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

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

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

Seznam uporabljenih kratic

Povzetek

V vzorcu je predstavljen postopek priprave diplomskega dela z uporabo okolja LaTeX. 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 LaTeX. 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 Eclipsa je urejevalnik, ki omogoča samodejno dopolnjevanje izrazov in opozarja na napake, ter razhroščevalnik in prikaz do-

kumentacije.

2.2 Hranjenje mrežnega modela

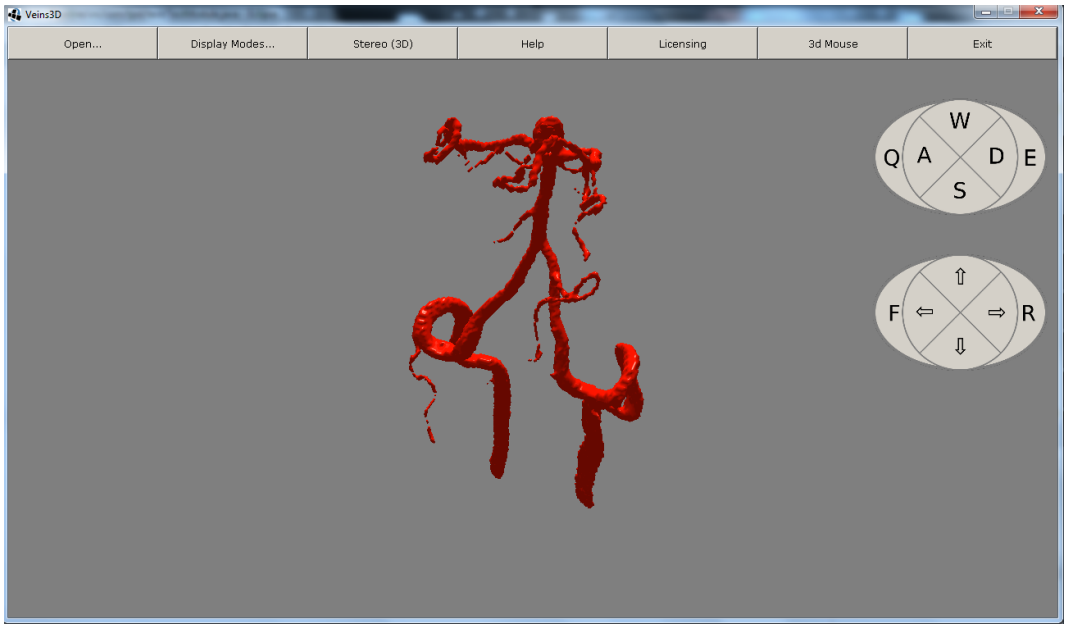
Za hranjenje mrežnega modela je uporabljena spremenjena različica standarda Obj datotek [citat], ki ga je razvilo podjetje Wavefront technologies. Standard je odprt in ima široko podporo med orodji za vizualizacijo 3-dimenzionalnih (3D) mrežnih modelov. Standard je tekstoven, tako da so datoteke zlahka berljive, slabost tega pa je večja velikost datotek.

```
v -1 -1 0
v 1 -1 0
v -1 1 0
v 1 1 0
vt 0 0 0
vt 1 0 0
vt 0 1 0
vt 1 1 0
vn 0 0 1
vn 0 0 1
vn 0 0 1
vn 0 0 1
f 1/1/1 2/2/2 4/4/4
f 1/1/1 4/4/4 3/3/3
```

Datoteka 2.1: Primer obj datoteke, ki predstavlja kvadrat.

Različica podpira samo del standarda, in sicer zapis za položaje vozlišč ter definicijo ploskev, ne podpira pa naprimer teksturnih koordinat in normal, saj se v programu ne uporabljajo oziroma se izračunajo.

Lahko podaš tudi primer zapisa trikotnika ali kakšne druge geometrije



Slika 2.1: Glavni prikaz aplikacije za vizualizacijo.

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 aplikacijskih programerskih vmesnikov, najbolj znana in razširjena pa sta DirectX [referenca] in OpenGL [referenca]. DirectX je razvil Microsoft leta 1995 in je namenjen za razvoj aplikacij na operacijskem sistemu 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.

Tukaj bi bilo smiselno mogoče nekaj več napisati tudi o samem OpenGL-

u. Zakaj ga sploh imamo, kaj nam omogoča, ipd.

V diplomski nalogi sem uporabil odprtokodno knjižnico LWJGL (angl. lightweight java game library), ki omogoča uporabo vmesnika OpenGL.

Opis OpenGLja.

