ROAD CONSTRUCTOR

Support **Contact** **More Tools**

UNLOCKED DISCOUNTS

		
Railway Constructor	**Animated Construction Site**	**Road Elements Collection**

SUMMARY

INSTALLATION

Requirements

- Minimum Unity Version 2022 LTS +

Dependencies

- unity.burst
- unity.collections
- unity.splines

Installation

If you have other tools from Pampel Games installed in your project, it is recommended that you update them to the latest version before importing the asset.

After importing, the 'PampelGames' folder can be moved to a different location within the 'Assets/' folder if desired.

QUICK START

If you want to start building roads immediately without reading further details, here's how you can do that.

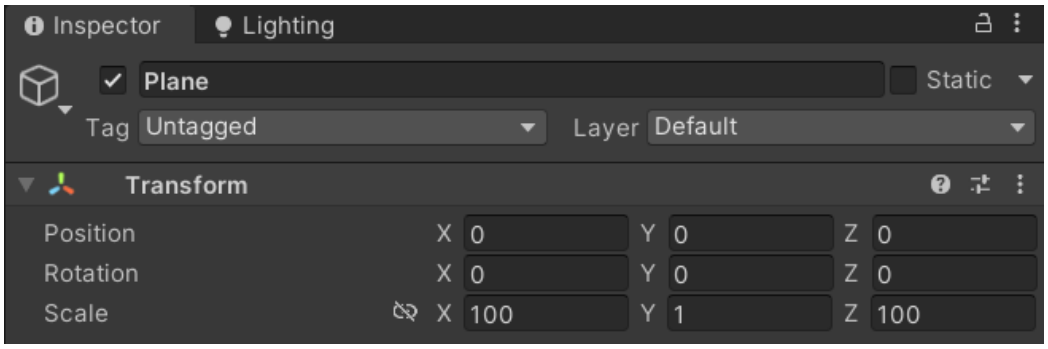
However, it is recommended to at least take a quick look at the How It Works section, as a basic understanding of how the tool functions can help you achieve your goals more quickly.

Start

First, navigate to the 'PampelGames/RoadConstructor/Demo' folder, where you will find the prefabs you can use to start building.

Road Constructor allows you to build roads either in the editor or during gameplay. Depending on your use case, you'll need either the 'Road Builder Editor' or 'Road Builder Player' prefab, which can be dragged and dropped into your scene.

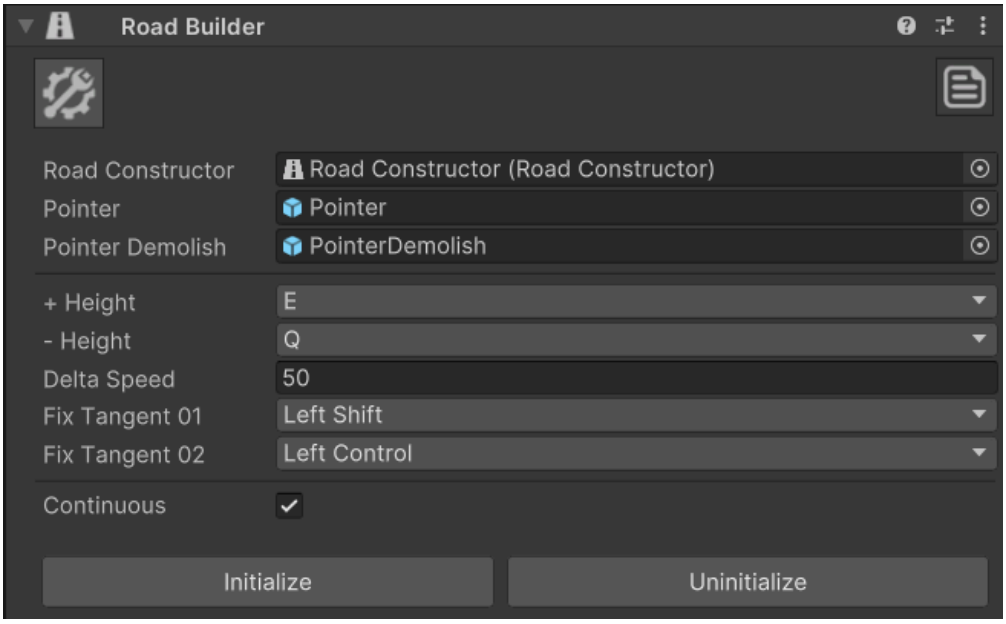
Either way, you'll need a ground to place the objects on as well. If you don't already have a mesh or terrain to use, create a plane in your hierarchy and set its scale to 100.



Editor

Important: To use the Road Builder Editor, you must enable Gizmos in the scene view.

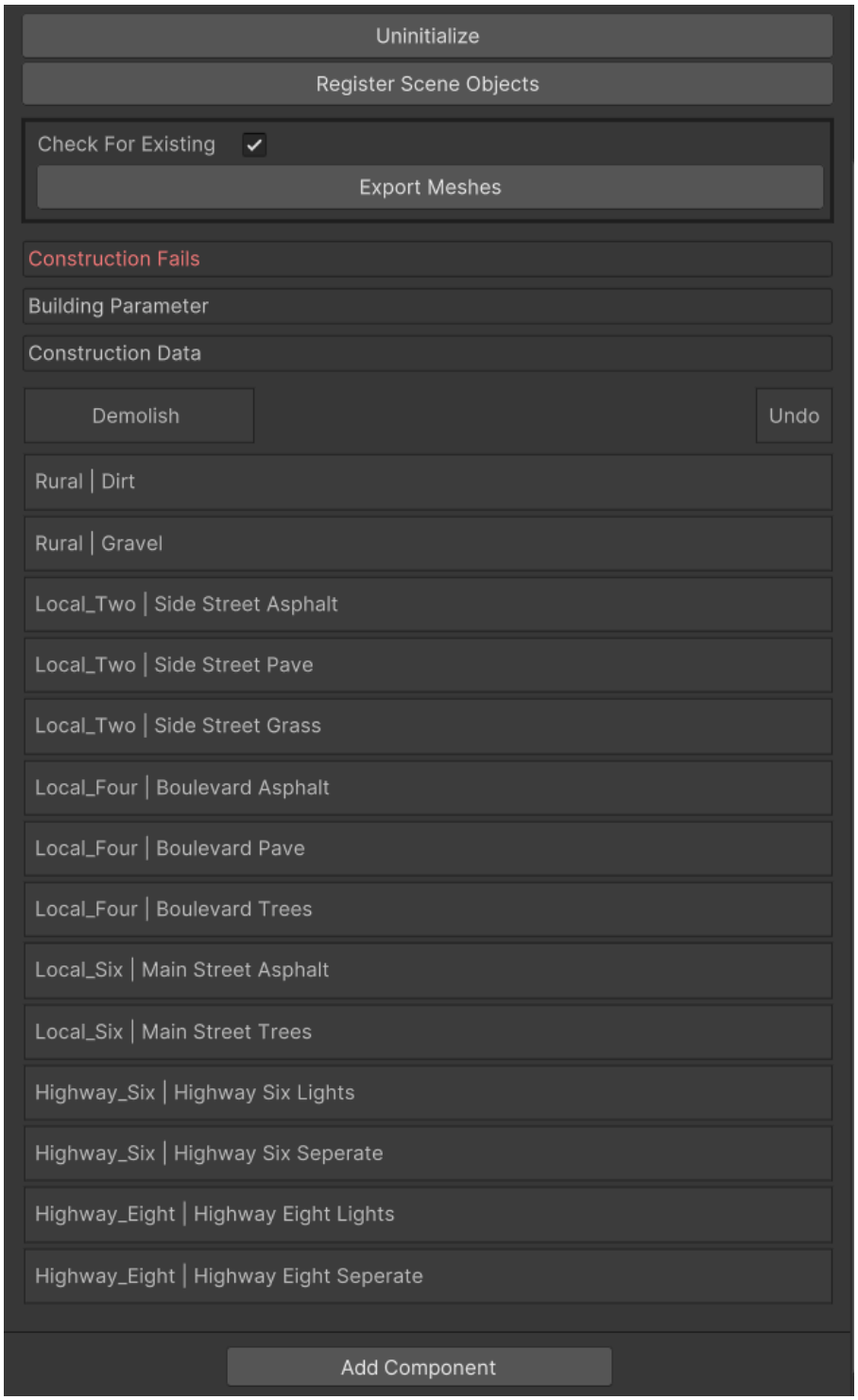
Drag the ‘Road Builder Editor’ prefab into your scene and select the parent GameObject.



To show or hide the builder settings, click the construction icon at the top left of the component.

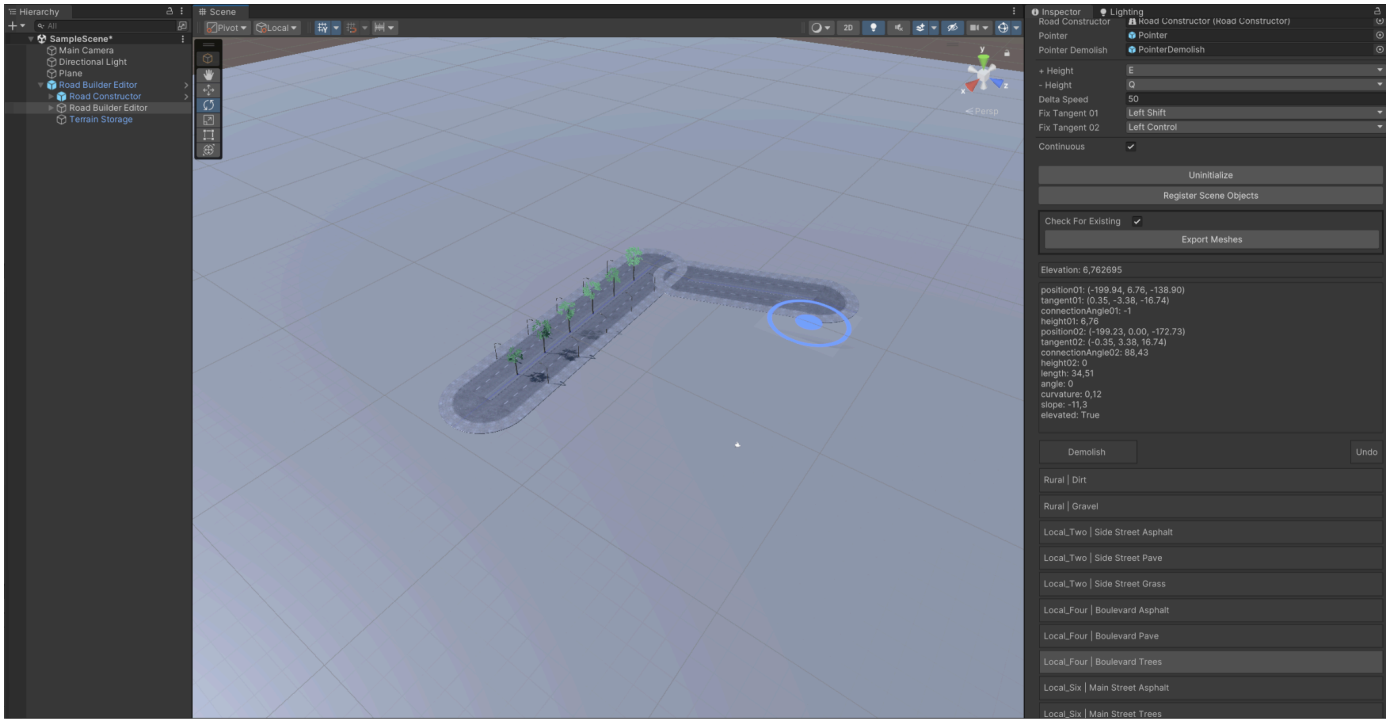
Next, click on the ‘Initialize’ button.

