Lists

Objective
To make a car using Unity's basic

Make a car

Make a car

Setup
- Install the provided package
- Create a basic terrain to drive around
- Place the Car mesh into the scene
- Put the **CarD** material on the car and each of the wheels
- Create a script called **Vehicle** and place it on the car

Lesson

Make the car

- Put a RigidBody onto the car
- Change the mass of the Rigidbody to 2000
- Add a Mesh collider to the car (with **Convex)** selected
- Select all the wheels and add wheel colliders onto them
    - Change their radius to 0.4
    - Change Centre Y to 0.15

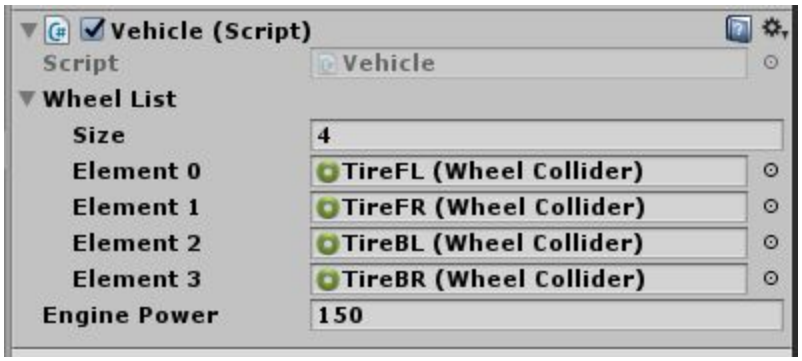
Press play and the call should fall and land on its wheels

Note: *if done correctly, once you press play the car will hover slightly above the ground. This is because the wheel colliders and the mesh as treated separately, the distance is the center of the game object + the 'Suspension Distance'*
To fix this is outside of the scope of this tutorial

Open up the Vehicle Script.

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

public class Vehicle : MonoBehaviour {

    public List<WheelCollider> wheelList;

    public float enginePower = 150.0f;

    void Update()
    {
        for (int i = 0; i < wheelList.Count; i++)
        {
            wheelList[i].motorTorque = enginePower * Time.deltaTime * 250.0f * Input.GetAxis("Vertical");
        }
    }

}
```

Once on the car you will need to add the four wheels to the wheelList ensuring the FL and FR are the first two wheels on the list (so we can steer them later).



Run the game and you should be able to move the card forwards and backwards.

Let's make the car steer, Update the **Vehicle** script

```
public class Vehicle : MonoBehaviour {

    public List<WheelCollider> wheelList;

    public float enginePower = 150.0f;

    public float steer = 0.0f;
    public float maxSteer = 25.0f;

    void Update()
    {
        for (int i = 0; i < wheelList.Count; i++)
        {
            wheelList[i].motorTorque = enginePower * Time.deltaTime * 250.0f * Input.GetAxis("Vertical");
        }

        wheelList[0].steerAngle = Input.GetAxis("Horizontal") * maxSteer;
        wheelList[1].steerAngle = Input.GetAxis("Horizontal") * maxSteer;
    }

}
```

If you run the game you should be able to steer the car, but you will notice the car tips when you steer too sharply.
This is because the center of mass is too high.
Let's change that

```
public class Vehicle : MonoBehaviour {

    public List<WheelCollider> wheelList;

    public float enginePower = 150.0f;

    public float steer = 0.0f;
    public float maxSteer = 25.0f;

    public Vector3 centerOfMass = new Vector3(0, -0.5f, 0.3f);

    void Start()
    {
        GetComponent<Rigidbody>().centerOfMass = centerOfMass;
    }


