

# Como crear un menú para tu juego

## Créditos

- » **Autor:** Hugo Ruscitti
- » **Fecha:** Diciembre del 2007

## Resumen

Construiremos un menú de opciones para un videojuego, debatiendo diferentes estrategias de diseño e implementación. Este menú le permitirá a los usuarios del videojuego seleccionar diferentes opciones como 'iniciar el juego', 'ver los creditos', 'terminar el programa' etc.

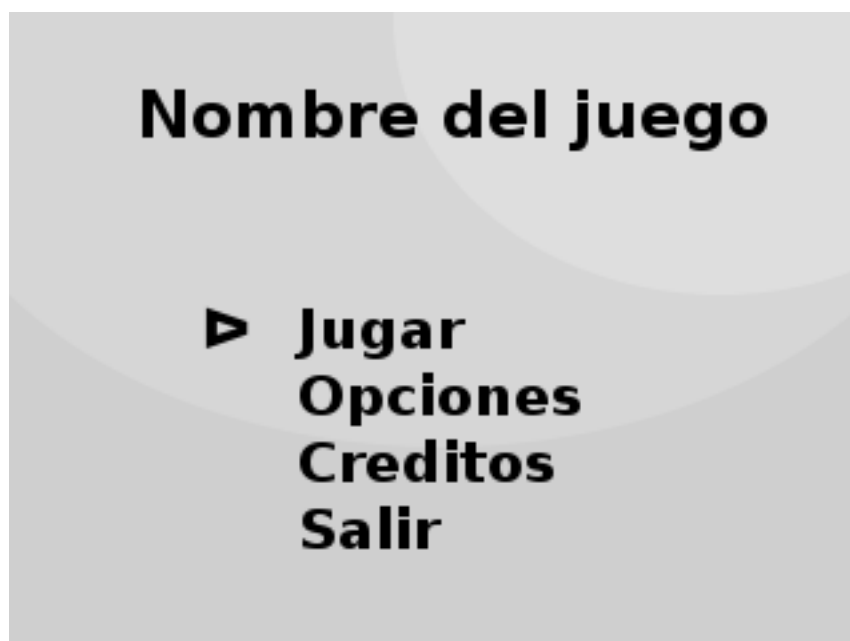
Utilizaremos el lenguaje de programación Python junto a la biblioteca multimedia "pygame" para proponer algunos ejemplos, que serán de utilidad para ilustrar diferentes ideas y metodologías.

Al final de este artículo encontrará un programa de ejemplo ilustrativo para descargar.

## ¿Por que brindarle importancia?

Muchas veces, cuando se desarrolla un videojuego, quienes programamos comenzamos a descuidar algunos aspectos básicos de nuestro proyecto; solemos asumir que la interacción entre nuestro programa y los usuarios no-desarrolladores es un asunto secundario, y puede esperar hasta que nuestro juego esté completamente terminado. Dejando de lado la implementación de gráficos, sonidos y los menús de principales.

Conforme el desarrollo de un juego avanza comenzamos a notar la necesidad de permitirle al usuario seleccionar el modo de juego, alterar los parámetros de configuración, e incluso mostrar información acerca de quienes desarrollaron el juego, como sus nombres o direcciones de correo:



Aunque parezca extraño al principio, sabemos tanto de un juego como alguien que desconoce por completo como programar. Mostrar tu juego a amigos y familiares forma parte de nuestro aprendizaje, y puedes adoptarlo como una actividad necesaria para evaluar el proceso de desarrollo y sus resultados.

Seguramente, cuando muestres tu juego a otras personas, lo deseable sería permitirles 'jugar' por cuenta propia, sin tener que explicar cada uno de los pasos a seguir. Implementar un menú principal ayudará a esta tarea: podrás darles el 'control' de juego a tus nuevos usuarios.

