TRS 2018* Content Creation Guide

*Working Title. Product in Development

1. Introduction

Who is this Guide for?

This Content Creation Guide (CCG) is designed for people interesting in modelling, scripting or creating assets for Trainz. It is aimed at the "Trainz Railroad Simulator 2018" (working title) version of Trainz due to be released in 2018.

How to use this Guide

This CCG aims to provide a central source of information that will assist in creating content for the next version of Trainz. It references the Trainz Wiki heavily, and provides links to tutorials and other online resources.

Here are some tips to assist you make the most use of this Guide:

• Clicking on hyperlinks will take you either to the appropriate section in this Guide, or it will take you to an external webpage.
• Navigate by using the Table of Contents on the left
• Use the Index or Search tabs to help find the topic you are looking for.

Disclaimer

Trainz project began life in the year 2000. This means that it has been running a long time, and there have been many changes since the first iteration of Trainz.

Certain content creation guidelines published 5, 10 or 15 years ago may now be out of date. Also, the Trainz Wiki, which this document references heavily, contains some information that is obsolete for future versions of Trainz.

2. Where Do I Start?

Whether you are new to Trainz content creation or a veteran, we highly recommend that you work you way through this Content Creation Guide (CCG) to discover the wide range of information that will help you make better quality content for Trainz.

The CCG is divided into Chapters beginning with more general topics and working towards more advanced topics, then finishing off with a list of additional resources to review.

Online Resources

There are four main locations that provide information relating specifically to creating content for Trainz. These are listed here:

- Content Creation Blogs
- Trainz Channel on Twitch TV
- Content Creator Video Playlist
- Trainz Wiki

2.1. Content Creation Blogs

These blogs are aimed at providing all users an overview of the various features the Trainz Development team are working on for TRS18 and beyond.

Here is the list of Content Creation Blogs as at Jan 2018:

- Introduction to PBR
- Height-Map Ground Textures
- Detail Height-Map Ground Textures
- Controlling Level of Detail
- Draw Calls and Materials
- Mesh Libraries
- Texture Atlases
- Ground Clutter
- Parallax vs normal maps
- Crossfade LOD transitions for Clutter
- Detail Maps
- TurfFx Layers
- Parallax Ground Textures

2.2. Trainz Channel on Twitch TV

The Trainz Channel (on Twitch TV) provides a wide range of content related video tutorials.

Topics include 3D modelling tips as well as end-user guides to help create your own routes and sessions.

2.3. Content Creator Video Playlist

There are a large (and growing) number of Trainz Content Creation videos on Youtube.

2.4. Trainz Wiki Pages

What is Trainz Wiki?

The Trainz Wiki is the primary repository for reference materials for all Content related features as well as providing links for in-game help.

The Trainz Wiki provides a selection of How To Guides covering a wide range of topics, Content Creation topics for new and experienced content creators, as well as detailed documentation including terminology, configuration requirements, and performance-optimisation information.

Link to Trainz Wiki: Content Creation

3. Understanding Assets

In the next few Chapters, we provide a summary of the main topics you will need to understand in order to create content for Trainz.

We explain the concepts of Assets, Materials, Meshes, Textures, Attachment Points and Animations.

3.1. Structure of a Trainz Asset

Each item of Trainz content ("an asset") is created as the contents of a single operating system folder, containing a config.txt file and some number of supporting files and/or folders and their files. Files located in the content folder must obey the Filename character restrictions and each item should be directly relevant to the asset in question.

When submitted or as in newer Trainz releases, tested for validation before submitting, the asset's folder contents and the config.txt file will be validated against one another, the ideal minimum definitions will be bounced against those and each must be in balance and correct with respect to one another.

3.2. Config.txt file

The config.txt file is the central descriptor for a Trainz content item. Every content item has a single filed named config.txt in its folder, which determines the name, kind and unique ID of the item, as well as various kind-specific details. All other files used by the content item are located via the config.txt file.

The config.txt file is stored in the ACS Text Format. When hand-editing the file, care must be taken to use correct syntax or the meaning of parts or the entirety of the file may become permanently lost. When content is installed into Trainz, or when it is archived (in CDP Format or similar) the config.txt file is stored in a machine-optimised binary format. If the content is later extracted using Content Manager, the config.txt file is converted back to text format, however any custom formatting or erroneous syntax is already lost and cannot be restored.

