mcblend

Release 10.2.0

Nusiq

Jul 15, 2023

OVERVIEW

CHAPTER

ONE

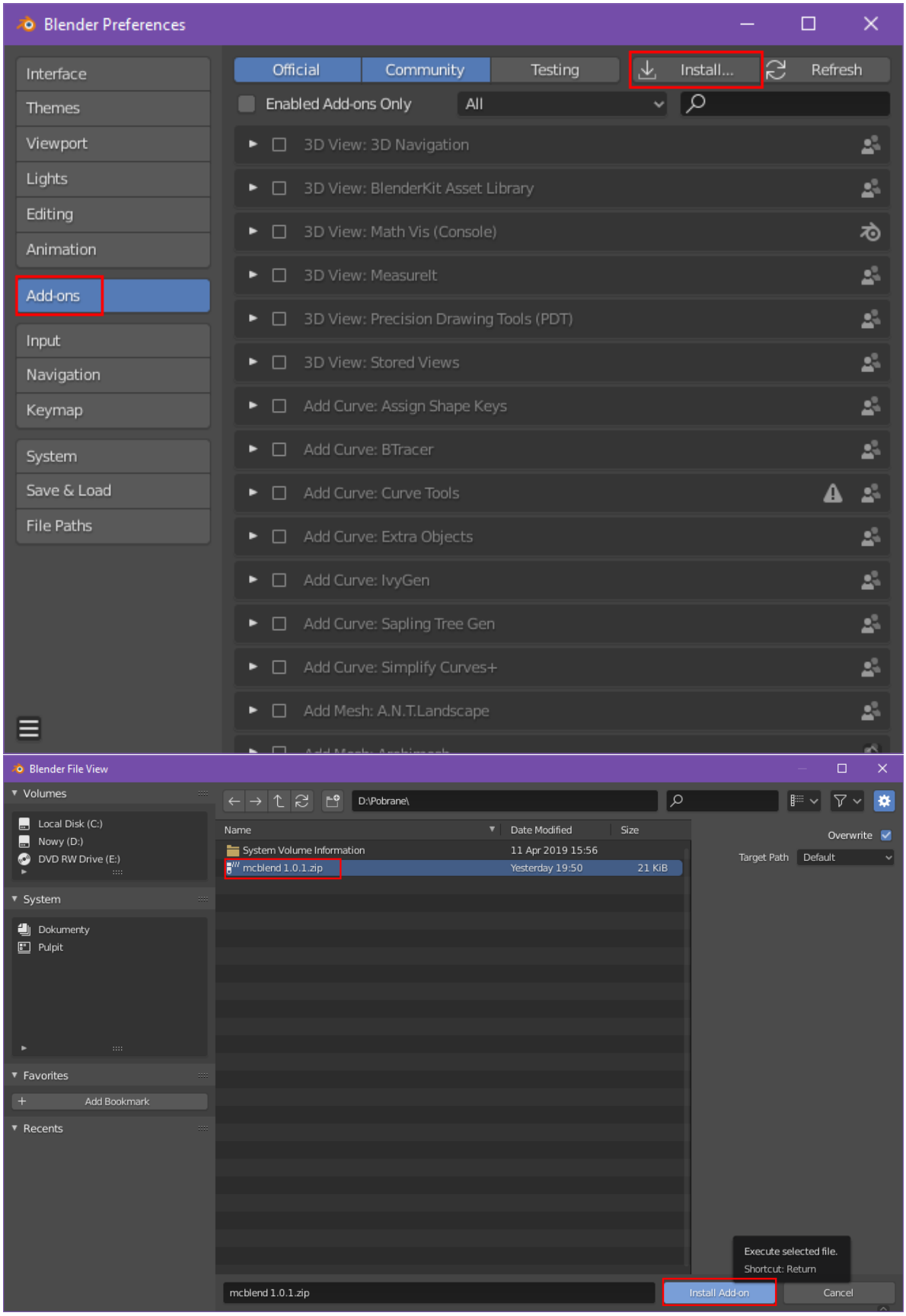
INSTALLATION

1. Download and install Blender 3.3 (the current LTS version) from the official website: https://www.blender.org/download/releases/3-3/
2. Download the latest version of Mcblend from the project page on GitHub: https://github.com/Nusiq/mcblend/releases.

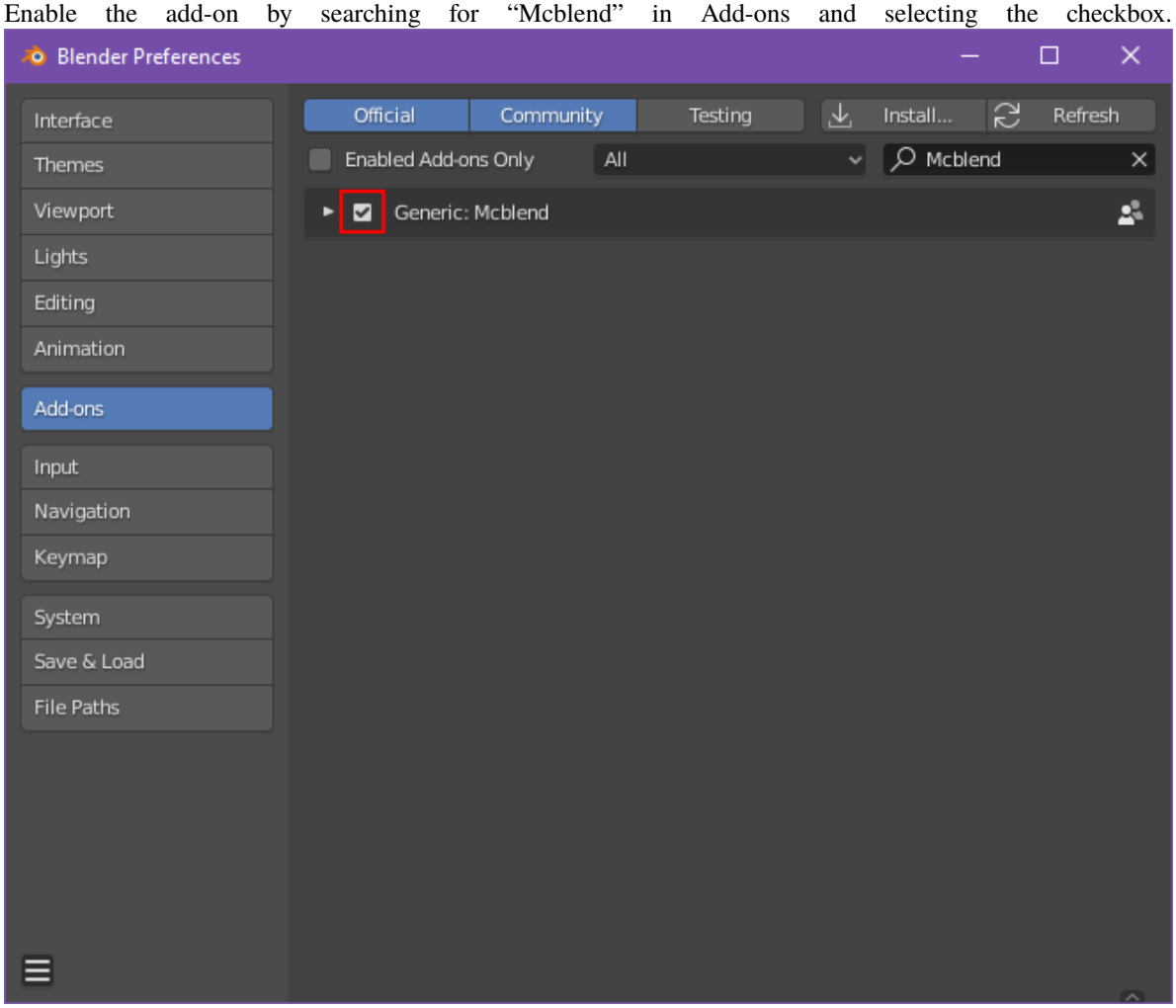
Note: If a pre-release version is available, it may include additional features and bug fixes. The pre-release versions are stable but not yet fully documented. The user interface may also differ from the version described in the documentation.

---

3. In Blender, go to `Edit -> Preferences -> Add-ons -> Install...` and select the downloaded zip file.


Blender Preferences
Interface
Themes
Viewport
Lights
Editing
Animation
Add-ons
Input
Navigation
Keymap
System
Save & Load
File Paths
Official
Community
Testing
Install...
Refresh
Enabled Add-ons Only
All
3D View: 3D Navigation
3D View: BlenderKit Asset Library
3D View: Math Vis (Console)
3D View: MeasureIt
3D View: Precision Drawing Tools (PDT)
3D View: Stored Views
Add Curve: Assign Shape Keys
Add Curve: BTracer
Add Curve: Curve Tools
Add Curve: Extra Objects
Add Curve: IvyGen
Add Curve: Sapling Tree Gen
Add Curve: Simplify Curves+
Add Mesh: A.N.T.Landscape
Blender File View
Volumes
Local Disk (C:)
Nowy (D:)
DVD RW Drive (E:)
System
Dokumenty
Pulpit
Favorites
Add Bookmark
Recents
D:\Pobrane\
Name
Date Modified
Size
System Volume Information
11 Apr 2019 15:56
mcblend 1.0.1.zip
Yesterday 19:50
21 KiB
Overwrite
Target Path
Default
Execute selected file.
Shortcut: Return
mcblend 1.0.1.zip
Install Add-on
Cancel

4. Enable the add-on by searching for “Mcblend” in Add-ons and selecting the checkbox.



1.1 Updating

To update Mcblend, you must uninstall the old version and then install the new one. There is currently no system in place for updating the add-on directly within Blender.

CHAPTER

TWO

LIMITATIONS

Minecraft models and animations have certain constraints that must be followed. Blender was not designed specifically for Minecraft, so there are a few limitations to be aware of when using Mcblend. This section outlines these limitations and provide guidelines for working within them.

2.1 Modeling

In order to use Mcblend, you must adhere to the following modeling constraints:

- Each cube and polymesh must be a separate object.
- The model must have exactly one armature.
- Cubes and polymeshes must be parented to the bones of the armature using "bone parenting.".
- Bones in the armature represent the bones of the Minecraft model, and the objects parented to them represent cubes or polymeshes.
- Empties can be used as Minecraft's "locators."
- Objects and empties can be parented to other objects (but not to empties) as long as there is a bone at the top of their hierarchy.
- If a bone has no parents or children, it will be ignored during export. This is useful because in Blender, you may sometimes want to use such bones for inverse kinematics.

Note: Mcblend provides tools to make it easier to follow these rules. The *"separate cubes operator"* can help you separate meshes that contain multiple cubes into multiple objects with properly aligned rotations. This means you can create your model in a single mesh and, as long as its parts have proper shapes, you can separate them into a format that can be used by Mcblend.

You can also mark certain objects as *"polymeshes,"* which allows you to use shapes other than cuboids for your model. However, be aware that polymeshes are an experimental feature in Minecraft and may be removed from the game in the future.

Modeling limitations are the result of the format of Minecraft's model files. As shown in the code snippet below, a Minecraft model is made up of bones, with each bone containing a list of cubes and/or a single polymesh. Each cube and polymesh has its own pivot and rotation, and Mcblend needs this information in order to export the model correctly. This means that it is not possible to pack everything into a single mesh, as a mesh is simply a collection of vertices without a concept of rotation of its separate parts. Instead, you must create separate meshes for each cube and polymesh.

```
{
    "format_version": "1.16.0",
    "minecraft:geometry": [
        {
            "description": {
                ...
            },
            "bones": [
                {
                    "name": "my_bone",
                    "pivot": [0, 0, 0],
                    "rotation": [90, 0, 0],
                    "locators": {
                        "my_locator": {
                            "offset": [0, 0, 0],
                            "rotation": [-45, 0, 0]
                        }
                    },
                    "cubes": [
                        {
                            "uv": [0.0, 0.0],
                            "size": [32, 32, 32],
                            "origin": [-16, -16, -16],
                            "pivot": [0, 0, 0],
                            "rotation": [-90, 0, 0]
                        },
                        {
                            ...
                        }
                    ],
                    "poly_mesh": {
                        ...
                    }
                },
                {
                    "name": "my_bone2",
                    "parent": "my_bone",
                    ...
                }
            ]
        }
    ]
}
```

The rule of using a single armature per Minecraft model helps to make the mapping between the model in Blender and the model in Minecraft more intuitive and easier to understand. It also simplifies the process of working with multiple models. In earlier versions of Mcblend, it was possible to use hierarchies where some bones were represented by empties, but this made the models confusing and difficult to interpret, so the feature was removed. By enforcing the use of a single armature, it becomes clearer how the various parts of the model in Blender correspond to their counterparts in Minecraft.

2.2 UV mapping

In Minecraft, the UV maps of the faces of cubes must be rectangular and aligned with the orientation of the texture image. They also must be properly rotated to match the rotations allowed by Minecraft. A proper rotation is when:

- The front, back, left, and right faces have their top and bottom edges aligned horizontally, with the remaining edges being vertical.
- The top and bottom faces have their front and back edges aligned horizontally, and their left and right edges aligned vertically.

Note: It can be difficult to understand these alignment rules when looking at a model with rotated cubes. However, Mcblend provides the *Fix invalid UV mapping* operator to rearrange the UVs of selected objects to match the Minecraft rules. If you see warnings in your export such as: `Cube based on Blender object "Cube": "north" face has invalid UV mapping. Skipped.` you can use this operator to fix the UV mapping.

There are two types of UV mapping in Minecraft: per-face UV mapping and default UV mapping. The default UV mapping is not very flexible, as the size and position of the faces are based on the size of the cube. The vector passed to the "uv" property defines the offset. With Mcblend, you don't have to worry about the type of UV mapping you use. If the faces are arranged in a way that allows saving the UV in default format, Mcblend will do so (because it is more compact). Otherwise, the UV is saved using the per-face mapping format.

Unfortunately, the per-face UV mapping is also limited. It cannot rotate the UV by 90 degrees. It uses two vectors to define the mapping of the face: the "uv" (offset) and the "uv_size". This format allows for flipping the rectangle, but not rotating it.

Examples of both types of UV mapping in code are shown below:

The default UV mapping:

```
"uv": [0.0, 64.0],
```

The per-face UV mapping

```
"uv": {
    "north": {"uv": [48.0, 32.0], "uv_size": [32.0, -32.0]},
    "east": {"uv": [48.0, 128.0], "uv_size": [32.0, -32.0]},
    "south": {"uv": [48.0, 96.0], "uv_size": [32.0, -32.0]},
    "west": {"uv": [48.0, 64.0], "uv_size": [32.0, -32.0]},
    "up": {"uv": [112.0, 64.0], "uv_size": [-32.0, -32.0]},
    "down": {"uv": [16.0, 32.0], "uv_size": [32.0, 32.0]}
},
```

2.3 Animating

There is a limitation on the amount of rotation that can occur between two keyframes in an animation created with Mcblend. The maximum amount of rotation allowed is 180°. If the rotation between two keyframes exceeds this amount, the exported animation will not match the preview in Blender.

Note: A quick fix for this issue is to add additional keyframes for wide angle rotations.

This limitation is a result of the way Mcblend calculates rotations internally.

Blender supports multiple rotation modes and uses different types of rotations for different kinds of objects. For example, bone rotations in armatures use quaternions, while meshes use Euler angles. Additionally, users can choose different rotation modes for each object. Minecraft uses Euler angles, but the axes are set differently.

Mcblend can export models and animations regardless of the rotation modes used, but internally everything is converted to quaternions or translation matrices. The decision to use quaternions internally was made because they help avoid certain calculation errors.

However, the quaternion number system has only one unique representation for each rotation orientation, so it is not possible to distinguish a full rotation (360°) from no rotation (0°). This means that angles greater than 180° between two keyframes cannot be used, as Mcblend will always try to export the smallest possible rotation to the animation.

CHAPTER

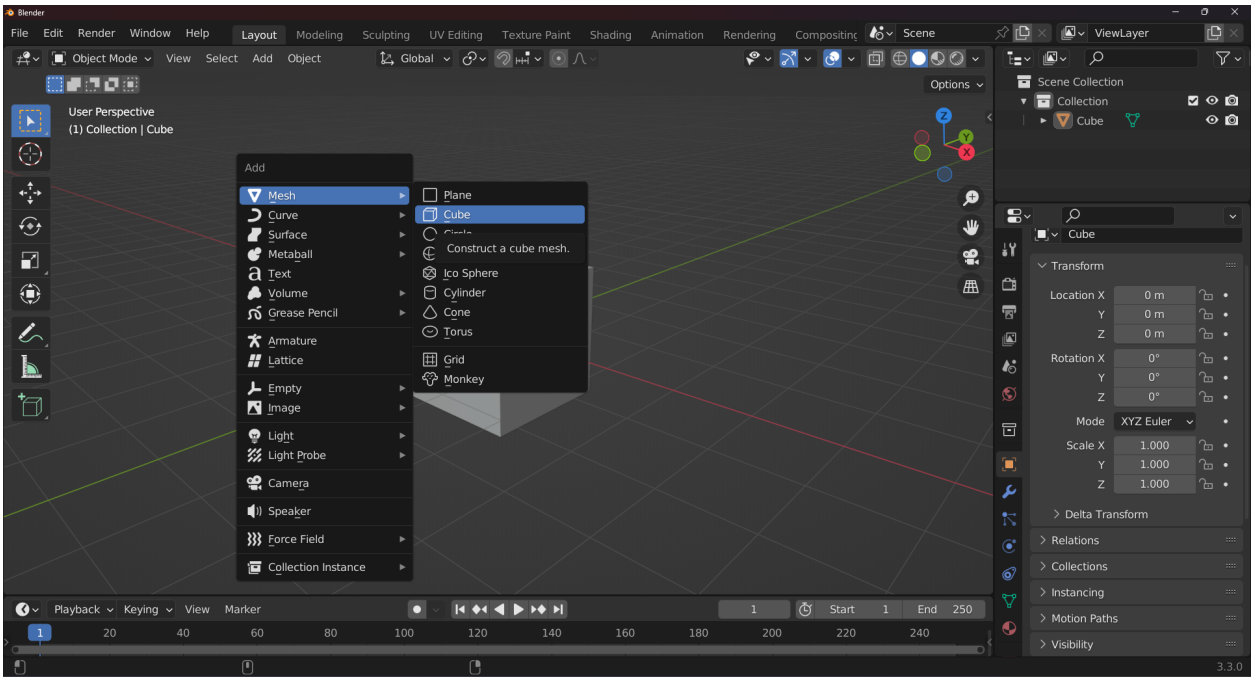
THREE

CREATING MODELS FROM SCRATCH

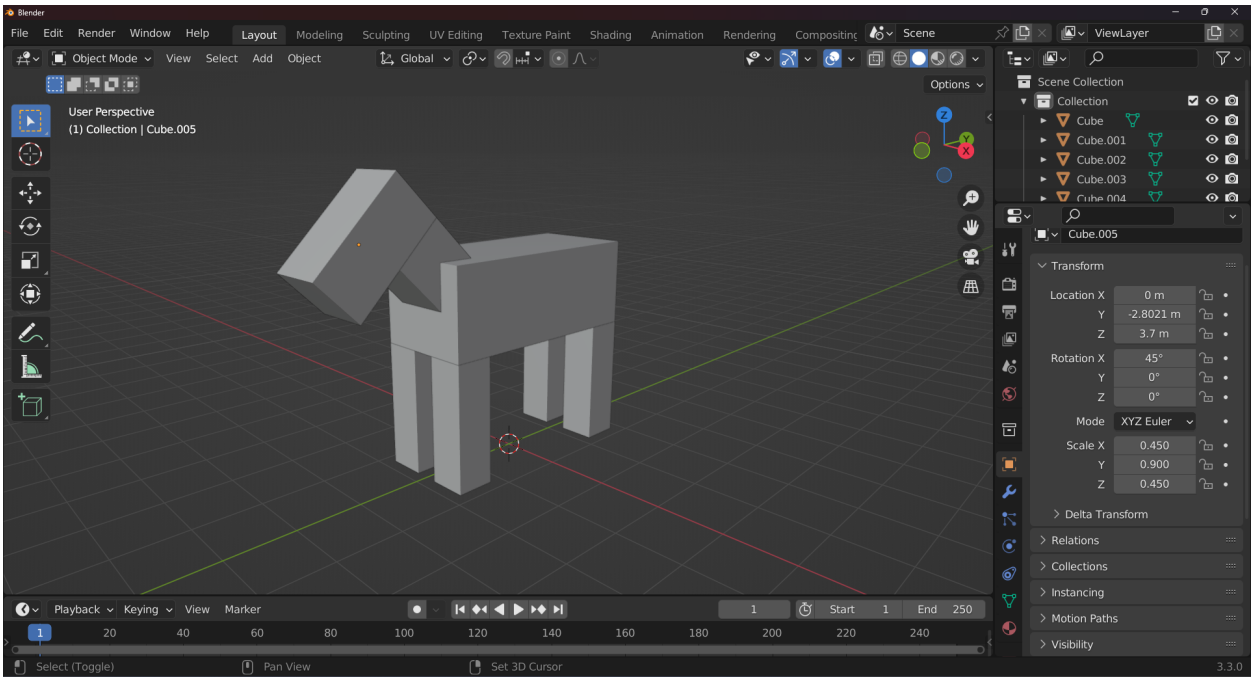
On this page, you will learn how to create a Minecraft model using Mcblend from the very beginning. This includes adding cubes to the scene, creating an armature, parenting the cubes to the armature, generating the UV maps and texture for the model, and exporting the final model for use in Minecraft. This is a minimal explanation and some of the details are left out. You can find more information about the various steps in the rest of the documentation.

3.1 Adding cubes

Assuming your scene is empty, start by adding cubes to it. To add a cube to your scene, press `Shift+A` and select `Mesh` > `Cube` from the menu that appears.

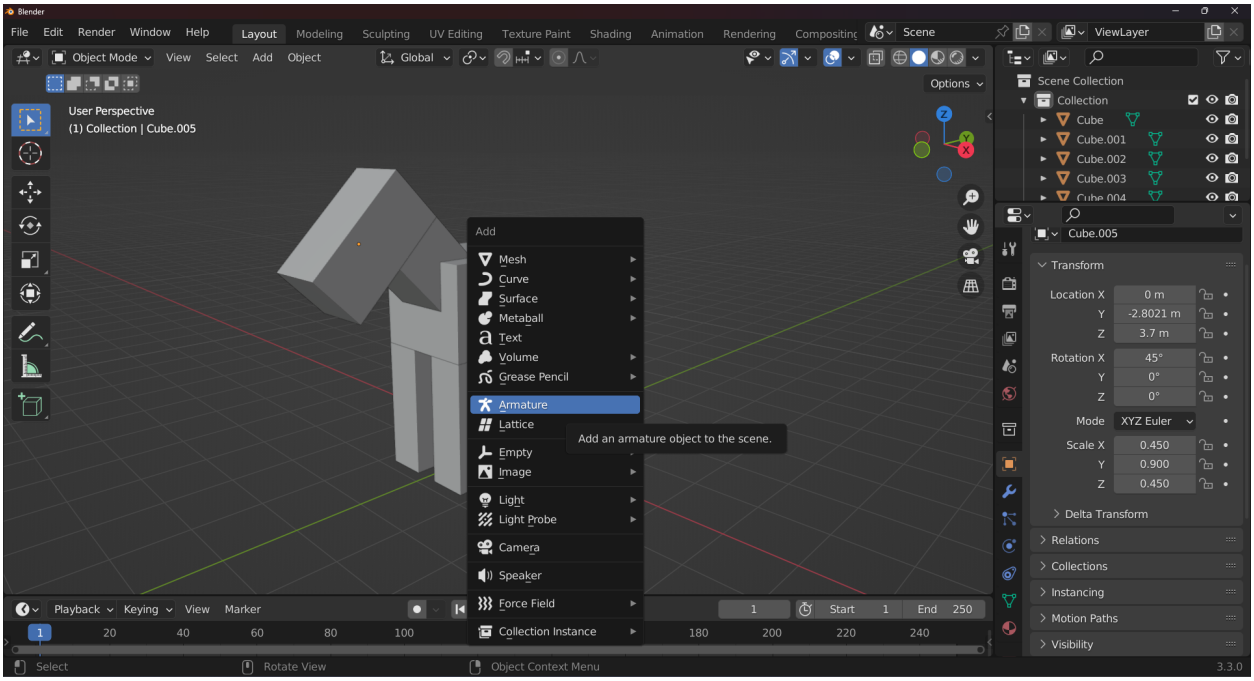


You can then transform the cube in object mode or edit mode by using the various transform tools, such as moving, rotating, and scaling. Just be careful not to deform the shape of the cube when using edit mode, as the cube must still have a cuboid shape and be aligned to the axis of the object that holds the mesh. Once you have added and transformed your cubes, you can proceed to the next steps of creating your Minecraft model.



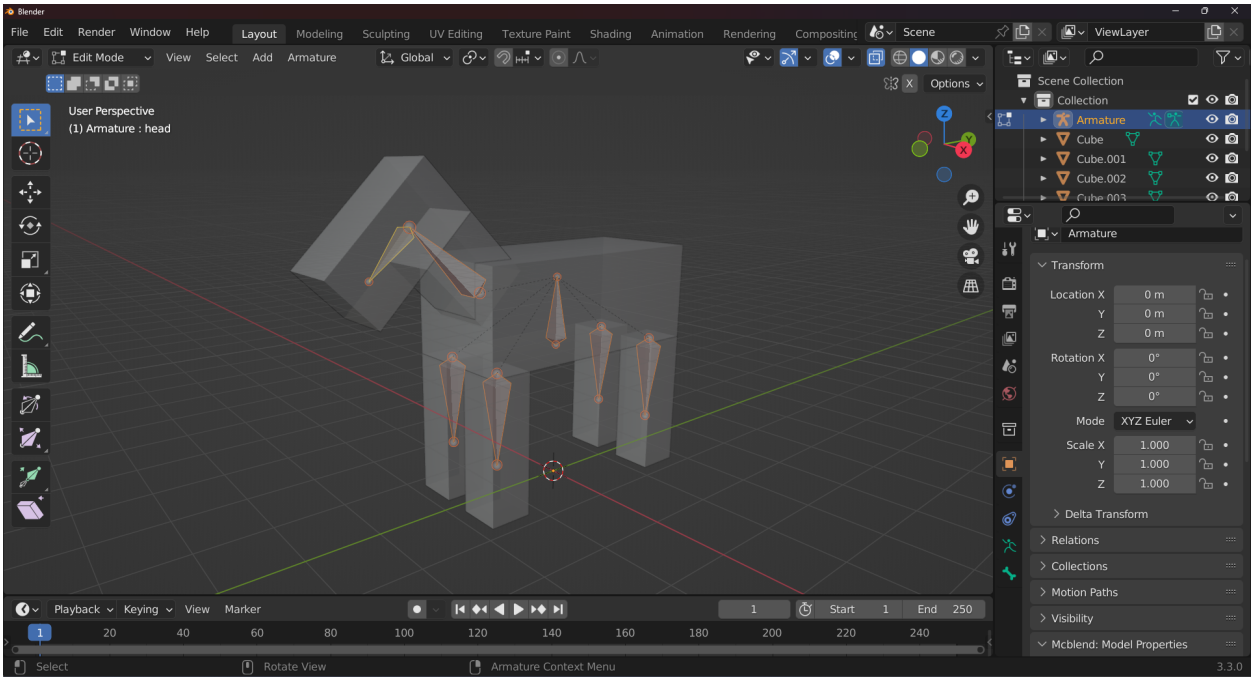
3.2 Creating an armature

Next, create an armature for your model. To create an armature, use the `Shift+A` shortcut and select `Armature` from the menu that appears.



To add bones to your armature in Mcblend, first select the armature object in Object Mode. Then, enter Edit Mode and use the `Shift+A` shortcut to add bones to the armature. Position and orient the bones to match the model you are creating, paying careful attention to the placement and orientation of each bone in relation to the model. Remember to name your bones properly using the `F2` shortcut. The bones don't need to be connected in Minecraft, as the length of

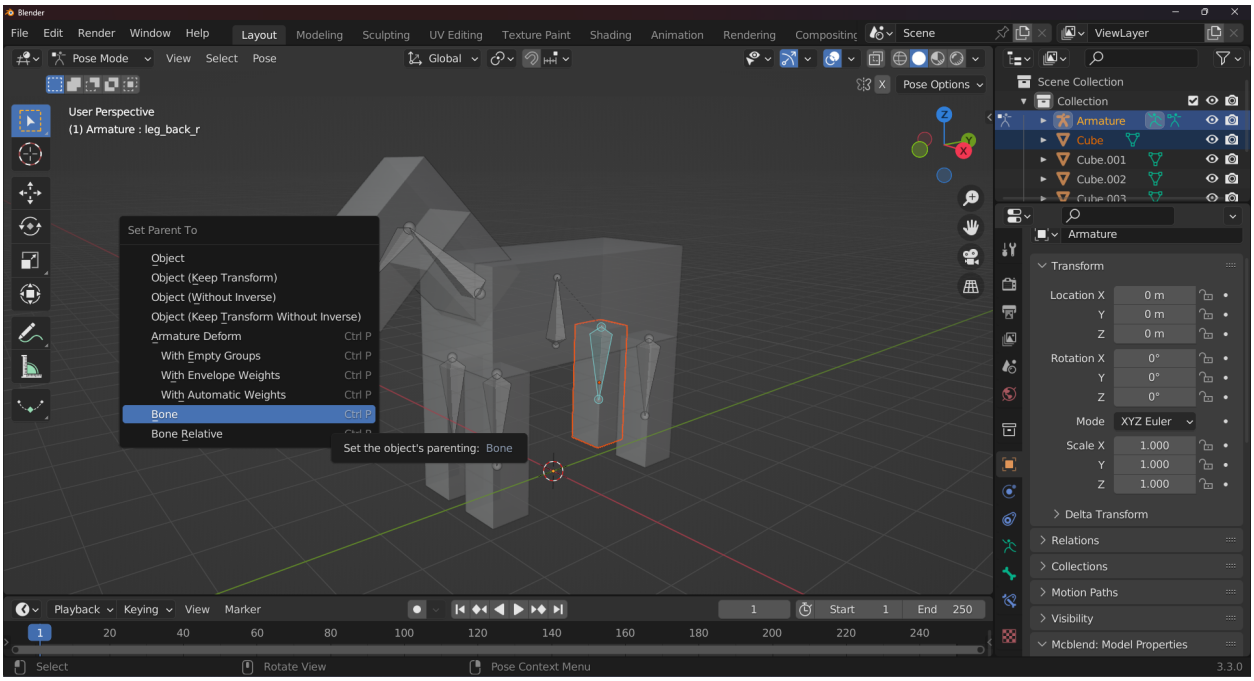
the bone is not used and is lost when the model is exported. However, it is still a good idea to create an armature that makes sense in Blender, as it will be easier to work with.



3.3 Parenting the cubes to the armature

When you're done with adding bones to the armature, you can exit Edit Mode and start parenting the cubes to it. **Mcblend requires that you use the specific Bone parenting mode.** This mode is rarely used, so you may not be familiar with it. In order to parent an object to a bone using the Bone parenting mode, you have to follow these steps:

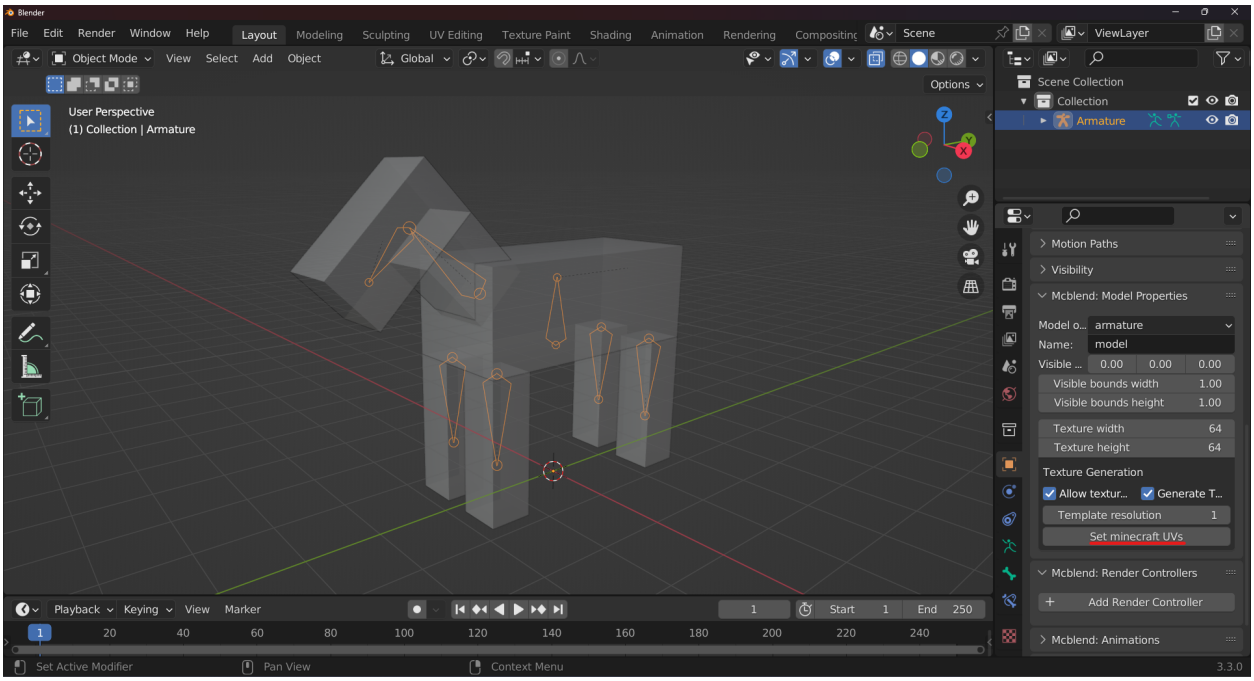
1. Select the armature and enter Pose Mode.
2. Select the cube you want to parent to the bone using the Outliner menu on the right of the screen.
3. Select the cube by clicking it in the 3D viewport. At this point the bone should be highlighted in a light blue color and the cube should be highlighted in an orange color (see image below).
4. Parent the cube to the bone using the `Ctrl+P` shortcut and selecting the Bone option.



Repeat this process until all the cubes in your model are parented to the armature.

3.4 UV mapping and texturing

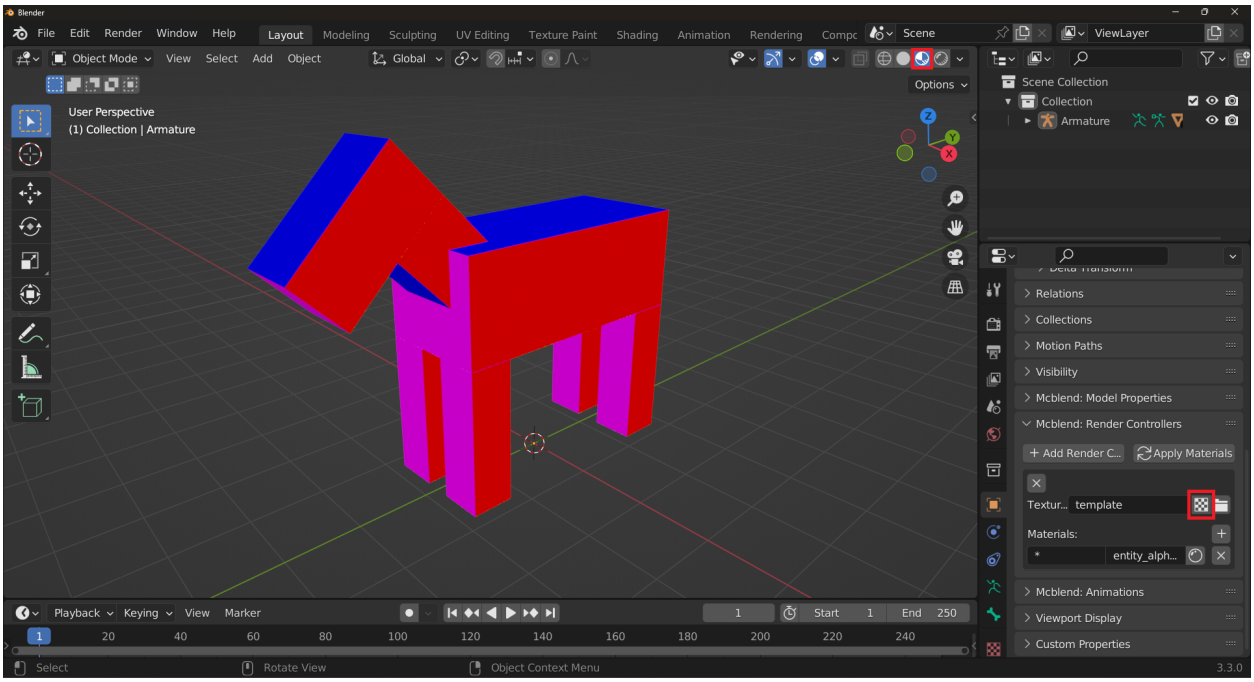
Generate the texture for your model by selecting the armature in Object Mode and pressing the `Set minecraft UVs` button in the `Object Properties` panel in the `Mcblend: Model Properties` tab. This will adjust the UV maps of the cubes connected to the armature, and create a texture called `template`.



The texture won't be connected to the model automatically. To do this, you'll need to use the *Render controllers* feature

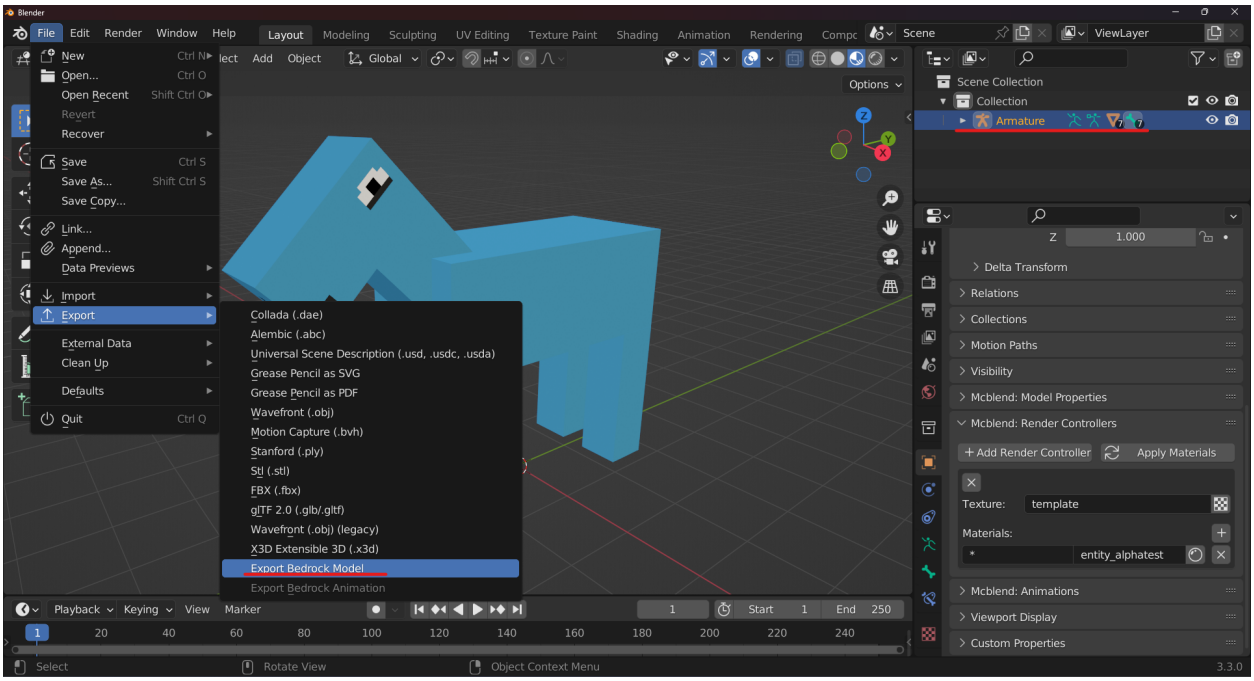
in Mcblend. These work in a similar way to the render controllers in Minecraft. They define which textures should be used by the model, which cubes they should be applied to, and the properties of the material of each cube. Internally, in Mcblend, render controllers are just a tool for generating Blender materials similar to those used in Minecraft.

To generate a material for your model, select the armature in Object Mode and go to the `Mcblend: Render Controllers` tab. Then, press the `Add Render Controller` button to create a box with additional properties. You can ignore most of these properties for now. To apply the `template` texture to your model, use the button with a texture icon next to the `Texture` field to select the `template` texture, and then press the `Apply Material` button. This will apply the material to the cubes of your model. You can view the textures on the model by switching the viewing mode in the 3D viewport to `Material Preview`.



The texture generated by Mcblend is an internal Blender resource. You can can save it just like any other texture by going to the `Image Editor` and pressing the `Save Image` button. To further customize the look of your model, you can edit the generated texture in an external software like Gimp.

Finally, you can export your model by selecting the armature and pressing the `Export Bedrock Model` button in the `File > Export` menu. The exported JSON file can then be used as a model in Minecraft. For more information on configuring your model before exporting, including additional settings and options, see the *"Exporting models"* section.


File
Edit
Render
Window
Help
Layout
Modeling
Sculpting
UV Editing
Texture Paint
Shading
Animation
Rendering
Scene
ViewLayer
New
Open...
Open Recent
Revert
Recover
Save
Save As...
Save Copy...
Link...
Append...
Data Previews
Import
Export
External Data
Clean Up
Defaults
Quit
Collada (.dae)
Alembic (.abc)
Universal Scene Description (.usd, .usdc, .usda)
Grease Pencil as SVG
Grease Pencil as PDF
Wavefront (.obj)
Motion Capture (.bvh)
Stanford (.ply)
Stl (.stl)
FBX (.fbx)
glTF 2.0 (.glb/.gltf)
Wavefront (.obj) (legacy)
X3D Extensible 3D (.x3d)
Export Bedrock Model
Export Bedrock Animation
Scene Collection
Collection
Armature
Delta Transform
Relations
Collections
Motion Paths
Visibility
Mcblend: Model Properties
Mcblend: Render Controllers
Add Render Controller
Apply Materials
Texture:
template
Materials:
entity_alphatest
Mcblend: Animations
Viewport Display
Custom Properties
Playback
Keying
View
Marker
Start
End
Select
Rotate View
Object Context Menu
3.3.0

CHAPTER

FOUR

IMPORTING MODELS FROM A FILE

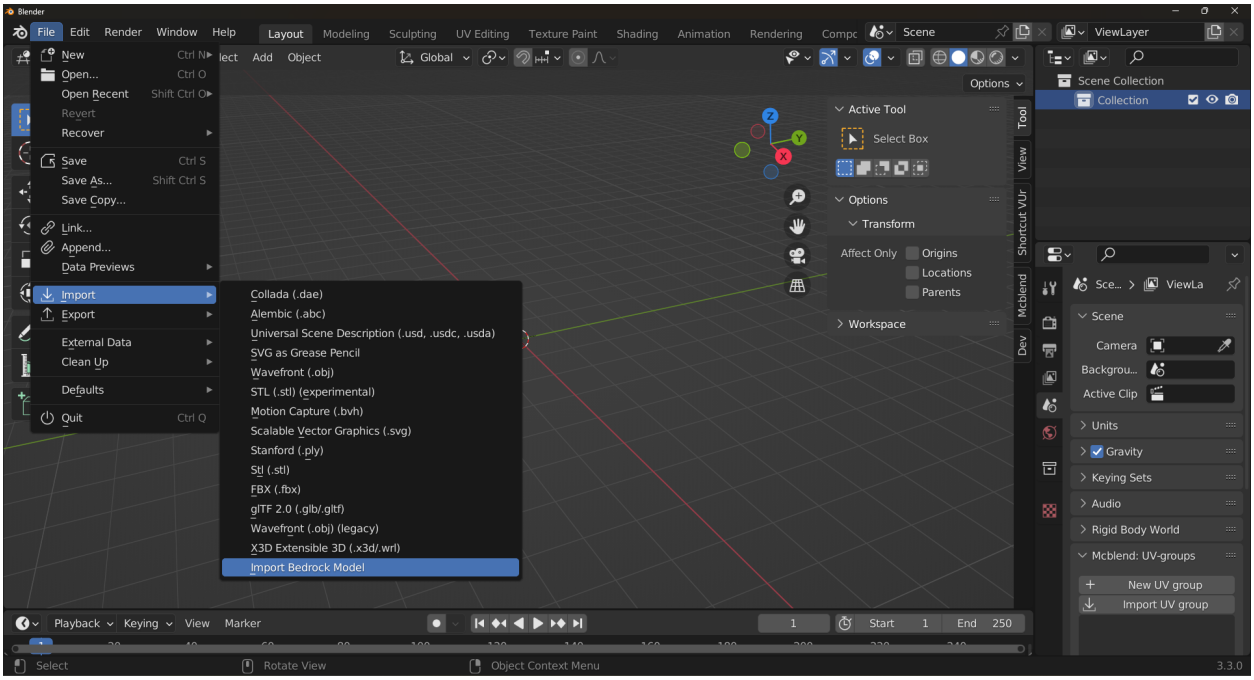
In this page, you will learn how to import a Minecraft model into Mcblend from a file. This is useful if you want to modify an existing model or if you have a custom model created outside of Mcblend.

Note: It is recommended to *import models from resource* packs whenever possible, as it will automatically set up the texture and render controllers for you using the information from the resource pack. You can learn how to import models from resource packs in the next page.

4.1 Importing the model

To import a Minecraft model from a file, go to `File > Import > Import Bedrock Model`. This will open a file browser where you can select the model file. The model files usually have the `.geo.json` extension.

After selecting the file, the model will be imported into your Blender scene. The model will be represented by a collection of cubes parented to an armature.



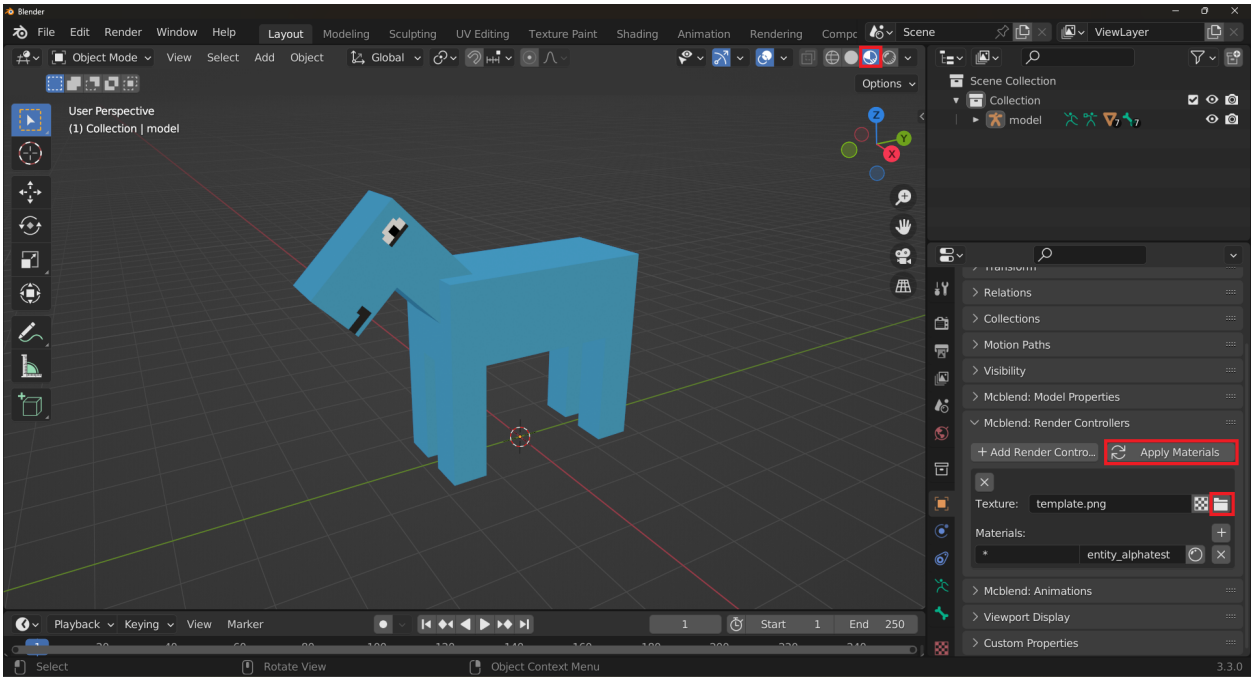
4.2 Importing and applying the texture to the model

To import the texture and apply it to your model, you'll need to use the Render controllers feature in Mcblend. These work in a similar way to the render controllers in Minecraft, defining which textures should be used by the model, which cubes they should be applied to, and the properties of the material of each cube.

To import and apply the texture to your model:

1. Select the armature object in `Object Mode` and go to the `Mcblend: Render Controllers` tab in `Object Properties`.
2. Press the `Add Render Controller` button to create a new render controller.
3. Click on the folder icon next to the `Texture` field and select the texture file using the file browser.
4. Press the `Apply Material` button to apply the material to the model.

You can view the textures on the model by switching the viewing mode in the 3D viewport to Material Preview.



CHAPTER

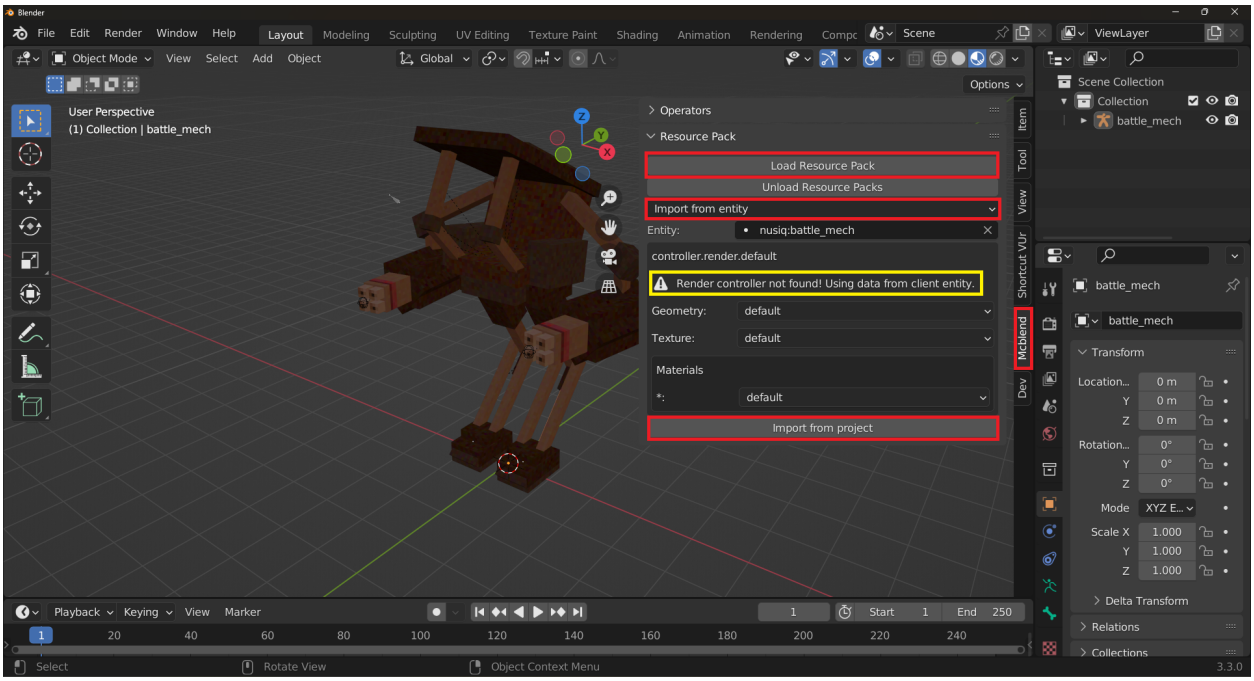
FIVE

IMPORTING MODELS FROM RESOURCE PACKS

Importing models from resource packs is a quick and easy way to bring Minecraft models into Mcblend. It allows you to automatically set up the texture and render controllers for you using the information from the resource pack.

