open rails

BLENDER CONTENT CREATION NOTEBOOK

A guide to making content for Open Rails

Pete Willard

Version .0.5, 2023-08-10

Content Creation Notebook

Open Rails, Blender Content Creator Notebook . . . 2
About This Document . . . 3
- About The Author . . . 3
- Contributor and Resources . . . 3
- Some References . . . 4

Conventions Used . . . 5
Preface . . . 6
- Here is the list of things that I assume about the reader. . . . 6
- Organization of This Document . . . 6

Making Content for Open Rails . . . 8
- Common Terms and Definitions . . . 9
- Content Generation Involves: . . . 12
  - Programs you should have: . . . 12
  - What I use: . . . 12
    - Why I Use Blender . . . 12
- Why Use Blender to Make Game Assets? . . . 13
  - Open-Source and Free . . . 13
  - Versatility . . . 13
  - Integrated Workflow . . . 13
  - Asset Optimization . . . 13
  - Community and Resources . . . 13
  - Non-Destructive Workflow . . . 13
  - Constant Development . . . 13
  - What I Can Recommend: . . . 13
  - How to Install Blender . . . 14
  - Setting up our Blender Environment . . . 16
    - Interface . . . 17
    - Themes-3D Viewport . . . 19
    - Viewport . . . 19
    - System . . . 20
    - Save & Load . . . 21
    - File Paths . . . 22
    - Add-Ons . . . 24
      - Enabling and Disabling Add-Ons . . . 24
      - 3rd Party Add-Ons . . . 26
    - Exit and Save Preferences . . . 27
  - Other Settings . . . 28
    - TABS . . . 28
    - Scene Settings . . . 28
    - Gismo settings . . . 29
    - Overlays . . . 29
    - Shading Settings . . . 30
    - Units . . . 31
    - Scene (Rendering) . . . 32
    - Shading . . . 33
  - Done, for now . . . 34
- Content Creation Overview . . . 37
  - Orientation . . . 37
  - Engine/Wagon Model Hierarchy . . . 37

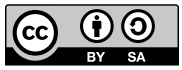
Standard 2 Axle Freight Bogies . . . 38
Isolated Axles . . . 38
Configuration Files . . . 39
Various General Notes about Content Creation . . . 39
General Modelling Standards from Erick Cantu . . . 41
Erick Says: . . . 42
Additional guidance from Blender Brothers . . . 44
Blender Brother's Gold Rule . . . 44
General Texture Mapping Guidelines . . . 44
Making A 3D Model . . . 46
Some Basic Things About 3D Models. . . . 46
Some Basic Things About Modeling . . . 46
Polygon Count and Vertex Density . . . 46
Preparation of Work . . . 47
Background Images . . . 47
Some things I will miss about using 3D Canvas . . . 50
Where to start? . . . 53
Do we need a quick 3D Modeling primer? . . . 53
The Modeling Interface . . . 54
Shortcut Keys . . . 56
View Controls . . . 56
Modeling Modes . . . 59
The 4 major edit-mode tools you are likely to use the most are listed below. . . . 61
EXTRUDE . . . 61
INSET . . . 62
BEVEL . . . 64
LOOP CUT & SLIDE . . . 65
Additional Tips . . . 67
MIRROR and FLIP . . . 67
Additional Modeling Tips . . . 67
Setting up your Initial Workspace . . . 67
Actually Modeling Something . . . 70
Model Building Exercise #1 . . . 70
Texturing . . . 82
Let's Begin Texturing . . . 82
Creating Level Of Detail Distance levels . . . 89
Guidelines for creating LOD versions . . . 89
Building a Library of Reusable Parts . . . 91
A Brute-Force Library solution . . . 91
Importing Existing Objects . . . 94
Fixing UV MAP Assignment Issues on Imported Objects so they work with the MSTS exporter . . . 95
Model Exercise #2 . . . 97
Intermediate Modeling . . . 97
Building the General Shape . . . 97
Importing the Background Images . . . 98
Setup . . . 98
Making the Interior . . . 103
Making the Doors . . . 104
Making a Vehicle . . . 109
Finding the Right Reference Images . . . 109
Organizing Your Reference Material . . . 110
Using Background Images . . . 110

The Modeling Process 111
Iteration and Refinement 112
Let's do it. 112
Making Rolling Stock - Project #4 112
Using Background Images 112
Starting from the ground up 112
Vehicle LOD - Level of Detail Settings 112
Bitmap Editing 115
Layered Format Files 115
Using DDS Textures 116
Preparing a texture 117
Decals 118
Coding alternatives 118
Layering Basics 122
Grime Layer 122
Rust Layer 122
Dirt Layer 122
Dust Layer 122
Finalization of effects 122
Applying Fonts and Lettering 123
Layering Tips from Erick 123
How to Make Night Textures 126
How to Make Night Textures with Backlighting 127
How to Make Snow Textures 128
Highlights and Shadows 131
Generated Shadows 132
More About Baking Ambient Oclusion 139
clusion Steps 139
Introduction to Blender Scripting 140
Actually Doing It 140
Blender and ChatGPT 141
Incorporating ChatGPT into your scripts 142
Getting an API Key from OPENAI 142
API costs 143
Running a Blender Python Script Inside Blender 143
A Collection of Scripts 144
Fixing Material Problems 144
Making copies 145
Make Selected Objects Become ASSETS 146
Automating the Boring Stuff with Python 147
So what are we going to do? 147
Creating Rolling Stock and Locomotives 154
Open Rails Units of Measure 155
Using *INCLUDE* 156
Basic ENG and WAG file Details 157
Should there be a ENG/WAG standard layout? 157
Universal Settings 158
Example WAG file with INCLUDE 159
Option 1 - The Benefits of Common Folders for Re-skinners 165
Option 2 - Completely New Content (with possible help from NAVS) 165
Gathering Details 166
Another Example of Rolling Stock 169

Specifying lights on locomotives and wagons . . . 171
Introduction . . . 171
Available Tokens for Lights . . . 171
Setting requirements . . . 171
Defining lights . . . 172
Fade-in and Fade-out times . . . 173
Cycle . . . 173
States . . . 173
Examples . . . 174
Example 1 – the Dash 9's front headlights . . . 174
Example 2 – the Dash 9's front right flashing (ditch) light. . . . 175
Example 3 – the Dash 9's rear red light . . . 176
A brief explanation of hexadecimal numbers . . . 177
Effects . . . 178
Freight Animations . . . 178
Creating 3D Cabs . . . 180
Appendix A . . . 194
Appendix II . . . 205
Consider Transitioning away from MSTS to Open Rails . . . 205
The ORTS Program . . . 205
Index . . . 207

Open Rails, Blender Content Creator Notebook

2023-08-10 2.3.0.5 Pete Willard



Open Rails, Blender Content Creator Notebook

About This Document

This guide covers the use of Blender 2.8, including Blender 3.6 LTS or higher for creating content for Open Rails. It is recommended to use the latest Long Term Support (LTS) release of Blender for the best compatibility with the instructions provided in this document. Please note that older versions of Blender will not be covered or supported in this guide. At the time of this update, Blender Version 3.6 is the most recent release and Blender 3.6 LTS is the most recent Long Term Support release.

The Long Term Release or LTS version of Blender is preferable because there are only bug fixes applied to these releases, not new features so things like the User Interface will not change.

About The Author

The individual who initially wrote this guide is not an expert in content creation and has mostly used 3DCanvas/3D Crafter for creating content for Microsoft Train Simulator from 2002 to 2016. They have also experimented with Blender, but found earlier versions difficult to use due to its cumbersome user interface. However, they found Blender 2.8 to be a vast improvement in terms of user interface and do not plan to switch to a different 3D modeling tool in the near future.

Writing in 3rd person is weird.

Contributor and Resources

This document is made possible with the contributions and feedback from people like:

- Wayne Campbell
- Erick Cantu
- Michelle McKell
- Dave Nelson
- Peter Newell
- Curtis Holt
- Darrin Lile
- Josh Gambrell
- and various Blender add-on authors

Useful resources:

- http://www.elvastower.com
- http://trainsim.com
- http://www.textures.com You need to login to download[1]
- https://www.blendersecrets.org/ Author of many tips and tricks videos as well as a book
- https://curtisholt.online/ A really good Blender Trainer and add-on creator
- https://blender.org Where to download Blender from
- https://www.coalstonewcastle.com.au/physics/ ORTS Information
- http://www.nordicfx.net/ HDRI backgrounds
- https://polyhaven.com/ Textures and HDRI Backgrounds
- https://github.com/Stromberg90/Scripts/blob/master/Blender/Edge_To_Curve.py -Simple but handy script add-on for Pipes and Handrail creation.

Some sections of the Microsoft Train Simulator technical documents are also included here as reference material since some of these details exist no where else.

Some References

NAVS

North American V-Scale - Content created by or related to Erick Cantu. Using a set of standards he developed.

Coals to Newcastle

Meaning, "A silly venture", as Newcastle has plenty of coal. Also, A website run by Peter Newell that has really good Open Rails Train Simulator Physics documentation.

[1] The site offers Free Registration and free downloads once signed up

Conventions Used

Regular Note.

Pay attention to these.

You should know this.

With care, you can succeed.

Optional, but good to know.

`Highlighting`

BOLD

Italic

```
Source code
```

`LMB`
: Left Mouse Button

`MMB`
: Middle Mouse Button

`RMB`
: Right Mouse Button

N-Panel
: Number Panel, `N` Hotkey in Main 3D Window

[ENTER]
: Sometimes you will see KEYBOARD entry look like this

Referenced footnotes appear at the **end** of Chapters

Web Links should be active and will open in your web browser if your PDF reader supports it.

Preface

OpenRails is an open-source train simulator platform designed to replicate the experience of operating trains on various railway systems. It was developed as an alternative to Microsoft Train Simulator (MSTS) and is compatible with MSTS content, allowing users to enjoy a wide range of routes, locomotives, and rolling stock created for MSTS.

Here are some key points about OpenRails:

Compatibility: OpenRails is built to run on modern operating systems such as Windows 7, 8, and 10. It offers improved stability and performance compared to MSTS, especially on newer hardware.

Enhanced Graphics: OpenRails supports a variety of graphical improvements, including higher screen resolutions, improved lighting and shading effects, enhanced textures, and anti-aliasing. These enhancements provide a more immersive visual experience.

Physics and Realism: OpenRails aims to provide realistic train physics and behavior. It features improved simulation of train dynamics, braking, and traction control, making the virtual driving experience more authentic.

Third-Party Content: OpenRails is compatible with a wide range of third-party content created for MSTS. This includes routes, locomotives, rolling stock, scenery objects, and other add-ons developed by the MSTS community. Users can install these addons in OpenRails and enjoy them with improved performance and visual quality.

Community and Development: OpenRails benefits from an active and passionate community of developers and enthusiasts who contribute to its development. Regular updates and improvements are made to the simulator, ensuring its continued growth and refinement.

Open-Source Nature: OpenRails is an open-source project, which means its source code is freely available for modification and enhancement. This allows developers to contribute to the project and create new features or fix issues.

It's worth noting that OpenRails is not a standalone game but rather a simulation platform. While it provides a framework for train simulation, it relies on third-party content and addons to provide the routes, trains, and other elements of the virtual railway environment.

Overall, OpenRails offers a robust and customizable train simulation experience for enthusiasts who enjoy operating trains on various virtual railway systems.

Here is the list of things that I assume about the reader.

- You are new to Blender
- You are reasonably new to content creation for Open Rails
- You have Open Rails installed and working
- You have installed TSRE5 for Consist management
- You have a texture file editor that supports PSD, PNG and DDS formats.[2]
- You have some idea about how Open Rails and Microsoft Train Simulator content is created
- You have a project in mind, but you understand that you will not create your Magnum Opus on a first try.

If you have prior experience with GMAX, Train Sim Modeler or 3D Canvas, it will be a plus, but it is not required.

Organization of This Document

As the title implies, this document is based on a collection of notes I made while learning to make content for Open Rails. As a result, it may not be the most organized or useful document on the subjects discussed within it but it is primarily aimed at filling a void in the available documentation set for Open Rails. This document is FREE and Open Source. It will ultimately live on in a GITHUB repository created for it. This means that others are welcome to fork a copy, edit it to add sections or fix errors, if desired. I, the author, do not claim unrealistic ownership of the ideas and concepts contained within. In fact, I struggle to consider myself the sole author of this manual at this point. Of course, at the moment, I am entirely responsible for its content, but I have created, gathered, reorganized and rephrased the contents as needed since

it comes from many different sources.

Due to the way this document originated, it may seem to jump around and be repetitive at times. My apologies in advance if this bothers you but if you are so inspired, you are welcome to help fixing any errors, problems or omissions.

[2] A good example is the Paint.net, KRITA or GIMP all of which are free

Making Content for Open Rails

About ORTS

The Open Rails Train Simulator is an open source project that exists primarily because the promised Microsoft Train Simulator 2 never happened. Open Rails strives to be backwardly compatible with nearly all existing MSTS content while adding many new features and corrects old issues that existed in the Microsoft/Kuju releases. Very often, you can use MSTS and ORTS terms interchangeably, but keep in mind that ORTS can handle higher poly count (more complex) models than MSTS can as well as having many other changes added that are unique to Open Rails.

> The key advantage that Open Rails currently offers over Microsoft Train Simulator is that good frame rates can be maintained with a much higher number of polygons, so curves can be smoother and more detail can be modelled. Open Rails also displays 32-bit color (whereas Microsoft Train Simulator is limited to 16-bit). Another advantage is the longer viewing distances, adjustable from 2km out to 10km. — *from OpenRails.org website*

Since this document will be about using Blender, other 3d modeling tools will not be mentioned much unless it's for a point of contrast.

Common Terms and Definitions

ALPHA

A bit map with a portion of the image that is masked-out, making the location appear see-though in the image.

DISTANCE LEVELS

A range of shapes originating from the one main shape allowing polygon count to be kept low at further viewing distances. Often referred to as Level of Detail, or LOD.

EDGE

A line created by the linking of 2 vertices (known as a "2-dimensional object")

ENG FILE

A file that describes to the Simulator the details of an Asset to be used as an ENGINE within Open Rails. It contains physical details such as Motive and Braking values as well as Coupler type, etc.

FACE

A surface created by 3 or more vertices.

FOOBIE

A model made specifically lacking in detail to allow many variants to be made using a single base model, relying on the TEXTURE to define the details and not the actual 3D objects. "Foobie" is a contraction of "fake boobies".

HARD SURFACE MODELING

A generalized term for modeling objects that represent things like vehicles, buildings, weapons, roads. These make use of the more standard mesh techniques and objects like cube, cylinder, plane, cone, torus etc.

HDRI

HDRI is short for High Dynamic Range Image. Digital cameras only have a limited dynamic range â€” that's why some areas of a photo appear darker than they do in real life. HDRIs give photo editors a chance to brighten the corners and create an image that looks more natural. Environment maps or HDRI maps are one of the most efficient and quickest way to light your 3D scene and achieve realistic results in Blender. HDRIs are essentially a snapshot of the the real world lighting which contain accurate lighting detail through high dynamic range imaging (HDRI)

HIERARCHY

The assignment of parenting of objects within a shape so that an objects children will only move when they do but they can move independently too but are always affected by their parent.

LEVEL OF DETAIL

(LOD) Distance values at which the base model is replaced by reduced poly count shapes to account for the inability to see details at a distance. Higher values should have less detailed models. There are no hard rules about LOD ranges, but you should try to use them.

MANIFOLD SHAPE

A manifold shape is a 3d object that considered to be water tight. The vertices, faces and edges are aligned and connected in such a way that if you poured water into to the INSIDE of the object, none would leak out. A default `cube` primitive is an example a manifold shape.

MATERIAL

A material is assigned a texture with additional components such as lighting palette and shading.

MESH

A 3D Model "shape" made up of 1 or more "faces" without texture

NGON

Is a face with 5 or more vertices. Many Blender tools and features rely on the underlying mesh to be made of NGONS and will not work with triangulated faces.

NORMAL

A Normal is a vector in 3D-space that is perpendicular to two other vectors (determined by two edges of a face)footnote [See Wiki https //en.wikipedia.org/wiki/Normal_(geometry)] A Normal controls the facing direction of applied textures.

OBJECT

A 2-dimensional object has length and height, but no depth. Examples of 2D objects are planes, polygons and lines. A 3-dimensional object has length, height, and depth. Examples of 3D objects are cubes and spheres.

ORGANIC MODELING

Modeling techniques that use a more sculpted approach for things like animals, people, plants, trees, and even automobiles. There are specialized modeling tools and techniques for this including *SURFACE NURBS* (Non-uniform rational basis spline, Google it) and the Sculpting tab in Blender. Blender's Sculpting modes are used for organic modeling and will not be discussed in this document.

ORIGIN

See Pivot Point

PIVOT POINT

A position used to align the shape to its environment, also known as its origin.

POLY COUNT

The "poly count" is a total of the number of "polygons" used to create a 3d mesh. A polygon is made up of vertices. It is used as an indicator of how complex a model is.

POLYGON

Refers to the planar face of this closed shape, edges are the straight edges that define it, and points/vertices are where those various edges connect with one another. Blender calls these "tris", short for faces made of triangles and "quads", short for 4 vertex faces.

QUAD

Faces made up of 4 vertices

SHAPE

A three dimensional object that is finished in all aspects and ready to be placed within the game environment.

SMOOTHING

A smoothing group is a shade value that allows individual polygons to be lit by the game environment in different ways.

TRI

Faces made up of 3 vertices, also referred to as triangles

UV

UV is the 3D modeling process of projecting a 2D image to a 3D model's surface for texture mapping. The letters "U" and "V" denote the axes of the 2D texture because "X", "Y", and "Z" are already used to denote the axes of the 3D object in model space, while "W" (in addition to XYZ) is used in calculating quaternion rotations, a common operation in computer graphics.

VERTEX

(Plural = vertices) a point in 3d space with x,y,z coordinates, often used as part of a location where edges and faces meet.

WAG FILE

A file that describes to the Simulator the details of an Asset to be used as an WAGON within Open Rails. It contains physical details such as Braking values as well as Coupler type, etc.

Also, here are some design choices that you can make that you should clarify in the README section of your completed content.

Made for MSTS

Meaning it **can** run in ORTS, but it might need some editing to the WAG or ENG file since some Physics values from lagacy Microsoft Train Simulator might cause the vehicle to not perform optimally

Made For ORTS - No other files required

Meaning it **can** only run in ORTS as it is configured specifically with Open Rails configurations and parameters

Made for ORTS - but requires other add-ons

Meaning it **can** only run in ORTS as it is configured specifically with Open Rails configurations and parameters and it will need additional files downloaded to work

Made for ORTS but Compatible with MSTS

Meaning it can run in Open Rails but has been configured to also be compatible with Microsoft Train Simulator

Made for ORTS but requires MSTS

Meaning it can run in Open Rails but has been configured to also be compatible with Microsoft Train Simulator and also requires files that only come with an installation of Microsoft Train Simulator on the system

Made for ORTS but requires MSTS and other add-ons

Meaning it **can** only run in ORTS as it is configured specifically with Open Rails configurations and parameters but also requires Microsoft Train Simulator to be installed and relies on additional products.

Content Generation Involves:

- A mesh - Your 3D model
- Online resources and references (This is an area that is somewhat lacking with regards to 3D modeling for ORTS, but these include https://msts.steam4me.net/tutorials/index.html https://www.trainsim.com/vbts/forum.php and http://www.elvastower.com/forums/index.php)
- A texture - A 2-dimensional bitmap image (material) that you will apply to your model, also referred to as a 'skin'
- A UVmap of the mesh - The instructions on how to map your 2-dimensional texture to your 3d model
- A configuration file or set of files that describes your content to the simulator
- A thumbnail jpg (optional but helpful)
- Instructions - Let's not leave the work half done

Programs you should have:

- A 3D program, Blender 3.6 LTS is recommended, but even the LATEST release - currently 3.6 - should also be fine unless the **Blender's PYTHON API** has dramatically changed.
- An Exporter add-on for the MSTS/ORTS format, we will use Wayne Campbell's *S File Exporter* from Elvas Tower web site[3]
- A paint program that has channels and layers (There are multiple options here)
- A text editor that handles UNICODE files

If you are also making models for Trainz Simulator, note that the latest versions of Trainz will accept the native Blender FBX exporter file format as output.

What I use:

- 3d Modeler: Blender 3.6 LTS
- Bitmap tool (any one of these, interchangeably): Serif Affinity Photo, Serif Affinity Designer, PaintShopPro Version 7, Paint.net or Photoshop CS2
- UV/Shader tool: Blender has this built in, 3rd party options exist, however I don't use them.
- Text Editor: *Microsoft Visual Studio Code*
- Metric Conversion Calculator (I have one built into hand held calculator, but you can use Google for this)
- A Scale Calculator: Converting dimensions from a scale drawing. (there is on on my website at http://www.railsimstuff.com)
- A texture snipping tool, a) Windows has a built in Snipping tool, b) SHOEBOX, an Adobe Air Application, designed for use with game creation.

I recommend just using METRIC for measurements. I realize that some people have become very attached to their Imperial units but Metric does make for worldwide compatibility and in the end it just ends up being easier if you stick with it.

Why I Use Blender

In 2002, I started using 3D Software to create content for games, primarily for Microsoft Train Simulator, and I have worked with different tools on different projects and in the end I found Blender to now be my personal favorite. I have used Abacus Train Sim Modeler, 3D Canvas, Gmax and Sketchup and while 3D Canvas was my favorite for many years, my use of it now is only as a file format conversion tool. Blender 3.6 LTS has everything I need to be making 3D models and includes many things that 3D Canvas doesn't have, including active technical support.

[3] Download from https://github.com/pwillard/Blender_MSTS_ORTS_Exporter

Why Use Blender to Make Game Assets?

Blender is a powerful and versatile 3D modeling and animation software that has gained popularity for creating game assets, among other things. Here are some reasons why you might consider using Blender for making game assets:

Open-Source and Free

Blender is open-source software, which means it's free to use and has an active community of developers and users. This makes it accessible to a wide range of users, including hobbyists, indie game developers, and professionals.

Versatility

Blender offers a comprehensive set of tools for 3D modeling, sculpting, texturing, rigging, animation, rendering, and more. It can handle a variety of asset types, from characters and environments to props and special effects, making it a versatile choice for game development.

Integrated Workflow

Blender provides an integrated workflow, allowing you to create, edit, and animate assets seamlessly within a single application. This can save time and reduce the need to switch between different software tools.

Asset Optimization

Blender provides tools to help optimize your assets for real-time rendering in games. You can create LODs (Level of Detail), bake textures, and manage UV maps to ensure your assets are efficiently displayed in the game engine.

Community and Resources

Blender has a large and active community of users, artists, and developers. This means there are plenty of tutorials, forums, and resources available to help you learn and improve your skills.

Non-Destructive Workflow

Blender supports non-destructive modeling techniques through modifiers and procedural workflows, allowing you to make changes to your models without losing the original data.

Constant Development

Blender's development is ongoing, with new features and improvements being added regularly. This ensures that the software stays up-to-date with industry trends and user needs.

Ultimately, the choice of using Blender for making game assets depends on your specific needs, preferences, and familiarity with the software. If you're looking for a robust and cost-effective tool with a wide range of capabilities, Blender is definitely worth considering for your game asset creation workflow.

To export your model for using in MSTS or ORTS, Wayne Campbell created a very capable exporter for MSTS format `.s` files.

What I Can Recommend:

- 3D Modeler: Blender 2.93 LTS version or Blender 3.6 LTS or even 3.6 should be fine. (If you are already well accustomed to Blender 2.79 it's OK, but you will be on your own here) Did I mention that it is free?
- BitMap Tools: *Serif Affinity Photo*, *Photoshop CS2* (You can still get this for free from Adobe if you google for it) , *Paint.net* (free), *PaintShop Pro* (even version 7 still works), or download the latest version of *GIMP* for free.

- UV tool: While there are 3rd party options for this, you can just use UV and shader tools that come with Blender (You could look at Meshmixer or even SubstancePainter)
- Text Editor: *Microsoft Visual Studio Code* , *Context.exe* or *Sublime*

CONTEXT is an abandoned editor, but it has a syntax highlighter for ENG and WAG configuration files available at **steam4me.com** website. There is an early version of a context highlighter in the works for ENG/WAG files in VScode, but I've not finished it yet. https://github.com/pwillard/engwag

How to Install Blender

According to the Blender.org website, a new version is released about once per quarter. I'm going to assume you are installing Blender on a Windows 64 BIT platform. I would recommend that you download the LTS or Long Term Support version and if you are really brave, you can download the latest available stable version. Currently, this is3.6.

While Blender CAN run from a USB stick in a portable mode, it's best to just use the MSI installer.

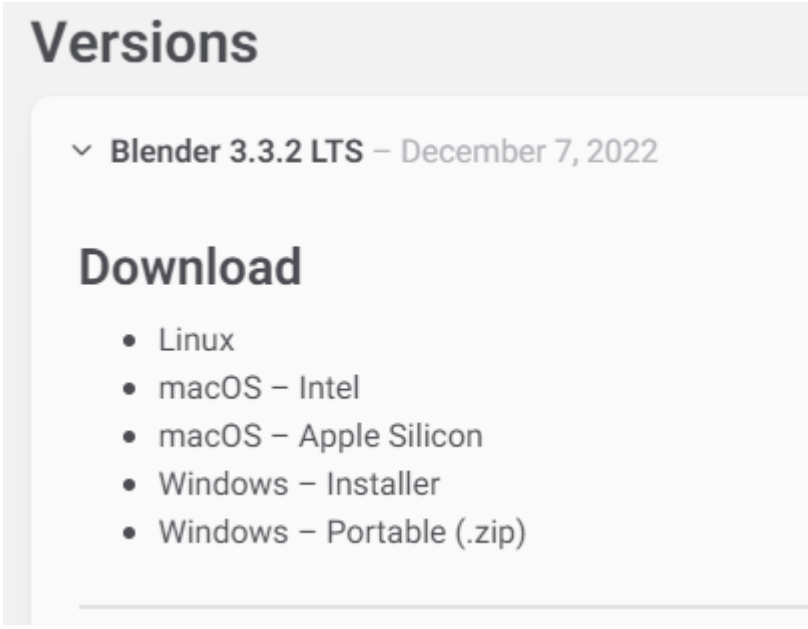
If you are still using Windows 7, you won't be able to install a Blender version 2.93 or newer. You really should upgrade. Much of the content in this document will still apply to version 2.93 except for new features.

If you install Blender for FREE from STEAM, the STEAM Library interface will auto-update to the latest version of Blender for you by default. You may not like this behavior so you are warned.

Between each version update, the Blender developers might move some user interface features around a bit. You might not see the same screens shown in the examples included in this document.

Let's use the Blender installer from the **Blender.org** website. https://Blender.org/download/

The LTS version will be a link on the page referred to on the "Looking for Long-Term Support? Get Blender 3.6 LTS". It can be found here: https://www.Blender.org/download/lts



Hopefully, Blender.org will continue its practice of creating LTS releases.

From here you see various download options available. Locate the `Installer` option and download it. Double-clicking the downloaded `.MSI` file will begin the install. The `.MSI` file does all the work.

Blender will default to using your `DOCUMENTS` folder for Models and your `%APPDATA%` folder to store program configuration data and addons under the "Blender Foundation" folder structure. Blender will use unique entries for each version of Blender installed so it is perfectly fine to have multiple versions of Blender installed on the same PC.

It does get tricky related to the *File Associations* pertaining to which version of Blender will open when you click on a **.Blend** file. The default will become whichever version of Blender you most recently installed. Be warned that you might need to tweak this at times.

The APPDATA folder is normally a hidden folder in your windows File Explorer. You can reach the folder from a command prompt by typing `cd %appdata%`. You can also modify your File Explorer settings to not HIDE folders from you by changing the settings under **View › Options › View › Show hidden files, folders and drives**.

According to Windows POWERSHELL on **my system**, the %APPDATA% Environment variable points to:

```
PS C:\Users\willard> $env:APPDATA
C:\Users\willard\AppData\Roaming
```

This means that Blender USER DATA such as User Preferences, Default Startup Blend Files and 3rd Party Add-Ons will be stored in the `Blender Foundation` folder under

`C:\Users\willard\AppData\Roaming\`

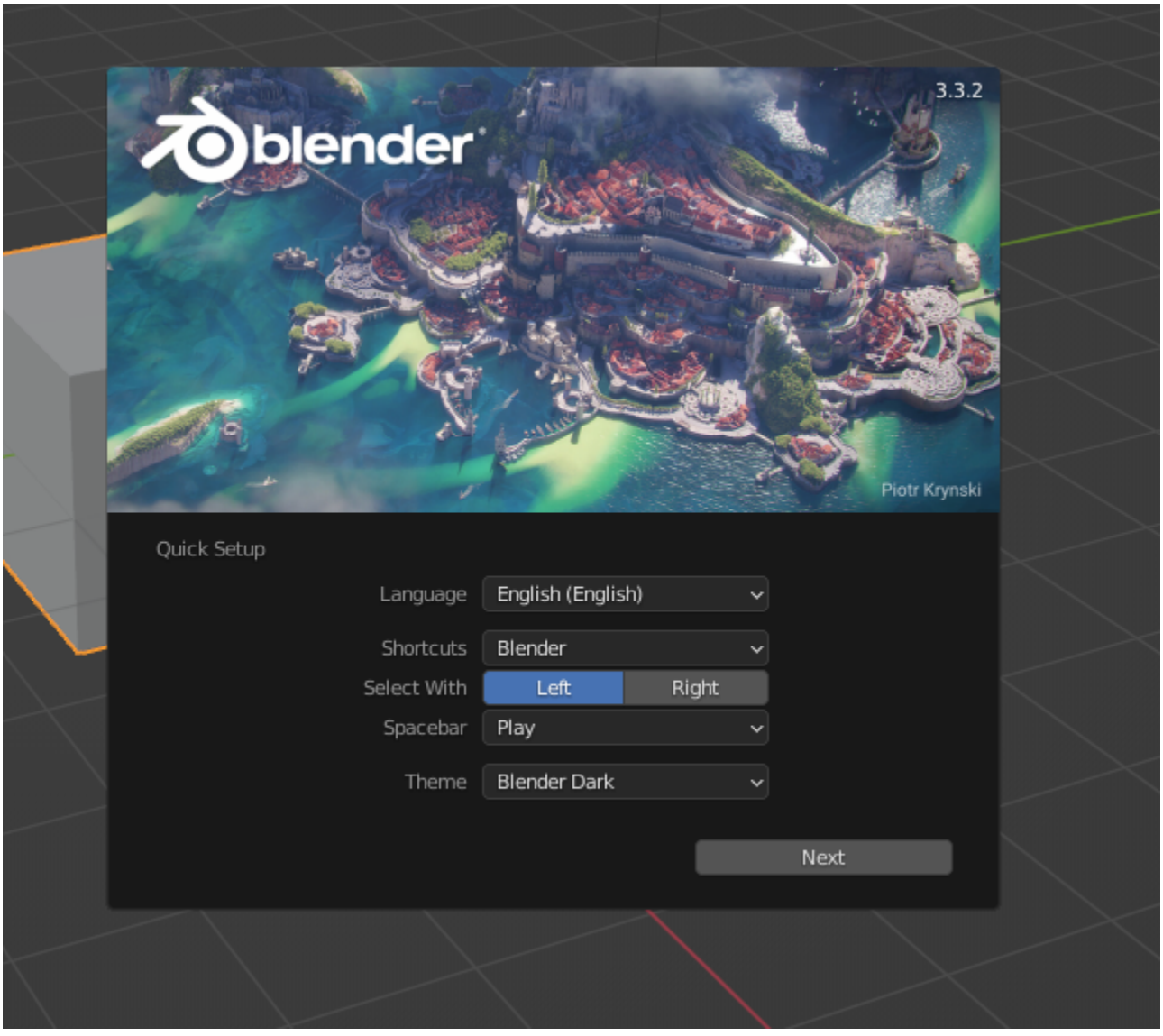
in other words, my config files and addons are located at

`C:\Users\willard\AppData\Roaming\Blender Foundation\Blender\3.3\config`

Below is an old video link, but you will get the general idea of how to install and initially configure Blender… These steps are also contained below. Video Link: https://youtu.be/ad4vTwCGodo

Setting up our Blender Environment

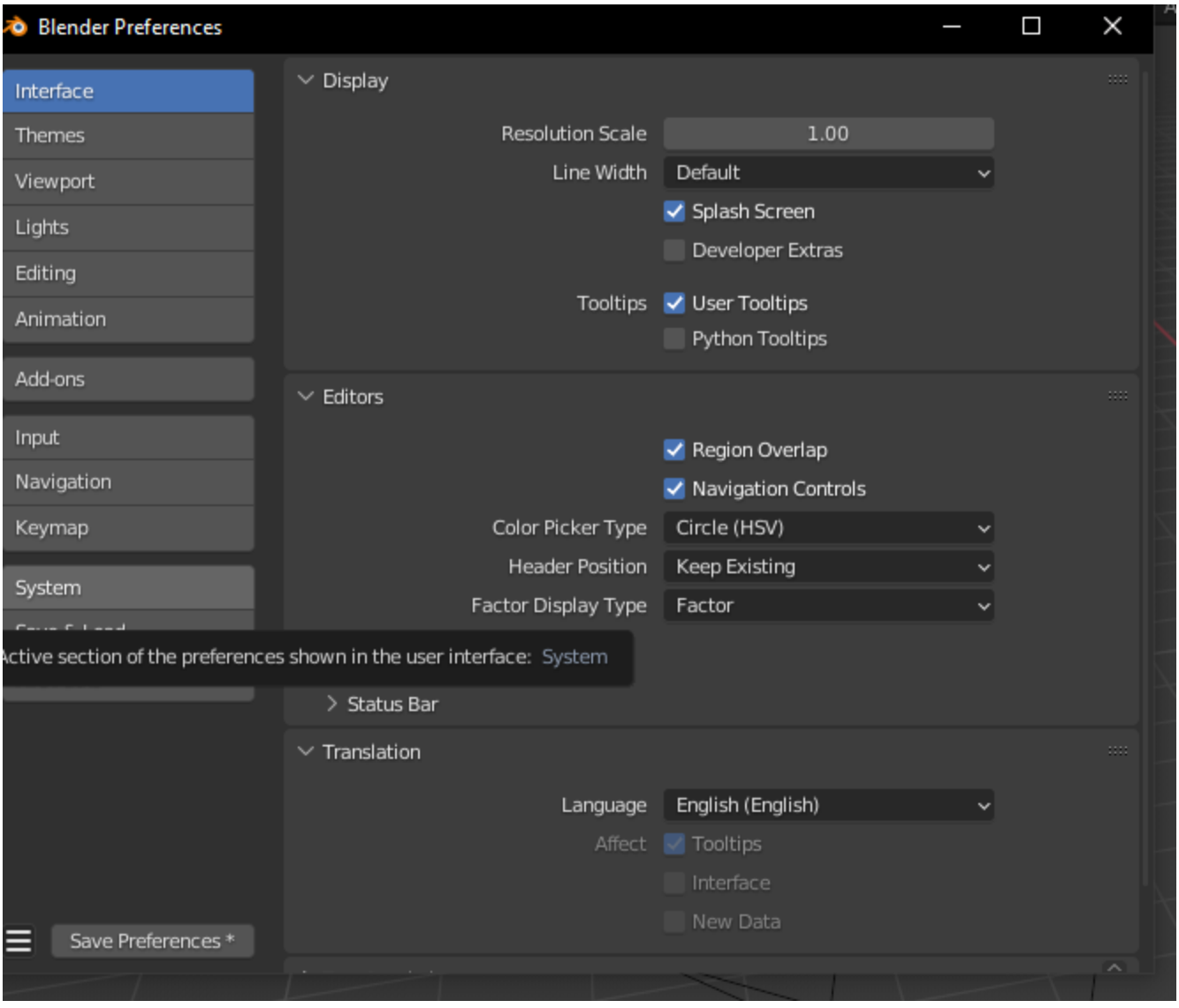
There are some post installation steps we will follow to customize Blender for the kind of work we will be doing. We will start on the main one-time setup splash screen where we will make one change. We will change the `SPACEBAR` key to perform a `SEARCH` instead of `PLAY ANIMATION`.



You will only see this screen when your "User Preferences" file doesn't exist yet, so you won't see it again unless you do a fresh installation or you choose to delete the USER PREFERENCES file.
`%appdata%\Blender Foundation\Blender\3.3\config\userpref.blend`

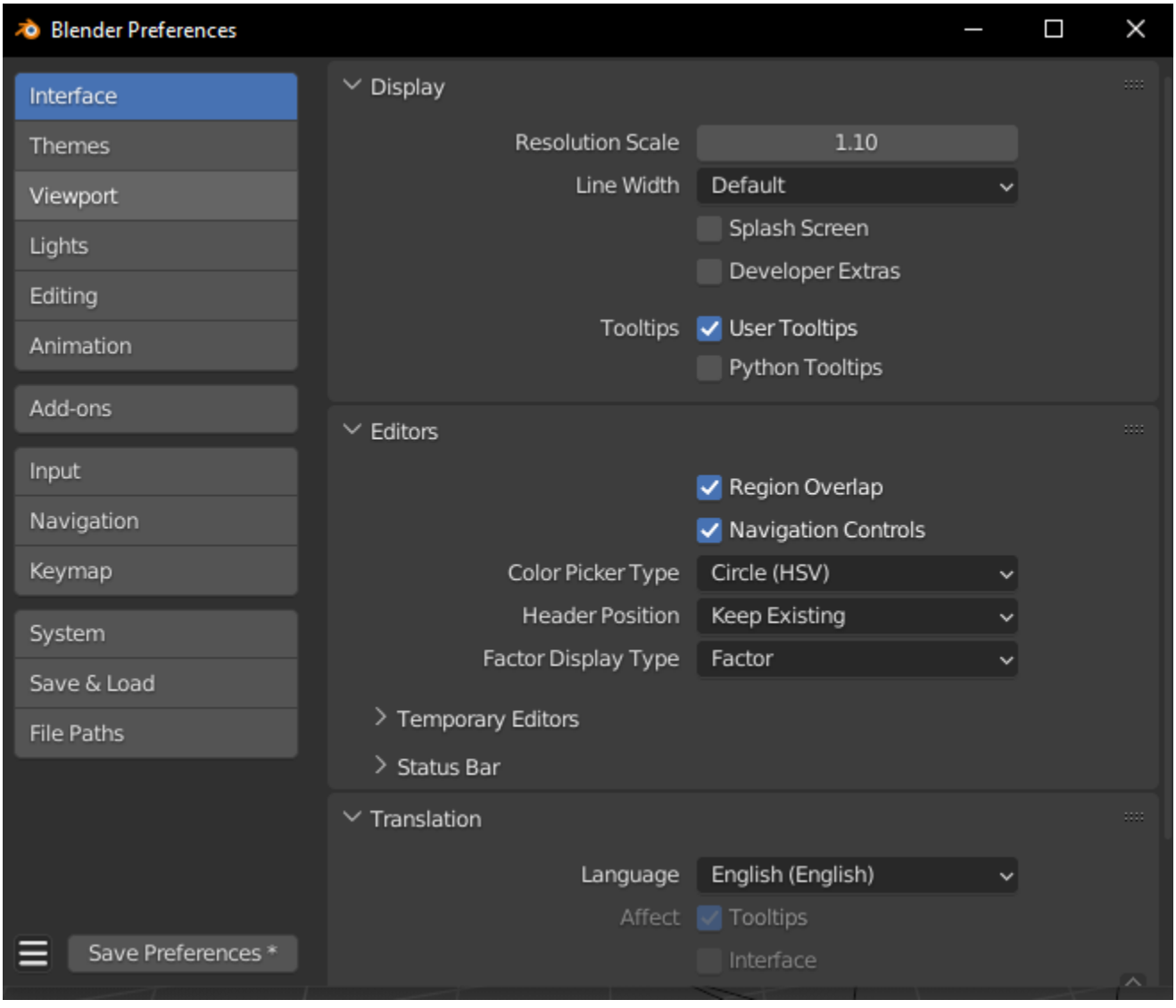
Next, we will go through the steps of customizing our PREFERENCES settings. These are located under the **EDIT › PREFERENCES** menu pull-down on the top bar. (Look for the Gear Icon)

Interface



In this screen, you might want to adjust the `Resolution Scale` to get the most readable text size based on your monitor's resolution. You can use the mouse to slide that value left or right to adjust the screen content size.

I would also consider unchecking the "Splash Screen" option as once you see it a few times… you realize that you don't need to see it.



Screen with options changed to what I prefer…

Themes-3D Viewport

In the **THEMES › 3D VIEWPORT** section, a common practice is to adjust Face Orientation Alpha setting (currently BLUE)- By adjusting the alpha setting to the LEFT, the BLUE will no longer show for outside facing Normals, but will still show RED for inside facing Normals. The BLUE is a bit jarring when the "FACE ORIENTATION" setting is enabled. Change the Alpha value for the Blue to 0.

At the bottom of this menu, you can adjust the VERTEX sizes to make them more visible by changing them from the default of 3 to a value of 5. If your screen resolution is high, making the vertex size 5 or 7 can help with visibility.

Viewport

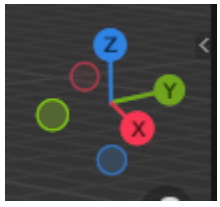
For better view screen visual results, you might want to adjust these values in the **VIEWPORT › QUALITY** section:

- SAMPLES = 16

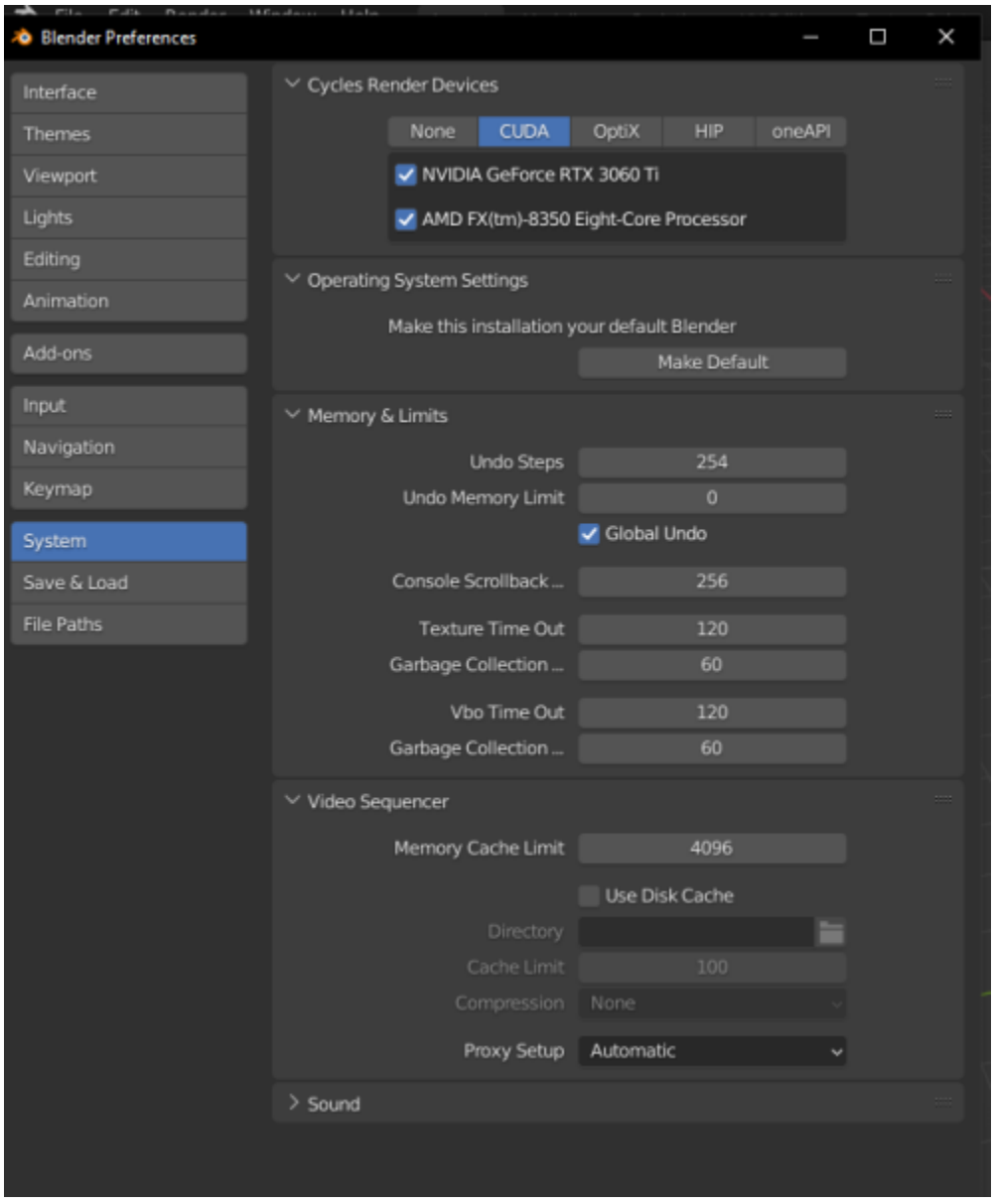
In the **VIEWPORT › TEXTURES** section:

- ANISOTRIPIC = 4x

You can also adjust the 3D Viewport Axis to a smaller X,Y,Z line reference (change to Simple Axis) if you don't like the big navigation gizmo.



System



In the **SYSTEM › CYCLES RENDER TAB** section:

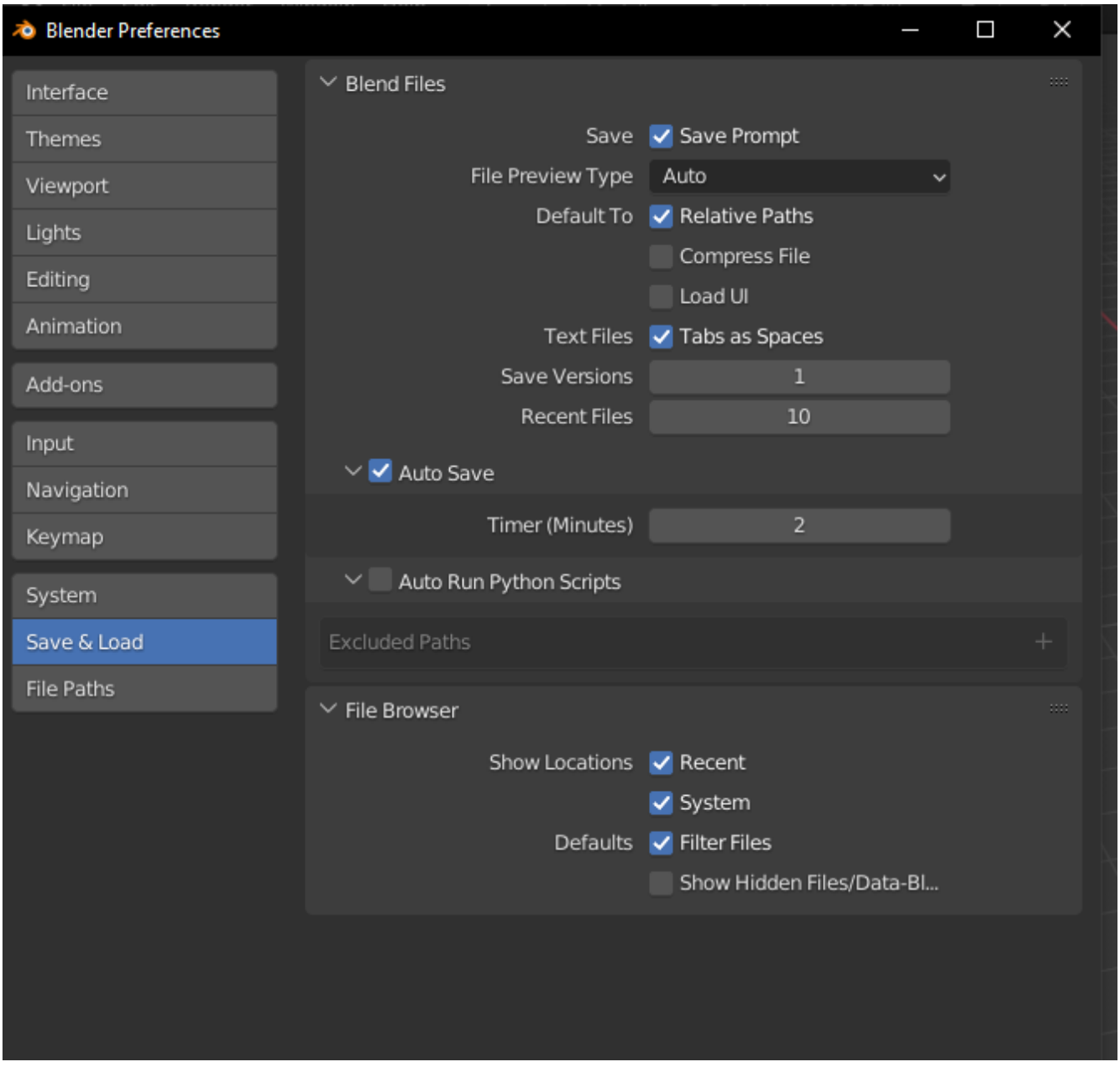
- Adjust CUDA settings to use both CPU and GPU - if possible.

In the **SYSTEM › MEMORY & LIMITS** section,

- Change UNDO steps to 200 or higher. I believe that 254 is the maximum value though. The default is only 32 undo steps, which isn't much.

Unless you are using the modifier stack to perform non-destructive modeling actions, most if not all actions in Blender are one-way. This means that once you modify a shape, there is no going back unless you perform UNDO STEPS `CTRL` + `Z` in the current editing session. I advise saving a copy of your work using a revision system like `crane_V1.blend`, `crane_v2.blend` where you create a new version at the start of each session. This way you can revert to earlier editing sessions if you need to give up on your latest revisions in favor of using an older editing session.

Save & Load



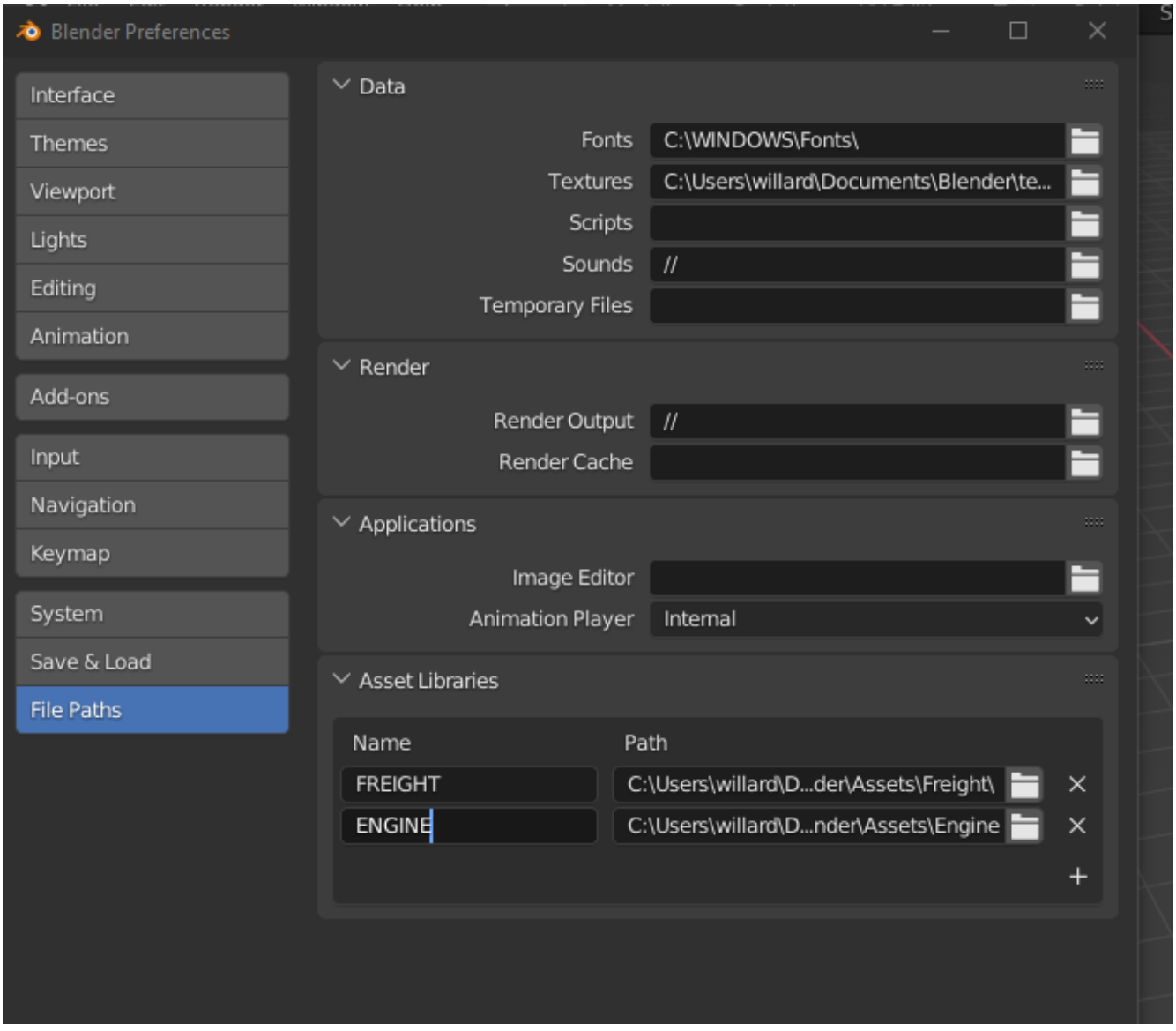
In the **SAVE & LOAD › BLEND FILES** section:

Consider unchecking the `Load UI` option. When this is checked, the User Interface settings come from the .Blend file that was just opened and not the normal User Interface you have setup as default for newly created .Blend files. This is especially troublesome, if left enabled, when you import someone else's .Blend file, as the user interface could then be very different from what you normally use. Change this as needed, but you should know how to include (or not include) the User Interface settings when loading .Blend files. You may find that you often want to switch this setting on and off.

I believe that you would want to RE-ENABLE this option if you have .Blend files that are using millimeters, common settings for 3D Printing model files, so that you would maintain proper UNITS settings when working on smaller objects while using Blender for things other than Open Rails.

I would also adjust the "SAVE VERSIONS" option to allow for automatic backup copies of the current .Blend file to be created.

File Paths



Blender relies on a number of default or user-defined file locations for certain things like FONTS, TEXTURES, TEMPORARY files, etc… this is where these settings can be changed. Many of these locations default to your standard "DOCUMENTS" folder on Windows.

In the **FILE PATHS › DATA** section:

- I usually define a common "textures" location for my 'library' objects, items that I share across multiple models. %USERPROFILE%/DOCUMENTS/Blender/TEXTURES, for example.
- The SCRIPTS location is where you would place your Blender Python Scripts (There is one we may need to use that we will discuss in another section of this document)

`The next section applies only if you have installed Blender version 3.0 or newer.`

In the **FILE PATHS › Asset Libraries** section:

- This setting will default to %USERPROFILE%/DOCUMENTS/Blender/ASSETS
- If you would like to have multiple ASSET library sections, you would `LMB` click on the `+` symbol on the bottom right of the "Asset Libraries" window to add a new ASSET Folder(s) to the list.

You will see that I have renamed the default "user" location's name to be "FREIGHT" and changed the folder location it references and I also added an additional library folder for ENGINE related library objects.

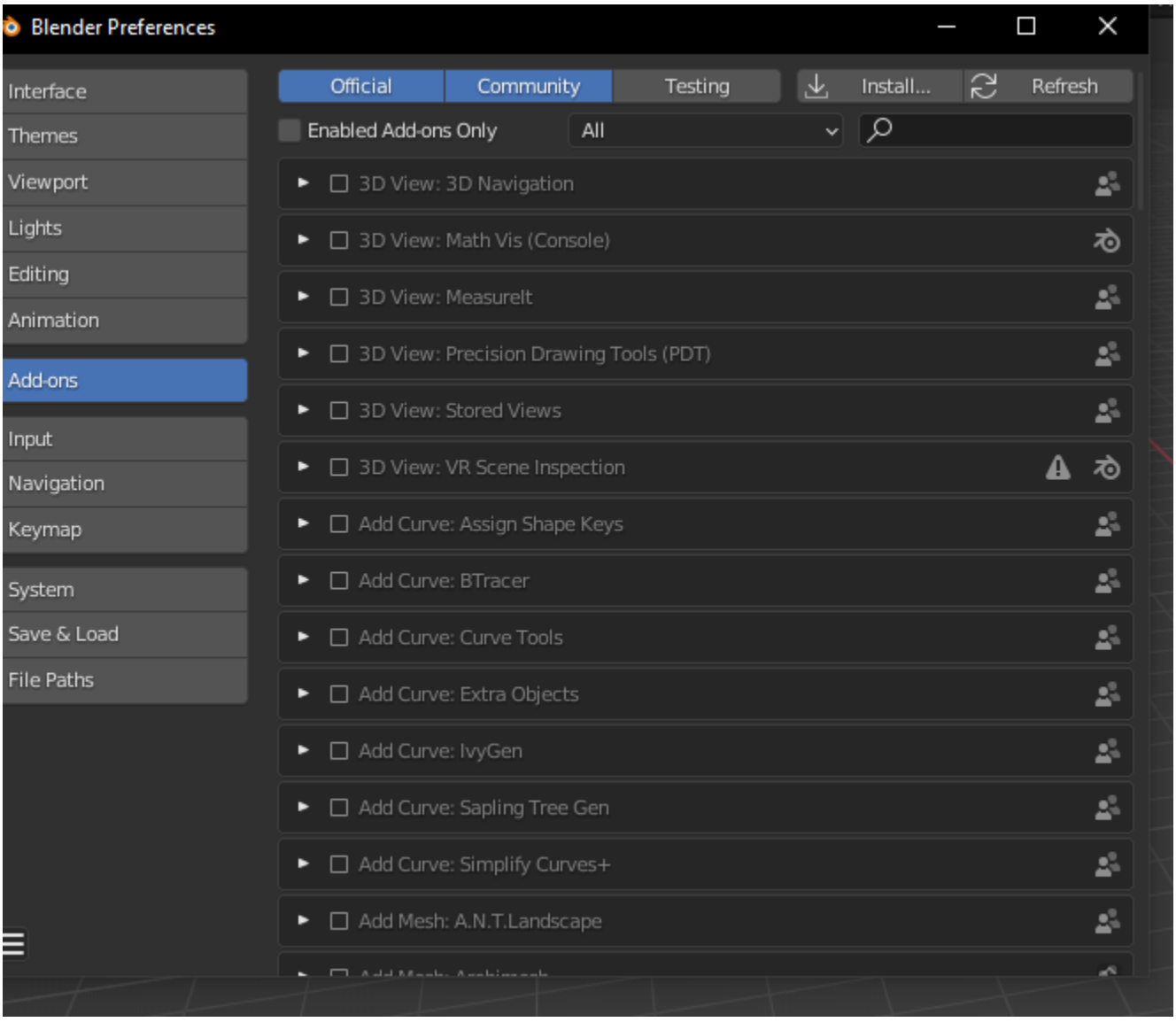
Asset Library files are .Blend files with 1 or more objects in them that are specifically **marked** as ASSET OBJECTS. This will make them show up in the Asset Browser window. Objects in the .Blend files that are NOT marked as asset objects will not be seen and will not be available as Asset Library Objects.

To see the Asset Library while you are editing 3D models, using the "Layout" tab at the top of the screen you will see the Animation Window at the bottom. We will replace this window with the Asset Browser window by clicking the small "clock" pulldown menu next to the PLAYBACK tab of the Animation window and then select the "Asset Browser" item from this menu.

Resize the Asset Browser window by pulling up on the top of the Asset Browser window when the cursor changes to double arrows. This will allow you to better see the Asset Browser's available objects. For example:

Various libraries can be selected using the menu options on the left.

Add-Ons



In the **ADD-ONS › OFFICIAL + COMMUNITY** section:

The Add-ons section lets you manage secondary scripts, called “Add-ons” that extend Blender’s functionality. In this section you can search, install, enable and disable Add-ons. Blender comes with some useful Add-ons built-in that are ready to be enabled. You can also develop and add your own, or install any of the ones you might find on the web.

Blender’s add-ons are split into two groups depending on who writes or supports them:

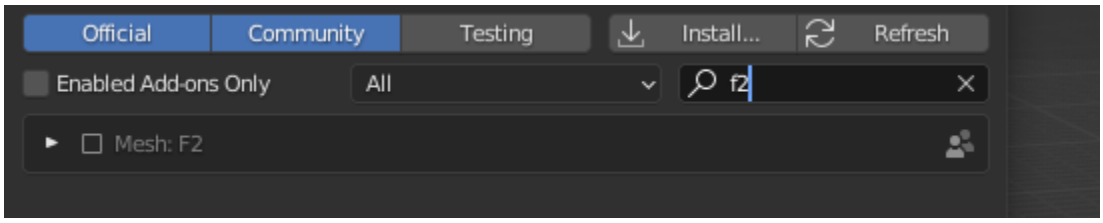
Official
: Add-ons that are written by Blender developers.

Community
: Add-ons that are written by people in the Blender community.

Enabling and Disabling Add-Ons

Enable and disable an add-on by checking or unchecking the check-box of the add-on you have selected. Add-ons are immediately available when checked, or disabled when unchecked.

To locate a built-in add-on, use the search option and then use the check-box to enable it.



Add-ons are grouped by their TYPE, such as MESH, IMPORT_EXPORT, CURVE, etc.

To get us started, here are a few built in Add-ons I recommend installing:

(When searching, use the word on the right (see below),for example type; `F2` to locate the built-in or already installed addons)

- MESH: `LoopTools`
- MESH: `F2`
- MESH: `Edit Mesh Tools`
- NODE: `Node Wrangler`
- ADD CURVE: `Extra Objects`
- ADD MESH: `Extra Objects`
- MESH: `Automirror`

Loop Tools

This and EDIT MODE addon that has a lot of operators for turning any number of edges into a circle or curve, It can bridge edge loops **and** add segments while doing it, it can turn sloppy loops into a perfect curve, it can flatten things at graduated angles unlike using `S` `X` `0`…. It can TWIST things… It has a cool LOFT option (PLAY WITH IT) and finally, it can average out the distance between a series of vertices using the SPACE option. When in EDIT MODE, it will show up on the right side *N-Panel* menu under the `EDIT` tab or in the `RMB` menu.

F2

A quick FACE creation (filling holes) add on to save keystrokes when manually adding faces, especially in repetitive face creation. It used the `F` key.

Edit Mesh Tools

Another EDIT MODE addon tool and, like Loop Tools, it has a right side *N-Panel* menu and a `RMB` menu. It has sections that deal with vertices, faces, and edges. It has some features that require multiple steps to do normally. The menu options are pretty self explanatory, so its worth doing some experiments to see how its features can be helpful. Some of the best features are related to face manipulations.

Node Wrangler

This add-on gives you several tools that help you work with nodes quickly and efficiently. `CTRL-SPACE` for general menu, `CTRL-T` for texture and UV Mapping nodes, `CTRL-SHIFT-T` for PBR Texture nodes, `CTRL-SHIFT-LMB` to isolate a texture, `ALT-RMB` for Mixed Shader nodes, and `CTRL-RMB` for quick links.

ADD CURVE: `Extra Objects`

Will add a number of additional Curve Object Primitives, such as Arc, Arrow, Cogwheel, Cycloid, Flower, Helix, Noise, N-sided, Profile, Rectangle, Splat and Star, some various knots, and additional variants of available primitives.

ADD MESH: `Extra Objects`

Will add a number of additional Mesh Object Primitives, such as five types of beams with Beam Builder, gears, honeycomb, diamonds, pipe joints, stars, some additional shapes similar to the Suzanne Monkey head, Add a single vertex, and wall builders.

Automirror

A quick and easy object mirroring tool with multiple options. A bit simpler than the Mirror Modifier.

3rd Party Add-Ons

We will initially start with some freely available 3rd Party add-ons from the Blender community that we will want to have, including one optional one if you plan to model for Dovetail Games Train Simulator.

Add-on	Where to get it	What it does
Blender28toMSTS	Available From: https://github.com/pwillard/Blender_MSTS_ORTS_Exporter/blob/main/Blender_MSTS_ORTS_Exporter.zip See the video:[4] Instructions say to un-zip the the file before using the Blender `install` option, as it won't install correctly due to the additional support files included with the distibuted `.zip` file. DOCS: https://github.com/pwillard/Ebook-MSTSORTSExporter/blob/main/MSTSORTSExporter.pdf	Edge to Curve
Available From: https://github.com/Stromberg90/Scripts/blob/master/Blender/Edge_To_Curve.py	A python script to create "Curves shapes from Edges" that can be converted back to a mesh. It can be used to create handrails and pipes.	Optional: Briage28

Note: These are free add-ons, but some of the add-ons that you could want to install later might be add-ons you would need to pay for.

There are hundreds of add-ons that are not distributed with Blender and are developed by others. To add them to the list of available add-ons, they must be installed into Blender.

To install these, choose the **`[Install…]`** button and use the `File Browser` to select the `.zip` or `.py` add-on file.

You will then have the option to enable to disable the installed add-on using the check-box.

From Marek at ELvas Tower: I have been using the free `BlenderKit` add-on recently. It's enabled via **Preferences>Add-on**. `BlenderKit` has over 5000 materials and ones like rubber, metal, plastic come in handy for rolling stock texturing. I found a checker plate procedural material that I applied to the walkway sections of my model below. Because it's procedural you can scale the checker without loosing image quality.

I guessed at the scale and then rendered the image using the top down camera I had used to render the AO. I then edited the resultant render image to remove the roof and other details so only the walkway was left and placed in on the texture file in GIMP. Because the scale of the orthographic camera did not change compared to the AO everything was lined up. Blended with the AO layer below and the subtle shading its light years ahead of what I have been able to achieve in the past.

And this was just the first try as a test that I spent about 20 minutes on. The render engine in Blender, along with 3D texture painting are tools that texture artists should look at as additional tools at their disposal.

Exit and Save Preferences

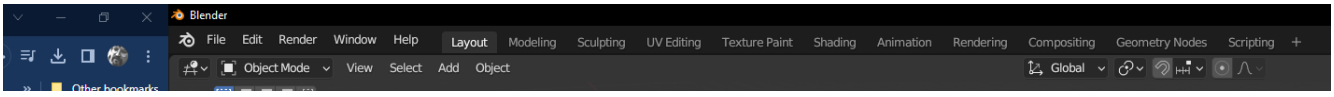
Exiting (closing) the **EDIT › PREFERENCES** section will save your changes. By default, it automatically saves your changes unless you have unchecked `Autosave Preferences` in the *hambuger* menu at the bottom of the **Preferences** window.

Other Settings

We will continue making changes to the header and properties panels near the top and right side of the screen respectively. Some. if not all, of these changes are purely optional but they are worth checking out.

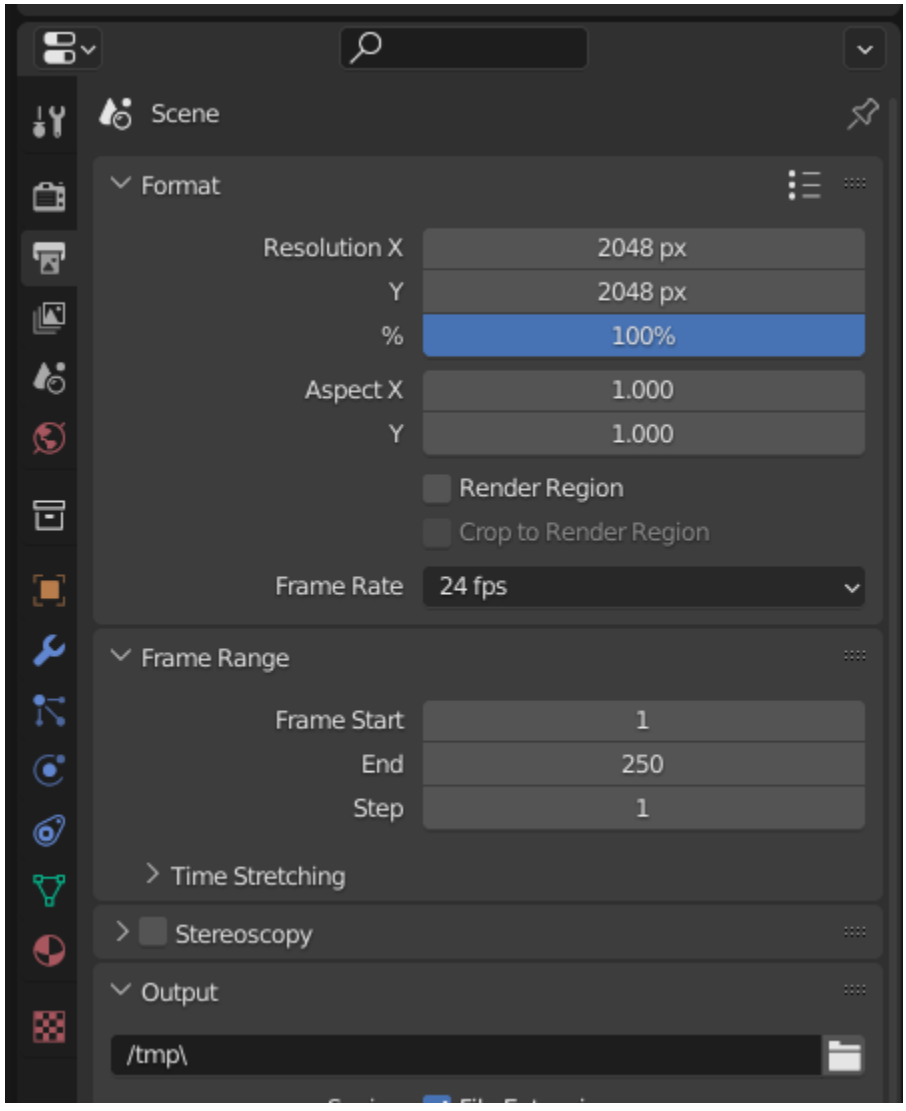
TABS

Along the top of the main Blender window, there are a series of tabbed menus. We don't actually need all of these to be available to us while modeling Open Rails content and you might notice that some of them seem redundant. Let me explain… the **Modeling** tab is actually a holdover window from earlier versions of Blender and is nearly identical to the newer **Layout** tab. Feel free to delete the **Modeling** tab as you really won't be needing it. Use **Layout** instead. To same a bit of screen real estate, you can delete the Sculpting tab as well since you would hardly need it for *Hard Surface Modeling*.



You can always tweak the TAB menus to your liking as these changes can be undone by clicking on the + symbol in the menu to **add** additional TAB menu options. Be sure to save your changes using the file menu option **File › Defaults › Save Startup File**.

Scene Settings



Define your default output resolution as 2048x2048 since we will generally be working with square textures.

Gismo settings

Below is a change that is very related to your personal preference, so it is optional. Some feel this creates a messy window.