Link To Wiki Page: Config.txt File

Sub-Topics on this page:

- Standard Tags
- Localisation
- Common Containers
- Related Links

3.3. Content Types

Each asset built for the Trainz environment follows a preset template, which defines what tags can be included in the config.txt file, what capabilities the asset will have, and how the user can manipulate the asset in-game. The following types of custom content are supported by Trainz:

There are currently over 40 supported Trainz content types, such as Locomotives (traincar), Scenery, Track and Engine.

Link to Wiki Page: Content Types

3.4. Asset Validation

All Trainz assets must be correctly formed in order to work properly across a wide range of software and hardware configurations. To aid in this process, Trainz includes asset validation functionality which checks each item of content for a range of known flaws. This validation process is performed both on your local Trainz installation, and also as part of the Download Station Upload Process. Assets which fail validation cannot be used in-game.

Link to Wiki Page: Asset Validation

Link to Wiki Page: List of Validation Errors

4. Understanding Materials

Note: Different Material Types are covered in more detail in the Material Types wiki page and in this Chapter.

Materials

Materials are a concept in 3D rendering that describes how to render a particular surface. Each render chunk is comprised of a single material, a vertex buffer, and an index buffer.

Each material is defined with a material type, one or more texture files (eg. a Diffuse map, Normal map, etc.) and some number of other parameters (strength, scale, color, etc.)

Materials do not exist as individually editable files, but typically reside within other files such as trainzmesh files. Each material is assigned a unique identifier (typically composed from the parent asset's KUID and a material name assigned in a modeling package such as Autodesk 3ds Max. This means that it is possible for meshes within the same parent asset to share materials, but it is not possible for meshes in separate assets to (accidentally, or deliberately) share materials. Where deliberate material sharing is desired, content creators may make use of a "mesh library" which is a non-placeable asset containing a number of meshes and textures acting as a common storage for the actual placeable assets.

See Also:

- Understanding TRS18 Material Types
- Understanding Rendering

Link to Wiki Page: Materials

4.1. Texture.txt Files

Trainz Texture files are ASCII text format specification files describing a single texture. Texture files have a ".texture.txt" name extension, however when a texture file is specified within a config.txt file or similar, the ".txt" must be omitted (as you are specifying a resource name, not a file path), leaving only the ".texture" extension.

Texture files (*.texture.txt) should be distinguished from image files (*.tga, etc.) and materials. A material may utilize zero or more texture files, and a texture file may utilize one or more image files. While we often refer to the image data as "a texture", this usage isn't strictly correct as any given texture also has a number of metadata attributes which are not present in the source image. Additionally, a texture may be compressed (at installation time) which may actually lose some of the data originally present in the source image.

When importing a trainzmesh file from an FBX file, any image files referenced in the FBX which do not already have corresponding texture files will have generic texture files created automatically. These files may be suitable for basic purposes, but may need to be customized for more advanced functionality. Once the file has been created (and optionally customised) the importer will not recreate/overwrite it on subsequent runs.

This page describes content format v4.6.

Link to Wiki Page: .texture.txt file

4.2. Texture atlas

A texture atlas is a single larger texture which combines what would normally be several distinct smaller textures. This technique can be a significant performance optimisation.

Link to Wiki Page: Texture_atlas

5. Understanding Meshes

A mesh is a collection of render chunks, an optional skeleton, often some attachment points and potentially other metadata. It is stored in a mesh file. This Chapter provides information on the various elements the comprise a Mesh.

Link to Wiki Page: Mesh

5.1. Trainzmesh files

The Trainzmesh file format is a simple mesh file format created for use with TANE. It serves as a simple replacement for the more complex IM file format which was used by Auran Jet in older versions of the Trainz product.

Trainzmesh files have the extension ".trainzmesh" and are created by importing FBX files using Content Manager.

Link to Wiki Page: Trainzmesh_file

5.2. IM files

Please note that .im files are obsolete as of trainz-build 4.5. Trainz now imports from the FBX file format, which is widely used as a generic interchange format.

The IM file format is still supported for backwards compatibility but may not support newer features and techniques.

Link to Wiki Page: IM_Files

5.3. Mesh stitching

Mesh stitching is a rendering optimization that is automatically applied to many types of static meshes. It allows multiple smaller objects to be uploaded to the GPU as a single object, reducing per-object overheads that would otherwise apply.

Link to Wiki Page: Mesh_stitching

5.4. Mesh Libraries

In order to allow several Trainz assets to all reference the same materials you need to create a Mesh Library asset. This process is super simple.

