```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

//      Движение тела
// Задание 1.
// Объект движется равномерно. Определить путь, пройденный за время t.
// Исходные данные: скорость объекта.

// Задание 2.
// Определить координату и пройденный путь для объекта, движущегося
// равномерно, через время t после начала движения.
// Исходные данные: начальная координата, скорость.

// Задание 3.
// Объект движется равноускоренно. Определить пройденный путь, скорость и
// координату в момент времени t.
// Исходные данные: ускорение, начальная скорость.

// Задание 4.
// Выполнить задания 1-3 с реализацией движения по 2-м координатам.
// Скорости по координатам x и y могут быть различными.


public class CircleController : MonoBehaviour
{
    [SerializeField] private SpriteRenderer shipSprite;
    [SerializeField] private float maxSpeed = 10f;
    [SerializeField] private float maxAcc = 10f;


    [Header("Stats")]
    [SerializeField] Text statsPosition;
    [SerializeField] Text speed;
    [SerializeField] Text path;
    [SerializeField] Text accel;
    [SerializeField] Text statTime;


    [Header("UI-Speed")]
    [SerializeField] private Slider xSpeedSlider;
    [SerializeField] private Slider xAccSlider;
    [SerializeField] private Slider ySpeedSlider;
    [SerializeField] private Slider yAccSlider;


    [Header("UI-Buttons")]
    [SerializeField] private Button startStopBtn;
    private Text startStopBtn_text;


    Vector3 startPosition;
    private float xSpeed = 0f;
    private float xAcc = 0f;
    private float ySpeed = 0f;
    private float yAcc = 0f;
    private float time;
    private float S;



    private void Start()
    {
        xSpeedSlider.onValueChanged.AddListener(delegate { xSpeedListener(); });
        ySpeedSlider.onValueChanged.AddListener(delegate { ySpeedListener(); });

        xAccSlider.onValueChanged.AddListener(delegate { xAccListener(); });
        yAccSlider.onValueChanged.AddListener(delegate { yAccListener(); });

        shipSprite = gameObject.GetComponentInChildren<SpriteRenderer>();
        startPosition = gameObject.transform.position;

        startStopBtn_text = startStopBtn.GetComponentInChildren<Text>();
    }


    //Updating
    void FixedUpdate()
    {
        if (startStopBtn_text.text == "Stop")
```

```
        {
            if (Mathf.Abs(xSpeed + ySpeed) > 0)
                time += Time.fixedDeltaTime;

            rotate();
            Move();
            updateStats();
        }
    }


    private void Move()
    {
        float dt = Time.fixedDeltaTime;

        float Sx = xSpeed * dt + xAcc * dt * dt / 2;
        float Sy = ySpeed * dt + yAcc * dt * dt / 2;
        S += (Mathf.Sqrt(Sx * Sx + Sy * Sy));


        xSpeed += xAcc * dt;
        ySpeed += yAcc * dt;
        gameObject.transform.position += new Vector3(xSpeed * dt, ySpeed * dt, 0);
    }


    private void rotate()
    {
        Vector3 normSpeed = Vector3.Normalize(new Vector3(xSpeed, ySpeed, 0));
        float angle = Mathf.Atan2(normSpeed.y, normSpeed.x) * Mathf.Rad2Deg;

        gameObject.transform.rotation = Quaternion.Lerp(
            gameObject.transform.rotation,
            Quaternion.Euler(0, 0, angle),
            Time.fixedDeltaTime * 10
        );

    }


    private void updateStats()
    {
        statsPosition.text = $"Position: {round(gameObject.transform.position.x)}, {round(gameObject.transform.position.y)}";
        speed.text = $"Speed: {round(xSpeed)}, {round(ySpeed)}";
        statTime.text = $"Time: {round(time)}";
        accel.text = $"Acceleration: {round(xAcc)}, {round(yAcc)}";
        path.text = $"Path: {round(S)}";
    }



    //UI / Listeners

    void xSpeedListener()
    {
        xSpeed = xSpeedSlider.value * maxSpeed;
        updateStats();
    }

    void ySpeedListener()
    {
        ySpeed = ySpeedSlider.value * maxSpeed;
        updateStats();
    }

    void xAccListener()
    {
        xAcc = xAccSlider.value * maxAcc;
        updateStats();
    }

    void yAccListener()
    {
        yAcc = yAccSlider.value * maxAcc;
        updateStats();
    }

    public void onResetClick()
    {
        gameObject.transform.position = startPosition;
        gameObject.transform.rotation = Quaternion.Euler(0, 0, 0);
```

```
        xSpeedSlider.value = 0;
        xAccSlider.value = 0;
        ySpeedSlider.value = 0;
        yAccSlider.value = 0;
        xSpeed = 0;
        ySpeed = 0;
        time = 0;
        S = 0;
        updateStats();

    }

    public void onStartStopButtonClick()
    {
        if (startStopBtn_text.text == "Stop")
        {
            startStopBtn_text.text = "Start";
        }
        else
        {
            startStopBtn_text.text = "Stop";
        }
    }


    //Utils
    private float round(float value)
    {
        return (float)(Mathf.Round(value * 100.0f) * 0.01);
    }
}
```