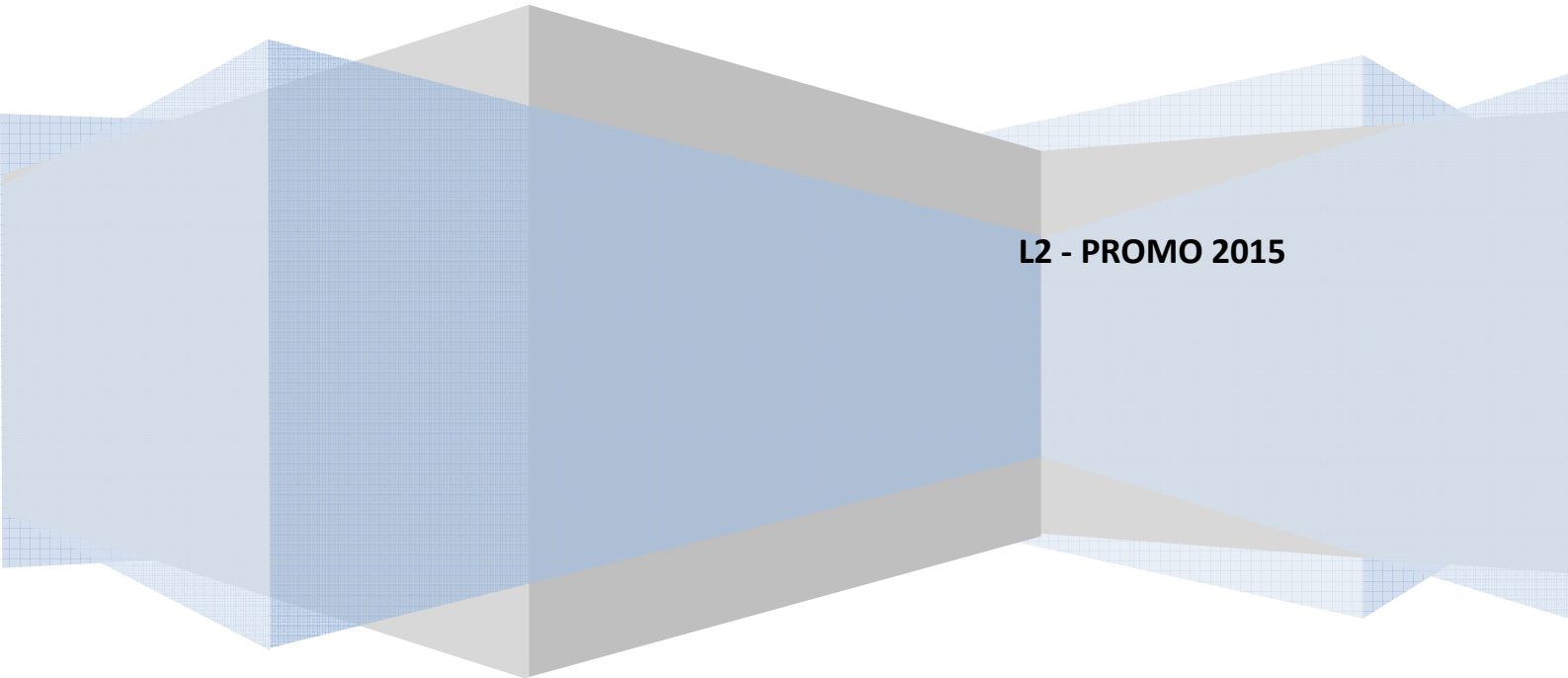


Beloeil Sébastien, Yousra Aitssi
Groupe A

Simpson world

Projet Unix



L2 - PROMO 2015

Sommaire

Introduction.....	3
IHM	4
Menu	4
Création du joueurs.....	4
Sauvegarde	5
Récupération du jeu	6
GamePlay.....	8
Armes	8
Potions.....	8
Combat	9
Passage secret	10
Enigmes	11
Fin de partie.....	11
Génération du jeu.....	12
Gestion du jeu	12
ANNEXE	15
Code Source.....	15
labyrinth.sh.....	15
joueur.sh.....	17
datagame.sh	19
initgame.sh	23
gameplay.sh.....	26
game.sh	29
Bonus	34

Introduction

Le répertoire d'exécution de labyrinth.sh est récupéré dès le début. Ainsi on peut exécuter le script depuis n'importe quel dossier sans ce soucié de créer plein de fichiers partout.

Le jeu est codé sur plusieurs fonctions, aidé de la fonction "source" qui permet d'exécuter les fonctions d'autres fichiers bash sans avoir à les rappeler, ni à s'occupé du passage des variables

Cela a l'avantage supplémentaire de sécurité l'appel des fonctions dans les fichiers. Principe de fonctionnement quasi similaire au header en C.

Le jeu affiche toujours des informations au joueur tout en haut.
Veillez à avoir une fenêtre assez grande si le resize ne marche pas. (Dans un tty il n'y a pas de problème normalement).



```
/bin/bash
/bin/bash 68x22
#####
##### Simpson's RPG Game #####
#####

note : les touches ne respect pas la casse
Ctrl + C ne marche pas => y'a un menu pour ca

#####

Chargement et Verification.

#####

Apuyer sur n'importe quel touche .

#####
|
```

Le jeu est découpé en plusieurs fichiers script bash.

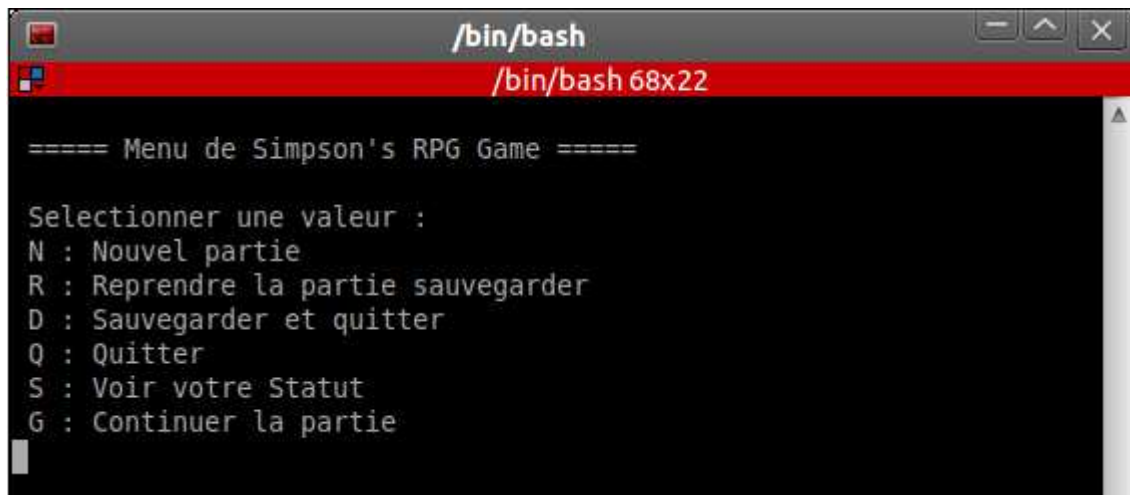
- joueur.sh : gère la création du joueur, son statut, et le chargement des données du joueur.
- game.sh : affiche l'écran principal de la partie
- labyrinth.sh : gère le lancement du jeu.
- gameplay.sh : gère toutes les actions du jeu (combat, prise de potions ...)
- datagame.sh : gère les données du jeu, notamment la suppression et les sauvegardes.
- initgame.sh : gère la création du terrain de jeux (création des dossiers et pop aléatoire)

L'ensemble des commandes affichent généralement un message en haut de l'écran. Il faut donc y faire attention !

IHM

Menu

Le jeu commence directement dans le menu, base du projet qui permet d'accomplir toutes les actions relatives aux données du jeu, gameplay exclu.



```
/bin/bash
/bin/bash 68x22

===== Menu de Simpson's RPG Game =====

Selectionner une valeur :
N : Nouvel partie
R : Reprendre la partie sauvegarder
D : Sauvegarder et quitter
Q : Quitter
S : Voir votre Statut
G : Continuer la partie
```

Création du joueurs

Tout le profil du joueur (généré à chaque tour) est ainsi stocker dans un fichier joueur respectant la même norme que les autres fichiers dans un souci d'uniformité.

