CFPT-INFORMATIQUE

Travaux de Diplômes 2017

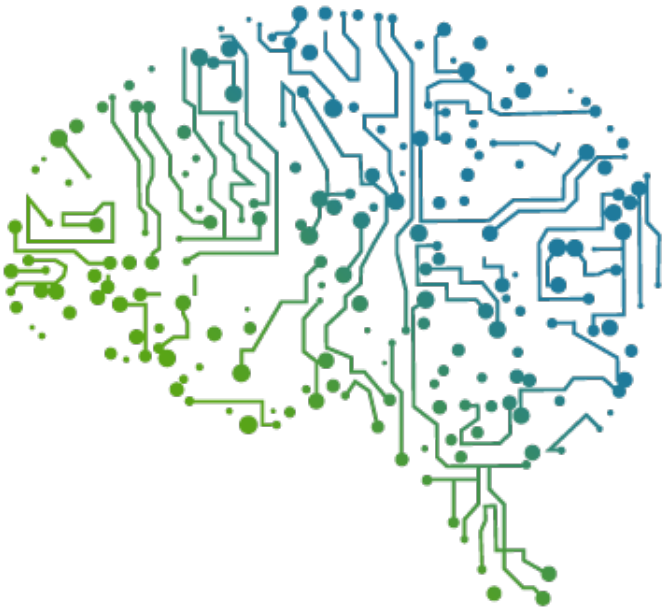
T-Rex Ceptional

Journal de bord

Loris De Biasi

Supervisé par :

M. Wanner



T.IS-E2A

12 juin 2017

5 avril 2017

- Explication donné par M.Marechal sur le déroulement du travail de diplôme
- Création de la structure du journal de bord ainsi que de la documentation technique
- Création d'un repository sur github
- Réalisation du Gantt
- Rédaction du résumé
- Rédaction de l'Abstract
- Recherche pour l'étude d'opportunité
  - Flappy learning
  - MarI/O
  - Forza Drivatar
  - Startcraft DeepMind
- Rédaction de l'étude d'opportunité
  - Rédaction de Flappy learning
  - Rédaction de MarI/O

6 avril 2017

- Rédaction de l'étude d'opportunité
  - Fin de la rédaction de MarI/O
  - Rédaction de Forza Drivatar
  - Rédaction de Startcraft DeepMind
- Se que j'avais en fait rédigé comme étant le résumé et l'abstract était enfaîte l'introduction. Le résumé et l'abstract doivent être fait à la fin du travail.
- Création de maquette d'interface, besoin de :
  - Affichage du jeu
  - Affichage des boutons pressés
  - Affichage de certaine variable (distance, vélocité, taille et la génération actuelle)
- Rédaction de l'analyse fonctionnelle
  - Rédaction de la partie expliquant le choix du modèle et les différents types
  - Rédaction de la partie expliquant le fonctionnement et l'utilité des diverses parties du modèle
  - Début de la rédaction de l'explication sur les algorithmes génétiques

7 avril 2017

- Continuation de la rédaction de l'analyse fonctionnelle
- Recherche sur les réseaux de neurones
  - lecture d'article pour apprendre à faire un réseau de neurones https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/
- M.Wanner est passé pour vérifier le bon déroulement du projet.
- Arrêt de la rédaction de l'analyse fonctionnelle. N'étant pas certain de certaine chose, je préfère développer un petit réseau de neurones avant de continuer.
- Programmation d'un petit réseau de neurones en Javascript pour vérifié que je n'ai rien oublié dans l'analyse fonctionnelle
  - Création de la classe "NeuralNetwork"
  - Création de la fonction "forward" qui permet de faire passer une valeurs au travers du réseau de neurones

  - Création d'une fonction "multiplyMatrix" permettant de faire de la multiplication de matrice
  - Création de la fonction d'activation Sigmoid
  - Création de la fonction SigmoidPrime
  - Création de la fonction costFunctionPrime
  - Création de la fonction operationOnMatrix qui prend deux matrices ainsi qu'une fonction en paramètre. Elle permet, comme son nom l'indique, d'effectuer une opération sur les deux matrices (exemple : addition)
  - Création de la fonction generateWeightsArray qui permet de générer les poids de base du réseau de neurones
  - Création de la fonction getMaxValue permettant de récupérer la valeur la plus élevée d'un tableau
  - Création de la fonction transposeMatrix permettant de transposer les valeurs d'une matrice (inverser axe X et Y)
- Problème dans les fonctions utilisant les matrices. Je me suis aidé de ce site : http://www.calcul.com/show/calculator/matrix-multiplication
- Problème dans la fonction operationOnMatrix, mauvaise taille de matrice en sortie.

10 avril 2017

- Correction du problème de operationOnMatrix
- Création de nouvelle fonction
  - getParams permettant de récupérer les poids du réseau sous forme d'un tableau à une dimension
  - setParams Permettant de changer les poids du réseau de neurones (prend un tableau 1d en entré)
  - computeGradient permettant de calculer le gradient
  - computeNumericalGradient permettant de tester le réseau de neurones
  - ravel permettant de transformer un tableau multidimensionel en tableau 1d
  - slice permettant de "découper" un tableau pour n'en récupérer qu'un partie
  - reshape permettant de changer la forme d'un tableau (exemple : 4x2 en 2x4)
  - operationOnMatrixValuesByNumber permettant d'effectuer une opération sur une matrice (prend une matrice, un nombre et une opération en entré)
- Problèmes rencontrés lors de l'utilisation des fonctions de tests. Le résultat obtenu n'est pas le bon.
- Après avoir parcouru tout le code et corriger beaucoup de bug, je ne trouve toujours pas d'où vient le problème, je vais donc continuer et je reviendrai dessus à la fin (même en sachant que le résultat est faux, le but de ce projet est d'être sûr que je n'ai rien oublié).
- Gros problème rencontré. Il faut maintenant que je crée la méthode d'entraînement du réseau de neurones et la méthode utilisée contient très peu de documentation et est très complexe.
  https://en.wikipedia.org/wiki/Broyden-Fletcher-Goldfarb-Shanno_algorithm
- Je vais devoir effectuer quelque recherche et lire quelque article sur le sujet
- Je pense que demain je ferai une pause sur le code pour commencer le poster

11 avril 2017

- Début de la création du poster
- Visite de M. Wanner pour venir voir le déroulement du projet
  - Étant bloqué sur la mise en place de la méthode d'entraînement du modèle, nous avons discuté et M.Wanner a en l'idée d'effectuer des recherches sur github directement ainsi que de rechercher "BFGS for idiots" pour voir si une explication simple existait.
    - recherche de BFGS sur github : plusieurs résultats qui ont l'air utile.
    - recherche de BFGS for idiots : aucun résultat.
  - J'ai donc décidé que je ferais le poster avant le début des vacances et pendant celle-ci j'essaierai de me documenter sur BFGS
- Fin du poster
- Début de recherche sur les méthodes d'entraînement
- visionnage et lecture de plusieurs tutoriels sur internet https://www.youtube.com/watch?v=9aHJ-FAzQaE https://www.youtube.com/watch?v=KkwX7FkLfug https://blog.webkid.io/neural-networks-in-javascript/

