# Crtanje teksta i slika

Sve do sada, korišćenjem Canvas API-i, obavljali smo crtanje različitih geometrijskih oblika koje smo samostalno kreirali i to programabilno, tokom izvršavanja aplikacije. Pored proizvoljne grafike, unutar canvas elementa je moguće nacrtati i neku postojeću sliku. Takav pristup je veoma koristan, pogotovu za razvoj igara.

Pored crtanja slika, Canvas API obezbeđuje i nekoliko metoda koje omogućavaju crtanje teksta. Lekcija koja je pred Vama baviće se pristupima za crtanje teksta i slika unutar canvas elementa.

Kako se obavlja crtanje neke slike unutar canvas elementa?

Crtanje neke slike unutar canvas elementa je proces koji se sastoji iz dva koraka:

1. dobijanje reference na sliku, što zapravo predstavlja kreiranje objektne predstave neke realne slike, koja postoji na nekoj putanji, unutar trenutnog HTML dokumenta ili nekog drugog canvas elementa
2. crtanje slike unutar canvas-a uz eventualnu promenu njenih dimenzija, perspektive ili kompozicije

## Kreiranje objektne reprezentacije slike

Prvi korak u kreiranju objektne reprezentacije neke slike jeste kreiranje objekta tipa `HTMLImageElement`, korišćenjem `Image()` konstruktora:

```
let img = new Image();
```

Nakon kreiranja objekta koji će predstavljati sliku, neophodno je definisati putanju na kojoj se slika nalazi:

```
let img = new Image();
img.src = 'enemy.png';
```

Korišćenjem svojstva `src` definiše se putanja na kojoj se slika nalazi. Čim izvršavanje koda dođe do ovakve naredbe, započinje učitavanje slike. S obzirom da učitavanje može potrajati neko vreme, u zavisnosti od veličine slike i brzine internet konekcije, jasno je da je neophodno definisati određenu logiku koja će nam dojaviti kada se učitavanje završi.

## Dobijanje dojave o završetku učitavanja slike

Preduslov za crtanje neke slike unutar canvas elementa jeste kompletan završetak njenog učitavanja. Ukoliko se pokušava obaviti crtanje slike koja još nije učitana, u modernim web pregledačima neće se dogoditi ništa, dok će u starijim doći do podizanja izuzetka.

Kako bismo bili sigurni da je učitavanje slike završeno, moguće je koristiti standardni pristup DOM API-a, koji podrazumeva registrovanje logike koja će se aktivirati prilikom emitovanja `load` događaja:

```
var img = new Image();

img.addEventListener('load', function () {

    // code for drawing

}, false);

img.src = 'enemy.png';
```

Metoda `addEventListener()` pozvana je nad objektom koji predstavlja sliku. Na taj način je obavljeno registrovanje logike koja će se aktivirati kada se slika u potpunosti učita. Možda se pitate zbog čega je poziv metode `addEventListener()` postavljen pre definisanja vrednosti `src` svojstva. Razlog su stariji web pregledači (IE7 i stariji), kod kojih učitavanje slike blokira izvršavanje naredne naredbe, pa se u takvim situacijama, funkcija za obradu `load` događaja nikada ne bi ni aktivirala.

## Crtanje slika

Nakon dobijanja reference na sliku i njenog potpunog učitavanja, može se preći na crtanje slike unutar `canvas`-a. Crtanje slike obavlja se korišćenjem metode `drawImage()`.

**Metoda drawImage()**

Metoda `drawImage()` koristi se za crtanje slike unutar `canvas` elementa. Poseduje brojne oblike koji se međusobno razlikuju po broju i nameni ulaznih parametara. Osnovni oblik ove metode prihvata tri parametra:

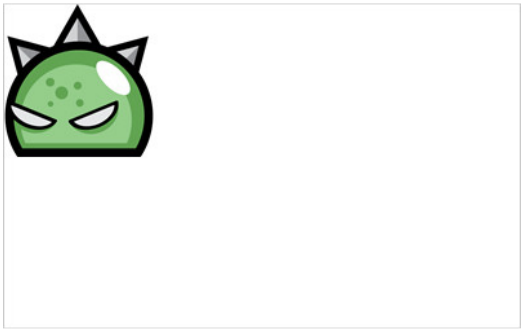
```
drawImage(image, x, y)
```

Prvi parametar se odnosi na sliku koju je potrebno nacrtati, a druga dva na `x` i `y` koordinate referentne tačke na kojoj će se naći gornji, levi ugao slike.

