Türen animieren

Table Of Contents

- 1 Voraussetzungen
- 2 Folgende Dateien müssen angefaßt werden
- 3 Fangen wir mit den Türen an
- 4 Schauen wir uns die Events an
- 5 Animation nur auf der Bahnsteigseite
- 6 Zusammenfassung

Türen zu animieren ist leichter als man vermuten würde. Die Bewegungsrichtung ist nicht auf den Zug bezogen, sondern auf die Tür selbst und deren Origin. Das bedeutet, daß eine linke Tür immer nach links rutscht, egal ob sie an backbord ist und nach vorne rutscht oder an steuerbord und nach hinten. Es ist immer der gleiche Wert und die gleiche Richtung, was die Skripterei stark vereinfacht. Also legen wir los!

Ich benutze meinen Mod der BR 270 als Beispiel für Code und Pfade. Ihr müßt Eure Pfade entsprechend anpassen und auch die Koordinaten und Indices der Meshes sowie der Weg der Animation ist bei Euch garantiert anders. Zu guter letzt decke ich hier lediglich Schiebetüren ab. Eine Animation mit Rotation ist ebenso einfach zu steuern, erfordert jedoch mit Sicherheit etwas mehr probiererei, bis man die richtigen Werte gefunden hat.

Ich weiß nicht, ob es einen besseren Weg gibt, da dies mein erster Mod ist, aber ich hab die genauen Wege und Richtungen durch Probieren herausgefunden, indem ich die Skripte geändert hab, das Spiel geladen, geschaut wie es aussieht, Spiel geschlossen, Skript angepaßt und das ganze von vorn.

1 Voraussetzungen

Bevor wir überhaupt anfangen uns Gedanken über Türanimationen zu machen, muß klar sein, daß dafür gewisse Grundvoraussetzungen notwendig sind. Türen können nur dann animiert werden, wenn sie ein eigenes Mesh besitzen. Das bedeutet, beim Modellieren wurden die Türen als eigene Objekte erstellt und am oder im Fahrzeug platziert. Ist dies nicht der Fall und Eure Türen sind fester Bestandteil des Zug-Meshes, könnt Ihr hier zu lesen aufhören, denn diese Türen sind nicht animierbar. Nun, vielleicht lest Ihr trotzdem weiter, denn wenn Ihr jetzt Euer Modell ändert und die Türen einzeln erstellt, solltet Ihr auf Origins und Normals achten. Dies wird weiter unten noch erklärt und könnte Euch im Vorfeld eine Menge Ärger ersparen.

2 Folgende Dateien müssen angefaßt werden

Datei	Pfad	Funktion
mdl-Datei für den Zug	\res\models**model**\vehicle\train\br270-tw.**mdl**	Erstellen der Events für die Türanimation
msh-Datei für die linke Tür	\res\models**mesh**\vehicle\train\br270-tw\br270_door-left.**msh**	Erstellen des Skriptes für die Animation der linken Tür
msh-Datei für die rechte Tür	\res\models**mesh**\vehicle\train\br270-tw\br270_door-right.**msh**	Erstellen des Skriptes für die Animation der rechten Tür

3 Fangen wir mit den Türen an

In der Mesh-Datei (.msh) für die linke Tür befindet sich eine leere Node "animations":

Code

```
1. animations = {
2. },
```

Hier gehört der Code hinein, der beschreibt, wie sich die Tür bewegen soll. Wir benötigen diese Bewegung nur einmal. Sie wird sowohl für das Öffnen als auch das Schließen verwendet und einfach nur rückwärts "abgespielt".

