Projekt 2
Endpräsentation

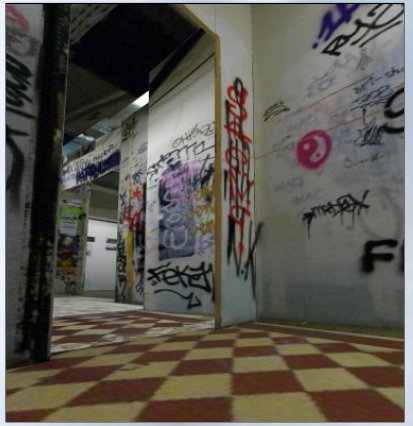
Web ▪ 3D-Modelling

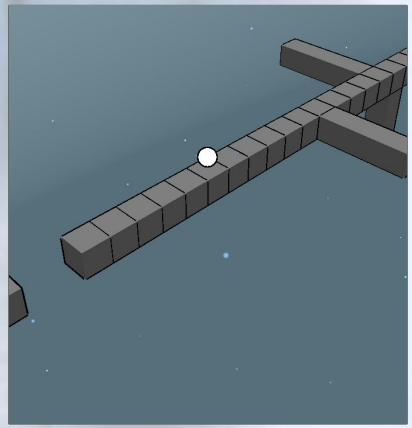
Angela Scherle
28.01.2020

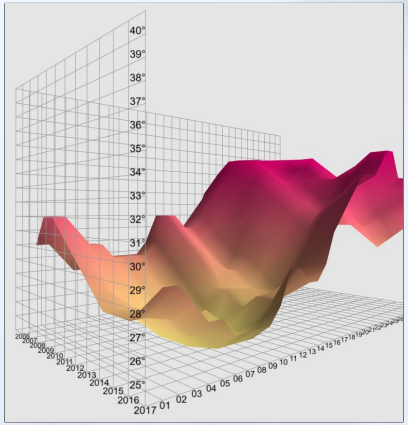
Projekt bei
Prof. Marius Hofmeister
&
Prof. Markus Lauterbach

