POWERPUFF GIRLS

Hendrik Pastunink
Jana Butorac
Katharina Rupp
Jessika Maier

Gliederung

- Spielidee
- Demonstration
- Projektordnerhierarchie
- Systemhierarchie
- Zum Zeitplan
- Zum Arbeitsaufwand
- Aufgabenverteilung
- HD-Bilder und Video

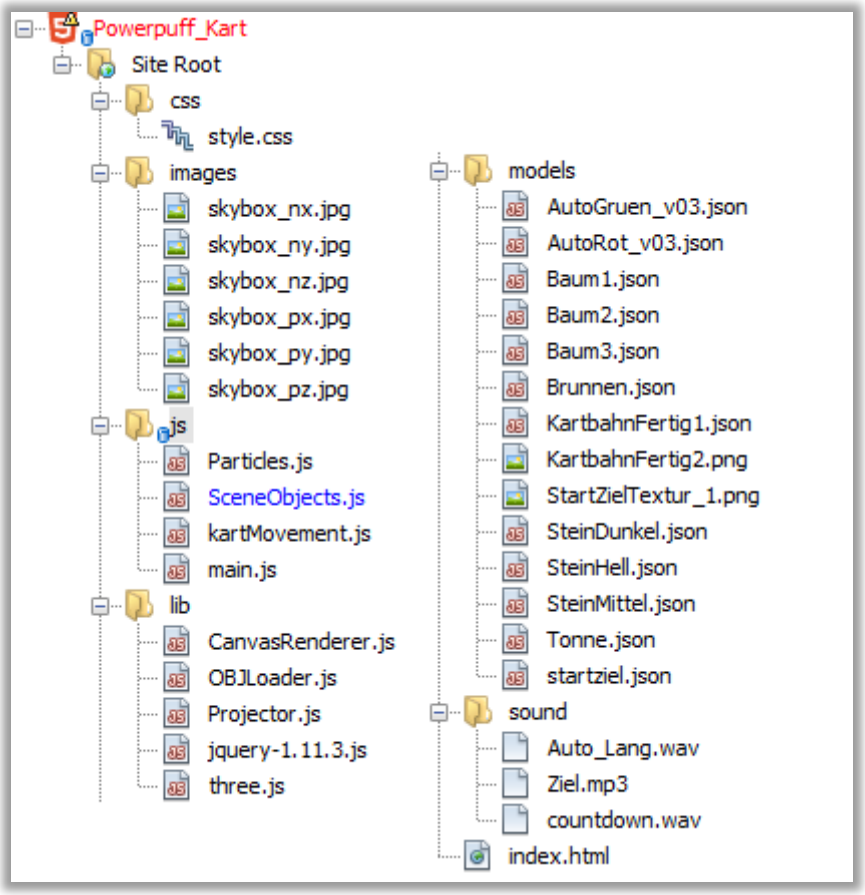
Spielidee

- 2-Spieler Kartbahnspiel
- 3 Runden
- Funktion ähnlich Carrerabahn

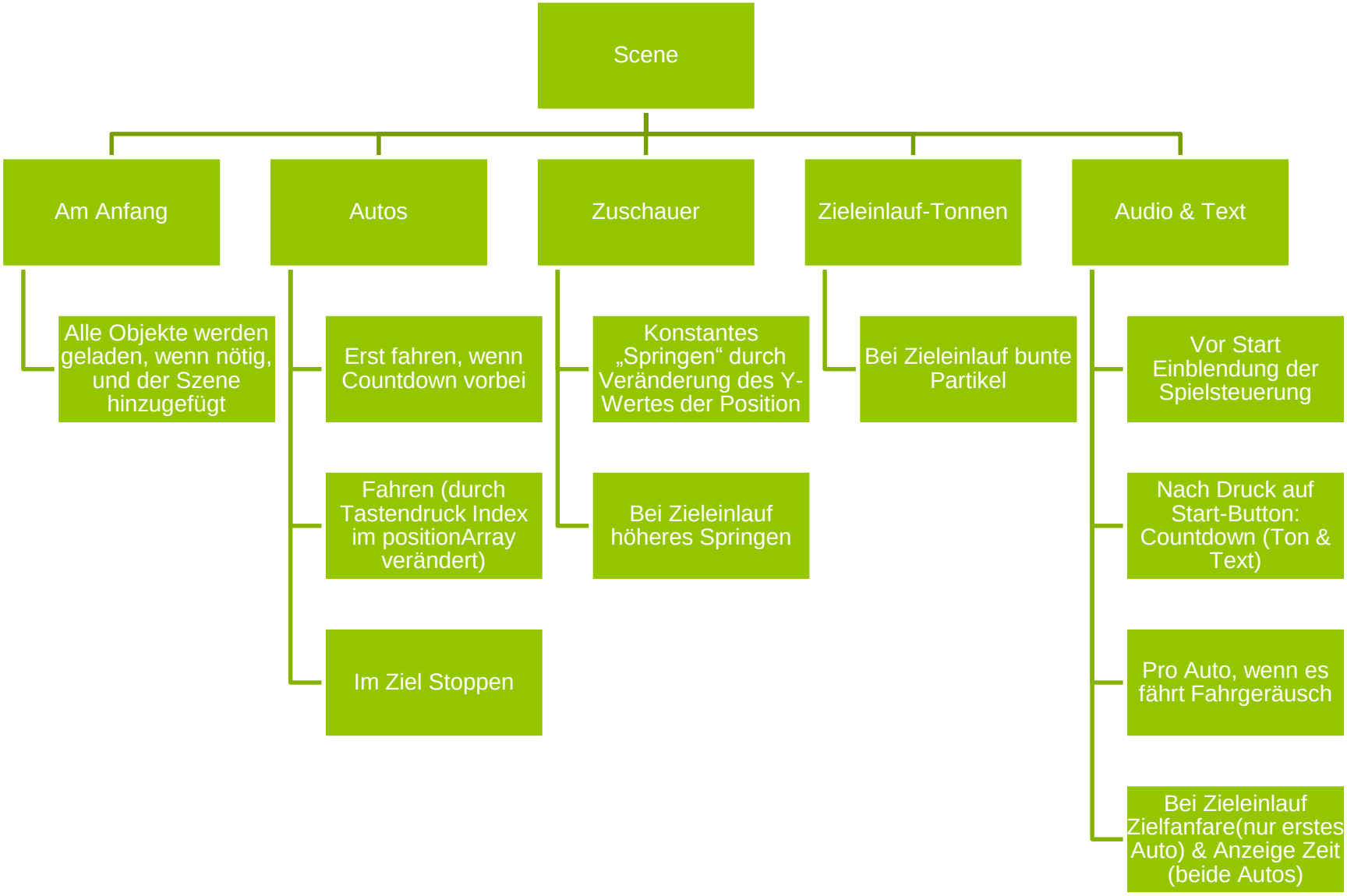
Demonstration

- Für die Demonstration index.html im Projektordner in einem Browser öffnen.
- Es kann sein, dass das Projekt auf schwach ausgestatten Rechnern nicht flüssig läuft.

