Guide d'utilisation du Discord Computer Olympiad Interface

Quentin Cohen-Solal
quentin.cohen-solal@dauphine.psl.eu

16 octobre 2025

Ce document explique comment utiliser les deux fonctionnalités principales du Discord Computer Olympiad Interface (DCOI), à savoir :

- effectuer un match en mode manuel : un humain entre textuellement les actions de son programme dans la conversation discord (Section 1).
- effectuer un match en mode automatique : utiliser un bot Discord joueur qui écrit à votre place les actions de jeu indiqué par votre programme dans la conversation discord (Section 2).

Si l'on souhaite utiliser uniquement le mode automatique, il est toutefois nécessaire de lire la section suivante, qui en plus de décrire succinctement comment jouer en mode manuel, explique le DCOI afin de faciliter la compréhension des sections suivantes, ce qui inclut comment choisir un jeu et lancer une partie. Cette première section traite également du jeu Free_game qui permet de jouer (en mode manuel ou automatique) à n'importe quelle jeu à information parfaite, en déléguant aux joueurs la détermination et la vérification des actions valides ainsi que de la fin de la partie.

Remarque 1. si vous rencontrez des frustrations ou des bugs à la lecture de ce document ou à l'utilisation du DCOI, n'hésitez pas à me les signaler.

Table des matières

1 Description du DCOI et utilisation en mode manuel

Pour utiliser le DCOI, vous devez pouvoir utiliser Discord. Vous devez donc installer l'application ou utiliser la version web de Discord et créer un compte, si ce n'est pas déjà fait (voir `https://discord.com`). Cette section commence par présenter le fonctionnement global (Section 1.1), puis présente l'accès aux salons des matchs (Section 1.2), suivi du mode d'emploi pour réaliser une partie (Section 1.3), en expliquant notamment la syntaxe des actions (Section 1.3.5). Quelques mises en garde sont également décrites dans les Sections 1.2.2 , 1.2.3 et 1.5. Enfin, nous parlons brièvement du système de logging des matchs (Section 1.4).

1.1 Fonctionnement global

1.1.1 DCOI et bot arbitre

Le DCOI est à la fois une librairie Python et un ensemble de bots Python tout prêt ou à hériter pour réaliser des bots joueurs et des bots arbitres, permettant de jouer et de gérer des matchs au sein d'une discussion Discord.

Les joueurs et opérateurs rejoignent le serveur Discord de la compétition. Lorsque les commandes de lancement de matchs sont inscrites dans la conversation d'un salon Discord, la partie commence. Celle-ci est ainsi gérée par un bot arbitre (mis en place par les organisateurs). Les joueurs et bots joueurs doivent alors inscrire dans la conversation leurs actions de jeu lorsque le bot arbitre les y invitent. Le bot arbitre vérifie la légalité des coups, gère les erreurs, chronomètre le temps des joueurs et indique une fin de partie (normal ou du à un dépassement du temps autorisé) et le vainqueur de la partie.

1.1.2 Focus sur les bots joueurs

Toute la partie concernant Discord est automatisée avec les bots Discord. Les utilisateurs n'ont qu'une chose à faire : connecter leur programme au bot Discord joueur, qui s'occupera de gérer les événements Discord et d'écrire dans la conversation par exemple. Pour faire l'interface entre les deux programmes, il faut soit hériter un bot joueur et implémenter la méthode "plays" en utilisant les méthodes de jeux du DCOI soit implémenter le protocole de communication Go Text Protocol. Dans les deux cas, il peut aussi y avoir besoin d'effectuer une conversion de la syntaxe des actions de jeu du DCOI dans la syntaxe de votre programme. Enfin, il faut créer un compte de bot et spécifier un fichier de configuration pour faire le lien entre le compte de bot et le programme de bot.

1.2 L'application Discord et les salons des matchs

1.2.1 Discord : serveurs et salons

Nous passons rapidement en revue les différents éléments de l'application Discord dont nous avons besoin pour réaliser un match.

Une capture d'écran de l'application est disponible dans la Figure 1. La première étape consiste à rejoindre le serveur de la compétition (les serveurs sur discord s'appellent des guildes). Lors du premier accès, il faut cliquer sur le "+" en bas de la liste des serveurs (voir Figure 2). Lors de la compétition ou de la beta test, cliquez sur "rejoindre un serveur" puis entrer le lien du serveur fournit par les organisateurs. Pour la beta test, il faut rejoindre le serveur `https://discord.gg/mbHBdH3K`. Une fois le serveur ajouté à votre compte, vous pouvez à tout moment y retourner en cliquant sur l'icone du serveur désormais présente dans la liste des serveurs (voir Figure 2). Une fois rejoint le serveur, il faut rejoindre le salon du match, en cliquant sur son nom. Les salons disponibles se trouvent dans la zone prévue à cet effet (voir la Figure 3). Vous pouvez aussi rejoindre le salon général / discussion pour parler avec les autres participants ou tout autre salon, notamment pour demander de l'aide. Comme dans n'importe quelle messagerie instantanée, pour écrire dans le salon choisi, il suffit d'écrire en utilisant votre clavier dans la zone prévue à cette effet (Figure 4). Une fois appuyé sur entrée, votre message est envoyé et rejoindra la zone affichant l'historique de la conversation (milieu de la fenêtre, si besoin voir la Figure 5).

1.2.2 Matchs et discussions

Notez que vous pouvez aussi parler dans les salons des matchs, à condition de ne pas écrire comme seul élément de votre phrase une action au format textuel. En effet, si le match a commencé et que vous faites parti des joueurs, votre phrase sera interprétée comme une action de jeu. Par exemple, n'écrivez jamais juste "A1" ou "A2-B3" lors d'une conversation. Mais vous pouvez écrire à la place "Pourquoi tu as joué A1 ?" ou même juste "A1." ou "A1 ?". De manière générale, évitez autant que possible de discuter dans le salon des matchs pour ne pas le polluer, au moins durant les matchs.

1.2.3 Mise en garde sur la modification / suppression des messages

Discord vous permet de modifier et supprimer vos messages. Lors d'un match que cela soit vous ou votre bot, n'utilisez sous aucun prétexte cette fonctionnalité. Il y a une chance faible mais non négligeable que cela entraine une instabilité du système empêchant de finir la partie. Si un tel événement se produit, vous serez considéré responsable et vous perdrez la partie. Si vous modifiez votre action par erreur, remodifiez votre message pour remettre l'action initiale. Il ne devrait alors y avoir aucun problème. Si vous supprimez votre action par erreur, c'est malheureusement définitif et il faut espérer que cela n'impactera pas la suite de la partie. En cas de soucis, contactez un organisateur.

1.3 Lancement d'une partie

Nous entrons maintenant dans le vif du sujet et présentons ce qu'il faut savoir et comment faire pour lancer une partie.

