# Astérix Perdu Dans La Pyramide

Projet UNIX

AFLALO – SASPORTAS – SAYADA

L2 Promo 2015 Groupe B

efrei PARIS SUD Ingénieurs en mouvement

## TABLE DES MATIERES

## INTRODUCTION

Astérix est perdu dans la pyramide … Il faut l'aider à en sortir ! Heureusement vous, utilisateur êtes là pour ça ! Oui il va falloir vous déplacer dans toutes ces salles, un vrai labyrinthe va s'offrir à vous.

Mais ce n'est pas aussi simple que vous le pensez, détrompez-vous, ennemis (comme Amonbeaufils voire même MenuBestOfPlus) et énigmes vont tout faire pour vous empêcher d'arriver à vos fins ! (Rires machiavéliques). Heureusement vous n'êtes pas seuls avec vos petits points, dans la pyramide vous pourriez tomber sur Panoramix, Obélix ou ses menhirs perdus pour vous défendre et vous aider à terminer votre mission.

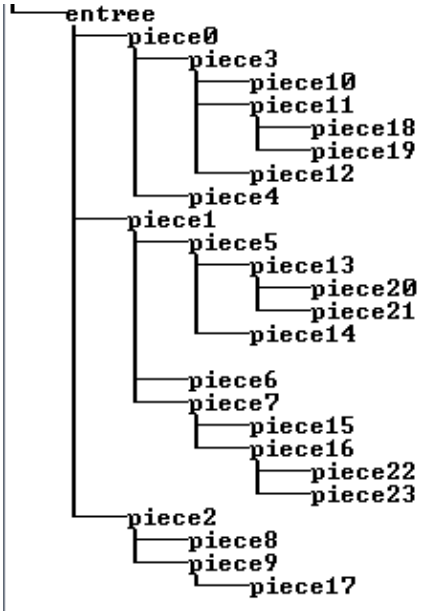
D'autre part, ne pensez pas pouvoir espionner les autres pièces sans y être entré, vous risqueriez de mettre en colère Osiris qui viendrait vous taper sur les doigts, celui-ci a beaucoup de pouvoir étant donné qu'il fait partie des dieux, il faudrait qu'il vous donne les droits mais ce n'est pas gagné !

Ce jeu développé dans un script Shell nous permet de compléter et endurcir nos connaissances et notre pratique sous UNIX et ce dans la plupart des outils mis à disposition par le système d'exploitation.

# CONSTRUCTION DE LA PYRAMIDE

## UNE AFFAIRE DE DOSSIERS

La structure des répertoires représente le labyrinthe. Notre arborescence est stocké dans l'archive entree.tar.gz. Nous avons choisi une arborescence de cette forme :




La pièce « sortie » doit changer de place à chaque nouvelle partie. Pour se faire, nous avons créé une fonction *creasortie*.

La fonction *creasortie* a créé un tableau dans lequel on a mis les « feuilles » de l'arbre, soit les dossiers n'ayant aucun enfant. Ainsi on pioche aléatoirement un de ces dossiers et on crée la sortie dans celui-ci, la sortie étant un simple dossier.

## DES ARMES, DES POTIONS, DES ENIGMES, DES MECHANTS

Tout le contenu d'une pièce (objets, ennemis, énigmes, description…) est représenté par des fichiers contenus dans le répertoire symbolisant la pièce.

Nous avons créé un fichier *potions*. Regardons plus en détail la première potion du fichier.

*potion-romaine:Imite mais jamais egale, les romains essaient de copier Panoramix mais ca ne marche pas:2*

Les divers caractéristiques sont séparées par des *:* comme proposé dans l'énoncé. Vous remarquerez également qu'il y a des sauts de lignes dans le fichier. Cela permettra à certaine pièce de ne pas contenir de potion. Le même procédé sera utilisé dans le fichier *armes* et *enigmes*.

