LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

“HŌSEN” – RĪKS 3D MODEĻU TEKSTŪRU KRĀSOŠANAI UN UZKLĀŠANAI

KVALIFIKĀCIJAS DARBS

Autors: **Čaks Levits**

Studenta apliecības Nr.: cl15002

Darba vadītājs: Jānis Zuters, Dr. Dat.

RĪGA 2017

ANOTĀCIJA

Hōsen ir lietojumprogrammatūra, ar kuras palīdzību var radīt dažāda veida tekstūras, kuras ir sastopamas datorgrafikā, piemēram, difūzijas tekstūras un normālkartes. Primārā funkcija ir ļaut lietotājam, zīmējot tekstūru, uzreiz redzēt kā tā izskatās uzklātā veidā uz trīsdimensionāla modeļa. Lietotājam ir iespējams importēt Wavefront .obj modeli, kā arī četras tekstūras (.bmp rastrus): difūzijas, normāļu karti, spekulāro karti un vides karti, kuras tiek izmantotas modeļa attēlošanai.

ABSTRACT

Hōsen is an application which enables its user to paint and create some of the most common types of textures in computer graphics such as diffuse textures and normal maps. The primary function is to paint textures and see them applied on a 3D mesh in real time. The user may import Wavefront .obj files – the geometry as well as four textures (.bmp bitmaps) – a diffuse map, a normal map, a specular map and an environment map, which are then used to render the mesh.

SATURS

DEFINĪCIJAS

Ēnotājs (Shader) – programma, kuru izpilda grafiskais procesors (GPU – graphics processing unit) jeb grafikas paātrinātajs. Programmas raksta ēnošanas valodās (shading language), kuras tiek kompilētas par GPU instrukcijām.

Difūzijas karte (Diffuse map) – tekstūra, kas satur objekta krāsas informāciju.

Atspulgu/vides kartēšana (Reflection/environment mapping) – apgaismojuma tehnika, kurā atstarojoša priekšmeta virsmas izskats tiek aproksimēts ar iepriekš sagatavotu attēlu.

Spoguļatstarošanās karte (Specular map) – tekstūra, kas apraksta virsmas spoguļatstarošanās/difūzās atstarošanās intensitāti. Tumšāki pikseļi ir vairāk matēti, gaišāki – vairāk spīdīgi.

Izciļņu karte (Bump map) – attēls – tekstūra, ar kuras palīdzību 3D modeļos rada nelīdzenu virsmu un sīkāku ģeometrisku detaļu ilūziju.

Normālkarte (Normalmap/Dot3 bump map) – izciļņu kartes paveids.

Reljefa karte (Heightmap) – matrica (attēls), kuras elementi (pikseļi) ir gaišuma (augstuma) vērtības no melna līdz baltam.

OpenGL (Open Graphics Library) – daudzplatformu lietojumprogrammas saskarne (API) priekš 2D un 3D grafikas.

GLFW (OpenGL framework) – OpenGL atbalsta bibliotēka lietotņu izstrādei.

GLEW (OpenGL Extension Wrangler Library) – daudzplatformu C/C++ bibliotēka darbam ar OpenGL paplašinājumiem.

GLM (OpenGL Mathematics) – C++ OpenGL matemātikas bibliotēka.

GLSL (OpenGL Shading Language) – OpenGL ēnošanas valoda.

FLTK (Fast, Light Toolkit) – Bibliotēka lietotāja saskarnes radīšanai.

BMP (Bitmap picture) – vienkāršs rastra attēlu formāts.

Wavefront .obj – ģeometrisku figūru (poligonu tīklu) aprakstošs formāts.

Poligonu tīkls (polygon mesh) – trīsdimensionāla objekta virsotņu, šķautņu un skaldņu kopa, kas apraksta objekta ģeometrisko jēgu.

ATSLĒGVĀRDI

- OpenGL
- Renderēšana
- Datorgrafika
- Ēnošana
- Modelēšana
- 3D
- Teksturēšana

IEVADS

Kopš 90-o gadu beigām grafiskie paātrinātāji ir kļuvuši aizvien izplatītāki. Mūsdienās jaudīgu videokaršu masīvi ir superdatoru mugurkauls, pateicoties to spējai daudz ātrāk izpildīt peldošā komata operācijas un lielam atmiņas joslas platumam. Protams, mūsdienās saskarties ar datoru ar papildus komerciālo grafisko paātrinātāju nav rets notikums.

Grafisko paātrinātāju parādīšanās ir pozitīvi ietekmējusi ievērojami lielu cilvēku grupu: 3D mākslinieki, kuri darbojas datoranimācijas un spēļu industrijās. Lai atvieglotu šo cilvēku darbu, izstrādāta tiek dažāda veida augstas veiktspējas lietojumprogrammatūra, un līdz ar šo programmu veiktspēju un datortehnikas skaitļošanas jaudu aug arī to funkcionalitāte, kura iespējo mākslinieku izpausties aizvien vairāk. Lietojumprogrammatūras "*Hōsen*" galvenais mērķis ir ļaut tās lietotājam reāllaikā rediģēt 3D modeļa tekstūras:

- difūzijas tekstūru, kas satur informāciju par priekšmeta virsmas krāsu,
- izciļņu karti, ar kuras palīdzību rada nelīdzenas virsmas izskatu,
- spoguļatstarošanās un kubiskās maskas, kuras nosaka priekšmeta virsmas spoguļgludos un matētos apgabalus,
- caurspīdīguma maska, kura nosaka, kuri virsmas apgabali ir caurspīdīgi.

Programma "*Hōsen*" ir rakstīta C++ valodā, izmantojot bibliotēkas: *GLFW*, *GLEW*, *GLM* un *FLTK*. Par izstrādes vidi tika izvēlēts *Microsoft Visual Studio 2015*.

1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

1.1.Ievads

1.1.1. Nolūks

Programmatūras prasību specifikācijas (PPS) dokumenta nolūks ir iepazīstināt tā lasītāju ar lietojumprogrammas "Hōsen" funkcionālajām un nefunkcionālajām prasībām. Izstrādes laikā tika pieņemts lēmums ieviest šajā PPS būtiskas izmaiņas, lai līdzinātos spējās izstrādes procesam un apvienotu spējās izstrādes un ūdenskrituma modeļa elementus, kas liek dokumentam atšķirties no standarta. Būtiskākās prasības ir zināmas jau pirms izstrādes, savukārt sīkākas prasības ar zemāku prioritāti rodas izstrādes laikā, ko var izskaidrot ar pasūtītāja vēlmju dinamiskumu, nepastāvību.

1.1.2. Darbības sfēra

Kvalifikācijas darba ietvaros ir izstrādāta programma kā arī tās dokumentācija, kurā esošie lietotāju stāsti apraksta lietojumprogrammatūras prasības un veicamās funkcijas. Programmatūras mērķis ir nodrošināt lietotājus, piemēram, 3D modelētājus un māksliniekus ar rīku, kas ļautu krāsot tekstūras, uzreiz redzot, kā tās vizuāli ietekmē objekta – modeļa izskatu.