The component should look something like this:



Now you can select the different roads and place them onto your ground, demolish them, or undo the last operations using the dedicated buttons.

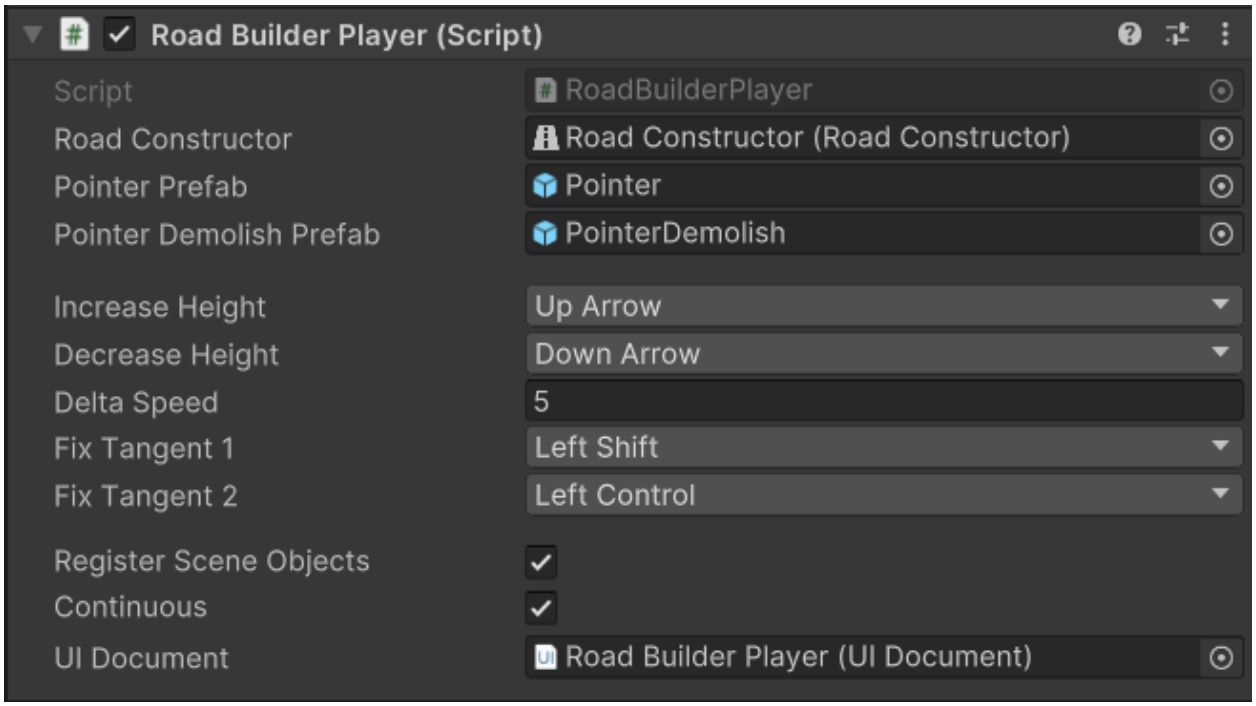
Feel free to experiment with the builder settings as well as the component settings on the ‘Road Constructor’ child GameObject; but more on that in the next chapter.



Player

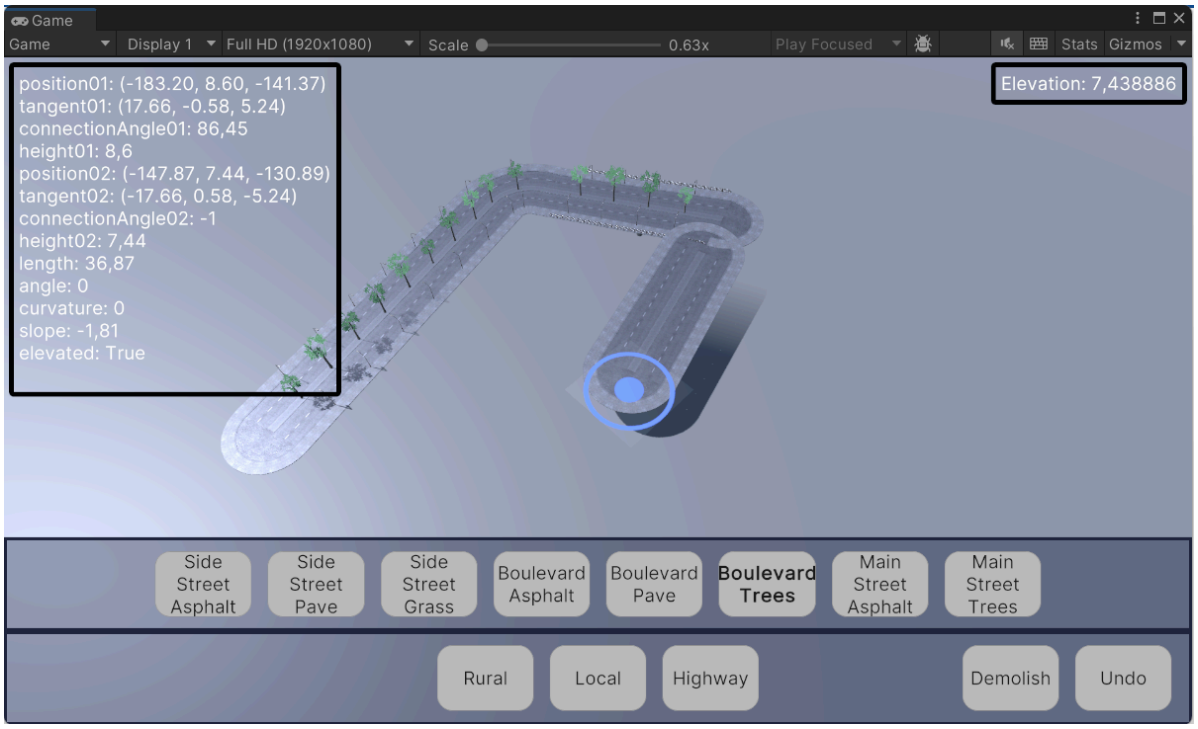
Drag the ‘Road Builder Player’ prefab into your scene and select the parent GameObject.

Here, you can find the current hotkeys for the player.



Next, make sure the Main Camera has its view focused on the ground from a distance of about 100 units.

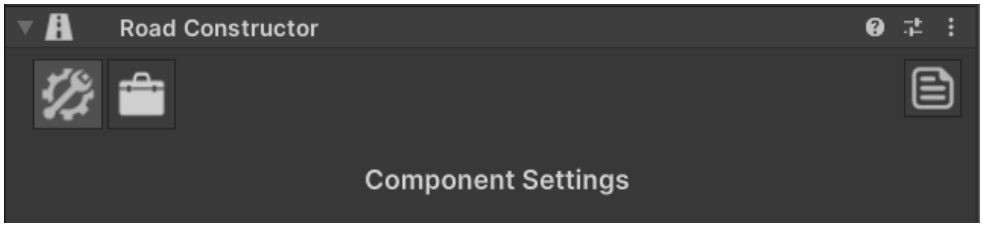
Hit play, and you can immediately start building roads using the menu panel.



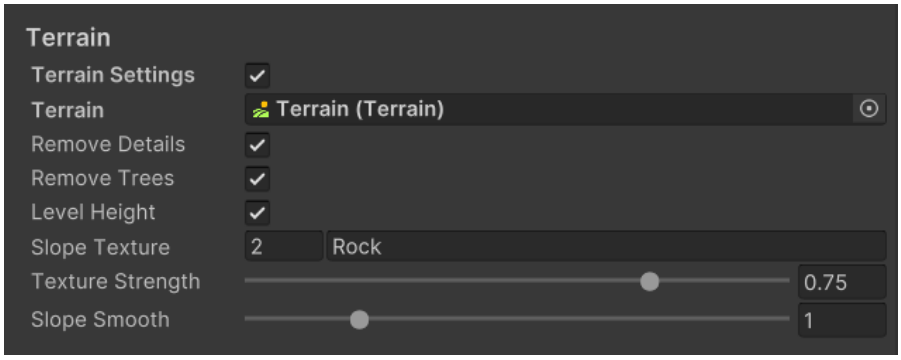
Terrain

Whether you are working in the player or editor, if you have ground terrain on which you want to build, you can apply helpful terrain settings.

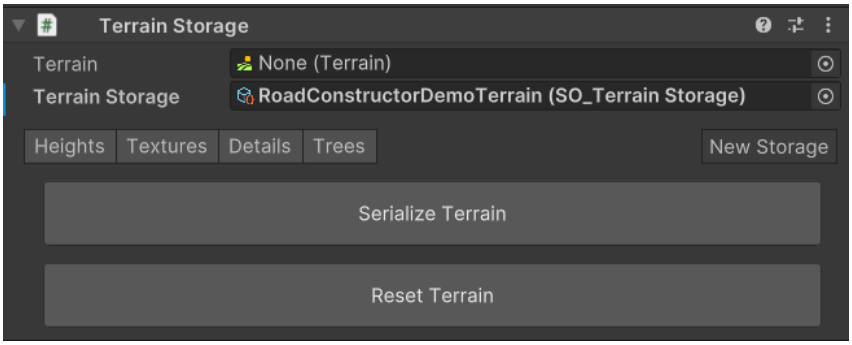
Select the ‘Road Constructor’ GameObject in the hierarchy and make sure you have the settings tab enabled.



Locate and enable the terrain settings, reference your terrains and then toggle the terrain sub-settings as desired.