Pour remplir certaines pièces d'objets, d'ennemis et d'énigmes, nous avons créé une fonction appelée *remplipiece*. Dans les 10 premières pièces, nous avons uniquement placé les ennemis les plus faibles pour accroitre la difficulté dans la suite du jeu. En effet, chaque ennemi faible a un nombre de point de vie, et nombre de point de dégâts maximum inférieur aux autres ennemis.

Grâce à la commande *shuf* deux ennemis seront choisi au hasard parmi la liste des ennemis possibles (présents dans le fichier ennemis, les lignes vides ne mettent pas de monstre). Nous répéterons l'opération jusqu'à ce que tous les dossiers aient été parcourus. La même commande *shuf* sera utilisée pour les potions, les armes et les énigmes.

Le chemin sera quant à lui trouvé grâce à la commande :

```
chemin=$(find entree/ -name piece$cpt)
```

Ainsi une fois le chemin trouvé, les fichiers contenant les ennemis, les potions, les armes et les énigmes pourront être placés au bon endroit (c'est-à-dire dans la bonne pièce).

## PARTONS A L'AVENTURE

### UN HEROS

C'est la fonction *creaperso* qui va générer le fichier joueur. Le fichier joueur contiendra le surnom de notre héros Astérix, sa description, son arme(les dégâts qu'elle apporte), ainsi que son nombre de vie maximum et sa vie actuelle. Au début de la partie sa vie maximum correspond à sa vie actuelle.

Sa vie maximum sera générée aléatoirement. La fonction *$RANDOM* native de BASH va générer un nombre pseudo-aléatoire entre 0 et 32767 qui va ensuite être modifié par des opérations mathématiques pour donner notre nombre aléatoire compris entre 150 et 300.

L'utilisateur pourra choisir l'une des trois premières armes disponibles dans le fichier *armes*. Il aura donc le choix entre un glaive, une momie, ou Idéfix pour se battre ; chacune des trois armes affligent 2 points.

### DE SALLE EN SALLE

Pour le déplacement du joueur, on va tout d'abord vérifier que nous ne sommes pas dans un cul de sac ! Pour cela on va vérifier qu'il existe bien des sous-dossiers dans le dossier actuel à l'aide des commandes ls, grep et que l'on comptera lorsque cela sera nécessaire avec wc.

On demandera alors au joueur dans quelle pièce il voudra aller, il n'aura qu'à taper le numéro de la pièce correspondante, pour la salle précédente le numéro à taper lui est indiqué à chaque fois. En fait, il faut taper le numéro de la pièce la plus petite, à laquelle on a le choix, soustrait de 1. Par exemple si les salles proposées sont 6, 7, 8, il faudra taper 5.

Lorsque l'utilisateur aura entré une réponse autorisée, le fichier joueur se déplacera dans le dossier suivant (ou précédent) grâce à la commande *mv* et nous suivrons ce fichier à l'aide de la commande *cd*.

Le système syntaxique d'UNIX est intéressant ici puisque il est tout à fait possible d'utiliser la syntaxe : `mv joueur piece$reponse/`

Où *$reponse* représente le numéro de la pièce entré par l'utilisateur. Il va en fait y avoir concaténation entre le mot *piece* et le résultat de la variable *$reponse*.

Il est important de prendre en compte le cas où nous serions dans l'entrée et ainsi refuser un retour en arrière.

Après l'affrontement contre les monstres et avant de vérifier si l'on est dans une impasse, on vérifie si la sortie n'est pas présente ! Sa présence permet au joueur de gagner la partie et la stopper ainsi.

## ARMEMENT

Tout d'abord pour vérifier la présence du fichier d'arme ! Oui même si certains sont vides lorsqu'on en ramasse une, le fichier est détruit et lors d'un retour en arrière cela poserait problème.

Ensuite on vérifie si le fichier n'est pas vide en regardant le nombre de caractères sur la première ligne, on utilise pour cela : `${#ligne}`

S'il ne l'est pas on propose à l'utilisateur de la prendre s'il le désire en comparant les dégâts de cette nouvelle arme avec celle qu'il détient déjà.

