Lista delle opzioni che ti permette di fare il Game Engine

Premessa

Questo manuale serve sia ai principianti che ai professionisti che non si ricordano una determinata cosa e si vogliono rinfrescare gli studi .Per i principianti serve come studio ,ad ogni modo buona lettura .Se trovate una cosa del genere keyboard==>and==>motion sono i mattoncini di logica da inserire ,il primo campo in questo caso keyboard è il sensore , and il controllore mentre il terzo campo è attuatore cioè colui che attua ed esegue le azioni per il 90 % dei casi. Il restante dei casi troverete una scrittura di colore arancione che rappresenta il codice in Python

Come cambiare la telecamera al tocco di un Oggetto

touch==>and==>scene (set camera)
impostando il nome della telecamera nel campo OB

Come inserire un testo nel game engine

Modalita' non Glsl

-Aggiungere un piano
-Texturizzarlo con una delle font disponibili qui :
http://www.tutorialsforblender3d.com/GameDoc/Text/FontSheet_1.html
-Eseguire l' UvUnwrap in modo che si visualizzi solo la @ come texture.
-Nel pannello Texture Face selezionare TEXT e ALPHA.
-Creare una property per l'oggetto chiamata Text e con valore uguale al testo da inserire.

Modalità Glsl

-Eseguire l'Uv Unwrap come prima e sulla stessa immagine.
-Aggiungere un materiale all'oggetto, e inserire la stessa texture con MapInput impostato su UV
-In MapTo lasciare la funzione COL e applicare ALPHA
-Attivare ZTransp e abbassare l'Alpha al minimo
-Seguire gli stessi passaggi di prima per la property
Input di Testi nel Game Engine (Richiede la lettura dei tutorial precedenti)
-Nel piano in cui e' stato inserito il testo nel tutorial di prima inserire un altra variabile e chiamatela come volete, l'importante e' che sia una BOOLEAN
-Inserite un Sensore Keyboard e in LogToggle inserite il nome della variabile BOOLEAN e in TARGET inserite "Text"
-Ora ogni volta che la property BOOLEAN sara' settata su TRUE digitando sulla tastiera si andra' a cambiare il testo visualizzato. Questo puo' facilmente essere usato per creare sistemi di log-in, ecc...

Creazione di Texture Animate (Come esplosioni, fiamme, ecc...)

-Prima di tutto avrete bisogno di una texture con le varie immagini. Io vi consiglio questo programma per provare : http://www.positech.co.uk/content/explosion/explosiongenerator.html
Andate in Uv/Image Editor ==> View ==> Realtime Properties attivare Tiles e inserire a X quante sono le immagini in orizzontale (in questo caso 4) e in Y quante sono quelle in verticale (ancora 4)
-Attivare ANIM e inserire in END l'ultimo fotogramma dell'animazione e in questo caso e' il 15

(4x4=16 e quindi dallo 0 al 15) e in SPEED inserire la durata di ogni singolo fotogramma
-Uw Unvrappare il piano in modo che circondi una solo delle 16 immagini ora uguali dell'esplosione. Far partire il gioco

Utilizzare file audio OGG Vorbis nel Game Engine

Molto spesso, durante lo sviluppo di applicativi 3D nel Game Engine di blender, si fa uso del settore audio. In questo campo possono essere utilizzati esclusivamente file con estensione .wav i quali sono perfetti (e pensati appositamente) per riprodurre effetti sonori, ma sono poco adatti per le musiche di sottofondo. In modo da ottenere una migliore qualità per quest'ultime, in questo tutorial verrà esposta una semplice via per usare file audio OGG Vorbis. A questo scopo utilizzeremo semplicissimi script python che fanno uso del modulo pygame (http://www.pygame.org/download.shtml). E' quindi necessario istallarlo sulla propria macchina per far si che tutto funzioni correttamente.
Fatto ciò apriamo blender ed entriamo nel suo editor di testo. Inseriamo quindi il seguente codice:

```
from pygame import mixer
mixer.init()
mixer.music.load("C:/music.ogg")
mixer.music.play()
```

Come si può notare, la prima riga importa il mixer del modulo pygame (E' inutile importarlo tutto se l'unica parte che ci serve è questa) mentre la seconda lo esegue. La terza istruzione invece carica e prepara il file .ogg all'esecuzione e infine la quarta lo avvia (Ovviamente è necessario sostituire il percorso C:/music.ogg dell'esempio con quello del vostro file musicale). Chiamiamo lo script appena creato "play.py". Ora è il momento di creare un altro script che fermi l'esecuzione della musica, che chiameremo "halt.py". Niente di più semplice, basta utilizzare l'istruzione "stop" :)

```
from pygame import mixer
mixer.music.stop()
```

Non ci rimane che testare i nostri script :)
Per farlo, selezioniamo la telecamera della nostra scena ed entriamo nei tasti realtime (F4). Qui aggiungiamo due sensors (Keyboard) l'uno attivato mediante tasto P e l'altro con H. In seguito aggiungiamo due controllers entrambi "Python" che puntano rispettivamente agli script "play.py" e "halt.py". Uniamo le due coppie di mattoni logici e il risultato dovrebbe essere come quello mostrato in figura: Fatto ciò, spostiamoci nella cartella del modulo pygame, che varia a seconda del sistema operativo in uso (Su windows è /Python24/Libs/site-packages/pygame) e copiamola nella cartella di Blender (Ad esempio /Programmi/Blender Foundation/Blender/). Rinominiamo il file /Programmi/Blender Foundation/Blender/SDL.dll in SDL.old e sostituiamolo con quello contenuto nella cartella pygame. Adesso l'applicazione dovrebbe funzionare direttamente da blender (Si usino i tasti "P" ed "S" per suonare e fermare il file .ogg). Se si desidera creare un eseguibile è necessario copiare la stessa cartella pygame, il file SDL.dll e python24.dll nella directory dove è contenuto. (NOTA: Ora non è più necessario avere pygame istallato)

Suoni 3d (suoni che diminuiscono o aumentano di intensita' in base alla distanza)

-Importare un suono
-Nei Logic Brick : ALWAYS ==> AND ==> SOUND - Scegliere il suono - Loop Ping Pong Stop
-Nel menu' Scene (F10) nel Sound Block Button attivare 3d SOUND e variare il valore di Scale per decidere la distanza.

Spostare o ruotate un oggetto l'ungo un asse a scelta(x,y,z)

keyboard ==>and==>motion
su keyboard nella riga dove c'e messo key clicciamo su quel quadratino accanto dopo di ciò premiamo un tasto per esempio freccietta avanti
su motion selezioniamo simple motion
ci sono due righe che si chiamano rispettivamente loc rot
Loc sta per spostare un oggetto mentre rot sta per rotazione
nella tabella che vedete sono in ordine rispettivo x ,y ,z

Riavviare/avviare/chiudere il gioco

keyboard ==>and==>game
Su keyboard impostiamo il tasto che volgiamo cliccando su key e nella casella accanto.
Su game possiamo selezionare queste scelte
-start new game ,avviare un nuovo gioco nella riga di sotto ci dobbiamo passare il nome del file da avviare
-restart this game con questa opzione riavviamo il gioco corrente si può usare per esempio quando si fa game over
-quit this game esce dal gioco avviato

Style metal gears solid

creare delle mesh a forma di telecamere e unite anche al cono di visibilità della telecamera e fatele muovere destra e sinsitra segnatevi il tempo del range
per farle muovere sempre le telecamere si fa
-always==>and===>ipo
-nel attuatore ipo mettete il range che vi ho detto di segnarvi
-start e da dove comincia mentre end e dove finisce mettete questi due numeri .
adesso se volete che quando la telecamera o per meglio dire il cono di luce tocchi il nostro personaggio si riavvia il gioco fate così
-selezionate ancora la mesh fatta a forma di telecamera
touch==>and==>game
in game dovete mettere riavvia il nuovo gioco quindi selezioniamo Restart this game

Inserire i pulsanti

-Creiamo l'oggetto per il pulsante
-Con i Logic Brick possiamo creare vari tipi di pulsanti e quindi
Pulsanti sul terreno (Leggere anche inserire i pulsanti)
-Aggiungiamo una Property al pulsante di tipo BOOLEAN
-Inseriamo Collision (Inseriamo in Property il nome di una porperty che dovra' essere data anche al personaggio) ==> And ==>Property , nome della property del pulsante, valore = True
-Ancora Collision (Inseriamo in Property lo stesso nome di prima) , premiamo il tasto INV ==> And ==>Property , nome della property del pulsante, valore = False

Ogni qualvolta il personaggio cammina sul pulsante esso si attiva e da il valore True alla Variabile e quando scende essa ritorna False.

