Palantir Project

Guide du développeur

Donato GENTILE
Version 1.0.0

Table de contenu

1 Qu'est-ce que Palantir ?

Palantir est le companion de jeu pour les débutants de Age Of Empire II: Definitive Edition.

Avec Palantir, vous passerez plus de temps à découvrir les mécaniques avancées du jeu qu'à vous attarder sur les détails de débutants.

1.1 Comment cela fonctionne ?

Palantir combine de l'intelligence artificielle et une base de stratégies provenant de guides et forums de Age Of Empire II: Definitive Edition. Il met en relation toutes ces informations stockées pour pouvoir vous afficher, en temp réel et en superposition au jeu, non seulement le meilleur contre pour la sélection de team actuelle, mais aussi la stratégie de base à suivre pour ne pas être en retard durant la partie.

1.2 En quoi cet overlay me rend un meilleur joueur ?

Palantir va vous alléger la lourde tâche d'adaptation au jeu et à ses stratégies de bases ainsi qu'à ses timings serrés que les débutants n'acquièrent que plus tard dans le jeu. Il va donc vous permettre de vous attarder aux spécialités des civilisations que vous jouez, mais aussi vous donner plus de temps pour vous familiariser avec l'interface du jeu sans pour autant être totalement perdu par ce qui se passe à l'écran. En somme, vous prenez du plaisir à jouer dés le début, sans devoir attendre d'en devenir un expert.

1.3 Quel est l'intérêt de ce guide du développeur ?

Ce guide va vous permettre de comprendre comment fonctionne chaque partie du code de l'API de l'application. Notez bien que cette documentation n'est qu'une petite extension à la documentation déjà présente sur le code. Dans cette documentation, vous y trouverez quelques exemple d'utilisation pour chacune des fonctions de l'API. L'API de l'interface graphique (wxPython) n'est pas expliquée ici. Pour cela, referez-vous directement à la documentation de ce module python. De ce fait, énormément de choses (notamment sur les structures des fichiers) sont déjà décrites dans le guide de l'utilisateur.

2 Installation

Palantir vous propose deux méthode d'installation. Une installation plus "barebone" où vous pourrez lancer le logiciel directement avec le fichier python principal, et une installation où vous aurez directement un fichier .exe comme tout autre application pour pouvoir lancer le programme.

2.1 Pré-requis

- Python 3.6 ou plus (Langage de programmation)
- OpenCV 4.7.0.72 ou plus (Reconnaissance d'image)
- PyAutoGUI 0.9.53 ou plus (Capture d'écran)
- wxPython 4.2.0 ou plus (Interface utilisateur)
- Tesseract 5.3.1 ou plus (Reconnaissance de caractères)
- Age Of Empire 2: Definitive Edition (Jeu)

2.2 Installation de Tesseract

1. Téléchargez Tesseract-OCR.
2. Exécuter l'installer et complétez l'installation.
3. Ajouter le dossier racine d'installation à vos variables d'environnement.

2.2.1 Comment ajouter une variable d'environnement ?

1. Dans le champ de recherche du menu démarrer, saisissez "Modifier les variables d'environnement système".
2. Dans la fenêtre des propriétés système que vous venez d'ouvrir, cliquez sur "Variables d'environnement".
3. Dans la zone Variable utilisateur pour ..., cliquez sur Path puis sur Modifier.
4. Dans la fenêtre Modifier la variable d'environnement, cliquez sur Nouveau.
5. Une nouvelle ligne est ajoutée et sélectionnée. Cliquez sur Parcourir.
6. Sélectionnez le dossier racine d'installation de Tesseract-OCR.
7. Cliquez sur OK 3 fois.

