Analiza delovanja CSS animiranih spletnih grafik v dveh brskalnikih

Maša Ornik[1], Helena Gabrijelčič Tomc[1]

[1]*Katedra za in formacijsko in grafično tehnologijo, Oddelek za tekstilstvo, grafiko in oblikovanje, Naravoslovnotehniška fakulteta, Univerza v Ljubljani, Snežniška 5, Ljubljana*
E-pošta: helena.gabrijelcic@ntf.uni-lj.si

Abstract. *The aim of this paper is to present techniques of CSS animations on graphic elements, to analyse the time requirement in relation to the complexity of the animations and to test them in browsers. The experimental work is divided into two parts i.e., making and optimization of animations and testing in two browsers. The first part contains design of concept for animations and their elements. This is followed by the creation of graphics and their export ready for animation use. Graphic elements were created in a vector design program. Each graphic represents a certain vehicle: a car, a balloon, a submarine and a spaceship with different level of animation complexity. The graphics were animated with the help of CSS and code commands were written based on the desired movement. Then the written code was optimized for browser support and tested in two different browsers. In the results difficulty and time consumption of the experiment depended on the number of graphic elements and the complexity of the desired movement were presented and analysed. The main findings were formed regarding the usefulness, functionality and practicality of CSS as a tool for web animation.*

1 Uvod

Animacije in prehodi elementov na spletu so ena najbolj uporabljenih in prepoznavnih lastnosti interaktivnih spletnih strani. S spletnimi animacijami lahko določene elemente na strani izpostavimo. Omogočajo nam atraktivnejšo predstavitev informacij za katere želimo, da izstopijo iz ozadja in pritegnejo uporabnikovo pozornost. Zaradi svoje razširjenosti se je skozi čas razvilo več različnih načinov izdelave in izvajanja spletnih animacij. V praksi so nam na voljo različna orodja, formati, tehnike ter viri za pomoč pri optimalni izdelavi. Uspešnost izvedbe pa temelji tudi na ustrezni pripravi grafičnih elementov [1,2].

Poleg CSS tehnike poznamo še JavaScript, SVG animacije, Lottie animacije, Web Animations API itd. Prednost programa spisanega izključno v CSS je, da brskalnika ne obremenjujemo z izvajanjem JavaScript kode, če to ni potrebno [3].

CSS3 je uradno najnovejša verzija jezika, ki je izšla leta 1999. Razdeljena je na več modulov, vsak od njih pa vsebuje nove zmogljivosti in dopolnitve predhodnih verzij [3]. Verzija CSS3 je med drugim prinesla možnost izdelave prehodov in animacij. CSS prehod (angl. transition) je sprememba iz enega niza CSS lastnosti v drugega, ki se odvije v določenem časovnem okviru [4]. S CSS3 animacijskim modulom in njegovimi dodatnimi funkcijami gremo lahko še korak dlje od prehodov. Z animacijami izvajanje še nadgradimo, in sicer animiramo iz enega niza lastnosti v drugega, nato v tretjega itd. Animacije se lahko tudi ponavljajo, zaustavijo in premaknejo na začetek izvajanja glede na vnose uporabnika. Za njihovo izvedbo prožilec ni obvezen; lahko se začne že ob sami osvežitvi strani [5]. Najpomembnejši sestavni del CSS animacij so ključne sličice angl. *keyframes*. Pri CSS animaciji ključne sličice določajo slog animiranega elementa ob določenem času [6]. Vsaka animacija mora biti sestavljena iz najmanj dveh ključnih sličic, nato lahko določimo njihove lastnosti. Vsa vmesna stanja izriše brskalnik sam na podlagi določenih sličic.

Pri izdelavi animacije s CSS upoštevamo dva pomembna koraka – opredelitev animacije in nato apliciranje le-te na element. Definiranje animacije pomeni določanje in pripravo ključnih sličic, ki vsebujejo lastnosti katere nameravamo animirati [6].

Za upravljanje CSS animacij uporabljamo lastnosti, ki spominjajo na lastnosti prehodov: dolžino animacije nadzorujemo z lastnostjo *animation-duration,* z lastnostjo *animation-iteration-count* lahko animacijo ponovimo večkrat, glede na poljubno število ponovitev ki jih določimo sami. Z vrednostjo *infinite* pa lahko določimo tudi trajanje v neskončnost oz. dokler uporabnik ne zapre zavihka strani, itd. [6].