To import a model from a resource pack:

1. Open the sidebar in the 3D viewport by pressing `N`.
2. In the Mcblend tab, press the `Load Resource Pack` button. You can load as many resource packs as you want, and if multiple packs define the same thing (such as a texture, model, or render controller), the definition of the most recently loaded pack will be used. It's like having multiple packs in a Minecraft world.
3. Select the entity or attachable you want to import from the dropdown list. Mcblend will be able to figure out which render controller, texture, and model should be used.
4. All of the render controllers used by the entity will be listed. Each render controller will have lists for geometry, texture, and material (there can be multiple material lists per render controller if it defines multiple materials for different bones). In most cases, the dropdown lists will have only one option to select from, but more complex render controllers may require user input. If a render controller is not found, you will see a warning that says *"Render controller not found! Using data from client entity."* This is not a mistake, as some hard-coded render controllers are used in Minecraft. In this case, Mcblend will assume that the render controller you selected uses one material, one texture, and one model, and it will allow you to choose from every material/texture/model defined in the entity/attachable definition.
5. After selecting the entity and its render controller configuration, press the `Import from project` button. This will load all of the models, textures, and render controllers of the entity/attachable based on the provided configuration.



This may seem complicated, but in most cases you just need to select an entity and press the `Import from project` button, as usually there is no advanced configuration of the render controller. However, Mcblend does support advanced entities that have multiple render controllers with complex configuration.

CHAPTER

SIX

EXPORTING MODELS

This section explains how to export Minecraft models and mentions some additional settings that can be configured before exporting.

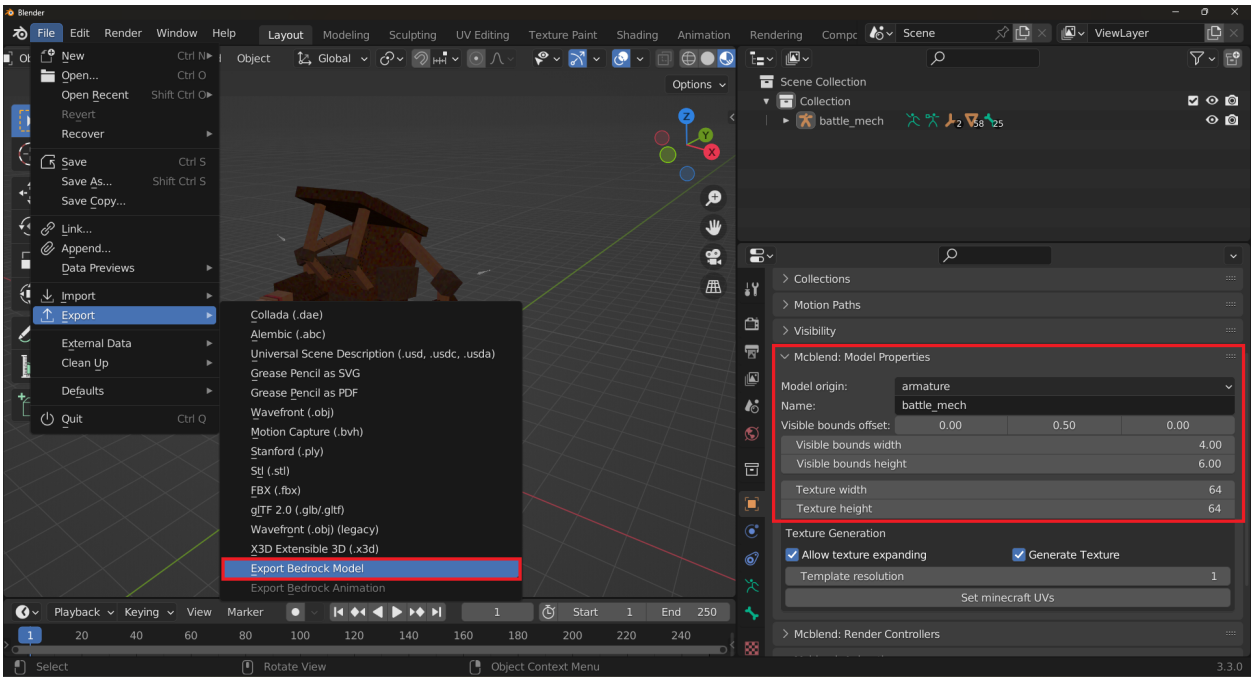
There are a few settings in the `Object Properties` panel in the `Mcblend: Model Properties` tab that can be configured before exporting the model. This tab is only visible when an armature is selected.

- `Model origin` - This setting can be set to either `armature` or `world`. The default value is `armature`, which means that the positions of the bones are based on the local space of the armature. If you select `world`, the positions will be based on the world space.

The other settings in the `Mcblend: Model Properties` tab are directly exported to the Minecraft model:

- `Name` - The name of the model. In Minecraft, models are named using the pattern *geometry.*, but you don't have to put geometry. in this field as it's always the same and never changes (it's added automatically).
- `Visible bounds offset` - This value is directly moved to the `visible_bounds_offset` property of the model.
- `Visible bounds width` - This value is directly moved to the `visible_bounds_width` property of the model.
- `Visible bounds height` - This value is directly moved to the `visible_bounds_height` property of the model.

Exporting the model is the same process as explained in the *"Creating models from scratch"* documen. Simply go to `File > Export > Export Bedrock Model` and select the export path.


File
Edit
Render
Window
Help
Layout
Modeling
Sculpting
UV Editing
Texture Paint
Shading
Animation
Rendering
Scene
ViewLayer
New
Open...
Open Recent
Revert
Recover
Save
Save As...
Save Copy...
Link...
Append...
Data Previews
Import
Export
External Data
Clean Up
Defaults
Quit
Collada (.dae)
Alembic (.abc)
Universal Scene Description (.usd, .usdc, .usda)
Grease Pencil as SVG
Grease Pencil as PDF
Wavefront (.obj)
Motion Capture (.bvh)
Stanford (.ply)
Stl (.stl)
FBX (.fbx)
glTF 2.0 (.glb/.gltf)
Wavefront (.obj) (legacy)
X3D Extensible 3D (.x3d)
Export Bedrock Model
Export Bedrock Animation
Scene Collection
Collection
battle_mech
Collections
Motion Paths
Visibility
Mcblend: Model Properties
Model origin:
armature
Name:
battle_mech
Visible bounds offset:
0.00
0.50
0.00
Visible bounds width
4.00
Visible bounds height
6.00
Texture width
64
Texture height
64
Texture Generation
Allow texture expanding
Generate Texture
Template resolution
1
Set minecraft UVs
Mcblend: Render Controllers
Playback
Keying
View
Marker
Start
End
Select
Rotate View
Object Context Menu
3.3.0

CHAPTER

SEVEN

CUBES VS POLYMESHES

Minecraft models can be made up of cube-based parts and/or polymesh parts. This section explains the differences betweeen them and how to use them in Mcblend.

7.1 Polymeshes

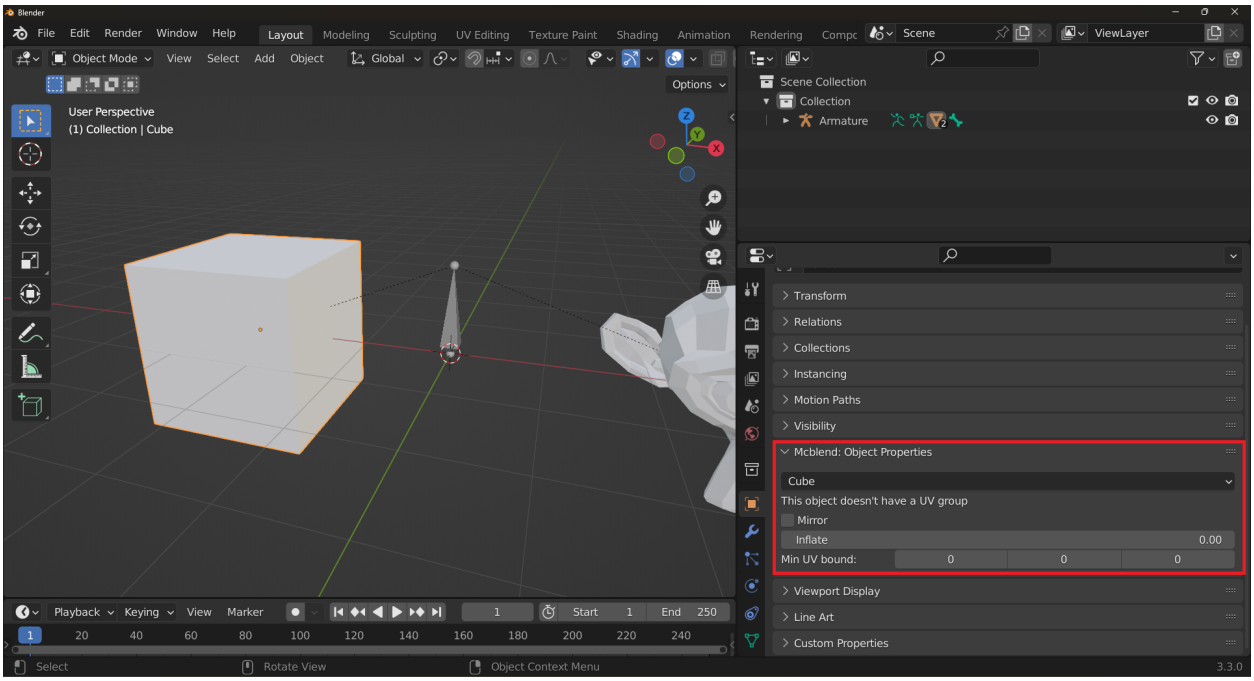
Polymeshes are meshes with any shape, made up of vertices, faces, and normals. They are closer to regular non-Minecraft modeling and are an experimental feature in Minecraft. While Mcblend supports polymeshes, they may not be supported by all modeling tools for Minecraft. This means that if you try to open a model in such a tool it won't be visible. However, you can still open the model in Minecraft to see if it works correctly.

If you try to export a non-cuboid shape as a cube, Mcblend will print a warning and skip the object from exporting.

7.2 Cubes

Cubes are simply cubes and cuboids, and are the default format for Minecraft models. In Mcblend, you can use the built-in *automatic UV mapping* feature for cubes, but it will ignore polymeshes (however, Blender has many other features for UV mapping regular meshes).

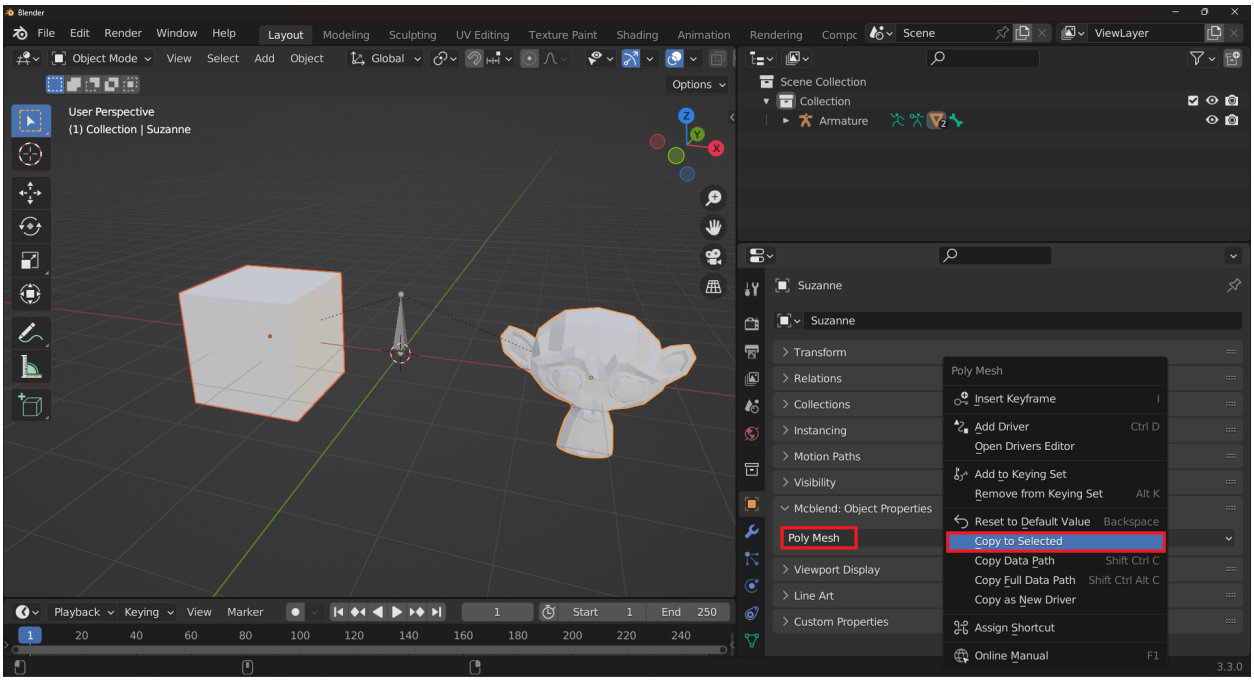
Cubes have additional settings available in their configuration. These settings are: `Mirror`, `Inflate`, and `Min UV bound`. They are explained in detail in different sections of the documentation, where they are more relevant. They all affect the process of automatic UV mapping. The `Mirror` and `Inflate` properties are also directly related to cube-based format, so when you export a model, they will be exported as well.



7.3 Setting the type of an object

In Mcblend, you can choose which type of parts an object should be treated as by selecting it and using the dropdown list in the `Mcblend: Object Properties` tab in the `Object Properties` panel. By default, every object is treated as a cube.

Selecting the same option for multiple objects might be tedious. Luckily there is a solution to that - to quickly set the same property for multiple objects, select the objects and right-click on the property, then select the `Copy to Selected` option. This is useful when you wnat to set everything to `Poly Mesh`.



CHAPTER

EIGHT

THE INFLATE PROPERTY

The inflate property is a property of cubes in Minecraft models. It's similar to scaling, but not exactly. The inflate property changes the size of the cube by adding or removing the same value to each dimension of the cube. For example, if you have a cube of size 3x4x5 and inflate it by 1, the size of the cube will be 4x5x6. Note that this is not the same as scaling the cube - different sides are affected differently in proportion to their size.

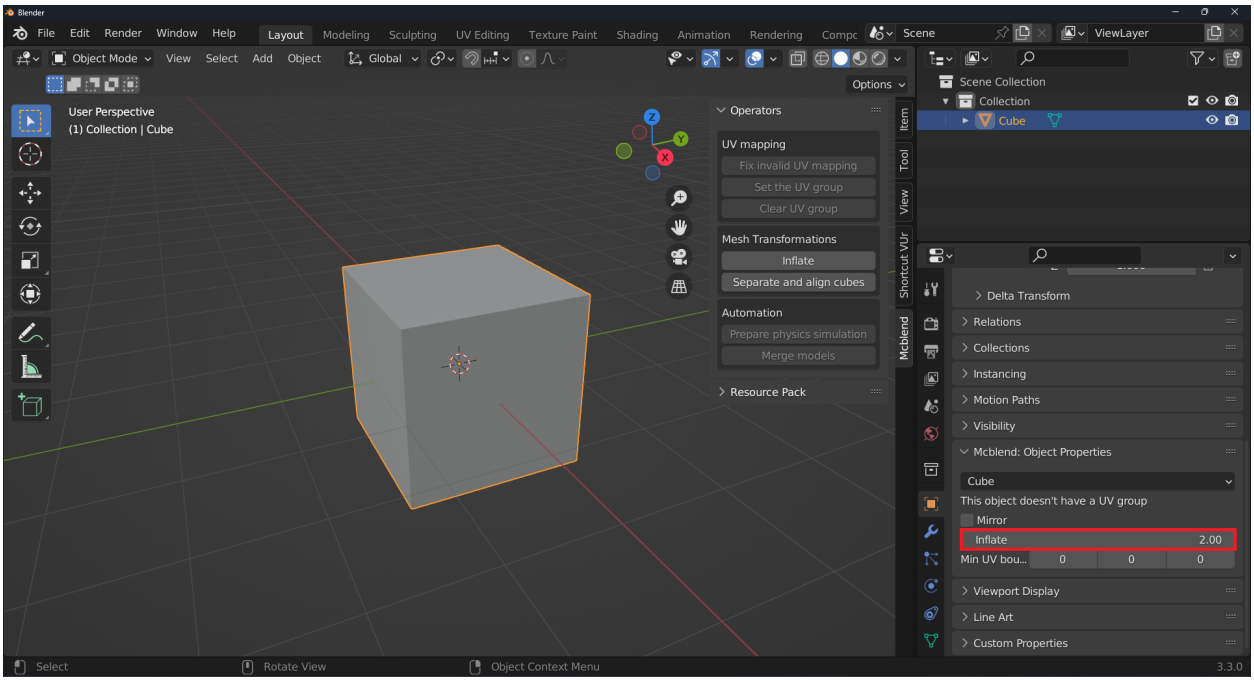
Note: One unit of size in Minecraft models is equal to 1/16 of a block size (1/16 of a meter). If the model is for a resource pack that uses the standard Minecraft texture resolution, then one unit of size is equal to 1 pixel on the texture.

You can change the dimensions of the cube to imitate the inflate property and as long as you don't change the UV mapping, the difference will be unrecognizable.

There are two types of cube UV mapping in Minecraft: the default UV mapping and the per-face UV mapping. The per-face UV mapping defines the UV map for each face of the cube separately. The default UV mapping is based on the dimensions of the cube. This format is probably the reson why the inflate property was added to the game. The default UV mapping is not affected by the inflate value. It only uses the base size of the cube. The inflate property is useful in Minecraft because it allows you to use the default UV mapping for two different cubes that have the same apparent size, but with different inflate values they'll still have a different surface on the texture.

8.1 Changing the Inflate Value Directly

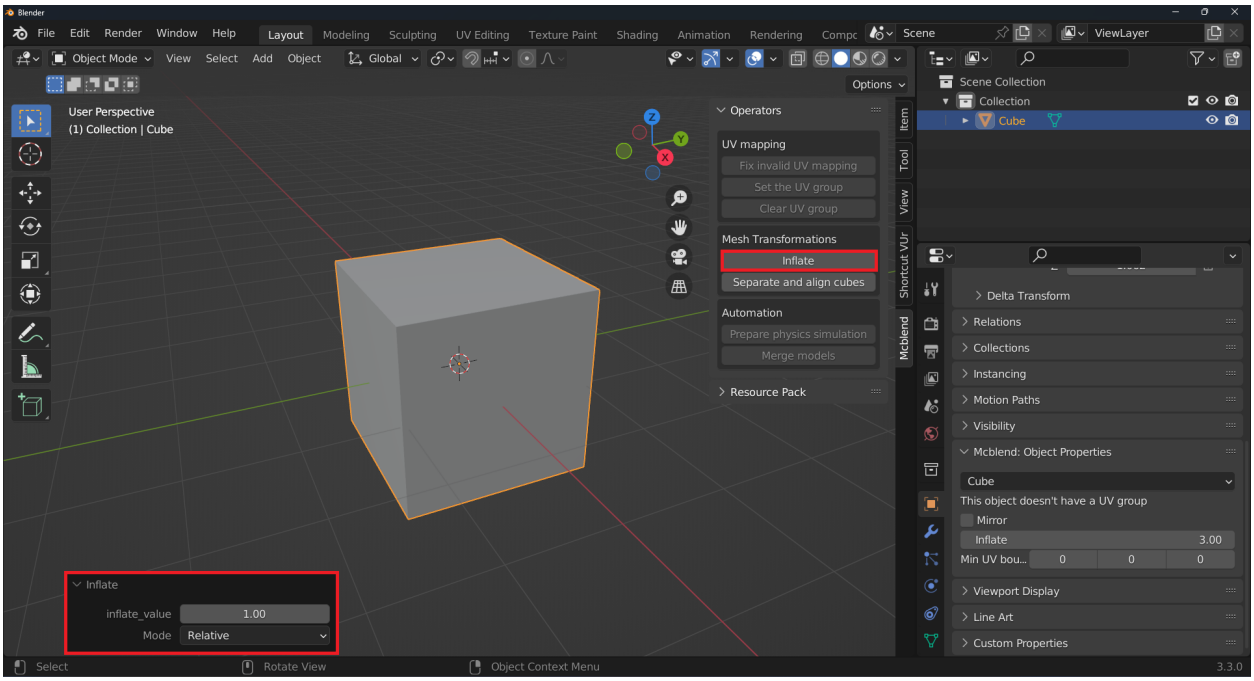
In Mcblend, you can set the inflate property of a cube directly in its properties. This is useful for creating new models or changing the inflate value of a single object without actually changing its size. In order to access the `Inflate` property directly, select the cube and go to the `Mcblend: Cube Properties` tab in `Object Properties`.



8.2 Changing the Inflate Value Using an Operator

The `Inflate` operator allows you to change the inflate value of an object and adjust its size accordingly the 3D viewport at the same time. To use the operator, select the object(s) you want to modify and click the `Inflate` button on the `mcblend` sidebar in the 3D viewport under the `Operators` panel.

The Inflate operator has two modes: `Relative` and `Absolute`. In `Relative` mode, the inflate value is changed relative to the object's current inflate value. In `Absolute` mode, the inflate value is set directly to the specified value. You can use the operator to modify multiple objects at once.



8.3 The Inflate Property in Modeling with Mcblend

In Mcblend, the inflate property is not as important as in other Minecraft modeling tools. This is because Mcblend automatically selects between per-face and default UV mapping based on the size of the cube and its surface on the texture. If the size and surface meet the criteria for default UV mapping, Mcblend will use it. Otherwise, it will use per-face UV mapping.

The inflate property is only relevant when using the *automatic UV mapping* feature in Mcblend. This feature arranges the UV maps of the cubes on the texture in a way that prevents overlap and allows for export using the default UV mapping. More information about the automatic UV mapping feature can be found in other sections of the documentation.

CHAPTER

NINE

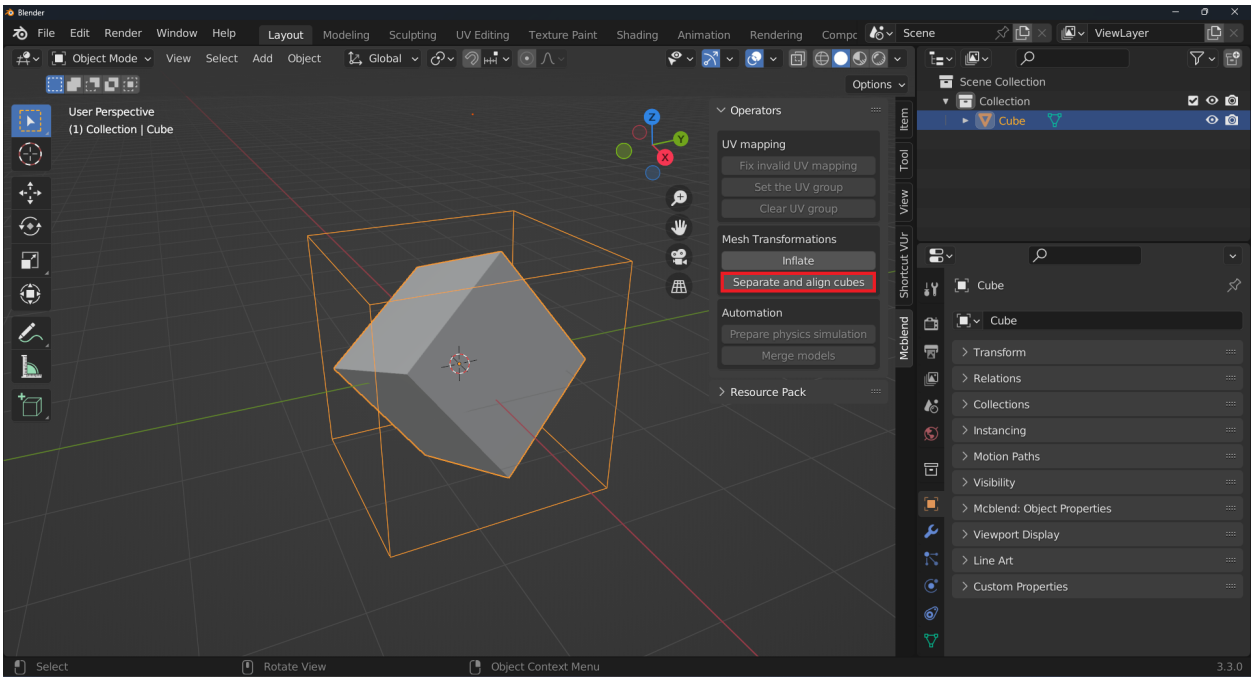
FIXING INVALID CUBES

Invalid cubes are objects in a model that don't meet the requirements for being exported as a cube but are still marked as cube type. In Mcblend, in order to export an object as a cube, it must have a cuboid shape and its cuboid mesh must be aligned to the object's rotation. This means that the object can only hold one cube. If an object has multiple cubes or its mesh is not aligned to its rotation, it will be marked as an invalid cube and will be skipped during export.

9.1 Changing the Cube Type to a Polymesh

One solution to the problem of invalid cubes is to change the object's cube type to a polymesh. Polymeshes don't have the limitations of cubes, so you can have any number of cubes and any shape in a single object. However, it's worth noting that polymeshes are an experimental feature in Minecraft and may not be supported by all modeling tools for the game. Additionally, using polymeshes is generally not recommended as Minecraft doesn't guarantee to support them and they can be removed at any point. You can read more about polymeshes and cubes in the *Cubes vs polymeshes* section of the documentation.

9.2 Using the Separate and Align Cubes Operator



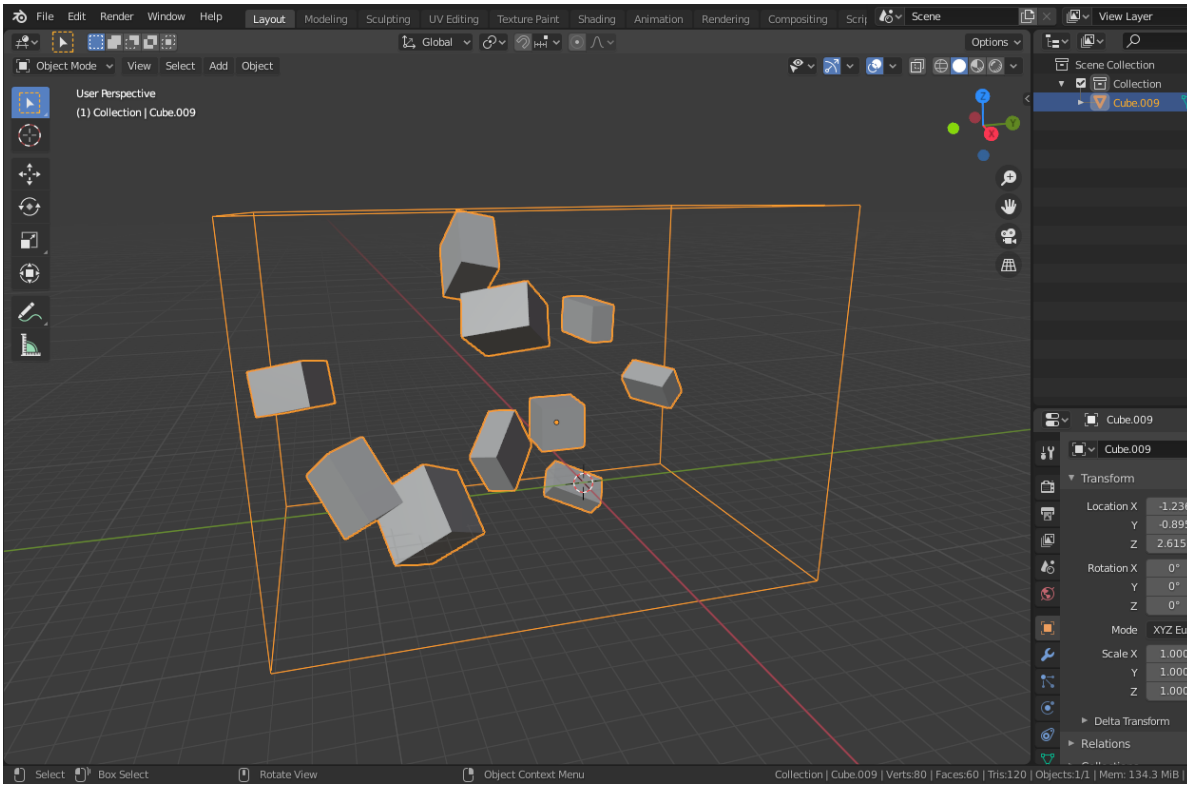
Another solution to the problem of invalid cubes is to use the `Separate and Align Cubes` operator. This operator is designed to fix invalid cubes and also enables a workflow where you can create a model in a single mesh and then fix it using the operator.

To run the operator, press the `Separate and Align Cubes` button in the `Mcblend` sidebar in the `Operators` panel. The operator separates all of the disconnected parts of the mesh and tries to align the object's rotation to match the rotation of the cubes in the mesh. The operator works on all selected objects.

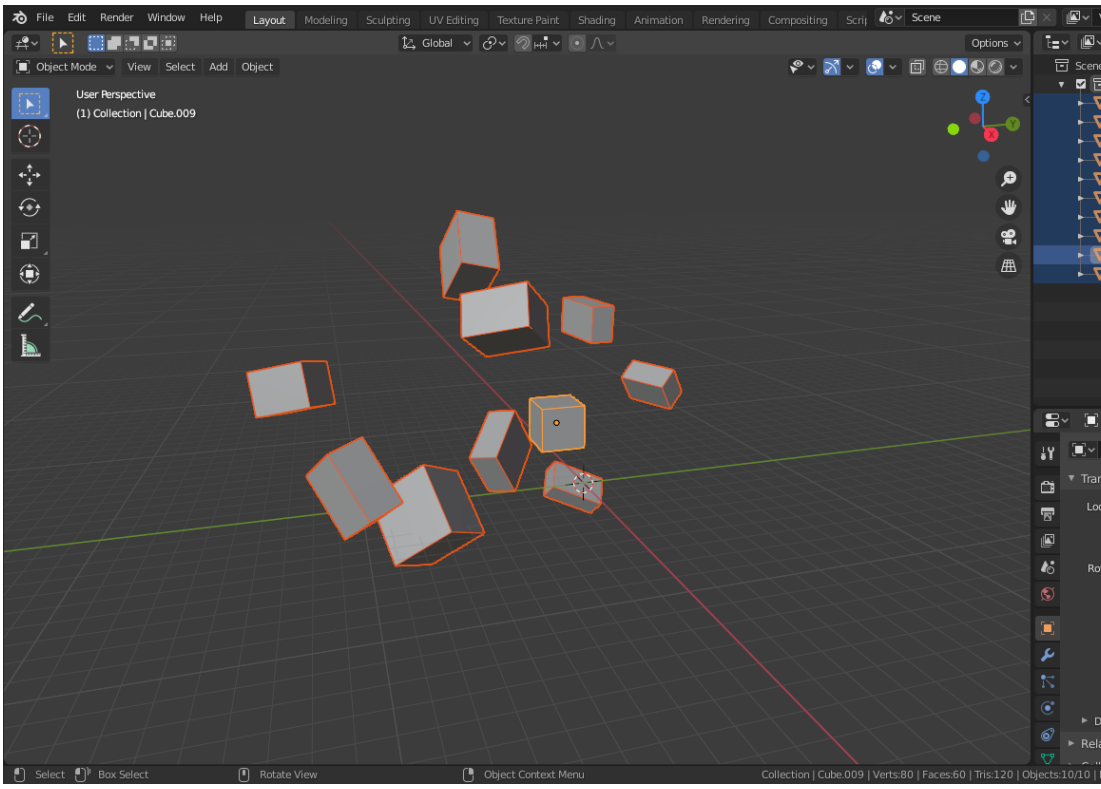
It's worth noting that the `Separate and Align Cubes` operator is different from the `Separate` operator that is part of Blender by default (found under `Mesh > Separate`). The `Separate and Align Cubes` operator also aligns the separated meshes, whereas the default `Separate` operator does not.

If the separated object isn't a cube, this operator won't be able to fix it. In such cases, you may need to use a different solution or consider changing the object's cube type to a polymesh.

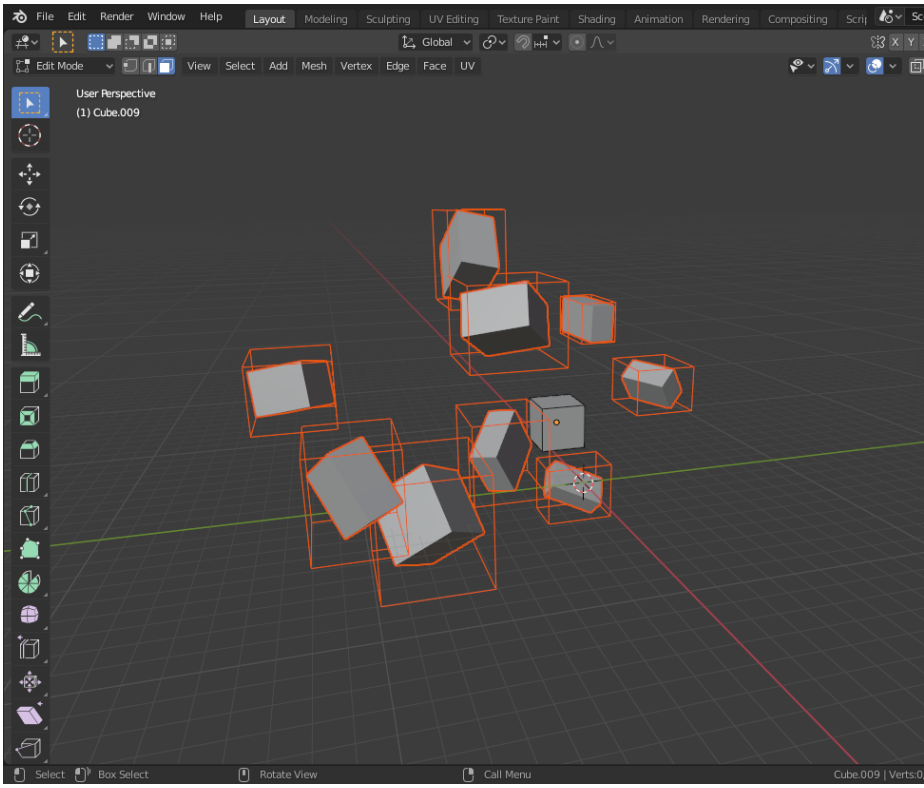
The images below illustrate the difference between the default `Separate` operator and the `Separate and Align Cubes` operator. The cuboids around the objects represent their bounding boxes.



Before separating cubes



Objects separated with Mcblend



Objects separated using default Blender operator

CHAPTER

TEN

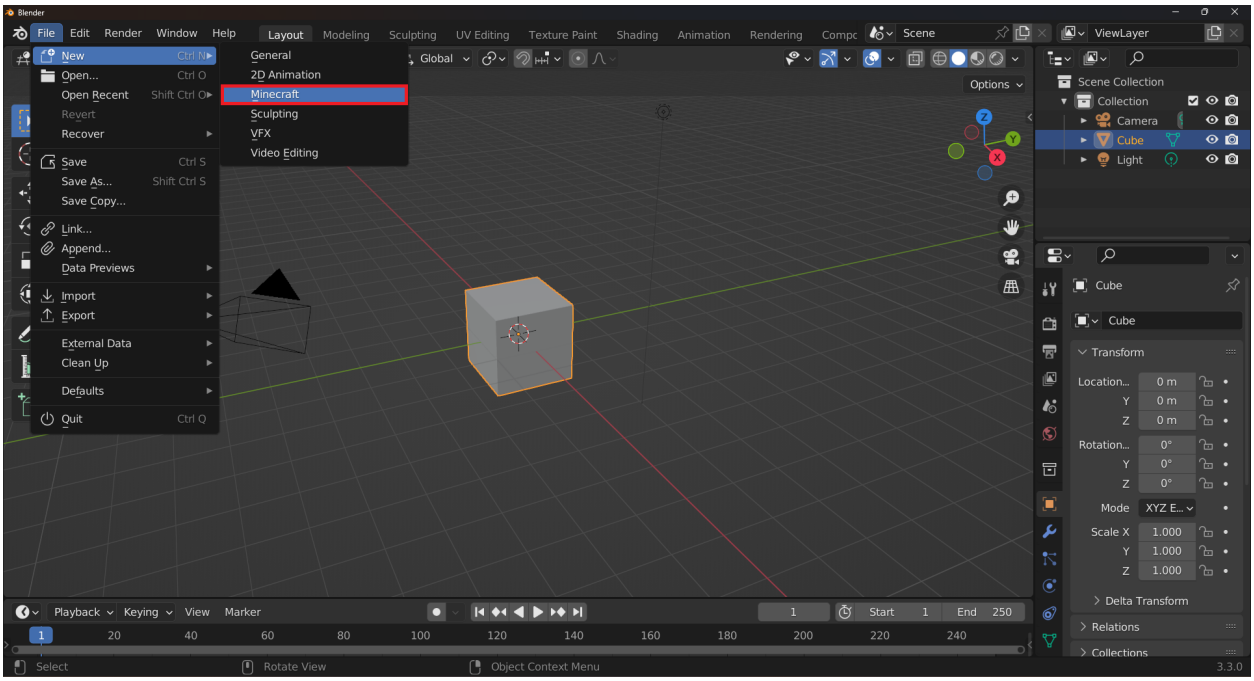
ATTACHABLE ITEM MODELS

This tutorial will show you how to adjust the position of an attachable item so that it fits into the hand of a player. However, it does not cover the process of creating or adding the model into the game. If you need a more comprehensive guide, you can refer to the following links:

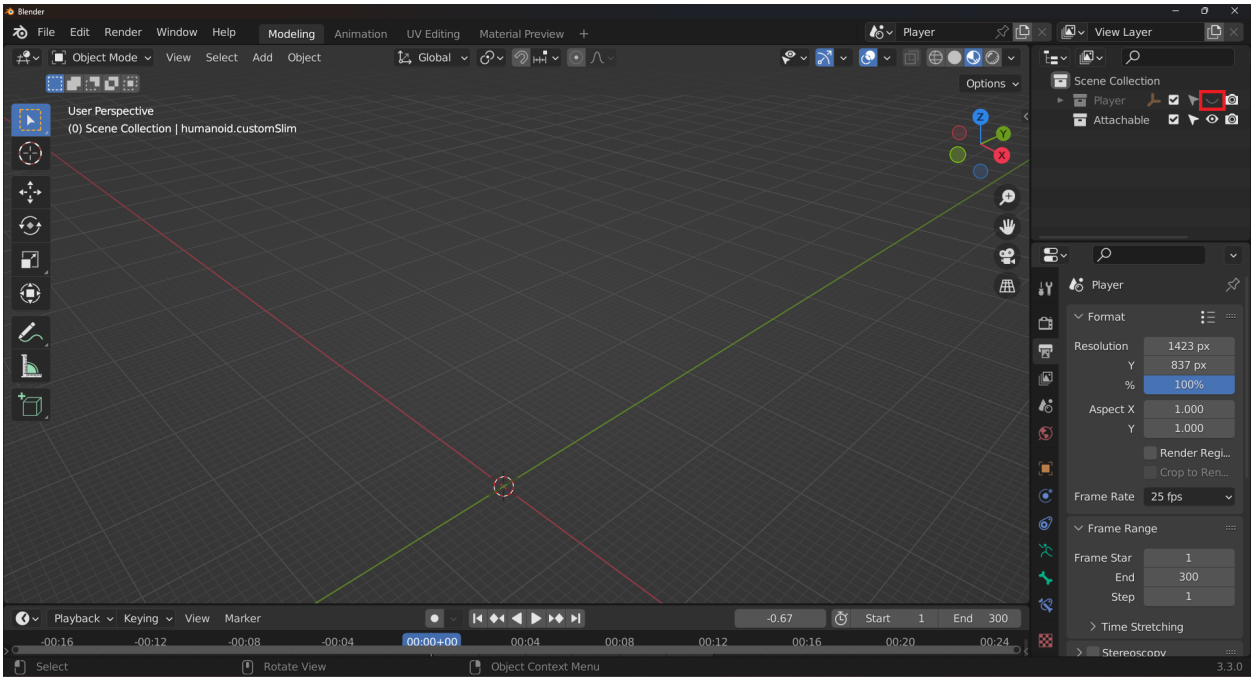
- https://mcblend.readthedocs.io/en/v10.0.0/advanced_tutorials/attachables_and_1st_person_animations/ is a tutorial from an older version of Mcblend's documentation that explains how to create, animate, and add an attachable item to the game. However, it utilizes a removed feature, the `World origin` property of animations, which has been replaced by the `Model origin` property of models.
- https://wiki.bedrock.dev/items/attachables.html is a tutorial from the Bedrock Wiki (created by the Minecraft Add-on making community) that explains how to create attachable items and add them to the game, but it does not use Mcblend.

10.1 Positioning the model

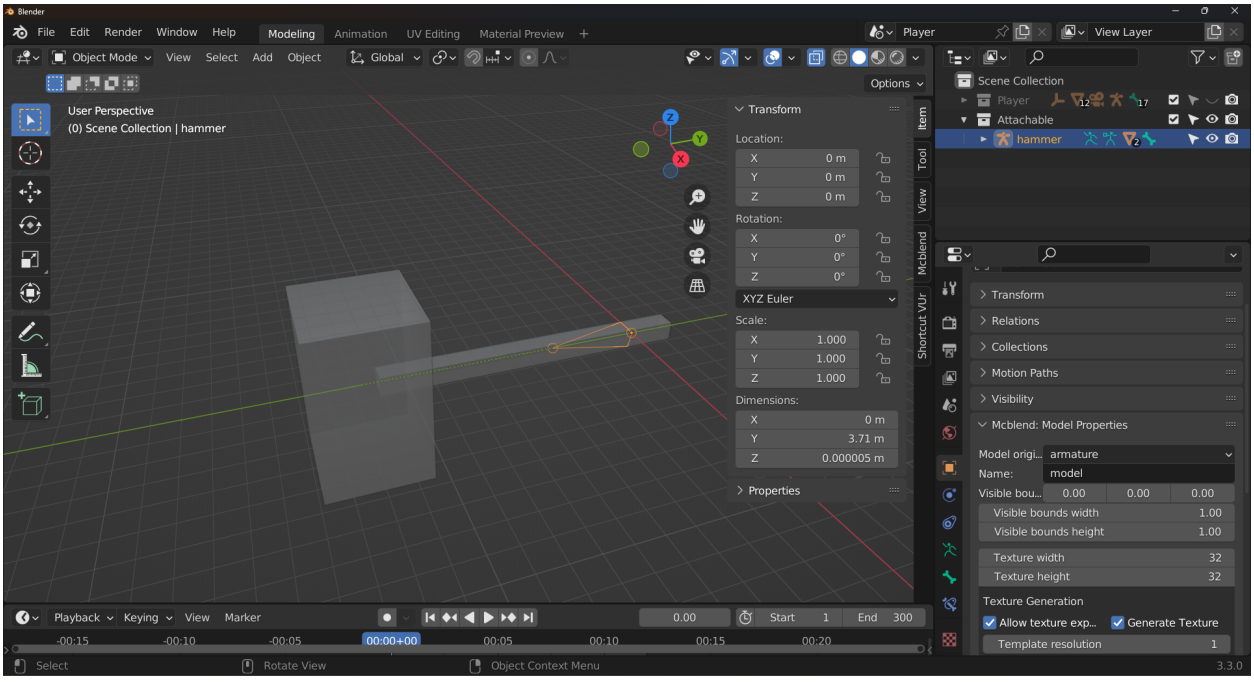
This tutorial uses the Mcblend template project, which you can learn more about in the *dedicated section of the documentation*. The Mcblend template project is specially designed to work with Minecraft models, and it also includes the player model, which can be useful when animating the attachable item later on. You don't have to use the template, but it's recommended because you won't have to set up the project from scratch.



To begin working on the attachable model, we can make the player model invisible for the time being.

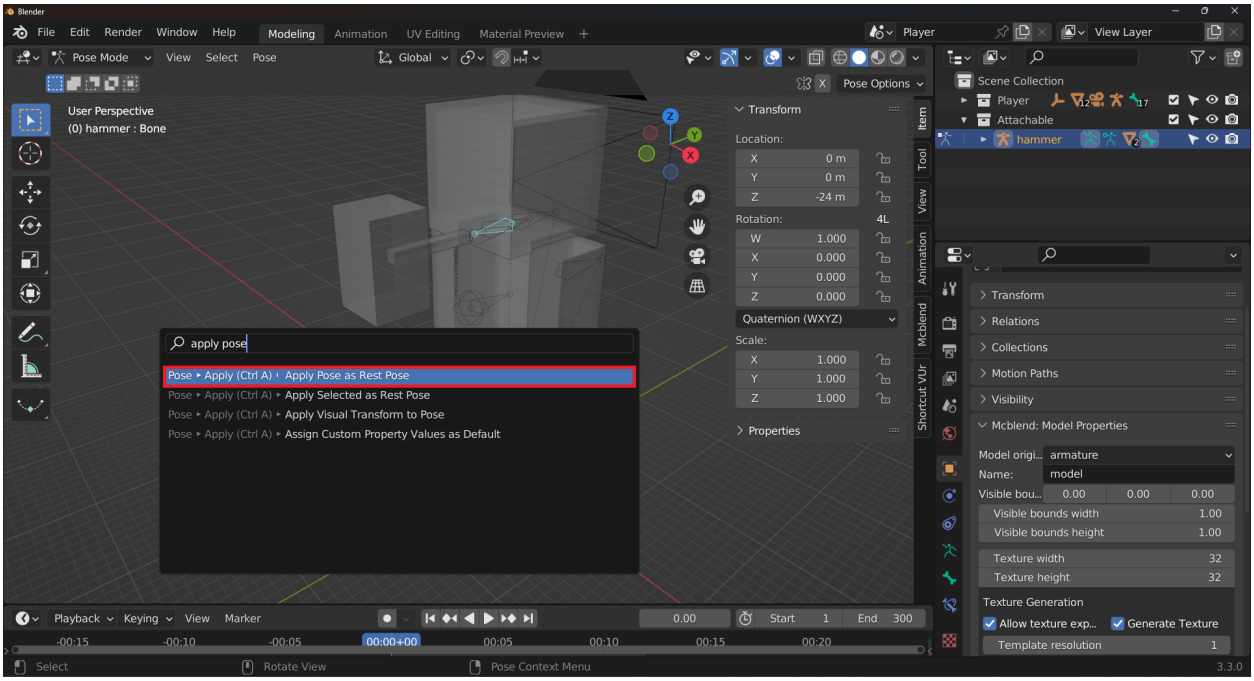


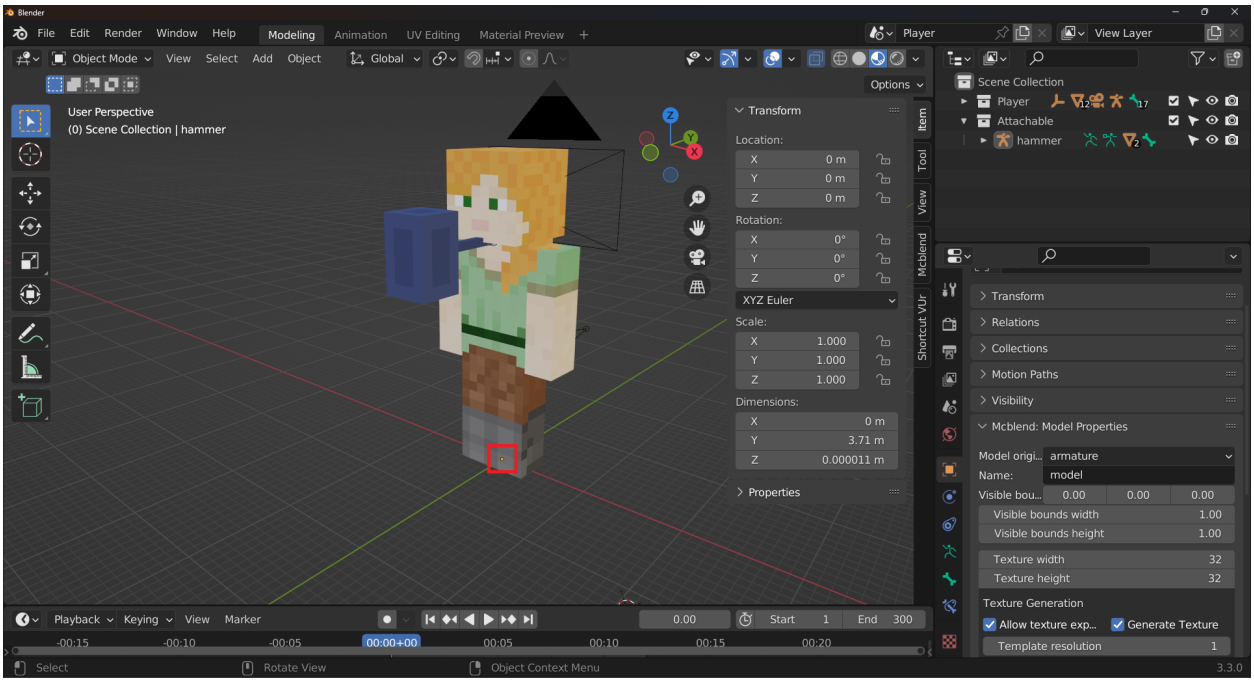
The model of the attachable should be oriented in a way that corresponds to the player's hand. In this example, the tutorial model is a hammer. The player model is facing forward, and the player is holding their hands down. In this posture, it is natural to hold the hammer with the head of the hammer facing forward.