Pour afficher les dégâts de cette nouvelle arme on utilise la commande suivante :

```
$(cut -d: -f 3 armes | head -1)
```

Enfin on supprime le fichier *armes*.

## UNE PETITE SOIF ?

Pourquoi ne pas prendre une petite potion ?

Lorsque le joueur arrive dans une salle il peut trouver une potion de vie, qui peut lui faire gagner un nombre de points de vie donné. Comme pour les armes on vérifie si le fichier existe et s'il n'est pas vide.

Ensuite il s'agit d'ajouter les points de vie de la potion aux points de vie du joueur en prenant garde à ne pas dépasser les points de vie maximums.

Une simple addition des variables permet de mettre à jour les points de vie :

```
vie=$((vie + viepotion))
```

## READY, FIGHT !

Pour un combat il est possible d'avoir à affronter jusqu'à deux ennemis. On va donc tester les deux premières lignes pour savoir si des ennemis sont à combattre.

On récupère dans des variables la vie de l'ennemi et sa force :

```
vieennemis=$(cut -d: -f 3 ennemis | head -1)
forceennemi=$(cut -d: -f 4 ennemis | head -1)
```

La commande *cut* va séparer chaque partie de la ligne par le délimiteur *': '*.

C'est un combat à mort. Donc il ne se terminera pas tant que le joueur ou l'ennemis est mort pour cela on utilise une boucle *while*, ensuite pour déterminer le tour de l'attaque, c'est-à-dire si c'est au tour du joueur d'attaquer ou au tour de l'ennemis, on utilise l'opérateur % :

```
if [ $((tour % 2)) -eq 0 ]
```

En initialisant *tour* à 0 on est sûr que le joueur commencera à attaquer.

De même pour la probabilité de toucher ou de se faire toucher lors d'une attaque, on utilise la fonction *$RANDOM* et on utilise % pour calculer si le coup est pertinent ou non. Par exemple :

```
((oasar=$RANDOM))
((oasar=$oasar%3))
if [ $((oasar % 3)) -eq 0 ]
```

Si l'ennemi est mort, on peut le supprimer du fichier texte. Ainsi en supprimant la première ligne, la seconde va se placer en première (commande : `sed -i 1d ennemis`), et une fois tous les ennemis morts il n'y a plus qu'à supprimer le fichier *ennemis*.

## QUELQUES QUESTIONS

Avant de pouvoir changer de salle, il se peut qu'intervienne une énigme. Tout simplement il est nécessaire que l'utilisateur rentre 1 ou 2 suivant la réponse.

La ligne du fichier qui est du type :

*enigme de Numerobis:C'est un homme qui s'appelle On, il a un phare c'est :le Pharaon:le Faraday:le Pharaon*

Il suffit de comparer la réponse entré par l'utilisateur avec la cinquième partie de la ligne (séparation par les deux points). Par exemple si l'utilisateur entre 2 on va comparer : *le Faraday* avec *le Pharaon*. Si les deux expressions sont différentes le joueur doit recommencer.

# LES PLUS

## DES DROITS ~~ET DES DEVOIRS~~

Des droits ? Oui ! Mais pas question de devoirs pour notre perdu dans la pyramide ! Il ne faut surtout pas qu'il puisse connaitre l'arborescence du jeu, ce qui ferait qu'il pourrait tricher et regarder les dossiers l'amenant à la sortie.

Ainsi, en début de partie, nous allons définir récursivement les droits de tous les dossiers présent à partir de notre entrée, on ne les autorisera qu'à l'exécution (par le programme).

```
chmod -R u=--x,g=---,o=---  $cheminjoueur/entree/
```

L'utilisateur pourra donc seulement exécuter les fichiers, et donc ne pas consulter l'arborescence, les monstres qui l'attendent, etc…

À chaque tour, on rend au joueur les droits sur le dossier courant uniquement.

## DES PASSAGES SECRETS

Dans cette pyramide, le passage secret est amovible. Il change de départ et de destination à chaque partie.