Projektordnerhierachie



Systemhierarchie



Zum Zeitplan

- Die Zieldaten/Meilensteine wurden nicht eingehalten, da
- Projekt in die Semesterferien verschoben
- Arbeitsaufwand entsprach ungefähr den Erwartungen

Zum Arbeitsaufwand

- Für einen großen Teil der Projektarbeit haben wir uns als Gruppe getroffen, um am Projekt zu arbeiten. Dabei haben wir uns gegenseitig bei vielen Aufgaben geholfen, wodurch die meisten fast die gleiche Stundenzahl an Arbeit in die Aufgaben andere Gruppenmitglieder wie in die eigenen gesteckt haben. Dadurch müssten bei vielen Aufgaben noch andere mit dazugerechnet werden. Weil das aber zu unübersichtlich und schwer nachzuvollziehen ist, wird nur der Arbeitsaufwand der Hauptverantwortlichen für eine Aufgabe dargestellt.

Aufgabenverteilung

- Jana Butorac
  - Skybox einfügen (ca. 3h)
  - Modellierung Bäume (ca. 1/2h)
  - Einbindung Modelle Autos, Bäume, Steine (ca. 1h)
  - Zeitmessung (ca. 1/2h)

- Jessika Maier
  - Zuschauererstellung, -positionierung & -animation (ca. 4h)
  - Zaunerstellung, -vervielfältigung & -positionierung (ca. 2,5h)
  - Modellierung & Texturierung Kartbahn, Start/Ziel-Banner (ca. 2h)

Aufgabenverteilung

- Katharina Rupp
  - Partikelsystem (ca. 3h)
  - Modellierung Tonne (ca. 1/2h)
  - Sound finden / erstellen & einbinden (ca. 3h)
  - Countdown (ca. 1/2h)

- Hendrik Pastunink
  - Fahrbahnberechnung (ca. 5h)
  - Autobewegung (ca. 2h)
  - Gruppenberatung (ca. 10h)

