitching high-resolution images. By moving the sample horizontally and vertically, individual images will be in consistent angles and magnification, and can be easily stitched into a big image without the trade-off of resolution.

Feeding Apparatus

2021 (15 years old)





Consistency test

A modularized feeding apparatus that can be used to feed aquatic research animals in lab. Designed to be capable of depositing a consistent amount of food with ease.

[Video Demo] [STL Files]

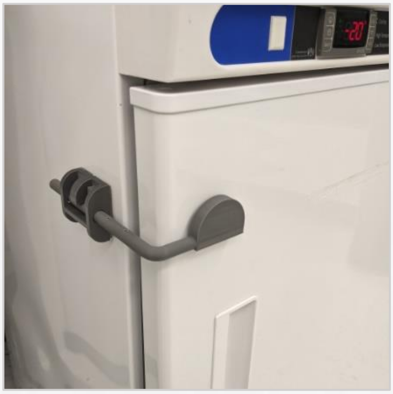
Targeted Problem: Feeding animals for research is critical for experiments but time-consuming, frequently a dilemma to choose between manual feeding (affordable but inconsistent) and automatic feeders (precise but very expensive).

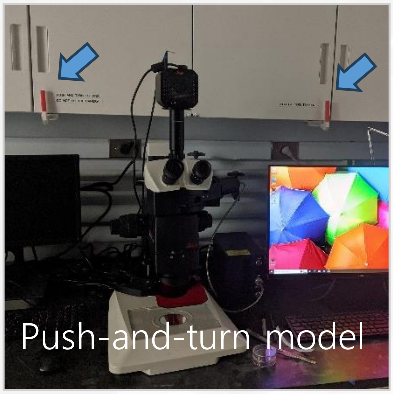
Designed Solution: I designed manual feeders as a good middle ground solution between full manual feeding and automatic feeders. Make feeding easier, faster, and more consistent.

The whole apparatus is comprised of a funnel which connects to the tank, an adapter, and the single-print feeder. The funnels fit various size of tanks, providing a universal base for the feeder. The feeder stores food in an upper hopper, which feeds into a rotating compartment. When the lever is pushed, this compartment rotates and dispenses the food. The inlet and outlet of the rotating compartment is offset in such a way that food is only allowed to either flow in or flow out, but never at the same time. This guarantees that the amount of food dispensed per lever push is consistent. Two models with different feeding amounts are available.

Door Lock

2022 (16 years old)





A latch lock that can be used to secure fridges, freezers, and cabinets. Designed to have an adjustable arm with two variants to fit any fridge, freezer, or cabinet.

[Video Demo] [STL Files]

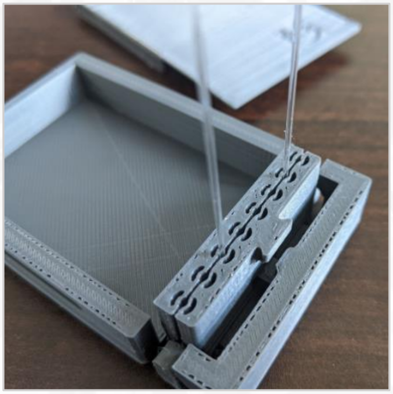
Targeted Problem: Freezers full of reagents and samples might be accidentally left with an unclosed door. A nightmare situation in the lab.

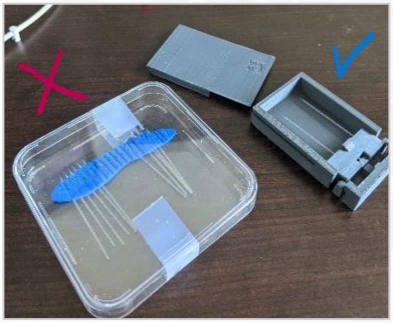
Designed Solution: I designed this latch to ensure a door is properly shut and stays shut. Each unit consists of a base which mounts to the side of the target object, an L shaped bar to lock the fridge, and a stopper to adjust the working length of the bar. It only cost 43 cents to make.

Later, I added a new variant for the opposite purpose, to prevent a door from being accidentally opened. The new design added a '*push-and-turn*" unlocking mechanism. Without unlocking first, the door will stay shut when trying to open the door with a pulling force outward. This is especially useful for protecting lab equipment from accidental collisions.

Needle Holding Kit

2023 (17 years old)





A device that can be used to store and secure delicate microinjection needles. Designed to be a compact and easy-to-use system that supports housing needles with various diameters.

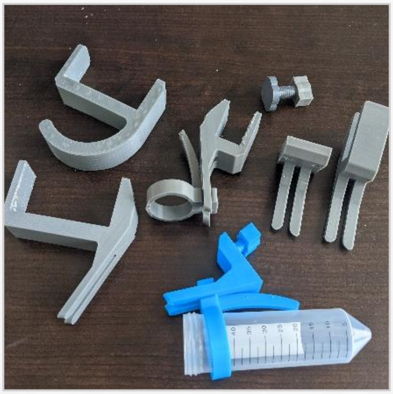
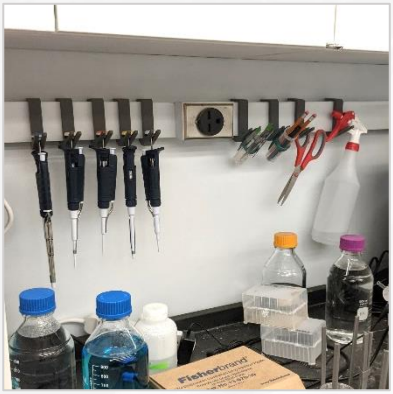
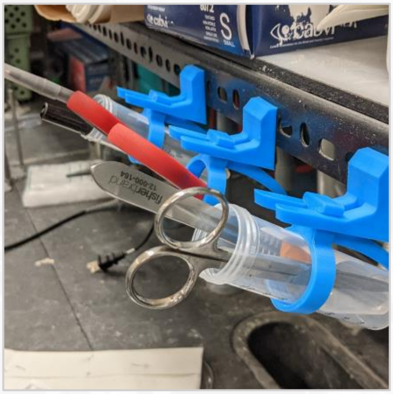
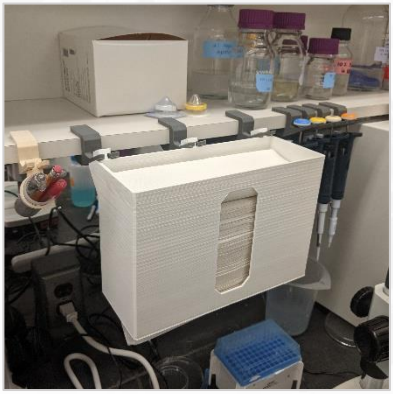
[Video Demo] [STL Files]

Targeted Problem: Delicate microinjection needles tend to be accidentally damaged in storage. Microinjection needles are extremely delicate items for injecting reagents into cells or embryos. The needles are commonly stored in home-made containers with clay which dries out and needs to be replaced with time.

Designed Solution: I designed a 3D printable needle holding device that is easier to use and lasts forever. For the needling holding slots, I structured them with thin layers to generate the required flexibility from rigid materials. The needles thus can be easily inserted, stored, and removed. Furthermore, the needle-holding assembly can be rotated upright, making it easy to access when in use.

Multi-purpose Holders

2022 (16 years old)

A modularized holder set that can be broadly used for multiple purposes in the lab. Designed to fit into various lab settings with many base variants and to be able to expand its functionality with add-on modules.

[Video Demo] [STL Files]

Targeted Problem: Lack of tools that maximize the use of lab space without compromising convenience and functionality. Space is one of the most precious resources in the lab, but the space layouts vary significantly.

Designed Solution: I designed the multi-purpose holders with variable bases to fit into different lab settings.

For the base, my design provides both fixed and adjustable bases, convenient to mount to all surfaces, including common shelves and tables.

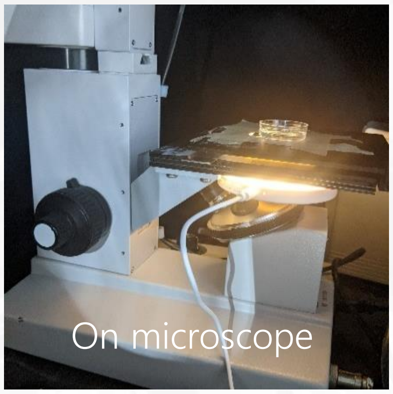
For the holder, my design allows them to directly hold large items such as pipettes, scissors, tools, clothes & bags, as well as arranging wires and cords. The holders can also couple with adaptors to further expand their functions, for instance, an O-ring to hold smaller items such as tweezers, or a common lab 50ml tube for placing pens and scalpels.

I love this project because of its usefulness and direct impact on research. Most of the commercially available counterparts are ridiculously expensive. One good example is the pipette holder, which usually charges >$150 for simply a piece of plastic and is incompatible between brands. All these are hurdles for research accessibility. With that in mind, I specifically chose to work with the pipette head knob, as it is the component shared by most of the brands. I then designed an additional adaptor to cover the other brands without the head knob.

This is an open-ended project and I continue to expand the sets. One of the later additions is a version that can be taped or screwed onto any flat surface. Another is a dispenser that can be used to hold and dispense paper tissue tower; designed to be positioned anywhere in the lab using my multi-purpose hangers.

Lens Ring Lamp

2023 (17 years old)

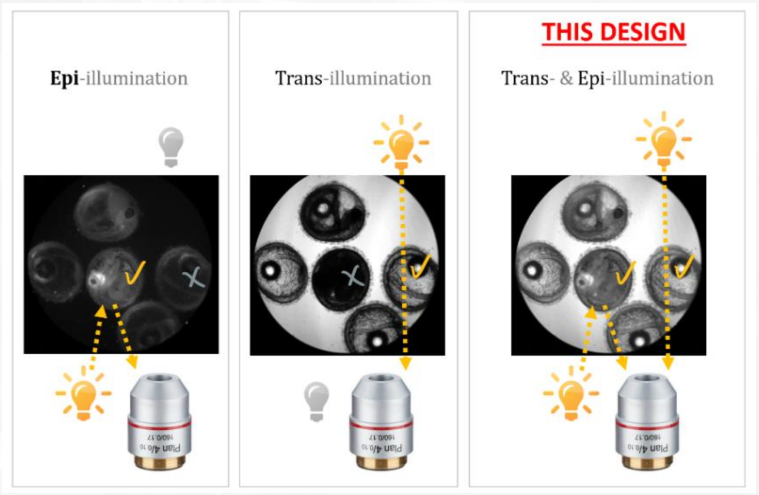


An adaptor set that can attach a commercial LED ring lamp directly onto microscope objectives. Designed to provide an affordable epi-illumination light source for inverted microscopes.

[Video Demo] [STL Files]

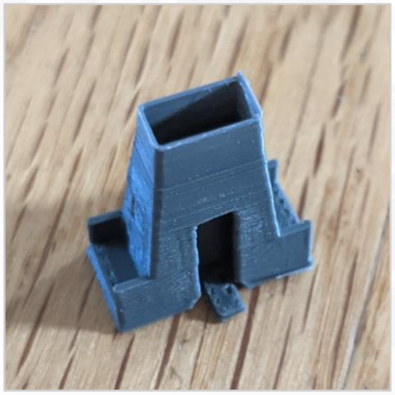
Targeted Problem: Cannot have light sources both above and below the samples. In microscopy, epi- and trans-illuminations are used for opaque and transparent specimen respectively (defined by whether the light source and the detector on the same or opposite side of specimen). Microscopes usually only equip one of the two. It is thus tricky to image both transparent and non-transparent specimens at the same time.

Designed Solution: I designed an adapter to attach commercial ring LEDs around the lens, to achieve the goal of having both trans- and epi-illumination at the same time. The adaptor was designed to be flexible. The grip on the lens is tight enough to be used on up-right microscopes where the lens is on top of the specimen.



Disposable Gel Extractor

2022 (16 years old)



A disposable kit that can be used to extract and preserve a targeted agarose gel piece. Designed to be extremely low-cost and very compact.

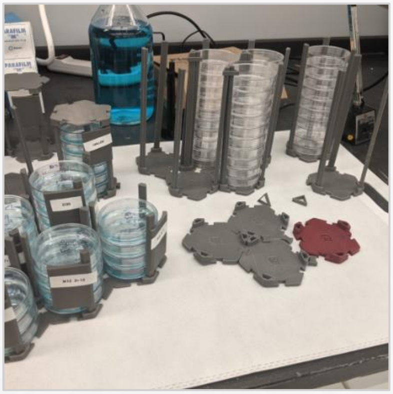
[Video Demo] [STL Files]

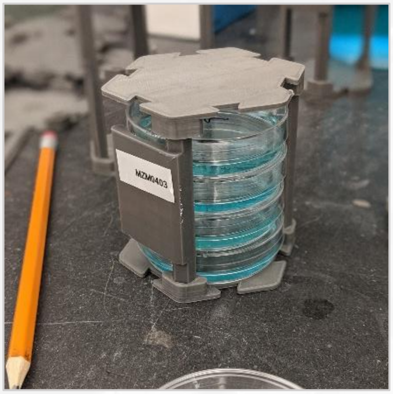
Targeted Problem: No convenient method for gel cutting. It is usually done by reused razor blades which carry risk of both sample contamination and sharp object injury.

Designed Solution: I designed a *single-print (print-in-place)* one-time-use square tube equipped with a slider to eject the cut agar. I made the slider and tube print in a single piece with some internal stoppers to prevent the slider from falling out. The final gel extractor is barely 1 gram and cost ~2 cents to produce.

Petri Dish Holder

2021 (15 years old)





A modularized organizer that can be used to efficiently manage Petri dishes in lab. Designed to be connected with hexagonal bases to minimize footprint.

[Video Demo] [STL Files]

Targeted Problem: Commercial petri dish organizers are bulky, not flexible with fixed uses, and overpriced. Petri dishes are probably the most used consumable in labs. They are sold in sleeves without a fitted container or holder.

Designed Solution: I designed a modularized petri dish organizer set that can be freely assembled and disassembled based on needs and for multiple purposes. To minimize the footprint, I designed a hexagonal base which can be tiled with dovetail connectors. Vertical poles can be interlocked to stabilize the assembly.

The petri dish holder can be easily converted into a petri-dish container by applying the hexagon base as a top lid with the labeling area included poles.

Scalable Universal Measuring Spoons

2022 (16 years old)

A scoop that can be used to directly deposit a fixed amount of agarose for gel making. Designed to save time from weighing agarose.

[Video Demo] [STL Files]

Targeted Problem: Repetitive weighing tasks for making the same agarose every time. People always made agarose gel using the same amount of agarose into the same amount of buffer solution. Even so, people still weigh the gel constitutes every time.

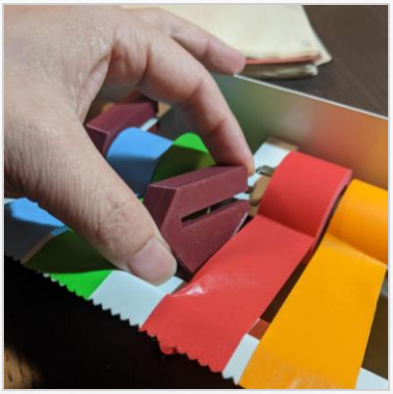
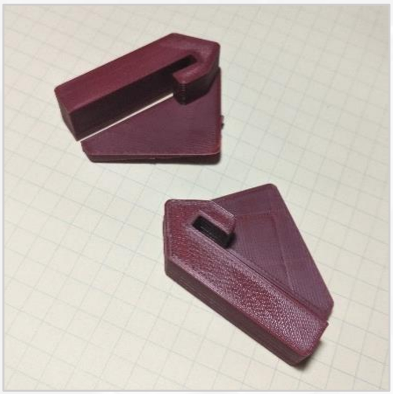
Designed Solution: I designed a small scoop to be kept in the agarose container and give a fixed amount of agarose powder. I made a ready-to-print 0.6 g scoop, easy for preparing commonly used 1.2% & 1.5% gels. I also released the original Inventor file, so people can adjust the scoop volume, going by the formula:

875w = 4/3r^3 + 5r^2 Units: w(g), r(mm)

The formula makes it scalable to any amount, and possibly extendable to other reagents and chemicals as well.

Tape Holder Holder

2022 (16 years old)

An attachment that can be used to secure rolls of tape inside a dispenser. Designed to address an infamous issue of a lab tape dispenser.

