```
from direct.showbase.ShowBase import ShowBase
from direct.task import Task
from direct.actor.Actor import Actor
from direct.interval.IntervalGlobal import Sequence
from panda3d.core import Point3
from pandac.PandaModules import WindowProperties
import math
import sys


class MyApp(ShowBase):

    def __init__(self):
        ShowBase.__init__(self)

        #Disable mouse controls
        self.disableMouse()

        #Load enviromnment model
        self.scene = self.loader.loadModel("models/environment")

        #Reparent model top renderer
        self.scene.reparentTo(self.render)

        #Apply scale and position transforms on the model
        self.scene.setScale(0.25, 0.25, 0.25)
        self.scene.setPos(-8, 42, 0)

        #Add task spinCameraTask procedure to task manager
        self.taskMgr.add(self.setCameraTask, "setCameraTask")
        self.taskMgr.add(self.keyInput, "keyInput")
        self.taskMgr.add(self.moveCharacter, "movecharacter")
        self.taskMgr.add(self.lookCharacter, "lookCharacter")



        #Positions of character/camera
        self.xpos = 0
        self.ypos = 0
        self.zpos = 10

        self.angleH = 0
        self.angleP = 0

        self.xInterval = 0.0
        self.yInterval = 0.0

        #Speed at which the character/camera moves
        self.movementInterval = 1

        self.lookInterval = 2

        #Dictionary of key states
        self.keys = {"w" : False,
                     "s" : False,
                     "a" : False,
                     "d" : False,
                     "arrow_right" : False,
                     "arrow_left": False}

        #Configuring cursor settings
        wp = WindowProperties()
        wp.setMouseMode(WindowProperties.MRelative)
        wp.setCursorHidden(True)
```

```
    self.win.requestProperties(wp)

    self.mw = self.mouseWatcherNode




#Define a procedure to move the camera
def setCameraTask(self, task):
    self.camera.setPos(self.xpos, self.ypos, self.zpos)
    self.camera.setHpr(self.angleH, self.angleP, 0)

    return Task.cont

def setWToTrue(self):
    self.keys["w"] = True

def setWToFalse(self):
    self.keys["w"] = False



def setSToTrue(self):
    self.keys["s"] = True

def setSToFalse(self):
    self.keys["s"] = False



def setAToTrue(self):
    self.keys["a"] = True

def setAToFalse(self):
    self.keys["a"] = False




def setDToTrue(self):
    self.keys["d"] = True

def setDToFalse(self):
    self.keys["d"] = False




def setArrowRightToTrue(self):
    self.keys["arrow_right"] = True

def setArrowRightToFalse(self):
    self.keys["arrow_right"] = False



def setArrowLeftToTrue(self):
    self.keys["arrow_left"] = True

def setArrowLeftToFalse(self):
    self.keys["arrow_left"] = False

def killGame(self):
    sys.exitfunc()
    sys.exit()
def keyInput(self, task):
    self.accept("w", self.setWToTrue)
    self.accept("w-up", self.setWToFalse)

    self.accept("s", self.setSToTrue)
    self.accept("s-up", self.setSToFalse)

    self.accept("a", self.setAToTrue)
    self.accept("a-up", self.setAToFalse)

    self.accept("d", self.setDToTrue)
    self.accept("d-up", self.setDToFalse)

    self.accept("escape", self.killGame)

    return Task.cont
```

```
    #Method that changes angle of camera
    def lookCharacter(self, task):

        if self.mw.hasMouse():
            mouseX = self.mw.getMouseX()
            mouseY = self.mw.getMouseY()

            #Resets mouse
            base.win.movePointer(0, self.win.getXSize() / 2,
self.win.getYSize() / 2)

            self.angleH += float(mouseX * -30)

            #These account for Windows beign mean with the dimensions of the
            #fullscreen


            #If the window height is an even number, do your thing.
            if float(self.win.getYSize()) % 2.0 == 0.0:

                self.angleP += (float(mouseY) * 30)



            #If the window height is an odd number, which causes a whole load
of
            #problems do fun things that fix said problems

            elif float(self.win.getYSize()) % 2.0 != 0.0:

                self.angleP += (float(mouseY) - 0.000879507453647) * 30




            #Angle correction
            if self.angleP > 80:
                self.angleP = 80

            if self.angleP < -80:
                self.angleP = -80

            if self.angleH == 360:
                self.angleH = 0

            if self.angleH > 360:
                self.angleH = 360 - self.angleH
```

```
        return Task.cont

    #Method that moves character in-game
    def moveCharacter(self, task):
        #Moving FORWARDS
        if self.keys["w"] == True:

            #90 and 270 are on the wrong sides as far as this operation is
concerned
            angleForMovement = 360 - self.angleH

            if angleForMovement == 360:
                angleForMovement = 0

            self.ypos += math.cos(math.radians(angleForMovement))
            self.xpos += math.sin(math.radians(angleForMovement))



        #Moving BACKWARDS
        if self.keys["s"] == True:

            #90 and 270 are on the wrong sides as far as this operation is
concerned
            angleForMovement = 360 - self.angleH

            #Make sure angleForMovement is 0 instead of 360, as cosine
doesn't work
            #with 0
            if angleForMovement == 360:
                angleForMovement = 0

            self.ypos -= math.cos(math.radians(angleForMovement))
            self.xpos -= math.sin(math.radians(angleForMovement))

        #Moving LEFT
        if self.keys["a"] == True:
            angleForMovement = self.angleH

            #Make sure angleForMovement is 0 instead of 360, as cosine
doesn't work
            #with 0
            if angleForMovement == 360:
                angleForMovement = 0

            self.ypos -= math.sin(math.radians(angleForMovement))
            self.xpos -= math.cos(math.radians(angleForMovement))


        #Moving RIGHT
        if self.keys["d"] == True:
            angleForMovement = self.angleH

            #Make sure angleForMovement is 0 instead of 360, as cosine
doesn't work
            #with 0
            if angleForMovement == 360:
                angleForMovement = 0

            self.ypos += math.sin(math.radians(angleForMovement))
            self.xpos += math.cos(math.radians(angleForMovement))

        return Task.cont
```

```
app = MyApp()

props = WindowProperties()
props.setTitle("Caleb's Game")
app.win.requestProperties(props)


app.run()
```