



SA1.02 COMPARAISON D'APPROCHES ALGORITHMIQUES

1 Compétences ciblées

Cette SAÉ vise à travailler sur la **compétence 2** *Appréhender et construire des algorithmes*.

Les apprentissages critiques visés sont les suivants :

- **AC12.01** Analyser un problème avec méthode (découpage en éléments algorithmiques simples, structure de données...)
- **AC12.02** Comparer des algorithmes pour des problèmes classiques (tris simples, recherche...)

Cette SAÉ se fera en groupe de 3 ou 4. Ces groupes vous sont imposés. Il y aura plusieurs rendus à faire :

- un rendu le **mercredi 18 janvier 8h** sur l'implémentation des fonctions des API demandées,
- un rendu le **vendredi 20 janvier 8h** sur l'IA que vous aurez développée,
- une soutenance le **vendredi 20 janvier matin** au cours de laquelle vous présenterez votre travail.

La SAÉ se terminera par un tournoi le **vendredi 20 janvier** après midi où vos IA s'affronteront.

2 Présentation du sujet

Splat IUT'O est un jeu librement inspiré et largement simplifié de Splatoon (voir figure 1). Le jeu se joue sur un plateau où les cases sont

- soit des couloirs où les joueurs peuvent se déplacer,
- soit des murs qui ne sont pas franchissables.





Chaque joueur est associé à une couleur et son objectif est d'arriver à peindre un maximum de cases avec sa couleur. Pour cela à chaque tour, les joueurs doivent prendre deux décisions

- choisir entre peindre dans une direction ou ne pas peindre,
- se déplacer sur une case adjacente (le déplacement est obligatoire).

Les joueurs ont une réserve de peinture. Cette réserve diminue à chaque fois que le joueur lance de la peinture. Repeindre une case non peinte ou déjà peinte par le joueur coûte une unité dans la réserve. Repeindre une case déjà peinte par un autre joueur coûte deux unités de la réserve. Les murs ne peuvent pas être repeints sauf dans le cas où on obtient un objet spécial (voir plus loin). Lorsque le joueur touche d'autres joueurs lors de son action, il leur *vole* des unités de peinture qui iront dans sa réserve. Toucher un joueur qui n'a plus de peinture ne rapporte rien. La portée du jet de peinture est limitée.

Pour remonter sa réserve de peinture un joueur doit se déplacer sur des cases qui sont peintes de la couleur de ce joueur. Se déplacer sur une case peinte d'une autre couleur que la sienne fait baisser la réserve de peinture.

Au cours du jeu des objets spéciaux peuvent apparaître de manière aléatoire sur le plateau. Pour obtenir un objet, il suffit de se déplacer sur la case où se trouve l'objet. Le joueur qui obtient un objet gagne un bonus en unités de peinture. Un joueur ne peut avoir qu'un seul objet à la fois et cet objet a une durée de vie limitée. Voici la liste des objets et leur pouvoir :

- La bombe  : cet objet permet de peindre dans toutes les directions en commençant par celle choisie par le joueur et en tournant dans le sens horaire.
- Le pistolet  : cet objet permet de peindre les murs. L'avantage de peindre les murs est qu'il est plus difficile à un autre joueur de repeindre ces murs puisqu'il doit trouver lui-même un pistolet.
- Le bouclier  : cet objet permet au joueur d'être protégé si un autre joueur lui lance de la peinture en évitant d'être volé.
- Le bidon  : Cet objet permet aux joueurs qui ont une réserve négative de la remettre à 0. Le bidon a un effet immédiat et si le joueur possédait déjà un objet, il ne le perd pas.

Lors de la création de la partie la durée (en nombre de tours) de celle-ci est fixée ainsi que tous les paramètres de bonus et malus. Le tableau ci-dessous donne les valeurs qui seront utilisées lors du tournoi.

Durée partie	200	Durée vie objet	5	Réserve initiale	20
Bonus recharge	3	Bonus joueur touche	5	Bonus objet	5
Pénalité	2	Portée peinture	5		

Au début de la partie, les joueurs sont placés aléatoirement sur le plateau. La partie se fait tour par tour où chaque joueur effectue une action de peinture et un déplacement. Afin de ne pas privilégier un joueur par rapport aux autres, l'ordre des joueurs est pris aléatoirement à chaque tour. A la fin de la partie, c'est le joueur qui a le plus de cases peintes de sa couleur qui gagne. En cas d'égalité c'est la réserve des joueurs qui les départage.

