Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»

Институт Информационных и Вычислительных Технологий

Кафедра Прикладной Математики и Искусственного Интеллекта

КУРСОВАЯ РАБОТА

по дисциплине «Численные методы»

Тема: «Полет вокруг Луны»

Студент
гр. А-13б-20 Имамкулов Х.М
Студент
гр. А-13б-20 Захаров И.В

Руководитель Крымов Н.Е.
(подпись)

Москва **2022**

СОДЕРЖАНИЕ

1. Постановка задачи

Рассмотрим модель из трех тел: Земля, Луна и летательный аппарат (далее ЛА).

Радиус Луны 1737400 м.

Радиус Земли 6371000 м.

Будем считать Землю неподвижной материальной точкой в начале координат с массой m1 = 5.9742 × 10^{24} кг.

Луна представляет собой материальную точку с массой m2 = 7.36 × 10^{22} кг и движется со скоростью v2 = 1020 м/с.

Координаты луны определяются равенствами:

$$r_2(t) = \begin{pmatrix} x_2(t) \\ y_2(t) \end{pmatrix}, \quad x_2(t) = 384400000\cos t, \quad y_2(t) = 384400000\sin t$$

ЛА стартует со скоростью v3(0) из точки с координатами

$$r_3(0) = \begin{pmatrix} x_3(0) \\ y_3(0) \end{pmatrix}, \quad x_3(0) = 6371000\cos\alpha, \quad y_3(0) = 6371000\sin\alpha$$

Направление скорости соответствует вектору $(\cos\alpha \quad \sin\alpha)^T$.

Массой ЛА можно пренебречь.

Радиус-вектор $\boldsymbol{r_3}(t)$ ЛА удовлетворяет дифференциальному уравнению

$$\frac{d^2\boldsymbol{r_3}}{dt^2} = Gm_1\frac{-\boldsymbol{r_3}}{|\boldsymbol{r_3}|^3} + Gm_2\frac{\boldsymbol{r_2} - \boldsymbol{r_3}}{|\boldsymbol{r_2} - \boldsymbol{r_3}|^3},$$

где $G = 6.67430(15) \times 10^{-11}$ м$^3 \cdot$ с$^{-2} \cdot$ кг$^{-1}$ – гравитационная постоянная.

Определите величину v3(0) и параметр α так, чтобы решить каждую из следующих инженерных задач:

1. Считая, что ЛА несет термоядерный заряд достаточной мощности, взорвите Луну.

2. Сделайте полный оборот вокруг Луны, не врезавшись в нее.

Примечание. Скорость v3(0) не может быть меньше второй космической скорости, равной 11200 м/с. Сведите задачу к задаче Коши для системы ОДУ. Задачу Коши решать методом Адамса-Башфорта 4-го порядка[1]. Результатом решения задачи должна быть анимация полета ЛА по полу

2. Построение алгоритма

1. Представим дифференциальное уравнение в виде системы:

$$\frac{d^2 r_3}{dt^2} = Gm_1 \frac{-r_3}{|r_3|^3} + Gm_2 \frac{r_2 - r_3}{|r_2 - r_3|^3}$$

$$r_3 = r; r_3' = r'; r_3' = v; r_3'' = v'$$

$$\begin{cases} r' = v \\ v' = Gm_1 \dfrac{-r}{|r|^3} + Gm_2 \dfrac{r_2 - r}{|r_2 - r|^3} \end{cases}$$

2. Первые 4 приближенных значения системы найдем методом Рунге-Кутты 4ого порядка точности[1] с заданным шагом (h = dt = 1c.)

$$Y_{i+1} = Y_i + \frac{1}{6} dt(K_1 + 2K_2 + 2K_3 + K_4),$$

$$K_1 = F(t, Y_i);$$
$$K_2 = F\left(t + \frac{dt}{2}, Y_i + \frac{dt}{2} K_1\right);$$
$$K_3 = F\left(t + \frac{dt}{2}, Y_i + \frac{dt}{2} K_2\right);$$
$$K_4 = F(t + dt, Y_i + dtK_3);$$

$$Y_i = \begin{pmatrix} r_i \\ v_i \end{pmatrix}; \quad F = \begin{pmatrix} v_i \\ Gm_1 \dfrac{-r_i}{|r_i|^3} + Gm_2 \dfrac{r_2(t) - r_i}{|r_2(t) - r_i|^3} \end{pmatrix}$$

3. Для дальнейшего расчета положения ЛА в пространстве будем использовать метод Адамса-Башфорта 4ого порядка точности с заданным шагом (h = dt = 1c.)

$$Y_{i+1} = Y_i + \frac{dt}{24}\big(55F(t, Y_i) - 59F(t - dt, Y_{i-1}) + 37F(t - 2dt, Y_{i-2}) - 9F(t - 3dt, Y_{i-3})\big),$$

$$Y_i = \begin{pmatrix} r_i \\ v_i \end{pmatrix}; \quad F = \begin{pmatrix} v_i \\ Gm_1 \dfrac{-r_i}{|r_i|^3} + Gm_2 \dfrac{r_2(t) - r_i}{|r_2(t) - r_i|^3} \end{pmatrix}$$

Применение алгоритма.

Решение задачи 1.

Имея начальный вектор из начального положения ЛА по x и y (Система координат относительно Земли) и проекций начальной скорости ЛА на оси x, y получим 3 начальных вектора методом РК 4ого порядка точности для возможности применения четырехшагового метода Адамса-Башфорта 4ого порядка точности. Далее для расчета положения ЛА в пространстве будем использовать метод Адамса-Башфорта. Будем считать, что ЛА покинул рассматриваемую область, если расстояние от него до Земли превысило 2 расстояния от Земли до Луны. Экспериментальным путем установлено, что, имея начальную скорость равной второй космической, ЛА независимо от точки старта на Земле в течении 150000 секунд не покинет рассматриваемую область, не врежется в Землю и не врежется в Луну. Поэтому на протяжении такого временного промежутка после старта не будем делать соответствующих проверок. Далее будем выполнять эти проверки каждые 60 секунд. Если за 7 суток не выполнилась ни одна проверка будем расценивать это как промах. Координаты ЛА через каждый час полета будем записывать в специальный массив, с помощью которого будет строиться анимация. Корректировать угол старта ЛА будем, опираясь на результаты анимации. После корректировок получим, что при стартовой скорости равной второй космической подходящий угол для попадания в Луну – 26.6 градусов.

