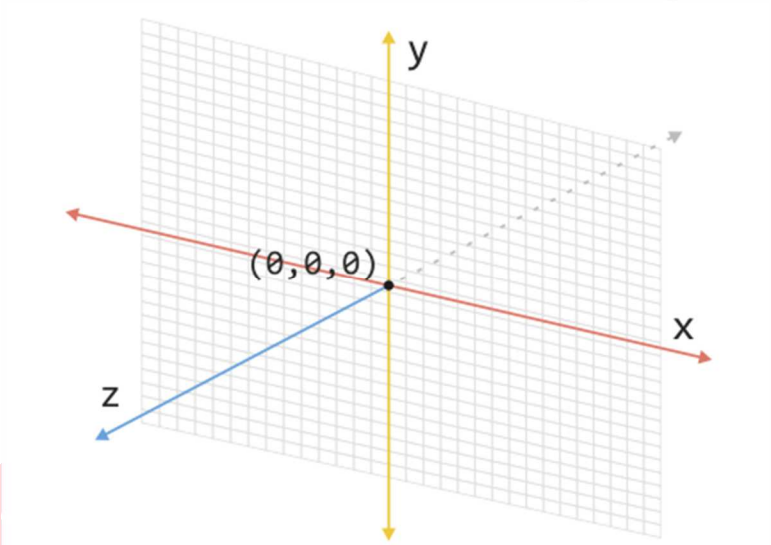
# 3D CSS transformacije

U prethodnoj lekciji ilustrovane su različite CSS transformacije u dvodimenzionalnom koordinatnom sistemu. Pored 2D transformacija, CSS omogućava transformisanje HTML elemenata i u koordinatnom sistemu sa tri dimenzije. Lekcija pred vama biće posvećena različitim tehnikama za postizanje takvih transformacija.

## 3D koordinatni sistem CSS-a

Pre nego što se upustimo u praktično transformisanje elemenata u 3D prostoru, dobro je upoznati se sa osobinama koordinatnog sistema sa tri ose, unutar koga se obavljaju spomenute transformacije. Sve osobine dvodimenzionalnog koordinatnog sistema iznete u prethodnoj lekciji primenjuju se i na sistem sa tri dimenzije. Jedina razlika jeste postojanje jedne ose više (slika 6.1).




*Slika 6.1. 3D koordinatni sistem*

Kao i kod 2D koordinatnog sistema, horizontalna osa je zapravo x-osa, dok je vertikalna osa y-osa. Treća osa (z-osa) upravna je na ravan koju grade x i y osa.

Postojanje jedne ose više nalaže i predstavljanje tačaka korišćenjem tri vrednosti, odnosno jedne za svaku od osa (x, y, z). Tako je i na slici 6.1. referentna tačka 3D koordinatnog sistema predstavljena koordinatama (0,0,0).

## Koje 3D transformacije postoje i kako se definišu?

Korišćenjem CSS-a moguće je sprovesti tri različite vrste transformacija u trodimenzionalnom koordinatnom sistemu:

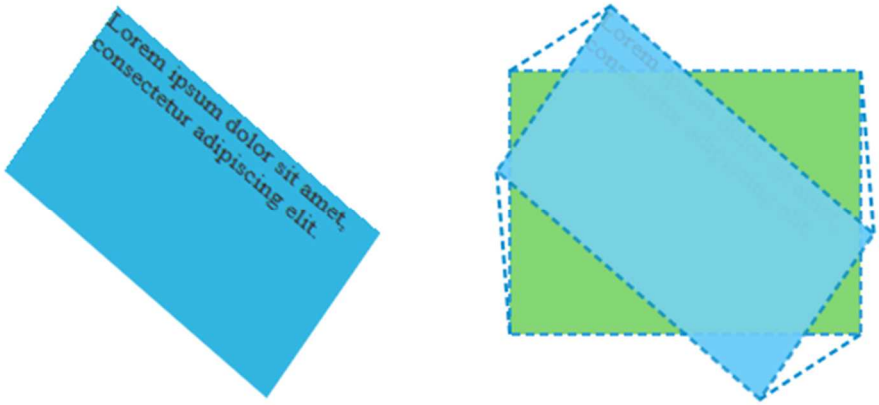
- pomeranje (*translate*);
- skaliranje (*scale*);
- rotiranje (*rotate*).