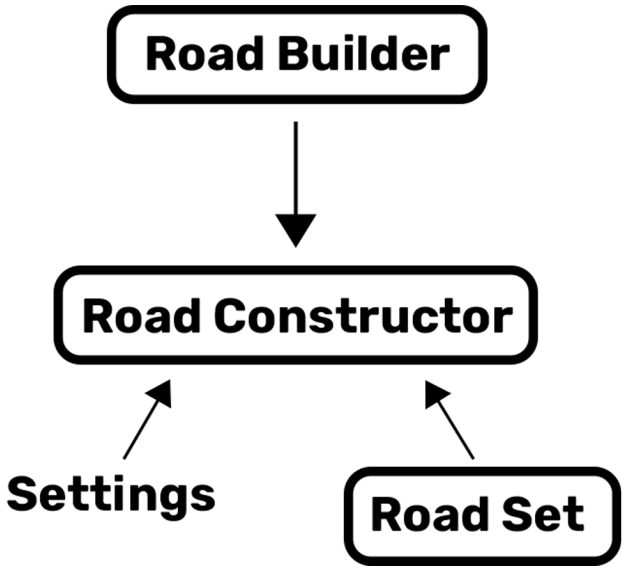
Remember that a Unity terrain is an asset, meaning that all modifications made to it are permanent.
You may find the 'Terrain Storage' component helpful, as it allows you to serialize and reset the terrain data dynamically. You can find it in the hierarchy of both Road Builder components.



HOW IT WORKS

General

The basic structure of Road Constructor is as follows:



Road Constructor: This component is the heart of the system, it holds both the construction settings and a reference to a Road Set.

Settings: These are all the construction settings that can be applied by the user. The settings are configured and stored directly on the Road Constructor component.

Road Set: A scriptable object that stores the roads and other objects available for construction. A road set is required for construction, where one set can be shared among multiple Road Constructors.

Road Builder: A custom script that accesses the Road Constructor API. Two builders are included in the asset (Editor and Player), and you can also create your own if you need a custom solution.

Construction

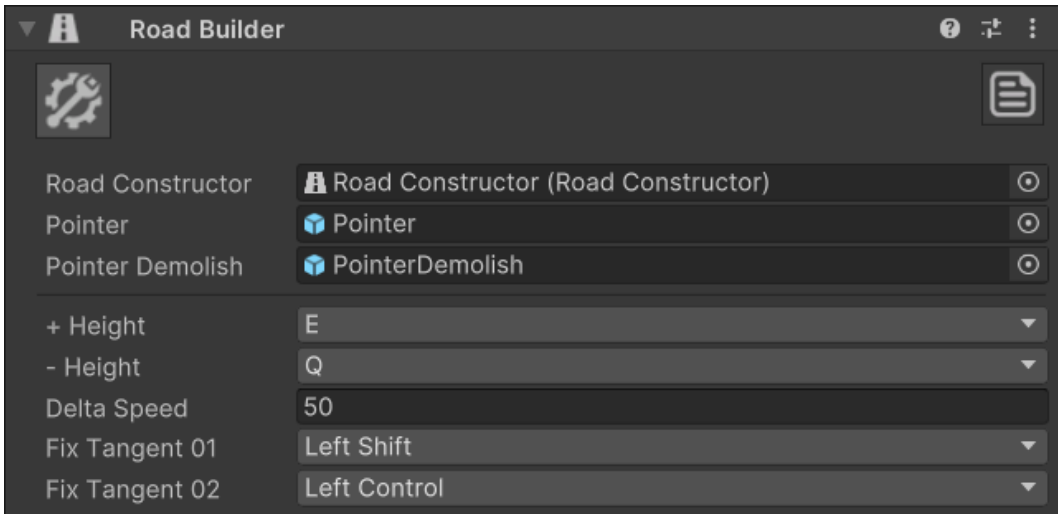
Both of the included road builder components (player and editor) share the same principles regarding how to construct the roads.

You will need a raycastable ground (terrain, mesh) with a layer that matches the Ground Layer in the Road Constructor settings. **For the Builder Editor, make sure the scene gizmos are enabled!**

After initializing the component, you can select a road and start placing it into the scene.

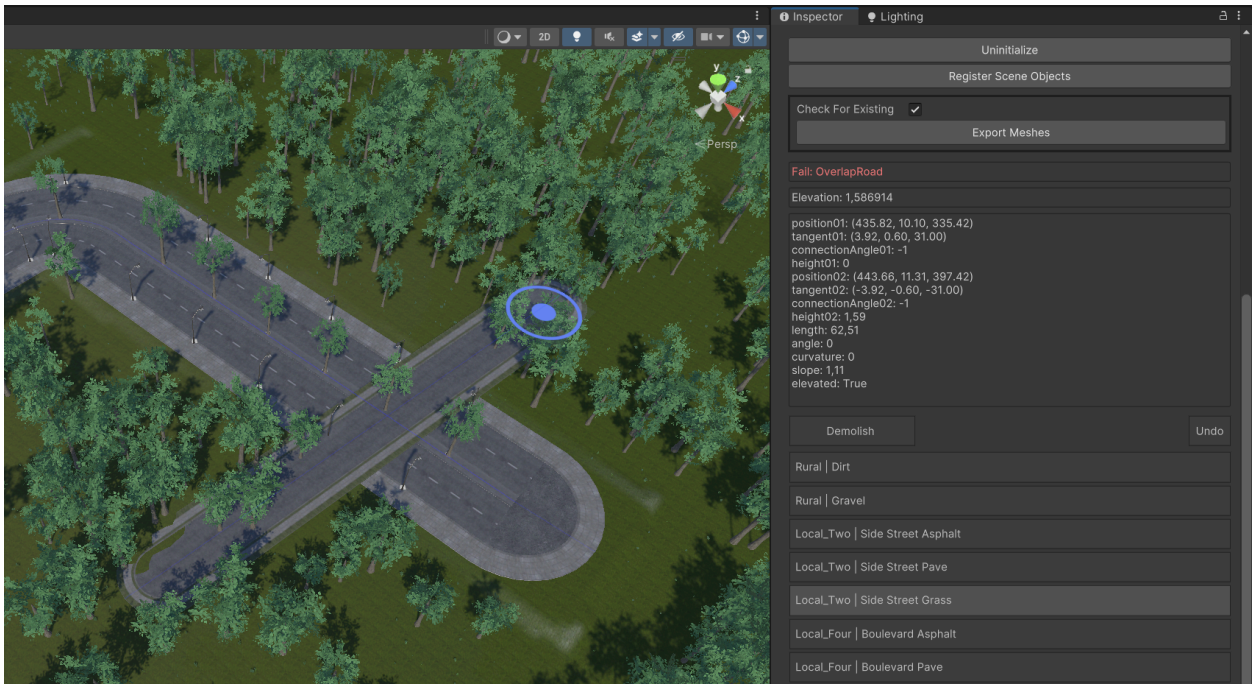
If you want to continue with objects already created from another session, you can register those objects. In the editor builder via the ‘Register Scene Objects’ button.

To display the different hotkeys available, click the settings icon in the top-left corner of the builder editor component.



When you place new roads onto existing ones, intersections are generated procedurally, taking into account the incoming curve angles, road width, slopes, and other factors. If you elevate your roads, the tool will detect overlapping roads and intersections, adjusting the construction accordingly.

Both included builders display real-time construction data, such as informational details like angles and heights, as well as error data (in red), which can prevent construction based on conditions specified in the Road Constructor component settings.

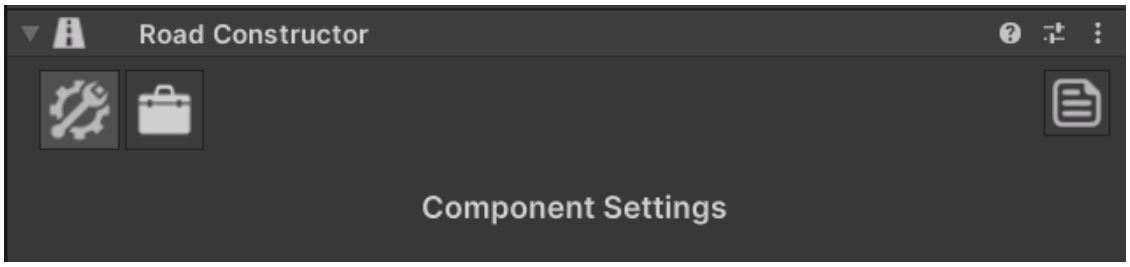


Remember that if you are working in the editor, you should always export your meshes once you are done with construction; otherwise, the mesh references on the roads may be lost.

Settings

These refer to the settings of the Road Constructor component.

When you add a Road Constructor component to a GameObject, you can access its settings by clicking the construction icon in the top-left corner of the component.



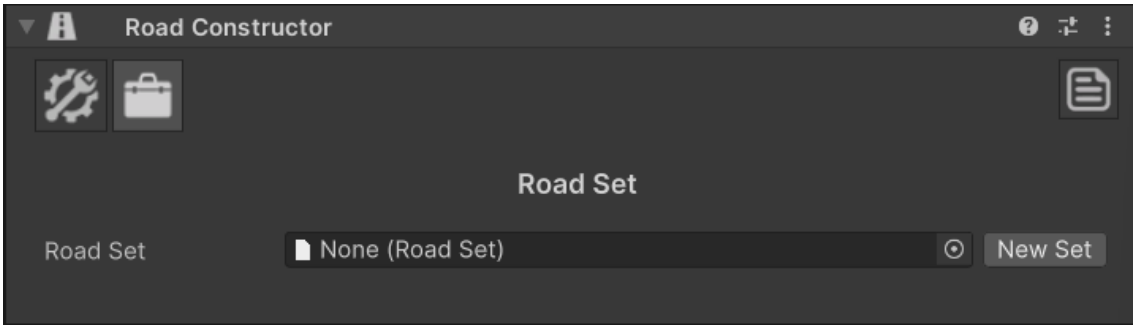
Each setting is a field that is serialized directly within the component itself, whether it's in the scene hierarchy or wrapped in a prefab.

Settings that are not self-explanatory should have tooltips to assist with understanding.
In practice, it's always recommended to experiment with the settings during actual construction. This way, you'll quickly learn what best fits your needs.

Road Set

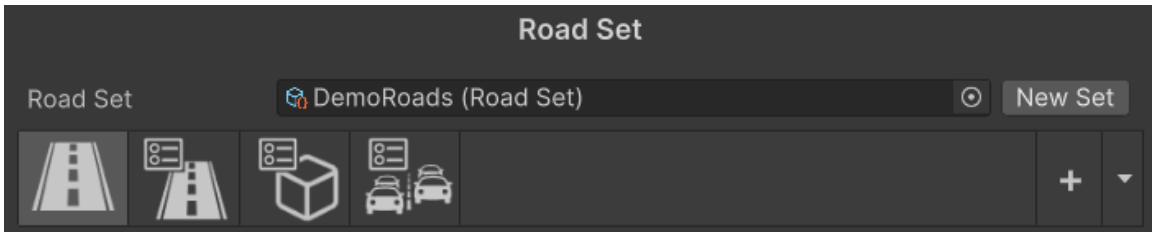
A Road Set is a ScriptableObject where roads are created and stored. ScriptableObjects serialize their data in the project folder, not in a scene hierarchy or prefab instance. A single road set can be used by multiple Road Constructor components.

