Универзитет у Београду

Математички факултет

Мастер рад

Електронске лекције о html5 Canvas технологији

Драгана Станковић 1080/2017

Ментор

др Мирослав Марић

Септембар 2021.

Садржај

Увод **1**

1 Електронске лекције **2**

2 Увод у *HTML5 Canvas* **3**
2.1 Карактеристике *HTML5 Canvas* технологије 3
2.2 Могућности *HTML5 Canvas* технологије 4
2.3 Дефиниција површине за цртање *HTML5 Canvas* технологије . . 4

3 Координатни систем *HTML5 Canvas* технологије **5**
3.1 Цртање помоћу елемента *HTML5 Canvas* коришћењем JavaScript језика 6

4 Цртање линије **7**
4.1 Постављање дебљине линије 8
4.2 Постављање боје линије 9
4.3 Постављање стила крајева линије 9

5 Цртање криве **11**
5.1 Цртање квадратне криве 12
5.2 Цртање Безиерове криве 14

6 Цртање путање **15**
6.1 Начин надовезивања линија 16
6.2 Закривљење путање 17

7 Цртање облика **18**
7.1 Цртање правоугаоника 19
7.2 Цртање круга 20
7.3 Цртање полукруга 21

8 Додавање слике **21**
8.1 Постављање димензије слике 22
8.2 Исецање слике 23

9 Трансформације **25**
9.1 Транслација . . . 25
9.2 Ротација . . . 25
9.3 Скалирање . . . 26
9.4 Прилагођене трансформације . . . 27
9.5 Ресетовање трансформације . . . 27

10 Креирање композиција **28**
10.1 Додавање сенке . . . 28
10.2 Подешавање прозирности . . . 29
10.3 Градијент . . . 30

11 Анимације **32**
11.1 Брисање садржаја . . . 33
11.2 Линеарно кретање . . . 34
11.3 Анимација са убрзањем . . . 36
11.4 Анимација са осциловањем . . . 37
11.5 Анимација са реакцијом корисника . . . 38

12 Креирање аналогног сата **40**
12.1 Додавање оквира . . . 40
12.2 Додавање бројчаника . . . 41
12.3 Додавање бројева . . . 42
12.4 Додавање казаљки . . . 44
12.5 Стартовање сата са радом . . . 46

Закључак **49**

13 Литература **50**

Увод

Појава интернета и општа доступност рачунара довели су до великог напретка у области информационих технологија. Због општег развоја интернета, савремене веб технологије су константо у експанзији. Стога је данас тешко замислити веб-сајт без анимираних слика, различитих стилова типографије, мултимедијалних садржаја и елемената који омогућују више интерактивности.

HTML5 је језик за структуирање и презентовање садржаја на интернету. То је пета верзија *HTML* стандарда и његов основни циљ је побољшање језика подршком за најновије мултимедијалне садржаје, при чему је посебна пажња посвећена разумљивости и читљивости. Састоји се од скупа елемената који се представљају коришћењем етикета (тагова). Сваки елемент описује одређени садржај веб-странице.

Елемент *Canvas* је један од најпопуларнијих елемената *HTML5* језика намењен цртању. Увођење елемента *Canvas-a* је отворило велики број могућности које раније нису биле изводљиве на вебу: креирање дијаграма и графикона на основу података, 3D графика, апликација за цртање, анимације и игре. Подржан је од стране свих већих претраживача и може се користити на разним уређајима од десктоп рачунара до таблета и паметних телефона.

Кроз овај мастер рад су приказане електронске лекције о основним карактеристикама *HTML5 Canvas* технологије и могућностима креирања динамичких графика и анимација које реагују на корисничку интеракцију помоћу *JavaScript* скрипт језика. Лекције су написане са циљем да буду разумљиве свима који се први пут сусрећу са *HTML5 Canvas* технологијом. Састоје се од неопходних текстова, слика, урађених задатака, програмских кодова и корисних линкова. На крају рада представљена је једноставна анимација која илуструје обрађене концепте и показује како *HTML5 Canvas* омогућује креирање динамичких елеманата у оквиру веб странице.

Сви креирани материјали су јавно доступни у оквиру платформе eŠkola veba на адреси:
`http://edusoft.matf.bg.ac.rs/eskola_veba/#/course-details/canvas`.

1 Електронске лекције

Платформа еШкола веба је електронска школа веб програмирања на којој се налазе курсеви о популарним веб технологијама. Сваки курс је организован у виду електронских лекција, које су бесплатне и доступне на српском језику. Изглед платформе приказан је на слици 1. Више о платформи еШкола веба се може видети у [1].

Слика 1: *Почетна страна платформе еШкола веба*

Међу доступним курсевима налази се и курс о *HTML5 Canvas* технологији коме се може приступити на адреси `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course-details/canvas`. Приликом покретања курса отвара се страница са слике 2.

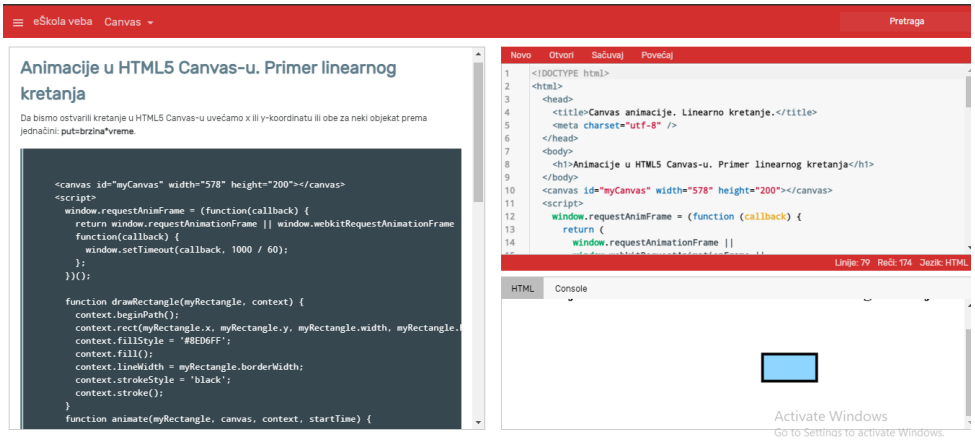
Слика 2: *Насловна страна курса о* HTML5 Canvas *технологији*

У оквиру курса о *HTML5 Canvas* технологији креиране су следеће електронске лекције:

- Координатни систем елемента *HTML5 Canvas*;
- Цртање линије помоћу елемента *HTML5 Canvas*;
- Цртање криве помоћу елемента *HTML5 Canvas*;
- Цртање путање помоћу елемента *HTML5 Canvas*;
- Цртање облика помоћу елемента *HTML5 Canvas*;

- Додавање слика помоћу елемента *HTML5 Canvas*;
- Трансформације у *HTML5 Canvas-y*;
- Креирање композиција помоћу елемента *HTML5 Canvas*;
- Анимације у *HTML5 Canvas-y*;
- Креирање аналогног сата помоћу елемента *HTML5 Canvas*.

Изглед једне од доступних лекција приказан је на слици 3.



Слика 3: *Лекција у оквиру курса*

Овакав тип лекција на енглеском језику можете погледати на [3].

2 Увод у *HTML5 Canvas*

Елемент *HTML* `<canvas>` је таг који се појавио као један од HTML5 новина. Користи се за исцртавање графичких елемената, фотографских композиција и креирање једноставних анимација помоћу неког скрипт језика, најчешће *JavaScript*. *Canvas* `<canvas>` елемент је само контејнер за графику који се одређује помоћу *width* и *height* атрибута. Права моћ елемента *Canvas-a* долази до изражаја кроз употребу *HTML5 API-ja*. Употребом *JavaScript-a* се може приступити читавом сету функција за цртање путања, квадрата, кругова, линија, текста и додавање слика.

2.1 Карактеристике *HTML5 Canvas* технологије

- *Canvas* је потпуно интерактиван. Одговара на акције корисника слушајући догађаје тастатуре или миша. Тако да програмер није ограничен само на статичку графику и слике.
- Сваки објекат u *Canvas-y* може бити анимиран. Ово омогућава креирање анимација свих нивоа, од једноставних лоптица које скакуђу до напреднијих и комплексних анимација.

- Омогућује исцртавање линија, разних облика, текста, слика - са или без употребе анимације. Додавање видео и/или аудио материјала је такође могуће помоћу елемента *Canvas*.

- *HTML5 Canvas* је подржан од стране већине претраживача и може се користити на разним уређајима, од десктоп рачунара до таблета и смарт телефона.

- *HTML5 Canvas* је саставни део *HTML5* технологије. Једном направљен може бити употребљен свуда - од компјутера до мобилних уређаја, без било каквих помоћних програма.

- Основни алати за креирање *HTML5 Canvas* апликација су претраживач и код едитор по избору. Такође, постоји много сајних и бесплатних библиотека и алата за напредније креирање *Canvas* апликација.
  Више о карактеристикама елемента *HTML5 Canvas* се може видети у [5].

2.2 Могућности *HTML5 Canvas* технологије

- *HTML5 Canvas* сет функција је идеалан за прављење разних 2D i 3D игрица.

- *HTML5 Canvas* је одлична замена за прављење банера и реклама базираних на употреби flash технологије.

- Може прикључати податке из глобалних извора података и користити за генерисање визуелно атрактивних табела и графикона.

- Комбинацијом текста, слике, видео и аудио материјала могу се направити атрактивни материјали намењени у едукативне сврхе.