Pulsanti col mouse

-Aggiungiamo una Property al pulsante di tipo BOOLEAN
-Inseriamo due sensori : Mouse, Left Click e Mouse, Mouse Over e li colleghiamo entrambi ad un And ==> Property, Toggle, nome della property del pulsante
Ora ogni volta che si clicca sul pulsante esso cambia il suo valore da True a False

Metodo per le piattaforme

Keyboard1-->AND-->Move
Keyboard2-->AND-->Move
Collision(con la piattaforma)-->AND-->Parent(set parent'Nome oggetto piattaforma')
Keyboard1-->OR-->Parent(Remove parent)
Keyboard2-->^
Importare un oggetto esterno tramite Python

```
import bpy
import Blender
from Blender import *
scn= bpy.data.scenes.active #attiva la scena
lib = bpy.libraries.load('//prova.blend') #carica il documento prova.blend, mi raccomando le due barre se il file che dovete caricare si trova nella stessa cartella del loader
pseudoOb = lib.objects.append('mio_oggetto') #nome dell'oggetto che si trova all'interno della scena che avete caricato
ob = scn.objects.link(pseudoOb) #attiva il link
Blender.Window.RedrawAll() #riaggiorna le viste!
```

Funzione track to Si può usare per fare si che le ruote girano quando si preme un tasto,anche per fare un flipper,un pendolo

La fonte l'ho presa in una domanda che ho fatto molto tempo fa
http://www.blender.it/forum/viewtopic.php?f=13&t=10918&start=10
Always==> AND ==>[Edit Object
Nelle opzioni selezionare Track to, e nel campo OB: metti il nome dell'empty che le ruote devono tracciare....Se la l'empty si muoverà, le ruote si volteranno dal suo lato.

Animare dei cespugli effetto vento

-aggiungete un piano con la texture di un cespuglioppure erba o prato
-create un animazione
-Always==>and==>ipo
mettete i parametri
start cioè la parte da dove iniziare l'animazione
end cioè la parte da dove finisce l'animazione
-Selezionae pimp pong per avere una cosa più realistica se no avete un'animazione discontinua .

Rimbalzi

Un sistema per creare rimbalzi come palle da ping pong o anche oggetti che cadono in acqua e rimangono " a galla ".
-Aprire il poco usato pannello Dynamic (DYN) nel materiale dell'oggetto e cambiare le impostazioni della forza del rimbalzo, la distanza, la frizione, ecc....
-Nell'oggetto che deve rimbalzare, attivare "Do Fh" nel pannello Logic.

Far nascondere/comparire una mesh

Keyboard==>and==>visibility impostiamo per non farlo apparire)
Keyboard==>and==>visibility (impostiamo per farlo comparire)
su keyboard mettiamo un tasto
sull'attuatore possiamo premere dei tasti tra cui
visibile Se attiaviamo questo tasto rendiamo la mesh visibile al GE
invisible SE attiviamo questo tasto rendiamo la mesh NON visibile .

Near

Attiva l'azione solo se una determinata property si trova nel suo raggio d'azione.
-Dare all'oggetto da rilevare una property con un nome qualunque
-Aggiungere un Sensor Near all'oggetto rivelatore, consigliabile un Empty, per funzioni di vario tipo , nel caso di un nemico che rileva a distanza il personaggio e' meglio usare il sensore Ray
-Dare all'oggetto l'attributo Sensor nella fisica (presente solo in blender 2.49)
-Inserire in Property il nome della property data all'oggetto prima
-Inserire in Dist la distanza dalla quale iniziare a rilevare l'oggetto
-Inserire in Reset la distanza alla quale smettere di rilevare l'oggetto

Stati

Gli stati sono uno strumento potente utilizzato per creare i vari stati ,appunto, di un personaggio, come salto,corsa,fermo,nuoto,ecc... senza inserire troppi logic brick.
Selezionando lo State sotto i Controller si puo' vedere quali sono i controller relativi a quello stato.
Esempio
-KEYBOARD W==>AND (State 1 e quindi il primo quadratino)==>MOTION 1.00 (il personaggio si muove premendo W)
-KEYBOARD SHIFT==>AND (State 1) ==>STATE COPY STATO 2 (Secondo quadratino e quindi lo stato 2)
-KEYBOARD W (lo stesso di prima) ==> AND (State 2) ==> MOTION 2.00 (il personaggio si muove piu' veloce)
-KEYBOARD SHIFT INV (se shift non e' premuto) ==> AND (State 2) ==> STATE COPY STATO 1
In questo modo abbiamo creato un sistema di stati in cui se si preme W il personaggio cammina, se invece si tiene premuto SHIFT corre. Cio' era ottenibile anche senza stati ma cio' semplifica molto il lavoro quando si creano personaggi complessi e con molte azioni.
Creare una video-Texture
Premessa:Si può usare per un gioco dove possiamo trovare una tv e lo schermo cambia oppure possiamo usarlo in delle parti che vogliamo mettere un video per far capire a chi gioca il perchè della storia .Serve anche per il giocatore per far ambientare meglio nella storia stessa .Volendo possiamo usare per creare il nuovo cortometraggio in stile Harry Potter per esempio i quadri con le persone dentro che parlano e si muovono
- Creiamo un piano
  - Carichiamo un immagine texture & UV unwrap impostiamo su "map input" UV

- Scrivere uno script in python che permette di mettere quel deteminato frame di quel secondo nella texture

```
import VideoTexture
# -- Gets the Python controller associated with this Python script.
contr = GameLogic.getCurrentController()
# -- Gets the game object associated with this logic brick.
obj = contr.owner
# -- Check if the "video" property has been defined on "GameLogic"
if not hasattr(GameLogic, 'video'):
# -- Get the material that is using our texture
matID = VideoTexture.materialID(obj, 'IMbbb.png')
# -- Create the video texture
GameLogic.video = VideoTexture.Texture(obj, matID)
# -- Get the path to the video file
movie = GameLogic.expandPath('//trailer_400p.ogg')
# -- Load the file
GameLogic.video.source = VideoTexture.VideoFFmpeg(movie)
# -- scale the video
GameLogic.video.source.scale = True
# -- play the video
GameLogic.video.source.play()
-Aggiornare la texture cioè avanzare di qualche secondo per far sembrare tutto più fluido e realistico usando il python.
# -- Check if the "video" property has been defined on "GameLogic"
if hasattr(GameLogic, 'video'):
# -- The video has to be refreshed every frame because
# it is not a background process
GameLogic.video.refresh(True)
```

Keyboard==>Python
impostiamo un tasto su keyboard
e su Python Inseriamo lo script Per caricare il video
Always==>Python
qui inseriamo anche qui lo script ma sta volta quello per aggiornare il frame.
fonte http://www.rozengain.com/blog/2009/06/1 ... me-engine/