1.1.3. Saistība ar citiem dokumentiem

Par pamatu autora kvalifikācijas darba dokumentācijas daļai ir ņemts LVS 68:1996 "Programmatūŗas prasību specifikācijas ceļvedis".

1.2. Vispārējs apraksts

1.2.1. Produkta perspektīva

Rīks "Hōsen" ir paredzēts māksliniekiem – 3D modelētājiem, kā profesionāļiem, tā arī tiem, kuriem 3D modelēšana ir hobijs. Rīks tiek radīts ar nodomu vēlāk paplašināt tā funkcionalitāti. Eksistē daudz un dažādu tekstūru veidu, kuru efektus implementēt būtu iespējams vēlāk, piem., *displacement mapping*, *parallax mapping* – pilnveidoti izciļņu/normāļu kartēšanas varianti, *lightwarp* (lokāla krāsas korekcija) un daudzas citas tehnikas. Lai iespējotu lietotāju manipulēt ar ēnotājiem, lietderīgs būtu mezglu redaktors.

1.2.2. Produkta funkcijas

Šajā nodaļā lietotāju stāstu veidā tiek aprakstītas produkta gala prasības. Izstrādājamās lietojumprogrammas galvenais uzdevums ir ļaut lietotājam radīt dažāda veida tekstūras un aplūkot tās uz modeļa. Lai gan ir zināmas pamatprasības, ir jāņem vērā tas, ka pasūtītāja vēlmes ir dinamiskas.

Lietotāja stāsts 1:

“Kā lietotājs vēlos iespēju importēt un 3D skata logā apskatīt modeli.”

Lietotāja stāsts 2:

“Kā lietotājs es vēlos iespēju 3D skata logā kustināt kameru ap apskatāmo modeli.”

Lietotāja stāsts 3:

“Kā lietotājs es vēlos iespēju pārslēgt 3D skata projekcijas režīmu no perspektīvās uz taisnleņķa.”

Lietotāja stāsts 4:

“Kā lietotājs es vēlos iespēju importēt tekstūru un redzēt to uzklātu uz modeļa.”

Lietotāja stāsts 5:

“Kā lietotājs es vēlos iespēju radīt jaunu tekstūru un redzēt to uzklātu uz modeļa.”

Lietotāja stāsts 6:

“Kā lietotājs es vēlos, lai ir iespēja krāsot vairāku tipu tekstūras: difūzijas, izciļņu, spoguļatstarošanās, caurspīdīguma un vides/kubiskos attēlus.”

Lietotāja stāsts 7:

“Kā lietotājs es vēlos, lai ir iespēja uzlikt filtrus: melnbalts, ģenerēt izciļņu tekstūru, izpludināt, asināt.”

Lietotāja stāsts 8:

“Kā lietotājs es vēlos, lai ir iespēja apstrādāt attēlu ar konvolūcijas matricu.”

Lietotāja stāsts 9:

“Kā lietotājs es vēlos iespēju mainīt otas izmēru, krāsu.”

Lietotāja stāsts 10:

“Kā lietotājs es vēlos, lai tekstūrā ieviestās izmaiņas ir uzreiz redzamas uz modeļa .”

Lietotāja stāsts 11:

“Kā lietotājs es vēlos iespēju eksportēt un saglabāt radītās tekstūras.”

1.2.3. Lietotāja raksturiezīmes

Tiek pieņemts, ka, lai lietderīgi izmantoto programmu, lietotājam ir pazīstami 3D modelēšanas un datorgrafikas pamati, ir pieredze ar attēlu apstrādes lietotnēm, un lietotājs ir pazīstams ar iespējoto tekstūru veidiem un pielietojumiem: difūzijas kartes, normālkartes (izciļņu kartes), reljefa kartes, spekulārās kartes.

1.2.4. Vispārējie ierobežojumi

Minimālās sistēmas prasības:
Operētājsistēma: Windows 7/8/10 32/64-bit
Procesors: 2+ GHz divkodolu procesors
Operatīvā atmiņa: 1 GB
Videokarte: Grafiskais paātrinātājs ar 384 MB videoatmiņu un OpenGL atbalstu.

1.2.5. Pieņēmumi un atkarības

Lietojumprogramma "*Hōsen*" ir izstrādāta priekš x86-64 Windows versijām ar OpenGL atbalstošu grafisko paātrinātāju. Lietotājam ir jāprot izmantot Windows operētājsistēmu. No lietotāja ir sagaidāma izpratne par 3D modelēšanu. Lietotājam ir pieejams interneta pieslēgums programmas lejupielādei.

1.3.Funkcionālās prasības

1.3.1. Lietotāja funkcijas

1.3.1.1. Modeļa importēšana:

1.1 tabula

Identifikators: loadMesh
Mērķis
Funkcijai ir jānolasa modeļa datne, lai to var izmantot renderēšanā.
Ievaddati

Ceļš
Apstrāde
Nolasa modeļa datnes ģeometrisko informāciju
Izvaddati
Nav
Kļūdu paziņojumi
“Neizdevās ielādēt modeli”

1.3.1.2. Tekstūras importēšana:

1.2 tabula

Identifikators: loadTexture
Mērķis
Funkcijai ir jānolasa tekstūras datne, lai to var rediģēt izmantot renderēšanā.
Ievaddati
Ceļš
Apstrāde
Nolasa tekstūru
Izvaddati
Nav
Kļūdu paziņojumi
“Neizdevās ielādēt tekstūru”

1.3.1.3. Tekstūras radīšana:

1.3 tabula

Identifikators: createTexture
Mērķis
Funkcijai ir jārada jauna tekstūra
Ievaddati
Tekstūras izmērs

Apstrāde
Rada jaunu tekstūru
Izvaddati
Nav
Kļūdu paziņojumi
“Neizdevās radīt tekstūru”

1.3.1.4. Aizvērt tekstūru:

1.4 tabula

Identifikators: closeTexture
Mērķis
Funkcijai ir jāaizver tekstūra un jāaizstāj tā pēc noklusējuma.
Ievaddati
Nav
Apstrāde
Atmet tekstūru
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.5. Saglabāt tekstūru:

1.5 tabula

Identifikators: saveTexture
Mērķis
Funkcijai ir jāsaglabā aktīvā tekstūra.
Ievaddati
Ceļš
Apstrāde

Pēc izsaukšanas aktīvo tekstūru ieraksta datnē.
Izvaddati
BMP attēls
Kļūdu paziņojumi
“Neizdevās rakstīt tekstūru”

