G0 - Epitech Diversity

B-G0-000

Coding Club

SuperMarIAWorld

Bonjour à toi,

Aujourd'hui nous allons ensemble voyager dans le futur, dans ce qui n'est bientôt plus un mythe, l'intelligence artificielle ! L'atelier qui va t'être présenté va te permettre de comprendre les bases d'une intelligence artificielle, et comment en créer une.

Pour cela les Cobras vont te mettre à disposition un logiciel qui va pouvoir lancer un petit jeu pas très connu, Super Mario World, et d'injecter un programme que nous allons développer ensemble. Les cobras sont formés pour cela et pourront t'aider à répondre aux questions que tu pourrais avoir, n'hésite donc pas à leur demander de l'aide !

Bien, maintenant que tout cela est dit, commençons par quelques notions de ce que nous allons aborder aujourd'hui ! La théorie de l'évolution, et ce qu'est un réseau de neurones.

Je pense que vous avez déjà dû entendre parler de la théorie de l'évolution créé par Charles Darwin. Si jamais ce n'est pas le cas ou que vous ne saisissez pas complètement le principe, prenons un exemple simple :

Je pense que vous savez tous ce qu'est une girafe, mais si jamais en voici une.

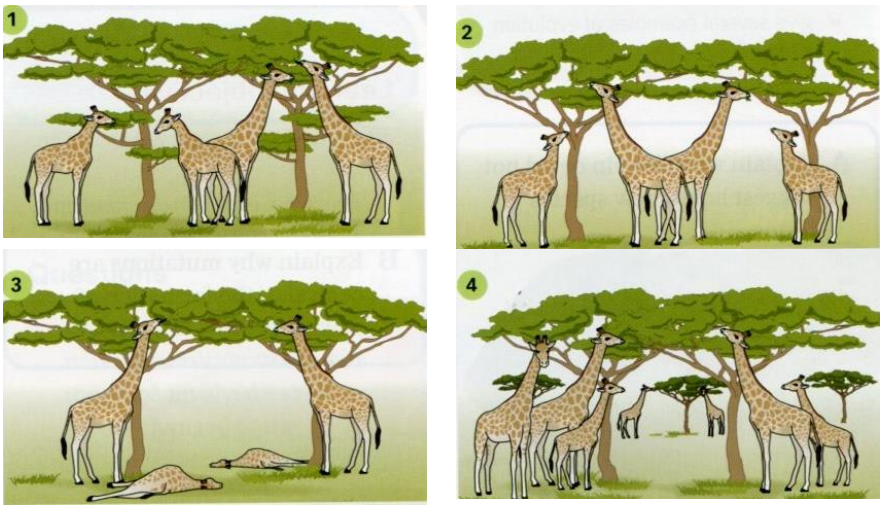
Appelons cette girafe Sophie !

Sophie est une girafe qui vit avec ses parents dans la belle savane africaine. Tous les jours elle se nourrit de feuilles dans les grands acacias, ce qu'elle est le seul animal à pouvoir le faire. Eh oui ! Sophie en est capable grâce à son long coup ! Autrement, elle ne pourrait pas se nourrir correctement.

À ce moment-là vous ne saisissez peut-être pas encore mais ce coup, comment l'a-t-elle obtenue ?

Eh bien pour répondre à cette question, remontons plusieurs années en arrière, mais vraiment énormément d'années en arrière ! Nous retrouvons alors les ancêtres de Sophie, dont d'ailleurs Zuri qui bizarrement n'a pas le même coup que notre chère Sophie !

Eh oui, Zuri a un petit coup, dans sa vie cela va être gênant pour lui, voir même extrêmement pénalisant s'il ne trouve plus d'acacias assez bas en hauteur pour se nourrir ! C'est donc pour cela que soit il finira par mourir sans laisser d'enfants avec ce facteur pénalisant, soit sa descendance aura plus de chance et grandira avec un long coup ou alors c'est elle qui finira par périr.



La théorie de l'évolution veut donc qu'une espèce évolue avec les facteurs les plus avantageux pour son espèce.

Si vraiment tu n'as pas compris ces explications, je te laisse regarder cette vidéo quand tu le voudras pour mieux comprendre !

C'est pas sorcier -THEORIE DE L'EVOLUTION : de darwin a la genetique

Bien, maintenant cela expliqué, ce que nous voulons appliquer à notre programme sera le même principe, il est question de lâchez une espèce avec plusieurs individus aux facteurs différents et de voir qui survit le plus longtemps pour créer la prochaine génération qui aura héritier de beaucoup de facteur du survivant précédent, mais aussi d'autres facteurs aléatoires. Cela peut se faire autant de fois que nous le voulons afin d'obtenir une descendance toujours plus performante et adaptée à son environnement !

Un algorithme génétique sert donc à résoudre un problème qui n'a pas une seule solution définie.

