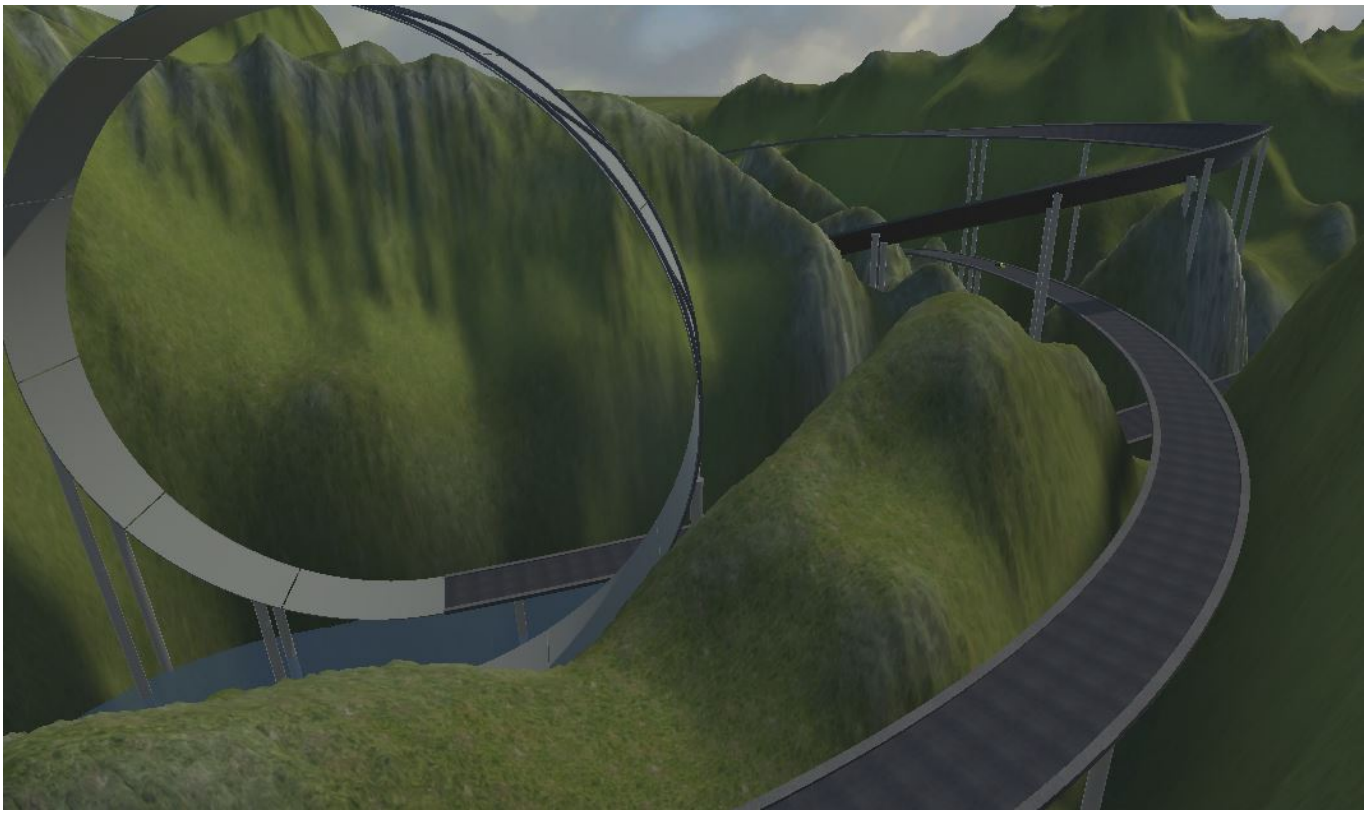
Help and support

For help or questions please email me at: tmulgrew@slingshot.co.nz

Overview

This racetrack building system allows you to build racetracks by warping meshes around a series of curves. It is particularly suited to building elevated, rollercoaster-like tracks. It features the ability to create jumps, banked corners and even loops.

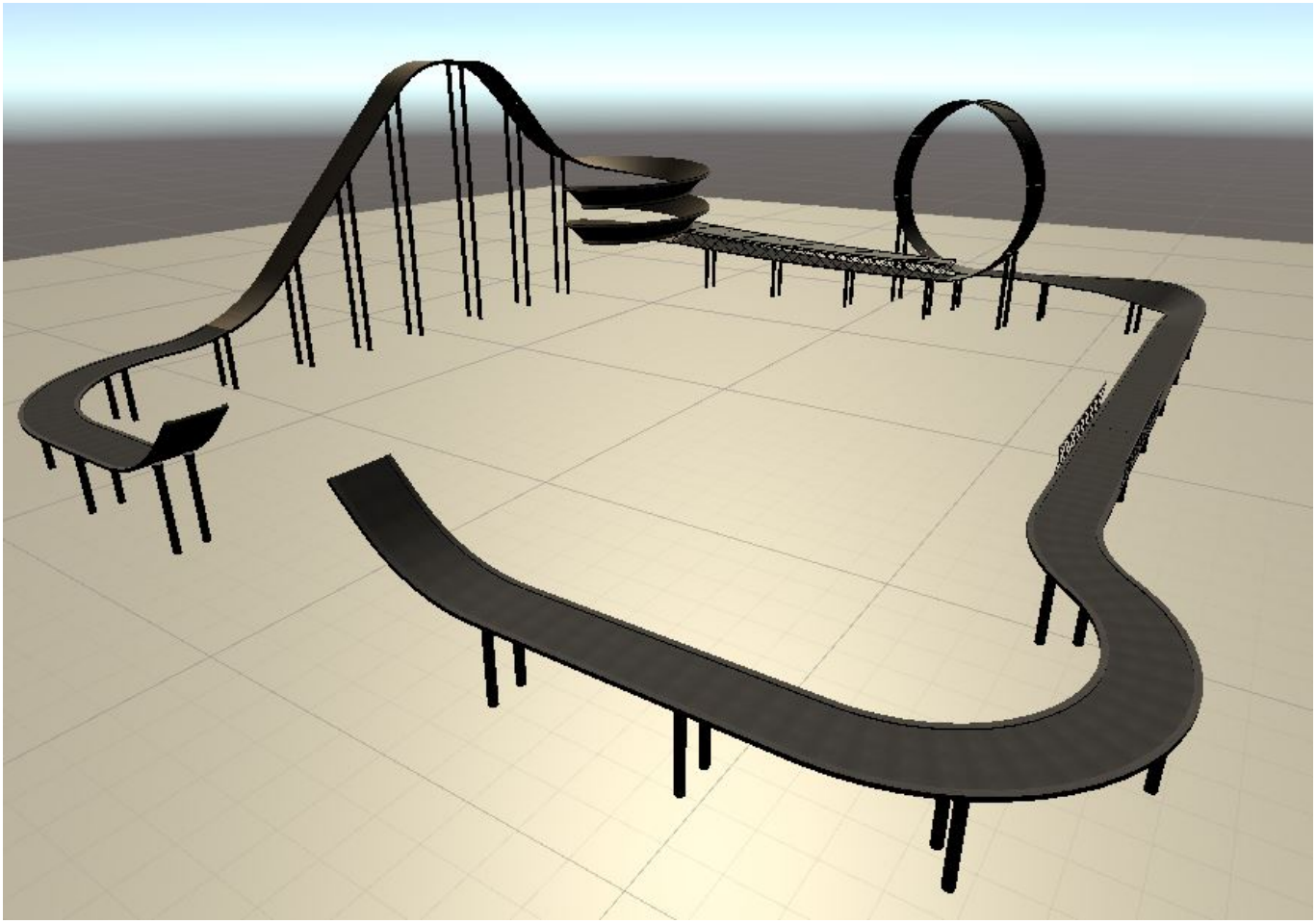


The resulting mesh is fully drivable, for example using the Unity Standard Assets car, and contains runtime information for tracking a vehicle's progress.

The package contains a "progress tracker" component that can be attached to a car to track its progress around the race track, detect when it has fallen off and automatically place it back on track after a short delay.

Example scene

See the Assets/Racetrack Builder/Example.unity scene for a simple example racetrack.



Driving on the track

You can drive on the track using the Unity Standard Assets car.

