LAPINS NOIRS

89

FRITURE

Affichage aléatoire

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

INDEX

Avant-propos :

Exercices principaux :

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'utilliser 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 en équipe, la communication entre équipe est interdite.

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. (*~, #*#)

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 les tables de la norme, sans exception.

04 – Construction partielle 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.

Médailles accessibles :

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

05 – Fonctions interdites

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

Fonctions systèmes	Fonctions de la LibLapin
- rand - srand - time	- bunny_start - bunny_stop - bunny_new_pixelarray - bunny_blit - bunny_display - bunny_usleep

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 – noise

Mettre dans le fichier : noise.c

Programmez la fonction suivante :

```
void e89_noise(t_bunny_pixelarray *picture);
```

Cette fonction, pour l'instant, remplit l'image envoyée en paramètre de pixel aléatoires.

Pour générer une valeur aléatoire, utilisez la fonction **rand**.

Dans votre main, vous pouvez utiliser **srand(time(NULL))** afin de générer des nombre aléatoires toujours différent : la fonction **srand** initialise le générateur d'aléatoire à partir du paramètre. La fonction **time** renvoi le nombre de seconde écoulée depuis le 1er janvier 1970. En conséquence, à chaque seconde passée, les nombres aléatoires générés par **rand** changeront.

07 – noise_clip

Mettre dans le fichier : noise.c

Modifiez la fonction précédente de manière à prendre en compte les attributs situés dans le t_bunny_clipable de t_bunny_pixelarray limitant la portée de certaines fonctions. Les attributs en questions sont clip_x_position, clip_y_position, clip_width et clip_height.

Ces quatre attributs indiquent le coin supérieur gauche du rectangle, la largeur et la hauteur du rectangle qui seront impacté dans l'image par la fonction e89_noise.

08 – noise_gray

Mettre dans le fichier : noise_gray.c

Programmez la fonction suivante :

```
void e89_noise_gray(t_bunny_pixelarray *picture);
```

Cette fonction fonctionne de la même manière que e89_noise à l'exception du fait qu'elle ne produit que du bruit gris. Le gris est une couleur qui a la particularité d'avoir la même valeur pour chaque composante de couleur.

09 – Flou

Mettre dans le fichier : flou.c

Maintenant, programmez la fonction suivante :

```
t_bunny_pixelarray *e89_blur(t_bunny_pixelarray *picture,
                             int                radius);
```

Cette fonction génère une nouvelle image contenant, pour chaque pixel de celle ci, la moyenne des pixels situés dans **picture** dans un rayon de **radius.**

Pour savoir si un pixel est à une distance inférieure à radius, vous pouvez calculer la distance. La formule est la même que pour la norme d'un vecteur : la position du pixel dont vous souhaitez connaître la distance étant aux coordonnées x0, y0 et le point dont vous souhaitez calculer la moyennne étant x1, y1. La norme d'un vecteur étant la racine carrée de la somme des carrés des différence entre ces points.

Le flou se calcule composante par composante de couleur et non comme étant un unique entier.