TERRA GENESIS · AUTHORING GUIDE

Model Prep Guide

Vehicles, Trailers & Equipment

A practical handbook for getting a Blender model ready for TERRA GENESIS: scale, axes, origins, names, empties, collisions, wheels, hitches, and the checklists to run before you export. Written for builders coming from Farming Simulator 25, using Blender 5.2 and Unreal Engine 5.2.

For Miranda
Game: TERRA GENESIS · Engine: Unreal Engine 5.2 (Linux)
Modeling: Blender 5.2.4 · American spelling
November 2026

Use the standard names in this guide when they fit. When they don’t, make your own — the game is being set up so new attachers, slots, and points can be added in data, without a code change.

Contents

1. How to use this guide
2. Scale and units
3. Axes, export direction, and origins
4. Standard names
5. Custom names
6. Object, mesh, and hierarchy naming
7. Wheels
8. Empties: placement, orientation, and display
9. Attachers, receivers, and trailer chains
10. Collisions
11. Materials, textures, and LODs
12. Vehicles (trucks and pickups)
13. Trailers
14. Equipment and machines
15. Lights
16. Export checklist (Blender 5.2 → Unreal 5.2)
17. Pre-flight checklists
18. Where the game data lives
19. Appendix: FS25 collision boxes, shared wheels, and paths

`WORKS NOW` already in the project and safe to rely on

`PLANNED` agreed and being implemented — use the name now so the export script can pick it up

> A NOTE ON “PLANNED”
>
> Some systems in this guide are already in the game (fifth-wheel to kingpin sockets, the New Equipment window, the current accessory slot types). The empty-export script at `~/TerraGenesis_ModKit/blender_export_empties.py` is ready to use. Others are approved or written but not finished in a playable build yet: data-driven hitch types and chains of up to six trailers (`APPROVED, BEING IMPLEMENTED`), strap and chain tie-downs (`WRITTEN, NOT YET TESTED`), and the New Equipment “Tie points (trailers)” tab (written, not compiled or tested yet). Please use the approved empty names now so the export script and the upcoming hitch code can pick them up.

1. How to use this guide

If you can get a truck, trailer, or machine looking right in Blender, this guide is the rest of the trip into TERRA GENESIS: how big it should be, which way is forward, what to name things, where to put empties, and how collision works in Unreal (it is a little different from Giants / FS25).

Coming from FS25

A lot of your habits still help. Simple box collisions are exactly what we want. Separate wheels, named attach points, and a clean hierarchy all carry over. A few things change:

- **Units.** Model at real factory size in **meters**, scale 1. Do **not** pre-scale the mesh 100×. Unreal's FBX importer converts meters to centimeters for you.
- **Axes.** In Blender, +Y is the front of the vehicle and +Z is up. After the standard Unreal export, that becomes +X forward, +Y right, +Z up in the game.
- **Collision names.** Unreal only picks up custom boxes automatically when they start with `UCX_` (or `UBX_` for plain boxes) followed by the exact mesh name. Your FS25 boxes can keep their shapes; they just need the Unreal names (a script can rename them for you).
- **Wheels must be separate meshes** if they need to spin. A wheel packed into the trailer body cannot be spun later without splitting the mesh.
- **Game data is not written into the .blend.** Empties are exported into a JSON data file by `~/TerraGenesis_ModKit/blender_export_empties.py` (and can also be typed into the New Equipment window). The mesh and the data stay in sync through names and positions.

The three kinds of model

The game treats everything as an **equipment definition**. The same window adds all three, but the mesh work differs a little, so this guide has a section for each:

- **Vehicles** — trucks and pickups that you drive. Usually a skeletal mesh, wheel bones, a fifth wheel or pintle, and accessory slots.
- **Trailers** — towed. A kingpin or other receiver in front, optional attacher at the rear, a deck you can drive onto, tie points, and (ideally) separate wheel meshes.
- **Equipment / machines** — feller bunchers, skidders, loaders, processors, and attachments. Moving arms and booms as separate meshes or bones, cylinder anchors, and tool attach points.

2. Scale and units

Model at real size

Build the truck, trailer, or machine at **factory scale**. A 15.5 m low loader is 15.5 meters long in Blender, not 15.5 centimeters and not 1,550 meters. A steer tire that is 126 cm tall in real life has a radius of 0.63 m in Blender.

> DO NOT PRE-SCALE 100×
>
> In FS25 / Giants it is common to think in meters and then watch a game engine want centimeters. For TERRA GENESIS, leave the mesh in meters. Unreal's FBX importer converts meters to centimeters on import. If you scale the model 100× in Blender first, it will come into the game 100 times too big.

Blender scene settings

- Scene properties → Units: **Metric**, Unit Scale **1.000000**, Length **Meters**.
- Every object's scale should be **1, 1, 1**. When the model looks right, use **Apply Scale** and **Apply Rotation** (Ctrl-A) so there is no hidden transform left on the object.
- Leave Location alone until the origin is where you want it, then you can Apply Location if you need the origin baked.

What "centimeters in the data file" means

The vehicle data (the sidecar `*_ue.json`, the equipment definition, and the New Equipment window) stores positions in **centimeters, in Unreal space** (X forward, Y right, Z up). A hub at 2.52 m in front of the origin and 1.07 m to the left becomes `[252, -107, 83]` in that file. You do not type that by hand if you run the export script on your empties — but if you do, do not multiply by 100 a second time. The reader already expects centimeters.

> ALREADY-CONVERTED PROJECT FILES
>
> Some models already in the project were converted from FS25 and baked into centimeters inside the .blend, with axes already pointed the Unreal way. If you open one of those to edit it, do not scale it or re-bake the axes. The rules in this guide are for **new** models you build at real-world size in meters. When in doubt, drop a 1-meter cube next to the truck: in a meters-based file it should be the size of a meter stick; in a baked centimeters file it will look tiny.

3. Axes, export direction, and origins

Which way is forward?



Figure 1. Blender's +Y is the front of the vehicle. After a standard Unreal FBX export, that becomes Unreal's +X. The sidecar data file records this as forward = X, right = Y, up = Z, in centimeters.

In the game and in every equipment data file, the convention is:

Axis in Unreal	Meaning	In Blender (Z-up)
+X	Forward (the front of the truck)	+Y
+Y	Right	+X
−Y	Left (driver's side on a North American truck)	−X
+Z	Up	+Z

FBX export direction (new models)

For a new model built in Blender at real size, use the standard Unreal settings:

- **Forward:** −Z Forward
- **Up:** Y Up
- **Apply Scalings:** FBX All
- **Scale:** 1.00 · **Apply Unit** on · **Use Space Transform** on

Those settings are what the project's trailer export script uses. After import, the front of the vehicle must point down Unreal's +X, the roof up +Z, and a 1-meter feature must measure about 100 cm. If something comes in sideways or mirrored, fix the export or the import "Force Front X Axis" checkbox — do not scale the mesh to compensate.