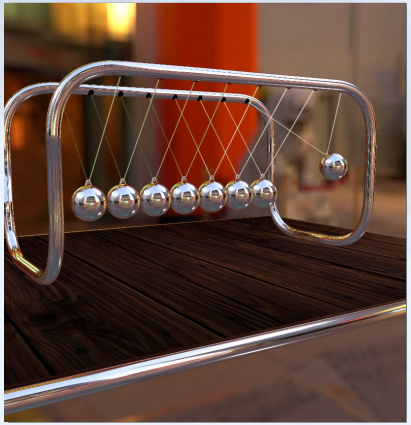
Inspiration - Idee

www.Babylonjs.com



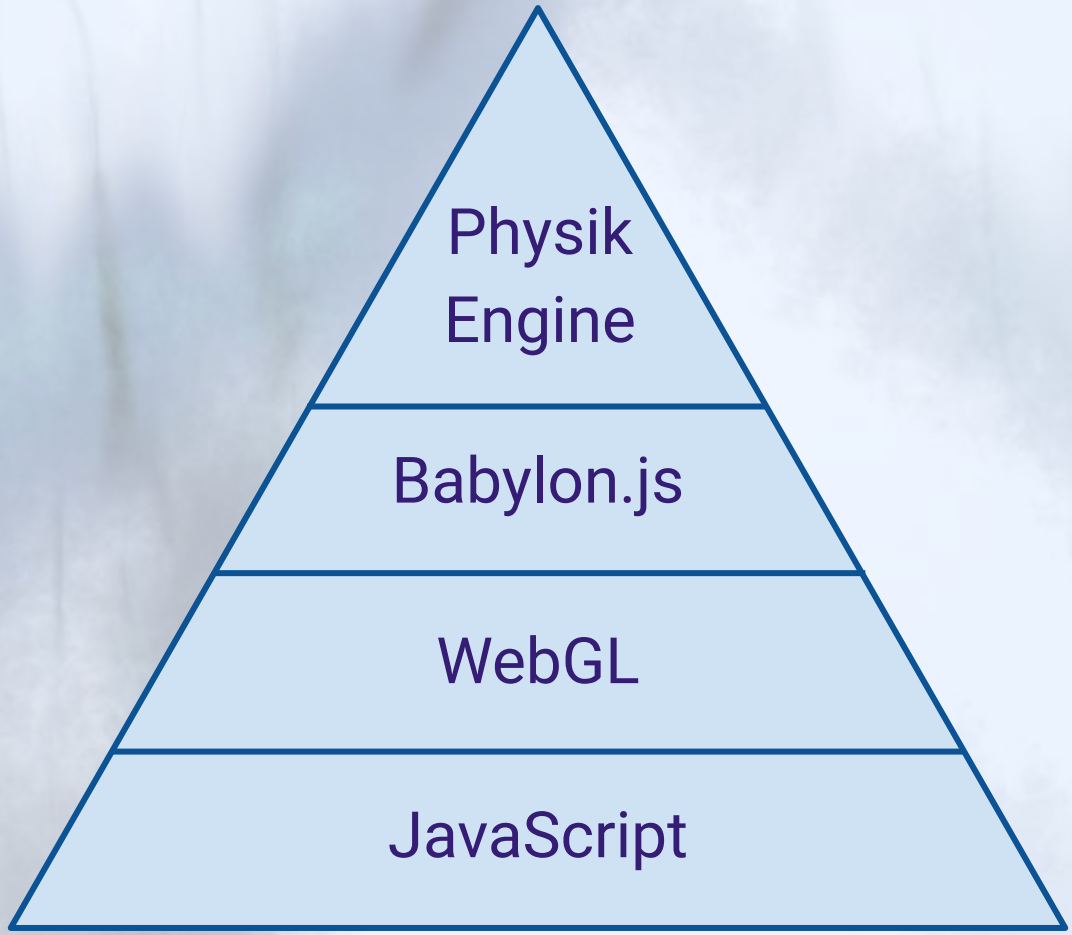






- ➔ 3D Kugelbahn mit Babylon.js
- ➔ Persönliche Ziele:
  - ◆ JavaScript Kenntnisse erlernen/erweitern
  - ◆ Blender Kenntnisse erweitern
  - ◆ Visuell ansprechendes Web-Projekt erzeugen

Über Babylon.js


Physik
Engine
Babylon.js
WebGL
JavaScript

➔ WebGL:

- Web Graphics Library
- JavaScript-Programmierschnittstelle
- Basiert auf OpenGL ES
- Ermöglicht Darstellung von 3D Grafiken im Webbrowser
- Grafiken können durch Code erzeugt oder von 3D-Programmen importiert werden

- Babylon.js:
  - Wurde als 3D Spiele Engine entwickelt
  - Erweiterung der WebGL-Bibliothek
  - Heute vielzählige Anwendungen
  - Ermöglicht Einbindung einer Physik-Engine ins Web

- Physik Engines für Babylon.js:
  - Cannon.js
  - Oimo.js
  - Energy.js
  - Ammo.js