Creare un azione combo

keyboard==>and==>motion
keyboard---|^| (collegato all'and di sopra)
il primo e il secondo keyboard possiamo impostare i tasti quelli che vogliamo,Questi due devono essere collegati in un unico and così da usare la tabella della verità dell'And.Il Motion impostiamo una qualsiasi azione per farla muovere. Se noi premiamo questi due tasti che abbiamo impostato farà quella determinata azione che gli abbiamo detto all'attuatore cioè il Motion
la tabella della verità dell'and è questa qui
a |b|Risulta
0 |0| 0
0 |1| 0
1 |0| 0
1 |1| 1

Generare un oggetto quando il personaggio va in conflitto

collision==>and==>edit object
il sensore "collision " appena collide o va in conflitto esegue un azione dell'attuatore.
-l'attuatore "edit object" selezioniamo "add object"
-inseriamo nel campo ob cioè object la mesh che vogliamo che si inserisca nella scena .
-Nel parametro time mettiamo il tempo di vita fino a quando scomprare la nostra mesh generata

Si avvicina il personaggio si attiva una azione

near==>and==>ipo
sul sensore near mettiamo la distanza minima e la massima che la nostra azione sia valida.
Sulla parte ipo inseriamo da dove inizia a dove finisce la nostra animazione

Visualizzare il mouse nel Game Engine

-Create un nuovo empty
-Nei Logic Brick inserite Always ==> Python (Text)
Aprite il Text Editor e nel testo Text inserite le seguenti linee

```
import Rasterizer
Rasterizer.showMouse(1)
```

Mouse Personalizzato
-Create una nuova scena
-Creare una nuova Camera
-Create un nuovo piano e posizionatelo in modo che sia al centro della camera
-Spostate il centro dell'oggetto nell'angolo in alto a sinistra
-Texturizzarlo con l'immagine di un cursore personalizzato
-Tornare nella prima scena
Camera 1 scena : Always==>And>Scene , Add Overlay Scene (nome della seconda scena)
Cursore personalizzato 2 scena : Always,True (il pulsante con i tre puntini in alto) + Mouse, Mouse Over Any (chiamatelo MousePosi) ==>Python , Text
Aggiungete una property Init di tipo Int e di valore 0 al piano.
Nel Text Editor inserire in Text il seguente codice
http://www.blender.it/forum/posting.php?mode=edit&f=13&p=102152

```
import Rasterizer as R
cont = GameLogic.getCurrentController()
own = cont.getOwner()
mPosi = cont.getSensor("MousePosi")
if own.init == 1:
R.setMousePosition(R.getWindowWidth()/2, R.getWindowHeight()/2)
own.init = 0
cursorPosi = mPosi.getRaySource()
own.setPosition(cursorPosi)
```

Ora cambiate il valore di Start della camera nella Scena mouse per rimpicciolire il cursore personalizzato.

Fare un click su un oggetto e farà un azione di nostra scelta
Colleghiamo con lo stesso and i sensori mouse click e mouse over e l'and la colleghiamo ad un azione da noi scelta
mouse over+mouse click==> And==>azione da noi scelta
Fare uno screenshoot della visuale
-selezionare un oggetto preferibilmente camera
-Nei Logic Brick inserite keyboard ==> Python (Text)
Aprite il Text Editor e nel testo Text inserite le seguenti linee

```
import Rasterizer
Rasterizer.makeScreenshot("nome che vogliamo#")
```

Per zoommare come un mirino
dovete prima di tutto fare lo script con i valori che volete esempio:

```
g = GameLogic
c = g.getCurrentController()
o = c.getOwner()
zoom = o.zoom
if zoom == 1:
o.lens = 60.0
if zoom == 2:
o.lens = 100.0
if zoom == 0:
o.lens = 35.0
if zoom == 3:
o.lens = 150.0
if zoom == 4:
o.lens = 200.0
```

mentre i logic brick sono della telecamera attiva e sono i seguenti:
mouse wheelup==>And==>Property-add +1
mouse whelldown==>and==>property-add -1
tutto di seguito riferito a la prop del testo qua sopra in questo caso zoom
diverso da 1==>
diverso da 2==>
diverso da 3==>And==>property-assign 0
diverso da 4==>

diverso da 0==>
always==>python(nome assegnato allo script)

Rendere Invisibile il personaggio:
[Sensore che si vuole]o=o[and]o=o[visibility*]
* = selezionando il pulsante visible l'oggetto è VISIBILE, se NON lo selezioniamo l'oggetto nn si vedrà

il Random (Casuale) si attiva e si disattiva casualmente , in base a un "seme" (seed) che gli si da. Se invece si vuole dare un valore Random a una property si usa l'Actuator Random che un base al tipo di property (Float,Int,Boolean) ha funzioni diverse.

Esempio, Lancio di un dado

Si da a un Empty un property di tipo Int
-Keyboard (Space)==> And ==> Random (Int Uniform), Nome della property, Min=0 , Max=6
Se si vuole invece creare un dado piu' serio la faccenda e' complicata, infatti bisogna creare un rigid body del dado, lanciarlo e poi in base alla faccia puntata verso l'alto con il sensore Ray dare il valore alla variabile.

Shape Key
Le shape key sono invece delle "azioni" che non modificano l'armatura ma la mesh. Esse servono a creare espressioni facciali, ecc... Bisogna fare, Add Shape Key in Editing e apparirà la key Basic, che indica il vostro personaggio normale. Ancora Add Shape Key e poi si modifica la mesh. Nel menù Action si aggiunge una nuova azione che sarà in automatico ShapeAction e cambiando il fotogramma si aumenta il valore della seconda shape (Value). -Keyboard (scegliete voi) ==> And ==> ShapeAction, Start frame = 1 , Finale Frame = ... (quello che avete usato nella Shape Action)

Filtri
I filtri sono dei cambiamenti della visualizzazione del gioco. Il pass indica in quale posto va messo il filtro. Es. Blur , Pass 1 -- Sepia , Pass 2 farà vedere entrambi i filtri, mentre se assegnati allo stesso Pass se ne vedrà solo 1.
Dof (Depth of Field = Profondità di Campo)
Dof (Depth of Field = Profondità di Campo) ed è un filtro Custom.
Bisogna trovare lo script del filtro, inserirla nel Text Editor e poi in Custom Filter inserire il nome del testo. Ecco un file (non mio) in cui ci sono tutti i filtri Custom possibili.
http://2mean.com/s/Indee/HDR2.0.blend.rar
Il treadh da dove l'ho preso.
http://blenderartists.org/forum/showthr ... 482&page=7

Far spostare con il mouse la visuale (usabile nei giochi di guerra o di simulazione)
NEl sito di ize c'e un proggetto che si chiama walk through ed ho trovato gli script
i mattoncini di logica sono
mouse (moviement) ==> python (vertical script)
mouse (moviement) ==> python (horrizontal script)==>motion(Destra)+motion(Sinistra)
gli script sono questi qui

```
#---------------hori-Script--------------------
#----------------------------------------------
#------------------Import--------------------------
from GameLogic import *
from Rasterizer import *
from math import *
#-----------------Initialize----------------------
Cont = getCurrentController()
Own = Cont.getOwner()
Sens = Cont.getSensors()
Sensor = Sens[0]
#-----------------Variable------------------------
Height = getWindowHeight()/2
Width = getWindowWidth()/2
Xpos = Sensor.getXPosition()
Ypos = Sensor.getYPosition()
RightMove = Cont.getActuator("Right")
LeftMove = Cont.getActuator("Left")
#------------------Move--------------------------
#---------Own.move2-is-the-speed-------------------
if (Xpos > Width):
  XDiff = Xpos - Width
RightMove.setDRot(0,(XDiff / Own.move2),0,1)
addActiveActuator(RightMove,1)
if (Xpos < Width):
 XDiff = Xpos - Width
LeftMove.setDRot(0,(XDiff / Own.move2),0,1)
addActiveActuator(LeftMove,1)
addActiveActuator(LeftMove,0)
addActiveActuator(RightMove,0)
setMousePosition (Width, Ypos)
#------------------END---------------------------
```

