Projet Splashmem

Création d'un jeu multijoueur, 4 joueurs max. Chaque joueur est un programme dont l'objectif est de remplir des cases mémoires. Le programme/joueur qui aura rempli le plus de cases avec ses crédits gagne la partie.

Chaque “programme-joueur” a un crédit d'action de 9000 points.

Liste des actions et leurs coût associés:

Code	**Cout**	**Action**
ACTION_MOVE_L, ACTION_MOVE_R, ACTION_MOVE_U, ACTION_MOVE_D,	1	Déplace le joueur d'une case.
ACTION_DASH_L, ACTION_DASH_R, ACTION_DASH_U, ACTION_DASH_D,	10	Déplace le joueur de 8 cases dans une direction donnée.
ACTION_TELEPORT_L, ACTION_TELEPORT_R, ACTION_TELEPORT_U, ACTION_TELEPORT_D,	2	Téléporte le joueur de 8 cases dans une direction.
ACTION_SPLASH	8	Marque toutes les cases autour du joueur.
ACTION_BOMB	9	Place une bombe qui marque 9 cases et qui se déclenche après 5 tours.
ACTION_STILL	1	Pas d'action

Dès qu'un joueur arrive sur une case, celle-ci est automatiquement marquée.

La taille du plateau de jeux est de 100 cases par 100 cases. (configurable)
Chaque joueur commence à la même distance des autres
Lorsqu'un joueur dépasse un bord, il est renvoyé sur le côté opposé. (comme un pacman)

Chaque programme-joueur a des coordonnées x,y, la coordonnée (0,0) et en haut à gauche de la zone de jeu.

Codage

Le projet utilise la bibliothèque graphique SDL2, normalement pré installée sur la VM.

Installer la library SDL 2:
$ sudo apt update
$ sudo apt install libsdl2-dev

Moteur Graphique

Le “moteur graphique” et d’autres éléments du jeu sont déjà codés. Le code est disponible sur github
git clone https://github.com/chaminaud/splashmem

Pour compiler le projet taper:
`make`

L'exécutable s'appelle `splash`

`ATTENTION le jeu dans son état plante au bout d’un moment.`

Programme-joueur

Les programmes joueurs sont des bibliothèques dynamiques. Il faudra passer en paramètre les 4 bibliothèques.

Des exemples de joueur sont fournis avec le code source du jeu dans le répertoire “pl”

Il est important d’avoir 4 fichiers bibliothèques différents (ne pas lancer le programme avec plusieurs fois la même bibliothèque… effets de bord garantis)

Le programme joueur doit avoir au minimum la fonction

```
char get_action()
```

Cette fonction renvoie un des codes actions définie dans le fichier actions.h

Sujet projet

Compléter le code du jeu:

- Prendre en paramètre du programme les bibliothèques dynamiques.
- Rattacher la bibliothèque à un joueur du moteur.
- Récupérer les actions des joueurs (grâce à la bibliothèque)
- Altérer les propriétés des joueurs en fonction de l'action.
- Décompter les crédits des joueurs

Détails

Les fichiers .so (bibliothèques) sont passés en paramètre de l'application:
`./splash p1.so p2.so p3.so p4.so`

Les paramètres correspondent au chemin d'accès du fichier de la library dynamique. Ce chemin d'accès sera utile pour la fonction `dlopen`()

Le point **"."** dans `./pl/p1.so` veut dire répertoire courant:
Donc `./pl/p1.so` correspond à `/home/administrateur/tp5/pl/p1.so`

Le champ `so_handle` de la structure `t_player` sera rempli avec la référence résultant de la fonction **dlopen()**

La fonction **dlopen**() permet l'ouverture d'une bibliothèque dynamique. Une fois ouverte le code de la bibliothèque est maintenant disponible.

La fonction **dlsym**() permet de récupérer une référence sur une fonction présente dans la bibliothèque.
Dans notre cas la bibliothèque contient une fonction **`get_action()`** il faut stocker la référence à cette fonction dans le **champ** `get_action` de la structure `t_player`.

Exemple et référence:
https://dwheeler.com/program-library/Program-Library-HOWTO/x172.html
https://www.systutorials.com/docs/linux/man/3-dlopen/

Les informations des joueurs

Le champ `get_action` est un pointeur de fonction qui retourne un char et ne prend pas de paramètre.

La fonction :
```
char  get_action(void);
```

La déclaration du pointeur de fonction dans la structure t_player:
```
char (*get_action)(void);
```

sont accessible via **un tableau pointeur de structure** déclaré en dans le fichier world.h et accessible n'importe ou:

```
extern t_player *players[];
```

La taille du tableau de joueur est `MAX_PLAYERS` (égale à 4)