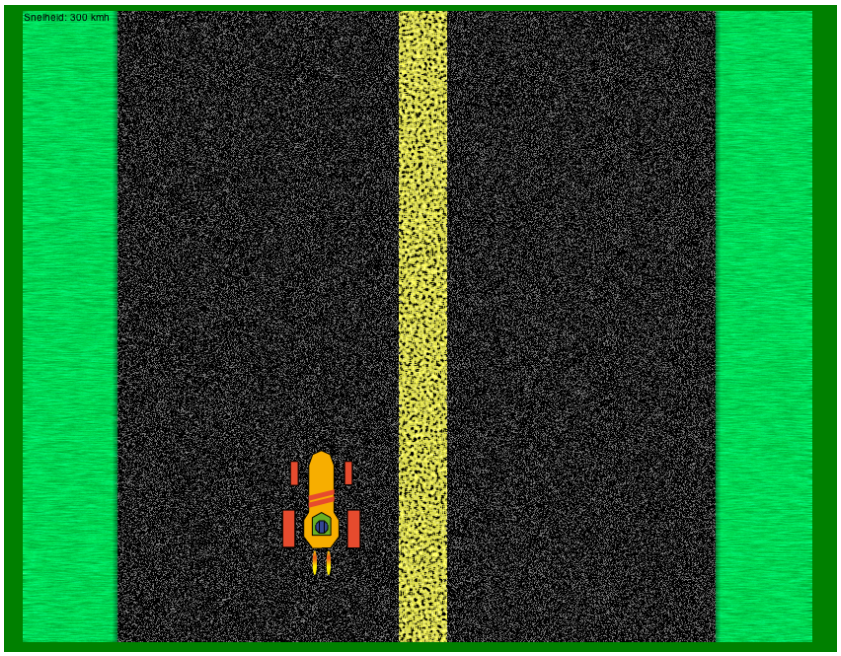
JavaScript Game

In deze lesbrief gaan we kijken hoe je de basis van een eenvoudig spel kunt maken in JavaScript.

Runway racer

Voorbeeld van een eenvoudige basis voor een spel is het fantastische Runway Racer. Met JavaScript wordt er op een HTML canvas een achtergrond getekend en een raceauto. Door middel van de pijltjestoetsen kan de speler gas geven zodat de auto snelheid krijgt. Naar links en naar rechts sturen kan ook met de pijltjestoetsen.

Als de auto sneller gaat, begint de auto ook te trillen. Meer 'spel' elementen zitten er niet in. Om er echt een spel van te maken zou er bijvoorbeeld een computer gestuurde race-auto toegevoegd kunnen worden om er een wedstrijd van te maken, of het kan worden uitgebouwd tot een spel waarbij er objecten ontweken moeten worden.



De opbouw van het spel

Het spel bestaat uit HTML, CSS en JavaScript. Met name het JavaScript gedeelte is van belang voor het spel.

HTML

In de HTML wordt net als in de vorige lesbrief een canvas tag gebruikt waar het spel op wordt getekend. We gebruiken de 'document ready' functie van jQuery om er voor te zorgen dat de HTML pagina volledig is geladen door de browser voordat we op het canvas gaan tekenen.

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Runway racer</title>
<script src="http://code.jquery.com/jquery-latest.min.js" type="text
    /javascript"></script>
<script src="script.js"></script>
<link href="style.css" rel="stylesheet" />
</head>
<body>
<div id="container">
  <canvas id="game" width="800" height="640">
    </canvas>
</div>
<script>
$(document).ready(function() {
    init();
  initAnimation();
});
</script>
</body>
</html>
```

CSS Stylesheet

De CSS stylesheet is ook hetzelfde als in de vorige lesbrief:

```
@charset "UTF-8";
/* CSS Document */
html {
  background: black;
  min-height: 100%;
}

