Отчет по моделированию.

Выполнили:

1 команда:

Селиховкина Е.И.

Мироненко Егор

2 команда:

Хлучин Г.В.

Гумбатов Влад

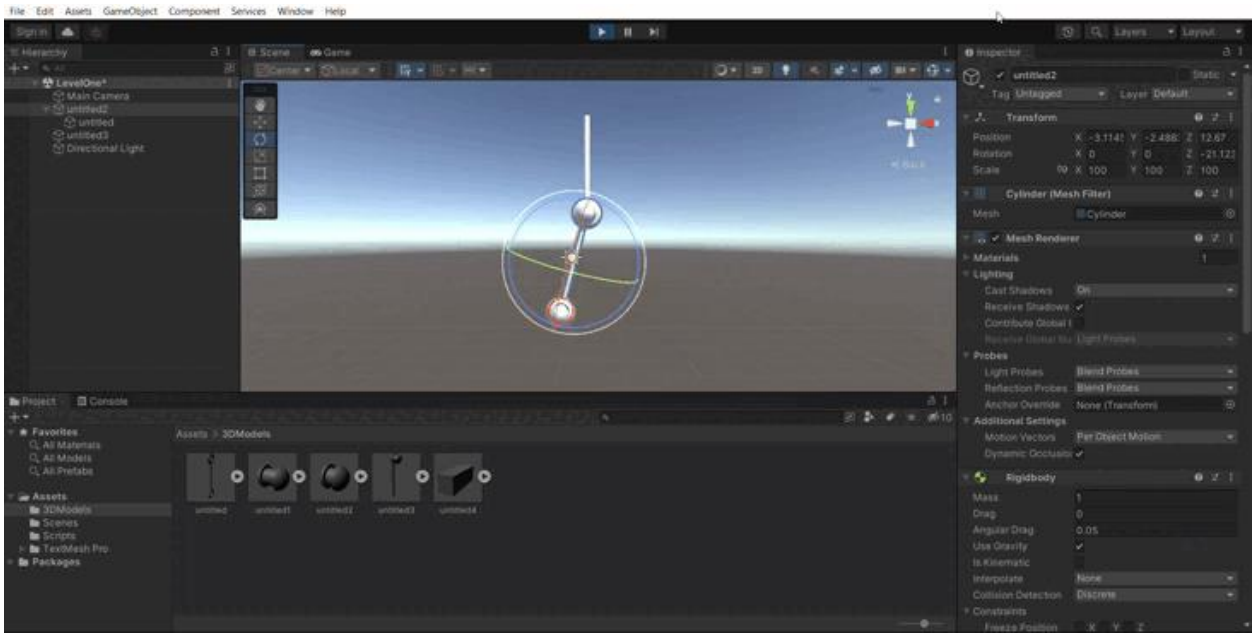
Первая команда выполняла моделирование физического маятника в unity + делала графики к физическому и математическому маятникам.

Вторая команда выполняла моделирование двойного маятника в unity + делал графики движения двойного физического маятника.

Обе команды работали на UI в unity

Отчет первой команды.

Графическая часть была выполнена в unity с помощью средств которые предоставляет движок. Были использованы такие физические события как rigitbody, joint



Формулы:

$\varphi(t) = \varphi_0 * \cos(\omega_0 t + \alpha) -$ уравнение гармонических колебаний, α $-$ начальная фаза колебаний, $\varphi_0 -$ амплитуда колебаний, ω_0 $-$ собственнная циклическая частота