1.3.1.6. Transformācijas:

1.6 tabula

Identifikators: transforms
Mērķis
Funkcijai ir jātransformē attēls: jāpagriež par 90, -90, 180 grādiem, jāapgriež otrādi pret asi
Ievaddati
Transformācijas tips
Apstrāde
Atkarībā no ievadītā tipa transformē attēlu.
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.7. Filtri - izpludināšana:

1.7 tabula

Identifikators: FiltergaussianBlur
Mērķis
Funkcijai ir jāizpludina attēls
Ievaddati
Nav
Apstrāde
Apstrādā attēlu ar izpludināšanu

Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.8. Filtri - asināšana:

1.8 tabula

Identifikators: Filtersharpen
Mērķis
Funkcijai ir jāasina attēls
Ievaddati
Nav
Apstrāde
Apstrādā attēlu ar asināšanu
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.9. Filtri - melnbalts:

1.9 tabula

Identifikators: FilterGreyscale
Mērķis
Funkcijai ir jāpārvērš attēls melnbaltā
Ievaddati
Nav
Apstrāde
Pārvērš attēlu par melnbaltu
Izvaddati

Nav
Kļūdu paziņojumi
Nav

1.3.1.10. Filtri - konvolūcija:

1.10 tabula

Identifikators: FilterConvolve
Mērķis
Funkcijai ir jāapstrādā attēls ar konvolūcijas matricu
Ievaddati
Konvolūcijas 3x3 matrica, koeficients
Apstrāde
Apstrādā attēlu ar filtru
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.11. Filtri – izciļņu karte:

1.11 tabula

Identifikators: gaussianBlur
Mērķis
Funkcijai ir jāpārvērš attēls par izciļņu karti
Ievaddati
Nav
Apstrāde
Attēlu pārvērš par izciļņu karti
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.12. Skats – rādīt 3D skatu:

1.12 tabula

Identifikators: viewportVisible
Mērķis
Funkcijai ir jāpaslēpj vai jāparāda 3D skats
Ievaddati
Nav
Apstrāde
Paslēpj skatu, ja tas ir redzams, un parāda, ja slēpts.
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.13. Skats – rādīt otas rīku kopu:

1.13 tabula

Identifikators: toolboxVisible
Mērķis
Funkcijai ir jāpaslēpj vai jāparāda otas rīki.
Ievaddati
Nav
Apstrāde
Paslēpj rīkus, ja tie ir redzami, un parāda, ja slēpti.
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.14. 3D skats – rādīt projām vērstas virsmas:

1.14 tabula

Identifikators: backcullEnable
Mērķis
Funkcijai ir jāpaslēpj vai jāparāda prom vērstas virsmas.
Ievaddati
Nav
Apstrāde
Paslēpj virsmas, ja tās ir redzamas, un parāda, ja slēptas.
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.15. 3D skats – projekcija:

1.15 tabula

Identifikators: setProjection
Mērķis
Funkcijai ir jāmaina projekcijas režīms 3D skatā.
Ievaddati
Nav
Apstrāde
Katram izsaukumam maina no taisnleņķa uz perspektīvo projekciju un otrādi.
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.16. Skats – rādīt info logu:

1.16 tabula

Identifikators: showAbout
Mērķis
Funkcijai ir jāpaslēpj vai jāparāda info logs.
Ievaddati
Nav
Apstrāde
Paslēpj logu, ja tas ir redzams, un parāda, ja slēpts.
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.17. Krāsošana:

1.17 tabula

Identifikators: putPixels
Mērķis
Funkcijai ir aktīvajā tekstūrā jāieliek noteiktas krāsas pikseļi.
Ievaddati
R,G,B krāsas komponentes Otas pozīcija Otas rādiuss
Apstrāde
Aizkrāso pikseļus ar krāsu (R,G,B) otas rādiusā ap tās pozīciju.
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.1.18. 3D skats – renderēšana:

1.18 tabula

Identifikators: drawLoop
Mērķis
Funkcijai kalpo kā galvenais cikls.
Ievaddati
Nav
Apstrāde
Renderē ainu un apdarina pienākošos notikumus.
Izvaddati
Nav
Kļūdu paziņojumi
Nav

1.3.2. Ārējās saskarnes prasības

1.3.2.1. Lietotāja saskarne

Lietotāja saskarnei ir nepieciešamas divas būtiskas sastāvdaļas. Pirmkārt, attēlu redaktors – galvenais logs, kurā lietotājs spēj zīmēt tekstūras, mainīt aktīvo darba virsmu (tekstūru, ar kuru strādā), izsaukt visus pārējos logus. Otrkārt, 3D skata logs, kuru izsauc no galvenā loga un kurā ir redzams modelis. Treškārt – citi no galvenā loga izsaucamie rīki – otas rīkjosla un filtri ar lietotāja noteiktiem parametriem.

1.3.2.2. Veiktspējas prasības

3D skata renderēšanai ir jānotiek vismaz ar 30 kadriem sekundē.

Lietojamprogrammatūrai nav lietojamības laika ierobežojumu. Programmatūra ir uzstādīta uz lietotāja datora, tāpēc lietotājs to var izmantot jebkurā laikā.

1.3.2.3. Drošība

Tā kā šī ir bezsaistes programmatūra, kura neapstrādā sensitīvus datus, drošības jautājumā var tikai minēt nepieciešamību pēc mainīto datu integritātes. Lietojumprogrammatūrai ir jānodrošina nebojātu datu saglabāšana, proti, datnes, kuras ir modificētas ar šo lietojumprogrammu, saglabā savu struktūru, lai tās var lietderīgi izmantot citur.

2. PROGRAMMAS PROJEKTĒJUMA APRAKSTS

2.1. Ievads

2.1.1. Nolūks

Dokumenta nolūks ir iepazīstināt lasītāju ar programmas Hōsen projektējuma aprakstu (PPS). Šis dokuments ir tapis pamatojoties uz augstāk formulētās programmatūras prasību specifikācijas.

2.1.2. Darbības sfēra

Kvalifikācijas darba ietvaros ir izstrādāta programma kā arī tās dokumentācija, kurā esošie lietotāju stāsti apraksta lietojumprogrammatūras prasības un veicamās funkcijas. Programmatūras mērķis ir nodrošināt lietotājus, piemēram, 3D modelētājus un māksliniekus ar rīku, kas ļautu krāsot tekstūras, uzreiz redzot, kā tās vizuāli ietekmē objekta – modeļa izskatu.

2.1.3. Saistība ar citiem dokumentiem

Šis dokuments ir saistīts ar augstāk formulēto programmatūras prasību specifikāciju, kura kalpo par pamatu programmas projektējumam, kā arī LVS 72:1996 “Ieteicamā prakse programmatūras projektējuma aprakstīšanai”.

2.2.Dekompozīcijas apraksts

2.2.1. Moduļu dekompozīcija

Lietojumprogrammā tiek izšķirti trīs savstarpēji atkarīgi moduļi, kuri apmierina lietotājstāstos traktētās funkcionālās prasības.

