**Coordinate Systems**[**#**](http://effbot.org/tkinterbook/canvas.htm#coordinate-systems)

The **Canvas** widget uses two coordinate systems; the window coordinate system (with (0, 0) in the upper left corner), and a canvas coordinate system which specify where the items are drawn. By scrolling the canvas, you can specify which part of the canvas coordinate system to show in the window.

The **scrollregion** option is used to limit scrolling operations for the canvas. To set this, you can usually use something like:

canvas.config(scrollregion=canvas.bbox(ALL))

To convert from window coordinates to canvas coordinates, use the **[canvasx](http://effbot.org/tkinterbook/canvas.htm" \l "Tkinter.Canvas.canvasx-method)**and **[canvasy](http://effbot.org/tkinterbook/canvas.htm" \l "Tkinter.Canvas.canvasy-method)** methods:

def **callback**(event):

canvas = event.widget

x = canvas.canvasx(event.x)

y = canvas.canvasy(event.y)

print canvas.find\_closest(x, y)

from tkinter import \*

master = Tk()

w = Canvas(master, width=820, height=580)

w.pack()

i=w.create\_line(50, 50, 0, 0,fill="red")

w.coords(i,100,100,0,0)

w.itemconfig(i,fill="blue")

# remove all items

mainloop()

### Item Specifiers: Handles and Tags [#](http://effbot.org/tkinterbook/canvas.htm#item-specifiers)

The **Canvas** widget allows you to identify items in several ways. Everywhere a method expects an item specifier, you can use one of the following:

* item handles (integers)
* tags
* **ALL**
* **CURRENT**

**Item handles** are integer values used to identify a specific item on the canvas. Tkinter automatically assigns a new handle to each new item created on the canvas. Item handles can be passed to the various canvas methods either as integers or as strings.

**Tags** are symbolic names attached to items. Tags are ordinary strings, and they can contain anything except whitespace (as long as they don’t look like item handles).

An item can have zero or more tags associated with it, and the same tag can be used for more than one item. However, unlike the **Text** widget, the **Canvas**widget doesn’t allow you to  create bindings or otherwise configure tags for which there are no existing items. Tags are owned by the items, not the widget itself. All such operations are ignored.

You can either specify the tags via an option when you create the item, set them via the **[itemconfig](http://effbot.org/tkinterbook/canvas.htm" \l "Tkinter.Canvas.itemconfig-method)** method, or add them using the **[addtag\_withtag](http://effbot.org/tkinterbook/canvas.htm" \l "Tkinter.Canvas.addtag_withtag)** method. The **tags** option takes either a single tag string, or a tuple of strings.

item = canvas.create\_line(0, 0, 100, 100, tags="uno")

canvas.itemconfig(item, tags=("one", "two"))

canvas.addtag\_withtag("three", "one")

To get all tags associated with a specific item, use **gettags**. To get the handles for all items having a given tag, use **find\_withtag**.

>>> print canvas.gettags(item)

('one', 'two', 'three')

>>> print canvas.find\_withtag("one")

(1,)

Por si acasooo

elif tecla.char=="e":

marco.move(jeep,3,-3)

marco.after(1, marco.move(jeep,3,-3))

elif tecla.char == "q":

marco.move(jeep,-3,-3)

marco.after(1, marco.move(jeep,-3,-3))

# mover el canvas en este solo se mueve la imagen

for x in range (0,120):

marco.move(miniCuper,1,5,)

marco.update()

time.sleep(0.05)

# mover el canvas en si todo

for x in range (0,120):

marco.move(1,5,0)

miniCuper.update()

time.sleep(0.05)

instanciador

using UnityEngine;

using System.Collections;

public class Instanciador : MonoBehaviour {

public GameObject[] objetos;

public float min = 1f;

public float max = 2f;

// Use this for initialization

void Start () {

Instanciar();

}

// Update is called once per frame

void Instanciar () {

Instantiate(objetos[Random.Range(0, objetos.Length)],transform.position,Quaternion.identity);

Invoke("Instanciar", Random.Range(min, max));

}

}

using UnityEngine;

using System.Collections;

public class camerafollow : MonoBehaviour {

public GameObject Jugador;

// Update is called once per frame

void Update () {

transform.position = new Vector3(Jugador.transform.position.x + 6, transform.position.y, transform.position.z);

}

}