Rich Media Development

# Inleiding

* **canvas**: een rechthoek in je pagina waarin je met JavaScript alles kan tekenen wat je wilt
* **SVG**: een tekst-gebaseerde grafische taal die afbeeldingen beschrijft met vectorvormen, tekst en ingebedde rasterafbeeldingen
* **WebGL**: een JavaScript API om interactieve 3D computer graphics en 2D graphics mee te renderen binnen elke compatibele webbrowser, zonder het gebruik van plug-ins.

|  |  |  |
| --- | --- | --- |
| Verhoudingen | Van graden naar radialen | Van radialen naar graden |
| |  |  | | --- | --- | | Graden | Radialen | | 0° | 0 | | 90° | PI / 2 | | 180° | PI | | 360° | 2 \* PI | |  |  |

# Graden/radialen

# Canvas

## Paths

**// Beginnen**

context.beginPath();

**// Vul- en omtrekkleuren instellen**

context.strokeStyle = "#FFFF00";

context.fillStyle = "#FFFF00";

**// De onzichtbare cursor verplaatsen naar x = 100, y = 100**

context.moveTo(100, 100);

**// Een lijn tekenen van de onzichtbare cursor naar x = 100, y = 200**

context.lineTo(100, 200);

**// Eindigen**

context.closePath();

### Linecap

= opmaak aan het einde van de lijn

* butt (recht)
* round (afgerond)
* square (recht, iets langer dan butt)

### Linejoin

= opmaak van de knooppunten in een lijn

* miter join (rechte hoek)
* round join (afgeronde hoek)
* bevel join (afgeplatte hoek)

## Arc

= een volledige/deel van een cirkel tekenen

* Deel v.e. cirkel: context.arc(x, y, r, startHoek, eindHoek, tegenDeKlok);
* Volledige cirkel: context.arc(x, y, r, 0, 2 \* Math.PI);

## Rectangle

= een rechthoek tekenen

* rect → tekent een rechthoek (moet tussen beginPath en endPath staan)
* fillRect → tekent een gevulde rechthoek a.d.h.v. de fillStyle
* strokeRect → tekent de omtrek van een rechthoek a.d.h.v. de strokeStyle
* clearRect → wist alle pixels binnen de rechthoek

## Curves

= gebogen lijnen

### Quadratic curve

Een kwadratische Bézier-curve vereist twee punten. Het eerste punt is een besturingspunt dat wordt gebruikt bij de berekening en het tweede punt is het eindpunt van de curve.

|  |  |
| --- | --- |
| http://www.html5canvastutorials.com/demos/tutorials/html5-canvas-quadratic-curves/html5-canvas-quadratic-curves-diagram.png | context.quadraticCurveTo(controlX, controlY, endingX, endingY); |

### Bézier curve

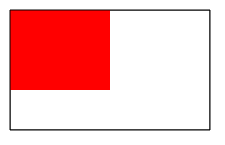
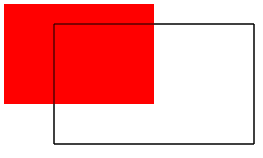
Een kubieke Béziercurve vereist drie punten. De eerste twee punten zijn controlepunten die gebruikt worden in de berekening en het laatste punt is het eindpunt van de curve.

|  |  |
| --- | --- |
| http://www.html5canvastutorials.com/demos/tutorials/html5-canvas-bezier-curves/html5-canvas-bezier-curves-diagram.png | context.bezierCurveTo(control1X, control1Y, control2X, control2Y, endingX, endingY); |

## Clipping

= het beperken van de tekenruimte tot een bepaalde vorm

// Clip een rechthoeking gebied

context.beginPath();

context.rect(50, 20, 200, 120);

context.stroke();

context.clip();

// Teken een rechthoek na clip()

context.fillStyle = "red';

context.fillRect(0, 0, 150, 100);

## fillStyle / strokeStyle

= vulstijl / omtrekstijl

* werkt met:
  + kleur (CSS kleurwaarde)
  + gradient (createLinearGradient / createRadialGradient)
  + pattern (createPattern)

