



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INŽINERINĖS GRAFIKOS KATEDRA

Šarūnas Paškevičius

**KANONINĖS FORMOS PAVIRŠIŲ ATPAŽINIMAS DAUGIAVIRŠŪNIO
TINKLO MODELyje**

**RECOGNITION OF A CANONICAL SURFACE IN THE MODEL OF A
TRUNCATED SURFACE**

Baigiamasis magistro darbas

Informacinės technologijos, valstybinis kodas 6211BX016
Inžinerinės ir kompiuterinės grafikos specializacija
Informatikos inžinerija

Vilnius, 2020

Vilniaus Gedimino technikos universitetas Fundamentinių mokslų fakultetas Inžinerinės grafikos katedra		ISBN Egz. sk. Data-.....-.....	ISSN
Antrosios pakopos studijų Informacinių technologijų programos magistro baigiamasis darbas 4 Pavadinimas Kanoninės formos paviršių atpažinimas daugiaviršūnio paviršiaus modelyje			
Autorius Vadovas		Šarūnas Paškevičius Saulius Dereškevičius	
Kalba: lietuvių			
Anotacija Baigiamajame magistro darbe nagrinėjamas daugiaviršūnio tinklo modelio konvertavimas į B-reprezentacijos modelį. Buvo susipažinta ir išanalizuota daugiaviršūnio tinklo modelio ir B-reprezentacijos modelio sudarymo tvarka. Susipažinta su įvairiais atvaizdavimo formatais. Išnagrinėti ir aprašyti algoritmai, kurie bus naudojami plokštumos, cilindro, tiesės ir apkritimo atpažinimui. Darbą sudaro 8 dalys: įvadas, esamų formatų atsiradimo istorija, ribojančiosios geometrijos informacijos struktūra, daugiakampių tinklų informacijos struktūra, matematiniai tinklelių generavimo metodai, atvaizduojamų formų formatų klasifikacija, trianguliuotų tinklų aproksimacijos pagrindai ir taikymas, išvados ir rekomendacijos, literatūros sąrašas. Darbo apimtis – 53 p. teksto be priedų, 33 iliustr., 5 lent., 22 bibliografiniai šaltiniai.			
Prasminiai žodžiai: daugiaviršūnis tinklas, B-reprezentacija, cilindras, plokštuma, trimatis objektas, tinklo akis, tinklelis, trianguliacija, trikampiai			

Vilnius Gediminas Technical University Faculty of Fundamental Sciences Department of Engineering Graphics		ISBN Copies No. Date-.....-.....	ISSN
Master Degree Studies Information Technologies study programme Master Graduation Thesis 4			
Title	Recognition of a Canonical Surface in a Model of a Truncated Surface		
Author	Šarūnas Paškevičius		
Academic supervisor	Saulius Dereškevičius		
Thesis language: Lithuanian			
Annotation The final master's thesis deals with the conversion of a polygon mesh model to a B-representation model. The procedure for creating a polygon mesh model and a B-representation model was introduced and analyzed. Familiarize yourself with a variety of display formats. Algorithms for plane, cylinder, line, and circle recognition are analyzed and described. The works consist of 8 parts: introductions, history of existing shapes, boundary geometric information structure, polygonal network information structure, mathematical grid generation methods, classification of represented shape formation, basics, and applications of triangulated network approximation, conclusions, and recommendations, references. The thesis consists of 53 p. text appendixes, 33 pictures, 5 borrowed, 22 bibliographical entries.			
Keywords: polygon mesh, B-representation, cylinder, plane, 3D object, mesh, triangulation, triangles, triangular mesh			

Turinys

IVADAS.....	8
1. Esamų formatų atsiradimo istorija	12
1.1. IGES formato istorija	12
1.2. STL formato istorija	13
2. Ribojančiosios geometrijos informacijos struktūra.....	14
2.1. Kietakūnis objektas	14
2.2. Topologinis kietakūnio objekto aprašymas	15
2.3. B - reprezentacijos sąsajos tarp vienas kito	18
3. Daugiakampių tinklų informacijos sudarymas.....	21
3.1. Daugiakampiai tinklai (angl. <i>Polygon mesh</i>).....	21
3.2. Elementai, sudarantys daugiakampį tinklą	21
4. Matematiniai tinklelių generavimo metodai taikomi konvertuojant kanoninius paviršius į daugiaviršūnių tinklų formą.....	24
4.1. Delone daugiaviršūnis tinklelis	24
4.2. Reguliarus tinklelis	24
4.3. Projekcinis tinklelis	25
4.4. Plintančiojo fronto metodas.....	26
5. Atvaizduojamų formų formatų klasifikacija	29
5.1. „Native CAD“	29
5.2. „Neutral“	30
5.2.1. „Initial Graphics Exchange Specification“ (IGES).....	30
5.2.1.1. IGES failų sekcijų gilesnis aptarimas	32
5.2.1.2. IGES Trūkumai	35
5.2.2. Gaminio modelio duomenų keitimo standartas (STEP)	35
5.2.2.1. STEP formato apžvalga	36
5.2.2.2. STEP formato panaudojimo aplinka	36

5.3.	„Lightweight representations“	37
5.4.	„Tessellated“	37
5.4.1.	STL formato apžvalga	37
5.5.	Apibendrinimas	39
6.	Daugiaviršūnių tinklų aproksimacijos pagrindai ir taikymas.....	40
6.1.	2D Poligoninės kreivės konvertavimas į tiesių ir lankų rinkinį.....	40
6.2.	3D Daugiaviršūnių tinklų konvertavimas į plokštumą.....	43
6.3.	3D Daugiaviršūnių tinklų konvertavimas į cilindrą	45
	Išvados ir rekomendacijos.....	49
	Literatūros sąrašas	50

Lentelių sąrašas

1 lentelė. IGES failų skyriai (Kennicott, 2001).....	31
2 lentelė. IGES failų struktūra (Kennicott, 2001)	31
3 lentelė. Katalogų įvedimo skyriaus subjekto požymiai.....	33
4 lentelė. Pagrindinių IGES failų objektų sąrašas	34
5 lentelė. Parametrinių duomenų sekcija (Kennicott, 2001)	35