#container {
  width: 870px;
  margin-left: auto;
  margin-right: auto;
}
#game {
  width: 800px;
  height: 640px;
}
```

JavaScript

Het grootste gedeelte van het spel is de JavaScript. We behandelen de belangrijkste functies per stuk, de (bijna) complete code volgt daarna. Dat is de code inclusief de initialisatie functie en declaratie van variabelen.

De animatie loop

Voor actiespellen zoals dit is het belangrijk dat het scherm steeds opnieuw wordt opgebouwd, optimaal is het als dit met 60fps wordt gedaan. Het is afhankelijk van wat er allemaal moet gebeuren binnen het spel en de kracht van de computer of dat aantal van 60fps wordt gehaald.

In de JavaScript code zorgt de **requestAnimationFrame** functie hiervoor. Om er voor te zorgen dat dit in alle browsers werkt moet je gebruik maken van prefixes. Het belangrijkste stukje code is de **animLoop** functie, die wordt 60 keer per seconde aangeroepen en roept zelf weer de **draw** functie aan die het scherm op gaat bouwen:

```
function initAnimation() {
  // animatie
  // vraag aan de browser om maximaal 60 fps te animeren
  window.requestAnimFrame = (function(callback){
    return window.requestAnimationFrame ||
    window.webkitRequestAnimationFrame ||
    window.mozRequestAnimationFrame ||
    window.oRequestAnimationFrame ||
    window.msRequestAnimationFrame ||
    function(callback){
      window.setTimeout(callback, 1000 / 60);
    };
  })();

  (function animloop(){
    requestAnimFrame(animloop);
    // belangrijkste regel: 'draw' wordt 60 keer per seconde
        aangeroepen om het scherm te tekenen!
    draw();
  })();
}
```

De **animLoop** functie zorgt voor de animatie loop. Het belangrijkste wat in ons geval moet gebeuren is het tekenen van het scherm. De **draw** functie zorgt hiervoor.

De schermopbouw tekenen

De **draw** functie zorgt ervoor dat het scherm leeg wordt gemaakt met de **clearRect** functie, daarna wordt de **drawBackground** die de achtergrond laat zien. De snelheid van de auto wordt in het scherm geschreven met de **fillText** functie.

Als dit is gebeurd wordt de auto getekend. De auto kan op scherm naar links en rechts bewegen, dit doen we door de context te *transleren* naar de juiste positie met de **ctx.translate** aanroep. Transleren is een ander woord voor verschuiven. Je verschuift de context naar een nieuwe uitgangspositie waar je wilt gaan tekenen.

Door te transleren is het eenvoudiger om de auto te tekenen, de coordinaten bij het tekenen blijven dan altijd hetzelfde en hoeven niet opnieuw berekend te worden.

Op deze manier kunnen we ook roteren en transformeren. Voor dat we transleren, roteren of transformeren is het belangrijk om de context op te slaan om er voor te zorgen dat hetgene wat we al hebben getekend niet mee verschuift of draait. Dit doen we met de **ctx.save** functie.

Als we de auto hebben getekend op het canvas roepen we tot slot een (onbelangrijk) stukje 'artificial intelligence' aan om de auto wat te laten trillen als hij sneller gaat.

```
// het tekenen van het scherm
function draw() {
  var ctx = canvas.getContext("2d");
  // canvas leeg maken, het canvas is 800px breed en 640px hoog
  ctx.clearRect(0, 0, 800, 640);
  // teken de bewegende achtergrond
  drawBackground(ctx);
  // schrijf snelheid op het scherm omgerekend naar km/h op het
      canvas
  ctx.fillText("Snelheid: " + Math.round(speed * (300/40)) + " kmh",
       1, 10);
    // bewaar deze situatie
  ctx.save();
  // transleer de context, zodat de auto op de juiste plaats wordt
      getekend
  ctx.translate(x, y);
  // teken de auto
  drawCar(ctx);
  // artificial intelligence aanroepen (trillen van de auto)
  ai();
}
```

De achtergrond

Om het idee te geven dat de auto aan het rijden is, verplaatsen we de auto niet maar wordt de achtergrond verschoven. De afbeelding die moet dus hoger zijn dan de hoogte van het canvas. Bijvoorbeeld twee keer zo hoog. Meer mag ook om te voorkomen dat het opvalt dat er steeds dezelfde weg voorbij komt.

Om de achtergrond te herhalen moeten we bijhouden hoever we de achtergrond hebben verschoven. Als er niet verder geschoven kan worden beginnen we opnieuw. Als de auto sneller gaat, moet de achtergrond sneller scrollen, daarom wordt de variabele *speed* gebruikt om de verschuiving te bepalen.

Deze manier werkt, zij het met meerdere achtergronden, goed bij platform spellen zoals Mario.

