Modificar Tomb Raider 1 al 5 con OpenTomb

fanta <fanta@56k.es>

Sumario

Compilar Opentomb

Hola amigo/a/e/i/o/u. Buenos días.

1. Lo primero que has de saber es que yo **no soy un especialista en opentomb** por escribir sobre opentomb. Simplemente me he pegado un poquito con esto y quería compartirlo.

2. Lo segundo que has de saber es que opentomb es un **proyecto de software libre** que mola bastante porque te permite **poder jugar a los Tomb Raiders del 1 al 5.**

3. Lo tercero que has de saber es que lo que más mola de opentomb es que **podrás trastear bastante** con los juegos porque **se extiende con scripts en lenguaje LUA .**

Compilar el binario Opentomb desde diferentes distros

La compilación sobre una **Debian/Ubuntu/**… podría ser así:

```
# apt install cmake gcc g++ zlibc libopenal-dev libpthread-stubs0-dev libboost-thread-dev libsdl2-dev libglu1-mesa-dev zlib1g-dev libpng-dev ffmpeg git lua freetype2
$ git clone https://github.com/opentomb/OpenTomb
$ cd OpenTomb/
$ mkdir build
$ cd build
$ cmake ..
$ make
```

La compilación sobre **Arch Linux** podría ser así:

```
$ sudo pacman -S gcc zlib openal ffmpeg libpng boost-libs sdl2 glu git lua freetype2
$ yaourt -S libpthread-stubs
$ git clone https://github.com/opentomb/OpenTomb
$ cd OpenTomb/
$ mkdir build
$ cd build
$ cmake ..
$ make
```

Sea como sea **el resultado final tendría que ser un archivo con el nombre Opentomb .**

Ese archivo nos lo tendríamos que guardar en otro directorio. Ese archivo es el binario del motor del juego y no funcionará por si mismo si lo tratamos de ejecutar.

```
sh-5.1$ ls -la
total 3868
drwxr-xr-x  7 fanta fanta    4096 abr 17 01:15 .
drwxr-xr-x  6 fanta fanta    4096 abr 17 01:16 ..
-rw-rw-rw-  1 fanta fanta     607 abr 17 01:08 autoexec.lua
-rw-rw-rw-  1 fanta fanta    2239 abr 17 01:15 config.lua
drwxr-xr-x 10 fanta fanta    4096 abr 17 00:15 data
-rwxr-xr-x  1 fanta fanta 3921424 abr 17 00:04 OpenTomb
drwxrwxrwx  6 fanta fanta    4096 mar  3  2018 resource
drwxrwxrwx  2 fanta fanta    4096 feb  3  2018 save
drwxrwxrwx 13 fanta fanta    4096 feb 16  2018 scripts
drwxrwxrwx  2 fanta fanta    4096 feb 16  2018 shaders
sh-5.1$
```

Ese binario requiere de una estructuras de directorios y de determinados archivos en .lua para funcionar bien. Requiere que los archivos de los diferentes juegos estén en determinadas rutas.

Ejecutar y alterar Tomb Raider I

Vamos a ver como trastear con Tomb Raider 1 usando el engine Opentomb. Y para simplificar todo el proceso he dejado un tar.gz descargable que lleva todo listo para trastear.

Descargar todo el tinglado en un tar.gz

Descargar todo el tinglado se puede desde aquí: http://fanta.56k.es/games/1990/1996-Tomb_Raider_I/tomb_raider_1-opentomb-linux.tar.gz

Ocupa unos 188Mb comprimido. Lleva ya compilado Opentomb.

Desde línea de comandos podríamos descargar **TR1** y empezar a trastear siguiendo estos simples pasos:

```
$ mkdir -p ~/opentomb/tomb1
$ cd ~/opentomb/tomb1
$ wget -q "http://fanta.56k.es/games/1990/1996-Tomb_Raider_I/tomb_raider_1-opentomb-linux.tar.gz"
$ tar xfvz tomb_raider_1-opentomb-linux.tar.gz
$ mv tomb_raider_1/* .
$ rm -rf tomb_raider_*
$ ./OpenTomb
```