When the model is complete and you are satisfied with its appearance, you can proceed to the next steps to use it as an attachable item in Minecraft. Note that attachable models have a 1.5m offset that must be taken into consideration when creating them. To ensure proper positioning, go to `Pose Mode` and move all of the bones of the model 1.5m up. If you're using the Mcblend template project, the world scale is set to 16x (to match Minecraft modeling size units), so you should move the model 24m up. This will ensure that the model is correctly positioned slightly below the player's

head. Once the model is in the correct position, apply the pose as the rest pose by selecting all bones in `Pose Mode`, pressing `F3`, and finding the `Apply Pose as Rest Pose` operation. This operation sets the positions and rotations of the bones as the default position of the model. It is important to ensure that the pivot point of the armature remains at the origin of the world, which can be seen as a little orange dot in the 3D view when in `Object Mode` and the armature is selected.

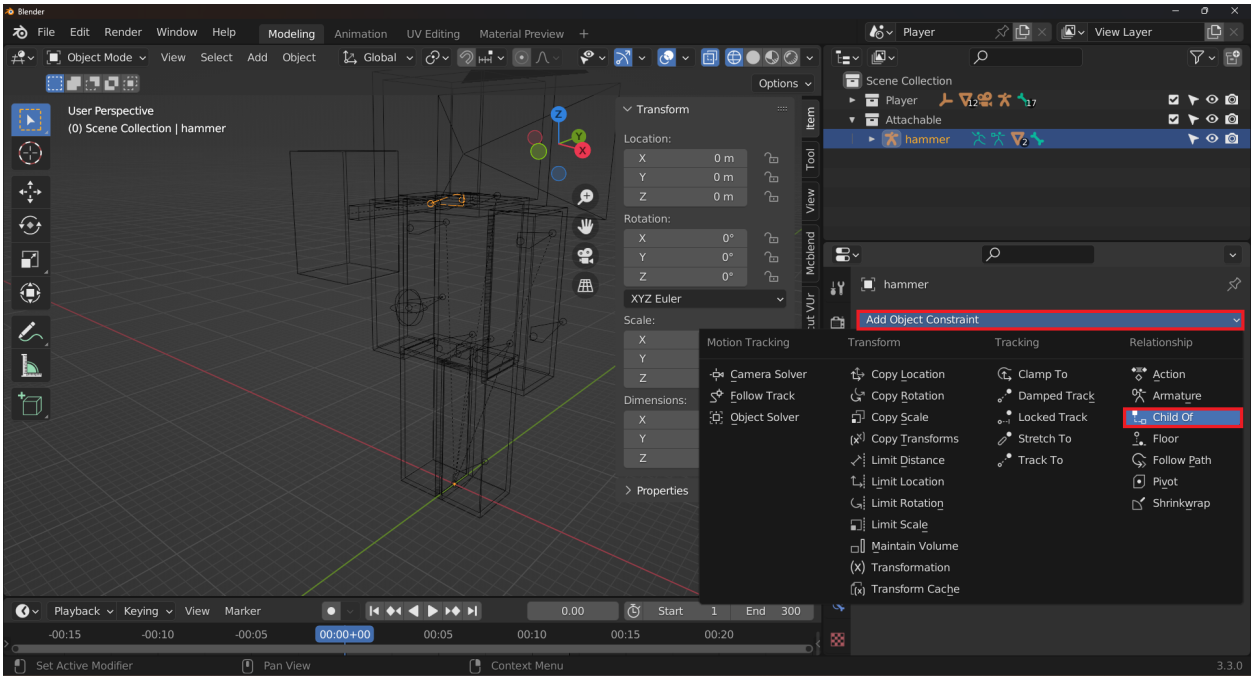




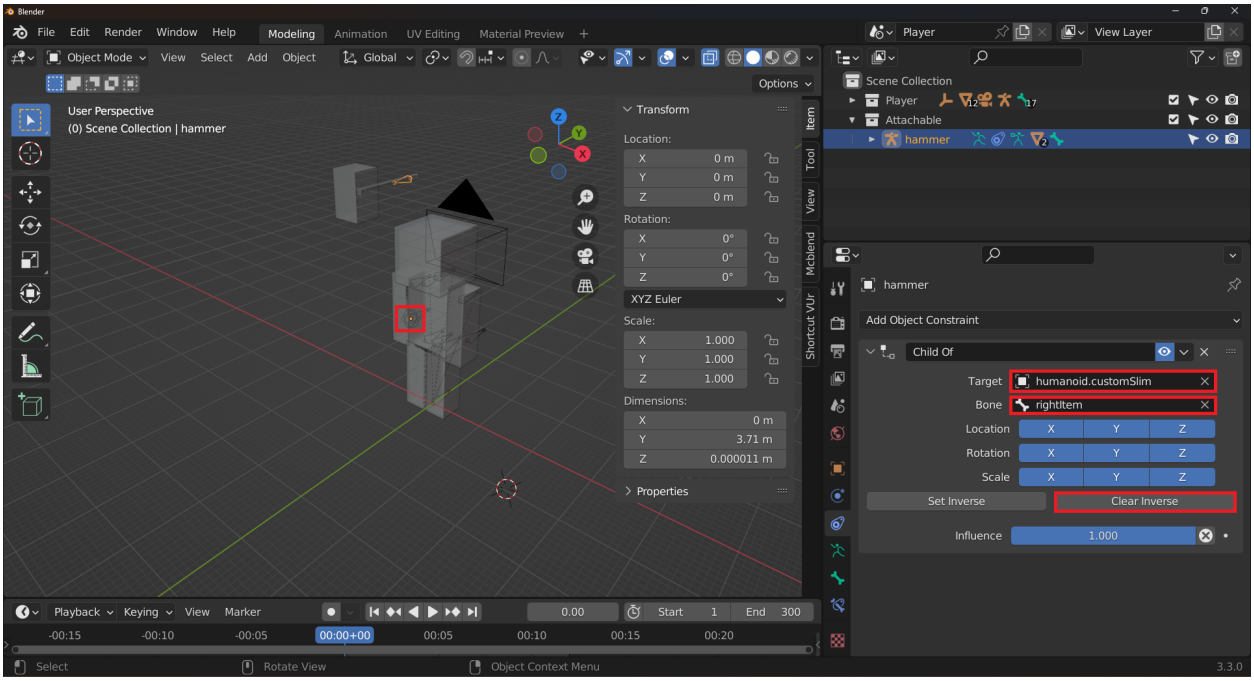
To make adjusting the position of the attachable model easier, you can add a `Child Of` constraint to its armature. This will parent the armature of the attachable model to the armature of the player model, allowing you to see how the model will appear in-game while you make adjustments.

To add the constraint:

1. Go to `Object Mode` and select the armature of the attachable model.
2. In the `Constraints` tab, add a new constraint by clicking on the `Add Object Constraint` button and selecting `Child Of`.
3. In the `Child Of` constraint settings, set the `Target` to be the armature of the player model and the `Bone` to be `rightItem`.

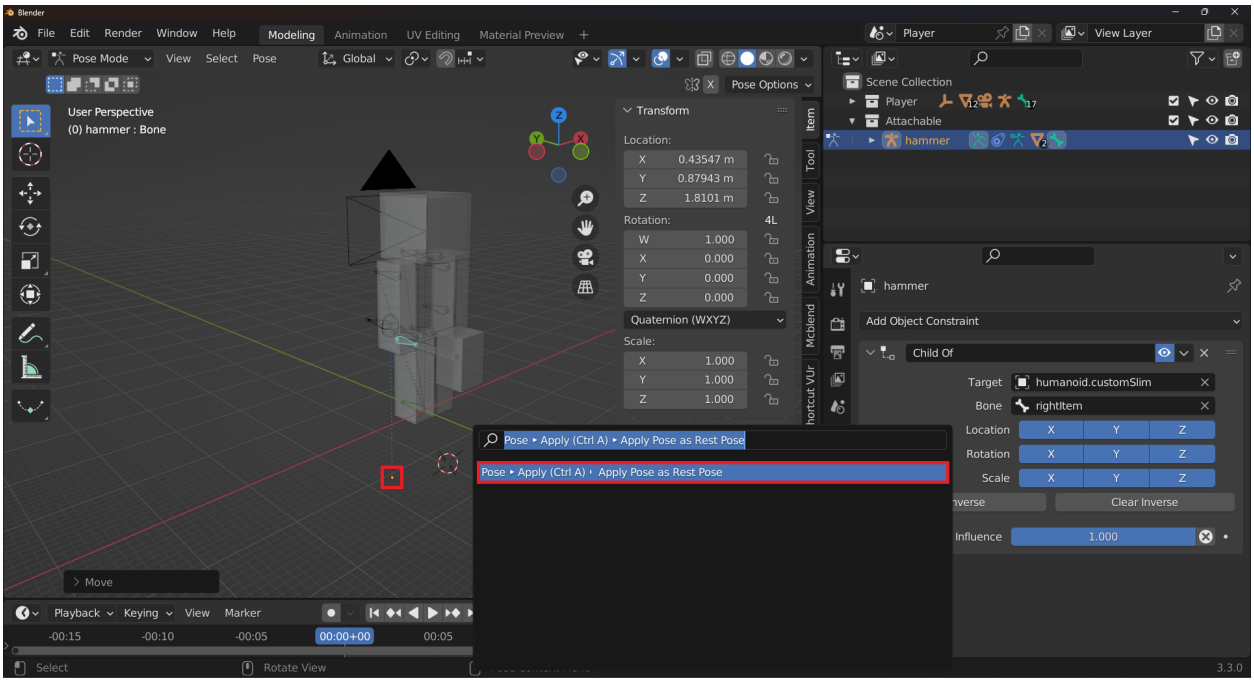


After adding the constraint, press the `Clear Inverse` button. This will place the attachable model in a position above the player's hand, with a 1.5m offset.

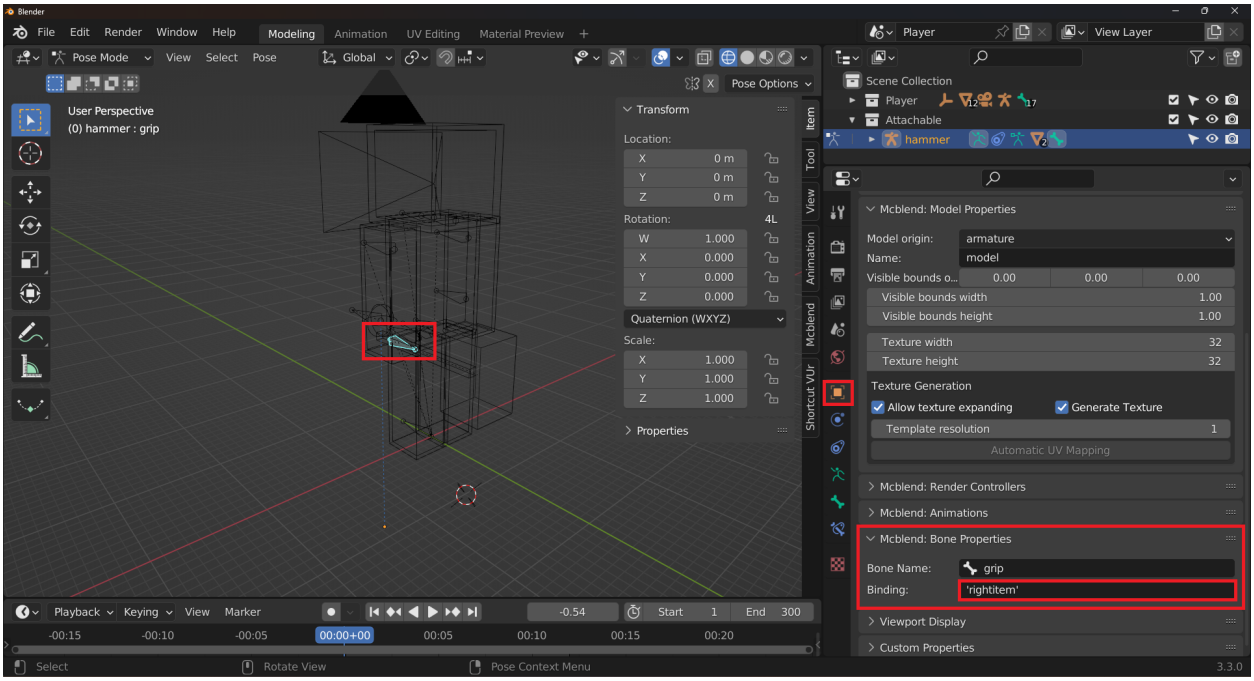


To position the attachable model correctly, you should move the armature of the model 1.5m down in `Object Mode`.

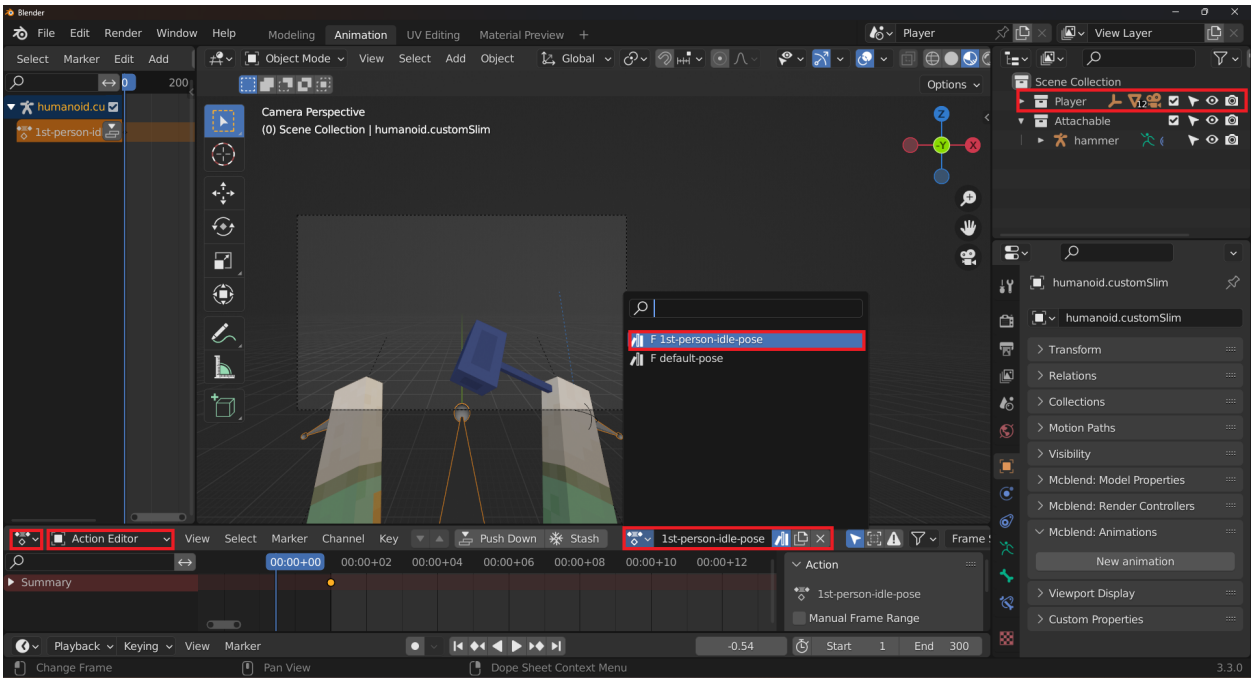
This will place the item roughly in the right position. You can then fine-tune the position further by adjusting the bones in `Pose Mode`. After you've finished adjusting the position, apply the pose again (as described in previous steps of this tutorial). The final position of the origin of the armature should be 1.5m below the `rightItem` bone of the player model and the item should be positioned in the player's hand in a believable and natural way.



To make your attachable item functional in Minecraft, you must add a special property to the root bone of the model. In this case, the model has only one bone. This property is called "binding" and it should contain the name of the bone that the attachable should be attached to. To add this property, you will use the Molang expression. The proper format of a Molang expression for a bone named `rightItem` is `'rightitem'` (with single quotes and all letters in lowercase). To set this property in Mcblend, go to `Pose Mode`, select the root bone of the armature of the attachable, select the `Object Properties`, and in the `Mcblend: Bone Properties` set the `Binding` property to `'rightitem'`.



The player model in the template project has an additional animation that allows you to preview how the attachable item will be held in the game in 1st person mode. This can be useful as a reference when creating animations for the attachable item.



Once you have the attachable model in the desired position, you can animate both the player model and the attachable simultaneously. The process of creating animations for attachables is no different than creating animations for any other model. The key is to pay attention to the armature that you have selected while exporting the model or animation. If the armature of the player is selected, Mcblend will export a model/animation for the player. If the armature of the attachable is selected, Mcblend will export a model/animation for the attachable.

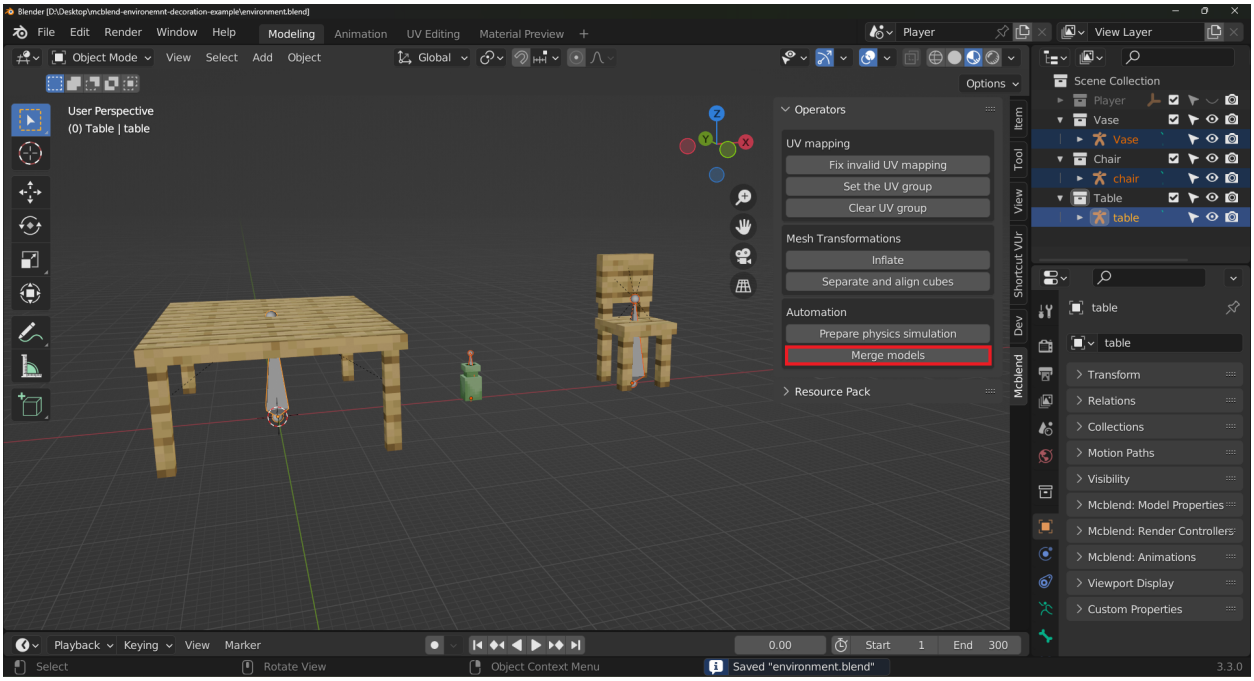
When exporting the attachable, ensure that the `Model Origin` in the `Object Properties > Mcblend: Model Properties` is set to `armature`. This is the default setting, so unless you have changed it, you don't need to worry about it. This setting ensures that all animations of the armature are relative to its origin and not the world, so the movement of the hand holding the item is not exported as part of the animation of the attachable.

CHAPTER

ELEVEN

MERGING MODELS

The Merge Models feature in Mcblend allows you to combine multiple Minecraft models and their textures into a single model with a new texture. This can be useful for creating more complex models by combining simpler ones.



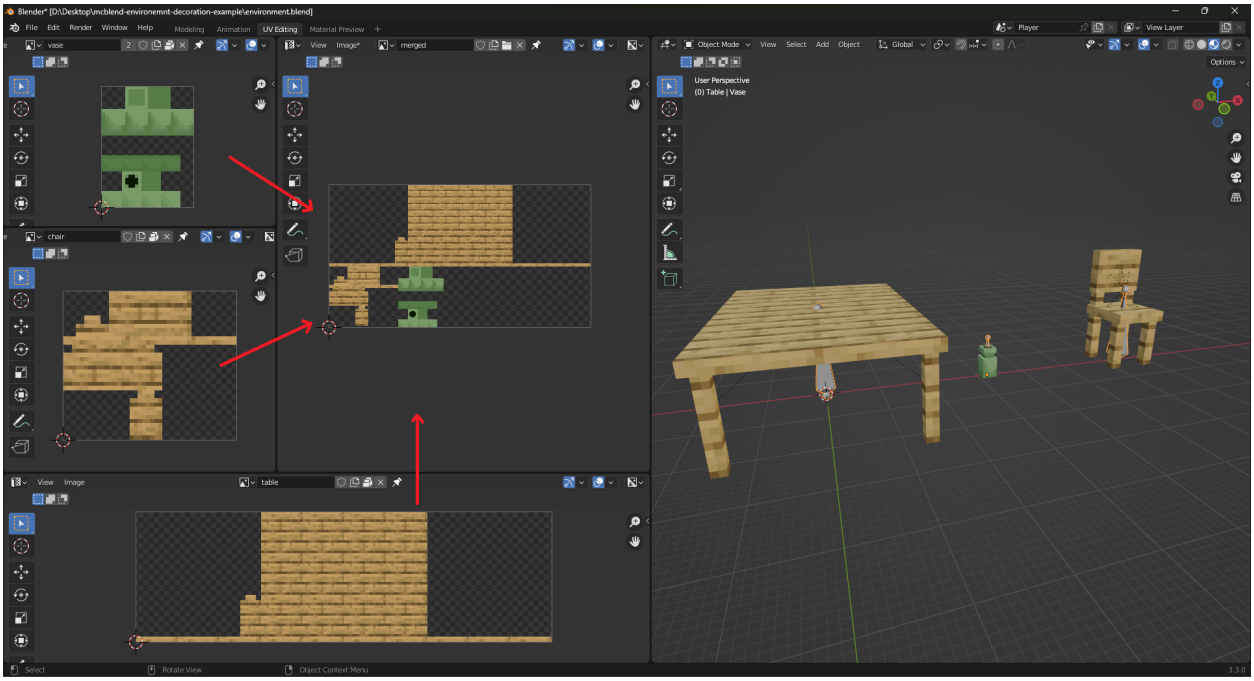
11.1 How to Use Merge Models

Using the Merge Models feature is straightforward:

1. Import the models that you want to merge into your project.
2. Select the armatures that contain the models you want to merge.
3. Press the Merge Models button.
4. Press `OK` in the dialog that appears. In the dialog, you'll see the list of the models that will be merged and the names of the textures that will be used for each model. The `Merge model` operator always uses the texture from the first render controller of each model.

After the merging process is complete, you will have a new model with a new texture that contains the textures of all the merged models. All of the bones in the merged models will have been renamed to avoid naming collisions, and the new model will have a new render controller that maps all of the bones to the new texture.

Image below shows the result of merging 3 models:



11.2 Limitations

It's important to note that the Merge Models feature has some limitations:

- The feature won't work properly if some of the models use complex render controllers or multiple textures at once. The merged texture is based on the first texture of each model being merged, so if the models use multiple textures, some of the textures will not be included.
- The Merge Models feature doesn't preserve the names of the bones.
- If you merge animated models, the animations may won't work properly.
- In order for the operator to work, every model must have a render controller and a texture assigned to it.

CHAPTER

TWELVE

AUTOMATIC UV MAPPING

In Minecraft, most models use a single, low resolution texture. This is because the models usually consist of cuboids, which allows for simple textures as the UV coordinates can be aligned to the pixel grid. Mcblend has a feature which helps with UV mapping and texturing based on this workflow - the automatic UV mapping. It arranges the UV coordinates of the cube faces and creates a template texture for you to draw on. Automatic UV mapping is highly customizable, however, you don't have to use it - you can move the UV coordinates manually or generate and edit them after that.

Using automatic UV mapping, you can adjust a number of properties for each cube, including the *minimum space* it should take up on the texture, *the grouping of cubes* (some can be mapped to the same space on the texture, resulting in the same texture for those cubes), the mirroring of the texture for each cube (using Minecraft's mirror property), and the way each face of the cube is textured on the *template texture* (using UV groups to allow for advanced configurations). Some of the more advanced features are explained in different pages. This page explains the basics, which should be sufficient for most users.

12.1 UV mapping polymeshes

In Mcblend, there is no special support for texturing *polymeshes*, so if you are using polymeshes you should use regular Blender features for texturing. If your model is partially cube-based and partially polymesh-based, you can use automatic UV mapping for the cube-based parts and regular Blender features for the polymesh-based parts.

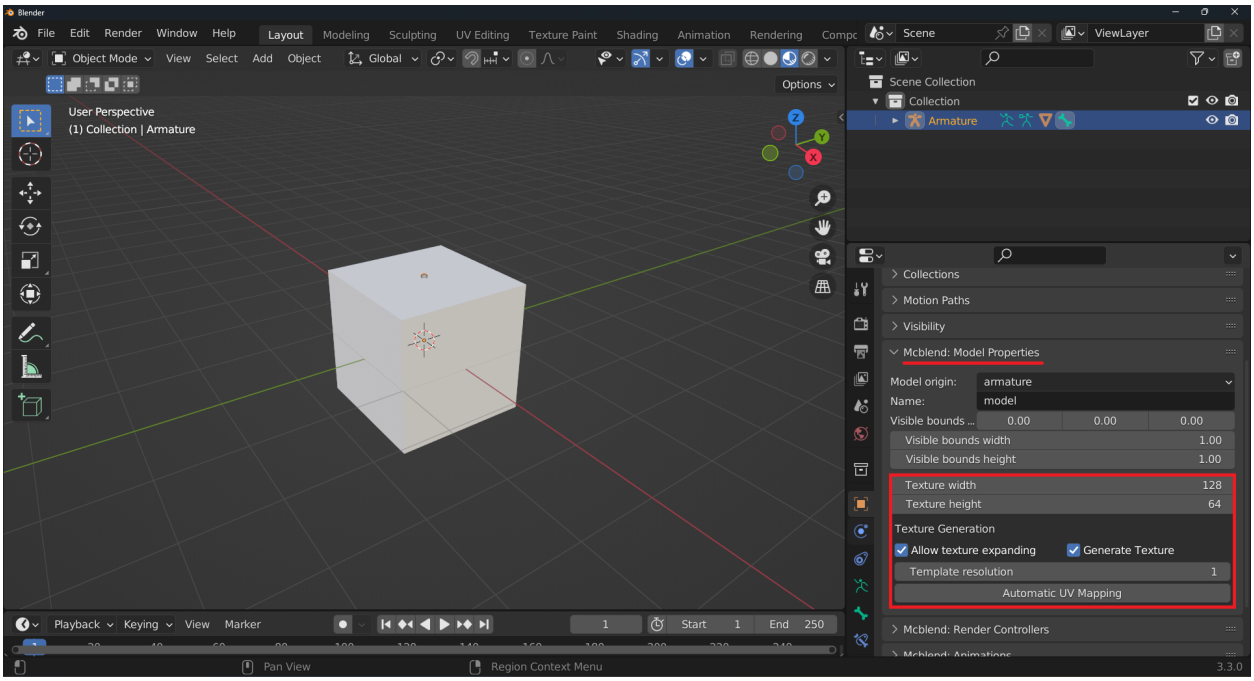
12.2 How to use automatic UV mapping (and texture generation)

Assuming you have a cube-based model that is already connected to an armature, you can find the Automatic UV mapping feature in the `Object Properties` panel under the `Mcblend: Model Properties` in the Minecraft properties of the model. These settings are visible only when you have an armature selected and are in `Object Mode` in Blender. Before running the `Automatic UV mapping` operator, you can adjust its settings as follows:

- The `Generate Texture` checkbox (enabled by default) controls whether the operator should also generate a texture. The generated texture is saved as an internal Blender resource named `template` and is not saved to the disk.
- The `Template resolution` property allows you to set the resolution of the generated texture. If the `Generate Texture` checkbox is disabled, this property disappears. The `Template resolution` property is set to 1 by default, meaning that 1 pixel in the generated texture corresponds to 1 unit size of the model (which is 1/16 of a block in Minecraft). You can increase the value to increase the resolution; any integer bigger than 0 is valid. For example, setting it to 2 will mean that 1 pixel in the generated texture corresponds to 1/32 of a block.
- The `Texture width` and `Texture height` properties allow you to specify the desired size of the texture in pixels. When automatic UV mapping is used, Mcblend will try to lay out the cubes to fit them into the texture. If the texture is too small, an error may be shown or the texture may be expanded based on the `Allow texture`

`expanding` setting (enabled by default). The `Texture width` and `Texture height` are also the properties that are directly exported to the exported Minecraft model file when you do the export.

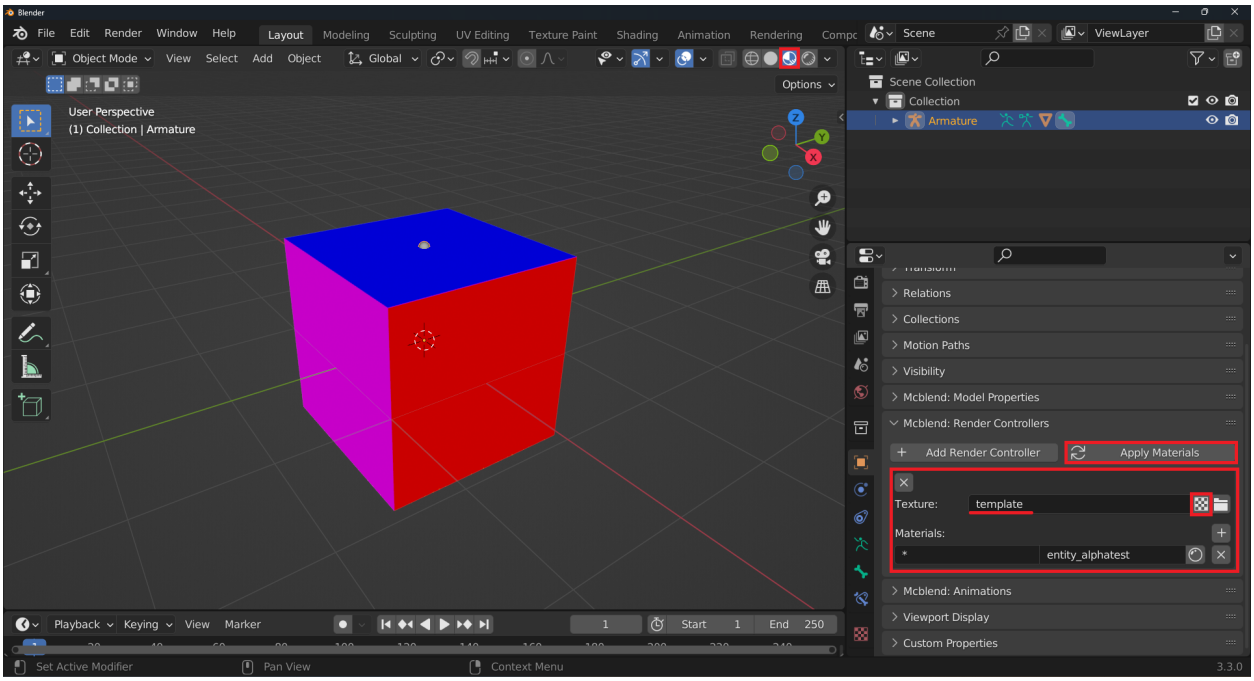
- The `Allow texture expanding` setting, when enabled, will cause Mcblend to lay out the cubes left to right on the texture, starting with the biggest cubes and then going to smaller ones. If the texture of a cube is too wide to fit into the current row, it will be moved to the next row. If the texture doesn't fit into an entire row, the width of the texture will be increased. If there is no space to make a new row, the height of the texture will be increased.



Other features of automatic UV mapping, such as grouping cubes on the same texture space or adjusting their properties, are explained later in the documentation.

Once you have your desired configuration, press the `Automatic UV mapping` button. Just running the operator won't apply the generated texture to the model. To do that, you must use the render controllers feature.

You can access the render controllers configuration from the `Mcblend: Render Controllers` panel in the `Object Properties` panel. If the armature doesn't have any render controllers yet, you can add one by pressing the `Add Render Controller` button. Press the button with the texture icon next to the `Texture` field and select the template texture. Then, press the `Apply Materials` button to generate and apply a material to the model. To see the texture, you must switch to `Material Preview` mode in the 3D viewport.

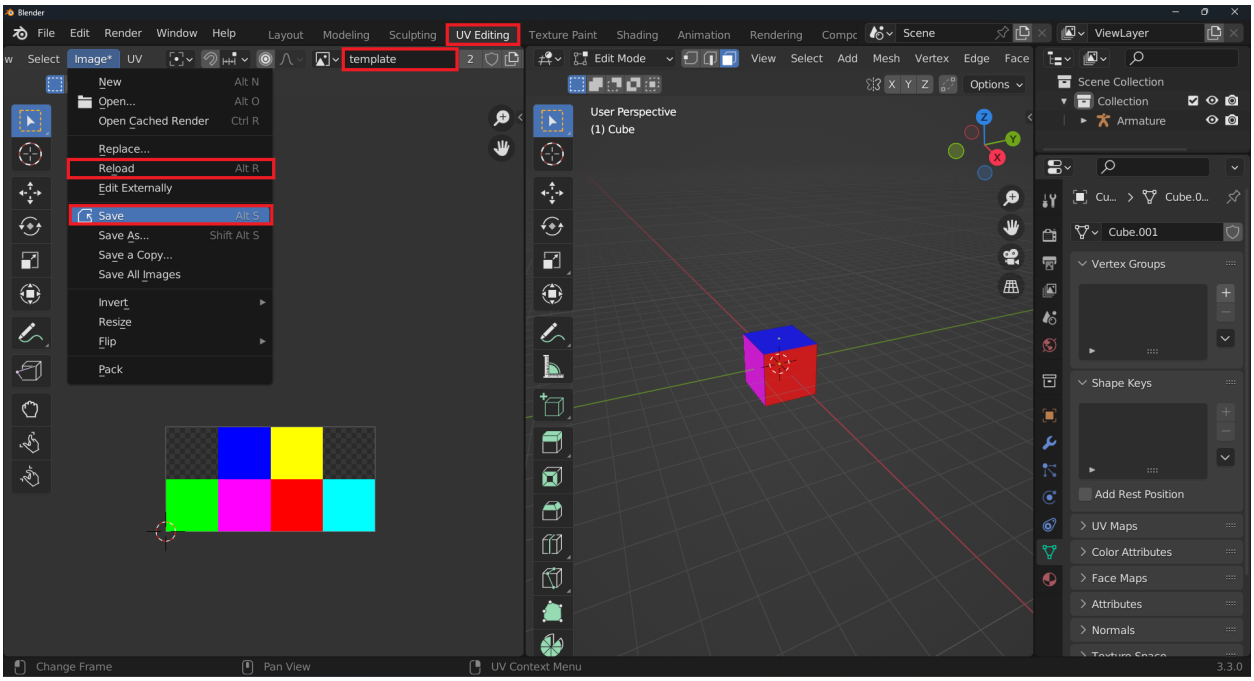


12.3 Modifying the texture and UV manually

If you want to modify the texture or UV map manually, you can do so by following these steps:

1. First, save the texture to your computer by using the `UV Editing` workspace.
2. In the `UV editor` screen, select the `template` texture and use the `Image > Save` menu to save it.
3. Edit the texture with an external program.
4. When you finish editing, use the `Image > Reload` menu to reload the modified image into Blender.
5. You can also move the UV map like any other UV map in Blender.

This process is not specific to Mcblend, it's just how you can edit textures in Blender.


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering
Select Image* UV
template
New Alt N
Open... Alt O
Open Cached Render Ctrl R
Replace...
Reload Alt R
Edit Externally
Save Alt S
Save As... Shift Alt S
Save a Copy...
Save All Images
Invert
Resize
Flip
Pack
Edit Mode View Select Add Mesh Vertex Edge Face
Options
User Perspective
(1) Cube
Scene Collection
Collection
Armature
Cube.001
Vertex Groups
Shape Keys
Add Rest Position
UV Maps
Color Attributes
Face Maps
Attributes
Normals
Change Frame
Pan View
UV Context Menu
3.3.0

CHAPTER

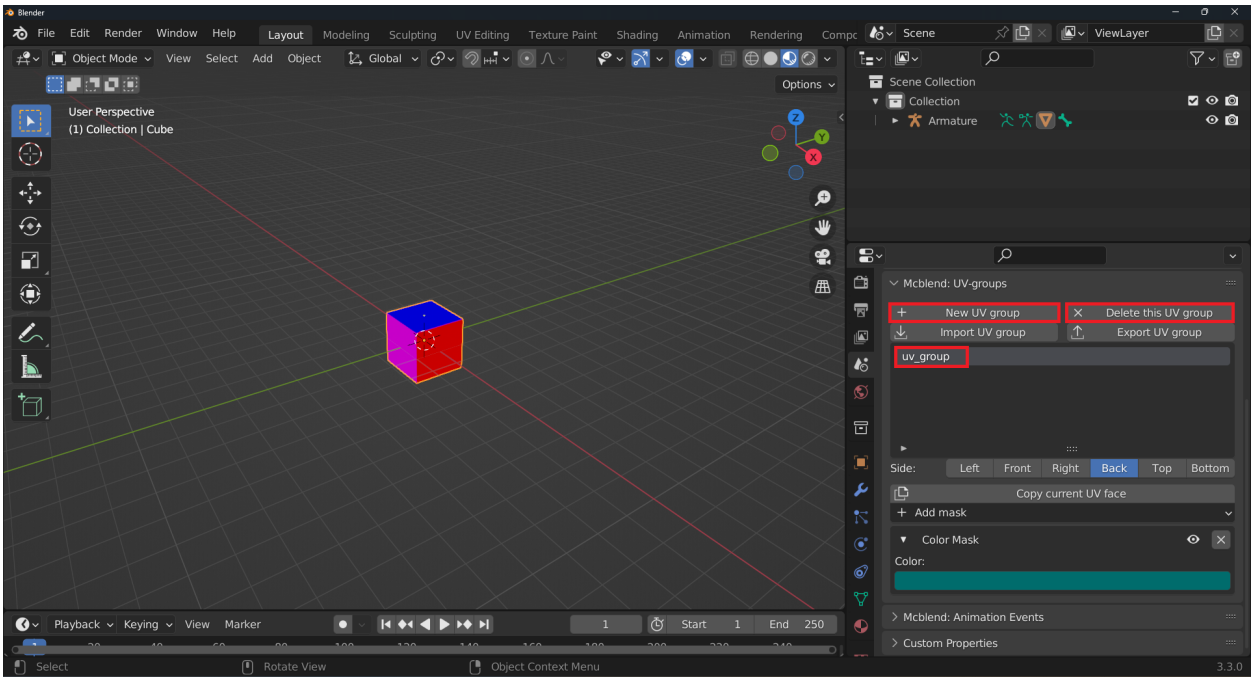
THIRTEEN

UV GROUPS

UV groups in Mcblend are used to group cubes together and apply the same texture to them. They can be managed in the `Scene Properties` panel under `Mcblend: UV groups`. UV groups affect how *automatic UV mapping* works. If two cubes have the same size and belong to the same UV group, they will be mapped to the same texture space.

13.1 Creating UV groups

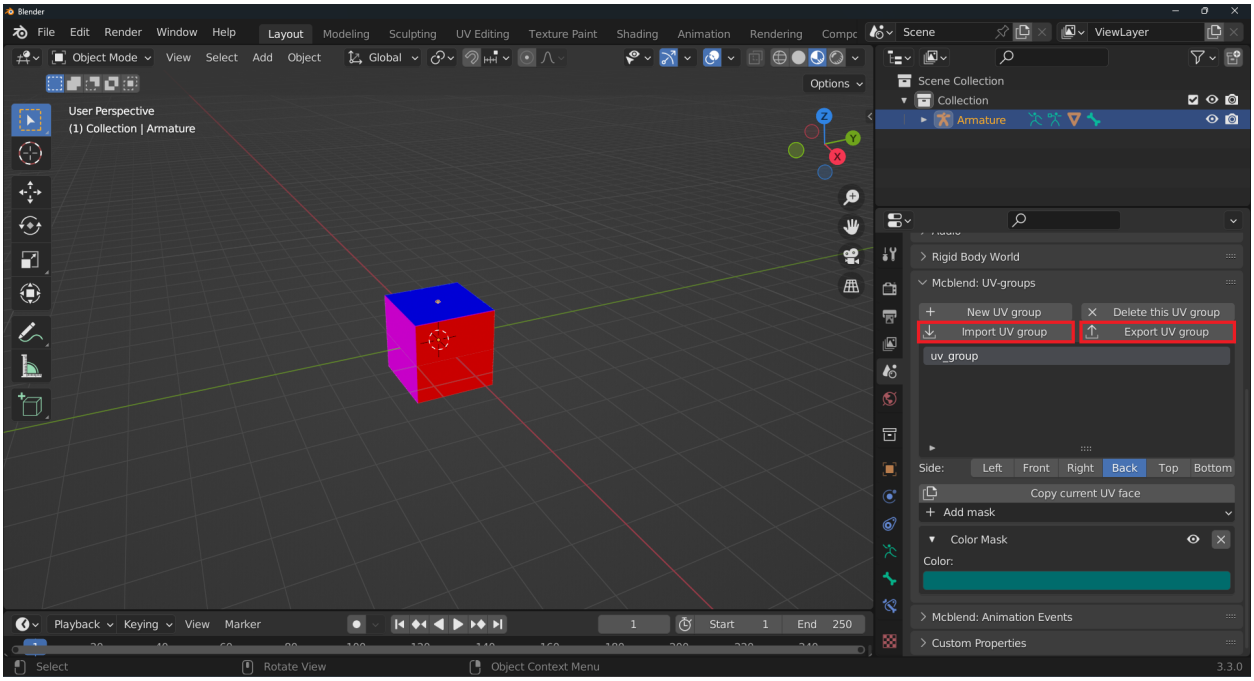
To create a new UV group, press the `New UV group` button in the `Mcblend: UV groups` panel. By default, this will generate a texture similar to the default texture, but slightly darker, allowing you to easily identify which cubes are assigned to a particular UV group when generating textures. You can rename UV groups by double clicking on them in the list within the `Mcblend: UV groups` panel.



UV groups can be deleted using the `Delete UV group` button in the `Mcblend: UV groups` panel.

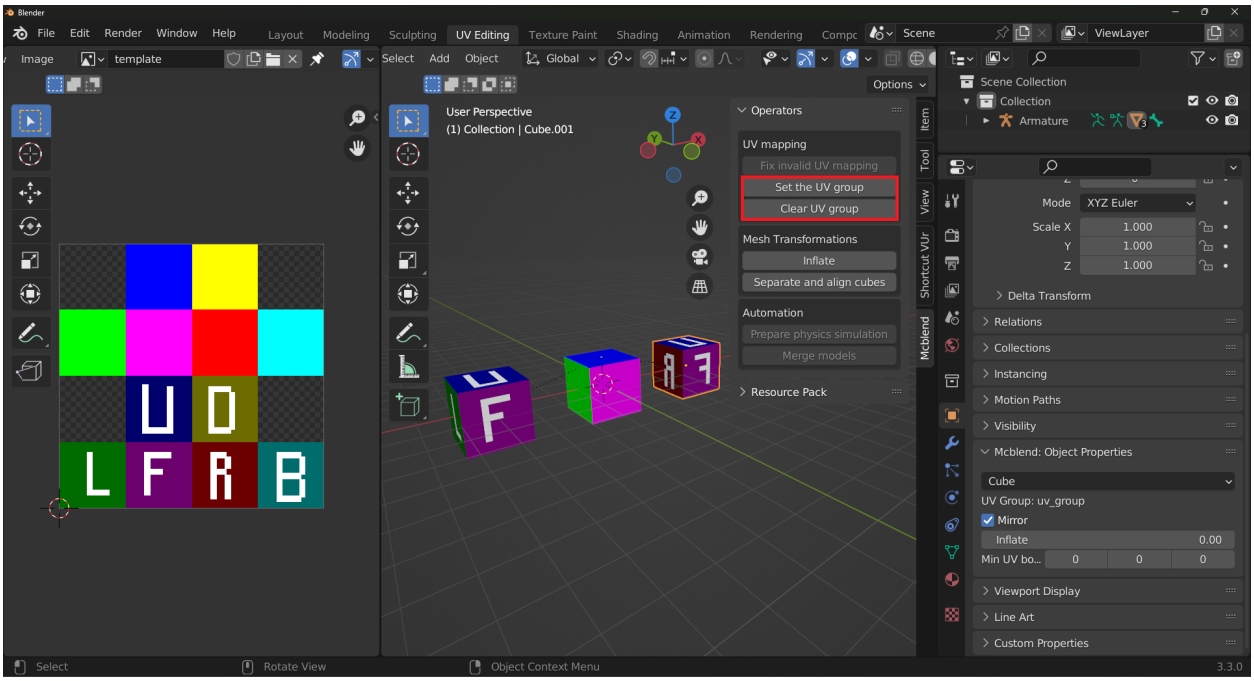
13.2 Importing and exporting UV groups

UV groups can be imported and exported as a JSON file, including the texture generation settings for the group. Use the `Import UV group` and `Export UV group` buttons in the `Mcblend: UV groups` panel to do so.



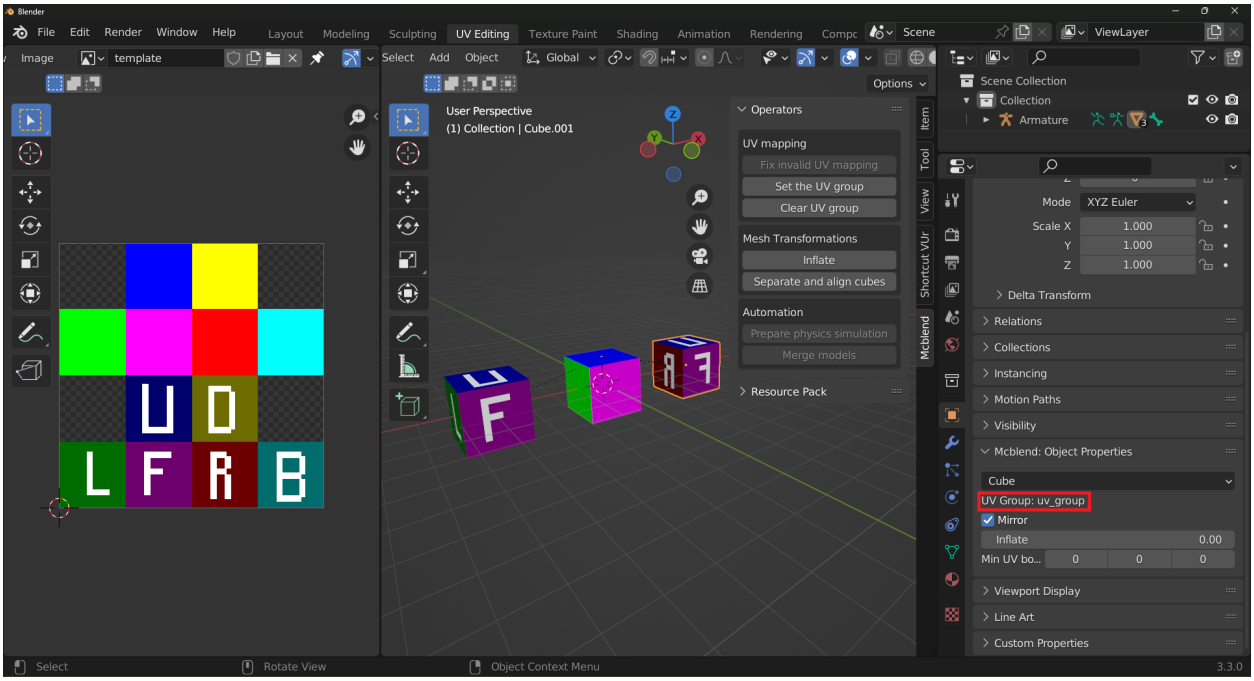
13.3 Assigning UV groups to cubes

To assign or unassign cubes to a UV group, use the `Set UV group` and `Clear UV group` operators in the Mcblend sidebar within the 3D viewport. These operators will affect all selected cubes.



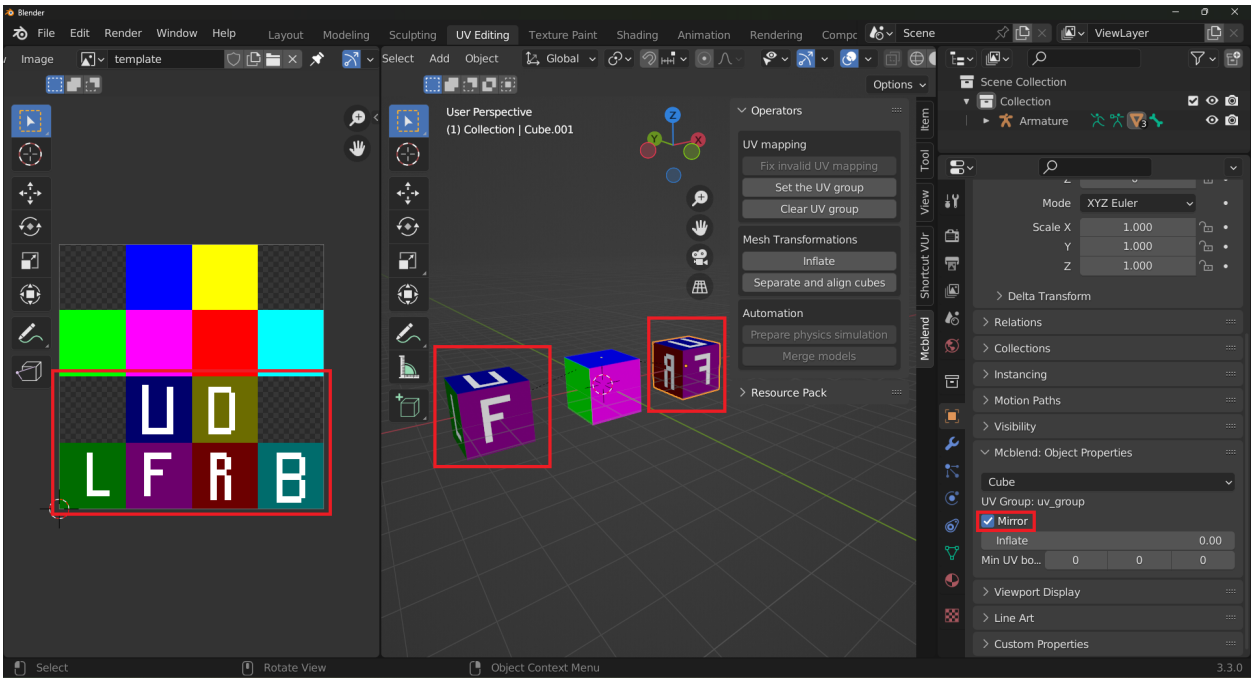
13.4 Displaying UV group information

To check which UV group a cube belongs to, select the cube and open the `Object Properties`. In the `Mcblend: Object Properties` panel, you will find a label that displays the UV group the cube belongs to.



