Guía de conversión a la nueva versión

Realizado por: **Fernando Salamero**

En este documento se recoge de manera detallada los cambios que se han producido entre las dos versiones de pilasengine (de las 0.8x iniciales a las >=0.9x), con el fin de que proyectos desarrollados previamente puedan portarse para que funcionen correctamente con la versión más actualizada. Si tienes dificultades con ello… ¡aquí tienes la respuesta!

¡Comencemos!

Primero importar la librería correcta (ahora pasa a llamarse pilasengine):

Antes

```
import pilas
```

Después

```
import pilasengine
```

Crear el objeto pilas (desde él se accede a todas las características del engine).

Antes

```
pilas.iniciar()
```

Después

```
pilas = pilasengine.iniciar()
```

Las parámetros en los actores derivados se han de cargar nombrando el argumento.

Antes

```
pilas.actores.Actor("huevo.png")
```

Después

```
pilas.actores.Actor(imagen="huevo.png")
```

El módulo atajos desaparece, se accede directamente a su funcionalidad más usual.

Antes

```
pilas.atajos.ocultar_puntero_del_mouse()
pilas.atajos.mostrar_puntero_del_mouse()
pilas.atajos.modo_pantalla_completa()
pilas.atajos.modo_ventana()
```

Después

```
pilas.ocultar_puntero_del_mouse()
pilas.mostrar_puntero_del_mouse()
pilas.definir_pantalla_completa(True)
pilas.definir_pantalla_completa(False)
```

Los nuevos actores se derivan de pilasengine.actores.Actor, no es necesario usar el método __init__ (pasando su funcionalidad al método iniciar) y deben vincularse al objeto pilas.

Antes

```
class Spiderman(pilas.actores.Actor):
    def __init__(self, y=0):
        ….

prota = Spiderman(500)
```

Después

```
class Spiderman(pilasengine.actores.Actor):
    def iniciar(self, y=0):
        ...

pilas.actores.vincular(Spiderman)

prota = pilas.actores.Spiderman(y=500)
```

Los grupos de actores residen ahora en el módulo actores.

Antes

```
pilas.grupo.Grupo()
```

Después

```
pilas.actores.Grupo()
```

Las escenas se implementan de forma coherente con el sistema de vinculación y declaración que hemos visto en los actores.

Antes

```
class EscenaMenu(pilas.escena.Base):

    def __init__(self):
        pilas.escena.Base.__init__(self)
        ….

    def iniciar(self):
        pilas.fondos.Fondo("corazones.png")
        ….

pilas.cambiar_escena(EscenaMenu())
```

Después

```
class EscenaMenu(pilasengine.escenas.Escena):

    def iniciar(self):
        pilas.fondos.Fondo("corazones.png")
        ….


pilas.escenas.vincular(EscenaMenu)


pilas.escenas.EscenaMenu()
```

Las colisiones se pueden agregar directamente desde su módulo (aunque el método anterior sigue siendo posible también).

Antes

```
pilas.escena_actual().colisiones.agregar(sprite, sprite2, rebot
```

Después

```
pilas.colisiones.agregar(sprite, sprite2, rebotar)
```

Las tareas ahora están en un módulo que las agrupa a todas:

Antes

```
pilas.mundo.agregar_tarea(tiempo, funcion_a_llamar)
pilas.mundo.agregar_tarea_una_vez(tiempo, funcion_a_llamar)
pilas.mundo.agregar_tarea_siempre(tiempo, funcion_a_llamar)
pilas.mundo.agregar_tarea_condicional(tiempo, funcion_a_llamar)
```

Después

```
pilas.tareas.agregar(tiempo, funcion_a_llamar)
pilas.tareas.una_vez(tiempo, funcion_a_llamar)
pilas.tareas.siempre(tiempo, funcion_a_llamar)
pilas.tareas.condicional(tiempo, funcion_a_llamar)
```

Los controles personalizados no requieren que se les pase la escena

Antes

```
mandos = pilas.control.Control(pilas.escena_actual(), teclas)
```

Después

```
mandos = pilas.control.Control(teclas)
```

Los comportamientos (personalizados) se siguen usando en conjunción con el método hacer() pero no con una instancia de ellos, como se hacía antes, si no después de vincularlos (como los actores) y utilizando "etiquetas". Si se añaden varios comportamientos, se harán uno tras otro. Para finalizar uno de manera inmediata, éste debe devolver True.

Antes

```
class Esperando(pilas.comportamientos.Comportamiento):
    def __init__(self, receptor):
        ….


mi_actor.hacer(Esperando())
```

Después

```
class Esperando(pilasengine.comportamientos.Comportamiento):
    ….

pilas.comportamientos.vincular(Esperando)
mi_actor.hacer("Esperando")
```

Las habilidades (personalizadas) siguen de forma coherente el esquema de los comportamientos

Antes

```
class Cantar(pilas.habilidades.Habilidad):
    def __init__(self, receptor)
        ….

mi_actor.aprender(Cantar)
```

Después

```
class Cantar(pilasengine.habilidades.Habilidad):
    ….

pilas.habilidades.vincular(Cantar)
mi_actor.aprender("Cantar")
```

Los ángulos se miden ahora como es usual en matemáticas; desde el eje positivo de las x. Así que un ángulo de 0º indica "hacia la derecha" (lo que antes era -90º)

Antes

```
#En el archivo inicial

import pilas
pilas.iniciar()
…

#En otro archivo diferente

import pilas

class EscenaMenu(pilas.escena.Base):

    def __init__(self):
        pilas.escena.Base.__init__(self)
        ….

    def iniciar(self):
        pilas.fondos.Fondo("corazones.png")
```

Después

```
#En el archivo inicial

import pilasengine
pilas = pilasengine.iniciar()
…

#En otro archivo diferente

import pilasengine

class EscenaMenu(pilasengine.escenas.Escena):

    def iniciar(self):
        self.pilas.fondos.Fondo("corazones.png")
        ….
```

Eso es todo, el resto de las llamadas serán idénticas.