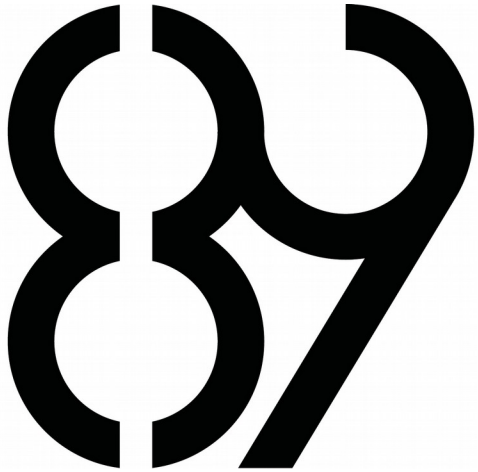

Volume

- Le Laboratoire aux Lapins Noirs -
pedagogie@ecole-89.com

*Ce mini-projet consiste à réaliser un logiciel d'affichage d'objets en 3D filaires en **perspective**.*

Ce projet est à rendre le 7 Février 2021.

INDEX

01 – Détails administratifs

Votre travail doit être envoyé via l'interface de ramassage de **l'Infosphère** :

Pour cette activité, vous rendrez votre travail sous la forme d'une archive au format tar.gz. Cette archive devra contenir l'ensemble de votre travail tel que demandé dans la section 4.

Pour créer cette archive .tar.gz, il vous suffit d'utiliser la commande suivante :

```
$> tar cvfz mon_fichier.tar.gz fichier1 fichier2 fichier3
```

Le nom « mon_fichier.tar.gz » étant à remplacé par le nom que vous souhaitez donner votre fichier archive, et « fichier1 », « fichier2 », « fichier3 » par les fichiers ou dossiers que vous souhaitez mettre dans cette archive. Vous pouvez vérifier le contenu de votre archive à l'aide de la commande « **tar -t mon_archive.tar.gz** ».

Ce travail est a effectuer seul. Vous pouvez bien sur échanger avec vos camarades, néanmoins vous devez être l'auteur de votre travail. Utiliser le code d'un autre, c'est **tricher**. Et tricher annule **toutes** les médailles que vous avez reçu sur l'activité. La vérification de la triche est réalisée de la même manière que la correction : de manière **automatique**. Prenez garde si vous pensez pouvoir passer au travers.

Médailles accessibles :

Réussir à rendre :
Vous avez réussi à envoyer votre travail au système de correction.

02 – Propreté de votre rendu

Votre rendu, c'est à dire le contenu de l'archive ou du dépôt que vous entrez sur l'interface du TechnoCentre, doit respecter **strictement** l'ensemble des règles suivantes :

- Il ne doit contenir **aucun** fichier objet. (*.o)
- Il ne doit contenir **aucun** ichier tampon. (*~, #*#)
- Il ne doit pas contenir votre production finale (programme ou bibliothèque)

La présence d'un fichier interdit mettra immédiatement fin à votre évaluation.

Médailles accessibles :

Rendu propre
Votre rendu respecte les règles de propretés imposées.

03 – Règlement quant à la rédaction du code C

Vous devez respecter l'intégralité des tables de norme pour ce projet.

04 – Construction de votre rendu

Le programme de correction va construire une sous-partie déterminée de votre rendu afin d'effectuer des tests dessus. En voici les paramètres :

- Les fichiers qui seront compilés sont ceux qui auront l'extension *.c.
- Seuls les fichiers dans le(s) dossier(s) ./ seront compilés.
- Les fichiers seront compilés avec -W -Wall -Werror.

- Tous les fichiers seront compilés **ensemble**, cela signifie donc que chaque fonction doit être unique, et que vous pouvez utiliser les fonctions des autres exercices dans chaque exercice.

L'ensemble du code que vous rendez doit pouvoir être compilé.
En cas d'échec de la compilation, vous ne serez pas évalué.

Le nom de votre fichier exécutable sera volume.

Médailles accessibles :

Construction partielle
Les éléments requis de votre projet se construisent séparément.

05 – Fonctions interdites

Vous n'avez le droit à aucune autre fonction que celle précisée dans la liste ci-dessous :

Fonctions systèmes

- open
- close
- write
- read
- ioctl
- sin
- cos

Fonctions de la LibLapin

- bunny_start
- bunny_stop
- bunny_new_pixelarray
- bunny_delete_clipable
- bunny_save_picture
- bunny_blit

- bunny_set_*_function
- bunny_set_*_response
- bunny_loop

- bunny_malloc
- bunny_free

- bunny_open_configuration
- bunny_configuration_*
- bunny_delete_configuration

L'utilisation d'une fonction interdite est assimilée à de la triche. La triche provoque l'arrêt de l'évaluation et la perte des médailles.