Nom:description:vie:vie_restante:nom_arme:descr_arme:degat_arme:position

```
/bin/bash
/bin/bash 68x22

===== Creation du joueur =====
Nom : sebastien
Description : beloeil

Quel arme voulez-vous ?

Arme Numero : 1
Nom : bouteille de duff
Description : Se casse a la première utilisation
Dégat : 2

Arme Numero : 2
Nom : Bouteille de duff cassé
Description : Résultat des batailles de bistrot ...
Dégat : 3
```

Sauvegarde

Le joueur a le choix du nom de la sauvegarde, la date de sauvegarde est incluse avec. Jusqu'à 9 parties différentes, bien que l'on soit obligé de quitter lorsque l'on sauvegarde (imposé par le sujet)

Le jeu est entièrement détruit après sauvegarde, à la fin de chaque partie, au début de chaque nouvelle, et quand on quitte normalement, le Ctrl+C a été laissé sans supprimer afin d'avoir une espèce de sauvegarde de secours.

```
/bin/bash
/bin/bash 68x22

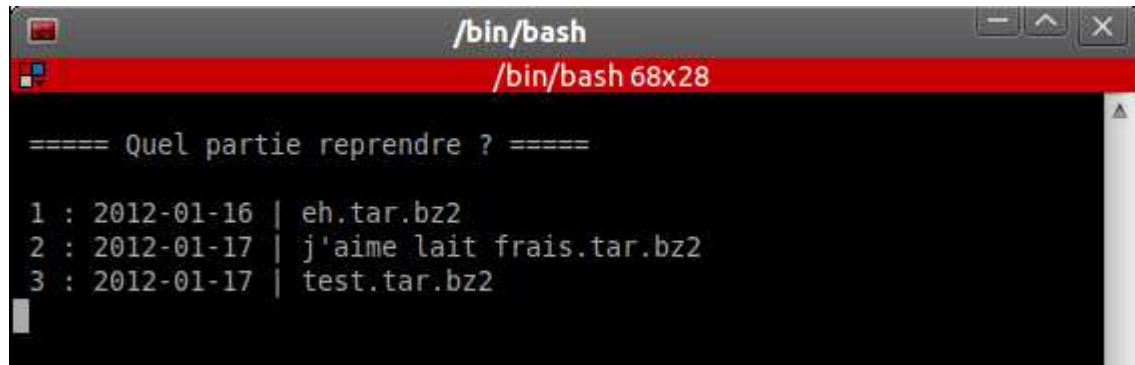
Sauvegarder et quitter ?
===== Choisissez une option =====
N : Non
Y : Oui
```

```
/bin/bash
/bin/bash 68x28

Parti Sauvegardé
ethan@01 /mnt/NAS/Dropbox/web/endrpojet/projet $
```

Récupération du jeu

La récupération des sauvegardes est aussi sympathique; Notons qu'à cause des liens symboliques certains problèmes de droit peuvent apparaître si la sauvegarde change de machine. (Notons que le joueur peut entrer des noms avec espace grâce au changement provisoire de séparateur de champ)

A screenshot of a terminal window titled "/bin/bash". The window has a red title bar and standard window controls. The terminal content shows a menu titled "==== Quel partie reprendre ? =====". Below the title, there is a list of three saved game slots, each with a date and a filename separated by a vertical bar. The first slot is "1 : 2012-01-16 | eh.tar.bz2", the second is "2 : 2012-01-17 | j'aime lait frais.tar.bz2", and the third is "3 : 2012-01-17 | test.tar.bz2". A cursor is visible at the end of the third line.

```
/bin/bash
/bin/bash 68x28

==== Quel partie reprendre ? =====
1 : 2012-01-16 | eh.tar.bz2
2 : 2012-01-17 | j'aime lait frais.tar.bz2
3 : 2012-01-17 | test.tar.bz2
```

La feuille du statut affiche toutes les informations du joueur. Avec un petit bonus pour les têtes en l'air.

```
/bin/bash
/bin/bash 68x33

===== Feuille de personnage =====

Nom : eth
Description : pala
Point de vie : 5 / 19
Armes : Pique : De chez Herman | Degats : 4
Lieu : salle2

===== Partie en cour =====
Acutellement : entree/salle2

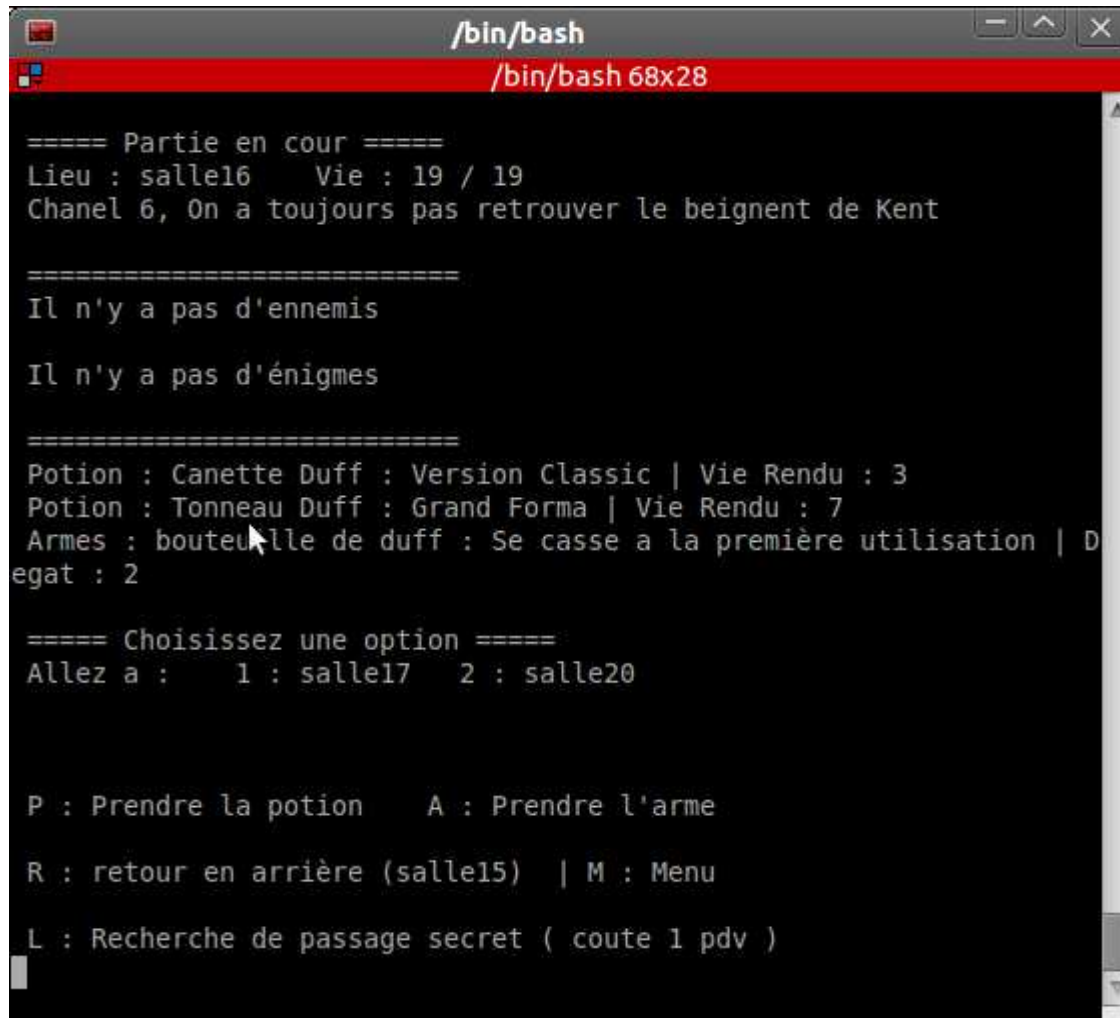
===== Carte decouverte =====
entree
entree/salle15
entree/salle15/salle16
entree/salle15/salle16/salle17
entree/salle15/salle16/salle17/salle18
entree/salle15/salle16/salle17/salle18/salle19
entree/salle15/salle23
entree/salle15/salle23/salle24
entree/salle15/salle23/salle24/salle25
entree/salle2
entree/salle2/salle10
entree/salle2/salle10/salle11
entree/salle2/salle10/salle11/salle12

===== Choisissez une option =====
G : Retour a la Partie
M : Menu
```

GamePlay

Armes

Lorsque le joueur prend une arme, il laisse tomber l'actuel, il peut donc s'il n'en est pas satisfait revenir la prendre plus tard.



```
/bin/bash
/bin/bash 68x28

==== Partie en cour ====
Lieu : salle16    Vie : 19 / 19
Chanel 6, On a toujours pas retrouver le beignent de Kent

=====
Il n'y a pas d'ennemis

Il n'y a pas d'énigmes

=====
Potion : Canette Duff : Version Classic | Vie Rendu : 3
Potion : Tonneau Duff : Grand Forma | Vie Rendu : 7
Armes : bouteille de duff : Se casse a la première utilisation | D
egat : 2

==== Choisissez une option ====
Allez a :    1 : salle17    2 : salle20

P : Prendre la potion    A : Prendre l'arme
R : retour en arrière (salle15) | M : Menu
L : Recherche de passage secret ( coute 1 pdv )
```

Potions

Le joueur ne peut pas prendre de potions si ses points de vie sont au maximum, sorte de fleur que nous lui accordons.

