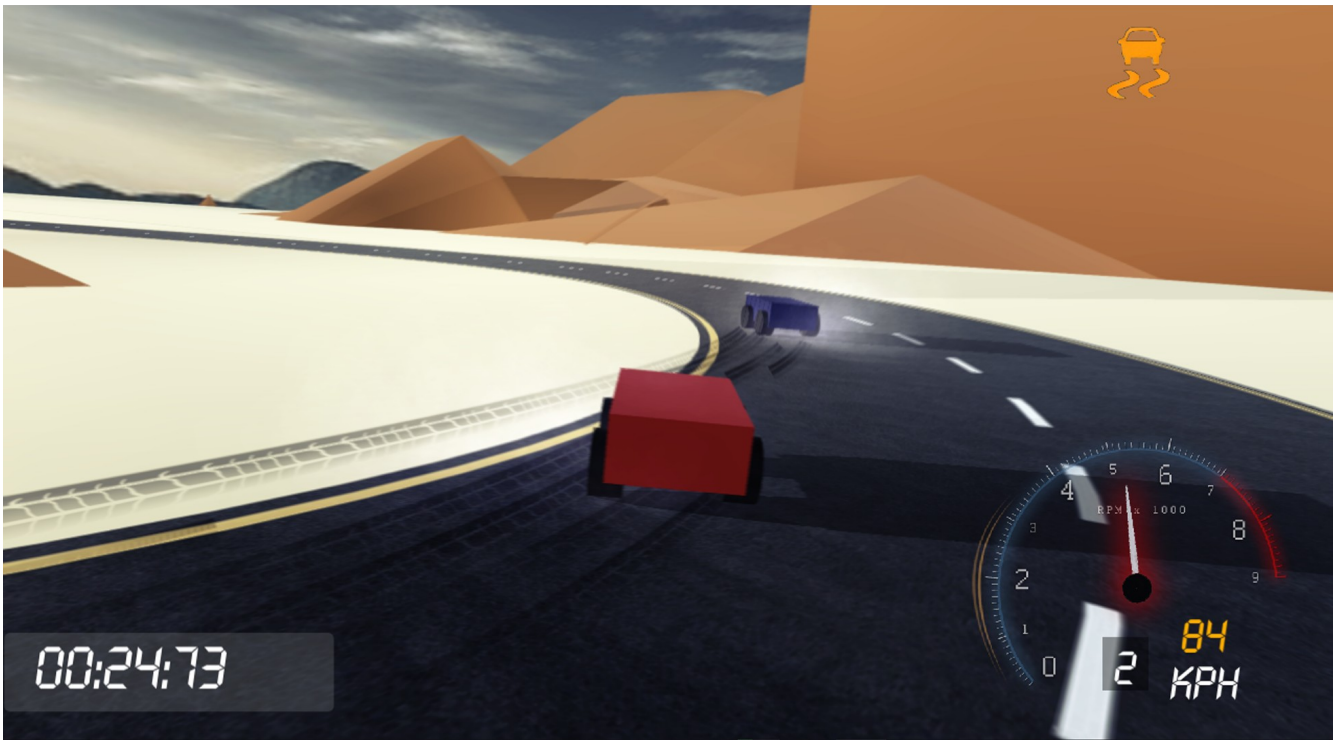
Car Systems Asset



For developing a Car's game, this Asset will makes driving the cars more fun and realistic. It is not about the models or the effects, It's about the systems in the car to make it controllable.
This Asset uses WheelColliders for the car and all the Scripts are in C#.

How it works:

Maybe you should start from the "userInput" script, where it reads the user input and send it to the "CarControl".
The "CarControl" Script adjusts the Car's status (going forward, backward, braking….etc).
If the Car is going forward, it reads the engine's torque and send it to the wheels, or the TCS.
The "PlayerCar" & "SpeedoMeter" Scripts listens to the Car's status and update the UI. "BrakeLights", "RevLights" and "FrontLights" listen to car events and turns on or off.
The "CarControl" asks the engine and transmission to update them self depending on the car's speed.

There is a Prefab in the demo folder for a Car GameObject that shows the scripts that should be added to the car to work.

Key Features:

- CarEngine Script: defines Car’s engine power using HP or RPM curve.
- Transmission Scripts: uses Gear’s ratio curve for speed shifting.
- Brake System with ABS.
- HandBrake
- Lights Scripts: ◦ Headlights ◦ Brake lights ◦ Reverse lights
- TCS
- EPS
- Front & Rear Spoiler
- Nitro
- DriftSystem
- SpeedoMeter

Note: Maybe some knowledge in C# is required to be able to merge these scripts with your project. There is a demo scene that may help.

CarControl.cs

This is the main script for controlling the car. “userInput” script reads the user Input and calls CarControl.Move method:

```
public void Move(float steering, float accel, float footbrake, float handbrake)
```

At the beginning of this function, it asks the CarEngine, Transmission & Wheel Scripts to update there values, and updates the car current speed. Those values like wheel’s rpm, wheels’ slip & current engine torque will be saved in variables at their classes, so no need to call functions or to calculate their values every time we ask for them.
After that the “Move” function applies the steering for the front wheels. Then it calls “`ApplyDrive`” function.

```
private void ApplyDrive()
```

The “ApplyDrive” decides if the car should go forward, go reverse, apply brake, or do nothing, based on the user’s input and the transmission’s currentGear.

```
public void goForward()
```

This function changes the car status to `CarStause.Forward` and calls `sendTorque(car.engine.getWheelsTorque())`, which will decide to send the torque to the TCS if it is active, or just send the torque direct to the motor wheels.