2.5 Projekcija tekstura na 3D model

Tole potrebuje precej "rewrita". To podpoglavje precej predelaj. Po korakih razloži kako se izvede posamezni korak in pomožnosti dodaj kakšno skico, ki ponazarja kaj se v posameznem koraku zgodi. Prav tako navedi vire, v kolikor si kakšnega uporabljal.

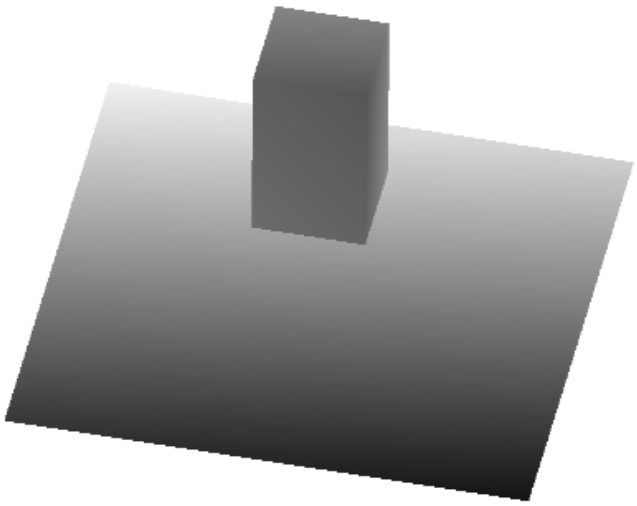
Projekcija tekstura na objekt deluje v treh korakih. Najprej je treba izrisati globinsko sliko iz pogleda projektorja, nato ugotoviti kaj na modelu dejansko zadanejo žarki projektorja, potem pa še pravilno pobarvati dele, ki jih žarki zadanejo.

Globinska slika

Luč se v računalniški grafiki obnaša enako kot kamera. Objekte se, tako kot pri kameri, projecira v pogled luči, nato pa se objekte izriše, pri tem pa se beleži le razdalja do luči oziroma globina najbližjih objektov v vsaki točki. Slika 2.2 je primer izrisa globinske slike.

Izris objekta s sencami

Za izris objekta s sencami, se objekt najprej projecira v pogled kamere in pogled luči. Za vsak delec objekta v pogledu kamere se nato v globinski sliki preveri, če je ta delec viden tudi v pogledu luči. Če je delec viden, se ga osvetli z barvo luči, drugače pa je temnejši. Slika 2.3 je primer tako osenčenega modela.



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

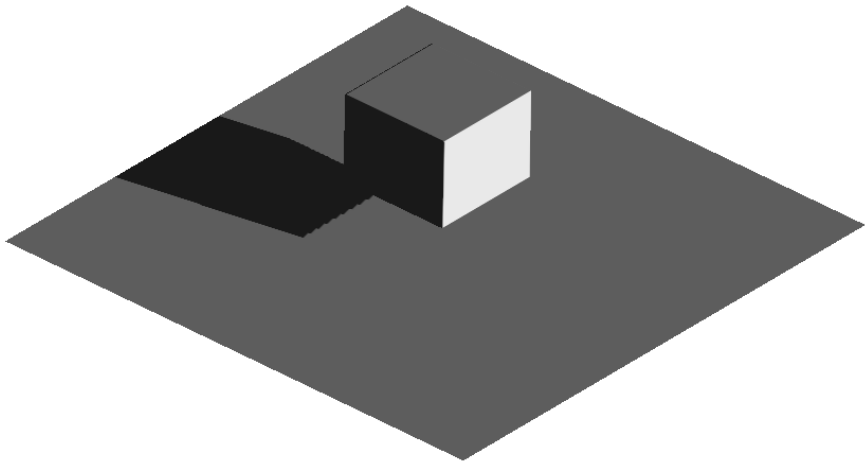
Izris objekta s projecirano sliko

Za izris s projekcijo moramo, če je delec viden, izračunati kje v sliki se ta delec nahaja, nato pa namesto bele svetlobe vzeti barvo svetlobe, ki je na tem mestu v sliki. Slika 2.4 je primer izrisa s projecirano sliko, kjer je projecirana slika vijolično-siva šahovnica.

Tukaj dodaj še opis podatkov, ki jih bomo uporabljali. Predstaviš da so podatki v OBJ pretvorjeni iz volumetričnih zajetih podatkov kot tudi kakšen je format v katerem so x-ray slike.

