M1102: Introduction à l'algorithmique et à la programmation

Projet n°1

1 Présentation

L'objectif de ce projet est de programmer un jeu inspiré à la fois du jeu du Labyrinthe [1] et de Splatoon [2].

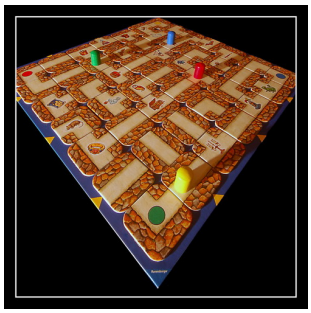


Figure 1 – Le plateau du labyrinthe

Ce jeu se joue à deux, trois ou quatre joueurs sur un plateau du Labyrinthe (voir figure 1), c'est-à-dire une grille 7×7 sur laquelle on place des cartes cartonnées. Chacune de ces cartes représente un morceau de labyrinthe. Certaines cartes sont fixes (les quatre coins, puis une case sur deux sur tout le plateau), les autres cartes sont amovibles. Au début du jeu on dispose de manière aléatoire les cartes amovibles sur le plateau. Il y a une carte amovible de plus que de places sur le plateau. Chaque joueur se voit attribuer un pion de couleur qu'il doit placer dans le coin correspondant à cette couleur et il dispose d'une réserve de peinture. L'objectif de chaque joueur est d'essayer de peindre un maximum de cases du labyrinthe avec sa couleur. Pour ce faire, à chaque tour de jeu, un joueur doit

1. peindre dans une direction
2. choisir entre
   — se déplacer vers une case voisine ou
   — modifier le labyrinthe.

1. `https://fr.wikipedia.org/wiki/Labyrinthe_(jeu)`
2. `https://fr.wikipedia.org/wiki/Splatoon`

1.1 Peindre dans une direction

Lorsque le joueur peint dans une direction (Nord, Est, Sud ou Ouest), un jet de peinture part de la case où se trouve le joueur dans la direction choisie. Le jet est arrêté par les murs ou l'extrémité du labyrinthe : toutes les cases rencontrées sont recouvertes de peinture dans la limite de la réserve de peinture du joueur. Lorsque le joueur touche un autre joueur dans cette phase, il lui *vole* trois unités de peinture de sa réserve. Un joueur peut choisir de ne pas peindre (pour économiser de la peinture).

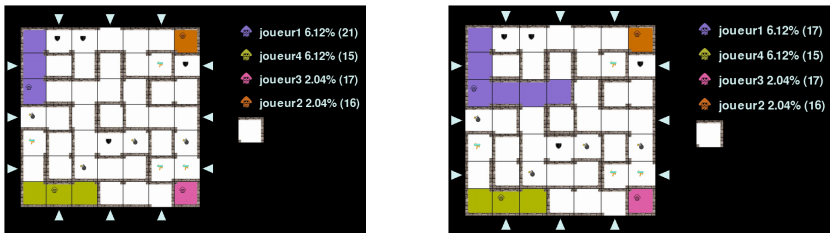


FIGURE 2 – Peindre vers l'est

1.2 Se déplacer

Lorsqu'un joueur se déplace, il ne peut pas traverser les murs du labyrinthe. S'il arrive sur une case de sa propre couleur, sa réserve de peinture augmente de 1, s'il arrive sur une case peinte par un autre joueur, il perd une unité de peinture de sa réserve.

1.3 Modifier le labyrinthe

Le joueur peut tourner la carte amovible supplémentaire et choisir un endroit pour insérer cette carte. Pour placer cette carte amovible, il pousse la ligne ou la colonne choisie d'un cran et récupère la carte *expulsée* par ce décalage. Si un joueur se trouvait sur la carte expulsée, ce joueur est placé sur la carte insérée. Si la carte expulsée était peinte elle redevient neutre. Sur la figure 3, le joueur 1 a insérer la carte amovible au sud dans l'avant dernière colonne. Le joueur rose s'est fait expulsé et donc se retrouve sur la carte qui vient d'être insérée. De plus, la carte sur laquelle il se trouvait, est la nouvelle carte amovible. Elle était peinte en rose et a perdu sa couleur.
La seule restriction pour la modificaiton du labyrinthe est l'interdiction de remettre la carte amovible à l'emplacement d'où elle vient d'être expulsée.