PAVEIKSLIŲ SĄRAŠAS

1 pav. CAD dizaino pavyzdys.....	8
2 pav. Sferinės frezos B-reprezentacija ir reprezentacija daugiaviršūniu tinklu.....	10
3 pav. Kietakūnio modelio pavyzdys	14
4 pav. Kietakūnio objektas išreikštas lopiniais	14
5 pav. B-reprezentacijos modelio struktūra	15
6 pav. 4 paviršių piramidė.....	16
8 pav. Briaunomis apibrėžta piramidė	16
9 pav. Taškais apibrėžta piramidė	17
10 pav. Vidinės ir išorinės kilpos pavyzdys	17
11 pav. Kūno aprašymas B-reprezentacijos	18
12 pav. Lopinių hierarchinė schema	18
13 pav. Lopinio hierarchinė schema	19
14 pav. Kraštinės hierarchinė schema.....	19
15 pav. Viršūnės hierarchinė schema.....	20
16 pav. Elementai sudarantys daugiakampį tinklą.....	21
17 pav. Taškus ir kraštines aprašanti lentelė	22
18 Pav. Paviršiaus ir viršūnės tinklelio pavyzdys (angl. <i>Face-vertex meshes</i>).....	22
19 pav. Paviršiaus ir viršūnės tinklelio pavyzdys	23
20 pav. Daugiaviršūnio tinklelio sudarymo būdai	24
21 pav. Keturkampis reguliarus tinklelis	25
22 pav. 3D reguliaraus tinklelio pavyzdys.....	25
23 pav. Dimensijų projekcija	26
24 pav. Plintančio fronto tinklelio generavimas	28
25 pav. Kreivės konvertavimas į tiesių ir lankų rinkinį algoritmo schema.....	40
26 pav. Taškų rinkinio pavyzdys	41
27 pav. Programa apkritimui per 3 taškus rasti.....	42
28 pav. Informacijos rinkinys po programos pritaikymo.....	43
29 pav. Plokštuma per 3 taškus	43
30 pav. Atstumas tarp taško ir plokštumos	45
31 pav. Cilindras išreikštas daugiaviršūniu tinklu	46
32 pav. Segmentai, sudarantys cilindrą.....	46
33 pav. Pasirinkti segmentai	47

Trumpinių sąrašas

USAF – Jungtinių Valstijų oro pajėgos

CAD – kompiuterinis dizainas

CAM – kompiuterinė gamyba

APT – automatinis programavimo įrankis

IGES – angl. *Initial Graphics Exchange Specification*

PDES – produktų duomenų mainų specifikacija

3D – trijų dimensijų vaizdas

IVADAS

Esant išsivysčiusioms internetinėms technologijoms, didėja galimybės pasidalinti darbais. Vienas proceso dalyvis gali perduoti savo darbo rezultatus kitam proceso dalyviui, ir šis gali tęsti darbą panaudodamas ankstesniame etape sukurtus duomenis. Ne išimtis yra ir dizaineriai, konstruktoriai. Pavyzdžiui, tokią darbo pasidalinimo tvarką galima stebėti interjero projektus įgyvendinančiose įmonėse. Interjero dizaineris rūpinasi estetiniu vaizdu, užpildo interjerą reikalingais baldais, įranga ir kitais aksesuarais, o baldų konstruktorius tęsia darbą naudodamasis dizainerio sukurtą medžiaga ir toliau informaciją papildo taip, kad būtų galima pateikti gamybai. 1 pav. pateiktame pavyzdyje matyti, kaip dizaineris sprendžia turinį virtuvės komponavimą interjere, jos funkcinių turinių, priima sprendimus dėl spalvų. Baldų konstruktorius pratęsia darbą užpildydamas interjero dizainerio sukurtą turinį reikalingais konstrukciniais sprendimais. Jis nusprendžia, iš kokių detalių bus sudarytas baldas, kaip viduje bus išdėstytos lentynėlės, kokios konstrukcijos panaudoti stalčiai, kokiais tvirtinimo elementais detalės bus sutvirtintos tarpusavyje, konstrukciją susieja su konkrečia buitine technika ir t.t.



1 pav. CAD dizaino pavyzdys

Kaip matyti iš pateikto pavyzdžio, interjero dizainerio sukurtą CAD informaciją, yra nuosekliai panaudojama baldų konstruktoriaus darbe. Problema, kad CAD įrankiai, naudojami kiekvieno šio proceso dalyvio darbe, turi skirtingus reikalavimus, nes uždaviniai, kuriuos reikia spręsti, yra skirtingi. Dizaineriui reikia turėti įrankį, kuriuo greitai ir patogiai galima kurti tūrius, manipuluoti jais. Taip pat svarbu manipuluoti medžiagų spalvomis, tekstūromis, turėti galimybę generuoti fotorealistinius vaizdus. Baldų konstruktorius pirmiausia susiduria su uždaviniu – kurti baldą kaip gaminį, kurį sudaro atskiri komponentai: jo struktūrą, tiksliai suformuoja baldo komponentų geometrija. Pageidautina turėti galimybę parametrais apibrėžti baldo konstrukciją, taip valdant parametrus gauti vis kitokių gabaritų ar

sudėties gaminį arba parastai ir efektyviai koreguoti jo gabaritus. Greta to turi būti gerai išvystytas techninės 2D dokumentacijos generavimo mechanizmas. Dar vienas aspektas, kuris skiria šias sistemas, yra kaina. Vieną ar kitą dizainerio funkciją reikia realizuoti praktiškai kiekviename pardavimo salone, tai reiškia, kad su tuo susidurs daug didesnis ratas žmonių negu ruošiant medžiagą gamybai. Taigi, pradiniam etape pageidautina turėti didesnę darbo vietų kiekį, paprastesnio vartojimo ir pigesnius įrankius, o konstravimo etape jau pasitelkiami sudėtingesni, aukštesnės klasės įrankiai. Atitinkamai yra didesnė ir jų kaina.

Dažniausiai problemų kyla duomenų apsikeitime. Santykinai, pagal turinio modelio saugojimo principą, visas CAD sistemas galime skirstyti į dvi kategorijas:

Sistemos, kurios saugo duomenis apie projektuojamo objekto geometriją, vadinamos daugiaviršūniais tinklais (angl. *Polygonal mesh*).

Sistemos, kurios saugo duomenis matematiškai aprašomais (kanoninės formos) paviršių (plokštumos, cilindrai, piramidės, torai, B-splainai) lopiniais, topologiškai susietais į vieną sandarų kūną, vadinamos B-reprezentacija (angl. *Boundary Representation*, ribinė reprezentacija).

Kiekvienas saugojamo darbo su informacija principas būdingas tam tikroms sritims. Daugiaviršūniais tinklais paremtos sistemos labiau tinka greitai ir patogiai manipuliuoti konstruojamu objektu, ypač vartotinos įvairiuose spalvinimuose, reikalauja mažesnių kompiuterio resursų ir talpina didelius objektų kiekius viename projekte. Tokių sistemų tipiniai atstovai:

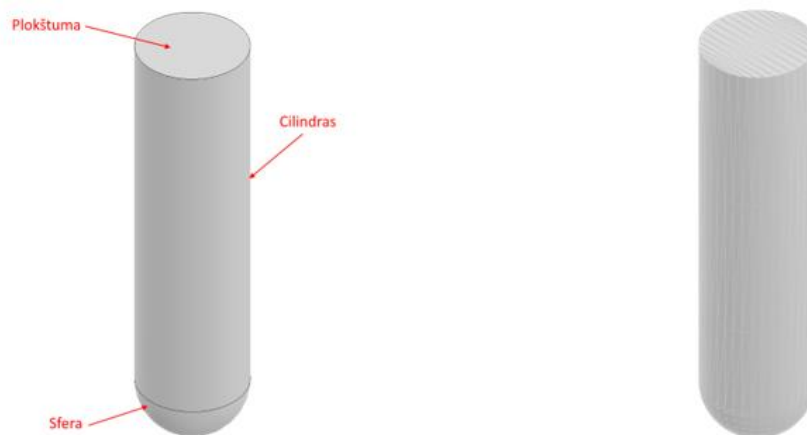
1. „SketchUp“
2. „Blender“
3. „3D Studio Max“
4. „PRO100“

B-reprezentacija paremtos sistemos tinka ten, kur reikalingas aukštas darbo tikslumas, preciziniai 2D brėžiniai. Praktiškai visos dabartinės CAD sistemos, kuriose sprendžiami inžineriniai uždaviniai, yra paremtos B-reprezentacijos modeliu. Tokių sistemų atstovais galima įvardinti:

1. „Autodesk Inventor“
2. „Revit“
3. „AutoCAD“
4. „Solid Works“
5. „Solid Edge“ ir t.t

Gana neblogai yra išspręstas duomenų perdavimas iš vienos tos pačios kategorijos sistemos kitai, nes duomenys, iš principo, užrašomi tokioje pačioje formoje. Pakankamai gerai vyksta duomenų

pasikeitimas tarp B-Reprezentacijos ir sistemos su daugiaviršūniu tinklu, nes kanoninės formos lopinį yra gana nesudėtinga konvertuoti į daugiaviršūnį tinklą.