13.5 The Mirror property

The `Mirror` property in the automatic UV mapping process affects how the texture is applied to the object. In the example below, two dark cubes are assigned to the same UV group, but one of them has the `Mirror` property enabled. This means that the texture will be mirrored along the X axis on that particular object. Both cubes are assigned to the same texture space, but the texture will appear differently on the object with the `Mirror` property enabled due to the mirroring effect.

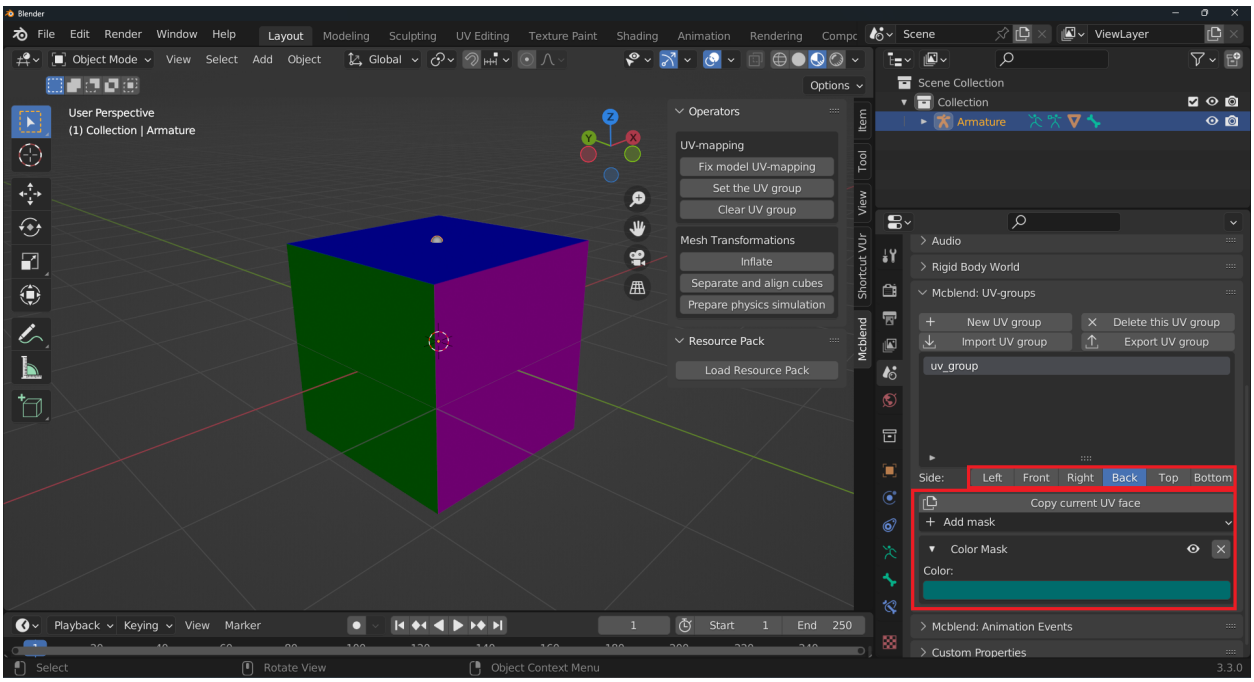


CHAPTER

FOURTEEN

CUSTOMIZING TEXTURES USING UV GROUPS

UV groups can be used to customize the appearance of textures in your model. The textures are created using collections of UV masks, with total of six collections available, one for each side of the cube - left, front, right, back, top, and bottom. You can switch between these sides using the `Side` radio button and configure the appearance of each side using a collection of UV masks. The changes to the UV groups configuration are applied during the *automatic UV mapping* process, which generates a new texture for your model. Keep in mind that you will need to generate a new texture every time you make changes to the UV group configuration in order to see the updates in your model.



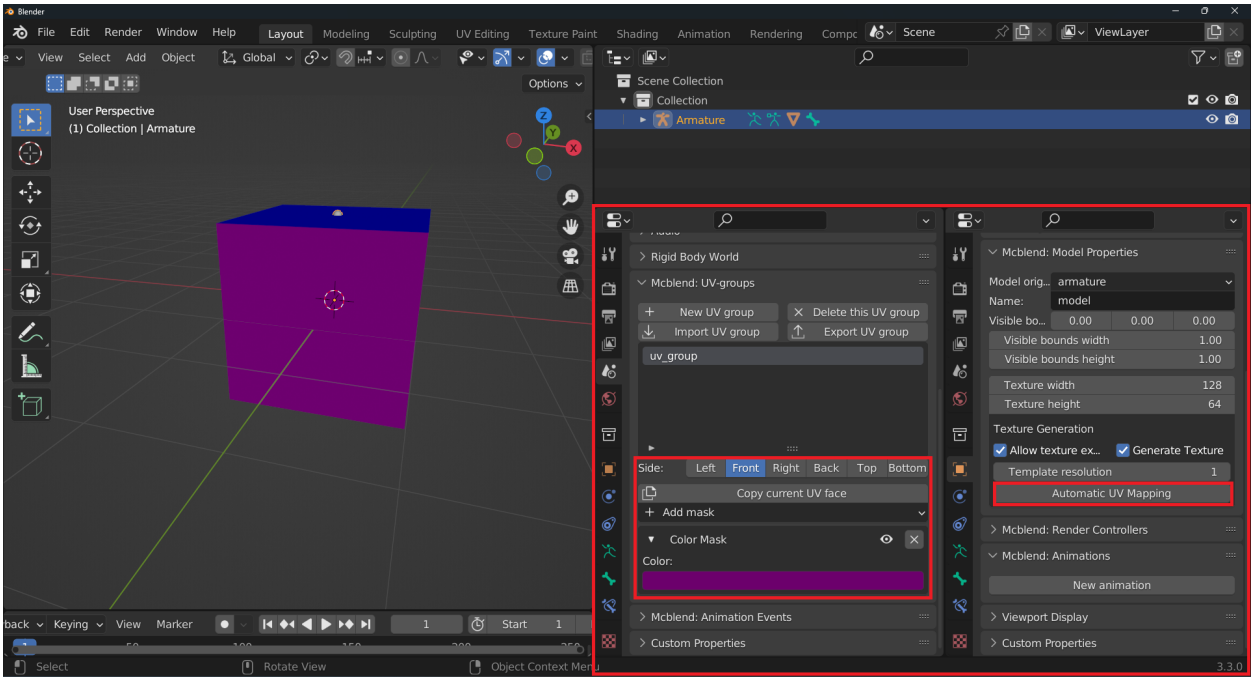
14.1 Gold block texture using UV groups (example)

This example demonstrates how to create a texture similar to the vanilla Minecraft gold block using UV groups. Note that this example does not cover every type of UV group available, but it should provide you with a general understanding of how to use them. If you would like to learn more about all the available UV groups, you can find their *complete description in the GUI reference section of the documentation*.

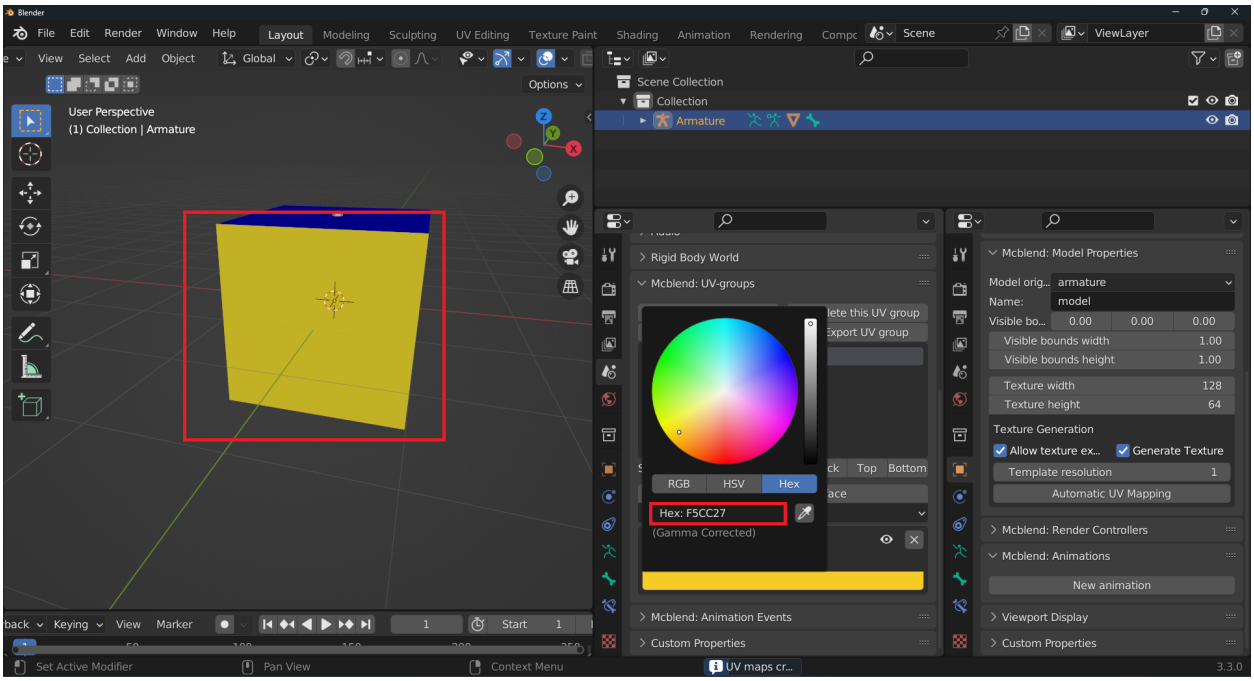
This tutorial assumes that you already have a basic model created with Mcblend, a default UV group created, and a cube assigned to it. The tutorial begins with a model that has a single cube assigned to a default UV group.

While working with UV groups for texture customization, you will frequently switch between the `Object properties` and `Scene properties` panels. The UV group configuration is located in `Scene properties`, but the button for

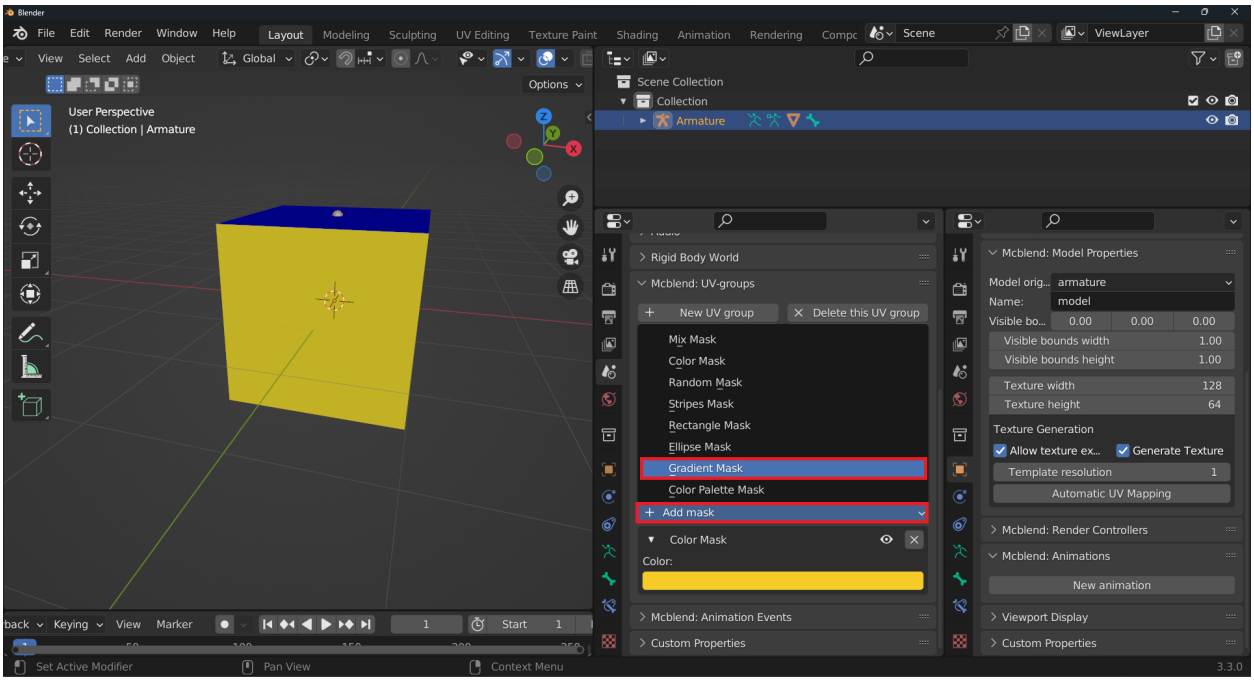
generating the texture is in `Object properties`. To make it easier to switch between these panels, you can split the properties panel into two panels. Alternatively, you can use a Blender feature to add the `Automatic UV Mapping` button to your quick favorites by right-clicking on it and selecting `Add to Quick favorites`.



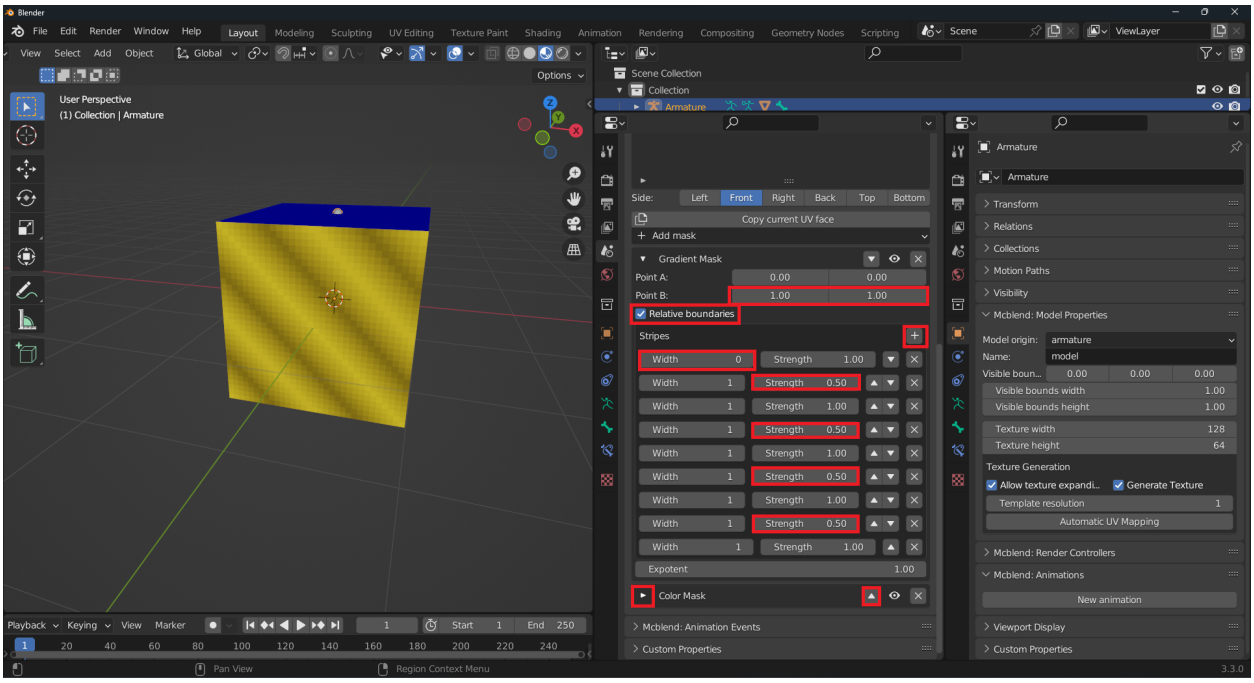
It is recommended to work on one side of a UV group and then copy the final result to the other sides if all sides are the same. Begin by switching the color of the front side to gold.



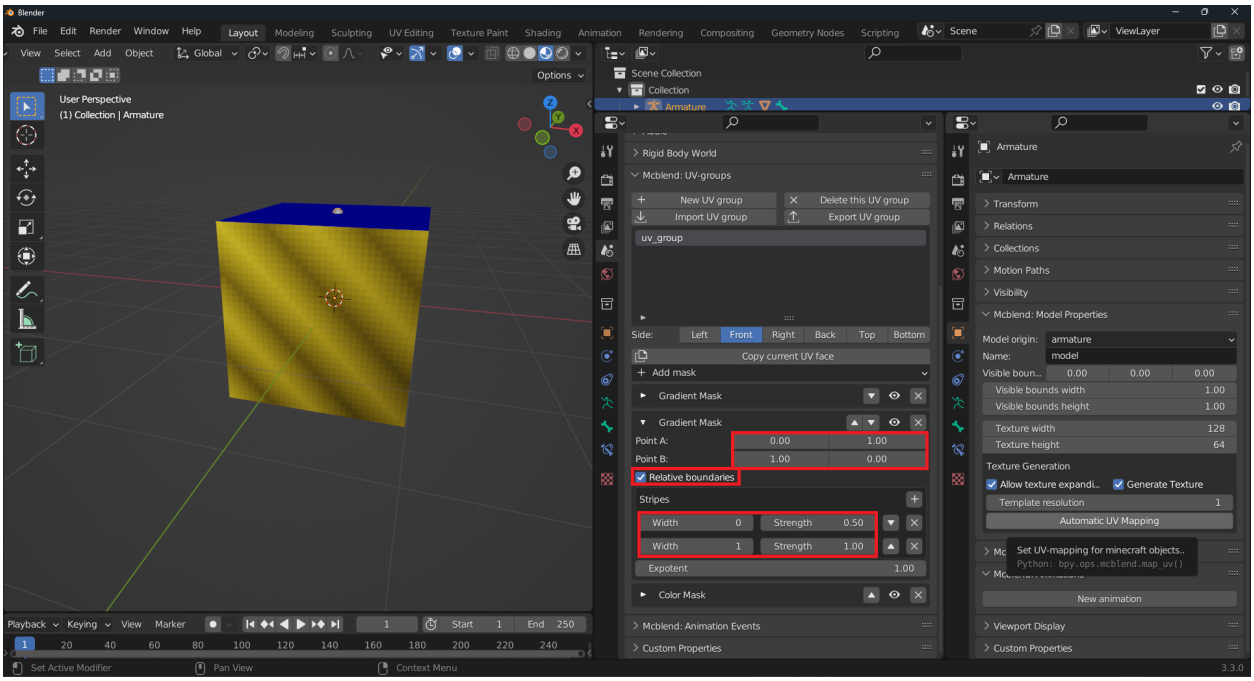
To add more masks, click the `Add mask` button. For this tutorial, add a `Gradient Mask`. Keep in mind that when you add a mask, it is added to the end of the list. You can use the arrow buttons next to the mask name to move the mask up or down in the list. If you want to hide the mask information, click the arrow on the left side of the mask name. Make sure that the `Gradient Mask` is above the `Color Mask` in the list.



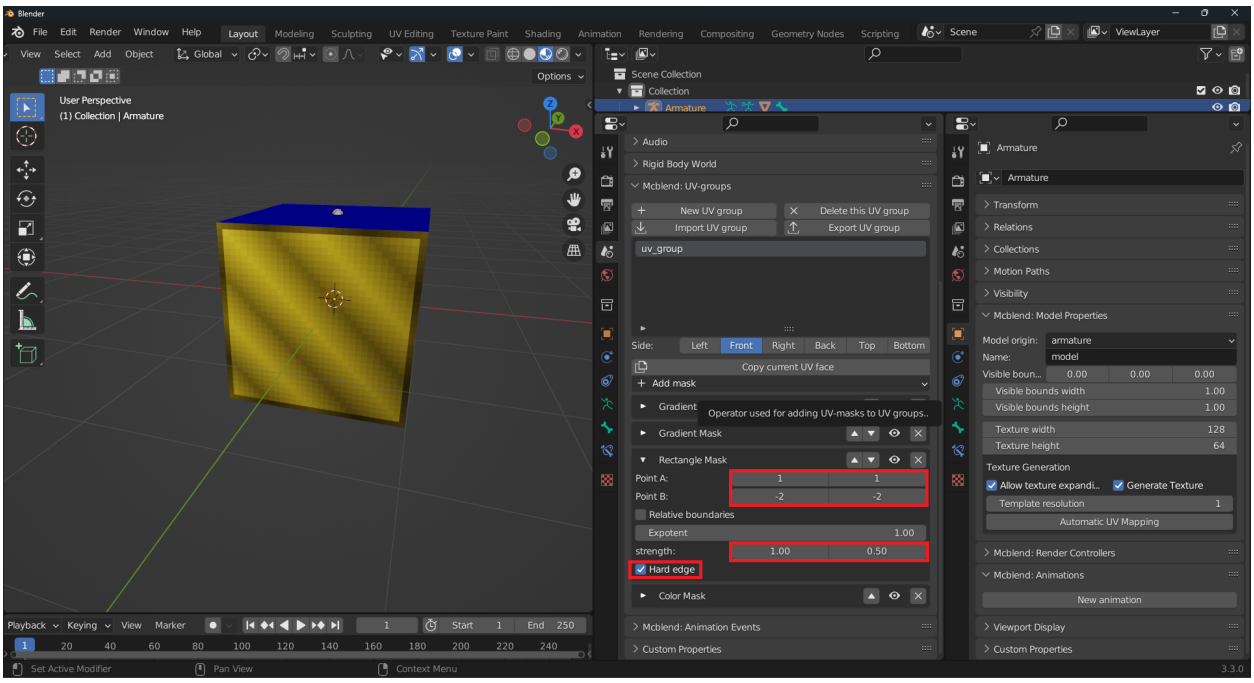
To create the diagonal stripes of a Minecraft gold block texture, you can modify the configuration of the gradient mask. Set the `Point A` to [0.0, 0.0] (the bottom left corner), and `Point B` to [1.0, 1.0] (the top right corner). The mask uses `Relative boundaries`, which means that the coordinates of `Point A` and `Point B` are represented as values between 0 and 1 (by default, these values are absolute and represent the number of pixels from the bottom left corner). There are 9 stripes alternating between values 1.0 and 0.5, creating a texture that alternates between brighter and darker colors. The first stripe's width is set to 0, as the `Width` value represents a position on the line between `Point A` and `Point B`, with 0 being the starting value.



To add a shadow in the bottom right corner, add another `Gradient Mask` after the one for the stripes.



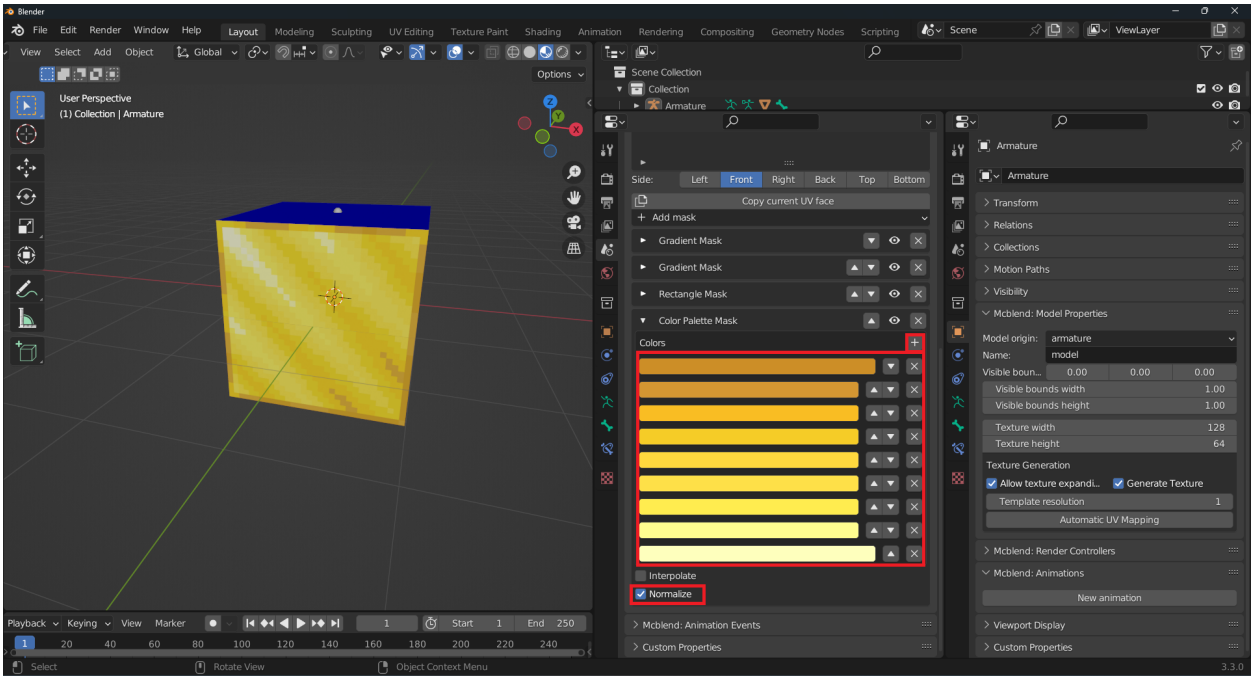
The `Rectangle Mask` creates a grayscale image of a rectangle between `Point A` and `Point B`. The grayscale image is then multiplied by the input image. In this case, the `Rectangle Mask` is used to add a frame with a width of 1 pixel. It's not using `Relative boundaries`, so the coordinates are absolute. The negative values for `Point A` and `Point B` are used to count pixels from the opposite corner. -2 represents 1 pixel counting from the top right corner (0 represents the bottom left corner, and -1 represents the top right corner). The rectangle uses `Hard edge`, which means that the transition between the bright and dark side is instant.



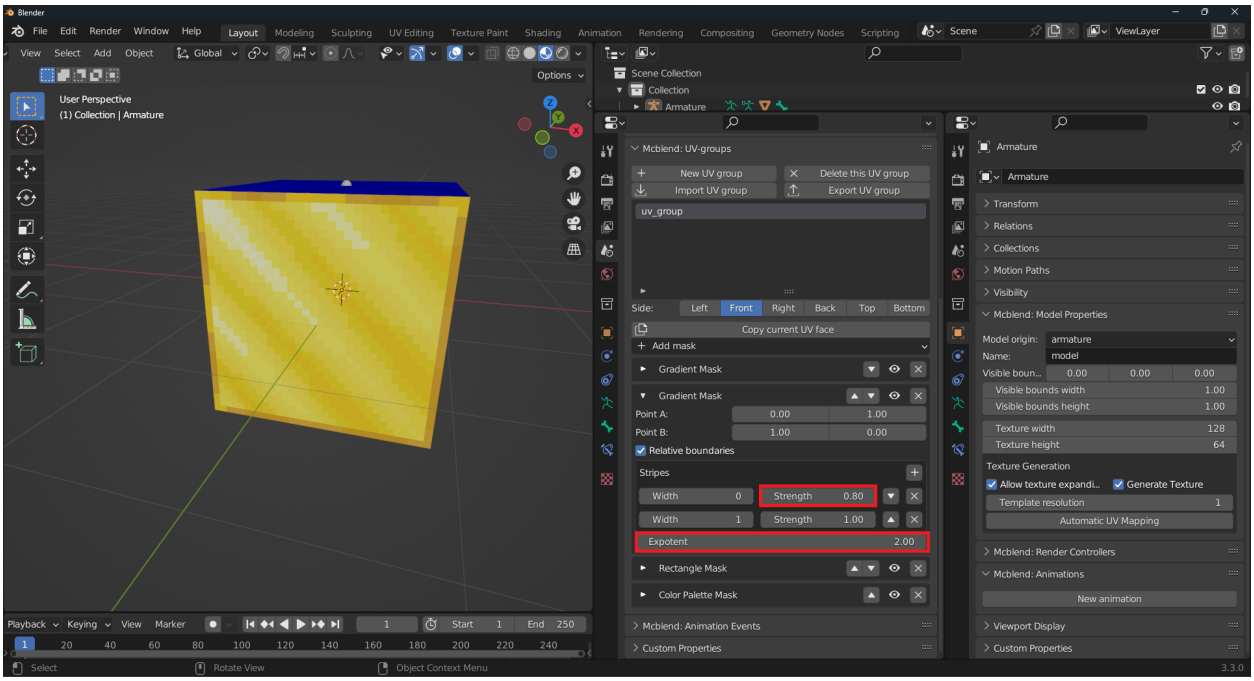
The gold color doesn't look very good, so let's remove it and add a `Color Palette` Mask instead. This mask maps the brightness values of the image to a color image based on a palette defined as a list of colors. By enabling the `Normalize` option, the input values are normalized so that the entire palette is used. The `Interpolate` option allows for a smooth transition between the colors in the palette by mapping brightness values to colors that are not defined in

the palette based on interpolation between the defined colors. In this case, we are not using the `Interpolate` option, as our palette has 9 colors, which should be sufficient. The colors in the palette were based on those used in the actual Minecraft gold block texture, which can be extracted using external software like Gimp. The list of colors in our palette is as follows:

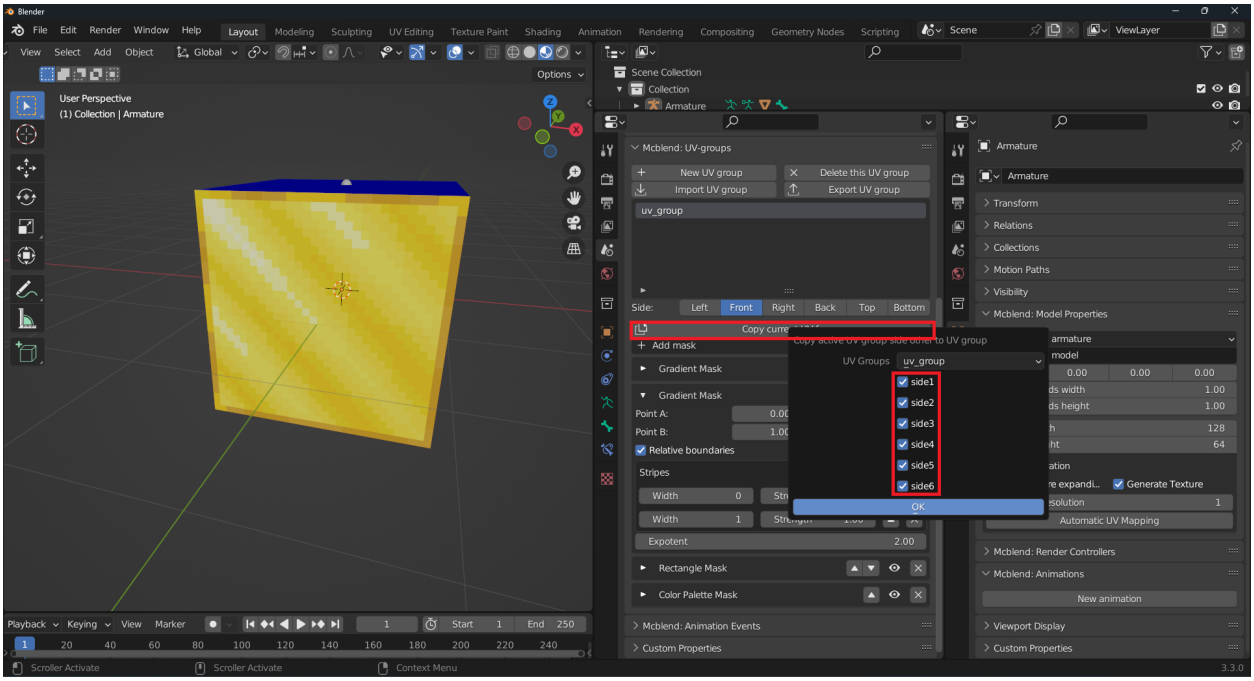
- #cc8e27
- #d39632
- #f9bd23
- #f5cc27
- #ffd83e
- #fee048
- #ffec4f
- #fffd90
- #feffbd



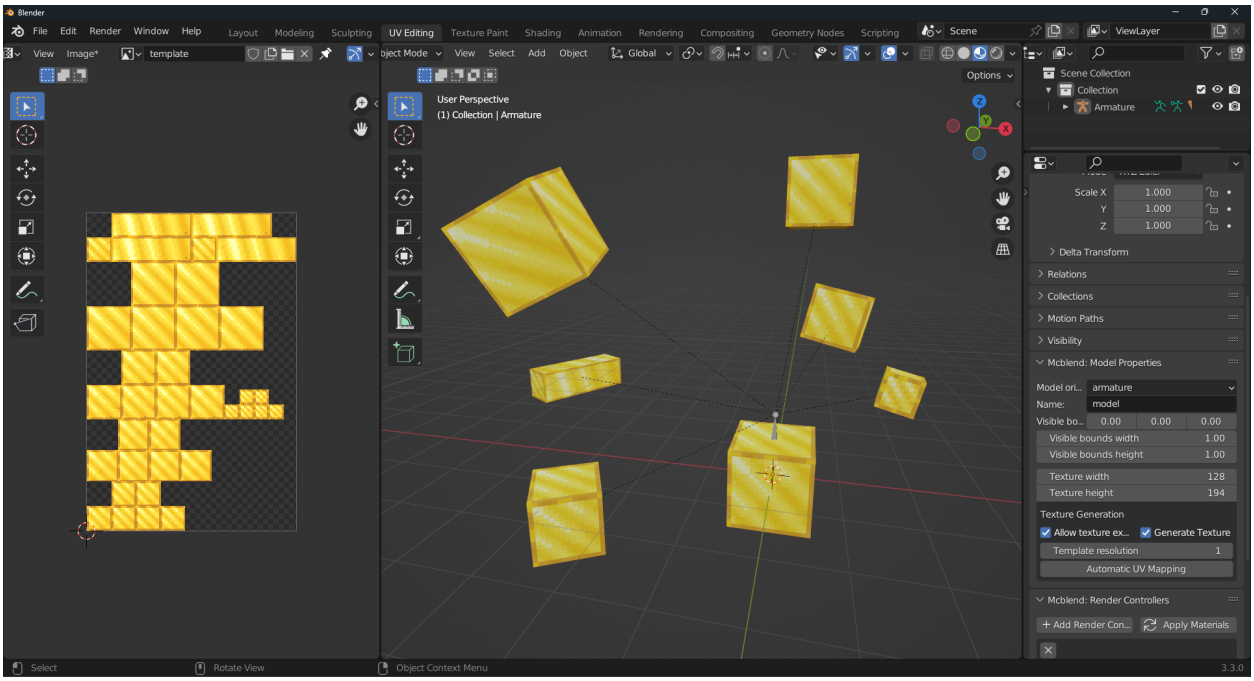
The `Gradient Mask` used for creating shadows was modified to achieve a more realistic look. The color value for the darker shade was changed to 0.8 and the Exponent was set to 2.0. The exponent value causes the color change to be more sudden, resulting in a more defined shadow. Keep in mind that fine-tuning the settings of the masks may be necessary to achieve the desired result.



After you have finished defining the texture, you can use the `Copy current UV face` button to copy it to other faces.



Once you have completed the UV group configuration, you can apply it to cubes of different shapes and sizes and the texture will be automatically generated. This allows you to easily apply the same texture to multiple objects without needing to manually configure each one.



The UV group can be exported to a JSON file for use in other models or projects. The JSON file contains all the configuration details of the UV group, including the masks and their properties. An example of the exported UV group used in this tutorial is provided below for reference. You can import it from the file. Simply click the `Import UV group` button and select the desired file.

```
{
        "version": 1,
        "name": "gold",
        "side1": [
                {
                        "mask_type": "Gradient Mask",
                        "p1": [0.0, 0.0],
                        "p2": [1.0, 1.0],
                        "stripes": [
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                },
                                {
                                        "strength": 0.5,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                },
                                {
                                        "strength": 0.5,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
```

(continues on next page)

(continued from previous page)

```
                                "width": 0.1
                        },
                        {
                                "strength": 0.5,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        },
                        {
                                "strength": 0.5,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        }
                ],
                "relative_boundaries": true,
                "expotent": 1.0
        },
        {
                "mask_type": "Gradient Mask",
                "p1": [0.0, 1.0],
                "p2": [1.0, 0.0],
                "stripes": [
                        {
                                "strength": 0.800000011920929,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        }
                ],
                "relative_boundaries": true,
                "expotent": 2.0
        },
        {
                "mask_type": "Rectangle Mask",
                "p1": [1, 1],
                "p2": [-2, -2],
                "relative_boundaries": false,
                "expotent": 1.0,
                "strength": [1.0, 0.5],
                "hard_edge": true
        },
        {
                "mask_type": "Color Palette Mask",
                "colors": [
                        [0.60382724, 0.27049762, 0.02028864],
```

(continues on next page)

(continued from previous page)

```
                                [0.65140569, 0.3049871, 0.03189614],
                                [0.94730628, 0.50888097, 0.01680754],
                                [0.91309863, 0.60382688, 0.02028875],
                                [1.0, 0.68668491, 0.04817205],
                                [0.9911015, 0.74540383, 0.06480349],
                                [1.0, 0.83879864, 0.07818766],
                                [1.0, 0.98224992, 0.27889451],
                                [0.99110109, 1.0, 0.50888151]
                        ],
                        "interpolate": false,
                        "normalize": true
                }
        ],
        "side2": [
                {
                        "mask_type": "Gradient Mask",
                        "p1": [0.0, 0.0],
                        "p2": [1.0, 1.0],
                        "stripes": [
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                },
                                {
                                        "strength": 0.5,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                },
                                {
                                        "strength": 0.5,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                },
                                {
                                        "strength": 0.5,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                },
                                {
                                        "strength": 0.5,
                                        "width": 0.1
                                },
                                {
```

(continues on next page)

(continued from previous page)

```
                                        "strength": 1.0,
                                        "width": 0.1
                                }
                        ],
                        "relative_boundaries": true,
                        "expotent": 1.0
                },
                {
                        "mask_type": "Gradient Mask",
                        "p1": [0.0, 1.0],
                        "p2": [1.0, 0.0],
                        "stripes": [
                                {
                                        "strength": 0.8000000011920929,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                }
                        ],
                        "relative_boundaries": true,
                        "expotent": 2.0
                },
                {
                        "mask_type": "Rectangle Mask",
                        "p1": [1, 1],
                        "p2": [-2, -2],
                        "relative_boundaries": false,
                        "expotent": 1.0,
                        "strength": [1.0, 0.5],
                        "hard_edge": true
                },
                {
                        "mask_type": "Color Palette Mask",
                        "colors": [
                                [0.60382724, 0.27049762, 0.02028864],
                                [0.65140569, 0.3049871, 0.03189614],
                                [0.94730628, 0.50888097, 0.01680754],
                                [0.91309863, 0.60382688, 0.02028875],
                                [1.0, 0.68668491, 0.04817205],
                                [0.9911015, 0.74540383, 0.06480349],
                                [1.0, 0.83879864, 0.07818766],
                                [1.0, 0.98224992, 0.27889451],
                                [0.99110109, 1.0, 0.50888151]
                        ],
                        "interpolate": false,
                        "normalize": true
                }
        ],
        "side3": [
                {
```

(continues on next page)

(continued from previous page)

```
                "mask_type": "Gradient Mask",
                "p1": [0.0, 0.0],
                "p2": [1.0, 1.0],
                "stripes": [
                        {
                                "strength": 1.0,
                                "width": 0.1
                        },
                        {
                                "strength": 0.5,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        },
                        {
                                "strength": 0.5,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        },
                        {
                                "strength": 0.5,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        },
                        {
                                "strength": 0.5,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        }
                ],
                "relative_boundaries": true,
                "expotent": 1.0
        },
        {
                "mask_type": "Gradient Mask",
                "p1": [0.0, 1.0],
                "p2": [1.0, 0.0],
                "stripes": [
                        {
                                "strength": 0.800000011920929,
                                "width": 0.1
```

(continues on next page)

(continued from previous page)

```
                                },
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                }
                        ],
                        "relative_boundaries": true,
                        "expotent": 2.0
                },
                {
                        "mask_type": "Rectangle Mask",
                        "p1": [1, 1],
                        "p2": [-2, -2],
                        "relative_boundaries": false,
                        "expotent": 1.0,
                        "strength": [1.0, 0.5],
                        "hard_edge": true
                },
                {
                        "mask_type": "Color Palette Mask",
                        "colors": [
                                [0.60382724, 0.27049762, 0.02028864],
                                [0.65140569, 0.3049871, 0.03189614],
                                [0.94730628, 0.50888097, 0.01680754],
                                [0.91309863, 0.60382688, 0.02028875],
                                [1.0, 0.68668491, 0.04817205],
                                [0.9911015, 0.74540383, 0.06480349],
                                [1.0, 0.83879864, 0.07818766],
                                [1.0, 0.98224992, 0.27889451],
                                [0.99110109, 1.0, 0.50888151]
                        ],
                        "interpolate": false,
                        "normalize": true
                }
        ],
        "side4": [
                {
                        "mask_type": "Gradient Mask",
                        "p1": [0.0, 0.0],
                        "p2": [1.0, 1.0],
                        "stripes": [
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                },
                                {
                                        "strength": 0.5,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
                                        "width": 0.1
```

(continues on next page)

(continued from previous page)


```
                        },
                        {
                                "strength": 0.5,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        },
                        {
                                "strength": 0.5,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        },
                        {
                                "strength": 0.5,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        }
                ],
                "relative_boundaries": true,
                "expotent": 1.0
        },
        {
                "mask_type": "Gradient Mask",
                "p1": [0.0, 1.0],
                "p2": [1.0, 0.0],
                "stripes": [
                        {
                                "strength": 0.800000011920929,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        }
                ],
                "relative_boundaries": true,
                "expotent": 2.0
        },
        {
                "mask_type": "Rectangle Mask",
                "p1": [1, 1],
                "p2": [-2, -2],
                "relative_boundaries": false,
                "expotent": 1.0,
```


(continues on next page)

(continued from previous page)

```
                        "strength": [1.0, 0.5],
                        "hard_edge": true
                },
                {
                        "mask_type": "Color Palette Mask",
                        "colors": [
                                [0.60382724, 0.27049762, 0.02028864],
                                [0.65140569, 0.3049871, 0.03189614],
                                [0.94730628, 0.50888097, 0.01680754],
                                [0.91309863, 0.60382688, 0.02028875],
                                [1.0, 0.68668491, 0.04817205],
                                [0.9911015, 0.74540383, 0.06480349],
                                [1.0, 0.83879864, 0.07818766],
                                [1.0, 0.98224992, 0.27889451],
                                [0.99110109, 1.0, 0.50888151]
                        ],
                        "interpolate": false,
                        "normalize": true
                }
        ],
        "side5": [
                {
                        "mask_type": "Gradient Mask",
                        "p1": [0.0, 0.0],
                        "p2": [1.0, 1.0],
                        "stripes": [
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                },
                                {
                                        "strength": 0.5,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                },
                                {
                                        "strength": 0.5,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                },
                                {
                                        "strength": 0.5,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
```

(continues on next page)

(continued from previous page)

```
                                "width": 0.1
                        },
                        {
                                "strength": 0.5,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        }
                ],
                "relative_boundaries": true,
                "expotent": 1.0
        },
        {
                "mask_type": "Gradient Mask",
                "p1": [0.0, 1.0],
                "p2": [1.0, 0.0],
                "stripes": [
                        {
                                "strength": 0.800000011920929,
                                "width": 0.1
                        },
                        {
                                "strength": 1.0,
                                "width": 0.1
                        }
                ],
                "relative_boundaries": true,
                "expotent": 2.0
        },
        {
                "mask_type": "Rectangle Mask",
                "p1": [1, 1],
                "p2": [-2, -2],
                "relative_boundaries": false,
                "expotent": 1.0,
                "strength": [1.0, 0.5],
                "hard_edge": true
        },
        {
                "mask_type": "Color Palette Mask",
                "colors": [
                        [0.60382724, 0.27049762, 0.02028864],
                        [0.65140569, 0.3049871, 0.03189614],
                        [0.94730628, 0.50888097, 0.01680754],
                        [0.91309863, 0.60382688, 0.02028875],
                        [1.0, 0.68668491, 0.04817205],
                        [0.9911015, 0.74540383, 0.06480349],
                        [1.0, 0.83879864, 0.07818766],
                        [1.0, 0.98224992, 0.27889451],
                        [0.99110109, 1.0, 0.50888151]
```

(continues on next page)

(continued from previous page)

```
                    ],
                    "interpolate": false,
                    "normalize": true
            }
        ],
        "side6": [
            {
                    "mask_type": "Gradient Mask",
                    "p1": [0.0, 0.0],
                    "p2": [1.0, 1.0],
                    "stripes": [
                            {
                                    "strength": 1.0,
                                    "width": 0.1
                            },
                            {
                                    "strength": 0.5,
                                    "width": 0.1
                            },
                            {
                                    "strength": 1.0,
                                    "width": 0.1
                            },
                            {
                                    "strength": 0.5,
                                    "width": 0.1
                            },
                            {
                                    "strength": 1.0,
                                    "width": 0.1
                            },
                            {
                                    "strength": 0.5,
                                    "width": 0.1
                            },
                            {
                                    "strength": 1.0,
                                    "width": 0.1
                            },
                            {
                                    "strength": 0.5,
                                    "width": 0.1
                            },
                            {
                                    "strength": 1.0,
                                    "width": 0.1
                            }
                    ],
                    "relative_boundaries": true,
                    "expotent": 1.0
            },
            {
```

(continues on next page)

(continued from previous page)

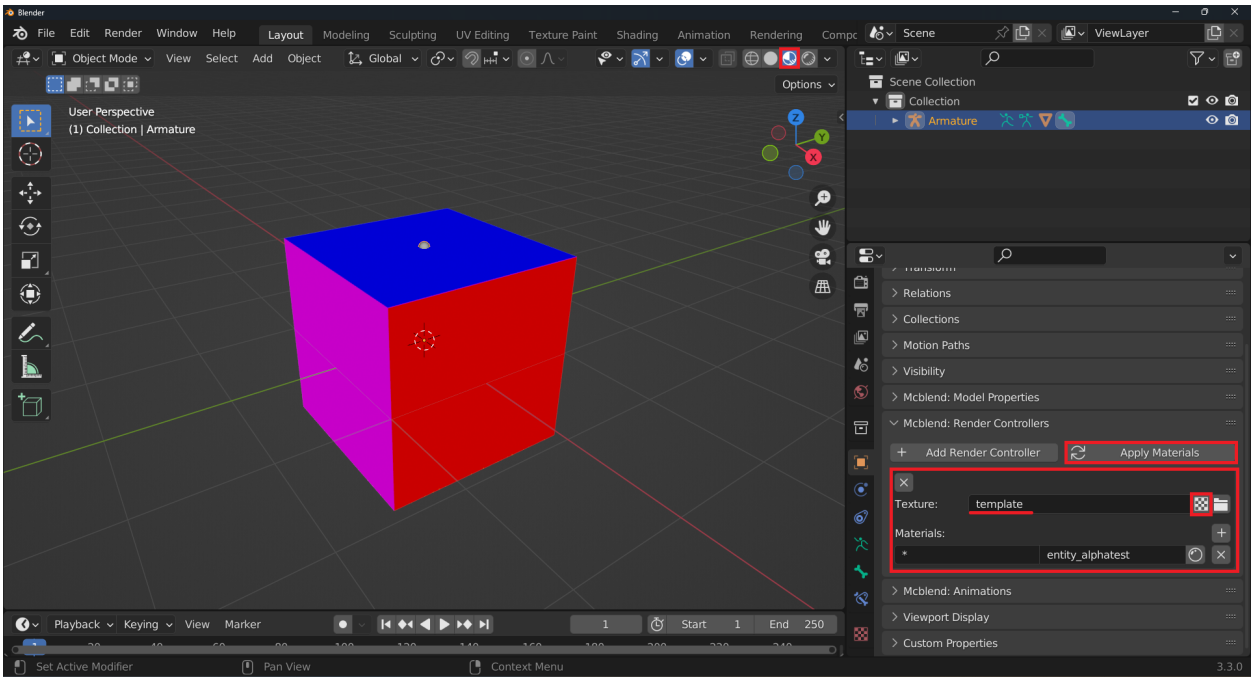
```
                        "mask_type": "Gradient Mask",
                        "p1": [0.0, 1.0],
                        "p2": [1.0, 0.0],
                        "stripes": [
                                {
                                        "strength": 0.800000011920929,
                                        "width": 0.1
                                },
                                {
                                        "strength": 1.0,
                                        "width": 0.1
                                }
                        ],
                        "relative_boundaries": true,
                        "expotent": 2.0
                },
                {
                        "mask_type": "Rectangle Mask",
                        "p1": [1, 1],
                        "p2": [-2, -2],
                        "relative_boundaries": false,
                        "expotent": 1.0,
                        "strength": [1.0, 0.5],
                        "hard_edge": true
                },
                {
                        "mask_type": "Color Palette Mask",
                        "colors": [
                                [0.60382724, 0.27049762, 0.02028864],
                                [0.65140569, 0.3049871, 0.03189614],
                                [0.94730628, 0.50888097, 0.01680754],
                                [0.91309863, 0.60382688, 0.02028875],
                                [1.0, 0.68668491, 0.04817205],
                                [0.9911015, 0.74540383, 0.06480349],
                                [1.0, 0.83879864, 0.07818766],
                                [1.0, 0.98224992, 0.27889451],
                                [0.99110109, 1.0, 0.50888151]
                        ],
                        "interpolate": false,
                        "normalize": true
                }
        ]
}
```

CHAPTER

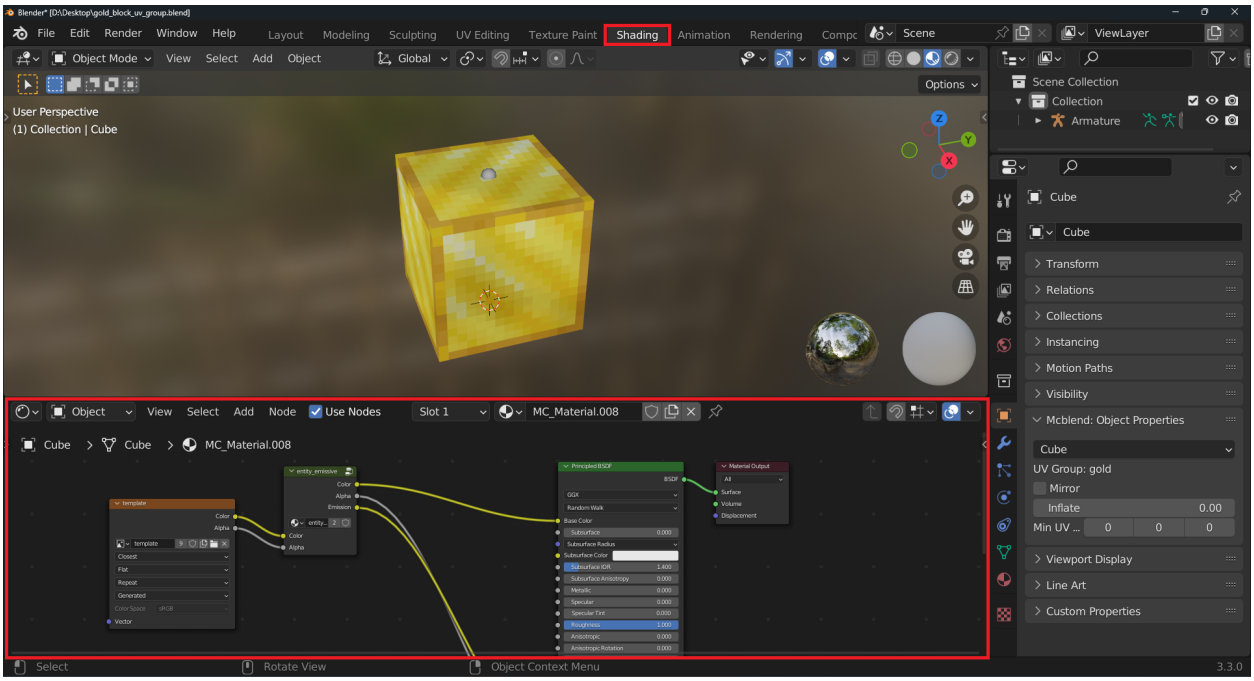
FIFTEEN

MATERIALS AND RENDER CONTROLLERS

In Mcblend, render controllers control which bones of a model use which materials and textures. You can configure the render controllers of the model by selecting an armature and opening the `Mcblend: Render Controllers` panel in the `Object properties`. These render controllers are used during the process of *automatic UV mapping*, and changes to them can be seen by running the `Automatic UV Mapping` operator.