12 avril 2017

- Fin du visionnage de la vidéo sur la création d'un réseau de neurones, très bien expliqué par ailleurs. https://www.youtube.com/watch?v=KkwX7FkLfug
- prise de note et début de création du diagramme UML
- Je suis en train de me demander si l'utilisation d'un réseau de neurones avec des algorithmes génétiques est ce qu'il y a de plus "optimisé" et si c'est vraiment ce que je souhaite mettre en place. Plus je fais de recherche, plus je me dis qu'un réseau de type "reinforcement learning" serait le plus adapté. Je continue donc de me documenter sur les réseaux de neurones et je verrai après avoir effectué des recherches pendant les vacances si la mise en place d'un réseau de type "reinforcement learning" est quelque chose de faisable. Si c'est le cas, je demanderai à M.Wanner s'il est possible de changer le cahier des charges (bien que le projet soit déjà commencé). Je savais déjà avant de commencer ce travail que le "Reinforcement learning" serait plus "logique" et plus rapide, mais étant donné que je n'avais jamais fait de machine learning ou bien vu le code d'un réseau de neurones je ne me suis même pas dit que mettre en place du "Reinforcement learning" serait possible. https://www.youtube.com/watch?v=2pWv7GOvuf0
- Recherche sur l'utilité des "bias" http://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks
- Recherche de framework pour l'interface graphique du site, j'ai trouvé le framework "materializecss" qui me semble être bien. http://materializecss.com/grid.html
- Création de l'interface du site avec le framework "materializecss"
- Récupération des informations utiles et affichage de celle-ci sur l'interface
  - Vitesse
  - Distance
  - Taille
- Je suis en train de me dire que le T-Rex ne saura pas différencier un cactus d'un ptérodactyle. Si j'utilise le réseau de neurones que j'avais prévu au début, il fau-

dra que je rajoute un nouveau neurone "input" qui sera certainement l'axe Y. Si par contre j'utilise le "Reinforcement learning", je crois qu'il est possible de faire autrement qu'en lui renseignant toutes les valeurs qu'il doit analyser. Il faudra que je me documente à ce sujet.

24 avril 2017

- Après avoir longuement réfléchi pendant ses vacances, je vais finalement rester sur mon idée de départ qui est d'utiliser un réseau de neurones et de l'entraîner en utilisant des algorithmes génétiques. L'autre option était d'utiliser le "Reinforcement learning" qui est plus "logique" et plus complexe. Ce qui m'a fait m'orienter vers mon idée de départ est que j'ai pensé à plusieurs choses qui peuvent être intéressantes à mettre en place.
  - Mise en place d'exportation du réseau de neurones sous format texte (xml peut être)
  - importation de réseau de neurones (soit en format texte, soit un fichier)
  - visualisation et modification en direct du réseau de neurones
- Toutes les choses citées au-dessus sont des ajouts qui, je trouve, seraient très intéressante à mettre en place. Je vais donc essayer, dans les mesures du possible, de toutes les implémenter.
- Mise à jour de l'interface d'après les idées citée au-dessus.
  - Affichage de la position Y des obstacles sur le site
  - Ajout d'un bouton permettant d'accéder à un menu
  - Ajout d'une option "Export" permettant d'exporter le réseau de neurones
  - Ajout d'une option "Import" permettant d'importer un réseau de neurones
  - Ajout d'une option "Show/modify neural network" permettant d'afficher et de modifier le réseau de neurones.
- Les trois boutons ajoutés ouvrent une fenêtre pour effectuer les actions.
- La fenêtre "Export" contient une textarea affichant le réseau sous format texte, un bouton "Export to file" permettant de télécharger le réseau de neurones (pour l'instant en format txt) ainsi qu'un bouton permettant de fermer la fenêtre
- La fenêtre "Import" contient un bouton permettant de sélectionner un fichier local contenant un réseau de neurones, une textarea qui affiche le contenu du fichier sélectionné et un bouton qui permet de valider l'importation du réseau de neurones. (ainsi qu'un bouton pour fermer la fenêtre)
- La fenêtre "Show/modify the neural network" permettra (pas fait car requiers un réseau de neurones) d'afficher le réseau de neurones en direct et de modifier chacun des poids via des inputs.
- Les fenêtres peuvent être fermées en cliquer en dehors de celle-ci
- Les textarea des fenêtres "Import" et "Export" peuvent être remplis a la main.
- Ajout d'une fonction permettant de nettoyer les fenêtres lors de la fermeture.
- Ajout d'un message "toast" à chaque fois qu'une action est effectuée ou qu'une erreur survient. Exemple : "Neural net succesfully exported", "Neural net succesfully imported", etc..
- Ajout d'une fenêtre "Contact" qui contiendra diverses informations sur moi-même
- Ajout d'une fenêtre "About" qui contiendra diverses informations sur le projets
- Création d'une fonction permettant de télécharger un fichier

- Création d'une fonction permettant de lire un fichier et d'en afficher les contenus dans un textarea. (utilisation de l'objet FileReader)
- Création de la structure du code
  - Main
  - neural_network
  - layer
  - neuron
  - connection
  - activation_function
- Début de la création de la classe neural_network, layer, neuron et connexion

25 avril 2017

- Création des classes neural_net, layer, neuron, connection.
- neural_net contient :
  - Un constructeur qui prend en paramètre une topology (renseigne la structure du réseau sous forme d'un tableau 1d) et qui permet de créer le bon nombre de layer.
  - La méthode "feedForward", qui prend en entrée un tableau 1d qui permet de renseigner les valeurs qui doivent être passées dans le réseau et de les faires passer tout au long du réseau de neurones. (propager les valeurs d'entrées jusqu'a la sortie)
  - La méthode "backPropagation" qui ne sera pas utilisée dans ce projet mais qui peut avoir une utilité. Elle permet de calculer la marge d'erreur du réseau de neurones en parcourant le réseau et en lancant des fonctions présente dans d'autre classes.
  - La méthode "getResults", qui permet de retourner le résultat du réseau de neurones.
- layer contient :
  - Uniquement un constructeur qui prend en entrée un nombre de neurones et qui s'occupe de créer des neurones.
- neuron contient :
  - Un constructeur qui prend en entrée un nombre d'output permettant de savoir combien de valeur il va falloir stocker ainsi que la création de connexion entre les neurones et de deux valeurs (eta et alpha) qui sont des valeurs renseignées à la main qui sont utilisées dans le calcul de poids.
  - Une méthode "feedForward" qui prend un tableau représentant la layer précédent et qui fait la somme de celui-ci pour trouver les valeurs du layer actuel.
  - Une méthode "calculateOutputGradients" qui permet de calculer le delta entre la valeur souhaitée et la valeur trouvée (utilisation de la fonction tanh dérivée)
  - Une méthode "calculateHiddenGradients" qui prend le layer suivant en paramètre
  - Une méthode "updateInputWeights" qui permet de mettre à jour les poids
- connection contient :
  - Un constructeur qui va s'occuper de créer des poids aléatoires
  - Une méthode "randomWeight" qui prend une valeur min ainsi qu'une valeur max pour pouvoir tirer un nombre aléatoirement entre ces deux valeurs (les

deux valeurs inclues). Fonction trouvée sur le site de mozilla (url renseigné dans le code)

- Ajout d'un bouton sur l'interface permettant de lancer l'IA (réseau de neurones), lié a une fonction qui va afficher un message toast (soit "Stopped", soit "Started"), ainsi que changer une variable boolean permettant de savoir si le jeu est démarré ou non.
- Ajout d'un bout de code dans le jeu du T-Rex directement dans la partie "update" (fonction qui gère le rafraîchissement de tout le jeu). le bout de code lance une fonction chaque X temps lorsque le jeu est lancé. Cette fonction va s'occuper d'afficher les données du jeu à l'écran (distance, vélocité, etc..) et lorsque l'IA est lancé, la simulation va se lancer.
- Maintenant que le réseau de neurones est fini (ou en tout cas j'en ai l'impression), je commence à créer le lien entre le jeu du T-Rex et le réseau pour que le T-Rex effectue des actions selon celui-ci.
- Recherche sur comment simuler la pression d'une touche
- J'ai trouvé plein de chose mais rien ne fonctionne
- Après avoir cherché pendant 1h environ, il s'avère qu'il faut utiliser la fonction "document.dispatchEvent()" pour que cela fonctionne. Cela permet "d'envoyer" l'évènement sur le "document".
- Création de la fonction prenant en entrée un keycode (nombre représentant une touche) qui permettra de simuler la pression d'une touche.
- Changement de l'instanciation du jeu du T-Rex. De base, le jeu se lance depuis le fichier javascript du jeu, cela a été changé. Je l'ai mis dans le "main" pour avoir un accès à tout ce que je souhaite, principalement pour du débogage et pour tout avoir de regrouper.

