```
##APENDICE D MODELO 3D DEL AVATAR DE REMO

import direct.directbase.DirectStart
from pandac.PandaModules import*
from direct.task import Task
from direct.actor import Actor

#actor importa el avatar3d
#Task me permite manejar tareas

from direct.interval.IntervalGlobal import*
from direct.showbase import DirectObject
from serial_bluetooth import *
from direct.gui.OnscreenText import OnscreenText
from math import *


## base.oobe()
class human_class(DirectObject.DirectObject):
    def __init__ (self):

        self.archivo = open('record.txt', 'w')

        #importacion del avatar *.egg al motor de juegos.
        self.human = Actor.Actor('hombre_base2')
        self.human.reparentTo(render)
        base.camera.reparentTo(self.human)
        base.camera.setY(-16)
        base.camera.setZ(3)
        base.disableMouse()

        self.accept("q",self.cambiaFun,[1])
        self.accept("a",self.cambiaFun,[-1])
        self.accept("z",self.reset)

        self.accept("w",self.selectPart,[0])
        self.accept("e",self.selectPart,[1])
        self.accept("r",self.selectPart,[2])
        self.accept("s",self.selectPart,[4])
        self.accept("d",self.selectPart,[5])
        self.accept("f",self.selectPart,[6])
        self.accept("x",self.selectPart,[8])
        self.accept("c",self.selectPart,[9])
        self.accept("v",self.selectPart,[10])

        self.creaProxy()

        #asignacion del avatar al nodo principal del motor grafico

        self.human.reparentTo(render)

        #seccion de control de articulacion del avatar

        self.Pelvis          = self.human.controlJoint(None,"modelRoot","Pelvis")

        self.Hip_L           = self.human.controlJoint(None,"modelRoot","Hip_L")

        self.UpperLeg_L      =
self.human.controlJoint(None,"modelRoot","UpperLeg_L")
        self.LowerLeg_L      =
self.human.controlJoint(None,"modelRoot","LowerLeg_L")
        self.Foot_L          = self.human.controlJoint(None,"modelRoot","Foot_L")

        self.Hip_R           = self.human.controlJoint(None,"modelRoot","Hip_R")

        self.UpperLeg_R      =
self.human.controlJoint(None,"modelRoot","UpperLeg_R")
```

```
        self.LowerLeg_R      =
self.human.controlJoint(None,"modelRoot","LowerLeg_R")
        self.Foot_R          = self.human.controlJoint(None,"modelRoot","Foot_R")

        self.Back            = self.human.controlJoint(None,"modelRoot","Back")

        self.Neck            = self.human.controlJoint(None,"modelRoot","Neck")

        self.Head            = self.human.controlJoint(None,"modelRoot","Head")

        self.Shoulder_L      =
self.human.controlJoint(None,"modelRoot","Shoulder_L")
        self.UpperArm_L      =
self.human.controlJoint(None,"modelRoot","UpperArm_L")
        self.LowerArm_L      =
self.human.controlJoint(None,"modelRoot","LowerArm_L")
        self.Palm_L          = self.human.controlJoint(None,"modelRoot","Palm_L")
        self.Fingers_L       =
self.human.controlJoint(None,"modelRoot","Fingers_L")
        self.Thumb_L         = self.human.controlJoint(None,"modelRoot","Thumb_L")
        self.Shoulder_R      =
self.human.controlJoint(None,"modelRoot","Shoulder_R")
        self.UpperArm_R      =
self.human.controlJoint(None,"modelRoot","UpperArm_R")
        self.LowerArm_R      =
self.human.controlJoint(None,"modelRoot","LowerArm_R")
        self.Palm_R          = self.human.controlJoint(None,"modelRoot","Palm_R")
        self.Fingers_R       =
self.human.controlJoint(None,"modelRoot","Fingers_R")
        self.Thumb_R         = self.human.controlJoint(None,"modelRoot","Thumb_R")


        self.lista = []

        self.parteEscogida = 0

        self.lista.append(self.Palm_R)
        self.lista.append(self.LowerArm_R)
        self.lista.append(self.UpperArm_R)
        self.lista.append(self.Shoulder_R)

        self.lista.append(self.Palm_L)
        self.lista.append(self.LowerArm_L)
        self.lista.append(self.UpperArm_L)
        self.lista.append(self.Shoulder_L)

