Lab 5

Exercise 1

Maybe to improve it, pause on the title frames for a bit so they're readable but otherwise I don't know, it seems to work well

Exercise 2

The movement was smooth. For something this barebones there isn't much more you could do.

Lab 6

Exercise 1

You could remove the downwards acceleration and add enemies coming into the screen from random angles, having to shoot at them

Exercise 2

A – Reducing the mass makes the spring much bouncier and increasing it makes it slower and squishier I guess. It makes it heavier and lighter. Changing the constant really messes with the spring if you change it too much, it tended to just end up flashing on the entire screen for me. The damping changes how long it stays springy for, if it goes above one it makes the spring bounce more and more till it breaks the sketch. The rest position just changes where it goes to.

C – If you could connect 2 springs together, have one on a hinge and the other have a fixed angle, that could be interesting.

Exercise 3

Creation – Particles are created once per frame every frame by pushing a new particle into an array in the ParticleSystem.

Attributes – Origin – Where the particles are originating from, which is width/2, 50. They each start at a random velocity, with a lifespan of 255 that reduces by 2 each frame. Their opacity is equal to their lifespan.

Deletion – When their lifespan goes below 0 they are spliced from the array

Dynamics – They start with a random velocity, and all accelerate downwards at a rate of 0.05 pixels per frame.

Rendering – They are ellipses with fills of 127 greyscale and the alpha is their current lifespan.

Lab 7

Exercise 1

No the code doesn’t handle -0.3, -0.4, because p5 doesn’t do angles how maths does. So use the cross product and if its positive then the arrows going right, if its negative the arrows going left, (or maybe its flipped), so then just add a minus sign to the rotate function

You could do with something like acos(velocity.dot(dir)/(velocity.mag()*dir.mag())) but when angleBetween() works just fine I don’t see why you would.

Exercise 2

The worked after changing some parameters to make it a bit more user friendly. Oh and I gave him a top hat.

Exercise 3

A x B is the inverse of B x A in code. At least that’s what it looks like.

Lab8

Exercise 1

About 18500 Hz on my good headphones, my laptop speakers don’t have the same range so not as sure. The loudest was about 2000 hz on headphones and 4000 hz on speaker. Not sure why. The 1600 1700 pair sound closer than the 400/500 pair. I’d put it to 2000, which sounds right to my ears.

Lab 9

Exercise 1

i)	f
ii)	1
iii)	0
iv)	3f
v)	1/3
vi)	0
vii)	(2k-1)f
viii)	1/(2k-1)
ix)	0

B) The waveform looks like a sine wave. As numHarmonics increases it approaches the shape of a square wave. It sounds more like a square wave. Honestly I noticed no difference. On my laptop the sound cracks but is very high and the visualisation is very close to a perfect square wave.

Exercise 3

22050 hz

10000hz

10000 hz

1000hz

I expect the frequency to increase, as the frequency is increasing, and it does

I expect it to increase as the frequency is increasing but it actually falls

I expect it to increase and decrease as that’s what happened earlier and that’s exactly what happens, and it sounds just like increasing from 0 – 44000 hz, increasing from 44100 to 66150, decreasing beyond that.