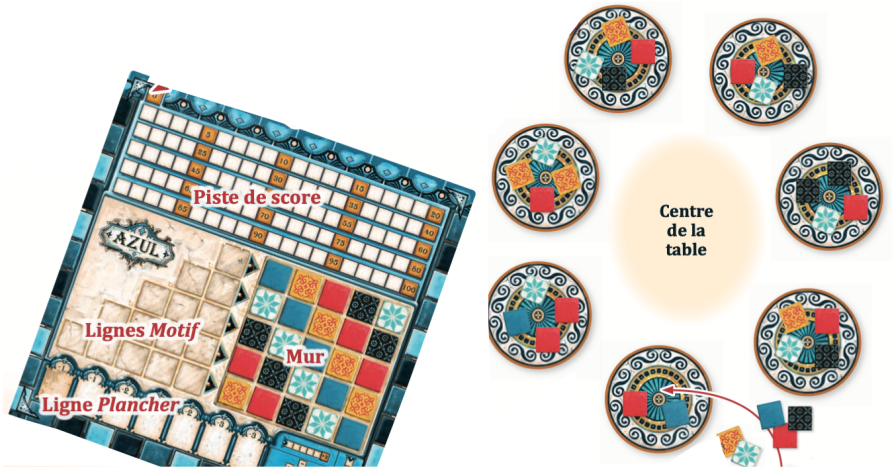
Projet : AZUL

Le but du projet est de réaliser une version pour ordinateur du jeu de plateau Azul. Il s'agit d'un jeu abstrait, de 2 à 4 joueurs, où l'on doit décorer les murs d'un palais avec des petits carreaux de faïence appelés "azulejos". La règle du jeu en pdf est fournie avec ce sujet. Si vous n'avez jamais joué, vous pouvez tester une version *online* non officielle qui se trouve ici : `http://boardwebgames.com/rojo/`.

Important : Vous devez lire les règles de jeu avant de lire la suite du sujet.

1 Le programme à réaliser



On veut obtenir une version numérique de ce jeu, qui doit pouvoir se jouer à plusieurs sur un seul ordinateur (en utilisant la même souris). Pour cela, vous allez créer une fenêtre qui affichera les plateaux des joueurs (avec les lignes de motifs, le mur et la ligne de plancher, ainsi que le score courant), les fabriques et une zone correspondant au centre de la table. Tour à tour, chaque joueur devra pouvoir sélectionner des tuiles (en fait une seule suffit pour indiquer l'ensemble des tuiles choisies) et les placer sur une de ses lignes de motifs. À chaque fin de manche, le jeu actualise le mur de chaque joueur et affiche le score courant (avec le détail du calcul pour cette manche). Et lorsque le jeu est terminé, le programme affiche le vainqueur.

L'objectif final est d'obtenir un jeu proche du jeu officiel, mais nous allons vous guider un peu dans les étapes pour arriver à ce résultat. Vous allez réaliser votre jeu en 3 phases et chaque phase devra être terminée et fonctionnelle avant de pouvoir passer à la suivante. Cependant, il est important de garder en tête ce que sera le jeu final pour faire vos choix de conception.

1.1 Phase 1 : l'offre des fabriques

Dans un premier temps, vous allez créer une seule manche du jeu, sans la partie "décoration du mur". Vous commencerez par une version à deux joueurs : soit deux humains, soit un humain contre l'ordinateur qui joue au hasard.

Pour vous aider : sur e-learning, vous avez un exemple de programme qui produit une interface simple utilisant la souris pour sélectionner une zone de la fenêtre graphique.

