Introducción a Python y

Andrés Pastorini - Alvaro Martínez

TDIVJ2D – Tecnólogo Informático

Introducción a Python y PyGame

Objetivos

- Brindar una introducción al lenguaje Python.
- Brindar una introducción a la libreria Pygame.

Introducción a Python y PyGame

TEMARIO

1. **Python**
2. Pygame

Introducción a Python y PyGame

Es un lenguaje de programación de propósito general con las siguientes características:

- Interpretado
- Portable
- Tipos Dinámicos
- Tipos Fuertes
- Multiplataforma
- Multiparadigma

Introducción a Python y PyGame

Tipos Básicos

- Números (enteros, como flotante, complejos)
- Cadenas
- Booleanos(True, False)

```
# esto es una cadena
c = "Hola Mundo"

# y esto es un entero
e = 23
```

```
# type(entero) devolvería int
entero = 23

# type(entero) devolvería long
entero = 23L
```

Obs.: No se indica el tipo de la variable al definirla.

Introducción a Python y PyGame

Tipos Básicos

```
# esto es una cadena
c = "Hola Mundo"

# y esto es un entero
e = 23

# podemos comprobarlo con la función type
type(c)
type(e)
```

Introducción a Python y PyGame

Operadores

Operador	Descripción	Ejemplo
+	Suma	r = 3 + 2 # r es 5
-	Resta	r = 4 – 7 # r es -3
-	Negación	r = -7 # r es -7
*	Multiplicación	r = 2 * 6 # r es 12
**	Exponente	r = 2 ** 6 # r es 64
/	División	r = 3.5 / 2 # r es 1.75
//	División entera	r = 3.5 // 2 # r es 1.0
%	Módulo	r = 7 % 2 # r es 1

Obs.: Cuando en una operación aparecen distintos tipos, se convierten todos los tipos al tipo más complejo que aparece en la expresión.

Introducción a Python y PyGame

Booleanos

Una variable de tipo Boolean puede tomar los valores True o False.

Operador	Descripción	Ejemplo
and	¿se cumple a y b?	`r = True and False # r es False`
or	¿se cumple a o b?	`r = True or False  # r es True`
not	No a	`r = not True  # r es False`

Introducción a Python y PyGame

Booleanos

Operador	Descripción	Ejemplo
==	¿son iguales a y b?	`r = 5 == 3 # r es False`
!=	¿son distintos a y b?	`r = 5 != 3 # r es True`
<	¿es a menor que b?	`r = 5 < 3  # r es False`
>	¿es a mayor que b?	`r = 5 > 3  # r es True`
<=	¿es a menor o igual que b?	`r = 5 <= 5 # r es True`
>=	¿es a mayor o igual que b?	`r = 5 >= 3 # r es True`

Introducción a Python y PyGame

Colecciones

- Listas
- Tuplas
- Diccionarios

Introducción a Python y PyGame

Colecciones

La **lista** es un tipo de colección ordenada. Equivalente a lo que en otros lenguajes se conoce como arreglos. Las listas pueden contener cualquier tipo de dato(números, cadenas, booleanos, etc.).

```
l = [22, True, "una lista", [1, 2]]
l = [11, False]
mi_var = l[0] # mi_var vale 11
mi_var = l[1][0] # mi_var vale 1
l[0] = 99 # Con esto l valdrá [99, True]
```

Obs.: Los elementos de una lista no tienen porque tener el mismo tipo.

Colecciones

Se puede utilizar un número negativo como índice, donde se empieza a contar desde el final hacia la izquierda.
Con [-1] se accede al último elemento de la lista por ejemplo.
Se puede utilizar el **slicing** o particionado *que consiste en ampliar este mecanismo para permitir seleccionar porciones de la lista.*

l(inicio:fin)	**l = [99, True, “una lista”, [1, 2]]**
l(inicio:fin:salto)	**l[0:2] = [0, 1] # l vale [0, 1, “una lista”, [1, 2]]**

Tuplas

Es un tipo de dato inmutable, es decir no se pueden modificar una vez creados, por lo tanto tienen un tamaño fijo.
Distintas posiciones pueden tener distintos tipos de datos.

t = (1, 2, True, “python”)
x =(1,) # tupla de un elemento

El constructor de la tupla es la coma.

Tuplas

Para referirnos a elementos de una tupla, como en una lista, se usa el operador []:

```
mi_var = t[0] # mi_var es 1
mi_var = t[0:2] # mi_var es (1, 2)
```

Introducción a Python y PyGame

Diccionarios

Los diccionarios, son llamados también matrices asociativas, son colecciones que relacionan una clave y un valor.

d = {“Love Actually “: “Richard Curtis”,
“Kill Bill”: “Tarantino”}

Como clave se puede utilizar cualquier valor inmutable.

Introducción a Python y PyGame

Consideraciones

Los **bloques** en **Python** no se indican con corchetes ({ }) como en otros lenguajes, sino por espacios(indentación).

Introducción a Python y PyGame

Control de Flujo

Sentencias condicionales

```
if condición :

var = val1 if (cond) else val2

if condición:
else:

if condición:
[elif  codnición:]*
[else:]
```

Bucles

```
while cond:
        sent1
        sent2


For  elemento in secuencia
       sent1
      sent2
```

Funciones

Una función es un fragmento de código con un nombre asociado que realiza una serie de tareas y devuelve un valor. En Python no existen los procedimientos, cuando no sed especifica un valor de retorno la función devuelve el valor None.

En Python las funciones se declaran de la siguiente forma:

```
def mi_funcion(param1, param2):
    print param1
    print param2
```

Funciones

Los valores mutables se pasan por referencia, y los inmutables por valor.

Se pueden definir valores por defecto para los parámetros, agregando un signo de igual y el valor del parámetro.

```
def mi_funcion(param1, param2=valor):
     sent1…
```

Funciones

Se pueden definir funciones con un número variable de argumentos, colocando un último parámetro para la función cuyo nombre debe precederse de un signo *, el tipo del parámetro con * es una tupla.

```
def varios(param1, param2, *otros):
   for val in otros:
```

Funciones

También se puede preceder el nombre del último parámetro con **, que será de tipo diccionario. Las claves de este diccionario serían los nombres de los parámetros indicados al llamar a la función y los valores del diccionario, los valores asociados a estos parámetros.

```
def varios(param1, param2, **otros):
    for i in otros.items():
        print i
```

Invocación: `varios(1, 2, tercero = 3)`

Funciones

Se utiliza la palabra **return** para devolver valores, y se puede devolver más de un valor.

```
def f(x, y):
    return x * 2, y * 2

a, b = f(1, 2)
```

```
def f(x, y):
    return x * 2, y * 2

a, b = f(1, 2)
```

Introducción a Python y PyGame

Orientación a objetos

En Python las clases se definen mediante la palabra clave **class** seguida del nombre de la clase, dos puntos (:) y a continuación, indentado, el cuerpo de la clase.

```
#clase de estilo nuevo python
class B(object):
   pass
        o
#clase tradicional
class B():
   pass

#variable de tipo B
b=B()
```

Introducción a Python y PyGame

Orientación a objetos

```
class Coche:
    """Abstraccion de los objetos coche."""
    def __init__(self, gasolina):
        self.gasolina = gasolina
        print "Tenemos", gasolina, "litros"

    def arrancar(self):
        if self.gasolina > 0:
            print "Arranca"
        else:
            print "No arranca"

    def conducir(self):
        if self.gasolina > 0:
            self.gasolina -= 1
            print "Quedan", self.gasolina, "litros"
        else:
            print "No se mueve"
```

Orientación a objetos

El método __init__, se ejecuta justo después de crear un nuevo objeto a partir de la clase.

El primer parámetro de todos los métodos de la clase es siempre **self** que sirve para referirse al objeto actual. Éste mecanismo es necesario para poder acceder a los atributos y métodos del objeto diferenciando.

Introducción a Python y PyGame

Orientación a objetos

Para indicar que una clase hereda de otra se coloca el nombre de la clase de la que se hereda entre paréntesis después del nombre de la clase.

```
class A:
 def  __init__(self,param1):
     self.att1=param1
 def f():
     sent1

class B(A):
     pass
```

Orientación a objetos

Si no se indica método **__init__** se tomara el de la clase base de forma explicita.
Para invocar algún método de la clase base se hace ClaseBase.metodo(…).