Aufgaben – Jana Butorac

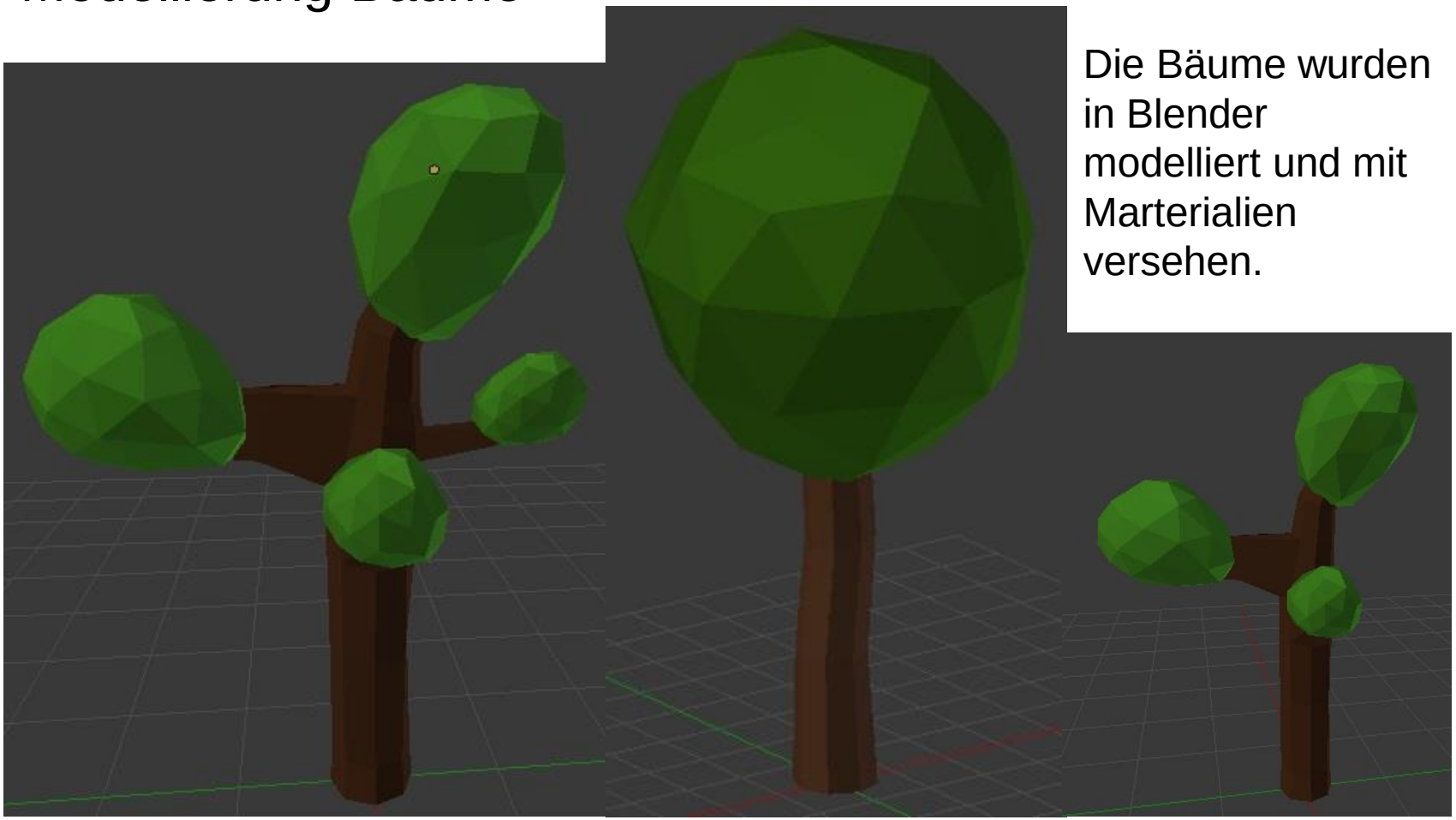
- Skybox einfügen

Die Skybox wurde wie rechts gezeigt zusammengesetzt und in die Szene eingefügt.

```
//Skybox
function addSkybox(scene){
    texture_placeholder = document.createElement('canvas');
    texture_placeholder.width = 128;
    texture_placeholder.height = 128;

    var skyboxContext = texture_placeholder.getContext('2d');
    skyboxContext.fillStyle = 'rgb( 200, 200, 200 )';
    skyboxContext.fillRect(0, 0, texture_placeholder.width, texture_placeholder.height);

    var skyboxMaterials = [
        loadTexture('images/skybox_px.jpg'), // right
        loadTexture('images/skybox_nx.jpg'), // left
        loadTexture('images/skybox_py.jpg'), // top
        loadTexture('images/skybox_ny.jpg'), // bottom
        loadTexture('images/skybox_pz.jpg'), // back
        loadTexture('images/skybox_nz.jpg')  // front
    ];

    var skyboxMesh = new THREE.Mesh(new THREE.BoxGeometry(300, 300, 300, 7, 7, 7), new THREE.MeshFaceMaterial(skyboxMaterials));
    skyboxMesh.scale.x = -1;
    scene.add(skyboxMesh);
}

function loadTexture(path) {
    var texture = new THREE.Texture(texture_placeholder);
    var material = new THREE.MeshBasicMaterial({map: texture, overdraw: 0.5});

    var image = new Image();
    image.onload = function () {
        texture.image = this;
        texture.needsUpdate = true;
    };
    image.src = path;

    return material;
}
```

Aufgaben – Jana Butorac

- Modellierung Bäume

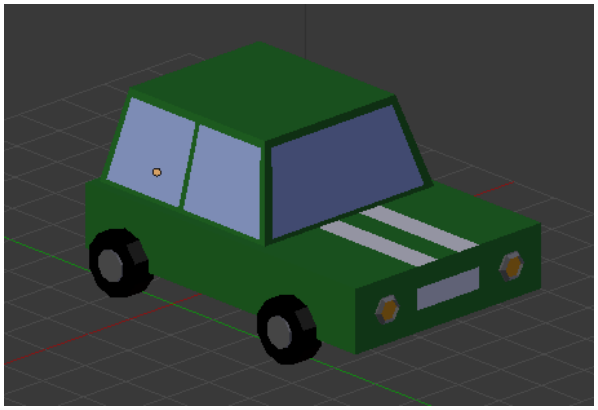


Die Bäume wurden in Blender modelliert und mit Marterialien versehen.

Aufgaben – Jana Butorac

- Einbindung Modelle

  Autos, Bäume, Steine



Die Autos und Steine wurden von Mitgliedern dieser Gruppe für das Bachelorprojekt modelliert und für diese Projekt wiederverwendet.

Autos, Bäume und Steine wurden wie rechts am Beispiel eines Autos gezeigt, in die Szene eingebunden.

```
//Car 1
var car1Manager = new THREE.LoadingManager();
car1Manager.onProgress = function (item, loaded, total) {
    console.log(item, loaded, total);
};

var car1 = new THREE.Mesh();
var car1Loader = new THREE.JSONLoader(car1Manager);
car1Loader.load('models/AutoRot_v03.json', function (geometry, materials) {
    car1 = new THREE.Mesh(geometry, new THREE.MeshFaceMaterial(materials));

    car1.scale.x *= 0.2;
    car1.scale.z *= 0.2;
    car1.scale.y *= 0.2;
    car1.castShadow = true;

    scene.add(car1);
});
```

Aufgaben – Jana Butorac

- Zeitmessung

Beim Countdown-Ende wird die Startzeit der Autos genommen. Wenn eines der Autos drei Runden gefahren ist, wird von der aktuellen Zeit in Millisekunden seit dem 1. Jan 1970 die Startzeit in Millisekunden Abgezogen, wodurch sich die Renn-zeit für das Auto ergibt.

```
function OnCountdownEnded(){
    var startDate = new Date();
    startzeit = startDate.getTime();
    mayDrive1 = true;
    mayDrive2 = true;
}
```

```
//Zeitmessung

startzeit = null;

var runde1 = 0;
var runde2 = 0;
var lastIndex1 = 0;
var lastIndex2 = 0;
```

```
function moveCar(car, number) {
    switch (number) {...50 lines }

    var timeDiv1 = document.getElementById("time1");
    var timeDiv2 = document.getElementById("time2");

    if (runde1 === 3 || runde2 === 3) {
        var zeitNow = new Date();
        var zeit = zeitNow.getTime() - startzeit;
        if(runde1 === 3 && mayDrive1){
            mayDrive1 = false;
            timeDiv1.innerHTML = "Spieler Rot hatte eine Zeit von " + zeit/1000 + " sec";
            console.log("Spieler 1 hatte eine Zeit von " + zeit/1000 + " sec");
            car1Finished = true;
        }
        if(runde2 === 3 && mayDrive2){
            mayDrive2 = false;
            timeDiv2.innerHTML = "Spieler Grün hatte eine Zeit von " + zeit/1000 + " sec";
            console.log("Spieler 2 hatte eine Zeit von " + zeit/1000 + " sec");
            car2Finished = true;
        }
        if(playZielAudio){
            playZielAudio = false;
            document.getElementById("zielAudio").play();
            spawnCubes = true;
        }
```

```
if (lastIndex1 > positionArray1.length * 0.9 && index1 < positionArray1.length / 10) {
    runde1++;
    console.log("Runde: " + runde1);
}
```

