Практическая работа 6
Солнечная система

```
import flet as ft
import math
import threading
import time

class Planet:
    def __init__(self, name, orbit_radius, speed, size, color, distance_km,
orbital_period):
        self.name = name
        self.orbit_radius = orbit_radius
        self.speed = speed
        self.size = size
        self.color = color
        self.angle = 0
        self.distance_km = distance_km
        self.orbital_period = orbital_period
        self.x = 0
        self.y = 0
        self.control = None
        self.orbit_control = None

class SolarSystem:
    def __init__(self, page: ft.Page):
        self.page = page
        self.page.title = "Солнечная система"
        self.page.window.width = 1100
        self.page.window.height = 750

        self.running = False
        self.speed_multiplier = 1.0
        self.zoom = 1.0
        self.center_x = 350
        self.center_y = 300

        self.planets = []
        self.stack_controls = []
        self.init_planets()
        self.setup_ui()
```

```
def init_planets(self):
    planets_data = [
        ("Меркурий", 0.8, 0.04, 6, "#A0522D", 57.9, 88),
        ("Венера", 1.2, 0.03, 9, "#DEB887", 108.2, 225),
        ("Земля", 1.6, 0.025, 10, "#4169E1", 149.6, 365),
        ("Марс", 2.0, 0.02, 7, "#CD5C5C", 227.9, 687),
        ("Юпитер", 2.8, 0.012, 20, "#DAA520", 778.5, 4333),
        ("Сатурн", 3.4, 0.009, 17, "#F4A460", 1432, 10759),
        ("Уран", 4.0, 0.006, 12, "#87CEEB", 2867, 30687),
        ("Нептун", 4.5, 0.005, 11, "#4682B4", 4515, 60190)
    ]

    for i, (name, orbit, speed, size, color, dist, period) in enumerate(planets_data):
        planet = Planet(name, orbit, speed, size, color, dist, period)
        planet.angle = i * 45
        self.planets.append(planet)

def setup_ui(self):
    self.info_text = ft.Text("", size=12, color="white")

    self.speed_slider = ft.Slider(
        min=0.1,
        max=5.0,
        divisions=49,
        label="Скорость: {value}x",
        value=1.0,
        on_change=self.on_speed_change
    )

    self.zoom_slider = ft.Slider(
        min=0.5,
        max=2.0,
        divisions=15,
        label="Масштаб: {value}x",
        value=1.0,
        on_change=self.on_zoom_change
    )

    self.planet_info = ft.Text("", size=11, color="cyan")

    for planet in self.planets:
        orbit_r = planet.orbit_radius * 60 * self.zoom
        planet.orbit_control = ft.Container(
```

```
            width=orbit_r * 2,
            height=orbit_r * 2,
            border=ft.border.all(1, "#333333"),
            border_radius=orbit_r,
            left=self.center_x - orbit_r,
            top=self.center_y - orbit_r
        )
        self.stack_controls.append(planet.orbit_control)

    self.sun_control = ft.Container(
        content=ft.Text("☀", size=30, color="yellow"),
