Animieren von Steuerflächen bei Flugzeugen

Deutsch Image not found or type unknown

Deutsch Image not found or type unknown

0. Baue die Steuerflächen und lege den Ursprungspunkt auf die gewünschte Drehachse. Wo genau auf dieser Achse der Punkt liegt, ist egal - er kann an einer beliebigen Stelle liegen; ich lege ihn bei meinen Modellen immer gerne ans rumpfseitige innere Ende der Fläche.

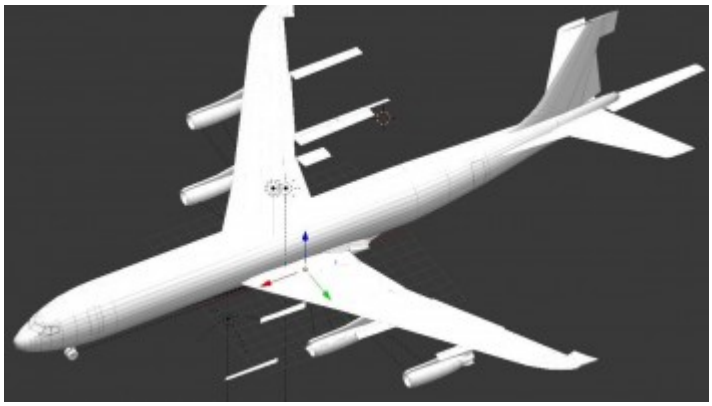
Hier kann man ein wenig spielen. Bei Rudern ist es vernünftig, daß die Drehachse genau durch das Teil verläuft. Bei Landeklappen ist es oft sinnvoll, sie auch ein wenig nach hinten ausfahren zu lassen: das erreicht man, indem man die Drehachse deutlich unter das Teil legt. Ein guter Anfang sind 20-50cm.

1. Dreh alle Steuerflächen AUSSER dem Seitenruder so, daß ihre gewünschten Drehachsen parallel zur x-Achse liegen.
Um die X-Achse drehst du sie einfach so, daß sie "flach" liegen. Um die y-Achse drehst du sie von oben betrachtet für alles, was hinten am Profil liegt, **im** Uhrzeigersinn, für alles, was vorne dran liegt und nach unten klappen soll (Vorflügel), **gegen** den Uhrzeigersinn.

2. Schreib dir für jede Steuerfläche auf, um wie viel Grad du sie aus ihrer Ursprungsposition um beide Achsen verdreht hast - 1/10 Grad reicht als Genauigkeit.

3. "Apply Rot/Scale" auf alle so zurechtgedrehten Flächen.

4. Exportieren. Wenn du es richtig gemacht hast, dann sieht das Modell im Model Viever so aus wie die 707 hier.



5. Öffne die .mdl von deinem Modell.

6. Schreibe gleich als erste beide Zeilen vor alles andere das hier:

Code

```
local vec3 = require "vec3"
local transf = require "transf"
```

7. Such dir die Blöcke raus, in denen die zu animierenden Teile definiert werden. Die sehen so aus:

Code

```
{
    id = "vehicle/(Typ)/(Name)/(Name des Teils).m
  transf = {1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, x, y, z, 1.0,},
                                                                    type
},
```

8. Ersetze den Block "transf = {...", also die ganze Zeile 3 im obigen Code mit folgender Zeile:

Code

```
transf = transf.rotZYXTransl(transf.degToRad(Alpha, Beta, Gamma), vec3.new(x, y, z, 1.0)),
```

Die Variablen erhalten folgende Zahlenwerte:
x, y, z: die Zahlenwerte x, y, z aus der ersetzten Zeile "transf = {..." von oben,
Alpha, Beta, Gamma: die gemerkten Werte, um die du die Steuerflächen um die Z-, Y- und X- Achse gedreht hast; logischerweise mit umgekehrtem Vorzeichen, um die Flächen wieder zurückzudrehen.

9. Speichern und im Model Viewer nachsehen, ob die Fläche richtig sitzt :) Image not found or type unknown

Wenn die Flächen dann in der .mdl richtig angesprochen werden, dann sollten sie im TPF dann auch animiert sein.

Oben habe ich die Ausnahme Seitenruder schon angesprochen. Der Unterschied zu allen anderen Rudern ist, daß dessen Drehachse parallel zur Z-Achse, also zur Vertikalen, gedreht werden muß. Der entsprechende Winkel kommt dann so in die "transf"-Zeile:

Code

```
transf = transf.rotZYXTransl(transf.degToRad(0,-Winkel,0), vec3.new(x, y, z, 1.0)),
```

Es ist möglich, die Steuerflächen in der .mdl mehrfach anzusprechen und verschiedene Bewegungen zu überlagern. Gelegentlich wird man zum Beispiel das Höhen- und Querruder in einer Steuerfläche kombinieren wollen: die Concorde, die Tu-144 und viele Militärflugzeuge verwenden diese technische Lösung. Das sieht in der .mdl so aus:

Code

```
configs                                        =                                       {
  {



      },

    },



      },

    },


                                                                  6,        7,        8,        9,
      },

    },
  },
},
```

Alles anzeigen

Wie man sieht, sind die Teile 6 bis 9 als linkes Querruder und die Teile 10 bis 13 als rechtes definiert. Die Teile sind dann im nächsten Block gleich nochmal als Höhenruder angesprochen. Im Spiel werden dann die Ausschläge für beide Ruder aufsummiert und die Steuerflächen schlagen entsprechend aus.