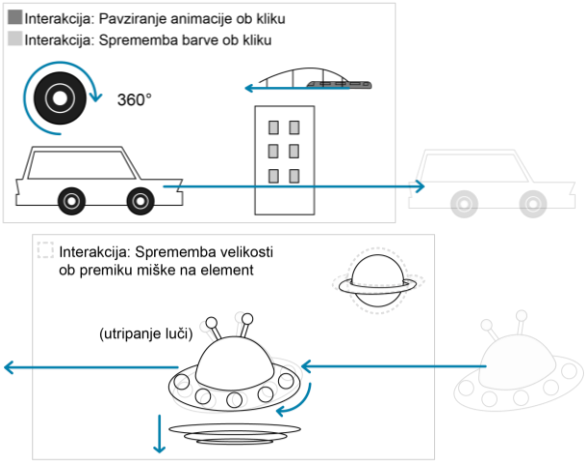
V okviru raziskave nas je zanimala animacija avtorskih grafičnih elementov na spletni strani z različno stopnjo kompleksnosti in različnim številom animacijskih principov s pomočjo CSS kaskadnih stilskih predlog. Želeli smo ugotoviti kako preprosta oziroma zahtevna je izbrana tehnika na področju CSS animacije, koliko časa bomo porabili za izvedbo eksperimenta, ter katere principe gibanja grafičnih elementov lahko uporabimo na strani in v izbranih testnih brskalnikih z namenom optimalnega predvajanja avtorskih animacij na spletu.

2 Eksperimentalni del

V sklopu eksperimentalnega dela smo izdelali štiri animacije z namenom preizkušanja različnih možnosti in lastnosti animiranja s CSS. Vsaka animacija je bila sestavljena iz glavnega elementa, ki predstavlja prevozno sredstvo, ter njegovega ozadja. Določili smo štiri osrednje elemente animacij: toplozračni balon, avtomobil, podmornica in vesoljska ladja, tako da smo jim lahko kasneje v postopku

animiranja dodelili raznolike lastnosti in vrednosti gibanja, od najenostavnejših pri balonu do najkompleksnejših pri vesoljski ladji. Prav tako smo vnaprej določili na kakšen način bo uporabnik lahko upravljal s posameznimi elementi. Vrste sprememb, ki smo jih določili za testiranje so bile: hitrost, prehodi, smer, gibanje po oseh x in y, spreminjanje velikosti elementa, spreminjanje barve elementa, spreminjanje prosojnosti elementa, spreminjanje elementa glede na ukaz.

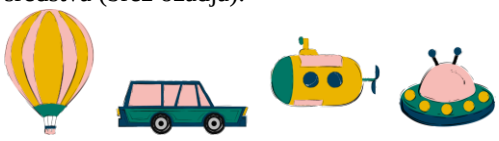
Slika 1 prikazuje načrt za animacijo avtomobila in vesoljske ladje (z največ in najbolj kompleksnimi animacijskimi tehnikami).



Slika 1: Načrt animacije avtomobila in vesoljske ladje

Grafične elemente za animacije smo oblikovali in pripravili v računalniškem programu za risanje vektorskih grafik Adobe Illustrator.

Slika 2 prikazuje končno oblikovana prevozna sredstva (brez ozadja).



Slika 2: Končno oblikovani elementi

Elemente in njihove plasti smo ustrezno poimenovali, saj smo jih tako lažje poiskali in uporabili v kasneje izvoženi kodi skalabilne vektorske grafike SVG (angl. Scalable Vector Graphics) [7, 8]. Po izvozu grafik v SVG format in preden smo se lotili postopka animiranja, smo SVG strukturo izdelkov še optimizirali. V našem primeru smo uporabili spletno orodje SVGOM GUI. Ta ponuja možnost urejanja strukture na hiter, enostaven in učinkovit način.

2.1 Izdelava CSS animacij

Zaradi predhodne uporabe, ter seznanjenosti s ponujenimi funkcijami in orodji, smo za izdelavo animacij uporabili program IntelliJ IDEA. Gre za je integrirano razvojno okolje, ki se uporablja za razvoj računalniške programske opreme. Razvilo ga je češko podjetje JetBrains, ki se ukvarja s pripomočki za razvijalce programske opreme [9].

V nadaljevanju predstavljamo le oblikovanje in izdelavo scene in animacij avtomobila, ter vesoljske ladje, saj sta nam omogočili najštevilčnejšo uporabo raznolikih animacijskih tehnik.