Kod za crtanje prethodno učitane slike može da izgleda ovako:

```
var img = new Image();

img.addEventListener('load', function () {

    ctx.drawImage(img, 0, 0);

}, false);

img.src = 'enemy.png';
```

Na ovaj način, unutar canvas elementa se dobija slika predstavljena enemy.png fajlom koji se nalazi u istom folderu kao i HTML dokument sa prikazanim JavaScript kodom (slika 7.1).



*Slika 7.1 - Crtanje slike unutar canvas-a*

Sliku iz prikazanog primera, možete da preuzmete korišćenjem sledećeg linka:

*enemy.png*

**Napomena**

I u ovoj lekciji koristimo šablon koji smo uspostavili na početku upoznavanja sa Canvas API-em. Stoga je prikazani kod potrebno pisati unutar draw() metode koja se aktivira kada je kompletan HTML dokument učitan, nakon naredbe za dobijanje konteksta za crtanje:

```
<canvas id="my-canvas" width="400" height="250">
    Your web browser does not support canvas element.
</canvas>

<script>
    window.onload = draw;

    function draw() {
        let myCanvas = document.getElementById("my-canvas");
        if (myCanvas.getContext) {
            var ctx = myCanvas.getContext('2d');
            var img = new Image();
            img.addEventListener('load', function () {
                ctx.drawImage(img, 0, 0);
            }, false);
            img.src = 'enemy.png';
        } else {
            alert("Canvas is not supported.");
        }
    }
</script>
```

## Rukovanje veličinom slike

Podrazumevano, slika koja se crta unutar canvas-a ima svoju izvornu veličinu (u našem primeru je to 116x118px). U nekim situacijama može se javiti potreba za promenom veličine slike koja se crta unutar canvas-a. Ukoliko je potrebno promeniti veličinu slike, moguće je koristiti metodu drawImage() sa dva dodatna parametra:

```
drawImage(image, x, y, width, height)
```

Dva dodatna parametra (width i height) odnose se na širinu i visinu koju će slika imati unutar canvas elementa, respektivno.

Prilikom eksplicitnog definisanja veličine slike potrebno je voditi računa o sledećem:

- proporcije slike (odnos stranica) moraju odgovarati originalnoj slici; u protivnom će doći do deformacije slike, što je retko ono što programer želi da postigne
- nije dobro da veličina slike unutar canvas elementa bude veća od njene stvarne veličine, jer u tom slučaju dolazi do pojave pikselizacije i degradacije kvaliteta

Na osnovu ovoga možemo da zaključimo da je mogućnost eksplicitnog definisanja veličine slike unutar canvas elementa najkorisnija za smanjivanje slike, naravno poštujući njene izvorne proporcije. Kod za obavljanje takvog postupka, ilustruje sledeći primer:

```
var img = new Image();

img.addEventListener('load', function () {

    ctx.drawImage(img, 0, 0, img.width / 2, img.height / 2);

}, false);

img.src = 'enemy.png';
```

Primer ilustruje logiku za crtanje slike u dva puta manjoj veličini od izvorne. Možete videti da se prilikom formiranja novih dimenzija, uzimaju u obzir originalne, koje se dobijaju korišćenjem svojstava width i height.

