INFOB132 - Projet de programmation
Méthodologie détaillée

Prof. Benoît Frénay

1 Contexte

Depuis 5 mois, vous avez (re)découvert la programmation et fait vos preuves en résolvant divers problèmes. Confiant dans vos capacités, votre chef de projet vous propose de relever un challenge : créer un jeu original et le doter d'une puissante intelligence artificielle (IA). Fort de vos expériences précédentes, vous acceptez. Cependant, prudent, vous demandez plus de détails à votre chef (vous commencez à le connaître et supputez que la tâche ne sera pas simple) et lui demandez de vous adjoindre deux collègues pour vous aider dans votre tâche.

2 Déroulement du projet

Vous disposez de trois mois pour votre projet. Soucieux que vous fassiez les choses proprement et conscient que c'est tentant (mais rarement efficace) de directement coder, votre chef insiste pour que vous respectiez le timing suivant :

30/01 - 05/02 : découverte du projet

30/01 - 12/02 : design UI et structures de données

13/02 - 19/02 : découpage et spécification des fonctions

20/02 - 26/03 : implémentation du jeu avec IA naïve

13/03 - 19/03 : réflexion algorithmique pour l'IA

27/03 - 30/04 : implémentation et test de l'IA

01/05 - 12/05 : tournoi opposant les IAs

Attention : les dépôts **ne sont pas toujours** le lundi. Une séance encadrée par les assistants est prévue chaque semaine. À vous de vous organiser en dehors des séances prévues pour avancer (indispensable !). N'hésitez pas à commencer l'étape suivante si (et seulement si) vous avez terminé l'étape en cours. Vous pouvez évidemment discuter entre groupes, mais le plagiat (par ex. copier des parties de code ou de rapport) n'est pas permis. Votre travail doit être le vôtre (et donc ne peut être fait, intégralement ou en partie, par une autre personne ou par un programme informatique), sinon vous n'apprendrez rien...

Si vous le souhaitez, un carnet de bord est disponible sur WebCampus pour vous aider (non-obligatoire) à prendre note pendant vos séances de groupe.

2.1 Découverte du projet (30/02 - 05/02)

Durant cette phase, vous constituerez votre groupe, prendrez connaissance du projet à réaliser et discuterez ensemble de l'organisation de votre travail. Cela implique par exemple le choix d'un mode de fonctionnement (but des réunions, timing du projet, politique en cas de retard/absence, partage d'idées, etc.), des outils que vous utiliserez, de la répartition des rôles (par ex. tournante pour le secrétariat), etc. Après la première séance, vous rendrez un court "règlement de groupe" (max. 1 page) détaillant vos choix et accepté par tous les membres du groupe. Durant le projet, chaque étudiant sera responsable [1] d'une partie du projet ; la répartition doit être clairement indiquée dans ce document. Les groupes doivent impérativement être constitués de quatre étudiants de la même cohorte (INFO/INGMI). La constitution des groupes est libre, mais chaque groupe doit être inscrit sur WebCampus pour le 29 janvier à 20h00. Lors de la première séance, faites chacun une liste des règles du jeu et comparez-la entre vous : le but est de vérifier que vous en avez la même compréhension.

- **29/01** : deadline pour encoder les groupes sur WebCampus
- **02/02** (INFO) ou **04/02** (INGMI) : séance de kickoff du projet
- **06/02** : dépôt du règlement de groupe et de l'analyse réflexive individuelle décrite dans l'annexe de l'énoncé (WebCampus)

Comme pour INFOB131, Python sera utilisé pour l'implémentation. Les conventions sont les mêmes, sauf si spécifié dans l'énoncé. Tous les détails techniques sont décrits dans le document "Détails pratiques et techniques".

2.2 Design UI et structures de données (30/01 - 12/02)

Durant cette phase, vous réfléchirez à l'interface utilisateur (*user interface* en anglais, UI) et aux structures de données que vous utiliserez. L'UI est simplement la façon dont vous demanderez à un joueur ce qu'il joue et comment vous afficherez le plateau de jeu. Vous rendrez une courte explication avec un croquis (à la main ou avec un programme de dessin) montrant à quoi l'UI ressemblera. Comme pour INFOB131, utilisez des chaînes de caractères pour construire l'UI (voir Figure 1) : pas de graphismes en 3D, d'interface graphique avec des boutons, etc. Par contre, vous pouvez utilisez des symboles UTF8 (par ex. formes, smileys, symboles, etc.) et changer les couleurs d'avant-plan (les caractères) ou d'arrière-plan (le fond) (voir le document "Détails pratiques et techniques").