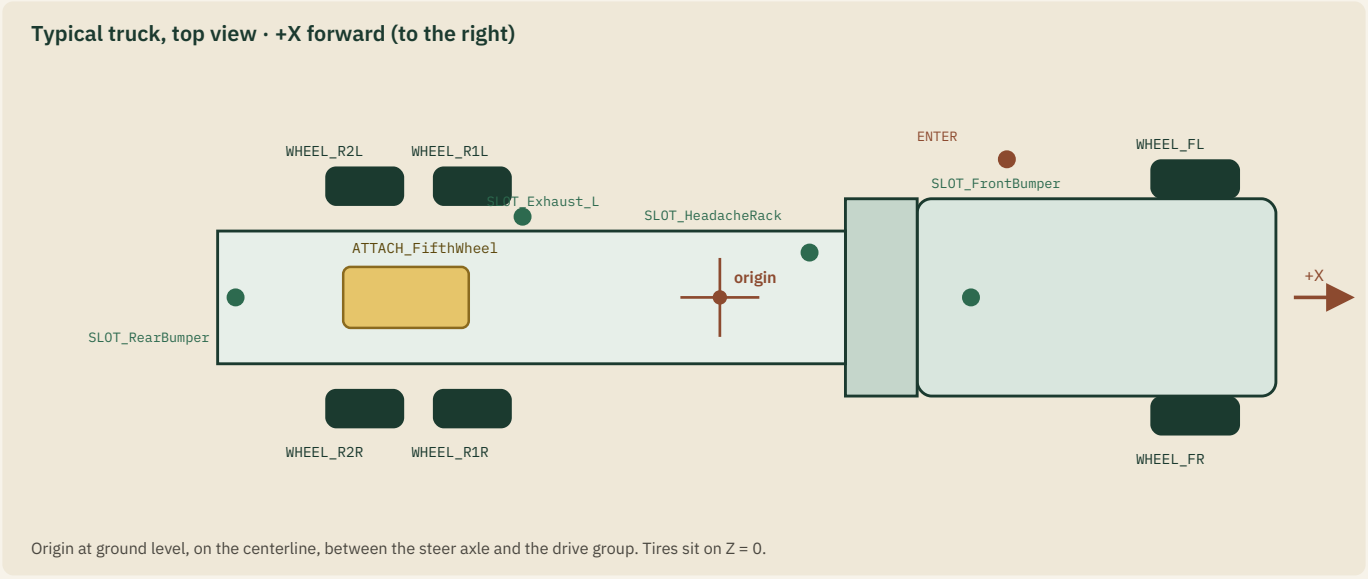


Figure 2. A typical three-axle truck. Put empties at the hub centers, hitch, slots, and the driver's door. The Kenworth T800 in the project follows this pattern: steer axle ahead of the origin, tandem behind it, fifth-wheel plate on the frame.

Where to put the origin (pivot of the whole model)

Kind	Origin placement
Truck or pickup	On the **ground plane** (tires rest on Z = 0), on the **centerline** (Y = 0), **between the steer axle and the drive group**. This is how the Kenworth T800 in the project is set up: steer hubs ahead of the origin, the tandem behind it, fifth-wheel plate on the frame.

Kind	Origin placement
Trailer	On the ground, on the centerline, typically **at or just under the kingpin** (a gooseneck nose a little ahead of the origin is fine). +X is toward the towing vehicle. Place a clearly named kingpin empty — do not make the game guess from the bounding box if you can help it.
Machine	On the ground, on the centerline, under the house or at the mid-wheelbase / mid-track, whichever matches how the machine sits. Articulated machines also need a pivot at the hinge (see Equipment).
Each wheel	At the **hub center**. The tire mesh is centered on that pivot so it spins in place.
Moving part (door, hood, boom, ramp)	At the **hinge or joint**, not in the middle of the panel or the boom.

Unreal +X is also how automatic fallbacks work today: if a trailer has no kingpin socket, the hitch code looks toward the **front of the bounds (+X)** and insets about 40 cm. A well-placed `RECEIVER_Kingpin` empty is always better than that guess.

4. Standard names

Use these names when they fit. They are the shared language between Blender empties, the export script (`~/TerraGenesis_ModKit/blender_export_empties.py`), the New Equipment window, and the game. Write them in this capitalization (CamelCase after the prefix). Matching is case-insensitive, and Blender's `.001` suffixes are dropped, so `ATTACH_FifthWheel.001` still counts as `ATTACH_FifthWheel`.

The export script reads empties whose names start with `TIE_`, `SLOT_`, `ATTACH_`, `RECEIVER_`, or `WHEEL_` (any suffix after the prefix is fine), plus an optional `ENTER` empty and a `ROOT` object for the origin. Positions are written in Unreal centimeters, relative to `ROOT` if it is in the scene, otherwise the world origin.

Wheels

Name the **empties** with the uppercase `WHEEL_` prefix. That is what the export script looks for.

Empty	Where	Notes
`WHEEL_FL` / `WHEEL_FR`	Front axle, left / right	Usually the steer axle on a truck.
`WHEEL_R1L` / `WHEEL_R1R`	First axle behind the front	First drive axle on a typical tandem tractor.
`WHEEL_R2L` / `WHEEL_R2R`	Second axle behind the front	Continue `WHEEL_R3L`, `R4L`, … as needed.
`WHEEL_T1L` / `WHEEL_T1R`	Frontmost trailer axle	Then `WHEEL_T2L`, `T3L`, … down the trailer.

> EMPTIES VS. MESH OR BONE NAMES
>
> Already-converted vehicles in the project (the T800 and others) use **lowercase bone names** on the skeletal mesh, such as `wheel_FL` and `wheel_R1L`. Those bones are how the existing vehicle code spins and steers the wheels. They are not the empties. For new Blender work, put `WHEEL_FL` (and the rest) empties at the hub centers and let the export script write them out. If you also author a new skeleton, you may keep the older lowercase bone names so they line up with current vehicle code — but the empties you export should still be `WHEEL_…`. Wheel *meshes* can be named however you like (for example `Wheel_Steer_L`, or the shared Alcoa assets).

Left is Unreal −Y (Blender −X on a Z-up vehicle). Put the empty at the hub center, not on the ground and not on the outside face of the tire. A second fifth wheel or a second wheel of the same name uses a `_2` suffix, for example `WHEEL_R3L` already distinguishes axles; `ATTACH_FifthWheel_2` is the pattern for two attachers of one type.

Attachers and receivers

The **towing** side (a truck, or the rear of a trailer that can pull another) gets `ATTACH_<Type>`. The **towed** side (the front of a trailer) gets `RECEIVER_<Type>`. A trailer's kingpin is `RECEIVER_Kingpin` — never `ATTACH_Kingpin`. A trailer can hold **both** a receiver in front and an attacher at the rear.

Towing empty (ATTACH_)	Matching receiver (RECEIVER_)	What it is	Status
`ATTACH_FifthWheel`	`RECEIVER_Kingpin`	Semi fifth-wheel plate and trailer kingpin	Coupling by the existing `FifthWheel` / `Kingpin` sockets WORKS NOW. These empty names are the approved standard. Multi-type hitching and chains BEING IMPLEMENTED
`ATTACH_Pintle`	`RECEIVER_Lunette`	Pintle hook and lunette ring (including a converter dolly's front)	The Kenworth 521 has a pintle point (data / component). Multi-type hitch gameplay BEING IMPLEMENTED
`ATTACH_Ball`	`RECEIVER_Coupler`	Ball hitch and coupler (pickups, light trailers)	BEING IMPLEMENTED
`ATTACH_Gooseneck`	`RECEIVER_GooseneckCoupler`	Gooseneck hitch and gooseneck coupler	BEING IMPLEMENTED
`ATTACH_Pole`	`RECEIVER_PoleReach`	Pole truck / reach connection	BEING IMPLEMENTED

Those five pairs are the built-in standard. There is no Drawbar or Clevis type in the list. A **converter dolly** is just a short trailer with `RECEIVER_Lunette` at the front and `ATTACH_FifthWheel` at the rear. A **Super B lead trailer** has `RECEIVER_Kingpin` at the front and `ATTACH_FifthWheel` at the rear. Need a second attacher of the same type? Add `_2`, `_3`, and so on (`ATTACH_FifthWheel_2`).

Today a truck with "Has Fifth Wheel" on can still couple a semi at the `Kingpin` socket, and rear hitch type (for example `Pintle`) can already be stored on the equipment definition. Put the approved empties on every new model now; the data-driven hitch list will read them.

Accessory slots TYPES WORK NOW

These are the slot types the game already knows. A truck can have any number of slots, including several of the same type (two toolboxes, left and right exhaust, many light pads). Give each slot its own empty and its own name, starting with `SLOT_`.

Standard empty	Slot type in the game	Typical use	Notes
`SLOT_HeadacheRack`	`HeadacheRack`	Rack behind	Optional.