$\omega_0 = \sqrt{\frac{g}{l}} -$ циклическая частота

$T = \frac{2\pi}{\omega_0} -$ период колебаний

$\nu = \frac{1}{2\pi}\sqrt{\frac{g}{l}} -$ частота колебаний

$\mathrm{I} = ml^2 -$ момент инерции

$mL\ddot{\theta} = -mg\sin\theta$

$\ddot{\theta} + \frac{g}{L}\sin\theta = 0$

Дифференциальные уравнения затухающих колебаний

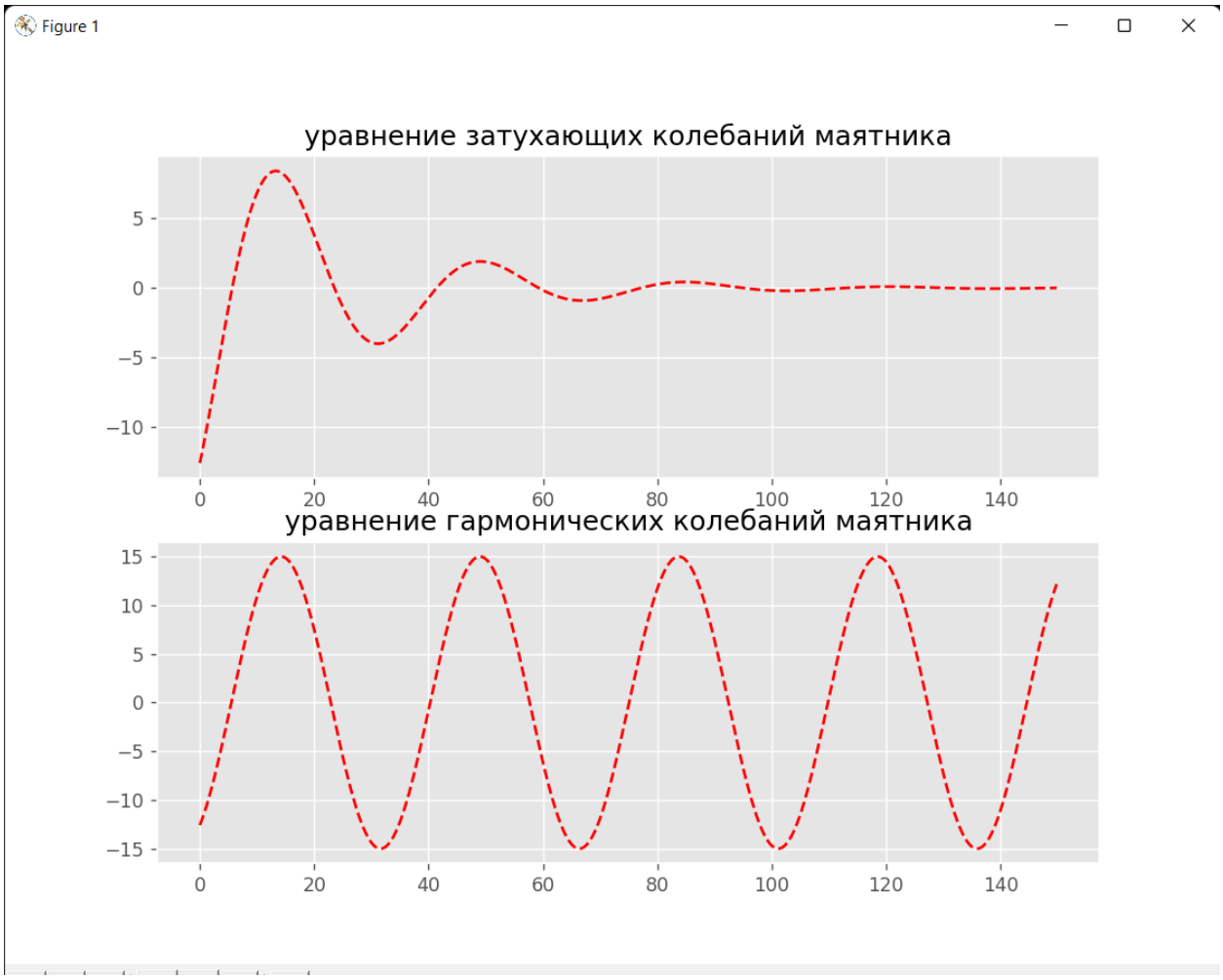
$\frac{d^2x}{dt^2} + 2a\frac{dx}{dt} + \omega_0^2 x = 0$

