

# Blender Drakan Scripts

---

by BuXXe

## Inhalt

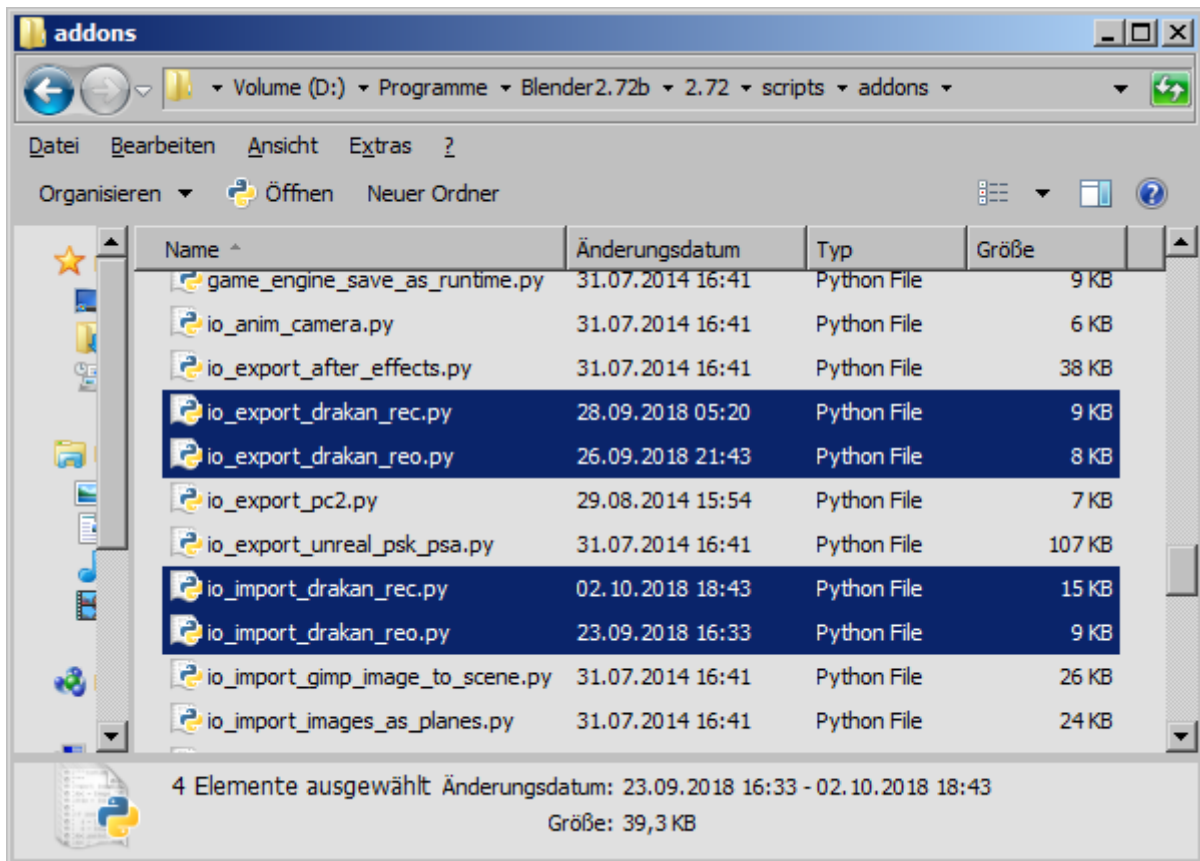
1. Install .....	2
2. Usage .....	4
2.1 Riot Engine Objects (.reo) .....	4
Import .....	4
Export .....	7
2.2 Riot Engine Characters (.rec) .....	11
Import .....	11
Export .....	12

## 1. Install

*Note: The scripts were tested for Blender 2.72b.*

Copy the 4 blender script files to the Blender addons folder.  
The folder should be located in:

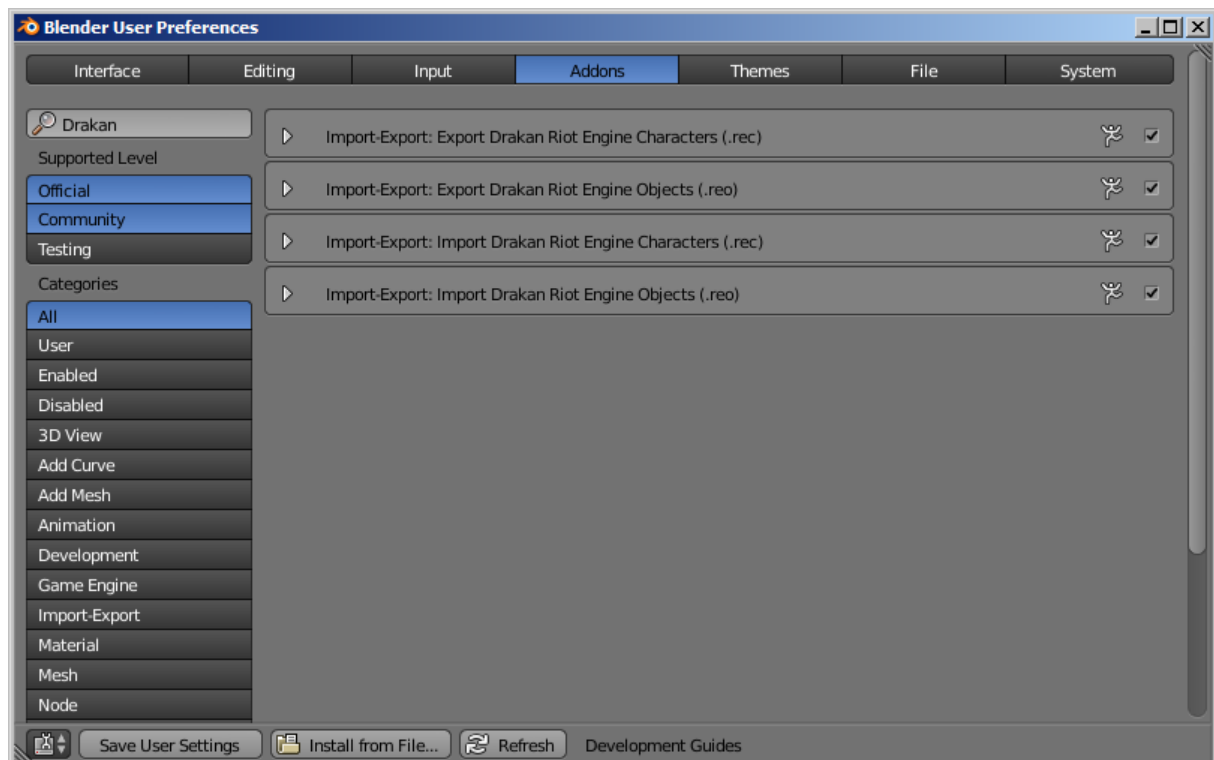
Blender2.72b / 2.72 / scripts / addons



Start Blender and open:

File > User Preferences (Shortcut: Ctrl Alt U)

In the “Addons” tab you will find a search bar. Type in “Drakan” to find the 4 scripts you just copied in the Addons folder.



By checking the checkboxes you activate each script. If you want to save this configuration for future starts of Blender, click on the “Save User Settings” button on the lower left.

The import / export entries are now available in:

File > Import > Drakan Riot Engine (Objects / Characters)

File > Export > Drakan Riot Engine (Objects / Characters)

## 2. Usage

In this section we will show the usage of each importer and exporter.

### 2.1 Riot Engine Objects (.reo)

#### Import

The importer has the following options

- ❖ **Import .reo materials**

Read the material entries of the .reo and create materials with these textures attached.

- ❖ **Import .reo UVs (Texture coordinates)**

Add the UV coordinates for each polygon.

- ❖ **Invert face normals**

The order in which the vertices are placed in a .reo polygon definition determines the direction in which this polygons normal points and therefor which side is outside and which is inside. By default this option is active because the .reo polygons have an inverted order in contrast to the standard definition of face normals.

- ❖ **Import bounding**

Imports the bounding information and creates bounding boxes or bounding spheres.