Gibanje avtomobila se je izvajalo od leve proti desni strani, izven vidnega polja, po ravnini vidnega, in spet izven vidnega polja. Realistično gibanje avtomobila zahteva tudi rotacijo njegovih koles. Na sliki 3 je prikazana animacija rotacije koles avtomobila. Uporabili smo le končno stopnjo animacije (100 %) in transformacijo z rotacijo okoli z za 360 stopinj.

```
@keyframes rotacija_kolesa {
        100%  {
            transform: rotateZ(360deg);
        }
    }
```

```
#kolo1 {
    -webkit-animation: rotacija_kolesa 6s infinite linear;
    -moz-animation: rotacija_kolesa 6s infinite linear;
     animation: rotacija_kolesa 6s infinite linear;
     transform-origin: center;
    }
```

Slika 3: Lastnosti CSS animacije za rotacijo koles

Zatem smo se lotili animacije ozadja in interaktivnosti elementov, ki ga sestavljajo. Odločili smo, da ozadju dodamo dodaten element, ki se bo gibal v nasprotno smer od avtomobila. Gibanje vlaka na mostu smo dosegli na enak način kot se premika element avtomobila. Razlikujeta se le v smeri in dolžini gibanja, ter pospešku. Elementu vlaka pa smo dodali še interaktivno lastnost, in sicer možnost zaustavitve animacije s klikom na element. Kako smo to dosegli je prikazano na spodnji sliki 4.

```
#vlak:active, #vlak:focus {
        -webkit-animation-play-state: paused;
        -moz-animation-play-state: paused;
        animation-play-state: paused;
    }
```

Slika 4: Primer CSS kode za ustavitev animacije ob kliku

Pri animaciji vesoljske ladje je šlo za najbolj intenzivno in hitro premikanje elementa. Posledično je element zahteval tudi bolj kompleksno animacijo, ki je definirana na sliki 5. Ključne sličice smo razdelili na devet delov. Objekt je priletel iz desne strani, izven vidnega polja, v kader negativno rotiran za 10 stopinj. Na približno polovici animacije se je ladja sunkovito ustavila in rahlo rotirala v nasprotno smer. Par sekund je postala na sredinski poziciji, ter nato s pospeškom odpeljala naprej v levo izven vidnega polja.

V času postanka smo dodali še žarke v obliki elipse, ki so se sprožili ob spodnjem delu trupa. Slika 6 prikazuje animacijo žarkov. Za dosežen želeni efekt smo spreminjali nasičenost, angl. *opacity*, elips, jih rahlo rotirali, povečali ter premaknili po y osi.

```
@keyframes nlp {
  0% {transform: rotate(-10deg); left: 200%;}
  15% {transform: rotate(-10deg);}
  30% {left: 45%; transform: rotate(10deg);
  40% {left: 50%; transform: rotate(0deg) translateY(0px);
  45% {left: 50%; transform: rotate(0deg) translateY(-
10px;}
  50% {left: 50%; transform: rotate(0deg) translateY(0px);}
  60% {left: 55%; transform: rotate(-10deg);}
  61% {left: 55%;}
  100% {left: -100%;}
}
```

Slika 5: Lastnosti CSS animacije gibanje vesoljske ladje

```
@keyframes abduct {
  0% {opacity: 0;}
  42% {opacity: 0;}
  43% {opacity: 0.8; transform: rotateX(75deg)
       translateY(0px);}
  45% {transform: rotateX(75deg) translateY(200px)
       scale(0.8); opacity: 0.8;}
  47% {transform: rotateX(75deg) translateY(0px); opacity:
       0.8;}
  48% {opacity: 0;}
  100% {opacity: 0;}
}
```

Slika 6: Lastnosti CSS animacije žarkov

Animacija za eno od luči je prikazana na sliki 7. Za efekt hitrega utripanja smo določili, da animacija traja 0,5 sekund v neskončnost in z linearnim potekom prehoda.

```
@keyframes luc1 {
        0% {fill: #023D5D;}
        25% {fill: #ebb400;}
        50% {fill: #f6bbb6;}
        100% {fill: white;}
    }
```

Slika 7: Lastnosti CSS animacije žarkov

Namesto popolnoma črnega ozadja smo se določili, da vanj umestimo animacijo nežno utripajočih zvezd. Sintaksa za animacijo zvezd v ozadju je prikazana na sliki 8.

```
.zvezdice {
        background:#000 url(zvezdice.png) repeat top
center;
        z-index:0;
        margin: 0;
        padding: 0;
    }

.utripanje {
        background:transparent url(utripanje.png) repeat
top
        center;
        z-index: 1;
        margin: 0;
        padding: 0;
        -webkit-animation:utripanje 200s linear infinite;
        -moz-animation:utripanje 200s linear infinite;
        animation:utripanje 200s linear infinite;
    }

@keyframes utripanje {
        from {background-position:0 0;}
        to {background-position:-10000px 5000px;}
```

Slika 8: Lastnosti CSS animacije utripanja zvezd

V ozadje animacije smo dodali še dva elementa planeta z možnostjo interakcije (povečava elementa in povečana nasičenost barv ob drsanju z miško na element), ter element ozvezdja, in galaksije, ki se počasi vrti okoli svoje osi. Slika 9 prikazuje zajem zaslona končne animacije Vesoljska ladja.