Mentre vertical script è questo qui
Codice:

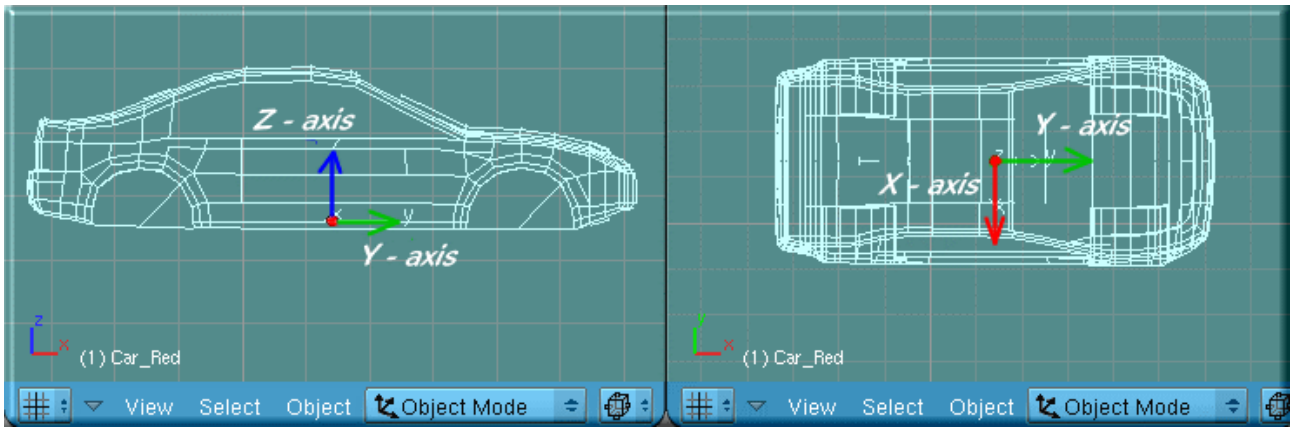
```
#---------------Vertical-Script--------------------
#----------------------------------------------
#------------------Import--------------------------
from GameLogic import *
from Rasterizer import *
from math import *
#-----------------Initialize----------------------
Cont = getCurrentController()
```

```
Own = Cont.getOwner()
Sens = Cont.getSensors()
Scene = getCurrentScene()
Node1 = Scene.getObjectList()["OBNode_1"]
Node2 = Scene.getObjectList()["OBNode_2"]
Sensor = Sens[0]
#-----------------Variable-------------------------
Height = getWindowHeight()/2
Width = getWindowWidth()/2
Xpos = Sensor.getXPosition()
Ypos = Sensor.getYPosition()
#------------------Move---------------------------
#----------Own.move-is-the-speed------------------
if (Ypos < Height):
  YDiff = Height - Ypos
Own.frame = 25 + float(YDiff * Own.move / Height)
if (Ypos > Height):
  YDiff = Ypos - Height
Own.frame = 25 - float(YDiff * Own.move / Height)
Node1.frame = Own.frame
Node2.frame = Own.frame
#------------------END----------------------------
```

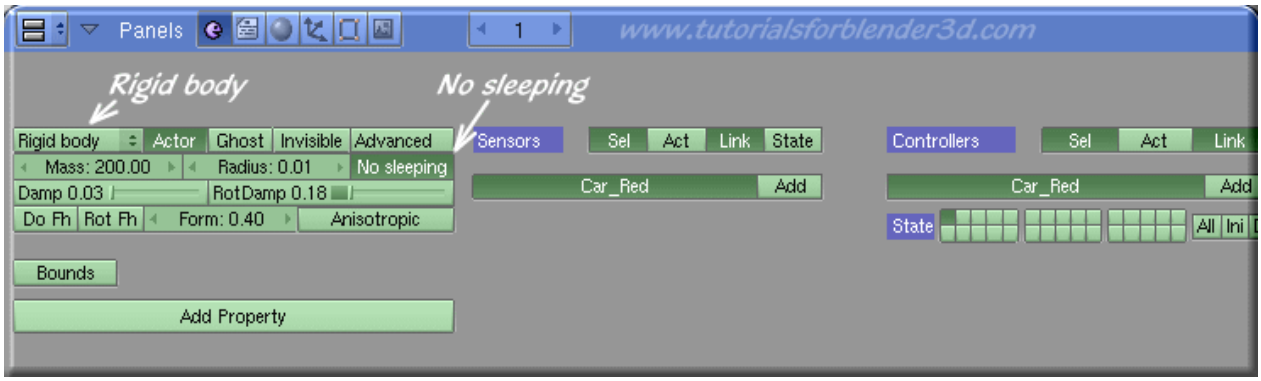
TUTORIAL
COME CREARE UNA MACCHINA NEL GAME ENGINE
(testo tradotto)
http://www.tutorialsforblender3d.com/Game_Engine/Vehicle/Vehicle_1.html

Per fare la nostra macchina e usarla nei nostri giochi dovete fare questi passi.
1) Modellare una macchina e posizionare il “puntino” al centro della macchina .Adesso dobbiamo vedere se vogliamo vedere il nostro modello con i fili ,cioè quelle linee che compongono il nostro disegno .E' consigliabile usarlo perchè così ci accorgiamo subito se ci sono normali invertite .
-ecco un immagine di come ho impostato io nel mio progetto

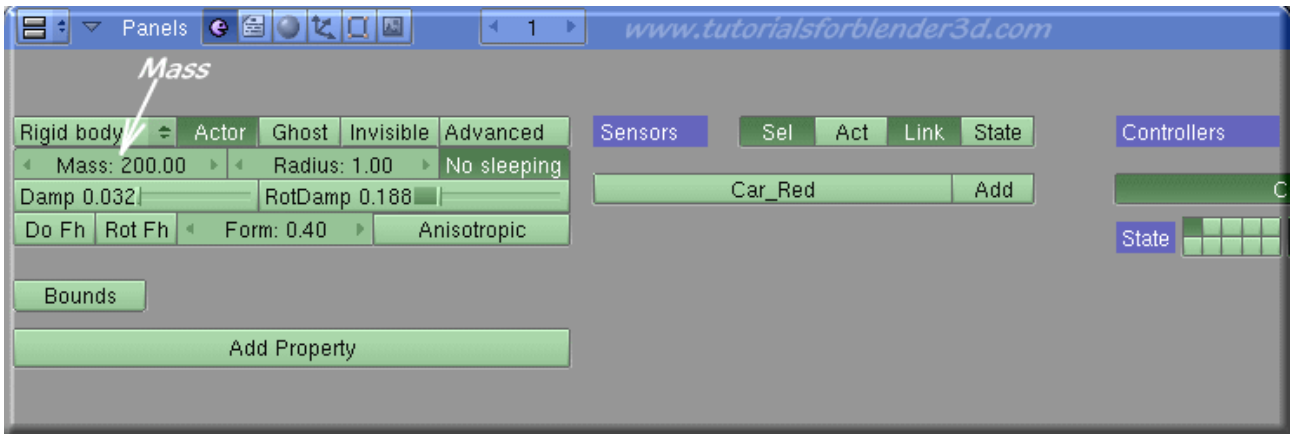
2)impostiamo gli assi premettiamo che gli assi sono rispettivamente
L'asse Y è per la lunghezza .
L'asse X è per larghezza .
L'asse Z è per l'altezza .



