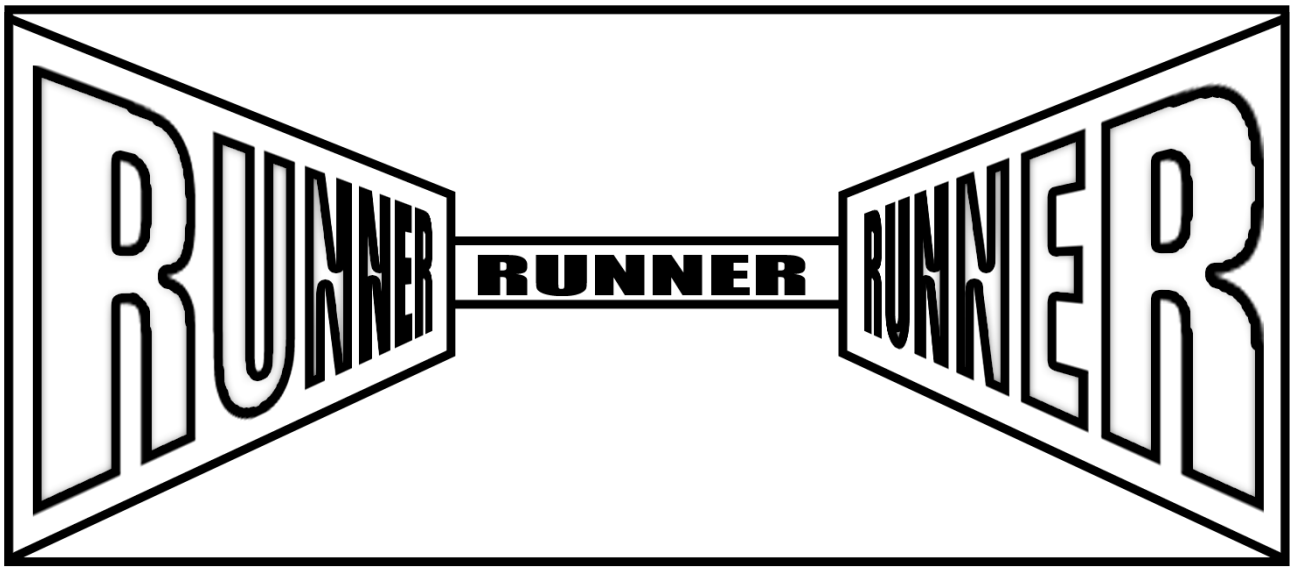
LAPINS NOIRS

89




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

*Ce projet consiste à réaliser un jeu vidéo en pseudo 3D exploitant le mécanisme connu sous le nom de « Raycasting ».*



# INDEX

## Avant-propos :

01 – Détails administratifs
02 – Propreté de votre rendu
03 – Règlement quant à la rédaction du code C
04 – Construction de votre rendu
05 – Fonctions interdites

## Projet :

06 – Pré-requis
07 – Raycasting
08 – Votre jeu



# 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 runner.

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
- sqrt

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_getf
- 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)
```

