MAZE RUNNER

Maze Runner se présente sous la forme d'un labyrinthe, dupliqué sur deux parties de la fenêtre. Deux joueurs jouent simultanément, chacun de leur côté.

Objectif: être le premier à sortir.

REGLES DU JEU

Au lancement, les deux joueurs voient apparaître deux labyrinthes identiques sur chaque moitié de la fenêtre. Les joueurs y sont positionnés au même **emplacement aléatoire**. La partie commence. Le premier joueur qui réussit à sortir a gagné. Les sorties sont matérialisées sur le labyrinthe, et il peut y en avoir plusieurs.

LE PROJET

Nous vous proposons de réaliser ce jeu, où deux joueurs jouent simultanément sur le même clavier. Plusieurs niveaux de développement vous sont proposés.

Nous vous fournissons une fonction qui permet de charger dans un tableau un labyrinthe codé sous la forme d'un fichier texte.

NIVEAU DE BASE

- Toutes les règles précédemment citées fonctionnent. Un joueur joue à l'aide des touches curseur, l'autre avec les touches **Q, S, D, Z**.
- Jeu en **mode bloquant**, c'est à dire que le programme attend que chaque joueur joue à son tour. En attendant, le jeu est bloqué. Un **témoin** (couleur, pointeur, texte) indique alternativement quel joueur doit jouer.
- Si un joueur joue dans une direction impossible car il y a un mur, il perd son tour.
- Il y a plusieurs **niveaux de difficulté**, définis par le temps dont disposent les joueurs pour jouer leur tour (voir la nouvelle fonction bloquante `attendre_touche_duree`). Plus ce temps est court, plus la difficulté augmente.
- **Compte à rebours** optionnel: les joueurs ont un temps déterminé pour sortir.

NIVEAU INTERMÉDIAIRE

- Passage en **mode non-bloquant**, les deux joueurs jouent simultanément en temps réel.
- Ajout de **3 récompenses**, placées aléatoirement dans les labyrinthes au même endroit pour chaque joueur, que chacun doit ramasser pour que les sorties s'ouvrent.
- Possibilité d'**éditer graphiquement** un labyrinthe et de le sauvegarder dans un fichier. Au lancement du jeu, choix parmi plusieurs labyrinthes.

NIVEAU AVANCÉ

Quelques suggestions (liste non exhaustive) :

- **Animations graphiques**: les personnages donnent l'impression d'effectuer des mouvements.
- **Mémorisation des scores** des joueurs (dans un fichier): les meilleurs scores sont enregistrés d'une partie à l'autre.
- **Jeu contre l'ordinateur**: écriture d'une IA pas trop stupide. Attention: la recherche du plus court chemin est un algorithme connu. Vous pouvez décider de l'implémenter mais vous devrez en comprendre le fonctionnement global.
- Ajout de **monstres mobiles** (auquel cas le plan utilisé devra comporter plusieurs chemins valides, afin qu'on puisse les contourner).
- **Sons**?

DOCUMENTATION TECHNIQUE

FICHIER LABYRINTHE

Votre programme aura besoin d'un labyrinthe, qui lui sera fourni sous la forme d'un fichier texte respectant le format suivant:

- sur la 1ère ligne, deux entiers indiquant les dimensions du plan (63 45 dans l'exemple);
- sur les 63 lignes suivantes, 45 caractères ' ' ou '*', le premier figurant un chemin, le second un mur.

REPERTOIRES DE TRAVAIL

Dans le répertoire que nous vous fournissons, vous trouverez:

- dans `lib`, la dernière version de la `libgraphique` avec sa documentation mise à jour
- dans `projet`:
    - `maze` : un exemple de labyrinthe
    - `charge_labyrinthe.c/.h` : contiennent la fonction `charge_labyrinthe` qui lit un fichier labyrinthe (par exemple `maze`) et le charge en mémoire dans un tableau de caractères
    - `exemple.c` : un exemple de main utilisant `charge_labyrinthe`, qui charge le fichier `maze` et l'affiche
    - `makefile` : adapté aux fichiers présents

Vous pouvez évidemment modifier les librairies et fonctions que nous vous fournissons afin de les adapter à vos besoins.

VOTRE TRAVAIL

PENDANT LE PROJET

- Travail en **binôme obligatoire**. Exceptionnellement par 3, si pas d'autre possibilité et uniquement sur autorisation de votre chargé de TP.
- Certains **TP de programmation seront consacrés à vos projets**.
- Toutes les fonctions doivent être commentées, ainsi que les structures et les passages délicats du programme. Les noms des fonctions et des variables doivent être judicieusement choisis.

UNE FOIS LE PROJET TERMINE

- Date de rendu des projets: **mercredi 15 novembre à minuit**.
- Vous déposerez votre projet dans la **rubrique Travaux de l'espace Programmation sur e-campus**, sous la forme d'une **archive** au format **`NOM1_NOM2.tar.gz`** où NOM1 et NOM2 sont les noms des deux membres du binôme.
- Dans votre archive, vous joindrez un document **README** indiquant le travail accompli, les problèmes rencontrés, les bugs connus, les idées originales…etc

EVALUATION

Les soutenances auront lieu avant fin novembre.

Les **compétences** qui seront évaluées sur ce projet seront à priori :

- *Présenter oralement son travail, de façon pertinente, claire et synthétique*
- *Concevoir des types structurés simples* (il est fortement conseillé d'utiliser des structures pour votre projet)
- *Rendre votre programme esthétique, intéressant, attrayant, jouable* (soyez imaginatifs, faites-vous plaisir)
- *Incorporer des solutions techniques innovantes* (soyez imaginatifs, faites-vous plaisir)
- *Découper judicieusement son code en fonctions et sous-fonctions pertinentes*
- *Commenter son code, nommer correctement ses types et variables*

ET MAINTENANT?

Vous pouvez (devez!) commencer votre projet dès maintenant, choisir un.e binôme, entamer avec elle/lui une réflexion sur la façon dont vous allez le mettre en œuvre, comment vous répartir les rôles…

N'hésitez surtout pas à questionner vos chargés de TP ou M.Marsan si vous avez des questions.

Bon travail!