2 pav. Sferinės frezos B-reprezentacija ir reprezentacija daugiaviršūniu tinklu

Kaip matyti iš aukščiau pateikto paveikslėlio, tas pats objektas, supaprastintas sferinės frezos modelis, yra pateikiamas skirtingai. Jeigu kairėje pusėje B-reprezentacijos modelyje yra nesunku nustatyti, kokios formos yra paviršius, koks yra, pavyzdžiui, frezos diametras, jos galo užapvalinimo spindulys, tai daugiaviršūniame tinkle (dešinėje), mes neturime tokios informacijos, negalime pasakyti, iš kokių paviršių yra sudarytas pateikiamas objektas. Toks objektas yra užrašomas kaip elementarių trikampių visuma. Iš to, kas parašyta aukščiau, aišku, kad dizaineriai dažniausiai vartoja daugiaviršūniais tinklais paremtas sistemas, o konstruktoriai naudoja B-Reprezentacija paremtas sistemas. Reiškia informacijos judėjimas vyksta iš daugiaviršūnio į B-reprezentaciją. Kaip tik šita, mums aktuali duomenų apsisikeitimo kryptis, yra mažai nagrinėta, ir aš savo darbe ėmiausi jos tyrinėjimo, sprendimų paieškos.

Pasirinkau konkrečias sistemas, išsikėliau tikslą išspręsti duomenų perdavimo uždavinį: daugiaviršūnio tinklo modelio konvertavimą į B-reprezentaciją. Pagrindinė problema, kaip daugiaviršūniame tinkle atpažinti kanoninės formos lopinius. Šiame darbe apsiribojau tik plokščių ir cilindrinų paviršių atpažinimu, nes laikas ribotas.

Kaip eksportuojančią duomenis pasirinkau CAD sistemą „SketchUp“, kaip priimančią CAD sistemą Autodesk „Inventor“. Tai sąlygojo keli faktoriai:

1. „SketchUp“ sistema yra paplitusi architektų, interjero dizainerių tarpe, todėl sprendžiama problema yra aktuali.
2. „SketchUp“ turi gerai išvystytą įskiepių kūrimo sistemą (API), kurios pagrindą sudaro Rubi algoritminė kalba.

3. Autodesk „Inventor“ yra man, kaip studentui, lengvai prieinama ir aš ją gerai pažįstu.
4. Autodesk „Inventor“ turi „STEP“ transliatorius, kurie man leis įtraukti importuotus duomenis į Autodesk „Inventor“ nerašant specialių įskiepių.

1. Esamų formatų atsiradimo istorija

1.1. IGES formato istorija

IGES formatas buvo pradėtas kurti Jungtinių Valstijų oro pajėgų (USAF) ir integruotas kompiuterinės gamybos (ICAM) projekto iniciatyva 1976 m. (Ruiter, 1988)

ICAM siekė sukurti procedūras, procesus ir programinę įrangą (CAD / CAM), kurios integruotų visas aviacijos ir kosmoso gamybos operacijas ir taip smarkiai sumažintų sąnaudas. Anksčiau USAF gamybos technologijos programa aptarnavo automatinio programavimo įrankių kalbą, skirtą skaitmeniniu būdu valdomų staklių programavimui. Norint panaikinti duomenų atkūrimą tarp projektavimo ir gamybos, vienas iš ICAM tikslų buvo sukurti CAD programinę įrangą, kuri automatiškai sugeneruotų skaitmeninio valdymo programas labai sudėtingoms kompiuterinėms skaitmeninio valdymo staklėms, naudojamoms visoje aviacijos ir kosmoso pramonėje. Rimta problema buvo duomenų, gautų tuo metu naudojamų daugelio CAD sistemų, nesuderinamumas. USAF / ICAM 1978 m. sukvietė posėdį Nacionaliniame standartų biure (dabar žinomą kaip Nacionalinis standartų ir technologijos institutas arba NIST), kad būtų išspręsta ši problema. „Boeing“ pasiūlė parduoti savo CAD vertimo programinę įrangą USAF už vieną JAV dolerį. USAF priėmė šį pasiūlymą ir sudarė sutartį su NIST, kad suburtų vartotojų ir pardavėjų grupę, įskaitant „Boeing“, „General Electric“, „Xerox“, „Computervision“, „Application“ ir kitus, kad toliau tobulintų ir išbandytų šią programinę įrangą. Nors praktika buvo pradėti ICAM plėtojimą pavadinti žodžiu integruotas manant, kad bus sparčiai plėtojama grafinių mainų programinė įranga, USAF nusprendė, kad IGES bus pirminis šios grafikos mainų programos pavadinimas (AutoDesk, 2020).

Nuo 1988 m. DoD reikalavo, kad visa skaitmeninė gaminio ir gamybos informacija (PMI), susijusi su ginklų sistemų sutartimis (inžineriniai brėžiniai, schemas ir kt.), būtų teikiama elektronine forma, pavyzdžiui, IGES formatu. Todėl „CAx“ programinės įrangos pardavėjams, norintiems parduoti savo gaminius „DoD“ subrangovams ir jų partneriams, reikėjo, kad jų programinės įrangos exportas ir importas būtų paremtas IGES formatu.

Nuo 1980 m. ANSI standartas, IGES buvo naudojamas automobilių, aviacijos ir laivų statybos pramonėje. Jis buvo naudojamas ginklų sistemoms, nuo „Trident“ raketų nukreipimo sistemų iki ištisų orlaivių vežėjų. Šiuos dalių modelius gali tekti naudoti po metų, nes originalios projektavimo sistemos pardavėjas pasitraukė iš verslo. IGES failai suteikia galimybę pasiekti šiuos duomenis praėjus dešimtmečiams. Šiandien interneto naršyklių papildinių žiūrintieji turi galimybę prieš 20 metų sukurtus IGES failus žiūrėti iš bet kurios pasaulio vietos (TransMagic, 2020).

Po pirmojo STEP leidimo (ISO 10303) 1994 m. susidomėjimas tolesne IGES plėtra sumažėjo, o 5.3 versija 1996 m. buvo paskutinis paskelbtas standartas.

