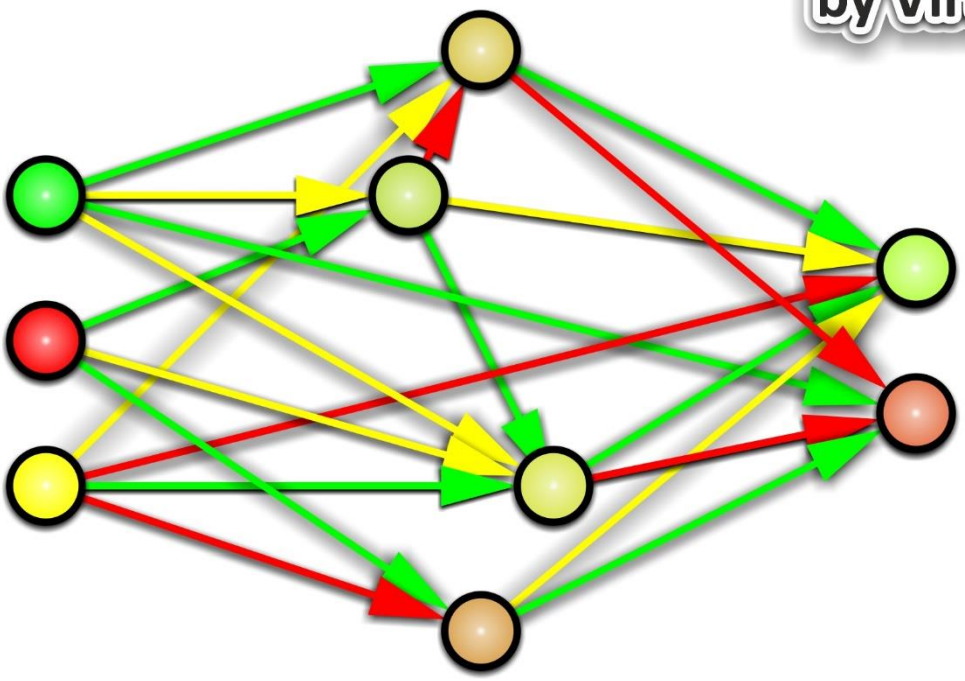
ARTIFICIAL NEURAL NETWORK
ANN & NEAT
by VirtualSUN



Набір скриптів для створення та навчання власної штучної нейронної мережі.

Створення власної штучної нейронної мережі (ШНМ) не є великою проблемою. А от навчити ШНМ виконувати ті чи інші дії та завдання – це вже справжній виклик. Для того, щоб полегшити Вам завдання створення та навчання ШНМ Я написав ряд скриптів, які все зроблять за вас. Єдине, що від вас потребується - це правильно «пояснити» ШНМ, що вона має вивчити.

2019

ЗМІСТ

Загальний опис скриптів.

ANN.cs – не MonoBehaviour скрипт штучної нейронної мережі (ШНМ). Цей скрипт автоматично створить ШНМ за вашими параметрами, вирішить поставлене завдання (якщо пройде «навчання»), збереже та завантажить вашу ШНМ.

ANNNode.cs - не MonoBehaviour скрипт який містить основну інформацію вузла (нейрону) ШНМ.

ANNConnection.cs - не MonoBehaviour скрипт який містить основну інформацію з'єднань між вузлами (нейронами) ШНМ.

ANNInterface.cs – цей скрипт можна використовувати для полегшення створення, візуалізації, збереження та завантаження ШНМ під час періоду навчання. Це інтерфейс для ШНМ у ігровому режимі.

ANNLearnByNEAT.cs – не MonoBehaviour скрипт для навчання ШНМ методом нейроеволюції доповнюючих топологій. За допомогою гнучких налаштувань Ви можете швидко і легко завершити навчання ШНМ.

ANNLearnByNEATInterface.cs – це скрипт інтерфейсу для навчання ШНМ методом нейроеволюції доповнюючих топологій у ігровому режимі.

InterfaceGUI.cs – не MonoBehaviour скрипт (static) для скриптів інтерфейсів.

DrawANNWeight.cs – не MonoBehaviour скрипт (static) для візуалізації вагових зв'язків ШНМ. Використовується скриптом ANNInterface.cs.

Formulas.cs – не MonoBehaviour скрипт (static) для полегшення вирішення циклічних чи подібних завдань.

Детальний опис основних скриптів.

public class ANN – не MonoBehaviour скрипт штучної нейронної мережі (ШНМ).