    void Update()
    {
        for (int i = 0; i < wheelList.Count; i++)
```

Of course you can change these values to whatever you want them to be.

Now to get the camera to follow the car make a scripts called **FollowCamera** and add the following

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

public class FollowCamera : MonoBehaviour {

    public Transform target;
    public float distance = 8.0f;
    public float height = 3.0f;
    public float damping = 5.0f;
    public bool smoothRotation = true;
    public bool followBehind = true;
    public float rotationDamping = 10.0f;

    void FixedUpdate()
    {
        Vector3 wantedPosition;
        if (followBehind)
            wantedPosition = target.TransformPoint(0, height, -distance);
        else
            wantedPosition = target.TransformPoint(0, height, distance);

        transform.position = Vector3.Lerp(transform.position, wantedPosition, Time.deltaTime * damping);

        if (smoothRotation)
        {
            Quaternion wantedRotation = Quaternion.LookRotation(target.position - transform.position, target.up);
            transform.rotation = Quaternion.Slerp(transform.rotation, wantedRotation, Time.deltaTime * rotationDamping);
        }
        else transform.LookAt(target, target.up);
    }

}
```

Attach this script to the camera and set up the **Target** in the inspector

Once done, make your car a prefab for later use.

Make a better car

Now that we have made a vehicle you might have noticed that it doesn't feel right and there is a lot of settings to tweak before we get it to feel right.
But we are all about doing things the easy way, so find a shortcut.

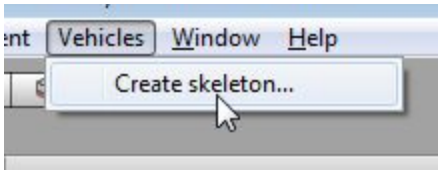
Setup

Continued from previous Lesson

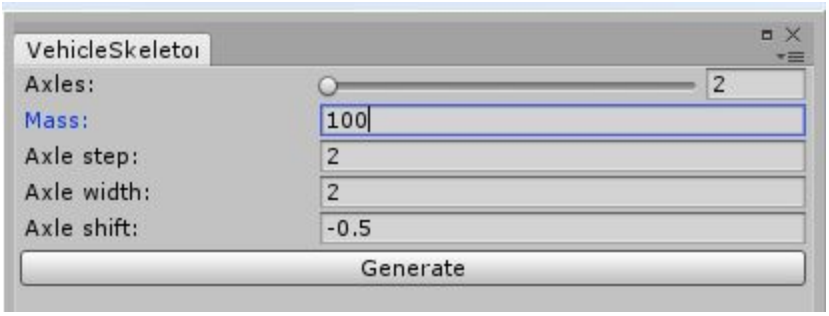
Open up the Unity asset store and find and install **Vehicle Tools** this is a package made by Unity that allows you to easily build vehicles.

Lesson

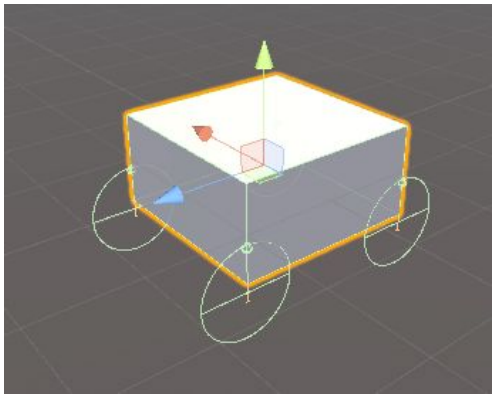
You will now have a new menu item called **Vehicles**, select this and press **Create skeleton...**