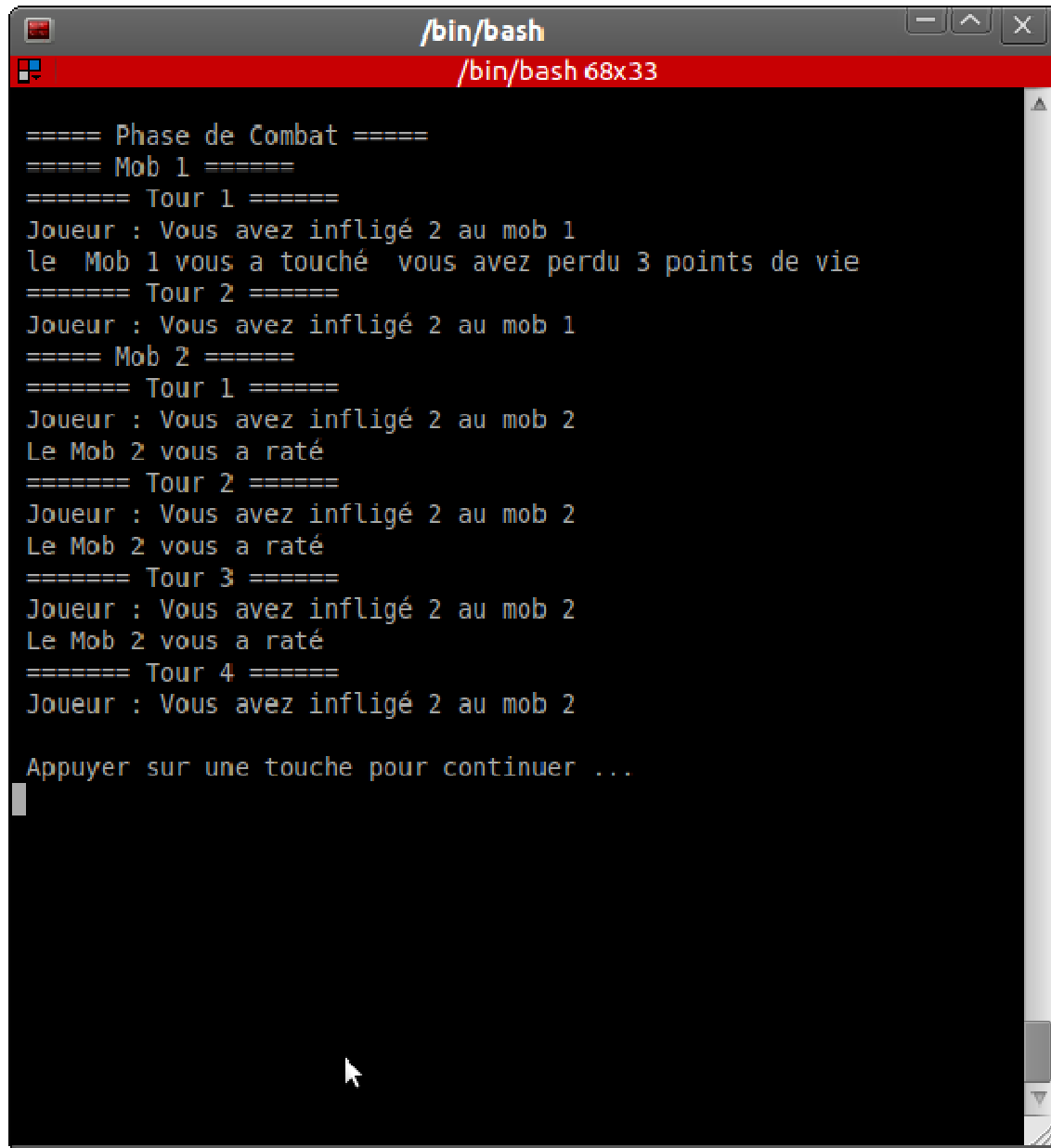


```
/bin/bash
/bin/bash 68x33

=====
Prendre la potions :
=====
Potion 1 : Canette Duff : Version Classic | Vie Rendu : 3
Potion 2 : Tonneau Duff : Grand Forma | Vie Rendu : 7
```

Combat

Le combat se déroule tour à tour à l'initiative du joueur, il affronte d'un coup tous les monstres présents.

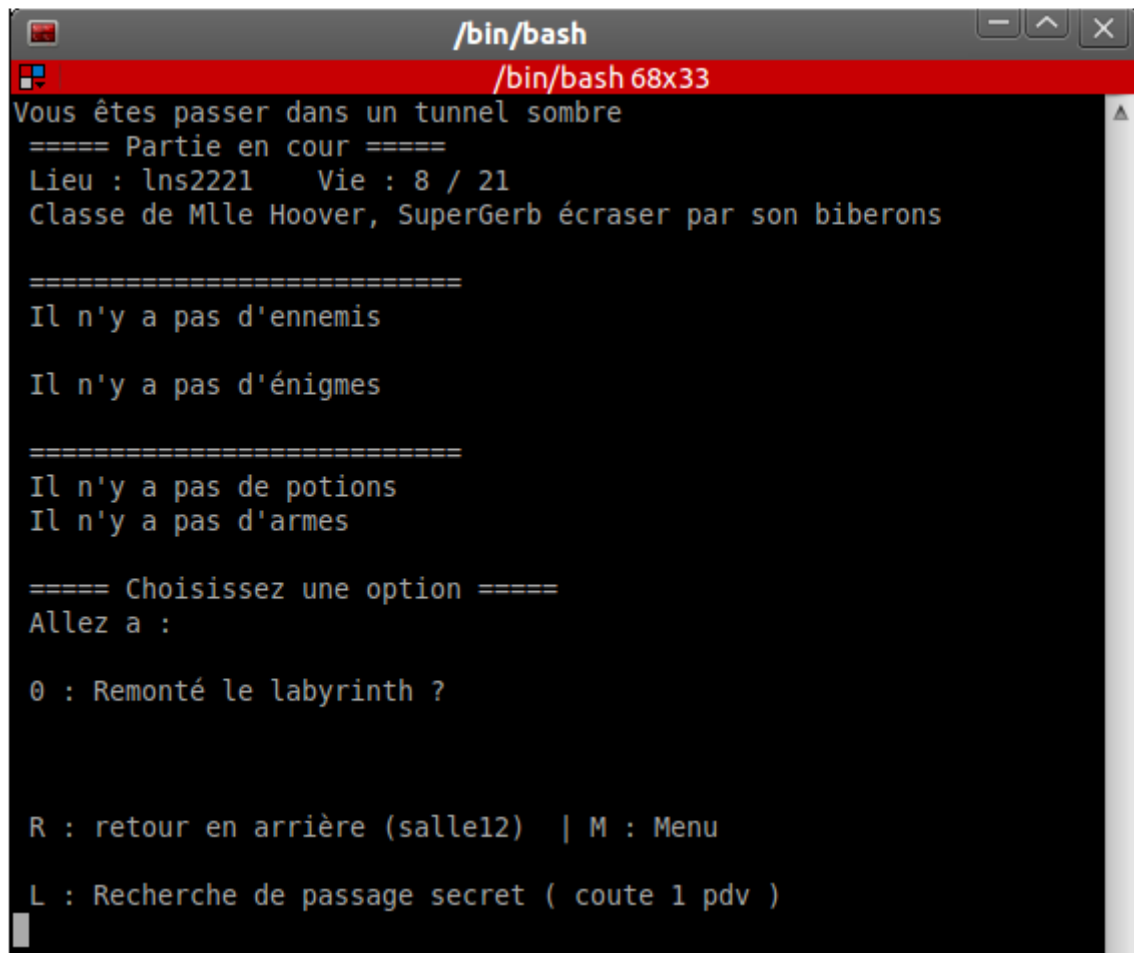
A screenshot of a terminal window titled "/bin/bash" with a red title bar. The terminal displays a combat log in French. The log starts with "==== Phase de Combat ====" followed by "==== Mob 1 =====". Then it shows "==== Tour 1 =====" and the player's action: "Joueur : Vous avez infligé 2 au mob 1". This is followed by the mob's action: "Le Mob 1 vous a touché vous avez perdu 3 points de vie". The log continues with "==== Tour 2 =====" and the player's action: "Joueur : Vous avez infligé 2 au mob 1". Then it shows "==== Mob 2 =====" and "==== Tour 1 =====". The player's action is "Joueur : Vous avez infligé 2 au mob 2", followed by the mob's action: "Le Mob 2 vous a raté". The log continues with "==== Tour 2 =====" and the player's action: "Joueur : Vous avez infligé 2 au mob 2", followed by "Le Mob 2 vous a raté". Then it shows "==== Tour 3 =====" and the player's action: "Joueur : Vous avez infligé 2 au mob 2", followed by "Le Mob 2 vous a raté". The log continues with "==== Tour 4 =====" and the player's action: "Joueur : Vous avez infligé 2 au mob 2". Finally, it shows "Appuyer sur une touche pour continuer ...".

```
/bin/bash
==== Phase de Combat ====
==== Mob 1 =====
==== Tour 1 =====
Joueur : Vous avez infligé 2 au mob 1
Le Mob 1 vous a touché vous avez perdu 3 points de vie
==== Tour 2 =====
Joueur : Vous avez infligé 2 au mob 1
==== Mob 2 =====
==== Tour 1 =====
Joueur : Vous avez infligé 2 au mob 2
Le Mob 2 vous a raté
==== Tour 2 =====
Joueur : Vous avez infligé 2 au mob 2
Le Mob 2 vous a raté
==== Tour 3 =====
Joueur : Vous avez infligé 2 au mob 2
Le Mob 2 vous a raté
==== Tour 4 =====
Joueur : Vous avez infligé 2 au mob 2
Appuyer sur une touche pour continuer ...
```

Le meurtre de tous les monstres permet d'avancer aux salles suivantes

Passage secret

Les passages secrets ont de loin été le plus problématique. Nous nous en sommes tenus aux consignes. Ainsi les passages sont en sens uniques et ne sont pas marqués sur la carte. Lorsque l'on remonte le labyrinthe par l'entrée normale de la pièce, le passage s'effondre. Sinon on revient par le passage secret.



```
/bin/bash
/bin/bash 68x33
Vous êtes passer dans un tunnel sombre
==== Partie en cour ====
Lieu : lns2221 Vie : 8 / 21
Classe de Mlle Hoover, SuperGerb écraser par son biberons

=====
Il n'y a pas d'ennemis
Il n'y a pas d'énigmes

=====
Il n'y a pas de potions
Il n'y a pas d'armes

==== Choisissez une option ====
Allez a :

0 : Remonté le labyrinth ?

R : retour en arrière (salle12) | M : Menu
L : Recherche de passage secret ( coute 1 pdv )
```

En contrepartie on révèle une partie de l'arborescence au joueur (de la salle à la racine) lorsque celui-ci remonte par l'entrée normale.