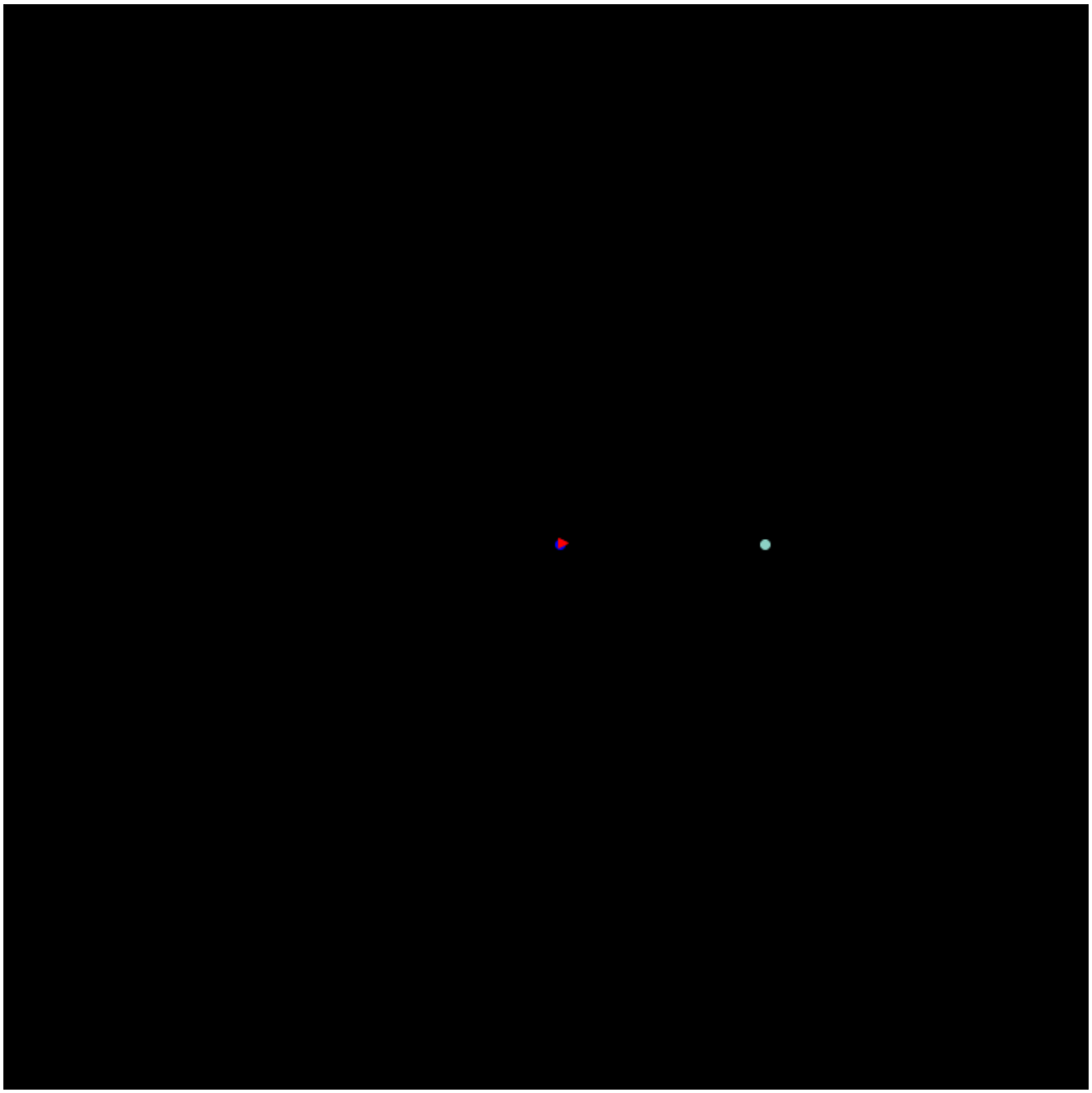
Решение задачи 2.

Для того, чтобы сделать оборот вокруг Луны нужно попасть на ее орбиту. Выберем начальную скорость, равную второй космической т.к. при увеличении скорости ЛА труднее удержаться на орбите Луны. Угол будем искать методом половинного деления, основываясь на 2-ух состояниях: столкновение с Луной и его отсутствие. Таким образом, получим угол, при котором ЛА будет пролетать по ближайшей к Луне орбите, не врезаясь в нее. Экспериментально было выявлено, что ЛА не может совершить оборот вокруг Луны, пролетая даже по самой близкой орбите.

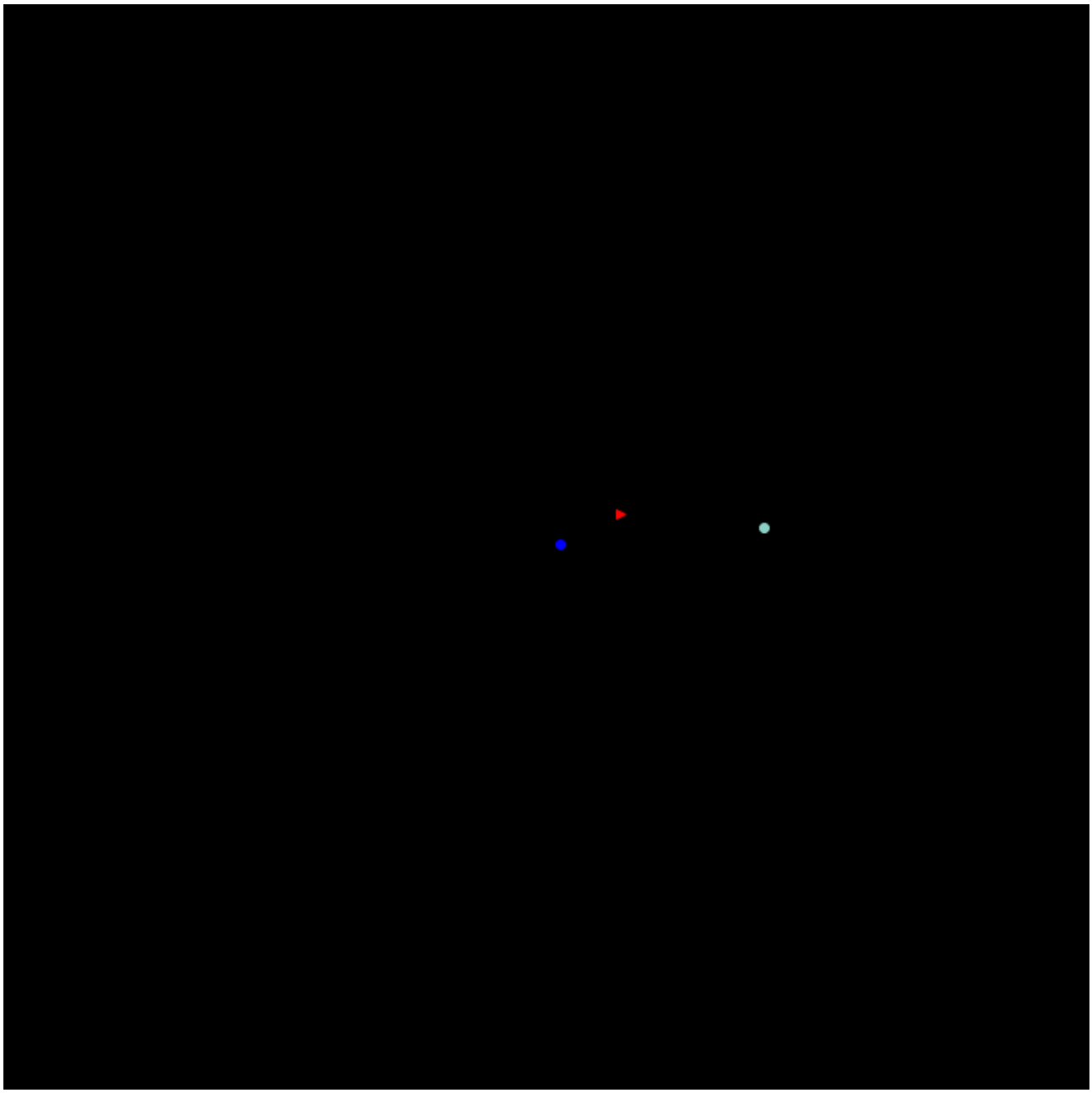
Построение симуляции полета ракеты.

Для построения анимации будем использовать библиотеки matplotlib.animation[2] и wand[3] и функцию FuncAnimation[4] для создания gif-файла. Укажем параметры таким образом, чтобы за 2 секунды реального времени в анимации проходили сутки.

