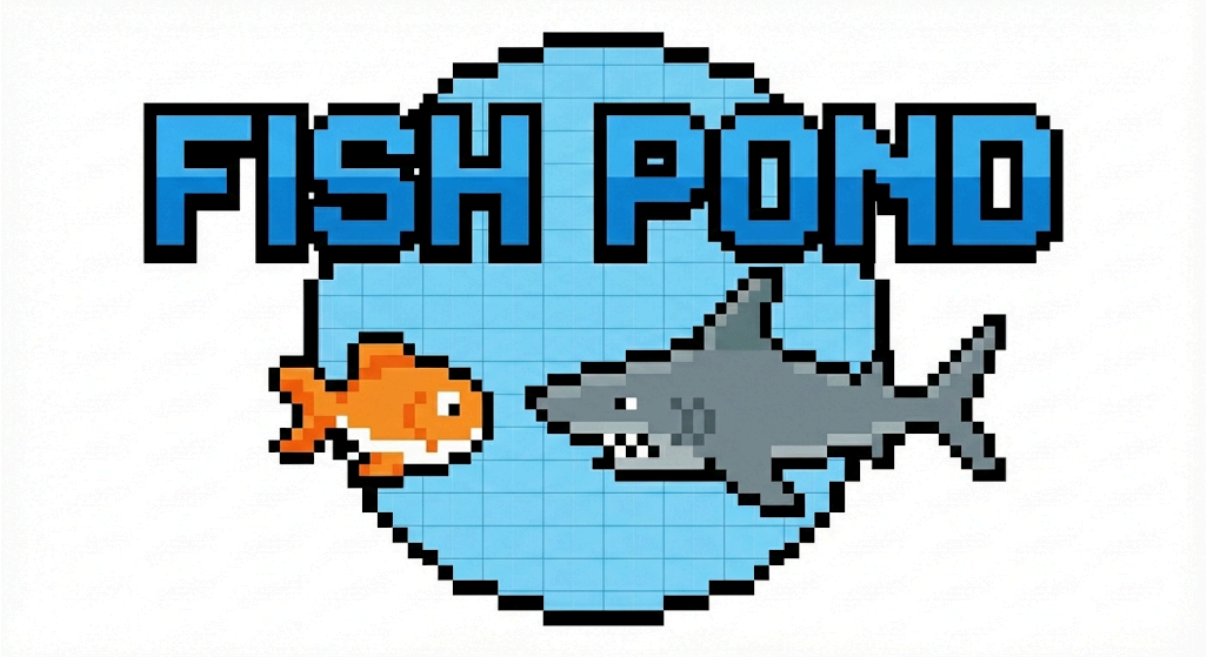
M3403



Vitthal Humbe

Vectors

A vector is a physical quantity defined by both magnitude (size / amount) and direction, unlike scalars which only have magnitude.

Examples : Force, Velocity, Displacement, etc.

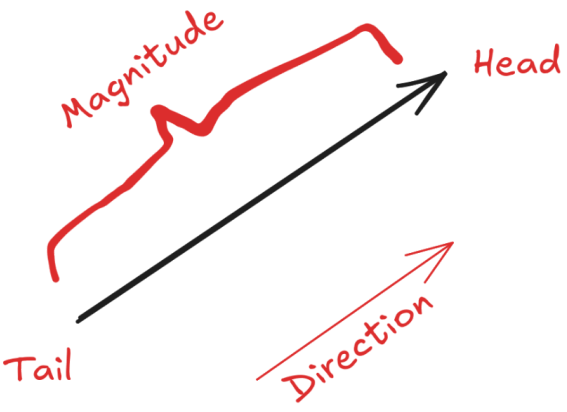
Scalar : 50 km/h (Speed)
Vector: 50 km/h North (Velocity)

Representation

Vector is represented as an arrow where it's length shows magnitude and the arrowhead shows direction.

symbolic way to represent : $\vec{v} = (x \quad y)$

In computer programming : we store vectors as a tuple of numbers, e.g. - v = [x, y]