26 avril 2017

- envoie des données du jeu dans l'IA
- Problème. la valeur de sortie obtenu était toujours la même, ce qui ne devrait pas être le cas. Les données que l'IA recevait étaient envoyées en string, ajout de "parseFloat".
- Le problème persiste, j'ai du faire une erreur dans la méthode "feedForward"
- Changement de la structure de la classe "activationFunction". Elle contient maintenant une classe abstract qui se nomme "activationFunction" ainsi que (pour l'instant) deux classes qui se nomment respectivement "tanh" et "sigmoid". Le but de cela est que je peux maintenant changer de méthode d'activation à la volée dans le constructeur du réseau de neurones.
- Je pense savoir d'où viennent mes erreurs, il faut que je normalise les données d'entrée. Pour cela je avoir besoin des valeurs maximales possibles en entrée.
- Le problème vient effectivement de là, après avoir renseigné les valeurs max a la main et normalisé les valeurs d'entrées, les calculs s'effectuent normalement et correctement. Il faut maintenant que je récupère les valeurs directement depuis le jeu du T-Rex.
- J'ai réussi à récupérer deux valeurs (velocity et size), mains la distance et la position en Y sont, soit introuvable, soit je n'arrive pas à les récupérer depuis le "main" sans instancier un objet du jeu. Pour l'instant ces deux valeurs sont renseignées à

la main.

- Pour l'instant l'IA "fonctionne". Elle effectue des actions, mais puisqu'elle n'est pas entraînée elle ne fait rien de bien fou.
- Avant de commencer à coder l'entraînement, je vais m'occuper de faire en sorte que si l'IA perd, elle relance le jeu.
- j'ai réussi à trouver le bon évènement (.crashed), mais le jeu ne se relance pas alors que je simule l'appui de la touche du haut qui devrait faire redémarrer le jeu.
- Après quelques tests, la simulation de touche n'a pas l'air de fonctionner lorsque le joueur "meurt".
- Finalement, j'ai enlevé le bout de code que j'avais mis directement dans le script du jeu du T-Rex, j'ai fait en sorte de pouvoir l'exécuter depuis le "main", ce qui est plus logique et me permet de gérer le rafraîchissement et le comportement comme je le souhaite.
- Après plusieurs recherches, il s'avère que le jeu ne modifie pas le DOM lorsque le joueur perd, je ne comprends donc pas pourquoi l'évènement ne fonctionne pas uniquement dans ce cas.
- J'ai finalement réussi à trouver comment faire. Normalement, la simulation d'appui de bouton se faisait via l'évènement "keydown" et fonctionne pour le jeu. Il se trouve que pour redémarrer, l'application regarde l'évènement "keyup", d'où le non-fonctionnement de ma fonction. J'ai donc modifié la fonction pour qu'elle prenne en paramètre le type d'actions (exemple : "keyup", "keydown") et maintenant ça fonctionne.
- Je pense que tout est en place pour que je commence à mettre en place l'entraînement. Je vais donc commencer à faire des recherches.
- Création de la classe "Generation" qui contient :
  - Un constructeur qui prend en entrée la topology (pour pouvoir la transmettre), le nombre de génome souhaité ainsi que la fonction d'activation (pour pouvoir la transmettre également). le constructeur s'occupe d'instancier le bon nombre de génomes et les ajoute dans un tableau.
  - Une méthode "run" qui prend en entrée toutes les valeurs d'input qui vont passer dans le réseau de neurones et qui s'occuper de les normaliser, de les envoyer au réseau de neurones pour appliquer la méthode "feedForward" ainsi que gérer l'action du T-Rex selon le résultat de sortie.
  - Une méthode "nextGen" qui prend en entrée le fitness (score du T-Rex) et qui va soit changé le currentGenome, soit changer la génération actuelle pour plus tard effectuer le crossover ainsi que la mutation
- Création de la classe "Genome" qui contient pour l'instant :
  - Un constructeur qui s'occupe de transmettre la topology ainsi que la fonction d'activation. Il s'occupe également d'instancier un neuralNet.
  - Une méthode "setFitness" qui permet de changer le fitness (stocker dans la classe genome). Le fitness du T-Rex est, pour l'instant, simplement la distance qu'il a parcourue. Je ne pense pas que ce soit la bonne valeur à prendre comme valeur de fitness, mais les tests qui seront faits une fois l'application terminée me le diront. Ce que le T-Rex doit faire est d'éviter les obstacles, je pense donc qu'un meilleur fitness serait de calculer le nombre d'obstacles sautés.
- Ajout de l'affichage du génome actuel sur l'interface (en dessous de la génération)

- Changement de l'interface temporairement. J'ai ajouté un endroit en dessous du jeu du T-Rex où sont affichées tous les poids pour me permettre de debuger plus facilement. La génération se fait automatiquement et c'est dynamique. Du coup, la partie "controls" a été déplacé dans la partie "informations"

27 avril 2017

- Visite de M.Wanner. Je lui ai montré l'avancement du projet et une idée est sortie, il serait intéressant de pouvoir afficher plusieurs jeux du T-Rex avec une IA différente.
- Malheureusement, c'est impossible à faire sans utiliser de bibliothèque (nodejs, etc..) et peut-être même impossible tout simplement. Le problème est qu'il faut pouvoir simuler l'appui d'une touche sur un canvas en particulier et simuler plusieurs pressions de touche en même temps. De plus, je ne crois pas qu'il existe de système de threading bien que je croie que les "workers" soient quelque chose de similaire.
- Création de la fonction "ravel" qui permet de transformer un tableau 2d en tableau 1d
- Ajout de méthode pour la classe "Generation"
  - "selection" qui permet de sélectionner les meilleurs génomes et de les transformer en tableau 1d pour les envoyer dans une méthode de crossover
  - "singlePointCrossover" qui est une méthode de crossover. Elle récupère tous les poids sous forme de tableau 1d et effectue de l'enjambement (crossover) en un point (division d'un genome en 2), puis créer une nouvelle génération. Je pense ajouter d'autre méthode de crossover dans le futur telle que : twoPointCrossover et uniformCrossover.
- Beaucoup de problèmes rencontré pour créer une copie d'un objet et non pas une référence. Par défaut, lors d'une assignation d'un objet a une variable, javascript le fait toujours par référence, le problème est qu'il n'existe pas de méthode pour préciser que l'on souhaite en créer une copie. J'ai donc modifié légèrement la classe "Generation" pour pouvoir l'instancier depuis elle-même (stockage de certaines valeurs).
- Modification de la classe Genome, neural_net, layer, neuron et connection
  - Création d'une méthode "getWeights" qui permet de récupérer les poids sous forme de tableau 1d.
  - Création d'une méthode "setWeights" qui prend un tableau 1d en entrée et qui permet de changer la valeur des poids d'un réseau de neurones, d'un layer, d'un neurone, ou bien d'une connection
- Bug dans la méthode "setWeights" de la classe "neural_net". Je dois découper le tableau 1d d'entrer en plusieurs parties puis les envoyer sur la bonne partie du réseau de neurones. Le découpage ne se fait pas comme il faut.

