CMPSC/MATH 455 Honors Option

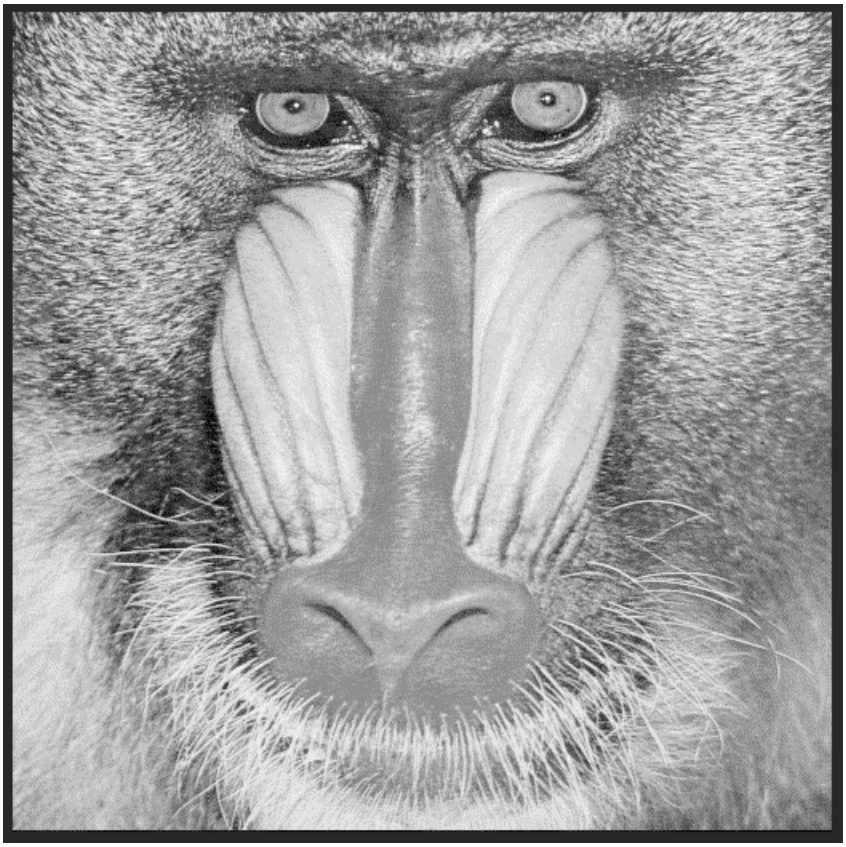
Jared Gluck

```
vagrant@ubuntu-focal:/vagrant/ch13$ ./wavelet-test
original vector:
  1.0000   0.5000   0.3333   0.2500   0.2000   0.1667   0.1429   0.1250

transformed vector:
  0.3397   0.1250   0.1620   0.0208   0.1811   0.0083   0.0175   0.0045

reconstructed vector:
  1.0000   0.5000   0.3333   0.2500   0.2000   0.1667   0.1429   0.1250

original matrix:
  1.0000   0.5000   0.3333   0.2500   0.2000   0.1667   0.1429   0.1250
  0.5000   0.3333   0.2500   0.2000   0.1667   0.1429   0.1250   0.1111
  0.3333   0.2500   0.2000   0.1667   0.1429   0.1250   0.1111   0.1000
  0.2500   0.2000   0.1667   0.1429   0.1250   0.1111   0.1000   0.0909

transformed matrix:
  0.2238   0.0500   0.0732   0.0119   0.0934   0.0056   0.0121   0.0032
  0.0393   0.0295   0.0333   0.0029   0.0314   0.0008   0.0016   0.0004
  0.0604   0.0333   0.0417   0.0048   0.0432   0.0016   0.0031   0.0007
  0.0107   0.0029   0.0048   0.0008   0.0061   0.0004   0.0007   0.0002

reconstructed matrix:
  1.0000   0.5000   0.3333   0.2500   0.2000   0.1667   0.1429   0.1250
  0.5000   0.3333   0.2500   0.2000   0.1667   0.1429   0.1250   0.1111
  0.3333   0.2500   0.2000   0.1667   0.1429   0.1250   0.1111   0.1000
  0.2500   0.2000   0.1667   0.1429   0.1250   0.1111   0.1000   0.0909
```

Above you can see the matrix and vector being transformed then reconstructed, showing that it works fully.