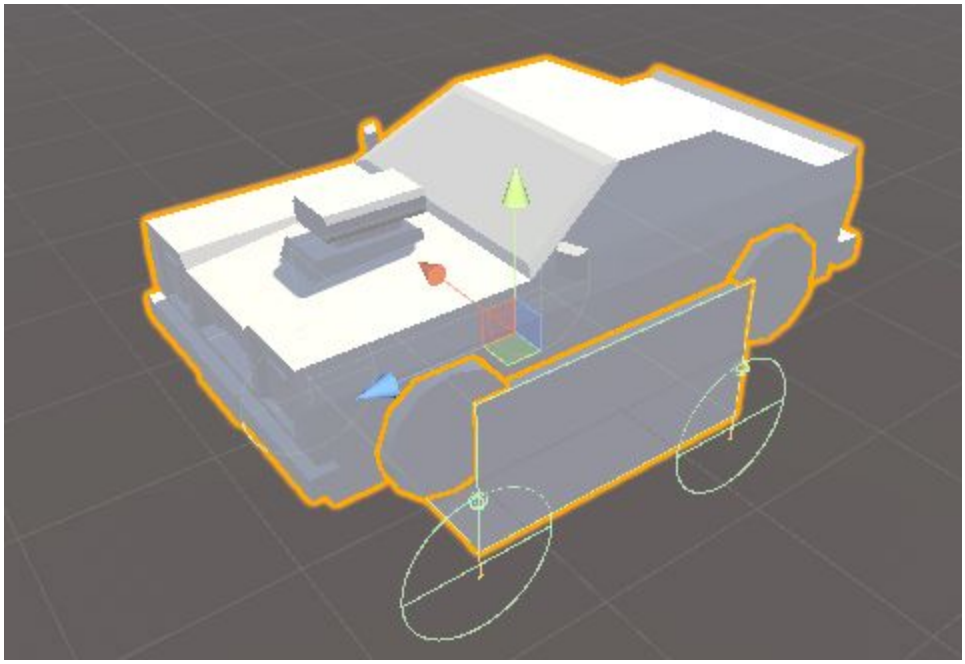
Use the default settings

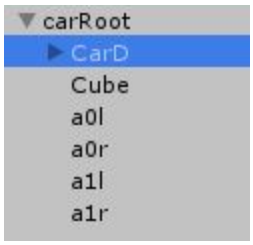


We can now make a car using all the pieces we already have

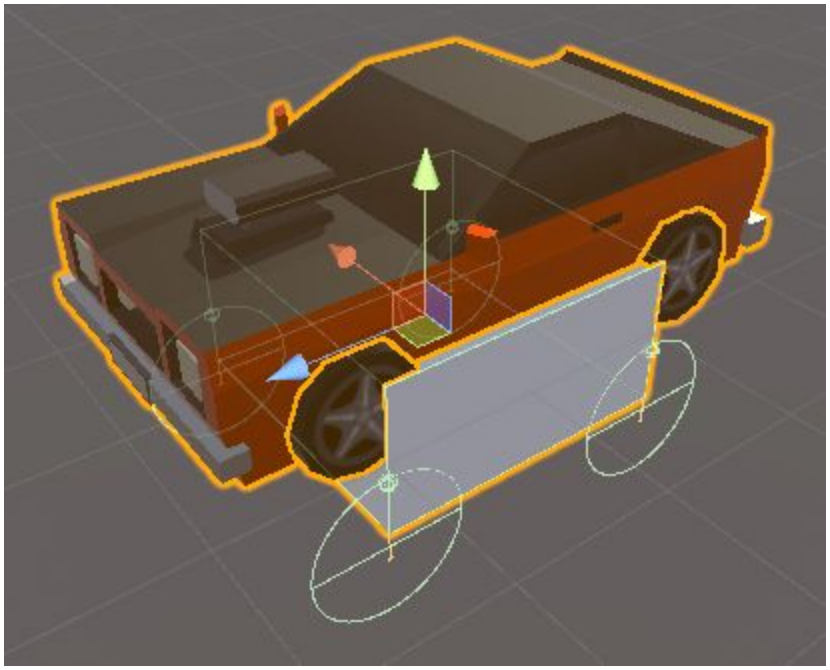


