

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Jure Kolenko

**Orodje za ročno poravnavo 2D in 3D
medicinskih posnetkov**

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Matija Marolt

Ljubljana 2015

Fakulteta za računalništvo in informatiko podpira javno dostopnost znanstvenih, strokovnih in razvojnih rezultatov. Zato priporoča objavo dela pod katero od licenc, ki omogočajo prosto razširjanje diplomskega dela in/ali možnost nadaljne proste uporabe dela. Ena izmed možnosti je izdaja diplomskega dela pod katero od Creative Commons licenc <http://creativecommons.si>

Morebitno pripadajočo programsko kodo praviloma objavite pod, denimo, licenco *GNU General Public License*, različica 3. Podrobnosti licence so dostopne na spletni strani <http://www.gnu.org/licenses/>.

Besedilo je oblikovano z urejevalnikom besedil L^AT_EX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Fiddy diddling.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Jure Kolenko sem avtor diplomskega dela z naslovom:

Vizualizacija medicinskih podatkov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matije Marolta,
- 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 na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 24. avgusta 2015

Podpis avtorja:

Thanks Obama.

Posvetilo.

Kazalo

Povzetek

Abstract

1	Uvod	1
2	Metode in orodja	3
2.1	Java in Eclipse	3
2.2	Hranjenje mrežnega modela	4
2.3	Aplikacija za vizualizacijo	4
2.4	Strojno pospešen izris 3D grafike	4
2.5	Projekcija texture na 3D model	5
3	Implementacija	9
3.1	Programska ovojnica za LWJGL	9
3.2	Samostojna aplikacija	11
3.3	Integracija v aplikacijo za vizualizacijo	12
4	Zaključek	13

Seznam uporabljenih kratic

Povzetek

V vzorcu je predstavljen postopek priprave diplomskega dela z uporabo okolja L^AT_EX. Vaš povzetek mora sicer vsebovati približno 100 besed, ta tukaj je odločno prekratek.

Ključne besede: Muh keyword1, hue.

Abstract

This sample document presents an approach to typesetting your BSc thesis using L^AT_EX. A proper abstract should contain around 100 words which makes this one way too short.

Keywords: Muh keyword1, hue.

Poglavje 1

Uvod

Poglavje 2

Metode in orodja

2.1 Java in Eclipse

Za razvoj programskega dela diplomske naloge sem uporabil programski jezik Java in razvojno okolje Eclipse, saj sem z obema že seznanjen.

Java

Java je objektno orientiran programski jezik, ki ga je razvilo podjetje Sun Microsystems leta 1995. Osnovan je na jezikoma C in C++, vendar pa je višjenivojski, saj pred uporabnikom skrije različne konstrukte, kot so naprimer kazalci in upravljanje s pomnilnikom. Prevede se v nižjenivojski jezik, ki ga nato tolmači javanski navidezni stroj (angl. Java virtual machine - JVM). To omogoča princip 'napiši enkrat, izvajaj kjerkoli', kar je bil eden izmed ciljev razvoja tega programskega jezika.

Eclipse

Eclipse je odprtokodno integrirano razvojno okolje. V osnovi je namenjeno programiranju Java aplikacij, vendar obstajajo vtičniki tudi za mnoge druge jezike. Prednost uporabe Eclipsea je urejevalnik, ki omogoča samodejno dopolnjevanje izrazov in opozarja na napake, ter razhroščevalnik in prikaz do-

kumentacije.

2.2 Hranjenje mrežnega modela

Za hranjenje mrežnega modela je uporabljena spremenjena različica standarda Obj datotek, ki ga je razvilo podjetje Wavefront technologies. Standard je odprt in ima široko podporo med orodji za vizualizacijo 3-dimenzionalnih (3D) mrežnih modelov. Različica podpira samo del standarda, in sicer zapis za pozicije vozlišč ter definicijo ploskev, ne podpira pa naprimer teksturnih koordinat in normal, saj se v programu ne uporabljajo oziroma se izračunajo.

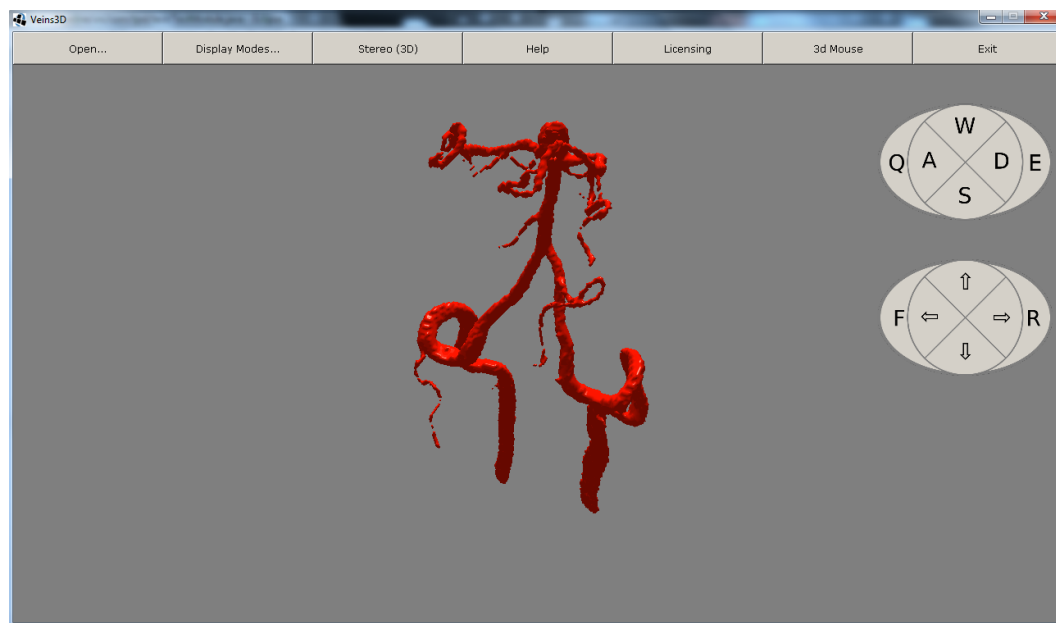
2.3 Aplikacija za vizualizacijo

Aplikacija je namenjena prikazovanju ožilja pacienta. Model se da poljubno obračati in premikati, lahko pa se ga tudi ogleduje v stereo prikazu. Modeli žil so prebrani iz datotek tipa Obj.

2.4 Strojno pospešen izris 3D grafike

Za strojno pospešen izris 3D grafike obstaja več aplikacijsko programerskih vmesnikov, najbolj znana in razširjena pa sta DirectX in OpenGL. DirectX je razvil Microsoft leta 1995 in je namenjen za aplikacije za operacijski sistem Windows, OpenGL pa je leta 1992 razvil Silicon Graphics Inc., sedaj pa ga nadzira neprofitna skupnost Khronos Group, in je namenjen strojno pospešenemu izrisu na večih platformah in v večih jezikih.

V diplomski nalogi sem uporabil odprtokodno knjižnico LWJGL (lightweight java game library - lahka javanska knjižnica za igre), ki omogoča uporabo vmesnika OpenGL.



Slika 2.1: Glavni prikaz aplikacije za vizualizacijo.

2.5 Projekcija texture na 3D model

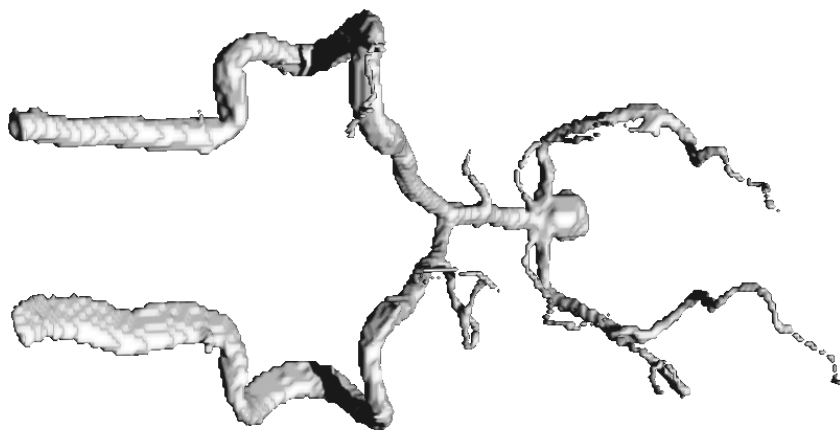
Za projeciranje texture je najprej potrebno vedeti kako se izrisujejo sence, prehod na projekcijo pa je nato precej lahek.

Za izris senc je potrebno prizorišče najprej izrisati iz pogleda luči. Luč, ki meče sence, se v računalniški grafiki obnaša enako kot kamera. Ob izrisu ne uporabljamo barv, temveč zapišemo samo globino objekta, oziroma razdaljo do kamere. Pri izrisu prizorišča je nato treba za vsak objekt projecirati tako v pogled kamere kot v pogled luči, nato pa ob senčenju objekta v globinski teksturi preveriti, če je v pogledu luči, in če je ali je najbližji objekt in je osvetljen ali ni najbližji in je zatemnjen.

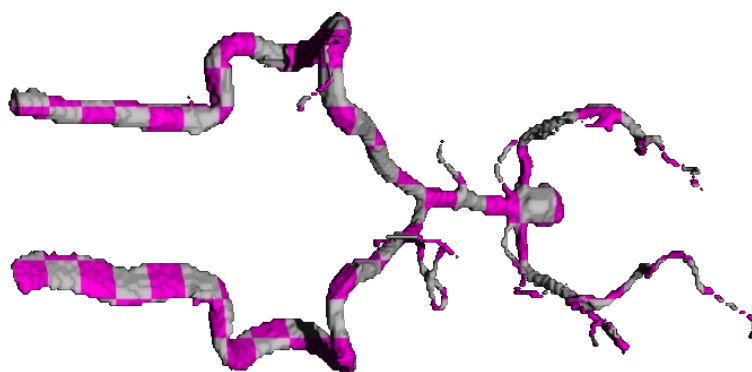
Pri izrisu projekcije je potrebno poleg globinske texture dodati še teksturo s sliko projekcije, nato pa, če je objekt osvetljen namesto uniformne barve svetlobe uporabiti barvo s texture.



Slika 2.2: Primer globinske slike. Svetlejši kot je piksel, dlje od kamere je objekt.



Slika 2.3: Primer osenčenega modela. Luč je na desni strani pred modelom.



Slika 2.4: Primer projekcije teksture na model.

Poglavje 3

Implementacija

3.1 Programska ovojnica za LWJGL

Za lažje delo z OpenGL sem napisal programsko ovojnico, ki temelji na knjižnici LWJGL. OpenGL zahteva veliko knjigovodstva, saj je treba vsa sredstva, kot so medpomnilniki, senčilniki, texture in več, ročno uničiti, da se pomnilnik na grafični enoti ne zasiči. Ovojnica to poenostavi tako, da OpenGL sredstva zavije v smiselno poimenovane razrede. Ovojnica nudi tudi transformacije med kvaternioni in evklidskimi koti in razrede za transformacije in kamere.

Medpomnilniki

Prenašanje podatkov na grafično enoto in z nje v OpenGL poteka prek medpomnilnikov. Ovojnica podpira medpomnilnike za sezname, ki se uporabljajo za prenašanje podatkov kot so pozicije oglišč, normale v ogliščih in teksturne koordinate, medpomnilnike za elemente, ki grafični enoti povedo iz katerih oglišč so sestavljene ploskve, ter medpomnilnike za okvirje (angl. frame), ki hranijo slike za izris.

Senčilniki

Senčilniki so podprogrami za izris, ki jih mora napisati uporabnik. V starejših verzijah se je uporabljal fiksni cevovod, ki pa je okornejši in manj zmogljiv od uporabniških programov. Ovojnica podpira pet vrst senčilnikov, in sicer senčilnike za oglišča (angl. vertex shader), za geometrijo (geometry), za drobce (fragment) in za deljenje (tessellation), obstaja pa še senčilnik za izračunavanje (compute shader), ki pa ni podprt. Senčilniki se združujejo v programe, ki tvorijo fleksibilen cevovod za izris.

Teksture

Teksture v OpenGL so objekti, ki vsebujejo eno ali več slik, ter so uporabni kot vir slikovnih podatkov v senčilnikih ali kot tarča za izris v medpomnilnikih za okvirje. Ker je inicializacija tekstur v OpenGL precej kompleksna, ovojnica poskrbi za inicializacijo osnovnih parametrov.