Slika 9: Animacija Vesoljska ladja

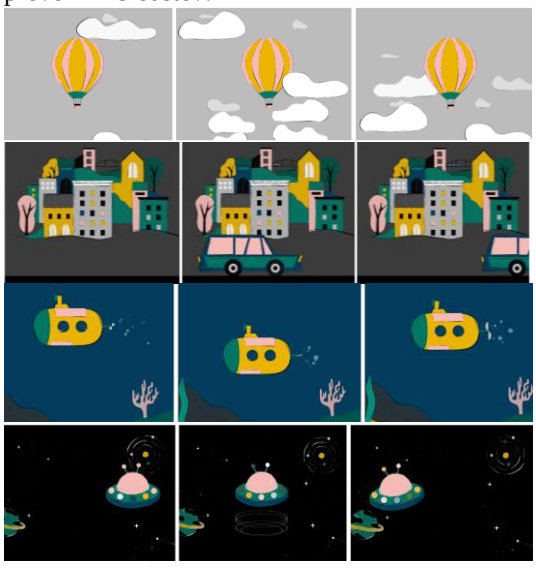
2.2 Testiranje v brskalnikih

Za testiranje smo uporabili vse štiri izdelane animacije; testirali smo jih v dveh različnih brskalnikih – Google Chrome verzija 101 in Mozilla Firefox verzija 100. Za testni računalnik smo uporabili prenosnik povprečnega uporabnika s procesorjem Intel Core i5-8250U (1,60GHz) in 16 GB RAM. Za natančen pregled uporabljenih funkcij, vrednosti in lastnosti, ter interakcij, smo ob opazovanju predvajane animacije, sledili tudi njeni spisani programski kodi in sledili uspešnosti izvedbe določenih ukazov v obeh brskalnikih. Na takšen način smo lahko hitro našli odstopanja v primeru, da ena od dodeljenih lastnosti ni delovala oziroma se ni izvajala pravilno (npr. prepočasi).

3 Rezultati in razprava

Rezultat so štiri različne animacije s tematiko prevoznih sredstev z možnostjo interakcije v obliki spreminjanja barve, velikosti in prosojnosti, ter zaustavitve gibanja glede na ukaz uporabnika, ki smo jih testirali v dveh brskalnikih.

Slike 10 prikazujejo kadre končnih animacij prevoznih sredstev.



Slika 10: Trije kadri končnih animacij

Preglednica 1 prikazuje koliko časa smo porabili za oblikovanje grafičnih elementov in koliko časa za pisanje kode posameznih animacij, ter število uporabljenih tehnik in vrstic končne kode vsake animacije. V preglednici 2 je poleg rezultatov testiranja v brskalnikih predstavljeno kateri principi so bili uporabljeni v štirih scenah.

Preglednica 1: Primerjava časovne zahtevnosti animacij, števila uporabljenih tehnik in vrstic kode

Animacija	Čas oblikovanja (ure)	Čas pisanje kode (ure)	Število uporabljenih tehnik	Število vrstic kode
Balon	5	15	4	813
Avto	10	20	5	199
Podmornica	5	20	4	612
Vesoljska ladja	15	25	5	517

Preglednica 2 prikazuje delovanje animacij v dveh brskalnikih in uporabljene animacijske principe, kjer vidimo, da med brskalnikoma ni vidnih razlik v izvajanju izbranih CSS prehodov in animacij, prav tako smo ugotovili, da so se vse animacije grafičnih elementov predvajale tekoče in bi lahko bile uporabljene na spletni strani za izboljšanje uporabniške izkušnje.

