Juego BASH

DOCUMENTACIÓN

Pelea de Terminal

Salvador Jiménez Fernandez 1ª DAW

Juego BASH

PRESENTACIÓN

Este programa es un proyecto basado en hacer un juego interactivo en Bash, en mi caso yo elegí hacer un juego basado en la idea de los RPG por turnos.

Las bases son simples, es un combate, tu como jugador tienes que elegir una acción y luego el enemigo, el cual actúa de forma aleatoria.

El objetivo es bajar la vida del enemigo a 0 sin que la tuya llegue a 0.

INICIO

Lo primero que hace el programa es asignar una variables.

Para el jugador:

```
personaje_salud=100
```
* La salud base del jugador *

```
personaje_mana=50
```
* El mana base del jugador *

```
personaje_ataque=15
```
* El ataque base del jugador *

Para el enemigo:

```
enemigo_salud=200
```
* La salud base del enemigo *

```
enemigo_ataque=10
```
* El ataque base del enemigo *

```
acto_enemigo=4
```
* Esta variable sera para que el enemigo actué de forma aleatoria *

Funciones de aleatoriedad:

```
rango_ataque(){echo $((RANDOM % $1 + 5))}
```

* Coge el valor del ataque y usa ese valor como maximo, luego elige un numero entre 0 y ese valor y le suma 5 al resultado
EJ: ATQ 15 → RANDOMresultado = 7 → 7+5 = 12 → El resultado es 12 en esa accion

```
rango_enemigo(){echo $((RANDOM % $1))}
```

* Se elige un numero aleatorio entre del rango que es en este caso 4 y según el resultado tendremos una tabla con lo que realizara el enemigo *

Salvador Jiménez Fernandez 1ª DAW

Juego BASH

EJECUCIÓN

El jugador sera bienvenido con una breve explicación y se le pedirá que introduzca su nombre por consola.

Captura

El jugador introducirá su nombre, este sera registrada en la variable `$jugador`.

Tras esto se entra en un bucle en el cual el usuario tendera que elegir su clase, la cual le dará una ventaja en el juego

Juego BASH

BUCLE 1: CLASE

El usuario vera un mensaje por pantalla y se le dará a elegir entre 3 opciones, la elección del usuario se guardara dentro de la variable `$clase,`

Si el usuario introduce cualquier otra cosa que no sea una de las opciones saltara un error y volverá al comienzo del bucle.

Tras elegir una se le dará una breve descripción de esta y se le pedirá que confirme usando la letra S, el programa reconoce tanto la *S* mayúscula como la *s* minúscula, introducir otra cosa hace que vuelva al comienzo del bucle y aceptar una clase que salga de este.

EFECTO DE LAS CLASES

Las clases modifican las variables del jugador que se declararon al comienzo:

Guerrero : `personaje_salud=200`

Mago : `personaje_mana=100`

Pícaro : `personaje_salud=75 y personaje_ataque=$((personaje_ataque*2))`

Captura

```
¡Elige que clase quieres ser!

Elige una para saber mas de ella

(1) Guerrero
(2) Mago
(3) Picaro
 3


El Picaro tiene un daño muy alto pero tiene menos vida

Eliges Picaro? S/N
```

Juego BASH

BUCLE 2 : DIFICULTAD

El siguiente bucle modifica la dificultad del juego.

Al usuario se le ofrecerá 3 opciones y que introduzca por terminal su elección, si lo introducido no esta dentro de los valores volverá al comienzo de este bucle.

