VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

NURBS paviršiai mobiliuosiuose įrenginiuose

Atliko: IIM kurso studentai:

Vaida Masiulionytė

Tautvydas Dagys

Darbo vadovas:

doc. Rimvydas Krasauskas

Vilnius
2006

TURINYS

Anotacija

Darbe yra apžvelgiamos NURBS paviršių atvaizdavimo realizavimo problemos mobiliesiems įrenginiams. Pagrindinis darbo tikslas yra įsitikinti tokios realizacijos galimumu, bei surasti kiek įmanoma efektyvesnį būdą, išsprendžiantį mobiliųjų platformų realizacijų problemas. Dar vienas tikslas, sukurti prototipinį sprendimą, padengiantį NURBS standartą. Analizei pasirinkta M3G mobiliosios grafikos standartas, kuris šiuo metu yra labiausiai paplitęs. Realizacijų palyginimui buvo pasirinkti du algoritmai: Oslo ir tiesioginis. Atlikus veikimo analizę nustatyta, jog tiesioginis algoritmas yra efektyvesnis, kurio pagrindu buvo išbaigtas NURBS standartas, realizuojant Trimmed NURBS modelius.

Summary

NURBS surfaces on mobile devices

In the thesis it is analyzed the problems related with NURBS rendering implementation for mobile devices. The main goal of the thesis is to make find out if such implementation is possible at all and to find most effective way to implement it. One more goal is to create a prototype implementation for NURBS standard including Trimmed NURBS. M3G mobile graphics platform was chosen for analysis, which is mostly adopted across the vendors nowadays. There was chosen two rendering algorithms for analysis purposes: Oslo and directional. During analysis it was discovered, that Directional algorithm is more effective than Oslo, and it was adapted to prototype solution – Trimmed NURBS.

Įvadas

Šiuo metu viena iš labiausių besivystančių technologijų yra mobiliosios technologijos. Mobilieji telefonai, delniniai kompiuteriai, GPS įrenginiai bei begalė kitų tampa vis plačiau naudojami, jų galimybės bei pajėgumai didėja labai sparčiai. Šiuo metu, kai išvystytos pagrindinės tokių įrenginių funkcijos, toliau vystimasis pasisuka daugialypės terpės link. Tai aukštos kokybės garsas, aukštos kokybės video bei trimatė grafika.

Trimatė grafika mobiliesiems įrenginiams yra pradinėje vystimosi fazėje. Šiuo metu yra keletas standartų išvystytų atskirų gamintojų standartų, OpenGL ES standartas, kurį galima panaudoti taip pat tik kai kuriuose įrenginiuose bei M3G Java standartas, kuris yra plačiausiai palaikomas visų mobiliųjų įrenginių gamintojų ir diegiamas beveik visuose naujai išleidžiamuose gaminiuose. Dėl jo plačiausio paplitimo, M3G standartas yra pats įdomiausias tyrinėjimams bei analizei. Šiame standarte yra realizuoti paprastieji objektai (Mesh), transformacijos, apšvietimai, kameros, animacijos. Tai yra pagrindiniai trimatės grafikos objektai, tačiau trimatėje grafikoje taip pat plačiai yra taikomi parametrizuotieji paviršiai, tokie kaip NURBS paviršiai.

NURBS (Non-Uniform Rational B-Splines) paviršiai yra gerai išvystyta 3D modeliavimo technologija, kuri šiuo metu jau tapusi standartu „de facto“. NURBS yra integruoti į daugumą 3D modeliavimo, CAD/CAM sistemų. Nors ir plačiai naudojami praktikoje NURBS paviršiai dar nėra „perkelti“ į mobiliuosius įrenginius. Viena iš pagrindinių to priežasčių yra ta, jog NURBS modelių atvaizdavimo bei redagavimo mechanizmai reikalauja didelių resursų sąnaudų. Kadangi mobiliųjų įrenginių galimybės sparčiai vystosi, tai jau tik laiko klausimas, kada mobiliuosiuose įrenginiuose įsitvirtins antrosios kartos trimatė grafika. Antrajai kartai ir gali būti priskirti parametrizuotieji paviršiai.

Pagrindiniai NURBS paviršių privalumai yra tie, jog aprašyti NURBS modeliui reikia žymiai mažiau vietos negu standartinio aprašo trimačiam modeliui (išreikštam viršūnėmis, trikampiais ir normalėmis). Palyginimui paimkime NURBS modelį iš 308 kontrolinių taškų, kuriam dar papildomai reikia aprašyti 160 mazgų, bei nurodyti jo eilę abiem kryptimis. Tuo tarpu standartinėje išraiškoje tokiam modeliui aprašyti reiktų 2500 trikampių, kas susiveda į 2550 viršūnių, tiek pat normalių bei 20 įrašų apie trikampių juostų ilgius. Sudauginę dydžius iš jų sudedamųjų dalių gauname, kad NURBS modeliui aprašyti reikia 308*4+160+2 = 1394

skaitinių vienetų (pavyzdžiui double tipo išraiškoje), o standartinėje išraiškoje aprašytam: 2550*3*2+20 = 15320 skaitinių vienetų. Gauname apie 10 kartų didesnius skaičius. NURBS technologiją galima traktuoti kaip 3D objektų glaudinimo mechanizmą iš ko išsiveda šios technologijos trūkumas. Kuo objektai bus daugiau suspausti ir sutaupyta vieta, tuo daugiau reikės apdorojimo pajėgumo, tokiems objektams atstatyti į pradinę jų formą. Taigi atvaizduoti NURBS modelį, kuris užima žymiai mažiau vietos reikia žymiau daugiau apdorojimo pajėgumų. Augant visiems mobiliųjų įrenginių pajėgumams, saugojimo vieta vis tiek išlieka kritiškiausias veiksnys, ypatingai kalbant apie ryšio perdavimo kanalus (kur visviena išlieka perduodamos informacijos srauto apribojimai, bei dažniausiai tai yra apmokestinama). Taigi šiuo atžvilgiu NURBS modeliai yra labai gera išeitis 3D grafikai mobiliuosiuose įrenginiuose. Vienintelis čia reikalingas veiksnys yra apdorojimo sparta (kas susiveda į procesoriaus galimybes, operatyviosios atminties spartą bei kiekį, atskirus apdorojimo įrenginius dedikuotus būtent trimatei grafikai ir t.t.)

Taigi pagrindinis šio darbo tikslas yra surasti pakankamai efektyvią NURBS realizaciją mobiliesiems įrenginiams, o iš to seka tokie išvestiniai tikslai:

- Įsitikinti ar tai įmanoma realizuoti turimomis priemonėmis
- Parinkti pakankamai efektyvų realizacijos būdą
- Įvertinti gauto rezultato atitikimą poreikiams, poreikius bei galimybes.

Šio darbo metu buvo išnagrinėtas teorinis NURBS paviršių pagrindas, mobiliųjų įrenginių aplinka bei galimybės, išanalizuoti realizaciniai niuansai. Realizuoti prototipai, praktiškai išbandyti NURBS realizacijas bei jų efektyvumą. Kad pilnai realizuoti NURBS standartą, papildomai buvo išnagrinėti ir realizuoti iškarpytųjų NURBS (Trimmed NURBS) atvaizdavimo algoritmai. Iškarpytieji NURBS paviršiai yra apibendrinantis NURBS modelių standartas, įgalinantis atvaizduoti, išsaugoti ir perduoti bet kokius gamtoje ir pramonėje sutinkamus tūrinius objektus.

Darbas organizuotas tokia seka: Mobiliųjų platformų bei rinkos apžvalga, 3D grafikos mobiliuosiuose įrenginiuose apžvalga; NURBS paviršių teorinė apžvalga, apimanti pačių NURBS bei Trimmed NURBS paviršių teorinį pagrindą, jų išraiškas, atvaizdavimo algoritmus; Praktinė atvaizdavimo algoritmų realizacija, jų palyginimas; Trimmed NURBS realizacija; Apibendrinimas.

Mobiliųjų įrenginių apžvalga

Mobiliųjų įrenginių rinka yra viena iš dinamiškiausių rinkos segmentų šiai dienai. Rinka sparčiai plečiasi, mobiliųjų telefonų gamintojai išleidžia vis daugiau įrenginių gebančių apdoroti 3D grafiką. Šių įrenginių tobulėjimo tendencijos labai džiuginančios. Trumpai galima apibrėžti – viskas auga ir spartėja. Telefonų ekranai didėja, didėja ir jų raiška, bei spalvinės galimybės, mažėja inertiškumas. Didėja saugojimo galimybės, plečiasi vidinės atmintinės, atsiranda įmontuojami kietieji diskai, bei išorinės atminties korteles. Dabar vidutiniškai telefono saugojimo atmintis siekia iki šimto megabaitų, o maksimaliai gali siekti, net iki kelių gigabaitų. Aišku neatsilieka ir operatyviosios atminties talpa ir sparta, taip pat kaip ir centrinis procesorius.

Kaip pagrindinis 3D grafikos standartas mobiliesiems įrenginiams yra vystomas M3G Java (JSR-184). Jo panaudojimas ir taikymas vis labiau aktualėja. Jau dauguma gamintojų šį standartą naudoja, standartiniams savo telefonų vartotojo interfeisams programuoti. Dauguma šiandieną išleidžiamų mobiliųjų telefonų yra su trimačiais animuotais meniu bei kitais ekranais. Taip pat atsiranda trimatės ekrano užsklandos, bei kiti elementai. Taip pat populiarėja ir 3D žaidimai mobiliesiems telefonams.

Kaip matosi, kad mobiliosios platformos evoliucionuoja tokia pačia kryptimi, kaip ir personalinių kompiuterių platformos per savo istoriją, tačiau augimo greitis yra žymiai didesnis. Žinoma čia jau padeda patirtis ir įdirbis personalinių kompiuterių platformų tobulinime ir spartinime.

3D grafikos mobiliuosiuose įrenginiuose apžvalga

Nepaisant to jog su trimate grafika susiję įrenginiai bei programos mobiliuosiuose telefonuose maždaug 10 atsilieka nuo visų šiuolaikinių 3D technologijų, tačiau jo vystimosi tempai yra du ar tris kartus greitesni nei jo pirmtako. Dėl to trimatė grafika mobiliems telefonams gali pasiekti masinio vartojimo lygį per artimiausius keletą metų. Ypatingai tai paspartinti turėtų dedikuota trimatės grafikos aparatinė įranga, kaip 3D spartintuvai, 3D atmintinė ir t.t. Tai sujungus su šiandien jau egzistuojančiais trimatės grafikos standartais, vartotojas gautų galimybę kurti įspūdingas interaktyvias programas, ko nebuvo galima daryti iki šiol.

Labiausiai džiuginanti naujiena yra ta jog pagaliau rinkoje atsirado mobilieji telefonai su aparatūriniu 3D grafikos spartintuvu, bei atskiru procesoriumi. Tai galima sakyti prasidėjo naujas etapas mobiliojoje 3D grafikoje. Vienas iš pirmųjų su aparatiniu 3D spartinimu pasirodžiusių įrenginių yra Sony Ericsson W900 Walkman. Jis pristatytas rinkai 2005 metų spalio mėnesį. Žinoma aparatinis spartinimas atveria visiškai naujas galimybes 3D kūrimui, bet savo ruožtu atsineša ir naujų problemų. Aišku šiek tiek paradoksalu, nes pirmosios aparatūrinės 3D grafikos atvaizdavimo realizacijos yra beveik tiek pat efektyvios ar netgi kai kur atsilieka, nei šiuo metu egzistuojančios pačios efektyviausios programinio atvaizdavimo realizacijos. Tačiau kita karta aparatinių trimatės grafikos realizacijų mobiliesiems telefonams ženkliai pralenks, iki šiol naudotą programinį atvaizdavimo mechanizmą. Daugiausiai aparatinio atvaizdavimo realizacijos darbo suteiks programuotojams, tačiau vartotojai gaus žymiai geresnę kokybę.