3D transformacije funkcionišu na sličan način kao i 2D transformacije prikazane u prethodnoj lekciji. To znači da se i za njihovo definisanje upotrebljava svojstvo **transform**. Ipak, CSS funkcije koje se postavljaju za vrednost ovog svojstva nešto su drugačije. Na primer, u prethodnoj lekciji korišćena je CSS funkcija `rotate()` koja je element rotirala u 2D koordinatnom sistemu. Postizanje rotacije u 3D koordinatnom sistemu može se obaviti na sledeći način:

```
#transformed {
transform: rotateX(30deg) rotateY(30deg) rotateZ(30deg);
}
```

Primer ilustruje korišćenje funkcija `rotateX()`, `rotateY()` i `rotateZ()` kao vrednosti svojstva `transform`.

Prikazani primer proizvodi efekat kao na slici 6.2.




*Slika 6.2. 3D rotiranje*

Identičan efekat može se postići i na sledeći način, objedinjenim definisanjem rotacije po svim osama korišćenjem jedne specijalne CSS funkcije namenjene 3D transformacijama:

```
transform: rotate3d(1,1,1, 30deg);
```

Ovoga puta je iskorišćena funkcija `rotate3d()`. Prva tri parametra se odnose na ose oko kojih će se obaviti rotiranje. Broj 1 označava da će se rotiranje obaviti, a 0 da neće. Inače, prva tri parametra se odnose na `x`, `y` i `z` ose, respektivno. Poslednji parametar definiše ugao rotacije.

Iz upravo prikazanih primera može se videti da su CSS funkcije za postizanje 3D transformacija nešto drugačije. One se mogu podeliti na dve grupe – na funkcije koje obavljaju transformacije po pojedinačnim osama (tabela 6.1) i funkcije koje omogućavaju objedinjeno definisanje transformacija po više osa (tabela 6.2).

| Funkcija | Opis |
|---|---|
| `translateX(x)` | pomeranje po x-osi |
| `translateY(y)` | pomeranje po y-osi |
| `translateZ(z)` | pomeranje po z-osi |
| `scaleX(x)` | promena veličine, ali samo po x-osi |
| `scaleY(y)` | promena veličine, ali samo po y-osi |
| `scaleZ(z)` | promena veličine, ali samo po z-osi |
| `rotateX(angle)` | rotiranje, ali samo po x-osi |
| `rotateY(angle)` | rotiranje, ali samo po y-osi |
| `rotateZ(angle)` | rotiranje, ali samo po z-osi |

*Tabela 6.1. Funkcije za transformisanje po pojedinačnim osama*

Pored funkcija iz tabele 6.1, za postizanje 3D transformacija mogu se koristiti i specifične funkcije koje objedinjuju sve tri ose (tabela 6.2).

| 2D funkcija | 3D funkcija | Opis |
|---|---|---|
| `translate(x,y)` | `translate3d(x,y,z)` | 3D pomeranje |
| `rotate(angle)` | `rotate3d(x,y,z,angle)` | 3D rotiranje |
| `scale(x,y)` | `scale3d(x,y,z)` | 3D promena veličine |
| `matrix3d (n,n,n,n,n,n)` | `matrix3d (n,n,n,n,n,n,n,n, n,n,n,n,n,n,n,n)` | objedinjeno definisanje svih transformacija |

*Tabela 6.2. Funkcije za objedinjeno definisanje transformacija*

Unutar radnog okruženja prikazan je primer sa prethodno definisanim metodama. Pokušajte da izmenite parametre iskorišćenjih metoda kako bi dobili prikaz drugih geometrijskih figura.