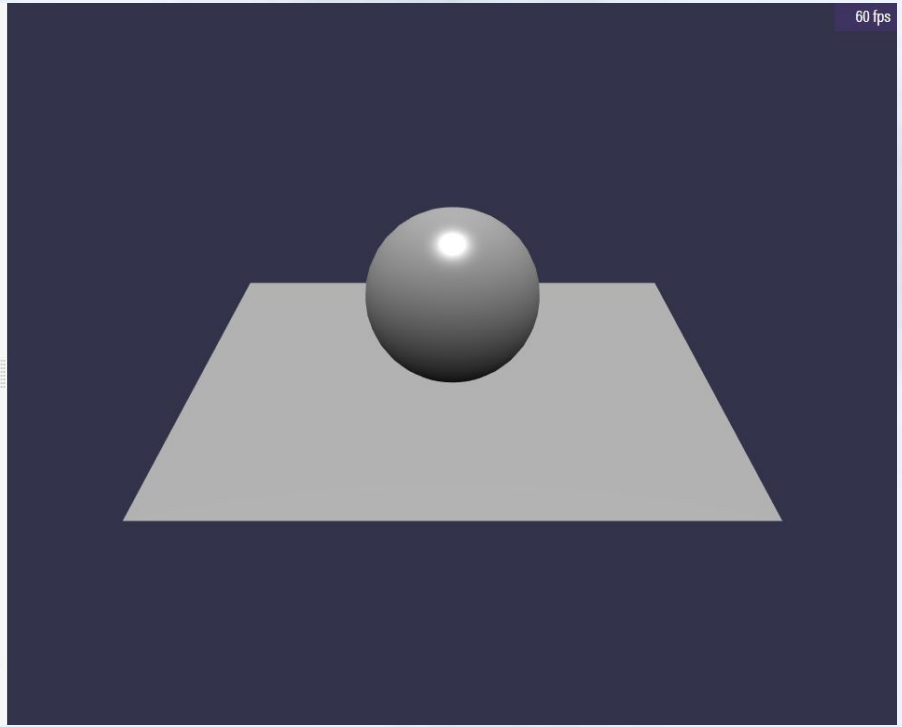
Arbeitsprozess

- ➔ Beispiele auf Babylon.js
- ➔ https://doc.babylonjs.com/examples/

```
var scene = new BABYLON.Scene(engine);
var camera = new BABYLON.FreeCamera("camera1", new BABYLON.Vector3(0, 5, -10), scene);
camera.setTarget(BABYLON.Vector3.Zero());
camera.attachControl(canvas, true);
var light = new BABYLON.HemisphericLight("light1", new BABYLON.Vector3(0, 1, 0), scene);
light.intensity = 0.7;
var sphere = BABYLON.Mesh.CreateSphere("sphere1", 16, 2, scene);
sphere.position.y = 1;
var ground = BABYLON.Mesh.CreateGround("ground1", 6, 6, 2, scene);

scene.exclusiveDoubleMode = false;

scene.onPrePointerObservable.add( function(pointerInfo, eventState) {
    console.log('%c PrePointerObservable: pointer down', 'background: red; color: white');
    //pointerInfo.skipOnPointerObservable = true;
}, BABYLON.PointerEventTypes.POINTERDOWN, false);
scene.onPrePointerObservable.add( function(pointerInfo, eventState) {
    console.log('%c PrePointerObservable: pointer up', 'background: red; color: white');
    // pointerInfo.skipOnPointerObservable = true;
}, BABYLON.PointerEventTypes.POINTERUP, false);
scene.onPrePointerObservable.add( function(pointerInfo, eventState) {
    console.log('%c PrePointerObservable: pointer pick: ' + pointerInfo.pickInfo.pickedMesh.name, 'background:
}, BABYLON.PointerEventTypes.POINTERPICK, false);
scene.onPrePointerObservable.add( function(pointerInfo, eventState) {
    console.log('%c PrePointerObservable: pointer tap', 'background: red; color: white');
}, BABYLON.PointerEventTypes.POINTERTAP, false);
scene.onPrePointerObservable.add( function(pointerInfo, eventState) {
    console.log('%c PrePointerObservable: pointer double tap', 'background: red; color: white');
}, BABYLON.PointerEventTypes.POINTERDOUBLETAP, false);
scene.onPointerObservable.add( function(pointerInfo, eventState) {
    console.log('%c PointerObservable: pointer down', 'background: blue; color: white');
}, BABYLON.PointerEventTypes.POINTERDOWN, false);
scene.onPointerObservable.add( function(pointerInfo, eventState) {
    console.log('%c PointerObservable: pointer up', 'background: blue; color: white');
}, BABYLON.PointerEventTypes.POINTERUP, false);
scene.onPointerObservable.add( function(pointerInfo, eventState) {
    console.log('%c PointerObservable: pointer pick: ' + pointerInfo.pickInfo.pickedMesh.name, 'background: blue
}, BABYLON.PointerEventTypes.POINTERPICK, false);
scene.onPointerObservable.add( function(pointerInfo, eventState) {
    console.log('%c PointerObservable: pointer tap', 'background: blue; color: white');
```