2.3 Дефиниција површине за цртање *HTML5 Canvas* технологије

Сваком *HTML5 Canvas* елементу се мора одредити *context*. *Context* дефинише која врста *HTML5 Canvas* API-ja ће бити коришћена. 2d context се користи за цртање 2D графике и манипулацију bitmap слика. 3d context се користи за 3D креирану графику и манипулацију. Касније је названо WebGL и базирано је на OpenGL ES. Поред *width* и *height* атрибута, може се користити и *CSS* за подешавање величине *Canvas* елемента. Ипак подешавање величине елемента *Canvas* помоћу *CSS-a*, није исто као када се то ради помоћу *width* и *height* атрибута. То је зато што *Canvas* има две величине: величину самог елемента и величину површине за цртање елемента. Када подесите атрибуте *width* и *height*, постављате величину елемента и величину површине за цртање, али када за одређивање величине користите *CSS*, постављате само величину елемента, али не и његове површине за цртање. Када се величина елемента не поклапа са величином површине за цртање, претраживач сам подешава површину како би се уклопила у елемент.

У примеру 1 приказано је креирање елемента *Canvas*, односно његове површине за цртање, *дужине* 200px, *висине* 100px, *дебљине линије* 1px, црне боје. Видети слику 4.

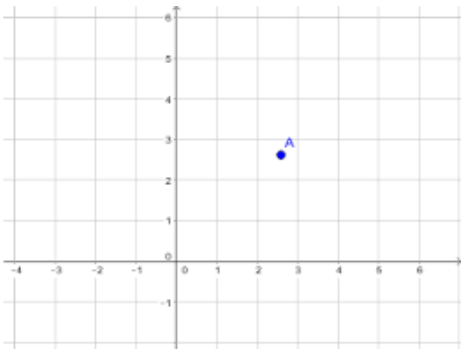
Слика 4: *Canvas површина за цртање*

Пример 1: Canvas површина за цртање

```
<canvas width="200" height="100" style="border:1px solid #000000;">
</canvas>
```

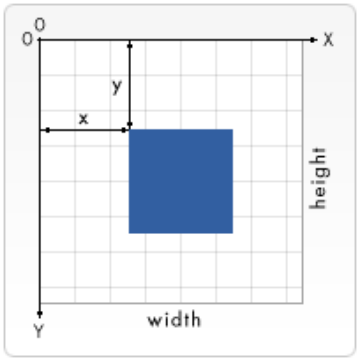
3 Координатни систем *HTML5 Canvas* технологије

У $2D$ простору позиције се одређују употребом x и y координата. x оса се простире хоризонтално, а y вертикално. Центар има позицију $x = 0$ и $y = 0$, може бити и представљено као $(0, 0)$. Ова метода позиционирања објеката се у математици назива Декартов координатни систем (слика 5).



Слика 5: *Декартов координатни систем*

Координатни систем елемента *HTML5 Canvas* (слика 6) поставља почетну тачку у горњем левом углу *Canvas* површине за цртање са x координатом која се простире удесно и y координатом која се простире до дна *Canvas* површине за цртање. За разлику од стандардног Декартовог координатног система, *Canvas* простор нема видљивих негативних тачака.



Слика 6: *Canvas координатни систем*

3.1 Цртање помоћу елемента *HTML5 Canvas* коришћењем JavaScript језика

Сваки вид цртања у *HTML5 Canvas-у* захтева одређени скрипт језик, најчешће *JavaScript*.

Корак 1. Пре свега, треба пронаћи `<canvas>` елемент користећи *HTML DOM* метод *getElementById()*:

```
var canvas = document.getElementById("myCanvas");
```

Корак 2. Дефинисање објекта за цртање користећи *HTML* методу *getContext()*:

```
var ctx = canvas.getContext("2d");
```

Корак 3. Цртање објекта постављањем својства *fillStyle* на црвену боју. Подразумевана вредност за *fillStyle* је црна боја:

```
ctx.fillStyle="#FF0000";
```

Метода *fillRect(x,y,width,height)* црта правоугоник, при чему су x и y координате темена горњег левог угла, а *width* и *height* димензије правоугаоника.

```
ctx.fillRect(0, 0, 150, 75);
```

У примеру 2 приказано је исцртавање правоугаоника чија је унутрашњост попуњена црвеном бојом.

Пример 2: Цртање правоугаоника

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #
    c3c3c3;">
</canvas>
<script>
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");
    ctx.fillStyle="#FF0000";
    ctx.fillRect(0, 0, 150, 75);
</script>
```

Изглед примера 2 у веб претраживачу приказан је на слици 7.

Слика 7: *Цртање правоугаоника*

У наставку рада приказано је исцртавање различитих објеката у оквиру веб странице, а доступност тог садржаја на енглеском језику можете погледати на [4].

4 Цртање линије

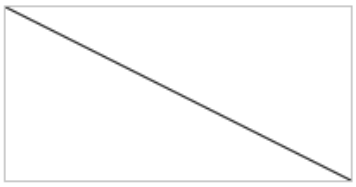
За цртање линије у *Canvas-y*, користи се следеће методе:

- *moveTo(x,y)* - дефинише почетну тачку линије,
- *lineTo(x,y)* - дефинише крајњу тачку линије.

Пример 3: Цртање линије

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #
    c3c3c3;">
</canvas>
<script>
      var canvas = document.getElementById('myCanvas');
      var context = canvas.getContext('2d');
      context.beginPath();
      context.moveTo(0, 0);
      context.lineTo(200, 100);
      context.stroke();
</script>
```

Да би се уствари повукла црта између дефинисане почетне и крајње тачке неопходан је позив методе: *stroke()*. Дефинишемо почетну тачку на позицији $(0, 0)$, и крајњу тачку на позицији $(200, 100)$. Затим позовемо *stroke()* метод за исцртавање линије. Видети слику 8.



Слика 8: *Цртање линије*

4.1 Постављање дебљине линије

Да би се дефинисала дебљина линије потребно је доделити вредност својству *lineWidth*. Вредност својства *lineWidth* мора бити додељена пре позива методе *stroke()*. Пример 4 илуструје исцртавање линије са почетком у тачки (0, 0) и крајем у тачки (200, 100), дебљине 15px.

Пример 4: Постављање дебљине на 15px

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #
    c3c3c3;">
</canvas>
<script>
      var canvas = document.getElementById('myCanvas');
      var context = canvas.getContext('2d');
      context.beginPath();
      context.moveTo(0, 0);
      context.lineTo(200, 100);
      context.lineWidth = 15;
      context.stroke();
</script>
```

Изглед примера 4 у веб претраживачу приказан је на слици 9.

Слика 9: *Цртање линије дебљине 15px*

4.2 Постављање боје линије

Да би се дефинисала боја линије потребно је доделити вредност својству *strokeStyle*. Вредност својства *strokeStyle* може бити додељена у облику текстуалне вредности, хексадецималне вредности или у облику низа целобројних вредности црвене, зелене и плаве.

У примеру 5 приказано је исцртавање линије са почетком у тачки (0, 0) и крајем у тачки (200, 100), црвене боје, дебљине 10px.

Пример 5: Цртање линије црвене боје

```
<canvas id="linija" width="200" height="100" style="border:1px solid #
    c3c3c3;">
</canvas>
<script>
      var canvas = document.getElementById('myCanvas');
      var context = canvas.getContext('2d');
      context.beginPath();
      context.moveTo(0, 0);
      context.lineTo(200, 100);
      context.lineWidth = 10;
      context.strokeStyle = '#ff0000';
      context.stroke();
</script>
```

Изглед примера 5 у веб претраживачу приказан је на слици 10.

Слика 10: *Цртање линије црвене боје*

4.3 Постављање стила крајева линије

Да би се дефинисао изглед крајева линије потребно је поставити вредност својста *lineCap*. Постоје три дефинисане вредности својства *lineCap*: *round* - крајеви линија су заобљени, *square* - крајеви линије се квадрирају додавањем оквира једнаке ширине линије и половине дебљине линије, и *butt* - крајеви линије су квадратни на крајњим тачкама. Подразумевана вредност је *butt*. Својству је неопходно доделити вредност пре позива методе *stroke()*.

У примеру 6 приказано је исцртавање линије црвене боје, дебљине 20px са различитим стиловима крајева линије.

Пример 6: Цртање линије различитих типова крајева

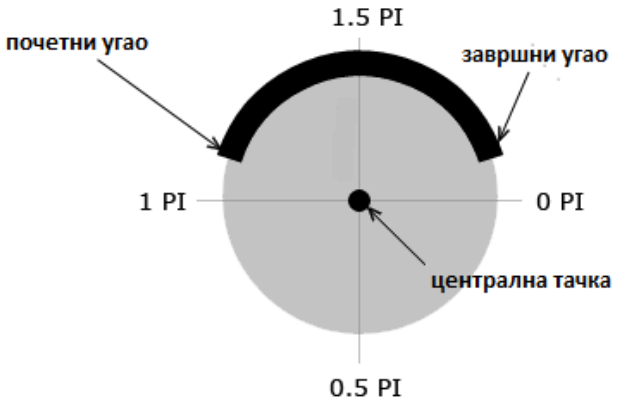
```
<canvas id="myCanvas" width="578" height="200"></canvas>
   <script>
     var canvas = document.getElementById('myCanvas');
     var context = canvas.getContext('2d');

     context.beginPath();
     context.moveTo(200, canvas.height / 2 - 50);
     context.lineTo(canvas.width - 200, canvas.height / 2 - 50);
     context.lineWidth = 20;
     context.strokeStyle = '#0000ff';
     context.lineCap = 'butt';
     context.stroke();

     context.beginPath();
     context.moveTo(200, canvas.height / 2);
     context.lineTo(canvas.width - 200, canvas.height / 2);
     context.lineWidth = 20;
     context.strokeStyle = '#0000ff';
     context.lineCap = 'round';
     context.stroke();

