In [1]:
```
#libraries
import numpy as np
import matplotlib.pyplot as plt
import celluloid
import sympy
from sympy import *
```

In [2]:
```
#semi major axes in au
b_a = 0.01154
c_a = 0.01580
d_a = 0.02227
e_a = 0.02925
f_a = 0.03849
g_a = 0.04683
h_a = 0.06189

#orbital period in years
b_p = 1.510826 / 365.25
c_p = 2.421937 / 365.25
d_p = 4.049219 / 365.25
e_p = 6.101013 / 365.25
f_p = 9.207540 / 365.25
g_p = 12.352446 / 365.25
h_p = 18.772866 / 365.25

#eccentricities
b_e = 0.00267
c_e = 0.00654
d_e = 0.00837
e_e = 0.00510
f_e = 0.01007
g_e = 0.00208
h_e = 0.00567

#time
t = np.linspace(0,24*b_p,1000)
```

In [3]:
```
#creating a function that uses Newton's method to solve kepler's equation and return the eccentric anomaly

def newton(e, M, E0, acc):
    '''Uses Newton's method to solve Kepler's Equation and return the eccentric anomaly '''

    #functions
    E = Symbol('E')
    F = E - e*sin(E) - M
    F_prime = sympy.diff(F,E)
    F = lambdify(E, F)
    F_prime = lambdify(E, F_prime)

    #inital values
    n = 0
    n_all = [n]

    #allows loop to start
    diff = 1
    E = E0

    #Newton's method
    while diff >= acc:
        n = n+1
        En = E - F(E)/F_prime(E)
        diff = abs(En - E)
        n_all.append(n)
        E = En
    #returns eccentric anomaly
    return(En)
```

In [6]:
```
#calculating position of planet b

#mean motion
b_n = 2*np.pi/b_p
#mean anomaly
b_M = b_n*t
#eccentric anomaly
b_E = []
for M in b_M:
    b_E_i = newton(b_e, M, M, 10**(-10))
    b_E.append(b_E_i)
b_E = np.array(b_E)
#distance to star
b_r = b_a*(1 - b_e*np.cos(b_E))
#true anomaly
```

```
b_f = 2*np.arctan(np.sqrt(((1+b_e)/(1-b_e)))*np.tan(b_E/2))
#x position
b_x = b_r*np.cos(b_f)
#y position
b_y = b_r*np.sin(b_f)
```

In [5]:
```
#calculating position of planet c

#mean motion
c_n = 2*np.pi/c_p
#mean anomaly
c_M = c_n*t
#eccentric anomaly
c_E = []
for M in c_M:
    c_E_i = newton(c_e, M, M, 10**(-10))
    c_E.append(c_E_i)
c_E = np.array(c_E)
#distance to star
c_r = c_a*(1 - c_e*np.cos(c_E))
#true anomaly
c_f = 2*np.arctan(np.sqrt(((1+c_e)/(1-c_e)))*np.tan(c_E/2))
#x position
c_x = c_r*np.cos(c_f)
#y position
c_y = c_r*np.sin(c_f)
```

In [8]:
```
#calculating the position of planet d

#mean motion
d_n = 2*np.pi/d_p
#mean anomaly
d_M = d_n*t
#eccentric anomaly
d_E = []
for M in d_M:
    d_E_i = newton(d_e, M, M, 10**(-10))
    d_E.append(d_E_i)
d_E = np.array(d_E)
#distance to star
d_r = d_a*(1 - d_e*np.cos(d_E))
#true anomaly
d_f = 2*np.arctan(np.sqrt(((1+d_e)/(1-d_e)))*np.tan(d_E/2))
#x position
d_x = d_r*np.cos(d_f)
#y position
d_y = d_r*np.sin(d_f)
```

