I do a lot of personal programming projects. You can find links to my projects and GitHub profile at

lunaphippscostin.com

In particular, I make free software video games and post them on the site

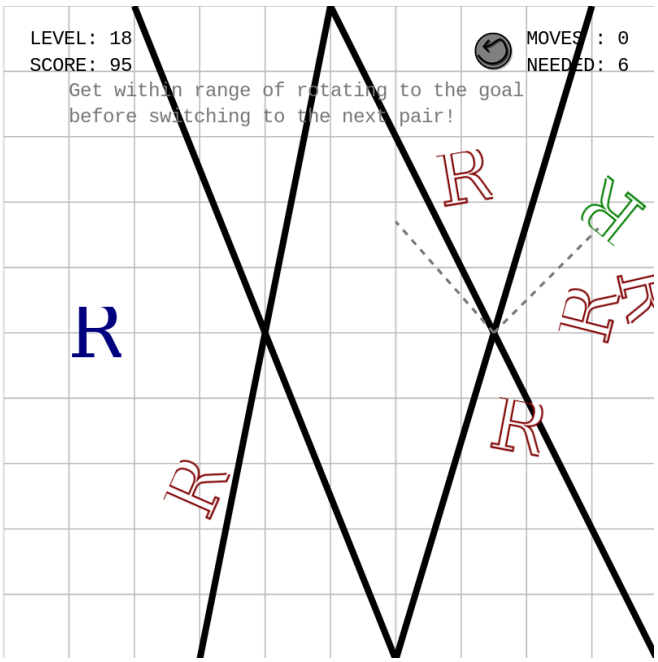
cosinegaming.com

Here are some of my projects that are more mathematical in nature.

```haskell
-- Project Euler Problem 12

triangles :: [Int]
triangles = scanl1 (+) [1..]

-- Taken from 3.hs / Problem 3
factorize :: Int -> [Int]
factorize 1 = [1]
factorize x
    | factor == x = [x]
    | otherwise = factor:(factorize $ x `div` factor)
    where factor = head $ filter (\c -> x `mod` c == 0) (2:[3,5..])

-- Given prime factors, find total factors
-- Doesn't count - uses algebraic formula, product of (exponents + 1)
factorcount :: [Int] -> Int
factorcount [] = 1 -- Itself; ie (n)^0, exp+1 = 1
factorcount factorlist = (countfirst + 1) * factorcount rest
    where countfirst = length $ filter (== firstfact) factorlist
          rest = filter (/= firstfact) factorlist
          firstfact = factorlist !! 0

numfactors n = factorcount $ factorize n

p12 = head $ filter (\(n,f) -> f>500) trianglefactors
    where trianglefactors = [(n, numfactors n) | n <- triangles]

main = print $ p12

```

The programming language haskell is a beautiful functional programming language (an unusual type of language). I wanted to learn this language, so I decided to solve problems from Project Euler. To the left is my solution to Problem 12, “What is the value of the first triangle number to have over five hundred divisors?” The output is **76576500**.

All of my solutions are visible on my GitHub in the repository **euler**.



Here’s a game I made, called **Flip**. It’s one of my favorite games I’ve made, and one of the most mathematical. Your character is that blue “R,” and your goal is to make it to the red “R.” Clicking on one of those black bars reflects your character over that axis. This is an interesting problem, because it asks the player to find the solution Coxeter group, given certain possible reflections. Moreover, I had to learn the mathematics of reflections in a way that I can present it in a puzzle game without sounding like math!

Another fun thing to point out is the code used to calculate the “R”s’ positions. All of the reflections are calculated using Matrix multiplication (so the browser can render the transformation). Here’s the Javascript I used to calculate these matrices:

```
var b = line[1] - m*line[0];
var q1 = m + (1 / m);
var q2 = 1 + m*m; // ...
flip = new SVG.Matrix(2/m/q1 - 1, 2*m/q2, 2/q1, 2*m*m/q2 - 1, -2*b/q1, 2*b/q2);
```

Another fun especially mathematical project is my [subreddit classifier](). It downloads images from the social media site [reddit](). Reddit has subfora (“subreddits”) that focus on specific topics. I created a Convolutional Neural Network that attempts to classify what subreddit an image should belong to.

The project, when it had all of its data, was able to classify certain subreddits with 90% accuracy (unfortunately the >20GB of data were lost in a HDD failure)

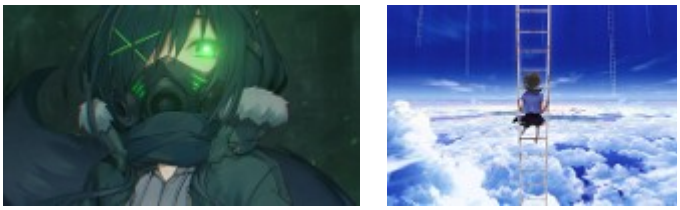
Pictured is the classifications made (with less data and training) of the subreddit “animewallpaper” vs all combined subreddits (“all”).

Here are some randomly selected images and how they were classified:

Classified as /r/**all**

The first image is in fact from /r/animewallpaper, and “all” recieves around 40% false-negatives.

Classified as /r/**animewallpaper**



As you can see, /r/animewallpaper receives very few false positives. (In fact, the network is trained for this, as false-positives are more destructive in my goals than false-negatives).

Here is the data on the last epoch (of the only 10 I was able to run in my re-creation)
`839/839 - 79s - loss: 0.2714 - precision: 0.8927 - matthews_correlation: 0.7855 - val_loss: 1.0235 - val_precision: 0.6302 - val_matthews_correlation: 0.2604`

The “val_precision” represents the precision for the testing (not before seen) data. The loss represent a function that the computer learns from to see how well it is performing.