CFPT Informatique - Travail de semestre

Omni Pong

Rapport de Projet

Raphaël Lopes | Yannis Perrin

## 

## 

21 Novembre 2019

## 

[**Introduction**](#_fzitywg3o4r2) **2**

[**Cahier des charges Omni Pong**](#_oyvotk9w9587) **3**

[Sujet](#_49qh69qhee0m) 3

[But](#_ul8leeac3ix6) 3

[Spécifications](#_wm1u3cgqw9y7) 3

[Restrictions](#_my47q5yl7430) 3

[Livrables](#_t55x5bjjuedy) 3

[Reddition](#_hvycfq60mxkw) 3

[Planification](#_hm8rxdut7xkv) 4

[**Manuel d’utilisateur**](#_b9nuec5koqw7) **5**

[Contrôle](#_t4lwympq9f9i) 5

[Version ordinateur](#_ea8u3vu301k1) 5

[Version borne d’arcade](#_tpgo2cjlg4go) 5

[Lancer le jeu](#_f8wjz0yx59ro) 5

[Version ordinateur.](#_j3zfx47t2kh) 5

[Version borne d’arcade](#_7ynoy6mq77sb) 5

[**Analyse fonctionnelle**](#_14v1pojkrtse) **6**

[Réseaux Neuronaux](#_zct1jxi2a12b) 6

[**Analyse organique**](#_pmzooyib8inn) **7**

[Le jeu](#_axvhby5s0tku) 7

[ML Agents](#_r0sc9a64s0bd) 9

[Entrainement de l’IA](#_217k5vi9wjig) 10

[Améliorations possibles](#_3mjpmux8olzn) 11

[Problèmes rencontrés](#_wvjy7c7dtmz5) 11

[**Protocole de test**](#_rc4osil0ndj6) **12**

[**Conclusion**](#_sl8hcjedvkgy) **12**

[Bilans personnels](#_kwv6txdvvoqy) 12

[Yannis](#_bg4diwu6c5lw) 12

[Travail accomplis](#_kj95xv54vtq0) 12

[Raphaël](#_kllp28pvq4b0) 12

[Travail accomplis](#_ahtuzwmdl2ei) 12

[**Bibliographie**](#_24x9ef3ws8a2) **12**

## 

# Introduction

Dans le cadre du premier travail de semestre de la deuxième année de formation de Technicien ES en informatique, nous avons choisi comme projet le développement d’un jeu Pong ou il serait possible d’affronter une IA[[1]](#footnote-1). Cette IA serait contrôlée par un réseau neuronal qui va évoluer grâce à un algorithme d’apprentissage. Le projet s'appelait “smart pong” mais par la suite nous avons changé le nom par “Omni Pong”,

L’intérêt de ce travail est de découvrir le Machine Learning et les réseaux neuronaux, afin d’apprendre ces nouvelles technologies en les implémentant dans ce jeu.  
Ces connaissances nous servirons possiblement dans de futurs projets de notre carrière professionnelle.

# 

# Cahier des charges Omni Pong

## Sujet

Création d’un jeu ressemblant à Pong, contrôlable par un 2ème joueur ou une IA évolutive devenant progressivement imbattable.

## But

* Création d’un jeu ressemblant à Pong
* Utiliser la borne d’arcade de l’école
* Construction d’un système d’apprentissage non (ou semi) supervisé remplaçant le 2ème joueur

## Spécifications

Le jeu est inspiré de Pong, il simule une partie de tennis de table en vue du dessus. La partie s’arrête lorsque l’un des 2 joueurs atteint le score de 10 points.

Notre version de ce jeu comportera deux modes :

* Joueur contre Joueur :

Chaque joueur contrôle une raquette pour renvoyer la balle de l’autre côté.

* Joueur contre l’IA :

L’IA remplace le 2ème Joueur et s’améliorera jusqu’au point où elle deviendra imbattable.

## Restrictions

* La méthode d’apprentissage de l’IA n’est pas définie au préalable Environnement
* Borne Arcade
* C#, Unity
* 2 PC avec Windows 10 x64
* Visual Studio 2017/2019

## Livrables

* Poster
* Documentation
* Journal de bord
* Code source et projet

## Reddition

* 03/10/2019 : rendu du poster
* 17/10/2019 : rapport intermédiaire
* 21/11/2019 : rapport finale
* 28/11/2019 : exposé oral

## Planification

## 

# Manuel d’utilisateur

Omni Pong est un jeu pong qui utilise le framework ML Agent (Machine Learning Agent) qui est une technologie proposée par Unity permettant de créer des réseaux neuronaux et y appliquer des algorithmes de Machine Learning. Le but du jeu est de renvoyer la balle avec la raquette en évitant qu'elle rentre dans notre goal.