Si la clase no define métodos ni atributos, sintácticamente se necesita agregar la palabra clave **pass.**

Introducción a Python y PyGame

Orientación a objetos

En Python, se permite la herencia múltiple.

```
class C(Class1,Class2):
    pass
```

En el caso de que alguna de las clases padre tuvieran métodos con el mismo nombre y número de parámetros, las clases sobrescribirían la implementación de los métodos de las clases más a su derecha en la definición.

Orientación a objetos

La encapsulación se refiere a impedir el acceso a determinados métodos y atributos de los objetos estableciendo así qué puede utilizarse desde fuera de la clase.
En Python no existen los modificadores de acceso, y lo que se suele hacer es que el acceso a una variable o función viene determinado por su nombre: si el nombre comienza con dos guiones bajos se trata de una variable o función privada, en caso contrario es pública. Los métodos cuyo nombre comienza y termina con dos guiones bajos son métodos especiales que Python llama automáticamente bajo ciertas circunstancias.

Introducción a Python y PyGame

Orientación a objetos

```
class Fecha():
    def __init__(self):
        self.__dia = 1

    def getDia(self):
        return self.__dia

    def setDia(self, dia):
        if dia > 0 and dia < 31:
            self.__dia = dia
        else:
            print "Error"

mi_fecha = Fecha()
mi_fecha.setDia(33)
```

Introducción a Python y PyGame

Programación Funcional

Es un paradigma en el que la programación se basa casi en su totalidad en funciones, entendiendo el concepto de función según su definición matemática, y no como los simples subprogramas de los lenguajes imperativos.

En los lenguajes funcionales puros un programa consiste exclusivamente en la aplicación de distintas funciones a un valor de entrada para obtener un valor de salida.
Python, sin ser un lenguaje puramente funcional incluye varias características tomadas de los lenguajes funcionales como son las funciones de **orden superior** o las funciones **lambda** (**funciones anónimas**).

Introducción a Python y PyGame

Programación Funcional

El concepto de **funciones de orden superior** se refiere al uso de funciones como si de un valor cualquiera se tratara, posibilitando el pasar funciones como parámetros de otras funciones o devolver funciones como valor de retorno.
Esto es posible porque en Python todo son objetos. Y las funciones no son una excepción.

Programación Funcional

Las funciones de orden superior se pueden pasar como argumentos de las funciones map, filter y reduce.

Estas funciones permiten sustituir los bucles típicos de los lenguajes imperativos mediante construcciones equivalentes.

Programación Funcional

map(function, sequence[, sequence, ...])
La función map aplica una función a cada elemento de una secuencia y devuelve una lista con el resultado de aplicar la función a cada elemento.

```
def cuadrado(n):
   return n ** 2
l = [1, 2, 3]
l2 = map(cuadrado, l)
```

Programación Funcional

filter(function, sequence)

La función filter verifica que los elementos de una secuencia cumplan una determinada condición, devolviendo una secuencia con los elementos que la cumplan.

```
def es_par(n):
  return (n % 2.0 == 0)
l = [1, 2, 3]
l2 = filter(es_par, l)
```

Programación Funcional

reduce(function, sequence[, initial])
La función **reduce** aplica una función a pares de elementos de una secuencia hasta dejarla en un solo valor.
A continuación un ejemplo en el que se utiliza reduce para sumar todos los elementos de una lista.

```
def sumar(x, y):
  return x + y
l = [1, 2, 3]
l2 = reduce(sumar, l)
```

Modulos y paquetes

Los módulos son entidades que permiten una organización y división lógica del código. Los archivos son su contrapartida física: cada archivo Python almacenado en disco equivale a un módulo.

```
import modulo

import os, sys, time
```

Modulos y paquetes

Es necesario preceder el nombre de los objetos que se importan de un módulo con el nombre del módulo al que pertenecen.

Sin embargo es posible utilizar la construcción **from-import** para ahorrar el tener que indicar el nombre del módulo antes del objeto.

```
from time import asctime
```

Modulos y paquetes

Si los módulos sirven para organizar el código, los paquetes sirven para organizar los módulos. Los paquetes son tipos especiales de módulos (ambos son de tipo module) que permiten agrupar módulos relacionados. Mientras los módulos se corresponden a nivel físico con los archivos, los paquetes se representan mediante directorios.

Modulos y paquetes

Para hacer que Python trate a un directorio como un paquete es necesario crear un archivo __init__.py en dicha carpeta.

Para hacer que un cierto módulo se encuentre dentro de un paquete, basta con copiar el archivo que define el módulo al directorio del paquete.

Thread

El trabajo con threads se lleva a cabo en Python mediante el módulo thread. Este módulo es opcional y dependiente de la plataforma.

Además de thread, también se cuenta con el módulo threading que se apoya en el primero para proporcionarnos una API de más alto nivel, más completa, y orientada a objetos

Thread

El módulo threading contiene una clase Thread que se debe extender para crear nuevos hilos de ejecución. El método run contendrá el código que se ejecute el thread.

```
import threading
class MiThread(threading.Thread):
```

Thread

El método join se utiliza para que el hilo que ejecuta la llamada se bloquee hasta que finalice el thread sobre el que se llama. En este caso se utiliza para que el hilo principal no termine su ejecución antes que los hijos.

El método join puede tomar como parámetro un número en coma flotante indicando el número máximo de segundos a esperar.

DocStrings

Todos los objetos cuentan con una variable especial __doc__ mediante la que se puede indicar el propósito y uso del objeto. Estos son los llamados *docstrings o cadenas de documentación.*

DocStrings

Si el primer estamento de la definición del objeto es una cadena, esta se asocia a la variable __doc__ automáticamente.

```
def haz_algo(arg):
“””Este es el docstring de la funcion.”””
```

Introducción a Python y PyGame

Distribucion

Una vez terminado el desarrollo de la aplicación es conveniente empaquetarla de forma que sea sencillo para los usuarios instalarla, y para nosotros distribuirla.

En Python existen dos módulos principales para este cometido: distutils, que es parte de la librería estándar y setuptools, que extiende la funcionalidad de distutils.

Distribución

py2exe, es una extensión para distutils que permite crear ejecutables para Windows a partir de código Python, y que permite ejecutar estas aplicaciones sin necesidad de tener instalado el intérprete de Python en el sistema.

Introducción a Python y PyGame

TEMARIO

1. Python
2. **Pygame**

Introducción a Python y PyGame

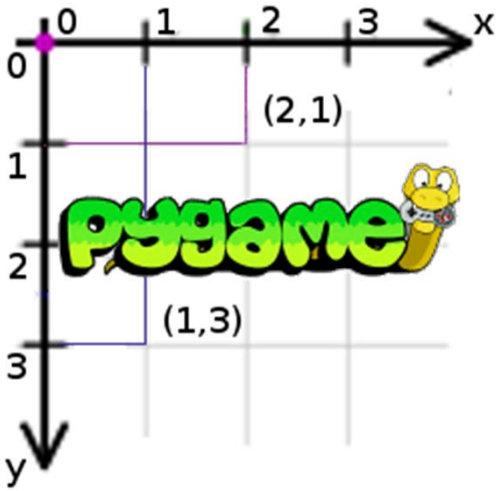
PyGame

Es una biblioteca de Python, que encapsula a SDL.
Se pueden manipular gráficos, audio, sonido, y está orientada al desarrollo de video juegos.

Consta de una serie de módulos que implementan distintas características.

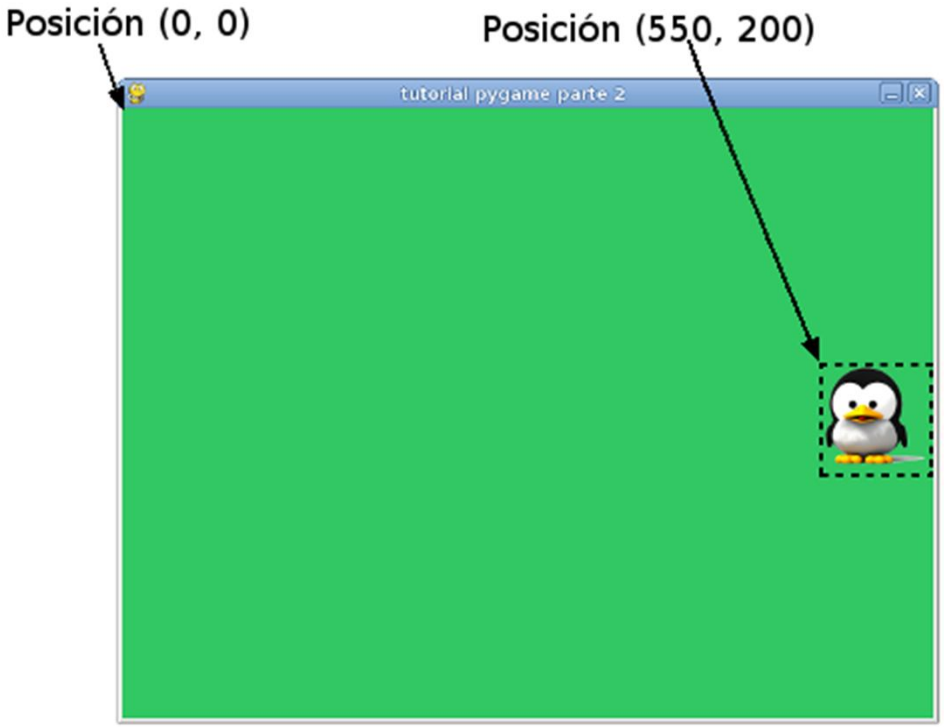
Introducción a Python y PyGame

Módulo-Display
Controla la ventana y pantalla de visualización.
Pygame contiene una sola superficie de visualización que puede estar contenida en una ventana o bien correr en pantalla completa. La pantalla se trata como una superficie más.