[Video Demo] [STL Files]

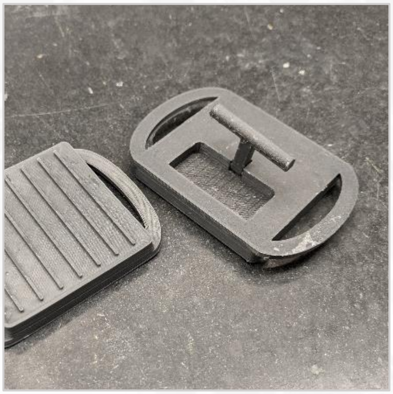
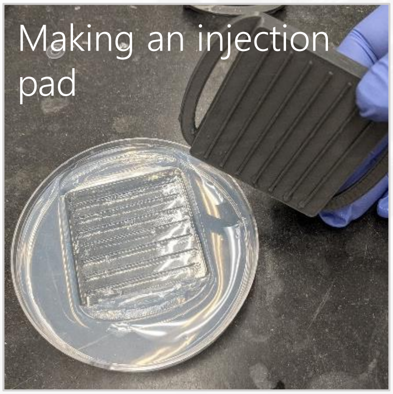
Targeted Problem: It was originated from a tweet in 2022: a scientist complaining his lab tape dispenser, a model universally used in all labs notoriously infamous for its bad design. The tape rolls consistently fall out from the dispenser when pulling the rolls straight up, a problem that everyone working in the lab has suffered with no answer.

Designed Solution: Within hours, I designed a printable clip over the frame to hold the spool down. The design is simple, with a special structural consideration for resisting pulling force.

The design was then quickly released and immediately helped many resolve this longtime pain (all my research designs are released through my father's Twitter account to reach a broader researcher network). This whole experience made me realize that even a very small simple design can have a broad and meaningful impact on a lot of people. It is also a good showcase of the power of 3D printing and how it can be helpful to everybody's life and work.

Microinjection Mold

2021 (15 years old)

An imprinting mold that can be used to imprint embryo-holding agarose plates for microinjection. Designed to operate with a print-in-place handle for convenience.

[Video Demo] [STL Files]

Targeted Problem: Microinjection imprinting molds are inconvenient to make by machine shop. They are used to make injection plates, which are required to hold embryos in position for microinjection, commonly made from molding an agarose plate.

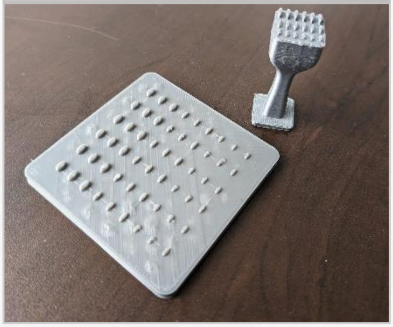
Designed Solution: I designed a mold which will not deform under the heat. I minimized the surface area contacting the agar and included three handles for convenient lifting. Higher temperature materials such as PETG can also be used to prevent warping.

This also include a print-in-place joint for a lift-out handle on the top of the mold to make operation easy.

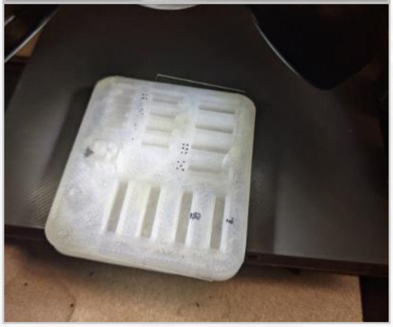
More General Lab Tools

2021-2023 (15-17 years old)

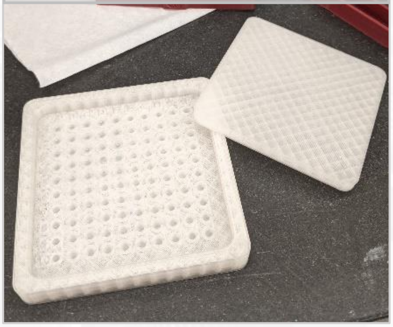
Brain Holding mold



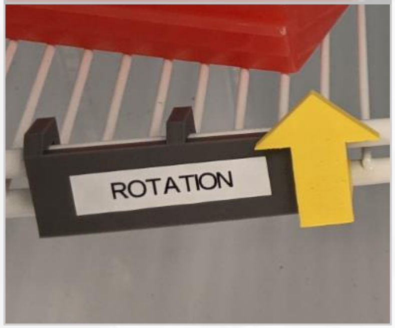
Fry Measuring Pad



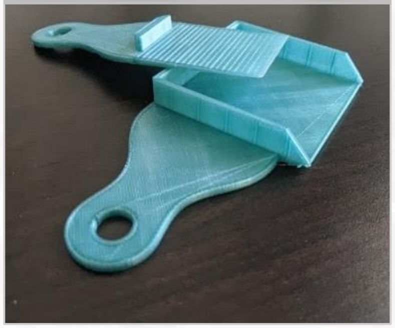
Embryo Shipping Kit



Universal Fridge Label Set



Gel Shovel Kit



Bottle Opener Set

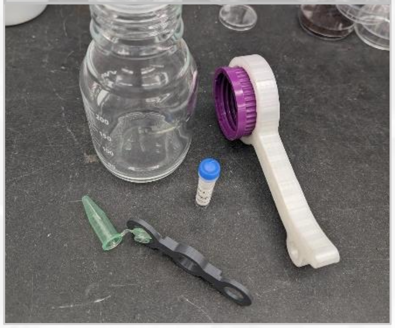
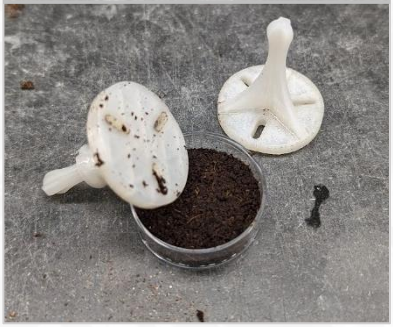
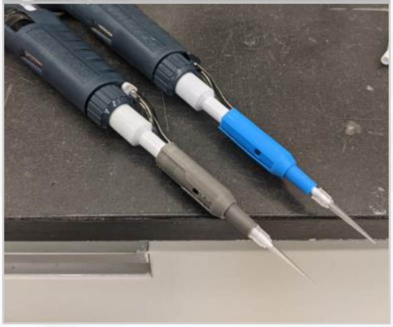


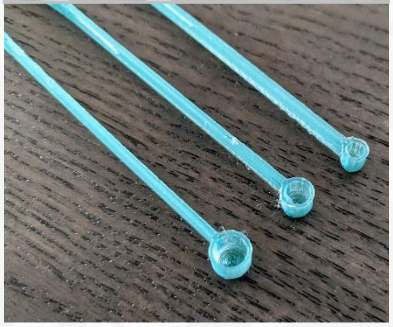
Plate Pusher Stamp



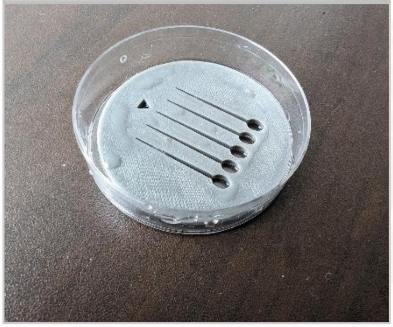
Pipette Tip Ejector



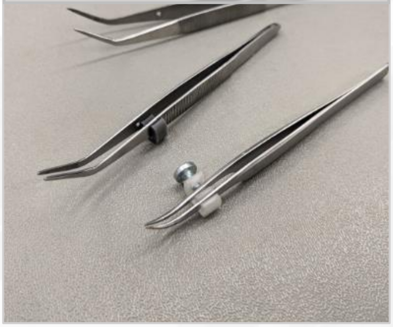
Micro-measuring Spoon



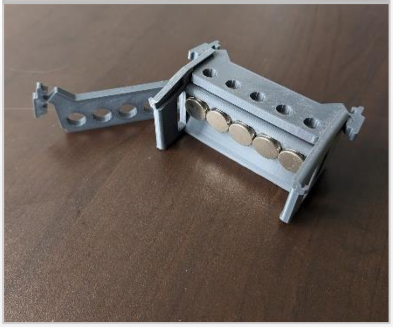
Embryo Imaging Pad



Tweezer Limiter



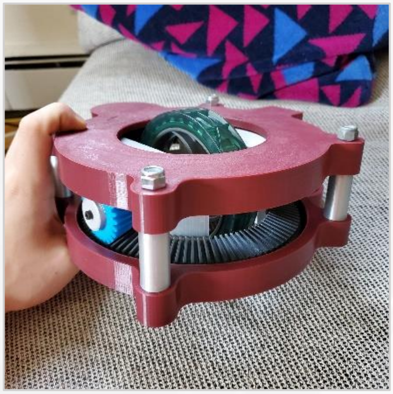
Switchable Magnetic Rack

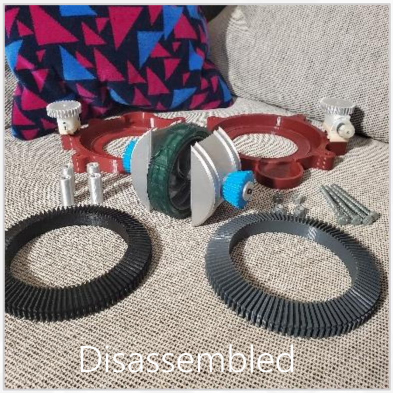


Engineering | School Projects

Engineering related projects in school, including for Principles of Engineering, Robotics, and designs for school activities. I aim to go above and beyond on every assignment.

Differential Swerve Drive 2022 (16 years old)





An experimental swerve drive, designed as an in-house drive system for robotics. A proposal for the First Robotic Competition (FRC) season.

[Video Demo] [Fusion Files]

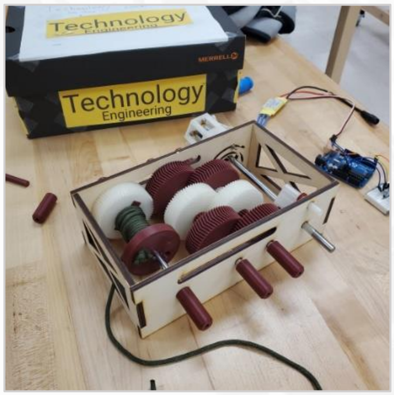
The idea was to develop a home-grown swerve drive for the Ward Melville robotics team. Swerve drive is a type of drive system in robotics where each wheel can be individually steered. It grants significant maneuvering advantages and thus is broadly used by most high-level teams. This prototype was produced as a proof of feasibility to the rest of the team.

After researching swerve, I chose differential swerve as the target drive system. A differential swerve would have greater power using both motors to steer and drive, and each module has a lower part count meaning it would be easier to manufacture.

The design is centered around a VEX robotics wheel, with two large 3D printed ring gears driving it. A 3D printed case holds the entire system together and acts as a mounting point for motors and electronics.

Once assembled, the swerve drive worked exactly as designed, demonstrating both forwards movement and steering.

Gearbox 2021 (15 years old)



A hoist designed to lift over 100 pounds using 3D printed plastics. The first project for my Principles for my Engineering class.

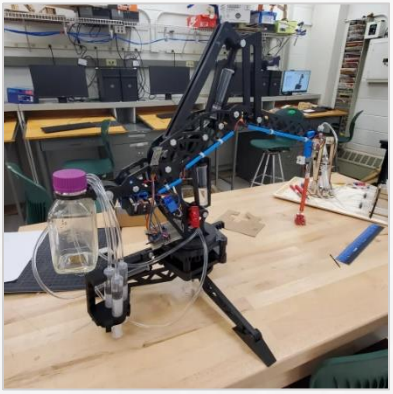
[STL Files]

The idea was to engineer a hoist with a high lift-to-weight ratio working in teams of two. After researching different types of gearboxes and materials. I chose 3D printing over other methods such as laser cutting due to its flexibility to work in all three dimensions.

In the final design, I integrated a chain of herringbone gears for its advantage in strength and ability to lock gears together, stacked on two shafts to conserve space. I utilized various materials, including steel shafts, laser-cut wooden sides, and 3d printed gears.

Hydraulic Arm

2021 (15 years old)



A hydraulic arm designed to provide maximal strength using generative design tools. The final project for my Principles of Engineering class.

[Video Demo] [STL Files] [Engineering Drawings]

The idea was to understand how a hydraulic system works and the engineering, physics, and math behind it. Each team of two in the class was given a limited budget to build their arm within. To pursue maximum functionality and cost-effectiveness, I created the entire design via 3D printing.

To ensure stability when lifting heavy objects, a wide legged base was used. Sitting on the base was the turntable, the most complex piece of engineering in the project. To have full 360 rotation, I designed a planetary gearbox hidden in the central stack that steps up the travel of a small 9 cm syringe, the hydraulic piston, to significantly more movement.

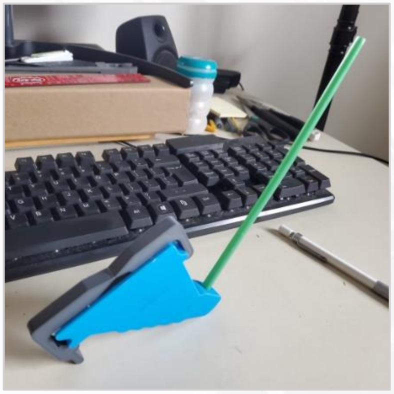
The arm structure is optimized for a balance between strength and movement, and was pressure-tested using simulations and scale prototypes. To increase strength and rigidity, I used Autodesk Inventor's built-in generative design feature. This process produced the unique geometry of each segment.

The project was built on the knowledge of my various other projects. I used planetary gears from the gearbox project, low friction guide rails from the microscope gliding table, and integrated usage of hardware.

After the end of the project, I took the arm home to display. Later, the design was repurposed for holding up a Lego Saturn V moon rocket. Minor upgrades have been made to improve strength and durability for this purpose.

Drinkinator

2022 (16 years old)



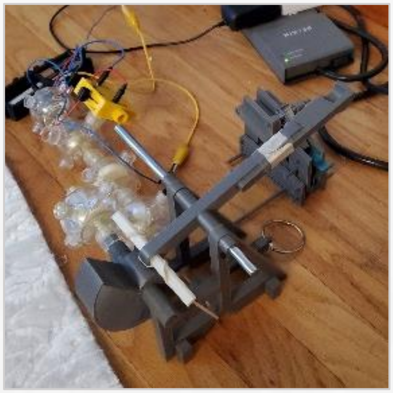
A device designed to hack the water fountains in school using the water bottle refill stations. A personal project for school life.

[STL Files]

The idea is one of the small hacks to improve my life at school. Even after the threat of Covid has passed, for a long while, my school turned off all the regular water fountains, but left all the water bottle refills on. This led to a huge inconvenience among students. I decided to solve the problem with engineering.

I designed a cup with a straw at the bottom in a flat shape to be portable with a special channel for excess water to drain out of. Furthermore, I integrated an ergonomic grip and a compliant lid.

Marble Machine – (I) Automatic Marble Trebuchet 2021 (15 years old)



A trebuchet designed to throw out marbles and reload automatically and repeatedly. The first part of the second project for my Principles of Engineering class.

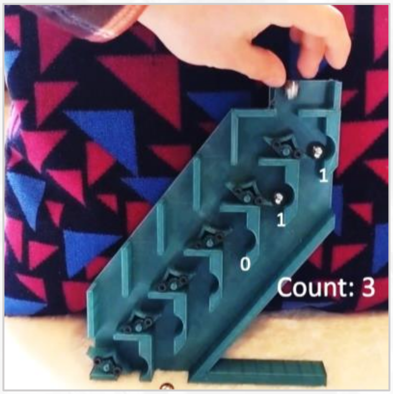
[Video Demo] [STL Files]

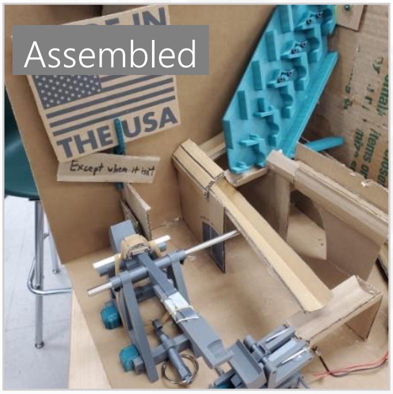
The idea was to create a marble machine utilizing recycled materials such as cardboard and toothpicks. This is the first part of the machine: a module for marble throwing.

