Mini-projet : Labyrinthe

Pour le choix des fonctionnalités supplémentaires, nous en avons rajouté trois :

1. Un mode infini, qui permet au joueur de continuer à jouer tant qu’il réussit à résoudre le labyrinthe et qui rend le labyrinthe de plus en plus dur.
2. Un chronomètre, qui oblige le joueur à agir vite sinon il revient à sa position de départ et doit recommencer le niveau.
3. La possibilité de générer un nouveau labyrinthe en appuyant sur une touche spécifique si l’utilisateur considère que le labyrinthe qui est proposé n’a pas de solutions possibles.

Explication des choix de codage :

entree\_x, entree\_y = None, None / sortie\_x, sortie\_y = None, None

* Ici, on a fait le choix d’initialiser nos variables dans la partie principale du code afin de les rendre globales et les réutiliser après qu’on leur ait attribué la valeur désirée dans une fonction. Si on ne faisait pas ça, nos variables auraient été simplement locales et on aurait dû les appeler en paramètre à chaque fois.

def creer\_labyrinthe(taille, proportion\_chemins):

* Ici, on a rajouté la proportion de chemins en paramètre afin de pouvoir rendre plus tard les niveaux de plus en plus difficiles

entree\_x, entree\_y = random.choice(coins)

* On a également rendu les entrées aléatoires afin que ce soit plus intéressant que toujours commencer d’en haut à gauche. Les sorties s’adaptent en fonction de l’entrée.

curses.napms(200)

* Tout au long du programme, on a utilisé cette méthode afin que notre affichage s’affiche pendant un délai puis disparaisse pour palier au fait qu’une autre partie du programme rende l’affichage trop rapide et donc illisible

def mode\_infini(taille, proportion\_chemins, niveau, nb\_objet):

* Dans la fonction, on a fait en sorte qu’à chaque fin de niveau, le niveau suivant soit plus dur c.-à-d que la matrice devient plus grande, le nombre d’objets à collecter augmente, et la proportion de chemins faiblit (sans passer sous 0.7)

def chronometre(stdscr, temps\_limite, debut, labyrinthe):

* Dans la fonction, on calcule temps écoulé, le temps restant et on crée l’affichage du temps restant. Cette fonction renvoie également « True » si le temps est écoulé.

def main(stdscr):

* Pour structurer le programme et le rendre plus lisible et surtout appelé toutes les fonctions dans un même endroit.

stdscr.addstr(…, f"Objets collectés : {…}")

* Utilisation des f-strings pour afficher la variable qui change à chaque fois comme les .addstr ne fonctionne pas comme les print, on a du trouver une alternative qui est les f-string.

curses.wrapper(main)

* Utilisation de cette méthode plutôt que celle proposée parce qu’elle s’occupe automatiquement de l’initialisation et de la restauration du terminal.