Introducción a Python y PyGame

Módulo-Display



Introducción a Python y PyGame

Módulo-Display

pygame.display.init(): return None

Inicializa el módulo display de pygame. Esto generalmente se hace de forma automática cuando llama a la función de mayor nivel pygame.init().

pygame.display.quit(): return None
Apagará el módulo display, cualquier pantalla activa se cerrará.

Introducción a Python y PyGame

Módulo-Display

pygame.display.get_init(): return bool
Retorna True si el módulo pygame.display está inicializado.

pygame.display.get_surface(): return Surface
Retorna una referencia a la superficie de pantalla actual. Si no se ha definido un modo de video esta función retornará None.

Introducción a Python y PyGame

Módulo-Display

pygame.display.set_mode(resolution=(0,0), flags=0, depth=0): return Surface

Esta función construirá una superficie para la pantalla. Los argumentos que se le envían solicitan el tipo de pantalla deseado.
El argumento resolution es una tupla de números que representan el ancho y alto. El argumento flags es una colección de opciones adicionales.
El argumento depth representa el número de bits utilizados por color.

Introducción a Python y PyGame

Módulo-Display

pygame.display.set_mode(resolution=(0,0), flags=0, depth=0): return Surface

El objeto **Surface** retornado se puede manipular como un objeto **Surface** normal, solo que los cambios eventualmente se podrán observar en el monitor.
El argumento *flags* define el tipo de pantalla. Hay varias opciones para elegir, e incluso se puede combinar varias opciones usando operaciones de bits, usando el operador "|" .

Introducción a Python y PyGame

Módulo-Display

La siguiente línea crea la pantalla donde se podrá visualizar el juego**.**

pygame.display.set_mode(resolution=(0,0), flags=0, depth=0): return Surface

Opción	Significado
pygame.FULLSCREEN	Genera una visualización de pantalla completa
pygame.DOUBLEBUF	Recomendado para combinar con HWSURFACE u OPENGL
pygame.HWSURFACE	Aceleración por hardware, solo funciona conjuntamente con FULLSCREEN
pygame.OPENGL	Genera una pantalla que se puede dibujar con opengl
pygame.RESIZABLE	La ventana se debe poder cambiar de tamaño
pygame.NOFRAME	La ventana no deberá tener bordes, titulo o controles

Introducción a Python y PyGame

Módulo-Display

pygame.display.flip(): return None

Actualiza el contenido de la pantalla entera. Si su modo de video usa las
opciones **pygame.HWSURFACE** y **pygame.DOUBLEBUF**, esta operación esperará el retraso vertical e intercambiará las superficies.

Introducción a Python y PyGame

Módulo-Display

pygame.display.update(rectangle=None): return None
pygame.display.update(rectangle_list): return None

Permite que solo una porción de la pantalla se actualice, en lugar de toda el área de ella. Si no se pasan argumentos, se actualizará la superficie completa.
Puede pasar a la función un rectángulo, o una secuencia de rectángulos. Es mas eficiente pasar varios rectángulos de una sola vez en lugar de llamar muchas veces
a update con un solo rectángulo o una lista parcial de ellos.

Módulo-Display

pygame.display.set_icon(Surface): return None.

Define el icono de ejecución que el sistema usará para representar la ventana. Por defecto todas las ventanas muestran el logo de pygame como icono de la ventana. Se puede especificar cualquier superficie, aunque la mayoría de los sistemas esperan una imagen pequeña estilo 32 × 32.

Introducción a Python y PyGame

Módulo-Display

pygame.display.set_caption(title, icontitle=None): return None

Si la pantalla tiene un título de ventana, esta función cambiará ese título. Algunos sistemas soportan un titulo alternativo mas corto para utilizarse en ventanas minimizadas.

Introducción a Python y PyGame

Módulo-Surfaces
Un objeto **Surface** de pygame se utiliza para representar cualquier imagen. La superficie tiene un formato de pixel y resolución fija.

pygame.Surface((width, height), flags=0, Surface): return Surface

Al invocar a pygame.Surface() se creara un nuevo objeto image.

Introducción a Python y PyGame

Módulo-Surfaces

pygame.Surface((width, height), flags=0, Surface): return Surface

El único argumento requerido es el tamaño. La superficie se creará con el formato que mejor coincida con la pantalla actual si no se especifican los argumentos adicionales.
El argumento flags es una combinación de características adiciones para la superficie.
Puede utilizar cualquier combinación de estas:
• **HWSURFACE**: Genera la imagen en la memoria de video.
• **SRCALPHA**: El formato de pixel incluirá transparencias por pixel.

Módulo-Surfaces

Las superficies pueden tener varios atributos adicionales como planos alpha, colores clave o recortes.

Estas funciones afectan principalmente a la forma en que se imprime la superficie sobre otra.

Las rutinas blit intentarán usar aceleración de hardware cuando sea posible, en caso contrario usarán métodos de impresión por software muy optimizados.

Introducción a Python y PyGame

Módulo-Surfaces

Existen tres tipos de transparencia en pygame:

- **colores clave**
- **transparencia de superficie**
- **transparencia de pixel.**

Introducción a Python y PyGame

Módulo-Surfaces

- **La transparencia de superficie**

Es un valor individual que cambia la transparencia de la imagen completa o en parte. Una transparencia de superficie de 255 será opaca mientras que un valor de 0 será completamente transparente.

- **La transparencia por color clave**

Hace transparente un solo color. No se imprimirán los pixeles que coincidan con el color clave.

Módulo-Surfaces

- **La transparencia de pixel**

Se almacena el valor de transparencia para cada pixel. Se utiliza algún formato que soporte transparencias para almacenar la imagen.

Introducción a Python y PyGame

Módulo-Surfaces

Surface.blit(source, dest, area=None, special_flags = 0): return Rect

Dibuja una superficie source sobre otra. La impresión se puede posicionar usando el argumento dest.
dest puede ser un par de coordenadas representando la esquina superior izquierda o bien un rectángulo, cuya esquina superior izquierda representará la posición destino de impresión.

Introducción a Python y PyGame

Módulo-Surfaces

Surface.convert(depth, flags=0): return Surface
Surface.convert(masks, flags=0): return Surface
Surface.convert(): return Surface

Genera una nueva copia de la superficie con un formato de pixel modificado. Este formato será el mas rápido de imprimir.

Introducción a Python y PyGame

Módulo-Surfaces

Surface.convert_alpha(Surface): return Surface
Surface.convert_alpha(): return Surface

Genera una nueva copia de la superficie con el formato de pixel deseado. La superficie nueva tendrá
un formato adecuado para imprimirse mas rápidamente el formato indicado con canal alpha. Si no
se especifica el argumento surface, la nueva superficie se optimizará para el formato de pantalla actual.

Módulo-Surfaces

Surface.copy(): return Surface

Hace una copia duplicada de un superficie. La superficie nueva tendrá el mismo formato de pixel, paletas de colores y configuración de transparencia que la original.

Introducción a Python y PyGame

Módulo-Surfaces

Surface.fill(color, rect=None, special_flags=0): return Rect

Pinta la superficie con un color solido. Se pintará la superficie entera si no se especifica el argumento rect. El argumento rect limitará la modificación al área especificada.