Основні змінні скрипту ANN.cs	
public float AFS	Розмір функції активації.
public bool B	Якщо «вірно», то у кожному вузлі (нейроні) активується додаткова вага зміщення. Не діє на вхідні нейрони.
public bool AFWM	Якщо «вірно», то всі нейрони сприймають значення від -1 до 1. Це стосується також вхідних та вихідних нейронів. Якщо «невірно» - то від 0 до 1.
public float[] Input	Вхідний шар ШНМ . Розмір масиву вказує кількість вхідних значень.
public float[] Output	Вихідний шар ШНМ . Розмір масиву вказує кількість вихідних значень.
public Dictionary<int, ANNNode> Node	Словник вузлів (нейронів). Містить кожен вузол ШНМ.
public Dictionary<int, ANNConnection> Connection	Словник зв'язків між нейронами. Містить кожен зв'язок ШНМ.

Команда скрипту	**Змінні команди**	**Пояснення**
public void Create		Створення ШНМ за допомогою параметрів.
	float AFS	Розмір функції активації.
	bool B	Якщо «вірно», то у кожному вузлі (нейроні) активується додаткова вага зміщення. Не діє на вхідні нейрони.
	bool AFWM	Якщо «вірно», то всі нейрони сприймають значення від -1 до 1. Це стосується також вхідних та вихідних нейронів. Якщо «невірно» - то від 0 до 1.
	int NumberOfInputs	Кількість вхідних нейронів.
	int NumbersOfOutputs	Кількість вихідних нейронів.
public void Load		Завантаження ШНМ з файлу. Підтримує файли збереження “ANN Perceptron”.
	string ANNFile	Ім’я файлу для завантаження.
public void Solution		Рішення ШНМ.
private float ActivationFunction		Функція активації нейронів.
	float Sum	Сума всіх значень нейронів помножених на їх вагу.
private void CreatingNeurons		Створення нейронів та їх зв’язків між собою.
	StreamReader SR	Вказаний потік з файлу завантаження. Використовуйте "null", якщо файл не використовується.
public void Save		Збереження параметрів ШНМ у файл.
	string ANNFile	Ім’я файлу для завантаження.
public void FixConnections()		Сортування нумерації з’єднань по порядку.
public void NumberingCorrection()		Сортування нумерації вузлів по порядку.

public class ANNNode – не MonoBehaviour скрипт який містить основну інформацію вузла (нейрону) ШНМ.

Основні змінні скрипту ANNNode.cs	
public float Neuron	Значення нейрону (зв’язка).
public float Bias	Значення зміщення нейрону.
public ArrayList ConnectionIn	Список вхідних номерів з'єднань.
public int Fullness	Повнота нейронних зв'язків при вирішенні.
public Vector2 Position	Позиція при візуалізації.

Команда скрипту	**Змінні команди**	**Пояснення**
public ANNNode		Введення інформації про нейрон.
	float Neuron	Значення нейрона (вузла).
	float Bias	Значення зміщення нейрону.
	ArrayList ConnectionIn	Список вхідних номерів з'єднань.
	int Fullness	Повнота нейронних зв'язків при вирішенні.
	Vector2 Position	Позиція при візуалізації.
	string Info	Текстовий вигляд нейрона (вузла).
public override string ToString		Перекладає інформацію вузла у текстовий вигляд.

public class ANNConnection – не MonoBehaviour скрипт, який містить основну інформацію про з'єднання ШНМ.

Основні змінні скрипту ANNConnection.cs	
public int In	Номер вхідного нейрону.
public int Out	Номер вихідного нейрону.
public float Weight	Вага зв’язка.
public bool Enable	Стан зв’язка.

Команда скрипту	**Змінні команди**	**Пояснення**
public void ANNConnection		Введення інформації про зв’язок.
	public int In	Номер вхідного нейрону.
	public int Out	Номер вихідного нейрону.
	public float Weight	Вага зв’язка.
	public bool Enable	Стан зв’язка.
	string Info	Текстовий вигляд зв’язка.
public override string ToString()		Перекладає інформацію зв’язка у текстовий вигляд.