3)impostiamo la fisica della macchina quindi clicchiamo su rigid body e clicchiamo su No sleeping

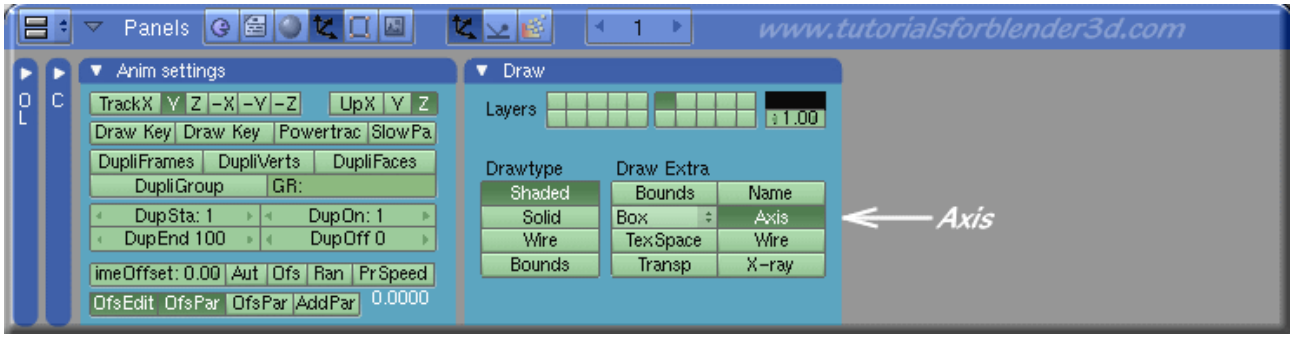


4)impostiamo la massa della nostra macchina
pesatura della macchina l'intervallo va da : 0.01 a 10.000.,00

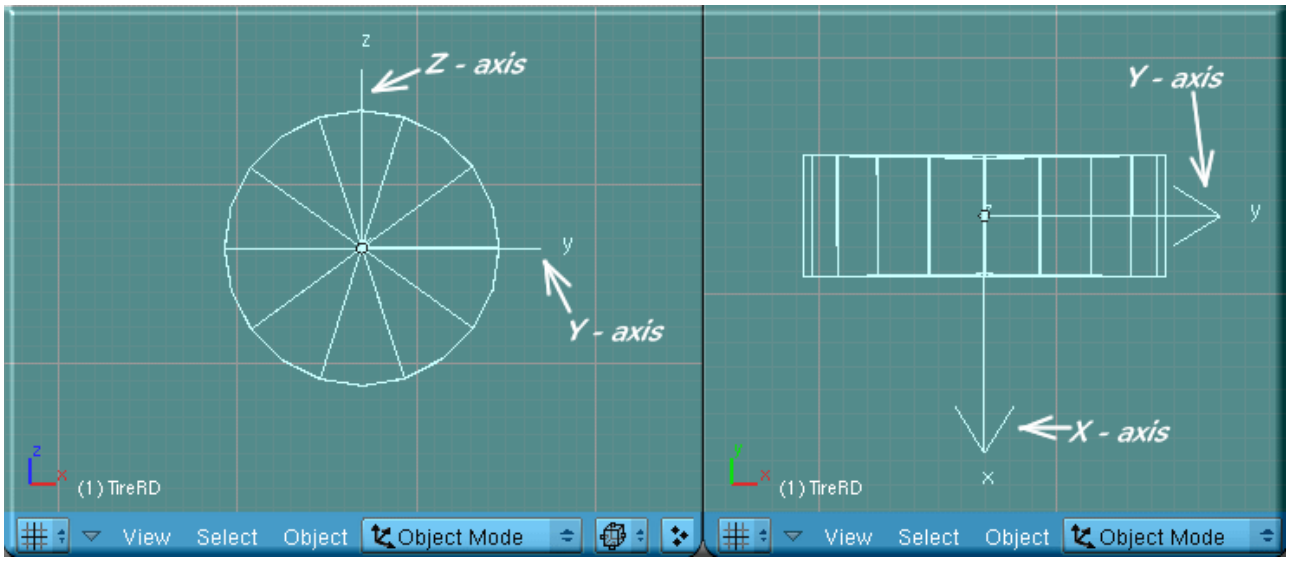


5) Possiamo impostare la sagoma del nostro veicolo ,è disponibile la sagoma “scatola” che è la più semplice da gestire ed infine abbiamo triangolo delle maglie che è molto complesso da gestire .

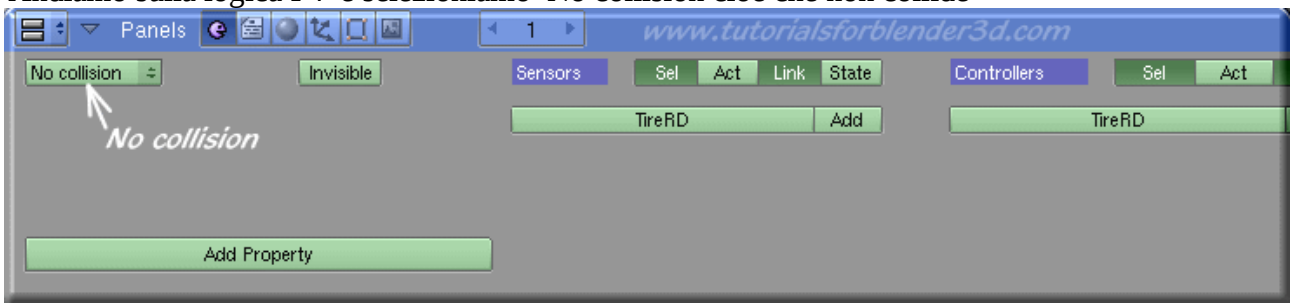
6)Impostiamo le ruote ,selezionate le vostre ruote che avete precedentemente modellato . Premiamo f7 e andiamo sulla scheda draw ,clicchiamo su axis cioè assi .



6) Mettiamo la nostra empy in modo che l'angolo coincide con quello della ruota se così non fosse quando girerà la rotazione non sarà uniforme .



Andiamo sulla logica F4 e selezioniamo No collision cioè che non collide



adesso scaricate questi moduli Python sul vosro frullatore :)
Il nome dei pneumatici sono, tires Tire_FD, Tire_FP, Tire_RD and Tire_RP
Ho chiamato lamia stanca Tire_FD, Tire_FP, Tire_RD e Tire_RP

LEGENDA
F=FRONT FRONTALE
R=RETRY RETRO
P=PASSEGGERO
D=DRIVER PILOTA
per esempio se trovate FP vuol dire che è FRONTALE ed è dalla parte del PILOTA