Introducción a Python y PyGame

Módulo-Surfaces

Surface.set_colorkey(Color, flags=0): return None
Surface.set_colorkey(None): return None

Define el color clave para la superficie. Cuando imprima esta superficie sobre otra, cualquier pixel
que tenga el mismo color que el color clave no se imprimirá. El argumento color puede ser un
color RGB o un índice de una paleta de colores. Si se envía None como argumento entonces se deshabilitará el color clave.
Se ignorará el color clave si la superficie tiene un formato para usar valores alpha por pixel.

Módulo-Surfaces

Surface.set_alpha(None): return None

Define el valor de transparencia para la superficie. Cuando se imprima esta superficie sobre otra los
pixels se dibujarán ligeramente transparentes. El valor de transparencia es un número entero de 0 a
255, 0 representa completamente transparente y 255 completamente opaco. Se deshabilitará la
transparencia de la superficie si se pasa None como valor de transparencia.
Se ignorará este valor de transparencia si
la superficie contiene pixels con transparencia.

Módulo-Surfaces

Surface.set_clip(rect): return None
Surface.set_clip(None): return None

Cada superficie tiene una área de recorte activa. Este recorte es un rectángulo que representa a los pixels que se pueden modificar en una superficie. Toda la superficie se podrá modificar si se pasa None como área de recorte.

Introducción a Python y PyGame

Módulo-Surfaces

Surface.get_rect(**kwargs): return Rect

Retorna un nuevo rectángulo que cubre la superficie entera. Este rectángulo siempre comenzará en
la posición (0, 0) y tendrá el ancho y alto idéntico al tamaño de la imagen.
Estos argumentos se aplicarán a los atributos del rectángulo antes de ser retornado.

Introducción a Python y PyGame

Módulo-Surfaces

Surface.subsurface(Rect): return Surface

Retorna una nueva superficie que comparte sus pixels con su superficie pariente. La nueva superficie se considera hija de la original. Las modificaciones a los pixels de cualquier de las dos superficies afectará a la otra.
La información de la superficie como el área de recorte o los colores clave son únicos para cada superficie.

Introducción a Python y PyGame

Módulo-Eventos

Pygame maneja todos sus mensajes de eventos a través de una cola de eventos. Los eventos de entrada son extremadamente dependientes del módulo **display**.

La cola de eventos no funcionará a menos que se halla inicializado el módulo **display** y el modo de video.

La cola de eventos es una lista de objetos de tipo **Event**, y hay diversas maneras de acceder a los eventos que esta contiene. Desde consultar por la existencia de eventos, a extraerlos directamente de la pila.

Introducción a Python y PyGame

Módulo-Eventos

Para mantener a pygame en coherencia con el sistema necesita llamar a pygame.event.pump periódicamente, usualmente una vez por ciclo del bucle de juego.
La cola de eventos ofrece una forma de filtro simple. Esto puede ayudar a mejorar el rendimiento bloqueando ciertos tipos de eventos de la cola, las funciones **pygame.event.set_allowed() y pygame.event.set_blocked()** para trabajar con este filtrado. Por defecto todos los eventos están permitidos.
Los controles de Joystick no emitirán ningún evento a menos que se inicialice el dispositivo.

Módulo-Eventos

Un objeto Evento contiene un tipo de evento y un conjunto de miembros, o atributos, de solo lectura. El objeto Evento no contiene métodos, solo información. Estos objetos se obtienen desde la cola de eventos
El programa debe seguir ciertos pasos para evitar que la cola de eventos se sobrepase del límite. Si el programa no limpia o elimina los eventos de la cola de eventos en intervalos regulares, esta podría desbordarse

Introducción a Python y PyGame

Módulo-Eventos

Tipo (atributo `type`)	Atributos
QUIT	none
ACTIVEEVENT	gain, state
KEYDOWN	unicode, key, mod
KEYUP	key, mod
MOUSEMOTION	pos, rel, buttons
MOUSEBUTTONUP	pos, button
MOUSEBUTTONDOWN	pos, button
JOYAXISMOTION	joy, axis, value
JOYBALLMOTION	joy, ball, rel
JOYHATMOTION	joy, hat, value
JOYBUTTONUP	joy, button
JOYBUTTONDOWN	joy, button
VIDEORESIZE	size, w, h
VIDEOEXPOSE	none
USEREVENT	code

Introducción a Python y PyGame

Módulo-Eventos

pygame.event.pump(): return None

Para cada cuadro de visualización del juego se necesita hacer alguna clase de llamada a la cola de eventos.

Esta función no se necesita si el programa procesa eventos de manera consistente a través de otras funciones de pygame.event.

Introducción a Python y PyGame

Módulo-Eventos

pygame.event.get(): return Eventlist
pygame.event.get(type): return Eventlist
pygame.event.get(typelist): return Eventlist

Obtiene todos los mensajes de eventos y los elimina de la cola.

Si se le especifica un tipo o secuencia de tipos de evento solo esos mensajes se eliminarán de la cola.

Introducción a Python y PyGame

Módulo-Eventos

pygame.event.wait(): return Event

Retorna un evento individual de la cola. Esta función espera hasta que un evento llegue si es que la cola está vacía.

El evento se elimina de la cola una vez que ha sido retornado.

Mientras el programa está esperando el proceso dormirá en un estado de espera.

Módulo-Eventos

pygame.event.peek(type): return bool
pygame.event.peek(typelist): return bool

Retorna True si existen algunos de los eventos solicitados esperando en la cola. Si se pasa una secuencia de tipos de eventos, se retornará True si alguno de esos eventos está en la cola.

Módulo-Eventos

pygame.event.clear(): return None
pygame.event.clear(type): return None
pygame.event.clear(typelist): return None

Elimina todos los eventos (o de un tipo específico) de la cola. Esta función tiene el mismo efecto
que pygame.event.get() excepto que no retorna nada.

Módulo-Mask

Módulo de pygame para gestionar máscaras de imágenes. Útiles para detectar colisiones entre píxeles. Una máscara utiliza 1 bit por *píxel* para almacenar qué partes de una imagen pueden colisionar.

pygame.mask.from_surface(Surface, threshold = 127) -> Mask

Interpreta las partes transparentes de la imagen como no-colisionables y las partes opacas como colisionables, y devuelve una mascara.

Módulo-Rectángulos

Es un módulo que permite almacenar y manipular areas rectangulares. Un objeto Rect se puede crear a partir de una combinación de valores izquierda, arriba, ancho y alto.

Los métodos de Rect que cambian la posición o el tamaño del rectángulo retornan una nueva copia del objeto con los cambios realizados. El rectángulo original no se modifica.

Módulo-Rectángulos

Rect.inflate(x, y): return Rect

Retorna un nuevo rectángulo con el tamaño alterado en la cantidad indicada. El rectángulo se mantiene centrado en el mismo punto. Valores negativos reducen el tamaño del rectángulo.

Módulo-Rectángulos

Rect.move(x, y): return Rect

Retorna un nuevo rectángulo que está desplazado en la cantidad indicada.

Módulo-Rectángulos

Rect.clip(Rect): return Rect

Retorna un nuevo rectángulo que se recorta para estar completamente dentro de otro indicado por parámetro. Si los dos rectángulos no están en colisión, se retornará un rectángulo de tamaño 0.

Módulo-Rectángulos

Rect.union(Rect): return Rect

Retorna un nuevo rectángulo que cubre completamente el área de otros dos rectángulos dados como parámetro. Pueden haber áreas dentro del nuevo rectángulo que no estén en el área de los originales.

Rect.unionall(Rect_sequence): return Rect

Retorna la unión de un rectángulo con una secuencia de varios rectángulos.

Introducción a Python y PyGame

Módulo-Rectángulos

Rect.contains(Rect): return bool
Retorna True cuando el argumento está completamente dentro del objeto receptor.

Rect.collidepoint(x, y): return bool
Rect.collidepoint((x,y)): return bool
Retorna True si el punto dado está dentro del rectángulo. Un punto sobre el costado derecho o inferior del rectángulo no se considera que está dentro del rectángulo.

Módulo-Rectángulos

Rect.colliderect(Rect): return bool

Retorna True si cualquier parte de alguno de los rectángulos están en contacto (excepto los bordes superior+inferior o derecha+izquierda)

Rect.collidelist(list): return index

Consulta si el rectángulo colisiona con otro de una secuencia de rectángulos. Se retorna el índice de la primer colisión que se encuentra. Se retorna el índice -1 si no se encuentran colisiones.

Introducción a Python y PyGame

Módulo-Sprites

Son los objetos del juego, son fáciles de manejar.
Están compuestos por una imagen y una posición.