La elección del usuario se guardara dentro de la variable `$dificultad`

EFECTO DE LAS DIFICULTADES

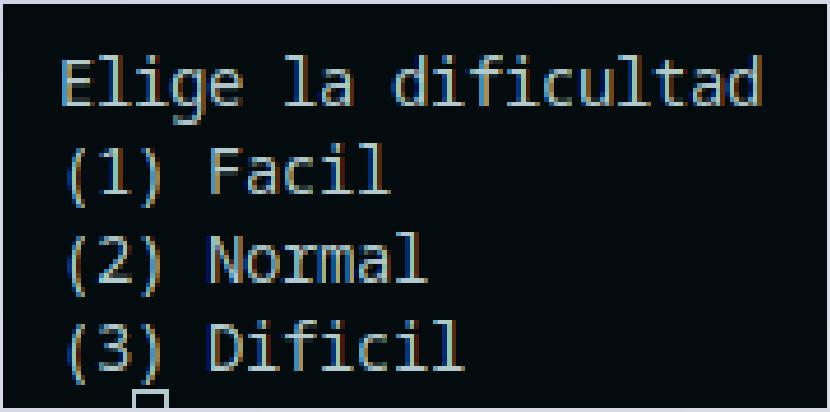
Las dificultades modifican las variables del enemigo que se declararon al comienzo:

Fácil : `enemigo_salud=$((enemigo_salud/2))` y `enemigo_ataque=$((enemigo_ataque/2))`.

Normal : Se mantiene los valores.

Difícil : `enemigo_salud=$((enemigo_salud*2))` y `enemigo_ataque=$((enemigo_ataque*2))`

CAPTURA



Juego BASH

MENSAJE DE BIENVENIDA

Primero se aplicara un ‘clear’ para limpiar toda la información anterior.

Antes de entrar en el ultimo bucle, asignaremos un par de variables mas:

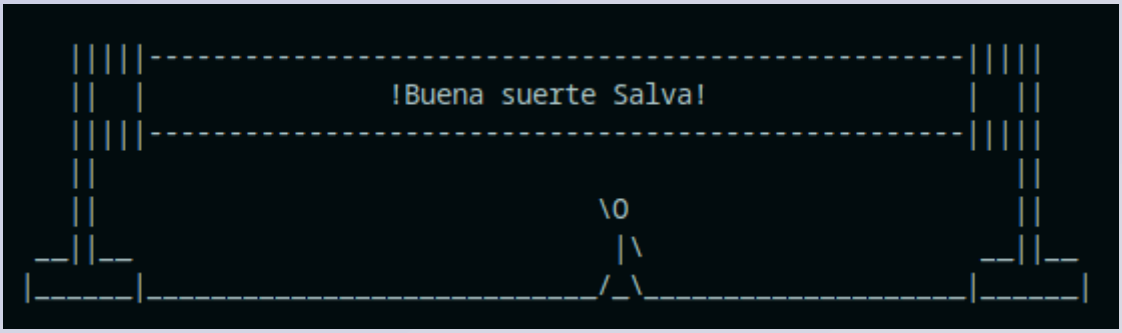
```
mensaje="!Buena suerte $jugador!"
ocupado=$((((51-${#mensaje})/2) + ${#mensaje}))
espacios=" "
```

Estas variables nos servirán para hacer un mensaje de bienvenida al usuario, usaremos un gráfico parecido a un panel y el centro aparecerá el un mensaje con el nombre del usuario centrado en este, la formula es la siguiente:

```
printf " || |%$((ocupado))s%$((51-ocupado))s| ||\n" "$mensaje" "$espacios"
```

Y el resultado se vería algo así

CAPTURA



Después asignaremos 3 variables mas que nos servirán para guiar al jugador para que sepa cual son sus valores máximos en cada campo.

```
readonly vida_fija=$personaje_salud
readonly vida_fija_enemigo=$enemigo_salud
readonly mana_fijo=$personaje_mana
```

La linea ‘readonly’ se asegurara que su valor no cambie.

