TRACK TECHNOLOGY DOCUMENTATION

JANUARY 2012

Let me start by stating that this document is not a full-on tutorial and does not teach you how to draw or map a polygon. There are already plenty of tutorials on the basics of 3D design on the internet. Instead, assuming you already master the basics of rFactor track building, this collection of information should put you in a position to implement the new technology in an existing track project.

The main goal of this document is to provide people with a non-technical background (i.e. “in English”) of the new features and how they are meant to be applied. This is by no means a definitive piece of text though, and will most likely be updated in time with added information and further refinements where precisions need to be made as modders hit potential stumbling blocks this document fails to cover. Troubleshooting and fixing mistakes will undoubtedly deserve their own articles on the rFactor 2 Wiki (`http://wiki.rfactor.net`) if and when the need arises.

Good luck, and happy modding!

First export of a track

The process of exporting a track hasn't changed much at all. A scene is still built up by loading the various object instances listed in the SCN file. However, with the introduction of the various new features such as RealRoad and extended animations, some new attributes and options have been added to the exporter. These will be covered in the next sections of this document.

At this point, it's probably a good idea to mention a few things you have to keep in mind before adding the new technology. Worth noting is that the old `Sky.gmt` object is obsolete and does not need to be exported anymore, because the sky and cloud system in rFactor 2 is now a global and independent component.

Something else to keep in mind is that the timing beams, commonly called `XSectors`, are now a necessity in order to load a track in the game. If they are missing, the game will crash. In that case, you'll hopefully remember this line before starting a trial-and-error process to find the culprit.

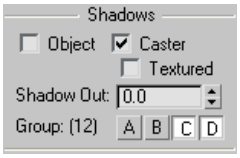
Polygons which make use of chroma transparency are, and always have been faster to process than alpha blend transparency. In the past, chroma transparency seriously lacked in the quality department, and this was a reason why most people would always aim for using the alpha blend method[1]. Fortunately, chroma transparency quality has been improved significantly, and although the traditional alpha blend will still look slightly better, chroma transparency now offers satisfying quality results without the drawbacks alpha blending has, making it a much better compromise that should always be considered when setting up materials, especially when they originate from older assets[2]. **So please give chroma transparency the chance it deserves: it looks alright, and doesn't hurt performance the way alpha blend transparency does, dramatically**.

Lastly, it is now recommended to model painted road lines, and export them as separate objects, detached from the road surface. This will allow you to set up a different TDF reaction, as the lines should offer less grip when they're wet. Previously, this wasn't something people used often, because the lines would flicker with the surface. This is no longer the case when the line's object is tagged as a `Decal` in the exporter. The only requirement is that the Decal objects should sit 0.001m (1mm) above the surface, so the engine knows it should be rendered on top of it. This is not a problem for road lines, as the paint used in real life is not infinitely thin anyway. If the 1mm physical bump of decal road lines is a problem for physical accuracy, then this offset can be undone in the TDF, as explained later. So, Decals and their resulting flickering should now reduce the number of track builders' headaches.

[1] Of course the alpha blend method had its own disadvantages, and people ended up using both at the same time, at the cost of extra performance, using cloned chroma/alpha blend meshes.
[2] So if you decide to go for chroma transparency, don't forget to remove the cloned alpha blend version some rFactor tracks used.

Casting shadows

By design, all objects in rFactor 2 are self-shadowing. However, this doesn't mean that an object will cast a shadow on other objects. In fact, this is turned off by default for obvious performance reasons. Shadows will hurt performance, and therefore require some attention to get a good balance between visual pleasure and performance. This means, for instance, that none of the TrackSide Objects (TSO) cast any shadows on the terrain.



Setting up an object as a shadow caster is as easy as checking the `Caster` checkbox in the Shadows section of the Instance rollout, and providing a `Shadow Out` value. This defines at which distance the engine will stop rendering the object's shadow for performance reasons. As rFactor 2 reaches Gold stage, this section could be enhanced by the inclusion of a "general convention" table for shadow rendering distances, so everyone will be in roughly the same playing field.

Rendering a `Textured` object's shadow requires more processing power, as it will take the textures (and their respective alpha channel) into account. This is only useful for objects such as fences or vegetation, and should not be used as a quick solution for the entire scene, again, for performance reasons. It is therefore advisable to `Detach` all polygons using materials involving transparency to a separate object. This way, you can tag the object containing transparency as a `Textured` shadow `Caster`, and leave the original object minus transparent polygons as a basic `Caster`. Once more, **performance is the reason why the more advanced "Textured" shadow rendering method should not be used for objects that have no visual benefit from it**.

Lastly, the `Shadow Group` controls at which detail level[3] the shadow will be visible[4], with `A` to `D` corresponding with Low to Max Shadow Detail respectively. Please be aware that setting the Shadow Group to High (C) does not mean it will also be enabled when you are running Max Shadow Detail (D). Although this is an optional object parameter, it is recommended to give Shadow Groups some thought. Those with less powerful systems will appreciate it. Again, as rFactor 2 reaches a more polished and final stage, this section could be enhanced by the inclusion of a "general convention" table for shadow groups.

One important fact to keep in mind is that shadows are a reaction to an object occluding a light source (i.e. the sun). The sun in rFactor 2 has to be able to 'see' the polygon in question for the engine to render its shadow. This may seem very obvious, but some older tracks only have one-sided walls for instance to improve performance (i.e. only the polygons facing the player). Because the backside of the object is missing, the rFactor 2 sun will not see the polygon that would generate the shadow, simply because it is … missing. It's important to keep this in mind when you find out that some shadows are not being generated. Luckily, this can be fixed easily by quickly creating the required polygons for the backside. Selecting the border edges and extruding them in the right direction will do the job, and there is no vital need to perfect the mapping on these polygons, because the player won't be able to see them as that's the reason why they were taken out in the first place! Of course, you may want to apply some believable mapping to those polygons as in some cases they might be visible in extreme trackside camera views.