Link to Wiki Page: Mesh_Libraries

5.5. Animation

Any mesh with a skeleton may have animations applied. Animations in Trainz are stored in the "*.kin" file format. These are created during FBX file import and are referenced in the "mesh-table" Container etc. If a mesh has a skeleton but does not have any animation applied, it will be displayed in its bind pose. Animations are not strongly tied to the mesh from which they are initially exported; they may be reused with any compatible mesh.

Sub-Topics:

- Partial Animation
- Idle Animations
- Animating Multiple Meshes

Link to Wiki Page: Animation

5.6. Skeleton

Animation in Trainz (and most other GPU-based renderers) involves several steps:

1. Creating a mesh, laid out in bind pose.
2. Creating a skeleton for that mesh, comprised of bones arranged in a hierarchy.
3. For each vertex of the mesh, defining which bones influence the vertex and how strong the influence is.
4. Defining animation sequences for the bones.

Link to Wiki Page: Skeleton

5.7. Attachment Points

Each mesh file may have zero or more attachment points, which can be used within the game to identify a specific position and orientation in 3D space relative to the mesh. Attachment point names should be prefixed with "a." (eg. "a.light") to avoid confusion with other nodes in the scene hierarchy. Attachment points may be used for a number of purposes, including:

• From within a config.txt file, to identify where a specific effect is to be attached to a mesh.
• From within a config.txt file mesh-table element when attaching to a parent mesh, to specify where on the parent mesh the origin of the child mesh will attach.
• From within the game code, as one of the Vehicle Attachment Points predefined for use in Trainz.
• From within an asset's script for various purposes.

Link to Wiki Page: Attachment Points

5.8. Level of Detail

Level of Detail (LOD) refers to a technique where several variants of an object are created, ranging from "high quality and high performance cost" through to "low quality and low performance cost". Each variant should look as close to possible as identical with the exception that overall object quality is lost as you go down the scale. These variants are then packaged together as a single asset with instructions to the game engine to use the most appropriate variant based on how far the object is zoomed out on the screen. When the player is close to the object, the "high detail LOD" variant will be displayed, showing the object in its full glory. As the player moves away from the object, the game can swap from the detailed version to the lower-detail-but-faster-to-render variant since the player is too far away to see the details anyway.

Sub-Topics:

• LOD in Textures
• LOD in Meshes
• Related WiKi pages, Developer Blog Articles and Useful Trainz Forum Threads

Link to Wiki Page: Level_of_Detail

Link to Wiki Page: LOD_Example

6. 3D Modeling in Trainz

The Trainz Wiki Modeling Home Page provides a good starting point for your journey to explore 3D modelling for Trainz. It lists a series of links to introductory pages and high-level concepts that are an important part of creating content for Trainz.

While you can work through the Wiki at your own pace, clicking on the various links, we recommend that you can work through the CCG Chapters one by one to discover each of the links in order from "beginner" to "expert".

Link to Wiki Page: Modeling Home Page

6.1. Modeling Guidelines

This topic provides a framework for considering performance when creating new models. Blindly making a model without considering performance is likely to lead to poor results, both in terms of frame rate, and in terms of visual quality. People often assume that better-looking objects will be inherently poor performers. This is not a good general-case summary- in practice, the better that objects perform, the more detail can be used in a scene. With this in mind, we can see how an object built with performance in mind can actually provide significantly better visual quality than an object built for visuals alone.

It's also worth considering that polygon count and performance are not the only trade-offs involved when modeling. Modeling techniques and context often prove a significantly larger effect on frame rate than the raw polygon count numbers.

Sub-Topics:

- Overview
- Where is the object used?
- Performance Impacts
- Performance Techniques
- Poor Performers

See Also:

TRS18 - Art Recommendations

Link to Trainz Wiki: Modeling_Guidelines

6.2. Modeling Scenery

"Scenery" is the Trainz term for hand-placed modelled objects. Scenery types are used for modeling discreet objects such as buildings, stations, foliage and custom rock-faces. The route creator positions each scenery object individually in the route. By default, Trainz will place the scenery origin at ground height, but the creator may choose to embed the object into the ground or suspend it in the air.

Sub-Topics:

• Orientation
• Animation
• Level Of Detail
• Materials
• Night Mesh
• Effects
• Typical Polygon Counts
• References

Link to Trainz Wiki: Modeling_Scenery

6.3. Modeling Splines

The second major primitive type in Trainz modeling is the "spline". Spline types are used for modeling continuous objects such as track, bridges, roads, hedges, fences and standalone walls. The route creator initially specifies a number of points along the spline. Trainz will then take the geometry that you supply, and bend it to fit a curve that travels through each of the points. Your model geometry may also be adjusted to take into account variations in the ground height, super-elevation, etc.

