UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matej Puntar

3D na internetu

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Saša Divjak

Ljubljana, 2012

Rezultati diplomskega dela so intelektualna lastnina Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01876/2012
Datum: 05.10.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATEJ PUNTAR**

Naslov: **3D NA INTERNETU**
3D ON INTERNET

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Predstavite uveljavljene in nove tehnologije za prikazovanje 3D vsebin v spletnem brskalniku, njihov razvoj ter prednosti in pomanjkljivosti posameznih tehnologij. Poseben poudarek naj bo na najnovejši tehnologiji WebGL, ki omogoča prikazovanje 3D vsebin v brskalnikih. Opišite njeno delovanje in praktično uporabnost. Podajte tudi njene pomanjkljivosti. Kot konkreten primer uporabe 3D vsebin v spletnem brskalniku razvijte večigralsko spletno 3D igro, kar omogoča uporabo tehnologij WebGL in WebSocket.

Mentor:

prof. dr. Saša Divjak

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Matej Puntar,

z vpisno številko 63000252,

sem avtor diplomskega dela z naslovom:

3D na internetu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  prof. dr. Saša Divjak
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 10.10.2010 Podpis avtorja:

Zahvala

Rad bi se zahvalil staršem, ki so mi omogočili študij in me ves čas spodbujali z neskončno mero potrpljenja.

Hvala prof. dr. Saši Divjaku za mentorstvo in pomoč pri diplomskem delu ter vedno prijazen odnos.

Zahvaljujem se tudi cimrom in kolegom, s katerimi sem študiral. Brez vas čas študija ne bi bil tako pester in zabaven.

Kazalo

Seznam uporabljenih kratic in simbolov

3D - Tridimenzionalno
API - Application Programming Interface - programski vmesnik.
DOM - Document Object Model - objektni model dokumenta
DoS - Denial-of-Service - ohromitev storitve
FPS - Frames per second - število slik na sekundo.
GPU - Graphics Processing Unit - grafična procesna enota.
HTML5 - HyperText Markup Language 5 - označevalni jezik za hipertekst
HTTP - HyperText Transfer Protocol - protokol za prenos hiperteksta
ISO - International Standards Organization - mednarodna organizacija standardov
JNLP - Java Network Launching Protocol - protokol za mrežni zagon java aplikacij
NURBS - Non-Uniform Rational B-Splines - neenotne racionalne orisne krivulje
OpenGL ES - OpenGL for Embedded Systems - OpenGL za vgradne sisteme
VRML - Virtual Reality Modelling Language - jezik za modeliranje navidezne resničnosti
WebGL - Web Graphics Library - knjižnica za spletno grafiko
WebSocket - spletna vtičnica
X3D - Extensible 3D Graphics - razširljiva 3D grafika
XML - eXtensible Markup Language - razširljivi označevalni jezik.

Povzetek

Namen diplomske naloge je predstavitev že uveljavljenih in novih tehnologij za prikazovanje 3D vsebin v spletnem brskalniku.

V začetku je predstavljena kratka zgodovina 3D vsebin na internetu in njihov razvoj ter prednosti in pomanjkljivosti posameznih tehnologij. Te so podrobno predstavljene, tako njihova uporaba kot razlike med njimi.

Poseben poudarek je namenjen WebGL, najnovejši tehnologiji prikazovanja 3D vsebin v brskalnikih. Natančno je prikazano njeno delovanje in praktična uporaba. Kot vsaka tehnologija ima tudi WebGL nekaj pomanjkljivosti, ki so predstavljene v diplomskem delu.

Praktična uporaba 3D vsebin v spletnem brskalniku je prikazana v napredni večigralski spletni 3D igri z uporabo tehnologij WebGL in WebSocket.

Ključne besede:

3D, internet, VRML, X3D, XML, HTML, HTML5, WebGL, JavaScript, WebSocket

Abstract

The purpose of this thesis is the presentation of already established and new technologies of displaying 3D content in a web browser.

The thesis begins with a short presentation of the history of 3D content available on the internet and its development together with advantages and disadvantages of individual technologies. The latter two are described in detail as well is their use and the differences among them.

Special emphasis has been given to WebGL, the newest technology of 3D content presentation in browsers. The technology is presented in detail and so is its use. Like any other technology WebGL, too, has its own disadvantages which are presented in this thesis.

Practical use of 3D content in a web browser is presented through an advanced multiplayer 3D game developed with WebGL and WebSocket technology.

Key words:

3D, internet, VRML, X3D, XML, HTML, HTML5, WebGL, JavaScript, WebSocket

Poglavje 1

Uvod

Interaktivni prikaz 3D objektov na spletnih straneh je bil še pred kratkim zelo redek pojav. Že od začetka interneta se ljudje trudijo, da bi prikazali 3D elemente na internetu, toda do sedaj je bila tehnologija za prikazovanje takšnih elementov zelo omejena in v večini primerov tudi zelo počasna.

Uporaba interneta in s tem tudi uporaba spletnih brskalnikov se je razširila v sredini devetdesetih. V tem času je bil najpopularnejši brskalnik Netscape Navigator z več kot 50% tržnim deležem. Pri Microsoftu so pomembnost interneta in spletnih brskalnikov prepoznali zelo pozno. Da bi dohiteli Netscape, so začeli razvijati svoj brskalnik.

Iz zgodovine vemo, da je tehnološki razvoj najhitrejši v času vojn in to velja tudi za spletne brskalnike. Vojna brskalnikov se je začela leta 1995, ko je Microsoft izdelal svoj prvi brskalnik Internet Explorer 1.0 in ga uporabnikom ponudil kot Internet Jumpstart Kit v paketu Microsoft Plus! za Windows 95 [2].

Pri Microsoftu so se odločili, da bodo premagali Netscape in izdelali najbolj razširjeni spletni brskalnik. Da mislijo resno, je bilo kmalu očitno, saj so že 3 mesece kasneje v novembru 1995 izdali brezplačni Internet Explorer 2.0, ki je med drugim prinesel podporo za piškotke in podporo za vsebine VRML ter je deloval na operacijskih sistemih Windows in Mac OS. Za razliko od Netscape Navigatorja je bil Internet Explorer brezplačen, toda to mu ni prineslo večjega tržnega deleža, ker je bil glede na zmožnosti konkurenta še vedno primitiven. [3]

Z verzijo Internet Explorerja 3.0, ki je izšla avgusta 1996, je Microsoft dosegel kvaliteto in zmogljivosti konkurenta in začel postajati vse bolj popularen. V začetku leta 1996 je imel Netscape Navigator po nekaterih anketah skoraj 90% tržni delež, ob koncu leta je ta procent padel na malo več kot 80%.

Internet Explorer mu je začel odžirati tržni delež.

V drugi polovici leta 1997 je izšel Internet Explorer 4.0, ki je bil integriran v Windows operacijski sistem. [3]

Vojna brskalnikov je prinesla hiter napredek v tehnologijah za prikazovanje spletnih strani in tehnologije, kot so: CSS, JavaScript, piškotki, okvirji in tudi VRML, ki je omogočal prikazovanje 3D elementov v brskalniku.

S pojavom vtičnikov za brskalnike se je stanje izboljšalo. 3D objekti se sedaj lahko prikažejo s Flash, Java applet, Silverlight in drugimi vtičniki. Toda vsi ti vtičniki so še vedno procesorsko zelo zahtevni in od uporabnika zahtevajo namestitev vtičnika za vsako tehnologijo posebej.

S popularizacijo in razvojem standarda HTML5 vtičniki kmalu ne bodo več potrebni. V diplomi bom opisal prednosti in slabosti omenjenih vtičnikov za prikaz 3D objektov ter pokazal kako se lahko 3D objekti prikazujejo in animirajo z uporabo tehnologije WebGL.

Poglavje 2

Zgodovina

2.1 Kratka zgodovina 3D vsebin na internetu

Prikazovanje 3D vsebin v brskalniku se je prvič pojavilo leta 1994, ko sta Mark Pesce in Tony Parisi predstavila prvi osnutek standarda VRML. Standard VRML 1.0 je omogočal izdelavo statičnih 3D modelov in njihovo združevanje v kompleksnejše scene in svetove. [1]

Standard je vzbudil veliko zanimanja, saj je kmalu po njegovi predstavitvi 13 podjetij napovedalo podporo jeziku. Leta 1996 sta imela oba vodilna brskalnika Netscape Navigator 3.0 in Microsoft Internet Explorer 3.0 podporo za VRML 1.0 že vgrajeno in za prikaz vsebin VRML nista potrebovala nobenega dodatnega vtičnika. [4]

Tako kot je pogosto z novimi tehnologijami, se je tudi za VRML izkazalo, da ima nekaj večjih pomanjkljivosti. Ena glavnih je bila, da ne omogoča interakcije uporabnika s 3D objekti. To pomanjkljivost je odpravila že naslednja verzija VRML 2.0. Zaradi kompleksnosti VRML 2.0 se podpora standardu ni več vgrajevala v brskalnike. Če je uporabnik želel gledati vsebine VRML v brskalniku, si je moral namestiti vtičnik za spletni brskalnik, ki je dodal podporo za VRML.

Kljub temu, da smo dobili tehnologijo, ki je omogočala prikazovanje 3D svetov in interakcijo s 3D elementi v brskalniku, je avtorji spletnih strani niso pogosto uporabljali. Domači računalniki tistega časa niso bili kos zahtevnim izračunom za animiranje 3D svetov v brskalniku.

Kljub temu se je razvoj tehnologije za prikaz 3D vsebin na internetu nadaljeval. VRML 2.0 je bil osnova za nov standard X3D. X3D je prinesel ogromno naprednih 3D funkcij, kot so: programirljivo senčenje, osvetljevanje, materiali, vektorska 2D grafika, združevanje 2D in 3D elementov, humanoidna animacija,

preoblikovanje, izbiranje in klikanje na 3D elemente v sceni, kamere, detekcija trkov in še veliko več. Žal tudi ta standard ni postal priljubljen.

Sočasno z VRML-jem se je razvijala tudi tehnologija, imenovana Flash. Za uporabo vsebin Flash v brskalniku so si uporabniki prav tako morali namestiti vtičnik. Flash od začetka ni bil namenjen prikazovanju 3D vsebin. V svet spletnih vsebin je prinesel enostavno izdelavo in hitro nalaganje 2D animacij. Do popularizacije Flash-a so bile animirane sličice gif edini način za prikaz animacije v spletnem brskalniku. Zaradi dobrih razvojnih orodij in naprednih možnosti animacije in manipulacije vektorske in rasterske grafike je Flash kmalu postal najpopularnejši vtičnik za spletne brskalnike. Z verzijo 6 je Flash dobil skriptni jezik ActionScript in vmesnik za programiranje grafike. To je omogočilo izdelavo 3D vsebin. [13]

Ker je večina uporabnikov že imela nameščen vtičnik Flash, so brez težav lahko gledali 3D Flash vsebine. S pojavom prvih knjižnic za izdelavo in animacijo 3D elementov v Flash-u je popularnost in uporaba 3D vsebin na spletu začela naraščati.

Kot vemo, sta izrisovanje in animacija 3D elementov zelo zahtevni operaciji in skoraj vsi sodobni računalniki imajo poseben čip, namenjem le preračunavanju 2D in 3D grafičnih operacij. Vsi namizni grafični programi za hitrejšo obdelavo podatkov uporabljajo grafični procesor. Spletni brskalniki in njihovi vtičniki so začeli uporabljati grafični procesor šele pred kratkim. Flash verzija 11 je v oktobru 2011 dobil podporo za strojno pospešeno 3D grafiko. Razvijalci Flasha so novembra 2011 najavili, da ne bodo več razvijali vtičnika za mobilne brskalnike in se bodo osredotočili na namizne računalnike in razvoj za mobilne brskalnike prepustili standardu HTML5.