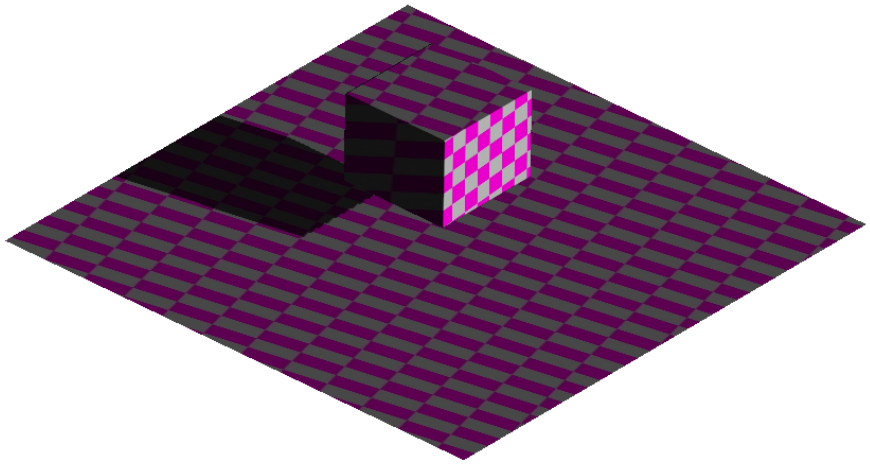
2.6 Nabor podatkov

3D posnetki so volumetrični posnetki ožilja možganov, narejeni z magnetno resonanco, ki so nato s pomočjo funkcionalnosti že obstoječe aplikacije



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

pretvorjeni v mrežni model, 2D posnetki pa so rentgenski posnetki ožilja shranjeni kot slikovna datoteka tipa jpeg [citation needed].



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 Ni ravno ustrezen izraz. To opiši., saj je treba vsa sredstva, kot so medpomnilniki, senčilniki, teksture 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 Povej kateri vsi medpomnilniki obstajajo in za kaj se uporabljajo. Ovojnica podpira medpomnilnike za sezname, ki se uporabljajo za prenašanje podatkov kot so položaji 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. To raje razdelaj po točkah, kjer za vsako točko opišeš kaj posamezna vrsta medpomnilnikov hrani.

Senčilniki

V katerem jeziku se pišejo. Dobro bi bilo omeniti, da senčilniki nadomestijo del fiksnega cevovoda in katerega.

Senčilniki so podprogrami za izris, ki jih mora napisati uporabnik. V starejših verzijah se je uporabljal fiksen 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 (tesselation), obstaja pa še senčilnik za izračunavanje (compute shader), ki pa ni podprt. Senčilniki se združujejo v programe, ki tvorijo fleksibilen cevovod za izris.

OpenGL senčilniki se pišejo v jeziku GLSL (angl. OpenGL Shading Language). GLSL je osnovan na jeziku C in omogoča prenosljivost med operacijskimi sistemi in različnimi proizvajalci strojne opreme.

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 teksture 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

Mogoče bi bilo smiselno kaj več o samih transformacijah napisati v predhodnjem poglavju.

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.

Izjeme

Ovojnica podpira več vrst izjem. OpenGL sam ne ustavi izvajanja programa ob napaki, vendar pa je izris nepravilen, zato ob nekaterih operacijah ovojnica preverja, če je na grafični enoti vse v redu. Izjeme se lahko prožijo, ko se senčilnik ne prevede pravilno, ko se program ne poveže pravilno ali ko se medpomnilnik za okvirje ne inicializira pravilno.

Ostalo

Poleg že naštetega ovojnica 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.

Dobro bi bilo dodati neko shemo tvoje ovojnice, kjer se vidi iz katerih komponent je sestavljena.

3.2 Samostojna aplikacija

Funkcionalnost orodja sem najprej implementiral kot samostojno aplikacijo, saj je tako lažje preveriti pravilnost delovanja. Aplikacija nima uporabniškega vmesnika, implementiran pa je le osnovni uporabniški nadzor, saj služi le kot primer 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 Kaj je en, kaj je drug, kakšna je razlika, morda skica. 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. Tole je potrebno opisati na bolj razumljiv način.

2D posnetek

Dvo dimenzionalni posnetek je rentgenska slika ožilja. Slika se projecira na pravokotnik, ki je prilagojen velikosti slike. Sliko 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, naložen iz Obj datoteke. 3D posnetek se 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*)

Nalaganje datotek

Funkcionalnost za nalaganje datotek je obstajala že v osnovni aplikaciji. Uporabnik s pritiskom na gumb *'odpri'* dobi pojavno okno, v katerem izbira datoteke. Podprti formati so bili mhd za volumetrične datoteke, ter obj za mrežne modele. Moral sem dodati še možnost za nalaganje slikovnih datotek v standardnih formatih jpeg [citation needed] in png [citation needed]. Java podpira ta dva formata v svoji standardni knjižnici, tako da je bila implementacija preprosta.

Manjka ti še predstavitev osnovne aplikacije in predstavitev modulov, ki si jih adaptiral, uporabljal pri svojem delu.

Poglavje 4

Zaključek

Ful sm zadovoljen, huehuehue.