1.2.STL formato istorija

STL 1987 m. išrado „Albert Systems 3D Systems“ konsultavimo grupė. Formatas buvo sukurtas pirmiesiems „3D Systems“ komerciniams 3D spausdintuvams. Pradinio išleidimo formatas 22 metus išliko palyginti nepakitęs. 2009 m. buvo pasiūlyta atnaujinti formatą, pasivadinusį STL 2.0. (Chakravorty, 2020)

2. Ribojančiosios geometrijos informacijos struktūra

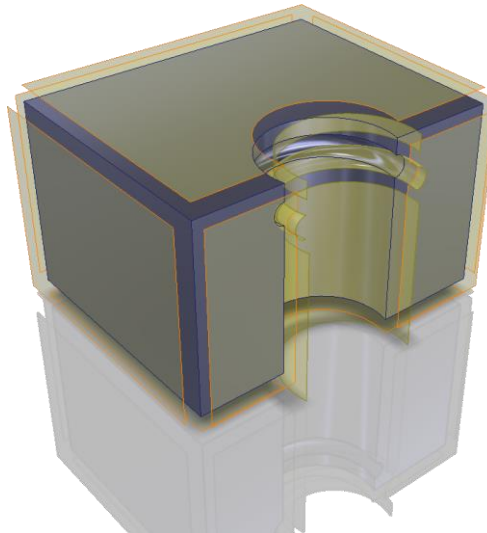
2.1. Kietakūnis objektas

Tai geometrinis kūnas, kuris sukuriamas CAD sistemoje, skirtas saugoti duomenis apie realios detalės geometrinę formą. Tokie duomenys yra pateikiami CAD sistemai, kuri sugeba šiuos duomenis interpretuoti ir atvaizduoti ekrane. Naudodamas CAD sistemoje esančias funkcijas, vartotojas gali patogiai formuoti šiuos duomenis, redaguoti, taip pat virtualioje aplinkoje kurti esamus arba projektuojamus realius pasaulio objektus – detales. (žr. 3 pav.)



3 pav. Kietakūnio modelio pavyzdys

Kietakūnis modelis iš tikrųjų nėra kietakūnis, tai yra grupė lopinių, kurie dengia kietakūnio tūrį (žr. 4 pav.). Toks modelio aprašymo būdas vadinamas B-reprezentacijos (angl. B-Rep). B-reprezentacija pateikia išsamų kietakūnio geometrijos aprašymą, geometriją.



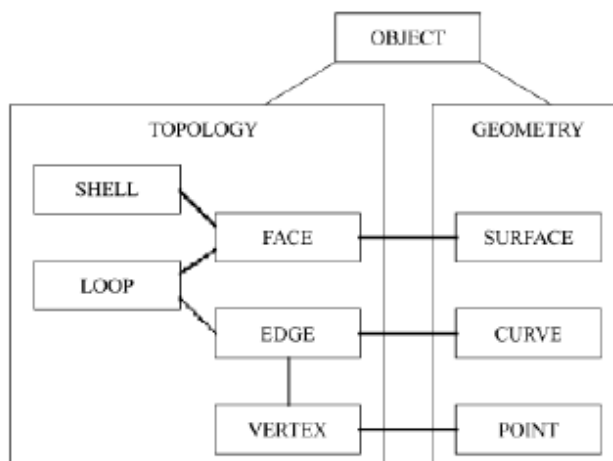
4 pav. Kietakūnio objektas išreikštas lopiniais

Kietakūnis yra ypatingas B-reprezentacijos atvejis, kai paviršiai yra sandariai sujungti išilgai visų kraštų ir sudaro uždarą tūrį. Ribojančiosios geometrijos modelį sudaro dvi dalys: topologinė ir

geometrinė. Topologija nusako ribojančiosios geometrijos modelio struktūrą, tai yra hierarchinę modelio struktūrą. Kaip minėta prieš tai, vien topologijos negana apibūdinti modelio formą, topologija nusako modelio struktūrą arba kaip modelis yra sujungtas. Norėdami visiškai suprasti modelio formą, turime naudoti modelio paviršius ir kraštines.

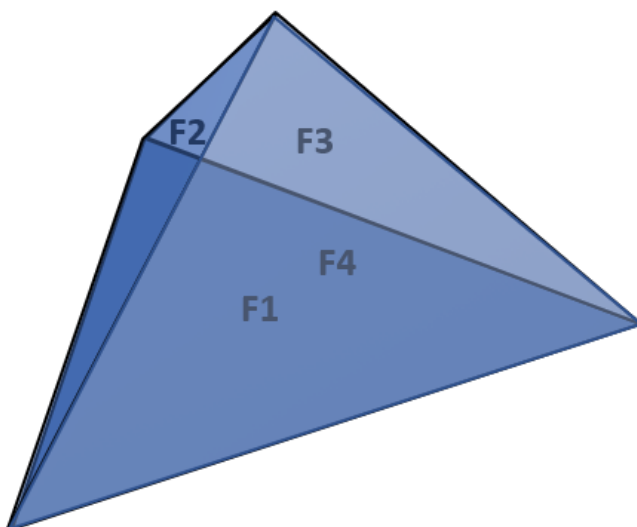
2.2. Topologinis kietakūnio objekto aprašymas

B-representacija yra duomenų rinkinys, kurį sudaro dvi pagrindinės dalys, geometrija ir topologija (Stroud, 2006). Geometrinėje dalyje yra pateikiama informacija apie Euklido erdvę, kurioje galima rasti paviršiaus, kreivės ir taškų formos aprašymus. Topologijoje yra aprašyti geometrijos elementų tarpusavio ryšiai. B-representacijos struktūros apžvalga pateikiama 5 pav.



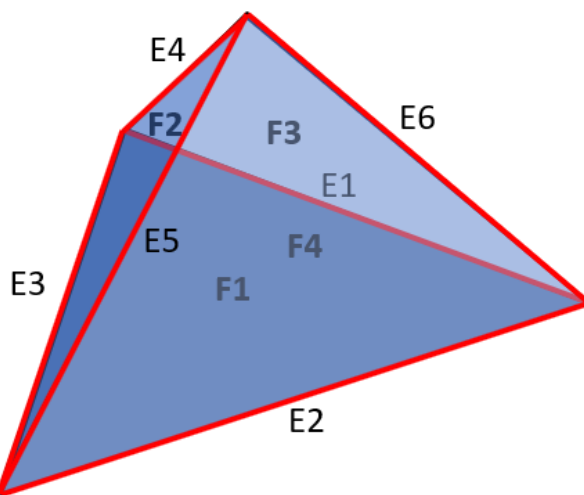
5 pav. B-representacijos modelio struktūra

3D erdvėje esantys paviršiai gali sudaryti kietakūnį. Ribos esančios tarp regionų, kur paviršius egzistuoja ir kur paviršius neegzistuoja yra vadinamos ribojančiomis, atkertančiomis, apkapojančiomis kreivėmis (angl. *boundary*, *trimming edge*). Pavyzdys, B-representacijos, turinčios keturis paviršius, yra piramide (žr. 6 Pav.).