Explication d'un réseau de neurones

Un réseau de neurones est une technique d'intelligence artificielle qui s'inspire du fonctionnement du cerveau humain pour résoudre des problèmes complexes. Imaginez-le comme un ensemble de "neurones" artificiels, qui sont essentiellement des petites unités de calcul. Ces neurones sont organisés en couches, à savoir une couche d'entrée, des couches cachées intermédiaires et une couche de sortie.

La couche d'entrée reçoit des données brutes, telles que des pixels d'une image, et chaque neurone dans cette couche correspond à une caractéristique particulière de ces données. Les informations sont ensuite transmises aux couches cachées, où les neurones effectuent des calculs complexes pour extraire des caractéristiques importantes et créer une compréhension plus profonde des données. Enfin, la couche de sortie produit la réponse finale du réseau, par exemple, en indiquant si une image représente un chat ou un chien.

Ce qui rend les réseaux de neurones puissants, c'est la capacité d'apprentissage. Les connexions entre les neurones ont des poids qui déterminent l'importance de chaque connexion. Au départ, ces poids sont ajustés de manière aléatoire. Cependant, à mesure que le réseau est exposé à des exemples d'entraînement, il ajuste ces poids pour minimiser les erreurs entre ses prédictions et les résultats réels.

Le processus d'apprentissage implique généralement une étape de rétropropagation de l'erreur, où le réseau compare ses prédictions à la réalité, calcule les erreurs et ajuste progressivement les poids des connexions pour s'améliorer. Ce processus se répète jusqu'à ce que le réseau atteigne un niveau de performance souhaité.

Les réseaux de neurones ont montré leur efficacité dans de nombreuses applications, de la reconnaissance d'images à la traduction automatique, en passant par la prédiction de séries temporelles. Ils sont au cœur de nombreuses avancées en intelligence artificielle et continuent d'évoluer pour résoudre des problèmes de plus en plus complexes.

Je pense que maintenant que nous avons vu les deux concepts principaux nous pouvons nous attarder sur le langage de programmation que tu vas utiliser, le Lua. Pour cela on va faire un rapide tour global.

Commençons par les variables, en Lua, elles sont globales par défaut, mais tu peux les déclarer comme locales si nécessaire.

```lua
x = 10
local y = 20
```

Le Lua prend en charge plusieurs types de données de base, tels que les nombres (entiers et à virgule flottante), les chaînes de caractères et les booléens. Pour ce qui est des tableaux, ils peuvent contenir des valeurs de différents types et être utilisées pour créer des listes, des dictionnaires et des objets complexes.

```lua
local fruits = {"pomme", "banane", "orange"}
print(fruits[2]) -- Affiche "banane"
```

Le Lua prend en charge les boucles "for", et "while", ainsi que les structures conditionnelles "if", "else", et "elseif".

```lua
for i = 1, 5 do
    print(i)
end

if x > 10 then
    print("x est supérieur à 10")
else
    print("x est inférieur ou égal à 10")
end
```

On peut bien entendu définir et appeler des fonctions en Lua. Les fonctions peuvent avoir des paramètres et renvoyer des valeurs.

```lua
function addition(a, b)
    return a + b
end

local resultat = addition(5, 3)
print(resultat) -- Affiche 8
```

Les tables Lua sont des structures de données très polyvalentes. Elles peuvent être utilisées pour créer des tableaux, des dictionnaires, des enregistrements, et bien plus encore. Les tables peuvent contenir des valeurs de différents types, y compris d'autres tables.

```lua
local personne = {
    nom = "Jean",
    age = 30,
    ville = "Paris"
}
print(personne.nom) -- Affiche "Jean"
```

Les tables en Lua peuvent être utilisées comme des listes associatives ou des dictionnaires, où chaque élément est associé à une clé unique.

```lua
local capitales = {
    France = "Paris",
    Allemagne = "Berlin",
    Espagne = "Madrid"
}
print(capitales.France) -- Affiche "Paris"
```

Lua ne dispose pas d'une structure de file intégrée, mais vous pouvez facilement implémenter une file en utilisant une table. Les opérations d'ajout (enqueue) et de suppression (dequeue) se font en modifiant la table.

```lua
local file = {}
table.insert(file, "premier")
table.insert(file, "deuxième")
local element = table.remove(file, 1) -- Retire le premier élément (premier)
```

Les piles peuvent également être implémentées en utilisant des tables Lua. Vous pouvez ajouter des éléments au sommet et les retirer du sommet.

```lua
local pile = {}
table.insert(pile, "élément 1") -- Ajoute un élément au sommet
local sommet = table.remove(pile) -- Retire l'élément du sommet
```

Bien que Lua ne dispose pas nativement d'une structure de données de type ensemble, vous pouvez simuler un ensemble en utilisant des tables où les clés sont utilisées pour représenter les éléments.

```lua
local ensemble = {}
ensemble["élément1"] = true -- Ajoute un élément à l'ensemble
local existe = ensemble["élément1"] -- Vérifie si l'élément existe dans l'ensemble
```

Ceci résume les bases de Lua. Il s'agit d'un langage de programmation polyvalent, utilisé dans divers domaines, y compris le développement de jeux, la programmation embarquée et la création de scripts pour automatiser des tâches.

