INFOB132 - Projet de programmation
Détails pratiques et techniques

Prof. Benoît Frénay

1 Réalisation du projet

Pour votre projet, vous pouvez utiliser les différents outils vus au cours INFOB131. En particulier, les boucles, les structures de données, les fichiers et les algorithmes seront nécessaires. Attention, l'utilisation de `break` et `continue` n'est pas permise. L'utilisation de `input` est permise, mais uniquement pour demander à un joueur humain ce qu'il souhaite jouer à un tour du jeu. N'utilisez pas de programmation objet (par ex. `keys` pour les dictionnaires), sauf accord préalable du professeur pour des cas très spécifiques (par ex. `append` pour les listes [1]). L'utilisation de modules supplémentaires n'est pas permise, sauf accord préalable du professeur, afin de garder intact l'intérêt pédagogique du projet. Des squelettes sont proposés sur WebCampus avec les modules autorisés. Tout projet qui ne respectera pas ces restrictions recevra directement la note 0/20 pour la partie implémentation concernée (qui ne sera pas corrigée).

Pour réaliser votre projet, vous avez à votre disposition des séances encadrées par les assistants qui vous aideront et vous guideront, mais il sera nécessaire de travailler en dehors. Ce cours nécessite au moins une petite dizaine d'heures par semaine durant le quadrimestre. Vous n'aurez pas d'examen "classique" sur ce cours. Par contre, la participation active de chaque étudiant sera vérifiée, notamment grâce à des rencontres à la fin du quadrimestre. La participation aux séances encadrées est obligatoire (2 absences non-justifiées max.).

À la fin de chaque phase, votre solution sera validée sur base d'un rapport. Réfléchissez soigneusement à la façon dont vous allez structurer les données et écrivez soigneusement les spécifications de vos fonctions. Une fois vos spécifications validées par les assistants, vous pourrez passer à l'implémentation. Il peut être utile de dessiner un graphe explicitant quelle fonction appelle quelle autre fonction (et le joindre au rapport). Pour travailler efficacement, vous pouvez alterner travail individuel et périodes de mise en commun en groupe.

1. Une liste `my_list = []` peut être créée vide et ensuite remplie en utilisant `my_list.append('first')`, `my_list.append('second')`, `my_list.append('third')`, etc. Un élément peut être supprimé d'une liste ou d'un dictionnaire avec l'instruction `del`, par ex. `del my_list[2]` ou `del my_dict['first']`. Des fonctions telles que `pop`, `index`, `count`, `remove`, `insert`, `keys` ou `items` ne sont normalement pas nécessaires pour le projet.

2 Délivrables attendus et critères d'évaluation

L'ensemble des délivrables de votre projet sont décrits dans le document "Méthodologie". Le planning proposé vous permet d'avancer progressivement, mais les dates de remise doivent être respectées. Si aucune heure n'est indiquée, **le délivrable est à rendre par défaut à 20:00** (début de soirée, pas d'extension de deadline). Tout retard de dépôt sera sanctionné par un zéro. À la fin du projet, l'ensemble du code doit être fonctionnel et conforme à ses spécifications.

Lorsqu'un rapport est demandé, postez-le sur WebCampus dans l'espace prévu. Les rapports doivent être rendu au format PDF (Libre/Open/Microsoft Office est capable d'exporter en PDF) et s'appeler rapport_gr_xx_part_y.pdf où xx est votre numéro de groupe en deux chiffres (01... 52) et y est le numéro de l'étape (c-à-d le numéro de section : y = 2 pour "design UI et structures de données"). La convention est similaire pour les modules (`engine_gr_xx.py` et `AI_gr_xx.py`) et les vidéos (`test_engine_gr_xx.mp4` et `test_AI_gr_xx.mp4`). Chaque rapport doit indiquer le numéro du groupe et les noms de ses membres. Respectez scrupuleusement ces conventions, sinon vos délivrables ne seront pas corrigés et votre IA ne pourra pas participer pas au tournoi.

