\

Symulacja Masy Sprężystej z Użyciem Tkinter i Matplotlib

Ten artykuł wyjaśnia działanie interaktywnej symulacji fizycznej przedstawiającej spadającą kulę, uwzględniającej siły takie jak grawitacja, tarcie i wiatr. Symulacja ta została zaimplementowana w Pythonie z użyciem bibliotek **Tkinter** i **Matplotlib**. Omówimy jej działanie, wspierając się wzorami fizycznymi oraz blokami kodu.

\

Koncepcja Fizyczna

Symulowana jest cząstka o masie \$m = 0.15\$ kg poruszająca się w dwuwymiarowej przestrzeni pod wpływem:

* siły ciążenia:

$$ F_g = m \cdot g $$

* siły wiatru (stała, zadawana suwakiem):

$$ F_w = \text{stała} \in \langle -2, 2 \rangle \ \text{[N]} $$

* siły tarcia (działającej przy kontakcie z podłożem):

$$ F_t = \mu \cdot m \cdot g $$

Ruch cząstki opisany jest przez drugą zasadę dynamiki Newtona:

$$ \vec{a} = \frac{\vec{F}}{m} $$

\

Inicjalizacja Symulacji

```
def reset_simulation():
    global x, y, vx, vy, trace, sim_time, time_list, speed_list, acc_list
    x, y = 0, 30  # Start higher in the bigger graph
    vx, vy = 0, 0  # Reset velocities
    trace = []     # Reset trace
    sim_time = 0.0
    time_list = []
    speed_list = []
```

```
    acc_list = []
```

Tutaj ustalamy początkowe warunki: kulka znajduje się na wysokości 30 m i nie porusza się (zerowe prędkości). Resetowane są także listy do wykresów czasowych.

\

Obliczenia Dynamiki Ruchu

```
ax_acc = Fw / m      # Horizontal acceleration due to wind
ay_acc = -g          # Vertical acceleration (gravity)

vx += ax\_acc \* time\_step
vy += ay\_acc \* time\_step

x += vx \* time\_step + 0.5 \* ax\_acc \* time\_step**2
y += vy \* time\_step + 0.5 \* ay\_acc \* time\_step**2
```

Przy każdym kroku czasowym obliczane są przyspieszenia ze wzoru:

$$ a_x = \frac{F_w}{m}, \quad a_y = -g $$

Po czym aktualizujemy prędkości i pozycje na podstawie równań ruchu:

$$ \vec{v}_{t+\Delta t} = \vec{v}_t + \vec{a} \cdot \Delta t $$

$$ \vec{r}_{t+\Delta t} = \vec{r}_t + \vec{v} \cdot \Delta t + \frac{1}{2} \vec{a} \cdot \Delta t^2 $$

\

Odbicia i Tarcie

```
if y <= y_min:
    y = y_min
    vy = -vy * restitution
    ...
    friction_acc = friction_coefficient * g
```

Kiedy kulka uderza w ziemię, następuje odbicie ze współczynnikiem sprężystości (domyślnie 0.7):

$$ v_y \rightarrow -v_y \cdot e $$

Jeśli prędkość pionowa jest niska, dodawane jest przyspieszenie tarcia, redukujące prędkość poziomą:

$$ a_{\text{tarcie}} = \mu \cdot g $$

\

Wektory Sił

Symulacja przedstawia wektory: grawitacji, wiatru i prędkości.

```
draw_arrow(x, y, 0, -1, 'blue')      # Gravity
...
```

Dodatkowo, rysowane są na wykresie biegunowym (polar plot):

```
draw_polar_arrow(polar_ax, gravity_angle, gravity_magnitude, 'blue',
'Gravity')
```

Wartości te są konwertowane na postać biegunową:

$$ \theta = \arctan2(v_y, v_x), \quad r = |\vec{v}| $$

\

Wykresy Prędkości i Przyspieszenia

```
line_speed.set_data(time_list, speed_list)
line_acc.set_data(time_list, acc_list)
```

Podczas symulacji zapisywana jest prędkość:

$$ |□ec{v}| = \sqrt{v_x^2 + v_y^2} $$

i wartość przyspieszenia:

$$ |\vec{a}| = \sqrt{a_x^2 + a_y^2} $$

Są one rysowane w czasie jako funkcje.

\

Interfejs Użytkownika

Tkinter obsługuje:

* suwaki do zmiany siły wiatru i współczynnika sprężystości * przycisk resetu symulacji * pole tekstowe z informacjami w czasie rzeczywistym

```
wind_slider = tk.Scale(...)
spring_slider = tk.Scale(...)
```

\

Podsumowanie

Symulacja w przystępny sposób ukazuje podstawowe zjawiska fizyczne takie jak siła ciążenia, tarcie, odbicia i ruch z siłą zewnętrzną. Wykorzystuje do tego animację, wykresy i reprezentacje wektorowe. Kod można łatwo rozbudować o dodatkowe efekty fizyczne lub rozszerzyć na inne układy mechaniczne.