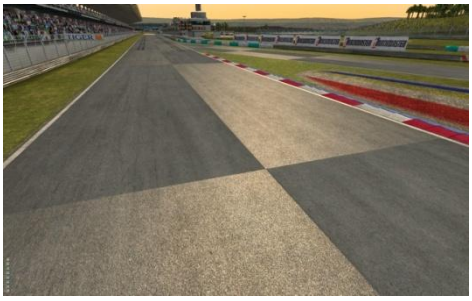
[3] This is linked to the ingame shadow detail GUI element.
[4] In essence, these are similar (but inverted) to the standard VisGroups, but operating on a shadow level.

Implementing RealRoad

RealRoad is probably the most important feature everyone has been looking forward to. To activate the dynamic RealRoad technology on a road surface, you should assign a material to the road polygons using the `Road Shader Two Diffuse Maps` shader. A quick glance at the shader description gives the impression that we're dealing with something fairly complex, so let's go over all the various texture stages to set up the road material:

Stage	Type	Map Channel	Description
1	Diffuse map	1	Classic, basic road texture (requires alpha)
2	Multi map	2	Asphalt detail map
3	Normal map	1	Highlights diffuse map's bumps and cracks
4	RaceGroove map	3	Rubber that will be put down by track activity
5	Specular map	2	Pronounces Stage 2 detail map under sunlight
6	Marbles map	2	Rubber bits and debris off the racing line
7	Reflection map	3	Reflects the environment when wet

Making the material change will undoubtedly require a bit of texture reworking, as the diffuse map now requires an Alpha Channel. This is where things can get tricky, depending on the complexity of the diffuse map and the amount of detail it contains, because this Alpha Channel not only controls the amount of specular intensity, but its inverted version is also used to control the wet look of the surface. An extreme example, using a black/white chequered Alpha Channel, is shown in the images below, both taken at sunset from the same position:



Notice that, as the sun goes down, the specular effect on the road in the image on the left is what you'd expect in places for white pixels in the Alpha Channel. The dark patches are black in the Alpha Channel. This means that the **specular effect gets stronger where the Alpha Channel is lighter**.

In the same way, you can see in the image on the right, taken on a wet track, that the totally black alpha bits now act as a perfect mirror[5], whilst the white bits, which resulted in a great specular effect, now appear bone dry, even though the surface has been set to be at its wettest. This means that the **wet effect gets stronger where the Alpha Channel is darker**.

The reason why the specular and wet effects react in an inverted way related to each other is because small cracks in the diffuse map would be filled with water if it rains (dark alpha pixels), yet at the same time they would be less affected by specular effects because the crack is subdued in the surface (again, dark alpha pixels). Both specularity and wetness need to be kept in mind when working on a road map's Alpha Channel.

To make the Alpha Channel an even more crucial element is the fact that repetition can become a big issue if you're not careful and subtle enough when trying to find a contrast balance between

[5] Not quite perfect, as it's distorted by a normal map.

darkness and lightness, or in other words wetness/specularity. This is where experience and a few trial-and-error iterations will come in handy, so don't be disappointed if the first effort doesn't end up looking the way you had planned. On the track texture side, this texture's Alpha Channel is probably the bit you'll want to spend as much time as possible on.

Detail maps for texture stages 2 and 5 can most likely be borrowed from one of the stock tracks. These work best with tight tiling values, around 2.0m x 2.0m. Be aware that using extremely tight tiling values comes at the risk of having visual repetitive patterns.

You'll also need a subtle normal map based on the diffuse map for stage 3. It's very important that this map is kept subtle, because it can look intense and critical at extreme sun positions or on wet surfaces[6], which wouldn't look natural or realistic at all.

Stages 4 and 6 are the visual representation of RealRoad's track evolution system. As the rubber gets laid onto the track surface, RealRoad will allow stage 4 to become visible according to the grip levels. For a more immersive result, the groove uses its own texture, instead of just darkening the diffuse map. Similarly, all bits of debris and rubber that get pushed off the racing line (i.e. the effect known as Marbles) are put in another separate texture. Both RaceGroove and Marbles textures that ship with rFactor 2 have been designed to work with a large variety of road surfaces, so this requires minimal work on the texture side.

Notice that the RaceGroove and Marbles use their own dedicated Map Channel. This is done because the RaceGroove contains skid marks which greatly enhance the organic character of the surface. These skid marks are nothing more than horizontal lines in the RaceGroove texture. Allowing this texture to use the same Map Channel as the diffuse map, would look totally wrong, as the rubber markings would not really follow the curve of the ideal racing line. So we'll need to tweak the mapping for these RealRoad textures. The best way to map the RealRoad textures is by using Spline Mapping, which would approximate the ideal racing line. Although recent versions of 3ds Max have a built-in Spline Mapping feature, there is always a workaround method by using a `Projection` modifier in older software versions. This procedure is explained in Appendix A.

Worth noting is that the RealRoad texture stages are actually "T1 Specular Alpha" stages, which will also alter the surface's specularity as the sun interacts with the rubber. This means that the RGB portion of the texture acts as a multiplicative blend with the diffuse map, and the Alpha Channel controls the specularity[7]. Because of this multiplicative nature, it is advisable not to go too dark in the RGB range.

Stage 7 is used for the wet surface reflection. The only point of importance here is that `Live Mapper` should be enabled, to inform the renderer that this stage should not be handled as a traditional texture stage, meaning it doesn't matter which texture and mapping channel you assign to this stage.

That covers all basic material settings. There is one last change that needs to be made, in the Instance rollout of the GMT Exporter. To activate RealRoad, the `Deform` option should be marked, and the object name should start with `RaceSurface_`.

[6] Small cracks love normal maps, and will give great results even in the wet. Bumps and larger dips, on the other hand, will end up looking like small mountains in low sun conditions, so you may want to blur those out a little at any point during the normal map creation process.
[7] White Alpha results in more specularity.

Terrain blending

Seams seem to be the nightmare of every track builder who wants a clean looking terrain yet still make use of different textures for extra variation. In the past, multitexturing and clever shader usage were the only ways to minimize the seams caused by the transition between the various textures. Not anymore …

One of the advantages of linear interpolation is that we can blend between gMotorTextures. Just like RealRoad, this will require a bit of extra geometry in the terrain which will serve as key points to control the blend[8].

