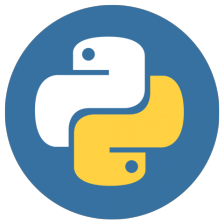
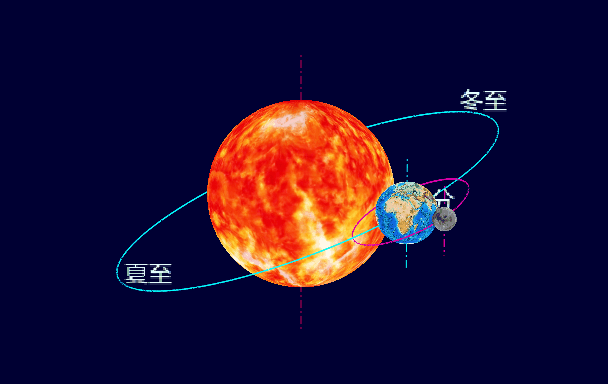
**画一个日月地球的轨道模型**

[[](https://blog.csdn.net/xufive/category_6727443.html)](https://blog.csdn.net/xufive/category_6727443.html)

无言相守45亿年，太阳、地球和月球这三个好基友究竟是怎样的关系呢？从孩提时代我就一直在想，要是能有一个可以直观演示太阳、地球和月球运行轨迹的模型就好了。今天，我终于实现了小时候的梦想：用WxGL画出了太阳、地球和月球的动态轨道模型。配上简单的解说，小朋友也可以秒懂四季更迭、日蚀月蚀、黄赤交角。



在开始绘制模型前，让我们先来了解一下太阳、地球和月球的起源，以及它们的大小、远近和行踪路线。

* 主流的大爆炸理论认为，宇宙大爆炸发生在138.17亿年前。
* 约66亿年前，一颗超新星爆炸后产生了一团星云，逐渐形成了太阳系的雏形。
* 约46亿年前，这团星云最中心的巨大氢球，开始产生聚变反应，太阳就此诞生。
* 约45亿年前，一颗火星大小的行星“忒伊亚”撞上了地球的雏形，体积增加了近一倍，地球诞生。
* 在这次撞击中，“忒伊亚”残骸的一部分形成了月球。
* 太阳半径696000km，地球半径6371km，月球半径1738km，三者之比约为为400:3.67:1。
* 太阳绕银河系中心公转周期约2.5亿年。
* 太阳也在自转，不过因为是等离子体，所以不同纬度有不同的自转速度。赤道区域自转最快，周期为24.47天。
* 地球自转周期为1天，公转周期为365.2564天（恒星年）或365.2422天（回归年）。
* 地球公转轨道是一个近似圆的椭圆，半长轴为149600000km，半短轴为149580000km，曲率为0.016722。
* 地球自转轴并不垂直于地球公转轨道面，地球公转轨道面（黄道面）与地球赤道面夹角为23.43° 。
* 月球公转周期为27.32日（恒星月），或29.53天（朔望月），月球自转周期为27.32日（恒星月）。
* 月球轨道面对黄道面的倾角是5.145°，月球赤道面对黄道面的倾角是1.543°，月球轨道面与月球赤道面夹角为6.688° 。
* 月球轨道是一个椭圆，半长轴为385000km，离心率是0.0549。
* 地球公转和自转方向、月球公转和自转方向、太阳的自转方向，是一致的，都是自西向东。

