UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Anže Rozman

Novosti HTML5 in grafični vmestnik canvas

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Saša Divjak

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01672/2010
Datum: 05.04.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ANŽE ROZMAN**

Naslov: **NOVOSTI HTML5 IN GRAFIČNI VMESNIK CANVAS**
NEW ELEMENTS IN HTML5 AND CANVAS 2D API

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Predstavite bistvene novosti označevalnega jezika HTML5 in jih preizkusite s programski primeri.
Podrobno predstavite nov HTML element canvas, ki je je ena izmed najbolj zanimivih in uporabnih novosti HTML5. Podajte primere manipulacije slikovnih pik video vsebine. Kot primer uporabe naj bo animirana interaktivna panoramska karta Bohinja. Vmesnik canvas 2D primerjajte tudi z ostalimi tehnologijami za prikaz dvodimenzionalne grafike kot je SVG. Opišite še tehnologijo WebGL za prikaz trodimenzionalne grafike znotraj brskalnika, ki prav tako sloni na HTML5 elementu canvas.

Mentor:

prof. dr. Saša Divjak

Dekan:

prof. dr. Franc Solina

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01672/2010
Datum: 05.04.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ANŽE ROZMAN**

Naslov: **NOVOSTI HTML5 IN GRAFIČNI VMESNIK CANVAS**
NEW ELEMENTS IN HTML5 AND CANVAS 2D API

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Predstavite bistvene novosti označevalnega jezika HTML5 in jih preizkusite s programski primeri.
Podrobno predstavite nov HTML element canvas, ki je je ena izmed najbolj zanimivih in uporabnih novosti HTML5. Podajte primere manipulacije slikovnih pik video vsebine. Kot primer uporabe naj bo animirana interaktivna panoramska karta Bohinja. Vmesnik canvas 2D primerjajte tudi z ostalimi tehnologijami za prikaz dvodimenzionalne grafike kot je SVG. Opišite še tehnologijo WebGL za prikaz trodimenzionalne grafike znotraj brskalnika, ki prav tako sloni na HTML5 elementu canvas.

Mentor:

prof. dr. Saša Divjak

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Franc Solina

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Anže Rozman,

z vpisno številko 63030047,

sem avtor diplomskega dela z naslovom:

Novosti HTML5 in grafični vmestnik canvas.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Saše Divjaka;
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 28.9.2010 Podpis avtorja: _____

Za pomoč in vodenju pri izdelavi diplomske naloge se iskreno zahvaljujem mentorju prof. dr. Saši Divjaku.
Hvala Turizmu Bohinj in Turističnemu društvu Bohinj za panoramsko karto Bohinja in grafično ikono.
Maruši, za vso moralno podporo in spodbujanju pri izdelovanju diplomskega dela.
Posebna zahvala pa gre staršema, ki sta mi omogočila študij in me vseskozi podpirala.

Kazalo

Seznam uporabljenih kratic in simbolov

HTML	HyperText Markup Language (jezik za označevanje nadbesedila)
DOM	Document Object Model (objektni model dokumenta)
W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
RIA	Rich Internet Application
CSS	Cascading Style Sheets (prekrivni slogi)
API	Application programmable interface (programski aplikacijski vmestnik)
WYSIWYG	What You See Is What You Get
SVG	Scalable Vector Grafic

Povzetek

Nova verzija HTML predstavlja veliko novih elementov in nam olajšuje uporabo in izdelovanje modernih spletnih strani. Nekateri od njih so semantične zamenjave starih ali dodajajo čisto novo funkcionalnost. Čeprav HTML5 še ni postal standard, že večina brskalnikov vsaj delno podpira njegove nove funkcionalnosti.

V prvem delu diplomskega dela sem predstavil bistvene novosti označevalnega jezika HTML. Vse HTML novosti sem preizkusil tudi v praksi z programski rešitvami, ki jih lahko spletni razvijalci uporabijo ali prilagodijo za ponovno uporabo v svojih rešitvah.

V nadaljevanju pa sem podrobno predstavil nov HTML element canvas. Element canvas je eden izmed najbolj zanimivih in uporabnih novosti HTML5. Predstavljamo si ga lahko kot pravokotno resolucijsko odvisno bitno platno, na katerem s pomočjo skriptnega jezika JavaScript rišemo in prikazujemo dvodimenzionalno grafiko. Poleg tega pa canvas omogoča tudi vključitev video in avdio vsebine ter izdelovanja animacij. Podal sem primere manipulacije slikovnih pik video vsebine. Glavni izdelek pa je animirana interaktivna panoramska karta Bohinja. Z animirano ikono se s pomočjo tipk ali miške sprehajamo po karti in glede na lokacijo prikazujemo besedilo, slikovno gradivo, predvajamo audio in video vsebino.Vmestnik canvas 2D sem primerjal tudi z ostalimi tehnologijami za prikaz dvodimenzionalne grafike kot je SVG.

Nazadnje pa sem opisal tehnologijo WebGL za prikaz trodimenzionalne grafike znotraj brskalnika, ki prav tako sloni na HTML5 elementu canvas. Opisal pa sem tudi druge trenutne 3D tehnologije za prikaz grafike ter jih primerjal med seboj.

Kjučne besede:

HTML5, canvas, animacija,WebGL

Abstract

The new version of the HTML standard introduces a number of new elements and attributes that reflect typical usage on modern websites and help us code our websites. Some of them are semantic replacements for common uses of generic block and inline elements. Other elements provide new functionality through a standardized interface. Although HTML5 is currently under development, most main browsers at least partially support its new functionality.

In the first part of my diploma thesis I present the main innovations of HyperText Markup Language. I have tested all of them in practice as software solutions that Web developers can use or adapt for reuse in their solutions.

In the main part I introduce a new HTML canvas element in detail. Canvas element is one of the most interesting and useful innovations of HTML5. HTML 5 defines the canvas element as a resolution-dependent bitmap canvas which can be used for rendering graphs, game graphics, or other visual images on the fly. A canvas is a rectangle on your page where you can use JavaScript to draw anything you want. Besides, this canvas allows the integration of video and audio content and can make animations. I have included examples of pixel manipulation of video content. The main product is an animated interactive panoramic map of Bohinj. I compared canvas 2D API with other computer 2D graphics technologies such as SVG.

In the end I describe web computer 3D graphics technology WebGL, which is also based on the HTML5 canvas element. I also describe other current 3D web graphics technologies and compare them with each other.

Keywords:
HTML5, canvas, animation,WebGL

1 Uvod

1.1 Problemsko področje

HTML5 je nova verzija označevalnega jezika HTML. Sedanja verzija HTML je izšla leta 1999. Svetovni splet se je od takrat veliko spremenil. HTML 5 je še v razvoju in je predlagan kot naslednji standard za HTML 4.01, XHTML 1.0 in DOM Level 2 HTML. Ideja o HTML5 je nastala kot odločitev o sodelovanju med W3C, ki je razvijal verzijo XHTML 2.0, in WHATWG, ki je deloval na spletnih formah in aplikacijah. [1,2] Njihovi cilji in smernice so:

- zmanjšati potrebo po zaščitenih bogatih spletnih aplikacijah RIA kot so Adobe Flash, Microsoft Silverlight in Sun JavaFX, ki jih namestimo v brskalnik z vtičniki.
- Sklenili so, da bodo nove vsebine slonele na HTML, CSS, DOM in JavaScript.
- HTML5 naj bo neodvisen od naprave, operacijskega sistema.
- Boljša obravnava napak.
- Proces razvoja standarda naj bo viden javno.

Specifikacija jezika se je začela razvijati junija 2004 in čeprav HTML5 še ni postal standard, že večina brskalnikov spletnih brskalnikov, predvsem Safari, Chrome, Firefox in Opera vsaj delno podpira njegove nove funkcionalnosti. Novi Internet Explorer 9 pa bo prav tako že podpiral nekaj novosti HTML 5. [1,3]

HTML 5 predstavlja veliko novih elementov in atributov in olajšuje uporabo in izdelovanje modernih spletnih strani. Nekateri od njih so semantične zamenjave starih ali dodajajo čisto novo funkcionalnost. Najbolj zanimive nove vsebine HTML5 so:

- element HTML canvas kot platno za risanje in prikazovanje 2D in 3D vsebin
- elementa HTML video in audio za medijsko predvajanje
- Boljša podpora za delovanje spletnih vsebin v lokalnem brezpovezavnem načinu
- Nove značke za predstavitev posameznih delov spletnih strani kot so article, footer, header, nav, section.
- Nove spletne kontrole in forme za koledar, čas, datum, email, url, search.

V prvi polovici mojega diplomskega dela vam predstavljam novosti označevalnega jezika HTML. V nadaljevanju pa sem se osredotočil predvsem na nov element HTML 5 canvas, na katerem s pomočjo skriptnega jezika JavaScript rišemo in prikazujemo dvodimenzionalno grafiko. Poleg tega pa canvas omogoča tudi vključitev video in avdio vsebine ter izdelovanja animacij. Vmestnik canvas 2D sem primerjal tudi z ostalimi tehnologijami za prikaz dvodimenzionalne grafike kot je SVG.

Nazadnje pa sem predstavil tehnologijo WebGL za prikaz tridimenzionalne grafike v spletnih brskalnikih na elementu HTML canvas. Opisal pa sem tudi druge trenutne 3D tehnologije za prikaz grafike ter jih primerjal med seboj. Nazadnje pa bom poskušal odgovoriti na vprašanje, če je 3D grafika že pripravljena za splet.

1.2 Cilji

Temeljni cilj diplomskega dela je predstaviti bistvene novosti jezika HTML. Vse HTML novosti sem preizkusil tudi z programskimi rešitvami. V drugem delu pa je moj cilj predstaviti novo značko HTML canvas za prikaz dvodimenzionalne grafike in razvijanje animacij. Izdelal sem animirano interaktivno panoramsko karto Bohinja. Tako moje diplomsko delo ni samo predstavitev novosti HTML v teoriji, ampak ponuja številne primere iz prakse, ki jih lahko spletni razvijalci uporabijo ali prilagodijo za ponovno uporabo v svojih rešitvah. Nazadnje sem predstavil novo tehnologijo WebGL za prikaz trodimenzionalne grafike, ki prav tako sloni na elementu HTML canvas.

2 Novosti HTML 5

2.1 Znački video in audio

Standard za prikazovanje video ali audio vsebine na spletnih straneh še ni obstajal. Vedno je bilo potrebno namestiti določen vtičnik, ki je omogočal predvajanje tovstnih vsebin (npr. Flash vtičnik). HTML5 pa je opisal standard za vključitev video in audio vsebin v spletne brskalnike z video oziroma audio značko.

2.1.1 Značka video

Star način za za vključitev video vsebine na spletno stran je potreboval veliko kode:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
width="425" height="344"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/sw
flash.cab#version=6,0,40,0">
      <param name="allowFullScreen" value="true" />
      <param name="allowscriptaccess" value="always" />
      <param name="src
      value="http://www.youtube.com/v/oHg5SJYRHA0&hl=en&fs=1&" />
      <param name="allowfullscreen" value="true" />
      <embed type="application/x-shockwave-flash" width="425"
      height="344"
      src="http://www.youtube.com/v/oHg5SJYRHA0&hl=en&fs=1&"
      allowscriptaccess="always" allowfullscreen="true">
</embed>
</object>
```

Novejši način potrebuje bistveno manj kode:

```
<video width="320" height="240" controls preload="none"
poster="videoframe.jpg">
      <source src="demo-video.mp4" type="video/mp4" />
      <source src="demo-video.ogv" type="video/ogg" />
</video>
```

Slika 1: Grafični vmestnik za prikaz video značke v brskalniku Mozilla Firefox

Če nam kontrolni elementi, ki nam jih nudijo brskalniki, niso vizualno všeč, lahko prek vmestnika DOM API izdelamo lastne kontrole:

```
<video width="320" height="240" preload="none"
poster="videoframe.jpg">
      <source src="demo-video.mp4" type="video/mp4" />
      <source src="demo-video.ogv" type="video/ogg" />
</video>

<script>
      var video = document.getElementsByTagName('video')[0];
      function toggleMute() {
      video.muted = !video.muted;
      }
</script>
<p>
      <a href="JavaScript:video.play();">Play</a> |
      <a href="JavaScript:video.pause();">Pause</a> |
      <a href="JavaScript:toggleMute();">Sound On/Off</a>
</p>
```

Play | Pause | Sound On/Off

Slika 2: Video značka z lastnimi kontrolami

Trenutno HTML5 podpira dva video formata za prikaz v znački video [3]:

	Internet Explorer	Firefox 3.5	Opera 10.5	Chrome 3.0	Safari 3.0
Ogg		x	x	x	
MPEG 4				x	x

Tabela 1: Glavni spletni brskalniki in njihova podpora video formatom v novi HTML 5 znački video.

Poleg tega pa lahko uporabljamo vmestnik DOM API še za mnoge druge stvari. Lahko uporabimo značko HTML canvas, na katerem prikažemo video. V tem primeru lahko beremo oziroma zajamemo posamezne podatke iz slikovnih pik ali videosličic in z njimi upravljamo. Lahko naredimo zelo zanimive stvari, ki vam jih bom predstavil v drugem delu.

2.1.2 Značka audio

Skriptna koda za vključitev zvočne vsebine na spletno stran v obliki novega elementa HTML 5 <audio>:

```
<audio controls>
      <source src="demo-audio.ogg" />
      <source src="demo-audio.mp3" />
</audio>
```

Slika 3: Grafični vmestnik za prikaz audio značke v brskalniku Mozilla Firefox.

Trenutno so v HTML 5 podprti trije formati za značko audio [3]:

	Internet Explorer	Firefox 3.5	Opera 10.5	Chrome 3.0	Safari 3.0
Ogg Vorbis		x	x	x	
Mp3				x	x
Wav		x	x		x

Tabela 2: Glavni spletni brskalniki in njihova podpora zvočnim formatom v novi HTML 5 znački audio.

Tudi pri tem elementu HTML lahko uporabljamo metode vmestnika API in ročno krmilimo zvočno vsebino:

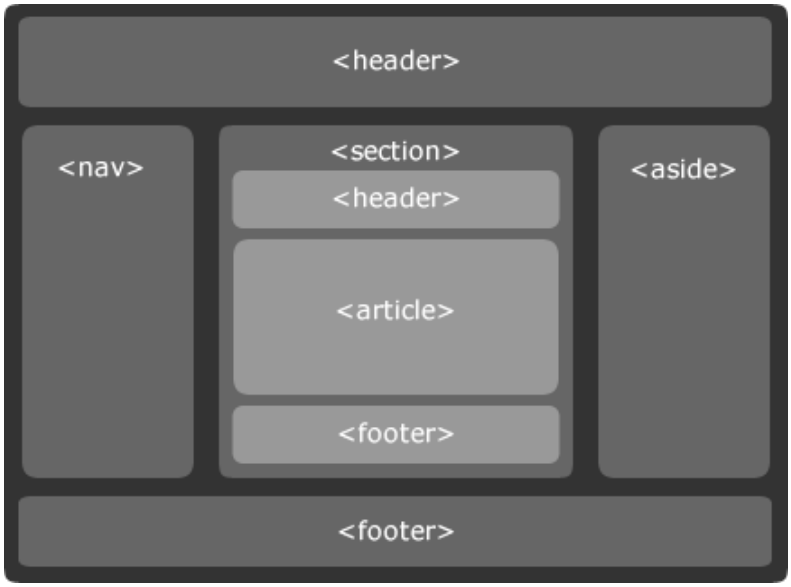
- ***play()*** - predvajaj avdio
- ***pause()*** - premor
- ***canPlayType()*** - komunicira z brskalnikom, če je tip MIME možno vzpostaviti za predvajanje
- ***buffered()*** - atribut, ki časovno določi začetek in konec trenutno naložene datoteke v medpomnilniku

2.2 Semantično označevanje

V HTML 5 je definiranih kar nekaj novih elementov za bolj specifično označevanje vsebine, ki bodo služili zgolj kot smiselna alternativa oznake . Nove oznake HTML so:

- **<*article*>** označuje vsebinsko zaključeno celoto. Vsebina je lahko novica, članek, besedilo iz bloga ali drugega spletnega mesta, foruma ali druge vsebine iz zunanjega vira.
- **<*section*>** značka določa segment, blok, razdelek v nekem dokumentu kot je naprimer poglavje, glava, noga ali kak drug segment dokumenta, ki ima svoj naslov in tematsko združuje vsebino.
- **<*aside*>** značka določa vsebino, ki se povezuje z ostalo vsebino.
- **<*hgroup*>** je namenjen združevanju naslovov in podnaslovov.
- **<*header*>** označuje glavo ali uvod v dokument ali razdelek. Uporabimo ga, kadar hočemo določiti naprimer naslov celotne spletne strani, naslov <*article*> vsebine ali naslov <*section*> vsebine.

- <***footer***> označuje nogo določenega razdelka ali dokumenta. Tipično vsebuje ime avtorja, kontaktne informacije in datum dokumenta. Ponavadi ga uporabljamo na dnu spletne strani.
- <***nav***> določa odsek v dokumentu, ki je namenjen za navigacijo do ostalih podstrani ali drugih spletnih strani.
- <***time***> oznaka določa čas ali datum ali oboje v Gregorijanskem koledarju, opcijsko tudi z zamikom cone.
- <***mark***> določa označeno besedilo. Z njim poudarimo, označimo besedilo.
- <***figure***> uporabimo, kadar hočemo združiti več skupnih HTML elementov. Vsebina znotraj figure značke je samostojna. Uporabimo jo kadar hočemo razložiti, opisati del dokumenta, oziroma da združimo grafiko in video.
- <***figcaption***> besedilo, ki opisuje vsebino figure značke. Je znotraj figure značke.



Slika 4: Prikaz nove strukture strani z semantičenim označevanjem.

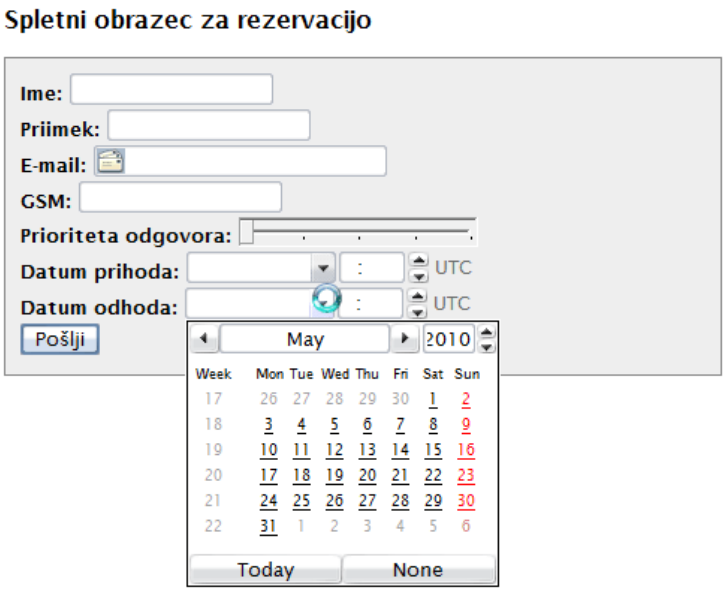
Vsi omenjeni novi elementi so že vključeni v najnovejših verzijah modernih brskalnikov kot so Firefox 3+, Safari 3.1+, Chrome 2+ in Opera 9.6+. Izjema je Internet Explorer, ki naj bi v novi različici vse te elemente tudi vključil.

2.3 Razširitev spletnih obrazcev

HTML 5 nam ponuja kar nekaj novih spletnih elementov za vnos različnih tipov podatkov v spletne obrazce. Do sedaj je bilo potrebno podatke, ki smo jih vnesli v spletni obrazec (naprimer obrazec za registracijo ali enostavni kontaktni obrazec) preverjati, če smo podatke vpisali pravilno, ali če so veljavni in obvezni. To je bilo mogoče storiti le programsko s pomočjo JavaScript, PHP ali drugega programskega jezika za spletno programiranje. Z novo specifikacijo v HTML 5 pa bo mogoče veliko lažje obdelovati podatke, ki bodo poslani preko obrazca. Samo enostavni atribut HTML5 bo potreben za preverjanje pravilnosti in veljavnosti za najpogostejše vnešene podatke. Zaenkrat jih je večino možno uporabljati v spletnem brskalniku Opera [4,5,6].

Novi vhodni tipi podatkov so:

- ***search***
- ***tel***
- ***url***
- ***email***
- ***datetime, date, time, month, week, day, datetime-local***
- ***number***
- ***range***
- ***color***



Slika 5: Primer spletnega obrazca z novimi kontrolami HTML5.

Novi atributi so:

- ***required*** - atirbut je tipa boolean, ki ga uporabimo, če je vnosno polje potrebno obvezno izpolniti.
- ***autofocus*** – atribut se na naša na vnosno polje. Takoj, ko se stran naloži, se samodejno premakne kazalec v prvo vnosno polje s tem atributom.
- ***placeholder*** – s tem atributom določimo privzeto vrednost vnosnega polja.
- ***pattern*** – s tem atributom določimo regularen izraz, ki je dovoljen za vnos v polje.
- ***list*** – kazalec na datalist, ki vsebuje seznam vrednosti. Je zelo uporaben atribut, saj nam ni potrebno uporabiti knjižnice JavaScript za seznam, temveč lahko s pomočjo rokovanja z dogodki (onchange) in z nekaj kode AJAX pridobimo, kaj je uporabnik vtipkal v polje.

```
<input name="form_url" id="form_url" type="url"
list="url_list">
<datalist id="url_list">
        <option value="http://www.google.com" label="Google">
        <option value="http://www.nettuts.com" label="NetTuts">
</datalist>
```

- ***autocomplete*** atribut uporabimo, kadar hočemo zaščititi naše vnose v polje. Brskalniku namreč onemogočimo, da si zapomni vrednosti, ki smo jih že enkrat vnesli v to vnosno polje.

Novi elementi HTML5 so:

- **<*keygen*>** je oznaka za zgeneriran ključ.
- **<*datalist*>** označuje seznam možnosti za izbiranje.
- **<*output*>** je značka za izpis podatkov, ki nam jih na primer vrne skripta ali preprost matematični izraz:

```
<input name="number_1" type="number">
+ <input name="number_2" type="number">
=  <output onforminput="value = number_1.valueAsNumber + number
_2.valueAsNumber"></output>
```

- **<*meter*>** in **<*progress*>** sta zelo podobna elementa HTML5 z eno razliko. Značko <*progress*> uporabimo kadar imamo namen meriti potek neke naloge. Naprimer, če rešujemo anketo ali raziskavo na večih straneh s pomočjo <*progress*> značke prikazujemo koliko strani oziroma vprašanj imamo še do konca ali naprimer kolikšen del datoteke je že prenešen iz nekega spletnega mesta. Seveda pe je treba za dinamični prikaz uporabiti JavaScript.

```
<p>Downloaded:
  <progress value="1534602" max="4603807">33%</progress>
</p>
```

Značko <*meter*> pa uporabimo lahko za numerično vrednost v določenem območju. Z to značko prikazujemo naprimer kolikšna kapaciteta diska je še na voljo, kolikšen del volivcev je glasovalo za določeno stranko itd.

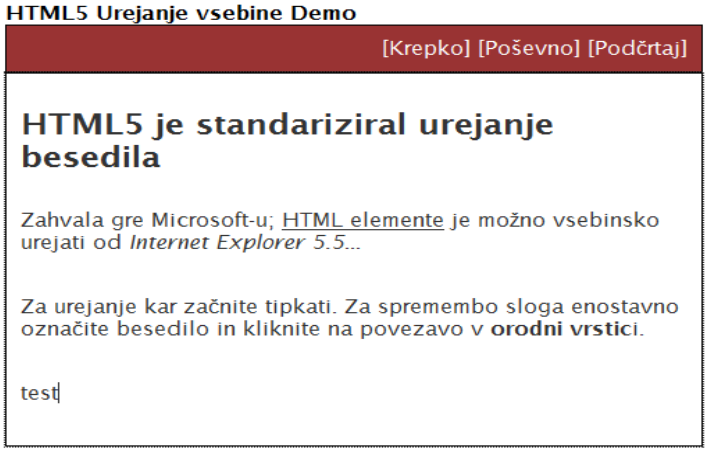
```
<p>An entry level programmer in Silicon Valley
can expect to start around <meter>$90,000</meter> per year.
</p>
```

```
<p>Rezultat vašega kolokvija je
<meter value="88.7" min="0" max="100" low="65" high="96"
optimum="100">ocena 9</meter>.
</p>
```

2.4 Urejanje vsebine

Za urejanje besedila na spletni strani je potrebno dodati v znački za prikaz besedila atribut ***contenteditable*** ter ga postaviti na *true*. Ta atribut je del HTML 5 specifikacije in je podprt s strani večine glavnih spletnih brskalnikov (Internet Explorer 5.5+, Firefox 3+, Safari 3.1+, Chrome 2+, in Opera 9.6+).

Postavitev atributa *contenteditable="true"* omogoča uporabnikom urejanje, brisanje in vstavljanje vsebine. Toda to ne bo samodejno prikazalo gradnikov za urejanje in shranjevanje vsebine kot jih na primer vidimo v urejevalnikih WYSIWYG. Potrebno je uporabiti še kodo JavaScript za dodajanje vseh teh gradnikov.



Slika 6: Primer dela spletne strani za urejanje besedila.

Primer kode za zgornji enostavni urejevalnik vsebine:

```
<script language="javascript">
      function editStyle(cmd) {
      document.execCommand(cmd, null, null);
      }
</script>

<section>
      <header>
            <h1>HTML5 Urejanje vsebine Demo</h1>
      </header>
      <div id="editingcontrols" autofocus>
            <a href="#" onClick="editStyle('bold');">[Krepko]</a>
            <a href="#" onClick="editStyle('italic');">[Poševno]</a>
            <a href="#"
onClick="editStyle('underline');">[Podčrtaj]</a>
      </div>
      <div id="editor" contenteditable="true">
            <h2>HTML5 je standariziral urejanje besedila</h2>
      <p>Zahvala gre Microsoft-u; HTML elemente je možno vsebinsko
urejati od Internet Explorer 5.5...</p>
      <p>Za urejanje kar začnite tipkati. Za spremembo sloga
enostavno označite besedilo in kliknite na povezavo v orodni
vrstici.</p>
      </div>
</section>
```

2.5 Primi in spusti (drag and drop)

Naslednja zanimiva vsebina je vključitev vmestnika API *drag and drop*. Vlečenje oziroma *»drag and drop«* v računalniškem grafičnem smislu je dejanje, ki ga izvedemo z dogodkom *mousedown* (klik na nek element, objekt), nato mu sledi serija dogodkov *mousemove* (premikanje z miško po zaslonu) ter dogodek *drop* (spustimo miško) s katerim spustimo objekt, element na neko površino, ki to omogoča.

Implementacija vmestnika *»drag and drop«* brez vtičnika je bila vse do danes odvisna le od kompleksne JavaScript kode ali knjižnice JavaScript kot na primer *script.aculo.us*. Novost HTML 5 *»drag and drop«* bo spremenila način spletnega programiranja, oziroma bo

imela vpliv na način izdelave uporabniškega vmestnika. Trenutno ima brskalnik Firefox 3.5+ najboljšo podporo za uporabo te novosti v HTML5, ostali brskalniki jo podpirajo le delno [7].

Značilnosti vmestnika *»drag and drop«* API*:*

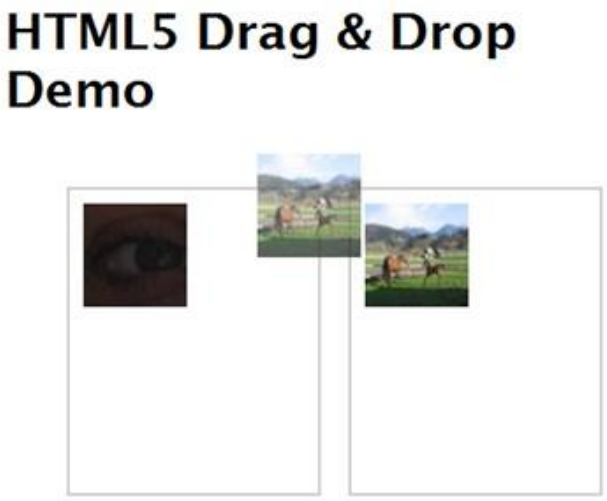
- ***draggable*** – nov atribut html5 elementa s katerim omogočimo vlečenje elementa. HTML element slika ali hiperpovezava ima privzeto vlečenje.
- Naslednji dogodki sestavljajo delovanje *»drag and drop«*:
  - ***dragstart*** se sproži, ko pričenemo vleči objekt
  - ***drag*** se sproži vsakič, ko z miško vlečemo objekt
  - ***dragenter*** se sproži, ko je objekt, ki ga vlečemo, prvič povlečen v ciljni objekt
  - ***dragover*** se sproži vsakič, ko je objekt, ki ga vlečemo, premaknjen v ciljni objekt
  - ***dragleave*** se sproži vedno, ko je vlečen objekt povlečen izven ciljnega objekta
  - ***drop*** se sproži vedno, ko je vlečen objekt spuščen v ciljni objekt
  - ***dragend*** se sproži, ko spustimo miško in s tem objekt, ki smo ga vlekli

  Potrebno je napisati funkcije za upravljanje dogodkov (*ang. event handlers*) s katerimi poskrbimo za to kar potrebujemo v trenutku, ko bodo ti dogodki sproženi.

- Dogodki drag and drop nam omogočajo tudi prenos podatkov preko objekta, ki ga vlečemo. To dosežemo z metodama ***getData*** in ***setData*** preko lastnosti dogodka ***dataTransfer***:
  - *setData(dataType, data)* – postavi podatek, ki ga lahko delimo z ciljnim objektom
    - *dataType*: Tip podatka, ki ga lahko nastavimo. Do zdaj sta edina tipa podatkov, ki ga lahko prenašamo preko brskalnika, text in url
    - *data*: podatek ki ga nastavimo
  - *getData(dataType)* – s to metodo pridobimo podatek, ki je bil predtem postavljen z metodo *dataTransfer.setData().* Lahko kličemo tudi podatek, ki je bil postavljen z metodo *dataTransfer.setData()* iz druge spletne strani, iz drugega okna istega brskalnika ali okna drugega brskalnika na istem računalniku. Prav tako lahko pridobimo podateke z namiznih aplikacij (npr. Windows WordPad)
    - *dataType*: Tip podatka, ki ga lahko nastavimo. Do zdaj sta edina tipa podatkov, ki jih lahko prenašamo preko brskalnika, text in url

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>HTML5 Drag & Drop Demo</title>
    <link rel="stylesheet" href="html5reset.css" type="text/css" />
    <link rel="stylesheet" href="html5simple.css" type="text/css" />
    <!--[if lt IE 9]>
        <script src="html5.js"></script>
    <![endif]-->

    <script>
    function DragHandler(target, e) {
        e.dataTransfer.setData('Text', target.id);
    }

    function DropHandler(target, e) {
```

```
        var id = e.dataTransfer.getData('Text');
        target.appendChild(document.getElementById(id));
        e.preventDefault();
    }
    </script>
</head>
<body>
    <header>
        <h1>HTML5 Drag & Drop Demo</h1>
    </header>
<div id="dndcontainer">
    <div ondrop="DropHandler(this, event)" ondragenter="return
false" ondragover="return false" class="dndbox">
        <img src="item-1.png" id="club"
ondragstart="DragHandler(this, event)" width="75" height="75"
border="0" alt="" />
        <img src="item-2.png" id="heart"
ondragstart="DragHandler(this, event)" width="75" height="75"
border="0" alt="" />
        <img src="item-3.png" id="spade"
ondragstart="DragHandler(this, event)" width="75" height="75"
border="0" alt="" />
        <img src="item-4.png" id="diamond"
ondragstart="DragHandler(this, event)" width="75" height="75"
border="0" alt="" />
    </div>

    <div ondrop="DropHandler(this, event)" ondragenter="return
false" ondragover="return false" class="dndbox"></div>

    <div id="demonotes">Drag and drop images from one container to
another. Works in all major browsers except Opera.</div>

</div>

</body>
</html>
```



Slika 7: Primer grafičnega vmestnika za drag and drop.

2.6 WEB STORAGE

Web storage je aplikacijski programski vmestnik za obstojno shranjevanje podatkov v spletnih brskalnikih. Sprva je bil del specifikacije HTML5, sedaj pa je samostojna standarizirana specifikacija W3C. Podpirajo jo naslednji brskalniki: IE8+, Firefox 3.5+, Safari 4.0+, Chrome 2.0+ in Opera 10.5+. Specifikacija Web Storage prinaša dva nova atributa za kratkoročno in dolgoročno ohranitev podatkov, podobno kot seje HTTP in piškotki. Atributa se sta naslednja:

- ***sessionStorage*:** atribut uporabimo za shranjevanje podatka, kadar želimo, da je shranjen podatek prisoten toliko časa, kot je odprta spletna stran v oknu ali zavihku brskalnika
- ***localStorage*** atribut uporabimo, kadar želimo shraniti podatek za daljše obdobje. Podatek ostane v brskalnikovem pomnilniku tudi, če ponovno zaženemo brskalnik ali računalnik.

Tako atribut *localStorage* kot tudi atribut *sessionStorage* uporabljata spodnji vmestnik API:

```
window.localStorage and window.sessionStorage {
  long length; // Number of items stored
  string key(long index); // Name of the key at index
  string getItem(string key); // Get value of the key
  void setItem(string key, string data); // Add a new key with value
data
  void removeItem(string key); // Remove the item key
  void clear(); // Clear the storage
};
```

Spodnji primer programske kode nam ponazarja, kako shranimo in kako beremo niz znakov (string):

```
// save the string
function saveStatusLocally(txt) {
  window.localStorage.setItem("status", txt);
}

// read the string
function readStatus() {
   return window.localStorage.getItem("status");
}
```

Lastnosti *localStorage* in *sessionStorage* so omejene glede na izvor strani HTML *(scheme + hostname + non-standard port).*To pomeni, da ima *window.localStorage* za spletni naslov *http://rtvslo.si* drugačno instanco kot *window.localStorage* za spletni naslov *http://24.com.*

```
<p>You Have Viewed This Page <b>
<script>
    if (!localStorage.pageCounter)
        localStorage.setItem('pageCounter',0);

localStorage.setItem('pageCounter',parseInt(localStorage.pageCounter
)+1);
    document.write(localStorage.pageCounter);
</script>
```

```
</b> Time(s).</p>

<p><input value="Clear localStorage" type="button"
onClick="localStorage.clear();location.reload(true);" />
<input value="Reload Page" type="button"
onClick="location.reload(true);" /></p>
```

Programska koda za preprost števec strani, ki šteje toliko časa, kot bo stran v brskalniku odprta. Če zamenjamo primerek *localStorage* s *sessionStorage*, se bo števec shranil in naprej štel obiske strani tudi v primeru, če računalnik ali brskalnik ponovno zaženemo.

Na web storage lahko enostavno gledamo kot izboljšavo piškotkov, le da nam ponuja veliko večjo pomnilniško kapaciteto (5MB za posamezno domeno v spletnih brskalnikih Mozilla Firefox, Google Chrome in Opera, 10MB za spletni brskalnik Internet Explorer). Poleg tega pa nam nudi tudi boljši vmestnik za programiranje [8,9].

Web storage je vmestik na odjemalčevi strani (*client-side interface*). Do piškotkov smo lahko dostopali tako na strežniški kot odjemalčevi strani. Web storage pa je namenjen za skriptiranje izključno na odjemalčevi strani. Podatki spletnega skladišča Web storage niso poslani do strežnika za vsak *HTTP request* in strežnik ne more direktno pisati v spletno skladišče (Web storage) [8,9].

2.7 Offline Web Aplications

Večina spletnih brskalnikov shranjuje spletne strani v lokalni predpomnilnik, kar nam omogoča, da jih lahko prebiramo tudi, če imamo prekinjeno internetno povezavo. To deluje predvsem za statične spletne strani. Problem pa se pojavi, če imamo dinamično vsebino, ki deluje v povezavi z podatkovno bazo kot naprimer Gmail, Facebook ali Twitter.

V HTML5 so vključili podporo za spletne aplikacije za delovanje v brezpovezavnem načinu. Brskalnik, ki ima to podporo vključeno (Firefox 3.5+, Chrome 4.0+, Safari 4.0+ in Mobile Safari 3.1+), prenese vse potrebne datoteke za delovanje aplikacije v brezpovezavnem načinu. Sedaj uporabnik lahko uporablja aplikacijo v brezpovezavnem načinu in vse spremembe se ob ponovni povezavi z internetom prenesejo na strežnik.

Za delovanje spletne aplikacije v brezpovezavnem načinu je potrebno določiti posebno datoteko, ki se imenuje ***manifest***. V tej datoteki je zapisano katere datoteke je potrebno prenesti za delovanje v brezpovezavnem načinu.

V HTML 5 datoteki določimo datoteko *manifest* na naslednji način:

```
<!DOCTYPE html>
<html manifest="offline.manifest">
<head>
...
```

Vsaka html stran, ki vsebuje *<html manifest="offline.manifest"*> oznako se samodejno prenese v predpomnilnik za delovanje v brezpovezavnem načinu.

Manifest datoteka zgleda takole:

```
CACHE MANIFEST

# VERSION 10
```

```
CACHE:
offline.html
base.css

FALLBACK:
online.css offline.css
```

Prva vrstica nam pove, da gre za datoteko *manifest*. Pod vrstico CACHE pa določimo katere datoteke je potrebno prenesti v predpomnilnik za delovanje v brezpovezavnem načinu. Nato imamo vrstico FALLBACK in datoteki pod njo. Brskalnik skuša najprej dostopati do datoteke *online.css*. Če je dostop neuspešen oziroma, če ni internetnega dostopa, bo brskalnik odprl naslednjo datoteko v vrstici *offline.css*. [10]

3 HTML5 CANVAS

Ena izmed najbolj zanimivih novosti HTML5 je značka <*canvas*> . Je zelo uporaben element HTML5, ki je zaenkrat še razmerom nerazširjen. Ta element, ki si ga lahko predstavljamo kot pravokotno resolucijsko odvisno bitno platno, omogoča dinamično risanje oziroma prikazovanje grafike na spletni strani. Za to pa je potrebno nekaj programerskega znanja JavaScript.

Canvas podpirajo vsi moderni spletni brskalniki kot so Firefox 3, Safari 3.1, Chrome 2 in Opera 9.6. Izjema je le Internet Explorer, ki naj bi ga tudi vključili v novi verziji. Zaenkrat so za prikaz canvas značke v Internet Explorer pripravili rešitev *ExplorerCanvas*. To je JavaScript datoteka, ki jo vključimo v spletno stran na naslednji način:

```
<head>
<!--[if IE]><script src="excanvas.js"></script><![endif]-->
</head>
```

Canvas so razvili v podjetju Apple, ki so ga zaščitili. Vsem razvijalcem brskalnikov je bil všeč, zato so ga vsi vgradili v svoje brskalnike in kmalu je postal predpisan kot odprtokodna W3C specifikacija. Je način neposrednega prikaza grafike, ki ne potrebuje vtičnikov. Če hočemo na spletni vsebini prikazovati dinamično grafiko, je potrebno dinamičnost zagotoviti preko strežnika, kar je počasno, ali uporabljati vtičnike, kot naprimer Flash ali Javo. Canvas pa je vmestnik za neposredno prikazovanje grafike, je neke vrste navidezna ploskev, na katero rišemo z JavaScriptom.

Canvas lahko uporabimo za prikazovanje grafičnih elementov kot so:

- Enostavni diagrami
- Elegantne uporabniške vmestnike
- Animacije in igre
- Razpredelnice in grafe
- Vgrajene spletne risalne aplikacije
- ...

Orodja ki so nam na voljo:

Risalna orodja:

- Pravokotniki
- Krogi in krožni loki
- Črte in sledi (paths)
- Bezierove in kvadratne krivulje
- Besedilo
- Slike

Efekti:

- Enostavne oblike (funkcije za izris)
- Senčenja
- Linearni in radialni prelivi
- Alfa prosojnost
- Sestavljanja

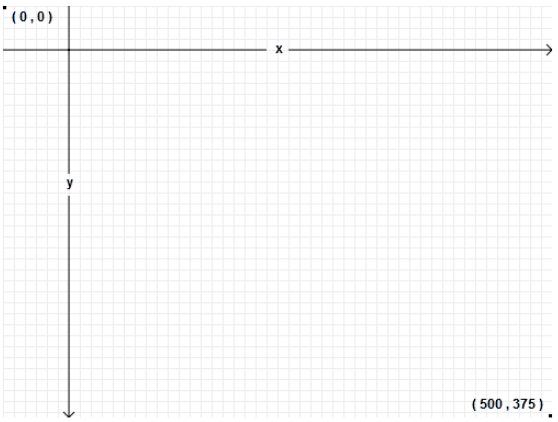
Transformacije:

- Skaliranje
- Rotacije
- Translacije
- Transformiranje matrik

Element canvas je torej grafični vsebnik, ki mu lahko določimo širino ter višino ter ID za objekt, da ga lahko kasneje preko vmestnika DOM API najdemo v kodi JavaScript. Vstavljenje značke canvas na spletno stran je enostavno, kot dodajanje drugih oznak HTML:

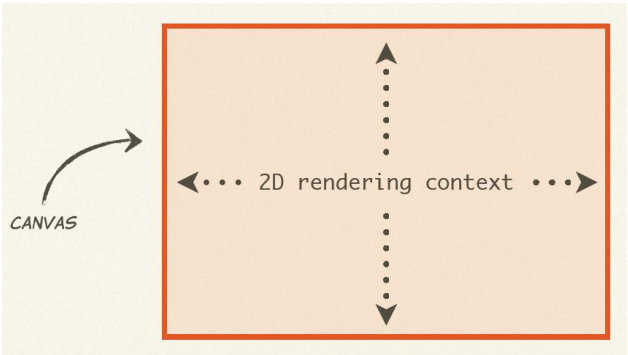
```
<canvas id="example" width="200" height="200">
     To besedilo se prikaže v primeru, da vaš brskalnik ne podpira
     HTML5 Canvas.
</canvas>
```

Realno na canvas gledamo kot dvodimenzionalno koordinatno mrežo. Koordinatno izhodišče *(0,0)* leži na zgornjem levem kotu canvas mreže. Na *x* osi se povečujejo vrednosti v desno smer, na *y* osi pa se povečujejo vrednosti proti dnu canvas mreže, kot je prikazano na spodnji sliki:



Slika 8: Canvas kot dvodimenzionalna koordinatna mreža. Slika se nahaja na http://diveintohtml5.org/canvas.html

Če bi radi kaj prikazali na tem platnu, moramo uporabiti JavaScript vmestnik *2D context API*. Canvas značko najdemo z metodo *getElementById*. Nato pa še določimo referenco na objekt z metodo *getContext('2d')*. Naslednja verzija specifikacije HTML5 pa bo najverjetneje definirala že trodimenzionalni grafični kontekst. [11,12]



Slika 9: Ploskev za prikazovanje dvodimenzionalne grafične vsebine.

Če uporabimo skripto, lahko narišemo na canvas rdeč pravokotnik na naslednji način:

```
var example = document.getElementById('example');
var context = example.getContext('2d');
context.fillStyle = "rgb(255,0,0)";
context.fillRect(30, 30, 50, 50);
```

S pomočjo vmestnika JavaScript (*2D context API*) lahko torej prikazujemo raznovrstne like. Lahko jim dodamo raznovrstne grafične učinke. V naslednjaem poglavju vam bom predstavil osnovne elemente vmestnika:

3.1 Osnovne črte in oblike (Basic lines and strokes)

V zgornjem primeru smo se prepričali, kako enostavno je narisati barvast pravokotnik. Če uporabimo lastnosti *fillStyle* in *strokeStyle* lahko nastavimo barvo za prikazovanje likov ali črt. Privzeta barva je črna. Lahko uporabimo enake barvne vrednosti kot v CSS: *hex codes*, *rgb()*, *rgba()* in celo *hsla()*, če jo brskalnik podpira (Firefox 3 in Opera 10). Z lastnostjo *fillStyle* poleg barve lahko določimo tudi vzorec ali preliv. Lastnost *strokeStyle* je identična lastnosti *fillStyle*, le da je namenjena robovom in črtam.
Z metodo *fillRect(x, y, width, height)* lahko rišemo polnjene pravokotnike. Barvo mu določimo z omenjeno lastnostjo *fillStyle*.
Z metodo *strokeRect(x, y, width, height)* pa lahko rišemo prazne nepobarvane pravokotnike samo z obrobami. Barvo roba določimo z že omenjeno lastnostjo *strokeStyle*.
Površino na canvas pa lahko tudi brišemo v obliki pravokotnika, z metodo *clearRect(x, y, width, height)*.
Vse tri metode imajo enake argumente *(x, y, width, height).* Prva dva določata koordinate na mreži canvas, druga dva pa širino in višino pravokotnika.
Z lastnostjo *lineWidth* pa določimo debelino črte.

Vse to sem upošteval in prikazal na spodnjem primeru:

```
context.fillStyle   = "green";
context.strokeStyle = "gray";
context.lineWidth   = 5;

context.fillRect  (2,2, 150, 50);
context.strokeRect(2,62, 150, 50);
context.clearRect (30, 25,90, 60);
context.strokeRect(30, 25,  90, 60);
```

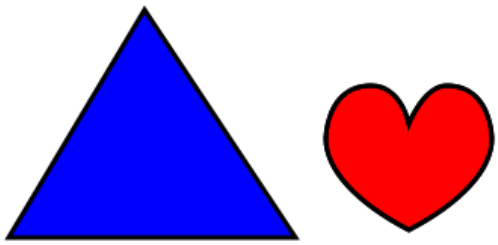
Slika 10: Izris pravokotnih likov na canvas.

3.2 Črte ali sledi (paths)

Canvas nam omogoča risanje raznovrstnih likov s pomočjo črt, potez. Poteka podobno kot rišemo z pisalom. Najprej narišemo obris (*pot*) neke poteze, lika, nato pa vse to poljubno zapolnimo z barvo ali kakšnim drugim vzorcem.
Risanje enostavnega lika je preprosto:

- ***beginPath()*** – s to metodo določimo začetek risanja poti
- ***moveTo(x,y)*** – s to metodo določimo začetno koordinato na canvas znački.
- ***lineTo(x,y)*** – metoda s katero narišemo črto od koordinate, ki smo jo določili z prejšnjo metodo *moveTo(x,y)* (ali *lineTo(x,y)*) do poljubne koordinate, ki jo določimo z metodo *lineTo(x,y).* Omenjeni dve metodi ponavljamo v zaporedju večkrat glede na to kakšen lik imamo namen narisati.
- Vmestnik 2D Contex API poleg metode za risanje ravnih črt nudi tudi metode za risanje raznih krivulj (kvadratne krivulje, bezierove krivulje,..)
- Ko smo končali z risanjem poti lahko uporabimo metodo ***fill*** in ***stroke***, da se nam izrišejo poteze v poljubni barvi, ki jih določimo z lastnostmi *fillStyle, strokeStyle* ter *lineWidth* za debelino črt (poti).
- Na koncu zaključimo naše risanje z metodo ***closePath()***

Primer poljubnega lika:



Slika 11: Izris poljubnih likov na canvas.

```
ctx.beginPath();
ctx.lineWidth = 3;
ctx.strokeStyle = "#000000";
ctx.fillStyle = "blue";
ctx.moveTo(100, 50);
ctx.lineTo(10, 200);
ctx.lineTo(200, 200);
ctx.closePath();
ctx.fill();
ctx.stroke();


//bezierova krivulja
x = 250;
y = 100;
ctx.lineWidth = 3;
ctx.fillStyle = "red";
ctx.beginPath();
ctx.moveTo(x + 25, y + 25);
ctx.bezierCurveTo(x + 25, y + 25, x + 20, y, x, y);
```

```
ctx.bezierCurveTo(x - 30, y, x - 30, y + 35,x - 30,y + 35);
ctx.bezierCurveTo(x - 30, y + 55, x - 10, y + 77, x + 25, y + 95);
ctx.bezierCurveTo(x + 60, y + 77, x + 80, y + 55, x + 80, y + 35);
ctx.bezierCurveTo(x + 80, y + 35, x + 80, y, x + 50, y);
ctx.bezierCurveTo(x + 35, y, x + 25, y + 25, x + 25, y + 25);
ctx.closePath();
ctx.fill();
ctx.stroke();
```

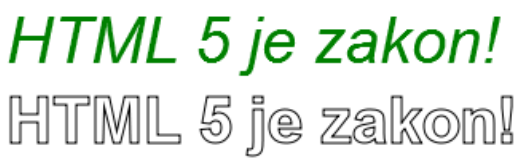
3.3 Vstavljanje besedila

Na značko canvas lahko vstavimo tudi besedilo. Pri pisanju na canvas nimamo modela škatle kot pri tehniki CSS (*padding, margin, floats, word wrapping*). Imamo naslednje atribute za določitev lastnosti in poravnave pisave:

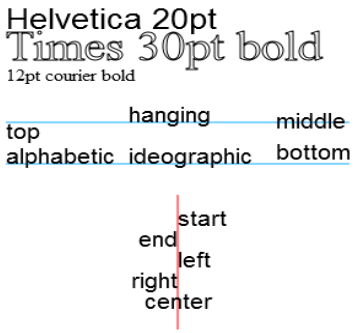
- ***font*** – tako kot pri pravilih CSS za pisavo tudi na canvas lahko določimo slog pisave, različico pisave, debelino in velikost pisave, višino vrstice, družino pisav.
- ***textAlign*** – določa horizontalno poravnavo besedila. Podobno kot pri pravilih CSS *text-align*. Vrednosti za poravnavo besedila so: *start* (začetek), *end* (konec), *left* (levo), *right* (desno) in *center* (sredina).
- ***textBaseline*** – določa vertikalno poravnavo besedila. Določimo kje vertikalno na canvas mreži je lokacija besedila glede na začetni točki canvas ploskve. Vrednosti so: *top, hanging, middle, alphabetic, ideographic, ali bottom.*

Za izrisanje besedila imamo dve metodi: ***fillText*** in ***strokeText***. Prva nam izriše besedilo v obliki, ki jo določimo z lastnostjo *fillStyle*, druga pa besedilo z obrobljenimi črkami, ki jo določimo z lastnostjo *stokeStyle*. Obe metodi imata po tri argumente: besedilo, ki ga želimo prikazati ter koordinati (x,y), ki določata, kje na platnu naj se besedilo izriše.

```
var cnv = document.getElementById('c2');
var context= cnv.getContext('2d');
context.fillStyle    = 'green';
context.font         = 'italic 40px sans-serif';
context.textBaseline = 'top';
context.fillText  ('HTML 5 je zakon!', 0, 0);
context.font         = 'bold 40px sans-serif';
context.strokeText('HTML 5 je zakon!', 0, 50);
```



Slika 12: Izris besedila na canvas.



Slika 13: Primer poravnave besedila na canvas. Slika se nahaja na naslovu http://blog.burlock.org/html5/108-text

3.4 Senčenje

Canvas nam omogoča tudi senčenje. Vmestnik Shadows API nam ponuja štiri lastnosti:

- ***shadowColor*** – določimo barvo senčenja
- ***shadowBlur*** – določimo količino slikovnih pik za senčenje. Večja kot je vrednost, močnejše je senčenje
- ***shadowOffsetX*** in ***shadowOffsetY*** – določamo x in y oddaljenost senčenja, v slikovnih pikah

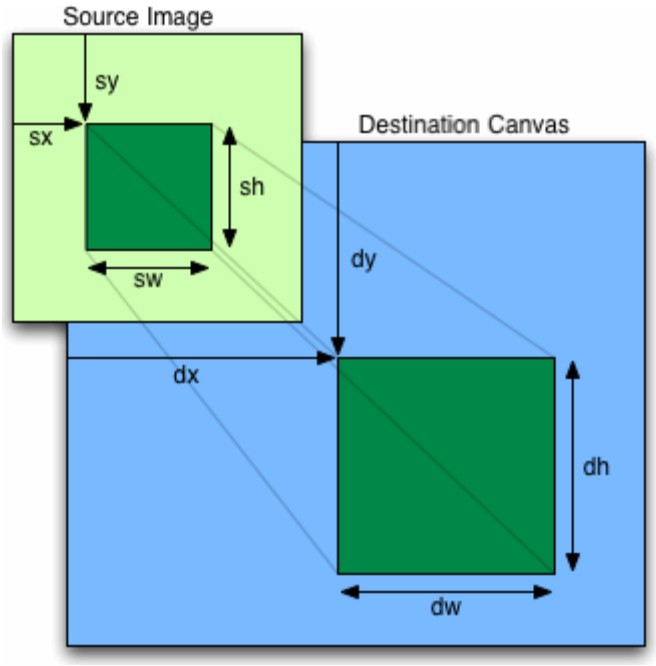
```
context.shadowOffsetX = 20;
context.shadowOffsetY = 20;
context.shadowBlur    = 60;
context.shadowColor   = 'rgba(50, 50, 10, 0.5)';
context.fillStyle     = 'green';
context.fillRect(20, 20, 150, 100);
```

Slika 14: Senčenje na canvas.

3.5 Izrisovanje slik na canvas

Na canvas lahko izrisujemo tudi slike. To nam omogoča metoda *drawImage*, ki ima lahko tri, pet ali devet argumentov:

- ***drawImage(image, dx, dy)*** nam nariše sliko (image) na canvas. Koordinati *dx*, *dy* določata zgornji levi kot slike na canvas
- ***drawImage(image, dx, dy, dw, dh)*** metoda je podobna kot prejšnja, le da z argumentoma *dw* in *dh* določimo velikost izrisane slike (širina, višina).
- ***drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh***) metodo uporabimo, kadar želimo zajeti oziroma odrezati del slike (image) v obliki pravokotnika, ki ga določimo z koordinatami (*sx, sy, sw, sh*). Nato pa jo metoda na platnu prikaže in poljubno raztegne, tako kot smo to naredili z prejšno metodo.



Slika 15: Primer izrisovanja slik z metodo drawImage. Slika je iz naslova http://dev.w3.org/html5/canvas-api/canvas-2d-api.html

3.6 Manipulacija s slikovnimi pikami

V vmestniku 2D Context API imamo tri metode, s katerimi lahko zajemamo, ustvarjamo in izrisujemo slikovne pike: ***getImageData***, ***createImageData***, ***putImageData***. Neobdelane slikovne pike so zajete v objektu tipa ImageData. Vsak objekt ima tri lastnosti: ***širino (width), višino (height)*** in ***podatki (data)***. Lastnost *data* je tipa *CanvasPixelArray*, ki vsebuje natanko *width*height*4* število elementov.

Zmnožek *širina*višina* (v slikovnih pikah) nam pove, iz koliko slikovnih pik je slika sestavljena. Za vsako slikovno piko lahko določimo štiri podatke: rdeča (red), zelena (green), modra (blue) in alfa vrednost (RGB + alpha =RGBA barvni model). Vse vrednosti so v intervalu od 0 do 255. Za razumevanje je potrebno poznati RGBA barvni model.

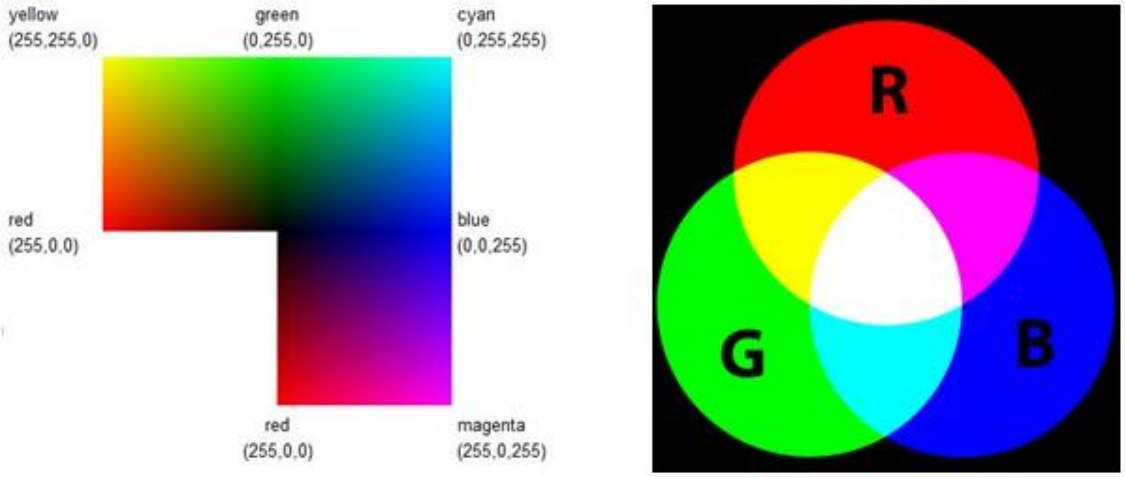
3.6.1 RGB in RGBA

RGB je kratica za barvni model in prihaja iz angleškega jezika, ker določa angleške začetnice treh barv: red, green, blue (rdeča, zelena, modra). Z mešanjem teh treh barv dobimo paleto barv. Vsaka od teh barv se pojavi v 256 odtenkih, kar skupno znaša 256*256*256 (16 777 216) barv.