The goal for this portion of the project was to lift marbles upwards, so I decided to build a trebuchet. The mechanism uses a large rubber band on a 3D printed frame to provide the force to throw the marble and a snail cam to enable repetitive motion. The device performed with precision, hitting almost the exact same spot many times in a row.

The most difficult part of the project was loading one marble at a time into the system. My solution was a seesaw which is pushed down by the trebuchet arm and lifts a marble out of the queue, but balanced in such a way that marbles would not be pushed forward by other marbles.

Marble Machine – (II) Binary Marble Calculator 2021 (15 years old)





A calculating device designed to perform and visualize the process of binary counting. The second part of the second project for my Principles of Engineering class.

[Video Demo] [STL Files]

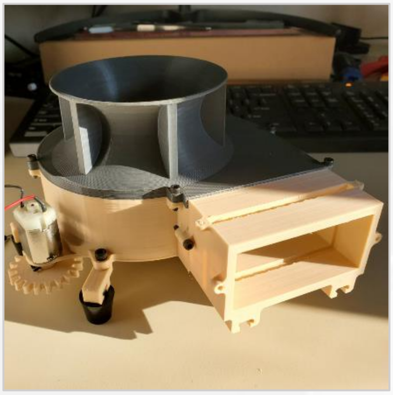
This is the second part of the machine: a module for marbles to go down through.

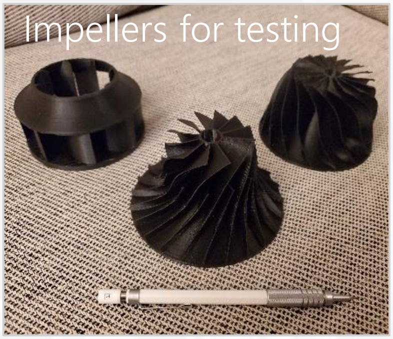
The goal for this portion of the project was to create a module that the marble would interact with as it heads down the machine. I got inspiration from mechanical calculators and decided to add a marble counter using a series of switches. The device is an easy visualization of a binary counting computer As the first marble enters it gets locked in the first switch, representing a 1. When a second marble arrives, it releases the first marble and locks in the second switch, representing 10, or 2 in binary. This process repeats to 63 before the calculator is automatically reset to 0.

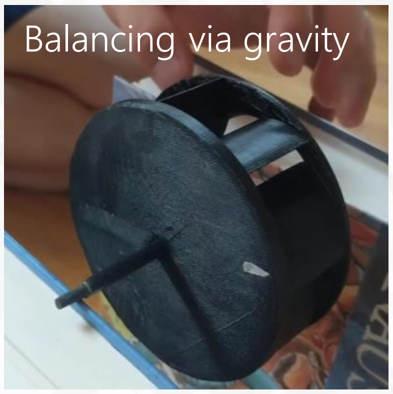
With the automatic trebuchet and the binary marble calculator put together, it forms one complete circuit as a marble machine. The trebuchet launches the marbles up the machine, and the calculator counts the marble as it rolls back down, ready to be launched again.

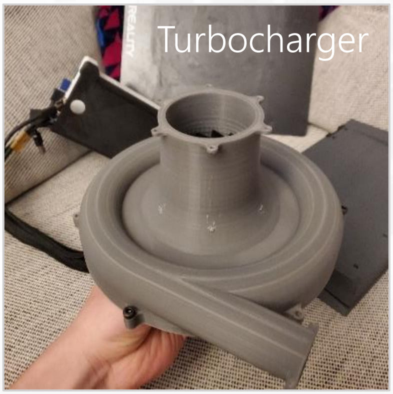
My projects with functions around the house. Most are born out of a genuine need, intriguing me to think outside of the box. These projects have been a major driver of my learning, continuously pushing me to explore new areas and try out new interests.

Air Purifier 2022 (16 years old)









A functional air purifier made with 3D printed parts. Designed to filter out dangerous fumes and particulates from the air using on-hand components.

[Video Demo] [STL Files]

As an active 3D printer user, I needed a filtering system to purify the air in the room post-printing, as many materials (such as ABS) are a hassle to print due to unhealthy fumes it produces. The goals for this project were to make a working device to keep my printing room safe, with mostly 3D printed or inexpensive parts, and at the same time, using this as an opportunity to further my understanding of the aerodynamics underneath.

A significant effort was focused on creating the fan that drives the air purifier. I went with a snail blower which would move sufficient air with significant pressure. I designed several different impellers and optimized the final design based on the testing results.

Before an impeller could be used, each had to be balanced to ensure smooth operation at high RPMs. Printed completely solid, I dynamically balanced each product afterward using gravity to locate heavy spots and a pair of scissors to trim material.

To power the fan, I used a small DC motor. It directly drove the fan for the prototypes and was changed to driving a gear chain in order to save space in the final version. The intake features a velocity stack as a part of the lid which ensures smoother airflow into the fan.

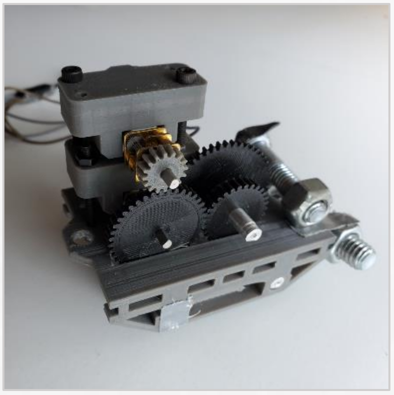
To see how far I could push the small DC motor, I designed a remix of the project based on a turbocharger. The impeller for the turbo is one of the most complex shapes I have ever 3D printed. all of it was produced in Autodesk Inventor using layered 3D sketches and lofting them together.

Printing was a real challenge, as the impeller has to be perfectly balanced, and accuracy was paramount. I was able to successfully complete the print by using my salvaged Monoprice Ultimate 3D printer which offers the greatest accuracy out of my printers. Balancing was also much more difficult, requiring hours of work to get it within working limits.

It sounded like a jet engine when I spun it up for the first time and produced a constant stream of high-pressure air. At this point, the motor was the limiting factor for the fan's performance, and I plan to revisit this project in the future with a more powerful device.

GeoTrax Train

2022 (16 years old)



A 3D printable toy train that is fully compatible with the GeoTrax train set. Designed to be able to tow around rail cars and advertise my robotics team during our competition.

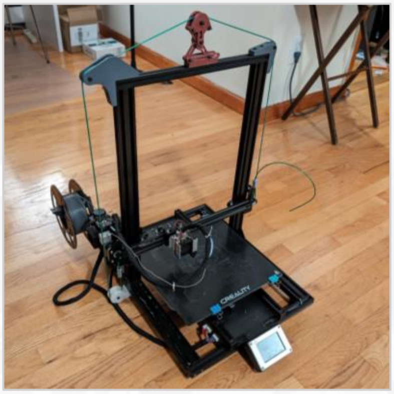
[Video Demo] [STL Files]

I started off with designing the wheels: a universal component on the project. I tested multiple different profiles, tweaking the angle and flange size until the wheels could handle curves without derailing.

the locomotive was significantly more complicated. It primarily used a cog to move, which engages with a matching rack on the track. Like my wheels, it took several iterations to get the cog to engage smoothly. All this was driven by a brushed motor, powered by 4 AA batteries in a 3D printed battery pack. To transmit torque from the motor to the wheels, I designed a 4-gear chain connected to the drive axle.

Printer Pulley System

2022 (16 years old)



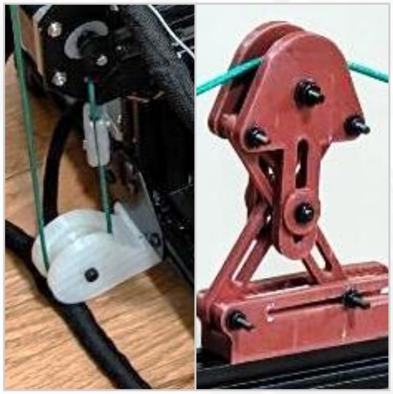
A self-invented pulley system. Designed as a home-made fix to my printer's sag using pulleys.

[STL Files]

After I noticed one side of my printer sagging, I needed to design something to make it level again. Usually, the solution would be to add another z-axis motor on the sagging side, but I did not have another stepper motor and the printer manufacturer did not design the printer to accommodate that. The solution I came up with was to make a pulley system that transfers the height from the stable side to the sagging side.

By tying a cable to the bottom of one side, I could run it to a pulley to the base of the printer, then up over the top, and attach it to the sagging side. I bought some slot nuts and designed some brackets that mounted on the top of the printer frame. Using a thin steel cable, I was able to attach and fix the sag.

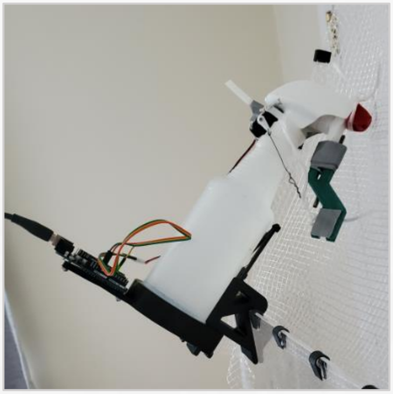
At last, because of the cable gradually unstiffening, it would become loose with time. To properly tension the cable, I then made a scissor mechanism mounted to the top of the printer that lifted the cable upwards at the top of the printer.



Pulley & Cable tensioner

Automatic Mister

2021 (15 years old)



A rapidly developed automatic gecko mister. Designed to take care of my geckos while I am away during the 2021 summer.

[STL Files] [Source Code]

With an impending one-week trip to London, I worried about my geckos not getting enough moisture, so I quickly make an automatic misting machine.

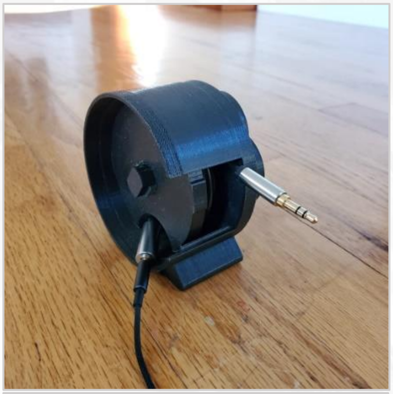
I used Arduino to activate a spray bottle pointed into my geckos' cage. The Arduino mounted on the body of the spray bottle would turn a servo every 30 minutes. The servo was attached to the spray bottle with zip ties and pulled the trigger with a steel wire. the entire assembly would hang on the top of the gecko cage aimed downwards.

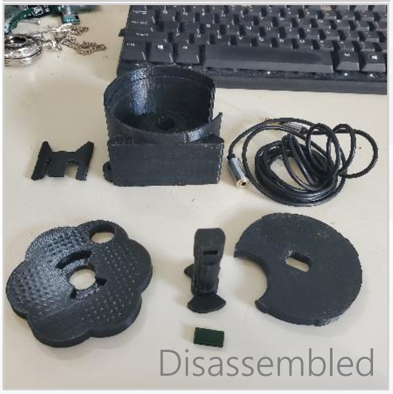
In the few days left, I did an overnight test to determine the water consumption rate. It turned out to be fairly efficient, only using a fraction of its maximum capacity. On the day of my departure, I filled the bottle completely and started the contraption. Every 3 hours, it would also reboot in case the Arduino ran into a problem.

When I returned from London, it was still going with around a quarter of the water left. My geckos were also alive and well, successfully achieving the goal.

Wire Coiler

2020 (14 years old)





A wire storing spool that is compact and simple to use. Designed as a significant step-up in my 3D printed designs.

[STL Files]

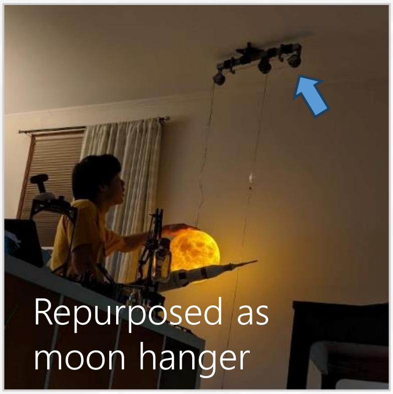
I wanted to create an easy way to roll and unroll cords.

The wire from my headphones kept getting tangled in my set-up, and when I had it neatly bundled, I would pull out the cord accidently. My design was to make a spool that the wire winds around with a rotating cap that automatically wound and unwound the wire. It would also have a knob at the bottom of the spool to make winding and unwinding faster.

When I stopped using wired headphones, it was easily transferred to other useful functions. Now, the device is used for organizing the cords of my electric violin and amp and saving space with long wires.

Curtain Slider

2020 (14 years old)



A unique 3D printable curtain started.

[STL Files]

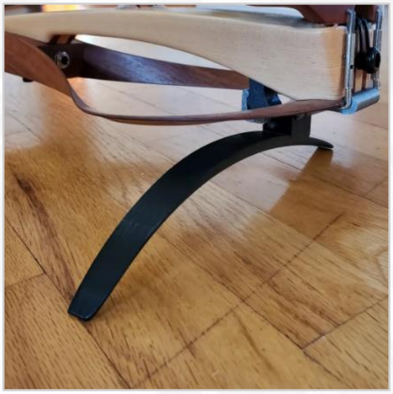
After moving into my new house in New York, I needed curtains that would fit the industrial style of my room.

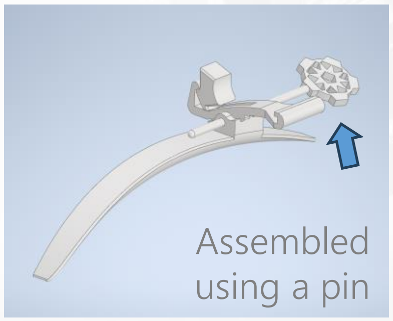
The idea was to make a sliding curtain that rolls along a long curtain rod. One obvious challenge was to 3D print such a long object. To solve this, I made many small segments that could join together using integrated connectors. To roll, I designed wheeled segments that suspended the curtain below.

Later, this project would discover an alternative use as well. I needed a way to suspend a 3D printed moon from the ceiling as part of my moon rocket diorama for the hydraulic arm. I repurposed the curtain rod and flipped it upside-down to act like a pulley instead. The pulleys let me raise and lower the height of the moon easily. Then, I printed a mount to stick it onto the ceiling and strengthening sleeves to keep them from breaking.

Shoulder Rest

2020 (14 years old)





A custom shoulder rest for my YEV-105 electric violin designed. Designed to fit into my shoulder shape perfectly.

[STL Files]

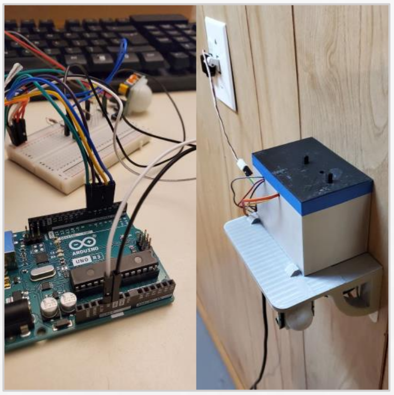
After a crack developed in one of the wooden sides of my electric violin, a regular shoulder rest would no longer work, and I needed a new one.

There was no commercial product that fit my unique needs, so I designed everything from scratch. It mounts on the central body of the violin, not putting any stress on the cracked wooden sides. The mounting point is a two-part 3D printed clip which snaps in place and provides a stable mounting point. A pin is then used to connect it to a large curved part which is designed to fit my shoulder shape.

As a result of the non-geometric shape of the violin and human body, this is one of my most difficult projects in terms of the geometry involved. I learned how to use splines and curves in Inventor and how to design interfaces between them.

Motion Sensor Light Switch

2022 (16 years old)



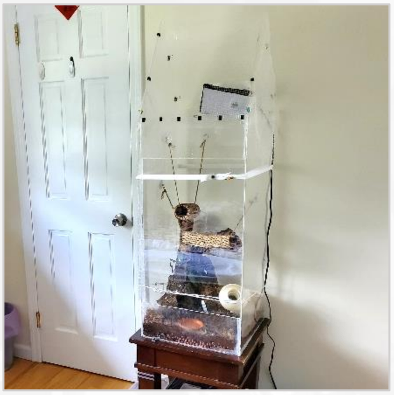
A motion sensing light switch using an Arduino. Designed to make the basement much more convenient to access.