**Pitanje**

Slika se unutar canvas elementa crta korišćenjem metode:

**a) drawImage()**
b) drawImg()
c) drawPicture()
d) drawRaster()

**Objašnjenje**

*Metoda drawImage() koristi se za crtanje slike unutar canvas elementa. Poseduje brojne oblike koji se međusobno razlikuju po broju i nameni ulaznih parametara.*

## Kreiranje isečka slike

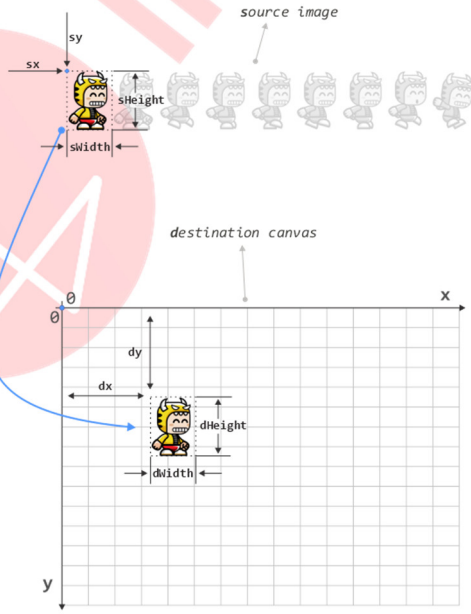
Još jedna, veoma korisna intervencija koju je moguće sprovoditi nad slikama koje se crtaju unutar `canvas` elementa jeste mogućnost crtanja samo jednog dela slike. Za obavljanje takvog posla koristi se `drawImage()` metoda sa sledećim parametrima:

```
drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)
```

Uloga prikazanih parametara je sledeća:

- `image` - slika čiji isečak će biti nacrtan
- `sx` - x koordinata referentne tačke isečka slike unutar izvorne slike
- `sy` - y koordinata referentne tačke isečka slike unutar izvorne slike
- `sWidth` - širina isečka
- `sHeight` - visina isečka
- `dx` - x koordinata referentne tačke isečka unutar `canvas` elementa
- `dy` - y koordinata referentne tačke isečka unutar `canvas` elementa
- `dWidth` - širina koju će isečak zauzimati unutar `canvas` elementa
- `dHeight` - visina koju će isečak zauzimati unutar `canvas` elementa

Pored `image` parametra, svi ostali su za nas u ovom trenutku nepoznanica. Prvo je neophodno da primetite da prva polovina parametara započinje prefiksom **s**, a druga polovina prefiksom **d**. Ovi prefiksi se odnose na *source* i *destination*, respektivno. Tako se parametri sa prefiksom *s* koriste za kreiranje isečka unutar izvorne slike, odnosno za njegovo pozicioniranje i definisanje njegove veličine. Parametri sa prefiksom *d*, odnose se na poziciju i veličinu prethodno dobijenog isečka unutar odredišnog `canvas` elementa. Sve ovo je ilustrovano slikom 7.2.




*Slika 7.2 - Uloga različitih parametara drawImage() metode prilikom kreiranja isečka slike*

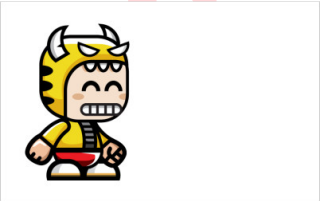
Mogućnost crtanja samo jednog dela neke slike, intenzivno ćemo koristiti u narednom modulu prilikom kreiranja jedne igre. Naime, česta praksa prilikom kreiranja igara jeste grupisanje većeg broja prikaza jednog elementa, unutar jedne slike. Upravo takvu situaciju ilustruje slika 7.2. Različita stanja u kojima se može naći glavni karakter igre koju ćemo kreirati u narednom modulu, objedinjena su unutar jedne slike. Korišćenjem metode `drawImage()`, obavlja se isecanje samo jednog prikaza našeg karaktera. Evo kako izgleda konkretan kod kojim će da bude obavljeno ono što je ilustrovano slikom 7.2:

```
var img = new Image();

img.addEventListener('load', function () {

    ctx.drawImage(img, 0, 0, img.width / 9, img.height, 25, 25,
img.width / 9, img.height);

}, false);

img.src = 'runner.png';
```

Sliku koja se koristi u primeru, možete da preuzmete sa sledećeg linka:

*runner.png*

Prikazanim kodom se unutar `canvas` elementa dobija prikaz kao na slici 7.3.