**Radno okruženje**

**HTML fajl:**

```
<div class="box"></div>
<div class="box2"></div>
```

**CSS fajl:**

```
.box {
    background: blue;
    width: 100px;
    height: 100px;
    transform: translate3d(50px, 50px, 10px) rotate3d(1,0,1, 20deg);
}
.box2 {
    background: red;
    width: 100px;
```

```
    height: 100px;
    transform: translate(300px, 50px) scale3d(1.2, 2, 3.2);
}
```

\

## Perspektiva

Priča o transformacijama u trodimenzionalnom prostoru ne može proći bez spominjanja pojma perspektive. Perspektiva je pojam koji se odnosi na način na koji posmatrač vidi neki predmet u prostoru. Perspektiva je posebno značajna za kreiranje trodimenzionalnih prikaza na podlozi se dve dimenzije. Upravo takva situacija postoji na webu, s obzirom na to da su displeji uređaja na kojima se prikazuju web sajtovi dvodimenzionalne površine.

Perspektiva govori koliko je posmatrač udaljen od predmeta koji se predstavlja u trodimenzionalnom prostoru. Udaljenost posmatrača od predmeta u njegovim očima stvara drugačiju sliku o predmetu. Tako nešto se može zaključiti i iz realnog života. Predmeti koji su bliži posmatraču čine se većim, dok oni udaljeniji izgledaju manji.

U CSS-u, perspektiva se može definisati na dva načina:

- **korišćenjem funkcije perspective()**

U ovom slučaju se perspektiva definiše kao deo vrednosti `transform` svojstva na konkretnom elementu. Na primer:

```
transform: perspective(700px) rotateX(30deg) rotateY(30deg)
rotateZ(30deg);
```

Potrebno je obratiti pažnju na to da je perspektivu potrebno definisati pre ostalih CSS funkcija za transformisanje.

- **korišćenjem svojstva perspective**

Svojstvo `perspective`, za razliku od istoimene funkcije, omogućava definisanje perspektive na roditeljskom elementu. U takvoj situaciji, definisana vrednost perspektive se primenjuje na svim elementima potomcima:

```
perspective: 700px;
```

Korišćenjem oba načina postiže se definisanje udaljenosti referentne tačke na z-osi od posmatrača. Na taj način se na displeju uređaja može stvoriti drugačiji prikaz određenih elemenata nad kojima su primenjene 3D transformacije. Kao idealan primer poslužiće transformacija pomeranja po z-osi:

```
transform: translate3d(0, 0 , 100px);
```

Prikazanom linijom CSS koda definisano je pomeranje u trodimenzionalnom prostoru, i to isključivo po z-osi, za vrednost od 100px. Ukoliko ovakav CSS opis primenite nad nekim HTML

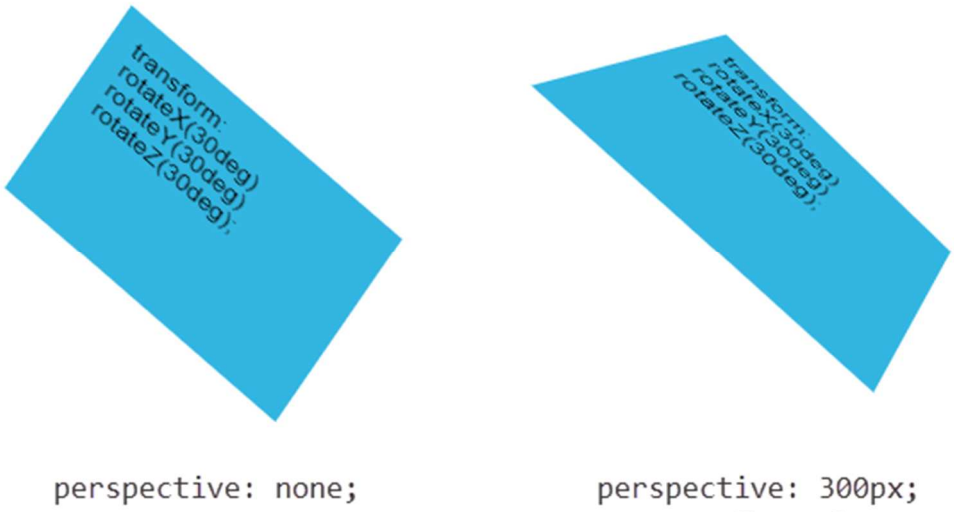
elementom, moći ćete da vidite da se ništa neće dogoditi. Drugim rečima, vizualna predstava elementa će biti identična sa prikazanim CSS opisom i bez njega. Razlog za ovakvo ponašanje leži u činjenici da je podrazumevana vrednost `perspective` svojstva `none`. To znači da je, inicijalno, perspektiva isključena, te da 3D transformacije neće izgledati onako kako se to očekuje.