28 avril 2017

- Il se trouve que le problème ne venait pas de la méthode "setWeights", en tout cas pas complètement. Premièrement, les poids des meilleurs réseaux étaient mal récupéré et mal "ordonné", a la place d'avoir 15 poids, j'en avais 19. Deuxièmement, lors de la création de nouveaux poids, je me suis trompé sur un index, du coup il manquait 1 poids. Troisièmement, la structure envoyée pour le "setWeights" n'était pas bonne (tableau 2d a la place de 1d). Maintenant les enfants sont créés

correctement, mais la méthode "setWeights" pose toujours problème.
- Le problème est corrigé et maintenant le crossover "fonctionne". Les valeurs créées s'assignent correctement. Le problème venait simplement du fait que j'utilise une méthode "slice" qui permet de découper un tableau 1d en renseignant la position de début et de fin, le problème est que je pensais qu'elle avait besoin d'un index de début ainsi qu'une range a récupéré.
- Maintenant un autre problème survient. Je ne sais pas encore si le problème vient de la création des nouveaux poids ou de l'assignation, mais tous les réseaux ont les mêmes poids après un crossover.
- Après avoir regardé, il semble que le problème vienne de la création des nouveaux poids.
- Correction du calcul du crossoverPoint. Le calcul effectué était mauvais, je tirais un nombre aléatoire dans la bonne range, mais le problème est qu'il s'agissait d'un nombre à virgule, j'ai donc rajouté un "Math.floor". Cela a permis en partie de corriger le problème des nouveaux poids.
- Modification du crossoverPoint pour qu'il se calcule pour chaque nouveau neurone (correction d'une partie du problème).
- Modification de la fonction "singlePointCrossover". Jusqu'à présent, je prenais une partie d'un réseau de neurones ainsi qu'une partie du deuxième réseau de neurones, mais ils étaient toujours mélangé dans le même ordre (le premier, puis le second), j'ai donc rajouté le tirage d'un boolean aléatoire qui permet de définir quel sera la première partie du génome et ainsi augmenter la diversité (corrige un peu le bug du dessus en quelque sorte).
- Le calcule des nouveaux poids se fait maintenant correctement. Je commence donc à faire la mutation.

2 mai 2017
- Création de la méthode mutation qui permettra de faire des modifications aléatoires sur chaque génome.
- Recherche sur comment faire de la mutation, plusieurs sources intéressantes trouvées
  - https://www.burakkanber.com/blog/machine-learning-genetic-algorithms-part-1-javascript/
  - https://www.tutorialspoint.com/genetic_algorithms/index.htm
- Modification du poster en vue de la reddition intermédiaire
- La méthode mutation est terminée. Elle s'occupe de récupérer tous les poids de chaque réseau de neurones, définir un nombre de neurones à modifier, puis effectué une modification dessus.
- Changement du calcul du fitness. Après avoir quelque peu réfléchi, je pense que le fait de lui donner son score comme fitness n'est définitivement pas une bonne idée. Ce que l'on souhaite, c'est qu'il apprenne à sauter le plus d'obstacle, pas à aller le plus loin possible, bien que les deux soit lié. Le problème est qu'il se peut qu'aucun obstacle soit généré pendant un long moment et le fitness de ce génome pourrait être plus grand qu'un autre qui aurait passé plus d'obstacle.
- Je pense changer la méthode de mutation par une autre méthode permettant d'appliquer un taux de mutation sur chaque neurone. Actuellement le nombre de neu-

rones a modifié ainsi que les index à modifier sont tirés aléatoirement et il se peut que certains neurones soit tout le temps omis. Je souhaite donc faire en sorte que le programme passe sur chaque neurone et selon un taux, effectue une modification dessus. De cette façon, tous les neurones ont leurs chances. (je ne pense pas que cela change grand-chose, mais je trouve cela intéressant)

- Ajout d'un timer dans les infos affichées pour savoir depuis combien de temps le logiciel tourne.
- Plusieurs problèmes rencontrés. Premièrement, l'application ne fonctionne pas. J'ai pu remarquer plusieurs choses. Tout d'abord, lorsqu'une génération contient que des enfants qui font la même chose (ex : s'accroupir h24), les enfants feront la même chose. Ce n'est pas vraiment un problème puisque c'est logique, mais le problème survient du fait qu'après un certain nombre de générations, cela n'ait pas changé. Grâce à la mutation, cela ne devrait pas arrivé, il doit donc y avoir un problème, pourtant les poids ont bien l'air d'être modifiés. Cela peut venir du fait que j'ai des poids négatifs. Cela m'a fait penser à quelque chose, il faut que je change le cross over et la mutation pour créer des enfants qui n'ont aucun rapport avec leurs parents pour éviter de rester bloqué trop longtemps. Il faut également que je corrige le problème cité au-dessus.
- Chose à faire :
  - Création d'enfant non lié au parents pour chaque génération
  - correction du bug de poids ? (voir pourquoi le réseau peut rester bloqué)
- Lecture d'article sur les algorithmes génétiquess.
- Après environ 30 générations, il semblerait que l'IA ait réussi à sortir de cette boucle, il faudra que j'effectue des tests.
- J'ai effectué quelque test sur le jeu en changeant la variable FPS pour voir comment allait se comporter le jeu et il se trouve que je ne pourrai pas l'utiliser. Tous les éléments se comportent normalement sauf le T-Rex qui a une physique de saut complètement cassé. L'inertie et la gravité on l'air de poser quelques problèmes, un ami a alors eu l'idée d'utiliser le logiciel "cheatEngine" pour augmenter la vitesse de google chrome, mais pour l'instant rien de concluant, je verrai cela lorsque tout sera fini.

3 mai 2017

- Modification du code pour ne plus avoir de poids négatif.
- Après avoir effectué quelque recherche, il semblerait que les poids négatifs ne soient pas un problème. Je remets donc les choses comme avant
- Modification du taux de variation des poids. Après quelques recherches, la variation est quelque chose qui est propre à chaque algorithme et avoir un taux bas permet de varier légèrement mais de possiblement rester bloqué dans un optimum local, alors qu'un taux trop élevé peut empêcher la population de converger vers une solution optimale, mais permet de chercher "plus loin". Son but principal étant d'empêcher l'IA de rester bloqué dans un optimum local et d'explorer plus en profondeur pour peut-être trouver une nouvelle solution au problème.
- Modification du code pour ajouter la création de génome aléatoire (non lié au parent)
- Ajout de la création des génomes aléatoires est terminé. Pour chaque génération je crée : le nombre de parents sélectionné divisé par deux.