In the Layout window there is the GIZMOS Drop Down menu. (Look for an arc with an arrow icon) You can enable the MOVE option with a check-box. This will give you AXIS based MOVE arrows that you can grab to assist with moving an object around your scene. (Or you can just use the "G" key options.)

Overlays

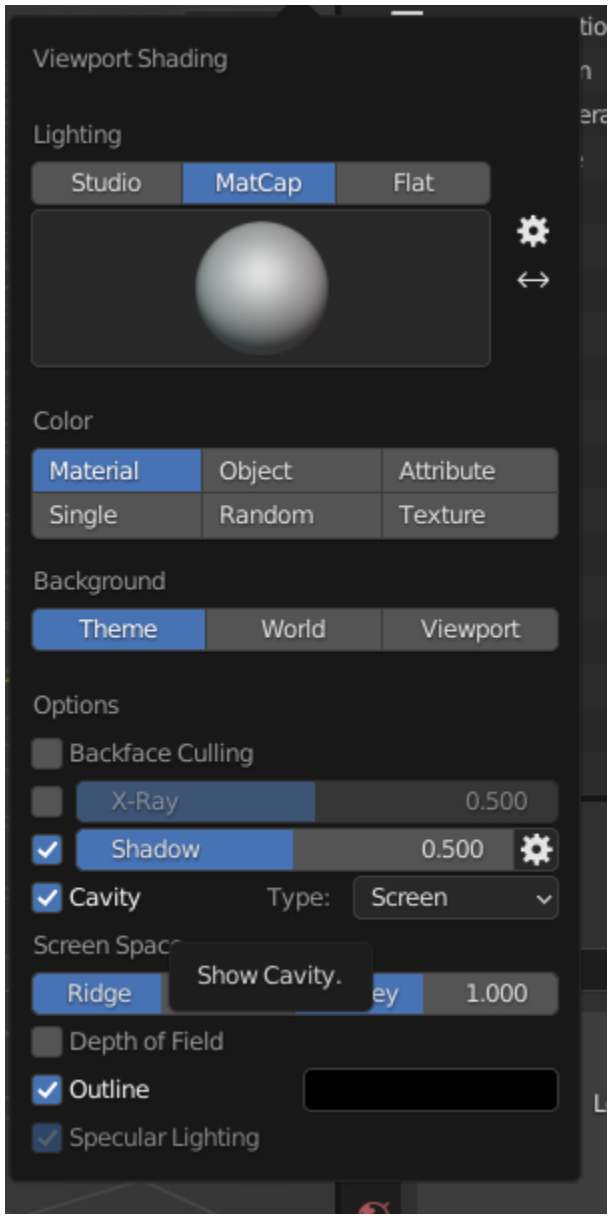
Right next to the GIZMO options, there is an icon for OVERLAYS.

An option in Overlays that you might want enabled is `Statistics` at it will help you keep an eye on your poly count budget.

Below is a change that is also very related to your personal preference so it is optional. Some feel this creates a **really** messy edit window on complex shapes.

Under Overlays (Select the default cube and change to `EDIT MODE` using `TAB`. If there is no default cube, then use **Add › Mesh › Cube** to place a cube in the scene and go to EDIT mode using `TAB`.) In the `OVERLAYS` drop down, enable the `EDGE LENGTH` check-box. This will show the actual edge unit lengths of selected objects when you have and edge or edges selected.

Shading Settings

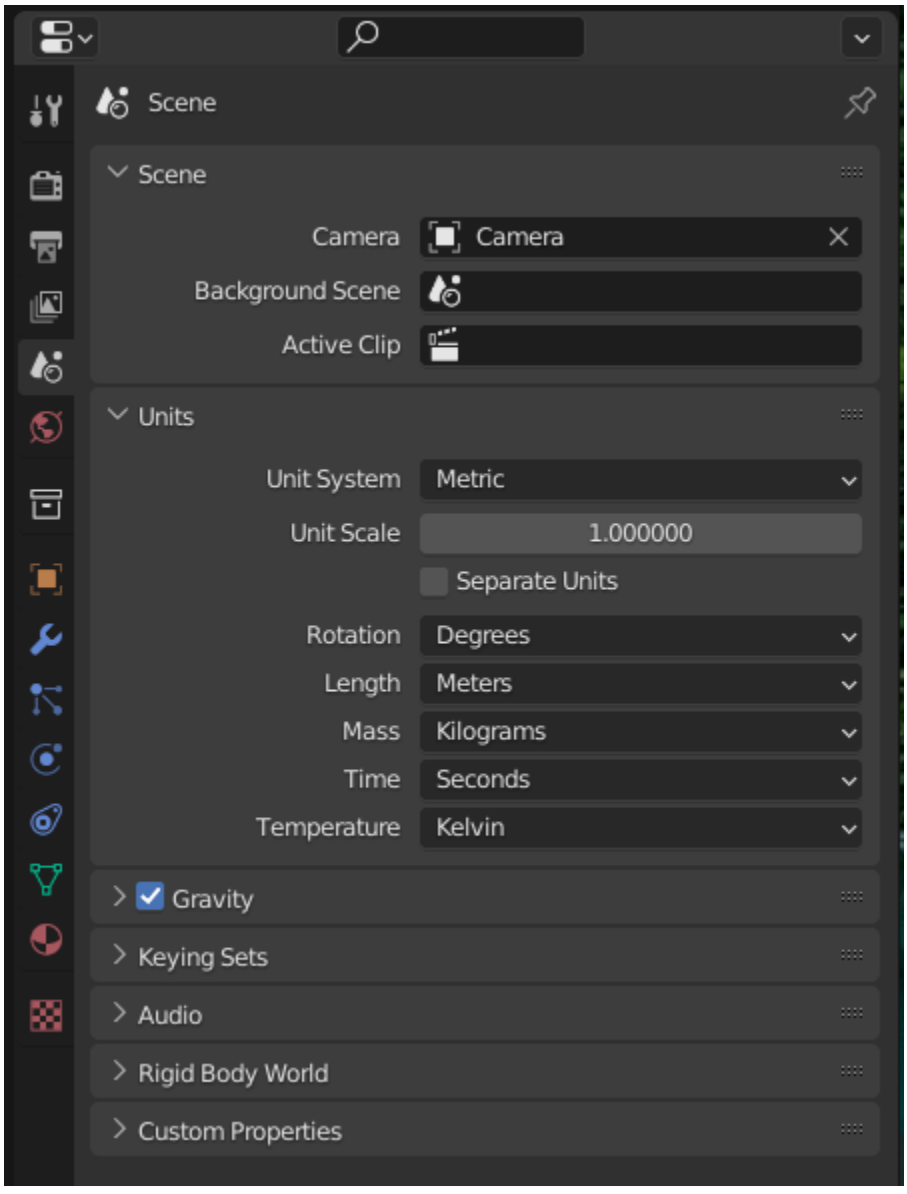


- Under SOLID VIEWPORT SHADING (Locate the Solid Circle Icon and use pulldown on the right of it), change the following:
  - Enable the check-box next to Shadow
  - Enable the check-box next to Cavity

These steps help to make things more visible while editing.

I also sometimes choose MATCAP and pick the leftmost MAPCAP option for better visual representation of what I'm working on in the early stages of modeling.

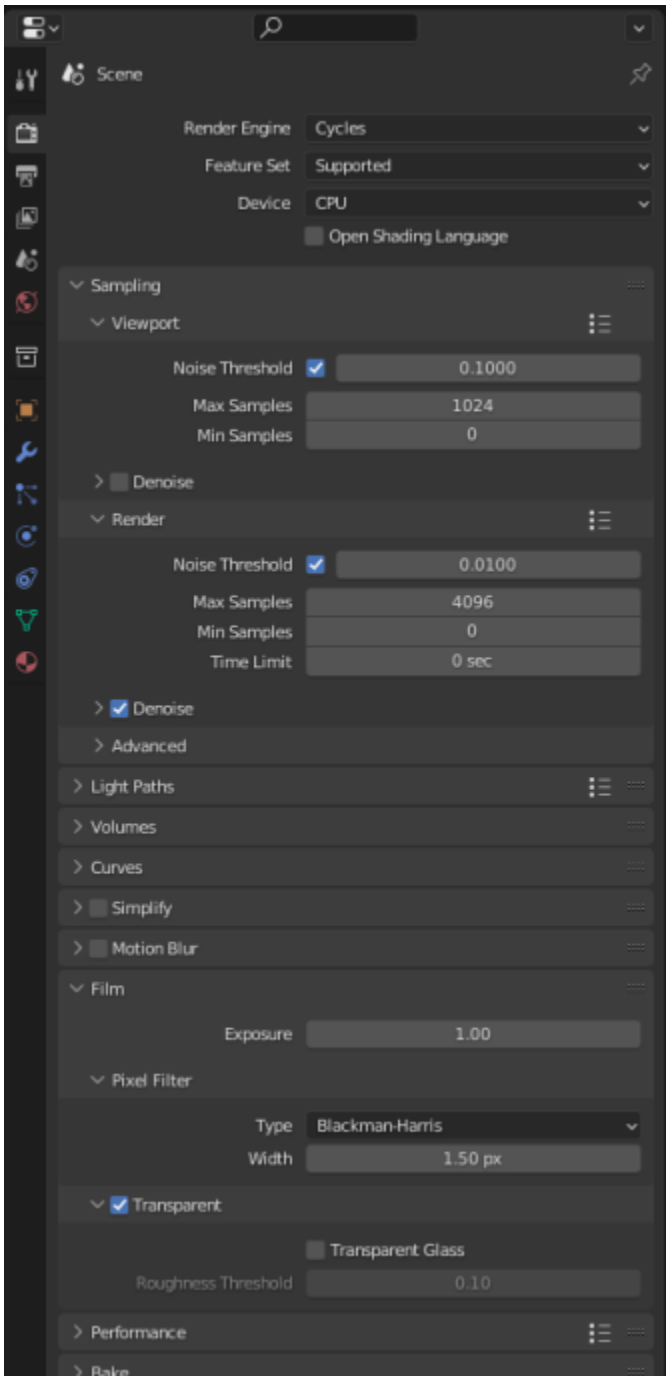
Units



If you need to work in Imperial Units versus Metric, you would adjust the setting under the UNITS SYSTEM option in Scene Properties.

Changes to `UNIT` settings can have effects in other areas, such as Camera View settings related to `Focal Length`, `Clip Start` and `Clip End`. To adjust these values aftsr changing UNITS settings, use the *N-Panel* `View` tab to make adjustments.

Scene (Rendering)



The settings here are optional but give better render results.

Properties Panels › Scene › Render Engine › Eevee and modify SAMPLING > RENDER option to 200 SAMPLES

Properties Panels › Scene › Render Engine › Cycles and modify SAMPLING > RENDER option to 200 SAMPLES

Properties Panels › Scene › Film In the Film section, Enable the check-box for "Transparent" (This removes any background from renders, you **will** want this)

Shading

Related to the `Shading` Tab on the Top Bar Menu, we will make some adjustments for lighting. Rather than rely on actual `light` objects in our scene, we can create general illumination effects based on HDRI images.

This step requires that you have already downloaded an example HDRI file from one of the HDRI WEB SITE locations mentioned at the beginning of this document. The downloaded HDRI files should be located in your **Documents › Blender › HDRI** folder that you create for this purpose. See TIPS below for what I am using.

In the "World Properties" panel (globe icon on the right), Add "Background" in the Surface pulldown menu. In the Color section, select "Environment Texture". This will allow us to chose the HDRI we downloaded and made available for Blender to use. Select the "Open" button and choose the HDRI file you want to use using the "File Browser".

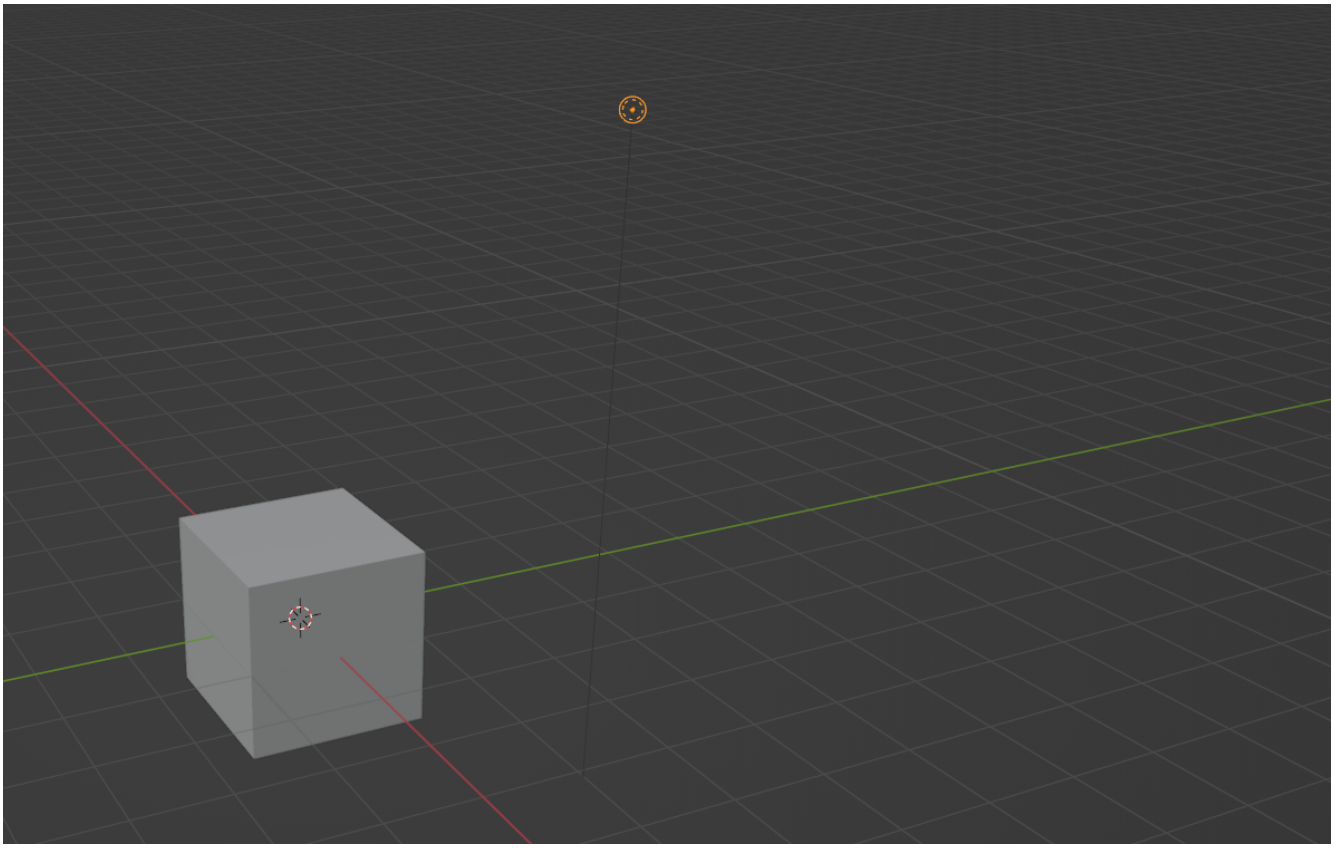
Now, when you chose Render, you will be using the lighting from the HDRI file.

If you now select the `Shading` tab and then choose the `Viewport Shading` view icon (The one to the right of Solid Shading Icon) you would then choose the pulldown on the right (viewport shading) and then adjust the default lighting setting to `Scene World`. You will now see your work with the new HDRI background in place.

When you now Render your model, this would be your background if you had not chosen "transparent" in the FILM option earlier.

I have used the `Abandoned Slipway` HDRI from https://hdrihaven.com as recommended by **Josh Gambrell** for neutral outdoor lighting in the past. Currently though, I am using a file called `hdri_004_nordicfxnet.hdr` aka "Railroad Crossing" from http://www.nordicfx.net/ as it seemed more appropriate. I did scale it down from 4K to 2K though.

There is a free HDRI addon that makes setting up the World HDRI lighting image files a bit easier called "EASY HDRI". Totally optional of course.

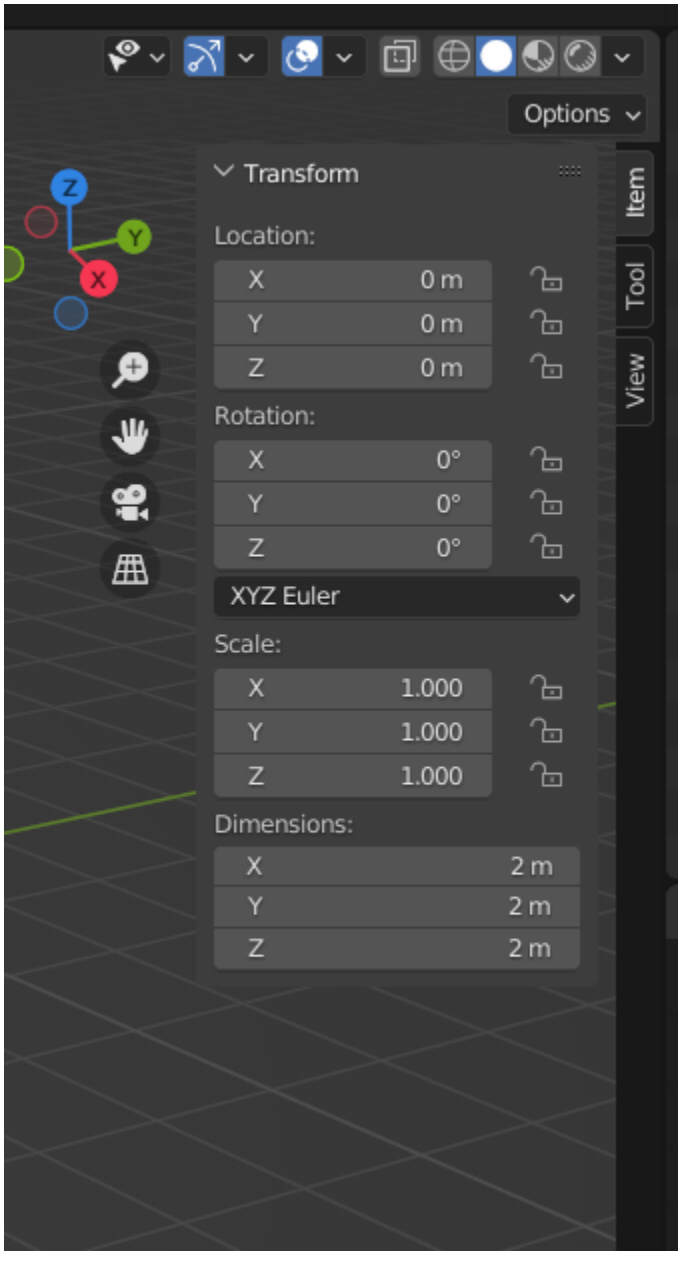


The last thing we need to do now that we have replaced how we do our lighting for renders is to delete the default light source in the default file. In the Scene Collection, or in the main 3d Window select and delete the default "light" object.

Done, for now

One last step…

Press the `N` key to bring out the `Number Panel`, referred to as *N-Panel*, where you can adjust objects settings and locations by entering numbers. This will have the *N-Panel* available when we start new Blender projects.



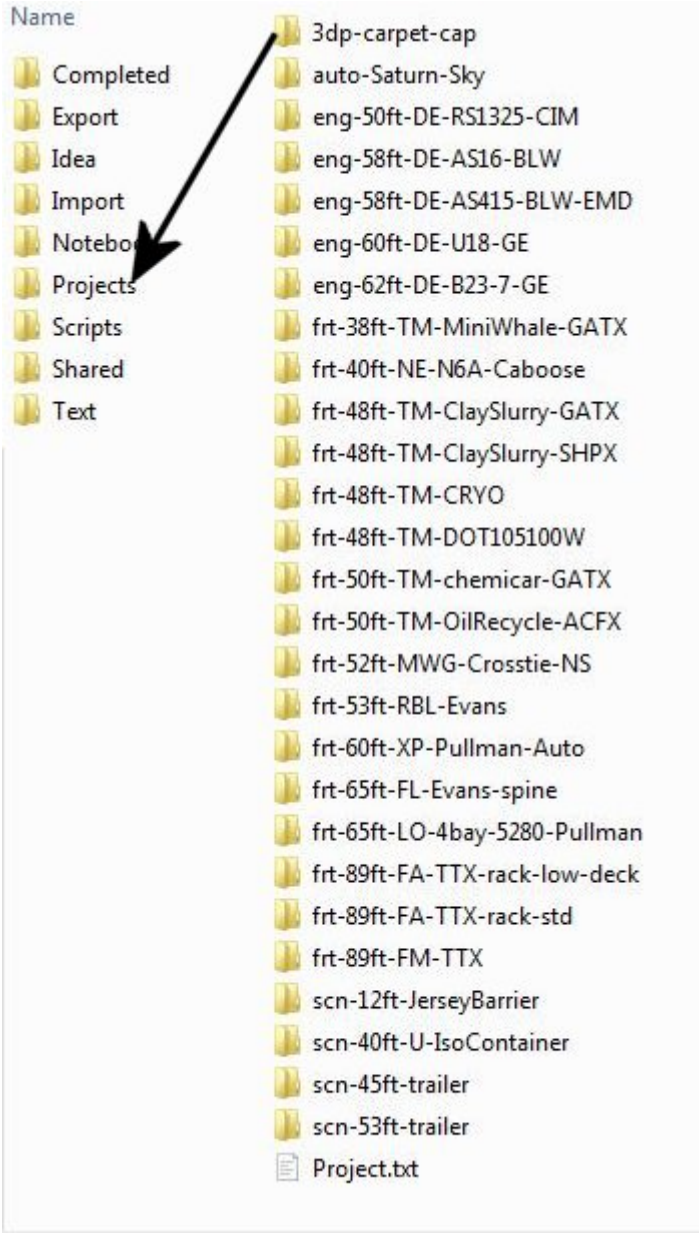
With these changes done, its time to save our settings.

FILE › DEFAULTS › SAVE STARTUP FILE › Confirm

We will now have all of our basic user preferences and startup file options the way we want them for 3D modeling simulator assets.

== Folder structure

The layout of your project development folder is completely personal depending on how you organize your work. I'll share what I currently use as an example.



I snipped a lot of details out for brevity and left some in as examples. EXPORT is for renders and such, IMPORT is for things I'm converting from 3DCANVAS and TSM. PROJECTS, should be obvious, but I use prefixes and a naming standard to make things easy to find. I use a SHARED folder for things that all projects will share, like Asset Libraries and common textures. I prefix my Freight cars with FRT, Scenery with SCN and Engines with ENG… you get the idea.

[4] Youtube https://youtu.be/j3AVw7s9qoA

Content Creation Overview

The information presented below is not specifically about modeling with Blender, it is about how to create items that comply with the simulator requirements and guidelines. Some references to my old 3DC notes are included here.

Reminder, this is basically a "notebook" and started life as my collection of content creation files. It may seem to jump around a lot as a reult.

Orientation

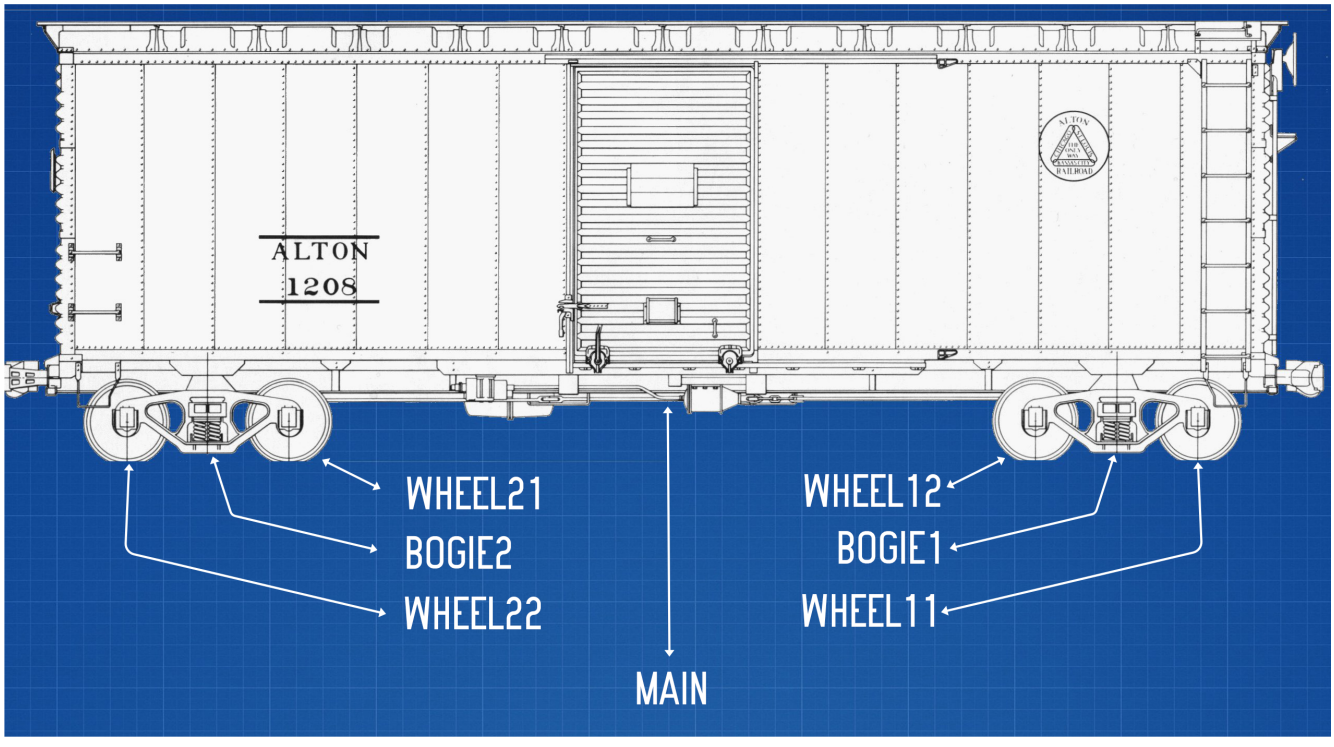
There is a requirement to properly orient your model if you are making rolling stock or an engine. If you have used 3D modeling software in the past, maybe 3D Crafter / 3D Canvas or Maya you might be accustomed to Z axis being DEPTH and the X axis being WIDTH and Y axis being HEIGHT. Blender orientation is similar to 3DS MAX where Z is HEIGHT, X is WIDTH and Y is DEPTH.

When working on things that roll on the track, the FRONT of a model is aimed towards the Positive values of the Y axis and the REAR of the model faces the negative Y axis.

Blender's default point of view differs slightly in that it considers the FRONT to be facing the -Y direction. This comes into play when using the view keys on the keypad, where pressing `1` for front view will have you looking at the back end of your 3D model from a MSTS and Open Rails perspective.

Engine/Wagon Model Hierarchy

For a model to work correctly in Open Rails, there are some requirements that need to be met if you want to have the simulator properly automate animations for wheels and bogies. For simplicity, I am implying that the A-END of the boxcar is on the right and the B-END (with Brakes) is on the left. The Boxcar's default direction of travel in this case then would be this way, →, or from Left to Right. (Refer to `ORIENTATION` section above)



The way I have found to get the correct layout of a dual 2 axle `BOGIE WHEEL` arrangement is to have all wheels use the center of their axle as the pivot point and the bogie use its default center of mass as its pivot point. In general, all other parts in a model will use world origin as the pivot point.

The Main body part does not need to be called MAIN. Modelers have been using that as a convention since 2002 so it has become a standard thing to do but it's not a requirement for Open Rails. The Blender `S` file exporter by Wayne Campbell only requires that you use `MAIN` as the name of the `COLLECTION` that refers to where your model objects reside.

Be careful with selecting all objects in object mode and then applying "all transformations", as it will reset all objects to have their origin (pivot point) to be `world origin`. This would change the pivot points of the bogies and wheels.

Standard 2 Axle Freight Bogies

If you are looking at the model from the Left Side View, the forward direction of the model is facing right. Starting at the right, the bogies and wheels are named according to the diagram above, where associated WHEELS parented to the related BOGIE1 or BOGIE2.

If an additional axle is needed, use WHEELS13 or 23 located behind the bogie pivot relative to forward motion and shift wheel set 2 to the center of the related BOGIE. Its is important that the naming sequence remains (11 to 23) as shown in the diagram or wheels will turn backward and shift improperly in MSTS.

You would not have a 2 axle BOGIE with WHEELS13 or WHEELS23.

Isolated Axles

The MSTS naming standard for isolated axles with non-bogie wheels, is WHEEL1, WHEEL2, WHEEL3.

These are primarily used for STEAM locomotives. Animating these wheels is not automatic and must be done using the animation tools within Blender. The Steam wheel and linkage animation uses a series of 16 frames. In general, it's non-trivial to to create this animation and I won't be covering it in this document. (unless someone provides us with good notes about how it is done)

In ORTS, it appears that only the WHEEL and BOGIE prefix is required.

Some MSTS documentation leads you to believe that a third bogie is possible in MSTS - it isn't. However, if you are modeling specifically for Open Rails, then you should know that the simulator will properly animate anything with the correct BOGIE and WHEEL prefixes as long as you follow the guidelines for parenting and local pivot origins.

Microsoft recommended the following topology for the Acela as an example:

```
MAIN ENGINE
PANTOGRAPHTOP1
        PANTOGRAPHBOTTOM1
    PANTOGRAPHTOP2
        PANTOGRAPHBOTTOM2
    BOGIE1
        WHEELS11
        WHEELS12
    BOGIE2
        WHEELS21
        WHEELS22
    MIRRORRIGHT1
    MIRRORLEFT1
    WIPERARMLEFT1
        WIPERBLADELEFT1
    WIPERARMRIGHT1
        WIPERBLADERIGHT1
```

Configuration Files

The Open Rails (OR) manual provides a good understanding of the features only available in Open Rails, which can be used to create more capable and accurate content. Although the current manual is lacking in content creation details, much of the information available for Microsoft Train Simulator content creation still applies.

While I'm not going to create a full guide to `sd`, `ref`,`eng` and `wag` files here in this document, we will need to create a working file if we plan to add content to Open Rails. Peter Newell's website has an in-depth look into creating good `ENG` and `WAG` files for Open Rails.[5]

Also, see [Creating Rolling Stock and Locomotives]

Various General Notes about Content Creation

An interesting discussion occurred in 2017 about making these files better and more useful in the post-Microsoft Train Simulator world.

> KUJU, an organization, defined all of the folder names we use in MSTS. The Include file concept, as applied to .engs and .wags, led me to conclude that something similar to KUJU's \common.cab directory tree was necessary for .inc files. By looking at how payware vendors used folders in \trains, I noticed that they sometimes used \common.cab, other times a product-specific folder, and occasionally something in between, such as a vendor-named folder for the unskinned mesh (e.g., 3DTrains_FPack).
>
> In KUJU's example of a CAB file, you will see a good template for locating the "include" files. After much experimentation I'm proposing am solution that addresses these needs:
>
> - A folder for shared .inc files, much like what is in \common.cab. Recognition that many end-users have routes and equipment from many countries and therefore it might be useful to group certain files for each country.
> - Acceptance that many payware vendors sell the same mesh skinned for many railroads but when distributed they use a unique folder for each railroad.
> - Addressing the easiest to solve problems with minimal commonality, where everything belongs in one folder.
>
> — Erick Cantu, 20 November 2017

Proposal for saving space on a PC:

Making use of a COMMON folder to store shared details across multiple bits of content. This can also be about improving performance as well

Within the \trains directory,

Add \common.fleet

and

Add \common.model

Within both of those directories, add folders (one for your own country and others only as needed) for country codes.

Examples:

```
\AUS Australia
\AUT Austria
\BRA Brazil
\CAN Canada
\CHE Switzerland
\CHN China
\CZE Czechia
\DEU Germany
\FRA France
\GBR United Kingdom
\HRV Croatia
\HUN Hungary
\IND India
\ITA Italy
\JPN Japan
\NLD Netherlands
\POL Poland
\RUS Russian Federation
\SVK Slovakia
\SVN Slovenia
\SWE Sweden
\UKR Ukraine
\USA United States of America
\ZAF South Africa
```

For myself, this means I should strive to have (at least) this format somehow:

\common.fleet\USA

and

\common.model\USA

General Modelling Standards from Erick Cantu

I feel these comments about creating content from Erick are so important that they need to be documented here as a reference. These details have been extracted from forum posts at Elvas Tower and are meant as a guide if you want to emulate his results. While these are not hard rules, they definitely give us something to work with.

Erick supplies a lot of details about his approach to modeling here. Its worth reading and keeping these as guidelines for your own modeling.

Erick Says:

- Match the right trucks to the right car. This goes beyond an A-3 or S-2 in that I make sure that my cars have the right wheel size and truck wheelbase (i.e. tonnage rating), hence why I spent so much time building a library of freight car trucks. I was planning on making my truck library available. Importing the trucks into Blender for modellers using that program could actually be very helpful.
- I map to a resolution of 1/2 inch per pixel on standard-definition cars. The alignment of all adjoining car surfaces should be within 1/1000 of a pixel so that continuous lines are not broken. I map all car sides separately. Most cars fill either the top or bottom half of a 2048 x 2048 pixel texture sheet so that large surfaces are continuous and one texture sheet can be used for two paints (for flat cars, it might be three because they take up so much less space). My standard setup now is to produce four shape files, with the first two mapped to the top and bottom of the first texture sheet, and the second two mapped to the top and bottom of the second sheet. This allows for four different paint schemes or weathering patterns per car folder, with numbers applied with decals so that each main shape can represent many cars.
- Texture mapping needs to be intuitive and functional. The key questions I ask myself are: "Would I want to paint this?" "Would I need to request a modified version of this car to do [insert railroad here]'s livery?" "Can I remove parts I might not need with the alpha channel?" "Could I paint this car without using shape viewer to check my work?" I have seen some really bad mapping in the MSFS world where it was clear that the model builder was more interested in impressing everyone with how good they were at fitting some major part between spaces at an oblong angle than creating something functional (imagine entire airliner fuselage sections angled 20 or 30 degrees on a texture sheet to fit in a space - how are we supposed to do straight lines on that!?).
- Cars should all use consistent bitmap resolution and be designed with a wide variety of systems in mind. This consistency should apply both between cars and between the constituent parts of the cars themselves, including the texture mapping scale. Ideally, triangle counts should be around 12,000 or less at the top LOD, with aggressive optimization of vertex counts through the limiting of hard edges and careful welding of UV coordinates wherever possible.
- There should be no more than one texture sheet for the carbody and no more than one for the trucks. Because most cars will need at least some alpha work, the entire carbody should use a single BlendATexDiff material (with the -Trans flag OFF) unless there is a compelling reason to apply a second material. Reasons include a need to have specular highlights applied to only some parts of the mesh, or, in the case of the Foobie hoppers, to allow the car side to be mapped as a single unit, with only parts of it using an alpha channel. Adding a material means adding a drawcall, so this should be a very conscious process. It's okay to have trucks in a separate texture because animated parts are going to be a separate drawcall anyway.
- The texture resolution and mesh resolution must be congruent and consistent with a target viewing distance. One of the things I really hate is when people build these super high resolution parts with super hi-res textures and then put them on a model where the gestalt is a much lower resolution. It's a waste of vertices and real estate on the textures for the detail parts because everything around them is going to look like garbage in comparison. The same goes for detail parts that are a high resolution when the rest of the mesh isn't, so you have this hilariously detailed builder's plate right next to a wheel with like 20 sides, so you'll never see the difference in the builder's plate detail without zooming in to a point where the wheel right next to it completely breaks the suspension of disbelief. A cursory glance at Turbosquid or CGTrader makes it plain that 3D model builders are terrible at this as a rule. There are some practical exceptions when running against OR limitations. For example, I backlight my number boards with lights that change based on the loocmotive's coupling status. But OR will render any alpha part ahead of another alpha part with a 1-bit alpha channel, and the light halo is a flat polygon with an alpha channel, so the only way to make the numbers look good is to bump up the texture resolution (again, 1-bit edges always look like garbage unless they're perfectly square).
- Splitting car sides into multiple sections is not recommended, as it adds unnecessary mesh divisions and UV coordinates. It may be necessary for very long cars, such as auto-racks, however, for most cars of 60 feet or less, it's really not necessary if you plan ahead. Using the top half of a texture sheet for one car and the bottom half for another creates a large, rectangular area to work with and allows for multiple cars to use a single texture sheet. Remember, Open Rails is sensitive to both drawCall counts and the total number of images across a train, so putting multiple cars in one texture isn't actually a bad idea.
- The top node's pivot should be 2 inches below the rail to ensure that wheels sit on the rails correctly. (Keep in mind that you might need to tweak this for proper ride height)

- Part of this consistency will be making the cars look good together, which means consistent air hose heights. I am willing to supply sample cars that can also be cannibalized for parts. If compatibility with all of the cars I am building is a goal, then the tips of brake lines should be 14.5" above the rail, extended to a position where it will meet the air hose on the next car, and held in place the way most are in the real world, with a chain or cable (I use a simple cable shape).
- Couplers going through other couplers are the worst, so it's probably best to have them as either part of the truck mesh (which is what Erick Cantu of NAVS does) or have them linked to the trucks in the hierarchy. The exception, of course, is cars where the truck centers and couplers are far apart, such as auto-racks, boxcars with end cushioning, and so on. Obviously, the couplers would have much too wide a range of motion through most curves under these circumstances.
- Keeping draw-call counts low is important. Keeping overall texture counts low is even more important. Car bodies should strive to use both a single texture sheet and a single material for that sheet unless there is a need to have more than one material (e.g., for specular roofs on cars with flat sides). We can always place multiple car-bodies in a single sheet if a single square texture is not adequate. This has been NAVS standard practice for freight cars for some time now.
- 1-bit alpha channels and aliased edges are unnecessary and unacceptable. OR renders 32-bit textures correctly, so there's no need to have separate materials just for alpha parts, accordingly, having separate materials for 1-bit alpha parts and 8-bit alpha parts is adding unnecessary drawcalls. It's much more efficient to use a single material with an 8-bit alpha channel if there's going to be any need for greyscale alpha on a model at all. The only time a 1-bit alpha channel is acceptable is when a transparent area is perfectly square. Otherwise, any curved or angled lines just look like hot garbage.
- The MSTS convention of having the underside of all freight cars be completely devoid of any geometry, leaving the user to see sky when the car is viewed from below. This flies in the face of the fact that bridges are a thing, so this practice is best avoided. Car undersides do not need to be complex, but they should be present.
- Generally, it's wise to use whatever units match your reference materials to avoid unnecessary conversions. For example, when I build a Boeing, I work in decimal inches. When I build an Airbus, I work in decimal meters. For most US stock, reference materials will be in inches. Its not a rule though. If you have a scale calculator handy, its not hard to convert Imperial to Metric and visa-versa.
- No "imagineered" details
- Within the target viewing distance I set, I try to model everything that should be visible but at a consistent resolution with the rest of the model, for example, I build the brake rigging in its entirety, but I use very simple shapes to do it because my target viewing distance for a standard-definition car is no closer then when the whole car is visible on-screen. The resolution of every part on the model needs to increase exponentially as you get closer, so this was the balance I struck.
- Taper your wheel treads. Please, just do it. It's really obvious when you don't.
- Ensure that all UV coordinates that can be welded are welded and plan your mapping ahead of time to ensure that you can weld as many UV coordinates as is possible. Texture vertices matter just as much as drawcalls.
- For the same reason, plan hard edges with purpose. If an edge is always going to be in a dark area, smooth it.
- The cars should look good at all conceivable viewing angles, this includes the many situations where you might be viewing a car from below.
- Blender has an option to EXPORT UV IMAGE, which will provide shape outlines that can be imported into your paint program as a reference layer to show where your shapes are located in texture. This eliminates guess work.
- Round parts need fewer sides as the radius gets smaller, this is why my wheels might be 24 sides at the rims but 12 or 10 sides in the interior of the dish. Edges matter, but you can hide a lot behind them.
- Coupler heights may vary. Air hose heights may not. The ends of my brake hoses have been at the exact same height since 2004 so that there is never a mismatch. Maybe someday, we'll be able to script hoses so it won't matter…
- If a feature can't be done right, it doesn't get added. This is why my track sounds don't incorporate switch and crossing sounds, even though OR allows for it, because you can't account for the truck spacing or the different number of wheels as it presently stands, so it's always going to break the suspension of disbelief.

The main point is to really be purposeful about the choices that you make. Little things will really accumulate over the course of a long train, especially when AI trains enter the equation. This is why I was so intent on redoing my car sounds, because I had been less purposeful about the way that my streams were set up and it was maxing out the number of streams active at any given time. This is also why I have moved from individual car textures to paint and weathering

variations with car numbers applied via decals, because the number of textures, based on my testing, is the single biggest determinant of performance across an entire train. The tradeoff was one extra drawcall per car (the decal shape) so that an entire train with discrete car numbers and multiple weathering patterns could be built with a very small number of textures. In the case of my BN hoppers, 100 cars use 13 texture sheets (every 25 cars uses two car body texture sheets and one decal sheet, with all cars using the same truck texture). To do the same thing without the decal shapes would have taken 51 sheets (one for every two cars plus the truck texture).

Additional guidance from Blender Brothers

On of my mentors while I was in my beginning stages of learning Blender was Josh and Ryuu of Blender Brothers. I subscribed to their Youtube channel and I get regular email updates about their ideas and helpful tips.

Josh and Ryuu of Blender Brothers Say:

"What's the secret to nailing the detailing stage?"

> Well, it boils down to understanding where to add detail, how much to use, and what type suits best. Sounds easy, huh? But trust me, it's not as simple as it sounds.

Blender Brother's Gold Rule

Always think about visual anchors when I add details. Visual anchors are like stars in a constellation. Remove a star, and the constellation loses its sense. Your eyes dart from one star to another, ignoring everything else. What lies between the stars is just negative space. Let's dive into five tips that can help you get a grip on this concept and apply it:

- **Embrace the 70/30 Rule** Keep 70% of your design clean (that's your negative space) and cluster your details in the remaining 30%. This approach creates visual anchors, helping the eyes move from one cluster to another.
- **Keep Details to scale** Adding an oversized detail to an object can make it look cartoonish. If needed, use a scale reference to keep things realistic.
- **Do your homework** on reference images. If you're designing a sci-fi room, for example, look at screenshots from sci-fi games, movies, and so on.
- **Less is more** Instead of going all-in on detailing a specific spot, I hop from one area to another, progressively adding details.
- **Create your own library** of decals, trim sheets, kitbash items, etc. This helps you stay true to your style and speeds up the detailing process.

By the way, the Blender Brothers love making their own decals. They have even recorded a video showing their full decal workflow with the Decal Machine, a Blender Add-on. It's worth looking at as well as their other videos.

General Texture Mapping Guidelines

- Microsoft had recommended using SQUARE texture shapes that corresponded to a specific evenly sized shapes with MSTS, like 512x512, 256x256, 128x128, etc. This limitation no longer applies in Open Rails and you can reasonably use a 2048x1024 sized file if you desire.
- You should be using a paint program that understands layers and is able to save an image document in a format that maintains these layers between sessions. (PSD is a layered format)
- PNG files understand ALPHA channels but they do not maintain layers
- Consider working with PNG files while modeling and DDS files in your final model for Open Rails
- Consider modeling Reporting Marks and Car Numbers as modeled DECALS instead of baking them into the main texture.
- Export and check results as soon as you have created a UV Map for your model before proceeding to adding details. It's better to find issues early.

[5] https://www.coalstonewcastle.com.au/physics/format/

Making A 3D Model

Some Basic Things About 3D Models.

Scale

Open Rails Train Simulator as well as Microsoft Train Simulator each use a scale of 1:1, that is, 1 unit in the 3D drawing area is 1 meter in the simulator and this means all things are in real-world measurements.

3D Space

According to **Blender's view of the world**, `-X` value is left, `+Z` value is up and a `-Y` value is the front side of the object. The "ground" level is where `Z=0`

Material

In Blender, a material is a set of properties that defines how a surface appears when it is rendered. Materials can include information about the surface's color, texture, reflectivity, transparency, and other visual characteristics.

When creating rolling stock items for ORTS, Pressing `Keypad 1` for front view will actually have you looking at the REAR end of the car or engine. You would press `CTRL` + `Keypad 1` to get a front view of your engine or wagon while drawing it. You have the potential to create backwards rolling stock if you get this wrong.

The **MAIN** object of your modeling collection should always be positioned at world origin, which is `X=0`,`Y=0`,`Z=0`, based on the orientation described above.

There are some general rules that all content creators should follow, regardless of the tools used. Take the next section to heart.

While you *can* have Blender work in Imperial units (feet and inches), I find it far simpler to always work in the Metric System and you won't need to change the "UNIT" settings in Blender.

Some Basic Things About Modeling

Some recommendations From Erick Cantu of NAVS

1. In Open Rails, it is important to try to keep the polygon count of your models low, even though it is not as critical as in some other applications. While modern PCs and simulator software have improved in their ability to handle complex models and textures, it is still important to use complexity wisely. In Blender, you can view the polygon count and other details of your model by going to Object EDIT MODE and looking at the lower right corner of the screen. When creating your models, try to avoid including faces and edges that will rarely or never be visible in the final model.
2. Keep the texture files (that will eventually be included with your final model) to a minimum and use as much of the allotted texture as you can. Each texture creates a fresh API call (state change) to a new group versus being able to batch calls together to a single texture.
3. The most important tip. Don't get hung up drawing things that nobody will see. Avoid drawing details that will remain in shadows or small details like nuts and bolts if the camera will never see it. Inversely, now that we have the ability to have more complex models with Open Rails, we **can** add more modeled details where we historically were forced to rely heavily on alpha-channels in our textures.

Polygon Count and Vertex Density

If you read item #3 in the previous section, you might think you can go crazy with details. However, there is always a balance to be achieved not just between graphical quality and performance, but also in computational performance. Every vertex that has to be mapped to a texture takes computing power. Thus, it's good to save polygons whenever possible and use texture maps instead for details like cutouts, structural members, and bolts. If you really feel like you want to model small details, keep in mind that every detail added might prevent the more generic use of the model for re-skinning later on. A flat sided boxcar is more flexible than one with specifically modeled door layout, latches and modeled roof details.

For large panels, though, a minimal amount of modeling is necessary. A flat plate like a wall or door can be made with only a few polygons. Like every rule, you can break it when you really want to show a particular detail, but also keep in mind that you should make good use of "Level of Detail" ranges (LOD) to reduce computational work when an object is farther away.

Preparation of Work

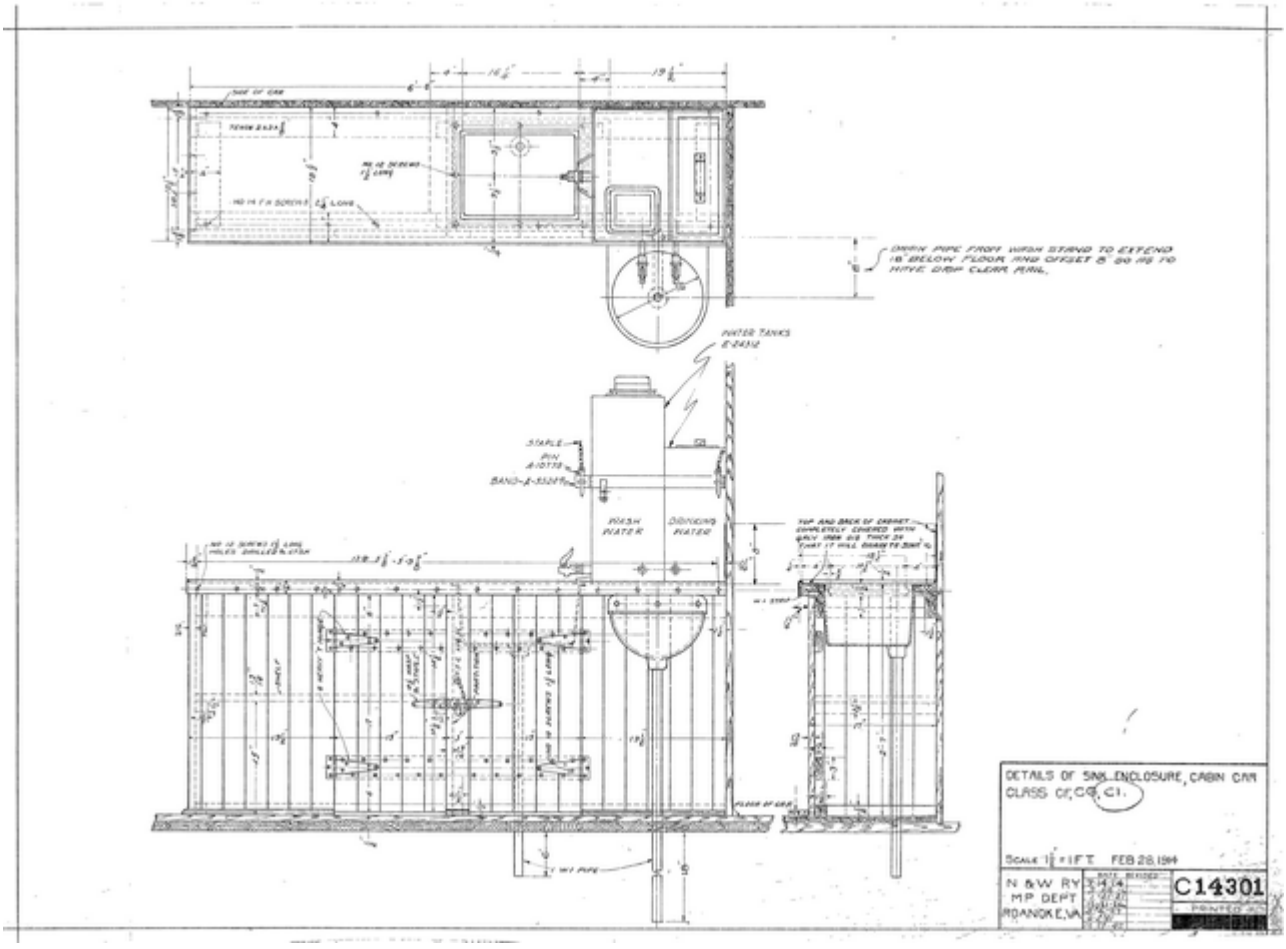
- Of course, you need to figure out what you want to model
- Find some design references (Blueprints, Scale Drawings). Referring to back issues of model railroading magazines can be helpful. There are also some railroad cyclopedia books for sale that were published by Model Railroader Magazine as well as issues of Railroad Model Craftsman.
- Create a collection of reference images, if doing modern content, grab your camera and snap some of your own references. There are a few railroad picture sites that are highly useful as well. (https://www.railpictures.net/, http://www.rr-fallenflags.org/, https://www.railcarphotos.com/)
- Determine which images you will use for background references in Blender. You should have available at least side and front images. In some situations, you can use the same image for both if the views are present.

Background Images

It is important to note that many scanned images of drawings and plans that can be found online may contain errors, as they are often reduced in size to fit on scanners and screens. Older plans, which are often very large (nearly one meter wide), are particularly prone to this issue. Plans created using computer-aided design systems tend to be more accurate and smaller in size, but these are less common and may only be found in hobby magazine articles. Some plans may also be stored as vectors, which allows them to be scaled without loss of quality. Keep these limitations in mind when using background reference images.

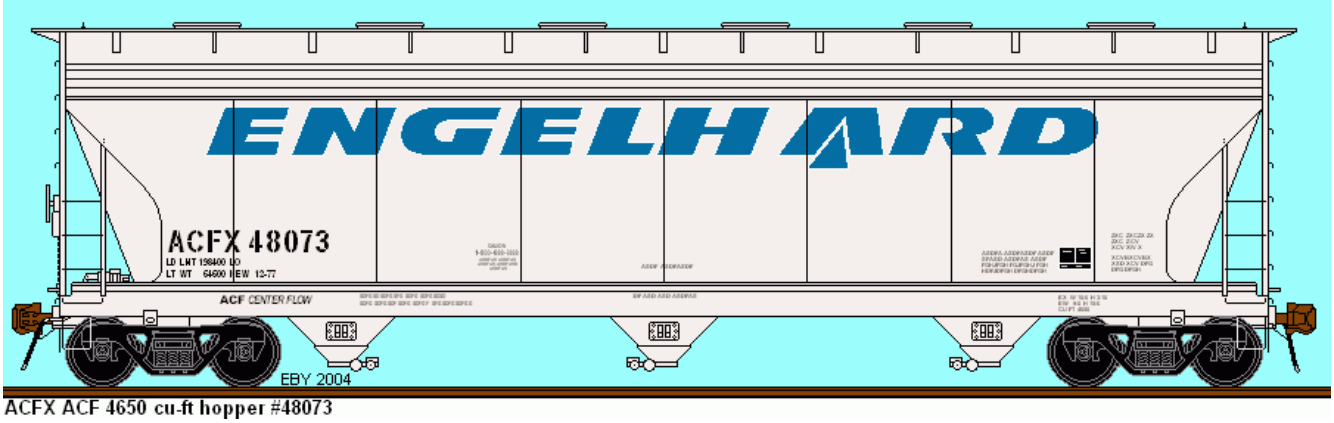
Original Plans

There are web sites online that will sell you scans of original plans. These are often related Railroad Historical Society websites. Unfortunately, there are large images and often quite costly. Example: https://www.nwhs.org/archivesdb/



General Drawings

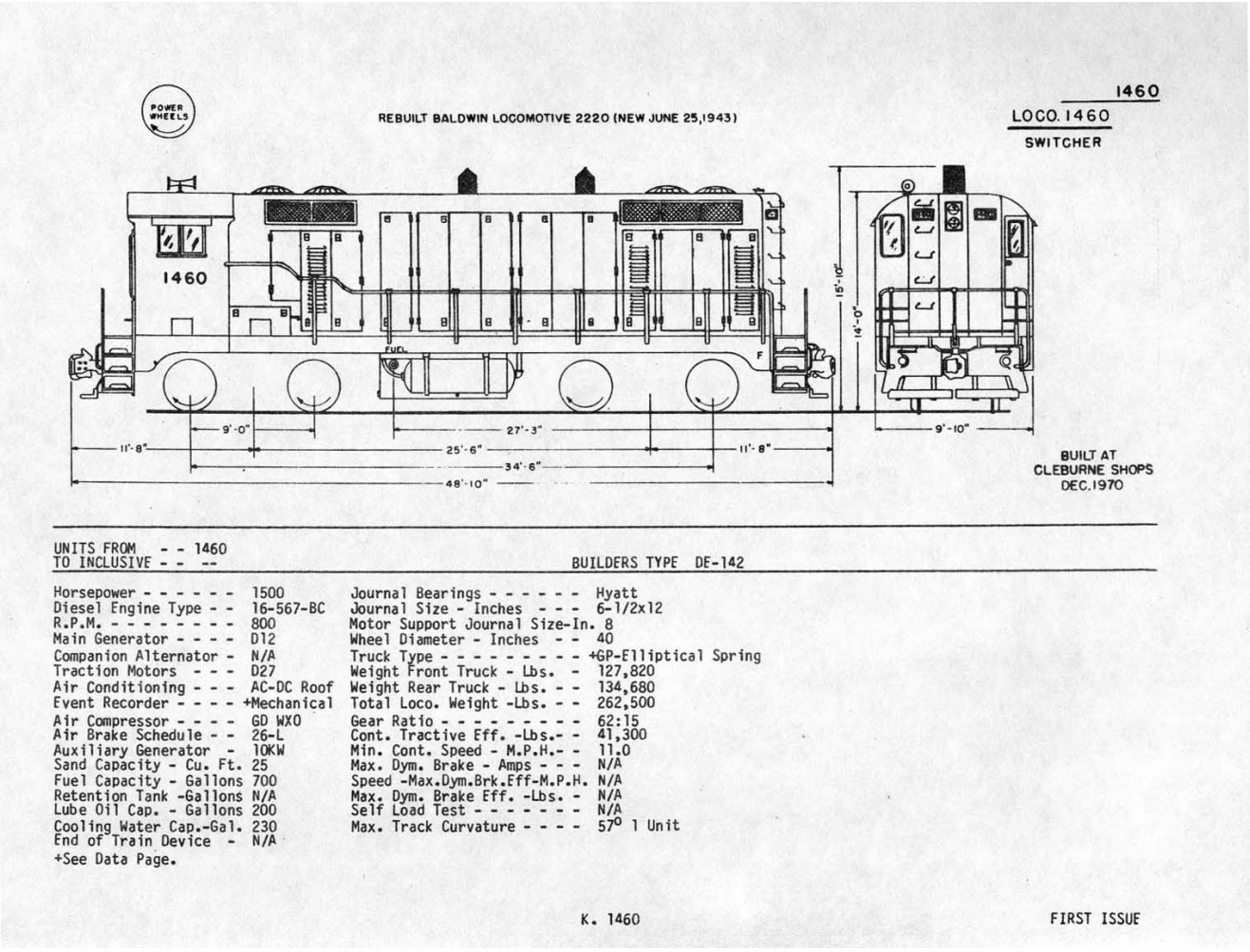
These are often found in back issues of Model Railroader and Railroad Model Craftsman magazines as well as sites like https://www.trainiax.net. These usually contain a good about of Steam Engine, Diesels and Rolling Stock but might lack all the details such as piping and such. These drawings are made based on originals by the staff of the magazines or staff that created the Cyclopedia collections. These have the ability to be a few percentage points off-scale even when they include a scaler reference.



ACFX ACF 4650 cu-ft hopper #48073

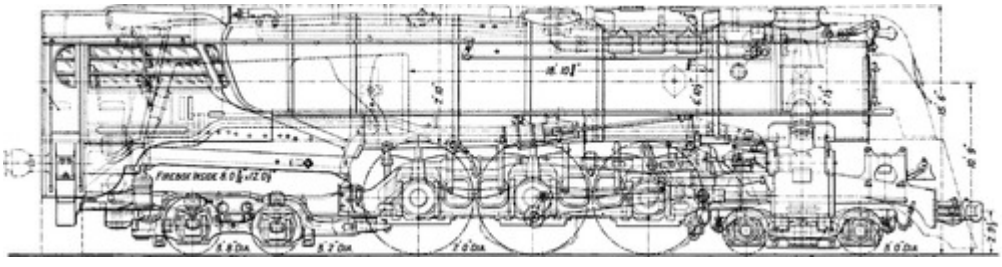
Classification and Painting and Lettering references

These seem to be rather plentiful and contain lots of useful dimensional data. They are often from discarded rail road documents and might have notes scribbled in them. The drawings are often hand drawn and might be a bit messy. They are not that useful for small details but can get you started on a model if you also have good pictures to refer to.



Blueprint Websites

Sites like http://www.the-blueprints.com offer a variety of drawings to choose from. While this might seem like a good resource, I would use it with caution. The content is all user generated so there are varied degrees of clarity and accuracy.



Using background images is optional, but many modelers use blueprint backgrounds as modeling aids.

When inserting background reference drawings into Blender, it is not critical that you match up the image size to equal world space dimensions, but it can save a lot of re-scaling steps later. One of the main issues with using backgrounds in Blender is that some methods you might be accustomed to with other modeling tools don't work in Blender, such as applying backgound image to a cube and using that as your reference, since in Blender, you only see the texture in a texture viewing mode. It will dissapear in Xray/Wireframe and even solid shading modes.

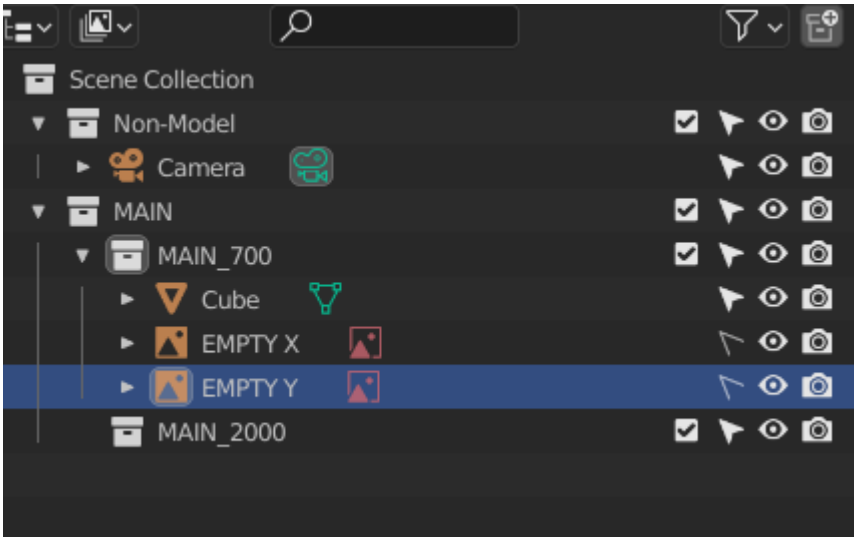
When you insert a background using Blender's **Add › Image › Background**, the image WILL show up while you are in wireframe modes… but note that by default the image will ONLY be visible from the axis it faces. Be careful when you install a background image as you need to to be in the Axis View you want, since the added image will orient itself to your current view. If you want to change this, you can enable the `Perspective View` check-box by clicking on the `EMPTY` and

adjusting the setting it's properties window.

Background images are added to our scene as `EMPTIES`, which means that they are not going to be part of an exported model. To avoid accidently moving or changing the background image, you can use the filter options in the outliner, first enabling the "mouse cursor" icon and then setting the restriction toggles to disable the "mouse cursor" icon for the empties you added to the list, which will prevent you from accidently selecting the background images.

The outliner filter options are pretty handy as you can also SHOW or HIDE the background image empties as needed by clicking the "eye" Icon.

A nice feature of the Image Background is the ability to adjust the ALPHA setting, controlling the with the "transparency" slider to get a see-through effect. First, click the Opacity check-box to enable the feature and then adjust the opacity slider to your liking.



If you want to save the background images INTO your .Blend file, you can use the **File › External Data › Automatically Pack Reources**, which is OFF by default.

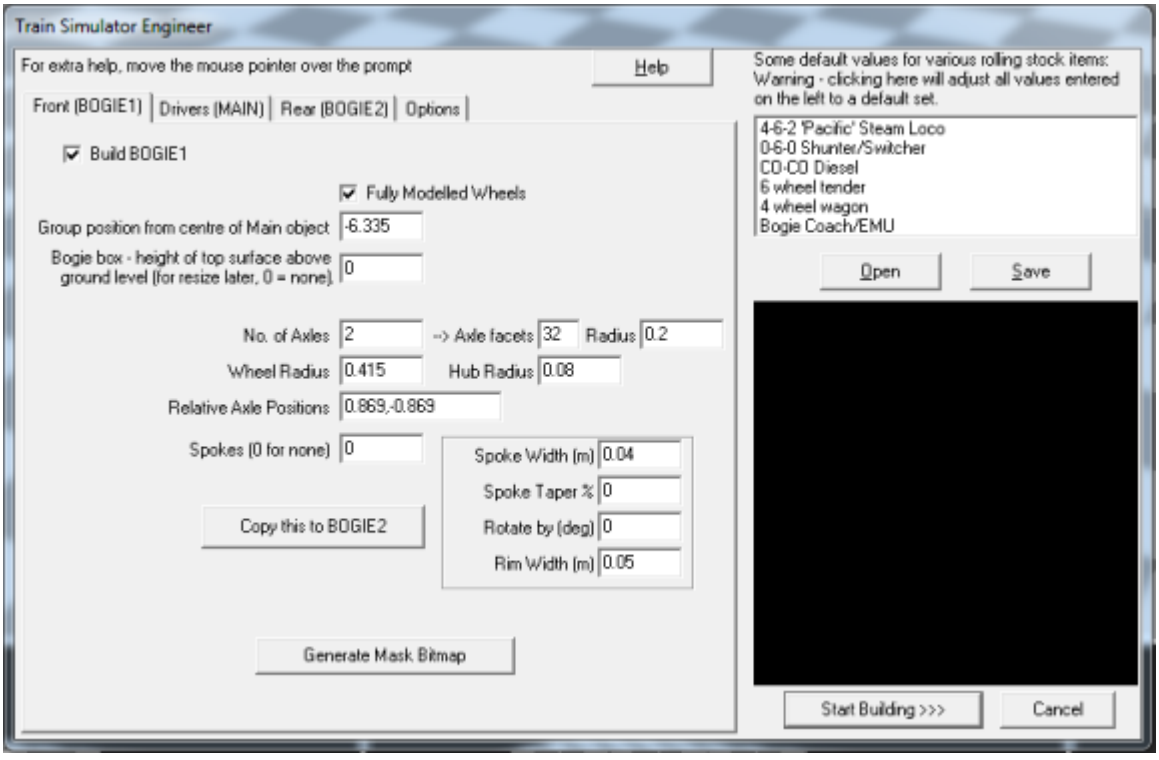
When using background images and modeling a freight car, for example, you would first figure out the distance between truck centers in meters and match the background drawing scale using the grid in the 3D view. Adjust the background image scale using the `S` and position using `G`.

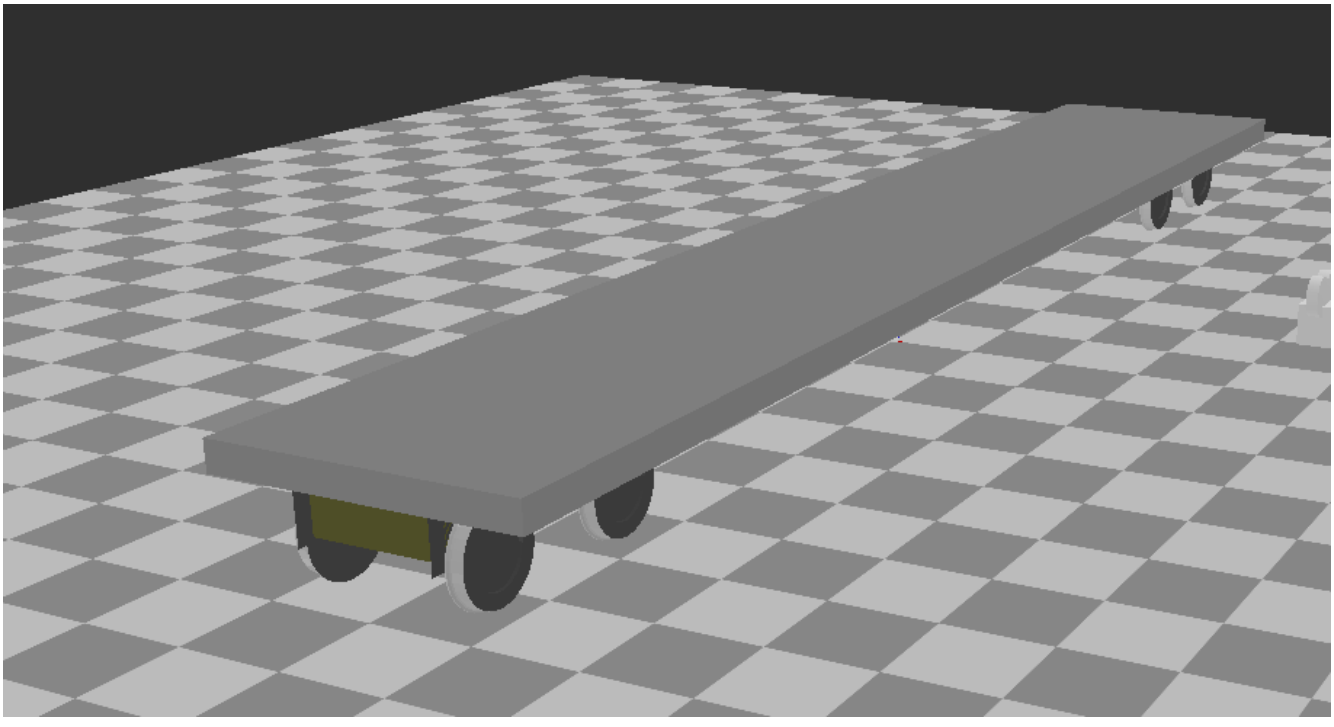
Speaking of working with images, Blender has an add on that you can install called "Insert Images as Planes". The ADD menu will now have this option available. It will assign the material of the image to the plane and allow you to rely on things like ALPHA masking. It is also affected by scene lighting, but like I mentioned above, this texture will only show up in a TEXTURE view mode, so this feature is not useful for a background refrence drawing.

Some things I will miss about using 3D Canvas

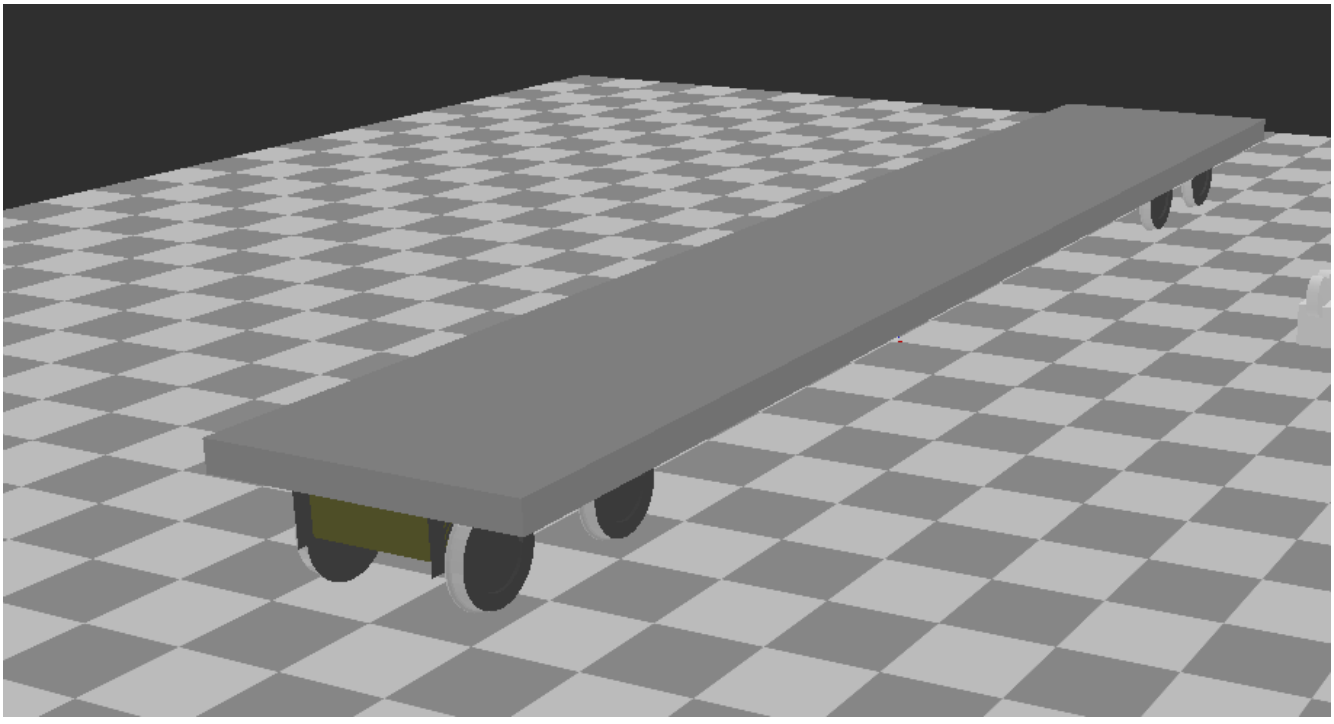
I will digress for a moment to share one of the reasons why using 3D CANVAS was so helpful in the initial stages of making a model.