Render controllers in Mcblend are simply tools for creating Blender materials. You can viewed these materials in the `Shading` workspace.



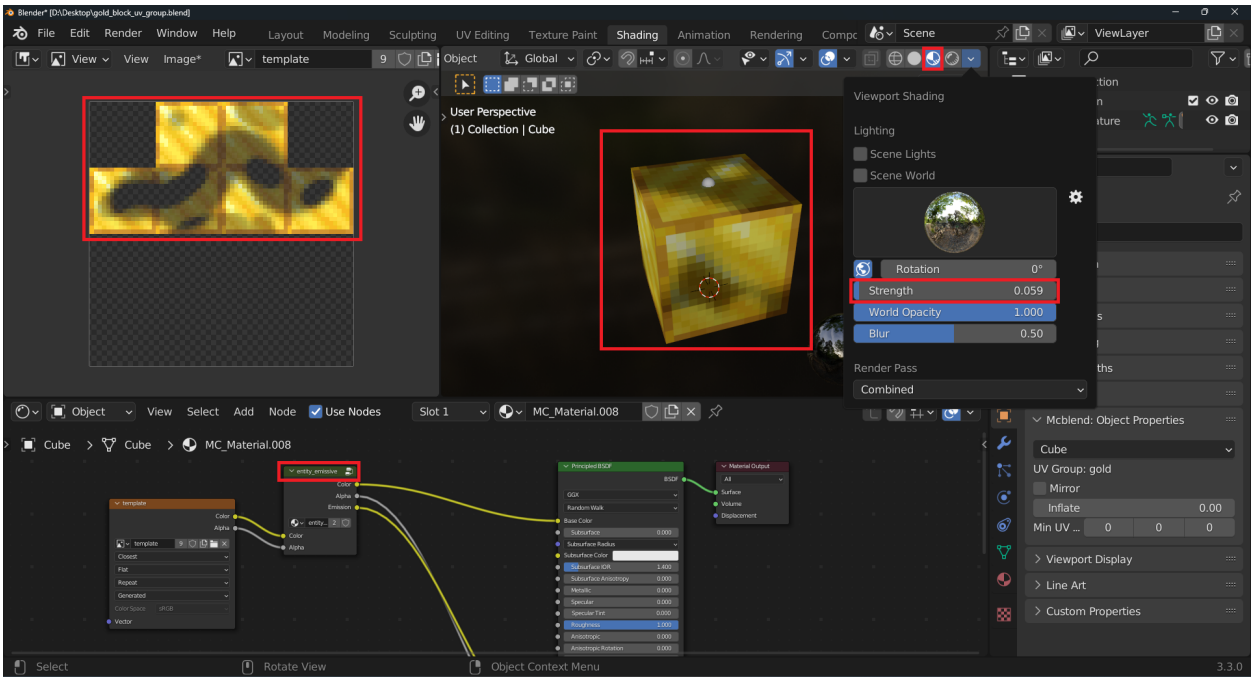
If one bone is displayed by multiple render controllers, their properties will be stacked on top of each other in the same way as they would be in Minecraft for a model with multiple render controllers displaying the same bone. The star patterns in Mcblend render controllers work in the same way as the patterns in Minecraft for matching the names of bones.

There are six materials supported by Mcblend that can be used to configure render controllers:

- `entity`: imitates the entity material, with no emissive properties, complete opacity, and no backface culling.
- `entity_emissive`: uses the alpha channel for emissive properties, complete opacity, and no backface culling.
- `entity_alphablend`: no emissive properties, uses the alpha channel, and has backface culling.
- `entity_alphatest_one_sided`: no emissive properties, backface culling, and parts of the model below 50% transparency are invisible, while the rest is opaque.
- `entity_alphatest`: no emissive properties, parts of the model below 50% transparency are completely transparent, the rest is opaque, and there is no backface culling.

Mcblend also recognizes some other materials used in Minecraft when their names are inserted into the text field, but these are simply aliases for the materials listed above.

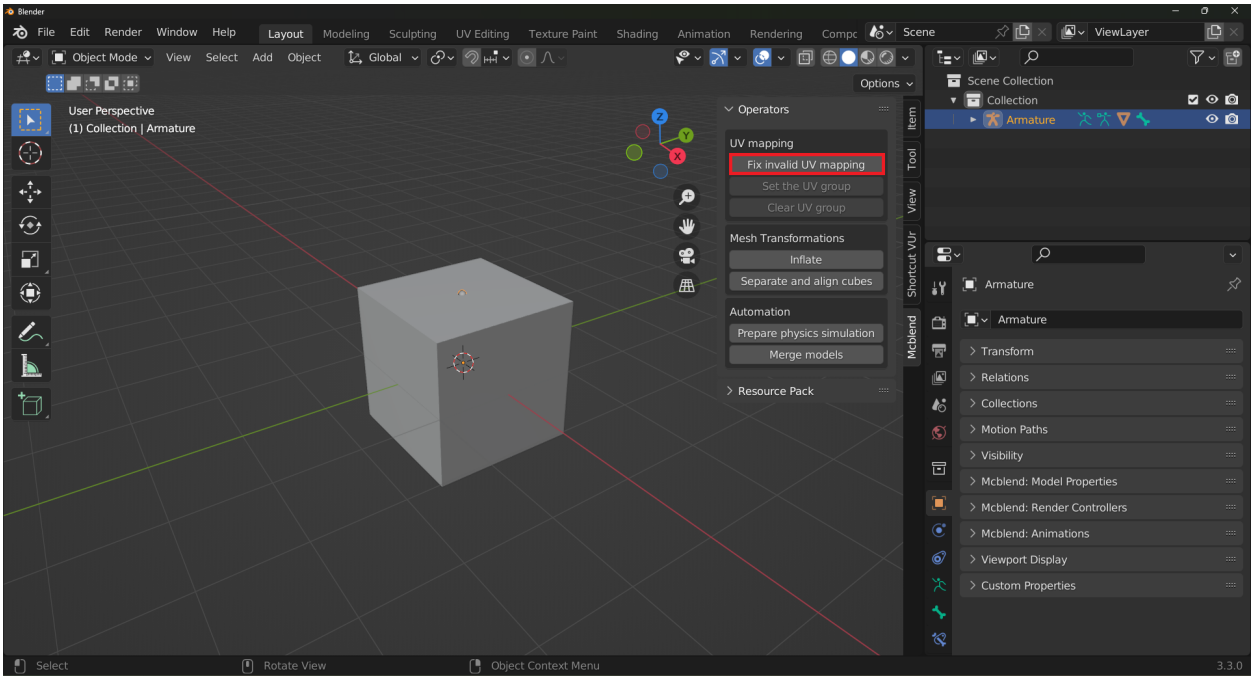
The emissive properties of a texture can be seen by lowering the `Strength of the studiolight` while viewing the model in `Material Preview` mode in the viewport shading. The image below shows an emissive material and its texture.


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering
Scene
ViewLayer
User Perspective
(1) Collection | Cube
Viewport Shading
Lighting
Scene Lights
Scene World
Rotation 0°
Strength 0.059
World Opacity 1.000
Blur 0.50
Render Pass
Combined
Object View Select Add Node Use Nodes
Slot 1
MC_Material.008
Cube > Cube > MC_Material.008
Mcblend: Object Properties
Cube
UV Group: gold
Mirror
Inflate 0.00
Min UV ...
Viewport Display
Line Art
Custom Properties
Select
Rotate View
Object Context Menu
3.3.0

CHAPTER

SIXTEEN

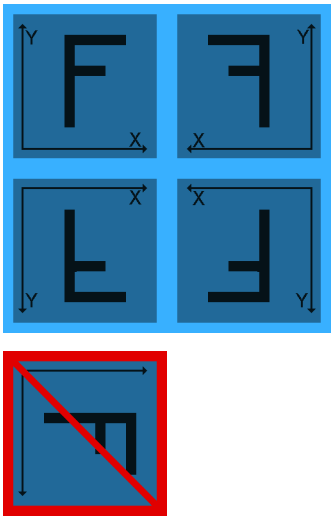
FIXING INVALID UV MAPPING



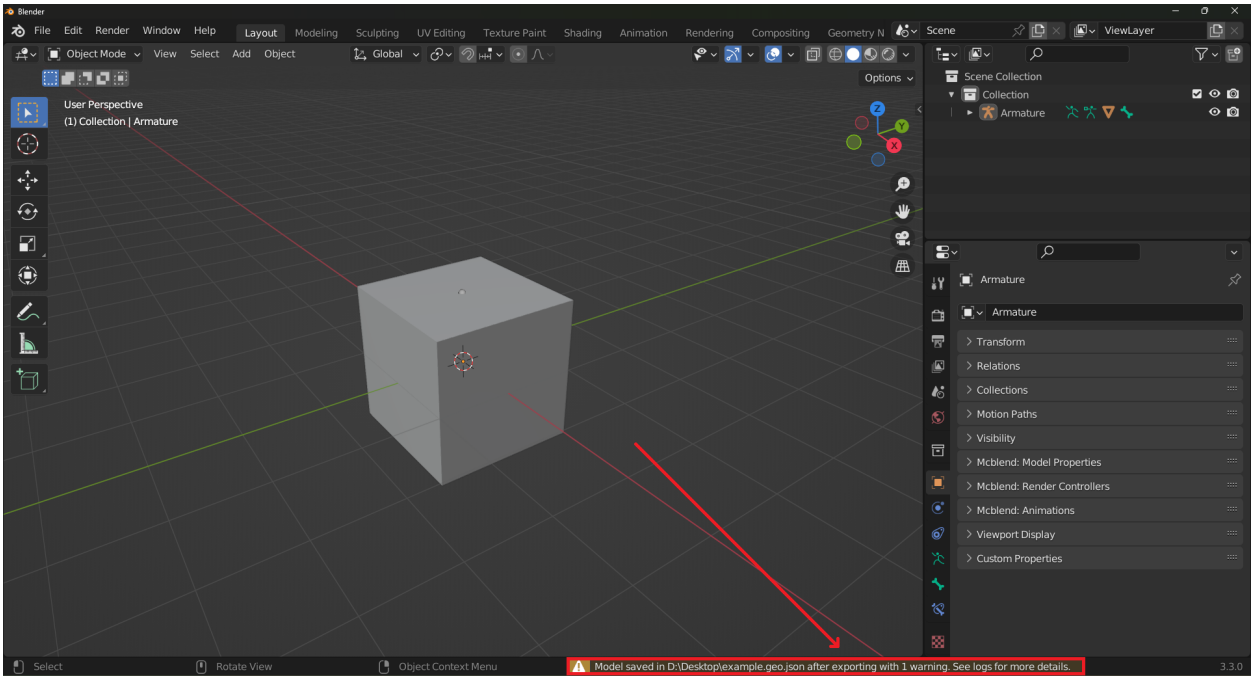
The `Fix invalid UV mapping` operator is a tool for fixing rotated rectangles in the UV maps of a model. In Minecraft, there are two types of UV mapping for cubes: default and per-face. Mcblend automatically selects the appropriate mapping mode, so you don't have to worry about it while working with the model.

The per-face UV mapping is more flexible, as it allows you to set the UV map of each face of the cube separately. However, it is still quite limited. Internally, it saves the UV information as two vectors: one representing the UV coordinates on the texture, and the other representing the size of the rectangle on the texture to be mapped onto the face of the cube. This format allows for mirroring the cube along the X and Y axes, but does not allow for rotating it by 90 degrees.

The image below shows a texture with the letter "F" on it and its possible transformations achieved through mirroring. The second image depicts a rotated "F" which cannot be achieved by mirroring alone.



The Fix invalid UV mapping operator is used to fix rotated rectangles in the UV maps. If a cube in the model has an invalid UV map and you try to export it, Mcblend will do so, but it will also print a warning message about the invalid UV map and will not include the UV map information in the cube.



To fix invalid UV maps on your model, select the armature and use the `Fix invalid UV mapping` button in the 3D viewport sidebar under the `Mcblend` tab. This will fix any invalid UV maps for all cubes in the model.

CHAPTER

SEVENTEEN

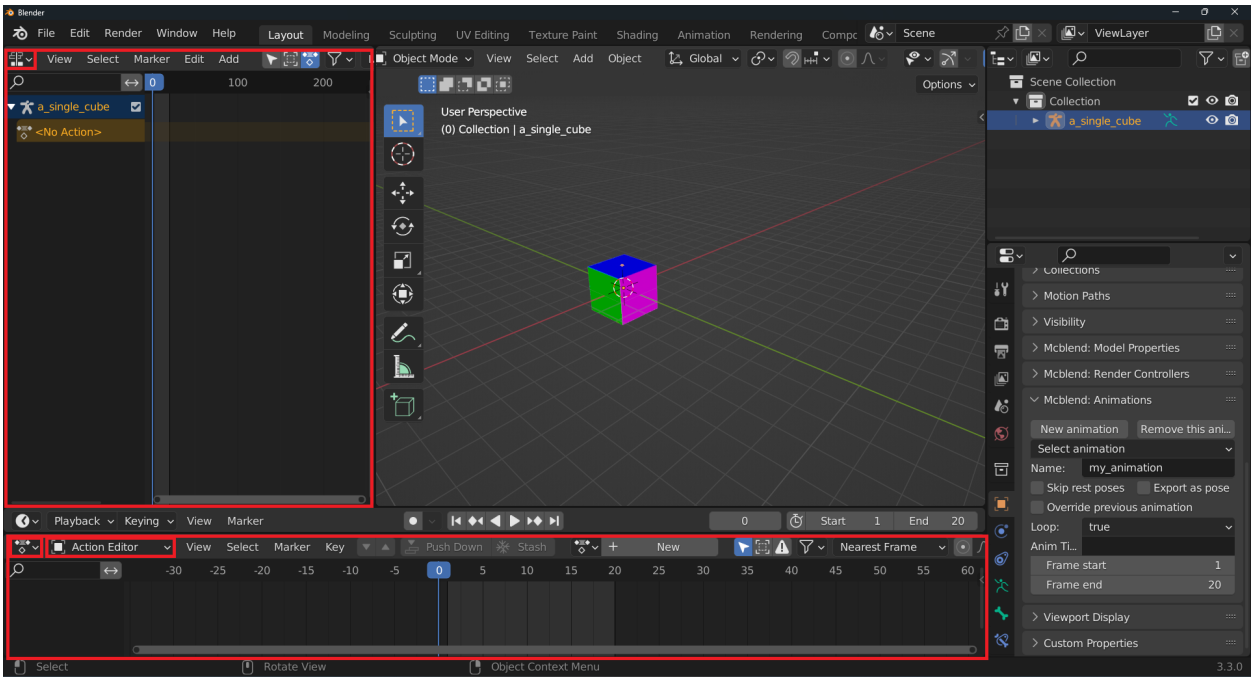
ESSENTIAL CONCEPTS FOR CREATING ANIMATIONS IN MCBLEND

This page covers essential concepts to consider when creating animations using Mcblend. It is recommended that you read through this information before beginning to animate with Mcblend.

17.1 Recomended workspace layout

Animations in Mcblend are based on Non Linear Animation (NLA) tracks. A single animation can contain multiple NLA tracks running at the same time, each track containing action strips. While working with animations in Mcblend, you'll be using the NLA tracks and action editor frequently. It is recommended to set up a workspace layout like the one shown in the image below for working with animations in Mcblend.

The editor on the left side of the screen is the `Nonlinear Animation` editor. The editor at the bottom is the `Dope Sheet` editor set to the `Action editor` mode.



17.2 Animation as a sequence of poses

During the export process, the exported data for animations is not solely based on the NLA tracks. Mcblend exports the same animation that is visible in the 3D viewport during the preview, which can be affected by various factors such as constraints, inverse kinematics, and physics. This means that some bones in the model may not be animated in the NLA tracks, but they will be animated in the exported animation due to the influence of other factors that affect their movement. The keyframe times for the animations are based on the keyframe times in the NLA tracks, but not at the bone level. Essentially, if there is any keyframe at a certain time, Mcblend compares the pose of the entire model at that time to the pose of the model at the previous keyframe. If the pose is different for a given bone, Mcblend will add a keyframe for that bone to the exported animation. Any pose changes that occur between keyframes are not checked, which can lead to unexpected results, particularly when using physics, which often generates complex movement that requires many keyframes to be animated correctly.

17.3 Frame 0

In Mcblend, animations are based on comparing the poses of the model at different times. To do this, there must be a frame that defines the base pose of the model. This frame is frame 0 (it's 1 frame before the first frame and is technically not a part of the animation). Frame 0 has a special role in Mcblend and the model should be in its base pose during this frame. If it is not, the animations will not export properly. Frame 0 is also the frame used during the `Export Bedrock Model` operation.

CHAPTER

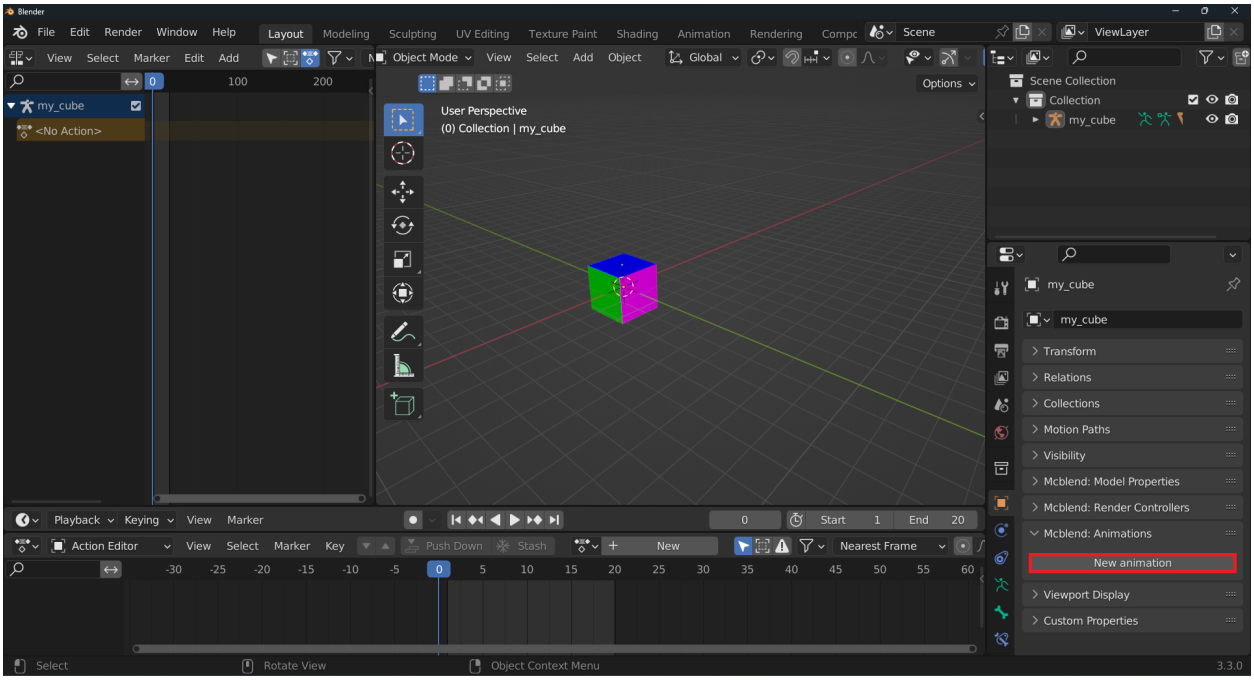
EIGHTEEN

CREATING ANIMATIONS FROM SCRATCH

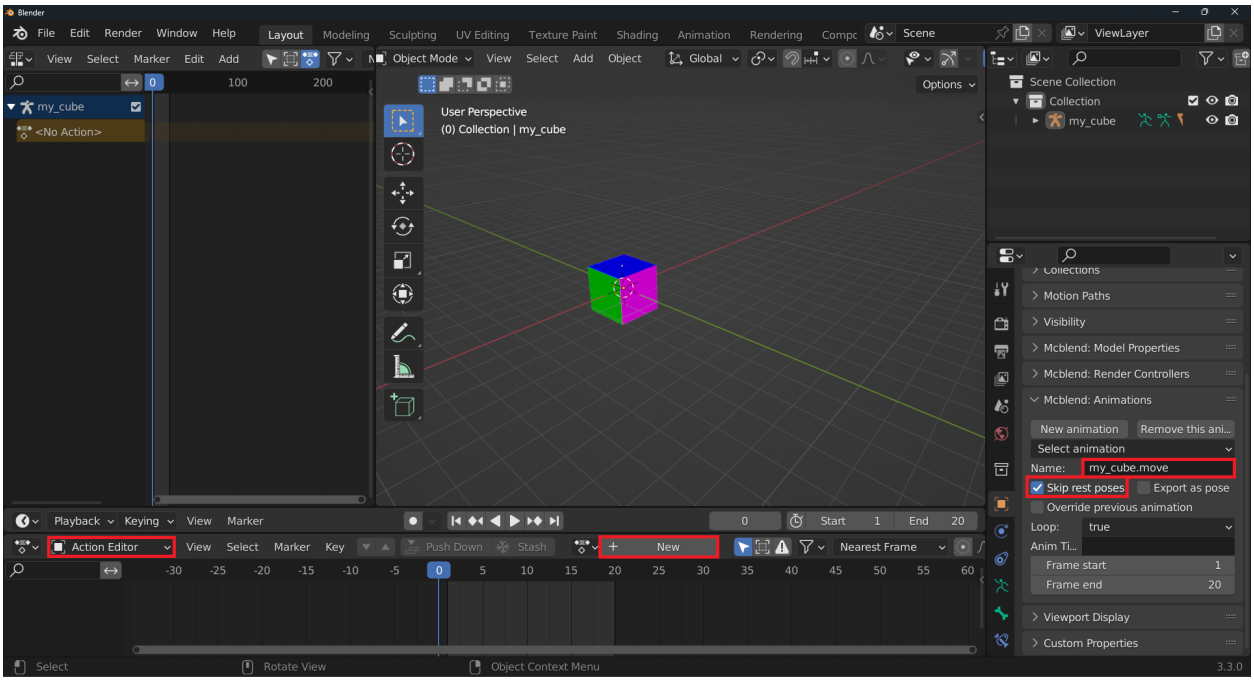
In this section, you will learn how to create animations in Mcblend from scratch. It is assumed that the reader already knows how to make and export models in Mcblend. We will start with a simple model that has a single bone and a single cube connected to it.

18.1 Creating movement animation

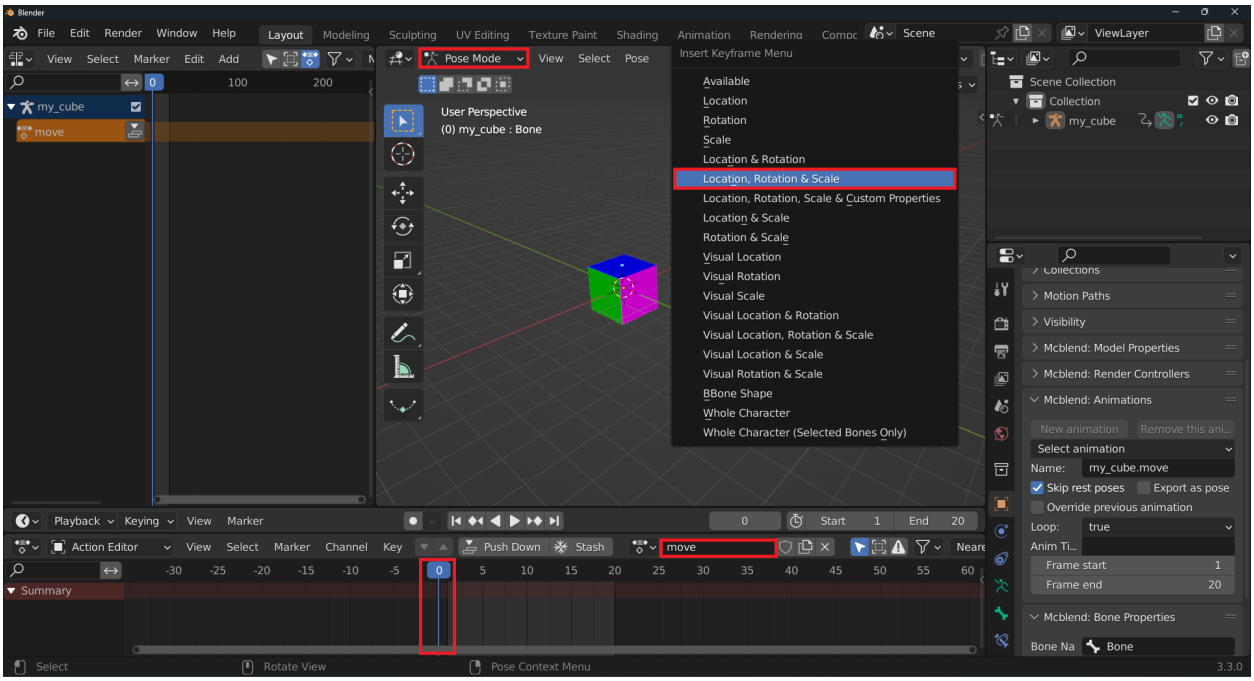
1. Select the armature
2. Go to the `Mcblend: Animations` panel under the `Object Properties` tab and click the `New Animation` button



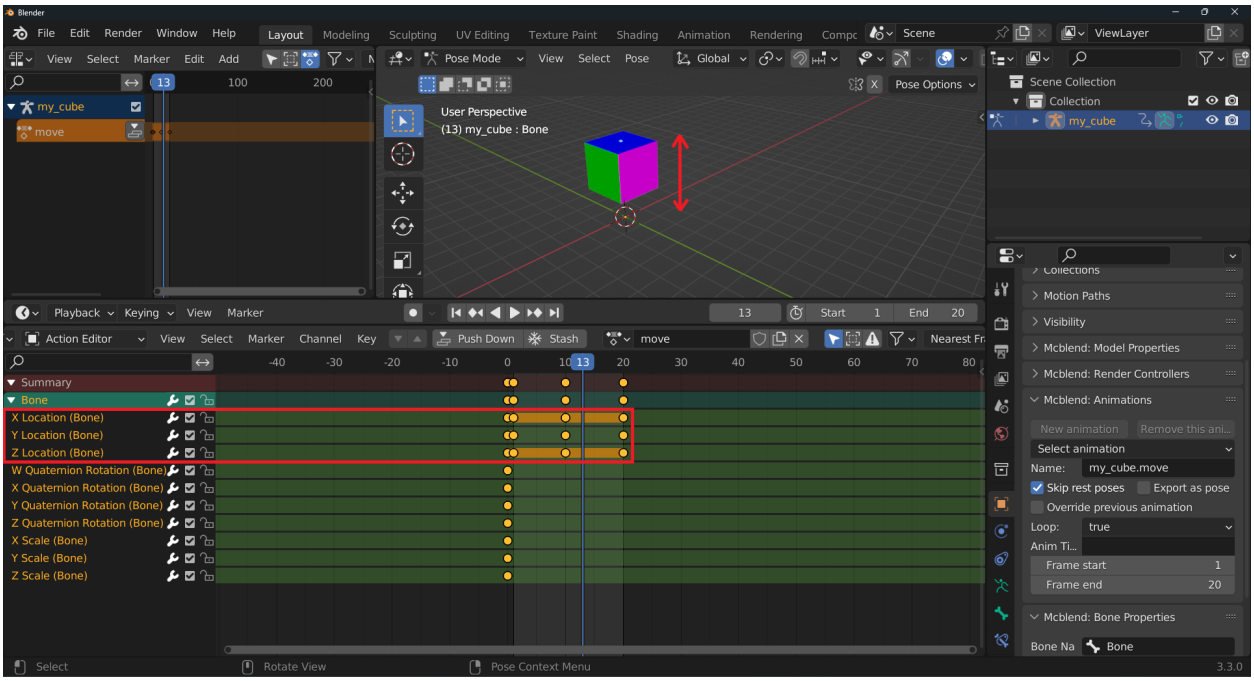
3. Rename the animation to `my_cube.movement`. It is a good practice to name the animations using the pattern: `<model_name>.<animation_name>`. In our case, the model is called `my_cube`.
4. Select the `Skip rest poses` checkbox (the configuration of the animation is explained later in the documentation)
5. In the action editor, press the `New` button to create a new action for the armature. You can rename the action to the same name as the animation, but it is not necessary. It will be easier to find the action in the action editor if you do so.



6. Add a keyframe to location, rotation, and scale at frame 0. This will be the rest pose of the animation. The keyframe should be added in Pose mode. You should add the keyframe to all bones in the armature. In this case, we only have one bone. Select all of the bones by pressing `A` while in `Pose mode` and hovering over the 3D viewport. Then, press `I` to add the keyframe and select the `Location, Rotation & Scale` option.



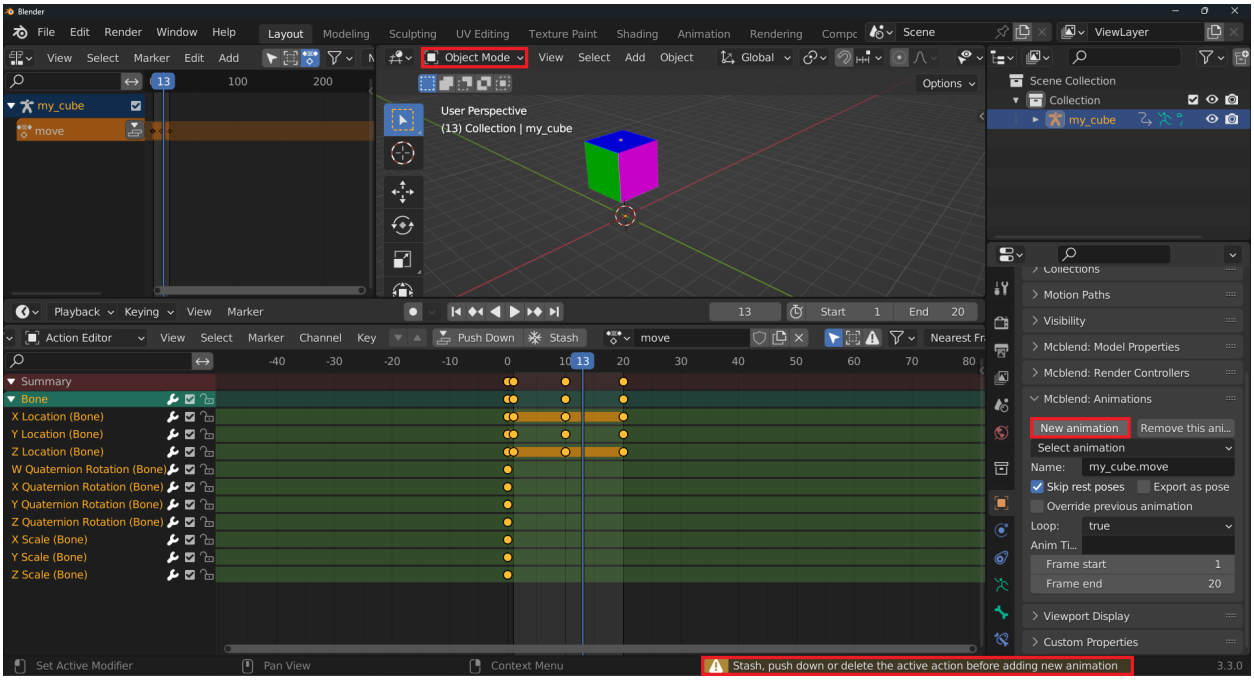
7. On the first frame, without moving the cube, add a keyframe to the location only.
8. Move the timeline and move the cube to the desired position, then add a keyframe to the location only.
9. Move the timeline to the last frame and reset the cube to the rest pose, then add a keyframe to the location only.
10. At this point, you should have an animation of a moving cube. Only the location of the cube should be animated.



18.2 Managing animations using the NLA editor

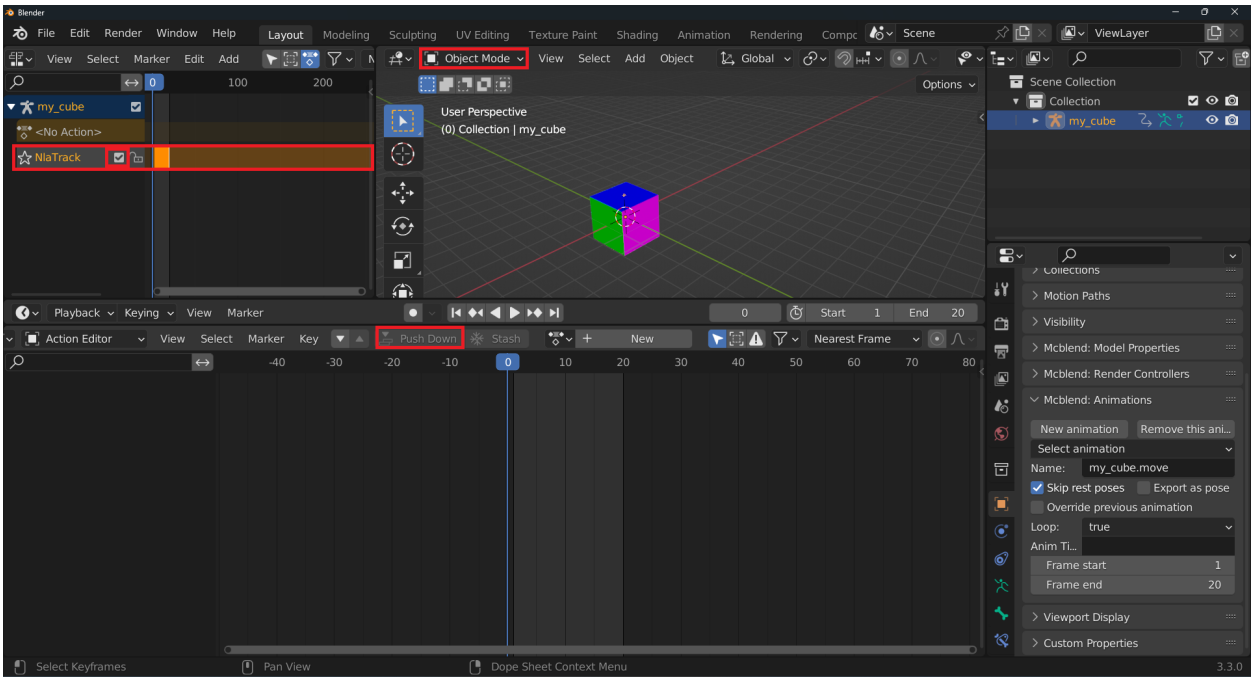
In this section, we will learn how to use the NLA editor to manage animations in Mcblend. This includes the ability to combine multiple actions into a single animation.

To create a new animation, return to Object mode and press the `New Animation` button. If you try to create a new animation while an active action is present, you will receive a warning to either "Stash or Push Down" the active action.

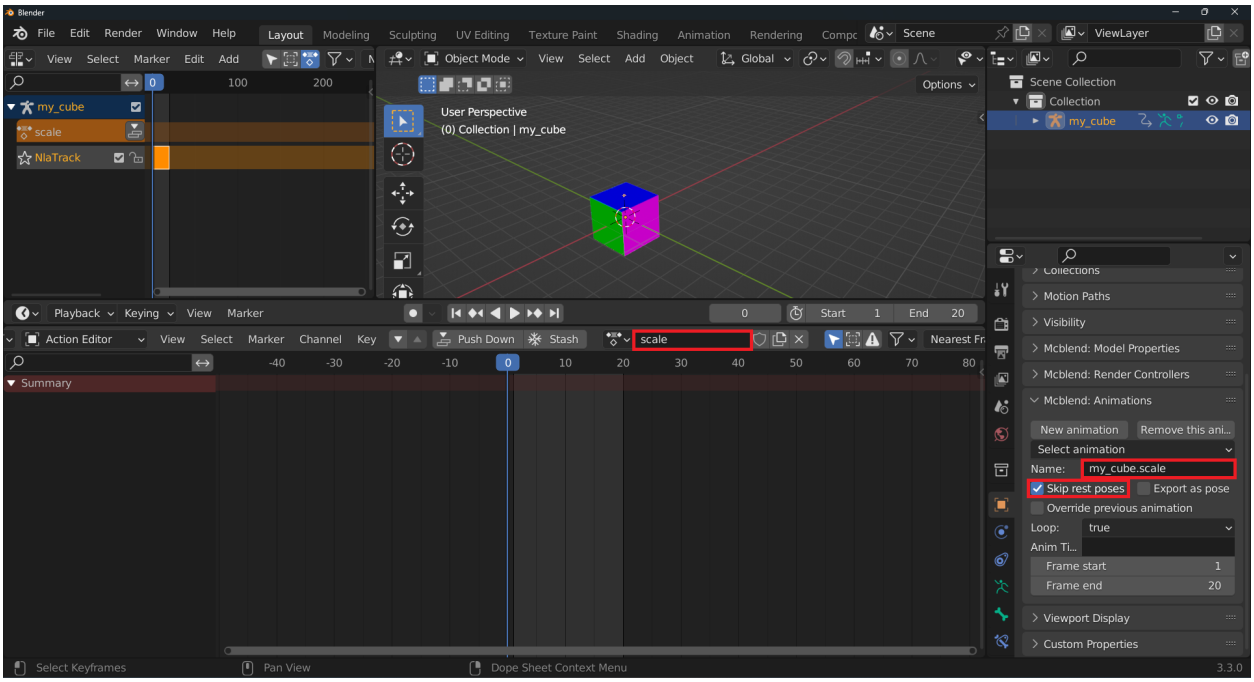


1. In the `Action Editor`, press the `Push Down` button to add the animation to the NLA track and activate it. The

Stash button does not activate the animation.

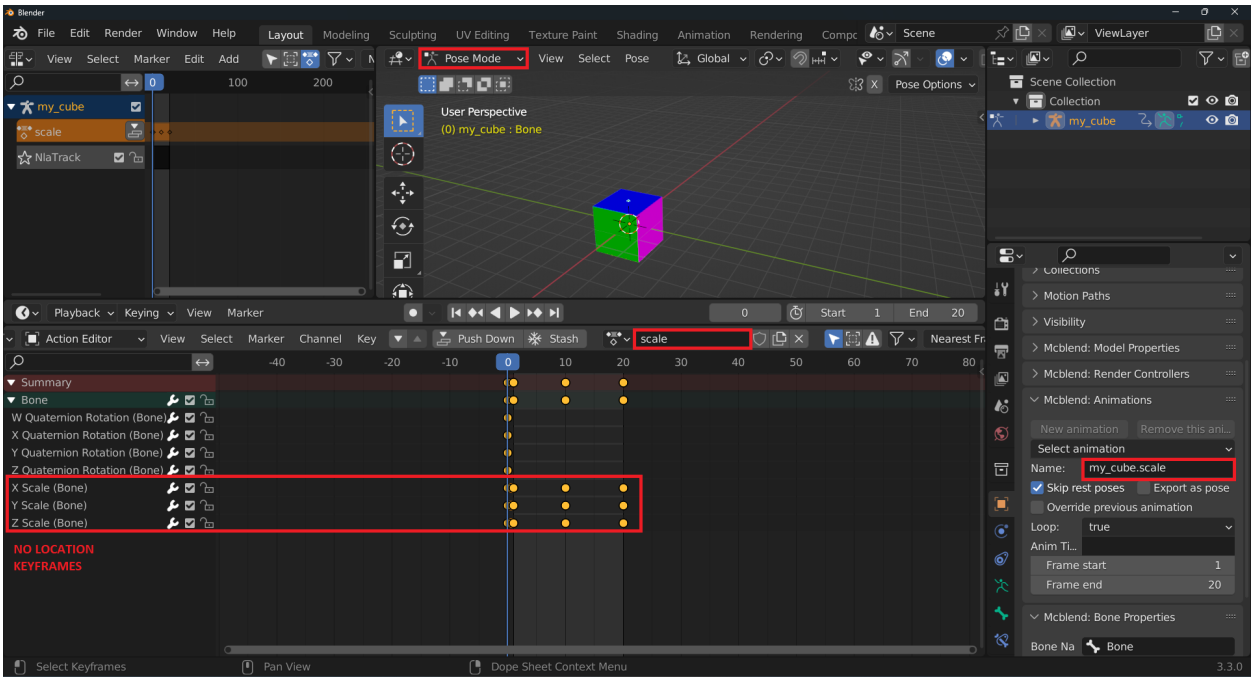


2. To create a new animation, repeat the steps from the previous section, naming it `my_cube.scale` and creating a new action for it. Note that the action from the previous animation is still active in the NLA editor. You can disable it to prevent it from affecting the new animation.



3. Add keyframes as in the previous section, including a rest pose at keyframe 0. This time, do not add a keyframe to the location at frame 0 (we want to use the location animation from the previous animation). Instead, animate the scale of the cube. Remember to animate bones in Pose mode, not the armature.

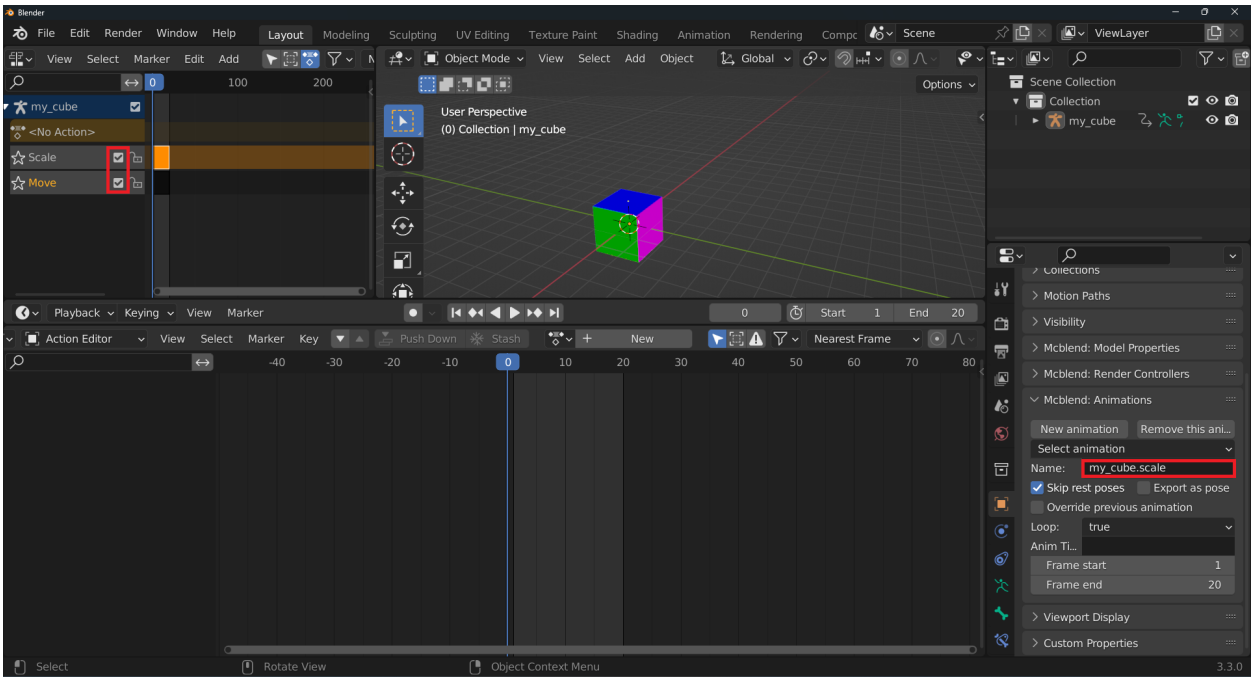
4. If configured correctly, the currently open action should change the scale of the cube while the action from the NLA track makes it move.

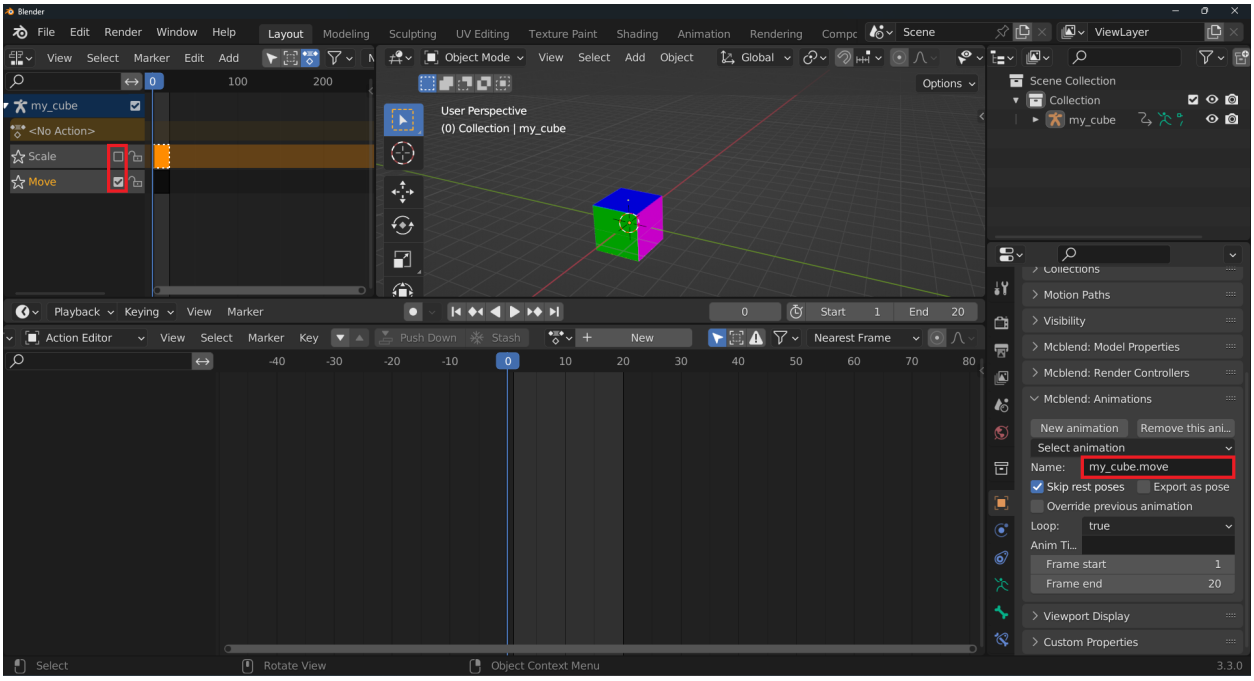


5. Push down the animation to add it to the NLA track. You should now have two active NLA tracks. You can disable an animation from the NLA track by pressing the checkbox next to the track name.

6. You can rename the tracks for easier identification, but be aware that Mcblend remembers tracks by their names. Renaming a track used by an animation will unlink it from the animations that use it.

7. To switch between animations in the `Mcblend: Animations` panel, use the `Select Animation` dropdown list while in `Object Mode`. Keep in mind that switching the animation also activates different NLA tracks in the NLA editor. In this example, we want the `Move` animation to only affect the location of the cube, while the `Scale` animation should influence both movement and scale (using both NLA tracks). To accomplish this, the `Move` animation should have only the `Move` track active, and the `Scale` animation should have both the `Move` and `Scale` tracks active.

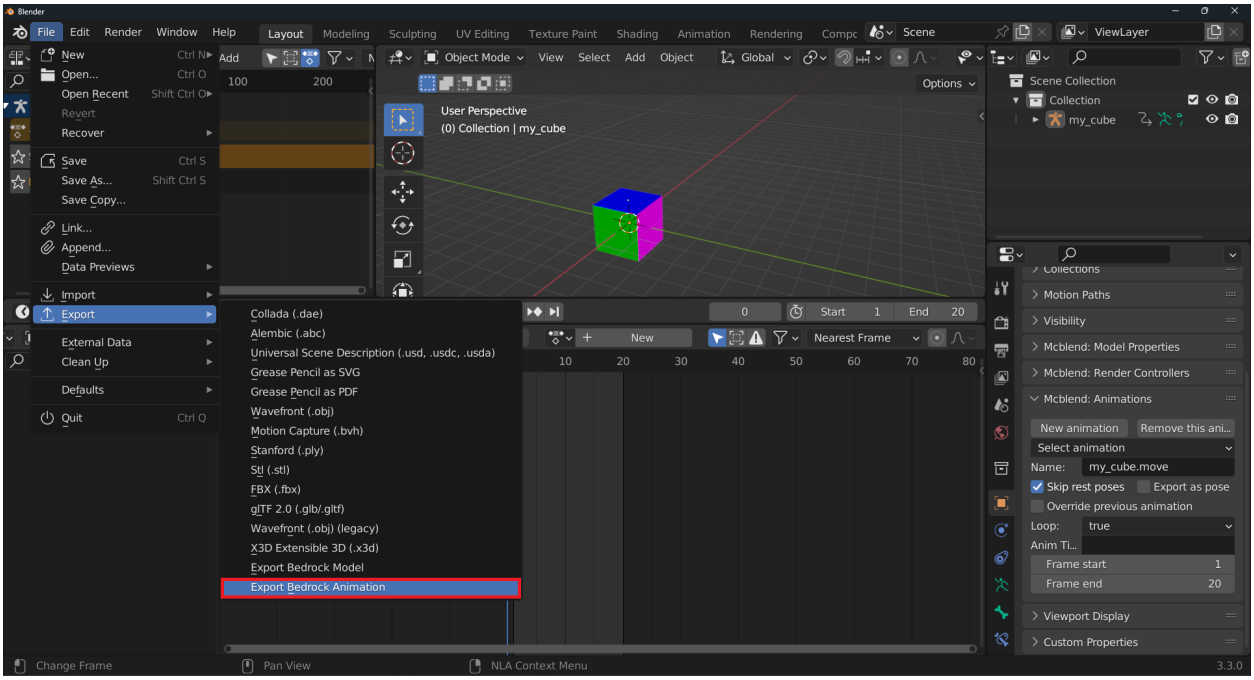

my_cube
<No Action>
Scale
Move
User Perspective
(0) Collection | my_cube
Scene Collection
Collection
my_cube
Motion Paths
Visibility
Mcblend: Model Properties
Mcblend: Render Controllers
Mcblend: Animations
New animation
Remove this ani...
Select animation
Name:
my_cube.scale
Skip rest poses
Export as pose
Override previous animation
Loop:
true
Frame start
1
Frame end
20
Viewport Display
Custom Properties


my_cube
<No Action>
Scale
Move
User Perspective
(0) Collection | my_cube
Scene Collection
Collection
my_cube
Motion Paths
Visibility
Mcblend: Model Properties
Mcblend: Render Controllers
Mcblend: Animations
New animation
Remove this ani...
Select animation
Name:
my_cube.move
Skip rest poses
Export as pose
Override previous animation
Loop:
true
Frame start
1
Frame end
20
Viewport Display
Custom Properties

18.3 Exporting animations

To export animations in Mcblend, follow these steps:

1. Select the desired animation in the `Mcblend: Animations panel`, found under the `Object Properties` tab.
2. Choose `File > Export > Export Bedrock Animation` from the menu.
3. Select the location where you would like to save the exported animation file.
4. Repeat these steps to export any additional animations. Note that exporting multiple animations to the same file will not overwrite the previous animation - both will be included in the same file.