- ➔ Dokumentation in Babylon.js
- ➔ https://doc.babylonjs.com/resources/rotation_conventions

Rotation Conventions

There are several methods of achieving rotations within BabylonJS all of which use a particular convention.

Euler Angles

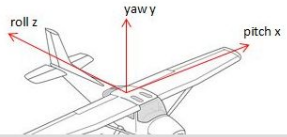
In 3D space Euler angles can produce any possible orientation by providing three angles to rotate about each of three axes in a given order.

For three axes X, Y and Z there are 12 different permutations for the order of the angles. Since X, Y and Z can be treated as *World* or as *local* axes this means there is a potential of 24 different possibilities. Most, if not all,of these are in use in different systems around the world. So you need to be very careful that you know very clearly the convention that the system you are working in uses.

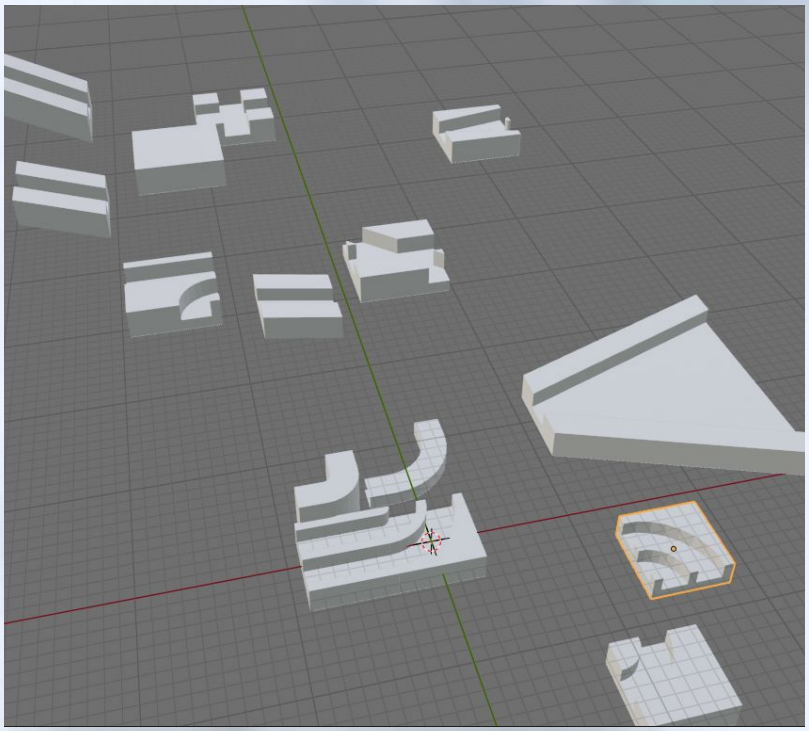
Mesh.rotation(alpha, beta, gamma) uses the three Euler angles alpha, beta and gamma which are rotations about the X, Y and Z axes respectively. The convention that Babylon.js uses is based on the yaw, pitch and roll convention and so is carried out around X, Y and Z as local axes in the order Y, X, Z.

