SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3905

UPORABA TRODIMENZIJSKIH PRIKAZA U INTERNET PREGLEDNICIMA

Gabrijel Mrgan

Zagreb, lipanj 2015.

Sadržaj

Uvod

Ovaj rad se bavi izradom podrške za ostvarivanjem trodimeznijskog prikaza u okviru Web-preglednika. Kao primjer za demonstriranje WebGL tehnologije ostvaren je pomoću Javascript-a te biblioteka HTML2Canvas, Jquery i Lodash razvojni okvir za izradu tranzicija između dvije ili više web stranica. Također je ostvareno više primjera trodimenzionalnih tranzicija uz pomoć Three.js biblioteke koja koristi WebGL za trodimenzijski prikaz te Tween.js biblioteke za korištenje interpolacijskih animacija.

Razvojni okvir podržava promjenu između dvije stranice uz pomoć preddefinirane tranzicije, te odlazak u tranzicijsku mapu – tranzicija u kojoj se može odabrati sljedeća željena stranica.

Razvojni okvir je izveden modularno, tako da se nove tranzicije i tranzicijske mape mogu lagano dodavati u već postojeći sustav.

1. Izrada razvojnog okvira

Kao ispomoć za izradu razvojnog okvira potrebno je imati osnovni HTML dokument na kojem će biti učitan glavni tranzicijski kontroler. Osnovna struktura HTML-a za izradu i korištenje okvira je sljedeća:

- Glava dokumenta – standardna HTML glava
- Tijelo dokumenta
    - HTML jedne stranice
    - HTML jedne stranice
    - ...
    - Script oznake za učitavanje Javascript datoteka

Takva struktura podrazumijevati će se dalje kroz rad. Kada se u radu spominje stranica misli se na jednu od više HTML struktura definiranih u tijelu dokumenta.

Primjer HTML dokumenta:

```
<html>
 <head>
   ...
 </head>
 <body>
   <!-- Prva stranica -->
   <div class="trans-page trans-js-page-1">
     ...
   </div>
   <!-- Druga stranica -->
   <div class="trans-page trans-js-page-2">
     ...
   </div>
   <!-- Učitavanje Javascript datoteka -->
   <script src="scripts/transMaster.js"></script>
   ...
 </body>
</html>
```

1.1. Izrada osnovne strukture tranzicijskog kontrolera

Kontroler je izrađen kao Javascript funkcija-konstruktor, kojoj prilikom stvaranja korištenjem ključne riječi *new* predajemo opcije za inicijalizaciju. Inicijalizacija kontrolera trebala bi biti u

zasebnoj skripti. Ukoliko se nikakve opcije ne predaju koriste se standardne postavke definirane skripti kontrolera.

Opcije koje možemo predati kontroleru biti će definirane kroz detalje izrade samog kontrolera.

1.2. Izrada izmjene stranica

Zbog načina na koje web okruženje radi, nije moguće napraviti tranzicije između dvije stranice koristeći standardne hiper-veze, jer bi prilikom promjene stranice sav Javascript kod s klijentske strane bio ugašen. Zbog toga samu izmjenu stranica potrebno je implementirati u razvojnom okviru.

Implementacija koju ovaj rad opisuje bit će jednostavna zamjena HTML strukture uz pomoć Javascript-a. Prilikom inicijalizacije svaka stranica će biti maknuta iz DOM-a te spremljena u memoriju pod unikatnim nazivom. Kada se stranica treba učitati iz memorije se uzima cijela struktura te postavlja direktno ispod tijela dokumenta ili na mjesto definiranom opcijom kontrolera.

U implementaciji razvojnog okvira koju ovaj rad opisuje stranice se definiraju pomoću selektora klasa, zajedno sa svojim unikatnim imenom prilikom inicijalizacije kontrolera, također, jednoj stranici se predaje i zastavica ukoliko je ta stranica trenutna. Na taj način je korisniku okvira dopuštena potpuna kontrola nad dijelom stranice koji se izmjenjuje.

Primjer instanciranja kontrolera s definiranim stranicama:

```
var TM = new TransMaster({
  pages: [
    {
      current: true,
      name: 'first',
      container: '.trans-js-page-1',
    },
    {
      name: 'second',
      container: '.trans-js-page-2',
    },
    {
      name: 'third',
      container: '.trans-js-page-3'
    }
  ],
  pageContainer: 'body',
});
```

1.3. Priprema kontrolera za tranzicije

Glavni problem koji se javlja kod izrade tranzicija između HTML stranica je što Javascript nema direktan pristup iscrtanoj stranici. Da bi iscrtali HTML stranicu u trodimenzionalnom prostoru pomoću WebGL-a potreban je način dohvaćanja te iscrtane stranice.

Srećom Javascript ima pristup potpunoj strukturi DOM-a te pripadajućem CSS-u, pa je moguće stranicu simulirano iscrtati na **canvas** element. Sa canvas elementa moguće je dobiti iscrtanu sliku enkodiranu u bazi 64 pomoću funkcije *canvas.toDataURL(),* koja se nadalje može koristiti kao bilo koja druga slika na web-u.

Kako bi ostvarili iscrtavanje HTML stranice u ovom radu za izradu razvojnog okvira koristi se vanjska biblioteka HTML2canvas koja će obraditi potrebni CSS te stranicu iscrtati na canvas.

Korištenje biblioteke HTML2canvas je vrlo jednostavno, poziva se funkcija *html2canvas()* kojoj se predaju opcije te povratna funkcija koja će vratiti canvas element s iscrtanom HTML stranicom.