CHAPTER

NINETEEN

EXPORTING ANIMATIONS

This section explains how to export animations from Mcblend and mentions some additional settings that can be configured before exporting.

To export an animation, you first need to create one using the `New animation` button in the `Object Properties` panel under the `Mcblend: Animations` tab. This tab is only visible when an armature is selected. If you want to remove an animation, use the `Remove this animation` button.

Before exporting the animation, you can configure some additional settings in the `Object Properties` panel. The `Skip rest poses` checkbox allows you to create more optimized animations by skipping unnecessary keyframes that don't affect the pose of the model. The `Export as pose` option changes the animation to a single pose, creating a looped animation with a single frame.

The `Override previous animation`, `Loop`, and `Anim Time update` settings are properties of Minecraft animations and are directly exported to the file. They have no effect on Mcblend. The `Frame start` and `Frame end` settings determine the range of frames that will be exported to the animation.

Exporting the animation is similar to exporting a model, as explained in the *"Creating animations from scratch"* documentation page. Simply go to `File > Export > Export Bedrock Animation` and select the export path.

CHAPTER

TWENTY

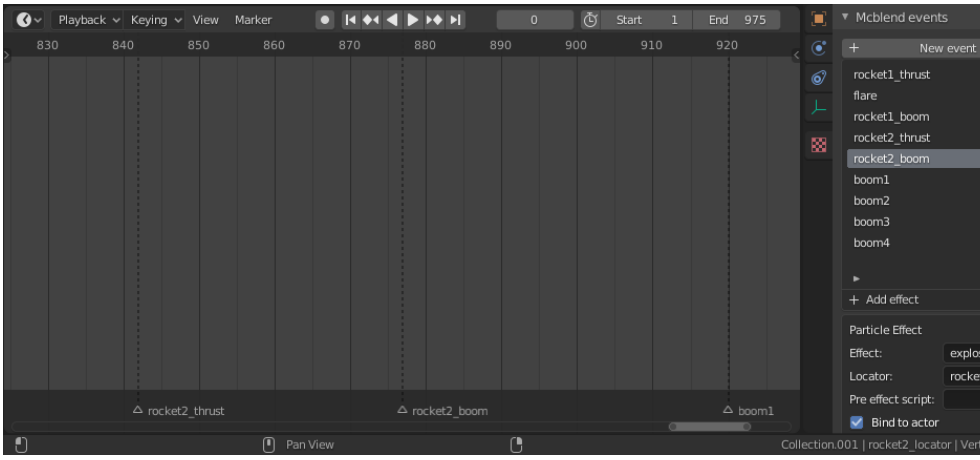
ANIMATING SOUND AND PARTICLE EFFECTS

The sound effects and particle effects are animated with the use of events. You can define events in the `Mcblend: Animation Events` menu in `Scene Properties`. One event can contain multiple particle and sound effects. The effects are not visible in the animation preview in Blender. They only add some information to the exported animation.

Events can be attached to the animation by adding markers in the timeline with the name of the event. You can trigger the same event multiple times in the animation by adding multiple timeline markers with the same name.

Timeline markers that do not have a matching event name are ignored when exporting the animation and serve the same purpose as any other timeline marker in Blender.

A timeline with timeline markers for events:



CHAPTER

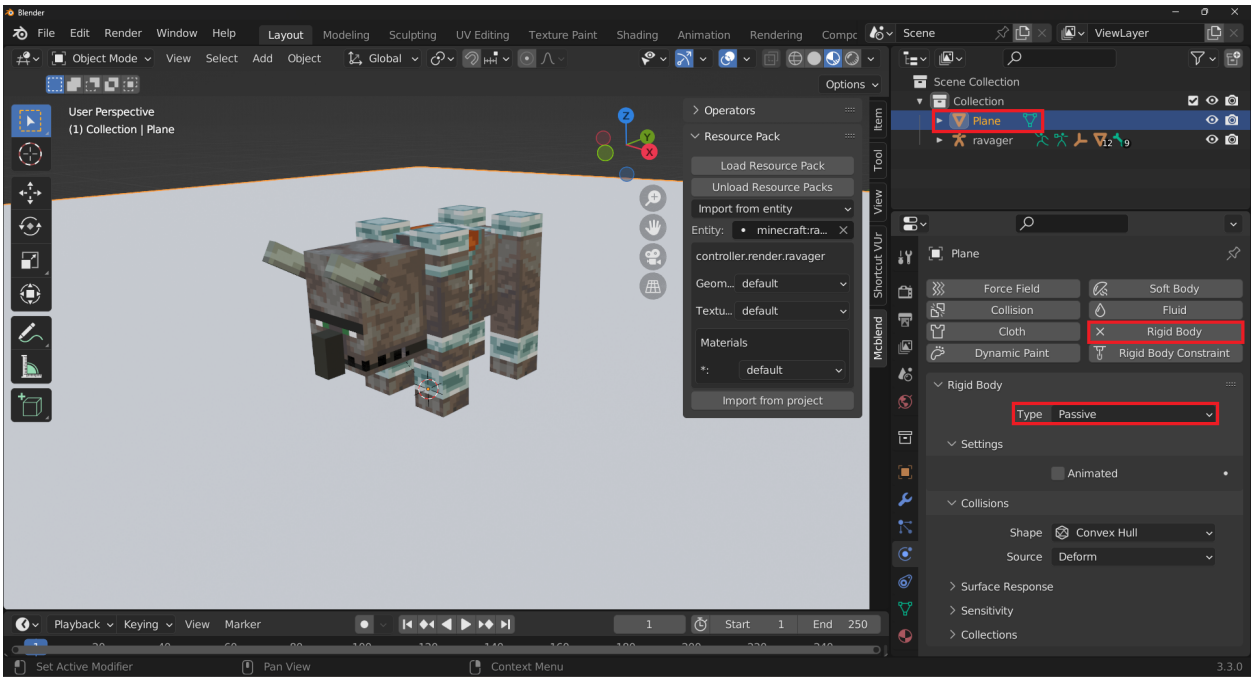
TWENTYONE

PHYSICS SIMULATION

This tutorial explains how to use Blender tools to easily set up physics simulations using rigid bodies. It is assumed that the reader knows how to *import models from resource packs* and export and manage animations. These actions are explained in different parts of the documentation.

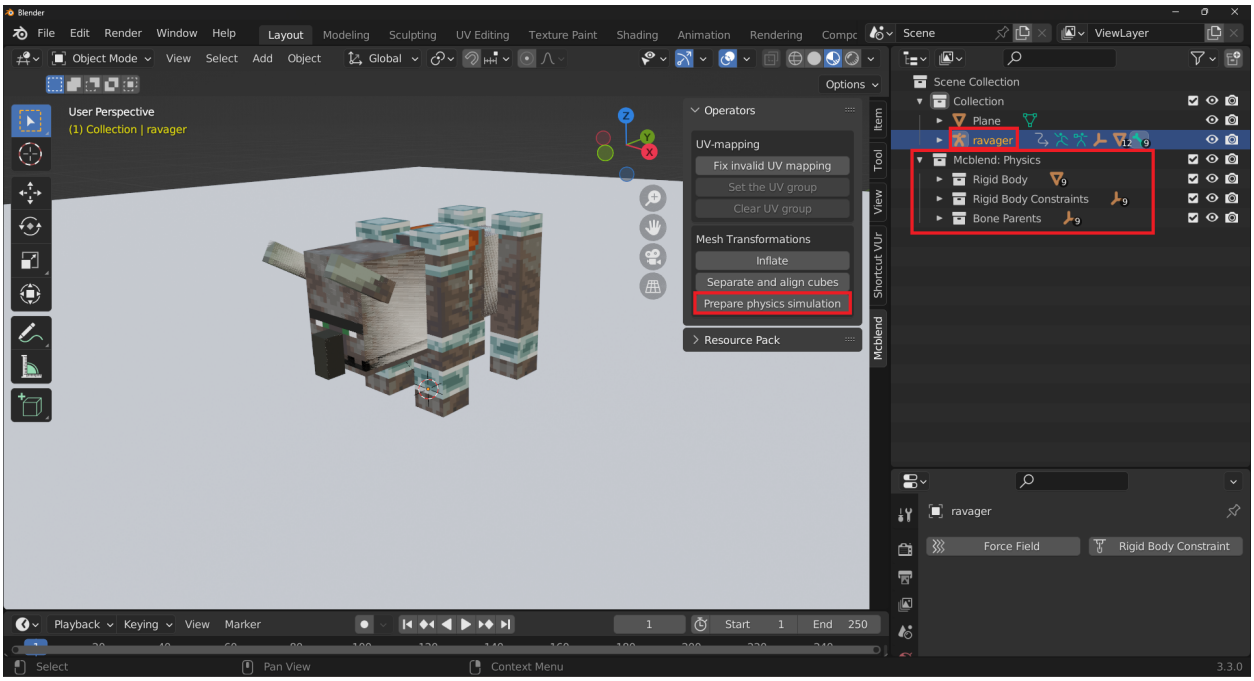
The default Minecraft resource pack can be downloaded from Mojang's Bedrock Samples: https://github.com/Mojang/bedrock-samples/releases/latest.

1. Import `minecraft:ravager` model. We pick ravager because it has a simple geometry and unlike some othe mobs it doesn't use animations to move it to a default pose which means that our animation doesn't have to recreate that.
2. Add a plane, scale it up 30 times, select it and open `Physics Simulation` and press the `Rigid Body` button, change the `Type` to `Passive`. This plane will act as a floor for the model parts to interact with.



3. To prepare a model for physics simulation, select the armature of the model and in the `Mcblend` sidebar, press the `Prepare physics simulation` button.

This operation adds three new collections to the outliner: `Rigid Body`, `Rigid Body Constraints`, and `Bone Parents`. It also configures the model to be ready for physics simulation. Any frames after the one selected when the `Prepare physics simulation` button is pressed are simulated, while the frames before that point can be animated manually. The simulation button also adds keyframes to the animation that enable the simulation one frame after the frame active at the moment of pressing the button.



Note: The explanation of the collections created by the `Prepare physics simulation` button:

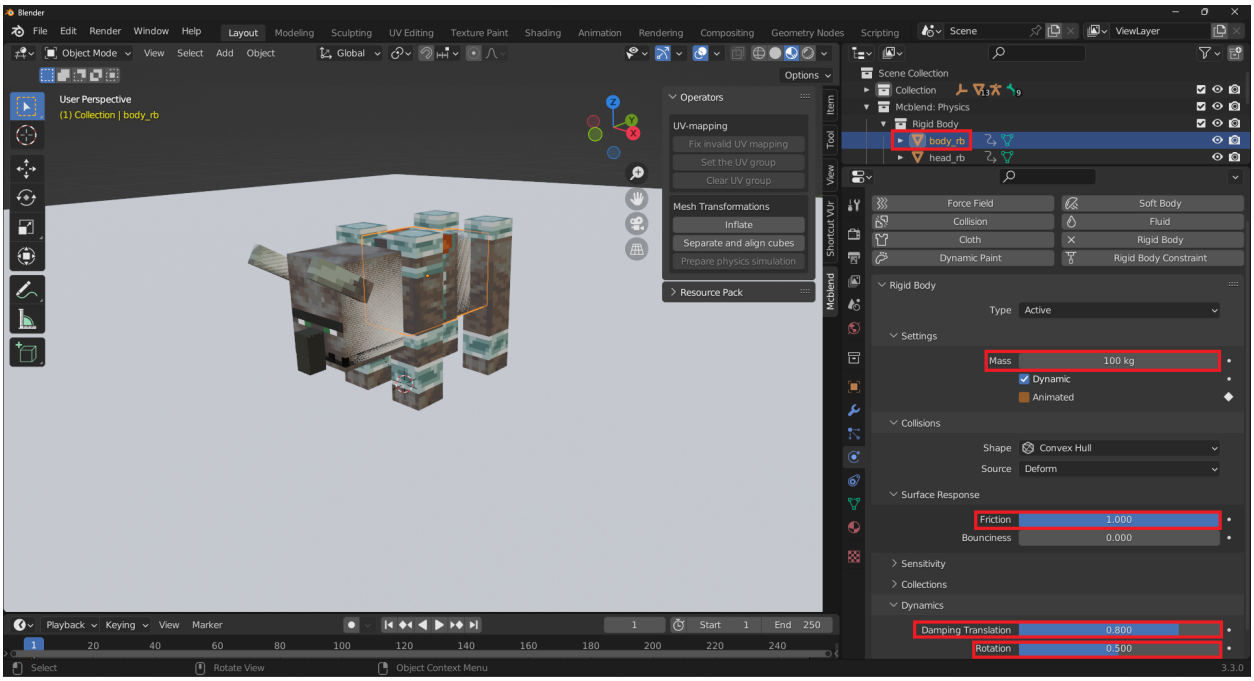
- The `Rigid Body` collection contains meshes with the same shapes and sizes as the parts of the model. These objects are simulated and during the simulation their movement is copied to the corresponding bones of the model.
- The `Rigid Body Constraints` collection contains empties that define the connection points between the bones. They're used to define the bone structure of the model. The constraints are set to `Fixed` by default which means that the bones can't move in relation to each other.
- The `Bone Parents` are empties that correspond to the bones of the model. The bones of the model use the `Copy Transforms` constraint to copy their transformations during the simulation. The `Bone Parents` empties copy the transformations of the rigid bodies during the simulation.

4. The properties of the rigid bodies and rigid body constraints usually need to be adjusted for a good simulation.

The objects from the `Rigid Body Constraints` group are only useful when your model has more complex bone structure. In the ravager model everything except `mouth` is a root bone so most of the constraints don't do anything. In some cases you would change the default `Fixed` type of the connection to something different to allow the movmement of the bone but we won't do that because the animation looks better without it.

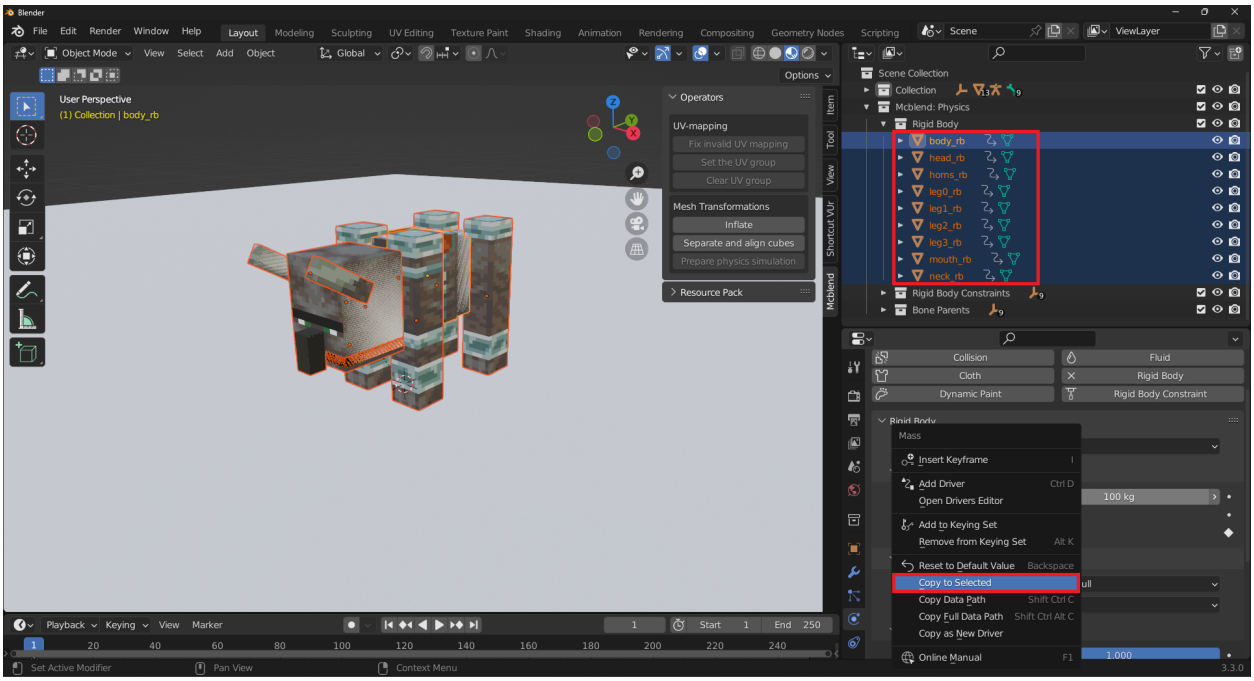
We need to adjust the objects from the `Rigid Body` group. They should be heavier and have more friction. Set their:

- `Friction` to 1
- `Weight` to 100kg
- `Dumping Translation` to 0.8
- `Rotation` to 0.5



Warning: Blender sometimes changes the names of the properties of the rigid bodies with updates. It appears that the names are changed more often than other parts of the UI, so if you're using a different version of Blender than the one used to write this tutorial (3.3), the names of the properties may be different.

Use `Copy to selected` to copy all of these properties to every object from the `Rigid Body` group.

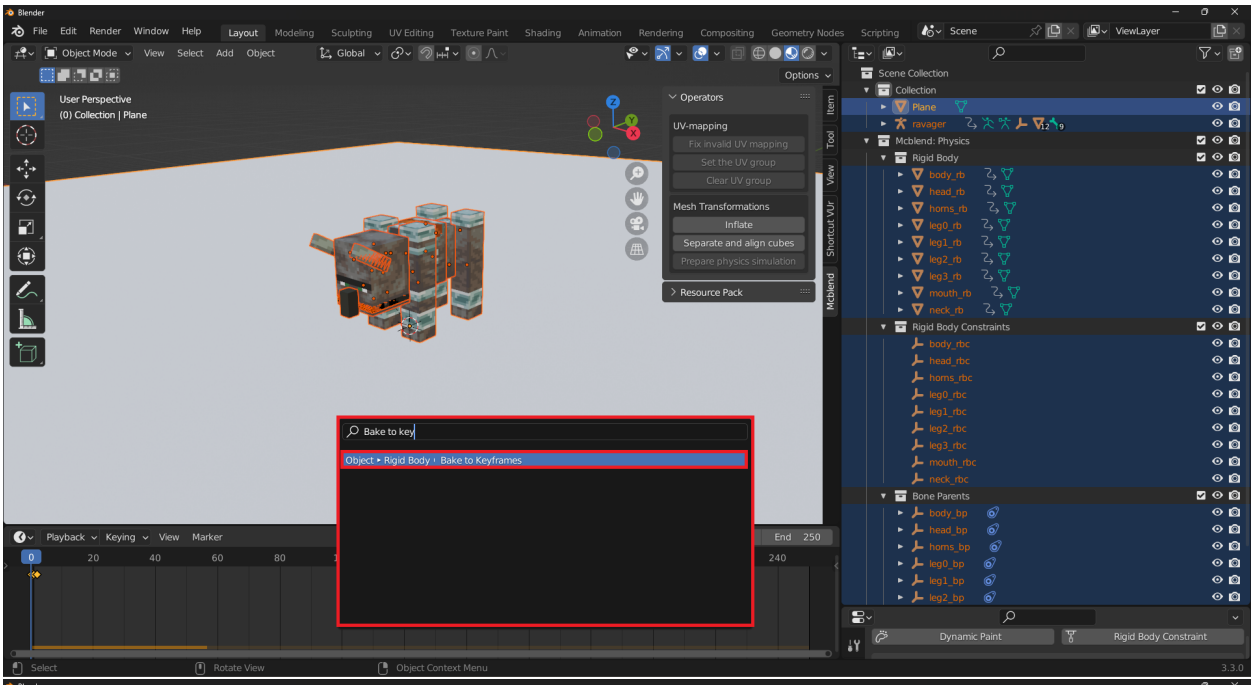


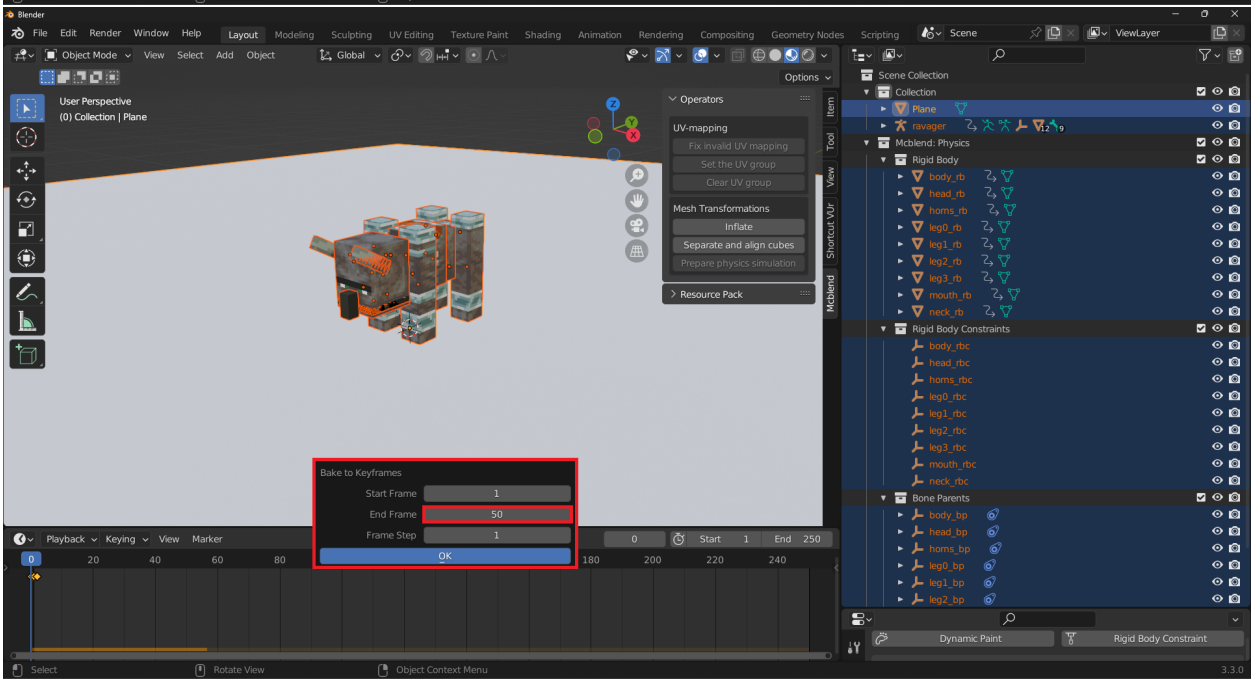
The `Bone Parents` group never needs to be adjusted.

5. Starting the simulation by pressing `space` will cause the model to fall apart and explode, due to the overlapping

parts. The limbs and head bones are not parented to the body bone, allowing them to fly away. The previously configured settings of increased friction and weight prevent the parts of the model from sliding too much on the ground.

6. To bake the animation, first select everything in the scene (by presing `A`), then press `F3`. In the search field, type `Object > Rigid Body > Bake to Keyframes`. The full simulation will take around 50 frames, so set the range to 50 frames.

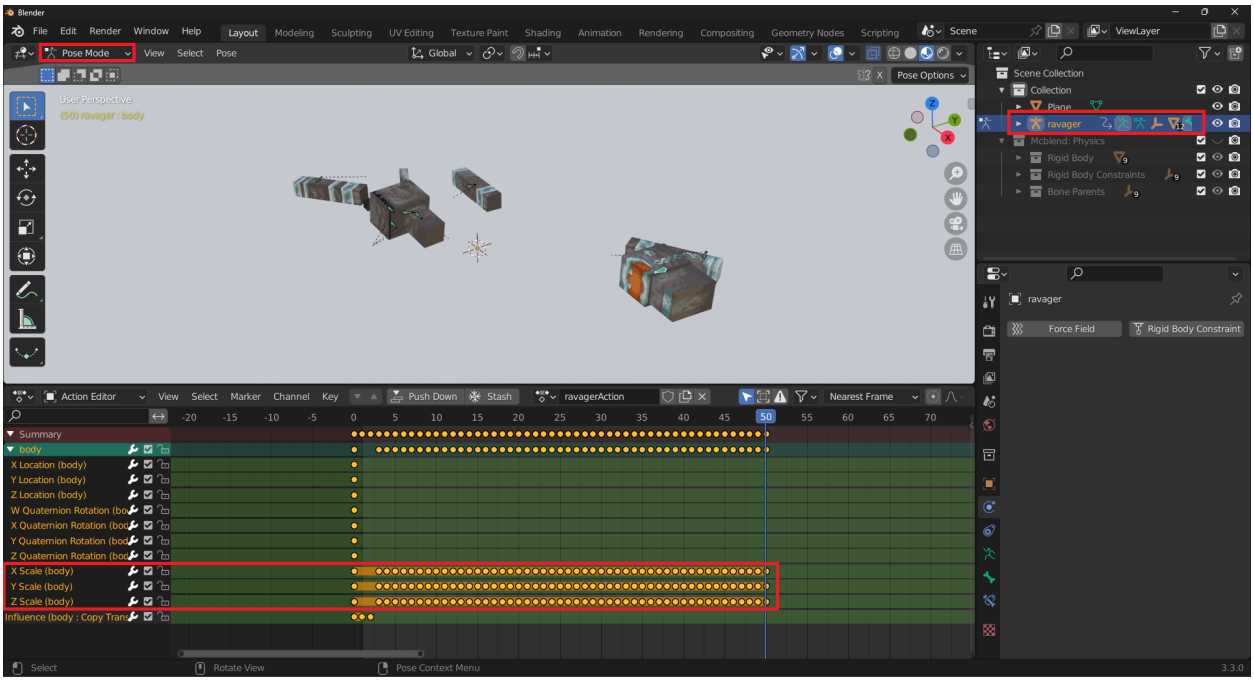




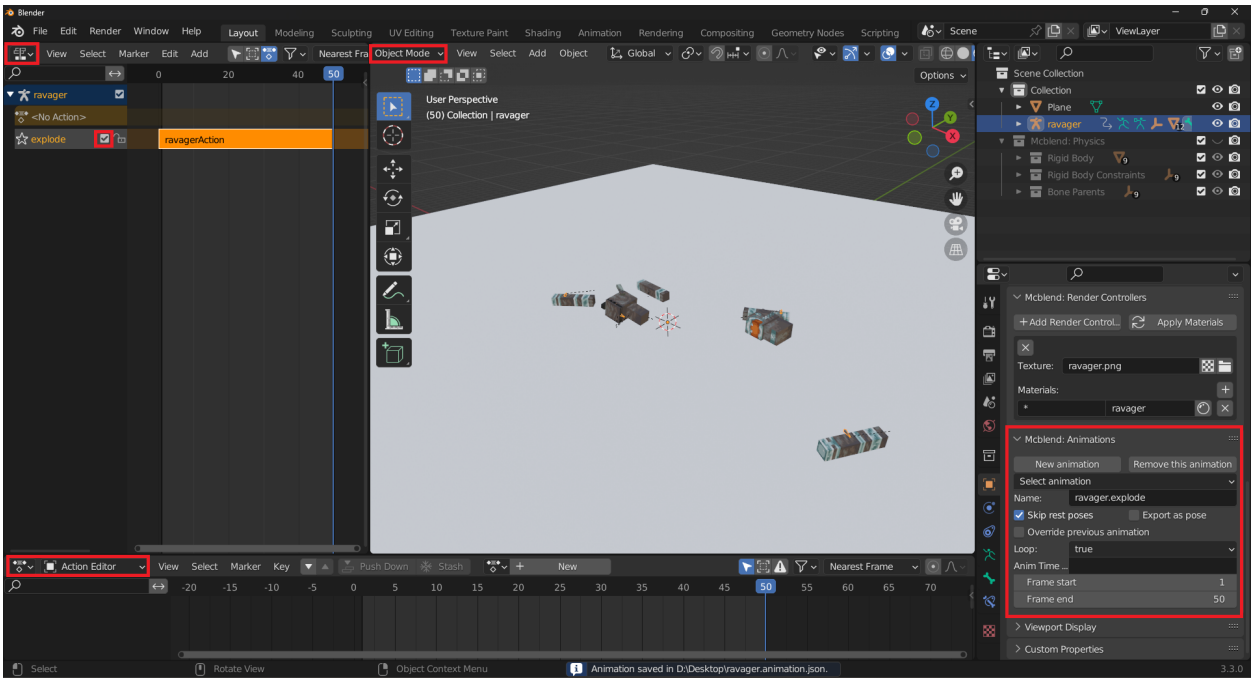
7. The keyframes added by the `Bake to Keyframes` operation are not sufficient to properly export the animation. Mcblend requires keyframes to be added to the armature of the model, but the keyframes from the simulation are added to the rigid bodies.

To successfully export the animation, you will need to manually add keyframes to the armature of the model. The movements of the simulation are complex, so you will need to add one keyframe for each frame of the simulation, approximately 50 frames in total. Mcblend does not currently have an automatic method for adding these keyframes, so you will need to do it manually. It doesn't matter what kind of keyframe you add - Mcblend uses the keyframes as points in the timeline that mark an important event in the animation, and then analyzes the pose of the model to compare it to the previous point in the animation.

We can simply add the `Scale` keyframes to the root bone. It doesn't matter that the scale is the same on every frame, the important thing is that the keyframes are there.



8. *Export the animation* using the knowledge from previous tutorials.



CHAPTER

TWENTYTWO

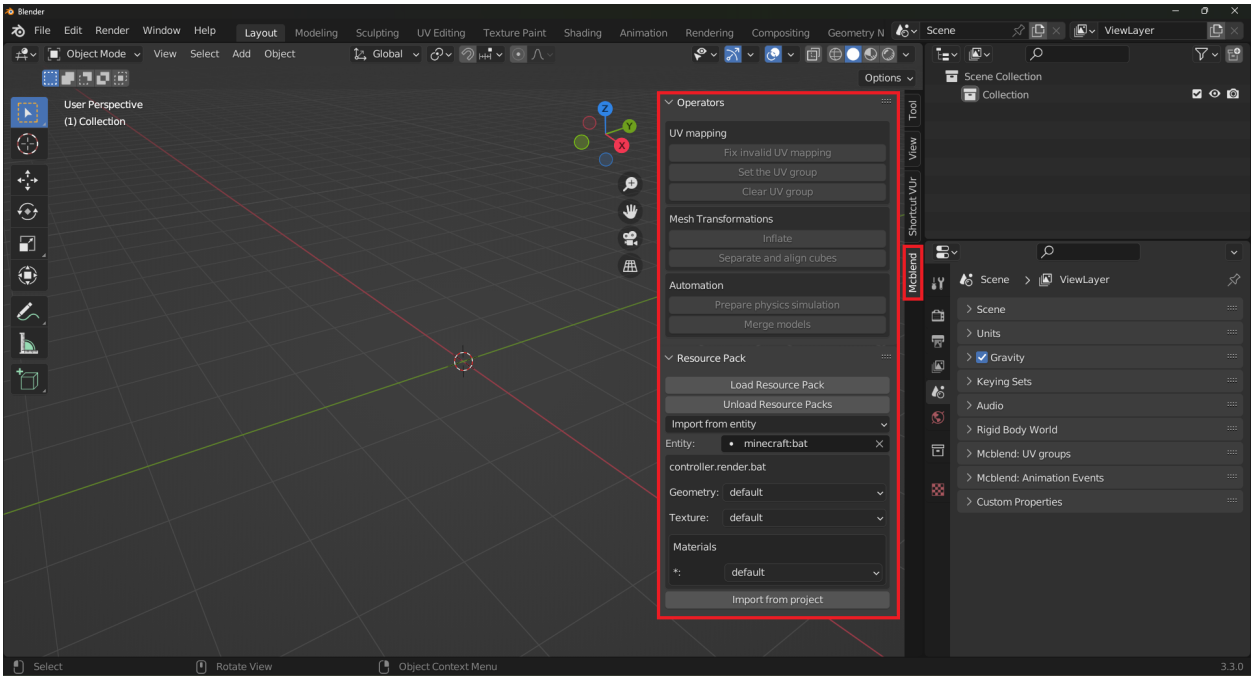
OVERVIEW

The GUI section of the documentation for the Mcblend plugin provides a list and explanation of all additional graphical user interface elements added by the plugin. This section serves as a reference, providing information on the various GUI elements added to Blender by Mcblend. It is not meant to be a comprehensive guide on how to use the plugin, but rather a resource to refer to while working through user guides and tutorials to understand the buttons, tabs, and other GUI elements relevant to the tasks being performed.

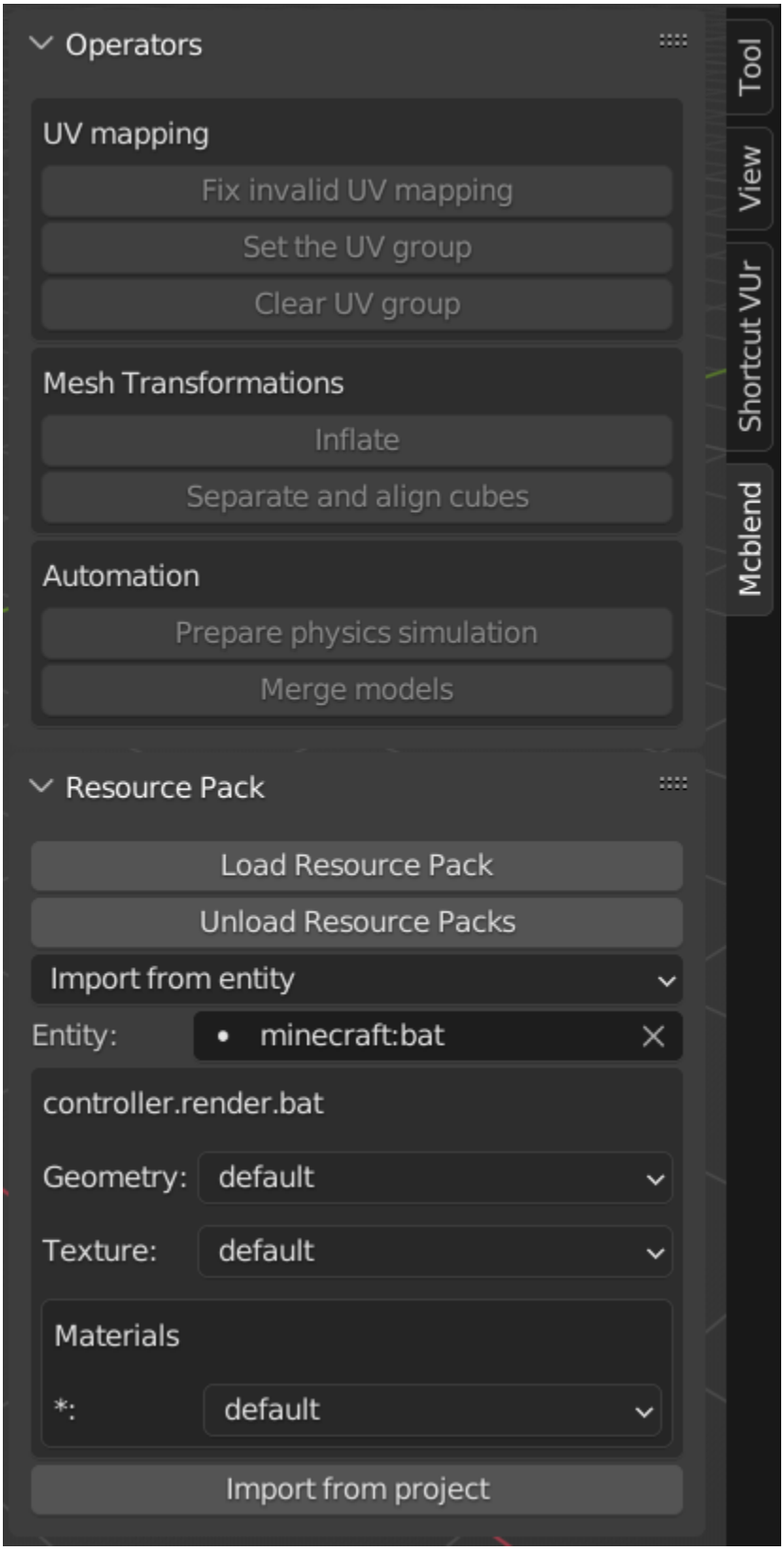
CHAPTER

TWENTYTHREE

3D VIEWPORT SIDEBAR



The 3D viewport sidebar includes a new `Mcblend` section, which has two panels. One panel contains buttons with commonly used functions, while the other panel is for resource pack integration. You can open the sidebar by pressing `N` while hovering over the 3D viewport if you're using the default shourtcuts.


Operators
UV mapping
Fix invalid UV mapping
Set the UV group
Clear UV group
Mesh Transformations
Inflate
Separate and align cubes
Automation
Prepare physics simulation
Merge models
Resource Pack
Load Resource Pack
Unload Resource Packs
Import from entity
Entity:
minecraft:bat
controller.render.bat
Geometry:
default
Texture:
default
Materials
*:
default
Import from project
Tool
View
Shortcut VUr
Mcblend

23.1 Operators

23.1.1 UV mapping panel

- `Fix invalid UV mapping` - Operator used to fix invalid UV mapping of the model's cuboids. All faces of the cuboids in the Minecraft model must be rectangular and have a certain rotation. This operator ensures that these conditions are true.
- `Set the UV group` - Adds the selected objects to one of the existing UV groups.
- `Clear UV group` - It removes selected objects from UV groups.

23.1.2 Mesh Transformations panel

- `Inflate` - Inflates the selected object using Minecraft's inflate property. Running this operator opens a panel in the bottom left corner of the 3D viewport. You can use this panel to adjust the `inflate` value.
- `Separate and align cubes` - Detects cubes grouped in a single mesh and splits them into separate objects. Unlike the vanilla Blender operator (`Mesh -> Separate`), the `Separate and align cubes` operator from Mcblend is designed for working with cuboids and can detect their rotations.

23.2 Automation panel

- `Prepare physics simulation` - Automatically creates objects, which can be used for physics simulation.
- `Merge models` - Merges simple models, creates a common texture file for them and moves their UV coordinates to the correct position on the new texture.

23.3 Resource Pack panel

The Resource Pack panel in Mcblend allows you to connect a resource pack to your project. Once you open Blender, you will see a single button called `Load Resource Pack`. This button loads information about the entities and attachable items that are available for import.

After loading the resource pack, additional GUI elements will appear:

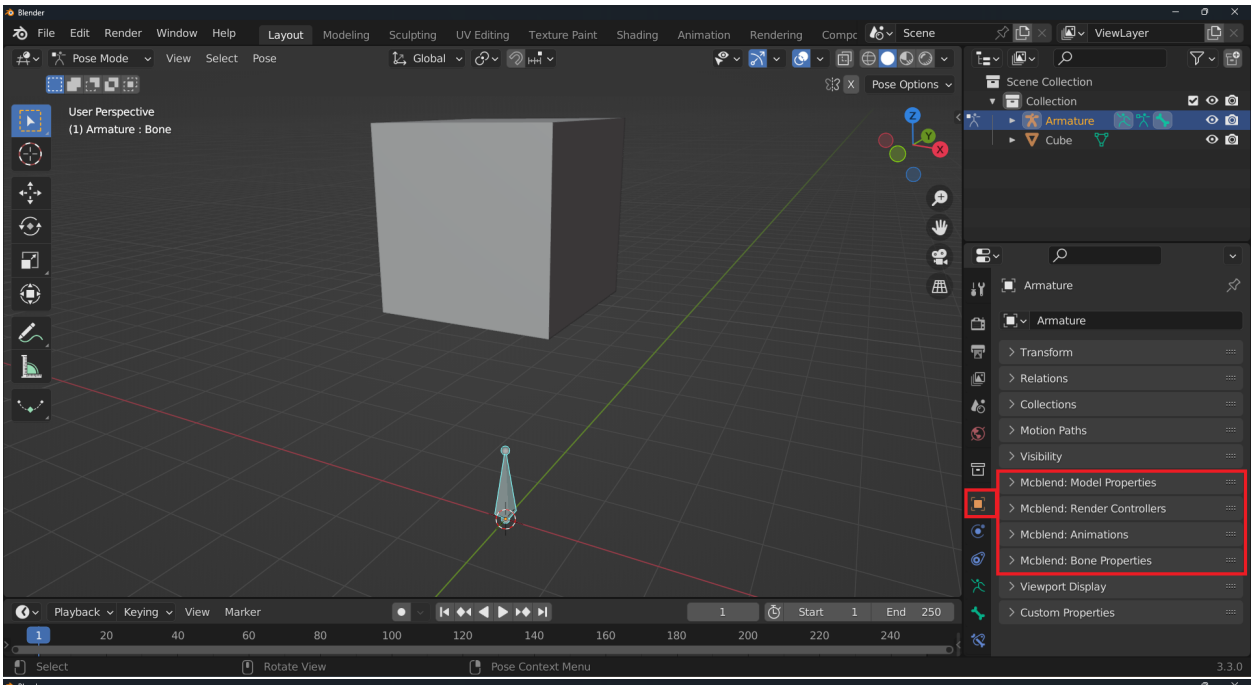
- The top dropdown list allows you to select whether you want to import an entity or a model of an attachable item.
- Below the dropdown list, there is a text box that allows you to select the item or entity you want to import.
- At the bottom of the panel, additional options will appear, allowing you to select textures, materials, and models. These options depend on the properties of the render controllers of the thing you want to import.
- The `Import from project` button allows you to import selected items to the project.

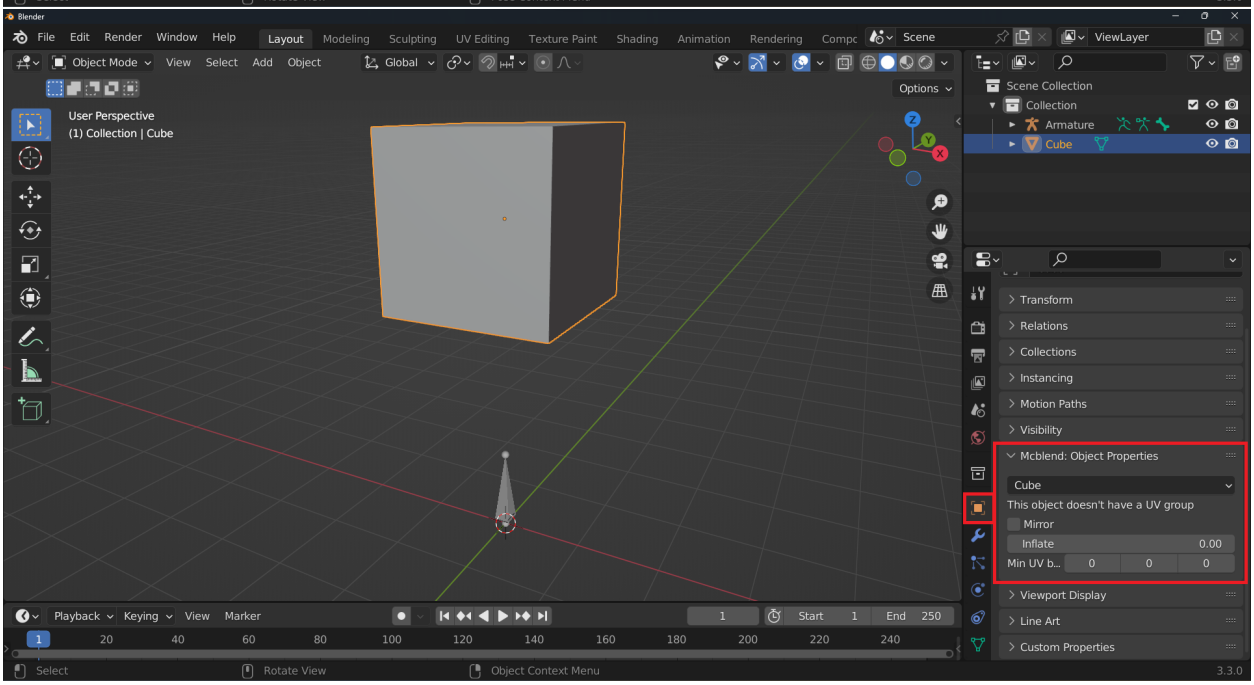
You can use the `Unload Resource Pack` button to unload the currently loaded resource pack. Note that the data about the resource pack is not saved in the project file, so you will have to load it every time you open the project.

CHAPTER

TWENTYFOUR

OBJECT PROPERTIES

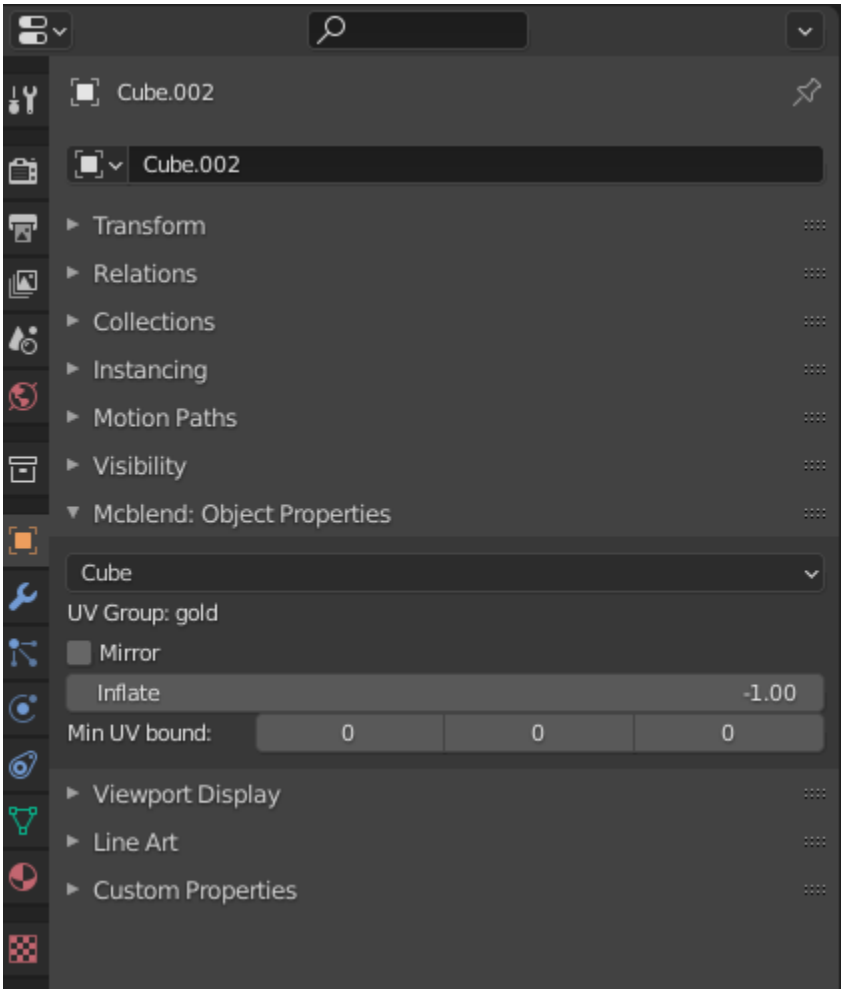




24.1 Object Properties (mesh)

This is the panel visible when opening the `Object Properties` when the active object is a mesh.

24.1.1 Mcblend: Object properties

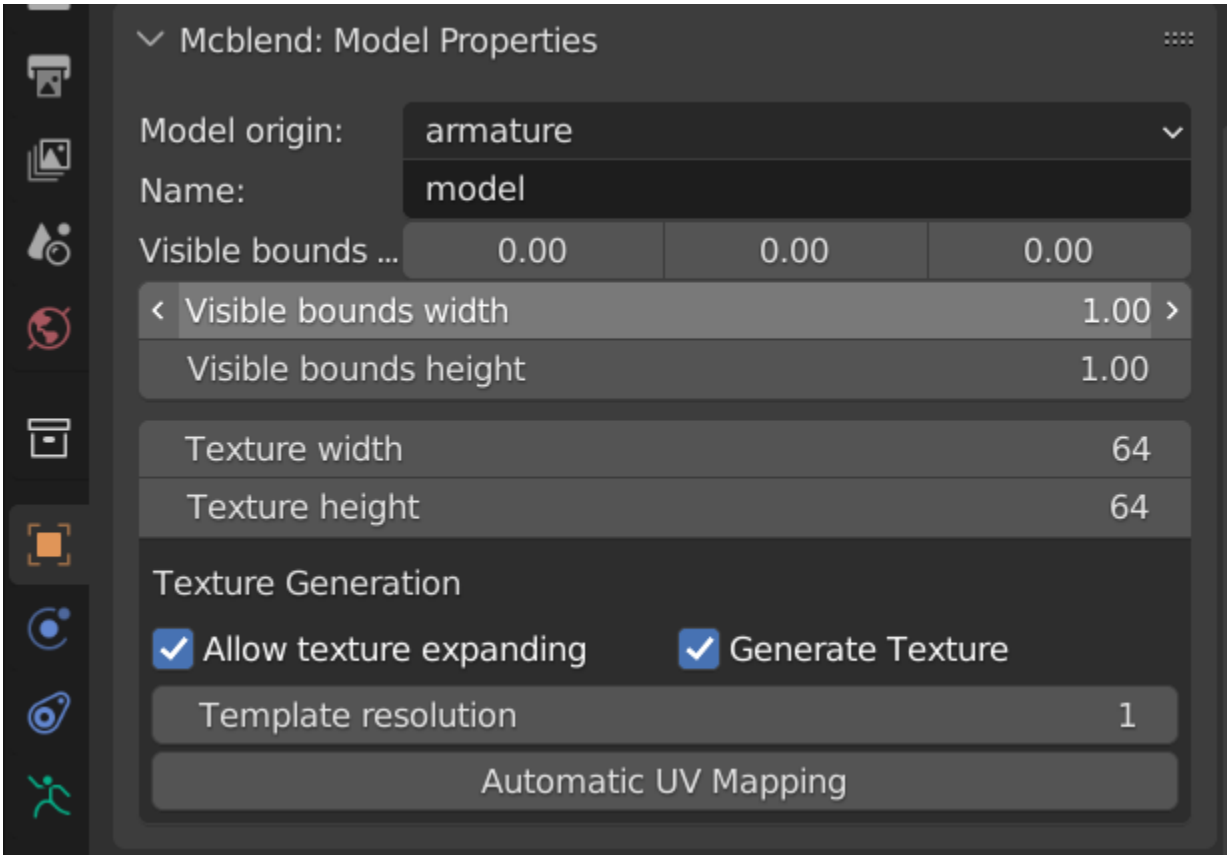


- `Mesh type` - a dropdown list that lets you select between `Cube` or `Poly mesh`. Decides if the object should be exported as a cube or polymesh.
- `UV Group` (textfield) - displays the UV group name of the selected object.
- `Mirror` - the mirror property is used only during the UV mapping. It affects the mapping of the object's faces in the same way as Minecraft's mirror property of a cube.
- `Inflate` - stores the Minecraft's inflate property value of the cube.
- `Min UV bound` - this property is used during automatic UV mapping. It defines the minimal space on the texture used for a cube. If a cube has width, height or depth lower than one unit of length, this property can be used to make sure that every face will get some space on the texture.