$x = A\, e^{-Bt} \cos(\omega t + \varphi_0)$

r – коэффициент трения

$\beta = \frac{r}{2m}$ – коэффициент затухания

$\omega = \sqrt{\omega_0^2 - \beta^2}$



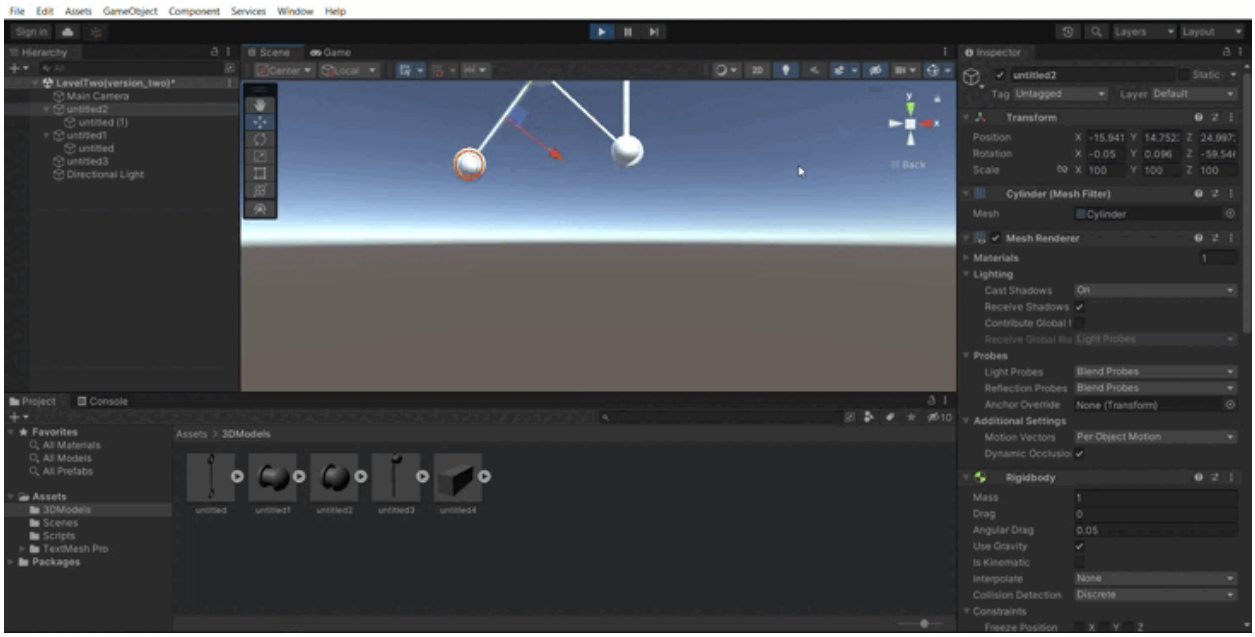
Код:

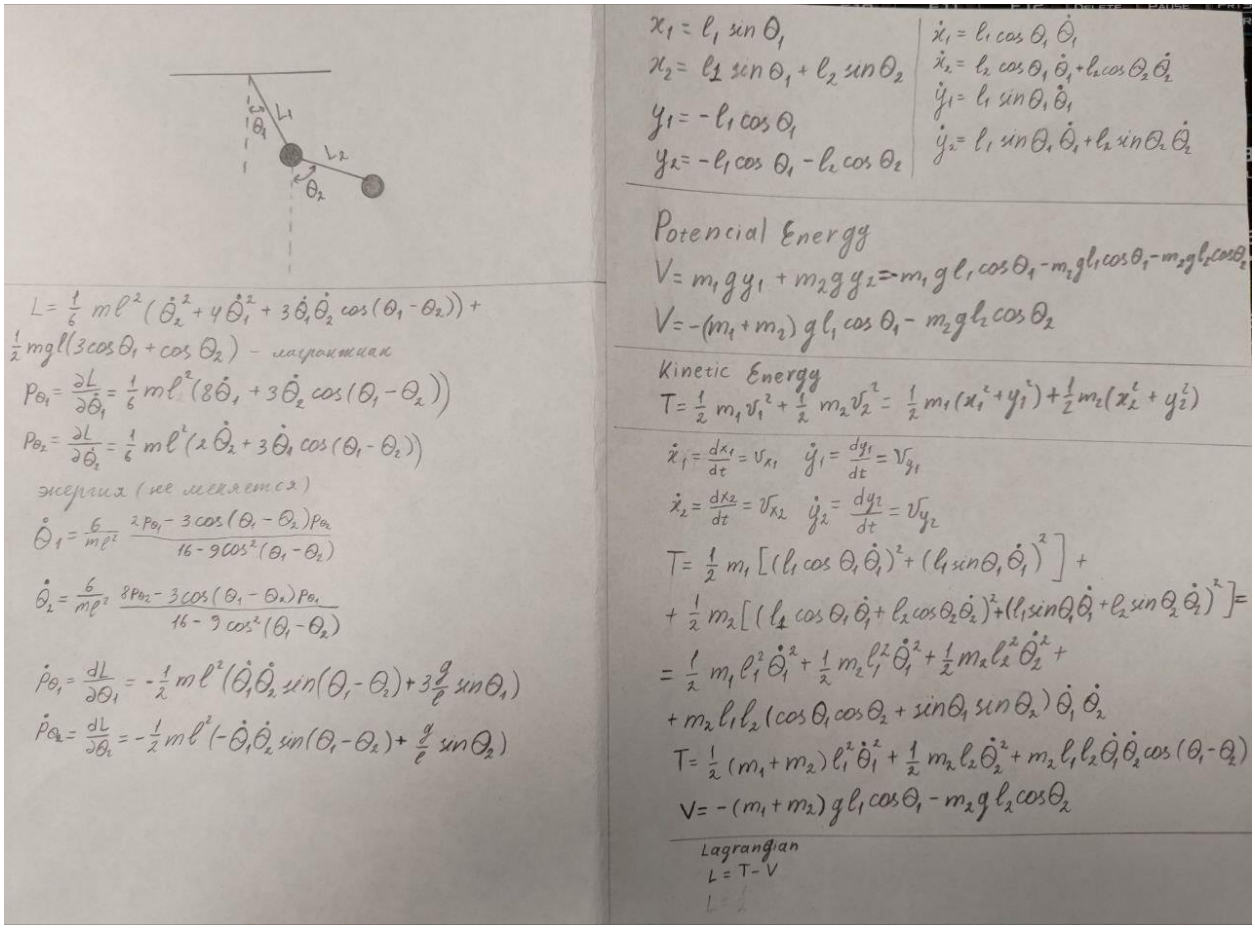
```
import matplotlib.pyplot as plt
import numpy as np
import math

plt.style.use('ggplot')


G = 9.8
e = 2.71

a = 10 #= int(input('введите начальную фазу колебаний: '))
```