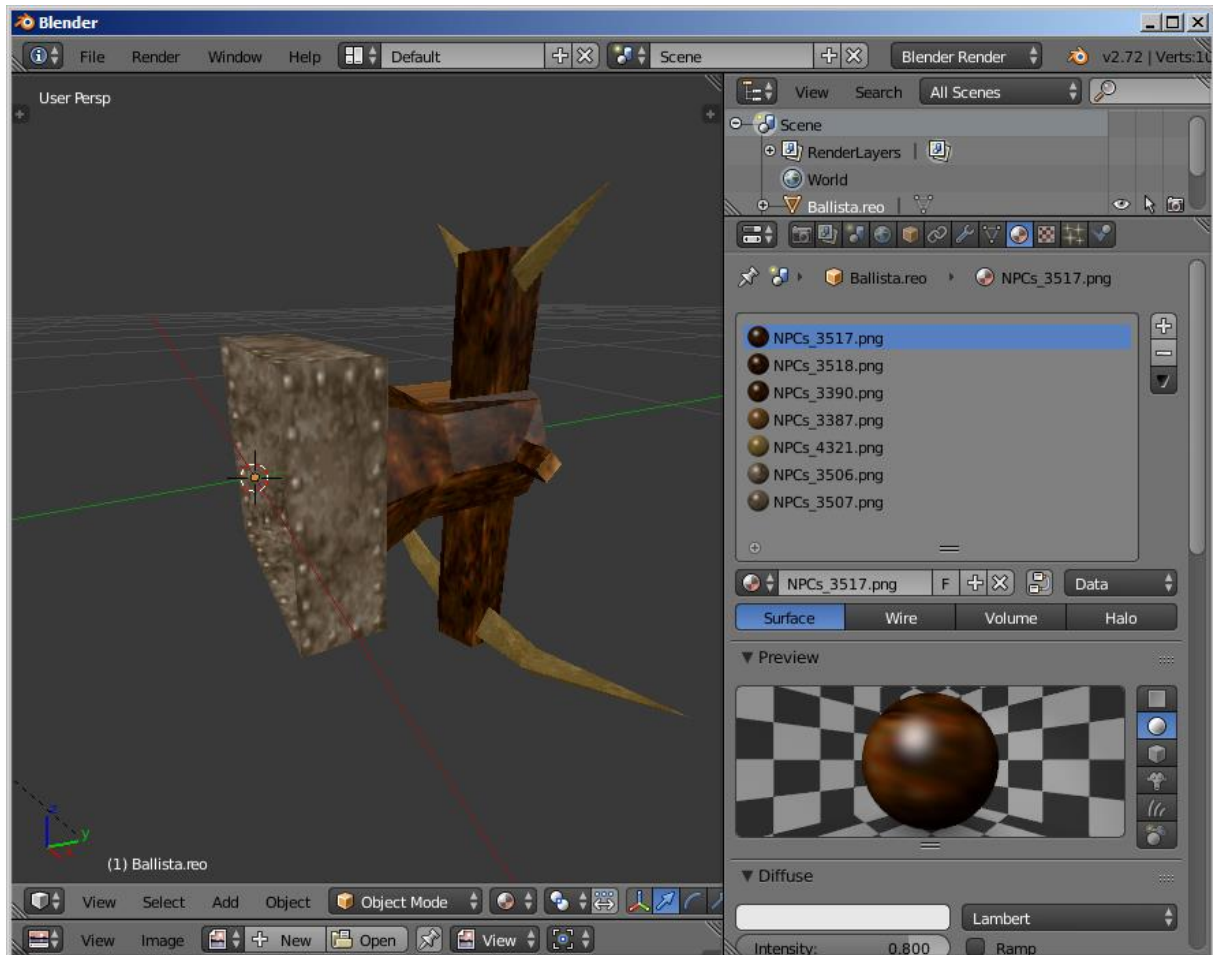
- **Hide bounding objects**

Some models have many bounding objects which would block the view and result in a confusing scene. If you check this option, the bounding objects will be imported but hidden initially. That way you can activate them manually to keep the overview.