## Context state

= een momentopname van alle stijlen en transformaties die zijn toegepast, bestaat uit het volgende:

* transformaties
* clip-gebieden
* canvas-attributen

save() Voegt de huidige staat toe aan de stack

restore() Haalt de bovenste staat van de stack en herstelt de context naar deze staat

## Transformaties

### Translate

verplaatst het hele tekengebied

context.translate(x, y);

### Scale

vergroot/verkleint het hele tekengebied

context.scale(xMultiplier, yMultiplier);

### Rotate

roteert het hele tekengebied

context.rotate(radialen);

## scaleOnHiDPI

= functie in CanvasHelper.js die de breedte- en hoogteattributen aanpast in CSS en HTML zodat de canvastekening scherm weergegeven wordt op schermen met alle pixel densities.

* uit te voeren voor het tekenen:

this.update();

this.canvas.scaleOnHiDPI();

this.draw();

## Afbeeldingen

var image = new Image();

image.addEventListener("load", handleImageLoad);

image.src = "logo.png";

function handleImageLoad(event) {

var image = event.target,

x = 0, y = 0,

w = image.width \* 0.5, h = image.height \* 0.5;

context.drawImage(image, x, y, w, h);

}

## Tiles

zie code in de slides van hoorcollege 3

## Patterns

var patternImage = new Image();

patternImage.addEventListener("load", handleImageLoad);

patternImage.src = "pattern.png";

function handlePatternLoad(event) {

var image = event.target,

pattern = context.createPattern(image, "repeat");

context.fillStyle = pattern;

context.fillRect(0, 0, 500, 500);

*OF*

context.strokeStyle = pattern;

context.lineWidth = 80;

context.strokeRect(100, 100, 300, 300);

}

## Text

context.fillText(text,x,y)

tekent 'gevulde' tekst op het canvas

strokeText(text,x,y)

tekent ongevulde tekst op het canvas

measureText(text)

berekent de breedte van tekst zonder het te tekenen op het canvas

context.font([argumenten])

bepaalt het lettertype en eventuele stijlen in één string

(zie http://www.w3schools.com/tags/canvas\_font.asp)

ImageData

= een object met de onderliggende pixelwaarden van (een bep. gebied op) het canvas

var imageData = context.getImageData(x, y, width, height);

context.putImageData(imageData, x, y);

imageData.data = een Uint8ClampedArray met de pixeldata in RGBA-volgorde,

met integerwaarden tussen 0 en 255

## Filters

= itereren over een array met pixeldata en individuele waarden aanpassen, waardoor een effect wordt toegepast op de gehele afbeelding

* **convolution**: the operation where you take the sum of products of elements from two 2D functions, where you let one of the two functions move over every element of the other function.
* **meer info**: http://lodev.org/cgtutor/filtering.html

### Voorbeeld

Filters.brightness = function (pixels, adjustment) {

var d = pixels.data;

for (var i = 0; i < d.length; i += 4) {

d[i] += adjustment;

d[i + 1] += adjustment;

d[i + 2] += adjustment;

}

return pixels;

};

# HTML5 Video

<video controls autoplay>

<source src="video/small.mp4"></source>

<source src="video/small.webm"></source>

<source src="video/small.ogv"></source>

<source src="video/small.3gp"></source>

</video>

## Filters

var canvas, video;

function init() {

canvas = new CanvasHelper("canvasOne");

video = document.getElementById('videoOne');

video.addEventListener('play', handleVideoPlay, false);

video.addEventListener('pause', handleVideoPause, false);

}

function handleVideoPlay(event) {

if (video.paused || video.ended) return false;

draw();

}

function handleVideoPause(event) {

window.cancelAnimationFrame(draw);

}

function draw() {

canvas.context.drawImage(video, 0, 0);

window.requestAnimationFrame(draw);