[Video Demo] [STL Files] [Source Code]

I wired up the design using an Arduino as the core. I pulled from my programming experience to make the switch as smart as possible, including automatically turning off after a few minutes. I further pushed my skills by adding two buttons, one as an on/off switch and the other to deactivate the motion detector. After wiring the electronics, I designed a 3D printable shelf and box to mount everything in.

Gecko Cage

2021 (15 years old)



A homemade acrylic gecko cage. Designed to be larger and cheaper than commercial alternatives.

[Video Demo]

After moving to New York, I needed to figure out how to house my geckos.

As the old cage from Palo Alto was a bit small, I was intrigued to build my gecko a spacious cage. I decided on a design by drawing it out in Inventor. Once complete, I cut acrylic panels for the sides and assembled them with aquarium sealant.

After waiting a few days for the sealant to cure, I moved the cage into my room and filled it up with dirt and the terrarium decor. After a year of service, I cleaned it out and repaired some damaged sealant, altogether it has held together remarkably well.

Lizard Tube System

2018 (12 years old)

A collection of designs for my reptiles, including the first of my 3D designs starting in 7th grade (12 years old).

[STL Files]

My very first design and also the reason I got into 3D printing in the first place: I wanted to created a collection of reptile-related resources for my pet geckos.

My very first project was a set of tubes that my geckos could crawl through; this set was continuously expanded over the years, and it is still in use in my modern gecko habitat.

This branched out into other projects for my lizards like a two-compartment lizard carrier that interfaced with the tube system and many themed decorations.

Computer Science

```
);
Image.Width;
Image.Width * sourceRa
Image.Height * sourceRa

Image, new Size(
* sx, gecko
sy);
```

I first learned how to code using Unity in 9th grade, making video games during the pandemic. I have been an active member of the United Computations Club at Gunn, and an organizer for their annual GunnHacks hackathon since 10th grade. I also have been participating in the Computer Science Club at Ward Melville.

Currently, I am familiar with, in order of competence: C#, Unity, C, HTML CSS JS, Python, and a bit of Kotlin.

A Snapshot of the Mosaic Maker Code (2020)

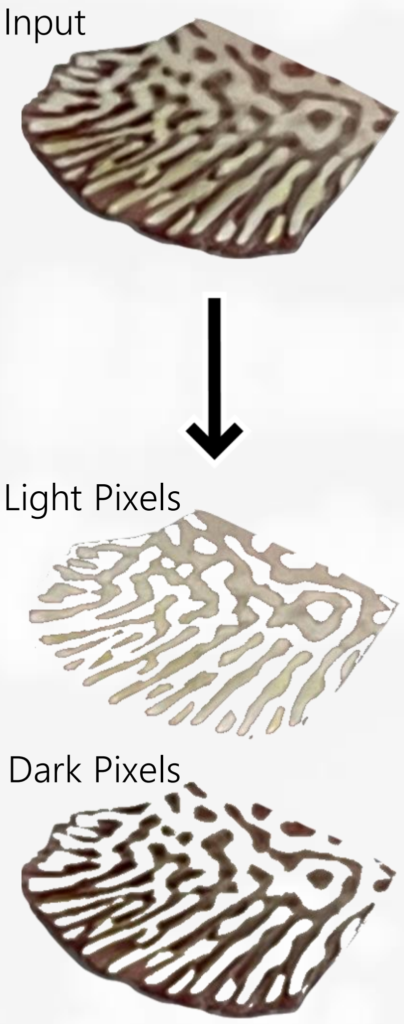
Pattern Quantifier [1/2] — 2022 (16 years old)

A pixel quantification tool to help elucidate the principle of color pattern formation in animals.

[Video Demo] [Source Code] [Live Website]

This software analyzes and quantifies the pixels of the target object in images. For each image, the software automatically calculates and determines the cutoff between light and dark pixels, and separates the original image into two subset images composed of only light or dark pixels respectively. It will then return the percentage of light and dark pixels of the target objects in the original image.

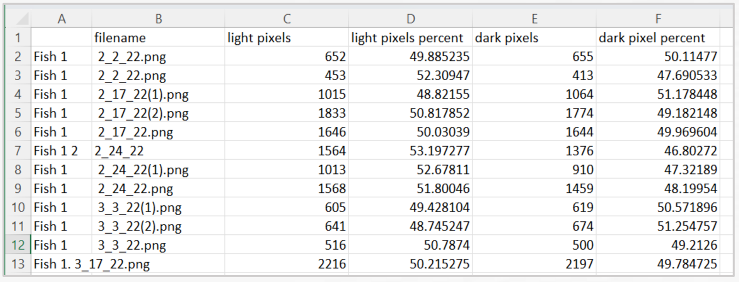
This tool is a key component of a remote outreach program specifically designed for high school students who cannot afford the high expense of on-site research programs. With this tool, students can work remotely to analyze the image dataset, generate their own questions, and try to find answers. My father is into outreach and developed this program in his new lab. He needed someone to develop the software tool, leading myself to help out. Using my experience in image processing and analysis, I created a console application in C# which could process an entire folder of images and output results to a spreadsheet.



The developmental process and the trial run were a learning experience for me, as I had first-hand exposure to the inequality of educational resources. We had students from New York and Texas who are FGLI (First Generation, Low Income). The C# based application is meant to run on PC, for its popularity among the target audience. However, we soon realized that for some students, the only computer they have are Chromebooks borrowed from their high school. To solve this problem, I rewrote the application in JavaScript and hosted it on GitHub. It produced nearly identical results with the console app and anyone could easily run it with internet access, maximizing accessibility.

In terms of the research subject, the African killifish has very beautiful color patterns on their fins. Like our fingerprints, these patterns are similar but unique between individuals. Also like us, the colors start to fade with age.

The outreach program provides a dataset of weekly images from 8 killifish across their whole lives. Encouraged to form their own hypotheses on how the color patterns develop and change, the students crop out the interested area of the fin and use my software tool to generate data and validate their hypotheses.

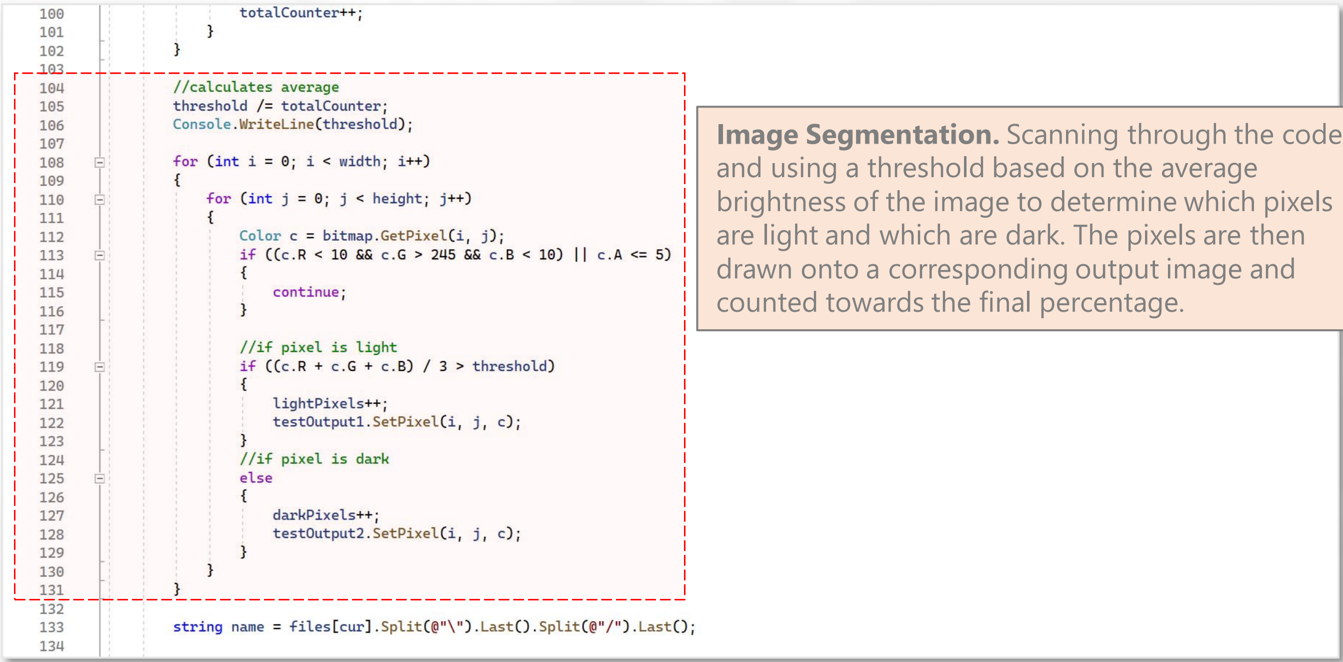


	A	B	C	D	E	F
1		filename	light pixels	light pixels percent	dark pixels	dark pixel percent
2	Fish 1	2_2_22.png	652	49.885235	655	50.11477
3	Fish 1	2_2_22.png	453	52.30947	413	47.690533
4	Fish 1	2_17_22(1).png	1015	48.82155	1064	51.178448
5	Fish 1	2_17_22(2).png	1833	50.817852	1774	49.182148
6	Fish 1	2_17_22.png	1646	50.03039	1644	49.969604
7	Fish 1 2	2_24_22	1564	53.197277	1376	46.80272
8	Fish 1	2_24_22(1).png	1013	52.67811	910	47.32189
9	Fish 1	2_24_22.png	1568	51.80046	1459	48.19954
10	Fish 1	3_3_22(1).png	605	49.428104	619	50.571896
11	Fish 1	3_3_22(2).png	641	48.745247	674	51.254757
12	Fish 1	3_3_22.png	516	50.7874	500	49.2126
13	Fish 1. 3_17_22.png		2216	50.215275	2197	49.784725

Measurement output

Pattern Quantifier [2/2]

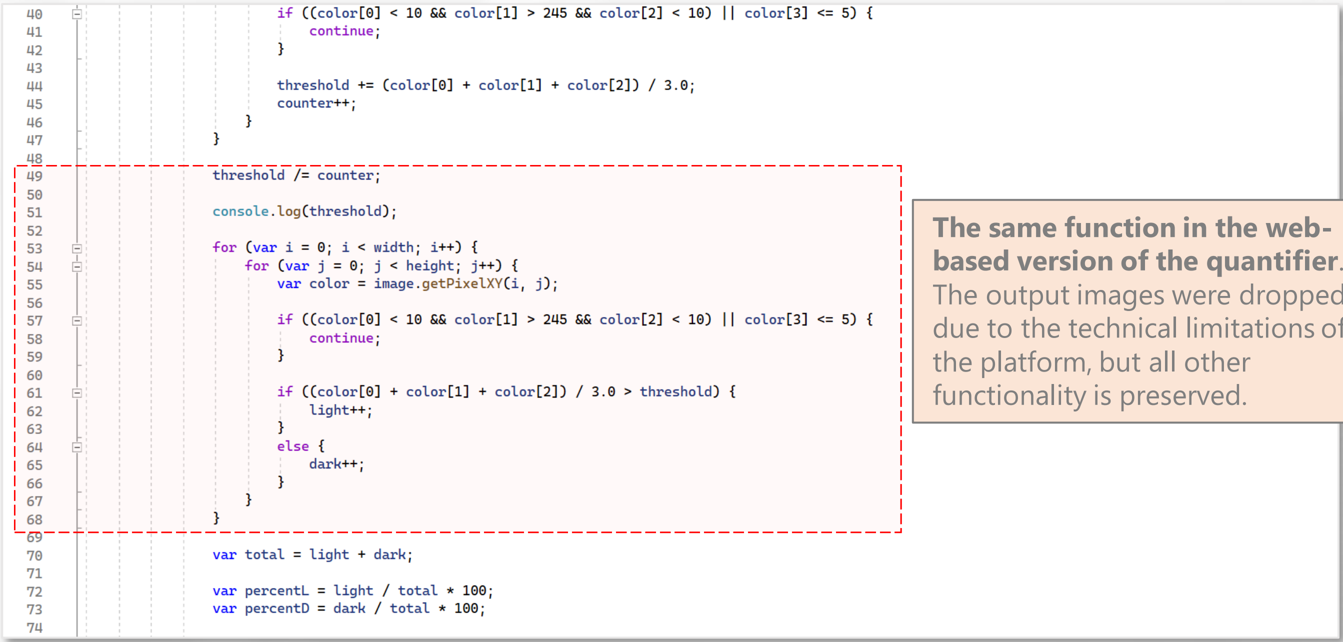
Pattern-quantifier/Program.cs | Line 104-131 (with C#)



```
100             totalCounter++;
101         }
102     }
103
104     //calculates average
105     threshold /= totalCounter;
106     Console.WriteLine(threshold);
107
108     for (int i = 0; i < width; i++)
109     {
110         for (int j = 0; j < height; j++)
111         {
112             Color c = bitmap.GetPixel(i, j);
113             if ((c.R < 10 && c.G > 245 && c.B < 10) || c.A <= 5)
114             {
115                 continue;
116             }
117
118             //if pixel is light
119             if ((c.R + c.G + c.B) / 3 > threshold)
120             {
121                 lightPixels++;
122                 testOutput1.SetPixel(i, j, c);
123             }
124             //if pixel is dark
125             else
126             {
127                 darkPixels++;
128                 testOutput2.SetPixel(i, j, c);
129             }
130         }
131     }
132
133     string name = files[cur].Split(@"\").Last().Split(@"/").Last();
134
```

Image Segmentation. Scanning through the code and using a threshold based on the average brightness of the image to determine which pixels are light and which are dark. The pixels are then drawn onto a corresponding output image and counted towards the final percentage.

Pattern-quantifier/web/processor.js | Line 49-68 (with JS)

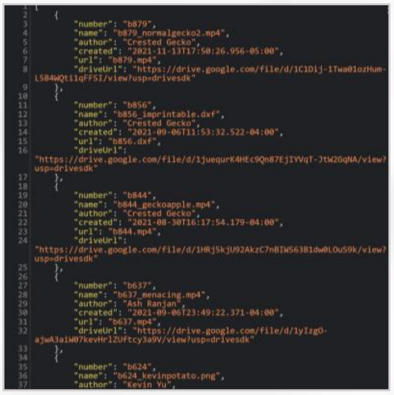


```
40             if ((color[0] < 10 && color[1] > 245 && color[2] < 10) || color[3] <= 5) {
41                 continue;
42             }
43
44             threshold += (color[0] + color[1] + color[2]) / 3.0;
45             counter++;
46         }
47     }
48
49     threshold /= counter;
50
51     console.log(threshold);
52
53     for (var i = 0; i < width; i++) {
54         for (var j = 0; j < height; j++) {
55             var color = image.getPixelXY(i, j);
56
57             if ((color[0] < 10 && color[1] > 245 && color[2] < 10) || color[3] <= 5) {
58                 continue;
59             }
60
61             if ((color[0] + color[1] + color[2]) / 3.0 > threshold) {
62                 light++;
63             }
64             else {
65                 dark++;
66             }
67         }
68     }
69
70     var total = light + dark;
71
72     var percentL = light / total * 100;
73     var percentD = dark / total * 100;
74
```

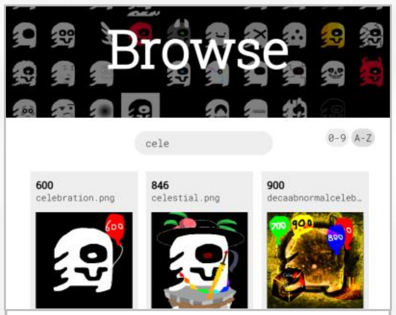
The same function in the web-based version of the quantifier. The output images were dropped due to the technical limitations of the platform, but all other functionality is preserved.