Man kann auch verschiedene Ruder verschieden stark ausschlagen lassen. Wenn man zum Beispiel ein inneres Hochgeschwindigkeitsquerruder nur um 5 Grad und das äußere um 10° ausschlagen lassen möchte, dann setze man den Ausschlagwinkel auf 5° und spreche das äußere Querruder zweimal an (11,11,...). Für jeden Aufruf bekommt die Steuerfläche den bemessenen Ausschlag zugewiesen.

Image not found or type unknown

English

A short tutorial on how to animate control surfaces on aircraft.

0. Build the control surfaces and place their origin somewhere on the desired axis. The exact position on the axis does not matter; I personally like to put the origin on the surfaces end facing the fuselage, but this is only personal preference and not obligatory at all.

Obviously, primary controls will have their axes lie somewhere close to the front of and intersecting the part. With flaps, things look different: it is often desirable to have them extend a bit backward from the wing while they are moved. To achieve this, place the origin somewhere below the part. A good starting point is 20-50cm lower; try to see if you like the effect and readjust as required.

1. Turn all the surfaces EXCEPT for the rudder to get their axes lying in parallel to the grand X axis.

About the X axis you can just turn them to get them to lie flat.

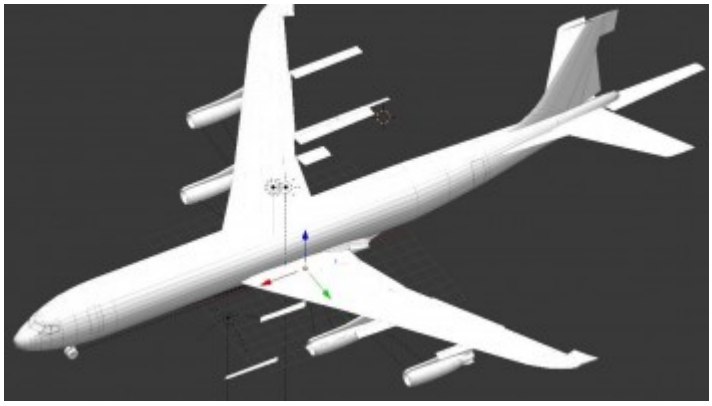
About the Y axis it depends on the part. Parts situated to the rear of the airfoil generally need to be moved clockwise viewed from top. Slats typically need to be turned counterclockwise. Kruger flaps that move in the opposite direction to normal slats will be turned clockwise obviously.

2.

Take note of the precise angle You applied to the parts about all relevant axes. A tenth of a degree is typically sufficient in accuracy. A sheet of paper and a pen is recommended.

3.
"Apply Rot/Scale" to all the surfaces to be animated.

4.
Export the surfaces to the relevant directories. If done right, the model should look similar to the 707 in the Model Viewer:



5.
Open your models .mdl file.

6.
RIght in front of everything else, paste the following two lines:

Code

```
local vec3 = require "vec3"
local transf = require "transf"
```

7.
Look up the lines defining the parts You would like to be animated. They look like this and are usually found in the 2nd third of the file:

Code

```
{
                          id            =            "vehicle/(Type)/(Name)/(Part
 transf = {1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, x, y, z, 1.0,},
                                                                                 type
},
```

8.
Replace the line "transf ={" etc. with the following:

Code

```
transf = transf.rotZYXTransl(transf.degToRad(Alpha, Beta, Gamma), vec3.new(x, y, z, 1.0)),
```

The variables will take the following values:
Alpha, Beta, Gamma: the angles You have noted down for the part previously, obviously with negative value (you need to turn the parts back now), about the Z, Y and X axes.
x, y, z: the coordinates of the part in the model as taken from line 3 under point 7. Just copy and paste those.

9.
Save the file and check the Model Viewer to see if the parts are in the right position again.

If you have adressed those parts properly further down in the .mdl (or via the Blender export plugin), they ought to be animated now.

Earlier, I have named the rudder as the exception to these guidelines. This surface is not turned in parallel to the X, but in parallel to the Z axis, or the Vertical, if you will. Again, note the angle and apply Rot/Scale. The proper line for the rudder looks as follows in the .mdl:

Code

```
transf = transf.rotZYXTransl(transf.degToRad(0,-angle,0), vec3.new(x, y, z, 1.0)),
```

Another note: It may become desirable to apply various animations to one surface. On occasion, an elevator might double up as an aileron (various military types, Tu-144, Concorde), or an aileron might be deflected down together with the flaps. This is possible in the game as well. Just call up the parts multiple times in the .mdl:

Code

```
configs                                     =                              {
  {


      },

    },


      },

    },

                                                        6,      7,      8,      9,
      },

    },
  },
},
```

Alles anzeigen

Note how parts 6 to 9 are nominated as left aileron and parts 10 to 13 as right aileron. Then, they are all called up again as elevators in the next block. The game will sum up both deflections and apply the total to the surface at any time. It is also possible to have one part deflect more than another one: just set the deflection angle to the value appropriate for the less moving part and adress the part you want to deflect more strongly twice or thrice. The parts total deflection will be n times the maximum angle, with n being the number of calls. This is useful for high and low speed ailerons for example.