}

function drawGray() {

canvas.context.drawImage(video, 0, 0);

var pixels = canvas.context.getImageData(0, 0, canvas.width, canvas.height);

canvas.context.putImageData(Filters.greyscale(pixels), 0, 0);

window.requestAnimationFrame(drawGray);

}

# Webcam

= momenteel niet overal ondersteund en dus niet betrouwbaar

navigator.getMedia = (navigator.getUserMedia || navigator.webkitGetUserMedia || navigator.mozGetUserMedia || navigator.oGetUserMedia || navigator.msGetUserMedia);

function captureWebcam(video, audio){

navigator.getMedia({

video: video,

audio: audio

}, stream, error);

function stream(stream) {

if(navigator.mozGetUserMedia) {

video.mozSrcObject = stream;

} else {

var vendorURL = window.URL || window.webkitURL;

video.src = vendorURL.createObjectURL(stream);

}

video.play();

}

function error(error) {

Debugger.log("An error occured! " + error);

}

# Collision detection

zie: <https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection>

# Edge detection

= het detecteren van randen van objecten op een afbeelding

zie: <http://codepen.io/taylorcoffelt/pen/EsCcr>

# Audio

var audio = new Audio('audio\_file.mp3');

audio.play();

## Audio pool

= verzamelobject voor alle geluidseffecten binnen je spel

function SoundPool(maxSize) {

var size = maxSize; // Max sounds allowed in the pool

var pool = [];

this.pool = pool;

var currSound = 0;

// Populates the pool array with the given sound

this.init = function(object) {

if (object == "laser") {

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

// Initalize the sound

laser = new Audio("sounds/laser.wav");

laser.volume = .12;

laser.load();

pool[i] = laser;

}

}

else if (object == "explosion") {

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

var explosion = new Audio("sounds/explosion.wav");

explosion.volume = .1;

explosion.load();

pool[i] = explosion;

}

}

};

// Plays a sound

this.get = function() {

if(pool[currSound].currentTime == 0 || pool[currSound].ended) {

pool[currSound].play();

}

currSound = (currSound + 1) % size;

};

}

# Cross-origin error

= een fout die zich voordoet bij het bewerken van afbeeldingen van een vreemde server

Oplossing: het bestand hosten op dezelfde server als de website

# Box2D

## Features

* Zwaartekracht
* Wrijving
* Elasticiteit
* Mechanica van materialen
* Collision detection

## Begrippen

|  |  |
| --- | --- |
| shape | A 2D geometrical object, such as a circle or polygon. |
| rigid body | a chunk of matter that is so strong that the distance between any two bits of matter on the chunk is constant. They are hard like a diamond. |
| fixture | binds a shape to a body and adds material properties such as density, friction, and restitution. |
| constraint | a physical connection that removes degrees of freedom from bodies (e.g. pinning to a wall). |
| contact constraint | a special constraint designed to prevent penetration of rigid bodies and to simulate friction and restitution. |
| joint | a constraint used to hold two or more bodies together. |
| joint limit | restricts the range of motion of a joint. (e.g. the human elbow only allows a limited angles). |
| joint motor | what drives the motion of the connected bodies according to the joint's degrees of freedom. |
| physics world | a collection of bodies, fixtures, and constraints that interact together. |
| solver | used to advance time and to resolve contact and joint constraints. |
| continuous collision | continuously checking the position of an object while it is moving, to prevent it from passing through another object at high speed. |

### World

= container voor alle Box2D-lichamen; definieert de zwaartekracht

world = new b2World(

new b2Vec2(0, 10), // Set gravity

true // Allow sleep

);

### Body

#### 3 types:

* static body: beweegt niet en reageert niet op collisions
* dynamic body: beweegt en reageert op collisions
* kinematic body: beweegt maar reageert niet op collisions

var ballDef = new b2BodyDef;

ballDef.type = b2Body.b2\_dynamicBody;

### Shape

#### 4 types:

* Polygon
  + Box Polygon
  + Custom Polygon (enkel convex polygon = polygon zonder naar binnen gerichte hoeken)
* CircleShape
* ChainShape
* EdgeShape