De plus la recherche de passages secrets ne coûte 1 point de vie. On peut donc mourir de désespoir ... (Le cout a été désactiver pour le rendue du projet, ne devant pas faire mourir le joueur ...)

Notons que la recherche peut indiquer un passage secret trouvé mais ne le débloquent pas, Cela indique qu'une des salles supérieures à la salle visée est bloquée par une énigme. (Note il suffirait en faite de débloquer provisoirement le passage et de le remettre a la normale après, ceci nous a apparu trop fastidieux en plus de son intérêt moyen.

Enigmes

Les énigmes sont de simples QCM à deux réponses possibles. N'ayant pas le droit de faire mourir le joueurs de cette façon, on peut les retenté à l' infini donc.



```
/bin/bash
/bin/bash 68x28

=====
Enigme : Simpsons
Comment se prenome le Vendeur de BD ?
=====
1 : Jeff | 2 : Lyonel
```

Une bonne réponse à l'énigme débloque le passage des salles suivantes. La différence avec les combats est que les énigmes changent les droits de passage sur ces salles alors que les mob empêchent juste d'avancer

Fin de partie

La fin de la partie, gagnée comme perdue, permet de recommencer une partie ou de quitter. Le labyrinthe courant est supprimé quoi qu'il arrive.



```
/bin/bash
/bin/bash 68x33

Bravo vous avez gagné

M : Retour au menu
Q : Quitter
```



```
/bin/bash
/bin/bash 90x28

===== Phase de Combat =====
===== Mob 1 =====
===== Tour 1 =====
Joueur : Vous avez raté le mob 1
le Mob 1 vous a touché

Vous avez perdu :

M : Retour au menu
Q : Quitter
```

Génération du jeu

Le jeu est généré à partir d'un plan préétabli de salle numérotée de 1 à 25 sur 6 niveaux de profondeur, entrée incluse. Les passages secrets sont générés à ce moment et sont fixés.

La sortie est de sélectionner aléatoirement sur l'une des 4 les plus profondes (6^{ème} niveau)

Les armes et les monstres sont générés sur une base aléatoire, mais contrôlés. C'est-à-dire que les mobs étant rangées dans la base de données, on pioche au fur et à mesure qu'on avance en profondeur.

Les potions sont générées sur une base aléatoire et en plusieurs exemplaires. Afin de faciliter le jeu on augmente progressivement la profondeur possible, mais toujours à partir de l'entrée. Ainsi il est plus probable d'avoir des potions au début du jeu.

Les descriptions sont distribuées aléatoirement sur toutes les salles sauf sur l'entrée. (On commence toujours dans une entrée identique.

Les énigmes sont générées aléatoirement sans distinction sur 9 des 25 salles, la liste des salles à modifier est mise en mémoire pour faire un chmod depuis le fond du labyrinthe afin d'éviter les problèmes liés au chmod.

Gestion du jeu

L'ensemble des informations du joueur sont contenu dans le fichier joueur, au début de la partie (nouvelle ou reprise) les informations sont chargées en mémoire, et à chaque action entraînant une modification, le fichier joueur est écrasé par un nouveau à jours

Le fichier carte regroupe les salles visités au fur et à mesure par le joueur.

Les autres fichiers contiennent les informations relatives à la salle (description etc. ...)

Conclusion

Les plus gros problèmes du sujet ont été d'ordres algorithmique, surtout avec la gestion des passages secrets suite aux différents `chmod`. Nous avons très tôt cherché à organiser correctement le code, facilitant ainsi, grâce à « source » l'appel de fonction, la gestion des variables, etc., et la correction des différents bugs rencontrés a posteriori « A chaque fonction ... sa fonction ».

Le point le plus décevant, malgré une liberté, somme toute, très large, est l'obligation de faire des `chmod`, qui complique inutilement le code et la gestion du jeu.

D'un côté tant pédagogique que technique ce projet ne nous a rien apporté. D'un côté nous avons un sujet original, bien que pauvre techniquement qui ne nous a rien appris de plus que ce que nous savions déjà. De l'autre côté il nous a laissé un peu perplexe car le langage `bash` a été créé dans le but d'automatiser les tâches et non pas de créer ce genre de programme.

Certain point aurait pu ainsi être amélioré mais l'intérêt aurait été succinct. Comme par exemple une fonction de cheat code, la gestion des passages secrets etc.

ANNEXE

Code Source

```
labyrinth.sh
#!/bin/bash

### Fichier principal
# menu
# quit
# commande de lancement avec affichage top kikoo

##### TODO EXISTENTIEL #####
# Que faire du trap ctrl + c ?
# proposer la fonction quit ?
# tout del sans confirmation ?
# insulter l'utilisateur ?

printf "\e[8;45;67;t" # recadre la fenetre , surtout en longueur pour
prévoir large, non obligatoire, ne fonctionne pas sur un tty (innutile en
théorie)

pos=$(echo "$PWD/${dirname $0}" | sed s#'\./'##g) # on stock la position
dès le debut
# le sed remplace le /. si le fichier est executer depuis le dossier
courant pour eviter certain bug dans la carte
cd $pos # et on y va histoire de...

# pseudo pseudo-equivalent au header en C
source $pos/joueur.sh # relatif au joueurs
source $pos/datagame.sh # relatif au donnée du jeu
source $pos/game.sh # relatif au jeu lui même
source $pos/gameplay.sh # relatif au deroulement du jeu
source $pos/initgame.sh # relatif a la génération du jeu

# Display le menu
function menu {
clear
echo $@
while true ;do
echo " ===== Menu de Simpson's RPG Game ===== "
echo ""
echo " Selectionner une valeur : "
echo " N : Nouvel partie"
echo " R : Reprendre la partie sauvegarder" #TODO
echo " D : Sauvegarder et quitter" #TODO
echo " Q : Quitter"
echo " S : Voir votre Statut"
echo " G : Continuer la partie"

read -n 1 choice

case "$choice" in
[nN] ) newgame ;;
[rR] ) recupgame ;;
[dD] ) savegame ;;
[qQ] ) quit ;;
[sS] ) statut ;;
[gG] ) game ;;
```

```

        * ) clear; echo " Mauvaise saisie , recommencez" ;;
    esac
done # aucun break car on veut pas sortir du menu
}

#quitter le jeu sans sauvegarder # tout detruire
function quit {
clear
echo $@
while true ;do
    echo " Tout arrêter, sans sauvegarder ?"
    echo " ===== Choisissez une option ===== "
    echo " N : Non"
    echo " Y : Oui"
    read -n 1 choice

    case "$choice" in
        [nN] ) menu ;;
        [yY] ) del;clear;exit;;
        * ) clear; echo " Mauvaise saisie , recommencez" ;;
    esac
done
}

trap 'clear; echo "Pas bien de quitter comme ca ! "; exit ' SIGINT SIGTERM

clear
echo " ##### "
echo " ##### Simpson's RPG Game ##### "
echo " ##### "
echo ""
echo " note : les touches ne respect pas la casse "
echo " Ctrl + C ne marche pas => y'a un menu pour ca"
echo ""
echo " ##### "
echo ""

### c'est juste pour faire genre on est d'accord ...
echo -n " Chargement et Verification "
    for j in $(seq 1 15); do
        echo -n ".";sleep 0.03
    done
echo ""
echo ""
echo " ##### "
echo ""

echo -n " Apuyer sur n'importe quel touche "

    for i in $(seq 1 5); do
        echo -n ".";sleep 0.075
    done
echo ""
echo ""
echo " ##### "

read -n 1

```

```

if [[ -e $pos/joueur ]]; then
    loadplayer ## on est pas censé quitter sans del ou sav , de cette
façon le ctrl + c permet de reprendre sa partie en cours ...
    menu " Partie charger automatiquement. Attention la prochaine fois que
vous quittez n'importe comment"
fi
menu

```

joueur.sh

```

#!/bin/bash

### Gestion général du joueur
# player
#   # créé le joueur
# statut
#   # affiche le statut et la carte decouverte du joueur
# genplayer
#   # Génère le fichier joueur
# gencarte
#   # Génère la carte du joueur

# Génération de la fiche de présentation du joueur
function player {
clear
echo $1
echo " ===== Creation du joueur ===== "
echo ""
echo " Votre nom explorateur ? "
echo -n " "
read nom
nom=$(echo "$nom" | sed 's/\:/\ /g') # on veut pas de ":"
echo " Décrivez vous"
echo -n " "
read descr
descr=$(echo "$descr" | sed 's/\:/\ /g') # idem
clear; echo ""
while true ;do

    echo " ===== Creation du joueur ===== "
    echo " Nom : $nom"
    echo " Description : $descr"
    echo ""
    echo " Quel arme voulez-vous ?"
    echo ""

    for i in `seq 1 2`;do ### par défaut on a le choix qu'entre les 2
première
        echo " Arme Numero : $i"
        echo " Nom : $(sed -n $i\p $pos/bdd/armes | cut -d: -f1)"
        echo " Description : $(sed -n $i\p $pos/bdd/armes | cut -d: -f2)"
        echo " Dégat : $(sed -n $i\p $pos/bdd/armes | cut -d: -f3)"
        echo ""
    done

    read -n 1 choice

    case "$choice" in
[1-2] ) arm=$(sed -n $choice\p $pos/bdd/armes); break ;;
* )clear; echo "Mauvaise saisie ";;
    esac
done
}