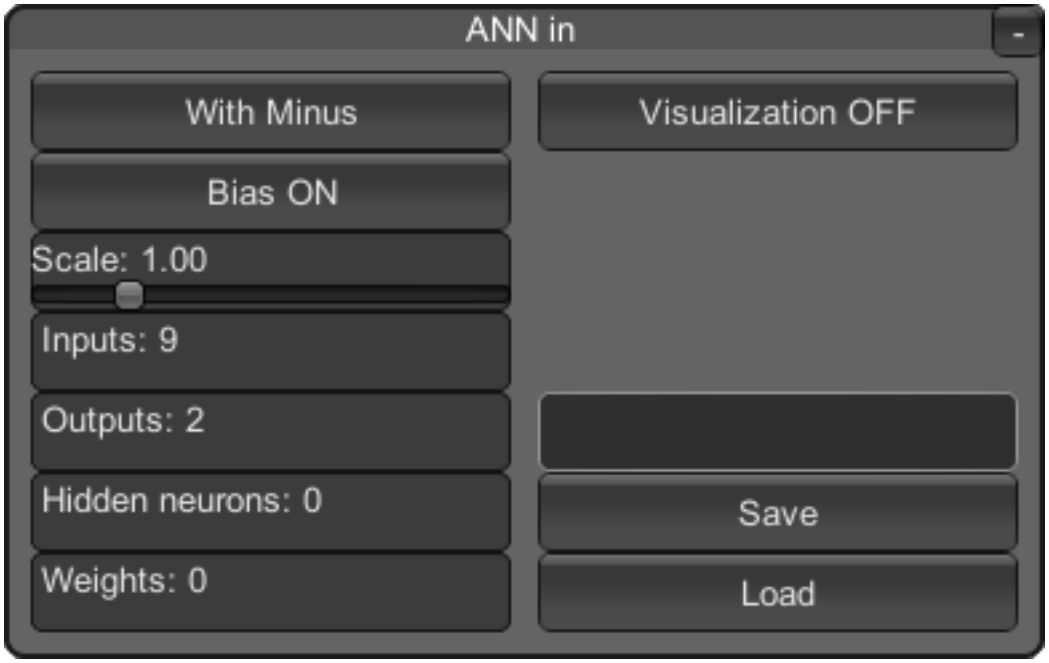
public class ANNLearnByNEAT – не MonoBehaviour скрипт для навчання ШНМ.

Основні змінні скрипту ANNLearnByNEAT.cs	
public ANN Ann	ШНМ яка навчається.
public int AmountOfChildren	Кількість «дітей» у кожному поколінні.
public int ChildrenInWav	Кількість "дітей" у хвилі кожного покоління.
public bool ChildrenByWave	Якщо "true" - хвилями. Якщо "false" - використовувати максимум за один раз.
public int BestGeneration	Краще покоління на даний момент.
public int Generation	Загальна кількість поколінь.
public int ChildrenInGeneration	Кількість "дітей" в даному поколінні.
public bool Cross	Дозвіл на "схрещування" двох найкращих "дітей".
public bool PerceptronStart	Якщо "true", то в першому поколінні дітей всі вхідні нейрони будуть з'єднані з вихідними нейронами.
public bool Autosave	Авто-збереження при навчанні.
public int AutosaveStep	Крок авто збереження.
public string AutosaveName	Початкове ім'я файлу для авто-збереження.
public bool IgnoreCollision	Якщо "true", то буде проведено ігнорування зіткнення дітей.
public float BestLongevity	Краще «довголіття» на даний момент.
public float ChildrenDifference	Різниця вагових зв’язків між поколіннями.
public float Chance	Шанс вибрати поточне гірше покоління. Він змінюється за рахунок параметру public float ChanceCoefficient з кожним новим поколінням.
public float ChanceCoefficient	Коефіцієнт впливу на шанс випадковості вибору теперішнього гіршого покоління. Впливає при умові, що не дорівнює нулю.
public float MutationAddNeuron	Пропорція використання мутації "додати нейрон".
public float MutationAddWeight	Пропорція використання мутації "додати вагу".
public float MutationChangeOneWeight	Пропорція використання мутації "змінити одну вагу".
public float MutationChangeWeights	Пропорція використання мутації "змінити всі ваги".
public float MutationChangeOneBias	Пропорція використання мутації "змінити одне зміщення".
public float MutationChangeBias	Пропорція використання мутації "змінити всі зміщення".
public float MutationSum	Сума пропорційних значень мутації.
public int AddingWeightsCount	Кількість одночасного додавання зв'язків між нейронами у кожної "дитини".
public float ChangeWeightSign	Шанс на зміну знаку зв'язка.

Команда скрипту	**Змінні команди**	**Пояснення**
public void StudentData		Збір даних для навчання.
	GameObject Student	Головний ігровий об’єкт (GameObject) який має вчитися.
	Object HereIsANN	Скрипт який містить ШНМ.
	string ANNName	Ім’я змінної (ANN) яким названо ШНМ у скрипті який містить ШНМ (HereIsANN).
	Object StudentControls	Скрипт керування головного ігрового об’єкту.
	string StudentCrash	Ім’я змінної (bool) яким названо причина «аварії» ігрового об’єкту у скрипті керування (StudentControls).
	string StudentLife	Ім’я змінної (float) яким названо «довголіття» ігрового об’єкта у скрипті керування (StudentControls).
public void Learn		Навчання ШНМ.
public void StopLearn		Негайна зупинка навчання з передачею інформації вагових зв’язків кращої ШНМ з кращого покоління на ШНМ яка навчається.
public void Reset		Скинути інформацію про навчання.