- L'application n'a toujours pas l'air de fonctionner, je vais donc devoir vérifier un à un chaque objet et méthode. En attendant je laisse tourner le jeu en fond.
- J'ai réduit le nombre d'output de 3 à 2 (enlever le fait de ne pas faire d'action) pour voir si cela peut aider.
- Correction d'un bug qui augmentait le fitness de 1 après le premier génome.
- Je viens de trouver un bug dans la sélection des meilleurs genomes. Si un réseau obtient un fitness de 2, un autre de 1 et tous les autres de 0, il y aura plusieurs fois le réseau à 2 de fitness qui sera sélectionné.
- Apparemment le problème est plus complexe que ce que je pensais. Le problème vient d'un ancien "slice" que j'avais fait et il s'agit certainement du même problème que j'avais pu rencontrer précédemment avec cette fonction.
- Un des problèmes est celui dont je pensais, le slice ne crée pas une copie de mon objet, il va donc falloir que je me débrouille pour le faire moi-même. Le second problème, que je réglerai après, est que les réseaux de neurones déjà sélectionné, peuvent l'être à nouveau.
- Après avoir passé 3 ans à comprendre pourquoi le slice ne faisait pas ce que je souhaitais alors que tout sur internet dit que cela devrait marcher, il s'agit en fait simplement qu'il existe deux fonctions qui ont quasiment le même nom. La fonction slice, qui permet d'extraire une partie d'un tableau sans le modifier et la fonction splice qui fait la même chose mais enlève les index du tableau.
- La méthode de sélection est maintenant (normalement) complètement déboguer. Je continue donc à chercher.
- Les problèmes rencontrés peuvent venir de la méthode de crossover. Peut-être que du crossover uniforme résoudrait tout. Je garde ça de coter pour l'instant.
- Je suis en train de faire des tests en utilisant seulement 1 neurone en entrée (la distance) pour voir s'il s'en sort mieux avec moins d'informations.
- J'ai ajouté la décision que prend le T-Rex dans la partie "Network infos", cela me permet entre autres de débuguer plus facilement. Je me rends d'ailleurs compte que cette valeur ne monte que très rarement. Un réseau de neurones entraîné devrait normalement avoir cette valeur qui augmente plus il approche d'un obstacle, le problème (peu être pas ?) est que les 9/10 du temps cette valeur diminue, cela veut peut-être dire que j'ai fait une erreur de calcul quelque part.
- M.Marechal est passé voir le déroulement de chaque projet et après lui avoir expliqué le mien ainsi que les soucis que je rencontre actuellement, il a été proposé d'arrêter de faire de la programmation pour continuer la rédaction de la documentation technique. C'est une très bonne idée puisque lorsque je vais devoir expliquer tous les principes utilisés, je peux me rendre compte d'une erreur que j'ai faite et surtout de me changer les idées.

4 mai 2017

- Comme j'ai pu le dire hier, aujourd'hui je vais essayer de ne pas coder pour préparer la reddition intermédiaire et peut-être avancer la documentation technique. Si je n'ai pas envie de faire de la doc technique j'irai certainement lire des articles en rapport avec mon travail pour voir si quelque chose m'a échappé.
- M.Wanner est passé pour voir l'avancement du projet. Je lui ai parlé du fait que j'avais des problèmes avec l'IA et je lui ai montré le poster que j'ai créé. Le poster a l'air de lui plaire, quant au problème, je lui ai dit que j'en avais déjà parlé avec

M.Marechal et que le fait de faire une pause et de continuer la doc peut aider.

- Restructuration de la documentation technique. J'ai divisé différemment les parties de l'analyse fonctionnelle pour avoir quelque chose de plus logique.
- Début de la rédaction de l'analyse organique. Je commence par créer la structure.
- Rédaction de la classe "activationFunction" et un début de la classe "neuron"
- Création d'une carte du site pour montrer comment accéder à chaque partie et quelles sont les parties.
- Rédaction de la classe "Connection"
- J'ai eu l'idée de rajouter un glossaire à la fin du document pour expliquer plusieurs mots techniques. La section est créée pour ne pas oublier l'idée, mais se sera certainement fait a la fin.
- En relisant ma documentation je me suis rendu compte de quelque chose. En regarder les fonctions d'activation j'ai vu que la range de tanh était de -5 à 5 alors que sigmoid était de 0 à 1. Certains articles que j'ai lus utilisaient tanh alors que d'autres utilisaient sigmoid, en codant j'ai donc peut-être fait une erreur en me mélangeant les pinceaux.
- Avant de retourner sur le code pour faire des tests et vérification je vais finir de faire l'analyse fonctionnelle. J'ai décidé d'ajouter certaines choses dedans.

5 mai 2017

- Comme hier, aujourd'hui je vais continuer la doc et surtout corriger les fautes d'orthographe du journal de bord ainsi que de la doc technique.
- Fin de la partie que je souhaitais ajouter hier dans l'analyse fonctionnelle.
- Fin de la correction de la documentation technique.
- Début de la correction du journal de bord
- Fin de la correction du journal de bord

8 mai 2017

- J'ai réfléchi ce week-end et je pense avoir trouvé un problème dans mon code. Lors de la sélection, je compare le fitness de chaque réseau pour trouver le meilleur et pour éviter d'en avoir aucun en sortie j'ai défini la valeur de la variable qui me sert à garder en mémoire le meilleur fitness comme étant égale à -1. Grâce à cela, même si tous les réseaux avaient un fitness de 0 j'en aurais en sortie. Je pense que cela est une erreur de ma part, je vais donc remplacer cette valeur par 0 et faire en sorte que si aucun réseau n'a été sélectionné alors je les génère tous aléatoirement. Je vais expliquer pourquoi, je pense, que cela est une erreur. Lorsque je sélectionne des parents pour les faire se reproduire c'est parce qu'ils sont censés être meilleur que les autres, alors que ce n'est pas le cas. Actuellement si tous les réseaux ont un fitness de 0, alors le premier sera sélectionné et cela sans raison particulière. Cela peut énormément ralentir l'avancement puisqu'il est possible que le réseau parte complètement dans une mauvaise direction et ne jamais trouver une "bonne" solution.
- Je mettrai en place ce que j'ai dit au-dessus plus tard. Pour l'instant je vais faire l'affichage sous forme de dessin du réseau de neurones juste en-dessous de celui-ci.
- Après réflection, si je n'ai qu'un seul parent en sortie, je suis embêté. Il faut donc que je trouve un moyen.
- calcul de la taille d'un neurone. Voici la démarche : je regarde quel est le nombre max de neurone sur un layer et le stocke de coté. je divise la height du canvas par

le nombre max de neurone et la width par le nombre de layer. Le résultat le plus petit des deux est celui qui va me permettre de définir la taille d'un neurone (pour avoir un rond). je viens retirer une valeur donnée qui est la distance entre chaque neurone d'un même layer que je souhaite avoir.

9 mai 2017

- J'ai changé un peu l'algorithme du dessin pour que le dessin prenne tout l'espace disponible. (Ajout d'une variable "spaceFromBorder" qui définit les marges)
- Il faut maintenant que je place les poids au bon endroit en que je trouve comment faire pour rendre le tout "lisible"
- En ayant réfléchi hier soir, je me suis dit qu'ajouter les valeurs de sortie de chaque neurone dans ceux-ci pourrait être une bonne idée.
- Je viens de me rendre compte d'un problème. La valeur de décision du T-Rex est NaN, j'ai dû modifier quelque chose sans faire exprès. Il faudra que je regarde après. (erreur que je n'avais pas hier)
- Visite de M.Wanner. Il a vu l'IA faite pour flappy bird (dont je parle dans la doc technique) et souhaite que j'implémente quelque chose de similaire. Instancier plusieurs T-Rex dans le même jeu pour accélérer le processus. Cela demandera beaucoup de modifications. Malheureusement j'ai oublié de lui dire qu'avec la simulation de pression de touche il est impossible de dire "celui-ci uniquement". Il faut donc que je voie avec lui s'il souhaite que je modifie complètement le jeu pour faire cela. Dans mon cahier des charges il est dit que je ne toucherai pas au code du jeu.
- Changement du calcul de l'espacement entre les neurones
- Ajout des poids dans la visualisation. Le calcul du positionnement des poids m'a pris beaucoup de temps et je ne suis pas vraiment fan du résultat donc il va peut-être changer.
- Ajout de la "outputValue" au milieu de chaque neurone.
- J'ai un problème avec le poids des bias. Des chiffres s'affichent dans mon interface, mais lorsque je vais regarder à la main dans mon objet il est vide.
- Le problème vient apparemment de la récupération des poids.