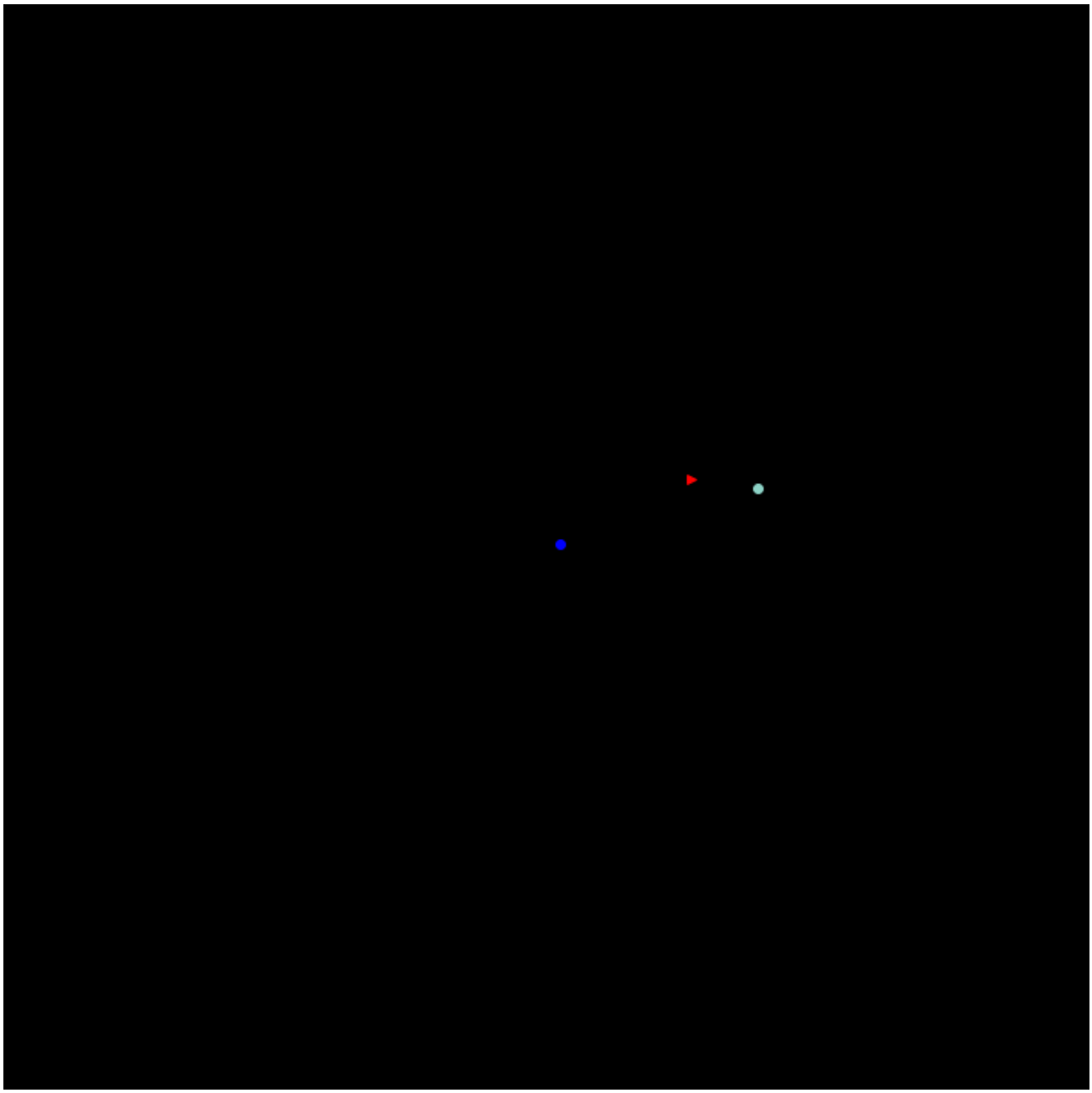
Симуляция задачи 1.