Get the **CarD** model and put it under the **carRoot** we have just created

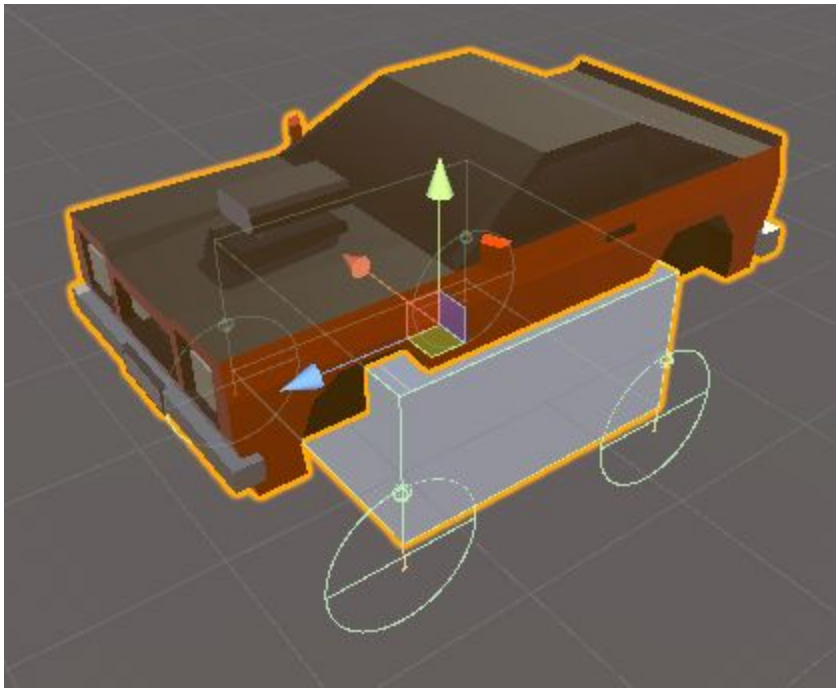




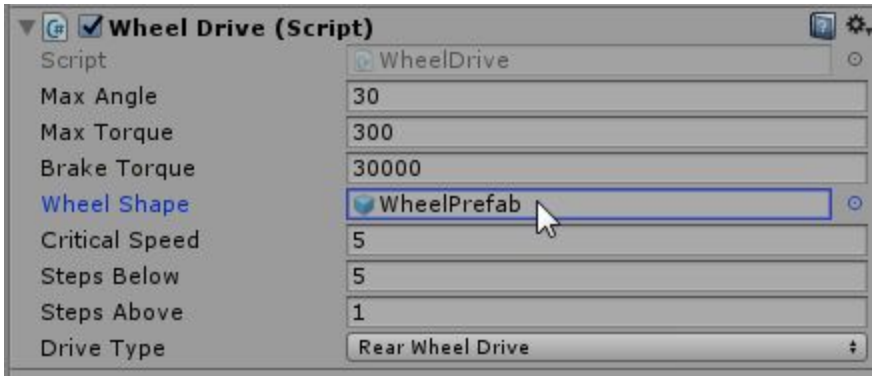
Apply the material to the car and the wheels