Search the Asset Store for the "standard assets" package. Import everything in the "Standard Assets/Vehicles/Car" and "Standard Assets/CrossPlatformInput" subfolders of the package. It is not necessary to import the rest of the package.

To setup the car:

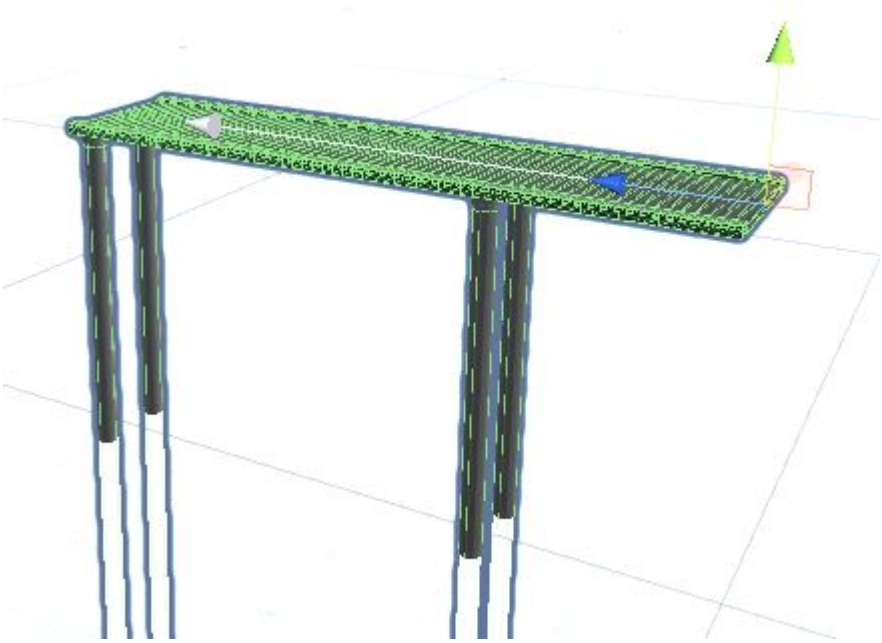
1. Drag the "Car" prefab from "Assets/Standard Assets/Vehicles/Car/Prefabs" into the scene.
2. Drag the "Main Camera" in the scene tree and make it a child object of the Car.
3. Select the "Main Camera". Reset its transform in the Inspector window.
4. Move the camera behind and above the car a little (e.g. Y=2.3, Z=-6)
5. Drag the "RacetrackProgressTracker" script from "Assets/Racetrack builder/Scripts/Runtime" onto the car.
6. Select the car and find the "Racetrack Progress Tracker (Script)" component in the Inspector window, and click "Reset car".
7. Run the scene and drive with the arrow keys. If you fall off the track, wait 10 seconds and you will respawn back onto it.

Quick start - Creating a track

It only takes a few minutes to get started building a race track:

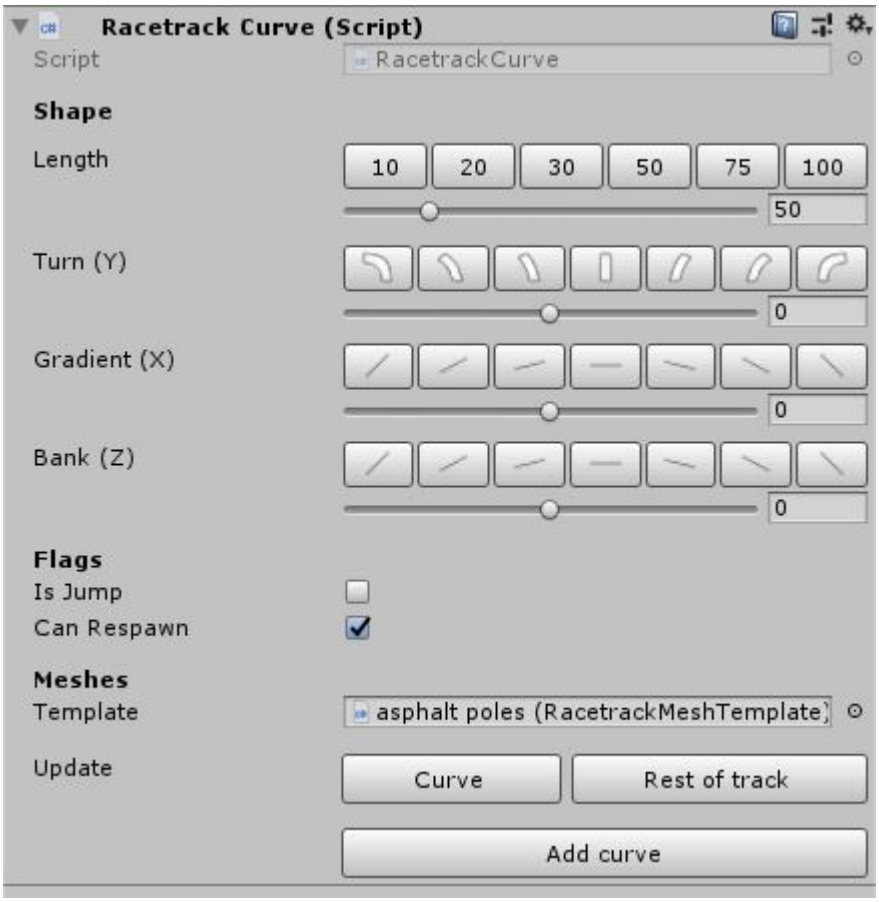
First create or open a scene (File > New Scene), and make sure it has some ground (e.g. a plane).

Now create the racetrack object by creating "GameObject > 3D Object > Racetrack" from the menu.



Racetrack Builder has created a racetrack with a single curve object. The white arrow shows the path of the curve, and the mesh has been fitted around it. The curve is initially a straight line, so the road mesh is accordingly straight.

The curve is automatically selected, and the Inspector window shows the UI for editing it:



Try clicking the buttons in the **Shape** section to see how the track adapts to the updated curve.

For finer control you can set the values via the slider or key them in. You will need to explicitly rebuild the mesh model by clicking the "Update" button in this case.

Once you're happy with the initial curve, click the "Add curve" button to create a new curve and add it onto the end of the racetrack.

This is the basic workflow for Racetrack Builder.

Note: If you need to reposition your racetrack, be sure to update the transform on the main "Racetrack" object, rather than the curves. Racetrack Builder automatically manages the transformations for the curve objects. If you get into a mess, select the main racetrack object and click "Update whole track" to reposition all the curves and rebuild the all the meshes.

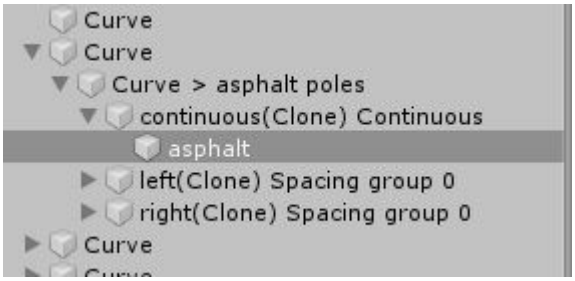
More about track building

Selecting corners

If you need to go back and change a previous corner, there are a couple of ways to do this.

You can expand the "Racetrack" object in the scene tree and click on the corresponding "Curve" child object.

Alternatively, you can click on the track in the editor view. This selects the mesh that was generated from the curve, which is placed underneath the curve in the scene tree.



Look back up the scene tree to find the corresonding Curve and select it.

Changing a previous curve has a flow on affect that affects where the rest of the curves in the track are positioned.