2.3 Installation "barebone"

1. Clonez le repository GitHub avec la commande suivante:

```
git clone
↪ https://github.com/UNamurCSFaculty/2223_INFOB318_Palantir.git
```

2. Lancez une console dans le dossier racine du projet cloné.
3. Installez les packages requis à python pour lancer le projet avec la commade suivante:

```
pip install -r ./resources/requirements.txt
```

4. Lancez le projet en exécutant la commande `python3 palantir_main.py`.

2.4 Installation avec exécutable

1. Téléchargez la dernière release de Palantir depuis le repository GitHub.
2. Extrayez le contenu du dossier compressé.
2. Exécutez le fichier "Palantir.exe" pour lancer le programme.

2.5 Configuration du jeu

Pour que Palantir fonctionne correctement, il va falloir que vous changiez quelques paramètres en jeu. Dans la Figure 1 et 2, vous pourrez voir ce à quoi doivent ressembler vos paramètre pour utiliser Palantir.

Dans cette Figure 1, 5 paramètres sont primordiaux pour que Palantir fonctionne. La langue du jeu doit être en Anglais, la font en "Smooth Serif Font", la taille de l'HUD à 125%, le jeu en plein écran et la résolution en "1920 x 1080". Nous vous conseillons aussi de mettre l'option "Tooltip Position" en "Fixed" pour que ceux-ci ne viennent pas se mettre devant du texte que Palantir pourrait vouloir lire.

Figure 1: Configuration du jeu requise

3 Parseur de données JSON

3.1 Fonction "parse_strategy_from_json_file"

Cette fonction prenant en argument le chemin d'accès de votre fichier "counters.json" sert à parser le dictionnaire de contre sous format "JSON". Grâce à cela, il sera plus aisé de manipuler la donnée plus tard dans d'autres fonctions de parsing. Par défaut, le chemin d'accès se trouve à la racine du dossier "resources" projet.
Exemple d'utilisation:

```
my_counter_dictionary =
 ↪  parse_strategy_from_json_file("path/to/my/counters.json")
```

3.2 Fonction "get_civilization_counters"

Cette fonction prenant en paramètre le nom de la civilisation, permet de récupérer la liste des civilisations qui contre la civilisation donnée.
Exemple d'utilisation:

```
civ_counters = get_civilization_counters("Aztecs")
for counter in civ_counters:
    print(f"{counter} counters the Aztecs civilization! You should
     ↪  use it to win against them!")
```

3.3 Fonction "get_civilization_strategy"

Cette fonction prenant en paramètre le nom de la civilisation, permet de récupérer sous forme de dictionnaire la stratégie de la civilisation entrée. Le chemin d'accès par défaut pour un fichier de stratégie se trouvant dans le dossier "strategies" à la racine du dossier "resources" du projet. Exemple d'utilisation:

```
aztecs_strategy = get_civilization_strategy("Aztecs")
aztecs_strategy_dark_age = aztecs_strategy["dark_age"]
```

4 Parseur de stats

4.1 Fonction "get_stats"

Cette fonction, vous permet de récupérer depuis une capture d'écran du jeu en 1920*1080 et le HUD en 125% de récupérer les informations relatives aux statistiques du joueur. Dans ces statistiques nous avons, le bois, la nourriture, l'or, la pierre et la population. Ces données vous serons renvoyée sous forme d'une liste avec ce format: [Bois, Nourriture, Or, Pierre, Population Actuelle/Population Max]. Exemple d'utilisation:

```
stats = get_stats()
print(f"Wood: {stats[0]}")
print(f"Food: {stats[1]}")
print(f"Gold: {stats[2]}")
print(f"Stone: {stats[3]}")
print(f"Population: {stats[4]}")
```

4.2 Fonction "get_villagers_count"

Cette fonction vous renvoie sous format de chaîne de caractères, le nombre de villageois que vous avez en ce moment d'après la capture d'écran en 1920*1080 et le HUD en 125%.

```
villagers = get_villagers_count()
print(f"Villagers: {villagers}")
```