The blend level (0.0 – 100.0) is controlled by the `Vertex Alpha` property. This can be done either manually, vertex by vertex, or by using a `VertexPaint` modifier. The latter option will allow you to gradually paint the blend onto the mesh. Just make sure that you're painting on the Vertex Alpha channel. With the mesh now set up for blending, all you need is a shader that allows the graphics engine to deal with the blend.

The "`Blended Grass Infields`" shader is a universal solution that is especially effective for blending different grass materials, as the name suggests. Let's take a look at the shader description: "diffuse + specular map, with bump map, T1 lerp T2 according to vertex alpha, mul T3, T3 and specular masked by T1 lerp T2 alpha".

In plain English, this means that the blend interpolation is done on two levels: a blend between two diffuse textures T1 and T2, and their respective alpha channels, according to each vert's alpha attribute as explained above. The blended alpha channel masks the detail maps T3 and T4. This means that the detail maps will be shown for areas with non-zero alpha[9]. Pure black alpha channel pixels will result in a diffuse-only output, so there will be no specularity or a multiplicative T3 blend.

Note that the normal map is not part of the blend and will be applied everywhere, regardless of the vertex alpha or the alpha channel, and uses the same mapping channel as T2. In most cases, this will decide which texture you want to use for which stage[10].

Because of a bug in 3ds Max, it is strongly recommended to start blending once the terrain geometry is finalized. Let me repeat that: **do not blend until the geometry is final**. This is because the vertex numbering will change when vertices are added, and 3ds Max will not update the Vertex Alpha channel appropriately. Removing vertices will not have any detrimental impact on the blending, but always make sure that you do not need to add extra vertices before you start blending.

Should you be forced to add more vertices, you will have to perform a procedure to clear the Vertex Alpha information and start the blending process again, from scratch. This procedure is explained in Appendix B.

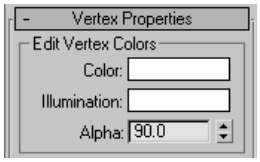
[8] Pretty much like the use of 'key frame' parameters in animating or video editing.
[9] In other words, anything other than pure black alpha channel pixels.
[10] For instance, a tightly mapped 2m*2m grass T2 map will require a different normal map than when you'd use a 10m*10m dirt T2 map; i.e. the normal map would also be stretched to 10m*10m in the latter case.

Billboards

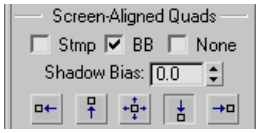
One of rFactor's visual annoyances was the fact that 2D vegetation had to be represented in the form of an X shaped object for it to give a better impression of its volume. This meant that tree textures could not be complex if you wanted to hide the seam along the middle, and there would always be a risk of having mirror-patterned trees.

In order to 'fix' this, rFactor 2 'introduces' an old technology called billboarding, which means that the object always faces the viewer. The interesting part of its implementation in rFactor 2 is that billboarding can now be done on an element level, as long as the elements are tris or quads. Of course this could have been done in rFactor as well, with each tree being exported as a separate billboard object. This, obviously, would have resulted in a massive amount of objects, which is not a very optimized way of doing things. So, in essence, individual billboarding no longer comes at the cost of performance[11].

The technology requires the use of shaders which contain "Stamp Vertex". If you look at the description of one of these shaders, you'll notice that "`normals and faces always face viewer for non-zero alpha`". Vertex Alpha is the property that activates billboarding for the face of which the vertex is part of[12]. To enable it, simply select the vertices of the polygon and set the Alpha in the Vertex Properties rollout to any non-default value[13].



In the past, when a complete object was marked as `Billboard` in the exporter, the object would turn around the object's pivot point as set up by the user. So now we face the 'problem' that each element inside the object needs to have its own correct pivot point. This is done by means of the "box and arrow" buttons in the `Screen-Aligned Quads` subsection of the `Instance` rollout. They indicate in which direction, from the centre of the plane, each plane will pivot. So, a 2D tree, for instance, would have the pivot set to the bottom, as shown in the image in this paragraph. A quick test of all buttons will intuitively show the different reactions of each element's pivot point. It's not unimportant to mention at this point that the Z axis of the object should obviously point up to prevent unexpected results.



The use of billboarding doesn't have to be limited to a refreshed implementation of the old billboard tree system. Another interesting application of this technique would be to do something similar with so-called 'leaf cards' – the building blocks of a full 3D tree – which consist of small branch groups with leafs. These can then be placed in key positions on larger, modelled, branches and trunks. Only this time, you'll want the quads to always face the viewer in all axes, and their pivot point to be centred, to give a better impression of the tree's volume when the lighting interacts with the object. This is done by changing the Billboarded (`BB`) tag into Stamped (`Stmp`) in the `Screen-Aligned Quads` subsection, and changing the individual pivot point to the centre[14]. In other words, the `Stmp` (2), `BB` (1), and `None` (0) options control the number of axes an element will pivot around.

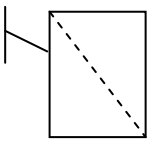
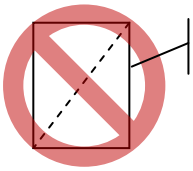
By applying this technique to a 3D tree, the viewer will perceive the tree with complete leaf coverage, when in reality you've actually only used about a quarter of the precious polygons required to come up with a similar tree without using this technology.

---

[11] Classic billboard OBJECTS also remain supported in exactly the same way as before.
[12] So you don't need to create a second material without the stamp properties for the remaining faces.
[13] Please note that the default 'zero' value is actually 100.0; I typically use 90.0 for no real reason.
[14] This is represented by the 'Box with Arrows pointing in all four directions'.

Performance is not the only reason why you should consider making use of billboards. Let's take the crowd as an example. In the past, 2D crowds would be described as plain and simple 'cardboard people'. We could now replace those with individual people polygons, and billboard them like a standard 2D tree as described above. So although the 'cardboard people' in the rFactor 2 grandstands remain 'cardboard people', they will now face the viewer to give a better impression of filled grandstands from all angles, without having to detach them individually to billboard objects. This may seem like a minor improvement, but in stressful race conditions, subtly animated billboard crowds will now do a much more believable job of tricking the human eye, and having individual crowd elements means the grandstands can be filled more randomly than before.