1.3.1 Choix du jeu et des autres paramètres

Pour choisir le jeu du match, écrivez dans la discussion :

" !set game <GAME_NAME>".

Exemple 2. " !set game Clobber".

Remarque 3. Si cela a marché, le bot arbitre écrira dans la conversation : "Game set to <GAME_NAME>".

Pour connaitre les jeux disponibles, vous pouvez écrire dans la conversation la commande suivante :

" !available_games".

Si vous avez besoin de modifier le temps total accordé à chaque joueur par partie (défaut 30 minutes par joueur par partie), vous pouvez utiliser la commande :

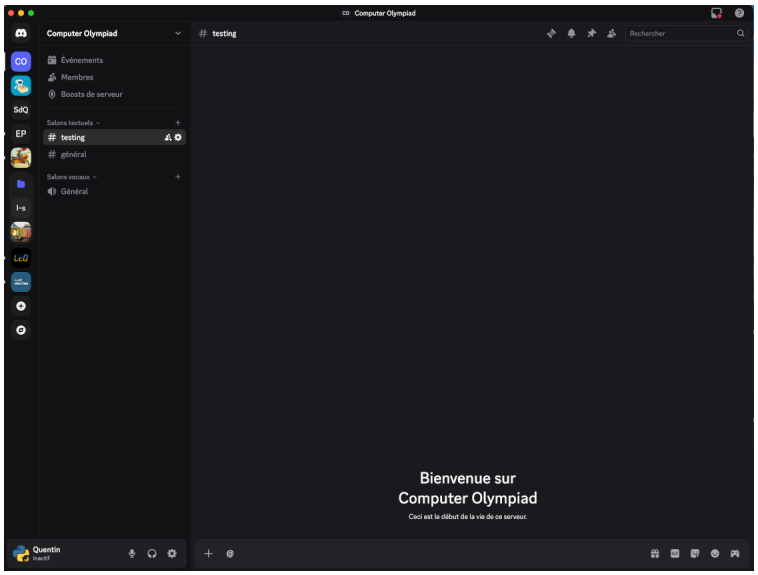

FIGURE 1 – Discord App

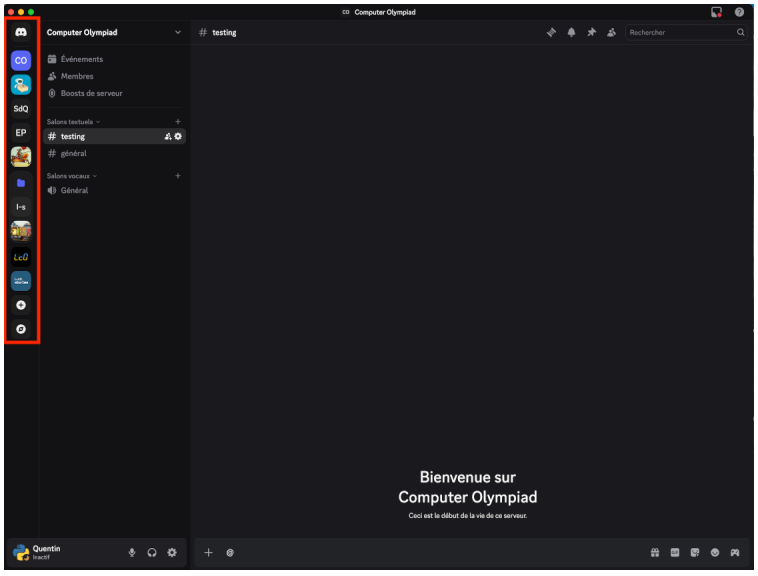

FIGURE 2 – Discord servers list (red box)

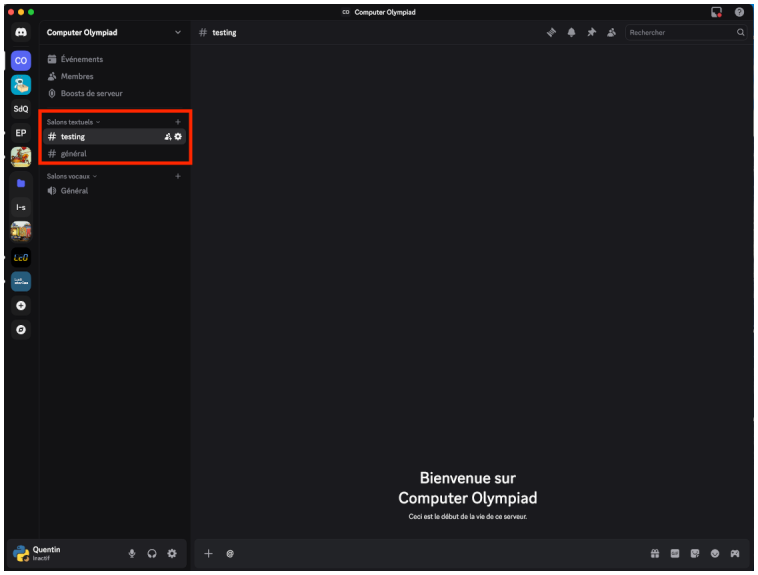

FIGURE 3 – Discord channel list (red box)

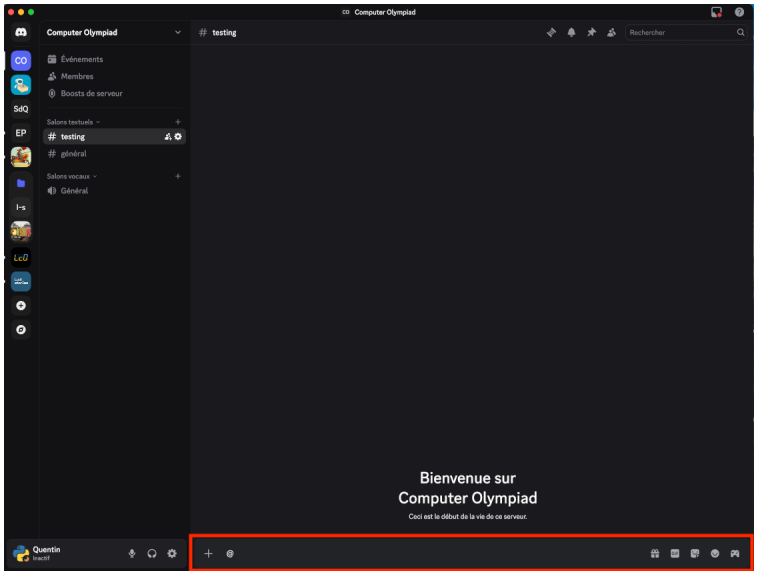

FIGURE 4 – Discord input area for conversation and match playing (red box)