Esto ya nos genera una estructura de directorios en nuestro home preparada para desplegar otros tomb raiders (por ejemplo el 2, 3, 4 y 5). Pero por el momento **vamos a centrarnos en el Tomb Raider I y en manejar un poco Opentomb.**

Comprobar la versión de los niveles del juego

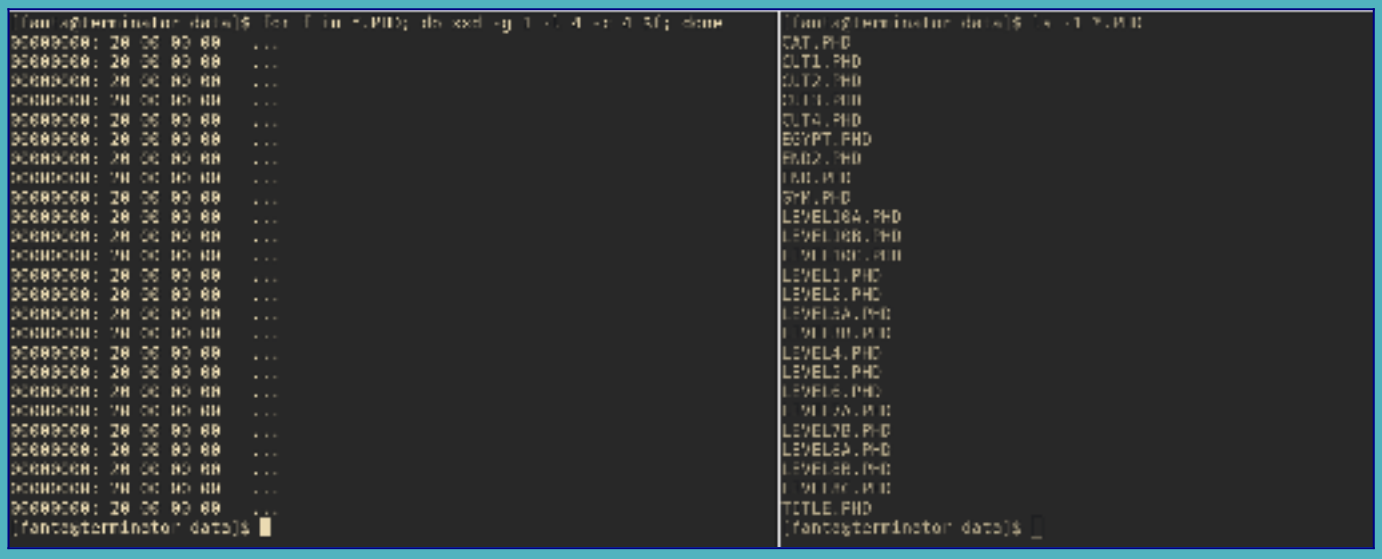
Los niveles del juego están en archivos con extensión .PHD, .TUB, .TR2 . **Los primeros 4 bytes de esos archivos indican la versión**. Esto se utiliza para que los editores de niveles o el propio motor del juego diferencie que tipo de nivel cargará. Leen esos 4 bytes y determinan si es un nivel de Tomb Raider 1, 2, 3, …

Ejemplos:

```
0x00000020 Tomb Raider 1, Gold, Unfinished Business
0x0000002d Tomb Raider 2
0xFF080038 Tomb Raider 3
0xFF180038 Tomb Raider 3
```

Desde la línea de comandos en linux vamos a poder **comprobarlo** así:

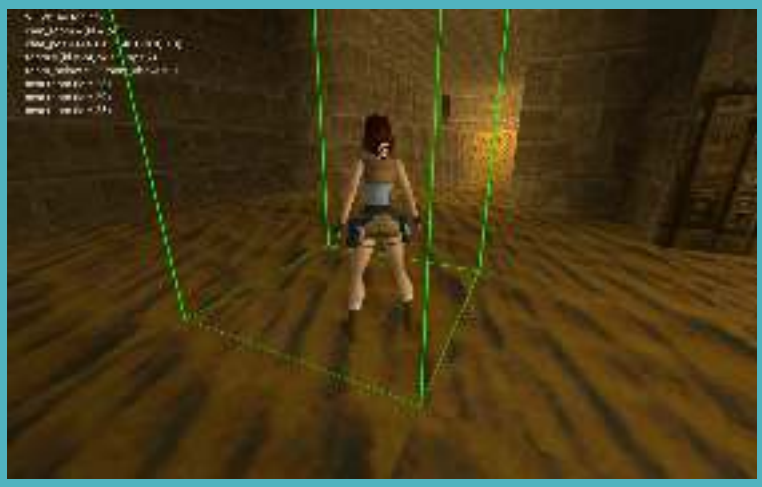
```
$ for f in *.PHD; do xxd -g 1 -l 4 -c 4 $f; done
```