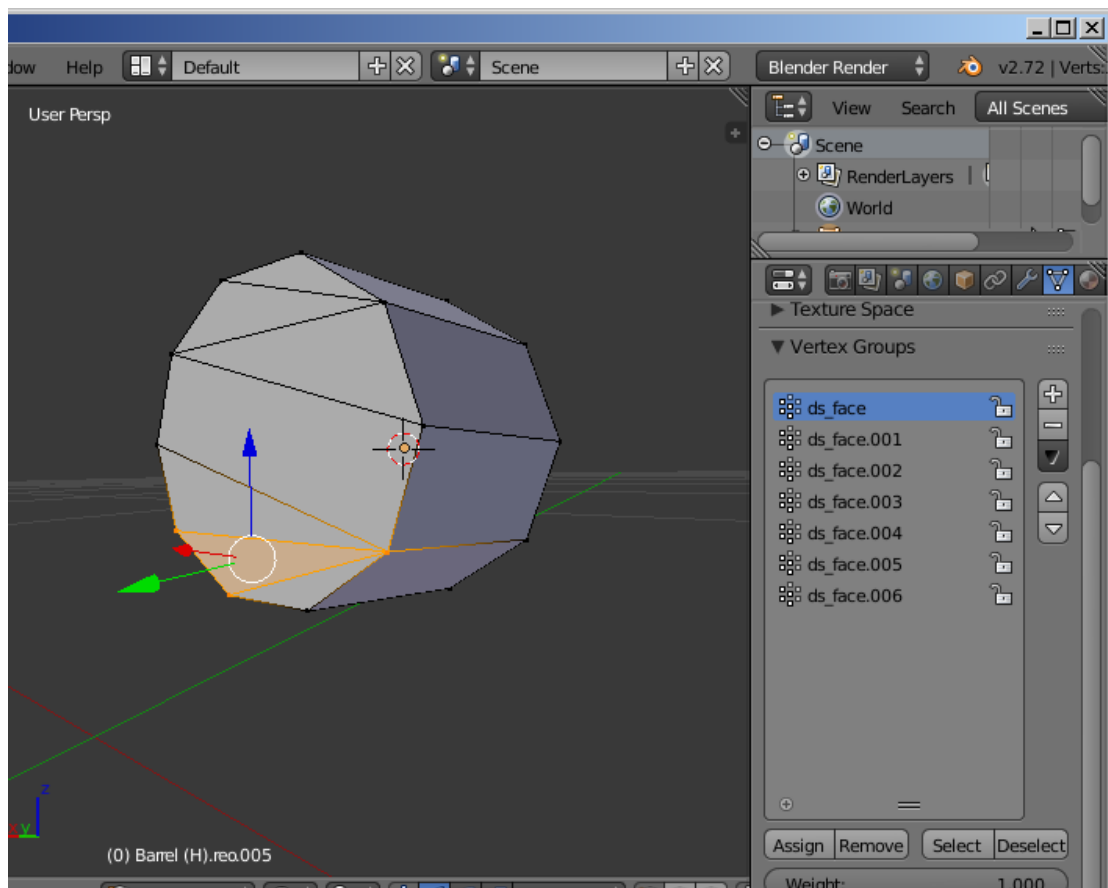
- ❖ **Omit invalid polygon definitions**

Some models have weird invalid polygon definitions (ex. Grimstone Mines CrystalShard01). They use vertices like 3,3,3 or 3,3,0. Although they import fine in blender, if you want to leave them out, check this option.

After importing a .reo with the default values you will have the model, bounding information and the materials in Blender. You need to change the viewport shading method to “Material” in order to display the model with its textures. Keep in mind that the models are pretty small and that the orientation is defined with the Y-axis as up.



The double sided polygons will have entries in the vertex groups. The convention is that these vertex groups have a name starting with “ds\_face”. Each group contains one set of vertices which defines the double sided polygon.



## Export

The exporter has the following options

### ❖ Export materials

Write the materials as material entries in the .reo. The exporter reads the material slots and uses the path of the first attached texture as .reo texture entry.

