Blender Tutorial

Creazione gioco in FPS

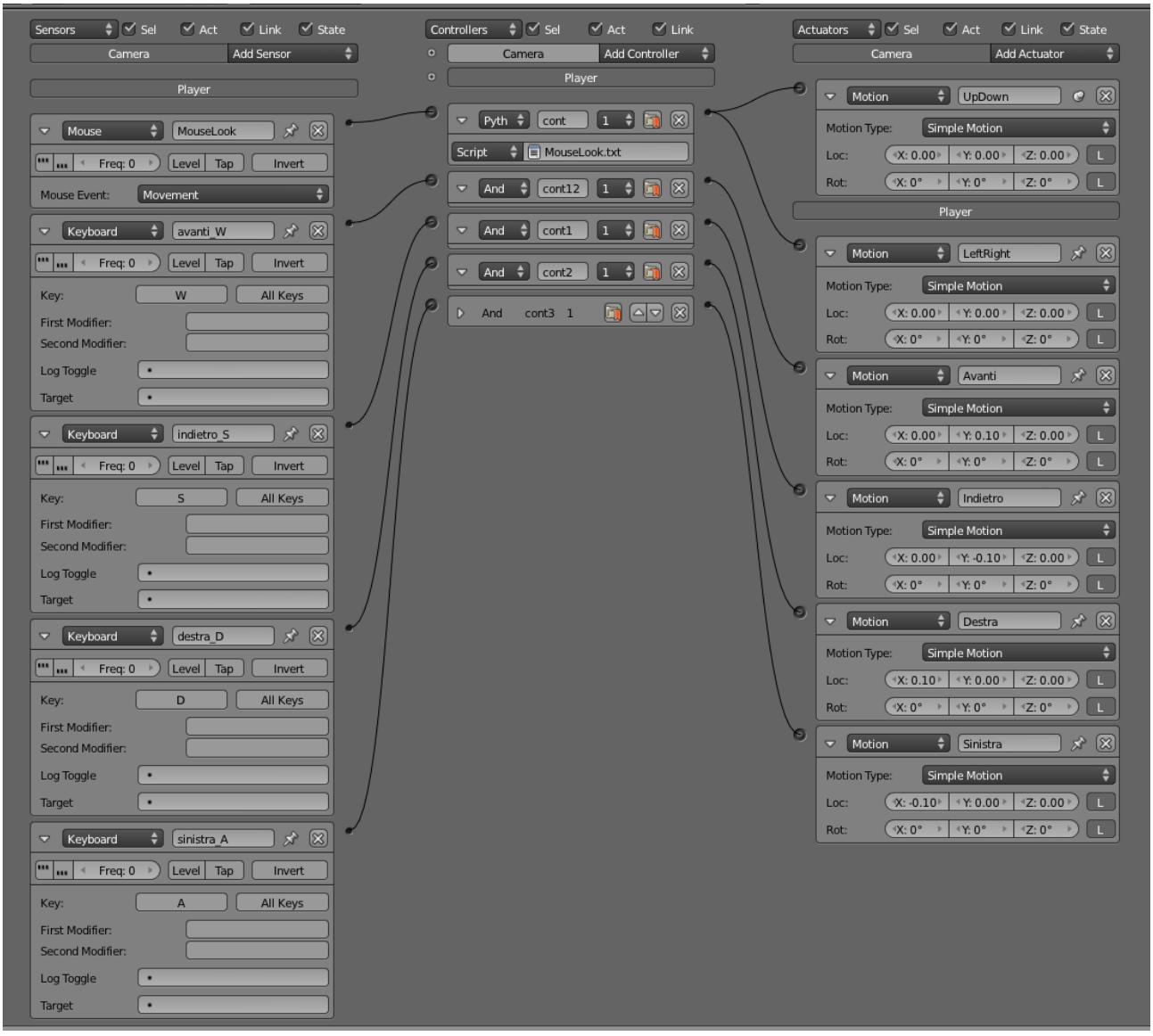
Questo video tutorial è stato preso
da you tube e lo script richiama il sito
www.tutorialsforblender3d.com .