3 Principes du serveur de jeu

L'objectif de la SAÉ n'est pas d'implémenter entièrement le jeu mais plutôt de programmer une intelligence artificielle permettant de défier les autres équipes lors du tournoi.

Voici la structure du répertoire `SAE_splat_iuto` contenu dans l'archive

- le sujet en pdf,
- le fichier `EQUIPE` que vous devrez remplir pour donner un nom à votre groupe et indiquer les membres de votre projet,
- tous les fichiers sources et les fichiers de tests,
- un répertoire `images` contenant les images utilisées par la partie graphique,
- un répertoire `cartes` qui contient différentes cartes utilisés par par le serveur de jeu,
- un répertoire `plans` qui contient les différents plans utilisés dans les tests.

La différence entre les plans et les cartes est que les cartes ne contiennent que la structure du labyrinthe alors que le plan contient d'autres informations comme la position des joueurs et des trésors.

Le jeu du tournoi repose sur un serveur de jeu qui effectue les actions suivantes :

- crée un jeu à partir d'une carte qui définit les murs et les couloirs du plateau,
- attend l'enregistrement de 4 joueurs,
- lance la partie.

Au cours de la partie, le serveur

- envoie à chaque joueur, l'état du plateau et des joueurs,
- attend les ordres envoyés par chaque joueur,
- applique les ordres de chaque joueur en tirant au sort l'ordre de passage.

Chaque joueur est un programme client qui va se connecter au serveur. Un exemple de programme client vous est fourni dans le fichier `client_joueur.py`. Toute la partie connexion et communication avec le serveur vous est fournie, vous *n'aurez qu'à* implémenter la fonction `mon_IA` pour déterminer les ordres à envoyer au serveur.

Etant donné qu'un joueur doit indiquer à chaque tour de jeu uniquement la direction où il veut peindre, et la direction vers où il veut aller, il est très facile de faire un joueur complètement aléatoire (voir le fichier `client_joueur.py`).

Lancement d'une partie

Pour lancer une partie, il faut ouvrir au moins deux fenêtres dans le répertoire du projet. Dans la première, vous lancez le serveur avec la commande `python3 serveur.py` dans la première fenêtre. Dans la seconde, vous allez lancer le client d'affichage puis quatre joueurs

```
1 python3 affichage.py&
2 python3 client_joueur.py --equipe joueur1&
3 python3 client_joueur.py --equipe joueur2&
4 python3 client_joueur.py --equipe joueur3&
5 python3 client_joueur.py --equipe joueur4&
```

Enfin dans la fenêtre du serveur vous pouvez appuyer sur la touche **Entrée** pour lancer la partie.

4 Objectifs de la SAÉ

Comme indiqué précédemment, l'objectif de la SAÉ est d'aller au-delà du joueur aléatoire et de construire une IA qui prenne des décisions en fonction de l'état du plateau à un instant donné. Pour cela, il va falloir décoder ce que le serveur envoie car toutes les informations sont fournies sous forme de chaînes de caractères. Il faudra aussi stocker ces informations dans des structures de données adéquates. Ensuite il faudra écrire des fonctions qui vous permettent de connaître l'état du jeu et prendre des décisions.

Ne perdez pas de vue qu'au-delà de l'efficacité de votre IA, la SAÉ sert à évaluer les apprentissages critiques rappelés section 1.

1er rendu : Implémentation des API

Pour la première partie (décodage et stockage de l'information), il vous est demandé d'implémenter les API suivantes :

- `joueur.py` qui gère les informations liées à un joueur donné (son nom, sa réserve de peinture etc.).
- `case.py` qui gère une case du plateau avec les objets et joueurs qui sont dessus.

— `plateau.py` qui gère un tableau en deux dimensions de cases.

Notez bien que certaines fonctions de ces API ne sont pas strictement indispensables à la création de l'IA mais peuvent malgré tout vous aider à comprendre la structure de données.