20 00 00 00 que serían dados la vuelta 0x00000020

Mostrar información en pantalla

Con la tecla Y vamos a poder ir mostrando (cada vez que la pulsemos cambiará para mostrar otra información) variables del juego e incluso poder visualizar los limites del mapa.
En esta captura se puede ver como los mapas de tomb raider en realidad son cubos.



Abrir la consola

Yo básicamente uso la consola para salir del juego. Aunque se puede usar para muchas otras cosas como cambiar de nivel o incluso de juego y cargar un nivel del Tomb Raider 2 del tirón mientras estabas jugando al Tomb Raider 1.

La consola se abre con la tecla que suele estar encima del tabulador (ºª\) a la izquierda del 1.

Pero lo que mola es empezar a alterar el juego. Vamos ahora con eso.

Extensible mediante scripts en LUA

Lo que hace potente a OpenTomb es la posibilidad de no tener que re-compilar el engine cada vez que queramos alterar algo. Para alterar el propio motor del juego y todas las variables y funciones vamos a poder hacerlo mediante scripts en Lua.

Y esto es algo que mola mucho porque nos permite cambiar el juego completamente o adaptarlo a nuestro gusto.

Estos scripts se actualizan cada frame.

El archivo autoexec.lua

Vamos a tener 2 **autoexec.lua** . El que nos interesa es el que NO está dentro del directorio scripts.

Cuando ejecutamos sin parámetros OpenTomb busca ese archivo en el directorio donde está el binario OpenTomb de modo que como existe es ese el que usará. Si falla algo en ese script leerá el que está dentro de scripts.

Ejemplo de mi autoexec.lua ahora mismo:

```
-- LUA autoexec config file
-- CVAR's section. here you can create and delete CVAR's

if(cvars == nil) then
cvars = {};
end;


-- CVAR's section. here you can create and delete CVAR's
cvars.show_fps = 0;
cvars.free_look_speed = 2500;

-- AUTOEXEC LINES
setLanguage("english");


setGravity(0, 0, -5700.0);
mlook(0);
freelook(0);
--DrawCrosshair(0);
cam_distance(1024.0);
setCameraViewDistance(32768);
noclip(0);
playVideo(base_path .. "data/tr1/fmv/CORE.RPL");
loadMap(base_path .. "data/tr1/data/LEVEL1.PHD", 1, 0);

print("fanta!");
addItem(player, ITEM_LARGE_MEDIPACK, 9);
disableEntity(0x3A);

--DrawCrosshair(0);
```

Si nos fijamos en esa línea vemos que lleva -- delante de DrawCrosshair(0). En lua una forma de comentar líneas es añadiendo -- delante.

```
print("fanta!");

addItem(player, ITEM_LARGE_MEDIPACK, 9);
```

Imprimimos «fanta!» en la consola solo por saber en que sitio estamos. Luego lo que hago es añadir 9 items large medipack nada más cargar el juego. Si pulsamos ESC lo podemos comprobar y veremos que nada más comenzar el juego ya tenemos 10. **Son 10 porque he añadido 9 y ya se parte con 1.**

