

BZZZ : sujet de SAE 1.01 année 2025-2026



Résumé

Ce document contient le sujet de la SAE 1.01, qui est la première des deux SAE de développement :

- SAE 1.01 : Implémentation d'un besoin client
- SAE 1.02 : Comparaison d'approches algorithmiques

Dans la SAE 1.01, vous allez réaliser l'implémentation d'un jeu en tour par tour nommé BZZZ, dans lequel des colonies d'abeilles s'affrontent pour la conquête du nectar. Vous réaliserez une interface graphique permettant de jouer au jeu contre des adversaires réels ou simulés, en respectant scrupuleusement le cahier des charges fourni, en particulier sur les règles du jeu. Dans la SAE 1.02, on vous fournira un moteur pour le jeu BZZZ et vous réaliserez une IA, c'est à dire un programme qui joue du mieux possible au jeu BZZZ. Cette SAE sera évaluée, en la faisant jouer contre quelques adversaires prédéfinis, ainsi que par un tournoi entre toutes les IAs des participants.

Règles du jeu BZZZ

Vous trouverez les règles du jeu dans le fichier `regles_bzzz.pdf` ci-joint. Il importera de bien comprendre ces règles et de les implémenter très exactement. Il est possible d'ajouter des options avec des variantes de règles, mais dans ce cas un menu doit permettre d'activer/désactiver ces options pour suivre les règles standard.

Objectifs d'implémentation

Votre objectif final pour cette SAE est de livrer un répertoire contenant un programme python qui permette de jouer au jeu BZZZ, qui implémente de façon exacte toutes les règles, soit :

- à plusieurs sur le même ordinateur (mode *hot-seat*),
- contre des adversaires gérés par l'ordinateur (des *IAs*).

Vous devrez livrer une version graphique du jeu, avec des options plus ou moins avancées. Des objectifs progressifs sont détaillés par les niveaux donnés dans ce document. Essayez de réaliser complètement un niveau (modèle ou graphique) avant de passer au suivant, et de façon générale donnez priorité au modèle avant le graphique.

En parallèle, un travail vous sera demandé pour la ressource *anglais*, renseignez-vous auprès de votre enseignante.

Contraintes techniques

Votre programme devra être écrit en python, en utilisant la bibliothèque graphique tkiteasy, ou bien directement tkinter. Le code doit tourner sans autre installation préalable sur les machines de l'IUT.

Structure du code

Les points suivants sont tous très importants et seront évalués :

- Votre programme doit être découpé en fonctions (ou méthodes) de taille raisonnable. Une fonction trop longue (à titre indicatif, plus de 25 lignes) est une fonction qui pourrait probablement être redécoupée en unités fonctionnelles plus significatives.
- Le programme doit être commenté : chaque fonction doit avoir sa docstring, et l'intérieur des fonctions doit être commenté en donnant les grandes étapes du code sans le paraphraser.
- Utilisez des variables globales uniquement pour les constantes comme les dimensions de la fenêtre, la taille des cases, les constantes des règles, etc. En dehors de ces paramètres aucune variable globale ne devrait être utilisée.
- Attention à la répétition de code avec juste quelques variations mineures. *Si du code est dupliqué, c'est qu'il fallait utiliser une fonction avec un paramètre pour indiquer ce qui change.*
- Vos variables et fonctions doivent avoir des noms intelligents qui permettent de comprendre.
- Le moteur de jeu ou modèle (la partie logique et données) doit être séparé au maximum de la partie graphique au niveau des fichiers.
- Pensez ergonomique dans les contrôles du jeu. Pensez aussi que le jour du test on doit arriver rapidement au jeu sans entrer des informations au clavier dans le terminal comme le nom du joueur, etc.

L'évaluation et mise en garde sur la collaboration entre groupes ou avec des IAs

L'évaluation se passera par soutenance. Votre groupe fera une démonstration de son programme devant les enseignants. Nous regardons en détail le code et posons des questions individuelles dessus. La note dépend largement autant de votre maîtrise et la clarté/conception

du code que de l'application en elle-même. Nous chercherons à évaluer ce que vous avez rendu, mais aussi votre maîtrise de ce qui a été rendu. Il y aura également une petite évaluation écrite lors du DS final d'Initiation au Développement.

Le travail en binôme sur le projet est obligatoire. Il s'agit d'une occasion de beaucoup progresser, aussi ne faites pas tout, et ne laissez pas l'autre tout faire ! Vous seriez également pénalisés lors de l'évaluation où nous nous assurons que **chacun des deux maîtrise l'ensemble du projet** par des questions (la réponse "ce n'est pas moi qui ai développé cette partie" *ne sera pas acceptée* et vous pénalisera individuellement).

De même, faites **très attention** à votre utilisation des IAs génératives. Si vous rendez un travail que vous n'êtes pas capables d'expliquer, ce sera sanctionné. Utilisez éventuellement les IAs pour comprendre et vous former, mais créez par vous-même, sinon vous ne progresserez pas.