To visualize a road set, click the suitcase icon in the top-left corner of a Road Constructor component, next to the settings icon.



Here, you can reference an existing set or create a new one using the dedicated button on the right.

After assigning a road set, it will be displayed in the Inspector.



Of the three available menus, the icon on the far left represents the actual roads, while the other two icons only serve as presets. Once you become familiar with the road creation process, you may find these presets to be a convenient way to share settings and save time.

You can start creating new roads using the + button on the right.

For further information on the process, please refer to the ROAD CREATION section.

Road Builder

A 'Road Builder' is simply a term used to describe a script that interacts with the Road Constructor API.

A builder calls methods such as 'roadConstructor.ConstructObjects' or 'roadConstructor.Demolish,' and it manages the user interface for displaying the roads.
It is kept completely separate from the main component, allowing for greater flexibility when creating your own solution.

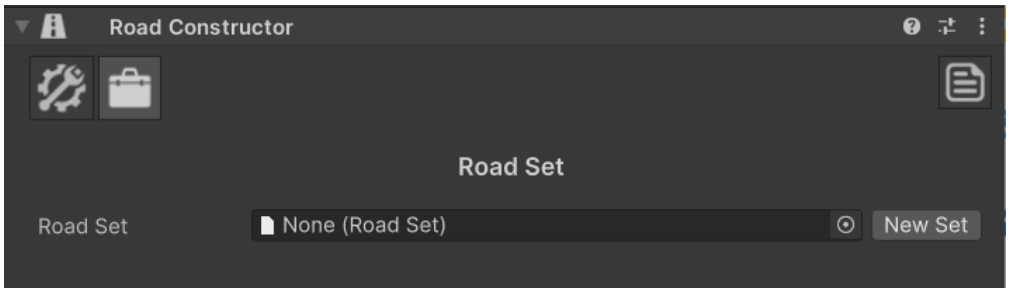
You can find example Road Builders in the 'PampelGames/RoadConstructor/Demo' folder. These builders are designed to automatically integrate custom roads.

If you intend to build your own builder or expand on the existing ones, you'll find the 'RoadBuilderPlayer.cs' script in the 'Demo/Scripts' folder. It is well commented and provides a common practical example for road building in strategy games.

ROAD CREATION

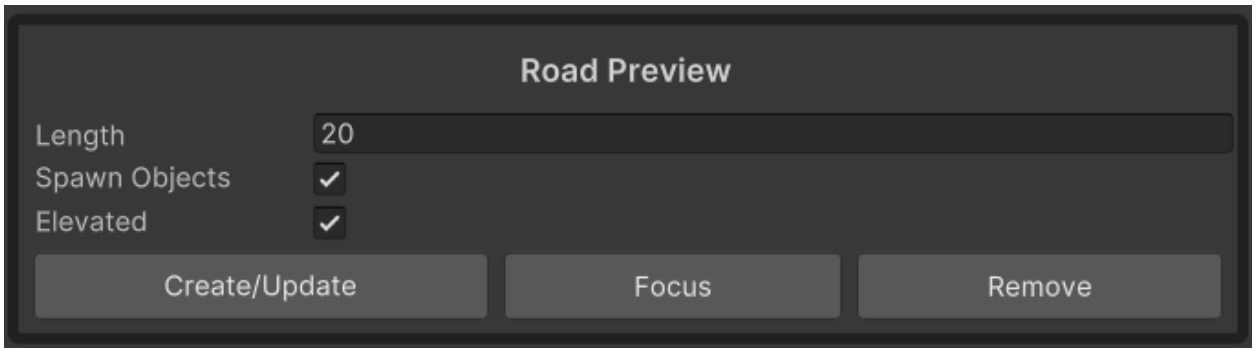
General

Roads are created, modified, and stored within Road Sets, which are scriptable objects (data containers) in your project folder. After adding a Road Constructor component to a GameObject, you can either reference an existing set or create a new one.

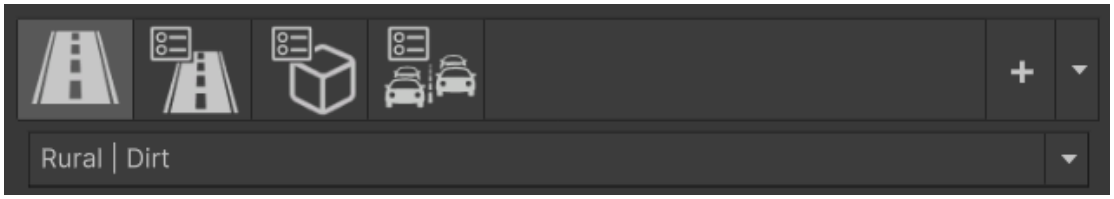


Roads can be previewed, providing a more convenient way than having to create the roads manually each time to test their appearance.

The Road Preview foldout is located just below the Road Set reference:

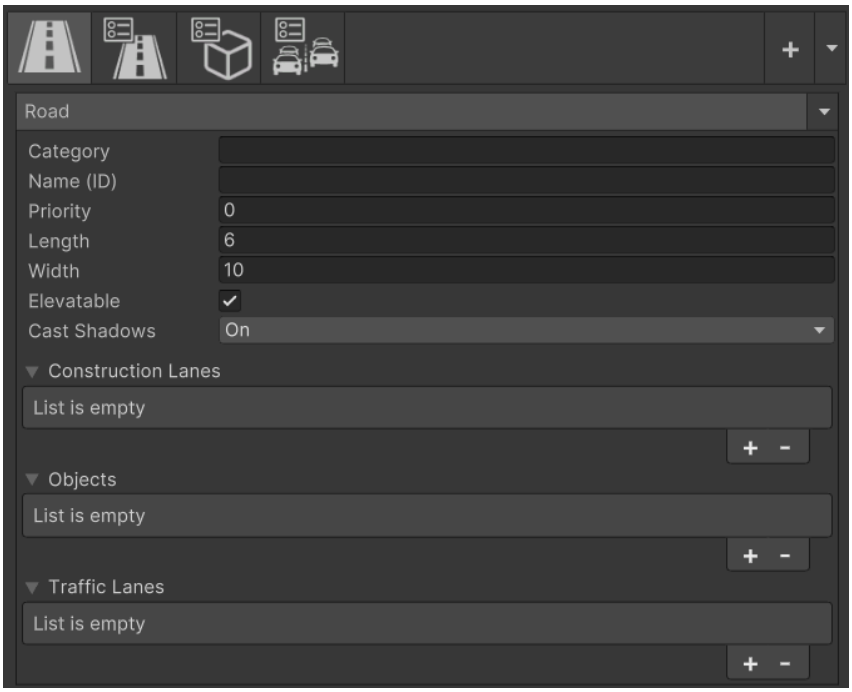


To preview roads, all you need to do is to ensure they are expanded within the Road Setup tab and not collapsed before clicking the Create/Update button.



Tip:
For a smoother workflow, you can open a second Inspector by right-clicking the Road Constructor component in the hierarchy and selecting “Properties…”.
You can then use the second Inspector solely for the “Create/Update” button.

Of the four available menus, the one on the left contains the actual roads for construction. New roads can be created using the '+' button on the right.



The category is used only internally for presets (which we will cover in the next chapter), the name is required and must be unique within this road set.

Lanes

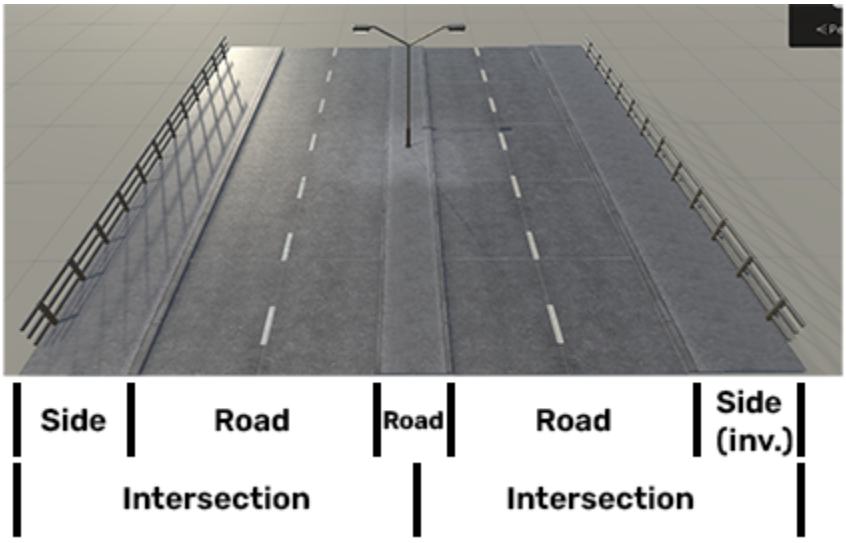
The 'Construction Lanes' list is where lanes are added. The position corresponds to the road's width, while the UVs represent the 'U' coordinate (horizontal) of the texture's UV data.

All included road materials can be found in the 'PampelGames/RoadConstructor/Content/Roads/Materials' folder. Feel free to include your own textures and materials (more information in the custom materials section).