Existe la clase Group que permite agruparlos, y realizar operaciones sobre todo el conjutno de sprites, y detectar colisiones entre grupos.

Introducción a Python y PyGame

Módulo-Sprites

pygame.sprite.spritecollide(sprite, group, dokill, collided = None): return Sprite_list

Retorna una lista que contiene todos los sprites en un grupo que están colisionando con otro sprite. La intersección se determina comparando el atributo Sprite.rect de cada sprite.
Si dokill vale True todos los sprites que colisionan se eliminarán del grupo.
El argumento collided es una función que se utiliza para calcular si dos sprites están en contacto, esta función debería tomar dos sprites como argumentos y retornar un valor True o False indicado si están colisionando.

Introducción a Python y PyGame

Módulo-Sprites

pygame.sprite.spritecollide(sprite, group, dokill, collided = None): return Sprite_list

Si no se especifica el valor para el argumento, todos los sprites deberán tener un valor rect, que es el rectángulo del área de sprite, que se usará para calcular la colisión.

Funciones de colisión:

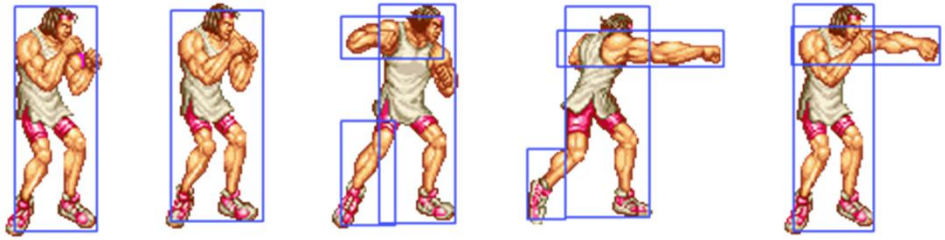
- collide_rect
- collide_rect_ratio
- collide_circle
- collide_circle_ratio
- collide_mask

Introducción a Python y PyGame

Módulo-Sprites

pygame.sprite.collide_rect(left, right): return bool

Consulta la colisión entre dos sprites. Usa la función collideрect del módulo rect para calcular la colisión. Los sprites deben tener atributos rect.



Introducción a Python y PyGame

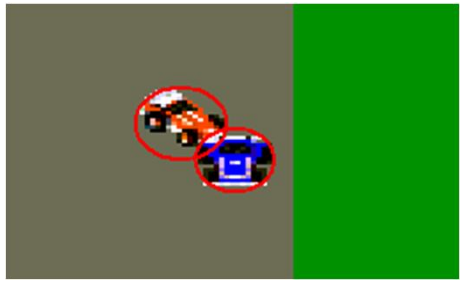
Módulo-Sprites

pygame.sprite.collide_rect_ratio(ratio): return collided_callable

Verifica colisiones entre dos sprites usando usa versión reducida de los rectángulos de sprite.
Se generan con un radio, y la instancia retornada está diseñada para ser enviada como una función de colisión a las funciones generales de colisión.

Introducción a Python y PyGame

Módulo-Sprites



pygame.sprite.collide_circle(left, right): return bool

Verifica la colisión entre dos sprites, verificando si dos círculos centrados en los sprites están en contacto. Si el sprite tiene un atributo radius este se usará para crear en círculo, en caso de que no exista se creará un círculo lo suficientemente grande para contener todo el rectángulo del sprite indicado por el atributo rect.
Los sprites deben tener los atributos rect y radius (este último es opcional).

Módulo-Sprites

pygame.sprite.collide_circle_ratio(ratio): return collided_callable

Verifica colisiones entre dos sprites usando usa versión reducida de los círculos de sprite.
Comprueba si los dos círculos con centro en los sprites están en contacto luego de haberlos alterado de tamaño.
Lo los sprites tienen un atributo radius este se usará para crear el círculo, en otro caso se creará un círculo lo suficientemente grande para contener por completo el rectángulo de sprite según su atributo rect.

Introducción a Python y PyGame

Módulo-Sprites

pygame.sprite.collide_circle_ratio(ratio): return collided_callable

El objeto creado verifica la existencia de colisión entre dos sprites, comprobando si los dos círculos con centro en los sprites están en contacto luego de haberlos alterado de tamaño. Lo los sprites tienen un atributo radius este se usará para crear el círculo, en otro caso se creará un círculo lo suficientemente grande grande para contener por completo el rectángulo de sprite según su atributo rect. Los sprites deben tener los atributos rect y radius (este último es opcional).

Módulo-Sprites

pygame.sprite.collide_mask(SpriteLeft, SpriteRight): return bool

Verifica la colisión entre dos sprites, probando si sus máscaras de bits se superponen. Si el sprite tiene un atributo mask, este atributo se usará como máscara, en otro caso se creará la máscara a partir de la imagen del sprite.

Los sprites deben tener un atributo rect y un atributo opcional de nombre mask.

Módulo-Sprites

pygame.sprite.groupcollide(group1, group2, dokill1, dokill2): return Sprite_dict

Esta función encontrará intersecciones entre todos los sprites de dos grupos. Las intersecciones se determinan comparando los atributos Sprite.rect de cada Sprite. Cada sprite dentro del grupo group1 se agrega al diccionario de retorno como clave. El valor de cada elemento será una lista de los sprites del grupo group2 que colisionan con el primero.

Introducción a Python y PyGame

Módulo-Sprites

pygame.sprite.spritecollideany(sprite, group): return bool
Consulta si el sprite dado colisiona con algún sprite en el grupo. La intersección se determina comparando el atributo Sprite.rect de cada sprite.
Esta prueba de colisión puede ser mas rápida que pygame.sprite.spritecollideany() dado que tiene menos trabajo para hacer. Retornará al encontrar la primer colisión.

Sprites y Group

Los objetos Sprite tienen métodos para trabajar con los grupos.

__init__: En el constructor se le puede especificar a que grupos pertenece en el momento de su creación.
add(): Agrega el sprite a un grupo.
remove(): Elimina el sprite de un grupo.
kill(): Elimina el sprite de todos los grupos.
alive(): Nos devuelve true si el sprite pertenece todavía a algún grupo.

Módulo-Group

Los objetos Group tienen métodos para trabajar con los sprites.

__init__: En el constructor se puede especificar que sprites pertenecen en el momento de su creación.

add(): Agrega un sprite al grupo.

remove(): Elimina un sprite del grupo.

empty(): Vacía el grupo.

copy(): Devuelve una copia del grupo con todos sus miembros.

has(): Comprueba si pertenecen al grupo los sprites especificados.

Módulo-Group

sprites(): Devuelve una lista de los sprites del grupo.
update(): Ejecuta el método update() en cada sprite del grupo.
len(): Devuelve el numero de sprites del grupo.

Tipos de Grupos

Group
Todos los demás grupos derivan de el añadiendo funcionalidad.
GroupSigle
Solo almacena un sprite, cuando se inserta uno elimina al anterior.
RenderPlain
Añade el método draw que dibuja todos los sprites que contiene. Los sprites necesitan tener los atributos image y rect para saber que y donde dibujar.

Introducción a Python y PyGame

Tipos de Grupos

RenderClear
Hereda de RenderPlain, se agrega el método clear() que limpia la posición anterior de los sprites.
Utiliza una imagen de fondo para rellenar las posiciones antiguas.

RenderUpdates
Hereda del grupo RenderClear, modica el metodo draw para que devuelva una lista de rects con las posiciones que han cambiado.

Tipos de Grupos

Es necesario saber que tipo de Render Group utilizar. Para juegos con fondos muy dinámicos es mejor usar RenderPlain.

En cambio, para juegos con fondo estático será mejor usar RenderUpdates.

Introducción a Python y PyGame

Módulo-Font

Para trabajar con textos pygame se dispone del módulo Font.

- **pygame.font.Font()**: crea un objeto font con la fuente y el tamaño pasados por parámetro.
- **render()**: Devuelve una superficie con el texto y el color que le pasamos como parámetro. También se puede establecer si se usa o no antialiasing.
- **set_bold()**: Pone el texto en negrita.
- **set_italic()**: Pone el texto en italica.
- **set_underline()**: Pone el texto subrayado.

Introducción a Python y PyGame

Módulo-Imagen

pygame.image.load(): Devuelve una superficie con la imagen que se le ha pasado como parámetro.
pygame.image.save(): Guarda en un archivo con formato TGA o BMP la superficie que se le pasa como parámetro.
pygame.image.tostring(): Pasa una superficie a una cadena de texto.
pygame.image.fromstring(): Pasa una cadena a una superficie.

