Question 3. By modifying ‘ode-3body.py’, simulate the Armageddon project. Assuming a certain value of split speed by explosion, determine the closest limit point of explosion to avoid the disaster.

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integ
```

In [2]:

```
# A vector function. y is a two-compo vector, i.e. (x,v).
# f returns two-compo vector, i.e. (f1,f2)=(.
def F(z,t):
    x1,y1,x2,y2,x3,y3,v1x,v1y,v2x,v2y,v3x,v3y=z

    r12=np.sqrt((x1-x2)**2+(y1-y2)**2)
    r23=np.sqrt((x2-x3)**2+(y2-y3)**2)
    r31=np.sqrt((x3-x1)**2+(y3-y1)**2)

    f1x=-m2*(x1-x2)/r12**3 - m3*(x1-x3)/r31**3
    f1y=-m2*(y1-y2)/r12**3 - m3*(y1-y3)/r31**3

    f2x=-m1*(x2-x1)/r12**3 - m3*(x2-x3)/r23**3
    f2y=-m1*(y2-y1)/r12**3 - m3*(y2-y3)/r23**3

    f3x=-m1*(x3-x1)/r31**3 - m2*(x3-x2)/r23**3
    f3y=-m1*(y3-y1)/r31**3 - m2*(y3-y2)/r23**3

    return [v1x,v1y,v2x,v2y,v3x,v3y,f1x,f1y,f2x,f2y,f3x,f3y]
```

In [3]:

```
h=float(0.01)   # time step
nStep=int(40)  # num. steps to run
m1 = float(1)   # mass of object 1
m2 = float(1)   # mass of object 2
m3 = float(30)   # mass of object 3
```

In [4]:

```
t=np.arange(0,h*nStep,h)  # an array of discretized time
# initial condition : m1 and m2 should have same condition
# z0=[0.5,0,  -1.5,0, 0,0,  0,4.3, 0,-1.5,  0,0]  # initial conditions: x0=1, v0=0
x1 = -1.;y1 = 0.
x2 = -1.1;y2 = 0.
x3 = 0.;y3 = 0. #initial position
r = 0.3 #radius or m3
```

If distance between m1 and m3 or m2 and m3 is smaller than r, It means m1 or m2 crash into m3. (since m3 has radius r).

In [21]:

```
def Armageddon(theta) :
    v0 = 7
# To split m1 and m2 perfectly, relative vy  should be larger than escape velocity
    vx1 = v0*np.cos(theta)
    vy1 = v0*np.sin(theta)
    vx2= v0*np.cos(theta)
    vy2=-v0*np.sin(theta);vx3 = 0.;vy3 = 0.
    z0=[x1,y1,x2,y2,x3,y3,vx1,vy1,vx2,vy2,vx3,vy3]
    sol = integ.odeint(F,z0,t) # sol is [[x0,y0,vx0,vy0],..]


    for i in range(nStep) :
        d1 = np.sqrt((sol[:,0]-sol[:,4])**2+(sol[:,1]-sol[:,5])**2 )
        d2 = np.sqrt((sol[:,2]-sol[:,4])**2+(sol[:,3]-sol[:,5])**2)
        if d1[i] <=r or d2[i]<=r: # if m1 or m2 crash into m3
#             print(d1[i],d2[i])
            print("boom!")
            result = 'boom!'

            figure, axes = plt.subplots()

            draw_circle = plt.Circle((sol[i,4],sol[i,5]), r,fill=False)
            axes.set_aspect(1)
            axes.add_artist(draw_circle)

            plt.scatter(sol[i,0],sol[i,1])
            plt.scatter(sol[i,2],sol[i,3])
            plt.scatter(sol[i,4],sol[i,5])
            plt.plot(sol[:,0],sol[:,1],color = 'r')
            plt.plot(sol[:,2],sol[:,3], color = 'b')
            plt.plot(sol[:,4],sol[:,5], color = 'g');plt.show()
            break
        if i == nStep -1 :
            print("Safe!")
            result = 'Safe!'

    return result
```

In [22]:

```
Armageddon(np.pi/4)
```

```
Safe!
```

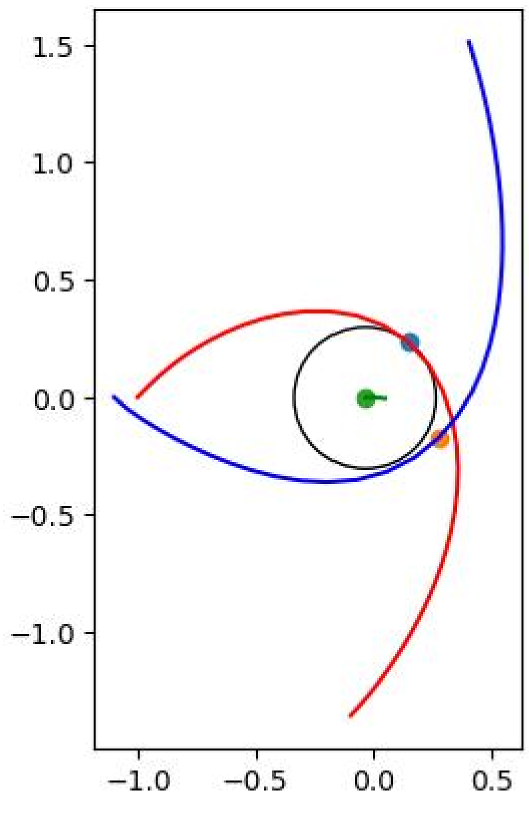
Out[22]:

```
'Safe!'
```

In [24]:

```
theta = np.pi/4
while True :
    result = Armageddon(theta)
    if result == 'boom!' :
        print("limit angle:",theta)
        break
    else :
        theta = theta - 0.001
```

```
Safe!
Safe!
Safe!
Safe!
Safe!
Safe!
Safe!
Safe!
Safe!
Safe!
boom!
```



```
limit angle: 0.7753981633974483
```

In [ ]: