MACHINE LEARNING

1. Genetic algorithms

2. Perceptron

3. NN

**Genetic algorithms:**

**1. Camouflage: (camouflage folder, camo scene)**

**Introduction**

Breed bodies with specific color by genetic algorithm, which calculate best bodies by time to die value.

**Population manager**

Create new population every elapsed time and breed it by time to die

**DNA script**

Stored RGB, size, time to die and methods by click on body

**2. Runners: (walker folder, WalkStraight scene)**

**Introduction**

Breed characters with specific directions by genetic algorithm. Calculate the best directions by biggest alive time values

**PManager**

Instantiate characters with brains and set best directions values via the biggest time alive values.

**Brain**

Set properties like direction move, crouch, jump and check if character is alive and how long.

**Dna**

Mix set of genes values, set values of every gene, store gene with length and count.

**3. Maze walker (GA Walekr/Maze folder, maze walker scene)**

**Introduction**

Breed character directions with brains via distance travelled

**MazePM**

Instantiate characters with brains and set best directions values via the biggest distance travelled and alive values.

**Brain maze**

Check walls, check dead, check genes and change direction if character see wall via ray cast

**Generate maze**

Generate specific maze for characters

**4. Flappy birds (birds folder, training room scene)**

**Introduction**

Like maze walker, but with another properties. Breed birds with biggest travel distance

**Perceptron**

**Simple Perceptron (perceptron1 folder, Perceptron scene)**

**Introduction**

Calculate simple perceptron with 1 layer for understanding how it works

**Void Perceptron ( Void network folder, Dodge ball scene)**

**Introduction**

Teach character to avoid the ball via logistic regression and perceptron

**Throw script**

Throw out the ball or the cube to character. Need to use 1,2,3,4 buttons.

Character must avoid 1st button.

**Void Perceptron**

Use simple perceptron.

Use space button to re initialize weights. S button to save, l to load data from file.

Firstly, initialize weights, bias via random values.

Train our set of weights. Update weights reset weights and bias via error

Error calculate like predicted output – actual output.

Actual output calculate via activation function which take logistic regression function.

Function looks like Sum += bias + weight[i]\*input[i]

So, if result = 0, our character use crouch and avoid the ball

**Goodly Perceptron**

**BrainPerceptron**

Set parameters of ANN, count of iterations, train and output

**Layer**

Count of neurons in layer, list of this,

**Neuron**

Count of entered neurons, list of weights, output and error gradient

**ANN(artificial neural network)**

Constructor get: count of neurons which come in the start, count of outputs, count of layers, how many neurons in 1 layer, learning rate.

**Go (inputs, desiredOutput)**

This method go through the all of the layers every neuron and calculate new values .

1stly if layer not 1, puts outputs from previous layer to inputs in current layer and clear outputs.

2nd loop though the layers, neurons, inputs in every neuron and calculate the output. After that update weights.

**UpdateWeights(output, desired output)**

It’s like back propagation.

Check error function via output and desired output and calculate new weight value

**Graph**

**GraphPerceptron**

Create a graph via graphic and perceptron.

Create a UI

Train > Initialize weights > Update weights

Show on UI the result of training

**SimpleGraph**

Paint the UI

**PongNN**

**BrainPong**

Calculate the positions for NN and set in ann class

ANN, Layer, Neuron classes is the same

**//TODO Pong NN**