Documentation du Projet

# Introduction

Dans le cadre du module 122 il nous est demandé de développer un script PowerShell qui effectue une tâche de notre choix. Je suis partie sur un Morpion, alors que d’autres ont décidé de faire un Snake en intégrant une base de données SQL, ou alors un script de sauvegarde de dossiers.

Mon choix s’est porté sur le Morpion car il permet de bien s’approprier les bases d’un langage de programmation. Les règles étant simples cela permet aussi de bien se concentrer sur le code et son fonctionnement plutôt que se casser la tête sur la compréhension du jeu.

# Cahier des Charges Initial

- Description détaillée du projet

Le projet aura pour but de créer un Morpion jouable sur un (si j’ai le temps et la motivation 2) ordinateur. Les joueurs pourront, à la manière des échecs, choisir une case dans laquelle ils veulent placer (A1, A2, A3, B1, B2, B3). La grille s’affichera en forme d’art ASCII qui sera enregistré dans un fichier au préalable, afin d’éviter de « hard coder » des données. En paramètres il sera possible de donner les noms des joueurs et un fichier de log pourra garder une liste des joueurs ainsi que le nombre de parties gagnées. Si aucun nom n’est donné, ça sera joué par Anonyme1 et Anonyme2. Le programme aura aussi pour travail de garder un fichier de log pour permettre de suivre s’il y a une erreur ou bug dans le code. Ces logs auront une forme de fichier JSON pour permettre de les classer par lancement du script.

- Modèle de cahier des charges validé par l’enseignant



# Description Technique du Projet

- Fonctionnement global du script PowerShell

Le script se lance avec deux paramètres obligatoires : Les pseudonymes choisis par les joueurs. Il va ensuite demander ce qui est voulu de la part des joueurs (jouer/afficher le scoreboard/quitter).

SI jouer : Une boucle while(!termine) va se lancer jusqu’à ce qu’il y ait un gagnant annoncé

SI scoreboard : Affichage du scoreboard ordré par le score le plus haut

SI quitter : Le script se termine à l’aide d’un « exit »

- Utilisation éventuelle de REST API

Bien que l’envie soit présente, le temps va me manquer pour implémenter une API de manière complète et fonctionnelle. Il n’y en a donc pas pour ce projet. Je reste cependant intéressée par ce fonctionnement et vais sûrement finir par en implémenter dans mes projets personnels.

# Détails du Code

Les erreurs sont contrôlées de deux manières différentes. Soit c’est via le « default » d’un switch, soit via des try-catch. A chaque étape un log [ERROR] est ajouté au fichier si une erreur apparaît.

- Portabilité du code (variables d’environnement)

Le code n’utilise que des chemins relatifs et aucune variables externes à lui-même, il devrait donc être portable sur tout système permettant l’exécution d’un code PowerShell.

