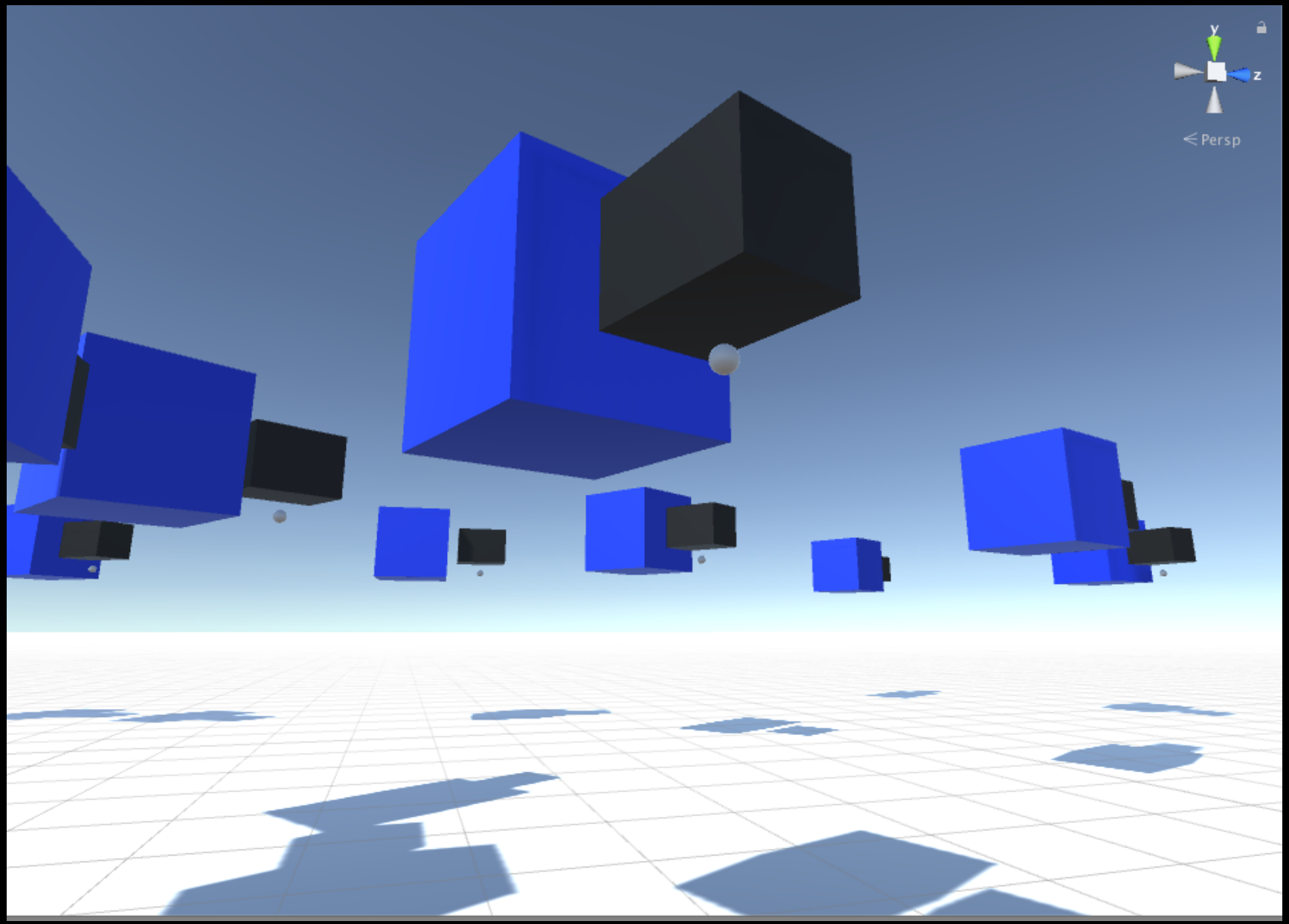
Camott-EcoSim



- Stuff has happend in this time!

Refresh

- “We want to create a simple ecosystem simulator in which an original set of “creatures” changes and evolves over time.”
- “We hope to see behaviours such as preditor-pray relationships emerge over time.”
- Creatures should start with a simple body and evolve...