Car status:

```
public enum CarStause{Forward,    Stopped,    Backward,    Braking,    Neutral}
```

When the CarControl Changes the car’ status through `setCarStatus(CarStause c_status)` it triggers events, for example when change to `CarStause.Braking` it triggers “brakingListner” and the braklights will turn on. Or when changing to Backward the Transmission changes the currentGear to -1.

```
public void nitro(bool active)
```

The “userInput” scripts always calls `nitro(bool active)` function in the CarControl script to update the “isActive” variable in the Nitro script. IsActive = true when player is “left Alt” key is down.

```
public void toggleDriftSystem()
```

This function to toggle on & off the DriftSystem. I didnt create a user input for this function but it will be the same as the key for turning on and of the TCS system which is `KeyCode.A`.

Car.cs

The car script contains most of the cas’s specifications. And at its start, it tries to detect and save wheels groups in arrays for faster access.

```
public Wheel[] frontWheels = new Wheel[2];
public Wheel[] backWheels = new Wheel[2];
public Wheel[] leftWheels = new Wheel[2];
public Wheel[] rightWheels = new Wheel[2];
```

Steering.cs

```
public float maxSpeedSteeringAngle = 3; // steering Angle to apply when the car at its
                                          maximum speed
public float lowSpeedFSteerAngle = 15; //steering Angle to apply when the car at its
                                          minimum speed
```

This script is for the car's steering (and the car's counter-steering technique if this technique is enabled in the car's Accessories)

CarEngine.cs

the cars engine updates the current engine's RPM based on the vehicle speed.

```
public float getEngineTorque()
```

This function calculates the Engine torque based on the HP if the HP>0 or returns the torque value from the car's RPM_Curve.

```
public float getWheelsTorque()
```

This functions gives the torque for the wheels considering the current gear ratio and the differential ratio. Note that this value is the sum torque for all wheels.

Transmission.cs

The Transmission script adds listener for the car status. To change between R, N, and 1.
It reads the gears ration from the car's `GearRation_Curve`.

```
public void setCurrentGear(int i)
```

When changing the current gear, this function makes sure not to change the current gear too soon after the last time it was changed, by waiting for the "blockTimer" to be 0.
and it simulates the shifting speed.

```
public void updateCurrentGear()
```

updateCurrentGear calculates the current gear based on the car's speed and the top speed for every gear.

Wheel.cs

this script contains the wheelcollider and the wheel's mesh game objects. It also have `engineTorquePercent` variable, that determines how much torque it takes from the sum wheels torque that the engine produce. This is helpful if you have for example a 4WD car and want to have more torque for the rear wheels than the front wheels.

Brake.cs

This script applys the brake on the correspondent wheel. And it has an ABS system to make sure that the wheel is not locked while braking.

Accessories.cs

It has the car's accessories attributes.

```
public bool hasABS = true;
public bool hasCountersteering = true;
public TCSSystem tcsSystem;
public Nitro nitro;
public DriftSystem driftSystem;
public EPSSystem epsSystem;
public Wings wings;
```

Wings.cs

It simulates the cars front & rear Spoilers, by adding a downforce on the car.

TCSSystem.cs

when wheelsTractionSystem function gets the wheels torque from the carControl, it checks if there is a forward wheel spin and adjust the torque before forwarding it to the wheel.

EPSSystem.cs

It adds more stability and control over the car while turning by reducing the torque on the inner wheels, and applying more torque for the outer wheels.

Nitro.cs

It simulates car's nitro, by adding forward force to the car.

DriftSystem.cs

It makes drifting easier by adjusting wheel's stiffness. But I found drifting using handbrake is acceptable, and no need for the driftSystem.
For drifting with handbrake to work, the TCS should be off, That can be done while playing by pressing the "A" key to turn the TCS off, and "left Ctrl" for handbraking.

CarLight.cs

FrontLight.cs, BrakeLight.cs, and RevLights.cs are all sub-classes from CarLight and everyone adds its own listener to car events to turn on or off the light.
The frontlights can be turned on by calling frontLightsOn() in the carControl, or frontLightsOff() to turn it off.
The frontLights can be turned on or off autocratically in the game for example if the car enters a tunel, by adding a trigger Collider gameobject and add `LightCollider.cs` component to that game object so when the cars enters the collider it triggers the events in the carControl to turn the frontlights on.

And when the car exits the collider it triggers the event to turns the lights off.

PlayerCar.cs

This script is only necessary for the player's car. It updates the UI information like showing the TCS & ABS signals. For AI cars this script should not be included in the car's game object combonents.

I hope this documentation will help you to understand how these scripts work, and makes it easier for you to modify, expand or merge it with your project.

Please do not hesitate to contact me if you have any further questions at my email:
ashraf82de@yahoo.de

WebGL Demo:
http://unity.mits-ye.com/car/car_systems/

Windows Demo:
http://unity.mits-ye.com/car/car_systems/windows.zip

Youtube Demo Video:
https://youtu.be/Olntepd5F1A