## Contrôle

### Version ordinateur

Dans le menu

* ESC ou 0 : pour quitter l’application
* W ou Flèche du haut pour monter
* S ou Flèche du bas : pour descendre
* Enter ou 8 : pour valider la sélection

En jeu

* Joueur 1 :
  + W = monter la raquette
  + S = Descendre la raquette
* Joueur 2 :
  + Flèche du haut = monter la raquette
  + Flèche du bas = Descendre la raquette

### Version borne d’arcade

Dans le menu

* Bouton 0 : pour quitter l’application
* Joystick droit au gauche en haut : pour monter
* Joystick droit au gauche en bas : pour descendre
* Bouton 8 : pour valider la sélection

En jeu

* Joueur 1 :
  + Joystick droit = monter / Descendre la raquette
* Joueur 2 :
  + Joystick gauche= monter / Descendre la raquette

## Lancer le jeu

### Version ordinateur.

Il suffit de lancer le OmniPong.exe

### Version borne d’arcade

Sélectionner “Omni Pong” sur la borne d’arcade et appuyer sur le bouton 8

# 

# Analyse fonctionnelle

## Réseaux Neuronaux

Le projet étant de créer une intelligence artificielle contrôlée par un réseau neuronal, il faut d’abord savoir exactement ce que c’est.

Un réseau neuronal est un système qui imite le fonctionnement des neurones biologiques, il est souvent représenté de la manière suivante :



Les ronds dans les figures précédentes représentent les neurones du réseau et les lignes les liaisons entre eux. Cette architecture permet à chaque neurone d’associer chaque connexion à un poids qui le fera prendre plus ou moins compte le signal que cette liaison lui transmet. En plus de ces poids, chaque neurone ajoute à son calcul une valeur de partialité (“bias” en anglais) pour biaisés ses résultats. Ensuite, le neurone va calculer, à l’aide de sa fonction d’activation (“g” sur la figure 2) le résultat qu’il va passer aux neurones qui le suivent.

En informatique, ce système est très souvent couplé à un algorithme d’optimisation pour que le réseau puisse “apprendre” par lui-même. Pour “apprendre”, le réseau va modifier les poids de chaque neurone aléatoirement et, par des fonctions de probabilité ou autre va modifier ces poids petits à petits jusqu’à ce que l’IA fasse ce qu’on lui demande de faire, à la quasi perfection.

# 

# Analyse organique

## Le jeu

Au début d’une partie on place les bords du terrain de jeu par rapport à la caméra, ensuite, on place les raquettes sur leurs côtés respectifs. On place la balle au centre du terrain puis après un compte à rebours. Enfin la balle part dans une direction et un angle aléatoire (de 135° à 225° pour le joueur gauche et 315° à 45° pour le joueur droit). La balle va par la suite rebondir sur les murs ce qui va modifier sa direction vertical (Y), et dès qu’elle rebondira sur une raquette sa vitesse augmentera et repartira de l’autre côté en changeant sa direction horizontale(X).

Dès le début nous avons voulu implémentée une vue responsive, c’est à dire que peu importe la dimension de l’écran le jeu doit s’afficher avec les bords de chaque côté, Les raquettes à droit et à gauche de l’écran et enfin la balle au milieu. Nous avons ajouté à cela des collisions avec les murs joueurs avec la physique d’unity. Malheureusement, après des tests nous avons remarqué que le moteur physique d’unity n’arrivait pas à gérer les collisions correctement après un certain temps de jeu car la balle accélère à chaque rebonds sur une raquettes, c’est pourquoi nous avons décidé d’enlever les murs et tout simplement vérifier si la position X,Y De la balle, Pour le rebonds de la balle nous vérifions là où la balle rebondit sur la raquettes et appliquons un effet dessus. Par exemple, si une balle rebondit sur le point le plus bas d’une raquette alors elle va aller plus vers le bas.



## ML Agents

ML Agents, est un framework proposé par Unity pour créer des intelligences artificielles basées sur des réseaux neuronaux et permet de les intégrer facilement dans l’environnement Unity. Nous avons utilisé cette technologie pour créer l’intelligence artificielle de l’IA contre laquelle le joueur combattra.

Avant d’implémenter ce framework dans notre projet, nous avons d’abord installé python et TensorFlow car ils sont nécessaires pour que le framework puisse créer et entraîner des réseaux neuronaux. Les étapes d’installations que nous avons suivies pour mettre en place le framework sont disponibles sur leur Github :  
<https://github.com/Unity-Technologies/ml-agents>

Suite à l’installation du framework et des programmes nécessaires à son fonctionnement, nous avons mis en place le minimum requis pour créer une IA pour le joueur 2 et sommes arrivées au fonctionnement suivant (voir figure 5) :



Le code du jeu a été adapté pour que les contrôles de n’importe quel joueur puissent être automatisé dans le code. Ce fonctionnement a permis de très facilement raccorder les réponses du réseau neuronal aux contrôles du 2ème joueur.