2.2.1.1. Redaktora modulis

Identifikators: editorClass

Mērķis:

Moduļa uzdevums ir inicializēt visu saskarni: galveno logu – redaktora saskarni, kurā lietotājs strādā ar tekstūrām, kā arī ieiet programmas galvenajā ciklā. Redaktora modulis ļauj lietotājam izmantot tekstūru apstrādes modulī pieejamos algoritmus tekstūru rediģēšanai.

2.2.1.2. Tekstūru apstrādes modulis

Identifikators: textureClass

Mērķis:

Tekstūru apstrādes modulis. Moduļa uzdevums ir nodrošināt tekstūru datņu ievadi, apstrādi, ielādēšanu tekstūru buferī un izvadi. Modulī ir norealizēti algoritmi attēlu apstrādei ar izpludināšanas un asināšanas filtriem, konvolūcijas matricu. Otas funkcionalitātei eksistē funkcija noteikta pikseļu masīva (attēla) apgabala aizpildīšana ar noteiktas krāsas pikseļiem.

2.2.1.3. Renderēšanas modulis

Identifikators: viewportClass

Mērķis:

Moduļa funkciju uzdevums ir ēnotāju kompilēšana un modeļu ģeometrisko datu attēlošana 3D skata logā.

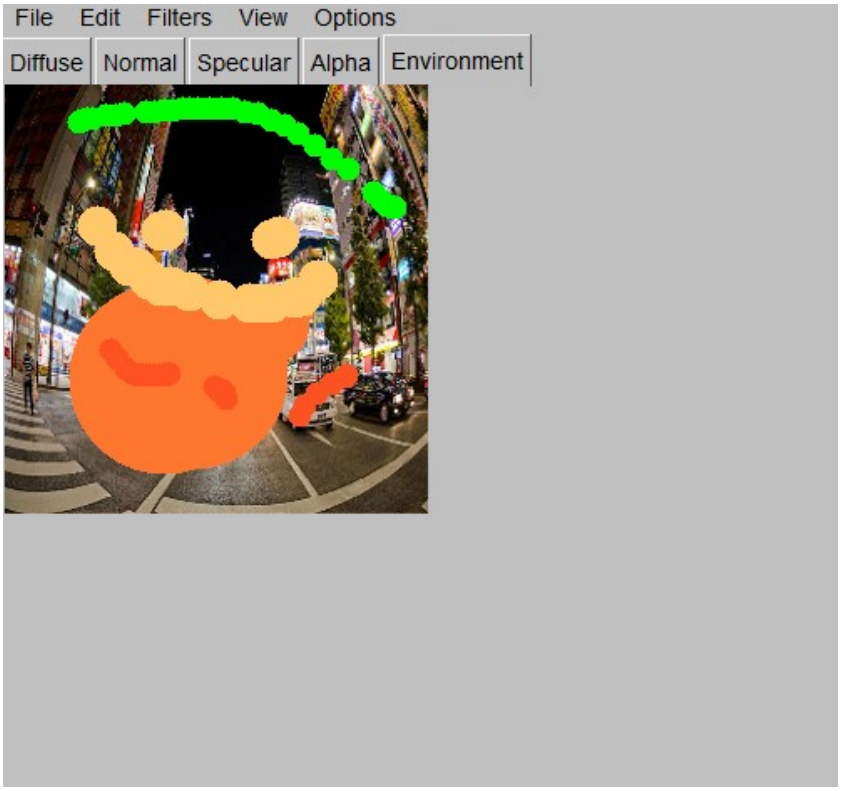
2.3. Saskarnes apraksts

2.3.1. Vispārējais apraksts

Lietotāja saskarni veido divi galvenie logi – 3D skata logs, kurā redzams modelis, un attēlu redaktora logs, kas vienlaikus kalpo par galveno logu, jo no tā var piekļūt visiem pārējiem logiem.

2.3.2. Galvenais logs

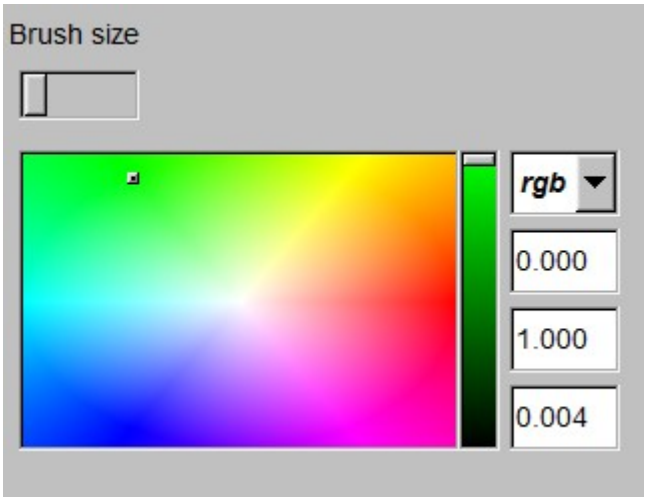
Attēlā ir redzams galvenais logs - redaktors. Tajā atrodas galvenā darba virsma ar visiem tekstūru veidiem kā arī izvēlnes josla. Attēli, kuri ir pārāk lieli lai ietilptu redaktora logā ir apstaigājami ar slīdņiem.



2.1. att. Galvenais logs – attēlu redaktors

2.3.3. Otas rīku kopa

Attēlā ir redzama otas rīku kopa ļauj mainīt otas parametrus: krāsu un rādiusu.



2.2. att. Otas rīku kopa

2.3.4. 3D skats

Renderēšana notiek atsevišķā logā. (Attēls 2.3)

2.3. att. Modelis renderēts 3D skatā

3. TESTĒŠANA

3.1. Ievads

Testēšanas dokumentācijā tiek aprakstīta lietotnes "*Hōsen*" testēšanas gaita.

3.1.1. Nolūks

Dokumenta nolūks ir iepazīstināt lasītāju – produkta pasūtītāju, ar lietotnes testēšanas norisi un tās rezultātiem.

3.2. Testēšanas plāns

Testēšanā tika pielietots baltās kastes princips un lietošanas scenāriji. Visa testēšana ir manuāla.