### ❖ Export texture filenames only

If checked, the texture in the .reo will be represented by the filename. That way, the texture files need to be in the same place as the .reo file to be correctly imported into the engine tools or using the Blender import scripts. Keep in mind that only bitmap files are supported by the Riot Engine tools.

### ❖ Export UVs (Texture coordinates)

If checked, the exporter will write the UV coordinates for each polygon to the .reo file. The exporter uses only the first UV mapping of the model.

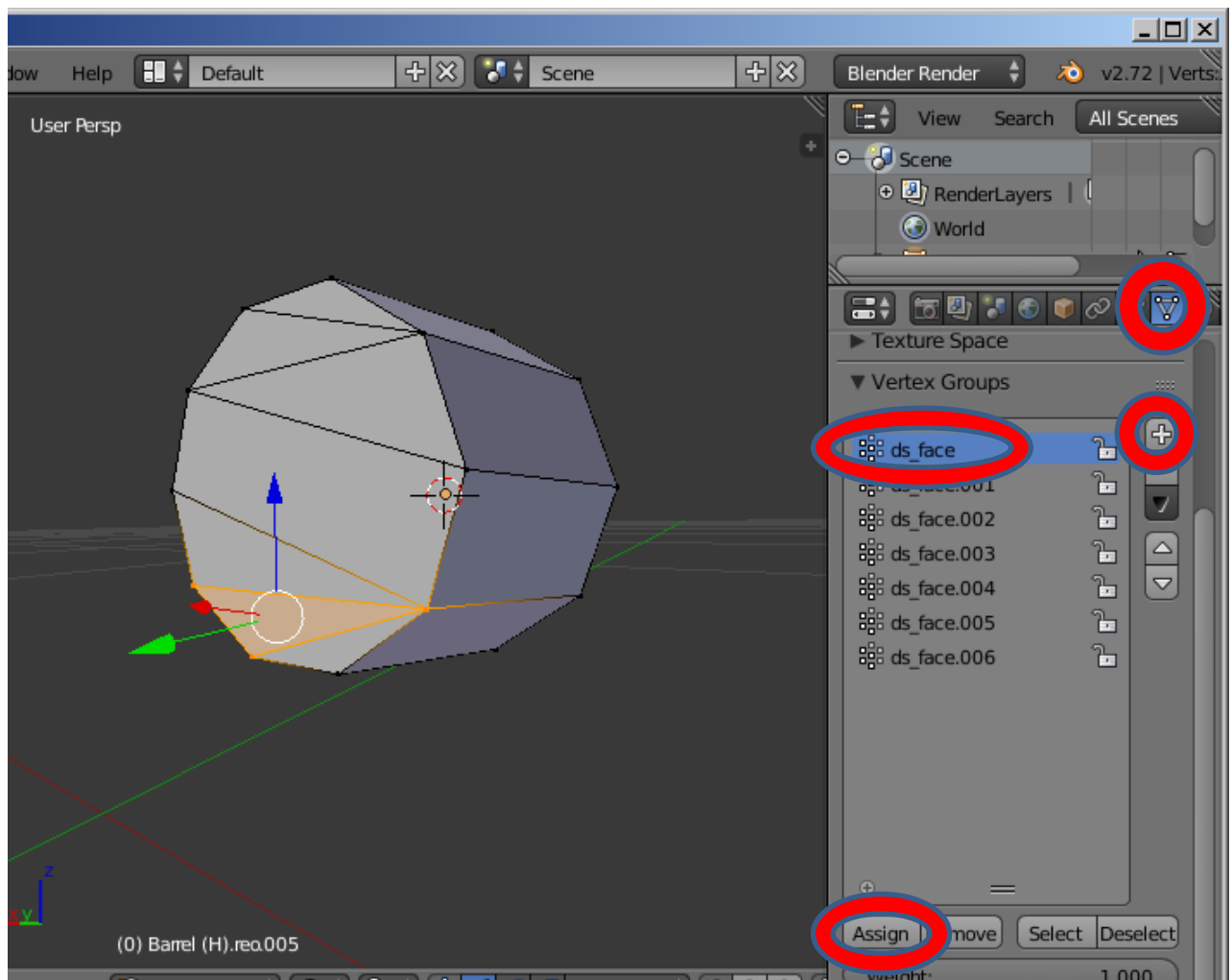
### ❖ Export bounding information

If checked, the exporter writes back the bounding information. For a detailed explanation of the bounding system, look into the extra section.