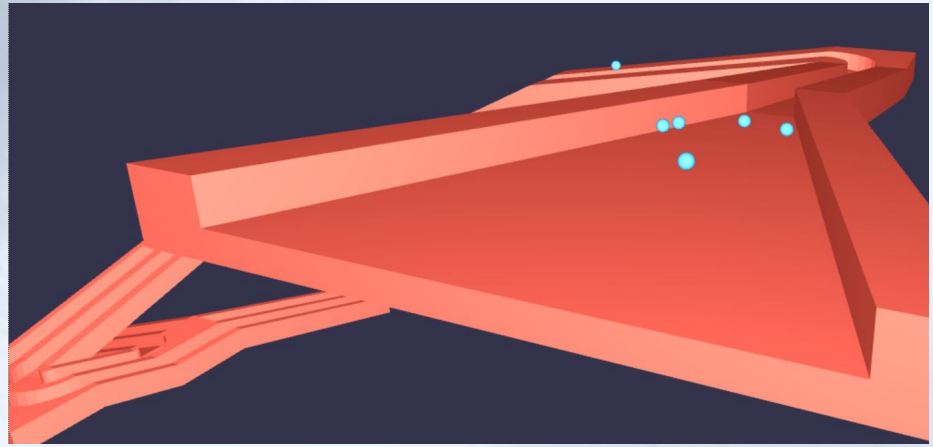
References to Euler angles within the Babylon.js community can usually be taken to mean the angles to use with the *rotation* method.

