OLD MODEL

DNA
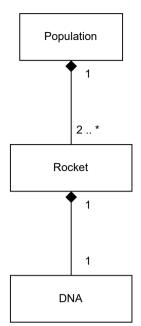
+ genes: Array

+ maxForce: int

+ constructor(): void

+ crossover(DNA): DNA

+ mutate(float): void

Rocket

+ acceleration: Vector

+ velocity: Vector

+ position: Vector

+ r: int

+ fitness: float

+ dna: DNA

+ geneCounter: int

+ constructor(int, int, DNA): void

+ calculateFitness(): void

+ run(): void

+ applyForce(float): void

+ update(): void

+ show(): void

Population

+ mutationRate: float

+ population: Array

+ matingPool: Array

+ generations: int

+ constructor(float, int): void

+ live(): void

+ fitness(): void

+ selection(): void

+ reproduction(): void

+ weightedSelection(): DNA

Population
◆ 1
2 .. *
Rocket
◆ 1
1
DNA

- -

NEW MODEL

Rocket

+ acceleration: Vector

+ velocity: Vector

+ position: Vector

+ r: int

+ fitness: float

+ bestDistance: float?

+ finishCounter: int

+ hasHitObstacle: bool

+ hasHitTarget: bool

+ dna: DNA

+ geneCounter: int

+ constructor(int, int, DNA): void

+ calculateFitness(): void

+ run(Obstacle Array): void

+ applyForce(float): void

+ update(): void

+ show(): void

+ checkObstacles(Obstacle Array): void

+ checkTarget(): void

DNA

+ genes: Array

+ maxForce: int

+ constructor(): void

+ crossover(DNA): DNA

+ mutate(float): void

Population

+ mutationRate: float

+ population: Array

+ matingPool: Array

+ generations: int

+ constructor(float, int): void

+ live(Obstable Array): void

+ fitness(): void

+ selection(): void

+ reproduction(): void

+ weightedSelection(): DNA

+ targetReached(): bool

Obstacle

+ position: Array

+ width: int/float

+ height: int/float

+ constructor(int, int. int/float, int/float): void

+ contains(Array): bool

+ show():

Population
◆ 1
2 .. *
Rocket
◆ 1
1
DNA

Process of creation of GA and object life cycle

Process of creation of GA (Setup)

Create class for objects

Create Phenotype (Create class fields and methods)

Create DNA class

Encode Phenotype (create Genotype [DNA] and assign values)

Create fitness function

Create crossover function

Create mutation Function

Determine the rate at which it "sets into" the aproximate answer

Determine how new genes are created

Determine the mutation rare.

Determine how fitness will be calculated

Many strategies available.

Set appropriate new random values

Generate an initial population with random genes

object life cycle
(Draw)

Let every object from the population live for N amount of time

Repeat the simulation until you reach a satisfactory result

Calculate the fitness scores of every object in the population

Whichever fitness strategy path is chosen, it remains the same for the whole simulation

Gives decimal values for fitness scores

Use linear formula

Use quadradic or exponential formula

gives whole value for fitness scores (unless scaled down)

Scale value to whole number

Use weighed selection with current population to choose 2 parents

add to mating pool this amount of copies

pick 2 random parents

(invariant) 2 parents have been chosen

Make a child. Use Crossover algorithm.

Mutate the child's genes (if opportunity occurs)

Build a new population pool and replace old one
OR
replace the population pool in the same container

Repeat the simulation
to refine the result

End the simulation