     context.beginPath();
     context.moveTo(200, canvas.height / 2 + 50);
     context.lineTo(canvas.width - 200, canvas.height / 2 + 50);
     context.lineWidth = 20;
     context.strokeStyle = '#0000ff';
     context.lineCap = 'square';
     context.stroke();
   </script>
```

Изглед примера 6 у веб претраживачу приказан је на слици 11.

Слика 11: *Цртање линије различитих типова крајева*

5 Цртање криве

Да би се нацртао лук помоћу *HTML5 Canvas-a*, користи се метода *arc()*. Лукови су дефинисани централном тачком, полупречником, почетним углом, завршним углом и правцем цртања (у смеру казаљке на сату или у супротном смеру). Лукови се могу обликовати својствима *lineWidth*, *strokeStyle* и *lineCap*.



Слика 12: *Цртање лука*

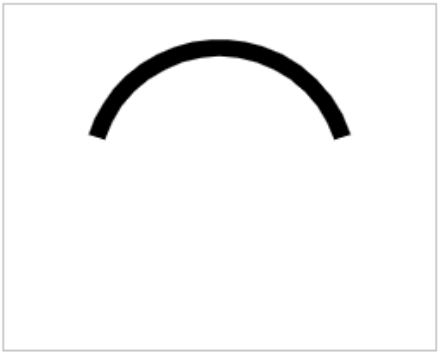
Лук није ништа друго него део замишљеног круга. Овај замишљени круг се може дефинисати помоћу координате центра x и y, и полупречника круга. Лук се може дефинисати и са две тачке дуж замишљеног круга помоћу *startAngle* и *endAngle*. Ова два угла су дефинисана у радијанима и формирају замишљене линије које крећу из центра круга и пресецају крајеве лука који желимо да исцртамо. Видети слику 12.
Последњи аргумент методе *arc()* дефинише смер путање лука између две почетне и крајње тачке. Осим ако није другачије назначено, овај аргумент је подразумевано постављен на *false*, што доводи до повлачења лука у смеру казаљке на сату.

Пример 7 илуструје исцртавање лука као дела замишљеног круга полупречнка 75px, црне боје, дебљине 10px. Углови које граде замишљене линије из центра круга са крајевима лука износе $1.1 * PI$ и $1.9 * PI$, а смер путање лука између почетне и крајње тачке је у смеру казаљке на сату.

Пример 7: Цртање лука

```
<canvas id="myCanvas" width="578" height="200"></canvas>
  <script>
     var canvas = document.getElementById('myCanvas');
     var context = canvas.getContext('2d');
     var x = canvas.width / 2;
     var y = canvas.height / 2;
     var radius = 75;
     var startAngle = 1.1 * Math.PI;
     var endAngle = 1.9 * Math.PI;
     var counterClockwise = false;

     context.beginPath();
     context.arc(x, y, radius, startAngle, endAngle, counterClockwise);
     context.lineWidth = 10;

     context.strokeStyle = 'black';
     context.stroke();
   </script>
```

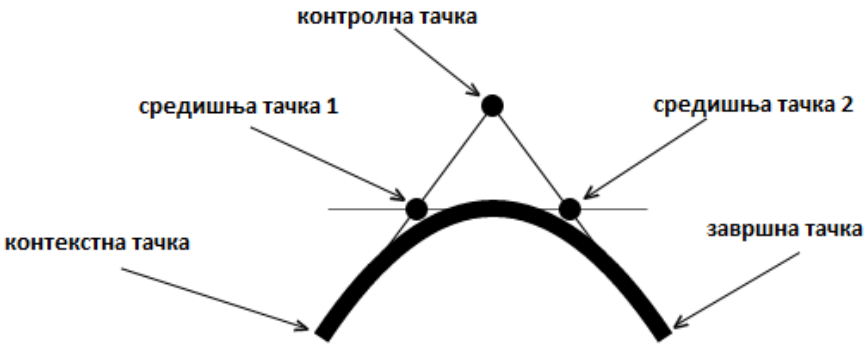
Изглед примера 7 у веб претраживачу приказан је на слици 13.



Слика 13: *Цртање лука*

5.1 Цртање квадратне криве

За цртање квадратне криве помоћу елемента *HTML5 Canvas* користи се метода *quadraticCurveTo()*. Квадратне криве су дефинисане контекстном тачком, контролном тачком и завршном тачком (слика 14). Квадратне криве се могу стилизовати помоћу својстава *lineWidth*, *strokeStyle* и *lineCap*.



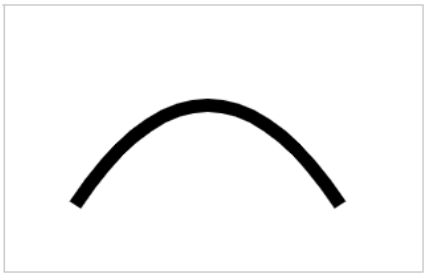
Слика 14: *Цртање квадратне криве*

Контролна тачка дефинише закривљеност квадратне криве креирањем две замишљене тангенцијалне линије које су повезане са контекстном тачком и завршном тачком. Контекстна тачка је дефинисана методом *moveTo()*. Померањем контролне тачке даље од контекстне тачке и крајње тачке исцртава се оштрија кривина, а померањем контролне тачке ближе контекстној тачки и крајњој тачки исцртавају се шире криве. Пример 8 илуструје исцртавање квадратне криве дебљине 10px, црне боје. Координате контекстне тачке су (188, 150), координате контролне тачке су (288, 0), а координате завршне тачке (388, 150).

Пример 8: Цртање квадратне криве

```
<canvas id="myCanvas" width="578" height="200"></canvas>
   <script>
     var canvas = document.getElementById('myCanvas');
     var context = canvas.getContext('2d');
     context.beginPath();
     context.moveTo(188, 150);
     context.quadraticCurveTo(288, 0, 388, 150);
     context.lineWidth = 10;
     context.strokeStyle = 'black';
     context.stroke();
   </script>
```

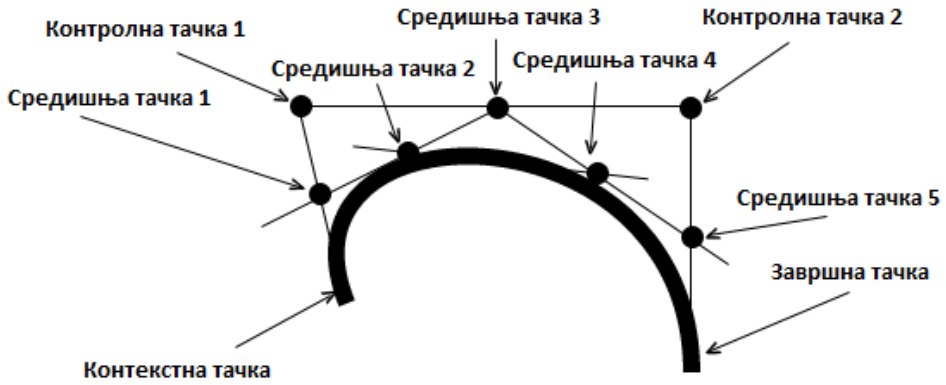
Изглед примера 8 у веб претраживачу приказан је на слици 15.



Слика 15: *Цртање квадратне криве*

5.2 Цртање Безиерове криве

За цртање Безиерове криве помоћу елемента *HTML5 Canvas* користи се метода *bezierCurveTo()*. Безиерове криве су дефинисане контекстном тачком, двема контролним тачкама и завршном тачком. За разлику од квадратних крива, Безиерове криве су дефинисане помоћу две контролне тачке уместо једне, што нам омогућава стварање сложенијих кривих. Видети слику 16.



Слика 16: *Цртање Безиерове криве*

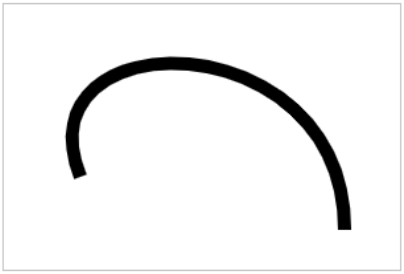
Први део криве је тангентан на замишљену линију која је дефинисана контекстном тачком и првом контролном тачком. Други део криве је тангентан на замишљену линију која је дефинисана другом контролном тачком и завршном тачком. Пример 9 илуструје исцртавање Безиерове криве дебљине 10px, црне боје. Координате контекстне тачке су (188, 130), координате контролних тачака су (140, 10) и (388, 10), док су координате завршне тачке (388, 170).

Пример 9: Цртање Безиерове криве

```
<canvas id="myCanvas" width="578" height="200"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
```

```
    context.beginPath();
    context.moveTo(188, 130);
    context.bezierCurveTo(140, 10, 388, 10, 388, 170);
    context.lineWidth = 10;
    context.strokeStyle = 'black';
    context.stroke();
  </script>
```

Изглед примера 9 у веб претраживачу приказан је на слици 17.



Слика 17: *Цртање Безиерове криве*

6 Цртање путање

Креирање путање помоћу елемента *HTML5 Canvas* се постиже надовезивањем више линија. Завршна тачка сваке нове линије постаје нова контекст тачка. То се постиже позивом познатих метода *lineTo()*, *arcTo()*, *quadraticCurveTo()* и *bezierCurveTo()* за креирање одређеног типа линије које надовезивањем дају крајњи резултат као на слици 18. Такође се користи метода *beginPath()* да би се започело цртање нове путање.

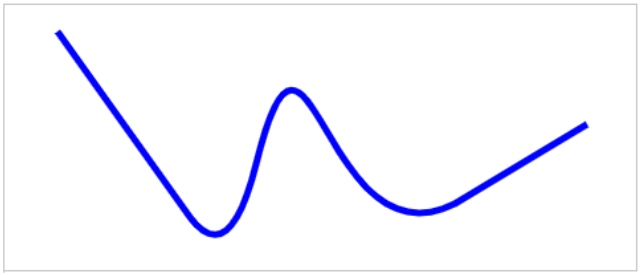
Слика 18: *Цртање путање*

Пример 10 илуструје исцртавање путање која се креира исцртавањем линије са почетком у тачки (100, 20) и крајем у тачки (200, 260) и надовезивањем квадратне и Безиерове криве са крајњом тачком у (500, 90).

Пример 10: Цртање путање