HTML5 je v razvoju že vrsto let in postaja zelo kompleksen standard, ki prinaša ogromno novosti, med drugim tudi WebGL, ki je programski vmesnik za strojno pospešeno 2D in 3D grafiko. To pomeni, da je sedaj 3D grafika v brskalniku tako dobra in hitra kot v namiznih programih in igrah.

Poglavje 3

Tehnologije prikazovanja 3D objektov na internetu

3.1 VRML - Virtual Reality Modeling Language

Razvoj 3D vsebin na internetu se je začel leta 1994, ko sta Mark Pesce in Tony Parisi predstavila osnutek standarda VRML na prvi mednarodni konferenci World Wide Web v Ženevi, Švica. [1]

V tistem času je bil HTML edini standard za izdelavo spletnih vsebin. HTML je bil idealni jezik za izdelavo 2D vsebin za splet, kot so besedilo in slike, toda manjkala mu je podpora za tretjo dimenzijo, zato je bil neuporaben za 3D. VRML je bil na začetku mišljen kot analogija HTML-ju za 3D vsebine, zato je kratica VRML na začetku pomenila Virtual Reality Markup Language. [1]

Markup Language, v prevodu označevalni jezik, je umetni jezik za določanje strukture in oblikovanja besedila, pri katerem so med besedilo postavljeni ukazi, predstavljeni z zaporedji navadnih znakov.

Primeri označevalnih jezikov so LaTeX, HTML (HyperText Markup Language), XML (Extensible Markup Language). Pri prikazu besedila uporabniku se označevalni ukazi ne prikazujejo, ampak se uporabijo le za izračun načina prikaza besedila.

Razvijalci VRML-ja so kmalu ugotovili, da označevalni jezik ne bo dovolj zmogljiv za vse cilje, ki so si jih zastavili pri zasnovi VRML-ja. Pomen kratice VRML se je kmalu spremenil v Virtual Reality Modeling Language. VRML je bil zasnovan, da bo deloval tudi na povprečnih računalnikih tistega časa in se

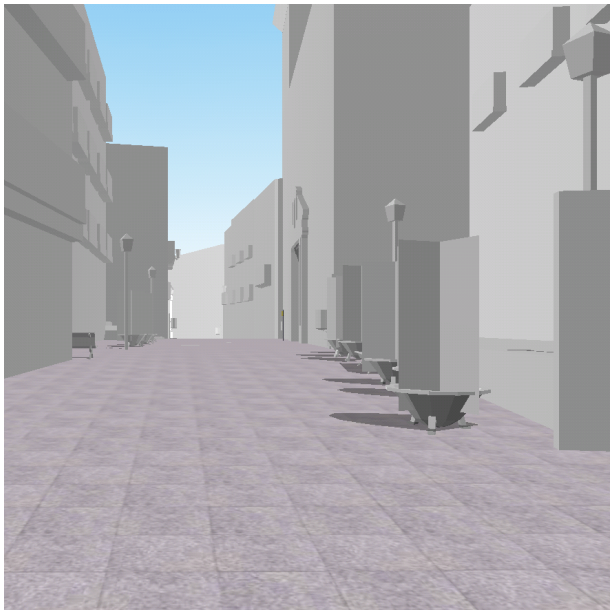
bo dovolj hitro nalagal tudi preko počasnejše povezave na internet 14.4 kbit/s.

Standard VRML 1.0 je omogočal izdelavo statičnih 3D modelov in njihovo združevanje v kompleksnejše scene in svetove. [1]

Standard je vzbudil veliko zanimanja, saj je kmalu po njegovi predstavitvi 13 podjetij napovedalo podporo jeziku. Leta 1996 sta imela oba vodilna brskalnika Netscape Navigator 3.0 in Microsoft Internet Explorer 3.0 podporo za VRML 1.0 že vgrajeno in za prikaz datotek VRML nista potrebovala dodatnega vtičnika. [5, 6]

Toda kmalu se je izkazalo, da je jezik preveč omejujoč, saj ne omogoča interakcije s 3D objekti. Zaradi omenjene pomanjkljivosti je bil razvit VRML 2.0, ki je leta 1997 uspešno prestal teste in so ga razglasili kot ISO (International Standards Organization) standard. Ker je bil kot standard proglašen leta 1997, se ga je prijelo ime VRML97. Zaradi kompleksnosti standarda VRML 2.0 se je praksa vključevanja podpore VRML v brskalnike opustila, saj je standard postal tako kompleksen, da je brskalnik VRML (brskalnik za pregledovanje le VRML datotek) postal program v velikosti 3MB. Nestcape Navigator 3.0 je bil leta 1996 velik le 3,4 MB. Podporo za prikazovanje vsebin VRML v spletnem brskalniku so prevzeli dodatni vtičniki za spletne brskalnike. Poleg velikosti brskalnikov VRML so priljubljenost vsebin VRML na internetu omejevali tudi počasni domači računalniki. V času izdelave standarda VRML 2.0 domači računalniki niso bili dovolj hitri, da bi lahko dovolj gladko prikazovali animacije 3D virtualnih svetov. Večina domačih računalnikov je komaj omogočala prikazovanje 3D animacije s hitrostjo do 8 slik na sekundo. [5]

Razvoj jezika se je nadaljeval in na osnovi VRML 2.0 je bil razvit nov standard s kratico X3D, kar pomeni Extensible 3D Graphics.



Slika 3.1: Prikaz scene VRML preden so upodobljene vse teksture

Slika 3.2: Prikaz popolnoma upodobljene scene VRML

3.2 X3D nadgradnja VRML-ja

X3D je ISO ratificiran odprt standard za prenos in prikaz 3D vsebin, ki je bil prvič predstavljen na konferenci SIGGRAPH (Special Interest Group on GRAPHics and Interactive Techniques) leta 2002. [6]

X3D je evolucija standarda VRML97 in tako prinaša kar nekaj novosti. Z evolucijo iz standarda VRML97 se nobena funkcija ni izgubila. X3D omogoča VRML 2.0 kompatibilni zapis, kar pomeni, da se večina datotek VRML 2.0, ki ne vsebujejo skriptov, lahko pretvori v X3D s samo manjšimi popravki. [6]

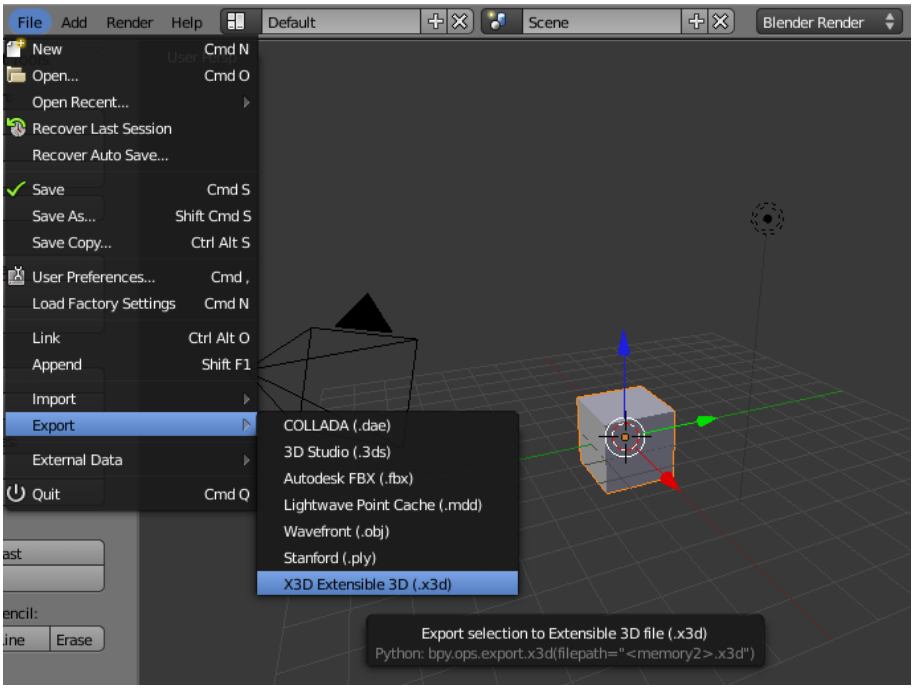
X3D ni le aplikacijski programski vmesnik niti le datotečni format za shranjevanje geometrijskih podatkov. X3D združuje geometrijske podatke in opis obnašanja pri izvajanju v eno samo datoteko in ima več različnih kodiranj datotek. [7] Vsebine X3D se lahko shranijo kot datoteka XML, klasična datoteka VRML ali stisnjena datoteka. [6]

Poleg novega zapisa podpira X3D tudi veliko naprednih 3D funkcij, kot so: programirljivo senčenje, osvetljevanje, materiali, 2D vektorska grafika, združevanje 2D in 3D elementov, krivulje NURBS, humanoidna animacija, morfanje, izbiranje in klikanje na 3D elemente v sceni, kamere, detekcija trkov in še veliko več. Zelo pomembna lastnost je tudi strojno pospeševanje, ki omogoča računanje funkcij za prikaz 3D objektov na grafični kartici, specializirani za takšne tipe problemov. Strojno pospeševanje bistveno pospeši delovanje.

Ker je X3D podobno kot VRML narejen za spletno uporabo, je možno 3D sceno sestaviti tudi iz elementov, ki jih naložimo iz različnih strežnikov na internetu. 3D objekti na sceni so lahko tudi hiperpovezave do drugih 3D scen ali drugih virov na internetu. X3D je tako za 3D na internetu to, kar je HTML za standardne dvodimenzionalne spletne strani.

Web3D consortium, organizacija, ki vodi razvoj standarda X3D, se trudi, da bi X3D vključili v standard HTML5 in bi tako bil podprt v vseh brskalnikih, ki podpirajo HTML5. To bi zelo razširilo uporabo standarda, saj za prikaz 3D elementov na spletni strani ne bi bilo treba namestiti vtičnikov za brskalnik, kot je to potrebno danes. [8]

Velika težava standarda X3D je pomanjkljiva podpora v orodjih za 3D modeliranje in animiranje. Večina orodij za 3D modeliranje omogoča izvoz scene v format VRML, ne pa tudi v X3D. Trenutno je Blender 3D, odprtokodni program za 3D modeliranje, edini, ki omogoča izvažanje v X3D format brez dodatnih vtičnikov.



Slika 3.3: Možnost izvoza v X3D iz programa za 3D modeliranje Blender.

3.3 Java in Java 3D

Java je programski jezik, ki ga je izdelal James Gosling v podjetju Sun Microsystems in je glavni del platforme Java. Java se prevede v vmesno kodo, poimenovano Java bytecode, ki se izvaja na JVM (Java virtual machine). To omogoča, da se program razvije in prevede na eni platformi in uporablja na vseh platformah, za katere obstaja JVM. Sun Microsystems je to poimenoval WORA (Write Once, Run Anywhere). [9]

Kmalu po napovedi Jave leta 1995 je podjetje Netscape, v tistem času izdelovalec vodilnega spletnega brskalnika Netscape Navigator, napovedal podporo Javi v svojem brskalniku. Brskalnik je omogočal izvajanje Java applet programov kot del spletne strani. Java je kmalu postala zelo popularna in je še danes med najpopularnejšimi programskimi jeziki na svetu. [10]

Aplikacije Java applet se uporabljajo za izdelavo naprednih interaktivnih programov za spletne strani, saj omogočajo veliko več kot le standardne spletne tehnologije, kot so HTML, CSS in JavaScript. Ker so napisane v Javi in prevedene v vmesno kodo, ki se izvaja na JVM, je njihova hitrost izvajanja primerljiva z ostalimi programskimi jeziki, ki se prevajajo v strojno kodo, kot

je C++. To je bistveno hitreje kot JavaScript, ki se izvaja v brskalniku in je programski jezik, ki se najpogosteje uporablja za spletne aplikacije in spletne strani. Aplikacije Java applet so tako hitre tudi zato, ker lahko uporabljajo 3D strojno pospeševanje, kar do pred kratkim ostale spletne tehnologije niso omogočale.