### Joint

* tussen dynamic én static bodies
* parent class = abstracte ‘klasse’
* zijn dynamisch (at runtime) aanpasbaar

#### Common properties

* bodyA
* bodyB
* anchor points (verbindingspunten)
* limits (begrenzingen)

#### Types

* pivot joint (vorm A hangt vast aan één bep. punt op vorm B)
* distance joint (vorm A en B zijn verbonden door een lijn met vaste lengte)
* piston, friction, weld, wheel, pulley, touch, rope & gear joint

## Units

canvas gebruikt pixels en FPS, Box2D een alternatief metrisch systeem

* omzetten!

|  |  |  |
| --- | --- | --- |
|  | Box2D | Canvas |
| Afstand | meter | pixel |
| Massa | kilogram | n.v.t. |
| Tijd | seconde | frames (FPS) |

## Code examples

zie slides hoorcollege 7

# Animation

setInterval(function, delay)

Start een functie te herhalen met een delay in milliseconden

setTimeOut(function, delay)

Voer een functie eenmalig uit met een delay in milliseconden

requestAnimationFrame(callback)

Vertel de browser dat je een functie wenst aan te roepen voor de browser een repaint doet van het scherm

codevoorbeeld → zie slides van hoorcollege 3

# Events

### 3 mogelijkheden

* <body onload="init()">
* window.onload = init;
* window.addEventListener("load", init, false)

### Beste manier

window.addEventListener(event, function, useCapture)

* useCapture: bepaalt of het event uitgevoerd moet worden in de **capturing**- of **bubbling**fase

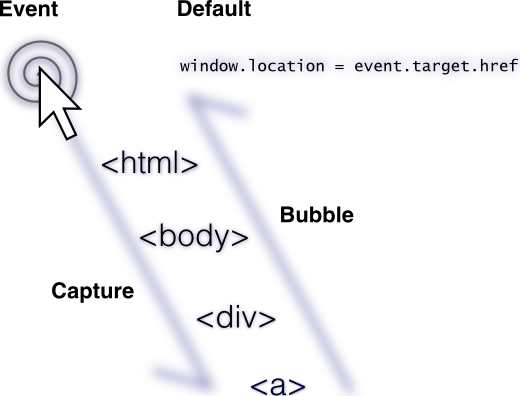
## Event propagation

in het geval dat een event zich voortdoet op een element binnen een ander element, en deze elementen allebei een listener voor dit event hebben geregistreerd, bepaalt de **event propagation mode** welk element als eerste het event ontvangt

* **bubbling**: het event wordt opgevangen door het diepst geneste element en vervolgens doorgegeven van binnen naar buiten toe
* **capturing**: het evenement wordt opgevangen door het buitenste element en vervolgens doorgegeven naar binnen toe.

### Voorbeeld

|  |  |
| --- | --- |
| <div>  <ul>  <li></li>  </ul>  </div> | * bubbling: li > ul > div * capturing: div > ul > li |



### Event object

voor een lijst van properties, methodes en events zie <http://www.w3schools.com/jsref/dom_obj_event.asp>

# JavaScript, the right way

## Strict mode

= code wordt uitgevoerd zonder bepaalde fouten te negeren die anders getolereerd worden

function doeIets() {

"use strict";

doeIetsAnders();

}

## Tools

### JSHint

= verifieert je codestijl, -patronen en eventuele fouten in je code

### Chrome Debugger

met breakpoints, pauzeren van uitvoering, etc.