So sieht mein Code aus, die Erklärung folgt danach:

Code

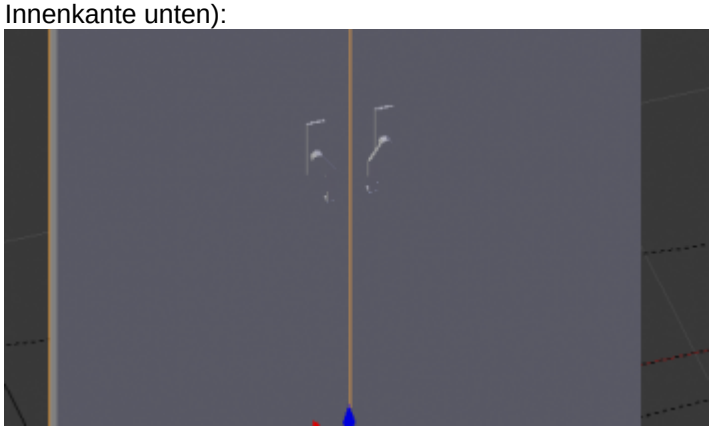
```
1. animations = {
2. opend1 = {
3. type = "KEYFRAME",
4. params = {
5. origin = { 0, 0, 0, },
6. keyframes = {
7. {
8. time = 0,
9. rot = { 0, 0, 0 },
10. transl = { 0, 0, 0 }
11. }, {
12. time = 800,
13. rot = { 0, 0, 0 },
14. transl = { 0, 0.28, 0 }
15. }
16. }
17. }
18. },
19. },
```

Display More

opend1 ist der Name des Datenblocks. Dieser ist an diese Tür gebunden. Für die andere Tür müssen wir ebenfalls derlei Animations-Werte zuweisen, allerdings mit einem anderen Namen (opend2).

origin bestimmt den Ausgangspunkt der Animation bezogen auf die Tür. Die drei werte sind jeweils die X-, Y- und Z-Koordinaten von denen ausgehend sich die Tür bewegen soll. Ich hab meinen Origin (in Blender) instinktiv an die Innenkante der Tür gelegt, schon allein um sie am Modell gut ausrichten zu können. Dies hat sich als sehr nützlich erwiesen und so brauchte ich am *origin* hier nichts zu ändern. Sollte das bei Euch nicht der Fall sein, empfehle ich, die Origins an Eurem Modell in Blender neu zu setzen. Ihr könnt ihn natürlich auch hier bei *origin* eintragen, aber das wird mit Sicherheit viel Frickelei und Probiererei. Auch die Ausrichtung der Tür (Normals) sollte exakt sein, da man dann nur mit einer Koordinate arbeiten muß, aber dazu kommen wir später.

Im folgenden Screensot seht Ihr, wo mein Origin an der linken Tür liegt (der orangene Punkt an der Innenkante unten):



Die *keyframes* bestimmen die notwendigen Schritte, um die Tür zu öffnen/schließen. bei einer Schiebetür reichen zwei Keyframes aus, einen für den geschlossenen Zustand und einen für den offenen. Bei komplexeren Animation, z.B. wenn die Tür erst nach außen kippt und dann zur Seite rutscht, sind entsprechend mehr Keyframes möglich: einer für den geschlossenen Zustand, einer für den ausgeklappten Zustand und ein dritter für den verschobenen Zustand.

Ein *keyframe* hat drei Eigenschaften:
- *time* ist der Zeitpunkt, an dem der Keyframe durchlaufen wird. Im geschlossenen zustand ist das offensichtlich 0 und im geöffneten Zustand die Zeit, die verstrichen sein soll, bis die Tür vollständig geöffnet ist. Ich vermute, daß es sich hier um Millisekunden handelt, weiß es aber nicht. In meinem Beispiel hat sich die Tür nach 800 Millisekunden (?) vollständig aufgeschoben. Das Ändern dieses Wertes bestimmt demnach die Zeit, die die Tür zum öffnen benötigt.
- *rot* bestimmt die Rotation der Tür in Grad auf jede Raumachse (X, Y und Z)
- *transl* bestimmt die Bewegung der Tür auf einer oder mehrerer der drei Raumachsen (X, Y und Z) in Metern. In meinem Beispiel wird die Tür entlang der Y-Achse um 0,28 Meter verschoben. Wenn Ihr nochmal auf meinen Screenshot oben schaut, seht Ihr, daß meine Y-Achse exakt nach vorn zeigt, d.h. die Tür wird in diese Richtung verschoben. Auf der anderen Seite des Wagens ist die Tür um 180° gedreht, das bedeutet, deren Y-Achse zeigt zum Heck des Zuges und wird somit ebenfalls um den gleichen Weg in die richtige