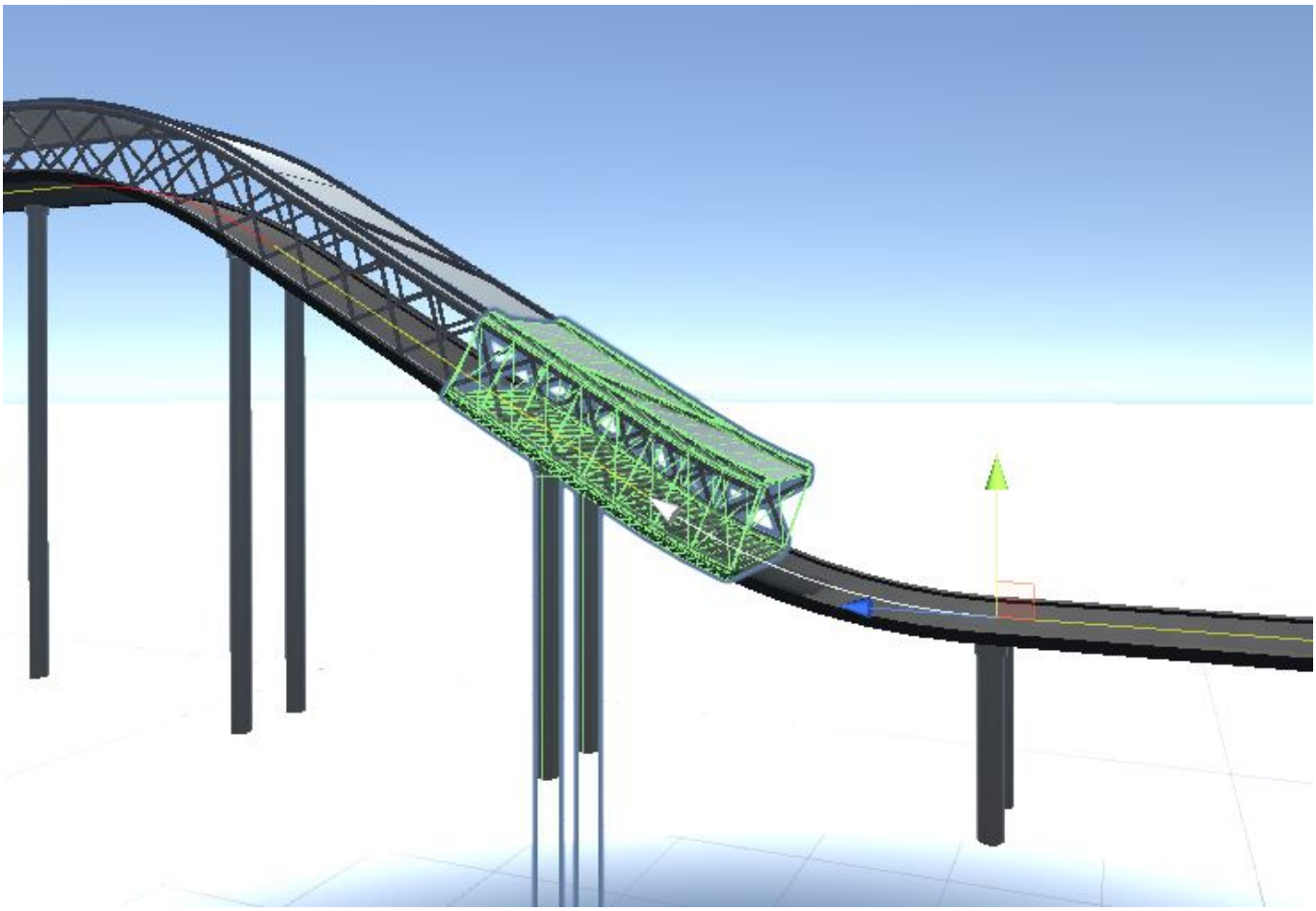
Be aware that changing a curve's length can also be quite slow, as all the meshes for the remaining curves in the track must be regenerated so that the meshes line up correctly.

Changing a curve's angles is faster, because only the curve and its immediate neighbours need to be rebuilt. (The remaining curves are simply repositioned, which is faster.)

Changing the track type

You can change the type of track by dragging a different prefab from the "Assets/Racetrack builder/Prefabs/Track Templates" project folder and dropping it on the "Template" field in the properties of the corresponding curve.

Then click "Rest of track" in the curve properties, to update the remainder of the track.

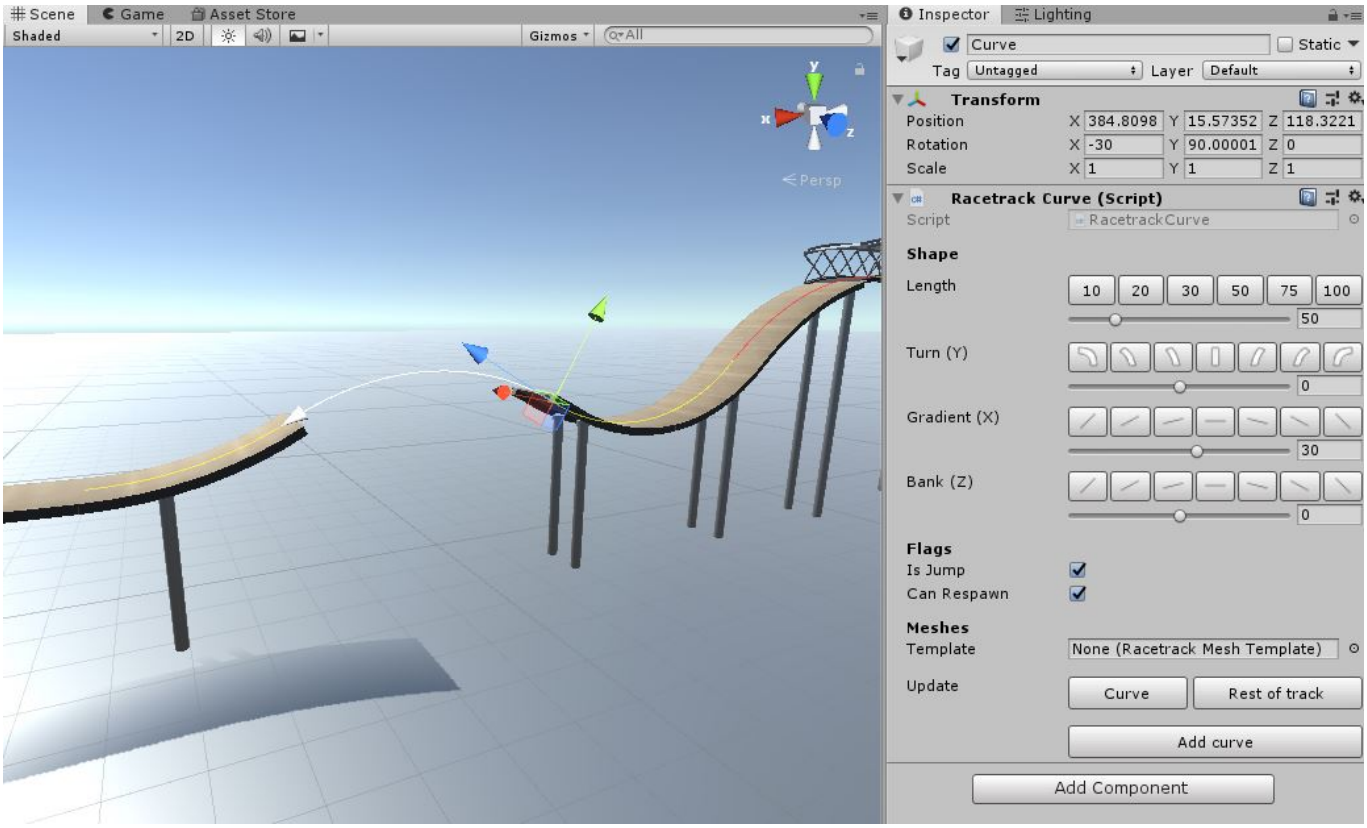


This will set the track type from this curve onward, or until it is changed again in a later curve.

Creating jumps

To create a jump, select a curve and tick "Is Jump" property. Then click "Rest of track" to update the rest of the track.

No meshes will be generated for the curve, resulting in a gap which the player must jump over.



Respawning

The RacetrackProgressTracker component can be added to the player car to automatically respawn them back on the track after they fall off. By default the player is respawned on the last curve that they drove on.

