The Process (3D Model Guide)

Created	@September 22, 2024 1:39 PM
Tags	

Phase 1

A- Checking size scale to UE standard:
Ensure your model’s scale matches Unreal Engine’s standard units. Typically, 1 unit in Maya equals 1 centimeter. Adjust the scale settings in Maya to match this standard.

▼ Resources

For MAYA.

https://youtu.be/q0ev5MbI3gk

For Blender.

Copilot

Generando respuestas…

To ensure your Blender models are correctly scaled for Unreal Engine (UE), follow these steps:

1. **Set the Scene Scale in Blender**:
   - Go to the Scene Properties tab.
   - Under Units, set the Unit Scale to 0.01. This makes 1 Blender unit equal to 1 centimeter, matching Unreal Engine's default unit
2. **Adjust Object Scale**:
   - Select your object.
   - Press `S` to scale and adjust the size as needed.
   - Apply the scale by pressing `Ctrl+A` and selecting "Scale".
3. **Export Settings**:
   - When exporting your model as an FBX file, ensure the scale is set to 1.0 in the export settings.

B- Base model:
Use the modeling tools to shape your model as needed.

▼ Resources

For Maya.

https://youtu.be/eBEitxaRYQs

For Blender.

https://youtu.be/sc24Yu9eCS8

For UE.

https://youtu.be/NZYxf2IbL5E

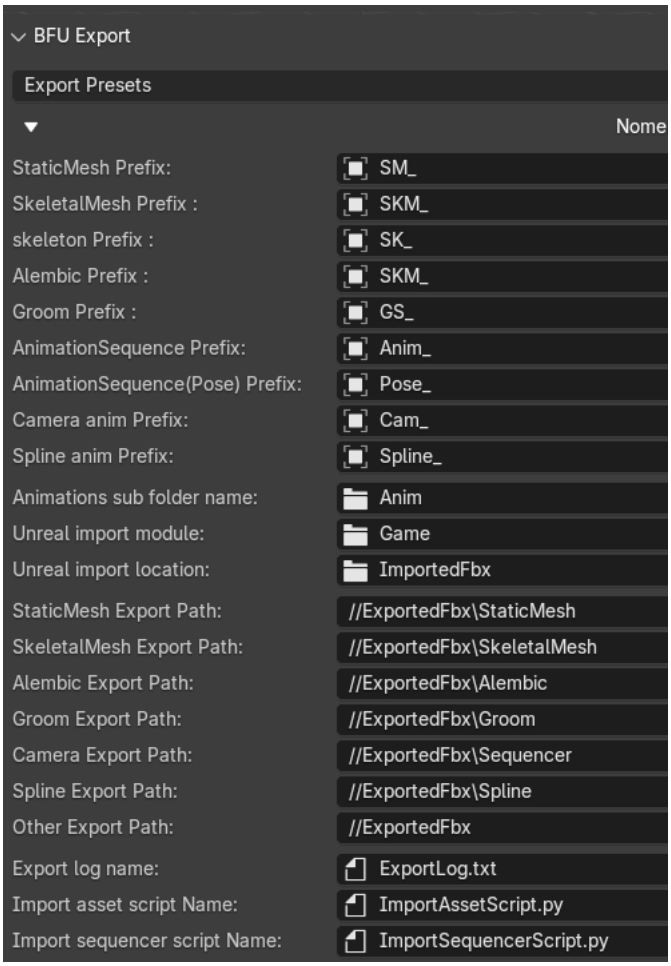
https://youtu.be/DaEJZHnz0V0

C- Appropriate naming structure:

Examples can be found on the WIP Proton Drives LINK HERE

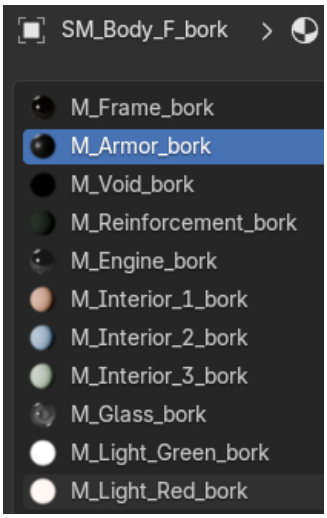
▼ Resources

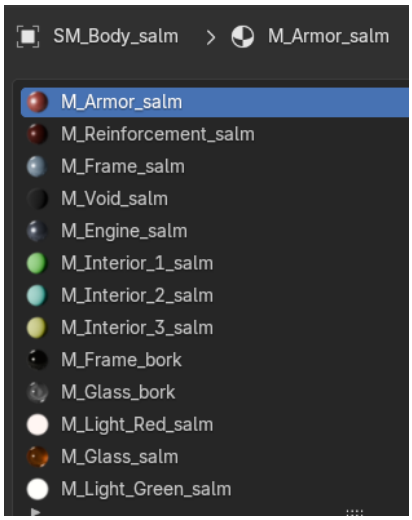
1. Due to the specifics of Unreal Engine, we require all assets, such as Static Mesh and others, to use naming conventions for Unreal Engine.



We also use some of our own naming conventions. For a complete example, visit the Plunger Blender project and have a look around.

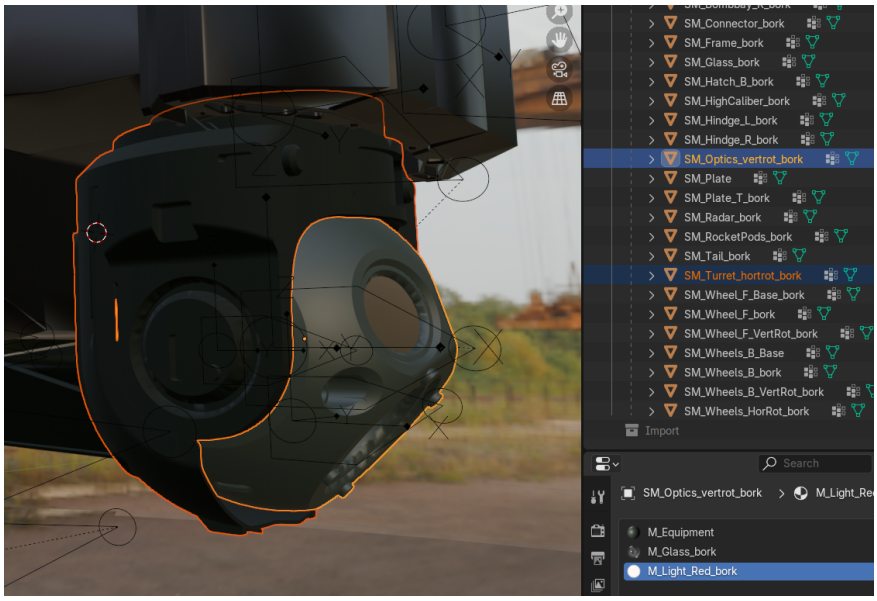
2. Materials follow a simple M_Nameofmaterial format. If there are multiple variations of a model with different materials, adding a suffix is sufficient. For example, Salmon and Borka have _salm and _bork. The same goes for static meshes, empties, etc.



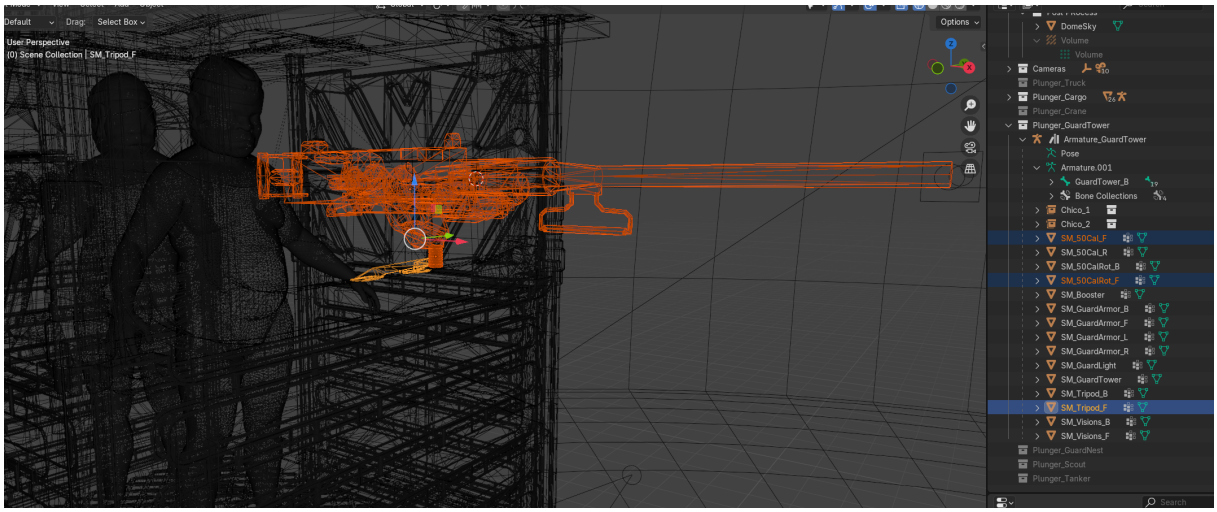


3. For moving parts, we usually divide them into the base model that connects them to the non-movable part (_base), the horizontal rotating part (_hortrot), and the vertical rotating part (_vertrot). Guns, turrets, spotlights, engines, and wheels usually have two or all three of these.

   If it's only one piece of the part and it rotates, or if there are two parts rotating in the same direction, then just adding _Rot is enough.



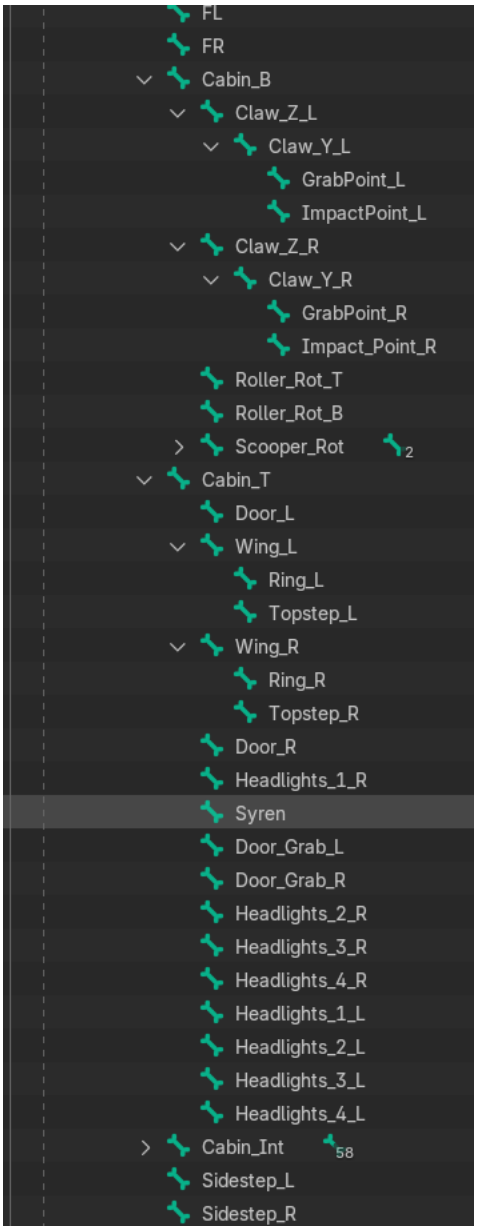
Sometimes we just name them with directional suffixes, as it is enough to identify them.