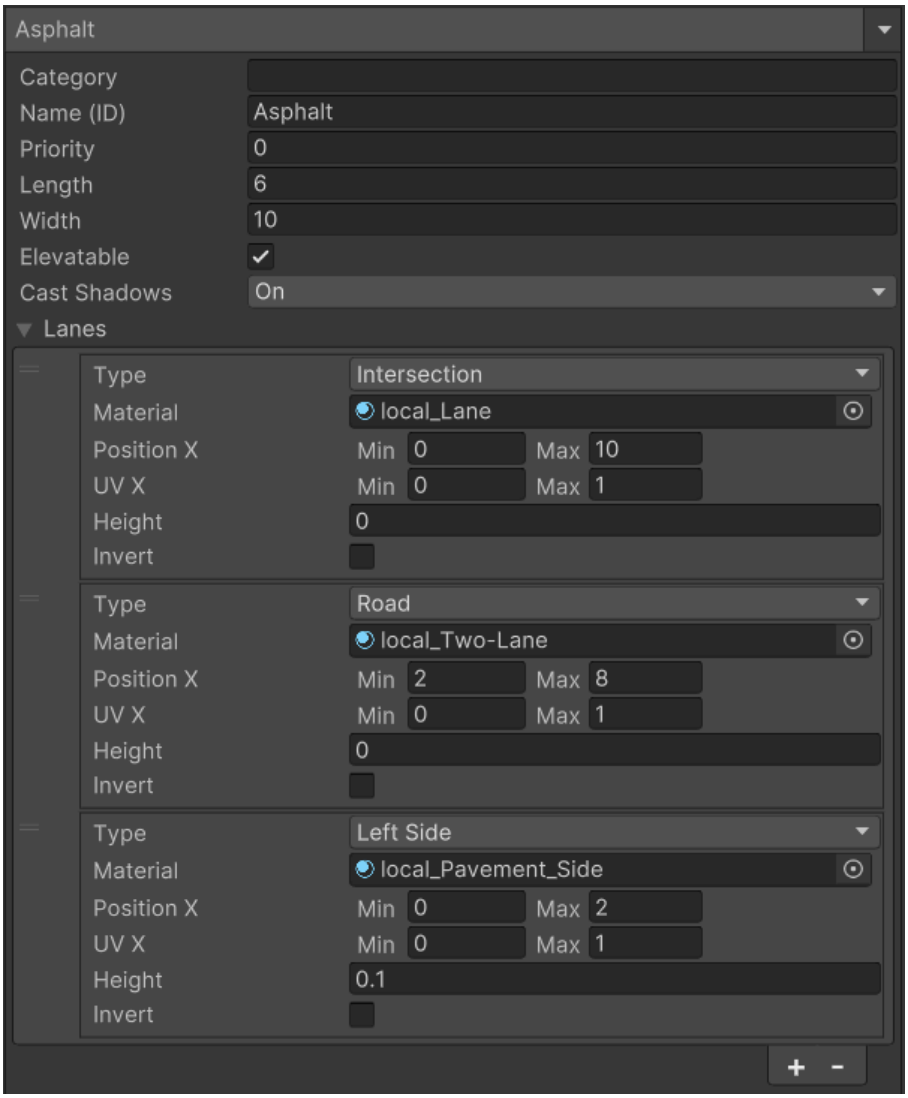
Important:

You can have any number of lanes for each type and arrange them in any order you prefer.

Just keep in mind that the road width is calculated by the tool using the minimum and maximum Width values. Lanes of type Road and Left Side (which is mirrored to the right side) should together cover the full road width, and lanes of type Intersection should also span the entire road width.



Another example of a simple road:

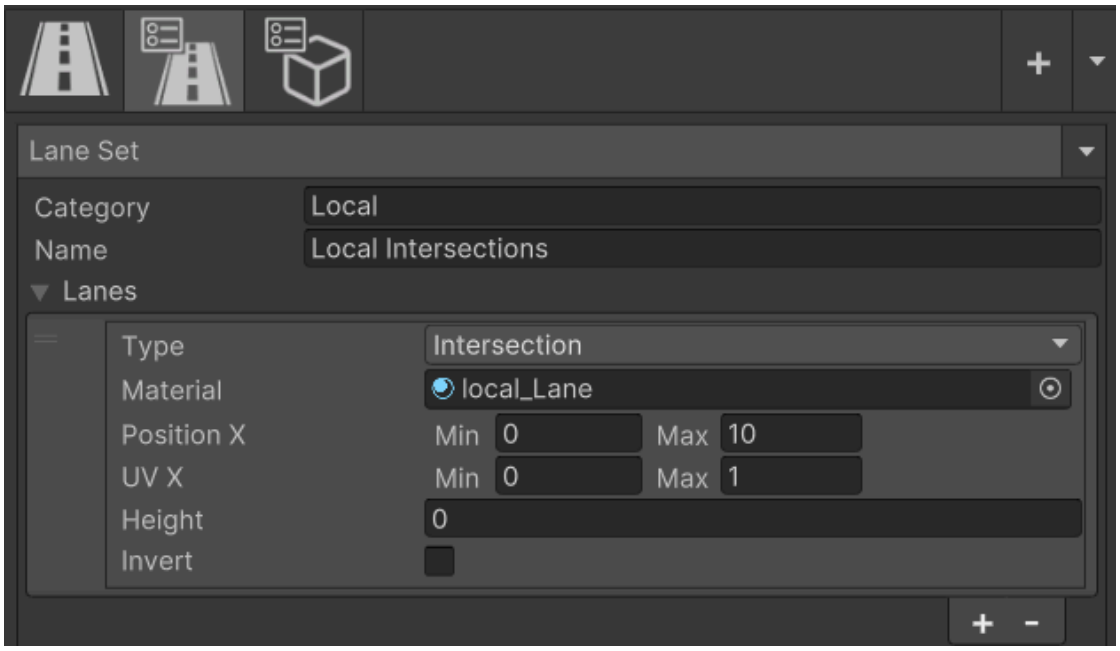


Construction Lane Presets

Lane presets are a great way to save time and maintain a cleaner, more manageable road set, especially for larger collections.

You can display the construction lane presets by clicking on the second icon.

Lanes are added in a manner similar to how roads are added, and a lane preset will be applied to all roads that belong to the matching category.

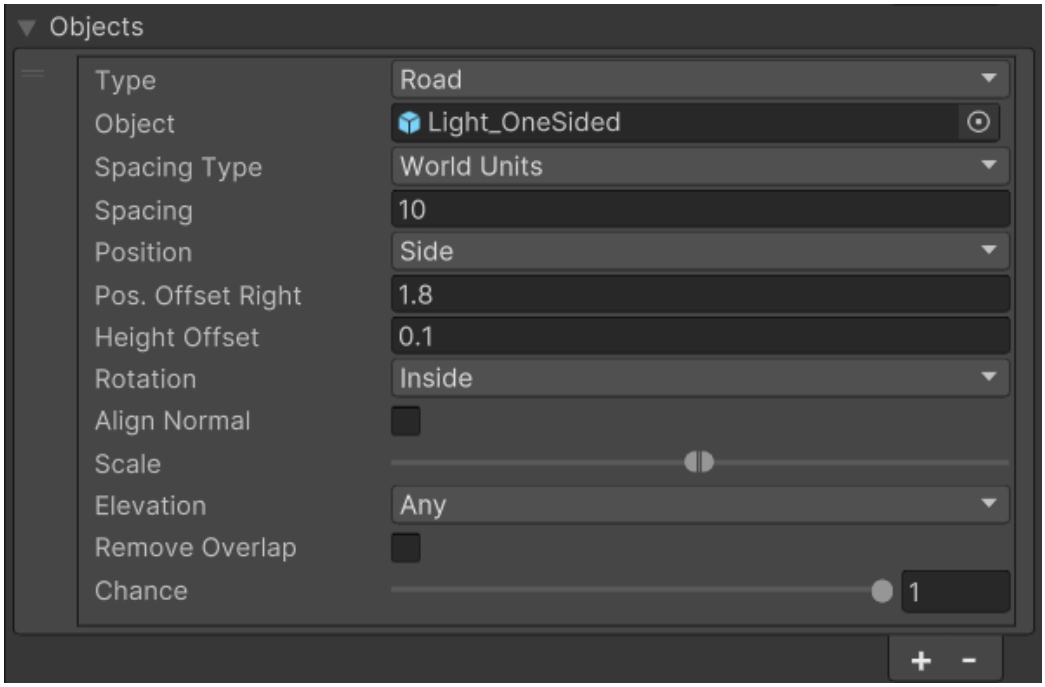


Object Presets

Objects are independent GameObjects that are parented to the created roads. These can include traffic lights, road signs, props, and more.

All the objects included with the asset can be found in the 'PampelGames/RoadConstructor/Content/Objects/Prefabs' folder.

To spawn objects, you can add them through the 'Objects' list available on each road. You can also create object presets, which function similarly to line presets.

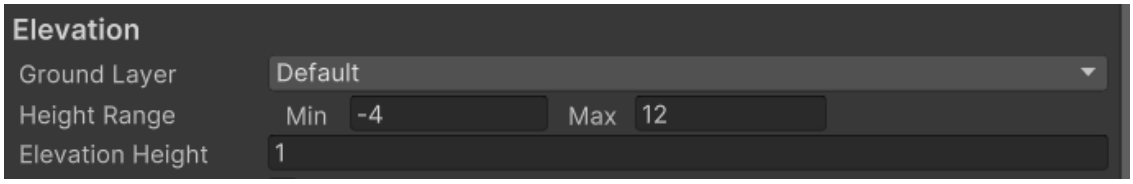


For more detailed examples of how objects can be set up, please refer to the 'DemoRoads' road set, located in the Demo folder.

CONSTRUCTION

Bridges

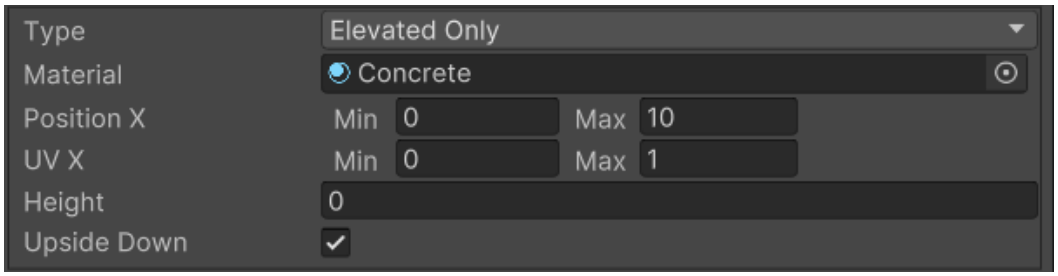
For the asset, bridges are simply elevated roads. A road becomes elevated when its height above the ground exceeds the 'Elevation Height' setting within the Road Constructor settings:



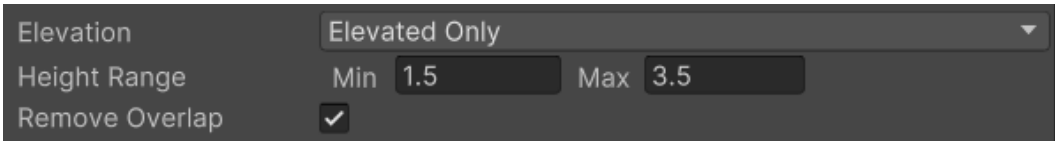
A road can consist of multiple elevated and non-elevated sections consecutively, such as on uneven terrain.

To make bridges (elevated roads) stand out from regular roads, there are two features you can use:

1. Construction lanes can be configured to apply only when the road is elevated. This is especially useful for creating geometry beneath the roads:



2. Objects can be spawned based on the elevation as well. This allows you for example to place specific objects beneath elevated roads, such as support pillars.