Pour implémenter ces API vous pouvez choisir la représentation que vous souhaitez (mais les dictionnaires sont fortement conseillés). Des jeux de tests vous sont fournis afin que vous puissiez vérifier que vos fonctions sont correctes. La spécification des fonctions est indiquées dans les fichiers Python. Cette partie sera notée en fonction des résultats des tests.

Des détails sur le codage des informations vous sont donnés section 5



Affichage graphique

Le script `affichage_case.py` et `affichage_plateau.py` fournis avec le sujet vous permettent d'avoir un retour visuel du contenu de vos structures. Pour configurer ces scripts, il faut se placer tout à la fin. Dans le script `affichage_plateau.py` plusieurs plan de plateau sont proposés.

2eme rendu : Définition de l'IA et son implémentation

Afin d'implémenter votre IA vous allez devoir établir une stratégie qui va reposer sur une observation de l'état du plateau. Il va donc falloir définir un certain nombre de fonctions qui vont donner des informations vous permettant de prendre des décisions. Une bonne partie de ces fonctions vont reposer sur la connaissance des distances entre un point du plateau et les autres cases de ce plateau (voir TP12). Voici un certain nombre d'informations que vous pourriez avoir besoin pour votre IA.



Quelques idées de fonctions d'observation

En fonction de ma couleur et de ma position sur le plateau :

- Quelle direction prendre pour atteindre l'objet le plus proche ?
- Quelle direction prendre pour atteindre la case la plus proche peinte d'une certaine couleur ?
- Combien de joueurs se trouvent à portée de peinture dans une certaine direction (c'est-à-dire que je peux toucher ou qui peuvent me toucher) ?

Evidemment ce ne sont que des exemples mais vous pouvez imaginer toute sorte de stratégies qui auront besoin d'autres informations. Vous pouvez aussi avoir des stratégies différentes en fonction de votre réserve de peinture ou de la durée restante etc.

Pour cette partie, votre rendu sera composé

1. un rapport décrivant
 - la stratégie de votre IA
 - les principaux algorithmes que vous aurez implémentés. Pour chaque algorithme, il faudra donner une approximation de sa complexité.
 - un état de ce que vous avez réussi à faire et de ce qui ne marche pas
2. l'implémentation de votre IA et des fonctions d'observation. Chaque fonction devra être documentée par une docstring.

Une partie de la notion portera sur la clarté de vos implémentations et de la documentation.

Conseil

Commencez par mettre en place une stratégie simple puis enrichissez la au fur et à mesure. Faites bien des sauvegardes à chaque fois que vous avez une version qui fonctionne.

5 Codage des informations

Comme indiqué précédemment, le serveur de jeu envoie à chaque tour de jeu l'état du plateau et des joueurs sous la forme d'une chaîne de caractères. Cette section indique le codage utilisé pour décrire ces informations.

Dans le jeu les couleurs sont représentées par des lettres de l'alphabet en partant de 'A'. Les objets sont représentés par un nombre en 0 et 4, 0 indiquant aucun objet. La signification des autres nombres est définie dans le fichier `const.py`.

Les directions sont exprimées par un caractère : E pour Est, S pour Sud, O pour ouest, N pour Nord et X pour indiquer pas de direction.

5.1 Le plateau

Pour vous indiquer l'état du plateau, le serveur de jeu enverra un texte de la forme de celui qui est à la gauche de la figure 1. La représentation graphique de ce plateau est donnée sur la partie droite de la figure 1.

```
9;9
#AAAa ##
    #B#
#    B##
#  #B #
#  #B
# ## ##
#
#  # #
# #   ##
2
A;0;1
B;4;4
1
3;3;1
```