Juego BASH

BUCLE 3 : JUEGO

Ahora entraremos en el ultimo bucle, el juego principal, lo primero que hace es asignar unas variables para crear la interfaz gráfica.

```
jugador_info=       "  $jugador $personaje_salud/$vida_fija       Enemigo $enemigo_salud/$vida_fija_enemigo "
mana_info=          " Mana $personaje_mana/$mana_fijo "
```

Ahora crearemos el marco donde se vera la informacion.

```
printf '%.0s-' {1..50}          * Para crear el marco superior *
echo " "
```

```
printf "%*s\n" $(((${#jugador_info}+50)/2)) "$jugador_info"
```

```
printf "%*s\n" $(((${#mana_info}+50)/2)) "$mana_info"
```

* Mostrara la información de la variable centrada *

```
printf '%.0s-' {1..50}          * Cierra el marco *
```

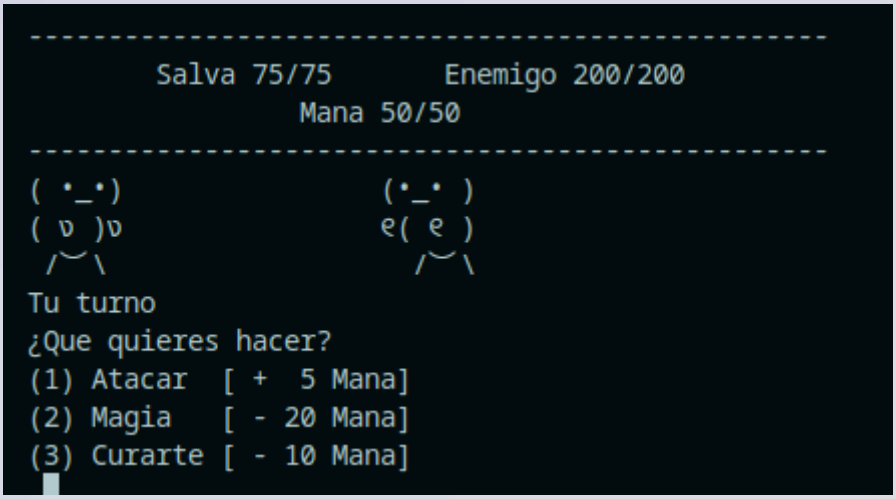
```
echo " "
echo "( •_•) (•_• )"
echo "( ง )ง ɐ( ɐ )"
echo ' /‾‾\ /‾‾\'
```

* Decoración para representar los personajes *

```
echo "Tu turno"
echo "¿Que quieres hacer?"
echo "(1) Atacar [ + 5 Mana]"
echo "(2) Magia [ - 20 Mana]"
echo "(3) Curarte [ - 10 Mana]"
```

* El menú de acción *

CAPTURA



Juego BASH

ACCIONES

El usuario deberá introducir por pantalla la acción que elija en caso de no estar dentro de las opciones posibles saltara lo siguiente:

```
echo " "
echo "Accion invalida"
echo " "
continue
```

Saldrá ese mensaje por pantalla y volverá al comienzo del bucle gracias al 'continue'

Siempre que el usuario introduzca cualquier dato se pasara un 'clear' para mantener la pantalla limpia.

Ahora las acciones como jugador validas:

(1) ATACAR

El usuario elige atacar, se mostrara un un dibujo por ASCII de la accion y un mensaje de texto con el daño realizado.

Para calcular el daño que hacemos usamos una variable nueva:

```
ataque_personaje=$(rango_ataque $personaje_ataque)
```

Se usa la variables que declaramos antes para crear el numero del daño

Este numero se le resta al enemigo usando la siguiente variables

```
enemigo_salud=$((enemigo_salud - ataque_personaje))
```

Usamos un 'IF' aquí, si la vida del enemigo baja de 0 se rompera el bucle y saldra un mensaje de victoria y acabara el juego.

```
if [[ $enemigo_salud -le 0 ]]; then
# Ganas
echo "¡Ganaste!"
echo " "
echo "ᕦ( ᐛ )ᕤ"
echo " "
break
```

También la acción ataque recupera el Mana de usuario en +5, hay un 'IF' para comprobar que si el Mana pasa del máximo en base a su clase no sobrepase del máximo de clase.

```
personaje_mana=$((personaje_mana + 5))
if [[ $clase == "2" && personaje_mana -gt 75 ]]; then
personaje_mana=75
elif [[ personaje_mana -gt 50 ]]; then
personaje_mana=50
else
personaje_mana=$personaje_mana
fi
```

Juego BASH

(2) MAGIA

Magia tiene la misma funcionalidad que ATACAR, pero con unas diferencias.

