Eulaire, le jeu avec les graphes

Azizi Marwan Coutard Amelia Crague Ilian
Morel Victor Bernard Malaurie

11 mai 2023

1 Le jeu

1.1 Présentation

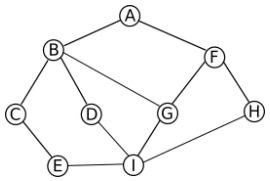
Eulaire est un jeu réalisé sous la direction d'Adrienne Lancelot. Il faut parcourir un graphe en passant exactement une unique fois par chaque arête le composant. Le but est d'enchaîner les niveaux le plus vite possible pour se faire une place dans le classement des meilleurs joueurs du monde !

1.2 Lancer le jeu

Le code source du projet se trouve sur notre dépôt gaufre. Une fois cloné, il suffit d'exécuter *make run* à la racine

2 Les graphes

Les graphes ont une place centrale dans notre projet, prétant leur nom à celui de notre jeu, et leur structure étant l'élément principal autour duquel tout le *gameplay* prend place.



2.1 Qu'est-ce qu'un graphe pour nous ?

Pour nous, un graphe est simplement un ensemble de sommets reliés par des arêtes non-orientées, qui peuvent **relier plusieurs fois deux sommets**, et ne peuvent pas relier un sommet avec lui-même. On peut le considérer plus formellement comme une paire $G = (V, f)$ avec

- V un ensemble de sommets, communément appelés *noeuds*, ou *vertex*
- $f : V^2 \to \mathbb{N}$ une application symétrique et telle que $\forall x \in V, f(x, x) = 0$.

2.2 Quelle implémentation ?

Nous avons choisi de travailler sur des graphes non-orientés, dans lesquel un sommet ne peut être relié à lui-même, cela n'apportant rien au jeu.

Durant la phase de développement du projet, nous avons utilisé deux implémentations différentes des graphes. La première est la **liste d'adjacence**, qui consiste à stocker notre graphe comme une liste de noeuds, chaque noeud contenant uniquement la liste des sommets adjacents à ce dernier, avec répétitions si les sommets étaient reliés plusieurs fois entre eux. La seconde implémentation est la **matrice d'adjacence**, une matrice composée d'entiers, l'élément à l'indice (i, j) de la matrice indiquant le nombre, potentiellement nul, de segments reliant les sommets numérotés i et j.

Chaque graphe était stocké en mémoire sous ces deux formes, l'intérêt de ce mode de fonctionnement nous parraissait être de pouvoir choisir l'implémentation la plus adaptée à chaque opération. En effet les deux implémentations ont leurs avantages : dans le cadre de notre jeu, nous sommes amenés à vérifier si il reste des arêtes dans le graphe. Dans le cadre de l'implémentation sous forme de liste, il suffit de parcourir une fois la liste des n noeuds et de vérifier que chaque élément ne possède pas de voisin (n opérations), mais il faut parcourir un peu moins de la moitié de la matrice ($n(n-1)/2$ opérations). Cependant pour voir si deux sommets d'indices i et j sont reliés, il suffit de regarder la valeur de l'élément à l'indice (i, j) de la matrice (1 opération), alors qu'il faut parcourir tous les noeuds adjacents à un des sommets et voir si l'autre y apparait pour la liste (n opérations, avec n le nombre de sommets adjacents). Nous faisions donc chaque opération de modification du graphe sur les deux implémentations

Nous avons poursuivi de cette façon durant la grande majorité de la phase de développement, et avons fini par abandonner l'implémentation sous forme de liste, ne gardant ainsi que la **matrice d'adjacence**. En effet il nous a semblé que faire chaque opération sur les deux implémentations n'était pas rentable par rapport au fait d'exécuter chaque opération (dont les plus couteuses) uniquement sur les matrices.

2.3 Trouver un chemin eulérien

L'algorithme pour faire ceci est celui de Hierholzer. Il est très simple.

Tout d'abord, il faut vérifier que ce chemin existe, que le graphe est bien eulérien. Pour cela, il suffit de regarder le degré de chacun des sommets. S'il y a 0 ou 2 points qui en ont un nombre impair, le graphe est eulérien : un chemin existe. Sinon, il ne l'est pas.

Ensuite, il suffit de partir d'un point (n'importe lequel si aucun point n'a un nombre impair de segments, un de ceux qui en ont un nombre impair sinon), puis de suivre des segments au hasard jusqu'à ce que ce ne soit plus possible (en passant seulement une fois par chaque sommet). Ensuite, tant qu'on n'est pas encore passé par tous les segments, on choisit un sommet par lequel on est déjà passé et qui possède toujours des segments, puis on refait le même procédé : on terminera sur le même sommet, et on pourra donc insérer cette boucle dans le parcours. Une fois qu'il n'y a plus de segments restants, on a fini.

3 Le jeu

Principe de base : Le but, dans chaque niveau (i.e. graphe, avec optionnellement une image de fond), est de trouver un chemin qui passe par toutes les arêtes une et une seule fois. Pour tracer le chemin, on peut cliquer sur les sommets dans l'ordre, ou maintenir le clique et passer sur les sommets dans l'ordre.