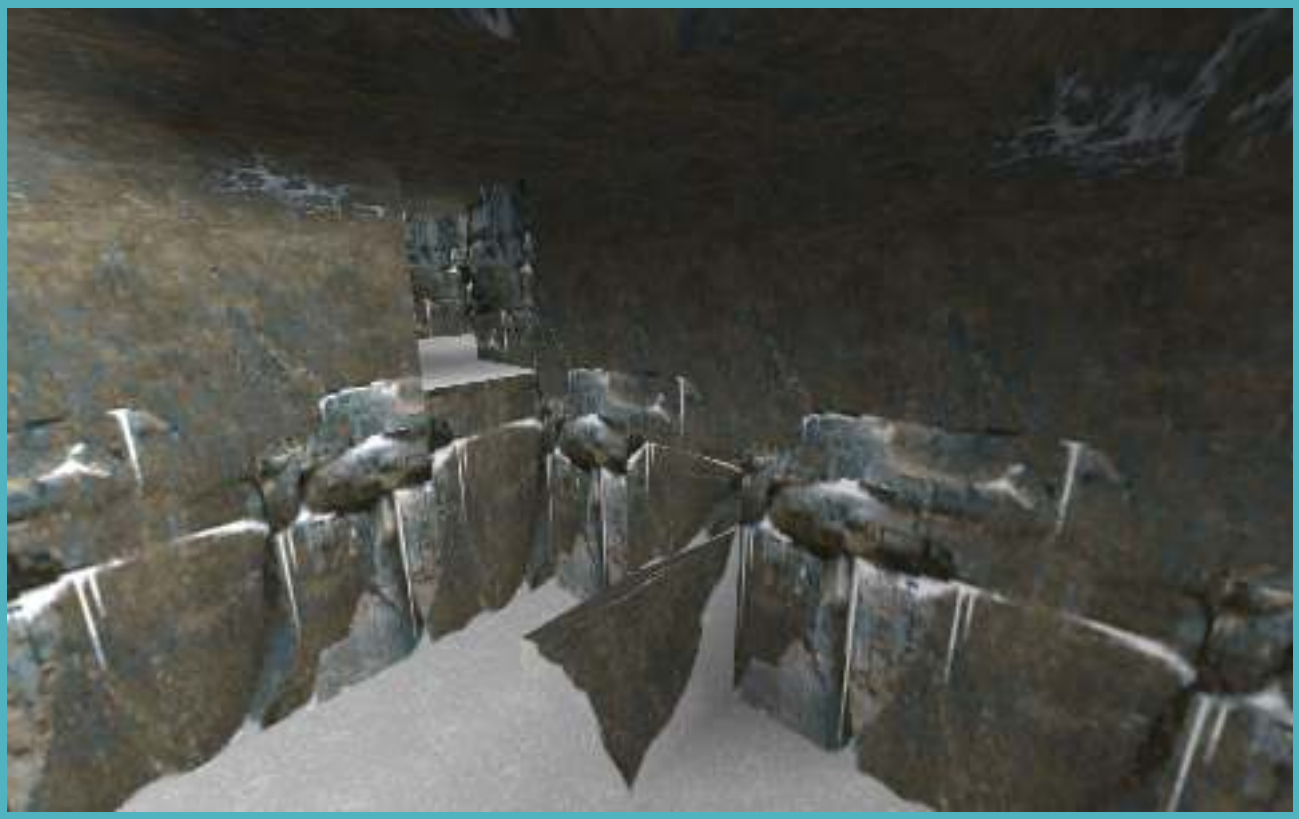

```
disableEntity(0x3A);
```

Algo que mola es deshabilitar una Entity. Por ejemplo yo he desactivado una que es un small medipack . Con la tecla Y he mostrado la información en pantalla de las entidades y eso me permite ver su ID (en este caso 0x3A).

No tiene mucho sentido desactivar algo que te da vida en el juego pero para probar es buena cosa.

Moviendo entidades en Tomb Raider I

El primer botiquín pequeño que puedes obtener en **Tomb Raider 1** está saltando en ese hueco que se ve en esta captura:



Esto supone tener que ir a esa zona y pegar un par de saltos. Luego ya arriba en el hueco se encuentra el botiquín.

Con OpenTomb **podemos mover el botiquín de las coordenadas originales** en las que se encuentra (x,y,z).

El botiquín en cuestión tiene la EntityID **0x3A**. Eso lo vemos si pulsamos la tecla Y varias veces hasta la vista **Room Objects**.

En nuestro **autoexec.lua** podemos añadir la siguiente línea para **ver en que posición se encuentra**:

```
print (getEntityPos(0x3A));
```

Eso mostrará en la consola de Opentomb algo así como esto:

```
90624.0
38400.0
-1024.0
```

Podemos **hacer que se posicione abajo para cogerlo sin tener que llegar a saltar**:

```
moveEntityGlobal(0x3A, -1000.0, -2000.0, -2050.0);
```

Ojo que **NO estamos indicando unas coordenadas.** Estamos indicando los valores de x,y,z que queremos que varíen de su posición original.