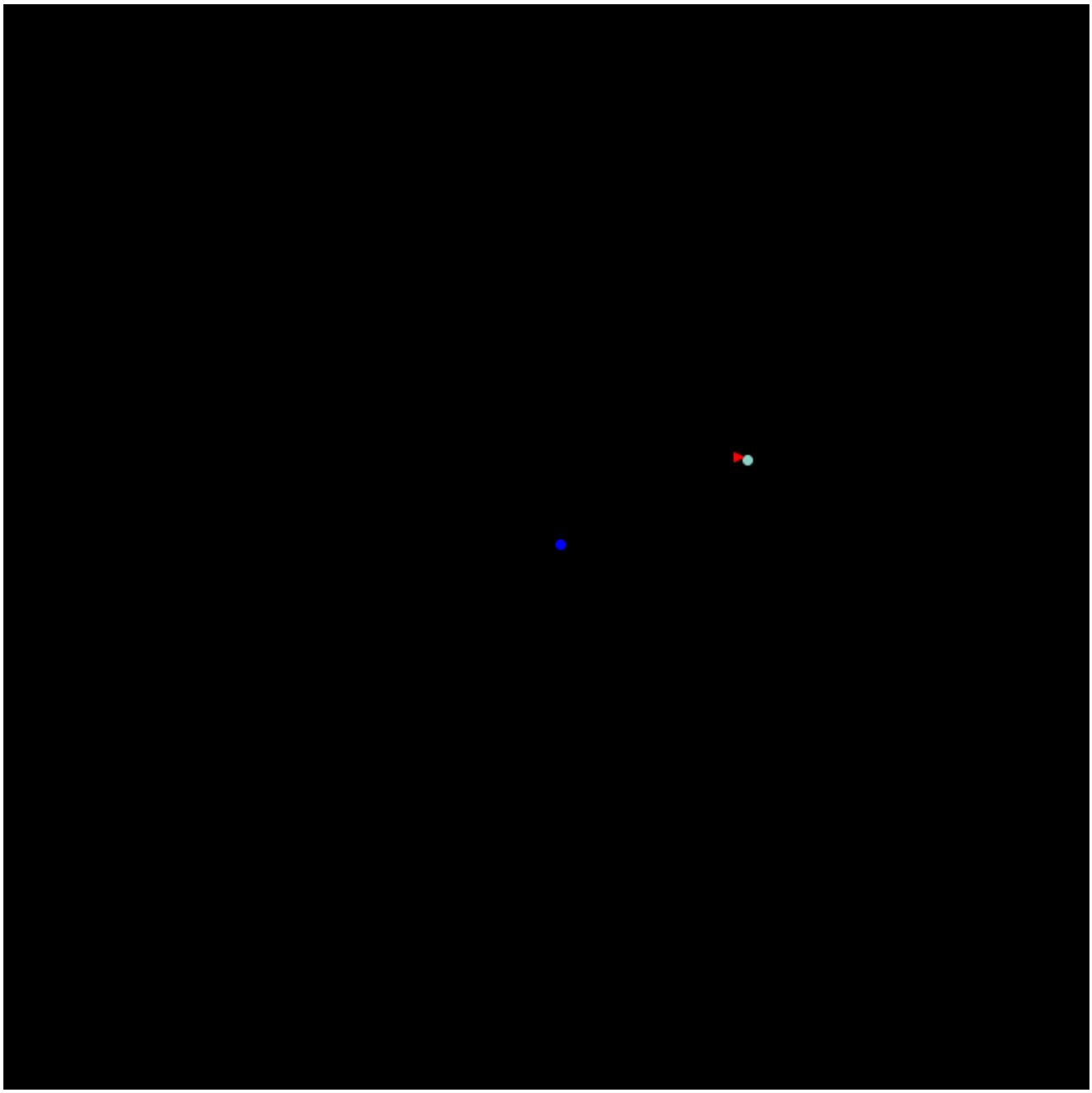
Первые сутки полета ЛА



Вторые сутки полета ЛА



Третьи сутки полета ЛА

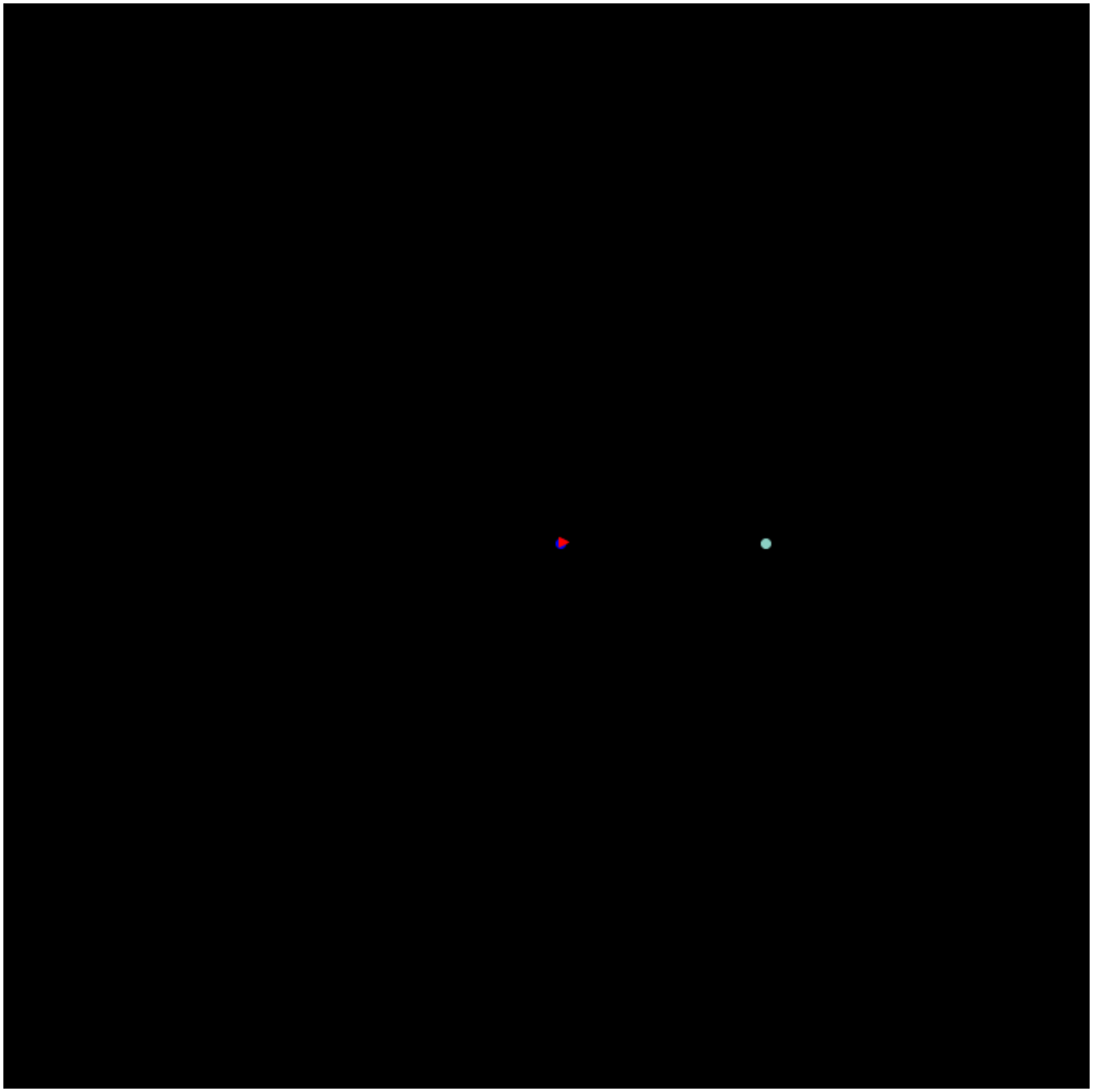


Симуляция задачи 2.

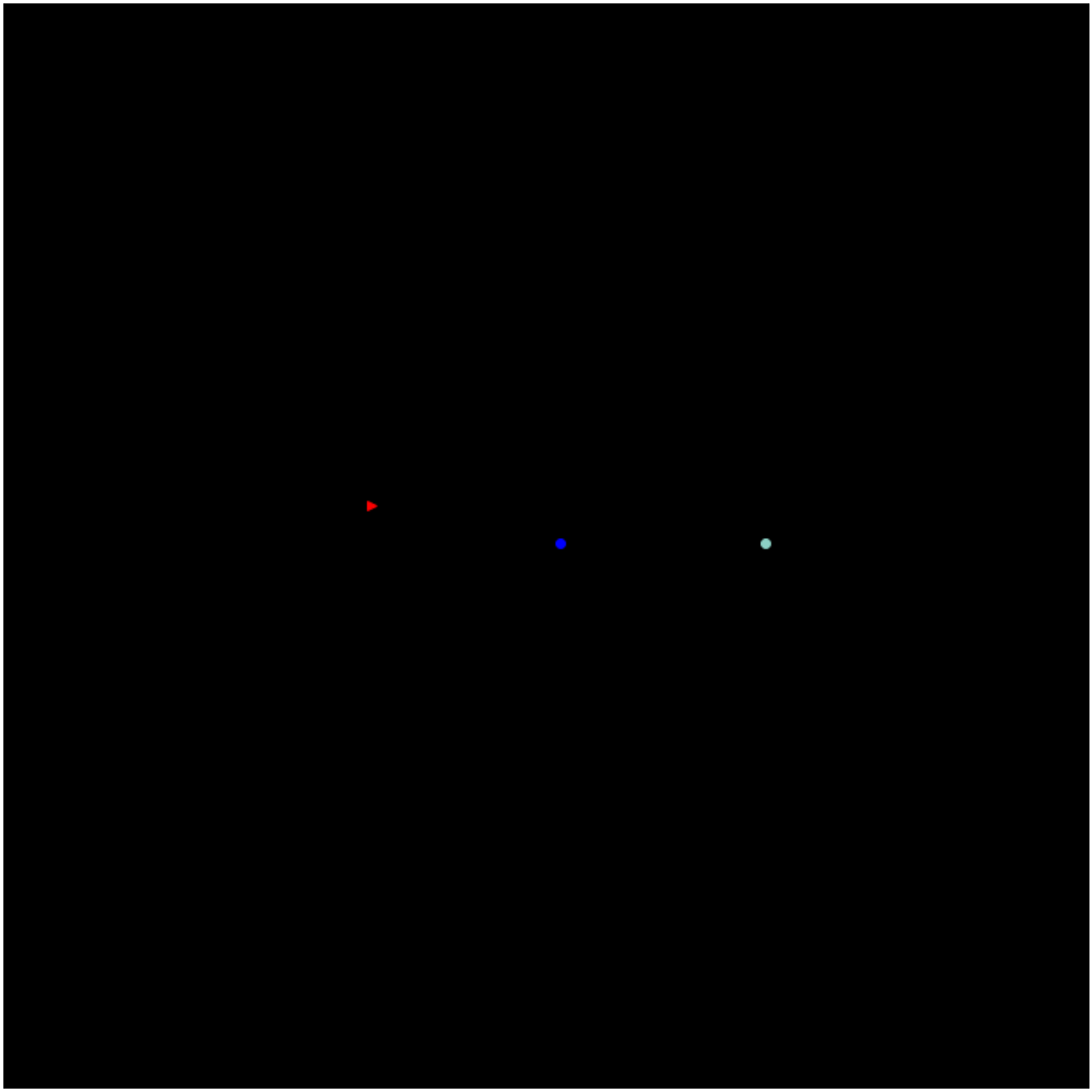
Т.к. сделать оборот вокруг Луны оказалось невозможно приведем симуляцию полета ЛА на протяжении определенного времени (76 дней 8 часов). В результате симуляции ЛА вылетел за пределы рассматриваемой области.

Для данной анимации использовался поворот системы координат, чтобы Луна имела статичное положение.