3D Canvas/Crafter has a train content ENGINEER add-on `Trainworks → Train Simulator Engineer` that behaves as a quick start tool which places fully modeled wheels, axles and truck shapes and the base body element, all of which are customizable. This add-on is sorely missed in Blender as nothing even close to it exists. In reality though, it only saves you a few minutes. If you have a ready made truck-set and couplers, for example, it is pretty easy to import and place them at the correct location.





Since this is really just a set of preset shapes with customizable values and locations, it's not impossible to consider that Blender's scripting language could be used to recreate the same tool using Python. While I'm not in any position to create this Blender Script myself, it would be a welcome sight to see one or literally any thing else Open Rails related beyond the existing `.S` File Exporter we have from Wayne Campbell.



Using this plugin with 3D Canvas we haven't even started modelling yet and we have a number of things in place. In Blender, however we are going to have to create all of this content ourselves.

Something we could do with 3D Canvas was to enable "render 2-side faces" and it is not very clear how to do that with Blender. So, let me explain:

To create a double-sided plane in Blender, you can use the following steps:

1) Start by creating a new plane in Blender. You can do this by going to the "Add" menu in the 3D View and selecting **Mesh › Plane**.

2) With the plane selected, go to the "Object" menu in the Properties panel and enable the "Double Sided" option. This will cause the plane to be visible from both sides.

Alternatively, you can use a Solidify modifier to add thickness to the plane and make it double-sided. To do this, select the plane:

1) Go to the "Modifiers" tab in the Properties panel.

2) Then, click the "Add Modifier" button and choose "Solidify".

3) In the Solidify settings, you can adjust the thickness of the plane.

If you want to apply the Solidify modifier permanently to the plane, you can do so by clicking the "Apply" button. This will apply the Solidify modifier to the plane and make it double-sided.

Where to start?

I feel like we can try to make a simple scenery object as a first project and try to make it by manipulating the default cube.

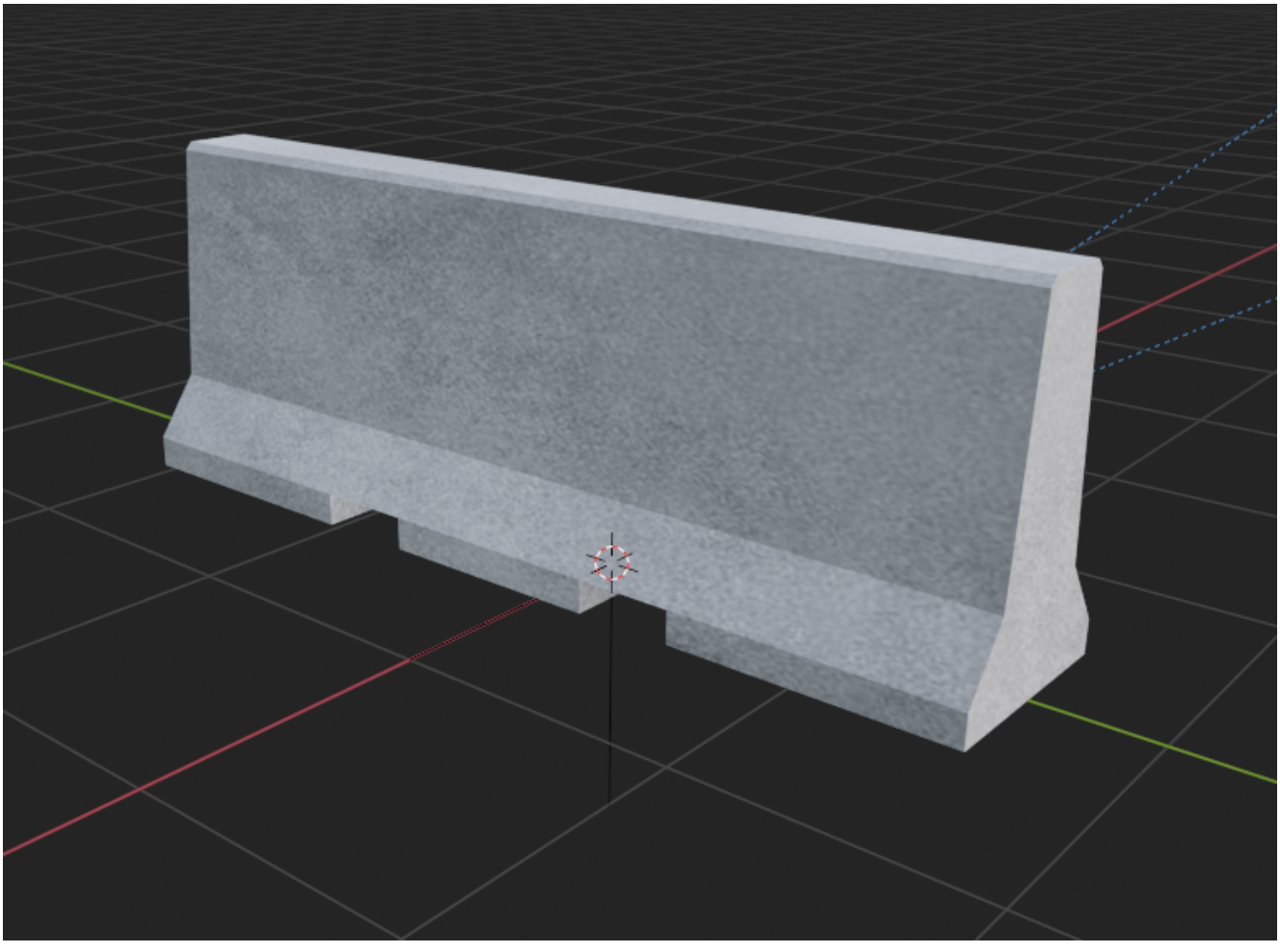
Do we need a quick 3D Modeling primer?

A basic Blender concept is that you will spend a lot of time with your right hand on the mouse and your left hand typing keyboard shortcuts. Blender is very much still a key-based program, though in recent releases with the additional of optional pie-menus and onscreen widgets, you can initiate quite a number of operations with just the mouse or a single key press and the mouse.

Every piece of 3D software I've used is based on a set of core concepts and constructs. These are often called primitives and Blender is loaded with them. By far, the most used primitive when creating new hard surface models is the `cube` closely followed by the `cylinder` and `plane`. Blender refers to these items as "mesh objects".

By adjusting the scale, rotation, and location or by adding to or subtracting from the faces, vertices and edges of these primitives, we can create the various adjustments and transformations needed to create a particular shape. It is honestly a bit overwhelming at first. The only way to learn how to become better and faster a is by **not** only watching or reading tutorials, but by just "doing it" in Blender yourself.

So, of course I mean that you should watch and read tutorials to get better acquainted with Blender features. However, you are still going to need to just try and make a few things yourself. For example, my first real export into Open Rails was a 12' long Jersey style concrete barrier. Nice and simple.



My experience has been that I begin to work on something with the complete understanding that I will trash it, or some of it, as I discover a better way to do it.

As I said, Blender isn't the easiest program to learn, but even a first-week novice should be able to produce a simple model like the one above using a few fundamental commands:

KEY	ACTION
G	Grab - To Move things. Can be followed by an Axis option and distance option
R	Rotate - Can be followed by Axis and Angle options
S	Scale - Can be included with other keys to limit scaling to desired axis

Pressing GG as in pressing the G key twice, will allow you to SLIDE move along an edge.

When scaling with the mouse, the closer your mouse is to the object, the more impact the mouse has on scale movement. If you want more control, begin the S operation with the mouse cursor further way from the selected item's origin.

These keys can manipulate basic objects. By using these keys and following them with additional commands like X, Y, or Z you can constrain actions to the specified axis. By pressing SHIFT prior to an axis key, you remove it from the list.

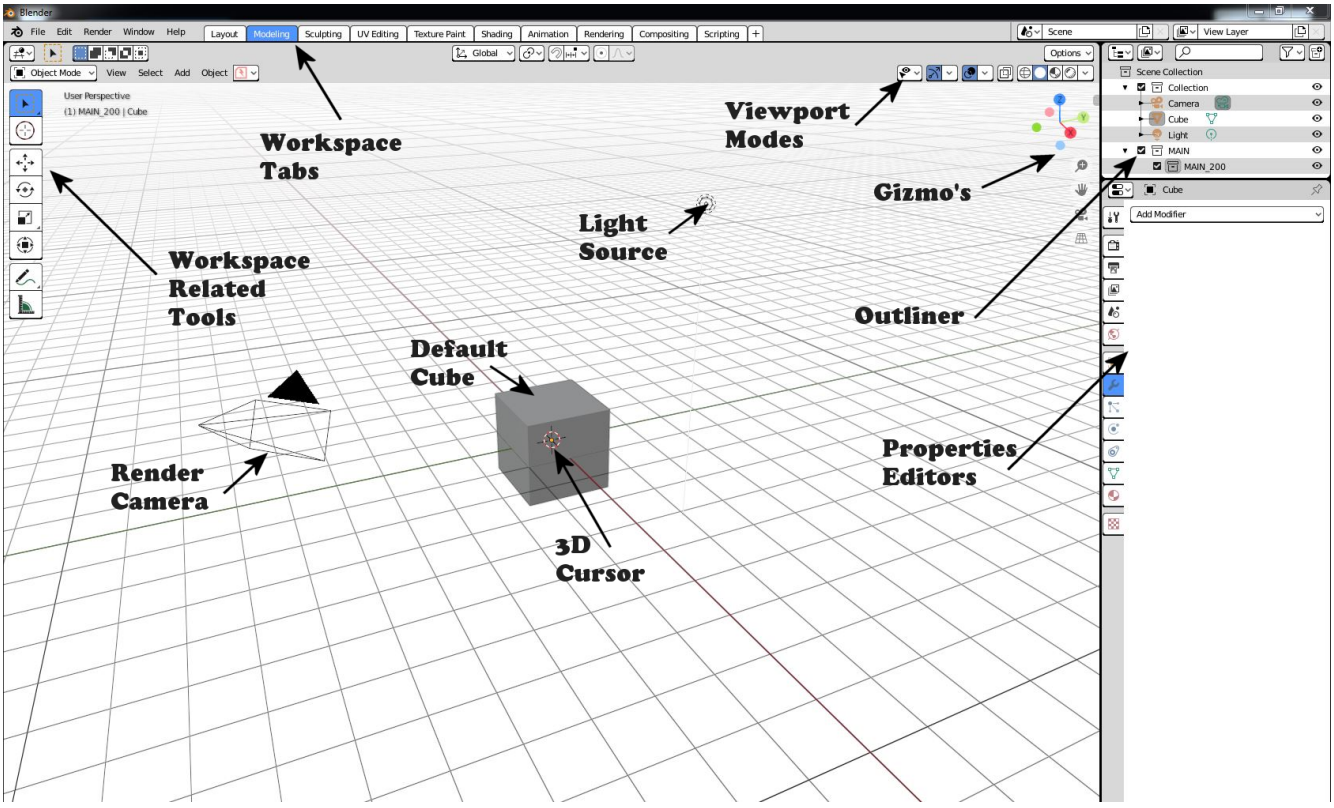
Example1

By pressing the sequence: G SHIFT X you will be able to move the selected item in the Y and Z axis, but not X.

The Modeling Interface

This is the initial default screen layout you will see when you start up Blender.

For added clarity on printed copies of this document, I've switched to a high contrast theme in Blender Preferences. (Prefereces-→Themes-→Presets[Pulldown for options])

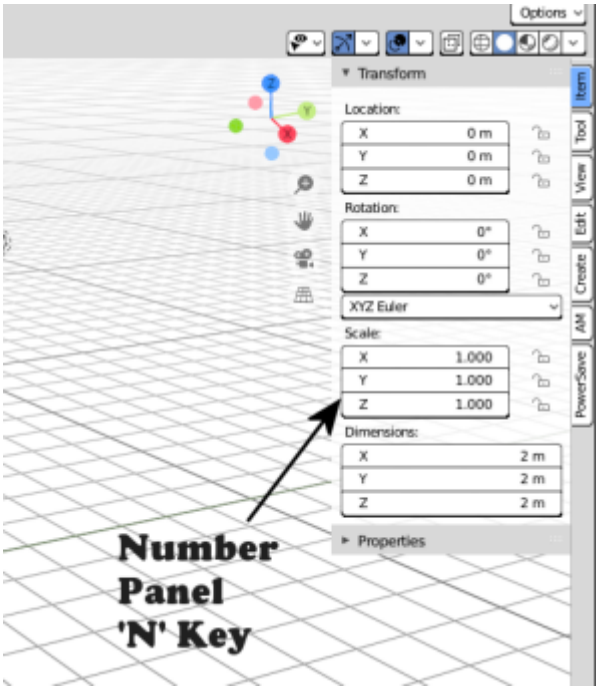


The 3D cursor is where any new object will be placed. By default it is at world origin **0,0,0** but it can easily be moved to new locations. The position of the 3D Cursor can also be manipulated using the cursor widget on the left side panel. It is available in both OBJECT and EDIT MODE. You can also place the 3d cursor position using the `Shift` + `S` **SNAP** Pie Menu.

Shortcut Keys

MENUS

To hide or reveal the Number Panel on the right side use `N` key (Not shown above)
To hide or reveal the tools Panel on the left side use `T` key
To see your "Quick Favorites" menu, use the `Q` key
Use `F3` for access to the search feature
Use `CTRL` + `TAB` combination for access to the `MODE` pie menu
Use `~` for access to the `VIEWPORT` pie menu
Use `.` for access to the `PIVOT POINT` pie menu
Use `,` for access to the `AXIS ORIENTATION` pie menu
Use `SHIFT` + `S` to access the `SNAPPING` pie menu
Use `Z` to access the `VIEWPORT SHADING OPTIONS` pie menu



The above image shows what it looks like when rolled out. It is also where some installed add-ons will show up.

VIEWS

Use `HOME` to show all objects
Use `H` to hide a selected object
Use `ALT` + `H` to un-hide all objects
Use `SHIFT` + `H` to show all objects
Use `ALT` + `Z` to toggle X-Ray mode

The Number Pad Keys are also used to select various views.

View Controls

There is a check-box option in **Preferences › Input › Emulate 3-Button Mouse** for people who do not have a Middle Mouse Button. To create a `MMB` press while this mode is enabled, you hold `ALT` while pressing the `LMB`. There is also a checkbox to EMULATE the number pad for systems that lack a keypad. This option will enable the number pad key mapping on the TOP ROW number keys instead. (This will affect some options, like switching edit modes for vertex, line and face)

Orbiting

Select the default cube and press and hold your middle mouse button, `MMB`. Moving the mouse right and left, you will orbit your view around the selected object.

Panning

Select the default cube and press and hold your `MMB` and then press the `SHIFT` key. Moving left and right will PAN left and right on the screen view.

Snapping

Select the default cube and press and hold `MMB`, then press the `ALT` key and by moving left, right, up, down you will snap the screen to various orthographic views.

Zoom

Select the default cube and press and hold `MMB` and then press the `CTRL` key. Moving up, down will zoom the views in and out. You can also use the mouse `SCROLL WHEEL`

Camera View

Pressing the `INS` key on your keypad will toggle the Camera View. In this view, you will see what the camera sees of your screen which is also what your render will output.

Zoom to Selected

Pressing the `DEL` key on your keypad will zoom in and give preference to the selected object.

Front/Rear View

`Keypad 1` / `CTRL` + `Keypad 1`

Side View Left/Right

`Keypad 3` / `CTRL` + `Keypad 3`

Top/Bottom View

`Keypad 7` / `Ctrl` + `Keypad 7`

Rotate Z axis

`Keypad 4` & `Keypad 6`

Rotate X axis

`Keypad 2` & `Keypad 8`

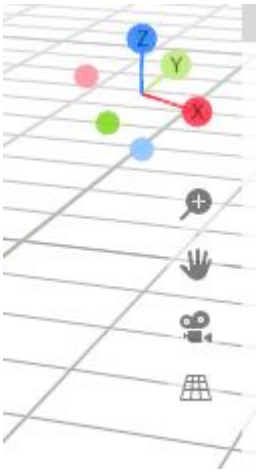
Toggle Perspective and Orthographic view modes

`Keypad 5`

Most of the time, you will be using the default `BOX SELECT` mode of the Arrow (Select) tool. Use the `B` to switch back to `BOX SELECT` if you end up changing it from default. Other modes include `CIRCLE SELECT` `C` (Note, use `RMB` to exit `CIRCLE SELECT` mode), and `LASSO SELECT` `CTRL` + `RMB`. The `W` key will cycle between modes sequentially. You can also **INVERT** selections by using `CTRL` + `I` and add to an existing selection by holding `SHIFT` and selecting with `LMB`.

BOX SELECT ICON

You can move around using the mouse with the screen gizmos (X,Y,Z)in the upper right. The multi-color Axis tool will allow you to drag to a new orientation, the Magnifier is for Zoom, the Hand is for panning , the Camera icon will toggle the Camera view and the Plane icon will toggle between Perspective and Orthographic. These on screen items are helpful when using a laptop that does not have a keypad.



ON SCREEN GIZMO

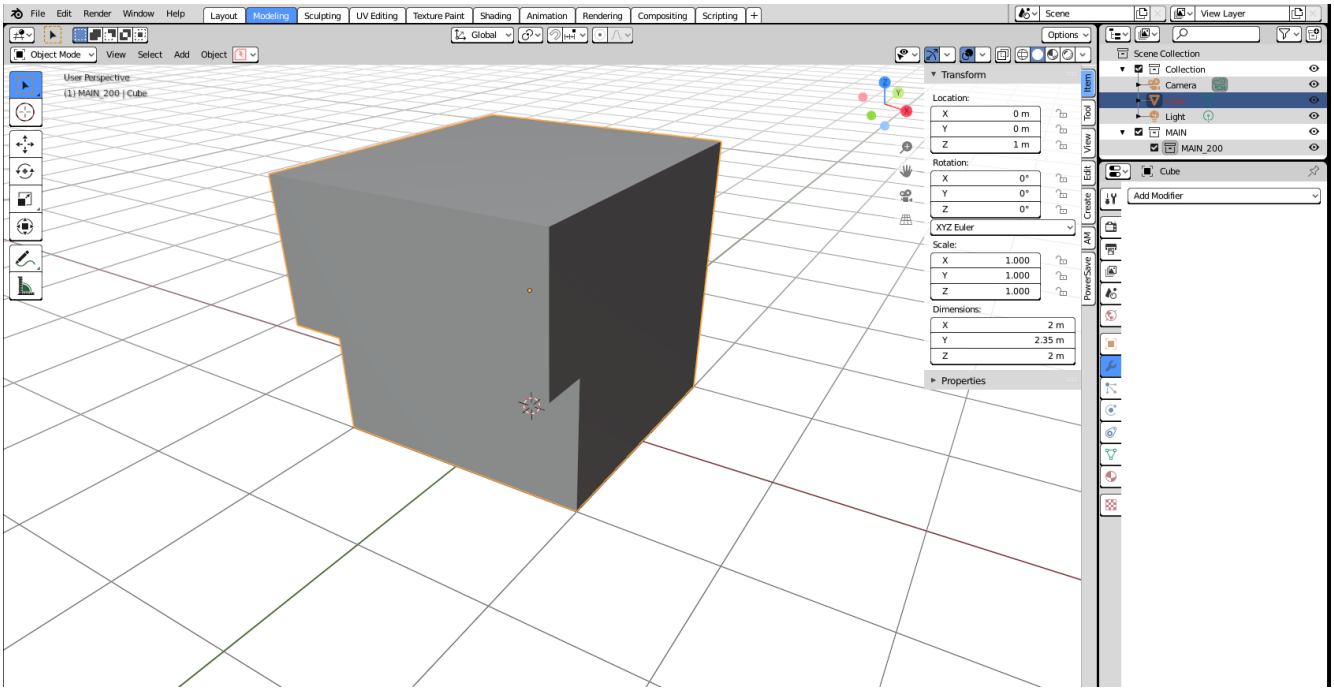
Modeling Modes

In Blender, there are 2 primary 3D model manipulation modes. These are `OBJECT MODE` and `EDIT MODE`.

Use the `TAB` key to switch modes.

OBJECT MODE

With `OBJECT MODE`, which is the default mode in Blender, actions are available for all object types since this mode is dedicated to Object data-block editing (e.g. position, rotation, size) as well as Modifiers. Edges, Faces and Vertices cannot be modified in this mode.



In this mode, you can select individual objects that make up your design so they can be further manipulated in `EDIT MODE`.

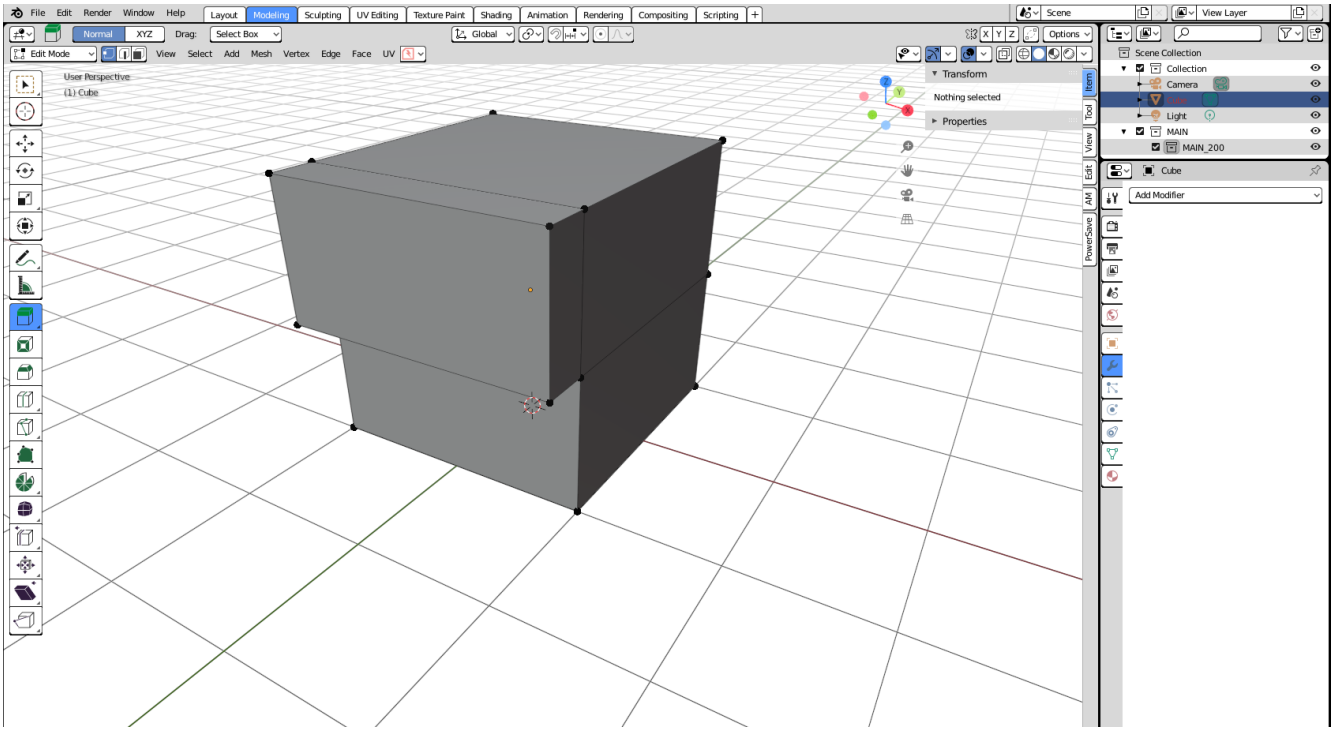
In object mode, the following shortcut keys are useful:

Use `SHIFT` + `A` to Add a new object
Use `SHIFT` + `TAB` to toggle `SNAPPING` modes
Use `CTRL` + `A` to Apply transformations
Use `CTRL` + `J` to Join objects together
Use `SHIFT` + `C` to reset cursor to center
Use `SHIFT` + `D` to Duplicate the selected object

EDIT MODE

The **selected** item in OBJECT MODE becomes the focused object when moving to EDIT MODE. EDIT MODE is a focused mode and you will not accidentally select other parts of the model in this mode. This mode available for all object types that can be rendered, as it is dedicated to manipulating their "shape". The EDIT MODE allows adjustment of Vertices, Edges and Faces for mesh object types as well as the control points for curves, surfaces and points.

In EDIT MODE, the object selected will show Vertices, Edges and Faces. Selected edges, vertices or faces will adopt a highlight color when selected. Also notice the larger tool set on the left compared to OBJECT MODE.



In Edit mode, the following shortcut keys are useful:

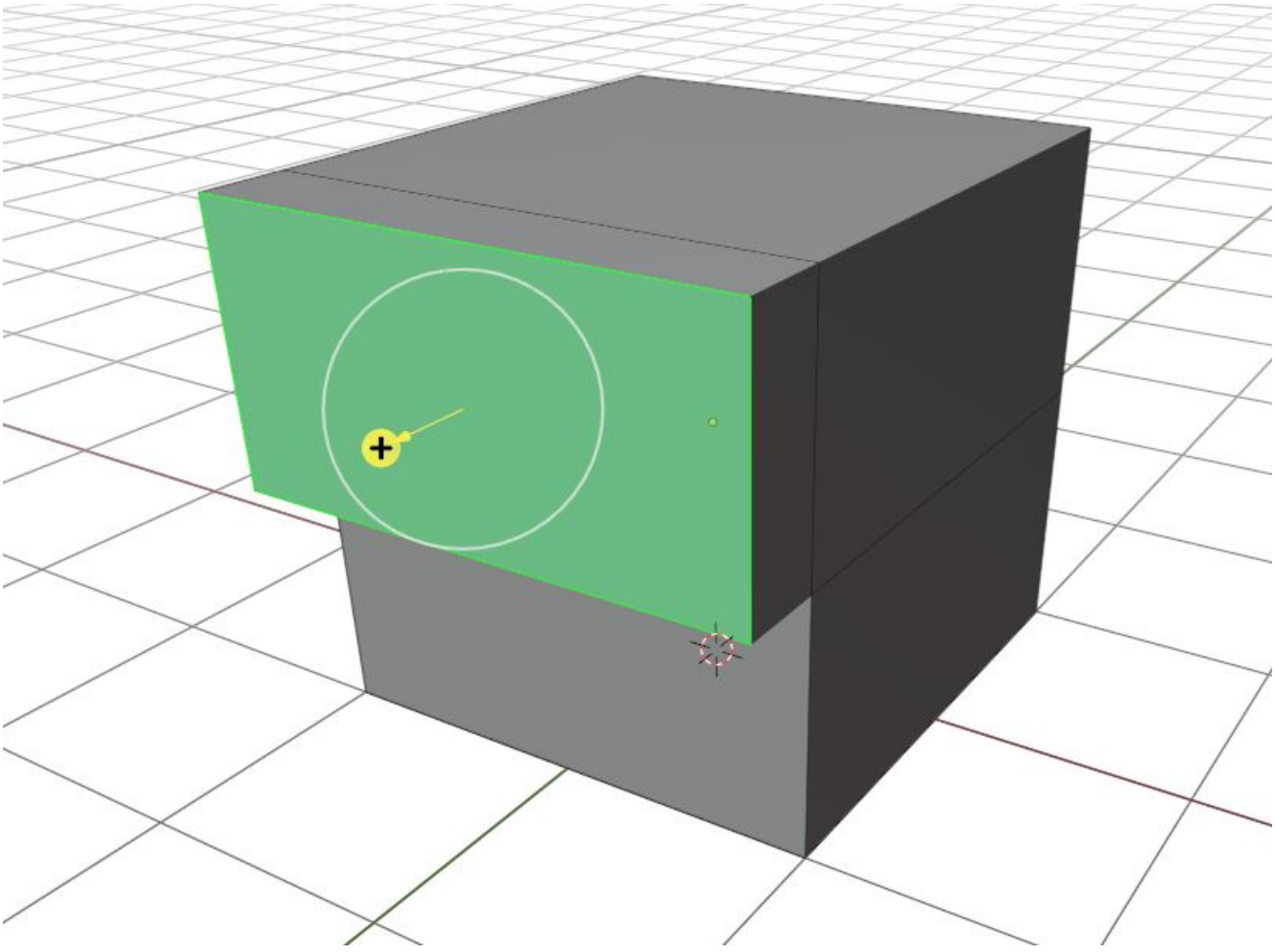
Use 1 Vertex Mode
Use 2 Edge Mode
Use 3 Face Mode
Use P Create Separate objects from the selection in various ways
Use F Fill Face

There are 3 sub-modes in EDIT MODE; 1 Vertex Edit, 2 Edge Edit and 3 Face Edit.

The 4 major edit-mode tools you are likely to use the most are listed below.

There are now thousands of YouTube videos about modeling in in Blender if any of these concepts mentioned here are not clear.

EXTRUDE



`E` *Key*

The official definition of the extrusion operation is: *The* **`extrusion`** *operation duplicates vertices, while keeping the new geometry connected with the original vertices. Vertices are turned into edges and edges will form faces.*

There are various options with extrusion operations that define how the extrusion will behave, these include "Extrude Region", "Extrude Individual", "Extrude Edge". Extrude works by shifting position along "Normals".

When using EXTRUDE and you don't get the behavior you wanted, try using `ALT` + `E` to get the Extrude Options pop-up menu. Try the other EXTRUDE options.

When using EXTRUDE, you can extrude to the mouse gizmo location (MOVE MODE) in by using `CTRL` + `E` and right click. This object tool is directly under the ARROW SELECT tool.

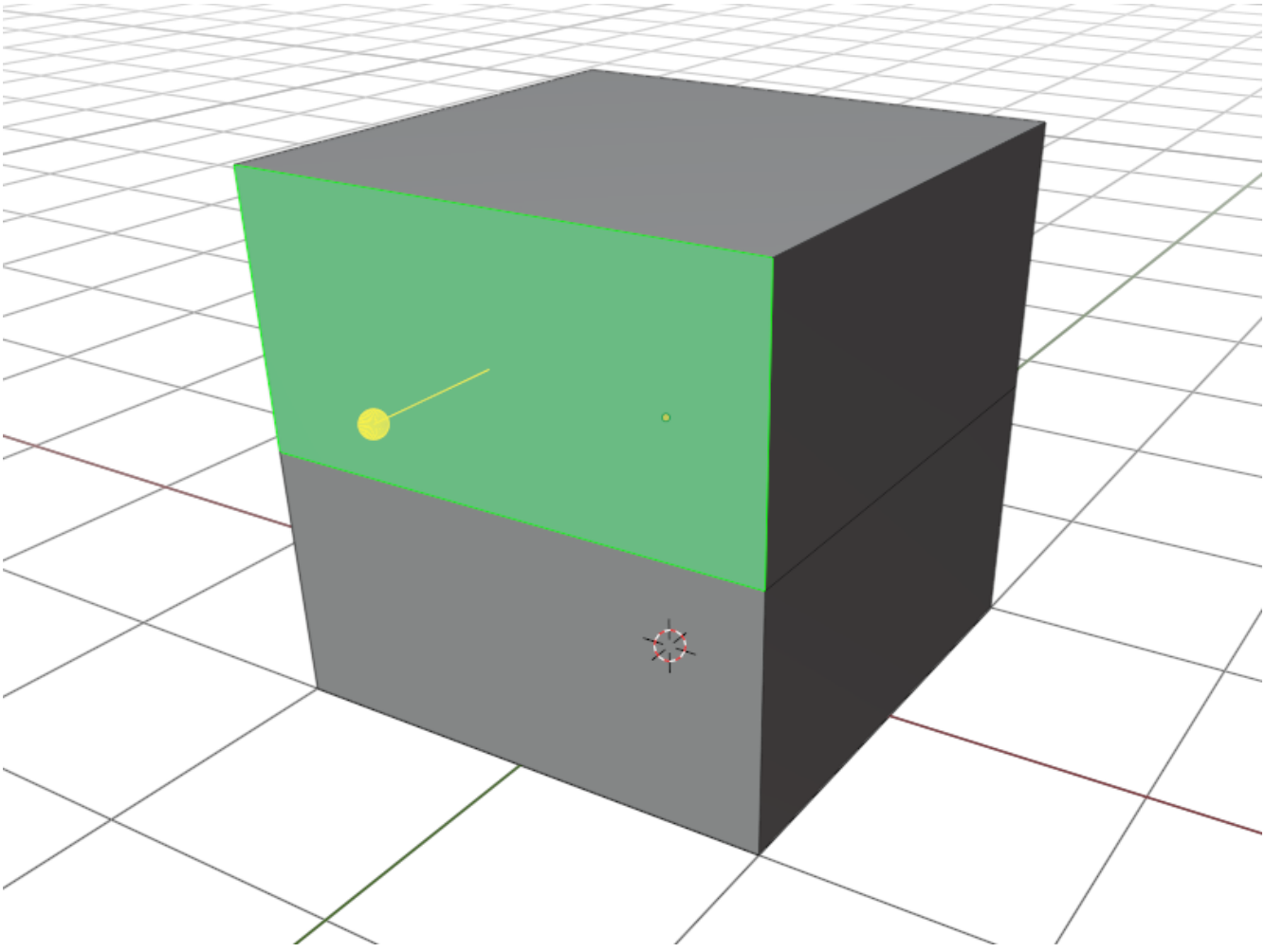
If you select extrude and accept the extrude without dragging a distance away from the selection, the extrude "still happened" even though you might not be able to see it.

INSET

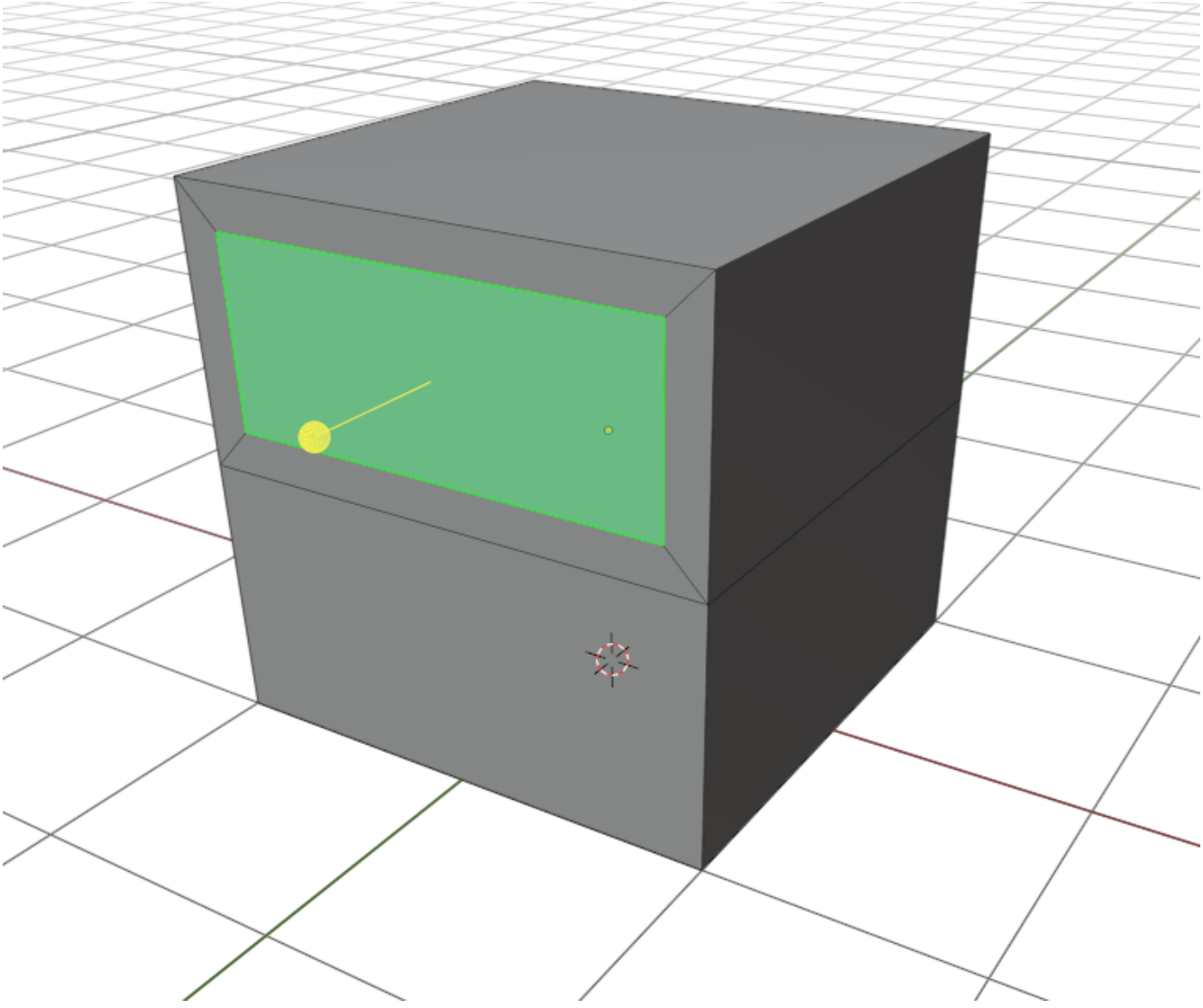
I *Key*

This tool takes the currently selected faces and creates an inset of them, with adjustable thickness and depth. (For clarity, we are referring to the letter i on the keyboard)

- Select the faces to inset:



- Press I to inset:



When you use inset and the inset amount appears to be unevenly applied, you likely have not applied your scale transformations to the underlying object ahead of time. In other words, your scale values in the object transformation numbers panel are not all set to 1.00. You correct this in `OBJECT MODE` by selecting `OBJECT→APPLY→SCALE` from the top menu.

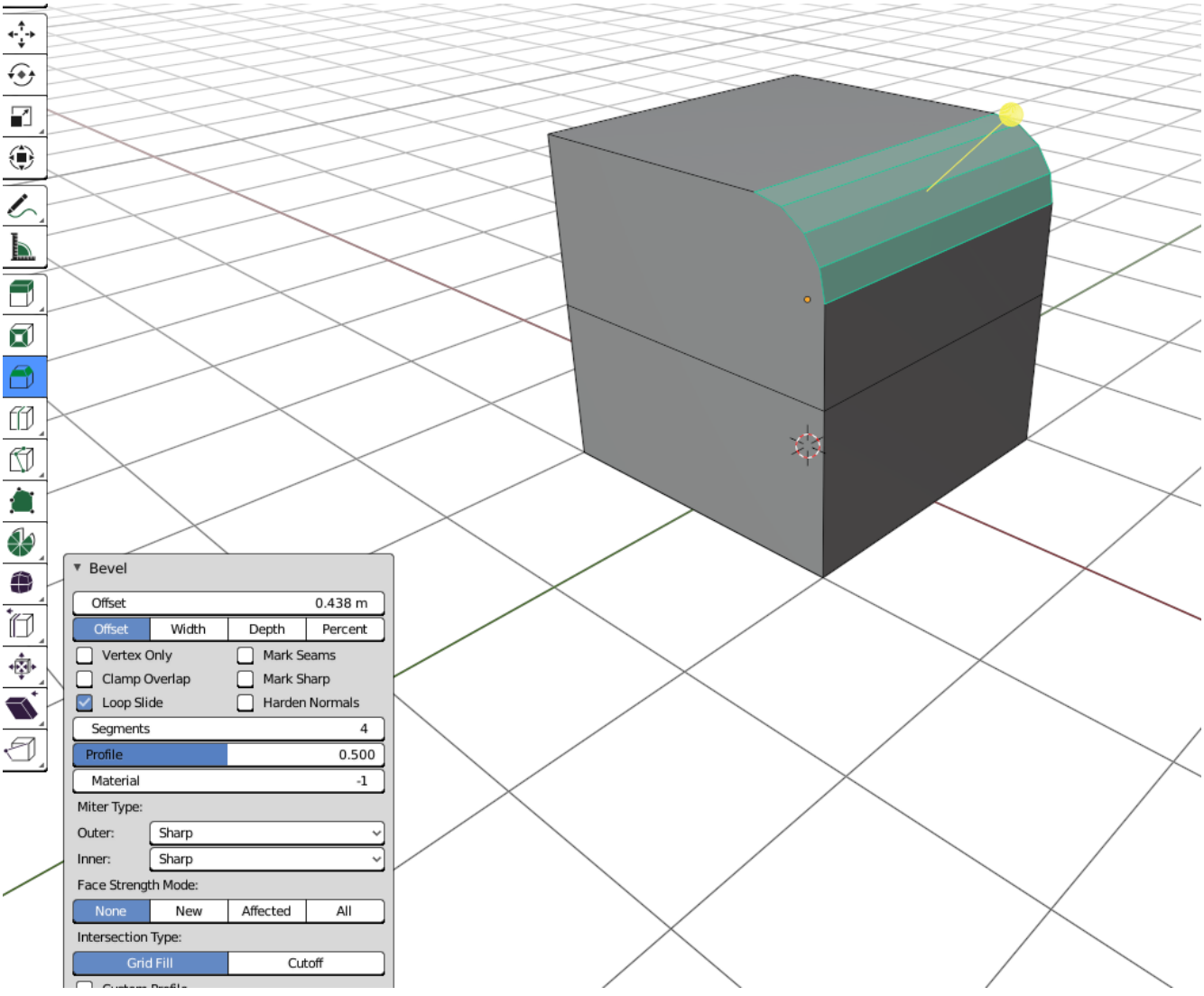
You might need to press `I` again to enable INDIVIDUAL ORIGINS if you notice that inset is not doing what you expect.

BEVEL

CTRL + B *Key Combination*

The Bevel Edges tool works only on selected edges with exactly two adjacent faces. It will recognize any edges included in a vertex or face selection as well, and perform the bevel the same as if those edges were explicitly selected. In "vertex only" mode, the Bevel Vertices tool works on selected vertices instead of edges, and there is no requirement about having any adjacent faces.

The Bevel tool smooths the edges and/or "corners" (vertices) by replacing them with faces making smooth profiles with a specified number of segments.



A Bevel on a FLAT PLANE will create an INSET.

LOOP CUT & SLIDE

`CTRL` + `R` *Key Combination*

This tool splits a set of faces by inserting new edge loops intersecting the chosen edge(s). It will preview the loop cut as you move the mouse cursor around the object, snapping from horizontal to vertical based on position. The Loop-Cut will stop at NGON Intersections so it is better to use this tool early before you add a lot of geometry changes or booleans. If your mesh has non-NGON shapes, IE; it has been triangulated already… then this and other some other tools won't work as expected.

Press `E` if you want it to evenly match one of the adjacent edges.

Press `F` will flip the sdge it matches

Use the Scroll Wheel on your mouse (before pressing `E`) to add or remove additional cuts.

After **`[Left-Clicking]`** the mouse, you can slide the loop cuts up or down and **`[Left-Click]`** again to confirm. `Right-Click` center the Loop Cut.

The operation has an "Adjust Last Operation" panel where you can still adjust some values. Use **`[F9]`** to open this option if you don't see it on the screen.

For more precise adjustments, you can turn on the "Edge Length" feature in "Viewport Overlays" to visually see the edge lenths.

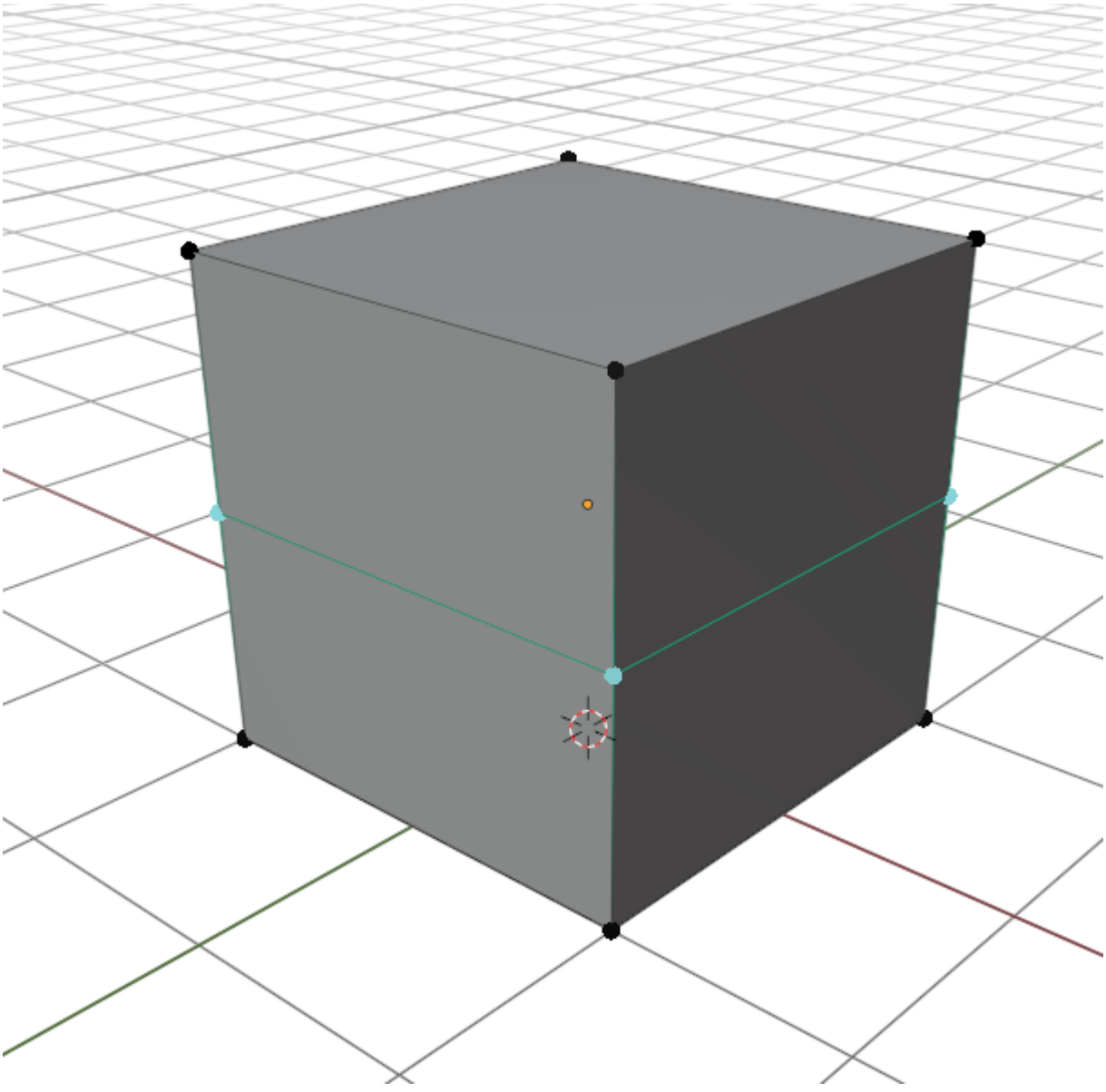
The "smoothness" value will scale the selected loop cuts using a fallof profile.

You can use the function `CTRL-F` or Search for "Tris to Quads" to un-triangulate a shape to possibly regain the ability to use the Loop Cut options again.

Later on… you can also shift selected vertices or edges with `GG` (G twice) to slide the selected vertices along edges… (You can do this with faces as well, but, well, its usefulness is limited.)

You can use the mouse to SLIDE the the selected edge loop(s) into position before confirming. You can add multiple edge loops at once by using the scroll wheel on the mouse or by entering a value with the keyboard before confirming.

You can also use the options box that shows up in the bottom left of the screen to adjust parameters manually.



Additional Tips

Here are some tips from some lessons I've learned while using Blender.

MIRROR and FLIP

Sometimes, when working on a vehicle model particularly, you might find that using the MIRROR option is not enough. You might also need to FLIP the item along an axis. For example, lets say you are adding coupler lift bar to your model and you have finished adding it to the rear end of your model. You now want to do the logical thing and mirror it to the opposite end of your model. But, mirror alone will have the part end up on the opposite end of the model, but on the same side of the model. It needs to also be flipped.

STEPS:

- On your active part (coupler lift bar), `CTRL` + `A` and choose, "Apply Rotation and Scale"
- `RMB`, from pop-up menu. choose "Set Origin to 3D Cursor", assuming 3D Cursor is current World Origin. If not, it should be for this.
- `Shift` + `D` to duplicate the current object, then chose **Object › Mirror › Global Y**
- And finally, flip it. `S X -1` which translates to `SCALE`, in the `X` axis, amount `-1`, which will FLIP the object in the X-AXIS.

You might be tempted to try to use the Mirror Modifier, however, that operation will ultimately LINK the objects and the option within the modifier that says "FLIP X" actually actually doesn't do what you might think. The FLIP option here changes the active side of the Mirror.

Additional Modeling Tips

- Model one side of the mesh and mirror it to the other size using the Mirror Modifier. This helps to make sure you get symmetrical results
- If you have an item that will be replicated many times, like a handrail stanchion, create one, UV Map it and then replicate it. You can shift the UV coordinates later, but you only need to uv map it once
- Add MARK SEAM and MARK SHARP settings while you work instead of saving these steps for later
- Export your model as soon as you get the initial UV mapping completed so you can find issues earlier rather than later as a time saver
- Remember that you can `hide` and `un-hide` collections and objects in the Outliner to help with focused modeling
- Remember to use `Smooth Edges` and `Auto Smooth` settings

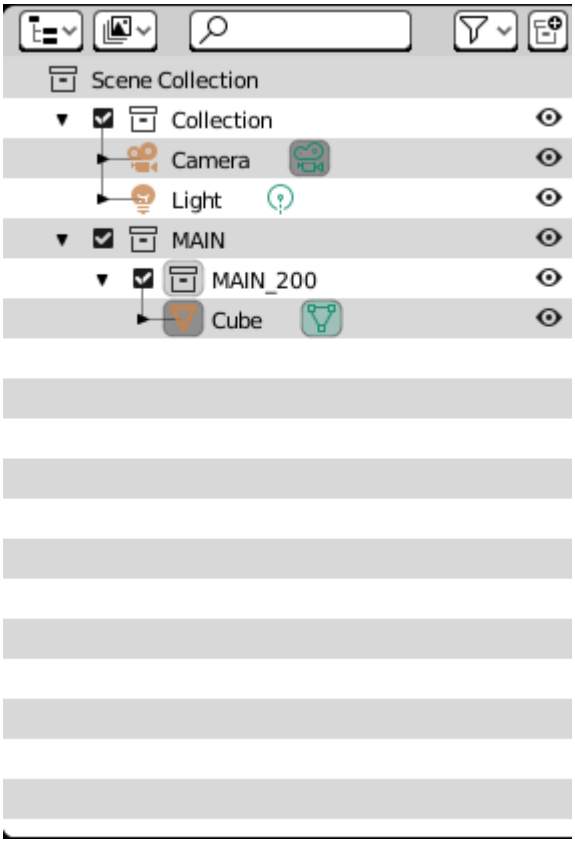
Setting up your Initial Workspace

You have a way to set some custom options that you always need to change and then save that as default to avoid having to perform those changes when you start a new project.

The default workspace in Blender 2.8+ is something you will see in a lot of in YouTube tutorial videos. The first thing they will often do is select and then delete the "default cube". Rather than go through this every time, it is possible for you to delete it, and then save your current cube-less `.blend` file as your new startup file. (I don't actually recommend it though)

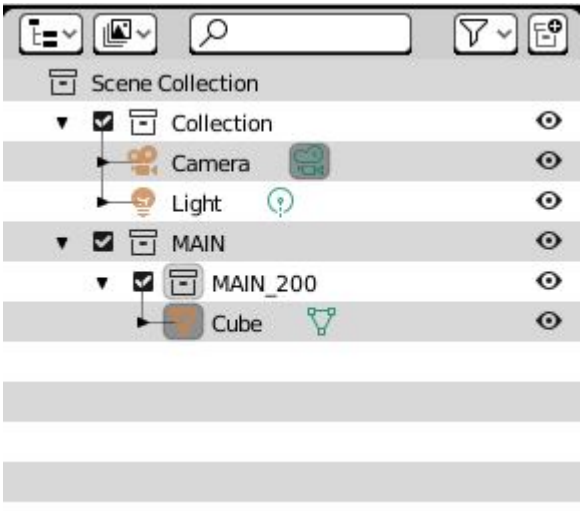
`FILE→DEFAULT→Save Startup File`

Before you do that just yet, go over to your `Outliner` window… and do the following:



1. Create a new collection by clicking the `box icon` in the upper right `Collections` window with a plus sign on it. Create a new collection called `MAIN`. (All uppercase)
2. Click the new collection `MAIN` and then create a new `Collection` again so it becomes a child collection under `MAIN` and call it `MAIN_0700` for LOD distance, or use any LOD distance value that makes good sense to you.
3. Now click on `SCENE COLLECTION` at the top and rename the original main collection to `Camera`.
4. Drag and Drop the Camera object into `Camera`.
5. Now you can save your `.blend` file as your default startup file and you will have the scene outline setup that will work with the MSTS exporter.
6. Optional: If you are going tocontinue to use the default cube, drag it from where it is to the MAIN_0700 collection.

You would later create as many LOD based MAIN_xxxx collections as needed for your model.



```
You will want to download and install the "Blender 2.8 to MSTS Exporter".
https://github.com/pwillard/Blender_MSTS_ORTS_Exporter/blob/main/Blender_MSTS_ORTS_Exporter.zip

The Documentation is included in the ZIP file.  The documentation is also available separately at this
location: https://github.com/pwillard/Ebook-MSTSORTSExporter/blob/main/MSTSORTSExporter.pdf  Use the
DOWNLOAD button to get a readable local copy.
```

Actually Modeling Something

This is a quick tutorial on general model building. To get started, we are making a very simple scenery item.

Model Building Exercise #1

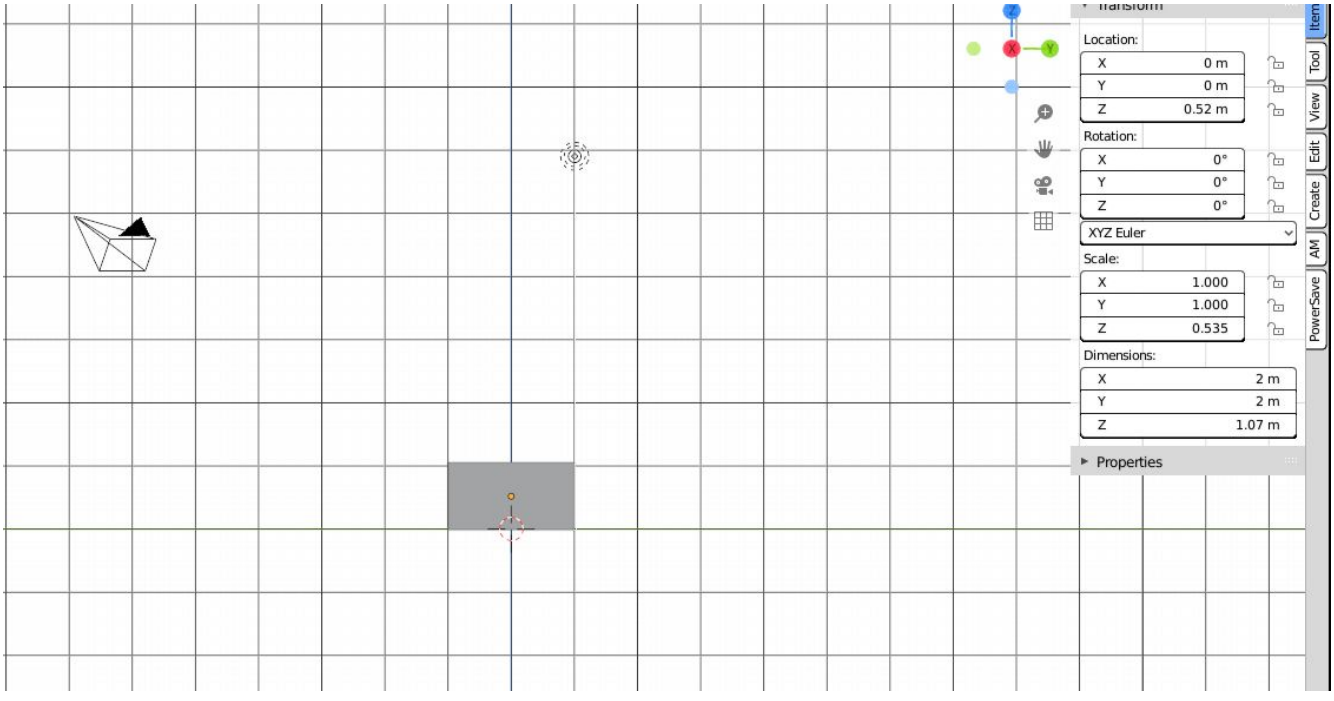
- We are building a very simple shape without using a background image.
- We only need a few general dimensions.
- We will re-make my first Blender project for MSTS; The Jersey Barrier, 12', by 42" by by 32".

Key items to take away from exercise #1

- Object and View Manipulation
- Basic Edit Tools
- Quick UV Texture Mapping

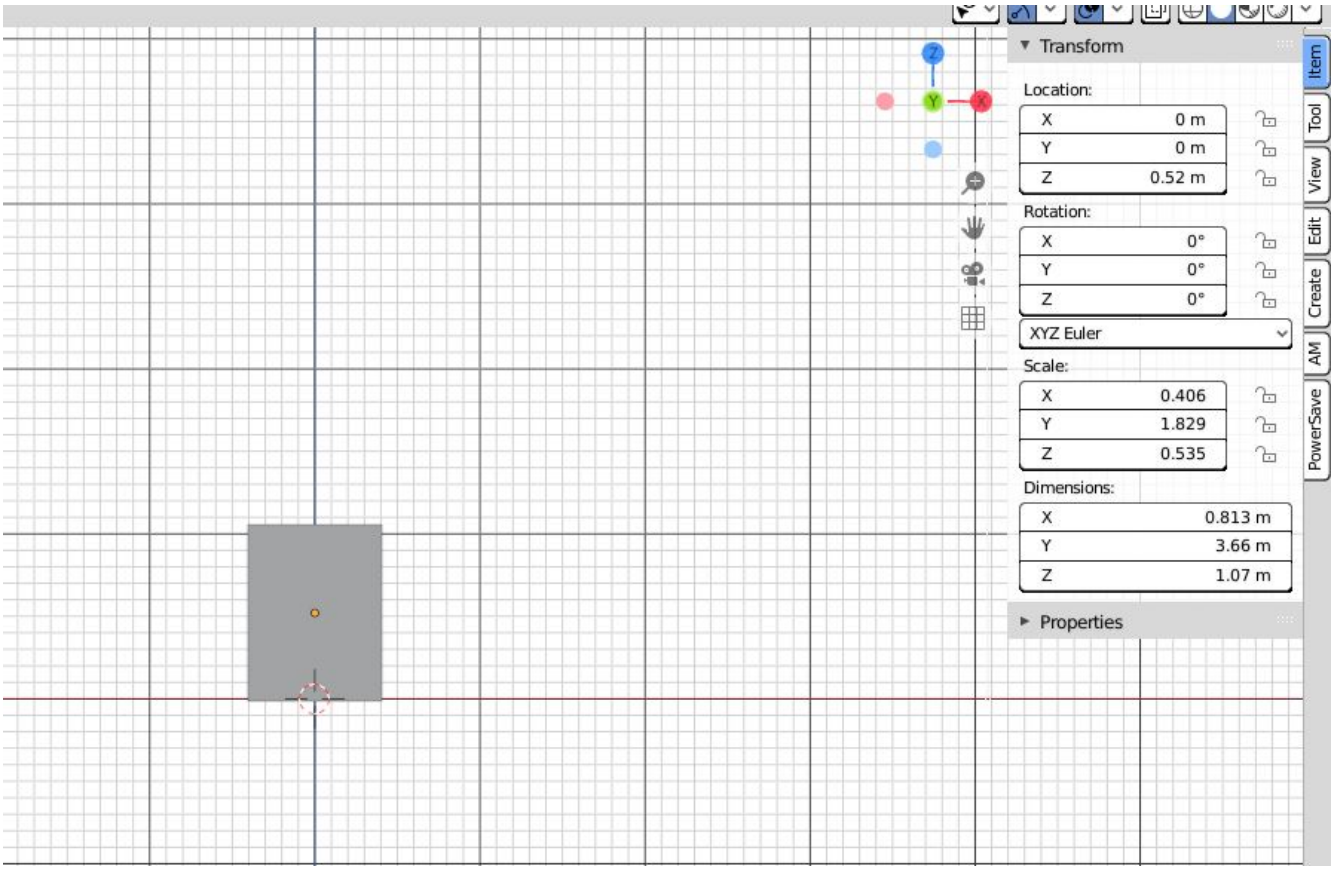
For added clarity on printed copies of this document, I've switched to a high contrast theme found in Blender Preferences.

- Start with the default cube, and select it with `LMB`.
- Shift it up 1 meter. `G` `Z` `1` `ENTER`
- As an aid to modeling, pull out the right side number panel by pressing `N`
- Scale to 42" (Z Height = 1.077m) Manually enter 1.07 in the Z dimension field.
- Shift it back to ground level. `Keypad 3` on keypad for side view, `G` key `Z`, drag down to about ground level (Roughly: Locaton 0.52m in Z axis if you manually enter the Z position)



- Adjust your view so you can see the Y axis `MMB` + `DRAG` left to right (You can press "Y" in the axis gizmo on the upper right or use `KEYPAD 3`)
- Adjust cube length to 12', (3.6576m) by using `S` `Y` and dragging, or by manually entering 3.6576 in the Y Dimension field
- Switch to front view and adjust X width to 32" (0.8128m) `Keypad 1` then `S` `X` and `DRAG` or enter .8128 in the X Dimension

field.

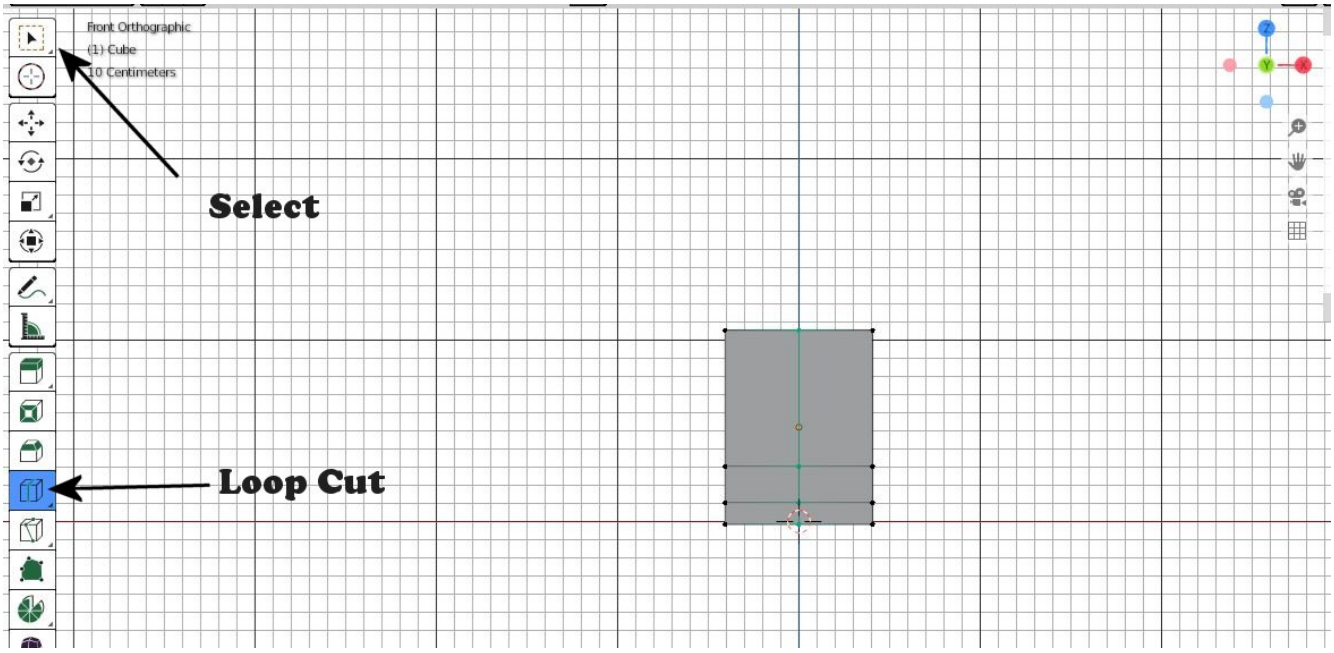


- Choose **Object › Apply Scale**
- With the CUBE still selected, switch to EDIT MODE using the TAB KEY.

We are doing this without worrying about EXACT dimensions just to keep things simple.

- Add a LOOP CUT CTRL + R and slide it down (Z axis) to the grid line closest to the bottom, and add another LOOP CUT CTRL + R and drag it down to be 2 grid lines above the first one.
- Add a final LOOP CUT CTRL + R but this time, add it vertically. It should end up dead-center by default.

Drag the mouse around to get it to snap to a vertical loop.

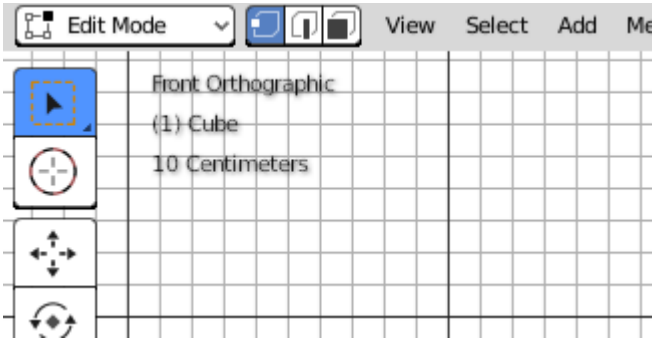


- Go back to SELECT Mode by clicking the Arrow icon if its not already selected since we are done with LOOP CUTS for now.

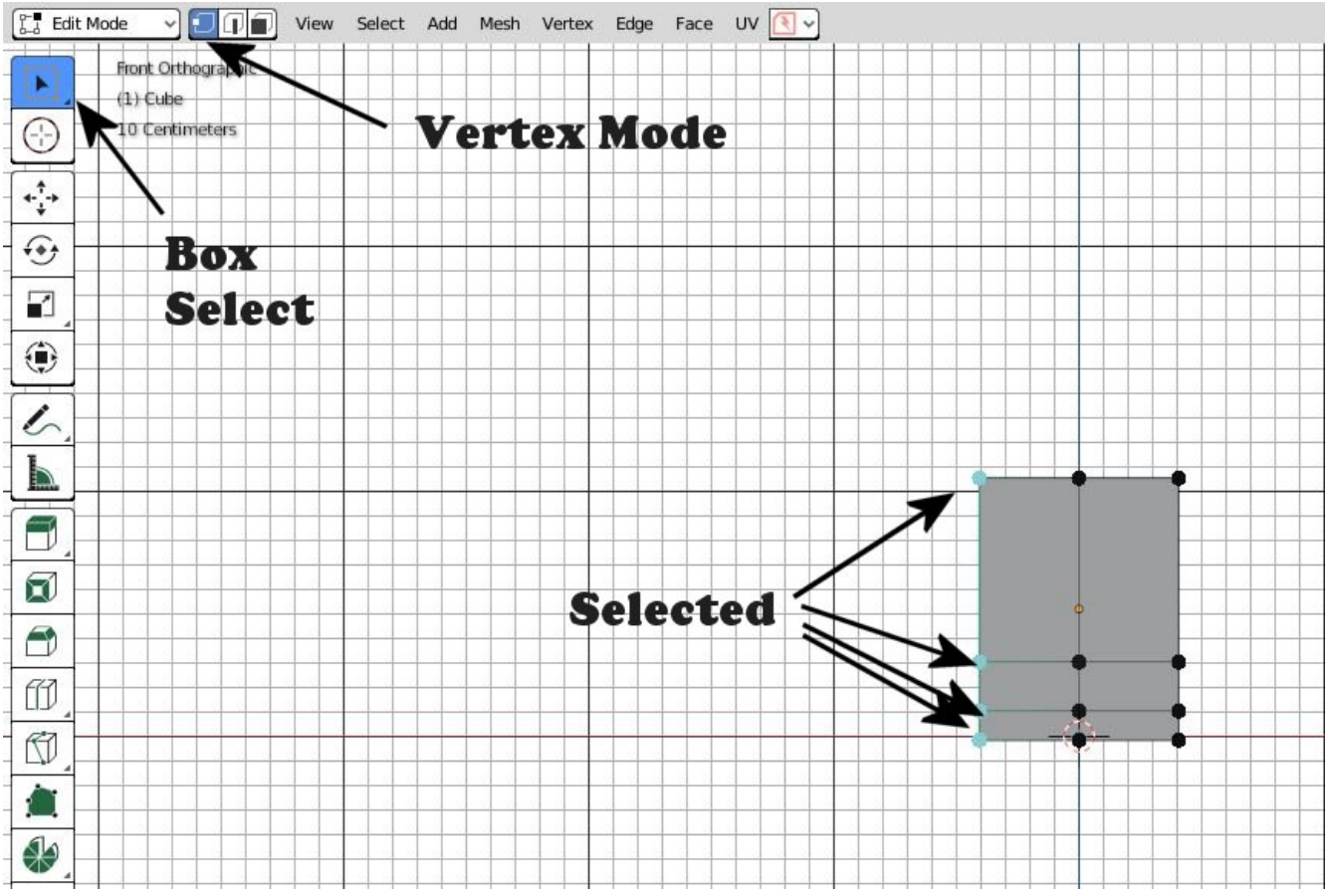
In these next steps, we will be using Vertex EDIT MODE. From the front view, we will DELETE the vertices on the left side of the object because we are going to use the Mirror Modifier to create a symmetrical object.

- Press KEYPAD 1 for front view, and then 1 on the keyboard to select Vertex EDIT MODE. You should see the vertex dots on the selected object.

You can also select the vertex mode with the screen menu. Its the small square icon with a dot on one side next to the view tab.