In [9]:
```
#calculating the position of planet e

#mean motion
e_n = 2*np.pi/e_p
#mean anomaly
e_M = e_n*t
#eccentric anomaly
e_E = []
for M in e_M:
    e_E_i = newton(e_e, M, M, 10**(-10))
    e_E.append(e_E_i)
e_E = np.array(e_E)
#distance to star
e_r = e_a*(1 - e_e*np.cos(e_E))
#true anomaly
e_f = 2*np.arctan(np.sqrt(((1+e_e)/(1-e_e)))*np.tan(e_E/2))
#x position
e_x = e_r*np.cos(e_f)
#y position
e_y = e_r*np.sin(e_f)
```

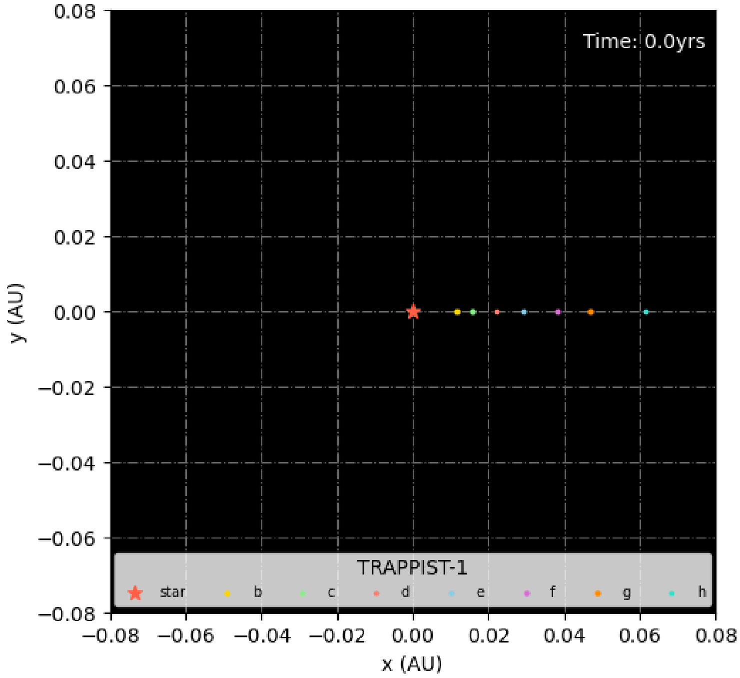
In [10]:
```
#calculating the position of planet f

#mean motion
f_n = 2*np.pi/f_p
#mean anomaly
f_M = f_n*t
#eccentric anomaly
f_E = []
for M in f_M:
    f_E_i = newton(f_e, M, M, 10**(-10))
    f_E.append(f_E_i)
f_E = np.array(f_E)
#distance to star
f_r = f_a*(1 - f_e*np.cos(f_E))
```

```
#true anomaly
f_f = 2*np.arctan(np.sqrt(((1+f_e)/(1-f_e)))*np.tan(f_E/2))
#x position
f_x = f_r*np.cos(f_f)
#y position
f_y = f_r*np.sin(f_f)
```

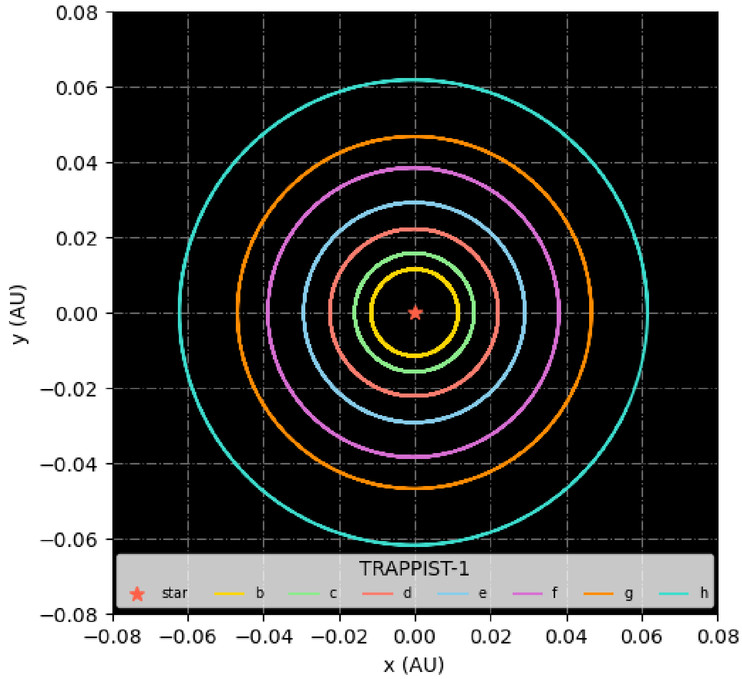
In [11]:
```
#calculating the position of planet g

#mean motion
g_n = 2*np.pi/g_p
#mean anomaly
g_M = g_n*t
#eccentric anomaly
g_E = []
for M in g_M:
    g_E_i = newton(g_e, M, M, 10**(-10))
    g_E.append(g_E_i)
g_E = np.array(g_E)
#distance to star
g_r = g_a*(1 - g_e*np.cos(g_E))
#true anomaly
g_f = 2*np.arctan(np.sqrt(((1+g_e)/(1-g_e)))*np.tan(g_E/2))
#x position
g_x = g_r*np.cos(g_f)
#y position
g_y = g_r*np.sin(g_f)
```