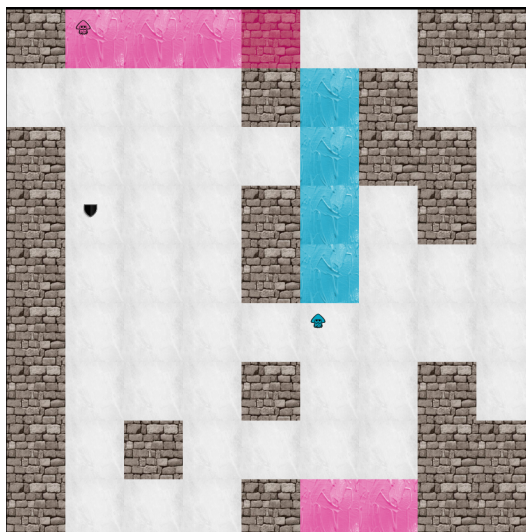


FIGURE 1 – Le codage d'un plateau et sa représentation graphique

La première ligne du texte indique le nombre de lignes et le nombre de colonnes du plateau. Les lignes suivantes représentent le plateau avec la convention suivante :

- # indique que cette case est un mur non peint,
- une lettre minuscule indique que cette case est un mur peint avec la couleur correspondant à la lettre (comme la case (0,4) qui est un mur peint en rose),

- une lettre majuscule indique que cette case est un couloir peint avec la couleur de la lettre,
- un espace indique que cette case est un couloir non peint.

Pour compléter les informations, la liste des joueurs et leur position et la liste des objets et leur position sont fournies après la description du plateau. Ainsi le 2, indique qu'il y a deux joueurs. Le premier a la couleur A et se situe en position (0,1). Le second est de couleur B et se situe sur la case (4,4). Enfin le 1 indique qu'un seul objet est sur le plateau, il a pour identifiant 3 et se situe en (3,1).

Dans l'API, les couleurs des peintures sont représentées par la lettre majuscule correspondant à cette couleur. Par extension les joueurs seront identifiés par la lettre majuscule correspondant à sa couleur. Les objets seront représentés par leur identifiant.

5.2 Les joueurs et leurs caractéristiques

Dans le codage du plateau, seule les positions des joueurs et des objets sont indiquées mais il n'y a aucune information concernant la réserve de peinture des joueurs par exemple. Le serveur de jeu vous enverra les informations des joueurs sous la forme d'un texte comme ci-dessous :

```
A;15;4;2;3;0;1;le peintre
B;7;3;0;0;4;4;l'artiste
```

Chaque ligne décrit un joueur. Les informations sont séparées par des ; et sont données dans l'ordre suivant : couleur;reserve;surface;objet;duree_objet;ligne;colonne;nom

5.3 Les caractéristiques générales du jeu

Ces informations sont données afin de permettre aux IA de connaître des informations qui peuvent être importantes dans la prise de décision. Elles sont données sous la forme d'une ligne dont les informations sont séparées par des ; dans l'ordre suivant :

```
duree_ecoul;duree_tot;reserve_init;duree_obj;penalite;bonus_touche;bonus_recharge;bonus_objet;dist_max
```

duree_ecoul est la durée écoulée depuis le début de la partie, **duree_totale** est la durée totale de la partie (la partie s'arrête quand **duree_act** est égale à **duree_totale**). **reserve_init** indique la réserve initiale des joueurs, **duree_obj** précise le nombre de tours qu'un joueur peut utiliser un objet qu'il a pris. **penalite** indique le nombre d'unités de peintures que coûte le fait d'effectuer une action inconnue ou illégale (se déplacer dans un mur par exemple) ou de se déplacer sur une case peinte par un autre joueur. **bonus_touche**, **bonus_recharge**, **bonus_objet** sont les nombres d'unités de peinture que le joueur gagne si respectivement, il touche un autre joueur avec sa peinture, il se déplace sur une case peinte de sa propre couleur et il prend un objet. Enfin **dist_max** indique jusqu'à combien de cases peut porter un tir de peinture.

6 Rappel des rendus



Important

Pour chaque rendu, vous rendrez une archive `.zip` de votre répertoire `SAE_splat_iuto` sans changer son nom.

La SAÉ va se dérouler en trois grandes étapes :

1. **18 janvier 9h** : Rendu des API joueur, case et plateau
2. **20 janvier 8h** : Rendu du répertoire implementation avec une IA fonctionnelle (même si elle est presque complètement aléatoire). Ce répertoire devra contenir votre rapport contenant les informations indiquées section 4
3. **20 janvier** : Soutenance (15 min) au cours de laquelle vous présenterez votre travail et votre organisation
4. **20 janvier après midi** : Tournoi des IA !