Prilikom inicijalizacije kontrolera poziva se HTML2canvas za svaku stranicu definiranu u opcijama, koje se potom iscrtavaju. Dobiveni canvas element se onda zajedno s pripadajućom slikom spremaju s informacijama o stranici. Također, radi izrade složenijih tranzicija, svaki podelement stranice se iscrtava te sprema uz informacije o poziciji elementa i njegovoj veličini.

Nakon spremanja slika svih stranica njihov izgled možemo simulirati unutar trodimenzionalnog prikaza u tranzicijama tako da sliku postavimo kao teksturu određenog elementa.

1.4. Izrada učitavanja tranzicija

Tranzicije bi trebale biti definirane u zasebnim Javascript skriptama kao enkapsulirane cjeline, pa je potrebno napraviti način da se tranzicije učitaju u kontroler. Kako bi se to ostvarilo bez da korisnik mora ručno učitavati svaku tranziciju nakon što je skripta dodana u HTML, potreban je globalni objekt s kojim se to može ostvariti.

Objekt je vrlo jednostavan, s funkcijama za dodavanje i dohvaćanje tranzicija koje su dodane, svaka tranzicija se registrira pozivom funkcije za dodavanje. Svaka tranzicija je definirana kroz svoje ime, svoje funkcije, te svojim opcijama.

Radi jednostavnosti implementacije, tranzicije i tranzicijske mape imaju svoje zasebne liste, iako su definirane gotovo isto, sa svojim imenom, funkcijama i opcijama.

Tijekom inicijalizacije kontrolera se poziva objekt te se dohvaćaju sve učitane tranzicije i tranzicijske mape, koje se onda dodaju u sam kontroler pod svojim unikatnim imenom.

Potrebno je također dodati nove opcije za kontroler tranzicija – ime podrazumijevane tranzicije koja će se koristiti za promjenu stranice i ime podrazumijevane tranzicijske mape ukoliko se koristi.

1.5. Funkcija izmjena stranice

Pozivanje same izmjene stranice ne obavlja se unutar kontrolera, već unutar korisničke skripte koja je instancirala sam kontroler. Kao primjer, to se može vrlo jednostavno ostvariti korištenjem JQuery biblioteke tako da se postavi događaj za klik miša na gumb.

Primjer pozivanja promjene stranice i pozivanja otvaranja tranzicijske mape.

```
$('body').on('click', '.js-change-page', function() {
  var $this = $(this);
  TM.goToPage($this.data('page'));
});

$('body').on('click', '.js-map', function() {
  TM.openMap();
});
```

Funkcija izmjene stranice (kao i funkcija za otvaranje mape) zadužena je za pozivanje tranzicije te stvarnu zamjenu HTML strukture stranice.

Svaka tranzicija je definirana kao:

- Opcije – sve opcije koje su potrebne unutra funkcije tranzicije
- Ime – ime preko kojega se može dohvatiti tranzicije
- Funkcije tranzicije
    - Funkcija prije tranzicije – funkcija koja se poziva prije same tranzicije
    - Funkcija tranzicije – funkcija koja obavlja tranziciju

- Funkcija poslije tranzicije – funkcija koja se poziva nakon obavljene tranzicije

Funkcija izmjene stranice početno postavlja objekt **transitionTemp**, kao memoriju za korištenje unutar tranzicije te poziva funkciju prije tranzicije, koja bi trebala služiti kao inicijalizacijska funkcija (Implementacija može varirati ovisno o tranziciji). Nakon završetka funkcije prije tranzicije, kontroler miče stranicu iz DOM strukture, te poziva funkciju tranzicije. Nakon funkcije tranzicije, kontroler poziva funkciju poslije tranzicije nakon koje se postavlja nova stranica na koju se došlo izmjenom. U slučaju tranzicijskih mapa, stranica na koju se dolazi se dobije od funkcije poslije tranzicije, a sve ostalo se ponaša isto.

Implementacija funkcije za izmjenu stranica:

```
change: function(currentPage, nextPage) {
  //Set up transition options
  t.transitionTemp = {
    o: self._transitions[transType].options,
  };

  t.beforeTransition.call(currentPage, nextPage, function(status) {
    this._removePage(currentPage);

    t.doTransition.call(currentPage, nextPage, function(status) {

      t.afterTransition.call(currentPage, nextPage, function(status) {
        this._putPage(nextPage);
      });
    });
  });
}
```

2. Definicija tranzicije i tranzicijske mape

Tranzicije i tranzicijske mape se ostvaruju neovisno o tranzicijskom kontroleru kao zaseban Javascript dokument. Svaku tranziciju koju korisnik želi koristiti potrebno je učitati pomoću script tag-a u HTML dokumentu, iza učitanog tranzicijskog kontrolera.

2.1. Struktura tranzicije i tranzicijske mape

Struktura objekta tranzicije koju tranzicijski kontroler zahtjeva je sljedeća:

- options – opcije tranzicije, sve opcije koje tranzicija koristi u svojim funkcijama, mogu se mijenjati kroz tranzicijski kontroler
- name – unikatno ime tranzicije, koristi se za dohvaćanje željene tranzicije kroz kontroler
- beforeTransition(current, next, callback) – funkcija tranzicije koja se poziva prije same tranzicije, dobiva trenutnu stranicu (sa svim informacijama o stranici), iduću stranicu, te povratnu funkciju koju je potrebno pozvati pri završetku funkcije
- doTransition(current, next, callback) – funkcija same tranzicije, gdje se odvija tranzicija, također dobiva trenutnu i iduću stranicu te povratnu funkciju koju je potrebno pozvati kada funkcija završi
- afterTransition(current, next, callback) – funkcija poslije tranzicije, poziva se nakon to je tranzicija završila, dobiva sve argumente kao i prethodne dvije, pri završetku poziva povratnu funkciju

Nakon definicije tranzicije potrebno ju je registrirati u kontroler što se čini pozivom na globalni objekt za registriranje tranzicija.