S obzirom na to da se upravo prikazanim primerom vrši pomeranje elementa po z-osi, a uvidom u sliku 6.1 se može videti da je to osa koja upravno prolazi kroz displej korisničkog uređaja, očekivano je da se transformisani HTML element približava posmatraču, odnosno korisniku, ili se udaljava od njega. Da bi se tako nešto uistinu i dogodilo, neophodno je postaviti vrednost perspektive:


*Slika 6.3. Efekat perspektive na transformisanom elementu*

Slika 6.3. ilustruje efekat perspektive na jednom `div` elementu koji je transformisan korišćenjem `translate3d()` funkcije. Prvi prikaz ilustruje element koji je pomeren po `z`-osi, ali za `perspective` ima vrednost `none`. Središnji prikaz ilustruje isti takav element na kome je aktivirana perspektiva i za njenu vrednost postavljeno 1000px. Na kraju, desno se može videti prikaz elementa sa vrednošću perspektive od 300px. Očigledno je da je element sa nižom vrednošću perspektive krupniji, baš kao da je bliži posmatraču. Analogno, više vrednosti perspektive čine da ovakav element deluje udaljenije od korisnika, pa samim tim i sitnije.

Definisanje perspektive je nezaobilazno za kreiranje realističnih 3D prikaza prilikom transformisanja. Istina je takva da čak ni prvi primer u ovoj lekciji koji je ilustrovao 3D rotiranje nije izgledao adekvatno bez upotrebe perspektive (slika 6.4).




*Slika 6.4. 3D rotacija sa perspektivom i bez perspektive*

Na levoj polovini slike 6.4. može se videti 3D transformacija rotacije prikazana na početku ove lekcije. Rotiranje se obavlja po sve tri ose trodimenzionalnog koordinatnog sistema, ali se iz samog prikaza ne može zaključiti da je reč o transformaciji u 3D prostoru. Zbog toga je na elementu definisana perspektiva od 300px, čime je dobijen prikaz kao na desnoj polovini slike 6.4. 3D efekat je na ovaj način mnogo uočljiviji i izraženiji.

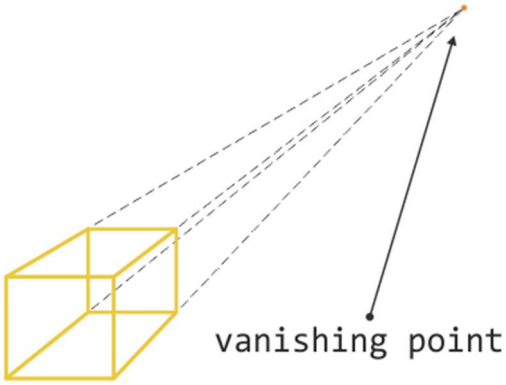
**Funkcija perspective() i svojstvo perspective**

Nešto ranije je rečeno da se definisanje perspektive može obaviti na dva načina: korišćenjem funkcije `perspective()` i svojstva `perspective`. Na oba načina se definiše vrednost perspektive, odnosno fiktivna udaljenost posmatrača od referentne tačke po z-osi. Ipak, između ova dva načina definisanja perspektive postoji jasna razlika:

- funkcijom **perspective()** definiše se perspektiva za svaki element pojedinačno; ovaj pristup je dobar samo ukoliko je potrebno perspektivu definisati na jednom elementu, zato što se na ovaj način svaki element smešta u zaseban 3D prostor;
- ukoliko je potrebno da veći broj elemenata čini zajednički prostor, najbolje je koristiti svojstvo **perspective**; ovo svojstvo definiše perspektivu za sve elemente potomke elementa nad kojim je definisano; ovo praktično znači da se svojstvo `perspective` definiše na **roditeljskom** elementu, dok efekat ima na svim potomcima.