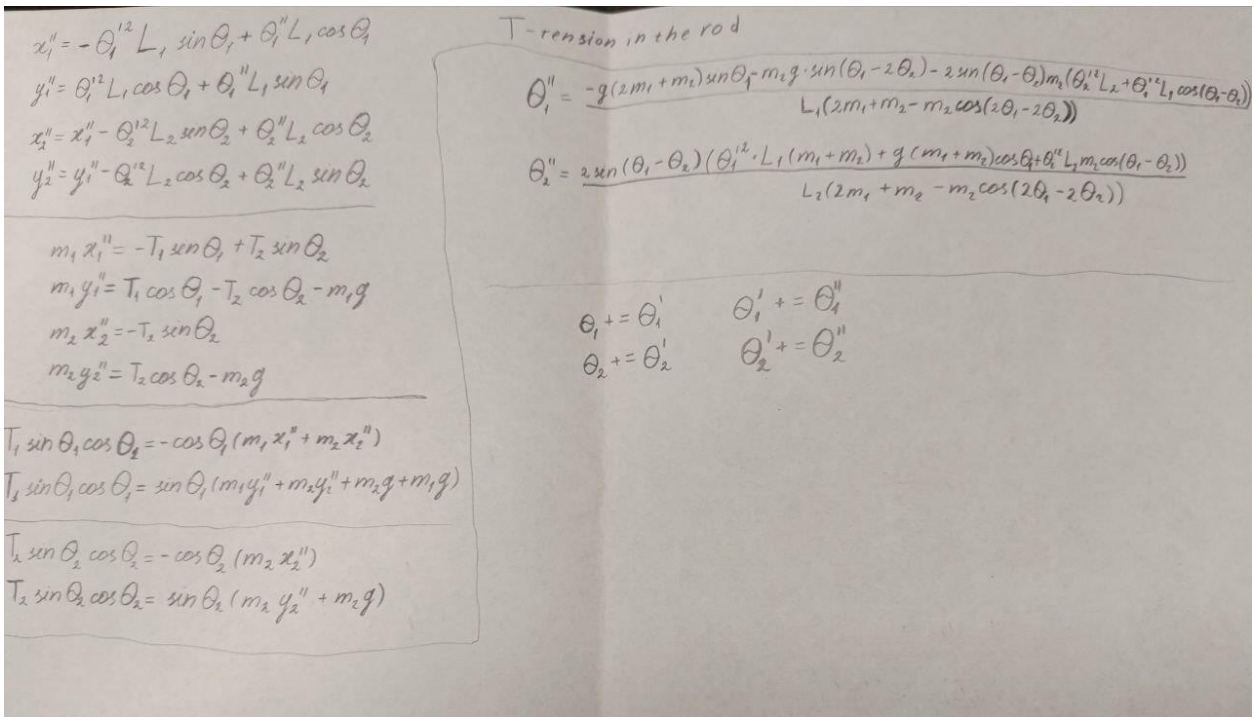
```
l = 300 #= int(input('введите длину стержня: '))

A = 15 #= int(input('введите аммплитуду колебаний: '))

w0 = math.sqrt(G/l)

# массив времени t от 0 секунд до 5
t = np.arange(0., 150., 0.2)

f = A*np.cos(w0*t + a)

r = 1#= int(input('введите коэффициент трения: '))
m = 12 #= int(input('введите массу: '))

#коэффициент затухания
B = r / (2*m)

#циклическая частота
w = math.sqrt((w0**2 - B**2))

#уравнение затухающих колебаний
x = A * (e**(-B*t))*np.cos(w*t + a)

print('Гармонические колебания:')
print('собственная частота = ', w0)
print('период = ', 2 * 3.14 / w0)
print('частота колебаний = ', w0 / (6.28))
print('момент инерции = ', m * l**2)

print()
print('Затухающие колебания:')
print('собственная частота = ', w0)
print('период = ', 6.28 / w)
print('частота колебаний = ', w)
print('коэфициент затухания = ', B)

plt.subplot(2, 1, 1)
# отображение графиков
plt.plot(t, x, 'r--')
plt.title('уравнение затухающих колебаний маятника')

plt.subplot(2, 1, 2)
# отображение графиков  math.cos(w0*t + a)
plt.plot(t, f, 'r--')
plt.title('уравнение гармонических колебаний маятника')

plt.show()
```