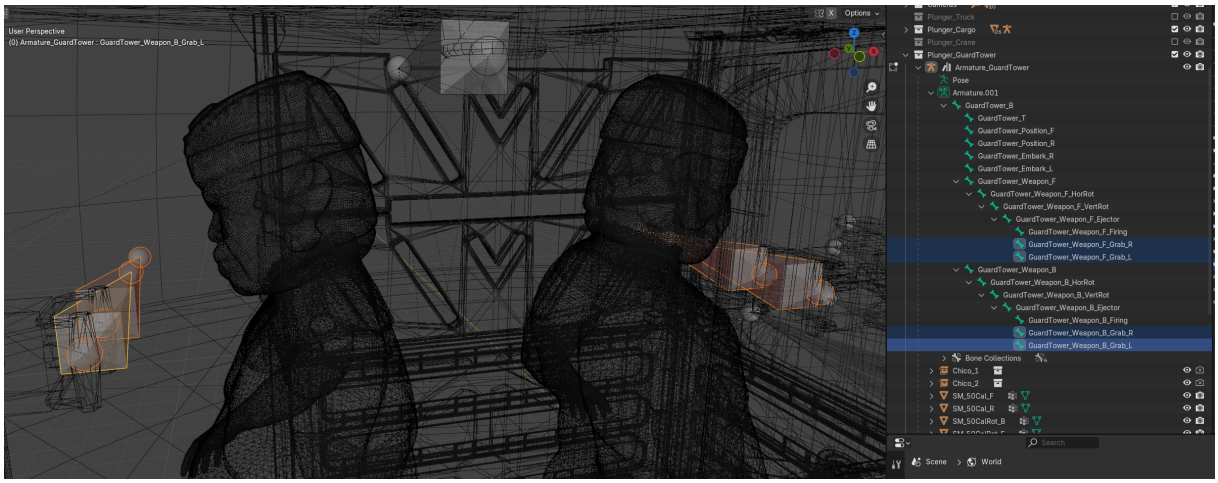
4. Armatures are named Armature_nameofmodel. Rig bones usually have a root, and in the case of vehicles, parts are named nameofbone_direction, starting from the root:

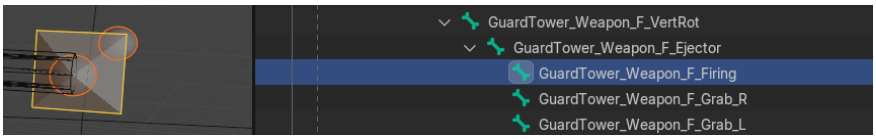
   FR (front right)
   FL (front left)
   BR (back right)
   BL (back left)
   T (top) after direction, e.g., nameofpart_F_T
   B (bottom) after direction, e.g., nameofpart_F_B
   F (front)
   B (rear) before vertical, nameofpart_B_B would be rear bottom
   R (right)
   L (left)

Plunger Example (WIP)

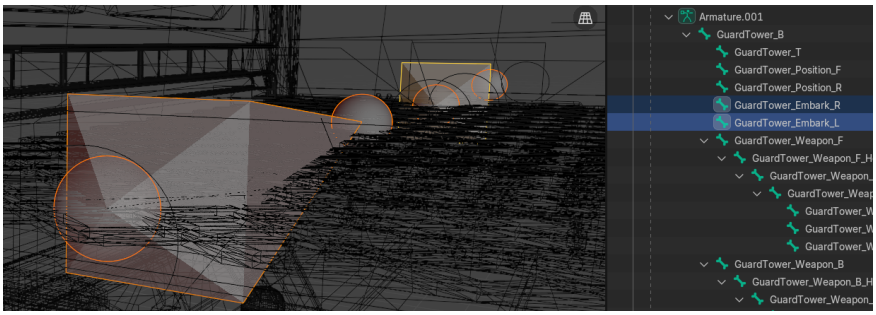


So, as an example in the GuardTower of Plunger, the front machine gun has a grab for the hand on the left side. That would be nameofbone_F_L or _R.



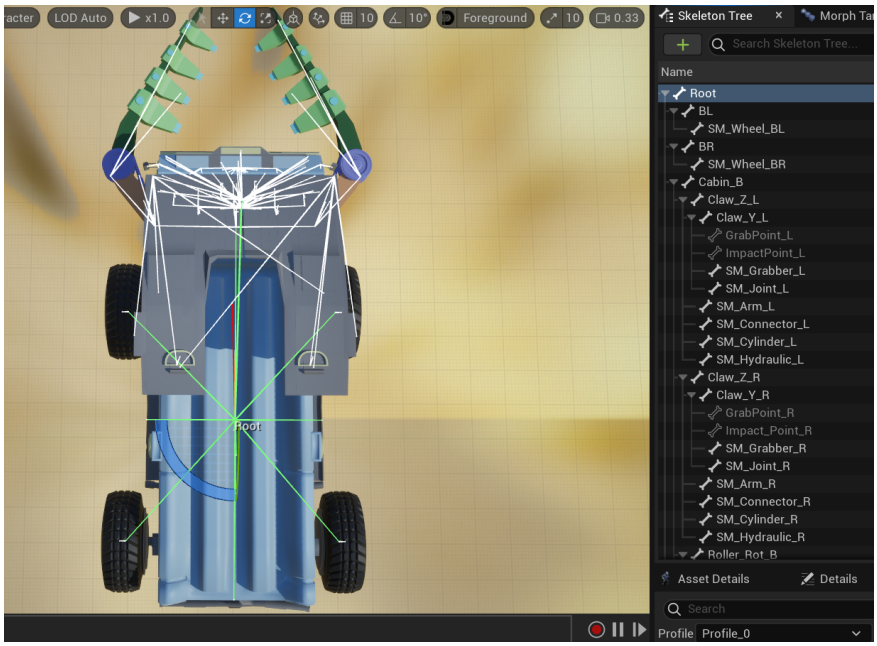


The GuardTower also has a bone for animators to easily connect the hands of characters. There are two on that piece, serving as stepping points for climbing or releasing a VFX, one on the left and one on the right, so _L and _R.

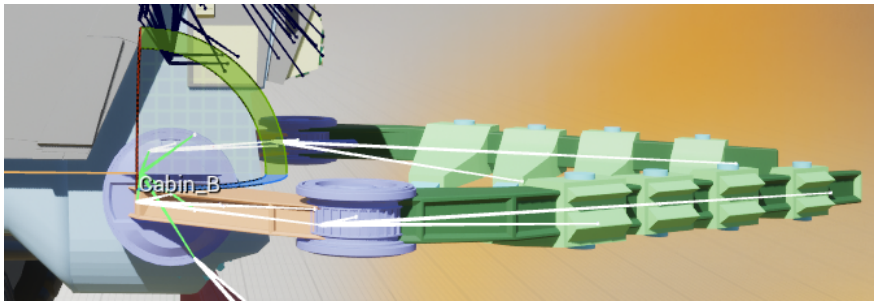


If there is a point for VFX like engine smoke, we add a bone with the suffix bonename_VFX. If it is an impact point for coders, located on the front top part of the car, then it is ImpactPoint_T_F, etc.

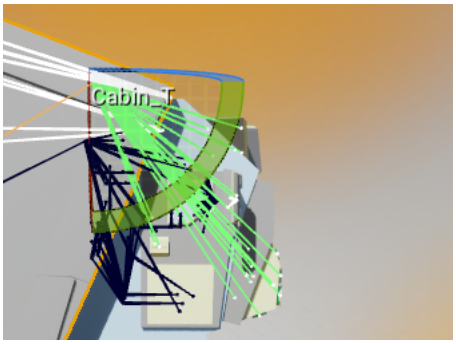
5. Here is an awesome example of proper naming for a rig and mesh.



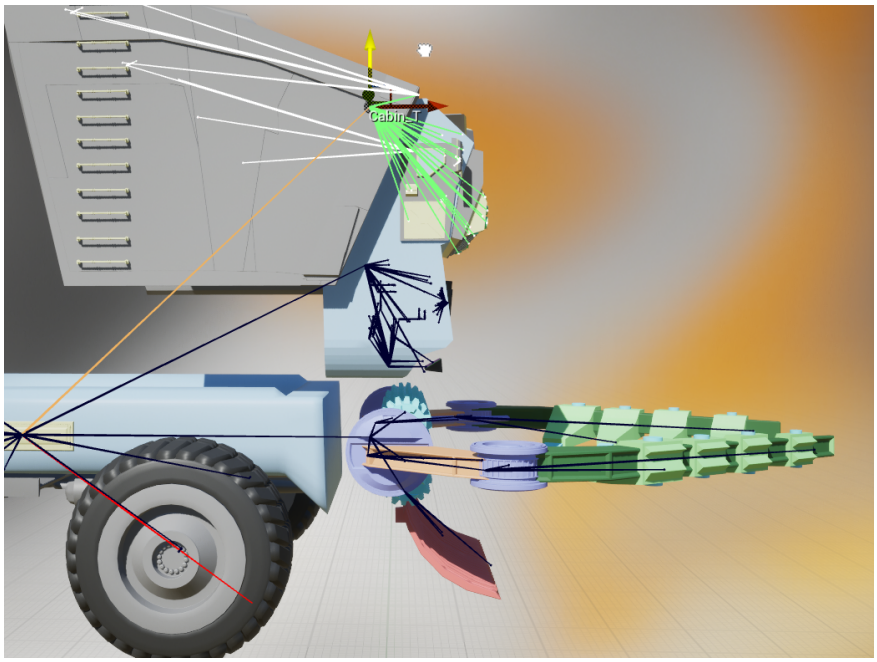
Helps identify and divide all parts accordingly, such as one cabin having three management zones.



The top one also makes it easy to avoid a spaghetti-like hierarchy.



It is easy to manipulate or divide sections when needed. It is a goldmine for animators and coders.



All of this has its reasons and eases the workflow for other disciplines.

D- Material slots:
Assign materials to different parts of your model and assign them to your model's faces or objects.

▼ Resources

For Maya.

https://youtu.be/aJhXITez20Y

For Blender.

https://youtu.be/K3wMqyjvgtE

For Unreal.

https://youtu.be/2d30APnrJBk

E- Centering and setting transformation:
Center the pivot point of your model to geometry. Check in Blender for examples.

▼ Resources

For Maya.

https://www.youtube.com/watch?v=OmBxBNJwB8w

For Blender.

https://youtu.be/LHIvN5ZqdTU

For Unreal.

https://youtu.be/zdjG2pjGpsE

F- Setting up parenting, creating a hierarchy:
Parent objects to create a hierarchy. Select the child object, then the parent object, and press P to parent them. This creates a hierarchical relationship between objects.

▼ Resources

For MAYA.

https://youtu.be/1ahz51RWT2A

For Blender.

https://youtu.be/VnXW_r9tTvo

For Unreal.

https://youtu.be/k92eaGP4pss

Phase 2

A- Creation of an armature and setting up bones:
Create a skeleton for your model by placing joints at key points to form the skeleton.

▼ Resources

Export/Import Tips.

In past year we used Maya for aditional effects on characters (especially for cloth simulations) and we discovered that the best way to import is `.mdd`.

This format transfers the maya animation and convert it into animated shapes keys:

1. Get your first pose (in maya) and export as `.obj` object
2. Export the animation from maya as `.mdd`
3. Import the `.obj` object in blender
4. Select your object and import the animation as `.mdd`

Always check "keep vertex orders".