Pour ce faire on définit une entrée et une sortie. La sortie étant définit par un lien symbolique sur une pièce tirée au hasard et toujours supérieure à la salle 10 :

```
ln -s piece$sortiepassagesecret passagesec
```

On se déplace dans le dossier parent (du dossier, sortie du passage secret), on crée le lien symbolique sur le dossier en question.

On joue la partie normalement et dans chaque salle on va demander au joueur s'il veut chercher un passage secret. S'il répond oui on va chercher dans le dossier courant le fichier *passagescrchemin* qui symbolise l'entrée.

S'il est présent on se déplace dans la salle détenant le lien symbolique et la partie continue ici.

## AU MENU

**MenuBestOfPlus**

Conformément au cahier des charges, nous avons mis en place un menu, avec un accès possible à n'importe quelle saisie, en saisissant le caractère *m*. Les différentes possibilités du menu sont les suivantes :

1. Lancer une nouvelle partie
2. Continuer une partie sauvegardée
3. Sauvegarder la partie en cours
4. Voir la feuille du personnage
5. Quitter la partie

Pour lancer une nouvelle partie, il faut tout d'abord sauvegarder le chemin initial qu'on utilisera au fil du script. Puis on supprime récursivement le dossier *entree*, pour supprimer l'ancien labyrinthe, puis on extrait l'archive *entree.tar.gz*, contenant notre arborescence de dossiers. Il s'agira alors ici d'appeler les fonctions *creaperso, creasortie* et *remplipiece* pour régénérer un nouveau personnage, avec un nouveau labyrinthe et des pièces rempli différemment. Ensuite, on changera les droits pour les dossiers, et on lancera enfin le jeu.

A la différence de la création de partie, lorsqu'on continue une ancienne partie cela se passe autrement. On extrait l'archive *sauvegarde.tar.gz* dans laquelle a été archivé tout le labyrinthe. On récupère alors un fichier placé à l'entrée dans lequel sont disposées toutes les informations du fichier joueur, pour les mettre dans nos variables de jeu.

Lorsque l'on doit sauvegarder une partie, c'est le moment de sauvegarder toutes nos variables dans le fichier texte *joueur*, qui servira alors de repère pour reprendre cette partie. On copie ce fichier à l'entrée en l'appelant *sauvegarde*. Enfin, on compresse tout le dossier *entree*.

L'option ici présentée est la visualisation de la feuille du personnage, on pourra alors afficher toutes les informations demandées, à savoir le nom, la description, les points de vie courant et maximaux, les dégats de l'arme et le chemin.

La dernière option proposée est la sortie du jeu, elle se fera tout simplement grâce à la commande *exit* qui va arrêter le script.

Pour afficher le menu n'importe où, nous avons disposé dans chaque saisie du jeu, à savoir après la création du personnage, des conditions dans nos saisies. Ainsi on dispose une condition vérifiant si la saisie correspond à notre lettre-menu(m), et si c'est le cas nous lançons la fonction *menu*.

## CONCLUSION

Après de nombreuses péripéties, Astérix a enfin réussi à sortir de la pyramide (Ok, il est peut être mort avant mais bon c'est rare) ! L'aventure n'a pas été si simple, Astérix a affronté des ennemis plus que barbares, et résolu de nombreuses énigmes.  Grâce aux armes, potions et passages secrets qu'il a rencontrés, la victoire a été plus simple.

Plus sérieusement, nous avons réussi à appréhender ce projet après avoir respecté l'ensemble du cahier des charges dans les temps impartis. L'aspect ludique a été une réelle motivation pour l'ensemble de l'équipe. Le sujet a permis de cerner les commandes UNIX dans beaucoup de domaines, au niveau des fichiers, des droits, des scripts (à quand le jeu en réseau ? ).