Točkovni nizi

Točkovni nizi (angl. vertex array object) so OpenGL objekti, ki hranijo stanje potrebno za izris objektov. Hranijo le stanje, ne pa tudi samih podatkov, to je reference na medpomnilnike in obliko podatkov v teh medpomnilnikih, ne pa tudi samih podatkov iz medpomnilnikov. Prednost se kaže v hitrosti in velikosti kode, saj je potrebno veliko manj sprememb stanja.

Transformacija

Transformacija (angl. transform) je razred, ki je namenjen hranjenju transformacije objekta, to je rotacija, položaj in velikost. Položaj in velikost sta predstavljena kot vektorja dolžine 3, rotacija pa kot kvaternion. Podprte so rotacije in premiki v lokalnem ter globalnem koordinatnem sistemu, ter rotacije v poljubnem sistemu. Razred vsebuje tudi funkcije za pretvorbo iz kvaternionov v eulerjeve kote, za izračun transformacijske matrike iz stanja

transformacije, ter za izračun kvaterniona iz rotacijske matrike.

(Formule za pretvorbe?)

Kot posebna transformacija se obravnava tudi kamera, le da so pri kameri premiki ravno obrnjeni, ima pa še dodatne parametre za pogled kamere. Podprti sta dve vrsti kamere: perspektivna in vzporedna.

Ostalo

Poleg že naštetega ovojnika vsebuje še funkcijo za razhroščevanje, ki napake, ki jih vrača OpenGL, pretvori v berljive nize ter izpiše vrstico v kateri je bila funkcija klicana, kar močno olajša iskanje napak.

3.2 Samostojna aplikacija

Funkcionalnost orodja sem najprej implementiral kot samostojno aplikacijo, saj je tako lažje preveriti pravilnost delovanja. Aplikacija nima uporabniškega vmesnika, implementirane pa so le osnovne kontrole(?), saj služi le dokazovanju delovanja, ne pa tudi dejanski uporabi. Sestavljena je iz testnega razreda, razreda ki implementira dejansko funkcionalnost in razredov za branje Obj datotek, ki pa niso uporabljeni v končni aplikaciji, saj ima ta svoj bralnik. Posnetki se v samostojni aplikaciji naložijo avtomatsko iz testnih datotek.

(SLIKA)

Premikanje pogleda

Aplikacija podpira dva pogleda, pogled iz kamere ter pogled iz projektorja. Pogled iz projektorja je namenjen poravnavi 2D in 3D posnetkov, saj je v tem pogledu 2D posnetek pri miru, premika pa se le 3D posnetek. Nepremičnost posnetka v tem pogledu je doseženo s tem, da se pogled in pravokotnik, na katerem je posnetek, obračata in premikata za iste vrednosti.

2D posnetek

Dvo dimenzionalni posnetek je rentgenska slika ožilja. Slika se projecira na pravokotnik, ki je prilagojen velikosti slike. Slika se da v prostoru rotirati neodvisno od modela, kar služi poravnavi, ko pa sta model in slika poravnana, pa se lahko rotira skupaj z modelom.

3D posnetek

Tri dimenzionalni posnetek je mrežni model ožilja, naložena iz Obj datoteke. 3D posnetek sam po sebi se ne premika, premika se le kamera, kar pa da uporabniku podoben vtis.

3.3 Integracija v aplikacijo za vizualizacijo

Za konec sem funkcionalnost integriral še v aplikacijo za vizualizacijo. Poleg funkcionalnosti samostojne aplikacije je bilo potrebno zagotoviti, da se modul obnaša enako kot že obstoječa aplikacija, kar pa ni bilo čisto mogoče, saj modul uporablja drugačno vrsto kamere kot preostala aplikacija.

Premikanje pogleda

Model žil se premika enako kot v že obstoječi aplikaciji, s smernimi tipkami za premik levo, desno, naprej in nazaj ter s tipkama 'r' in 'f' za gor in dol. Obrača se s tipkami 'w', 'a', 's', 'd', 'q' in 'e' ter z miško, tako da se lev miškin gumb drži nad objektom in miško premika. Premikanje z miško je, podobno kot v že obstoječi aplikaciji, narejeno s projekcijo kazalca na navidezno kroglo okoli modela, vendar pa je zaradi drugačne vrste kamere izračun preseka drugačen. (*matematika za izračunom*)

Opis problemov z integracijo v Neck veins.

Poglavje 4

Zaključek

Ful sm zadovoljen, huehuehue.