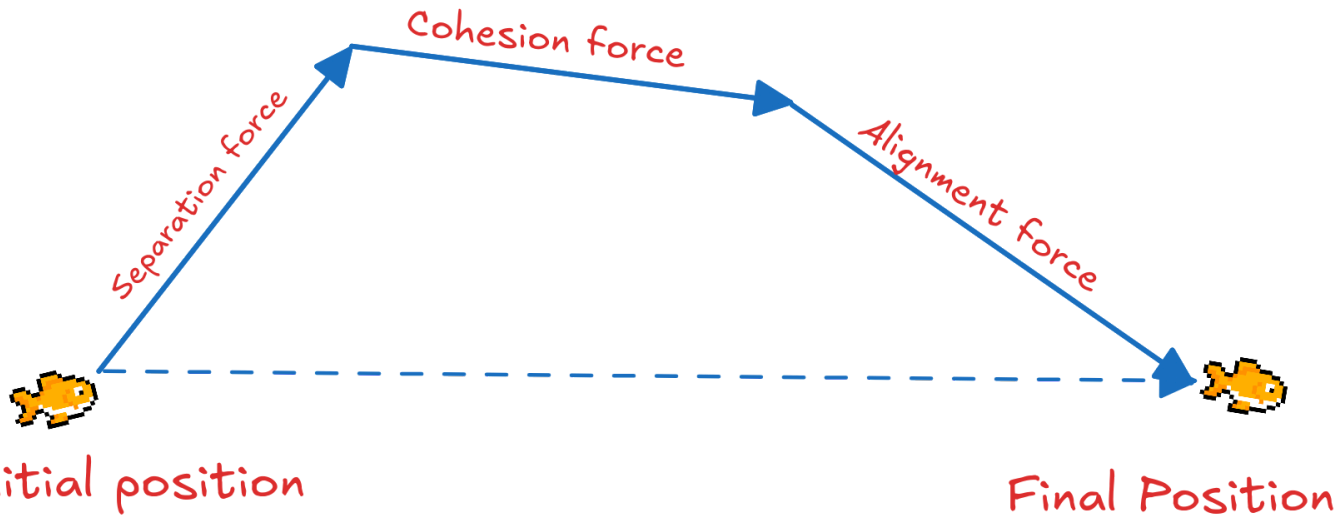
But Vitthal, Why we are learning vectors ?

> Because, vector is way to tell computer, how to move things. scalar like speed=10 is useless becuase the computer doesn't know where to go. (No direction). Vectors combine magnitude and direction into single representation like (x, y). without vectors, these fish whould just vibrate in place, or teleport; with vectors, they can flee, and swim smoothly across the screen.

Vector Arithmetic

The Principle of superposition of force states that the total force on any object in a system is the vector sum of all individual forces acting on it from all other objects.

Which simply means Vector Addition



MATH : We simple add the matching components together.
X adds to X and Y adds to Y.