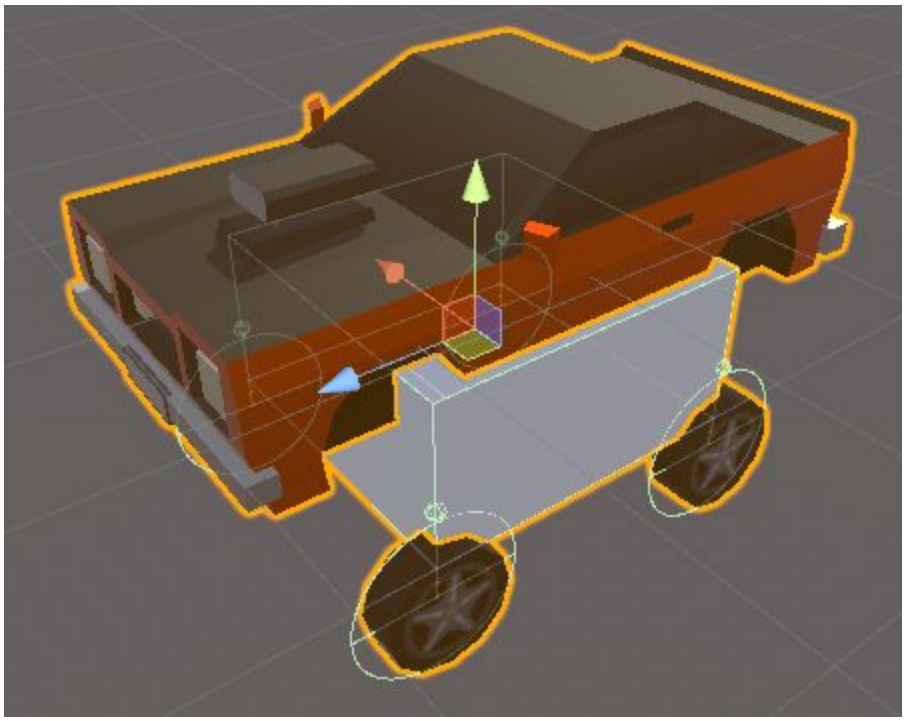
Select one of the wheels and save it as a **WheelPrefab** then Delete all the wheels from the scene



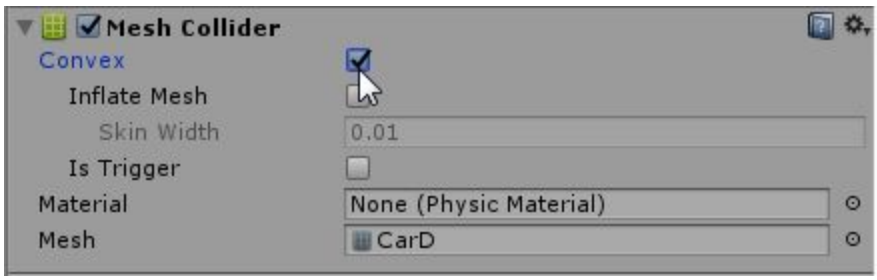
In the inspector place the newly created **WheelPrefab** in the **Wheel Shape** field



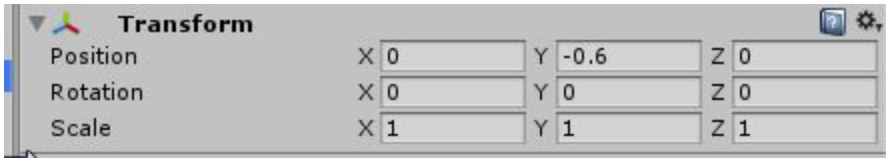
If you press play you will see that the code automatically spawns in wheels at the correct spots and makes them rotate.



Let’s make everything line up.
Delete the **Cube**
Add a MeshCollider to the **Car Mesh** and select **Covex**