Introducción a Python y PyGame

Módulo-Sonidos

pygame.mixer.Sound(): Devuelve un objeto de tipo sound haciendo referencia al archivo que se pasa como parámetro.
play(): Este método reproduce el sonido, se le puede especicar el numero de iteraciones y el tiempo máximo para reproducir el sonido.
stop(): Para todos las reproducciones de éste sonido.
set_volume(): Establece el volumen de reproducción, valor entre 0.0 y 1.0.
get_length(): Devuelve el número de segundos de datos que tiene el archivo.

Introducción a Python y PyGame

Ejemplo PyGame

Los distintos componentes que componen Pygame se pueden inicializar por separado o todos juntos.

Inicializando todo pygame con la línea:

```
pygame.init()
```

Esta se debe ejecutar antes de empezar a usar Pygame

```
if __name__ == '__main__':
    pygame.init()
    main()
```

Introducción a Python y PyGame

Ejemplo PyGame

```
import sys, pygame
from pygame.locals import *

def main():
    screen = pygame.display.set_mode((WIDTH,  HEIGHT))
    pygame.display.set_caption("Pruebas Pygame")
    while True:
        for eventos in pygame.event.get():
            if eventos.type == QUIT:
                sys.exit(0)
    return 0
```

Introducción a Python y PyGame

Ejemplo PyGame

Cargando imágenes

```
def load_image(filename, transparent=False):
        try: image = pygame.image.load(filename)
        except pygame.error, message:
                raise SystemExit, message
        image = image.convert()
        if transparent:
                color = image.get_at((0,0))
                image.set_colorkey(color, RLEACCEL)
        return image
```

Introducción a Python y PyGame

Ejemplo PyGame

```
def main():
    screen = pygame.display.set_mode((WIDTH, HEIGHT))
    pygame.display.set_caption("Pruebas Pygame")

    background_image = load_image('images/fondo.jpg')

    while True:
        for eventos in pygame.event.get():
            if eventos.type == QUIT:
                sys.exit(0)

        screen.blit(background_image, (0, 0))
        pygame.display.flip()
    return 0
```

Introducción a Python y PyGame

Ejemplo PyGame

Creando un Sprite

```
class Bala(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = load_image("images/bala.png", True)
        self.rect = self.image.get_rect()
        self.rect.centerx = WIDTH / 2
        self.rect.centery = HEIGHT / 2
        self.speed = [1.5, -0.5]
```

Ejemplo PyGame

Creando un reloj

Para crear un reloj que por ejemplo sea util para saber cuanto tiempo paso desde la anterior actualización, se puede hacer desde la siguiente manera:

Introducción a Python y PyGame

Ejemplo PyGame

Creando un reloj

clock = pygame.time.Clock()

Esta línea va justo antes de entrar en el bucle del juego y sirve para crear el reloj con el que gestionar el tiempo. Para saber cuanto tiempo pasa cada vez que se ejecuta una interacción del bucle, dentro del bucle ponemos como primera línea: time = clock.tick(FRAMERATE)

El FRAMERATE que se le pasa como parámetro es un valor numérico positivo y con el se consigue acotar la velocidad máxima en la que el juego corra en cualquier equipo.

Introducción a Python y PyGame

Ejemplo PyGame

```
while True:
     time = clock.tick(60)
     keys = pygame.key.get_pressed()
     for eventos in pygame.event.get():
         if eventos.type == QUIT:
             sys.exit(0)


def actualizar(self, time):
    self.rect.centerx += self.speed[0] * time
    self.rect.centery += self.speed[1] * time
    if self.rect.left <= 0 or self.rect.right >= WIDTH:
        self.speed[0] = -self.speed[0]
        self.rect.centerx += self.speed[0] * time
    if self.rect.top <= 0 or self.rect.bottom >= HEIGHT:
        self.speed[1] = -self.speed[1]
        self.rect.centery += self.speed[1] * time
```

Introducción a Python y PyGame

Ejemplo PyGame

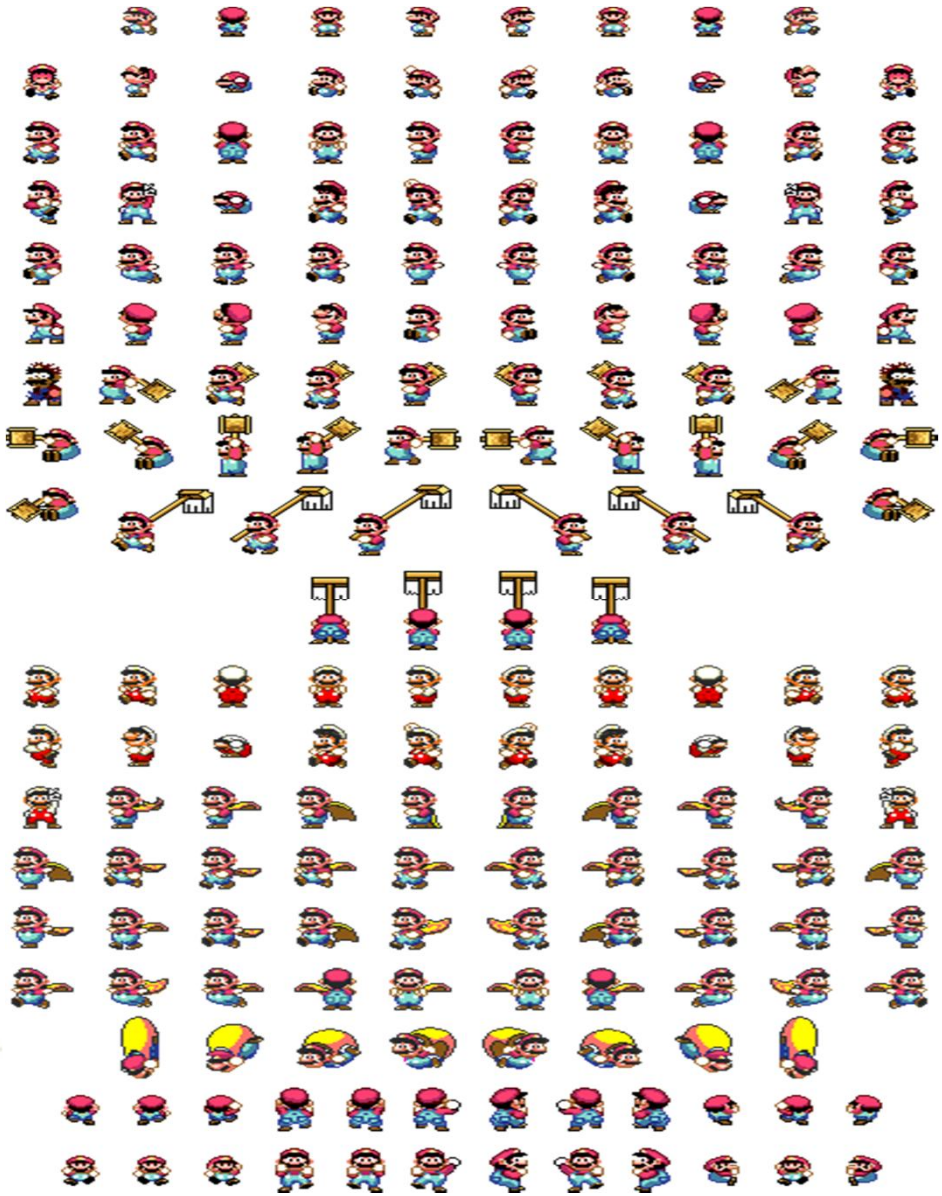
```
  def mover(self, time, keys):
        if self.rect.top >= 0:
            if keys[K_UP]:
                self.rect.centery -= self.speed * time
        if self.rect.bottom <= HEIGHT:
            if keys[K_DOWN]:
                self.rect.centery += self.speed * time
```

```
import pygame
class Mundo(object):
    def __init__(self):   self.image =
    pygame.image.load("images/background.png")
    self.rect = self.image.get_rect()   # [[x1, x2, y], [[x1,x2, y], etc]
    self.solids = [ [-10, 290, 355],
                    [70, 165,  291],
                    [38, 120, 228],
                    [70, 149, 164],
                    [119, 216, 116],
                    [259, 336, 132]
                  ]
```

