OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

INFORMATIK UND
MATHEMATIK

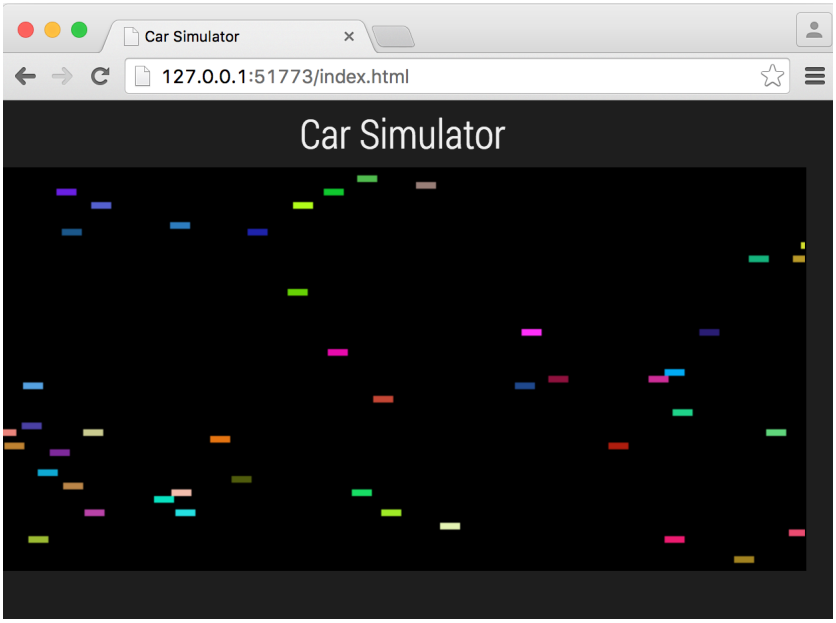
Übungsaufgaben Blatt 7 - Grafikprogrammierung mit Canvas

Medieninformatik Webtechnologien - Prof. Dr. Markus Heckner

Aufgabe 1 : Car-Simulator

Erstellen Sie ein Programm, das den Verkehr auf einer mehrspurigen Straße animiert (vgl. dazu Video aus Grips, neben dieser Aufgabenstellung):

- Alle Fahrzeuge starten am linken Rand der Zeichenfläche.
- Die Fahrzeuge fahren in Spuren, jede Spur ist so hoch wie die Fahrzeuge (die Fahrzeughöhe ist konstant).
- Auf einer Spur können mehrere Fahrzeuge mit unterschiedlichen Geschwindigkeiten fahren.
- Fährt ein Fahrzeug rechts aus dem Bild heraus, so wird seine Position wieder auf den Anfang der selben Spur gesetzt und die Geschwindigkeit wieder zufällig berechnet.
- Jedes Fahrzeug hat eine zufällige Farbe inkl. Alpha-Transparenz (vgl. Hinweise am Ende dieser Aufgabenstellung).
- Die Geschwindigkeit des Fahrzeugs wird ebenfalls zufällig bestimmt (zwischen 2 und 10 Pixel pro Animationsschritt.



Implementieren Sie die Fahrzeuge in der Datei **Car.js** nach dem Muster Konstruktorfunktion und Objektprototyp. Ein Objekt vom Typ **Car** ist vollständig für die Berechnung der Fahrzeugfarben und Positionen zuständig. Aus dem Modul **CarSimulator** werden lediglich die Methoden `update()` und `draw()` der **Car**-Objekte aufgerufen.
Gegeben ist der folgende Code:

```
var carSimulator = (function () {
    var context;

    var CANVAS_WIDTH = 600;
    var CANVAS_HEIGHT = 600;

    var CAR_WIDTH = 15;
    var CAR_HEIGHT = 5;

    var CAR_NUM = 100;
    var BG_COLOR = "black";

    var cars = [];

    function init(carCanvas) {
        context = carCanvas.getContext("2d");

        //setupCars();
        window.requestAnimationFrame(draw);
    }

    function draw() {
        //drawBackground();
        //drawCars();
        window.requestAnimationFrame(draw);
    }

    /* Your code here */

    return {
        init: init
    }

})();
```

```
var Car = function(carWidth, carHeight, canvasWidth,
    canvasHeight, context) {
    this.MAX_SPEED = 10;

    this.carWidth = carWidth;
    this.carHeight = carHeight;

    this.canvasWidth = canvasWidth;
```

```
    this.canvasHeight = canvasHeight;
    this.context = context;

    //this.color = this.createRandomColor();
    //this.speed = this.calculateRandomSpeed();

    //this.yPos = this.createRandomLane();
    this.xPos = 0;
};

Car.prototype.draw = function() {
   //todo
};

Car.prototype.update = function() {
   //todo
};

/* Your code here */
```

Den Code finden Sie auch im Starterprojekt neben dieser Aufgabenstellung.

Hinweise zur Bearbeitung:

Sie müssen in dieser Aufgabe u.a. die Farben zufällig berechnen, um Sie dann z.B. über `context.fillStyle` für den Canvas zu setzen. Das folgende Codebeispiel beschreibt einen Farbwert aus drei Komponenten, den Sie z.B. wie folgt nutzen können:

```
context.fillStyle = "rgba(10, 10, 30, 0.7)";
```

Parameter 1 bis 3 repräsentieren die Farben mit einem Wertebereich zwischen 0 und 255, der letzte Parameter repräsentiert die Transparenz mit einem Wertebereich zwischen 0 und 1. Überlegen Sie, wie Sie die Einzelbestandteile von rgba berechnen können und einen entsprechenden String zusammensetzen können.

Aufgabe 2 : Car-Simulator mit ECMAScript 6

Seit 2015 gibt es den JavaScript Standard ECMAScript 6 (ES6). Diese Version bietet Ihnen eine erweiterte Syntax zum Anlegen eigener Klassen, beispielsweise durch das Schlüsselwort `class`. Somit verfügt ES6 über eine zugänglichere Syntax als die Verwendung des `prototype`-Objekts.

Eine Dokumentation zur Erstellung von Klassen in ECMAScript 6 finden Sie unter dem folgenden Link:

https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Klassen.

***Refactoren* Sie den Car-Simulator so, dass Sie die neuen Syntaxelemente aus ES6 verwenden.**