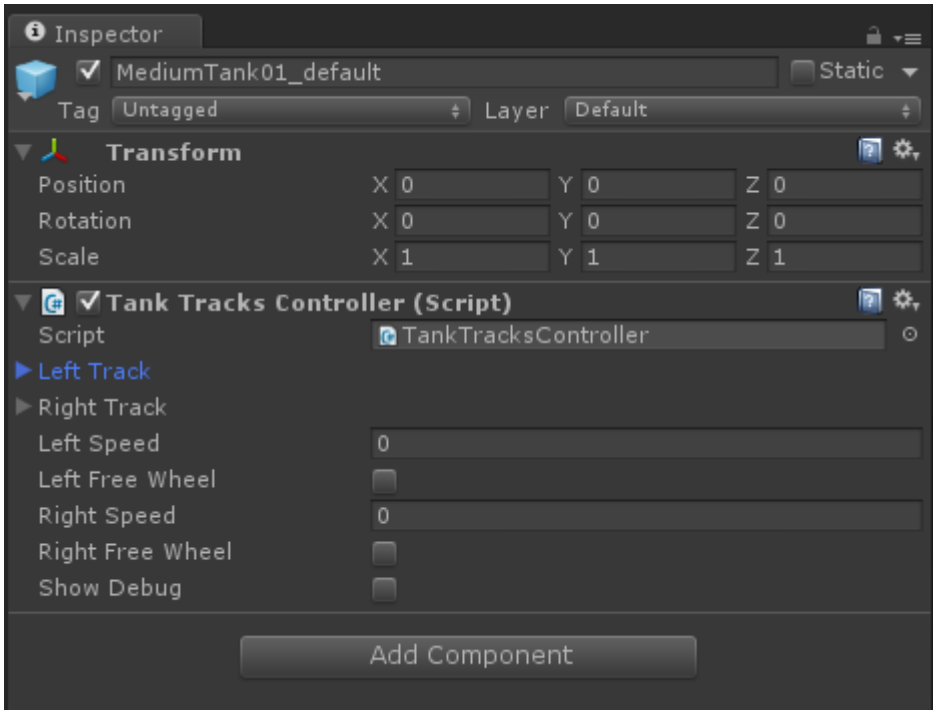
Controller scripts

The tank prefabs in this package make use of a pair of scripts to control the movement of the tracks and the rotation of the wheels.

- TankTracksController.cs – The main control script is the only one that has to be explicitly included into a tank prefab.
- TrackController.cs – Controls the track movement and wheel rotation of one track. The C# object defined in this script is used twice in the main script to control each of the track assemblies individually. This script does not need to be explicitly included into the tank prefab as it is automatically included by the main script.

Using the pre-configured tank prefabs

The pre-configured prefabs come with the script attached and configured to drive the movement of the tracks and wheels.



The speed of the tracks can be driven directly by setting the “Left Speed” and “Right Speed” parameters. These parameters are in terms of local coordinate velocities for the track objects on the prefab, so if any scaling is used it needs to be taken into account.

From a programming standpoint reference to the TankTracksController is needed and the “leftSpeed” and “rightSpeed” public variables can be accessed directly. The following C# code snippet shows the sort of values you would set for a tank turning on the spot.

…

```
tracksController.leftSpeed = 2.5f;

tracksCotroller.rightSpeed = -2.5;
```