```
<canvas id="myCanvas" width="578" height="200"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  context.beginPath();
  context.moveTo(100, 20);
  context.lineTo(200, 160);
  context.quadraticCurveTo(230, 200, 250, 120);
  context.bezierCurveTo(290, -40, 300, 200, 400, 150);
  context.lineTo(500, 90);
  context.lineWidth = 5;
  context.strokeStyle = 'blue';
  context.stroke();
</script>
```

Изглед примера 10 у веб претраживачу приказан је на слици 19.



Слика 19: *Цртање путање*

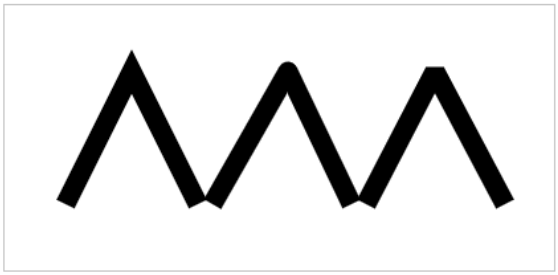
6.1 Начин надовезивања линија

За одређивања начина надовезивања линија користи се својство *lineJoin*. Постоје три начина на које се могу надовезивати линије: оштар, заобљен и раван тип. Видети слику 20. Пример 11 илуструје надовезивање линија: оштар, заобљен и раван тип.

Пример 11: Начин надовезивања линија

```
<canvas id="myCanvas" width="578" height="200"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  context.lineWidth = 15;
```

```
    context.beginPath();
    context.moveTo(99, 150);
    context.lineTo(149, 50);
    context.lineTo(199, 150);
    context.lineJoin = 'miter';
    context.stroke();

    context.beginPath();
    context.moveTo(210, 150);
    context.lineTo(267, 50);
    context.lineTo(315, 150);
    context.lineJoin = 'round';
    context.stroke();

    context.beginPath();
    context.moveTo(326, 150);
    context.lineTo(378, 50);
    context.lineTo(431, 150);
    context.lineJoin = 'bevel';
    context.stroke();
  </script>
```



Слика 20: *Начин надовезивања линија*

6.2 Закривљење путање

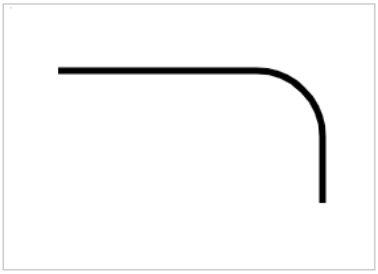
Постизање ефекта закривљења путање се постиже позивом методе *arcTo()* која је дефинисана контролном тачком, завршном тачком и полупречником. У примеру 12 приказано је исцртавање путање која се креира надовезивањем линија и лука дебљине 5px.

Пример 12: Закривљење путање

```
  <canvas id="myCanvas" width="578" height="200"></canvas>
  <script>
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    var rectWidth = 200;
    var rectHeight = 100;
```

```
  var rectX = 189;
  var rectY = 50;
  var cornerRadius = 50;
  context.beginPath();
  context.moveTo(rectX, rectY);
  context.lineTo(rectX + rectWidth - cornerRadius, rectY);
  context.arcTo(rectX + rectWidth, rectY, rectX + rectWidth, rectY +
      cornerRadius, cornerRadius);
  context.lineTo(rectX + rectWidth, rectY + rectHeight);
  context.lineWidth = 5;
  context.stroke();
</script>
```

Изглед примера 12 у веб претраживачу приказан је на слици 21.



Слика 21: *Закривљење путање*

7 Цртање облика

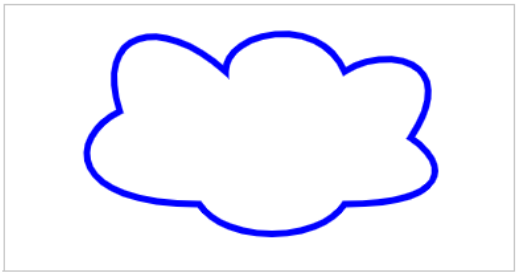
Цртање различитих типова облика помоћу елемента *HTML5 Canvas* се постиже цртањем путања и позивом методе *closePath()*. Применом већ познатих метода *lineTo()*, *arcTo()*,*quadraticCurveTo()* или *bezierCurveTo()* се исрцтава одређени тип линије који надовезивањем и затварањем дају одређени облик. Пример 13 илуструје исрцтавање неправилног облика надовезивањем више Безиерових кривих.

Пример 13: Цртање неправилног облика

```
<canvas id="myCanvas" width="578" height="200"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');

  context.beginPath();
  context.moveTo(170, 80);
  context.bezierCurveTo(130, 100, 130, 150, 230, 150);
  context.bezierCurveTo(250, 180, 320, 180, 340, 150);
  context.bezierCurveTo(420, 150, 420, 120, 390, 100);
  context.bezierCurveTo(430, 40, 370, 30, 340, 50);
```

```
    context.bezierCurveTo(320, 5, 250, 20, 250, 50);
    context.bezierCurveTo(200, 5, 150, 20, 170, 80);

    context.closePath();
    context.lineWidth = 5;
    context.strokeStyle = 'blue';
    context.stroke();
  </script>
```

Изглед примера 13 у веб претраживачу приказан је на слици 22.



Слика 22: *Цртање неправилног облика*

7.1 Цртање правоугаоника

Цртање правоугаоника помоћу елемента *HTML5 Canvas* се постиже позивом методе *rect(x, y, width, height)*. Позиција правоугаоника се одређује горњим левим углом задавањем вредности координата x и y, а димензије правоугаоника путем *width* и *height* својстава. Пример 14 илуструје исцртавање правоугаоника чији је горњи леви угао задат у тачки (40, 40), димензије правоугаоника су 200px и 80px, унутрашњост је попуњена жутом бојом, док је дебљина ивица 7px, црне боје.

Пример 14: Цртање правоугаоника

```
  <canvas id="myCanvas" width="250" height="130"></canvas>
  <script>
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    context.beginPath();
    context.rect(40, 40, 200, 80);
    context.fillStyle = 'yellow';
    context.fill();
    context.lineWidth = 7;
    context.strokeStyle = 'black';
    context.stroke();
  </script>
```

Изглед примера 14 у веб претраживачу приказан је на слици 23.

Слика 23: *Цртање правоугаоника*

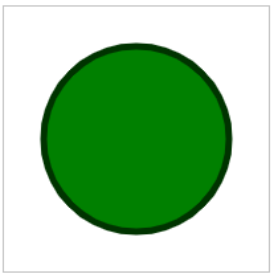
7.2 Цртање круга

Цртање круга помоћу елемента *HTML5 Canvas* се постиже позивом методе *arc()*, постављањем нуле за почетни угао и $2PI$ за крајњи угао. Пример 15 илуструје исцртавање круга у смеру казаљке на сату, са центром у пресеку дијагонала *Canvas context* правоугаоника, полупречника 70px.

Пример 15: Цртање круга

```
<canvas id="myCanvas" width="578" height="200"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  var centerX = canvas.width / 2;
  var centerY = canvas.height / 2;
  var radius = 70;
  context.beginPath();
  context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
  context.fillStyle = 'green';
  context.fill();
  context.lineWidth = 5;
  context.strokeStyle = '#003300';
  context.stroke();
</script>
```

Изглед примера 15 у веб претраживачу приказан је на слици 24.



Слика 24: *Цртање круга*

7.3 Цртање полукруга

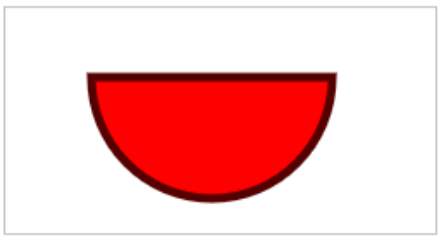
Цртање полукруга помоћу елемента *HTML5 Canvas* се постиже позивом методе *arc()* постављањем вредности (почетни угао + PI) за крајњи угао.

Пример 16 илуструје исцртавање полукруга у смеру казаљке на сату, са центром у тачки (120, 40), полупречника 70px.

Пример 16: Цртање полукруга

```
<canvas id="myCanvas" width="250" height="130"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  context.beginPath();
  context.arc(120, 40, 70, 0, Math.PI, false);
  context.closePath();
  context.lineWidth = 5;
  context.fillStyle = 'red';
  context.fill();
  context.strokeStyle = '#550000';
  context.stroke();
</script>
```

Изглед примера 16 у веб претраживачу приказан је на слици 25.



Слика 25: *Цртање полукруга*

8 Додавање слике

Додавање слике помоћу елемента *HTML5 Canvas* се постиже позивом методе *drawImage()* којој се прослеђује објекат слике и одредишна тачка. Одредишна тачка дефинише горњи леви угао слике у односу на горњи леви угао *Canvas* правоугаоника. Обзиром да метода *drawImage()* захтева објекат слике, неопходно је најпре креирати слику и сачекати њено учитавање пре него што се позове метода *drawImage()*. То се може постићи позивом својств *onload* за објекат слике.

Пример 17 илуструје додавање слике чији је горњи леви угао у односу на горњи леви угао *Canvas* правоугаоника у тачки (69, 50).

Пример 17: Додавање слике

```
<canvas id="myCanvas" width="578" height="400"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  var imageObj = new Image();
  imageObj.onload = function() {
  context.drawImage(imageObj, 69, 50); };
  imageObj.src = 'http://www.html5canvastutorials.com/demos/assets/
      darth-vader.jpg';
</script>
```

Изглед примера 17 у веб претраживачу приказан је на слици 26.

Слика 26: *Додавање слике*

8.1 Постављање димензије слике

За постављање димензија слике потребно је проследити два аргумента методи *drawImage()*: *width* за постављање дужине и *heigth* за постављање висине.

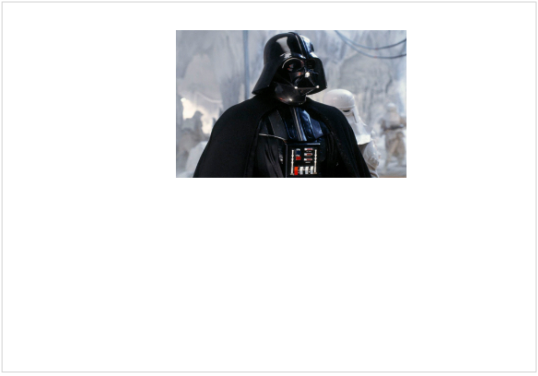
Пример 18 илуструје додавање слике чији је горњи леви угао у односу на горњи леви угао *Canvas* правоугаоника у тачки (188, 30), димензија 250px и 160px.

Пример 18: Постављање димензије слике