À la fin de la phase 1, on doit avoir les fonctionnalités suivantes :

1. Le jeu se met en place avec un sac de 100 tuiles et prépare une manche en effectuant le remplissage des fabriques (leur nombre dépend du nombre de joueurs, voir la règle).
2. À son tour, chaque joueur peut sélectionner des tuiles, soit dans une fabrique, soit dans la zone du centre. En fait, il suffit de choisir une seule tuile pour désigner l'ensemble des tuiles de cette couleur dans la fabrique ou le centre.
3. Lorsque les tuiles sont sélectionnées, il peut alors choisir sur quelle ligne de motif elle seront placées, à condition que ce choix respecte les règles du jeu. Si ça déborde, les tuiles restantes sont placées sur la ligne plancher.
4. Ensuite, le jeu met automatiquement à jour les fabriques et la zone du centre et passe la main au joueur suivant.
5. Le programme s'arrête lorsque plus aucune tuile n'est disponible.

Conseils :

- N'essayez pas de produire des effets de type *drag-and-drop*, la bibliothèque `upemtk` n'est pas prévue pour ça. Et pensez à laisser la possibilité de dé-sélectionner une tuile (en re-cliquant dessus, par exemple).
- Vous avez fortement intérêt à séparer votre programme en 3 parties principales :
  - celle qui contrôle le jeu (la boucle principale du jeu, la gestion de la souris, ...) ;
  - celle qui gère l'affichage (et notamment la correspondance entre votre structure de données et les éléments affichés) ;
  - celle qui maintient toutes les données du jeu (contenu des fabriques et du centre, remplissage des lignes de motif, score, ...)
- Testez votre programme de façon intensive ! Il n'y a rien de plus énervant qu'un jeu qui plante en plein milieu de la partie...

1.2 Phase 2 : la décoration des murs du palais

À ce stade, le remplissage des lignes de motif doit fonctionner. Vous allez maintenant compléter le jeu pour :

- effectuer plusieurs manches jusqu'à ce que la condition de fin de partie soit réalisée ;
- ajouter la phase de décoration des murs du palais à chaque manche ;
- afficher le calcul des points à chaque fin de manche et le score courant ;
- ajouter la phase de préparation de la manche suivante (en gérant notamment le cas où le sac est vide) ;
- rajouter le calcul des points supplémentaires en fin de partie et afficher le gagnant ;
- pouvoir jouer jusqu'à 4 joueurs (avec un écran de sélection du nombre de joueurs) ;
- sauvegarder une partie en cours, fermer le programme, et pouvoir la relancer ensuite ;
- ajouter de quoi pouvoir choisir une configuration différente pour les murs du palais (entièrement ou partiellement colorée) en la lisant dans un fichier.

1.3 Phase 3 : IA et extensions

À ce stade, vous devez avoir un jeu complet et fonctionnel. Si c'est le cas, vous pouvez choisir d'implémenter un ou plusieurs des 3 éléments suivants :

1. Améliorer l'IA de l'ordinateur. Soyez créatifs, essayez de donner du fil à retordre à un joueur humain.
2. Rajouter la variante solo décrite ci-dessous.
3. Ré-utiliser au mieux votre code afin d'implémenter un autre jeu de la même famille : "Les vitraux de Sintra" (voir la règle fournie en pdf).

Variante solo Il s'agit d'un jeu à un seul joueur, avec un automa (attention, c'est différent du jeu à 2 joueurs dont un ordinateur, car les règles changent un peu).

- L'automa...
  - ... commence toujours ;
  - ... choisit une fabrique au hasard (ou le centre de la table si celle choisie est vide) ;
    - ... en partant du bas, cherche si une ligne de motif peut-être complétée (même si ça dépasse la ligne, mais en respectant les contraintes du mur). Si oui, il fait ça. S'il y a plusieurs possibilités : il choisit hasard (en mode normal) ou ce qui fait le plus de points potentiels (en mode difficile).
    - ... sinon (pas de complétion possible) : il choisit une des couleurs possibles au hasard. S'il peut placer une tuile dans une ligne de motif, il le fait le plus bas possible (même si ça déborde). Sinon, les tuiles de la couleur correspondante sont placées dans le couvercle.
  - ... n'a jamais de pénalité : les tuiles qui débordent sont juste mises dans le couvercle.
- Le joueur joue comme d'habitude, sauf qu'il ne peut pas prendre au centre tant qu'il reste des tuiles dans les fabriques.
- Les points sont calculés normalement à chaque fin de manche.
- Le jeu s'arrête quand le joueur a au moins 3 lignes pleines ET 5 tuiles de la même couleur. Le calcul des point est fini à ce moment là (pas de bonus), le joueur gagne s'il a plus de points que l'automa.