了解了这些，就可以开始绘制模型了。关于WxGL模块的安装，请参考我近期的文章，这里不再赘述。下面是完整的代码，大约一百余行。

# -\*- coding: utf-8 -\*-

import numpy as np

from scipy.spatial.transform import Rotation as sstr

from wxgl import wxplot as plt

R\_SUN = 1 # 以太阳半径696000km为1个单位

R\_EARTH = 30 \* 6371/696000 # 地球半径6371km，此处放大30倍

R\_MONTH = 50 \* 1738/696000 # 月球半径1738km，此处放大50倍

A\_EARTH = (149600000/696000)/100 # 地球公转轨道半长轴149600000km，此处缩小100倍

E\_EARTH = 0.016722 # 地球公转轨道离心率

A\_MONTH = 385000/696000 # 月球公转轨道半长轴385000km

E\_MONTH = 0.0549 # 月球公转轨道离心率

T = 36 # 每天36个计数周期

T\_SUN\_SELF = int(24.47 \* T) # 太阳自转周期24.47天

T\_EARTH\_SELF = 1 \* T # 地球自转周期1天

T\_MONTH\_SELF = int(27.32 \* T) # 月球自转周期27.32天

T\_EARTH = int(365.2564/3 \* T) # 地球公转周期365.2564天，此处取1/3

T\_MONTH = int(27.32 \* T) # 月球公转周期27.32天

A\_E\_ORBIT = 23.43 # 地球轨道面与地球赤道面夹角

A\_M\_ORBIT = 23.43 + 5.145 # 月球轨道面与地球赤道面夹角

A\_M\_EQUATOR = 23.43 - 1.543 # 月球赤道面与地球赤道面夹角

# 地球轨道倾角旋转器

ROTATOR\_E\_ORBIT = sstr.from\_euler('xyz', (0, -A\_E\_ORBIT, 0), degrees=True)

# 月球轨道倾角旋转器

ROTATOR\_M\_ORBIT = sstr.from\_euler('xyz', (0, -A\_M\_ORBIT, 0), degrees=True)

def rotate\_merge(av1, av2):

"""将两个轴角旋转合并为一个

av1 - 元组，首元素（浮点型）为旋转角度（逆时针为正，右手定则），尾元素（列表或元组）为旋转向量

av2 - 元组，首元素（浮点型）为旋转角度（逆时针为正，右手定则），尾元素（列表或元组）为旋转向量

"""

a1, v1 = av1

a2, v2 = av2

v1, v2 = np.array(v1), np.array(v2)

r1 = sstr.from\_rotvec(np.radians(a1)\*v1/np.linalg.norm(v1))

r2 = sstr.from\_rotvec(np.radians(a2)\*v2/np.linalg.norm(v2))

m = np.dot(r1.as\_matrix(), r2.as\_matrix())

r = sstr.from\_matrix(m)

vec = r.as\_rotvec()

phi = np.degrees(np.linalg.norm(vec))

return phi, vec

def get\_ellipse\_orbit(a, e, n):

"""计算椭圆轨道，a为半长轴， e为离心率，n为轨道点数"""

t = np.linspace(0, 2\*np.pi, n)

r = a\*(1-e\*e)/(1-e\*np.cos(t))

xs = r\*np.cos(t)

ys = r\*np.sin(t)

zs = np.zeros(r.shape)

return xs, ys, zs

def rotate\_s(i):

"""太阳自转函数：沿Z轴旋转，T\_SUN\_SELF个计数周期旋转一周"""

return (i%T\_SUN\_SELF)\*360/T\_SUN\_SELF, (0,0,1)

def rotate\_e\_orbit(i):

"""地球轨道旋转函数：沿y轴旋转23.43°（黄赤夹角）"""

return -A\_E\_ORBIT, (0,1,0)

def rotate\_e(i):

"""地球自转函数：沿z轴旋转，T\_EARTH\_SELF个计数周期旋转一周"""

return (i%T\_EARTH\_SELF)\*360/T\_EARTH\_SELF, (0,0,1)

def translate\_e(i):

"""地球位移函数：T\_EARTH个计数周期循环一次"""

phi = (i%T\_EARTH)\*2\*np.pi/T\_EARTH

r = A\_EARTH\*(1-E\_EARTH\*E\_EARTH)/(1-E\_EARTH\*np.cos(phi))

d = np.array((r\*np.sin(phi), -r\*np.cos(phi), 0))

return ROTATOR\_E\_ORBIT.apply(d)

def rotate\_m\_orbit(i):

"""月球轨道旋转函数：沿y轴旋转28.575°后再跟随月球自转"""

av1 = -A\_M\_ORBIT, (0,1,0)

av2 = (i%T\_MONTH\_SELF)\*360/T\_MONTH\_SELF, (0,0,1)

return rotate\_merge(av1, av2)

def translate\_m\_orbit(i):

"""月球轨道位移函数：T\_EARTH个计数周期循环一次"""

phi = (i%T\_EARTH)\*2\*np.pi/T\_EARTH

r = A\_EARTH\*(1-E\_EARTH\*E\_EARTH)/(1-E\_EARTH\*np.cos(phi))

d = np.array((r\*np.sin(phi), -r\*np.cos(phi), 0))

return ROTATOR\_E\_ORBIT.apply(d)

def translate\_m(i):