Richtung verschoben. Die Y-Achse meiner rechten Tür zeigt in die selbe Richtung, was bedeutet, daß ich dort einen negativen Wert (-0.28) eintragen muß, damit sich die Tür nach rechts verschiebt.

Den genauen Weg, den meine Tür zurücklegen sollte wußte ich natürlich nicht aus dem Kopf. Meine Vorlage verwendete 0,5 Meter und das hab ich zunächst übernommen, im Spiel ausprobiert, dann auf 0,25 Meter herabgesetzt, ausprobiert, dann wieder auf 0,26 Meter heraufgesetzt, ausprobiert und schließlich nochmal auf 0,27 Meter erhöht, bis es perfekt war. So hab ich mich herantasten müssen. Ich hab auch zuerst nur mit den vier linken Türen an backbord angefangen, um zu verstehen, wie es funktioniert und was pasiert, und das ganze dann später auf die restlichen Türen übertragen.

Das gleiche muß natürlich analog für die rechte Tür gemacht werden, denkt aber daran es dort nicht "opend1" zu nennen, sondern "opend2". Nun ist die Animation selbst fertig.

4 Schauen wir uns die Events an

In der Modell-Datei (.mdl) müssen wir für jeden LOD entsprechende Events erstellen. Da ich (bisher) nur einen LOD habe, ist dies (noch) übersichtlich:

Code

```
1. lods = {
2. {
3. animations = {
4. },
5. children = {
6. {
7. id = "vehicle/train/br270-tw/br270-tw_body.msh",
8. transf = {
9. 0.0, 0.49406, 0.0, 0.0, -0.49406, 0.0, 0.0, 0.0, 0.0, 0.0, 0.49406, 0.0, -0.0, -0.0, 0.0, 1.0,
10. },
11. type = "MESH",
12. }, {
13. id = "vehicle/train/br270-tw_lod_0_bogie-front.grp",
14. transf = {
15. 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 5.77282, 0.0, 0.0, 1.0,
16. },
17. type = "GROUP",
18. }, {
19. id = "vehicle/train/br270-tw_lod_0_bogie-rear.grp",
20. transf = {
21. 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, -6.3899, 0.0, 0.0, 1.0,
22. },
23. type = "GROUP",
24. }, {
25. id = "vehicle/train/br270-tw/br270_door-left.msh",
26. transf = {
```

```
0.0, -1.86898, 0.0, 0.0, 1.86898, 0.0, 0.0, 0.0, 0.0, 0.0, 1.86898, 0.0, -7.25102, 1.4637, 1.11867, 1.0,
},
type = "MESH",
},
<---- snipp ---->
```

Display More

Hier schneide ich mal ab. Mein erster LOD besitzt wieder eine "animations"-Node, aber diese ignorieren wir dieses mal völlig. Stattdessen schauen wir uns die *children* an. Dies sind alle Meshes und Groups, die zu unserem Modell gehören. Als erstes der Body (br270-tw_body.msh), darunter das vordere (br270-tw_lod_0_bogie-front.grp) und hintere Drehgestell (br270-tw_lod_0_bogie-rear.grp). Danach kommt die erste linke Tür (br270_door-left.msh). Es folgen noch 7 weitere linke Türen, 8 rechte Türen, die Scharfenbergkupplung und die Kurzkupplung, uns interessieren aber nur die Türen. Denn wir müssen nun "zählen", welchen Index jede Tür hat. Die Indices sind nicht nullbasiert, sondern fangen mit 1 an. Mein Body hat also den Index 1, weil er das erste Child ist. Die Drehgestelle haben entsprechend den Index 2 und 3, meine erste Tür den Index 4. Die rechten Türen starten bei Index 12 usw. Diese müssen wir uns merken, aufschreiben oder einfach später nochmal durchzählen, denn nun erstellen wir eine neue Node "events" direkt unterhalb der Children:

Code

```
animations = {
},
children = { <---- snipp ----> },
events = {
close_all_doors = {
},
open_all_doors = {
},
},
```

In der "events"-Node erstellen wir noch die Nodes "close_all_doors" und "open_all_doors". Dort muß nun für jede Tür (bezeichnet durch den zuvor ermittelten Index) die richtige Animation festgelegt werden und ob sie vorwärts oder rückwärts abgespielt wird:

Code

```
events = {
close_all_doors = {
[4] = {
forward = false,
name = "opend1",
},
[5] = {
forward = false,
name = "opend1",
```

```
10. },
11. <---- snipp ---->
```

Display More

Um also meine ersten beiden linken Türen (Index 4 und 5) zu schließen, muß unsere zuvor im Mesh erstellte Animation "opend1" rückwärts abgespielt werden: "forward = false". Dies muß ich natürlich für alle 8 linke Türen machen. Andere Modder haben Ihre Türen jeweils zu größeren Objekten zusammengefügt, so daß sie nur ein Mesh verschieben müssen, das aus mehreren Türen besteht. Das wird oft gemacht, da dann diese Skripte sehr viel übersichtlicher werden. Mein Mod hat allerdings so viele Polygone, daß ich Ressourcen sparen muß und so eine einzige Tür 8 mal wiederverwenden kann. Für meine ersten beiden rechten Türen mit dem Index 12 und 13 benutzen entsprechend die in ihrem Mesh definierten Werte für "opend2", genau wie meine restlichen rechten Türen, die ich hier mal nicht mit anzeige:

Code

```
1. [12] = {
2. forward = false,
3. name = "opend2",
4. },
5. [13] = {
6. forward = false,
7. name = "opend2",
8. },
```

Das ganze machen wir jetzt nochmal für das Öffnen der Türen in der Node "open_all_doors", doch diesmal lassen wir die Animation vorwärts ablaufen (und formatieren es etwas platzsparender):

Code

```
1. open_all_doors = {
2. [4] = { forward = true, name = "opend1", },
3. [5] = { forward = true, name = "opend1", },
4. <---- snipp ---->
5. [12] = { forward = true, name = "opend2", },
6. [13] = { forward = true, name = "opend2", },
7. <---- snipp ---->
8. },
```

Das sind nun die Animationen für alle Türen.

5 Animation nur auf der Bahnsteigseite

Wollen wir nur die Türen auf der Bahnsteigseite öffnen, müssen wir dem Spiel sagen, welche Türen an Backbord bzw. Steuerbord sind. Dazu erstellen wir die gleichen Events wie zuvor, außer daß wir nur die Indices verwenden, die für die Türen auf der jeweiligen Wagenseite stehen. Dazu fügen wir vier weitere Nodes in der "event"-Node ein:

Code

```
1. animations = {
2. },
3. children = {
4. <---- snipp ---->
5. },
6. events = {
7. close_all_doors = { <---- snipp ----> },
8. open_all_doors = { <---- snipp ----> },
9. close_doors_left = {
10. },
11. open_doors_left = {
12. },
13. close_doors_right = {
14. },
15. open_doors_right = {
16. },
17. },
18. },
```

Display More

Es empfiehlt sich, nun die Einträge für "close_all_doors" und "open_all_doors" zu kopieren und nur die Indices zu löschen, die sich nicht auf der linken bzw. rechten Wagenseite befinden.