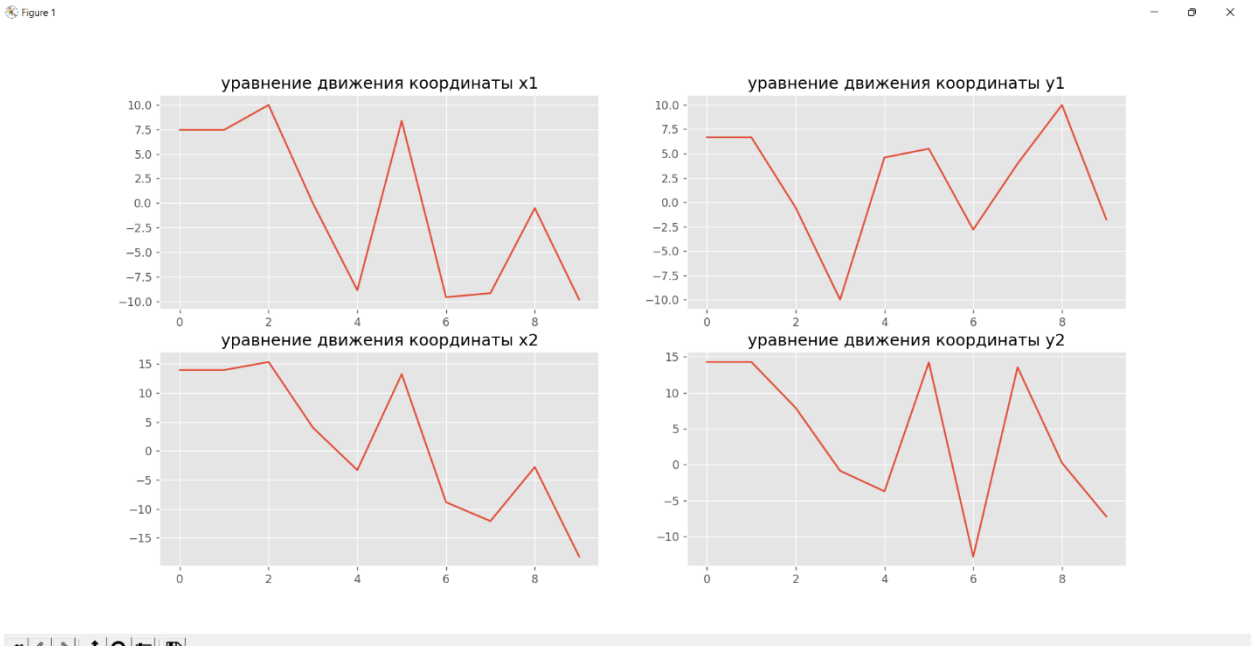
Отчет второй команды.