10 mai 2017

- Le problème vient effectivement de la récupération et la modification des poids.
- Changement des méthodes "setWeight" et "getWeight" pour qu'elles fonctionnent comme voulu.
- Le T-Rex fonctionne et arrive à apprendre. J'ai maintenant une première version fonctionnelle que je pourrais utiliser dans ma démonstration ce soir.
- Changement de la couleur des bias pour rendre cela plus visuelle et bien se rendre compte qu'ils ne font pas "directement" partie du réseau de neurones
- J'ai toujours plusieurs choses à faire
  - Changer la méthode de sélection qui peut améliorer la vitesse d'apprentissage.
  - Ajouter des légendes sur la visualisation en live.
  - Trouver un moyen d'accélérer le processus d'apprentissage.
  - Afficher un graph qui va permettre de voir le fitness moyen de chaque génération
- Préparation pour la visite de ce soir.

11 mai 2017

- Modification de l'interface pour y ajouter un graphique qui affichera le fitness

moyen de chaque génération au fur et à mesure, permettant ainsi d'avoir un suivi de l'évolution.
- Ajout d'une méthode qui retourne le fitness moyen d'une génération
- Utilisation de la bibliothèque "Chartjs" pour dessiner facilement des graphiques
- Modification de l'interface pour faire en sorte que la taille du canvas soit dynamique.
- Création d'un eventListener qui va modifier la taille du canvas lorsque la taille de la fenêtre change.
- Modification du css du jeu du T-Rex pour faire en sorte qu'il soit complètement responsive.
- Après réflexion, l'ajout de légende sur la visualisation en live est chiante a mettre en place et n'apporte pas grand chose.
- Avant de m'occuper de trouver une méthode pour accélérer le jeu je vais faire le seconde méthode de sélection.
- En faisant des recherches sur la sélection "roulette wheel selection" je suis en train de me demander si c'est normal que des générations suivantes soient plus "bêtes" que des anciennes. Logiquement j'ai envie de dire que ça l'est puisqu'en mélangeant 2 parents on peut obtenir un résultat moins bon, mais cela me semble bizarre, je vais donc faire un peu de recherche de ce côté.
- Je viens de voir qu'il existerait une méthode de reproduction qui s'appelle "elitism". Elle consiste simplement, en plus de faire les étapes normales, à garder les parents dans la prochaine génération, cela permet de ne jamais avoir un résultat moins bon, mais peut empêcher de trouver la solution optimale.
- Il existe également une alternative au crossover qui est de faire du crossover avec un taux, comme pour la mutation. Cela n'est pas forcément bénéfique, mais peut être une alternative à "l'elitism".
- Pour mettre en place "roulette wheel selection" il va falloir que je normalise les fitness. Actuellement, les fitness peuvent entre 0 et infini. Le problème est que la méthode que je souhaite mettre en place sélectionne les parents au hasard, plus un fitness est grand plus il a de chance d'être sélectionné. Il faut donc que je transforme mes fitness en pourcentage.

12 mai 2017

- Aujourd'hui je vais essayer d'implémenter toutes les méthodes dont j'ai pu parler les jours précédents. Cela peut me permettre d'améliorer les performances.
- Après quelques recherches, il semblerait que la façon dont je calcule mon fitness n'est pas vraiment très bien. Très souvent, les premières générations ont toutes un fitness de 0, ce qui ne devrait pas arriver. Dans mon cas le fitness augmentera seulement lorsque quelqu'un aura réussi à sauter le cactus ce qui permet de résoudre mon problème. Il faut que je fasse en sorte que 2 T-Rex qui n'ont sauté aucun cactus puisse avoir un fitness supérieur à 0. Il faudrait que je trouve un moyen d'évaluation qui montre que même si un T-Rex n'arrive pas à sauter un obstacle, il s'est amélioré. C'est ce problème qui fait que mon T-Rex peut prendre beaucoup de temps a sauté un obstacle au début. Le problème est que je ne vois absolument pas ce que je pourrai évaluer en dehors du nombre d'obstacles sautés.
- Je ne vais finalement pas pouvoir faire ce que je souhaitai faire aujourd'hui, ou en tout cas pas ce matin. Normalement nous aurions dû avoir des visites cette

après-midi, mais il s'avère que c'est toute la mâtiné.
- Fin de la visite des élèves. Je vais pouvoir continuer.
- Chose à faire dans l'ordre :
  - Changer fitness function
  - Implémenter "roulette wheel selection"
  - Modifier jeu du T-Rex pour accélérer le processus
  - Implémenter "elitism", "crossover" avec taux, etc..
- Depuis ce matin je cherche un moyen de changer le fitness mais je ne trouve rien. Je me suis dit que je pourrais faire un rapport entre le nombre de sauts effectués et le nombre d'obstacles passés. Ensuite je me suis dit que je n'avais qu'a donné une valeur fictive dans le réseau de neurones et de regarder le résultat (très mauvaise idée je pense).
- Je pense que je ne vais pas réussir à trouver la bonne façon de calculer le fitness (si elle existe). Il faut pourtant que je trouve un moyen de montrer que même si un réseau obtient un score de 0, il peut être meilleur qu'un autre. Les critères que j'ai trouvé pour l'instant qui permettrait d'évaluer ça serait : Il arrive à sauter, Il saute au bon moment. Il faut vraiment que je trouve un moyen, cela peut vraiment énormément réduire le temps besoin pour avoir un T-Rex "viable".
- Après avoir longuement réfléchi, je pense qu'il n'y a aucun moyen. Pour faire en sorte que la "roulette wheel selection" fonctionne, je vais faire en sorte d'ajouter 1 à chaque fitness lorsque je fais la sélection (leur vrai fitness reste inchangé, bien que cela ne change rien). Je pourrai toujours changer cela après.
- La mise en place de "roulette wheel selection" est faite. Je vais maintenant faire quelque test pour vérifier qu'elle fonctionne correctement et me sort de bons résultats.
- Le fait d'avoir +1 en fitness était beaucoup trop gros, je l'ai donc réduit a +0.1. Je fais des tests pour voir le fonctionnement.
- Je crois qu'il y a quelques soucis avec la "roulette wheel selection", je regarderai ça ce week-end ou lundi.

15 mai 2017
- Il y avait bien un problème dans la "roulette wheel selection", cela venait de plusieurs choses. Premièrement, la méthode utilisée pour tirer un nombre aléatoire était fausse. Je l'arrondissais alors qu'il ne fallait pas. Ensuite, La valeur qui me sert à savoir si j'ai dépassé le seuil requis pour sélectionner un génome était une ancienne variable qui contenait déjà un nombre. Je crois que c'est tout.
- Je vais maintenant essayer de modifier le jeu pour ajouter plusieurs T-Rex
- Je suis en train de modifier le jeu du T-Rex et c'est plutôt complexe. Tout part de tous les côtés et tout n'est pas organisé comme je l'aurai pensé/fais, du coup, certaines choses sont relativement compliquées à faire.
- Dû au fait que je ne pensais pas faire ça, la structure n'est pas adapté et cela donne n'importe quoi. Tout part de tous les côtés sans queux ni tète. Le projet devient vraiment bordélique.
- Bug trouvé
  - Lorsqu'un trex meurt, les actions des autres sont étranges
  - le score augmente bien trop rapidement
  - J'ai l'impression que les réseaux de neurones ne sont pas liés au trex (échange

de réseau de neurones ?)
- Lorsqu'un trex est en l'air et qu'un autre meurt, il y a un bug de physique. Le trex va planer.