Standard empty	Slot type in the game	Typical use	Notes
		the cab	
`SLOT_Exhaust`	`Exhaust`	Stacks, one mesh or split	Required on current trucks. Both stacks can live in one mesh, or use `SLOT_Exhaust_L` and `SLOT_Exhaust_R` as two Exhaust slots. WORKS NOW
`SLOT_FrontBumper`	`FrontBumper`	Front bumper	Required. Split parts should stay in place at 0,0,0 relative to the body.
`SLOT_RearBumper`	`RearBumper`	Rear bumper	Required.
`SLOT_BedOrDeck`	`BedOrDeck`	Service body, flat deck, etc.	Optional. Parts can also store capacity (kg) and deck length (cm); cargo gameplay is not using those yet.
`SLOT_Toolbox`	`Toolbox`	Frame or deck toolbox	Optional. Use `SLOT_Toolbox_L` / `_R` when you have more than one.
`SLOT_Lights`	`Lights`	Light bars, markers, beacons	Type exists. No base truck has a Lights slot yet. Mods can already add one.
`SLOT_Visor`	Visor (new type)	Cab visor / sun visor	PLANNED as a standard type. Until it ships, do not invent a different meaning for this name.

The Modder Kit already ships placeholder blocks at `~/TerraGenesis_ModKit/Generated/Blocks/SLOT_<Type>.fbx`. Each block is the right size, with a bump that points forward (+X) and the pivot at the slot origin. Snap your part to the block, then delete the block before you export.

Tie points WRITTEN, NOT YET TESTED

Empty / socket	Where	Notes
`TIE_01`, `TIE_02`, `TIE_03`, …	Along both deck edges, where a strap or chain would land	Any suffix after `TIE_` is fine (`TIE_01`, `TIE_Bunk_L1`). Number from the front, left then right, or clockwise — just be consistent. Mesh sockets named `TIE_*` or `TieDown*` are also read. If none are set, the game can place automatic points along the deck; named empties are better.

Tie points can come from, in this spirit: the New Equipment window's **Tie points (trailers)** tab (the tab is written, not compiled or tested yet), `tie_points` in a mod vehicle JSON, `TIE_` empties from the export script, mesh sockets `TIE_*` / `TieDown*`, or automatic deck-edge points if nothing else is set.

Play plan WRITTEN, NOT YET TESTED: **F** on foot at a point does up or undoes that one point. **F** in the cab does up or undoes every point at once. Chains are for machines and trucks; straps are for cargo. When every point on a load is done, the load is locked to the deck. There is no warning and no speed limit for an unsecured load.

Lights RECOMMENDED EMPTY NAMES

The export script does not yet collect `LIGHT_` empties (it reads `TIE_`, `SLOT_`, `ATTACH_`, `RECEIVER_`, and `WHEEL_`). The vehicle already has light *sockets* you can list on the equipment definition. The empty names below are still the standard to put in Blender so a later pass, or a hand edit, can fill those lists.

Empty	Use
`LIGHT_Head_L` / `LIGHT_Head_R`	Low beam / main headlights
`LIGHT_High_L` / `LIGHT_High_R`	High beam
`LIGHT_Tail_L` / `LIGHT_Tail_R`	Tail / stop
`LIGHT_Turn_FL` / `FR` / `RL` / `RR`	Turn signals
`LIGHT_Reverse_L` / `LIGHT_Reverse_R`	Backup lights
`LIGHT_Marker_*`, `LIGHT_Clearance_*`	Clearance and marker lights
`LIGHT_Work_*`, `LIGHT_Beacon_*`	Work lamps and beacons
`LIGHT_Running_*`	Any extra running lamp

Other useful names

Empty or object	Use	Status
`ROOT`	The export origin. Put an object named `ROOT` at the vehicle origin (ground, centerline, the same pivot as in section 3). Every empty is measured from this. If it is missing, the script uses the world origin, which is only right if your model already sits there.	Read by `blender_export_empties.py` `WORKS NOW`
`ENTER`	Where the player walks up to enter. Sit it by the driver's door, near ground or step height. The export script looks for this exact name.	Enter points already exist on the definition. The `ENTER` empty is what the script writes.
`CYL_<Name>_Barrel`, `_Rod`, `_BarrelAnchor`, `_RodAnchor`	Hydraulic cylinder meshes and the two pivot points they stretch between	`WORKS NOW` as component tags / bones (see the T800 steering rams and forestry machines). Not read by the empty-export script.
`TOOL_<Name>`	A working tool's attach or pivot (head, grapple, winch fairlead)	Moving tools already target a bone or a component tag. Not read by the empty-export script.

5. Custom names

The standard list is a starting point, not a cage. If you are building a logging bunk, a hay spike, a spreader bar, or a hitch that isn't one of the five built-in pairs, you can make up a clear name and register it in data. You should not need a code change.

How to invent a name

- Keep the prefix so the export script and the game know what the empty is for: `ATTACH_` (towing), `RECEIVER_` (towed), `SLOT_`, `WHEEL_`, `TIE_`. `ROOT` and `ENTER` are fixed names, not prefixes. `LIGHT_`, `CYL_`, and `TOOL_` are useful in the .blend even though the current export script does not collect them.

- After the prefix, use **letters, digits, and underscores only**, **1 to 32 characters**. No spaces. CamelCase is recommended: `ATTACH_LoggingBunk`, `SLOT_Winch`, `TIE_Bunk_L1`, `RECEIVER_LoggingBunk`.
- Matching is **case-insensitive**. Blender's `.001`, `.002` suffixes are dropped, so a duplicated empty is not a new type.
- To have more than one of the same type, add `_2`, `_3`, and so on: `ATTACH_FifthWheel_2`, `SLOT_Toolbox_2`.
- Make the name describe the joint or the part, not the truck. `ATTACH_Pintle` can live on many vehicles; `ATTACH_MyT800Thing` should not.
- Do not reuse a standard name for a different meaning. If you need something new, pick a new word.
- For a custom type, pair `ATTACH_YourName` with `RECEIVER_YourName` (same type word on both sides) and list what it can couple to in the hitch-type data.

Where you define it APPROVED, BEING IMPLEMENTED

Custom hitch types are declared in one of two places, with no C++ work:

- **New Equipment window** — the **Hitch types** list (Tools → TERRA GENESIS → New Equipment...).
- **Mod JSON** under `Mods/<mod>/Hitches/*.json`. A file holds a `hitch_types` list. Each type has a `name`, a `side` (towing, towed, or both), `compatible` types it may couple to, yaw / pitch / roll limits, a couple radius, and a joint style. Vehicle files in `Vehicles/*.json` get `attachers` and `receivers` arrays (the export script already writes those keys).

> HITCH-TYPE FILE (APPROVED SHAPE; BEING IMPLEMENTED)
>
> A type record looks like: name `LoggingBunk`, side, compatible list (for example `LoggingBunk`), yaw / pitch / roll limits in degrees, couple radius in centimeters, and joint style. A Super B lead does not need a new type: it uses the built-in pair, with `RECEIVER_Kingpin` at the front and `ATTACH_FifthWheel` at the rear. A converter dolly uses `RECEIVER_Lunette` in front and `ATTACH_FifthWheel` at the rear. Until the hitch-type hook ships, still place the empties and, for a custom type, add the JSON so it is ready.

What already works today, without a new type

- **Custom slot names of an existing type.** You can add `SLOT_Toolbox_L` and `SLOT_Toolbox_R` as two slots of type `Toolbox`, or thirty `Lights` pads, each with its own location. The slot's *name* is free; the *type* must be one of the types in section 4 until custom slot types ship.
- **Extra wheel empties** following the R-series or T-series pattern. There is no fixed axle cap in the naming.
- **As many `TIE_` points as the deck needs**, with any suffix.