Glavni namen barvnega modela RGB je prikazovanje, zaznavanje in predvajanje slikovnega gradiva na elektronskih sistemih. Tipične vhodne elektronske naprave RGB so profesionalne video kamere, skenerji in digitalne kamere. Izhodne naprave RGB pa so televizorski in računalniški zasloni, kot je LCD zaslon, plazme in CRT zasloni, mobilni telefonski zasloni, videoprojektorji, večbarvni zasloni LED in velikanski zasloni JumboTron. [14]

Barvni model RGB je predstavljen z 24-biti. To pomeni da je vsaka od treh barv (rdeča, zelena, modra) predstavljena z 8 bitnim nepredznačenim številom (interval vrednosti od 0 – 255). Ta predstavitev je tipična za slikovne datotečne formate JPEG, BMP, TGA ter TIFF.

- *(0, 0, 0) - črna*
- *(255, 255, 255) - bela*
- *(255, 0, 0) - rdeča*
- *(0, 255, 0) - zelena*
- *(0, 0, 255) - modra*
- *(255, 255, 0) - rumena*
- *(0, 255, 255) - sinja*
- *(255, 0, 255) - turkizna*

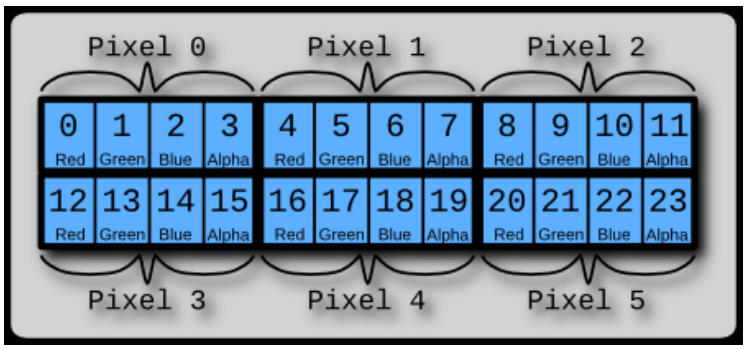


Slika 16: Mešanje barv v RGB barvnem modelu

RGBA pa je 32-bitna predstavitev slikovne pike, kar pomeni da je vsaka pika predstavljena še dodatno z osmimi biti. Ta dodatni 8-bitni podatek določa prosojnost ali alfa vrednost. Datotečni format za tovrstno predstavitev slik je PNG.

Slika 17: Barvna paleta RGB z vrednostjo alfa za prosojnost (RGBA) na kockastem ozadju.

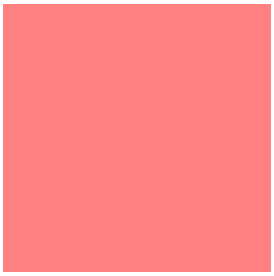
Vrnimo se nazaj k vmestniku JavaScript 2D Context API in na objekt *ImageData*, ki vsebuje podatke o slikovnih pikah. Lastnost objekta *ImageData*, *CanvasPixelArray*, je polje vseh slikovnih pik, ki so povrsti zajete od leve proti desni ter vrstico za vrstico od vrha proti dnu na določeni sliki [12,14].



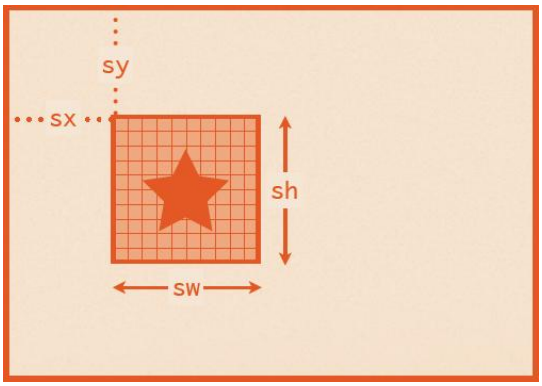
Slika 18: Grafični prikaz canvasPixelArray, polje vrednosti vseh slikovnih pik.(Vir: http://beej.us/blog/2010/02/html5s-canvas-part-ii-pixel-manipulation)

Za lažje razumevanje si oglejmo spodnji primer:

```
var cnv = document.getElementById('c2');
var context= cnv.getContext('2d');

// Ustvarimo ImageData objekt.
var imgd = context.createImageData(100,100);
var pix = imgd.data;

// Z zanko se sprehodimo čez vse slikovne pike in nastavimo slikovno
piko prosojno rdečo.
for (var i = 0; n = pix.length, i < n; i += 4) {
  pix[i] = 255; // kanal rdeče barve
  pix[i+3] = 127; // alfa kanal
}
// Narišemo ImageData objekt na podanih (x,y) koordinatah.
context.putImageData(imgd, 0,0);
```



Slika 19:Izris kvadrata z nastavitvijo barvne vrednosti preko polja canvasPixelArray.



Slika 20: Grafični prikaz, kako z metodo getImageData(sx, sy, sw, sh) zajamemo del slike.

Z objektom *ImageData* pa lahko naredimo še nešteto drugih in zanimivih stvari. Lahko delamo raznovrstne manipulacije s slikami (slikovnimi pikami), jih filtriramo ali delamo raznovrstne matematične vizualizacije.

S spodnjo kodo lahko preprosto naredimo filter za barvno inverzijo slike:

```
// Zajamemo slikovne pike v polju CanvasPixelArray za podane
koordinate in dimenzije
var imgd = context.getImageData(x, y, width, height);
var pix = imgd.data;

// Sprehodimo se čez slikovne pike in invertiramo barvo
for (var i = 0, n = pix.length; i < n; i += 4) {
      pix[i  ] = 255 - pix[i  ]; // rdeča
      pix[i+1] = 255 - pix[i+1]; // zelena
      pix[i+2] = 255 - pix[i+2]; // modra
      // i+3 je alfa (četrti element)
}

// Narišemo ImageData objekt na podanih (x,y) koordinatah.
context.putImageData(imgd, x, y);
```

Slika 21: Primer barvne inverzije vseh slikovnih pik.

3.7 Animacije na znački canvas

V prejšnjem poglavju sem opisal vse osnovne metode in tehnike za izrisovanje črt, krivulj, likov, slik, manipulacije z barvami različnih slikovnih elementov. Vse to znanje zadostuje za izdelovanje preprostih animacij.

Zelo preprosto je izdelovati (interaktivne) animacije na znački canvas, če seveda uporabimo JavaScript. Ker canvas sprva ni bil namenjen kot predloga za animacije, kot je naprimer Flash, je tu kar nekaj omejitev. Največja omejitev je seveda način izrisovanja na canvas. Ko je lik namreč narisan na canvas, ostane na tem mestu. Če bi ga radi premaknili na drugo mesto, ga je potrebno ponovno izrisati, prav tako pa tudi vse ostale elemente, ki so bili že prej narisani. To seveda vzame veliko časa za ponovno izrisovanje, posebno, če je potrebno ponovno izrisati kompleksne okvirje. Učinkovitost predvajanja oziroma izrisovanja je v veliki meri odvisna od hitrosti našega računalnika.

Osnovni koraki izdelovanja animacij (koraki, ki so potrebni za izris določenega slikovnega okvirja animacije):

1. **Počistimo element canvas.** Če je na canvas izrisan kakšen lik, slika, skratka element, razen, če bi radi, da je v animaciji prisoten kot ozadje, ga je potrebno izbrisati, preden narišemo neko novo stvar. Najenostavneje to naredimo z metodo *clearRect.*
2. **Shranimo trenutno stanje na canvas.** Če hočemo ohraniti stanje na canvas (npr. kot ozadje) za naslednje slikovne okvirje, radi pa bi izrisali nek nov element, lik, oziroma spremenili trenutno stanje z raznimi transformacijami, je porebno trenutno stanje najprej shraniti (*context.save*())
3. **Izrisovanje animirane oblike na canvas.** Korak, kjer izrišemo trenutni okvir za prikazovanje.
4. **Povrniti shranjeno stanje na canvas.** Če smo v prejšnjem koraku shranili stanje na canvas, in bi ga radi uporabili v novem slikovnem okvirju, je potrebno povrniti kar smo shranili, pred korakom 3. (*context.restore())*

3.7.1 Upravljanje animacij

Elemente oziroma like izrišemo na canvas s pomočjo metod vmestnika canvas neposredno, ali posredno s klicem funkcij v katerih uporabimo te metode. Rezultat vidimo na canvas, ko se skripta izvede. Animacije pa ni možno izvesti z for zanko.
Potrebujemo način za izvajanje naših funkcij za izrisovanje na canvas v nekem časovnem obdobju. Poznamo dva načina za kontrolo izvajanja animacij [15]:

1. Funkciji ***setInterval*** in ***setTimeout***, ki jih uporabimo za klic določenih funkcij za izrisovanje v določem časovnem trenutku:
    - *setInterval(animateShape,500)* uporabimo, kadar želimo, da se funkcija *animateShape* izvede zaporedoma vsake pol sekunde
    - *setTimeout(animateShape,500)* se izvede le enkrat po preteku časa, ki ga določimo (po pol sekunde)
2. Drugi način za animiranje je s pomočjo uporabnika. Če želimo narediti naprimer igro, uporabimo interakcijo miške ali tipkovnice (ang. *mouse and keyboard event*). Te dogodke potem ujamemo z poslušalci dogodkov (ang. *eventListner*) in izvedemo našo animacijsko funkcijo.

Prvi način sem uporabil tudi v praksi, kjer sem izdelal preprosto panoramo:

Slika 22: Izvorna slika panorame.

Programska koda za preprosto panoramo:

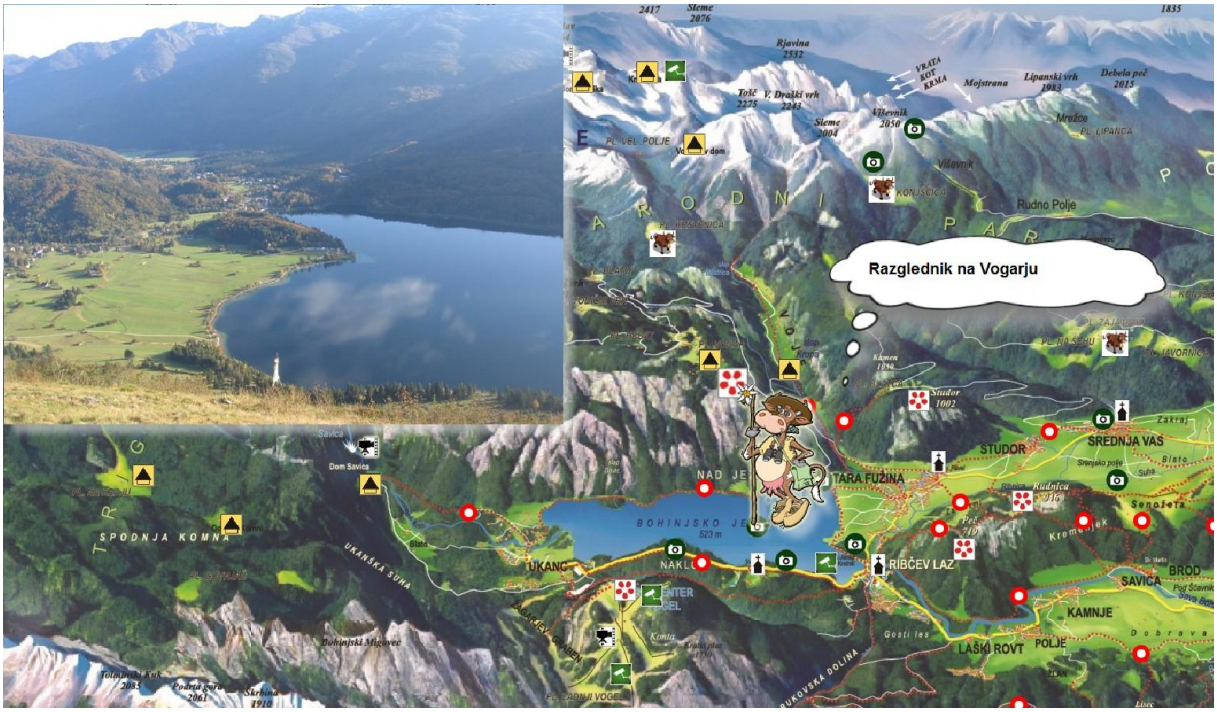
```
<script>
var img = new Image();
img.src = 'panorama.jpg';
var CanvasXSize = 1024;
var CanvasYSize = 227;
var speed = 30;
var y = 0;
var dx = 0.75;
var imgW = img.width;
var imgH = img.height;
var x = CanvasXSize-imgW;
var ctx;

function init() {
      ctx = document.getElementById('canvas-pan').getContext('2d');
      return setInterval(draw, speed);
}

function draw() {
      ctx.clearRect(0,0,imgW,imgH);
      if (x > (CanvasXSize-imgW)) { ctx.drawImage(img,x-
imgW+1,y,imgW,imgH); }

      ctx.drawImage(img,x,y,imgW,imgH);
      if(x>imgW)
      x = CanvasXSize-imgW;
      x += dx;
}
</script>
```

Vsake 30ms pobrišemo canvas in izrišemo ponovno del izvorne panoramske slike z zamikom v levo za trenutno vrednost *x-imgW+1*.

Slika 23: Del slike v preprosti panorami v določenem trenutku.