Pagrindiniai veiksniai šiandieną ribojantys trimatės grafikos panaudojimą mobiliuosiuose įrenginiuose yra:

- Lėtas procesorius
- Fiksuoto kablelio aritmetika
- Nedidelis atminties pralaidumas ir jos kiekis
- Nėra aparatinio lygio spartinimo
- Nėra aparatinio lygio dalybos operacijos
- Nedidelė ekrano rezoliucija
- Nedidelis spalvų kiekis ekrane
- Ribotas programos dydis
- Ir t.t.

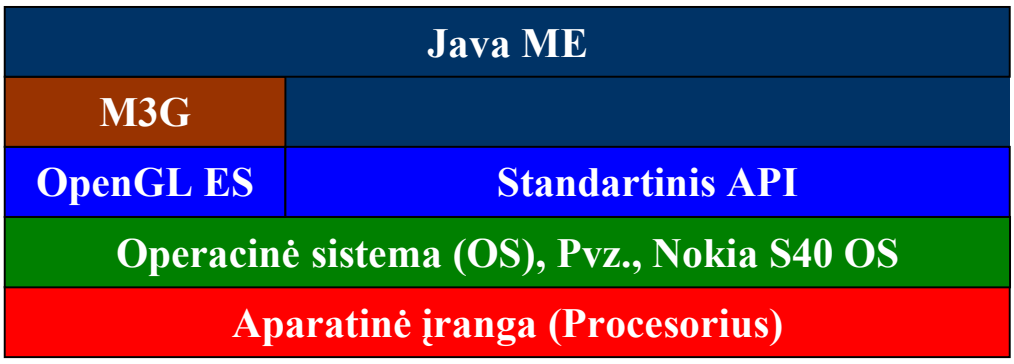
Visa tai galima pavadinti ribotais resursais. Taigi visi mobiliosios 3D grafikos uždaviniai ir susiveda į šių ribotų resursų taupymą.

Pagrindiniai standartai 3D grafikai išreikšti mobiliuosiuose įrenginiuose yra: tam tikros platformos sukurtos pačių įrenginių gamintojų ir diegiamos tik tam tikruose modeliuose; OpenGL ES standartas diegiamas daugumoje modelių, tačiau tiesioginį kūrimą jo pagrindu įgalina tik tam tikros platformos, kaip pavyzdžiui, Symbian OS (S60) Nokia gamintojo platforma; M3G Java standartas, diegiamas beveik visuose naujuose įrenginiuose bei paplitęs kone tarp visų mobiliųjų įrenginių gamintojų. Tyrimams buvo pasirinktas M3G standartas, kaip

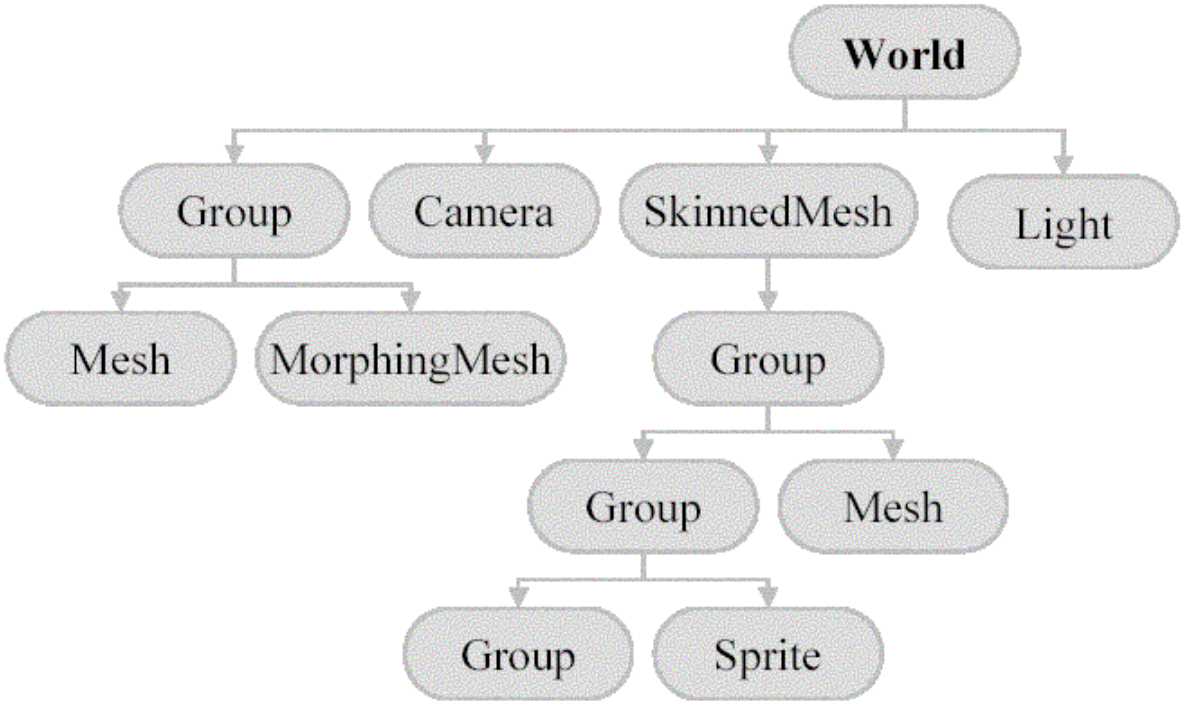
naujausia tokio tipo platforma, kaip labiausiai paplitusi platforma, bei kaip platforma turinti visas reikiamas galimybes NURBS paviršiams realizuoti. Taigi toliau plačiau bus panagrinėtas M3G standartas.

M3G Java – trimatės grafikos standartas mobiliesiems įrenginiams

Naujoji sistema M3G (Mobile 3D graphics API for Java™2 Micro Edition), kuri yra J2ME plėtinys, dar žinoma JSR-184 pavadinimu. Tai yra visiškai naujas 3D grafikos standartas, baigtas kurti tik 2004 metų pradžioje. Mobilieji telefonai, palaikantys šią sistemą pasirodė, tik 2004 metų pabaigoje. Taigi mes ir panorome išbandyti šią sistemą, kadangi tai yra naujas produktas trimatės grafikos kūrimo mobiliesiems telefonams, rinkoje. M3G buvo sukurtas kaip efektyvus API, egzistuojančiai J2ME platformai, taip, kad būtų suderintas su CLDC/MIDP (connected limited device configuration/mobile information device profile). M3G platforma specialiai pritaikyta ribotų resursų įrenginiams: ribota apdorojimo galia, ribotu atminties kiekiu, neturinčiais aparatinio (hardware) trimatės grafikos, bei slankaus kablelio matematikos palaikymo. Mobiliuosiuose telefonuose M3G buvo sukurta, tam, kad galėtų būti taikoma tokiose srityse, kaip žaidimai, žemėlapių vizualizacijos, produktų vizualizacijos, animuoti pranešimai, įvairios vartotojo sąsajos, ekrano užsklandos (screen savers). Šios tikslinės sritys kelia įvairius skirtingus poreikius trimatės grafikos sistemai: kai kurioms reikia tik paprasto turinio sukūrimo, kai kurios reikalauja aukšto realaus laiko našumo, dar kitoms reikia aukštos kokybės nejudančių vaizdų su specialiais efektais.



M3G yra sudaryta iš dviejų dalių: žemesnio ir aukštesnio programinio lygio. Žemesnio lygio dalis, kitaip Immediate (tiesioginė) yra tiesioginis OpenGL ES (embeded systems) apvalkalas (wrapper). Aukštesnio lygio dalis, kitaip Retained, yra žemesnio lygio dalies apvalkalas (wrapper), suteikiantis galimybę formuoti sceną objektiškai.



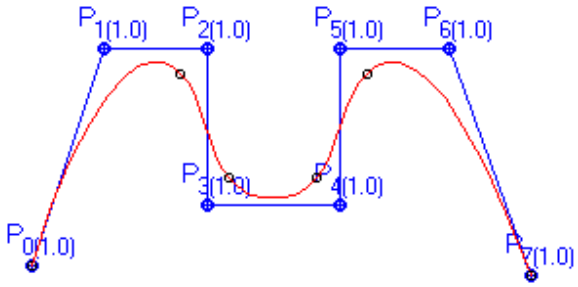
Paveiksliukas 1 - M3G Java Objektinio modelio pavyzdys

Mobile 3D graphics API for Java 2 Microedition, pateikia labai paprastą ir intuityvų programavimo interfeisą, kuris yra pateiktas viename pakete: javax.microedition.m3g Čia yra pateiktos klasės atitinkančios natūralų objektinį suvokimą: Mesh, World, Group, Camera, Lamp, Background, Fog, Image2D, Texture2D, Transform, ... M3G platforma palaiko įvairių 3D objektų kūrimą (Mesh, VertexArray, VertexBuffer, TriangleStripArray, MorphingMesh, SkinnedMesh), transformacijas (Transform), tekstūras (Texture2D, Image2D), apšvietimą (Light), išoriniuose failuose saugomų objektų pakrovimą (Loader), Animacijas (AnimationController, AnimationTrack, KeyframeSequence), scenos foną (Background), 3D Objektų išvaizdą (Material, Appearance), rūką (Fog), Objektų grupavimą (Group, World), ir dar keletas kitų pagalbinių klasių, kaip antai Graphics3D – klasė, kuri atlieka perteikimą (render).

NURBS paviršių teorinė apžvalga

NURBS (Non - Uniform Rational B-Splines) yra matematiniai vaizdavimai trimatės geometrijos, kurie gali tikslai apibūdinti bet kokią figūrą, nuo paprasčiausios linijos, apskritimo ar kreivės, iki sudėtingiausių trimačių paviršių ar figūrų. Dėl jų lankstumo ir tikslumo, NURBS, modeliai gali būti naudojami bet kur, nuo paprastos animacijos iki įvairių pramonės šakų.

NURBS, kreivės ir paviršiai aprašomi panašiais būdais. Kadangi kreivė yra paprastesnė, tai norint labiau susipažinti su NURBS'ais panagrinėsime ją. NURBS kreivę nusako 4 parametrai: Laipsnis, kontroliniai taškai, mazgai, atvaizdavimo taisyklė.



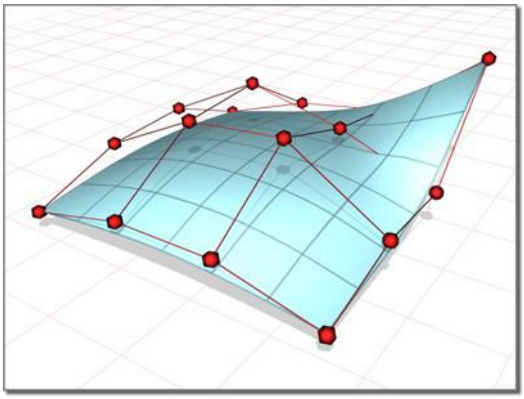
Laipsnis - tai teigiamas sveikas skaičius. Dažniausiai, tai yra skaičiai 1,2,3 ar 5. NURBS linijos paprastai yra pirmo laipsnio, apskritimai antro, įvairios kreivės, trečio arba penkto.

Kontroliniai taškai - yra sąrašas mažiausiai Laipsnis + 1 taškų. Vienas iš lengviausių būdų pakeisti NURBS kreivės formą yra kontrolinių taškų pajudinimas.

Mazgai - Yra sąrašas Laipsnis + N -1 skaičių, kur N yra kontrolinių taškų skaičius. Kartais šis sąrašas yra vadinamas, mazgo vektoriumi.

Atvaizdavimo taisyklė - yra matematinė formulė, kuri paima skaičių ir priskiria tašką.