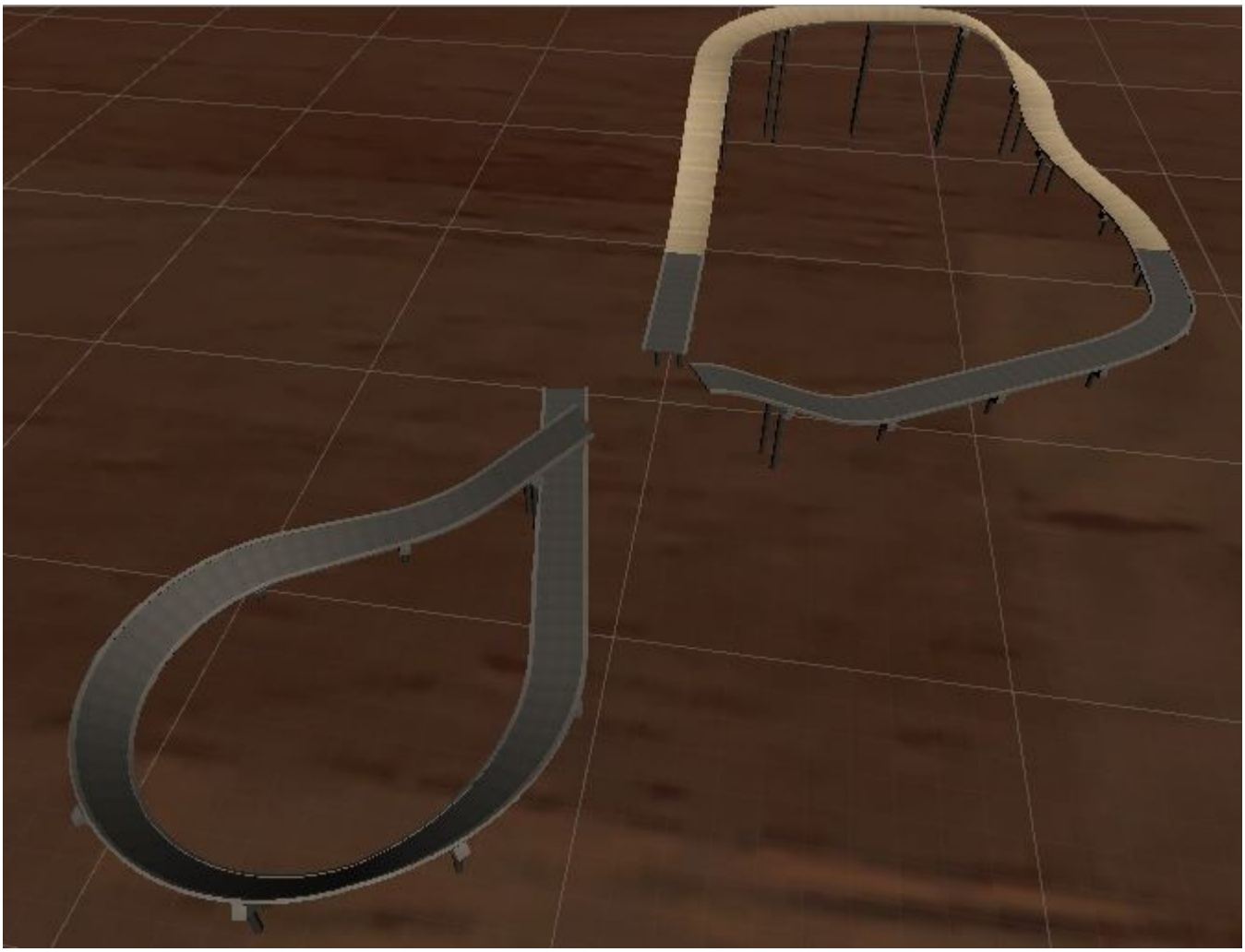
Sometimes a curve is not suitable as a respawn point, e.g. if the player cannot get enough speed to complete the next jump, and therefore cannot progress.

To avoid this, untick the "Can Respawn" property of the curve. The respawn logic will search backwards from the last curve the player drove on until it finds a curve where "Can Respawn" is ticked.

Creating a closed circuit

Racetrack Builder has an *experimental* feature to help create closed circuit racetracks. It is not perfect, but usually gets pretty close, then you can fix it up with small manual adjustments.

It works best on a racetrack needs that is close to being a closed circuit, but needs some help lining up the start and end curves.



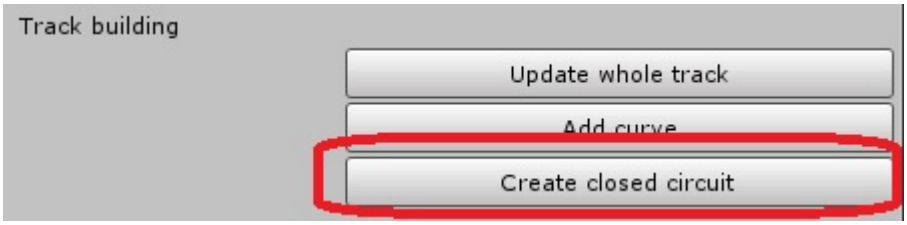
To use it you must select *exactly 3* curves that Racetrack Builder can lengthen or shorten so as to shift the racetrack so that the start and end curves line up.

The closer the curves are to being at right angles with each other, the smaller the adjustments Racetrack Builder will have to make. If two curves are pointing in the same direction or all three are on the same plane then Racetrack Builder will not be able to find a combination of length adjustments that will create a loop.

Also remember that changing the length of a curve will shift the racetrack from that curve onwards. So if you have carefully lined up your racetrack with a terrain or other objects (for example) then you may want to select three curves towards the *end* of the track to minimise this.

Mark each of the 3 curves by selecting it in the Unity scene, and ticking "Auto Adjust Length".

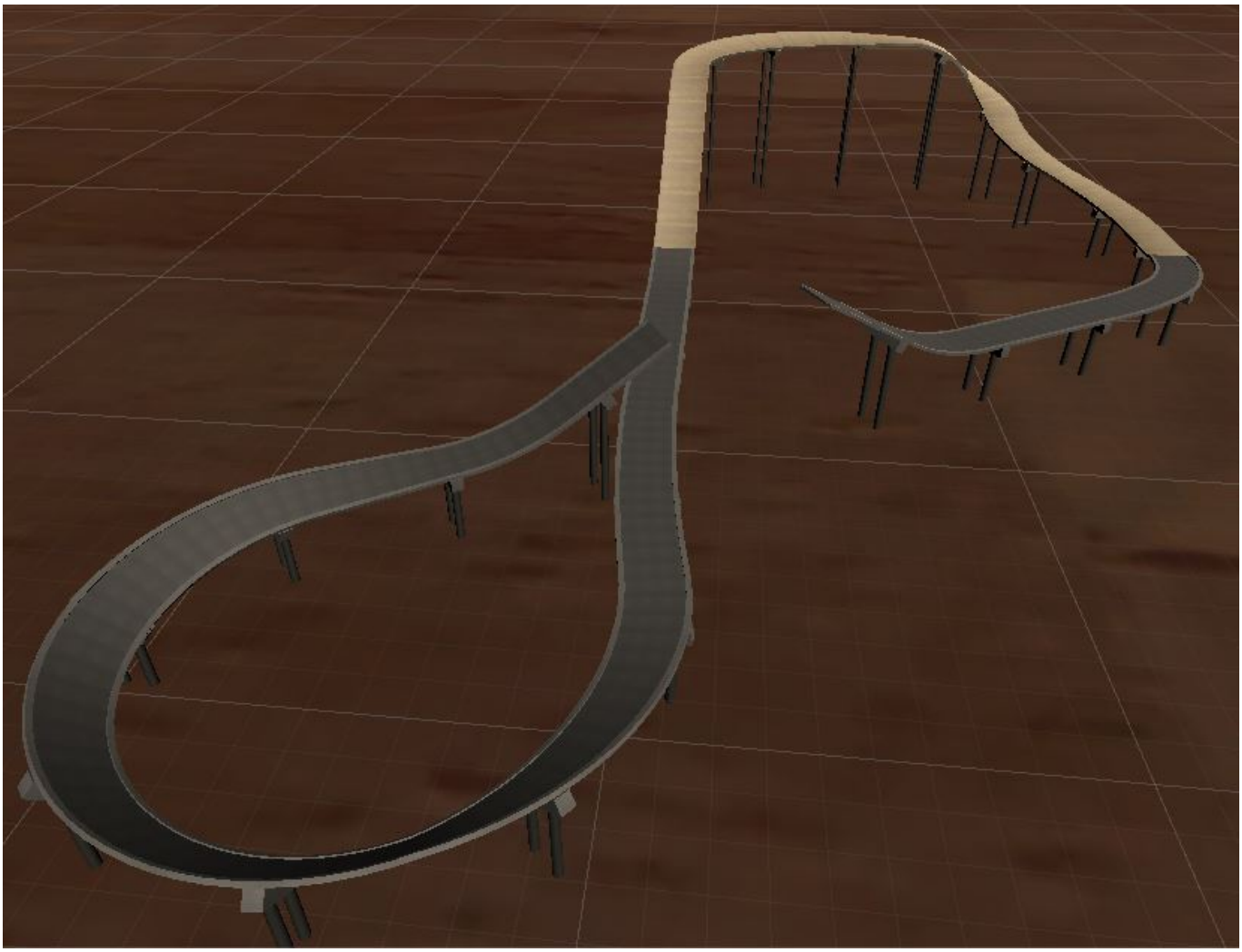
Then select the racetrack object and click "Create closed circuit".