24.2 Object properties (armature)

This is the panel visible when opening the `Object Properties` when the active object is an armature.

24.2.1 Mcblend: Model properties

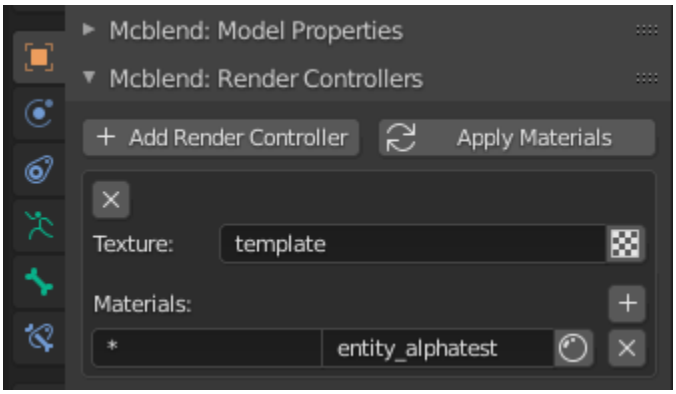


This panel has some of the basic properties of the Minecraft model like the width and height of visible bounds (they have the same names as in the Minecraft model) and some properties used for texture generation.

- `Model origin` - allows you to select whether the transformations of the bones and cubes in the exported model should be relative to the armature of the model or to the world origin.
- `Name` - the name of the model (excluding the `geometry.` prefix).
- `Visible bounds width` - the width of the visible bounds of the model.
- `Visible bounds height` - the height of the visible bounds of the model.
- `Visible bounds offset` - the offset of the visible bounds of the model.
- `Texture width` - the width of the texture used for the model.
- `Texture height` - the height of the texture used for the model.
- `Allow texture expanding` - allows changing the texture width and height during automatic UV mapping.
- `Generate Texture` - a checkbox that decides whether the texture should be generated during automatic UV mapping or not.

- `Template resolution` defines the size of the texture. The real resolution of the generated texture image is equal to texture width and height multiplied by texture resolution.
- `Automatic UV mapping` - a button that triggers the automatic UV mapping.

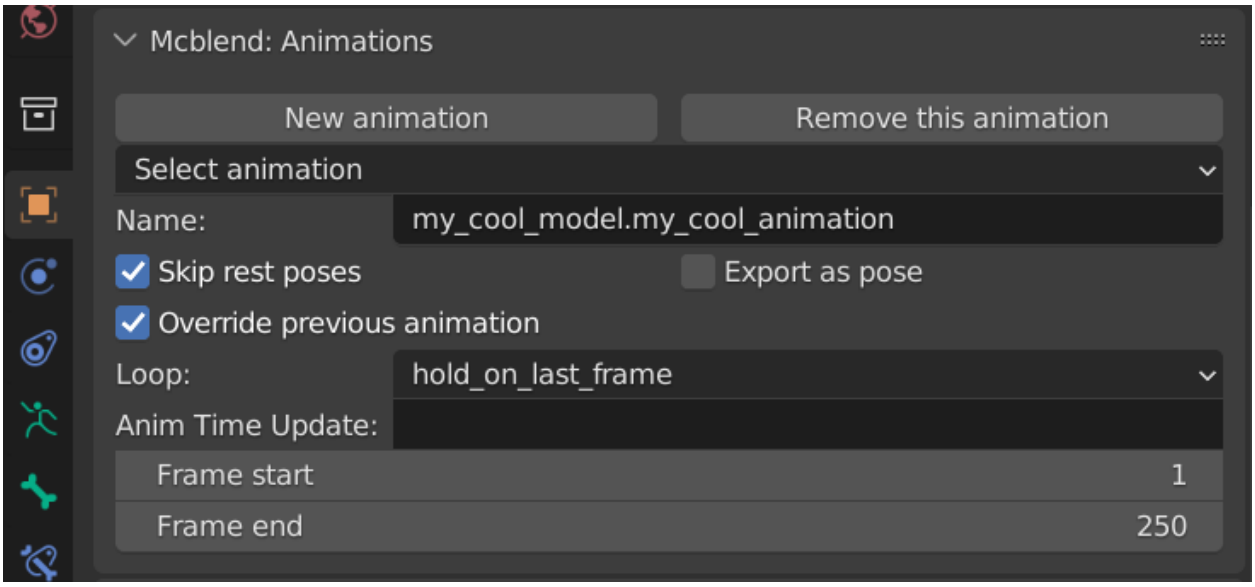
24.2.2 Mcblend: Render Controllers



The Materials panel allows you to quickly create materials that are similar to those found in Minecraft. To do this, you can add multiple render controllers to your model. Each render controller can only have one texture, as multitexture materials are currently not supported. However, a render controller can have multiple Minecraft materials, which can be assigned using name patterns in the same way as you would in Minecraft.

Once you've set up your render controller, you can use the `Apply materials` button to automatically create the materials for preview in Blender. This will help you get a better sense of how your model will look in Minecraft.

24.2.3 Mcblend: Animations

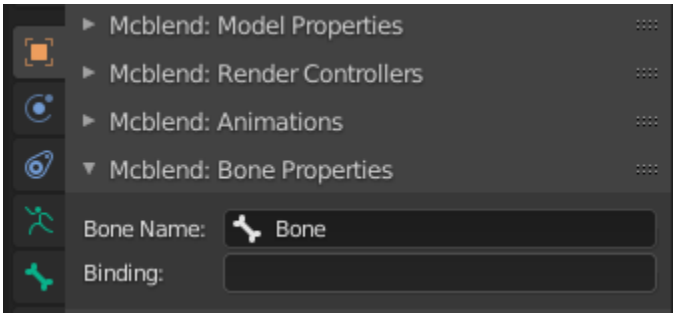


The `Mcblend: Animations panel` allows you to easily switch between animations in your project. These animations are represented as NLA tracks on the armature, with additional data attached to them. Selecting a different animation in the panel will also switch the active NLA track.

- The `New animation` button creates a new animation. You can't use this operator while editing an action of the armature. If you want to create a new action, you need to stash the other action first.
- The `Remove animation` button removes the currently active animation.
- The `Select animation` dropdown list lets you select the animation to edit.
- The `Name` field sets the name of the animation.
- The `Skip rest poses` checkbox enables animation export optimization. If enabled, the keyframes that don't affect the armature (because they are the rest poses) are skipped in the exported file. In most cases, it's recommended to enable this option.
- The `Export as pose` checkbox, if enabled, causes the exported animation to contain only one looped frame.
- The `Override previous animation` field directly translates to the override_previous_animation property of the Minecraft animation. It doesn't affect how the animation is rendered in Blender.
- The `Loop` field directly translates to the loop property of the Minecraft animation file. There are three options: `true`, `false`, and `hold_on_last_frame`.
- The `Anim Time Update` field directly translates to the anim_time_update property of the Minecraft animation file. You should either leave it empty (if you don't want to have anim_time_update in your animation) or put a Molang expression in it. It doesn't affect the animation in Mcblend because Mcblend doesn't support Molang.
- The `Frame start` field indicates the first frame of the animation. It tells Mcblend where the animation starts.
- The `Frame end` field indicates the last frame of the animation. It tells Mcblend where the animation ends.

24.3 Object properties (bone of armature)

24.3.1 Mcblend: Bone properties



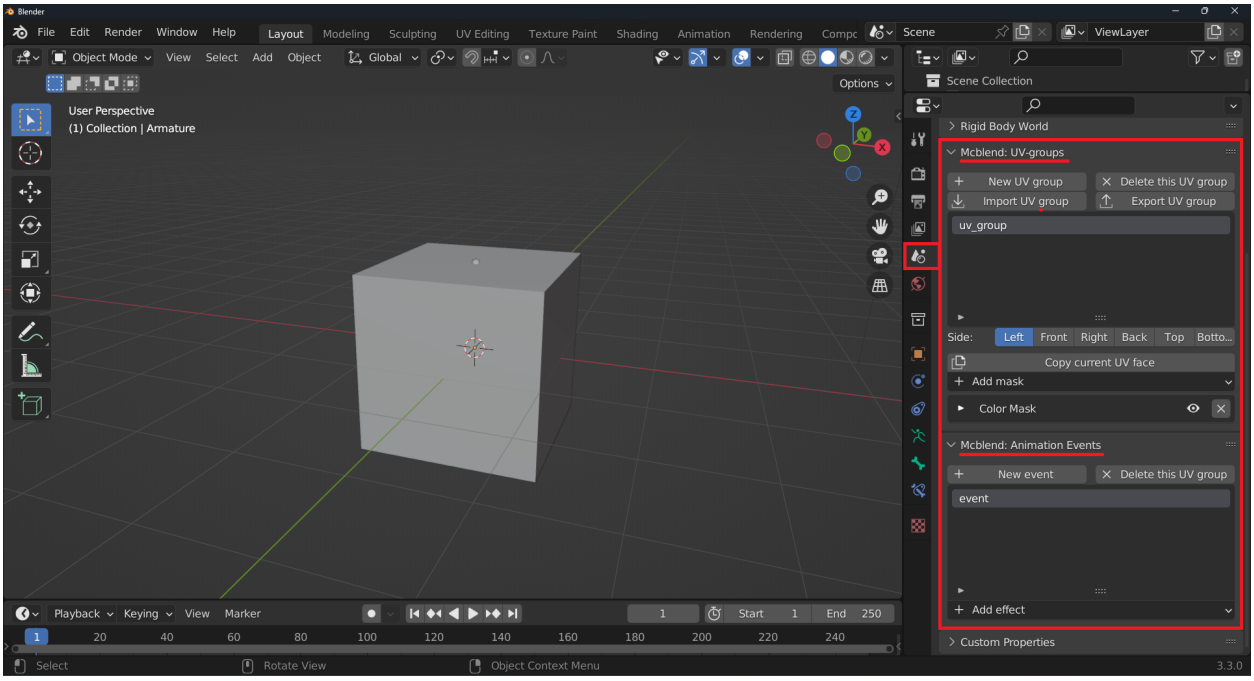
This is the panel visible when opening the `Object Properties` when the active object is a bone in a `Pose Mode`.

- `Bone name` - This displays the name of the currently selected bone. You can also use it to rename the bone.
- `Binding` - This property corresponds to Minecraft's binding property. It is useful for creating models of attachables, but does not affect the model in Blender.

CHAPTER

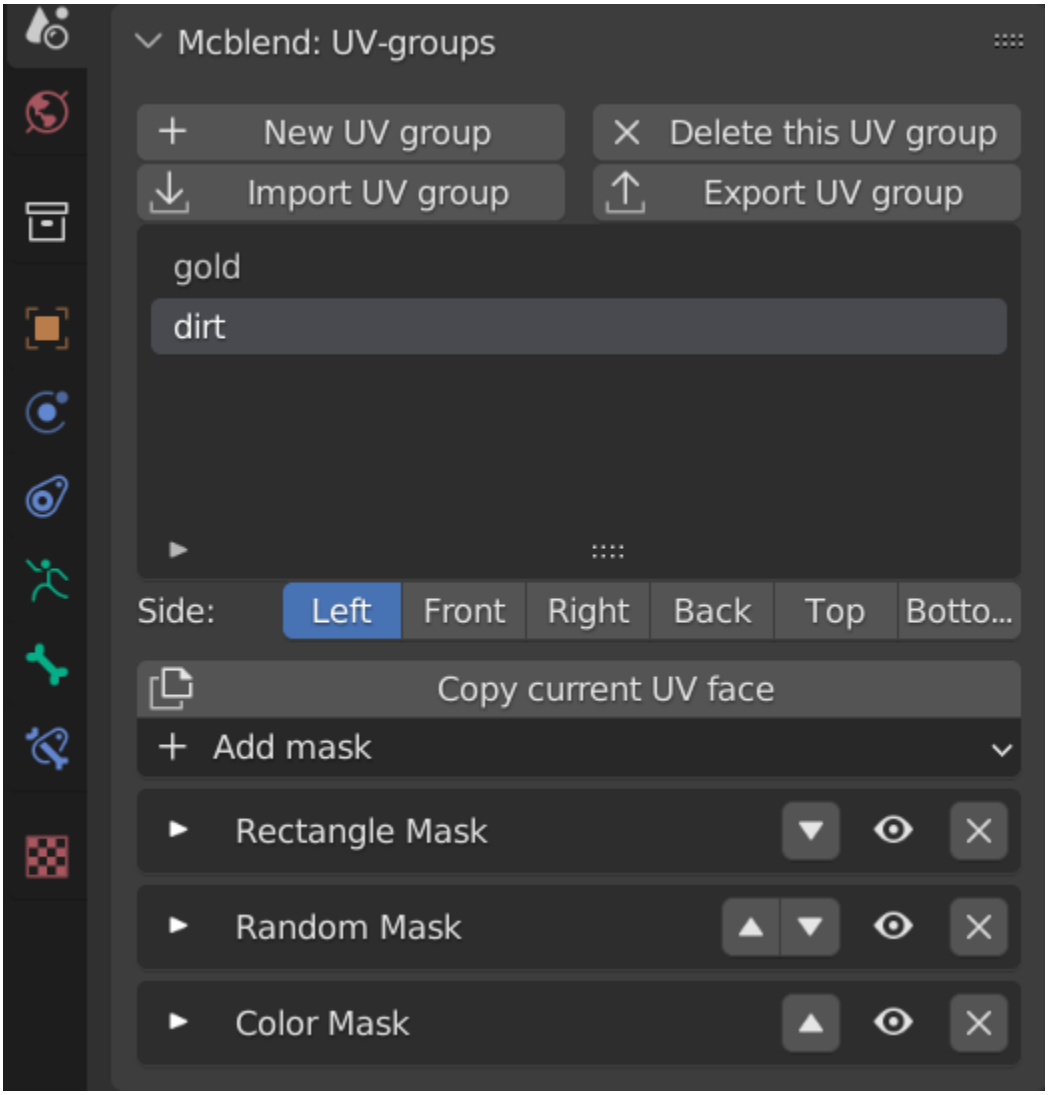
TWENTYFIVE

SCENE PROPERTIES



The Scene properties tab has two new panels:

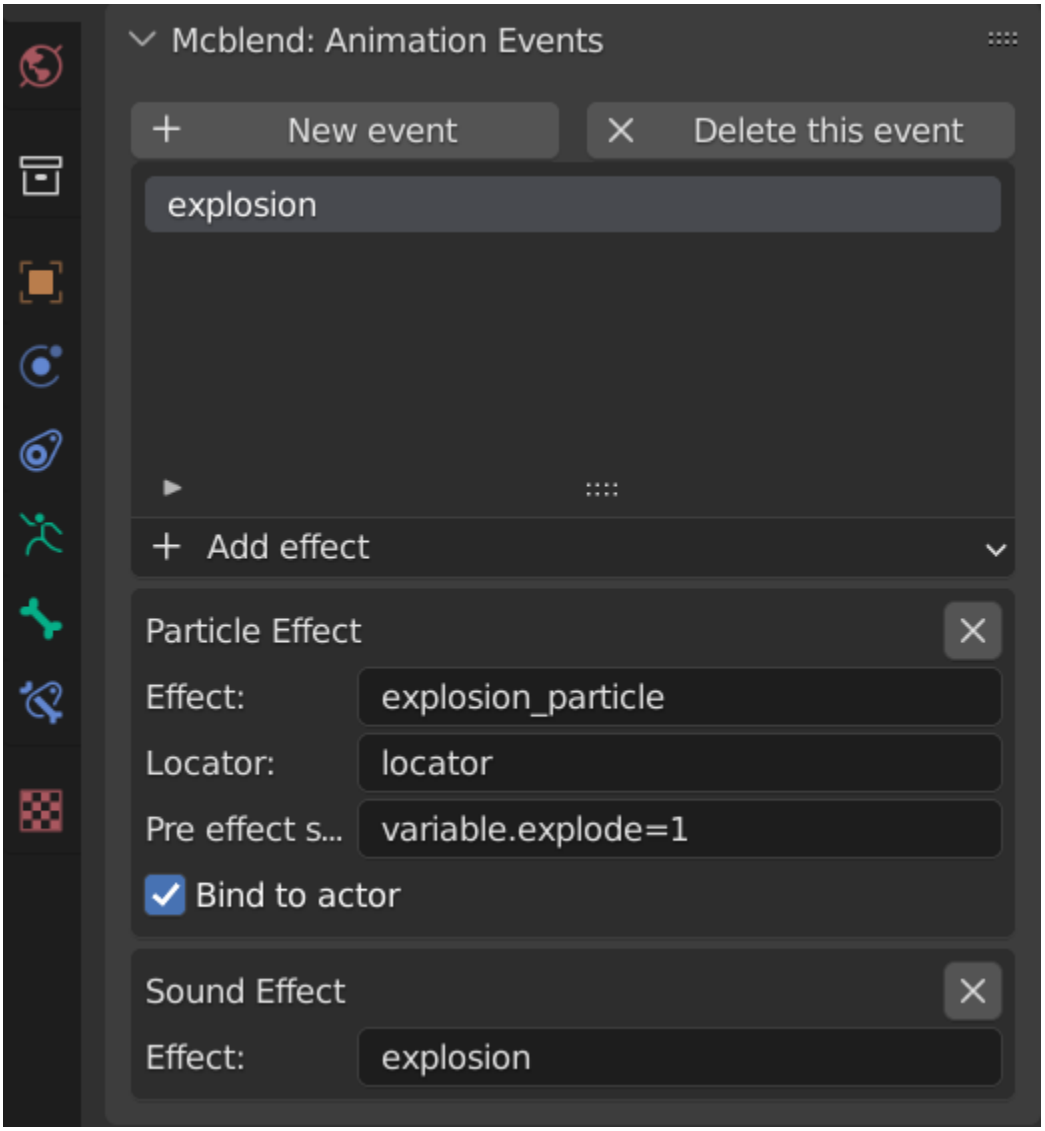
25.1 Mcblend: groups



This panel is used for managing the UV groups.

- The `New UV group` button creates a new UV group.
- The `Delete this UV group` button deletes the selected UV group.
- The `Import UV group` button imports a UV group from a file.
- The `Export UV group` button exports the selected UV group to a file.
- The `Side` radio button select the side of the UV group to edit.
- The `Copy current UV face` button opens a menu for copying the UV face to the other sides.
- The `Add mask` button adds a mask to the selected UV group. The UV masks are described in different sections of this documentation.

25.2 Mcblend: Animation Events



This panel is used for managing sound and particle effects animations.

- The `New event` button creates a new event.
- The `Delete this event` button deletes the selected event.
- The `Add effect` button adds a sound or particle effect to the selected event.

The sound and particle effects have can have additional parameters, all of them are directly related to the settings that you can find in Minecraft animations.

CHAPTER

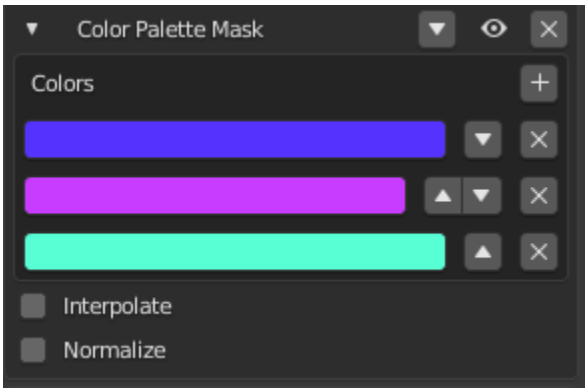
TWENTYSIX

UV GROUP MASKS

UV gorup masks can be added in the `Mcblend: UV groups` menu to configure the process generating textures during *automatic UV mapping*.

All masks have an eye icon in the upper right corner that can be used to temporarily disable/enable the mask.

26.1 Color Palette Mask



This mask takes the grayscale image as an input and maps its brightness values to a color image with a palette defined as a list of colors.

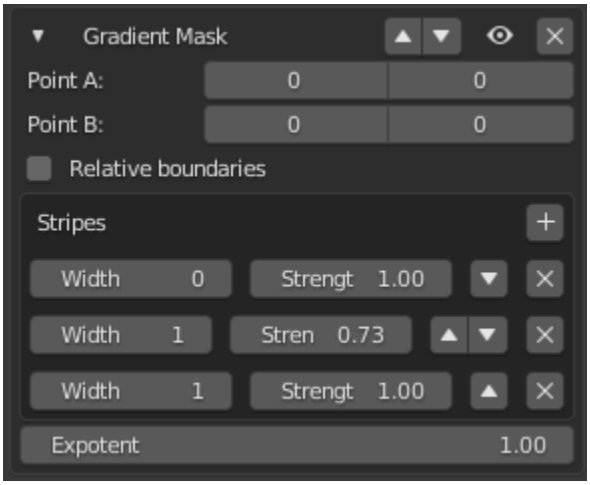
Properties:

- `Colors`: a list of colors to use in the palette.
- `Interpolate`: a toggle that determines whether there should be a smooth transition between the colors in the palette.
- `Normalize`: a toggle that normalizes the input values to ensure that the entire palette is used.

Note: If the input image is not grayscale, it will be converted to grayscale before the mask is applied.

Note: The Color Palette Mask cannot be used inside the Mix Mask. If it is placed inside the Mix Mask, it will have no effect.

26.2 Gradient Mask

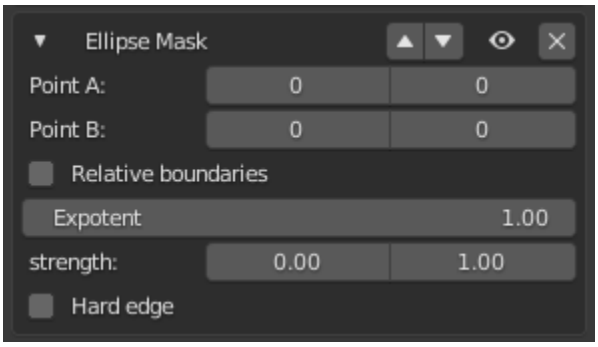


The Gradient Mask generates a grayscale gradient with stripes of different darkness and widths. The direction of the stripes is defined by two points on the texture. The grayscale image is then multiplied by the input image.

Properties:

- `Point A`: the starting point for drawing the gradient stripes.
- `Point B`: the end point for drawing the gradient stripes.
- `Relative boundaries`: a toggle to determine whether points A and B are passed as absolute values (pixels from the lower left corner of the texture) or as a fraction of the texture size (0.0 is the lower left corner, 1.0 is the upper right corner). Negative absolute values represent the number of pixels from the top right corner, starting at -1.
- `Stripes`: a list of stripes in the gradient, including their colors (strengths) and widths. The widths determine the placement of the stripes along a line between points A and B, so the width of the first stripe is usually 0, indicating that it should be drawn at Point A.
- `Exponent`: the filter image is raised to this power before it is multiplied by the image.

26.3 Ellipse Mask

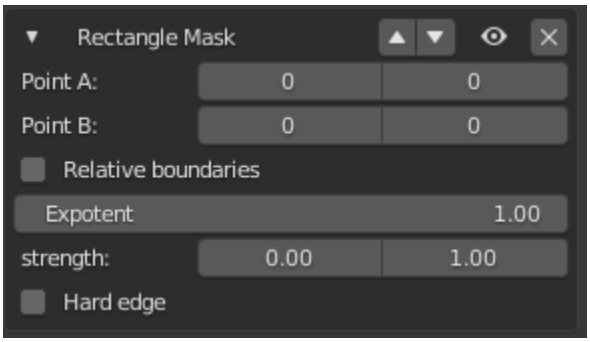


The Ellipse Mask is a tool that creates a grayscale image of an ellipse between points A and B. This image is then multiplied by the input image.

Properties:

- `Point A` and `Point B` define the boundaries of the ellipse.
- The `Relative boundaries` property determines whether the points are given as absolute values (number of pixels from the lower left corner) or as fractions of the texture size (0.0 lower left corner, 1.0 upper right corner). Negative values represent the number of pixels from the top right corner, starting at -1.
- The `Exponent` property determines the power to which the filter image is raised before it is multiplied by the input image.
- The `Strength` property sets the minimum and maximum values of brightness for the filter image.
- The `Hard edge` property determines whether the edges of the ellipse should be smooth or hard.

26.4 Rectangle Mask

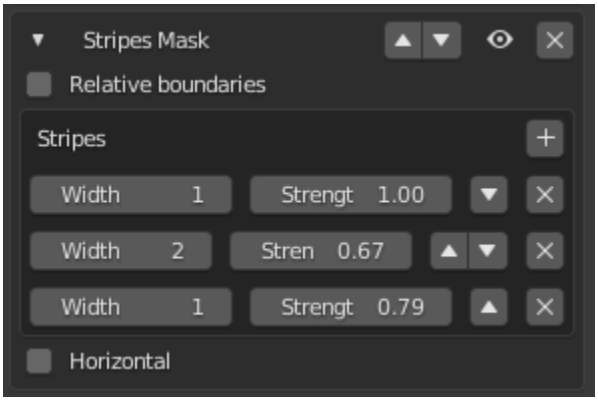


The Rectangle Mask creates a grayscale image of a rectangle between two points, `Point A` and `Point B`. This grayscale image is then multiplied by the input image.

Properties:

- `Point A` and `Point B`: Opposite corners of the rectangle.
- Relative boundaries: Whether points A and B should be passed as absolute values (number of pixels from the lower left corner of the texture) or as a fraction of the texture size (0.0 lower left corner, 1.0 upper right corner). The absolute values can also be negative, representing the number of pixels from the top right corner (starting at -1).
- `Exponent`: The filter image is raised to this value before it is multiplied by the input image.
- `Strength`: The minimum and maximum values of brightness for the created filter image.
- `Hard edge`: Determines whether the edges of the rectangle should be hard or smoothly interpolated towards the edges of the image.

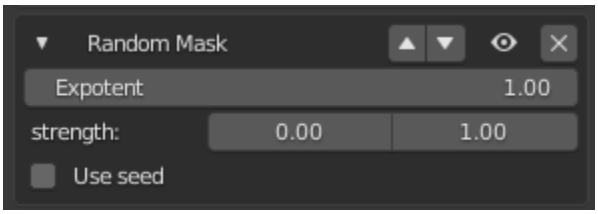
26.5 Stripes Mask



The Stripes Mask creates a grayscale image with repeating stripes of a specified width and brightness. When this grayscale image is multiplied by the input image, it creates a striped effect. The mask has the following properties:

- `Relative boundaries`: Indicates whether the width of the stripes is expressed as a fraction of the image's width or height.
- `Stripes`: A list of the stripes, including their width and brightness.
- `Horizontal`: Determines whether the stripes should be vertical or horizontal.

26.6 Random Mask



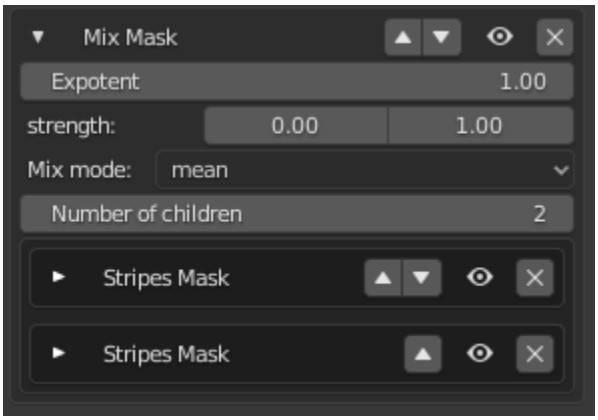
The Random Mask creates a grayscale image with randomly bright pixels (a random noise). When applied, this image is multiplied by the input image. The following properties can be adjusted:

- `Exponent`: the filter image is raised to this power value before being multiplied by the input image.
- `Strength`: the minimum and maximum brightness values of the pixels on the filter image.
- `Use seed`: allows you to set the seed for the color randomization, allowing for consistent results when using the same seed.

26.7 Color Mask

The color mask multiplies the input mask by a color.

26.8 Mix Mask



The Mix Mask allows you to mix multiple masks using different methods besides the default multiplication.

Properties:

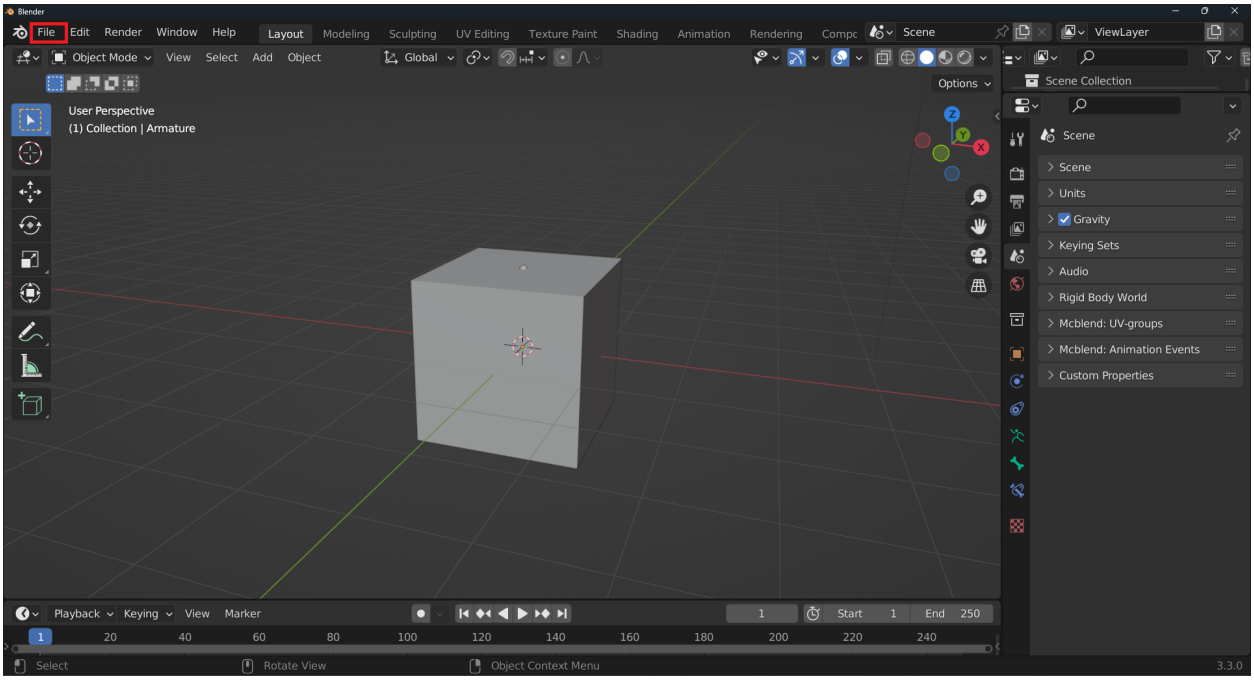
- `Exponent`: Raises the filter image to the power of this value before multiplying it with the input image.
- `Strength`: Defines the minimum and maximum values of brightness for the returned filter image. The brightness values of the filter image are mapped to fit within this range.
- `Mix mode`: Determines how to mix the filter images produced by other masks. There are four options: min, max, mean, and median.
- `Number of children`: The number of masks being mixed. Takes the masks below this mask and mixes them together.

Note: Mix mask ignores the color palette masks, since the color palette masks do not create a filter image (they just alter the image from the input).

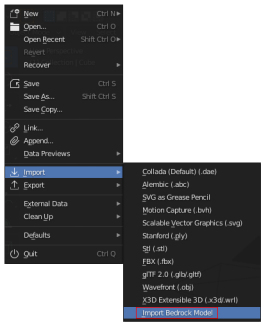
CHAPTER

TWENTYSEVEN

GUI - MENU ITEMS

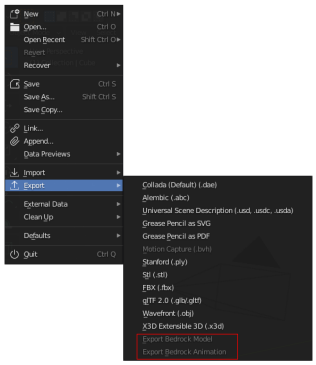


27.1 File > Import menu



- The `Import Bedrock Model` menu item imports a Bedrock model from a file.

27.2 File > Export menu



- The `Export Bedrock Model` menu item exports the selected model to a file.
- The `Export Bedrock Animation` menu item exports the selected animation to a file.

CHAPTER

TWENTYEIGHT

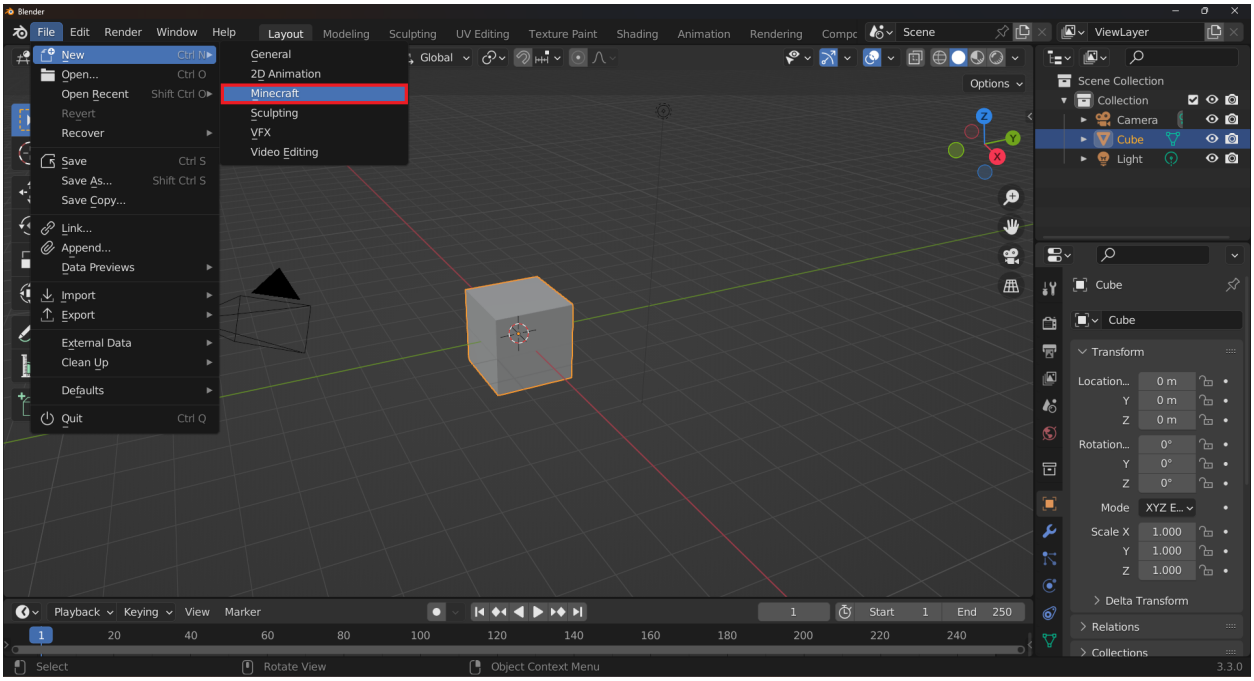
THE MCBLEND BLENDER TEMPLATE

The “Tips and tricks” section of the documentation contains tips related to Mcblend. Some of them are related to the addon and some of them explain some basics of using Blender, which can be useful for working with Mcblend. Most of the configuration tips are already applied as a default configuration of the the Blender template designed for Mcblend.

You can download the template from the Mcblend Project Template repository:

https://github.com/Nusiq/mcblend-project-template/releases/tag/mcblend-10.1

The template can be added to the `File > New` menu used for starting new projects.



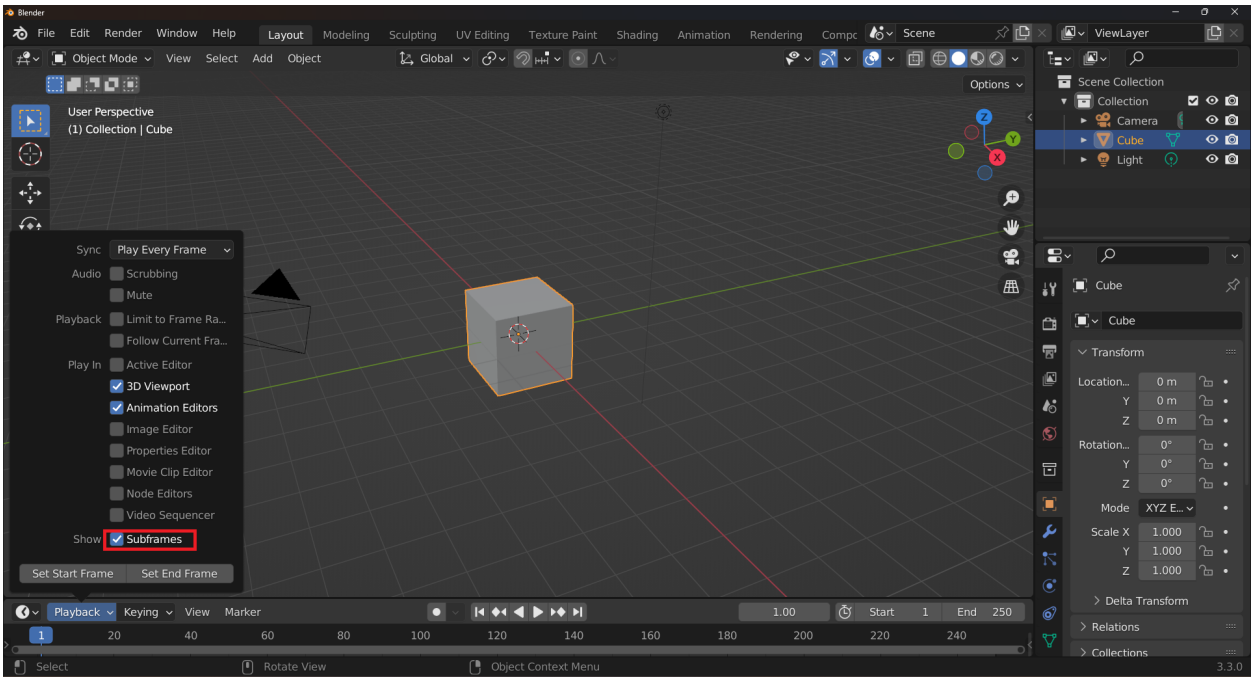
You can find more details about the template in the README file of the repository.

CHAPTER

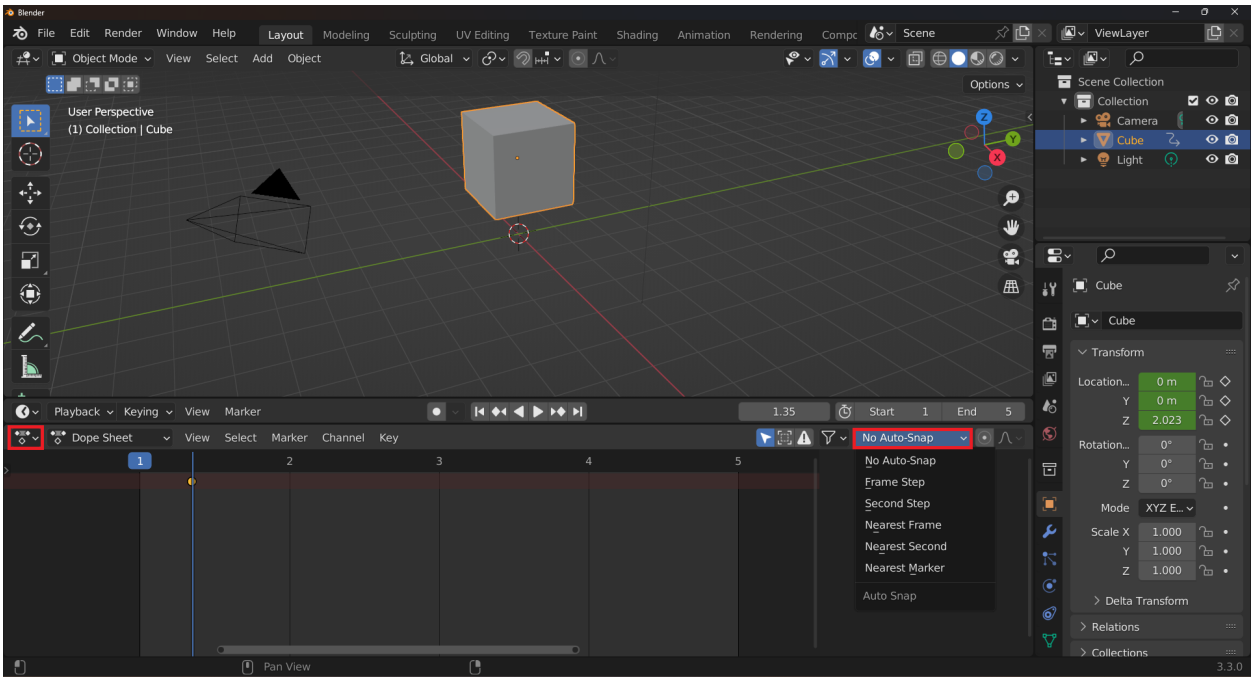
TWENTYNINE

ENABLING SUBFRAMES

Starting from Mcblend version 10.1.0, you can use subframes in your animations. Subframes are a type of keyframe that is set on a timeline between two frames. To view subframes in Blender, enable the `Subframes` option in the `Timeline` panel in the `Playback` section.



However, even if you enable subframes, the `I` shortcut will only add keyframes on whole frames. To add subframes, first add a keyframe and then use the `Dope Sheet` to move it to the desired position on the timeline. Before moving the keyframe, select the `No auto snap` option in the `Dope Sheet` panel.


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compc
Scene
ViewLayer
Object Mode View Select Add Object
Global
Options
User Perspective
(1) Collection | Cube
Scene Collection
Collection
Camera
Cube
Light
Cube
Cube
Transform
Location... 0 m
Y 0 m
Z 2.023
Rotation... 0°
Y 0°
Z 0°
Mode XYZ E...
Scale X 1.000
Y 1.000
Z 1.000
Delta Transform
Relations
Collections
Playback Keying View Marker
1.35
Start 1 End 5
Dope Sheet View Select Marker Channel Key
No Auto-Snap
No Auto-Snap
Frame Step
Second Step
Nearest Frame
Nearest Second
Nearest Marker
Auto Snap
Pan View
3.3.0

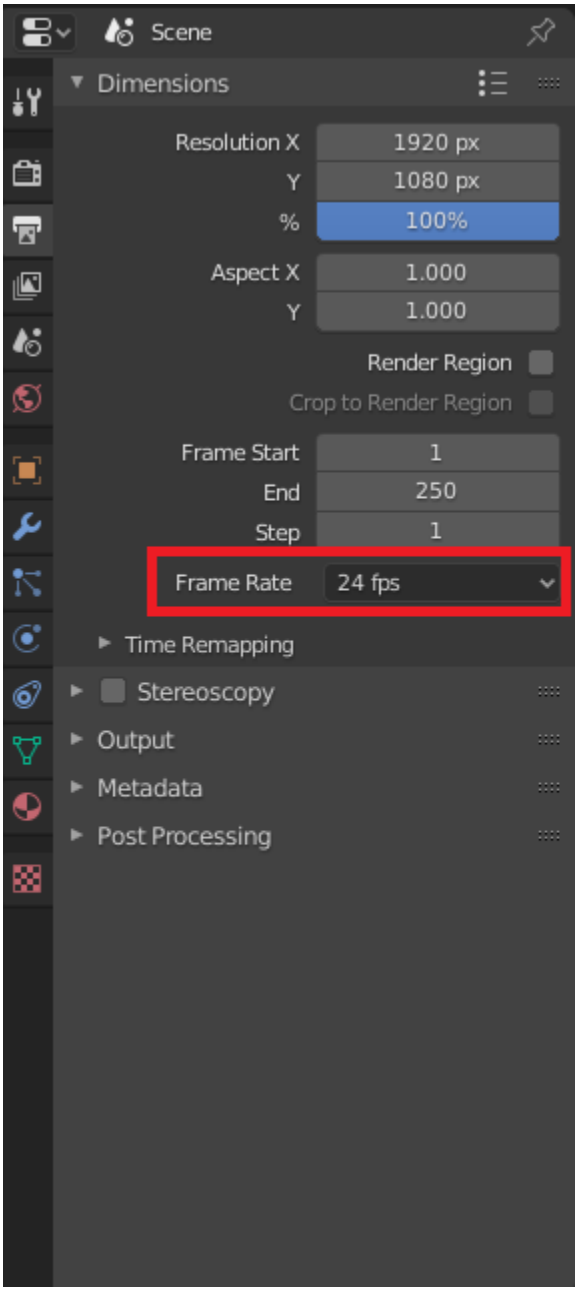
CHAPTER

THIRTY

MATCHING FRAMERATE

By default, Blender uses 24FPS framerate. Minecraft uses seconds to define the timestamps of keyframes in animation. It's good to change the framerate setting into something that divides 1 second period into something nice - for example (25FPS or 20FPS). 1/24 is 0.0416666 but 1/25 is 0.04 which looks neater in the animation file.

You can find the framerate setting in `Output Properties -> Frame Rate`.

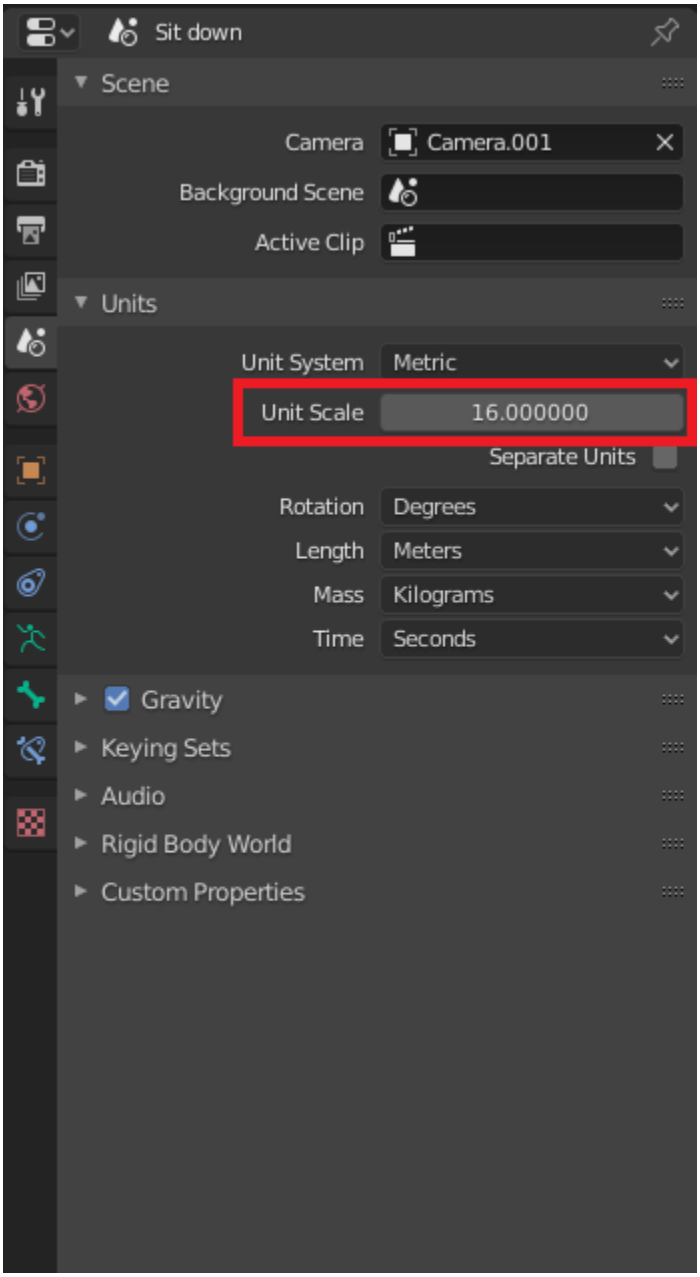

Scene
Dimensions
Resolution X
1920 px
Y
1080 px
%
100%
Aspect X
1.000
Y
1.000
Render Region
Crop to Render Region
Frame Start
1
End
250
Step
1
Frame Rate
24 fps
Time Remapping
Stereoscopy
Output
Metadata
Post Processing

CHAPTER

THIRTYONE

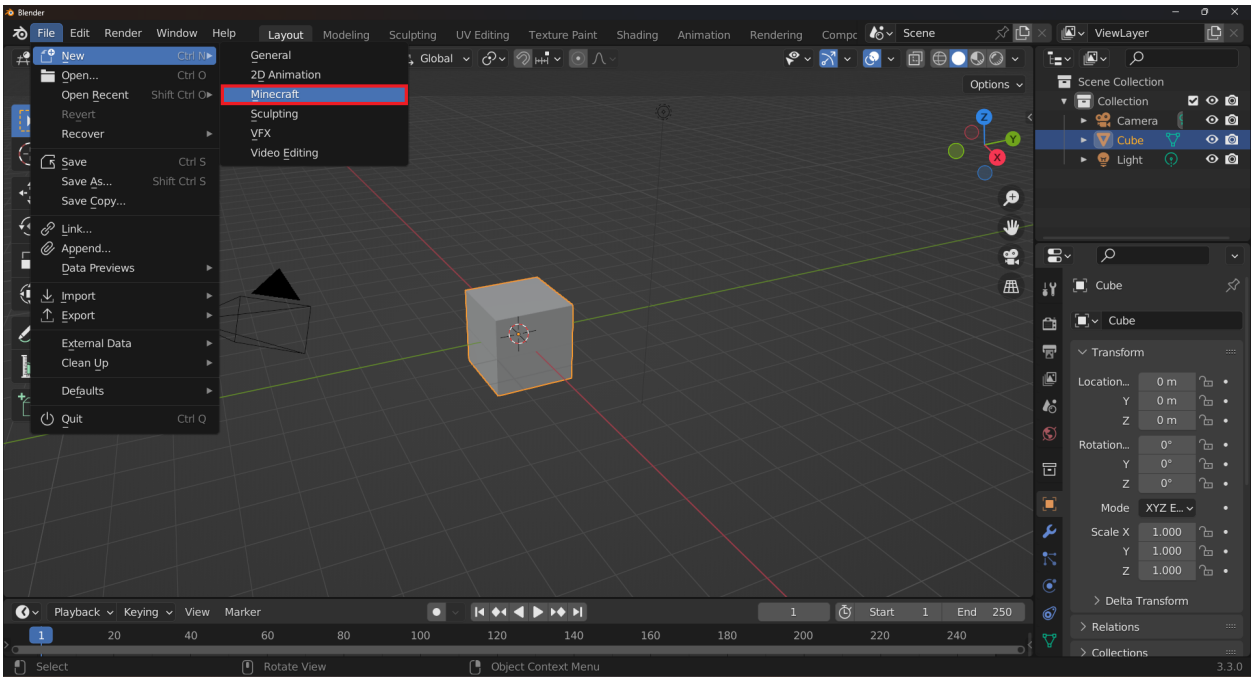
WORLD UNIT SCALE

By default, 1 meter in Blender is equal to one block in Minecraft. One Minecraft meter is 16 units of length of the model. You might want to measure the size of the model using these units instead of meters. You can go to `Scene properties -> Unit scale` to scale the units used in Blender. Changing the value of this property to 16 will set one length unit in Blender equal to one length unit of a model.