> UNTIL CUSTOM HITCH TYPES SHIP
>
> A new slot *type* (something that is not HeadacheRack, Exhaust, FrontBumper, RearBumper, BedOrDeck, Toolbox, or Lights — and Visor once it is added) will not appear in the store or workshop yet. A new hitch type will not couple in the current playable build. Putting the empties and a `Hitches/*.json` record in the mod is still the right work. This hitch-type system is **approved and being implemented**.

6. Object, mesh, and hierarchy naming

Unreal and the export scripts are much happier with plain, unique names: letters, digits, and underscores. Avoid spaces in names you want the game to find (`5th wheel` on a converted truck works only because a special-case script looks for it — don't add more of those).

Suggested mesh names

Kind	Example	Notes
Truck / machine body (skeletal)	`T800_Body`, `959M_Body`	One body mesh parented to the armature.
Trailer or static body	`LowLoader_Deck`, `TalbertRGN_Body`	Name the visible mesh; collision boxes follow it.
Split parts	`T800_FrontBumper`, `T800_Exhaust`	Exported from the **same file at the same scale and origin** as the body, so they line up at 0,0,0.
Wheels	`Wheel_Steer_L`, or reuse the shared Alcoa meshes	Separate object per wheel, or a shared left/right pair instanced at each hub.
Trailer ramps, necks, bolsters	`front_bolster`, `LowLoader_Ramp_L`	Whatever you use must be the exact name after `UCX_`.
Collision box	`UCX_front_bolster_01`	See Collisions.

Hierarchy

- Armature (if skeletal) at the vehicle origin, rest pose already correct, no leftover rotation on the armature object.
- Body mesh parented to the armature (skeletal) or sitting at the origin (static).
- Wheels as separate meshes — not joined into the body.
- Split accessories (bumper, visor, exhaust, rack) as their own objects, still in the same scene position they had on the truck. When they are exported alone, they keep that same world position so a 0,0,0 slot places them correctly. This is how the T800 bumper and exhaust split is set up.
- Empties either at the root or parented to the part they belong to (a boom empty should follow the boom). The export script writes each empty's position **relative to `ROOT`** (or the world origin if `ROOT` is missing), so parenting is for your sanity, not a requirement.
- Collision boxes parented to, or sitting with, the mesh they describe.

> SPLIT PARTS MUST SHARE SCALE AND ORIGIN
>
> If you export the bumper from a file that was moved, scaled, or has a different origin than the body, it will not sit on the truck. Always export split parts from the same Blender file (or a perfect copy), at the same scale, with the same origin. Leave them where they were on the body; do not move a bumper to 0,0,0 in Blender just to "clean it up" unless you also record that offset in the slot.

7. Wheels

Wheel pivots · one mesh per wheel, origin at the hub center



- Export each wheel as its own mesh (or its own object in the same FBX).
- Do not bake wheels into the body or trailer mesh if they need to spin.
- Left and right can be separate meshes (the Alcoa sets are L and R).
- The tire's lowest point should sit on the ground (Z = 0) when the empty is placed.
- Shared Alcoa wheel files: reuse them, do not edit them.

Figure 3. The low loader and Talbert RGN currently have wheels baked into one mesh, so those wheels cannot spin until the meshes are split. New trailers should ship with separate wheel meshes from the start.

Rules that matter in-game

- **One wheel mesh per corner** (or a shared L/R mesh instanced at each empty). Duals can be one mesh that includes both tires, still pivoted at the hub, or two meshes — pick one style and stick to it.
- **Pivot at the hub center.** Apply transforms so the wheel object's origin is the spin axis.
- **Do not bake wheels into a trailer or body mesh** if they should spin. The current low loader and Talbert RGN have baked wheels, so those wheels stay still. New models should not repeat that.
- **Left and right.** The shared Alcoa sets have separate left and right meshes. Face the outside of the wheel outward. If a mesh comes in facing inward, the wheel set has an extra-yaw field — but it is better to fix the mesh.
- Converted wheel FBXs in this project often have their axle along X and use a 90° yaw in the wheel-set data so Unreal's wheel axle (+Y) lines up. If you model a new wheel, you can either match that or model with the axle already on Y and leave the extra rotation at 0. Check a test import; do not guess twice.

Shared Alcoa wheels — do not modify

The project has a shared semi wheel set. Reuse it on new trucks and trailers when the size is right. **Do not edit these files** to suit one model; you would change every truck that uses them.

Asset	Path (in the project)
Wheel set	`/Game/Wheels/Semi/DA_WheelSet_Semi_Alcoa`
Steer L / R	`Content/Wheels/Semi/Steer/SM_Semi_Steer_alcoa_L` (and `_R`)
Drive L / R	`Content/Wheels/Semi/Drive/SM_Semi_Drive_alcoaAluminum_L` (and `_R`)

If a truck needs a different size or style, make a **new** wheel set and new meshes. Leave the Alcoa files alone. Hide any tires that are modeled into the body (the wheel set can hide named bones or material slots) so you do not see two tires in the same place.

The game does not use engine motion blur (it is off project-wide), so a very low-segment tire will look faceted at speed. Give the tread and rim enough segments to read as round, but there is no need for a race-sim wheel.

8. Empties: placement, orientation, and display

Empties are how you tell the game where wheels, hitches, slots, ties, lights, doors, and cylinders are, without building that into the mesh. Name them from the standard (or custom) list, leave them at factory scale (1,1,1), and include a `ROOT` object at the vehicle origin. Run `~/TerraGenesis_ModKit/blender_export_empties.py` to write the JSON (you can also pack the empties into the FBX).

Display tips in Blender

- Viewport display: **Plain Axes** or **Arrows**, size about **0.2 to 0.5 m** so you can see them next to a truck but they don't fill the screen.
- Put empties in a collection named `Empties` or `Sockets`. Hide the collection from render; keep it visible while you work.
- A different color or a short custom shape per family (wheels, attachers, ties) helps when the deck is crowded.
- Lock scale. If you accidentally scale an empty, Apply Scale or reset it to 1.

Orientation rules

- Align each empty to the vehicle: no random tilt. After import, its local +X should match the vehicle's forward, +Z up, unless the point has a real facing of its own.
- **Headlights and work lights** should shine forward (Unreal +X, which is Blender +Y before export). Point the empty's local forward that way.
- **Tail lights** face rear (Unreal –X).
- **Attachers and receivers** sit on the coupling: the fifth-wheel empty on the plate, at hook height; the kingpin empty on the pin axis, at the height the plate should meet. A pintle empty sits in the throat of the hook. Rotation 0,0,0 in vehicle space is the default unless the hook itself is rotated.
- **Slots** use the same space as the truck (cm, X forward). The ModKit block's bump points forward. A part modeled on that block, with the same pivot, drops in cleanly. The T800 factory bumper and exhaust use a zero offset because the part was cut from the body and left in place.
- **Tie points** sit on the deck surface, just inboard of the side rail, where a hook would land — not floating a meter in the air and not buried under the deck.
- **Wheel empties** at hub center, with the empty's axes lined up with the vehicle (spin is handled by the game).

The export script writes each empty's position (and rotation when it matters) into JSON, already converted to Unreal centimeters, relative to `ROOT`. It collects `TIE_`, `SLOT_`, `ATTACH_`, `RECEIVER_`, and `WHEEL_` empties, plus `ENTER`. Run it in Blender with `blender my_model.blend --background --python ~/TerraGenesis_ModKit/blender_export_empties.py -- out.json`, or paste the script into the Text Editor and press Run (it writes `<blend name>_empties_ue.json` next to the .blend). You can also enter positions in the New Equipment window (slots, Hitch types, Tie points (trailers), fifth-wheel rest height, rear hitch location).