What went wrong

- NN: Big and complex.
- RNN: Bigger and more complex.
- Unity: Has no support for arbitrary matrices, Libraries are hard, and in general not our way to program...
- Oh, time, where did you go?

What we learnt about Neural networks

- Computationally expensive: Get a GPU, use it. (Or at-least multithread it)
- Good for well-defined behaviours, (I tested it: it can do logic gates.)
- Why all the simulations are twitchy: Because neural networks don’t apply well (Atleast at this level of implmentation)
- It did kinda-work though! They learn to go forwards and hit stuff to eat it!
- See it: https://github.com/ttoocs/Cpsc565_Project

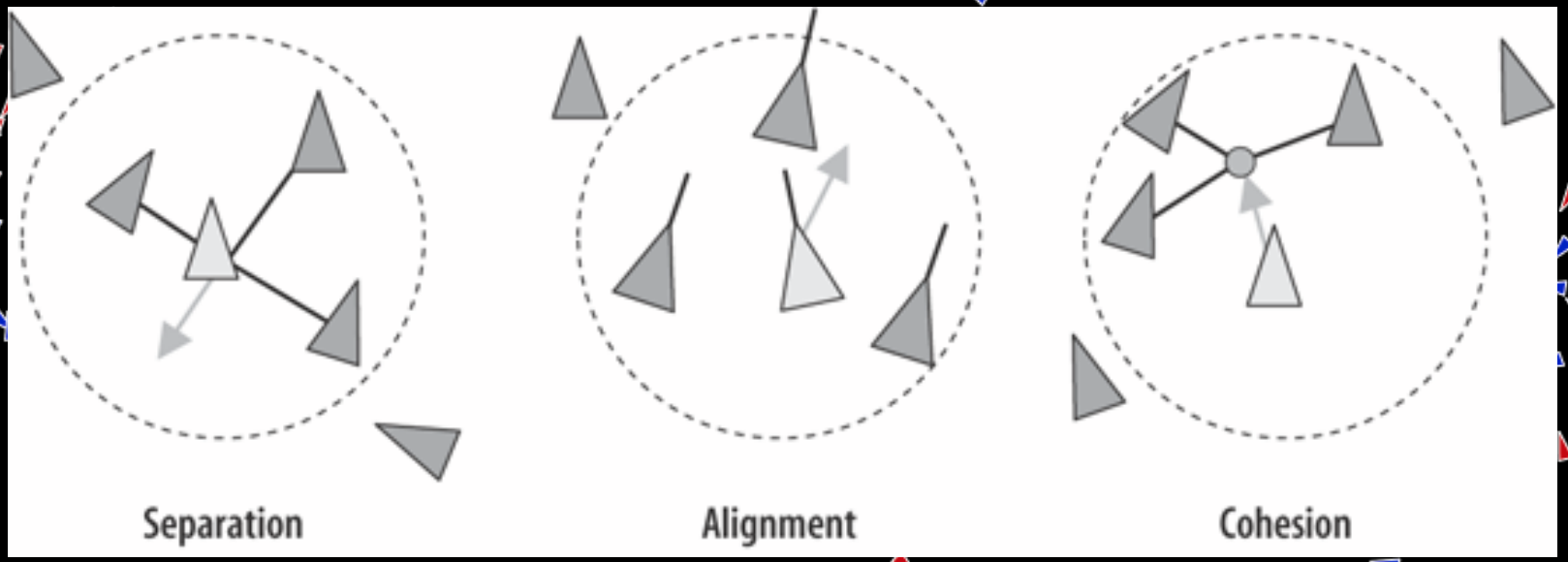
BOIDS!

Boids?

Yes Boids

A new hope BOIDS!

What’s a boid? Can I eat it?