4.3 Fonction "get_actual_age"

Cette fonction vous renvoie sous format de chaîne de caractères, l'âge de votre civilisation que vous avez en ce moment d'après la capture d'écran en 1920*1080 et le HUD en 125%. Exemple d'utilisation:

```
civ_age = get_actual_age()
if civ_age == "Dark Age":
    print("You are in the Dark Age !")
```

5 Parseur d'écrans de jeu

5.1 Fonction "is_on_team_selection_screen"

Cette fonction, vous permet de savoir si vous êtes actuellement sur l'écran de sélection des équipes sous forme d'un booléen. Exemple d'utilisation:

```
is_on_team_selection_screen = is_on_team_selection_screen()
if is_on_team_selection_screen:
    print("You're on the team selection screen!")
```

5.2 Fonction "is_on_civilization_selection_screen"

Cette fonction, vous permet de savoir si vous êtes actuellement sur l'écran de sélection des civilisation (sous menu du sélecteur d'équipe) sous forme d'un booléen. Exemple d'utilisation:

```
is_on_civilization_selection_screen =
↪  is_on_civilization_selection_screen()
if is_on_civilization_selection_screen:
    print("You're on the civlisation selection screen!")
```

5.3 Fonction "is_on_game_start_screen"

Cette fonction, vous permet de savoir si vous êtes actuellement sur l'écran de chargement de la partie sous forme d'un booléen.

/!\Cette fonction utilise de la reconnaissance d'image pour fonctionner. L'image utilisée est la suivante: "./resources/images/game_start_splash.png" /!\
Exemple d'utilisation:

```
is_on_game_start_screen = is_on_game_start_screen()
if is_on_game_start_screen:
    print("You're on the game start screen!")
```

5.4 Fonction "is_on_in_game_screen"

Cette fonction, vous permet de savoir si vous êtes actuellement sur l'écran de chargement de la partie sous forme d'un booléen.
/!\Cette fonction utilise de la reconnaissance d'image pour fonctionner. Les images utilisées sont les suivants: "./resources/images/in_game_template_[0-1].png" /!\

Exemple d'utilisation:

```
is_on_in_game_screen = is_on_in_game_screen()
if is_on_in_game_screen:
    print("You're in game!")
```

6 Parseur de choix de civilisation

6.1 Fonction "get_menu_teams"

Cette fonction, vous permet de récupérer durant la sélection des équipes, les équipes adverses. Une équipe en Random, sera bien notée Random, la liste peut donc contenir des doublons.

```
get_menu_teams = get_menu_teams()
for (index, team) in get_menu_teams:
    print(f"Player {(index + 1)} selected the {team}
    ↪  civilisation")
```

6.2 Fonction "get_player_team_in_game"

Cette fonction, vous permet de récupérer la civilisation que vous jouez en pleine partie grâce à son emblème (par exemple si l'overlay est lancé en pleine partie et non à partir du sélecteur d'équipes).
/!\Cette fonction utilise de la reconnaissance d'image pour fonctionner. Les images utilisées sont dans le dossier suivant: "./resources/images/emblems" /!\
Exemple d'utilisation:

```
get_player_team_in_game = get_player_team_in_game()
if get_player_team_in_game == "Azetcs":
    print("You're playing Aztecs civilization!")
```

6.3 Fonction "get_players_amount"

Cette fonction, vous permet de récupérer le nombre de joueur qu'il y a dans la partie. La détection se fait à partir de l'écran de choix des équipes. Ce chiffre est notamment utile pour calculer l'offset de cropping pour la détection des équipes dans l'écran de chargement. Exemple d'utilisation:

```
get_players_amount = get_players_amount()
if get_players_amount > 4:
    print("You should play on a medium sized map!")
```

6.4 Fonction "get_counters"

Cette fonction, vous permet de récupérer dans un dictionnaire tout les contres de chaque civilisation. Le format du dictionnaire est le suivant: civilisation:[contres].
Exemple d'utilisation:

```
get_counters = get_counters()
print(f"The Aztecs have {len(get_counters["Aztecs"])} counters
 ↪  which are:")
for (index, aztecs_counter) in get_counters["Aztecs"]:
    print(f"{index}. {aztecs_counter}")
```