9. Attachers, receivers, and trailer chains

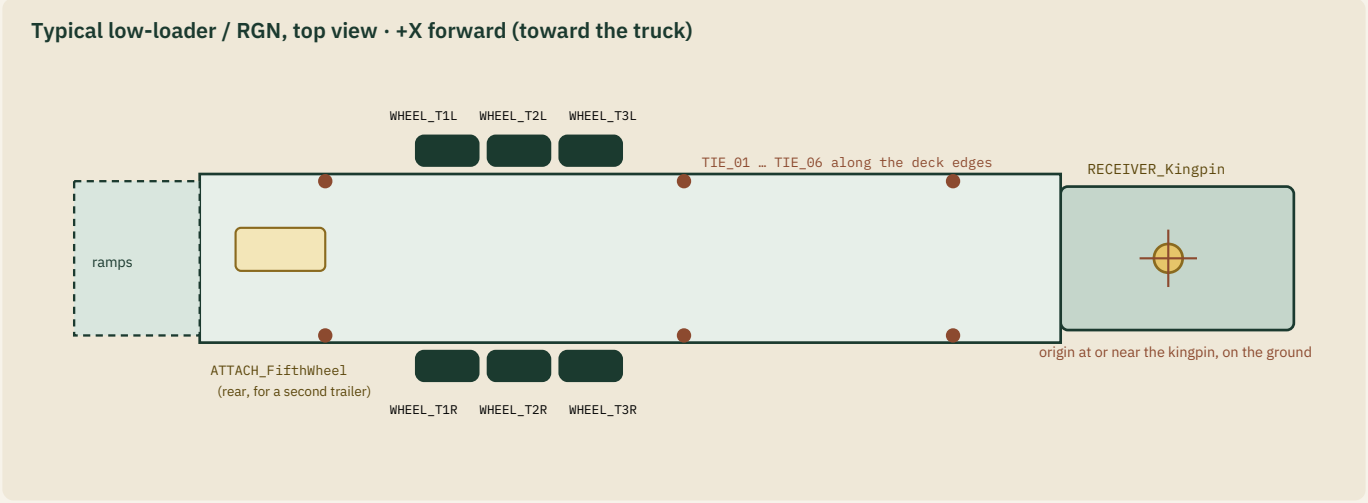


Figure 4. A trailer can carry a receiver at the front (kingpin) and its own attacher at the rear, so a second trailer can hook to it — or hook straight to a truck. Tie empties sit on the deck edges where straps and chains land.

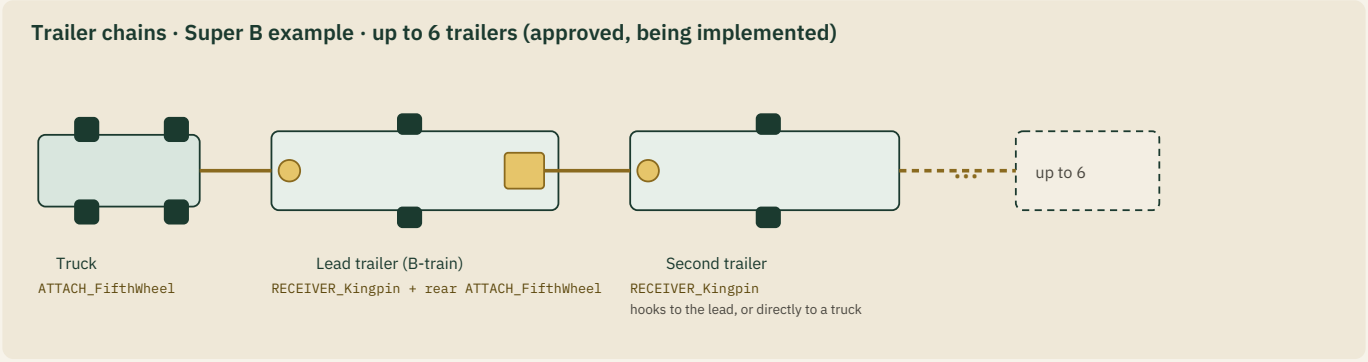


Figure 5. A Super B (B-train): the lead trailer has RECEIVER_Kingpin in front and ATTACH_FifthWheel at the rear. The same second trailer can also couple straight to a truck's fifth wheel. A converter dolly uses RECEIVER_Lunette plus a rear ATTACH_FifthWheel. Chaining up to six trailers is approved and being implemented.

What works today

- A truck with **Has Fifth Wheel** on can couple a semi trailer. The plate is a slidable fifth-wheel component; the trailer is held at a `Kingpin` socket (or a tagged component, or a bounds guess). Put `ATTACH_FifthWheel` and `RECEIVER_Kingpin` empties on new models anyway.
- The hitch key is **H**. Once chaining ships, H couples the nearest compatible free pair anywhere in the chain; if nothing is in reach, H unhooks the last trailer. Brakes, lights, and ABS pass down the chain.
- The Kenworth 521 has **Has Fifth Wheel** off and a pintle point at the rear (from `SOCKET_Attacher_PanelHitch` in its converted file). It cannot pull a fifth-wheel low loader or RGN. Rear hitch type can already be stored as a name such as `Pintle`. Use `ATTACH_Pintle` on the truck and `RECEIVER_Lunette` on the trailer.

What to model for every new truck or trailer

- Place the approved empties even if a type's gameplay is still being implemented. A pole trailer should have `RECEIVER_PoleReach` at the front (or `RECEIVER_Lunette`, if that is the real joint) and, if it can lead another trailer, a rear `ATTACH_…` as well.
- Height matters. Measure the real fifth-wheel or pintle height and put the empty there. A kingpin that is 20 cm too high will stand the trailer on its nose or hang it in the air.
- If the fifth wheel slides, the rest position is the empty (or the rest position in data). Slide limits are stored in centimeters along X (the T800 slides roughly from −598 to −346 cm, resting near −511).

Chains of trailers — Super B, dollies, and up to six APPROVED, BEING IMPLEMENTED

The rule for new trailers: **a trailer may have a receiver in front and its own attacher at the rear**. That is how a Super B (B-train) works — the lead trailer has `RECEIVER_Kingpin` and a rear `ATTACH_FifthWheel`; the second trailer's `RECEIVER_Kingpin` can hook to the lead *or* directly to a truck. A converter dolly is the same idea with different types: `RECEIVER_Lunette` in front and `ATTACH_FifthWheel` at the rear, so a pintle truck can pull a fifth-wheel trailer.

The game will allow a chain of **up to six trailers**. **H** couples the nearest compatible free pair anywhere in the chain. If nothing is in reach, **H** unhooks the last trailer. Brakes, lights, and ABS pass down the chain. This behavior is **approved and being implemented**.

New hitch types (a logging bunk pin, anything we have not listed) follow section 5: `ATTACH_YourName` on the towing side, `RECEIVER_YourName` on the towed side, declared in `Mods/<mod>/Hitches/*.json` or the New Equipment Hitch types list, with no code change.

10. Collisions

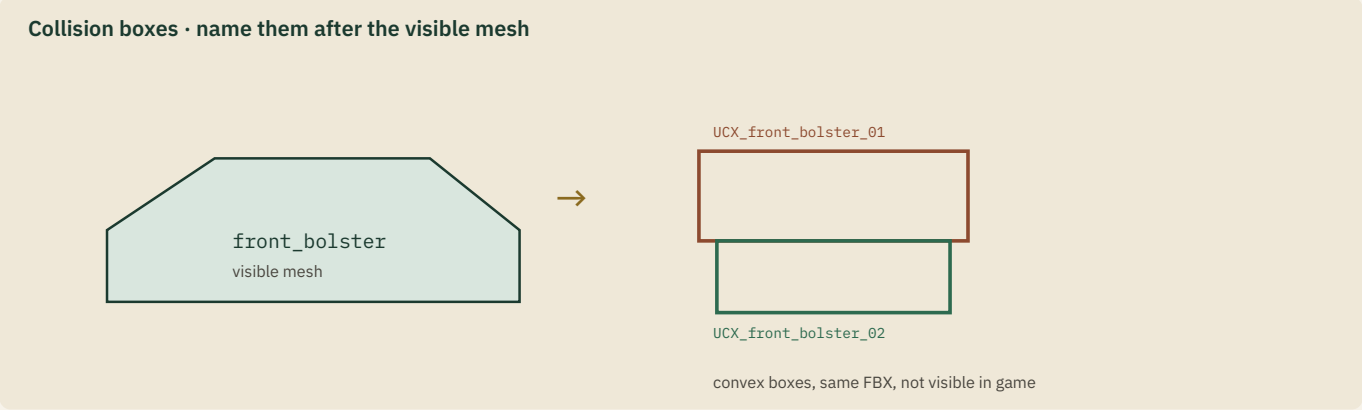


Figure 6. The part after UCX_ must match the visible mesh name exactly, then _01, _02, and so on. UCX_front_bolster_bottom only works if the mesh itself is named front_bolster_bottom.