Drugi način pa sem uporabil v interaktivni panoramski karti Bohinja. Izdelal sem animacijo, v kateri sem uporabil interakcijo miške in tipkovnice. Z določenimi tipkami vodimo grafično ikono (maskoto Bohinja) po panoramski karti. Glede na lokacijo ikone na

karti prikazujemo določeno zanimivost na njej v obliki besedila, slike, zvoka ali video vsebine. V tem primeru ne izrisujemo slike na canvas po preteku določenega časa ali v intervalih kot v prvem načinu, temveč le v trenutku ko sprožimo dogodek z miško ali tipkovnico. V nasprotnem primeru bi bil procesor preveč obremenjen, saj bi bilo potrebno vsakih nekaj milisekund izrisovati na canvas preveč objektov. V animaciji sem namreč uporabil platno (canvas) z veliko resolucijo: 2000×1415 slikovnih pik.



Slika 24: Interaktivna karta Bohinja

Koraki pri izdelavi animacije:

I. Na canvas izrišemo panoramsko sliko v enaki resoluciji kot canvas (2000×1415).
II. Na canvas izrišemo vse ikone, ki so obkljukane v legendi pod značko canvas.
III. Na canvas izrišemo grafično ikono (kravico), glede na trenutno lokacijo (x,y) in trenuten gib.
IV. Preverimo trenutno pozicijo grafične ikone in po potrebi izrišemo besedilo, sliko, video ali predvajamo audio.
V. Ker ima canvas zelo veliko resolucijo, se glede na resolucijo okna našega brskalnika po potrebi premakne vsebino okna za dano število pik vodoravno ali navpično, da je naša ikona vedno v vidnem polju.

Pri vsakem dogodku z miško ali tipkovnico izvedemo vse korake po zgornjem vrstnem redu. V primeru dogodka s tipko uporabimo naslednje tipke za krmiljenje grafične ikone: (A – levo, D – desno, W – navzgor, S – navzdol, Q - navzgor levo, E - navzgor desno, Y - navzdol levo, X - navzdol desno). Pri vsakem dogodku z tipko se spremenita koordinati *x* ali *y* glede na smer premika za določeno število slikovnih pik (9 px). Grafična ikona je animirana, kar pomeni da ima 9 gibov za vsako od smeri:

Slika 25: Izvorna slika za animiranje, ki vsebuje vse možne gibe.

Na zgornji sliki vidimo vse možne gibe ikone. V določenem dogodku s tipko izrišemo le del zgornje slike na canvas (za primer sem obarval ozadje z modro barvo). To storimo z nam že znano metodo *ctx.drawImage(img1,bofestSprite,0,110,176,x,y,110,176)*, kjer imamo v spremenljivki *img1* nastavljeno zgornjo sliko, v spremenljivki *bofestSprite* pa shranjujemo trenutno horizontalno pozicijo dela slike (v velikosti 110×176 slikovnih pik), ki se nastavlja glede na smer gibanja. V spremenljivkah *x* in *y* pa je shranjena lokacija na canvas oziroma na naši panoramski karti.
V koraku IV. preverjamo trenutno pozicijo naše kravice. Vsak objekt ima podatek o lokaciji grafične ikone, vrsti grafične ikone, besedilu, ki opisuje objekt, podatek o lokaciji video, audio ali slikovne vsebine.
Glavni razred *Sign* in metoda *isOver*, s katero preverjamo pozicijo naše kravice na canvas:

```
function Sign(ix, iy, description, imgPath, icon) {
      this.ix = ix;
      this.iy = iy;
      this.description = description;
      this.imgPath = imgPath;
      this.icon=icon;

      if(icon==signIcon) {
            this.shape = new Image();
            this.shape.src="data/" + imgPath;
      }
      if(icon!=(signIcon||videoIcon)) {
            this.photo = new Image();
            this.photo.src=imgPath;
      }
      if(icon==meadowIcon) {
            this.audio = new Audio('cowbells.ogg');
      }
      if(icon==viewPointIcon) {
            this.audio = new Audio('avsenik.ogg');
      }
      if(icon==videoIcon) {
            this.video = document.getElementById(imgPath);
      }
}
Sign.prototype.isOver = function() {
      if (left && x >= this.ix && x <= this.ix+30 &&
       y >= this.iy  && y <= this.iy+30) {
       return true;
      }
      else if(right && x+110 >= this.ix && x+110 <= this.ix+30 &&
       y >= this.iy  && y <= this.iy+30) {
       return true;
      }
      else {
       return false;
}}
```

Element canvas prikazuje od resolucije odvisno rastersko sliko. Vse kar nanj narišemo je eno, ravnina, slika. Vsaka sprememba zahteva ponovno risanje celotne slike.
To pa ne velja za vektorsko grafično tehnologijo SVG (*scalable vector grafic*). Elementi, ki jih rišemo so ločeni objekti DOM in so shranjeni scenskem grafu.

3.8 SVG - Scalable Vector Graphics (skalaribilna vektorska grafika)

SVG je specifikacija za opis dvodimenzionalne vektorske in vektorsko-rastrske grafike v formatu XML. Vključuje transparentnost, filtrske učinke (senčenje, osvetljevanje), skriptne jezike in animacije. Septembra 2001 ga je za uporabo predlagal W3C. Pri nastanku SVG formata so sodelovala vsa najpomembnejša podjetja s področja slikovnih storitev (Adobe, Macromedia, Corel,...). Je označevalni jezik za opisovanje dvodimenzionalne vektorske grafike, ki temelji na XML in je kompatibilen z XML 1.0. Pozna tri tipe grafičnih elementov: vektorski elementi (npr. linije, krivulje, poligoni), rastrske slike in besedilo. Vsak lik je shranjen kot objekt v scenskem grafu (ang. scene graph) ali v DOM. Nad objekti je mogoče izvajati vse običajne slikovne transformacije. S pomočjo skripnega jezika JavaScript lahko naredimo SVG grafiko interaktivno in dinamično. SVG format ima glede na druge formate za opise slik naslednje prednosti:

- dober nadzor nad barvami,
- možnost povečave izbranih delov slike,
- kratke datoteke z opisom slike,
- besedila so shranjena z opisom slike – možnost iskanja,
- neodvisnost od izhodnih naprav in vrst računalnika
- interaktivnost in sprotno ustvarjanje slik
- animacija.

Za pregledovanje slik v obliki SVG potrebujemo vtičnik (plug-in), ki si ga lahko snamemo na Adobe-ovi strani brezplačno. Slike v formatu SVG lahko urejamo s programi kot so Adobe Illustrator, Corel Draw,... Je alternativa za slikovne formete GIF, PNG, JPEG, predvsem zaradi hitrejšega prenosa podatkov, zaradi povečevanja oziroma zmanjševanja in tiskanja z visoko resolucijo.

3.8.1 Smiselnost uporabe SVG

SVG grafiko lahko uporabljamo na spletnih straneh ali v namiznih aplikacijah, ki so lahko v interakciji z tekstovnimi podatki. SVG stran je lahko zgrajena iz podatkovne baze, lahko jo je najti z iskalnikom. Lahko pretvorimo numerične XML podatke v grafične s pomočjo transformacije XSL. XVG stran je skalaribilna in interaktivna. S pomočjo SVG lahko prikazujemo mesta ali zgradbe in uporabnik lahko išče po njih z izbiranjem lokacij. SVG lahko prosto uporabljamo brez potrebne licence.

Uporaba formata SVG na odjemalcu:

- predstavlja alternativo za formate GIF, JPEG, PNG (hitrejši prenos, povečevanje oz. pomanjševanje, tiskanje z visoko resolucijo),
- prihodnost načrtovanja svetovnega spleta je v mešanju formatov HTML (tekst, tabele, forme), SMIL (multimedija - implementiran je v RealPlayerju, podprt pa tudi v IE5.5) in SVG (precizna grafika),
- SVG na dlančnikih.

Uporaba formata SVG na strežniku:

- enostavno generiranje (samo tekst),
- kompatibilen z XSLT in preostalimi specifikacijami XML,
- enostavno konvertiranje v druge formate,
- SVG kot format za izmenjavo podatkov.

3.8.2 Uporaba SVG

Datoteka SVG je XML dokument z grafičnimi oznakami. Pregledujemo ga lahko z brskalnikom, ki ima nameščen potreben vtičnik (plug-in) ali s posebnim pregledovalnikom. SVG lahko uporabimo kot:

- samostojno spletno stran
- vgrajeno z vključenimi oznakami: img src, object, applet, iframe
- povezave z oznakami za povezave <a>
- vključene kot CSS ali XLS lastnost v XHTML
  Primer:
  ```
  <embed src="webmasting.svg" type="image/svg+xml" width="200"
  height="100">
  ```

3.8.3 Glavne funkcije SVG

- transformacije
- transparentnost
- filtrski učinki (senčenje, osvetljevanje)
- prelivi, teksture
- polnitev: barvanje vsebine kot so liki in besedilo
- očrtanje: barvanje obrob likov ali besedila
- izrezovanje
- filter učinki
- predloge objektov
- vstavljanje simbolov ali slik po koordinatah
- interaktivnost, upravljanje z dogodki
- dinamičnost
- skriptni jeziki
- animacije

3.8.4 Programiranje elementov SVG

Javanski programi lahko za prikaz, izdelavo in prirejanje grafike SVG uporabljajo orodni pribor Batik SVG Toolkit. Tudi SVG potrebuje risalno površino, ki jo imenujemo canvas. Batik je opremljen z platnom Java imenovanim *»SVGCanvas2D«* ki pretvori narisano v SVG. Če hočemo vizualno zgraditi SVG dokument, potrebujemo komponento, ki prikazuje SVG vsebino in nam omogoča interakcijo z vsebino (vstavljanje objektov, rotiranje, vstavljanje besedila,..). JSVGCanvas je Java komponenta, ki to omogoča. Potrebno je tudi upravljati z SVG dokumentom preko skriptnega jezika z vmestnikom DOM API.

3.8.5 Komponente SVG

Dokument SVG je sestavljen iz naslednjih podatkovnih tipov:

- vektorskih grafičnih oblik
- slik

- besedil
- skladišč drugih elementov
- lastnosti slogov

Elementi:

- canvas: površina na katero se riše
- viewport: pravokotna področja na canvas
- use: predloga
- property: parameter za način prikazata dokumenta
- grafične oblike: pravokotnik, krog, elipsa, črta, krivulja, poligon
- slika
- besedilo
- simboli
- kontejner

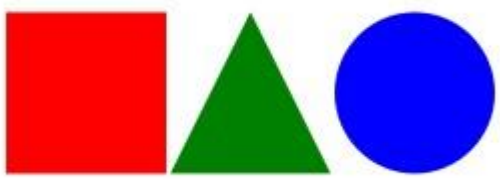
3.8.6 Datotečni format SVG

SVG datoteka ima končnico ».svg« pod OS Linux in Windows. Kompresirana, skrčena datoteka ima končnico ».svgz«. SVG format lahko pretvarjamo v formate GIF, PNG, JPG ali druge rastrske formate slik.
SVG datoteka nima oznake tipa dokumenta (doctype), namesto tega uporablja informativno oznako:

```
<svg version="1.1"
baseprofile="full"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ev="http://www.w3.org/2001/xml-events">
```

Primer SVG kode:

```
<g transform="translate(250,150)">
    <g transform="translate(-82,0)">
      <rect id="square" fill="red" x="-40" y="-40" width="80"
height="80"
          onmousedown="doMouseDown(evt)"/>
    </g>
    <g transform="translate(82,0)">
      <circle id="circle" fill="blue" cx="0" cy="0" r="40"
          onmousedown="doMouseDown(evt)"/>
    </g>
    <g transform="translate(0,0)">
      <polygon id="triangle" fill="green" points="-40,40 0,-40
40,40"
        onmousedown="doMouseDown(evt)"/>
    </g>
  </g>
```



Slika 26: Izris likov v SVG tehniki.

Slika 27: Primer rasterske in vektorske slike.

Slika ponazarja razliko med rastrskimi in vektorskimi slikami. Rasterska slika je sestavljena iz točno določenega števila slikovnih pik, medtem ko je vektorska iz točno določenega števila oblik. Vektorsko sliko lahko poljubno raztegujemo in kvaliteta ostane ista, medtem ko pri rasterski raztegujemo slikovne pike.

3.9 SVG in CANVAS

SVG prikazuje grafiko v brskalniku na bistveno višjem nivoju kot canvas. Vsaka narisana krivulja ali lik je shranjena kot objekt v scenskem grafu ali v DOM, ki je takoj zatem prikazana na rasterskem zaslonu. Če spremenimo atribut objektu, SVG tako samodejno ponovno prikaže samo spremenjeni objekt na zaslonu.

Kot sem že prej opisal, je drugače pri tehnologiji HTML5 canvas. Če na primer narišemo na canvas rdeč pravokotnik, si sistem ne zapomni, da je bil narisan. Če mu torej spremenimo položaj na canvas platnu, se celotna slika ponovno izriše, vključno z objekti, ki so sedaj mogoče prekriti s pravokotnikom. Toda v ekvivalentnem vmestniku SVG API je enostavno samo spremenjena pozicija objekta in brskalnik se sam odloči kako ga prerisati. Možno je ustvariti canvas v večih plasteh in potem lahko le spreminjamo posamezno plast. SVG DOM lahko predstavimo v statičnem XML in kompleksne prikaze upravljamo in urejamo z XML urejevalniki.
SVG scenski graf omogoča dogodke povezane z objekti, tako da lahko ustvarimo pravokotnik z *onmousedown* dogodkom. Če želimo doseči enako funkcionalnost v canvas primeru, je potrebno ročno ujeti koordinate, kjer smo z miško klikali, da narišemo pravokotnik. V tem je SVG bistveno lažje uporabiti.

Canvas je torej v primerjavi z protokolom SVG na nižjem nivoju. Seveda je to normalno, saj sta neodvisna standarda. Canvas naj bi bil hitrejši kot SVG, ker mu ni potrebno ustvariti scenskega grafa. Scenski graf pa je uporaben za optimiziranje animacij. Canvas je primernejši za igre ali animacije, ko je veliko objektov izrisanih zaporedoma, medtem ko je SVG tehnika primernejša za prikaz velikih površin (npr. Google Maps).[16,17]

SVG	**CANVAS**
Scenski graf, vsak lik kot objekt DOM	Samostojen HTML element, upodabljanje grafike preko skriptnega jezika JavaScript
Osnovni elementi liki (drevo objektov)	Osnova slikovne pike
Težje ga je kombinirati z HTML	Obnaša se kot slika v HTML
Za animacije srednje primeren	Animacije na visokem nivoju, primerno za igre, 20% hitrejši
Enostavno rokovanje z dogodki	Težje rokovanje z dogodki

Tabela 3: Primerjava SVG in Canvas.

