

LAPINS NOIRS

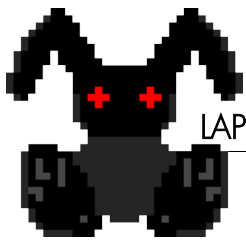
- La Caverne Aux Lapins Noirs -

Bitmap

Charger une image depuis un fichier binaire .bmp

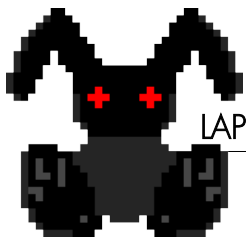
- La Caverne aux lapins Noirs -

Ce document est strictement personnel et ne doit en aucun cas être diffusé.



INDEX

- 01 – Avant-propos
- 02 – Fonctions autorisées
- 03 – Charger un fichier .bmp
- 04 – Niveaux de gris
- 05 – Noir et blanc



01 – Avant-propos

Votre travail doit être rendu via le dossier `~/tp/bitmap/` dans votre espace personnel.

Si vous faites erreur et que le dossier que vous utilisez pour votre rendu est différent, vous ne serez pas évalué faute d'avoir pu trouver votre travail.

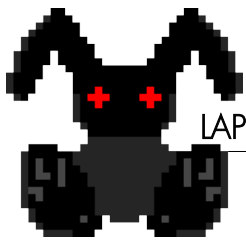
Ce travail est à effectuer seul. Vous pouvez bien sûr échanger avec vos camarades, néanmoins vous devez être l'auteur de votre travail. Utiliser le code d'un autre, c'est **tricher**. Et tricher annule **toutes** les médailles que vous avez reçu sur l'activité. La vérification de la triche est réalisée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

Votre rendu doit respecter **strictement** l'ensemble des règles suivantes :

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

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

Votre programme doit respecter les Tables de la Norme dans leur intégralité. Vous êtes invité à les observer depuis **l'Infosphère**. Elles sont disponibles comme ressource de cette activité.



02 – Fonctions autorisées

La bibliothèque logicielle venant avec le C est vaste et disponible. La LibLapin, que vous utilisez dans vos projets multimédia, est également vaste... Cependant nous avons fait le choix de vous interdire leurs utilisation intégrales, afin de vous amener progressivement à reprogrammer vous même ses fonctionnalités les plus utiles.

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.

Vous n'avez le droit d'utiliser aucune fonction issue de la LibC ou de la LibLapin à l'exception de celles que nous vous autoriserons explicitement.

Pouvoir utiliser une fonction ne signifie pas nécessairement que celle-ci soit utile à votre cas.

Pour cette activité, issu de la LibC, vous n'avez le droit qu'à la liste suivante :

- | | |
|---------|----------|
| - open | - write |
| - close | - alloca |
| - read | - atexit |
| - srand | - rand |
| - cos | - sin |
| - atan2 | - sqrt |

Remarquez bien l'absence de **malloc** et de **free**. En effet ils sont **interdits** ! Issu de la LibLapin, vous avez le droit aux fonctions suivantes :

- | | | |
|-------------------------|------------------------|------------------------------|
| - bunny_start | - bunny_set_*_function | - bunny_open_configuration |
| - bunny_stop | - bunny_set_*_response | - bunny_delete_configuration |
| - bunny_malloc | - bunny_loop | - bunny_configuration_getf |
| - bunny_free | - bunny_create_effect | - bunny_configuration_setf |
| - bunny_new_pixelarray | - bunny_compute_effect | - bunny_configuration_* |
| - bunny_delete_clipable | - bunny_sound_play | - bunny_set_memory_check |
| - bunny_blit | - bunny_sound_stop | - bunny_release |
| - bunny_display | - bunny_delete_sound | - bunny_usleep |

La suite sur la page d'après.



Pour utiliser **bunny_malloc**, vous pouvez soit programmer directement avec, soit en utilisant le modèle de projet qui vous a été transmis, mettre **1** dans la variable de Makefile **BMALLOC**, qui transformera **malloc** en **bunny_malloc**.

Vous appellerez **bunny_set_memory_check** au début de votre fonction **main** de sorte à provoquer une vérification de vos allocations à la fermeture du programme.

bunny_malloc, par défaut, limitera votre consommation de RAM à 20Mo.

Pour information :

Une image en 1920*1080 fait environ 8Mo.

Une musique en 44kHz de 1 minute en stéréo fait environ 10Mo.

Vous devrez donc disposer d'une discipline de fer avec vos allocations... et probablement trouver des compromis.

L'utilisation de **bunny_malloc** parfois **cachera** des erreurs dans votre programme, et parfois en **révélera** : son principe d'allocation était différent de **malloc**, il sera parfois plus « fort » ou plus « faible ». Ne vous mentez pas à vous en disant « l'utilisation de **bunny_malloc** fait planter mon programme », ce n'est pas **bunny_malloc**, c'est *vous*.

N'hésitez pas à l'activer, à le désactiver (Et lorsqu'il est désactivé, à utiliser **valgrind**). Et n'oubliez pas que désormais, votre demande de RAM a de véritables chances d'échouer. Car exploiter aussi peu de RAM, cela va très vite...

Dans votre rendu, il ne devra pas y avoir la moindre trace de **malloc**.



03 – Charger un fichier .bmp

Programmez la fonction suivante :

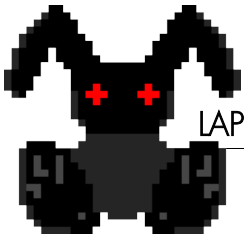
```
t_bunny_pixelarray *std_load_bitmap(const char *file);
```

Cette fonction ouvre le fichier passé en paramètre et en lit les informations. Une image est créée à partir de ces informations et ses pixels sont remplis.

Les images que vous trouverez avec ce TP sont des bitmaps 32 bits.

Vous aurez besoin de savoir comment lire un fichier bmp... Direction le web, Wikipédia, et d'autres sources. *Le format est ancien, standard, l'information est **partout**.*

Si vous réussissez cet exercice, vous aurez le droit d'utiliser les fonctions `bunny_load_pixelarray` et `bunny_load_picture` de la `LibLapin`, et serez à même de charger des fichiers png, jpg, tiff, psd, etc...



04 – Niveaux de gris

Programmez la fonction suivante :

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

Cette fonction transforme toutes les couleurs de l'image en niveau de gris en faisant la moyenne des composantes rouge, vertes et bleus et en assignant cette moyenne aux trois composantes.

La composante alpha est épargnée.



05 – Noir et blanc

Programmez la fonction suivante :

```
void std_black_and_white(t_bunny_pixelarray *px,  
                          unsigned int toggle) ;
```

Cette fonction transforme toutes les couleurs de l'image en noir ou en blanc en faisant la moyenne des composantes rouge, vertes et bleus et en testant si celle-ci est inférieure ou supérieure ou égale à **toggle**. Si la moyenne est inférieure, alors le pixel sera noir, sinon il sera blanc.

La composante alpha est épargnée.