## Requisitos

Antes de comenzar a escribir nuestro menú, es importante destacar aquellas características que quisiéramos observar en él. Buscamos desarrollar un menú sencillo, que permita:

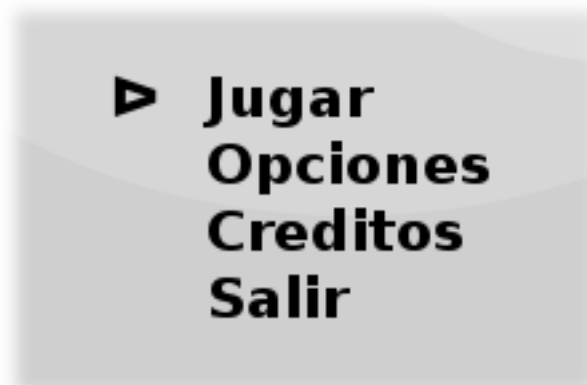
- » Mostrar opciones de texto, que sean fácilmente intercambiables.
- » Facilitar la inclusión del menú en otras escenas de un juego.
- » Navegar entre opciones utilizando los direccionales de teclado.
- » Asociar funcionalidad a cada una de las opciones.

Este paso de planificación es muy importante, porque nos permite organizar las tareas que debemos llevar a cabo en nuestro desarrollo y administrar mejor nuestro tiempo. De otra forma, podríamos dedicar mucho esfuerzo a crear características que luego no nos resultan demasiado útiles...

Como verá, no buscamos nada especial. Una vez construido el menú, podríamos añadirlo a nuestros primeros prototipos de juego sin problemas, y mejorar la experiencia de uso para cualquier otra persona ajena al desarrollo.

## Primer aproximación

Generalmente, un menú de videojuego muestra varias opciones en pantalla, y mediante el teclado podemos deslazar algún indicador, como una flecha o similar:



Posiblemente, en este instante, podríamos suponer que la solución mas simple consistiría en tener una variable que indique la opción actual, como el indicador que se observa en la imagen anterior. Luego, al pulsar una tecla como SELECT evaluaríamos el valor de esta variable y así reconoceríamos la opción pulsada:

```
if key == UP:
    opcion = opcion - 1
elif key == DOWN:
    opcion = opcion + 1

if key == SELECT:
    if opcion == 1:
        print "Elige: Jugar"
    elif opcion == 2:
        print "Elige: Opciones"
    elif opcion == 3:
        print "Elige: Creditos"
    elif opcion == 4:
        print "Elige: Salir"
```

El problema de esta primer estrategia es su flexibilidad: a medida que busquemos añadir opciones, o modificar las existentes, tendremos que lidiar con bloques de código muy grandes y vinculados, nuestro código estará concentrado en un solo lugar donde un cambio podría ocasionar errores de inconsistencia.

Tomemos como ejemplo el código anterior, imagine que deseamos añadir una opción nueva, como 'Reiniciar partida', justo debajo de la opción 'Jugar':

Tendríamos que cambiar todos los índices: 'Jugar' sería la opción 1, 'Reiniciar partida' la opción 2, 'Opciones' ahora cambiará para convertirse en la opción 3... Estos detalles pueden ocasionar algunos contratiempos, es fácil cometer errores en casos como ese.

Y cada vez que busquemos implementar un menú similar encontraremos las mismas dificultades.

Por suerte existen muchas otras formas de implementar funcionalidades como esta, y aquí veremos solo una:

### Otra aproximación: utilizando Objetos

Una forma interesante de construir nuestro menú de manera organizada consiste en distribuir los componentes del programa en diferentes entidades.

En este caso podemos representar el comportamiento completo del menú en una clase de nombre 'Menu'. Veamos su estructura general en python:

```
class Menu:

    def __init__(self, opciones):
        pass

    def actualizar(self):
        pass

    def imprimir(self, screen):
        pass
```

Mediante esta clase podemos representar el comportamiento básico de un menú. El método '\_\_init\_\_' será invocado al momento de crear un objeto de la clase Menu. Por ejemplo ante una sentencia como:

```
mi_menu = Menu(lista_opciones)
```

python generará una nueva instancia de objeto 'Menu'.