Preglednica 2: Delovanje animacij v dveh brskalnikih

Scena	Animacija, interakcija	Google Chrome 101 in Mozilla Firefox 100
Balon	Premikanje po osi x	Brez vidnih napak
	Premikanje po osi y	Brez vidnih napak
	Rotacija	Brez vidnih napak
	Sprememba barve ob kliku	Brez vidnih napak
Avto	Premikanje po osi x	Brez vidnih napak
	Premikanje po osi y	Brez vidnih napak
	Rotacija	Brez vidnih napak
	Sprememba barve ob kliku	Brez vidnih napak
	Zaustavitev animacije ob kliku	Brez vidnih napak
Podmornica	Premikanje po osi y	Brez vidnih napak
	Rotacija	Brez vidnih napak
	Povečava elementa po premiku miške na element	Brez vidnih napak
	Zmanjševanje nasičenosti z gibanjem	Brez vidnih napak
	Sprememba nasičenosti po premiku miške na element	Brez vidnih napak
Vesoljska ladja	Premikanje po osi y	Brez vidnih napak
	Rotacija	Brez vidnih napak
	Povečava elementa po premiku miške na element	Brez vidnih napak
	Spreminjanje nasičenosti z gibanjem	Brez vidnih napak
	Sprememba nasičenosti po premiku miške na element	Brez vidnih napak

Hitrost nalaganja spletne strani in vseh štirih animacij je zelo zadovoljiva. Stran se gladko naloži, brez zaostajanja elementov.

Po končanem eksperimentu menimo, da so CSS animacije primerne za izdelavo manjših in preprostih animacij na spletnih straneh, kakor so bile tudi naše animacije na avtorskih grafičnih elementih. Takšne animacije so lahko na primer animacija grafičnega elementa ob nalaganju spletne strani ali sprememba stanja preprostega grafičnega elementa ob uporabnikovem kliku nanj. Zaradi omejenih možnosti manipuliranja z elementi, CSS animacije niso primerne za daljše, kompleksne animacije z veliko detajli in velikim številom grafičnih elementov. Pri velikem številu elementov, je CSS za animiranje lahko zamuden [10].

4 Zaključek

CSS animacije omogočajo osnovne oblike gibanja grafičnih elementov. Sami smo že v fazi načrtovanja omejene možnosti animiranja s CSS vzeli v zakup.

Menimo da so CSS animacije primerne za animiranje manjšega števila grafičnih elementov na spletni strani in predvsem tistih elementov, ki ne zahtevajo kompleksnih gibanj. CSS se je izkazal za uporabno orodje na področju kratkih prehodov in osnovnih premikov grafičnih elementov na strani. Predlagali bi uporabo tovrstnega animiranja za krajše animacije in prehode na spletnih straneh, kot je na primer animacija logotipa na spletni strani. Uporaben je tudi, ko želimo omogočiti preproste interaktivne prvine na spletne strani, na primer sprememba barve sestavnega elementa strani, ko uporabnik z miško klikne nanj. V primeru, da bi se želeli lotiti zelo kompleksnih animacij z veliko detajli, ki bi zahtevale karseda realističen učinek gibanja elementov, pa izključno CSS zaradi svoje zamudne in programerske narave najverjetneje ne bi uporabili. V primeru potrebe po animiranju grafičnih elementov na spletni strani z veliko detajlov in kompleksnejšimi animacijskimi tehnikami, bi se tega lotili s CSS v kombinaciji z JavaScript, predvsem za izboljšano interaktivnosti elementov.

Literatura

[1] A. Brundrett: Motion Design: An Intro to UX Choreography. User Experience Magazine, 16(4), 2016.

[2] V. Head: *Designing Interface Animation: Improving the User Experience Through Animation*. Rosenfeld Media, 2016

[3] K. Chinnathambi: *Creating Web Animations: Bringing Your UIs to Life*. O'Reilly Media, 2017

[4] D. S. McCfarland: *CSS3: the missing manual.* 3rd edition. Beijing; Cambridge; Farnham; Köln; Sebastopol; Tokyo; O'Reilly Media, 2013

[5] CSS Animations. *W3Schools*, [dostopno 20. 2. 2022] https://www.w3schools.com/css/css3_animations.asp

[6] Pseudo-classes and pseudo-elements. *MDN web docs*, [dostopno 28. 2. 2022] https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Pseudo-classes_and_pseudo-elements

[7] S. Drasner: *SVG Animations: From Common UX Implementations to Complex Responsive Animation.* O'Reilly Media, 2017

[8] A. Bellamy-Royds, K. Cagle, D. Storey: *Using SVG with CSS3 and HTML5: Vector Graphics for Web Design*, O'Reilly Media, 2017

[9] T. Boskovic: Understanding CSS vendor prefixes. V *DZone: programming & DevOps news, tutorials & tools*, [dostopno 28. 2. 2022] https://dzone.com/articles/understanding-css-vendor-prefixes

[10] E. Meyer, E. Weyl: CSS: The Definitive Guide: Visual Presentation for the Web 4th Edition, O'Reilly Media, 2017