DAEMONLAB

Rogue

Votre premier jeu vidéo

- DaemonLab -
pedagogie@ecole-89.com

Cette colle consiste à réaliser un petit jeu vidéo, vraiment très petit, inspiré d'un très grand jeu, ***Rogue !***

(En vérité, il est plus proche d'un petit jeu du nom de ***DungeonUp****)*

INDEX

Avant-propos :

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 – Charger, montrer, sauvegarder

Vous allez réaliser un programme. Ce programme prendra en paramètre un unique fichier carte. Ce genre de fichier, appelons le « format carte » :

```
12,6,3
############
#T...M.....#
#.#.##.#.#.#
#.####.#.#.#
#.M......P.#
############
```

En première ligne, vous trouverez la largeur de la carte, suivi d'une virgule suivi de la hauteur de la carte. Le troisième nombre est le nombre de vie du joueur.

Le caractère '#' indique un mur. Le caractère 'P' indique la position de départ du joueur. Les caractères 'M' indiquent des monstres et le caractère 'T' indique le trésor.

Votre objectif est déjà de charger ce fichier. Ensuite, votre programme affichera un « prompt », c'est à dire une petite suite de symbole servant à indiquer au joueur qu'il peut taper quelque chose au clavier :

```
$> ./rogue map
ROGUE> show
############
#T...M.....#
#.#.##.#.#.#
#.####.#.#.#
#.M......X.#
############
```

La commande « show » lui servira à afficher la carte. Commencez par réaliser ce travail. Le 'X' est la position actuelle du joueur.

La commande « save » lui permettra de sauvegarder la partie, c'est à dire la carte, dans un fichier qui s'appellera comme la carte suivi de l'extension « .save », au format carte.

04 – Se déplacer, frapper, gagner

Vous allez implémenter une commande supplémentaire : la commande « status ». Elle affichera le nombre de point de vie du joueur, suivi d'un saut de ligne.

Vous allez maintenant ajouter quatre commandes supplémentaires : les commandes « up », « down », « left » et « right » qui serviront à déplacer le joueur sur la carte.

- Si le joueur se déplace vers un mur, rien ne se passe.
- Si il se déplace vers un monstre, il perdra un point de vie et il n'est pas déplacé.
- Si le joueur se déplace sur le trésor, il gagne la partie, le programme quitte après avoir affiché « Victory ! » suivi d'un saut de ligne.

Ensuite, la commande « hit » permettra au joueur de frapper un monstre. Si un monstre se situe à gauche, à droite, en haut ou en bas, il sera détruit. Le fait de frapper un monstre fait perdre un point de vie au joueur par monstre frappé.

Si le joueur n'a plus de point de vie, il perd la partie, le programme quitte après avoir affiché « Defeat ! » suivi d'un saut de ligne.

05 – Powerups, monstres, pièges

Vous allez maintenant ajouter un nouveau type de « tuile » : un powerup : la tuile 'H', qui fait gagner 3 points de vie au joueur.

De plus, désormais, à chaque action de déplacement du joueur, vous allez déplacer les monstres : chaque monstre fera un déplacement aléatoire en haut, à gauche, en bas, à droite, **si c'est possible** d'une unique case. Les monstres n'iront jamais sur la carte trésor, ni sur un autre monstre, ni sur un mur ou un powerup.

Vous ajouterez également une tuile 'C', la chausse-trappe : elle n'apparaît pas quand le joueur demande à afficher la carte. Quand il marche dessus, ca écrit « **Outch !** » suivi d'un saut de ligne et lui fait perdre un point de vie. La chausse-trappe disparaît après qu'on ai marché dessus. Les monstres ne marchent pas dessus.

Le joueur peut sentir les chausses trappes à l'aide de la commande « **detect** » : si une chausse-trappe est sur une case voisine, « **I feel a trap !** » est affiché suivi d'un saut de ligne. Si aucune trappe n'est a coté, « **I don't feel any trap !** » est affiché suivi d'un saut de ligne.

06 – « Vidéo »

Bonus : Concevez une interface graphique à votre jeu. Vous devez quand même proposer l'interface texte, mais en plus, proposez une interface graphique. Le choix se fait avec une option envoyé à votre programme.