Aufgaben - J. Maier

- Zuschauererstellung in Three.js

Zuschauer wird aus Rechtecken zusammengebaut

```
//Zuschauer
function makeSpectator() {
    var zuschauerGeometry = new THREE.BoxGeometry(2.5, 5, 2.5);
    var zuschauerMaterial = new THREE.MeshLambertMaterial({color: 0x2E64FE});
    var zuschauer = new THREE.Mesh(zuschauerGeometry, zuschauerMaterial);

    zuschauer.position.y = 2.5;

    var halsGeometry = new THREE.BoxGeometry(1, 0.5, 1);
    var halsMaterial = new THREE.MeshLambertMaterial({color: 0xFFE5B2});
    var hals = new THREE.Mesh(halsGeometry, halsMaterial);

    hals.position.y = 5;

    var kopfGeometry = new THREE.BoxGeometry(2, 2, 2);
    var kopfMaterial = new THREE.MeshLambertMaterial({color: 0xFFE5B2});
    var kopf = new THREE.Mesh(kopfGeometry, kopfMaterial);

    kopf.position.y = 6.25;

    var armGeometry = new THREE.BoxGeometry(1, 4, 0.5);
    var armMaterial = new THREE.MeshLambertMaterial({color: 0xFFE5B2});
    var arm = new THREE.Mesh(armGeometry, armMaterial);

    arm.position.z = 1.5;
    arm.position.x = 0;
    arm.position.y = 5;

    var armlGeometry = new THREE.BoxGeometry(1, 4, 0.5);
    var armlMaterial = new THREE.MeshLambertMaterial({color: 0xFFE5B2});
    var arml = new THREE.Mesh(armlGeometry, armlMaterial);

    arml.position.z = -1.5;
    arml.position.x = 0;
    arml.position.y = 5;

    zuschauerObj = new THREE.Object3D();
    zuschauerObj.add(zuschauer);
    zuschauerObj.add(kopf);
    zuschauerObj.add(hals);
    zuschauerObj.add(arm);
    zuschauerObj.add(arml);
    zuschauerObj.scale.x *= 0.1;
    zuschauerObj.scale.y *= 0.1;
    zuschauerObj.scale.z *= 0.1;
    zuschauerObj.position.x = -5;
    zuschauerObj.position.z = 5;
    zuschauerObj.rotation.y = Math.PI/2;

    return zuschauerObj;
}
```

Aufgaben - J. Maier

- Zuschauerpositionierung

Zuschauer erhalten einen zufälligen Y-Wert für ihre Position und werden zuschauerArray hinzugefügt. Gleichzeitig wird für jeden Zuschauer zufällig festgelegt, ob er sich gerade in einer Aufwärtsbewegung (true) oder einer Abwärtsbewegung (false) befindet und der entsprechende Wert dem Array zuschauerOrientation hinzugefügt. Dann werden die Zuschauer in einem bestimmten Bereich im Spiel in einem regelmäßigen Raster platziert.

```
function multiplySpectator(scene){

    for(var i=0; i<81; i++){

        var zuschauer = makeSpectator();

        zuschauer.position.y = getRandomY();

        scene.add(zuschauer);

        zuschauerArray.push(zuschauer);

        var rnd = Math.random();
        if(rnd <= 0.5){
            zuschauerOrientation.push(true);
        }
        else{
            zuschauerOrientation.push(false);
        }
    }
}

function setSpectatorPosition(){
    var locationArray = [];
    for(var i = -5; i > -14;i--){
        for(var j = 4; j < 13; j++){
            locationArray.push([i,j]);
        }
    }

    for(var i = 0; i < zuschauerArray.length;i++){
        zuschauerArray[i].position.x = locationArray[i][0];
        zuschauerArray[i].position.z = locationArray[i][1];
    }
}

//zufälliger Y-Wert für Zuschauerposition
function getRandomY(){
    return THREE.Math.randFloat(0, 0.5);
}
```

Aufgaben - Jessika Maier

- Zuschaueranimation „Springen“

Je nach aktueller Orientierung wird der Y-Wert der Position für jeden Zuschauer einmal pro Frame durch Aufruf der nebenstehenden Methode aus der render-Methoder heraus verändert. Sollte der Zuschauer einen der Grenzwerte erreicht haben, wird die Orientierung gewechselt, sodass er sich im nächsten Frame in die entgegengesetzte Richtung bewegt.

```
//Zuschauer Bewegung
function movementSpectator(){
    if(!spawnCubes){
        var jumpSpeed = 0.05;
        var maxY = 0.5;
    }
    else{
        var jumpSpeed = 0.075;
        var maxY = 0.75;
    }

    for( var i=0; i < zuschauerArray.length; i++){

        //true = up, false = down
        if(zuschauerOrientation[i]){
            zuschauerArray[i].position.y += jumpSpeed;
            if(zuschauerArray[i].position.y >= maxY){
                zuschauerOrientation[i] = false;
            }
        }
        else{
            zuschauerArray[i].position.y -= jumpSpeed;
            if(zuschauerArray[i].position.y <= 0){
                zuschauerOrientation[i] = true;
            }
        }
    }
}
```

Aufgaben - J. Maier

- Zaunerstellung,
  -vervielfältigung &
  -positionierung

Alle drei Zaunreihen werden nach dem gleichen Prinzip, wie rechts und unten gezeigt, aus Zaunstücken, die aneinandergereiht werden, erstellt.

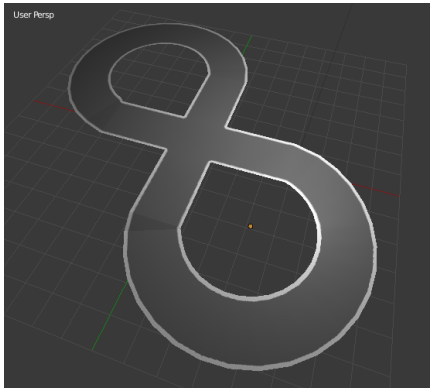
```
function SpawnFencesMitte(scene){
    var startX = 13.9;
    for(var i = 0; i < 32; i++){
        var fence = makeFenceMitte();
        fence.position.x = startX;

        scene.add(fence);

        startX -= 0.9;
    }
}
```

```
//Zäune
function makeFenceMitte(){

    var balkenGeometry = new THREE.BoxGeometry(0.5, 1, 7);
    var balkenMaterial = new THREE.MeshLambertMaterial({color: 0x3B240B});
    var balken = new THREE.Mesh(balkenGeometry, balkenMaterial);

    balken.position.y = 10;

    var balken1Geometry = new THREE.BoxGeometry(0.5, 1, 7);
    var balken1Material = new THREE.MeshLambertMaterial({color: 0x3B240B});
    var balken1 = new THREE.Mesh(balken1Geometry, balken1Material);

    balken1.position.y = 8;

    var balken2Geometry = new THREE.BoxGeometry(0.5, 7, 1);
    var balken2Material = new THREE.MeshLambertMaterial({color: 0x3B240B});
    var balken2 = new THREE.Mesh(balken2Geometry, balken2Material);

    balken2.position.z = -2;
    balken2.position.y = 8;

    var balken3Geometry = new THREE.BoxGeometry(0.5, 7, 1);
    var balken3Material = new THREE.MeshLambertMaterial({color: 0x3B240B});
    var balken3 = new THREE.Mesh(balken3Geometry, balken3Material);

    balken3.position.z = 2;
    balken3.position.y = 8;

    fence = new THREE.Object3D();

    fence.add(balken);
    fence.add(balken1);
    fence.add(balken2);
    fence.add(balken3);

    fence.position.y = 0;
    fence.position.z = 13;
    fence.scale.x *= 0.12;
    fence.scale.y *= 0.12;
    fence.scale.z *= 0.12;
    fence.rotation.y = Math.PI/2;

    return fence;
}
```

Aufgaben - Jessika Maier

- Start/Ziel-Banner
- Kartbahn

Beide Modelle wurden in Blender modelliert und mithilfe von Photoshop texturiert. Sie wurden dann durch nebenstehende Methoden eingebunden.

