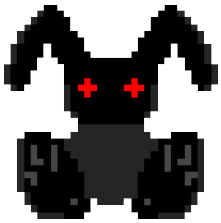


EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

Wolf3D

Jeu de tir à la première personne

- Le Laboratoire aux Lapins Noirs -
lapinsnoirs@epitech.eu

Ce projet consiste à construire un programme affichant une perspective interne d'un labyrinthe.

Nom du dépôt de rendu : wolf3d

Index

1 – Détails administratifs

Votre dépôt de rendu doit s'appeler wolf3d.

Votre programme devra être compilé avec un Makefile.

Votre binaire devra être nommé « wolf3d ».

Vous devez respecter la norme.

Votre rendu ne devra pas comporter votre programme compilé, un fichier .o ou un fichier tampon type "#*" ou "*~" sous peine d'avoir 1,5.

Votre programme doit compiler avec -W -Wall -Werror.

Les lignes de liaison avec les bibliothèques doit être les suivantes :

```
-I/home/${USER}/.froot/include
-L/home/${USER}/.froot/lib
-llapin -lsfml-audio -lsfml-graphics -lsfml-window -lsfml-system -lstdc++ -ldl -lm
```

Votre programme **devra** s'exécuter en 1024*768 ou au-delà.

La première ligne de code votre main, directement après les déclarations de variables, **doit** être un appel à la fonction bunny_set_maximum_ram. La quantité **maximale** de RAM autorisé pour ce projet est de **20Mo** (Soit 20971520 octets).
La quantité de RAM est susceptible d'être modifié durant l'évaluation.

Vous **devez** appeler la fonction bunny_set_memory_check en lui passant true, cela avant de quitter votre programme ou encore plus tôt.

La taille **totale** de votre dépôt ne **doit pas** excéder **50Mo**.

Prenez garde à régler les droits d'accès de votre dépôt et de vos dossiers et fichiers.

2 – Fonctions autorisées

Ci-dessous, la liste des fonctions systèmes autorisées pour réaliser ce projet :

open, close, read, write
alloca, setjmp, longjmp, assert
srand, rand, time
opendir, readdir, closedir

L'intégralité des fonctions de la libdl
L'intégralité des fonctions nécessaires à la mise en place d'un réseau
Ainsi que l'intégralité de la LibMath.

En cas de jeu en réseau, il n'y a aucune restriction de fonction coté serveur.