```
<canvas id="myCanvas" width="578" height="400"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  var x = 188;
  var y = 30;
  var width = 250;
```

```
    var height = 160;
    var imageObj = new Image();

    imageObj.onload = function() {
      context.drawImage(imageObj, x, y, width, height);
    };
    imageObj.src = 'http://www.html5canvastutorials.com/demos/assets/
        darth-vader.jpg';
  </script>
```

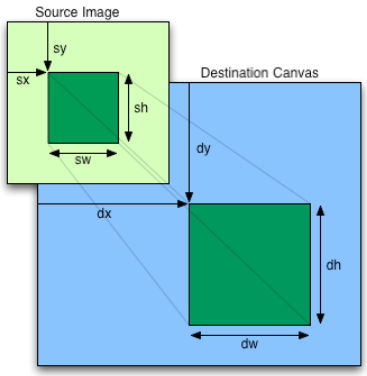
Изглед примера 18 у веб претраживачу приказан је на слици 27.



Слика 27: *Постављање димензије слике*

8.2 Исецање слике

За исецање слике користећи елемент *HTML5 Canvas* неопходно је доделити вредност следећим аргументима методе *drawImage()*: *sourceX*, *sourceY*, *sourceWidth*, *sourceHeight*, *destWidth*, *destHeight*. Ови аргументи дефинишу локацију и димензију правоугаоника који се користи приликом исецања слике. Видети слику 28.



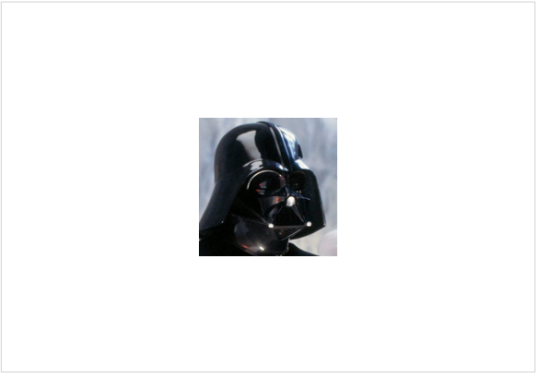
Слика 28: *Дефиниција методе за исецање слике*

Пример 19 илуструје додавање слике која се налази у пресеку дијагонала *Canvas* правоугаоника. Својство *sourceX* има вредност 150px, а својство *sourceY* 0px, што значи да се исецање слике врши на дужини од 150px по x оси, а на висини од 0px по y оси. Димензије слике након исецања су 150px које се задају путем својстава *sourceWidth* и *sourceHeight*.

Пример 19: Исецање слике

```
<canvas id="myCanvas" width="578" height="400"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  var imageObj = new Image();

   imageObj.onload = function() {

   var sourceX = 150;
   var sourceY = 0;
   var sourceWidth = 150;
   var sourceHeight = 150;
   var destWidth = sourceWidth;
   var destHeight = sourceHeight;
   var destX = canvas.width / 2 - destWidth / 2;
   var destY = canvas.height / 2 - destHeight / 2;

   context.drawImage(imageObj, sourceX, sourceY, sourceWidth,
       sourceHeight, destX, destY, destWidth,
    destHeight);
  };
   imageObj.src = 'http://www.html5canvastutorials.com/demos/assets/
      darth-vader.jpg';
</script>
```

Изглед примера 19 у веб претраживачу приказан је на слици 29.



Слика 29: *Исецање слике*

9 Трансформације

Трансформација је начин модификовања *Canvas context* објекта који се манифестује на садржај *Canvas context* објекта заједно са начином дефиниције сопствене трансформације слободне форме. У наставку рада су обрађени следећи типови трансформације:

- транслација,
- ротација,
- скалирање,
- прилагођене трансформације.

9.1 Транслација

Транслација *HTML5 Canvas context* објекта се постиже позивом методе *translate()*. Транслацијом се постиже кретање унутрашњих елемената *Canvas context-a* позивом само једне методе што је јако згодно у случају сложених објеката где би било неопходно мењати вредности x и y координате за сваку тачку посебну. На овај начин се врши трансформација *context* објеката, потом исцртавање графичких објеката.

У примеру 20 се транслира *context* објекат за половину дужине и половину висине *context* елемента, затим црта правоугаоник дужине 100px и висине 50px.

Пример 20: Пример транслације

```
<canvas id="myCanvas" width="200" height="100"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  var rectWidth = 100;
  var rectHeight = 50;
  context.translate(canvas.width / 2, canvas.height / 2);
  context.fillStyle = 'blue';
  context.fillRect(rectWidth / -2, rectHeight / -2, rectWidth,
     rectHeight);
</script>
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_transformacije_translacija`

9.2 Ротација

Ротација помоћу елемента *HTML5 Canvas* се постиже позивом методе *rotate()* која се задаје углом у радијанима. Да би се дефинисала тачка ротације мора се најпре транслирати *context* објекат тако да се горњи леви угао *context* објекта доведе до жељене тачке ротације.

У примеру 21 се најпре транслира *context* објекат до центра правоугаоника, затим заротира правоугаоник око његовог центра за угао од $PI/4$.

Пример 21: Пример ротације

```
<canvas id="myCanvas" width="200" height="150"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  var rectWidth = 100;
  var rectHeight = 50;
  context.translate(canvas.width / 2, canvas.height / 2);
  context.rotate(Math.PI / 4);
  context.fillStyle = 'blue';
  context.fillRect(rectWidth / -2, rectHeight / -2, rectWidth,
     rectHeight);
</script>
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_transformacije_rotacija`

9.3 Скалирање

Скалирање помоћу елемента *HTML5 Canvas* се постиже позивом методе *scale()* која мења вредности димензија *context* објекта.

У примеру 22 се транслира *context* објекат за половину дужине и половину висине елемента *Canvas*, затим се постави вредност y координате на 0.5. Овим се добија висина правоугаоника за 0.5 мања од почетне вредности.

Пример 22: Пример скалирања

```
<canvas id="myCanvas" width="200" height="100"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  var rectWidth = 150;
  var rectHeight = 80;
  context.translate(canvas.width / 2, canvas.height / 2);
  context.scale(1, 0.5);
  context.fillStyle = 'blue';
  context.fillRect(rectWidth / -2, rectHeight / -2, rectWidth,
     rectHeight);
</script>
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_transformacije_skaliranje`

9.4 Прилагођене трансформације

Прилагођене трансформације помоћу елемента *HTML5 Canvas* постижу се позивом методе *transform()*. Ова метода прихвата 6 аргумента 3 x 3 матрицу према следећој конвенцији:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

У примеру 23 се транслира правоугаоник за половину дужине и половину висине елемента *Canvas* путем методе *transform()*.

Пример 23: Пример трансоффрмације

```
<canvas id="myCanvas" width="250" height="150"></canvas>
<script>
  var context = canvas.getContext('2d');
  var rectWidth = 150;
  var rectHeight = 75;

  // 1 0 tx
  // 0 1 ty
  // 0 0 1

  var tx = canvas.width / 2;
  var ty = canvas.height / 2;
  context.transform(1, 0, 0, 1, tx, ty);
  context.fillStyle = 'blue';
  context.fillRect(rectWidth / -2, rectHeight / -2, rectWidth,
      rectHeight);
</script>
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_prilagodjene_transformacije`

9.5 Ресетовање трансформације

Ресетовање трансформације помоћу елемента *HTML5 Canvas* постиже се позивом методе *setTransform()* користећи следећу матрицу трансформације:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

У примеру 24 се ресетује транслиран правоугаоник за половину дужине и половину висине *Canvas* елемента.

Пример 24: Пример ресетовања трансофрмације

```
<canvas id="myCanvas" width="450" height="200"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  var rectWidth = 150;
  var rectHeight = 75;
  context.translate(canvas.width / 2, canvas.height / 2);
  context.fillStyle = 'blue';
  context.fillRect(rectWidth / -2, rectHeight / -2, rectWidth,
      rectHeight);

  // 1 0 0
  // 0 1 0
  // 0 0 1

  context.setTransform(1, 0, 0, 1, 0, 0);
  context.fillStyle = 'red';
  context.fillRect(0, 0, rectWidth, rectHeight);
</script>
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_resetovanje_transformacije`

10 Креирање композиција

На веб страници позадина је најнижи слој, затим следе *HTML* елементи позиционирани испод елемента *Canvas* и *CSS* позадина примењена на елемент *Canvas*. Различити типови композиције у *HTML5 Canvas-y* омогућују додавање ефеката попут сенке, прозирности и градијента било ком објекту *Canvas context* објекта.

У наставку рада су обрађени следећи типови композиције:

- додавање сенке,
- прозирност,
- линеарни и радијални градијент.

10.1 Додавање сенке

Сваком исцртаном објекту у оквиру *Canvas* површине за цртање може се додати сенка, чиме се ствара ефекат треће димензије. Додавање сенке помоћу елемента *HTML5 Canvas* се постиже постављањем следећих својстава:

- shadowColor - боја сенке,
- shadowBlur - представља величину замагљености сенке. Нижа вредност представља оштрију сенку,
- shadowOffsetX - представља помак сенке по x оси у односу на горњи леви угао објекта,
- shadowOffsetY - представља помак сенке по y оси у односу на горњи леви угао објекта.