Geckoimages API [1/2]

2020 (14 years old)



```
[
    {
        "number": "b879",
        "name": "b879_normalgecko2.mp4",
        "author": "Crested Gecko",
        "created": "2021-11-13T17:50:26.956-05:00",
        "url": "b879.mp4",
        "driveUrl": "https://drive.google.com/file/d/1C1Dij-1Twa01ozHum-
L5B4WQti1qFFSI/view?usp=drivesdk"
    },
    {
        "number": "b856",
        "name": "b856_imprintable.dxf",
        "author": "Crested Gecko",
        "created": "2021-09-06T11:53:32.522-04:00",
        "url": "b856.dxf",
        "driveUrl":
"https://drive.google.com/file/d/1juequrK4HEc9Qn87EjIYVqT-JtW2GqNA/view?
usp=drivesdk"
    },
    {
        "number": "b844",
        "name": "b844_geckoapple.mp4",
        "author": "Crested Gecko",
        "created": "2021-08-30T16:17:54.179-04:00",
        "url": "b844.mp4",
        "driveUrl":
"https://drive.google.com/file/d/1HRj5kjU92AkzC7nBIWS63B1dw0LOuS9k/view?
usp=drivesdk"
    },
    {
        "number": "b637",
        "name": "b637_menacing.mp4",
        "author": "Ash Ranjan",
        "created": "2021-09-06T23:49:22.371-04:00",
        "url": "b637.mp4",
        "driveUrl": "https://drive.google.com/file/d/1yIzgO-
ajwA3aiW07kevHrlZUftcy3a9V/view?usp=drivesdk"
    },
    {
        "number": "b624",
        "name": "b624_kevinpotato.png",
        "author": "Kevin Yu",
```



This API provides images for the Gecko Collection website

A fully functional programming interface for an art project "Geckoimages".

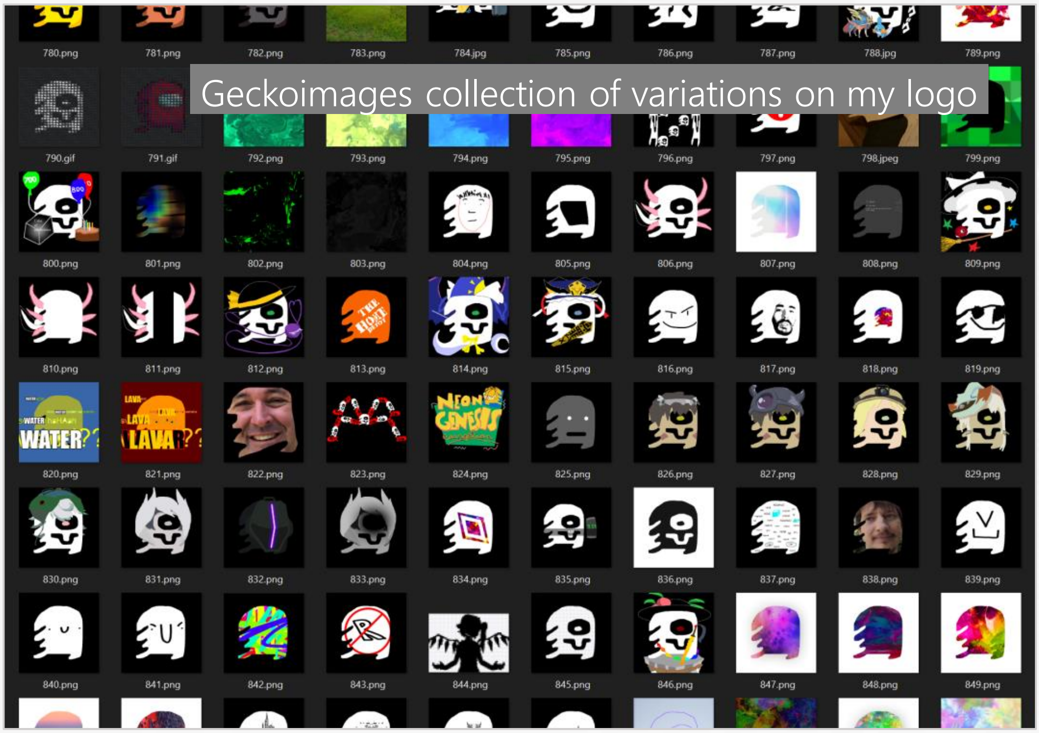
[Video Demo] [Source Code] [Live website]

The Geckoimages is an art project of variations on a single base image, and as it grew, I wanted to make a simple and intuitive way to use them in any future project.

The collection is primarily hosted through Google Drive, which is complicated and inconvenient to use for projects. My solution was to create an Application Programming Interface or API, which allows outside programs to easily get images and information.

I coded the API in C# using Microsoft's ASP.NET framework. To this, I combined Google Drive's own API to download the source images and to automatically store a local backup of them. When requested, the API returns the image along with all relevant information including the author and date created. After much fiddling, I upgraded the API onto a secure connection (HTTPS), which allows anyone with internet access to request from the API.

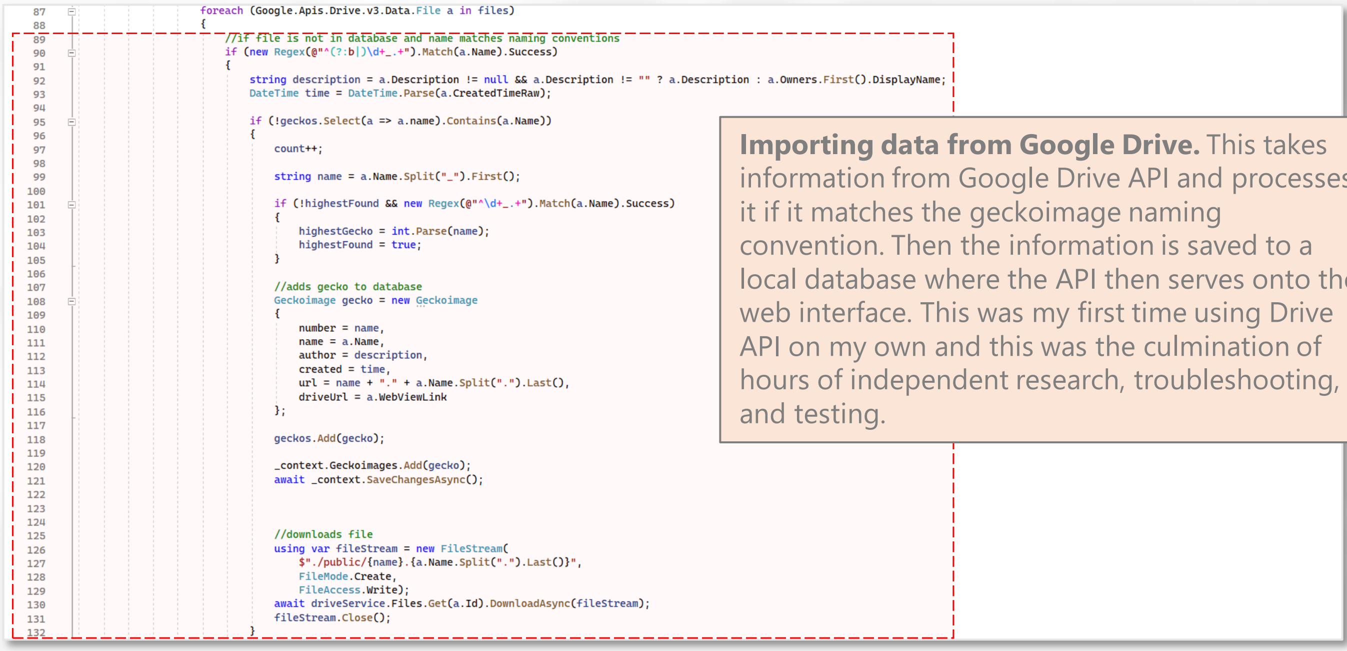
I was immediately able to use this new tool to provide all the images for the Geckoimages Website, thus saving on server costs.



Geckoimages collection of variations on my logo

Geckoimages API [2/2]

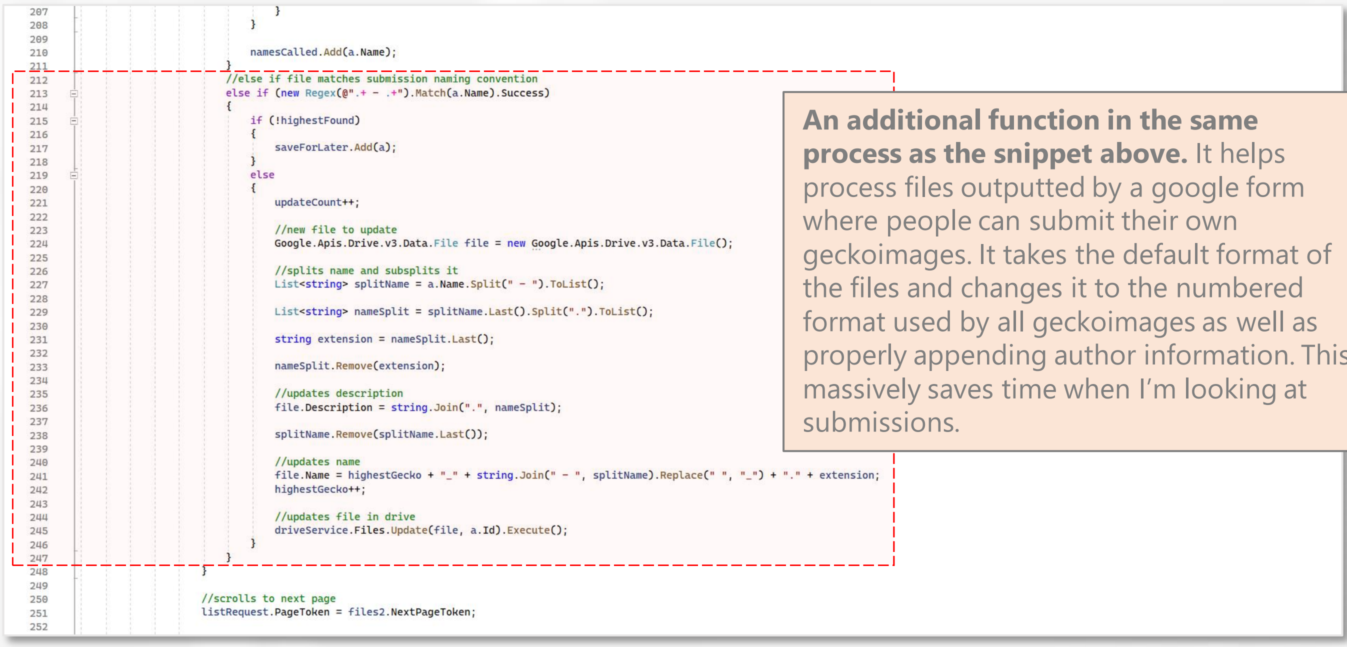
GeckoimagesApi/Program.cs | Line 89-132 (with C#)



```
 87  foreach (Google.Apis.Drive.v3.Data.File a in files)
 88  {
 89      //if file is not in database and name matches naming conventions
 90      if (new Regex(@"^(?:b|)\d+_.+").Match(a.Name).Success)
 91      {
 92          string description = a.Description != null && a.Description != "" ? a.Description : a.Owners.First().DisplayName;
 93          DateTime time = DateTime.Parse(a.CreatedTimeRaw);
 94
 95          if (!geckos.Select(a => a.name).Contains(a.Name))
 96          {
 97              count++;
 98
 99              string name = a.Name.Split("_").First();
100
101              if (!highestFound && new Regex(@"^\d+_.+").Match(a.Name).Success)
102              {
103                  highestGecko = int.Parse(name);
104                  highestFound = true;
105              }
106
107              //adds gecko to database
108              Geckoimage gecko = new Geckoimage
109              {
110                  number = name,
111                  name = a.Name,
112                  author = description,
113                  created = time,
114                  url = name + "." + a.Name.Split(".").Last(),
115                  driveUrl = a.WebViewLink
116              };
117
118              geckos.Add(gecko);
119
120              _context.Geckoimages.Add(gecko);
121              await _context.SaveChangesAsync();
122
123
124
125              //downloads file
126              using var fileStream = new FileStream(
127                  $"./public/{name}.{a.Name.Split(".").Last()}",
128                  FileMode.Create,
129                  FileAccess.Write);
130              await driveService.Files.Get(a.Id).DownloadAsync(fileStream);
131              fileStream.Close();
132          }
```

Importing data from Google Drive. This takes information from Google Drive API and processes it if it matches the geckoimage naming convention. Then the information is saved to a local database where the API then serves onto the web interface. This was my first time using Drive API on my own and this was the culmination of hours of independent research, troubleshooting, and testing.

GeckoimagesApi/Program.cs | Line 212-247 (with C#)

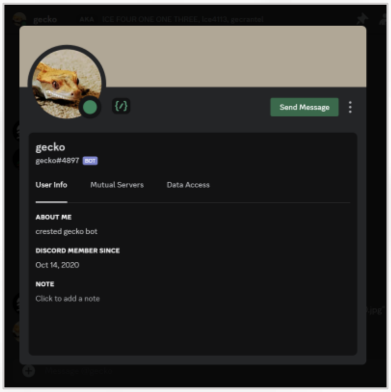


```
207                  }
208              }
209
210              namesCalled.Add(a.Name);
211          }
212          //else if file matches submission naming convention
213          else if (new Regex(@".+ - .+").Match(a.Name).Success)
214          {
215              if (!highestFound)
216              {
217                  saveForLater.Add(a);
218              }
219              else
220              {
221                  updateCount++;
222
223                  //new file to update
224                  Google.Apis.Drive.v3.Data.File file = new Google.Apis.Drive.v3.Data.File();
225
226                  //splits name and subsplits it
227                  List<string> splitName = a.Name.Split(" - ").ToList();
228
229                  List<string> nameSplit = splitName.Last().Split(".").ToList();
230
231                  string extension = nameSplit.Last();
232
233                  nameSplit.Remove(extension);
234
235                  //updates description
236                  file.Description = string.Join(".", nameSplit);
237
238                  splitName.Remove(splitName.Last());
239
240                  //updates name
241                  file.Name = highestGecko + "_" + string.Join(" - ", splitName).Replace(" ", "_") + "." + extension;
242                  highestGecko++;
243
244                  //updates file in drive
245                  driveService.Files.Update(file, a.Id).Execute();
246              }
247          }
248      }
249
250      //scrolls to next page
251      listRequest.PageToken = files2.NextPageToken;
252
```

An additional function in the same process as the snippet above. It helps process files outputted by a google form where people can submit their own geckoimages. It takes the default format of the files and changes it to the numbered format used by all geckoimages as well as properly appending author information. This massively saves time when I'm looking at submissions.

Geckobot [1/2]

2020 (14 years old)



gecko
gecko#4897 BOT

User Info Mutual Servers Data Access

ABOUT ME
crested gecko bot

DISCORD MEMBER SINCE
Oct 14, 2020

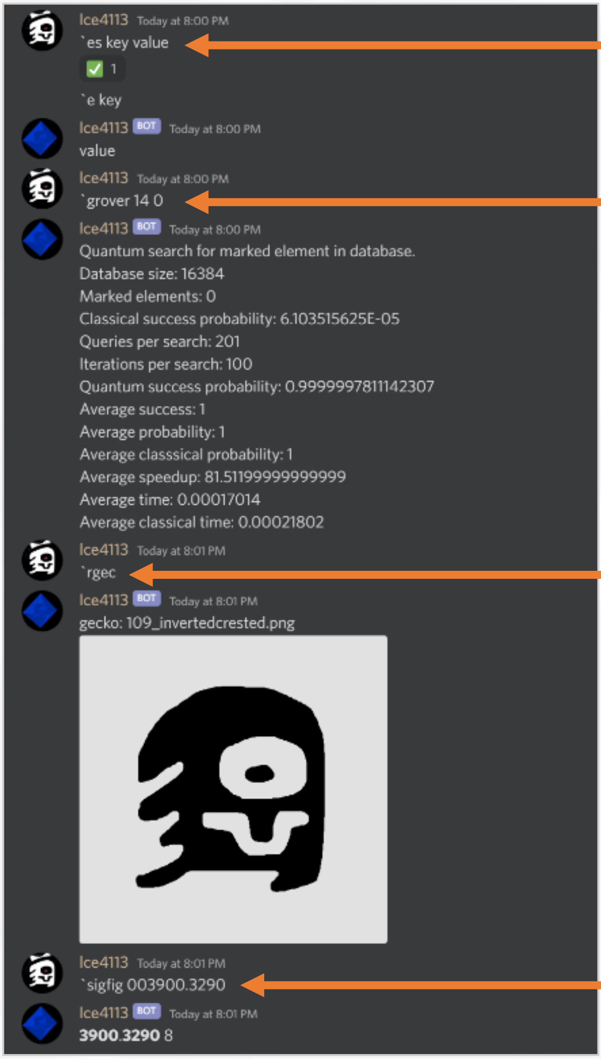
NOTE
Click to add a note

A Discord bot that was one of my first coding projects, but slowly built up to over 6000 lines of code.

[Video Demo] [Source Code]

I created Geckobot in 9th grade as an activity for the United Computations Club at Gunn High School. This was my first real experience coding outside of Unity and I used it as a tool to help me understand C# and computer science better.