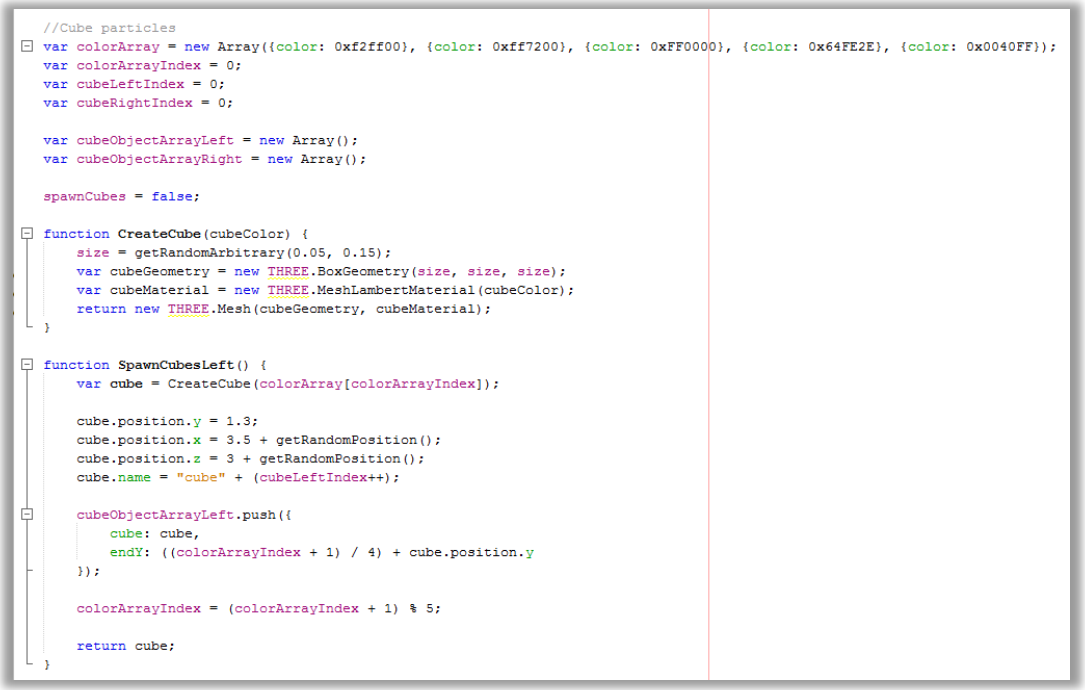



```
function addKartbahn(scene){
    //Kartbahn
    var kartbahnManager = new THREE.LoadingManager();
    kartbahnManager.onProgress = function (item, loaded, total) {
        console.log(item, loaded, total);
    };

    var kartbahn = new THREE.Mesh();
    var kartbahnLoader = new THREE.JSONLoader(kartbahnManager);
    kartbahnLoader.load('models/KartbahnFertig1.json', function (geometry) {
        var tex = new THREE.ImageUtils.loadTexture('models/KartbahnFertig2.png');
        var material = new THREE.MeshBasicMaterial({map:tex});
        kartbahn = new THREE.Mesh(geometry, material);

        kartbahn.position.y += 0.1;
        kartbahn.scale.x *= 1.9;
        kartbahn.scale.z *= 1.9;

        scene.add(kartbahn);
    });
}

function addStartZiel(scene){
    var startZielManager = new THREE.LoadingManager();
    startZielManager.onProgress = function(item, loaded, total){
        console.log(item, loaded, total);
    };
    var startZiel = new THREE.Mesh();
    var startZielLoader = new THREE.JSONLoader(startZielManager);
    startZielLoader.load('models/startziel.json',function(geometry){
        var tex = new THREE.ImageUtils.loadTexture('models/StartZielTextur_1.png');
        var material = new THREE.MeshBasicMaterial({ map : tex} );
        startZiel = new THREE.Mesh(geometry, material);

        startZiel.position.x = -0.5;
        startZiel.position.y += 6;
        startZiel.position.z = -5.5;
        startZiel.scale.x *= 0.3;
        startZiel.scale.y *= 0.3;
        startZiel.scale.z *= 0.3;
        startZiel.rotation.y = -0.5 * Math.PI;
        scene.add(startZiel);
    });
}
```

Aufgaben – Katharina Rupp

- Partikelsystem I (Cube-Spawning)

Die Würfel werden mit einer zufälligen Größe und einer zufälligen Farbe aus dem colorArray erstellt und dem cubeObjectArray hinzugefügt. Pro Tonne gibt es eine SpawnCubes-Methode und ein cubeObjectArray.



```
//Cube particles
var colorArray = new Array({color: 0xf2ff00}, {color: 0xff7200}, {color: 0xFF0000}, {color: 0x64FE2E}, {color: 0x0040FF});
var colorArrayIndex = 0;
var cubeLeftIndex = 0;
var cubeRightIndex = 0;

var cubeObjectArrayLeft = new Array();
var cubeObjectArrayRight = new Array();

spawnCubes = false;

function CreateCube(cubeColor) {
    size = getRandomArbitrary(0.05, 0.15);
    var cubeGeometry = new THREE.BoxGeometry(size, size, size);
    var cubeMaterial = new THREE.MeshLambertMaterial(cubeColor);
    return new THREE.Mesh(cubeGeometry, cubeMaterial);
}

function SpawnCubesLeft() {
    var cube = CreateCube(colorArray[colorArrayIndex]);

    cube.position.y = 1.3;
    cube.position.x = 3.5 + getRandomPosition();
    cube.position.z = 3 + getRandomPosition();
    cube.name = "cube" + (cubeLeftIndex++);

    cubeObjectArrayLeft.push({
        cube: cube,
        endY: ((colorArrayIndex + 1) / 4) + cube.position.y
    });

    colorArrayIndex = (colorArrayIndex + 1) % 5;

    return cube;
}
```

Aufgaben – Katharina Rupp

- Partikelsystem II (Cube-Movement)