"""月球位移函数：T\_EARTH个计数周期循环一次"""

phi = (i%T\_EARTH)\*2\*np.pi/T\_EARTH

r = A\_EARTH\*(1-E\_EARTH\*E\_EARTH)/(1-E\_EARTH\*np.cos(phi))

d1 = np.array((r\*np.sin(phi), -r\*np.cos(phi), 0))

d1 = ROTATOR\_E\_ORBIT.apply(d1)

phi = (i%T\_MONTH)\*2\*np.pi/T\_MONTH

r = A\_MONTH\*(1-E\_MONTH\*E\_MONTH)/(1-E\_MONTH\*np.cos(phi))

d2 = np.array((r\*np.sin(phi), -r\*np.cos(phi), 0))

d2 = ROTATOR\_M\_ORBIT.apply(d2)

return d1 + d2

def rotate\_m(i):

"""月球自转函数：T\_MONTH\_SELF个计数周期旋转一周"""

#return (i%T\_MONTH\_SELF)\*360/T\_MONTH\_SELF, (0,-np.tan(np.radians(A\_M\_EQUATOR)),1)

return (i%T\_MONTH\_SELF)\*360/T\_MONTH\_SELF, (0,0,1)

# 初始化画布

plt.figure(elevation=5, azimuth=-25) # size=(1280,1080), zoom=0.45

plt.axis(grid=False)

# 绘制太阳及其自转轴

plt.sphere((0,0,0), R\_SUN, texture='res/sun.jpg', rotate=rotate\_s, order='R', light=0)

plt.plot((0,0), (0,0), (1.5,-1.5), color='red', style='dash-dot')

# 绘制地球公转轨道

xs\_e, ys\_e, zs\_e = get\_ellipse\_orbit(A\_EARTH, E\_EARTH, T\_EARTH)

plt.plot(xs\_e, ys\_e, zs\_e, color='cyan', width=1, rotate=rotate\_e\_orbit, order='R')

# 标注冬夏至和春秋分点

i\_summer, i\_winter = np.argmin(xs\_e), np.argmax(xs\_e)

i\_autumn, i\_spring = np.argmin(ys\_e), np.argmax(ys\_e)

plt.text('夏至', size=32, pos=ROTATOR\_E\_ORBIT.apply((xs\_e[i\_summer], ys\_e[i\_summer], zs\_e[i\_summer])))

plt.text('冬至', size=32, pos=ROTATOR\_E\_ORBIT.apply((xs\_e[i\_winter], ys\_e[i\_winter], zs\_e[i\_winter])))

plt.text('春分', size=32, pos=ROTATOR\_E\_ORBIT.apply((xs\_e[i\_spring], ys\_e[i\_spring], zs\_e[i\_spring])))

plt.text('秋分', size=32, pos=ROTATOR\_E\_ORBIT.apply((xs\_e[i\_autumn], ys\_e[i\_autumn], zs\_e[i\_autumn])))

# 绘制地球及其自转轴

plt.sphere((0,0,0), R\_EARTH, texture='res/earth.jpg', rotate=rotate\_e, translate=translate\_e, order='TR', light=0)

plt.plot((0,0), (0,0), (0.5,-0.5), color='cyan', style='dash-dot', rotate=rotate\_e, translate=translate\_e, order='TR')

# 绘制月球公转轨道

xs\_m, ys\_m, zs\_m = get\_ellipse\_orbit(A\_MONTH, E\_MONTH, T\_MONTH)

plt.plot(xs\_m, ys\_m, zs\_m, color='magenta', width=1, rotate=rotate\_m\_orbit, translate=translate\_m\_orbit, order='TR')

# 绘制月球及其自转轴

plt.sphere((0,0,0), 0.1, texture='res/month.jpg', rotate=rotate\_m, translate=translate\_m, order='TR', light=0)

plt.plot((0,0), (0,0), (0.3,-0.3), color='magenta', style='dash-dot', rotate=rotate\_m, translate=translate\_m, order='TR')

plt.show()

代码中用到了太阳、地球和月球的纹理图片，读者可自行下载或替换为自己喜欢的图片。