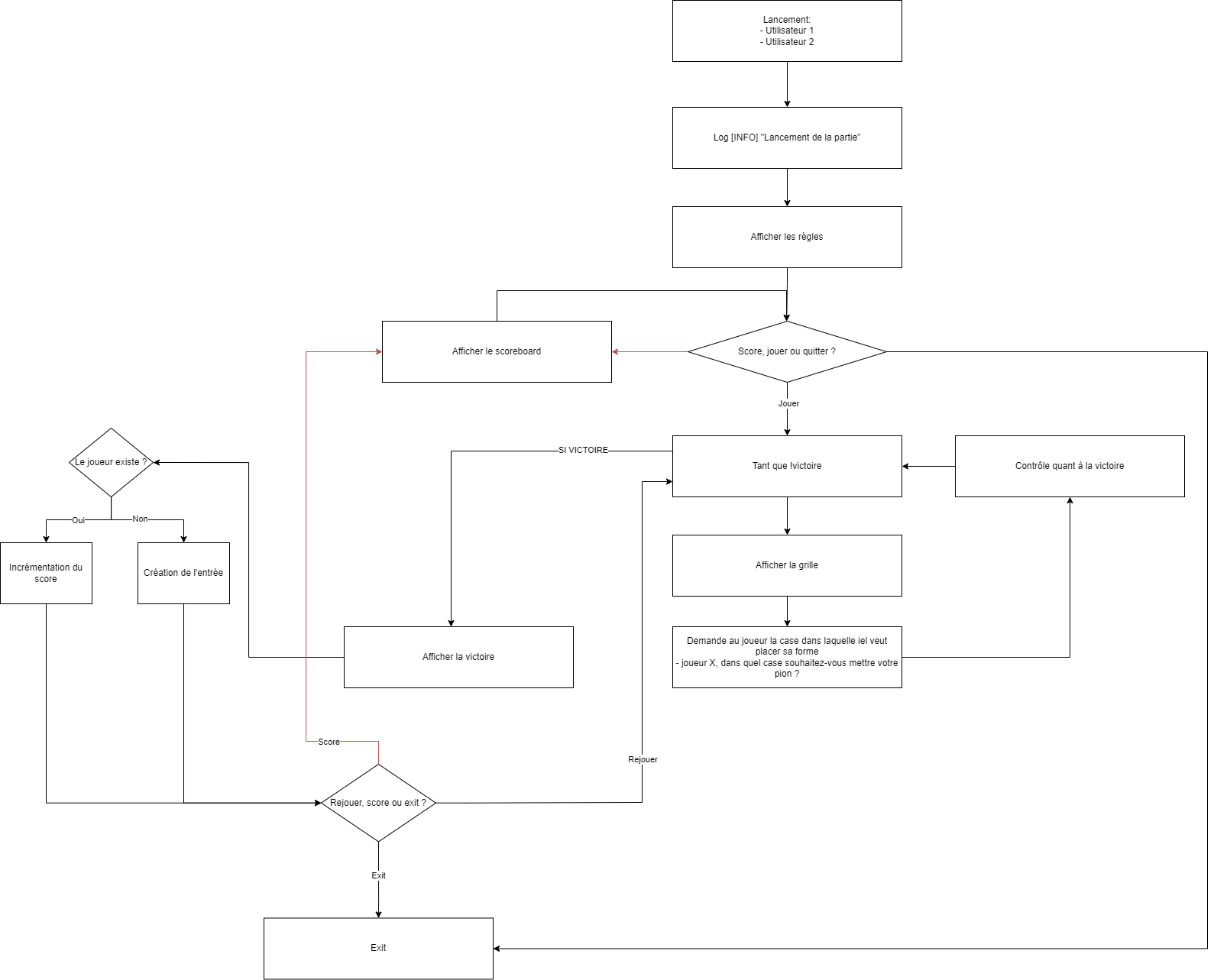
# Instructions pour l’Exécution du Script

Pour lancer le script il suffit de le lancer en double cliquant dessus avec la souris. Cela va vous demander les pseudonymes des joueurs, puis lancer le script.

Pour le lancer depuis le terminal, voici la commande à entrer depuis le dossier dans lequel se trouve .\morpion\_lucielle.ps1 (Modifier ce qui est entre parenthèses):

.\morpion\_lucielle.ps1 -joueur1 (votre pseudo) -joueur2 (votre pseudo)

# Diagramme de Flux





# Protocole de Test

Voici les documents concernant les différents tests effectués (Les deux fichiers sont les mêmes, juste sous format différent)



# Différences par Rapport au Cahier des Charges Initial

La seule différence notable entre le cahier des charges initial et le résultat est le passage d’un fichier de log JSON vers un fichier de log sous forme de texte. C’est une méthode plus simple et sûre afin de s’assurer que tout fonctionne correctement et demande moins de temps d’implémentation. C’est aussi pour suivre un standard que cette décision a été prise

# Conclusion

Ce projet a démarré très vite. J’ai pu avancer rapidement et sans grosses embuches. Cependant, cela n’a pas duré, j’ai vite rencontré, avec frustration, des fonctionnalités de powershell qui me semblent très illogique. Heureusement que monsieur Badoud a su me sortir de ces embêtements. Durant ce module j’ai compris pourquoi mes connaissances et amies codeuses répètent sans cesse la phrase suivante :

« Une méthode effectuent UNE fonctionnalité, rien de plus. »

Ce projet m’a permis de réaliser que ce n’est pas extrêmement difficile de se lancer dans un projet et l’effectuer. Maintenant que ceci est acquis, il me faut des projets, ce qui est rapidement plus compliqué.

J’ai beaucoup aimé ce projet et ai hâte d’en avoir plus à effectuer.

# Annexes

<https://learn.microsoft.com/en-us/powershell> A été un sauveur pour la plus grande partie des questions et incompréhensions que j’ai eues.