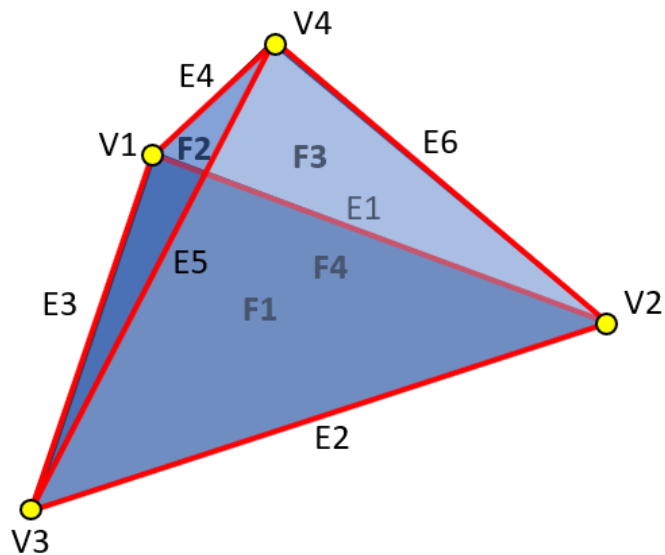
6 pav. 4 paviršių piramidė

Kiekvienas erdvėje esantis paviršius dar yra apjungiamas briaunomis (žr. 8 pav.).



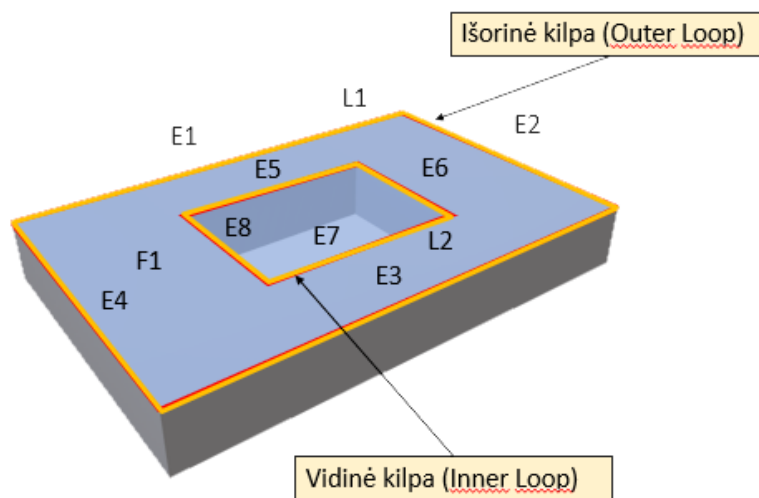
7 pav. Briaunomis apibrėžta piramidė

Kiekvienos briaunos gale yra taškas. Vienai briaunai nusakyti reikalingi 2 taškai, vienas briaunos taškas gali priklausyti kelioms briaunoms. Kaip matome (žr. 9 pav.) paveiksliuke piramidė turi 4 taškus, kurie jungia 6 briaunas.



8 pav. Taškais apibrėžta piramidė

Jeigu modelyje yra iškirtimas, kaip pavaizduota paveiksluke (žr. 10 pav.), tuomet tokio modelio paviršiams aprašyti yra naudojamos kilpos (angl. *Loop*). Toks paviršius gali būti apibrėžtas dviejų tipų kilpomis. Viena iš kilpų, apibrėžiančių modelio kontūrą, vadinama išorine kilpa (angl. *Outer loop*). Ši kilpa nusako abstraktaus kanoninio paviršiaus ribas ir taip paviršiaus ir kilpos junginys sudaro lopinio aprašą. Kita kilpos tipas vadinamas vidine kilpa (angl. *Inner Loop*). Ši kilpa nurodo visus iškirtimus, kurie buvo atlikti detalėje. Tokio tipo kilpų detalėje gali būti begalybė.



9 pav. Vidinės ir išorinės kilpos pavyzdys

2.3.B - reprezentacijos sąsajos tarp vienas kito

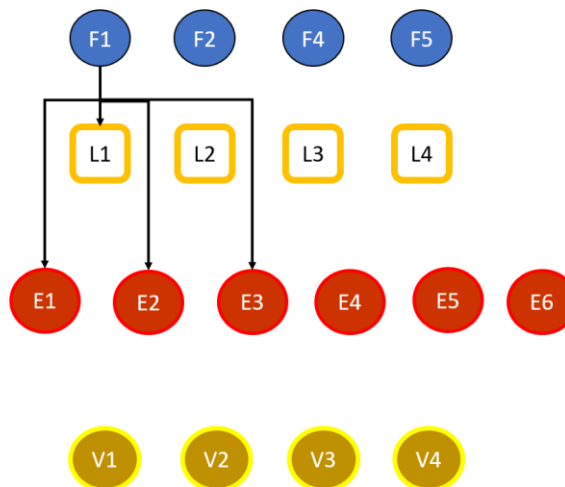
B-reprezentacijos aprašymas prasideda nuo kūno (angl. *Body*). Kad geriau iliustruoti B-reprezentacijos struktūrą, panaudokime piramidę (žr. 6 pav.), kurią aptarėme praeitame poskyryje. Kūno aprašymo topologinė forma atrodo taip:



10 pav. Kūno aprašymas B-reprezentacijos

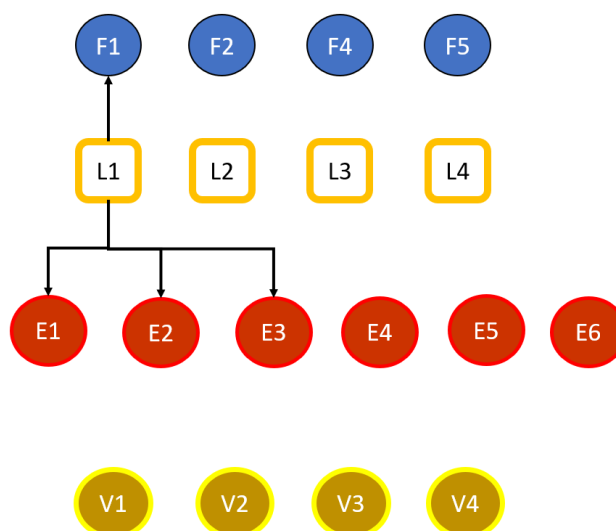
Kūno aprašymas prasideda nuo jo paviršių. Paviršius sudaro kūnų, kevalų, lopinių ir briaunų sąrašai.

Pradėkime nuo lopinių. Kaip matome iš paveiksluko (žr. 12 pav.), lopinys žino informaciją, kokiai kilpai jis priklauso ir kokios kraštinės jam priklauso.