In [12]:
```
#calculating the position of planet h

#mean motion
h_n = 2*np.pi/h_p
#mean anomaly
h_M = h_n*t
#eccentric anomaly
h_E = []
for M in h_M:
    h_E_i = newton(h_e, M, M, 10**(-10))
    h_E.append(h_E_i)
h_E = np.array(h_E)
#distance to star
h_r = h_a*(1 - h_e*np.cos(h_E))
#true anomaly
h_f = 2*np.arctan(np.sqrt(((1+h_e)/(1-h_e)))*np.tan(h_E/2))
#x position
h_x = h_r*np.cos(h_f)
#y position
h_y = h_r*np.sin(h_f)
```

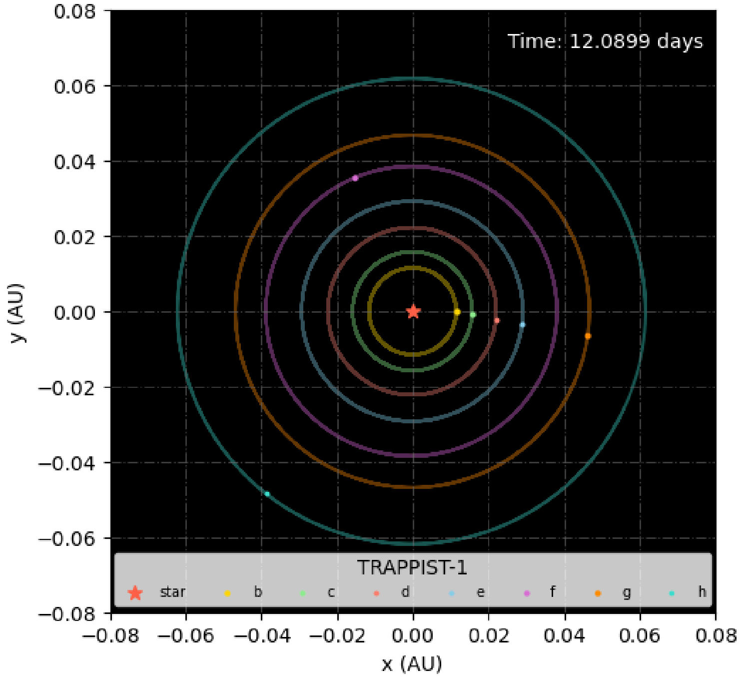
In [14]:
```
#plotting orbits at t=0

fig = plt.figure(figsize = (5.5,5.5))
ax = plt.axes()
ax.set_facecolor('black')

plt.ylim(-0.08,0.08)
plt.xlim(-0.08,0.08)
plt.scatter(0,0, c = 'tomato', s = 4*13.08,  label = 'star', zorder = 2, marker = '*')
plt.scatter(b_x[0],b_y[0], s = 4*1.116,  label = 'b', c = 'gold', zorder = 2)
plt.scatter(c_x[0],c_y[0], s = 4*1.097, label = 'c', c = 'lightgreen', zorder = 2)
plt.scatter(d_x[0],d_y[0], s = 4*0.788, label = 'd',  c = 'salmon', zorder = 2)
plt.scatter(e_x[0],e_y[0], s = 4*0.920, label = 'e', c = 'skyblue', zorder = 2)
plt.scatter(f_x[0],f_y[0], s = 4*1.045, label = 'f',  c = 'orchid', zorder = 2)
plt.scatter(g_x[0],g_y[0], s = 4*1.129, label = 'g', c = 'darkorange', zorder = 2)
plt.scatter(h_x[0],h_y[0], s = 4*0.755, label = 'h', c = 'turquoise', zorder = 2)
plt.text(0.045,0.07,'Time: '+str(round(t[0],3))+'yrs', color = 'white')

plt.xlabel("x (AU)")
plt.ylabel("y (AU)")
plt.grid(linestyle = '-.', c = 'dimgray', zorder = 0)
plt.legend( title = "TRAPPIST-1", ncol= 8, loc="lower center", fontsize = 7)
plt.show()
```