Primjer najjednostavnije tranzicije:

```
var simpleTransition = {
  options: {},
  name: 'simple-transition',
  beforeTransition: function(cur, next, callback) {
    callback();
  },
  doTransition: function(cur, next, callback) {
    callback();
```

```
  },
  afterTransition: function(cur, next, callback) {
    callback();
  }
}

TC.registerTransition(simpleTransition);
```

Struktura tranzicijske mape se razlikuje u imenima funkcija te u argumentima koje prihvaćaju

- beforeTransitionMap(current, pages, callback)
- doTransitionMap(current, pages, callback)
- afterTransitionMap(current, pages, callback)

Funkcije tranzicijske mape dobivaju sve stranice na raspolaganje jer ih sve i trebaju prikazati te omogućiti odabir željene stranice.

2.2. Objašnjenje funkcija tranzicija i tranzicijskih mapa

U funkciju prije tranzicije najbolje je postaviti svu inicijalizacijsku logiku, te osigurati da se povratna funkcija pozove tek kada su svi željeni resursi spremni.

U funkciji tranzicije se definira sama tranzicija, najčešće je to canvas element preko cijelog ekrana s trodimenzionalnim prikazom pomoću Three.js renderer-a. Unutar te funkcije ide logika animacije tranzicije. Kada animacije završe pozvati povratnu funkciju. U slučaju tranzicijske mape potrebno je ostvariti način odabira sljedeće stranice, te je proslijediti u funkciju poslije tranzicije.

U funkciji poslije tranzicije potrebno je počistiti memoriju nakon tranzicije, radi boljeg rada stranice. Nakon čišćenja potrebno je pozvati povratnu funkciju. U slučaju tranzicijske mape povratna funkcija se treba pozvati s argumentom stranice na koju se prelazi.

3. Izrada tranzicije

Izrade tranzicija u radu prikazat će se na dva primjera, jednostavne tranzicije s kockom, te složenije tranzicije s manipuliranjem ploča.

3.1. Jednostavna tranzicija s kockom

Jednostavna tranzicija s kockom je tranzicija u kojoj se trenutna i sljedeća stranica izmjenjuju kao dvije strane kocke u trodimenzionalnom prostoru.

Sl. 3.1 Krajnji rezultat jednostavne tranzicije

3.1.1. Funkcija prije tranzicije

U funkciji prije tranzicije se radi inicijalizacijska logika. Korištenjem Three.js biblioteke potrebno je stvoriti prikaz trodimenzionalnog prostora te iscrtati kocku sa slikama stranica.

Pozadina tranzicije može biti bilo koja slika kao CSS svojstvo pozadine na samom tijelu dokumenta.

Za bilo koji trodimenzionalan prikaz uz pomoć Three.js biblioteke potrebne su 3 stvari:

- Iscrtavač (renderer) – Način na koji Three.js biblioteka iscrtava scenu
- Scena (scene) – Trodimenzionalni prostor u kojoj svi elementi postoje, iscrtava se pomoću iscrtavača
- Kamera (camera) – Pogled u trodimenzionalnu scenu

Svaki Three.js objekt se instancira pomoću globalnog objekta THREE. Three.js podržava više različitih iscrtavača, ali u ovaj rad se fokusira samo na WebGL iscrtavač. Za instanciranje iscrtavača koristi se funkcija *THREE.WebGLRenderer()*. Kao argument funkcije mogu se predati različite opcije u obliku Javascript objekta. Za ovu tranziciju potrebna je opcija *alpha* koja osigurava da je pozadina iscrtavača prozirna, tako da se vidi pozadina na tijelu dokumenta.

Također potrebno je postaviti veličinu iscrtavača na ekranu, koja se može napraviti pozivanjem funkcije *renderer.setSize(width, height).*

Kako bi mogli vidjeti iscrtavanje uz pomoć WebGL tehnologije potrebno je imati canvas element. Three.js to olakšava jer inicijalizacijom WebGLRenderera stavlja u *renderer.domElement* canvas element koji sadrži iscrtavač, pa ga lako dodamo u DOM ispod tijela dokumenta.

Scena se instancira pomoću funkcije *THREE.Scene().*

U Three.js biblioteci postoje dvije glavne vrste kamera, ortogonalne i perspektivne. U ovoj tranziciji koristit će se perspektivna kamera. Kamera se instancira pomoću funkcije *THREE.PerspectiveCamera(fov, ratio, near, far).* Potrebna su 4 argumenta da se definira kamera:

1. Vertikalni kut gledanja koji kamera pokriva
2. Omjer širine i visine, najčešće potrebno staviti kao omjer širine i visine iscrtavača
3. Najmanja udaljenost od kamere koja je potrebna da objekt bude iscrtan
4. Najveća udaljenost od kamere koja je potrebna da objekt bude iscrtan

Za potrebe ove tranzicije vertikalni kut gledanja treba biti 45°, omjer širine i visine treba biti omjer širine i visine iscrtavača, jer iscrtavač treba pokrivati cijelu stranicu, a najmanja i najveća udaljenost se mogu mijenjati pomoću opcija tranzicije.