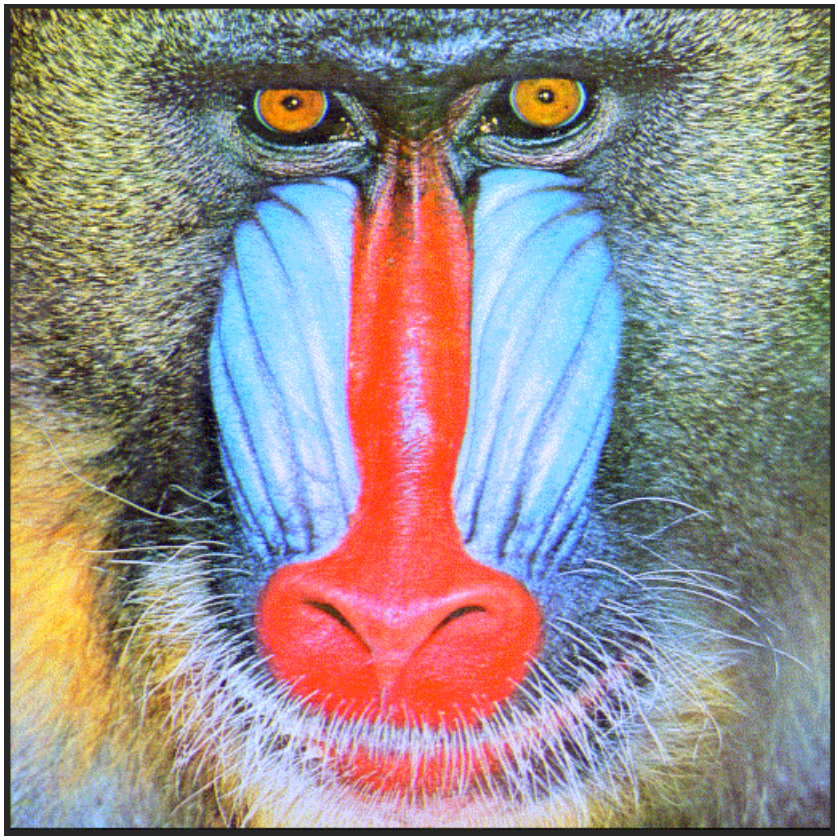
Chapter 14

Original Images:



Above is the PGM of the image I used for this chapter. The black border is because I copy/pasted it from a screenshot to transfer to Microsoft word.

PPM on the next page.

