

# PSA/PSK exporter for modo

The exporter is based on the existing collada exporter. The **green** text indicates features which are already done. The “strike through” formatting marks existing features which are obsolete.

## **Overview:**

The LUA script exports geometry information from modo in Epic's PSK/PSA format: a PSK file defines skeletal structure while a PSA contains a number of named animation sequences.

The script always processes the first of the foreground layers and all of its children layers, if any. Passed parameters control the script's behavior, so it can export data in different ways:

- ~~Object mode: The first layer is the render geometry, the children layers are collision geometries.~~
- *Rigid body mode*: The first layer and its children are making a rigid body hierarchy.
- *Soft body mode*: The first layer is the render object and its children define the bone hierarchy.

The UV set called “UV1” is exported along with material names, ~~vertex color, collision geometries~~ and generated bones. It considers active morphmaps when processing vertex data, and weight maps for bone influences. Bone and item animation is exported and can be arranged to animation sequences. The animation output is baked frame by frame data.

## **General behavior:**

Skeletal data is exported as a .PSK file while animation sequences are exported as a .PSA file.

The exported files path is defined by the “save method” script parameter:

- “SaveMode1”: The PSK and PSA files will be exported to the directory the original scene file resides in. (Default behavior, will be used if no save mode param is present.)
- “SaveMode2”: The files will be exported to the LW content directory.
- “SaveMode3”: The user can define the target directory and the base filename in a file requester window.

Exported filename is the same as the first layer's name with the name of an active morphmap appended to it, and with an underscore between the two. (For example: “Meatball\_Squashed”) Existing files are automatically overwritten.

Bones are named after the related layers.

Referenced items should not be processed.

Separated polygon islands define geometry shading. Each polyisland has its own smoothing group.

The size of the exported object must be adjusted to compensate for different unit systems. 1 unit in modo (meter, inch, etc) must be 1 unit in unreal eventually.

Example script snippet:

```
meter_per_gunit = lxq("pref.value units.gameScale ?") [1]    -- Get accurate game units.

-- Determining scale factor for different unit systems --
if UnitSystem == "GAMEUNIT" then UnitScale = (1/meter_per_gunit)
elseif UnitSystem == "METER" then UnitScale = 1
elseif UnitSystem == "MILLIMETER" then UnitScale = 1000
elseif UnitSystem == "FEET" then UnitScale = 3.2808398950131
elseif UnitSystem == "INCH" then UnitScale = 39.3700787
else UnitScale = 1 end
```

## **Naming conventions:**

The script processes only layers and vertexmaps with proper names.

Vertex map names:

- UV: “UV1”, “UV2”, “UV3”.
- RGB: “Color”.
- Weight: Whichever matches the name of a bone.
- Morph: Any active.

Collision primitive names:

- Box (Bounding box): “UBX”
- Sphere (Bounding sphere): “USP”
- Convex mesh: “UCX”

Anim sequences (see bellow): “@SequenceName-StartFrame-EndFrame”

In rigid body mode, bones get the name of their respective layer.

In soft body mode bones get the name of their respective layer, stripped from the “>???” prefix. (The “???” is 3 letters defining the influence volume type. See bellow.)

## **Processing modes:**

### Object mode:

~~The first layer is the render geometry, the children layers are collision geometries (if any). (...)~~

### Rigid body mode:

The first layer and its children are making a rigid body hierarchy. The polygons of the layers are merged and will be used as the render object. Each part of the geometry (found on a separate layer originally) will have a bone assigned to it. That bone has a 100% influence on the vertices of the related layer, and 0% on every other vertex. The bone hierarchy reflects the layer hierarchy.

A bone is created from parent's pivot to own pivot. If no parent present (first layer) then it starts at the origin.

### Soft body mode:

The first layer will be the render object, they children hierarchy will be used as the template for bone generation. The name is considered as a bone if its name starts with “>”, otherwise it is discarded and its branch is not discovered any further.

The layer name also contains a 3 letter influence volume type and a “\_”. (For example: “>BOX\_Head”)

The influence volume type can be “USP” “SPH” for spherical/elliptical shape, “UBX” “BOX” for box shape or “WGH”.

