# Animate doors

Animating doors is easier than you'd expect. The moving direction is not related to the train, but to the door itself and its origin. In other words a left door always opens to the left, no matter if it's installed on starboard (moving towards the aft) or port side (moving towards the front) of the train. It's the same value and direction every time which makes scripting the animation pretty easy. Let's go ahead!

I'm using my own mod train (BR 270) as example code snippets which means you cannot use my unchanged code examples in your own mods. Your mesh koordinates and pathes will be different.

This tutorial only covers sliding doors since I'm still a rookie myself. However adding rotation or movements in other directions should be as easy but might require some additional trial and error until the correct values are found.

I don't know if there are better ways to do this since it was my very first modification, but I had to test a lot in-game to find the correct movement axes and values. I changed the scripts, imported the train, tried it out, closed the game, changed the script, imported it again, tried it again and so on.

## Requirements

In order to even start thinking about animating doors we should make sure that certain requirements are met. Doors can only be animated if they consist of their own objects within your model. So I take it you created your train body first and later created doors and placed them along or within your body without merging the meshes. If this is not the case and your doors are a part of the body mesh you can stop reading now, because these doors cannot be animated. Or maybe you go on reading if you plan to change your mod and add your doors separately, because it is important to have an eye on origin and orientation (normals) of your objects. This is covered by this tutorial and might save you some additional time and problems later on.

## The following files need to be touched

| File | Path | To do |
|---|---|---|
| mdl file for the train | \res\models**model**\vehicle\train\br270-tw.**mdl** | Create events for the door animations |
| msh file for the left door | \res\models**mesh**\vehicle\train\br270-tw\br270_door-left.**msh** | Create script to animate all left doors |
| msh file for the right door | \res\models**mesh**\vehicle\train\br270-tw\br270_door-right.**msh** | Create script to animate all right doors |

## Let's start with the doors

Open your msh file for the left door and you'll find an empty node "animations":

Code

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

This is where the code for the door movement and timing goes. We can use the same animation for opening and closing by running it in reverse, or we can use two different animations.

This is my animation code, the explanation follows below:

Code

```
1. animations = {
2. opend1 = {
3. type = "KEYFRAME",
4. params = {
5. origin = {
6. 0, 0, 0,
7. },
8. keyframes = {
9. {
10. time = 0,
11. rot = { 0, 0, 0 },
12. transl = { 0, 0, 0 }
13. }, {
14. time = 800,
15. rot = { 0, 0, 0 },
16. transl = { 0, 0.28, 0 }
17. }
18. }
19. }
20. },
21. },
```

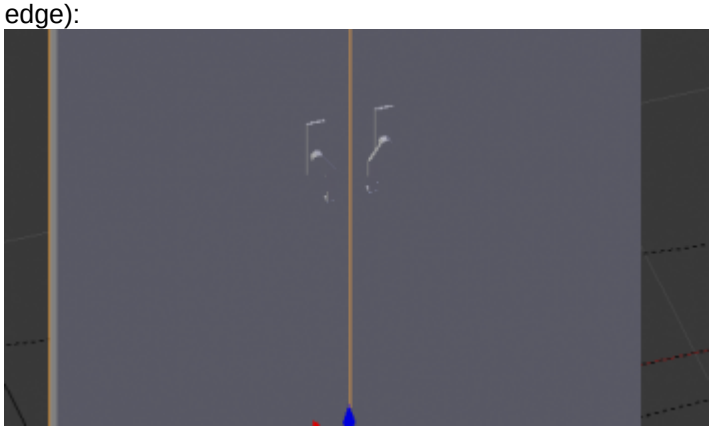
Display More

*opend1* is the name of the data block describing the animation and is bound to the door. For the other door we have to make the same data block with respective values but we'll use the name "opend2". (I presume any name is possible, but I'm not sure. I took these names from existing mods while reverse-engineering how it works.)

*origin* defines the starting point of the animation in relation to your door's position. The three values describe

the X-, Y- and Z-coordinates from where the movement shall start. In Blender I've set my doors' origins instinctively at the inner edges for easier positioning within my model. This came in very handy during animation, because I didn't have to change the origin coordinates in this script. If this is not the case for your model, you can re-define the origin here I guess, but I'd recommend to fix it in your 3D model, because it will be very annoying to find the correct script values by trial and error. If the orientation (normals) of your door is precise, you only have to move it along one axis (for a sliding door) otherwise you should fix that in your model, too. But we'll get into this later on.

The following screenshot shows origin and normals for my left door (the orange dot at the bottom inner edge):



*keyframes* define the required steps to open the door. For a slidinbg door like mine two keyframes are enough, one for the closed position and one for the opened position. For more complex animations on multiple axes (like flipping out first then sliding away) you can add multiple frames, e.g. one for the closed position, one for the flipped out position and one for the end position after sliding.