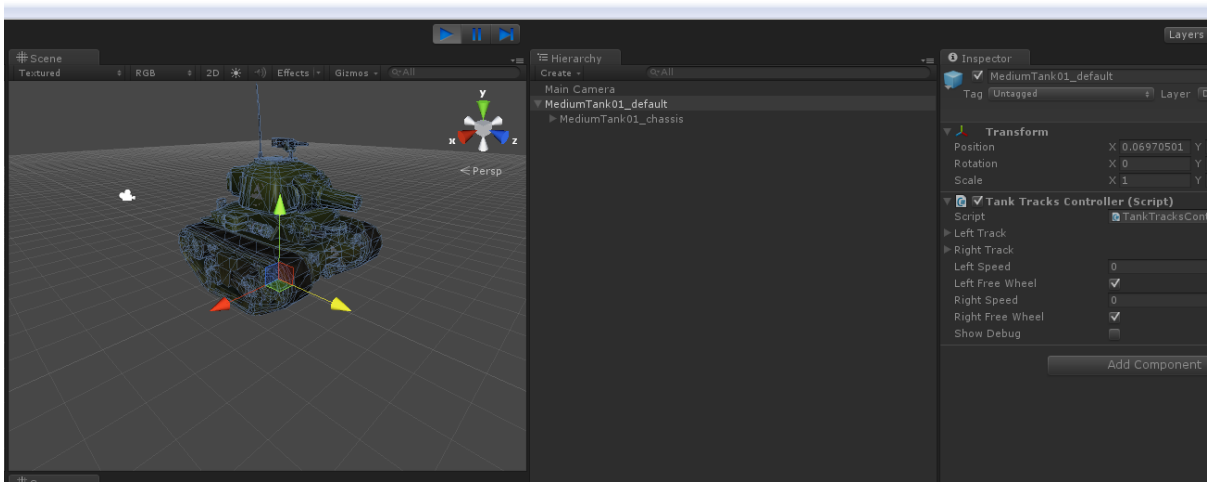
…

Note that the track movement will not cause the tank to actually move in the game world. Any such movement will have to be driven by further code/physics.

The second method of driving the tracks is to use the “Left Free Wheel” and “Right Free Wheel” check boxes. When checked any movement/rotation of the tank object in the game world gets translated into track movement. As with the setting the track speeds directly, actually moving the tank object needs to be driven by further code/physics.

In this movement mode the tank tracks should look like they are gripping on the game world and driving the movement.

This can be seen in action by creating a scene and putting one of the tank prefabs running the scene in the Unity editor and manipulating the tank’s position directly



- Any translation along the forward/backward axis (Z) of the tank should cause the tracks to animate correctly with the tank’s movements.
- Any rotation around the up/down axis (Y) of the tank should cause the tracks to animate correctly to follow the tank turning.

Using the script with a new tank.

The TankTracksController.cs script can be attached to a new tank prefab provided it meets or nearly meets the following criteria.

- The tank when created is orientated with forwards/backwards along the Z axis.
- The tracks and wheels to be controlled are separate game objects inside the hierarchy for the Unity prefab.
- Wheels on the same side need to be built so that they share the same orientation of rotation axis and be built to rotate around the local X axis.
- The more uniform the UV setup is around the track the better the coordination between the track and the wheels will be.
- The script treats one wheel as if it was a toothed drive wheel. The drive wheel and track on each side should be created with this in mind.
- It is normally best to have a consistent naming convention for the wheel and track objects in the prefab as this will aid in the configuration.

For the free wheel flags to work properly the track objects need to be aligned so that movement in the local Z direction of the track corresponds to the tank moving back/forwards.

Wheel and track movement is achieved by the script explicitly manipulating the objects for the tracks and wheels. The meshes and resulting Unity game objects need to be individually accessible.

The rotation orientation needs to be the same for each wheel on one side, as only one rotation direction is applied by the script for each track assembly. Rotation is applied on the local X axis for each wheel. If a wheel has been rotated 180 degrees around the y axis it effectively reverses the direction of rotation from an observer's perspective.

The movement of the track is achieved by adjusting the texture offset of the track to simulate movement. The more uniform the UV coordinates are around the length of the track mesh the less noticeable any distortion artefacts will be as the track moves in concert with the wheels.

The script was created with cogged drive wheels having teeth engaging with the track in mind and as a result should keep the teeth perfectly aligned with indentations on the track, provided it is created and configured correctly.