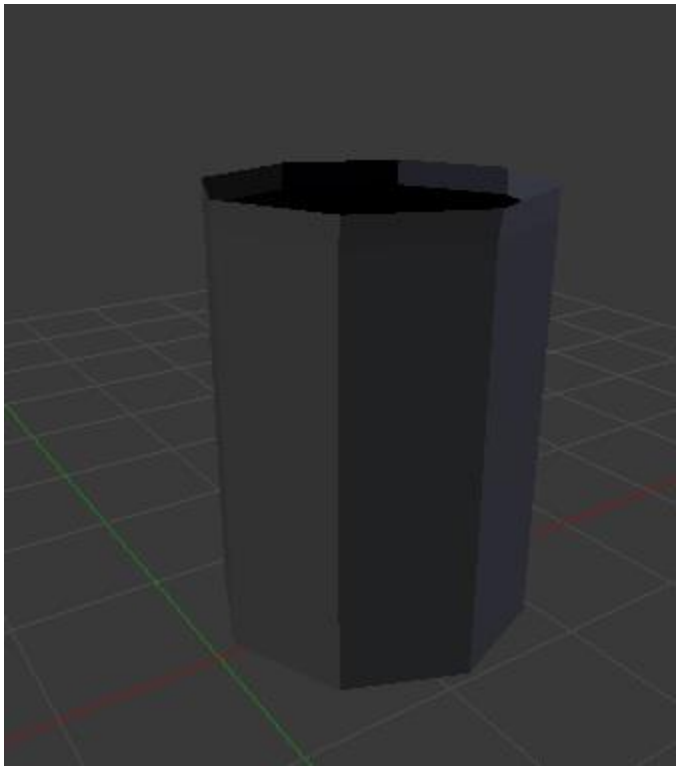
Bei jedem Frame wird bei beiden Tonnen für jeden Würfel im jeweiligen cubeObectArray die jeweilige cubeObejctCallback-Methode aufgerufen. Sie bewegt die Würfel nach oben und entfernt Würfel aus Array und Szene, wenn sie eine Maximalhöhe erreicht haben. In der render-Methode werden, solange im Array weniger als 200 Würfel sind, neue Würfel gespawnt.

```
var cubeObjectCallbackLeft = function (element, index, array) {
    if (element.cube.position.y < element.endY) {
        element.cube.position.y += 0.05;
    }
    else {
        scene.remove(element.cube);

        cubeObjectArrayLeft.splice(index, 1);
    }
};

//Render Function
render();
function render() {
    if (cubeObjectArrayLeft.length < 200 && spawnCubes) {
        scene.add(SpawnCubesLeft());
        scene.add(SpawnCubesLeft());
    }
    cubeObjectArrayLeft.forEach(cubeObjectCallbackLeft);
```

Aufgaben – Katharina Rupp

- Modellierung Tonne

Die Tonne wurde in Blender modeliert, mit einem dunklen LamberMaterial versehen und auf die gleiche Art wie die anderen Objekte in die Szene eingebunden



Aufgaben – Katharina Rupp

- Sounds finden/ erstellen
  und einbinden

Wie unten gezeigt werden die Audio-Dateien in die HTML-Datei eingebunden. Es wird ihnen dann per JavaScript ihr Lautstärkewert festgelegt. Bei Druck auf den Start-Knopf werden der Countdown und die Autogeräusch-Dateien, wie links gezeigt, abgespielt.

```
$(function () {
    $('#autoAudio1')[0].volume = 0;
    $('#autoAudio2')[0].volume = 0;
    $('#zielAudio')[0].volume *= 0.6;
```

```
function OnStartButtonClick(){
    $('#startButton').remove();
    displayCountdown = true;
    $('#countdownText').css("display","inline");
    $('#erklärung').css("display","none");
    $('#countdownAudio')[0].play();
    $('#autoAudio1')[0].play();
    $('#autoAudio2')[0].play();
}
```

```
<audio src="sound/Ziel.mp3" id="zielAudio"></audio>
<audio src="sound/Auto_Lang.wav" id="autoAudio1" loop></audio>
<audio src="sound/Auto_Lang.wav" id="autoAudio2" loop></audio>
<audio src="sound/countdown.wav" id="countdownAudio" onended="OnCountdownEnded()"></audio>
```

Aufgaben – Katharina Rupp

- Sounds finden/ erstellen und einbinden

Wie rechts unten gezeigt, wird die Lautstärke der jeweiligen Geräusche dynamisch an die Geschwindigkeit des Autos angepasst. Wenn das Auto steht, wird das Abspielen der Autogeräusch-Datei pausiert.

Wie rechts oben zu sehen, wird die Zielfanfare abgespielt, wenn eines der Autos ins Ziel einläuft.

```
if(playZielAudio){
    playZielAudio = false;
    document.getElementById("zielAudio").play();
    spawnCubes = true;
}
```

```
//AUTO - AUDIO
function OnCountdownEnded(){
    mayDrive1 = true;
    mayDrive2 = true;
}

var audio1playing = true;
var audio2playing = true;
function controlAutoAudio(){
    if(carSpeed1/carMaxSpeed > 0 && !audio1playing){
        $('#autoAudio1')[0].play();
    }
    if(carSpeed2/carMaxSpeed > 0 && !audio2playing){
        $('#autoAudio2')[0].play();
    }
    $('#autoAudio1')[0].volume = carSpeed1/carMaxSpeed;
    $('#autoAudio2')[0].volume = carSpeed2/carMaxSpeed;
    if(carSpeed1 === 0){
        $('#autoAudio1')[0].pause();
        audio1playing = false;
    }
    if(carSpeed2 === 0){
        $('#autoAudio2')[0].pause();
        audio2playing = false;
    }
}
```

Aufgaben – Katharina Rupp

- Countdown

Wie rechts oben zu sehen, wird bei Druck auf den Start-Button ermöglicht, dass der Countdown angezeigt wird. Während der Countdown angezeigt wird, wird die aktuelle Zeit mit der Startzeit des Countdowns verglichen und bei Bedarf der Countdowntext verändert. Ist der Countdown abgelaufen wird er wieder versteckt.

```
function OnStartButtonClick(){
    $('#startButton').remove();
    displayCountdown = true;
    $('#countdownText').css("display","inline");
    $('#erklärung').css("display","none");
    $('#countdownAudio')[0].play();
    $('#autoAudio1')[0].play();
    $('#autoAudio2')[0].play();
}
```

```
//Render Function
render();
function render() {
    if(displayCountdown){
        updateCountdownText();
    }
```

```
countdownStart = null;
displayCountdown = false;
var setTime = false;
var updateCountdownText  = function()
{
    if(!setTime){
        countdownStart = new Date();
        setTime = true;
    }
    var now = new Date();
    if(now.getTime() - countdownStart < 1000){
        $('#countdownText').text("3");
    }
    else if(now.getTime() - countdownStart < 2000){
        $('#countdownText').text("2");
    }
    else if(now.getTime() - countdownStart < 3000){
        $('#countdownText').text("1");
    }
    else{
        $('#countdownText').css("display","none");
        displayCountdown = false;
    }
};
```