Sub-Topics:

- Orientation
- Model Variations
- Bending
- Symmetry
- Materials
- Typical Polygon Counts
- Level Of Detail Tree
- References

Link to Trainz Wiki: Modeling_Splines

6.4. Modeling Trains

As Trainz has advanced over the years it's features have been constantly expanded and refined. While the trains seen in the game were once reasonably simple in terms of geometry, resolution, lighting and configuration; today they are much more complex and their setup requires greater consideration.

Following is an outline of the fundamental technical and style recommendations for modeling Trainz locomotives and rolling stock.

Sub-Topics:

- Introduction
- Unit Scale
- Material Format
- Opaque Materials
- Opaque Material Components
- Semi Opaque Materials
- Locomotive Components
- Modeling Locomotives
- Modeling Interior Views
- Coach Components
- Modeling Coaches
- Level Of Detail

Link to Wiki Page: Modelling Trainz for Trainz

6.5. ACS Coupling System

The ACS System is an extension of the BlueStar coupling standard for TRS models (effectively BlueStar V2) and provides for the effect of working couplings on rolling stock. From version 5 of the ACSLib ACS vehicles are compatible with older BlueStar/CoupleStar ones.

Link to Wiki Page: ACS Coupling System – How to create models that use working couplings

6.6. Latest Techniques (TRS18)

With the introduction of new materials and techinques, some of the wiki pages may now be out of date. Please ensure you read the chapter TRS18 - New Features for an overview of what has changed, and well as the following chapters:

8. Understanding Rendering
11. Understanding Material Types
12. Analysing Performance

Link to Wiki Page: TRS18 Content Creation

7. Exporting Assets

This Chapter provides tutorials for exporting from both 3DS Max and from Blender. There is useful information in both tutorials, so we recommend you read through both tutorials, even if you are using the "other" modelling package.

7.1. Exporting Assets Using 3DS Max

This page describes how to export from the Autodesk 3DS Max modelling software to FBX file format for use in Trainz. Please note that the technologies described on this page are not owned or controlled by N3V Games and may be subject to change at the whim of the owner.

The techniques described here require the use of 3DS Max 2011 or newer.

Trainz supports the FBX file format in assets from trainz-build 4.5 onward.

Link to Wiki Page: HowTo/Export from 3DS Max using FBX

7.2. Exporting Assets Using Blender

This guide provides information for the export of FilmBoX (FBX) files from Blender.

Trainz supports the FBX file format in assets from trainz-build 4.5 onward.

Link to Wiki Page: HowTo/Export from Blender using FBX

8. Understanding Rendering

The Chapter provides information on the technical aspects of rendering objects to your screen.

8.1. Draw Call

A Draw Call is the act of submitting geometry to the GPU for rendering. There is a strong correlation between the number of draw calls used to render a scene and the overall scene performance.

Draw calls are not directly controlled by the content creator, but various content creation techniques and detail settings can directly or indirectly affect the number of draw calls required.

Link to Wiki Page: Draw_call

8.1.1. Mitigating Draw Call Overhead

There are a variety of techniques used internally by E2 to attempt to mitigate the draw call overhead:

• Mesh stitching and hardware instancing, where applicable, each help collapse a large number of draw calls into a single draw call.
• Individual draw calls which share a single material may be grouped together to reduce state changes.

Link to Wiki Page: Draw_call

8.2. Render Chunk

A render chunk, or simply "chunk", is the encapsulated data required to issue a single* draw call. This includes a material, a vertex buffer, an index buffer and a small amount of metadata. A 3D mesh is comprised of a number of render chunks. The render chunk's material may be shared with other render chunks in the same mesh or other meshes in the same asset.

Render chunks are a data packet (ie. noun) rather than an action on the GPU (ie. verb). The act of submitting a render chunk to the GPU for rendering is known as a draw call.

Link to Wiki Page: Render_chunk

8.3. Vertex Buffer

A vertex buffer is a component of a render chunk which stores an array of vertices. A vertex buffer in E2 is limited to 65k vertices; any mesh which is larger than this will be split into additional render chunks, each with their own vertex buffer.

Sub-Topics:

Vertex Format
Correlation with Polygons
Correlation with Indices
Impact on Performance

Link to Wiki Page: Vertex_buffer

8.4. Polygon

A polygon (also "primitive" or "triangle") is the most basic element of a 3D mesh. Each polygon is defined by three indices, each of which specifies a single vertex.