```

```

    esac
done

# si il choisit la premiere, il peut tjrs choisir la 2eme dans l'entree du
coup ...
if [ $choix -eq 1 ];then
    echo "$(sed -n 2p $pos/bdd/armes | cut -d: -f1):$(sed -n 2p
$pos/bdd/armes | cut -d: -f2):$(sed -n 2p $pos/bdd/armes | cut -d: -f3)" >
$pos/entree/armes
else
    echo "$(sed -n 1p $pos/bdd/armes | cut -d: -f1):$(sed -n 1p
$pos/bdd/armes | cut -d: -f2):$(sed -n 1p $pos/bdd/armes | cut -d: -f3)" >
$pos/entree/armes
fi

cd $pos/entree # on entre dans le labyrinth
# on stock dans des variable pour ne pas avoir a lire/modifier inutilement
le fichier, on le regenere au moment de sauvegarder
vie=$((RANDOM%6+16))
vies=$vie
lieu=$(basename $PWD)

#on la genere une fois au debut (histoire de dire que le jeu a commencer)
genplayer
gencarte
statut
}
# Affiche le statut du joueur
function statut {

if [[ ! -e $pos/joueur ]]; then menu " Aucun jeu en cour"
fi

clear
echo $@
while true ;do
    echo " ===== Feuille de personnage ====="
    echo ""
    echo " Nom : $nom"
    echo " Description : $descr"
    echo " Point de vie : $vies / $vie"
    echo " Armes : $(echo "$arm" | cut -d: -f1 ) : $(echo "$arm" | cut -d:
-f2 ) | Degats : $(echo "$arm" | cut -d: -f3 )"
    echo " Lieu : $(basename $lieu)"

    echo ""
    echo " ===== Partie en cour ====="
    echo " Actuellement : $lieu "
    echo ""
    echo " ===== Carte decouverte ====="
    cat $pos/carte

    echo ""
    echo ""
    echo " ===== Choisissez une option ====="
    echo " G : Retour a la Partie"
    echo " M : Menu"

    read -n 1 choice

    case "$choice" in

```

```

    [gG] ) game ;;
    [mM] ) menu ;; # je sais je creer une boucle infini dans une boucle
infini ... en cour de correction
    * ) clear; echo " Mauvaise saisie , recommencez" ;;
    esac
done
}

# Génération du fichier joueurs
function genplayer {
echo "$nom:$descr:$vie:$vieres:$arm:$lieu" > $pos/joueur
}

# Sert a recharger en mémoire les données du personnage
function loadplayer {
nom=$(cat $pos/joueur | cut -d: -f1)
descr=$(cat $pos/joueur | cut -d: -f2)
vie=$(cat $pos/joueur | cut -d: -f3)
vieres=$(cat $pos/joueur | cut -d: -f4)
arm=$(cat $pos/joueur | cut -d: -f5):$(cat $pos/joueur | cut -d: -f6):$(cat
$pos/joueur | cut -d: -f7)
lieu=$(cat $pos/joueur | cut -d: -f8)
cd $pos/$lieu
}
#affiche la progression du joueur ## je me trouve trop bon avec lui quand
meme, il pourrait manger des carottes pour la mémoire

# Génère progressivement la carte
function gencarte {

if [ $(echo "$lieu" | grep -c "lns") -eq 0 ]; then # les passage secret
n'apparaisse pas sur la carte pour l'instant
    echo " $lieu" >> $pos/carte
    sort -d -f -b $pos/carte | uniq > $pos/tmp
    mv $pos/tmp $pos/carte
fi
}

```

datagame.sh

```

#!/bin/bash

### Gestion des données relatives au jeu lui meme
# del
# Sert a supprimer complètement tout ce qui rapport au jeu
# newgame
# Sert a lancer une nouvelle partie
# initgame
# Sert a initialiser entierement le jeu
# recupgame
# Sert a recuperer une partie sauvegarder
# Laisse le choix au joueurs de la sauvegarde utiliser
# savegame
# Sauvegarde le jeu en cour et quitte la partie

# il y a plusieurs cas ou il faut détruire le jeu
function del {
    cd $pos
    chmod -R 700 *
}

```

```

    rm -rf $pos/carte $pos/joueur 2>/dev/null # si le jeu n'a pas commencer
, on redirige l'erreur vers le néant
    # rf car si un ... bref a creer un dossier , entre temps , on s'assure
qu'on puisse jouer
    rm -rf $pos/entree/ # on detruit le labyrinth et tout ce qu'il contient
}

# on recommence :)
function newgame {
clear
echo "$@"
choice= #choice etant une variable utiliser plusieurs fois , on la libère
sous peine de résultat hasardeux #en théorie n'arrive pas dans la dernière
version du code vu que l'on recharge par default si le fichier existe
if [[ -e $pos/joueur ]]; then
    while true ;do
        echo " Une partie en cour : Recommencer quand même ?"
        echo ""
        echo " ===== Choisissez une option ====="
        echo " N : Non"
        echo " Y : Oui"
        read -n 1 choice

        case "$choice" in
            [yYnN] ) break ;;
            * ) clear; echo " Mauvaise saisie , recommencez" ;;
        esac
    done
fi

choice=${choice:-Y} # si le jeu n'etait pas lancer, le choix par default est
oui

if [[ $choice = Y || $choice = y ]];then
case "$choice" in
    [nN] ) clear;echo " Retour a la partie " break ;;
    [yY] ) initgame ;;
esac
else
clear;echo " Retour a la partie ";
fi
}

# on initialise le jeu
function initgame {
    cd $pos # on revien au debut
    del # on fait le menage
    initlab # on initialise le terrain de jeu
    player # on recreer un joueur
}
# recupere la partie sauvegarder
function recupgame {
clear
echo "$@"

if [ $(ls $pos/save | grep -wc "tar.bz2" ) -eq 0 ];then
menu "Aucune Partie Sauvegarder"
fi

while true ;do
echo " Reprendre la partie depuis une dernière sauvegarde ?"

```

```

echo " ===== Choisissez une option ===== "
echo " N : Non"
echo " Y : Oui"
echo -n " "
read -n 1 choice

case "$choice" in
    [nNyY] ) break ;;
    * ) clear; echo " Mauvaise saisie , recommencez" ;;
esac
done

ifs=$IFS
if [[ $choice = Y || $choice = y ]];then
clear;echo ""
while true ;do

    echo " ===== Quel partie reprendre ? ===== "
    echo ""
    n=0
    # pour tout les fichiers du repertoire courant
    IFS="$(echo -e "\n\r")" #on change le séparateur de bash par default
momentanement
    for fil in $(find $pos/save/* -maxdepth 0 -type f 2>/dev/null) ;do
        n=$((n+1))
        echo " $n : $(basename $fil)"
    done

    read -n 1 choice

    case "$choice" in
        # on reliste les partie disponible dans le dossier
        [1-$n] ) break ;;
        * ) clear ; echo " Mauvaise saisie, recommencez" ;;
    esac
done

    cd $pos # on veut etre sur que l'autre n'a pas une partie en cour qui
fausse tout
    del #on degage son jeu en cour
    save=$(basename $(find $pos/save/* -maxdepth 0 -type f | sed -n
$choice\p ))
    cp save/"$(echo "$save" | sed 's/ /\ /g')" $pos # obliger de copier a
cause d'un bug a la decompression # pas trop grave
    tar xjf "$(echo "$save" | sed 's/ /\ /g')" # on selectionne la
sauvegarde voulu
    rm "$(echo "$save" | sed 's/ /\ /g')"
    IFS=$ifs #on retabli le séparateur
    loadplayer
    loadright # on remplace le chmod enlever pour la sauvegarde
    cd $pos/$lieu #on retourne la ou il etait
    clear; echo " Parti recharger depuis la sauvegarde"

    else #si le choix etait non
    clear; echo "Aucun Chargement effectuer"
    fi
    IFS=$ifs
}

# sauvegarde le jeu et quit
function savegame {