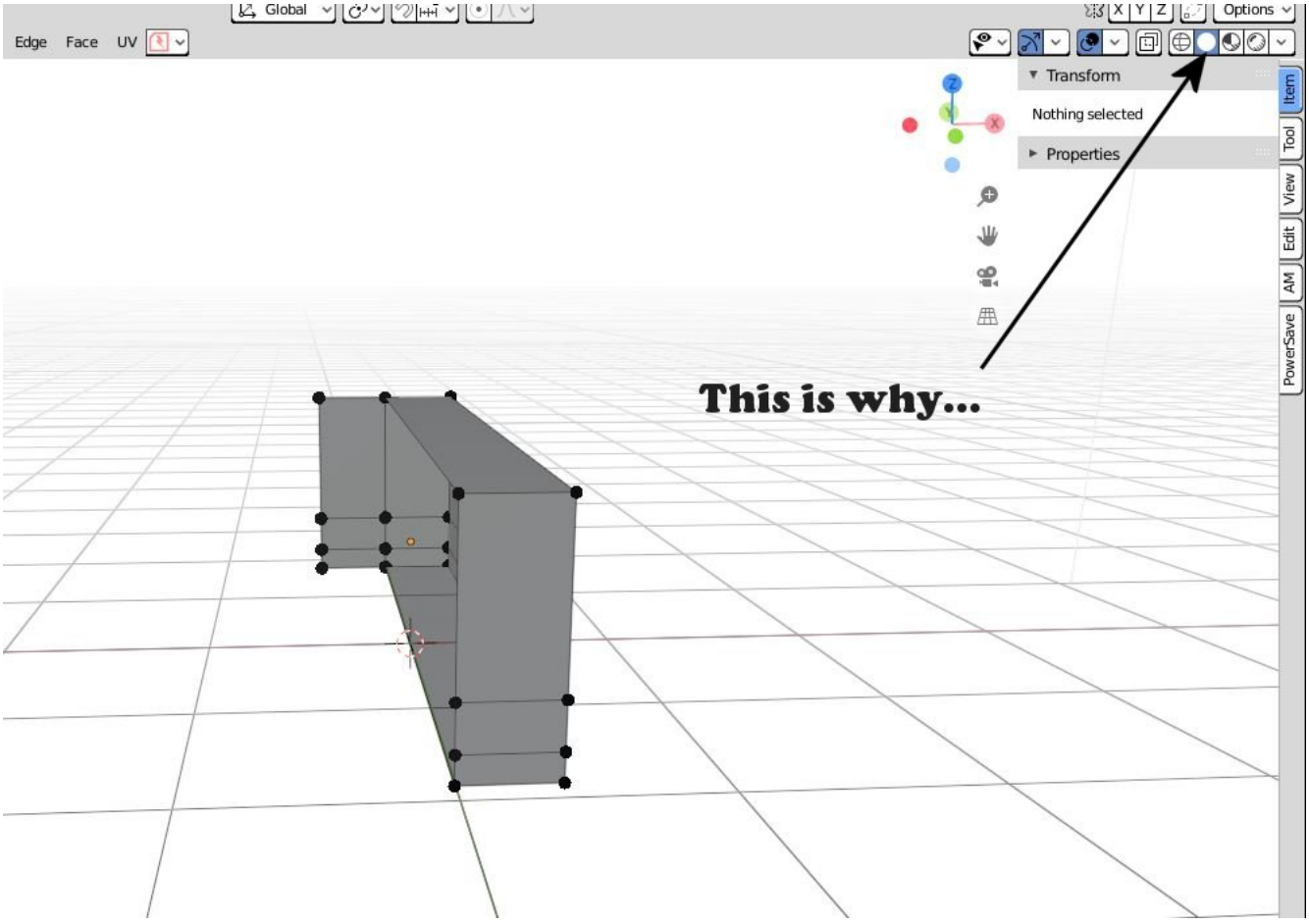


- Now use BOX select on the vertices on the left side with your mouse. (They will change to the SELECTED color)



- Press `X` for Delete and in the pop-up window, choose to `Delete - Vertices`. Blender will delete the selected vertices. Wait… What just happened?

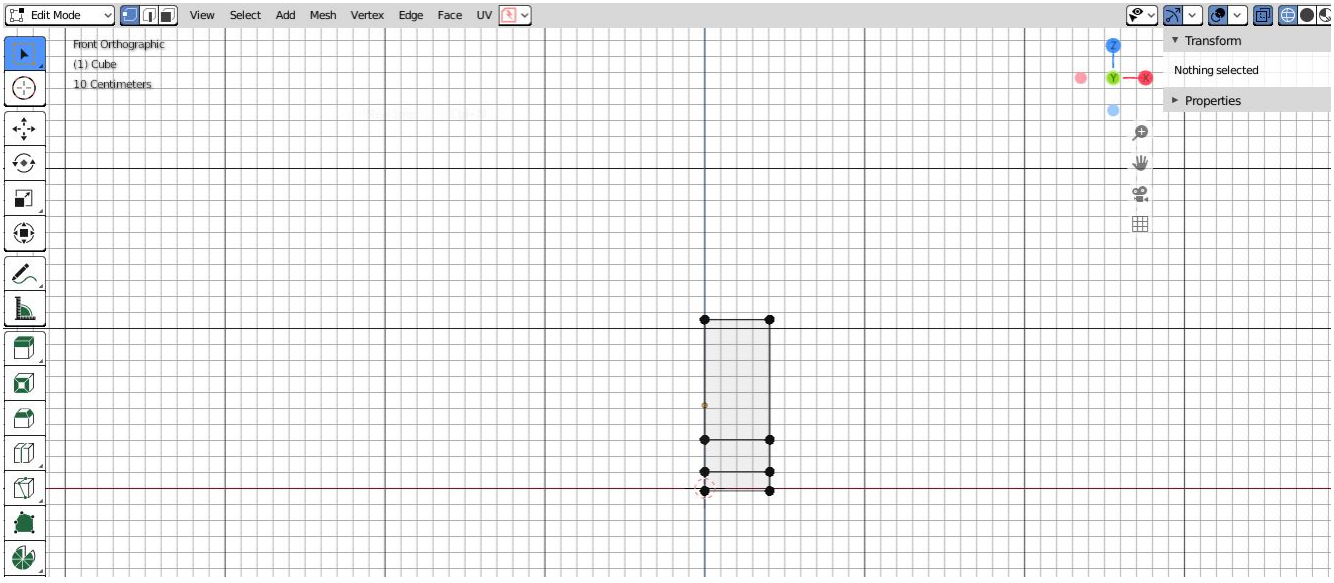
It didn't perform a delete? Oh my, yes it did, but not what we wanted!



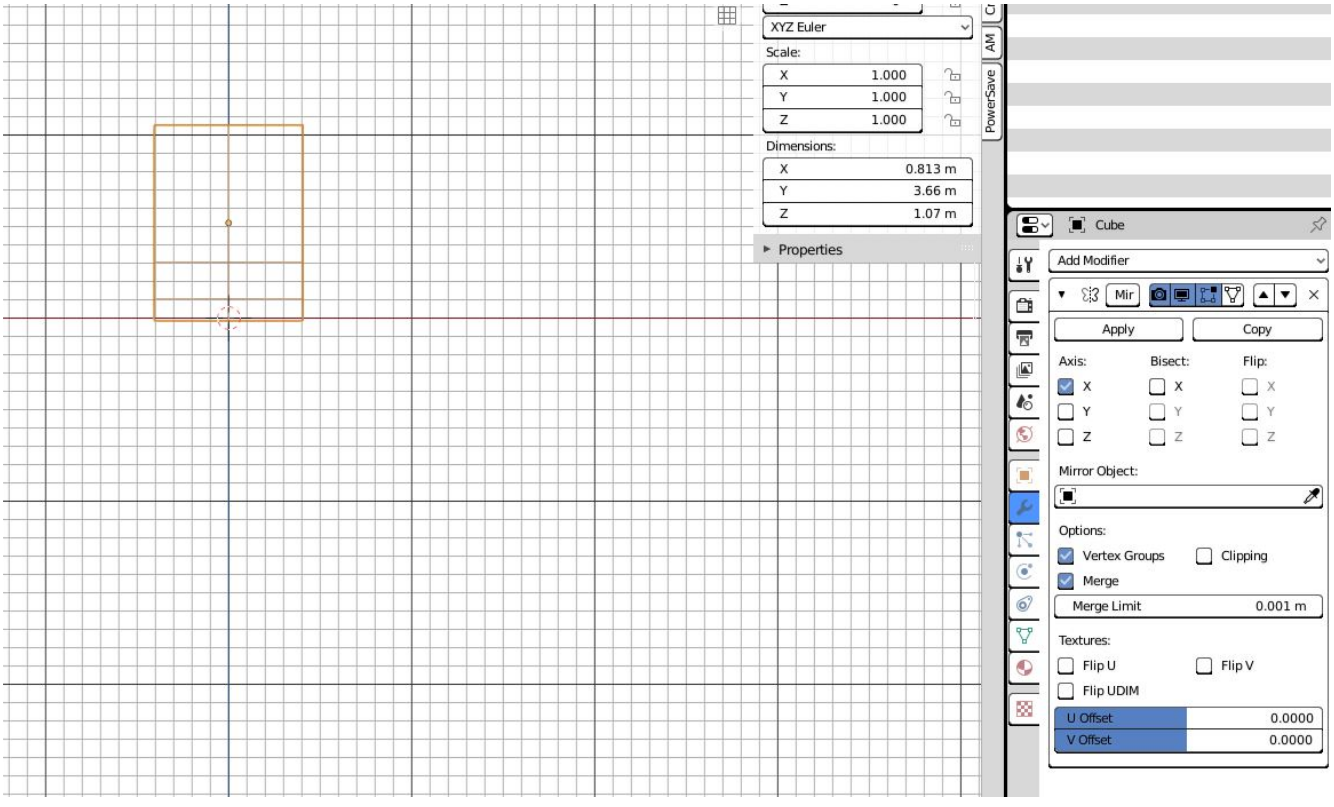
We are in ViewPort "Shading Solid" Mode.

Do you understand what happened? We only selected the FRONT facing vertices! We didn't touch the ones in the back. Press `CTRL` + `Z` to undo if you completed the above step. To select ALL of the vertices that we really want to select, we need to be in **XRAY/Wireframe** mode. To chose this mode, press `Z` and chose **WIREFRAME**, making sure that the viewport mode on the top right of the screen agrees. You can also toggle Wireframe mode by using `ALT` `Z`. The Circle with LINES in it and the **X-RAY** icon next to it should also be highlighted.

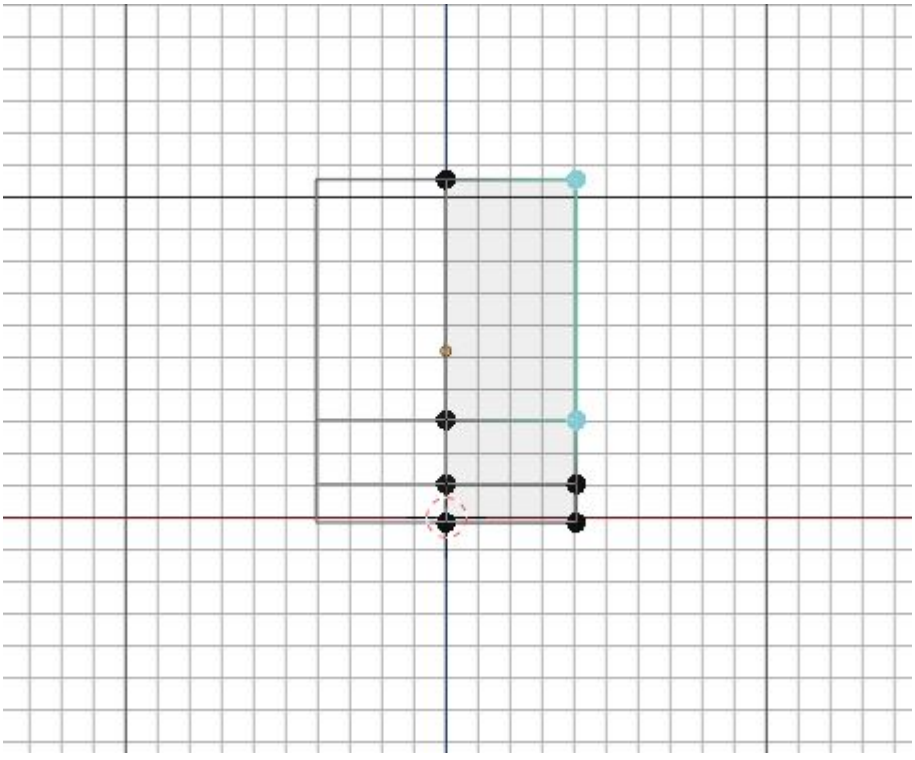
- Make sure you are in vertex select mode `1`, and also in the front view `1` we will remove the left side vertices. Press the `Z` key and select **WIREFRAME**, Press `Keypad 1`, and then `1` on the keyboard to select front view & vertex mode. You should see your vertex dots and the model will look transparent now and not solid.
- Now, **BOX** select the vertices on the left side, like before. (They will change to the SELECTED color) and press `X` and choose to Delete Vertices. Blender will delete the selected vertices. Now, you will finally only see 1/2 of your object remaining.



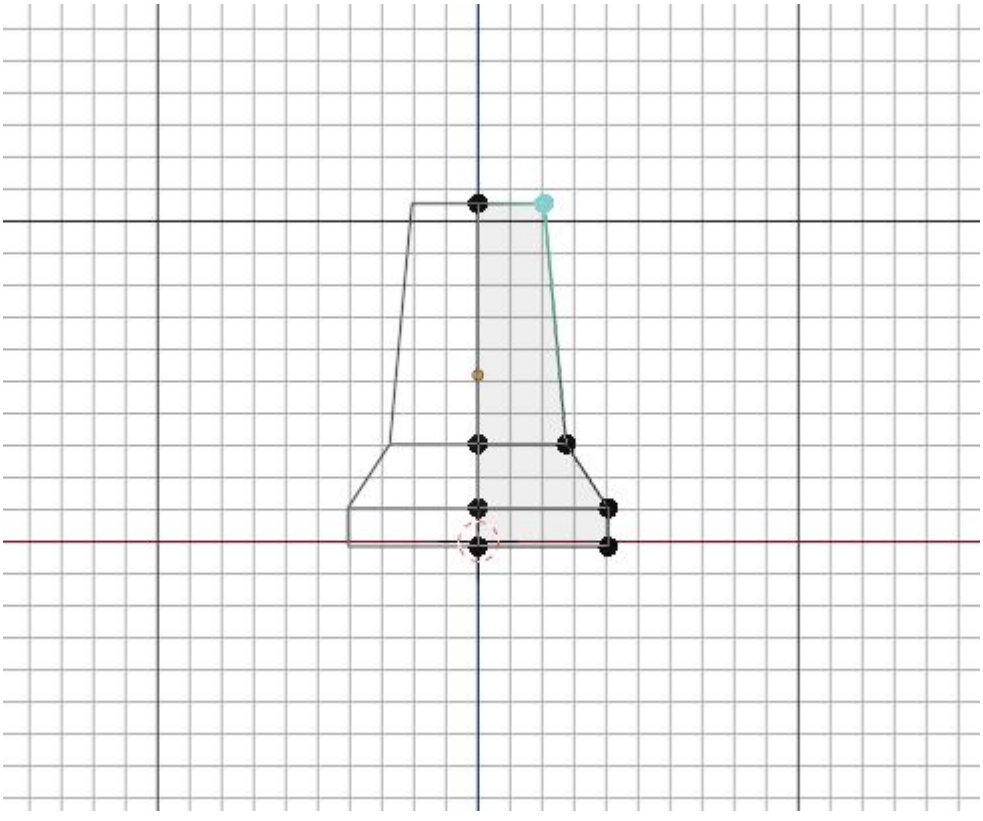
- TAB back to OBJECT MODE and with the current OBJECT selected, locate the WRENCH icon on the right panel on the screen.
- From the Add Modifier dropdown menu, select MIRROR Modifier. You should see the section we deleted above come back into view since the MIRROR is using the X axis to mirror of the original object by default. (See the Check Box that is already checked)



- TAB back to EDIT MODE. Note that you should now only see Vertex dots on the right side of the object, but you will see the full shape with the mirrored side visible. It's mirrored now, and whatever you do on the right side gets mirrored to the left side.



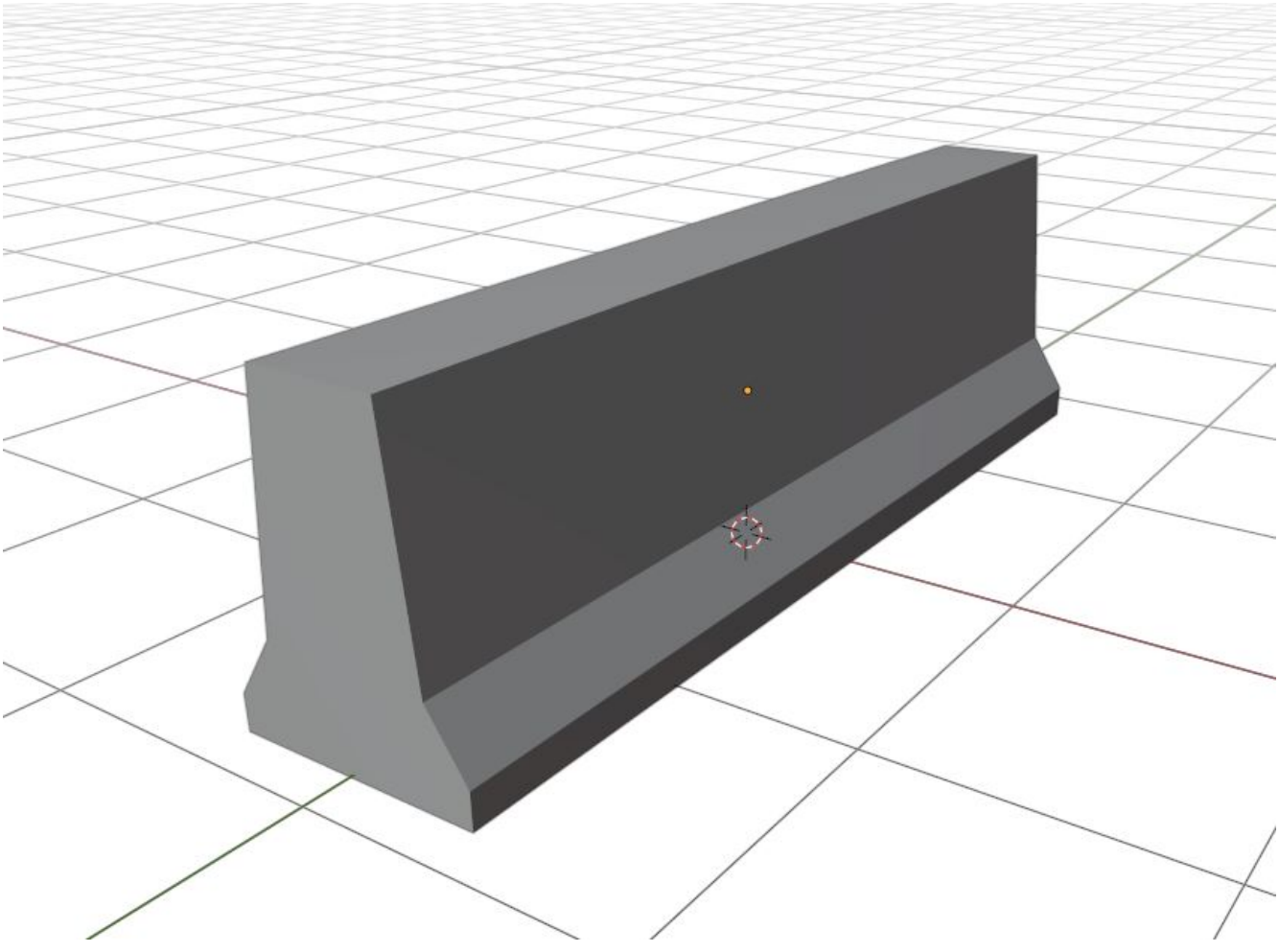
- Select the top 2 right side vertices and press G X to shift them inward until you get about a 55 degree angle.
- Repeat the same process with just the top right vertices until you get about an 85 degree angle.



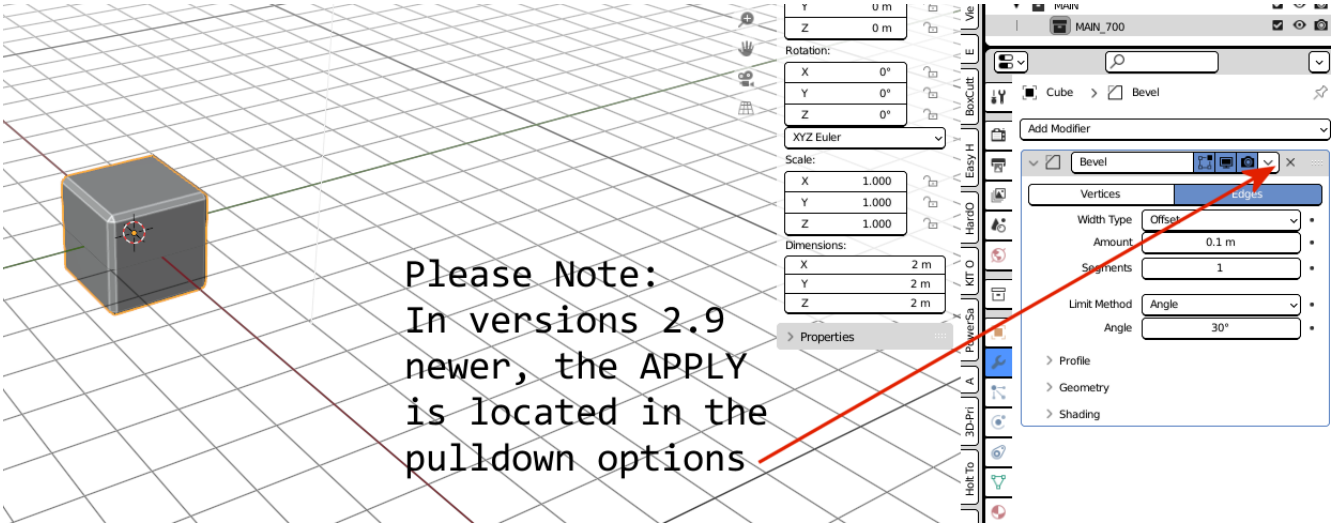
So now we have a basic shape of the concrete barrier. The next steps will complete the shape.

We can go back to `ViewPort SOLID` mode now.

If we look at the barrier closely, we will see that the edges are not sharp. They are beveled. So now we will use the Bevel tool. For this next operation, we no longer need the Mirror Modifier so we can Apply it.



The Apply button for modifiers was MOVED into the pulldown options in Blender version 2.9 and newer.



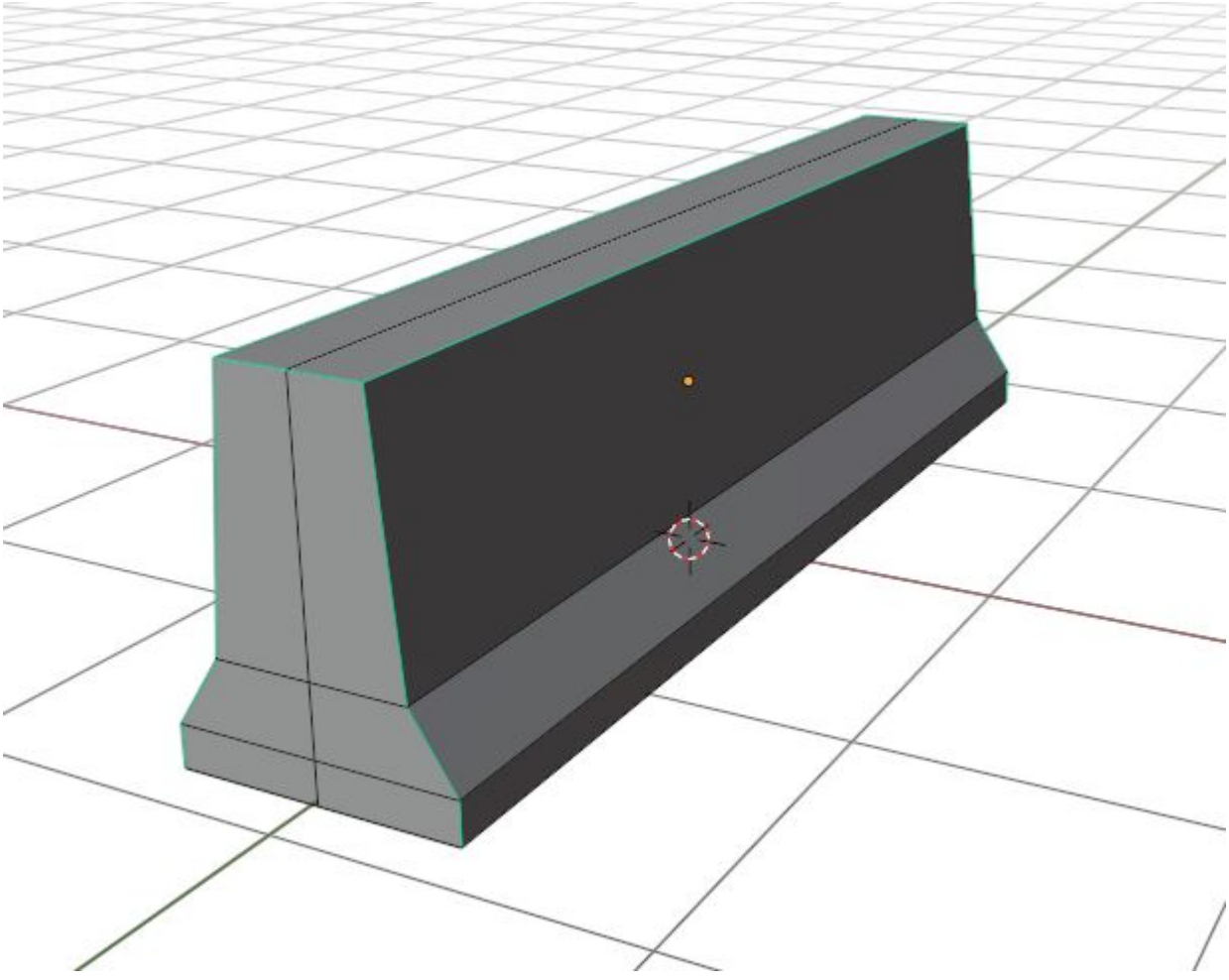
- Go back to OBJECT mode, select the WRENCH Icon and with our object selected, click APPLY. The modifier will apply and go away.

When you apply a modifier, you lose the ability to adjust it further. Prior to applying it, you can still make adjustments to the shape. In our case, were were done with making a symmetrical object, so it was OK to apply the modifier.

- Now, TAB back to EDIT MODE and select the EDGE select mode with 2 key.
- We will select the visible edges of the shape. You will need to SELECT multiple EDGES so here is what we will do. Hold SHIFT then select the TOP LEFT Edge, you will need to shift your view with the MMB to get a good viewpoint for selection. The TOP EDGE will be selected… Now click the remaining "outside" edges while still holding SHIFT.

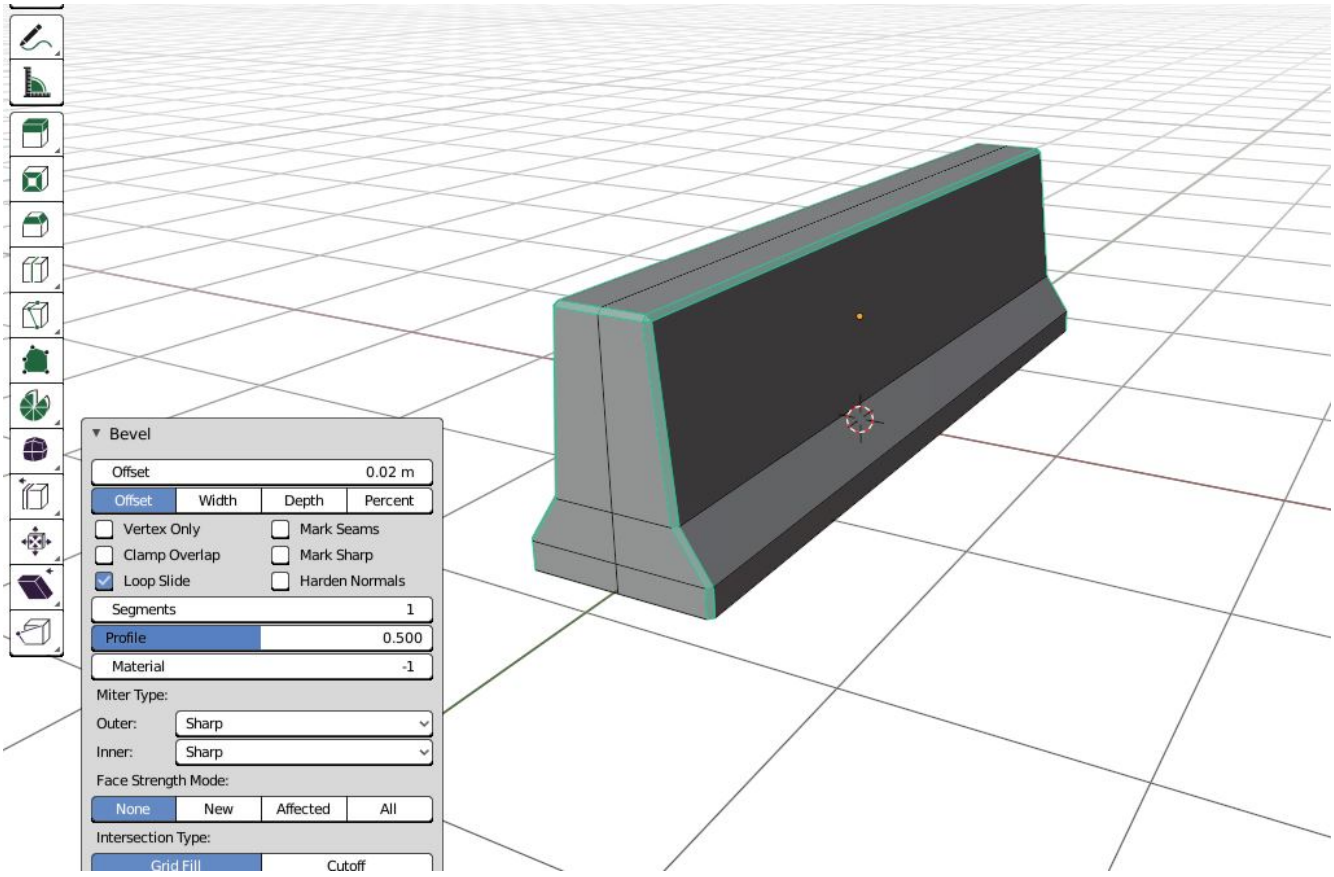
This creates a selection group. If you left click an edge again without holding SHIFT, you will lose the selection group and will need to reselect all the desired edges again.

If you hold ALT + SHIFT you will select all connected edges at once. It often will select more than you actually need, so you might need additional LMB to unselect unwanted edges.

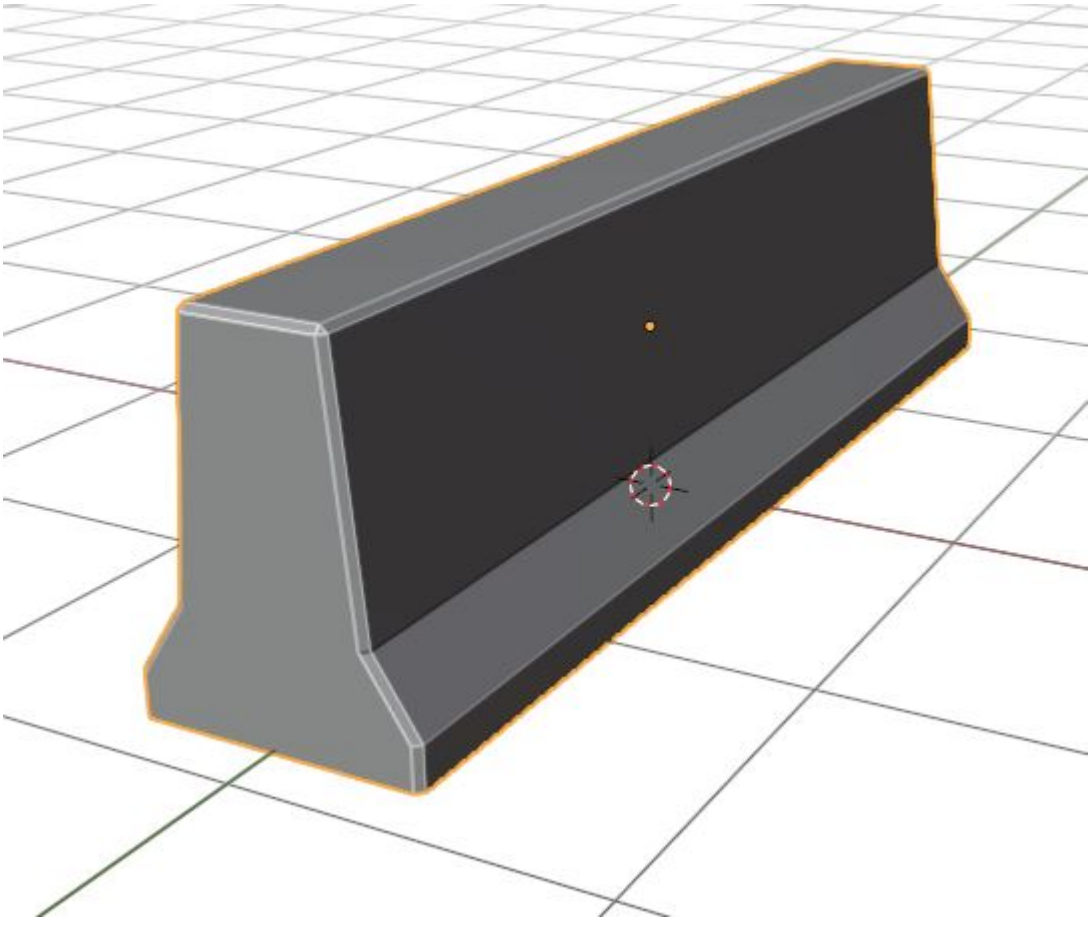


- Press CTRL + B to use the BEVEL TOOL. and adjust the offset to be about 0.02 and Left Click the mouse to accept.

I am aware that I could have left the modifier on during the BEVEL operation. I did not this time because it is good practice to rotate around a model and select specific edges manually.



Here is what we have now after the bevel operation.



If we look at the object back in OBJECT MODE with the Solid Viewport Shader, we see this.

Texturing

The goal with texturing is to be able to apply a 2 dimensional bitmap to a 3 Dimensional object. Its rather tricky and there are multiple ways to do it. The easiest is to just `Mark Seams` and then UNWRAP the object, then moving the resulting `UV Islands` into position on your bitmap.

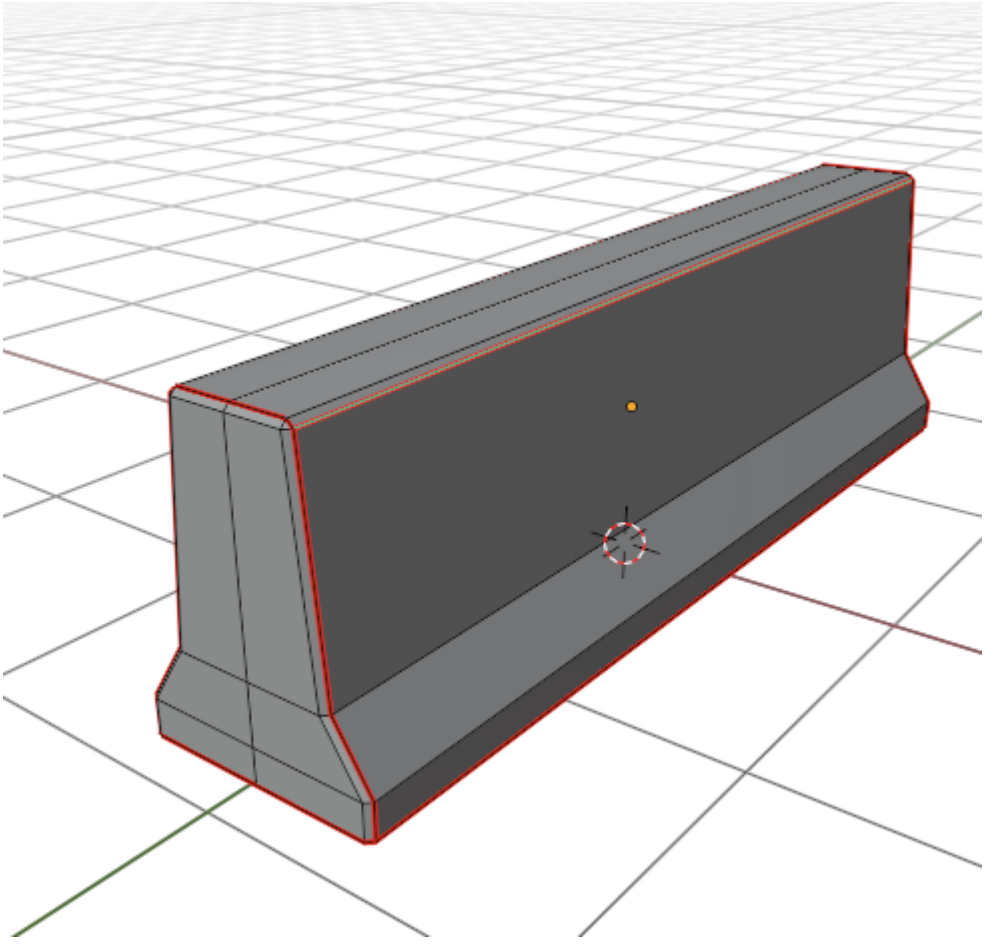
While I say it is the easiest way here, that is a bit misleading. Sometimes, using the `Mark Seams` and `Unwrap` steps create more work for texture creation than is reasonable. Especially if the object is complex but non-organic, like a Boxcar, Engine or Building.

For the texturing steps, we will use a 512x512 texture that looks like this:

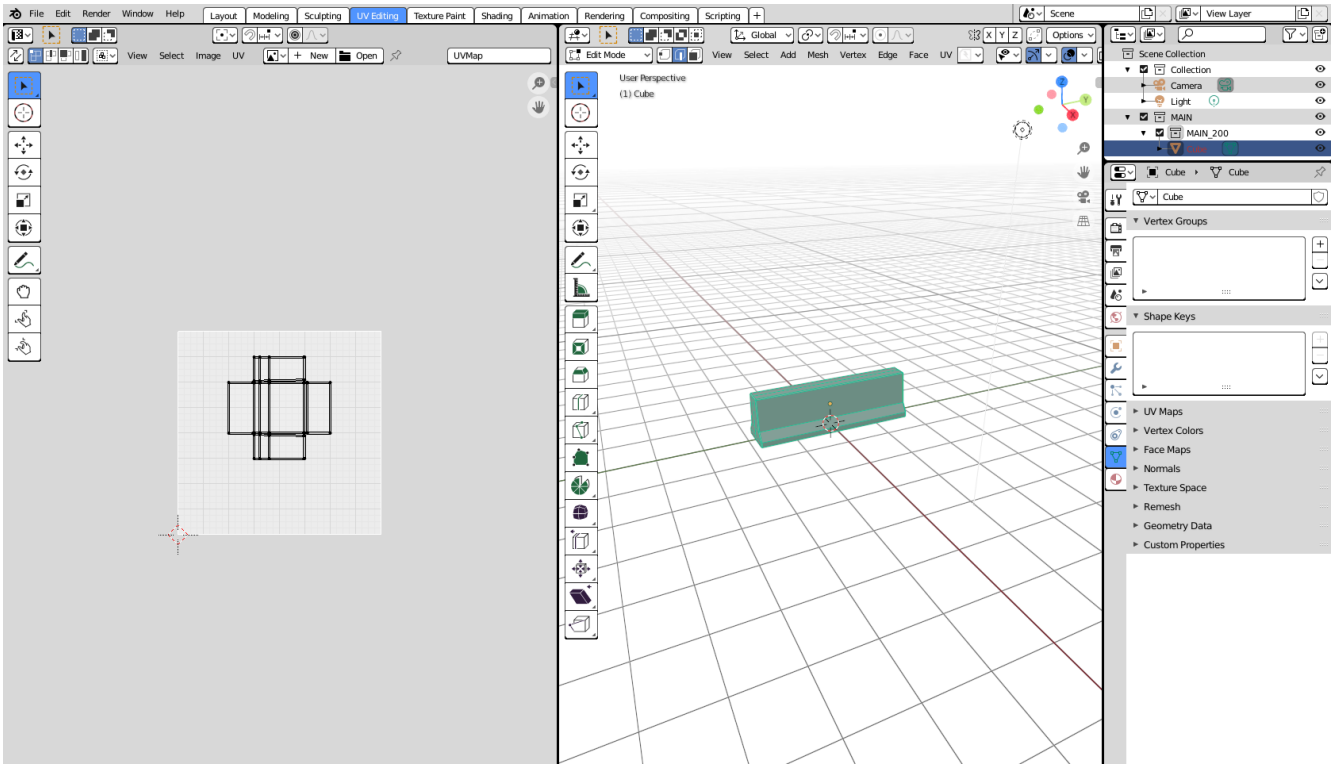
Let's Begin Texturing

`Marking Seams`

You would Mark Seams (Edges) where the flat edges stop and in the case of our Jersey Barrier model, we mark TOP, BOTTOM, FRONT, BACK and both SIDES by selecting all relevant edges and then use `EDGE→MARK SEAM` to define seams. These will now highlight in the SEAM COLOR.



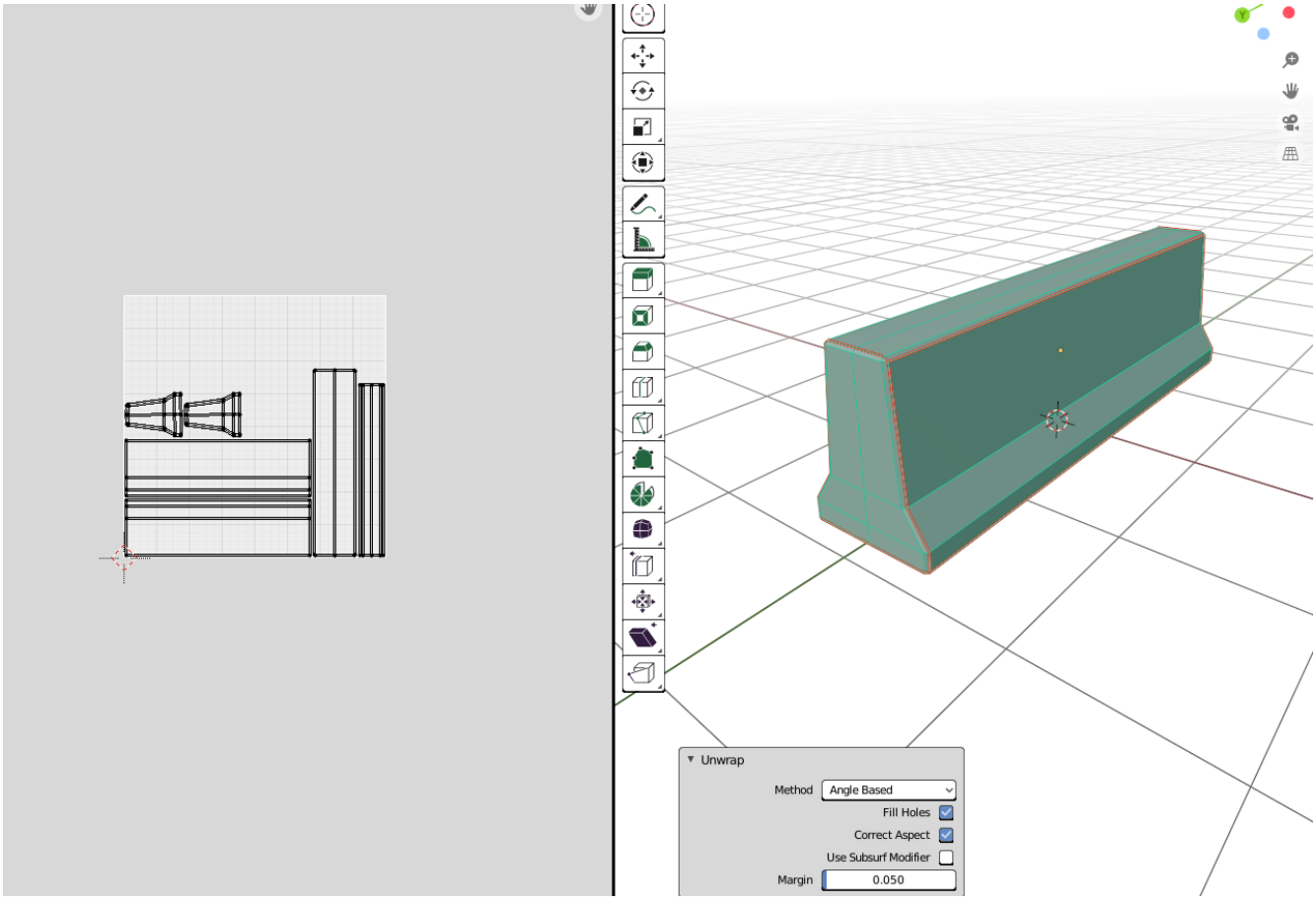
- Make sure you are in `EDIT MODE` and select ALL parts of the object by pressing `A` and then change your `WORKSPACE` tab to `UV EDITING`. (Top of the screen)
- You will now have your `EDIT` window and the `UV EDIT` window on your screen. Also, you might see the UV UNWRAP of the default cube… which is not what we want.



- In the EDIT MODE window select the UV tab , or press U and then chose UNWRAP.

If nothing happens in the left UV EDIT window, you probably didn't have everything selected.

- Now, before you do anything else, locate the UNWRAP tab that showed up at the bottom of the screen and adjust the Margin to be a value of 0.05 or 0.08. The default value 0.001 is just too small for our needs. This will give a greater separation between te generated UV Islands. we will want to move them around.



- Give your model a new material. In the PROPERTIES window on the right side, locate the sphere with a grid inside it near the bottom of the Properties window. That is the material panel.

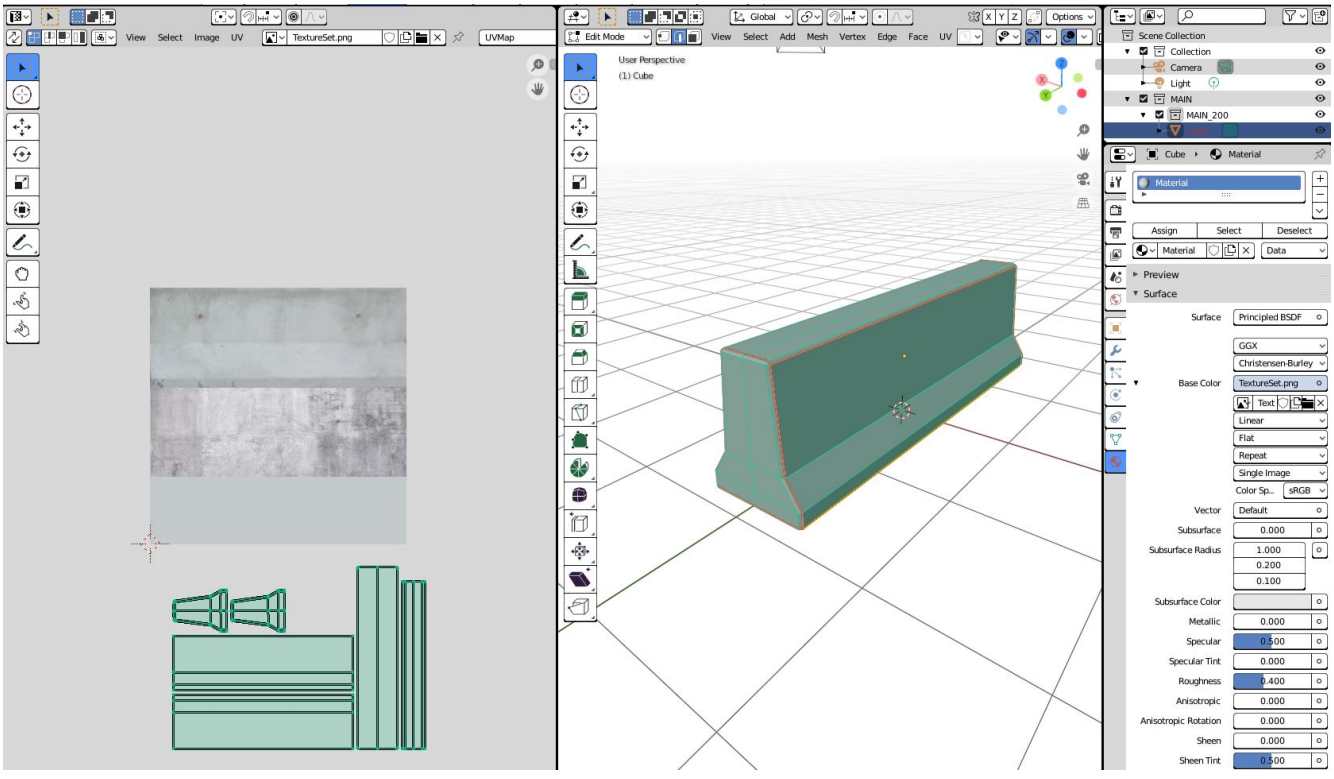
To create a material in Blender, you can use the Material tab in the Properties panel. In the Material tab, you can specify the material's color and texture, as well as adjust its properties such as specular intensity, roughness, and transparency. You can also use the Shader editor to create more complex materials using nodes.

Once you have created a material, you can apply it to an object in your scene by selecting the object and then clicking the "Assign" button in the Material tab. The material will be applied to the surface of the object, and will be used to determine how the object appears when it is rendered.

Materials are an important part of creating realistic and visually appealing 3D scenes in Blender, as they allow you to control the appearance of objects and surfaces in your scene

- Your object likely received a default empty material. Let's update it. Under SURFACE PROPERTIES, Click the small circle on the left side of the BASE COLOR field. You will get a list of options.
- Choose Image Texture and we will thhen locate the Concrete texture we will use. (You can create your own or use the one I created for this) You should have some texture file ready in advance.
- Under BASE COLOR you now see +NEW and OPEN icons. Click OPEN and chose your existing texture file. The UV SQUARE background image in the UV Window should now display your image behind your UV ISLANDS. If you choose NEW, you will have a blank image assigned to the material, which in this case is not helpful.

The GITHUB page has a DEMO1.ZIP file that contains the concrete texture I used.

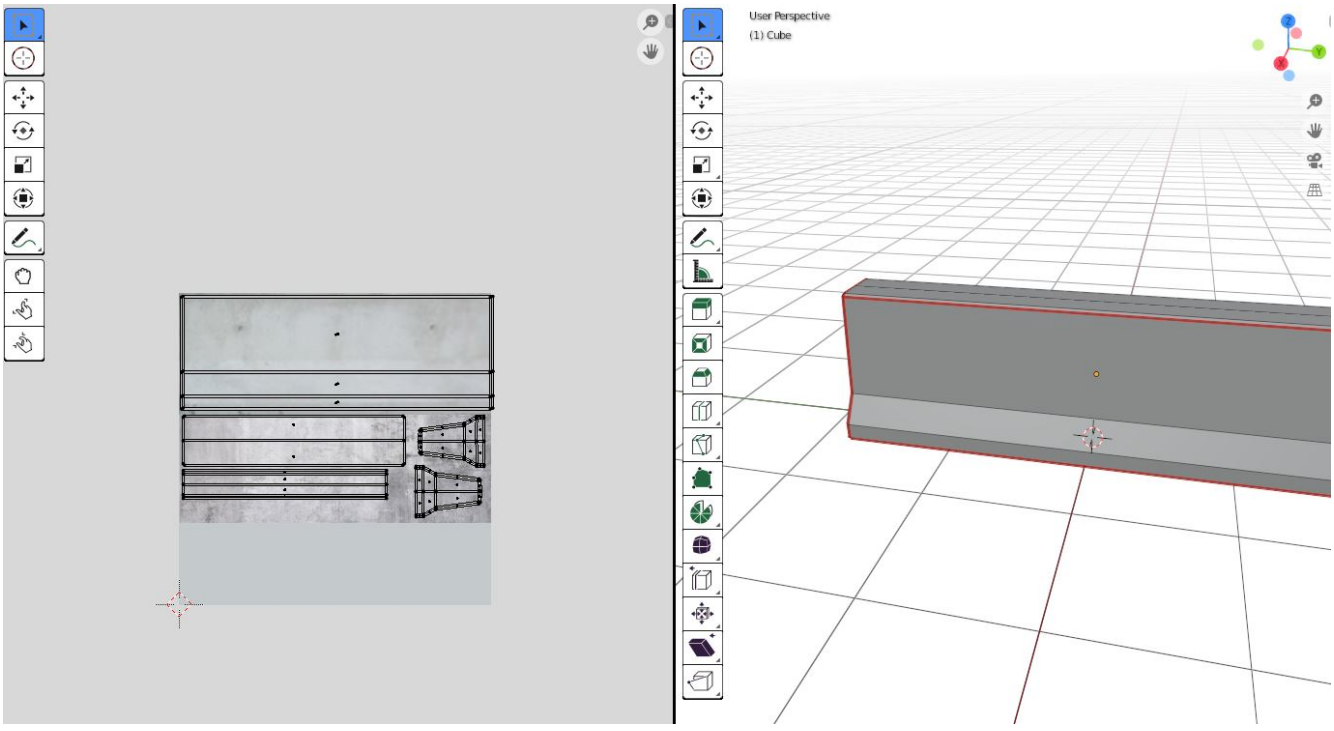


- The task now is to arrange the UV ISLANDS (Using the standard tools G X, G Y keys as well as resizing with S and rotate with R) You can temporarily shift islands outside the 1x1 UV SQUARE, but by the end of this process, all of the islands will need to be back within the 1x1 texture space.

When editing UV Coordinates, the X axis is left-right and the Y axis is up-down.

- Because we added some space between islands, they should be much easier to grab, rotate and place on the background. I recommend moving ALL UV islands outside the space and moving them back in 1 at a time. If you enable the Double Arrow icon in the UV WORKSPACE, updates will be reflected in both workspaces so you can see what you are doing in real-time.
- You can select faces in the EDIT window to isolate them in the UV window or use the L key to select linked faces.
- In this specific case, we will overlap the Left and the Right Side on the same texture space. When appearance is not critical, this works out just fine. Unlike 3DC, there is no "paint" with defined UV coordinates option.

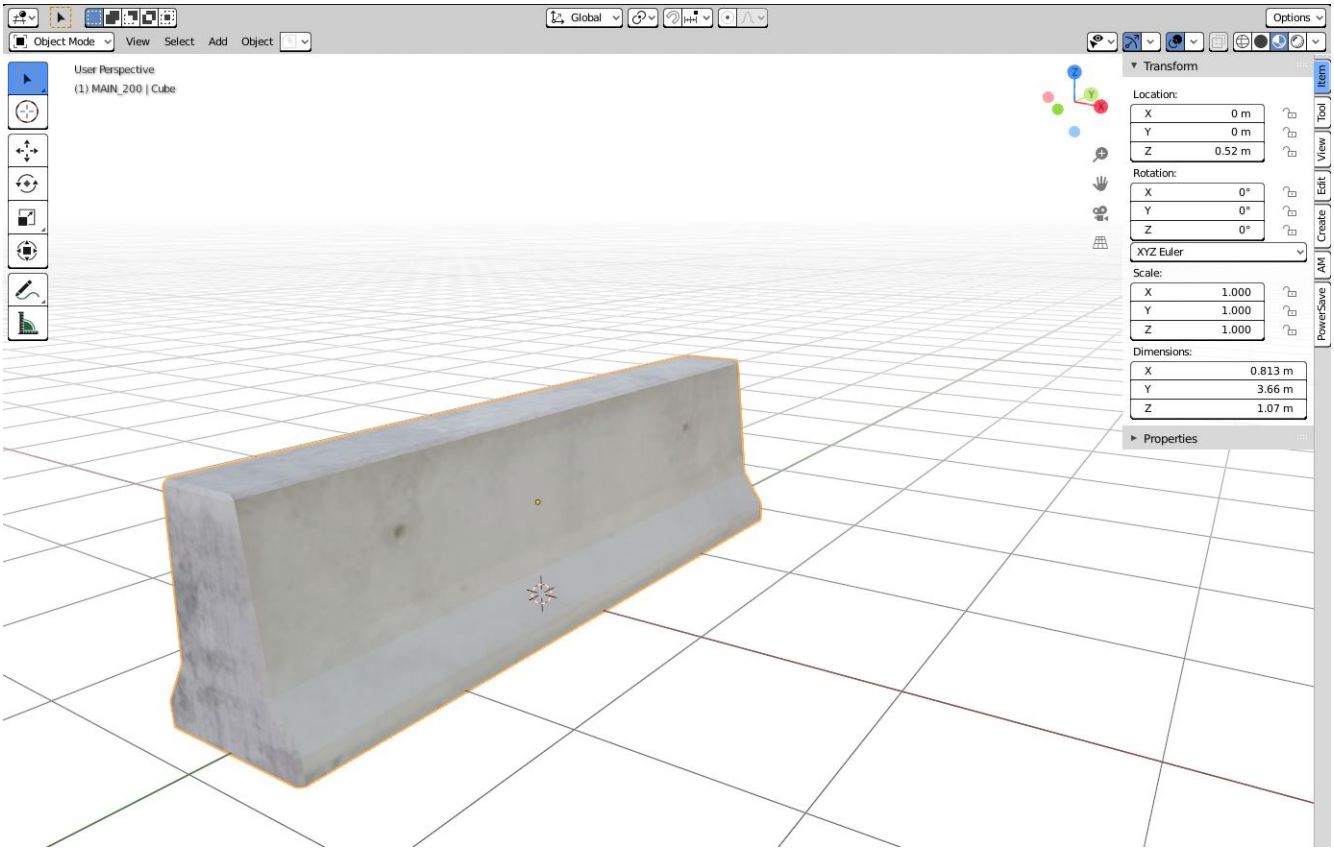
What I ended up with is shown below. Yours will likely be different based on what you marked as seams and how you laid out the islands. Again, its not critical with this type of model.



- Now we can return to the main modeling workspace and then change the viewport shader to Material Preview mode.

MATERIAL PREVIEW 

We now see the object with the material applied.



This model is now ready for the MSTS exporter.

The MSTS Materials shader properties provided by the exporter follow the standard material options available in MSTS:

- Solid – A material that is using an opaque texture only.
- Trans – A material that is using a transparent texture only.
- Alpha – A material that is using a semi-transparent texture only.
- Specularity – A lighting highlight effect that gives the illusion of shine.
- Gloss – A material that is using a glossmap, which is an artificially created reflection.
- Cruciform – A tree shape that requires it's own material so that it can prioritize with the terrain and the surrounding objects.
- FullBright – A material usually assigned to the inside of trains so they never get dark.
- HalfBright – The same as full bright but at half the intensity.
- Dark Shade – The opposite to full bright, it reduces the overall intensity of light.

In `OBJECT MODE`, under the Materials property panel, scroll down to the bottom to where the MSTS exporter section is, titled: `MSTS Materials`.

1. Update the `BaseColorFilePath` (using the Folder Button) to select the texture of the highlighted object we have been working on.
2. Now switch to the `SHADING` window using the top menu bar. Note: that the `NODES` window now has a `UVMap` node and the text field says "UVMap". This is needed for the MSTS `S` file export.
3. With this verified, you can now use the MSTS exporter.

- Chose **FILE › EXPORT › OPENRAILS/MSTS(s)** file, and choose where to save. Use the browser to select your project's `\final\` folder and modify the filename as needed… I chose `barrier.s` for example. The bottom of the screen will say "Finished OK" when done.

The exporter will ONLY create the `.S` file and by default assume your texture is `.ace`. Converting your `TGA`, `.JPG` or `.PNG` textures to `.ACE` (or .DDS) remains for you do as you will need to do this manually ourside of Blender.

For a sanity check… edit your exported `.S` file with a unicode editor to make sure your texture reference is correct.

Version 4.4+ of the MSTS EXPORTER includes a button to define the textures used in the `.S` file as `.DDS` instead of `.ACE`

```
images ( 1
       image ( TextureSet.dds )
  )
```

ABOUT DDS SUPORT With a master texture file, in a `PSD` format for example, and using PAINT.NET to keep the layers intact, you cave save it from PAINT.NET as a `.DDS` file natively. As a result, a minimal amount of after-the-fact editing is needed. To work in Blender, save a copy the master `PSD` file as a `PNG` file and to work with in Blender. Then save a copy of the master `PSD` file as a `DDS` file.

Again, the Exporter script will **only** create the model `.S` file. You still need to create all the final textures and the ENG or WAG or similar related files that define your model to the simulator.

The final step would be to copy the contents of the project's `FINAL` folder into the Open Rails `trains` content folder for testing.

Creating Level Of Detail Distance levels

Once you have your main model created, you should consider creating Level of Detail or "LOD" entries. These are easy to add or change by editing LOD collection name, eg create MAIN_0500 or rename MAIN_0700 to MAIN_0500. You add content to a new LOD level collections by dragging parts around or using the M key to assign collections. With collections, all your LOD assignments show in the outline panel and you can hide or un-hide whole collections as needed since your LOD assignments are also clearly shown in the object's 'Collection' panel. Making use of the filter options and checkboxes of the Outliner helps to focus your work.

Blender WILL rename your parts when you add them to a new LOD, but the MSTS exporter is looking at the COLLECTION and BASENAMES of the objects, like WHEELS, which means you won't need to rename entries that suddenly have 001 appended to their name.

You can select a group of objects in the Outliner by using the `SHIFT-LMB` to copy multiple items at once to a new COLLECTION. It would be good to experiment with the Outliner filter options so you are aware of what is available.

Guidelines for creating LOD versions

Level of Detail of versions of a base model are determined by how much detail is visible based on the distance the camera is from the camera (or player view). When your camera is very close to an object in Open Rails, then you want to see the maximum detail available. As your view of an object becomes far away, there is no reason to display the full detail of the object as much of it will be hidden or to small to see. The reduced polygon models (LODS) will increase graphic performace by reducing the amount of drawing that needs to be performed by the graphics card. For example, your main model might have 2000 polygons, but at 750 meters, you might only need to display 200 polygons and at 3000 meters, you

might only need to display 50 or so polygons to get the same overall effect.

You define your intentions for each level of detail in your outliner. As explained earlier, you define your LODS using a naming convention under the MAIN collection where each lod distance setting is part of the sub-collection name, for example:

NAME	DEFINITION
MAIN	MAIN COLLECTION
MAIN_0500	0 - 500 meters
MAIN_1000	500 - 1000 meters
MAIN_2000	1000 - 2000 meters

```
MAIN
    MAIN_0500
         BODY (20,000 polys)
         BOGIE1
            WHEELS11
            WHEELS12
         BOGIE2
            WHEELS21
            WHEELS22
         etc

    MAIN_1000
      BODY (3000 polys)

    MAIN_2000
      BODY(200 polys)
```

You can use the DECIMATE operation in Blender to quickly reduce the polycount of a shape while maintaining most of the topology of the original shape. This can give you a head start on reducing what gets drawn on each LOD level in additional to just removing the finer details that won't be seen.

Building a Library of Reusable Parts

This section mainly applies to users who still use a version of Blender prior to version 3.0.

Blender 3.0+ has a new Asset Library feature where you can store pre-made objects and textures that are shared among your projects. A perfect place to store various pre-made Bogies, Couplers and other common objects. In fact, this is one of the best reasons to consider upgrading to 3.0.

It is common to see regular parts used multiple times in a single design. Handgrips, railings, wheels, trucks etc are often regularly available parts purchased by Rail Vehicle companies as commodity items. In the same way, we should not have to keep making the same parts over and over again.

One way to accomplish the re-use of the common parts is to **duplicate** one with `SHIFT` + `D`. A selected part will be duplicated and ready for a location transform to move it to a new location. This is easier to manage after you have completed the UV-Unwrapping of the object. All duplicated parts will share the same UV MAP locations so if you need them to have different mapping, its a matter of shifting them around in the UV Editor.

You should not use `CTRL` + `C` and `CTRL` + `V` to duplicate objects as an alternative to `SHIFT` + `D` because the standard copy/paste operation will make new copies of everything, including Materials. So your MAIN texture would become MAIN001, MAIN002 etc on each copy/paste operation.

A particularly nice add-on for Blender, if you are using a version older than 3.0, is "Asset Management". Yes, I know, it costs $40, but I don't regret it. I don't even use all of its features. It does let me export objects into an asset library. It creates a thumb-nail of object to assist with locating the item in the future from a grid of object pictures. When you select an object from Asset Management, it will be inserted into your project.

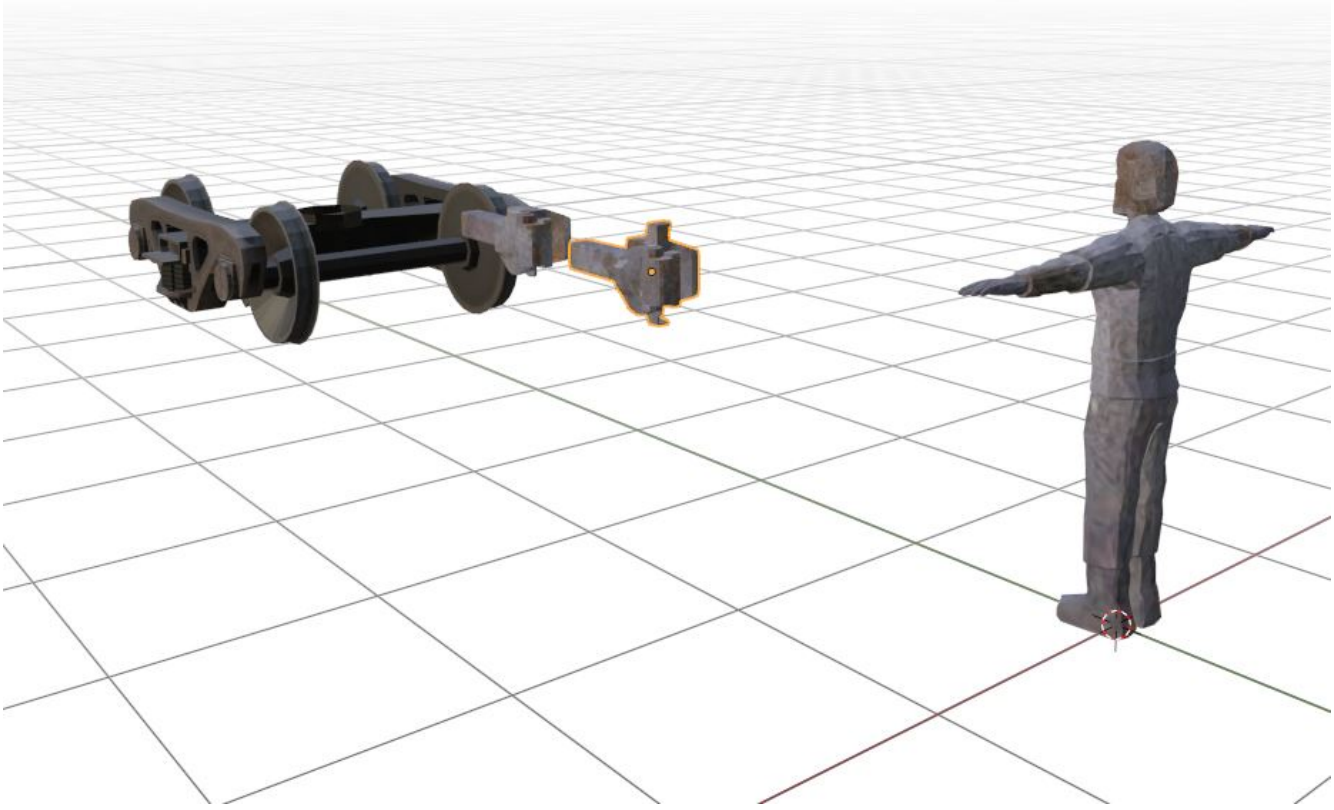
https://Blendermarket-production.herokuapp.com/products/asset-management

This add-on has been made somewhat redundant now that Blender 3.0+ comes with an Asset Browser but it does still work in version 3.0.

A Brute-Force Library solution

A simple and rather easy way of creating a collection of re-usable parts is to **COPY** and **PASTE** specific parts, adding them from your current file to a library specific `.blend` file. It does require that you have 2 copies of Blender running.

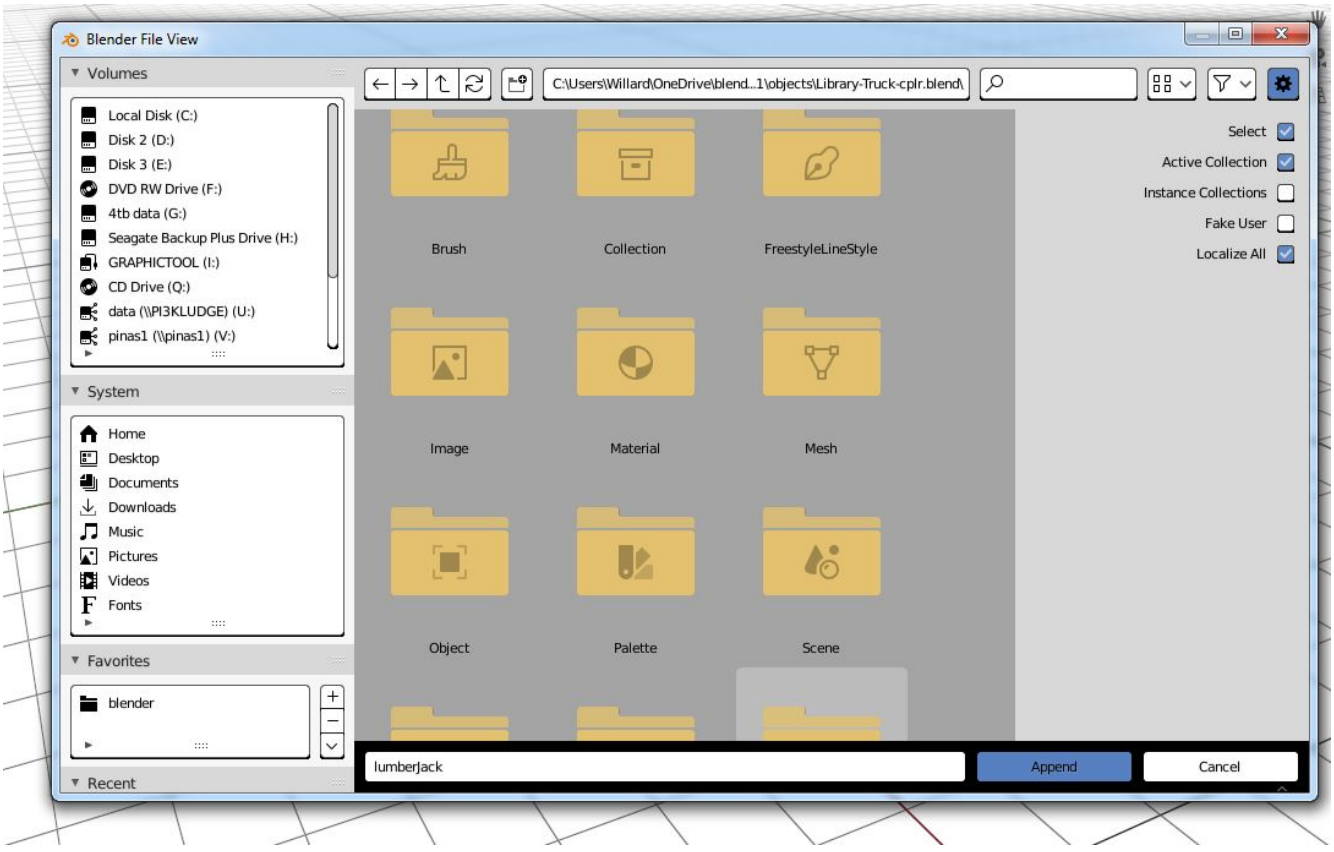
- Save your current project, just in case.
- Highlight the object you want to export to a library file, but first make sure you set the origin to object geometry. **View › OBJECT MODE › OBJECT › SET ORIGIN › ORIGIN to GEOMETRY**
- To copy, press `CTRL` + `C`. The object will be copied to a system buffer.
- Open a *fresh instance of Blender* so you have 2 blender sessions running, clear out all unwanted default items from your new scene and press `CTRL` + `V`
- The copied object(s) from your first Blender session will be placed in your new blend file. Copy as many objects into this file as you need. It is best to avoid overlapping them.