Como buscamos un menú que pueda ser utilizado una y otra vez, en diferentes escenas, podríamos separar a las opciones del mismo menú. El parámetro 'opciones' de la clase 'Menu' hace precisamente eso, consiste en una lista de opciones para representar.

Cada opción tiene asociado un texto y una acción. Por lo tanto representaremos las opciones del menú como tuplas: una de las formas mas sencillas relacionar valores:

```
opcion_nuevo_juego = ('Nuevo juego', funcion_para_iniciar_el_juego)
opcion_salir = ('Salir del programa', funcion_para_terminar)

lista_opciones = [opcion_nuevo_juego, opcion_salir]

menu = Menu(lista_opciones)
```

las tuplas son un tipo de dato que se incluyen en python y muchos otros lenguajes de programación. Tal vez no se encuentre muy familiarizado con ellas: es simple, una tupla permite agrupar dos o mas elementos, sin importar su tipo.

En este caso, 'opcion\_salir' es una referencia a una tupla que tiene como primer elemento a una cadena ('Salir del programa') y como segundo elemento tiene el nombre de una función.

De esta forma, crear un menú será tan simple como generar un objeto Menú junto a una lista de opciones; veamos un ejemplo mas completo:

```
opciones = [
    ('Jugar', funcion_para_iniciar_el_juego),
    ('Opciones', funcion_para_crear_menu_opciones),
    ('Creditos', funcion_para_mostrar_creadores_del_juego),
    ('Salir', funcion_para_terminar_el_programa)
]

menu = Menu(opciones)
```

Uno de los aspectos mas prácticos de esta aproximación consiste en algo denominado "encapsulación": La referencia 'menu' apunta a un objeto que almacena atributos y comportamiento, por lo tanto desde el exterior podemos 'comunicarnos' con este objeto utilizando sus métodos, por ejemplo "menu.imprimir(screen)". Y preocuparnos por su composición interna.

A partir de ahí, mediante la referencia 'menu', debemos actualizar e imprimir el menú periódicamente para que este gestione el comportamiento del teclado:

```
import pygame
```

```
# define el modo de video en un tamaño de
# 320 pixeles de ancho por 240 pixeles de alto.
screen = pygame.display.set_mode((320, 240))

while not salir:

    # atiende el evento que se produce al cerrar la ventana principal.
    for e in pygame.event.get():
        if e.type == QUIT:
            salir = True

    # limpia la pantalla.
    screen.fill((200, 200, 200))

    # actualización del menú
    menu.actualizar()
    menu.imprimir(screen)

    pygame.display.flip()
```

El código anterior es un extracto del bucle principal de un juego, en él se atiende la llegada de eventos, la actualización del menú y la impresión en pantalla.

Note que la 'verdadera' funcionalidad del programa está en el interior del objeto Menú, desde el bucle de juego solo llamamos a sus dos métodos 'actualizar' e 'imprimir'.

Veamos ahora como es el objeto Menú por dentro.

## Métodos de la clase Menu

### Método Actualizar

La tarea que debe realizar el método 'actualizar' es gestionar la pulsación de teclas y destacar la opción seleccionada. Para realizar esta tarea debemos conocer el estado del teclado y ejecutar funciones si se pulsa una tecla designada como 'Enter'.

Veamos un extracto de ese código:

```
Class Menu:

    def __init__(self, opciones):
        self.opciones = opciones
        self.seleccionado = False
        self.mantiene_pulsado = False

    def actualizar(self):
```

```

    """Altera el valor de 'self.seleccionado' con los direccionales."""

    tecla = pygame.key.get_pressed()

    if not self.mantiene_pulsado:
        if tecla[K_UP]:
            self.seleccionado -= 1
        elif tecla[K_DOWN]:
            self.seleccionado += 1
        elif tecla[K_RETURN]:

            # Invoca a la función asociada a la opción.
            titulo, funcion = self.opciones[self.seleccionado]
            print "Selecciona la opción: ", titulo
            funcion()

    # procura que el cursor esté entre las opciones permitidas
    if self.seleccionado < 0:
        self.seleccionado = 0
    elif self.seleccionado > self.total - 1:
        self.seleccionado = self.total - 1

    # indica si el usuario mantiene pulsada alguna tecla.
    self.mantiene_pulsado = tecla[K_UP] or tecla[K_DOWN] or tecla[K_RETURN]

    # [...] continúa con otros métodos