One last thing that needs to be kept in mind when adapting existing objects to make use of the **Billboard** screen-aligned quads technology, is that each quad needs to be triangulated from the top-left to the bottom-right corner as shown below. Top-right to bottom-left triangulation will result in the quad being rendered 90° rotated, as the renderer is set up to expect the quads in a top-left-to bottom-right format. This mainly applies to billboarding, as stamped quads will pivot around two axes anyway, so in most cases, it doesn't really matter if they're rotated 90°.

Top-left to bottom-right   Top-right to bottom-left

To be on the safe side, it may be good practice to reset the transform after making sure the triangulation is set from top-left to bottom-right and the object's Z axis faces up, before cloning the objects. At the end of the cloning process, one can simply attach and export them as a single object.

Triangulation is not the only important factor. The way the quad is drawn in 3ds Max decides the vertex numbering, which in turn defines the base line for the quad's pivot point. If the numbering is incorrect, it will result in the quad being rendering upside-down, because the top edge is being used as the pivot's base line. A quick Local 180° rotation action in Polygon mode and a UVW Xform flipping the V coordinates will fix this.

Because billboards can be tricky to set up, it is recommended to export a test object before cloning.

Billboard and Stamp technology may have been written-off by many, but hopefully this section proves that it still offers a no-brainer balance between eye-candy effects and performance, with a minimal amount of work involved.

Reflections

There are three types of reflection maps in rFactor 2: REFLECTEDENV, Static## and REFMAP0. The latter is used for the cube map for car reflections, and although they are not used by track parts, it is necessary for track makers to add it in order for the cars to have dynamic reflections. So let's start with that.

Real-time dynamic reflections (cars) – REFMAP0

This is the name of the reflection mapper that cars use for real-time, dynamic reflections. To create it, use the "Reflection Mapper" panel of the export plugin. In the "Reflection Mapper:" dropdown, type in "REFMAP0" and hit the enter key. This mapper should now appear if you unroll the dropdown. REFMAP0 doesn't require anything other than the user telling it how often to update, the map size and which objects should be reflected. Typical values to set are:

- Update Rate: 100.00
- Tex Size Slider: 512
- "Cubic" Radio Button: Set

In the "Reflector Instances:" portion of the roll-out, set:

- "Normal" Radio Button: Set
- "Reflected" Radio Button: Set
- "Include" Radio Button: Set

Now, click the "select list" button located next to the instance list, and start multi-selecting the objects you want to be reflected by REFMAP0. Now, when you export the .SCN file, you should have something similar to this:

```
ReflectionMapper=REFMAP0
{
  Type=Cubic
  TextureSize=(512)
  UpdateRate=(100.000)
  StaticSwitch=(100.000)
  TrackingIns=True
  IncludeIns=SkyBoxi
  IncludeIns=RaceSurface_01
  IncludeIns=RaceSurface_02
}
```

REMEMBER TO DOUBLE-CHECK: Sometimes you may see this:

```
  StaticSwitch=(100.000)
  Pos=(0.000000,0.000000,0.000000)
```

Rather than this:

```
  StaticSwitch=(100.000)
  TrackingIns=True
```

If this happens, just replace the "Pos=(0.000000,0.000000,0.000000)" line with "TrackingIns=True".

Wet weather dynamic surface reflections (REFLECTEDENV)

This reflection mapper is used by the track surface for dynamic wet reflections. To create it, use the "Reflection Mapper" panel of the export plugin. In the "Reflection Mapper:" dropdown type in "REFLECTEDENV" and press enter. This mapper should now appear if you unroll the dropdown. There are a few extra steps in order to properly create this mapper. First, set up the map values in the same manner you did REFMAP0 using these settings:

- Update Rate: 100.00
- Tex Size Slider: 512
- "Planar" Radio Button: Set

In the "Reflector Instances:" portion of the roll-out, set:

- "Normal" Radio Button: Set
- "Reflected" Radio Button: NOT Set
- "Include" Radio Button: Set

Now that the basic mapper is set up, you'll need to set up not only a Global reflector plane, but also reflector planes for certain objects as well as set object reflection properties. Let's go step by step:

The Global reflector gets used when no Local reflector is set for any given object. To create it, simply make a Plane Object in 3ds Max. Do NOT turn it into and Editable Mesh or Editable Poly—leave it as a Plane. Name this object "refPlane_Global" and then position it where needed. This would be useful for a track that is VERY flat. The plane object can also be tilted so that it aligns better with the ground. Now, in the "Scene File" rollout of the exporter click the Select List button next to the "Reflection Plane:" field and pick your "refPlane_Global" object.

The next step will be assigning objects with reflection properties. You can do this either one at a time or on a group of objects. To do this, select the object(s) in question and use the "Reflect" portion of the "Instance" rollout in the plugin. Check "Reflect". If you want to use the global reflection plane then keep "Use Local Plane" unchecked. But, if the object sits above or below the global plane then you'll want to make a reflection plane specifically for it. Do this by:

1. Create a Plane object;
2. Position and orient it under the object in question;
3. Name the Plane object "refPlane_<some_name>".

Now, check "Use Local Plane:" and select the reflector plane object you just created. Now the object is ready to be assigned to the REFLECTEDENV mapper.

Do this in the same manner as you did for adding objects to the REFMAP0 mapper. The only difference here is that, since you left the "Reflected" Radio Button un-set, ONLY objects that have been set to "reflect" will appear as valid objects to be added to the mapper. Select what you need for this mapper. Please note that even hidden objects that are set to "reflect" will appear in the selection list.

The last step is to set up materials to use the REFLECTEDENV mapper. These will be for things like the track surface or water. For instance, create a material for your track surface. The typical track surface shader is: "Road Shader Two Diffuse Maps". Notice the greyed-out "Reflection Mapper:" drop down in the upper right corner. To activate this go to the final texture stage ("Reflection"), add

a placeholder texture just to assign something to the "Bitmap" slot, and then check "Live Mapper". Now, when you back out, you'll see that the "Reflection Mapper:" drop-down is now active. Open it and select your "REFLECTEDENV" mapper.