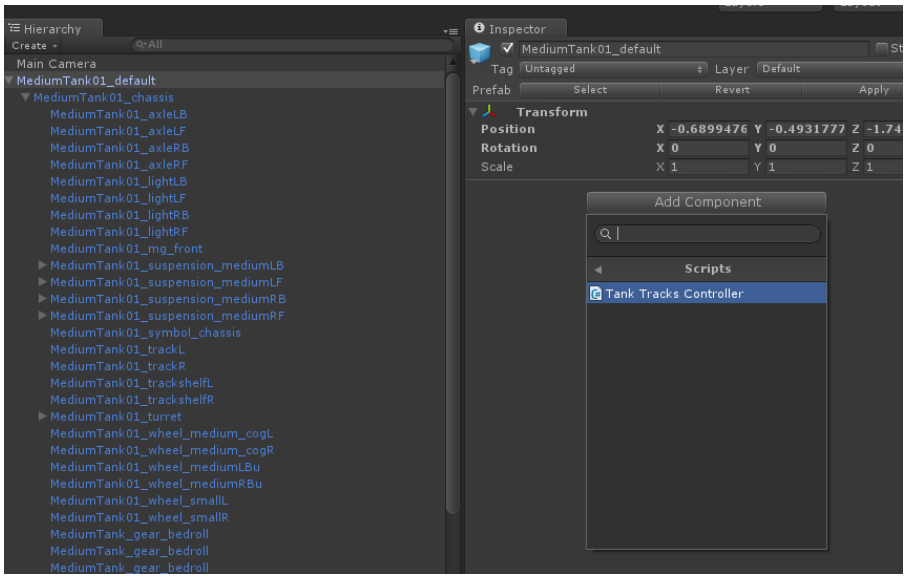
Effectively the track is treated as if it had a number of equal sized sections and that an integer number of those sections go round the drive wheel. If the wheel and track is constructed with this in mind it should keep the wheel and track movement in synchronisation.

A naming scheme for the tracks and wheels in the tank prefab will help considerably with the configuration. Once attached the script requires the various object components to be dragged into the script parameters. Being able to tell which object is which, will make things considerably less error prone.

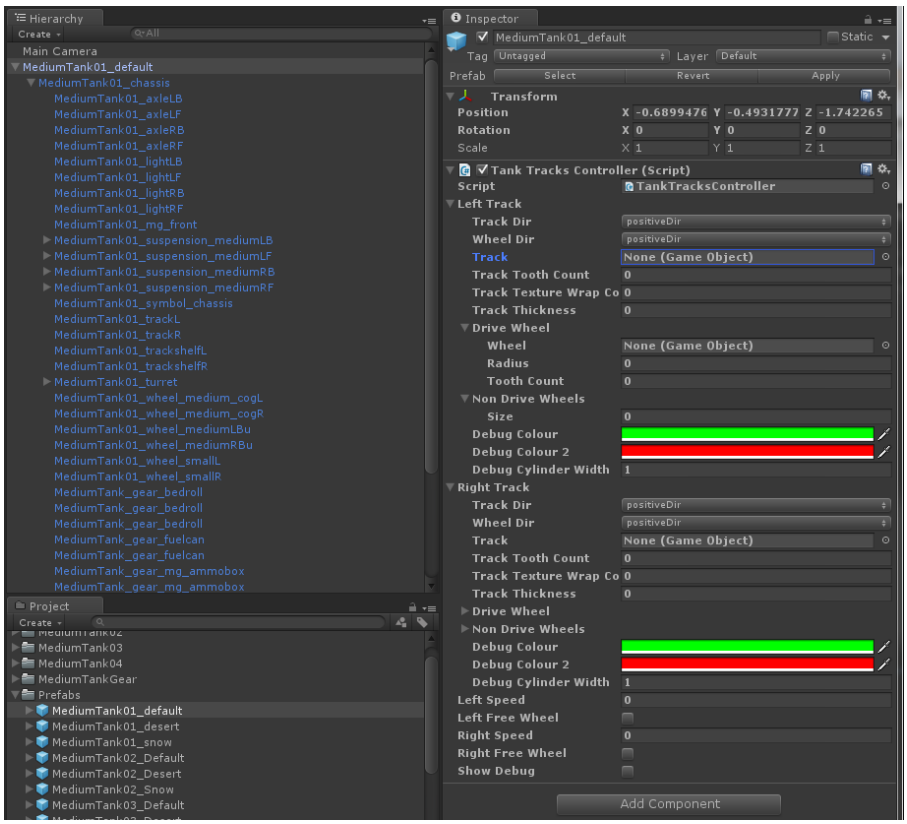
Configuring the script

With a suitable tank prefab model in Unity the following steps will configure the script to control it.

Firstly choose an appropriate object in the prefab hierarchy to add the TankTracksController script to. Use “Add Component” or drag the script to the object inspector for that object.