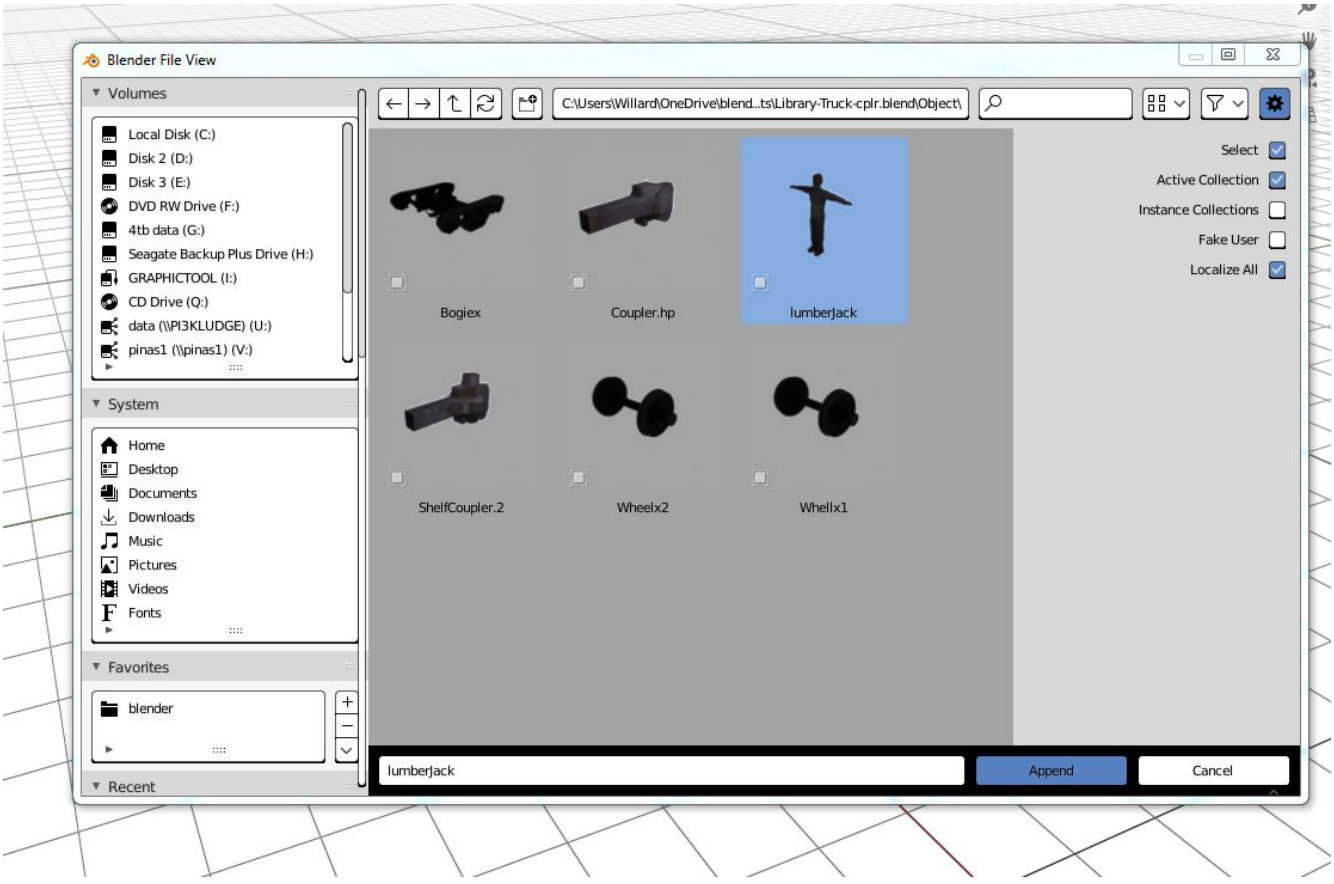
- Save it now with a name that refers to the new object library you are making. Example: `Library_Freight.blend` (Don't skip this step)
- Now the tricky bit, open a **NEW GENERAL Blender FILE**
- Choose **FILE › DATA PREVIEWS › BATCH GENERATE PREVIEWS** and chose the file name you used in the prior steps. Example: `Library_Freight.blend`

This will take a few moments and will eventually present you with a console screen when complete. By doing this, we are telling Blender to create Object Icons in the saved file.

- In the future, you just need to use **FILE › APPEND** option to insert the object into your current projects, but, the bonus is that you can select them using the `THUMBNAIL` view option in the `FILE APPEND` menu, so now you will know which part you are appending to your current `blend` file in a visual way.
- Choose append ans select `THUMBNAIL` view you will see file menu.
- Choose the library file you saved



- Choose OBJECT folder and you will see ICONS of the parts in the file. Select one and it will be appended to your current file.



When using this technique for an object library or multiple object libraries, you should to consider your file structure because when you build up a repository of `.blend` files it can become confusing when searching for a specific item later on. Using a well planned and organized file structure is better than having files all over the place or all in a single bucket.

It might mean that there are some redundancies in your files in the long run, but consider populating the OBJECTS and TEXTURE folders of each project local to that project versus using a master folder for all projects. This way you can make folder-relative reference to files and you have the ability to move folder as well as share with others without breaking Texture file references, for example.

Another somewhat flexible option is to export your selected parts as an FBX or Collada (DAE) files. These can be imported later with a `File-Import` process but it also means that you have something that can be imported to other applications as well. The main difference with this option is that these are no longer native `.blend` files and there could be some mangling of contents when being imported back into Blender.

For the curious,

A file structure that I often use for individual project folders is shown below:

```
<project>
<project>\final
<project>\mesh
<project>\objects
<project>\textures
<project>\tmp
<project>\reference
```

I would store any `local` object library collections created with the COPY/PASTE method under `\objects` folder. There is a handy tool available https://www.dcmembers.com/skwire/download/text-2-folders/ "Text2folders", that makes creating this consistent folder structure easy. Just replace the text "<projects>" with the name of your current project and pass this text file to the "Text2folders" application and it will create all the folders for you. It can even do sub-folders.

Keeping the Part Library and Texture files available under the current project structure is helpful if you share your modeling content with others, provided you have the available disk space to allow duplicates in multiple project folders. It also allows for small tweaks that are specific to a project.

If using the `Asset Browser` feature of Blender 3.0, you would use the **3D View › Object › Asset › Mark as Asset**. Then, `Save the file` in your ASSET folder, otherwise your asset edits won't be available to other files. This is why it might be better to create specific library collection `.blend` files to be stored in the asset folders.

Importing Existing Objects

Importing shapes from other 3D Software can be a bit tricky and there are numerous reasons why. I will try to outline a few basic steps and possible pitfalls that can be avoided.

- Imported shapes often get renamed with a `S_` prefix when using Blender importers which might require you to to rename your objects in the Outliner.
- When your source objects are in a Autodesk `3DS` file format, which is pretty common, you need to cope with the fact that, for some reason, Blender 2.8+ dropped support for it. 3rd-party software like 3D CANVAS or some online converters can do the conversion for you.

You can also run multiple versions of Blender side by side. Blender version 2.79 still supports importing and exporting 3DS files and it is a simple as importing a .3DS file into Blender 2.79, perform a Select-All and COPY and then PASTE into your current Blender 2.8 session.

As of version 3.6 LTS, Blender once again supports importing of 3DS files.

- UVMapping can survive an import operation… but material and texture selection probably didn't.
- The import step has also probably has assigned a new material for each object and to be honest, it's easier to manage if you have all objects that use the same texture also use the same material (unless you need special alpha or reflective properties for something like metal or glass).
- You also will probably need to manually tell the S file exporter (under MSTS Materials) where to find the texture file under the BaseColorFilePath input field.
- Lastly, from personal experience, if you are importing a DAE (Collada format) file from say 3DC, you might encounter an issue with UVMAP assignment in SHADER EDITOR where you are not getting the right UV map assigned and you see no texture in the Material Preview mode (Commonly called LOOKDEV mode).

This the second tab from the right in the viewport selections. Everything will LOOK right in the Material Preview, but exports will fail with a "Missing UVMAP in:" error when you try to use the S file exporter. *See the next section for how to deal with this problem.*

Material Preview 

Fixing UV MAP Assignment Issues on Imported Objects so they work with the MSTS exporter

Each mesh that you export must have a uv map named UVMap.

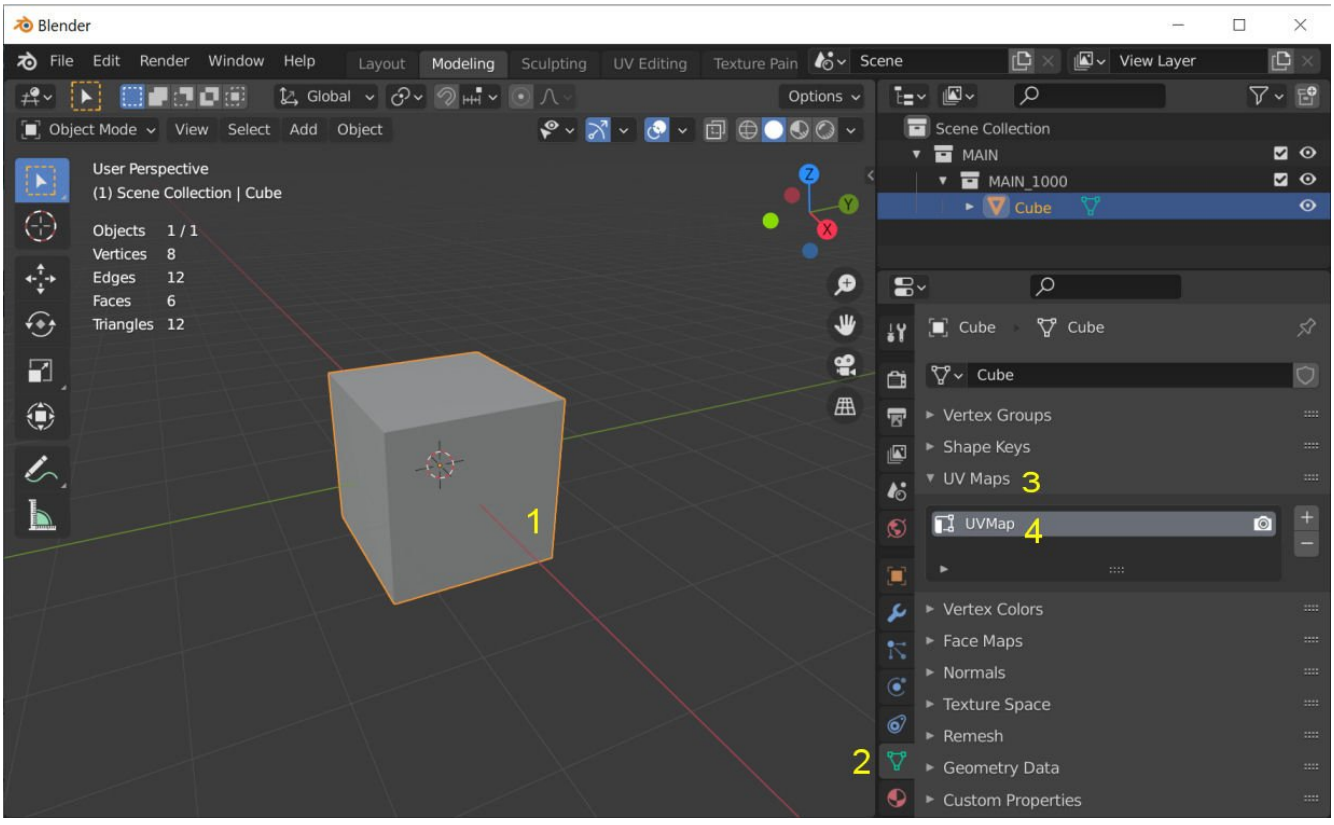
Since UVMap is the **default** name assigned by Blender for new UV MApping data, if you create everything from scratch, you won't encounter the problem when you export.

The mapping reference of UVMap is the only map that will get exported to an S file by the MSTS exporter. When using the DAE importer, for example, you might end up with a UVMap that is referenced using an object name from the time of importing.

There are some other cases where this could be an issue, for example:

- An advanced user may have multiple uv maps with other names assigned. (eg layered shaders, baking, etc)
- When importing a mesh from some other program it may come with different uv map names
- Some non-english versions of Blender may use a different default name.

You could check or try modify the name of a uv map like this:



1. `LMB` Click on your mesh object.
2. Select the `MESH PROPERTIES` tab.
3. Open the `UV Maps` panel.
4. Double click on a `map name` to change it to 'UVMap'

There is a tested script that will do this for you.

```
import bpy

for mesh in bpy.data.meshes :           # for every mesh in the .blend file
    if len( mesh.uv_layers ) == 0:      #   if it doesn't have any uv maps
        mesh.uv_layers.new()            #   create one, if it doesn't have one
                                        #   with the default name
    if not mesh.uv_layers.get('UVMap'):
        firstmap = mesh.uv_layers[0]    #     rename the first map
        firstmap.name = 'UVMap'         #     to the default name
```

To use this code:

- You would select the `Scripting` Tab and `LMB` + `CLICK` the NEW option to get a new edit window.
- You would copy and paste this code snippet into the text editor window.
- Last, you would press the "RUN" arrow icon in the file window menu to run it.

It will fix the UVMAP naming on all objects for you.

Model Exercise #2

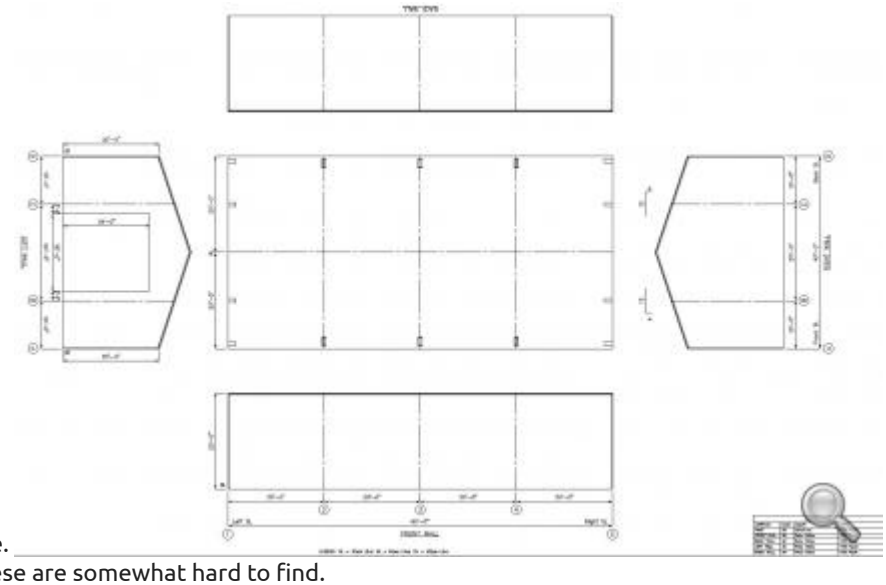
Intermediate Modeling

Key items to take away from exercise #2

- Interiors
- Glass
- Texture preparation
- Back Ground Images

Building the General Shape

Let’s say we are modeling a building. I’m thinking… 2 stall engine house 40' x 80', and 20' tall with two 12’x18' garage doors and standard metal door with an office window.



I will use this for a background image.
Apologies for the terrible image. These are somewhat hard to find.

Importing the Background Images

Much of my previous content was created using Amabilis 3D Canvas modeling software. It was rather difficult to create and use good background images, so I seldom used this method. With Blender, the process is much easier and I highly recommend it.

You should pre-determine the center-point of each view and align them with the axis lines in Blender.

If you can find blueprints of at least the front/back and side view, you are all almost all set to use them as backgrounds.

If your blueprint is black and white, you might consider modifying it to be a negative image with the background as black and the convert the background to clear alpha.

In the "LAYOUT" workspace (in the top menu) in `OBJECT MODE` select SIDE VIEW (KEYPAD 3). Use the ADD → Image → Background menu tree to load in your side view image. Select FRONT VIEW (KEYPAD 1) and then ADD → Image → Background menu tree to load in your front view image.

By default in Blender 2.8x, you will only see the Side or Front background images when in front ortho or right ortho view. The images will go away when you pan around your model. This is a setting in the Data Properties of the Background image object.

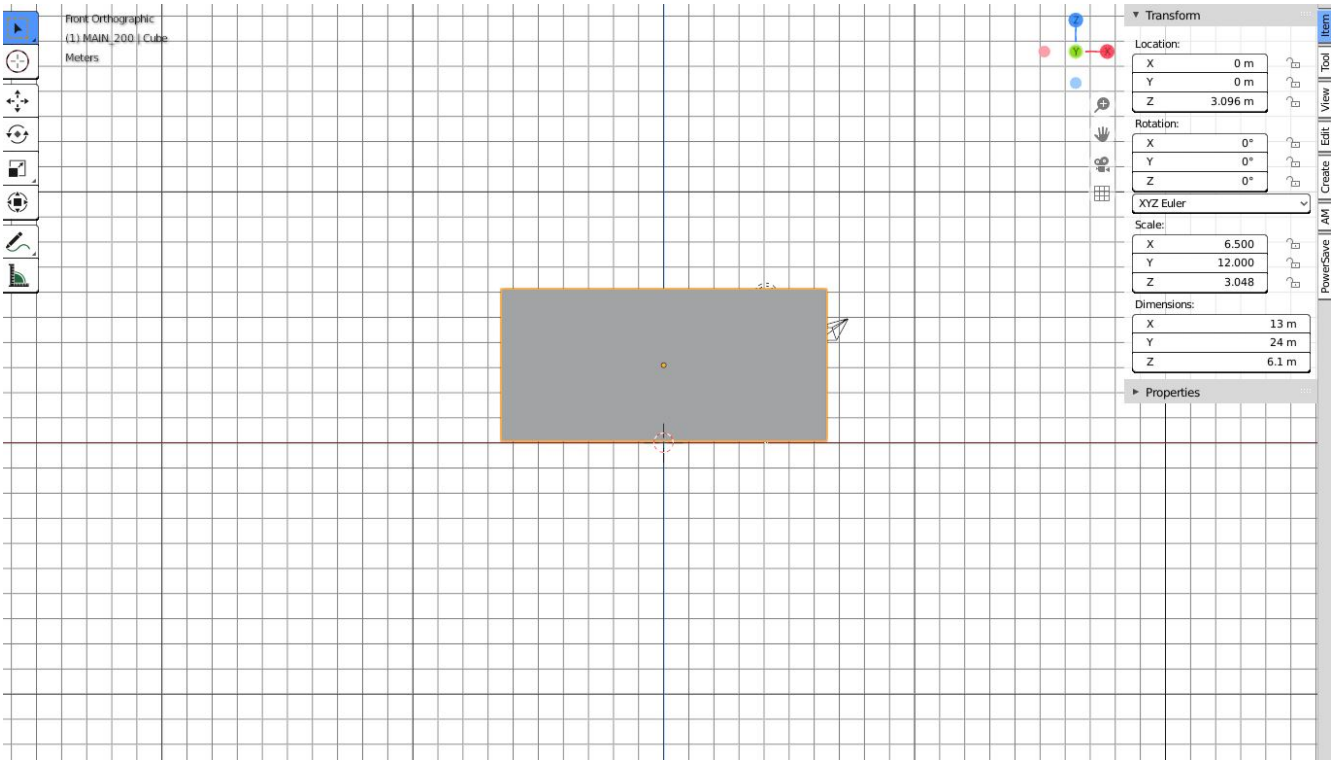
Setup

Use the file structure that was outlined earlier.

```
twostall
twostall\final
twostall\mesh
twostall\objects
twostall\textures
twostall\tmp
twostall\reference
```

- Add the Background image to `reference` folder

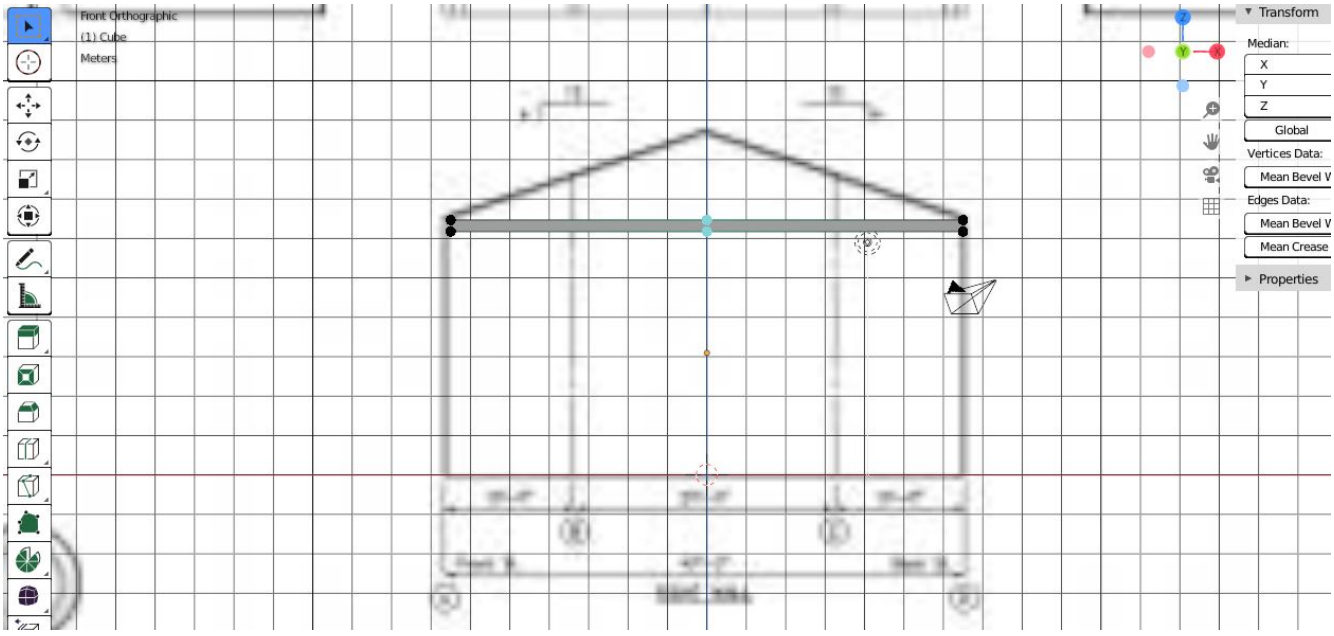
- Adjust the default cube Dimensions to be 40x80x20 (X=12.192, Y=24.384, Z=6.096). This will be the basic shape.
- Apply the scale with OBJECT→APPLY→SCALE



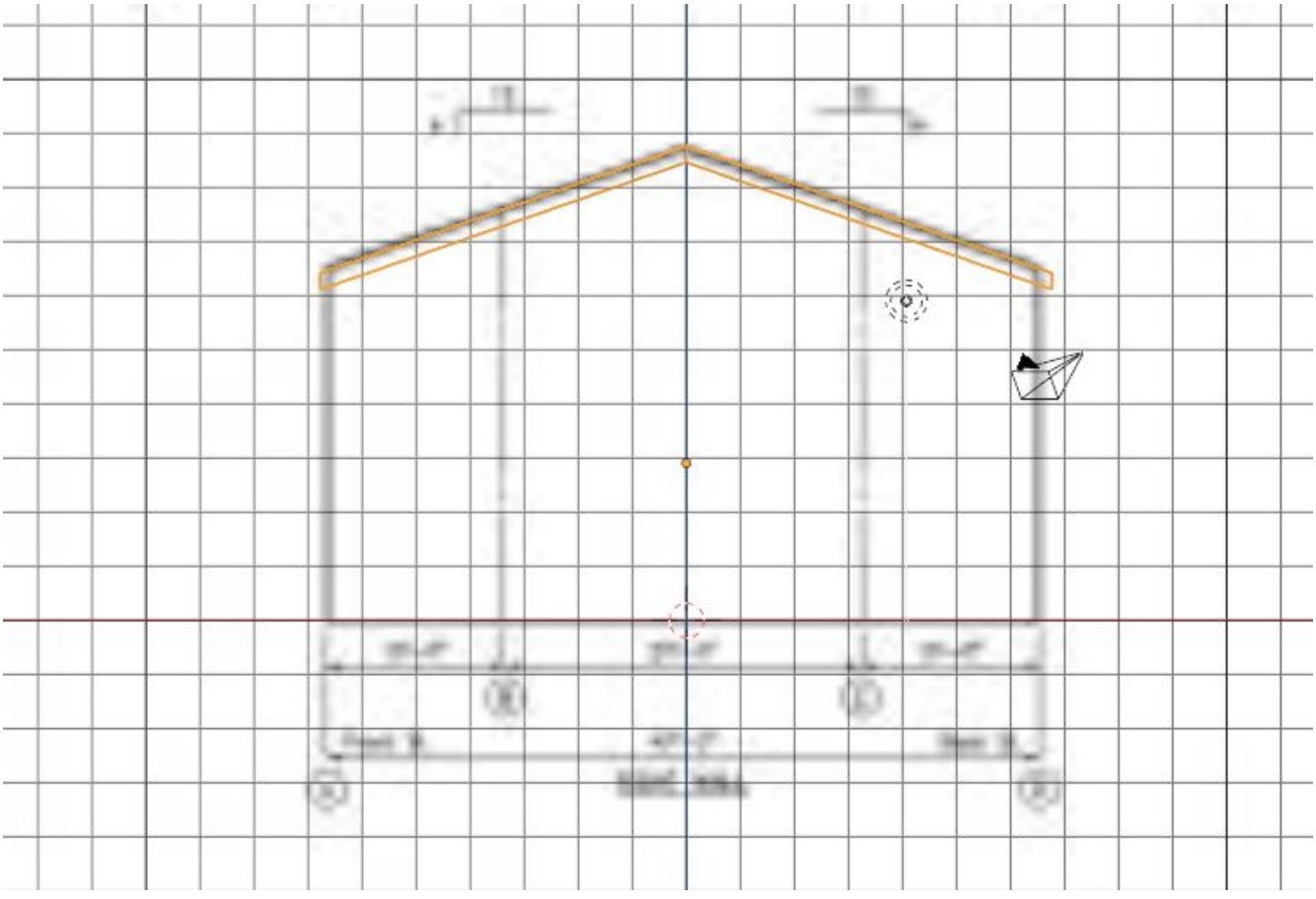
- Save the twostall.blend file under twostall\mesh folder.
- Switch to front view KEYPAD 1 and select ADD→IMAGE→BACKGROUND and chose the image from twostall\reference we saved their earlier.
- Position and scale it to match the basic shape, but not the roof.

Now, it might seem intuitive to just use the basic outline we have now, insert a loop cut on the top and shift the top vertices upward to make the roof shape. Here is why that is the hard way. You won't have a roof overhang and all roof edges have an overhang. We will instead shift the current shape so it is only as tall as the roof overhang.

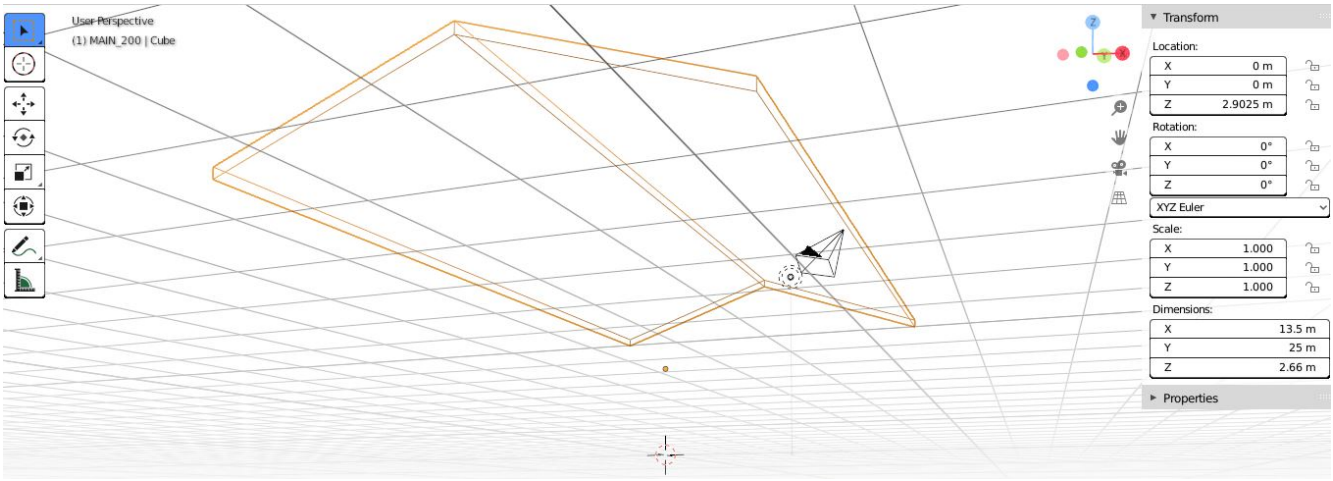
- Press TAB to reach EDIT MODE and 1 for vertex select and Z and select (WIREFRAME).
- Add a vertical Loop Cut CTRL + R in the center.
- BOX select the bottom vertices and shift G Z to be about the right height for the roof overhang.



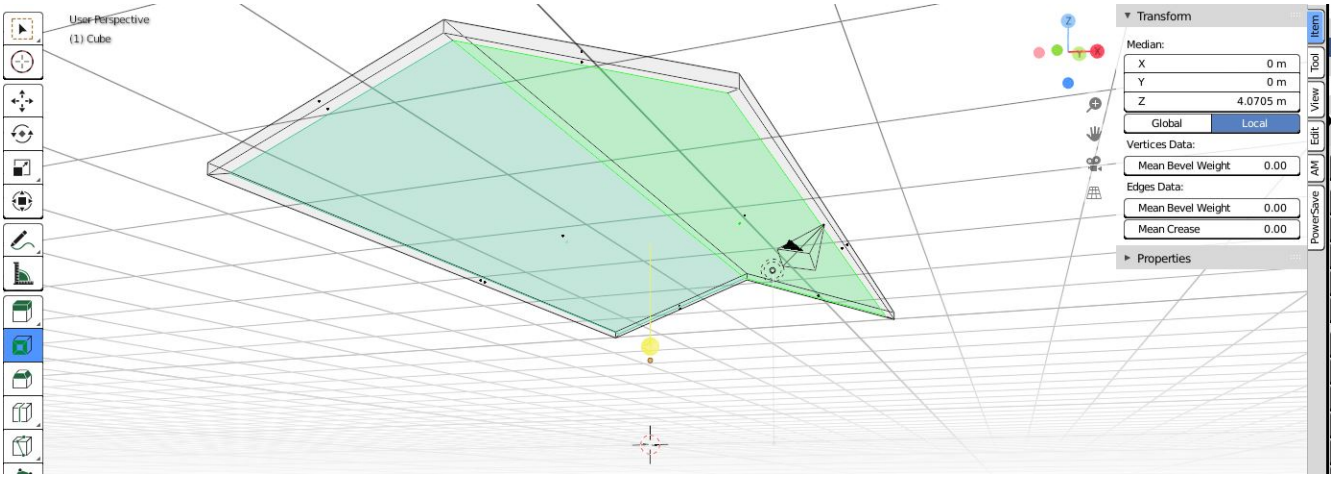
- Use BOX select to chose the central vertices and shift them upward to match the roof angle, G Z
- Switch to Face mode, 3



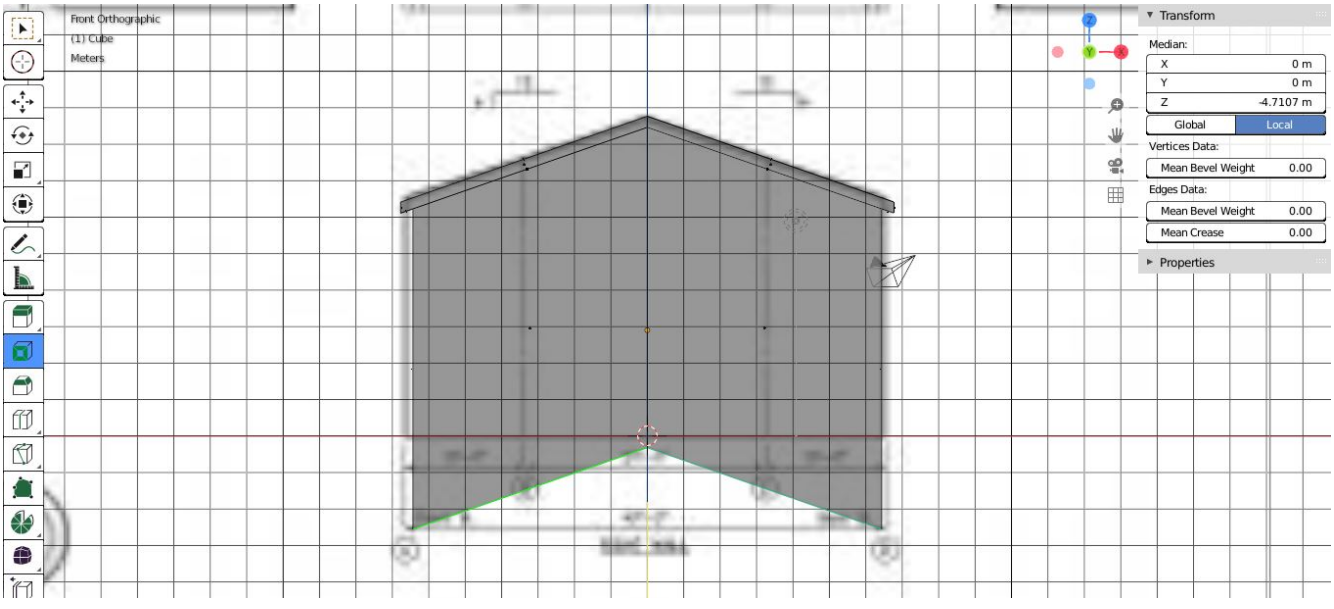
- Shift your view to be looking to the underside of the roof shape.



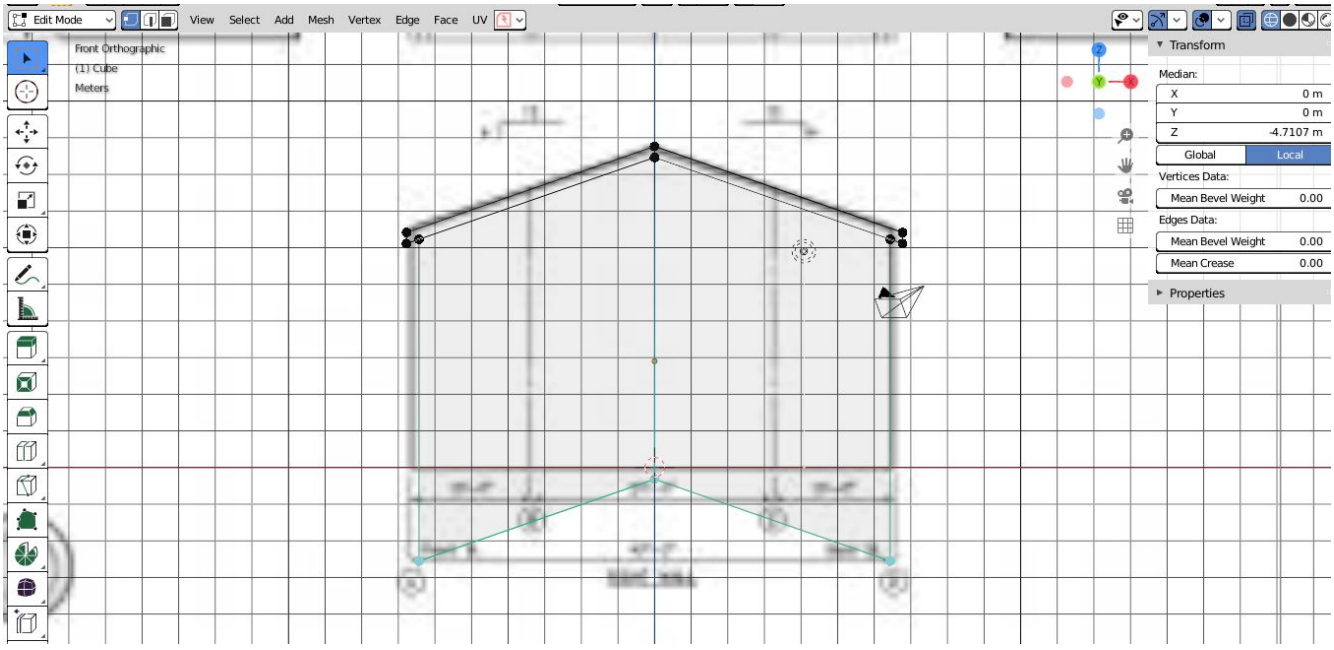
- Select the underside faces and use `I` to inset the selected faces to the amount of overhang that looks right.



- With these inset faces still selected switch to front view `KEYPAD 1` use the Extrude option `E` `Z` and drag the inset faces to ground level using the central vertices as a guide. Note: I used `solid view` mode for clarity.



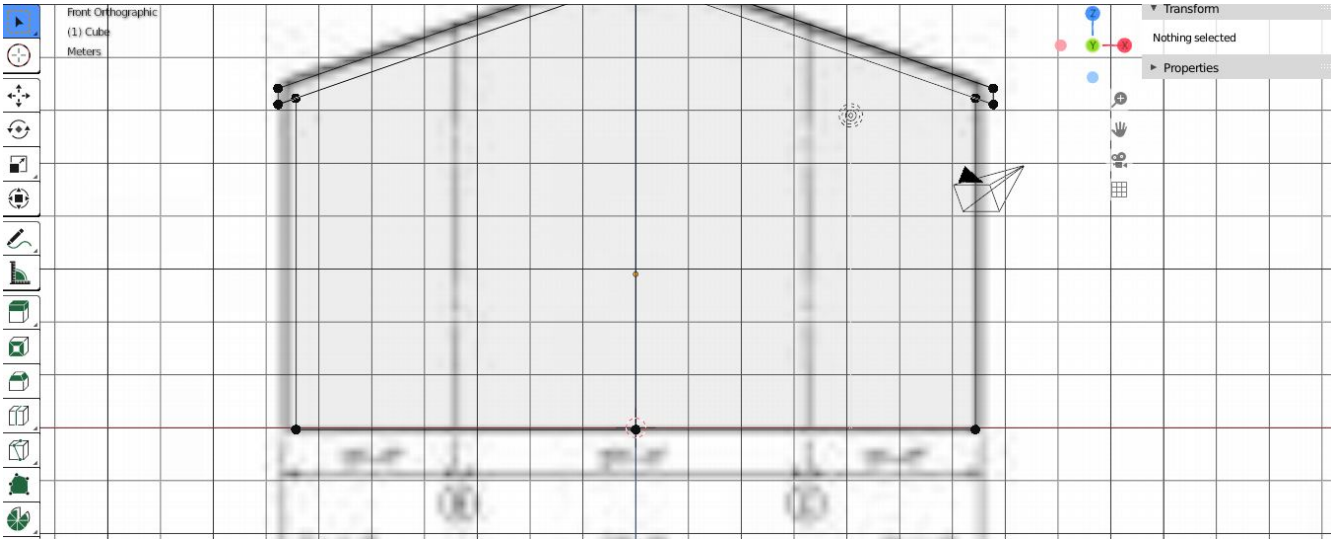
ONLY the middle vertices will reach the ground, as the side vertices still have the roof angle and will temporarily dip below ground.



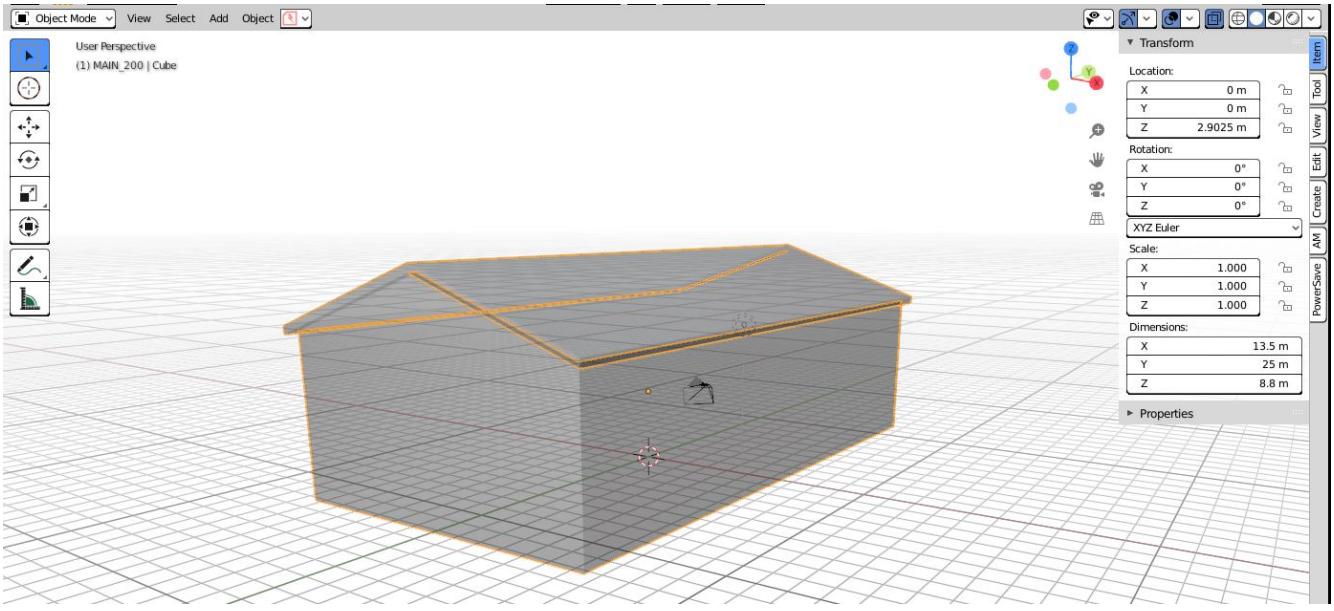
Some what hard to see, sorry

- ALT Z to toggle WIREFRAME mode on. Using `Wireframe/X-Ray mode`, switch to **vertex edit** mode with 1 and select all of the bottom vertices.
- Type: S Z 0 to flatten out the vertices on the bottom.

S X,Y or Z 0 is a nice magical incantation to remember to align vertices to a specified axis

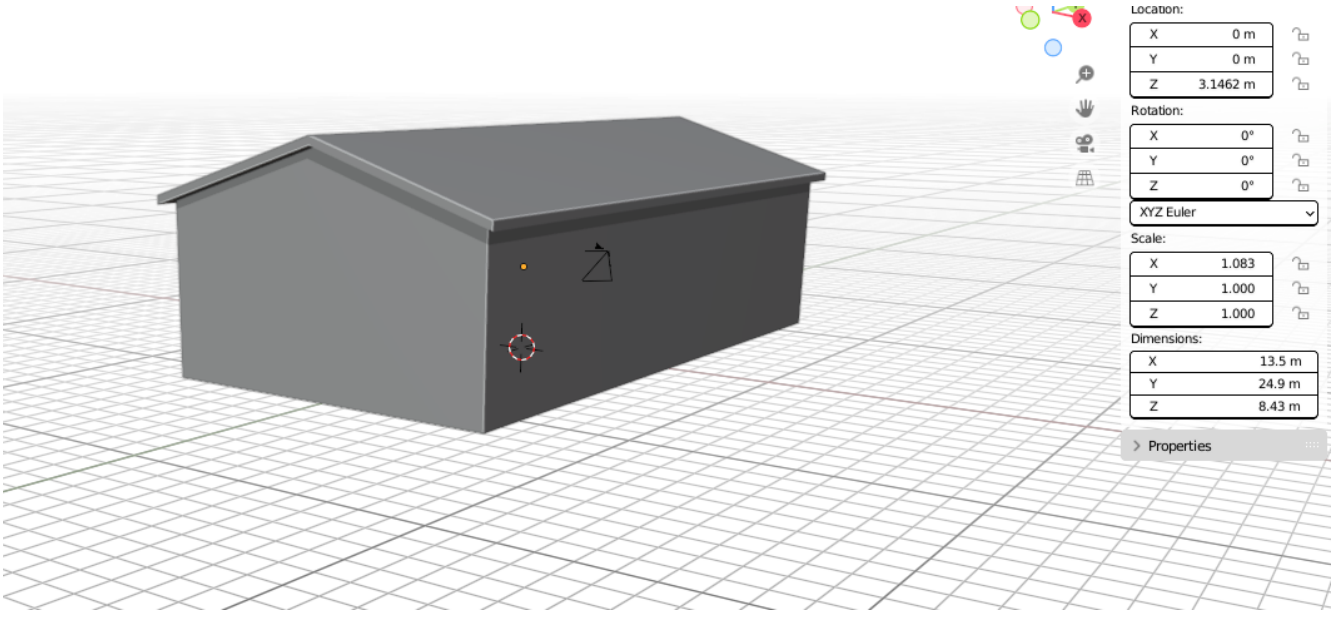


- While still in front view (KEYPAD 1), Shift all of the bottom vertices back to ground level with G Z as needed



- TAB back to OBJECT MODE and use Solid View, Z and select (SOLID), to view the result in greater detail.

Making the Interior



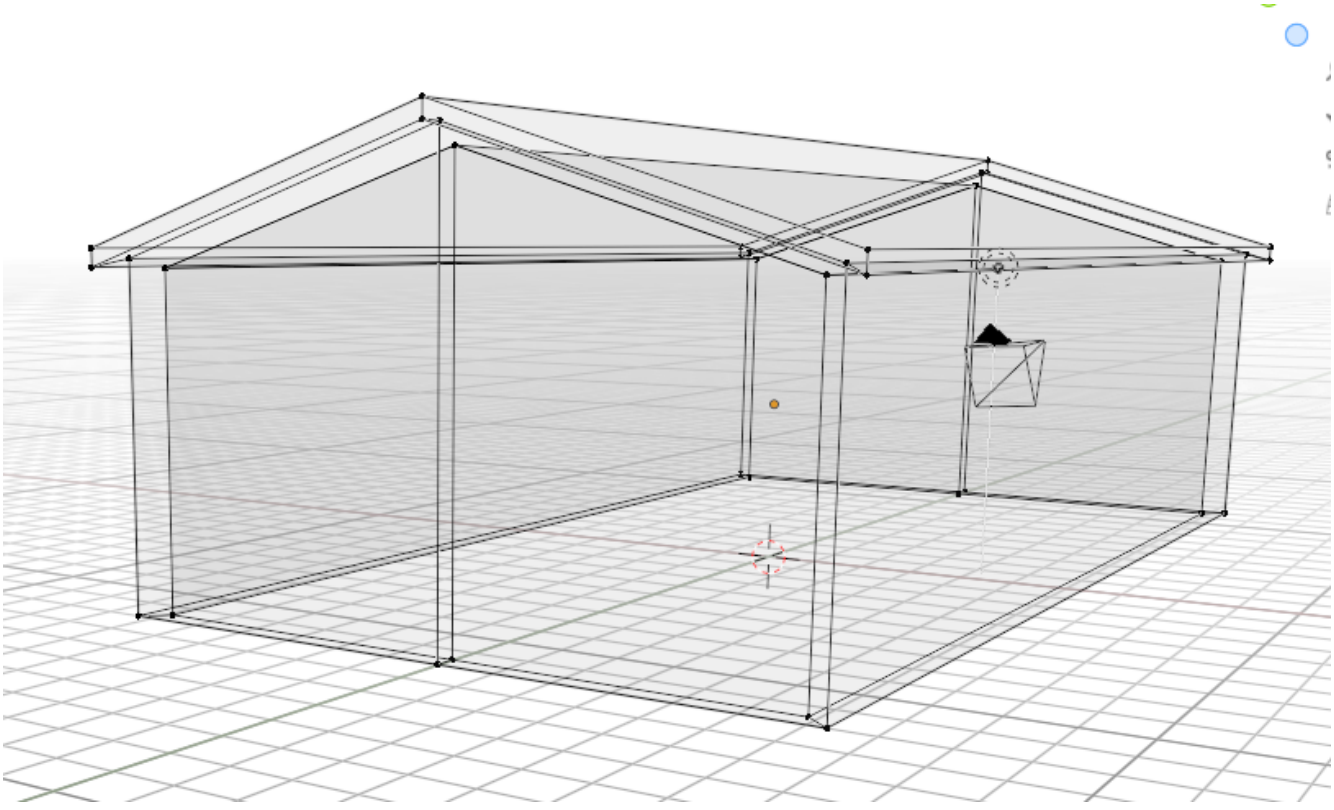
Now we need to give the building an interior.

- Shift your view to be below the model.
- Select the building and press TAB to enter edit mode (Unless you were already in edit mode)
- Chose X-RAY mode by clicking the X-RAY button on the top right or ALT Z
- Press 3 to use face selection mode and chose the bottom faces.
- Press i to Inset and adjust until you get a realistic wall thickness and press [ENTER] to accept.
- Press keypad 1 for front view and then E Z to extrude upward into (but not through) the model.

Note: The top of the inside of the model will be flat like the floor.

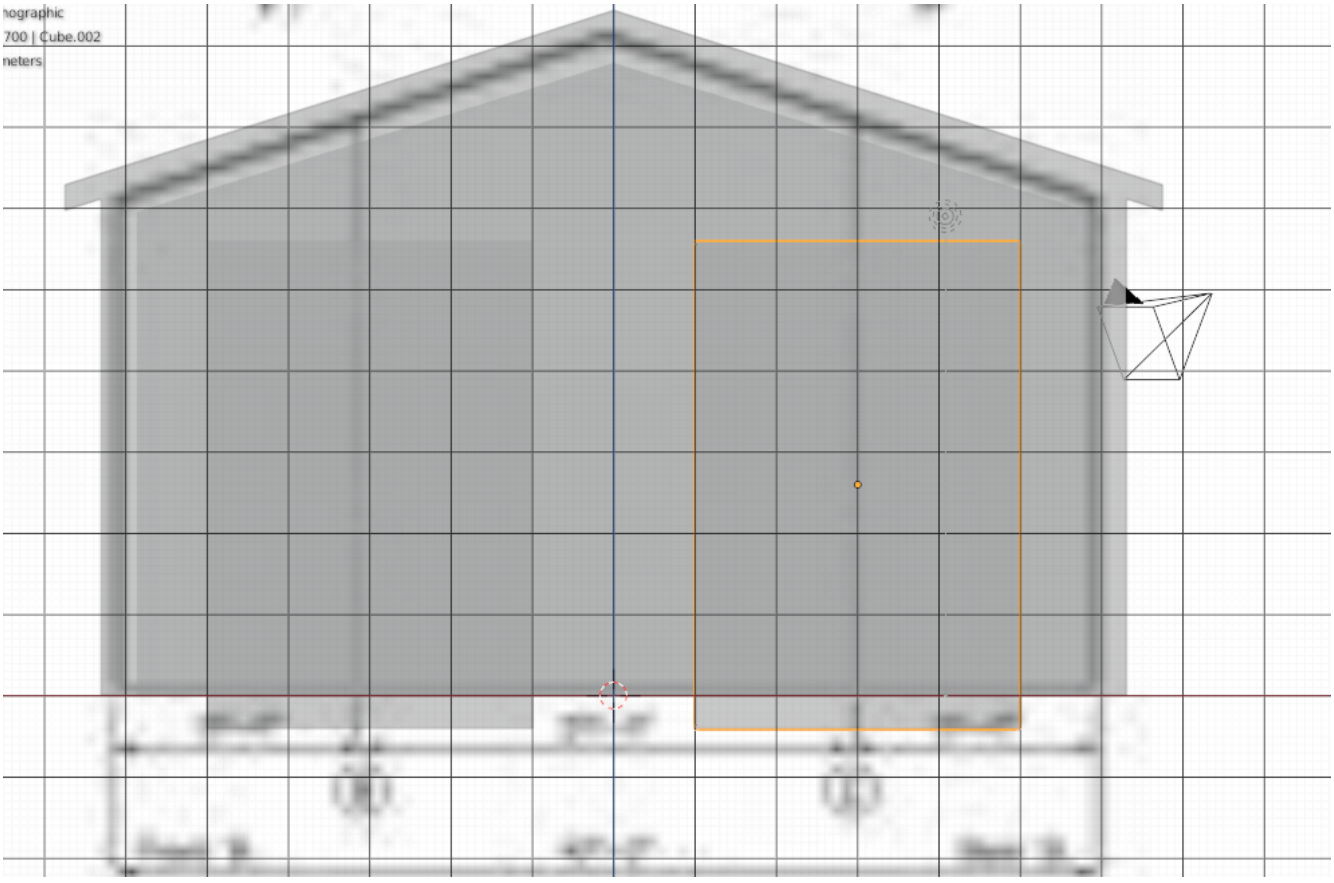
To make the roof match the top roofline and staying with the front view mode…

- Press `1` for vertex selection mode and then box select the middle vertices on top of the inside being cut out.
- Press `G` then `Z` and drag the vertices up to match the angle of the roofline.

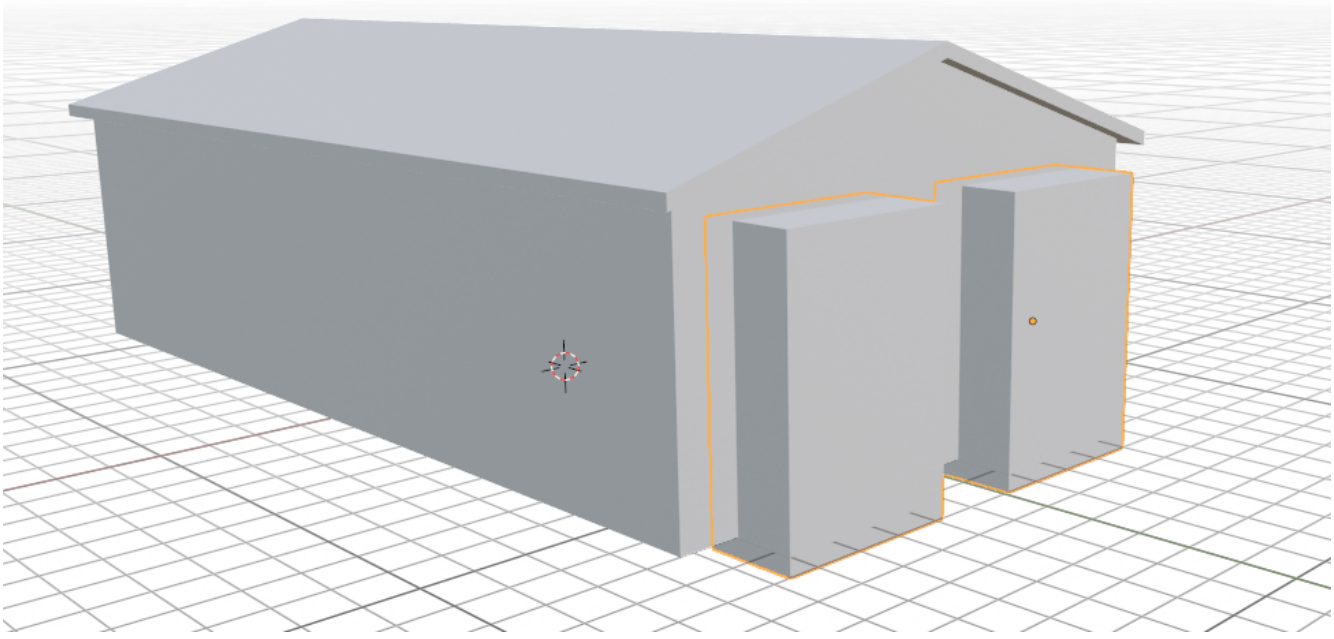


Making the Doors

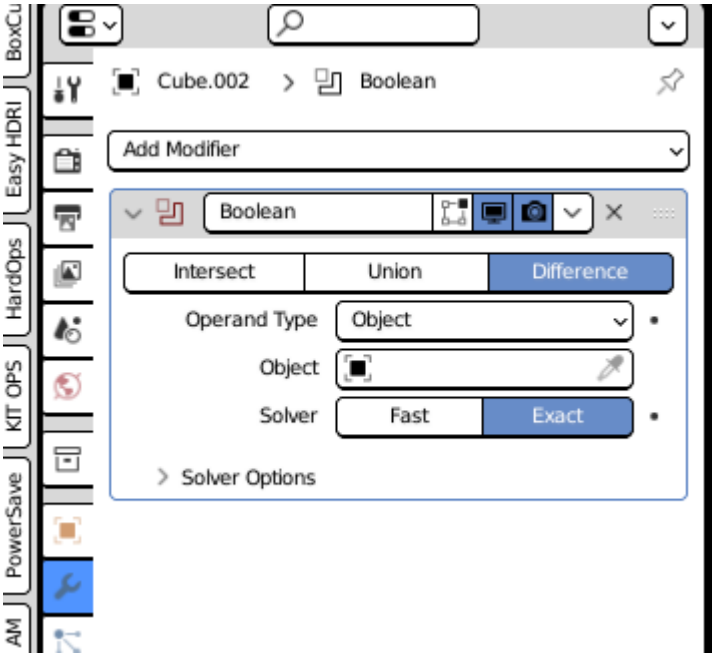
To make the main doors, we will use a boolean cutout method to remove the shape needed for door access.



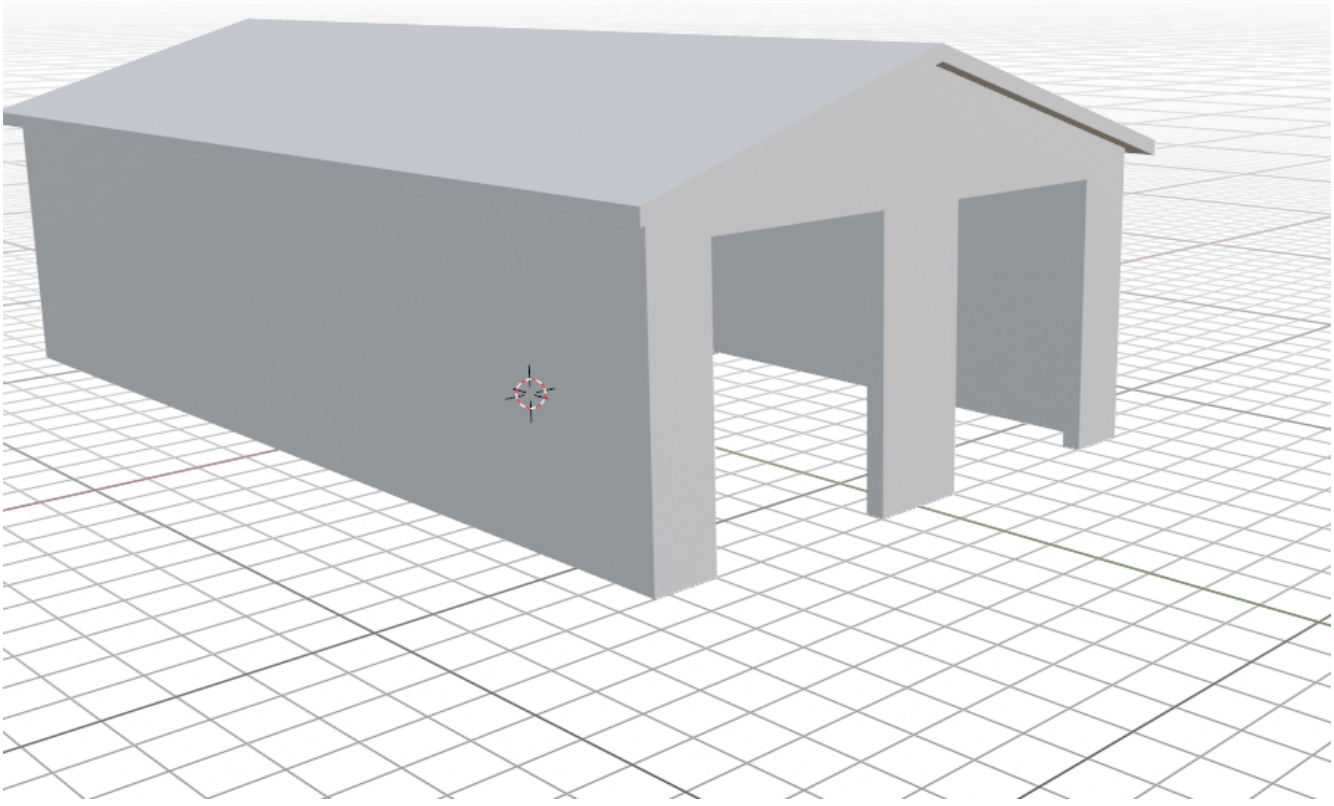
- Create a cube and resize it to: `X = 4m Z = 6m` using the dimensions fields in the N-Panel (press `N`)
- Move the cube so it's X position is -3m. `G` `X` `-3m`
- Move the cube so it's Z position is 2.6m. `G` `Z` `2.5m`
- Duplicate the cube: `SHIFT` `D` and set its X location to `3m`
- Select BOTH cubes and press `CTRL` `J` to join them
- Make sure you are in Xray mode: `ALT` `Z` and top view `Keypad 7`
- Move the joined cubes along the Y axis until it intersects the front wall from both sides. `G` `Y` or set Y position to about `-12.3xx`
- Select the Building object and use the modifier panel (wrench) on the right side of the screen.



- Chose `ADD MODIFIER` and select Boolean
- Make sure boolean option is set as `Difference` and click the EYEDROPPER icon



- With the EYEDROPPER, select the cubes we just created to make the door shapes. Since we won't be adjusting the door frame further, you **could** apply the modifier and we will see the new cutouts for the doors.



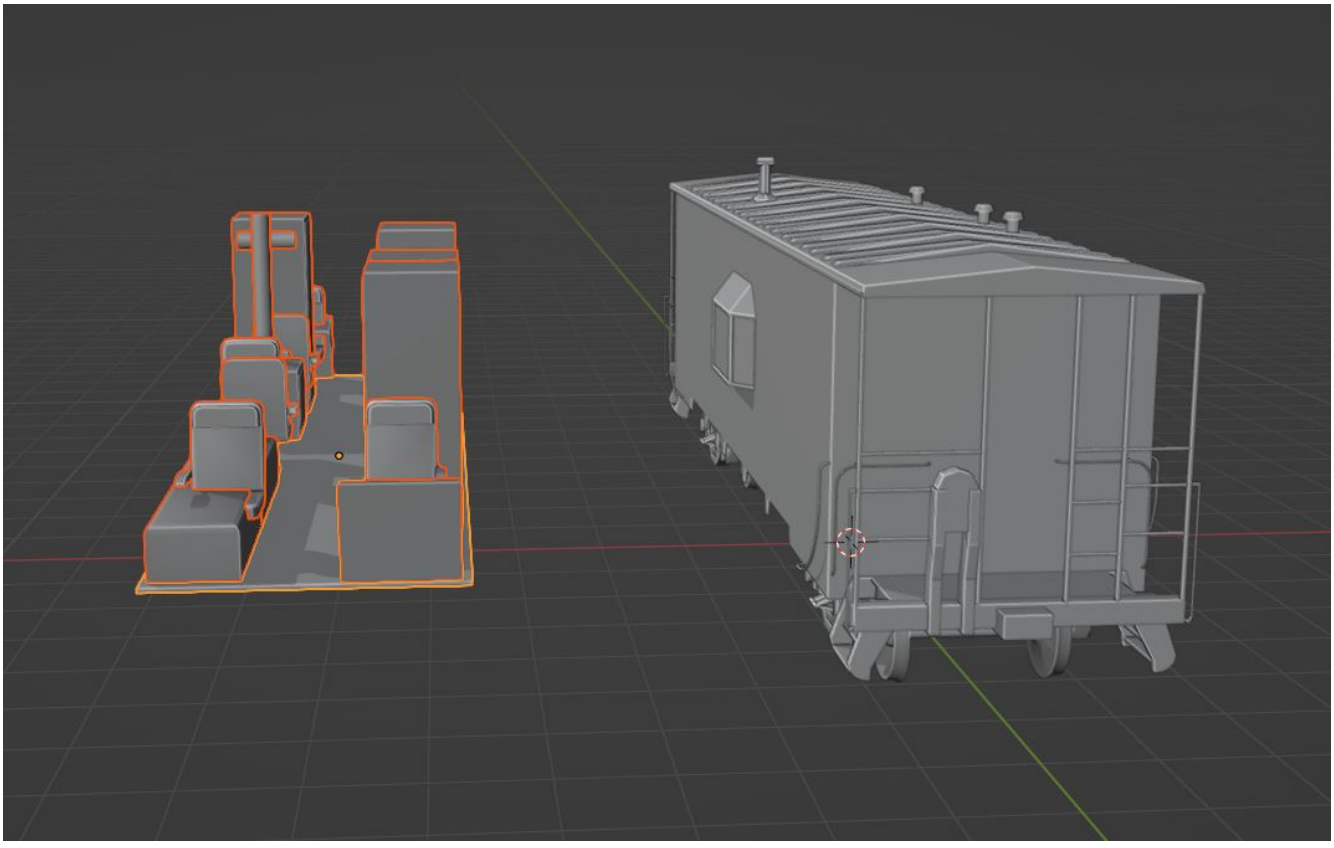
Moving multiple objects at once using the transform panel makes use of the `ALT` otherwise you are only able to move objects individually.

Example

Moving (transform location) ALL SELECTED objects at once.

Setup:

- Select a primary object in the group of objects that you want to move as a group.
- Press `SHIFT` `S` for the PIE menu and "Set Cursor to Active"
- Select ALL of your objects you want to move together and use **Object › Origin › Set Origin to 3D Cursor**



- Press SHIFT S for the PIE menu and "Set Cursor to World Origin"
- Now, lets say you wanted to move all of the objects to 0 on the X axis. Using the Transform > Location panel for X value input, HOLD ALT and key on "0" in the field.
- All of the objects should have moved to X = 0 location in the 3D View together.

Making a Vehicle

Creating a road vehicle is a challenge as many road vehicles involve complex shapes and curves. In this section, we are going to focus on making a vehicle used as a load on an open rack auto carrier.

As we delve deeper into the world of 3D modeling, we often find ourselves needing inspiration and reference materials to guide our creative processes. One powerful tool in the arsenal of a 3D artist is the use of background images to model a specific object or scene. In this section, we will explore how to use background images to model a classic 1970s Ford van, capturing the essence of that era's automotive design

Finding the Right Reference Images

Before we begin modeling, we need to gather a collection of reference images that showcase the van from various angles and perspectives. Luckily, the internet is a treasure trove of information and images from the 1970s, including photographs and advertisements for the Ford van. You might even be able to find an online model that you can rotate and position to take snapshots from various angles.

Online Searches

Start with a simple online image search using keywords like "1987s Ford van," "Ford Econoline 1970s," or "1970s van

advertisements." This will yield a variety of images that can serve as your references.

Classic Car Websites

Explore classic car websites and forums. Enthusiasts often share high-quality images of vintage vehicles, which can be invaluable for your modeling project.

Archives and Magazines: Check out digital archives of automotive magazines from the 1970s. These publications often contain advertisements and feature articles that can provide detailed reference material.

Organizing Your Reference Material

Once you have collected a sufficient number of reference images, it's essential to organize them for easy access during the modeling process. Create a folder on your computer dedicated to this project and categorize your images based on different aspects of the van:

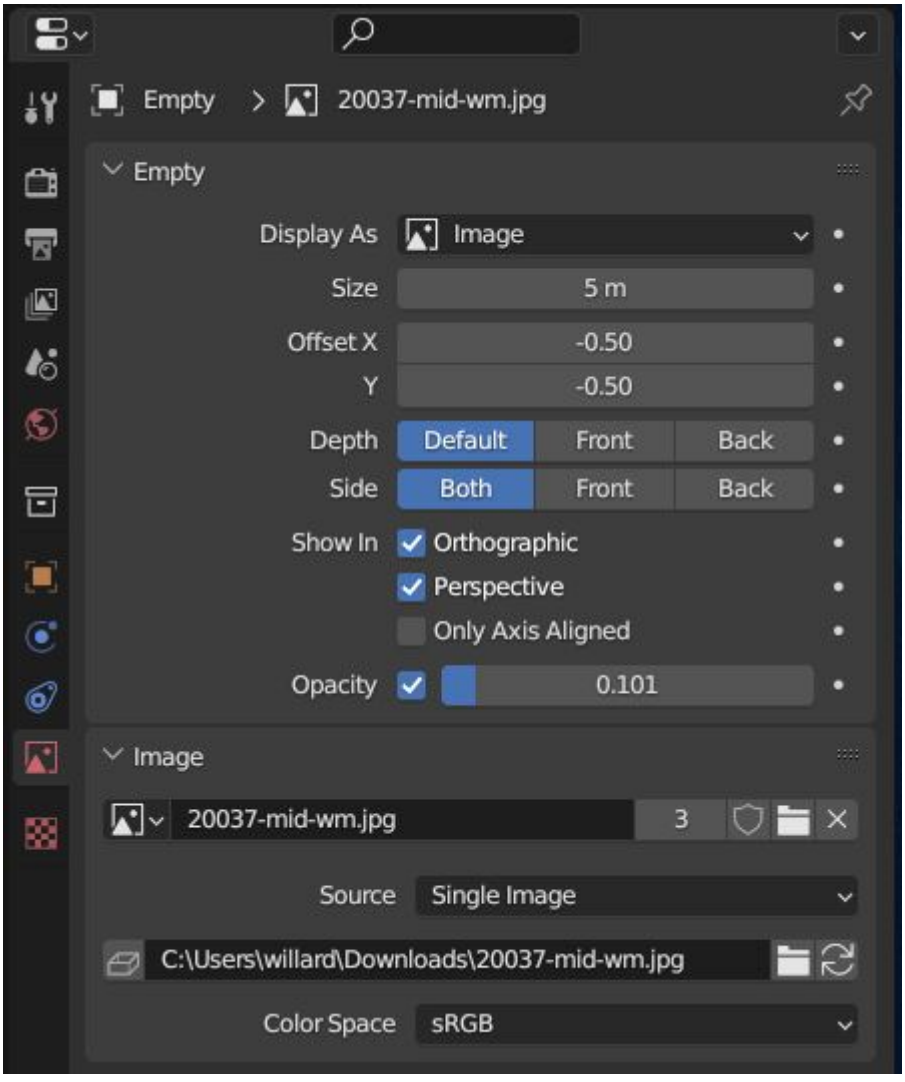
- Exterior: Include photos of the van from various angles, showing details like headlights, taillights, grille, and body lines.
- Color and Materials: Pay attention to the color schemes and materials used in the van's design. This will help you accurately recreate textures and finishes.