Traduzione da Paolo Lo Bello Alias
Wildlux, sito personale dell'autore
www.blending.altervista.org

Per la creazione di un gioco FPS ci devono essere per forza questi ingredienti :

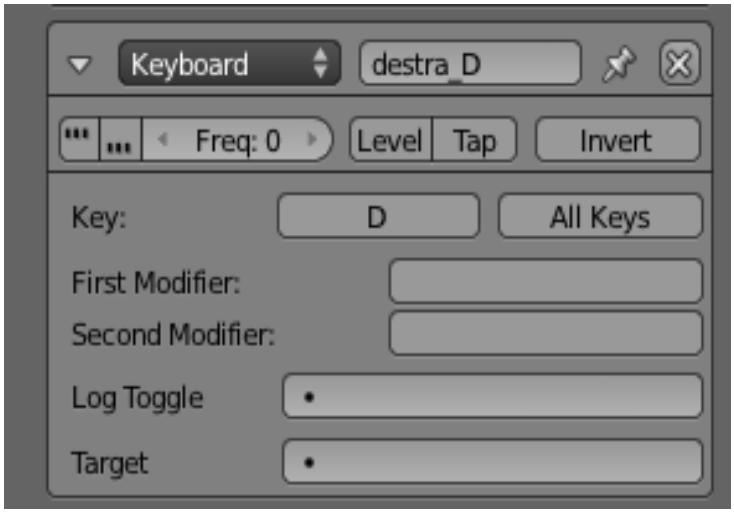
- Script (Mouse Look.txt)
- Camera
- Cubo

Iniziamo a collegare i mattoncini di logica come qui sotto.
Nelle pagine successive vedremo tutto questo in dettaglio.

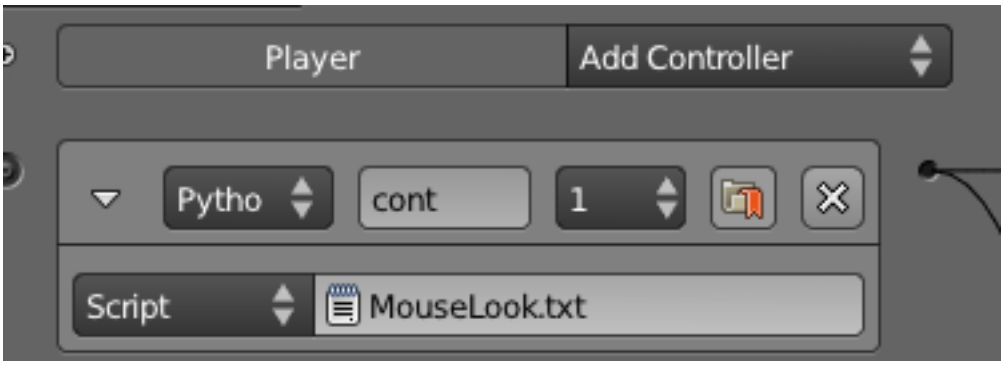


Impostiamo la logica dell'oggetto della scena Cubo.

Creiamo 4 sensori di tipo Tastiera (keyboard) ed uno mouse, come in figura qui sotto.

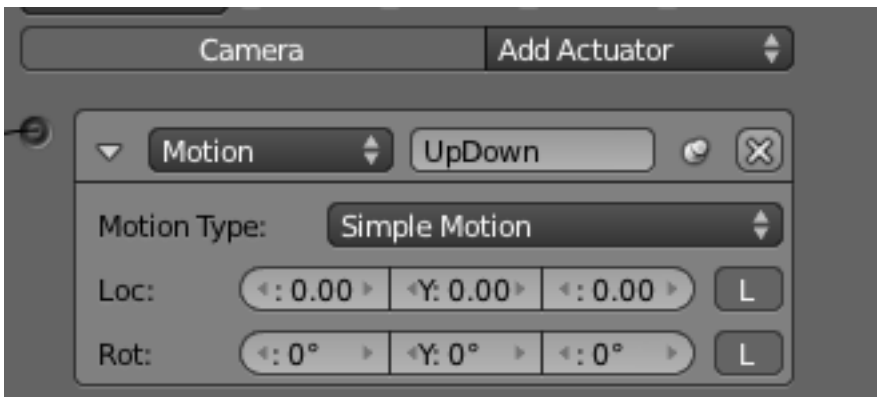


Il nome del sensore Mouse deve essere lo stesso dello script che vedremo più avanti.