$F_{sep} = [x1, y1]$

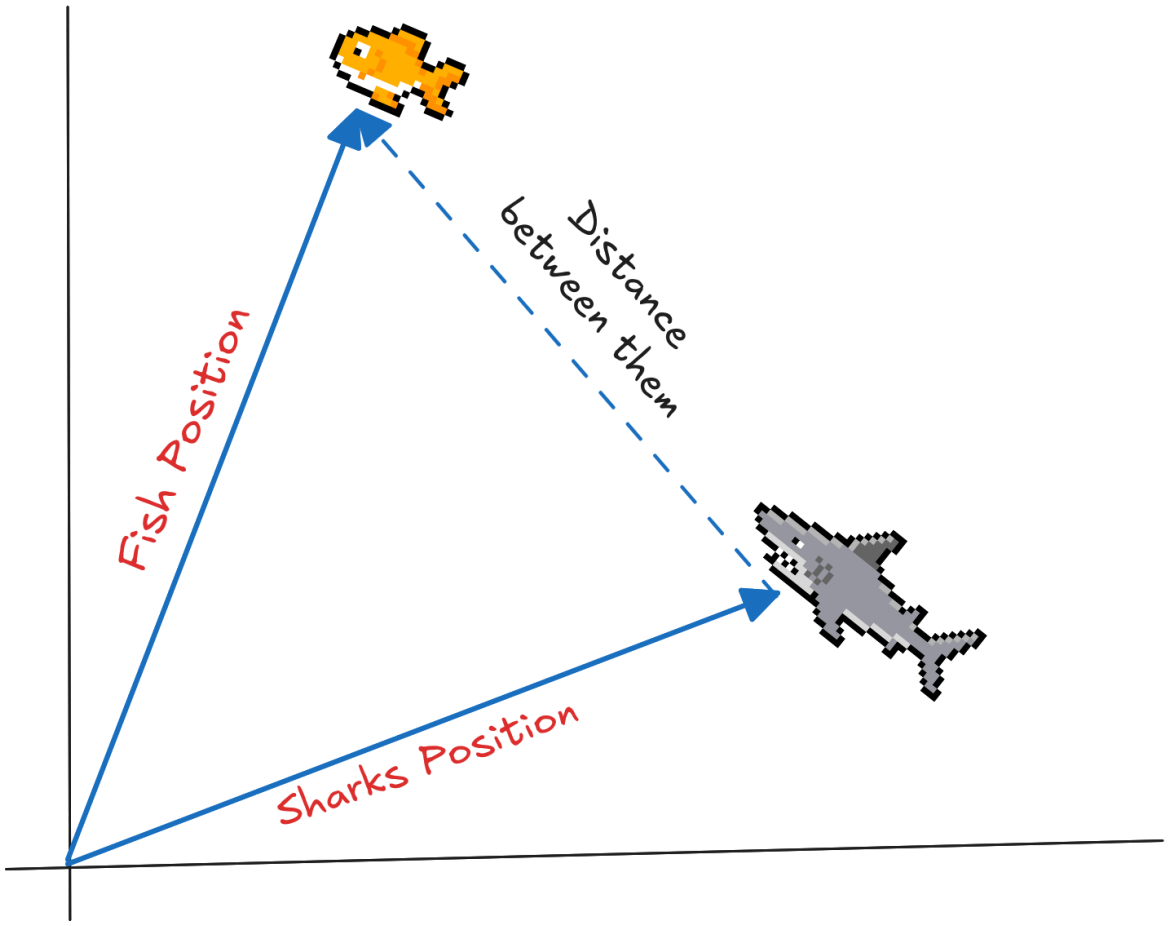
$F_{coh} = [x2, y2]$

$F_{algn} = [x3, y3]$

Hence, $F = [x1 + x2 + x3, y1 + y2 + y3]$

Vector Subtraction

Subtraction is used to find the vector that connects two points. The shark needs to know where the fish is relative to itself to chase it.



Math : Just like addition, we subtract components

$Pos_{fish} = [x1, y1]$

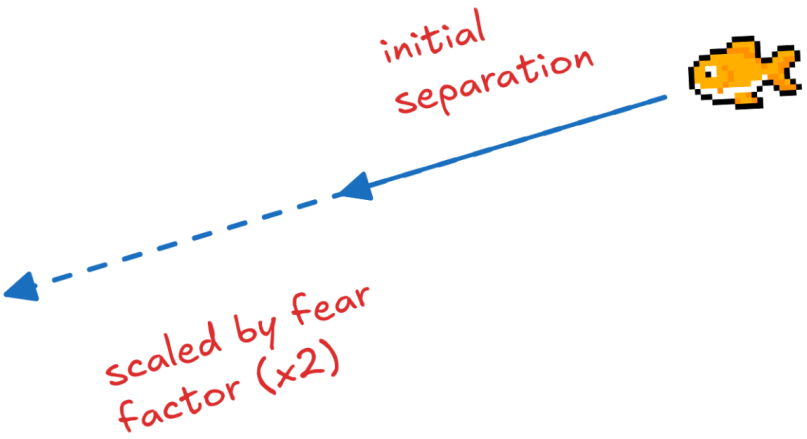
$Pos_{shark} = [x2, y2]$

Then, direction = $[x1 - x2, y1 - y2]$

Scalar multiplication

Sometimes we have correct direction... but the length (magnitude) is wrong. that's why we use the scalar multiplication, which enables us to stretch or shrink a vector without changing it's direction

Some fishes values the separation (or other vectors) more than others. We have to scale the force by a *GENE* value.



this fish has a Gene (fear factor) valued 2

which tends to it's separation from the group of fishes.

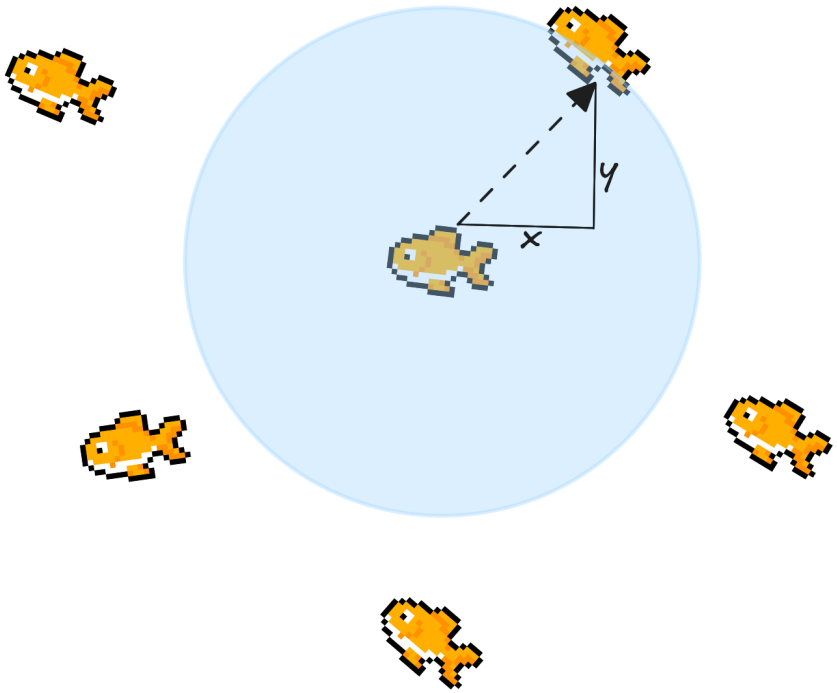
Vector Analysis

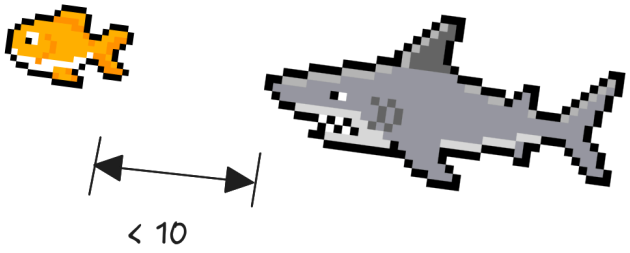
Vector analysis are the main concept of this simulation. while arithmetic used to moves the agents, Analysis helps shark/fishes measure the world to make decisions.

