

Brückenkonfiguration

Inhaltsverzeichnis

- [1 Brückenkonfiguration](#)
 - [1.1 materialsToReplace](#)
 - [1.2 updateFn](#)
- [2 Pfeiler](#)
 - [2.1 configurePillar](#)
- [3 Brückenträger](#)

Mit Stichtag 22.Juni 2020 sind die untenstehenden Informationen veraltet. Eine detaillierte und aktuelle Aufstellung gibt es in englischer Sprache unter <https://transportfever2.com/wiki/index.php?id=modding:bridgestunnels>

[h1]

1 Brückenkonfiguration

[/h1]

Die Brückenkonfigurationen liegen im Ordner `res/config/bridge`. Sie enthalten eine `data()`-Funktion, die die folgenden Daten zurückgibt:

Name	Datentyp	Beispielwert (iron.lua)	Beschreibung
name	String	_("Iron bridge")	Der Name der Brücke, der als Tooltip im Brückenauswahlmenü gezeigt wird
yearFrom	Integer	1910	Ab wann ist die Brücke verfügbar?
yearTo	Integer	0	Bis wann ist die Brücke verfügbar?
carriers	Liste aus "RAIL", "ROAD"	{ "RAIL" , "ROAD" }	Ist die Brücke als Straßen- und/oder Eisenbahnbrücke verfügbar?
speedLimit	Double	180.0 / 3.6	Erlaubte Höchstgeschwindigkeit in m/s
pillarLen	Integer	3	(?) Länge eines Pfeilers in Brückenrichtung
pillarMinDist	Double	18.0	minimaler Abstand zwischen 2 Pfeilern
pillarMaxDist	Double	66.0	maximaler Abstand zwischen 2 Pfeilern
pillarTargetDist	Double	36.0	optimaler Abstand zwischen 2 Pfeilern

cost	Double	300.0	Kostenfaktor
materialsToReplace	Liste (siehe unten)	(siehe unten)	Tabelle zur Ersetzung von Texturen bei Straßen auf Brücken
updateFn	Funktion (siehe unten)	bridgeutil.makeDefaultUpdateFn(config)	Funktion die zur Zusammensetzung der Brückenmodelle genutzt wird

1.1 materialsToReplace

Diese Tabelle wird genutzt, um bei Straßen Materialien auszutauschen, wenn sie auf Brücken gebaut werden. Möglich sind dabei alle Materialien, die auch bei Straßen angegeben werden:

- streetPaving, streetBorder, streetLane, streetStripe, streetStripeMedian
- streetTram, streetTramTrack, streetBus
- crossingLane, crossingBus, crossingTram, crossingTramTrack, crossingCrosswalk, crossingStopline
- sidewalkPaving, sidewalkLane, sidewalkBorderInner, sidewalkBorderOuter, sidewalkCurb, sidewalkWall

1.2 updateFn

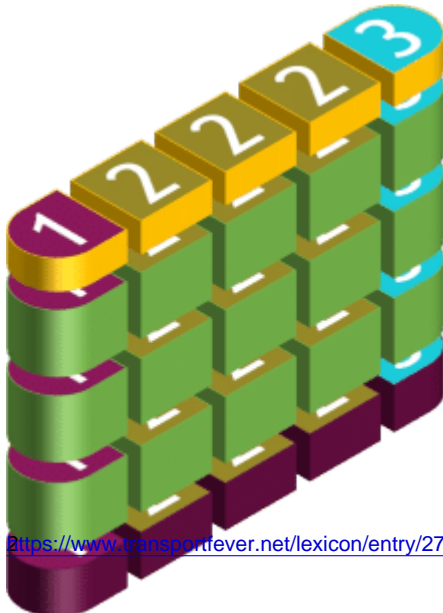
Diese Funktion wird genutzt, um das Brückenmodell aus Einzelteilen zu konstruieren. In `scripts/bridgeutil.lua` wird eine vorgefertigte Funktion angeboten, die auch von den Vanilla-Brücken genutzt wird. Diese bekommt eine config-Tabelle als Parameter übergeben. Die Inhalte dieser Tabelle werden im folgenden beschrieben:

[h1]

2 Pfeiler




[/h1]