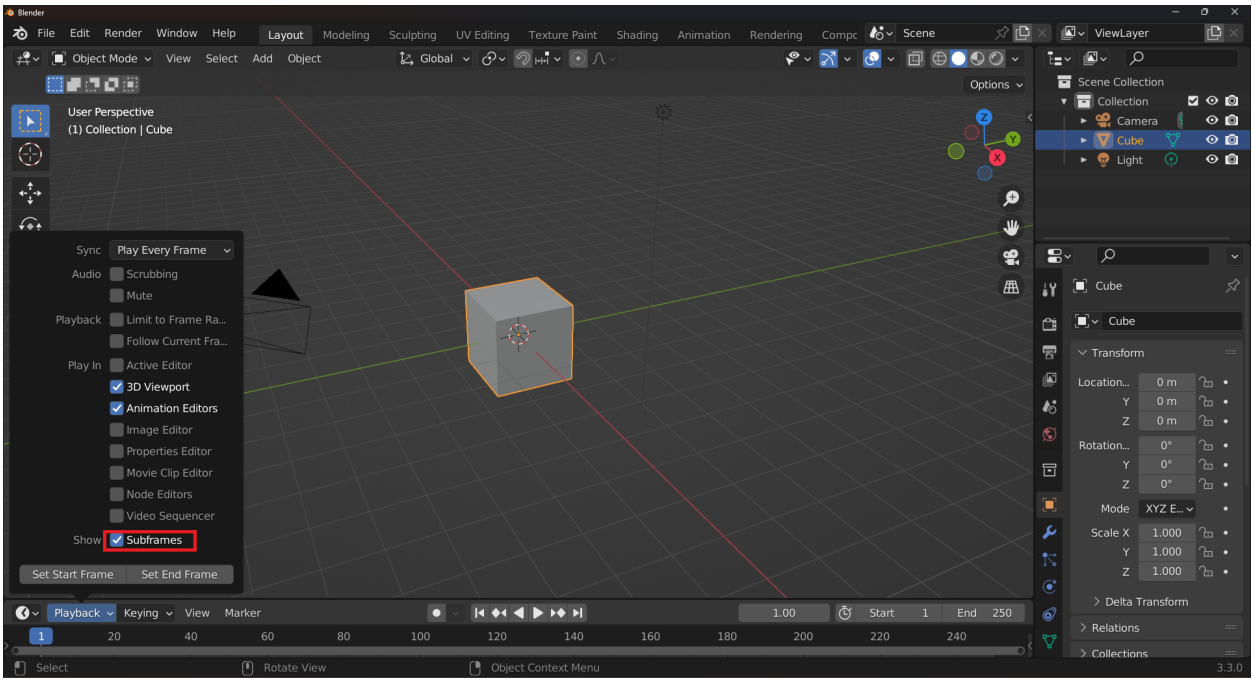
31.1 The Mcblend Blender template

You can download a template project for blender and add it to your `File > New` menu for starting new projects. The description of what the template contains and how to add it to Blender can be found on the template repository: https://github.com/Nusiq/mcblend-project-template/releases/tag/mcblend-10.1

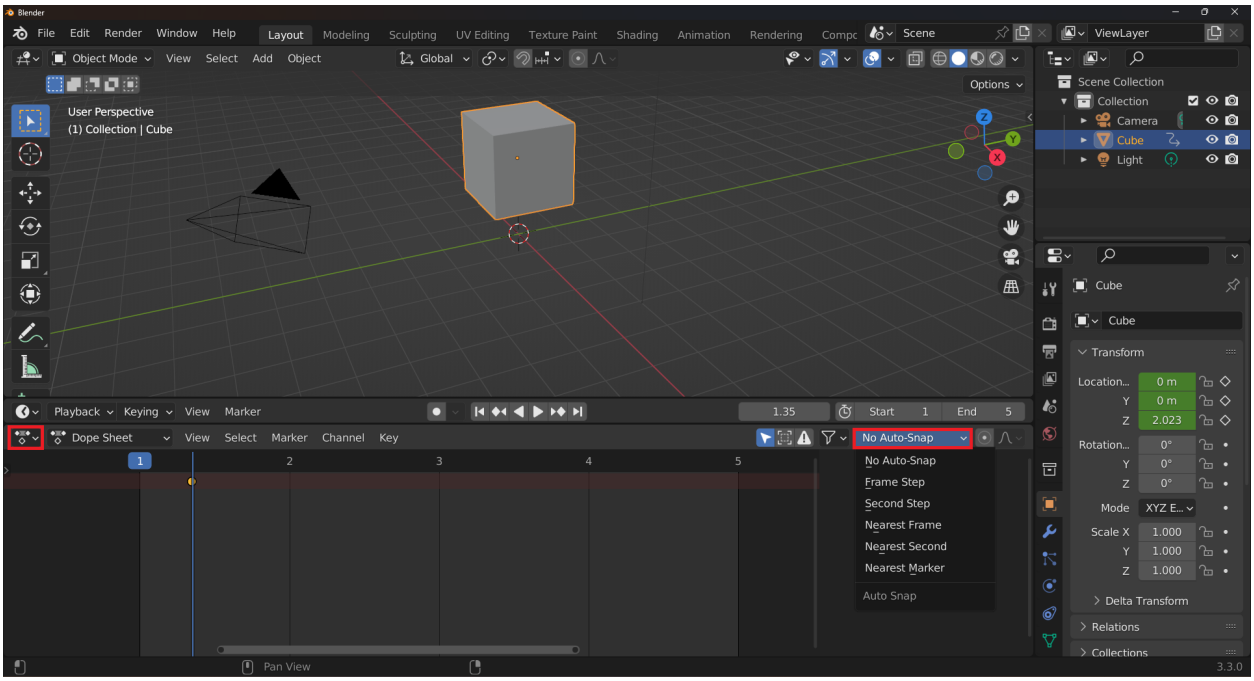


31.2 Enabling subframes

Starting from Mcblend version 10.1.0, you can use subframes in your animations. Subframes are a type of keyframe that is set on a timeline between two frames. To view subframes in Blender, enable the `Subframes` option in the `Timeline` panel in the `Playback` section.



However, even if you enable subframes, the `I` shortcut will only add keyframes on whole frames. To add subframes, first add a keyframe and then use the `Dope Sheet` to move it to the desired position on the timeline. Before moving the keyframe, select the `No auto snap` option in the `Dope Sheet` panel.


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering
Object Mode View Select Add Object
Global
Options
User Perspective
(1) Collection | Cube
Scene Collection
Collection
Camera
Cube
Light
Transform
Location...
Rotation...
Mode
Scale X
Delta Transform
Relations
Collections
Playback Keying View Marker
1.35
Start 1 End 5
Dope Sheet View Select Marker Channel Key
No Auto-Snap
No Auto-Snap
Frame Step
Second Step
Nearest Frame
Nearest Second
Nearest Marker
Auto Snap
Pan View
3.3.0

CHAPTER

THIRTYTWO

USING MCBLEND TO DECORATE THE LEVELS IN THE GAME

This tutorial explains a concept of creating custom models to be used as decorations for specific areas in the game. It doesn't go into the details of using Mcblend, because the basics are explained in other sections of the documentation.

32.1 The concept

The decoration entities are entities that are added to the game for purely visual purposes. They are not used for gameplay, but they can be used to add some detail to the game world. One way to implement them is to create multiple models and spawn the entities that use them directly in the game. This approach is easy to implement, but it means that your world will have a lot of entities, and it also gives you less control over the placement of the decorations.

This tutorial explains a different approach that has its own advantages and disadvantages. The idea is to create a single model that contains all the decorations for a given area. It takes advantage of the fact that the game allows you to export the models of the environment using structure blocks, and that Mcblend has a model merging feature. With these two features you can design the decorations in Blender and then export them to the game as a single model.

32.2 Preparation

Before you start, you need to have a level in the game that you want to decorate and a bunch of models that you want to use as decorations. In this tutorial we will use a simple wooden house.


Minecraft


Minecraft

32.3 Exporting the environment model

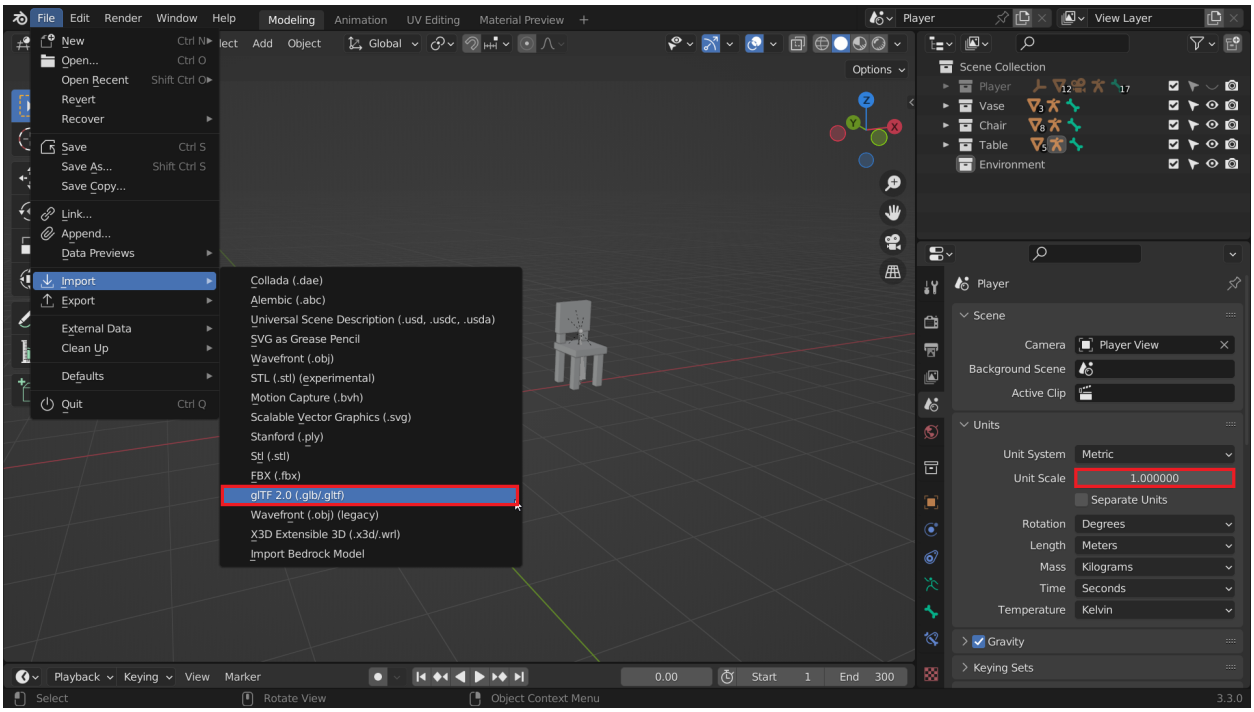
Exporting the model is a feature of Minecraft. You simply need to get the structure block using the `/give @s structure_block` command and place it in the game. Then you can use the interface to select the area and export the model. You can learn more about structure blocks in the Minecraft Wiki.

32.4 Importing the environment model into Blender

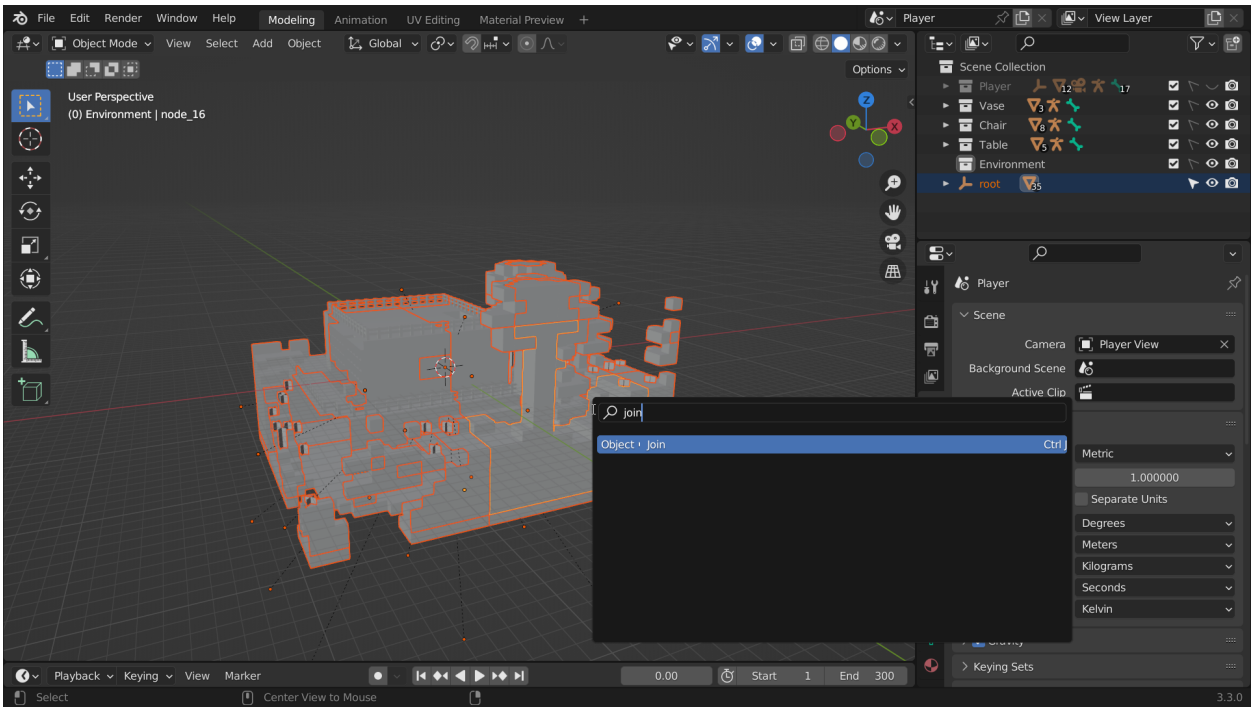
Once you have exported the model, you can import it into Blender. The models exported from Minecraft use the `.glb` format, so you need to import it with the appropriate importer. You can find the importer in the `File > Import > glTF 2.0 (.glb/.gltf)` menu.

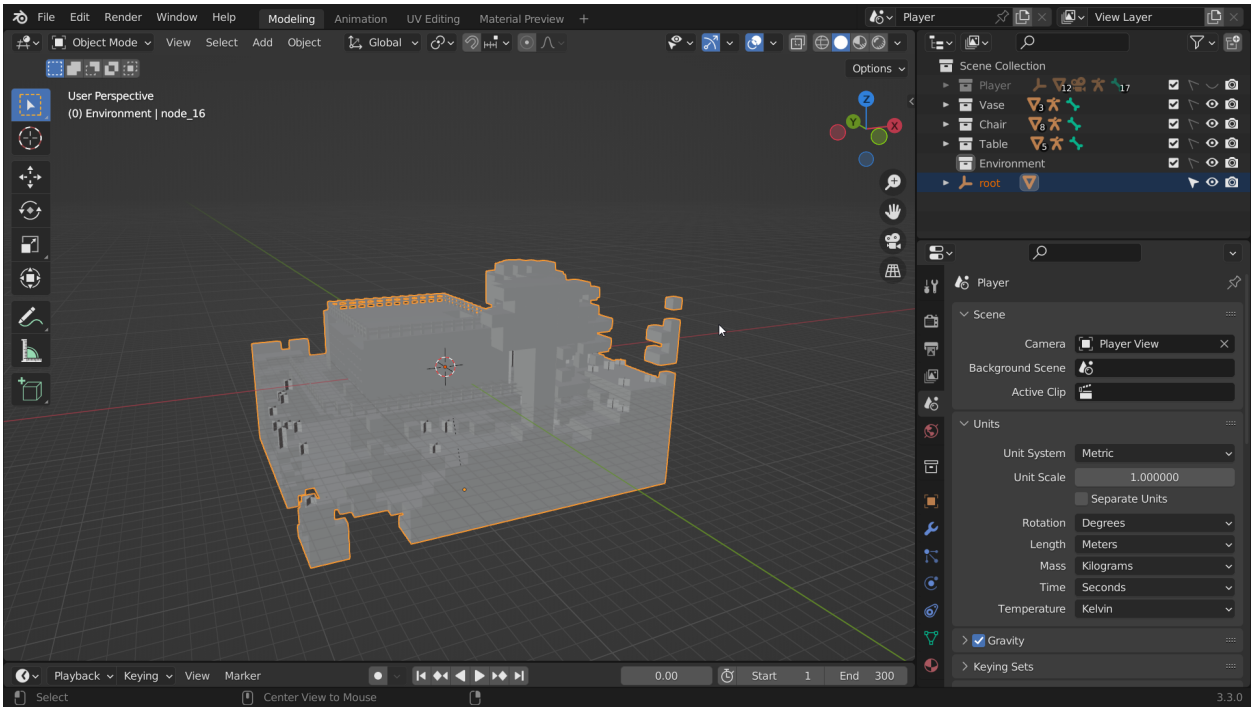
Note: It's important to note that you should set the `Unit Scale` of the scene to `1.0`, otherwise the model will be bigger or smaller than it should be. Blender's default settings already use this value, but some tutorials in the documentation recommend changing it to 16.0, which makes more sense for Minecraft models. You can change this value at any time. It only affects the display of distances and the scale of imported models.

You can learn more about the `Unit Scale` in the *World Unit Scale* page of the documentation.

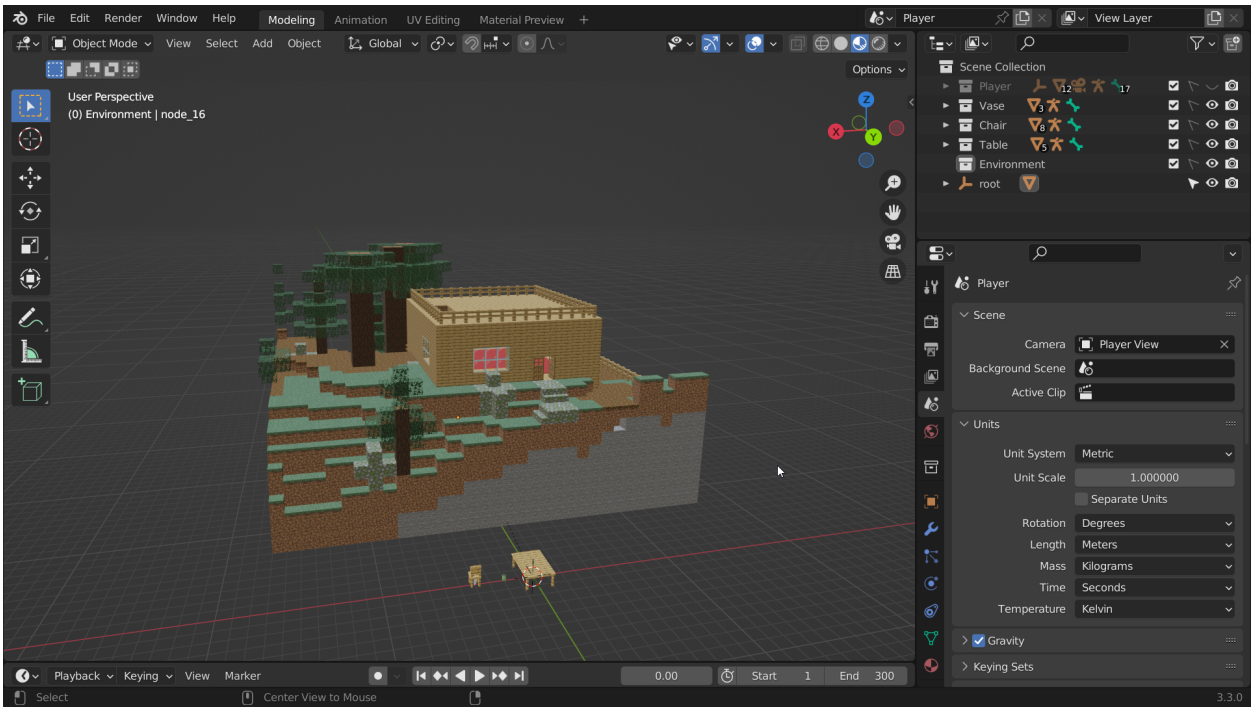


The imported model will contain several objects. For your convenience, you can merge them into a single object. You won't be interacting much with this model, it's only purpose is to give you a sense of space when placing the decorations.



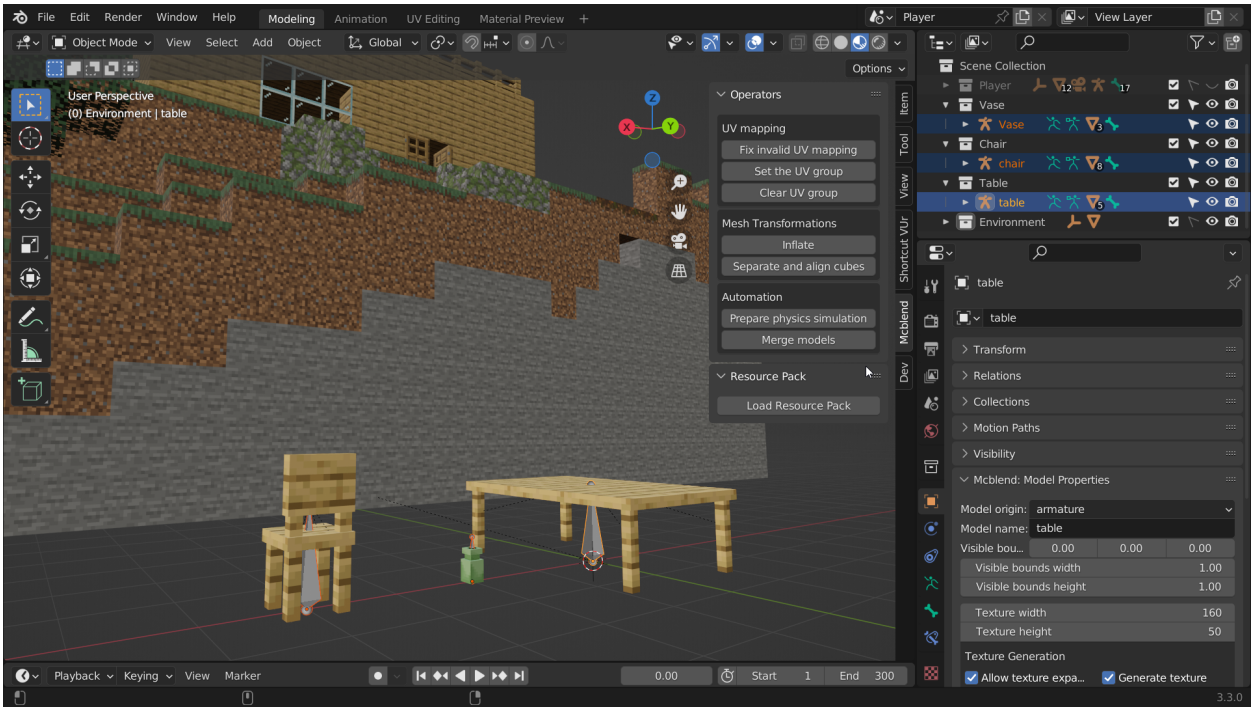


Once the objects are joined, place the model where it won't interfere. As you move the model, make sure it is aligned with the grid, otherwise it will be harder to properly align the decorations after they are imported into the game.



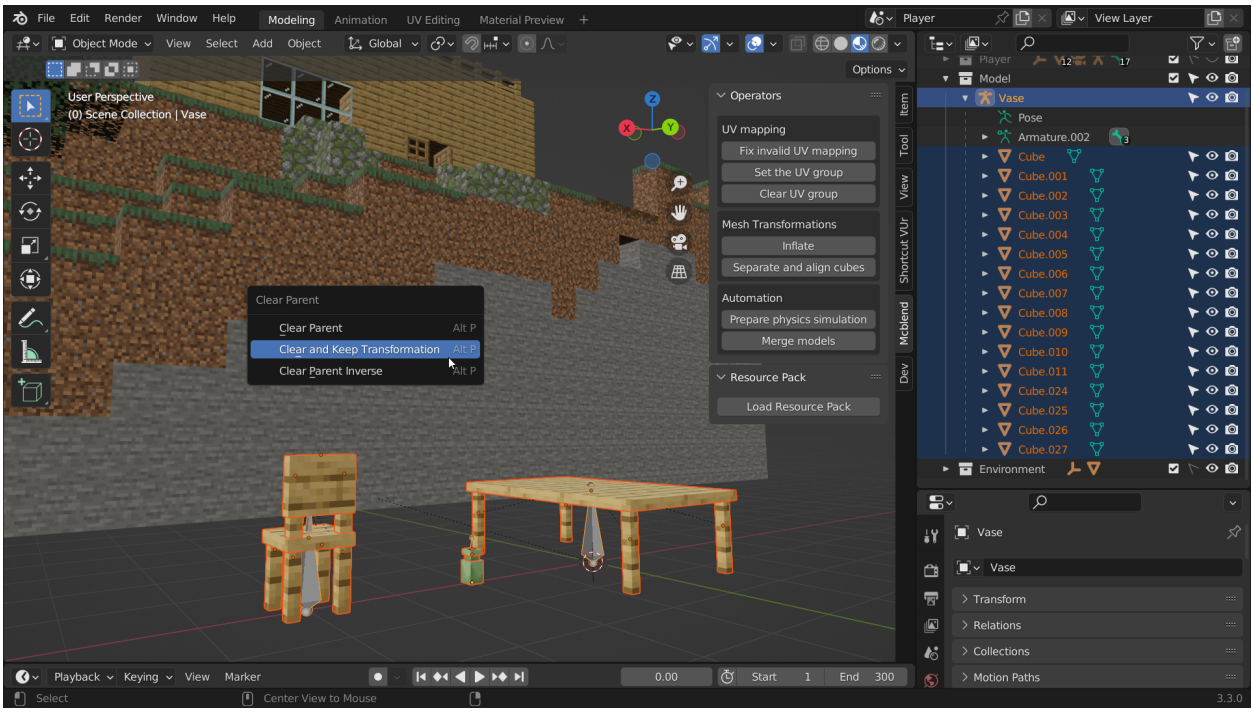
32.5 Merging the decoration models

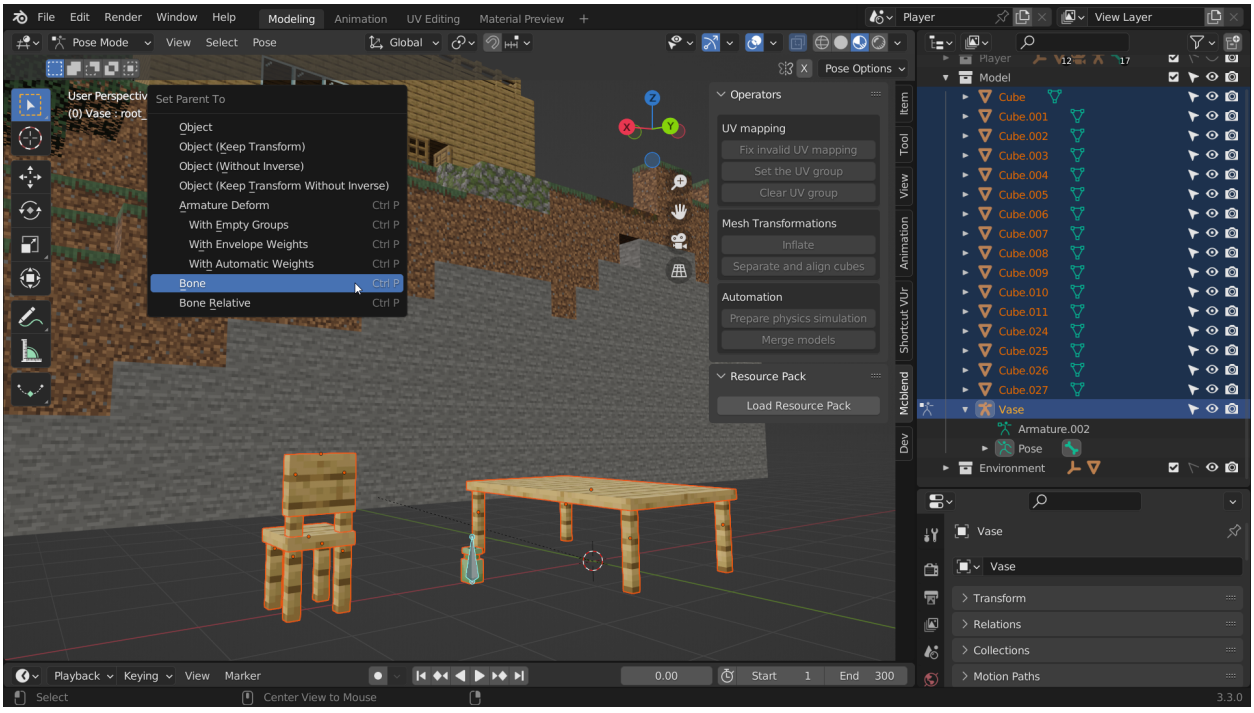
Before you start decorating the level, you should prepare all the models you want to use. In this example we will only use 3 models - a table, a chair and a vase. You can learn more about merging models in the *Merging models* page of the documentation.

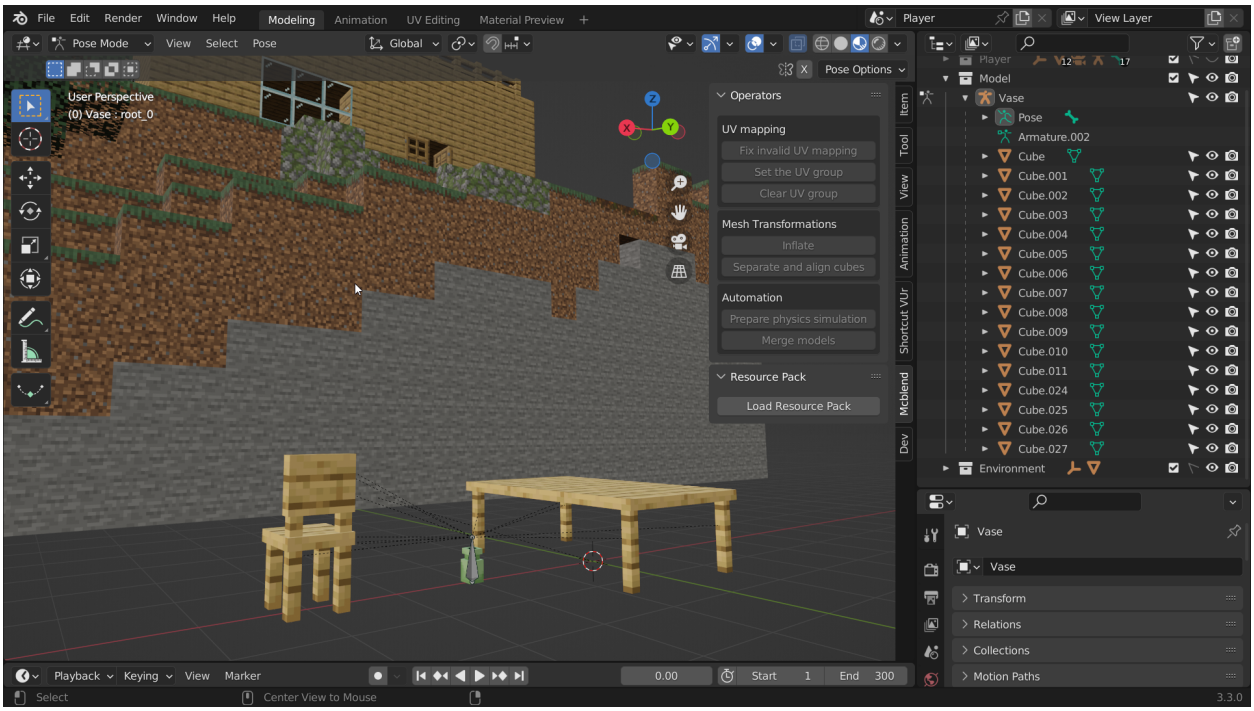


After merging the models, you can simplify the bone structure. Our model is completely static, so we only need a

single bone to be used as the parent for all objects. This is optional, but you can simplify the armature by deleting the parents of all objects, deleting all bones except one, and then setting the parent of all objects to this single remaining bone.



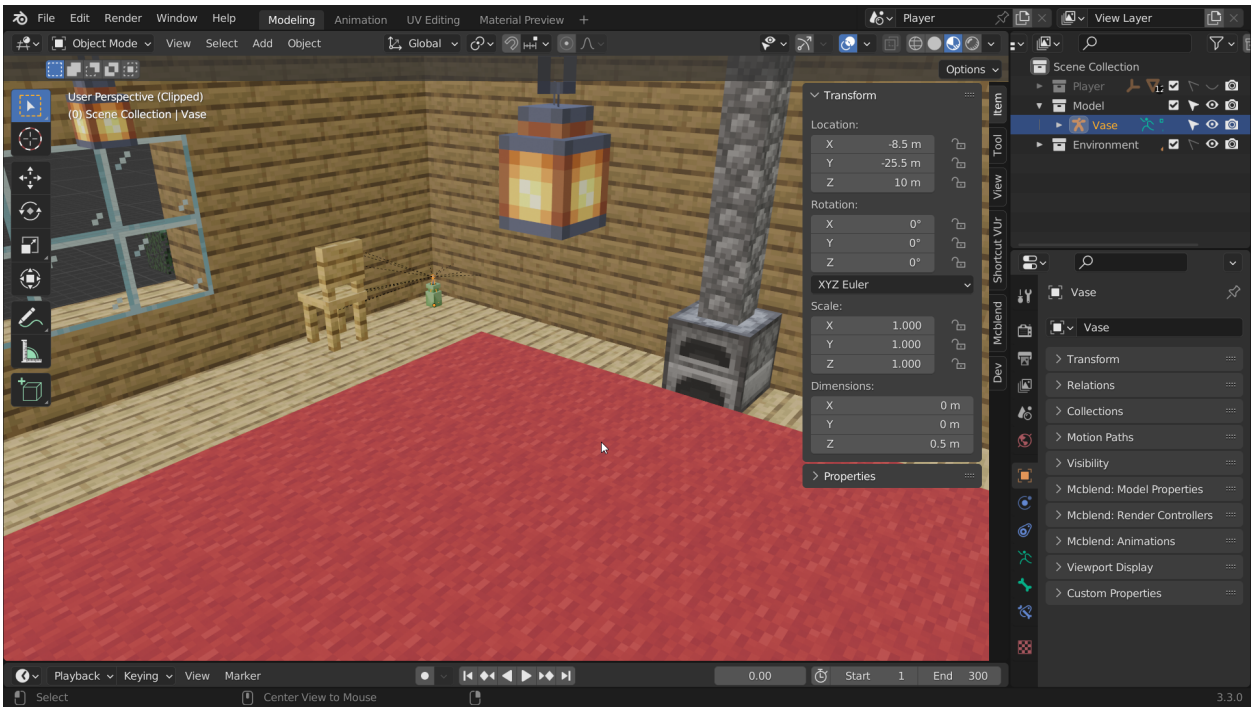




32.6 Selecting the origin of the model

Everything will be packed into a model represented by an entity. The entity needs to be placed somewhere in the game world. Since Minecraft's render distance is limited, it's important to place the entity in a place where it will be visible. If the area you are decorating is small, like the room in the example, you don't need to worry about this too much. A good place for the model is a corner of the room, because it's easy to find the correct spawn position when importing the model into the game.

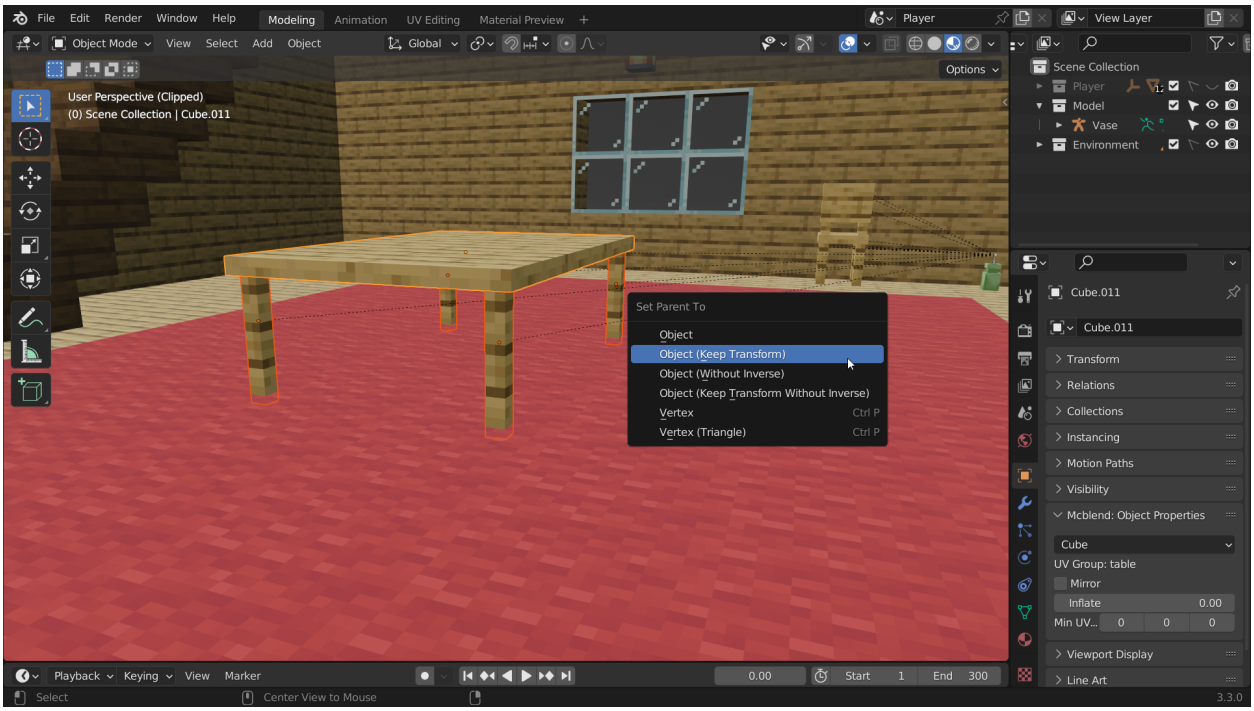
Move the anchor with the combined model so that its origin is in the corner of the room in the center of the block. Don't worry about the parts of the model clipping through the walls, we haven't started rearranging them to decorate the room yet.



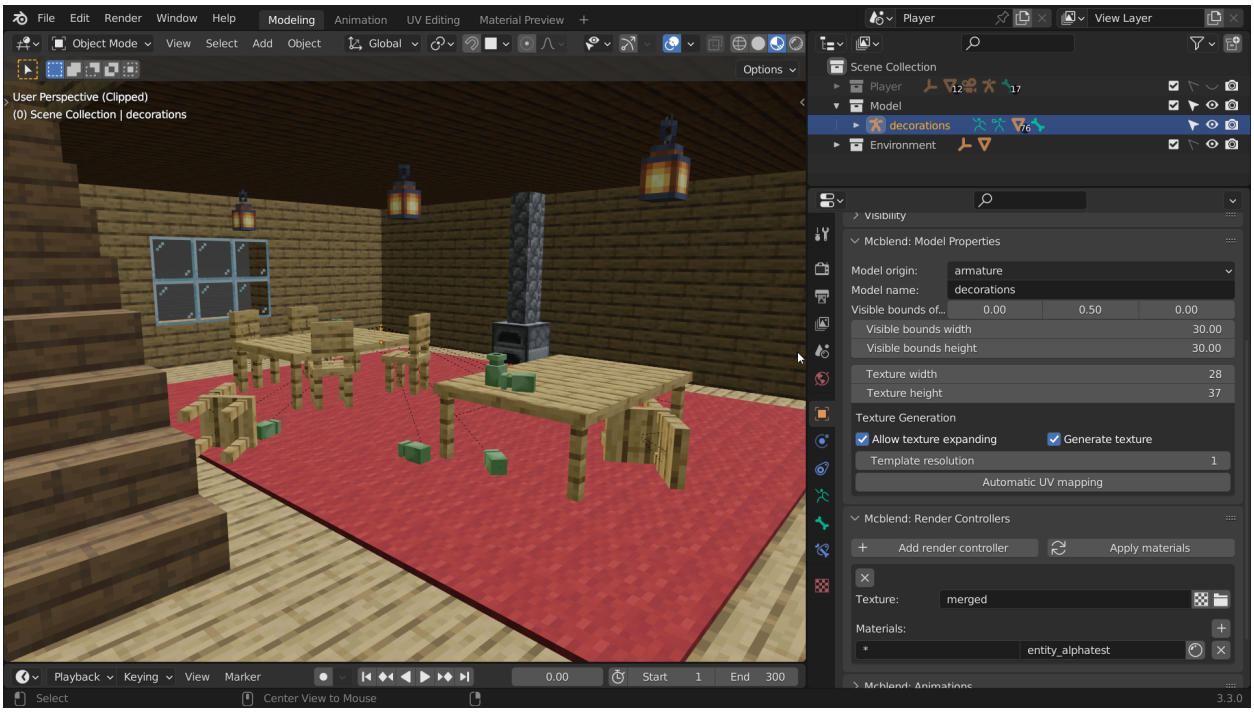
32.7 Placing the decorations

Now you can start placing decorations. There are no special steps for this, you can just copy the models and place them inside your imported environment model.

Since a single model in Mcblend contains multiple objects, it's nice to group them together. You can do this by simply parenting the meshes of a single decoration to its largest object. Mcblend doesn't require the objects to be parented directly to the armature, so this doesn't affect the structure of the model. This setup makes it easier to copy the decorations because you can simply select them using the `Select Grouped` option in the `Select` menu (`SHIFT + G`).

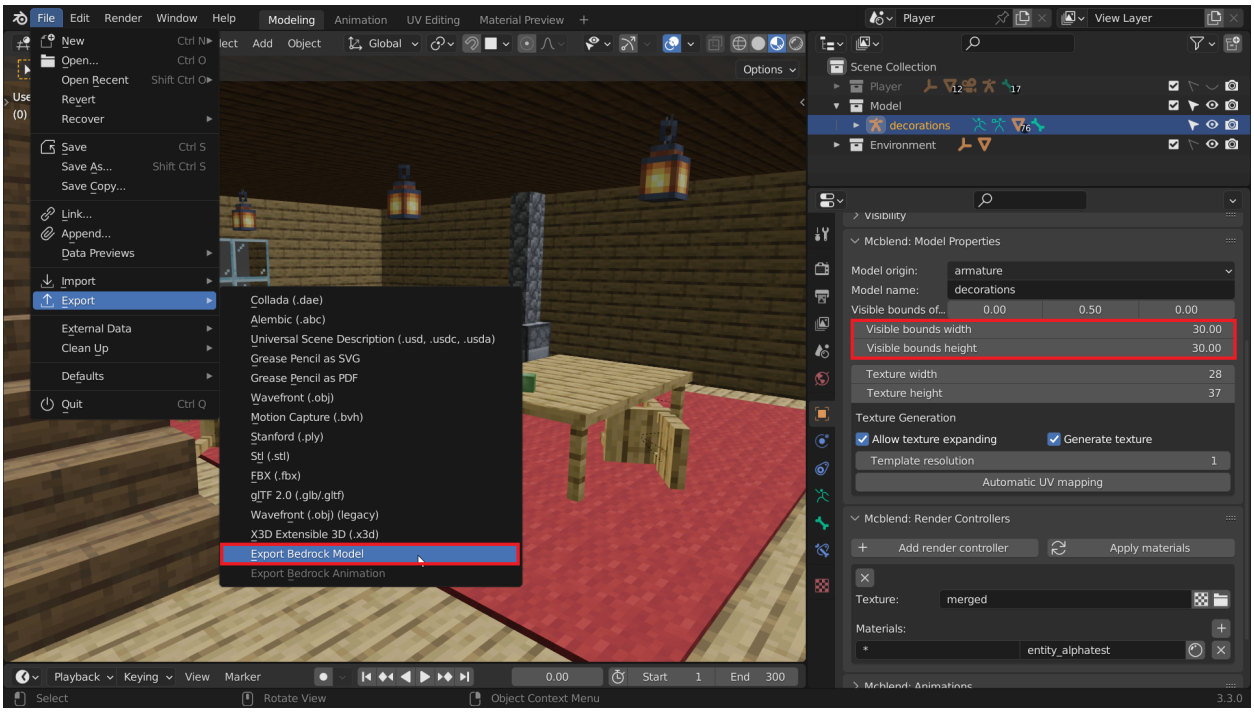


The advantage of decorating the levels in Blender is that you can rotate, move and scale the decorations freely. In this example, the models aren't aligned with the grid, some of them are slightly rotated and some are even lying on their sides on the ground. With an entity-based approach, you would have to have an animation for each custom rotation like this.

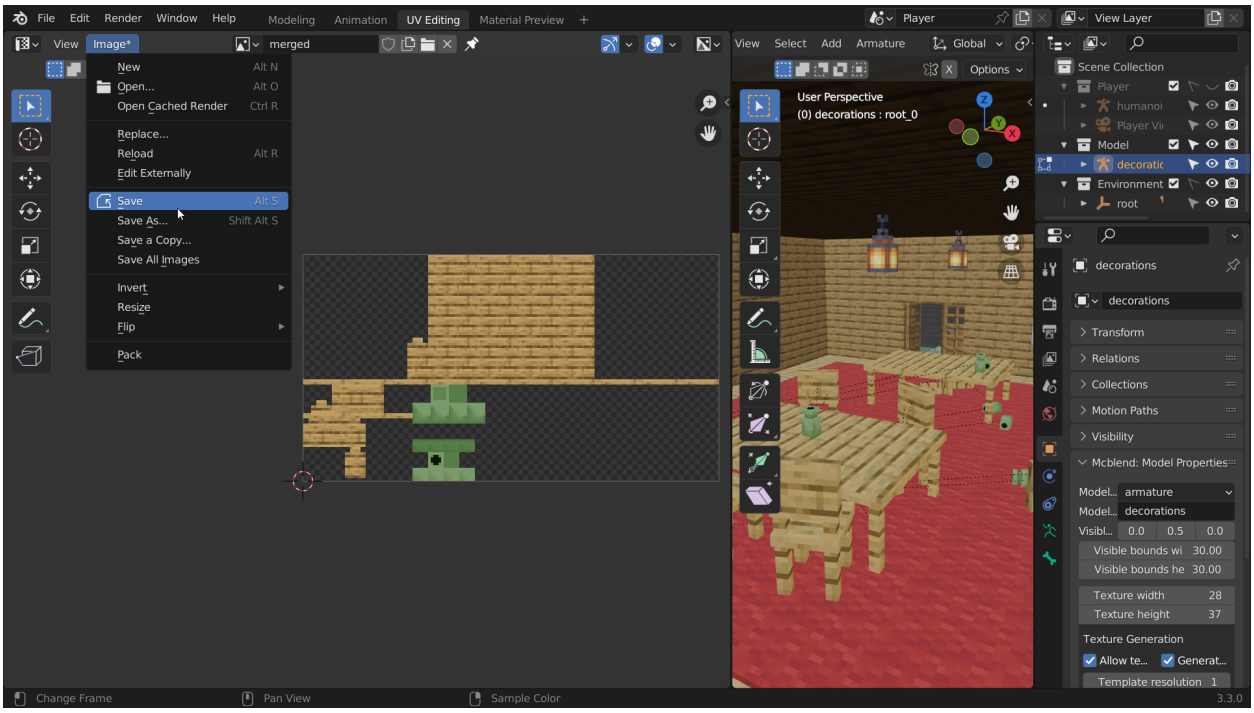


32.8 Exporting the model

When you're happy with your decorations, the last step is to export the model. You can read about exporting Mcblend models on the *Exporting Models* page of the documentation. Before exporting, make sure that you have set large visible bounds for the model. All the decorations of the model should fit inside the bounds. In this example, 30x30 blocks should be sufficient.



You should also save the texture of the model so that you can use it in the game.



32.9 Summoning the entity

This tutorial will not cover the details of creating entities in Minecraft, as that is outside the scope of Mcblend. You can find resources on this in the Bedrock Wiki or in the official Minecraft Bedrock Edition Creator documentation. The behavior used in the example is the simplest you can create. The only purpose of the entity is to display the model.

After setting up the resource and behavior packs, all you need to do is invoke the entity at the same location you chose for the model in Blender.

32.10 The final result

CHAPTER

THIRTYTHREE

OVERVIEW

Mcblend is a Blender plugin designed for Minecraft creators. It enables users to create and modify Minecraft Bedrock Edition models within the Blender interface. With Mcblend, creators can use Blender's tools to design and texture new models for Minecraft, or bring existing models into Blender for further editing. Mcblend is a useful tool for anyone looking to use Blender's powerful 3D modeling and animation tools to enhance their Minecraft creations.

33.1 Features

- Exporting and importing models for Minecraft Bedrock Edition, including support for polymesh and cube-based models of attachables and entities.
- Exporting keyframe animations and poses for entities and attachables, and baking animations into Minecraft format, including support for using inverse kinematics and constraints to create complex movements.
- Generating UV maps and textures for Minecraft models.
- Access to Blender features such as physics simulation and rigging for animating Minecraft models, and a function for automatic rigid body setup for a model.

33.2 Planned features

Some of the planned features for the Mcblend project can be found in the "Issues" tab of the GitHub repository. The page is open for suggestions and feature requests.