Definicija osnovnog trodimenzionalnog prostora pomoću Three.js biblioteke:

```
var renderer = new THREE.WebGLRenderer({
  alpha: true
});
var renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);

var camera = new THREE.PerspectiveCamera(45, window.innerWidth /
window.innerHeight, options.near, options.far);

var scene = new THREE.Scene();
```

Za tranziciju potrebno je i iscrtati kocku sa slikama stranica na stranicama kocke.

Za kreiranje kocke potrebno je prvo stvoriti geometriju kocke. Geometrija u Three.js biblioteci je bilo koji skup točaka, linija i ploha u prostoru koja čini neki element. Za kreiranje geometrije kocke koristi se funkcija *THREE.BoxGeometry(width, height, depth).* Argumenti funkcije su širina, visina i dubina kocke.

Također potrebno je pripremiti materijal za kocku. Postoji više različitih materijala unutar Three.js biblioteke koji se razlikuju po načinu iscrtavanja. Za potrebe ovog rada koristit će se MeshLambert materijal. Prvo je potrebno načiniti zasebne materijale stranica, pa ih sve zajedno spojiti u zajednički materijal koji se može postaviti na kocku.

Materijal jedne stranice instancira se pomoću funkcije *THREE.MeshLambertMaterial().* Funkcija prihvaća opciju *map* gdje se predaje slika. Za učitavanje slike mogu se koristiti pomoćne funkcije od Three.js biblioteke *THREE.ImageUtils.loadTexture()*, koja je u mogućnosti prihvatiti sliku enkodiranu u bazi 64. Također moguće je predati opciju *color* koja će stvoriti materijal koji je samo obojan jednom bojom.

Nakon stvaranja svih potrebnih materijala potrebno ih je zajedno spojiti da se stvori materijal za kocku. Korištenjem Three.js funkcije T*HREE.MeshFaceMaterial(materials).* Kao argument predaje se Javascript niz koji sadrži sve materijale koje će se koristiti za svako lice kocke.

Završno, za kreiranje kocke od njezine geometrije i materijala koristi se funkcija *THREE.Mesh(geometry, material).* Kao argumenti se predaju geometrija kocke i materijal

kocke. Da bi tu kocku mogli vidjeti u trodimenzionalnom prostoru potrebno ju je dodati u scenu, što se čini naredbom *scene.add(element).*

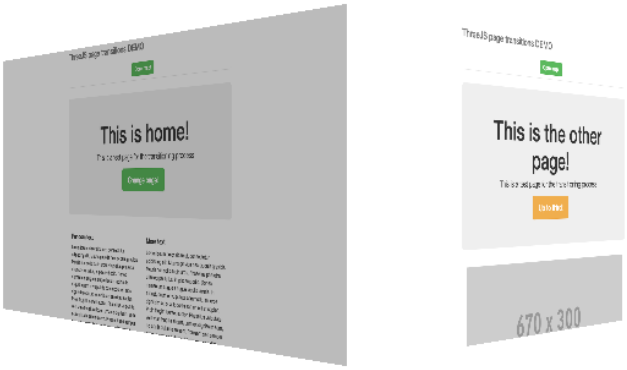
Inicijalizacija kocke:

```
var boxGeometry = new THREE.BoxGeometry(width, height, depth)

var materialCurrent = new THREE.MeshLambertMaterial({
  map: THREE.ImageUtils.loadTexture(current.image)
});
var materialNext = new THREE.MeshLambertMaterial({
  map: THREE.ImageUtils.loadTexture(next.image)
});
var materialEmpty = new THREE.MeshLambertMaterial({
  color: 0xFFFFFF
});

var materials = [materialNext, materialNext, materialEmpty,
materialEmpty, materialCurrent, materialCurrent];

var boxMaterial = new THREE.MeshFaceMaterial( materials )

var box = new THREE.Mesh(boxGeometry, boxMaterial);

scene.add(box);
```

Da bi se kocka vidjela u trodimenzionalnom prostoru potrebno je dodati i svjetla. Postoji više različitih svjetla u Three.js biblioteci, koristit će s ambijentalno i usmjereno svjetlo.

Ambijentalno svjetlo osvjetljava svaki element sa svih strana podjednako, instancira se funkcijom *THREE.AmbientLight(color).* Argument koji prihvaća je boja svjetla. Svjetla se kao bilo koji drugi element scene treba dodati na scenu da bi bilo vidljivo u iscrtavaču. To se radi jednako kao iza kocku funkcijom *scene.add(element).*

Usmjereno svjetlo osvjetljava elemente s određene strane, a dodaje se funkcijom *THREE.DirectionalLight(color).* Mijenjanjem pozicije usmjerenog svijetla odabiremo odakle svjetlo dolazi. To činimo funkcijom *directionalLight.position.set(x, y, z).normalize()*. Svaki element na sceni sadrži objekt position i rotation koji sadrže poziciju i rotaciju objekta u sceni. Koristeći funkciju *set* tog objekta moguće je postaviti svjetlo na određenu poziciju. Funkcija *normalize* se koristi da vektor pozicije svjetla bude normaliziran, jer je za usmjereno svjetlo nebitno koliko daleko se nalazi, već samo kut na kojem je od pozicije 0,0,0. Kao i ambijentalno svjetlo i kocku potrebno ga je dodati u scenu.



Sl. 3.2 Kocka u trodimenzionalnom prostoru sa stranicama

Sve što je ostalo od inicijalizacijske logike je pripremiti animacijsku logiku.