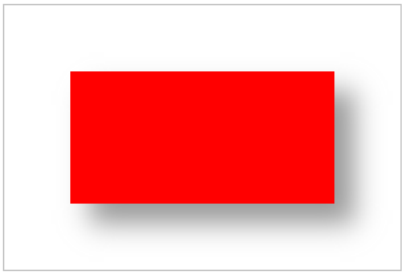
Својства *shadowOffsetX* и *shadowOffsetY* могу имати и негативне вредности, док својство *shadowBlur* само позитивне вредности.

У примеру 25 се поставља сенка на правоугаоник црвене боје.

Пример 25: Пример додавање сенке

```
<canvas id="myCanvas" width="300" height="200"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  context.rect(50, 50, 200, 100);
  context.fillStyle = 'red';
  context.shadowColor = '#999';
  context.shadowBlur = 20;
  context.shadowOffsetX = 15;
  context.shadowOffsetY = 15;
  context.fill();
</script>
```

Изглед примера 25 у веб претраживачу приказан је на слици 30.



Слика 30: *Пример додавање сенке*

10.2 Подешавање прозирности

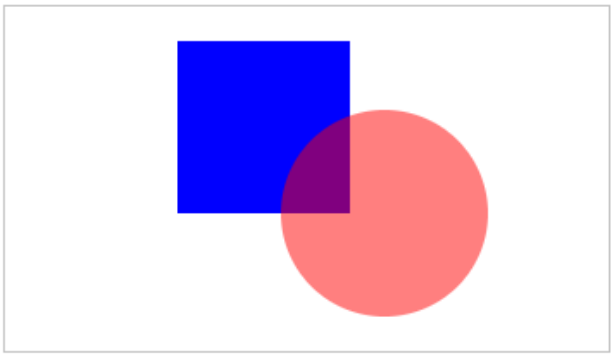
Прозирност се у *HTML5 Canvas-y* подешава постављањем својства *globalAlpha* на реалне вредности између 0 и 1, при чему 0 представља потпуну прозирност, а 1 потпуну непрозирност.

Пример 26 илуструје прозирност круга у односу на правоугаоник за вредност 0.5.

Пример 26: Пример прозирности

```
<canvas id="myCanvas" width="350" height="200"></canvas>
<script>
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');

  context.beginPath();
  context.rect(100, 20, 100, 100);
  context.fillStyle = 'blue';
  context.fill();

  context.globalAlpha = 0.5;
  context.beginPath();
  context.arc(220, 120, 60, 0, 2 * Math.PI, false);
  context.fillStyle = 'red';
  context.fill();
</script>
```

Изглед примера 26 у веб претраживачу приказан је на слици 31.



Слика 31: *Пример прозирности*

10.3 Градијент

Градијент се користи за попуњавање унутрашњости правоугаоника, кругова, линија, текстуалних садржаја и слично. Постоје два типа градијента:
createLinearGradient(x,y,x1,y1) - креира линеарни градијент, и
createRadialGradient(x,y,r,x1,y1,r1) - креира радијални/кружни градијент.

Једном градијенту се могу додати две или више боја. Метода *addColorStop()* одређује боју којом се попуњава неки објекат и позицију те боје дуж градијента. Та позиција се креће у опсегу између 0 и 1. Да би се применио градијент, неопходно је дефинисати вредност својства *fillStyle* или *strokeStyle* на дефинисани градијент. Након тога се цртају објекти.

Пример 27: Пример примене линеарног градијента над правоугаоником

```
<canvas id="myCanvasL" width="200" height="100" style="border:1px solid
    #c3c3c3;"></canvas>
   <script>
    var c = document.getElementById("myCanvasL");
    var ctx = c.getContext("2d");
    var grd = ctx.createLinearGradient(0, 0, 200, 0);
    grd.addColorStop(0, "red");
    grd.addColorStop(1, "white");
    ctx.fillStyle = grd;
    ctx.fillRect(10, 10, 150, 80);
  </script>
```

Изглед примера 27 у веб претраживачу приказан је на слици 32.

Слика 32: *Пример линеарног градијента*

Пример 28: Пример примене радијалног градијента над правоугаоником

```
<canvas id="myCanvasR" width="200" height="100" style="border:1px solid
    #c3c3c3;"></canvas>
   <script>
    var c = document.getElementById("myCanvasR");
    var ctx = c.getContext("2d");
    var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);
    grd.addColorStop(0, "red");
    grd.addColorStop(1, "white");
    ctx.fillStyle = grd;
    ctx.fillRect(10, 10, 150, 80);
</script>
```

Изглед примера 28 у веб претраживачу приказан је на слици 33.

Слика 33: *Пример радијалног градијента*

11 Анимације

За креирање било ког типа анимације користећи елемент *HTML5 Canvas*, користи се *requestAnimFrame* подлога која омогућава прегледачу да одреди оптимални *FPS* за анимацију. Сваки оквир за анимацију даје могућност ажурирања елемената у *HTML5 Canvas context-y*, брисање истих, прецртавање, а затим креирање другог оквира за анимацију.

Пример 29 представља један вид анимације о којој ће бити више речи у наставку.

Пример 29: Пример анимације

```
<canvas id="myCanvas" width="578" height="200"></canvas>

    <script>
      window.requestAnimFrame = (function(callback) {
        return window.requestAnimationFrame || window.
            webkitRequestAnimationFrame || window.mozRequestAnimationFrame
            || window.oRequestAnimationFrame || window.
            msRequestAnimationFrame ||
        function(callback) {
          window.setTimeout(callback, 1000 / 60);
        };
      })();
      function drawRectangle(myRectangle, context) {
        context.beginPath();
        context.rect(myRectangle.x, myRectangle.y, myRectangle.width,
            myRectangle.height);
        context.fillStyle = '#8ED6FF';
        context.fill();
        context.lineWidth = myRectangle.borderWidth;
        context.strokeStyle = 'black';
        context.stroke();
      }
      function animate(myRectangle, canvas, context, startTime) {
        var time = (new Date()).getTime() - startTime;

        var linearSpeed = 100;

        var newX = linearSpeed * time / 1000;

        if(newX < canvas.width - myRectangle.width - myRectangle.
            borderWidth / 2) {
          myRectangle.x = newX;
        }

        context.clearRect(0, 0, canvas.width, canvas.height);

        drawRectangle(myRectangle, context);
```

```
      requestAnimFrame(function() {
        animate(myRectangle, canvas, context, startTime);
      });
    }
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');

    var myRectangle = {
      x: 0,
      y: 75,
      width: 100,
      height: 50,
      borderWidth: 5
    };
    drawRectangle(myRectangle, context);

    setTimeout(function() {
      var startTime = (new Date()).getTime();
      animate(myRectangle, canvas, context, startTime);
    }, 1000);
</script>
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_linearno_kretanje`

У наставку рада су обрађени следећи типови анимација:

- брисање садржаја,
- линеарно кретање,
- анимација са убрзањем,
- анимација са осциловањем,
- анимација са реакцијом корисника.

11.1 Брисање садржаја

Да би се испразнио садржај *Canvas context* објекта користи се метода *clearRect()*. Ова техника брисања је боља од других техника брисања, као на пример ресетовање ширине и висине *Canvas* објекта или поништавање *Canvas* елемента, па поновно креирање истог. Пример 30 илуструје брисање правоугаоника кликом на дугме 'Clear'.

Пример 30: Пример брисања елемената *Canvas context-a*

```
    <canvas id="pravougaonik" width="200" height="100" style="border:1px
       solid #c3c3c3;"></canvas>
    <div id="buttons">
    <input type="button" id="clear" value="Clear">
     </div>
```

```
    <script>
    var canvas = document.getElementById("pravougaonik");
    var ctx = canvas.getContext("2d");
    ctx.fillStyle="#FF0000";
    ctx.fillRect(0, 0, 200, 100);
    document.getElementById('clear').addEventListener('click', function()
        {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
   }, false);
</script>
```

Изглед примера 30 у веб претраживачу приказан је на слици 34.

Слика 34: *Пример брисања елемената* Canvas context *објекта*

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_brisanje`

11.2 Линеарно кретање

Да би се остварило кретање помоћу елемента *HTML5 Canvas* увећамо x или y координату или обе за неки објекат према наредној једначини:

$$пут = брзина * време$$

Пример 31 симулира кретање правоугаоника од леве ивице *Canvas context* објекта ка десној ивици, брзином од 100 пиксела у секунди. Докле год је пређени пут мањи од ширине *Canvas context* објекта правоугаонику се постављају координате темена сваких 1000 милисекунди све док је лева ивица правоугаоника у оквиру *Canvas context* објекта. Затим се врши брисање правоугаоника и понавља иста акција позивом *requestAnimFrame*.

Пример 31: Пример линеарног кретања

```
    <canvas id="myCanvas" width="350" height="200" style="border:1px solid
        #c3c3c3;">
    </canvas>
    <script>
   window.requestAnimFrame = (function(callback) {
      return window.requestAnimationFrame || window.
          webkitRequestAnimationFrame || window.mozRequestAnimationFrame
          || window.oRequestAnimationFrame || window.
          msRequestAnimationFrame ||
```

```
    function(callback) {
      window.setTimeout(callback, 1000 / 60);
    };
  })();
  function drawRectangle(myRectangle, context) {
    context.beginPath();
    context.rect(myRectangle.x, myRectangle.y, myRectangle.width,
        myRectangle.height);
    context.fillStyle = '#8ED6FF';
    context.fill();
    context.lineWidth = myRectangle.borderWidth;
    context.strokeStyle = 'black';
    context.stroke();
  }
  function animate(myRectangle, canvas, context, startTime) {
    var time = (new Date()).getTime() - startTime;
    var linearSpeed = 100;
    var newX = linearSpeed * time / 1000;

    if(newX < canvas.width - myRectangle.width - myRectangle.
        borderWidth / 2) {
      myRectangle.x = newX;
    }
    context.clearRect(0, 0, canvas.width, canvas.height);
    drawRectangle(myRectangle, context);
    requestAnimFrame(function() {
      animate(myRectangle, canvas, context, startTime);
    });
  }
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');

  var myRectangle = {
    x: 0,
    y: 75,
    width: 100,
    height: 50,
    borderWidth: 5
  };
  drawRectangle(myRectangle, context);
  setTimeout(function() {
    var startTime = (new Date()).getTime();
    animate(myRectangle, canvas, context, startTime);
  }, 1000);
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_linearno_kretanje`

11.3 Анимација са убрзањем

Креирање квадратне анимације помоћу елемента *HTML5 Canvas* се постиже увећањем v_x (хоризонтална брзина), v_y (вертикална брзина) или и v_x и v_y објекта за сваки оквир, а затим ажурирањем положаја објекта, према наредној једначини:

$$растојање = брзина * време + \frac{1}{2} * убрзање * \text{време}^{\,2}$$

Пример 32 илуструје кретање правоугаоника од врха *Canvas context* до дна *Canvas context* тако што се на сваких 1000 милисекунди поставља ширина правоугаоника према наведеној формули.

