



LAPINS NOIRS

# ÉPONGE

Strictement, il s'agit d'un Tapis de Sierpinski

- 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 – Méthode de construction
  
- 04 – Cantor
- 05 – Sierpinski



## 01 – Avant-propos

Votre travail doit être rendu via le dossier `~/colle/eponge/` 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 en binômes aléatoires. Vous pouvez communiquer avec votre binôme mais pas avec les autres équipes. Votre équipe doit être l'auteur de votre travail. Utiliser le code d'une autre équipe, 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   |
| - time  |          |

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 |
| - <b>bunny_malloc</b>   | - bunny_loop           | - bunny_configuration_getf   |
| - <b>bunny_free</b>     | - 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 – Méthode de construction

Il va vous être demandé de rendre des programmes. Il vous sera toujours demandé de fournir un **dossier** pour l'exercice le requérant. Un **Makefile** vous sera également demandé. Le **nom du programme** de sortie vous sera précisé à chaque fois. Un Makefile incorrect, un mauvais nom de programme, et votre correction n'aura pas lieu...

Des fonctions peuvent être demandées explicitement, mais dans le cadre de cette activité, cela ne signifie rien d'autre qu'une obligation de la fournir pleinement fonctionnelle : elle sera simplement testée séparément du reste, bien qu'il vous soit demandé de la fournir en même temps que le reste.

Votre compilation devra toujours comporter les options **-W**, **-Wall** et **-Werror**.

Dans le cadre de la programmation multimédia, le système de correction établira toujours la variable d'environnement **BMALLOC** à 1. Si vous utilisez le modèle de projet, cela provoquera l'utilisation de **bunny\_malloc** dans votre bibliothèque personnelle comme dans votre projet rendu.

**L'objectif de cette organisation est de permettre une correction sans pour autant vous empêcher de profiter de votre travail.**



## 04 – Cantor

Écrivez la fonction suivante :

```
void      std_cantor(t_bunny_pixelarray  *px,
                    int                    color,
                    int                    depth);
```

Cette fonction dessine un ensemble de cantor dans l'image, profitant de sa largeur et s'arrête quand **depth** est atteint ou quand la poursuite n'est plus possible. Vous n'avez le droit à aucune fonction de la Liblapin ou du système, mais vous êtes bien sur autorisé à écrire vos propres fonctions.

La nature d'une fractale est d'être **récursive**.

Le principe de cette fractale est le suivant : elle dessine une ligne entière sur toute sa zone d'effet, avant de séparer en trois sections la ligne du dessous. Elle se relance elle-même sur la première et la dernière section qu'elle a définie. Cela jusqu'à ce que sa zone d'effet soit nulle ou que **depth** ai atteint zéro.

Dans l'illustration ci-dessous, les lignes blanches horizontales sont placées à des fins d'illustration : vous n'avez pas à les dessiner.

La fonction **cantor** ne dessine que la fractale : elle ne remplit pas le fond de l'image. Le dessin est réalisé à l'aide de la couleur **color**.





## 05 – Sierpinski

Écrivez la fonction suivante :

```
void      std_sierpinski(t_bunny_pixelarray *px,  
                        int color,  
                        int depth);
```

Cette fonction dessine dans **px** un tapis de **Sierpinski**. La profondeur du dit tapis étant **depth**. Si **depth** vaut zéro, **px** sera entièrement remplie de blanc. Les trous du tapis doivent être noir. L'algorithme s'arrête si il est impossible de continuer.

Ci-dessous, l'illustration de ce qu'est un tapis de **Sierpinski**, avec une profondeur de 1, 2 et 3.