```

```

clear
echo $@

if [[ ! -e $pos/joueur ]]; then menu " Aucun jeu en cour" ; fi

while true ;do
    echo " Sauvegarder et quitter ?" # on m'a dit qu'on pouvait pas
choisir de save sans quitter
    echo " ===== Choisissez une option ===== "
    echo " N : Non"
    echo " Y : Oui"
    read -n 1 choice

    case "$choice" in
        [nNyY] ) break ;;
        * ) clear; echo " Mauvaise saisie , recommencez" ;;
    esac
done

if [[ $choice = Y || $choice = y ]];then
    clear;echo ""
    cd $pos/entree
    chmod -R 700 * # on peut pas tar si des dossier sont chmod
    echo " ===== Entré un nom pour la partie svp ? ===== "
    echo ""
    echo -n ""
    read choice
    choice=$(echo "$(date +%Y-%m-%d) | $choice" | sed 's/\:/\ /g') # on
veut pas de ":")
    cd $pos #(on se replace a la racine )
    #j'aime le bz2 ...
    tar -jcf "$(echo "$choice" | sed 's/ /\ /g').tar.bz2" entree/ joueur
carte #on recupère les donné du jeu

    if [ $? -eq 0 ] ;then
        mv "$(echo "$choice" | sed 's/ /\ /g').tar.bz2" save/ #obliger de
deplacer sinon bug ## a fixé
        del #on supprime tout
        clear; echo "Parti Sauvegardé"; exit
    else
        clear; echo " Erreur inattendu, sauvegarde non effectuer
correctement"
        fi

        else #si le choix etait non
        clear; echo " Aucune Sauvegarde Effectuer "
        fi
    }

# Recharges les droits des dossiers contenant des enigmes
function loadright {
cd $pos/entree
dir="$(find * -type d)"

for fil in ${dir}; do
    if [ $(cat $pos/entree/${fil}/enigme | wc -w ) -ne 0 ];then # si le
fichier enigme existe
        domhc="$(find $pos/entree/${fil}/* -maxdepth 0 -type d)"
        for smurf in ${domhc}; do
            echo "$(echo "$smurf"| wc -m ):${smurf}" >> $pos/tmp2
        done
    fi
done

```

```

        fi
done

cat $pos/tmp2 | sort -gr > $pos/tmp

for fil in $(cat $pos/tmp); do
    chmod 000 $(echo "$fil" | cut -d: -f2)
done
rm $pos/tmp $pos/tmp2
}

```

initgame.sh

```

#!/bin/bash

### Relatif a la génération du terrain complet
# initlab
# Sert a creer le labyrinth
# Initialiser les fichiers plats

function initlab {
# génération en dur du labyrinth
mkdir entree

mkdir entree/salle1
mkdir -p entree/salle2/salle3/salle4/salle5/salle6
mkdir -p entree/salle2/salle3/salle4/salle5/salle7
mkdir -p entree/salle2/salle3/salle8/salle9
mkdir -p entree/salle2/salle10/salle11/salle12
mkdir -p entree/salle2/salle10/salle11/salle13/salle14
mkdir -p entree/salle15/salle16/salle17/salle18/salle19
mkdir -p entree/salle15/salle16/salle20/salle21
mkdir -p entree/salle15/salle16/salle20/salle22
mkdir -p entree/salle15/salle23/salle24/salle25
cd $pos/entree/salle1/ ; ln -s $pos/entree/salle15/salle16/salle17/salle18/
lns811
cd $pos/entree/salle15/salle23/ ; ln -s
$pos/entree/salle2/salle10/salle11/salle13/ lns3132
cd $pos/entree/salle15/salle16/salle20/salle21/ ; ln -s
$pos/entree/salle2/salle3/salle4/salle5/ lns512
cd $pos/entree/salle2/salle10/salle11/salle12/ ; ln -s
$pos/entree/salle15/salle16/salle20/salle22/ lns2221

### génération des fichiers vide du jeu complet
cd $pos/entree
dir="$(find * -type d)"
for fil in ${dir}; do
    touch ${fil}/armes
    touch ${fil}/enigme
    touch ${fil}/ennemis
    touch ${fil}/potions
    touch ${fil}/description
    touch ${fil}/simlink
done

### on selectionne aleatoirement la sortie parmi les plus profond du jeu :
4 possible
dir="$(find * -maxdepth 4 -mindepth 4 -type d | shuf -n 1)"
mv $dir "$(dirname $dir)/sortie"

```

```

chmod -R 700 * # tout les droits partout

### génération des fichier de la case d'entree
touch $pos/entree/armes
touch $pos/entree/enigme
touch $pos/entree/ennemis
touch $pos/entree/potions
touch $pos/entree/simlink
echo " Vous etes dans une pièce noir, seul le contour des portes relui" >
$pos/entree/description

### génération aleatoire des fichiers
## ennemis

cat $pos/bdd/ennemis > $pos/bdd/tmp # on ne manipule pas la bdd directement
# ennemis couche 1 # 13 dossier sur les 3 premier niveau , 5 mob faible
dir="$(find * -maxdepth 2 -mindepth 0 -type d | shuf -n 5)"
n=9 # 5 parmi 9 = 4 restant
for fil in ${dir}; do
    rand=$((($RANDOM%n+1)) # on choisit parmi les n premier de la liste
    (qui decrois au fur et a mesure de fait )
    cat $pos/bdd/tmp | sed -n $rand\p >> $pos/entree/${fil}/ennemis
    tmp=$(cat $pos/bdd/tmp | sed $rand\ d) # on supprime l'ennemi de la bdd
    echo "${tmp}" > $pos/bdd/tmp
    n=$((n-1)) # on decremente le $range disponible pour la selection
done

# ennemis couche 2 # 14 dossiers sur le niveau 3-4 # on pop 3 mob un
# peu plus fort
dir="$(find * -maxdepth 3 -mindepth 2 -type d | shuf -n 3)"
n=6 # 3 parmi 6 = 3 + 4 restant = 7
for fil in ${dir}; do
    rand=$((($RANDOM%n+5)) # il reste 4 mob faible, on choisit 3 parmi les
    6 suivants
    cat $pos/bdd/tmp | sed -n $rand\p >> $pos/entree/${fil}/ennemis
    tmp=$(cat $pos/bdd/tmp | sed $rand\ d) # on supprime l'ennemi de la bdd
    echo "${tmp}" > $pos/bdd/tmp
    n=$((n-1))
done

# ennemis couche 4 # 12 dossier sur le niveau 4-5 # on pop 2 mob quest
# sur 4
dir="$(find * -maxdepth 4 -mindepth 3 -type d | shuf -n 2)"
n=4 # 2 parmi 4
for fil in ${dir}; do
    rand=$((($RANDOM%n+8)) # il reste 7 mob faible, on choisit 2 parmi les
    4 suivants
    cat $pos/bdd/tmp | sed -n $rand\p >> $pos/entree/${fil}/ennemis
    tmp=$(cat $pos/bdd/tmp | sed $rand\ d) # on supprime l'ennemi de la bdd
    echo "${tmp}" > $pos/bdd/tmp
    n=$((n-1))
done

## armes
# meme chose que pour les ennemis , on fait evoluer l'arme au fur et a
# mesure
cat $pos/bdd/armes > $pos/bdd/tmp # on ne manipule pas la bdd directement
# ennemis couche 1 # 13 dossier sur les 3 premier niveau , 5 mob faible
dir="$(find * -maxdepth 1 -mindepth 0 -type d | shuf -n 5)"
n=9 # 5 parmi 9 = 4 restant

```

```

for fil in ${dir}; do
    rand=$((($RANDOM%n+1)) # on choisit parmi les n premier de la liste
    (qui decroiss au fur et a mesure de fait )
    cat $pos/bdd/tmp | sed -n $rand\p >> $pos/entree/${fil}/armes
    tmp=$(cat $pos/bdd/tmp | sed $rand\p) # on supprime l'ennemi de la bdd
    echo "${tmp}" > $pos/bdd/tmp
    n=$((n-1)) # on decremente le $range disponible pour la selection
done

    # ennemis couche 2 # 14 dossiers sur le niveau 3-4 # on pop 3 mob un
    peu plus fort
    dir="$(find * -maxdepth 2 -mindepth 2 -type d | shuf -n 3)"
    n=6 # 3 parmi 6 = 3 + 4 restant = 7
    for fil in ${dir}; do
        rand=$((($RANDOM%n+5)) # il reste 4 mob faible, on choisit 3 parmi les
        6 suivant
        cat $pos/bdd/tmp | sed -n $rand\p >> $pos/entree/${fil}/armes
        tmp=$(cat $pos/bdd/tmp | sed $rand\p) # on supprime l'ennemi de la bdd
        echo "${tmp}" > $pos/bdd/tmp
        n=$((n-1))
    done

    # ennemis couche 4 # 12 dossier sur le niveau 4-5 # on pop 2 mob quest
    sur 4
    dir="$(find * -maxdepth 3 -mindepth 3 -type d | shuf -n 2)"
    n=4 # 2 parmi 4
    for fil in ${dir}; do
        rand=$((($RANDOM%n+8)) # il reste 7 mob faible, on choisit 2 parmi les
        4 suivants
        cat $pos/bdd/tmp | sed -n $rand\p >> $pos/entree/${fil}/armes
        tmp=$(cat $pos/bdd/tmp | sed $rand\p) # on supprime l'ennemi de la bdd
        echo "${tmp}" > $pos/bdd/tmp
        n=$((n-1))
    done

## potions
    # on va pop plus de potions sur le debut qu'a la fin, le joueur peu
    aisement revenir en chercher si besoin
    # 7 potions sur les 13 dossiers disponible dans les 3 premier couches
    cat $pos/bdd/potions > $pos/bdd/tmp
    dir="$(find * -maxdepth 1 -mindepth 0 -type d | shuf -n 4)"
    for fil in ${dir}; do
        rand=$((($RANDOM%10+1))
        cat $pos/bdd/tmp | sed -n $rand\p >> $pos/entree/${fil}/potions
    done
    dir="$(find * -maxdepth 2 -mindepth 0 -type d | shuf -n 4)"
    for fil in ${dir}; do
        rand=$((($RANDOM%10+1))
        cat $pos/bdd/tmp | sed -n $rand\p >> $pos/entree/${fil}/potions
    done
    dir="$(find * -maxdepth 3 -mindepth 0 -type d | shuf -n 4)"
    for fil in ${dir}; do
        rand=$((($RANDOM%10+1))
        cat $pos/bdd/tmp | sed -n $rand\p >> $pos/entree/${fil}/potions
    done
    dir="$(find * -maxdepth 4 -mindepth 0 -type d | shuf -n 4)"
    for fil in ${dir}; do
        rand=$((($RANDOM%10+1))
        cat $pos/bdd/tmp | sed -n $rand\p >> $pos/entree/${fil}/potions
    done