Animacija se izvodi pomoću vanjske biblioteke Tween.js. Tween.js je jednostavna biblioteka koja pomaže u izvođenju interpolacijskih animacija. Zahtjeva definiciju početnog i krajnjeg stanja i odabiranje načina interpolacije. Nakon toga u animacijskoj logici se samo pokreće animacija i u svakom trenutku biblioteka vraća trenutno stanje interpolacije.

Početno i završno stanje su bilo koji Javascript objekti, npr:

```
var startState = { x: 0 }
var endState = { x: 100 }
```

Zatim je potrebno kreirati tween objekt uz pomoć globalnog objekta TWEEN. Za kreiranje jednog tween objekta koristi se funkcija *TWEEN.Tween(startState).to(endState, time).* Argument *Tween* funkcije je početno stanje, a *to* funkcije je završno stanje i vrijeme koliko će animacija trajati. Da bi kocka u tranziciji mogla reagirati na promjene u animaciju potrebno se povratnom funkcijom pretplatiti na tween objekt, uz pomoć funkcije *tween.onUpdate(callback).* Argument funkcije je povratna funkcija koja će se pozvati na svakom koraku animacije.

Također može se postaviti način interpolacije uz pomoć funkcije *tween.easing(easing)* , u slučaju jednostavne tranzicije s kockom koristi se *TWEEN.Easing.Cubic.InOut*.

Primjer animacije za jednostavnu tranziciju kocke:

```
var begin = { cameraZ: camera.position.z, boxRotationY: 0 }
var end = { cameraZ: -200, boxRotationY: Math.PI * (3/2) };

tween = new TWEEN.Tween(begin).to(end, 100); //in ms
tween.onUpdate(function(){
  camera.position.z = this.cameraZ;
  box.rotation.y = this.boxRotationY;
});

tween.easing(TWEEN.Easing.Cubic.Out);
```

Za početnu i krajnju poziciju kamere i kocke bitno je da se ne primijeti razlika između iscrtane HTML stranice i trodimenzionalnog prostora. To se postiže pozicioniranjem kamere tako da jedna stranica (trenutna ili sljedeća) potpuno zauzima cijelo vidno polje. Pozicija se može trigonometrijski izračunati poznavajući vertikalni kut gledanja kamere, visine i širine kocke i vidnog polja. Na taj način je udaljenost kamere od sredine kocke:

```
Math.tan(67.5 * Math.PI/180) * (boxHeight/2) + (boxWidth/2);
```

Nakon što je sve definirano, poziva se povratna funkcija funkcije prije tranzicije.

3.1.2. Funkcija tranzicije

Za animaciju na web-u koristeći Javascript koristi se funkcija *requestAnimationFrame(animateFunction).* Povratna funkcija u argumentu će se u teoriji pozivati svakih 16ms (ili 60 puta u sekundi), dok stvarno vrijeme pozivanja ovisi o implementaciji u pregledniku koji se pokreće, zahtjevnosti aplikacije i više drugih faktora.

Za animacijsku logiku jednostavne tranzicije s kocke unutar povratne funkcije pozivat će se funkcija iscrtavača i funkcija osvježavanja Tween.js-a.

Funkcija iscrtavanja poziva se sa *renderer.render(scene, camera).* Potrebno ju je pozvati na svakom koraku da se promjene vide na iscrtanom canvas elementu.

Funkcija osvježavanja poziva se sa *TWEEN.update().* Funkciju je također potrebno pozvati na svakom koraku da bi animacije bile glatke. Funkcija osvježavanja će napredovati svaku animaciju interpolacije za vrijeme koliko je prošlo od zadnjeg koraka.

Na kraju, na animaciju interpolacije, potrebno se pretplatiti na završetak animacije funkcijom *tween.onComplete(completeFunction),* čiji se argument povratne funkcije poziva na kraju interpolacijske animacije. U toj povratnoj funkciji se poziva povratna funkcija funkcije tranzicije, jer je tu kraj tranzicije.

3.1.3. Funkcija poslije tranzicije

Funkcija poslije tranzicije najjednostavnija je od funkcija tranzicije, ona samo treba počistiti sve elemente kreirane funkcijama tranzicije. Potrebno je maknuti canvas element iscrtavača, vratiti pozadinu tijela dokumenta na onu prije tranzicije te pozvati povratnu funkciju funkcije poslije tranzicije.

3.2. Složena tranzicija s manipuliranjem ploča

Složena tranzicija s manipuliranjem ploča je tranzicija u kojoj se trenutna i sljedeća stranica izmjenjuju tako da prva stranica, manipuliranjem točaka na ploči postane kao tekuća ploha te sljedeća stranica izroni iz nje.

Sl. 3.3 Krajnji rezultat složene tranzicije

3.2.1. Funkcija prije tranzicije

Kao i za jednostavnu tranziciju kocke prvo je potrebno inicijalizirati standardnu trodimenzionalnu scenu u Three.js biblioteci. Canvas element iscrtavača također ide preko cijelog ekrana i koristi se isti tip kamere.

Za kreiranje ploča potrebno nam je, kao i za kocku, geometrija ploče te materijal za ploču.

Pošto je ploča vrlo jednostavna geometrija materijal se može direktno koristiti Lambertov model. Instancira se pomoću funkcije *THREE.MeshLambertMaterial()* s opcijom *map* u koju postavljamo *THREE.ImageUtils.loadTexture(image).*

Za kreiranje geometrije ploče koristimo Three.js funkciju *THREE.PlaneGeometry(width, height, widthSegments, heightSegments).* Uz standardne argumente širine i visine predajemo i argumente za broj visinskih i širinskih segmenata, jer će se ploča morati deformirati.