The bot itself is modularized into several different sets of functions. It started from simple modules like math, branching out to significant figures, to storing user specified data, to trying out Google Drive API, to text-to-speech with DecTalk, and even dabbling in quantum computing using q#. The bot currently has *25* of these modules added throughout the years. This structure allowed me to be able to branch out to explore new ideas and make intangible concepts tangible through interacting with the bot on Discord.



```
Ice4113 Today at 8:00 PM
`es key value
✅ 1
`e key
Ice4113 BOT Today at 8:00 PM
value
Ice4113 Today at 8:00 PM
`grover 14 0
Ice4113 BOT Today at 8:00 PM
Quantum search for marked element in database.
Database size: 16384
Marked elements: 0
Classical success probability: 6.103515625E-05
Queries per search: 201
Iterations per search: 100
Quantum success probability: 0.9999997811142307
Average success: 1
Average probability: 1
Average classsical probability: 1
Average speedup: 81.51199999999999
Average time: 0.00017014
Average classical time: 0.00021802
Ice4113 Today at 8:01 PM
`rgec
Ice4113 BOT Today at 8:01 PM
gecko: 109_invertedcrested.png
Ice4113 Today at 8:01 PM
`sigfig 003900.3290
Ice4113 BOT Today at 8:01 PM
3900.3290 8
```

Key/Value Storing: one of the most popular functions of the bot for storing custom emotes

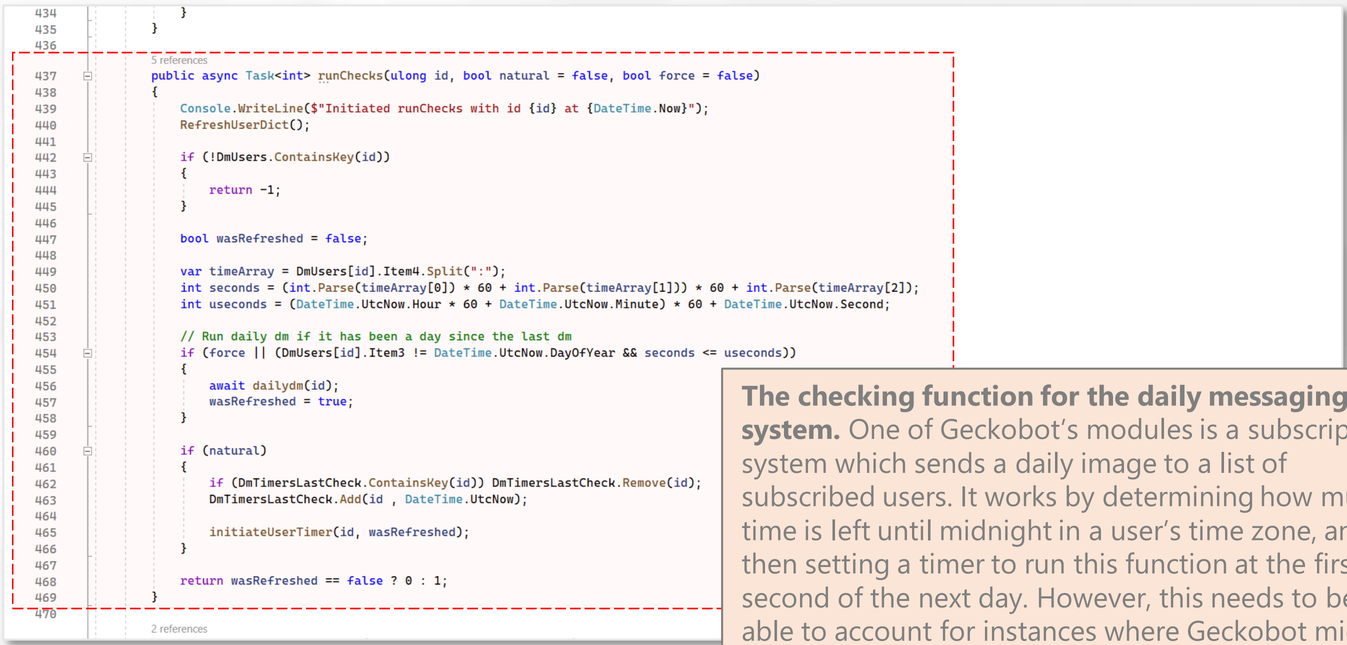
Quantum Grover Search: an experiment demonstrating a theoretical search algorithm faster than any classical approach.

Geckoimage integration: a set of functions which lets users search through the entire collection and sends daily images to subscribers

Significant Figures Calculator: a set of tools which counts, adds, subtracts, multiplies, and divides according to standard sig fig rules; first developed for 9th grade Bio H.

Geckobot [2/2]

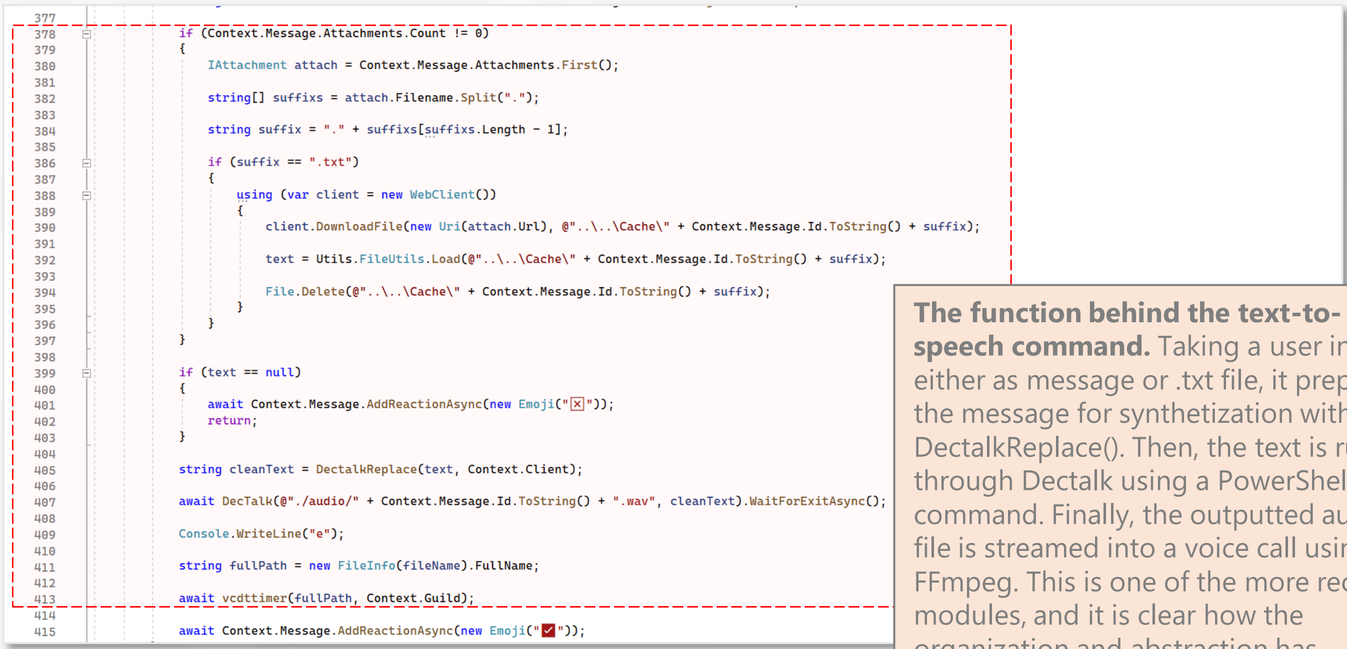
Geckobot/Commands/DailyDM.cs | Line 437-469 (with C#)



```
434            }
435        }
436
           5 references
437        public async Task<int> runChecks(ulong id, bool natural = false, bool force = false)
438        {
439            Console.WriteLine($"Initiated runChecks with id {id} at {DateTime.Now}");
440            RefreshUserDict();
441
442            if (!DmUsers.ContainsKey(id))
443            {
444                return -1;
445            }
446
447            bool wasRefreshed = false;
448
449            var timeArray = DmUsers[id].Item4.Split(":");
450            int seconds = (int.Parse(timeArray[0]) * 60 + int.Parse(timeArray[1])) * 60 + int.Parse(timeArray[2]);
451            int useconds = (DateTime.UtcNow.Hour * 60 + DateTime.UtcNow.Minute) * 60 + DateTime.UtcNow.Second;
452
453            // Run daily dm if it has been a day since the last dm
454            if (force || (DmUsers[id].Item3 != DateTime.UtcNow.DayOfYear && seconds <= useconds))
455            {
456                await dailydm(id);
457                wasRefreshed = true;
458            }
459
460            if (natural)
461            {
462                if (DmTimersLastCheck.ContainsKey(id)) DmTimersLastCheck.Remove(id);
463                DmTimersLastCheck.Add(id , DateTime.UtcNow);
464
465                initiateUserTimer(id, wasRefreshed);
466            }
467
468            return wasRefreshed == false ? 0 : 1;
469        }
470
           2 references
```

The checking function for the daily messaging system. One of Geckobot's modules is a subscription system which sends a daily image to a list of subscribed users. It works by determining how much time is left until midnight in a user's time zone, and then setting a timer to run this function at the first second of the next day. However, this needs to be able to account for instances where Geckobot might go down or my computer it is hosted on might sleep. My solution was to regenerate the timers at a regular time interval which is done by calling this function.

Geckobot/Commands/VoiceCall.cs | Line 378-413 (with C#)

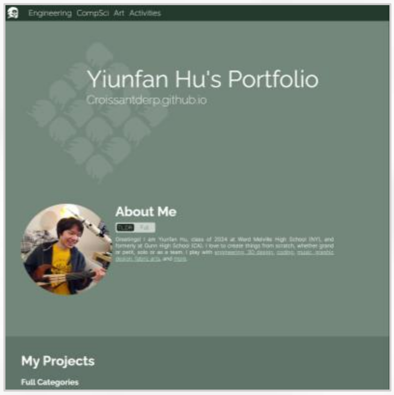


```
377
378        if (Context.Message.Attachments.Count != 0)
379        {
380            IAttachment attach = Context.Message.Attachments.First();
381
382            string[] suffixs = attach.Filename.Split(".");
383
384            string suffix = "." + suffixs[suffixs.Length - 1];
385
386            if (suffix == ".txt")
387            {
388                using (var client = new WebClient())
389                {
390                    client.DownloadFile(new Uri(attach.Url), @"..\..\Cache\" + Context.Message.Id.ToString() + suffix);
391
392                    text = Utils.FileUtils.Load(@"..\..\Cache\" + Context.Message.Id.ToString() + suffix);
393
394                    File.Delete(@"..\..\Cache\" + Context.Message.Id.ToString() + suffix);
395                }
396            }
397        }
398
399        if (text == null)
400        {
401            await Context.Message.AddReactionAsync(new Emoji("❎"));
402            return;
403        }
404
405        string cleanText = DectalkReplace(text, Context.Client);
406
407        await DecTalk(@"./audio/" + Context.Message.Id.ToString() + ".wav", cleanText).WaitForExitAsync();
408
409        Console.WriteLine("e");
410
411        string fullPath = new FileInfo(fileName).FullName;
412
413        await vcdttimer(fullPath, Context.Guild);
414
415        await Context.Message.AddReactionAsync(new Emoji("✅"));
```

The function behind the text-to-speech command. Taking a user input, either as message or .txt file, it prepares the message for synthetization with DectalkReplace(). Then, the text is run through Dectalk using a PowerShell command. Finally, the outputted audio file is streamed into a voice call using FFmpeg. This is one of the more recent modules, and it is clear how the organization and abstraction has improved since my earlier code.

Computer Science | Website with HTML, CSS, C#

Portfolio Website 2020 (14 years old)


Engineering CompSci Art Activities

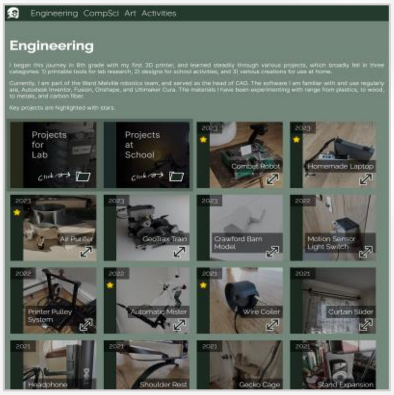
Yiunfan Hu's Portfolio

Croissantderp.github.io

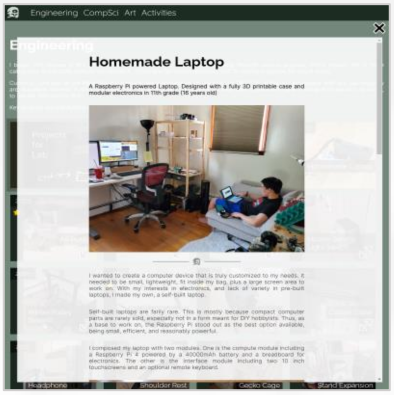
About Me

My Projects

Full Categories


Engineering CompSci Art Activities

Engineering


Engineering CompSci Art Activities

Homemade Laptop

A Raspberry Pi powered Laptop. Designed with a fully 3D printable case and modular electronics in 11th grade (16 years old)

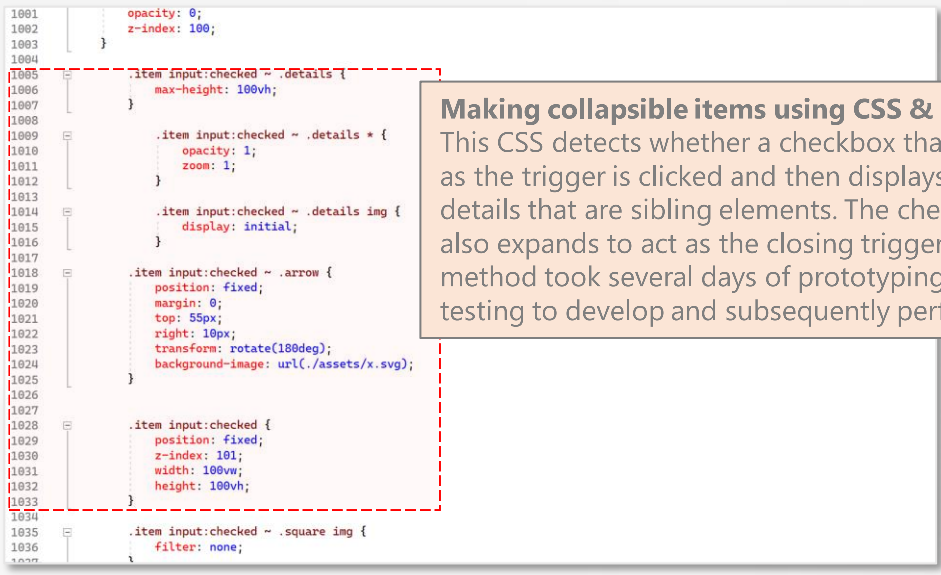
My portfolio website. It is the product of multiple years of development, designs, and redesigns.

[Video Demo] [Source Code: Website, Translator] [Live Website]

One obvious observation of the previous incarnations of my website is the abysmal loading speed, mostly as a result of overusing JavaScript. Thus, My primary goal was to not use JavaScript at all in the current version. This was the biggest head scratcher of the project. However, despite collapsible items and buttons, I made my website fully functional without any JS!

I had over 70 different projects that all needed descriptions, links, and photos. Writing out items in the ide was dull and time-consuming, so I needed a better way to organize and translate projects into HTML. My solution was to create a spreadsheet. This stores all the information, from descriptions, to category, to year. Based on the structure I created, I wrote a C# program to convert the spreadsheet into HTML, separated by category.