Crichiamo questi 3 script Python

```
##############################################
#
#   CarSetup.py  Blender 2.48
#
#   tutorial can be found at
#
#   www.tutorialsforblender3d.com
#
##############################################


def main():

        # get car object
        carObj = Car_Object()

        # create a vehicle constraint ID
        vehicleID = Car_Constraint(carObj)

        # get tire objects
        tireObj = Tire_Objects()

        # tire positions
        tirePos = Tire_Positions()

        # tire radius
        tireRadius = Tire_Radius()
```

```
	# tire suspension height
	tireSuspension = Tire_Suspension()

	# tire suspension angle
	tireSuspensionAngle = Tire_SuspensionAngle()

	# tire axis attached to frame
	tireAxis = Tire_Axis()

	# tire has steering?
	tireSteering = Tire_Steering()

	# add wheels to car
	Add_Tires(vehicleID, tireObj, tirePos, tireSuspensionAngle, tireAxis, tireSuspension,
tireRadius, tireSteering)

############################   Get Car

def Car_Object():

	# get current controller
	controller = GameLogic.getCurrentController()

	# get car the controller is attached to
	carObj = controller.getOwner()

	return carObj

############################   Vehicle Constraint

def Car_Constraint(carObj):

	# import PhysicsConstraints
	import PhysicsConstraints

	# get physics ID
	car_PhysicsID = carObj.getPhysicsId()

	# create a vehicle constraint
	vehicle_Constraint = PhysicsConstraints.createConstraint(car_PhysicsID, 0, 11)

	# get the constraint ID
	constraint_ID = vehicle_Constraint.getConstraintId()

	# save constraint ID as an object variable
	carObj.constraint_ID = constraint_ID

	# get the vehicle constraint ID
	vehicleID = PhysicsConstraints.getVehicleConstraint(constraint_ID)


	return vehicleID
```

```
############################   Tire names

def Tire_Objects():

        # tire names
        frontDriver    = "TireFD"  # front driver's tire
        frontPassenger = "TireFP"  # front passenger's tire
        rearDriver     = "TireRD"  # rear driver's tire
        rearPassenger  = "TireRP"  # rear passenger's tire

        # get current scene
        scene = GameLogic.getCurrentScene()

        # get list of objects in scene
        objList = scene.getObjectList()

        # tire Name
        tire_0 = objList["OB" + frontDriver]    # front driver's tire
        tire_1 = objList["OB" + frontPassenger] # front passenger's tire
        tire_2 = objList["OB" + rearDriver]     # rear driver's tire
        tire_3 = objList["OB" + rearPassenger]  # rear passenger's tire

        return (tire_0, tire_1, tire_2, tire_3)

############################   Tire positions

def Tire_Positions():

        # tire position
        tire_0_Pos = [ -2.0,   3.0,  0.0]  # front driver's tire
        tire_1_Pos = [  2.0,   3.0,  0.0]  # front passenger's tire
        tire_2_Pos = [ -2.0,  -3.0,  0.0]  # rear driver's tire
        tire_3_Pos = [  2.0,  -3.0,  0.0]  # rear passenger's tire

        return (tire_0_Pos, tire_1_Pos, tire_2_Pos, tire_3_Pos)

############################   Tire radius

def Tire_Radius():

        # tire radius
        tire_0_Radius = 0.75   # front driver's tire
        tire_1_Radius = 0.75   # front passenger's tire
        tire_2_Radius = 0.75   # rear driver's tire
        tire_3_Radius = 0.75   # rear passenger's tire

        return (tire_0_Radius, tire_1_Radius, tire_2_Radius, tire_3_Radius)

############################   Tire suspension

def Tire_Suspension():
```

```
	# tire suspension height
	tire_0_suspensionHeight = 0.2   # front driver's tire
	tire_1_suspensionHeight = 0.2   # front passenger's tire
	tire_2_suspensionHeight = 0.2   # rear driver's tire
	tire_3_suspensionHeight = 0.2   # rear passenger's tire

	return (tire_0_suspensionHeight, tire_1_suspensionHeight, tire_2_suspensionHeight,
tire_3_suspensionHeight)

############################   Tire suspension angle

def Tire_SuspensionAngle():

	# suspension angle from car object center
	tire_0_suspensionAngle = [ 0.0, 0.0, -1.0]   # front driver's tire
	tire_1_suspensionAngle = [ 0.0, 0.0, -1.0]   # front passenger's tire
	tire_2_suspensionAngle = [ 0.0, 0.0, -1.0]   # rear driver's tire
	tire_3_suspensionAngle = [ 0.0, 0.0, -1.0]   # rear passenger's tire

	return (tire_0_suspensionAngle, tire_1_suspensionAngle, tire_2_suspensionAngle,
tire_3_suspensionAngle)

############################   Tire axis

def Tire_Axis():

	# tire axis attached to axle
	tire_0_Axis = [ -1.0, 0.0, 0.0]   # front driver's tire
	tire_1_Axis = [ -1.0, 0.0, 0.0]   # front passenger's tire
	tire_2_Axis = [ -1.0, 0.0, 0.0]   # rear driver's tire
	tire_3_Axis = [ -1.0, 0.0, 0.0]   # rear passenger's tire

	return (tire_0_Axis, tire_1_Axis, tire_2_Axis, tire_3_Axis)


############################   Tire steering

def Tire_Steering():

	# tire has steering
	tire_0_Steering = True   # front driver's tire
	tire_1_Steering = True   # front passenger's tire
	tire_2_Steering = False   # rear driver's tire
	tire_3_Steering = False   # rear passenger's tire

	return (tire_0_Steering, tire_1_Steering, tire_2_Steering, tire_3_Steering)


############################   add tires

def Add_Tires(vehicleID, tireObj, tirePos, tireSuspensionAngle, tireAxis, tireSuspension,
```

```
tireRadius, tireSteering):

        # loop through variables:  add wheels
        for tire in range(0,4):

                obj = tireObj[tire]
                pos = tirePos[tire]
                suspensionAngle = tireSuspensionAngle[tire]
                axis = tireAxis[tire]
                suspension = tireSuspension[tire]
                radius = tireRadius[tire]
                steering = tireSteering[tire]

                # Add front driver tire
                vehicleID.addWheel( obj, pos, suspensionAngle, axis,
                                        suspension,  radius, steering )


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

# run program
main()
```

secondo script

```
##############################################
#
#   Powertrain.py  Blender 2.48
#
#   tutorial can be found at
#
#   www.tutorialsforblender3d.com
#
##############################################


# Main Program
def main():

        # get the current controller
        controller = GameLogic.getCurrentController()

        # get vehicle constraint ID
        vehicleID = ConstraintID(controller)

        # brakes
        brakes = Brakes(vehicleID, controller)

        # gas & reverse
        Power( vehicleID, controller, brakes)
```

```
        # steering
        Steering(vehicleID, controller)


############################################################  Vehicle ID

# get vehicle constraint ID
def ConstraintID(controller):

        # import PhysicsConstraints module
        import PhysicsConstraints

        # get car the controller is attached to
        carObj = controller.getOwner()

        # get saved constraint ID
        constraint_ID = carObj.constraint_ID

        # get the vehicle constraint ID
        vehicleID = PhysicsConstraints.getVehicleConstraint(constraint_ID)

        return vehicleID

############################################################  Brakes

def Brakes(vehicleID, controller):

        # set braking amount
        brakeAmount = 40.0  # front and back brakes
        ebrakeAmount = 100.0          # back brakes only

        # get sensors
        reverse = controller.getSensor("Reverse")              # sensor named "Reverse"
        brake = controller.getSensor("Brake")                          # sensor named "Brake
        emergency = controller.getSensor("EBrake")                  # sensor named "EBrake"

        # emergency brakes
        if emergency.isPositive() == True:

                front_Brake = 0.0
                back_Brake = ebrakeAmount
                brakes = True

        # brake
        elif brake.isPositive() == True and reverse.isPositive() == False:

                front_Brake = brakeAmount
                back_Brake = brakeAmount
                brakes = True

        # no brakes
```

```
        else:

                front_Brake = 0.0
                back_Brake = 0.0
                brakes = False

        # brakes
        vehicleID.applyBraking( front_Brake, 0)
        vehicleID.applyBraking( front_Brake, 1)
        vehicleID.applyBraking( back_Brake, 2)
        vehicleID.applyBraking( back_Brake, 3)

        return brakes

############################################  Gas & Reverse

# gas and reverse
def Power( vehicleID, controller, brakes):

        # set power amounts
        reversePower = 200.0
        gasPower = 800.0

        # get power sensors
        gas = controller.getSensor("Gas")                       # sensor named "Gas"
        reverse = controller.getSensor("Reverse")       # sensor named "Reverse"

        # brakes
        if brakes == True:

                power = 0.0

        # reverse
        elif reverse.isPositive() == True:

                power = reversePower

        # gas pedal
        elif gas.isPositive() == True:

                power = -gasPower

        # no gas and no reverse
        else:

                power = 0.0

        # apply power
        vehicleID.applyEngineForce( power, 0)
        vehicleID.applyEngineForce( power, 1)
        vehicleID.applyEngineForce( power, 2)
        vehicleID.applyEngineForce( power, 3)
```

```
################################################## Steering

def Steering( vehicleID, controller):

        # set turn amount
        turn = 0.3


        # get steering sensors
        steerLeft = controller.getSensor("Left")            # sensor named "Left"
        steerRight = controller.getSensor("Right")          # sensor named "Right"

        # turn left
        if steerLeft.isPositive() == True:

                turn = turn

        # turn right
        elif steerRight.isPositive() == True:

                turn = -turn

        # go straight
        else:
                turn = 0.0

        # steer with front tires only
        vehicleID.setSteeringValue(turn,0)
        vehicleID.setSteeringValue(turn,1)


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


# run main program
main()
```