El usuario gastara Mana para usar esta acción (20), en caso de que este sea insuficiente volverá al comienzo del bucle otra vez

```
echo "Lanzas un hechizo"
personaje_mana=$((personaje_mana-20))
if [[ personaje_mana -lt 0 ]]; then
personaje_mana=$((personaje_mana+20))
echo "No tienes mana suficiente"
continue
fi
```

Al igual que ATACAR usaremos la variable `$ataque_personaje` pero con la diferencia que ahora crearemos otra variable mas para calcular el daño.

```
magia=$((ataque_personaje * 3))
```

El resto de la función es exactamente igual que ATACAR.

(3) CURARTE

Esta acción permite al usuario recuperar salud, al comienzo consumirá Mana (10) y si falla mostrara el mensaje de error igual que en la acción Magia.

La cantidad de salud se calcula de la siguiente forma:

```
cura_personaje=$(rango_ataque $personaje_ataque)
cura=$(( cura_personaje * 2 ))
personaje_salud=$((personaje_salud + cura))
```

Para el caso que la curación sobrepase el máximo de la clase tenemos la siguiente función:

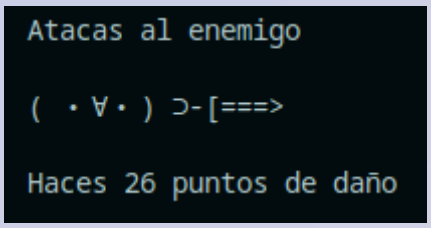
```
if [[ $clase == "1" && personaje_salud -gt 150 ]]; then
personaje_salud=150
elif [[ $clase == "3" && personaje_salud -gt 75 ]]; then
personaje_salud=75
elif [[ personaje_salud -gt 100 ]]; then
personaje_salud=100
else
personaje_salud=$personaje_salud
fi
```

Con esto ya tendríamos las acciones que puede realizar el usuario formadas.

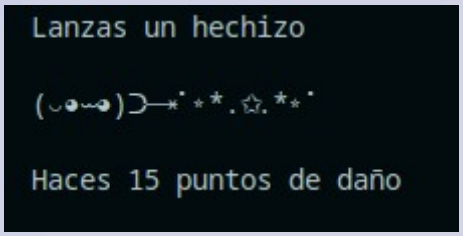
Juego BASH

CAPTURA

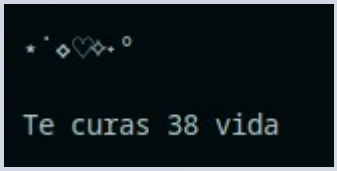
(1) ATACAR



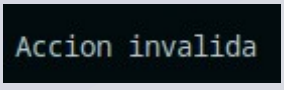
(2) MAGIA



(3) CURARSE



ERROR



Salvador Jiménez Fernandez 1ª DAW

Juego BASH

ENEMIGO

Con la variable que declaramos al comienzo `acto_enemigo=4 y rango_enemigo(){echo $((RANDOM % $1))}`

permitirá que el enemigo actué de una forma especifica según el valor que salga.

Las acciones del enemigo ocurren siempre que el usuario introduce una opción valida y son las siguientes

(1) ATACAR

Si el resultado es (1) el enemigo ataca de forma normal, la forma de que funciona es exactamente igual que el atacar del usuario.

Las diferencias son que se usan obviamente los valores del enemigo y el resultado se resta a la variable asignada a la salud del usuario `$personaje_salud`

Si la salud del usuario llega a (0) se le considerara una derrota y ocurrira lo siguiente.

```
if [[ $personaje_salud -le 0 ]]; then
# Pierdes
echo "¡Perdiste!"
break
fi
```

Y el bucle se cierra acabando el juego.