Le projet rendu par les étudiants sera jugé sur la qualité de la solution proposée, en termes de spécifications et de documentation, de découpages en fonctions, de l'utilisation adéquate des boucles, du choix des structures de données et de la lisibilité du code. Il est également attendu que toutes les fonctionnalités demandées soient implémentées. Les rapports seront jugés sur leur contenu, mais une forme (orthographe, grammaire, etc.) de qualité suffisante est attendue. Les étudiants d'un même groupe auront la même note qui comptera pour 45% de la note du cours. Une rencontre sera organisée avec certains étudiants au mois de mai pour vérifier leur participation active. Cette rencontre servira à attribuer 50% de la note du cours : ces 50% seront soit égaux à la note de groupe si l'étudiant a démontré sa participation active, soit égaux à 0 si l'étudiant n'a pas été capable de démontrer qu'il a participé activement. Les derniers 5% de la note sont attribués à l'analyse réflexive (cf. document annexé à l'énoncé). La participation aux séances encadrées est obligatoire : deux absences non-justifiées sont autorisées au maximum, sinon l'étudiant recevra 0/20 comme note pour le cours.

Le tournoi en fin de projet ne comptera pas dans la note finale. L'IA de chaque groupe sera évaluée sur base du rapport et du code rendu par le groupe. La note attribuée pour l'IA reflètera la qualité des algorithmes et de l'implémentation. Une IA simple, mais bien expliquée et implémentée sera préférée à une IA complexe, mais incompréhensible et implémentée n'importe comment.

3 Conseils pour la réalisation de l'interface

L'interface utilisateur (*user interface* en anglais, UI) de votre jeu détermine comment il est affiché et comment le joueur peut interagir avec. Dans le cas de ce projet, l'interaction est restreinte à un seul moment : si un des joueurs est humain, il doit à chaque tour donner ses ordres. Le jeu démarre en appelant la fonction principale `play_game`, ensuite il n'est plus nécessaire de toucher au jeu.

Pour afficher le plateau de jeu, vous devrez travailler en "mode texte". Vous ne pouvez utiliser que des chaînes de caractères pour construire l'UI : pas de 3D, d'interface graphique avec plein de boutons, etc. Mais pour cela, vous avez deux outils puissants : l'UTF8 et les séquences d'échappement ANSI.

D'une part, vous pouvez utilisez des symboles UTF8, répartis en plusieurs blocs (cf. par ex. `https://www.fileformat.info/info/unicode/block`) regroupant des caractères correspondant à des alphabets non-latins (arabe, cyrillique, chinois, etc.), des symboles géométriques (rectangles, cercles, etc.), des smileys, des animaux, des objets (bateau, voiture, etc.), etc. La figure 1 en montre quelques exemples. Soyez créatifs et utilisez ces possibilités infinies ! Lors de l'utilisation des caractères UTF8, prenez garde que certains d'entre eux prennent deux emplacements (ils sont deux fois plus larges) à l'écran.

D'autre part, vous pourrez utiliser des séquences d'échappement ANSI (cf. par ex. `https://en.wikipedia.org/wiki/ANSI_escape_code`). Tout comme `'\n'` pouvait être intégré dans une chaîne de caractère pour forcer un retour à la ligne, les séquences d'échappement ANSI peuvent y être intégrées pour changer la position du curseur, la couleur des caractères ou d'arrière-plan (derrière ceux-ci), effacer une ligne, etc. Heureusement, il existe en Python un module très pratique pour gérer cela : `blessed` (cf. `https://pypi.org/project/blessed`). En particulier, les commandes suivantes vous seront utiles :

- `term.home` permet de positionner le curseur au début de l'écran
- `term.clear` permet d'effacer tout l'écran (à utiliser avec `term.home`) ;
- `term.clear_eol` permet d'effacer une ligne à partir du curseur ;
- `term.red3` et `term.on_skyblue` permettent de changer les couleurs utilisées (cf. `ttps://blessed.readthedocs.io/en/latest/colors.html`)
- `term.move_yx` permet de positionner le curseur à un emplacement précis.
- `term.height` et `term.width` donnent la largeur et la hauteur de l'écran ; pour une liste complète de toutes les couleurs disponibles) ;

Pour les couleurs (et d'autres fonctionnalités), n'utilisez pas les autres fonctions de `blessed`, cela compliquerait les choses. Les coordonnées du curseur commencent à $(0, 0)$ en haut à gauche du terminal. Un exemple d'utilisation de `blessed` est donné sur WebCampus et il est déjà intégré dans le squelette fourni. En particulier, l'instruction `term = blessed.Terminal()` permet d'utiliser `term` dans vos fonctions.