16 mai 2017

- Je vais essayer de corriger tous les bugs trouvés hier (et corriger ceux que je n'ai pas encore trouvés)
- Une fois cela fait, il faudrait que je "nettoie" un peu la structure du code et le rende plus "propre". En parallèle, il faudra également que je test de lancer le trex avec toutes les valeurs d'entrées (réseau final [4,5,1]).
- Correction du bug du score.
- Changement de l'affichage des réseaux de neurones. J'ai ajouté un "select" pour permettre de sélectionner le réseau de neurones que l'on souhaite afficher. Cela me permet de régler le souci de savoir lequel je devais afficher et en plus, cela laisse le choix à l'utilisateur.
- C'est bien ce que je craignais, l'implémentation de plusieurs trex est impossible. Le problème est que lorsqu'un trex effectue une action, il peut modifier les variables et le comportement du monde, rendant ainsi le jeu injouable pour les autres trex (peut changer gravité, vitesse de chute et autres)
- J'abandonne donc l'idée d'instancier plusieurs trex dans le meme "jeu". En attendant de trouver une idée qui fonctionne et me permet d'accélérer le processus, je vais faire en sorte que l'on puisse changer de jeu pour passer sur flappy bird (ou autres) et commencer à "nettoyer" le code ainsi que modifier sa structure.
- Je vais finalement nettoyer le code avant d'implementer flappy bird.
- J'ai modifié un peu le script du trex pour faire en sorte que le jeu n'ait pas besoin d'avoir le focus pour "jouer"
- Correction du bouton "play", "stop". Ils fonctionnent maintenant normalement.
- Utilisation du design pattern "strategy". Je ne savais pas, mais lorsque j'ai créé la classe "activationFunction" j'ai en fait utilisé le design pattern "strategy". Je vais maintenant l'utiliser à nouveau pour permettre de changer la sélection, crossover et mutation.

17 mai 2017

- Je crois que j'ai fini de restructurer le code. 3 classes ont été ajouté : sélection, crossover, mutation. Elles permettent chacune d'instancier une méthode de leur "type" et de faire le processus demandé dans l'étape. Quelques optimisations ont été faite dans le "main" ainsi que "generation".
- Je vais maintenant essayer de trouver un jeu que je peux implémenter dans mon application.
- J'ai récupérer le code de quelqu'un sur github qui a reproduit le jeu.
- Modification du code du jeu pour être réutilisé dans mon application.
- Le jeu a été très mal fait (je trouve), il faut donc que je modifie quasiment tout le code pour le rendre utilisable.
- Avec la chaleur qu'il fait dans la salle, je peine à travailler et me concentrer. Néanmoins, j'ai bientôt fini de "recoder" le jeu du flappy bird, j'ai juste quelques problèmes avec les images du jeu.
- Je viens de finir le flappy bird, il faut maintenant que je fasse son intégration dans l'application.

18 mai 2017

- Je continue l'implémentation du jeu flappy bird dans l'application.
- Ajout d'un bouton sur la vue pour permettre de changer de jeu. J'ai du modifier le code du trex ainsi que le code du flappy bird.
- Le jeu du flappy bird "fonctionne". le réseau de neurones peut effectuer des actions et le jeu se comporte normalement. Néanmoins, il y a encore quelque petit souci qu'il faut que je corrige. Je pense que je calcule mal le fitness et les valeurs que je donne au réseau de neurones ne sont peut-être pas les plus utiles.
- modification du jeu flappy bird pour me permettre de faire pause.
- Quelques changements effectués sur le réseau de neurones pour le rendre plus "fort", mais rien de très concluant pour l'instant.
- Les deux jeux peuvent être interchangés sans problème. Lorsque l'on change de jeu, l'ia se met en pause, le graph se vide et l'autre jeu s'affiche
- Il faut encore que je modifie les valeurs affichées dans la zone anciennement appelée "trex values" pour qu'elles changent en fonction du jeu.
- Pour l'instant le script "main" est un peu bordélique et demande à être changé. Cela est dû au fait que je dois faire des tests pour chaque action pour savoir sur quel jeu on se trouve actuellement. Je pense changer cela plus tard en ajoutant un objet, comme par exemple "gameManager" qui permettrait de faire tout cela correctement.
- Il faudrait que je mette en place une méthode d'élitisme également.

19 mai 2017

- Une fois que l'implémentation de flappy bird est fait, il faudrait que j'avance la documentation technique
- Aujourd'hui il faut impérativement que je fasse l'abstract pour pouvoir le donner au prof d'anglais.
- Je pense que la seule chose qui fait que le flappy bird ne fonctionne pas très bien est que les valeurs que je lui donne en entrée ne sont pas les bonnes/plus utiles. En regardant le code de la personne qui a fait le jeu du flappy bird, je me suis rendu compte qu'il utilisait un réseau de neurones [1,1,1], alors que j'utilise un réseau [3,3,1]. Le problème vient certainement de là. Je me suis demandé comment cette personne avait fait pour n'avoir qu'à envoyer une seule donnée dans le jeu et j'ai demandé à un camarade comment il ferait lui. Il n'avait pas vraiment d'idée, mais ce qu'il a dit m'a fait penser à quelque chose. Je n'ai pas regardé le code de la personne car ce n'est pas le but et je ne trouve pas ça drôle. Je préfère trouver seul (ou en m'aidant de mes camarades). L'idée que j'ai eue est en fait d'envoyer la différence de distance entre le flappy bird et l'espace entre les tuyaux dans lequel il doit passer. Je n'ai pas encore vraiment réfléchi à si cette idée est bonne ou non, ce que je vais donc faire maintenant.
- Je pense que ma méthode ne fonctionnera pas, le problème est qu'il n'a aucun moyen de savoir s'il se trouve en dessous ou en dessus du trou. Je pense donc qu'il lui faut en tout cas 2 inputs.
- Je viens d'avoir une idée pour réduire de 1 mon nombre d'input. Actuellement, je donne la position y de l'oiseau, le point y du début du trou et le point y de fin du trou au réseau de neurones. Je ne pense pas que l'oiseau ait besoin du point de fin, il devrait pouvoir se débrouiller seulement avec un seul point (le haut ou le bas,

peu importe), cela permet de réduire le nombre de valeur que le réseau va devoir apprendre et peut être de faire en sorte qu'il fonctionne.
- C'était effectivement une bonne idée. Le jeu est actuellement en train de tourner et il est a un score de 30000, ce qui est énorme. Je pense qu'il ne s'arrêtera plus, et cela seulement après 5 générations.
- Je vais maintenant essayer de nettoyer le code et trouver un moyen de gérer le multi jeu de façon efficace et propre.
- Avant de faire cela, je vais rédiger l'abstract.
- J'ai rédigé mon abstract et le résumé.

22 mai 2017

- Création de la classe gameManager qui permet d'ajouter plus facilement des jeux dans l'application.
- Étant donné qu'il ne reste plus que deux semaines, je vais essayer de finir tout le code aujourd'hui pour faire que de la documentation après.
- La classe gameManager est finis et complètement implémentée. Pour l'instant elle ne permet pas d'ajouter un jeu en 2 secondes, mais facilite grandement l'ajout. Je vais la laisser comme ça pour l'instant et peut-être revenir dessus plus tard.
- Les fenêtres d'exportation et d'importation de réseau de neurones sont maintenant terminées/liées. Beaucoup de fonction avait déjà été faite, mais rien n'était lié. Modification de la fenêtre d'exportation pour pouvoir sélectionner le meilleur réseau de neurones (recherche) ou bien sélectionner à la main celui que l'on souhaite exporter. Tout cela se fait soit en format txt sous en créant un fichier qui va être téléchargé par l'utilisateur. La fenêtre d'importation permet d'importer depuis un fichier ou depuis du texte.
- Les valeurs du jeu (anciennement "T-Rex values") ont été retirées car inutile et me demandent du temps à mettre en place pour sélection de jeu.
- La structure de mon application devient étrange, il faudra que je voie tout cela demain lorsque je m'occuperai de la documentation.
- Après quelques tests, tout a l'air de fonctionner normalement. Il va encore falloir que je fasse la fenêtre "about" et "contact", mais cela ne devrait pas prendre beaucoup de temps.
- La fenêtre "about" a été faite, je ne sais pas si je vais faire la fenêtre "contact"
- Il semblerai que j'ai quelque soucis avec l'exportation et l'importation.
- J'ai corrigé un bug de l'importation. Lors de l'importation, tout est réinitialisé sauf le jeu, du coup l'IA commençait pas au début du jeu.
- Je crois que la fonction qui permet d'exporter le meilleur réseau ne marche pas. Il faut que j'effectue quelque test.