Packs : ce sont les listes de niveaux : on peut soit jouer à tous les niveaux d'un pack, soit à tous les niveaux de tous les packs. Dès qu'on complète un niveau, on passe automatiquement au suivant. Les plus haut scores sont enregistrés (dans un simple fichier) à la fin du pack avec le nom du joueur, si le joueur en décide ainsi.

Score : il est calculé de manière assez simple. Il s'agit du temps du début du pack à sa fin, plus 6 secondes par utilisation du bouton aide. Plus le score est bas, meilleur il est.

Aide : on peut obtenir de l'aide : si on clique sur le bouton aide, il nous indique le chemin à suivre, appui par appui. Attention cependant, chaque appui cause une pénalité dans le score final. Cette fonctionalité est implémentée à l'aide de l'algorithme de Hierholzer, expliqué plus haut.

Retry : il est possible, et même souvent nécessaire, de refaire un graphe car on a fait une erreur : il suffit d'appuyer sur le bouton retry. On peut aussi simplement appuyer sur la touche R.

Tutoriel : si on clique sur le bouton jouer sans avoir sélectionné de pack, on se retrouve dans le tutoriel, qui explique comment le jeu et l'éditeur fonctionnent.

Menu : pour revenir au menu, il faut et il suffit d'appuyer sur la touche échap.

Déplacement évident : si un seul segment est relié au point actuel, il suffit d'appuyer sur espace pour le traverser.

Mode memory : pour certains graphes, dès que le premier mouvement est effectué, on ne voit plus le graphe. Il faut donc trouver le chemin de tête. Si on appuie sur le bouton d'aide, le graphe est affiché jusqu'au déplacement suivant (il faut cliquer dessus deux fois pour qu'il indique le chemin à suivre comme dans le mode normal).

4 L'éditeur de graphe

Une fois le graphe implémenté en machine, il faut l'afficher à l'écran. Mais comment faire ?
Il est très difficile de déterminer arbitrairement une représentation adaptée au jeu où les sommets sont espacés, les arêtes ne se confondent pas, etc.
Nous avons donc choisi d'incorporer au projet un éditeur de graphe, qui permettrait au créateur de niveau de définir le graphe mais aussi de choisir comment il sera affiché.

4.1 Créer un graphe

Le but de l'éditeur est de permettre de créer un graphe le plus simplement et le plus directement possible.

Placer un sommet : cliquer là où on veut le placer

Relier plusieurs sommets : cliquer successivement sur les différents sommets pour former un chemin. Le sommet sélectionné apparaît en vert.
On peut cliquer à nouveau sur ce dernier pour le déselectionner. On peut relier plusieurs fois deux sommets.

Déplacer un sommet : cliquer sur le sommet, en maintenant le clic jusqu'à arriver à la position souhaitée.

Supprimer un sommet : sélectionner le bouton "supprimer", puis cliquer sur le sommet à supprimer (cela supprime bien sûr également les arêtes qui lui étaient reliées). Pour sortir du mode de suppression, cliquer à nouveau sur le bouton "supprimer".

Supprimer une arête : sélectionner le bouton "supprimer", puis cliquer et maintenir sur un sommet, déplacer le curseur jusqu'au sommet relié dont on veut supprimer la connexion.
On peut supprimer plusieurs arêtes à la suite en maintenant le clic.

4.2 Fonctionnalités avancées

Des fonctionnalités supplémentaires ont été ajoutées à l'éditeur pour le compléter.
Le but est de faciliter la conception de niveau mais aussi la prise en main de l'éditeur.

Supprimer tout : supprimer tous les sommets et toutes les arêtes.

Graphe random : générer un graphe avec un nombre d'arêtes et de sommets aléatoire. Les sommets sont également reliés de manière aléatoire.
Le nombre d'arêtes et de sommet peut être ajusté avec leurs boutons respectifs.

Importer / Exporter : importer et exporter des graphes réalisés au préalable dans l'éditeur.
Les graphes sont ensuite sauvegardés au format *mzr* (format inventé pour l'occasion), qui contient les coordonnées de chaque sommet ainsi que la matrice d'adjacence. Le format mzr est un format texte très simple, contenant la liste des positions des sommets du graphe, puis la matrice d'adjacence.

Tutoriel : le tutoriel du jeu, un pack (voir plus bas) spécial, contient un tutoriel pour l'éditeur également.

Jeu : on peut tester comment le graphe fonctionne en jeu : pour cela, il suffit de cliquer sur jeu. On peut revenir à l'éditeur en cliquant sur edit.

5 Réglages

Des raccourcis ont été ajoutés pour faciliter l'experience des joueurs. Ils permettent de

— retourner au menu (Touche <echap> par défaut)
— réessayer le graphe actuel (Touche r par défaut)
— faire un déplacement évident (si il n'y a qu'un chemin possible) (Touche <espace> par défaut)

Modifier un réglage : Pour modifier un réglage, il faut depuis le menu cliquer sur le bouton réglage puis,

1. Cliquer avec la souris sur le réglage à modifier
2. Cliquer sur la touche du clavier à attribuer

Aucun test n'est fait pour savoir si le joueur a déjà attribué une touche, on suppose que nos utilisateurs ont un minimum de bon sens.