6.5 Fonction "get_best_pick"

Cette fonction, vous permet de récupérer la civilisation qui représente le meilleur contre pour les civilisations ennemies lues dans l'écran de sélection des équipes. Cela permet au joueur de pouvoir choisir cette civilisation pour obtenir un avantage de choix d'équipe dés le début de la partie. Exemple d'utilisation:

```
get_best_pick = get_best_pick()
print(f"Le meilleur contre que vous pourriez jouer pour la
 ↪  sélection ennemie actuelle est {get_best_pick}")
```

7 Système de frames

7.1 Les frames et leurs sous-frames

wxPython nous laisse créer des "Frames" qui sont rien d'autre que des fenêtre auquels nous pouvont leur donner des attributs et des formes différentes pour qu'elles s'affichent sur l'écran. Dans notre cas, nous utilisons le système de frames et de sous frames pour contenir chaque frame derrière celle principale. Cette approche a le mérite de faire en sorte que lorsque la main frame est fermée, toutes les sous frames se ferment automatimquement ainsi que les threads allant avec.

7.2 Mises à jour des frames

Comme tout UI, les frames doivent se mettre à jour lorsque les informations à afficher changent. La main frame a un timer qu'on utilisera comme "Master Timer" qui va définir l'horloge des autres frames. Ce timer s'éxécutera toutes les 20 millisecondes et est injecté directement dans l'update des sous frames, un peu comme dans un observer pattern.

Dans la fonction constructrice de ces frames, vous y retrouverez tout le code relatif à wxPython pour fixer la mise en page et les attributs de nos frames. Celle-ci sont ici indiquées comme transparente (changement de l'opacité pour voir au travers), ainsi que transparente aux clics (possibilité de cliquer au travers de l'overlay là où il n'y a pas de boutons).
Dans la fonction "update", vous y trouverez toute la logique de mise à jour de la frame en question qui s'exécutera donc toutes les 20 millisecondes lorsque la main frame fera son tour d'horloge et qu'elle appellera cette même fonction pour chaque frame **visible**.
Pour plus d'informations sur ce que certains fonctions font, vous pouvez regarder aux chapitres précédents concernant l'API qui est utilisée pour dans les fonctions "update".

7.3 Émulateur de clavier virtuel

Palantir va devoir émuler un clavier virtuel pour enregistrer vos clics de souris sur le jeu. En effet, le jeu étant sous DirectX, les clics ne sont pas lu par l'API de windows, mais bien par l'API interne de DirectX. Grâce à la librairie "pywin32", nous pouvons faire semblant que notre programme est un HID et ainsi capturer les clics en lisant les appels à l'API de DirectX lorsque des clics sur le jeu sont réalisés.

De ce fait, dans le fichier "key_press_thread.py", vous y trouverez une implémentation de détection de clic souris dans la région d'un de nos boutons de notre overlay. Pour lancer cet émulateur, il suffit de lancer un thread **non bloquant** à la l'instantiation de la frame de l'overlay comme fait pour la frame "in_game_screen_frame". Lors de la creation de votre thread, il vous faudra renseigner la frame associées et il faudra bien enregistrer les boutons et leur fonction d'activation dans un dictionnaire nomé "buttons". Vous avez un exemple encore une fois dans "in_game_screen_frame". Cela permettra de lier un bouton avec une fonction que le thread activera si la souris se trouve dans la région du bouton.

7.4 Fermeture et désactivation automatique des frames

La main frame ayant 2 timers, l'un pour l'update et la désactivation automatique des frames (toutes les 20 millisecondes), l'autre est pour la fermeture automatique de Palantir (s'active toutes les secondes).