As long as you have at least one object assigned to the mapper you should see the sky and whatever objects you selected reflected by a wet track surface/water surface. In the .SCN file you should see something like this:

```
ReflectionMapper=REFLECTEDENV
{
  Type=Planar
  TextureSize=(1024)
  UpdateRate=(100.000)
  StaticSwitch=(150.000)
  TrackingIns=NULL
  IncludeIns=SkyBoxi
  IncludeIns=InflTubA_01
  IncludeIns=InflTubA_02
  IncludeIns=InflTubA_03
  IncludeIns=InflTubA_04
}
```

One final note: You do not need to apply a material to your reflector planes, and they should NOT be exported, nor be included in the .SCN file, other than the reference made to them by the objects that use them.

Static Cubic Reflectors

Lastly, you can also create static cube reflectors for non-moving trackside objects, such as building windows and TSO cars. You can have as many of these as you like. The mapper closest to an object will be the one used. In most situations one or two mappers are sufficient.

WARNING: The more static cube mappers you have the more video resources they require, particularly in texture memory!

We'll set up each static mapper as we did the REFMAP0 mapper. Each static cube mapper must be named "STATIC01", "STATIC02", "STATIC03"…

Use these values:

- Update Rate: 0.1
- Tex Size Slider: 512
- "Cubic" Radio Button: Set

In the "Reflector Instances:" portion of the roll-out, set:

- "Normal" Radio Button: Set
- "Reflected" Radio Button: Set
- "Exclude" Radio Button: Set

It should be noted that for these static cube mappers it may be more efficient on set up to "Exclude" rather than "Include" objects. When you exclude you'll pick the objects you do NOT want to be reflected by the mapper, and everything else in the scene will automatically be included. You CAN use "Include" if you prefer, but often "Exclude" is easier to work with.

The other thing you need to do is set a position for the cube mapper. You can either enter the x, y, z values in the 3 fields below the “Get Location From Ins:”, or you can check “Get Location From Ins:” and then pick an object. The pivot point of the selected object will be used for the position of the cube mapper.

Now, click the “select list” button located next to the instance list, and start multi-selecting the objects you want to be reflected by the Static Mapper. Now, when you export your .SCN file, you should have something similar to this:

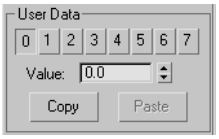
```
ReflectionMapper=STATIC03
{
  Type=Cubic
  TextureSize=(512)
  UpdateRate=(0.100)
  StaticSwitch=(100.000)
  Pos=(-299.439026,3.663275,65.743027)
  ExcludeIns=BldOffice_01
  ExcludeIns=Garage_IntDetails01
  ExcludeIns=Garage_IntDetails02
  ExcludeIns=Garage_IntDetails03
  ExcludeIns=Garage_IntDetails04
  ExcludeIns=Garage_IntDetails05
  ExcludeIns=Tires_HighDetail01
  ExcludeIns=Tires_HighDetail02
}
```

One final note: Any material with a “cube” shader will automatically use the static cube mappers. The cube mapper closest to an object with a reflective surface will be used for that object.

Applying Wind Effects

One of the environmental additions is wind. Obviously, not all objects should have wind interaction, so at some point during the conversion process, it will have to be set up individually for all the various vegetation elements that make use of a Stamp Vertex shader.

Load the exported track in the gMotor2.0 Scene Viewer or start rFactor 2's Mod Dev Mode in order to get access to the Wind panel, and simply 'walk' or drive to the object you wish to set up. Playing with the various wind values is a matter of empirical trial-and-error[15], but once you're happy with the result, write down the eight values.



Now, return to Max, select the object and look for the `User Data` section in the exporter's `Instance` rollout. Notice that there are, again, eight data fields. Manually enter the numbers you've written down with 0 corresponding to the `Leaf Pitch Displacement` value and 7 corresponding to `Wind Velocity`, although you will want to keep the latter at 0.0, as this field is hooked up with the actual wind code.

If the wind movement is really subtle, it can also be used to add an ambient animation to other Stamp Vertex objects, such as the crowd. The procedure is exactly the same, except for the `Wind Velocity` (User Data field 7), which would have to be initialized to a non-zero value. This is because the crowd movement is not related to the environment variables. By setting a fixed value, the crowd will always be in motion. By using the wind effect to an object like this, billboarding can look that tiny bit more natural than it used to.

Here is a table containing commonly used User Data for the stock rFactor 2 content:

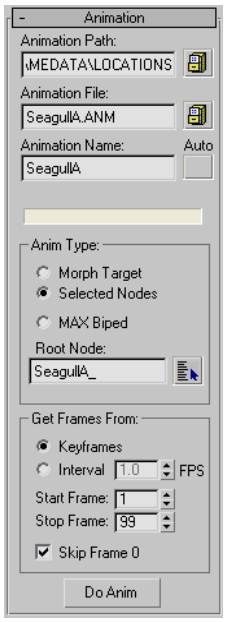
	Pitch	Yaw	Wind Displ.	A	B	C	D	Velocity
Grass	0.075	0.05	0.05	3	1	1	3	0
Shrubs	0.025	0.05	0.05	3	1	1	3	0
Small/Medium 2D Trees	0.02	0.04	0.04	0.28	0.45	0.45	0.28	0
Large 2D Trees	0.025	0.05	0.05	0.3	0.5	0.5	0.3	0
Small/Medium 3D Trees	0.05	0.05	0.06	1.9	0.1	0.1	1.9	0
Large 3D trees	0.05	0.05	0.06	2.5	0.1	0.1	2.5	0
Grandstand Crowds	0	0.08	0.01	20	1	1	20	0.075
Ground Crowds	0	0.08	0.01	20	1	1	20	0.025

[15] Looking for a subtle and natural looking result that doesn't immediately catch the eye is the key here.