Canvas pa ni le predloga za prikazovanje likov, črt in slik, temveč na njem lahko prikažemo tudi video vsebino ter predvajamo zvok. Prav tako pa lahko manipuliramo z video in avdio vsebino.

3.10 Video na znački canvas

Na elementu canvas je mogoče predvajati tudi video vsebino. S pomočjo JavaScript je mogoče z video vsebino tudi manipulirati in jo istočasno prikazovati na canvas, na podoben način kot slikovno gradivo. Ustvarjamo lahko raznovrstne vizualne efekte s JavaScript kodo.

Oglejmo si spodnji primer:

```
</head>
<script type="text/javascript;version=1.8" src="main.js"></script>
</head>
<body onload="processor.doLoad()">
<div>
<video id="video" src="video.ogv" controls="true"/>
</div>
<div>
<canvas id="c1" width="160" height="96"/>
<canvas id="c2" width="160" height="96"/>
</div>
</body>
```

V zgornji kodi sem ustvaril dva elementa canvas, *c1* in *c2*. Canvas *c1* sem uporabil za prikaz trenutnega slikovnega okvirja originalnega videa, canvas *c2* pa sem uprabil za prikaz zmanipuliranega videa. Originalnemu filmu sem odstranil ozadje. Za to sem uporabil datoteko *main.js*. Ko se dokument naloži, se izvede metoda *processor.doLoad()* v *main.js*. V metodi *doLoad()* pripravimo spremenljivke reference, ki jih potrebujemo za procesiranje slikovnih okvirjev in dodamo *event listner* metodo *addEventListener()*, da zaznamo, ko uporabnik začne z predvajanjem in procesiranjem videa:

```
doLoad: function() {
    this.video = document.getElementById("myVideo");
    this.c1 = document.getElementById("c1");
    this.ctx1 = this.c1.getContext("2d");
    this.c2 = document.getElementById("c2");
    this.ctx2 = this.c2.getContext("2d");
    var self = this;
    this.video.addEventListener("play", function() {
        self.width = self.video.videoWidth ;
        self.height = self.video.videoHeight ;
        self.timerCallback();
      }, false);
  },
```

Nato pa pokličemo metodo *timerCallback()* za začetek predvajanja videa in procesiranje videa:

```
timerCallback: function() {
    if (this.video.paused || this.video.ended) {
      return;
    }
```

```
    this.computeFrame();
    let self = this;
    setTimeout(function () {
        self.timerCallback();
      }, 0);
  },
```

Manipulacijo slikovnega okvirja pokličemo z metodo *computeFrame()*:

```
 computeFrame: function() {
    this.ctx1.drawImage(this.video, 0, 0, this.width, this.height);
    var frame = this.ctx1.getImageData(0, 0, this.width,
this.height);
           var l = frame.data.length;

    for (var i = 0; i < l; i++) {
      var r = frame.data[i * 4 + 0];
      var g = frame.data[i * 4 + 1];
      var b = frame.data[i * 4 + 2];
      if (g > 100 && r > 100 && b > 100)
        frame.data[i * 4 + 3] = 0;
    }
    this.ctx2.putImageData(frame, 0, 0);
    return;
  }
```

V drugi vrstici je video skopiran v grafični kontekst *ctx1*, ki se predvaja na canvas *c1*.
V tretji vrstici pa z metodo *getImageData()* zajamemo polje slikovnih pik trenutnega slikovega okvirja *ctx1*. Uporabimo for zanko, da se sprehodimo čez vse slikovne pike. V vsakem obhodu shranimo v spremenljivke rdečo *r*, zeleno *g* in modro *b* vrednosti slikovne pike ter za vse spremenljivke, ki imajo vrednost večjo od 100, določimo 100% prosojnost. Sedaj lahko postavimo zmanipuliran video na primerno ozadje.

Slika 28: Manipulacija slikovnih pik video vsebine.

Na spletu obstaja že veliko tovrstnih primerov. Na raznovrstne načine je možno eksperimentirati z canvas in video značkami.
Z preprosto metodo *Canvas.drawImage()* torej lahko ustvarimo mnogo zanimivih načinov prikazovanja slik ali le delov slike.
Kot sem že v poglavju 3.5 navedel, metodo *drawImage* lahko uporabimo na več načinov:

- *drawImage(image, dx, dy)*
- *drawImage(image, dx, dy, dw, dh)*
- *drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh)*

Podrobnosti posameznih argumentov sem že navedel v omenjenem poglavju. Ob tem pa bi dodal bistveno stvar te metode, da vmestnik Canvas 2D API omogoča, da je argument *image* lahko *HTMLImageElement* (slika), *HTMLCanvasElement* (nek drug canvas) ali *HTMLVideoElement* (video vsebina). To daje elementu canvas novo razsežnost. Vsak element lahko prosto rotiramo, premikamo, skratka lahko uporabljamo vse to v raznovrstnih animacijah:

V spodnjem primeru sem prikazal, kako enostavno prikažemo video na canvas s pomočjo *drawImage* metode:

```
<body onload="init()" >
<div>
<video id="sourcevid" autoplay="true" loop="true">
     <source src="nashida.ogg" type="video/ogg"/>
</video>
     <canvas id="sourcecopy" width="320" height="240"></canvas>
</div>
<div>
     <canvas id="output" width="640" height="360"></canvas>
 </div>
</body>
```

Slika 29: Primer prikaza video na canvas.

Uporabil sem metodo *drawImage* v dveh različicah. Prva enostavno prekopira video v izvornih dimenzijah na canvas *sourcecopy*. Na canvas *output* pa predvajamo obrezan video iz canvas *sourcecopy*. Poleg tega pa smo ga s pomočjo metode *draw.rotate* obrnili za 40 stopinj v desno:

```
<script type="text/javascript">
var video;
var copy;
var copycanvas;
var draw;
var RAD = Math.PI/180;

function init(){
video = document.getElementById('sourcevid');
copycanvas = document.getElementById('sourcecopy');
copy = copycanvas.getContext('2d');
var outputcanvas = document.getElementById('output');
draw = outputcanvas.getContext('2d');
setInterval("processFrame()", 33);

}
function processFrame(){
 copy.drawImage(video, 0, 0);
 draw.save();
 draw.translate(50, 50);
 draw.rotate(40*RAD);
 draw.drawImage(copycanvas, 10, 10, 200, 200, 150, -120, 200, 200);
 draw.restore();
}
</script>
```

Na spletu sem zasledil veliko zanimivih primerov, ki jih prikazujemo na canvas elementu. Naslednja dva sta narejena podobno kot moj, z le malo več inteligentnosti in javascript kode.

Na spodnjih slikah lahko vidimo primer kako s pomočjo javascripta in dogodka z miško uporabimo canvas za prikazovanje video vsebine na zabaven način. Tudi v tem primeru sta v uporabi dve znački canvas. V prvo je s pomočjo *drawImage* metode izrisana video vsebina. Video in prva značka canvas je na spodnji sliki nevidna. S tem ko smo skopirali video v *canvas 1* smo pripravili vsebino za manipulacijo videa. Sliki prikazujeta le drugega izmed dveh canvas elementov. Dokler ne uporabimo klika na video, na *canvas 2* prikazujemo video, ki smo ga skopirali v *canvas 1* in kot kaže prvi okvir na spodji sliki. Ko uporabimo dogodek z miško (klik), na naslednji sliki vidimo zanimivo predvajanje videa na *canvas 2* v večih kosih po celi znački *canvas 2*. Za preprosto animacijo je potrebno samo nekaj vrstic javascript kode. [18]

Slika 30: Primer 1 manipulacije video vsebine na canvas.

Poleg omenjenega primera pa je zanimiv še spodnji, ki je narejen na podoben način kot prejšnji. Video se namreč predvaja na *canvas 2*, z nekaj vrstic kode javascript pa omogočimo da se medtem vrti okoli svoje osi.

Slika 31: Primer 2 manipulacije video vsebine na canas.

Ta primer najbolje deluje na WebKit osnovanih spletnih brskalnikih.

Vsi zadnji trije primeri delujejo na principu:
[Video se predvaja] → [Video se izrisuje na canvas 1]→ [Izrisovanje fragmentov iz canvas 1 na canvas 2]

Vmestnik Canvas 2D context API deluje izredno hitro, zato z metodo *drawImage* video najprej prekopiramo na *canvas 1*, nato pa z metodami, ki jih omogoča vmestnik naprej manipuliramo z *canvas 1* in prikazujemo na *canvas 2*. To je hitrejša pot, kot da bi v zanki naredili referenco na video značko in brez elementa *canvas 1* predvajali na *canvas 2*. [18]

Seveda pa je uspešnost predvajanja v različnih brskalnikih odvisna od podprtih video formatov ki sem jih navedel v 2.1.1.

4 3D grafika na spletu

Trodimenzionalna grafika danes na spletu še ni razširjena v veliki meri. 3D grafiko najdemo v igrah, virtualnem svetu, za kar rabimo zmogljivejši računalnik in posebno programsko opremo. Uporabniki so postali zahtevni in bi radi brskali po spletu podobno kot uporabljajo programe, igrovje, v 3D principu. Potrošniki so se navadili 3D vsebine predvsem zaradi uporabe te tehnologije v filmih, igrah in drugih vrstah zabave. Zato se je pojavila zahteva za večji in lažji dostop do 3D vsebine na spletu. Obstaja kar nekaj tehnologij, ki se z njo ukvarja, a nobena od njih še ni pripomogla, da bi postala 3D grafika na spletu bolj razširjena. Skozi pregled tehnologij sem se prepričal zakaj je temu tako. [19, 20]

4.1 3D spletne tehnologije

Zmogljivost strojne in programske opreme za prikazovanje resnične 3D grafike se iz dneva v dan veča. Prav zaradi tega obstaja kar nekaj tehnologij, ki omogočajo prikaz 3D podatkov na internetu. V naslednjih odstavkih vam jih bom nekaj na kratko opisal ter predstavil njihove slabosti in prednosti. Opisal vam bom tiste tehnologije, ki so osredotočene le na prikaz 3D grafike.

4.1.1 VRML

VRML (Virtual Reality Modeling Language) ali jezik za modeliranje navidezne resničnosti, je eden najstarejših grafičnih formatov za prikazovanje 3D prizorov na internetu. Prva verzija VRML 1.0 je bila izdana 1995 in je bila osnovana na knjižnici Open Inventor. Open Inventor je objektno orientirano orodje oziroma knjižnica metod in objektov, s katerimi obvladujemo 3D področje. Predstavlja popolne opise 3D scen, prikazovanje poligonskih objektov, osvetljevanje, ima podporo materialom, efekte za povečanje realnosti itd. VRML predstavlja podmnožico Open Inventorja in uporablja samo najbolj pogosto uporabljene lastnosti Open Inventorja. Druga in zadnja verzija je bila izdana leta 1997 pod imenom VRML97 (ali VRML 2.0). V istem letu je bil VRML97 sprejet kot mednarodni standard ISO. VRML je bil in ga še vedno podpirajo glavni spletni brskalniki. Večina prikazovalnikov VRML je brezplačnih. Je večplatformna tehnologija. VRML ni nikoli postal tako razširjen kot so pričakovali. Ko se je 3D grafika na spletu sčasoma izpopolnjevala, je VRML ostal na istem nivoju in se zato ni obnesel kot napredna tehnologija. Danes je VRML znan kot zastarel in vsa pozornost je usmerjena v njegovega naslednika, X3D. [22]

Prednosti:

- Je platformsko neodvisen programski jezik
- Je sprejet kot mednarodni standard
- Je oblikovan izključno za predstavitev 3D vsebine na internetu
- Je podprt z programi za 3D modeliranje, kot sta 3ds Max in Maya

Slabosti:

- Nima realne integracije z HTML-jem
- Zadnja verzija je iz leta 1997 in od takrat ni bila nikoli več izboljšana

4.1.2 X3D

Naslednik VRML je standard X3D in temelji na XML. Je odprt format za predstavitev navideznih 3D-modelov in večpredstavnosti na spletu. Standardni format X3D podpira vektorsko in rastrsko grafiko ter je med drugim namenjen za njeno trirazsežno predstavitev v spletnih brskalnikih. Je posodobitev formata VRML97. Temelji na modelu, imenovanem scenski graf in sestavljenem iz grafičnih vozlišč, ki tvorijo navidezno 3D okolje. Podpira tudi animacijo in uporabniško interakcijo. Prikazovalniki X3D lahko nastopajo kot samostojni programi ali kot dodatki za spletne brskalnike. X3D je postal pogost izmenjalni format za orodja 3D in interne formate, ki jih avtorji želijo prenesti v spletne brskalnike. Ima podporo za video in zvok, animacijo, uporabniško interakcijo, navigacijo, 2D grafiko in CAD podatke. Manipulacija z 3D objekti je možna tudi s programskimi jeziki C, C++ ali Java. Osrednja lastnost X3D jezika ni izrecno proceduralno programiranje, temveč je prednost deklarativnega določanja vizualizacije, načrtovanje animacij in povezav med podatki. X3D je predvsem jezik za izrecno opredelitev in prikaz 3D-geometrije, ki deluje v realnem času. Dodan je uporabniški programski vmesnik API, ki zagotavlja interaktivno programsko okolje za 3D-grafiko in animacijo. Modeliranje in upodabljanje 3D-modelov je zasnovano medopravilno in neodvisno od operacijskega ali strojnega okolja, kar zagotavlja skladna podpora za nizkonivojska grafična pristopa OpenGL in DirectX. [21]
Prednosti:

- Neodvisnost od platforme
- Je standard za prikaz 3D vsebine na spletu
- Podprt z 3D modelirnimi programi kot sta Maya in 3ds max

Slabosti:

- Še vedno ne tako popularen med uporabniki spleta
- Razvoj naj ne bi bil dovolj prilagodljiv