Графическая часть была выполнена в unity с помощью средств которые предоставляет движок. Были использованы такие физические события как rigitbody, joint


File Edit Assets GameObject Component Services Window Help
Hierarchy
LevelTwo(version_two)*
Main Camera
untitled2
untitled (1)
untitled1
untitled
untitled3
Directional Light
Scene
Game
Inspector
untitled2
Transform
Position
Rotation
Scale
Cylinder (Mesh Filter)
Mesh
Cylinder
Mesh Renderer
Materials
Lighting
Cast Shadows
Receive Shadows
Probes
Light Probes
Reflection Probes
Anchor Override
Additional Settings
Motion Vectors
Per Object Motion
Rigidbody
Mass
Drag
Angular Drag
Use Gravity
Is Kinematic
Interpolate
Collision Detection
Discrete
Constraints
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
3DModels
Scenes
Scripts
TextMesh Pro
Packages
Assets > 3DModels
untitled
untitled1
untitled2
untitled3
untitled4


L1
θ1
L2
θ2
$x_1 = l_1 \sin\theta_1$
$x_2 = l_1 \sin\theta_1 + l_2 \sin\theta_2$
$y_1 = -l_1 \cos\theta_1$
$y_2 = -l_1 \cos\theta_1 - l_2 \cos\theta_2$
$\dot{x}_1 = l_1 \cos\theta_1 \dot{\theta}_1$
$\dot{x}_2 = l_2 \cos\theta_1 \dot{\theta}_1 + l_2 \cos\theta_2 \dot{\theta}_2$
$\dot{y}_1 = l_1 \sin\theta_1 \dot{\theta}_1$
$\dot{y}_2 = l_1 \sin\theta_1 \dot{\theta}_1 + l_2 \sin\theta_2 \dot{\theta}_2$
Potencial Energy
$V = m_1 g y_1 + m_2 g y_2 = -m_1 g l_1 \cos\theta_1 - m_2 g l_1 \cos\theta_1 - m_2 g l_2 \cos\theta_2$
$V = -(m_1 + m_2) g l_1 \cos\theta_1 - m_2 g l_2 \cos\theta_2$
Kinetic Energy
$T = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_1 (x_1^2 + y_1^2) + \frac{1}{2} m_2 (x_2^2 + y_2^2)$
$\dot{x}_1 = \frac{dx_1}{dt} = v_{x1}$ $\dot{y}_1 = \frac{dy_1}{dt} = v_{y1}$
$\dot{x}_2 = \frac{dx_2}{dt} = v_{x2}$ $\dot{y}_2 = \frac{dy_2}{dt} = v_{y2}$
$T = \frac{1}{2} m_1 \left[(l_1 \cos\theta_1 \dot{\theta}_1)^2 + (l_1 \sin\theta_1 \dot{\theta}_1)^2 \right] +$
$+ \frac{1}{2} m_2 \left[(l_1 \cos\theta_1 \dot{\theta}_1 + l_2 \cos\theta_2 \dot{\theta}_2)^2 + (l_1 \sin\theta_1 \dot{\theta}_1 + l_2 \sin\theta_2 \dot{\theta}_2)^2 \right] =$
$= \frac{1}{2} m_1 l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 +$
$+ m_2 l_1 l_2 (\cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2) \dot{\theta}_1 \dot{\theta}_2$
$T = \frac{1}{2} (m_1 + m_2) l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2 \dot{\theta}_2^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2)$
$V = -(m_1 + m_2) g l_1 \cos\theta_1 - m_2 g l_2 \cos\theta_2$
Lagrangian
$L = T - V$
$L = \frac{1}{6} m l^2 (\dot{\theta}_2^2 + 4\dot{\theta}_1^2 + 3\dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2)) +$
$\frac{1}{2} mgl(3\cos\theta_1 + \cos\theta_2)$ – лагранжиан
$P_{\theta_1} = \frac{\partial L}{\partial \dot{\theta}_1} = \frac{1}{6} m l^2 (8\dot{\theta}_1 + 3\dot{\theta}_2 \cos(\theta_1 - \theta_2))$
$P_{\theta_2} = \frac{\partial L}{\partial \dot{\theta}_2} = \frac{1}{6} m l^2 (2\dot{\theta}_2 + 3\dot{\theta}_1 \cos(\theta_1 - \theta_2))$
энергия (не меняется)
$\dot{\theta}_1 = \frac{6}{ml^2} \frac{2P_{\theta_1} - 3\cos(\theta_1 - \theta_2) P_{\theta_2}}{16 - 9\cos^2(\theta_1 - \theta_2)}$
$\dot{\theta}_2 = \frac{6}{ml^2} \frac{8P_{\theta_2} - 3\cos(\theta_1 - \theta_2) P_{\theta_1}}{16 - 9\cos^2(\theta_1 - \theta_2)}$
$\dot{P}_{\theta_1} = \frac{dL}{\partial \theta_1} = -\frac{1}{2} m l^2 (\dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) + 3\frac{g}{l} \sin\theta_1)$
$\dot{P}_{\theta_2} = \frac{dL}{\partial \theta_1} = -\frac{1}{2} m l^2 (-\dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) + \frac{g}{l} \sin\theta_2)$