(2) CRITICO

Si el resultado es (2) el enemigo realiza un ataque criticio, su funcion es igual que la de ataque pero se crea una variable para calcular el critico.

```
ataque_enemigo=$(rango_ataque $enemigo_ataque)
critico=$(( ataque_enemigo * 2 ))
```

El resto es idéntico a la acción de ATACAR pero usando critico para restar los valores

(3) CURA

Si el resultado es (3) el enemigo se cura, funciona exactamente igual que la del jugador pero cambiado las variables de los valores base, también tenemos un 'IF' que comprueba si la cura se pasa de la salud del enemigo teniendo en cuenta la dificultad de la partida.

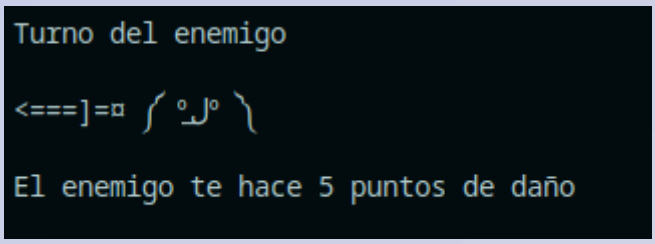
```
if [[ $dificultad == "1" && enemigo_salud -gt 100 ]]; then
enemigo_salud=100
elif [[ $dificultad == "2" && enemigo_salud -gt 200 ]]; then
enemigo_salud=200
elif [[ $dificultad == "3" && enemigo_salud -gt 400 ]]; then
enemigo_salud=400
else
enemigo_salud=$enemigo_salud
fi
```

(4) NADA = El enemigo no actúa, pierde el turno

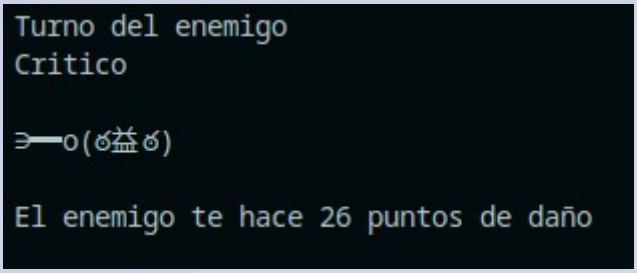
Juego BASH

CAPTURA

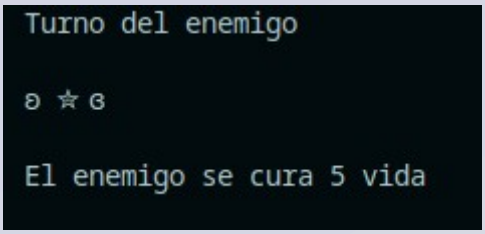
(1) ATACAR



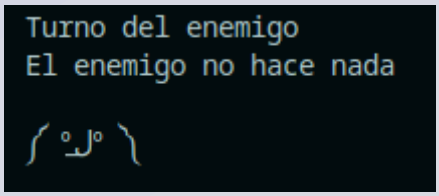
(2) CRITICO



(3) CURA



(4) NADA



Juego BASH

CONCLUSIÓN

Aquí acaba este proyecto de Bash, al principio trate de hacerlo sencillo, pero cada vez que podía ver era posible introducir mas opciones y acabo siendo mas complicado de lo que parecía.

Quizá el mayor reto fue el poder cuadrar los gráficos, Bash es bastante especial con ese campo.

Y también la cantidad de pruebas para probar si explotaba claro esta.

Me he pasado bastante bien haciendo este proyecto y me gustaría hacer algo similar, si es posible con otro lenguaje que sea mas potente, pues si es verdad que bash es rápido y ligero es un lenguaje muy limitado.

Pero eso lo dejare para un futuro quizás.

Hasta aquí mi conclusión sobre el proyecto

¡Muchas Gracias por leer este dossier!

¡ HASTA LUEGO !

Salvador Jiménez Fernandez 1ª DAW