Zaradi varnostnih razlogov imajo aplikacije Java applet stroge omejitve glede dostopanja do datotek računalnika, da onemogoči zlorabe. Če teh omejitev ne bi bilo, bi lahko zlonamerna aplikacija applet skopirala poljuben podatek iz računalnika in ga posredovala zlonamernežu ali naredila še kaj hujšega.

Slabost aplikacij Java applet pred ostalimi spletnimi tehnologijami je v tem, da si mora uporabnik namestiti Java vtičnik, predno ga lahko uporablja. Podjetje Sun je zato Javo razdelil na dva dela: JRE in JDK. JDK je Java development kit, ki si ga namestijo razvijalci, ker vsebuje prevajalnik Java in še nekatere pomožne programe. JRE kratica pomeni Java runtime environment in je bistveno manjši paket, ki si ga namestijo uporabniki, ki želijo le uporabljati Java applet in standardne aplikacije Java.

Na vrhuncu popularnosti Jave je bila ta nameščena na večino novih računalnikov, ne glede na operacijski sistem (Windows, Mac OS X, Linux). Danes temu več ni tako.

Java 3D je 3D API za Java programski jezik. Java 3D API omogoča enostavnejše programiranje 3D scen in temelji na 3D tehnologijah OpenGL in Direct3D ter ne zahteva podrobnejšega poznavanja teh tehnologij. Velika prednost te tehnologije je, da omogoča hitro izvajanje na vseh platformah. Za izvajanje 3D operacij se izbere tehnologija, ki je za izbrani operacijski sistem najhitrejša. Na operacijskem sistemu Windows se 3D operacije izvajajo z Direct3D tehnologijo, na ostalih platformah z OpenGL. [11]

Za uporabo knjižnice Java 3D v brskalniku mora imeti uporabnik nameščen vtičnik Java za izbrani brskalnik in knjižnico Java 3D. Namestitev vtičnika Java za brskalnik je zelo enostavna. Ko z brskalnikom na internetu obiščemo spletno stran, ki uporablja vtičnik Java, brskalnik sam zazna manjkajoči vtičnik in uporabniku ponudi avtomatično namestitev. Če se strinjamo s pogoji uporabe in kliknemo na gumb za namestitev, bo brskalnik avtomatsko z interneta prenesel zadnjo verzijo vtičnika Java za naš brskalnik in ga namestil. Brskalnik ponovno zaženemo in spletna stran bo uspešno prikazala Java applet v našem brskalniku. Toda Java applet, ki uporablja knjižnjico Java 3D, še vedno ne bo deloval, saj Java 3D ni del standardnega JRE paketa in moramo knjižnico Java 3D namestiti posebej. Brskalnik ne zazna manjkajoče Java 3D knjižnice in zato namestitev te ni tako enostavna. Uporabnik mora sam najti in prenesti knjižnico iz uradne spletne strani in jo namestiti. Dodatni korak namestitve

knjižnice je za mnoge spletne obiskovalce pretežak. [11]

Tega so se zavedali tudi pri podjetju Sun in težavo so rešili z novo tehnologijo Java Web Start. Java Web Start je pomožna aplikacija, povezana z brskalnikom. Ko uporabnik v brskalniku klikne na povezavo, ki kaže na posebno datoteko JNLP, brskalnik zažene aplikacijo Java Web Start, ki avtomatsko prenese in lokalno shrani vse datoteke in knjižnice, potrebne za zagon aplikacije. Ves proces se izvede samodejno brez poseganja uporabnika. Za razliko od Java Appletov se te aplikacije ne izvajajo v brskalniku. Ker je aplikacija osvobojena brskalnika, ima večjo svobodo pri dostopanju do sistemskih sredstev, vendar še vedno teče v zaščitenem okolju in ima le omejen dostop do datotek in mrežnih virov. Ker so vse potrebne datoteke za zagon aplikacije po prvem zagonu shranjene lokalno, prenos le-teh ob nadaljnjih zagonih ni potreben. To pomeni, da se takšna aplikacija lahko uporablja tudi, ko uporabnik nima dostopa do interneta. [12]

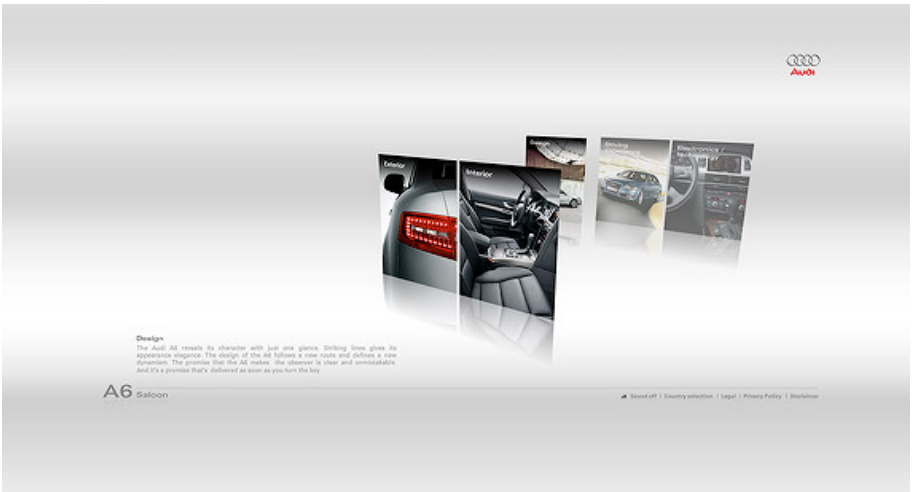
3.4 Flash

Danes najpopularnejši vtičnik za spletne brskalnike je Adobe Flash in ga lahko uporabljamo na vseh popularnih platformah, kot so Windows, Linux in Mac OS X, in tudi na manj popularnih, kot so Solaris, HP-UX, OS/2, QNX, BeOS in IRIX, v okrnjeni obliki z imenom Flash lite pa tudi na mnogih pametnih telefonih z enim od operacijskih sistemov Pocket PC, Windows CE, Symbian, Palm OS, Android. Kljub temu, da je Flash mogoče uporabljati na tako različnih platformah, najbolje deluje na operacijskem sistemu Windows. Tudi na vseh drugih platformah deluje kot pričakovano, toda za delovanje porabi veliko več procesorskega časa, kar so dokazali primerjalni testi. [14, 15]

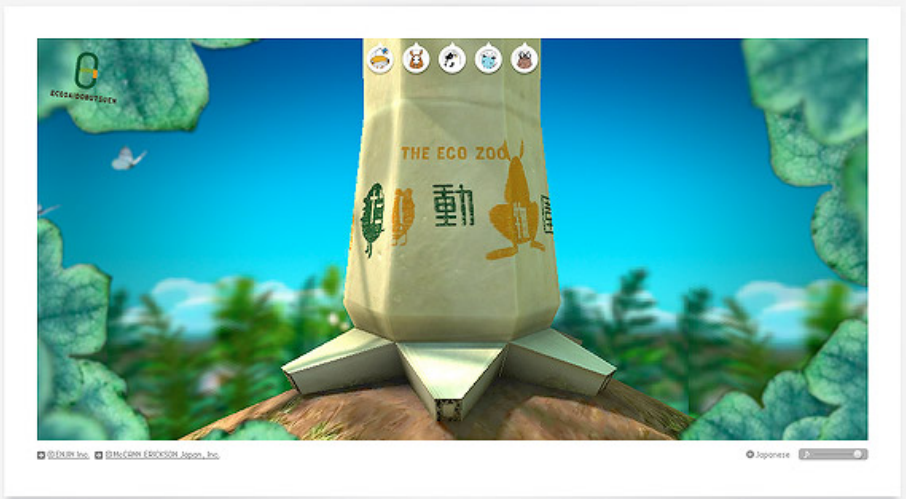
Razlika v hitrosti je očitna pri zahtevnih aplikacijah Flash kot je prikazovanje in animiranje 3D objektov.

Flash je svoj pohod na internetu začel leta 1996. FutureSplash Animator je bilo ime prve verzije vektorsko animatorskega programa za izdelavo spletnih animacij, predno ga je kupilo podjetje Macromedia. Macromedia je program preimenovala v Macromedia Flash 1. Najprej je bil Flash namenjen le animiranju vsebin na internetu in šele z verzijo Macromedia Flash 5 je dobil možnost naprednejšega programiranja z objektno orientiranim programskim jezikom ActionScript 1.0. ActionScript temelji na ECMAScript in je zato po sintaksi zelo podoben JavaScriptu. Že v tej fazi so razvijalci začeli izdelovati knjižnice za enostavno manipuliranje in prikazovanje 3D objektov. Velik napredek v prikazovanju 3D vsebin je proti koncu leta 2005 prinesel Sandy, prvi

3D pogon za Flash z vsemi pomembnimi funkcijami. [13] Ker je bil Sandy 3D pogon napisan z Actionscript 2.0, je moral imeti uporabnik za ogled 3D animacij nameščen predvajalnik Flash 7 ali novejšega. Sandy je bil kot prvi resnejši poskus 3D pogona za Flash zelo uspešen in je v svet animacij Flash prinesel svežino in novo dimenzijo inovacij. K uspešnosti pogona Sandy 3D je veliko pripomogla tudi zelo ugodna cena, saj je odprtokodni projekt in ga lahko vsakdo brezplačno uporabi za osebni ali komercialni projekt. Ko so razvijalci in uporabniki videli, da se lahko Flash uporablja za več kot le animacijo spletnih oglasov, se je razvoj knjižnic Flash razcvetel. Slabo leto po predstavitvi pogona Sandy 3D je na odprtokodno sceno 3D pogonov prišel nov tekmec z imenom Papervision3D. Glavni prednosti novega 3D pogona sta enostavna uporaba in uporaba zmogljivejšega programskega jezika Actionscript 3.0, ki ga uporablja Flash 9. Enostavnost uporabe novega 3D pogona je omogočila razvoj 3D vsebin za internet tudi spletnim oblikovalcem in ne le profesionalnim 3D razvijalcem. Zaradi razširjenosti vtičnika Flash za brskalnike in preproste uporabe pogona Papervision3D so 3D spletne strani postale bolj popularne. Ker je 3D na internetu še vedno nekaj posebnega, ga uporabljajo tudi podjetja, ki želijo imeti izstopajoče spletne strani. Med temi podjetji so tudi Coca Cola, Motorola, Audi, Orange. [16, 17, 18]



Slika 3.4: Spletna stran Audi A6 [16]



Slika 3.5: Spletna stran ecodazoo.com

Manipulacija 3D objektov in njihova animacija je računsko zelo zahtevna operacija. Flash 11 je zato oktobra 2011 dobil novo arhitekturo Stage 3D, ki omogoča strojno pospešeno upodabljanje 3D objektov. Hitrost in kvaliteta spletnih 3D animacij in iger, narejenih s Flash tehnologijo, se lahko sedaj primerja s standardnimi računalniškimi igrami in programi.

Slika 3.6: Nadgradnja Flash, ki je prinesla strojno pospešeni 3D

3.5 WebGL

Glede na to, koliko različnih tehnologij imamo na voljo za prikazovanje 3D vsebin na internetu, bi se lahko vprašali, ali je res potrebna še ena tehnologija. Ko natančneje preučimo možnosti obstoječih tehnologij, vidimo, da nobena ne ponuja vsega.