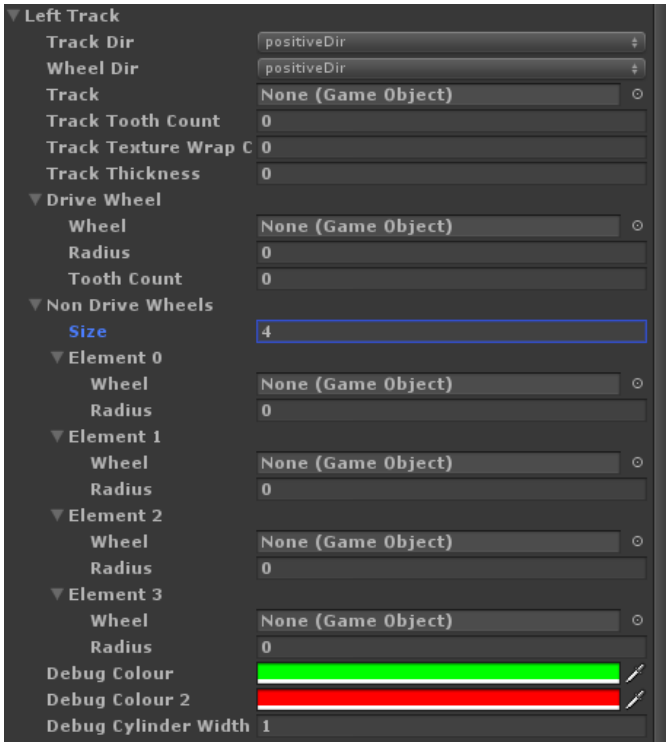


It is then possible to configure the parameters for the tracks and wheels.



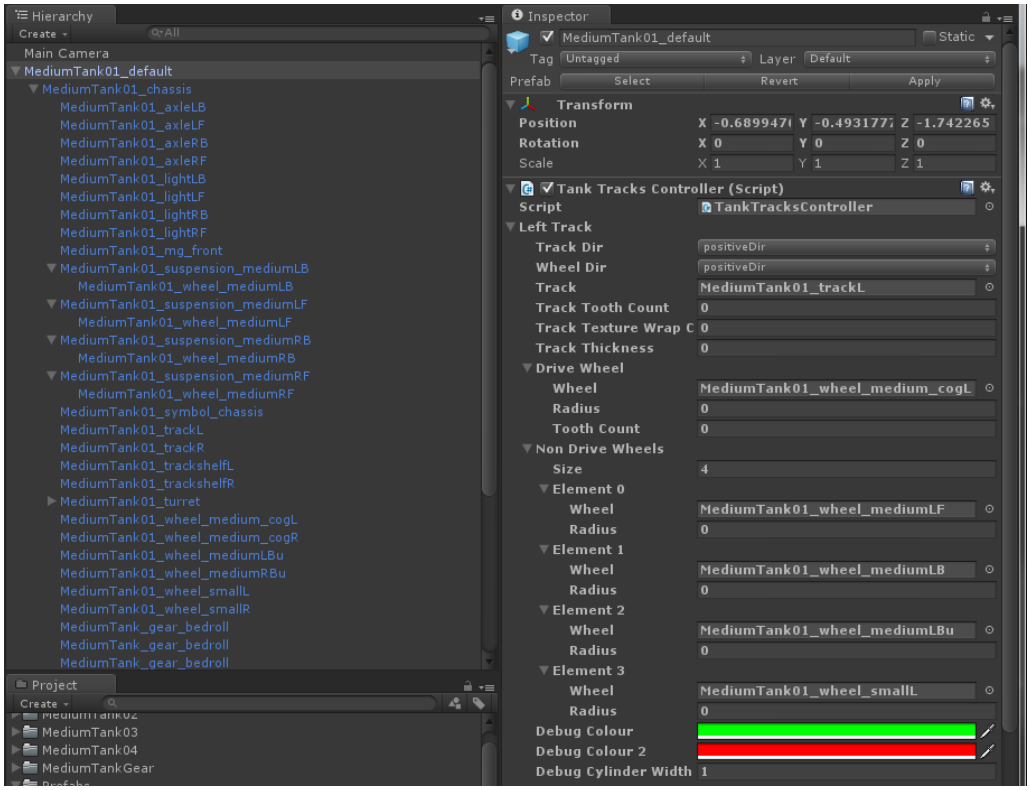
The example tank below has 5 wheels per side.

One wheel will be designated the drive wheel (the cogged one in this example) leaving 4 more non-drive wheels. To set up the configuration slots for the non-drive wheels, the “Non Drive Wheels” size parameter needs to be set to the number of non-drive wheels. In this case it will be set to 4.