Magnitude

Magnitude of a vector is simply it's *LENGTH*. Since a 2D vector [x, y] forms right angled trangle with the axes, we can calculate its length using the pythogorean theorem.

to check distances in this simulation we use this concept. (*VISION* or Eating of shark)





When the distance between shark and fish is <10 then we conclude that collision has happened and the fish got eaten by shark.

Math : For vector v = [x, y]

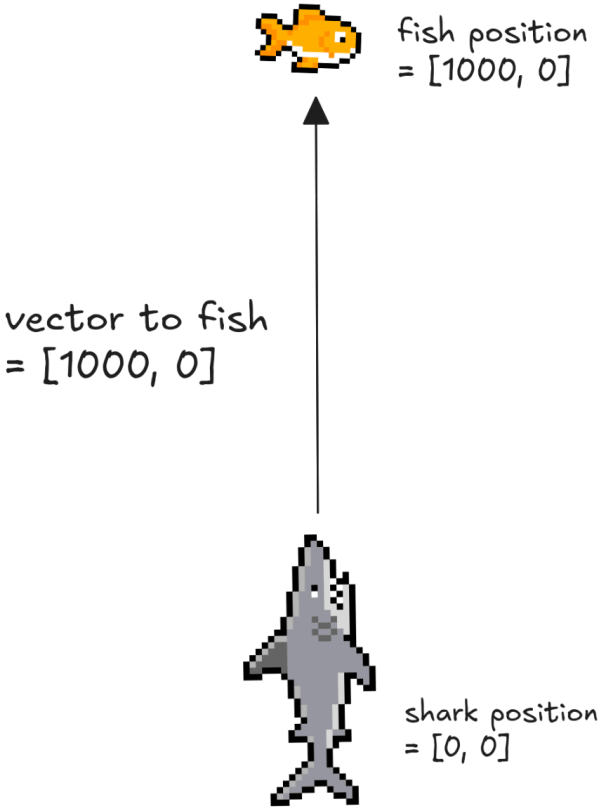
magnitude => $|\vec{v}| = \sqrt{x^2 + y^2}$

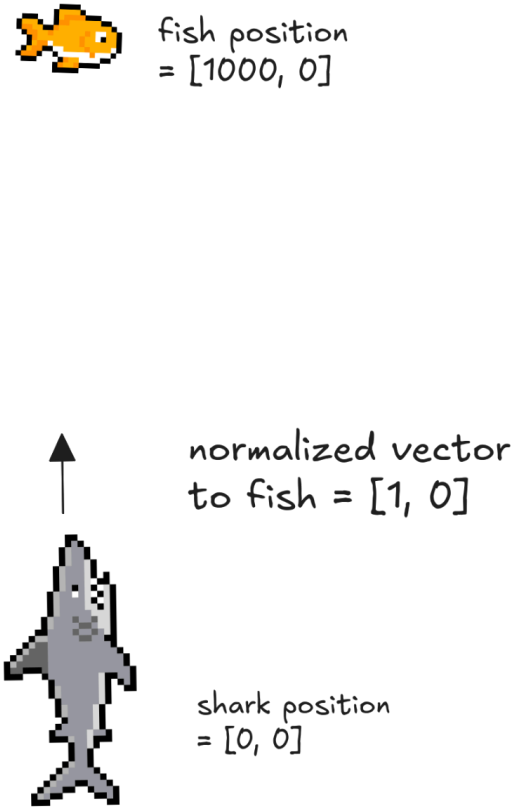
As we are in computer system, calculating square root is expensive... and might cause the simulation to lag. so we compare the squared values instead.

if Distance < 50, then Distance^2 < 2500.
if Distance < 10, then Distance^2 < 100.
Calculating $x^2 + y^2$ is much faster than finding square root.

Unit vector / normalization

Normalization is the process of taking a vector of any length and shrinking (or stretching) it so its length is exactly 1 unit. while keeping it's original direction.





If we add this vector to the shark's velocity directly then the shark will instantly jump 1000 pixels in one frame (that is teleportation)

fix :
1. normalize the vector.
2. scale it by the shark's max speed
3. now the sharks moves smoothly at max speed per frame towards the target.