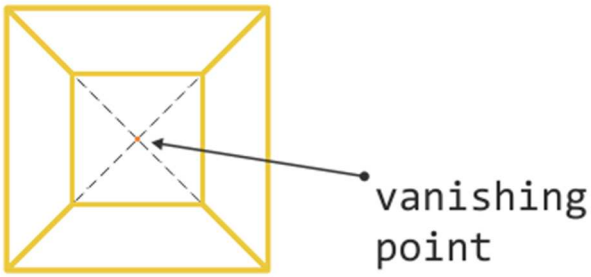
## Tačka nedogleda

Referentna tačka perspektive se drugačije naziva tačka nedogleda (*vanishing point*). To je tačka u kojoj se u daljini spajaju sve paralelne ravni 3D prikaza (slika 6.5).




*Slika 6.5. Tačka nedogleda*

Kada je reč o CSS 3D transformacijama, tačka nedogleda je podrazumevano u samom centru elementa sa definisanom perspektivom (slika 6.6).




*Slika 6.6. Podrazumevani položaj tačke nedogleda*

Analizom slika 6.5. i 6.6. može se zaključiti i to da je položaj tačke nedogleda direktno uslovljen položajem iz kojeg se posmatra neki prikaz u trodimenzionalnom prostoru.

Na položaj tačke nedogleda, pa samim tim i na položaj iz koga se transformisani element posmatra, može se uticati korišćenjem svojstva **perspective-origin**:

```
perspective-origin: x-position;
perspective-origin: x-position y-position;
```

Linije koda ilustruju različite oblike u kojima se `perspective-origin` može upotrebiti. Bitno je napomenuti da se svojstvo `perspective-origin` koristi u kombinaciji sa svojstvom `perspective`, što znači da se oba spomenuta svojstva definišu na roditeljskom elementu, a primenjuju na elementima naslednicima:

```
perspective: 300px;
perspective-origin: 0% 50%;
```

Nešto ranije je rečeno da je tačka nestajanja u samom centru elementa sa perspektivnom, što praktično znači da je podrazumevana vrednost ovog svojstva:

```
perspective-origin: 50% 50%;
```

**Pitanje**

Ukoliko je objedinjeno potrebno definisati perspektivu na svim elementima potomcima nekog elementa, koristi se:

- **svojstvo perspective**
- funkcija perspective()
- svojstvo perspective-origin
- ništa od navedenog

**Objašnjenje:**

*Funkcijom perspective() definiše se perspektiva za svaki element pojedinačno. Ukoliko je potrebno da veći broj elemenata čini zajednički prostor, najbolje je koristiti svojstvo perspective.*

## Očuvanje 3D prostora

Veoma zanimljiva situacija može nastati ukoliko se određena transformacija primeni nad elementom koji već poseduje elemente koji su transformisani korišćenjem 3D transformacija. Takva situacija biće ilustrovana sledećom HTML strukturom:

```
<div id="main-container">
      <div id="boxes-container">
            <div id="div-1" class="box"></div>
            <div id="div-2" class="box"></div>
      </div>
</div>
```

Ovakvi elementi stilizovani su na sledeći način:

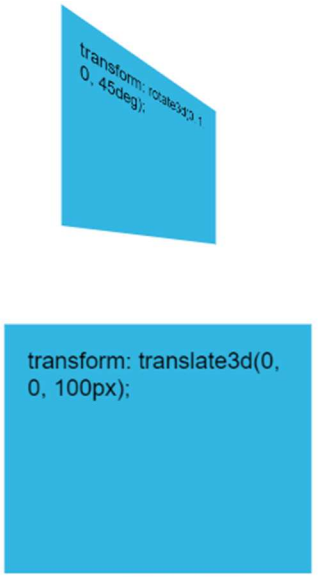
```
.box {
    width: 180px;
    height: 140px;
    background-color: #33B6E2;
    margin: 100px auto;
    padding: 16px;
}

#main-container {
    perspective: 300px;
}

#div-1 {
    transform: rotate3d(0, 1, 0, 45deg);
}
```

```
#div-2 {
    transform: translate3d(0, 0, 100px);
}
```