For MAYA.

https://youtu.be/J18m_RD8pII

For Blender.

https://youtu.be/jbFjTFFwPsM

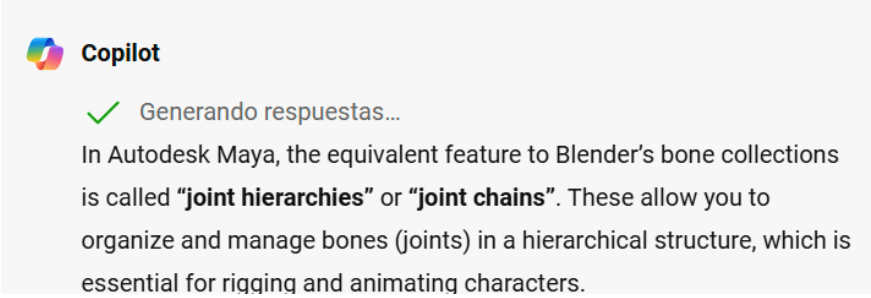
For Unreal.

https://youtu.be/txFsPf-BlEg

B- Creating bone collections based on areas or joint function:
Group bones logically based on their function or location (e.g., arm bones, leg bones). This helps in managing the rig more efficiently.

▼ Resources

MAYA/Blender difference.



MAYA Rigging

https://youtu.be/1wvdQy2Fdhw

Blender Rigging

https://youtu.be/DDeB4tDVCGY

Blender Rigging for Unreal with Free Unreal Plugin.

https://youtu.be/0lwKWxBeFcQ

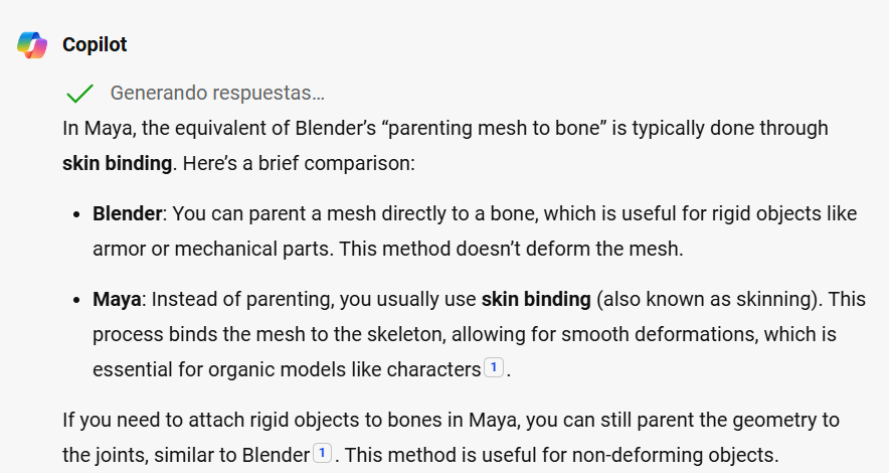
UE5.4 New Control Rig

https://youtu.be/0KrrE4heS7s

C- Parenting meshes to bones, positioning bones for later connections:
Attach meshes to bones by selecting the mesh, then the joint, and parenting it to it. Position the bones correctly to ensure proper deformation.

▼ Resources

Differences in Blender and MAYA, Unreal.



Copilot

Generando respuestas…

In Maya, the equivalent of Blender's "parenting mesh to bone" is typically done through **skin binding**. Here's a brief comparison:

- **Blender**: You can parent a mesh directly to a bone, which is useful for rigid objects like armor or mechanical parts. This method doesn't deform the mesh.
- **Maya**: Instead of parenting, you usually use **skin binding** (also known as skinning). This process binds the mesh to the skeleton, allowing for smooth deformations, which is essential for organic models like characters [1].

If you need to attach rigid objects to bones in Maya, you can still parent the geometry to the joints, similar to Blender [1]. This method is useful for non-deforming objects.

In Unreal Engine, the equivalent of skin binding in Maya or parenting a mesh to a bone in Blender is called **"Skeletal Mesh Binding"**. This process involves attaching a mesh to a skeleton so that the mesh deforms according to the movements of the skeleton.

Here's a brief overview of how you can achieve this in Unreal Engine:

1. **Importing the Skeletal Mesh and Skeleton**:
   - First, ensure that your skeletal mesh and skeleton are properly exported from Blender or Maya. Use the FBX format for compatibility.
   - Import both the skeletal mesh and skeleton into Unreal Engine.
2. **Setting Up the Skeleton**:
   - In Unreal Engine, open the Skeleton asset. Here, you can adjust the bone hierarchy and ensure that the bones are correctly mapped.
3. **Binding the Mesh to the Skeleton**:
   - Open the Skeletal Mesh asset in Unreal Engine.
   - In the Skeletal Mesh Editor, you can see the mesh and its associated skeleton. Ensure that the mesh is correctly bound to the skeleton.

5. **Animation**:
   - Once the mesh is bound to the skeleton, you can create and import animations. These animations will drive the skeleton, causing the mesh to deform accordingly.

For more detailed steps, you might find tutorials on YouTube or documentation from Unreal Engine helpful. If you have any specific questions about the process, feel free to ask!

Más información 1 youtube.com 2 youtube.com 3 youtube.com 4 reddit.com +4 más

For MAYA.

https://youtu.be/FSvEb91T7aM

For Blender.

https://youtu.be/ylpVj94_GNA

For Unreal.

https://youtu.be/8N3r92InMxI

D- Assigning and creating vertex groups of the meshes:
Define vertex groups for deformation. This assigns vertices to groups.

▼ Resources

For Blender.

Vertex Groups in Blender

Not to worry, we keep it simple as our models are usually hard surface parts. If the static meshes are properly set up, as in the example of the Plunger or other models we've uploaded, it's just a matter of adding the entire mesh part to its particular group.

Static Meshes

1. **Assigning Vertex Groups**:
   - Select all vertices of the static mesh part.
   - In the Object Data Properties panel, go to the **Vertex Groups** section.
   - Click **+** to create a new vertex group and name it according to the area (e.g., `cabin`, `wheel_fl`, `cargo`, `claw_r`, `claw_l`).
   - With the vertices still selected, click **Assign** to add them to the group.

Moving Parts

For moving parts that have a base, a horizontal rotation piece (hortrot), and a vertical rotation pie (vertrot), such as a turret or a machine gun:

1. **Assigning Vertex Groups**:
   - Select the base vertices.
   - Create a new vertex group named `partname_base` (e.g., `turret_base`) and click **Assign**.
   - Select the horizontal rotation vertices.
   - Create a new vertex group named `partname_hortrot` (e.g., `turret_hortrot`) and click **Assign**.
   - Select the vertical rotation vertices.
   - Create a new vertex group named `partname_vertrot` (e.g., `turret_vertrot`) and click **Assign**.

Example

- **Static Mesh**: Select all vertices of the `cabin` part, create a vertex group named `cabin`, and click **Assign**.
- **Moving Part**: For a turret:
  - Select the base vertices, create a vertex group named `turret_base`, and click **Assign**.
  - Select the horizontal rotation vertices, create a vertex group named `turret_hortrot`, and click **Assign**.
  - Select the vertical rotation vertices, create a vertex group named `turret_vertrot`, and click **Assign**.

https://youtu.be/OYg2fyQMtJc

For Maya.

Vertex Groups in Maya

In Maya, the equivalent of Blender's Vertex Groups is **Sets**. Here's how you can manage them:

1. **Creating Sets**:
   - Select the vertices you want to group.
   - Go to **Create > Sets > Quick Select Set**.
   - Name your set according to your naming convention (e.g., `cabin`, `wheel_fl`, `cargo`, `claw_r`, `claw_l`).
2. **Assigning Vertices to Sets**:
   - Select the vertices.
   - With the vertices selected, go to **Edit > Quick Select Sets** and choose the set you want to add them to.

Naming Conventions

For moving parts with bases and rotational pieces, you can follow a similar naming convention:

- **Base**: `partname_base` (e.g., `turret_base`)
- **Horizontal Rotation**: `partname_hortrot` (e.g., `turret_hortrot`)
- **Vertical Rotation**: `partname_vertrot` (e.g., `turret_vertrot`)

Example Workflow

1. **Static Meshes**:
   - Select all vertices of a static mesh part.
   - Create a set and name it according to the area (e.g., `cabin`, `wheel_fl`).

Example Workflow

1. **Static Meshes**:
   - Select all vertices of a static mesh part.
   - Create a set and name it according to the area (e.g., `cabin`, `wheel_fl`).
2. **Moving Parts**:
   - For a turret with a base and rotational pieces:
     - Select the base vertices and create a set named `turret_base`.
     - Select the horizontal rotation vertices and create a set named `turret_hortrot`.
     - Select the vertical rotation vertices and create a set named `turret_vertrot`.

For Unreal.

Using Vertex Colors

1. **Assigning Vertex Colors in Blender/Maya**:
   - Before exporting your model to Unreal Engine, you can assign vertex colors in Blender or Maya. These colors can be used to differentiate parts of your mesh.
   - In Blender, you can use the Vertex Paint mode to assign colors.
   - In Maya, you can use the Paint Vertex Color Tool.
2. **Importing to Unreal Engine**:
   - When you import your model into Unreal Engine, ensure that the vertex colors are imported as well.
   - In the import settings, check the option to import vertex colors.
3. **Using Vertex Colors in Unreal Engine**:
   - You can use vertex colors in your materials to create different effects or to identify different parts of your mesh.
   - In the Material Editor, you can use the **Vertex Color** node to access the vertex colors and apply different materials or effects based on these colors.

Using Material IDs

Another approach is to use **Material IDs** to differentiate parts of your mesh:

1. **Assigning Material IDs in Blender/Maya**:
   - Assign different materials to different parts of your mesh in Blender or Maya.
   - Each material will be imported as a separate Material ID in Unreal Engine.
2. **Importing to Unreal Engine**:
   - When you import your model, Unreal Engine will recognize the different Material IDs and allow you to assign different materials to each part.
3. **Using Material IDs in Unreal Engine**:
   - In the Static Mesh Editor, you can assign different materials to each Material ID.
   - This allows you to manage different parts of your mesh separately.

Example Workflow

1. **Static Meshes**:
   - Assign vertex colors or different materials to static parts of your mesh in Blender or Maya.
   - Import the mesh into Unreal Engine and use the Vertex Color node or Material IDs to manage these parts.
2. **Moving Parts**:
   - For moving parts like turrets, you can use separate meshes for each part (base, horizontal rotation, vertical rotation).
   - Import these meshes into Unreal Engine and set up the hierarchy and animations accordingly.

E- Creating modifiers to “bake in” said vertex groups to the armature:
In Blender, this involves creating a deform armature modifier, selecting the armature, selecting the previously defined vertex group, and applying it. That "bakes it in.

▼ Resources

For MAYA.