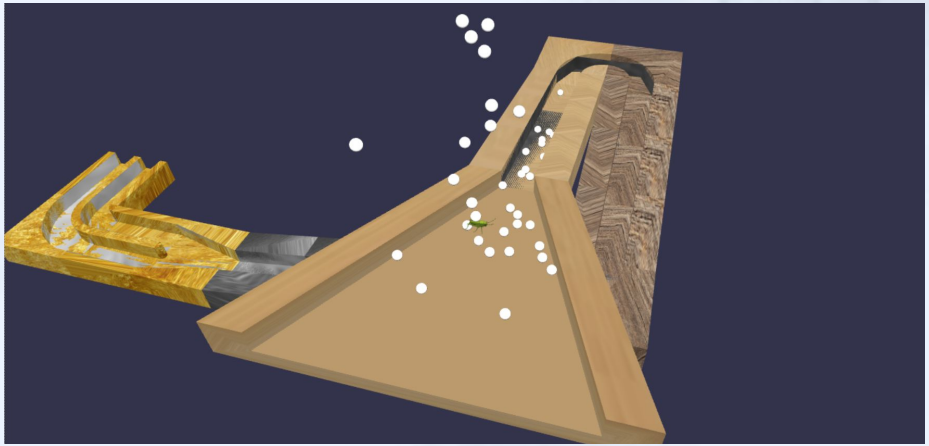
YXZ *Local Axes* Yaw, Pitch, Roll

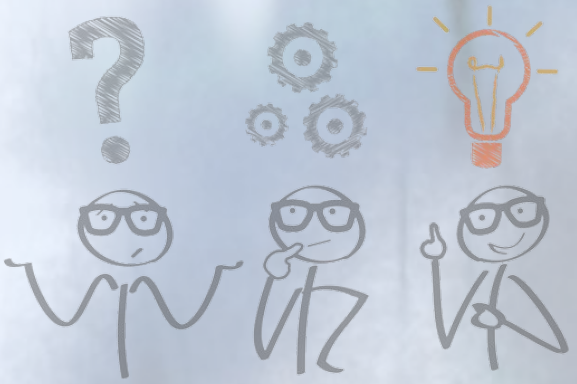


➔ Ursprünglicher Plan:
Kugelbahn aus Einzelteilen





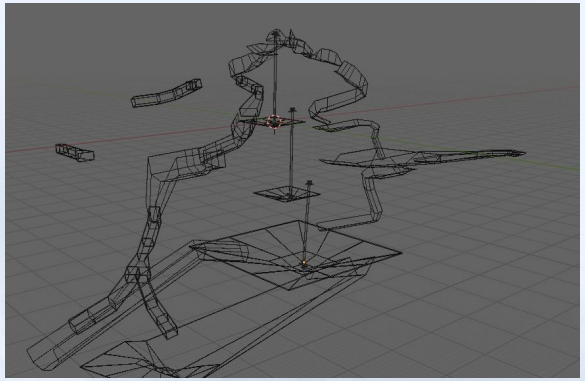


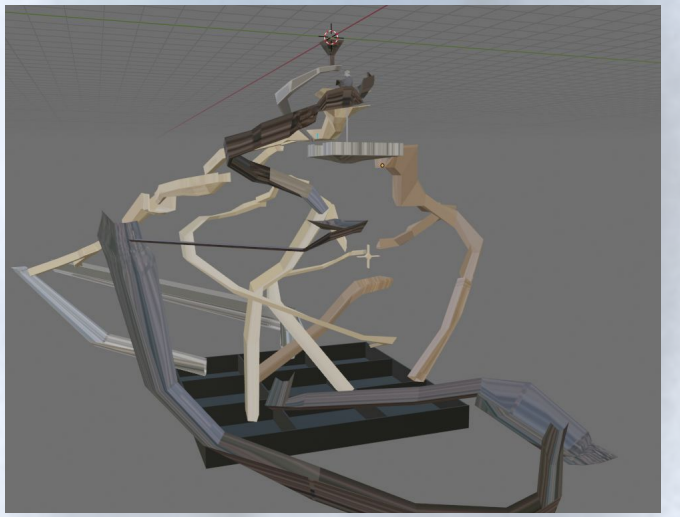


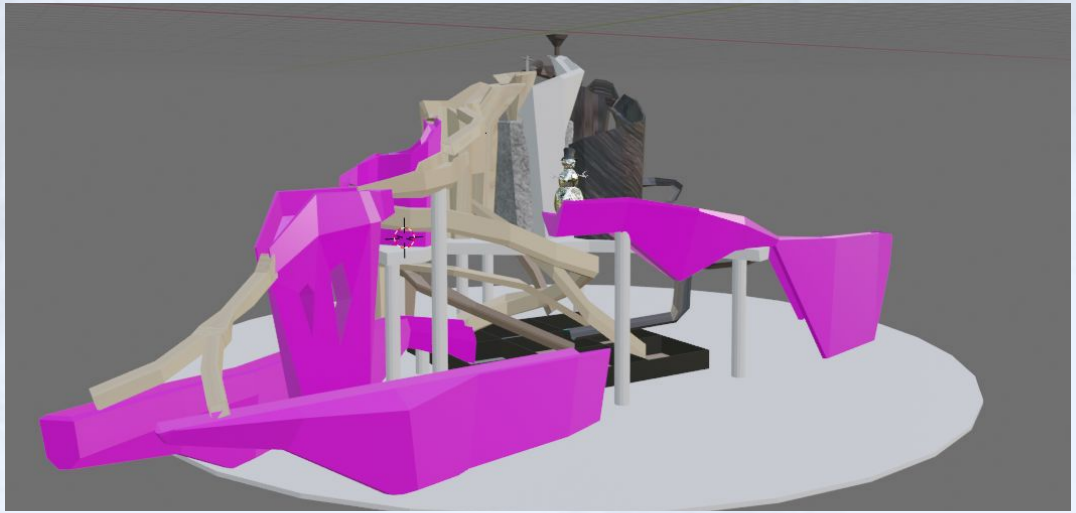
➔ Probleme:
- Unsichtbare “Löcher” in Meshes
- Export von Texturen nicht möglich