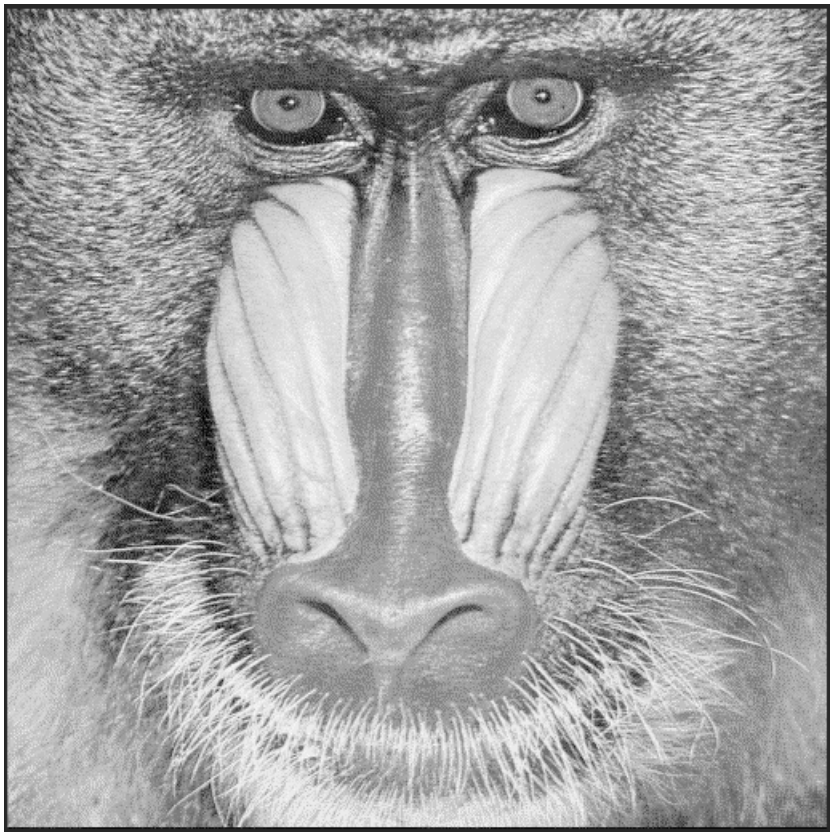


Above is the PPM image.

Test 0

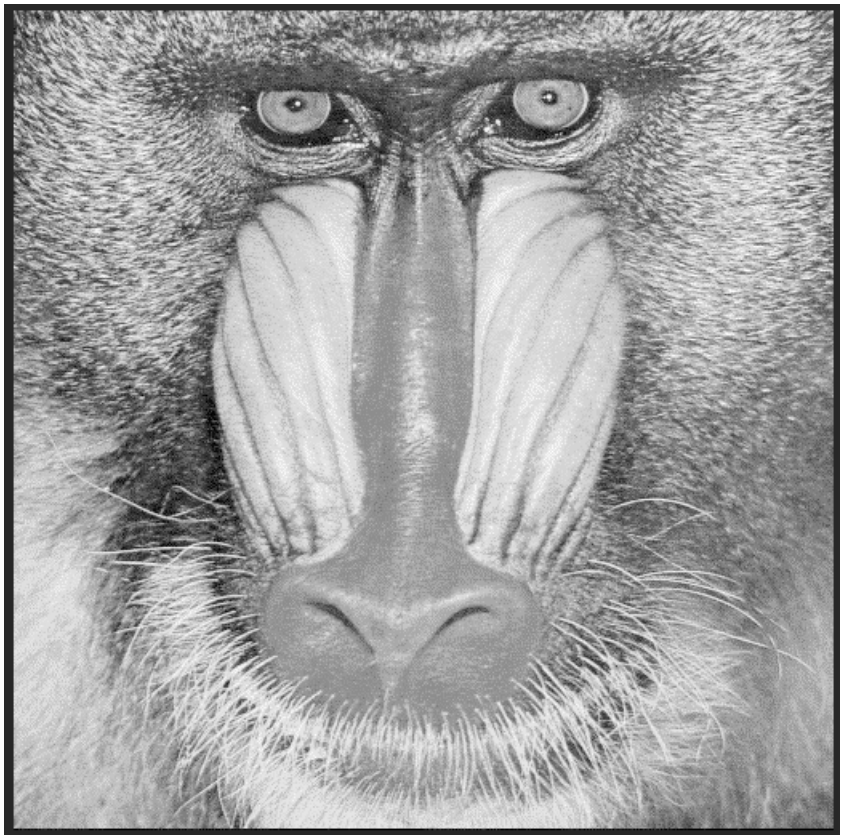
The first test image, called test.pgm, is just showcasing that the program compiles and is able to output an image. There are no modifications, so it is the same as the first pgm above.