Viską analogiškai turime ir su paviršiais:



NURBS Kreivės ir Paviršiai

Pirmiausiai apibrėžiama NURBS kreivė, kurios matematinė išraiška gali būti apibrėžta:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u)\omega_i P_i}{\sum_{i=0}^{n} N_{i,p}(u)\omega_i}, a \le u \le b$$

Kur $\{P_i\}$ yra kontroliniai taškai, sudarantys kontrolinį daugiakampį, $\{w_i\}$ yra svoriai, o $\{N_{i,p}(u)\}$ yra p-laipsnio B-splainų pagrindo funkcijos, apibrėžtos šio neperiodinio (neapibrėžtinio) mazgų (knot) vektorius.

$$U = \{a,...,a,u_{p+1},...,u_{m-p-1},b,...,b\}$$

Čia a ir b elementų kiekis yra p+1, ir jeigu nenurodyta kitaip, mes laikome, kad a = 0, b = 1, o $\omega_i > 0$, visiems i. Taigi mes galime pasižymėti:

$$R_{i,p}(u) = \frac{N_{i,p}(u)\omega_i}{\sum_{j=0}^{n} N_{j,p}(u)\omega_j}$$

Atlikus tokį pažymėjimą, mes galime perrašyti bendrinę NURBS kreivės išraišką:

$$C(u) = \sum_{i=0}^{n} R_{i,p}(u)P_i$$

$\{R_{i,p}(u)\}$ yra racionalios pagrindinės funkcijos, jos yra skiautės lygio racionalios funkcijos priklausomos nuo $u \in [0,1]$.

NURBS paviršiai gali būti apibrėžti analogiškai, tik čia dar įvedus po vieną papildomą dimensiją. p laipsnio u kryptimi ir q laipsnio v kryptimi NURBS paviršius yra dvikryptė vektorinė dalinė racionali funkcija, kurios išraiška yra tokia:

$$S(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)N_{j,q}(v)\omega_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)N_{j,q}(v)\omega_{i,j}},0\le u,v\le 1$$

Kur $\{P_i\}$ yra kontroliniai taškai, sudarantys dvikryptį kontrolinį tinklą, $\{w_{i,j}\}$ yra svoriai, o $\{N_{i,p}(u)\}$ bei $\{N_{j,q}(v)\}$ yra neracionalios B-splainų pagrindo funkcijos, apibrėžtos šiais mazgų (knot) vektoriais:

$$U=\{0,...,0,u_{p+1},...,u_{r-p-1},1,...,1\}$$

$$V=\{0,...,0,v_{q+1},...,v_{s-q-1},1,...,1\}$$

Kur pradžioje nulių yra atitinkamai po p+1 bei q+1, o vienetų gale atitinkamai po p+1 ir q+1. O r = n + p + 1 ir s = m + q + 1.

Išvedame racionalias pagrindo funkcijas:

$$R_{i,j}(u,v)=\frac{N_{i,p}(u)N_{j,q}(v)\omega_{i,j}}{\sum_{k=0}^{n}\sum_{l=0}^{m}N_{k,p}(u)N_{l,q}(v)\omega_{k,l}}$$

Taigi dabar galima perrašyti ir paviršiaus lygtį:

$$S(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}R_{i,j}(u,v)P_{i,j}$$

Pagrindinės NURBS aprašymo struktūros

NURBS kreivė arba paviršius susideda iš daugelio elementų, kur kiekvienas iš jų gali būti aprašytas, kaip Bezier kreivė ar paviršius. Tokia apibrėžtis leidžia lokaliai modifikuoti NURBS objektą, kadangi modifikuojant vieną kontrolinį tašką kita tik nedidelė splaino dalis.

Mazgų (knots) sąrašas, nusako parametrinės erdvės sudalijimo intervalus. Toliau mazgų paaiškinimą pateiksime NURBS kreivės pavyzdžiu. Kaip jau minėta mazgų seka nusako

konkrečius parametrinės erdvės intervalus, iš kur kreivės segmentai yra atvaizduojami objekto erdvėje. B-splainas yra vadinamas pastoviu (uniform), jei visi šie intervalai yra lygūs ir nepastoviu (non-uniform) priešingu atveju. Mazgų skaičius turi būti lygus kontrolinių taškų skaičiui plius eilei:

Mazgų kiekis = kontrolinių taškų skaičius + eilė

Pavyzdžiui, 4 eilės kreivė, sudaryta iš 6 kontrolinių taškų, gali turėti štai tokį pastovų mazgų vektorių:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

O štai kreivė su tokiu mazgų vektorium yra nepastovi:

[0, 0, 0, 0, 1.5, 2, 2, 4, 4, 4, 4]

Jei norima atvaizduoti tiesiog Bezier k-osios eilės kreivę, reikia naudoti tokią mazgų seką:

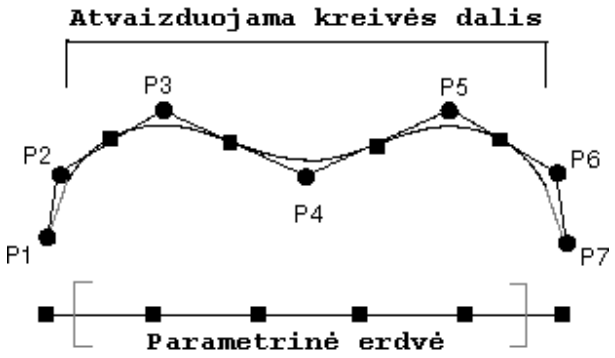
[0, 0,...k-kartų, 1,1,…,k-kartų]

Nepastovi mazgų seka ne tik suteikia galimybę įtakoti kreivės formą, bet ir turint keletą mazgų tame pačiame taške, leidžia įtakoti tolydumą tarp atskirų Bezier segmentų. Jei mes turime 2 mazgus viename taške, tai mes turime pirmos eilės tolydumą. Jei mes turime tris mazgus, tai mes turime pozicinį glodumą (segmentai susiliečia). O jeigu iš eilės eina keturi mazgai, tai kreivė praranda tolydumą tame taške (gauname trūkį). Taigi, išreiškime tolydumo laipsnio priklausomybę:

(tolydumo laipsnis mazge) = (eilė – 1 – mazgo kartotinumas)

Parametrų rėžiai [umin, umax], nurodo kuri tiksliai kreivės dalis bus atvaizduojama. Su NURBS kreivėmis bei paviršiais dažniausiai bus naudojama parametrų erdvės [0..1] ir [0..1]x[0..1].

Paveikslėlyje matosi sąryšis tarp parametrinės erdvės, bei tarp objekto erdvės, taip pat matosi, kaip parametrinėje bei objektinėse erdvėse išsidėsto mazgai (knots). Kontroliniai taškai čia pavaizduoti taškais ir atitinkamai pažymėti PN, o mazgai yra pavaizduoti kvadračiukais:

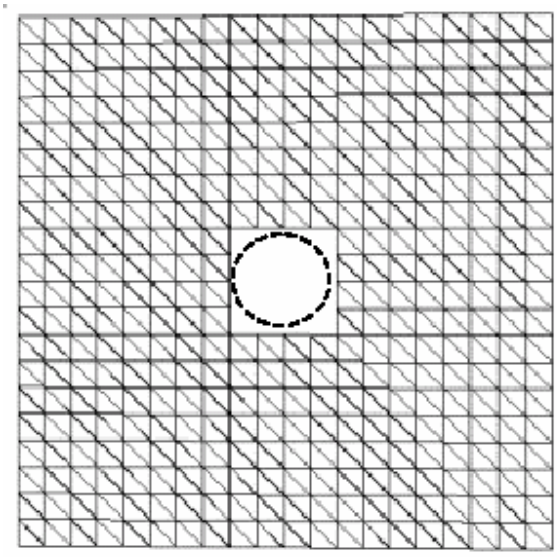


Taigi kad pilnai aprašyti NURBS kreivę reikia kontrolinių taškų vektoriaus, laipsnio ir mazgų (knots) vektoriaus. To pilnai užtenka, kad atvaizduoti NURBS kreivę.

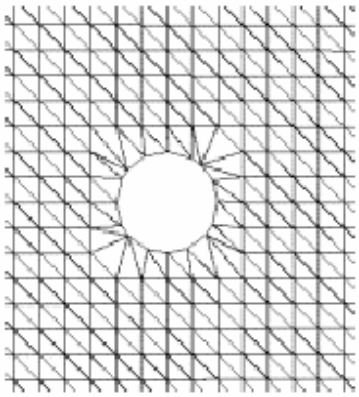
Analogiškai NURBS paviršius aprašomas kontrolinių taškų aibe (kontroliniai taškai apjungti į dvimatę matricą), laipsniu abejomis kryptimis, bei dviem mazgų vektoriais, atitinkamai u ir v kryptimis.

Trimmed (iškarpytieji) NURBS

Paviršių iškarpyti galima dviem būdais: kuriant paviršių skyles įtraukti iš anksto arba iš pradžių pagaminti paviršių ir jau poto jame iškirpti skyles. Pirmuoju atveju, yra išskaičiuojami skylės ir paviršiaus susikirtimo taškai ir tuomet iš visos taškų aibės iškart sugeneruojamas iškarpytas paviršius. Tačiau suskaičiuoti sankirtos taškus tarp norimos iškirpti skylės ir esamo paviršiaus nėra toks paprastas dalykas. Antrąjį būdą realizuoti yra žymiai paprasčiau ir jis yra efektyvesnis. Iškarpymo idėja yra tokia: paviršiaus taškų skaičiavimo metu (tesselation), reikia įsiminti, kurie taškai yra ant paviršiaus ir kurie ne. Norint tai padaryti reikia kiekvienam taškui atlikti patikrinimą: ar jis priklauso paviršiui ar skylei. Atlikus šiuos veiksmus gaunamas toks paviršius:



Rezultatas nėra visiškai pakankamas, tačiau jau yra gautas iškarpytas (trimmed) NURBS'as. Sekančiame žingsnyje reikia patobulinti šį paviršių taip, kad gautume apvalią skylę. Idėja yra paprasta: reikia perkelti taškus, kurie yra ant kontūro krašto ir pakeisti jų koordinates taip, kad gautume skylės formą. Pirma klausimas būtų, kaip atskirti tuos kraštinius taškus. Galime pastebėti tokį dalyką, kad taškas yra ant krašto, jei jis turi mažiau nei aštuonis kaimynus. Dabar, kai jau yra žinoma kokie taškai yra kraštiniai, reikia surasti kuris taškas tam tikroje vietoje yra artimiausias iki skylės kontūro ir jį perkelti ant to kontūro. Atlikus šiuos veiksmus yra gaunamas iškarpytas paviršius toks, koks ir buvo norėta:



Pagrindiniai NURBS realizacijų aspektai

Pagrindinis NURBS realizacijos uždavinys yra kaip iš turimos matematinės išraiškos yra atvaizduojamas paviršius arba kreivė. Bendrai 3D grafikoje pagrindinis vaizdavimo primityvas yra daugiakampis, dažniausiai trikampis. Tas pats yra ir mobiliosiose platformose, kur netgi mūsų nagrinėjamoje M3G Javoje, tai yra tik vienintelis būdas atvaizduoti paviršių (Trikampių seka). Taigi šios problemos sprendime mes turime gauti algoritmą (metodiką), kaip iš turimos NURBS išraiškos sugeneruoti trikampių seką, kurią jau galima atvaizduoti standartinėmis tos platformos priemonėmis. Angliškai šis procesas vadinamas Tessellation. Toliau šiuos algoritmus vadinsime NURBS atvaizdavimo algoritmais. Pirmiausia šioje srityje egzistuoja tiek keletas skirtingų išraiškų naudojamų NURBS paviršiams bei kreivėms aprašyti, bei kiekvienai išraiškai po keletą skirtingų atvaizdavimo algoritmų.