In [15]:
```
#plotting full orbital paths

fig = plt.figure(figsize = (5.5,5.5))
ax = plt.axes()
ax.set_facecolor('black')

plt.ylim(-0.08,0.08)
plt.xlim(-0.08,0.08)
plt.scatter(0,0, c = 'tomato', s = 4*13.08,  label = 'star', zorder = 2, marker = '*')
plt.plot(b_x,b_y, label = 'b', c = 'gold', zorder = 2)
plt.plot(c_x,c_y, label = 'c', c = 'lightgreen', zorder = 2)
plt.plot(d_x,d_y, label = 'd',  c = 'salmon', zorder = 2)
plt.plot(e_x,e_y, label = 'e', c = 'skyblue', zorder = 2)
plt.plot(f_x,f_y, label = 'f',  c = 'orchid', zorder = 2)
plt.plot(g_x,g_y, label = 'g', c = 'darkorange', zorder = 2)
plt.plot(h_x,h_y, label = 'h', c = 'turquoise', zorder = 2)
#plt.text(0.04,0.07,'Time:'+str(round(t[0],3))+'yrs')

plt.xlabel("x (AU)")
plt.ylabel("y (AU)")
plt.savefig('fig2.jpg')
plt.grid(linestyle = '-.', c = 'dimgray', zorder = 0)

plt.legend(title = "TRAPPIST-1", ncol= 8, loc="lower center", fontsize = 7)
plt.show()
```