Voilà je pense que tout est dit et qu'avec cela tu pourras mieux comprendre ce que tu vas faire, maintenant passons à ce qui nous intéresse le plus, faire en sorte que notre petit Mario ne soit plus immobile !

Pour commencer, je vais t'expliquer comment lancer le jeu et le programme :

- Commence par lancer l'émulateur, un cobra peut aider si jamais tu as un problème.
- Ensuite va sur la tirette "File", clique sur "Open ROM..." et sélectionne la ROM du nom de "Super Mario World (USA).sfc".
- Pour maintenant lancer le programme, va sur la tirette "Tools", clique sur "Lua Console", tu devrais voir une fenêtre s'ouvrir. Dans cette fenêtre, va sur la tirette "Script", et clique sur "Open Script", sélectionne maintenant "main.lua".

Maintenant que tu as lancer le programme, rien ne se passe ? Pourtant normalement tu devrais voir Mario ne pas bouger sur le premier niveau ?

Hmm, rend toi dans utils et va à la ligne 204 avec "function lancerNiveau()" et complète là ?

Voilà c'est mieux ! Maintenant tu devrais te rendre compte que pour le moment Mario ne bouge pas et que le jeu redémarre continuellement, pour résoudre cela on va aider Mario à retrouver l'usage de son cerveau !

On va commencer par apprendre à Mario comment créer un neurone ! Pour cela rend toi dans le fichier "neurone.lua", tu devrais y retrouver "function newNeurone()" à la ligne 9. Tu as quelques commentaires sur comment tu pourrais réparer cette fonction, utilises ce qui t'as était appris pour le faire, n'oublie pas que tu peux demander à un Cobra de t'aider !

Ensuite tu vas réparer la fonction "function ajouterNeurone(unReseau, id, type, valeur)" à la ligne 18. Cette fois Mario à quelques problèmes pour utiliser ses neurones qu'il arrive désormais à créer mais pas à les ajouter aux réseaux de neurones. Les commentaires devraient t'aider à régler tous ça !

On parlait justement de connexions, mais encore faudrait-il quelles marchent ! Dans le fichier "connexion.lua", Mario n'arrive plus à créer de Connexion ou à les ajouter, est-ce que tu peux régler cela ?

C'est bien, tu es arrivé à vite t'habituer à tout ça ! Mais je crois qu'il manque un élément, tu as bien réussi à réparer les neurones de Mario et leurs connexions mais il manque quelque chose…

Mais oui ! Ce sont les réseaux qui permettent d'acceuillir tout cela ! Va vite dans le fichiers "reseau.lua" tu devrais y retrouver "function newReseau()" à la ligne 9 et "function majReseau(un Reseau, marioBase) à la ligne 29. Elles ont, elle aussi quelques problèmes, tu pourrais t'en charger ?

Bon je crois que maintenant Mario à un cerveau bien fonctionnel, seulement je te rappelle que son objectif était de devenir le meilleur coureur dans le Royaume Champignon ! Pour cela il a eu l'idée de rêver continuellement de ses parcours, et il a besoin de toi pour créer ce rêve infini où il pourra continuellement s'entraîner !

Si Mario est seul il ne peut rien, par contre si on crée une population de pleins de Mario, alors là il arrivera à accomplir ses rêves ! Va dans le fichier "population.lua" et termine la fonction " newPopulation" !

Pour cela, il faut d'abord que tu récupères tout les inputs, Mario t'as donné quelques indications que tu peux retrouver dans le fichier "game.lua", il va falloir que tu termines les fonctions " getIndiceLesInputs" et "getLesInputs".

Bravo pour ton ton travail ! Si tu décommentes les lignes les lignes 126, 127 et 128 dans le fichier "main.lua", et que tu relances le script, Mario devrait être capable d'accomplir ses rêves !

Tout cela est bien, seulement tu n'es pas encore capable de voir sa progression ou même de la sauvegarder.

Bien, ça ne devrait pas être très compliqué pour quelqu'un qui vient d'aguerri comme toi ! D'abord pour te permettre de voir la progression de Mario rends toi dans "main.lua", là tu y trouveras une fonction "printLabel" où il y a quelques indications pour t'aider !

Ensuite pour ce qui est de sauvegarder la mémoire de Mario, rends-toi dans "utils.lua" et modifie les commentaires dans "getNomFichierSauvegarde".