        width=40,
        height=40,
        border_radius=20,
        bgcolor="orange",
        left=self.center_x - 20,
        top=self.center_y - 20,
        alignment=ft.alignment.center
    )
    self.stack_controls.append(self.sun_control)

    for planet in self.planets:
        planet.control = ft.Container(
            content=ft.Text(planet.name[0], size=8, color="white"),
            width=planet.size,
            height=planet.size,
            border_radius=planet.size // 2,
            bgcolor=planet.color,
            left=self.center_x,
            top=self.center_y,
            alignment=ft.alignment.center,
            on_click=lambda e, p=planet: self.show_planet_info(p)
        )
        self.stack_controls.append(planet.control)

    self.space_stack = ft.Stack(
        width=700,
        height=600,
        controls=self.stack_controls
    )

    self.space_area = ft.Container(
        content=self.space_stack,
```

```
            width=700,
            height=600,
            bgcolor="black"
        )

        start_btn = ft.ElevatedButton("Старт", on_click=self.start_simulation,
bgcolor="green")
        stop_btn = ft.ElevatedButton("Стоп", on_click=self.stop_simulation,
bgcolor="red")
        reset_btn = ft.ElevatedButton("Сброс", on_click=self.reset_simulation)

        left_panel = ft.Container(
            content=ft.Column([
                ft.Text("Солнечная система", size=20, weight=ft.FontWeight.BOLD,
color="white"),
                self.space_area,
                ft.Row([start_btn, stop_btn, reset_btn]),
                ft.Text("Скорость симуляции:", color="white"),
                self.speed_slider,
                ft.Text("Масштаб:", color="white"),
                self.zoom_slider
            ]),
            padding=10,
            bgcolor="#1a1a2e"
        )

        right_panel = ft.Container(
            content=ft.Column([
                ft.Text("Информация о планетах", size=18, weight=ft.FontWeight.BOLD,
color="white"),
                self.planet_info,
                ft.Divider(color="grey"),
                ft.Text("Расчёты:", size=16, weight=ft.FontWeight.BOLD, color="white"),
                self.info_text
            ]),
            padding=10,
            width=350,
            bgcolor="#1a1a2e"
        )

        self.page.add(ft.Row([left_panel, right_panel], expand=True))
        self.update_planet_positions()
        self.update_info()
```

```
    def show_planet_info(self, planet):
        self.planet_info.value = f"""
Планета: {planet.name}
Расстояние от Солнца: {planet.distance_km} млн км
Орбитальный период: {planet.orbital_period} дней
Текущий угол: {planet.angle:.1f}°
Позиция X: {planet.x:.1f}
Позиция Y: {planet.y:.1f}
"""
        self.planet_info.update()

    def on_speed_change(self, e):
        self.speed_multiplier = float(e.control.value)