## Object-oriented JavaScript

### Constructor

var Person = function(firstName) {

this.firstname = firstName;

};

### Statische functies

var Debugger = function() {}

Debugger.log = function(message) {

try {

console.log(message);

} catch (exception) {

return;

}

}

### Instantiefuncties

Person.prototype.sayHello = function() {

console.log("Hello, I'm " + this.firstName);

};

var person = new Person("Alice");

person.sayHello();

# Wiskunde voor ontwikkelaars

## Stelling van Pythagoras

In een rechthoekige driehoek is de som van de kwadraten van de lengtes van de rechthoekszijden gelijk aan het kwadraat van de lengte van de schuine zijde.

|  |  |
| --- | --- |
|  | a² + b² = c² |

## Goniometrie

Berekenen van de sinus, cosinus of tangens van een rechthoekige driehoek

|  |  |
| --- | --- |
| sinus |  |
| cosinus |  |
| tangens |  |

### http://www.spage.nl/data/image/goniometrische_formules_194.jpgEzelsbruggetje

SOS Sinus = Overstaand / Schuin

CAS Cosinus = Aaanliggend / Schuin

TOA Tangens = Overstaand / Aanliggend

## Boogtangens

## Vectors

## 3D Vectors

# Particles

## Vector

function Vector() {

this.x = 0;

this.y = 0;

}

Vector.prototype.set = function (X, Y) {

this.x = X;

this.y = Y;

};

Vector.prototype.add = function (vector2) {

this.x += vector2.x;

this.y += vector2.y;

};

## Particle

function Particle(image) {

this.size = Math.random() \* 10 + 15; // De grootte van de particle

this.particle = image; // De afbeelding van de particle

this.loc = new Vector(); // De locatie van de particle

this.vel = new Vector(); // De snelheid van de particle

this.acc = new Vector(); // De versnelling van de particle

this.lifespan = Math.random() \* 250; // De leeftijd van de particle

}

Particle.prototype.update = function() {

this.vel.add(this.acc);

this.loc.add(this.vel);

this.lifespan -= 1;

};

Particle.prototype.draw = function(context) {

context.drawImage(this.particle, this.loc.x, this.loc.y);

};

Particle.prototype.dead = function() {

if (this.lifespan < 0) {

return true;

} else {

return false;

}

};

# SVG

## Waarom?

|  |  |
| --- | --- |
| Canvas | SVG |
| script | XML / DOM |
| pixels | vectors |
| performance | animations, accessibility |

## Hoe?

aanmaken in vectorprogramma zoals Adobe Illustrator of Inkscape en exporteren

## Begrippen

meer info: https://sarasoueidan.com/blog/structuring-grouping-referencing-in-svg/

|  |  |
| --- | --- |
| <group> element | used for logically grouping together sets of related graphical elements. |
| <defs> element | used to **define** elements **without directly rendering** them and that element will serve as a **template** for future use. (e.g. a gradient) |
| <symbol> element | * used to define new scaleable **viewports** with the viewBox and preserveAspectRatio attributes * serves as a **template** that is instantiated using the use element |
| <use> element | lets you **reuse** existing elements, giving you a similar functionality to the copy-paste functionality in a graphics editor. |

## SVG in development

### Embedding

#### HTML (extern)

<img src="circle.svg" height="100" alt="Nice green circle" />

#### HTML (inline)

<svg width="100" height="100">

<circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />

</svg>

#### CSS (extern)

background-image: url('circle.svg');

#### CSS (inline)

background-image: url("data:image/svg+xml;utf8,<svg xmlns='http://www.w3.org/2000/svg' width='10' height='10'><linearGradient id='gradient'><stop offset='10%' stop-color='%23F00'/><stop offset='90%' stop-color='%23fcc'/> </linearGradient><rect fill='url(%23gradient)' x='0' y='0' width='100%' height='100%'/></svg>");

# Libraries

## CreateJS

Een suite van libraries die het eenvoudiger maken om bep. HTML5-technologieën te gebruiken

* EaselJS: vereenvoudigt het werken met canvas
* TweenJS: vereenvoudigt het tweenen en animeren van HTML5 en JavaScript properties
* SoundJS: vereenvoudigt HTML5 audio
* PreloadJ: vereenvoudigt het beheren en coordineren van het preloaden van assets

## Paper.js

Een library voor vector graphics scripting dat werkt met HTML5 Canvas.

## Box2D

Een physics engine voor JavaScript

## pixi.js

Een 2D webGL renderer met HTML5 Canvas fallback.