Aufgaben – Hendrik Pastunink

- Fahrbahnberechnung
  - 1. Keyframes

```
var keyframesArray1 = new Array();
var keyframesArray2 = new Array();

function fillKeyframesArray() {
    keyframesArray1[0] = [1, 0, 0, 0];
    pushToKeyFramesArray1(kurvenKeyFrames(new Array(6, 0, -6), 5, Math.PI, keyframesArray1[keyframesArray1.length - 1][3], 4, true));
    pushToKeyFramesArray1(kurvenKeyFrames(new Array(6, 0, -6), 5, Math.PI / 2, keyframesArray1[keyframesArray1.length - 1][3], 4, true));
    pushToKeyFramesArray1(kurvenKeyFrames(new Array(6, 0, -6), 5, 0, keyframesArray1[keyframesArray1.length - 1][3], 4, true));
    pushToKeyFramesArray1(kurvenKeyFrames(new Array(-6, 0, 6), 7, Math.PI, keyframesArray1[keyframesArray1.length - 1][3], 4, false));
    pushToKeyFramesArray1(kurvenKeyFrames(new Array(-6, 0, 6), 7, 3 * Math.PI / 2, keyframesArray1[keyframesArray1.length - 1][3], 4, false));
    pushToKeyFramesArray1(kurvenKeyFrames(new Array(-6, 0, 6), 7, 0, keyframesArray1[keyframesArray1.length - 1][3], 4, false));
```

```
function pushToKeyFramesArray1(array) {
    for (var i = 0; i < array.length; i++) {
        keyframesArray1.push(array[i]);
    }
}
```

Jedes Auto erhält am Ende der Fahrbahnberechnung ein Array mit Wegpunkten. Am Anfang der Fahrbahnberechnung steht jedoch pro Auto ein Array mit Keyframes, zwischen denen später interpoliert wird. Außer dem ersten werden alle Keyframes durch die Methode auf der nächsten Seite berechnet.

Aufgaben – Hendrik Pastunink

- Fahrbahnberechnung
  - 2. Kurvenkeyframes

Die Keyframes werden als gleichmäßig verteilte Punkte auf einem Kreisviertel berechnet. Es wird hierbei nicht nur die Position des Autos gespeichert sondern auch die Rotation, die es bei diesem Keyframe haben soll.

```
function kurvenKeyFrames(mittelPunkt, radius, anfangsWinkel, anfangsRotation, anzahl, clockwise) {
    var keyArray = new Array();
    var winkel = anfangsWinkel;
    var rotation = anfangsRotation;
    if (clockwise) {
        rotation += Math.PI / 2;
    }
    //console.log("Anfang: " +rotation);
    for (var i = 0; i < anzahl + 1; i++, winkel += (Math.PI / 2) / anzahl) {
        var array = new Array();
        array.push(mittelPunkt[0] + (Math.sin(winkel) * radius));
        array.push(0);
        array.push(mittelPunkt[2] + (Math.cos(winkel) * radius));
        //console.log(winkel + ", " +rotation);
        if (clockwise) {
            array.push(rotation);
            rotation -= (Math.PI / 2) / anzahl;
        }
        else {
            array.push(rotation);
            rotation -= (Math.PI / 2) / anzahl;
        }
        keyArray.push(array);

    }

    if (clockwise) {
        var finalArray = new Array();
        for (var i = keyArray.length - 1; i > -1; i--) {
            finalArray.push(keyArray[i]);
        }

        return finalArray;
    }
    return keyArray;
}
```

Aufgaben – Hendrik Pastunink

- Fahrbahnberechnung
  - 3. Interpolieren

Wenn die Keyframes errechnet wurden, werden die Punkte dazwischen interpoliert. Damit die Punkte gleichmäßig verteilt sind wird eine Interpolationskonstante mit dem durch die unten zu sehende getKeyFrameSqrt-Methode errechnet Abstand zwischen den KeyFrames multipliziert.

```
var positionArray1 = new Array();
var positionArray2 = new Array();
var interpolationPointCount = 300;

function fillPositionArray1() {
    for (var i = 0; i < keyframesArray1.length; i++) {
        var lastKeyframe = new Array();
        var nextKeyframe = new Array();

        for (var k = 0; k < keyframesArray1[i].length; k++) {
            lastKeyframe[k] = keyframesArray1[i][k];
            nextKeyframe[k] = keyframesArray1[(i + 1) % keyframesArray1.length][k];
        }
        positionArray1.push(lastKeyframe);
        var inPC = interpolationPointCount * getKeyFrameSqrt(lastKeyframe, nextKeyframe);
        //console.log(inPC);
        for (var j = 1; j < inPC + 1; j++) {
            var x = lastKeyframe[0] + getPositionValue(lastKeyframe[0], nextKeyframe[0], inPC, j);
            var y = lastKeyframe[1] + getPositionValue(lastKeyframe[1], nextKeyframe[1], inPC, j);
            var z = lastKeyframe[2] + getPositionValue(lastKeyframe[2], nextKeyframe[2], inPC, j);
            var rot = lastKeyframe[3] + getPositionValue(lastKeyframe[3], nextKeyframe[3], inPC, j);

            positionArray1.push([x, y, z, rot]);
        }
    }
}
```

```
function getPositionValue(lastKey, nextKey, intPC, n) {
    return (((nextKey - lastKey) / intPC) * n);
}

function getKeyFrameSqrt(keyA1, keyA2) {
    return Math.sqrt(Math.pow(keyA2[0] - keyA1[0], 2) + Math.pow(keyA2[1] - keyA1[1], 2) + Math.pow(keyA2[2] - keyA1[2], 2));
}
```

Aufgaben – Hendrik Pastunink

- Autobewegung

Die Bewegung der Autos wird durch eine Veränderung der Position entlang der Punkte im positionArray simuliert. Je schneller das Auto fährt, desto mehr Punkte werden im positionArray übersprungen.

