Skip to content

Rotation Modes

Blender lets you define rotations in several ways. Each one of them has a series of advantages and drawbacks; there is no best rotation mode, as each one is suitable for specific cases.

In all of these modes, positive angle values mean counter-clockwise rotation direction, while negative values define clockwise rotation.

Though you can rotate elements using the global or local transform orientations, these axes are not suitable to define rotations, as the effect of each of the cannot be isolated from the other two.

Take, for instance, any three values for X, Y and Z rotation. Perform each one of these using global or local axes. Depending on the order in which you perform these, you will end up with different final orientations. So proper rotation coordinate systems are needed.

Euler Modes

The axes system used for performing Euler rotations is the so called Euler gimbal. A gimbal is a particular set of three axes. The special thing about this is that the axes have a hierarchical relationship between them: one of the axes is at the top of the hierarchy, and has one of the other two axes as its immediate child; at the same time, this child axis is the parent of the remaining axis, the one at the very bottom of the hierarchy.

Which axis is on top, which one in the middle and which at the bottom, depends on the particular Euler gimbal: there are six types of them, as there are s possible combinations: XYZ, XZY, YXZ, YZX, ZXY and ZYX Euler rotation modes. These modes are named using the letters of the axes in order, starting from the axis at the bottom of the hierarchy, and finishing with the one on top.

The main problem of these systems comes when they lose their relative perpendicularity. And this happens when the axis in the middle rotates, causing th axis at the bottom to rotate with it. It keeps getting worse when this bottom axis approaches 90° (or equivalent angles). In that case, it will remain aligned with the axis on top of the hierarchy. In that moment we have just lost one axis of rotation. This can cause discontinuous interpolations when animating. This particular loss of axis is known as the “gimbal lock”.

Hint

The actual configuration of the gimbal axes can be seen in the 3D Viewport by enabling the *Rotate* object gizmo and setting it to *Gimbal* (from the gizmos button in the header). At the same time, rotation mode should be set to any of the Euler modes for the active object.

Now you can perform a rotation around the axis in the middle (e.g. in *XYZ Euler* mode that is the Y axis), and see how easy it is to end up having a gimbal with just two axes. In the specific case of the *XYZ Euler* mode with gimbal lock, a rotation around the X axis will have the same effect as rotatin around the Z axis, meaning, in practice, that no X axis rotations can be performed.

One advantage of this mode is that animation curves are easy to understand and edit. However, special attention must be done when the middle axis approaches values close to 90° (or equivalent angles).

Axis Angle Mode

This mode lets us define an axis (X, Y, Z) and a rotation angle (W) around that axis.

If we define the rotation using interactive rotations (with the rotation gizmo), the values of X, Y and Z will not exceed 1.0 in absolute value, and W will be comprised between 0 and 180 degrees.

If you wish to define rotations above 180° (e.g. to define multiple revolutions), you will need to edit the W value directly, but as soon as you perform an interactive rotation, that value will be adjusted again. Same thing goes for axis values.

This system is suitable for elements revolving around a fixed axis, or to animate one of the elements at a time (either the axis or the angle). The problem might come when animating (interpolating) both components at the same time: axis and angle. The resulting effect might not be as expected.

The *Gimbal* gizmo in this rotation mode shows a set of three orthogonal axes in which the Z axis goes along the defined rotation axis, i.e. it points toward the direction defined by the (X, Y, Z) point.

The axis-angle system is free from gimbal lock, but animation curves in this mode are not intuitive at all when animating axis and angle at the same time, in which case they are difficult to understand and edit.

Quaternion Mode

In this mode, rotations are also defined by four values (X, Y, Z and W). X, Y and Z also define an axis, and W an angle, but it does it quite differently from axis-angle. The important thing here is the relation between all four values.

To describe it in an intuitive way, let's take the effect of the X coordinate: what it does is to rotate the element around the X axis up to 180 degrees. The same goes for Y and Z. The effect of W is to avoid those rotations and leave the element with zero rotation. The final orientation is a combination of thes four effects.

As the relation between components is what defines the final orientation, multiplying or dividing all four numbers by a constant value will yield the very sa rotation.

This mode is ideal for interpolating between **any** pair of orientations. It doesn't suffer from gimbal lock or any interpolation undesired effect. The only drawback is that you cannot interpolate between two orientations that are at a distance greater than than 180°, as the animation will take the shortest pat between them. Thus to animate a revolving element you must set up many intermediate keyframes, 180° from each other at most.

The *Gimbal* gizmo in this mode is equivalent to the *Local* one, and doesn't have any special meaning.

The animation curves in this mode are not intuitive, so they are also difficult to understand and edit.

More about Quaternions

This section is not really useful for 3D artists, but it can be suitable for the curious or the scientist.

Quaternions are a number system extending the complex numbers. They represent a four component vector, whose components are called, in Blender, X Y, Z and W. When rotating interactively in quaternion mode, the so called norm (length) of the quaternion will remain constant. By definition, the norm o quaternion equals 1.0 (that's a **normalized** quaternion). When you select the quaternion mode in Blender, the XYZW components describe a normalize quaternion.

> Note
>
> The norm of a quaternion q is defined mathematically as:
>
> $$\lVert q \rVert = \sqrt{X^2 + Y^2 + Z^2 + W^2}$$

However, if one of the quaternion components is locked during the interactive transformation using the proper lock button, the norm will not remain unchanged, as that blocked component will not be able to adjust itself to keep the unit norm.

> Hint
>
> Interactive rotations with the gizmo don't change the norm of the current quaternion. Editing a single XYZW component individually you can change the norm. To make the norm 1.0 again you can switch to any rotation mode and back again into quaternion.

The rotation components of a quaternion keep a tight relation with those of axis-angle. To find a correspondence, first of all we must deal with the normalized version of the quaternion, that is, one whose norm equals 1.0. To normalize a quaternion, just divide each one of its components by its norm. As we have seen before, dividing all four values by the same number gives the same orientation.

Once we have calculated the components of the normalized quaternion, the relation with the axis-angle components is as follows:

- X, Y and Z mean exactly the same as in axis-angle: they just define an axis around which the rotation takes place.
- W can be used to retrieve the actual rotation around the defined angle. The following formula applies (provided that the *quaternion is normalized*): $W = \cos(\frac{a}{2})$, where a is actually the rotation angle we are looking for. That is: $a = 2 \arccos{W}$.

Other Considerations

In axis-angle and quaternion modes we can lock rotations in interactive modes in a per component basis, instead of doing it by axis. To do so we can activate this locking ability using the lock buttons next to the corresponding *Rotation* transform buttons.

Regarding rotation animations, all keyframes must be defined in the same rotation mode, which must be the selected rotation mode for the object throughout the entire animation.

Copyright © : This page is licensed under a CC-BY-SA 4.0 Int. License
Made with Furo

View Source
View Translation
Report issue on this page