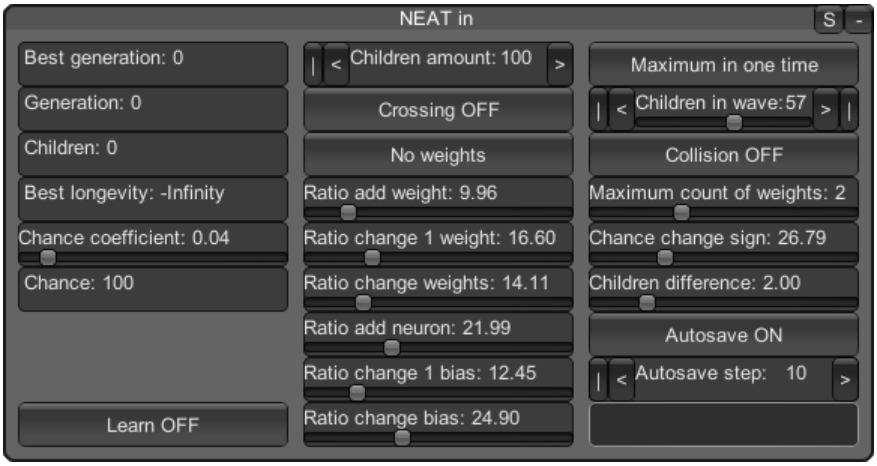
Опис допоміжних інтерфейсів.

ANNInterface.cs - цей скрипт можна використовувати для полегшення створення, збереження та завантаження ШНМ під час періоду навчання. Це інтерфейс для ШНМ у ігровому режимі. Керує параметрами скрипту **ANN.cs**.



Without Minus With Minus	Якщо «With Minus», то всі нейрони сприймають значення від -1 до 1. Це стосується також вхідних та вихідних нейронів. Якщо «Without Minus» - то від 0 до 1. Керує параметром public bool AFWM.
Bias OFF Bias ON	Якщо «Bias ON», то у кожному вузлі (нейроні) активується додаткова вага зміщення. Не діє на вхідні нейрони. Якщо «Bias OFF», то у нейронах зміщення не буде. Керує параметром public bool B.
Scale	Розмір функції активації. Керує параметром public float AFS.
Inputs	Вказує кількість нейронів у вхідному шарі.
Outputs	Вказує кількість нейронів у вихідному шарі.
Hidden neurons	Загальна кількість прихованих нейронів.
Weights	Загальна кількість вагових зв'язків.
Visualization OFF Visualization ON	Якщо «Visualization ON» - демонструє вигляд ШНМ з заданими параметрами. Якщо «Visualization OFF» - не демонструє.
GUI.TextField	Ім’я файлу для збереження або завантаження ШНМ.
Save	Зберігає параметри та всі вагові зв’язки ШНМ у файл, якщо вказано ім’я файлу. Керує командою public void Save.
Load	Завантажує параметри та всі вагові зв’язки ШНМ з файлу, якщо вказано ім’я файлу. Керує командою public void Load.

ANNLearnByNEATInterface.cs - інтерфейс для навчання ШНМ методом нейроеволюції доповнюючих топологій. Керує параметрами скрипту **ANNLearnByNEAT.cs**.



Best Generation	Показує номер кращого покоління на даний момент. Отримує данні з параметра public int BestGeneration.
Generation	Показує яке покоління на даний момент. Отримує данні з параметру public int Generation.
Children	Показує кількість «дітей» у поточному поколінні. Отримує данні з параметру public int ChildrenInGeneration.
Best longevity	Показує краще «довголіття» на даний момент. Отримує данні з параметру public float BestLongevity.
Chance coefficient	Коефіцієнт впливу на шанс випадковості вибору теперішнього гіршого покоління. Впливає при умові, що не дорівнює 0. Керує параметром public float ChanceCoefficient.
Chance	Показує шанс вибору поточне гірше покоління. Він змінюється з кожним новим поколінням. Отримує данні з параметру public float Chance.
Learn OFF Learn ON	Якщо «Learn ON», то починає навчання ШНМ. Керує командою public void Learn. Якщо «Learn OFF» і проводилось навчання, то зупиняє навчання. Використовує команду public void StopLearn.
Children amount	Кількість «дітей» у кожному поколінні. Керує параметром public int AmountOfChildren.
Crossing OFF Crossing ON	Увімкнути / вимкнути "схрещування" двох найкращих "дітей". Керує параметром public bool Cross.
No weights Perceptron	Якщо показує "Perceptron", то в першому поколінні дітей всі вхідні нейрони будуть з'єднані з вихідними нейронами. Керує параметром public bool PerceptronStart.
Ratio add weight	Пропорція використання мутації "додати вагу". Керує параметром public float MutationAddWeight.
Ratio change 1 weight	Пропорція використання мутації "змінити одну вагу". Керує параметром public float MutationChangeOneWeight.
Ratio change weights	Пропорція використання мутації "змінити всі ваги". Керує параметром public float MutationChangeWeights.
Ratio add neuron	Пропорція використання мутації "додати нейрон". Керує параметром public float MutationAddNeuron.
Ratio change 1 bias	Пропорція використання мутації "змінити одне зміщення". Керує параметром public float MutationChangeOneBias.
Ratio change bias	Пропорція використання мутації "змінити всі зміщення". Керує параметром public float MutationChangeBias.
Maximum in one time By waves	Можливість навчання хвилями в поколінні. Постійна максимальна кількість або певною кількістю в одній хвилі. Керує параметром public bool ChildrenByWave.
Children in wave	Кількість «дітей» в одній хвилі. Керує параметром public int ChildrenInWave.
Collision ON Collision OFF	Включити / виключити зіткнення між "дітьми". Керує параметром public bool IgnoreCollision.
Maximum count of weights	Максимальна кількість одночасного додавання зв'язків між нейронами у кожної "дитини". Керує параметром public int AddingWeightsCount.
Chance change sign	Шанс на зміну знаку вагового зв’язку. Керує параметром public float ChangeWeightSign
Children difference	Різниця вагових зв’язків між поколіннями. Керує параметром public float ChildrenDifference.
Autosave OFF Autosave ON	Включити / виключити авто збереження ШНМ під час навчання. Керує параметром public bool Autosave.
Autosave step	Зберігати ШНМ на кожному вказаному кроці. Керує параметром public int AutosaveStep.
GUI.TextField	Ім’я файлу для збереження або завантаження ШНМ.