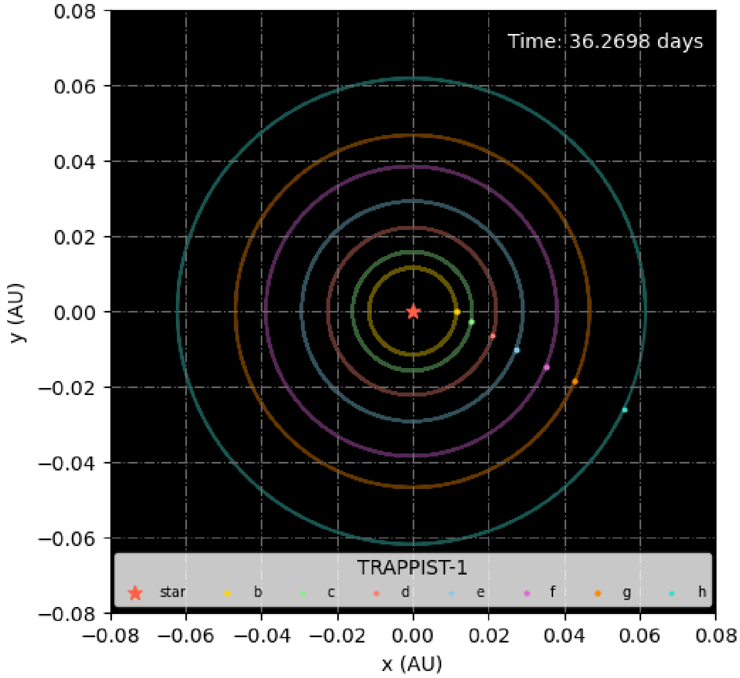


In [17]:
```
#plotting orbits at t=12

fig = plt.figure(figsize = (5.5,5.5))
ax = plt.axes()
ax.set_facecolor('black')
i = 333
plt.ylim(-0.08,0.08)
plt.xlim(-0.08,0.08)
plt.scatter(0,0, c = 'tomato', s = 4*13.08,  label = 'star', zorder = 2, marker = '*')
plt.scatter(b_x[i],b_y[i], s = 4*1.116,  label = 'b', c = 'gold', zorder = 2)
plt.scatter(c_x[i],c_y[i], s = 4*1.097, label = 'c', c = 'lightgreen', zorder = 2)
plt.scatter(d_x[i],d_y[i], s = 4*0.788, label = 'd',  c = 'salmon', zorder = 2)
plt.scatter(e_x[i],e_y[i], s = 4*0.920, label = 'e', c = 'skyblue', zorder = 2)
plt.scatter(f_x[i],f_y[i], s = 4*1.045, label = 'f',  c = 'orchid', zorder = 2)
plt.scatter(g_x[i],g_y[i], s = 4*1.129, label = 'g', c = 'darkorange', zorder = 2)
plt.scatter(h_x[i],h_y[i], s = 4*0.755, label = 'h', c = 'turquoise', zorder = 2)
plt.text(0.025,0.07,'Time: '+str(round(t[i]*365.35,4))+' days', color = 'white')

plt.plot(b_x[:3000],b_y[:3000], c = 'gold', zorder = 1, alpha = 0.4)
plt.plot(c_x[:3000],c_y[:3000], c = 'lightgreen', zorder = 1, alpha = 0.4)
plt.plot(d_x[:3000],d_y[:3000],  c = 'salmon', zorder = 1, alpha = 0.4)
plt.plot(e_x[:3000],e_y[:3000], c = 'skyblue', zorder = 1, alpha = 0.4)
plt.plot(f_x[:3000],f_y[:3000],  c = 'orchid', zorder = 1, alpha = 0.4)
plt.plot(g_x[:3000],g_y[:3000], c = 'darkorange', zorder = 1, alpha = 0.4)
plt.plot(h_x[:3000],h_y[:3000], c = 'turquoise', zorder = 1, alpha = 0.4)

plt.xlabel("x (AU)")
plt.ylabel("y (AU)")
plt.savefig('fig4.jpg')
plt.grid(linestyle = '-.', c = 'dimgray', zorder = 0, alpha = 0.6)
ax.legend(title = "TRAPPIST-1", ncol= 8, loc="lower center", fontsize = 7)
plt.show()
```



In [18]:

```
#plotting orbits at t=36.26

fig = plt.figure(figsize = (5.5,5.5))
ax = plt.axes()
ax.set_facecolor('black')
i = -1
plt.ylim(-0.08,0.08)
plt.xlim(-0.08,0.08)
plt.scatter(0,0, c = 'tomato', s = 4*13.08,  label = 'star', zorder = 2, marker = '*')
plt.scatter(b_x[i],b_y[i], s = 4*1.116,  label = 'b', c = 'gold', zorder = 2)
plt.scatter(c_x[i],c_y[i], s = 4*1.097, label = 'c', c = 'lightgreen', zorder = 2)
plt.scatter(d_x[i],d_y[i], s = 4*0.788, label = 'd',  c = 'salmon', zorder = 2)
plt.scatter(e_x[i],e_y[i], s = 4*0.920, label = 'e', c = 'skyblue', zorder = 2)
plt.scatter(f_x[i],f_y[i], s = 4*1.045, label = 'f',  c = 'orchid', zorder = 2)
plt.scatter(g_x[i],g_y[i], s = 4*1.129, label = 'g', c = 'darkorange', zorder = 2)
plt.scatter(h_x[i],h_y[i], s = 4*0.755, label = 'h', c = 'turquoise', zorder = 2)
plt.text(0.025,0.07,'Time: '+str(round(t[i]*365.35,4))+' days', color = 'white')

plt.plot(b_x[:3000],b_y[:3000], c = 'gold', zorder = 1, alpha = 0.4)
plt.plot(c_x[:3000],c_y[:3000], c = 'lightgreen', zorder = 1, alpha = 0.4)
plt.plot(d_x[:3000],d_y[:3000],  c = 'salmon', zorder = 1, alpha = 0.4)
plt.plot(e_x[:3000],e_y[:3000], c = 'skyblue', zorder = 1, alpha = 0.4)
plt.plot(f_x[:3000],f_y[:3000],  c = 'orchid', zorder = 1, alpha = 0.4)
plt.plot(g_x[:3000],g_y[:3000], c = 'darkorange', zorder = 1, alpha = 0.4)
plt.plot(h_x[:3000],h_y[:3000], c = 'turquoise', zorder = 1, alpha = 0.4)

plt.xlabel("x (AU)")
plt.ylabel("y (AU)")
plt.savefig('fig4.jpg')
plt.grid(linestyle = '-.', c = 'dimgray', zorder = 0)
ax.legend(title = "TRAPPIST-1", ncol= 8, loc="lower center", fontsize = 7)
plt.show()
```