Pagrindinės NURBS išraiškos

Pagrindinės aprašomos NURBS išraiškos yra matematinė, kada NURBS‘as išreiškiamas kaip B-Splainas. Atroji išraiška yra Bezier išraiška. Tai kada NURBS paviršius yra sudalintas į daug Bezier skaičių, kur kvadratinėje matricoje sudėti šie paviršiai geometriškai sudaro tokį patį paviršių, kaip ir matematinės išraiškos paviršius. Trečioji yra polinominė (power-basis) išraiška. Čia Bezier skiautės yra konvertuojamos į polinomus. Aišku, kad tokia konversija yra skaitiškai nestabili esant dideliems polinomo laipsniams. Kaip matosi kad pirmoji išraiška yra natūrali (artimiausia matematinei išraiškai), likusios dvi išraiškos yra išvestinės, dėl to nagrinėjant tokias išraiškas egzistuoja konvertavimo tarp išraiškų problema. Išvestinės išraiškos reikalauja daugiau elementų aprašyti tam pačiam paviršiui, nei natūralioji išraiška.

Pagrindiniai NURBS atvaizdavimo algoritmai

Kiekvienai iš šių išraiškų egzistuoja atskiri atvaizdavimo algoritmai. Natūraliosios išraiškos atvaizdavimas yra atliekamas taškus tiesiogiai išskaičiuojant iš matematinės išraiškos. Bezier išraiškos atveju atvaizdavimas atliekamas naudojant Bezier paviršių atvaizdavimo algoritmus.

Taigi pagrindiniai literatūroje sutinkami NURBS paviršių natūraliojoje išraiškoje atvaizdavimo algoritmai yra optimizuotas tiesioginio skaičiavimo, dviejų fazių Cox de Boor, mazgų įterpimo bei Oslo. Cox de Boor algoritmas padeda išvengti pakartotinių skaičiavimų tiems patiems komponentams arba tarpinių skaičiavimų (atvirkštiniams u ir v reikšmių skirtumams), iš anksto apskaičiuojant ir išsaugant reikšmes abejomis kryptimis, o tada generuojant tenzorius iš parametrizuotų reikšmių porų. Optimizuotas tiesioginio skaičiavimo algoritmas naudoja sąryšius tarp viršūnių, kad pagreitinti skaičiavimus tarp viršūnių esančių toje pačioje parametrinėje plokštumoje.

Bezier išraiškoje esantiems paviršiams atvaizduoti naudojami deCasteljau ir modifikuotas deCasteljau algoritmai. Algoritmo modifikacija naudoja išankstinį svorių, įtakojančių tarpinius rekursijos taškus, apskaičiavimą.

Paviršiams išsaugotiems polinominėje išraiškoje yra naudojamas Hornerio schemos algoritmas. Šiame metode koeficientai šalia polinomo laipsnių yra iš anksto apskaičiuoti ir išsaugoti.

Pagrindiniai aspektai NURBS realizacijai mobiliuosiuose telefonuose į kuriuos buvo orientuotasi, tai paprastumas, bei efektyvumas. Vizuali NURBS objektų kokybė liko antrame plane, nes mobiliųjų įrenginių platformoje pagrindiniai veiksniai, kuriuos reikia įveikti, tai riboti resursai. Realizacijai buvo pasirinkti du NURBS atvaizdavimo algoritmai – Oslo bei tiesioginis.

Oslo algoritmas

Viena iš priežasčių dėl ko buvo pasirinktas Oslo algoritmas, buvo jo paprastumas, bei panašumas į efektyvius deCasteljau algoritmus skirtus piešti Bezier kreivėms ir paviršiams. Taip pat Oslo algoritmas yra sutinkamas ir praktinėse realizacijose, pavyzdžiui, VRML NURBS plėtiniuose. DeCasteljau algoritmas veikia padalijimų principu, kada dalijimas pradedamas nuo kontrolinio poligono ir su lig kiekviena iteracija vyksta glodinimas ir artėjimas prie realiosios išraiškos. Oslo algoritmas veikia panašiai. Dar viena iš priežasčių lėmusių pasirinki šį algoritmą, buvo informacija. Šiek tiek keista, nors ir NURBS technologija yra jau ganėtinai sena ir plačiai taikoma, laisvai surasti informacijos nėra taip lengva. Dauguma informacijos yra arba paviršutiniška arba mokama.

Taigi Oslo algoritmo esmė yra padalijimai arba mazgų įterpimai. Mazgų įterpimas – tai operacija, kada yra įterpiami papildomi mazgai, nekeičiant pačios figūros formos. Taip pat norint išlaikyti tokią pačią figūros eilę reikia įterpti ir naujų kontrolinių taškų. O šis mechanizmas pasitarnauja, atvaizduojant NURBS paviršius, nes tikslinant kontrolinių taškų tinklą įterpinėjant mazgus, šis konverguoja prie aproksimuotos figūros. Pagrindiniai šio algoritmo parametrai yra tik pradiniai NURBS aprašantys duomenys, bei įterptinų mazgų skaičius abejomis kryptimis – u ir v. Visas algoritmo veikimas yra ganėtinai paprastas – iš pradžių mazgai yra įterpiami ir pertvarkomi kontroliniai taškai viena kryptimi – u, o po to analogiškai kita kryptimi – v. Čia įterpimas vykdomas ne po vieną mazgą, o visų reikiamų mazgų kiekį iš karto. Tai vadinama mazgų „tobulinimu“ (knot refinement). Kada yra sugeneruojama nauja aibė mazgų ir ji yra sujungiama (union) su senaisiais mazgais.

Tiesioginis algoritmas

Tiesioginis algoritmas buvo pasirinktas, dėl jo matematinio pagrindo, bei efektyvumo. Čia jis išskiriamas kaip efektyviausias, savo grupėje. Pagrindinė šio algoritmo idėja yra, kad naudojant matematinę NURBS paviršiaus išraišką yra išskaičiuojama tam tikra paviršiaus taškų aibė ir vėliau jie sutrikampiuojami. Trikampiai apjungiami į trikampių juostas (Triangle Strips) ir perduodami atvaizduoti standartinei 3D grafikos posistemei – M3G.

Šiame algoritme yra panaudotas reguliarus perėjimas. T.y. pagal nustatytą tikslumą parametrinė erdvė yra pereinama lygiais intervalais u ir v kryptimis ir kiekviename taške yra išskaičiuojamas atitinkantis paviršiaus taškas.

Kad paspartinti $S(u,v)$ išraiškos skaičiavimą, yra galima iš anksto apskaičiuoti koeficientus $N_{i,p}(u)$ ir $N_{j,q}(v)$ sudalijimo taškuose ir saugoti juos masyve. Šis apskaičiavimas yra atliekamas vieną kartą, tai kad to nereiktų daryti pagrindiniame cikle skaičiuojant paviršiaus taškus. Taigi vienintelis reikalingas veiksmas yra reikšmės paėmimas iš masyvo, bei sudauginimas. Jei pasikeičia kontroliniai taškai arba pasikeičia paviršiaus atvaizdavimo tikslumas, tada reikia perskaičiuoti, išsaugotas bazines funkcijas, naujuose taškuose.

Taigi čia pagrindinė idėja, kaip „išskaičiuojamas“ NURBS paviršius, tačiau sėkmingai jį atvaizduoti 3D grafikos posistemėje dar reikia apskaičiuoti paviršiaus normales, kurios yra

reikalingos matomumo bei apšvietimo skaičiavimams. Pagrindinė taisyklė čia – funkcijos išvestinė taške yra lygi liestinės kampo tangentui. Paėmus dvi liestines (vieną u ir vieną v kryptimi) taške ir apskaičiavus jų sudaromos plokštumos statmenį gausime šio taško normalę. Funkcija $S(u,v)$ turi dvi dalines išvestines (vieną pagal u ir vieną pagal v):

$$\frac{\partial \vec{S}(u,v)}{\partial u} = \frac{\left(\sum_{i=0}^{n}\sum_{j=0}^{m}\frac{dB_{i,p}(u)}{du}B_{j,q}(v)\vec{P}_{i,j}w_{i,j}\right)\left(\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,p}(u)B_{j,q}(v)w_{i,j}\right) - \left(\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,p}(u)B_{j,q}(v)\vec{P}_{i,j}w_{i,j}\right)\left(\sum_{i=0}^{n}\sum_{j=0}^{m}\frac{dB_{i,p}(u)}{du}B_{j,q}(v)w_{i,j}\right)}{\left(\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,p}(u)B_{j,q}(v)w_{i,j}\right)^2}$$

$$\frac{\partial \vec{S}(u,v)}{\partial v} = \frac{\left(\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,p}(u)\frac{dB_{j,q}(v)}{dv}\vec{P}_{i,j}w_{i,j}\right)\left(\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,p}(u)B_{j,q}(v)w_{i,j}\right) - \left(\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,p}(u)B_{j,q}(v)\vec{P}_{i,j}w_{i,j}\right)\left(\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,p}(u)\frac{dB_{j,q}(v)}{dv}w_{i,j}\right)}{\left(\sum_{i=0}^{n}\sum_{j=0}^{m}B_{i,p}(u)B_{j,q}(v)w_{i,j}\right)^2}$$

Tik nėra labai trivialu suskaičiuoti koeficientų $B_{i,p}(u)$ ir $B_{j,q}(v)$ išvestines. Galima tai būtų padaryti iš jų išraiškų, tačiau paprasčiau tai išskaičiuoti iš koeficientą atitinkančio polinomo. Suskaičiuoti polinomo išvestinę tereikia laipsnius sudauginti iš koeficientų, bei laipsnius sumažinti vienetu.

$$\frac{dB_{i,p}(u)}{du} = pC_{i,p,p}(u)u^{p-1} + (p-1)C_{i,p,p-1}(u)u^{p-2} + \ldots + C_{i,p,1}(u)$$

Išvestinių skaičiavimai šiame algoritme yra vienas iš sudėtingiausių veiksmų, tačiau jis bus atliekamas tik vieną kartą, kol nereikės keisti paviršiaus tikslumo.

Algoritmų realizacijos

Pagrindiniai realizacijos tikslai yra realizuoti bei adaptuoti mobiliesiems telefonams pasirinktus NURBS algoritmus, bei pamatuoti ir palyginti jų veikimą. Realizacinė platforma – J2ME (su M3G praplėtimais). Taigi rezultate turi būti gautas J2ME platformos diegiamasis paketas (JAR+JAD), kurį uždiegus bei paleidus mobiliajame įrenginyje arba mobiliojo įrenginio emuliatoriuje, bus atvaizduojami NURBS modeliai bei pateikiama jų atvaizdavimo algoritmo veikimo matavimai. Toliau bus šnekama apie konkretesnes realizacijos detales, bei realizuotas Java klases.

Kadangi realizuojami buvo du NURBS paviršių algoritmai, kurie turėjo būti palyginami vienodomis aplinkybėmis, pirmiausia buvo sukonstruota bendrinė architektūra. Joje numatytas interfeisas *Nurbs*, apibendrinantis visus NURBS paviršius. Bendrinė architektūra taip pat buvo numatytas *Creatable* interfeisas, kuris yra NURBS „gamintojas“. Tai yra, *Creatble* interfeisą realizuojančioje klasėje bus aprašyti, visi konkretaus NURBS modelio parametrai (kontroliniai taškai, mazgai, eilė ir t.t.), ten bus sukonstruojama *Nurbs* interfeisą realizuojanti klasė, kuri atitiks vieną konkretų NUBS paviršiaus modelį. Toliau yra pateikiama klasių diagrama atvaizduojanti bendrosios architektūros loginį modelį.

