Projet : Magic Maze

Le but du projet est de réaliser une version pour ordinateur du jeu de plateau Magic Maze. Il s'agit d'un jeu coopératif où 4 personnages (pions colorés) doivent récupérer des objets dans un labyrinthe avant de s'en échapper. Tous les joueurs peuvent contrôler n'importe lequel de ces personnages mais chaque joueur n'a que très peu d'actions possibles. L'objectif est d'arriver à faire sortir tous les personnages sans que les joueurs ne communiquent entre eux.

Important : Vous devez avoir lu les règles de jeu de base (pages 2 à 6) avant de lire la suite de ce sujet.

1 Le programme à réaliser

On veut obtenir une version numérique du jeu qui se joue à plusieurs sur un seul clavier et sans souris. Pour cela, vous allez créer une fenêtre qui affichera d'une part, le labyrinthe avec les personnages qui s'y déplacent et d'autre part, pour chaque joueur, une zone qui affiche les actions possibles et permet de sélectionner le pion à actionner avec une touche de défilement propre à chaque joueur. Chaque fois qu'un joueur veut qu'un pion fasse une action, il doit d'abord sélectionner la couleur du pion et déclencher l'action avec la touche du clavier correspondante (une touche par action et par joueur).
Sur e-learning, vous avez un exemple de programme qui produit un sélecteur simple.

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. La première correspond à une version très simplifiée du jeu. 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 : la version *Kids*

Dans un premier temps, vous allez créer la base du jeu, sans les effets "sophistiqués". Il s'agira d'une version à un joueur, avec uniquement les 4 déplacements directionnels comme actions possibles. Le labyrinthe sera créé en début de partie (pas d'action *explorer*) avec les 4 objets à récupérer et une sortie. Pour l'instant, le plateau de jeu sera codé "en dur", c'est-à-dire directement dans le programme. C'est à vous de choisir la structure de données que vous utiliserez pour cela.

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

1. On peut lancer le jeu avec un niveau de 10 x 15 cases (pas de murs pour l'instant, mais quelques cases vides), et les 4 pions de couleur au milieu.
2. Le joueur peut sélectionner une des 4 actions (directions) et un pion pour le faire bouger.
3. Après que chaque pion ait récupéré l'objet de sa couleur, la sortie devient active.
4. Si on atteint la sortie et qu'elle est active, la partie est gagnée.
5. On doit voir le temps écoulé depuis le début de la partie. Si le temps dépasse 3 minutes, et que les 4 pions ne sont pas sortis, la partie est perdue.
6. On ajoute un mode *debug* : une touche au choix permet d'activer un mode où le jeu "joue tout seul" de façon aléatoire. Si on appuie sur cette touche à nouveau, on peut recommencer à jouer normalement. On doit pouvoir passer en mode *debug* aussi souvent qu'on le souhaite.

Conseil 1 La liste ci-dessus est une bonne indication de l'ordre dans lequel procéder pour écrire votre programme (sauf la dernière, qui a intérêt à être implémentée dès que possible pour servir de test).

Conseil 2 Vous avez fortement intérêt à séparer votre programme en 3 parties principales :
- celle qui contrôle le jeu (la boucle principale du jeu, la gestion des touches, ...) ;
- celle qui gère l'affichage (et notamment la correspondance entre votre structure de données et les cases affichées) ;
- celle qui maintient toutes les données du jeu (emplacement des éléments sur la grille, position des pions, objets récupérés ...)

Conseil 3 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 : plus d'action !

À ce stade, le moteur de jeu doit fonctionner. Vous aller maintenant modifier le jeu pour :
- ajouter les actions escalator et vortex [1] ;
- pouvoir jouer à deux joueurs simultanément sur le même clavier : on répartit les actions possibles entre les joueurs (2 directions et 1 des précédentes au hasard) ;
- ajouter des murs entre certaines cases du tableau ;
- ajouter les cases sablier et la gestion du temps correspondante (retournement de sablier) ;
- ajouter un mode 3 joueurs et un sélecteur de mode en début de partie.

À la fin de cette phase de développement, on doit également pouvoir mettre la partie en pause, la sauvegarder et fermer le jeu, puis relancer le jeu avec la partie sauvegardée.

1.3 Phase 3 : exploration et extensions

Dans cette phase, vous allez commencer par rajouter l'action explorer. Cela signifie que le labyrinthe n'est plus visible dès le départ et que les joueurs vont pouvoir le créer au fur et à mesure. Attention, si vous n'anticipez pas un peu, cela pourrait nécessiter de gros changements dans le design de votre jeu... Le joueur qui explore doit pouvoir piocher (au hasard) une tuile de labyrinthe de 5 x 5 cases qui viendra se positionner avec la flèche blanche devant le pion correspondant (pion de la bonne couleur sur une case d'exploration).
Remarque : toutes les tuiles ont la flèche blanche et les éventuelles cases d'exploration au niveau de la deuxième case vers la droite sur le bord de la carte.

Les tuiles du jeu de base se trouvent dans la version "print and play" fournie avec le sujet.

Extensions Vous devez également **rajouter 2 extensions**. La première extension est celle qui rajoute des gardes (Magic Maze Maximum Security, voir livret de règles). Les gardes sont des pions comme les autres que les joueurs peuvent déplacer avec leurs actions (sauf vortex et exploration) et un pion normal n'a pas le droit de se trouver sur la même tuile qu'un garde. Commencez avec un garde, puis ajouter les gardes de renfort.

Enfin vous devez implémenter au moins une autre extension au choix parmi les suivantes (la plupart viennent de l'extension Maximum Security, voir le livret de règles pour les détails) :
- Les scénarios 2,3,4,5,6,10 et 17 du jeu de base et les murs cassés du barbare (pion jaune).
- La télékinésie de l'elfe (pion vert) : 2 fois dans la partie, l'elfe peut déplacer une tuile du labyrinthe (voir livret de règles pour les conditions).
- Les sorts de la magicienne (pion violet).
- La salle des gardes et les portes blindées, avec le système de codes.

1. Attention, pour les vortex, il faut choisir celui d'arrivée. Vous pouvez commencer par un seul vortex de chaque couleur ou un choix aléatoire. Quand ça fonctionne, ajoutez l'implémentation du choix du vortex.

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 8 novembre 2020 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 **6 décembre 2020 à 23h55**.
- La date limite pour le troisième rendu est le **10 janvier 2021 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 ; n'hésitez pas à morceler votre programme en fonctions pour y parvenir ;
   - plus généralement, le code doit être facile à comprendre : utilisez des noms de variables parlants, limitez le nombre de tests, 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.