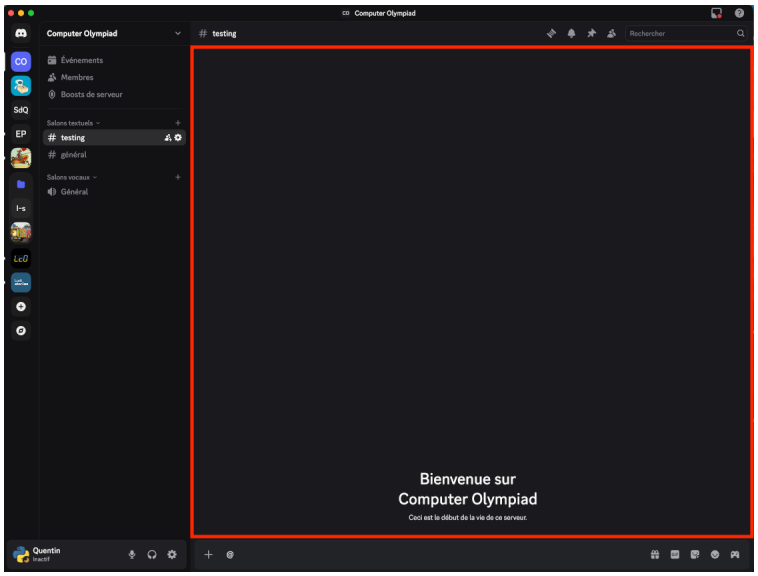

FIGURE 5 – Discord conversation / match history (red box)

" !time <T>"

avec <T> un nombre entier indiquant le temps en secondes ou un entier collé à un des mots suivants "s", "min", "h" pour spécifier l'unité (par exemple " !time 20min" ou " !time 1h").

Remarque 4. Si vous avez besoin de jouer une série de matchs sur le même jeu, il n'est pas nécessaire de fixer le jeu avant chaque lancement.

1.3.2 Cas particulier du Jeu Free_game

Le DCOI offre la possibilité de jouer à n'importe quel jeu à information parfaite sans hasard grâce au jeu nommé "Free_game". Notez toutefois que, sans que cela soit restrictif en pratique, les joueurs doivent jouer alternativement (c'est-à-dire jamais deux fois de suite). Ce dernier permet ainsi de jouer des matchs sur de tels jeux mais avec la contre-partie suivante : il n'y a pas de vérification des coups licites ni de détermination de vainqueur et les participants doivent jouer (écrire) à tour de rôle l'action "end" pour mettre fin à la partie. Cette fonctionnalité n'est donc à utiliser que si les participants sont sûr de jouer selon les mêmes règles et qu'ils disposent évidemment chacun leur propre moteur de jeu. Pour utiliser Free_game, faites :

" !set game Free_game".

Les actions autorisées avec ce jeu sont toutes les actions vérifiant la syntaxe des actions du DCOI (par exemple "A1", "B2-C3", "D4-E17-H23" ; voir la Section 1.3.5) auquel l'action "end" est rajoutée. Toutefois, vous pouvez aussi rajouter des actions spéciales au jeu Free_game avec la commande :

" !add_freegame_moves".

Vous pouvez afficher les actions spéciales additionnelles de Free_game avec la commande :

" !show_extra_freegame_moves".

Et vous pouvez retirer toutes les actions spéciales de Free_game avec la commande :

" !clear_freegame_moves".

Ces commandes sont bien évidemment à utiliser avant le début de la partie.

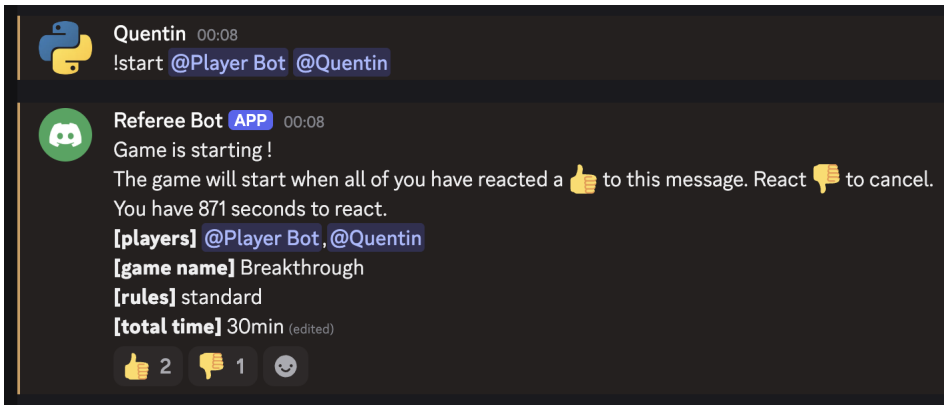

FIGURE 6 – Example of the beginning of a match (example of waiting for player confirmation to start the game)

1.3.3 Déclanchement de la partie

Nous nous intéressons maintenant sur comment réaliser en pratique le match. La première étape est de demander à l'arbitre de proposer une partie. Il faut utiliser la commande :

" !start @<Discord_Player_ID_1> @<Discord_Player_ID_2>".

Pour gagner du temps, vous pouvez utiliser l'alias " !s" à la place de " !start". Pour écrire cette commande un peu technique, commencez par écrire " !start @", Discord va alors vous afficher une liste des identifiants disponibles. Utiliser flèches du haut/bas pour choisir l'identifiant du premier joueur du match et faites "entrée". Ecrivez ensuite " @" et sélectionnez de la même façon le second joueur. Si l'un des joueurs utilise un bot, c'est l'identifiant de son bot qu'il faut utiliser et non l'identifiant de l'opérateur du bot. En outre, si l'un des joueurs utilise un bot, ce dernier doit être lancé avant que la commande !start ne soit écrite.

Remarque 5. Notez qu'il y a un bug rare de discord où une partie des joueurs / bots disponibles disparaissent du selecteur @. Ce bug se règle tout seul mais cela peut prendre du temps (heures voir jours). Cependant, il est possible de contourner simplement le bug de deux façons. On peut tout simplement utiliser un autre salon. Mais on peut aussi écrire la commande dans un autre salon et la copier coller dans le salon qui nous intéresse. Garder toujours un salon vide de toute écriture, car c'est apparemment l'envoi de messages qui provoque cet étrange bug.

Une fois la commande exécutée, l'arbitre affiche le détail de la partie proposée (voir la Figure 6). Les participants doivent alors cliquer sur le "pouce vers le haut" en bas du message pour valider la partie. Si vous utilisez un bot, cela est prévu par la librairie, vous n'avez rien à faire. Dans la configuration actuelle, les joueurs ont alors 15 minutes pour accepter la dite partie. Lorsque tous les joueurs ont cliqué sur le pouce, la partie débute. Si vous souhaitez décliner la proposition de partie : cliquez sur le "pouce vers le bas".