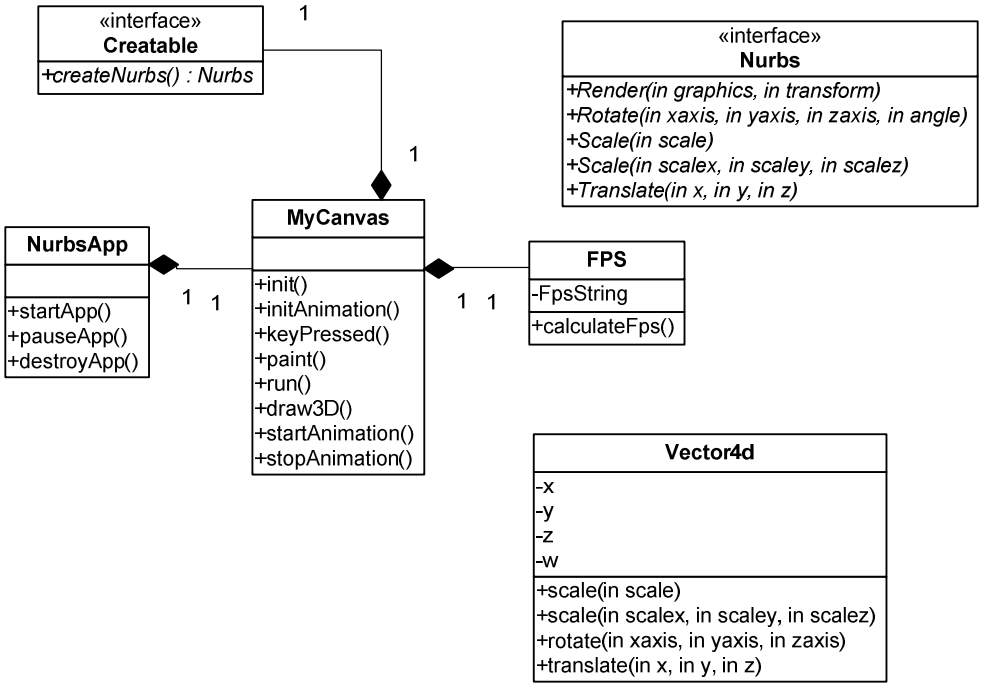


Diagrama 1 - Bendrosios architektūros klasių diagrama

Diagramoje be interfeisų *Nurbs* ir *Creatable* yra tokios klasės: **NurbsApp**, **MyCanvas**, **FPS** bei pagalbinė **Vector4d** klasė:

- **NurbsApp** – yra pagrindinė šios programos klasė, kadangi ji paveldi iš javax.microedition.midlet.MIDlet klasės, kuri yra paleidžiamoji J2ME klasė. Ši klasė realizuoja standartinius mobiliosios aplikacijos paleidimo, laikino sustabdymo bei išjungimo metodus. Ji sukuria **MyCanvas** klasės objektą su norimu Nurbs Modelio „gamintoju“.
- **MyCanvas –** yra pagrindinė vaizduojamoji klasė, nes kaip tik ją NurbsApp klasė priskiria, kaip dabar vaizduotiną. Ši klasė paveldi iš javax.microedition.lcdui.game.GameCanvas klasės bei papildomai realizuoja *Runnable* interfeisą (kas įgalins visus vaizdavimo skaičiavimus atlikti atskiroje gijoje (Thread), taip neapkraunant bendro programos veikimo). Pirmiausia ji pasiruošia visą aplinką (apšvietimus, kameras ir t.t.), „gamintojo“ pagalba susigeneruoja Nurbs objektus. Toliau ji tik aptarnauja vartotojo įvykius sužadintus įrenginio klavišų pagalba. Klavišais objektą galima sukioti, priartinti, bei paleisti animaciją. Animacija pasileidžia atskiroje gijoje ir sukasi tol kol yra nesustabdoma. Animacijos veikimo principas yra paprastas, paleidžiamas amžinas ciklas, kuriame yra atliekama transformacija ir perpiešimas. Taigi animacija nėra kaip nors ribojama, o sukasi maksimaliu greičiu, kuriuo gali veikti pats įrenginys. Tokiu būdu mes galime pamatuoti ir palyginti algoritmų efektyvumą.
- Klasė **FPS** – tai pagalbinė klasė, skirta matuoti kadrams per sekundę (angl. Frames per second (fps)). Matavimas kadrais per sekundę, tai trimatėje kompiuterinėje grafikoje yra paplitusi metodika. Kiekvieno perpiešimo metu yra kreipiamasi į šios klasės calculateFps() metodą, kuris patikrina ar šis piešimas vyksta dar einamąją sekundę ar jau naują.
- Pagalbinė **Vector4d** klasė – atitinka erdvinį tašką su svoriu, kurie yra naudojami išreikšti NURBS kontroliniams taškams.

Taigi toliau reikia realizuoti NURBS paviršių algoritmus, kaip klases implementuojančias *Nubs* interfeisą. O modelius perkelti, kaip klases implementuojančias *Creatable* interfeisą. Pagrindinės taisyklės realizuojant tokius algoritmus M3G grafikos pagrindu yra:

- Efektyvumas – realizacija turi būti kiek įmanoma efektyvesnė

- Tarpinių duomenų saugojimas. Jei įmanoma reikia saugoti tarpinius duomenis tarp skaičiavimų, kad jų nereiktų perskaičiuodinėti, taip būtų sutaupyta procesoriaus laiko kas labai daug lemia tokio tipo uždaviniuose
- Viena iš žinomų M3G Javos problemų yra, ta jog viršūnės bei normalės gali būti išreikštos tik sveikais skaičiais. Visur kitur galima naudoti realiuosius apart viršūnių bei normalių. Todėl šiai problemai buvo sugalvotas toks sprendimas – surinkus aibę viršūnių reikia išrinkti didžiausią absoliučią reikšmę bet kuriomis kryptimis (x, y, z), tada gautą atstumą reikia sutransformuoti į sveikąją erdvę. Viršūnėms bei išreikšti mes turime tipą byte arba short, kurių maksimalios reikšmės gali būti – 127 arba 32767 atitinkamai. Taigi išskaičiuojame reikiamą santykio koeficientą padalinę šias maksimalias reikšmes. O vėliau šiuos taškus atvaizduodami panaudosime atitinkamą transformaciją, su atvirkštiniu koeficientu, nes transformacijoje jau galima nurodyti realius skaičius. Taigi gaunasi taip, kad mes objektus padidiname iki maksimumo (kad, kuo mažiau prarasti tikslumo), o atvaizduodami atitinkamai sumažiname iki realaus objektų dydžio. Vizualiai tai niekaip neatsispindi. Tokiu būdu mes galime „apgauti“ M3G grafikos posistemę.

Oslo algoritmo realizacija

Realizuojant Oslo algoritmą buvo naudotas objektinis programavimo modelis, kur pagrindinis objektas (klasė) yra NurbsSurface. NurbsSurface atitinka vieną NURBS paviršių. Kaip pagalbinės klasės čia yra naudojamos ControlNet, ControlPoint ir KnotVector klasės. ControlPoint atspindi vieną 4D tašką, kuris yra naudojamas išreikšti kontroliniams taškams. Čia ketvirtoji komponentė yra w – svoris. Taip kiekvienam taškui galima nurodyti jo svorį. Dažniausiai praktikoje visi svoriai yra nurodomi kaip 1. Tačiau atliekant įvairius modifikavimo algoritmus svoris gali kisti, taip pat šioje Oslo algoritmo realizacijoje taškai yra skaičiuojami su svoriais. Kur atvaizduojant tašką, visos jo komponentės yra normalizuojamos svorio dydžiu:

$$x/w;\ y/w;\ z/w$$

Kita kasė yra KnotVector, kas atitinka mazgų vektorių. Čia pagrindiniai laukai yra: mazgų sąrašas, mazgų kiekis, eilė, bei tos krypties kontrolinių taškų skaičius. Pagalbiniai metodai, kaip reikšmių priskyrimui bei nuskaitymui, Objekto kopijavimui, mazgų reikšmių apytiksliam

palyginimui. Taip pat metodai skirti naujų mazgų aibės sugeneravimui, bei mazgų apjungimui (union).

Toliau – ControlNet klasė. Ji aprašo kontrolinių taškų aibę, bei generuoja savo atvaizdą trikampiais. Taigi šios klasės laukai ir yra kontrolinių taškų kiekis u kryptimi, kontrolinių taškų kiekis v kryptimi, bei pats kontrolinių taškų rinkinys išreikštas dvimačiu ControlPoint objektų masyvu. Veiksmai pirmiausiai yra susiję su reikšmių nuskaitymu bei naujų priskyrimu, taip pat metodas sukurti objekto kopijai. Pagrindinis metodas yra generuojantis trikampių seką, skirtą šiam tinklui atvaizduoti. Šis metodas turimus kontrolinius taškus pirmiausia normalizuoja pagal svorius, tada juos konvertuoja į sveikus skaičius, pagal 5 skyriuje pateiktą metodiką ir visus šiuos taškus trikampiuoja. Šioje fazėje yra sugeneruojamos trikampių juostos, kurios yra apjungiamos į Mesh objektą ir jis gražinamas atvaizdavimui standartinėmis šios platformos priemonėmis.

Paskutinė ir pagrindinė klasė – NurbsSurface, kuri ir atitinka NURBS paviršių. Jos laukai – kontrolinis tinklas išreikštas prieš tai aprašytu objektu, bei du mazgų vektoriai u ir v kryptimis. O metodai vėlgi pirmiausia – laukų priskyrimo, nuskaitymo; objekto kūrimo, bei kopijavimo. Tada turime pagalbinį metodą – transpose, kuris sukeičia visus u taškus su v taškais, tam kad būtų lengviau atlikti padalijimo veiksmus abejomis kryptimis. T.y. pirmiausiai padalijimas atliekamas u kryptimi, tada paviršius transponuojamas ir padalijimas vėl

atliekamas u kryptimi ir tada paviršius transponuojamas atgal. Tokiu būdu tereikia dalijimo algoritmą realizuoti tik vienai krypčiai. Taigi pagrindinis algoritmas čia yra sudalijimo tam tikru pateiktu tikslumu, kas vėliau naudoja sudalijimą viena kryptimi.

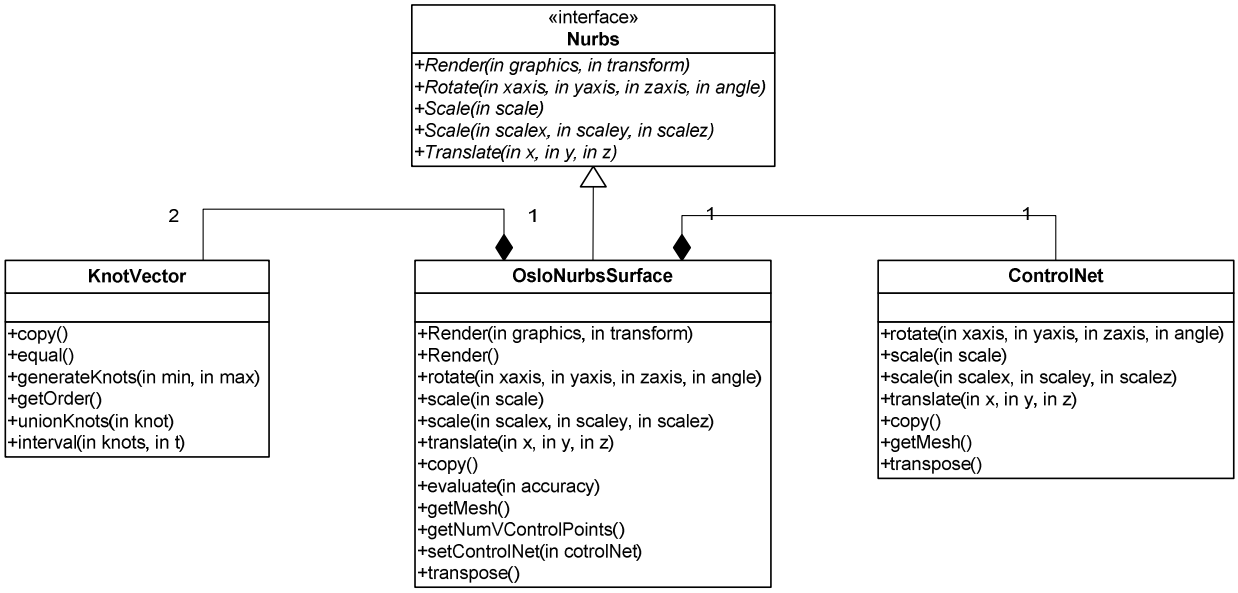


Diagrama 2 - Oslo algoritmo loginė klasių schema

Taigi pagrindinė NURBS atvaizdavimo schema atrodytų taip: Sukuriamas kontrolinis tinkas, bei du mazgų vektoriai, iš šių komponentų sukuriamas Nurbs paviršiaus objektas, kuriam paleidžiamas dalijimo algoritmas su tam tikru tikslumu. Sudalintas paviršius yra išreiškiamas trikampių sekomis, kurios yra atvaizduojamos naudojantis standartinėmis M3G Javos priemonėmis.