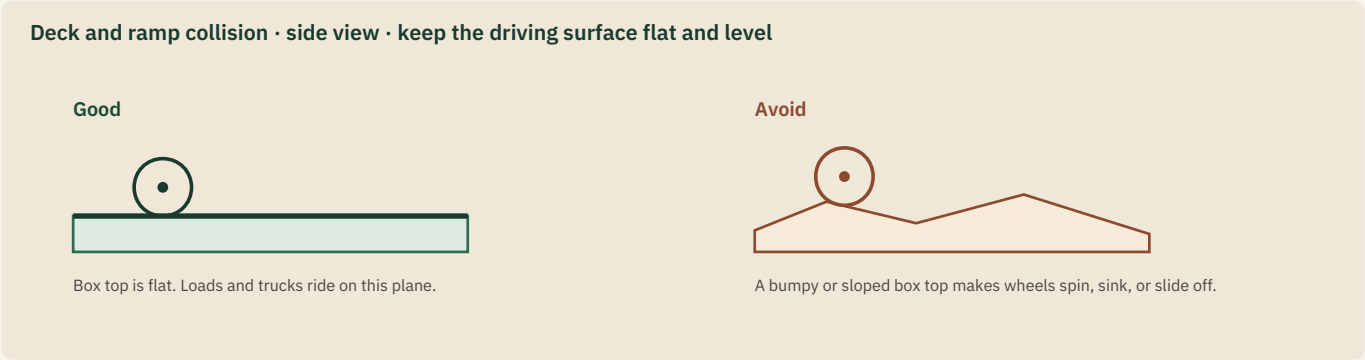


Figure 7. Collision, not the visible wood or diamond plate, is what the game drives on. Make deck and ramp box tops flat, level, and a hair above the visual boards if you need to hide gaps.

Keep the simple boxes

Your FS25 habit of building a few boxes, not a high-poly "accurate" collision mesh, is the right one. Unreal wants **convex** primitives. They are faster and they behave more predictably than using the visible model as collision — and a decorative bumper or a chain should not be what a tire drives on.

Naming (static meshes, including most trailers, decks, ramps, and accessories)

Put the boxes in the **same FBX as the visible mesh**. Unreal reads them automatically when the name matches this pattern:

Prefix	Shape	Example
`UCX_`	Convex mesh (the usual choice; a box is fine)	`UCX_front_bolster_01`
`UBX_`	Box	`UBX_LowLoader_Deck_01`
`USP_`	Sphere	`USP_Tank_01`
`UCP_`	Capsule	`UCP_Rollbar_01`

- The text after the prefix must be the **exact visible mesh name**, then `_01`, `_02`, … If the mesh is `front_bolster`, the boxes are `UCX_front_bolster_01`, `UCX_front_bolster_02`. `UCX_front_bolster_bottom` is not picked up unless the mesh is literally named `front_bolster_bottom`.
- Each box must be **convex** (a regular Blender cube is). Do not use a C-shaped or hollow mesh as one UCX piece — split it into convex bits.
- **Deck and ramp tops must be flat and level** along the driving surface. That plane is what trucks and loads ride on. A wavy or sloped box top will spin tires, drop a load through the wood, or throw a machine off the trailer.
- Skip collision on small details: lights, chains, air lines, mudflaps, mirrors, steps if they are only visual. They cost performance and they snag vehicles.
- Do not leave the collision boxes visible (no material you care about, or put them in a collection that is not rendered). Unreal will hide them as collision.

Skeletal meshes (typical trucks and machines)

A skeletal body does **not** use `UCX_` boxes in the same automatic way a static mesh does. After import, Unreal gives it a **Physics Asset**: simple bodies (boxes, capsules, spheres) parented to bones — chassis, cab, hood, each boom

section, and so on. Wheels are usually handled by the vehicle wheel system, not by a fat physics body on the tire mesh.

- You still want simple shapes, one per moving part. A single giant body on the root bone will fight the wheels and the hitch.
- Keep cab and frame boxes inside the visible metal so doors and the ground do not jitter.
- You can author hint boxes in Blender for yourself, but the Physics Asset in Unreal is what the game runs. Budget time after import to check it.

FS25 boxes you already have

Keep them. An export or import script can rename a collection of FS25 collision boxes to `UCX_<Mesh>_01` and so on. If you would like that script in your Blender file, ask and we can add it so you never have to rename boxes by hand.

11. Materials, textures, and LODs

Material slots

Give every material slot a stable, readable name before you export. The configurator and any hide-body-tire logic look at slot names. Good examples: `M_Paint`, `M_Chrome`, `M_Glass`, `M_Rubber`, `M_Lights`, `M_Interior`, `M_Decal`. Avoid `Material`, `Material.001`, and empty slots you do not use.

If a part should take the truck's paint color, put that surface on a dedicated paintable slot. Glass, chrome, and rubber stay on their own slots so a paint job cannot tint the windshield.

UVs and textures

- Unwrap to the 0–1 UV square. Mirroring is fine. Wild overlaps that were only for Giants shaders may look wrong in Unreal.
- Use power-of-two textures (2048, 4096, …). A typical set is Base Color, Normal, Roughness, and Metalness (or a packed ORM). If you bake a normal map in Blender (OpenGL), say so — Unreal expects DirectX normals unless the material flips the green channel.
- Do not embed **absolute** computer paths in the FBX (an older Talbert export did this and it breaks on another machine). Keep textures in a folder next to the file, or use Relative path mode.

Motion blur

Engine motion blur is **off** in TERRA GENESIS. Do not rely on blur to hide a chunky tire or a sharp shadow on a wheel. Model the rim and tread so they still look all right when the truck is moving.

LODs and polycount

LODs are optional but welcome. There is no hard cap, but new models should aim lower than the heaviest converted trucks if you can (the T800 body LOD0 is about 239,000 faces — that is a lot). Practical targets:

Piece	A healthy range (triangles)
Truck or machine body, LOD0	Try to stay under about 150,000 if it still looks good

Piece	A healthy range (triangles)
LOD1 / LOD2	Roughly 30–40% and 10–15% of LOD0
One wheel	5,000–20,000 is plenty
Accessory (bumper, rack, toolbox)	A few thousand to about 30,000
Collision	A handful of boxes, not a dense mesh

12. Vehicles (trucks and pickups)

Driveable vehicles are usually a **skeletal mesh** with wheel bones, a body, opening parts, and one or more hitch empties. Copy a close existing definition (T800 for a highway tractor, 521 for a pintle / pole truck) in the New Equipment window so physics, wheels, and the vehicle class come along.

What to include

- Body at the vehicle origin, tires on the ground.
- Wheel empties `WHEEL_FL`, `WHEEL_FR`, `WHEEL_R1L`, … (the export script's prefix). If you author an armature, existing project vehicles use lowercase bone names such as `wheel_FL` — see section 4. Doors and hood as separate bones if they open: the T800 uses `Door_L`, `Door_R`, `Hood`, with the pivot at the hinge.
- Hitch empties: `ATTACH_FifthWheel` and/or `ATTACH_Pintle`, `ATTACH_Ball`, and so on. Turn **Has Fifth Wheel** off in the definition when the truck truly has no plate (the 521).
- Slot empties for every place a part can go. Required slots on current semi trucks are Exhaust, Front Bumper, and Rear Bumper. If a part is still modeled into the body, mark the slot **Baked** until you split the mesh.
- A `ROOT` object at the vehicle origin, and an `ENTER` empty at the driver's door.
- Light empties (sections 4 and 15).
- A Physics Asset after import, with simple bodies on the chassis and cab — not one body that covers the tires.