Racetrack Builder will attempt to close the circuit as follows:

1. Adjust the angles of the last curve to line up the *direction* with the first curve
2. Adjust the lengths of the selected curves to line up the *position* of the last curve
3. Rebuild the racetrack meshes around the updated curves

This should result in a racetrack that is almost exactly a smooth, closed loop (but will likely still require some manual corrections).



In some cases Racetrack Builder will not be able to create a closed loop, such as when the start and end curves are not close enough in direction, or if there is no valid combination of length adjustments that will line up their positions. In this case you will need to select different curves to adjust and/or make some adjustments to the racetrack manually, then try again.

Creating mesh templates

A "mesh template" provides the meshes that are generated and fitted to the path of the curves, in order to create the racetrack.

They are a little bit like Unity prefabs, and are often stored as one. But they are instantiated a little differently, mainly due to the need to warp the meshes along the curves and clone the repeating parts (like support poles).

This package contains a set of mesh templates to get you started, but you can easily build your own to create your own road types.

Then select the racetrack object and click the "Create closed circuit" button.

Supplied templates

The sample mesh templates are in: Assets/Racetrack builder/Prefabs/Track Templates

There are also prefabs that can be composed together to create mesh templates in: Assets/Racetrack builder/Prefabs/Template parts

These are grouped into folders:

- Driving surface - The main part the player drives on
- Sides - Barriers etc that attach to the side of the track
- Supports - Support poles that hold up the track

Example walkthrough

As an example, we can can create a mesh template consisting of wooden planks, support poles and side barriers.

First navigate to the "Assets/Racetrack builder/Prefabs/Track Templates" project folder and drag the "template base" prefab into your scene. This creates the basic skeleton structure for a mesh template. You can examine the object in the scene tree:

- The main object ("template base") contains a **Racetrack Mesh Template** script component. This marks it as a mesh template and is mandatory.

- Underneath we have a "continuous" object with a **Racetrack Continuous** script. This denotes the meshes underneath it as "continuous", meaning they will be warped along the racetrack curves. In this prefab the driving surface and barrier meshes will be "continuous", so that they follow the racetrack curves.

- We also have a "spaced" object, which is an empty placeholder for now. This will be used for objects that will be placed along the track at regular intervals, like the supporting poles.

Create the road surface as follows:

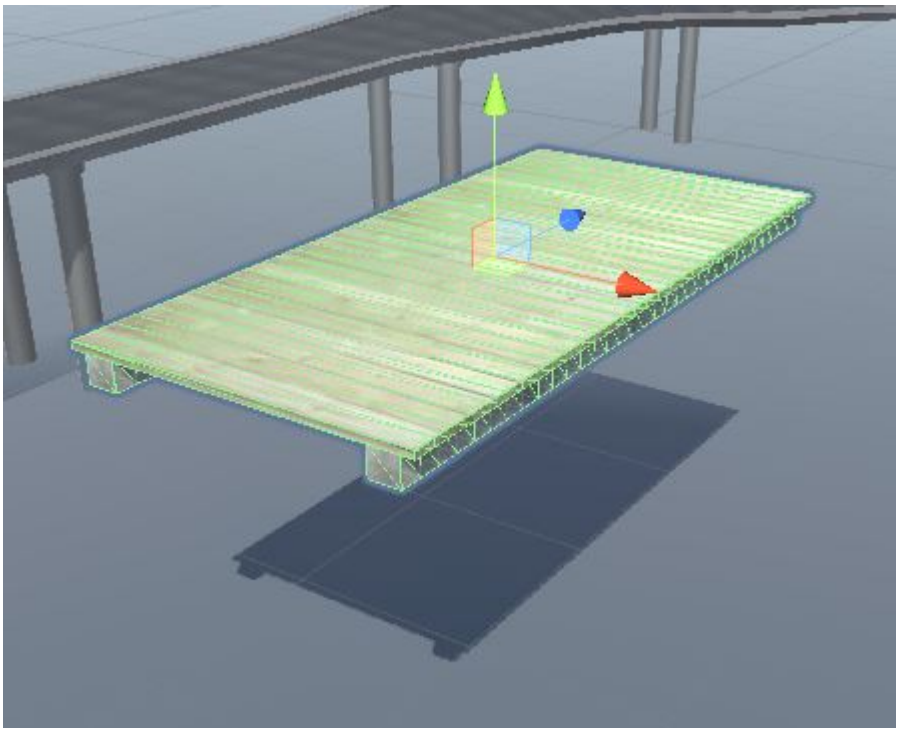
1. Make sure the "template base" object is expanded in the scene tree, and the "continuous" child object is visible.

2. Open the "Assets/Racetrack builder/Prefabs/Template parts/Driving surface" project folder.
3. Drag the "woodplanks" prefab into the *scene tree* and drop it onto the "continuous" child object.
4. Reset the "Position" of the new object to (0,0,0) in the Inspector window.

Placing the woodplanks mesh underneath the "continuous" object means it will be warped along the racetrack curves. The first continuous mesh is also special, as it will be treated as the main driving surface. This has certain implications:

- It defines the length of the mesh template.
- It means a "Racetrack Surface" script component will be attached to the mesh after it has been copied and warped. This is used by the RacetrackProgressTracker component to detect when the player is above the road. It also links back to the curve that generated it, so that the player's progress along the track can be calculated.

If you view the template object in the scene, you'll see it now has a woodplanks surface. This is now a fully functional mesh template, which you can drag from the scene tree and drop onto the "Template" property of a racetrack curve. But we are not finished yet.



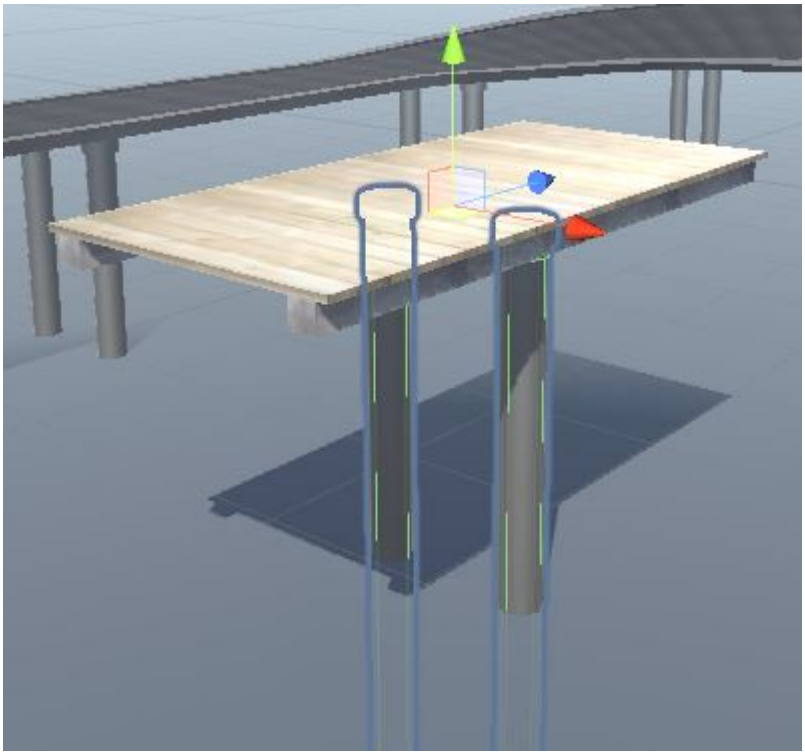
You may notice the template looks a little small. This is because the default mesh templates are all scaled by a factor of 3. Correct this as follows:

1. Click on the "template base" object in the scene tree.
2. Set the X, Y and Z components of the "Scale" to 3 in the Inspector window.