terzo ed ultimo script Python

```
##########################################
#
#   Suspension.py  Blender 2.48
#
#   tutorial can be found at
#
#   www.tutorialsforblender3d.com
#
##########################################
```

```
def main():

    # get the current controller
    controller = GameLogic.getCurrentController()

    # get vehicle constraint ID
    vehicleID = ConstraintID(controller)

    # set tire grip
    Tire_Grip(vehicleID)

    # set suspension compression
    Suspension_Compression(vehicleID)

    # set suspension damping
    Suspension_Damping(vehicleID)

    # set suspension stiffness
    Suspension_Stiffness(vehicleID)

    # set roll influence
    Roll_Influence(vehicleID)

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

    # get vehicle constraint ID
    def ConstraintID(controller):

        # import PhysicsConstraints module
        import PhysicsConstraints

        # get car the controller is attached to
        carObj = controller.getOwner()

        # get saved constraint ID
        constraint_ID = carObj.constraint_ID

        # get the vehicle constraint ID
        vehicleID = PhysicsConstraints.getVehicleConstraint(constraint_ID)

        return vehicleID


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

        # set tire grip
        def Tire_Grip(vehicleID):

            grip_0 = 30.0     # front driver's tire
            grip_1 = 30.0     # front passenger's tire
```

```
    grip_2 = 30.0     # rear driver's tire
    grip_3 = 30.0     # rear passenger's tire

    vehicleID.setTyreFriction(grip_0, 0)  # front driver's tire
    vehicleID.setTyreFriction(grip_1, 1)  # front passenger's tire
    vehicleID.setTyreFriction(grip_2, 2)  # rear driver's tire
    vehicleID.setTyreFriction(grip_3, 3)  # rear passenger's tire

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

    # set suspendion compression
    def Suspension_Compression(vehicleID):

        compression_0 = 6.0     # front driver's tire
        compression_1 = 6.0     # front passenger's tire
        compression_2 = 6.0     # rear driver's tire
        compression_3 = 6.0     # rear passenger's tire

    vehicleID.setSuspensionCompression(compression_0, 0)  # front driver's tire
    vehicleID.setSuspensionCompression(compression_1, 1)  # front passenger's tire
    vehicleID.setSuspensionCompression(compression_2, 2)  # rear driver's tire
    vehicleID.setSuspensionCompression(compression_3, 3)  # rear passenger's tire


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

    # set suspension damping
    def Suspension_Damping(vehicleID):

        damp_0 = 5.0     # front driver's tire
        damp_1 = 5.0     # front passenger's tire
        damp_2 = 5.0     # rear driver's tire
        damp_3 = 5.0     # rear passenger's tire

    vehicleID.setSuspensionDamping(damp_0, 0)  # front driver's tire
    vehicleID.setSuspensionDamping(damp_1, 1)  # front passenger's tire
    vehicleID.setSuspensionDamping(damp_2, 2)  # rear driver's tire
    vehicleID.setSuspensionDamping(damp_3, 3)  # rear passenger's tire


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

    # set suspension stiffness
    def Suspension_Stiffness(vehicleID):

        stiffness_0 = 12.5     # front driver's tire
        stiffness_1 = 12.5     # front passenger's tire
        stiffness_2 = 12.5     # rear driver's tire
        stiffness_3 = 12.5     # rear passenger's tire

    vehicleID.setSuspensionStiffness(stiffness_0, 0)  # front driver's tire
    vehicleID.setSuspensionStiffness(stiffness_1, 1)  # front passenger's tire
```

```
vehicleID.setSuspensionStiffness(stiffness_2, 2)  # rear driver's tire
vehicleID.setSuspensionStiffness(stiffness_3, 3)  # rear passenger's tire

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

# set roll influence
def Roll_Influence(vehicleID):

    roll_0 = 0.06     # front driver's tire
    roll_1 = 0.06     # front passenger's tire
    roll_2 = 0.06     # rear driver's tire
    roll_3 = 0.06     # rear passenger's tire

    vehicleID.setRollInfluence( roll_0, 0)  # front driver's tire
    vehicleID.setRollInfluence( roll_1, 1)  # front passenger's tire
    vehicleID.setRollInfluence( roll_2, 2)  # rear driver's tire
    vehicleID.setRollInfluence( roll_3, 3)  # rear passenger's tire


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

# run main program
main()
```

adesso salviamoli in un editor di testo e importiamoli su Blender adesso

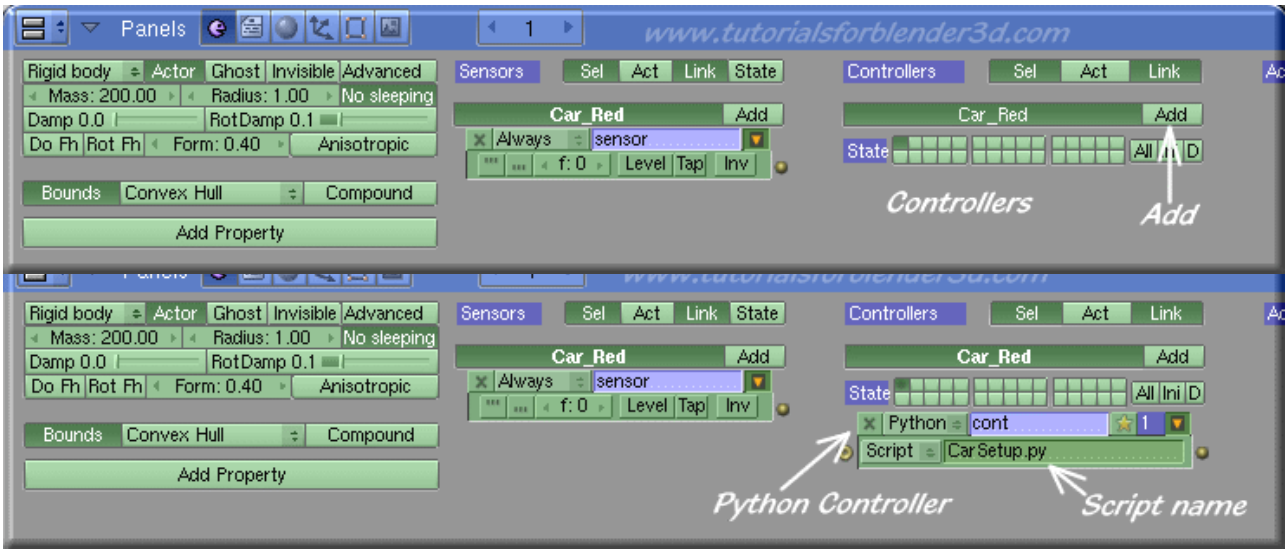
```
############################ Tire suspension angle

def Tire_SuspensionAngle():

# suspension angle from car object center
tire_0_suspensionAngle = [ 0.0, 0.0, -1.0] # front driver's tire
tire_1_suspensionAngle = [ 0.0, 0.0, -1.0] # front passenger's tire
tire_2_suspensionAngle = [ 0.0, 0.0, -1.0] # rear driver's tire
tire_3_suspensionAngle = [ 0.0, 0.0, -1.0] # rear passenger's tire

return (tire_0_suspensionAngle, tire_1_suspensionAngle, tire_2_suspensionAngle, tire_3_suspensionAngle)
```

17) Eguiamo questi semplici passaggi .
aggiungiamo il sensore Always e lo colleghiamo su Python