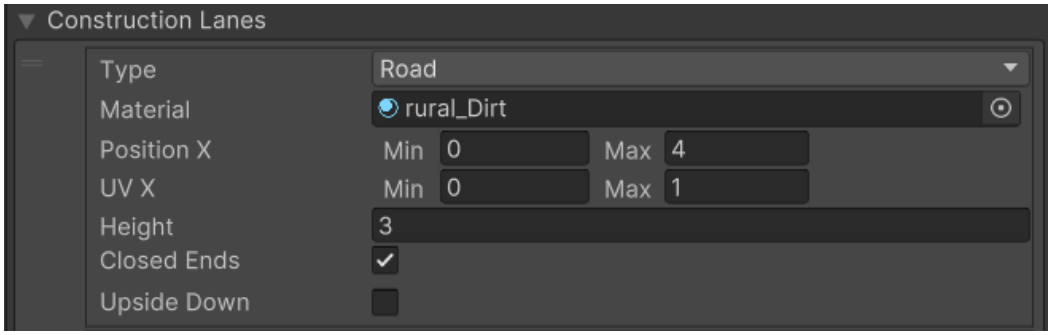


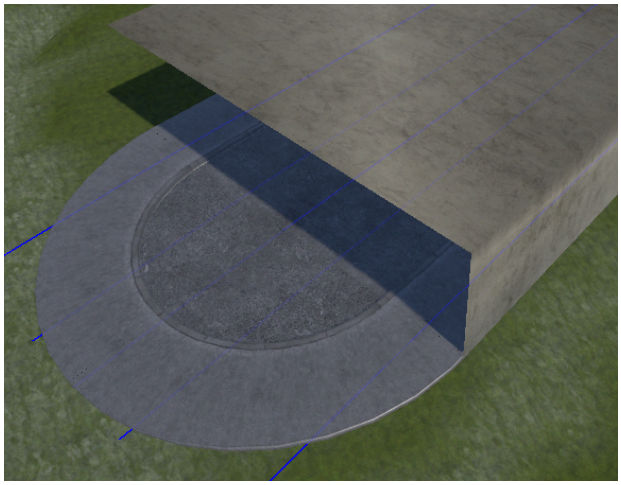
Tunnels

There is currently no official support for tunnels. However, there is a basic solution available that you may find useful for your project.

When you increase the height field in a construction lane, a checkbox 'Closed Ends' becomes visible. If you deactivate this setting, the lane will have open start and end sections, which can be used to create tunnels.

For example:





In practice, you'll typically want to ensure that the Position X covers the full width of the road and that the material being used supports backface rendering.

TRAFFIC SYSTEM

The included traffic system consists of two main features: traffic lanes and waypoints.

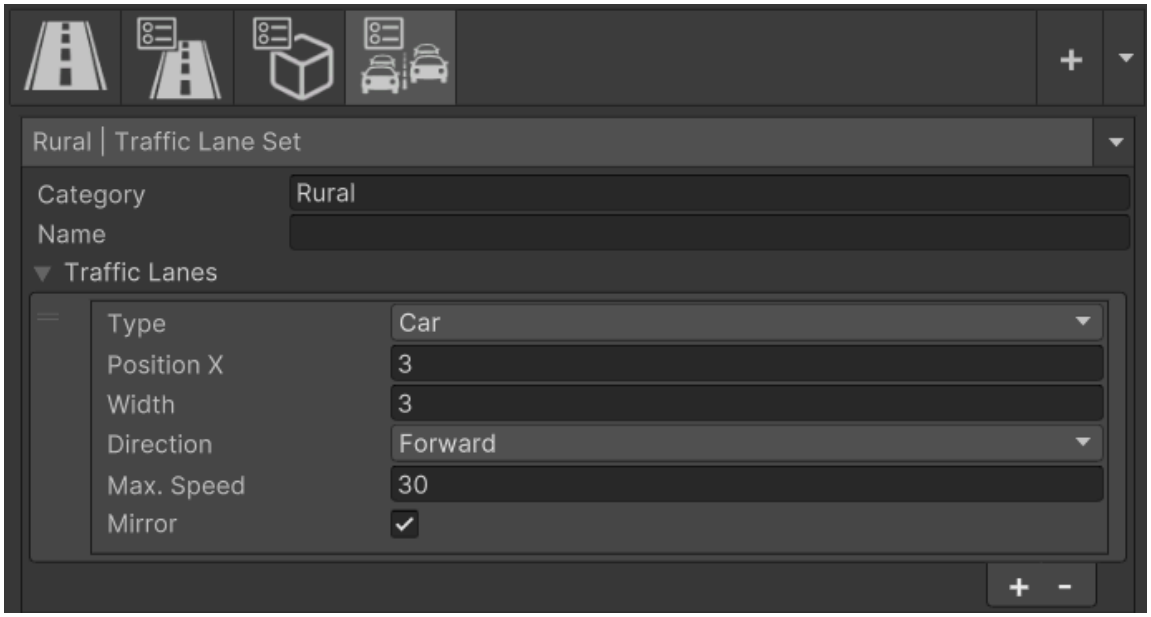
Note that for waypoints to be created, the road or intersection must have traffic lanes, and traffic lanes need to be defined in the road set.

So let's start at the traffic lanes setup and move on from there.

Traffic Lanes

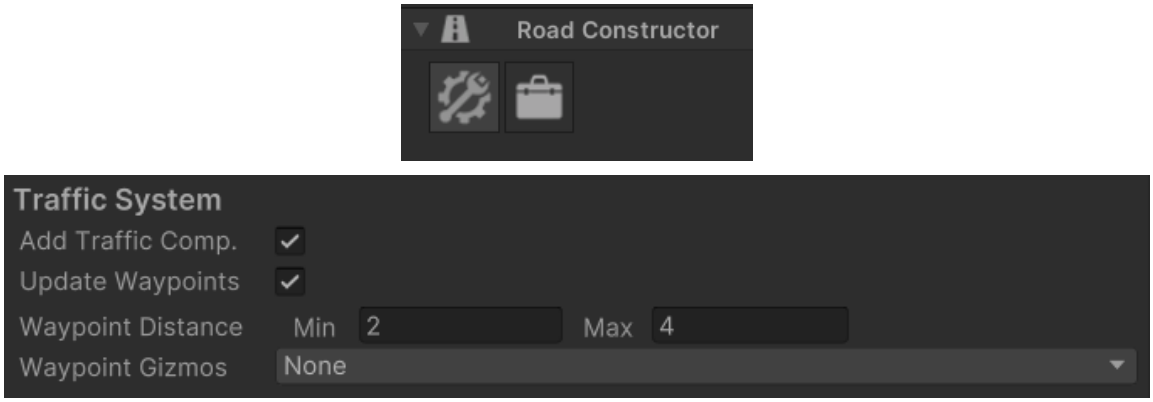
Traffic lanes are independent of construction lanes and contain separate splines for each lane (and direction), as well as additional data you may need.

Similar to construction lanes, traffic lanes can be added directly to individual roads, or you can use presets to conveniently apply a set of traffic lanes to multiple roads at once.

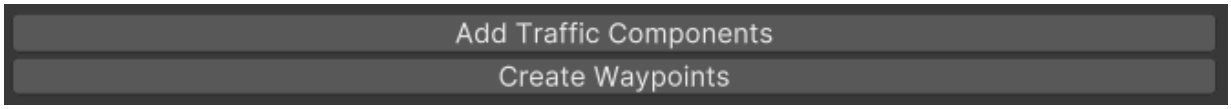


You can explore the DemoRoads construction set to find ready-made presets that you can use as a reference.

Traffic lanes can be added to constructed roads automatically by enabling the corresponding options in the Road Constructor component settings:



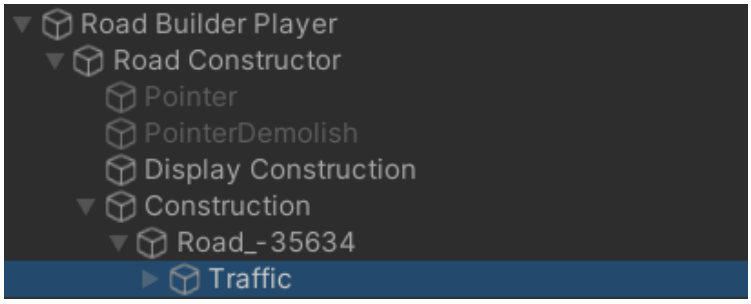
If you are working with the Road Builder Editor, you can (re-)create them manually using the dedicated buttons:



Or you can create them via code, either for single roads or for the full system at once (please refer to the API for more details).

```
roadConstructor.AddTrafficComponents();
roadConstructor.CreateAllWaypoints();
```

Once created, you can find the traffic components added as child GameObjects on each road and intersection.

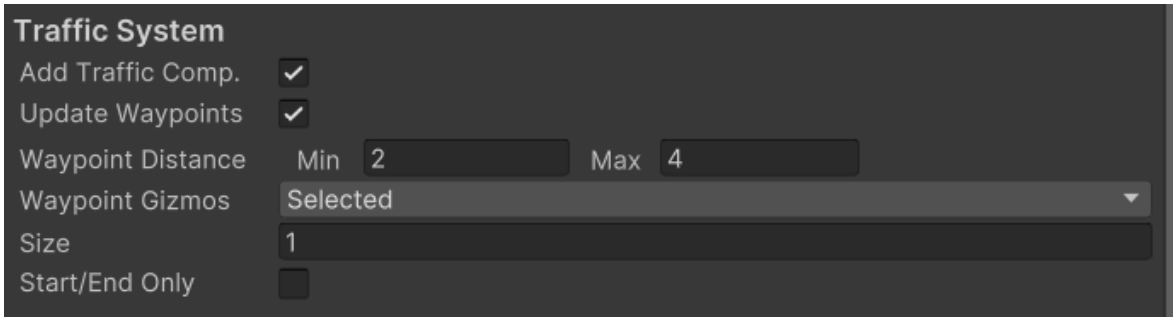


Waypoints

Waypoints are interconnected points along the traffic lane splines. They are created as MonoBehaviour components and are parented to the corresponding traffic lane components.

They do not require any additional manual setup, as they are automatically generated from the traffic lanes and updated dynamically during construction, demolition, and undo operations.

To visualize the waypoints in the scene, simply enable the gizmos within the Road Constructor settings.



Waypoints can be retrieved by code either individually:

```
var sceneObjects = roadConstructor.GetSceneObjects();

for (var i = 0; i < sceneObjects.Count; i++)
{
   var trafficLanes =
sceneObjects[i].GetTrafficLanes(trafficLaneType,
trafficLaneDirection);

   for (var j = 0; j < trafficLanes.Count; j++)
   {
       var waypoints = trafficLanes[j].GetWaypoints();
      …
```

Or simply by calling:

```
roadConstructor.GetWaypoints();
```

SAVE SYSTEM

The API provides convenient methods to retrieve easily serializable data, which you can use with custom save systems. Loading the data and recreating the road system is just as straightforward.