A *keyframe* has three properties:
- *time* is the point in time, when the keyframe is started. In the closed position this is obviously 0 and in the opened position it is the time the door should take to fully open. I assume this value is set in milliseconds, but I can't tell for sure. In my example the door is fully opened after 800 ms. Changing these values can increase or decrease the time, your doors need to open/close (I changed this value later on to synchronise my door's movements with the assotiated sound).
- *rot* defines the rotation of the door in degrees on every axis (X, Y and Z)
- *transl* defines the movement of the door on one or several axes (X, Y, Z) in metres. In my example the door will be moved by 0.28 m on the Y-axis. If you take another look at my screenshot above you'll see that my Y-axis points exactly towards the front of the train which means that the door will be moved in exactly this direction. On the other side of the train the door (linked object in Blender) is rotated by 180° which means its Y-axis points directly towards the rear of the train. It will be moved the very same amount into the opposite direction (it's also a left door). The Y-axis of my right door points in the same direction so I had to assign a negative value (-0.28) to make it slide to the right.

Of course I didn't know the exact amount of metres to move the door. The mod that I reverse-engineered used a value of 0.5. I took this value and tried it out in-game. The door moved far too far so I went back to the script and decreased it by half (0.25), tried it out in-game and increased to 0.26 again, tried it out and increased it once more to 0.27 until I was satisfied. This way of trial and error is easier, if you start with only

a couple of doors first. I only animated the 4 left doors on port side first to find the correct values and used them later for the remaining doors.

The same has to be done for the msh file of the right door, but remember to give your data block a different name (like "opend2"). The animation iself is done now.

## Let's have a look at the events

In the mdl file of the train we have to create events for every LOD. Since I only have one LOD (yet) this is (still) pretty clearly laid out:

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 = {
27. 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,
28. },
29. type = "MESH",
30. },
31. <---- snipp ---->
```

Display More

I'll cut here. My first LOD contains another *animations* node, but this we ignore completely. Instead we have to take a look at the *children* node. Here we see all meshes and groups within our model. First the body

(vehicle/train/br270-tw/br270-tw_body.msh), below the forward (vehicle/train/br270-tw_lod_0_bogie-front.grp) and rear bogie (vehicle/train/br270-tw_lod_0_bogie-rear.grp). Below the first left door (vehicle/train/br270-tw/br270_door-left.msh). Following (but cut off) are 7 more left doors and 8 right doors, the scharfenberg coupler and the short coupler, but we only care about the doors.
We have to "count" the index of every door now. The indices are not zero-based, but start with 1. This means the body has index 1, because it#s the first child. The bogies have index 2 and 3 respectively and my first left door has the index 4. The right doors start with index 12 and so on. We have to remember these, write them down or count again later on, but for now we create the events right below the *children* node:

Code

```
1. animations = {
2. },
3. children = {
4. <---- snipp ---->
5. },
6. events = {
7. close_all_doors = {
8. },
9. open_all_doors = {
10. },
11. },
```

Display More

Inside the *events* node we create the nodes *close_all_doors* and *open_all_doors*. For every door -identified by the its index- we have to define the proper animation and whether it's run in reverse:

Code

```
1. events = {
2. close_all_doors = {
3. [4] = {
4. forward = false,
5. name = "opend1",
6. },
7. [5] = {
8. forward = false,
9. name = "opend1",
10. },
11. <---- snipp ---->
```

Display More

To close the first two left doors (index 4 and 5) we have to run our animation "opend1" in reverse: "forward = false". This is to be done for all left doors of course. Other modders combined their doors into larger objects which means they only have to animate one left door on starboard which in fact contains all 4 doors within one object. This is frequently done to keep these scripts small for better maintainability, especially if you use several LODs. my mod however is pretty rich in polygons so I have to save whatever is possible. Using my

doors separately allows me to re-use the mesh 7 times (for each of the two doors).

My first two right doors with index 12 and 13 use the values from the data block "opend2" defined in their mesh file - as do the remaining 6 doors, which I don't show here:

Code

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

The same is to be done for the opening of the doors within the node *open_all_doors*, but this time we let the animation run forward:

Code

```
1. open_all_doors = {
2. [4] = {
3. forward = true,
4. name = "opend1",
5. },
6. [5] = {
7. forward = true,
8. name = "opend1",
9. },
10. <---- snipp ---->
11. [12] = {
12. forward = true,
13. name = "opend2",
14. },
15. [13] = {
16. forward = true,
17. name = "opend2",
18. },
19. <---- snipp ---->
20. },
```

Display More

Now these are the animation events for all doors. If we want to animate only the doors on the platform side we need to add events for them, too. This requires four additional nodes inside the "event" node:

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

Best approach would be to copy the values from our "close_all_doors" and "open_all_doors" events and then just remove the indices of the doors we don't want to open.

Following is the code for my BR 270 mod:

Code