Test 1



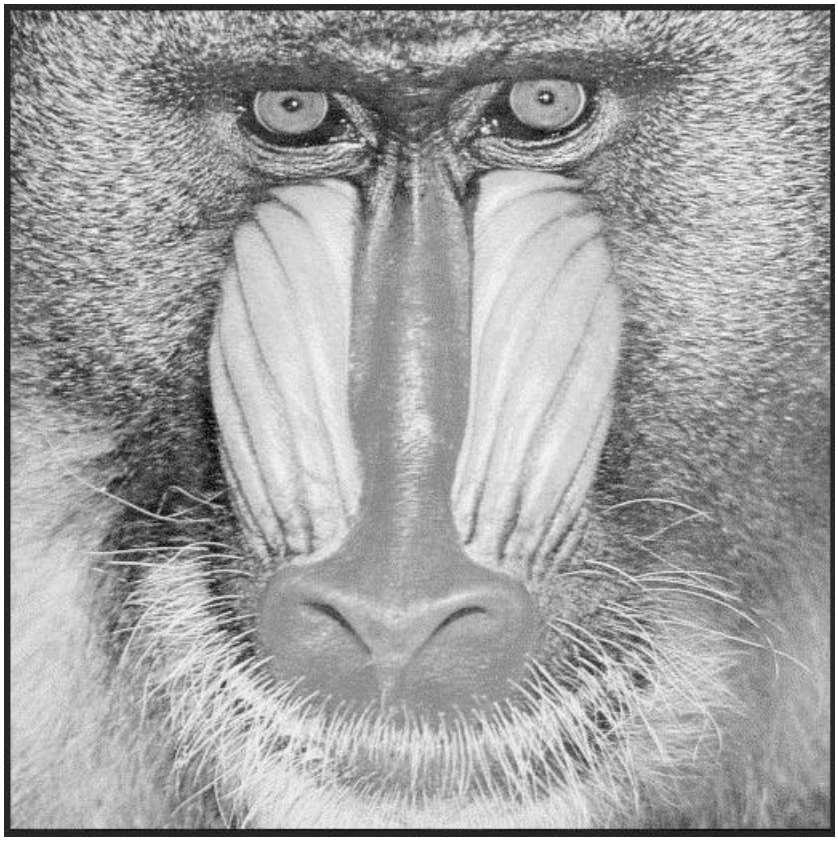
In the above test, I passed a ppm file (color) and converted it to greyscale. If you study this and the original pgm image, you will see they are slightly different.

Test 2



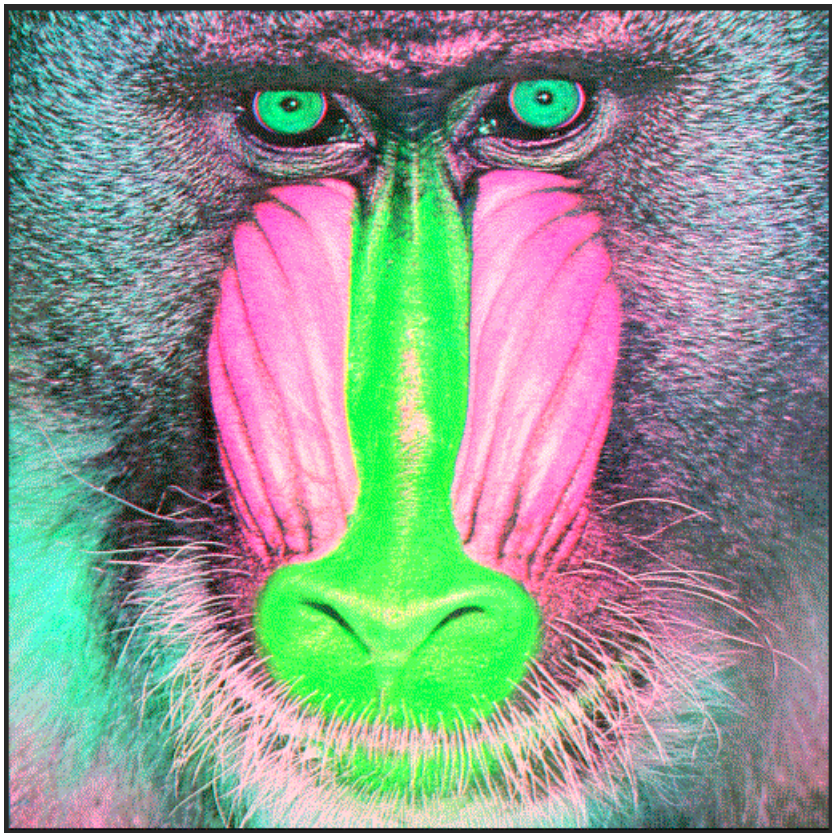
The above test is similar to Test 1, but this time the colors were weighted to try to present a more natural looking black and white image.