Remarque 6. Il est normal d'avoir le chiffre 1 pour les pouces haut et bas. C'est l'arbitre qui a ajouté ces réactions pour les rendre visible et faciliter leurs ajouts.

1.3.4 Déroulement et fin de la partie

Lors d'une partie, l'arbitre indique lorsqu'un joueur doit jouer. Il écrit dans la conversation une phrase du type "@<PLAYER> must play (he has <TIME> left)". A ce moment là, le joueur concerné peut écrire dans la conversation l'action choisie par son programme. Un joueur ne doit pas écrire avant qu'on lui a dit explicitement de jouer. La Figure 7 représente un début de partie, correspondant à la

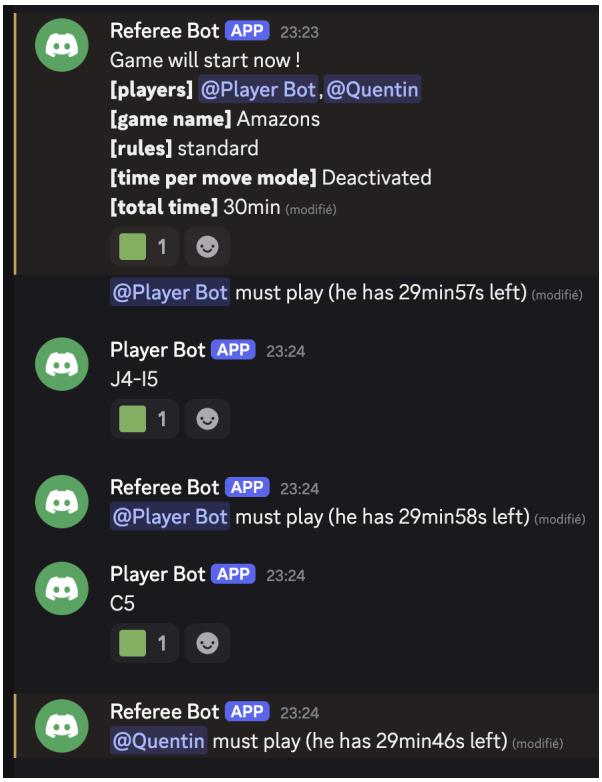

FIGURE 7 – Example of the beginning of a match

section d'action suivante : "J4-I5" puis "C5" (le premier joueur joue deux fois d'affilé). Remarquez que l'arbitre valide les actions en les annotant par un carré vert.

Si l'action écrite est incorrecte, l'action est annotée par un carré rouge et l'arbitre signale l'erreur et fourni la liste des actions licites (voir la Figure 8).

Lorsque la partie est finie, l'arbitre le signale et indique le vainqueur (voir la Figure 9).

Si un joueur dépasse son temps de réflexion, la partie se termine également et ce joueur perd la partie (voir la Figure 10).

Une fois la partie finie, vous pouvez lancer un nouveau match avec la commande !start.

Si vous avez besoin d'arrêter une partie en cours. Vous pouvez utiliser la commande " !stop". Cela va déclencher un vote. Cette commande ne fonctionne pas pour le moment si vous utilisez un bot Discord.

1.3.5 Syntaxe des actions (format textuel)

Avec le DCOI, il faut bien distinguer deux choses, la syntaxe des actions et les actions licites. Pour que le DCOI puisse différencier une conversation normal d'une action de jeu, le DCOI a besoin d'une syntaxe. Cette syntaxe inclut les actions du type "A1", "B2-C12", mais aussi "D5-E3-G15" combinés à des mots clefs spécifiques, par exemple "queen-A8". Pour connaitre la liste des mots clefs du jeu courant, il faut utiliser la commande :

!show_move_keywords.

Notez que la syntaxe générale des actions de jeu est définie par l'expression régulière suivante :

([a-zA-Z][0-9]{1,2} | keyword1 | ... | keyword_n)(-([a-zA-Z][0-9]{1,2} | mot-clé-1 | ... | mot-clé-n))*.

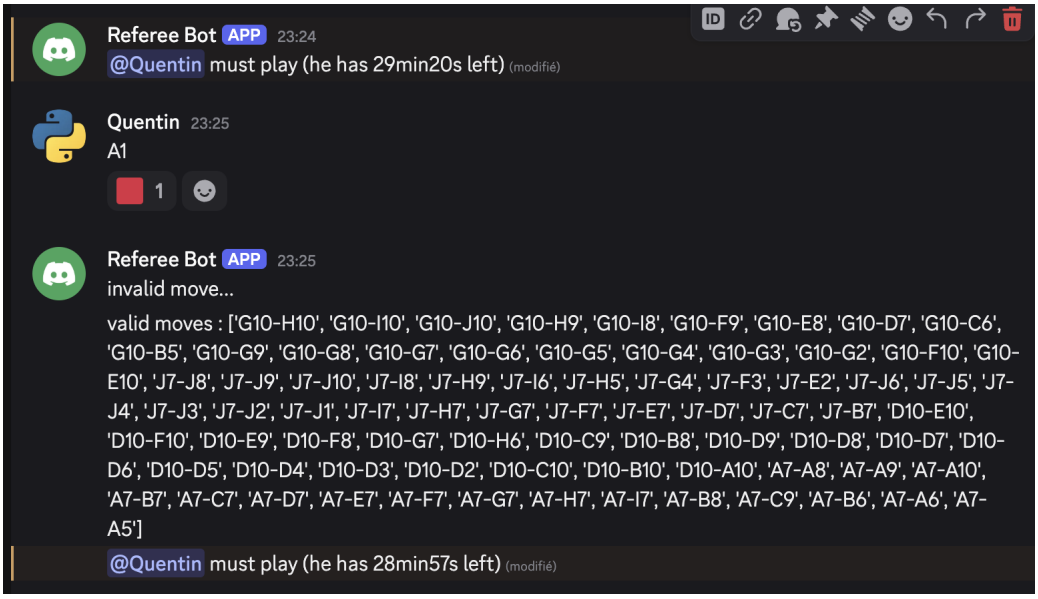

FIGURE 8 – Example of invalid move

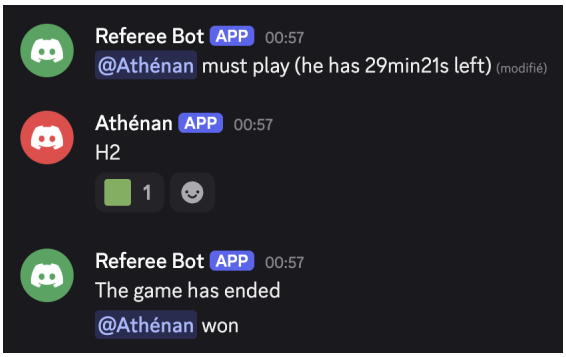

FIGURE 9 – Example of endgame

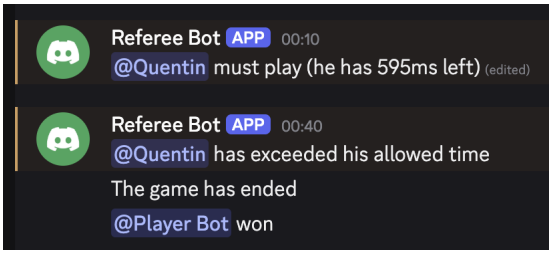

FIGURE 10 – Example of timeout

Autrement dit, une action au format textuel est une séquence de blocs de longueur supérieure ou égale à 1, séparés par des tirets qui sont soit une coordonnée (e.g. A12, B5, C18) soit un mot-clé.