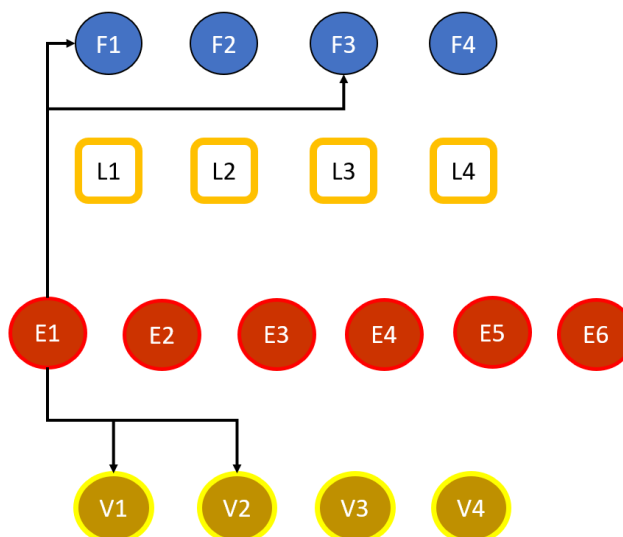
11 pav. Lopinių hierarchinė schema

Eidami žemyn sutinkame kilpas. Jos savyje turi informaciją apie tai, kokiam lopiniui jos priklauso ir kokios briaunos jį supa (žr. 13 pav.)



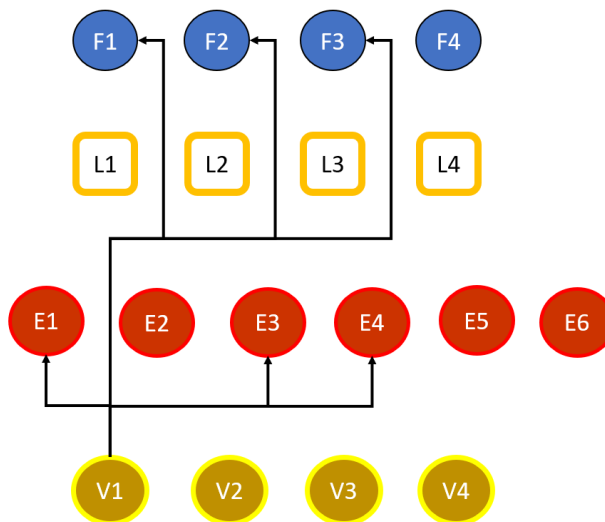
12 pav. Lopinio hierarchinė schema

Toliau schemoje yra kraštinės. Jos savyje užsirašo kokiems lopiniams priklauso, kadangi viena kraštinė gali priklausyti daugiau negu vienam lopiniui. Taip kraštinės žino apie savo viršūnes (žr. 14 pav.).



13 pav. Kraštinės hierarchinė schema

Paskutinė schemoje yra viršūnė. Ji žino, kokiems lopiniams ji priklauso, kadangi viena viršūnė gali priklausyti ne vienam lopiniui, taip pat žino, kokioms viršūnėms kraštinės priklauso (žr. 15 pav.).



14 pav. Viršūnės hierarchinė schema

Visa ši grafo struktūra yra apjungiamą į lopinių kevalą. Šį kevalą sudaro lopinių, kilpų, briaunų, viršūnių sąrašai.

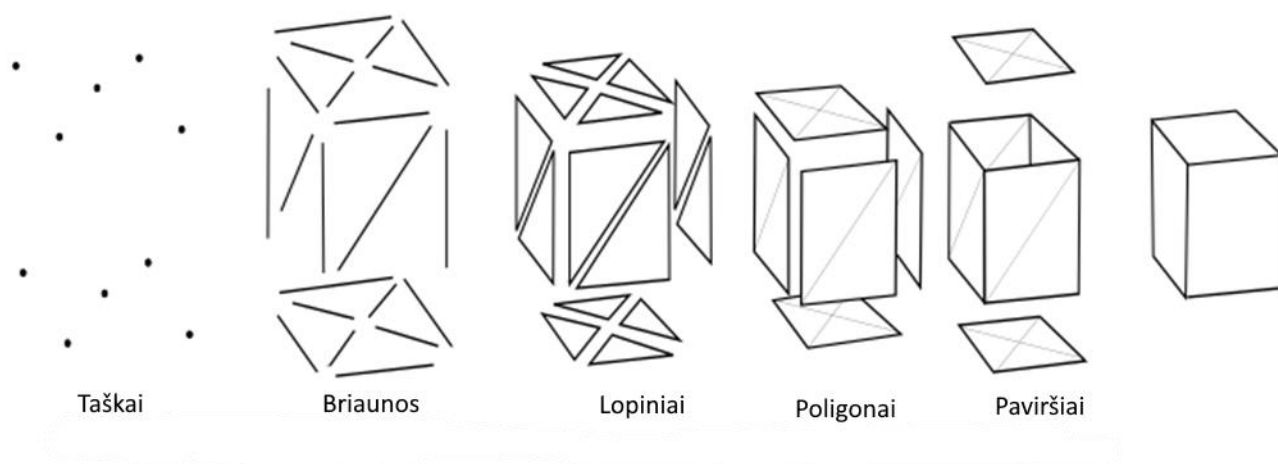
3. Daugiakampių tinklų informacijos sudarymas

3.1. Daugiakampiai tinklai (angl. *Polygon mesh*)

Daugiakampiai tinklai yra suglaustų tarpusavyje trikampių rinkinys. Jie yra užrašomi kaip viršūnių, kraštų ir plokščių paviršių rinkinys, kuris 3D kompiuterinėje grafikoje ir kietakūniame modeliavime apibrėžia daugiakampio objekto formą. Paviršius paprastai susideda iš trikampių (angl. *Triangle mesh*) nes tai reikalinga vykdant foto realistinį atvaizdavimą ar kitų uždavinių sprendimą, pavyzdžiui, frazavimo įrankio trajektorijos skaičiavimą.

3.2. Elementai, sudarantys daugiakampį tinklą

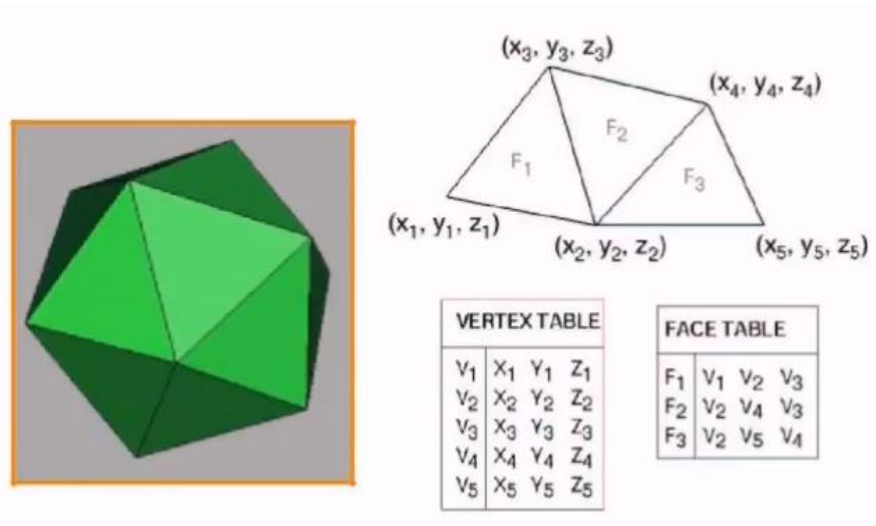
Objektai, sukurti naudojant daugiakampius tinklus, turi saugoti skirtingų tipų elementus. Tai apima viršūnes, kraštines, lopinius ir daugiakampius. Daugelyje programų saugomos tik viršūnės, kraštai ir paviršiai arba daugiakampiai. Foto realistinį atvaizdavimą gali palaikyti tik 3-jų kraštinių paviršius, todėl daugiakampiai turi būti sudaryti iš daugelio jų, kaip parodyta paveikslėlyje (žr. 16 Pav.)