Sub-Topics:

Correlation with Indices and Vertices
Impact on Performance

Link to Wiki Page: Polygon

8.5. Index buffer

An index buffer is a component of a render chunk which stores an array of indices. Each "index" is an integer which indexes into the corresponding vertex buffer. Each triplet of indices in the index buffer defines a single polygon (also known as a "triangle" or "primitive"). This indirect technique allows multiple polygons to share vertices- this is a significant performance saving as a single vertex consumes far more memory (and memory bandwidth) than a single index, and a shared vertex only needs to pass through the vertex shader once.

Sub-Topics:

Correlation with Polygons
Impact on Performance

Link to Wiki Page: Index_buffer

8.6. Overdraw

Overdraw is a rendering term which refers to cases where the same output fragment is rendered multiple times. This occurs when multiple polygons in the scene overlap the same fragment(s).

There are two possible outcomes from any given instance of overdraw:

- The fragment is rendered. The full fragment processing cost is paid.
  a. Especially for high-end materials such as those which use Parallax Occlusion Mapping, this cost may be significant.
  b. But don't overlook this cost, even for lighter materials. It adds up fast.
- The fragment is z-culled. A lesser cost is paid.

The content creator has little say in which outcome occurs for any given instance of overdraw. This is very dependent on optimizations in the Engine (which will do its best, but which priorities correct results over fast outcomes) and the GPU hardware (which varies per manufacturer and per hardware generation).

Regardless of which outcome results, overdraw is bad. It wastes a lot of GPU time (which is typically the major bottleneck to per-frame performance) on processing something which will typically make no user-visible difference to the scene. It is typically worth incurring small costs elsewhere (for example, small increases to object polygon count) to reduce overdraw.

Sub-Topics:

Specific Instances

Link to Wiki Page: Overdraw

8.7. Alpha masking

Alpha masking refers to a special case of alpha blending where all pixels are rendered as either fully opaque (100% alpha) or fully transparent (0% alpha) with no partial transparency. This can be treated as a special case because it does not require alpha sorting in order to produce visually correct results.

On most hardware, this provides a significant performance win over an equivalent alpha sorted material. On some hardware, this is substantially slower than an opaque material. This technique also guarantees correct visual outcomes, whereas alpha sorting is inevitably an approximation which will show incorrect results in certain edge cases.

Sub-Topics:

Materials which support alpha masking
Control of alpha masking via texture.txt file

Link to Wiki Page: Alpha-mask

8.8. Alpha blending

Alpha blending is a technique where a model is rendered as partially transparent. Without alpha blending, each screen fragment affected by a given model is simply overwritten by the model's color at that location. With alpha blending, a blend occurs between the fragment's original coloration and the model's color at that location. This blend is controlled by an opacity strength value, known as "alpha".

There are two typical sources used to determine alpha while rendering; the first is a per-object alpha value set at runtime. The second is a per-texel value set in the albedo map. Each value ranges from 0.0 (fully transparent) to 1.0 (fully opaque.) These are multiplied together to give a per-fragment alpha value used for blending.

Sub-Topics:

Order dependency

Link to Wiki Page: Alpha_blending

8.9. Fragment

A fragment is a rendering concept which is roughly analogous to a single pixel on the output display (as opposed to a texel on an input texture). In practice, there is not a 1:1 mapping between fragments and device pixels for a large variety of reasons, the most obvious of which is anti-aliasing (which introduces additional fragments for some pixels).

When rendering a given primitive, the fragment shader is run once per fragment hit (with the possible exception of fragments that are early z-culled).

When a fragment is hit by multiple overlapping primitives, overdraw occurs. This is considered a waste of GPU performance and should be minimized.

Link to Wiki Page: Fragment

8.10. Hardware instancing

Hardware instancing is a technique which allows the GPU to render multiple identical meshes in a single draw call, varying a limited amount of state (such as position and orientation) for each mesh. This is used to reduce the cost of drawing a moderate number of identical low-to-mid-detail meshes. Where mesh stitching is possible, it is currently used in preference to hardware instancing. Strictly speaking, instancing is performed per render chunk rather than per mesh.

Link to Wiki Page: Hardware_instancing

9. TRS18 - New Features

TRS18 is the working title of N3V Games' next product in the Trainz simulator series. While not yet formally announced, certain content creation features targeted at this product are being documented in this Guide, and on the Trainz Wiki to allow early understanding of the features we plan on delivering.

Please note that this feature list and any images or descriptions of the features are considered work-in-progress and may be changed prior to any product release.