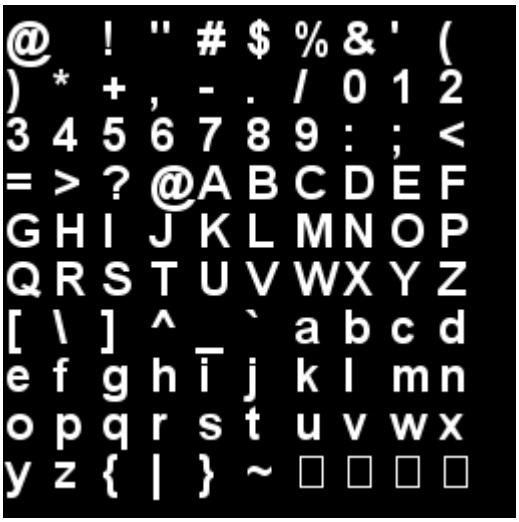


Collegare il sensore sempre al controllore di Python
Collegare il sensore Always al controllore AND
Collegare il sensore Always al controllore AND
Collegare il controllore AND al attuatore State
Collegare il controllore And al attuatore Stato

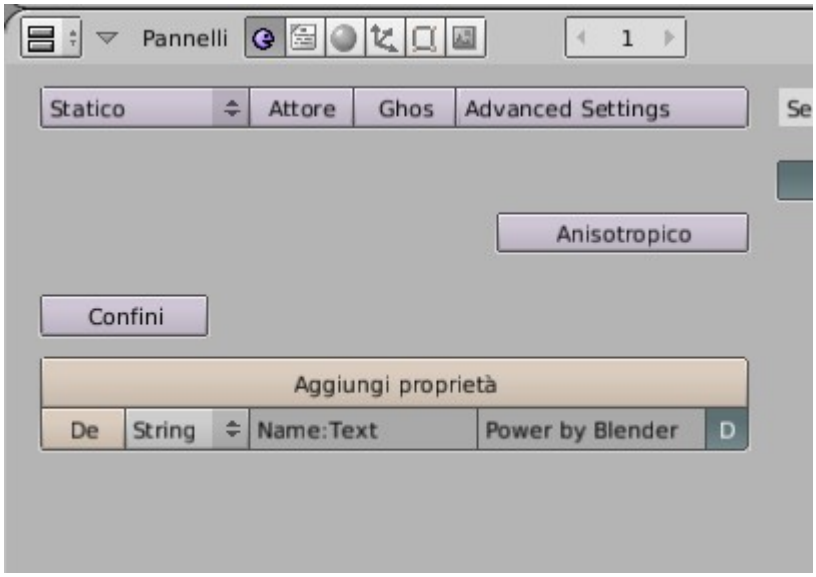
Adesso premete P per far partire il gioco

come inserire testo variabili

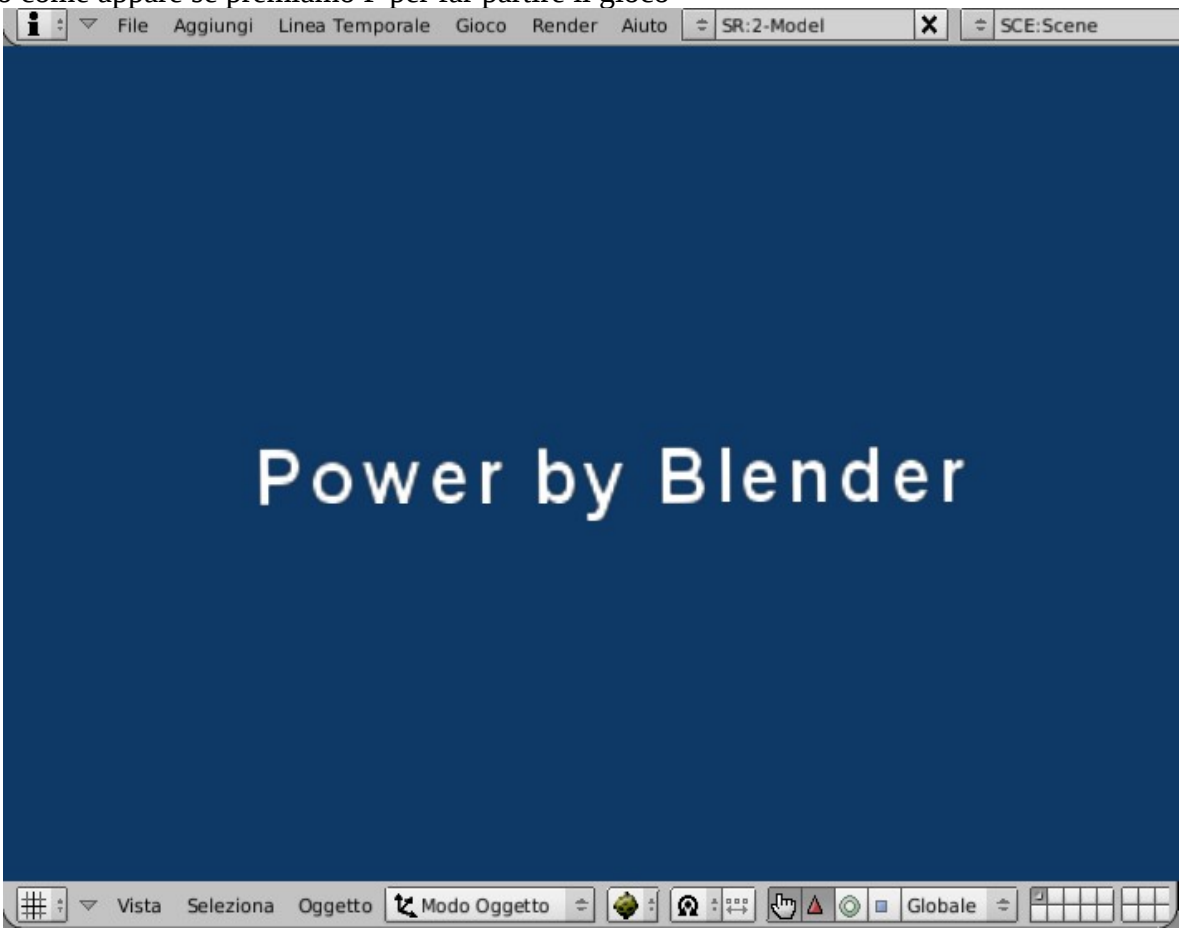
Per inserire un testo nei nostri giochi basta inserire un piano parallelo alla telecamera con il carattere chiocciola con l'editor uv-map e mettete questa texture o una simile .



Nelle variabili globali fate aggiungi proprietà e selezionate il tipo “stringa” o “string” alla sua destra inseriamo il nome della mesh per esempio text e come messagio mettiamo l parola che volgiamo far comparire allo schermo per esempio Power by Blender .
Ecco come dovrebbe essere fatto



Ecco come appare se premiamo P per far partire il gioco



Ringraziamenti

Questo manuale è stato reso grazie al sito www.blender.it che ci ha ospitato a parlare del software Blender in particolare hanno collaborato

Wildlux
BJ
Ize design
Giotherobot
Wox76
Luca_il-vec
Anonymus44
Yoshi
Bruce965

Se non siete soddisfatti di questa guida potete andare su internet e scrivete il vostro codice in Python Blender ,sono sicuro che sarete molto più soddisfatti di voi stessi.
Se avete tempo per favore posta delle cose che hai scoperto e pubblicale nel sito di Blender.it se puoi ,grazie per l'attenzione .

Ecco le guide Python Blender

Per la versione 2.49	http://www.blender.org/documentation/249PythonDoc/
Per la versione 2.48	http://www.blender.org/documentation/248PythonDoc/
Per la versione 2.47	http://www.blender.org/documentation/247PythonDoc/

Se trovate errori per favore diteceli al forum di blender.it grazie:)