Če želimo tehnologijo, ki omogoča strojno pospeševanje, uporabo brez namestitve posebnega programa ali vtičnika, odlično kvaliteto grafike in bo delovala na namiznih računalnikih in pametnih telefonih ter tablicah, nimamo prave izbire. VRML in X3D sta preveč omejena in ne omogočata napredne interakcije, ki je potrebna že za enostavne igre. Javo je potrebno namestiti na računalnik in za vsako spletno aplikacijo dovoliti pravice za dostop do lokalnih resursov. V zadnjih letih je imela Java zelo veliko varnostnih lukenj, kar ni pripomoglo k njeni popularnosti. Flash je enostavneje namestiti, saj te brskalnik vodi skozi postopek. Toda Flash je opustil razvoj za mobilne spletne brskalnike [19].

WebGL kot del HTML5 obljublja rešitve za vse omenjene težave. JavaScriptu v vsakem brskalniku, ki ga implementira, doda zmožnost ustvarjanja interaktivne 3D grafike brez potrebe po vtičnikih.

3.5.1 JavaScript

Spletni brskalnik prikazuje strukturirano HTML besedilo, ki ga lahko oblikujemo s stili CSS. Vsak sodobni spletni brskalnik ima tudi podporo za JavaScript, ki omogoča programiranje spletnih vsebin in doda novo dimenzijo interaktivnosti spletnim vsebinam. JavaScript je programski jezik z objektno orientiranimi zmogljivostmi in prototipnim načinom dedovanja. Je interpreterski jezik, ki je sintaktično podoben C-ju in si kar nekaj funkcionalnosti, kot so regularni izrazi in delo s polji, izposodi od Perla. Kljub temu, da ime zavaja, da je JavaScript skriptna verzija Jave, temu ni tako. JavaScript razen nekaj površinskih sintaktičnih podobnosti ni v sorodu z Javo, temveč je samostojen ter zelo zmogljiv programski jezik. [20]

JavaScript je bil razvit za uporabo v spletnih brskalnikih, ker so pri Netscapu ugotovili, da potrebujejo večjo interaktivnost spletnih vsebin z uporabnikom. Programski jezik se je prvič pojavil leta 1996 v brskalniku Netscape Navigator 2.0 in kmalu za tem tudi v Internet Explorerju 3.0, toda pod imenom JScript, da ne bi prišlo do kršenja avtorskih pravic. JavaScript se lahko enostavno doda v obstoječe spletne strani in se izvaja na strani uporabnika. Sprva je bila uporaba JavaScripta zahtevna, saj ga je vsak brskalnik imple-

mentiral malo drugače, zato je JavaScript deloval le v brskalniku, za katerega je bil napisan. Za ostale brskalnike je bilo potrebno napisati nekoliko drugačno JavaScript kodo. Ker je jezik kmalu postal de facto standard za izdelavo interaktivnih spletnih vsebin, je bila nekompatibilnost med implementacijami v brskalnikih velika ovira, zato so se pri Netscapu odločili, da nadaljnji razvoj jezika prepustijo mednarodni organizaciji ECMA International. Ko je jezik postal industrijski standard, so ga preimenovali v ECMAScript. Danes vsi sodobni brskalniki implementirajo ECMAScript skoraj identično in nekompatibilnosti med implementacijami v brskalnikih skoraj več ni. [20]

3.5.2 Kaj je WebGL?

WebGL je JavaScript programski vmesnik, ki uporabniku omogoča dostop do in uporabo grafične strojne opreme v brskalniku. Omogoča izdelavo interaktivne 3D grafike, ki se upodablja v realnem času. Ker je spletni standard, v brskalniku ni potrebno namestiti nobenega vtičnika. [21]

WebGL upravlja organizacija Khronos, ki upravlja tudi mnoge druge odprte standarde, med njimi knjižnico OpenGL in malo manj znano verzijo OpenGL za vgrajene sisteme, OpenGL ES.

WebGL je narejen na osnovi OpenGL ES 2.0 in omogoča podobne funkcije upodabljanja kot omenjena knjižnica. Zasnovan je kot kontekst upodabljanja za Canvas element v HTLM-ju. [21]

3.5.3 OpenGL ES

OpenGL ES je večplatformni brezplačni programski vmesnik, ki omogoča 2D in 3D grafiko v vgrajenih sistemih, med drugim tudi v igralnih konzolah, telefonih in vozilih. Opredeljen je kot podmnožica zmogljivosti naprednejše knjižnice OpenGL. OpenGL ES je prilagodljiv in zmogljiv vmesnik med programsko opremo in strojno pospešeno grafiko. Verzija 1.0 je namenjena enostavnejšim napravam, medtem ko verzija 2.0 omogoča veliko zmogljivejšo 3D grafiko. [22]

WebGL je zasnovan na OpenGL ES in ne na knjižnici OpenGL, da deluje na kar največjem številu grafičnih kartic in drugih naprav. Spletni brskalniki so danes zelo pomemben del pametnih telefonov, tabličnih računalnikov, televizij in igralnih konzol. Te naprave so manj zmogljive kot namizni računalniki, zato so razvijalci WebGL-ja za osnovo izbrali OpenGL ES, ker so le tako lahko omogočili, da bo WebGL dobro deloval tako na namiznih računalnikih kot na manj zmogljivih napravah. [22]

Specifikacija za WebGL 1.0 je bila dokončana 10. februarja 2011 in javno objavljena 3. marca 2011 na Game Developers Conference v San Franciscu. Razvijalci spletnih brskalnikov so z WebGL-om eksperimentirali že veliko prej, saj so bili prvi brskalniki s podporo za WebGL dostopni že pred objavo končne specifikacije:

- Google Chrome 9 - februar 2011
- Firefox 4.0 - marec 2011
- Safari 5.1 - julij 2011
- Opera 12 alpha - oktober 2011

[23]

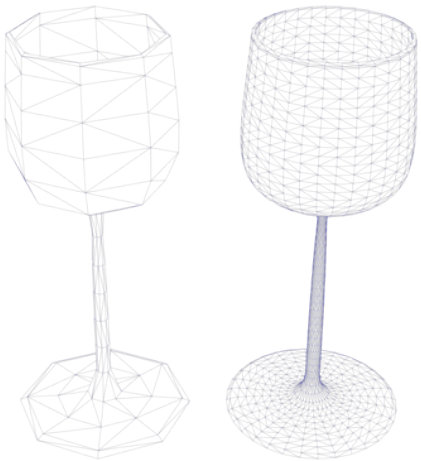
Microsoft Internet Explorer še ne podpira WebGL. Microsoft trenutno še ni objavil namere za podporo v prihodnjih različicah brskalnika Internet Explorer 10, ker meni, da tehnologija še ni dovolj zrela za širšo uporabo. [24]

Poglavje 4

WebGL

4.1 WebGL cevovod

Od prvih dni realnočasnega 3D-ja je trikotnik osnovni element, s katerim se izrisujejo 3D scene. Tri oglišča trikotnika v prostoru predstavljajo ravnino. Trikotnik je najenostavnejši tridimenzionalni objekt, zato so kompleksnejši 3D objekti predstavljeni z množico trikotnikov. Iz več trikotnikov, kot je sestavljen objekt, lepše in bolj naravno bo videti.



Slika 4.1: Mrežni model kozarca

Slika 4.2: Senčen model kozarca

Sliki 4.1 in 4.2 prikazujeta primerjavo objekta, ki je sestavljen iz manj trikotnikov, in objekta, sestavljenega iz veliko trikotnikov.

Kljub temu da so današnje grafične kartice zelo zmogljive in premorejo ogromno funkcij, je trikotnik še vedno osnovni element za izrisovanje 3D scen, kar se odraža tudi v knjižnici OpenGL.

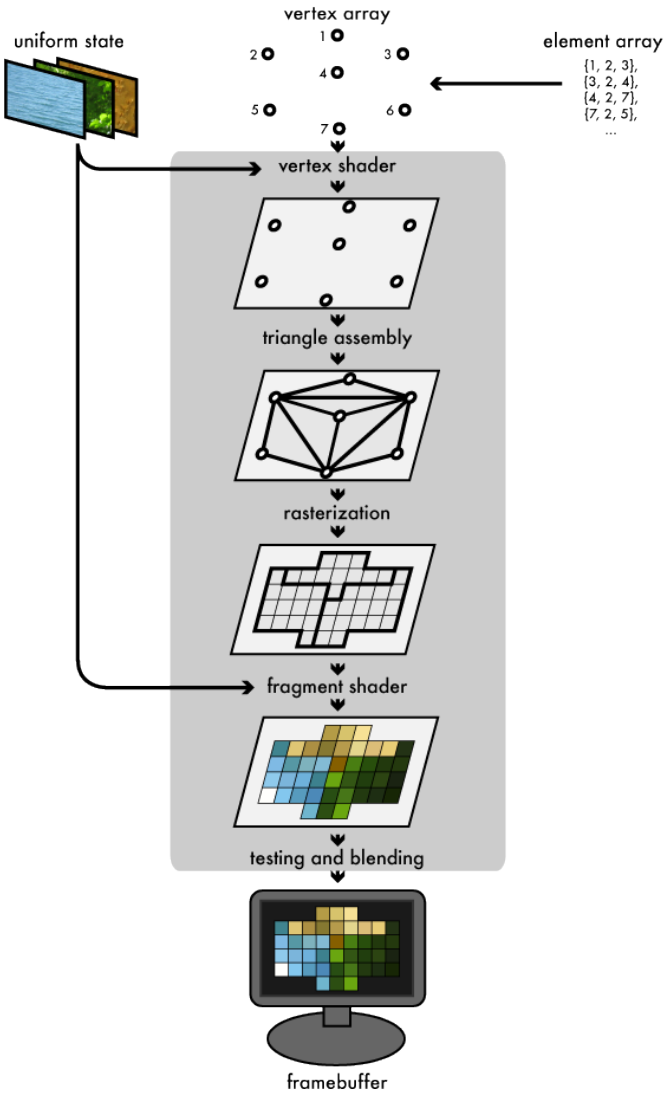
Program za izrisovanje 3D objekta poteka tako: program v pomnilnik OpenGL naloži seznam oglišč. Oglišča se projecirajo v 2D prostor (prostor zaslona) in sestavijo v trikotnike, ki se rasterizirajo v pike na zaslonu. Pikam se določijo barvne vrednosti in se izrišejo v slikovni predpomnilnik. Vsebina slikovnega predpomnilnika se izriše na zaslonu. [25]

Proces upodabljanja se začne s seznamom oglišč in seznamom atributov za vsako oglišče. Seznam vsebuje atribute oglišč, kot so položaj v 3D prostoru, barva, tekstura, normala in podobno. Za WebGL mora biti to polje oglišč zapisano v JavaScriptu.