Također za geometriju koju stvorimo moramo postaviti zastavicu *dynamic* kako bi Three.js biblioteka znala da će se geometrija mijenjati kroz vrijeme.

Kreiranje samog objekta ploče koristi se ista funkcija kao i za kocku – *THREE.Mesh(geometry, material).*

Na kraju kao i uvijek potrebno je element dodati na scenu uz pomoć *scene.add(element).*

Potrebno je inicijalizirati ambijentalno svjetlo, na isti način kao i za kocku, dok usmjereno svjetlo nije potrebno jer neće imati nikakvog različitog utjecaja na ploču od ambijentalnog.

Za inicijalizaciju animacijske logike potrebno je definirati animacije interpolacije sljedeće stranice, tako da dođe iz pozadine ispred trenutne stranice.

Nakon svega kao i za tranziciju kocke potrebno je pozvati povratnu funkciju funkcije prije tranzicije.

3.2.2. Funkcija tranzicije

Javascript funkcija *requestAnimationFrame(animateFunction(time))* povratnoj funkciji također vraća i trenutno vrijeme pozivanja. To vrijeme možemo koristiti za izradu vlastite animacije ploče. Ukoliko svako kretanje točaka ploče pomnožimo sa vremenom koje je prošle od zadnjeg koraka animacije, animacija će biti stabilna kroz vrijeme bez obzira na fluktuacije poziva povratne funkcije animacije.

Funkcija animacije u sebi mora pozvati osvježenje animacije interpolacije – funkcijom *TWEEN.update()*, i pozvati *render* funkciju iscrtavača.

Također u funkciji animacije trebamo odrediti novu poziciju točaka ploče. Pošto smo geometriju ploče definirali s više visinskih i širinskih segmenata, znamo da element sadrži određen broj točaka (vertices). Koristeći vanjsku biblioteku Lodash, možemo iterirati kroz sve točke geometrije, koristeći funkciju *_.each(list, function(element, index))*, te tako obaviti operaciju kretanja na svakoj točki geometrije. Pristup točkama geometrije obavlja se preko objekta vertices na objektu geometrije ploče.

Funkcija pokretanja točaka ploče u obliku valova:

```
_.each(plane.geometry.vertices, function(vertex, index) {
  vertex.z = plane.position.z + (Math.sin(time/1000 + (index % 
segments))/2 + 1) * multiplier;
});
```

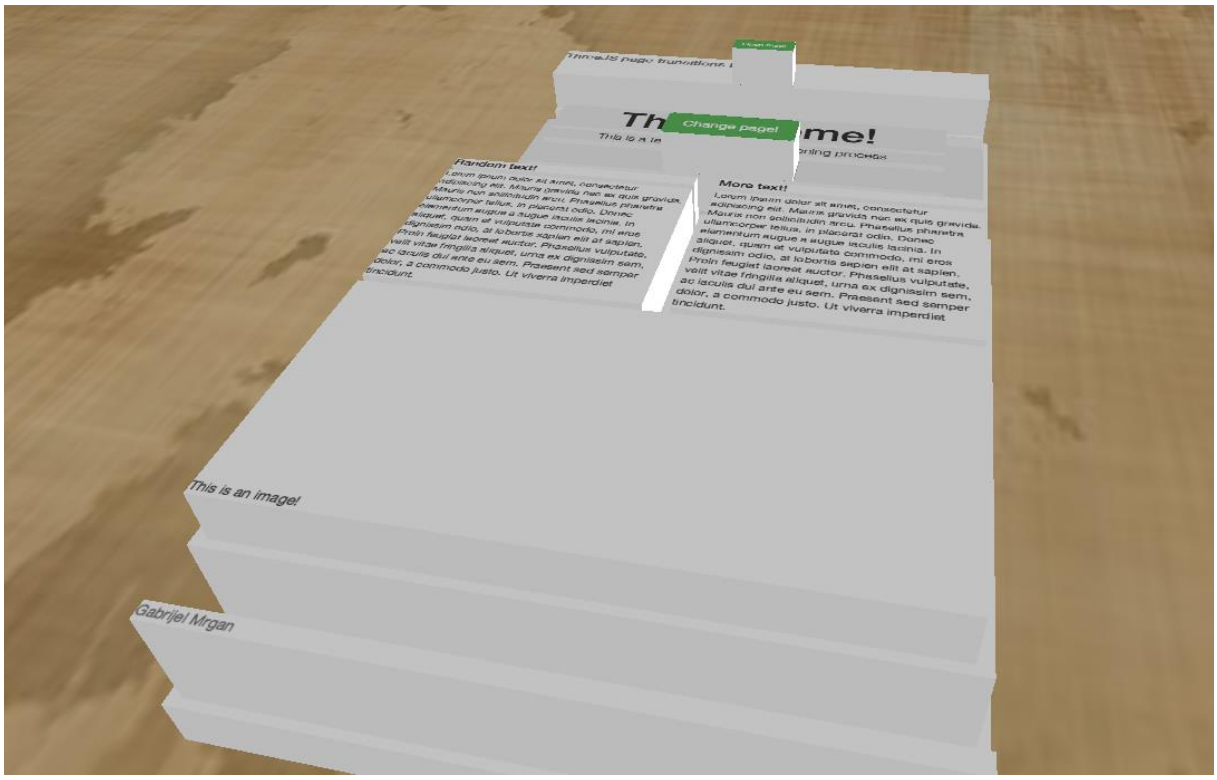
Bitno je za primijetiti da se vrijeme koristi unutar matematičke funkcije sinus, koja nam daje sinusoidalno kretanje, množi se amplitudom vala te se dodaje početna pozicija ploče. Zbog korištenja vremena u funkciji sinusa sigurni smo da i ako se funkcija animacije pozove ranije ili kasnije od željenog (60 puta u sekundi), animacija će se odvijati normalno.