## ANNEXE : SCRIPT LABYRINTHE.SH

```
#!/bin/bash

function menu
{
	menu=0
	until [ $menu = 1 ] || [ $menu = 2 ] || [ $menu = 3 ] || [ $menu = 4 ] || [ $menu = 5 ]
	do
		clear
		echo "Lancer une nouvelle partie, tapez 1"
		echo "Continuer une partie sauvegardee, tapez 2"
		echo "Sauvegarder la partie en cours, tapez 3"
		echo "Voir la feuille de personnage, tapez 4"
		echo "Quitter, tapez 5"
		read menu

		if [ $menu = 1 ] #Lancer une nouvelle partie: supprimer arborescence et décompresser entree.tar.gz
		then

			cheminjoueur=$(pwd) #Sauvegarde du chemin initial
			rm -R $cheminjoueur/entree 2> /dev/null #Supprimer dossier entree
			tar zxvf $cheminjoueur/entree.tar.gz > /dev/null #Decompresser entree
			#Appel des fonctions de jeu
			creaperso
			creasortie
			remplipiece
			mv joueur entree/
			cd entree/

			chmod -R u=--x $cheminjoueur/entree/ 2> /dev/null
			chmod u=-wx $cheminjoueur/entree/ 2> /dev/null
			finpartie=1
			while [ $finpartie = 1 ];
			do
				chmod 777 $(pwd) 2> /dev/null
				salleensalle
				danslasalle
				sleep 1
				yatillasortie
			done

		elif [ $menu = 2 ] #Décompresser sauvegarde (avec tous les fichiers)
		then
			cheminjoueur=$(pwd)
			rm -R $cheminjoueur/entree 2> /dev/null
			tar zxvf $cheminjoueur/sauvegarde.tar.gz  > /dev/null
			chmod 777 $cheminjoueur/entree/sauvegarde
			#Récupération des variables de jeu depuis le fichier sauvegarde
			nom=$(sed -n 1p $cheminjoueur/entree/sauvegarde)
			description=$(sed -n 2p $cheminjoueur/entree/sauvegarde)
			viemax=$(sed -n 3p $cheminjoueur/entree/sauvegarde)
			vie=$(sed -n 4p $cheminjoueur/entree/sauvegarde)
			pointsarme=$(sed -n 5p $cheminjoueur/entree/sauvegarde)
			jesuis=$(sed -n 6p $cheminjoueur/entree/sauvegarde)
			cd $jesuis

			chmod -R u=--x $cheminjoueur/entree/ 2> /dev/null
			chmod u=-wx $cheminjoueur/entree/ 2> /dev/null
			#On ne laisse que le droit d'exécution à l'utilisateur
			#chmod -R u=--x,g=---,o=--- $cheminjoueur/entree/
			finpartie=1
```

```
                        while [ $finpartie = 1 ];
                        do
                                chmod u=rwx $(pwd) 2> /dev/null
                                salleensalle
                                danslasalle
                                sleep 1
                                yatillasortie
                        done

                elif [ $menu = 3 ] #Compresser dossier sauvegarde
                then
                        echo "$nom" > joueur
                        echo "$description" >> joueur
                        echo "$viemax" >> joueur
                        echo "$vie" >> joueur
                        echo "$pointsarme" >> joueur
                        echo "$(pwd)" >> joueur

                        rm $cheminjoueur/sauvegarde.tar.gz
                        cp joueur $cheminjoueur/entree/sauvegarde
                        cd $cheminjoueur
                        tar -cvzf $cheminjoueur/sauvegarde.tar.gz entree  > /dev/null
                        exit

                elif [ $menu = 4 ] #Voir feuille personnage
                then
                        echo "Vous vous appelez: "
                        sed -n 1p joueur
                        echo "Telle est votre description: "
                        sed -n 2p joueur
                        echo "Votre nombre de points maximal est de : "
                        sed -n 3p joueur
                        echo "Nombre de points de vie courant: "
                        sed -n 4p joueur
                        echo "Votre arme affecte les dégats suivant: "
                        sed -n 5p joueur
                        echo "Vous vous trouvez au chemin suivant: "
                        sed -n 6p joueur

                elif [ $menu = 5 ] #Quitter la partie
