Clément FERREIRA

Bastien ROUVIERE

Rapport Slitherlink

Nous avons commencé à faire le projet sans tenir compte des détails du l'énoncé, c'est-à-dire en faisant directement l'interface graphique sans passer par les diverses fonctions tel que est trace(etat,segment), est_interdit(etat,segment) etc. Apres en avoir parler un prof on s'est rendu compte qu'il fallait repartir de zéro en utilisant cet fois l'aide apporter dans l'énoncé.

Donc nous avons commencer cet fois par créer les fonctions de la tâche 1 sans difficultés particulière sauf pour la fonction statut_case(indices,etat,case) car il a fallut créer une nouvelle qu'on a appelé segment_adj(case) qui donner les segment adjacents à une case donner pour ensuite pouvoir coder la fonction status_case(indices,etat,case)

Durant la tâche 2, la fonction longueur_boucle(etat,segment) nous a été compliquer à coder car nous avons eu du mal a visualiser les segments suivant ,courant et précédent et aussi lorsque la boucle n'était pas complète la fonction renvoyais None au lieu du nombre de segment parcouru.

La tâche 3 a été la plus dur à réaliser nous avons tout d'abords créer une fonction detect_clique_g() qui grâce à fltk nous donnes l'abscisse et l'ordonné du clique gauche lorsque qu'on clique sur la fenêtre

Ensuite nous avons fait une fonction coord_sommet(sommet) qui retourne

taille_marge + j * taille_case, taille_marge + i * taille_case qui nous permet d'avoir les coordonnées d'un sommet en pixel, dans la même idée nous avons fait coord_seg(segment) qui nous permet d'avoir les coordonnées du segment sur lequel on clique, cela a été assez dur à concrétiser car il a fallut créer une proximité pour que l'utilisateur n'est pas besoin d'être exactement sur le

segment au pixel près en laissant une marge de 20% à l'aide de l'indication de l'énoncé : -0.2 < dx - round(dx) < 0.2.

Dans un second temps il fallut afficher les indices ainsi que les sommets et pour fini les segments.

Pour les indices on a concrétiser une fonction affiche_indice(etat,indices) qui parcourt la liste d'indices et les affiches sur la grille de jeu de différente couleur par rapport à statut_case(indices,etat,case) c'est-à-dire que si le statut de la case est complété alors l'indice s'affiche en vert si il y a trop de segment sur la case de l'indice alors il s'affiche en rouge sinon il s'affiche en noir.

Pour après afficher les sommets on a parcouru le nombre de colonnes du plateau pour cela on a créé une variable largeur_plateau qui correspond à len(indices[0]) qui est la taille du nombre de valeur dans la première liste de indices car on le rappel indices est une liste de liste, et dans la même idée on a créé hauteur_plateau qui lui correspond à len(indices) donc le nombre de liste qu'il y a dans indices. Grace à cela nous avons afficher les sommets dans la grille.

Apres ça nous avons affiché les segments avec l'aide de la fonction coord_sommet(sommet) nous avons pas rencontré de problème par rapport à cet fonction il suffisait de faire une ligne ou de mettre une croix si le segment est interdit

Le plus dur étais de créer les segments donc de les tracer on a créé la fonction tracer_seg(etat,segment) qui quand on l'appel prend en paramètres (etat,coord_seg()) ou coord_seg() est le segment sur lequel on clique.

Avons de mettre tout ça dans le code principal il fallut faire l'écran titre avec les boutons pour choisir la grille que l'on veut jouer, on a donc créer 3 fonctions différente pour chaque boutons qui renvoyer la grille avec les indices cependant lorsque on appeler les 3 fonctions dans le code principal les 3 boutons ne s'afficher pas simultanément il fallait cliquer avec la souris pour que le 2em boutons s'affiche un second un clique pour afficher le 3em boutons car on utilisais la fonctions detec_clic_g() sauf que avec cet méthode le programme attendais un événement qui est le clic gauche.

On a compris que ce n'était donc pas la bonne méthode

On a créé une seule fonction appelé ecran_titre() avec dedans les 3 boutons c'est-à-dire des rectangles avec dedans du texte et pour détecter le clique gauche on va donner x =abscisse(donne_ev()) et y=ordonnee(donne_ev())

C'est la méthode qu'on va aussi utiliser pour coord_seg() ou cet fois on on met en paramètre x et y pour qu'on ai besoin d'attendre un événement à chaque fois qu'on utilise cet fonctions et on va instancier x et y comme on l'a expliquer plus mais cet fois dans le programme principal.

De cet manière la fonctions ecran_titre() va nous permettre d'avoir la grille sur laquelle on clique, après cela on va recréer une nouvelle fenêtre en fonction de la taille de la grille.

Pour finir on va mettre dans un while True efface_tout() puis les fonctions qui affiches les indices, les sommets et les segments. Le efface_tout() va permettre d'effacer des segments car sinon les segments reste présent graphiquement même si on les a supprimer du dictionnaire etat.

Dans ce même while True on va permettre au programme de détecter la souris avec fltk, donc on donne x=abscisse(donne_ev()) et y=ordonnee(donne_ev())

Et cela va permettre d'utiliser la fonctions coord_seg(x,y), on va faire ça pour le clique gauche pour tracer un segment si le segment n'est pas dans etat ou l'effacer si le segment est dans le dictionnaire etat pour cela il faut cliquer entre deux sommets

Et pareil pour la clique droite pour interdire un segment ou effacer cette interdiction.

Pour éviter les problèmes on ajoutera l'événement 'Quitte' qui si on clique sur la croix de la fenêtre nous sors de la boucle while et ferme la fenêtre sans avoir de bug.

Pour les conditions de victoire nous avons la fonction victoire() qui vérifie si le statut de la case est bonne ou pas c'est-à-dire si il y a autant de segments que d'indice si c'est le cas alors la fonction renvoie True et nous avons la fonction longueur_boucle(etat,segment). On sait qu'on a gagné lorsque victoire() renvoie True et que longueur_boucle(etat,segment) renvoie le nombre de segment tracé, nous avons alors créer une fonctions qui compte le nombre de segment tracé avec une boucle for, qui navigue dans etat et qui vérifie les valeurs.

Aussi comme amélioration possible, nous aurions pu si le joueur est bloqué et n'arrive pas à trouver la solution lui montrer des possibilités de segments ou alors lui montrer la grille finie.

Pour la répartition du travail, Clément à fait les fonctions de la première tâche ensuite nous avons fait la deuxième tache ensemble sur Discord pareil pour les fonctions que l'on a utiliser pour la partie graphique, le code principal a été fait par Clément.

Malheureusement nous n'avons pas fait le solveur