Les structures de données sont le "trait d'union" entre les fonctions que vous spécifierez et implémenterez, il est donc essentiel de les penser soigneusement. Vous rendrez une courte explication détaillée de vos structures de données et un croquis (à la main ou avec un programme de dessin) montrant à quoi vos structures de données ressembleront en pratique. Le rapport fera max. 2 pages.

1. Autrement dit, il doit jouer le rôle de leader pour cette phase (UI + structure de données, découpage des fonctions, implémentation sans IA, algorithmique IA ou implémentation IA).

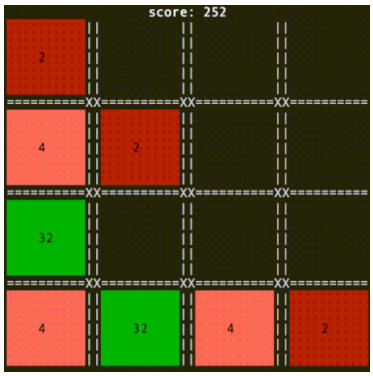

FIGURE 1 – Exemple d'UI (source : `https://pypi.org/project/blessed`).

— **13/02** : dépôt des deux parties du rapport (WebCampus)

Conseil : réfléchissez à **comment** vous utiliserez vos structures de données ! Autrement dit, songez un peu aux algorithmes que vous allez devoir concevoir.

2.3 Découpage et spécification des fonctions (13/02 - 19/02)

Durant cette phase, vous découperez votre projet en différentes fonctions. Chacune sera soigneusement spécifiée selon le formalisme du cours INFOB131. En outre, vous ajouterez à chaque spécification une clause "Version" qui mentionnera clairement l'auteur et la date de la spécification selon le format suivant :

```
def my_function(...):
    """Description of the function.

    ...
    ...
    ...

    Version
    -------
    specification: Isaac Asimov (v.1 03/03/16)

    """
```

L'auteur d'une spécification en est responsable. Toute modification majeure d'une spécification après cette phase devra être approuvée par un assistant en lui remettant la nouvelle version. La date dans la clause "Version" devra être modifiée en conséquence. À chaque "cycle de modification", incrémentez la version (`v.1` → `v.2` → ···). Prenez donc soin de réfléchir soigneusement à chaque spécification. À la fin de cette phase, vous rendrez un court document expliquant le découpage de vos fonctions et les choix de spécification que vous avez

faits (max. 1 page). Vous y inclurez un graphe montrant comment les fonctions s'appellent entre elles. Les spécifications seront rendues dans un **module** Python nommé `engine_gr_xx.py`. Attention, un module n'est pas un notebook.

— **20/02** : dépôt du rapport et du module (WebCampus)

2.4 Implémentation du jeu avec IA naïve (20/02 - 26/03)

Durant cette phase, vous implémenterez en Python les différentes fonctions constituant votre projet, conformément à leurs spécifications. Votre jeu devra pouvoir être joué par deux joueurs donnant leurs ordres à chaque tour de jeu sous la forme d'une chaîne de caractères (voir le document "Règles du jeu"), comme aux échecs où la chaîne "e4 e5" indique un déplacement d'un pièce de la case e4 à la case e5. Chaque joueur peut être de trois types différents :

— un joueur humain tape directement au clavier sa chaîne de caractère lorsque votre jeu la lui demande, via l'utilisation de `input` ;
— une intelligence artificielle (IA) qui reçoit l'état du jeu (structures de données) et renvoie une chaîne de caractère qui contient les ordres qu'elle aura trouvés, autrement dit une fonction avec des paramètres et un `return` ;
— l'intelligence artificielle d'un autre groupe (joueur distant) à laquelle votre jeu pourra se connecter et communiquer via un module spécifique (voir le document "Détails pratiques et techniques").

Votre IA sera à ce stade simple (30-50 lignes de code) : le but n'est pas qu'elle joue bien au jeu, mais plutôt qu'elle vous permette de corriger votre code. Ainsi, une IA qui joue aléatoirement (c-à-d jouer n'importe quel coup autorisé par le jeu et choisi au hasard) est intéressante car elle va pousser votre implémentation dans ses retranchements. Attention : l'IA peut aussi bien être le joueur 1 que le joueur 2 (ou même les deux, auquel cas elle jouera contre elle-même). Autrement dit, chaque joueur peut être un humain, une IA ou un joueur distant (même si en pratique, il y aura au plus un joueur distant : celui contre lequel vous jouez).