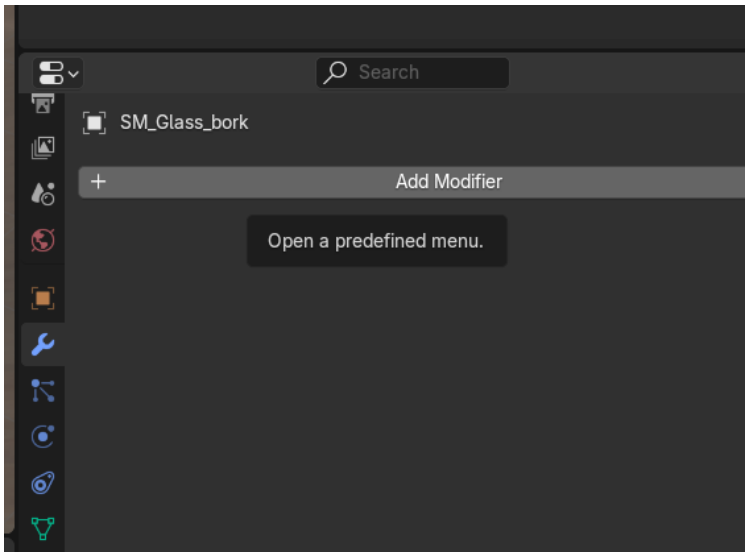
In Maya, the process of deforming a mesh with an armature (referred to as a skeleton in Maya) involves using skinning techniques. Here's a step-by-step guide:

1. **Create the Skeleton**: First, create the skeleton for your character or object. You can do this by using the Joint Tool found under the `Rigging` menu set.
2. **Bind the Skin**: Select the mesh you want to deform, then shift-select the skeleton. Go to `Skin > Bind Skin > Options`. Here, you can choose the binding method (e.g., closest distance, heat map, etc.) and other settings. Click `Bind Skin` to attach the mesh to the skeleton.
3. **Paint Skin Weights**: To refine how the mesh deforms, use the `Paint Skin Weights Tool` found under `Skin > Paint Skin Weights Tool`. This allows you to adjust the influence of each joint on the mesh vertices.
4. **Add Deformers**: You can add additional deformers to the mesh for more complex deformations. For example, you can use the `Lattice`, `Blend Shape`, or `Cluster` deformers found under the `Deform` menu.
5. **Adjust Deformer Attributes**: Each deformer has its own set of attributes that you can tweak to achieve the desired effect. For instance, with a `Lattice` deformer, you can adjust the lattice points to deform the mesh.
6. **Animate**: Once your mesh is properly skinned and deformed, you can animate the skeleton to see the deformations in action.

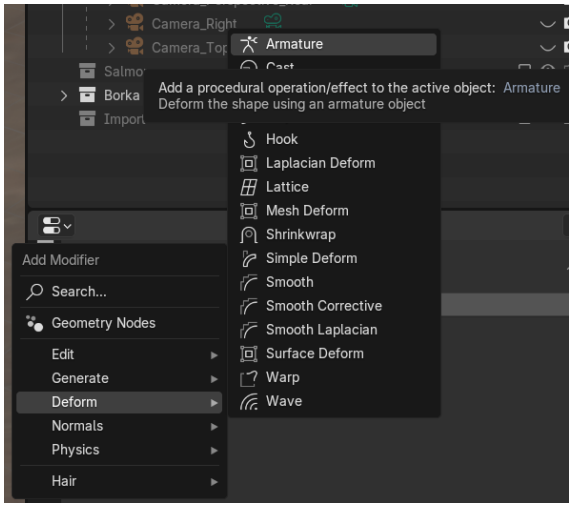
https://youtu.be/WQMjCm-KAis

For Blender.

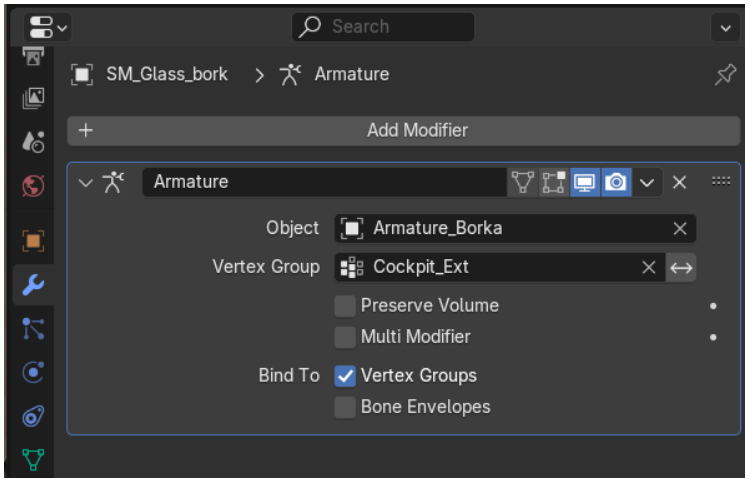
Add a modifier to meshes with created and set vertex groups.



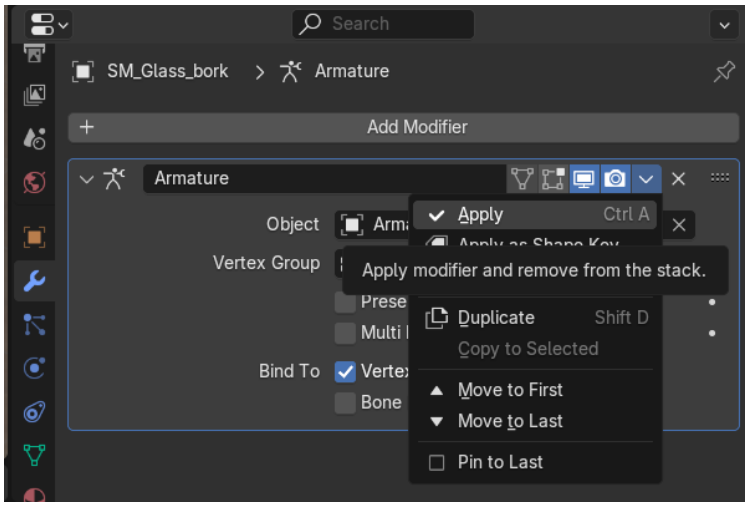
Choose Armature Deform Modifier.



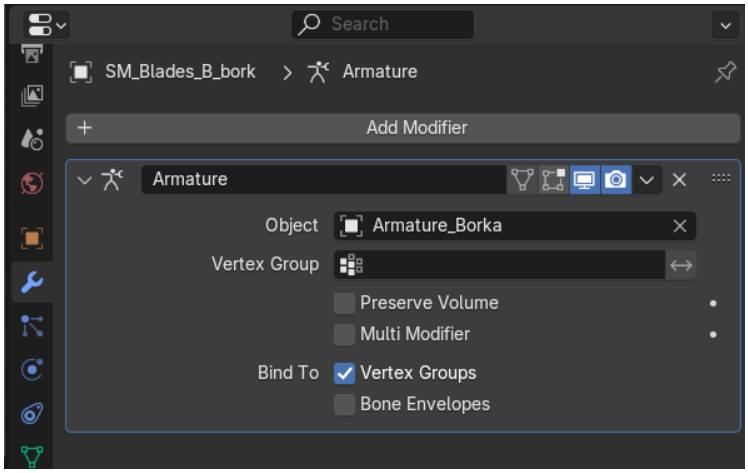
Choose parented armature and created vertex group.

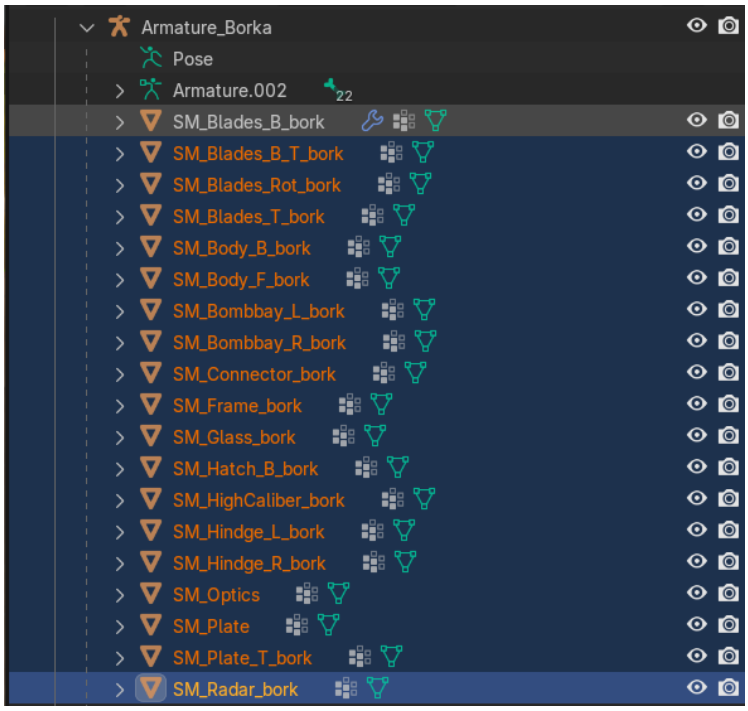


Apply. Repeat on all meshes.

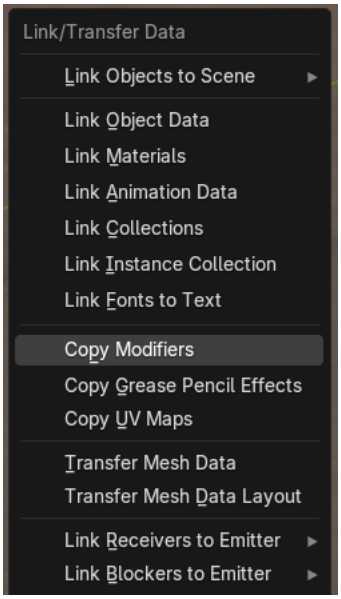


You can also select all the meshes (be careful to choose only meshes, which is why proper naming is important to help in these parts) and, as the last one, choose the one set to armature but with no vertex group selected. It would look like this.

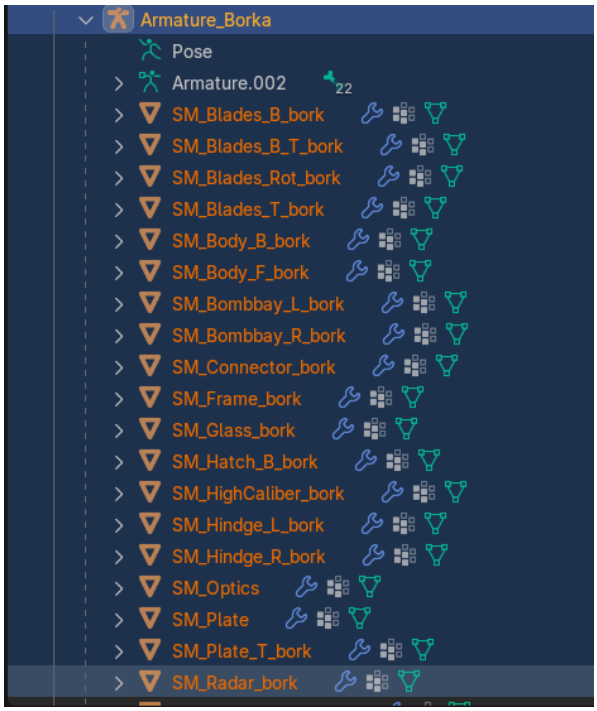




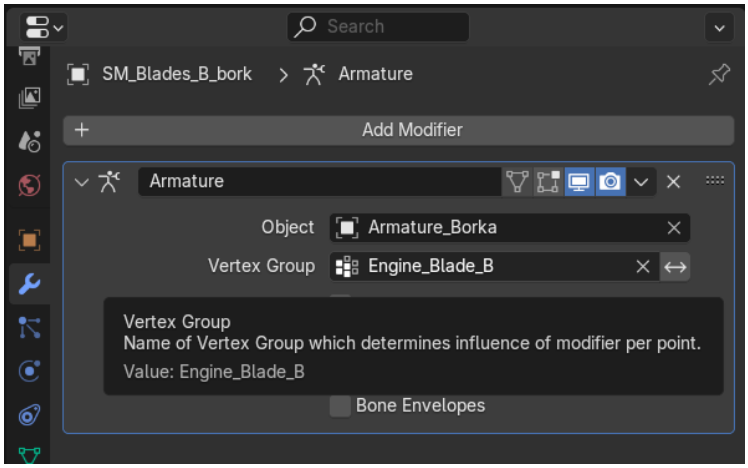
Select the last object with CTRL+Click, move the mouse to the main area, and press Ctrl+L. Select 'Copy Modifiers'.