15 pav. Elementai sudarantys daugiakampį tinklą

Kiekvienas trikampis elementas yra sudarytas iš taškų, kurie yra išsidėstę erdvėje. Šie taškai erdvėje yra sujungiami linijomis ar dar kitaip vadinamomis kraštinėmis, kurios sudaro uždarą briaunų rinkinį, sudarytą iš trikampių.

Kiekvienas sukurtas paviršius turi sąrašą sąsajų, kurios nurodo kur yra jam priklausantys taškai ir kraštinės (žr. 17 Pav.) Kaip matome iš taškų lentelės, kiekvienas taškas turi savo koordinates erdvėje. Vienas taškas gali priklausyti kelioms briaunoms, kurios sudaro paviršių. Šie paviršiai yra sujungiami ir sukuriamas trimatis paviršiaus modelis.



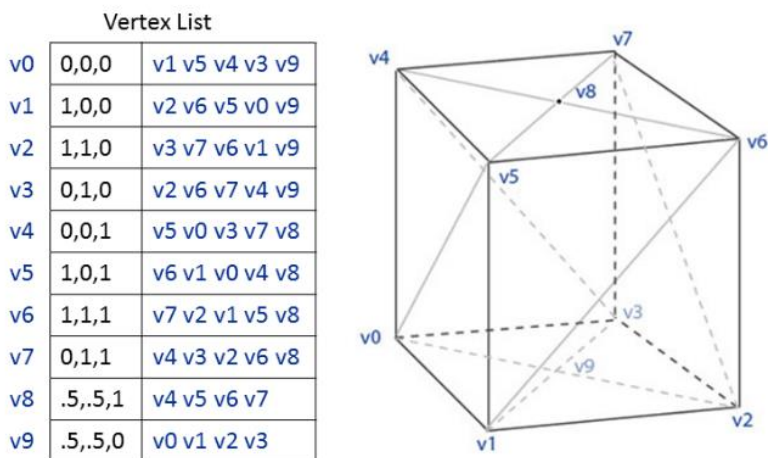
16 pav. Taškus ir kraštines aprašanti lentelė

Daugiakampis tinklelis gali būti vaizduojamas įvairiais būdais, naudojant skirtingus metodus, kurie leidžia išsaugoti informaciją apie viršūnes, kraštines ir paviršius. Šie metodai yra:

- Paviršiaus ir viršūnės tinklelis (angl. *Face-vertex meshes*)
- Viršūnės – viršūnės tinklelis (angl. *Vertex-vertex meshes*)

3.3. Viršūnės – viršūnės tinklelis (angl. *Vertex-vertex meshes*)

Šis daugiakampio tinklelio tipas vaizduoja objektą kaip viršūnių, sujungtų su kitomis viršūnėmis, aibę. Tai paprasčiausias atvaizdavimo tipas, tačiau jis nėra plačiai naudojamas, nes paviršiaus ir briaunos informacija yra numanoma. Taigi, norint sudaryti atvaizduojamų paviršių sąrašą, būtina pereiti duomenis. Be to, operacijos briaunomis ir paviršiais nėra lengvai atliekamos. Tačiau šio tipo tinkleliai yra naudingi dėl nedidelės saugojimo vietos.

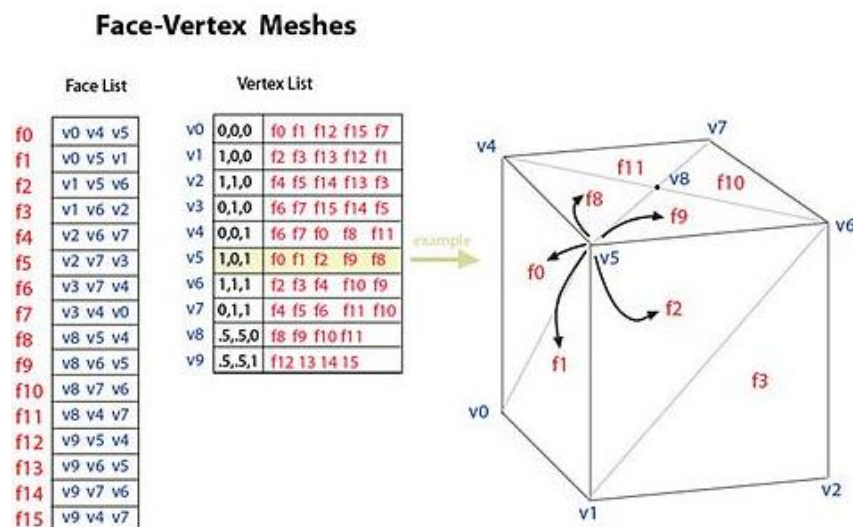


17 Pav. Paviršiaus ir viršūnės tinklelio pavyzdys (angl. *Face-vertex meshes*)

Paveikslėlyje (18 pav.) yra keturių pusių dėžutė, pavaizduota aptariamu tinkleliu. Kiekviena viršūnė indeksuoja savo kaimynines viršūnes. Atkreipkite dėmesį, kad paskutinės dvi viršūnės, 8 ir 9, esančios viršuje ir apačioje, turi keturias sujungtas viršūnes, o ne penkias. Bendroji sistema turi sugebėti valdyti savavališką skaičių viršūnių, prijungtų prie bet kurios viršūnės (Smith, 2006).

3.4. Lopinio ir viršūnės tinklelis (angl. *Face-vertex meshes*)

Vienas labiausiai paplitusių daugiakampių tinklelių vaizdavimo tipas yra lopinio ir viršūnės tinklelis. Šio tipo tinkleliai vaizduoja objektą kaip lopinių ir viršūnių rinkinį. Lopinio - viršūnės tinklelio modeliai pagerėja naudojant viršūnę – viršūnės tinklelį (Vertex-vertex meshes), nes jie leidžia aiškiai ieškoti lopinių viršūnių ir lopinių, supančių viršūnes. Paveikslėlyje (19 pav.) pateiktas šio tipo tinklelio pavyzdys. Paveikslėlyje matyti, kad viršūnė V5 yra išskirta, tai yra padaryta norint išskirti lopinius, supančius šią viršūnę. Reikia paminėti, kad šiame pavyzdyje kiekvienas lopinys turi būti tiksliai su 3 viršūnėmis. Tačiau tai nereiškia, kad kiekviena viršūnė turi tą patį aplinkinių lopinių skaičių.