Пример 32: Пример анимације са убрзањем

```
<canvas id="myCanvas" width="578" height="200"></canvas>
<script>
  window.requestAnimFrame = (function(callback) {
    return window.requestAnimationFrame || window.
        webkitRequestAnimationFrame || window.mozRequestAnimationFrame
        || window.oRequestAnimationFrame || window.
        msRequestAnimationFrame ||
    function(callback) {
      window.setTimeout(callback, 1000 / 60);
    };
  })();
  function drawRectangle(myRectangle, context) {
    context.beginPath();
    context.rect(myRectangle.x, myRectangle.y, myRectangle.width,
        myRectangle.height);
    context.fillStyle = '#8ED6FF';
    context.fill();
    context.lineWidth = myRectangle.borderWidth;
    context.strokeStyle = 'black';
    context.stroke();
  }
  function animate(myRectangle, canvas, context, startTime) {
    var time = (new Date()).getTime() - startTime;
    var gravity = 200;
    myRectangle.y = 0.5 * gravity * Math.pow(time / 1000, 2);

    if(myRectangle.y > canvas.height - myRectangle.height - myRectangle
        .borderWidth / 2) {
      myRectangle.y = canvas.height - myRectangle.height - myRectangle.
          borderWidth / 2;
    }
    lastTime = time;
    context.clearRect(0, 0, canvas.width, canvas.height);
    drawRectangle(myRectangle, context);
    requestAnimFrame(function() {
```

```
      animate(myRectangle, canvas, context, startTime);
    });
  }
  var canvas = document.getElementById('myCanvas');
  var context = canvas.getContext('2d');
  var myRectangle = {
    x: 239,
    y: 0,
    width: 100,
    height: 50,
    borderWidth: 5
  };
  drawRectangle(myRectangle, context);
  setTimeout(function() {
    var startTime = (new Date()).getTime();
    animate(myRectangle, canvas, context, startTime);
  }, 2000);

</script>
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_animacija_ubrzanje`

11.4 Анимација са осциловањем

Креирање анимације са осциловањем помоћу елемента *HTML5 Canvas* се постиже једначином за једноставан хармонички осцилатор за постављање положаја облика за сваки оквир:

$$x(t) = amplitude * sin(t * 2PI/period) + x_0$$

Пример 33: Пример анимације са убрзањем

```
<canvas id="myCanvas" width="578" height="200"></canvas>
  <script>
  window.requestAnimFrame = (function(callback) {
    return window.requestAnimationFrame || window.
        webkitRequestAnimationFrame || window.mozRequestAnimationFrame
        || window.oRequestAnimationFrame || window.
        msRequestAnimationFrame ||
    function(callback) {
      window.setTimeout(callback, 1000 / 60);
    };
  })();
  function drawRectangle(myRectangle, context) {
    context.beginPath();
    context.rect(myRectangle.x, myRectangle.y, myRectangle.width,
        myRectangle.height);
    context.fillStyle = '#8ED6FF';
```

```
      context.fill();
      context.lineWidth = myRectangle.borderWidth;
      context.strokeStyle = 'black';
      context.stroke();
    }
    function animate(myRectangle, canvas, context, startTime) {
      var time = (new Date()).getTime() - startTime;
      var amplitude = 150;
      var period = 2000;
      var centerX = canvas.width / 2 - myRectangle.width / 2;
      var nextX = amplitude * Math.sin(time * 2 * Math.PI / period) +
          centerX;
      myRectangle.x = nextX;
      context.clearRect(0, 0, canvas.width, canvas.height);
      drawRectangle(myRectangle, context);
      requestAnimFrame(function() {
        animate(myRectangle, canvas, context, startTime);
      });
    }
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    var myRectangle = {
      x: 250,
      y: 70,
      width: 100,
      height: 50,
      borderWidth: 5
    };
    drawRectangle(myRectangle, context);
    setTimeout(function() {
      var startTime = (new Date()).getTime();
      animate(myRectangle, canvas, context, startTime);
    }, 1000);
  </script>
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_animacija_oscilovanje`

11.5 Анимација са реакцијом корисника

У примеру 34 се кретање и заустављање објекта активира кликом на *Canvas context* правоугаоник.

Пример 34: Пример анимације са реакцијом корисника

```
  <canvas id="myCanvas" width="578" height="200"></canvas>
    <script>
    window.requestAnimFrame = (function(callback) {
      return window.requestAnimationFrame || window.
          webkitRequestAnimationFrame || window.mozRequestAnimationFrame
```

```
      || window.oRequestAnimationFrame || window.
      msRequestAnimationFrame ||
  function(callback) {
    window.setTimeout(callback, 1000 / 60);
  };
})();
function drawRect(myRectangle, context) {
  context.beginPath();
  context.rect(myRectangle.x, myRectangle.y, myRectangle.width,
     myRectangle.height);
  context.fillStyle = '#8ED6FF';
  context.fill();
  context.lineWidth = myRectangle.borderWidth;
  context.strokeStyle = 'black';
  context.stroke();
}
function animate(lastTime, myRectangle, runAnimation, canvas, context
    ) {
  if(runAnimation.value) {
    var time = (new Date()).getTime();
    var timeDiff = time - lastTime;
    var linearSpeed = 100;
    var linearDistEachFrame = linearSpeed * timeDiff / 1000;
    var currentX = myRectangle.x;
    if(currentX < canvas.width - myRectangle.width - myRectangle.
       borderWidth / 2) {
      var newX = currentX + linearDistEachFrame;
      myRectangle.x = newX;
    }
    context.clearRect(0, 0, canvas.width, canvas.height);
    drawRect(myRectangle, context);
    requestAnimFrame(function() {
      animate(time, myRectangle, runAnimation, canvas, context);
    });
  }
}
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');
var myRectangle = {
  x: 0,
  y: 75,
  width: 100,
  height: 50,
  borderWidth: 5
};
var runAnimation = {
  value: false
};
document.getElementById('myCanvas').addEventListener('click', function
```

```
        () {
       // flip flag
       runAnimation.value = !runAnimation.value;

       if(runAnimation.value) {
         var date = new Date();
         var time = date.getTime();
         animate(time, myRectangle, runAnimation, canvas, context);
       }
     });
     drawRect(myRectangle, context);
   </script>
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_animacija_na_klik`

12 Креирање аналогног сата

У наставку рада симулира се рад аналогног сата [6] помоћу претходног обрађених лекција.
Да би се симулирао рад аналогног сата неопходно је омогућити:

- додавање оквира,
- додавање бројчаника,
- додавање бројева,
- додавање казаљки.

12.1 Додавање оквира

У примеру 35 приказано је креирање оквира за аналогни сат.

Пример 35: Аналогни сат

```
<canvas id="canvas" width="200" height="200" style="background-color
    :#333">
</canvas>

<script>
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");
  var radius = canvas.height / 2;
  ctx.translate(radius, radius);
  radius = radius * 0.90
  drawClock();

function drawClock() {
  ctx.arc(0, 0, radius, 0 , 2 * Math.PI);
```

```
    ctx.fillStyle = "white";
    ctx.fill();
  }
 </script>
```

Изглед примера 35 у веб претраживачу приказан је на слици 35.

Слика 35: *Оквир за аналогни сат*

12.2 Додавање бројчаника

У примеру 36 приказано је креирање бројчаника за аналогни сат.

Пример 36: Цртање бројчаника

```
<canvas id="canvas" width="200" height="200" style="background-color:#333
    ">
</canvas>

<script>
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");
  var radius = canvas.height / 2;
  ctx.translate(radius, radius);
  radius = radius * 0.90
  drawClock();

function drawClock() {
  drawFace(ctx, radius);
}

function drawFace(ctx, radius) {
  var grad;
  ctx.beginPath();
  ctx.arc(0, 0, radius, 0, 2*Math.PI);
  ctx.fillStyle = 'white';
  ctx.fill();
```

```
  grad = ctx.createRadialGradient(0,0,radius*0.95, 0,0,radius*1.05);

  grad.addColorStop(0, '#333');
  grad.addColorStop(0.5, 'white');
  grad.addColorStop(1, '#333');

  ctx.strokeStyle = grad;
  ctx.lineWidth = radius*0.1;
  ctx.stroke();
  ctx.beginPath();
  ctx.arc(0, 0, radius*0.1, 0, 2*Math.PI);
  ctx.fillStyle = '#333';
  ctx.fill();
}
</script>
}
```

Изглед примера 36 у веб претраживачу приказан је на слици 36.

Слика 36: *Цртање бројчаника*

12.3 Додавање бројева

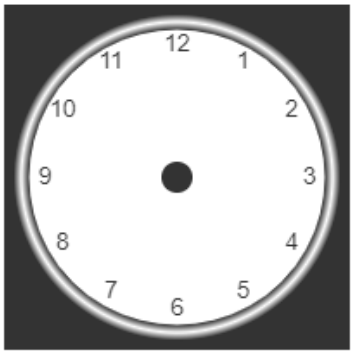
У примеру 37 приказано је цртање бројева за аналогни сат.

Пример 37: Цртање бројева