Tabula 3.1

N.p.k.	Tests	Vēlamais rezultāts	Rezultāts
1.	Galvenais logs, no kura var izsaukt 3D skatu, otu rīku kopu.	Atverot programmu, ir redzams attēlu redaktora logs un 3D skats. Redaktorā ir apskatāmas pēc noklusējuma ielādētās tekstūras un 3D skatā tiek zīmēts teksturēts modelis. Mainot sadaļas, mainās redzamā (aktīvā) tekstūra. Izvēlnes joslā tiek atvērtas apakšizvēlnes.	Tests veiksmīgs
2.	Apakšizvēlnē *File* ir jābūt redzamām opcijām atvērt (tekstūru vai datni), aizvērt modeli, izveidot jaunu tekstūru, saglabāt aktīvo tekstūru, aizvērt aktīvo tekstūru un iziet no programmas. Šīs	Korekta elementu hierarhija. Izvēloties opciju atvērt datni, parādās logs datnes atlasīšanai. Izvēloties opciju radīt jaunu tekstūru, parādās logs, kurā lietotājs ievada tekstūras izmēru. Izvēloties opciju aizvērt aktīvo tekstūru, aktīvās tekstūras vietā tiek	Tests veiksmīgs

	apakšizvēlnes elementus lietotājs darbina ar peli vai ar attiecīgu taustiņu kombināciju.	ielādēta un parādīta noklusējuma tekstūra. Izvēloties opciju saglabāt tekstūru, parādās logs datnes saglabāšanai.	
3.	Apakšizvēlnei *Edit* ir sava apakšizvēlne transformēšanai, kurā ir elementi: apakšizvēlne attēla rotēšanai, invertēt pa horizontālo asi, invertēt pa vertikālo asi.	Korekta elementu hierarhija. Izvēloties kādu no opcijām, ar aktīvo attēlu notiek transformācijas un izmaiņas ir uzreiz redzamas.	Tests veiksmīgs
4.	Apakšizvēlnes *Filters* elementi ļauj apstrādāt aktīvo attēlu ar izpludināšanas filtru, asināšanas filtru, padarīt melnbaltu, pārveidot par izciļņu karti un pielietot lietotāja definētu konvolūcijas matricu.	Korekta elementu hierarhija. Izvēloties filtru, parādās attiecīgais logs.	Tests veiksmīgs
5.	Apakšizvēlnes *View* elementi kontrolē 3D skata un otas rīku redzamību.	Korekta elementu hierarhija. Izvēloties kādu no opcijām, attiecīgais logs tiek paslēpts, ja tas ir redzams, un parādīts, ja paslēpts.	Tests veiksmīgs
6.	Apakšizvēlnes *Options* elementi ir apakšizvēlne, kuras elementi kontrolē 3D skata projekcijas veidu (perspektīvo vai ortogrāfisko projekciju), opcija ieslēgt vai izslēgt no kameras projām vērsto skaldņu atmešanu un opcija, kas parāda logu ar informāciju par programmu.	Korekta elementu hierarhija. 3D skatā mainās projekcija. 3D skatā netiek renderētas skaldnes, kuru normāle vērsta prom no kameras. Parādās *About* logs	Tests veiksmīgs

7.	Tekstūras atlasīšanas loga kreisajā pusē ir redzams mapes saturs. Labajā pusē – tekstūras priekšskatījums. Apakšā atrodas izvēles rūtiņa, kura nosaka, vai priekšskatījums tiek rādīts. Virs mapes satura jābūt nolaižamai izvēlnei, kurā var mainīt atlasīto datnes tipu. Blakus labajā pusē ir cita nolaižamā izvēlne, kurā lietotājs var pievienot tekošo mapi favorītu sarakstam. Apakšā ir redzams tekstlodziņš ar izvēlēto ceļu, pogas apstiprināšanai un atcelšanai.	Korekts loga izskats. Ar peli izvēloties elementus mapē un pārvietojoties pa direktorijiem, tiek atjaunināts ceļa tekstlodziņš. Kad ir izvēlēts ceļš uz datni, apstiprināšanas poga kļūst nospiežama. Apstiprināšanas pogu nospiežot, izvēlētā tekstūra nomaina aktīvo. Nospiežot atcelšanas pogu vai krustiņa vadīkla, logs tiek aizvērts, datne netiek nolasīta.	Tests veiksmīgs
8.	Modeļa atlasīšanas loga kreisajā pusē ir redzams mapes saturs. Labajā pusē – priekšskatījums, kurā datne redzama teksta veidā. Apakšā atrodas izvēles rūtiņa, kura nosaka, vai priekšskatījums tiek rādīts. Virs mapes satura jābūt nolaižamai izvēlnei, kurā var mainīt atlasīto datnes tipu. Blakus labajā pusē ir cita nolaižamā izvēlne, kurā lietotājs var pievienot tekošo mapi favorītu sarakstam. Apakšā ir redzams tekstlodziņš ar izvēlēto ceļu,	Korekts loga izskats. Ar peli izvēloties elementus mapē un pārvietojoties pa direktorijiem, tiek atjaunināts ceļa tekstlodziņš. Kad ir izvēlēts ceļš uz datni, apstiprināšanas poga kļūst nospiežama. Apstiprināšanas pogu nospiežot, 3D skatā modelis tiek aizstāts ar jauno. Nospiežot atcelšanas pogu vai krustiņa vadīkla, logs tiek aizvērts, datne netiek nolasīta.	Tests veiksmīgs

	pogas apstiprināšanai un atcelšanai.		
9.	Jaunas tekstūras izveides logā ir redzams tekstlodziņš, kurā ir jānorāda izveidojamās tekstūras izmērs.	Korekts loga izskats. Ja tekstūras izmērs netiek validēts, apstiprināšanas pogas nospiešanai nav rezultāta. Nospiežot atcelšanas pogu vai krustiņa vadīkla, logs tiek aizvērts.	Tests veiksmīgs
10.	Tekstūras saglabāšanas loga kreisajā pusē ir redzams mapes saturs. Labajā pusē – tekstūras priekšskatījums. Apakšā atrodas izvēles rūtiņa, kura nosaka, vai priekšskatījums tiek rādīts. Virs mapes satura jābūt nolaižamai izvēlnei, kurā var mainīt atlasīto datnes tipu. Blakus labajā	Korekts loga izskats. Ar peli izvēloties elementus mapē un pārvietojoties pa direktorijiem, tiek atjaunināts ceļa tekstlodziņš. Kad ir izvēlēts ceļš uz datni, apstiprināšanas poga kļūst nospiežama. Apstiprināšanas pogu nospiežot, aktīvā tekstūra tiek ierakstīta jaunā datnē vai pārraksta jau esošu datni. Nospiežot atcelšanas pogu vai krustiņa vadīkla, logs tiek aizvērts, datne rakstīta.	Tests veiksmīgs
11.	Izpludināšanas filtra logā ir redzams slīdnis, kas kontrolē, cik reižu filtrs tiek pielietots, apstiprināšanas un atcelšanas pogas.	Ievadot korektas vērtības un nospiežot apstiprināšanas pogu, aktīvā tekstūra tiek apstrādāta ar filtru un izmaiņas ir redzamas redaktora skatā.	Tests veiksmīgs
12.	Asināšanas filtra logā ir redzams slīdnis, kas kontrolē, cik reižu filtrs tiek pielietots, apstiprināšanas un atcelšanas pogas.	Ievadot korektas vērtības un nospiežot apstiprināšanas pogu, aktīvā tekstūra tiek apstrādāta ar filtru un izmaiņas ir redzamas redaktora skatā.	Tests veiksmīgs