                then
                        exit
                fi
        done
}

function creaperso
{
        echo "Entrez le surnom d'Asterix que vous souhaitez"
        read nom
        echo "$nom" > joueur
        echo "Entrez la description de votre heros"
        read description
        echo "$description" >> joueur

        viemax=$RANDOM
        viemax=$((viemax%150+150))
        echo "Votre vie est de $viemax"
        echo "$viemax" >> joueur
        vie=$viemax
        echo "$vie" >> joueur

        echo

        echo "Choisissez votre arme (numero)"
        cat armes | head -n 3 #On affiche les trois premières lignes du fichier armes
```

```
	larme=0
	until [ $larme = 1 ] || [ $larme = 2 ] || [ $larme = 3 ]
	do
		echo "Veuillez rentrer une arme entre 1 et 3"
		read larme
	done
	echo "$(sed -n $((larme))p armes | cut -d: -f 3 )" >> joueur
	pointsarme=$(sed -n 5p joueur)

	echo "$(pwd)" >> joueur
}


# Création de la sortie
function creasortie
{
	#On pioche au hasard la sortie
	tablo=(10 18 19 12 4 20 21 14 6 15 22 23 8 17)
	((temp=$RANDOM))
	((temp=$temp%14))
	sortie=${tablo[temp]}
	#echo $sortie
	chemin=$(find entree/ -name piece$sortie)
	mkdir $chemin/sortie
}

#Remplissage des pieces
function remplipiece
{
	cpt=0
	cat ennemis | head -n 4 > ennemisfaibles
	while [ $cpt -le 23 ];
	do
		chemin=$(find entree/ -name piece$cpt)
		if [ $cpt -le 10 ]
		then
			shuf -n 2 ennemisfaibles >  $chemin/ennemis #On tire deux lignes
au hasard
		else
			shuf -n 2 ennemis > $chemin/ennemis
		fi

		shuf -n 1 potions >  $chemin/potions
		shuf -n 1 armes >  $chemin/armes
		shuf -n 1 enigmes >  $chemin/enigmes
		cpt=$((cpt+1))
	done
	rm ennemisfaibles

	unlink passagesec
	((sortiepassagesecret=$RANDOM))
	((sortiepassagesecret=$sortiepassagesecret%12+11))
	chemin=$(find entree/ -name piece$sortiepassagesecret)
	cd $chemin | cd ..
	ln -s piece$sortiepassagesecret passagesec
	readlink passagesec
	cd $cheminjoueur
	((entreepassagesecret=$RANDOM))
	((entreepassagesecret=$entreepassagesecret%10))
	touch $(find entree/ -name piece$entreepassagesecret)/passagescrchemin
}

function danslasalle
{
#Afficher ici la description de la salle
	echo "Vous êtes dans $(sed -n 1p description | cut -d: -f 1)"
```

```
	echo "$(sed -n 1p description | cut -d: -f 2)"
	sleep 1
##########################################################ARME#########################
##############################
	if [ -s armes ]
	then
		ligne=$(sed -n 1p armes)
		ligne=${#ligne}
		if [ $ligne -gt 1 ]
		then
			echo
			echo "Tiens ! Une arme ! "
			cut -d: -f 1 armes | head -1
			cut -d: -f 2 armes | head -1
			echo "Votre arme actuelle inflige des degats de $pointsarme et
celle ci des degats de $(cut -d: -f 3 armes | head -1)"
			echo "Voulez vous cette arme ? (y/n)"
			rep=0
			until [ $rep = y ] || [ $rep = n ]
			do
				read rep
				if [ $rep = m ] #Dans ce cas, on affiche le menu
				then
					menu

				elif [ $rep = y ]
				then
					pointsarme=$(cut -d: -f 3 armes | head -1)
					echo "Votre arme inflige maintenant des dégats de
$pointsarme"
				elif [ $rep = n ]
				then

					echo "Vous gardez la même arme"
				else
					echo "Recommencez la saisie (y/n)"

				fi
			done
		fi
		rm armes
		sleep 1 #Le script "cesse" une seconde
	fi

##########################################################POTION#######################
##############################
	if [ -s potions ]
	then
		ligne=$(sed -n 1p potions)
		ligne=${#ligne}
		if [ $ligne -gt 1 ]
		then
			echo
			echo "Une potion trouvee : $(cut -d: -f 1 potions | head -1)"
			cut -d: -f 2 potions | head -1
			viepotion=$(cut -d: -f 3 potions | head -1)
			echo "Vous gagnez $viepotion points de vie"
			dif=$((viemax - vie))
			if [ $viepotion -lt $dif ]
			then
				vie=$((vie + viepotion))
			else