Test 3



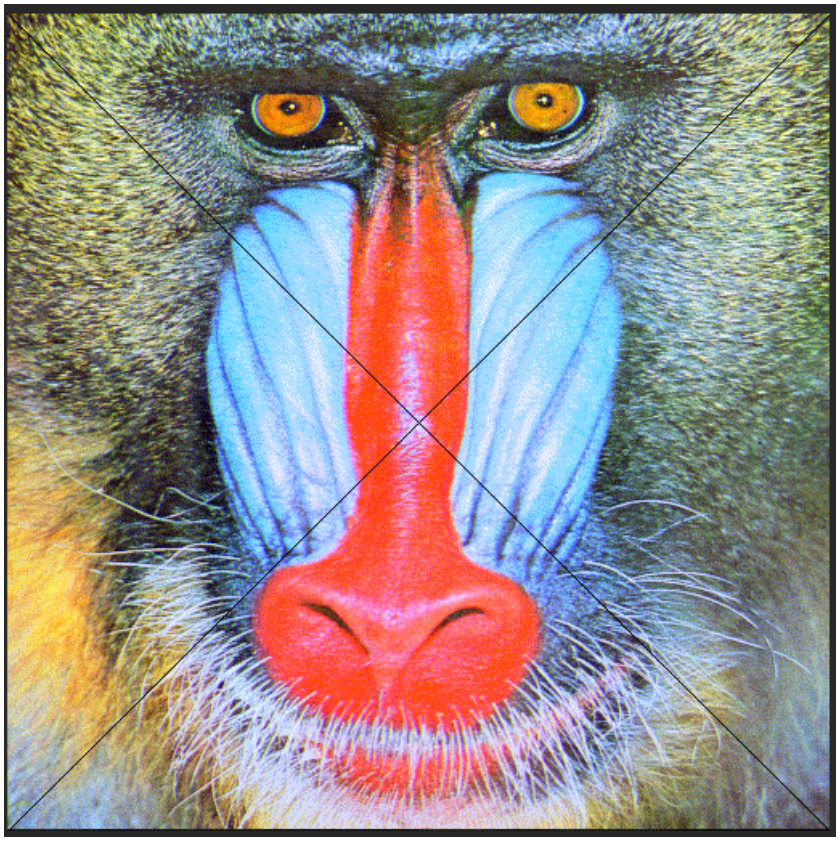
Yet another way to convert to black and white. There are slight differences between this and the previous two.

Test 4



The above test rotates the colors of each pixel by replacing the green values with red values, blue values with green values, and red values with blue values.

Test 5



This test case just draws a black X over the image.

Chapter 15

Unedited PGM photo

Unedited PPM photo

Above are the two unedited photos. What this chapter does is applies the haar wavelet transformation algorithm to modify the image. Below you will see the modifications of the PGM.

relative_error = 0.03

relative_error = 0.05

relative_error = 0.07

relative_error = 0.09

And below will the modifications done to the PGM image.

relative_error = 0.03

relative_error = 0.05

relative_error = 0.07

relative_error = 0.09

Chapter 16

This module's tests are just terminal outputs showing that the linked list functions work properly.

Test 1

```
vagrant@ubuntu-focal:/vagrant/ch16$ ./test1
the original linked list:
12 1000 -45 101
```

Above we can see that we are able to create a linked list

Test 2

```
vagrant@ubuntu-focal:/vagrant/ch16$ ./test2
the original linked list:
12 1000 -45 101
The list after popping 2 nodes
-45 101
the list after popping 3 more nodes
```

This second test shows that we can pop nodes from the linked list. Notably, if we pop more nodes than exists (here we pop 5 times and only have 4 nodes) there are no segfaults, showing that we are properly checking to make sure the access and free are valid.