Lorsqu'un texte est écrit dans la conversation, le bot arbitre vérifie si le texte respecte la syntaxe attendue. Si c'est le cas et si c'est à ce joueur de jouer, il va tenté de jouer cette action. Si cette action est valide, il la joue, sinon il signale que l'action n'est pas licite, comme décrit précédemment. Si la syntaxe n'est pas vérifiée, l'arbitre ignore tout simplement le texte.

1.4 Logging des matchs

Les parties sont sauvegardées dans des fichiers .json (dans le dossier discord_interface/log/bot_play_log/), au format

"play_<PLAYER_ID>_Computer-
Olympiad_<CHANNEL_NAME>_<DATE>_<TIME>.json".

Ce fichier contient un dictionnaire Python lisible par un humain possédant de nombreux attributs, et en particuliers les suivants : game_name, moves (history of moves of the match), players (list of PLAYER_ID), winner (True if you or your bot has won).

1.5 Avertissement sur les parties trop courtes

Notez qu'il y a une perte de temps lors de la communication des actions dû à la latence internet (qui dépend de la qualité de votre connexion) mais également de la latence de Discord et aussi dans une moindre mesure de celle du DCOI. Cela se compte en secondes, en moyenne autour d'une seconde. DCOI n'est donc pas adapté pour jouer des matchs avec un temps moyen de 1 seconde par action. En outre, et de toute façon, l'envoi de plus d'un message par seconde par un bot ou un utilisateur enfreint le contrat d'utilisation de Discord qui intreprète un tel comportement comme du flood. Ce comportement peut mener à diverses punitions (bans, shadows bans, ralentissement des fonctions discord, ...). Avec l'envoi d'actions de l'ordre de la seconde, nous n'avons cependant jamais rencontré de problèmes en pratique lors de nos nombreux tests.

2 Jeu automatisé par bot Discord

Dans cette section, nous allons expliquer les différentes façons de réaliser un match avec le DCOI en mode automatique.

Dans tous les cas, le jeu automatisé s'effectue grâce à un bot Discord joueur. Cela requiert de créer un "compte" bot Discord sur le site développeur de Discord (voir la section 2.2). Ensuite, la façon native de faire des matchs en mode automatique est d'étendre une des classes Player Python du DCOI (Section 2.3). L'autre façon, décrite dans la Section 2.4, est d'utiliser un bot déjà intégralement implémenté qui communique avec votre programme grâce aux entrées-sorties (Inter-process communication) en utilisant le Go Text Protocol (GTP). Cette deuxième méthode offre donc l'avantage de ne pas requérir de programmation utilisant la librairie du DCOI et donc de connaitre le langage Python.

Enfin, dans la Section 2.5, nous présentons des fonctionnalités avancées liées aux bots.

Nous commençons par détailler l'installation des packages Python requis par le DCOI.

2.1 Installation des packages requis du DCOI

Pour effectuer l'installation des packages du DCOI, vous avez deux méthodes :

Méthode rapide : Exécutez dans le terminal :
— "cd discord_interface"
— "chmod +x install.sh"

— “bash install.sh”

Remarque 7. Si vous déplacez le fichier discord_interface, vous devrez relancer “bash install.sh”.

Méthode pas à pas Le DCOI est basé sur la librairie Python officielle de Discord. Pour pouvoir utiliser la librairie du DCOI, vous avez donc besoin de l’installer. Pour ce faire, exécutez dans votre terminal : “python3 -m pip install -U discord.py”. Vous avez aussi besoin de la librairie numpy pour faire tourner les jeux : “python3 -m pip install -U numpy”. Si vous souhaitez utiliser GTP, vous avez besoin en plus de la librairie “pexpect” pour permettre les communications inter-processus, exécutez : “pip install pexpect”.

Remarque 8. Si vous utilisez GTP, si besoin, il est possible de se passer de la dépendance à numpy : me contacter pour que je modifie le code en conséquence.

Enfin, vous devez modifier le python path pour ajouter le dossier parent de “discord_interface” :

```
UPPER_DIR_DCOI='<path_to_discord_interface>'
echo 'export PYTHONPATH=$PYTHONPATH:'$UPPER_DIR_DCOI >> ~/.bashrc
source ~/.bashrc
```

Remarque 9. Si vous déplacez le fichier discord_interface, vous devrez modifier le PYTHON PATH avec le nouveau chemin.

2.2 Création du compte de votre bot Discord Joueur

Nous nous intéressons maintenant à la création du compte de votre bot, qui permet au programme du bot de se connecter à l’application Discord. Plus précisément, cette série d’étapes permet de récupérer diverses informations. Certaines de ces informations doivent être saisies dans le fichier de configuration “parameters.conf” à la racine du dossier discord_interface. La création du compte permet également de récupérer un lien à transmettre à l’organisateur pour qu’il ajoute votre bot sur le serveur Discord de la compétition.

2.2.1 Créer le compte du bot et inviter le bot dans le serveur

1. Allez sur le site web `https://discord.com/developers/applications`. Il faut vous connecter avec votre compte discord. Vous devez ensuite cliquer sur « New Application » en haut à droite de l’écran (si besoin, voir la Figure 11) et donner un nom au bot. Vous arrivez alors sur la page du bot.
2. Il faut ensuite activer l’option "Message Content Intent" dans l’onglet Bot (si besoin, voir la Figure 12).
3. Puis, il faut récupérer le TOKEN du bot, c’est en quelque sorte son mot de passe (il doit donc être gardé secret). Notez le quelque part, nous en aurons besoin pour spécifier le fichier de configuration dans la section suivante.
   (a) Allez sur l’onglet Bot (toujours sur la page du bot). Dans la Section TOKEN, cliquez sur le bouton “Reset Token” (si besoin, voir la Figure 13).
4. Allez ensuite sur l’onglet OAuth2, section “OAuth2 URL Generator”, sous-section “SCOPES”. Cochez l’option “bot”.
5. Puis dans la sous-section suivante “bot permissions”, cochez les options suivantes : "View Channels", "Send Messages", "Read Message History", "Add Reactions".
6. Pour terminer, allez en bas de la page, dans la zone “Generated URL” et copier l’url. Transmettez cet url à l’organisateur pour qu’il ajoute votre bot sur le serveur.