Tiesioginio algoritmo realizacija

Šis algoritmas vadinasi tiesioginis, nes jis tiesiogiai atitinka matematinę NURBS paviršių išraišką. Čia figūruoja bazinės funkcijos, bei jų išvestinės. O atvaizdavimo idėja yra tokia, kad mes susigeneruojame tam tikro tikslumo tinklelį parametrinėje erdvėje ir kiekviename šio tinklelio taške apskaičiuojame paviršiaus tašką bei normalę pasinaudodami matematine NURBS

paviršiaus išraiška. Tada šiuos taškus apjungiame į trikampių sekas. Šis algoritmas vadinamas reguliariuoju, ne mes parametrinėje erdvėje sugeneruojame reguliarųjį tinklelį. (tai yra, visos atkarpos padalijamos į lygias dalis, visomis kryptimis). Pirmiausia buvo sudaryta algoritmo loginė schema, kuri pateikiama toliau:

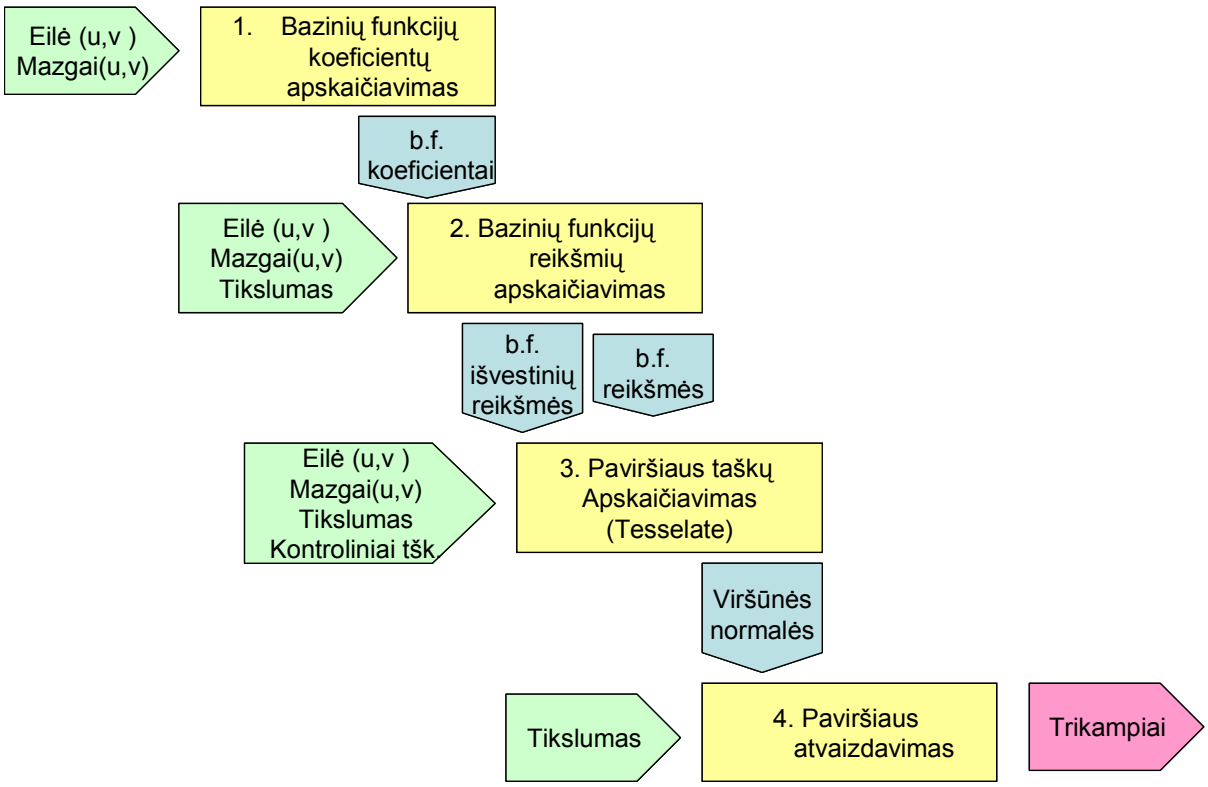


Diagrama 3 - Tiesioginio algoritmo veikimo diagrama

Algoritme išskirtos 4-ios fazės: 1-oji tai bazinių funkcijų koeficientų apskaičiavimas, 2-oji tai Bazinių funkcijų reikšmių bei išvestinių reikšmių apskaičiavimas, 3-oji – Paviršiaus taškų apskaičiavimas (angl. Tesselate), 4-oji Paviršiaus atvaizdavimas (angl. Render). Kiekvienoje fazėje yra pažymėti pradiniai duomenys, bei rezultate gauti duomenys. Taip pat išskirta, kurie duomenys yra perduodami tarp fazių bei kurie yra pradiniai viso algoritmo duomenys. Žiūrint bendruoju atvejų į algoritmą, tai apskritai pradiniai duomenys yra: kontroliniai taškai, mazgai u ir v kryptimis, paviršiaus eilė u ir v kryptimis, bei atvaizdavimo tikslumas. Rezultate turime gauti trikampių sekų aibę, kurią galės atvaizduoti standartinės trimatės grafikos posistemės, mūsų atveju –M3G Java.

Panagrinėjus šias fazes, bei jų priklausomybę nuo duomenų galima išskirti scenarijus, kuriais būtų skaičiuojamas šis algoritmas, priklausomai nuo pakitusių duomenų. Pas mus kisti gali tik paviršiaus atvaizdavimo tikslumas, bei patys kontroliniai taškai. Paviršiaus atvaizdavimo tikslumas gali kisti, kai bus norima objektą paglodinti (pasmulkinti) arba atvirkščiai pastambinti. Paviršiaus kontroliniai taškai gali kisti atliekant su jais arba su jų svoriais transformacijas, pavyzdžiui animuojant. Taigi gauname du scenarijus, kada keičiamas tikslumas, bei kada keičiami patys kontroliniai taškai. Iš algoritmo diagramos matome, kad anksčiausiai tikslumas yra panaudotas antroje fazėje, o kontroliniai taškai pirmą kartą panaudojami trečioje fazėje. Todėl jei išsisaugotumėme pirmosios fazės rezultatą – bazinių funkcijų koeficientus, keičiant modelio tikslumą nebereiktų perskaičiuoti pirmosios fazės. O jeigu išsisaugotumėme ir antrosios fazės rezultatą – bazinių funkcijų reikšmes ir bazinių funkcijų išvestinių reikšmes, modifikuojant kontrolinius taškus nebereiktų perskaičiuodinėti ir antrosios fazės. Taip pat jei atliekama animacija tiesiogiai su apskaičiuotais trikampiais ar modelis tiesiog peržiūrimas, nebūtina vykdyti ir likusių fazių. Tokiu atveju galima tiesiog saugoti galutinį variantą – trikampių seką.

Laikantis visų šių išvadų buvo realizuotas tiesioginis NURBS atvaizdavimo algoritmas kaip **DirectNurbsSurface** klasė. Taip pat realizuota pagalbinė klasės **Vector3d**, išreiškianti trimačius vektorius. Toliau pateikiama loginė klasių diagrama realizuojanti tiesioginį algoritmą:

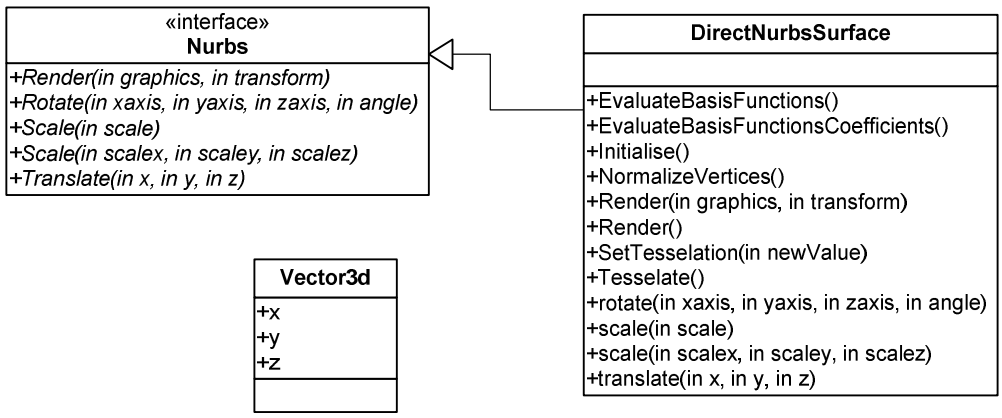


Diagrama 4 - Tiesioginio algoritmo loginė klasių diagrama

Taigi algoritmų palyginimui gavome štai tokią programos schemą:

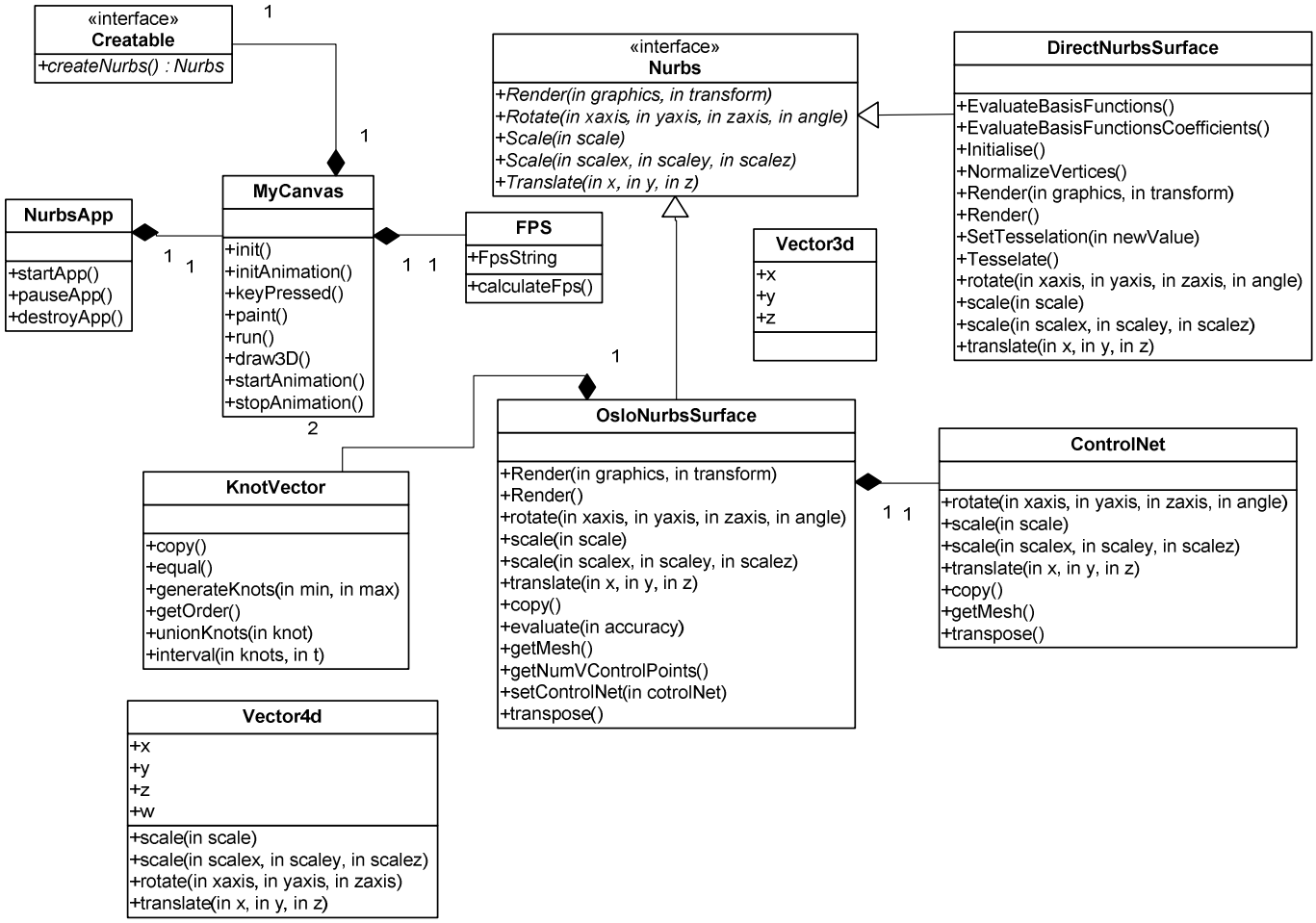


Diagrama 5 - Bendroji algoritmų palyginimo loginė klasių diagrama

Algoritmų palyginimas

Kad įsitikinti, kuris iš šių algoritmų yra efektyvesnis, buvo atliktas jų veikimo palyginimas. Palyginimui buvo panaudoti įvairūs NURBS modeliai: ančiukas (308 kontroliniai taškai), kvepalų buteliukas (281 kontrolinis taškas), plokštuma (16 viršūnių).