Pickups

Same rules, usually a ball or pintle and a `SLOT_BedOrDeck`. Pickup and pickup-trailer categories already exist as store tabs. Use `ATTACH_Ball` with `RECEIVER_Coupler`, or `ATTACH_Pintle` with `RECEIVER_Lunette`, as they fit. Those hitch types are being implemented; the empties should still go in.

13. Trailers

Trailers are towed. Their vehicle class must not be a driving class (Validate All will warn). A good starting copy in the editor is the Talbert RGN / low-loader definition, then change the mesh and the points.

What to include

- A clear front receiver at the real pin or ring: `RECEIVER_Kingpin`, `RECEIVER_Lunette`, `RECEIVER_Coupler`, `RECEIVER_GooseneckCoupler`, or `RECEIVER_PoleReach`. Do not rely on the automatic "front of the box" guess if you can help it.
- If this trailer can lead another, a rear attacher: Super B leads use `RECEIVER_Kingpin` plus rear `ATTACH_FifthWheel`; a converter dolly uses `RECEIVER_Lunette` plus rear `ATTACH_FifthWheel`.

- Separate wheel meshes and `WHEEL_T1L` … empties at the hubs.
- Flat, level collision on the deck and on every ramp. Ramps that fold should be their own meshes, with the pivot at the hinge (the low-loader ramp fold is a moving tool; the fold key in the current build is **X**).
- `TIE_01`, `TIE_02`, … along both deck edges. Put as many as the deck needs so a truck or a machine can be chained at the corners.
- Bolsters, bunks, and headboards as named meshes with their own UCX boxes if a log or a load should rest on them.

Loads on the deck

A hauled truck or machine has to ride the *deck*, not the ground under the trailer. That ride-along behavior is being finished in code. Your job on the model is: a flat deck collision, a ramp a vehicle can climb, and tie points. With all straps or chains done, the load will lock to the trailer; until then it can still slide on hard braking, which is intended (and there is no warning light for an unsecured load). This tie-down behavior is written, not yet tested.

14. Equipment and machines

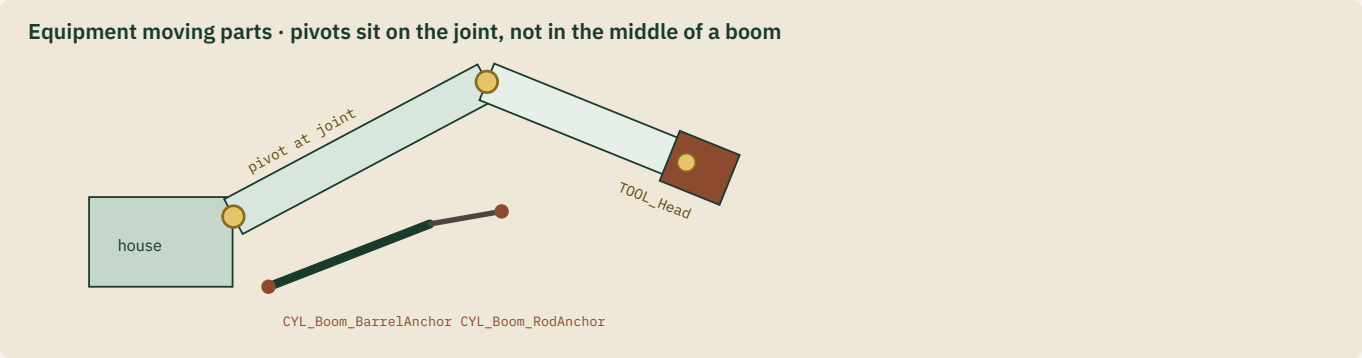


Figure 8. Each arm, boom, stick, and tool is a separate mesh or bone. The object's origin (or bone pivot) is the hinge. Cylinder empties mark the barrel and rod anchor points so the ram can stretch between them.

Copy the closest existing machine definition (skidder, feller buncher, log loader, processor, excavator) so tracks or wheels, mass, and moving tools come with it. Then replace the mesh and retarget the bones or tags.

Moving parts

- House, boom, stick, head, grapple, blade, winch drum — each is a **separate mesh or bone**.
- The origin of that part (or the bone pivot) is the **joint**, not the middle of the tube.
- Rest pose should be a natural parked pose. Angle limits live on the moving-tool data (min/max degrees or translation), not in the mesh.
- Articulated machines (skidders, some loaders) also need an articulation pivot — a bone or empty at the hinge, on the centerline. The definition already has an articulation block (min/max degrees, pivot in cm, default bone name `articulation`).

Hydraulic cylinders

A cylinder is a barrel mesh, a rod mesh, and two anchors. The game looks at the anchors each frame and slides the rod so the visible length matches the distance (clamped to the stroke). Name them in a family, for example:

- `CYL_Boom_Barrel` / `CYL_Boom_Rod` — the two meshes
- `CYL_Boom_BarrelAnchor` / `CYL_Boom_RodAnchor` — empties (or bones) at the pins

Those become component tags or bone names on the definition (the converted T800 steering rams already follow a `…_Barrel`, `…_Rod`, `…_BarrelAnchor`, `…_RodAnchor` pattern). Put the anchors on the parts that actually move, so the ram follows the boom.

Tool attach points

If a head, bucket, or grapple can be swapped or is a separate attachment, give it a `TOOL_Head` (or a custom `TOOL_…` / `ATTACH_…`) at the pin. Attachments are their own equipment category and definition; the parent machine holds the matching point.

Tracks

Tracked machines use track contact points (position, radius, width, left/right) rather than wheel hubs. If you are converting from FS25, the existing pipeline can fill those from the sidecar. For a new tracked model, place contact empties along the ground path and we can wire them into the definition the same way as hubs.

15. Lights

Two different things are easy to mix up:

- **Light placement empties** (`LIGHT_Head_L`, and so on) tell the game where to put a real light (a spot or a point) and which mesh should glow. These empty names are the standard, and wiring them from Blender is `PLANNED`. Today you can still add socket names on the equipment definition's light lists (low beam, high beam, running, tail).
- **A Lights accessory slot** (`SLOT_Lights`) is a pad where a player can install a light bar or a beacon as a part. That slot type already exists; no base truck has one yet, but a mod truck can.

Place empties at the lens, aimed the way the lamp should shine (forward for headlights, rear for tails). If you model a light bar as a swappable part, author it on a `SLOT_Lights` block from the ModKit, with its own small lamps parented to the part if needed.

16. Export checklist (Blender 5.2 → Unreal 5.2)

Run this in order for a new model built at real size in meters.

In Blender, before you export

- ☐ Scene units: Metric, unit scale 1.0, meters
- ☐ Object scales are 1,1,1 — Apply Scale and Apply Rotation
- ☐ Vehicle origin is on the ground, centerline, in the right place (section 3)
- ☐ Wheel, door, boom, and ramp pivots are at the hubs or hinges
- ☐ Empties are named, at scale 1, in an Empties collection
- ☐ Collision boxes are named UCX_<ExactMeshName>_01 (and friends), convex, flat on decks
- ☐ Material slots have real names; UVs are in 0–1; textures are relative, not absolute paths
- ☐ Wheels are separate meshes; split parts still sit in their original place
- ☐ Nothing you care about is at scale 100

FBX settings (File → Export → FBX)

Setting	Value
Path Mode	Copy (or Relative). Do not bake absolute machine paths into the file.
Object Types	Mesh, Armature, Empty
Apply Scalings	**FBX All**
Forward / Up	**–Z Forward**, **Y Up**
Scale	**1.00**
Apply Unit	On
Use Space Transform	On
Apply Transform	On is fine for static-only exports (a trailer deck, a bumper). Be careful with skeletal meshes — leave it off if the armature would be baked incorrectly. The armature object should already sit at the origin with a clean rest pose.
Smoothing	Face, or Normals if you have custom normals you want to keep
Apply Modifiers	On
Add Leaf Bones	**Off**
Bake Animation / Bake Armature Actions	Off unless you are exporting animation on purpose

In Unreal, on import

- Import uniform scale **1.0**. Convert Scene on. Convert Scene Unit on (meters → centimeters).
- **Force Front X Axis:** leave **off** for these standard Blender exports. (The project's older skeletal import script turns it on, because those converted FS25 files were already X-forward in the FBX. If you use that script on a new model and the truck comes in sideways, turn it off for that file.)