Код:

```
import numpy as np
import sympy as smp
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from matplotlib import animation
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import PillowWriter
import decimal


class DoublePendulum:

    def __init__(self):
        self.l1 = 10
        self.l2 = 10
```

```
        self.m1 = 12
        self.m2 = 8
        self.theta1 = 40
        self.theta2 = 15
        self.theta1_ = 0
        self.theta2_ = 0
        self.g = 9.8


    def get_coordinates(self, t):
        numerator = (-self.g*(2*self.m1+self.m2)*math.sin(self.theta1) -
self.m2*self.g*math.sin(self.theta1-2*self.theta2) - 2*math.sin(self.theta1-
self.theta2)*self.m2*((self.theta2_**2)*self.l2+(self.theta1_**2)*self.l1*mat
h.cos(self.theta1-self.theta2)))
        denominator = (self.l1*(2*self.m1+self.m2-
self.m2*math.cos(2*self.theta1-2*self.theta2)))

        theta1__ = numerator / denominator

        numerator = (2*math.sin(self.theta1-
self.theta2)*((self.theta1_**2)*self.l1*(self.m1+self.m2)+self.g*(self.m1+sel
f.m2)*math.cos(self.theta1)+(self.theta2_**2)*self.l2*self.m2*math.cos(self.t
heta1-self.theta2)))
        denominator = (self.l2*(2*self.m1+self.m2-
self.m2*math.cos(2*self.theta1-2*self.theta2)))

        theta2__ = numerator / denominator

        x1 = self.l1 * math.sin(self.theta1)
        y1 = -self.l1 * math.cos(self.theta1)

        x2 = x1 + self.l2 * math.sin(self.theta2)
        y2 = y1 - self.l2 * math.cos(self.theta2)

        self.theta1 += self.theta1_
        self.theta2 += self.theta2_

        self.theta1_ += theta1__
        self.theta2_ += theta2__

        return [
            x1,
            y1,
            x2,
            y2,
        ]



plt.style.use('ggplot')

t = np.arange(0., 10, 1)

double_pendulum = DoublePendulum()

x = np.sin(t)
arr_x1 = []
arr_y1 = []
arr_x2 = []
arr_y2 = []



for i in t:
```

```
    mass = double_pendulum.get_coordinates(t)
    arr_x1.append(mass[0])
    arr_y1.append(mass[1])
    arr_x2.append(mass[2])
    arr_y2.append(mass[3])


plt.subplot(2, 2, 1)
plt.plot(t, np.array(arr_x1))
plt.title('уравнение движения координаты x1')

plt.subplot(2, 2, 2)
plt.plot(t, np.array(arr_y1))
plt.title('уравнение движения координаты y1')

plt.subplot(2, 2, 3)
plt.plot(t, np.array(arr_x2))
plt.title('уравнение движения координаты x2')

plt.subplot(2, 2, 4)
plt.plot(t, np.array(arr_y2))
plt.title('уравнение движения координаты y2')

plt.show()
```