Ainsi que les fonctions de la LibLapin présente dans lapin/basic.h

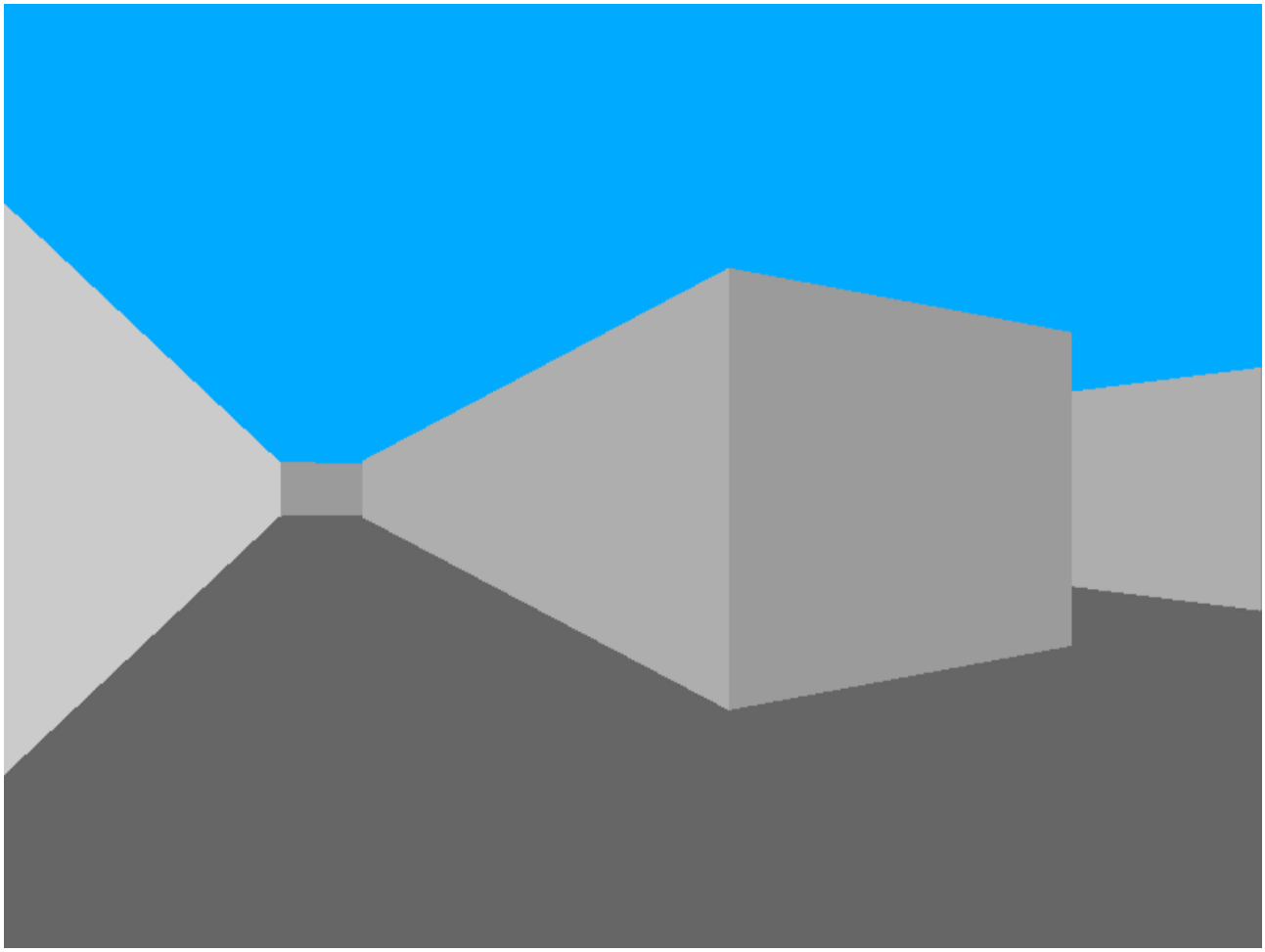
Toute fonction permettant d'allouer de la mémoire en dehors de bunny_*alloc, alloca ainsi que tous moyens de précipiter la fin du programme (exit...) sont **interdits**.

Les fonctions présentes dans lapin/advanced.h sont soumise à condition :

Toutes les fonctions **associées à un pointeur sur fonction** portant un nom type ''gl_bunny_my*'' doivent avoir été reprogrammée parfaitement. Une fonction disponible dans lapin/self_test.h vous aidera à vérifier leur fonctionnement. Les autres sont autorisées.

Si la fonction de test vous indique que votre fonction n'est pas une copie conforme, vous ne pouvez **pas** utiliser la fonction.

3 – Sujet



Ce projet consiste à créer une vue à la première personne à l'intérieur d'un labyrinthe. Vous devrez programmer cette vue en perspective ainsi que le déplacement de la caméra sur la carte. Le principe de dessin que vous devez employer est le « raycasting ».

Vous devrez pour cela pouvoir charger des fichiers carte, contenant un ou plusieurs labyrinthes, une position de départ, la position de divers éléments de jeux... Vous êtes libre d'employer n'importe quel format de carte. Nous vous encouragerons néanmoins à concevoir à plusieurs un format unique vous permettant d'échanger des cartes.

Nous devons pouvoir nous déplacer en temps réel sur la carte à l'aide de touches du clavier, éventuellement de la souris.

La couleur des murs doit varier suivant leur orientation ou présenter une texture.