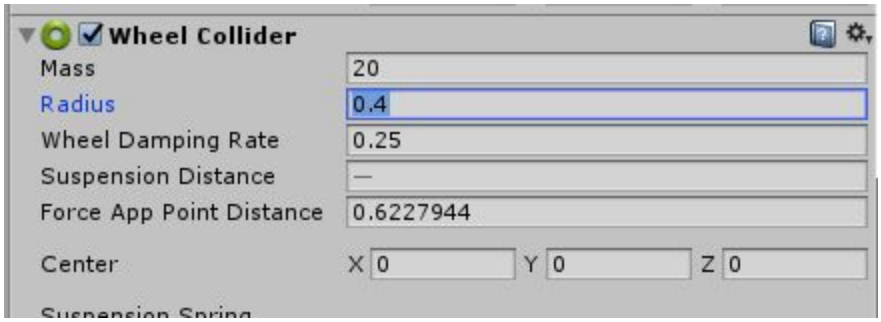


Change the car body's Transform to the following



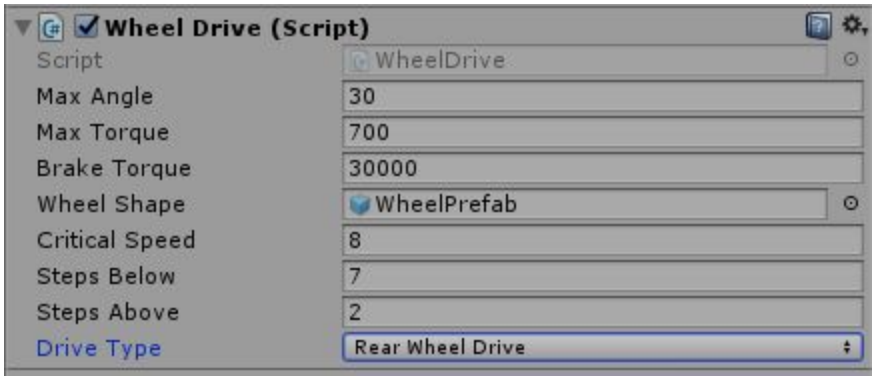
When run the body should be at the right height now.

Change the **Wheel Collider**'s **Radius** on each wheel to 0.4



And move each wheel collider to line up with the body

Change the settings of car in the inspector



And there we have it, It’s not perfect, but it’s pretty good.

Make a Racetrack

Making a track for the cars to drive around.

Setup

Continued from previous Lesson

- Make a script called **CheckpointManager**
- Add the **CheckpointManager** to the car
- Add a **BoxCollider** to the vehicle
- Tag the vehicle as **Player**
- Make a cube to act as a checkpoint trigger
    - Make the collider a Trigger
    - Make a script called **Checkpoint** and put it on the Cube
    - Delete the cubes **Mesh Renderer**
    - Use the **TriggerDisplay** script we used previously to make is visible only in the editor
    - Rename the cube to **Checkpoint**
    - Make the cube a prefab

Lesson

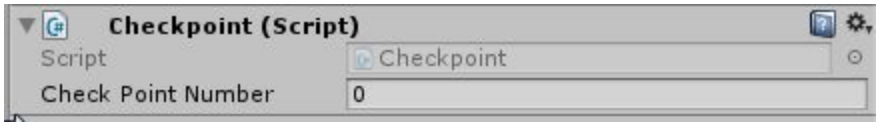
Then update the **CheckpointManager** Script

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

public class CheckpointManager : MonoBehaviour {

    public int currentCheckpoint = 0;
    public int numberOfCheckpoints = 0;

    void Start()
    {
        //scan the level for checkpoints and count how many you find
        numberOfCheckpoints = GameObject.FindObjectsOfType<Checkpoint>().Length;
    }

    public void HitCheckPoint(int checkpointNumber)
    {
        if (checkpointNumber == currentCheckpoint + 1)
        {
            currentCheckpoint = checkpointNumber;
            Debug.Log("Just hit checkpoint number " + checkpointNumber);
            if (checkpointNumber >= numberOfCheckpoints)
            {
                currentCheckpoint = 0;
            }

        }
        else
        {
            Debug.Log("Wrong Checkpoint for " + transform.name);

        }
    }

}
```

Open the **CheckPoint** Script

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

public class Checkpoint : MonoBehaviour {

    public int checkPointNumber;

    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Player")
        {
            other.GetComponent<CheckpointManager>().HitCheckPoint(checkPointNumber);
        }
    }

}
```

Now you can place checkpoints throughout the scene.
For each checkpoint, set the Check Point Number. Starting at 1 for the first, then 2,3,4… through to the last one.



Now that we have checkpoint system, we need a track to go with it.

In the Unity Asset store download **EasyRoads 3D Free**

And follow the tutorials to build a map with this awesome tool

http://www.unityterraintools.com/tutorials.php