Як користуватися.

Для початку треба створити певне завдання яка має виконувати ШНМ. Треба пам’ятати, що ШНМ має отримувати певні вхідні данні, і треба їх правильно підготувати. ШНМ сприймає вхідні данні від 0 до 1, або від -1 до 1 (див. ст. 4 «public bool AFWM»). Також треба визначитися в кількості вхідних даних.

Те саме стосується і вихідних даних. Отже їх треба вірно конвертувати для коректної відповіді на поставлене завдання.

Кількість прихованих шарів та нейронів в них вже залежить від навчання. В будь якому разі, кількість нейронів та зв’язків по різному впливає на якість та швидкість навчання ШНМ.

В цілому навчання ШНМ не сильно відрізняється від навчання у проекті "ANN Perceptron".

Завдання «Місія - не здохнути».

Розглянемо вже підготовлене завдання: Є «корова» яка з часом хоче їсти. Є «їжа» яку «корова» може з’їсти. «Корова помре» якщо не буде «їсти», або «з’їсть» забагато. «Корова» може «їсти» тільки передом, інакше «помре». Треба навчити «корову» правильно та вчасно «їсти».

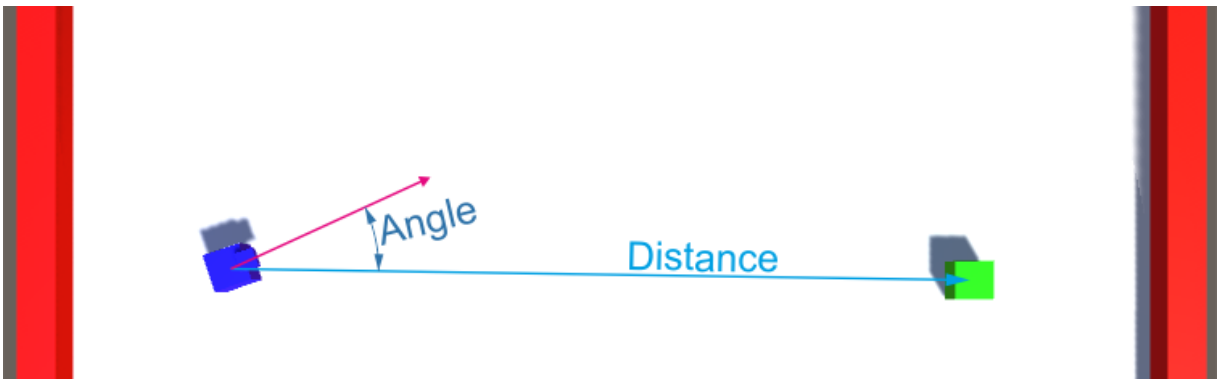
У папці «Tutorial» є вже заготовлені скрипти та сцена для завдання.

«Корова» отримує три значення:

1. Дистанція до «їжі». Діагональ рівня має біля 41.
2. Кут повороту зі знаком відносно переду «корови» до «їжі». Від -180 до 180.
3. «Ситість корови» яка зменшується з часом. 50 – максимальне значення для «виживання».

«Корова» керується двома значення:

1. Повернути до «їжі». Від -1 до 1.
2. Рухатися до «їжі». Від -1 до 1.



Також у «корови» є додаткові значення для отримання вище перерахованих.