Može se primetiti da su dva `div` elementa koja se nalaze najdublje u prikazanoj strukturi (`div-1` i `div-2`) transformisana korišćenjem funkcija `rotate3d()` i `translate3d()`. Na ovaj način biće postignuto transformisanje kao na slici 6.7.




*Slika 6.7. Dva div elementa sa definisanim 3D transformacijama*

Pokušajte da unutar radnog okruženja testirate i sledeći primer kako bi uvideli razliku vršenja transformacije nad roditeljskim elementom.

**Radno okruženje**

**HTML fajl:**

```
<div id="main-container">
<div id="boxes-container">
<div id="div-1" class="box"></div>
<div id="div-2" class="box"></div>
</div>
</div>
```

**CSS fajl:**

```
.box {
    width: 180px;
    height: 140px;
    background-color: #33B6E2;
    margin: 100px auto;
    padding: 16px;
}
```

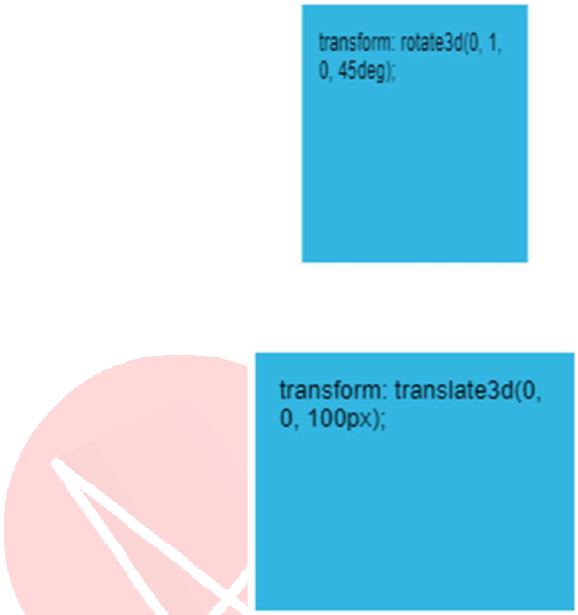
```
#main-container {
    perspective: 300px;
}

#div-1 {
    transform: rotate3d(0, 1, 0, 45deg);
}

#div-2 {
    transform: translate3d(0, 0, 100px);
}
```

Potpuno drugačiji prikaz se dobija ukoliko se izvrši transformisanje i roditeljskog elementa (`boxes-container`) dva upravo prikazana `div`-a (slika 6.8):

```
#boxes-container {
      transform: translate3d(100px, 0, 0px);
}
```




*Slika 6.8. Prikaz nakon transformisanja roditelja i potomaka*

Sa slike 6.8. se može videti da nakon transformisanja roditeljskog elementa (`boxes-container`) dolazi do čudne pojave na njegovim potomcima. Oni više ne izgledaju kao pre, već su poravnati sa ravni svog roditelja. Kako bi se transformacije koje su nad njima definisane adekvatno prikazale, odnosno kako bi se očuvao njihov 3D prostor, potrebno je koristiti svojstvo **transform-style**.

Svojstvo `transform-style` može imati četiri vrednosti:

- `flat`
- `preserve-3d`
- `inital`
- `inherit`

Potpuno očekivano, podrazumevana vrednost svojstva `transform-style` je `flat`, čime se postiže efekat koji je upravo ilustrovan slikom 6.8. Za očuvanje 3D prostora potomaka, za vrednost svojstva `transform-style` potrebno je postaviti `preserve-3d`.

```
transform-style: preserve-3d;
```

## Primer: Kreiranje carousela

Sve tehnike koje su ilustrovane u ovoj lekciji sada će biti iskorišćene za kreiranje jednog realnog primera. Primer će ilustrovati postizanje zanimljivog efekta vrteške (*carousel*). Finalni efekat će biti kao na slici 6.9.