13.	Konvolūcijas matricas logā ir redzami deviņi teksta lodziņi, apstiprināšanas un atcelšanas pogas.	Ievadot korektas vērtības un nospiežot apstiprināšanas pogu, aktīvā tekstūra tiek apstrādāta ar filtru un izmaiņas ir redzamas redaktora skatā.	Tests veiksmīgs
14.	Otas rīku logā ir redzams slīdnis otas izmēram un rīku kopa krāsas atlasīšanai.	Mainot otas izmēru vai krāsu, parametri, kuri tiek izmantoti krāsošanai tiek atjaunināti, par ko var pārliecināties, pakrāsojot tekstūru.	Tests veiksmīgs
15.	Krāsošanas darbvirsmas kreisajā augšējā stūrī tiek zīmēta tekstūra. Ja tekstūra pilnībā ietilpst redaktora logā, slīdņi, ar kuru pārbīdīt tekstūru, netiek rādīti.	Tekstūrai kura neiekļaujas redaktora logā jābūt slīdņiem. Krāsojot tekstūru, iekrāsotajiem pikseļiem ir jāatbilst peles pozīcijai un otas rīku parametriem: otas rādiusam un krāsai.	Tests veiksmīgs
16.	3D skata logs ļauj lietotājam pārvietoties telpā ar *WASD QZ* un peles palīdzību. *O* un *I* taustiņi maina skata leņķi. *P* maina projekcijas režīmu. Virzieni ir intuitīvi: *W*/*D* – kustība ±*x* virzienā, *A*/*S* - ±*y* virzienā, *Q*/*Z* ±z virzienā.	Ģeometriskās transformācijas ir korektas un lietotājs var brīvi pārvietoties 3D telpā ar tastatūras un peles palīdzību, mainīt skata leņķi un projekcijas režīmu.	Tests veiksmīgs

3.3. Secinājumi

Testēšanas laikā tika atrastas kļūdas, kuras tika izlabotas, piemēram, nepareizi peles pozīcijas aprēķini, kuru rezultātā netika iekrāsoti pareizie pikseļi. FLTK satvarā peles pozīcija tiek aprēķināta relatīvi pret logu, kurš šo notikumu saņem, tāpēc šīs nobīdes ir jākompensē, kas ne vienmēr notika pareizi.

4. PROJEKTA ORGANIZĀCIJA

Lietojumprogrammatūra "*Hōsen*" tika izstrādāta pēc jauktas izstrādes metodes, kombinējot spējo un ūdenskrituma modeli. Svarīgākās prasības bija skaidri zināmas, kurām izstrādes procesā ir veikta piedare.

Projekta ietvaros tika gūta pieredze darbā ar *FLTK* satvaru un tā grafisko saskarņu redaktoru.

Izstrādes noslēgumā tika veikta testēšana. Visos izstrādes posmos autors ir rīkojies patstāvīgi.

5. KONFIGURĀCIJAS PĀRVALDĪBA

Gan programmkoda versiju pārvaldībai, gan dokumentācijai tika izmantota versiju kontrole *Git*, un programmkods glabāts *BitBucket* repozitorijā. Programmkods un dokumentācija tika glabāti darba direktorijā. Būtiskas izmaiņas tika ielādētas repozitorijā vismaz reizi nedēļā, proti, ik pēc katra sprinta beigām.

6. KVALITĀTES NODROŠINĀŠANA

Nostādot mērķi nodrošināt izstrādājamās lietojumprogrammatūras kvalitāti, dokumentācija ir tapusi saskaņā ar valsts standartiem programminženierijā LVS 68:1996 "Programmatūras prasību specifikācijas ceļvedis" un LVS 72:1996 "Ieteicamā prakse programmatūras projektējuma aprakstīšanai". Izstrādes sākumā izvirzītās projektu virzošās pamatprasības, kā arī izstrādes laikā ieviesti nelieli pamatprasību labojumi un papildprasības ir izpildīti. Izstrādes virzību un kvalitāti nodrošināja mērķtiecīga visu funkcionālo prasību apmierināšana. Izstrādes beigās tika veikta rūpīga testēšana. Programmas kods ir rūpīgi komentēts ar domu maksimizēt tā lasāmību. Par izstrādes vidi kalpoja *Microsoft Visual Studio 2015*.

7. DARBIETILPĪBAS NOVĒRTĒJUMS

Uzsākot darbu, lai gan pieredze ar *OpenGL* bibliotēku un grafikas programmēšanu bija ievērības cienīga, iepriekšējā pieredzē tā ieņēma salīdzinoši nelielu lomu kā daļa no lielāka izstrādē pielietota satvara, piemēram, *SDL* vai *Qt*. Šī kvalifikācijas darba gadījumā, sākotnēji programmatūra tika izstrādāta izmantojot tikai *GLFW*, *GLEW* un *GLM* bibliotēkas, tā vietā, lai izmantotu satvaru. Šī iemesla dēļ, paredzētā darbietilpība ir nekas vairāk par minējumu.

Taču aptuveni pusotru mēnesi pirms darba nodošanas tika pieņemts lēmums pārveidot lietotāja saskarni ar *FLTK* satvara palīdzību. Šī izvēle tiek pamatota ar to, ka vecā saskarne nebija lietotājdraudzīga un izstrādātājam paplašināt funkcionalitāti būtu bijis daudz sarežģītāk, piem., pievienot jaunu slīdni vecajā realizācijā prasītu aptuveni 30 koda rindiņas, savukārt ar *FLTK* – vienu, turklāt, vēl viena *FLTK* priekšrocība ir tā *WYSIWYG* redaktors, kas ļauj ļoti ātri projektēt saskarni un ģenerēt tās kodu. Šīm izmaiņām paredzamā darbietilpība bija četras personnedēļas, taču tuvāk noslēgumam kļuva skaidrs, ka novērtējums ir bijis aplams, jo bija jāatmet un jāpārraksta ļoti liela daļa koda. Kopējais darbietilpības novērtējums, kurā tiek salīdzināts minējums un reālais patērētais laiks:

- 3D skats (paredzētais – 5 personned.; patiesais – >4 personned.)
- Redaktors (paredzētais – 5 personned,; patiesais - >5 personned.)
- Filtri, attēlu apstrāde (paredzētais – 5 personned.; patiesais – 3 personned.)
- Kopā pirms *FLTK* ieviešanas (paredzētais – ~15 personned.; ~13 personned.)
- *FLTK* ieviešana (paredzētais – 3 personned.; >5 personned.)
- Dokumentācija (paredzētais – 1 personned.; >1 personned.)
- **Kopā ieguldītais laiks** - ~19 personnedēļas (>4 personmēneši)

Gan programmkoda versiju pārvaldībai, gan dokumentācijai tika izmantota versiju kontrole Git, un programmkods glabāts *BitBucket* repozitorijā. Būtiskas izmaiņas tika ielādētas repozitorijā vismaz reizi nedēļā.