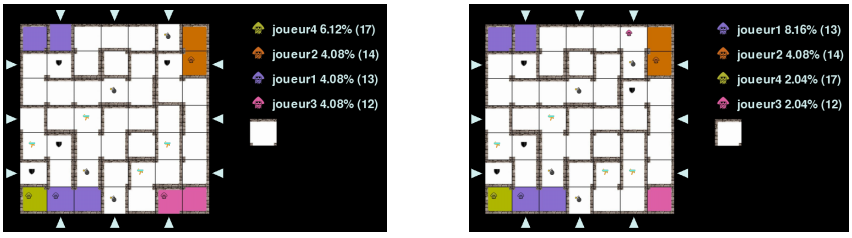


FIGURE 3 – Insérer au sud en avant dernière colonne

1.4 Les objets

Il y a trois types d'objets qui permettent au joueur qui les obtient d'avoir des pouvoirs supplémentaires :

— La bombe 💣 : lorsque le joueur possède cet objet, sa peinture est envoyée dans toutes les directions possibles, en commençant par la direction choisie et en tournant dans le sens des aiguilles d'une montre.
— Le pistolet 🔫 : lorsque le joueur possède cet objet, sa peinture peut traverser un mur ; c'est-à-dire qu'au lieu que son jet de peinture soit arrêté par le premier mur rencontré, il est arrêté par le second.
— Le bouclier 🛡 : lorsque le joueur possède cet objet et qu'il est touché par un autre joueur, il ne perd pas de peinture de sa réserve et l'autre joueur n'en gagne pas.

Les objets sont disposés de manière aléatoire sur le plateau au début de la partie. Un joueur obtient un objet en se déplaçant sur une carte possédant cet objet. La seule exception est quand le joueur arrive sur une carte possédant un objet suite à une modification du labyrinthe. Lorqu'il gagne l'objet, le joueur augmente sa réserve de peinture de 5 unités. Par contre s'il avait un objet celui-ci disparait. L'objet acquis est valide pendant 10 tours puis il disparait.

1.5 Fin de la partie

La partie se joue en un certain nombre de tours (100 par exemple). A la fin de la partie, c'est le joueur qui possède le plus de cartes de sa couleur qui a gagné. En cas d'égalité, c'est celui qui a la plus grande réserve de peinture qui l'emporte.

2 Travail à faire

Vous devrez développer individuellement un certain nombre de fonctions qui vous seront données au fur est à mesure des semaines. Ces fonctions devront vous permettre d'avoir un jeu fonctionnel pour la semaine de la rentrée en janvier. Une deuxième phase va consister à implémenter par groupe de quatre une intelligence artificielle pour faire jouer un *bot* au jeu du Splaby'O. Les IA s'affronteront lors d'un tournoi le 22 janvier.

3 Structure du projet

Le structuration du projet vous est imposée. Elle s'organise en plusieurs grandes parties qui vous permettront de proche en proche d'arriver à l'implémentation finale du jeu. Cette structuration vous permettra d'utiliser le script d'affichage en mode graphique qui vous est fourni. Le schéma présenté figure 4 représente la dépendance entre les différentes parties du projet. Une documentation HTML des fonctions à implémenter vous est fournie dans le répertoire `html`. Vous pouvez y acceder en ouvrant le fichier `index.html` dans un navigateur. Avant de commencer, les couleurs sont représentées par des chaines de caractères. Comme celles choisies pour le jeu proviennent des logos des différents départements de l'IUT'O, elles se nomment `chimie`, `gea`, `gmp`, `gte`, `informatique` et `qlio`. L'absence de couleur est représenté par la couleur nommée `aucune`.

3.1 Les joueurs

Un joueur possède un nom, une couleur et une réserve de peinture (qui se compte en unités). De plus, on stockera la surface du labyrinthe qu'il a recouverte. Cette surface se compte en nombre de cases. Un joueur peut posséder un objet (les objets sont des entiers entre 1 et 3), si c'est le cas, on mémorise le nombre de tours restants avant que cet objet ne disparaisse. Enfin un joueur a un type qui est soit ordinateur, soit humain. Si c'est un ordinateur, le