This will need to be done for both the left and right tracks. The next step is to drag the track and wheel objects to their corresponding configuration parameters for their sides. (It is for this step that it is simplest if the various objects are named so that it is easy to identify them.)

The following shows the objects configured for the left track.



With all the tracks and wheels in the configuration, they need their configuration parameters set up.

It is recommended that the “Show Debug” toggle is set on as it will show debug cylinders for the wheels which will aid with visualising the relationships between the wheels and the tracks. In addition setting the free wheel toggles will allow checking the movement when the editor is in play mode.

Also the “Debug Cylinder Width” needs to be set for each track. If the debug cylinder does not cover the volumes of the wheels properly, using a negative value should correct it.

Optionally the debug colours can be changed so it is possible to further identify the wheels by track.

The script drives all the movement based on the drive wheel and the track so these should be set up first.

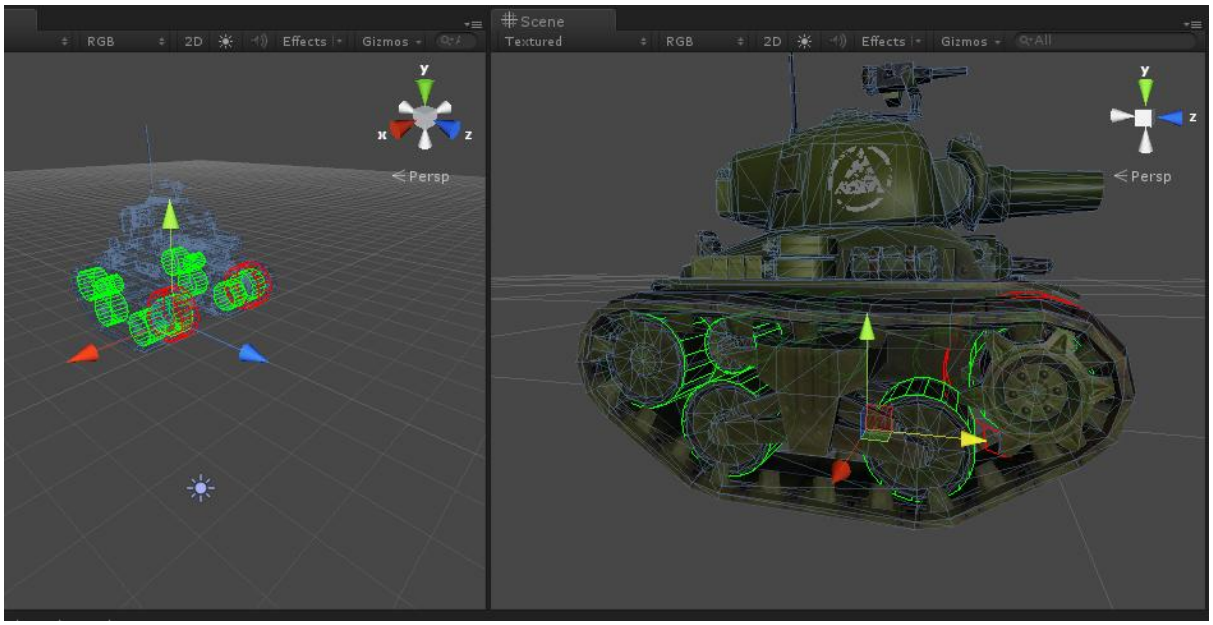
For the example the drive wheel has 6 cog teeth so tooth count on the drive wheel should be set to 6 and the radius to an appropriate value.

The track texture wrap count is the number of times the track texture is repeated around the length of the track mesh, for the example it is 5 times.

The track tooth count is the number of times one cog tooth's worth of drive wheel fits round the track. For the example this is 20 track teeth.

The track thickness is just the thickness of the track.

The individual non drive wheels then need to have their radius values set to the appropriate lengths.



Then it should be a matter of tweaking the various values until the tracks and wheels move convincingly.

For example, using a larger radius for wheels that have the track curving round them may be appropriate.

If the wheels or tracks move in the wrong direction, the "Track Dir" and "Wheel Dir" setups can be adjusted to correct this. However, if all the wheels on one side are not rotating in the same direction, there is an issue with the orientation of the wheel rotation axes not matching. In this case the base models for the various wheels will have to be corrected.