```
<canvas id="canvas" width="200" height="200" style="background-color:#333
    ">
</canvas>

<script>
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");
  var radius = canvas.height / 2;
  ctx.translate(radius, radius);
  radius = radius * 0.90
  drawClock();
```

```
function drawClock() {
  drawFace(ctx, radius);
  drawNumbers(ctx, radius);
}

function drawFace(ctx, radius) {
  var grad;
  ctx.beginPath();
  ctx.arc(0, 0, radius, 0, 2*Math.PI);
  ctx.fillStyle = 'white';
  ctx.fill();
  grad = ctx.createRadialGradient(0,0,radius*0.95, 0,0,radius*1.05);
  grad.addColorStop(0, '#333');
  grad.addColorStop(0.5, 'white');
  grad.addColorStop(1, '#333');
  ctx.strokeStyle = grad;
  ctx.lineWidth = radius*0.1;
  ctx.stroke();
  ctx.beginPath();
  ctx.arc(0, 0, radius*0.1, 0, 2*Math.PI);
  ctx.fillStyle = '#333';
  ctx.fill();
}

function drawNumbers(ctx, radius) {
  var ang;
  var num;
  ctx.font = radius*0.15 + "px arial";
  ctx.textBaseline="middle";
  ctx.textAlign="center";

for(num = 1; num < 13; num++){
  ang = num * Math.PI / 6;
  ctx.rotate(ang);
  ctx.translate(0, -radius*0.85);
  ctx.rotate(-ang);
  ctx.fillText(num.toString(), 0, 0);
  ctx.rotate(ang);
  ctx.translate(0, radius*0.85);
  ctx.rotate(-ang);
  }
}
</script>
```

Изглед примера 37 у веб претраживачу приказан је на слици 37.



Слика 37: *Цртање бројева*

12.4 Додавање казаљки

У примеру 38 приказано је цртање казаљки за аналогни сат.

Пример 38: Цртање казаљки

```
<canvas id="canvas" width="400" height="400"
  style="background-color:#333">
</canvas>

<script>
  var canvas = document.getElementById("myCanvas");
  var ctx = canvas.getContext("2d");
  var radius = canvas.height / 2;
  ctx.translate(radius, radius);
  radius = radius * 0.90
  drawClock();

function drawClock() {
  drawFace(ctx, radius);
  drawNumbers(ctx, radius);
  drawTime(ctx, radius);
}

function drawFace(ctx, radius) {
  var grad;
  ctx.beginPath();
  ctx.arc(0, 0, radius, 0, 2*Math.PI);
  ctx.fillStyle = 'white';
  ctx.fill();
  grad = ctx.createRadialGradient(0,0,radius*0.95, 0,0,radius*1.05);
  grad.addColorStop(0, '#333');
  grad.addColorStop(0.5, 'white');
  grad.addColorStop(1, '#333');
```

```
  ctx.strokeStyle = grad;
  ctx.lineWidth = radius*0.1;
  ctx.stroke();
  ctx.beginPath();
  ctx.arc(0, 0, radius*0.1, 0, 2*Math.PI);
  ctx.fillStyle = '#333';
  ctx.fill();
}

function drawNumbers(ctx, radius) {
  var ang;
  var num;
  ctx.font = radius*0.15 + "px arial";
  ctx.textBaseline="middle";
  ctx.textAlign="center";
 for(num = 1; num < 13; num++){
   ang = num * Math.PI / 6;
   ctx.rotate(ang);
   ctx.translate(0, -radius*0.85);
   ctx.rotate(-ang);
   ctx.fillText(num.toString(), 0, 0);
   ctx.rotate(ang);
   ctx.translate(0, radius*0.85);
   ctx.rotate(-ang);
 }
}

function drawTime(ctx, radius){
  var now = new Date();
  var hour = now.getHours();
  var minute = now.getMinutes();
  var second = now.getSeconds();
  hour=hour%12;
  hour=(hour*Math.PI/6) + (minute*Math.PI/(6*60)) + (second*Math.PI
      /(360*60));
  drawHand(ctx, hour, radius*0.5, radius*0.07);
  minute=(minute*Math.PI/30) + (second*Math.PI/(30*60));
  drawHand(ctx, minute, radius*0.8, radius*0.07);
  second=(second*Math.PI/30);
  drawHand(ctx, second, radius*0.9, radius*0.02);
}

function drawHand(ctx, pos, length, width) {
  ctx.beginPath();
  ctx.lineWidth = width;
  ctx.lineCap = "round";
  ctx.moveTo(0,0);
  ctx.rotate(pos);
  ctx.lineTo(0, -length);
```

```
  ctx.stroke();
  ctx.rotate(-pos);
}
</script>
```

Изглед примера 38 у веб претраживачу приказан је на слици 38.

Слика 38: *Цртање казаљки*

12.5 Стартовање сата са радом

У примеру 39 симулира се рад аналогног сата помоћу претходног обрађених лекција. Интервал је дат у милисекундама, што значи да се позив метод *drawClock* позива на сваких 1000 милисекунди.

Пример 39: Цртање казаљки

```
<canvas id="canvas" width="200" height="200"
 style="background-color:#333">
</canvas>

<script>
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
var radius = canvas.height / 2;
ctx.translate(radius, radius);
radius = radius * 0.90
setInterval(drawClock, 1000);

function drawClock() {
  drawFace(ctx, radius);
  drawNumbers(ctx, radius);
  drawTime(ctx, radius);
}

function drawFace(ctx, radius) {
 var grad;
 ctx.beginPath();
```

```
  ctx.arc(0, 0, radius, 0, 2*Math.PI);
  ctx.fillStyle = 'white';
  ctx.fill();
  grad = ctx.createRadialGradient(0,0,radius*0.95, 0,0,radius*1.05);
  grad.addColorStop(0, '#333');
  grad.addColorStop(0.5, 'white');
  grad.addColorStop(1, '#333');
  ctx.strokeStyle = grad;
  ctx.lineWidth = radius*0.1;
  ctx.stroke();
  ctx.beginPath();
  ctx.arc(0, 0, radius*0.1, 0, 2*Math.PI);
  ctx.fillStyle = '#333';
  ctx.fill();
}

 function drawNumbers(ctx, radius) {
   var ang;
   var num;
   ctx.font = radius*0.15 + "px arial";
   ctx.textBaseline="middle";
   ctx.textAlign="center";
  for(num = 1; num < 13; num++){
    ang = num * Math.PI / 6;
    ctx.rotate(ang);
    ctx.translate(0, -radius*0.85);
    ctx.rotate(-ang);
    ctx.fillText(num.toString(), 0, 0);
    ctx.rotate(ang);
    ctx.translate(0, radius*0.85);
    ctx.rotate(-ang);
  }
}

 function drawTime(ctx, radius){
   var now = new Date();
   var hour = now.getHours();
   var minute = now.getMinutes();
   var second = now.getSeconds();
   hour=hour%12;
   hour=(hour*Math.PI/6)+
   (minute*Math.PI/(6*60))+
   (second*Math.PI/(360*60));
   drawHand(ctx, hour, radius*0.5, radius*0.07);
   minute=(minute*Math.PI/30)+(second*Math.PI/(30*60));
   drawHand(ctx, minute, radius*0.8, radius*0.07);
   second=(second*Math.PI/30);
   drawHand(ctx, second, radius*0.9, radius*0.02);
 }
```

```
function drawHand(ctx, pos, length, width) {
  ctx.beginPath();
  ctx.lineWidth = width;
  ctx.lineCap = "round";
  ctx.moveTo(0,0);
  ctx.rotate(pos);
  ctx.lineTo(0, -length);
  ctx.stroke();
  ctx.rotate(-pos);
}
</script>
```

Упутство: Демонстрација примера се налази на адреси: `http://edusoft.matf.bg.ac.rs/eskola_veba/#/course/canvas/canvas_analogni_sat`

Закључак

Начин учења који подразумева коришћење едукативних садржаја на интернету је све више заступљен, па је самим тим главна идеја овог рада да почетницима пружи материјале у виду електронских лекција. Литература за учење веб технологија је обимна и доступна на интернету, често на страним језицима, најчешће на енглеском. Предност електронских лекција у односу на велики број материјала доступних на интернету је то што су лекције написане на српском језику, што би требало да значајно убрза процес учења свим корисницима.

Све лекције су написане пажљиво и методички са циљем да буду прилагођене почетницима. Свака лекција садржи теоријско објашњење теме коју обрађује, као и примере који илуструју дата објашњења. Представљене су тако да обухвате основне карактеристике *HTML5 Canvas* технологије и пруже могућност за даље усавршавање. Електронске лекције о *HTML5 Canvas* технологији имају за циљ да оспособе читаоца за креирање једноставних анимација и апликација за цртање на веб страницама.

Лекција креирана на крају рада помоћу претходно обрађених лекција приказује колико могућности пружа елемент *HTML5 Canvas* које раније нису биле изводљиве на вебу, што га чини једним од најпопуларнијих елемената *HTML5* језика.

Доступност електронских лекција, које су креиране у оквиру израде овог мастер рада, би требало да олакша рад и учење, као и да подстакне на размишљање и креативност у раду.

13 Литература

[1] Jurić Nemanja, Marić Miroslav, *eŠkola Veba*, Matematički fakultet, Beograd, 2016.

[2] TUTORIALS, HTML, CSS, JavaScript, Canvas, *w3schools.com*, 2006.

[3] Canvas tutorials and resources, `https://www.html5canvastutorials.com/`, Приступано септембра 2021.

[4] TUTORIALS, Canvas, `https://www.w3schools.com/graphics/canvas_drawing.asp`, Приступано септембра 2021.

[5] Canvas документација, `https://html.spec.whatwg.org/multipage/canvas.html`, Приступано септембра 2021.

[6] Canvas analogni sat, `https://www.w3schools.com/graphics/canvas_clock.asp`, Приступано септембра 2021.