Animations

In the past, track artists had to use their creative imagination to bring a track to life, being limited to the use of basic and rotational animations. rFactor 2 allows modders to stretch their imagination, as it is now possible to export keyframed animations directly from 3ds Max into an `.ANM` file. Although animating was never a part of rFactor and is probably new to a number of modders, this document will not go into the basics of rigging and animating an object. There are many decent tutorials on the web that cover the art of rigging biped and bone-based animations.

Since car modders will be more frequently confronted with biped animations for the driver, let's put our focus on exporting a bone-based animation. To prevent any potential and unwanted glitches when instancing the animation in the scene, always set up the animation at the world origin (0, 0, 0), so make sure to move the object in a position so that the pivot point is located at the world origin, and `Reset XForm` to make sure the object is entirely reset and aligned with the origin. The world origin is also where the Root Node bone should be located. This Node is used as a reference point for all other nodes/bones.



Once you're happy with the rigged object, try creating a couple of keyframes and move/rotate the bones as a test. To create the perfect loop, it is always a good idea to copy the first frame to the end of the Timeline. Now let's focus on exporting this basic test animation. Open the Animation rollout in the exporter as shown on the left and set up the Path and Filename. For simplicity, the Animation File should be the Animation Name appended by the .ANM extension. We're dealing with a bone/node-based animation, so the Animation Type should be set to Selected Nodes. List-select the Root Node (the one located at world-zero). Since this is a Keyframe animation, Get Frames From: Keyframes and enter the Start and Stop Frames according to the frame numbers in the 3ds Max Timeline. Frame 0 can be skipped if the first and last frames in the Timeline are identical. By doing this, a small stutter will be avoided when the animation loops back to the start. The same effect can be achieved by making the animation one frame shorter obviously. Move the Timeline to the Start Frame and create the .ANM animation file by hitting the "`Do Anim`" button.

Now, select the rigged object and go to the Instance rollout in the exporter, and scroll down to the Animation section. Select the newly-created .ANM file through the browser. This will add a line in the object's SCN Instance that links the object to the animation file.

Setting up High Dynamic Range Profiles

rFactor 2's scene lighting uses a physical simulation of the real sun and its interactions with the atmosphere and terrain. In HDR mode, the lighting values are real world luminosity values expressed in kilo lumens (1000 lm[16]), so we get actual luminosity values of 50 to 150 or more klm. These are tone-mapped down to the range 0.0 to 1.0 so they fit into the standard display.

Before setting up HDR, you need to have a basic understanding of each setting that affects the HDR output. Here's a short overview of the settings in the PostFX menu[17]:

The two most important values are "`T Lum`" and "`S Lum`". T Lum is transmitted (direct) sun luminance. This affects everything including clouds, but not the sky. S Lum is scattered luminance, which is light that bounces off the terrain and illuminates primarily the sky. Think of these as T(errain) Lum and S(ky) Lum. These two values are the key to getting the right balance between terrain brightness and sky tone.

`Terrain Lum` can range from 0.15 to 0.80 at mid-day in clear skies, and you usually want to increase it slightly as you get towards dawn/dusk.

`Sky Lum` can range from 0.35 to 1.0 or higher at mid-day, you usually want to decrease it near dawn/dusk to take the big white areas out of the sky. "`White point`" is also helpful for reducing big white sky. S Lum is critical to getting the proper realistic tone for the sky. If it is set too low, the sky will have an unnatural purplish or cobalt blue look that doesn't mix with the terrain. You can adjust T Lum and S Lum together, as you raise one, raise the other. The tone mapper will adjust to rebalance them.

Adapt Time refers to the range of luminance history used to average out scene luminance (in seconds). Basically, this setting defines how fast the tone mapper will adapt to changes in brightness[18]. Values below a couple of ms will cause the scene to flash annoyingly.

White Point defines the minimum luminance value which gets mapped to 1.0. Raise this number to decrease the blown-out whiteness of the scene. Too high a value will take all of the edge out of the scene.

Post Gamma: the tone mapping process must be done in linear colour space (which rFactor 2 is not using). This means the HDR scene must first be linearized (gamma corrected to 2.2). After tone mapping, it is transformed back to gamma space (inverse gamma correct). Post gamma controls this value. It should be 2.2, but you can adjust it to make the scene more washed out or more vibrant, or to correct for other extreme settings.

Auto Exposure is a constant in the modified Reinhart tone mapping equation, usually hard-wired to 1.03, but having it as an adjustable variable can prove to be useful.

rFactor 2 comes with a set of default HDR values for these settings. However, in order to free a modder's creative hands, there is a way to override these values, by creating a custom `*.HDP` profile for a track.

[16] The symbol of Lumen is 'lm'.
[17] This menu is only visible if HDR has been enabled in the rF Config tool.
[18] This is similar to the time the human eye requires to adjust to sudden changes in luminance.

Profiles should be placed in each layout's subfolder and should use the following naming convention:

- `<TrackLayoutName><Identifier>_CLEAR.HDP`
- `<TrackLayoutName><Identifier>_OVERCAST.HDP`

The Identifier is used to differentiate between different styles, such as a Realistic or a Cinematic look. If no Overcast profile exists, then only Clear will be used.

A profile consists of an unlimited list of 'reference point' elements throughout a full 24 hour cycle, meaning you can create as many elements as you want, should you wish to fine-tune the profile on a very detailed level. The engine will linearly adjust the lighting between elements. Normally, one element per hour should suffice to be in total control, but again, you can add as many points as you want. You'll notice that extreme conditions such as sunrise or sunset may require an extra element for better results.

To set up a new profile, load the track in Developer mode, go to the PostFX menu and "Save Profile". Now, go to the "Time/Location" menu, set Fog to Exponential, and set "Time Scale" to 0. This will stop time. Switch back to the PostFX menu and, using the time value, scroll time to the top of any hour[19]. A good suggestion would be to set the time to match any suitable reference material you may have at your disposal.

Now, click "Add Element". Adjust the values so the lighting is to your liking. Proceed to the next hour, and add another new element. Again, adjust the values, and repeat these steps for every hour. Again, around sunrise and/or sunset you may have to fine-tune at every half hour or more as the lighting changes very quickly.