Math : To normalize the vector we divide the vector by its own magnitude.

$$\vec{u} = \frac{v}{\| v \|}$$

Physics Basics

Now we have knowledge about the position, distances, and we can calculate the static numbers... now we will turn them into fuild motion so the simulation runs smoothly.

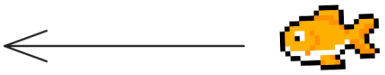
1. Hierarchy of Motion ?

   To simulate physics, we don't change position directly, we apply forces that trickle down this heirarchy.

   1. Position Where the object is right now (x, y)

fish position (100, 0)

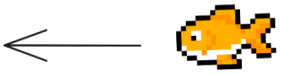
   2. Velocity : How fast (and where) the position changes every frame ? if V = [5, 0] the fish moves 5 pixels right.



Velocity vector = [5, 0]

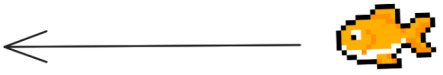
   3. Accelaration : How fast velocity changes ? when a shark scares a fish, it addes accelaration to it.

Before : fear factor is low



Velocity vector = [5, 0]

After : fear factor is high



Velocity vector = [10, 0]

The CYCLE : Integration Algorithm (Euler method)

This cycle will repeat every 1/60th second.

1. Reset Accelaration :

Forces are temporary. If we stop pushing, the accelaration stops. (Newton's 1st law).
So we must clear accilaration (A) to [0, 0] at the start of every frame.

2. Accumulate Forces :

Add every influence : such as Cohesion + Separation + Alignment + Fear

$$A_{total} = \sum \text{Forces}$$

Wait ... F = ma right ? then ?
we assume mass = 1 so that force eqauls Accelaration.

3. Velocity

simply add accelaration to velocity.

$$V_{new} = V_{old} + A_{total}$$

4. Limit Speed

Real physics doesn't have a speed limit (until light speed). but game physics must have one, or objects will fly off-screen instantly (High speed).