La désactivation automatique des frames s'opère dés lors que le jeu n'est plus en avant plan. Il permet ainsi de naviquer sur l'ordinateur sans avoir l'overlay toujours ouvert sur d'autres applications. Nous utilisons ici le module "pywin32" qui nous permet de voir quelle application est en avant plan.

La fermeture automatique des frames est vérifiée toutes les secondes par une deuxième horloge qui elle appelle un thread du fichier "auto_close_thread.py" vérifiant que le processus du jeu soit encore en cours, autrement, Palantir se ferme automatiquement.

8 Overlay de sélection d'équipe

8.1 Qu'est-ce que l'overlay de sélection d'équipe ?

Palantir va pouvoir vous aider pendant la sélection des équipes en analysant les équipes choisies par les autres joueurs en temps réel et ainsi vous donner la meilleure civilisation à jouer dans cette situation.

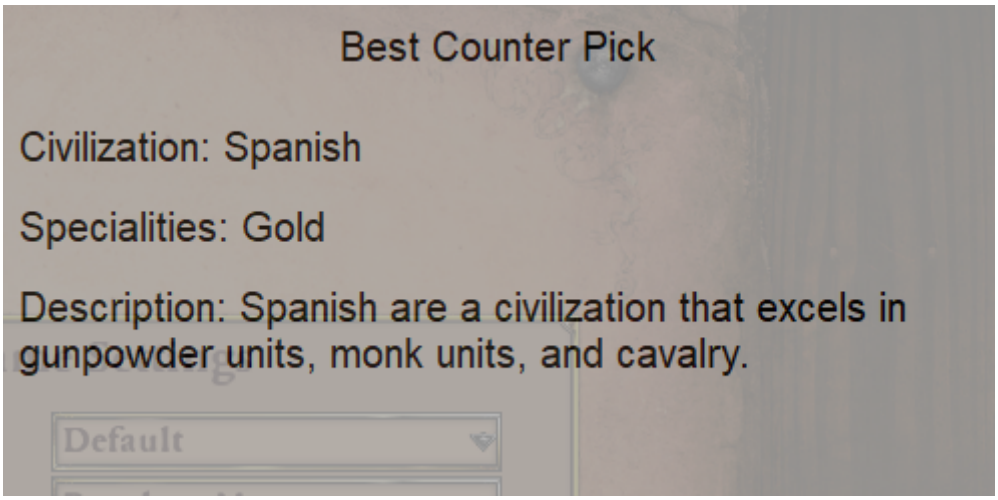

Figure 2: Overlay de sélection d'équipe

Notez bien que le calcul du meilleur contre se base sur les contres de chaque civilisation inscrite dans le fichier des contres "counters.json".

8.2 Fichier des contres "counters.json"

Dans le fichier des contres, vous aurez la possibilité d'ajouter ou de retirer un contre d'une civilisation particulière. Ceci peut être utile lorsqu'une civilisation se fait buff lors d'un patch et qu'elle n'est plus contrée par une certaine civilisation et vice-versa. La structure du fichier est simple et est la suivante:

```
{
    "NomDeCivilisation": {
        "counters": [
            "Civilisation1",
            "Civilisation2"
        ]
    }
}
```

Les 42 civilisations sont déjà pré-remplie avec des contres qui sont pertinents au moment de la publication de la dernière version de Palantir.

9 Overlays durant la partie

9.1 Qu'est-ce que sont les overlay en jeu ?

Palantir vous propose une stratégie de base qui change en fonction de la civilisation que vous aurez choisi. Celle-ci se mettra à jour en fonction de l'âge dans lequel vous êtes actuellement, vous donnant ainsi une stratégie base qui vous accompagne tout au long de votre partie pour vous aider à tenir les timings et les ressources bases pour profiter du jeu et avoir de la vraie compétition durant vos parties.

9.2 Overlay de stratégie

L'overlay de stratégie se base sur ***le fichier de configuration*** de la civilisation sélectionnée pour vous afficher chaque étape de la stratégie à l'âge dans lequel vous vous trouvez actuellement. Vous pouvez naviguer facilement dans la stratégie grâce aux boutons présents sur l'interface.