• Physically Based Rendering
• Asynchronous Route Streaming (Introduced in TANE SP2)
• FBX file format (Introduced in TANE SP2)
• Parallax Occlusion Mapping

Link to Wiki Page: TRS18 Overview

9.1. Physically Based Rendering

Physically Based Rendering (PBR) is a collection of guiding principles under which a fragment shading pipeline should operate to provide current-generation lighting results. Many guides exist online which explain the concept in far more detail than this document. Simplified to the extreme, this can be summarised as follows:

• Conservation of Energy. The maximum level of light output may not exceed the level of light input. Material inputs and lighting equations must be defined in terms which do not violate this principle.
• Lighting calculations are performed in linear space, but inputs and outputs are defined in sRGB space
TRS18 introduces PBR to Trainz and therefore content creators wanting to take advantage of these new features need to learn a new PBR workflow.

Developer Diary Blog: Introduction to PBR

Link to Wiki Page: Physically_Based_Rendering

9.2. Asynchronous Route Streaming

Asynchronous Route Streaming is a pre-release feature of Trainz. It has been discussed publicly in the TrainzDev forums and is summarised on this page, however users should understand that no release schedule has been provided for this feature. All discussions and details are speculative in nature and may not accurately reflect the final form of this feature.

TANE SP2 contains updated script APIs and file formats necessary for content to support this feature, however the streaming feature itself is not active in TANE SP2.

Link to Wiki Page: Asynchronous_Route_Streaming

9.3. FBX file format

The FBX file format is a proprietary file format owned by Autodesk, Inc. but used as a common interchange format for many products. As of trainz-build 4.5, Trainz uses FBX as a mesh import format.

Sub-Topics:

- Creating an FBX File
- Importing from FBX

Link to Wiki Page: FBX_file_format

9.4. Parallax Occlusion Mapping

Parallax Occlusion Mapping (POM) is a high-end GPU fragment shading technique which effectively adds a true height value to each texel.

Sub-Topics:

• Normal Mapping
• Parallax Occlusion Mapping
• Heightmap Data
• Performance
• Artifacts
• Samples

Link to Wiki Page: Parallax_Occlusion_Map

9.5. Effect Layer

Effect Layers are a new TRS18 feature which offer the ability to use Surveyor to paint various effects beyond the traditional ground height and ground texture. Each effect layer typically offers a number of configuration options, including a configurable data density so that route creators can tune the outcome to their specific requirements, dedicating performance to areas which are important to their route while lowering detail and/or flexibility in other less-important areas in order to maintain high overall performance.

Effect Layer Types

The current effect layer types are supported:

Clutter Effect Layer - for ground clutter such as rocks and foliage.
TurfFX Effect Layer - for various types of grass and grass-like flora.

Link to Wiki Page: Effect_Layer

10. TRS18 - Art Recommendations

This Wiki Page provides guidance for content creators working to the higher levels of quality and performance we are aiming at for TRS18.

Sub-topics:

• Modeling Overview
• Route-Building Overview
• Trains
• Flagship Scenery
• Large Scenery
• Trees
• Minor Scenery
• Detail Scenery
• Ground Textures
• Splines

Link to Wiki Page: Art_Recommendations

11. Understanding TRS18 Material Types

The Material Types Wiki Page provides guidance for content creators on the new Materials as well as Legacy materials.

Materials describe how a modelled surface will appear in the game. They specify a range of configurations and resources including classic material properties, textures and shaders. Each Material is an instance of a particular preset Material Type, from the selection described on this page.

Sub-topics:

- Material Naming
- PBR Materials
- Legacy Materials
- Material Examples

See Also:

Physically Based Rendering

Link to Wiki Page: Material Types

11.1. m.pbrmetal

m.pbrmetal is the most basic PBR-based material type used in Trainz, and should be the default material type of choice for any mesh unless some specific feature of some other material type is required. All PBR materials are defined using a multitude of texture channels (not just a simple RGB image) which define various aspects of the material on a per-texel basis. One major advantage of this approach is that a single in-game material can be used to display various different real-world materials, rather than requiring multiple separate in-game materials with different parameters. Trainz uses a Metallic/Roughness style PBR workflow.

Link to Wiki Page: m.pbrmetal

11.2. m.pbrmetalmasked

m.pbrmetalmasked is a minor variation on the m.pbrmetal material. It adds a masked alpha channel to allow a complex outline to be created without the use of large numbers of polygons. All PBR materials are defined using a multitude of texture channels (not just a simple RGB image) which define various aspects of the material on a per-texel basis. One major advantage of this approach is that a single in-game material can be used to display various different real-world materials, rather than requiring multiple separate in-game materials with different parameters. Trainz uses a Metallic/Roughness style PBR workflow.