The following methods are available to support the saving process (see API for details):

```
public List<SerializedSceneObject> GetSerializableRoadSystem()

public void AddSerializableRoadSystem(List<SerializedSceneObject>
serializedSceneObjects)

public void SetSerializableRoadSystem(List<SerializedSceneObject>
serializedSceneObjects)
```

With the exception of the UnityEngine.Splines.Spline class and UnityEngine.Splines.BezierKnot struct, all data is stored as value types, incl. Lists of value types.

Please see the next page for examples.

Example 1

Here an example of how to save and load a road system using EasySave from the Unity Asset Store. This should work similarly to other generic saving solutions:

Saving

```
var roadSystem = roadConstructor.GetSerializableRoadSystem();
ES3.Save(SaveKey, roadSystem);
```

Loading

```
var roadSystem = ES3.Load<List<SerializedSceneObject>>(SaveKey);
roadConstructor.SetSerializableRoadSystem(roadSystem);
```

Example 2

You can also use the PGSaveSystem, which is included in the asset by default.
To do this, you'll need an additional class to hold the road system. Mark this class as [Serializable], and mark the road list as [SerializeReference].

```
[Serializable]
public class RoadSerialization
{
   [SerializeReference] public List<SerializedSceneObject>
serializedSceneObjects;
}

private void SaveRoadSystem()
{
   var serializedSceneObjects =
roadConstructor.GetSerializableRoadSystem();
   var roadSerialization = new RoadSerialization();
   roadSerialization.serializedSceneObjects = serializedSceneObjects;
   PGSaveSystem.Save(roadSerialization, "roads", 0);
}

private void LoadRoadSystem()
{
   var roadSerialization =
PGSaveSystem.Load<RoadSerialization>("roads", 0);
   var serializedSceneObjects =
roadSerialization.serializedSceneObjects;
   roadConstructor.SetSerializableRoadSystem(serializedSceneObjects);
}
```

INTEGRATIONS

The following assets are integrated with Road Constructor. Please refer to the documentation links provided below for setup guides.

Railway Constructor

Documentation

Mobile Traffic System v2.0

Documentation

Mobile Pedestrian System

Documentation

DOTS Traffic City

Documentation

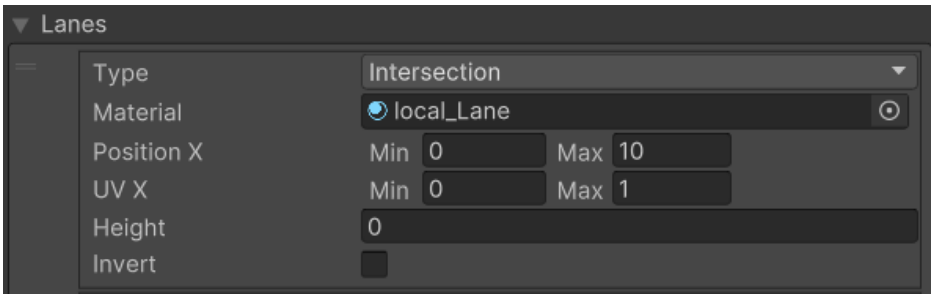
CUSTOMIZATION

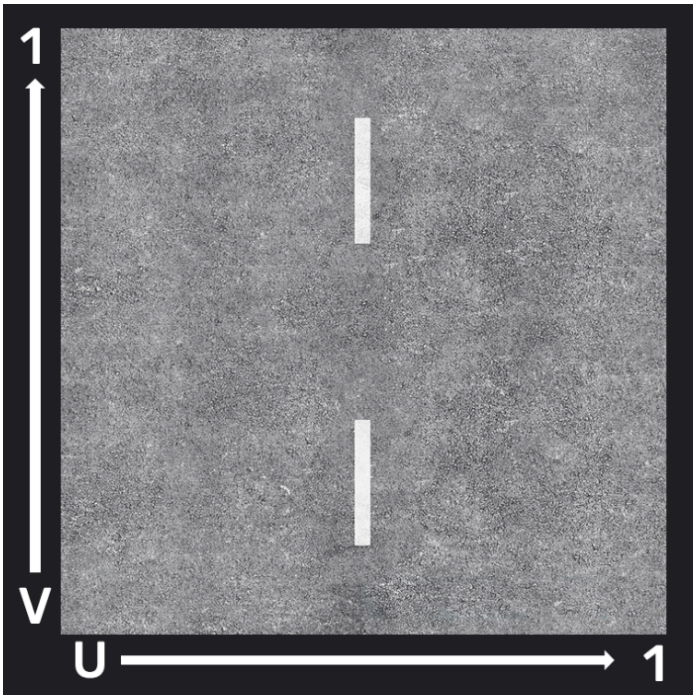
Materials

You can integrate custom road textures and materials as desired. Simply assign them in the material reference fields for the road lanes within the road set.

The length of the road always spans the full 'V' coordinate (vertical) of the texture. The 'U' coordinate is variable and can be set within the lanes. For example, if you have a texture that covers multiple roads or side lanes, this feature allows you to split the texture across those lanes.

If you don't want your 'V' coordinate to be cut off, you can use the 'Road UV' option within the Road Constructor's Construction settings.





Road Builder

A 'Road Builder' simply refers to a script that interacts with the Road Constructor API. You don’t have to create a 'Builder' at all; you can simply call the Road Constructor methods from any location you prefer.

The included 'RoadBuilder.cs' (editor) and 'RoadBuilderPlayer.cs' (runtime) scripts both inherit from 'RoadBuilderBase.cs' because they share certain functionalities.

The Road Builder Editor should cover everything you need for road creation in the editor. The logic is somewhat more complex because it runs in the 'OnSceneGUI' method within an editor script rather than simply in an update loop.

The Road Builder Player is located in the demo folder, as it is intended to provide you with an idea of how to utilize the Road Constructor in actual gameplay.

If you want to create your own game logic, you may want to at least take a quick look at the player script to see what methods are available in the Road Constructor API and how you might want to call them. The script is well commented, and with basic C# knowledge, you should be able to familiarize yourself with it. Just note that the demo uses the UI Toolkit, which you are not required to use.