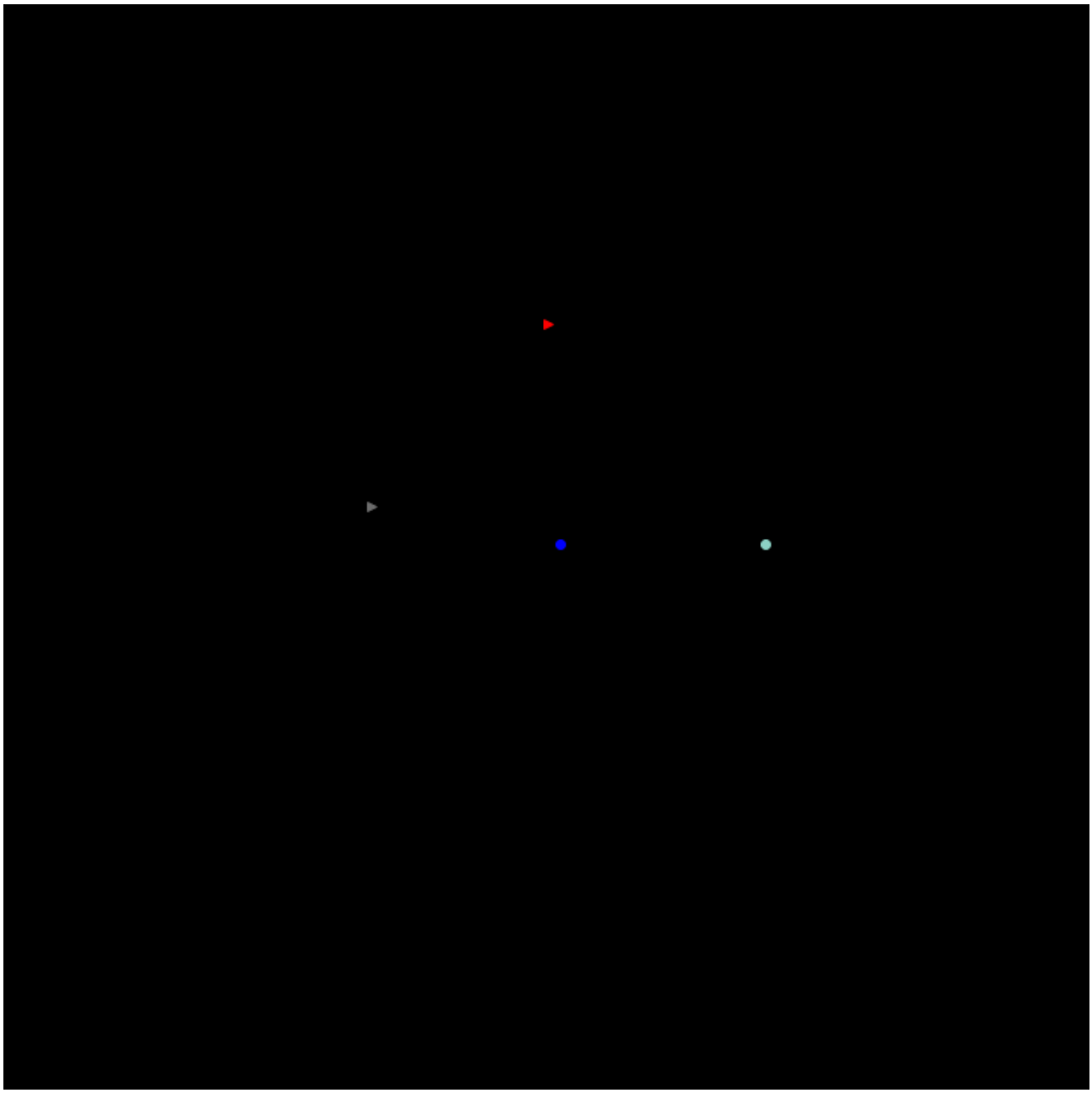
Серый треугольник – предыдущее положение ЛА.



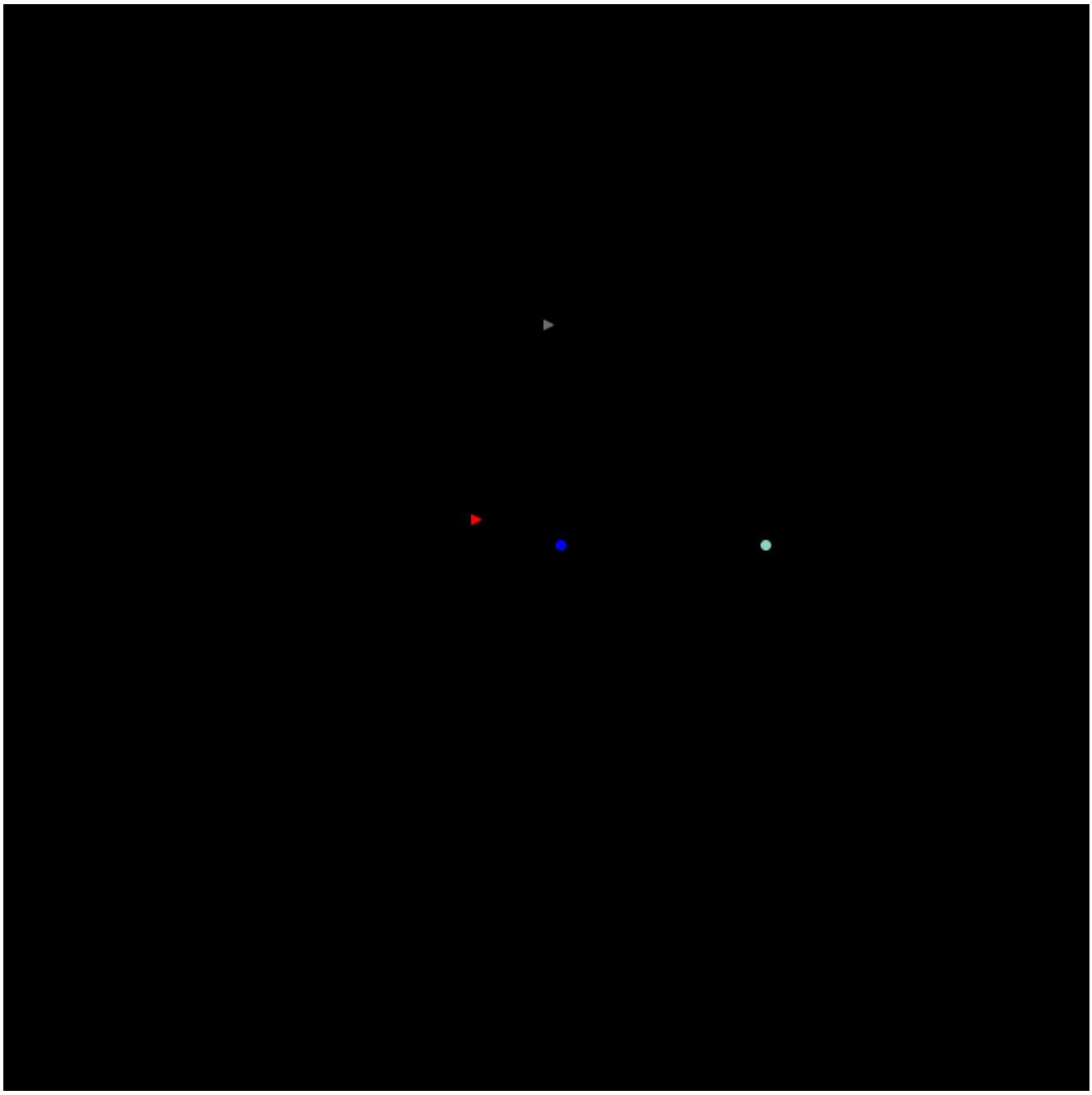
Первые десять дней полета:



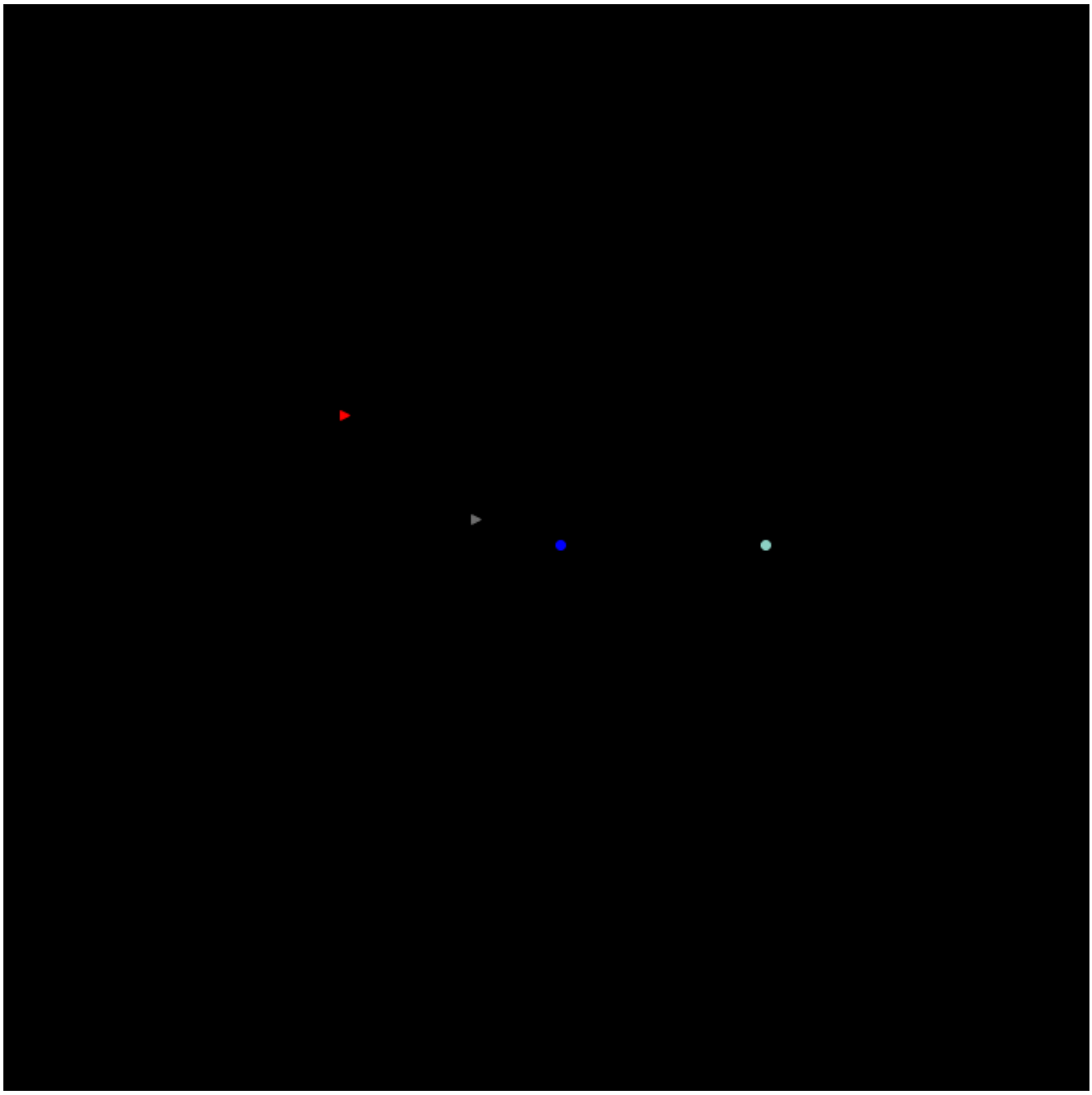
Двадцать дней полета:



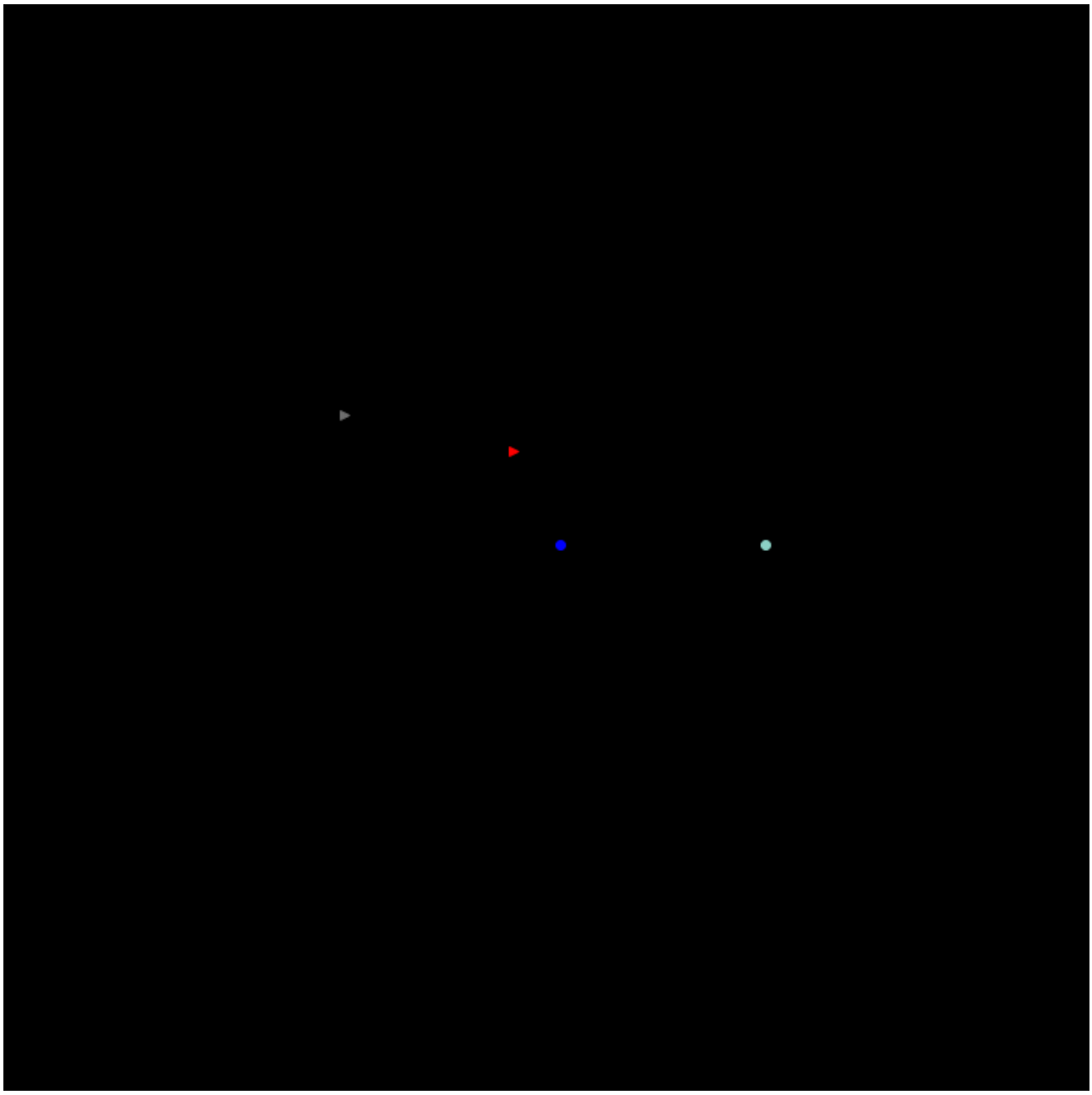
Тридцать дней полета:



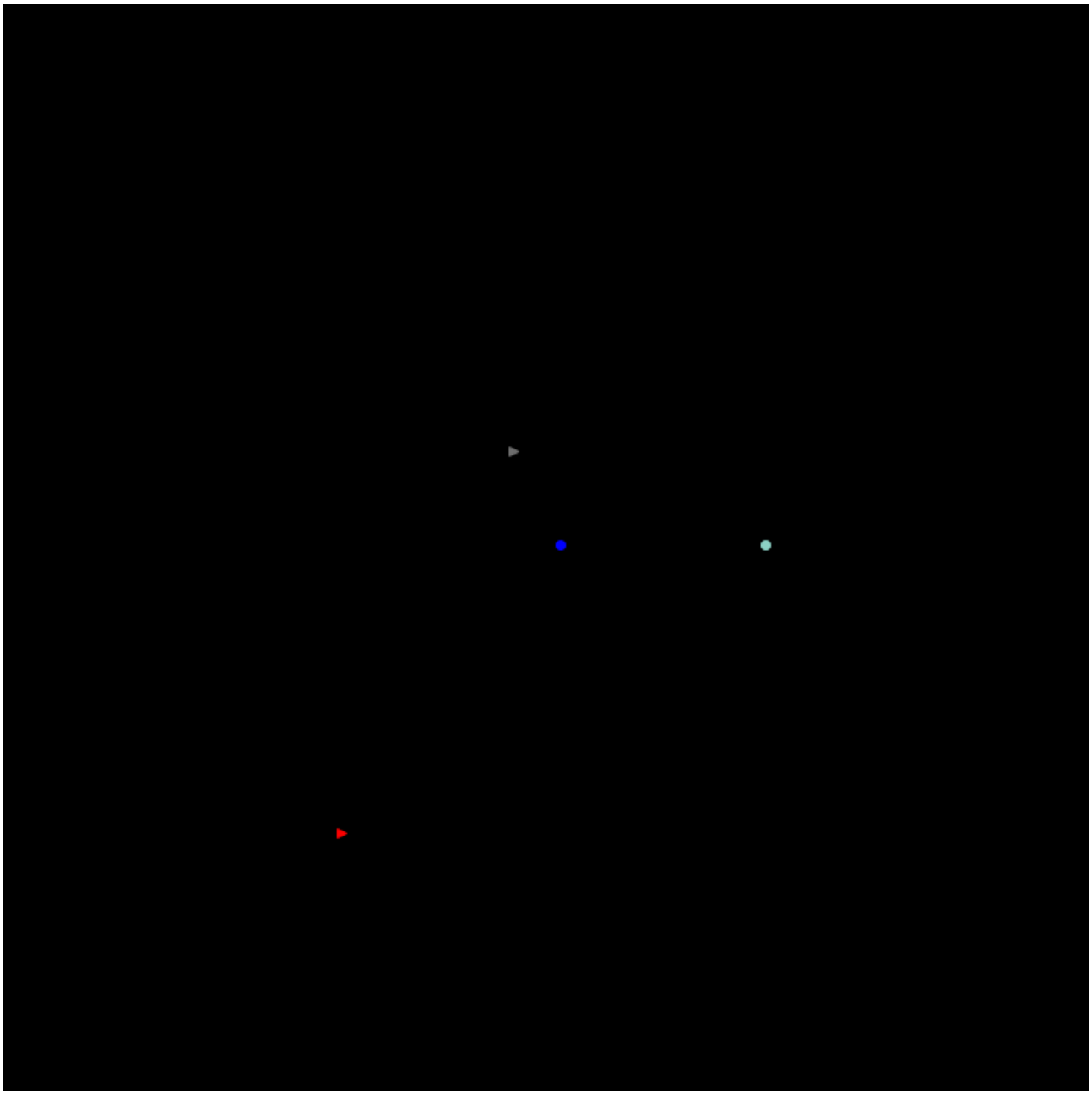
Сорок дней полета:



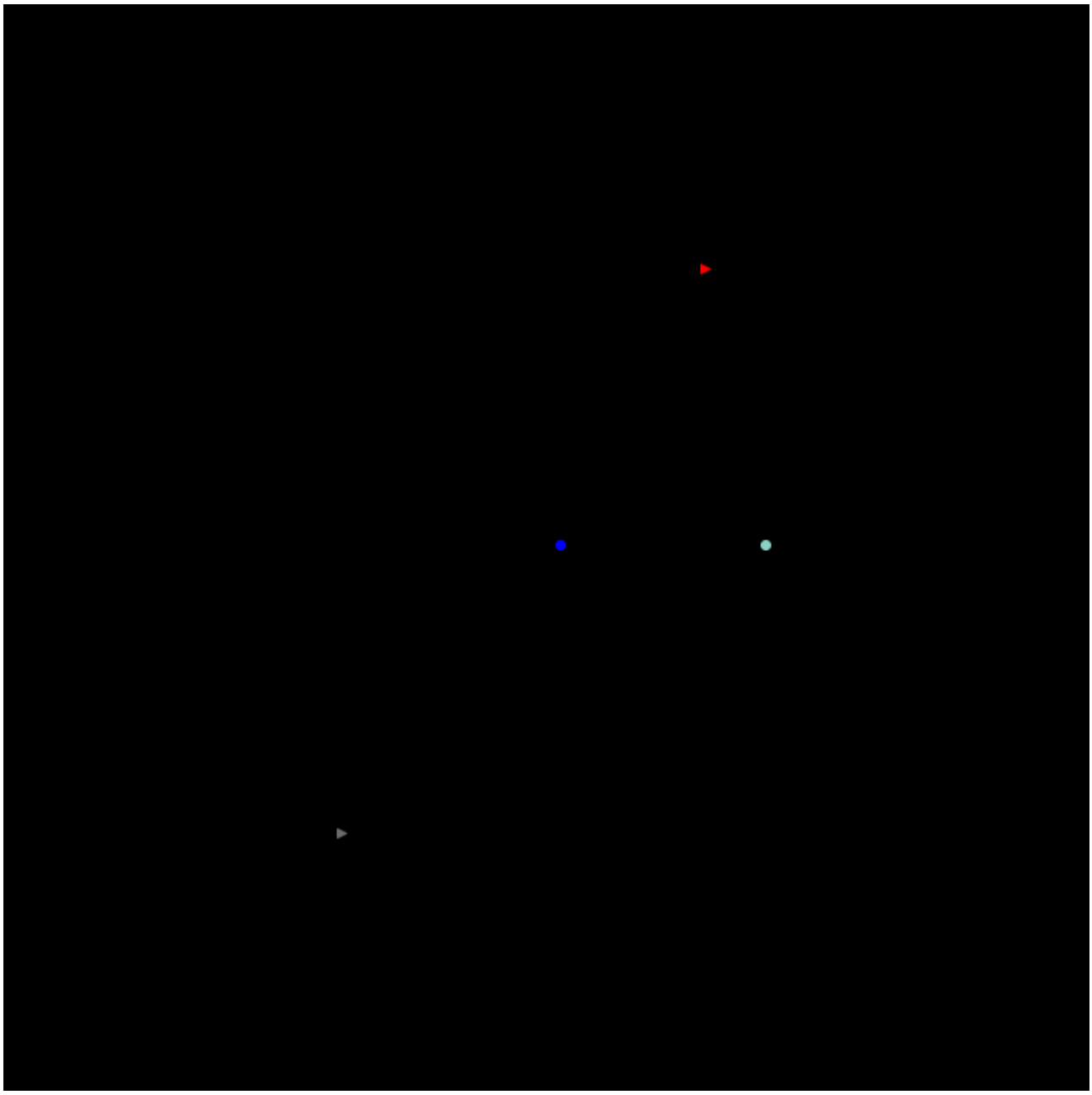
Пятьдесят дней полета:



Шестьдесят дней полета:



Семьдесят дней полета:



Заключение.

В ходе работы было выявлено, что первая задача является выполнимой. Однако для решения второй необходимы дополнительные условия, например, реактивные двигатели способные изменить траекторию полета ракеты.

Список использованных источников.

1. Амосов А. А., Дубинский Ю. А., Копченова Н. В. Уравнения математической физики. — 1999.
2. https://matplotlib.org/stable/api/animation_api.html
3. https://pypi.org/project/Wand/
4. https://matplotlib.org/stable/api/_as_gen/matplotlib.animation.FuncAnimation.html

