**SOLONG**

*Ce projet est un petit jeu 2D qui va nous permettre d'appréhender les concepts de textures et de sprites ainsi que certains éléments basiques d'un jeu.*

## I. Instruction de bases :

1. Le projet doit respecter la norme demandé par 42, il en est de même pour les parties bonus ;
2. Aucune fonction ne doit s'interrompre de façon inattendue (segmentation fault, bus error, double free, etc.) ;
3. Toutes mémoires allouées au tas doit être libérées si nécessaires ;
4. Aucun leaks sera toléré ;
5. Si le sujet le demande, il faudra soumettre un Makefile qui compilera toutes les sources vers la sortie requise avec les indicateurs -Wall, -Wextra et -Werror. De plus, le Makefile devra contenir les règles suivantes : $(NAME), all, clean, fclean et re (si les bonus sont réalisés, alors il faudra ajouter la règles bonus au Makefile) ;
6. Si votre projet vous permet d'utiliser votre libft, vous devez copier ses sources et ses Makefile associé dans un dossier libft avec son Makefile associé.

## II. Partie Obligatoire :

1. Règles Principales :

| | |
|---|---|
| Nom du Programme | so_long |
| Makefile | all , clean, fclean, re, bonus |
| Arguments | Map sous le format *.ber |
| Fonctions Externes Autorisées | - open ;<br>- close ;<br>- read ;<br>- write ;<br>- printf ;<br>- malloc ;<br>- free ;<br>- perror ;<br>- strerror ;<br>- exit ;<br>- Toutes les fonctions de la MiniLibx |
| Libft | Autorisées |
| Description | Le projet consiste à créer un petit jeu en 2D où un personnage va collecter un certains nombres d'objets de valeurs afin de pouvoir quitter l'univers du jeu. |

2. Contraintes Supplémentaire :

   a. La MiniLibX doit être utilisées, soit sous la version disponibles sur l'intra ;
   b. La gestion des fenêtre doit rester fluide ;
   c. La map possède 3 composant : des murs, des objets à collecter et des espaces vides où le joueurs pourra se déplacer ;
   d. Le but de joueur est de collecter les objets présent sur la map puis de quitter la map en réalisant le moins de mouvement possibles ;
   e. Pour chaque mouvement réalisé, le nombre de mouvement doit être affiché dans le terminal ;
   f. Le joueur doit pouvoir bouger de haut en bas et de droite à gauche ;
   g. Il faudra utiliser une vue 2D (de haut en bas ou de profil) ;
   h. Le jeu n'a pas besoin d'être en temps réel ;
   i. Le joueur ne pas se déplacer sur les murs ;
   j. Le program doit affiche une image dans un fenêtre tout en respectant les règles suivantes :
      - Les keys W, A, S et D seront utilisés afin de déplacer le joueur ;
      - ESC permet de fermer la fenêtre et de quitter le program proprement, il en de même quand on appuie sur la croix rouge de la fenêtre ;
      - L'utilisation d'images de la MiniLibX est grandement conseillées.

k. Le programme doit prendre en premier argument une map avec l'extension *.ber :

<table>
<tr><td rowspan="5">Composants</td><td>0</td><td>Cases Vides</td><td rowspan="5">Exemple :<br><br>1 1 1 1 1 1 1 1 1 1 1 1 1<br>1 0 0 1 0 0 0 0 0 0 0 C 1<br>1 0 0 0 0 1 1 1 1 1 0 0 1<br>1 P 0 0 1 1 E 0 0 0 0 0 1<br>1 1 1 1 1 1 1 1 1 1 1 1 1</td></tr>
<tr><td>1</td><td>Murs</td></tr>
<tr><td>C</td><td>Objets à collecter</td></tr>
<tr><td>E</td><td>Sortie</td></tr>
<tr><td>P</td><td>Joueurs</td></tr>
<tr><td rowspan="5">Règles Map</td><td colspan="3">La map doit être fermée (c'est-à-dire entourée de mur), si cela n'est pas le cas, alors il faudra renvoyer une erreur.</td></tr>
<tr><td colspan="3">La map doit posséder au moins une sortie, un objet à collecter et une position de départ.</td></tr>
<tr><td colspan="3">Il faudra vérifier s'il y a un chemin valide vers la sortie.</td></tr>
<tr><td colspan="3">La map doit être rectangulaire.</td></tr>
<tr><td colspan="3">Si une mauvaise configuration de n'importe quelle nature est rencontrée, le programme doit se terminer correctement et renvoyer "Error\n" suivi d'un message d'erreur explicite de votre choix.</td></tr>
<tr><td colspan="3">Il faudra être capable de parser toutes sortes de map tant que ces dernières respectent les règles citées précédemment.</td></tr>
</table>

## III. Fonctions Autorisées :

| Nom et lien | | |
|---|---|---|
| **Librairie** | **Prototype** | **Description** |
| **open :** http://manpagesfr.free.fr/man/man2/open.2.html | | |
| #include <sys/types.h><br>#include <sys/stat.h><br>#include <fcntl.h> | int open(const char *pathname, int flags); | Ouvrir ou créer éventuellement un fichier ou un périphérique |
| **close :** http://manpagesfr.free.fr/man/man2/close.2.html | | |
| #include <unistd.h> | int close(int fd); | Fermer un descripteur de fichier |
| **read :** http://manpagesfr.free.fr/man/man2/read.2.html | | |
| #include <unistd.h> | ssize_t read(int fd, void *buf, size_t count); | Lire depuis un descripteur de fichier |
| **write :** http://manpagesfr.free.fr/man/man2/write.2.html | | |
| #include <unistd.h> | ssize_t write(int fd, const void *buf, size_t count); | Écrire dans un descripteur de fichier |
| **printf :** http://manpagesfr.free.fr/man/man3/printf.3.html | | |
| #include <stdio.h> | int printf(const char *format, ...); | Formatage des sorties |
| **malloc & free :** http://manpagesfr.free.fr/man/man3/malloc.3.html | | |
| #include <stdlib.h> | void *malloc(size_t size);<br>void free(void *ptr); | Allocation et libération dynamiques de mémoire |
| **perror :** http://manpagesfr.free.fr/man/man3/perror.3.html | | |
| #include <stdio.h> | void perror(const char *s); | Afficher un message d'erreur système |
| **strerror :** http://manpagesfr.free.fr/man/man3/strerror.3.html | | |
| #include <string.h> | char *strerror(int errnum); | Obtenir le libellé d'un numéro d'erreur |
| **exit :** http://manpagesfr.free.fr/man/man3/exit.3.html | | |
| #include <stdlib.h> | void exit(int status); | Terminer normalement un processus |