Ось скрипт «корови» **TutorialCowControl.cs:**

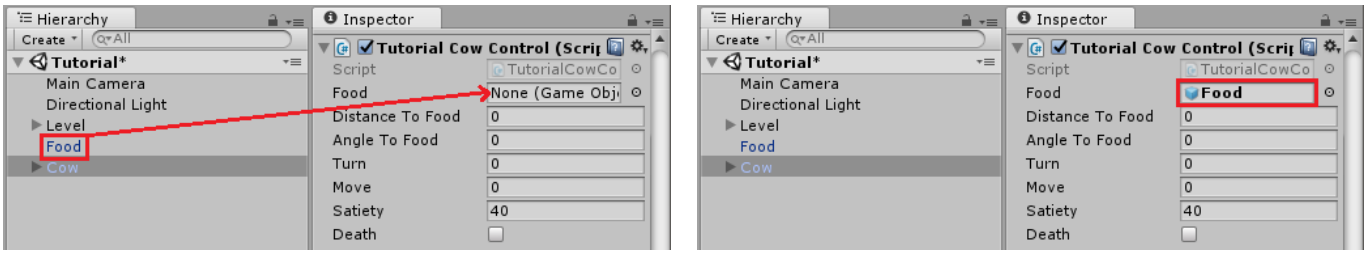
```
using UnityEngine;

public class TutorialCowControl : MonoBehaviour
{
    public GameObject Food;             //Food's GameObject
    public float DistanceToFood = 0;    //Distance to food
    public float AngleToFood = 0;       //Angle to food

    public float Turn = 0;              //Turn of cow
    public float Move = 0;              //Move of cow
    public float Satiety = 40;          //Satiety of cow
    public bool Death = false;          //If true - cow will die (reset position)

    void Update()
    {
        //Max & min turn
        if (Turn > 1)
            Turn = 1;
        else if (Turn < -1)
            Turn = -1;

        //Max & min move
        if (Move > 1)
            Move = 1;
        else if (Move < -1F)
            Move = -1F;

        //Cow reset
        if (Death)
        {
            Satiety = 40;
            transform.position = new Vector3(0, 0.5F, 0);
            transform.eulerAngles = new Vector3(0, transform.eulerAngles.y, 0);
            Death = false;
        }

        //Controls of cow
        transform.Rotate(0, Turn * 10F, 0);
        transform.Translate(0, 0, Move / 10F);

        //Food info
        DistanceToFood = Vector3.Distance(transform.position, Food.transform.position);
        AngleToFood = Vector3.Angle(transform.forward, Food.transform.position -
transform.position) * Mathf.Sign(transform.InverseTransformPoint(Food.transform.position).x);

        //The satiety of the cow decreases with time
        Satiety -= Time.deltaTime;
        if (Satiety < 0 || Satiety > 50)
            Death = true;
    }
}
```

«Їжа» впливає на «корову» при дотику. Якщо «корова» вірно «з'їсть їжу» (кут повороту не перевищує 5 градусів), то збільшує «ситість» (+15), а «їжа» міняє місце положення. Інакше «помре».

Ось скрипт «їжі» **TutorialFood.cs:**

```
using UnityEngine;

public class TutorialFood : MonoBehaviour
{
    private bool Moving = false;

    void Start ()
    {
        MoveFood();         //Move food
    }

    void Update()
    {
        if (Moving)
            Moving = false;
    }

    void OnCollisionEnter(Collision col)
    {
        TutorialCowControl TPC = col.gameObject.GetComponent<TutorialCowControl>();
        if (TPC != null && !Moving)        {
            //The cow must eat at a certain angle
            if (Mathf.Abs(TPC.AngleToFood) > 5)
                TPC.Death = true;
            else
            {
                TPC.Satiety += 15;
                MoveFood();
            }
        }
    }

    //Move food
    void MoveFood()
    {
        //Random position
        transform.position = new Vector3(Random.Range(-14F, 14F), 0.5F, Random.Range(-14F, 14F));
        Moving = true;
    }
}
```

Не забудьте вказати «корові» де «їжа»:



Керування «коровою» є, «їжа» є. А тепер треба створити «корові мозок».

Як створити ШНМ.

Щоб створити ШНМ треба створити MonoBehaviour скрипт та вписати в нього:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NameOfScript : MonoBehaviour
{
    public ANN AnnName = new ANN();
    . . .
    void Start()
    {
        AnnName.Create(3, 2);
        . . .
    }