2 Consignes de rendu

Vous devez rendre une version du projet à la fin de chacune des 3 phases

- La date limite pour le premier rendu est le **dimanche 7 novembre 2021 à 23h55**. Passé ce délai, la note attribuée à votre projet sera 0. Les séances de TP de la semaine suivante seront consacrées à la vérification de votre rendu et à amorcer la seconde partie. Remarque : vous ne devez pas commencer la phase 2 avant d'avoir parfaitement rempli l'objectif de la phase 1.
- La date limite pour le second rendu est le **12 décembre 2021 à 23h55**.
- La date limite pour le troisième rendu est le **9 janvier 2022 à 23h55**.
  Des mini-soutenances seront organisées quelques jours après le rendu, pendant lesquelles vous ferez une démonstration sur une machine des salles de TP et serez interrogés sur le projet (les choix techniques/algorithmiques que vous avez faits, les difficultés rencontrées, l'organisation de votre travail, ...). Les contributions de chaque participant seront évaluées, et il est donc possible que tous les participants d'un même groupe n'aient pas la même note.

Contraintes

- Ce projet est à faire en binôme (s'il y a besoin de faire une exception, contactez les enseignants). Vous devrez sélectionner votre binôme sur e-learning dans la semaine suivant la publication du sujet.
- **Vous DEVEZ utiliser** `upemtk`. L'utilisation de modules autres que les modules standards de Python est interdite ; en cas de doute, n'hésitez pas à poser la question.
- **Votre programme devra impérativement s'exécuter sans problème sur les machines de l'IUT.** Prévoyez donc bien de le tester sur ces machines en plus de la vôtre et d'adapter ce qui doit l'être dans ce cas (par exemple, les vitesses de déplacement). Il sera donc utile de permettre à l'utilisateur de préciser certaines options auxquelles vous attribuez des valeurs par défaut plutôt que de devoir modifier le code à chaque exécution.

Le format de chaque rendu est une archive à déposer sur e-learning et contenant :

1. **les sources de votre projet**, commentées de manière pertinente (quand le code s'y prête, pensez à écrire les doctests). De plus :
   — les fonctions doivent avoir une longueur raisonnable ; afin d'y parvenir, pensez à structurer votre programme en fonctions simples et dont le rôle est bien défini ;
   — plus généralement, le code doit être facile à comprendre : utilisez des noms de variables parlants, limitez les structures de contrôle (boucles, if, ...) imbriquées, et veillez à simplifier vos conditions ;
   — quand cela se justifie, regroupez les fonctions par thème dans un même module ;
   — chaque fonction et chaque module doit posséder sa *docstring* associée, dans laquelle vous expliquerez le fonctionnement de votre fonction et ce que sont les paramètres ainsi que les hypothèses faites à leur sujet.
2. un **fichier texte** `README.txt` expliquant :
   — ce qui a été implémenté ou pas,
   — l'organisation du programme,
   — les choix techniques,
   — les éventuels problèmes rencontrés.
   Vous pouvez y ajouter tous les commentaires ou remarques que vous jugez nécessaires à la compréhension de votre code.

Si le code ne s'exécute pas, la note de votre projet sera 0. Vous perdrez des points si les consignes ne sont pas respectées, et il va sans dire que si deux projets trop similaires sont rendus par 2 binômes différents, la note de ces projets sera 0.