You can use Previous Element and Next Element to quickly cycle though all your HDR reference points. When you've finished all 24 hours you can enable the profile by setting "Run Profile" to on and start time again by setting Time Scale to 1 or higher[20]. When you're happy with the results, do not forget to save the profile again. To go back and work on the same profile later, use "Load Profile".

Instead of starting from the default values, you could also "Load" a profile from another track and use that as a starting point and use the "Previous Element" and "Next Element" controls to quickly adjust each pre-defined reference point.

To set up an Overcast file, load the track with Overcast and Storm for all the time periods. You want storm as well because fog affects HDR a lot and has to be accounted for. Now, load your 'Clear' profile, cycle through each reference point and tweak it for the current conditions. Normally, you can just bump Auto Exposure down a little bit if it's above 1.00. Also, normally, the change you make for one reference point will generally work for all of them, but this isn't ALWAYS the case. When all elements have been updated, "Save Profile" with the new _Overcast name. It's also a good idea to backup the _Clear file before you work on the Overcast profile, in case you accidentally save to the wrong file.

As a side note: users can create their own profiles and save them to special track folders in the `UserData` area. You can see the gizmo to select a profile to use ingame if other profiles exist.

[19] It's easier if you work at the top of every hour, just to keep things organized.
[20] Time Scale would usually be higher so you can quickly see the transitions throughout the day.

Finishing touches

Finally, these are a few small but important bits that didn't fit elsewhere ...

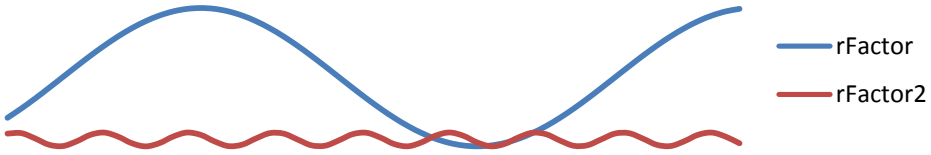
TDF File

In the past, the racing groove was a visualization of the AIW's Fast Path. This reaction was defined in the track's TDF file. Since the groove is now tied to the track surface, there is no more need for a graphical representation of the Fast Path. To remove this, look for the `[LOOSE]` section in the TDF and disable the line that starts with `Reaction=groove` either by deleting it or commenting it out by adding `//` in front of the line.

Having more geometry in the road surface not only means that the grip levels (and groove) are at the discretion of people's racing line and track activity, but because it is now possible to generate (and tweak) Noise levels for all these new vertices[21], it also means that the fake sine wave response can be taken out of the equation. Let's take a look at the differences between the Track Variables of a typical rF1 track and an updated rFactor 2 track. These can be found near the top of the TDF file.

rFactor	rFactor 2
`[TRACKVARS]`	`[TRACKVARS]`
`RoadBumpAmp=0.010`	`RoadBumpAmp=0.001`
`RoadBumpLen=13.0`	`RoadBumpLen=3.0`
`RumbleBumpAmp=0.015`	`RumbleBumpAmp=0.008`
`RumbleBumpLen=8.0`	`RumbleBumpLen=7.0`

Notice that `RoadBumpAmp` is now ten times smaller, and `RoadBumpLen` is also considerably shorter. The rF1 and rFactor 2 values are shown in the simplified illustration below, though slightly exaggerated in order to highlight the difference.



So why not get rid of the sine wave altogether? After all, it was one of the artificial 'concerns' people have with older tracks. Why does it actually still make sense to keep superimposing a signal onto the terrain mesh?

Firstly, you need to understand that polygon-based 3D meshes are still a collection of samples, whether the source data is a simple point-to-point drawing or a laser scanned point cloud. And although RealRoad allows us increase the resolution of the mesh, it is still a (sampled) approximation, no matter how many vertices you add. As tire models get more and more sophisticated nowadays, you'll be able to sense those sampled imperfections, and the result will feel 'wrong'. Therefore, it's a

[21] As in physically modelled bumps

good idea to still apply tiny amounts of noise[22] in the form of a high frequency signal – high enough not to get the impression that you're skiing on an alpine course – with a very small amplitude. Because this is imposed on top of the vertex noise (again, either by using point cloud data to start with or by applying a Noise modifier), the TDF wave will only serve to smoothen imperfections, rather than taking care of the entire road bumpiness effect as it was usually done in rFactor.

And secondly, the TDF waves in rFactor 2 are in fact more than simple sine waves. To reduce the repetition to an absolute minimum, hardcoded "fractal patterns" are used instead of a sine wave for extra randomness. This 'randomly' spiked noise will obviously operate within the specified TDF parameters, so the process of setting up the wave is no different than before, but the result will be different and more unique.

Don't forget that these variables are only used when they are called in the actual `[FEEDBACK]` response of a surface type, so you'll have to go through the entire TDF file and either link the `BumpAmp` and `BumpWavelen` to these variables, or replace the existing values with numbers that will bring the end result in roughly the same ballpark.

To make a modder's life easier, values can now be declared globally in the TDF file to reduce some of the trial-and-error process. These values can then be assigned to the various feedback instances, as shown in the example below:

```
[TRACKVARS]
RoadDryGrip=1.00
RoadWetGrip=0.85
RoadDustGrip=0.90
RoadBumpAmp=0.001
RoadBumpLen=3.0
MiscBumpAmp=0.07

// Roads
[FEEDBACK]
BumpAmp=RoadBumpAmp BumpWavelen=RoadBumpLen OnTop=-0.008
```

Lastly, if you feel you had to make a compromise on physical accuracy when adding Decals, there is an option to undo the damage. For instance, if a painted road line had to be placed 10mm above the racing surface for flickering reasons, you could add the following to the line's TDF entry, effectively making the physical line feel like 0.010 + (-0.008) = 0.002m on top of the racing surface:

```
OnTop=-0.008
```

No-Rain Zones

To create zones in which rain particles will not be drawn, and the track surface will not get wet (such as under bridges, in garages, or tunnels, etc) you simply need to create a Box object in 3ds Max and name it "`NoRainZone_<some_name>`". Shape, position and orient the object to encompass the area (such as the area under a bridge) where no rain should fall. You can have as many NoRainZones as needed, and they can overlap.