Le réseau de neurones étant créer par TensorFlow, nous ne connaissons pas sa structure exacte. En revanche, nous connaissons les inputs et outputs de celui-ci :



Le jeu va donc envoyer à l’interface de ML Agents les 2 informations du jeu qui sont importantes au réseau neuronal, soit la position verticale de la balle pour les niveaux 1 et 2 ou la position prédite d’où la balle arrivera dans le but de la raquette en question (l’étape d’observation), récupérer les réponses du réseau et faire bouger la raquette en conséquence (l’étape d’action) et évaluer sa performance (l’étape de récompense).

### Entrainement de l’IA

Suite à l’intégration de ML Agents dans, il a fallu créer un réseau neuronal et l’entraîner. Pour cela, TensorFlow permet, grâce à un fichier de configuration, de créer un réseau neuronal et de définir les paramètres de son entraînement. Les paramètres que nous avons changés par rapport à la configuration de base sont :

* buffer\_size : Le nombre d’expériences[[2]](#footnote-2) sur lequel l’algorithme va se baser pour optimiser le modèle du réseau neuronal.
* batch\_size : Lors de l’optimisation, le nombre d’expériences qui seront prises par itération de l’algorithme d'optimisation (l'algorithme fonctionne en faisant plusieurs passages sur le modèle).
* hidden\_units : Le nombre de neurones dans chaque couches cachées.
* num\_layers : Le nombre de couches cachées du modèle.

Toute la documentation des paramètres disponibles sont définis à l'adresse :

<https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Training-PPO.md>

## Améliorations possibles

* Un système de Power Up (raquettes qui s’agrandissent / rapetissent, balle qui accélère, ralentis, etc... )
* Une meilleur IA avec plus d'input dans le réseau neuronale.
* Plus de temps pour l'entraînement de l’intelligence artificiel

## Problèmes rencontrés

Un des majeurs problèmes rencontrés a été l’IA. Le premier problème a été l’implémentation de celle-ci avec le jeu. En effet, celle-ci a requise plus de temps que prévu à cause de la mauvaise organisation de la documentation de la librairie et de notre méconnaissance du sujet. Cependant, nous avons tout de même réussi à produire un projet fonctionnel et intéressant.  
Ensuite, nous regrettons de ne pas avoir pu apprendre plus à propos des architectures de réseaux neuronaux car le temps des séances d’entraînement a pris plus de temps que prévu à réaliser. Ce qui nous a laissés peu de temps pour expérimentés avec. De plus, nous n’avons pas réussi à mettre en place certaines techniques prévues qui peuvent pallier ce problème. L’une d’entre elle accélère le temps de simulation pour entraîner plus rapidement le modèle mais cela s’est vite avéré impossible car notre jeu utilise des propriétés physiques qui, lors de ces accélérations, ne répondent plus de manière prévisible. Une autre solution prévue par la librairie est de multiplier les simulations qui utilisent le modèle pour en récupérer plus de données. Cependant, cette solution s’est aussi avérée inutilisable dans notre cas à cause de son temps de mise en place.

# Protocole de test

Lors de la planification, nous avions prévu du temps pour tester l'application. Cependant, certaines tâches ont débordé sur celui-ci, ce qui ne nous a pas laissé le temps de faire un document de test. Entre outre, nous avons tout de même testés l’application tout au long du projet et pu corriger les bugs que nous avons trouvés, par exemple, le bug de la simulation physique des collisions avec les murs commençait à avoir des interactions étranges lors de collisions à grande vitesse ou à des angles presque colinéaires. C’est pourquoi nous avons décidé de changer la logique du code et de comparer la position horizontale et verticale de la balle avec des coordonnées orthogonales plutôt que de se fier à la simulation physique d’Unity.

# 

# Conclusion

Pour conclure, ce projet a été extrêmement intéressant et nous avons eu beaucoup de plaisir à travailler dessus. Cette expérience nous a fait développer de nouvelles compétences dans le domaine des IA et des réseaux neuronaux et nous a donné l’envie d’explorer plus en détails ce secteur de l’informatique. De plus, nous sommes fiers d’avoir réussi à implémenter une l'intelligence artificielle en résolvant tous les différents problèmes que nous avons rencontrés.

## Bilans personnels

|  |  |
| --- | --- |
| YannisTravail accomplis IA : 80%  Documentation : 40%  Pong Jeux : 30%  Poster : 50% | RaphaëlTravail accomplis IA : 20%  Documentation : 60%  Pong Jeux : 70%  Poster : 50% |

# Bibliographie

1. Tutoriel ML Agents

<http://www.immersivelimit.com/tutorials/machine-learning-pig-agents-unity>

1. Intelligence Artificielle [↑](#footnote-ref-1)
2. Étapes d’observation, d’action, ou de récompense du réseau neuronal [↑](#footnote-ref-2)