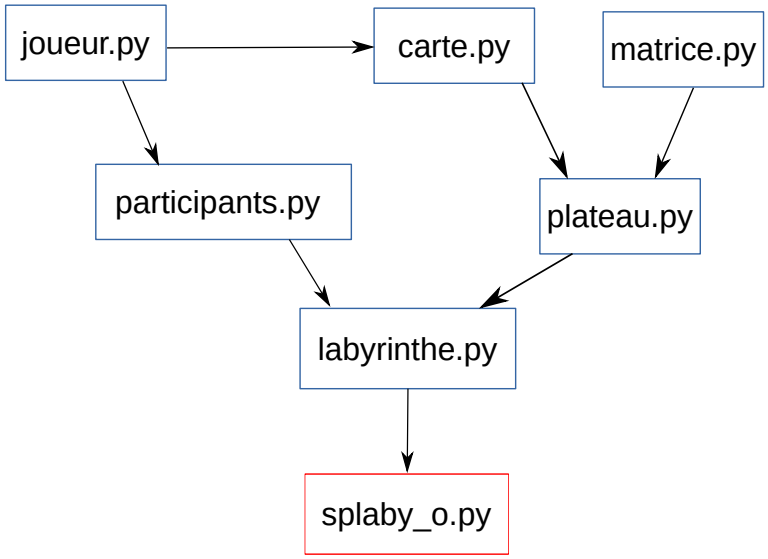


FIGURE 4 – Architecture du projet

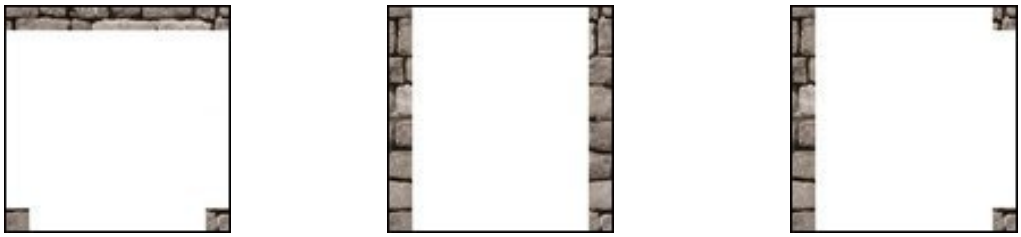
joueur aura une intelligence artificielle (IA) qui sera représentée par une fonction. Par défaut l'IA qui lui est attribuée est la fonction `dummy_ia` qui retourne des ordres aléatoires.
La fonction `Joueur` crée un joueur avec ses caractéristiques. Les fonctions de ce module sont simplement des fonctions qui permettent de modifier les caractéristiques du joueur ou de les consulter. La fonction `comparer` permet de comparer deux joueurs en fonction de la surface qu'ils ont recouverte et de leur réserve de peinture. Les spécifications des fonctions sont données dans le documentation. Notez bien que toutes les fonctions de ce module à part la fonction `Joueur` et les deux dernières, ne doivent pas dépasser 2 lignes de code.
Vous pouvez choisir la représentation que vous souhaitez pour vos joueurs, par contre, les différentes fonctions doivent être implémentées et **répondre aux spécifications demandées**.

3.2 Les participants

Les participants sont représentés par une structure qui permet de stocker la liste des joueurs qui participent à la partie mais qui doit aussi gérer le joueur courant et la notion de tour (c'est à dire détecter que tous les joueurs ont joué une fois). La fonction `Participants` crée une liste de joueurs à partir d'une liste de noms de joueurs et une liste de couleurs. Cette fonction va donc créer les joueurs (grâce à la fonction `Joueur` du module `joueur`. Par défaut les joueurs sont des ordinateurs avec l'IA par défaut. Les joueurs sont numérotés de 1 à 4 dans l'ordre de leur création. Si un humain participe à la partie, ce sera le joueur 1.
Les fonctions de ce module permettent de modifier et observer la liste des joueurs.
Vous pouvez choisir la représentation des participants que vous souhaitez à condition d'implémenter toutes les fonctions demandées **en suivant leur spécification**.

3.3 les cartes

Dans le module `carte.py` vous trouvez la signature des fonctions qui vous sont demandées pour gérer les cartes. Une carte se caractérise par la présence ou non de murs sur ses quatre directions : nord, est, sud et ouest (voir figure 5).
De plus, une carte peut contenir un objet et entre 0 et quatre joueurs. Elle peut aussi être



(a) Une carte ne possédant qu'un mur au nord (b) Une carte possédant des murs à l'est et à l'ouest (c) Une carte ne possédant qu'un mur à l'ouest

FIGURE 5 – Exemples de cartes

peinte d'une des couleurs des joueurs. Les opérations que l'on peut faire sur une carte sont les suivantes :

- mettre ou enlever un joueur
- mettre ou enlever un objet
- mettre ou enlever une couleur de la carte
- faire tourner la carte dans le sens horaire (ou anti-horaire). Par exemple la carte de la figure 5(c) est le résultat de la rotation dans le sens anti-horaire de la carte de la figure 5(a).