Die Pfeiler bestehen aus drei Schichten, wovon die mittlere Schicht mehrfach wiederholt werden kann:



pillarBase (lila)

Dies ist eine Liste mit Modellen, die für den unteren Abschluss eines Pfeilers genutzt werden. Die Länge der Liste kann variieren, möglich sind:

- 1
Modell  Dieses Modell wird mittig als Pfeiler gesetzt.
- 2
Modelle  Das erste Modell wird für die Pfeilerränder genutzt (einmal entsprechend rotiert), das zweite Modell wird für die Mitte des Pfeiler solange nebeneinander gesetzt und leicht skaliert, bis es die erforderliche Breite ausfüllt.
- 3
Modelle  Das erste Modell wird für den einen Pfeilerrand genutzt, das zweite Modell wird für die Mitte des Pfeiler solange nebeneinander gesetzt und leicht skaliert, bis es die erforderliche Breite ausfüllt und das dritte Modell wird für die andere Seite genutzt, dabei aber nicht rotiert.

pillarRepeat (grün)

Dies ist eine Liste mit Modellen, die für den höhenvariablen Teil des Pfeilers genutzt wird. Diese Modelle werden vertikal mehrfach aneinander gereiht und ggf. leicht skaliert, um auf die erforderliche Höhe zu kommen. Die Länge der Liste kann variieren, möglich sind die Kombinationen wie bei **pillarBase** beschrieben.

pillarTop (gelb)

Dies ist eine Liste mit Modellen, die für den oberen Abschluss des Pfeilers genutzt wird. Die Länge der Liste kann variieren, möglich sind die Kombinationen wie bei **pillarBase** beschrieben.

2.1 configurePillar

Wenn nicht die oben beschriebene Funktion zum Zusammensetzen der Pfeiler genutzt werden soll, sondern z.B. anhand der Höhe des Pfeilers ein anderer Pfeiler zusammengebaut werden, dann kann eine eigene Funktion dafür angegeben werden:

Lua

```
configurePillar = function(modelData, params, i, height, width) [...] end,
```

Diese hat folgende Parameter:

Name	Datentyp	Beschreibung
modelData	komplexe Liste	Eine Liste mit Brückenelement-Modellen und ihren Eigenschaften. Weitere Informationen siehe unten
params	komplexe Liste	Eine Liste mit den Parametern der gerade in Bau befindlichen Brückensektion (zumeist bis zu 3 Pfeiler lang). Weitere Informationen siehe unten
i	Integer	Die Nummer des Pfeilers in der aktuell in Bau befindlichen Brückensektion
height	Double	Die Höhe des zu bauenden Pfeilers in Metern
width	Double	Die Breite des zu bauenden Pfeilers in Metern

modelData

Die standardmäßige configurePillar-Funktion übergibt eine Liste mit allen geladenen Brückenelement-Modellen (nicht nur die von dieser Brücke!). Als Schlüssel der Listeneinträge wird der Pfad zur mdl-Datei verwendet. Für jeden Eintrag wird die BoundingBox bereitgestellt:

Lua

```
[ "bridge/iron/pillar_top_side.mdl" ] = {
    BoundingBox: {
        min: {
            x: 0,
            y: 0,
            z: 0
        },
        max: {
            x: 1,
            y: 1,
            z: 1
        }
    }
},
```

Alles anzeigen

Da das Spiel alle diese Bounding Box Daten lädt und extrahiert, ist es erforderlich, dass alle brückenbezogenen mdl-Dateien unter `res/models/model/bridge/` liegen!

params

Zur Laufzeit wird beim Methodenaufruf eine Tabelle mit diversen Parametern mitgeliefert. Diese sind:

- `pillarWidth`: Die Breite eines Pfeilers
- `pillarHeights`: Eine Liste mit den Höhen der Pfeiler
- `pillarLength`: Die Länge eines Pfeilers
- `state.models`: Eine Liste mit allen geladenen Brückenmodellen (siehe oben)
- `railingWidth`: Die Breite der Brücke
- `railingIntervals`: Eine Liste mit den Brückenträgerabschnitten. Für jeden Abschnitt gibt es drei Informationen:

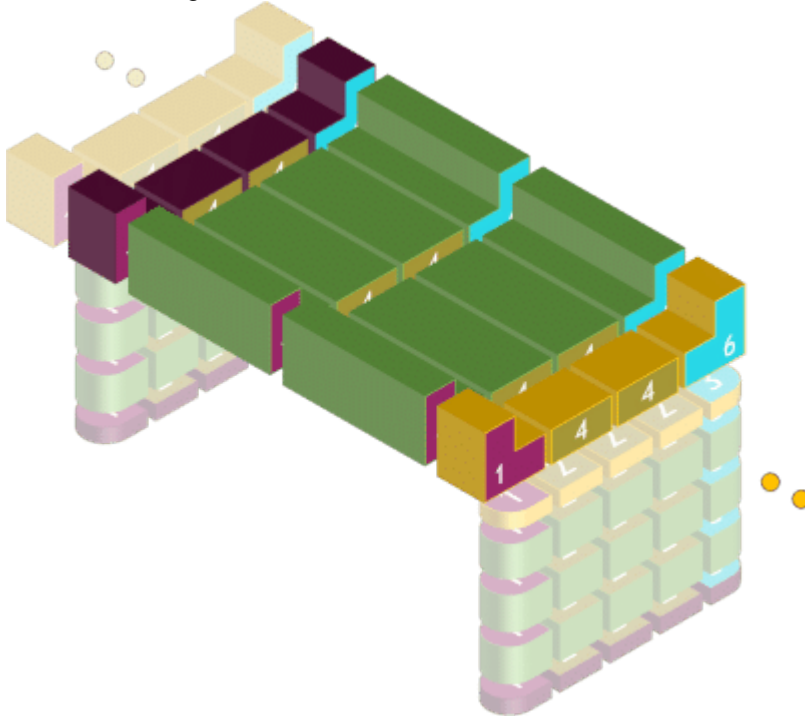
- `hasPillar`: Eine Paar aus zwei Booleanwerten, die vermutlich sagen, ob am Anfang und Ende des Abschnitts ein Pfeiler platziert wird
- `lanes`: ?
- `length`: Die Länge des Abschnitts

[h1]

3 Brückenträger

[h1]

Der Brückenträger besteht ebenfalls aus mehreren Reihen, die aneinandergesetzt werden:



railingBegin

Dies ist eine Liste mit Modellen, die für den Beginn eines Brückenträgersegments genutzt werden. Ein Brückensegment beginnt jeweils am Anfang einer Brücke und bei jedem Pfeiler. Es können entweder 5 oder 8 Modelle in der Liste enthalten sein. Leider ist bei einigen Elementen der Liste noch nicht ganz geklärt, welche Bedeutung sie haben. Wenn nur 5 Elemente enthalten sind, werden die Elemente 1-3 stellvertretend für 6-8 in rotierter Fassung verwendet:

- | | | |
|----------------------|--|---|
| <p>5
Modelle</p> | | <p>Das erste Modell wird für die Außenseiten genutzt (einmal entsprechend rotiert), das vierte Modell wird für die Mitte der Zeile solange nebeneinander gesetzt und leicht skaliert, bis es die erforderliche Breite ausfüllt.</p> |
| <p>8
Modelle</p> | | <p>Das erste Modell wird für die eine Außenseite genutzt, das vierte Modell wird für die Mitte der Zeile solange nebeneinander gesetzt und leicht skaliert, bis es die erforderliche Breite ausfüllt und das sechste Modell wird für die andere Außenseite genutzt, dabei aber nicht rotiert.</p> |

railingRepeat

Dies ist eine Liste mit Modellen, die für die Mitte eines Brückenträgersegements genutzt werden. Diese Modelle werden horizontal mehrfach aneinander gereiht und ggf. leicht skaliert, um auf die erforderliche Höhe zu kommen. Die Länge der Liste kann variieren, möglich sind die Kombinationen wie bei **railingBegin** beschrieben.

railingEnd

Dies ist eine Liste mit Modellen, die für das Ende eines Brückenträgersegments genutzt werden. Die Länge der Liste kann variieren, möglich sind die Kombinationen wie bei **railingBegin** beschrieben.