```
1. events = {
2. close_all_doors = {
3. [6] = {
4. forward = false,
5. name = "opend1",
6. },
7. [7] = {
8. forward = false,
9. name = "opend1",
10. },
11. [8] = {
12. forward = false,
13. name = "opend1",
14. },
15. [9] = {
16. forward = false,
17. name = "opend1",
18. },
19. [10] = {
20. forward = false,
21. name = "opend1",
22. },
```

```
23.  [11] = {
24.  forward = false,
25.  name = "opend1",
26.  },
27.  [12] = {
28.  forward = false,
29.  name = "opend1",
30.  },
31.  [13] = {
32.  forward = false,
33.  name = "opend1",
34.  },
35.  [14] = {
36.  forward = false,
37.  name = "opend2",
38.  },
39.  [15] = {
40.  forward = false,
41.  name = "opend2",
42.  },
43.  [16] = {
44.  forward = false,
45.  name = "opend2",
46.  },
47.  [17] = {
48.  forward = false,
49.  name = "opend2",
50.  },
51.  [18] = {
52.  forward = false,
53.  name = "opend2",
54.  },
55.  [19] = {
56.  forward = false,
57.  name = "opend2",
58.  },
59.  [20] = {
60.  forward = false,
61.  name = "opend2",
62.  },
63.  [21] = {
64.  forward = false,
65.  name = "opend2",
66.  },
67.  },
68.  open_all_doors = { <---- snipp ----> },
69.  close_doors_left = {
70.  [6] = {
71.  forward = false,
72.  name = "opend1",
73.  },
74.  [7] = {
75.  forward = false,
76.  forward = false,
```

```
77.  name = "opend1",
78.  },
79.  [8] = {
80.  forward = false,
81.  name = "opend1",
82.  },
83.  [9] = {
84.  forward = false,
85.  name = "opend1",
86.  },
87.  [14] = {
88.  forward = false,
89.  name = "opend2",
90.  },
91.  [15] = {
92.  forward = false,
93.  name = "opend2",
94.  },
95.  [16] = {
96.  forward = false,
97.  name = "opend2",
98.  },
99.  [17] = {
100. forward = false,
101. name = "opend2",
102. },
103. },
104. open_doors_left = { <---- snipp ----> },
105. close_doors_right = {
107. [10] = {
108. forward = false,
109. name = "opend1",
110. },
111. [11] = {
112. forward = false,
113. name = "opend1",
114. },
115. [12] = {
116. forward = false,
117. name = "opend1",
118. },
119. [13] = {
120. forward = false,
121. name = "opend1",
122. },
123. [18] = {
124. forward = false,
125. name = "opend2",
126. },
127. [19] = {
128. forward = false,
129. name = "opend2",
130. },
```

```
131.  [20] = {
132.  forward = false,
133.  name = "opend2",
134.  },
135.  [21] = {
136.  forward = false,
137.  name = "opend2",
138.  },
139.  },
140.  open_doors_right = { <---- snipp ----> },
141.  },
```

Display More

And that's it!

## What if i want a different animation for opening and closing?

In some case, like if you want to synchronize the doors animations with custom sounds you may want to have a different animation for opening and closing doors. You can do that by adding a second animation to the door with a different name and using in with the relevent event in the model file in the same way as above. Remember that in most case the origin of the second animation will have to start at the last coordinate of the opening animation and finish at the initial coordinate of the door.

Here is an exemple where the closing door animation is delayed to let the closing door warning sound play before the door actually close :

Code: door.mesh

```
1.  opend1 = {
2.  type = "KEYFRAME",
3.  params = {
4.  origin = {
5.  0, 0, 0,
6.  },
7.  keyframes = {
8.  {
9.  time = 0,
10.  rot = { 0, 0, 0 },
11.  transl = { 0, 0, 0 }
12.  }, {
13.  time = 800,
14.  rot = { 0, 0, 0 },
15.  transl = { -0.65, 0, 0 }
```

```
}
}
}
},
closed1 = {
type = "KEYFRAME",
params = {
origin = {
-0.65, 0, 0,
},
keyframes = {
{
time = 0,
rot = { 0, 0, 0 },
transl = { -0.65, 0, 0 }
}, {
time = 2000,
rot = { 0, 0, 0 },
transl = { -0.65, 0, 0 }
}, {
time = 2800,
rot = { 0, 0, 0 },
transl = { 0, 0, 0 }
}
}
}
},
},
```

Display More

Code: model.mdl

```
open_all_doors = {
[3] = { name = "opend1", forward = true },
},
close_all_doors = {
[3] = { name = "closed1", forward = true },
},
```

## Summary

All in all we only have to do two steps:

1) Create the animations for the left and right door within their respective msh files.

2) Define events for opening and closing for every door within our mdl file.

Any questions? Ask!