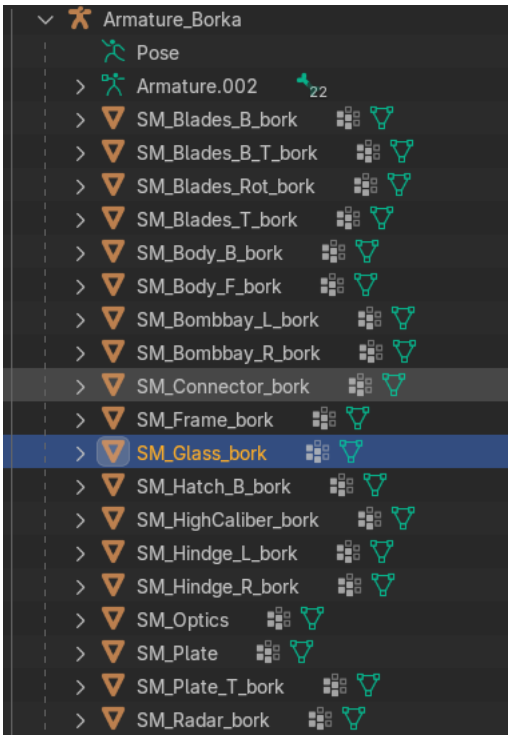
If properly applied, it will look like this.



A bit tedious, go one by one and add the vertex group for each. Apply.



The end result should look like this.



In Unreal.

Yes, Unreal Engine 5 also supports similar deformation techniques through its skeletal mesh system. Here's a brief overview of how it works in Unreal Engine 5:

1. **Importing the Mesh and Skeleton**: First, import your mesh and skeleton (created in Blender or Maya) into Unreal Engine. You can do this by using the `FBX` format, which preserves the skeletal structure and skin weights.
2. **Setting Up the Skeleton**: Once imported, you can view and edit the skeleton in the `Skeleton Editor`. This allows you to adjust the bones and their hierarchy.
3. **Skinning and Weight Painting**: Unreal Engine uses the skin weights from your original 3D software (Blender or Maya). However, you can further refine these weights using the `Weight Painting` tools available in Unreal Engine.
4. **Adding Animations**: You can import animations created in Blender or Maya and apply them to your skeletal mesh. Unreal Engine supports various animation formats and allows you to create complex animation sequences.
5. **Using Deformers**: Unreal Engine has its own set of deformers and modifiers, such as `Morph Targets` (similar to Blend Shapes in Maya) and `Control Rig` for more advanced rigging and animation control.
6. **Blueprints and Animation Blueprints**: Unreal Engine's Blueprint system allows you to create complex animation behaviors and interactions without writing code. You can use Animation Blueprints to control how your skeletal mesh deforms and animates in real-time.

https://youtu.be/mtven29plfc

F- Connecting the bones of the armature:
Connect bones and control their behavior.

▼ Resources

For MAYA.

https://youtu.be/depyrdgYSgo

For Blender.

Connecting Bones.

https://youtu.be/DKCPY1wsW90

More Explanation.

https://youtu.be/gdOaUv0_TC8

For Unreal.

https://youtu.be/MMe80uQr_Vg

G- UE export/import tests, fixing any issues:
Export your model to Unreal Engine using the FBX format (File > Export Selection). Import it into Unreal Engine and test for any issues. Fix any problems in Maya and re-export as needed.

▼ Resources

For MAYA.

https://youtu.be/YEyo1WkgDgE

Fir Blender.

Unreal Free Export Plugin.

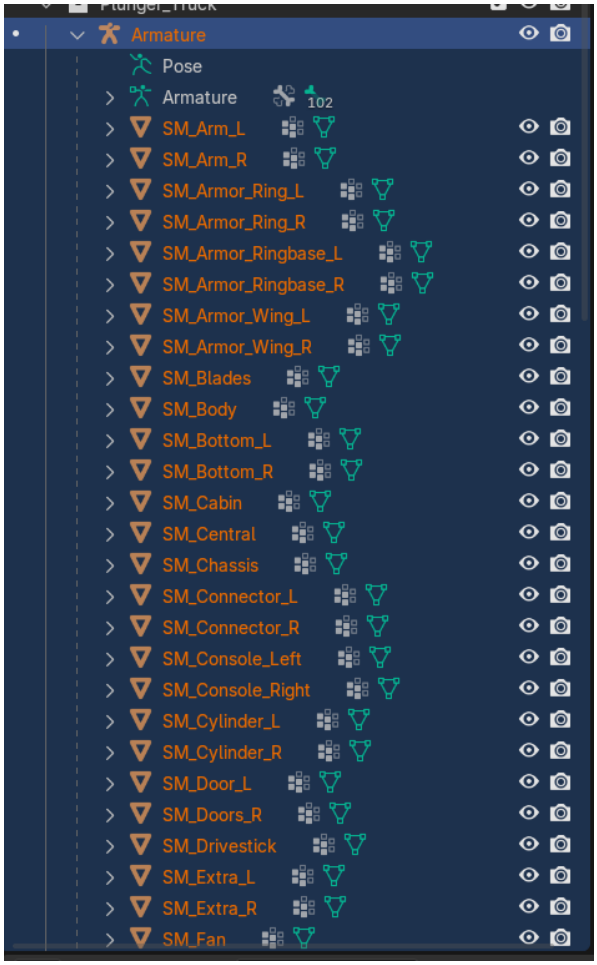
GitHub - xavier150/Blender-For-UnrealEngine-Addons: I have created this addons for export asset from Blender to Unreal Engine 4

I have created this addons for export asset from Blender to Unreal Engine 4 - xavier150/Blender-For-UnrealEngine-Addons

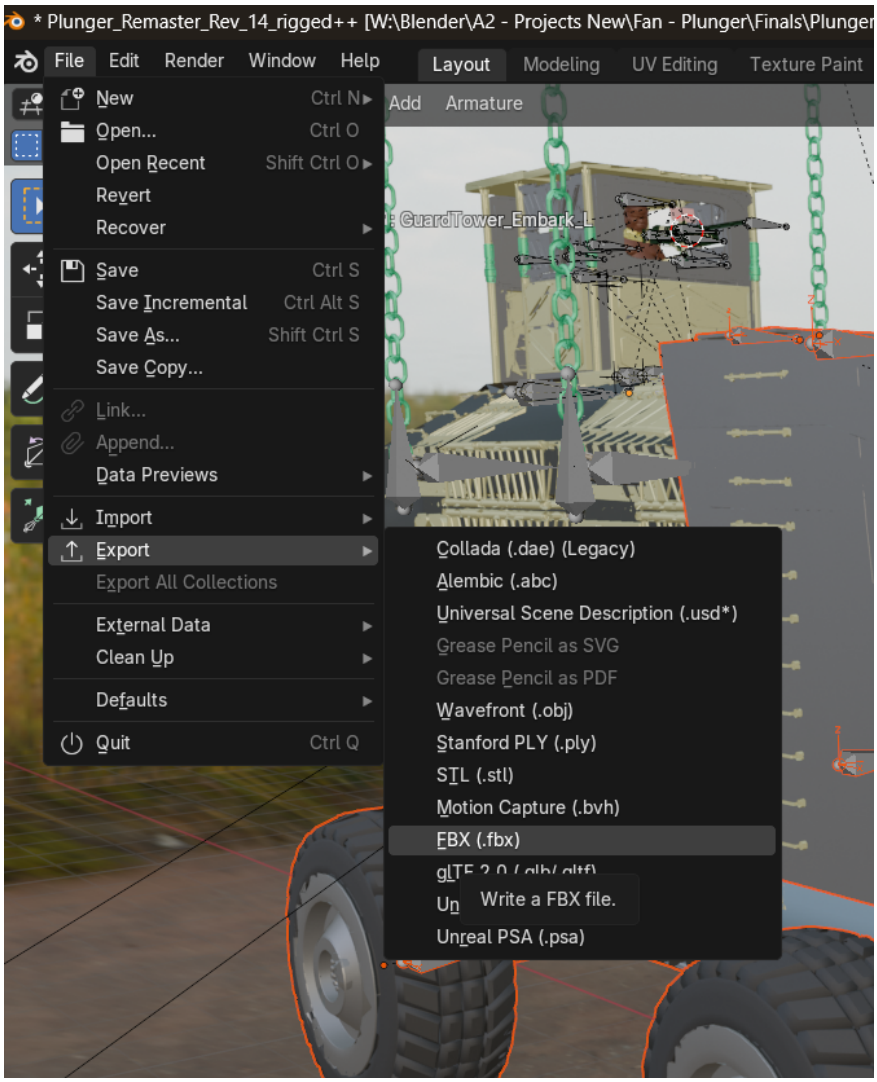
https://github.com/xavier150/Blender-For-UnrealEngine-Addons

Blender Export Settings.

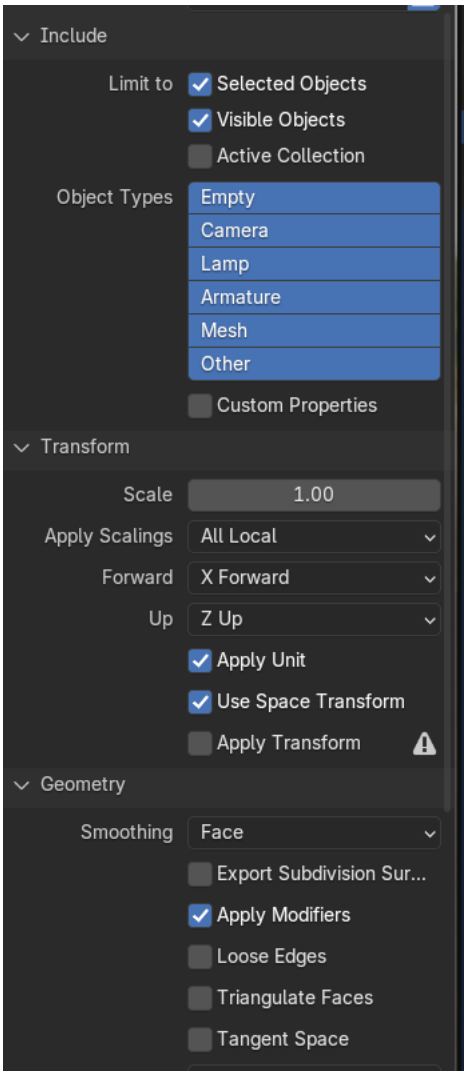
Make sure all selected and visible (the eyes on the right).

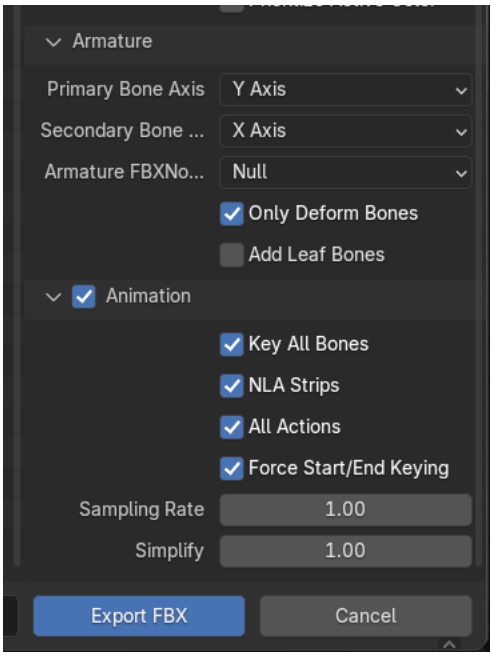


Select Export .FBX.