*Slika 6.9. Finalni izgled primera*

Za postizanje ovakvog efekta biće korišćena veoma jednostavna HTML struktura:

```
<div id="main-container">
    <div id="tile-container">
    <div class="tile" id="tile-1"></div>
    <div class="tile" id="tile-2"></div>
    <div class="tile" id="tile-3"></div>
    <div class="tile" id="tile-4"></div>
    <div class="tile" id="tile-5"></div>
    </div>
</div>
```

Unutar glavnog omotača (`main-container`) nalazi se još jedan omotač (`tile-container`), koji sadrži pet `div` elemenata. Svaki od ovih pet `div` elemenata predstavlja po jedan od pravougaonika koji se mogu videti na slici 6.9. Ovih pet `div` elemenata poseduju zajedničku klasu `tile`, a svaki je obeležen i `id` atributom specifične vrednosti.

Da bi se prikazani HTML elementi stilizovali tako da se dobije prikaz kao na slici 6.9, biće korišćeni sledeći CSS opisi:

```
#main-container {
	perspective: 400px;
	position: relative;
	width: 300px;
	height: 220px;
	margin: 80px auto;
}

#tile-container {
	transform: translateZ(-300px) rotate3d(0, 1, 0, 0deg);
	transform-style: preserve-3d;
}

.tile {
	width: 300px;
	height: 220px;
	border-radius: 8px;
	position: absolute;
	margin: 0 auto;
	background-repeat: no-repeat;
	background-position: center;
	background-size: 64px;
	opacity: 0.95;
}


#tile-1 {
	background-image: url("img/like.png");
	background-color: #914E67;
	transform: rotateY(0deg) translateZ(300px);
}

#tile-2 {
	background-image: url("img/idea.png");
	background-color: #4FBA6F;
	transform: rotateY(72deg) translateZ(300px);
}

#tile-3 {
	background-image: url("img/clock.png");
	background-color: #5192D2;
	transform: rotateY(144deg) translateZ(300px);
}

#tile-4 {
	background-image: url("img/star.png");
	background-color: #E8BF1C;
```

```
            transform: rotateY(216deg) translateZ(300px);
      }

      #tile-5 {
            background-image: url("img/note.png");
            background-color: #5D7586;
            transform: rotateY(288deg) translateZ(300px);
      }
```

Unutar radnog okruženja moguće je testirati gore pomenuti kod bez ikonica. Pokušajte da izmenite primer dodavanjem još jedne pločice unutar slajdera.

**Radno okruženje**

**HTML fajl:**

```
<div id="main-container">
   <div id="tile-container">
      <div class="tile" id="tile-1"></div>
      <div class="tile" id="tile-2"></div>
      <div class="tile" id="tile-3"></div>
      <div class="tile" id="tile-4"></div>
      <div class="tile" id="tile-5"></div>
   </div>
</div>
```

**CSS fajl:**

```
#main-container {
      perspective: 400px;
      position: relative;
      width: 300px;
      height: 220px;
      margin: 80px auto;
}

#tile-container {
      transform: translateZ(-300px) rotate3d(0, 1, 0, 0deg);
      transform-style: preserve-3d;
}

.tile {
      width: 300px;
      height: 220px;
      border-radius: 8px;
      position: absolute;
      margin: 0 auto;
      background-repeat: no-repeat;
      background-position: center;
      background-size: 64px;
      opacity: 0.95;
}


#tile-1 {
```

```
      background-color: #914E67;
      transform: rotateY(0deg) translateZ(300px);
}

#tile-2 {
      background-color: #4FBA6F;
      transform: rotateY(72deg) translateZ(300px);
}

#tile-3 {
      background-color: #5192D2;
      transform: rotateY(144deg) translateZ(300px);
}

#tile-4 {
      background-color: #E8BF1C;
      transform: rotateY(216deg) translateZ(300px);
}

#tile-5 {
      background-color: #5D7586;
      transform: rotateY(288deg) translateZ(300px);
}
```