23 mai 2017

- il y a un bug avec le graph. Je ne sais pas si c'est à cause de l'importation quand le jeu est lancé, il faut que je fasse des tests.
- Le problème doit effectivement venir de l'importation. Je vérifierai cela plus tard.
- J'ai refait la partie sur l'interface de l'application en la mettant à jour.
- J'ai pris toute la journée pour faire le diagramme de classes et je ne l'ai pas encore finis. Il ne me manque pas beaucoup à faire.

24 mai 2017

- Fin du diagramme de classe. Je vais faire la rédaction de l'analyse organique.

- Finalement, j'ai fait le tableau d'apport personnel. Je m'occupe maintenant de l'organique. Il va falloir que je commente le code en même temps. Après ça il ne me restera plus qu'à rédiger les tests, la conclusion et ajouter un dictionnaire.
- Étant donné que mon projet est sur github, tout le monde y a accès et certains camarades se sont amusé a le lancer sur leur poste. Une des personnes avait un flappy bird en génération 110+ qui n'avait pas fait un score de plus de 400 ce qui est extrêmement bizarre. Cela peut simplement venir du fait que lorsque la page n'est pas visible, chrome met les pages en arrêt. Cela a peut-être causé un souci avec l'ia.
- J'ai commencé l'analyse organique et finis la "classe main". Je vais attendre avant de continuer, car il y a quelque bug qu'il faut que je corrige.

26 mai 2017

- J'ai pu corriger quelques problèmes que j'ai trouvé dans l'application.
- J'ai essayé de reproduire le problème que mon camarade a eu pour voir ce que je peux faire. Pour cela j'ai essayé d'avoir une première génération qui n'arrive pas à planer pour avoir les oiseaux les plus "débiles" possible et ainsi être dans le pire cas possible. Actuellement, l'application est à la génération 40 et obtient un score de 372 max. C'est en fait normal, bien que dommage, que l'application prennent beaucoup de temps à avoir un gros fitness dans certains cas. Le but d'un réseau de neurones n'est pas de trouver une solution rapidement, mais bien de trouver une solution. En général il tourne pendant des jours pour obtenir un résultat. Je ne pense donc pas que cela soit un problème et je vais le laisser tourner en fond. Pour obtenir ce cas précis, j'ai dû actualiser la page beaucoup de fois, cela reste donc très rare.
- J'ai modifié une fonctionnalité pour quel soit plus logique et utile. Lorsque l'on souhaitait exporter le meilleur réseau de neurones depuis la pop-up d'exportation, cela exportait le réseau de neurones ayant le meilleur fitness dans la génération actuelle et non pas le meilleur depuis tous les temps. J'ai donc modifié 2 classes pour stocker en permanence le meilleur réseau de neurones et ainsi pouvoir l'exporter.
- Le score du flappy bird sur lequel je fais des tests continus d'augmenter au fil des générations. Il est maintenant à un score de 522 après 74 générations.
- Correction d'un bug lors de l'importation d'un réseau de neurones. La méthode que j'utilisais pour réinitialiser le graphique causait quelques problèmes. J'ai donc changé cette méthode et il n'y a plus aucun problème.
- Pour l'instant, tous les bugs trouvés sont corrigés. Je vais donc continuer la documentation.
- Après quelques heures de tests, le flappy est toujours bloqué à un score de 522. Il pourrait s'en sortir un jour, mais je me suis demandé pourquoi il se comportait comme ça. Le problème est en fait simple. Il se trouve que les poids reliés au neurone représentant la position y du flappy sont beaucoup plus élevés que les poids pour la position du tuyau (3 pour position Y et 0.1 pour position tuyau). De ce fait, il prête plus d'attention à sa position qu'a celle des tuyaux. Le problème est qu'étant donné que les poids ne peuvent être modifié que de 0.2, cela peut prendre beaucoup de temps. Pour faire simple, le flappy a mal appris et il va prendre pas mal de temps pour apprendre correctement.
- Pour essayer de régler le problème je vais essayer de changer le nombre de neurones

ou bien la range de modification des poids.
- Après avoir cherché à régler ce problème, je suis tombé sur quelque chose d'intéressant. Actuellement, les poids sont modifiés en ajoutant un nombre aléatoire tiré entre -0.2 et 0.2. La solution serait de faire en sorte que la modification se fasse selon un pourcentage de la valeur du poids à modifier.
- Correction d'un bug lors de la sélection du meilleur réseau de neurones. Il ne devrait plus y avoir de souci de ce côté normalement.
- Depuis que j'ai mis en place le gameManager j'ai l'impression que le réseau de neurones apprend moins bien, en faisant des changements je me suis peut-être emmêlé les pinceaux et commis quelque erreur, il faut que je fasse quelque test pour vérifier que tout fonctionne normalement.

29 mai 2017
- Avant de faire la documentation technique je vais finir de commenter tout le code. Je vais faire en sorte que si je le souhaite plus tard, je puisse l'exporter. Utilisation de JSDoc.
- L'ajout de commentaire dans le code est terminé, je vais maintenant commencer l'analyse organique
- J'ai fait 1/4 de l'analyse organique, avec la chaleur qu'il fait, ça devient compliqué de travailler. Je finirai certainement demain ou après-demain

30 mai 2017
- Continuation de l'analyse organique.

31 mai 2017
- Continuation de l'analyse organique.
- L'analyse organique est quasiment fini, il faudra que je la relise après et que je corrige les fautes d'orthographes.

1 juin 2017
- J'ai terminé la documentation technique, il faudra que je la relise après, mais je vais maintenant m'occuper de faire les tests unitaires de mon application.
- J'ai terminé de faire les units tests. J'ai utilisé une bibliothèque qui se nomme QUnit. Maintenant que c'est fait, je vais rédiger la partie teste dans la doc.
- Avant de rédiger la partie "tests" je vais m'occuper de faire la conclusion. Il faut encore que je réfléchisse à comment je vais faire la partie "tests".

6 juin 2017
- Modification de la forme du lexique. Avec l'ancienne méthode j'avais quelques soucis d'affichage, j'ai donc décidé d'utiliser "description".
- Fin du lexique, il ne me manque plus qu'à corriger ma documentation.
- Une fois la correction finie, il faudra que je fasse 2-3 trucs sur l'application. Principalement pour le trex, le flappy bird est terminé.
- Je suis en train de corriger la documentation et je me rends compte qu'il y a plein de phrases dans lesquelles ils manquent des mots, certainement dû à la fatigue.

7 juin 2017
- Continuation de la correction de la documentation
- En même temps que je corrige la documentation, je fais des tests sur le trex.
- J'ai rajouté une partie dans les tests pour expliquer comment faire des tests en machine learning
- J'ai finalement décidé de retirer cette partie.

- J'ai bientôt fini de checker tout le document une fois, il ne reste plus que la conclusion. Après ça, je ferai à nouveau une vérification.

8 juin 2017

- Correction de la documentation terminé, je vais la lire à nouveau.
- Ajout de terme dans le lexique.
- Modification de quelque phrases de la documentation.
- Correction de la documentation.

9 juin 2017

- Je continu de lire et relire ma documentation pour vérifier.
- Ajout d'un entête et d'un pied de page.

11 juin 2017

- Relecture de la documentation technique et correction.
- Correction du journal de bord

11 juin 2017

- Impression de la documentation