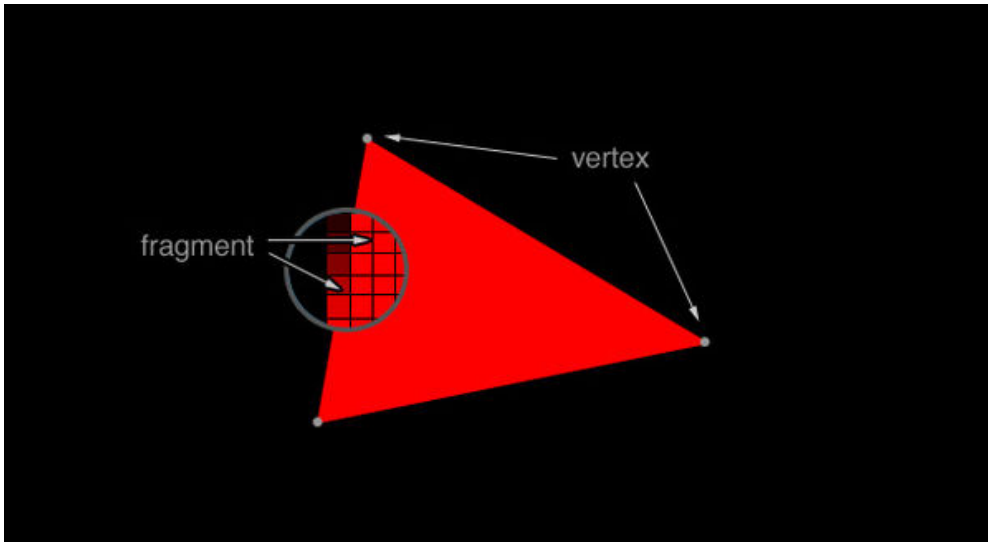
To lahko dosežemo na več načinov:

- s pretvorbo datotek, ki opisujejo 3D objekt (na primer .obj), v JavaScript. V tem primeru lahko uporabimo objekte, ki smo jih naredili v programu za 3D modeliranje.
- z uporabo JavaScript knjižnic, ki omogočajo izdelavo kompleksnih objektov iz geometrijskih primitivov in nam za naše objekte izdelajo polje oglišč.

Seznam atributov oglišč in seznam oglišč se pošlje grafičnemu procesorju. Procesor obdela vsako oglišče z ogliščnim modelom senčenja. Ogliščni model senčenja preračuna podane atribute v nove, kot so položaj oglišča na zaslonu, barva, koordinate teksture. Sledi sestavljanje oglišč v trikotnike, s pomočjo seznama oglišč, ki pove, katera oglišča skupaj tvorijo trikotnik. Nato se izvede rasterizacija, ki zavrže trikotnike, ki niso vidni na zaslonu, kot na primer tisti, ki so skriti za objektom pred njim ali so izven vidnega polja zaslona, in vse ostale dele razbije na delčke velikosti pike na zaslonu. Ogliščni model senčenja preračuna tudi barve za celoten trikotnik. Barvni atribut je določen samo za oglišča trikotnika, zato mora ogliščni model senčenja iz teh podatkov preračunati barvni preliv za vso površino in določiti barvo vsaki piki, ki sestavlja trikotnik. Generirane delčke velikosti pike obdela sedaj še fragmentni model senčenja. Ta program izračuna preslikavo teksture, osvetlitev in globino vsakega delčka in omogoča zelo zahtevne efekte. To je tudi najbolj časovno občutljiv del grafičnega cevovoda. Ker WebGL uporablja programirljiv grafični cevovod, je mogoče sprogramirati lasten ogliščni in fragmentni model senčenja.



Slika 4.3: Cevovod WebGL [26]



Slika 4.4: Modeli senčenja v WebGL-ju [28]

4.2 Uporaba WebGL

Ker je WebGL programski vmesnik za JavaScript, začnemo z osnovno HTML5 datoteko, ki vsebuje element <canvas> in <script>. V element canvas dodamo id atribut z vrednostjo canvasEl, da lahko enostavno dostopamo do elementa z JavaScriptom ter določimo širino in višino. V element script bomo dodali kodo JavaScript, ki bo z uporabo funkcij WebGL kreirala vsebino.

```
<!DOCTYPE HTML>
<HTML>
  <head>
  </head>
  <body>
    <canvas id="canvasEl" width="800" height="500"></canvas>
    <script>
    </script>
  </body>
</HTML>
```

V element script dodamo spremenljivko canvasRef, v katero shranimo referenco do elementa canvas v HTML-ju.

```
var canvasRef = document.getElementById('canvasEl');
```

V novo spremenljivko WebGL shranimo kontekts experimental-webgl. Kontekst WebGL je trenutno še v eksperimentalni fazi, zato je tudi tako poimenovan. Ko se bodo razvijalci brskalnikov odločili, da je WebGL dovolj zrel za vsakdanjo uporabo, se bo ime konteksta preimenovalo v WebGL in se bo beseda experimental izpustila.

```
var WebGL = canvasRef.getContext('experimental-webgl');
```

S klicem funkcije clearColor določimo barvo ozadja. Barvo določimo s štirimi števili med 0 in 1. Prva tri števila so vrednosti za RGB barve: rdeča, zelena in modra, četrta je vrednost prosojnosti. S klicem druge funkcije clear izbrišemo vse elemente, da se vidi ozadje. Funkciji clear posredujemo COLOR_BUFFER_BIT, ki hrani barve vseh izrisanih točk na zaslonu.

```
WebGL.clearColor(0,0,0,1);
WebGL.clear(WebGL.COLOR_BUFFER_BIT);
```

Končna koda je torej:

```
<!DOCTYPE HTML>
<HTML>
  <head>
  </head>
  <body>
    <canvas id="canvasEl" width="800" height="500"></canvas>
    <script>
     var canvasRef = document.getElementById("canvasEl");
     var WebGL = canvasRef.getContext("experimental-webgl");
     WebGL.clearColor(0,0,0,1);
     WebGL.clear(WebGL.COLOR_BUFFER_BIT);
    </script>
  </body>
</HTML>
```

To je najenostavnejši primer uporabe WebGL-ja, ki prikaže le modro ozadje. [27]

Ker je WebGL nizkonivojski programski vmesnik, nam omogoča tudi izdelavo lastnih modelov senčenja. Le te lahko programiramo z GLSL, ki je programski jezik za programiranje modelov senčenja v knjižnici OpenGL. Model senčenja je program, s katerim programiramo programirljiv cevovod upodabljanja grafične kartice ter s tem določamo, kako so elementi prikazani na zaslonu, kar nam omogoča izdelavo zelo kompleksnih grafičnih učinkov.

Če želimo prikazati uporaben 3D prikaz, moramo v WebGL-ju definirati modele senčenja. V HTML dodamo dva elementa script tipa x-shader/x-vertex in x-shader/x-fragment, ki bosta vsebovala kodo za modela senčenja.

```
<!DOCTYPE HTML>
<HTML>
  <head>
     <script id="vertex" type="x-shader/x-vertex"></script>
     <script id="fragment" type="x-shader/x-fragment"></script>
  </head>
  <body>
    <canvas id="canvasEl" width="800" height="500"></canvas>
    <script>
     var canvasRef = document.getElementById("canvasEl");
     var WebGL = canvasRef.getContext("experimental-webgl");
     WebGL.viewport(0, 0, canvasRef.width, canvasRef.height);
     WebGL.clearColor(0,0,0,1);
     WebGL.clear(WebGL.COLOR_BUFFER_BIT);
    </script>
  </body>
</HTML>
```

Zelo enostaven model senčenja oglišč:

```
attribute vec2 aVertexPosition;
void main() {
        gl_Position = vec4(aVertexPosition, 0.0, 1.0);
}
```

Vsak model senčenja oglišč (vertex shader) ima funkcijo main, ki se izvede ob upodabljanju. Spremenljivke, določene v glavi, so parametri funkciji main. aVertexPosition je parameter tipa atribut. Atribut je tabela podatkov. Funkcija main modela senčenja bo poklicana za vsak podatek iz te tabele. Podatek atributa je ponavadi položaj oglišča, barva oglišča in podobno, toda lahko je tudi kaj povsem drugega. Ker je tabela podatkov samo tabela številk, ta števila lahko interpretiramo tudi drugače. V našem modelu senčenja oglišč je vse, kar naredimo, to, da vrednost atributa zapišemo v spremenljivko gl_Position. To je edini način, da model senčenja vrne vrednost, ker zanj ne obstaja ukaz "return". Gl_Position zahteva štirikomponentni vektor, zato moramo poskrbeti, da vanj zapišemo podatek v pravi obliki. Pomeni števil štirikomponentnega

vektorja so: koordinata x, koordinata y, globina in homogena koordinata, ki se uporablja v perspektivni projekciji.

Enostaven model senčenja fragmentov:

```
#ifdef GL_ES
precision highp float;
#endif

uniform vec4 uColor;

void main() {
        gl_FragColor = uColor;
}
```

Prva vrstica preveri, ali brskalnik, v katerem se izvaja koda, podpira novejšo verzijo standarda, ki zahteva določitev natančnosti števil s plavajočo vejico. Druga vrstica nastavi visoko natačnost števil s plavajočo vejico. Podobno kot za model senčenja oglišč ima fragment shader funkcijo main, ki kot parametre dobi spremenljivke, definirane v glavi. uColor je definiran kot uniformna spremenljivka, kar pomeni, da bo vrednost v uColor enaka za vsako točko. uColor je štirikomponentni vektor z vrednostmi za rdečo, zeleno in modro barvo ter prosojnost. Vrednost za spremenljivki bomo določili z JavaScriptom. Fragmentni model senčenja je zelo preprost, saj le zapiše vrednost barve v spremenljivko gl_FragColor.

Modela senčenja smo programirali s programskim jezikom GLSL. Delo nadaljujemo z JavaScript-om. Ker se modela senčenja ne izvajata v brskalniku kot JavaScript, ju moramo najprej pripraviti za izvajanje. To naredimo s prevajanjem in povezovanjem.

```
var v = document.getElementById("vertex").firstChild.nodeValue;
var f = document.getElementById("fragment").firstChild.nodeValue;

var vs = WebGL.createShader(gl.VERTEX_SHADER);
WebGL.shaderSource(vs, v);
WebGL.compileShader(vs);

var fs = WebGL.createShader(WebGL.FRAGMENT_SHADER);
WebGL.shaderSource(fs, f);
WebGL.compileShader(fs);
```

V spremenljivki v in f s pomočjo DOM-a skopiramo kodo naših dveh modelov senčenja. Naredimo objekt ogliščnega modela senčenja, vanj naložimo kodo za naš model senčenja ter jo prevedemo.

Podobno naredimo še za fragmentni model senčenja.

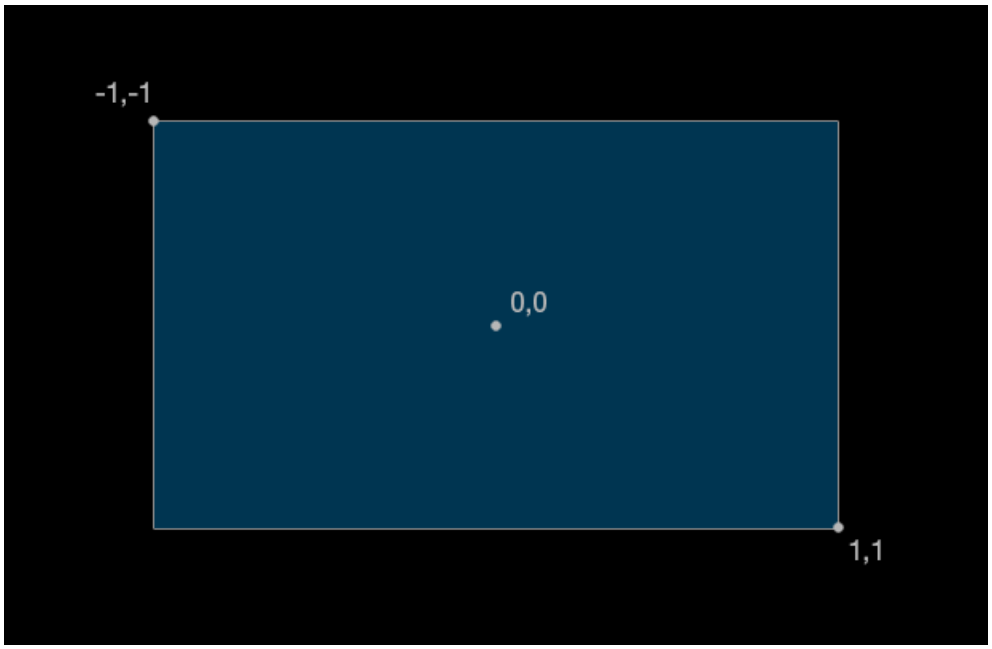
Oba modela senčenja povežemo v objekt, ki ga v WebGL-ju imenujemo program. To storimo s sledečo kodo:

```
program = WebGL.createProgram();
WebGL.attachShader(program, vs);
WebGL.attachShader(program, fs);
WebGL.linkProgram(program);
```

Modeli senčenja so pripravljeni. Narediti moramo še objekt, ki ga želimo izrisati.