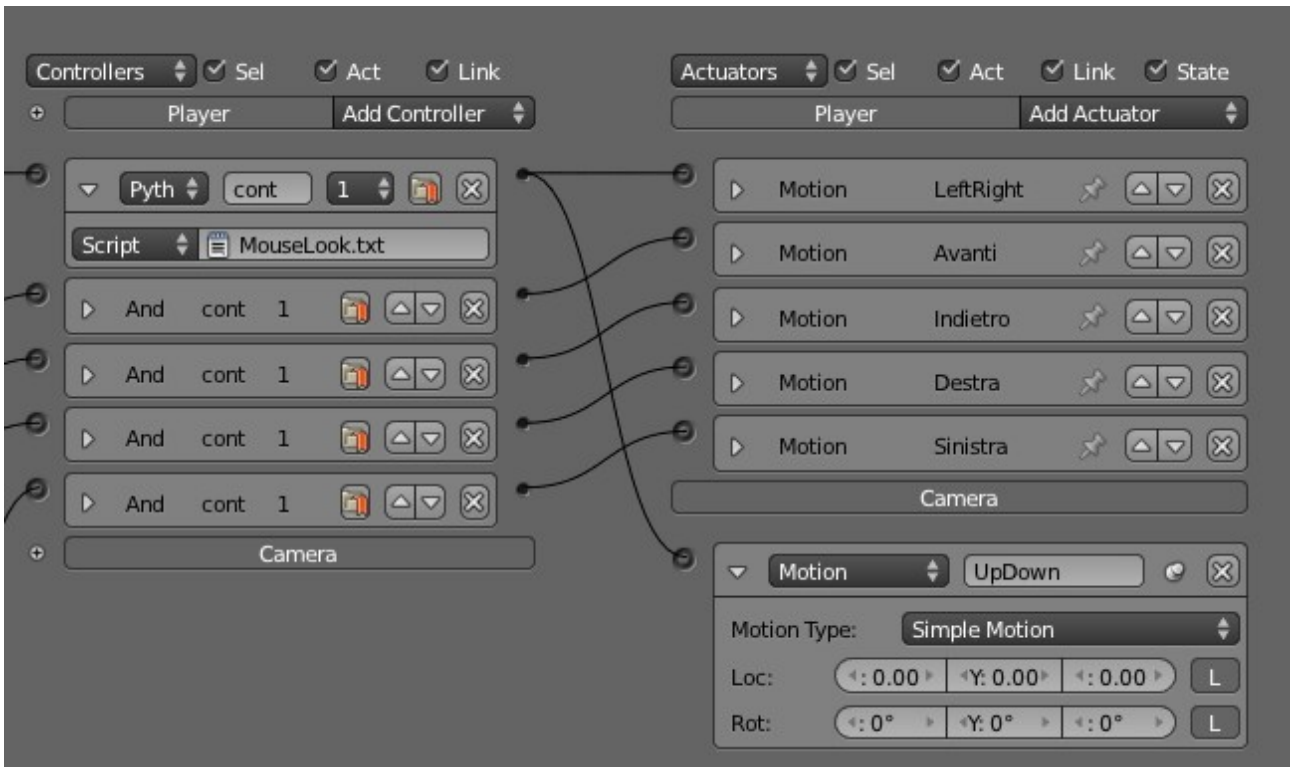


Per convenzione scrivete il nome del sensore mouse MouseLook con l'opzione Movement.
Creiamo 4+1 controllori di cui collegheremo le 4 delle keyboard a delle AND e poi uno di tipo Script Python selezioniamo in questa casella “Script” .

Selezioniamo sia la camera che il cubo.
Creiamo 2 Attuatori uno nella Camera ed uno nel Cubo.
Scriviamo il nome UpDown come in figura.



I'altro attuatore lo chiamiamo LeftRight.
Colleghiamo il controllore python con i due motion LeftRight e UpDown.