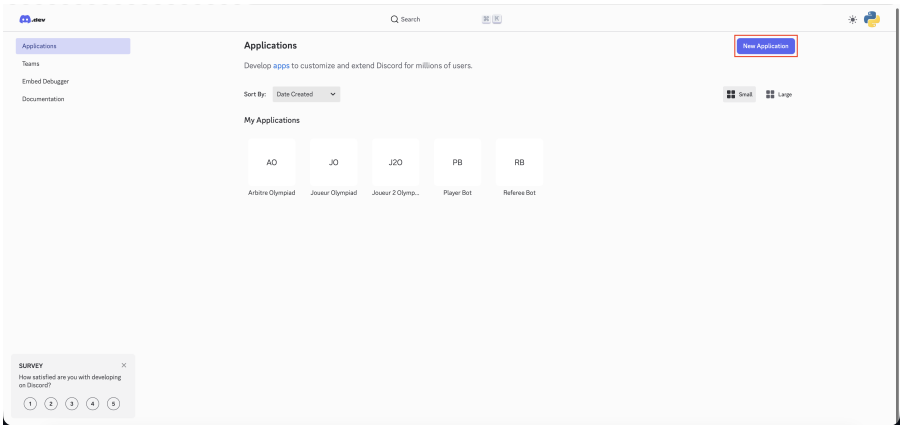

FIGURE 11 – New application button (red box)

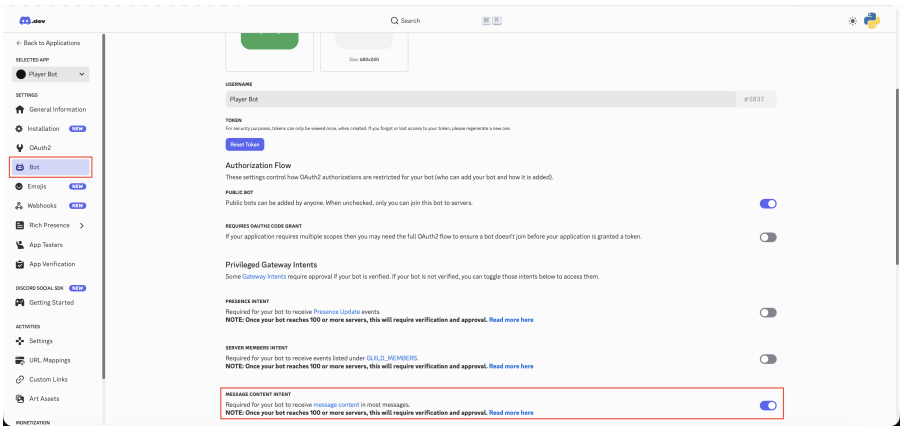

FIGURE 12 – Bot window and message content intent option to be activated (red boxes)

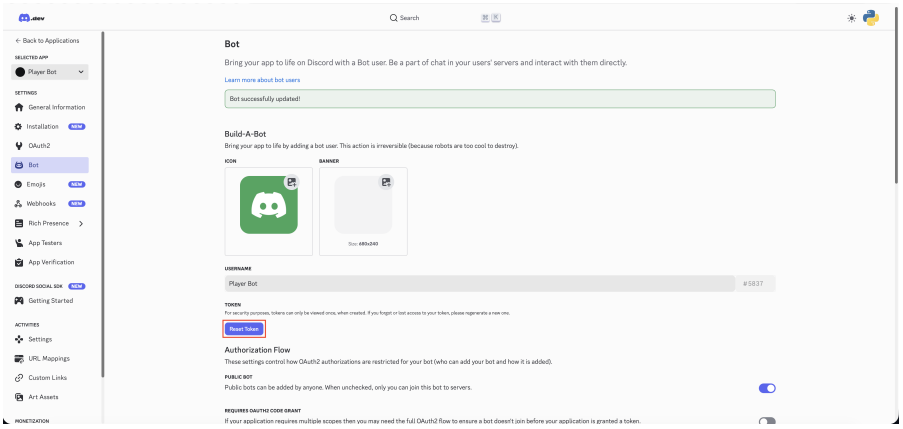

FIGURE 13 – Reset Token button (red box)

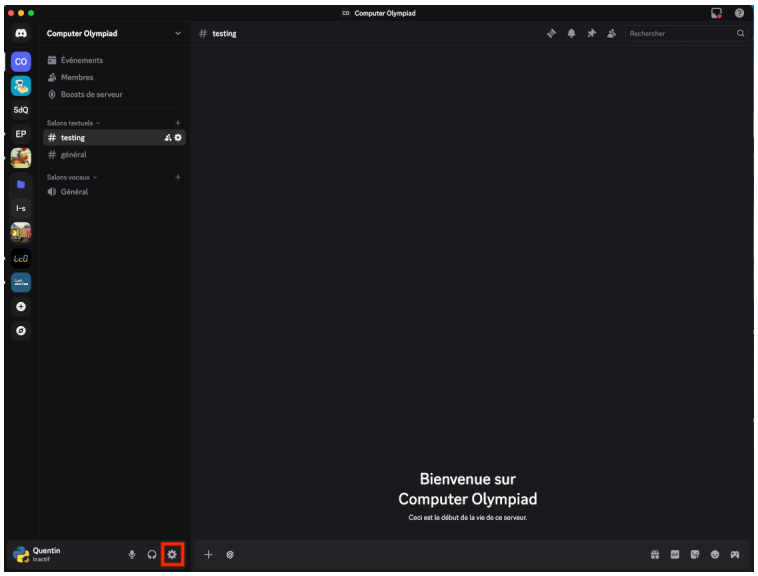

FIGURE 14 – Discord parameters button (red box)

2.2.2 Récupération de votre identifiant Discord (optionnel)

Ensuite, il est suggéré de récupérer votre identifiant Discord pour le spécifier dans le fichier de configuration. C'est optionnel, mais cela permet d'utiliser des commandes contrôlant votre bot pour accéder à certaines fonctionnalités. Par exemple, c'est requis pour pouvoir reprendre un match en cours après un crash (voir la Section 2.5). Nous décrivons dans cette section comment récupérer votre identifiant Discord.

La première chose à faire est d'activer le mode développeur de son compte discord. Pour se faire :

1. Ouvrir l'application Discord
2. Allez dans les paramètres (cliquez sur le bouton "engrenage", si besoin voir la Figure 14).
3. Cliquez sur l'onglet "Advanced" .
4. Activez l'option "developer mode".
5. Cliquez en bas à droite sur votre nom.
6. En bas, dans la mini-fenêtre, cliquez sur "Copier l'identifiant de l'utilisateur" (si besoin voir la Figure 16).
7. Retournez dans les paramètres et désactivez l'option "developer mode".