Vous modifierez les clauses "Version" en mentionnant clairement le(s) membre(s) du groupe ayant participé à l'implémentation selon le format suivant :

```
def my_function(...):
    """Description of the function.

    ...
    ...
    ...

    Version
    -------
    specification: Isaac Asimov (v.1 03/03/16)
    implementation: Frank Herbert, Arthur C. Clarke (v.1 16/03/16)

    """
```

La date indique quand la fonction a été modifiée pour la dernière fois. À chaque "cycle de modification", incrémentez la version (`v.1` → `v.2` → ⋯). À la fin de cette phase, vous rendrez un court document expliquant les difficultés rencontrées pour l'implémentation et vos choix pour les résoudre (max. 1 page). Le code Python implémentant les fonctions de votre projet sera rendu dans le module `engine_gr_xx.py`. En plus, vous rendrez une vidéo nommée `test_engine_gr_xx.mp4` [2] montrant **10 tours de jeu** contre l'IA naïve, ainsi qu'un début de partie où le joueur donne un coup illégal (contraire aux règles).

— **27/03** : dépôt du rapport, du module et de la vidéo (WebCampus)

2.5 Réflexion algorithmique pour l'IA (13/03 - 19/03)

Durant cette phase, vous réfléchirez à la façon dont votre IA jouera. Le but est ici de travailler en groupe à la conception d'un ou de plusieurs algorithmes relativement simples (vous n'avez pas trois mois pour les implémenter !). Ces algorithmes seront utilisés par l'IA pour décider de ses actions à chaque tour de jeu. À la fin de cette phase, vous rendrez un court document expliquant votre stratégie et présentant chaque algorithme composant votre IA (max. 2 pages). Il est fortement conseillé de vous aider de schémas pour vos explications, comme le fait par exemple la partie de l'énoncé qui explique les règles du jeu.

— **20/03** : dépôt du rapport (WebCampus)

2.6 Implémentation et test de l'IA (27/03 - 30/04)

Durant cette phase, vous implémenterez votre IA. Si vos algorithmes s'avèrent trop complexes, inadaptés ou inefficaces (comme indiqué par les assistants ou si vous vous heurtez à de grosses difficultés), vous pouvez les modifier **après** accord de l'assistant. À la fin de cette phase, vous rendrez un rapport expliquant les difficultés rencontrées pour l'implémentation et vos choix pour les résoudre (max. 1 page). Le code de votre IA sera rendu dans un module séparé du module précédent et nommé `AI_gr_xx.py` [3]. En plus, vous rendrez une vidéo nommée `test_AI_gr_xx.mp4`[2] d'une **partie complète** de votre IA contre elle-même.

— **01/05** : dépôt du rapport, du module et de la vidéo (WebCampus)

2.7 Tournoi opposant les IAs (01/05 - 12/05)

Pour s'assurer que le jeu qu'il a proposé soit doté d'une IA puissante, votre chef a demandé à d'autres équipes de travailler sur le même projet. Une fois les codes de chaque groupe remis, un tournoi désignera les meilleures IAs ! Le tournoi sera organisé selon le même format que les compétitions officielles de

2. L'extension `.mp4` est donnée à titre d'exemple et peut être différente en fonction de la façon dont vous créez votre vidéo et du logiciel utilisé (`.mpeg`, `.avi`, `.mkv`, etc.).
3. AI est l'abréviation anglophone de *artificial intelligence*, d'où le nom des fichiers.

jeux vidéos (par ex. les StarCraft II World Championship Series). À chaque tour éliminatoire, chaque IA jouera contre une autre IA et seule la gagnante pourra continuer. Le processus continuera jusqu'à la finale qui déterminera quelle IA mérite la première et la seconde place. Les deux perdants des matchs de demi-finale s'opposeront pour désigner quelle IA mérite la troisième place. Le tournoi sera à double élimination : les perdants s'affronteront pour être "repêchés".

Le document expliquant comment les IAs seront utilisées pour le tournoi vous sera remis plus tard. Si votre IA est impossible à utiliser, génère une exception ou triche, elle sera déclarée perdante et l'autre IA gagnera par forfait. Ce sera également le cas si votre IA propose des ordres invalides (format incorrect, trop/pas assez d'espaces, etc.) qui pourraient poser problème en tournoi. En cas de désaccord entre deux programmes, un arbitre décidera du gagnant. Les parties se joueront sur un mix des cartes réalisées par l'enseignant et des meilleures cartes proposées par les étudiants (choisies de façon totalement subjective ! ☺).

— **05/05** : dépôt des cartes (WebCampus)
— **12/05** : tournoi opposant les IAs (toute la matinée)

Le résultat obtenu après le tournoi par chaque IA n'influencera pas la note : n'hésitez donc pas à participer, même si votre IA n'est pas parfaite. L'expérience des années précédentes a montré que les codes simples (mais bien implémentés et efficaces) sont parfois les meilleurs en tournoi. Ce sera l'occasion de passer un chouette moment tous ensemble pour clôturer en beauté ce projet !