- Import normals as they are if you exported custom normals. Do not generate a second set “just in case” until you have looked at the mesh.
- For static meshes, do not “auto generate” collision if you already shipped UCX boxes — you want your boxes, especially on a deck.
- For skeletal meshes, open the Physics Asset and check the bodies before you drive the machine.
- Spot-check: front of the truck is +X, roof is +Z, a known length (wheelbase, deck length) matches the real machine in centimeters.

17. Pre-flight checklists

Every model

- ☐ Real factory size, meters, scale 1, transforms applied
- ☐ Not pre-scaled 100×
- ☐ Origin on the ground, centerline, documented place
- ☐ Forward is Blender +Y (Unreal +X after export)
- ☐ Standard (or registered custom) empty names
- ☐ Materials named, UVs sane, textures relative
- ☐ FBX uses –Z forward, Y up, FBX All, scale 1, empties included
- ☐ Test import faces the right way and measures correctly

Vehicles (trucks and pickups)

- ☐ Wheel empties: WHEEL_FL, WHEEL_FR, WHEEL_R1L … and any extra axles
- ☐ Each wheel is its own mesh, pivot at the hub; Alcoa shared files were not edited
- ☐ Hitch empties: ATTACH_FifthWheel, ATTACH_Pintle, ATTACH_Ball, ATTACH_Gooseneck, and/or ATTACH_Pole as needed
- ☐ Has Fifth Wheel matches the real truck
- ☐ Slot empties for bumper, exhaust, rack, toolbox, bed, visor, lights as needed
- ☐ Split parts exported from the same file, same origin, left in place
- ☐ ROOT at the vehicle origin; ENTER at the driver’s door
- ☐ Light empties aimed correctly
- ☐ Physics Asset bodies do not swallow the tires

Trailers

- ☐ Front receiver empty (RECEIVER_Kingpin, RECEIVER_Lunette, RECEIVER_Coupler, ...) at the real height
- ☐ Rear attacher empty if another trailer or dolly can hook here
- ☐ Wheel meshes are separate; WHEEL_T1L ... empties at each hub
- ☐ Deck and ramp UCX tops are flat, level, and driveable
- ☐ Ramps (if they fold) are separate meshes, pivot at the hinge
- ☐ TIE_01, TIE_02, ... on both deck edges
- ☐ Bolsters and bunks have their own named meshes and boxes if loads rest on them
- ☐ Definition uses a towed vehicle class, not a driving class

Equipment and machines

- ☐ Each moving part is a separate mesh or bone, pivot at the joint
- ☐ Articulation hinge marked on articulated machines
- ☐ Cylinder barrel, rod, and both anchors named as a family
- ☐ Tool attach empty if the head or attachment can come off
- ☐ Wheel or track contact points placed
- ☐ Enter point if the machine is rideable
- ☐ Physics Asset covers the house and booms without fighting the tracks or wheels

18. Where the game data lives

The mesh is only half of a machine. The other half is an **equipment definition** data asset (`DA_<Name>`). The store, the workshop, saves, and delivery all read from it. You do not have to edit C++ to add a truck, a trailer, or a machine.

New Equipment window

In the Unreal Editor, open it in any of these ways:

- **Tools → TERRA GENESIS → New Equipment...**
- The **+** button on the main toolbar (tooltip “New Equipment”), to the right of Platforms
- Type `ALS.NewEquipment` in the Cmd box at the bottom of the editor

The same window also has a **New Accessory** tab, a **Slots** tab, a **Hitch types** list, and a **Tie points (trailers)** tab (that last tab is written, not compiled or tested yet). You can also run the export script and paste the JSON.

Fill the form from top to bottom: pick or copy a mesh, pick a category, pick a similar model so physics and the vehicle class are copied, then set name, price, fifth wheel, pintle, slots, and so on. Press Create. After that, use

Validate All Equipment in that window (or on the Accessory tab) to check every model, slot, and part. Problems show in the window and in the log as `ALS_REGISTRY problem: …`.

What the definition stores (the bits that touch your model)

- Body mesh (skeletal or static), wheel set, optional fifth-wheel mesh
- Hubs (bone name, left/right, steers, radius and width in cm, position in cm)
- Fifth-wheel slide and rest position; Has Fifth Wheel; rear hitch type and location
- Mount points (slots): name, type, offset or socket, required, baked, factory part
- Tie points (cm, actor space), lights, moving tools, hydraulic cylinders, enter point
- Category, price, model tags (for example `Semi`, `LongFrame`), and the vehicle class to spawn

Mods

Outside the editor, a mod is a folder: `Mods/<mod>/Vehicles/*.json` and optional `Parts/*.json`, plus a small `mod.json`. The Modder Kit at `~/TerraGenesis_ModKit/` has a README, an example truck, the part catalog, and the `SLOT_<Type>.fbx` blocks. `validate_mod.py` checks the same rules as the game. Custom hitch types go in `Mods/<mod>/Hitches/*.json` (section 5); vehicle files use `attachers` and `receivers`. This is approved and being implemented.

A sidecar file next to a converted model (`*_ue.json`) is another way hubs, enter points, fifth-wheel slide, and tie points get into a definition. Units in that file are centimeters, axis X forward, Y right, Z up. Do not scale those numbers by 100 again.

19. Appendix

Renaming FS25 collision boxes

If your Blender file still has Giants-style collision boxes (often in a collision collection, named however FS25 wanted), you can keep the shapes. They need Unreal names before export:

1. Know the visible mesh name each box belongs to, exactly (for example `front_bolster`).
2. Rename the boxes `UCX_front_bolster_01`, `UCX_front_bolster_02`, …
3. Make sure each is a single convex hull (a cube or a convex mesh). Apply scale.
4. Export them in the same FBX as that mesh.

A small Blender script can do the rename from a collection, so your working names can stay as they are in the .blend. If you want that script dropped into your setup, say so and we will add it.

Useful project paths

What	Where
Unreal project	`~/Documents/Unreal Projects/TerraGenesis/`
Authoring docs	`~/TerraGenesis_Docs/` (how to add equipment, slots, parts, transmission, lockers)
Modder Kit	`~/TerraGenesis_ModKit/` — start with `README.md` and `Generated/Blocks/`

What	Where
Shared Alcoa wheels	`Content/Wheels/Semi/` — do not modify
Accessory parts	`/Game/Equipment/Accessories/ACC_…`
Categories	`/Game/Equipment/Categories/CAT_…`

A short “does this look right?” test

1. Import into an empty Unreal folder, not on top of a live asset.
2. The front points +X, the roof +Z, the driver’s door is on −Y for a North American truck.
3. A tape on the wheelbase or the deck length matches the real machine (in cm).
4. The vehicle sits on the ground, not 2 meters in the air and not buried to the axles.
5. On a trailer, you can drop a box on the deck and it rests on the wood, not in mid-air or through the frame.
6. Run **Validate All Equipment** after the definition is created.

> LAST WORD
>
> When a standard name fits, use it. When it doesn’t, make a clear custom name with the right prefix and register it in the New Equipment window or in mod JSON. Put the empties in the model now, even for systems still being implemented — hitch types, six-trailer chains, visors, and light empties included. The mesh you export today should not need a second naming pass later.