06 – Pré-requis

Programmez les fonctions suivantes, si vous ne les avez pas déjà faite, comme pré-requis au projet :

```
void e89_set_pixel(t_bunny_pixelarray   *px,
                   t_bunny_position     pos,
                   unsigned int         color)
```

Cette fonction dessine un pixel à la position pos dans px et de couleur color.

```
void e89_set_line(t_bunny_pixelarray    *px,
                  t_bunny_position      *pos,
                  unsigned int          *color)
```

Cette fonction trace une ligne depuis la position pos[0] jusqu'à la position pos[1] dans px. La couleur de la ligne sera soit color[0] soit un degradé allant de color[0] jusqu'à color[1].

La fonction de dessin de ligne n'est pas requis pour commencer le projet, simplement recommandée car débugger des nuages de point à l'œil nu peut-être complexe.

07 – Perspective

Programmez la fonction suivante, qui projete des coordonnées 3D en coordonnées 2D. Vous placerez le repère au centre, et plus Z est grand, plus l'objet sera loin.

```
t_bunny_position    e89_perspective(int      x,
                                    int      y,
                                    int      z);
```

Pour savoir comment projeter en perspective, inspirez vous de la **réalité**. Regardez cette photo, et utilisez la manière dont la profondeur influence la position sur la photo elle-même pour en tirer votre équation mathématique. **Graphez** vos valeurs, la fonction à employer sera évidente si vous la connaissez, sinon vous rechercherez parmis les classiques du lycée.

08 – Solides de révolution

Vous allez maintenant réaliser vos premiers solides de révolution. Dans un fichier DABSIC, peuvent également être présent l'objet suivant :

```
$> cat sphere.dab
{Objects
     [
          Type = "Sphere"
          X = 10
          Y = 10
          Z = 10
          Radius = 50
     ]
]
```

L'objet **Sphere** a quatre propriétés : une position **X, Y, Z** servant à placer le centre de la sphère et une valeur **Radius** déterminant le rayon de la sphère. Ci-dessous, l'équation paramétrique de la sphère :

$$\begin{cases} x &= r\cos\theta\ \cos\phi \\ y &= r\cos\theta\ \sin\phi \\ z &= r\sin\theta \end{cases} \qquad \left(\frac{-\pi}{2} \leq \theta \leq \frac{\pi}{2} \text{ et } -\pi \leq \phi \leq \pi\right)$$

Comment lire cette équation ? Que les points constituant la sphère sont calculés de la façon suivante :

> Pour chaque valeur de **Théta** allant de -Pi/2 à Pi/2 (donc une boucle)
> Et
> Pour chaque valeur de **Phi** allant de -Pi à Pi (donc une autre boucle, imbriquée)
>
> **X** vaut **Radius** * Cosinus (**Theta**) * Cosinus (**Phi**)
> **Y** vaut **Radius** * Cosinus (**Theta**) * Sinus (**Phi**)
> **Z** vaut **Radius** * Sinus (**Theta**)

Réalisez également les objets **Cone, Cylinder, Tore et Moebius.** Vous trouverez leurs équations sur **Wikipedia**. Réalisez ces objets **après** avoir permis de charger des objets construits dans la partie suivante.

09 – Objets construits

Les objets construits sont des objets dont les coordonnées sont données dans un fichier texte et non déduite d'une opération mathématique.

Vous trouverez un exemple de fichier complet **spaceship.dab** avec le projet sur l'Infosphère. Voici un exemple de position de triangle :

```
$> cat sphere.dab
{Mesh
 [
  Shape = "Triangle"
  Color = 0, 128, 0
  {Coord
   [
    Position = 0, -20, 0
   ],
   [
    Position = -50, -20, 13
    Color = 0, 64, 0
   ],
   [
    Position = -50, -20, -13
   ]
  }
 ]
}
```


10 – Votre jeu vidéo

Réalisez le jeu de votre choix en exploitant votre moteur 3D filaire.

Voici quelques suggestions :

Tempest
https://www.youtube.com/watch?v=AMto2HJJSSA

Night driver
https://www.youtube.com/watch?v=Wk-7BR9Dl9c

Star Wars
https://www.youtube.com/watch?v=iXOTExRQJSE&t=191s