If the influence definition is “WGH” then a weightmap should be used for influence weights. The relevant weightmap's name is the same as the name of the bone.

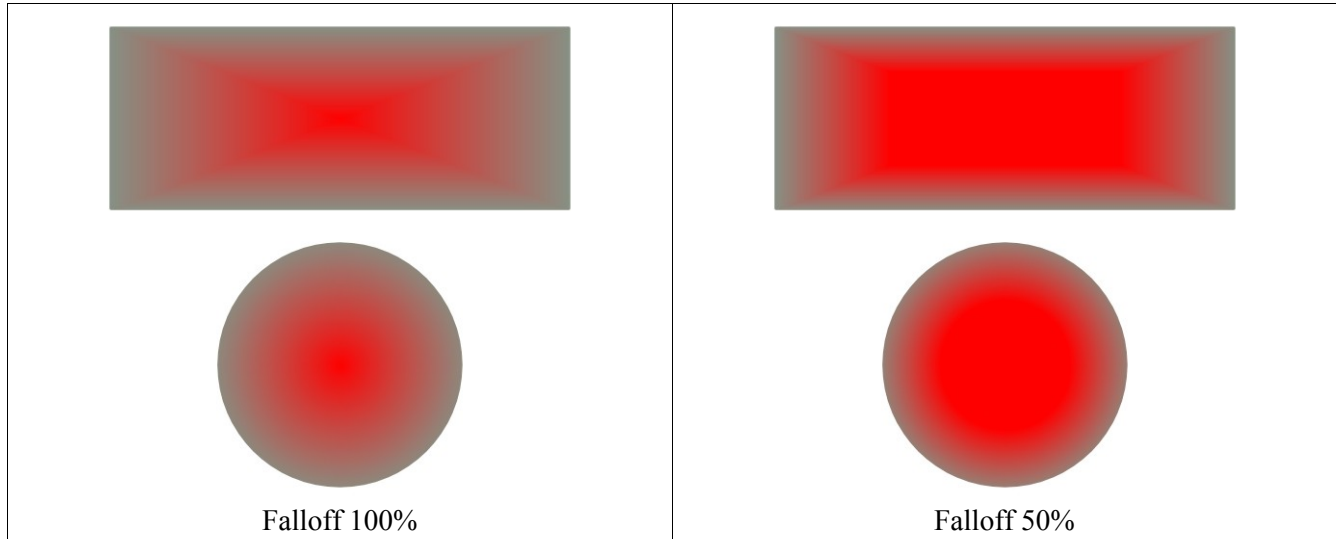
(For example the bone “>WGH\_Head” uses the weightmap called “Head”).

If no influence definition or polygonal data is present or a matching weightmap is missing, then the script aborts with a warning detailing the problem.

The sphere influence volume can only have uniform sizes. If the XYZ sizes are different, then the biggest value will be used for all axis. Negative scaling is treated as an absolute value.

Children should inherit their parent's transformations.

The influence falloff is set by the layer item's "Spline Patch Level" (`mesh.spatchSubdiv`). The falloff is a gradient from the center of the primitive toward the edge. The center point (start of the gradient) and the location of a vertex defines a vector which at some point intersects with the border of the volume. That will be the end of the falloff gradient for that vertex. The falloff value can be 1 to 100. "100" means that only vertices located in the center are getting 100% influence by the related bone. "50" means that only vertices on the outer 50% of the gradient have less than 100% influence.



### **Animation:**

Only the following animated channels are relevant to us: Position, Rotation and Scale.

Animated data in the scene can be cut into shorter clips, called *AnimationSequences*. "Pick Map" type vertex maps are used to store all info about AnimSeqs. The vertex maps will be generated by a utility script, the exporter only reads such data.

Pick maps describing AnimSeqs always have the following format:

"@SequenceName-StartFrame-EndFrame"

Animation sequences are exported as a single .PSA file.

The animation curves must be baked. During animation baking, the transformations of an item are sampled at each frame, and in the output data a keyframe is generated at every frame, using the linear curve type.

### **Rigid body mode:**

All meshes should be processed and saved as bone animations.

### **Soft body mode:**

Only bone definition meshes should be processed and saved as bone animations.