Si vous le préférez, le module `colored` (cf. `https://pypi.org/project/`

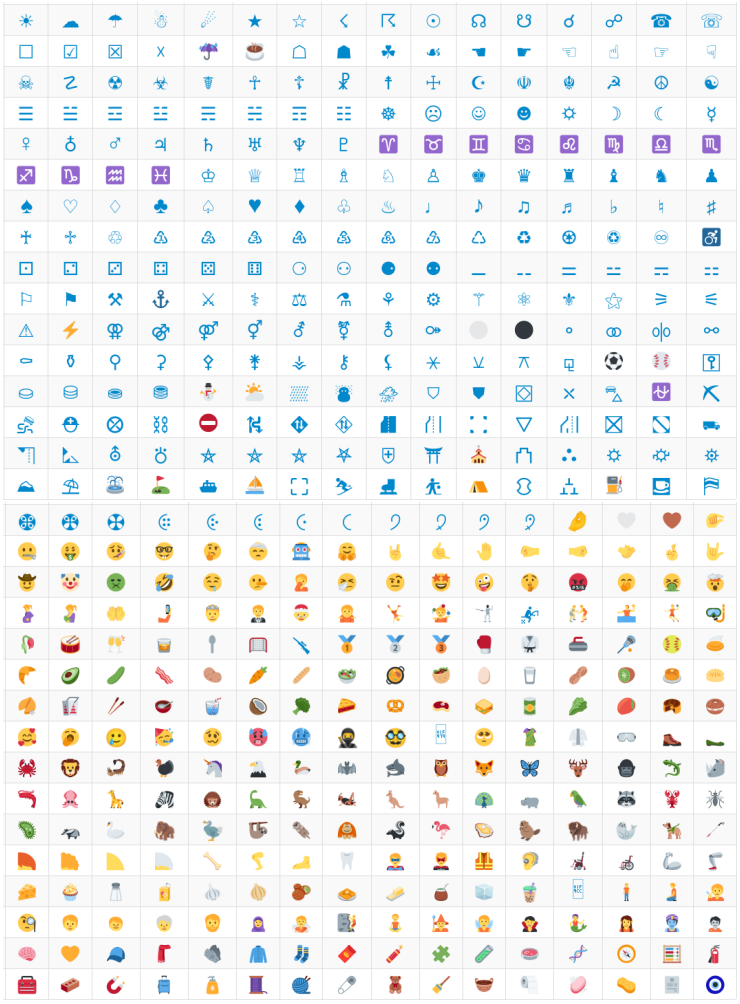

FIGURE 1 – Exemples de caractères UTF8.

`colored`) peut également être utilisé à la place de `blessed`. Il fonctionne différemment et ne permet pas de positionner le curseur à un endroit de l'écran, uniquement de changer les couleurs. Son utilisation est moins intuitive comme il faut afficher le plateau comme une seule grosse chaîne de caractères à chaque tour, mais il pose parfois moins de problèmes de compatibilité. Un exemple d'utilisation de `colored` est donné sur WebCampus. Ces deux modules doivent être installés avant toute utilisation, une explication est donnée sur WebCampus.

Attention, aussi bien l'UTF8 que les séquences d'échappement ANSI nécessitent de travailler sous Visual Studio Code, intégré à la distribution Anaconda. Sous Linux, cela fonctionne également dans n'importe quel terminal. Attention toutefois que l'apparence des caractères UTF8 est différente d'une système d'exploitation à l'autre (Windows, Linux, MacOS). Sous Windows, certains caractères UTF8 peuvent apparaître légèrement trop larges, mais cela disparaitra lorsque vous testerez votre programme au pool informatique (via X2Go).

Pour éviter les problèmes des années précédentes, quelques précisions :

- n'incluez pas de "menu" dans votre jeu, puisque vous ne pouvez pas utiliser `input` pour y naviguer et faire des choix ;
- vous ne pouvez pas utiliser d'outils comme Tkinter pour construire votre UI avec des composants graphiques (pas de boutons et autres widgets) ;
- vous ne pouvez utiliser que des caractères, éventuellement UTF8 pour avoir des symboles spéciaux ou colorés via `blessed` / `colored` ;
- évitez d'utiliser les séquences d'échappement `\t` et `\v` pour votre UI, leur comportement est parfois étrange, en particulier sous Windows ;
- ne vous souciez pas du temps pris par un joueur pour jouer (cela complexifierait l'implémentation et demanderait idéalement l'utilisation de threads, que vous n'avez pas vus au cours, pour être faite proprement) ;
- lorsque vous concevez votre IA, vérifiez tout-de-même qu'elle ne prenne pas trop de temps pour réfléchir (tout au plus quelques secondes) ;
- soyez très stricts avec le format des ordres, respectez-le à la lettre (en particulier, n'ajoutez pas d'espace **après** la séquence d'ordres).