4.1.3 3DMLW

3DMLW (3D Markup Language for Web) je prav tako na XML zapisu temelječ datotečni format za predstavitev različnih 3D in 2D interaktivnih grafičnih vsebin na spletu. Je prosto uporaben datotečni format in temlji na GPL licenci. Prikazovanje datotek 3DMLW v spletnih brskalnikih zahteva ustrezne dodatke, ki za upodabljanje uporabljajo grafični vmesnik OpenGL (Open Graphics Library). 3DMLW poleg golega ali oblikovanega besedila in 3D objektov dopušča tudi slikovne datoteke (*.jpg, .png, .tga*), zvočne datoteke (*.wav, .ogg*). Omogoča tudi interaktivno vsebino preko skriptnega jezika Lua. V prihodnosti nameravajo izdelati tudi podporo za video. Planirajo pa tudi podporo za DirectX. Tehnologija 3DMLW je dokaj mlada. Prva stabilna verzija je izšla v juniju 2009. [22]

Prednosti:

- Kombinacija 2D in 3D vsebine
- Podobnost z XHTML naredi lažje razumljiva
- Je v koraku s časom z novimi tehnologijami

Slabosti:

- Zaenkrat je podprta samo za Windows platformo

- Ni podpore za video format
- Relativno nova tehnologija, zato je še veliko napak in ni tako razširjena med uporabniki

4.1.4 XML3D

XML3D je čisto nov datotečni format za predstavitev 3D vsebine na spletu, ki je še v razvoju. Razvija ga posebna skupina na univerzi Saarland v Nemčiji. V prihodnosti naj bi bila ta tehnologija že vgrajena v najsodobnejše brskalnike brez dodajanja vtičnikov, kar bo uporabnike še bolj približalo k omenjeni tehnologiji. 3D prizor je predstavljen z XML sintakso in je kar del kode v HTML. Ima podporo za enostavne objekte kot so sfere, kocke, cilindri, itd. Uvoz modelov 3D iz drugih programov, kot na primer Maya, naj bi bil prav tako možen. Ima podporo za zvočne datoteke, video datoteke in animacije. Objekte in prizore lahko uporabimo kot povezave do drugih strani. Podobno kot 3DMLW tudi 3DXML namreč lahko kombinira 2D in 3D vsebino.[22]

Prednosti:

- Kombinacija 2D in 3D vsebine
- 3DXML že vgrajena v brskalnike
- Neodvisnost platforme
- Nova tehnologija, v koraku s časom
- 3D prizori so del HTML DOM

Slabosti:

- Nova tehnologija, veliko napak

4.1.5 O3D

O3D je odprtokodni vmestnik JavaScript API za ustvarjanje 3D grafičnih aplikacij, ki delujejo v spletnih brskalnikih in ga je razvilo podjetje Google. Uporabljamo ga za razvoj igrovja, 3D prikazovalnikov, oglaševanja, demo produktov, simulacij, kontrolnih in nadzornih sistemov ter ustvarjanja navideznih svetov. O3D uporablja arhitekturo na podlagi vtičnika in je napisan v programskem jeziku C. Zato lahko komunicira direktno z strojno opremo, kar pomeni, da je hitrost prikazovanje prizorov močno odvisna od grafične kartice. [22]

Slika 32: Primer grafike v O3D.

Prednosti:

- Neodvisnost platforme
- Nova tehnologija, v koraku s časom
- Podjetje Google je najavilo, da O3D ne bo več deloval na podlagi vtičnika, temveč bo JavaScript knjižnica, ki bo delovala v okviru WebGL.

4.2 WebGL

WebGL tehnologija prinaša v spletne brskalnike strojno pospešeno 3D grafiko brez potrebnih vtičnikov in deluje na vsaki platformi s podporo za OpenGL. Je vmestnik API in ne vtičnik, kar je bistvena razlika od drugih spletnih tehnologij za spletno predstavitev 3D grafike kot je VRML ali X3D. Razvijata ga Mozilla Fundation ter Khronos Group. Končna specifikacija standarda naj bi izšla v letu 2010. Standard bo omogočal izdelovanje 3D iger, animacij in vmestnikov, ki bodo lahko delovali v brskalniku tudi v brezpovezavnem načinu. [23, 24]

WebGL združuje dve obstoječi tehnologiji. Prva je JavaScript, programski jezik, ki je zelo razširjen in daje spletnim stranem inteligenco in interaktivnost. Čeprav se učinkovitost JavaScripta v brskalnikih izboljšuje danes relativno hitro, so programi napisani v tem jeziku relativno počasni in omejeni v primerjavi z tistimi programi, ki tečejo na računalnikih lokalno. Druga tehnologija pa je OpenGL ES, 2D in 3D programski grafični vmestnik za naprave kot so pametni telefoni ali naprimer avtomobilski navigacijski sistem, ki imajo omejno procesorsko moč. Če ima računalniški grafični sistem gonilnik OpenGL, lahko programi, napisani za uporabo OpenGL, za preračunavanje zahtevnejših spletnih tridimenzionalnih grafičnih oblik koristijo kar zmogljivost grafičnega procesorja.

4.2.1.1 Delovanje WebGL

OpenGL je programski vmesnik za pisanje računalniških programov, ki prikazujejo 3D in 2D računalniško grafiko. Pri programiranju z OpenGL lahko uporabljamo katerikoli programski jezik, prav tako pa OpenGL deluje na več platformah. Vmesnik sestavlja več kot 250 funkcij, ki omogočajo izrisovanje kompleksnih 3D scen sestavljenih iz preprostih osnovnih geometričnih oblik.[25]

WebGL je torej vmestnik, ki omogoča komunikacijo med JavaScript aplikacijo ter programsko knjižnjico OpenGL. To omogoča uporabo grafičnega procesorja s polno močjo, ki nam vrača 3D vsebino. OpenGL je verzija programske knjižnice za vsako grafično kartico in jo v glavnem podpirajo vsi operacijski sistemi.

Prikazovanje 3D vsebine sloni na novem elementu HTML5 canvas, ki sem ga podrobno predstavil v poglavju 3. Do elementa canvas dostopamo preko vmestnika DOM. Canvas je že implementiran v mnogih brskalnikih, medtem ko je podpora za WebGL trenutno šele v naslednjih treh brskalnikih:

Firefox

WebGL deluje samo v nestabilnih razvojnih izdajah brskalnika Firefox na platformah Windows, Linux in Mac OS X. Nestabilna izdaja Firefox se imenuje Minefield. Za preizkus WebGL je potrebno urediti še nekatere nastavitve. Gonilniki grafične kartice morajo podpirati OpenGL 2.1. V nasprotnem primeru jih je potrebno posodobiti. Za uporabnike operacijskega sistema Windows, ki nimajo grafičnih gonilnikov za OpenGL 2.1 pa obstaja programska

rešitev za prikaz 3D grafike na osnovi WebGL. Programska oprema se imenuje MESA software OpenGL. Seveda pa prikazovanje 3D grafike deluje občutno počasneje. [26]

Safari

WebGL deluje v brskalniku Safari le na operacijskem sistemu Mac OS X 10.6 (Snow Leopard). Potrebno je imeti najmanj 4. verzijo brskalnika Safari in nočno izdajo pogona WebKit za upodabljanje spletnih strani.[26]

Chrome

Delovanje WebGL v Google Chrome je možno le v nočnih izdajah tega brskalnika (Chromium). Deluje v vseh glavnih operacijskih sistemih Windows, Linux in Mac OS. V vseh treh sistemih je potrebno zagnati program z ukazom *»--enable-webgl«,* s katerim omogočimo delovanje WebGL. [26]

4.2.1.2 C3DL

Canvas 3D JS Library je knjižnica JavaScript, ki omogoča lažje pisanje 3D aplikacij na osnovi WebGL. Oskrbuje nas z množico razredov math, scene, 3D object, ki naredijo WebGL razvijalcem lažje dosegljivo. Primerna je za tiste, ki hočejo razviti 3D vsebino v brskalnikih in se nočejo poglobiti v delovanje 3D iz matematičnega vidika, ki je potrebna za delovanje. Knjižnica bo integrirana v brskalnikih Firefox, Chrome in Safari.

WebGL deluje z OpenGL na večini računalnikov. Za tiste uporabnike operacijskega sistema Windows, ki nimajo modula OpenGL, pa obstaja ANGLE (Almost Native Layer Graphics Engine). To je vmestnik, ki ga je izdelal Google in omogoča delovanje WebGL na Direct3D (DirectX).

Slika 33: Primer prikazovanja 3D grafike v WebGL.
Vir:[https://cvs.khronos.org/svn/repos/registry/trunk/public/webgl/sdk/demos/google/particles/index.html]

Slika 34: grafični prikaz povezave med OpenGL in WebGL. Vir [http://www.proe.com/print_article.php?cpfeatureid=45604]

5 Sklepne ugotovitve

V prvem delu svojega diplomskega dela sem želel predstaviti bistvene novosti označevalnega jezika HTML in kako jih programsko vključiti v spletne aplikacije. Večkrat sem naletel na težave, saj nove tehnologije ne podpirajo vsi spletni brskalniki. Tako sem krožil med brskalniki Opera, Chrome in Firefox. Internet Explorer trenutno ne podpira skoraj nobene HTML 5 novosti. V naslednji verziji naj bi podpiral tudi značko canvas.

V drugem delu sem predstavil vmestnik canvas 2D API. Ugotovil sem, da je zelo dobra pridobitev na spletu, saj omogoča enostavno risanje 2D grafike ter manipulacijo slikovnih pik in video vsebine in za svoje delovanje ne potrebuje vtičnikov. Ugotovil sem, da je zelo enostaven in zelo hiter vmestnik za izdelovanje animacij. Problem se je pojavil le pri rokovanju z dogodki. V animirani interaktivni karti sem moral namreč ročno ujeti koordinate na canvas za prikaz določene zanimivosti.

V tretjem delu sem predstavil tehnologije za prikazovanje 3D grafike. Preden bo 3D splet resnično zacvetel, bo naletel na številne prepreke. Za primer vtičniki, uporabljeni v nekaterih primerih, običajno povzročijo sesutje brskalnika in druge probleme. Da bi 3D vsebina nativno delovala v vseh brskalnikih, v vseh operacijskih sistemih in aplikacijah je velik izziv. Pri tem ima glavno vlogo standarizacija. Če se standarizacija ne bo pojavila, bo splet končal kot mešanica različnih formatov, ki bodo delovali v različnih brskalnikih. Prav tako izdelovanje 3D vsebine na spletu terja veliko časa za pisanje, in relativno le nekaj razvijalcev je seznanjenih z načinom obdelave.

Problem je tudi povečanje netbookov in pametnih telefonov, ki imajo počasnejše procesorje, kar pomeni, da je vedno več ljudi, ki uporabljajo te naprave, ne bodo pa sposobne gnati 3D vsebine. Kljub temu, da se je povečala pasovna širina in je strojna oprema postala občutno zmogljivejša, še vedno ni dovolj razširjena za prisotnost kompleksne 3D vsebine na spletu.

Seznam slik

Seznam tabel

Literatura

[1] HTML 5 Dostopno na:
http://en.wikipedia.org/wiki/HTML5

[2] HTML 5 Spec. Dostopno na:
http://dev.w3.org/html5/spec/Overview.html

[3] HTML 5. Dostopno na:
http://www.w3schools.com/html5/

[4] Rethinking Forms in HTML5. Dostopno na:
http://net.tutsplus.com/tutorials/html-css-techniques/rethinking-forms-in-html5/

[5] HTML5 Forms Are Coming. Dostopno na:
http://snook.ca/archives/html_and_css/html5-forms-are-coming

[6] The future of the web: how we'll create forms in HTML5. Dostopno na:
http://www.yourinspirationweb.com/en/the-future-of-the-web-how-well-create-forms-in-html5/

[7] HTML5 Unleashed: Tips, Tricks and Techniques. Dostopno na:
http://www.w3avenue.com/2010/05/07/html5-unleashed-tips-tricks-and-techniques/

[8] Web Storage. Dostopno na:
http://dev.w3.org/html5/webstorage/

[9] Web Storage. Dostopno na:
http://en.wikipedia.org/wiki/Web_Storage

[10] Using offline web applications to offer offline capatibilities sister specification to html5. Dostopno na:
http://robertnyman.com/2010/04/14/using-offline-web-applications-to-offer-offline-capabilities-sister-specification-to-html5/

[11] Let's Call It A Draw(ing Surface). Dostopno na:
http://diveintohtml5.org/canvas.html

[12] Pixel manipulation. Dostopno na:
http://dev.w3.org/html5/canvas-api/canvas-2d-api.html#pixel-manipulation

[13] HTML5's canvas Part II: Pixel Manipulation. Dostopno na:
http://beej.us/blog/2010/02/html5s-canvas-part-ii-pixel-manipulation/

[14] RGB color model. Dostopno na:
http://en.wikipedia.org/wiki/RGB_color_model

[15] Basic animations. Dostopno na:
https://developer.mozilla.org/en/Canvas_tutorial%3aBasic_animations

[16] Canvas element. Dostopno na:
http://en.wikipedia.org/wiki/Canvas_element

[17] Bit of SVG and CANVAS. Dostopno na:
http://nimbupani.com/bit-of-svg-and-canvas.html

[18] Blowing up HTML5 video and mapping it into 3D space. Dostopno na:
http://www.craftymind.com/2010/04/20/blowing-up-html5-video-and-mapping-it-into-3d-space/

[19] Is 3D Finally Ready for the Web? Dostopno na:
http://ieeexplore.ieee.org.nukweb.nuk.unilj.si/stamp/stamp.jsp?tp=&arnumber=5398776

[20] Will HTML 5 Restandardize the Web? Dostopno na
http://ieeexplore.ieee.org.nukweb.nuk.uni-lj.si/stamp/stamp.jsp?tp=&arnumber=5445161

[21] Trirazsežni pristopi za modeliranje stavb, mest in pokrajn. Dostopno na:
http://www.geodetski-vestnik.com/53/4/gv53-4_695-713.pdf

[22] 3D Web Technologies And Their Usability for The Project 3D Mobile Internet. Dostopno na:
http://www.rdc.cz/download/publications/Turonova_2009_3DWeb.pdf

[23] WebGL. Dostopno na:
http://en.wikipedia.org/wiki/WebGL

[24] Q&A. Dostopno na:
http://www.3d-test.com/interviews/khronos_2.htm

[25] OpenGL. Dostopno na:
http://sl.wikipedia.org/wiki/OpenGL

[26] Learning WebGL. Dostopno na:
http://learningwebgl.com/blog/?p=11