    void Update()
    {
        . . .
        AnnName.Input[0] = . . . ;
        AnnName.Input[1] = . . . ;
        . . .
        AnnName.Solution();
        . . .
        . . . = AnnName.Output[0];
        . . . = AnnName.Output[1];
        . . .
    }
    . . .
}
```

Пояснення дивіться на сторінці 4.

Перед AnnName.Solution() треба ввести конвертовані вхідні данні у вхідні нейрони.

Після AnnName.Solution() – вивести вихідні данні вихідних нейронів та конвертувати їх.

Для завдання «Місія - не здохнути» створимо мозок «корови» у файлі під назвою **Lesson4TutorialCow.cs**. Його додайте до об’єкту «корови».

У «корови» є три основних значення які вона має сприймати та два значення керування (див. ст. 9). Отже потрібна ШНМ з трьома вхідними та двома вихідними нейронами. Щоб менше гратися з конвертацією значень створимо ШНМ «з мінусом» (див. ст. 4, bool AFWM). А вже кількість прихованих нейронів та зв’язків між ними буде залежить від навчання ШНМ.

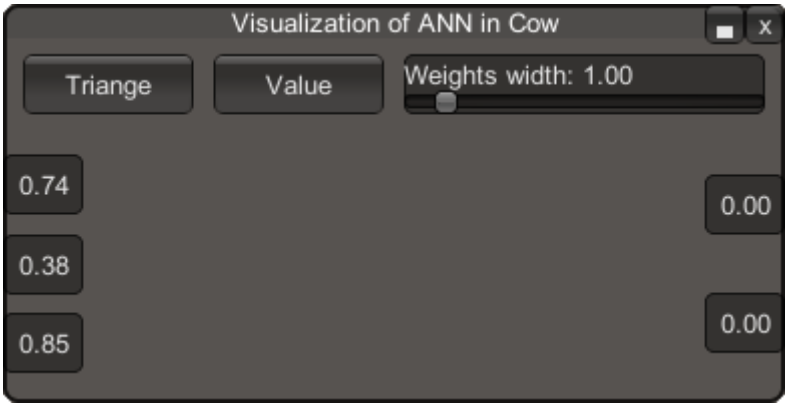
А на останок додаймо інтерфейс ШНМ для зручності зміни параметрів у ігровому режимі.

Не забуваємо, що для подачі даних ШНМ у вхідний шар потрібно ці данні конвертувати так щоб вони вірно сприймалися ШНМ. А вихідні значення конвертувати так щоб отримувати потрібні дані.

Ось скрипт «мозку корови» **Lesson4TutorialCow.cs:**

```
using UnityEngine;
public class Lesson4TutorialCow : MonoBehaviour
{
    private TutorialCowControl THC;     //Cow control
    public ANN Ann = new ANN();         //ANN
    private ANNInterface NI;            //ANN interface

    void Start()
    {
        //Find cow control
        THC = gameObject.GetComponent<TutorialCowControl>();

        //Create ANN
        Ann.Create(3, 2);

        //Add ANN interface to game object & add ANN to interface
        NI = gameObject.AddComponent<ANNInterface>();
        NI.Ann = Ann;
    }

    void Update()
    {
        //Convert vaule
        Ann.Input[0] = THC.AngleToFood / 180F;      //Work with angles. Min vaule = -180, max vaule =
180
        Ann.Input[1] = THC.DistanceToFood / 41F;    //Work with distance. Max vaule = 41
        Ann.Input[2] = THC.Satiety / 50F;           //Work with satiety. Min vaule = 0, max vaule = 50
        Ann.Solution();                             //ANN solution
        //For this tutorial not need to convert vaule
        THC.Turn = Ann.Output[0];
        THC.Move = Ann.Output[1];
    }
}
```

А ось як «мозок» буде виглядати по заданим параметрам зі скрипту вище:



І ТАК - тут пусто.

Урок №4. Навчання.

Для цього уроку знадобиться ввести «довголіття» до скрипту «корови» (див. ст. 10)

```
using UnityEngine;

public class TutorialCowControl : MonoBehaviour
{
    . . .
    public float LifeTime = 0;
    . . .
```

Це обумовлено тим, що при навчанні треба відслідковувати кращого «клона» у поколінні.

Крім того, бажано збільшувати «довголіття» при виконанні вірних дій, або/та зменшувати «довголіття» при виконанні не вірних дій.

Нехай «довголіття» збільшується з часом. І нехай зменшується «довголіття» коли «корова» невірно дивиться на «їжу».

```
    . . .
    void Update()
    {
        . . .
        LifeTime += Time.deltaTime - (Mathf.Abs(AngleToFood) / 180F) * Time.deltaTime;
    }
}
```

Також будемо скидати «довголіття» коли «корова помирає». Це не є обов'язковою дією (скрипт навчання сам анулює «довголіття»).

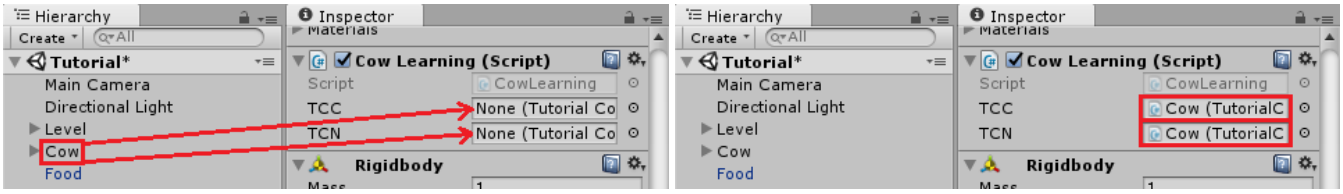
```
    . . .
    void Update()
    {
        . . .
        if (Death)
        {
            . . .
            LifeTime = 0;
        }
        . . .
```

Створимо скрипт **Lesson4CowLearning.cs** і додаємо його до будь-якого об'єкту:

```
using UnityEngine;

public class Lesson4CowLearning: MonoBehaviour
{
    . . .
```

Далі треба вказати скрипти керування «корови» та «мозок корови»:

```
    . . .
    public TutorialCowControl TCC;
    public Lesson4TutorialCow TCN;
    . . .
```



Додаємо інтерфейс навчання:

```
    . . .
    private ANNLearnByNEATInterface NLI;

    void Start()
    {
        NLI = gameObject.AddComponent< ANNLearnByNEATInterface >();
        . . .
```

Вкажемо інтерфейсу ШНМ яку треба навчити:

```
        . . .
        NLI.PCT = TCP. Ann;
        . . .
```

Та додаємо до методу нейроеволюції доповнюючих топологій потрібну інформацію щодо «корови»:

```
        . . .
        NLI. NL.StudentData(TCP.gameObject, TCP, "Ann", TCC, "Death", "LifeTime");
    }
}
```

Пояснення дивіться на сторінці 6.

Також, для більш ефективного навчання додаємо до **TutorialFood.cs** бонуси для «дітей», коли вони вірно «з'їдять їжу»:

```
. . .
    void OnCollisionEnter(Collision col)
    {
        . . .
        if (TCC != null && !Moving)
        {
            . . .
            else
            {
                . . .
                TCC.LifeTime += 15;
            }
        }
    }
. . .
```

Тепер можна запускати ігровий режим. Не використовуйте Without Minus (див. ст. 7) для даного уроку. Змінюйте настройки навчання по бажанню (див. ст. 8). Натискаємо на Learn OFF (див. ст. 8).

Тепер Ваша ШНМ буде навчатися. Ефект навчання зазвичай видно вже з перших поколінь. Але процес навчання може виявитися досить тривалим. Процес навчання можна рахувати завершеним, коли є суттєва різниця між Best Generation та Generation (див. ст. 8). Щоб зупинити процес навчання, та передати вагові зв'язки навчання до ШНМ який навчається натисніть Learn ON. Зберегти налаштування ШНМ, а потім завантажити їх ви зможете завдяки інтерфейсу ШНМ (див. ст. 7).

Як зберегти та завантажити налаштування ШНМ.

Як створити ШНМ ми вже розглядали (див. ст. 12). Як зберегти/завантажити налаштування та вагові зв'язки за допомогою інтерфейсу дивіться на сторінці 7. А от як зберегти та завантажити налаштування та вагові зв'язки у скриптах розглянемо далі.

Щоб зберегти налаштування та вагові зв'язки ШНМ використовуйте команду:

```
. . .
AnnName.Save("SaveName");
. . .
```

Де SaveName – ім'я файлу для збереження.

Файл налаштуваня та вагови зв'язків буде збережено за адресою:

```
Application.dataPath + "/ANN/ANN/" + ANNFile + ".ann"
```

ANNFile - ім'я файлу для збереження/завантаження (див. ст. 4).

Щоб завантажити налаштування та вагові зв'язки ШНМ замість використання команди «AnnName.Create (…)» використовуйте команду «AnnName.Load("LoadName");»:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NameOfScript : MonoBehaviour
{
    public ANN AnnName = new ANN();
    . . .
    void Start()
    {
        AnnName.Load("LoadName");
        . . .
```

Де LoadName – ім'я файлу для завантаження.

Заключне слово.

Сподіваюся, що Вам сподобається моя робота. Я намагався якомога полегшити завдання створення, навчання, збереження та завантаження налаштувань ШНМ та його вагових зв'язків.

Якщо у Вас виникнуть питання, або з'являться пропозицій щодо поліпшення даної роботи – пишіть на VirtualSUN13@gmail.com. З радістю відповім.

Особлива подяка Леониду Терешонкову за моральну підтримку та Юлі Павлович за допомогу у перекладі на англійську мову. ☺

PS: І не забувайте лишати свої відгуки про проект (http://u3d.as/1rTD). Буду Вам дуже вдячний.

Також, Ви можете оновити «ANN&NEAT» до «ANN&TOOLS».

Контакти.

YouTube: https://www.youtube.com/channel/UClblqhEzoATg-Jvk0AviVlw

Unity Connect: https://connect.unity.com/u/sergey-voroshilov-vladimirovich

Twitter: https://twitter.com/sun_virtual

Instagram: https://www.instagram.com/virtualsun13/

Forum: https://forum.unity.com/threads/released-ann-perceptron.558868/

E-mail: VirtualSUN13@gmail.com