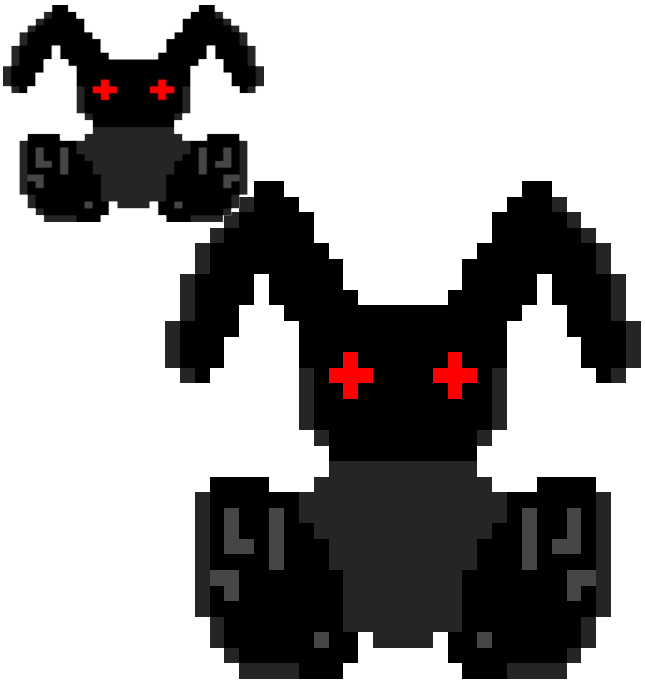

LAPINS NOIRS

Programmation graphique et audio

89

Capes et epees 2

- 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'Infosphere**.

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 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 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 – Scanlines

Écrivez la fonction suivante :

```
void e89_hscanline(t_bunny_pixelarray       *px);
```

Cette fonction dessinera dans px une ligne sur deux en noir. La première ligne sera noire.

```
void e89_vscanline(t_bunny_pixelarray       *px);
```

Cette fonction dessinera dans px une colonne sur deux en noir. La première colonne sera noire.

```
void e89_chessboard(t_bunny_pixelarray      *px);
```

Cette fonction dessinera dans px sur chaque ligne un pixel sur deux en noir. C'est à dire qu'un damier de pixel sera réalisé. Le premier pixel de la première ligne sera donc noir tandis que le premier pixel de la seconde ligne sera laissé tranquille. Le premier pixel de la troisième ligne sera noir, etc.

```
void e89_scandot(t_bunny_pixelarray         *px);
```

Cette fonction dessinera dans px un pixel noir dans chaque bloc de 2x2 pixels. Le pixel noir sera situé en bas a droite du bloc de 2x2.

06 – Scan...sounds ?

Mettez un peu de volume, puis programmez la fonction suivante :

```
void e89_clear_sound(t_bunny_effect         *fx);
```

Cette fonction placera des 0 sur l'entière surface du son. (Il n'y a donc pas de son)

```
void e89_scansound(t_bunny_effect       *fx,
                   int                  w);
```

Cette fonction ajoutera par alternance la valeur -10000 puis 10000, par tranche de **w**. Elle commencera par -15000 pendant **w** cases, puis continuera avec w 15000 avant de recommencer jusqu'à la fin de **t_bunny_effect**. Je précise à nouveau qu'il s'agit d'un **ajout**, d'une **addition** et non d'une assignation.

Pour tester votre programme, voici quelques valeurs potentielles de w : 200, 100, 50. Comme votre fonction fait une addition et non une assignation, vous pouvez appeler plusieurs fois **e89_scansound** sur un même son et les sons vont s'accumuler. Tentez de jouer avec **bunny_sound_play** après avoir rempli votre son avec 200, puis avec 100 seul… puis pourquoi pas les deux, etc.

```
void e89_limit_scansound(t_bunny_effect     *fx,
                         int                start,
                         int                stop,
                         int                 w);
```

Cette fonction a le même fonctionnement que **e89_scansound**, sauf que le remplissage n'a lieu qu'entre **start** et **stop.**

Le terme scansound n'a évidemment aucun sens. C'est une simple référence à l'exercice d'avant et à ses similaritées.

07 – Une vraie note

Votre **t_bunny_effect** dispose d'une fréquence échantillonnage, le fameux **sample_per_second**. C'est le nombre de case qu'il faut pour faire une seconde de son avec la configuration actuelle de l'effet sonore.

Les ondes sonores sont des… ondes, c'est à dire que ce sont des oscillations, des répétitions d'un même mouvement à une certaine fréquence. La fréquence est le nom de répétition par seconde. La fréquence est notée en **Hertz** (Hz).

La note LA internationale est de 220Hz, 220 oscillation par seconde, soit :

220 / 1s

Commencez par trouver comment calculer la longueur d'une seule oscillation à partir de cette information. **Il s'agit d'une unique opération mathématique.**

Ensuite, sachant que l'on dispose de **sample_per_second,** le nombre de case pour une seule seconde, cette longueur d'une seule oscillation doit pouvoir s'exprimer en cases… **Trouvez comme faire cette conversion.**

Une fois que vous avez trouvé, vous savez donc combien de case il faut remplir avec une oscillation pour faire correspondant à une note. « Oscillation » indique qu'il y a un mouvement de bascule, comme dans une sinusoïde ou **comme dans scansound**. Partagez donc en deux cette période, dans la première, mettez des valeurs dans le style de -10000 et dans l'autre 10000 et vous aurez une oscillation dite **carrée**.

Écrivez la fonction suivante :

```
void e89_set_wave(t_bunny_effect        *fx,
                  int                   start,
                  int                   stop,
                  int                   frequency);
```

N'oubliez pas de travailler avec des nombres à virgule.