The exporter will export the currently active object. Make sure you are in “Object Mode”. You should apply all transformations (Ctrl A) and keep the UV coordinates in image bounds. The Riot Engine Modeler cannot handle texture wrapping and will display wrong textures if the coordinates are not “normalized”. The Level Editor displayed the textures correctly but it is best to prevent using out of bounds UV. The Riot Engine Modeler can also fix large UV coordinates.

### How to define double sided polygons:

1. Select the vertices of the polygon you want to be double sided in edit mode.
2. In the Data tab create a new vertex group by clicking the “+”. Set the group name to something starting with “ds\_face” by double clicking it.
3. Now, with the vertex group and the vertices selected, click “assign” to add these vertices to the group.
4. Each double sided polygon has its own vertex group!



### The bounding system

Reo models may have bounding information represented as bounding spheres or bounding boxes. A model can only have ONE type of these bounding information and not both.

The exporter plugin registers 2 new primitives:

- **RIOT BBox**

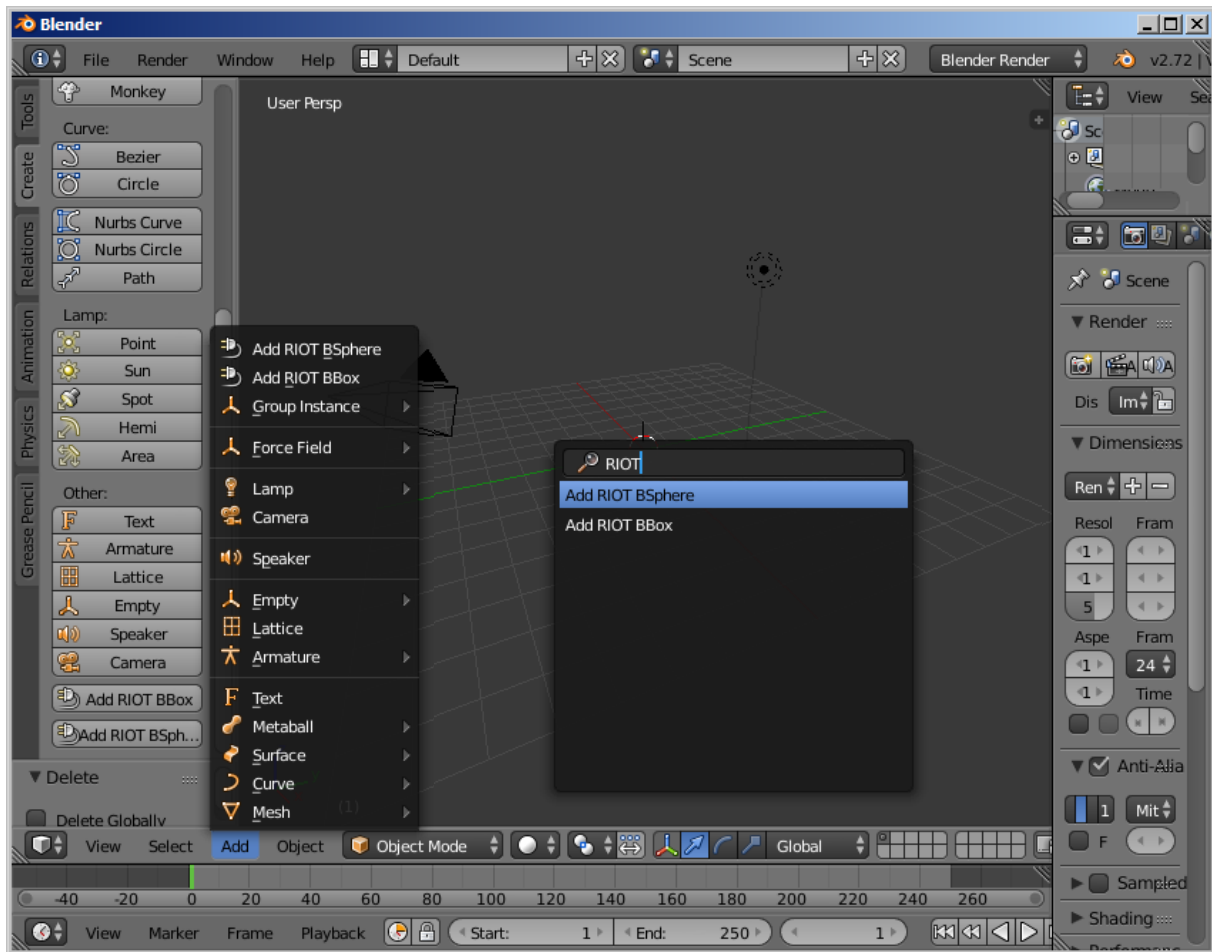
A special 1x1x1 box with its origin set to the 0-vertex of the cube.