De esta forma podemos mover el botiquín a otra posición. Como la que se ve en estas capturas.

Posición original arriba en el hueco:

Posición alterada abajo del hueco para no tener que subir:

Digamos que ya no tenemos que saltar. Nos evitamos eso y podemos obtener el botiquín rápidamente.

Esto mismo se puede hacer con llaves por ejemplo. Si están en un sitio que costará trabajo y tiempo acceder pues simplemente lo movemos y lo tenemos más a mano.

Entonces haciendo esto pierde todo el encanto el juego

Calma un poco que te noto en tensión.

Estás usando un motor de un juego que permite alterar mediante scripting el propio juego. Eso en si mismo es ya un juego que puede ser muy divertido también.

Si ya jugaste a Tomb Raider 1 original ahora puedes mover esas entidades a diferentes lugares y no necesariamente a sitios de más fácil acceso.

OpenTomb mola porque **nos permite alterar mediante scripts los diferentes Tomb Raider** más allá de poder jugarlos. **Ese es el juego.**

Ejecutar y alterar Tomb Raider II

Vamos a ver como se puede disfrutar de Tomb Raider II con **Opentomb** en Linux y luego un par de alteraciones que se le pueden hacer al juego.

Vamos a darle duro pues.

Descargar e instalar el tinglado

Se puede hacer como se relata en los siguiente pasos:

```
$ mkdir -p ~/opentomb/tomb2
$ cd ~/opentomb/tomb2
$ wget -q "http://fanta.56k.es/games/1990/1997-Tomb_Raider_II/tomb_raider_2-opentomb-linux.tar.gz"
$ tar xfvz tomb_raider_2-opentomb-linux.tar.gz
$ mv tomb_raider_2/* .
$ rm -rf tomb_raider_*
$ ./OpenTomb
```

Cambiar el tamaño de por ejemplo una llave

Con la función **setEntityScaling** vamos a poder cambiar el tamaño en x,y,z de algunos objetos del juego.
En el caso de Tomb Raider II primer nivel (la muralla china) se puede cambiar el tamaño de la primera llave que encontraremos así:

```
print(getEntityScaling(0x11));
setEntityScaling(0x11, 3.0, 3.0, 3.0);
print(getEntityScaling(0x11));
```

Recordemos que eso se puede añadir sin problemas al autoexec.lua y se ejecutará al iniciar opentomb.

Un autoexec.lua con algunas funciones

Si se desea empezar a trastear lo mismo vienen bien estas **3 funciones (p2l, tohex y getPositionAllEntities)** que dejo en este autoexec.lua completo:

```
-- LUA autoexec config file
-- CVAR's section. here you can create and delete CVAR's

if(cvars == nil) then
    cvars = {};
end;

-- CVAR's section. here you can create and delete CVAR's
cvars.show_fps = 0;
cvars.free_look_speed = 2500;

-- FUNCTIONS
function p2l(id) -- position to log
    logPos = {getEntityPos(id)}
    print(id, logPos[1], logPos[2], logPos[3])
        local file = io.open( "temporalData.log", "a" );
        file:write(id..";"..logPos[1]..";"..logPos[2]..";"..logPos[3]);
        file:write("\n");
        file:close();
end

function tohex(num)
    local charset = {"0","1","2","3","4","5","6","7","8","9","a","b","c","d","e","f"}
    local tmp = {}
    repeat
        table.insert(tmp,1,charset[num%16+1])
        num = math.floor(num/16)
    until num==0
    return table.concat(tmp)
end

function getPositionAllEntities ()
        for valor = 1,254,1
        do
                print("0x"..tohex(valor)); -- print position on console
                p2l("0x"..tohex(valor)); -- write position into logfile
        end
end

-- AUTOEXEC LINES
setLanguage("english");
setGravity(0, 0, -5700.0);
mlook(0);
freelook(0);
cam_distance(1024.0);
setCameraViewDistance(32768);
noclip(0);
loadMap(base_path .. "data/tr2/data/WALL.TR2", 1, 0);

getPositionAllEntities();
```