Lyginama buvo animuojant NURBS paviršius, bei skaičiuojant kiek kartų per sekundę objektai yra perpiešiami. Tai yra kompiuterinėje grafikoje naudojamas terminas – fps (Frames per second). Animacija buvo atliekama begaliniam cikle, tiesiog perskaičiuojant ir perpiešiant, be jokių papildomų užstabdimų. Buvo bandomos dviejų tipų animacijos: animuojant NURBS paviršius per kontrolinius taškus bei animuojant jau apskaičiuotus NURBS paviršius standartinėmis 3D posistemės galimybėmis.

Testavimas buvo atliekamas ant dviejų platformų:

- Sun Java Wireless toolkit 2.2 emuliatorius (Default Color Phone, Ekranas 240x320, 4096 spalvų) ant Personalinio kompiuterio (1.7GHz PentiumM procesorius, 512MB operatyvinės atminties, ATI Mobility Radeon X600 su 64MB grafinės operatyvinės atminties)
- Mobilusis telefonas Nokia 6230i. Nokia S40 Operacinė sistema. CLDC 1.1 bei MIDP 2.0 platforma. Ekranas: 208x208, 16 bit spalvų gylis. Dinaminės atminties dydis: 2MB. Didžiausias galimas programos paketo (jar) dydis – 500 KB.
- Mobilusis telefonas Nokia 6680. Nokia S60 Operacinė sistema (Symbian OS pagrindu). CLDC 1.1 bei MIDP 2.0 platforma. Ekranas: 176x208, 18 bit. Spalvų gylis. Dinaminės atminties dydis – neribotas. Programos paketo (jar) dydis – neribojamas

Modelis	Tikslumas	Trikampių sk.	Oslo algoritmas		Tiesioginis algoritmas	
			KT animacija	Mesh animacija	KT animacija	Mesh animacija
ančiukas	10	~670	2-3 fps	54-55 fps	5-6 fps	54-55 fps
	15	~1400	1-2 fps	52-53 fps	3-4 fps	52-53 fps

	20	~2500	1 fps	48-49 fps	2-3 fps	48-49 fps
Kvepalų buteliukas	10 (7)	~472	4-5 fps	58-59 fps	7-8 fps	58-59 fps
	15 (12)	~ 918	3-4 fps	58-59 fps	5 fps	58-59 fps
	20 (17)	~1604	2-3 fps	52-53 fps	3-4 fps	52-53 fps
plokštuma	10(15)	200	8-9 fps	56-58 fps	21-22 fps	56-58 fps
	15(20)	450	4-5 fps	55-57 fps	15-16 fps	55-57 fps
	20(25)	800	3-4 fps	53-55 fps	10-11 fps	53-55 fps

Lentelė 1 - Algoritmų veikimo palyginimas emuliatoriuje

Modelis	Tikslumas	Trikampių sk.	Oslo algoritmas		Tiesioginis algoritmas	
			KT animacija	Mesh animacija	KT animacija	Mesh animacija
ančiukas	10	~670	2-3 fps	19-21 fps	3-4 fps	19-21 fps
	15	~1400	1-2 fps	15-16 fps	2-3 fps	15-16 fps
	20	~2500	1 fps	11-12 fps	1-2 fps	11-12 fps
Kvepalų buteliukas	10 (7)	~472	4-5 fps	22-23 fps	5-6 fps	22-23 fps
	15 (12)	~ 918	3-4 fps	17-18 fps	3-4 fps	17-18 fps
	20 (17)	~1604	2-3 fps	14-15 fps	2-3 fps	14-15 fps
plokštuma	10(15)	200	8-9 fps	21-22 fps	14-15 fps	21-22 fps
	15(20)	450	4-5 fps	19-20 fps	10-12 fps	19-20 fps
	20(25)	800	3-4 fps	17-18 fps	9-10 fps	17-18 fps

Lentelė 2 - Algoritmų veikimo palyginimas mobiliajame telefone – Nokia 6230i

Modelis	Tikslumas	Trikampių sk.	Oslo algoritmas		Tiesioginis algoritmas	
			KT animacija	Mesh animacija	KT animacija	Mesh animacija
ančiukas	10	~670	1-2 fps	36-39 fps	1-2 fps	25-28 fps
	15	~1400	1-2 fps	28-30 fps	1-2 fps	20-22 fps

	20	~2500	1-2 fps	22-23 fps	1-2 fps	15-17 fps
Kvepalų buteliukas	10 (7)	~472	2-3 fps	36-39 fps	2-3 fps	32-34 fps
	15 (12)	~ 918	2-3 fps	30-32 fps	2-3 fps	25-27 fps
	20 (17)	~1604	1-2 fps	24-26 fps	1-2 fps	20-22 fps
plokštuma	10(15)	200	8-9 fps	51-58 fps	7-8 fps	43-50 fps
	15(20)	450	5-6 fps	42-48 fps	6-7 fps	36-40 fps
	20(25)	800	4-5 fps	34-36 fps	4-5 fps	26-32 fps

Lentelė 3 - Algoritmų veikimo palyginimas mobiliajame telefone - Nokia 6680

Pagrindinis dalykas, kuris yra matomas atlikus analizę, tai jog Tiesioginis algoritmas yra efektyvesnis už Oslo algoritmą (vidutiniškai 1,5-2 kartus). Taip pat galima pastebėti, jog Mesh animacijos yra tiek pat efektyvios abiem algoritmais. Taip yra todėl, kad abu algoritmai saugo savo galutinį variantą ir nesikeičiant jokiems paviršiaus parametrams, nereikia perskaičiuoti paviršiaus taškų. Taigi tokio tipo animacijoje, visas vaidmuo atitenka standartinei M3G Java posistemei. Palyginus algoritmų veikimo rezultatus, tarp emuliatoriaus ir telefono, matome jog kontrolinių taškų animacijos rezultatai beveik nesiskiria, ypač Oslo algoritmo. Didžiausias skirtumas matomas Mesh tipo animacijoje. Galbūt ~30 fps yra šio įrenginio viršutinė riba. O skirtingi pokyčiai tarp emuliatoriaus bei telefono, gaunami dėl to, kad visi NURBS skaičiavimai atliekami Javos pagalba (Kur emuliatoriuje yra ribojimas našumas, kad atitiktų realius įrenginius), o atvaizdavimas, bei transformacijos realizuotos vienu atveju OpenGL ES (mobiliajam telefone), o kitu atveju standartine OpenGL (emuliatoriuje), kurios galbūt nėra galimybių apriboti, taip pat dar personaliniame kompiuteryje egzistuoja toksai dalykas, kaip dedikuota grafinė atmintis, bei trimatės grafikos spartintuvas. Gauti rezultatai dar nelabai džiugina, t.y. tokio algoritmo našumo dar nelabai užtenka realaus laiko animacijoms ar žaidimams. Tačiau modelius persikelti

Paveikslėlis 1 - NURBS paviršiaus modelis

ir su jais atlikti nesudėtingus veiksmus – jau tikrai yra įmanoma.

Adaptyvus atvaizdavimas

Testavimai buvo atlikti bandant atvaizduoti NURBS paviršius skirtingais detalumo lygiais: žemos, vidutinės bei aukštos kokybės. Kuo didesnis detalumo lygis, tuo geresnė kokybė, tuo daugiau trikampių. Iš to seka, jog norėdami išgauti geresnę kokybę turime tam aukoti žymiai daugiau resursų. Tačiau mobiliųjų įrenginių specifika yra tokia, jog jie turi nedidelius ekranus ir diapazonas, kada objektas yra aiškiai matomas yra labai nedidelis. Todėl yra ganėtinai nelogiška, skirti tiek pat daug resursų atvaizduoti aukštos kokybės objektui, kai jis nepatenka į aiškaus matymo diapazoną. Šioje vietoje galima atlikti nedidelį pakeitimą, kuris mums sutaupytų nemažai resursų. Ši technologija vadinama adaptyviu atvaizdavimu, kada atvaizduojamo objekto tikslumas parenkamas dinamiškai, pavyzdžiui, atsižvelgiant į objekto atstumą nuo žiūrėjimo taško.

Toliau yra vystomas tik tiesioginis algoritmas, o Oslo algoritmas yra atmetamas kaip nepakankamai efektyvus šiai platformai. Kad tą realizuoti, į DirectNurbsSurface klasę (bei kartu Nurbs interfeisą) įterpsime Render metodo atitikmenį – RenderLOD, kuriam bus perduodama einamoji kameros pozicija, kad galima būtų nustatyti objekto nuotolį nuo žiūrėjimo taško ir taip parinkti reikiamą tikslumą. Taip pat reikės nustatyti standartinį tikslumą, bei tikslumo pokyčius, bei jo kitimo greitį.

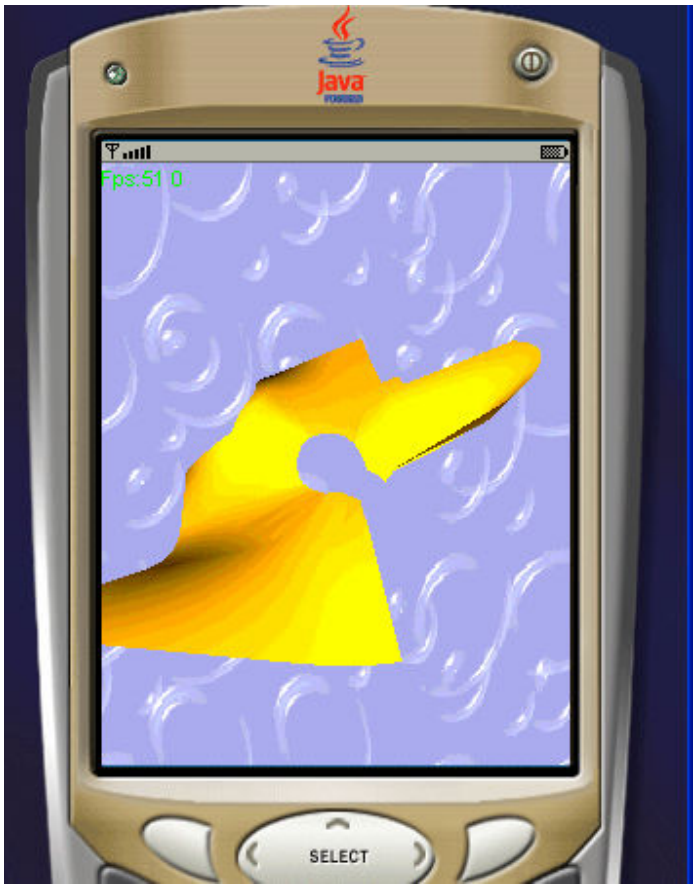
Tokiu būdu bus išgautas optimalus našumo bei kokybės santykis.