				vie=$viemax
			fi
			echo "Votre vie est maintenant de $vie"
		fi
```

```
		rm potions
		sleep 1
	fi


##########################################################MONSTRE#######################
################################
	if [ -s ennemis ]
	then
		for cpt in 1 2;
		do
			ligne=$(sed -n 1p ennemis)
			ligne=${#ligne}
			if [ $ligne -gt 1 ]
			then
				echo
				echo "Vous allez combattre : $(sed -n 1p ennemis | cut -
d: -f 1)"
				sed -n 1p ennemis | cut -d: -f 2
				vieennemis=$(cut -d: -f 3 ennemis | head -1)
				forceennemi=$(cut -d: -f 4 ennemis | head -1)
				tour=0
					while [ $vieennemis -gt 0 ] && [ $vie -gt 0 ]; do
					#echo "Vous avez $vie points de vie et l'ennemi en
a $vieennemis"
					if [ $((tour % 2)) -eq 0 ]
					then
						((oasar=$RANDOM))
						((oasar=$oasar%2))
						if [ $((oasar % 2)) -eq 0 ]
						then
							#echo "Vous attaquez, vous lui
infligez $pointsarme dégats"
							vieennemis=$((vieennemis -
pointsarme))
							#echo "Vie ennemi : $vieennemis"
						# else
							# echo "Vous manquez votre cible"

						fi
					else
						((oasar=$RANDOM))
						((oasar=$oasar%3))
						if [ $((oasar % 3)) -eq 0 ]
						then
							#echo "L'ennemi vous attaque et vous
inflige $forceennemi de dégats"
							vie=$((vie - forceennemi))
							#echo "Votre vie : $vie"
						#else
							#echo "L'ennemi vous manque"

						fi
					fi
					tour=$((tour + 1))
				done
				if [ $vieennemis -le 0 ]
				then
					echo "L ennemi est mort il vous reste $vie points
de vie"
					sed -i 1d ennemis
				else #Fin de la partie
					echo "Vous êtes mort"
					finpartie=0
				fi
			else
```

```
                        sed -i 1d ennemis
                fi
        sleep 1
        done
        rm ennemis
fi


#############################################################ENIGME#####################
##############################
if [ -s enigmes ] && [ $finpartie = 1 ]
then
        ligne=$(sed -n 1p enigmes)
        ligne=${#ligne}
        if [ $ligne -gt 1 ]
        then
                bonnereponse=0
                while [ $bonnereponse -ne 1 ] #Boucle tant que la réponse n'est
pas bonne
                do
                        #Affichage de l'enigme
                        echo
                        echo "Vous faites face a l $(sed -n 1p enigmes | cut -d:
-f 1)"
                        sed -n 1p enigmes | cut -d: -f 2
                        echo "Voici les deux reponses possibles :"
                        reponse=0
                        sed -n 1p enigmes | cut -d: -f 3
                        sed -n 1p enigmes | cut -d: -f 4
                        until [ $reponse = 1 ] || [ $reponse = 2 ];
                        do
                                echo "Entrez 1 ou 2"
                                read reponse
                                if [ $reponse = m ]
                                then
                                        menu

                                fi
                        done
                        if [ "$(sed -n 1p enigmes | cut -d: -f $((reponse + 2)))"
= "$(sed -n 1p enigmes | cut -d: -f 5)" ]
                        then
                                echo "Bonne reponse ! Vous pouvez passer dans la
piece suivante !"
                                bonnereponse=1
                        else
                                echo "Echec EFREI retentez votre chance !"
                        fi
                done
        fi
        rm enigmes
        sleep 1
fi

#########################################################PASSAGE
SECRET##############################################
if [ $finpartie = 1 ]
then
        echo "Voulez vous chercher un passage secret ? (y/n)"
        rep=0
        until [ $rep = y ] || [ $rep = n ]
        do
                read rep
