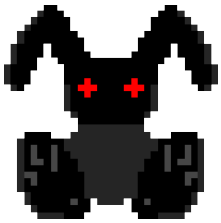


Fil de Fer 1

Projection parallèle

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

Ce projet consiste à construire un programme affichant un terrain en relief à l'écran. Le terrain en question est chargée depuis un fichier.

Nom du dépôt de rendu : fildefer1

Index

1 – Détails administratifs

Votre dépôt de rendu doit s'appeler fildefer1.

Votre programme devra être compilé avec un Makefile.

Votre binaire devra être nommé « fdf1 ».

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 **10Mo** (Soit 10485760 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 :

assert, alloca, setjmp, longjmp
open, close, read, write

La LibMath, comme sur l'ensemble des activités d'infographie
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 moyen de précipiter la fin du programme (exit...) sont **interdits**.

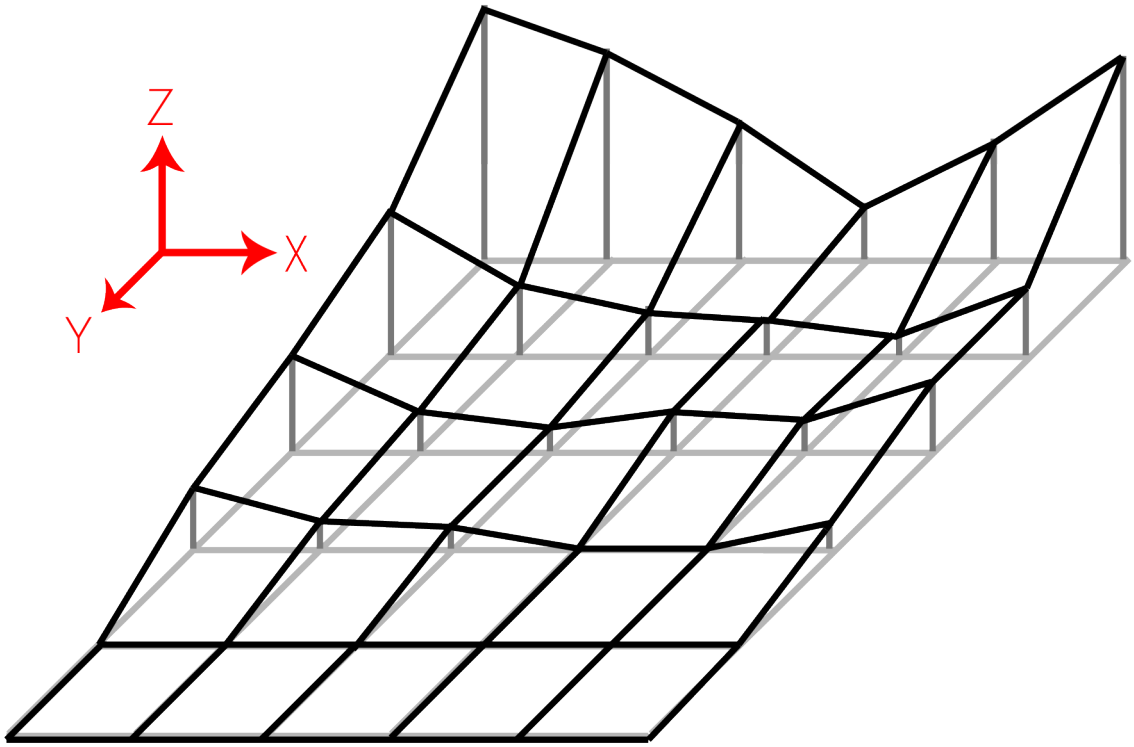
Les fonctions présentes dans lapin/advanced.h sont soumise à condition :

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.

Exceptionnellement, vous avez le droit à la fonction **bunny_set_line**, néanmoins, sachez que pour obtenir les points sur le tracé de ligne (1/3 des points), vous devez avoir codé la votre !

