using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CarController : MonoBehaviour

{

private const string HORIZONTAL = "Horizontal";

private const string VERTICAL = "Vertical";

private float horizontalInput;

private float verticalInput;

private float currentSteerAngle;

private float cureenrBreakForce;

private bool isBreaking;

private int gear = 0;

private bool isGear = false;

[SerializeField] private float motorForce = 0; //shte se promenq ot gears

[SerializeField] private float breakForce;

[SerializeField] private float maxSteerAngle;

[SerializeField] private WheelCollider frontLeftWheelColider;

[SerializeField] private WheelCollider frontRightWheelColider;

[SerializeField] private WheelCollider backLeftWheelColider;

[SerializeField] private WheelCollider backRightWheelColider;

[SerializeField] private Transform frontLeftWheelTransform;

[SerializeField] private Transform frontRightWheelTransform;

[SerializeField] private Transform backLeftWheelTransform;

[SerializeField] private Transform backRightWheelTransform;

private void FixedUpdate()

{

GetInput();

HandleMotor();

HandleSteering();

UpdateWheels();

UpdateGear();

/\*

if (frontLeftWheelColider.motorTorque == 0 && frontRightWheelColider.motorTorque==0)

{

IsStarting();

}\*/

}

private void IsStarting()

{

throw new NotImplementedException();

}

private void UpdateGear()

{

isGear = Input.GetKey(KeyCode.LeftShift);

if (isGear && Input.GetKey(KeyCode.Alpha1))

{

gear = 1;

motorForce = 10;

Debug.Log("gear 1");

}

}

private void HandleMotor()

{

frontLeftWheelColider.motorTorque = verticalInput \* motorForce;

frontRightWheelColider.motorTorque = verticalInput \* motorForce;

cureenrBreakForce = isBreaking ? breakForce : 0f;

if(isBreaking)

{

ApplyBreaking();

}

}

private void ApplyBreaking()

{

frontLeftWheelColider.brakeTorque = cureenrBreakForce;

frontRightWheelColider.brakeTorque = cureenrBreakForce;

backLeftWheelColider.brakeTorque = cureenrBreakForce;

backRightWheelColider.brakeTorque = cureenrBreakForce;

}

private void GetInput()

{

horizontalInput = Input.GetAxis(HORIZONTAL);

verticalInput = Input.GetAxis(VERTICAL);

isBreaking = Input.GetKey(KeyCode.Space);

}

private void HandleSteering()

{

currentSteerAngle = maxSteerAngle\*horizontalInput;

frontLeftWheelColider.steerAngle = currentSteerAngle;

frontRightWheelColider.steerAngle = currentSteerAngle;

}

private void UpdateWheels()

{

UpdateSingleWheel(frontLeftWheelColider, frontLeftWheelTransform);

UpdateSingleWheel(frontRightWheelColider, frontRightWheelTransform);

UpdateSingleWheel(backLeftWheelColider, backLeftWheelTransform);

UpdateSingleWheel(backRightWheelColider, backRightWheelTransform);

}

private void UpdateSingleWheel(WheelCollider wheelColider, Transform wheelTransform)

{

Vector3 pos;

Quaternion rot;

wheelColider.GetWorldPose(out pos, out rot);

wheelTransform.rotation = rot;

wheelTransform.position = pos;

}

}