    def on_zoom_change(self, e):
        self.zoom = float(e.control.value)
        self.update_orbits()
        self.update_planet_positions()

    def update_orbits(self):
        for planet in self.planets:
            orbit_r = planet.orbit_radius * 60 * self.zoom
            planet.orbit_control.width = orbit_r * 2
            planet.orbit_control.height = orbit_r * 2
            planet.orbit_control.border_radius = orbit_r
            planet.orbit_control.left = self.center_x - orbit_r
            planet.orbit_control.top = self.center_y - orbit_r

    def start_simulation(self, e):
        if not self.running:
            self.running = True
            threading.Thread(target=self.simulation_loop, daemon=True).start()

    def stop_simulation(self, e):
        self.running = False

    def reset_simulation(self, e):
        self.running = False
        for i, planet in enumerate(self.planets):
            planet.angle = i * 45
        self.speed_multiplier = 1.0
        self.zoom = 1.0
```

```
    self.speed_slider.value = 1.0
    self.zoom_slider.value = 1.0
    self.update_orbits()
    self.update_planet_positions()
    self.speed_slider.update()
    self.zoom_slider.update()

def simulation_loop(self):
    while self.running:
        for planet in self.planets:
            planet.angle += planet.speed * self.speed_multiplier
            if planet.angle >= 360:
                planet.angle -= 360
        self.update_planet_positions()
        self.update_info()
        time.sleep(0.05)

def update_planet_positions(self):
    for planet in self.planets:
        orbit_r = planet.orbit_radius * 60 * self.zoom
        planet.x = self.center_x + math.cos(math.radians(planet.angle)) * orbit_r
        planet.y = self.center_y + math.sin(math.radians(planet.angle)) * orbit_r

        planet.control.left = planet.x - planet.size * self.zoom / 2
        planet.control.top = planet.y - planet.size * self.zoom / 2
        planet.control.width = max(6, planet.size * self.zoom)
        planet.control.height = max(6, planet.size * self.zoom)

    self.space_stack.update()

def update_info(self):
    info = "Положения планет:\n\n"
    for planet in self.planets:
        orbit_r_km = planet.distance_km
        x_km = math.cos(math.radians(planet.angle)) * orbit_r_km
        y_km = math.sin(math.radians(planet.angle)) * orbit_r_km
        info += f"{planet.name}:\n"
        info += f"  X: {x_km:.1f} млн км\n"
        info += f"  Y: {y_km:.1f} млн км\n"
        info += f"  Угол: {planet.angle:.1f}°\n\n"

    self.info_text.value = info
    self.info_text.update()
```

```
def main(page: ft.Page):
    SolarSystem(page)

ft.app(target=main)
```

Теперь посмотрим на работу приложения на рисунке 1:

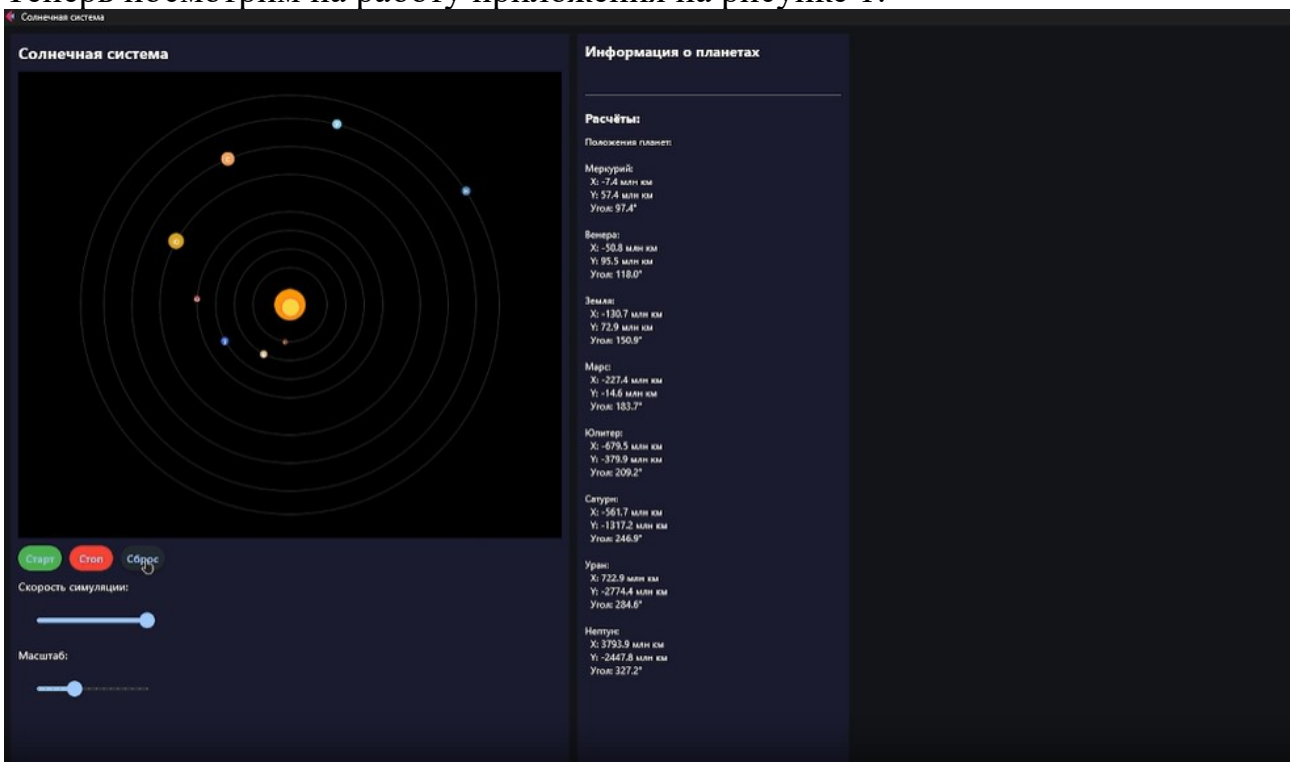

Рисунок 1 — Демонстрация работы