Trimmed NURBS realizacija

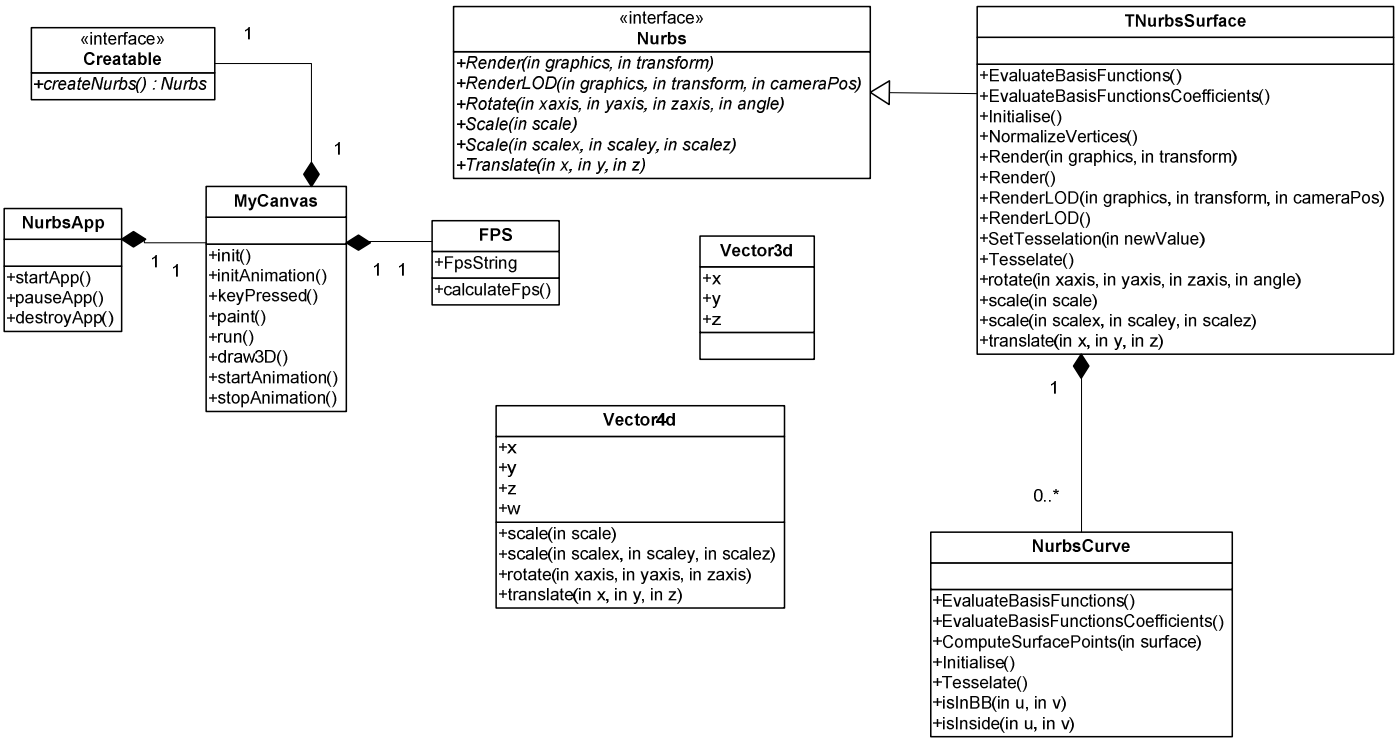
Taigi jau realizuotas pakankamai efektyvus algoritmas NURBS paviršiams atvaizduoti mobiliajame telefone. Kad pasiekti pilną rinkos standartą dar reikia pasidomėti NURBS paviršių

apkarpymais (Trimming). 3D grafikos industrijoje (kaip CAD ir CAM) yra naudojami TrimmedNurbs paviršiai, kurie įgalina atvaizduoti bet kokio sudėtingumo paviršius.

Kad papildyti dabartinę NURBS realizaciją, reikia apsibrėžti, kaip bus aprašomi iškarpymai. Patogiausia tai būtų padaryti NURBS kreivėmis, kurios įgalintų išgauti įvairiausias skylių formas. NURBS kreivės pilnai atitinka tą pačią schemą, kaip ir NURBS paviršiai bei turi tuos pačius pranašumus, pavyzdžiui, kompaktišką išraiškos formą. NURBS kreivės aprašomos parametrinėje erdvėje, tai yra, visi jos kontroliniai bei pačios kreivės taškai priklauso parametrinei erdvei.



Prie esamos realizacijos, buvo realizuota **NurbsCurve** klasė, aprašanti NURBS kreivę, realizuojanti tos kreivės apskaičiavimą, bei funkcionalumą susijusį su paviršiaus iškarpymais. Taip pat modifikuoti NURBS paviršiaus taškų apskaičiavimo, bei atvaizdavimo metodai, visur įvedant atskirą variantą – iškarpytojo NURBS paviršiaus apskaičiavimą bei atvaizdavimą. Tai nebuvo suvesta į vieną universalų algoritmą, mąstant taip jog iškarpytieji NURBS yra apibendrinantis atvejis. Pagrindinė to priežastis yra ta, jog iškarpymai reikalauja daug papildomų skaičiavimų, o tuo pačiu modeliu skaičiuojant paprastus NURBS paviršius atsiranda daug “tuščių” skaičiavimų, kas yra nekorektiška ieškant efektyvios realizacijos. Dėl to visi skaičiavimai atliekami su sąlyga ar tai yra iškarpytasis NURBS paviršius ar neiškarpytas. Atlikus šia korekcijas gauta galutinė realizacijos loginė klasių diagrama (Kadangi lieka tik vienas algoritmas, apibendrinant DirectNurbsSurface klasė pakeista į TNurbsSurface):



Kadangi iškarpytųjų NURBS skaičiavimuose atliekama daug daugiau papildomų skaičiavimų (Kreivės apskaičiavimai, nustatymai, ar taškas priklauso paviršiui, ar ne bei artimiausio taško paieška pertvarkant taškus prie skylės krašto), visa tai atsispindi veikimo analizėje. Tokie modeliai yra atvaizduojami ženkliai lėčiau. Todėl galima teigti jog kolkas iškarpytieji NURBS nelabai gali būti panaudojami interaktyviai, nes netenkina reikiamos atvaizdavimo spartos.

Išvados

Į pagrindinį šio darbo klausimą, ar galima realizuoti NURBS paviršių atvaizdavimą mobiliesiems įrenginiams, galima atsakyti – taip. Realizacija yra pavykusi ir ją galima praktiškai panaudoti, žinoma su tam tikrais apribojimais. Išanalizavus šio meto mobiliuosius įrenginius, bei trimatės grafikos standartus, paaiškėjo, jog kolkas nėra jokių standartinių priemonių NURBS paviršiams atvaizduoti. Analizei pasirinkta M3G mobiliosios grafikos standartas, dėl jo plataus paplitimo bei universalaus pritaikymo. Išanalizavus literatūrą analizei buvo pasirinkti du NURBS atvaizdavimo algoritmai – Oslo ir tiesioginis, dėl jų taikymo praktikoje bei optimalumo. Realizavus šiuos algoritmus bei pritaikius prie mobiliųjų įrenginių specifikos buvo atlikti jų veikimo bandymai emuliatoriuje bei realiuose įrenginiuose, panaudojant keletą skirtingo sudėtingumo NURBS modelių. Po šių algoritmų testavimų galime išskirti tokias pagrindines išvadas:

- Tiesioginis algoritmas yra daugumoje atvejų efektyvesnis už Oslo algoritmą
- Emuliatoriaus veikimas yra ganėtinai panašus į tikrus įrenginius, kai eina šneka apie skaičiavimus, tačiau kone dvigubai efektyvesnis, kas liečia patį atvaizdavimą. Problema yra tame, jog emuliatorius tiesiogiai naudoja kompiuterio OpenGL sistemą, kuri yra efektyvesnė, už mobiliuosiuose įrenginiuose naudojamą OpenGL ES.
- Algoritmų veikimo efektyvumas, patenkino lūkesčius ir galima teigti, jog jau yra techninės galimybės mobiliuosiuose įrenginiuose naudoti NURBS technologiją. Paskaičiavus vidutiniškai yra gaunama 3.6 fps animacijų sparta, kiekvieną kartą perskaičiuojant NURBS paviršių. Tai tikrai neblogas rezultatas. O jei kalbant apie, tai kad NURBS objektas yra apskaičiuojamas tik vieną kartą o po su juo manipuliuojama jau tik trikampių lygyje, tai čia spartos ir efektyvumo pakanka pilnai.
- Testuojant ant realių įrenginių pastebėta jog, ne visada galingesnis procesorius bei didesnė atmintinė lemia didesnę spartą. Į šiuos veiksnius susiveda nemažai vidinių realizacijos niuansų, kaip atminties valdymas, gijų valdymas ir t.t.

Atlikus bandymus, pastebėta, jog nedideliame įrenginių ekrane objekto matomumo zona nėra labai didelė ir jam nutolus yra sunku išskirti kažkokias smulkesnes detales. Dėl to buvo

realizuotas adaptyvus atvaizdavimas, kada NURBS modelio apskaičiavimo tikslumas parenkamas automatiškai nuo objekto ir stebėjimo taško tarpusavio padėties: kuo objektas arčiau, tuo jis tiksliau yra perpiešiamas, kuo jis daugiau nutolsta, tuo atliekama mažiau perskaičiavimo ciklų.

Kad pilnai išpildyti NURBS standartą buvo realizuota Trimmed NURBS modeliai. Atlikus bandymus su tokiais modeliais, jau yra matoma, kad interaktyviam naudojimui tokio algoritmo efektyvumo nepakanka: animacijų sparta svyruoja ~1fps.

Vienas iš lūkesčių, kurių nepavyko įgyvendinti tai eksportavimo įrankio realizacija, paprastesniam NURBS modelių perkėlimui iš 3D modeliavimo sistemos Blender3D. Pagrindinė problema, jog standartinis API naudojamas toje sistemoje nesuteikia priėjimo prie NURBS objektų. Priėjimas galimas tik prie NURBS kreivių. Todėl modeliai buvo konvertuojami iš VRML standarto failų.

Pagrindiniai aspektai įtakojantys šios srities vystymą yra aparatinės įrenginių dalies raida. Dabar viskas priklauso nuo to. Kuo greičiau padidės apdorojimo sparta, kuo greičiau paplis aparatinis 3D spartinimas ir t.t., tuo greičiau bus pasiekta kokybiškas NURBS objektų atvaizdavimas mobiliuosiuose įrenginiuose.

Literatūros sąrašas

- **[CM99]** G. Casciola, S. Morigi. The trimmed NURBS age, 1999
- **[Mac99]** Dean Macri. Using NURBS Surfaces in Real-time Applications, Gamasutra, 1999 Lapkričio 17, URL: http://www.gamasutra.com/features/19991117/macri_01.htm
- **[Pra01]** Vincent Prat. NURBS Curves and Surfaces tutorial, 2001.
- **[PT96]** Les Piegl, Wayne Tiller. The NURBS Book, 2nd Edition, Berlin, Germany: Springer-Verlag, 1996.
- **[SMOG04]** H. Sánchez, A. Moreno, D. Oyarzun, A. García-Alonso. Evaluation of NURBS surfaces: an overview based on runtime efficiency. 2004