4 Utilisation du pool informatique

Pour pouvoir participer au tournoi, votre programme est censé fonctionner sous Linux lorsqu'il sera lancé depuis un terminal. Pour cela, il faudra vous connecter au pool informatique. Si vous faites cela à distance (pour travailler chez vous ou si que l'accès au campus est interdit), la procédure est décrite dans le document `https://info.unamur.be/doc/x2go` qui utilise l'outil X2Go (`https://wiki.x2go.org/doku.php`). Lorsque vous codez, n'utilisez pas de notebook, mais travaillez dans un module avec un IDE comme Visual Studio Code.

5 Connection à un autre groupe via `remote_play`

Comme expliqué dans le document "Méthodologie", votre jeu devra pouvoir se connecter au jeu d'un autre groupe. Le but sera de pouvoir faire jouer votre IA contre celle d'un autre groupe (appelé "joueur distant"). Dans votre jeu, il ne peut y avoir qu'un joueur distant géré par le programme d'un autre groupe. Pour simplifier votre travail, un module `remote_play` est mis à votre disposition. Vous trouverez ci-dessous la documentation de `remote_play`, celui-ci est déjà intégré dans les squelettes sur WebCampus. Il est fortement conseillé de ne pas utiliser `other_IP`, à moins d'avoir des notions de base en réseaux.

```
def create_connection(your_group, other_group=0, other_IP='127.0.0.1', verbose=False):
    """Creates a connection with a referee or another group.

    Parameters
    ----------
    your_group: id of your group (int)
    other_group: id of the other group, if there is no referee (int, optional)
    other_IP: IP address where the referee or the other group is (str, optional)
    verbose: True only if connection progress must be displayed (bool, optional)

    Returns
    -------
    connection: socket(s) to receive/send orders (dict of socket.socket)

    Raises
    ------
    IOError: if your group fails to create a connection

    Notes
    -----
    Creating a connection can take a few seconds (it must be initialised on both sides).

    If there is a referee, leave other_group=0, otherwise other_IP is the id of the other grou

    If the referee or the other group is on the same computer than you, leave other_IP='127.0.
    otherwise other_IP is the IP address of the computer where the referee or the other group

    The returned connection can be used directly with other functions in this module.

    """


def close_connection(connection):
    """Closes a connection with a referee or another group.

    Parameters
    ----------
    connection: socket(s) to receive/send orders (dict of socket.socket)

    """


def notify_remote_orders(connection, orders):
```

```
    """Notifies orders to a remote player.

    Parameters
    ----------
    connection: sockets to receive/send orders (dict of socket.socket)
    orders: orders to notify (str)

    Raises
    ------
    IOError: if remote player cannot be reached

    """


def get_remote_orders(connection):
    """Returns orders from a remote player.

    Parameters
    ----------
    connection: sockets to receive/send orders (dict of socket.socket)

    Returns
    -------
    player_orders: orders given by remote player (str)

    Raises
    ------
    IOError: if remote player cannot be reached

    """
```

Pour pouvoir jouer avec un joueur distant, vous devez d'abord initialiser la communication avec celui-ci. Pour cela, la fonction `create_connection` nécessite vos deux numéros de groupe et devra être appelée une seule fois au début de la partie. Cela signifie que vous devez décider à l'avance de qui est le joueur 1 ou 2. La fonction `close_connection` n'est appelée qu'une seule fois à la fin de la partie. A chaque fois que qu'il faudra recevoir les ordres d'un joueur distant, vous devez utiliser la fonction `get_remote_orders`. Après chacun des ordres de l'humain ou de l'IA que votre programme gère, vous devez notifier les ordres qu'il a choisi au joueur distant grâce à la fonction `notify_remote_orders`. Pour le tournoi, la seule différence est que vous devrez indiquer que le numéro de l'autre groupe est 0. L'arbitre comprendra automatiquement ce qu'il doit faire.

N'utilisez pas les fonctions `create_server_socket`, `create_client_socket`, `wait_for_connection` et `bind_referee` également présentes dans `remote_play`.