In [ ]:

```
#plotting the orbit simulation
from celluloid import Camera

#point sizes are scaled to real sizes in terms of Earth Radii
fig = plt.figure(figsize = (5.5,5.5))
ax = plt.axes()
ax.set_facecolor('black')
camera = Camera(fig)
plt.xlabel("x (AU)")
plt.ylabel("y (AU)")
plt.ylim(-0.08,0.08)
plt.xlim(-0.08,0.08)
plt.grid(linestyle = '-.', c = 'dimgray', zorder = 0)

for i in range(len(b_x)):
    plt.scatter(0,0, c = 'tomato', s = 4*13.08,  label = 'star', zorder = 2, marker = '*')
    plt.scatter(b_x[i],b_y[i], s = 4*1.116,  label = 'b', c = 'gold', zorder = 2)
    plt.scatter(c_x[i],c_y[i], s = 4*1.097, label = 'c', c = 'lightgreen', zorder = 2)
    plt.scatter(d_x[i],d_y[i], s = 4*0.788, label = 'd',  c = 'salmon', zorder = 2)
    plt.scatter(e_x[i],e_y[i], s = 4*0.920, label = 'e', c = 'skyblue', zorder = 2)
    plt.scatter(f_x[i],f_y[i], s = 4*1.045, label = 'f',  c = 'orchid', zorder = 2)
    plt.scatter(g_x[i],g_y[i], s = 4*1.129, label = 'g', c = 'darkorange', zorder = 2)
    plt.scatter(h_x[i],h_y[i], s = 4*0.755, label = 'h', c = 'turquoise', zorder = 2)
    plt.text(0.025,0.07,'Time: '+str(round(t[i]*365.35,4))+' days', color = 'white')

    plt.plot(b_x[:3000],b_y[:3000], c = 'gold', zorder = 1, alpha = 0.4)
    plt.plot(c_x[:3000],c_y[:3000], c = 'lightgreen', zorder = 1, alpha = 0.4)
    plt.plot(d_x[:3000],d_y[:3000],  c = 'salmon', zorder = 1, alpha = 0.4)
    plt.plot(e_x[:3000],e_y[:3000], c = 'skyblue', zorder = 1, alpha = 0.4)
    plt.plot(f_x[:3000],f_y[:3000],  c = 'orchid', zorder = 1, alpha = 0.4)
    plt.plot(g_x[:3000],g_y[:3000], c = 'darkorange', zorder = 1, alpha = 0.4)
    plt.plot(h_x[:3000],h_y[:3000], c = 'turquoise', zorder = 1, alpha = 0.4)
    camera.snap()

plt.legend(['star','b','c','d','e','f','g','h'], title = "TRAPPIST-1", ncol= 8, loc="lower center", fontsize = 7)

animation = camera.animate()
animation.save('TRAPPIST-1.gif')
```