Vous devez mettre en place une procédure pour couper votre programme proprement : pression sur une touche, menu... Le procédé est libre.

4 – Bonus

La liste des bonus ci-dessous n'est pas exhaustive :

On ne rentre pas dans les murs
On glisse sur les murs si on les frotte au lieu d'être complètement arrêté

La personne dispose d'une inertie lorsqu'il se déplace

Des textures sur les murs, respectant la perspective
Une texture pour le ciel avec scrolling lorsqu'on tourne
Une texture pour le sol ou pour un plafond, respectant la perspective
Des textures sont animées

Il y a des objets dans le labyrinthes
Certains objets sont animés
On ne rentre pas dans certains objets
On peut ramasser certains objets
Certains objets présentent une intelligence (monstre…)

Il y a des portes qui peuvent s'ouvrir et se fermer
L'ouverture/Fermeture est animée

Antialiasing

Lumière, en fonction de la distance avec le personnage
Lumière, en fonction de la distance avec des objets de type lumière

Les cartes présentent des murs en biais
Les cartes présentent des escaliers, des trous et des hauteurs
Les cartes présentent des plans inclinés

On peut lever et baisser la tête
On peut bouger tourner l'œil à la souris
Le personnage peut sauter (Accessible seulement si la carte présentent de la hauteur)

Vue à la troisième personne
Mode sniper (Zoom)
Vision à 360°

Éléments de jeux :Points de vie, armure, munitions, clefs, leveling
Arme de tir / Arme de corps à corps

Éléments de jeux issu d'une bibliothèque dynamique

Mini-carte
Boussole

Édition des touches via le jeu lui-même
Menu
Cinématique d'entrée
Écran de game over
Écran de transition entre les différents menus ou étape du jeu

Jeu en réseau
Jeu en écran partagé

« Campagne » (Au moins 5 niveaux)

Génération procédurale des niveaux (Procédurale, pas aléatoire)

Différents modes de jeu (Classique – Course – Capture de drapeau (Si multijoueur)

Effets sonores
Musique adaptés aux situations/niveau

Établissement d'un format de carte partagée par plusieurs équipes et/ou d'un format de modules d'extensions (.so) partagé par plusieurs équipes dont il existe une documentation complète. (Documentation et logins de ceux qui partagent ce(s) format(s) exigés)

5 – Interface de correction automatique

Afin de permettre une correction automatique de votre travail, nous vous demandons d'implémenter les fonctions suivantes :

```
void        tekpixel(t_bunny_pixelarray *pix,
                     t_bunny_position *pos,
                     t_color *color) ;

double      vecnorm(t_bunny_position *coord0,
                    t_bunny_position *coord1) ;

void        go(t_bunny_position *curpos,
               double angle,
               t_bunny_position *newpos,
               int move) ;
```

Ces fonctions peuvent tout à fait ne pas être utilisé dans votre projet si vous ne le souhaitez pas.

L'échec à la correction automatique vous fermera l'accès à l'évaluation.
Faites donc très attention en travaillant ces fonctions.

Ces fonctions doivent se trouver dans des fichiers .c situés dans le dossier ./tcore/, lui même situé à la racine de votre dépôt. L'intégralité du contenu de ce dossier et exclusivement de ce dossier sera compilé avec la moulinette.
Faites attention à ce qu'aucun main ne s'y trouve !

Si vos fichiers incluent des fichier en-têtes (.h), veillez à les placer soit à la racine de votre dépôt, soit dans un dossier ./include/, soit dans un dossier ./inc/ eux même à la racine.

tekpixel dépose dans pix un pixel de couleur color à la position indiquée dans pos.

vecnorm retourne la norme du vecteur coord0 → coord1.

go prend en paramètre la position du personnage et sa direction en radian et doit mettre dans newpos sa nouvelle position après un mouvement de longueur move. Go simule un pas en avant.

Le non-fonctionnement de tekpixel ou vecnorm est éliminatoire.