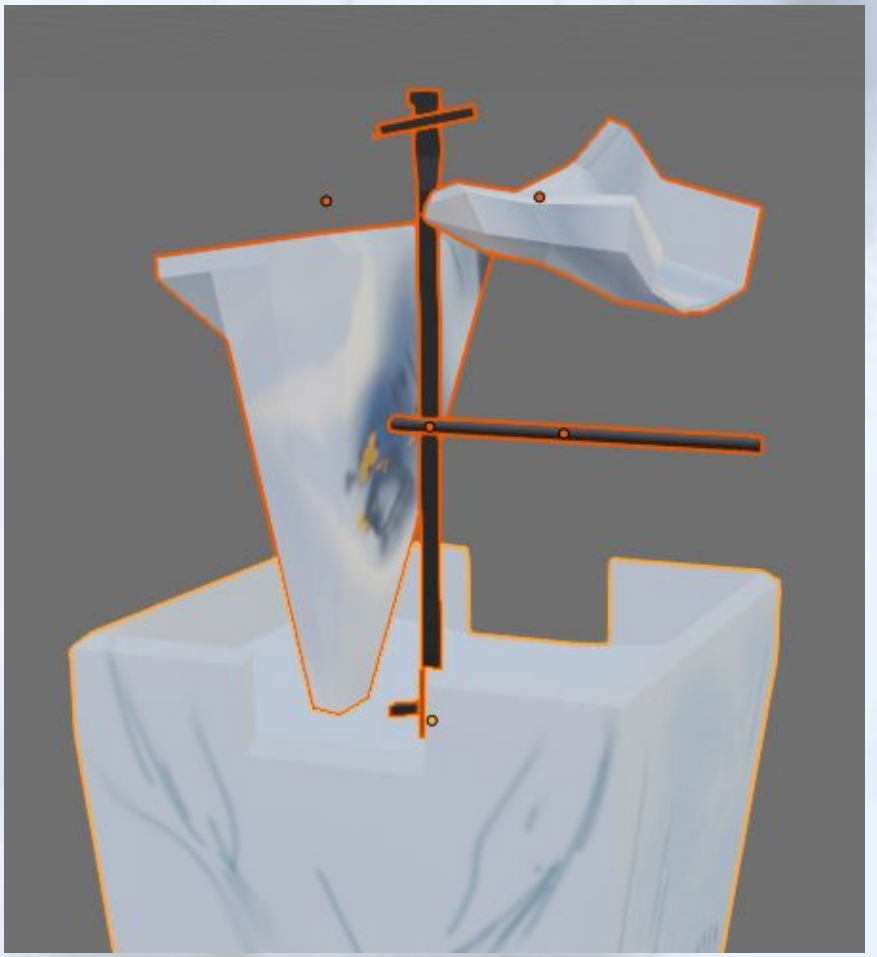
➔ Lösungen:
- Export mit .glb statt .babylon → Texturen vorhanden
- Änderung der Vorgehensweise