3 – Sujet



Le projet Fil de Fer 1 consiste à afficher à l'écran, dans une fenêtre graphique, un terrain en relief en utilisant une projection parallèle. Le terrain lui-même est contenu dans un fichier qui sera passé en paramètre au programme. Ce fichier est un fichier INI que vous pourrez charger à l'aide des fonctions de la LibLapin. Voici un exemple de format avec lequel vous devez être compatible :

```
#technocore> cat -e file.ini
[forme1]$
type=fdf1$
width=6$
height=6$
data=6,5,3,2,3,4,$
4,2,1,1,1,2,$
3,1,1,1,1,2,$
2,1,1,0,0,1,$
0,0,0,0,0,0,$
0,0,0,0,0,0$
#technocore>
```

La section « forme1 » contient des informations. Le champ type informe sur le format contenu par la section, le type « fdf1 » indique que cette section respecte le format imposé par ce sujet. (Note : Vous êtes libre d'en créer d'autres tant que vous êtes également compatible avec celui-ci.)

Concernant les sections dont le type est « fdf1 » :

Les champs width et height de cette section contiennent la largeur et la hauteur. Le champ data contient largeur x hauteur champs.

Chaque nombre du champ data correspond à un point à l'écran. La valeur du nombre correspond à l'altitude du point (Coordonnée Z) tandis que sa position vis à vis des autres nombres dans le tableau correspond aux coordonnées X et Y.

Chaque point est relié à ses voisins immédiats via une ligne.

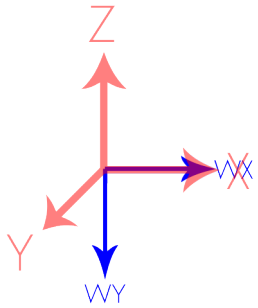
Le coin supérieur gauche dans le fichier correspond au coin supérieur gauche à l'écran.

Un champ color est susceptible d'être ajouté dans la section, faisant la même taille que le champ data. Ce champ contiendra les couleurs associées aux points. La couleur sera au format hexadecimal, préfixé du symbole « 0x ».
La composante alpha est considérée par défaut à 255, à moins d'être précisée.
Exemple non précisé : 0xFF0000, 0x00FFFF, 0xFFFFFF
Exemple précisée : 0x00FFFFFF, 0x80FF00FF, 0xFFFFFFFF

On doit pouvoir quitter le programme en appuyant sur la touche d'échappement.

3 – 1 Projection

La première étape consiste à afficher un terrain sans relief en utilisant une **projection 3D**. Une projection 3D est l'interprétation de coordonnées 3D (X, Y, Z) en coordonnée 2D (WX, WY) et donc représentable à l'écran.



Quelles sont les composantes de X, Y et Z dans le repère WX, WY ?

Tracer des lignes bêtement pour former une grille n'est pas une projection 3D.

3 – 2 Fichier

La seconde étape consiste à charger le fichier et à appliquer les informations que vous pouvez y trouver à votre terrain. Ainsi, au lieu d'un terrain sans relief, le contenu du fichier sera affiché.

Pensez à permettre aux utilisateurs de choisir entre appliquer les couleurs écrites dans le fichier ou appliquer une couleur dépendant de la hauteur.

4 – Bonus

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

Anti aliasing

Dégradé de couleur sur les lignes entre deux points de couleurs différentes.
Arrondissement des angles.

Zoom.
Rotation (Même partielle) autour de l'axe X.
Rotation (Même partielle) autour de l'axe Z.

Édition à l'exécution de l'altitude d'un point.

Translation des objets 3D.
Rotation des objets 3D.

Animation du terrain.

Remplissage des formes avec de la couleur.
Dégradé de couleur sur les parallélogrammes entre quatre points de couleurs différentes.

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) ;

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

void        tekllproject(t_bunny_position *out,
                         int              x,
                         int              y,
                         int              z) ;
```

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.

tekline trace une ligne dans pix de pos[0] à pos[1] de couleur color[0] à color[1].
Si votre fonction ne gère pas le dégradé de couleur, utilisez seulement color[0].

tekllproject écrit dans out les coordonnées projetées depuis x, y et z.

Le non-fonctionnement de tekpixel est éliminatoire.
Au moins **une** fonction entre tekline et tekllproject doit être fonctionnelle.