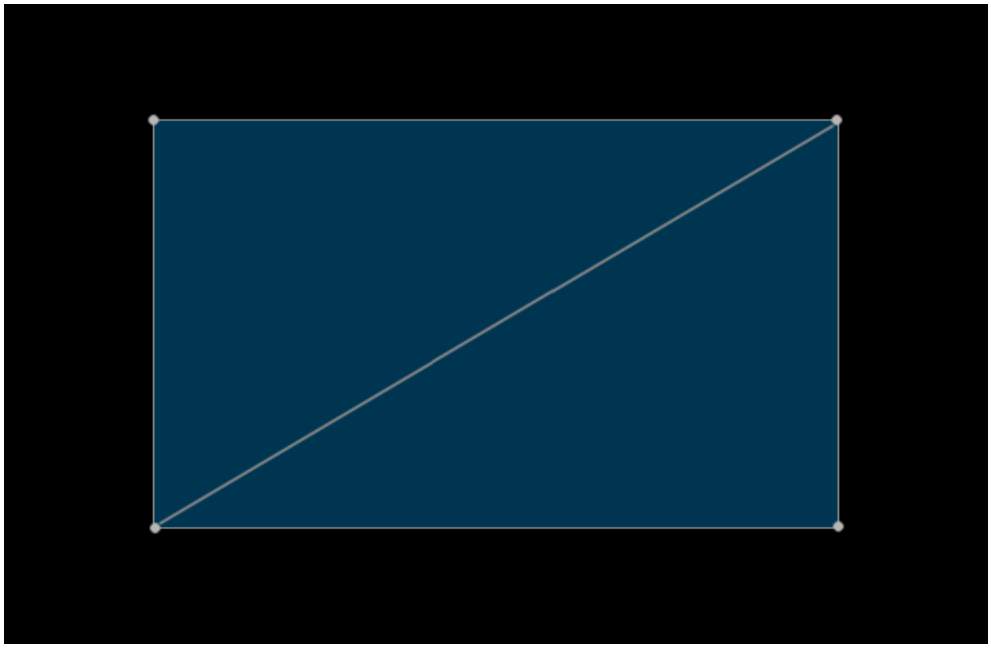
WebGL koordinatni sistem:

Risalna površina v WebGL-ju ima v zgornjem levem kotu koordinate -1,-1 v spodnjem desnem kotu 1,1 in v središču 0,0. To velja za vsako risalno površino v WebGL-ju, ne glede na razmerje stranic. To pomeni, da moramo biti pozorni na razmerje višine in širine naše risalne površine, da se bodo objekti pravilno prikazovali.



Slika 4.5: Koordinatni sistem v WebGL-ju

V WebGL-ju imamo tri vrste primitivov za risanje: točka, premica in trikotnik. Kot vemo, je trikotnik najpopularnejši primitiv za risanje v 3D-ju, zato ga bomo tudi tukaj uporabili za izris kvadrata. Za izris kvadrata potrebujemo dva trikotnika, kar lahko vidimo v sliki 4.6.



Slika 4.6: Kvadrat, sestavljen iz dveh trikotnikov

Za izris moramo narediti seznam točk za oba trikotnika. Seznam točk se v WebGL-ju imenuje buffer.

```
var razmerje = canvas.width / canvas.height;

var oglisca= new Float32Array([
        -0.5, 0.5*razmerje, 0.5, 0.5*razmerje,  0.5,-0.5*razmerje,
        -0.5, 0.5*razmerje, 0.5,-0.5*razmerje, -0.5,-0.5*razmerje
        ]);

vbuffer = WebGL.createBuffer();
WebGL.bindBuffer(WebGL.ARRAY_BUFFER, vbuffer);
WebGL.bufferData(WebGL.ARRAY_BUFFER, oglisca, gl.STATIC_DRAW);

stKoordinat = 2;
```

```
stOglisc =oglisca.length / stKoordinat;
```

V kodi smo določili spremenljivko razmerje, v katero zapišemo razmerje med stranicama, da lahko pravilno izrišemo štirikotnik, ne glede na velikost in razmerje stranic elementa canvas. V spremenljivko "oglisca"vpišemo seznam koordinat točk za oba trikotnika v 2D prostoru. Za vsak trikotnik potrebujemo x in y koordinato vsake od treh točk. Y koordinato pomnožimo z razmerjem širine in višine, da dobimo isto dolžino in širino kvadrata.

Ko imamo koordinate trikotnikov pripravljene v spremenljivki, naredimo WebGL buffer in povežemo seznam koordinat z bufferjem. To naredimo s klicem bindBuffer, ki označi ravnokar definirani buffer kot aktivnega. Vse operacije, ki jih delamo z WebGL-om, se izvajajo na aktivnem bufferju. Na koncu določimo še število koordinat za točke trikotnikov, ki smo jih zapisali v spremenljivko "oglisca"in izračunamo število točk.

Predno jih lahko narišemo, moramo podatke poslati modelu senčenja. Ogliščni model senčenja ima atribut tipa vec2 z imenom "aVertexPosition", model senčenja fragmentov ima uniform spremenljivko tipa vec4 z imenom "uColor". Ker sta oba modela senčenja prevedena v program, uporabimo referenco tega programa za pošiljanje podatkov modeloma.

```
WebGL.useProgram(program);
program.uColor = WebGL.getUniformLocation(program, "uColor");
WebGL.uniform4fv(program.uColor, [0.0, 0.5, 0.8, 1.0]);

program.aVertexPosition = WebGL.getAttribLocation(program,
                                          "aVertexPosition");
WebGL.enableVertexAttribArray(program.aVertexPosition);
WebGL.vertexAttribPointer(program.aVertexPosition, stKoordinat,
                                          WebGL.FLOAT, false, 0, 0);
```

Prvi ukaz določi, da se bo program uporabil za vse sledeče ukaze. S klicem funkcije getUniformLocation dobimo referenco do uniformne spremenljivke "uColor", ki se nahaja v fragmentnem modelu senčenja in jo shranimo v dinamično spremenljivko našega programa. Z naslednjim ukazom spremenljivki "uColor"določimo RGBa vrednost. RGBa vrednost je polje štirih števil med 0 in 1, ki določajo intenzivnost rdeče, zelene in modre barve ter prosojnost. Podobno kot smo naredili za uniformno spremenljivko sedaj s funkcijo getAttribLocation shranimo referenco do atributa "aVertexPosition"v dinamično spremenljivko programa. V naslednjem koraku eksplicitno omogočimo atribut in mu določimo kazalec. WebGL ve, da mora uporabiti podatke v spremenljivki

"oglisca", ker je to trenutno aktiven buffer, ki smo ga nastavili kot aktivnega z ukazom bindBuffer nekaj vrstic kode nazaj. V realnem programu koda ni tako enostavna. Pri določanju aktivnega bufferja je potrebna previdnost.

S klicem funkcije vertexAttribPointer povemo programu, da je atribut sestavljen iz dveh zaporednih števil. WebGL ju bo avtomatično prebral in zapakiral v spremenljivko tipa vec2, ki jo zahteva ogliščni model senčenja.

Sledi zadnja stopnja programa, izrisovanje:

```
WebGL.drawArrays(WebGL.TRIANGLES, 0, stOglisc);
```

Prvi parameter funkciji za izrisovanje določi način izrisovanja. Ker želimo izrisati površine, podamo parameter TRIANGLES. Na voljo imamo še dva načina izrisovanja: izrisovanje črt - LINES in izrisovanje točk - POINTS. Funkcija za izrisovanje bo uporabila trenutno aktiven buffer in poklicala program z modeli senčenja za vsak artibut tolikokrat, kot določimo z zadnjo spremenljivko stOglisc, kar je v našem primeru štirikrat. [28]

```
<!DOCTYPE HTML><HTML>
   <head>
     <script id="vertex" type="x-shader/x-vertex">
       attribute vec2 aVertexPosition;

       void main() {
         gl_Position = vec4(aVertexPosition, 0.0, 1.0);
       }
     </script>

     <script id="fragment" type="x-shader/x-fragment">
       #ifdef GL_ES
         precision highp float;
       #endif

       uniform vec4 uColor;

       void main() {
         gl_FragColor = uColor;
       }
     </script>
  </head>
  <body>
```

```
<canvas id="canvasEl" width="800" height="500"></canvas>
<script>
  var canvasRef = document.getElementById("canvasEl");
  var WebGL = canvasRef.getContext("experimental-webgl");
  WebGL.viewport(0, 0, canvasRef.width, canvasRef.height);
  WebGL.clearColor(0,0,0,1);
  WebGL.clear(WebGL.COLOR_BUFFER_BIT);

 var v = document.getElementById("vertex").firstChild.nodeValue;
 var f = document.getElementById("fragment").firstChild.nodeValue;

 var vs = WebGL.createShader(WebGL.VERTEX_SHADER);
 WebGL.shaderSource(vs, v);
 WebGL.compileShader(vs);

 var fs = WebGL.createShader(WebGL.FRAGMENT_SHADER);
 WebGL.shaderSource(fs, f);
 WebGL.compileShader(fs);

 program = WebGL.createProgram();
 WebGL.attachShader(program, vs);
 WebGL.attachShader(program, fs);
 WebGL.linkProgram(program);

 if (!WebGL.getShaderParameter(vs, WebGL.COMPILE_STATUS))
         console.log(WebGL.getShaderInfoLog(vs));

 if (!WebGL.getShaderParameter(fs, WebGL.COMPILE_STATUS))
         console.log(WebGL.getShaderInfoLog(fs));

 if (!WebGL.getProgramParameter(program, WebGL.LINK_STATUS))
         console.log(WebGL.getProgramInfoLog(program));

 var razmerje = canvasRef.width / canvasRef.height;

 var oglisca= new Float32Array([
         -0.5, 0.5*razmerje, 0.5, 0.5*razmerje,  0.5,-0.5*razmerje,
         -0.5, 0.5*razmerje, 0.5,-0.5*razmerje, -0.5,-0.5*razmerje
         ]);
```

```
    vbuffer = WebGL.createBuffer();
    WebGL.bindBuffer(WebGL.ARRAY_BUFFER, vbuffer);
    WebGL.bufferData(WebGL.ARRAY_BUFFER, oglisca, WebGL.STATIC_DRAW);

    stKoordinat = 2;
    stOglisc =oglisca.length / stKoordinat;


   WebGL.useProgram(program);

   program.uColor = WebGL.getUniformLocation(program, "uColor");
   WebGL.uniform4fv(program.uColor, [0.0, 0.5, 0.8, 1.0]);

   program.aVertexPosition = WebGL.getAttribLocation(program,
                                        "aVertexPosition");
   WebGL.enableVertexAttribArray(program.aVertexPosition);
   WebGL.vertexAttribPointer(program.aVertexPosition, stKoordinat,
                                        WebGL.FLOAT, false, 0, 0);

   WebGL.drawArrays(WebGL.TRIANGLES, 0, stOglisc);
   </script>
  </body>
</HTML>
```

Ker je WebGL nizkonivojski programski vmesnik, je zelo prilagodljiv, ampak tudi zahteven za uporabo. Še preden je bil standard WebGL uradno potrjen, so se pojavile knjižnice JavaScript, ki so bistveno olajšale delo razvijalcem.