Na kraju, potrebno se preplatiti na kraj animacije interpolacije, kada se sljedeća stranica zamijeni prethodnom, i u tom trenutku pozvati povratnu funkciju funkcije tranzicije.

3.2.3. Funkcija poslije tranzicije

U funkciji poslije tranzicije potrebno je maknuti canvas element iscrtavača, te očistiti memoriju od svih elemenata korištenih u sceni. Na kraju pozvati povratnu funkciju funkcije poslije tranzicije.

3.3. Ostale tranzicije



Sl. 3.4 Tranzicija koja manipulira svakim elementom stranice

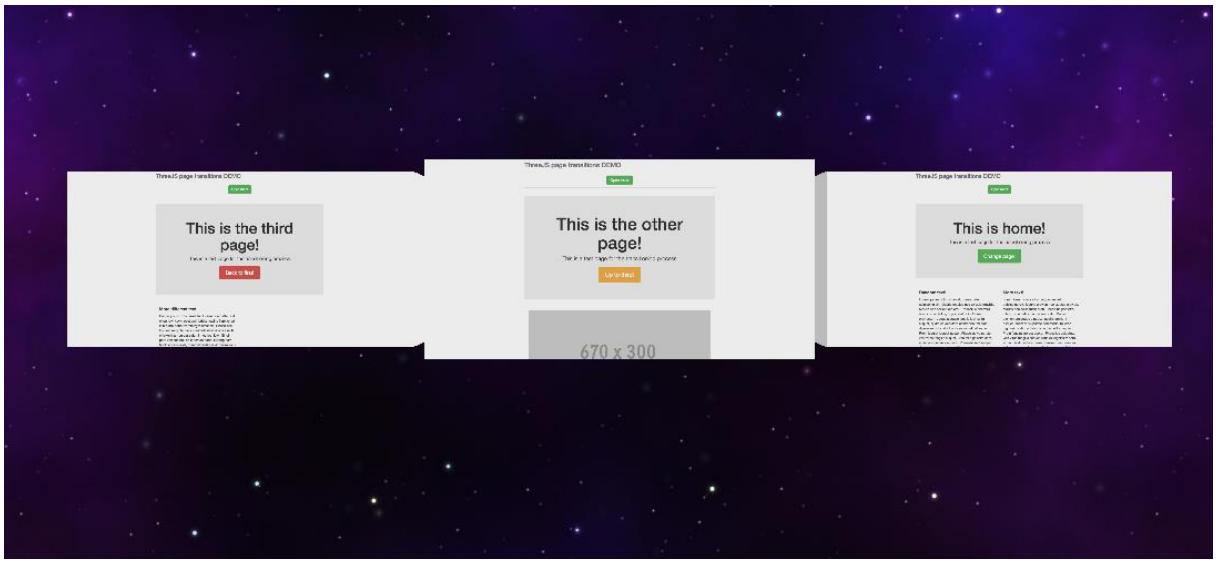
Za razvojni okvir moguće je napraviti različite tranzicije korištenjem tehnika opisanih ranije. Pošto funkcije tranzicije imaju pristup svim elementima stranice, moguće je napraviti tranzicije koje manipuliraju svakim elementom na stranici kao trodimenzionalnim objektom, kao na slici 3.4.

4. Izrada tranzicijske mape

Tranzicijske mape za razliku od tranzicija moraju podržavati odabir sljedeće stranice. Pri završetku funkcije poslije tranzicije moraju povratnoj funkciji vratiti ime stranice koja je odabrana. U ovom radu opisat će se jednostavna tranzicijska mapa s kockama.

4.1. Tranzicijska mapa s kockama

Tranzicijska mapa s kockama je tranzicijska mapa koja će svakoj kocki pridodijeliti stranicu, te će na klik miša odabrati željenu stranicu.



Sl. 4.1 Krajnji rezultat tranzicijske mape s kockama

4.1.1. Funkcija prije tranzicije

U funkciji prije tranzicije za tranzicijske mape se, kao i za tranzicije, obavlja inicijalizacijska logika. Za razliku od tranzicija, tranzicijske mape dobivaju sve stranice na koje se može doći, pa je potrebno definirati materijale i kocke za svaku stranicu. Ostatak inicijalizacije – iscrtavač, scena i kamera su iste kao i kod tranzicija.

Da bi se omogućila logika odabiranja kocki potrebno je odrediti gdje se nalazi miš na stranici, te da li je pozicija miša iznad neke kocke u trodimenzionalnom prostoru.

Kako bi znali gdje se miš nalazi, moguće se preplatiti na Javascript događaj *onMouseMove(),* koji će kao argument povratnoj funkciji predati objekt sa informacijama o poziciji miša na ekranu. Kako je miš nad canvas elementom dvodimenzionalna pozicija, da bi provjerili da li je miš iznad nekog elementa u trodimenzionalnom prostoru moramo iskoristiti tehniku bacanja zraka.