Introducción a Python y PyGame

Ejemplo PyGame

Sprite sheets



Introducción a Python y PyGame

Ejemplo PyGame

```
import pygame
from states import walking, running, jumping, falling, standing

class Mario(pygame.sprite.Sprite):
    """This class represents Mario"""
    def __init__(self, position, world):
        pygame.sprite.Sprite.__init__(self)
        self.world = world

        self.image = pygame.image.load('images/smw_mario_sheet.pr
        self.rect = self.image.get_rect()

        self.actions = {"front": (245, 75, 20, 30),
                        "back": (285, 75, 20, 30),
                        "left": (165, 75, 20, 30),
                        "left-walking1": (45, 75, 20, 30),
                        "left-walking2": (5, 75, 20, 30),
                        "left-jump": (165, 115, 20, 30),
                        "left-fall": (125, 115, 20, 30),
                        "left-run1": (165, 155, 20, 30),
                        "left-run2": (125, 155, 20, 30),
                        "left-run3": (85, 155, 20, 30),
                        "left-turn": (5, 115, 20, 30),

                        "right": (205, 75, 20, 30),
                        "right-walking1": (325, 75, 20, 30),
                        "right-walking2": (365, 75, 20, 30),
                        "right-jump": (205, 115, 20, 30),
                        "right-fall": (245, 115, 20, 30),
                        "right-run1": (205, 155, 20, 30),
                        "right-run2": (245, 155, 20, 30),
                        "right-run3": (285, 155, 20, 30),
                        "right-turn": (365, 115, 20, 30)
                        }

        self.action = "left"
        self.area = pygame.rect.Rect(self.actions[self.action])
        self.rect.topleft = position
```

Introducción a Python y PyGame

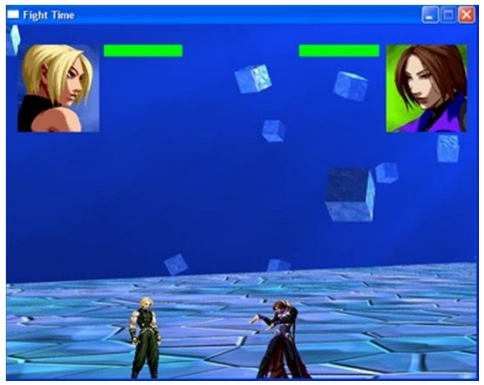
Ejemplo PyGame

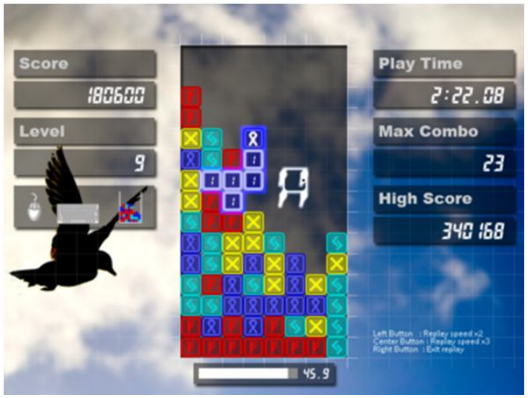
```
for event in pygame.event.get():
        if event.type == QUIT:
                return
        elif event.type == KEYDOWN:
                if event.key == K_a:
                        player1.moveup()
                if event.key == K_z:
                        player1.movedown()
                if event.key == K_UP:
                        player2.moveup()
                if event.key == K_DOWN:
                        player2.movedown()
        elif event.type == KEYUP:
                if event.key == K_a or event.key == K_z:
                        player1.movepos = [0,0]
                        player1.state = "still"
                if event.key == K_UP or event.key == K_DOWN:
                        player2.movepos = [0,0]
                        player2.state = "still"
```

Introducción a Python y PyGame

Juegos Implementados con Pygame

- The Fight Time
- Howitzer
- Elder War





Introducción a Python y PyGame

- http://pyspanishdoc.sourceforge.net
- http://python.org.ar
- Python para todos Raúl González Duque
- http://www.losersjuegos.com.ar/traducciones/pygame
- http://huscorp.nl/2009/08/animate-mario-using-pygame/
- http://pygame.org/project-Tiled+TMX+Loader-2036-3586.html
- http://www.mapeditor.org/
- http://razonartificial.com/2010/02/pygame-1-importar-inicializar/
- http://thepythongamebook.com/en:part2:pygame:step018


03
2 hits

Instalación del entorno de desarrollo para Python

Andrés Pastorini - Alvaro Martínez

Introducción a Python y PyGame

Instalación del SDK y del módulo PyGame

- Descargar e instalar la versión 2.5.4 desde http://www.python.org/getit/releases/2.5.4/

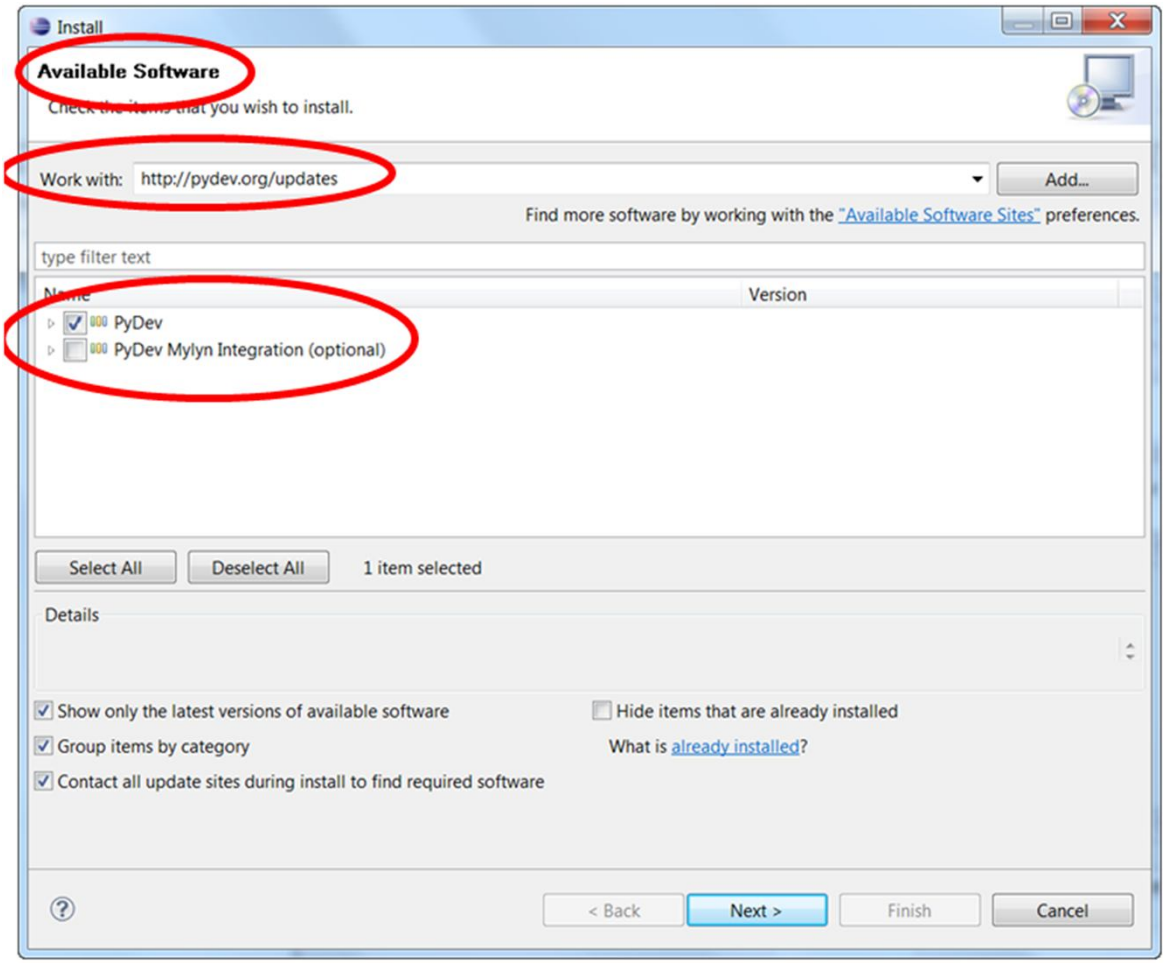
- Descargar e instalar pygame 1.9.1 para python 2.5.4 desde http://pygame.org/download.shtml

Introducción a Python y PyGame

Entorno de Desarrollo Eclipse

- Descargar de http://www.eclipse.org/downloads/, la versión Eclipse IDE para desarrolladores java.
- Ir a la opción de menu Help → Install New Software y en el campo work with ingresar http://pydev.org/updates
- Seleccionar la opción PyDev.

Introducción a Python y PyGame



Introducción a Python y PyGame

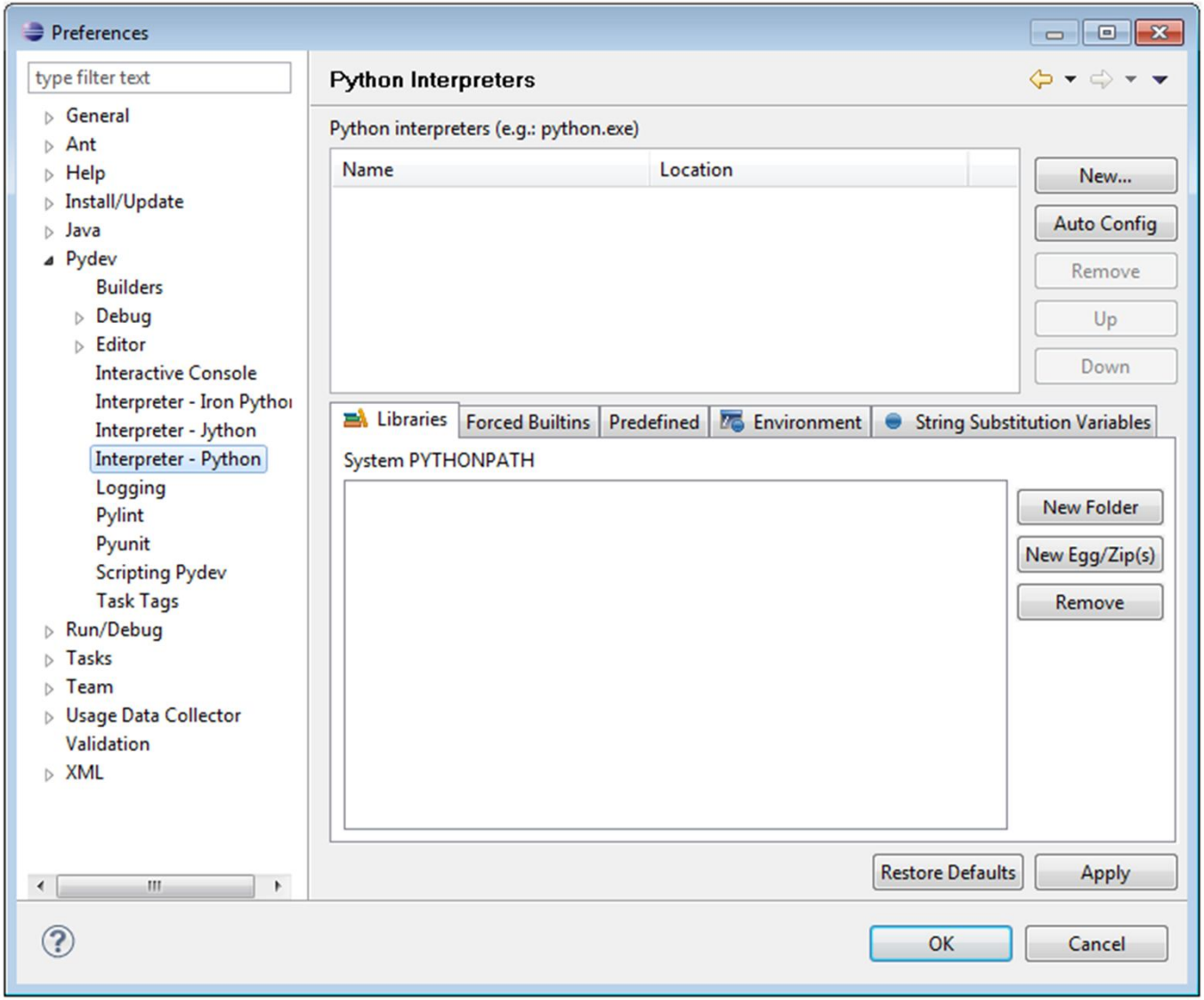
Configurar PyDev

•Ir a Window → Preferences, expandir PyDev y seleccionar interprete Python.

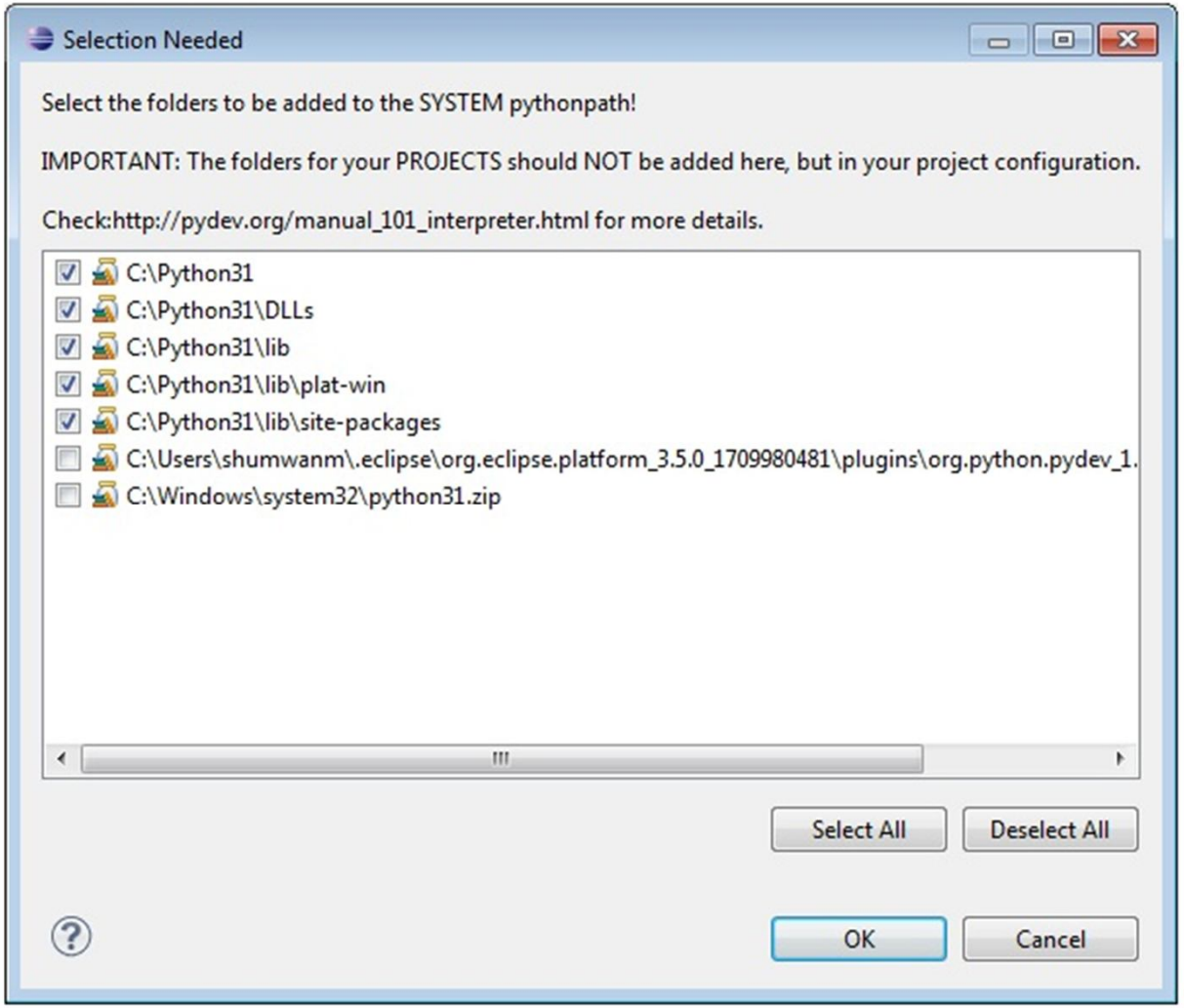
•Hacer click en nuevo e indicar la ruta donde se encuentra python sdk, seleccionar como tipo de interprete a Python.

•Para crear un nuevo proyecto, ir a la perspectiva PyDev y crear un PyDev Project.

Introducción a Python y PyGame



Introducción a Python y PyGame



Introducción a Python y PyGame

Netbean

Descargar netbean para python de http://dlc.sun.com.edgesuite.net/netbeans/6.5/python/ea/start.html?platform=windows&lang=en&option=python