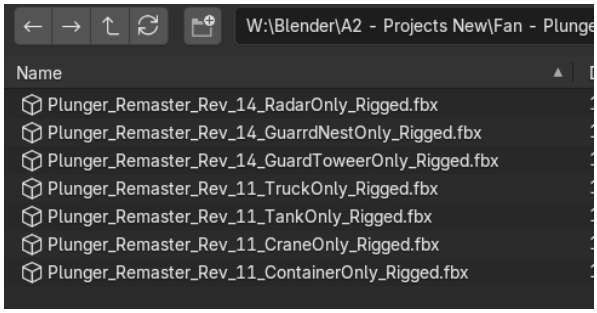

* Plunger_Remaster_Rev_14_rigged++ [W:\Blender\A2 - Projects New\Fan - Plunger\Finals\Plunger
File
Edit
Render
Window
Help
Layout
Modeling
UV Editing
Texture Paint
Add
Armature
GuardTower_Embark_L
New Ctrl N
Open... Ctrl O
Open Recent Shift Ctrl O
Revert
Recover
Save Ctrl S
Save Incremental Ctrl Alt S
Save As... Shift Ctrl S
Save Copy...
Link...
Append...
Data Previews
Import
Export
Export All Collections
External Data
Clean Up
Defaults
Quit Ctrl Q
Collada (.dae) (Legacy)
Alembic (.abc)
Universal Scene Description (.usd*)
Grease Pencil as SVG
Grease Pencil as PDF
Wavefront (.obj)
Stanford PLY (.ply)
STL (.stl)
Motion Capture (.bvh)
FBX (.fbx)
Write a FBX file.
Unreal PSA (.psa)

Settings


Include
Limit to
Selected Objects
Visible Objects
Active Collection
Object Types
Empty
Camera
Lamp
Armature
Mesh
Other
Custom Properties
Transform
Scale
1.00
Apply Scalings
All Local
Forward
X Forward
Up
Z Up
Apply Unit
Use Space Transform
Apply Transform
Geometry
Smoothing
Face
Export Subdivision Sur...
Apply Modifiers
Loose Edges
Triangulate Faces
Tangent Space



Name it appropriately.



Phase 3

A- Texturing:
Apply textures to your model using the Substance app. Create and assign materials, then use UV mapping to ensure textures fit correctly on your model.

▼ Resources

MAYA Substance

https://youtu.be/TjI9bPoP9LM

Substance in Blender

https://youtu.be/jCwTEEyDX3Y

Unreal Technical Details.

https://youtu.be/WqFTYh_KZkQ

B- Texture variations:
Create different texture sets for your model. Duplicate your material and adjust the textures to create variations (e.g., different colors, patterns).

▼ Resources

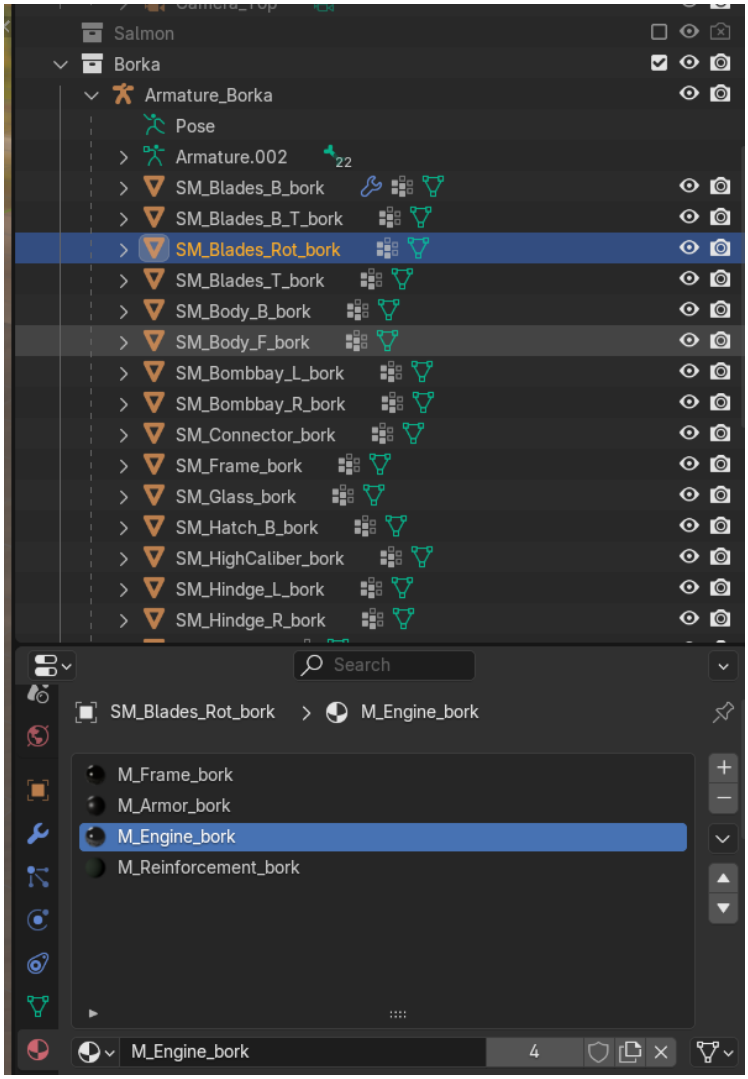
General Idea:

The only prerequisite is that the previous steps are done correctly. As I don't have experience in MAYA, I will use Blender next as an example of what needs to be done so it can be understood in MAYA too.

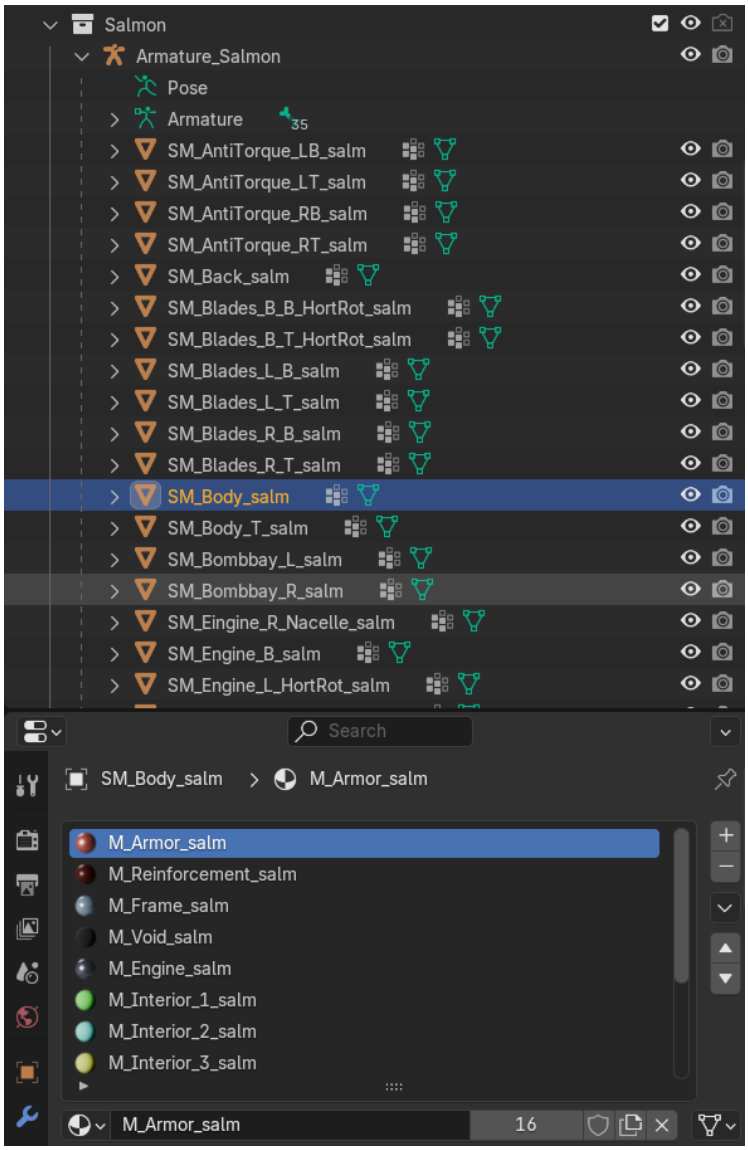
Blender has switchable folders. Notice the right-side selection. You create them and put stuff in them, then you get the option to turn them off and on.

Now switched. The same model variation is saved in the same Blender project as a separate model with its own folder.

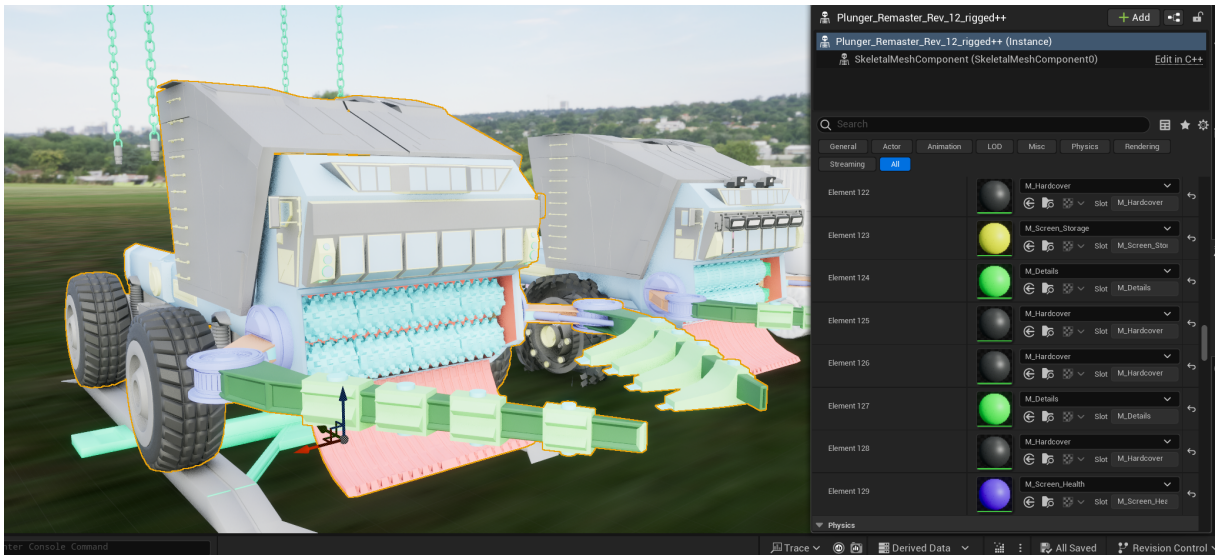
Then we need to set the name variations and name identification markings. Example Borka