If || V || > MaxSpeed, scale it down

5. Update the position

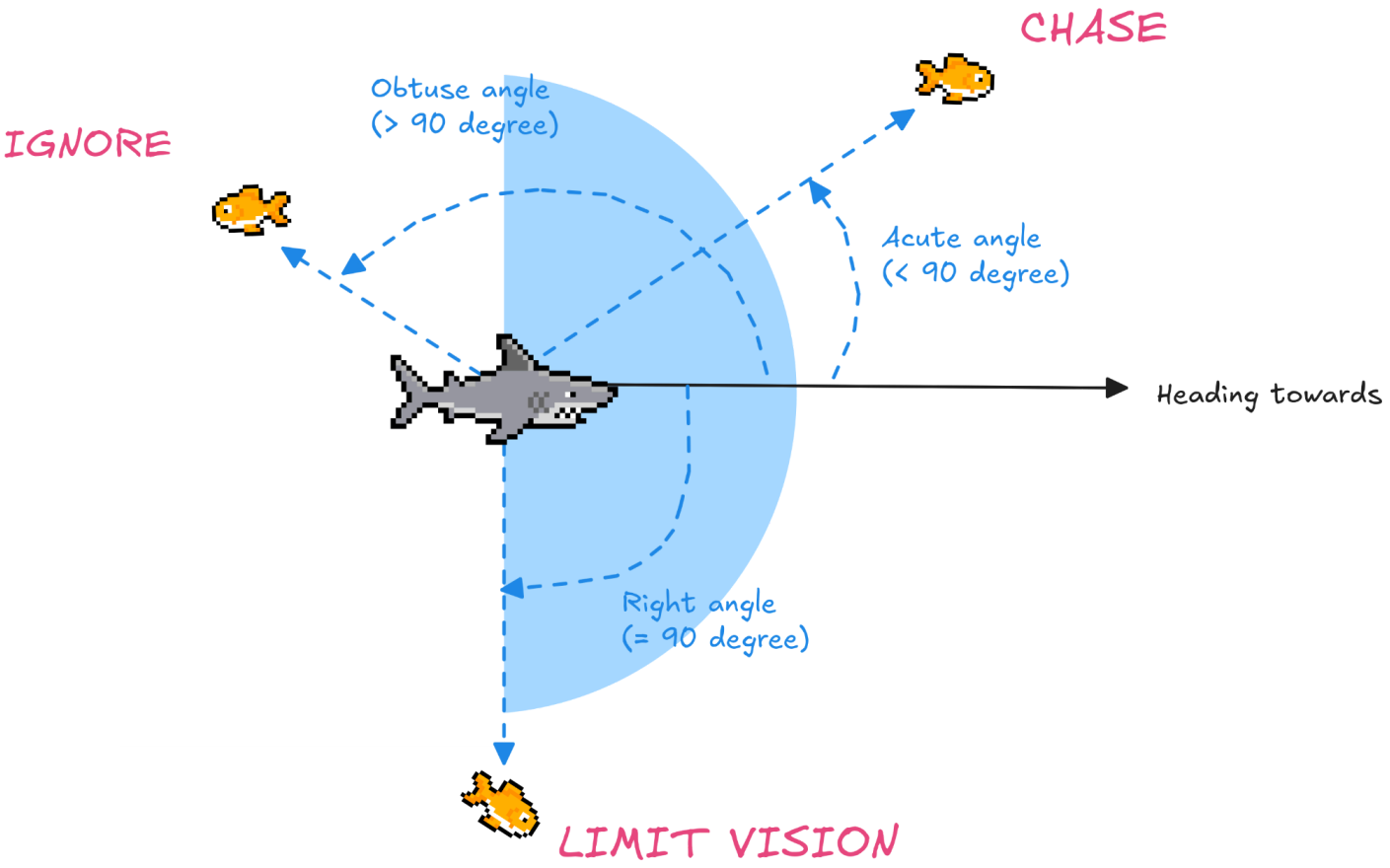
$$P_{new} = P_{old} + V_{new}$$

Trigonometry

Field of View of shark (VISION CONE)

The dot product is a scalar that tells us how much two vectors points in the same direction.

This is way to calculate vision cone for shark without doing expensive angle calculations.



MATH : $A \cdot B = (x1 \cdot x2) + (y1 \cdot y2)$

Result > 0 angle is acute, the fish is in front.... chase it.
Result < 0 angle is obtuse, the fish is behind ... ignore it.
Result = 0 angle is excatly 90 degree, we can limit the vision of shark to remove this ambiguity.

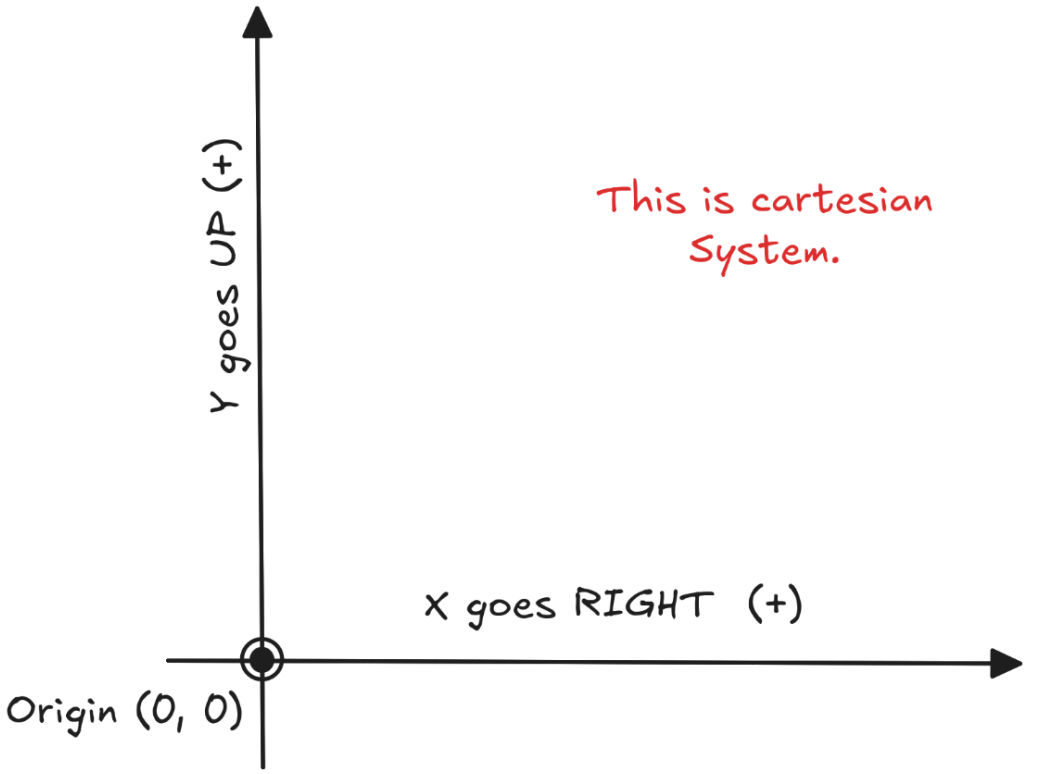
Atan2:

this physics engine usese vecotrs to move, but graphics engine (like pygrame) needs an angle (in degrees) to rotate the image. so we need to convert (x, y) -> Angle.