Par ailleurs, quatre fonctions permettent de savoir si il y a un passage entre deux cartes contiguës du plateau. Vous pouvez choisir la représentation que vous souhaitez pour vos cartes, par contre, les différentes fonctions doivent être implémentées et **répondre aux spécifications demandées**.

3.4 La matrice

Il s'agit simplement d'une version étendue des matrices que vous avez manipulées en TD. Seules des fonctions permettant d'effectuer des décalages de lignes ou de colonnes sont à implémenter. Un certain nombre de fonctions concernant les décalages dans une ligne ou une colonne sont ajoutées à l'API de base.
Vous pouvez choisir la représentation des matrices que vous souhaitez à condition d'implémenter toutes les fonctions demandées **en suivant leur spécification**.

3.5 Le plateau

Il s'agit simplement d'une matrice (telle que définie dans la section 3.4) de cartes (telles que définies dans la section 3.3). Dans le fichier `plateau.py`, on va gérer l'initialisation du plateau de jeu avec un placement aléatoire des cartes. Cette structure de données gère aussi les joueurs et les objets posés sur le plateau afin de les repérer, les poser ou les enlever. Enfin les fonctions de test d'accessibilité dans le labyrinthe se trouvent également dans ce fichier.
Vous ne devez pas implémenter de structures de données particulières pour le plateau, le plateau est une matrice de cartes.
Voici quelques précisions pour l'initialisation du plateau. Il y a trois types de cartes différentes dans le jeu

- les angles ╔ au nombre de vingt,
- les jonctions ╠ au nombre de dix-huit et
- les tout-droits ═ au nombre de douze.

La figure 6 représente le plateau avec ses cartes fixes. Dans cette figure, on peut voir comment sont repérées les 4 directions du plateau et la numérotation des lignes et des colonnes. Les quatre coins seront les points de départ respectifs des quatre joueurs.

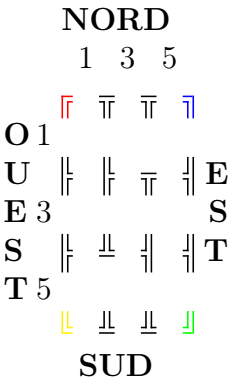


FIGURE 6 – Position des cartes fixes

3.6 Le labyrinthe

Les fonctions de `labyrinthe.py` permettent de gérer le jeu du labyrinthe. Concernant la structure de données, la seule contrainte est que le plateau d'un labyrinthe doit être une matrice de cartes et doit contenir des participants stockés dans la structure que vous avez définie dans la section 3.1.
C'est dans ce fichier que vous trouverez les fonctions qui implémentent les différentes actions que peuvent faire les joueurs (peindre, se déplacer etc.) ainsi que la fonction `finir_tour` qui permet de faire les mises à jour des participants et changer de joueur courant.

3.7 Splaby'O en mode graphique

Dans le fichier `splaby_o.py` vous n'avez rien à implémenter. Ce script utilise la bibliothèque `Pygame` (`www.pygame.org`).
Si toutes vos fonctions respectent les spécifications données, votre labyrinthe en mode graphique doit fonctionner sans modifier ce fichier.
Il vous suffira de taper `./labyrintheGraphique.py` pour jouer. Attention cependant à ce que le répertoire `images` soit bien présent et contiennent les images requises. Pour changer le nom et la couleur des joueurs vous pouvez changer les lignes 437 et 438, pour le fait d'admettre un joueur humain ou non, le nombre de tours de la partie, c'est la ligne 440

4 Rendu

Les instructions pour le rendu final vous seront données ultérieurement, ainsi que celles pour implémenter les IA. En attendant, vous devez implémenter de manière individuelle les modules

- `joueur.py` (6 décembre)
- `participants.py` (6 décembre)
- `carte.py` (13 décembre)
- `matrice.py` (20 décembre)
- `plateau.py` (4 janvier)

Des fichiers de tests vous seront fournis très prochainement.