                if [ $rep = m ] #Dans ce cas, on affiche le menu
                then
                        menu
```

```
                    elif [ $rep = y ]
                    then
                           if [ -e passagescrchemin ]
                           then

                                  echo "Il y a un passage ! On y va !"
                                  mv joueur $cheminjoueur
                                  cd $cheminjoueur
                                  mv joueur $(find $cheminjoueur/entree/ -name
$(readlink passagesec))
                                  cd  $(find $cheminjoueur/entree/ -name $(readlink
passagesec))
                                  echo "Vous arrivez du passage secret les ennemis ne
vous voient pas vous pouvez même avancer dans une autre salle discrètement"
                           else
                                  echo "Pas de porte, pas de passage. Pas de passage,
pas de passage !"
                           fi
                    elif [ $rep = n ]
                    then

                           echo "Ok vous le regretterez surement !"
                    else
                           echo "Recommencez la saisie (y/n)"
                    fi
             done
      fi
}

function salleensalle
{
      echo
      if [ "$(ls -F ./ | grep "*/")" = "" ]
      then
             echo "-------------------------------------------"
             echo "Choisissez la prochaine salle à explorer : "
             echo "Salle Precedente"
             ls -1d * | grep 'piece*'

             reponse=-2

             if [ $(ls -l | grep ^d | wc -l) = 0 ]
             then
                    mini=5
             else
                    mini=$(ls -1d * | grep 'piece*' | head -1 | cut -d'e' -f 3)
             fi
             temp=1
             temp2=$(ls -1d * | grep 'piece*' | wc -l)
             temp2=$((temp2 - 1))
             while [ $temp = 1 ];
             do
                    echo "Entrez un nombre correspondant a une salle ($((mini - 1))
pour la precedente)"
                    read reponse
                    if [ $reponse = m ]
                    then
                           menu
                           echo "Rentrez la salle souhaitée"
                           read reponse
                    fi

                    if [ $reponse -eq $((mini - 1)) ] && [ $mini -ne 0 ]
                    then
                           temp=0
```

```
                        chmod 777 ../
                        mv joueur ../
                        cd ..
                        chmod 777 description
                        #ls -l
                elif [ $reponse = $((mini - 1)) ] && [ $mini = 0 ]
                then
                        echo "Impossible ! Vous etes dans l'entree ! Recommencez
la saisie"
                elif [ $reponse -ge $mini ] && [ $reponse -le $((mini + temp2))
]
                then
                        temp=0
                        mv joueur piece$reponse/
                        cd piece$reponse/
                fi
        done
        echo "$nom" > joueur
        echo "$description" >> joueur
        echo "$viemax" >> joueur
        echo "$vie" >> joueur
        echo "$pointsarme" >> joueur
        echo "$(pwd)" >> joueur
        echo "-----------------------------------------"
    fi
}

function yatillasortie #Recherche de sortie
{
    if [ $finpartie = 1 ]
    then
        ligne=$(ls | grep 'sortie')
        echo $ligne
        ligne=${#ligne}
        if [ $ligne -gt 1 ]
        then
                echo "OMG ! Mais serait-ce la lumière du jour que je vois ? Mais
oui ! Voilà la sortie !"
                echo "Vous avez réussi à trouver la sortie ! Bien joué !"
                finpartie=0
        fi
    fi
}

menu
```