2.2.3 Spécification du fichier de configuration

Nous décrivons maintenant comment spécifier le fichier de configuration, "parameters.conf" situé à la racine du dossier "discord_interface", qui permet au programme de bot d'accéder à Discord.

1. Spécifier la valeur de "PLAYER_BOT_DISCORD_TOKEN" par le TOKEN du compte du bot (Section 2.2.1).
2. Optionnellement, spécifier la valeur de "OWNER_ID" par votre identifiant Discord (Section 2.2.2).

Si vous souhaitez vérifier que le fichier de configuration est correctement spécifié, lancez votre bot et faites :

```
cd <discord_interface_path>/discord_interface
python3 main_random_ai.py
```

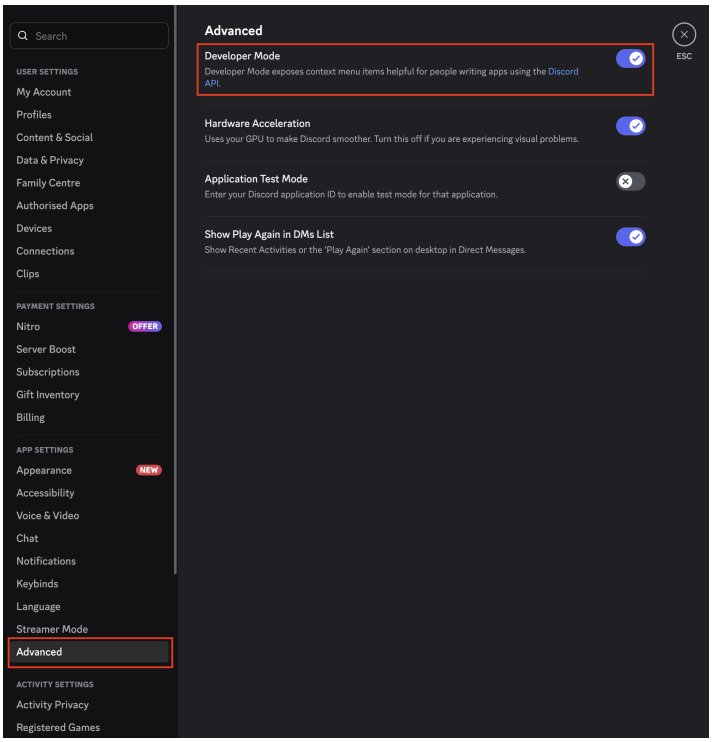

FIGURE 15 – Discord advanced window and developper mode option (red boxes)

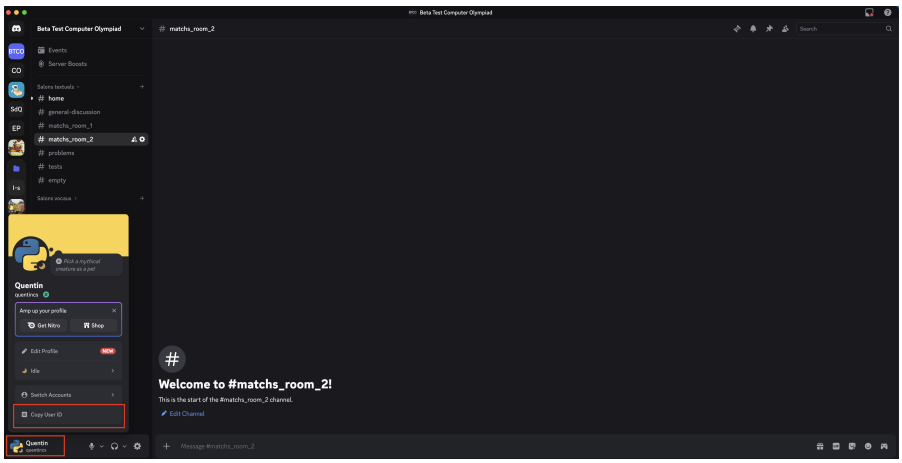

FIGURE 16 – Open owner window and copy Owner ID (red boxes)

game methods of any player	
self.game.get_numpy_board()	return a numpy array representing the game board
self.game.get_current_player()	returns i if the current player is the player i ($i \in \mathbb{N}$)
self.game.textual_legal_moves()	list of valid actions in text format (e.g. "A1" or "A2-B3")
self.game.ended()	return True si the game is ended
self.game.winner	return the index i of the winner player ($i \in \mathbb{N}$)
self.game.textual_plays(move)	apply the move in the current game state (i.e.self.game)
self.game.undo()	undo the last action of the current game state (i.e.self.game)

TABLE 1 – Elementary methods of any Player Python object for programming an AI player.

S'il y a dans le terminal l'erreur "Bad OWNER_ID", il faut donc corriger la variable OWNER_ID dans "parameters.conf".

Pour un test avancé, lancez votre bot et écrivez dans un salon du serveur discord de la compétition :

" !conf_test".

— Si rien ne se passe, contactez-moi.
— Si tout marche bien, "All is OK !" est inscrit dans le terminal.
— Si une erreur se déclanche dans le terminal (discord.errors.LoginFailure : Improper token has been passed) : le PLAYER_BOT_DISCORD_TOKEN a été mal spécifié.

2.3 Programmation d'un bot Discord grâce à Python

Nous décrivons maintenant comment programmer un bot Discord en Python. Il faut hériter une des classes Python "Player" proposées dans la librairie du DCOI. Nous décrivons ici la manière la plus simple de le faire.

Programmation du bot Pour programmer votre bot, il faut hériter la classe BasicPlayer du fichier "discord_interface/player/model/basic_player.py". La seule chose que vous avez à faire est de définir la méthode my_plays(self, game_history, time_left=inf, opponent_time_left=inf). Cette méthode doit retourner l'action que veut jouer votre programme dans l'état courant. L'état courant peut-être reconstruit à partir du paramètre game_history, qui contient la liste des actions au format textuel joué depuis le début de la partie. Il est également possible d'utiliser les méthodes de jeux natives fournies par le DCOI, ce qui permet de ne pas avoir besoin d'implémenter le jeu. Dans ce second cas d'usage, vous aurez besoin des méthodes décrites dans la Table 1.

Faites attention à la valeur du paramètre "time_left", il indique le temps qu'il reste à votre programme pour le reste du match. Si cette valeur atteint 0, vous perdez la partie.

Un exemple d'héritage est disponible dans la classe TextualRandomAI du fichier

"discord_interface/player/instances/textual_random_ai.py".

Ce dernier est un bot qui joue aléatoirement sauf s'il y a un coup gagnant parmi les actions disponibles. Il le joue alors.