Croissantderp.github.io/styles.css | Line 1005-1033 (with CSS)

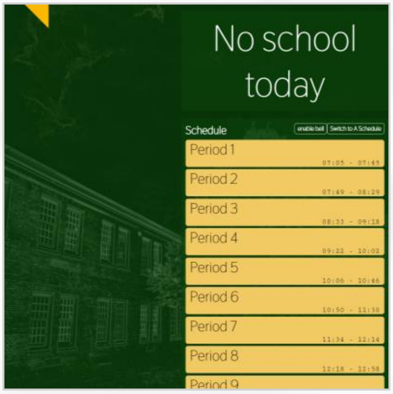


```
1001            opacity: 0;
1002            z-index: 100;
1003        }
1004
1005            .item input:checked ~ .details {
1006                max-height: 100vh;
1007            }
1008
1009                .item input:checked ~ .details * {
1010                    opacity: 1;
1011                    zoom: 1;
1012                }
1013
1014                .item input:checked ~ .details img {
1015                    display: initial;
1016                }
1017
1018            .item input:checked ~ .arrow {
1019                position: fixed;
1020                margin: 0;
1021                top: 55px;
1022                right: 10px;
1023                transform: rotate(180deg);
1024                background-image: url(./assets/x.svg);
1025            }
1026
1027
1028            .item input:checked {
1029                position: fixed;
1030                z-index: 101;
1031                width: 100vw;
1032                height: 100vh;
1033            }
1034
1035            .item input:checked ~ .square img {
1036                filter: none;
```

Making collapsible items using CSS & HTML. This CSS detects whether a checkbox that acts as the trigger is clicked and then displays all the details that are sibling elements. The checkbox also expands to act as the closing trigger. This method took several days of prototyping and testing to develop and subsequently perfect.

TheWard Schedule App

2021 (15 years old)

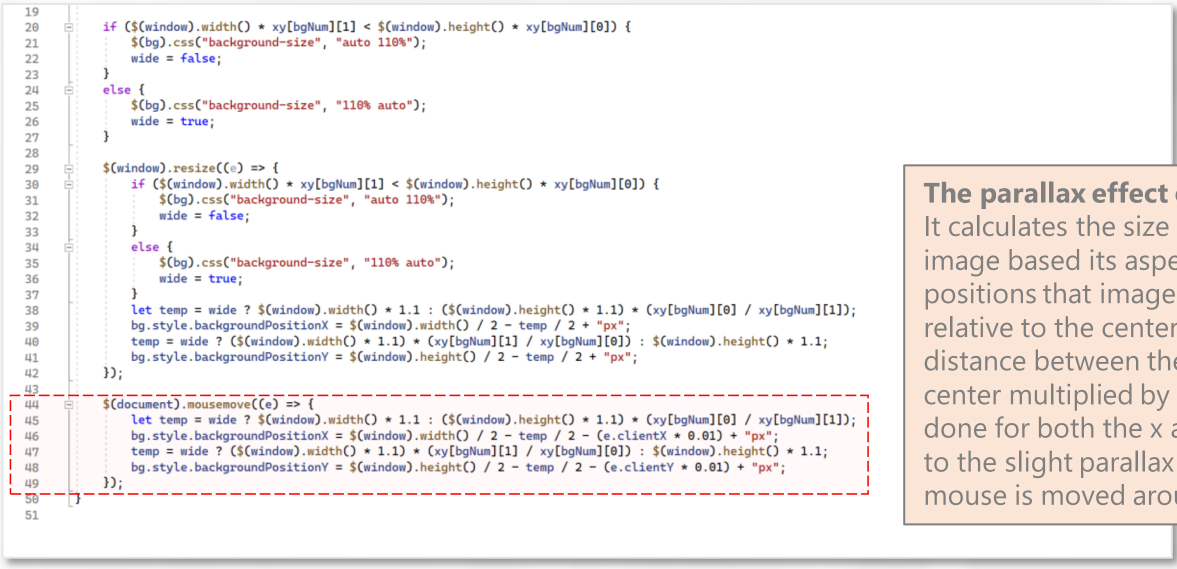


A web app for my new school with schedule, club list, and more.

[Video Demo] [Source Code] [Live Website]

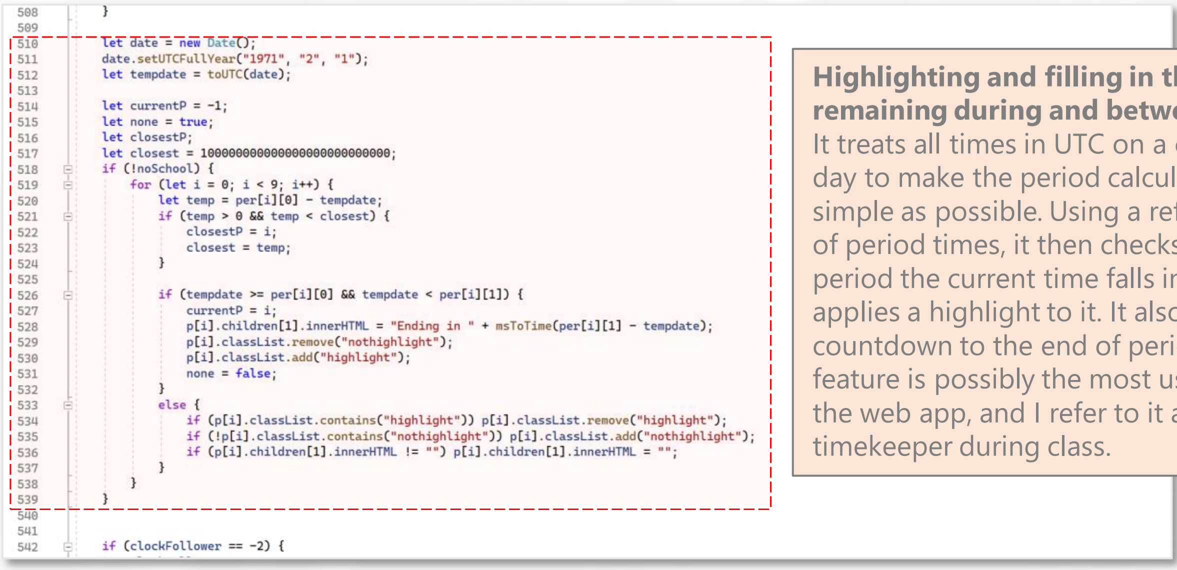
I wanted to create a curricular utility at my new school, Ward Melville. Before the school year began, I made TheWard with some basic features, like schedule and assignments. Throughout the year, I upgraded it with a club list, bell, and a calendar. At the beginning of Junior and Senior year, I updated the website for the new school year.

theWard/public/bg.js | Line 44-49 (with JS)


```
19
20      if ($(window).width() * xy[bgNum][1] < $(window).height() * xy[bgNum][0]) {
21          $(bg).css("background-size", "auto 110%");
22          wide = false;
23      }
24      else {
25          $(bg).css("background-size", "110% auto");
26          wide = true;
27      }
28
29      $(window).resize((e) => {
30          if ($(window).width() * xy[bgNum][1] < $(window).height() * xy[bgNum][0]) {
31              $(bg).css("background-size", "auto 110%");
32              wide = false;
33          }
34          else {
35              $(bg).css("background-size", "110% auto");
36              wide = true;
37          }
38          let temp = wide ? $(window).width() * 1.1 : ($(window).height() * 1.1) * (xy[bgNum][0] / xy[bgNum][1]);
39          bg.style.backgroundPositionX = $(window).width() / 2 - temp / 2 + "px";
40          temp = wide ? ($(window).width() * 1.1) * (xy[bgNum][1] / xy[bgNum][0]) : $(window).height() * 1.1;
41          bg.style.backgroundPositionY = $(window).height() / 2 - temp / 2 + "px";
42      });
43
44      $(document).mousemove((e) => {
45          let temp = wide ? $(window).width() * 1.1 : ($(window).height() * 1.1) * (xy[bgNum][0] / xy[bgNum][1]);
46          bg.style.backgroundPositionX = $(window).width() / 2 - temp / 2 - (e.clientX * 0.01) + "px";
47          temp = wide ? ($(window).width() * 1.1) * (xy[bgNum][1] / xy[bgNum][0]) : $(window).height() * 1.1;
48          bg.style.backgroundPositionY = $(window).height() / 2 - temp / 2 - (e.clientY * 0.01) + "px";
49      });
50  }
51
```

The parallax effect on the background. It calculates the size of the background image based its aspect ratio, then positions that image with a slight offset relative to the center. The offset is the distance between the mouse and the center multiplied by 0.01. This process is done for both the x and y axes, leading to the slight parallax effect when the mouse is moved around.

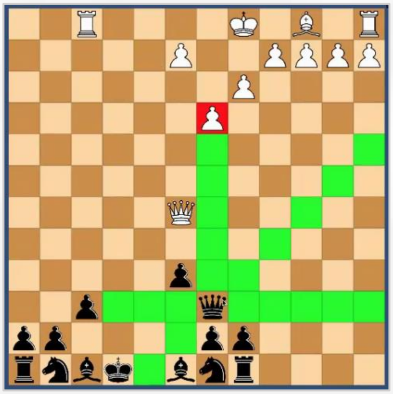
theWard/public/schedule.js | Line 510-539 (with JS)


```
508     }
509
510     let date = new Date();
511     date.setUTCFullYear("1971", "2", "1");
512     let tempdate = toUTC(date);
513
514     let currentP = -1;
515     let none = true;
516     let closestP;
517     let closest = 1000000000000000000000000000;
518     if (!noSchool) {
519         for (let i = 0; i < 9; i++) {
520             let temp = per[i][0] - tempdate;
521             if (temp > 0 && temp < closest) {
522                 closestP = i;
523                 closest = temp;
524             }
525
526             if (tempdate >= per[i][0] && tempdate < per[i][1]) {
527                 currentP = i;
528                 p[i].children[1].innerHTML = "Ending in " + msToTime(per[i][1] - tempdate);
529                 p[i].classList.remove("nothighlight");
530                 p[i].classList.add("highlight");
531                 none = false;
532             }
533             else {
534                 if (p[i].classList.contains("highlight")) p[i].classList.remove("highlight");
535                 if (!p[i].classList.contains("nothighlight")) p[i].classList.add("nothighlight");
536                 if (p[i].children[1].innerHTML != "") p[i].children[1].innerHTML = "";
537             }
538         }
539     }
540
541
542     if (clockFollower == -2) {
```

Highlighting and filling in the time remaining during and between periods. It treats all times in UTC on a consistent day to make the period calculations as simple as possible. Using a reference list of period times, it then checks which period the current time falls in, and applies a highlight to it. It also updates the countdown to the end of period. This feature is possibly the most useful piece of the web app, and I refer to it as my timekeeper during class.

Well Chess but Actually No

2020 (14 years old)



A hackathon project related to chess created with a small team of friends.

[Video Introduction] [Source Code] [Hackathon Submission]

Every year at Gunn High School, the United Computations club would hold GunnHacks; a hackathon where you can get together with up to 3 others and work together to create a project.

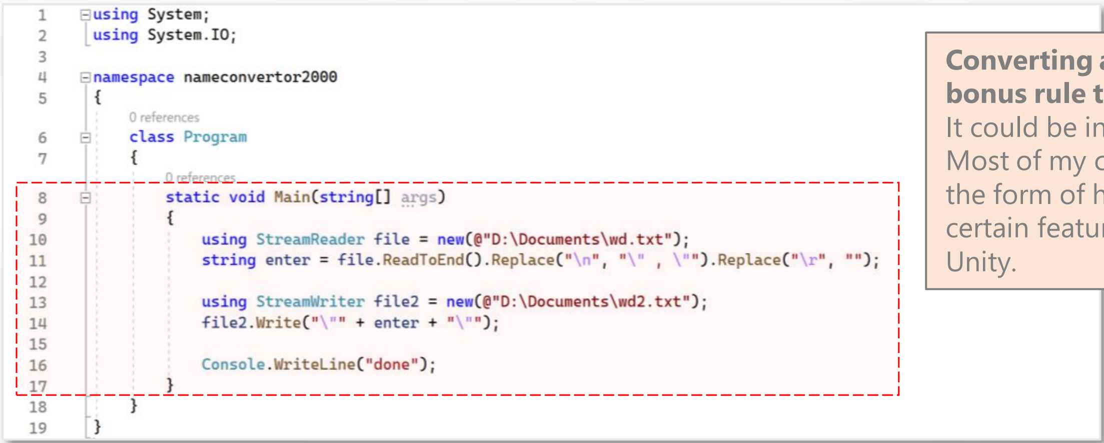
Before I became an organizer for GunnHacks 8.0/ 9.0/ 10.0, a couple friends and I participated in 7.0. For our project, we decided to make online chess, but with a twist! There would be a set of bonus rules which can be added to each game and combined to create a unique experience. My role in this project was management and music, and I also contributed to the code, art, design, and ideas.

For music, I composed 3 tracks in a swing style. Originally, I planned to record the tracks with my violin, but due to the time constraint I used a synthesized piano.

As part of management, I organized and typed up the Devpost submission page. Even though we did not place in the competition, we enjoyed working on the project together and produced a product that we are proud of.

I provided input and helped resolve bugs in the code, but this was a team effort, and the majority of the programming was handled by a teammate.

Nameconvertor2000/Program.cs | Line 8-17 (with C#)



```
1   using System;
2   using System.IO;
3
4   namespace nameconvertor2000
5   {
        0 references
6       class Program
7       {
            0 references
8           static void Main(string[] args)
9           {
10              using StreamReader file = new(@"D:\Documents\wd.txt");
11              string enter = file.ReadToEnd().Replace("\n", "\" , \"").Replace("\r", "");
12
13              using StreamWriter file2 = new(@"D:\Documents\wd2.txt");
14              file2.Write("\"" + enter + "\"");
15
16              Console.WriteLine("done");
17          }
18      }
19  }
```

Converting a spreadsheet of data for a bonus rule to a comma separated list. It could be inserted straight into Unity. Most of my code contribution were in the form of helping to debug and test certain features using my background in Unity.

Bin Packer

2020 (14 years old)

```
rossingover of two parents
erence
tic (int [], int []) crossover(int [] parent1, int [] par

 Random r = new Random();

 //length of a solution
 int len = parent1.Length;

 //children
 int [] child1 = new int [len];
 int [] child2 = new int[len];

 //the point where the solutions swap
 int splitPoint = r.Next(len);

 //creating the children
 for (int i = 0; i < len; i++)
 {
     //if a mutation has occured
     int mutation = 0;
     if (r.NextDouble() < mut
     {
         //decides which c
         mutation = r.Ne
     }

     //swaps which c
     if (i < splitP
     {
         child1[
```

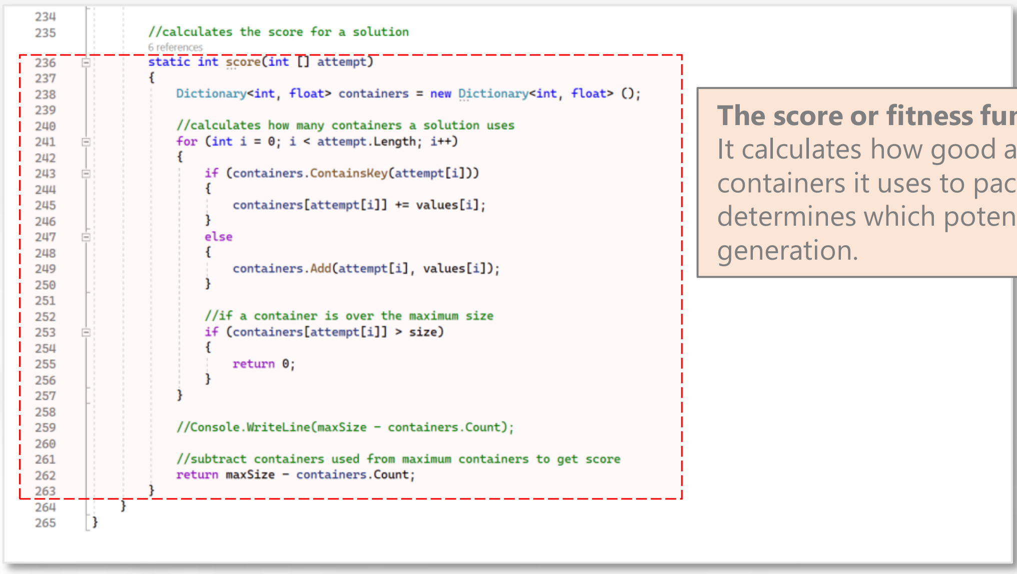
An evolutionary solution to the bin packing problem.

[Video Demo] [Source Code]

For one of my projects, I needed to know how many wood planks I needed to buy. I knew the sizes of all the final pieces I needed, but I had to fit those pieces onto the planks in a way to minimize the amount of planks used.

This turned out to be an application of the "bin packing problem", and I did some research into possible solutions. The solution I used was a genetic algorithm, which gradually approaches a solution using a similar process as natural selection. The algorithm got an answer of 7 planks, which after checking with another program, was indeed the minimum amount of planks needed. From this project, one of my main takeaways was how a genetic algorithm could be used to solve difficult problems, without actually "solving" them outright.