```
// bewegende achtergrond tekenen
// de afbeelding van de auto is 1280px hoog,
// iedere keer wordt het plaatje verschoven,
// en als helemaal tot bovenaan is verschoven
// opnieuw getekend
function drawBackground(ctx) {
  ctx.drawImage(bgImage,0 ,bgOffset);
    bgOffset += speed / 2;
    if (bgOffset > 0) {
    bgOffset = -640;
    }
```

```
}
```

Besturing met het toetsenbord

Om de auto sneller te laten rijden kan het toetsenbord gebruikt worden. Iedere keer als er een toets wordt ingedrukt wordt er in de **handleKeyboardInput** functie gekeken of er een van de pijltjestoetsen is ingedrukt. Als dat het geval is, wordt de bijbehorende functie aangeroepen, bijvoorbeeld up in het geval van het pijltje omhoog.

De manier waarop dit gebeurt heet een **switch**. Dit lijkt op een **if** functie maar is in sommige gevallen, waarbij er vele mogelijkheden zijn handiger en overzichtelijker. Vergeet bij iedere **case** niet af te sluiten met een **break**, anders wordt niet alleen de code van de huidige case uitgevoerd, maar ook de code van de opvolgende case totdat er een break gevonden wordt.

Andere toetsenbordcodes voor bijvoorbeeld spatie et cetera kun je o.a. hier vinden: `http://goo.gl/8F6zS`.

```
// pijltjes toetsen
function handleKeyboardInput(evt) {
    evt = evt || window.event;
    switch (evt.keyCode) {
    case 37:    // pijltje links
      left();
      break;    // break niet vergeten!
    case 38:    // pijltje omhoog
      up();
      break;
    case 39:    // pijltje rechts
      right();
      break;
    case 40:    // pijltje omlaag
      down();
      break;
    }
}
```

De gehele code

Naast de functies die hierboven staan zijn er nog de variabelen declaraties, de init functie en het tekenen van de auto. Hieronder de complete listing van runway racer, op het tekenen van de auto na. Het tekenen van de auto gebeurd door de bekende commando's op het canvas zoals ook in de vorige lesbrief te lezen is.

```
1  // variabelen
2  var canvas;
3  // het x coordinaat van de auto
4  var x = 205;
5  // het y coordinaat van de auto
6  var y = 450;
7  // initiele snelheid, 0 = stilstand
8  var speed = 3;
9  // achtergrondplaatje wordt onderaan getekend
10 var bgOffset = -640;
11 var bgImage = new Image();
```

