Spadek Swobodny

Autorzy: Marek
Milan

Lp.	Zadanie	Czas[h]			
		optymistyczny	pesymistyczny	prawdopodobny	estymata
1	Model	1	3	2	2
2	Rysunek sytuacyjny	1	3	2	2
3	Symulacja	3	5	4	4
4	Animacja	4	8	6	6

Table 1: Harmonogram zadań

1) **Opis projektu.**

Projekt polegał na symulacji spadku swobodnego ciała z pewnego punktu w przestrzeni i jego odbicia od powierzchni w czasie ciągłym. Obliczenia zostały wykonane dzieki bibliotece Numpy oraz Scipy. Po zasymulowaniu toru ruchu należało zrealizować animację przy użyciu dowolnej biblioteki w języku Python. W tym celu posłużono się popularną biblioteką matplotlib.

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

Zrzut ekranu 1. Importowane biblioteki

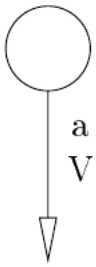
```
Ho=10.0
g=9.81
r=Ho*0.01
```

Zrzut ekranu 2. Inicjalizacja stałych

2) **Rysunek sytuacyjny.**

Projekt został zrealizowany poprzez uwzględnienie dwóch sytuacji: ruch kulki odbywa się na zmianę w dół lub w górę. W obu tych sytuacjach wektory sił grawitacji, a zarazem przyspieszenia działają w dół, zaś wektory prędkości mają przeciwne zwroty, dla ruchu w dół działają w kierunku do ziemi, a dla ruchu w górę działają do góry. W momencie zmiany kierunku ruchu następuje zmniejszenie prędkości do 80 % jej wartości, co ma symulować wytracanie energii w trakcie odbicia. a = przyspieszenie . V = prędkość.

3.1 Ruch ciała w dół.



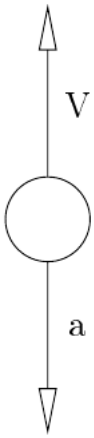
Rysunek 1 : Spadek

```
h1=Ho
v1=0.0
state=[h1,v1]

y = integrate.odeint(derivs, state, t)
y = [[e,s] for e,s in y if e > 0]
```

Zrzut ekranu 3. Inicjalizacja, obliczanie wartości położenia i prędkości w chwilach czasu do momentu uderzenia w ziemię

3.2 Ruch ciała w górę.



Rysunek 2 : Wznoszenie

```
h2=0.0
v2=-0.8*y[-1][1]

state=[h2,v2]
y2 = integrate.odeint(derivs, state, t)

y2 = [[e,s] for e,s in y2 if e > 0]
```

Zrzut ekranu 4. Inicjalizacja, obliczanie wartości położenia i prędkości w chwilach czasu dopóki prędkość jest większa od 0.

3) **Symulacja w czasie ciągłym**

W celu obliczania zmiennych stanu, wskazujących obecne położenie i prędkość ciała w przestrzeni posłużono się poniższymi równaniami różniczkowymi, w których x1 = położenie, x2 = prędkość, $\frac{dx2}{dt}$ = przyspieszenie.

$$\frac{dx_1}{dt} = x_2$$

$$\frac{dx_2}{dt} = -g$$

```
def derivs(state, t):
    dydx = np.zeros_like(state)
    dydx[0] = state[1]
    dydx[1] =-g

    return dydx

dt = 0.05
t = np.arange(0.05, 10, dt)
```

Zrzut ekranu 2. Zmienne stanu zapisane w funkcji obliczającej ich wartości w danych chwilach czasu

4) **Animacja**

Zrzut ekranu poniżej przedstawia fragment kodu programu wykonujący animowanie obiektu na podstawie obliczonych wcześniej zmiennych stanu dla każdej chwili czasu w obliczonym przedziale.

```
y= y+ y2 +y3
print(y)
x1=[0 for e,s in y]
y1=[e for e,s in y]
fig = plt.figure()
ax = fig.add_subplot(111, autoscale_on=False, xlim=(-0.6*Ho, 0.6*Ho), ylim=(0, Ho),frameon=False,picker=None)


line, = ax.plot([], [], '', lw=2)
time_template = 'time = %.1fs'
time_text = ax.text(0.05, 0.9, '', transform=ax.transAxes)

def init():
    line.set_data([], [])
    time_text.set_text('')
    return line, time_text

def animate(i):
    circle = plt.Circle((0, y1[i]), r , fc='green')
    plt.gca().add_patch(circle)
    time_text.set_text(time_template % (i*dt))
    return circle, time_text

ani = animation.FuncAnimation(fig, animate, np.arange(1, len(y)), interval=25, blit=True, init_func=init)
ani.save('spadekswobodny.htm', fps=15)

plt.show()
```

Zrzut ekranu 5. Fragment kodu odpowiedzialny za wygenerowanie animacji na podstawie obliczonych danych.

Wyjaśnienie kodu:
-w pierwszej linii kodu następuje zsumowanie oblczonych wcześniej położeń
-tworzy się wektor x1 o tych samych wymiarach co y, z zaincijowanymi wartościami na wartość równą 0, ponieważ ruch odbywa sie tylko w osi Y
-tworzy się wektor y1 zawierający współrzędnie w osi Y
-generuje się okno o zadanych parametrach: 111 - 1. wykres z siatką o wymiarach 1x1, brak autoskalowania, wyświetlane przedziały osi X i Y , wyłączenie widoku osi, picker wyłączony)
-narysowanie wykresu
- zapis obecnego czasu
- narysowanie w tym samym oknie obecnego czasu
- inicjalizacja animacji
- funkcja służąca do wykonania animacji, obiekt koła zostaje zainicjowany z danymi parametrami, a następnie dodany do wykresu. Funkcja zwraca dla każdej kolejnej wartości iteratora "i" zielone kółko , o zadanym promieniu w konkretnych miejscach na wykresie, a także generuje tekst przedstawiający obecny czas.
-następuje wywołanie animacji w oknie "fig", wykonanie funkcji animacji dla wszystkich parametrów y,
- zapis do pliku i prezentacja animacji