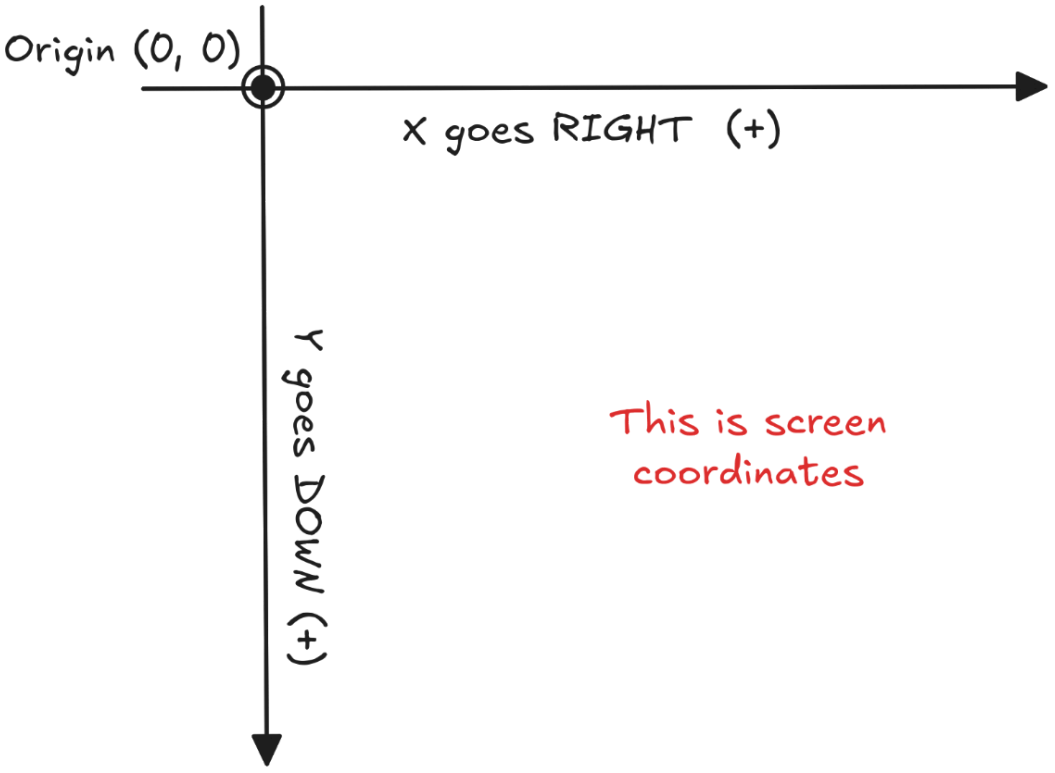
atan2(y, x) is a special function that handles all quadrants and zeros correctly (unlike tan function), returning a full 360 degrees (or 2π Radians) angle.

Coordinate System

Cartesian VS Screen Coordinates



This is cartesian System.



This is screen coordinates

Rules applied in the system

1. Separation

   if neighbor gets too close, the fish feels a repulsive force pushing it away. This prevents groups from collapsing into single black dot.

   Calculate vector point from neighbors to myself. (my_pos - their_pos)

   Normalize it and then scale it by how close are they.

2. Alignment

   Fishes (in nature) move with each other. This makes group look synchronized.

   Get the velocity of every neighbor within vision
   add them all up to gate average direction, then swim towards that average.

3. Cohesion

   Without cohesion fish would drift apart and be eaten by the shark alone. It pulls the fish towards the Center of mass of the group.

   Calculate the average position of all neighbors. then create a vector pointing from yourself to center point. and swim along that vector.

4. Fear

   if predator enters the vision radius then the fish ignores its friends and swims directly away from the danger at maximum speed.

   Calculate vector from shark to me, normalize it, scale it to max Speed. we can also use inverse square weight... the closer the shark the stronger the repulsive force becomes.

Evolution

Survival of the Fittest

Now the actual magic happens, previously we used the hardcoded values (e.g. Max_Speed = 4.0) we use a Genetic Algorithm (GA) to let the computer evolve the solution.

Cycle of Life

We treat our simulation as an optimization problem where "Survival Time" is the score.

1. Initialize : spawn 50 fish with completely random state(DNA).
2. Simulate: Run the physics loop. The shark hunts them.
3. Evaluate : When a fish dies, record how long is lived (Fitness)
4. Selection: when all (or most) are dead, pick the best parent.
5. Reproduction: Mix the parent's DNA to create new generation.
6. Mutation : randomly tweak values to discover new strategies.

the next generation should be slightly better at surviving.

Genome (DNA)

In biology, DNA is a complex molecule. In my code (or in GA), a "Gene" is just a floats, and a "DNA" is a Numpy Array.

{
max speed,
vision radius,
separation weight,
alignment weight,
cohesion weight,
fear weight
}

The Fitness function

now, we need a formula to rank the fishes !!, it's not only about the surviving, it's about surviving with style. the following formula is derived by me by keeping different factors in mind -

$$Fitness = \frac{T_{alive}}{T_{max}} + (W_1 \cdot C_{bonus}) + (W_2 \cdot D_{danger})$$

Time Alive (T): normalized by max duration.
Cohesion Bonus (C): reward for staying near friends.
Danger Distance (D): Reward for maintaining a safe buffer from the shark.