Pseudo Code

```
PROCEDURE move_all_boids_to_new_positions()
        Vector v1, v2, v3
        Boid b
        FOR EACH BOID b
                v1 = rule1(b)
                v2 = rule2(b)
                v3 = rule3(b)
                b.velocity = b.velocity + v1 + v2 + v3
                b.position = b.position + b.velocity
        END
END PROCEDURE
```

Separation

```
PROCEDURE rule2(boid bJ)
        Vector c = 0;
        FOR EACH BOID b
                IF b != bJ THEN
                        IF |
b.position - bJ.position| < 100
THEN
                                c =
c - (b.position - bJ.position)
                        END IF
                END IF
        END
        RETURN c
END PROCEDURE
```

Alignment

```
PROCEDURE rule3(boid bJ)
          Vector pvJ
          FOR EACH BOID b
                    IF b != bJ THEN
                              pvJ = pvJ + b.velocity
                    END IF
          END
          pvJ = pvJ / N-1
          RETURN (pvJ - bJ.velocity)
  END PROCEDURE
```

Pseudo code taken without minimum modification from:
http://www.kfish.org/boids/pseudocode.html

All credit goes to the original author Conrad Parker.

Cohesion

```
PROCEDURE rule1(boid bJ)
        Vector pcJ
        FOR EACH BOID b
                IF b != bJ THEN
                        pcJ = pcJ + b.position
                END IF
        END
        pcJ = pcJ / N-1
        RETURN (pcJ - bJ.position)
END PROCEDUR
```

Our implementation

- OpenGL an C++, the much better alternatives to unity
- 1 main loop to rule them all!
- Object representation of flocks and boids
- Text file representation for flock coefficients
- Source code:

  https://github.com/Makogan/emergent2

Verlet Integration

$$\vec{x}(t+\Delta t) = \vec{x}(t) + \vec{v}(t)\,\Delta t + \frac{1}{2}\,\vec{a}(t)\Delta t^2,$$

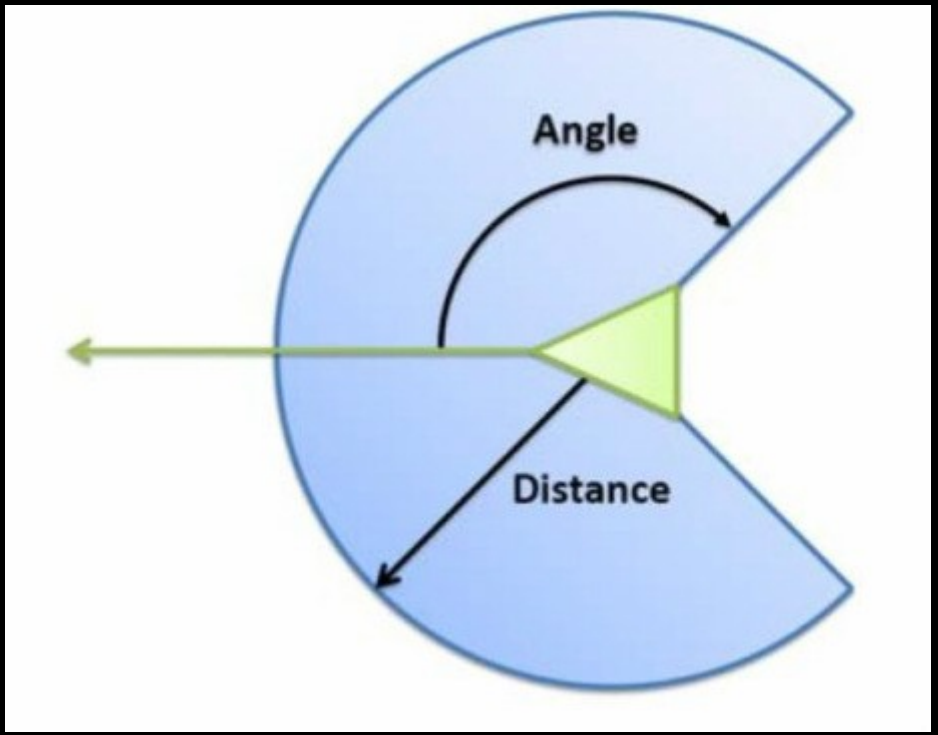
$$\vec{v}(t+\Delta t) = \vec{v}(t) + \frac{\vec{a}(t) + \vec{a}(t+\Delta t)}{2}\Delta t.$$

Fancy Maths!

When we put it all together

Click for magic:

https://www.youtube.com/watch?v=t3iOJO58kVQ&feature=youtu.be



The Future!

- Object avoidance
- Herding the flocks
- More flocks!
- Bigger flocks!
- Better BOIDS! (yes boids)
- Predators (maybe a use for our NN)