        self.lista.append(self.Head)
        self.lista.append(self.Neck)
        self.lista.append(self.Back)


        self.lista.append(self.Fingers_R)
        self.lista.append(self.Fingers_L)
        self.lista.append(self.Thumb_R)
        self.lista.append(self.Thumb_L)

        self.lista.append(self.Pelvis)
        self.lista.append(self.Hip_L)
        self.lista.append(self.Foot_L)
        self.lista.append(self.Hip_R  )
        self.lista.append(self.Foot_R)


        self.datos = []
        self.accept("Datos Serial", self.actualizaDatos)

        self.Wx = 0
        self.Wy = 0
        self.Wz = 0

        taskMgr.add(self.update, "Update human")


    def creaProxy(self):
        self.esfera = loader.loadModel('esfera')
        self.esfera.reparentTo(render)
        self.esfera.setTransparency(True)
        self.esfera.setTransparency(TransparencyAttrib.MAlpha)
        self.esfera.setScale(0.3)


        self.listaExposeJoints = []


self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Palm_R"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","LowerArm_R")
)

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","UpperArm_R")
)

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Shoulder_R")
)

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Palm_L"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","LowerArm_L")
)

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","UpperArm_L")
)

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Shoulder_L")
)

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Head"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Neck"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Back"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Fingers_R"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Fingers_L"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Thumb_R"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Thumb_L"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Pelvis"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Hip_L"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Foot_L"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Hip_R"))

self.listaExposeJoints.append(self.human.exposeJoint(None,"modelRoot","Foot_R"))


    def actualizaDatos(self, dat):
        textObject = OnscreenText(text = 'my text string', pos = (-0.5, 0.02),
scale = 0.07)
```

```
        self.datos = dat;

    def cambiaFun(self, num):

        self.parteEscogida += num
        self.parteEscogida = self.parteEscogida%len(self.lista)

    def selectPart(self, num):

        self.parteEscogida = num

    def filtro(self):

        self.Wx = self.datos[3] - 13.5
        if self.Wx > 5 and self.Wx < -5:
            self.Wx = 0

        self.Wy = self.datos[4] + 13
        if self.Wy > 5 and self.Wy < -5:
            self.Wy = 0

        self.Wz = self.datos[5] - 15
        if self.Wz > 5 and self.Wz < -5:
            self.Wz = 0


        ## print self.Wx, self.Wy, self.Wz

        self.Wx *= 0.0036
        self.Wy *= 0.0036
        self.Wz *= 0.0036


    def reset(self):
        self.lista[self.parteEscogida].setHpr(0,0,0)

    def update(self, task):

        self.filtro()
        self.lista[self.parteEscogida].setH( self.lista[self.parteEscogida].getH()
+ self.Wx)
        self.lista[self.parteEscogida].setP( self.lista[self.parteEscogida].getP()
+ self.Wz)
        self.lista[self.parteEscogida].setR( self.lista[self.parteEscogida].getR()
- self.Wy)

        self.esfera.setPos(self.listaExposeJoints[self.parteEscogida].getPos())
        self.esfera.setH(self.esfera.getH() + 20)

        frec = 1
        self.esfera.setScale(0.2+0.1*sin(task.time*frec*2*3.14))


        self.archivo.write(str(self.lista[self.parteEscogida].getH())+' '+
str(self.lista[self.parteEscogida].getP())+' '+
str(self.lista[self.parteEscogida].getR())+'\n')


        print self.lista[self.parteEscogida].getHpr()

        return Task.cont


human = human_class()

## base.oobe()
run()
```