Three.js nam nudi funkciju *THREE.Raycaster(),* kojom možemo instancirati bacač zraka za trodimenzionalni prostor. Da bi provjerili da je miš iznad nekog elementa, bacamo zraku od početne pozicije na ekranu prema dubini u prostor. To bi trebalo činiti na svakom pozivu povratne funkcije pokretanja miša, ali pošto je bacanje zraka prilično skupa funkcija, pozivat ćemo je samo u kraćim periodima od 100-200ms.

Kako bi iskoristili bacač zraka koristimo funkciju *raycaster.setFromCamera(mouse, camera)* koja će stvoriti zraku od trenutne pozicije miša na ekranu, zatim za provjeru da li je zraka prošla kroz neki objekt koristimo funkciju *raycaster.intersectObjects(elements)* kojoj predajemo listu svih objekata. Funkcija vraća listu svih pozicije gdje je zraka prošla kroz dijelove objekata.

Prolazeći kroz listu svih pozicija moguće je odrediti iznad koje kocke se miš nalazi, kojoj u tom trenutku dodajemo pripadnu animaciju izlaženja pred kameru, kako bi korisnik vidio da je ta kocka klikabilna.

Kako bi tranzicijska map reagirala na klik, potrebno se pretplatiti na još jedan Javascript događaj – *onMouseDown().* U povratnoj funkciji tog događaja potrebno je opet provjeriti uz pomoć bacača zraka gdje se miš nalazi, a ako se nalazi iznad kocke uključiti animaciju prilaženja kocke kameri, s ciljem završetka tranzicije. Animacije prilaženja te prilikom prelaska miša mogu se lagano definirati korištenjem Tween.js biblioteke.

4.1.2. Funkcija tranzicije

U funkciji tranzicije potrebno je samo, kao i prije, osvježavati animacije interpolacije, iscrtavati svaki korak te čekati na kraj animacije prilaženja, u kojem slučaju se bilježi koja kocka je označena. Nakon što animacija prilaženja završi, poziva s povratna funkcija tranzicije.

4.1.3. Funkcija poslije tranzicije

U funkciji poslije tranzicije, uz pomoć zabilježene označene kocke se određuje koja stranica je bila odabrana od strane korisnika. Najjednostavniji način za to je da se uz svaku kocku u objekt *THREE.Mesh()* upiše i ime stranice na koju se odnosi, pa se u funkciji poslije tranzicije samo učita to ime. Također se obavlja standardno brisanje canvas elementa iz DOM-a, te čišćenje korištenih resursa. Pozivom povratne funkcije poslije tranzicije sa argumentom odabrane stranice završava tranzicijska mapa.

Zaključak

U sklopu ovog rada pokazalo se kako napraviti prijelaz između stvarne HTML stranice i trodimenzionalnog prostora na canvas elementu uz pomoć WebGL tehnologije preko biblioteke Three.js. Three.js je vrlo jednostavna biblioteka koja izrazito pojednostavljuje WebGL tehnologiju, te pruža više korisnih pomoćnih funkcionalnosti koje se ne mogu očekivati od WebGL tehnologije.

WebGL tehnologija je u vrijeme pisanja ovog rada još uvijek nova tehnologija na internetu. Performanse su nesigurne, i izrazito zavise od internet preglednika u kojima se stranica izvodi. Biblioteke poput Three.js omogućavaju da se prikaz ujednači između preglednika, te značajno olakšavaju postupak razvijanja trodimenzionalnog prikaza unutar internetskog preglednika. S napretkom Javascript-a i WebGL tehnologije može se očekivati da će internetske stranice koje koriste WebGL tehnologiju biti u porastu.

Literatura

[1] NIKLAS VON HERTZEN, s Interneta, http://html2canvas.hertzen.com/documentation.html,

[2] RICARDO CABELLO, s Interneta, http://threejs.org/docs/,

[3] JOHN-DAVID DALTON, s Interneta, https://lodash.com/docs,

[4] SOLE, s Interneta, https://github.com/tweenjs/tween.js/,

[5] JOHN RESIG, s Interneta, http://api.jquery.com/

UPORABA TRODIMENZIJSKIH PRIKAZA U INTERNET PREGLEDNICIMA

Sažetak:

Rad se bavi uporabom trodimenzijskih prikaza u internet preglednicima na primjeru korištenja Three.js, Tween.js, JQuery i Lodash biblioteke za izradu Javascript razvojnog okvira za trodimenzionalne tranzicije između HTML stranica. U radu je opisana izrada takvog razvojnog okvira, te izrada trodimenzionalnih tranzicija i tranzicijskih mapa – trodimenzionalnog prikaza gdje se odabire nova stranica. U radu je pojašnjeno korištenje Three.js i Tween.js biblioteke za kreiranje trodimenzionalnih prostora te njihovo animiranje.

Ključne riječi:

Trodimenzionalni prikaz, web, internet, Three.js, Tween.js, HTML, Javascript, biblioteka, razvojni okvir, tranzicija, web stranica, WebGL.

APPLICATION OF THREE-DIMENSIONAL VISUALIZATION IN WEB BROWSERS

Summary:

The paper deals with making of a Javascript framework for three dimensional HTML page transitions using the Three.js, Tween.js, JQuery and Lodash libraries. In the paper it is explained how to design and create such a library, and how to design and create the transitions and transition maps for it. A transition map is a three dimensional representation of a sitemap. The paper explains how to use Three.js and Tween.js libraries to create three dimensional environments, and animate them.

Keywords:

Three dimensional view, web, internet, Three.js, Tween.js, HTML, Javascript, library, framework, transition, webpage, WebGL.