API

RoadConstructor.cs

```
/// <summary>
///     Initializes this <see cref="RoadConstructor" /> component for
constructing.
///     In play-mode, this is done automatically in Awake.
/// </summary>
public void Initialize()


/// <summary>
///     Deinitializes this <see cref="RoadConstructor" /> component,
///     which includes clearing the history and removing helper
objects.
/// </summary>
public void Uninitialize()


/// <summary>
///     Registers existing scene objects for construction.
///     The component has to be initialized first.
///     The active construction set needs to contain the road name for
each object to be registered.
/// </summary>
public void RegisterSceneObjects()


/// <summary>
///     Registers existing scene objects for construction.
///     The component has to be initialized first.
///     The active construction set needs to contain the road name for
each object to be registered.
/// </summary>
public void RegisterSceneObjects(out List<RoadObject>
sceneRoadObjects, out List<IntersectionObject>
sceneIntersectionObjects)
```

```
/// <summary>
///     Gets a <see cref="Road" /> by name.
/// </summary>
public bool TryGetRoad(string roadName, out Road road)


/// <summary>
///     Gets a <see cref="RoadDescr" /> by road name.
/// </summary>
public bool TryGetRoadDescr(string roadName, out RoadDescr roadDescr)


/// <summary>
///     Gets a list of all registered <see cref="RoadObject" />s from
the scene.
/// </summary>
public List<RoadObject> GetRoads()


/// <summary>
///     Gets a list of all registered <see cref="IntersectionObject"
/>s from the scene.
/// </summary>
public List<IntersectionObject> GetIntersections()


/// <summary>
///     Gets a list of all registered <see cref="RoadObject" />s and
<see cref="IntersectionObject" />s from the scene.
/// </summary>
public List<SceneObject> GetSceneObjects()


/// <summary>
///     Attempts to get a <see cref="SceneObject" /> from the scene
within a specific search radius at a given position.
/// </summary>
/// <param name="position">The position to search around.</param>
/// <param name="searchRadius">The radius of the search area.</param>
/// <param name="sceneObject">The retrieved <see cref="SceneObject" />
if found; otherwise, null.</param>
/// <returns>True if a <see cref="SceneObject" /> is found within the
specified radius; otherwise, false.</returns>
public bool TryGetSceneObject(Vector3 position, float searchRadius,
out SceneObject sceneObject)
```

```
/// <summary>
///     Gets the parent object holding all registered <see
cref="RoadObject" />s and <see cref="IntersectionObject" />s from the
scene.
/// </summary>
public Transform GetConstructionParent()


/// <summary>
///     Clears all displayed objects and displayed demolish objects in
the scene hierarchy.
/// </summary>
public void ClearAllDisplayObjects(bool activateSceneObjects = true)


/// <summary>
///     Clears all displayed objects in the scene hierarchy.
/// </summary>
public void ClearDisplayedObjects()


/// <summary>
///     Clears all displayed demolish objects in the scene hierarchy.
/// </summary>
public void ClearDisplayedDemolishObjects()


/// <summary>
///     Clears all displayed move intersection objects in the scene
hierarchy.
/// </summary>
public void ClearDisplayedMoveIntersectionObjects()
```

```
/// <summary>
///     Snaps a position to the nearest road, intersection or grid
using the width of the road.
/// </summary>
public Vector3 SnapPosition(string roadName, Vector3 position, out
Overlap overlap)


/// <summary>
///     Snaps a position to the nearest road, intersection or grid
using a custom radius.
/// </summary>
public Vector3 SnapPosition(float radius, Vector3 position, out
Overlap overlap)


/// <summary>
///     Undoes the previous construction operation and restores the
previous state.
///     Requires objects within the undo storage.
/// </summary>
public void UndoLastConstruction()


/// <summary>
///     Removes all registered undo objects from the scene.
/// </summary>
public void ClearUndoStorage()


/// <summary>
///     Creates a road object and two intersection objects (start and
end) for a segment between two positions.
///     These objects are not initialized in the system and should be
deleted in the next frame via
///     the <see cref="ClearAllDisplayObjects" /> method.
/// </summary>
/// <param name="roadName">The name of the road.</param>
/// <param name="position01">The first road position.</param>
/// <param name="position02">The second road position.</param>
/// <returns>The construction result, including new objects and
detailed construction failures.</returns>
public ConstructionResultRoad DisplayRoad(string roadName, float3
position01, float3 position02)
```

```
/// <summary>
///     Creates a road object and two intersection objects (start and
end) for a segment between two positions.
///     These objects are not initialized in the system and should be
deleted in the next frame via
///     the <see cref="ClearAllDisplayObjects" /> method.
/// </summary>
/// <param name="roadName">The name of the road.</param>
/// <param name="position01">The first road position.</param>
/// <param name="position02">The second road position.</param>
/// <param name="roadSettings">Optional settings that can be
dynamically applied.</param>
/// <returns>The construction result, including new objects and
detailed construction failures.</returns>
public ConstructionResultRoad DisplayRoad(string roadName, float3
position01, float3 position02, RoadSettings roadSettings)
/// <summary>
///     Constructs new intersection and road objects for a segment
between two positions and registers them into the construction system.
/// </summary>
/// <param name="roadName">The name of the road.</param>
/// <param name="position01">The first road position.</param>
/// <param name="position02">The second road position.</param>
/// <returns>The construction result, including new objects and
detailed construction failures.</returns>
public ConstructionResultRoad ConstructRoad(string roadName, float3
position01, float3 position02)


/// <summary>
///     Constructs new intersection and road objects for a segment
between two positions and registers them into the construction system.
/// </summary>
/// <param name="roadName">The name of the road.</param>
/// <param name="position01">The first road position.</param>
/// <param name="position02">The second road position.</param>
/// <param name="roadSettings">Optional settings that can be
dynamically applied.</param>
/// <returns>The construction result, including new objects and
detailed construction failures.</returns>
      public ConstructionResultRoad ConstructRoad(string roadName,
float3 position01, float3 position02, RoadSettings roadSettings)
```

```
/// <summary>
/// Displays a roundabout at the specified position with the given
road name and radius.
/// </summary>
/// <param name="roadName">The name of the road to use for the
roundabout.</param>
/// <param name="position">The position where the center of the
roundabout will be located.</param>
/// <param name="radius">The radius of the roundabout.</param>
/// <returns>The construction result of displaying the
roundabout.</returns>
public ConstructionResultRoundabout DisplayRoundabout(string
roadName, float3 position, float radius)
```

```
/// <summary>
/// Constructs a roundabout at the specified position with the given
road name and radius.
/// </summary>
/// <param name="roadName">The name of the road for the
roundabout.</param>
/// <param name="position">The position where the roundabout will be
constructed.</param>
/// <param name="radius">The radius of the roundabout.</param>
/// <returns>A ConstructionResultRoundabout indicating the result of
the construction.</returns>
public ConstructionResultRoundabout ConstructRoundabout(string
roadName, float3 position, float radius)
```

```
/// <summary>
/// Displays move intersection object.
/// </summary>
/// <param name="currentPosition">The current position where the
intersection object is displayed.</param>
/// <param name="searchRadius">The search radius within which the
intersection object will be found.</param>
/// <param name="newPosition">The new position where the intersection
object will be displayed.</param>
/// <param name="deactivateSceneObjects">A flag indicating whether to
deactivate scene objects.</param>
/// <returns>A list of GameObjects representing the displayed move
intersection objects.</returns>
public List<GameObject> DisplayMoveIntersection(Vector3
currentPosition, float searchRadius, Vector3 newPosition,
           bool deactivateSceneObjects = true)


/// <summary>
/// Moves an existing <see cref="IntersectionObject"/> to a new
position.
/// Undo history will be lost after this method is called.
/// </summary>
/// <param name="currentPosition">The position where to search for the
intersection.</param>
/// <param name="searchRadius">The radius within which to search for
the intersection.</param>
/// <param name="newPosition">The new position to move the
intersection to.</param>
/// <returns>A <see cref="ConstructionResultMoveIntersection"/>
representing the result of the operation.</returns>
public ConstructionResultMoveIntersection MoveIntersection(Vector3
currentPosition, float searchRadius, Vector3 newPosition)
```

```
/// <summary>
///     Reverses the direction of a road within the specified search
radius from the given position.
///     If a valid road is found, it is inverted and replaced with a
newly constructed road.
/// </summary>
/// <param name="position">The position at which to search for a
road.</param>
/// <param name="searchRadius">The radius within which to search for a
road.</param
/// <param name="replacedRoad">Newly constructed road.</param>
public bool ReverseRoadDirection(Vector3 position, float
searchRadius, out RoadObject replacedRoad)


/// <summary>
///     Reverses the direction of the specified road.
/// </summary>
/// <param name="road">The road object whose direction is to be
reversed.</param>
/// <param name="replacedRoad">The new road object created with the
reversed direction.</param>
public void ReverseRoadDirection(RoadObject road, out RoadObject
replacedRoad)
```

```
/// <summary>
///     Creates display objects for demolishing for the specified
position.
///     These objects are not initialized in the system and should be
deleted in the next frame via
///     the <see cref="ClearAllDisplayObjects" /> method.
/// </summary>
/// <param name="position">The position from which to demolish
objects.</param>
/// <param name="searchRadius">Search radius to find the demolishable
objects.</param>
/// <param name="deactivateSceneObjects">Deactivates the scene
objects.</param>
/// <returns></returns>
       public List<GameObject> DisplayDemolishObjects(Vector3
position, float searchRadius, bool deactivateSceneObjects = true)


/// <summary>
///     Demolishes objects within a specified radius from a given
position.
/// </summary>
/// <param name="position">The position from which to demolish
objects.</param>
/// <param name="searchRadius">Search radius to find the demolishable
objects.</param>
public void Demolish(Vector3 position, float searchRadius)
```

```
/// <summary>
///     Updates the colliders of the specified scene objects based on
the component settings.
/// </summary>
/// <param name="sceneObjects">The list of scene objects to update
colliders for.</param
public void UpdateColliders(List<SceneObject> sceneObjects)


/// <summary>
///     Update the layers and tags of scene objects based on the
component settings.
/// </summary>
/// <param name="sceneObjects">List of scene objects to update layers
and tags</param>
public void UpdateLayersAndTags(List<SceneObject> sceneObjects)


/// <summary>
///     Editor Only. Exports road and intersection meshes into the
project folder.
/// </summary>
/// <param name="checkExistingMeshes">
///     Checks the project folder for each mesh before exporting.
///     If set to false, each mesh will be exported regardless of
whether one already exists.
/// </param>
public void ExportMeshes(bool checkExistingMeshes)
```

```
/// <summary>
///     Removes existing <see cref="Traffic" /> components and creates
new ones for the full existing road system.
///     The traffic component is required for the <see
cref="AddWaypoints" /> method.
/// </summary>
public void AddTrafficComponents()


/// <summary>
///     Removes existing <see cref="Traffic" /> components and creates
new ones for the specified roads/intersections.
///     The traffic component is required for the <see
cref="AddWaypoints" /> method.
/// </summary>
public void AddTrafficComponents(List<RoadObject> roads,
List<IntersectionObject> intersections)


/// <summary>
///     Creates a list of interconnected waypoints for the specified
roads/intersections and adds them to the existing waypoint system.
///     Before using this method, make sure all registered roads have
a 'Traffic' component assigned.
///     Either by enabling 'Add Traffic Comp.' in the settings or by
invoking the AddTrafficComponents method.
/// </summary>
public void AddWaypoints(List<RoadObject> roads,
List<IntersectionObject> intersections)
```

```
/// <summary>
///     Creates a list of interconnected waypoints for the specified
roads/intersections and adds them to the existing waypoint system.
///     Before using this method, make sure all registered roads have
a 'Traffic' component assigned.
///     Either by enabling 'Add Traffic Comp.' in the settings or by
invoking the AddTrafficComponents method.
/// </summary>
/// <param name="maxDistance">Maximum space between each
waypoint.</param>
public void AddWaypoints(List<RoadObject> roads,
List<IntersectionObject> intersections, float maxDistance)


/// <summary>
///     Creates a list of interconnected waypoints for the specified
roads/intersections and adds them to the existing waypoint system.
///     Before using this method, make sure all registered roads have
a 'Traffic' component assigned.
///     Either by enabling 'Add Traffic Comp.' in the settings or by
invoking the AddTrafficComponents method.
/// </summary>
/// <param name="maxDistance">Maximum space between each waypoint,
based on curvature.</param>
public void AddWaypoints(List<RoadObject> roads,
List<IntersectionObject> intersections, Vector2 maxDistance)


/// <summary>
///     Creates a list of interconnected waypoints from all road and
intersection splines in the scene, overwriting any existing waypoints.
///     Before using this method, make sure all registered roads have
a 'Traffic' component assigned.
///     Either by enabling 'Add Traffic Comp.' in the settings or by
invoking the AddTrafficComponents method.
/// </summary>
public void CreateAllWaypoints()
```

```
/// <summary>
///     Creates a list of interconnected waypoints from all road and
intersection splines in the scene, overwriting any existing waypoints.
///     Before using this method, make sure all registered roads have
a 'Traffic' component assigned.
///     Either by enabling 'Add Traffic Comp.' in the settings or by
invoking the AddTrafficComponents method.
/// </summary>
/// <param name="maxDistance">Maximum space between each
waypoint.</param>
public void CreateAllWaypoints(float maxDistance)


/// <summary>
///     Creates a list of interconnected waypoints from all road and
intersection splines in the scene, overwriting any existing waypoints.
///     Before using this method, make sure all registered roads have
a 'Traffic' component assigned.
///     Either by enabling 'Add Traffic Comp.' in the settings or by
invoking the AddTrafficComponents method.
/// </summary>
/// <param name="maxDistance">Maximum space between each waypoint,
based on curvature.</param>
public void CreateAllWaypoints(Vector2 maxDistance)


/// <summary>
///     Removes waypoints to these roads in connecting intersections.
/// </summary>
public void RemoveConnectingWaypoints(List<RoadObject> roads)


/// <summary>
///     Returns a list of all registered waypoints of the specified
type.
/// </summary>
public List<Waypoint> GetWaypoints(TrafficLaneType trafficLaneType)
```

```
/// <summary>
/// For each road marked as 'expanded' in the Road Setup tab, a
corresponding road is created in the scene.
/// These objects are parented to the 'Road Preview' GameObject and
not registered in the system.
/// </summary>
public void ConstructPreviewRoads(Transform parent, float roadLength,
bool elevated)


/// <summary>
/// Retrieves a list of serialized representations of all scene
objects
/// within the road system. Each object is converted into simple,
serializable
/// fields for saving or transmission purposes.
/// </summary>
/// <returns>A list of <see cref="SerializedSceneObject"/>
representing the serialized road system (incl. inheritors).</returns>
public List<SerializedSceneObject> GetSerializableRoadSystem()


/// <summary>
/// Reads serialized scene objects and adds them to the existing road
system.
/// </summary>
/// <param name="serializedSceneObjects">A list of serialized scene
objects containing data needed to restore the road system.</param>
public void AddSerializableRoadSystem(List<SerializedSceneObject>
serializedSceneObjects)


/// <summary>
/// Reads serialized scene objects and reconstructs the road system.
/// </summary>
/// <param name="serializedSceneObjects">A list of serialized scene
objects containing data needed to restore the road system.</param>
public void SetSerializableRoadSystem(List<SerializedSceneObject>
serializedSceneObjects)


/// <summary>
///     Removes all registered objects from the road system.
/// </summary>
public void ClearRoadSystem()
```