Hier mal der komplette Code für die Türanimation-Events der BR 270 mit etwas platzsparenderer Formatierung:

Code

```
1. events = {
2. close_all_doors = {
3. [6] = { forward = false, name = "opend1", },
4. [7] = { forward = false, name = "opend1", },
5. [8] = { forward = false, name = "opend1", },
6. [9] = { forward = false, name = "opend1", },
7. [10] = { forward = false, name = "opend1", },
8. [11] = { forward = false, name = "opend1", },
9. [12] = { forward = false, name = "opend1", },
10. [13] = { forward = false, name = "opend1", },
11. [14] = { forward = false, name = "opend2", },
12. [15] = { forward = false, name = "opend2", },
13. [16] = { forward = false, name = "opend2", },
14. [17] = { forward = false, name = "opend2", },
15. [18] = { forward = false, name = "opend2", },
```

```
[19] = { forward = false, name = "opend2", },
[20] = { forward = false, name = "opend2", },
[21] = { forward = false, name = "opend2", },
},
open_all_doors = {
[6] = { forward = true, name = "opend1", },
[7] = { forward = true, name = "opend1", },
[8] = { forward = true, name = "opend1", },
[9] = { forward = true, name = "opend1", },
[10] = { forward = true, name = "opend1", },
[11] = { forward = true, name = "opend1", },
[12] = { forward = true, name = "opend1", },
[13] = { forward = true, name = "opend1", },
[14] = { forward = true, name = "opend2", },
[15] = { forward = true, name = "opend2", },
[16] = { forward = true, name = "opend2", },
[17] = { forward = true, name = "opend2", },
[18] = { forward = true, name = "opend2", },
[19] = { forward = true, name = "opend2", },
[20] = { forward = true, name = "opend2", },
[21] = { forward = true, name = "opend2", },
},
close_doors_left = {
[6] = { forward = false, name = "opend1", },
[7] = { forward = false, name = "opend1", },
[8] = { forward = false, name = "opend1", },
[9] = { forward = false, name = "opend1", },
[14] = { forward = false, name = "opend2", },
[15] = { forward = false, name = "opend2", },
[16] = { forward = false, name = "opend2", },
[17] = { forward = false, name = "opend2", },
},
open_doors_left = {
[6] = { forward = true, name = "opend1", },
[7] = { forward = true, name = "opend1", },
[8] = { forward = true, name = "opend1", },
[9] = { forward = true, name = "opend1", },
[14] = { forward = true, name = "opend2", },
[15] = { forward = true, name = "opend2", },
[16] = { forward = true, name = "opend2", },
[17] = { forward = true, name = "opend2", },
},
close_doors_right = {
[10] = { forward = false, name = "opend1", },
[11] = { forward = false, name = "opend1", },
[12] = { forward = false, name = "opend1", },
[13] = { forward = false, name = "opend1", },
[18] = { forward = false, name = "opend2", },
[19] = { forward = false, name = "opend2", },
[20] = { forward = false, name = "opend2", },
[21] = { forward = false, name = "opend2", },
},
open_doors_right = {
```

```
69.  [10] = { forward = true, name = "opend1", },
70.  [11] = { forward = true, name = "opend1", },
71.  [12] = { forward = true, name = "opend1", },
72.  [13] = { forward = true, name = "opend1", },
73.  [18] = { forward = true, name = "opend2", },
74.  [19] = { forward = true, name = "opend2", },
75.  [20] = { forward = true, name = "opend2", },
76.  [21] = { forward = true, name = "opend2", },
77.  },
78.  },
```

Display More

Und das wars eigentlich schon!

6 Zusammenfassung

Zusammenfassend müssen wir zwei Schritte erledigen:

1) Wir erstellen die Animation für die linke und die rechte Tür in deren jeweiliger msh-Datei.
2) Wir definieren den Event zum Öffnen und Schließen für jede Tür in unserer mdl-Datei.

Bei Fragen, fragen!