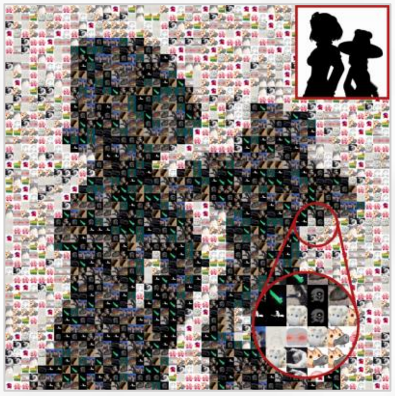
BinPacking/Program.cs | Line 236-263 (with C#)



```
234
235             //calculates the score for a solution
                6 references
236             static int score(int [] attempt)
237             {
238                 Dictionary<int, float> containers = new Dictionary<int, float> ();
239
240                 //calculates how many containers a solution uses
241                 for (int i = 0; i < attempt.Length; i++)
242                 {
243                     if (containers.ContainsKey(attempt[i]))
244                     {
245                         containers[attempt[i]] += values[i];
246                     }
247                     else
248                     {
249                         containers.Add(attempt[i], values[i]);
250                     }
251
252                     //if a container is over the maximum size
253                     if (containers[attempt[i]] > size)
254                     {
255                         return 0;
256                     }
257                 }
258
259                 //Console.WriteLine(maxSize - containers.Count);
260
261                 //subtract containers used from maximum containers to get score
262                 return maxSize - containers.Count;
263             }
264         }
265     }
```

The score or fitness function for the genetic algorithm. It calculates how good a solution is by how many containers it uses to pack items. This is the function that determines which potential solution will survive each generation.

Bad Apple!! 2022 (16 years old)



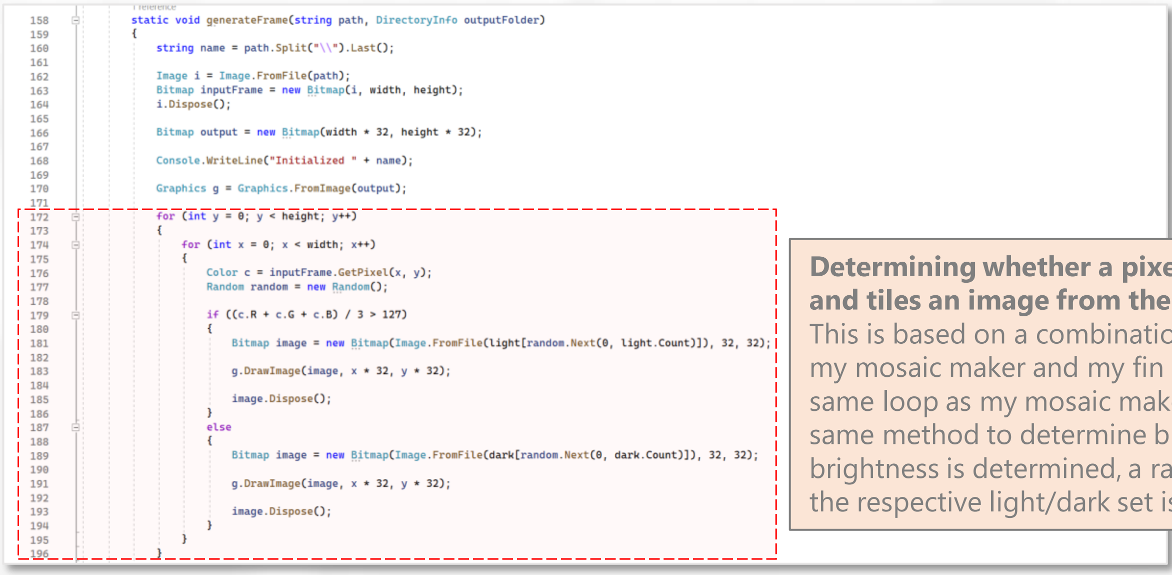
A large collaborative Christmas gift involving "Bad Apple!!", image processing, music arranging, and recording. Done for the holiday season of 11th grade (16 years old).

[Video Product] [Source Code]

Every year at Gunn High School, the orchestra holds a white elephant gift exchange. In 2022, alongside a team of seven others, we sought to make a cover of Bad Apple!!. The project was ambitious, as we wanted to recreate every component in it, from visual elements, song arrangement, to performing the music. We had a complex challenge ahead of us.

My main role was to create the video. Based on the experience gained from my previous "Geckoinator" mosaic maker, which replaces pixels in an image with smaller tiled images, as I couldn't reuse the mosaic maker due to it inevitably only using the single brightest/darkest image, I made an adaptation of the software for the new purpose. I inputted the frames of Bad Apple into my program and then replace each pixel with a random "light" or "dark" image based on brightness. I ran the program in batches of 1000 frames over the span of six days. Once the frames were ready, I utilized FFmpeg to stitch the video together.

BadappingStick/badappingStick/Program.cs | Line 8-17 (with C#)



```
static void generateFrame(string path, DirectoryInfo outputFolder)
{
    string name = path.Split("\\").Last();

    Image i = Image.FromFile(path);
    Bitmap inputFrame = new Bitmap(i, width, height);
    i.Dispose();

    Bitmap output = new Bitmap(width * 32, height * 32);

    Console.WriteLine("Initialized " + name);

    Graphics g = Graphics.FromImage(output);

    for (int y = 0; y < height; y++)
    {
        for (int x = 0; x < width; x++)
        {
            Color c = inputFrame.GetPixel(x, y);
            Random random = new Random();

            if ((c.R + c.G + c.B) / 3 > 127)
            {
                Bitmap image = new Bitmap(Image.FromFile(light[random.Next(0, light.Count)]), 32, 32);

                g.DrawImage(image, x * 32, y * 32);

                image.Dispose();
            }
            else
            {
                Bitmap image = new Bitmap(Image.FromFile(dark[random.Next(0, dark.Count)]), 32, 32);

                g.DrawImage(image, x * 32, y * 32);

                image.Dispose();
            }
        }
    }
```

Determining whether a pixel is light or dark and tiles an image from the corresponding set. This is based on a combination of experience from my mosaic maker and my fin quantifier. It uses the same loop as my mosaic maker, and uses the same method to determine brightness. After brightness is determined, a random image from the respective light/dark set is chosen and tiled.

Art

I have played violin since kindergarten and later branched into music composition and arrangement. I also picked up my sewing skills from my grandma and mom around 3rd grade and started projects in middle school.

From music, to fabric arts, to crochet, to 3D artistic modeling, to gardening, and more, my crafting projects are a diverse collection.

Inner Coating of a Gourd Bottle (2023)

Gourd Bottle

2022-2023 (16-17 years old)

A gourd-based water bottle. Grown, dried, and sealed over the course of a year.

After moving to Long Island, it was the first time in my life I had a yard and plenty space to garden. When it came time to start the garden in spring, I had an idea to grow and make a natural water bottle using gourds.

The whole gourd growing and water bottle making process was quite a learning and fun experience to me.

Spring 2022, I obtained the seeds from Home Depot and planted them in a line next to a fence. Throughout the summer, I took care of the plants which slowly meandered up the fence, dangling their fruit below. Soon, it became apparent that I planted a few too many, as the vines took over the entire corner of the garden. Rookie mistake!

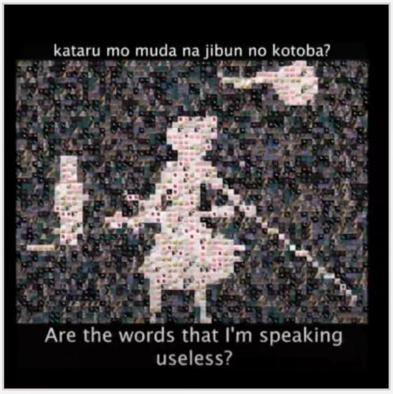
Fall 2022, after the gourds had fully matured, I cut them off the vine and put them in mesh bags to hang. For a good place to let them sit, I dangled them off a chair on the porch.

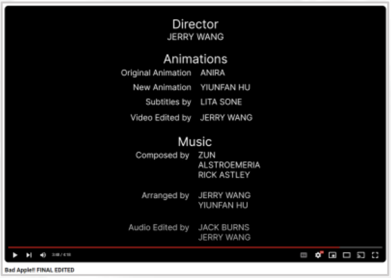
Spring 2023, after the winter, they were completely dried, taking a whole year to arrive at this step. In April, I cleaned the mold off the gourds with water and sandpaper. Cutting them open, I shook up the gourds to dislodge and remove the seeds with gravel, leaving a perfect empty shell.

I poured molten beeswax into the gourd and gave it a thorough inner-coating. I then collected the leftover wax, mixed with mineral oil, to create some easy wood polish, and buffed up the outside.

Despite the long timescale of this project, I greatly enjoyed every step along the way. The success of crossing my gardening hobby into my maker mentality left me excited for future possibilities, and for expanding my projects to new and interesting fields.

Bad Apple!! Music Video 2022 (16 years old)


kataru mo muda na jibun no kotoba?
Are the words that I'm speaking useless?



Director
JERRY WANG

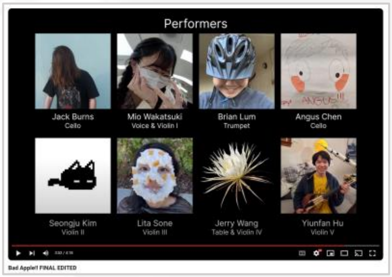
Animations

Original Animation	ANIRA
New Animation	YIUNFAN HU
Subtitles by	LITA SONE
Video Edited by	JERRY WANG

Music

Composed by	ZUN ALSTROEMERIA RICK ASTLEY
Arranged by	JERRY WANG YIUNFAN HU
Audio Edited by	JACK BURNS JERRY WANG

Bad Apple!! FINAL EDITED



A large collaborative Christmas gift involving "Bad Apple!!", animation recreating, music arranging, and recording. Done for the holiday season of 11th grade.

[Video Product] [Sheet Music]

Every year at Gunn High School (Palo Alto, CA), the orchestra holds a white elephant gift exchange. In 2022, while I moved to New York already, I worked alongside a team of seven friends in Palo Alto to make a cover of Bad Apple!!. The project was ambitious, as we wanted to recreate every component in it, from visual elements, and song arrangement, to performing the music. We had a complex challenge ahead of us.

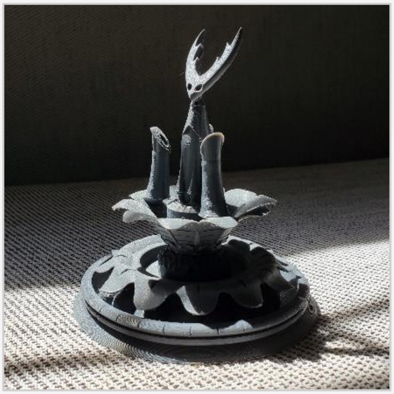
I contributed on multiple fronts of this project, as well as the final stage to integrate all recreated components into one product. With my knowledge of programming video and image processors (this part of the project is described on page 43), I was in charge of the creation of the video. Using my experience in musical arrangements and playing violin, I was also involved in the arrangement of the song for different instruments in our group.

For the music arrangement, with 5 violinists, 2 cellos, a trumpet, and a voice, we had to create something new for our unique group composition. I started off by creating the backbone of the score, adding in all the main rhythms and melodies. Then, my friend finished the arrangement, filling in the gaps, polishing, and proofreading. To push our available instrumental range beyond a classic violin, I recorded the Violin V part using my Yamaha 5-string electric violin.

My final task was to make a credit frame, which I created using my Figma skills, as well as assisting in the design of the bilingual captions. Then the video, audio, captions, and credits, were combined together to produce the final video.

Fountain Statue

2020 (14 years old)



A 3D-printable recreation of a fountain from a video game.

[STL Files]

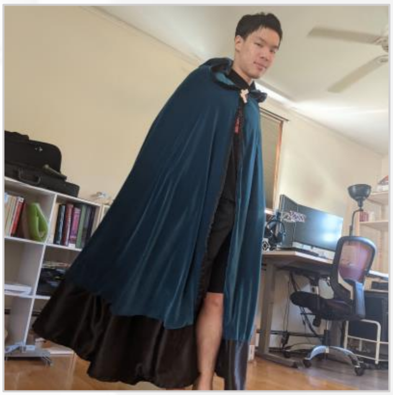
The object I wanted to design and print was the fountain from the City of Tears from the video game Hollow Knight. The original purpose was as a decoration for my geckos, but it works just as well as a desk ornament.

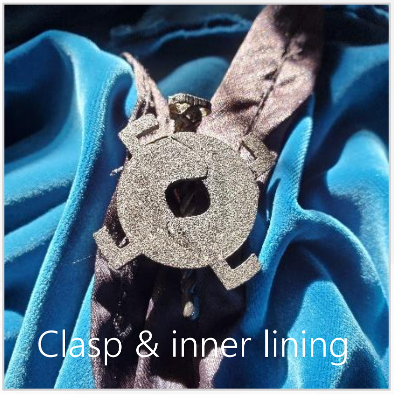
Inventor is not intended to be used for artistic purposes, but the symmetry of the fountain simplified the process. The result is one of the most complex designs I have ever made in terms of feature count.

This project allowed me to fully explore the suite of tools Inventor offered. I learned how to use the loft feature for complicated 3D geometry and combine multiple tools to produce fine detail.

Cloak

2019 (13 years old)





A full body wearable velvet cloak started.

[STL Files]

On one of the expeditions with my mom in the Palo Alto Joann Fabric and Crafts store, I found a bolt of turquoise velvet and I wanted to make a cloak with it.

Before getting started, I researched how to make cloaks. After figuring out that you need a semicircle of fabric. I used AutoCAD to draw a suitable design that would fit on the piece of velvet I had. Then, with help from my mom, I made sewing patterns to assist cutting out the cloak.

I used a combination of sewing machine and hand sowing to stitch the parts together. After the cloak itself was complete, I needed a clasp to finish it. I used 3D printing to create a modular clasp that let me design decorative elements in the future which could then attach to the existing clasp. The result was a rotationally symmetric three-part clasp which used a pin to secure the two halves together.

In fall of 2022, I decided to make some changes after moving to New York. Because of the much colder winters, I bought some warm fabric to line the inside of the cloak.

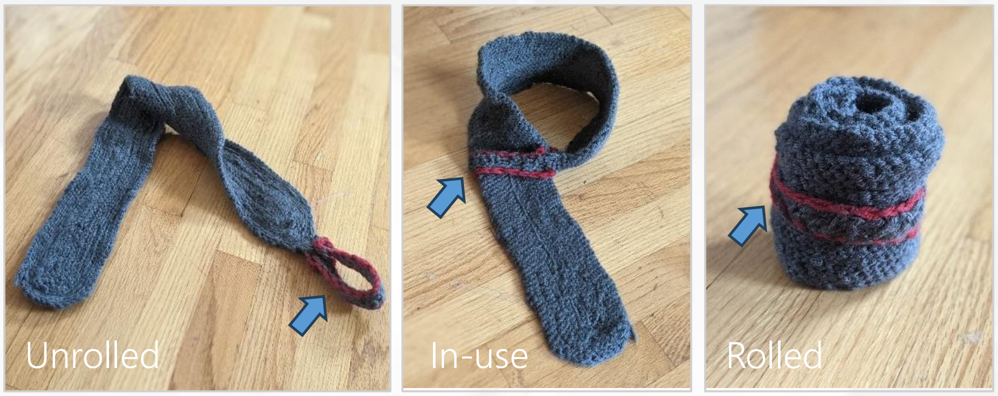
Crochet Scarf

2018 (12 years old)

A crochet scarf made in 7th grade.

When I was 12, I learnt how to crochet from my grandma. To test my skills, I decided to crochet a scarf. I had an idea to make the scarf easy to use and store, by *adding a band which could be used to keep the scarf rolled up*. I worked whenever I had free time and finished in about a month.

After finishing, my dad bought the scarf for $20.



Ace of Hearts Comforter

2017 (11 years old)

A card themed comforter sewed in 6th grade.

My first major sewing project. I made this twin-sized cover when I was 11 years old after learning how to sew a year prior. It is designed to be flippable for temperature control. The bottom side (blue) is made of a cooler fabric for warmer nights, while the top side is made of a warmer polyester for colder ones.

I am still using it to this day.

Thank you for your time !