*Slika 7.3 - Crtanje samo jednog isečka slike unutar canvas elementa*

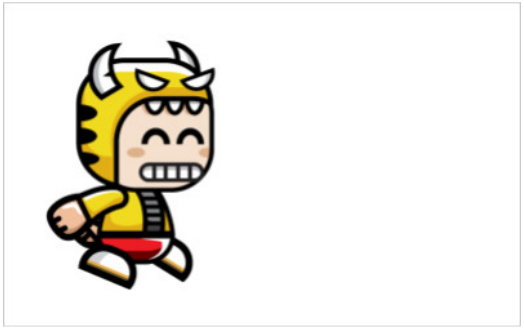
**Spite i Spritesheet**

Izvorna slika koja se koristi u upravo prikazanom primeru drugačije se naziva grupa sprajtova (*spritesheet, engl*). Spritesheet je sačinjen iz većeg broja pojedinačnih sprajtova (*sprite, engl.*). Sprajt se definiše kao dvodimenzionalna slika kojom se predstavlja jedan kadar animacije. U upravo prikazanom primeru, mi smo obavili izolovanje i crtanje jednog sprajta iz slike koja predstavlja spritesheet.

Bitno je da primetite da je visina sprajta formirana na osnovu visine izvorne slike (s obzirom da su pojedinačni sprajtovi složeni u jedan red), dok je širina dobijena deljenjem širine originalne slike sa brojem 9. Unutar originalne slike postoji 9 pojedinačnih sprajtova, pa se širina jednog, dobija deljenjem širine izvorne slike na 9 delova. S obzirom da smo želeli da prikažemo prvi sprajt, za izvorne koordinate (`sx` i `sy`) postavljene su vrednosti 0. Ukoliko želimo da dobijemo neki drugi sprajt, ove vrednosti je potrebno promeniti. Evo na primer kako izgleda kod za dobijanje trećeg sprajta:

```
ctx.drawImage(img, (img.width / 9) * 2, 0, img.width / 9, img.height,
25, 25, img.width / 9, img.height);
```

Jedina izmena tiče se udaljenosti isečka od leve ivice originalne slike (parametar `sx`). Kako bismo se pozicionirali na početak trećeg sprajta, vrednost parametara `sx` formirana je dobijanjem širine jednog sprajta, a zatim njenim množenjem sa 2, kako bi se preskočila prva dva sprajta. Unutar `canvas` elementa se dobija prikaz kao na slici 7.4.



*Slika 7.4 - Crtanje 3. sprajta*

## Crtanje teksta

Crtanje teksta unutar `canvas` elementa može se obaviti korišćenjem dve metode (tabela 7.1).

| Metoda | Opis |
|---|---|
| `fillText(text, x, y [, maxWidth])` | crta tekst sa ispunom |
| `strokeText(text, x, y [, maxWidth])` | crta tekst bez ispune, odnosno tekst koji je sačinjen samo iz okvira |

*Tabela 7.1 - Metode za crtanje teksta*

Obe metode za crtanje teksta prihvataju identične parametre:

- `text` - tekst koji je potrebno nacrtati
- `x` - x koordinata početne tačke za ispisivanje teksta (gde će tekst unutar `canvas`-a biti nacrtan)
- `y` - y koordinata početne tačke za ispisivanje teksta
- `maxWidth` - opcioni parametar kojim je moguće definisati maksimalnu širinu koju će tekst zauzeti unutar `canvas`-a; podrazumevano, ne postoji ograničenje kada je u pitanju maksimalna širina teksta koji se crta, pa tako veoma lako on može prevazići okvire u koje je planirano da bude smešten; kada se vrednost ovoga parametra definiše, web pregledač nastoji da tekst upakuje unutar definisane širine

Osnovni način za crtanje teksta može da izgleda ovako:

```
ctx.fillText('Hello World', 40, 50);
```

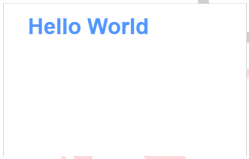
Na ovaj način će unutar canvas elementa da bude nacrtan tekst *Hello World*. Ipak on će imati podrazumevanu veličinu i stilizaciju. Kako bi se uticalo na osobine teksta koji se crta, moguće je koristiti sledeća svojstva konteksta za crtanje:

- `font` - veličina, stil i font teksta; sintaksa za definisanje vrednosti ovoga svojstva je identična kao i sintaksa istoimenog CSS svojstva; tako je na primer, moguće definisati `'bold 36px sans-serif'`
- `textAlign` - poravnanje teksta u odnosu na `x` koordinatu koja se prosleđuje `fillText()` metodi; moguće vrednosti su `start`, `end`, `left`, `right` i `center`
- `textBaseline` - poravnanje u odnosu na osnovnu liniju teksta (*baseline, engl.*); moguće vrednosti su `top`, `hanging`, `middle`, `alphabetic`, `ideographic` i `bottom`
- `direction` - usmerenje teksta; moguće vrednosti su `ltr`, `rtl` i `inherit`
- `fillStyle` - boja teksta

Primer crtanja stilizovanog teksta može da izgleda ovako:

```
ctx.font = 'bold 36px sans-serif';
ctx.fillStyle = "#4F95FF";

ctx.fillText('Hello World', 40, 50);
```

Na ovaj način se dobija tekst kao na slici 7.5.




*Slika 7.5 - Primer ispisivanja stilizovanog teksta unutar canvas elementa*

Nešto ranije ste mogli da vidite da metode za crtanje teksta omogućavaju definisanje maksimalne širine koju će tekst da zauzme unutar canvas elementa, navođenjem širine kao poslednjeg četvrtog parametra. Ipak, oslanjanje na web pregledač da tekst upakuje unutar definisane širine, veoma je loša praksa, pošto u najvećem broju slučajeva tekst završi deformisan. Najbolji način za kontrolisanje prostora koji će tekst da zauzme unutar canvas-a, jeste upotreba metode **measureText()**. Reč je o metodi koja omogućava da se tekst izmeri pre nego što bude nacrtan i da se na taj način stekne uvid u prostor koji će biti potreban za njegov prikaz:

```
let textToDraw = 'Hello World';
let textSize = 49;
let text;

do {
    textSize--;

    ctx.font = 'bold ' + textSize + 'px sans-serif';
    ctx.fillStyle = "#4F95FF";

    text = ctx.measureText(textToDraw);
} while (text.width >= 100);

ctx.fillText(textToDraw, 40, 50);
```

Ukoliko se utvrdi da tekst zauzima veći prostor od planiranog, najbolje je smanjiti njegovu veličinu, korišćenjem svojstva `font`. Upravo takav pristup ilustruje i prikazani primer. Željena širina teksta je `100px`. Korišćenjem `do/while` petlje obavlja se postepeno smanjivanje veličine teksta, sve dok njegova širina koju će zauzimati unutar `canvas` elementa ne postane jednaka ili manja od 100.

## Rezime

- unutar `canvas` elementa je moguće crtati proizvoljan tekst i slike
- kako bi neka slika bila nacrtana unutar `canvas`-a, neophodno je prvo kreirati njenu objektnu predstavu, korišćenjem konstruktorske funkcije `Image()`
- putanja do slike se postavlja korišćenjem `src` svojstva objekta koji predstavlja sliku
- preduslov za crtanje neke slike unutar `canvas` elementa jeste kompletan završetak njenog učitavanja
- metoda `drawImage()` koristi se za crtanje slike unutar `canvas` elementa
- ukoliko je potrebno promeniti veličinu slike, moguće je koristiti metodu `drawImage()` sa dva dodatna parametra: `width` i `height`
- crtanje samo jednog dela neke slike, može se obaviti korišćenjem `drawImage()` metode koja prihvata 9 parametara: `drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)`
- sprajt (*sprite, engl.*) je pojam kojim se definiše dvodimenzionalna slika kojom se predstavlja jedan kadar animacije
- veći broj sprajtova, grupisanih unutar jednog fajla drugačije se naziva spritesheet
- metoda `fillText()` crta tekst sa ispunom
- metoda `strokeText()` crta tekst bez ispune, odnosno tekst koji je sačinjen samo iz okvira
- metoda `measureText()` omogućava da se tekst izmeri pre nego što bude nacrtan i da se na taj način stekne uvid u prostor koji će biti potreban za njegov prikaz