Link to Wiki Page: m.pbrmetalmasked

11.3. m.pbrmetaldetail

m.pbrmetaldetail is a variation on the m.pbrmetal material. It adds a 'detail map' which is often used to provide fine surface texture repeated across the entire material. All PBR materials are defined using a multitude of texture channels (not just a simple RGB image) which define various aspects of the material on a per-texel basis. One major advantage of this approach is that a single in-game material can be used to display various different real-world materials, rather than requiring multiple separate in-game materials with different parameters. Trainz uses a Metallic/Roughness style PBR workflow.

Link to Wiki Page: m.pbrmetaldetail

11.4. m.clutter

m.clutter is a variation on the m.pbrmetalmasked material. It adds controlled fade distances which are intended to allow cross-fading between LODs. It does not include support for parallax mapping. All PBR materials are defined using a multitude of texture channels (not just a simple RGB image) which define various aspects of the material on a per-texel basis. One major advantage of this approach is that a single in-game material can be used to display various different real-world materials, rather than requiring multiple separate in-game materials with different parameters. Trainz uses a Metallic/Roughness style PBR workflow.

This pre-release material is intended for use with the Clutter Effect Layer, and will be tuned to suit that purpose to the detriment of any other possible use-case. While it is currently expected that the material will be available to other asset types, this availability is not locked in at the current time.

Link to Wiki Page: m.clutter

11.5. KIND Groundtexture

KIND Groundtexture is an extension of KIND Texture. A ground texture is tiled in Surveyor to color and cover the base grid.

Link to Wiki Page: KIND Groundtexture

11.6. Legacy Material Types

The material types here are intended for backwards-compatibility with older versions of Trainz. They are not PBR materials and offer the content creator less control over the end result. It is recommended that the PBR materials are used instead.

Link to Wiki Page: Legacy Material Types

11.7. Material sharing

There are numerous benefits to material sharing. Some are important to understand, while others just work in the background to give minor speed boosts. Each render chunk in a mesh file references a single material. It is beneficial for performance reasons to reduce the number of materials in use. To this end, it is possible to share a single material between multiple render chunks within a single asset.

Link to Wiki Page: Material_sharing

12. Analysing Performance

The Perfomance Wiki Page discusses performance from a content creation perspective. It seeks to enumerate the major factors which reduce game frame rate and reference the major strategies in avoiding problems. This document does not describe best practices (see the Art Recommendations document for that) but instead discusses how things work internally and how that affects performance.

Sub-topics:

• Performance Concerns
• The Low Level
• Multithreading
• Game Main Thread
• Per-frame Costs
• One-off Costs
• Loading
• PhysX Thread
• GPU
• Settings
• Profiling

Link to Wiki Page: Performance Wiki Page:

12.1. Runtime Performance Statistics

Trainz provides a simple tool for monitoring asset performance in Driver or Surveyor. Poor-performing assets can negatively affect both the frame rate and the available draw distance.

Link to Wiki Page: Runtime_Performance_Statistics

12.2. Asset Preview

The TANE Asset Preview Interface is used to give a quick preview of a single asset from within Trainz Content. The asset preview window captures a number of statistics which are of use to content creators. These statistics are not gathered every frame but are updated on a regular basis.

Link to Wiki Page: AssetPreview

13. How To Guides

A wide range of “How To” guides have been created for Trainz and reside on the Trainz Wiki. Here are the links to the broad category headings:

- Using “Manage Content” in TANE
- Content Creation: Routes
- Content Creation: Assets
- Content Creation: Scripting
- Creating and Configuring Content In-game

Click on the relevant link and from there you can find the relevant link in the wiki. Also note that some of these guides may be out of date for more recent versions of Trainz.

Link to Wiki Page: “How To” guides

13.1. Using “Manage Content” in TANE

Using “Manage Content” in TANE

Download an Asset from the Download Station
Configure and use Columns in content windows
Use the “Status” column in content windows
Edit and Submit content
View by thumbnail images
Fix Errors and Warnings
Use Search Filters
Find Missing Dependencies

Link to Wiki Page: Using_Manage_Content_in_TANE

13.2. Content Creation: Routes

Content Creation: Routes

Create a Route in 10 Minutes
Add Bumpy Track
Make a Track Cutting
Place Signals
Use Superelevation
Use Trig Stations
Configure an iPortal
Configure An Interlocking Tower