```
12
13 function init() {
14   // canvas met het id "game" opvragen uit HTML
15   canvas = document.getElementById("game");
16   // pijltjestoetsen afhandeling regelen...
17   document.onkeydown = handleKeyboardInput;
18   // achtergrond plaatje inladen
19   bgImage.src ="road.png";
20 }
21
22 function initAnimation() {
23   // animatie
24   // vraag aan de browser om maximaal 60 fps te animeren
25   window.requestAnimFrame = (function(callback){
26     return window.requestAnimationFrame ||
27     window.webkitRequestAnimationFrame ||
28     window.mozRequestAnimationFrame ||
29     window.oRequestAnimationFrame ||
30     window.msRequestAnimationFrame ||
31     function(callback){
32       window.setTimeout(callback, 1000 / 60);
33     };
34   })();
35
36   (function animloop(){
37     requestAnimFrame(animloop);
38     draw();
39   })();
40 }
41
42
43 // het tekenen van het scherm
44 function draw() {
45   var ctx = canvas.getContext("2d");
46   // canvas leeg maken, het canvas is 800px breed en 640px hoog
47   ctx.clearRect(0, 0, 800, 640);
48   // teken de bewegende achtergrond
49   drawBackground(ctx);
50   // schrijf snelheid op het scherm omgerekend naar km/h op het
         canvas
51   ctx.fillText("Snelheid: " + Math.round(speed * (300/40)) + " kmh",
          1, 10);
52     // bewaar deze situatie
53   ctx.save();
54   // transleer de context, zodat de auto op de juiste plaats wordt
         getekend
55   ctx.translate(x, y);
56   // teken de auto
57   drawCar(ctx);
58
59   // artificial intelligence aanroepen (trillen van de auto)
60   ai();
61 }
62
63 // pijltjes toetsen
64 function handleKeyboardInput(evt) {
65     evt = evt || window.event;
66     switch (evt.keyCode) {
67     case 37:    // pijltje links
68       left();
69       break;
70     case 38:    // pijltje omhoog
```

```
71        up();
72        break;
73      case 39:    // pijltje rechts
74        right();
75        break;
76      case 40:    // pijltje omlaag
77        down();
78        break;
79      }
80  }
81
82  function left() {
83    if (speed > 0) {
84      x-=speed/4;
85    }
86  }
87
88  function right() {
89    if (speed > 0) {
90      x+=speed/4;
91    }
92  }
93
94  function up() {
95    speed++;
96    // nooit sneller dan 40
97    if (speed > 40) {
98      speed = 40;
99    }
100 }
101
102 function down() {
103   speed--;
104   // snelheid is nooit negatief
105   if (speed < 0) {
106     speed = 0;
107   }
108 }
109
110
111 // bewegende achtergrond tekenen
112 // de afbeelding van de auto is 1280px hoog,
113 // iedere keer wordt het plaatje verschoven,
114 // en als helemaal tot bovenaan is verschoven
115 // opnieuw getekend
116 function drawBackground(ctx) {
117   ctx.drawImage(bgImage,0 ,bgOffset);
118     bgOffset += speed / 2;
119     if (bgOffset > 0) {
120     bgOffset = -640;
121     }
122 }
123
124 function ai() {
125   // wat random trilling afhankelijk van de snelheid....
126   if (speed > 4) {
127     var trilling = speed / 10;
128     x += -(trilling / 2) + Math.random() * trilling;
129     y += -(trilling / 2) + Math.random() * trilling;
130   }
131 }
132
```

```
133
134 // auto tekenen
135 function drawCar(ctx) {
136   // teken op het canvas de auto
137   // (weggelaten)
138 }
```

Opdrachten

Opdracht 1: Maak je eigen 'spel' Maak een begin van een eigen spel dat ook gebruik maakt van een canvas en van de gameloop zoals in dit spel. Bedenk eerst wat voor spel je wilt maken, maak schetsen en beschrijf wat je wilt maken. Laat je idee en schetsen aan je docent zien ter goedkeuring.

Je kunt denken aan een platform spel, een racespel of aan een ander soort spel.

Maak daarna alle vector tekeningen, bitmaps en html en css. Schrijf vervolgens de JavaScript code. Test alles goed voor je het inleverd!

Opdracht 2: Maak de webpagina af Maak een mooie website van je spel, alleen het canvas is zo kaal.

Beoordeling

De uitwerkingen van deze opdrachten inclusief schetsen en beschrijving van je game moeten worden ingeleverd. Je krijgt hiervoor een cijfer.