Lancement du bot Une fois le compte de votre bot créé, le fichier de configuration spécifié en conséquence, et la classe Player hérité, il ne reste qu'à lancer votre bot dans un fichier main. Pour cela, il faut utiliser la méthode bot_starting(AI_Class) du fichier discord_interface/player/model/bot_launcher.py avec pour AI_Class votre AI Player. Un exemple de programme lanceur de bot est disponible dans le fichier

"discord_interface/main_random_ai.py".

Il ne vous reste plus qu'à faire "python3 votre_main.py" dans le dossier "discord_interface". Par exemple :

```
cd <discord_interface_path>/discord_interface
python3 main_textual_random_ai.py
```

Le bot attend désormais que la commande !s @<Bot_name_1> @<Bot_name_2> soit lancée et que le bot arbitre affiche les informations de la partie. A ce moment là, le bot effectue son initialisation puis ajoute son pouce. La partie commence et le bot joue automatiquement lorsque c'est à lui, jusqu'à la fin de la partie.

Fonctionnalités avancées Pour finir, optionnellement, si vous avez besoin de fonctionnalités avancées, comme un code à effectuer à l'initialisation ou à la terminaison du match, vous pouvez hériter de la classe AdvancedBasicPlayer (fichier discord_interface/player/model/advanced_basic_player.py) et ainsi définir les méthodes my_initialisation(game_name) et my_end().

2.4 Utilisation du bot Discord Joueur générique grâce au GTP

Nous décrivons maintenant comment réaliser des matchs de manière automatique en utilisant GTP. Des informations à propos de GTP sont disponibles par exemple sur ce site : `https://www.lysator.liu.se/~gunnar/gtp/gtp2-spec-draft2/gtp2-spec.html`. Nous verrons dans la Section 2.4.2 comment spécifier un programme GTP. Nous commençons par décrire comment lancer le bot GTP, supposant que nous avons déjà un programme déjà spécifié au format GTP. Pour finir, nous verrons une fonction avancée permettant d'afficher des informations de votre programme dans le terminal (Section 2.4.3).

2.4.1 Lancement du bot GTP

Nous commençons par expliquer les pré-requis pour lancer votre bot GTP. Premièrement, il faut connecter votre programme au bot GTP. Vous avez donc besoin de spécifier trois paramètres :

- "program_name" : le nom de votre programme joueur,
- "program_arguments" : la liste des arguments de votre programme (séparés par des espaces),
- "program_directory" : le chemin absolu de votre programme (ou le chemin relatif au dossier discord_interface, par exemple vous pouvez utiliser une chaine vide si le programme se trouve à la racine du dossier "discord_interface").

Ces variables doivent être spécifiées dans le fichier de configuration "parameters.conf" à la racine du dossier "discord_interface".

Pour exécuter votre programme, vous n'avez plus qu'à faire

"python3 main_gpt_ai_from_conf.py"

dans le terminal depuis la racine du dossier "discord_interface".

Remarque 10. Si vous êtes à l'aise avec Python, vous pouvez aussi modifier et exécuter le fichier "main_gpt_ai.py" pour éviter de toucher au fichier de configuration.

2.4.2 Spécification des commandes GTP

Afin de pouvoir utiliser le bot basé GTP, il faut que votre programme lise dans l'entrée standard les commandes GTP (qui seront envoyées par le bot GTP) et écrive dans la sortie standard le résultat de ces commandes (qui seront ainsi lues et traitées par le bot GTP). Les commandes GTP requises pour faire tourner le bot GTP, à implémenter dans votre programme, sont décrites dans la Table 2. Pour vérifier que la spécification des commandes GTP est correcte, vous pouvez lancer le programme de test suivant dans le terminal : "python3 test_gtp_ai_from_conf.py".

Si besoin, un exemple de spécification en Python des commandes GTP est disponible dans le fichier "discord_interface/player/instances/autres/ia_random_gtp_amazons.py".

GTP command	return	effect
clear_board	=	restart the game
undo	=	cancel the last move
play <player> <move>	=	plays <move> for <player>
time_left <player> <time>	=	inform the time left in seconds
genmove <player>	=<move>	return the <move> to play and play it
quit	=	shutdown the program
player	=<player>	return the current player
game <game>	=	initialization for <game>

TABLE 2 – List of GTP commands to implement, the syntax of their return and their expected effect (<move> is an action in text format (see the syntax in Section 1.3.5) ; <player> is an integer $n \in \mathbb{N}$; <game> is the game name of the match). Remark : commands "game" and "player" are not part of standard GTP ; they are optional (make them only return "=") ; "game" is required only for multi-games participation ; "player" is required only for profiling (see Section 2.5.2)..

2.4.3 Afficher des informations dans le terminal

Pour terminer, nous parlons brièvement d'une option qui permet, lors de l'utilisation du bot GTP, d'afficher dans le terminal des informations provenant de son programme.

La mise en place de cette fonctionnalité est très simple. A chaque return de commande GTP, il suffit de concaténer au return le symbole "#" suivi de la ligne à afficher dans le terminal.

Exemple 11. Si l'on veut afficher dans le terminal "Le coup a été correctement annulé" lorsque la commande "undo" est exécutée, le return du undo doit être "=#Le coup a été correctement annulé".

Notez que la ligne à afficher dans le terminal ne doit pas contenir de saut de ligne. Il est toutefois possible d'effectuer des sauts de lignes avec l'ajout de plusieurs "#". Chaque "#" supplémentaire sera alors interprété comme un saut de ligne.

Exemple 12. Si le return de la commande genmove est "=<move>#Calcul du prochain coup terminé#Coup joué <move>#Temps du calcul 15 secondes", cela affichera dans le terminal :

```
Calcul du prochain coup terminé
Coup joué <move>
Temps du calcul 15 secondes
```

2.5 Fonctionnalités avancées

Nous décrivons maintenant plusieurs fonctionnalités avancées des bots joueurs.

2.5.1 Gestion des crashs

Si pour une raison quelconque votre bot vient à planter et doit être re-exécuté. Il faut pour reprendre le match en cours, après avoir relancé son bot, utiliser la commande

" !continue".

Son alias est " !c".

Avant la compétition, je vous suggère d'effectuer un kill en cours de partie, de relancer et de vérifier que la commande !continue fonctionne correctement pour être sûr de pouvoir reprendre une partie en cas de problème.

Si le bot arbitre ne réagit pas, contactez un responsable de la compétition pour qu'il relance l'arbitre et exécute également la commande " !continue".

2.5.2 Bot profiling

Du profilage peut être effectué avec la commande “ !p”.

Remarque 13. Si vous utilisez un bot GTP, certaines des valeurs fournies seront incorrectes si la commande GTP “player” n’a pas ou a été mal programmée.