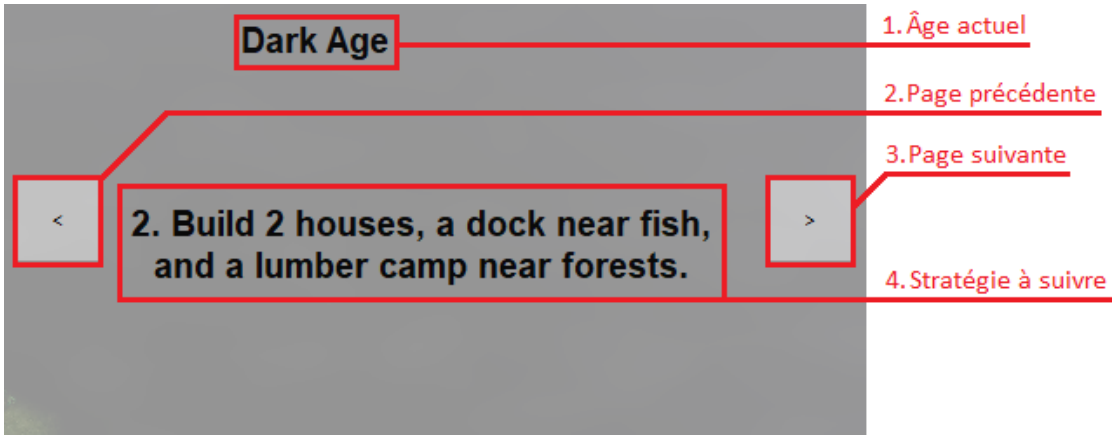

Figure 3: Overlay de stratégie de base

1. Âge actuel : Âge actuel de votre civilisation.
2. Page précédente : Bouton allant à la page précédente de la stratégie.
3. Page suivante : Bouton allant à la page suivante de la stratégie.
4. Stratégie à suive : Descriptif de l'étape de la stratégie à suivre.

9.3 Overlay par seuils

Cet overlay est aussi un des overlays qui s'afficheront durant la partie. Il ne s'affichera que lorsque certains seuils définis dans ***le fichier de configuration*** seront atteint. Par exemple, il s'affichera si vous avez atteint le nombre de ressource nécessaire pour le passage de votre civilisation à l'âge suivant comme montré dans la Figure 4.

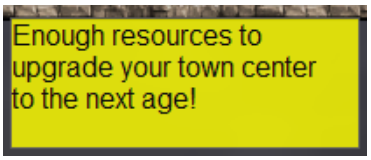

Figure 4: Overlay de seuil

10 Fichiers de configuration

Palantir offre un système de fichiers de configuration qui vous permettra de créer vos propre stratégie et de les suivre en jeu grâce à l'***overlay***.

10.1 Structure des fichiers de configuration

Chaque fichier de configuration est propre à une civilisation. À l'heure actuelle, il n'est possible d'avoir qu'un seul fichier de configuration pour chacune des civilisations. Dans de prochaines mise à jour, la possibilité de choisir un fichier parmi plusieurs pour une même civilisation sera ajoutée.
La structure du fichier est simple à comprendre et se présente de la façon suivante:

```
{
    "description": "Brève description de la civilisation avec ses
    ↪  points forts/négatifs."
    "speciality": "Spécialité d'une civilisation."
    "dark_age": [
        "Page 1 de la stratégie pour l'âge sombre.",
        "Page 2 de la stratégie pour l'âge sombre."
    ]
    "feudal_age": [...]
    "castel_age": [...]
    "imperial_age": [...]
    "thresh_holds": {
        "villagers": {
            "dark_age": Recommandé pour l'âge sombre
            "feudal_age": Recommandé pour l'âge feudal
            "castle_age": Recommandé pour l'âge des chateaux
        }
        "food": {
            "dark_age": Recommandé pour passer à l'âge feudal
            "feudal_age": Recommandé pour passer à l'âge des
            ↪  chateaux
            "castle_age": Recommandé pour passer à l'âge impérial
        }
        "wood": {...}
        "gold": {...}
        "stone": {...}
    }
}
```

Il est à noté que le nom du fichier doit être le nom **exact** (majuscule comprise) de la civilisation à lequel il appartient. Par exemple, un fichier de configuration pour la civilisation des Azetcs serait "Aztecs.json".