Obtener el ID y posición de algunas entidades del nivel

Con esas funciones (en concreto con getPositionAllEntities) vamos a poder hacer cosas como estas:

```
getPositionAllEntities();
```

Nos dejará un archivo llamado **temporalData.log** con valores separados por «;» de las posiciones de las diferentes entidades.

```
fanta@terminator10:~/opentomb/tomb2$ cat temporalData.log
0x1;31232.0;33280.0;0.0
0x2;35328.0;30208.0;1536.0
0x3;49664.0;34304.0;0.0
0x4;54784.0;31232.0;1024.0
0x5;49664.0;33280.0;768.0
0x6;25088.0;36352.0;256.0
0x7;25088.0;36352.0;256.0
0x8;29184.0;31232.0;2304.0
0x9;26112.0;36352.0;256.0
0xa;26112.0;33280.0;1024.0
0xb;44544.0;43520.0;-2816.0
0xc;30208.0;41472.0;1280.0
0xd;28160.0;43520.0;1024.0
0xe;53760.0;35328.0;3072.0
0xf;25088.0;36352.0;3584.0
0x10;25088.0;33280.0;3584.0
0x11;45568.0;39424.0;-7040.0
0x12;62976.0;27136.0;-5504.0
0x13;64000.0;26112.0;-5376.0
0x14;62976.0;23040.0;-1536.0
fanta@terminator10:~/opentomb/tomb2$
```

Ejemplo al que he añadido **algo de información**:

Entity	x	y	z	Description
0x1	31232.0	33280.0	0.0	Cerradura llave 1
0x2	35328.0	30208.0	1536.0	
0x3	49664.0	34304.0	0.0	Puerta 1
0x4	54784.0	31232.0	1024.0	palanca para abrir la primera puerta. La primera palanca que encontramos tras el salto.
0x5	49664.0	33280.0	768.0	
0x6	25088.0	36352.0	256.0	
0x7	25088.0	36352.0	256.0	
0x8	29184.0	31232.0	2304.0	
0x9	26112.0	36352.0	256.0	
0xa	26112.0	33280.0	1024.0	
0xb	44544.0	43520.0	-2816.0	Tigre cerca agua llave
0xc	30208.0	41472.0	1280.0	
0xd	28160.0	43520.0	1024.0	
0xe	53760.0	35328.0	3072.0	puerta suelo que se abre sola cuando te pones encima.
0xf	25088.0	36352.0	3584.0	Llave 2
0x10	25088.0	33280.0	3584.0	
0x11	45568.0	39424.0	-7040.0	Llave 1 en el agua
0x12	62976.0	27136.0	5504.0	Tigre inicio
0x13	64000.0	26112.0	-5376.0	Baldosa posición inicial de lara
0x14	62976.0	23040.0	1536.0	

Si deseamos obtener solamente la posición de una entity en el archivo de log se puede hacer así:

```
p2l("0x11");
```

p2l (position to log) mandará al iniciar Opentomb la posición de 0x11 al log. Lo añadirá al final del archivo.

Ejecutar Tomb Raider III

Vamos a ver una forma de hacer funcionar Tomb Raider III mediante el motor Open Tomb.
La verdad es que para jugar al juego no es lo mejor pero si para trastear y alterarlo.

Descargar e instalar todo el tinglado

Vamos a ver como es posible **descargar y hacer funcionar Tomb Raider III** con el engine Open Tomb. El proceso es el siguiente:

```
$ mkdir -p ~/opentomb/tomb3
$ cd ~/opentomb/tomb3
$ wget -q "http://fanta.56k.es/games/1990/1998-Tomb_Raider_III/tomb_raider_3-
opentomb-linux.tar.gz"
$ tar xfvz tomb_raider_3-opentomb-linux.tar.gz
$ mv tomb_raider_3/* .
$ rm -rf tomb_raider_*
$ ./OpenTomb
```

El motor gráfico original utilizado para Tomb Raider III es distinto que el utilizado para las dos entregas anteriores, es más rápido y dinámico, las texturas están más elaboradas.

No obstante eso nos da un poco igual porque estamos usando el motor Opentomb.