Приложение

Код программы (python)

```
import ffmpeg
import wand
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.animation import FuncAnimation

m1 = 5.9742e24
m2 = 7.36e22
G = 6.6743015e-11
alpha = np.pi/180*26 #26.3 - 25.5 - Попадание
EarthRadius = 6371000 #м
MoonRadius = 1737400 #м
startPosition = np.array([EarthRadius*np.cos(alpha), EarthRadius*np.sin(alpha)])
Vstart_r = 11200 #м/с
Vmoon_r = 1020 #м/с
MoonDistance = 384400000 #м
startVelocity = np.array([Vstart_r*np.cos(alpha), Vstart_r*np.sin(alpha)])

RVstart = np.array([startPosition, startVelocity])

#dt = np.pi/1183949
#t_r = Пr/V  t = П convert = t_r/t
convert = Vmoon_r/MoonDistance
dt = 1

# Для поворота радиус вектора на определенный угол (против часовой стрелки) реализуем функцию поворота:

def VectorRotate(r, t):
    x_rot = r[0]*np.cos(t*convert) + r[1]*np.sin(t*convert)
    y_rot = -r[0]*np.sin(t*convert) + r[1]*np.cos(t*convert)
    return np.array([x_rot, y_rot])



def MoonPosition(t):
    x = MoonDistance*np.cos(t*convert)
    y = MoonDistance*np.sin(t*convert)
    return np.array([x, y])

MoonPositionStatic = np.array([MoonDistance, 0.0])

def VectorLength(r):
    return np.sqrt(r[0]**2 + r[1]**2)
```

```
def IsInsideMoon(r1, r2):
    return VectorLength(r1 - r2) <= MoonRadius

def IsInsideEarth(r1):
    return VectorLength(r1) <= EarthRadius

def f2(t, r):
    r2 = MoonPosition(t)
    a = G*m1*(-r/VectorLength(r)**3) + G*m2*((r2 - r)/VectorLength(r2 - r)**3)
    return a

def F(t, Y):
    F1 = Y[1]
    F2 = f2(t, Y[0])
    return np.array([F1, F2])

def RungeKutta(t, Y, dt):
    k1 = F(t, Y)
    k2 = F(t + dt/2, Y + dt*k1/2)
    k3 = F(t + dt/2, Y + dt*k2/2)
    k4 = F(t + dt, Y + dt*k3)
    return Y + 1/6*dt*(k1 + 2*k2 + 2*k3 + k4)

def AdamsBashfort4(Y1, Y2, Y3, Y4, t, dt):
    return Y1 + dt/24*(55*F(t, Y1) - 59*F(t - dt, Y2) + 37*F(t - 2*dt, Y3) - 9*F(t - 3*dt, Y4))

t = 0
RV = np.zeros((10000, 2, 2))
RV[0] = RVstart
RVwork = np.zeros((4, 2, 2))
RVwork[0] = RVstart
for i in range(1, 4):
    RVwork[i] = RungeKutta(t, RVwork[i - 1], dt)
    t += dt
print(RVwork)

hours = 0
penetrationTime = 167760
for i in range(150000):
    RVnext = AdamsBashfort4(RVwork[3], RVwork[2], RVwork[1], RVwork[0], t, dt)
    RVwork[0] = RVwork[1]
    RVwork[1] = RVwork[2]
    RVwork[2] = RVwork[3]
    RVwork[3] = RVnext
    if t%3600 == 0:
        hours += 1
```

```
        RV[hours] = RVnext
    t += dt

print(hours)

DeadLine = MoonDistance + 100*MoonRadius
while hours < 24 * 7:
    #print(i)
    RVnext = AdamsBashfort4(RVwork[3], RVwork[2], RVwork[1], RVwork[0], t, dt)
    RVwork[0] = RVwork[1]
    RVwork[1] = RVwork[2]
    RVwork[2] = RVwork[3]
    RVwork[3] = RVnext
    if t%3600 == 0:
        Rend = RVwork[3][0]
        if VectorLength(Rend) > 2*MoonDistance:
            print("ЛА покинул рассматриваемую область")
            break
        hours += 1
        RV[hours] = RVnext
    if t%60 == 0:
        MoonPos = MoonPosition(t)
        Rend = RVwork[3][0]
        # Rend[0] > MoonPos[0] - MoonRadius and Rend[0] < MoonPos[0] + MoonRadius and Rend[1] > MoonPos[1] - MoonRa-
dius and Rend[1] < MoonPos[1] + MoonRadius
        if (IsInsideMoon(Rend, MoonPos)):
            RV[hours + 1] = RVwork[3]
            print("Попадание в Луну")
            print(t)
            print(hours)
            break
        if (IsInsideEarth(Rend)):
            RV[hours + 1] = RVwork[3]
            print("Попадание в Землю")
            print(t)
            print(hours)
            break
        '"if (VectorLength(Rend) > DeadLine):
            print("Промах!")
            break"'
    t += dt
    i+=1

def GetQuarter(r):
    if IsInsideMoon(r, MoonPositionStatic):
        return 'L'
```

```
    elif r[0] > MoonPositionStatic[0] and r[1] > MoonPositionStatic[1]:
        return 'a'
    elif r[0] < MoonPositionStatic[0] and r[1] > MoonPositionStatic[1]:
        return 'b'
    elif r[0] < MoonPositionStatic[0] and r[1] < MoonPositionStatic[1]:
        return 'c'
    elif r[0] > MoonPositionStatic[0] and r[1] < MoonPositionStatic[1]:
        return 'd'
    else:
        return ValueError


DeadLine = MoonDistance + 40*MoonRadius
path = "b"
right_path = "badcb"
alpha1 = np.pi/180*42
alpha2 = np.pi/180*40
alpha = (alpha1+alpha2)/2
startPosition = np.array([EarthRadius*np.cos(alpha), EarthRadius*np.sin(alpha)])
startVelocity = np.array([Vstart_r*np.cos(alpha), Vstart_r*np.sin(alpha)])
RVstart = np.array([startPosition, startVelocity])

while path != "badcb":
    path = "b"
    t = 0
    RV = np.zeros((1000, 2, 2))
    RV[0] = RVstart
    RVwork = np.zeros((4, 2, 2))
    RVwork[0] = RVstart
    for i in range(1, 4):
        RVwork[i] = RungeKutta(t, RVwork[i - 1], dt)
        t += dt

    hours = 0
    penetrationTime = 167760
    for i in range(150000):
        RVnext = AdamsBashfort4(RVwork[3], RVwork[2], RVwork[1], RVwork[0], t, dt)
        RVwork[0] = RVwork[1]
        RVwork[1] = RVwork[2]
        RVwork[2] = RVwork[3]
        RVwork[3] = RVnext
        if t%3600 == 0:
            hours += 1
            RV[hours] = RVnext
        t += dt
```

```
    penetration = False

    while hours < 72:
        #print(i)
        RVnext = AdamsBashfort4(RVwork[3], RVwork[2], RVwork[1], RVwork[0], t, dt)
        RVwork[0] = RVwork[1]
        RVwork[1] = RVwork[2]
        RVwork[2] = RVwork[3]
        RVwork[3] = RVnext
        if t%3600 == 0:
            hours += 1
            RV[hours] = RVnext
        if t%60 == 0:
            Rend = RVwork[3][0]
            RendRot = VectorRotate(Rend, t)
            cur_quarter = GetQuarter(RendRot)
            if path[-1] != cur_quarter:
                path += cur_quarter
            #MoonPos = MoonPosition(t)
            if (IsInsideMoon(RendRot, MoonPositionStatic)):
                RV[hours + 1] = RVwork[3]
                print("Есть пробитие!")
                penetration = True
                break
            if (VectorLength(Rend) > DeadLine):
                print("Промах!")
                break
        t += dt
        i+=1
    print(path)
    print(alpha/np.pi*180)
    if penetration:
        alpha2 = alpha
    else:
        alpha1 = alpha
    alpha = (alpha1+alpha2)/2
    startPosition = np.array([EarthRadius*np.cos(alpha), EarthRadius*np.sin(alpha)])
    startVelocity = np.array([Vstart_r*np.cos(alpha), Vstart_r*np.sin(alpha)])
    RVstart = np.array([startPosition, startVelocity])

from matplotlib.animation import FuncAnimation
import math
scale = MoonDistance / 500
r1 = 500
r2 = 250
```

```
plt.style.use('dark_background')
fig = plt.figure(figsize = (10, 10))
ax = plt.axes(xlim=(-1000, 1000), ylim=(-1000, 1000))
ax.plot(0, 0, '-bo')
plt.axis('off')
ax.set_aspect("equal")
point, = ax.plot(0, r1, marker="o")
point2, = ax.plot(0, r2, "r>")
ind = 0


def update(i):
    ind = round(i/3600)
    #x,y = MoonPosition(i)/scale
    x,y = MoonPositionStatic/scale
    point.set_data([x],[y])
    x2,y2 = VectorRotate(RV[ind][0], i)/scale
    #x2,y2 = RV[ind][0]/scale
    point2.set_data([x2], [y2])
    return point, point2,

ani = FuncAnimation(fig, update, interval=83, blit=True, repeat=True,
                    frames=np.linspace(0, 3600*hours, hours, endpoint=False))
ani.save('OneSmallRocketsJourney1.gif', writer='imagemagick')

plt.show()
```