Ecco tutto Il tutorial è finito adesso vi elencherò il codice da inserire in questo famoso MouseLook.txt

```
#########################################################
#
#   MouseLook.py        Blender 2.54
#
#           Adapted by ProperVandal from 2.49 source at tutorialsforblender3d.com
#
#   Released under the Creative Commons Attribution 3.0 Unported License.
#
#   If you use this code, please include this information header.
#
#########################################################

import bge

# define main program
def main():

        # set default values
        Sensitivity =  0.0005
        Invert = 1
        Capped = False


        # get controller
        controller = bge.logic.getCurrentController()

        # get the object this script is attached to
        obj = controller.owner

        # get the size of the game screen
        gameScreen = gameWindow()

        # get mouse movement
        move = mouseMove(gameScreen, controller, obj)

        # change mouse sensitivity?
        sensitivity =  mouseSen(Sensitivity, obj)

        # invert mouse pitch?
        invert = mousePitch(Invert, obj)

        # upDown mouse capped?
        capped = mouseCap(Capped, move, invert, obj)

        # use mouse look
        useMouseLook(controller, capped, move, invert, sensitivity)

        # Center mouse in game window
        centerCursor(controller, gameScreen)

#########################################################

# define game window
def gameWindow():

        # get width and height of game window
        width = bge.render.getWindowWidth()
        height = bge.render.getWindowHeight()
```

```
	return (width, height)

#########################################################

# define mouse movement function
def mouseMove(gameScreen, controller, obj):

	# Get sensor named MouseLook
	mouse = controller.sensors["MouseLook"]

	# extract width and height from gameScreen
	width = gameScreen[0]
	height = gameScreen[1]

	# distance moved from screen center
	x = width/2 - mouse.position[0]
	y = height/2 - mouse.position[1]

	# initialize mouse so it doesn't jerk first time
	if 'mouseInit' in obj == False:
		obj['mouseInit'] = True
		x = 0
		y = 0

	#########	stops drifting on mac osx

	# if sensor is deactivated don't move
	if not mouse.positive:
		x = 0
		y = 0

	#########  -- mac fix contributed by Pelle Johnsen

	# return mouse movement
	return (x, y)


#########################################################

# define Mouse Sensitivity
def mouseSen(sensitivity, obj):

	# check so see if property named Adjust was added
	if 'Adjust' in obj == True:

		# Don't want Negative values
		if obj['Adjust'] < 0.0:
			obj['Adjust'] = 0.0

		# adjust the sensitivity
		sensitivity = obj['Adjust'] * sensitivity

	# return sensitivity
	return sensitivity

#########################################################

# define Invert mouse pitch
def mousePitch(invert, obj):

	# check to see if property named Invert was added
	if 'Invert' in obj == True:

		# pitch to be inverted?
		if obj['Invert'] == True:
			invert = -1
		else:
			invert = 1

	# return mouse pitch
```

```
	return invert

##################################################

# define Cap vertical mouselook
def mouseCap(capped, move, invert, obj):

	# check to see if property named Cap was added
	if 'Cap' in obj == True:

		# import mathutils
		import Mathutils

		# limit cap to 0 - 180 degrees
		if obj['Cap'] > 180:
			obj['Cap'] = 180
		if obj['Cap'] < 0:
			obj['Cap'] = 0

		# get the orientation of the camera to world axis
		camOrient = obj.orientation

		# get camera Z axis vector
		camZ = [camOrient[0][2], camOrient[1][2], camOrient[2][2]]

		# create camera z axis vector
		vec1 = Mathutils.Vector(camZ)

		# get camera parent
		camParent = obj.parent

		# get parent orientation to world axis
		parentOrient = camParent.orientation

		# get parent z axis vector
		parentZ = [parentOrient[0][2], parentOrient[1][2], parentOrient[2][2]]

		# create parent z axis vector
		vec2 = Mathutils.Vector(parentZ)

		# find angle between two
		angle = Mathutils.AngleBetweenVecs(vec1, vec2)

		# get amount to limit mouselook
		capAngle = obj['Cap']

		# get mouse up down movement
		moveY = move[1] * invert

		# check capped angle against against camera z-axis and mouse y movement
		if (angle > (90 + capAngle/2) and moveY > 0)   or (angle < (90 - capAngle/2) and moveY < 0)  == True:

			# no movement
			capped = True

	# return capped
	return capped

##################################################

# define useMouseLook
def useMouseLook(controller, capped, move, invert, sensitivity):

	# get up/down movement
	if capped == True:
		upDown = 0
	else:
		upDown = move[1] * sensitivity * invert

	# get left/right movement
```

```
        leftRight = move[0] * sensitivity * invert

        # Get the actuators
        act_LeftRight = controller.actuators["LeftRight"]
        act_UpDown = controller.actuators["UpDown"]

        # set the values
        act_LeftRight.dRot = [ 0.0, 0.0, leftRight]
        act_LeftRight.useLocalDRot = False

        act_UpDown.dRot = [ upDown, 0.0, 0.0]
        act_UpDown.useLocalDRot = True

        # Use the actuators
        controller.activate(act_LeftRight)
        controller.activate(act_UpDown)

##############################################

# define center mouse cursor
def centerCursor(controller, gameScreen):

        # extract width and height from gameScreen
        width = gameScreen[0]
        height = gameScreen[1]

        # Get sensor named MouseLook
        mouse = controller.sensors["MouseLook"]

        # get cursor position
        pos = mouse.position

        # if cursor needs to be centered
        if pos != [ int(width/2), int(height/2)]:

                # Center mouse in game windowp
                bge.render.setMousePosition(int(width/2), int(height/2))

        # already centered.  Turn off actuators
        else:
                # Get the actuators
                act_LeftRight = controller.actuators["LeftRight"]
                act_UpDown = controller.actuators["UpDown"]

                # turn off the actuators
                controller.deactivate(act_LeftRight)
                controller.deactivate(act_UpDown)

##############################################

# Run program
main()
```