```

La variable de nombre 'tecla' representa el estado actual del teclado. Mediante esta variable consultamos el estado de las teclas 'subir' (K\_UP), 'bajar' (K\_DOWN) y 'seleccionar' (K\_RETURN).

Cuando se detecta la pulsación de la tecla 'arriba' o 'abajo' alteramos el valor de la variable 'self.seleccionado', que indica la opción seleccionada en ese momento.

En cambio, si se pulsa la tecla 'seleccionar', extraemos de la lista 'opciones' aquel elemento que ha sido seleccionado:

```

titulo, funcion = self.opciones[self.seleccionado]

```

como cada elemento de la lista 'self.opciones' es una tupla, extraemos el valor de la posición 'self.seleccionado' en dos variables. Para que la primera unifique con la cadena 'titulo' y la segunda variable quede asociada a la función a invocar.

Para comprender mejor esta rutina, recordemos como se almacenan las opciones del menú. Una opción estaba compuesta por una tupla:

```

tupla = ('Nuevo juego', funcion_para_iniciar_el_juego),

```

como tal, una 'tupla' representa un dato compuesto, que se puede descomponer con facilidad de la siguiente manera:

```
tupla = ('Nuevo juego', funcion_para_iniciar_el_juego),
titulo, funcion = tupla
```

luego, 'titulo' queda asociado a la primer componente de la tupla ('Nuevo juego') y 'funcion' queda asociado al nombre de la función que debemos invocar. Lo que sigue es la rutina es:

```
print "Selecciona la opción: ", titulo
funcion()
```

que imprime el titulo de la opción, y luego llama a la rutina apuntada por la variable 'funcion'.

## Método imprimir

Imprimir es una tarea mas sencilla, en el interior de la clase contamos con una lista de opciones y conocemos que opción se encuentra actualmente seleccionada. El objetivo de nuestro método será recorrer la lista de opciones e imprimir cada una de estas sobre el parámetro 'screen':

```
Class Menu:
    # [...]

    def imprimir(self, screen):
        """Imprime sobre 'screen' el texto de cada opción del menú."""

        total = self.total
        indice = 0
        altura_de_opcion = 30
        x = 200
        y = 300

        for (titulo, funcion) in self.opciones:

            # evalúa si debe destacar al opción
            if indice == self.seleccionado:
                color = (255, 255, 255)
            else:
                color = (230, 230, 230)

            imagen = self.font.render(titulo, 1, color)
            posicion = (x, y + altura_de_opcion * indice)
            indice += 1
            screen.blit(imagen, posicion)
```

el método recibe una referencia a cualquier superficie, donde imprimirá cada opción. La mayor parte de código está dedicado a posicionar cada texto en el lugar mas apropiado, buscando que la opciones aparezcan centradas en pantalla.

## Descarga

Puedes ver el código fuente de este artículo descargando el archivo menu.tar.gz. Ahí encontrarás dos tipos de menú para implementar en tus juegos.

## **Conclusión**

Existen muchas formas de resolver la creación de un menú para sus juegos, en este artículo he mostrado una forma que considero fácil de implementar y modificar. Anímese a incorporar todas las mejoras que guste, los usuarios de sus juegos lo agradecerán.

---

*Este documento ha sido generado automáticamente a partir del archivo 'menu.xml' el Sun Jan 25 09:34:32 2009  
La versión mas reciente de este documento se almacena en [www.losersjuegos.com.ar](http://www.losersjuegos.com.ar). Visitenos para obtener  
mas recursos y actualizaciones.*