Wenn die entsprechende Taste gedrückt wird, wird die Geschwindigkeit der Autos erhöht, wenn die Tasten nicht gedrückt werden, wird sie verringert. Dadurch entsteht der Eindruck von Be- und Entschleunigung.

Die moveCar-Methode wird für jedes Auto einmal in der render-Methode aufgerufen.

```
//current Index in positionArray
var index1 = 0;
var index2 = 0;

var carSpeed1 = 0;
var carSpeed2 = 0;

var mayDrive1 = false;
var mayDrive2 = false;

var playZielAudio = true;

carMaxSpeed = interpolationPointCount / 5;

car1Finished = false;
car2Finished = false;

function moveCar(car, number) {
    switch (number) {
        case 1:
            lastIndex1 = index1;

            car.position.x = positionArray1[index1][0];
            car.position.z = -positionArray1[index1][2];
            car.rotation.y = positionArray1[index1][3];

            if (Key.isDown(Key.UP) && mayDrive1) {
                index1 = (index1 + carSpeed1) % positionArray1.length;
                if (carSpeed1 < carMaxSpeed) {
                    carSpeed1++;
                }
            }
            else {
                index1 = (index1 + carSpeed1) % positionArray1.length;
                if (carSpeed1 > 0) {
                    carSpeed1--;
                }
            }
            if (lastIndex1 > positionArray1.length * 0.9 && index1 < positionArray1.length / 10) {
                runde1++;
                console.log("Runde: " + runde1);
            }
            break;
```

Aufgaben – Hendrik Pastunink

- Tastatureingabe

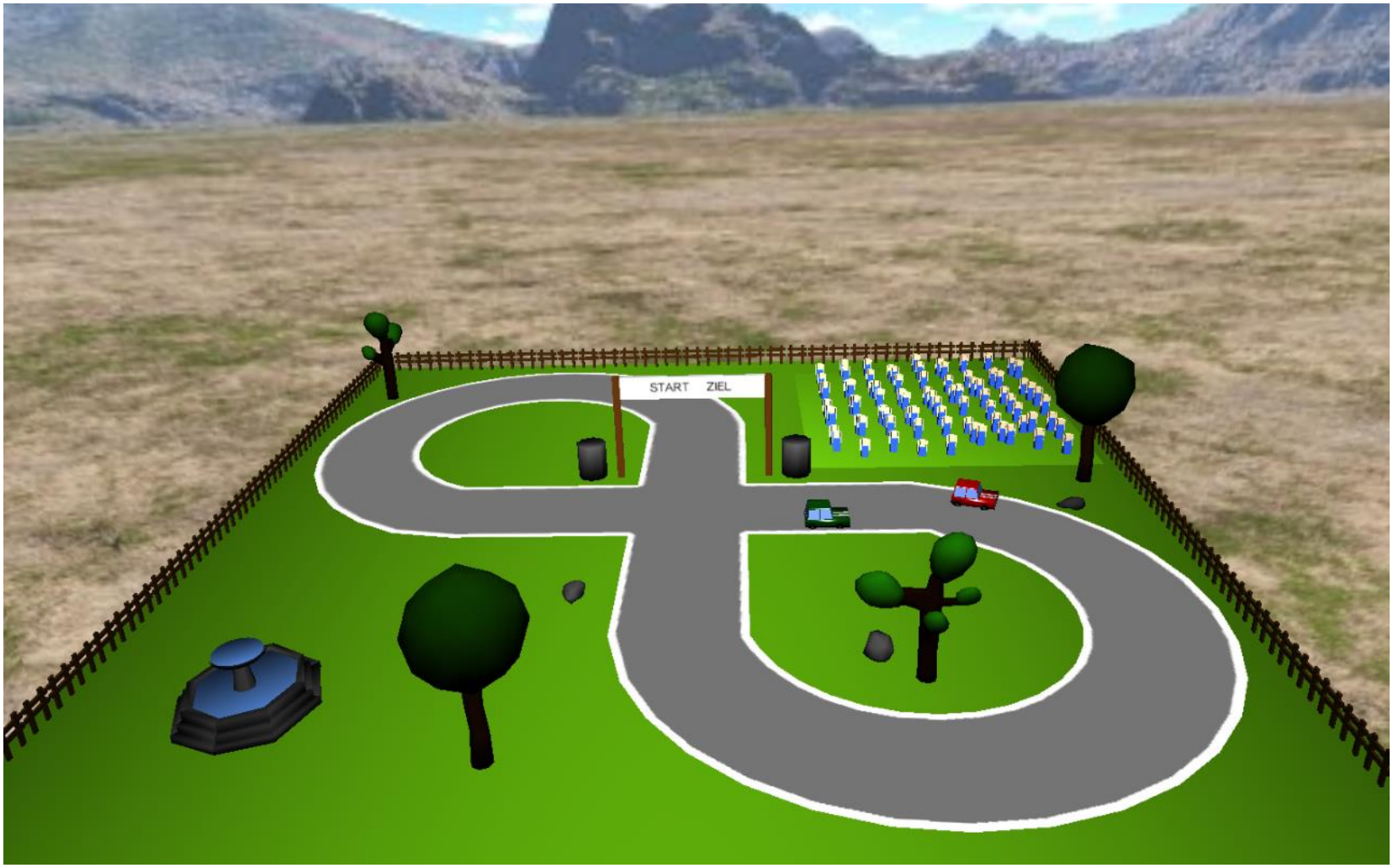
Die auf der unten angegebenen Seite gefundene Methode zum Feststellen von Tastatureingaben haben wir übernommen und an unsere Bedürfnisse angepasst.

```
var Key = {
    _pressed: {},
    SPACE: 32,
    UP: 38,
    isDown: function (keyCode) {
        return this._pressed[keyCode];
    },
    onKeydown: function (event) {
        this._pressed[event.keyCode] = true;
    },
    onKeyup: function (event) {
        delete this._pressed[event.keyCode];
    }
};

window.addEventListener('keyup', function (event) {
    Key.onKeyup(event);
}, false);
window.addEventListener('keydown', function (event) {
    Key.onKeydown(event);
}, false);
```

Mit Hilfe von: http://nokarma.org/2011/02/27/javascript-game-development-keyboard-input/

3 HD-Bilder – Bild 1

3 HD-Bilder – Bild 2



3 HD-Bilder – Bild 3

Video (klicken zum Starten)

Sollte das Video nicht starten, befindet sich eine Videodatei davon in diesem Ordner. Es ist außerdem auf YouTube unter folgendem Link zu sehen, sollte beides nicht funktionieren: https://www.youtube.com/watch?v=_xdRhx7Qrhc