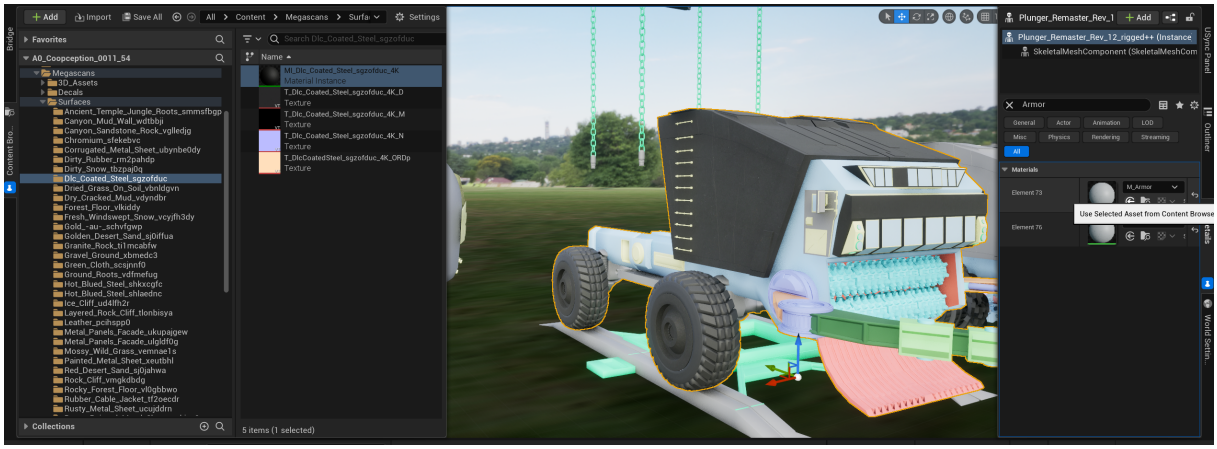
Salmon



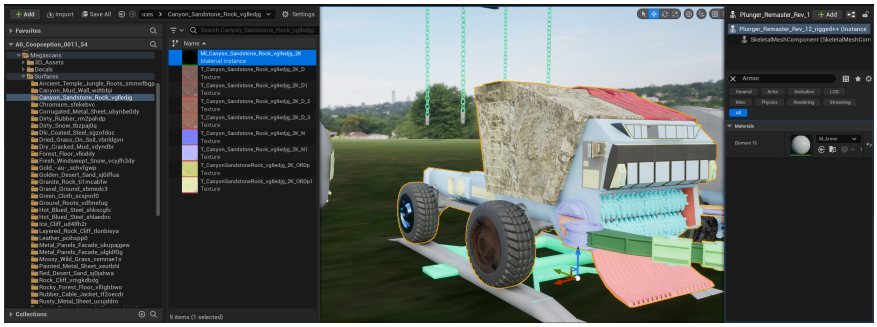
Names are inherited in Unreal, including mesh and material slots. They need to be correctly distinguished and named for each variation. We have a rename addon for this, and there's no way to avoid it. If previous phases are done incorrectly, the next step will not complete successfully.



If done correctly, you can see there are more than a hundred material slots. Does it look like it uses that many? No.



I can easily type all slots with the same name that are supposed to look alike in the right panel, select the material in the left panel, and just click a few times on all the apply material buttons in a matter of seconds. That means, like in real life, I can paint whole sections quickly or each one differently down to the minor realistic detail if desired.



So it is very important to get all of it right for technical reasons and premium features that increase price or attract enthusiasts.

C- Baking in the texture:
Bake textures to create texture maps that can be used in Unreal Engine. Use Lighting/Shading > Transfer Maps to bake textures like ambient occlusion, normal maps, etc.

▼ Resources

MAYA Baking

https://youtu.be/I9IVtq3wrbs

Substance 3d Baking

https://youtu.be/hYtHp4IXvsM

For Blender.

https://youtu.be/SDqpnfTRtIU

Addons.

For baking, we use the Bakemaster addon. However, if you have an equivalent in MAYA that you are comfortable with, please use that one. It just needs to be importable correctly.

https://bakemaster-blender-addon.readthedocs.io/en/latest/pages/about.html

For Unreal.

https://youtu.be/VNU_IGSkJ8s

Phase 4

Promotion renders:
A- Render images of your model for promotion. Set up lighting and cameras in Maya, then use Render > Render Sequence to create high-quality images.

▼ Resources

For MAYA.

https://youtu.be/xPgAolYL_qQ

For Blender.

https://youtu.be/VoHZ8lemEtk

For Unreal.

https://youtu.be/wjUhUhSQJBI

B- Packing & uploading for QA:
Prepare your model for quality assurance by organizing all files (e.g., textures, models, rig files). Compress them into a single package and upload it to your QA team or platform.

▼ Resources

In MAYA.

Quality assurance (QA) in Maya 3D models is crucial to ensure that the models meet the required standards and function correctly within a production pipeline. Here are some key aspects and tools to consider:

Key Aspects of QA in Maya 3D Models

1. **Geometry Checks**:
   - **Ngons**: Ensure there are no faces with more than four sides.
   - **Non-manifold Geometry**: Check for and fix any non-manifold edges or vertices.
   - **Normals**: Verify that all normals are facing the correct direction.
2. **UV Mapping**:
   - Ensure UVs are properly laid out and there are no overlapping UVs unless intended.
   - Check for proper UV scaling and alignment.

3. **Rigging and Skinning**:
   - Verify that the rig functions correctly and there are no broken connections.
   - Ensure skin weights are properly painted and there are no unexpected deformations.
4. **Textures and Materials**:
   - Check that all textures are correctly applied and there are no missing textures.
   - Ensure materials are correctly assigned and optimized for rendering.
5. **Animation**:
   - Verify that animations play back correctly and there are no unexpected issues.
   - Check for any keyframe issues or unintended movements.

Tools and Resources

- **Maya's Built-in Tools**:
  - **Select with Constraint Tool**: Useful for checking ngons and other geometry issues [1].
  - **UV Editor**: For managing and checking UV layouts.
  - **Hypershade**: For managing materials and textures.
- **Quality Assurance Frameworks**:
  - **GitHub Repositories**: There are QA frameworks available, such as the one by Robert Joosten, which provide collections for animators, modelers, riggers, and look-dev [2].
- **Community and Tutorials**:
  - **YouTube Tutorials**: There are many tutorials available that cover various aspects of QA in Maya. For example, tutorials on checking assets and increasing playblast quality [1].

Implementing a thorough QA process ensures that your 3D models are production-ready and meet the highest standards. If you have any specific questions or need further details on any of these aspects, feel free to ask!

In Blender.

Key Aspects of QA in Blender 3D Models

1. **Geometry Checks**:
   - **Ngons**: Ensure there are no faces with more than four sides.
   - **Non-manifold Geometry**: Check for and fix any non-manifold edges or vertices.
   - **Normals**: Verify that all normals are facing the correct direction.
2. **UV Mapping**:
   - Ensure UVs are properly laid out and there are no overlapping UVs unless intended.
   - Check for proper UV scaling and alignment.

3. **Rigging and Skinning**:
    - Verify that the rig functions correctly and there are no broken connections.
    - Ensure skin weights are properly painted and there are no unexpected deformations.
4. **Textures and Materials**:
    - Check that all textures are correctly applied and there are no missing textures.
    - Ensure materials are correctly assigned and optimized for rendering.
5. **Animation**:
    - Verify that animations play back correctly and there are no unexpected issues.
    - Check for any keyframe issues or unintended movements.

Tools and Resources

- **Blender's Built-in Tools**:
    - **Mesh Analysis**: Use the Mesh Analysis tool to check for issues like non-manifold geometry and intersecting faces.
    - **UV Editor**: For managing and checking UV layouts.
    - **Shader Editor**: For managing materials and textures.
- **Quality Assurance Frameworks**:
    - **Blender Add-ons**: There are various add-ons available that can help with QA tasks, such as the Mesh: 3D Print Toolbox for checking geometry [1].
- **Community and Tutorials**:
    - **YouTube Tutorials**: There are many tutorials available that cover various aspects of QA in Blender. For example, tutorials on optimizing render settings and improving render quality [2]

In Unreal.

Key Aspects of QA in Unreal Engine 5.4

1. **Geometry and Mesh Optimization**:
   - **Nanite**: Utilize Nanite for handling high-poly meshes efficiently.
   - **LOD (Level of Detail)**: Ensure proper LODs are set up for all assets to optimize performance.
2. **Textures and Materials**:
   - **Texture Streaming**: Verify that textures are correctly set up for streaming to manage memory usage.
   - **Material Instances**: Use material instances to optimize performance and ensure consistency.
3. **Lighting and Rendering**:
   - **Lumen**: Use Lumen for real-time global illumination and reflections.
   - **Anti-Aliasing**: Configure anti-aliasing settings to improve visual quality [1].
4. **Animation and Rigging**:
   - **Control Rig**: Use Control Rig for creating and managing character rigs.
   - **Sequencer**: Verify animations using the Sequencer tool to ensure smooth playback [2].
5. **Performance Testing**:
   - **Profiling Tools**: Use built-in profiling tools to monitor performance and identify bottlenecks.
   - **Automation Testing**: Implement automation tests for unit testing, feature testing, and content stress testing [3].

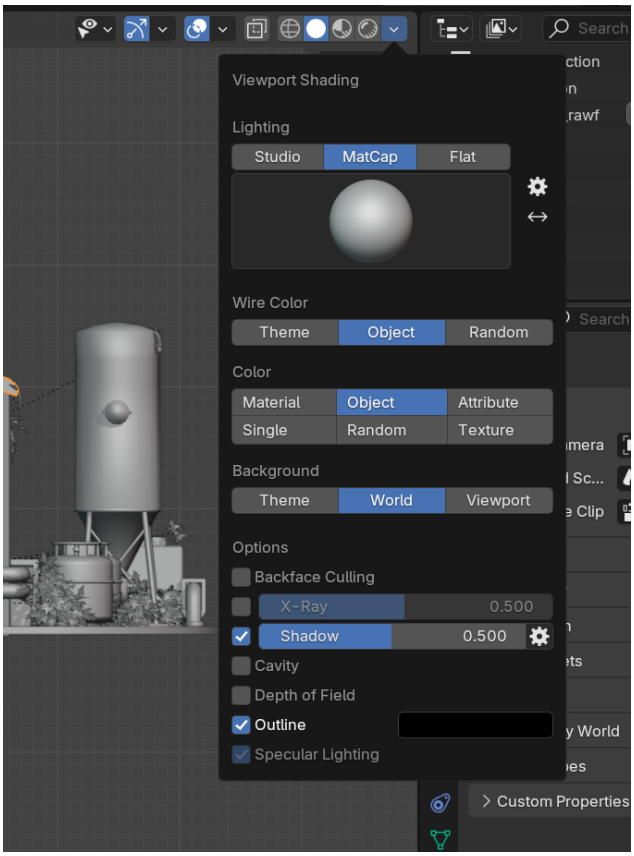
Tools and Resources

- **Unreal Engine Documentation**:
  - The official documentation provides detailed guides on various QA aspects, including testing and optimizing content [4].
- **Community and Tutorials**:
  - **YouTube Tutorials**: There are numerous tutorials available that cover different aspects of QA in Unreal Engine 5.4. For example, tutorials on animation tools and rendering high-quality frames [5] [6].