```

```

# on execute la boucle plusieurs fois en augmentant le niveau de profondeur
, ainsi on a moins de potions dans les limbes donc plus de chance d'en
retrouver en retournant en arriere
# au total 16 potions

```

```

## descriptions
# il y a 25 description a pop aleatoirement so ...
cat $pos/bdd/description > $pos/bdd/tmp
dir="$(find * -type d)"
n=25
for fil in ${dir}; do
    rand=$((RANDOM%n+1))
    cat $pos/bdd/tmp | sed -n $rand\p > $pos/entree/${fil}/description
    tmp=$(cat $pos/bdd/tmp | sed $rand\p)
    echo "${tmp}" > $pos/bdd/tmp
    n=$((n-1))
done

```

```

## enigmes
# il n'y a pas de niveau de difficulté a gérer so ...
# on pop 9 enigme dans 9 salles aleatoires
cat $pos/bdd/enigme > $pos/bdd/tmp
dir="$(find * -type d | shuf -n 9)"
n=14
for fil in ${dir}; do
    rand=$((RANDOM%n+1))
    cat $pos/bdd/tmp | sed -n $rand\p > $pos/entree/${fil}/enigme
    tmp=$(cat $pos/bdd/tmp | sed $rand\p)
    echo "${tmp}" > $pos/bdd/tmp
    domhc="$(find $pos/entree/${fil}/* -maxdepth 0 -type d)"
    for smurf in ${domhc}; do
        echo "$(echo "$smurf" | wc -m ):${smurf}" >> $pos/bdd/tmp2
    done
    n=$((n-1))
done

```

```

cat $pos/bdd/tmp2 | sort -gr > $pos/bdd/tmp

```

```

for fil in $(cat $pos/bdd/tmp); do
    chmod 000 $(echo "$fil" | cut -d: -f2)
done

```

```

rm $pos/bdd/tmp $pos/bdd/tmp2

```

```

}

```

gameplay.sh

```

#!/bin/bash

```

```

#### Relatif au commande du jeu
# opposants
# Combat contre les mob
# enigma
# Gère les question réponse aux énigme
# potion
# Gère l'injection des potions
# arme
# Gère le changements d'armes

```

```

# lostlife
# Gère la perte des points de vie et la fin de la partie

# Fonction gérant les combats
function opposents {
clear; echo ""
echo " ===== Phase de Combat ===== "
n=0
degplayer=$(echo "$arm" | cut -d: -f3)

while [ $(cat $pos/$lieu/ennemis | wc -w ) -gt 0 ] ; do
n=$((n+1))
echo " ===== Mob $n ===== "
moblife=$(cat $pos/$lieu/ennemis | sed -n 1p | cut -d: -f3) # on
recupere la vie du premier mob

i=0
while [ $moblife -gt 0 ]; do # si la vie du joueurs tombe en
dessous de 0 la fonction lostlife gère la perte
i=$((i+1))
echo " ===== Tour $i ===== "
rand=$((RANDOM%2)) # Entre 0 et 1 pour 50% de chance
if [ $rand -eq 0 ]; then
moblife=$((moblife-$degplayer))
echo " Joueur : Vous avez infligé $degplayer au mob $n"
else
echo " Joueur : Vous avez raté le mob $n"
fi
rand=$((RANDOM%3)) # on génère entre 0 et 2
if [ $moblife -gt 0 -a $rand -eq 1 ]; then
echo -n " le Mob $n vous a touché "
lostlife $(cat $pos/$lieu/ennemis | sed -n 1p | cut -d: -f4 )
elif [ $moblife -gt 0 ]; then
echo " Le Mob $n vous a raté "
fi

if [ $moblife -le 0 ];then
echo "$(cat $pos/$lieu/ennemis | sed 1d )" > $pos/$lieu/tmp
mv $pos/$lieu/tmp $pos/$lieu/ennemis
fi
sleep 0.25
done
done
echo ""
echo " Appuyer sur une touche pour continuer ... "
read -n 1
clear; echo ""
}

# Fonction gérant la réponse aux enigmes
function enigma {
clear; echo ""
while true; do
echo " ===== "
echo " Enigme : $(cat $pos/$lieu/enigme | cut -d: -f1 )"
echo " $(cat $pos/$lieu/enigme | cut -d: -f2 ) ?"
echo " ===== "
echo " 1 : $(cat $pos/$lieu/enigme | cut -d: -f3 ) | 2 : $(cat
$pos/$lieu/enigme | cut -d: -f4 )"

read -n 1 choice

```

```

        case $choice in
        [1-2] ) break ;;
        * ) clear; echo " Mauvaise saisie, recommencez "
        esac
done

        if [ $(cat $pos/$lieu/enigme| cut -d: -f $((2+$choice)) ) = $(cat
$pos/$lieu/enigme| cut -d: -f5 ) ]; then
        clear; echo " Bravo l'enigme est résolu"
        rm $pos/$lieu/enigme
        touch $pos/$lieu/enigme
        for dir in $(find $pos/$lieu/* -maxdepth 0 -type d 2>/dev/null) ;do
            chmod 700 $dir
        done
        else
        clear; echo " Vous vous êtes trompé : "
        #lostlife 2
        fi
    }

# Fonction gérant le gain de point de vie
function potion {
    if [ $vie -eq $vieres ];then
        clear; echo " Vos point de vie sont au maximum"
    else
        clear; echo ""
        while true; do
            echo " ===== "
            echo " Prendre la potions : "
            echo " ===== "
            n=0
            ifs=$IFS
            IFS="$(echo -e "\n\r")"
            for fil in $(cat $pos/$lieu/potions | sed -n p ); do
                n=$((n+1))
                echo -n " Potion $n : $(cat $pos/$lieu/potions | sed -n $n\p |
cut -d: -f1 )"
                echo -n " : $(cat $pos/$lieu/potions | sed -n $n\p | cut -d: -f2 )"
                echo " | Vie Rendu : $(cat $pos/$lieu/potions | sed -n $n\p | cut
-d: -f3 )"
            done
            IFS=$ifs
            read -n 1 choice
            echo " ===== "
            case "$choice" in
            [1-$n] )
                viepros=$vieres
                vieres=$(( $vieres + $(cat $pos/$lieu/potions | sed -n $choice\p | cut
-d: -f3 )))

                if [ $vieres -gt $vie ]; then
                    vieres=$vie
                fi

                # on supprime la ligne correspondant a la potion ingurgité
                echo "$(cat $pos/$lieu/potions | sed $choice\p )" >
$pos/$lieu/potions

                clear; echo " Vous avez remontez vos pv de $(( $vieres - $viepros ))
";break;;
            esac
        done
    fi
}

```

```

        * ) clear; echo " Mauvaise saisie, recommencez "
    esac
done
fi
}

```

Echange l'arme au sol avec celle en main

```

function arme {
    armch=$(cat $pos/$lieu/armes)
    echo "$arm" > $pos/$lieu/armes
    arm=$armch
    clear; echo " Armes Changé ... "
}

```

Fonction gérant la perte de point de vie et la mort.

```

function lostlife {

vieres=$((vieres-$1))

if [ $vieres -le 0 ];then
    while true ;do
        echo ""
        echo ""
        echo " Vous avez perdu : "
        sleep 1
        echo ""
        echo " M : Retour au menu "
        echo " Q : Quitter "

        read -n 1 choice

        case $choice in
            [mM] ) del; clear; menu ;;
            [qQ] ) del; clear; exit ;;
            * )
        esac
    done
else
    echo " vous avez perdu $1 points de vie"
fi
}

```

game.sh

```
#!/bin/bash
```

Fonction relative au deroulement de la partie

game

secret passage

affiche si un passage secret existe et creer un marqueur pour

win

si le joueur a gagner ...