Test 3

```
vagrant@ubuntu-focal:/vagrant/ch16$ ./test3
the original linked list:
12 1000 -45 101
linked list memory freed
```

This test just frees the linked list memory when we are done with it

Test 4

```
vagrant@ubuntu-focal:/vagrant/ch16$ ./test4
the original linked list:
12 1000 -45 101
the reversed linked list:
101 -45 1000 12
```

Here we see we are able to properly reverse the linked list.

Test 5

```
vagrant@ubuntu-focal:/vagrant/ch16$ ./test5
the original linked list:
12 1000 -45 101
the linked list sorted in ascending order:
-45 12 101 1000
the linked list sorted in descending order:
1000 101 12 -45
linked list memory freed
```

Here we showcase being able to sort in ascending or descending order.

Test 6

```
vagrant@ubuntu-focal:/vagrant/ch16$ ./test6
the original linked list:
12 1000 -45 101
the list after removing odds:
12 1000
the removed items:
101 -45
linked list memory freed
```

This test showcases the filter function, which allows us to remove elements from the linked list if they satisfy some criteria. Here, we are filtering out odd number values.

Test 7

```
vagrant@ubuntu-focal:/vagrant/ch16$ ./test7
the original linked list:
12 1000 -45 101
the list has 4 nodes
linked list memory freed
```

Finally, this test just calculates how many nodes the linked list has

.

Chapter 17

I won't be pasting the entire .wdf, as they all are in the repository, but I will include small snippets to show the trends.

Out-no-eden-1000.wdf

#(i j)	d	e	g[0]	g[1]	g[2]	g[3]	g[4]	g[5]	g[6]	g[7]
(4, 2)	6	155	5	6	5	5	5	4	5	5
(8, 10)	6	235	5	5	5	5	5	4	5	5

We can see no trends

Out-no-eden-10000.wdf

#(i j)	d	e	g[0]	g[1]	g[2]	g[3]	g[4]	g[5]	g[6]	g[7]
(18, 13)	6	84	10	6	7	5	4	4	4	6
(21, 26)	6	174	7	6	8	5	4	4	5	3

Very minor trends to g[1]

Out-no-eden-100000.wdf

#(i j)	d	e	g[0]	g[1]	g[2]	g[3]	g[4]	g[5]	g[6]	g[7]
(6, 15)	4	204	21	18	2	1	1	2	6	9
(17, 14)	7	192	20	19	1	1	1	2	6	9

Strong g[0] and g[1] trends, some g[7] trends

Out-no-edens-1000000.wdf

#(i j)	d	e	g[0]	g[1]	g[2]	g[3]	g[4]	g[5]	g[6]	g[7]
(2, 7)	1	106	53	13	3	2	1	1	2	8
(13, 10)	4	175	53	13	3	2	1	1	2	9

Extremely dominant g[0] and some strength to g[1] and g[7]

Out-no-edens-10000000.wdf

#(i j)	d	e	g[0]	g[1]	g[2]	g[3]	g[4]	g[5]	g[6]	g[7]
(21, 7)	3	15	137	28	1	1	1	2	1	34
(24, 22)	6	233	137	26	1	1	2	2	1	34

Same trends as before, just even higher numbers.