Using Background Images

Now that you have a well-organized collection of reference images, it's time to use them as background references in Blender. If possible, you modify the images to contain an Alpha channel so there will be less "white" making the drawing reference less distracting.

I often find that images are not 100% accurate, this one is no exception.

- Import images: In object mode, adjust your view to one of the primary viewpoints available in the reference images `Keypad 1`, `Keypad 3`, `Keypad 7`, for example. Using **Mesh › Add Image › Reference or Background)** to add an "empty" reference image to the scene as background images. Create separate views for the front, side, top, and rear of the van, and import the corresponding reference images for each view.
- Adjust Opacity: Reduce the opacity of the background images so that they don't obstruct your 3D model. You want them to serve as a visual guide rather than a distraction.



- Align and Scale: Ensure that the images are aligned correctly with your modeling workspace. You may need to adjust their scale and position to match the proportions of your model.

The Modeling Process

With your background images in place, you can now start the 3D modeling process. Use the images as a guide to create the van's exterior and interior components. Pay attention to the finer details, such as the curvature of the body panels, the shape of the windows, and the design of the wheels. You might want to break up the complex shapes as smaller faces that you can merge later.

The easiest way to start is to use a MESH PLANE, go to edit mode and use `CTRL-R` add then enough subdivisions `mousewheel` to match the edges in the drawings.

`To do: Add example`

Iteration and Refinement

Modeling a complex object like a Ford van from the 1980s may require several iterations and refinements. Continuously refer to your background images to ensure accuracy, and don't be afraid to make adjustments as needed. The goal is to capture the spirit of the era in your 3D model.

Let's do it.

Example Session

Making Rolling Stock - Project #4

Starting of this section with some tips related to creating rolling stock.

Remember these tips?

- Rolling stock is typically symmetrical so making use of the `Mirror` modifier to create the opposite side of a shape is a timesaver
- Consider "MARKING SEAMS" while you are modeling instead of saving it until you are ready to UV map your textures.
- Recycle as much as possible from existing models (Bogies, Couplers, Coupler Lift Bars, etc) Hint: Create a collection of re-usable parts.

Let's Say we are was making a simple flatcar. Nothing too complicated.

Using Background Images

As with the previous project, we load in our reference background images. I picked a random image from the internet using a Google search because… Why not?

Starting from the ground up

I like to start constructing vehicles starting with the trucks, and for this I'm going to recommend downloading a *.blend file to simplify this. Erick Cantu has graciously provided various trucks that we can include in our own models. They can be downloaded from Elvas Tower.

You really should not need to make your own truck and coupler models. These are pretty standardized objects with a minimal number of variations. Chances are good that you can find available models to import instead of making them yourself. In the long runm this will lead to consistency between modelers.

I have also uploaded a usable Blender example of 100T truck, based on Erick's there as well. Available items include: "Source files for NAVS N-Series freight car trucks", "Source files for NAVS Z-Series freight car trucks", "100T Roller Bearing Truck Blender Object" some of which are also available at Trainsim.com as well. Erick's trucks include attached couplers. I often choose to separate the couplers but that is left up to you to decide. Erick has good reasons for keeping couplers attached to truck rotations.

Using these existing objects, we can Append the ready-to-use model into our initial blend file.

I have been able to adjust the wheel radius as needed, though it's probably not entirely prototypical to do so on the trucks provided so far.

If you feel so inclined, you could create your own wheel sets from scratch, but I leave that up to you. Also note, since it is very likely we will be using LOD ranges with this model, understand that we will make the highest detail model first. It is much easier to take away than add objects later.

Vehicle LOD - Level of Detail Settings

In realtime computer graphics, a technique called "level of detail" (LOD) is commonly used in various applications ranging from commercial simulators to PC and console games.

When 3D games are running on a graphics card, there is a limit to the number of surfaces (polygons and triangles) and textures that can be drawn. If this limit is exceeded, the game or simulation starts to slow down and the movements become jerky.

If the graphics engine is asked to do too much then the program's framerate drops. This drop in framerate results in jerky movements and slower update rates, which can become very noticeable and take away from the sense of immersion.

Lets also look at an excerpt from the Open Rails Manual:

```
Many visual objects are modelled at more than one level of detail (LOD) so, when they are seen at a
distance, Open Rails can switch to the lesser level of detail without compromising the view. This use of
multiple LODs reduces the processing load and so may increase frame rates.
```

Always apply textures before defining any LOD settings.

Referring to some example snippets from the Outliner window in a Blender Working session… Watch out for the "GOTCHA's"…

```
Scene Collection
        MAIN
              MAIN_1000
                     BodyLOD1
                           Bogie1
```

Copying a part and adding it to the collection BodyLOD2. Outliner Result…

```
Scene Collection
        MAIN
             MAIN_1000
                    BodyLOD1
                         Bogie1



        MAIN_2000
             BodyLOD2
                   Bogie1.001
```

Looking at it now, shouldn't MAIN_2000 be directly under MAIN_1000 like this

```
Scene Collection
   MAIN
         MAIN_1000
                BodyLOD1
                      Bogie1


          MAIN_2000
                 BodyLOD2
                        Bogie1.001
```

Answer: YES, What happened is that MAIN_2000 was created at the same level MAIN not MAIN_1000. Only what resides below the MAIN collection will be considered for EXPORT.

Let's Look at some other issues here whith what is shown above… Don't forget capitalization.

For Example: All cases of **Bogie1** should be **BOGIE1**.

With the above example setup, the high resolution BOGIE1 part will disappear at 1000 meters. Then the low res BOGIE1.001 will take over. That low res bogie won't be animated, which is fine, since its too far away to see anyway.

Here's how to do it if you need the lower resolution bogie to be animated:

- Use an Empty for BOGIE1
- Link it to both MAIN_1000 and MAIN_2000 using drag and CTRL drop in the Outliner
- Attach the low and high resolution meshes to it in their respective LOD collections

```
Scene Collection
   MAIN
         MAIN_1000
                BodyLOD1
                BOGIE1
                    BogieLOD1


          MAIN_2000
                BodyLOD2
                BOGIE1
                    BogieLOD2
```

Don't forget to check out the sample locomotive file that is provided with the MSTS exporter download.

Once you copy the completed model to the Alternate LOD Collection, you can begin deleting details that won't normally be visible at longer distances.

Bitmap Editing

This section is about creating textures that will be applied to your 3d models. Its not about how good you are at it. Being good at textures takes time and practice, just like 3D modeling.

In an attempt to be remain bitmap tool agnostic, no particular tools will recommended or favored. Most options available have very similar capabilities, though some are just easier to use than others.

Layered Format Files

A feature of any proper bit map tool that is to be used with creating textures for our models is the ability to save files that retain isolated working layers. A bitmap tool that does not understand the concept of working with layered overlay modes is practically useless as you will want to be able to add weathering and decals to a base texture and control opacity values of layers.

To recommend a few tools and formats, the following list should be a guide on what you can use (These all support layers):

Tool	File Type	Comment
Affinity Designer	.afdesign	Proprietary, Native
Affinity Designer	.psd	An Import/Export Option
Affinity Designer	.tiff	An Import/Export Option
Gimp	.xcf	Native
Gimp	.psd	An Import/Export Option
Gimp	.ora	An Import/Export Option (plugin)
Krita	.krt	Native
Krita	.ora	An Import/Export option
Paint.Net	.pdn	Native
Paint.Net	.psd	An Import/Export Option
Paint.Net	.ora	An Import/Export Option (plugin)
PaintShopPro	.psp	Proprietary, Native
PaintShopPro	.psd	An Import/Export Option

- All of the above programs support the `PNG` image export format
- Some of the above support `DDS` image format, eg. Paint.Net and GIMP through the use of external plug-ins
- The `PSD` format was originally a proprietary format used by PhotoShop and some software has trouble opening all `PSD` format variations seen in the wild
- `TIFF` is also a layered format, though not as common and in some cases prone to some tools not fully understanding file contents. Testing needed.
- Open Raster, `ORA`, is a newer file format that is supposed to replace `PSD` as a standard solution for exchanging layered images between graphics editors. It's still not widely used.

One additional tool to mention is InkScape, which like Affinity Designer, is a Vector Graphics program that contains a "make a bitmap copy" option. Vector graphics are scalable without creating "jaggies" like you see with bitmaps. Unlike Affinity Designer, Inkscape is free.

'DDS' files use 'lossy' compression which means that you will lose detail and color information compared to your original texture.

Simple Tools like the native Paint options in Windows are not a good choice for creating textures for 3D Models.

While the list of tools is not a complete list of available software that supports layers, these are likely the most common. I have not included the obvious package named "Adobe PhotoShop" in the list. Adobe PhotoShop still uses the PSD format for its own files. If you own it (or rent it) then it will do the job. I no longer use this product so I can't determine which import/export file format options are available. I assume it can support PNG and DDS file formats as well.

In summary, choose a tool that has a native layered file format that can also support PNG and possibly DDS. For maximum compatibility, choose a free tool, like GIMP or Paint.net, otherwise, save your master documents in PSD or ORA format

Using DDS Textures

DDS is the file extension used to denote the Microsoft DirectDraw Surface (.dds) texture file format. The Microsoft DirectDraw Surface (.dds) file format stores textures and cubic environment maps, with or without mipmaps. It was introduced with DirectX 7.0. Although the format is not natively supported by most graphic editing applications, there are a number of Tools designed to handle the format.

Using .DDS textures for your materials has the advantage of reducing the memory usage of textures by slightly decreasing their quality (Though honestly, not as bad as the ACE file format). This is helpful if you have a limited amount of graphics card and system memory and want to include as many textures as possible. Additionally, this method is particularly suitable for organic/nature textures where precision is not critical.

It is advised that you always have a MASTER document saved in a native bitmap format since the DDS format is a LOSSY document format so you will potentially loose image quality with every edit session of a DDS file. The advice is to NEVER edit a DDS file and always work from a lossless master. If the creator of content only supplies a DDS file with their release, you should ask them for a master file, or only use the supplied file to generate a working master you can use to work from so you have minimal degradation.

For working with DDS, the easiest approach is to utilize Paint.net or GIMP as both can export DDS textures directly. Exporting large 32-bit ACE textures using the original MSTS tools is often not feasible and some people won't even have access to MSTS TOOLS is they only use Open Rails. The most significant advantage of using GIMP or Paint.net is that exporting is much more straightforward and faster compared to outdated programs like TgaTool2.

DDS is a useful format but many of the export options are not suitable for best performance.

The short explanation is:

1. Always use DXT1 compression with full MIPs, except…
2. DXT5 compression is OK only when you need alpha translucency (ie alpha values other than on/off)

One of the most significant performance issues with current GPUs is related to the texture size. All textures used on loaded tiles in a scene must fit into the GPU at once. Although modern GPUs have 2G, 4G or more, adding up the texture file sizes for all buildings, terrain, and rolling stock will quickly reveal that the GPU's capacity can be reached. This is especially true when using 2K and 4K textures. When the GPU reaches its limit, performance suffers because additional textures must be swapped out to the CPU every frame.

So maximum compression is the key to good performance. Just compare the size of uncompressed textures and you will see how bloated they are.

TYPE	USAGE	Comment
DXT1 no alpha	Textures without transparency	Normal maps without shine, All glow maps
DXT3	Menu icons / UI elements	No mipmaps but has transparency
DXT5	Textures with transparency	Mip Maps and Normal maps with shine (if we ever get that ability)

An alpha channel increases the file size so it should be left out unless it is needed. On color maps, the alpha channel is used for transparency, on normal maps for glossiness. If the texture has no transparency or the normal map has no glossiness saving them as DXT1 (no alpha) instead of DXT5 instantly saves on the file size for no loss.

You know how sometimes people report that their icons or textures become a rainbow pixel mess? That is related to gimp saving no mipmap textures wrong - it writes in the header that the image has 1 mipmap, but it has none and the game gets confused. I never used gimp and I can't find where I read about this right now, if I'll do, I'll update this post.

Again, when exporting compressed DDS files to the simulator, don't use them for editing. Ensure you are keeping uncompressed versions of the source files that you can load for editing so you don't accumulate compression artifacts.

With GIMP, you would use `export as` and then chose `select file type` and set the options for DDS such as Compression, MIPMAP, etc. Latest versions if GIMP seem to come with DDS support so no post-install plugin is needed.

With Paint.net versions newer than 4.2.2, DDS support comes bundled with the program. You would use the `save as` option and `save as type > DDS`. In the Save Settings window, you would select DXT1 and under Error Metric, check Generate Mip Maps and use Best Quality.

Preparing a texture

When creating textures. remember

> Texture mapping needs to be intuitive and functional.
>
> — Erick Cantu

Probably the best way to start a texture file is to create a background layer that contains the primary colors of your final model. If your base model is primarily "Tuscan Red", then fill your background with "Tuscan Red". Variants for Pennsylvania, for example, would be RED rgb(121,68,59), BROWN rgb(111,78,55), TAN rgb(166,123,91). Many tools allow you to enter a RGB color value into your editor.

You don't need to complete your texture before you apply it to your 3D Model. Even using the single base color would be OK.

For texture size, consider working with 2048x2048 textures. Try to avoid creating multiple smaller texture files versus one large file. You can always shrink your texture to 1024x1024 when all your work is done as the coordinate mapping will remain relative as long as the proportions remain the same. You could not, however, adjust 2048x2048 to 2048x1024 though, keep that in mind.

If your model is wide but not tall, as many vehicle models are, then you might consider starting with a 2048x1024 texture size, provided that you are only creating content for Open Rails. Open Rails will not have issues with textures that are not square, unlike Microsoft Train Simulator.

Once you have created your base texture for your model, you should a) Save it in the native format of your editor or in one of the Layered formats like `PSD` or `ORA`. b) Save a copy in `PNG` format for use with Blender. Copy the `PNG` file to your project folder for your current model so it is easily available during a Blender working session.

Keep in mind that some people who might consider repainting your model will want to make use of their own photographs of actual vehicles or buildings. This means that it would be "unkind" to these "re-skinners" if you were to split

up the sides of your model into multiple sections as they would have difficulty getting the sections to rejoin cleanly. Try to keep the side and top views as continuous shapes in your model and your textures.

Decals

A newer concept, and one championed by the NAVS technique, is to use a separate bitmap or multiple bitmaps to generate various car numbers without having the numbers backed into the main bitmap. This gives added flexibility to car rosters as custom car numbers are easily generated without resorting to difficult post-release editing of `.ACE` files and many have done in the past.

A DECAL is a small section of the main model that has a smaller `3d plane` object floated just above the surface of the model. This plane is assigned a set of UV coordinates that map to a specific number or numbers desired on a particular car. Using the Open Rails `INCLUDE` statement in a WAG or ENG file, you can specify decal mapping using the `FreightAnim` keyword to locate the related decal `S` file(s).

I'll share a Decal creation technique here using Python code for the so inclined.

Coding alternatives

For the more software minded, Python 3 can be used to layout your textures using a Python package named PILLOW, (PIL for short). Now this won't be a guide for using Python or Pillow, but I will share how I have been able to layout sections of a texture to create absolute placement and sizes for UV coordinates using code.

```
#!/usr/bin/python

from PIL import Image, ImageDraw, ImageFont

#from PIL import *

meter = 146

# This layout was used for a flatcar sides and end reporting marks that were assigned
# to `plane` objects that were "shrink-wrapped" to the main body in Blender.

if __name__ == '__main__':
    height = 2048
    width = 2048
    image = Image.new(mode='L', size=(height, width), color=255)

    draw = ImageDraw.Draw(image)

    # get a font from the LOCAL FOLDER
    # You need a local font for this to work.
    # get a font (disabled for now)
    #fnt = ImageFont.truetype("Hack-Regular.ttf", 40)


    # get a drawing context
    #draw.text((1,300),"^^^ Side",font=fnt)
    draw.rectangle(((10,5), (10+952,5+194)), fill = "black")

    # Draw End A
    #draw.text((1100,300),"End -->",font=fnt)
```

```
    draw.rectangle(((1000,5),(1000+554,5+505)),fill = "black")

    # get a drawing context
    #draw.text((1,300),"^^^ Side",font=fnt)
    draw.rectangle(((10,5+510), (10+952,5+194+510)), fill = "black")

    # Draw End A
    #draw.text((1100,300),"End -->",font=fnt)

    draw.rectangle(((1000,5+510),(1000+554,5+505+510)),fill = "black")

    # get a drawing context
    #draw.text((1,300),"^^^ Side",font=fnt)
    draw.rectangle(((10,5+510*2), (10+952,5+194+510*2)), fill = "black")

    # Draw End A
    #draw.text((1100,300),"End -->",font=fnt)

    draw.rectangle(((1000,5+510*2),(1000+554,5+505+510*2)),fill = "black")

        # get a drawing context
    #draw.text((1,300),"^^^ Side",font=fnt)
    draw.rectangle(((10,10+510*3), (10+952,10+194+510*3)), fill = "black")

    # Draw End A
    #draw.text((1100,300),"End -->",font=fnt)

    draw.rectangle(((1000,5+510*3),(1000+554,5+505+510*3)),fill = "black")

    del draw

    image.save("out.png","PNG")
```

I have also been able to automate the creation of various number styles with Alpha channel backgrounds. This technique utilizes TTF fonts to create each number as a 64x64 image that can be called as a separate decal. While not super efficient, it could allow for widely varying car numbers without too much effort. For USA, this method would rely on 6 separate decals and extra number slots would need a blank 64x64 alpha image.

Using this method is still in the experimental stage for me… but it is something I'm looking forward to making a standard technique I employ

```
#!/usr/bin/python
"""Script to generate small bitmaps with white numbers on
an alpha background for reporting marks.
 The output is a set of TGA files and a master file with
 items merged


Basic usage:
$ python3 reportingmark.py  (No file options are needed)

You need to edit values in the top of this file to change defaults
This code has been tested with Python 3.10.4 and requires the use of
the Python package "PILLOW".
```

```
To install PILLOW, use:

python3 -m pip install --upgrade pip
python3 -m pip install --upgrade Pillow

============================

Author: Pete Willard
Email: petewillard@gmail.com
Website: RailSimStuff.com
Date: June 8, 2023


Well, numbers make sense but you never know, there
is this guy at RailSimStuff.com that puts numbers
on !@#$%^&*() characters.

The TTF font you plan to use does not need to be installed in the system.
The TTF file just needs to be inthe same folder as the python script.

"""

from PIL import Image, ImageDraw, ImageFont, ImageOps
from pathlib import Path
import os


# all reporting mark numbers must be the same length
numberList = "120079","120100","120186","120156"
elements = len(numberList)
element0 = numberList[0]
len_element0 = len(element0)
len_number = len(numberList)

# Reporting Mark - Road Name - refer to font PDF to know which chars make the
# correct lettering
rm = "NS"
len_rm = len(rm)


# Gap Size (NOTE: Not all railsimstuff fonts have a *space* character)
# Mileage may vary
space = "  "
len_space = len(space)

# Lettering height and width
height = 64
multiply = len_rm + len_space + len_element0
width = 54 * multiply  # 64 * 10

print ("height = ",height)
print ("width =  ", width)
# You will need to tweak these values below based on the
# specific font being used so it fits the 'box' correctly
fontSize = 68       # Pitch
fontHorz = 20       # Start Position
fontVert = -4       # Start Position
pathToFont = "nslogo.ttf"   # Should be in the local folder where the Script is
```

```
fontColor = "255"           # 255 = white


print(pathToFont)

if __name__ == '__main__':

    image = Image.new('RGB', (1024, 1024), color=0)
    image.save('decal.tga', 'tga')

    # get the font
    #

    fnt = ImageFont.truetype(pathToFont, fontSize)


    #"""
    # Draw Character Black on White Background
    # then invert to White on Black Background (it's just easier)
    # since we can rely on defaults

    #We are looping through each member of the numberList
    #and writing out each character result individually
    #"""

    count = 0

    for elements in numberList:
        output = rm + ' ' + elements
        #output = 'NS !@)!&^'
        print (output)
        #Setup
        image = Image.new(mode='L', size=(width, height), color=0)
        draw = ImageDraw.Draw(image)
        #
        draw.text((fontHorz,fontVert),output,font=fnt,fill=255)
        #draw.text((10,),output,font=fnt,fill=255)



        # Save out the results

        out = str(count) +".tga"
        count = count +1

        # Not the most efficient routines
        # but I'm still designing this next section

        image.save(out,"TGA")

    img1 = Image.open(r"decal.tga") # Create a blank to paste into
    row = 0
    for items in range(count):
        img2 = str(items) + ".tga"
        img = Image.open(img2)
        img1.paste(img, (0,row), mask = img)

        row = row + 64
```

```
    img1.save("decal1.tga") # remove working copy
    os.remove("decal.tga")

# Well, it was SUPPOSED to make an alpha channel... :(
# Still working on it.
```

Layering Basics

When working on a texture for a model, you are going to want to add bitmap layers that help to achieve the desired final result. This means being able to adjust the parameters and effects that each layer brings to the final result. These include making layers with more opacity so the details of lower layers are not obscured, or adding upper layers that contain effects for Grime, Rust, Dirt, Dust, as explained in the now lost "Painting Guide" that once existed on the 3DTrains.com website.

Since the 3DTrains website is now off the Internet, some of the things that I learned from that website will be shared here. Hopefully,3dTrains folks don't mind that I've tried to share that information here as a sort of archive of what was at the website.

There was an explanaion there that you would have you add four NEW layers to your base image and name them Grime, Rust, Dirt and Dust.

Grime Layer

- Set the airbrush tool to a width between 150 and 200
- Select a black color, lets say RGB 10,10,10, for example
- Spray all over the layer making sure it looks uneven and spotty
- Now **hide** this layer from view

Rust Layer

- With the same airbrush settings, choose a rust color like RGB 136,57,4
- Make sure its sprayed on so you can still see through it.
- Now **hide** this layer from view

Dirt Layer

- With the same airbrush settings, choose a yellowish/brown color like RGB 126,113,38
- Make sure its sprayed on so you can still see through it.
- Now **hide** this layer from view

Dust Layer

- With the same airbrush settings, choose a light color like RGB 192,192,192
- Make sure its sprayed on so you can still see through it.
- Un-hide all the layers

Finalization of effects

- Set the transparency/opacity properties of each of the new layers to somewhere between 10 and 30 percent

- Adjust the percentages of each weathering layer to get the best effects
- Save the file in the layered format
- Save a copy in the PNG format for use with Blender

Applying Fonts and Lettering

With the layered format file open, create a new layer just above the BASE layer in the document. This will make sure that the lettering being added is below the weathering effects.

While it might seem like a shameless plug for my website, you will find a number of railroad related fonts at http://www.railsimstuff.com to help with adding lettering and logo details to your textures. In case you are wondering, the fonts are all free and I make no money from this web site, in fact it only costs me money to keep it running so it is a labor of love that I have provided for nearly 20 years.

As mentioned above, the opacity of the DECAL layer should also be adjusted so it does not hide underlying details. The effect for decals though needs to resemble having been painted on, so the opacity will mbe much closer to 75% versus a lower value.

Layering Tips from Erick

Eric Cantu on Weathering

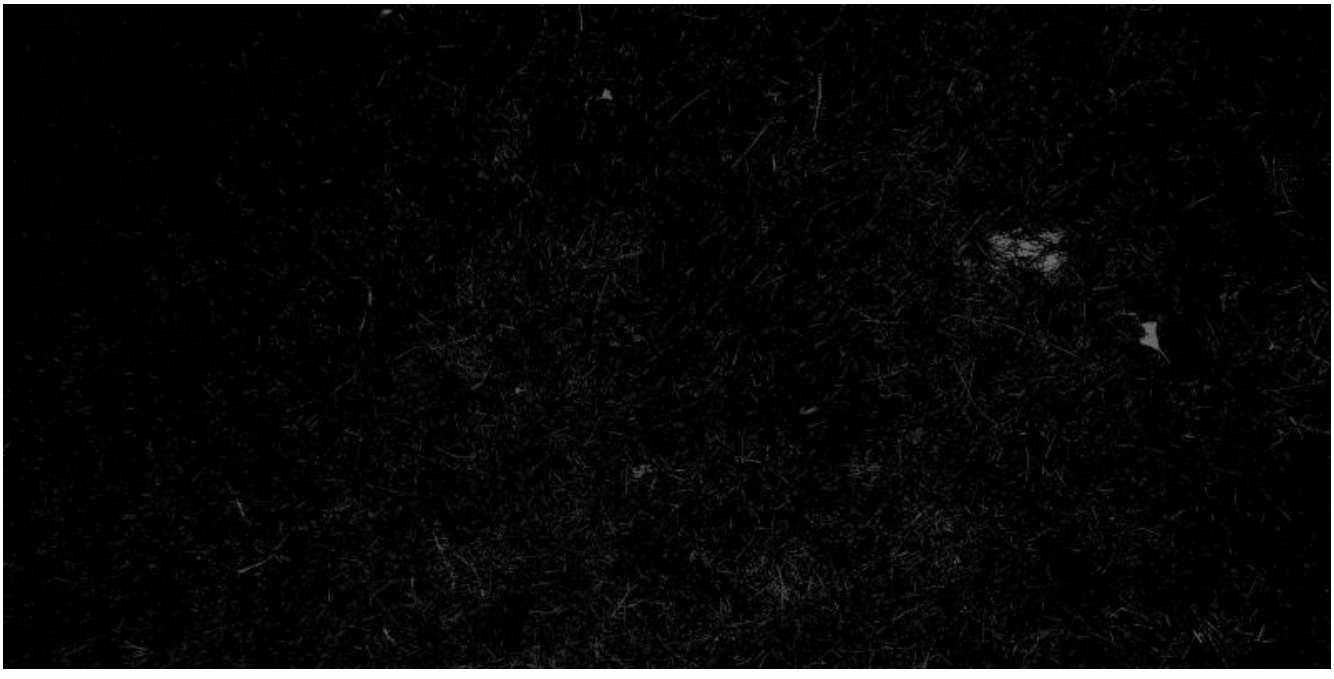
When I'm walking around, I often find myself taking photographs of dirt, gravel, grass, concrete, and the like. You might think that I take these photos to use as textures. You're sort or right, but mostly wrong. I take these photos primarily to create layer masks used in weathering cars.

The master textures for all of my cars are always set up like this, from top to bottom:

1. A top mask to keep the overall image tidy
2. Any standalone parts that need to not be affected by the shadow layer
3. A highlight layer which adds a little bit of highlight to selected areas
4. A shadow map which contains the bulk of the detail, including panel lines, ribs, and so on
5. Several weathering layers
6. Car markings
7. The base colour layer
8. A wireframe layer for reference (I never look at my cars in shape viewer as I'm painting - first, it wouldn't work, second, the wireframe layer makes it unnecessary)

Here's an example of a simple weathering technique. Freight cars get beaten up pretty severely on the road. They often end up with dents, gouges, and scratches. We can easily create textures for gouges and scratches with photographs of grass. I start with this photo:

I then turn it to greyscale and darken it significantly while bumping up the contrast:



When you use a greyscale image as a layer mask, pure white areas will be opaque, while pure black areas will be transparent, with values in between being semi-transparent to varying degrees. It's an opacity map. I can then take a photograph or dirt, or really any image of the right size with some dark colour, apply this image as a layer mask, set the properties to "multiply," and end up with dark, scratchy areas all over the carbody. But I can milk that image some more. If you're trying to maximize your output while minimizing your time, it pays to get the most out of all of your resources. I copy the layer, rotate it 180 degrees, invert the colours, and set the properties to "addition." I decrease the opacity to 30%. The end result is this:

Because the textures for the car are not perfectly symmetrical, it's hard to tell that the light, additive layer is the same image as the dark layer, but rotated 180 degrees. Similarly, I can rotate both images 180 degrees, change the opacity values slightly, and add perhaps another layer of spotty dirt, and those same layers easily create a carbody that looks totally different. You could go back to the original image and flip the scratch layers horizontally for a third carbody, or vertically for a fourth.

I am always on the lookout for walls with streaks of dirt from the rain, rusty metal, or anything that looks patchy. You can get so many great layer masks from those things, and most of us are carrying a perfectly-adequate camera in our pockets these days anyway. I used to hate weathering. Now it's quite easy and enjoyable, taking very little of my time.

How to Make Night Textures

Much of this is content is a summary of KUJU supplied documentation

Night textures on shapes are created by editing the original texture, darkening it and perhaps adding a couple of touches for effect.

The night textures take effect at a predetermined time within the game environment, so as to replicate a real world environment. The `extshape.dat` file must be updated with the correct parameters so that the shape is declared as having night textures (see the “How to write a .ref file” document for further clarification).

Once created, the night and day textures must have the same filename so that the code can pick up the correct texture. This means that the daytime / normal texture must be entered into the normal route textures directory and the night version into the route night textures directory.

Below are two textures taken from the Orient Express level. These were manipulated in a two dimensional drawing package:

image::images/image1.jpeg[]	image::images/image2.jpeg[]
image::images/image3.jpeg[]	image::images/image4.jpeg[]

This means that the texture on the left will be replaced with the texture on the right once the correct night timing has been switched on.

How to Make Night Textures with Backlighting

Example:

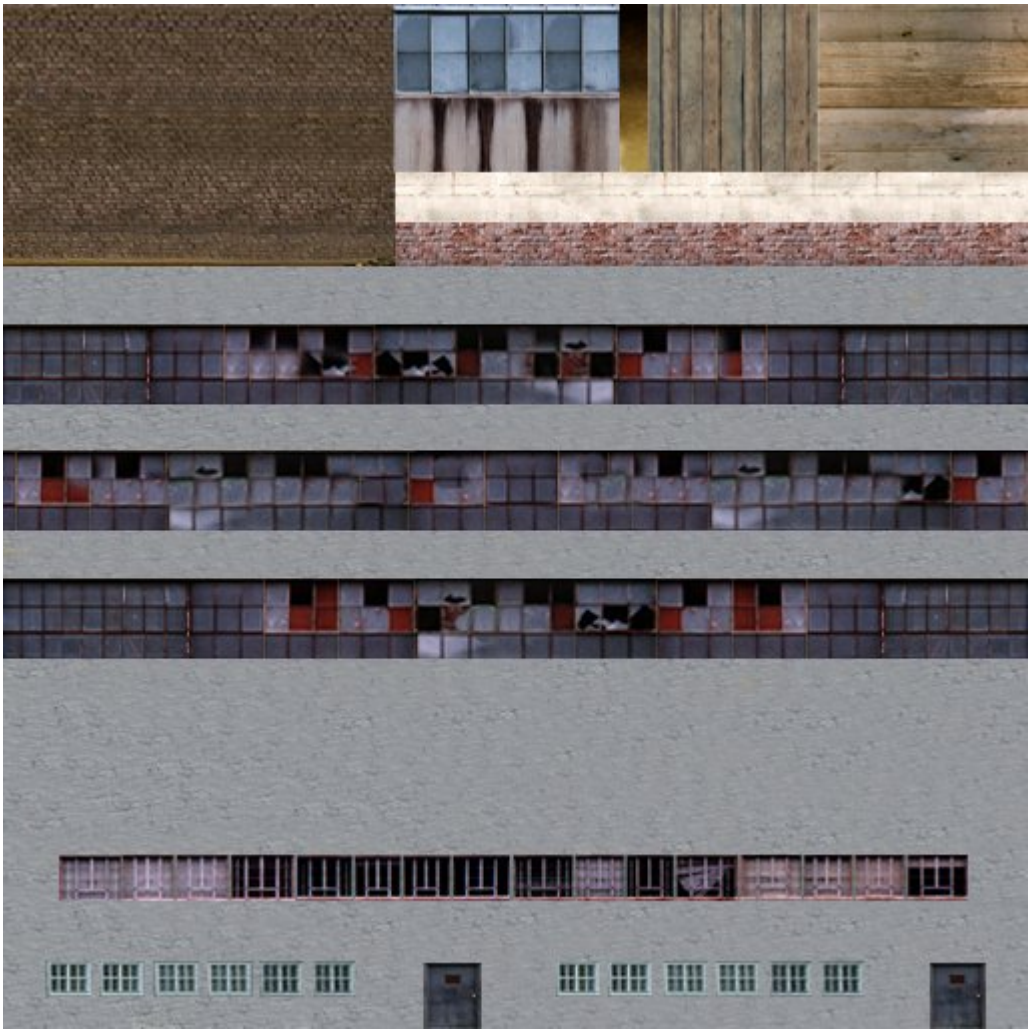
Create the image for the building you want to texture, making sure that anythng that requires backlighting is a separate objects and textures. Items such as windows should be separate objects, often created by just using a PLANE object. The basic buulding shape should not have modeled windows

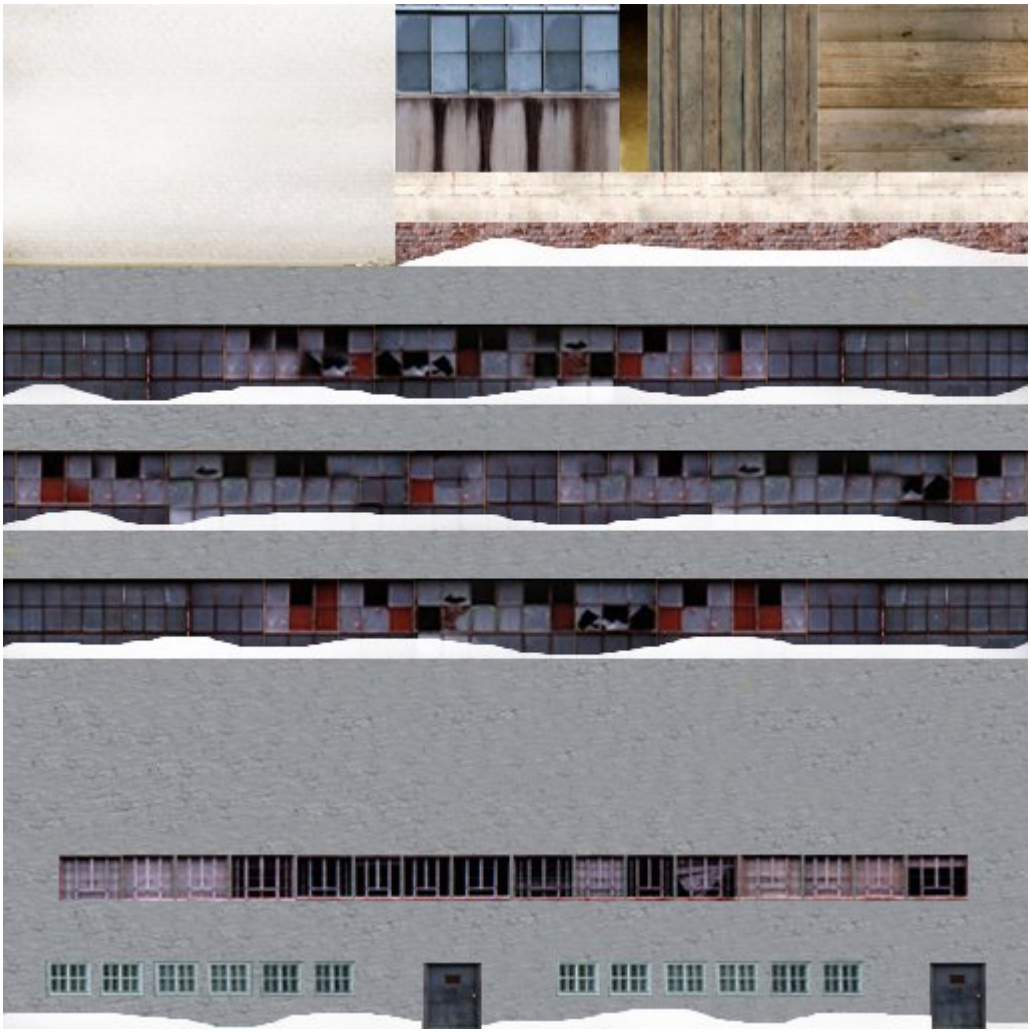
How to Make Snow Textures

Snow textures must also be created, so that your buildings and other objects will look correct when there is snow lying on the ground. These, too, are created using the original, daytime textures. The extshape.dat file must be updated in relation to this (see the “How to write a .ref file” document for further clarification) and must be placed in the route’s snow textures directory accordingly. The snow texture will only be used when the snow environment settings have been switched on through the Drive a Train User Interface.

Below are the snow versions of the textures above.


STAMS
STAMS





Highlights and Shadows

A layer, or layers, used for "hard coded" shadows and possibly highlights, which could otherwise be known as the clusion layer, should reside near the DECAL layer and the BASE layer for a decent effect, though you could migrate the highlights layer higher in the stack. How to create an clusion layer using the Blender render engine is covered in another section of this document, but if you do create this layer using Blender, this is where it would be inserted. The color of this layer will essentially be only black and white. You can always choose to create and edit this layer manually as well.

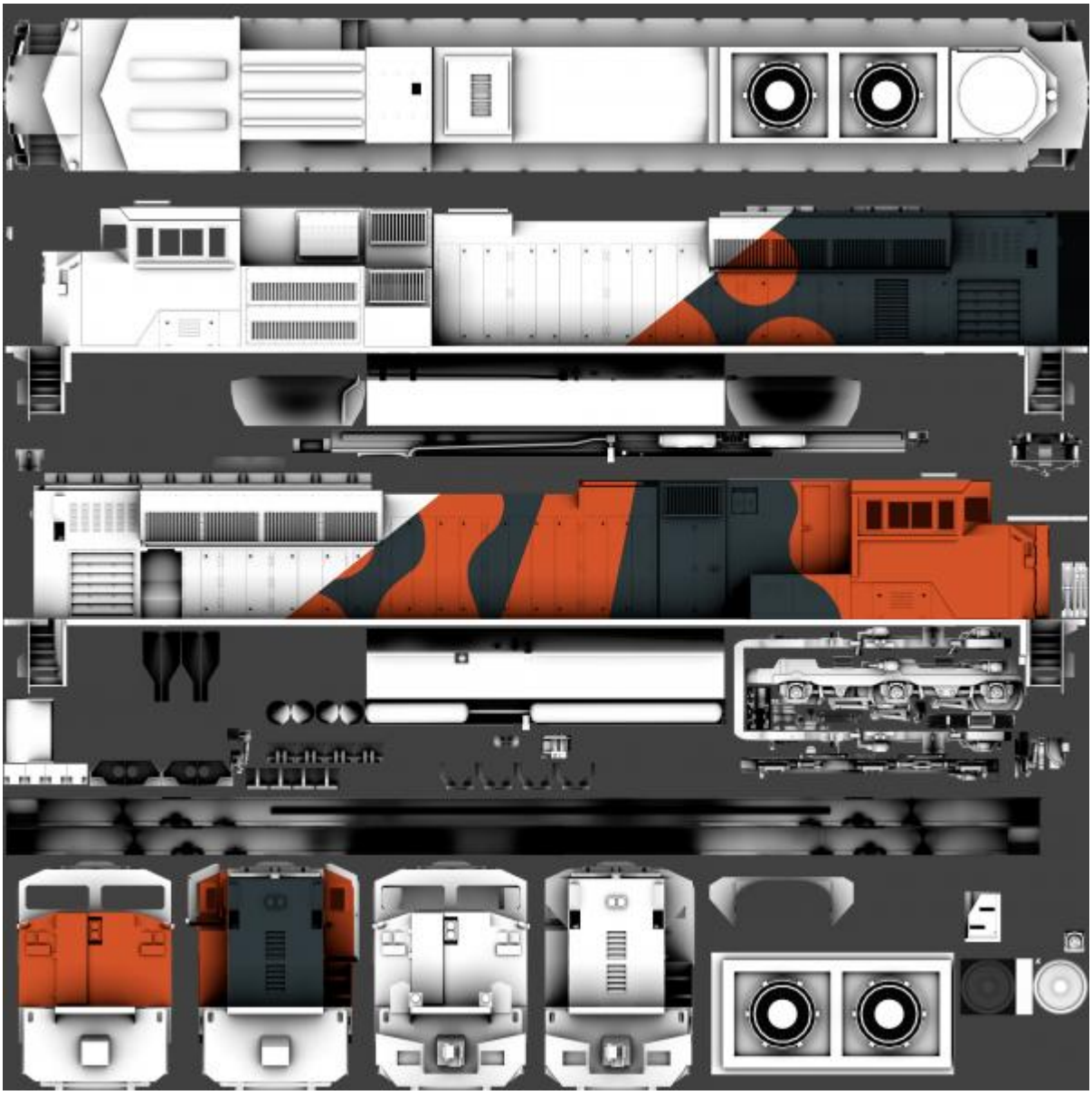
The blend mode for the [ao] layer could also be "multiply" instead of "normal"

Generated Shadows

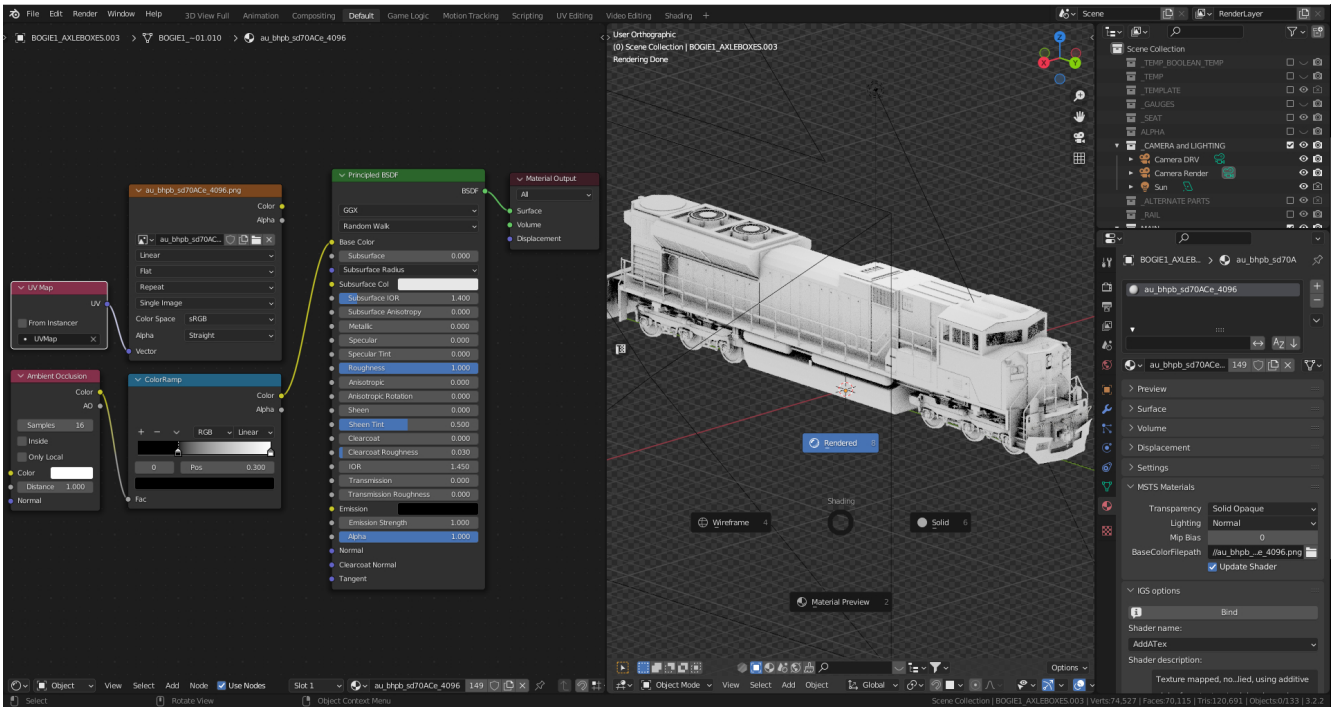
clusion is the generation of hard-coded or baked shadows instead of relying in dymanic lighting to generate shadows for an in-game asset. It provides extra depth to an asset that would not be achieved otherwise.

Marek on Elvas Tower shared how he achieves clusion on his models and it is shared here.

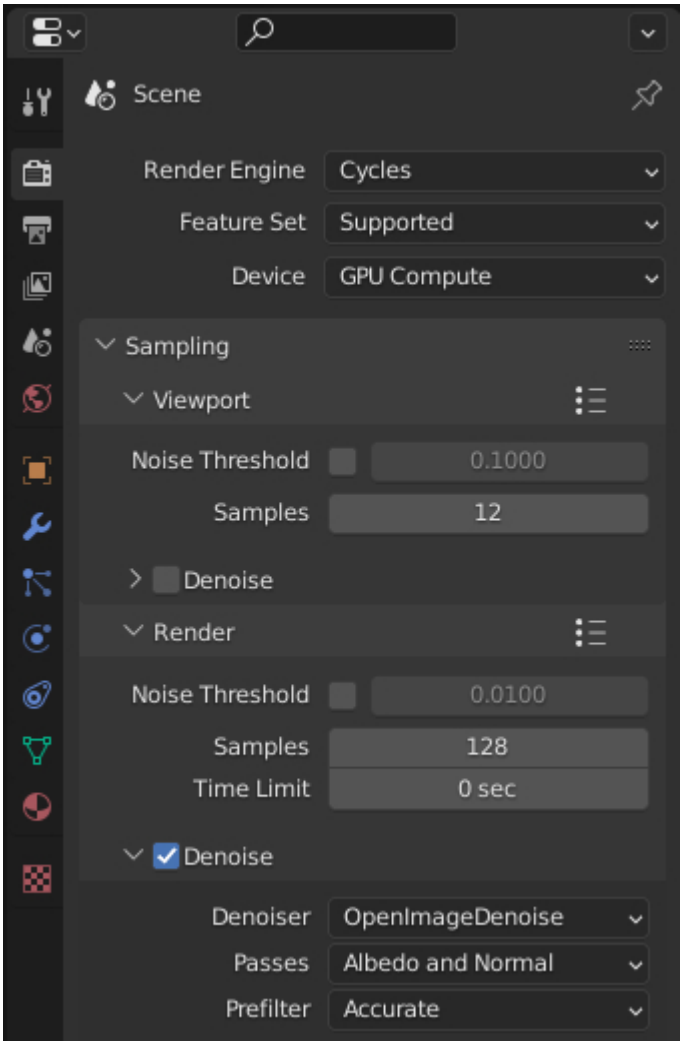
Your model needs to be UV UNWRAPPED prior to baking out an Ambient Occlusion image as described here.



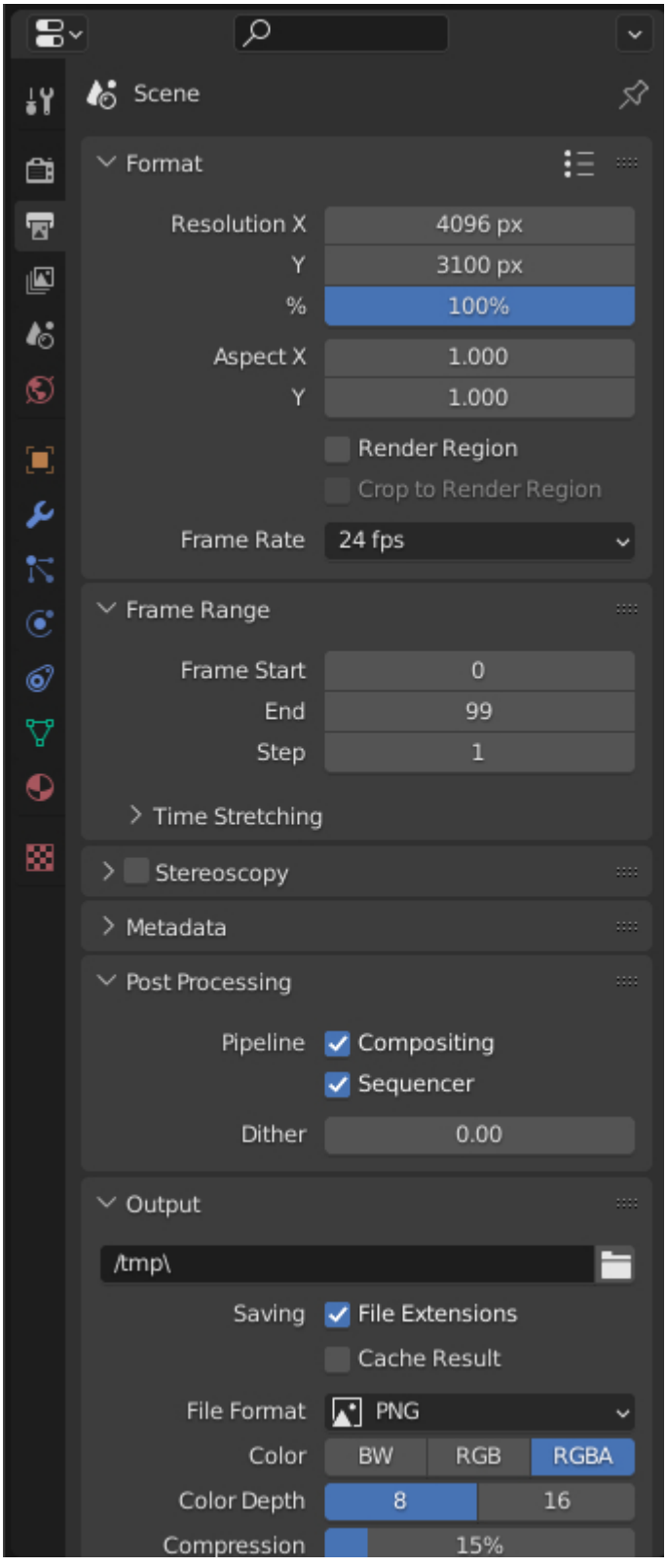
An orthographic camera can be used to render the clusion with all projection planes. Clipping planes can then be used on the camera to make certain parts invisible to the render camera, which helps exclude details that are not desired. The renders can then be imported into GIMP (or any other editor that supports layers) and used as the base for the final texture layout. The model can then be unwrapped to that. Livery colors can then be added in layers above the Ambient Occlusion layers and layer blend modes can be used to create the desired image. This method is more time-consuming than other methods, but it produces results that are far better than what can be achieved by hand in 2D alone.



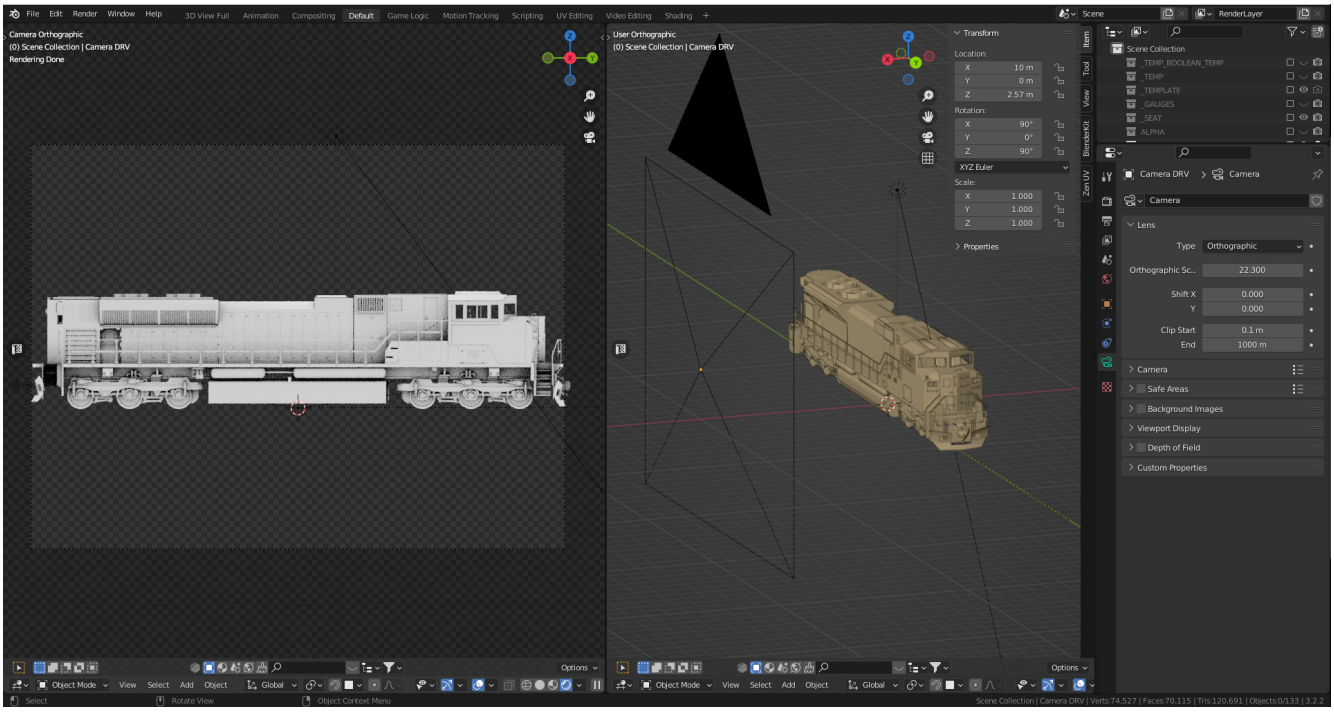
The image above shows a model with a single material applied to it called `au_bhpb_sd70ace_4096`. In the shader editor, two inputs can be chosen for the Base Color of the `Principled BSDF:` either an image file applied to the model at the top or an Ambient Occlusion shader via a Color Ramp node below it. When exporting to `.S` or wanting to view the texture in Blender, the `texture node` needs to be plugged in, and when rendering the Ambient Occlusion, the Ambient Occlusion Shader needs to be plugged in. By setting the 3D viewer to `Rendered`, one can get an idea of what the output will look like and can adjust the shadow effect via the sliders in the ColorRamp.



In the Render properties tab, switch your Render Engine to Cycles and your Device to GPU Compute for faster rendering. With a Render sample setting of 128, rendering can take a while on a PC. To get a faster render time when testing, try using a lower number initially. Don't forget to turn on `Denoise` to give you a cleanly rendered image.



In the Output properties tab, the `Format Resolution X = 4096 px` is used because there is a 4K texture being used on the model. You can adjust the Resolution Y value later when you know how much vertical space the render will take (see below).

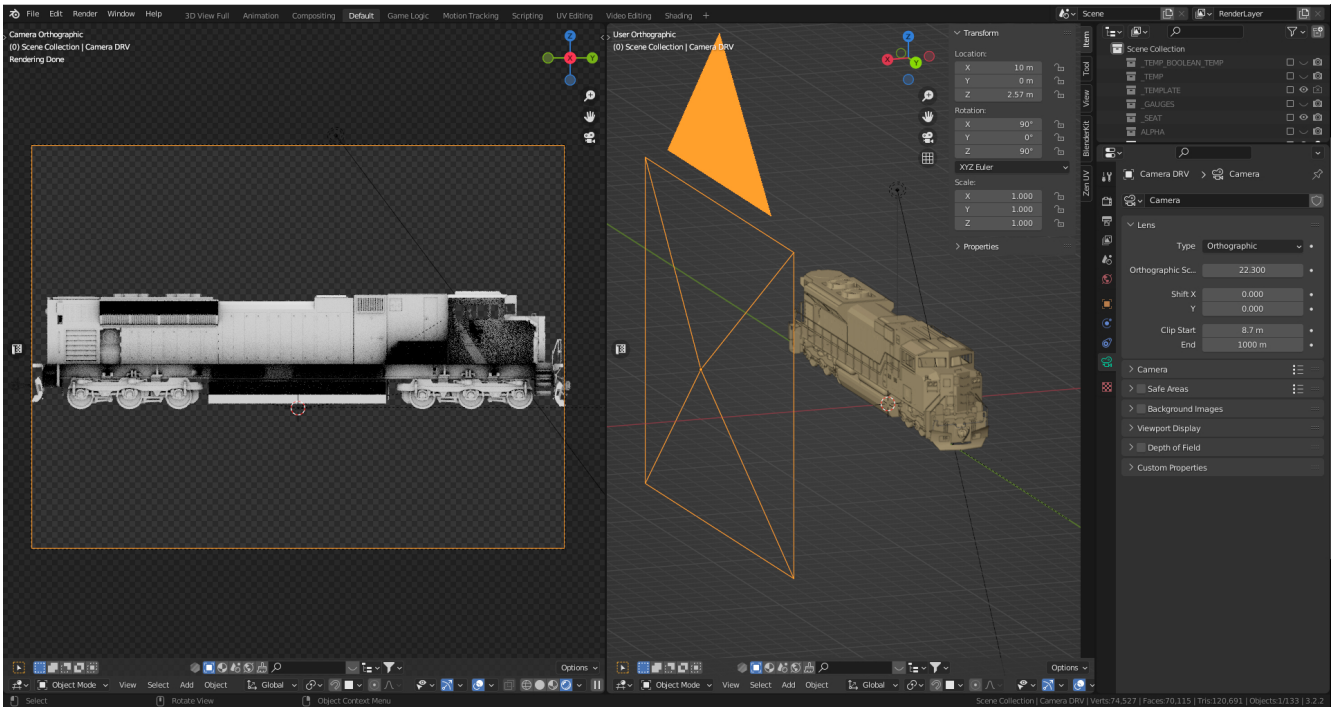


A camera should be added to the scene and moved to the side of the locomotive. The Rotation values in the Properties tab should be used to ensure that it is perpendicular to the locomotive. On the Object Data Properties tab, the Type of Camera should be changed to Orthographic. A separate 3D viewport should be opened and the Camera should be selected and `CTRL` + `Num 0` should be pressed to get a side view of the locomotive from the camera's point of view. If it is not already in Rendered view, pressing `Z` should allow for Rendered view to be selected; the Ambient Occlusion shader should be visible in the camera view.

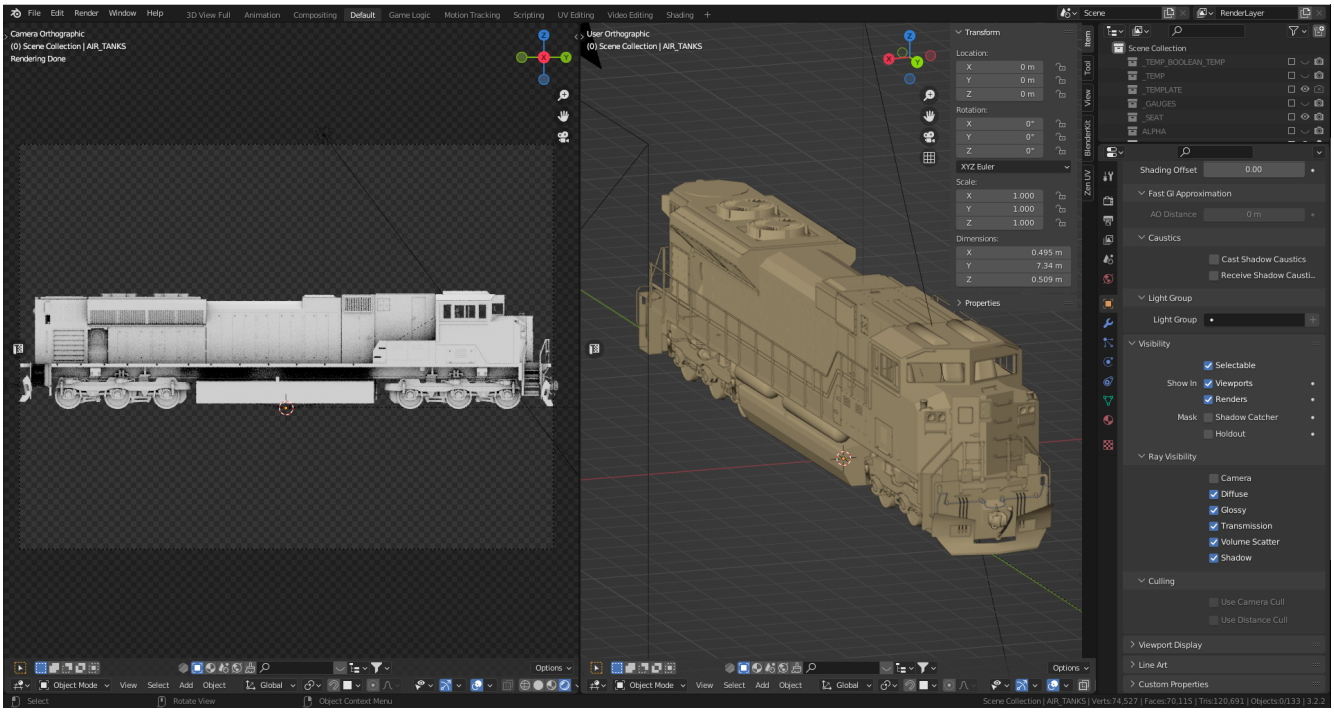
The Orthographic Scale should be adjusted for the camera so that the entire length of the locomotive body fills the camera view (for this locomotive, the scale is 22.300). The camera should be moved in the `Y` and `Z` axes to ensure that it is in frame. Then, the `Format Resolution Y = value` should be adjusted so that no empty space is rendered above and below the locomotive. When the entire locomotive is framed in the camera view, `F12` should be pressed and Blender will render the Ambient Occlusion into a new window.

If denoise is enabled, the rendering may appear to hang, but it will complete after a while. Making multiple copies of the camera and moving them to the sides, ends, top and bottom is recommended in order to render the Ambient Occlusion from the different projections. Keeping the `Orthographic Scale` the same on all the cameras will result in the render being at the same textural density. Additionally, unlike a perspective camera, the distance of the camera from the object does not change the size of the resultant render.

Once you have finished rendering the image, you can save it to your computer and import it into your 2D graphics program to use as a base for your texture. To make sure the texture has the same textural density, you should make multiple copies of the camera and move them to the sides, ends, top and bottom. Again, unlike with a perspective camera, the distance of the camera from the object will not affect the size of the render. However, the hand rails may obstruct the details on the body behind. To fix this, you have multiple options.



In the image above the `X` location of the camera is 10m to the side of the locomotive. The `Clip Start` distance has been changed from its default value of 0.1m to 8.7m. When rendered, the view no longer includes the handrails but displays the side of the body as the camera is drawing what it can see from 8.7m to 1000m, beyond the hand rail closest to the camera. This clipping results in the cab side, fuel tank and air tanks being cut off. The same outcome can be achieved by keeping the `Clip Start` and `End` at their default settings and moving the camera in the `X` direction, allowing for 'slices' of the scene similar to an MRI machine.



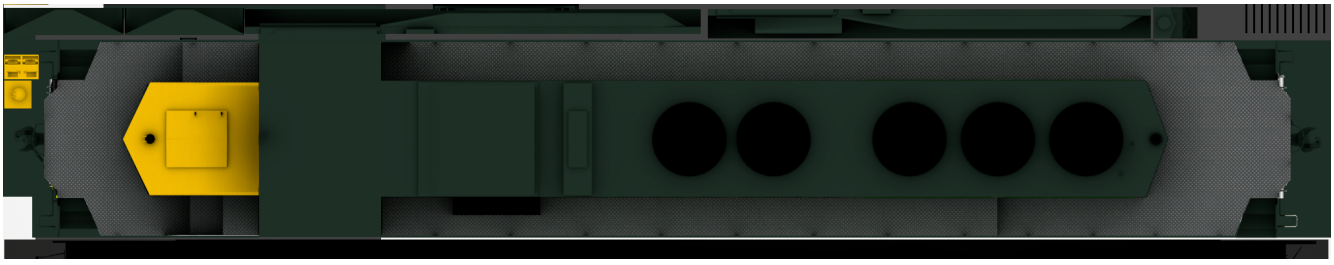
It is possible to make parts invisible to the render camera. In the image, the `Camera` checkbox in the Object Properties tab for the hand rails and air tank objects has been unchecked. This results in the entire side of the locomotive body being visible in the render view, but the handrails and air tanks not being rendered. These methods can be used to exclude objects in the foreground that are not required for the desired image.

The Ambient Occlusion can be rendered for all the different views and then mashed together in a 2D graphics program to

export as the `au_bhpb_sd70ace_4096` texture to use for unwrapping and in Open Rails.

This technique for Baking Ambient Occlusion is a good way to get the added shadow details on a texture but the method described here is not the only way to accomplish Ambient Occlusion shadows. It might not be the most effcient way either, but it seems to work.

It is important to be aware that any lights in the scene, including environment lighting, will impact the rendered Ambient Occlusion. Lights are not used when baking an Ambient Occlusion, so if the rendered Ambient Occlusion image is too dark, the environment brightness can be increased or area lights can be added, typically the length of the locomotive, placed on either side to minimize shadow casting. If the lighting is changed later on, this will impact subsequent renders and will not match earlier renders. It is essential to be aware of this.



More About Baking Ambient Oclusion

Ambient Occlusion Baking is mildy infuriating in Blender

When Baking clusion, Here are some tips:

1. Gather parts that will receiveAmbient Occlusion into a collection to make them easy to select.
2. Ensure the UV Maps for those parts do not wrap off the edge of the texture sheet.
3. Coplanar surfaces are a problem, including two sided, they bake black.
4. Set 'Render Engine' to Cycles.
5. Set Render Max Samples low eg 8, for testing, high for better quality , eg 128, 256 etc
6. If your model has secondary LOD's ensure they don't cast a shadow on your primary model, use 'Disable In Render' in the outliner

clusion Steps

To bake Ambient Occlusion in Blender, you can use the Bake tool in the Render tab of the Properties panel. Here's a step-by-step guide:

- Select the object or objects that you want to bake theAmbient Occlusion for.
- In the Properties panel, go to the Render tab and then click on the Bake tab.
- In the Bake tab, set the Bake Mode to clusion.
- Set the Samples value to the number of samples you want to use for theAmbient Occlusion bake. The higher the value, the more accurate theAmbient Occlusion will be, but the longer the bake will take.
- Set the Margin value to add a margin around the baked texture to prevent texture bleeding.
- Set the Space to either "Object" or "World" depending on whether you want theAmbient Occlusion to be baked in object space or world space.
- Check the "Clear" box to clear the image before baking.
- Check the "Normalized" box to normalize theAmbient Occlusion values, which can help with artifacts and banding.
- Click the "Bake" button to start theAmbient Occlusion bake.

TheAmbient Occlusion bake can take some time, depending on the complexity of the objects and the number of samples you are using. Once the bake is complete, theAmbient Occlusion map will be saved as an image in the UV/Image Editor. You can then use this image as a texture to apply theAmbient Occlusion effect to your object.

Introduction to Blender Scripting

Blender's true potential is unlocked through scripting. Blender scripting allows users to automate repetitive tasks, create custom tools, and extend the software's capabilities to suit their specific needs.

Blender provides a Python API (Application Programming Interface) that enables users to interact with its internal functionalities and control almost every aspect of the software programmatically. Whether you're a beginner or an experienced programmer, Blender scripting offers a flexible and efficient way to work with 3D objects, materials, animations, and much more.

With scripting, you can perform complex operations that would be tedious or time-consuming to achieve manually. Mastering Blender scripting opens up a many possibilities for efficiency in your workflow.

In this section, we will explore the fundamentals of Blender scripting, starting with some basics and gradually diving into more advanced concepts. We will cover essential topics such as accessing and manipulating objects, materials, as well as exploring scripting techniques for creating custom tools and add-ons.

Before diving into Blender scripting, it's recommended to have a basic understanding of the Python programming language. However, even if you're new to programming, this guide will try to provide you with the necessary foundations to get started.

Whether you're looking to automate repetitive tasks or wish to extend Blender's functionality, Blender scripting is a valuable skill that can enhance your workflow.

Actually Doing It

So we now consider diving in to Python. The Python that you find embedded in Blender is the same Python you find everywhere else, with one exception. That is, it includes the Python Blender Library (or API) for accessing the internal features of Blender. The `Library` is called "BPY". You will often see at the very top of Python code written for Blender the following line of code: `import bpy`. This is where you add all of the Blender functionality to your code.

To get started with Python though, you should probably set up a development environment for Python. This way you can begin to learn how to code in Python and practice and develop ideas without actually having to do it it inside Blender.

Getting started with Python is relatively easy and straightforward. Here's a step-by-step guide to help you begin your Python journey:

Install Python

Visit the official Python website (https://www.python.org) and download the latest version of Python for your operating system. Follow the installation instructions provided by the Python installer.

Set up a development environment

You'll need a code editor or an integrated development environment (IDE) to write and run your Python code. Some popular choices include Visual Studio Code, PyCharm, and Atom. Choose the one that suits your preferences and install it.

Learn the basics

Familiarize yourself with the fundamentals of Python programming. There are numerous online resources, tutorials, and books available to help you learn Python. Some recommended beginner-friendly resources include:

- "Python.org" - The official Python website provides a comprehensive tutorial (https://docs.python.org/3/tutorial/) to get you started.
- Codecademy's Python course (https://www.codecademy.com/learn/learn-python)
- "Automate the Boring Stuff with Python" by Al Sweigart (https://automatetheboringstuff.com/)
- "Python Crash Course" by Eric Matthes

Practice coding

As you learn the basics, it's essential to practice writing Python code. You can solve coding challenges on platforms like HackerRank (https://www.hackerrank.com/domains/tutorials/10-days-of-statistics) or LeetCode (https://leetcode.com/problemset/all/).

Explore Python libraries and frameworks

Python offers a vast ecosystem of libraries and frameworks that extend its capabilities. Depending on your interests and goals, you can explore libraries such as NumPy for numerical computations, Pandas for data manipulation, Matplotlib for data visualization, or Django for web development. Look for resources specific to the libraries or frameworks you wish to explore.

Join the Python community

Engaging with the Python community is a great way to learn, ask questions, and collaborate with others. Participate in online forums like the Python subreddit (https://www.reddit.com/r/Python/) or join Python-related communities on platforms like Discord or Slack.

Build projects

To solidify your Python skills, start working on small projects. Projects provide practical experience and help you apply what you've learned. Think of something you'd like to create—a web scraper, a calculator, a data analysis tool—and work towards building it using Python.

Remember, learning programming is an iterative process. Start small, be patient, and gradually tackle more complex concepts. With practice and persistence, you'll gain confidence and proficiency in Python.

Embarcadero has a Python IDE that is quite handy and much better than the one that comes with Python by default. You can get it here: https://www.embarcadero.com/free-tools/pyscripter/free-download

As mentioned, Blender has a built-in Python API and Python interpreter that allows you to control its functionality programmatically.

There is a Blender Add-On called SERPENS that will help you develop Blender HUI based add-ons from your code. Its a paid add-on, but its really neat.

Blender and ChatGPT

If you have not been living under a rock this lately, you have probably heard about ChatGPT. ChatGPT is, quite simply, the best artificial intelligence chatbot ever released to the general public. ChatGPT — which stands for "generative pre-trained transformer" — became an instant Internet hit. There are two ways to make use of the chatbot and these are:

1. Interactively using a chat session at https://chat.openai.com
2. Accessing the chatbot using code via the API using a private API KEY.

An API, or Application Programming Interface, is a set of rules and protocols that allows different software applications to communicate and interact with each other. It defines the methods, data formats, and conventions that applications should follow to request and exchange information. APIs play a crucial role in enabling software integration, fostering collaboration between different systems, and promoting the development of interconnected applications.

By integrating ChatGPT with Blender Python scripts, you can augment your knowledge of Python without the need for being a whiz-bang Python coder. Many of the scripts that will be shown in this section were actually supplied by ChatGPT. While the responses from ChatGPT sometimes contain coding errors, ChatGPT also can be told about errors and can generate updates to the script response until it finally works. The responses can also be used as a starting point for a more detailed Python script.