Relatif au déroulement du jeux.

```
function game {
```

```

if [[ ! -e $pos/joueur ]]; then menu " Aucun jeu en cour"
fi

```

```
clear
```

```

echo $@
while true ;do
    if [ $(basename $lieu) = sortie -a $(cat $pos/$lieu/ennemis | wc -w ) -
eq 0 -a $(cat $pos/$lieu/enigme | wc -w ) -eq 0 ]; then
        ## si le joueur est dans la case de sortie il a gagné # si il clean la
salle bien sur
            win
        fi

        genplayer
        # on génère la carte en meme temps
        gencarte

        echo " ===== Partie en cour ====="
        echo -n " Lieu : $(basename $lieu)"
        echo "      Vie : $vieres / $vie"
        echo "$(cat $pos/$lieu/description)"
        echo ""
        echo " ===== "
        ifs=IFS ## obliger de changer le separateur pour la boucle for ...
        IFS="$(echo -e "\n\r)"
        if [ $(cat $pos/$lieu/ennemis | wc -w ) -ne 0 ];then ##### Affichage
des ennemis
            n=0
            for fil in $(cat $pos/$lieu/ennemis | sed -n p ); do
                n=$((n+1))
                echo -n " Ennemis $n : $(cat $pos/$lieu/ennemis | sed -n $n\p |
cut -d: -f1 )"
                echo -n " : $(cat $pos/$lieu/ennemis | sed -n $n\p | cut -d: -f2 )"
                echo -n " | Vie : $(cat $pos/$lieu/ennemis | sed -n $n\p | cut -d:
-f3 )"
                echo -n " | Degat : $(cat $pos/$lieu/ennemis | sed -n $n\p | cut -d:
-f4 )"
            done
            else
                echo " Il n'y a pas d'ennemis"
            fi
            echo ""
            if [ $(cat $pos/$lieu/enigme | wc -w ) -ne 0 ];then ##### Affichage des
enigmes
                echo -n " Enigme : $(cat $pos/$lieu/enigme| cut -d: -f1 )"
                echo -n " : $(cat $pos/$lieu/enigme | cut -d: -f2 )"
            else
                echo " Il n'y a pas d'énigmes"
            fi
            echo ""
            echo " ===== "

            if [ $(cat $pos/$lieu/potions | wc -w ) -ne 0 ];then ##### Affichage
des potions
                n=0
                for fil in $(cat $pos/$lieu/potions | sed -n p ); do
                    n=$((n+1))
                    echo -n " Potion : $(cat $pos/$lieu/potions | sed -n $n\p | cut
-d: -f1 )"
                    echo -n " : $(cat $pos/$lieu/potions | sed -n $n\p | cut -d: -f2 )"
                    echo -n " | Vie Rendu : $(cat $pos/$lieu/potions | sed -n $n\p | cut
-d: -f3 )"
                done
                else
                    echo " Il n'y a pas de potions"
                fi
            fi
        fi
    fi
done

```

```

fi

if [ $(cat $pos/$lieu/armes | wc -w ) -ne 0 ];then ##### Affichage des
armes
    echo -n " Armes : $(cat $pos/$lieu/armes| cut -d: -f1 )"
    echo -n " : $(cat $pos/$lieu/armes | cut -d: -f2 )"
    echo " | Degat : $(cat $pos/$lieu/armes | cut -d: -f3 )"

else
echo " Il n'y a pas d'armes "
fi
IFS=$ifs

echo ""
#####
#####
#####
echo " ===== Choisissez une option ====="
echo -n " Allez a : "
n=0
# pour tout les dossier du repertoire courant
for dir in $(find $pos/$lieu/* -maxdepth 0 -type d 2>/dev/null) ;do
#on veut pas afficher qu'il n'y a pas de sous dossier ...
n=$((n+1))
echo -n " $n : $(basename $dir) "
done; echo "";echo ""
if [ $(cat $pos/$lieu/simlinked 2>/dev/null | wc -w ) -eq 0 ] ; then
echo -n ""
else
echo " 0 : Remonté le labyrinth ? "
fi
echo ""
echo ""

if [ $(cat $pos/$lieu/ennemis | wc -w ) -ne 0 ];then
echo -n " C : Allez au combat "
fi
if [ $(cat $pos/$lieu/enigme | wc -w ) -ne 0 ];then
echo " E : Répondre à l'enigme "
fi
if [ $(cat $pos/$lieu/potions | wc -w ) -ne 0 ];then
echo -n " P : Prendre la potion "
fi
if [ $(cat $pos/$lieu/armes | wc -w ) -ne 0 ];then
echo " A : Prendre l'arme "
fi;echo ""

# on recupere l'arboresence sans le repertoire courant
lieureturn=$(dirname $lieu)

if [ $(basename $lieu) != entree ]; then
echo -n " R : retour en arrière ($(basename $lieureturn) ) |"
fi

echo " M : Menu "
echo ""
if [ $(cat $pos/$lieu/simlink | wc -w ) -eq 0 ]; then
echo " L : Recherche de passage secret ( coute 1 pdv )"
else
echo " L : Emprunter le passage secret"
fi

```

```

read -n 1 choice

#####
#####
#####

case "$choice" in
    # on reliste les fichier disponible dans le dossier, on extrait
    # celui voulu avec sed et on fait un cd dessus
    [1-$n] )
        depl=$(basename $(echo "$(find $pos/$lieu/* -maxdepth 0 -type d)" |
sed -n $choice\p ))
        if [ $(cat $pos/$lieu/ennemis | wc -w ) -eq 0 -a $(cat
$pos/$lieu/enigme | wc -w ) -eq 0 ];then
            #if [ -x $pos/$lieu/$depl -o $(cat $pos/$lieu/ennemis | wc -w ) -eq
0 ]; then # si on a les droit de passage OU si les ennemis sont défait
            #if [[ -e simlinked ]]; then
            #rm simlinked
            #fi #
            cd $depl;lieu=$lieu/$depl;clear;echo""
            else
            depl=""
            clear; echo " Ennemis ou Enigme Restante(s) "
            fi ;;
        0 )
            if [ $(cat $pos/$lieu/simlinked 2>/dev/null | wc -w ) -eq 0 ] ; then
# dev null cart ca n'existe pas tjrs
            echo " Mauvaise saisie , recommencez"
            else
            if [ $(cat $pos/$lieu/ennemis | wc -w ) -eq 0 ]; then # si on est
passer c'est qu'il n'y a pas d'enigme au dessus, de meme pour revenir on a
pas a resoudre les enigmes des portes
                depl=$(dirname $(cat simlinked));rm simlinked
                cd $depl;lieu=$(echo "$depl" | sed s#$pos/## ) # la carte
devient le lieu de passage
                clear;echo""
            else
                clear; echo " Il reste des ennemis "
            fi
            fi;;

    [rR] )
        if [[ -e $pos/$lieu/simlinked ]]; then
            rm $pos/$lieu/simlinked
        fi
        if [ $(basename $lieu) != entree ]; then # on ne veut pas que le
joueurs sorte du circuit
            cd .. ; lieu=$lieureturn ;clear;echo""
        else
            clear; echo " Mauvaise saisie , recommencez"
        fi ;;
    [cC] )
        if [ $(cat $pos/$lieu/ennemis | wc -w ) -ne 0 ];then
            oppponents
        else
            clear; echo " Mauvaise saisie , recommencez"
        fi ;;
    [eE] )
        if [ $(cat $pos/$lieu/enigme | wc -w ) -ne 0 ];then

```

```

        enigma
    else
        clear; echo " Mauvaise saisie , recommencez"
    fi ;;
    [pP] )
    if [ $(cat $pos/$lieu/potions | wc -w ) -ne 0 ];then
        potion
    else
        clear; echo " Mauvaise saisie , recommencez"
    fi ;;
    [aA] )
    if [ $(cat $pos/$lieu/armes | wc -w ) -ne 0 ];then
        arme
    else
        clear; echo " Mauvaise saisie , recommencez"
    fi ;;
    [mM] ) menu ;;
    [vV] ) clear; lostlife 1 ;;
    [LL] )
    if [ $(cat $pos/$lieu/simlink | wc -w ) -eq 0 ]; then
        secretpassage
    else
        depl=$(basename $(find $pos/$lieu/* -maxdepth 0 -type l)); # echo
"$lieu" > $pos/$lieu/$depl/simlink
        if [ $(cat $pos/$lieu/ennemis | wc -w ) -eq 0 -a $(cat
$pos/$lieu/enigme | wc -w ) -eq 0 ]; then # si on a les droit de passage
            sim=$(cat simlink)
            cd $depl;lieu=$lieu/$depl;clear;
            echo "$sim" > simlinked ## on pose un marqueur avec le lien
            echo "Vous êtes passer dans un tunnel sombre"
        else
            clear; echo " Ennemis ou Enigme Restante(s)"
        fi
    fi ;;
    * ) clear; echo " Mauvaise saisie , recommencez" ;;
esac

done

}

# Gère la recherche des passages secrets
function secretpassage {

    if [ $(find * -maxdepth 0 -type l |wc -w ) -ne 0 ];then

        readlink -f $(find * -maxdepth 0 -type l) > $pos/$lieu/simlink

        clear; echo " Passage secret trouvé : "
    else
        clear; echo " Aucun passage secret détecté : "
    fi
    #lostlife 1 # a decommenté pour payer la recherche des PV
}

# Si le joueurs trouve la sortie
function win {

    while true ;do
        clear
        echo ""

```

```

echo " Bravo vous avez gagné"
echo ""
echo " M : Retour au menu "
echo " Q : Quitter "

read -n 1 choice

case $choice in
    [mM] ) del; clear; menu ;;
    [qQ] ) del; clear; exit ;;
    * ) clear;echo "" ;;
esac
done

```

```

}
```

Bonus

Le script qui m'a permis la recherche d'erreur en temps réel pour beta-tester le jeux

```

#!/bin/bash
n=0
while true;do
    clear; echo "$n"
    cat error
    n=$((n+1))
    sleep 1
done
# Ctrl + C pour quitter

```