Out-with-eden-1000.wdf

#(i j)	d	e	g[0]	g[1]	g[2]	g[3]	g[4]	g[5]	g[6]	g[7]
(0, 4)	6	228	6	5	5	5	5	5	5	5
(4, 0)	4	241	6	4	5	5	5	5	4	6

No interesting trends

Out-with-eden-10000.wdf

#(i j)	d	e	g[0]	g[1]	g[2]	g[3]	g[4]	g[5]	g[6]	g[7]
(0, 1)	3	159	4	5	10	9	9	6	5	4
(0, 2)	1	175	2	4	10	9	8	6	7	5

Some dominance with g[2], g[3], g[4], g[5]. This would correlate to animals that found the eden

Out-with-eden-100000.wdf

(25, 5)	5	105	1	2	24	12	5	10	3	3
(0, 26)	3	169	1	1	24	11	3	9	3	3
(22, 28)	6	133	14	12	1	2	2	1	1	7
(32, 29)	1	18	15	12	1	1	1	2	2	9

Here we see two different animals. One has dominant g[2], g[3] and the other has dominant traits g[0], g[1]

Out-with-eden-1000000.wdf

(15, 20)	2	24	5	1	89	28	4	6	1	1
(24, 3)	3	14	4	2	88	28	5	3	1	2
(14, 24)	2	103	25	25	3	3	1	3	2	39
(11, 25)	4	90	27	26	2	3	1	3	2	37

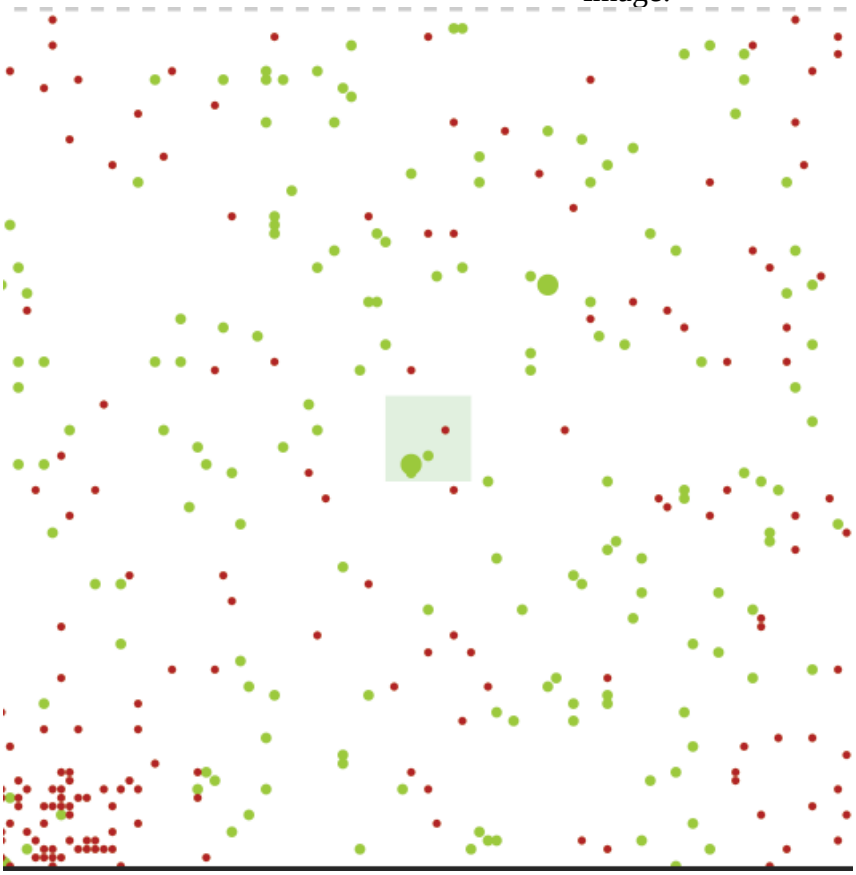
We see the same trend as before, but this time its much more dominant the difference between the two animals are larger.

Out-with-eden-10000000.wdf

(1, 12)	1	167	98	17	1	1	3	1	4	17
(13, 13)	2	79	5	45	314	1	1	2	4	3
(0, 13)	3	49	6	44	314	1	2	3	4	1

Here we see the same trend. It seems like the creatures who live in the eden are a lot better off, since their better gene mutation is more extreme than the ones in the desert.

Image:



Above is one of the output images. Red are the animals, green are the food. I did not put them all together into a gif as the author did, but I included one to just to showcase this functionality is working. All the .eps files are in the repository.