ChatGPT, based on the GPT-3.5 architecture, is designed to generate human-like text based on a given prompt. It has been trained on a vast amount of diverse and high-quality data, making it capable of understanding and responding to a wide

range of questions, instructions, and conversational prompts. By leveraging ChatGPT's language generation capabilities, you can solve specific problems or get answers to particular questions related to your Blender projects.

Incorporating ChatGPT into your scripts

Here are the steps to integrate ChatGPT with your Blender Python scripts:

Installation and Setup

Ensure you have Blender installed on your machine, along with a compatible version of Python. You can download Blender from the official website (blender.org) and install the required Python packages using pip or a package manager like Anaconda. (The authors preference is to avoid Anaconda)

Importing the ChatGPT Library

Incorporate the ChatGPT library into your Python environment. OpenAI provides an API that allows developers to interact with ChatGPT programmatically. You'll need to obtain an API key to access the OPENAI ChatGPT service.

Establishing a Connection

Use the API key to establish a connection with the ChatGPT service. This connection will enable you to send prompts and receive responses from the language model.

Script Integration

Within your Blender Python scripts, incorporate the necessary code to communicate with ChatGPT. You can prompt the model with questions, provide instructions, or engage in a dialogue with the model. Retrieve the generated text from ChatGPT and use it to influence the behavior of your Blender scene.

Remember to handle the limitations of ChatGPT, such as potential biases or generating incorrect information in response to questions. Preprocessing user inputs, providing context, and implementing appropriate error handling can help mitigate these issues and provide a better user experience.

To summarize, integrating ChatGPT with Blender Python scripts opens up a whole new aspect of Blender, especially for those that need a little extra help and don't have a human mentor handy.

There is an add-on available for Blender that integrates ChatGPT with Blender using the ChatGPT API. It's currently a $5.00 purchase… but feedback seems positive. The addon has the capability to either provide guidance for accomplishing a task or try to carry out the task independently. https://ryanaddons.lemonsqueezy.com/checkout/buy/70eb3e1e-8500-40ff-a652-beef1a511106 The author has not used it.

Getting an API Key from OPENAI

Blender Python code developers using ChatGPT are required to use an API key to utilize ChatGPT API from within Python **code**. They can generate this key by logging in to OpenAI's website and selecting "View API Keys". Here is the step-by-step guide.

You do not have to use the API or use Python code to access ChatGPT. You can always use the interactive chat by using the https://chat.openai.com web interface to ask your questions and have conversations.

To use ChatGPT through the API though, you must create a free account and generate the API key(s). Fortunately, it is pretty straightforward.

You can request an API KEY by signing up here: https://beta.openai.com/signup. You can use your Google or Microsoft account to sign up if you don't want to create a unique UserName & Password combination for OPENAI. You may need a valid mobile number to verify your account.

Once you have an account, you can chose the "View API Keys" option in your profile to "Create a new secret key", which is your API KEY.

Remember, the key created has access to both GPT-4 and ChatGPT models. You don't need separate API keys. ChatGPT-3.5 Turbo is the current ChatGPT default model.

You MUST save the key that shows up on the screen when you create a new key. It will never be shown to you again and if you lose it, you will need to make a new one.

API costs

API Queries are not free. Please note that you will be charged based on your monthly usage. The cost structure is as follows: 750 words (aka 1000 tokens). Below is a table representing the cost per 1000 tokens:

Model	Cost (Input)
GPT-3.5 Turbo (4k context)	$0.0015 (13x cheaper than GPT-3 Davinci)
GPT-3.5 Turbo (16k context)	$0.003

For example, if you use 10,000 tokens per day of GPT-3 Turbo for 20 days a month, you will be charged 400 cents per month, i.e. 0.002 * 10 * 20.

If you plan to use it regularly, you need to add your credit card information here https://beta.openai.com/account/billing/payment-methods

You can also set up a usage limit if you have a paid set up, say, $10/month at https://beta.openai.com/account/billing/limits

The author is currently not well versed on using the API, so can only provide limited feedback on the experience.

Running a Blender Python Script Inside Blender

To run a Blender script, follow these steps:

- Open Blender: Launch the Blender application on your computer. It's probably best to you have the latest version of Blender installed to have access to the most up-to-date features and improvements. As mentioned in other sections of this document, the Long Term Support, (LTS VERSION) of Blender is probably the best choice.
- Open the Text Editor: Once Blender is open, locate the Text Editor. You can find it by selecting the "Scripting" workspace from the top bar or by navigating to the "Editor Type" dropdown at the top left of any editor area and choosing "Text Editor".
- Create a New Text Block: In the Text Editor, click on the "New" button to create a new text block. This will provide you with a clean space to write or paste your script.
- Write or Import your Script: You have a few options here:
    1. Write the script: If you're familiar with Python scripting, you can directly write your script in the Text Editor. Ensure that your script follows the correct syntax and indentation rules of the Python language.
    2. Import an existing script: If you have a pre-existing script saved as a file on your computer, you can open it in the Text Editor by clicking on the "Open" button and navigating to the script's location.
    3. COPY/PASTE the script code from somewhere. For example, clicking `copy code` from a ChatGPT conversation and then pasting it into the code editor window of Blender.
- Run the Script: Once you have your script ready in the Text Editor, you can run it by clicking on the "Run Script" button or by using the keyboard shortcut `Alt` + `P`. Blender will execute the script, and its functionality will take effect within the Blender environment.
- Observe the Results: Depending on the script you've written or imported, you will see the desired outcome within the

Blender interface. This could involve creating or modifying objects, materials, animations, or any other aspect of the 3D scene.

It's important to ensure that your script interacts with Blender's API correctly and that it doesn't contain any errors that might cause unexpected behavior or crashes. Always double-check your code for syntax errors or logic mistakes before running it.

By following these steps, you can easily run a Blender script and leverage the power of scripting to automate tasks, create custom tools, or extend Blender's functionality to suit your specific needs.

A Collection of Scripts

The following scripts are tested examples of very simple scripts to help get you started with scripting. They are only a few lines each so you can take the time to digest them and learn how they work.

Fixing Material Problems

Let's say you have completed some steps with your project that involved using multiple copies of the same object to be placed around your scene. Let's also say that these parts actually came from another model so you ended up using a COPY/PASTE operation to get them into your scene. With this as our background setup, what is a possible issue that we might encounter? Do you know?

Well, one thing that happens when you COPY/PASTE, is that your parts get renamed to be unique, which is normal for a DUPLICATE operation as well, but your material reference gets replicated the same way, in other words, it gets renamed. So if your project has been using a Material named "MAIN"… you will likely end up with a material named "MAIN01" and so on. This will happen with any objects that were duplicated using COPY/PASTE instead of a DUPLICATE operation.

Duplicate objects are created by using `SHIFT` + `D` and the process will not try to create a duplicated unique material reference.

The goal of the script then So, if you can to fix this issue so you don't end up with 20 materials that are all actually 100% identical, you can run the following script:

```
import bpy

# Get the currently active material or create a new one
material_name = "main"  # Replace with your desired material name
material = bpy.data.materials.get(material_name)
if material is None:
    material = bpy.data.materials.new(name=material_name)

# Assign the material to each selected object
selected_objects = bpy.context.selected_objects
for obj in selected_objects:
    if obj.type == 'MESH':
        if obj.data.materials:
            # If the object already has materials, replace the first one
            obj.data.materials[0] = material
        else:
            # If the object has no materials, assign the new material
            obj.data.materials.append(material)
```

Now selected materials will just have the one assignment of your choosing in slot 0. It doesn't completely eliminate the issue of material mismanagement, but its a start.

Making copies

While making a copy of an object is easy enough with `SHIFT` + `D`, sometimes you want to make a copy and offset it by a specific amount. This next ChatGPT generated script will do that.

To use this script, follow these steps:

- Open Blender and make sure the object you want to duplicate is selected.
- Open the "Scripting" layout.
- Create a new text block in the text editor and paste the script.
- Run the script by clicking the "Run Script" button or pressing Alt+P.
- The selected object will be duplicated, and the duplicate will be offset by 3 meters in the -Y axis.

Make sure you have the desired unit settings in Blender before running the script to ensure the correct offset distance. Using the Metric units is assumed since it is the default setting.

```
import bpy

# Get the selected object
selected_obj = bpy.context.object

# Duplicate the selected object
duplicated_obj = selected_obj.copy()
bpy.context.collection.objects.link(duplicated_obj)

# Offset the duplicated object
duplicated_obj.location.y -= 3.0
```

Make Selected Objects Become ASSETS

The Asset Browser is a Blender feature introduced in version 3.0, therefore, this script will not work on older versions of Blender as the feature is not available.

To use this script, follow these steps:

- Open Blender and switch to the scripting layout.
- Create or open a new Blender file.
- Select the objects that you want to mark as assets.
- Open the "Text Editor" panel and create a new text block.
- Copy and paste the script into the text block.
- Click the "Run Script" button or press "Alt+P" to execute the script.

The selected objects will now be marked as assets.

Please note that this script assumes you have the necessary objects selected in Blender before running it. Also, make sure to save your work before running the script, as it will modify the objects in your scene.

```
import bpy

def mark_objects_as_asset():
    # Get the currently selected objects
    selected_objects = bpy.context.selected_objects

    # Iterate over each selected object
    for obj in selected_objects:
        # Mark the object as an asset
        obj.asset_mark()

# Call the function to mark selected objects as assets
mark_objects_as_asset()
```

Automating the Boring Stuff with Python

"Hey, isn't that the title of a book from Al Sweigert?"

"Why yes, it is and it is the inspiration for the following Blender Python code"

The book "Automate the Boring Stuff with Python" is available from NoStarch Press or Amazon but it is also free to read. https://automatetheboringstuff.com/#toc I highly recommend it.

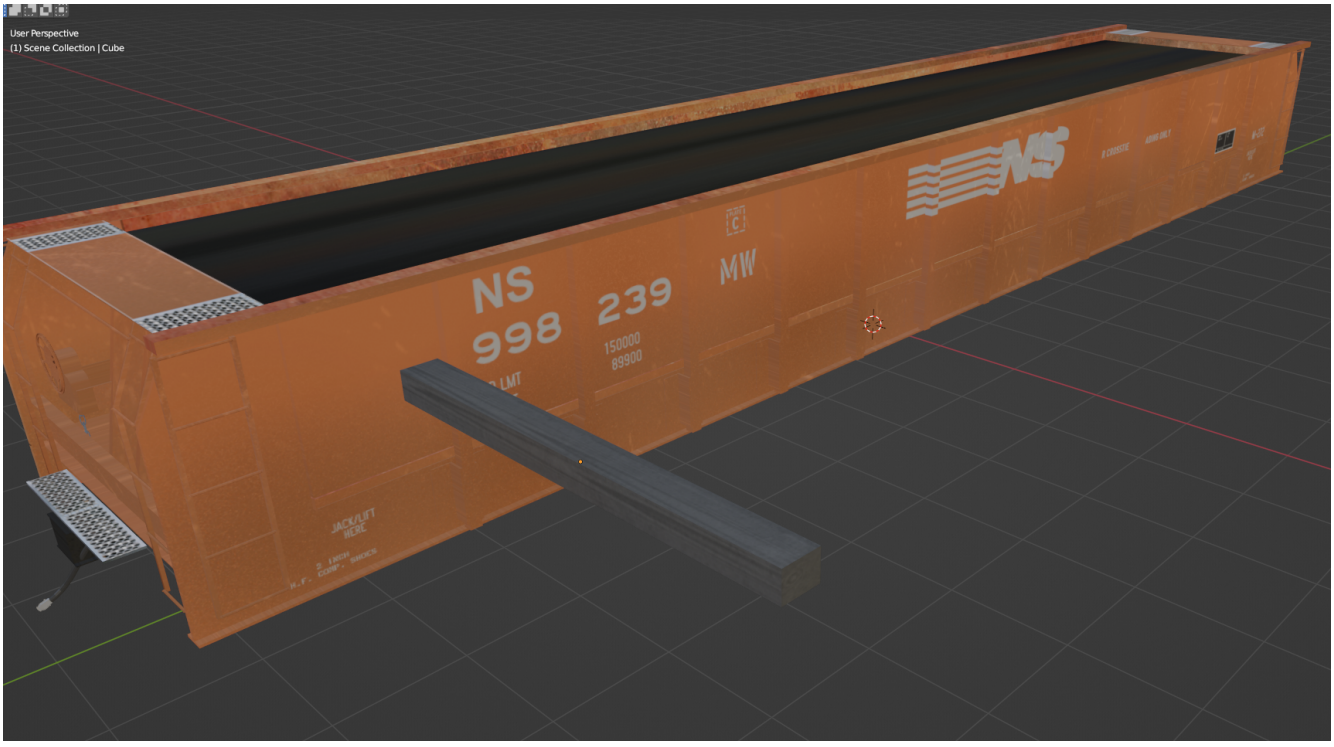
So what are we going to do?

When working on a Blender model, you often encounter tasks that require repetitive steps. These tasks can become tedious, especially when you need to repeat them a large number of times, such as 56 times. This is how it all began. Now, let's discuss what led up to this.

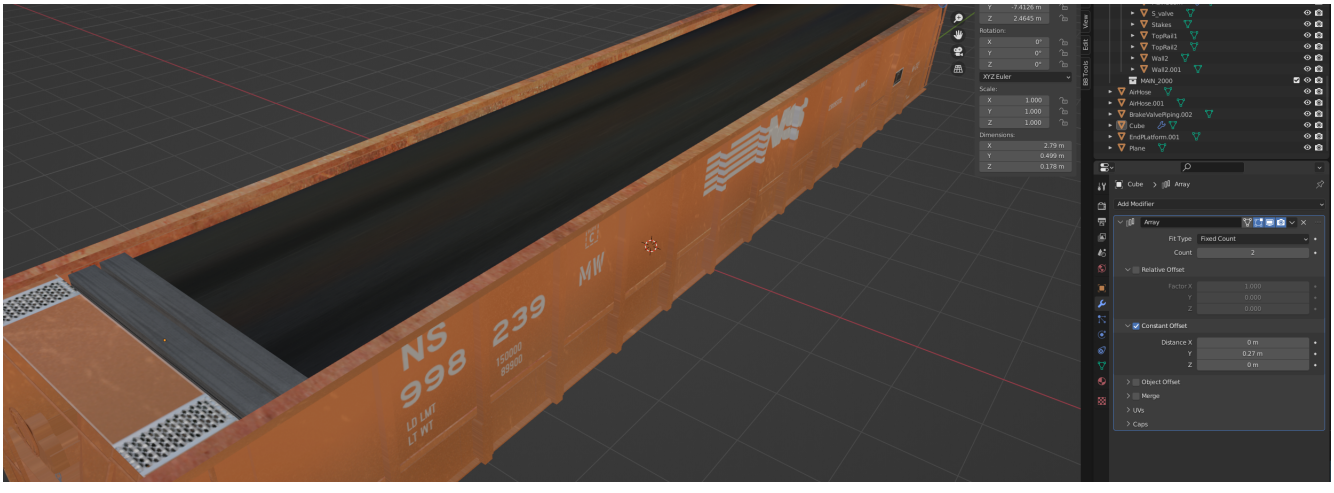
While working on the crosstie load for my Norfolk Southern Crosstie Gondola, it was realized that we only needed to create a single crosstie, apply a texture to it, and then use an ARRAY MODIFIER to duplicate it until the objects filled the gondola load area. Below is the final result.

But how did I get there?

The initial crosstie shape is created using a cube primitive. The dimensions used are 7" x 9" x 9' or in Blender → Metric (Z = 0.1778m Y = .2286m X = 2.7492m). The next steps are to Apply Rotation and Scale and then to texture it.

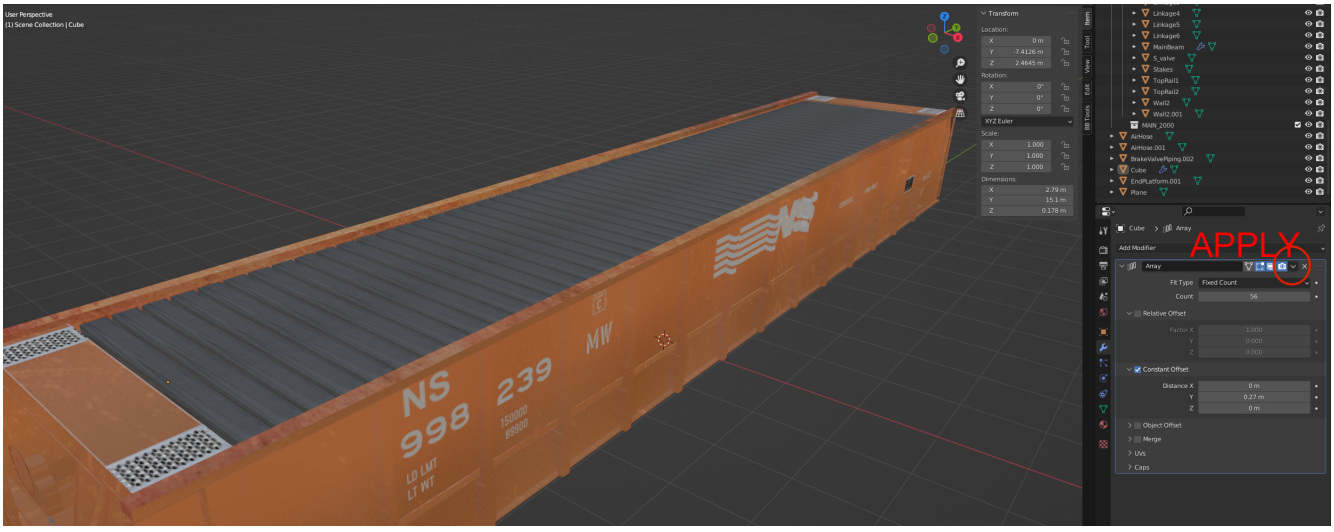


So now that we have a crosstie built and textured (Not covered here, sorry), it needs to be replicated to fill the gondola load area. This is made easier by using the **ARRAY MODIFIER** (Under Wrench Icon)

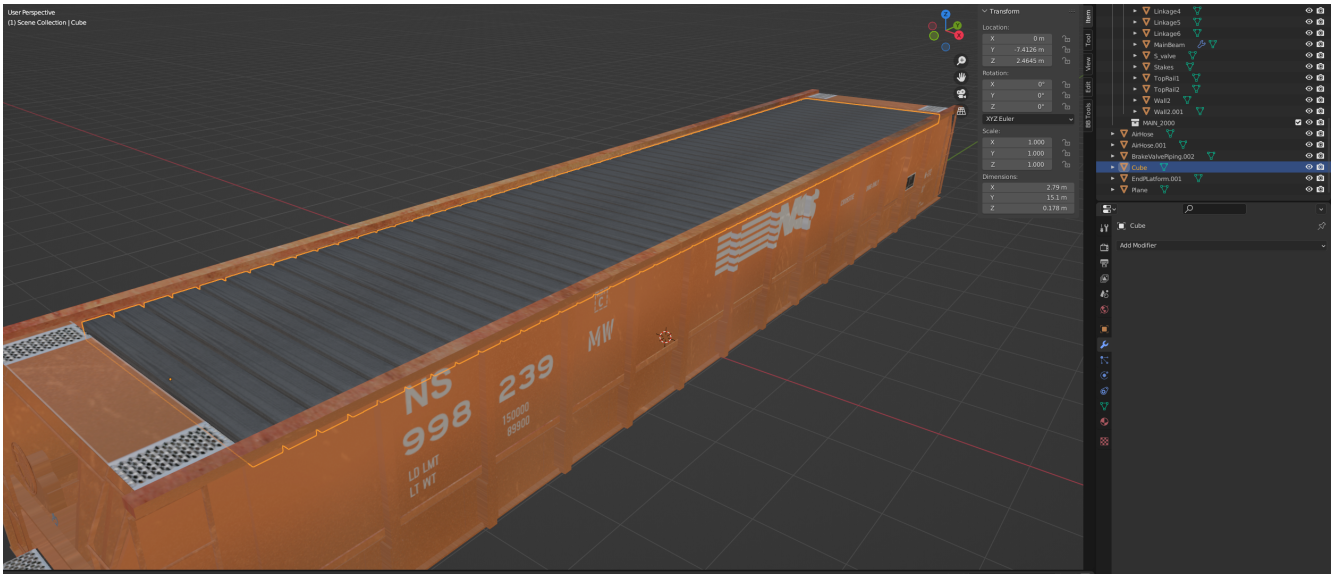


The settings used are: **CONSTANT OFFSET** (Not default) Y Axis OFFSET VALUE: 0.027 (or so).

Uncheck Relative Offset. Zero Out the X Field value. But we need to FILL the area… so the initial Count field is "2" and it needs to be increased to fit the area (in this scene, it needed 56 arrayed items to make it fit)



Great! We have a FILLED gondola with an array of crossties and we still only needed to make one crosstie to start with. BUT! They are all way too uniform and look unnatural. The problem is, these objects (as they are now) will all continue to imitate any changes we make to the original cube since the array modifier is still **active**. So now we need to **APPLY** the **ARRAY** modifier while in OBJECT mode.



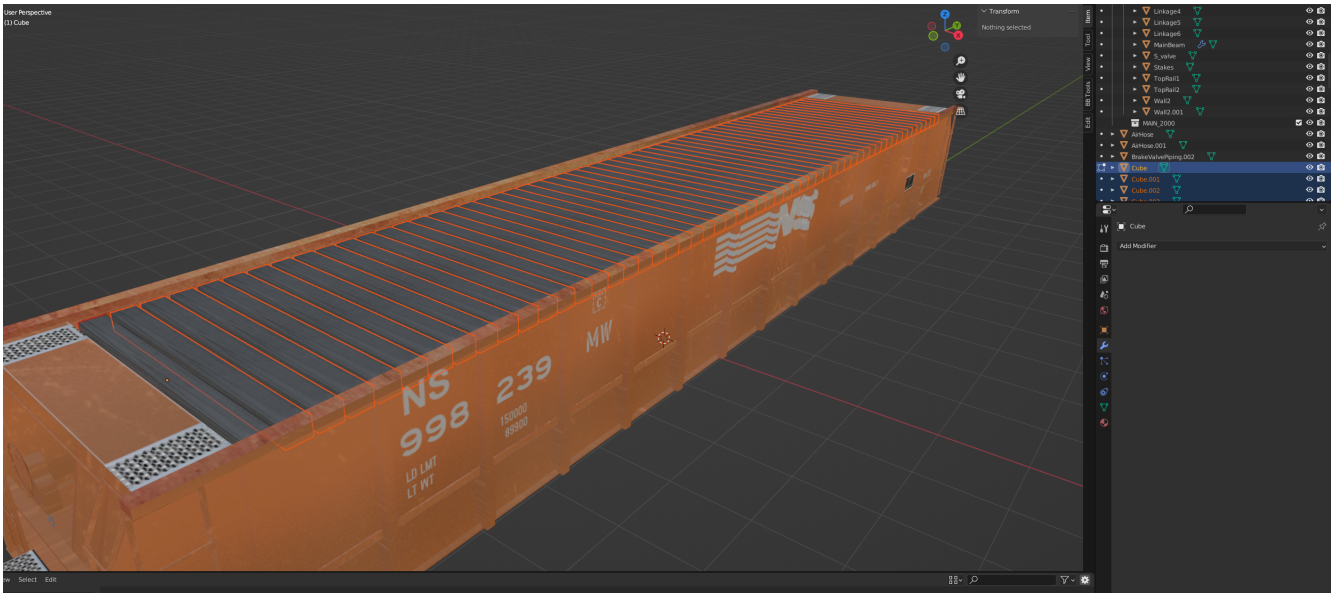
Our next problem is that these arrayed objects are all still basically the same grouped object. All of these objects will also share the SAME origin point as well. So the next step to do we to break them apart.

- Select the crosstie object (cube)
- Press `TAB` for EDIT MODE
- Press `P` - to Separate. Now choose: "By loose parts" from the menu.

Now all of the individual items are **created**, having been separated from the original. (Check your collection list as in this case we now have 56 CUBES that make up the crossties)

All the crossties share the same origin location, making individual adjustments to them is currently rather challenging. Therefore, it becomes necessary to assign each crosstie its own origin point based on its specific geometry. This will allow for easier customization and modifications to each crosstie.

Normally in this case, we would need to make 56 individual edits! Oh the DRUDGE! So, let's make use of Blender's scripting TAB.

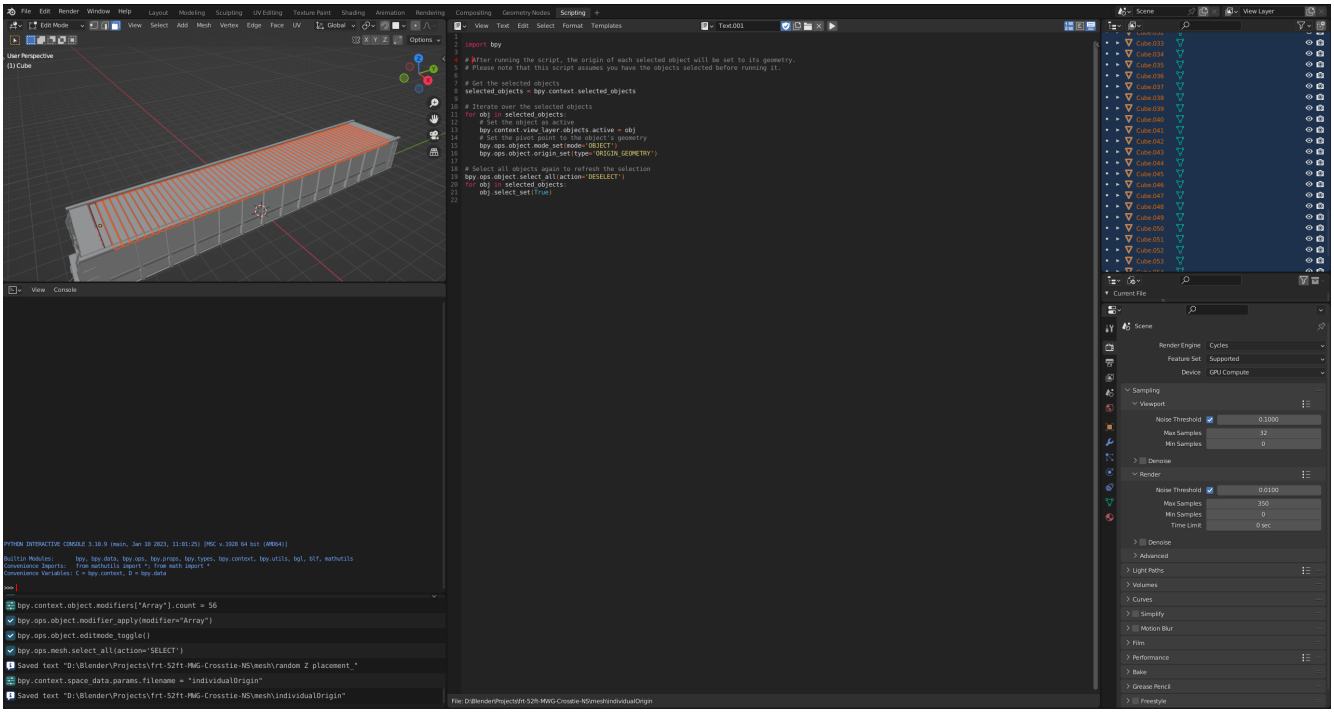


Make sure all of your objects (crossties) are currently selected. (You might want to just select them all from the Collection list.)

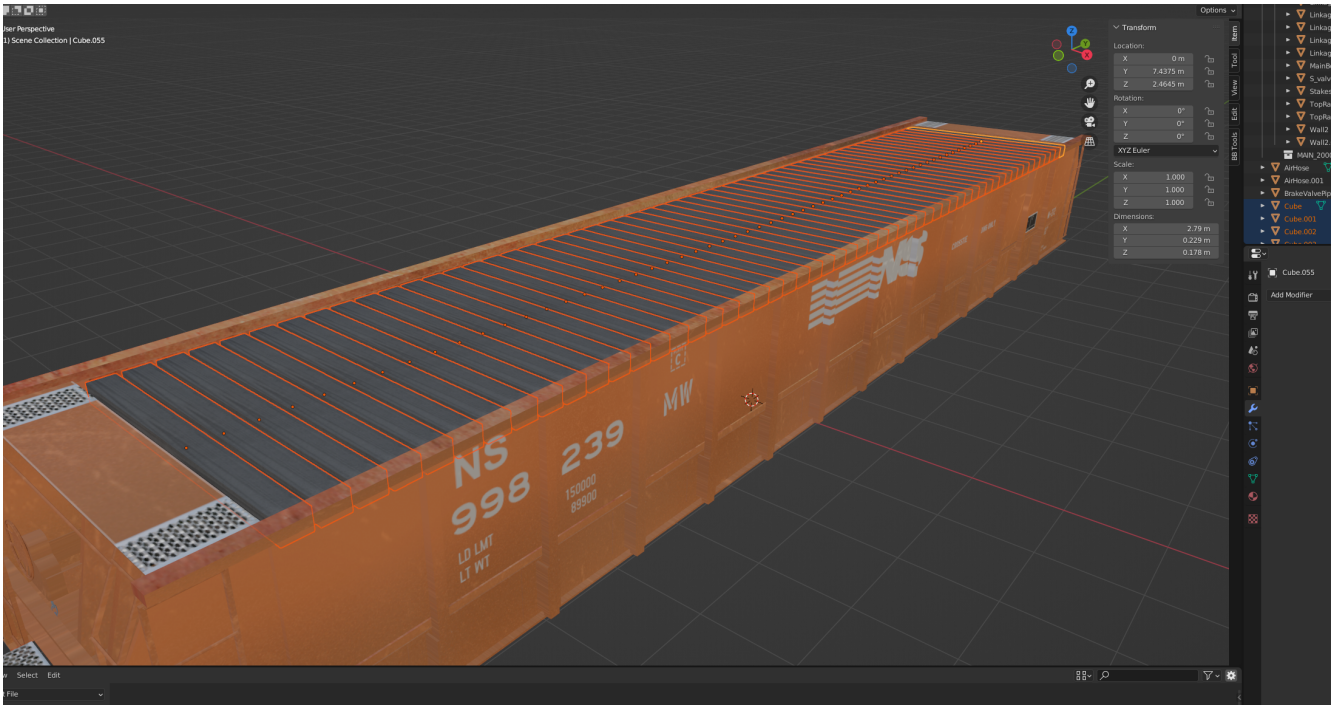
```
# After running the script, the origin of each selected object will be set
# to its geometry. Please note that this script assumes you have the
# objects selected before running it.
import bpy

# Get the selected objects
selected_objects = bpy.context.selected_objects

# Iterate over the selected objects
for obj in selected_objects:
    # Set the object as active
    bpy.context.view_layer.objects.active = obj
    # Set the pivot point to the object's geometry
    bpy.ops.object.mode_set(mode='OBJECT')
    bpy.ops.object.origin_set(type='ORIGIN_GEOMETRY')

# Select all objects again to refresh the selection
bpy.ops.object.select_all(action='DESELECT')
for obj in selected_objects:
    obj.select_set(True)
```

Open the scripting tab in Blender, follow these steps:

1. In Blender and navigate to the top header of the application.
2. Locate and click on the "Scripting" tab.
3. Within the "Scripting" tab, go to the "Text" tab.
4. Choose "Create New" to create a new file.
5. Enter or paste the provided code into the Blender Python Editor window.



Press the RUN icon to run the code. (You will see that it did something in the windows on the left). You can switch back to LAYOUT from the Scripting tab for a better view.

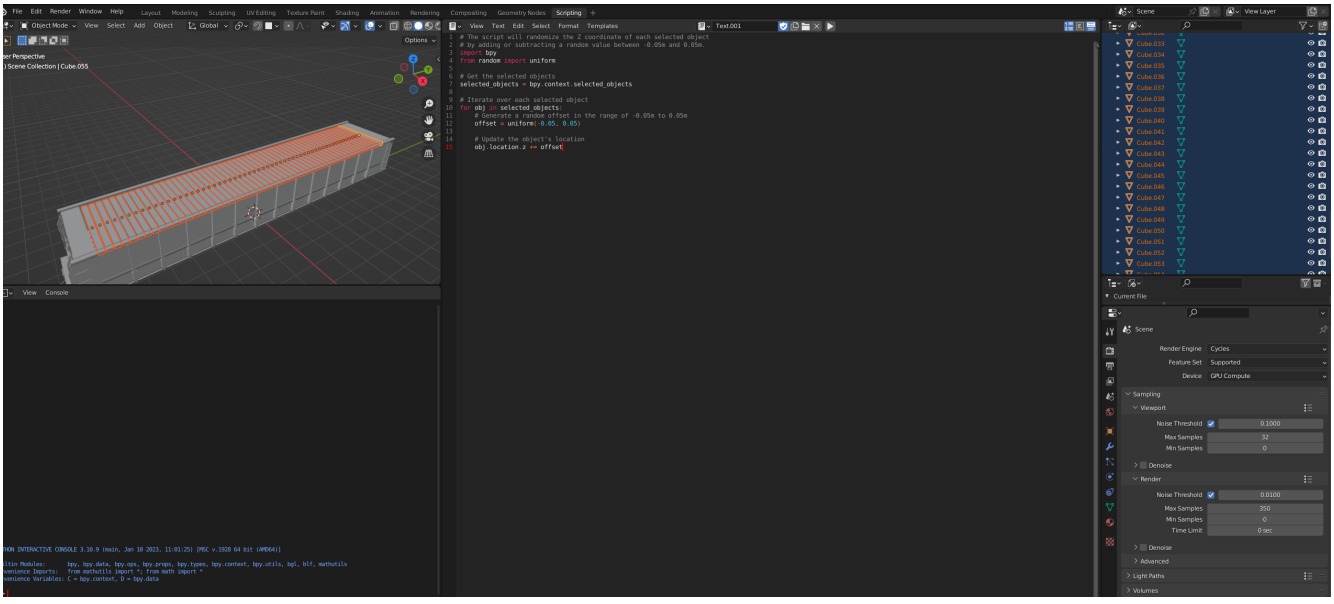


Now all of the objects have their own origin based on their specific geometry. (See all the orange origin dots on each crosstie?)

The next step is to randomize the Z axis location of these objects (crossties) so they look less uniform.

In the Blender Python Editor (Scripting Tab), create a NEW text document and enter/paste-in the following code:

```
import bpy
from random import uniform

# Get the selected objects
selected_objects = bpy.context.selected_objects

# Iterate over each selected object
for obj in selected_objects:
    # Generate a random offset in the range of -0.05m to 0.05m
    offset = uniform(-0.05, 0.05)

    # Update the object's location
    obj.location.z += offset
```



Press the RUN icon to run the code.

You will end up with the following...

and finally…

To summarize:

We encountered a situation where the crosstie load needed to be placed non-uniformly, but it was realized that creating just a single textured cube object and performing minimal additional steps, such as replication with offsets would suffice. Furthermore, as a result of this process, you now have two scripts in your collection that can assist with future object manipulation in Blender.

Creating Rolling Stock and Locomotives

In order to ensure the precise functioning of locomotives and other rolling stock, it is crucial to acquire as much authentic data as possible and comprehend the expected behavior of the locomotive or rolling stock. These bits of collected data will be used to flesh out the ENG and WAG files that accompany the 3D Model and textures that make up a particular content item that can be added to Open Rails.

You would need to gather enough prototypical data to accurately complete the physics descriptions in the WAG or ENG file.

Please note that there have been many improvements and additions to the simulator that go well beyond the capabilities an options of Microsoft Train Simulator. Many of the options listed in this section contain values and parameters that are unique to Open Rails. It is assumed that the reader is NOT trying to retain compatibility with legacy Microsoft Train Simulator and doing do is beyond the scope of this section. If you wish to retain Microsoft Train Simulator compatibility, please refer to the technical documents that come with Microsoft Train Simulator.

The process of creating content and config files is often described as a complicated mess, and that description is not entirely inaccurate. However, there is a side project underway to develop a set of replacement files for MSTS 3D models, config files, and textures that are free to use and do not violate anyone's copyrights. These replacement files will serve as examples of the standards for creating 3D models, textures, and config files for Open Rails, which can be shared with everyone.

Open Rails Units of Measure

Source: http://www.coalstonewcastle.com.au/physics/or-parameter-uom/

Aim - this section describes the standard units of measure (UoM) available in Open Rails Train Simulator. These are the UoM used to input values from the ENG or WAG files.

Note:

- The default UoM shown in the "Default" column is the UoM value that OR assumes if no UoM value is included in the parameter statement. **(However it is always good practice to include the relevant UoM)**
- The "Alternative UoMs" column shows alternate UoM currently supported by OR.
- The UoM should directly follow the numerical value in the parameter with any spacing, otherwise it will not be correctly interpreted.

Unit of Measure	Description	Default UoM	Alternative UoMs	Typical Examples
Mass	Describes the weight of an object	**kg**	**lb**, **t** (metric tonne), **t-uk** (imperial ton), **t-us** (us ton)	Mass (106t-us)
Distance	Describes the length or distance between two points or locations.	**m**	**cm**, **mm**, **km**, **ft**, **in**	WheelRadius (21.5in)
Area	Describes the area of an object.	**m^2**	**ft^2**	ORTSSuperheatArea (295ft^2)
Volume (of a space)	Describes the volume of an object.	**m^3**	**ft^3**, **in^3**	BoilerVolume (340ft^3)
Volume (fluid quantity)	Describes the fluid quantity contained by an object.	**l**	**g-uk** (gallons - uk), **g-us** (gals - us)	MaxDieselLevel (800g-uk)
Time	Describes the elapsed time between two events.	**s**	**m**, **h**	
Current	Describes the current flow in a circuit.	**A**		MaxCurrent (2700A)
Voltage	Describes the voltage of a circuit.	**V**	**kV**	
Speed	Describes the speed of an object.	**m/s**	**mph**, **km/h**	Sanding (20mph)
Mass Rate of Change	Describes the rate of change in mass for an object.	**lb/h**	**kg/h**	
Frequency	Describes the frequency of movement of an object.	**hz**	**rpm**, **rps**	

Unit of Measure	Description	Default UoM	Alternative UoMs	Typical Examples
Force	Describes the force applied to an object.	**N**	**kN**, **lbf**	MaxForce (227kN)
Power	Describes the power of an object.	**W**	**kW**, **hp**	MaxPower (2650hp)
Pressure	Describes the pressure of an object.	**psi**	**bar**, **kpa**, **inhg**	MaxBoilerPressure (250psi)
Pressure rate of change	Describes the rate of change in pressure of an object.	**psi/s**	**bar/s**, **kpa/s**, **inhg/s**	TrainPipeLeakRate (0.01inhg/s)
Energy Density	Describes the energy density of an object.	**kj/kg**	**btu/lb**	ORTSFuelCalorific (13700btu/lb)
Temperature	Describes the temperature of an object.	**degc**	**degf**	
Davis B Coefficient	Describes the UoM for the Davis B value.	**ns/m**	**lbf/mph**	
Davis C Coefficient	Describes the UoM for the Davis C value.	**Nm/s^2**	**lbf/mph^2**	
Stiffness	Describes the stiffness.	**n/m**		Stiffness (1e6N/m 2e6N/m)
Rotational Inertia	Describes the inertia due to the rotation of an object.	**kgm^2**		
Angle	Describes the angle of an object.	**rad**	deg	ORTSMaximumWheelFlangeAngle (75deg)

The contents of an ENG or WAG file use an S-expression type of notation for grouping of related items. Token definitions are delimited by opening (and closing) parenthesis. All strings are quoted using the double quote character (") and are UTF-8 encoded. Tokens can have zero or more attributes. Human readability is a design goal.

With Open Rails it is very common that things like coupler and brake settings are consistent between many different pieces of rolling stock. For this reason, it is common to see content releases that share these common settings in a specific folder on the hard drive and make call-outs to the common settings using the Open Rails `INCLUDE` token in the `ENG` or `WAG` file.

Using *INCLUDE*

The INCLUDE token is new to Open Rails and did not exist in Microsoft Train Simulator. Using INCLUDE looks like the following example, where this NAVS content `WAG` FILE makes a call-out to FreightAnims for car body and numbers and then coupler and then the remainder of the specific car details:

```
SIMISA@@@@@@@@@@JINX0D0t______

Wagon ( NAVX_105007_MT

    ORTSFreightAnims (
        WagonEmptyWeight ( 64400lb )
        FreightAnimStatic (
            Shape( ../NAVS_COMMON/Flatcar-GS-Type-53-6/Flatcar-GS-Type-53-6_BULK_Carbody-4.s ) ①
            Offset( 0, 0, 0 )
            FreightWeight( 0lb )
        )
            FreightAnimStatic (
            Shape( ../NAVS_COMMON/Flatcar-GS-Type-53-6/Decals/Flatcar-GS-Type-53-6_Decals_08.s ) ②
            Offset( 0, 0, 0 )
            FreightWeight( 0lb )
        )
    )

    Include( "../NAVS_COMMON/INCLUDE/AAR_Type-E.inc" ) ③
    Include( "../NAVS_COMMON/Flatcar-GS-Type-53-6/Flatcar-GS-Type-53-6_S2_Roller_Bulkhead_MT.inc" ) ④
```

① Get the 3D MODEL of this vehicle

② Get the additional added modeled items, in this case, the DECALS

③ Include the common Coupler Type

④ Include the physics details that define this particular vehicle

While you **can** continue to place all of your physics data in a single `WAG` or `ENG` file, you might find that this method of including other files helps out when you create many variants of the same basic model. It remains totally optional to make use of this technique.

Basic ENG and WAG file Details

The goal of this section is to assist the reader when creating new vehicles or when upgrading their old MSTS ENG and WAG files to current ORTS standards.

The list of Open Rails ENG/WAG file parameters is very likely to be a constantly moving target as new tokens/parameters are being added regularly.

Should there be a ENG/WAG standard layout?

So far, we have not been required to follow a rigid standard for ENG and WAG file layout and there is no indication that this will ever change. While this offers freedom of choice to the developer of content, it has created truly horrific examples of how the s-expression format does not lend itself to good results. Much uploaded content has suffered from the extra or missing parenthesis due opening and closing pairs not being very visible when spanning many lines of text.

There **should** also be an attempt by the reader to adopt a standardized or consistent way they create their own ENG/WAG files. The goal of making these files more easily readable and validated should be attempted.

One recommendation made by Peter Newell is to separate the contents of these files into sections with comment headers, for example:

```
Comment ( ****************************** General Information ******************************
          Included in this section - Type, Shape, Size, Mass, etc
          ********************************************************************************* )
```

It's not a bad idea, though you actually seldom see it being used.

There are essentially 2 sets of parameter types in Open Rails. These are BASIC and ADVANCED settings. For the most part, BASIC settings are compatible with legacy MSTS while ADVANCED settings are specifically used only in Open Rails.

For example:

Couplers Basic

- CouplingHasRigidConnection
- Coupling - Automatic Bar Chain
- Type
- Spring
- Break
- r0

Couplers Advanced

- ORTSTensionStiffness
- ORTSTensionR0
- ORTSTensionSlack
- ORTSCompressionStiffness
- ORTSCompressionR0
- ORTSCompressionSlack
- ORTSBreak

Universal Settings

These settings are found in nearly every type of rolling stock or locomotive content and while some are optional, most are really not. Many of the new ORTS tokens are related to Derailment Coefficient determination.

Token	MSTS	ORTS	Parameters
Type()	x	x	Steam Diesel Electric Freight Carriage Tender
Wagon()	x	x	Section contains all related wagon() tokens
WagonShape()	x	x	Defines the file path to locate the the shape ".S" file
Size()	x	x	3 Dimensions: X Y Z, in meters by default unless a Unit is supplied
WheelRadius()	x	x	Wheel Radius in meters unless a Unit is supplied. Example: "33in"
ORTSLengthCouplerFace ()		x	DC - Length between coupler faces (Is this the same a 'length over strikers' + about 24"?)

Token	MSTS	ORTS	Parameters
ORTSLengthCarBody ()		x	DC - Actual length of car
ORTSLengthBogieCentre ()		x	DC - Actual length of distance between center of bogies
ORTSRigidWheelBase ()		x	DC - The distance between immovable axles
ORTSNumberBogies ()		x	DC - Number of Bogies on the car
ORTSNumberAxles ()		x	DC = Number of total axles for all bogies
ORTSWheelFlangeLength ()		x	DC - Wheel Flange length
ORTSMaximumWheelFlange Angle (72deg)		x	DC - (Degrees or Radians) default is radians
include		x	Defines the file and path if external file to inserted into the current file

DC indicated that the token is related to the new Derailment Coefficient

See the [Appendix A] for a more complete list of tokens/parameters. Note: this list is still incomplete.

Example WAG file with INCLUDE

Probably some of the BEST examples of how to make use of the Open Rails ENG/WAG design goals is Erick Cantu with what he has done with the NAVS rolling stock and engines. Erick heavily makes use of the `INCLUDE` token and new `Freight Anim` options. I would recommend downloading and looking at one of Erick's SDK files for a good background on his techniques.

An `INCLUDE` related Excerpt from Elvas Tower Forum is below, where it is pointed out that you can redefine existing sections of a WAG/ENG file with replacement entries that come after the original definitions.

Let's start with the original WVOGrain.wag:

```
SIMISA@@@@@@@@@@JINX0D0t______

Wagon ( WVOGrain
        comment( WVO Covered hopper, hi-hip version)
        comment( Physics by Chris Lee)
        Type ( Freight )
        WagonShape ( WVOGrain.s )
        Size ( 3.243m 4.728m 17.503m)
        comment( 23.776t empty, 106.747t full )
        Mass ( 73.78t )
        WheelRadius ( 36in/2 )
        InertiaTensor ( Box (3m 3.6m 13.5m) )
        Coupling (
                Type ( Automatic )
                Spring (
                        Stiffness ( 1e6N/m 5e6N/m )
```

```
                Damping ( 1e6N/m 1e6N/m )
                Break ( 5.1e8N 5.1e8N )
                r0 ( 20cm 30cm )
        )
        comment( CouplingHasRigidConnection () )
        Velocity ( 0.1m/s )
)
Buffers (
        Spring (
                Stiffness ( 1e6N/m 5e6N/m )
                Damping ( 1e6N/m/s 1e6N/m/s )
                r0 ( 0m 1e9 )
        )
        Centre ( 0.5 )
        Radius ( 1 )
        Angle ( 0.5deg )
)

Adheasion ( 0.2 0.4 2 0 )
DerailRailHeight ( 4cm )
DerailRailForce ( 2.5N/kg*23t )
DerailBufferForce ( 400kN )
NumWheels (     8 )
Friction (
        871N/m/s                0               1mph            3.26N/m/s               1.8
        5.1N/rad/s              1               -1rad/s         0               1
)
Lights  (       2

        Light   (
                comment( Rear red light flashing dim )
                Type            ( 0 )
                Conditions      (
                        Headlight ( 2 )
                        Unit ( 3 )
                )
                FadeIn  ( 0.5 )
                FadeOut ( 0.5 )
                Cycle           ( 0 )
                States  (       2
                        State   (
                                Duration ( 0.35 )
                                LightColour ( 80ff0000 )
                                Position ( 0.0 1.2 -8.955 )
                                Azimuth ( -180 -180 -180 )
                                Transition ( 0 )
                                Radius ( 1.0 )
                        )
                        State   (
                                Duration ( 0.35 )
                                LightColour ( 00000000 )
                                Position ( 0.0 1.2 -8.955 )
                                Azimuth ( -180 -180 -180 )
                                Transition ( 0 )
                                Radius ( 1.0 )
                        )
                )
        )
```

```
                Light   (
                        comment( Rear red light flashing bright )
                        Type            ( 0 )
                        Conditions      (
                                Headlight ( 3 )
                                Unit ( 3 )
                        )
                        FadeIn  ( 0.5 )
                        FadeOut ( 0.5 )
                        Cycle           ( 0 )
                        States  (       2
                                State   (
                                        Duration ( 0.35 )
                                        LightColour ( 80ff0000 )
                                        Position ( 0.0 1.2 -8.955 )
                                        Azimuth ( -180 -180 -180 )
                                        Transition ( 0 )
                                        Radius ( 1.0 )
                                )
                                State   (
                                        Duration ( 0.35 )
                                        LightColour ( 00000000 )
                                        Position ( 0.0 1.2 -8.955 )
                                        Azimuth ( -180 -180 -180 )
                                        Transition ( 0 )
                                        Radius ( 1.0 )
                                )
                        )
                )
        )

        BrakeEquipmentType( "Handbrake, Triple_valve, Auxilary_reservoir, Emergency_brake_reservoir" )
<----- THIS SECTION HAS SOME SORT OF ISSUE THAT I'M NOT GOING TO CHECK,
        BrakeSystemType( "Air_single_pipe" )
<----- IN OPENRAILS THE BRAKES AREN'T RECOGNIZED
        MaxBrakeForce( 22.13kN )
!

        MaxHandbrakeForce( 22.13kN )
!
        NumberOfHandbrakeLeverSteps( 100 )

!
        EmergencyBrakeResMaxPressure( 110 )
        TripleValveRatio( 2.5 )
!
        MaxReleaseRate( 2.27 )
        MaxApplicationRate( 1.717 )
!
        MaxAuxilaryChargingRate( 1 )
        EmergencyResCapacity( 2.604 )
        EmergencyResChargingRate( 1 )
!
        BrakeCylinderPressureForMaxBrakeBrakeForce( 64 )
<-----------------------------------------------------------------------

        Sound ( "GenFreightWag1.sms" )
)
```

Make a new brake system section and typed it in a STD_Wag_Brake.inc:

```
        BrakeEquipmentType ( "Handbrake, Triple_valve, Auxilary_reservoir, Emergency_brake_reservoir" )
        BrakeSystemType( "Air_single_pipe" )
        MaxBrakeForce( 78kN )
        MaxHandbrakeForce ( 57.2kN )
        NumberOfHandbrakeLeverSteps( 100 )
        EmergencyBrakeResMaxPressure( 110 )
        TripleValveRatio( 2.5 )
        EmergencyResVolumeMultiplier ( 1.461 )
        MaxReleaseRate( 22.2 )
        MaxApplicationRate( 13.9 )
        MaxAuxilaryChargingRate( 20 )
        EmergencyResCapacity( 2.025ft^3 )
        EmergencyResChargingRate( 20 )
        BrakePipeVolume ( 0.307ft^3 )
        BrakeCylinderPressureForMaxBrakeBrakeForce( 90 )
```

Make an `Openrails` folder inside the WVOGrain folder, where I placed a text file with the following text called `WVOGrain.wag` (the well known include method, with comments)

```
                                                                        <----- REQUIRED EMPTY LINE
include ( "..\\WVOGrain.wag" )                                          <----- CALLS THE ORIGINAL WAG
FILE
Wagon (                                                                 <----- THIS MEANS: INSERT THE
NEXT LINES IN THE WAGON SECTION
        include ( "..\\..\\Common.inc\\Wagons\\Std_Wag_Brakes.inc" )    <----- CALLS THE .INC FILE WITH
THE NEW PARAMETERS
)                                                                       <----- CLOSE THE WAGON SECTION
```

This trick is in the manual, where it explains how to add instructions to remove the trees on rails in a route without touching the original .trk definition file.

The final result, to Open Rails, would be:

```
SIMISA@@@@@@@@@@JINX0D0t______                                                  <---- START OF
THE ORIGINAL WVOGRAIN.WAG

Wagon ( WVOGrain
        comment( WVO Covered hopper, hi-hip version)
        comment( Physics by Chris Lee)
        Type ( Freight )
        WagonShape ( WVOGrain.s )
        Size ( 3.243m 4.728m 17.503m)
        comment( 23.776t empty, 106.747t full )
        Mass ( 73.78t )
        WheelRadius ( 36in/2 )
        InertiaTensor ( Box (3m 3.6m 13.5m) )
        Coupling (
                Type ( Automatic )
                Spring (
                        Stiffness ( 1e6N/m 5e6N/m )
                        Damping ( 1e6N/m 1e6N/m )
                        Break ( 5.1e8N 5.1e8N )
```

```
                r0 ( 20cm 30cm )
        )
        comment( CouplingHasRigidConnection () )
        Velocity ( 0.1m/s )
)
Buffers (
        Spring (
                Stiffness ( 1e6N/m 5e6N/m )
                Damping ( 1e6N/m/s 1e6N/m/s )
                r0 ( 0m 1e9 )
        )
        Centre ( 0.5 )
        Radius ( 1 )
        Angle ( 0.5deg )
)

Adheasion ( 0.2 0.4 2 0 )
DerailRailHeight ( 4cm )
DerailRailForce ( 2.5N/kg*23t )
DerailBufferForce ( 400kN )
NumWheels (     8 )
Friction (
        871N/m/s                0               1mph            3.26N/m/s               1.8
        5.1N/rad/s              1               -1rad/s         0               1
)
Lights  (       2

        Light   (
                comment( Rear red light flashing dim )
                Type            ( 0 )
                Conditions      (
                        Headlight ( 2 )
                        Unit ( 3 )
                )
                FadeIn  ( 0.5 )
                FadeOut ( 0.5 )
                Cycle           ( 0 )
                States  (       2
                        State   (
                                Duration ( 0.35 )
                                LightColour ( 80ff0000 )
                                Position ( 0.0 1.2 -8.955 )
                                Azimuth ( -180 -180 -180 )
                                Transition ( 0 )
                                Radius ( 1.0 )
                        )
                        State   (
                                Duration ( 0.35 )
                                LightColour ( 00000000 )
                                Position ( 0.0 1.2 -8.955 )
                                Azimuth ( -180 -180 -180 )
                                Transition ( 0 )
                                Radius ( 1.0 )
                        )
                )
        )
        Light   (
                comment( Rear red light flashing bright )
```

```
                        Type            ( 0 )
                        Conditions      (
                                Headlight ( 3 )
                                Unit ( 3 )
                        )
                        FadeIn  ( 0.5 )
                        FadeOut ( 0.5 )
                        Cycle           ( 0 )
                        States  (       2
                                State   (
                                        Duration ( 0.35 )
                                        LightColour ( 80ff0000 )
                                        Position ( 0.0 1.2 -8.955 )
                                        Azimuth ( -180 -180 -180 )
                                        Transition ( 0 )
                                        Radius ( 1.0 )
                                )
                                State   (
                                        Duration ( 0.35 )
                                        LightColour ( 00000000 )
                                        Position ( 0.0 1.2 -8.955 )
                                        Azimuth ( -180 -180 -180 )
                                        Transition ( 0 )
                                        Radius ( 1.0 )
                                )
                        )
                )
        )

        BrakeEquipmentType( "Handbrake, Triple_valve, Auxilary_reservoir, Emergency_brake_reservoir" )
        BrakeSystemType( "Air_single_pipe" )
        MaxBrakeForce( 22.13kN )

        MaxHandbrakeForce( 22.13kN )
        NumberOfHandbrakeLeverSteps( 100 )

        EmergencyBrakeResMaxPressure( 110 )
        TripleValveRatio( 2.5 )
        MaxReleaseRate( 2.27 )
        MaxApplicationRate( 1.717 )
        MaxAuxilaryChargingRate( 1 )
        EmergencyResCapacity( 2.604 )
        EmergencyResChargingRate( 1 )
        BrakeCylinderPressureForMaxBrakeBrakeForce( 64 )

        Sound ( "GenFreightWag1.sms" )

        BrakeEquipmentType ( "Handbrake, Triple_valve, Auxilary_reservoir, Emergency_brake_reservoir" )
<---- THIS IS THE ADDED SNIPPET FROM THE .INC
        BrakeSystemType( "Air_single_pipe" )
        MaxBrakeForce( 78kN )
        MaxHandbrakeForce ( 57.2kN )
        NumberOfHandbrakeLeverSteps( 100 )
        EmergencyBrakeResMaxPressure( 110 )
        TripleValveRatio( 2.5 )
        EmergencyResVolumeMultiplier ( 1.461 )
        MaxReleaseRate( 22.2 )
        MaxApplicationRate( 13.9 )
```

```
        MaxAuxilaryChargingRate( 20 )
        EmergencyResCapacity( 2.025ft^3 )
        EmergencyResChargingRate( 20 )
        BrakePipeVolume ( 0.307ft^3 )
        BrakeCylinderPressureForMaxBrakeBrakeForce( 90 )
<---- ...TO HERE
)                                                                                <----- END
OF ORIGINAL FILE
```

Clear are mud still, I suppose… more examples are needed.

Option 1 - The Benefits of Common Folders for Re-skinners

Lets say you were going to repaint one of Erick's models. If this is the case, all you really need to do is create a new folder for your content, develop a new SKIN and adjust the existing SHAPE details to refer to the new TEXTURE. Many of the NAVS releases also include a paint kit release that offer much in the way of guidance for re-skinning, making these the best choice for newcomers that want to give it a try.

For example, if you used the NAVS 40 ft Boxcar kit as a base model, your WAG files would basically look like this (Taken from CP 40 foot Grain Boxcars by JW Mercer and based on NAVS_BOXCAR_40_PS-TYPE car by Erick Cantu):

```
SIMISA@@@@@@@@@@JINX0D0t______

Wagon ( CP_123541_LD ①

        Name ( "CPR_Grain_Boxcar_123541_LD" ) ②

    ORTSFreightAnims (
        MSTSFreightAnimEnabled ( 0 )
        FreightAnimStatic (
            Shape( ../NAVS_COMMON/Boxcar_40_PS-Type/Boxcar_40_PS-Type_carbody_2.s ) ③
            Offset( 0, 0, 0 )
            FreightWeight( 0lb )
        )
    )

    Include( "../NAVS_COMMON/INCLUDE/AAR_Type-E.inc" ) ④
    Include( "../NAVS_COMMON/Boxcar_40_PS-Type/PS-Type_40_LD_A3_Conv.inc" )

)
```

① The updated name of the new content, Note: Its is common to use ReportingMark_CarNumber_LD or MT (Loaded or Empty) format for naming.

② The Updated Descriptive name that will be used.

③ The updated .S file with the reference to the new texture being applied

④ The default includes for common data

If you did not have the NAVS 40 Ft Box car installed, you would need to install `NAVS_Boxcar_40_PS-Type.zip` (Elvas Tower or Trainsim) and `navsfr10.zip` (Trainsim).

Option 2 - Completely New Content (with possible help from NAVS)

When creating new content, the amount of work involved can seem daunting at first. Not only do you need to consider the 3D Model and Texturing details, you need to research the Physics and Sounds for the model.

For Sounds

It would be a good recommendation to consider employing the file: `navsfr10.zip` aka "Open Rails NAVS Car Sounds v2.5" available at trainsim.com. Current Link: https://www.trainsim.com/forums/downloadnew.php?fid=37487 If you are working on an ENG file, consider `navscf10.zip`, Current Link: https://www.trainsim.com/forums/downloadnew.php?fid=34168

For WAG file

It would be a good recommendation to use Erick's examples as guidance for how a model Open Rails WAG or ENG file could be formatted.

If you are not a legacy Microsoft Train Simulator user, you may not have access to some of the track and weather sounds used by Microsoft's simulator. If this is the case, you will also want `navs_env.zip` which is a public domain replacement for potentially missing environmental sounds.

If you do want to follow the **NAVS** guidelines, consider indicating this in the naming of your content by adding the `NAVS` prefix to your file names. So what does using this approach mean? Well, it is a well thought out plan for implementing rolling stock that differs greatly from legacy Microsoft Train Simulator. It is a framework that fully embraces Open Rails and it's feature set.

Erick Cantu is very particular about how to model a 3D Model freight car. Using his work as reference material is probably the best place to start. So let's take a deep dive into a model.

Gathering Details

Things that you already know would be Car Dimensions, Coupler Type, Brake Type but you might still need to gather details related to Car Empty Weight and Car Loaded Weight.

So, if we look at a typical NAVS Example as a guide, what will we find? Once again we see the minimal MAIN File that references a specific shape .S file and freightAnim references for Reporting Mark decals.

```
SIMISA@@@@@@@@@@JINX0D0t______

Wagon ( NAVS_Boxcar_NAVX_103100_LD

    ORTSFreightAnims (
        FreightAnimStatic (
            Shape( ../NAVS_COMMON/Boxcar_50_PS-Type/Boxcar_50_PS-Type_carbody_1.s )
            Offset( 0, 0, 0 )
            FreightWeight( 0lb )
        )
        FreightAnimStatic (
            Shape( ../NAVS_COMMON/Boxcar_50_PS-Type/DECALS/A1.s )
            Offset( 0, 0, 0 )
            FreightWeight( 0lb )
        )
    )

    Include( "../NAVS_COMMON/INCLUDE/AAR_Type-E.inc" )
    Include( "../NAVS_COMMON/Boxcar_50_PS-Type/PS-Type_50_LD_A3_Plain.inc" )

)
```

The rest is coming from INCLUDE references… so lets look at those as well. In the folder reference `../NAVS_COMMON/INCLUDE/AAR_Type-E.inc` we find the Coupler Type details and these values can be shared among many many vehicles.

```
Comment ( //////////AAR Type E Couplers\\\\\\\\\\ )

    Coupling (
        Type ( Automatic )
        Spring (
            Stiffness ( 1.1e6N/m 4.8e6N/m )
            Damping ( 1.1e6N/m/s 1.1e6N/m/s )
            Break ( 390000lbf 390000lbf )
            r0 ( 20cm 30cm )
        )
        Velocity ( 0.1m/s )
    )
    Coupling (
        Type ( Automatic )
        Spring (
            Stiffness ( 1.1e6N/m 4.8e6N/m )
            Damping ( 1.1e6N/m/s 1.1e6N/m/s )
            Break ( 390000lbf 390000lbf )
            r0 ( 20cm 30cm )
        )
        Velocity ( 0.1m/s )
    )
    Buffers (
        Spring (
            Stiffness ( 1e6N/m 5e6N/m )
            Damping ( 1.1e6N/m/s 1.1e6N/m/s )
            r0 ( 0m 1e9 )
        )
        Centre ( 0.5 )
        Radius ( 1 )
        Angle ( 0.5deg )
    )
```

And then we have the references to the specifics of this type of car in `../NAVS_COMMON/Boxcar_50_PS-Type/PS-Type_50_LD_A3_Plain.inc`. This folder would could many alternatives configurations of the main model as well.

```
    WagonShape ( ../NAVS_COMMON/Boxcar_50_PS-Type/Truck_50T_S2_Plain_50_PS-Type.s )

    Type ( Freight )
    Size ( 122.00in 180.00in 652.00in )
    Mass ( 169000lb )
    WheelRadius ( 16.5in )
    NumWheels ( 4 )
    AntiSlip ( 0 )

Comment ( //////////Friction\\\\\\\\\\ )

    ORTSBearingType ( Friction )
    ORTSDavis_A ( 225.84lb )
    ORTSDavis_B ( 3.8025lbf/mph )
    ORTSDavis_C ( 0.061998lbf/mph^2 )

Comment ( //////////Brakes - general\\\\\\\\\\ )

    BrakeEquipmentType( "Handbrake, Triple_valve, Auxilary_reservoir, Emergency_brake_reservoir,
Retainer_4_Position" )
    BrakeSystemType( "Air_single_pipe" )

    MaxBrakeForce ( 18928lbf )
    Comment ( 169,000-lb loaded weight x 35% loaded cast iron shoe brake ratio x 32% brake shoe friction
coefficient )

    MaxHandbrakeForce( 5408lbf )
    Comment ( AAR minimum 10% handbrake ratio x 169,000-lb Gross Rail Load x 32% brake shoe friction
coefficient )
    NumberOfHandbrakeLeverSteps( 100 )

    ORTSBrakeShoeFriction   (
                            000.00      0.32
                            016.09      0.15
                            032.19      0.10
                            048.28      0.08
                            064.37      0.07
                            096.56      0.05
                            128.75      0.05
                            160.93      0.05
                            )

    EmergencyBrakeResMaxPressure( 90 )
    EmergencyResVolumeMultiplier ( 1.50 )
    EmergencyResCapacity( 1.86ft^3 )
    EmergencyResChargingRate( 10 )

    TripleValveRatio( 2.5 )
    MaxReleaseRate( 2.50 )
    MaxApplicationRate( 10.00 )
    MaxAuxilaryChargingRate( 11 )
    BrakeCylinderPressureForMaxBrakeBrakeForce( 70 )

    Sound ( "freight_33_LD_castiron.sms" )
```

Notice how Erick has carefully layed out the design of the overall WAG file and the related INCLUDES to allow for sharing of

common settings and to allow for a refinement of specific details. Also note that the CAR BODY is actually a Freight Anim entry and the MAIN BODY of the vehicle is actually based on the type of BOGEY used. A novel idea that is worth imitating in our own designs. The TRUCK texture file (shared among all variants) is located in the same folder as `../NAVS_COMMON/Boxcar_50_PS-Type/PS-Type_50_LD_A3_Plain.inc` while the car body texture and decal texture is located in the same location as the main WAG file.

While you can still use a **default** MSTS-like WAG file with Open Rails, after seeing what Erick has done, you might be able to see why embracing the new way could be beneficial. This NAVS method is very re-painter-friendly and the fact that Erick also supplies a layered BITMAP file is the ultimate in "being nice to re-painters".

My recommendation is to use these settings as a guide for making new USA Based Rolling stock. If you would like guidance on other types of vehicles, I would recommend using the resources at Peter Newell's Coals to Newcastle website: https://www.coalstonewcastle.com.au/physics/

It's up to you how you want to handle your ENG and WAG files. You can go with the \common\

Another Example of Rolling Stock

Here is an example of a traditional MSTS style WAG file for USA style rolling stock. This is a BLLW release. It contains NONE of the new tokens related to Open Rails specifically so should work reliably in MSTS as well. It does utilize `FreightAnim` though the WAG for this car is to an EMPTY car, so the freight anim setting is commented out. In Open Rails, only one Coupling entry is used, so Open Rails will only need to see one in the WAG file.

The `SIMISA@@@@@@@@@@JINX0D0t` entry is a header requirement of [msts] and is not required for Open Rails

```
SIMISA@@@@@@@@@@JINX0D0t______

Wagon ( BLLW-H21a-PRR01
    Type ( Freight )
    Comment ( BLLW H21a 2.0 Model by Thomas J Pearce )
    Comment ( Models setup for Empty Coal or Ore Loads )
    Comment ( Freightanim ( BLLW-Coal.s 1 1 ) )
    Comment ( Freightanim ( BLLW-Ore.s 1 1 ) )
    Name ( "BLLW H21a PRR 137788 CK" )
    WagonShape ( BLLW-H21a-PRR01.s )
    Size ( 2.965933m 3.463495m 13.28m )
    CentreOfGravity ( 0m 1.771784m 0m )
    Mass       ( 22.90641t )
    WheelRadius ( 0.4604005m )
    InertiaTensor ( Box (2.965933m 3.463495m 13.28m) )
    Coupling (
        Type ( Automatic )
        Spring (
            Stiffness ( 9.8e5N/m 1.76e6N/m )
            Damping ( 1.77e6N/m/s 1.77e6N/m/s )
            Break ( 2.45e7N 1.78e6N )
            r0 ( 0cm 4cm )
        )
        Comment ( CouplingHasRigidConnection ( 1 ) )
        Velocity ( 0.1m/s )
    )
    Coupling (
        Type ( Automatic )
        Spring (
            Stiffness ( 9.8e5N/m 1.76e6N/m )
            Damping ( 1.77e6N/m/s 1.77e6N/m/s )
```

```
            Break ( 2.45e6N 1.78e6N )
            r0 ( 0cm 4cm )
        )
        CouplingHasRigidConnection ( 1 )
        Velocity ( -0.1m/s )
    )
    Buffers (
        Spring (
            Stiffness ( 9.8e5N/m 9.8e5N/m )
            Damping ( 1.7e6N/m/s 1.7e6N/m/s )
            r0 ( 20cm 30cm )
        )
        Centre ( 0.5 )
        Radius ( 1 )
        Angle ( 0.5deg )
    )
    Adheasion ( 0.2 0.4 2 0 )
    DerailRailHeight ( 4cm )
    DerailRailForce ( 2.5*27t )
    DerailBufferForce ( 400kN )
    NumWheels ( 8 )
    Friction (
        100N/m/s        1        -1mph        0        1
        5.1N/rad/s        1        -1rad/s        0        1
    )
    BrakeEquipmentType( "Handbrake, Triple_valve, Auxilary_reservoir, Emergency_brake_reservoir" )
    BrakeSystemType( "Air_single_pipe" )
    MaxBrakeForce( 50kN )

    MaxHandbrakeForce( 35kN )
    NumberOfHandbrakeLeverSteps( 100 )

    TripleValveRatio( 2.5 )
    MaxReleaseRate( 15 )
    MaxApplicationRate( 25 )
    MaxAuxilaryChargingRate( 5 )
    EmergencyResCapacity( 7 )
    EmergencyResChargingRate( 5 )
    EmergencyBrakeResMaxPressure( 90 )
    BrakeCylinderPressureForMaxBrakeBrakeForce( 50 )

      Sound (    "GenFreightWag2.sms" )
)
```

If you keep up with changes in Open Rails, you will see that some items have changes to be a bit more logical, for instance, if you supply the new DERAIL COEFICIENT tokens then you no longer need to supply `NUMWHEELS` as `ORTSNumAxles` from the Derail values makes it redundant. Additionally, you only need to supply one `COUPLING` token, not two. Refer to Derail Coefficient section of the Open Rails manual for further details.

Specifying lights on locomotives and wagons

Introduction

In Microsoft Train Simulator, you can use the in-built lighting system to create a variety of lights for locomotives and wagons, including headlights, warning lights, and ditch lights. These lights can be used on both computer-controlled and player-driveable trains. The lights can be set to be steady or flashing, and can also be programmed to change based on certain parameters, such as a penalty brake application, or to turn on automatically based on the time of day or weather.

Lights are specified as part of a locomotive/wagon's .eng file (the same file that defines the physical characteristics of the unit, such as its height and weight, and its cab controls). Essentially, a set of conditions is defined and then lights that should light in this case are listed). Like all of the hand-editable files in Train Simulator, the .eng file can be edited in any Unicode text editor.

Available Tokens for Lights

- Type Glow Cone
- FadeIn
- FadeOut
- Cycle
- Headlight
- Unit
- Penalty
- Control
- Service
- TimeOfDay
- Weather
- Coupling
- Duration
- LightColour
- Position
- Radius
- Azimuth
- Elevation
- Transition
- Angle

Setting requirements

The first thing you must specify is when you want the light or lights in question to come on. This is done by setting a number of flags to define the requirements for the lights concerned to light – if no requirements are listed, the light will always be on.