Now add some poles to hold up the track:

1. Make sure "template base" is still expanded in the scene tree, and that "spaced" is visible.
2. Open the "Assets/Racetrack builder/Prefabs/Template parts/Supports" project folder.
3. Drag the "Metal poles" prefab into the *scene tree* and drop it on the "spaced" child object.
4. Set the Y position to -0.33 in the Inspector window.

The mesh template now has two poles underneath the track. Once again it is fully functional. If you assign it to a curve and regenerate the track, the poles will be spaced along it at regular intervals.



Expand the new "Metal poles" object in the scene tree and examine it.

The "Metal poles" object has a **Racetrack Spacing Group** script component. This indicates that the content underneath will be spaced evenly along the track, and specifies the spacing.

The "Index" property is important to get correct spacing. Objects spaced at *different* intervals should be assigned to spacing groups with *different* indices, in order to get correct results. You should organise your spaced objects into distinct groups, each with a unique index between 0 and 15. The supplied mesh templates use index 0 for road support poles and index 1 for poles that hold up the side barriers.

The other properties are:

- Spacing Before - Amount of space to add before an object
- Spacing After - Amount of space to add after an object

In this case we have 10 units before and after each, meaning the poles will be spaced 20 units apart.

Underneath the "Metal poles" object we have two child objects, "left" and "right". Each has a **Racetrack Spaced** script component, indicating it has content to be repeated, with properties describing how:

- Is Vertical - This forces the object to be aligned vertically in world space. If unticked the object will be aligned with surface of the curves.
- Max Z angle - Maximum bank angle (positive or negative). If the curve exceeds this angle, the spaced content will not be created.
- Max X angle - Maximum pitch angle (positive or negative). Same as above.

Important: "Max Z angle" and "Max X angle"p[] only apply when "Is Vertical" is ticked.

In this case "Is Vertical" is ticked, so that the poles remain vertical regardless of how the road surface pitches or banks. The max X and Z angles ensure poles are not created if the track is upside down. (See the loop in the example scene for an example of upside-down track).

Underneath the "left" and "right" objects is the content to be generated. In this case we have a standard Unity mesh to represent the pole visually, and a capsule collider.

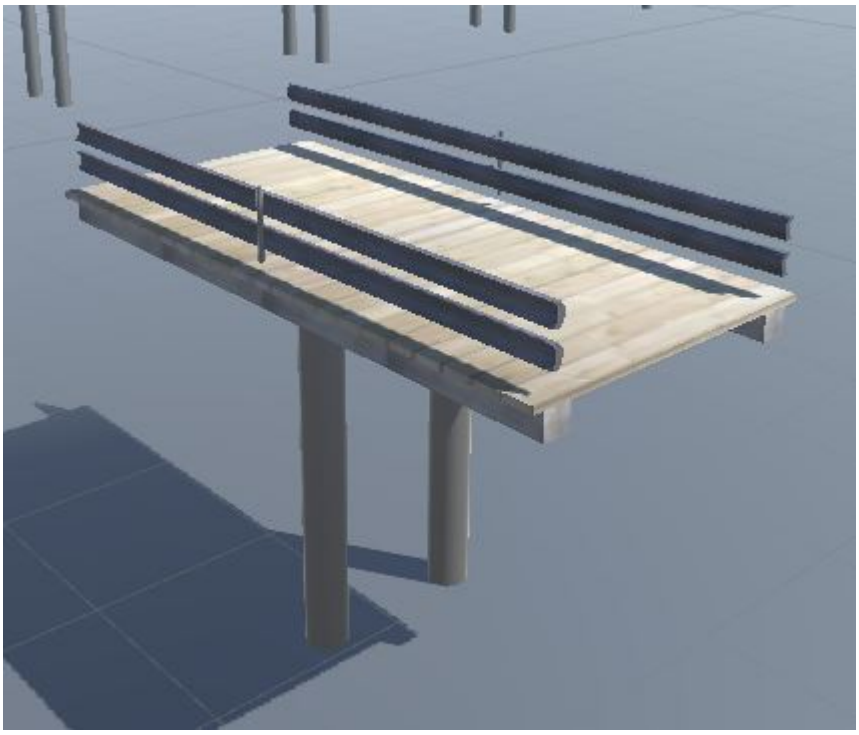
To complete the mesh template, add the side barriers as follows:

1. Open the "Assets/Racetrack builder/Prefabs/Template parts/Sides" project folder.
2. Drag the "rail barrier" prefab and drop it on the "template base" object in the scene tree. **Do not** drop it on the "continuous" or "spaced" child objects.

The prefab should not be a child of the "spaced" or "continuous" objects, because it contains both spaced and continuous content. If you expand the "rail barrier" object, you'll see it has its own "continuous" and "spaced" sub objects with the appropriate components attached to each. The "continuous" child object contains two railing meshes, which will be warped along the curve path. The "spaced" child object contains a small pole configured to be spaced every 6 units along the track.

The barrier runs along one side of the mesh template. To create the barrier on the other side:

1. Right click the "rail barrier" object in the scene tree and select "Duplicate"
2. In the Inspector window, set the rotation Y component to 180.

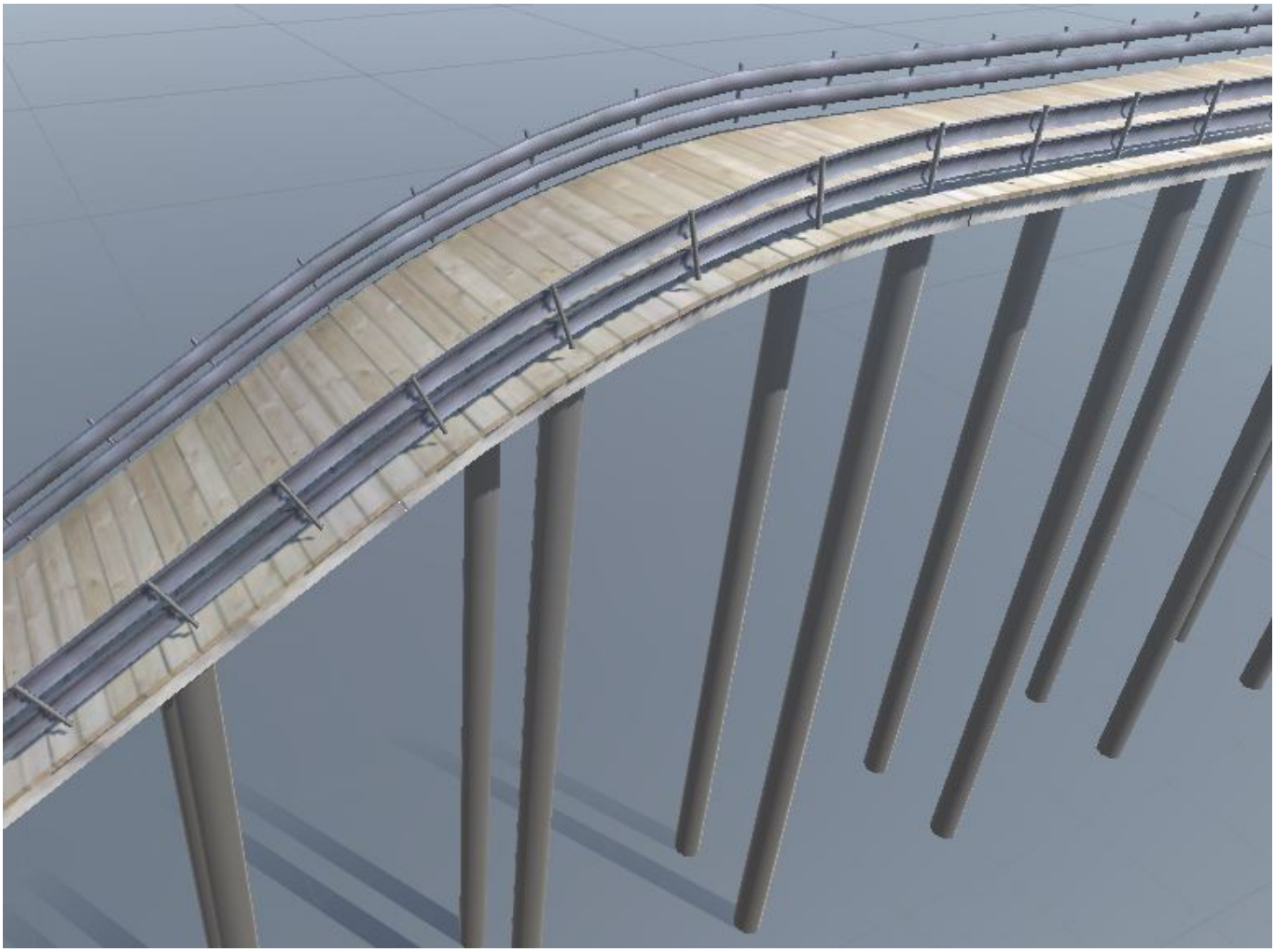


The mesh template is ready to use. The last step is to convert it into a prefab, so that it can be reused easily:

1. Click on the "template base" object in the scene tree.
2. Key in a new name in the Inspector window. E.g. "woodplank barriers"
3. Navigate to the "Assets/Racetrack builder/Prefabs/Track Templates" project folder.
4. Drag the new object from the scene tree into the project window to create a new template.
5. When prompted click "Prefab Variant" (although it doesn't make much difference in this case).

Once the prefab has been created, you can delete the object from your scene.

The template is now complete. Drag it from the project folder onto the "Template" field of a racetrack curve and click "Rest of track" to apply it to your track.



Creating meshes

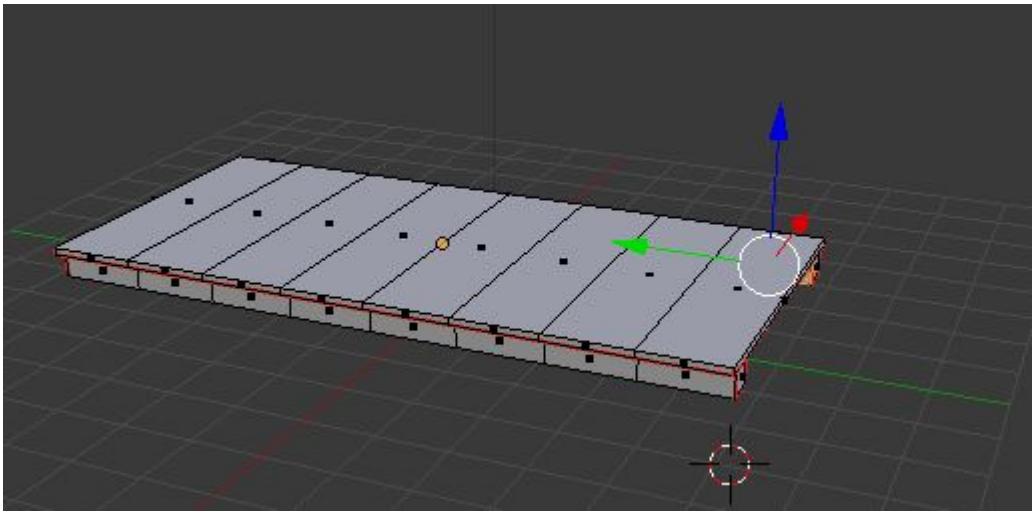
You can of course use your own meshes to create track templates and build your own custom tracks.

The standard Unity workflow applies. I.e. create them in a 3rd party modeller, import them, and use them to build your mesh template prefabs.

For continuous meshes, like the road surface, there are some guidelines to ensure they will warp around the racetrack curves correctly, and provide a smooth driving experience.

Split along the Z axis

Meshes are warped to fit a curve by transforming their existing vertices. The process does not split existing polygons or introduce new vertices, so you must provide sufficient vertices to make the result look smooth.



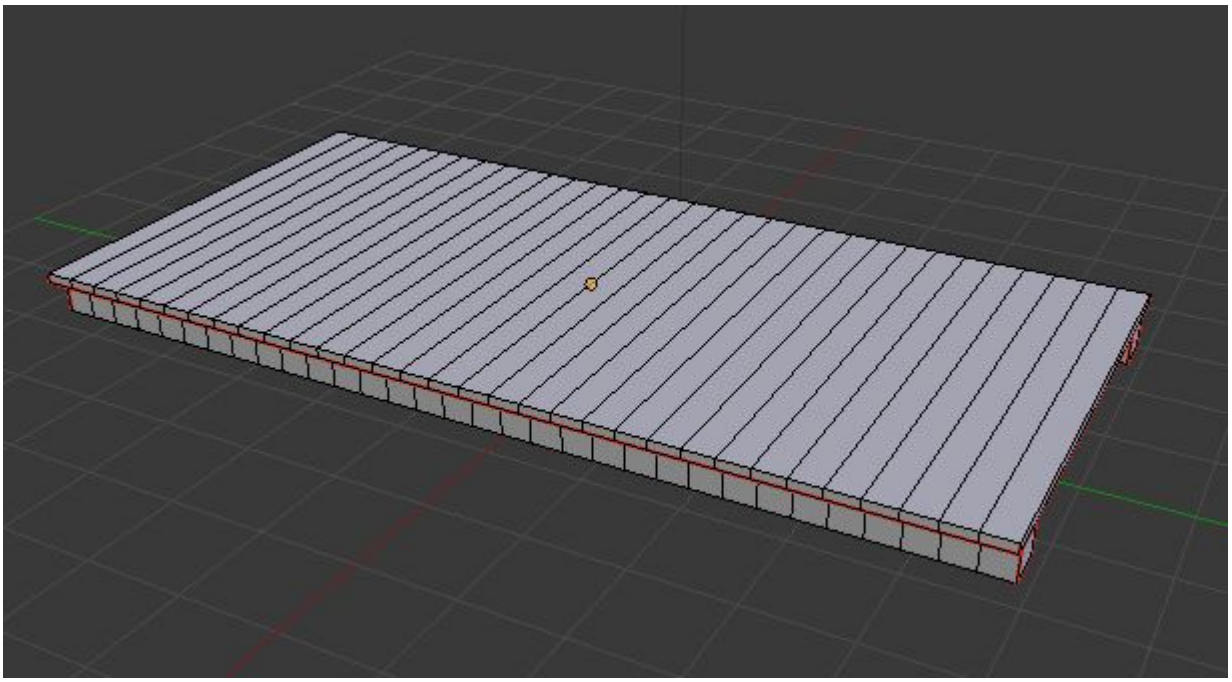
Here the road surface mesh has been sliced into 8 pieces, at regular intervals along the Z axis. (In this case the Blender3D "Loop cut" tool was used. Other 3D modelers should have a similar function.)

This is the visual mesh used for the "woodplank" road surface.

Separate high poly mesh for collisions

The visual mesh *looks* decent when warped around the racetrack curves, but does not have adequate detail to be used as the collision model. It will feel juddery to drive over when the road pitches or banks.

To fix this, create a second mesh, identical to the first, but with more polygons and vertices, so that it can be warped into a smoother surface.



Assign this mesh explicitly to the "Mesh Collider" component of your mesh object or prefab.

Unlike the rendered mesh, having a high resolution collision mesh has little performance penalty. Although it is worth keeping in mind the storage requirements of the mesh, especially as it will be duplicated many times along the different curves of the racetrack.

In this example the mesh has been sliced into 32 pieces, which gives a decent driving feel without being too excessive.

*Note: For continuous meshes that the player will **not** be driving on, like barriers or walls, it is not necessary to have a separate high poly mesh*

Use a mesh collider

For continuous meshes the collision mesh must be warped along the racetrack curves along with the visible mesh. Although other colliders, like the box collider, may fit neatly around the mesh template, they cannot be warped, which will result in an incorrect collision model on your racetrack.

Therefore always use a mesh collider with continuous meshes.

Align the top of drivable surfaces with the Y=0 plane

This is more of a suggestion than a rule, however it makes life a lot easier.

It means a continuous surface when the track switches from one mesh template to another. Also, applying a different scale factor to a mesh template will not alter the height of the surface, in case you want to vary the size of the track.