Link to Wiki Page: Content_Creation:_Routes

13.3. Content Creation: Assets

Content Creation: Assets

Make your first Trainz Asset
Export from 3DS Max using FBX
Export from Blender using FBX
Solve Blender Modelling Problems
Scaling setup in Blender
Build normal maps in Max
Localize an Asset
Build a scripted asset
Reskin an Asset
Reskin an Asset using an alias
Build a Bridge
Build a Tunnel
Build a Crossing
Build Catenary
Build a Car for Traffic (stand-alone version)
Build a Car for Traffic (mesh library version)
Build a cab interior
Build a drivable car
Render to Texture
Build a simple animated scenery asset in Blender
Build Passenger Enabled Assets
Create a High Voltage Power Line Spline
Create transparent windows in Blender
Create a Fence Spline Featuring LOD and Random Meshes
Create a Neon Font Effect with PhotoShop C4
Build an animated Neon Sign
Configure a spline for collisions
Build Procedural Track for T:ANE
Update your auto-generated next KUID
Create a displacement map

Link to Wiki Page: Content_Creation:_Assets

13.4. Content Creation: Scripting

Content Creation: Scripting

- Your first Trainz Script
- Hide- & Unhide Meshes

Link to Wiki Page: Content_Creation:_Scripting

13.5. Creating and Configuring Content In-game

• Create an Interactive Session
• Add Carz Traffic to a Route
• Tune a Steam Locomotive Enginespec
• Configure an Interactive Industry
• Implement a Destination Sign System
• Upload a new asset to the DownloadStation
• Update an existing asset on the DownloadStation

Link to Wiki Page: Creating_and_Configuring_Content_In-game

14. Getting Started in TrainzScript

This set of "Getting Started" tutorials on Trainz Script, written by Andi06 are written for older versions of Trainz. They will give you a good foundation for what is possible using TrainzScript, but there are new techniques required to deal with asset streaming in TRS18 and beyond.

- Hello World, Setting Up An Asset Script
- Getting Information From the Game, Writing Your Own Methods
- Talking to Yourself, Sending Messages
- Saving and Restoring Data, Using Properties
- Setting Up the Asset
- Automating Animations
- Handling Corona Effects
- Playing Sounds
- Hiding Meshes
- Handling Name Effects
- Texture Replacement

TrainzScript Wiki Reference Materials

For detailed information on TrainzScript and all the classes available, please follow these two links:

- Scripting
- Script Classes

15. Reference Materials

Following is a list of important reference materials found on the Trainz Wiki:

- Asset Validation
- Content Configuration
- Terminology
- Scripting
- Session Rules By Categories With Properties

16. Other Reference Materials

TRS2004 CCG (todo - find originals and host (perhaps in wikibooks page)
TRS2006 CCG
Trainz Classics Content Creation Guide

17. Link to Trainz Wiki Main Categories

The following are the main Categories used in the Trainz WIki. These are the "super-highways" of the various topics contained inthe wiki.

- Config Container
- Content Creation
- Content Configuration
- File formats
- Help
- How-to Guides
- Kind Engine Containers
- Material types
- Modeling
- Normal mapping
- Rules
- Texture mapping
- TrainzScript
- Trainz Versions
- Validation Errors

17.1. Category:Config_Container

Config Container

17.2. Category:Content_creation

Content Creation

17.3. Category:Content_configuration

Content Configuration

17.4. Category:File_formats

File formats

17.5. Category:Help

Help

17.6. Category:How-to_guides

How-to Guides

17.7. Category:Kind_Engine_containers

Kind Engine Containers

17.8. Category:Material_types

Material types

17.9. Category:Modeling

Modeling

17.10. Category:Normal_mapping

Normal mapping

17.11. Category:Rules

Rules

17.12. Category:Texture_mapping

Texture mapping

17.13. Category:TrainzScript

TrainzScript

17.14. Category:Trainz_Versions

Trainz Versions

17.15. Category:Validation_Errors

Validation Errors

18. Links to External Websites

The following sub-topics provide links and information regarding various external resources related to Trainz.

18.1. Trainz Forums

• Content Creation Support Forum
• Forum Articles
• General Trainz Forum
• TANE Support Forum
• TrainzDev Forum (Home for community feedback on the latest development tools and builds. Posting by invitation only.)
•

18.2. Trainz General Support

Support.TrainzPortal.com

18.3. 3rd Party Content Creator Websites

This Chapter provides a [incomplete] list of some of the more popular external sites realting to content creation for Trainz,

- Wikibooks Tutorial_for_Blender
- Blender Tutorial by Doug56
- Hack's list of stuff for new content creators

Table of contents