Como puede comprobarse **este proceso es similar al que hemos utilizado con el Tomb raider 1 y 2** anteriormente.

Ejecutar Tomb Raider IV

La verdad es que no recuerdo pasarme Tomb Raider IV nunca. Si que lo probé en su momento pero no lo terminé.

Lo cierto es que con Opentomb **no voy a disfrutarlo a nivel de querer pasarme el juego**. La finalidad de tenerlo empaquetado y listo para disfrutar usando el motor Opentomb es meramente para trastear con el juego.

Descargar y ejecutar el tinglado

El proceso para **descargarlo y ejecutarlo por primera vez** es el siguiente:

```
$ mkdir -p ~/opentomb/tomb4
$ cd ~/opentomb/tomb4
$ wget -q "http://fanta.56k.es/games/1990/1999-Tomb_Raider_IV/tomb_raider_4-opentomb-linux.tar.gz"
$ tar xfvz tomb_raider_4-opentomb-linux.tar.gz
$ mv tomb_raider_4/* .
$ rm -rf tomb_raider_*
$ ./OpenTomb
```

Ejecutar Tomb Raider V

Por fin Tomb Raider 5 :)

Descargar y poner en marcha Tomb Raider 5

El proceso para **descargar Tomb 5 y ejecutarlo por primera vez** es el siguiente:

```
$ mkdir -p ~/opentomb/tomb5
$ cd ~/opentomb/tomb5
$ wget -q "http://fanta.56k.es/games/2000/2000-Tomb_Raider_V/tomb_raider_5-
opentomb-linux.tar.gz"
$ tar xfvz tomb_raider_5-opentomb-linux.tar.gz
$ mv tomb_raider_5/* .
$ rm -rf tomb_raider_*
$ ./OpenTomb
```

Este juego incorpora **nuevos movimientos** entre los que se destacan: andar por la cuerda floja, girar en la barra, saltar mientras se está agachado y otros movimientos novedosos.

Los juegos Gold

El motor **OpenTomb** solamente **tiene soporte para los siguientes Tomb Raiders**:

- Tomb Raider 1
- Tomb Raider 2
- Tomb Raider 3
- Tomb Raider 4
- Tomb Raider 5

Y puedes pensar que se trata de solamente 5 juegos. No es así. **Soporta otros que salieron como «extensiones»**.

- Tomb Raider 1 Gold
- Tomb Raider 2 Gold
- Tomb Raider 3 Gold

Esos 3 también están soportados. Por mi parte no tengo muchas ganas de empaquetarlos la verdad y **me he ceñido solamente a los juegos del 1 al 5 sin contar «los Gold».**
Si a ti te interesa empaquetar también los gold nada te impide que lo hagas.

Saltos más largos con Lara

Esto puede usarse en todos los Tomb Raiders (del 1 al 5). Y la verdad es que sirve un poco para llegar a esos sitios que normalmente no llegaríamos de un salto.

Esta modificación rompe el juego totalmente. Y se ve que está diseñado teniendo en cuenta la variable gravedad. En cuanto esto cambia zonas diseñadas para no ser accesibles nunca pasan a ser accesibles.

Es entonces cuando nos damos cuenta de que **vivimos en el show de Truman,** que todo es un decorado.

Cambiar la Gravedad en Tomb Raider

La gravedad por defecto en Tomb Raider es -5700. Si la cambiásemos a 0 al pegar un salto Lara estaría subiendo y subiendo hasta llegar al techo.
Ese techo de cristal está allí. Es transparente y normalmente inalcanzable. Con este sencillo hack **vamos a poder cambiar la gravedad asignando valores a varias teclas**.

La tecla C para restaurar a los valores por defecto de la gravedad.
La tecla V para cambiarla a -3000.
La tecla B para cambiarla a -2000.

Estos fragmentos de código se pueden añadir al archivo **scripts/system/debug.lua**

```
if(checkKey(KEY_C, true)) then
  setGravity(-5700);
end;
if(checkKey(KEY_V, true)) then
  setGravity(-3000);
end;
if(checkKey(KEY_B, true)) then
  setGravity(-2000);
end;
```

Se tendría que ver como cambiando de tecla se salta más.

Saludos cordiales.