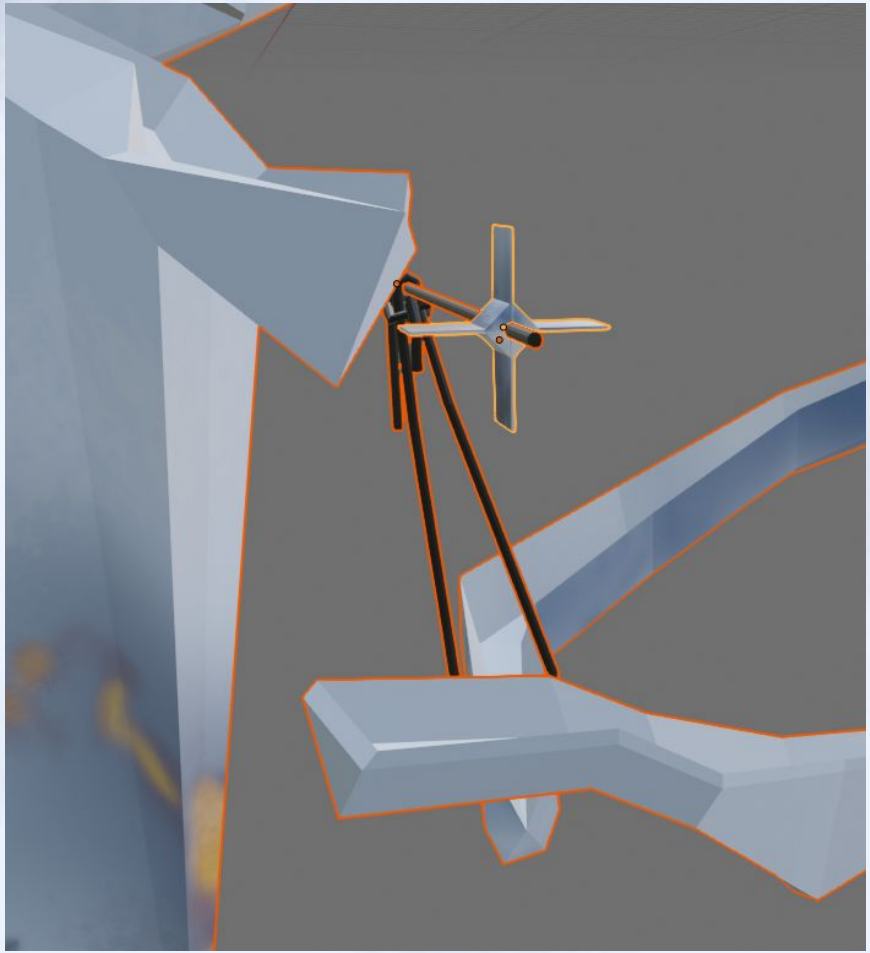
- ➔ Neue Herangehensweise: Bahnen erstellen, testen, bei Fehlern ersetzen
- ➔ Allmähliche Erweiterung und Ergänzung mit neuen Ideen



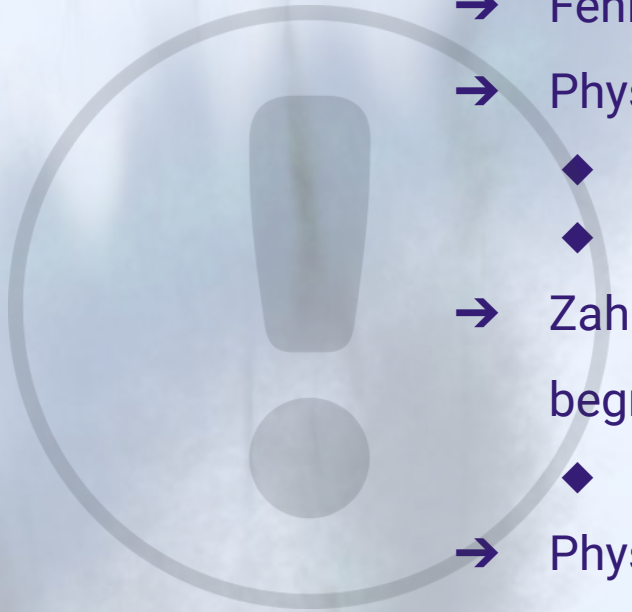








Zu Beachten



- ➔ Export von 3D Meshes fehleranfällig
- ➔ Fehler in Physik-Engine
- ➔ Physik Engine sehr Rechenaufwendig
  - ◆ Google Chrome: Hardware Acceleration notwendig
  - ◆ Grafikprozessor wird gefordert
- ➔ Zahl der Meshes bei Physikalischen Objekten begrenzt
  - ◆ Eis-Welt
- ➔ Physikalisches Verhalten vom Browser abhängig
  - ◆ Optimierung auf Google Chrome

Interaktionsmöglichkeiten

- ➔ Kugeltransport
- ➔ Weichen
- ➔ Klappe
- ➔ Bodenklappen

Marble Run