Spaced objects

Spaced objects are much simpler than continuous meshes. There is no mesh warping. They are simply instantiated (using standard Unity object instantiation) and positioned at regular intervals along the track.

This means:

- You do not need to add extra vertices.
- You do not restricted to a mesh collider. For example, you can use a box or capsule collider if suitable.

Appendix - Component reference

Track building components

Location: *Assets/Racetrack Builder/Scripts/Track*

Use these components to define the racetrack curves.

Racetrack

Represents the racetrack as a whole. Acts as the parent object for the Racetrack Curve objects that define the curves of the track.

Use "GameObject > 3D Object > Racetrack" to create a Racetrack object.

Properties

- Segment Length - Length of line segment generated from curves. Each curve is converted into a sequence of small line segments. Generally you should not need to change this setting.
- Respawn Height - Height of the respawn points above the racetrack surface.

Buttons

- Update whole track - Delete and recreate all generated meshes for all curves. This can take several seconds for larger racetracks.
- Add curve - Add a new curve to the end of the track and select it.
- Create closed circuit - Attempts to adjust the curves to create a closed circuit. Refer to the **Creating a closed circuit** section earlier in this document.
- Delete meshes - Delete all generated meshes.
- Clear templates - Clear the track templates from all curves (sets them to null).

Racetrack Curve

Represents a single curve in the racetrack. Stores the shape of the curve.

Properties

- Length - The curve length.
- Turn (Y) - How far the curve turns around the Y axis. Positive values create right hand corners.
- Gradient (X) - The X axis angle at the *end* of the curve. The curve curves from the previous gradient to the new one.
- Bank (Z) - The Z axis angle at the *end* of the curve. The curve curves from the previous bank angle to the new one.
- Is Jump - If true, no meshes will be created for the curve, leaving a gap in the racetrack.
- Can Respawn - If true, the Racetrack Progress Tracker component will respawn the car above this curve. Otherwise it will skip it and search backwards along the racetrack for a curve with "Can Respawn" set to true. Useful to ensure there is enough run-up before jumps etc.
- Template - The Racetrack Mesh Template that defines the meshes that will be warped along the Racetrack Curve objects to create the final model. Can be left to null to use the previous curve's template.

- Auto Adjust Length - Indicates that the curves length can be adjusted when creating a closed circuit. Refer to the **Creating a closed circuit** section earlier in this document.

Buttons

- Length (buttons) - Set the curve to a specific length. Also causes the meshes for the remainder of the track to be rebuilt, as if "Update Rest of track" was clicked.
- Turn (Y), Gradient (X) and Bank (Z) (buttons) - Set the curve angles to specific values. Causes the meshes for the curve to be rebuilt, as if "Update Curve" was clicked.
- Update Curve - Delete and recreate the meshes for this curve. Use this after changing the curve angle via the sliders (or keying in values). Note: The previous and next curve will be rebuilt as well, as their meshes can be affected by the curve's angles.
- Update Rest of track - Delete and recreate the meshes for this curve and all curves after it. Use this after changing the mesh template or "Is Jump" flag, or after changing the length via the slider (or keying in values), as all these actions can affect the meshes for the remaining curves of the track.

Mesh building components

Location: *Assets/Racetrack Builder/Scripts/Template*

Use these components to build the mesh templates that are used to create the 3D model of the racetrack.

Racetrack Mesh Template

Marks an object as a mesh template. This component is required in order to be assigned to a Racetrack Curve object's "Template" field.

No properties.

Racetrack Continuous

Indicates that all the meshes in the subtree are "continuous". The track building algorithm will copy the subtree, find all its meshes, and warp them around the racetrack curves.

Used for the road and other surfaces that follow the racetrack curvature, like barriers, tunnel walls etc.

IMPORTANT: The first "continuous" object containing a mesh defines the racetrack "surface". This is the mesh above which the car must be to be considered on the racetrack, and affects how the Racetrack Progress Tracker detects the car's progress along the racetrack and whether they have fallen off. ("First" in this case is the first applicable object based on a depth first search of the mesh template.)

A mesh template typically will have at least one of these.

No properties.

Racetrack Spacing Group

Assigns all Racetrack Spaced objects in the subtree to a spacing group, and specifies their spacing.

Used to space out repeating objects like support poles at regular intervals. Spaced objects are positioned along the racetrack curves, but their meshes are not warped to the curve.

Properties

- Index - The spacing group index. The mesh building algorithm maintains 16 "spacing groups", indices 0 to 15, which step down the racetrack spacing out objects. If a mesh template has different objects with different spacing, they should each be assigned to a spacing group with a different index.
- Spacing Before - Space to apply before an object
- Spacing After - Space to apply after an object

Racetrack Spaced

Indicates the subtree is a "spaced" object that will be copied and placed along the racetrack at regular intervals.

The actual spacing must be specified in a Racetrack Spacing Group object higher up in the object hierarchy.

Properties

- Is Vertical - True to align the object's Y axis with the world Y axis. False to align it to the curve's local Y axis.
- Max Z Angle - **Applies only when Is Vertical=true**. Specifies the maximum posive or negative Z axis angle (or "bank" angle) of the curves where an object will be placed. If the curve exceeds this angle, no spaced object will be created.
- Max X Angle - **Applies only when Is Vertical=true**. Specifies the maximum posive or negative X axis angle (or "pitch" angle) of the curves where an object will be placed. If the curve exceeds this angle, no spaced object will be created.

Note: Max Z and X Angles are useful to prevent road support poles from being created for upside down portions of the racetrack, or when the track is very steep.

Runtime components

Location: *Assets/Racetrack Builder/Scripts/Runtime*

Racetrack Progress Tracker

This component tracks a vehicle's progress along the racetrack, detects when it has come off the track and places it back on.

It must be added to an object with a "Rigid Body" component (typically a "Car" game object).

Properties

- Curve Search Ahead - How many curves to check to determine whether the car is on the track and whether it has progressed to the next curve. Must be at least 2 - one to check if the car is still on the current curve, and one to check if it has reached the next curve. May need to be higher if your racetrack has short curves, as they sometimes do not generate any meshes, so the tracker cannot detect the car driving over them. This value determines the number of ray casts per physics update.
- Off Road Timeout - The number of seconds of the car being off the racetrack before it will be automatically reset onto the track. The car is considered "on" the racetrack if it is above the racetrack surface mesh. This includes when the car is airborne - it does not have to actually be touching the road surface. Also "above" means above in the racetrack curve's local space (which can even be *below* in

world space, if the track is upside down). **IMPORTANT: If a car is jumping over an "Is Jump" curve, they will not register as being on the racetrack. If you have large jumps with a lot of airtime, make sure to set the Off Road Timeout long enough that the car can complete the jump!**
- Auto Reset - Enables the automatic reset logic.
- Current Curve - The index of the curve the car has progressed to.
- Lap Count - Increased after each lap for circuit racetracks.
- Off Road Timer - The amount of time the car has been off the racetrack. Once this value reaches the *Off Road Timeout* the car will be reset on the track. **Valid if AutoReset=true**. Otherwise will always be 0.
- Last Lap Time - The last lap time in seconds.
- Best Lap Time - The best (i.e. quickest) lap time in seconds.
- Current Lap Time - The current lap time in seconds.

Buttons

- Reset car - Place the car back on the racetrack, on the *Current Curve* (or before, if *Can Respawn*=false for that curve.)

Generated components

These components are automatically attached to the objects that are created when building the racetrack meshes.

Note: These objects should be considered to be automatically "generated". You should not modify them directly, as they are often destroyed and regenerated, meaning your changes will be lost.

Racetrack Template Copy

Indicates that the object was created by copying (and modifying) a Racetrack Mesh Template.

These objects are created when the racetrack is built, and placed underneath their corresponding Racetrack Curve objects in the scene hierarchy.

Properties

- Template - Points back to the Racetrack Mesh Template object from which this object was created.

Racetrack Surface

Attached to the copy of the first continuous mesh in the Racetrack Mesh Template to indicate that this is the main driving surface.

The Racetrack Progress Tracker looks for this component to determine if the car is above the racetrack and which curve.

Properties

- Start Curve Index - The index of the first curve the surface belongs to.
- End Curve Index - The index of the last curve the surface belongs to. Meshes can span two curves if their lengths don't line up with the curve lengths (or even more if there are curves that are shorter than the meshes).