Osnovne stvari na koje je potrebno obratiti pažnju unutar prikazanog CSS-a su:

- na glavnom omotaču (`main-container`) definisana je perspektiva od `400px`; veličina ovoga elementa je ograničena postavljanjem `width` i `height` svojstava, kako prilikom 3D transformisanja ne bi dolazilo do povećavanja širine i visine HTML stranice; takođe, postavljanjem margine na `80px auto`, dodat je gornji i donji spoljašnji razmak i postignuto je centriranje kompletnog prikaza;

- svaka od pet pločica koje postoje unutar carousela poseduje zajedničke osobine definisane klasom `tile`; ovom klasom su definisani visina i širina pločica, zaobljenje ivica, blaga providnost, kao i određene osobine pozadinskih slika; unutar klase `tile` obavlja se i apsolutno pozicioniranje svake od pločica;

- svaka pločica poseduje pozadinsku sliku; slike se nalaze unutar `img` foldera;

- selektorima `#tile-1`, `#tile-2`, `#tile-3`, `#tile-4` i `#tile-5` selektovane su pojedinačne pločice i na taj način definisane su njihove specifične osobine: pozadinska slika i boja pozadine, kao i dve transformacije – rotiranje po y-osi i pomeranje po z-osi;

- vrednosti rotiranja po y-osi za pojedinačne pločice nisu izabrane nasumično; jednostavno, kako bi se pločice ravnomerno rasporedile unutar vrteške, vrednost punog kruga podeljena je sa brojem pločica: 360/5=72; na ovaj način je dobijena vrednost koja predstavlja razliku uglova susednih pločica, te je lako zaključiti da su konačne vrednosti uglova: 0, 72, 144, 216 i 288;

- svaka od pločica je pomerena po z-osi, kako bi se u trodimenzionalnom prostoru stvorilo rastojanje između njih; logično, definisanjem nižih vrednosti, ploče vrteške će biti bliže jedna drugoj i obrnuto;

- kako bi kompletna vrteška mogla da se rotira kao celina, na unutrašnjem omotaču (`tile-container`) definisana je transformacija rotacije po `y`-osi; ovoj transformaciji prethodi pomeranje po z-osi, čime se može uticati na veličinu kompletnog prikaza, koji će biti bliži posmatraču ili udaljeniji od njega;

- s obzirom na to da su 3D transformacije definisane i na roditeljskom, ali i na elementima potomcima, na elementu `tile-container` je definisano CSS svojstvo `transform-style` sa vrednošću `preserve-3d`; na ovaj način je očuvan 3D prostor elemenata potomaka, odnosno u ovom primeru – pločica vrteške.

## Rezime

- 3D transformacije funkcionišu na sličan način kao i 2D transformacije, s tim što omogućavaju transformisanje elemenata korišćenjem koordinatnog sistema sa tri ose.
- U trodimenzionalnom koordinatnom sistemu mogu se obaviti transformacije pomeranja, skaliranja i rotiranja.
- 3D transformacije postižu se nešto drugačijim CSS funkcijama – onim koje omogućavaju transformisanje po pojedinačnim osama i onim koje omogućavaju objedinjeno definisanje osobina transformacija po više osa.
- Perspektiva je pojam koji se odnosi na način na koji posmatrač vidi neki predmet u prostoru.
- U CSS-u, perspektiva se može definisati na dva načina: korišćenjem funkcije `perspective()` i korišćenjem svojstva `perspective`.
- Funkcijom `perspective()` definiše se perspektiva za jedan element.
- Svojstvo `perspective` definiše perspektivu za sve elemente potomke elementa nad kojim je definisano.
- Referentna tačka perspektive se drugačije naziva tačka nedogleda, a reč je zapravo o tački u kojoj se u daljini spajaju sve paralelne ravni 3D prikaza.
- Na položaj tačke nedogleda može se uticati korišćenjem svojstva `perspective-origin`.
- Za očuvanje 3D prostora potomaka, za vrednost svojstva `transform-style` potrebno je postaviti `preserve-3d`.