```
t_bunny_accurate_position     e89_walk_to(t_bunny_accurate_position     start,
                                          double                       angle,
                                          double                       len);
t_bunny_position              e89_position(t_bunny_accurate_position    pos);
```

Plus généralement, le contenu du TP 

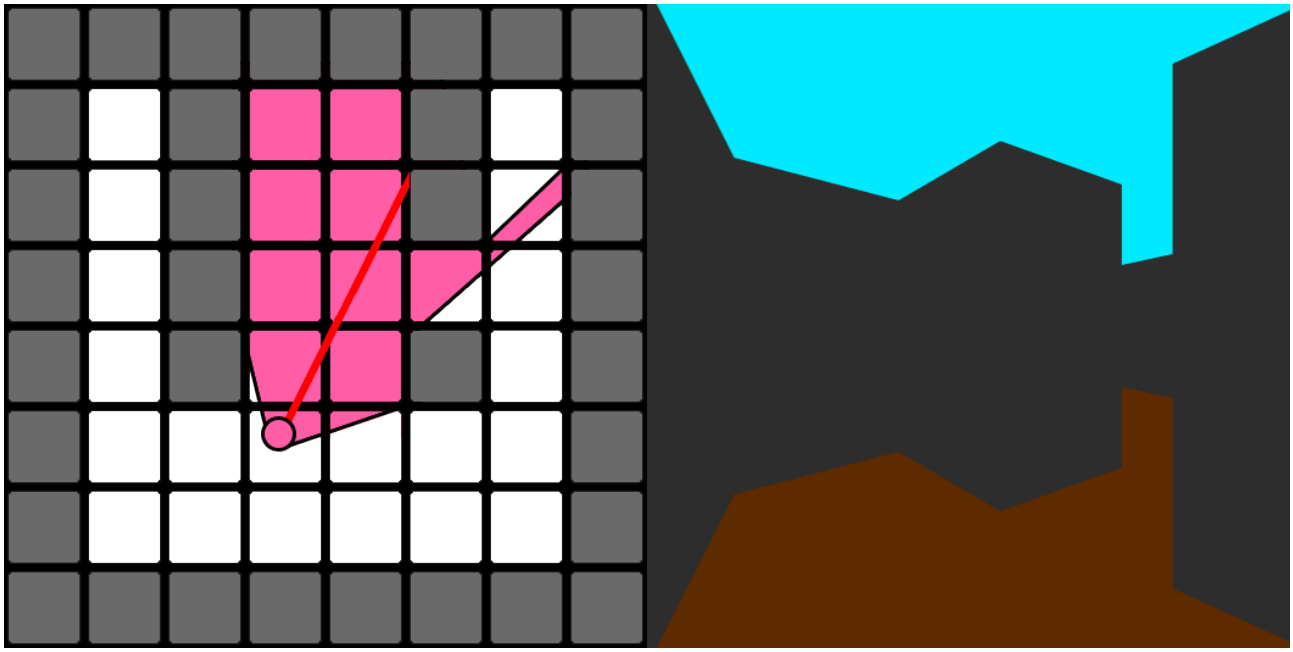
vous sera utile pour réaliser ce projet. A la condition de programmer un affichage en pseudo 3D, vous êtes ensuite libre de réaliser le jeu de votre choix.



# 07 – Raycasting

Pour parvenir à réaliser votre programme, vous devrez programmer un **raycasting** : à partir d'une position donnée (celle du joueur) et d'une direction (le regard du joueur), vous devrez modéliser ce que le joueur **voit**.

Dans le TP **Lazer**, vous apprenez comment effectuer un lancer de rayon avec une méthode simple, et ce rayon lancé vous sert à poser un pixel à l'écran. Dans ce projet, ce rayon lancé vous servira à calculer **la hauteur du mur** avec lequel le rayon est entré en collision.



Ci-dessus, une carte en vue du dessus et en vue à la première personne. Ces deux vues affichent les mêmes données d'une manière différente. Vous devez réaliser une vue à la manière de celle de droite, à la première personne.

Le champ de vision du joueur est une donnée arbitraire : à vous de choisir. Vous devez ensuite parcourir votre écran, colonne par colonne, en calculant l'angle associé à cette colonne, fonction du champ de vision et de la distance vis à vis de la colonne centrale de l'écran.

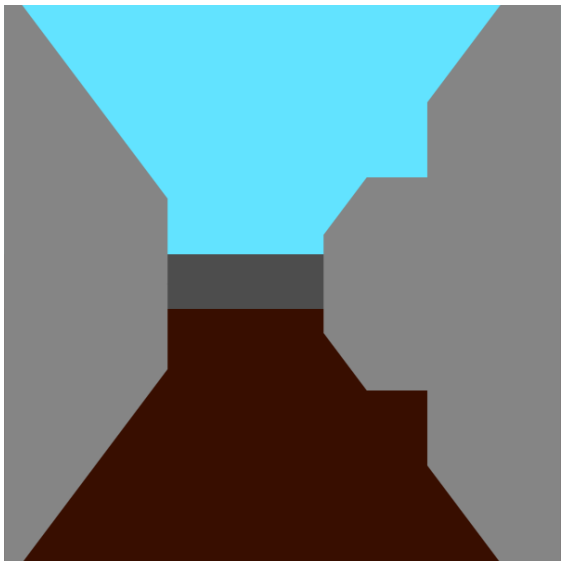
Concernant le format de carte, vous êtes libre d'exploiter celui de votre choix, même si nous vous recommandons de ne pas trop vous y attarder et d'utiliser **bunny_configuration**.



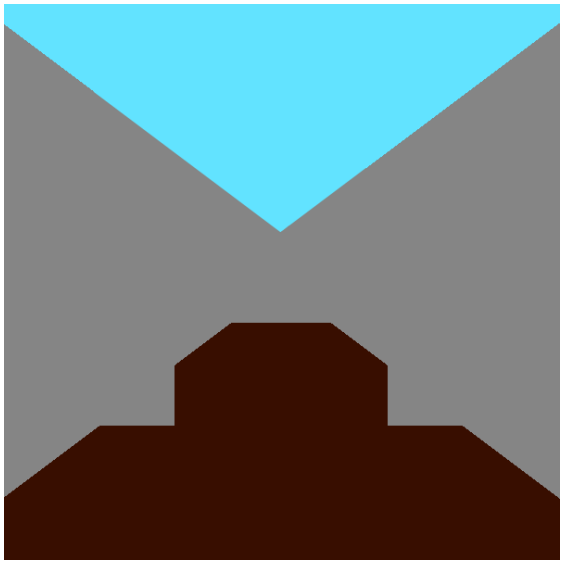
# 08 – Votre jeu

Vous êtes libre de réaliser le jeu de votre choix tant qu'il exploite une vue à la première personne. Pour commencer, vous n'êtes pas obligé d'avoir le sol en bas et le plafond en haut. Vous pouvez effectuer à la place de générer des murs, générer ce plafond et ce sol et utiliser les plans fixes comme étant des murs :

« Doomlike » / FPS



« Runner » / Platformer



Pour réaliser ce jeu, vous devrez évidemment permettre… de jouer. Pour cela vous devrez programmer la gestion du clavier, afin de permettre au joueur d'utiliser ZQSD, WASD, les touches flechées ainsi que n'importe quelle autre touche dont vous estimerez avoir besoin pour votre jeu. Vous **pourrez également programmer la gestion de manettes. Vous trouverez à l'école des manettes de jeux diverses pour vos essais.**



Au delà de la vue que vous allez programmer, le jeu l'exploitant peut lui-même être très différent en fonction de ce que vous permettez comme mouvement et les règles de « victoire ».

Ci-dessous, un jeu de tir :

Un jeu de course :