using UnityEngine;

using System.Collections;

public class cont : MonoBehaviour {

  // PUBLIC

  public bool driveable = false;

  // Wheel Wrapping Objects

  public Transform frontLeftWheelWrapper;

  public Transform frontRightWheelWrapper;

  public Transform rearLeftWheelWrapper;

  public Transform rearRightWheelWrapper;

  // Wheel Meshes

  // Front

  public Transform frontLeftWheelMesh;

  public Transform frontRightWheelMesh;

  // Rear

  public Transform rearLeftWheelMesh;

  public Transform rearRightWheelMesh;

  // Wheel Colliders

  // Front

  public WheelCollider wheelFL;

  public WheelCollider wheelFR;

  // Rear

  public WheelCollider wheelRL;

  public WheelCollider wheelRR;

  public float maxTorque = 20f;

  public float brakeTorque = 100f;

  // max wheel turn angle;

  public float maxWheelTurnAngle = 30f; // degrees

  // car's center of mass

  public Vector3 centerOfMass = new Vector3(0f, 0f, 0f); // unchanged

  // GUI

  //...

  public float RO\_speed; // READ ONLY (Debug)

  public float RO\_EngineTorque; // READ ONLY (Debug)

  public float RO\_SteeringAngleFL; // READ ONLY (Debug)

  public float RO\_SteeringAngleFR; // READ ONLY (Debug)

  public float RO\_BrakeTorque; // READ ONLY (Debug)

  // PRIVATE

  // acceleration increment counter

  private float torquePower = 0f;

  // turn increment counter

  private float steerAngle = 0f;

  // original wheel positions

  // Front Left

  private float wheelMeshWrapperFLx;

  private float wheelMeshWrapperFLy;

  private float wheelMeshWrapperFLz;

  // Front Right

  private float wheelMeshWrapperFRx;

  private float wheelMeshWrapperFRy;

  private float wheelMeshWrapperFRz;

  // Rear Left

  private float wheelMeshWrapperRLx;

  private float wheelMeshWrapperRLy;

  private float wheelMeshWrapperRLz;

  // Rear Right

  private float wheelMeshWrapperRRx;

  private float wheelMeshWrapperRRy;

  private float wheelMeshWrapperRRz;

  void Start () {

    GetComponent<Rigidbody>().centerOfMass = centerOfMass;

  }

  // Visual updates

  void Update () {

    if (! driveable) {

      return;

    }

    // SETUP WHEEL MESHES

    // Turn the mesh wheels

    frontLeftWheelWrapper.localEulerAngles = new Vector3(0, steerAngle, 0);

    frontRightWheelWrapper.localEulerAngles = new Vector3(0, steerAngle, 0);

    // Wheel rotation

    frontLeftWheelMesh.Rotate(wheelFL.rpm / 60 \* 360 \* Time.deltaTime, 0,0);

    frontRightWheelMesh.Rotate(wheelFR.rpm / 60 \* 360 \* Time.deltaTime, 0,0);

    rearLeftWheelMesh.Rotate( wheelRL.rpm / 60 \* 360 \* Time.deltaTime,0 ,0);

    rearRightWheelMesh.Rotate(wheelRR.rpm / 60 \* 360 \* Time.deltaTime,0, 0);

    // Audio

    GetComponent<AudioSource>().pitch = (torquePower / maxTorque) + 0.5f;

  }

  // Physics updates

  void FixedUpdate () {

    if (! driveable) {

      return;

    }

    // CONTROLS - FORWARD & RearWARD

    if ( Input.GetKey(KeyCode.Space) ) {

      // BRAKE

      torquePower = 0f;

      wheelRL.brakeTorque = brakeTorque;

      wheelRR.brakeTorque = brakeTorque;

    } else {

      // SPEED

      torquePower = maxTorque \* Mathf.Clamp( Input.GetAxis("Vertical"), -1, 1 );

      wheelRL.brakeTorque = 0f;

      wheelRR.brakeTorque = 0f;

    }

    // Apply torque

    wheelRR.motorTorque = torquePower;

    wheelRL.motorTorque = torquePower;

    // Debug.Log(Input.GetAxis("Vertical"));

    Debug.Log("torquePower: " + torquePower);

    Debug.Log("brakeTorque RL: " + wheelRL.brakeTorque);

    Debug.Log("brakeTorque RR: " + wheelRR.brakeTorque);

    Debug.Log("steerAngle: " + steerAngle);

    // CONTROLS - LEFT & RIGHT

    // apply steering to front wheels

    steerAngle = maxWheelTurnAngle \* Input.GetAxis("Horizontal");

    wheelFL.steerAngle = steerAngle;

    wheelFR.steerAngle = steerAngle;

    // Debug info

    RO\_BrakeTorque = wheelRL.brakeTorque;

    RO\_SteeringAngleFL = wheelFL.steerAngle;

    RO\_SteeringAngleFR = wheelFR.steerAngle;

    RO\_EngineTorque = torquePower;

    // SPEED

    // debug info

    RO\_speed = GetComponent<Rigidbody>().velocity.magnitude;

  }

}