Niveaux du modèle

Modèle niveau 1

Gestion du total de nectar, du déplacement des abeilles, de la ponte.

Modèle niveau 2

Gestion du reste des règles : fleurs, butinage, escarmouches, et des conditions de fin de partie.

Modèle niveau 3

Un mode "un joueur contre trois IAs" (peu importe la qualité des IAs).

Niveaux graphiques

Graphique niveau 0

Pas d'affichage graphique, on joue dans le terminal.

Graphique niveau 1

Affichage simple des éléments réalisés. On entre les choix au clavier directement.

Graphique niveau 2

Les choix sont maintenant entrés directement à la souris en cliquant sur des menus, sur les cases, etc. Le clavier n'intervient plus du tout. Des informations quantitatives s'affichent au fur et à mesure (par exemple, cliquer sur une abeille affiche ses informations).

Graphique niveau 3

Déplacement automatique : plutôt que de cliquer case par case à chaque tour, on peut indiquer où l'abeille doit se rendre, et à chaque tour elle avancera d'une case. Ceci permet d'automatiser plus facilement le jeu si la carte est grande.

Options supplémentaires

Vous pouvez ajouter des options supplémentaires de votre création. Exemples :

- sauvegardes
- règles additionnelles (autres types d'abeilles, de nectar, laissez aller votre imagination) mais il faut alors pouvoir activer et désactiver ces règles additionnelles avant de lancer la partie.

Recommandations supplémentaires

- Pensez pratique : pour le jour de la soutenance, il faut que nous soyons capables de lancer plusieurs fois le programme, de façon rapide. Evitez les préalables avec plusieurs menus, détails à entrer au clavier etc., avant de lancer le jeu proprement dit.
- De même, pensez "automatisation" et ergonomie comme dans un vrai jeu. Plutôt que de choisir au début de chaque tour si on veut une ponte, il serait préférable de pouvoir le faire à tout moment et avoir une "file d'attente de production d'abeilles". Tout ce qui peut accélérer la manipulation du jeu est bienvenu. L'option graphique niveau 3 de déplacement automatique peut aussi fluidifier le jeu (proposez d'autres choses !)
- Suivez la progression par niveaux proposée dans ce sujet afin que votre programme soit bien structuré. Nous préférons un programme bien écrit, stable, et qui va moins loin dans les niveaux, plutôt qu'un programme très sophistiqué mais mal conçu, pas clair et et plein de bugs.

Les échéances

- Vous déposez votre projet sur le dépôt e-campus sur le cours *Initiation au Développement* avant la date du **11 Janvier 2026**. Il doit être accompagné d'un README de quelques pages permettant de le prendre en main (comment le lancer, quelles sont les commandes, etc).
- Des soutenances de projet auront lieu le mardi **12 Janvier 2026** . Environ 10 à 15 minutes par binôme, nous regardons les projets, le code des projets, et votre maîtrise de ce code.

Les compétences qui seront évaluées sur ce projet seront les suivantes (il peut y avoir des changements mineurs) :

- Respect du cahier des charges de dépôt (deadline, README, archive nettoyée et au bon format, programme qui fonctionne directement sans erreurs et sans rien installer de nouveau sur les machines de l'IUT)
- Respect du cahier des charges du programme : jusqu'où vous êtes allés dans les niveaux.
- Spécifications et tests: chaque fonction doit documenter sa spécification, et les fonctions critiques du modèle doivent être accompagnées de leur fonction de test (test unitaire) dans un fichier séparé.
- La qualité du code : commentaires, découpage en fonctions, constantes, nommage correct...

- Considérer tous les cas particuliers d'un programme (ex: tel paramètre fait-il planter le programme?). Votre programme doit être robuste.
- Les options ajoutées, modèle et graphiques.
- Incorporer des solutions techniques innovantes : vous avez utilisé des algorithmes ou structures de données bien pensées.

Et maintenant ?

Go ! N'hésitez surtout pas à questionner vos enseignant.e.s si vous avez des questions. De plus, un salon dédié à la SAE permettra de dialoguer sur Discord. Bon travail ! Amusez-vous bien...

SAE 1.02 : Comparaison d'approches algorithmiques

La SAE 1.02 commencera quand la SAE 1.01 sera terminée pour une raison simple : nous vous fournirons un moteur de jeu, c'est-à-dire un programme python capable de simuler une partie de BZZZ (en mode sans graphique). Les joueurs seront des IAs que vous écrirez. Ce que vous écrirez en pratique c'est une méthode, par exemple `joueurIA(self, game : GameData) -> str` qui étant donnée une structure de donnée `GameData` qui décrit l'intégralité de l'état du jeu à un moment donné, doit prendre une décision et renvoyer la ou les actions que fait le joueur à ce tour. Cette méthode fera partie d'un objet `BZZZ_IA` permettant d'avoir une mémoire persistante entre les tours. Vous travaillerez sur ce projet le moment venu mais vous pouvez d'ores et déjà réfléchir aux meilleures stratégies lors de votre premier projet.