SECINĀJUMI

Kvalifikācijas darba ietvaros izstrādāta lietojumprogramma “Hōsen”, kura pilda aprakstītās funkcijas un apmierina izvirzītās prasības. Tā dod lietotājam iespēju reāllaikā apskatīt, kā tekstūru veidi maina renderēta 3D modeļa izskatu. “Hōsen” ir paredzēta gan profesionāliem 3D māksliniekiem, gan amatieriem. Kvalifikācijas darba ietvaros ir apgūts FLTK satvars. “Hōsen” izstrāde tiks turpināta ar mērķi paplašināt lietotājam pieejamo rīku klāstu, kopējo lietotnes funkcionalitāti un optimizēt attēlu apstrādes algoritmus.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. LVS 68:1996 Programmatūras prasību specifikācijas ceļvedis
2. LVS 72:1996 Ieteicamā prakse programmatūras projektējuma aprakstīšanai
3. GLM bibliotēkas dokumentācija - https://glm.g-truc.net/0.9.4/api/index.html
4. Otas rīka programmēšana - https://graphics.stanford.edu/courses/cs248-05/ho4/
5. Diskusija par izciļņu kartes ģenerēšanu - https://www.gamedev.net/topic/475213-generate-normal-map-from-heightmap-algorithm/
6. Izciļņu kartes ģenerēšanas algoritms - http://sunandblackcat.com/tipFullView.php?l=eng&topicid=6
7. Dažādi OpenGL materiāli - https://learnopengl.com/
8. FLTK saskarnes redaktors FLUID - http://www.fltk.org/doc-1.3/fluid.html
9. FLTK palīgmateriāli, piemēri - http://seriss.com/people/erco/fltk/
10. FLTK dokumentācija - http://www.fltk.org/doc-1.3/
11. OpenGL mācību materiāls, buklets - https://www.opengl.org/sdk/docs/reference_card/opengl45-reference-card.pdf
12. GLSL grafikas programmēšanas valodas specifikācija - https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.50.pdf
13. Terminoloģija - http://termini.lza.lv/