11 Foire Aux Questions

11.1 Pouvons-nous créer des stratégies par map ?

Malheureusement, il n'est pas encore possible de faire une stratégie en particulier pour chaque map. Cependant, c'est une fonctionnalité qui est en cours de développement.

11.2 On me demande des ressources que je ne vois pas.

Il se peut que, même après exploration de la carte avec un éclaireur, la stratégie de base ne soit pas adaptée à 100% à la carte que vous jouez. Ainsi, si il vous demande par exemple de chercher des buissons et qu'il n'y en a pas aux alentours, comprenez qu'ils vous faut récupérer de la nourriture au plus vite. Certaines map sont aussi dépourvue de points d'eau ce qui est parfois le seul point fort d'une civilisation, dans ce cas, la stratégie de base ne sera pas suffisant pour cette map. Nous ajouterons des stratégies particulières à chaque map dans de prochaines mises à jours.

11.3 Certaines stratégies ne me semblent plus d'actualité.

Le jeu est en constante évolution, il est de ce fait possible que certaines stratégies de base ne soient plus d'actualité et qu'il vaudrait mieux les mettre à jour, pour cela vous pouvez créer votre stratégie ou nous ouvrir un ticket sur le repository GitHub pour que l'on puisse la repasser en revue.

11.4 Comment puis-je soumettre ma propre stratégie ?

Si vous souhaitez mettre à jour ou soumettre une meilleure stratégie pour une civilisation, vous pouvez directement créer un P.R. (Pull Request) directement sur notre repository GitHub !

11.5 Je ne comprend pas grand chose aux guides.

Bien que ce soient des guides pour débutants, il est évident qu'il faut au moins suivre le tutoriel du jeu pour comprendre les mécaniques de base du jeu et pouvoir comprendre les informations affichées par l'overlay. Nous vous conseillons donc de faire toutes la série d'entrainement disponible dans le mode campagne.

11.6 Est-ce que je peux me faire bannir si j'utilise Palantir ?

Non. Palantir n'est pas un logiciel de triche. Nous avons tout fait pour que l'anti-cheat du jeu ne puisse pas détecté un faux positif avec Palantir. Palantir ne fournit que des informations basées sur le visuel que peut apporter le jeu. Il ne vous montre pas d'informations cachées dans la mémoire du jeu ou d'autres informations qui constituerait un avantage excessif par rapport aux autres joueurs.

11.7 J'ai trouvé un bug, comment puis-je le signaler ?

Pour nous signaler un bug, vous pouvez simplement ouvrir un ticket sur le repository GitHub avec votre version de Palantir ainsi que les étapes pour reproduire le bug, nous pourrons ensuite le corriger.

11.8 Suis-je obligé de mettre mon HUD à 125% ?

Oui. Palantir utilise un modèle de reconnaissance de texte qui nécessite une certaine grandeur du texte pour garantir une précision de détection du texte en jeu. Sans cela, les zones de détection du texte se retrouvent déplacée et la précision de détection du texte se trouve détériorée.

11.9 Puis-je mettre mon jeu en français ?

Non. Palantir n'est actuellement disponible que pour la version du jeu en Anglais. En effet, nous utilisons les noms anglais des civilisations et des âges pour la détection et l'affichage des stratégies.

11.10 Informations de contact

Vous pouvez nous contacter avec les informations suivantes:

Email: support@donatog.tech
GitHub: Palantir