Make sure that `Collision` is turned OFF for the object, and that the "`No Render`" checkbox in the "Instance" rollout of the exporter plug-in is checked so that the object is invisible. Also, in order to export the object you'll have to assign a material to it to make it legal for the exporter.

[22] This is very much similar to a process known as Dithering.

Frame Buffer Materials

You can use the frame buffer as a texture. This can be used for video monitors, jumbotrons, etc. This is pretty easy to do. Simply:

1. Set up a material to use as the surface of the video screen. Typically a T1 shader works fine. Be sure to set the Emissive color to full white so it can be seen at night.
2. For the material's T1 texture stage, assign a placeholder texture and check the "`Frame Buffer`" checkbox.

Make sure you have properly UV mapped the surface that will display the frame buffer texture. In order to prevent the effect of the frame buffer infinitely repeating its own image when the frame buffer is being rendered, simply place a black polygon behind it, and offset it slightly to prevent sorting issues.

Material "Offset" CheckBox

Sometimes you just CAN'T get two alpha-transparent objects to sort correctly. This is a last ditch resort to correct sorting and should be used CAREFULLY. Let's say you have alpha-blended painted lines on the track surface, and alpha-blended safety fence, and a car windshield. By default the materials of these three elements would have their "`Offset`" value set to 0, which means they all get drawn in the same pass.

Now, let's say that, with a trackside camera, the painted lines and the safety fence are not rendering in the correct order and the lines appear to be floating above the fence. To correct this you could set the material for the safety fence to have an offset of "1". This will force the safety fence material to render after the painted line material, thus getting these two elements to sort correctly.

Now, in this configuration, while in the cockpit, the windshield may not draw correctly with the safety fence, which now has a higher offset. To fix this you would give the windshield an even higher value.

ISI standards reserve Offset values of 0-4 for track elements, and 5 and beyond for cars. So, track builders should organize their materials to use no value higher than 4, and car-makers should set their cockpit windshield material value to 5 always at least…

Again, this is a last resort fix for sorting issues, but car-makers must be SURE to use a value of 5 or higher for their windshields to work properly with ISI-created tracks…

Steering Wheel track map

Steering wheels and cockpits can now display a map of the track. This is an optional feature, but it does add a nice environmental touch. The game will try to locate a DXT1 texture called `swheel_trackmap.dds` when it loads the track. It is advisable to place this texture in the subfolder containing the SCN, GDB, … files, so it becomes possible to load a different track map for each track layout variation.

Appendix A – Spline mapping in earlier versions of 3ds Max

Older versions of 3ds Max do not have inbuilt Spline Mapping features. An effective workaround for this problem can be used by projecting the UV mapping of a spline-based object onto the existing road mesh, to achieve a similar end result, albeit with a few more steps:

1. Draw a closed 2D[23] spline of the FastPath on top of the existing terrain;
2. Move this spline up so it doesn't intersect the terrain mesh;
3. Clone the spline (Copy);
4. Apply an `Extrude` modifier with an arbitrary height (10m for instance) and make sure '`Generate Mapping Coordinates`' is enabled – you don't need the '`Start/End Caps`';
5. Apply an `Edit Poly` modifier and go into Edge mode;
6. Make sure the extruded edges are selected (`Loop` selection is quite effective here);
7. Apply a `Push` modifier with a large enough value so it covers all of either the track inside or outside;
8. Collapse to `Editable Poly`, select all edges at the arbitrary height level, and move them back, down by the same amount as step 4;
9. At this point you may need to Flip the polygons to make them visible;
10. Repeat steps 4, 5, 6, 7, 8 and 9 with the cloned copy, but this time `Push` it to the other side;
11. `Attach` the inner object to the outer or vice versa and `Weld` all vertices within 0.001m;
12. Assign the RaceGroove texture to the resulting object;
13. Apply an `UVW Xform` modifier and manually enter values until you're happy with the rate of repetition[24];
14. Select the mapped RaceGroove object and apply a `Projection` modifier;
15. In the `Reference Geometry` rollout, '`Pick List`' the road surface objects and scroll down to the `Projection` rollout;
16. Hit '`Add`' to add a `Project Mapping`; this will open another rollout when you scroll down;
17. In the `Project Mapping` rollout, set up the `Source` and `Target` channels[25];
18. In the `Projection` rollout, hit `Project` to apply the mapping of the spline-based object onto the selected target race surface's mapping channel;
19. In the Material Editor, make the RaceGroove texture stage visible to check the result;
20. Some vertices may need a manual mapping tweak to fix glitches;

[23] Height data is irrelevant in this procedure.
[24] Use an integer for the length, as this will prevent a seam at the spline endpoints.
[25] Mapping Channel 3, as this is what the RaceGroove texture stage uses.

Appendix B – Vertex Alpha troubleshooting

Be aware of a bug in 3ds Max, which I believe is STILL there since version 5 or less. If you set a Vertex Alpha value (e.g. Vertex Alpha 90.0 for facing polygons) and then alter the vertex count of an object, the Vertex Alpha values, while looking fine in 3ds Max, will look scrambled in game. You can remove or reduce vertex counts (like breaking one larger object up into two) but you CAN NOT add any vertices (i.e. by cutting in or attaching new geometry). There is nothing you can do to fix this, except:

1. Export the object out in a .3DS format. This will remove all vertex data (and your materials as well, but there's a workaround);
2. Keep a copy of the bad object, but move it out of the way;
3. Import the object you just exported back into the scene. There is an option to resize the object. I believe you check this. (If the object comes in the wrong size, you have it set wrong);
4. Select all the vertices and do a 0.001m weld on them (exporting as .3DS breaks all the vertices for each face);
5. There are scripts out there that will copy material ID's from one like object to another. Find one of these and copy the ID from the old object to the new one;
6. Apply your material to the new object;
7. Delete the old.

The other option is to save a copy of the unaltered geometry BEFORE you start changing Vertex Alpha values and go back to that if you mess up along the way and have to start over. This becomes a rather tedious problem when dealing with vertex-blended terrains. After a while, you'll become very good at making sure everything related to the terrain is done before you start blending.