PIELIKUMS

textureClass.h

```
#pragma once
#include <cstdio>
#include <cstdlib>
#include <GL/glew.h>
#include <glm/glm.hpp>
#include <GLFW/glfw3.h>
#include <string>
#include <fstream>
#include <functional>
#include <Windows.h>
#include <FL/Fl_BMP_Image.H>

// HELPER FUNCTIONS


class texture
{

private:

	GLuint textureID;

	unsigned char * pixels;

	DWORD size;
	DWORD width;
	DWORD height;

	DWORD depth;

	DWORD type = 0;

	FLOAT _alpha = 0.8f;
public:
	Fl_BMP_Image * img;

	// HELPER FUNCTIONS

	// Help01 - Compute pixel index given x and y coordinate
	int pid(int x, int y)
	{
		if (x < 0) x + width;
		if (y < 0) x + height;
		return (y%height * width * depth) + (x%width * depth);
	}

	// Help02 - Get size
	DWORD getSize()
	{
		return size;
	}

	// Help03 - Get height
	DWORD getHeight()
	{
		return height;
	}

	// Help03 - Get width
```

```
	DWORD getWidth()
	{
		return width;
	}

	// Help04 - Get pixel array
	unsigned char * getPixelArr()
	{
		return pixels;
	}

	// Help05 - Get depth
	DWORD getDepth()
	{
		return depth;
	}

	// COPYING/BINDING texture data

	// Text01 - Generate texture
	GLuint genTexture()
	{
		glGenTextures(1, &textureID);
		glBindTexture(GL_TEXTURE_2D, textureID);
		glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, img->w(), img->h(), 0, GL_RGB,
GL_UNSIGNED_BYTE, pixels);

		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_LINEAR);
		glGenerateMipmap(GL_TEXTURE_2D);

		return textureID;
	}

	// Text02 - Delete pixel data
	void clear()
	{
		delete img;
		return;
	}

	// Text03 - Update texture buffer with new pixel data
	void reload()
	{
		glBindTexture(GL_TEXTURE_2D, textureID);
		glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, img->w(), img->h(), 0, GL_RGB,
GL_UNSIGNED_BYTE, pixels);
		return;
	}

	// Input/Output

	// IO01 Load BMP - loads pixel data
	void loadBMP(const char * imagepath)
	{
		img = new Fl_BMP_Image(imagepath);

		size = img->w()*img->h() * img->d();
		width = img->w();
		height = img->h();
		depth = img->d();
		pixels = (unsigned char *)img->array;
```

```
                printf("loaded image %d by %d px size %d bytes \n", width, width,
width*height * 3);
                return;
        }

        // IO02 Load texture and upload to texture buffer
        void loadTexture(const char * imagepath)
        {
                loadBMP(imagepath);
                reload();
                return;
        }


        // IO03 Write pixel data to BMP file
        int saveBMP(std::string path)
        {
                unsigned char fileheader[14] = { 'B','M',0,0,0,0,0,0,0,0,54,0,0,0 };
                unsigned char infoheader[40] = { 40,0,0,0, 0,0,0,0, 0,0,0,0, 1,0, 24,0 };
                unsigned char padding[3] = { 0,0,0 };

                FILE * f = fopen(path.c_str(), "wb");
                fwrite(fileheader, 1, 14, f);
                fwrite(infoheader, 1, 40, f);
                for (int i = 0; i<height; i++)
                {
                        fwrite(pixels + (width*(height - i - 1) * 3), 3, width, f);
                        fwrite(padding, 1, (4 - (width * 3) % 4) % 4, f);
                }
                return 1;
        }

        // IMAGE PROCESSING

        // Filter01 Generic convolution
        void convolve(float matrix[], float coef)
        {
                int temp[9];
                char accumulator;
                unsigned char *tmp = (unsigned char*)malloc(size);
                memcpy(tmp, pixels, size);

                for (int x = 0; x < width; x++) // each X
                {
                        for (int y = 0; y < height; y++) // each Y
                        {
                                for (int offset = 0; offset < depth; offset++) // each RGB
value
                                {
                                        temp[0] = (int)tmp[pid(x - 1, y - 1) + offset] *
coef*matrix[0];
                                        temp[1] = (int)tmp[pid(x, y - 1) + offset] *
coef*matrix[1];
                                        temp[2] = (int)tmp[pid(x + 1, y - 1) + offset] *
coef*matrix[2];

                                        temp[3] = (int)tmp[pid(x - 1, y) + offset] *
coef*matrix[3];
                                        temp[4] = (int)tmp[pid(x, y) + offset] *
coef*matrix[4];
                                        temp[5] = (int)tmp[pid(x + 1, y) + offset] *
coef*matrix[5];

                                        temp[6] = (int)tmp[pid(x - 1, y + 1) + offset] *
coef*matrix[6];
```

```
                                        temp[7] = (int)tmp[pid(x, y + 1) + offset] *
coef*matrix[7];
                                        temp[8] = (int)tmp[pid(x + 1, y + 1) + offset] *
coef*matrix[8];

                                        // temp is now the computed kernel - add up all
values
                                        accumulator = 0;
                                        for (int i = 0; i < 9; i++)
                                        {
                                                accumulator += temp[i];
                                        }
                                        pixels[pid(x, y)+offset] = accumulator;

                                }
                        }
                }
                img->uncache();
                delete(tmp);
                return;
        }

        // Filter02 Generic normalmap
        void normalmap()
        {
                float Z_x;
                float Z_y;
                int index;
                glm::vec3 N, A, B;

                unsigned char *heightmap = (unsigned char*)malloc(size);
                memcpy(heightmap, pixels, size);

                for (int x = 0; x < width; x++) // each X
                {
                        for (int y = 0; y < height; y++) // each Y
                        {
                                index = pid(x, y);

                                // along x
                                if (x == 0 || x == width - 1)
                                {
                                        Z_x = 0.001f * heightmap[pid(x, y)];
                                }
                                else
                                {
                                        Z_x = 0.001f * (heightmap[pid(x + 1, y)] -
heightmap[pid(x - 1, y)]);
                                }
                                // along y
                                if (y == 0 || y == height - 1)
                                {
                                        Z_y = 0.001f * heightmap[pid(x, y)];
                                }
                                else
                                {
                                        Z_y = 0.001f * (heightmap[pid(x, y + 1)] -
heightmap[pid(x, y - 1)]);
                                }

                                A = glm::vec3(1.0f, 0.0f, Z_x);
                                B = glm::vec3(0.0f, 1.0f, Z_y);
                                N = glm::cross(A, B);
                                N = glm::normalize(N);
```

```
                        pixels[index] = ((N.x+1.0f) / 2) * 255;
                        pixels[index + 1] = ((N.y+1.0f) / 2) * 255;
                        pixels[index + 2] = ((N.z+1.0f) / 2) * 255;
                }
        }

        delete (heightmap);
        heightmap = nullptr;;
        img->uncache();
        return;
}

// Filter03 Quick desaturate
void desaturate()
{
        int temp_grey;
        for (int i = 0; i < size; i += 3)
        {
                temp_grey = pixels[i];
                for (int offset = 0; offset < depth; offset++)
                {
                        pixels[i + offset] = temp_grey;
                }
        }
        img->uncache();
        return;
}

// Transform01 Image transform
void transform(int type)
{
        unsigned char tmp;
        int index = 0;
        int index_temp;
        if (type < 2)
        {
                for (int x = 0; x < width*depth; x += depth)
                {
                        for (int y = 0; y < height*depth; y += depth)
                        {
                                index_temp = pid(x, y);
                                for (int offset = 0; offset < depth; offset++)
                                {
                                        tmp = pixels[pid(x, y) + offset];
                                        pixels[pid(x, y) + offset] = pixels[pid(x, y)
+ offset];
                                        pixels[pid(x, y) + offset] = tmp;
                                }
                        }
                }
        }

        // rotate +90
        if (type == 0)
        {
                for (int x = 0; x < width; x++) {
                        for (int y = 0; y < height / 2; y++) {
                                for (int k = 0; k < 3; k++)
                                {
                                        tmp = pixels[pid(x, y) + k];
                                        pixels[pid(x, y) + k] = pixels[pid(x, width -
y - 1) + k];
                                        pixels[pid(x, width - y - 1) + k] = tmp;
                                }
                        }
```

```
                }
            }

            // rotate -90
            if (type == 1)
            {
                for (int y = 0; y < height; y++) {
                    for (int x = 0; x < width / 2; x++) {
                        for (int k = 0; k < 3; k++)
                        {
                            tmp = pixels[pid(y, x) + k];
                            pixels[pid(y, x) + k] = pixels[pid(y, height
- x - 1) + k];
                            pixels[pid(y, height - x - 1) + k] = tmp;
                        }
                    }
                }
            }

            // flip - vertically
            if (type == 2)
            {
                for (int x = 0; x < width; x++) {
                    for (int y = 0; y < height / 2; y++) {
                        for (int k = 0; k < 3; k++)
                        {
                            tmp = pixels[pid(x, y) + k];
                            pixels[pid(x, y) + k] = pixels[pid(x, width -
y - 1) + k];
                            pixels[pid(x, width - y - 1) + k] = tmp;
                        }
                    }
                }
            }

            // flip - horizontally
            if (type == 3)
            {
                for (int y = 0; y < height; y++) {
                    for (int x = 0; x < width / 2; x++) {
                        for (int k = 0; k < 3; k++)
                        {
                            tmp = pixels[pid(y, x)+k];
                            pixels[pid(y, x)+k] = pixels[pid(y, height -
x - 1)+k];
                            pixels[pid(y, height - x - 1)+k] = tmp;
                        }
                    }
                }
            }
            img->uncache();
            return;
        }

        // Paint01 Put Pixels
        void putPixels(
            // brush radius
            unsigned char brushRadius,
            // rgb values
            unsigned char r, unsigned char g, unsigned char b,
            // brush center
            unsigned short x, unsigned short y
        )
        {
```

```
            for (int k = x - brushRadius; k < x + brushRadius; k++) // each X within
square
            {
                  for (int j = y - brushRadius; j < y + brushRadius; j++) // each Y
within square
                  {
                        // Check whether pixel falls within area of inscribed
circle AND not outside
                        if ((k - x)*(k - x) + (j - y)*(j - y) <=
brushRadius*brushRadius)
                        {
                              // Check whether pixel is not outside the image
                              if (!((x - brushRadius < 0) || (x + brushRadius >
width) || (y - brushRadius < 0) || (y + brushRadius > height)))
                              {
                                    pixels[pid(k, j)] = r;
                                    pixels[pid(k, j) + 1] = g;
                                    pixels[pid(k, j) + 2] = b;
                              }
                        }
                  }
            }
            img->uncache();
            return;
      }
};
```

Kvalifikācijas darbs „***Hōsen – rīks 3D modeļu tekstūru krāsošanai un uzklāšanai*** ” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: ***Čaks, Levits*** _____ **___.09.2017**.

Rekomendēju darbu aizstāvēšanai

Darba vadītāj**s/a**: ***Dr. Dat, Jānis, Zuters*** _____ **___.09.2017.**

Recenzents:

Darbs iesniegts 12.09.2017.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.09.2017. prot. Nr. _____

Komisijas sekretārs(-e): _____