Possible Requirements

- Headlight (0) (ignore headlight control status)
- Headlight (1) (the headlight control is at "Off")
- Headlight (2) (the headlight control is at "Dim")

- Headlight (3) (the headlight control is at “Bright”)
- -----
- Unit (0) (ignore unit status)
- Unit (1) (the unit is in the train but isn’t at the front or back)
- Unit (2) (the unit is the frontmost unit of the train)
- Unit (3) (the unit is the rearmost unit of the train)
- -----
- Penalty (0) (ignore penalty status)
- Penalty (1) (the train has not “stopped due to a penalty brake application”)
- Penalty (2) (penalty) (the train has stopped due to a penalty brake application” (useful for specifying an emergency warning light)
- -----
- Control (0) (ignore control status)
- Control (1) (the player is not control of the unit [i.e. it is a helper locomotive or wagon)
- Control (2) (the player is in control of the unit)
- -----
- Service (0) (ignore service status)
- Service (1) (the unit is not in service e.g. a failed train lying in a siding)
- Service (2) (the unit is in service)
- -----
- TimeOfDay (0) (ignore time of day)
- TimeOfDay (1) (only come on if it’s daytime)
- TimeOfDay (2) (only come on if it’s night
- -----
- Weather (0) (ignore weather)
- Weather (1) (only come on if it’s fine)
- Weather (2) (only come on if it’s raining)
- Weather (3) (only come on if it’s snowing)
- -----
- Coupling (0) (ignore coupling status)
- Coupling (1) (original front of wagon is coupled)
- Coupling (2) (original rear of wagon is coupled)
- Coupling (3) (both ends of wagon are coupled)

Defining lights

Once you have defined a set of requirements, you must list all the lights which you want to light when those requirements are fulfilled.

Type of light

Two types of lights can be defined in Train Simulator.

Glow

Two textured triangles. Glows are relatively inexpensive in terms of processing time and are perfect for use when you merely wish to show that a light is on rather than off. They do not affect polygons around them (i.e. they do not "light things up").

Cone

A "real" light which has an effect on the polygons around it. Again, this takes more processing time than a glow and should only be used if you wish to "light something up" as opposed to merely showing a light is on. Headlights are a suitable use for cone lights – one cone can simulate the light from a bank of headlights, with a glow being used to highlight each individual light if desired.

```
Type(n) (where n = 0 is glow, n = 1 is cone)
```

Fade-in and Fade-out times

These values simply let you define a number of seconds that the light will take to brighten to the brightness of its current state when it is lit, and the number of seconds it will take to die away when it is switched off. If you specify 0 for both values, the light will light and darken instantaneously.

```
FadeIn(t) (where t is the time in seconds) FadeOut(t) (where t is the time in seconds)
```

Cycle

If this value is set to 1, the light's states will be cycled through first forwards, then backwards and so on (e.g. 1, 2, 3, 4, 5, 4, 3, 2, 1, 2, 3, 4…). If this value is left as 0, the states will be played through in order repeatedly (e.g. 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4…)

```
Cycle(n) (where n = 0 is play in order repeatedly, n = 1 is cycle)
```

States

A light has a number of states – a state is simply the form of the light at a given time. States can be played through in order repeatedly or made to cycle backwards and forwards (see above). For example, to define a red flashing light, you would define one state that had the light bright red for 1 second and another that had it off for one second. When the two states were cycled between, the light would appear to flash. By altering colours, coordinates and durations of states, you can make lights appear to change colour, move around or flash.

A state is made up of the following values:

- Duration(t) Specifies the duration of this light state where t is the time in seconds.
- Colour(aarrggbb) A 32-bit hexadecimal value specifying the colour and translucency of the light. aa, rr, gg, bb are 8-bit hexadecimal values specifying the translucency (how "solid" the light is - this only affects glows) and red, green and blue colour components of the light colour respectively. (If you are not familiar with hexadecimal, a brief explanation is included in the appendices of this document).
- Position(x y z) The position offset in Cartesian coordinates from the centre of the unit.
- Radius(r) This is the radius of the light sphere for cone lights and the radius of the polygon "disc" for glow lights.
- Azimuth(min centre max) This specifies the heading (rotation about the Y-axis) of glow lights – in the simplest case min, centre and max can all be set to the same angle (in degrees). For more complex situations where you want to define the arc of the light you can specify for example -45, 0 and 45 degrees respectively to produce a light with a 90 degree arc. Essentially glow lights attempt to always face the camera but are limited by the azimuth and elevation settings. The best way to see what the values do is to experiment - set up a light with a set of values and then move the camera around it to see what happens.
- Elevation(min centre max) This specifies the elevation (rotation about the X-axis) of glow lights and (apart from the axis of effect) works identically to the azimuth.
- Transition(n) This specifies how the transition from one state to the next occurs - if n equals zero then the transition is instantaneous, if it is one then the light parameters are interpolated from this state to the next.
- Angle(n) (Cone lights only) the radius of the cone, in degrees.

Examples

Here are a few examples, taken from the Dash 9 .eng file.

Example 1 – the Dash 9's front headlights

```
Lights  (   10
        Light   (
            comment( Sphere of light )
            Type        ( 1 )
            Conditions  (
                Headlight ( 3 )
                Unit ( 2 )
                )
            Cycle   ( 0 )
            FadeIn  ( 0.5 )
            FadeOut ( 0.5 )
            States  (   1
                State   (
                    Duration ( 0.0 )
                    LightColour ( ffffffff )
                    Position ( 0.0 3.5 18 )
                    Transition ( 0 )
                    Radius ( 400.0 )
                    Angle ( 15.0 )
                    )
                )
            )
        Light   (
            comment( Head light dim )
            Type        ( 0 )
            Conditions  (
                Headlight ( 2 )
                Unit ( 2 )
                )
            FadeIn  ( 0.5 )
            FadeOut ( 0.5 )
            Cycle   ( 0 )
            States  (   1
                State   (
                    Duration ( 0.0 )
                    LightColour ( 80ffffff )
                    Position ( 0.15 3.20 9.679 )
                    Azimuth ( 0.0 -5.0 5.0 )
                    Transition ( 0 )
                    Radius ( 1.0 )
                    )
                )
            )
        Light   (
            comment( Head light bright )
            Type        ( 0 )
            Conditions  (
                Headlight ( 3 )
                Unit ( 2 )
                )
            FadeIn  ( 0.5 )
```

```
            FadeOut ( 0.5 )
            Cycle   ( 0 )
            States  (   1
                State   (
                    Duration ( 0.0 )
                    LightColour ( ffffffff )
                    Position ( 0.15 3.20 9.679 )
                    Azimuth ( 0.0 -5.0 5.0 )
                    Transition ( 0 )
                    Radius ( 1.0 )
                    )
                )
            )

Light   (
            comment( Head light bright )
            Type        ( 0 )
            Conditions  (
                Headlight ( 3 )
                Unit ( 2 )
                )
            FadeIn  ( 0.5 )
            FadeOut ( 0.5 )
            Cycle       ( 0 )
            States  (   1
                State   (
                    Duration ( 0.0 )
                    LightColour ( ffffffff )
                    Position ( 0.15 3.20 9.679 )
                    Azimuth ( 0.0 -5.0 5.0 )
                    Transition ( 0 )
                    Radius ( 1.0 )
                    )
                )
            )
```

Example 2 – the Dash 9's front right flashing (ditch) light.

```
Light   (
            comment( Front right flashing light )
            Type        ( 0 )
            Conditions  (
                Headlight ( 3 )
                Unit ( 2 )
                )
            Cycle   ( 0 )
            FadeIn  ( 0.5 )
            FadeOut ( 0.5 )
            States  (   2
                State   (
                    Duration ( 0.3 )
                    LightColour ( ffffffff )
                    Position ( -0.60932 1.98713 10.600 )
                    Azimuth ( 0 -17.0 -17.0 )
                    Transition ( 0 )
                    Radius ( 1.5 )
                    )
                State   (
                    Duration ( 0.3 )
                    LightColour ( 00000000 )
                    Position ( -0.60932 1.98713 10.600 )
                    Azimuth ( 0 -17.0 -17.0 )
                    Transition ( 0 )
                    Radius ( 1.5 )
                    )

                )
            )
```

Example 3 – the Dash 9's rear red light

```
        Light   (
            comment( Rear red light )
            Type        ( 0 )
            Conditions  (
                Headlight ( 3 )
                Unit ( 2 )
                )
            FadeIn  ( 0.5 )
            FadeOut ( 0.5 )
            Cycle   ( 0 )
            States  (   1
                State   (
                    Duration ( 0.0 )
                    LightColour ( 80ff0000 )
                    Position ( 0.98651 2.12618 -10.677706 )
                    Azimuth ( -180 -180 -180 )
                    Transition ( 0 )
                    Radius ( 0.5 )
                    )
                )
            )

        )
```

A brief explanation of hexadecimal numbers

Hexadecimal is a base 16 counting system used by computers. Humans generally use base 10. In base 10, we have a units column, a tens column, a hundreds (ten squared) column, etc. and numbers can be described by specifying a value for each column.

For example, 57 has 5 in the tens column and 7 in the units column and (5 * 10) + 7 * 1 = 57.

In base 16 you have a units column, a sixteens column, a 256s (16 squared) column, etc. The extra digits are A to F so you count in hex like this : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (10 in decimal), B (11 in decimal), C (12), D (13), E (14), F (15), 10 (remember this is hex so that's one in the 16s column and zero in the units column giving you 16).

Another example the hex number 4D is 4 in the 16s column and D in the units column, but D is 13 so that's (4 * 16) + (13 * 1) =64 + 13 = 77.

In the case of lights, hexadecimal values are used to set an alpha, red, green and blue value for a light. Each value is set by a two-digit hexadecimal number, so can be anything between 00 (zero) and FF (F in the 16s column and F in the units column, so since F=15 that's (15*16) + (15*1) = 240 + 15 = 255). So a two-digit hexadecimal number can be anything between 0 and 255, i.e. there are 256 possible values.

Therefore the following setting:

```
0000FF00
```

…would result in a light that has no alpha (i.e. it's completely "transparent"), no red, full green and no blue. The result would be a transparent green light.

A more realistic example is:

```
80ff0000
```

The first two values (80) control the alpha. 80 = 8*16 = 128, so this light will be half-alphaed, or semi-transparent. The two red values are set to ff (255, as explained above) so the light's red value is at maximum. Since the green and blue values are set to 00, the light will be bright red (if they were set to ff as well, the colours would combine together to form a

bright white light).

One final example. The value:

```
60808000
```

Would result in a yellow, slightly more than half transparent (i.e. more transparent than the red light above) yellow light.

Effects

Available tokens for Effects

- StackFX
- CylindersFX
- WhistleFX
- SafetyValvesFX
- Exhaust
- Cylinders2FX Open Rails
- CompressorFX Open Rails
- GeneratorFX Open Rails
- Injectors1FX Open Rails
- Injectors2FX Open Rails
- HeatingSteamBoilerFX Open Rails
- WagonGeneratorFX Open Rails
- WagonSmokeFX Open Rails
- HeatingHoseFX Open Rails
- WaterScoopFX Open Rails
- TenderWaterOverflowFX Open Rails
- BearingHotboxFX Open Rails

Freight Animations

Available tokens for Freight Animations

- FreightAnim
- IntakePoint FuelWater FuelCoal FuelWood FuelSand FuelDiesel FreightGeneral FreightLivestock FreightFuel FreightGrain FreightCoal FreightGrvel FreightSand
- ORTSFreightAnims Open Rails
- FreightAnimContinuous Open Rails
- WagonEmptyWeight Open Rails
- EmptyMaxBrakeForce Open Rails
- EmptyMaxHandbrakeForce Open Rails
- EmptyORTSDavis_A Open Rails
- EmptyORTSDavis_B Open Rails
- EmptyORTSDavis_C Open Rails
- EmptyCentreOfGravity_Y Open Rails

- EmptyORTSWagonFrontalArea Open Rails
- EmptyORTSDavisDragConstant Open Rails
- FreightWeightWhenFull Open Rails
- FullMaxBrakeForce Open Rails
- FullMaxHandbrakeForce Open Rails
- FullORTSDavis_A Open Rails
- FullORTSDavis_B Open Rails
- FullORTSDavis_C Open Rails
- FullCentreOfGravity_Y Open Rails
- FullORTSWagonFrontalArea Open Rails
- FullORTSDavisDragConstant Open Rails
- MSTSFreightAnimEnabled Open Rails
- IsGondola Open Rails
- UnloadingStartDelay Open Rails
- FullAtStart Open Rails
- Shape Open Rails
- MaxHeight Open Rails
- MinHeight Open Rails
- ORTSWaterScoopFillElevation Open Rails
- ORTSWaterScoopDepth Open Rails
- ORTSWaterScoopWidth Open Rails

Creating 3D Cabs

TESTING

Train -Use the key page up/down .....

HOW TO MAKE 3D CABs

Includes a 3D Cab

A Quick Intro Guide

FOR OPEN RAILS WITH BLENDER 3

By ExRail

GETTING READY

Download blender 3.5 from
WWW.BLENDER.ORG
And learn to use it a bit!

Download S file Exporter plugin by
by Wayne Campbell
SITES.GOOGLE.COM/VIEW/BLENDERTOMSTS

SUPPORT TOOLS

Notepad++ (CVF 3D cab)
NOTEPAD-PLUS-PLUS.ORG

Gimp 2.10 (DDS support)
WWW.GIMP.ORG

Total Commander
WWW.GHISLER.COM

open rails

CONTENT

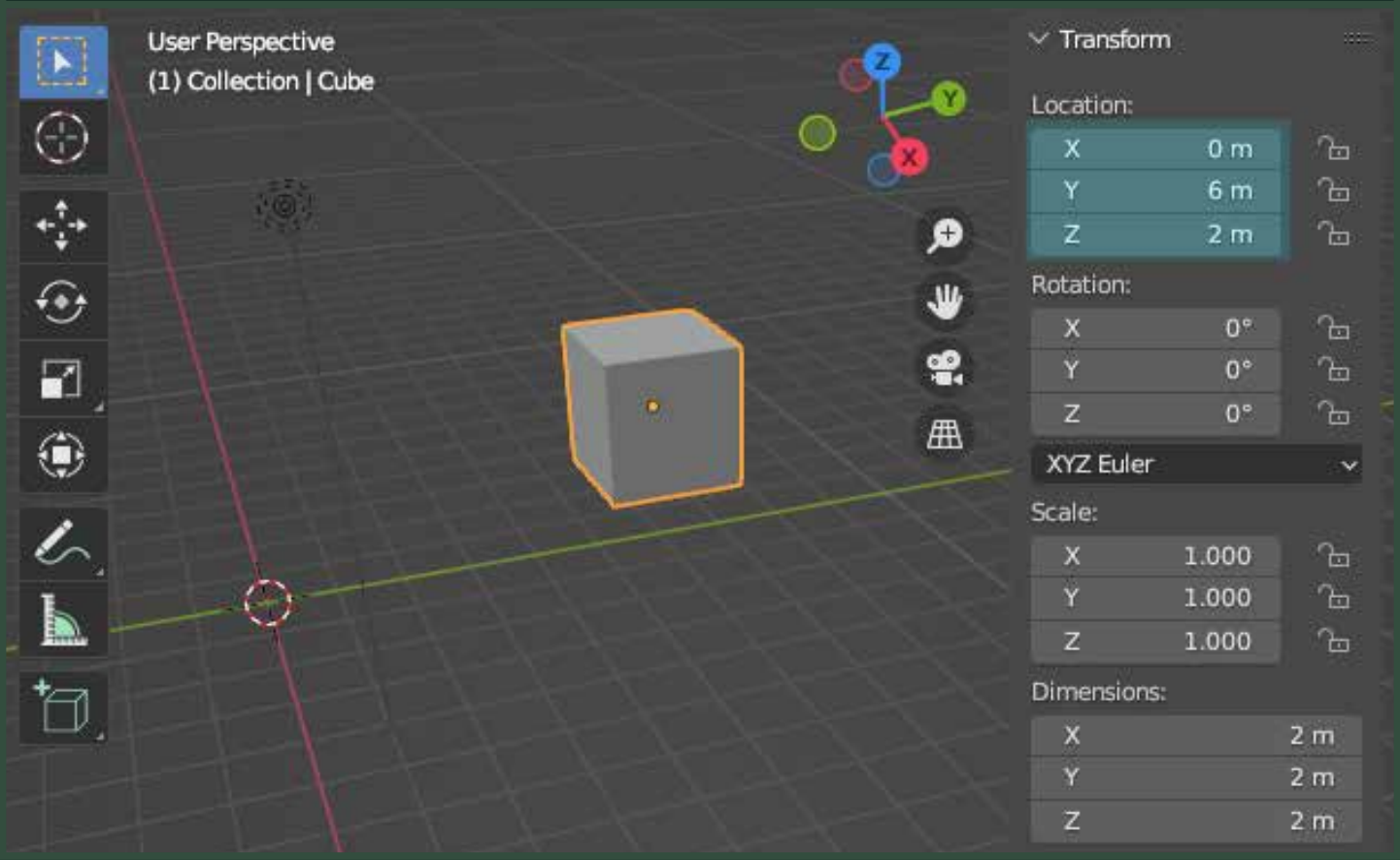
STARTING FROM SCRATCH

A QUICK GUIDE IN 9 PAGES

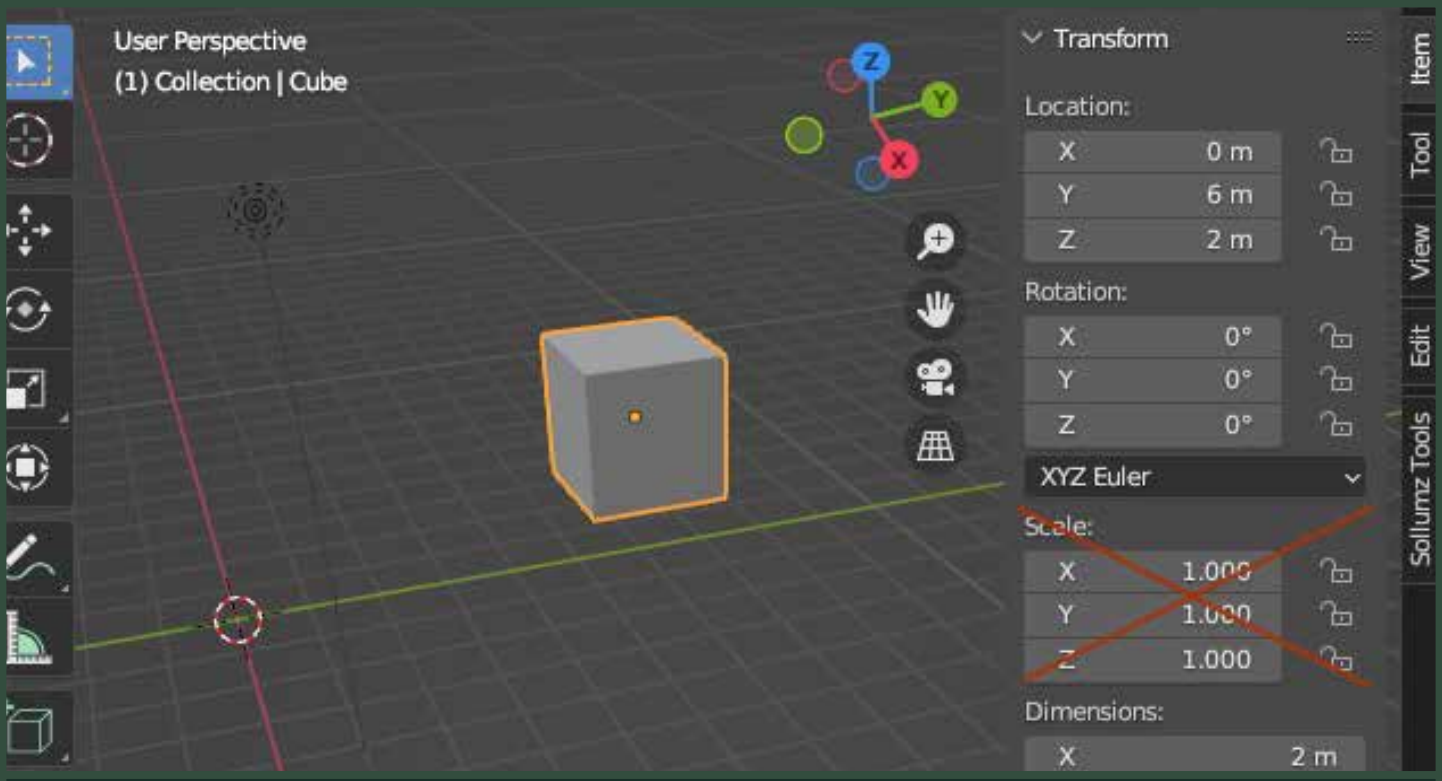
1. Placing The Default Cube
2. Adding a plane
3. Changing the dimensions
4. Matrial & Illumination
5. Collection and Instrument Names
6. The 3D cab .cvf file
7. The 3D cab & the eng file
8. Export .s file
9. Testing

open rails

PLACING THE DEFAULT CUBE

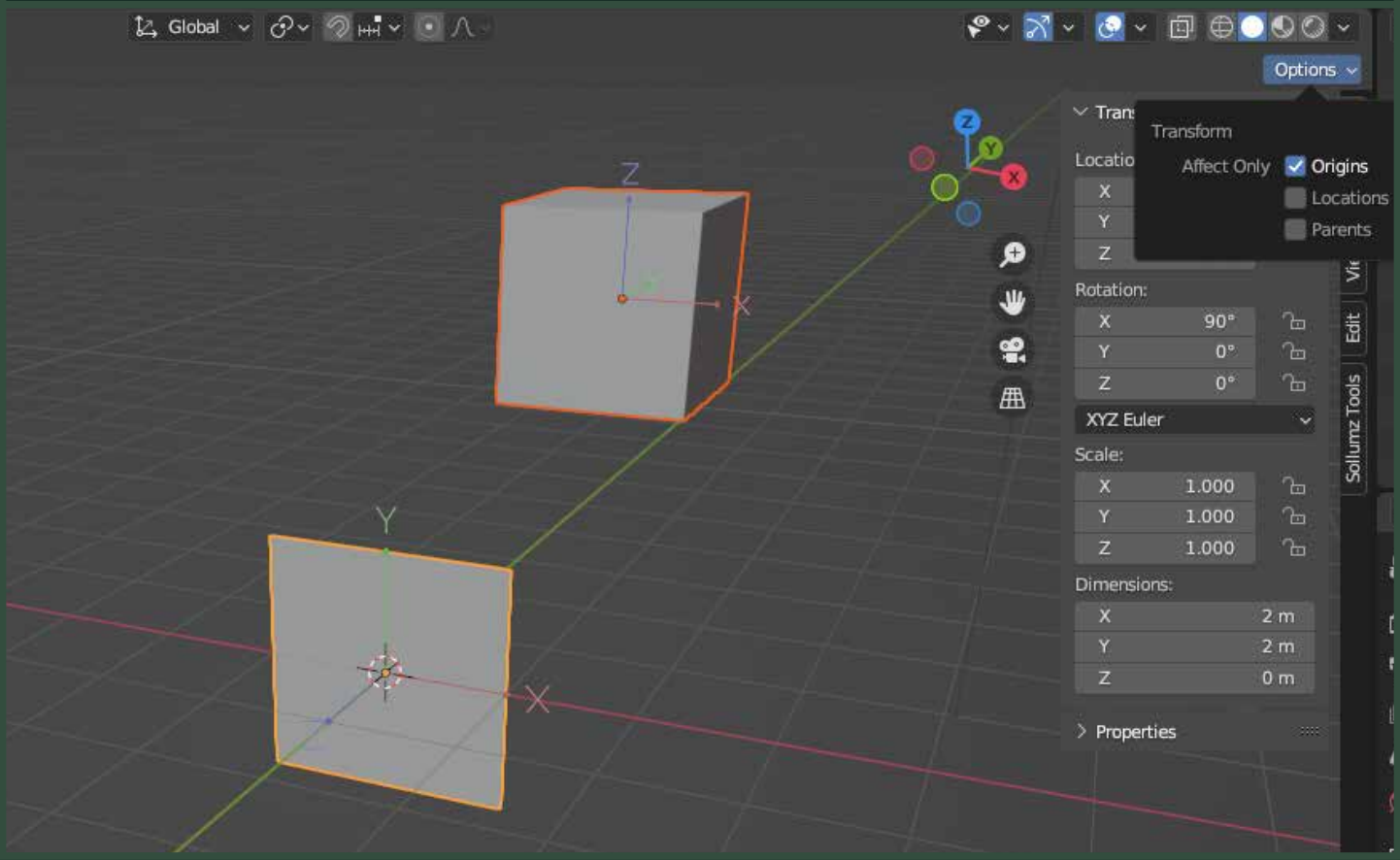


A good place to start is properly around **X=0, Y=6, Z=2,** change the numbers to move it there.



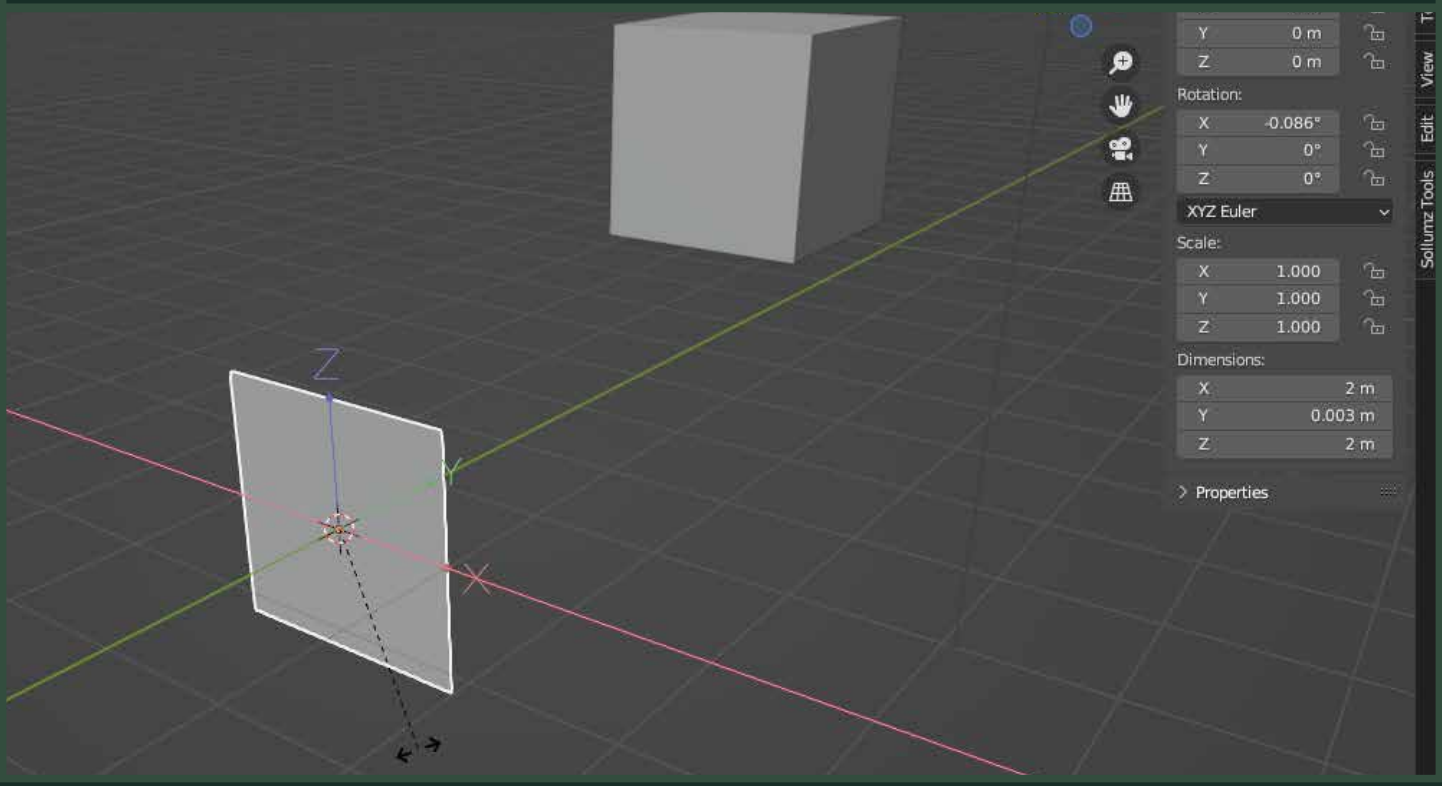
No scaling for objects to be used as instruments !
All instments must be made with a scale 1.0 or they will appear bigger or smaler in Open Rails 3D cab, scaling can be done in Edit Mode.

ADDING A PLANE

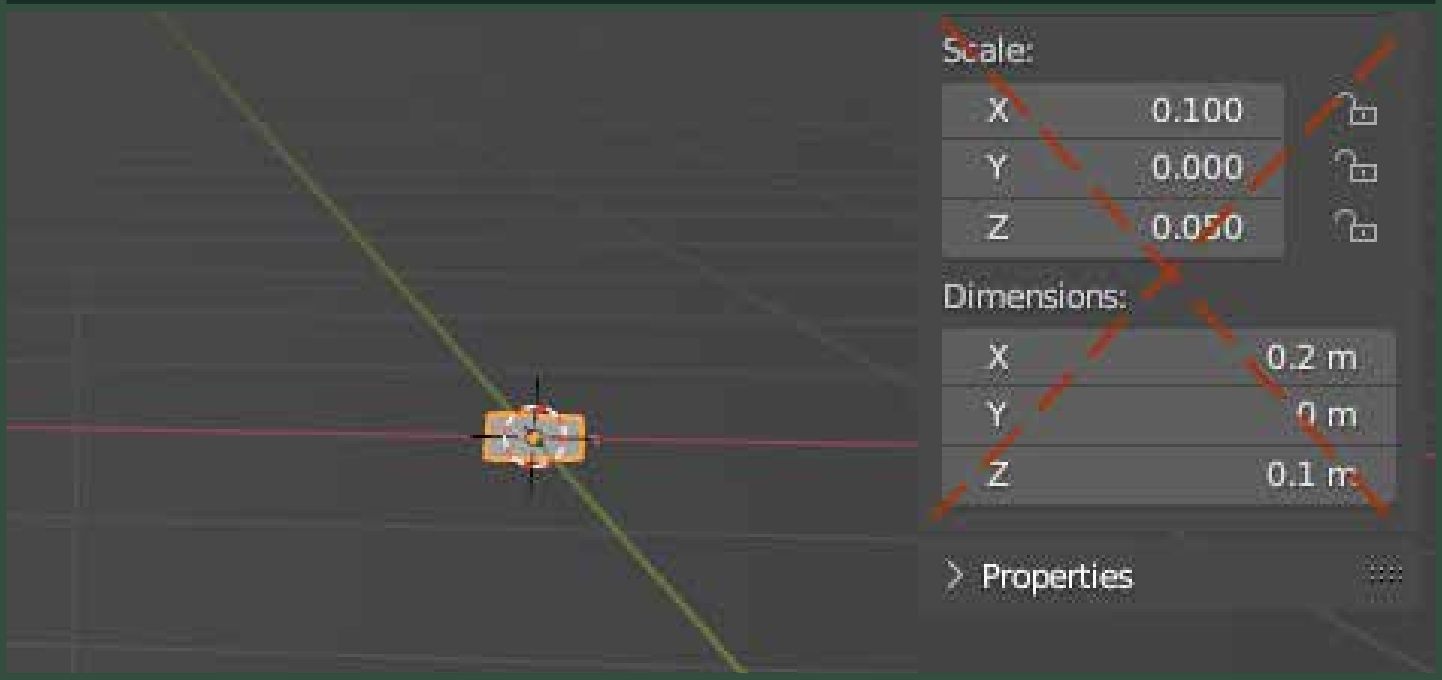


Add a plane and **Rotate it Into X=90°** to be used as a Digital clock.
Note that the orientation of the privot point is wrong! **Enable 'Affect Only'**

Rotate the privot point 90° to match the cubes so **Z is up, Y forward** and
Remember To Disable 'Effect Only' again

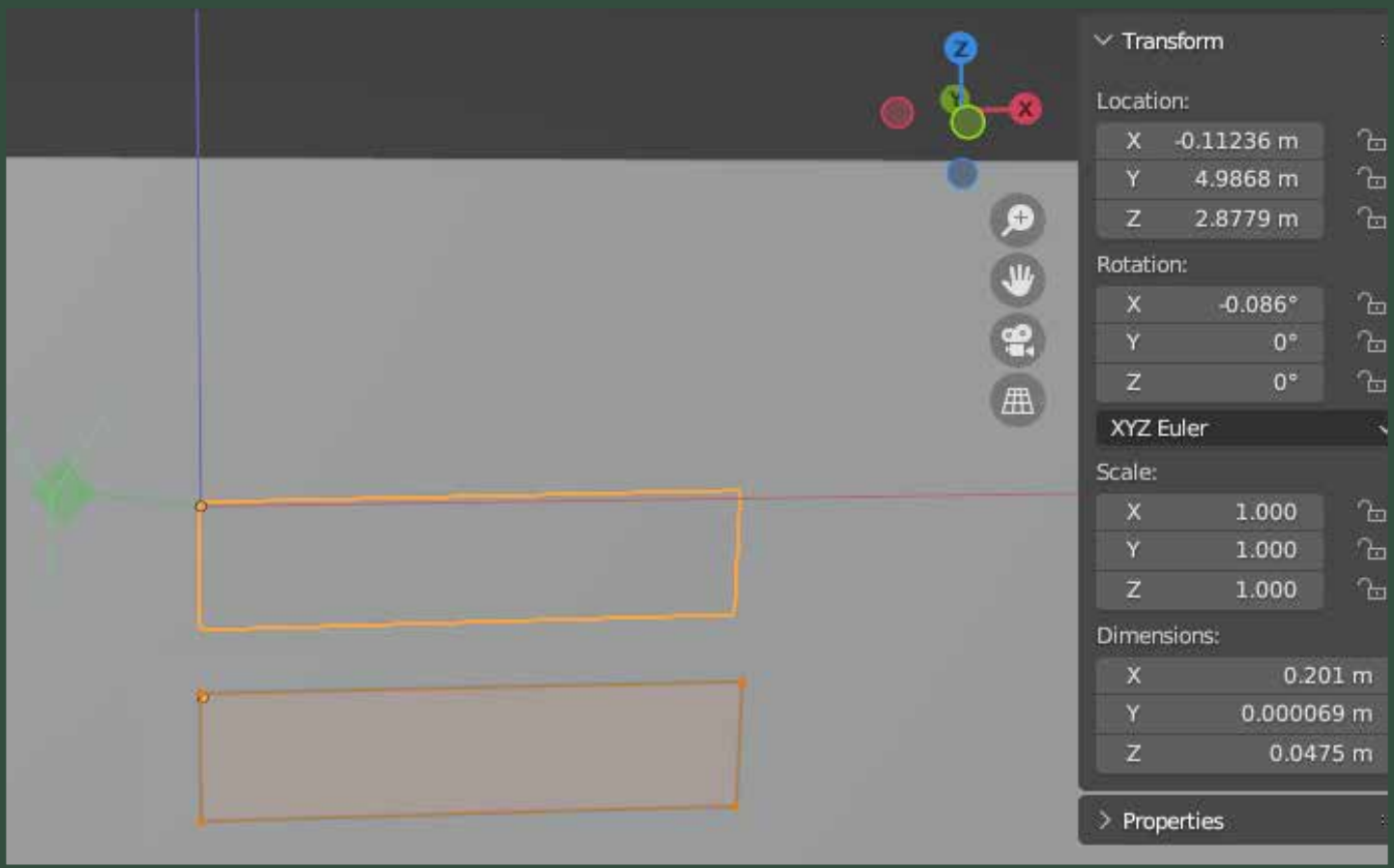


CHANGING THE PLANES DIMENSIONS



In Object mode altering the Dimensions also effect the scaling. **CTRL+A Scale set it to 1.0**

NB: Much time can be spend on fault finding where it turns out to be caused by scaling or privot orientation.



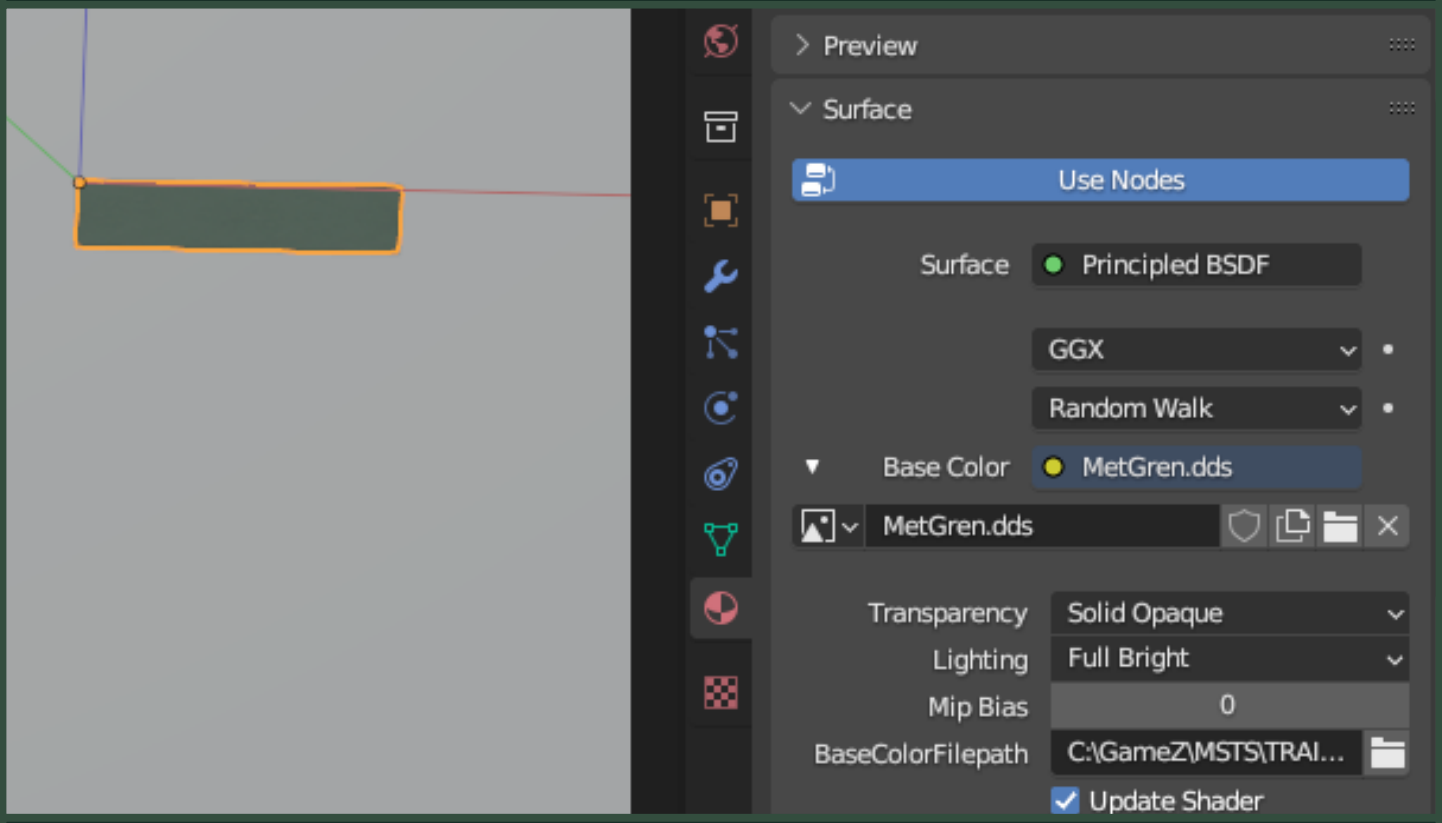
Go into Edit Mode

Change the vertex positions to get a **20 cm wide** and **5 cm high** rectangle, either by scaling or movement.

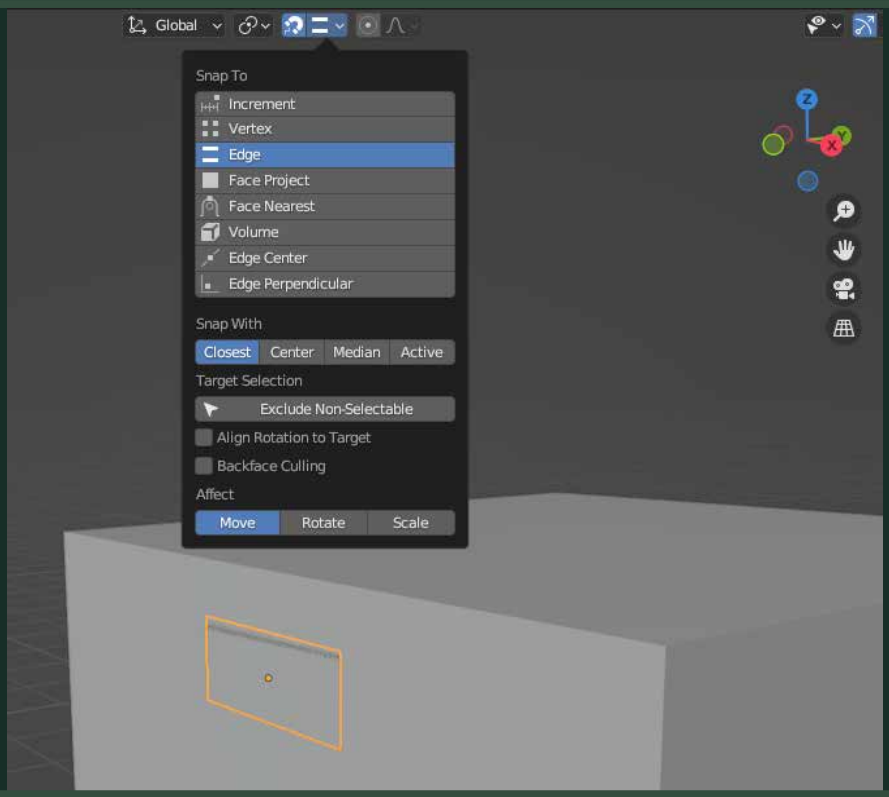
Go into Object Mode

Enable 'Affect Only' and move the privot point to the top left cornor **Disable 'Affect Only'**

MATRIAL

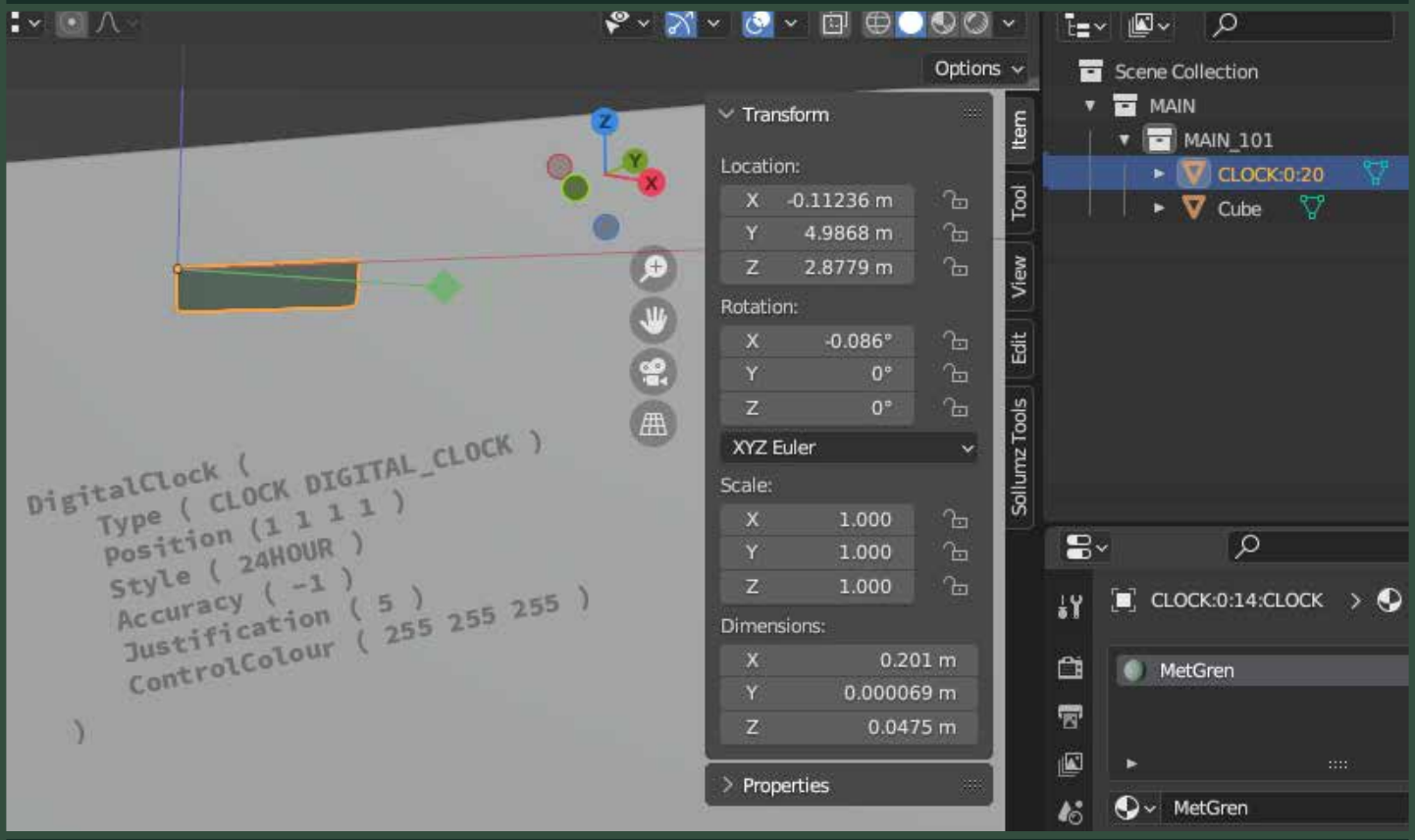


Create and assign a image texture material and set the illumination to **Full Bright** and point to the texture.



Enable Snap to Edge and move the plane close to the cube to snap it.
Note the flickering Z collision, move it a bit out from the cube to avoid that.

COLLECTION AND INSTRUMENT NAMING



Rename the default 'Collection' to MAIN
Create a new collection inside it with the name MAIN_0101
and **Move the objects into it.**

Rename the plane object to **CLOCK:0:20**

Tip: CLOCK uses the texture clock.ace by default, it can be changed by using:
CLOCK:n:fontsize:myfont

Tip: MAIN_XXXX = LOD Distance in meters, the 3D cap becomes invisible after 101 m

Look at:
Decoding CabViewControls 1.5 on page xxx for a list of controls, states and data.

THE 3D CAB .CVF FILE

```
SIMISA@@@@@@@@@@JINX0h0t______

Tr_CabViewFile (
    Position ( 0.0 0.0 0.0 )
    Direction ( 13 -17 0 )
    EngineData ( My 3D cab )
    CabViewControls ( 1

        DigitalClock (
            Type ( CLOCK DIGITAL_CLOCK )
            Position (1 1 1 1 )
            Style ( 24HOUR )
            Accuracy ( -1 )
            Justification ( 5 )
            ControlColour ( 255 255 255 )
        )   COMMENT( CLOCK:0:18 )

    )

)
```

Create a cvf file for the 3D cab - or copy the 2D cabs.

- *If the train already has a 2D cab, the name of the 3D cab cvf must be the same* or the 2D cab is not functionel - since the ENG file only has one entry.

Edit the text so it contains the following .

```
SIMISA@@@@@@@@@@JINX0h0t______
Tr_CabViewFile (
   Position ( 0.0 0.0 0.0 )
   Direction ( 0 0 0 )
   EngineData ( My3Dcab )
   CabViewControls ( 1
        DigitalClock (
             Type ( CLOCK DIGITAL_CLOCK )
             Position (1 1 1 1 )
             Style ( 24HOUR )
             Accuracy ( -1 )
             Justification ( 5 )
             ControlColour ( 255 255 255 )
        )
   )
)
```

```
    CabView ( "My3Dcab.cvf" )
    HeadOut ( -2 3 7 )

    AirBrakesAirCompressorPowerRating( 1.6 )
```

The .ENG file has the entry so be sure the filenames match.

THE 3D CAB & THE ENG FILE

```
569
570
571
572     ORTS3DCab(
573             ORTS3DCabFile ( Cab.s )
574             ORTS3DCabHeadPos ( 0.655 2.2 5.025 )
575             RotationLimit ( 40 60 0 )
576             StartDirection ( 0 0 0 )
577             Sound ( "Vectron_eng.sms")
578     )
579
580   )
581 Engine ( DSB_Vectron
582     Wagon ( DSB_Vectron )
583     Type ( Electric )
584     MaxPower ( 6400kW )
585     MaxForce ( 350kN )
586     MaxContinuousForce ( 300kN )
587     RunUpTimeToMaxForce ( 4.0 )
588     MaxVelocity ( 200km/h )
589     MaxCurrent ( 1640A )
590     WheelRadius ( 0.62m )
591     MaxSandingTime( -1s )
592     Sanding  ( 30km/h )
593     NumWheels ( 1 )
594
595     CabView ( "My3Dcab.cvf" )
596     HeadOut ( -2 3 7 )
597
598     AirBrakesAirCompressorPowerRating( 1.6 )
```

Open the ENG file of the loco you're desided to create a 3D cab for and add the following in the wagon section:

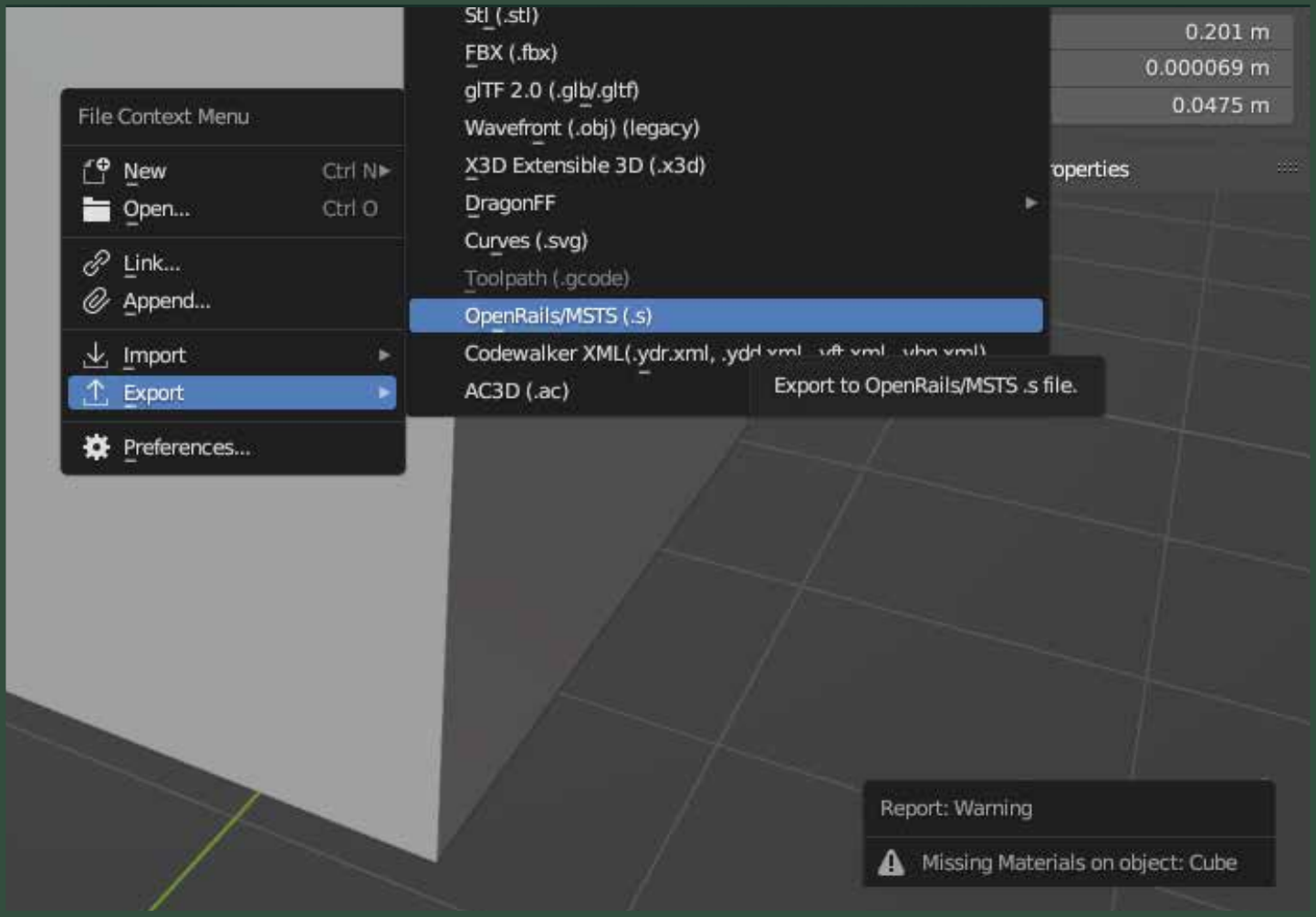
```
ORTS3DCab(
          ORTS3DCabFile ( MyCab.s )
          ORTS3DCabHeadPos ( 0.655 2.2 5.025 )
          RotationLimit ( 40 60 0 )
          StartDirection ( 0 0 0 )
          Sound ("Mytraincab_eng.sms")
)
```

CabFile (MyCab.s)
The exported shape file.

CabHeadPos (0.0 2.4 4.0)
Camera position - 2.2 m up and 5 meter forward from the locos center point

Direction (14.5 0 0)
- up / + down - looking 14.5° down

EXPORT



Exporter warning
Create a materiel texture for the cube to remove it.

THE FOLDER STRUCTURE

Train
- **.eng** - The train definition
- **.s** - The trains shape file/ Freight animation
- **.ace/dds** - The trains texture files
- **Cabview**
  - **.cvf** - The trains 2D Cab definition
  - **.ace/dds** - The 2D cabs texture files
- **3Dcabview**
  - **.cvf** - The trains 3D Cab definition
  - **.s** - The 3D Cab shape file
  - **.ace/dds** - The 3D Cab texture files

TESTING

Train -Use the key page up/down .....

Appendix A

Parameter	Program	Notes
Universal		
Type	MSTS	Steam Diesel Electric Freight Carriage Tender
WagonShape	MSTS	
Wagon	MSTS	
Size	MSTS	
Mass	MSTS	
WheelRadius	MSTS	
NumWheels	MSTS	
Sanding	MSTS	
Sound	MSTS	
CabView	MSTS	
HeadOut	MSTS	
Name	MSTS	
Description	MSTS	
Include	ORTS	
Steam Basic		
Type	MSTS	Steam Diesel Electric
WheelRadius	MSTS	
NumWheels	MSTS	
Sanding	MSTS	
IsTenderRequired	MSTS	
BoilerVolume	MSTS	
MaxBoilerPressure	MSTS	
MaxTenderWaterMass	MSTS	
MaxTenderCoalMass	MSTS	
SteamFiremanIsMechanicalStoker	MSTS	
NumCylinders	MSTS	
CylinderStroke	MSTS	
CylinderDiameter	MSTS	
ORTSSteamLocomotiveType	ORTS	Simple Compound Geared

Parameter	Program	Notes
ORTSDriveWheelWeight	ORTS	
ORTSSteamBoilerType	ORTS	Saturated Superheated
ORTSEvaporationArea	ORTS	
ORTSSuperheatArea	ORTS	
MaxSteamHeatingPressure	ORTS	
ORTSGrateArea	ORTS	
ORTSFuelCalorific	ORTS	
ORTSSteamFiremanMaxPossibleFiring Rate	ORTS	
LPNumCylinders	ORTS	
LPCylinderStroke	ORTS	
LPCylinderDiameter	ORTS	
ORTSSteamGearRatio	ORTS	
ORTSSteamMaxGearPistonRate	ORTS	
ORTSSteamGearType	ORTS	
Steam Advanced		
ORTSBoilerEvaporationRate	ORTS	
ORTSMaxIndicatedHorsepower	ORTS	
ORTSMaxSuperheatTemperature	ORTS	
ORTSSuperheatCutoffPressureFactor	ORTS	
ORTSCylinderPortOpening	ORTS	
ORTSCylinderExhaustOpen	ORTS	
ORTSBoilerEfficiency	ORTS	
ORTSBurnRate	ORTS	
ORTSCylinderInitialPressureDrop	ORTS	
ORTSCylinderBackPressure	ORTS	
Diesel Basic		
WheelRadius	MSTS	
NumWheels	MSTS	
ORTSDriveWheelWeight	MSTS	
Sanding	MSTS	
AntiSlip	MSTS	
MaxDieselLevel	MSTS	

Parameter	Program	Notes
MaxTemperature	MSTS	
MaxOilPressure	MSTS	
DieselUsedPerHourAtMaxPower	MSTS	
DieselUsedPerHourAtIdle	MSTS	
DieselEngineIdleRPM	MSTS	
DieselEngineMaxRPM	MSTS	
DieselEngineMaxRPMChangeRate	MSTS	
MaxPower	MSTS	
MaxForce	MSTS	
MaxVelocity	MSTS	
MaxContinuousForce	MSTS	
MaxCurrent	MSTS	
DieselSmokeEffectInitialMagnitude	MSTS	
DieselSmokeEffectMaxMagnitude	MSTS	
DieselSmokeEffectInitialSmokeRate	MSTS	
DieselSmokeEffectMaxSmokeRate	MSTS	
ORTSWheelSlipCausesThrottleDown	ORTS	
ORTSDieselEngineMaxPower	ORTS	
Diesel Advanced		
ORTSMaxTractiveForceCurves	ORTS	
ORTSCurtius_Kniffler	ORTS	
ORTSInertia	ORTS	
Couplers Basic		
CouplingHasRigidConnection	MSTS	
Coupling	MSTS	Automatic Bar Chain
Type	MSTS	
Spring	MSTS	
Break	MSTS	
r0	MSTS	
Couplers Advanced		
ORTSTensionStiffness	ORTS	
ORTSTensionR0	ORTS	

Parameter	Program	Notes
ORTSTensionSlack	ORTS	
ORTSCompressionStiffness	ORTS	
ORTSCompressionR0	ORTS	
ORTSCompressionSlack	ORTS	
ORTSBreak	ORTS	
Brakes Air		
BrakeEquipmentType	MSTS	handbrake retainer_3_position retainer_4_position vacuum_brake triple_valve graduated_release_triple_valve ep_brake ecp_brake auxilary_reservoir emergency_brake_reservoir distributor
BrakeSystemType	MSTS	Air_single_pipe Air_twin_pipe Vacuum_single_pipe Vacuum_twin_pipe ECP EP Air_piped Vacuum_piped
MaxBrakeForce	MSTS	
MaxHandbrakeForce	MSTS	
EmergencyResVolumeMultiplier	MSTS	
TripleValveRatio	MSTS	
MaxReleaseRate	MSTS	
MaxApplicationRate	MSTS	
MaxAuxilaryChargingRate	MSTS	
EmergencyResChargingRate	MSTS	
EmergencyResCapacity	MSTS	
BrakeCylinderPressureForMaxBrakeBrakeForce	MSTS	
AirBrakesMainMaxAirPressure	MSTS	
AirBrakesCompressor RestartPressure	MSTS	
AirBrakesMainResVolume	MSTS	
TrainPipeLeakRate	MSTS	
TrainBrakesControllerMaxSystemPressure	MSTS	
TrainBrakesControllerMaxReleaseRate	MSTS	
TrainBrakesControllerMaxQuickReleaseRate	MSTS	
TrainBrakesControllerMaxApplicationRate	MSTS	

Parameter	Program	Notes
TrainBrakesControllerEmergencyApplicationRate	MSTS	
TrainBrakesControllerFullServicePressureDrop	MSTS	
TrainBrakesControllerMinPressureReduction	MSTS	
ControlName	MSTS	
ORTSBrakeShoeFriction	ORTS	
BrakePipeVolume	ORTS	
ORTSBrakeShoeFriction	ORTS	
ORTSMainResChargingRate	ORTS	
ORTSBrakePipeChargingRate	ORTS	
ORTSBrakePipeTimeFactor	ORTS	
ORTSBrakeServiceTimeFactor	ORTS	
ORTSBrakeEmergencyTimeFactor	ORTS	
Brakes Vacuum		
BrakeEquipmentType	MSTS	handbrake vacuum_brake
BrakeSystemType	MSTS	Vacuum_single_pipe Vacuum_twin_pipe Vacuum_piped
MaxBrakeForce	MSTS	
MaxHandbrakeForce	MSTS	
MaxReleaseRate	MSTS	
MaxApplicationRate	MSTS	
ORTSNumberBrakeCylinders	MSTS	
VacuumBrakesMinBoilerPressureMaxVacuum	MSTS	
TrainPipeLeakRate	MSTS	
BrakesTrainBrakeType	MSTS	vacuum_single_pipe_eq vacuum_single_pipe
VacuumBrakesSmallEjectorUsageRate	MSTS	
VacuumBrakesLargeEjectorUsageRate	MSTS	
VacuumBrakesHasVacuumPump	MSTS	
TrainBrakesControllerMaxSystemPressure	MSTS	
TrainBrakesControllerMaxReleaseRate	MSTS	

Parameter	Program	Notes
TrainBrakesControllerMaxApplicationRate	MSTS	
TrainBrakesControllerEmergencyApplicationRate	MSTS	
TrainBrakesControllerFullServicePressureDrop	MSTS	
ControlName	MSTS	
ORTSBrakeShoeFriction	ORTS	
BrakePipeVolume	ORTS	
ORTSDirectAdmissionValve	ORTS	
ORTSAuxilaryResCapacity	ORTS	
ORTSBrakeCylinderSize	ORTS	
ORTSBrakePipeChargingRate	ORTS	
ORTSBrakePipeTimeFactor	ORTS	
ORTSBrakeServiceTimeFactor	ORTS	
ORTSBrakeEmergencyTimeFactor	ORTS	
ORTSBrakePipeDischargeTimeMult	ORTS	
ORTSVacuumBrakesMainResVolume	ORTS	
ORTSVacuumBrakesMainResMaxVacuum	ORTS	
ORTSVacuumBrakesExhausterRestartVacuum	ORTS	
ORTSVacuumBrakesMainResChargingRate	ORTS	
Resistance		
CentreOfGravity	MSTS	
ORTSBearingType	ORTS	Roller Low Friction
ORTSDavis_A	ORTS	
ORTSDavis_B	ORTS	
ORTSDavis_C	ORTS	
ORTSBearingType	ORTS	
ORTSDriveWheelWeight	ORTS	
ORTSTrackGauge	ORTS	
ORTSRigidWheelbase	ORTS	
ORTSUnbalancedSuperelevation	ORTS	

Parameter	Program	Notes
ORTSTrackSuperElevation	ORTS	
ORTSSingleTunnelArea	ORTS	
ORTSDoubleTunnelArea	ORTS	
ORTSSingleTunnelPerimeter	ORTS	
ORTSDoubleTunnelPerimeter	ORTS	
ORTSWagonFrontalArea	ORTS	
ORTSDavisDragConstant	ORTS	
ORTSTrailLocomotiveResistanceFactor	ORTS	
Lights		
Type	MSTS	Glow Cone
FadeIn	MSTS	
FadeOut	MSTS	
Cycle	MSTS	
Headlight	MSTS	
Unit	MSTS	
Penalty	MSTS	
Control	MSTS	
Service	MSTS	
TimeOfDay	MSTS	
Weather	MSTS	
Coupling	MSTS	
Duration	MSTS	
LightColour	MSTS	
Position	MSTS	
Radius	MSTS	
Azimuth	MSTS	
Elevation	MSTS	
Transition	MSTS	
Angle	MSTS	
Effects		
StackFX	MSTS	
CylindersFX	MSTS	

Parameter	Program	Notes
WhistleFX	MSTS	
SafetyValvesFX	MSTS	
Exhaust	MSTS	
Cylinders2FX	ORTS	
CompressorFX	ORTS	
GeneratorFX	ORTS	
Injectors1FX	ORTS	
Injectors2FX	ORTS	
HeatingSteamBoilerFX	ORTS	
WagonGeneratorFX	ORTS	
WagonSmokeFX	ORTS	
HeatingHoseFX	ORTS	
WaterScoopFX	ORTS	
TenderWaterOverflowFX	ORTS	
BearingHotboxFX	ORTS	
Freight Animations		
FreightAnim	MSTS	
IntakePoint	MSTS	FuelWater FuelCoal FuelWood FuelSand FuelDiesel FreightGeneral FreightLivestock FreightFuel FreightGrain FreightCoal FreightGrvel FreightSand
ORTSFreightAnims	ORTS	
FreightAnimContinuous	ORTS	
WagonEmptyWeight	ORTS	
EmptyMaxBrakeForce	ORTS	
EmptyMaxHandbrakeForce	ORTS	
EmptyORTSDavis_A	ORTS	
EmptyORTSDavis_B	ORTS	
EmptyORTSDavis_C	ORTS	
EmptyCentreOfGravity_Y	ORTS	
EmptyORTSWagonFrontalArea	ORTS	
EmptyORTSDavisDragConstant	ORTS	
FreightWeightWhenFull	ORTS	

Parameter	Program	Notes
FullMaxBrakeForce	ORTS	
FullMaxHandbrakeForce	ORTS	
FullORTSDavis_A	ORTS	
FullORTSDavis_B	ORTS	
FullORTSDavis_C	ORTS	
FullCentreOfGravity_Y	ORTS	
FullORTSWagonFrontalArea	ORTS	
FullORTSDavisDragConstant	ORTS	
MSTSFreightAnimEnabled	ORTS	
IsGondola	ORTS	
UnloadingStartDelay	ORTS	
FullAtStart	ORTS	
Shape	ORTS	
MaxHeight	ORTS	
MinHeight	ORTS	
ORTSWaterScoopFillElevation	ORTS	
ORTSWaterScoopDepth	ORTS	
ORTSWaterScoopWidth	ORTS	
Passenger Views		
Inside	MSTS	
PassengerCabinFile	MSTS	
PassengerCabinHeadPos	MSTS	
RotationLimit (No Longer Used ?)	MSTS	
StartDirection	MSTS	
Sound	MSTS	
ORTSAlternatePassengerViewPoints	ORTS	
ORTSAlternatePassengerViewPoint	ORTS	
Controllers		
EngineControllers	MSTS	
Throttle	MSTS	
Brake_Engine	MSTS	
Brake_Train	MSTS	

Parameter	Program	Notes
VacuumBrakesHasVacuumPump	MSTS	
NumNotches	MSTS	Dummy TrainBrakesControllerReleaseStart TrainBrakesControllerGraduatedSelfLapLimitedHoldingStart TrainBrakesControllerSuppressionStart TrainBrakesControllerContinuousServiceStart TrainBrakesControllerEmergencyStart
Notch	MSTS	
DirControl	MSTS	
BellToggle	MSTS	
AWS	MSTS	
Sanding	MSTS	
Horn	MSTS	
Wipers	MSTS	
EmergencyStopResetToggle	MSTS	
EmergencyStopToggle	MSTS	
Vigilance	MSTS	
BailOffButton	MSTS	
Regulator	MSTS	
Cutoff	MSTS	
FireDoor	MSTS	
Blower	MSTS	
HeatingTap	MSTS	
Shovel	MSTS	
DampersFront	MSTS	
DampersBack	MSTS	
Injector1Steam	MSTS	
Injector2Steam	MSTS	
Injector1Water	MSTS	
Injector2Water	MSTS	
CylinderCocks	MSTS	
Whistle	MSTS	
BellToggle	MSTS	
SmallEjectorOrCompressor	MSTS	

Parameter	Program	Notes
HeadLights	MSTS	

Appendix II

Consider Transitioning away from MSTS to Open Rails

By the year 2023, MSTS is roughly 23 years old clearly abandoned as a ded-end by its creators. The fact that it has lived on this long is a testament to its community and not specifically how well the program actually worked as it had numerous flaws that never got corrected. Many years after MSTS was introduced and the sequel to MSTS was ultimately cancelled a team of talented programmers decided to build an open source replacement. This is how we got OpenRails

At some point most MSTS users will want to transition away from MSTS to ORTS. ORTS, by design has an advantage over all other current train simulators in that it can utilize a vast amount of the existing legacy MSTS content that has been created over the years. Most content will work without many issues, however, some items may need a bit of tweaking to be able to run correctly in ORTS.

The ORTS Program

The ORTS program installs in a location that will remain totally isolated from simulator content, which is a fundamental difference from the way MSTS works. The benefits of this approach are numerous but the primary benefit is that you will never need to reinstall content when you perform an update or re-installation of ORTS. You can completely delete and reinstall ORTS and then inform it about where your content is and you are ready to go.

You are also able to isolate content as there are some cases where some content installations will affect others. Since content locations can be switched within ORTS, you can eliminate there issues and just select the isolated route installations as needed. This eliminates the need for legacy solutions like Trainstore with MSTS.

Technically, you do not need any MSTS content to use Open Rails but if you have an old MSTS installation installed or saved then hhe beauty of the ORTS solution is that the following items are the only things that have to be copied over from the old MSTS installations:

- GLOBAL folder
- SOUND folder
- ROUTES folder
- TRAINS folder

All other MSTS files are not required and are redundant under ORTS.

As a content developer, you will still need a set of utilities like Aceit, TGATOOLS2 and FFEDIT_SUB to actually access and manipulate content without the MSTS **UTILS** folder.

A typical Open Rails solution that does not depend on a complete MSTS installation could be set up like this:

Installation

Program location	Comment
C:\progam files (x86)\ORTS	Where the PROGRAM gets installed

Content

Program location	Comment
C:\ORTS\TrainSimulations\	Any TrainSimulations Freeware/Payware content (Global, Routes, Sound, Trains)
C:\ORTSWORLD\	Copied over legacy MSTS installation content folders (Global, Routes, Sound, Trains)

Program location	Comment
C:\OR\GNRR	Any custom or 3rd party installations that don't have MSTS dependencies
C:\OR_xxx_	Where *XXX* is any other route that will share (Global, Routes, Sound,Trains) from the OR folder
c:\OR\developer	Any **work in progress routes**
C:\OR\	(Global, Routes, Sound,Trains)

Index

A

Assets
  General Texture Mapping, 44

B

BLENDER
  Level Of Detail, 112
Blender
  Add-Ons, 24
  Blender, SHADING, 32
  Community Add-ons, 26
  Considering Level Of Detail, 89
  Free 3D Graphics Program, 13
  Importing, 94
  Importing Background Images, 98
  Installation, 14
  Installation Video, 15
  Installed via Steam, 14
  Long Term release, LTS, 14
  Modeling, 59
  Move selected objects, 107
  Setup, 16
  Units, Feet, Meters, Metric, Imperial, 31
  Workspace, 67

C

Configuration
  Files, 39
Content Creation
  Basic Overview, 37
Context
  An old but useful text editor, 14

O

Open Rails
  The Open Rails Train Simulator, free and open source project, 8
ORTS
  Example WAG File, 159
  Units, 155
  Universal Settings, 158
  Using INCLUDE, 156

P

Project
  Folder Layout, 35
  Folders, 35
PYTHON
  Example DECAL Layout, 118
  Example Reporting Marks, 119
  FIX UV MAP, 96
Python
  Runing A Script, 143

R

Reference File, 126

T

Texture
  About DDS, 89
  clusion, 132
  clusion Baking, 139
  DDS FILES, 116
  Decals, 118
  DXT COMPRESSION, 116
  LAYERING TECHNIQUE, 122
  Layering Tips, 123
  Seasonal, 126
  Shader Properties, 88
  Snow, 128

W

Windows 7
  No longer supported, 14