Topology expectations

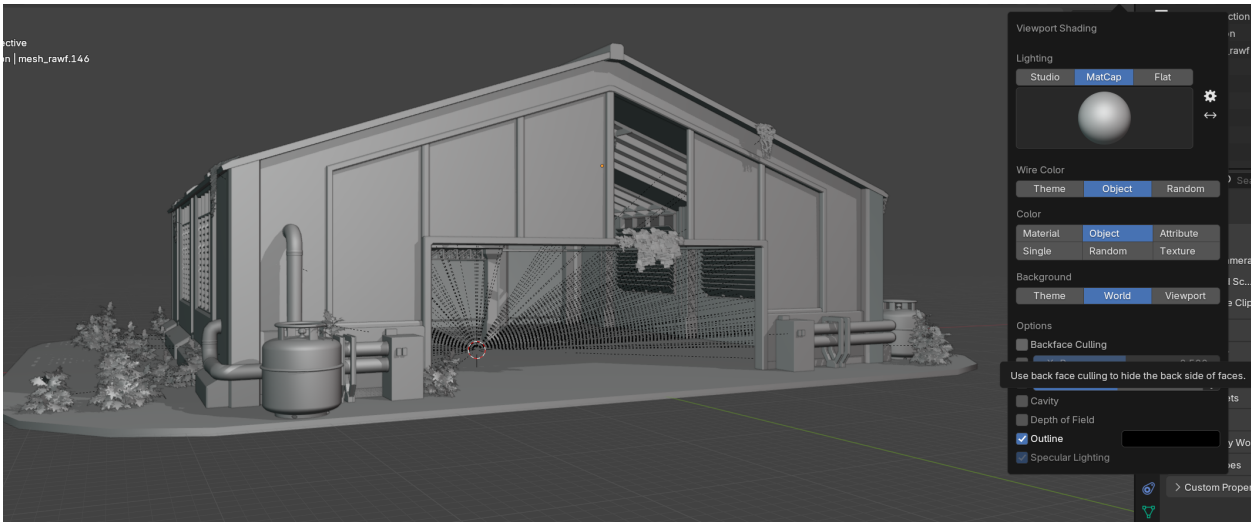
BLENDER

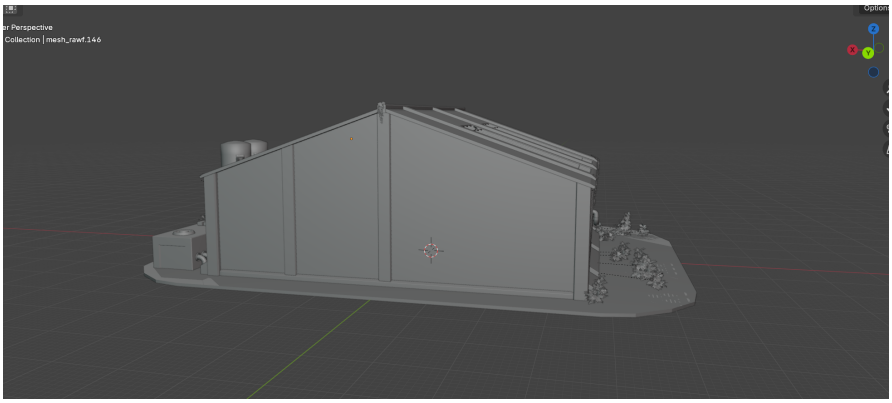
Do not forget to set your viewport to these settings to see face orientations. In UE, the import it not based on normals but on the face orientation.



IMPORTANT #1

EXAMPLE: A model that has seemingly no issues.





It will result in Backface Culling to show you the missing faces.

As of production, these models were not meant to be 'walk-in-able.' Therefore, they need some additional attention to fix this issue. All models should have double sides where you would expect them if you were to walk into them in real life.

▼ Resources

> https://cdn.discordapp.com/attachments/1301169666319847434/1301178796749230252/Blender_2024.10.30-08.37.mp4?ex=672f662f&is=672e14af&hm=bb74d97235c62e041711a7a461033229e414e52cf29d26fe62e4d05ff65df3ba&

IMPORTANT #2

If you work with glass material, make it a separate mesh from the rest of the object. UE does not support Nanite for transparent materials, so windows, glass, or anything transparent must be in its own mesh. Failure to do so means we can't use Nanite on the object, which results in performance loss.

A simple fix for a serious issue: just make windows and other transparent elements separate.

Comrade Lenin will help us with this task. As you can see, the issue is that Nanite with glass materials or anything transparent will not load. In a setting with hundreds or thousands of pieces, this can have a significant impact. The simple solution is to separate the transparent elements from the rest.

▼ Resources

https://cdn.discordapp.com/attachments/1301169666319847434/1301181990866911242/unknown_2024.10.30-08.50.mp4?ex=672f6929&is=672e17a9&hm=df640e291bb9f404d93dbc2604fe9de34c1efffa23d18fb79360b74a5c9150b3&

IMPORTANT #3

Unwrapping can be tedious, but we have a solution in the form of three addons: Unwarp Me - Auto UV, Symmetry++ to fix cross-face or edge topology issues, and Quad Star Fill for those super tedious circles.

1. https://blendermarket.com/products/unwrap-me
2. https://blendermarket.com/products/symmetry-
3. https://blendermarket.com/products/quad-star-fill

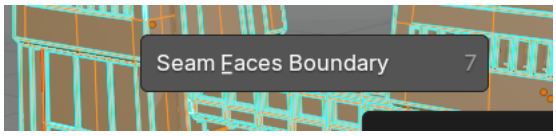
Study the descriptions and understand what they do, as this will save you countless hours.

What does it look like to work with this set of unwrapping tools? (applies when you don't need to work on the textures manually or by hand, for lets say prototypes or WIP's)

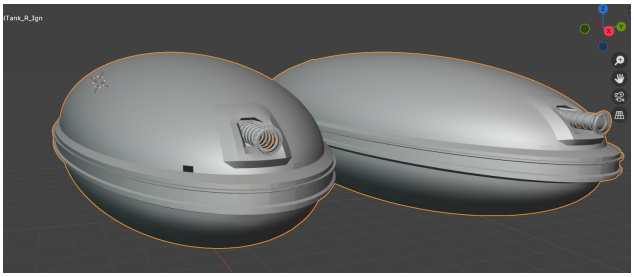
EXAMPLE

https://cdn.discordapp.com/attachments/1301169666319847434/1301188969580728521/Blender_2024.10.30-09.17.mp4?ex=672f6fa9&is=672e1e29&hm=523c1a68c5ea1741145a51f97fd9bea1c68f57f163df4fc031124861ecf2c1f9&

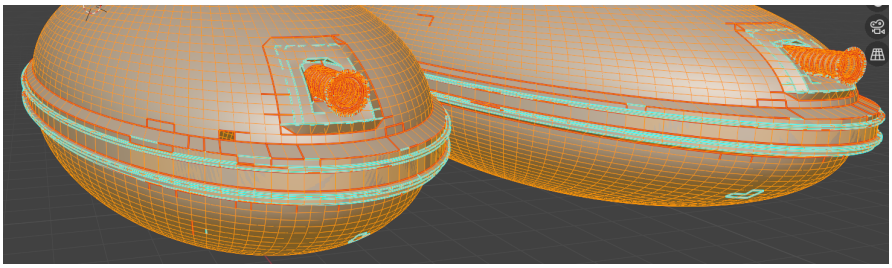
As you can see, it is a matter of a few clicks, and it works much better than the Unwrap function. It is also more compatible with UE. You should apply the textures after this point because, with the textures set, you cannot change the UVs anymore. So, it is very important to do this. This function also detects if Blender generated disconnected vertices or overlapping edges, as it marks the boundaries.



It can mark issues that are not visible in the standard viewport.



As this particular part of the model was kind of broken during its creation due to my lack of knowledge back in the day.



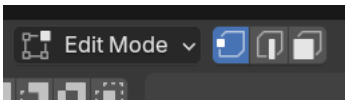
IMPORTANT #4

Use a UE5 mannequin to measure the scale of your buildings; we now use it as the standard. Also, set Blender's scale to centimeters. <Super important

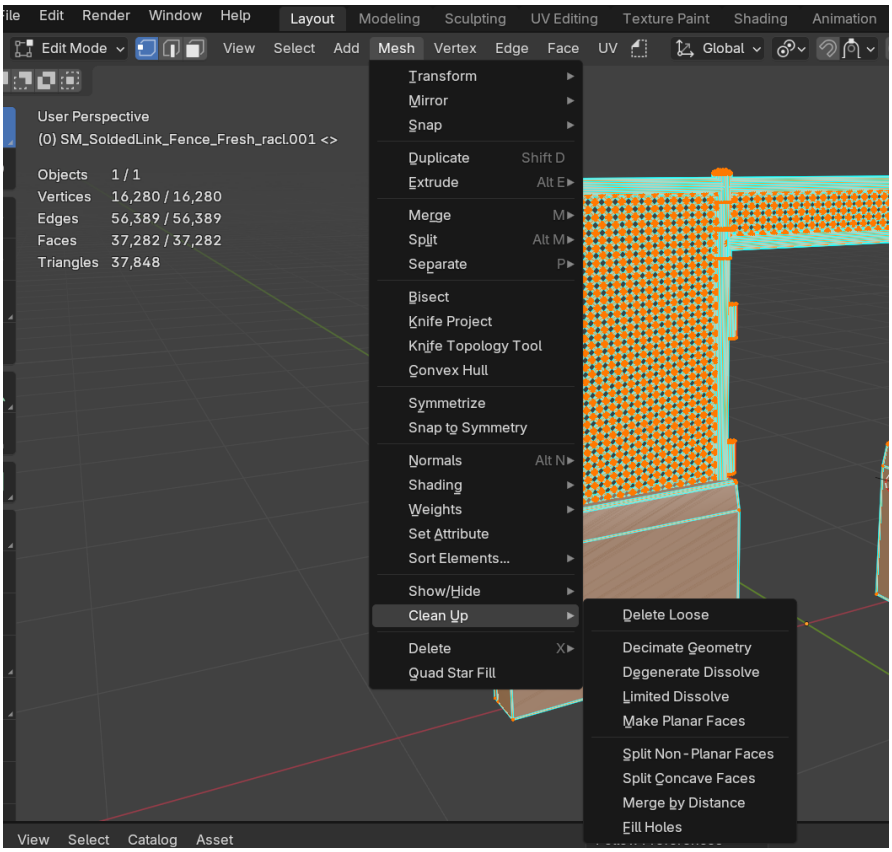
IMPORTANT #5

Blender sometimes creates unwanted vertices or "garbage". There is a quick fix you should apply before unwrapping, atlasing, or substance painting.

In Edit mode (when you edit the mesh):



In the Mesh tab, you'll find a wonderful set of tools called CleanUp. Let me show you how great it is on a seemingly ready mesh.



A great example is this soldered link fence.

https://cdn.discordapp.com/attachments/1301169666319847434/1301935134496264202/Blender_2024.11.01-10.43.mp4?ex=672f8394&is=672e3214&hm=5443a92f70f582f973f83ea97fab8fe67ce94b535bce730f717213968f89b9b6&

Notice the vertex count removed in the bottom pop-up, and it did not ruin the mesh at all. This is garbage being removed that was either created by user error or most likely by Blender misinterpreting how many times you clicked or due to engine lag.

Also, if your mouse or keyboard has a low battery or just does unreliable things.

It's an easy fix for making the model game-ready.

BE WARNED!

Merging vertices may dissolve two faces in some cases, creating holes in the mesh. Apply this fix before you address topology face issues. Use the correct viewport settings as mentioned in the above points to avoid this. There is nothing worse than discovering it after you have painted everything.

COMING SOON!

Phase 5 (For resident UE expert)

A. Unreal Blueprint Coding:
Implement all features to ensure the project is playable out of the box, adhering to requirements and specifications. Coding is performed in C++ and made accessible through Blueprint events. The project must be packageable and suitable for regular testing or sharing updates.

▼ Resources

B. Unreal Animation:
Create or import animation blueprints, sequences, and blends using a premade project.

▼ Resources

C. Game Readying & Optimization:
Utilize Nanite with meshless skeletons, virtual shadow maps, and Lumen for optimal performance. Ensure good Lightmass performance, resolution, and project cleanliness. Measure performance through metrics.

▼ Resources