Popularne knjižnice WebGL so: Three.js, PhiloGL, GLGE, J3D, CopperLicht.

4.3 WebGL težave

Kot vsaka nova tehnologija ima tudi WebGL nekaj začetnih težav. WebGL preko JavaScripta omogoča nizkonivojski dostop do grafičnega procesorja, kar obide vse varnostne mehanizme sodobnih spletnih brskalnikov. To povzroči kopico novih varnostnih težav, saj so do sedaj vse aplikacije, ki so uporabljale

nizkonivojski dostop do GPU-ja, bile programi, ki jim je uporabnik zaupal in jih je sam namestil na računalnik.

Zato proizvajalci grafičnih kartic pri izdelavi gonilnikov niso bili pozorni na varnost, saj niso pričakovali, da bodo lahko zlonamerneži pisali kodo, ki se bo izvajala na GPU-ju.

Z uporabo WebGL-ja na spletnih straneh ima sedaj vsak avtor spletne strani možnost dostopa do grafičnega procesorja, zato je zanesljivo in varno delovanje gonilnikov grafičnih kartic bistvenega pomena za varnost računalniškega sistema.

Kmalu po vključitvi WebGL-ja v sodobne spletne brskalnike so se pojavile težave z zanesljivostjo gonilnikov grafičnih kartic. Ker proizvajalci spletnih brskalnikov ne morejo vplivati na kvaliteto gonilnikov, so vpeljali črni seznam grafičnih kartic za operacijske sisteme Windows, Linux in Mac OS X. Na črni seznam so vpisane vse grafične kartice, katerih gonilniki imajo znane varnostne pomankljivosti, in so zato nevarne za uporabo v WebGL-ju, saj lahko zlonamerna koda povzroči sesutje gonilnika grafične kartice ali celo sesutje celega računalniškega sistema. [29]

Ker si pri izračunavanju kompleksne 3D grafike želimo predvsem hitrost in ker so bile do sedaj vse aplikacije, ki so imele dostop do GPU-ja, zaupne aplikacije, 3D grafične knjižnice kot so OpenGL, OpenGL ES, Direct3D ne delajo standardnih preverjanj pri dostopu do pomnilnika. Na primer: vse omenjene knjižnice omogočajo izrisovanje indeksiranih poligonalnih objektov in do pred kratkim nobena od omenjenih ni preverjala, ali vsak indeks kaže na veljavno točko. [29]

Pri WebGL-ju kličejo funkcije 3D grafične knjižnice programi, ki jim ne zaupamo. Zato je preverjanje veljavnosti dostopov do spomina zelo pomembno, da program ne spreminja predela spomina, ki je dodeljen drugim programom. Prekoračitev medpomnilnika so pogoste varnostne ranljivosti na CPU-ju.

Tega se zaveda tudi skupina Khronos, ki je v specifikacijo OpenGL že dodala varnostno razširitev GL_ARB_robustness, ki je specifično izdelana za preprečevanje napadov ohromitve storitve in preprečevanje napada z dostopom do spomina, ki je izven dostopa programa. [30, 31]

Primer ene od varnostnih lukenj kraja grafičnega spomina:

1) spletni strežnik uporabniku pošlje spletno stran, ki uporablja JavaScript kodo z WebGL zlonamerno kodo;

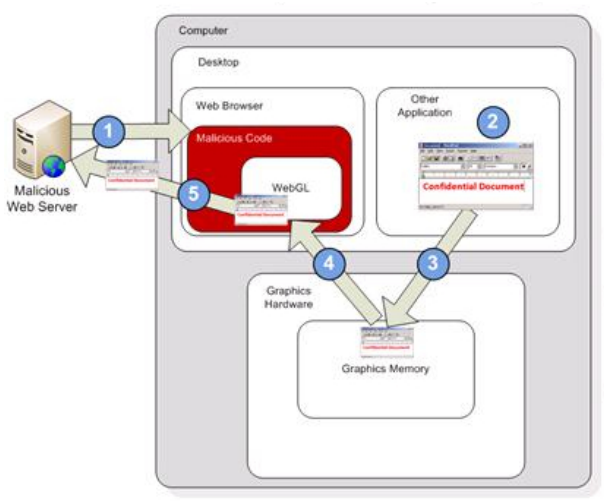
2) na računalniku uporabnika drugi program uporabi grafično kartico za prikaz vsebine zaupnega dokumenta na zaslonu;

3) izrisano okno programa z zaupnim dokumentom se zapiše v deljeni grafični spomin;

4) zaradi varnostne luknje v implementaciji WebGL-ja je okno programa, ki je zapisano v grafičnem spominu, dostopno zlonamerni kodi;

5) zlonamerna koda prebere okno druge aplikacije iz grafičnega spomina in jo kot sliko pošlje spletnemu stežniku.

[29]



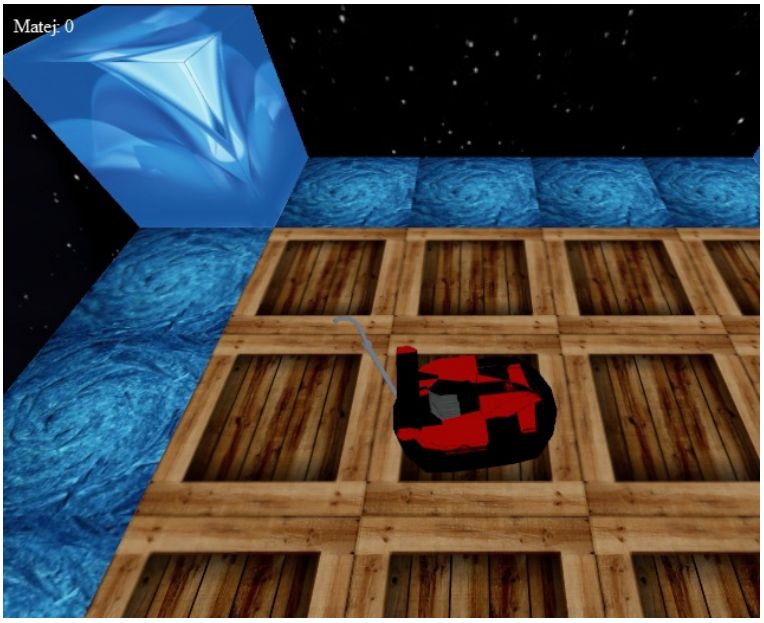
Slika 4.7: Kraja grafičnega pomnilnika [29]

Poglavje 5

3D WebGL igra

5.1 Igra

Kot primer uporabe WebGL-ja sem izdelal večigralsko spletno 3D igro, kjer vsak igralec vozi svoj avtomobilček. Igra se odvija na platformi, ki lebdi v zraku, in naloga igralca je v danem času z zaletavanjem v avtomobilčke drugih poriniti čim več igralcev čez rob platforme.



Slika 5.1: 3D igra, narejena v WebGL-ju

Igra je razdeljena na 3 komponente:

- Strežniški del: osrednji del igre je strežniški del, ki je napisan v pythonu in izračunava vse premike in trke v igri. Da se igralci s svojimi brskalniki lahko povezujejo na strežnik igre, mora biti ta dostopen na računalniškem omrežju ali internetu. Za HTTP strežnik s podporo za WebSocket povezave sem uporabil ogrodje Tornado , ki je prav tako napisano v pythonu.
- Hitra dvosmerna komunikacija med strežnikom in brskalnikom: hitra komunikacija med igralci in strežnikom je narejena s pomočjo HTML5 WebSocket povezav, ki omogočajo hitro dvosmerno komunikacijo med strežnikom in brskalnikom.
- Upodabljanje 3D objektov v brskalniku: na strani igralca je vse sprogramirano v JavaScriptu. Za upodabljanje 3D vsebin sem uporabil Copperlight JavaScript knjižnico, ki omogoča enostavno in hitro uporabljanje z WebGL-om.

5.2 Strežniški del

Strežniški del je izdelan v ogrodju Tornado. Tornado je ogrodje za izdelavo spletnih strežnikov. Ker je Tornado neblokirajoč in hiter, lahko sočasno vzdržuje več tisoč povezav, kar pomeni, da je idealen za spletne storitve, ki zahtevajo komunikacijo v realnem času. [32]

Za Tornado sem se odločil, ker je napisan v pythonu in podpira WebSocket protokol, kar mi je olajšalo izdelavo strežnika za igro in omogočilo enostavno uporabo hitre dvosmerne komunikacije z novodobnimi spletnimi brskalniki.

Igralec začne igro tako, da odpre spletni naslov, na katerem teče strežnik igre v spletnem brskalniku. Po vpisu svojega imena mu strežnik igre pošlje podatke za izris vseh elementov v igri:

- platforma in teksture platforme,
- avtomobilčki in njihove teksture,
- teksture ozadja.

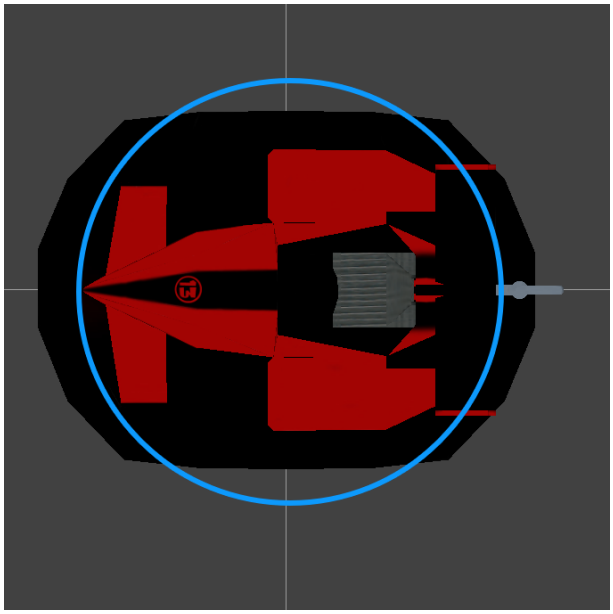
Vsi podatki o okolju igre in slike, ki se uporabijo kot teksture, se prenesejo le ob prijavi in ne ob vsaki komunikaciji s strežnikom. Po sprejemu vseh podatkov se igralcu izriše okolje igre in avtomobilčki vseh igralcev.

Strežnik igre vsebuje vso logiko in pravila igre. Izračunava in hrani vse podatke o igralcih, kot so:

- položaj avtomobilčka,
- hitrost,
- smer gibanja,
- trki ob ovire,
- trki ob druge igralce,
- točkovanje,
- povezave WebSocket za komunikacijo,

Za premikanje v igri se uporabljajo smerne tipke. Ob pritisku smernih tipk se kombinacija pritisnjenih tipk pošlje preko protokola WebSocket strežniku igre. Le ta preračuna nov položaj avtomobilčkov vseh igralcev in nove koordinate pošlje vsem igralcem. Podatke o novih položajih igralcev uporabi JavaScript v brskalniku igralca za izris avtomobilčkov na novih položajih.

Eden najzahtevnejših delov igre je zaznavanje trkov med objekti v 3D prostoru. Enostaven način iskanja trkov je primerjanje vsakega z vsakim in preverjanje, ali se objekta prekrivata. To je zelo počasen in procesorsko zahteven način. Da sem pospešil zaznavanje trkov, sem naredil dve poenostavitvi. Za zaznavanje trkov se išče presek samo med objekti, ki so v neposredni bližini avtomobilčka. Vse objekte v prostoru za zaznavanje presekov poenostavim v krog in iščem preseke samo med krogi, ki predstavljajo objekte. Primer kroga, ki predstavlja poenostavljen objekt, vidite na sliki 5.2.