- **RIOT BSphere**

A simple UV-sphere.

You can create these 2 primitives via the Create side menu, the Add menu or the Blender console (press Spacebar in 3d View and write "RIOT")





There are some rules connected to how these primitives may be manipulated.

You should only move (keyboard: G) or uniformly scale (keyboard: S) the spheres.

For bounding boxes, you can move (keyboard: G), rotate (keyboard: r) and scale (keyboard: S) them. Keep in mind that the origin is fixed to the 0-vertex NOT to the cubes center!

Further rules: Never apply the transformations to the bounding objects. Never edit the vertices in edit mode, stick to object mode. Do not manipulate the origin of the primitives.

Technically spoken: The boxes are written back using their world matrix. Spheres use the location and the scale[0] value.

Finally, the boxes and spheres follow a naming convention:

bbox:0\_c:0\_s:0\_pflag:0

bsphere:0\_c:0\_s:0

Each parameter is a "name:value" pair. These pairs are separated by "\_" entries. The first entry has to be either "bbox" or "bsphere" followed by the corresponding id. The "c" parameter holds the id of the first child of this node. "s" gives the id for the first sibling of this node. The id "0" is reserved as "no sibling / no child". Bounding boxes can also hold the "pflag" which indicates the "Build Polygon List" property of the RIOT modeler.

## 2.2 Riot Engine Characters (.rec)

### Import

The importer has the following options

#### ❖ Import .rec materials

Read the material entries of the .rec and create materials with these textures attached.

#### ❖ Import .rec UVs (Texture coordinates)

Add the UV coordinates for each polygon.

#### ❖ Invert face normals

The order in which the vertices are placed in a .rec polygon definition determines the direction in which this polygons normal points and therefor which side is outside and which is inside. By default this option is active because the .rec polygons have an inverted order in contrast to the standard definition of face normals.

#### ❖ Import double sided polygons

The .rec polygons may have an option which indicates that the polygon is double sided. This means there are 2 normals pointing in different directions. If this option is activated, the polygons are imported as 2 separate polygons, one with normal vertex order and the other with inverted vertex order. The Riot Engine Modeler allows the optimization of those polygons back to the double sided option definition.

#### ❖ Import skeleton information

Uses the hierarchy in the .rec to create a Blender armature. Furthermore, the model gets vertex groups which correspond to the hierarchy. These vertex groups have the associated vertex weights for each bone. The created armature is then defined as the parent of the model which gets an armature modifier attached.

The importer uses only the position of the nodes and does not use the rest of the matrix. Furthermore, it is known that the hierarchy for the Tiamat model (for example) uses many nodes with the same position which are then optimized out by the Editor. The importer cannot handle these models and the skeleton data will be corrupted! Optimized models should be no problem. In addition, it seems as if the associated vertices for each bone are correctly attached. Still, this is already the case for the Modeler itself and therefore, it seems to be correct behavior?

## Export

The exporter creates the materials and mesh entry with the model. Currently the skeleton and channels will not be exported! The exporter is used to have a good starting point.