The Selection Strategy

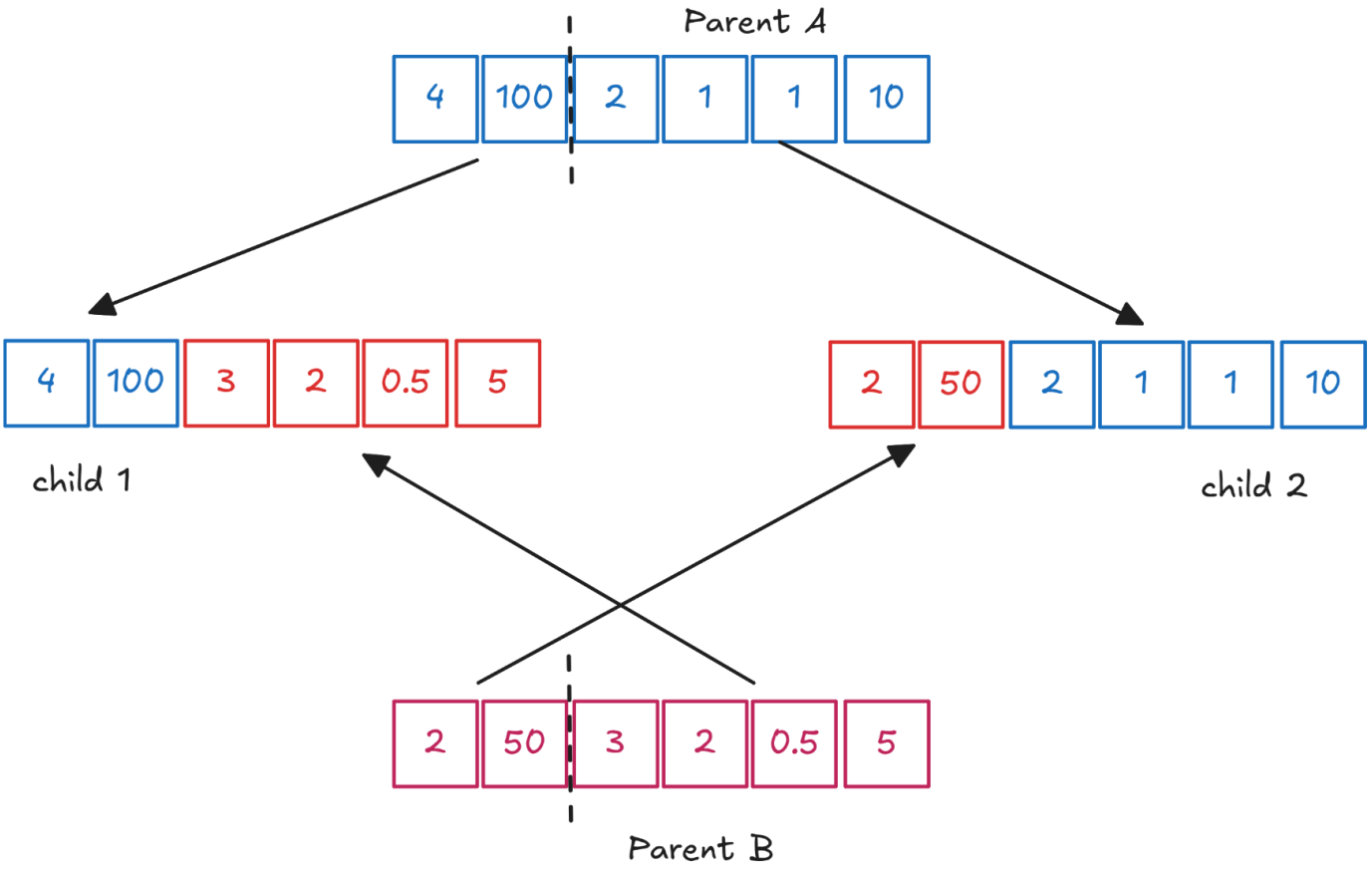
There are many selection strategies, so I came up with hybrid strategy to ensure the qaulity while keeping diversity.

1. Elitism : top 2 fishes from previous generation are copied directly (without change) to the new generation.

2. Tournament Selection : To pick the rest of the parents, we do this :
    1. pick 3 random fish from old generation.
    2. compare their fitness scores.
    3. the winner becomes parent A.
    4. repate to find parent B.

Reproduction (Crossover & Mutation)

1. Crossover pick a random split point in a DNA array, take the first part from parent A, and take the second part from Parent B, combine them to form the child.



2. Mutation After crossover, we apply some gaussian noise to one or none gene to introduce new traits that neither parents had.

New Value = old value + Random (-0.1, 0.1)

note, I applied max(0, value) to check the values should not get negatives, like -ve radius ? what the hell ?