Slika 5.2: Krog za poenostavljen način zaznavanja trkov

Slika 5.3: Napaka pri zaznavanju trkov

Ker objekti niso okrogli, pride do nepravilnosti pri zaznavanju trkov. To

je neizbežna posledica poenostavitve pri zaznavanju trkov, toda v primeru te igre ni preveč moteča.

5.3 Hitra dvosmerna komunikacija med strežnikom in brskalnikom

Protokol HTTP, ki se uporablja za komuniciranje med brskalnikom in strežnikom, ni namenjen dvosmernemu komuniciranju v realnem času, zato sem za komunikacijo med strežnikom igre in brskalnikom igralca uporabil protokol WebSocket. [34]

Specifikacija WebSocket se razvija kot del pobude HTML5 in v brskalniku ponuja JavaScript vmesnik, ki definira hkratno komunikacijo preko ene vtičnice, preko katere se lahko pošiljajo sporočila med odjemalcem in strežnikom. [35]

Uporaba protokola WebSocket je enostavna. Ustvarimo novo instanco WebSocket in ji posredujemo url naslov strežnika. Predpona ws:// v URL-ju sporoči, da bo to povezava WebSocket.

```
ws = new WebSocket('ws://10.0.0.2:8888/werecars-ws');

ws.onopen = function () {
    myname = prompt('Vpišite vaše ime:', 'Matej');
};
ws.onmessage = function (e) {
// Obdelava prejetih podatkov
}
ws.onclose = function () {
    log("WebSocket zaprt!");
};
ws.onerror = function () {
    log("WebSocket napaka!");
};
```

Povezava WebSocket se vzpostavi z nadgradnjo protokola HTTP v protokol WebSocket med uvodnim rokovanjem. Povezavo uporabljamo s funkcijami, ki jih priključimo dogodkom onopen, onmessage, onclose in onerror. Za pošiljanje podatkov uporabimo funkcijo send. Povezavo zapremo s funkcijo close.

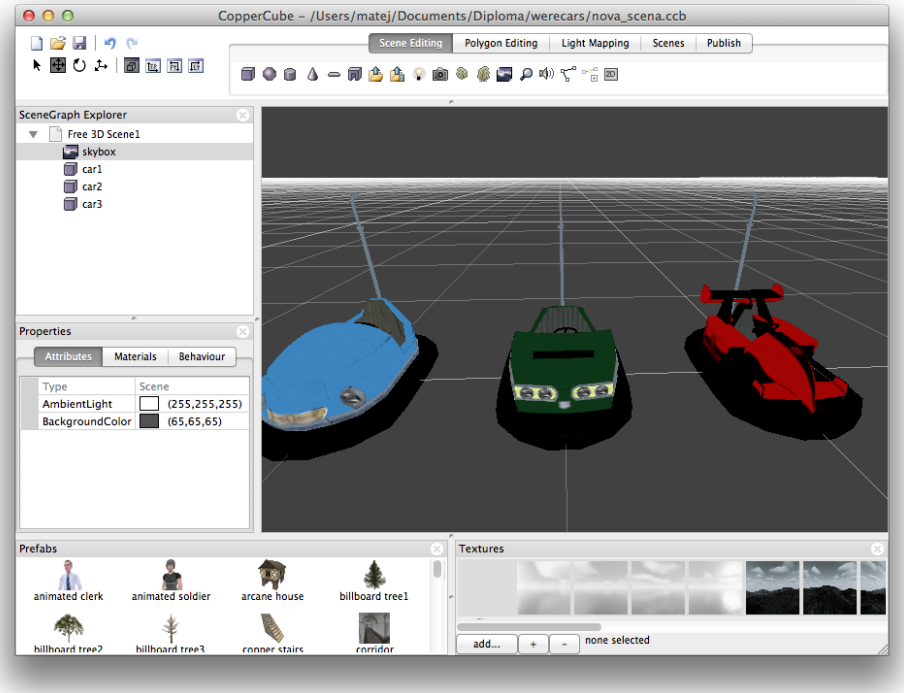
```
ws.send("Sporočilo strežniku!");
ws.close();
```

Predno se podatki pošljejo strežniku igre, se shranijo v JavaScript objekt, ki se pretvori v format JSON. Prav tako so podatki, ki jih strežnik vrne, v JSON formatu. JSON omogoča strukturiranje podatkov v elemente in podelemente z zelo malo dodatnih podatkov, kar je pomembno za hitro komunikacijo. [36]

5.4 Upodabljanje 3D objektov v brskalniku

Upodabljanje 3D objektov v brskalniku sem naredil s knjižnico Copperlight. [33] Za to knjižnico sem se odločil, ker je bila ena prvih knjižnic, ki je podpirala uvažanje 3D objektov, narejenih v programih za 3D modeliranje, kot so 3D Studio Max, Maya, Blender. Dodatna prednost je, da so avtorji knjižnice izdelali tudi urejevalnik 3D scen Coppercube, ki omogoča izvažanje scen v WebGL, Flash, program za Windows in program za Mac.

Z urejevalnikom Coppercube sem naredil sceno s tremi modeli avtomobilčkov in jo izvozil v obliko, ki jo lahko uporabim v JavaScriptu.



Slika 5.4: Urejevalnik Copercube

Scena z avtomobilčki se v brskalnik naloži ob inicializaciji igre. Platforma, po kateri se vozijo avtomobilčki, se izriše dinamično. Podatke o platformi pošlje strežnik igre, ko se igralec prijavi. To omogoča, da lahko enostavno spremenim okolje, v katerem se odvija igra.

Glavni del igre, ki se izvaja v brskalniku, je v funkciji, ki se proži ob WebSocket onmessage dogodku. V tej funkciji se prestrežejo in obdelajo podatki, ki jih pošlje strežnik. Glede na tip podatkov, ki jih brskalnik sprejme, se lahko izvede izrisovanje okolja, osveži seznam igralcev, prestavijo se avtomobilčki.

Poglavje 6

Zaključek

6.1 Sklepne ugotovitve

Spletni brskalnik je v zadnjih letih napredoval v nepogrešljiv program in platformo za izdelavo kompleksnih aplikacij. Razvoj zahtevnejših spletnih aplikacij se je začel z enostavnimi e-poštnimi odjemalci, danes lahko v brskalniku že urejamo slike, zvok, video in celo programiramo s spletnim IDE.

Naslednja stopnja v razvoju brskalnika kot platforme za zmogljive programe je prikaz in manipuliranje 3D objektov brez vtičnikov. Tehnologija WebGL se hitro razvija in spreminja. To sem opazil tudi pri izdelavi igre za diplomo, ko je igra prenehala delovati skoraj ob vsaki nadgradnji brskalnika. Podobne težave sem imel tudi s povezavami WebSocket. Kljub omenjenim težavam je delovanje obeh tehnologij zanesljivo in hitro.

Slike

Literatura

[1] Gregor Klajnšek, Borut Žalik, *Standard VRML*, Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko, Slovenija, 2002, pogl. 1.

[2] (2012) A history of Windows. Dostopno na: http://windows.microsoft.com/en-US/windows/history

[3] (2011) A history of Internet Explorer. Dostopno na: http://www.microsoft.com/windows/WinHistoryIE.mspx

[4] (2012) Microsoft releases Internet Explorer 3.0 second beta. Dostopno na: http://www.zdnet.com/microsoft-releases-internet-explorer-3-0-second-beta-3002063942/

[5] Aaron E. Walsh, Mikaël Bourges-Sévenier, *Core Web3D*, Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000, pogl. 20.

[6] Don Brutzman, Leonard Daly, *X3D: Extensible 3D Graphics for Web Authors*, Amsterdam: Elsevier/Morgan Kaufmann, cop., 2007, pogl. 1.

[7] (2012) What is X3D? Dostopno na: http://www.web3d.org/about/faq/#general-1

[8] (2012) X3D and HTML5. Dostopno na: http://www.web3d.org/x3d/wiki/index.php/X3D_and_HTML5

[9] (2012) Java (programming language). Dostopno na: http://en.wikipedia.org/wiki/Java_(programming_language)

[10] (2012) Frequently Asked Questions About Java. Dostopno na: http://www.oracle.com/technetwork/Java/faq-141681.html

[11] (2012) Java 3D. Dostopno na: http://en.wikipedia.org/wiki/Java_3D

[12] (2012) Java Web Start Technology. Dostopno na: http://download.oracle.com/javase/6/docs/technotes/guides/javaws/developersguide/overview.html

[13] Rob Bateman, Richard Olsson, *The Essential Guide to 3D in Flash*, friends of ED, 1st edition 2010, pogl. 1.

[14] (2012) Benchmarking Flash Player 10. Dostopno na: http://arstechnica.com/software/news/2008/10/benchmarking-Flash-player-10.ars

[15] (2012) Flash Performance Benchmarks. Dostopno na: http://www.tomshardware.com/reviews/macbook-air-chrome-16-firefox-9-benchmark,3108-8.html

[16] (2011) Audio A6. Dostopno na: http://microsites.audi.co.uk/microsites/RS6/index.html#/home/

[17] (2010) Making of Motorola RAZR 2. Dostopno na: http://direct.motorola.com/hellomoto/razr2/razr2makingof/

[18] (2012) Orange La collection spring summer 09. Dostopno na: http://www.orange.com/sirius/lacollection/printemps_ete09/index_en.html

[19] (2012) Flash to Focus on PC Browsing and Mobile Apps; Adobe to More Aggressively Contribute to HTML5. Dostopno na: http://blogs.adobe.com/conversations/2011/11/Flash-focus.html

[20] David Flanagan, *JavaScript: The Definitive Guide, Fifth Edition*, O'Reilly, 5 edition 2006, pogl. 1.

[21] (2012) WebGL Specification, Editor's Draft 09 July 2012. Dostopno na: http://www.khronos.org/registry/webgl/specs/latest/

[22] (2012) OpenGL ES - The Standard for Embedded Accelerated 3D Graphics. Dostopno na: http://www.khronos.org/opengles/

[23] (2012) Getting a WebGL Implementation. Dostopno na: http://www.khronos.org/webgl/wiki/Getting_a_WebGL_Implementation

[24] (2012) WebGL Considered Harmful. Dostopno na: http://blogs.technet.com/b/srd/archive/2011/06/16/webgl-considered-harmful.aspx

[25] (2012) An introduction to WebGL. Dostopno na:
http://dev.opera.com/articles/view/an-introduction-to-webgl/

[26] (2012) An intro to modern OpenGL. Chapter 1: The Graphics Pipeline. Dostopno na:
http://duriansoftware.com/joe/An-intro-to-modern-OpenGL.-Chapter-1:-The-Graphics-Pipeline.html

[27] (2012) Raw WebGL 101 — Part 1: getting started. Dostopno na:
http://dev.opera.com/articles/view/raw-webgl-part1-getting-started/

[28] (2012) Get started with WebGL: draw a square. Dostopno na:
http://www.netmagazine.com/tutorials/get-started-WebGL-draw-square

[29] (2011) WebGL - A New Dimension for Browser Exploitation. Dostopno na:
http://www.contextis.co.uk/resources/blog/webgl/

[30] (2012) Main Page/cms/security - WebGL Public Wiki. Dostopno na:
https://www.khronos.org/webgl/wiki/Main_Page/cms/security

[31] (2012) WebGL Security - khronos.org news. Dostopno na:
http://www.khronos.org/news/permalink/webgl-security

[32] (2012) Tornado. Dostopno na:
http://www.tornadoweb.org/

[33] (2012) CopperLicht™ - fast WebGL JavaScript 3D Engine. Dostopno na:
http://www.ambiera.com/copperlicht/

[34] (2012) What is WebSocket? Dostopno na:
http://websocket.org

[35] (2012) About HTML5 WebSockets. Dostopno na:
http://www.websocket.org/aboutwebsocket.html

[36] (2012) JSON. Dostopno na:
http://www.json.org/