18 pav. Paviršiaus ir viršūnės tinklelio pavyzdys

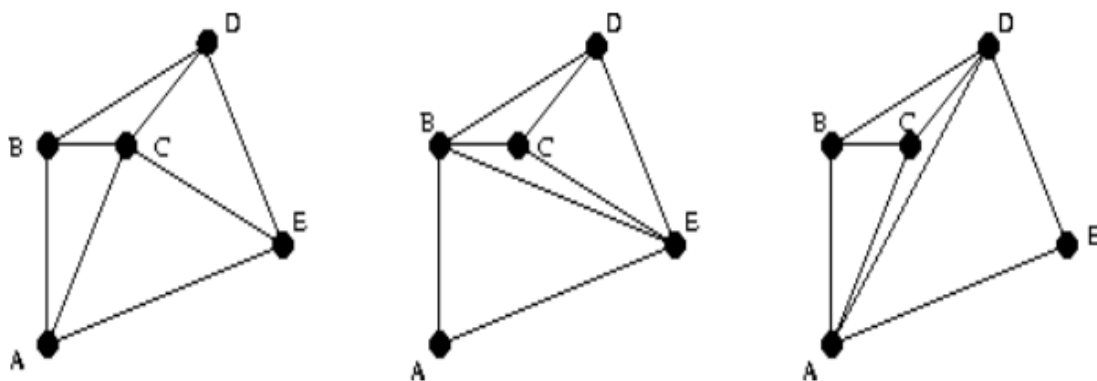
Modeliavimas reikalauja lengvo visų konstrukcijų judėjimo. Su lopinių - viršūnių tinkleliu lengva rasti paviršiaus viršūnes. Taip pat viršūnių sąraše yra paviršių, sujungtų su kiekviena viršūne, sąrašas. Tačiau kraštai yra netiesioginiai, todėl vis dar reikia ieškoti, kad būtų rasti visi paviršiai, supantys tam tikrą lopinį. Lopinio - viršūnės tinkleliams taip pat sunku atlikti kitas dinamiškas operacijas, tokias kaip paviršiaus padalijimas ar sujungimas.

4. Matematiniai tinklėlių generavimo metodai taikomi konvertuojant kanoninius paviršius į daugiaviršūnių tinklų formą

4.1. Delone daugiaviršūnis tinklėlis

Daugelis daugiaviršūnių tinklėlių generavimo algoritmų yra paremti Delone daugiaviršūnių tinklėliu ir Delone kriterijumi. Nors pats Delone metodas ir yra priskiriamas plintančio fronto algoritmui, tačiau Delone kriterijus ir jo idėjos yra taikomos daugelyje modeliavimo sistemų. (Blazek, 2015)

Kelis erdvėje esančius lopinius perteikti į daugiaviršūnį tinklėlį galima keliais būdais. Pavyzdyje pateiktus penkis taškus galima perteikti į daugiaviršūnį tinklėlį trimis būdais (Peterson, 2020):



19 pav. Daugiaviršūnio tinklėlio sudarymo būdai

Tam pasiekti dažniausiai naudojame Delone daugiaviršūnius tinklėlius, kadangi jie yra vieni iš geriausių ir dažniausiai naudojamų.

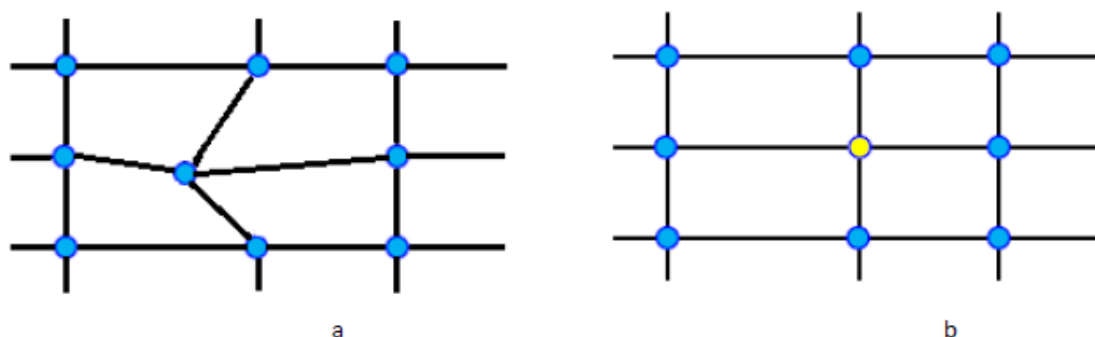
Duotus taškus, naudojant Delone daugiaviršūnius tinklėlius, galima aprašyti taip:

Dviejose dimensijose dar kitaip vadinamose 2D, beveik visose duotųjų taškų grupėse egzistuoja Delone daugiaviršūniai tinklėliai. Kadangi šie daugiaviršūniai tinklėliai visada yra unikalūs, aišku su prielaida, kad į apskritimą neįeina 4 taškai. Delone daugiaviršūniai tinklėliai sumažina mažus kampus, taip mažindami netinkamų (smailių) trikampių buvimą. Tačiau, jeigu visi taškai yra išsiriakavę vienoje tiesėje, Delone daugiaviršūnio tinklėlio įvykdyti yra neįmanoma. Jeigu yra naudojami 3D daugiaviršūniai tinklėliai, vietoje apskritimo yra naudojama sfera, kurios viduje taip pat neturi atsirasti papildomų taškų.

4.2. Reguliarus tinklėlis

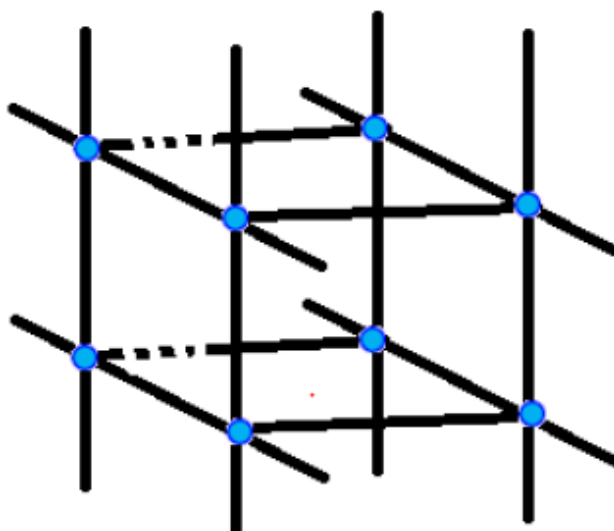
Reguliariais tinklėliais vadinsime tokius elementus, kurie turės toki patį kampų skaičių bei visos jų viršūnės taip pat turės toki patį kampų skaičių (Pascal Frey, Paul Louis George, 2000)

Tokio tinklelio pavyzdžiu gali būti kiekvienas elementas iš reguliaraus keturkampio tinklelio, kuris yra keturšonis, ir visos viršūnės turi keturi kampus (22 Pav. b). Tokio tipo tinklelio elementai nebūtinai turi būti taisyklingi keturkampiai (22 Pav. a).



20 pav. Keturkampis reguliarus tinklelis

Tokį tinklelį galime sudaryti ir 3D



21 pav. 3D reguliaraus tinklelio pavyzdys

Toks paviršiaus dalinimo principas nėra labai sudėtingas. Grubų paviršių, aprašytą tinkleliu, smulkiname, gaudami naujus taškus. Priklausomai nuo naudojamos padalinimo schemos, kiekvienam gautam naujam taškui priskiriame sumą kaimyninių taškų su svoriais. Todėl, apdorojant modelius tokiu tinkleliu, skaičiavimai yra greiti bei jo sudarymo laikas yra trumpas.

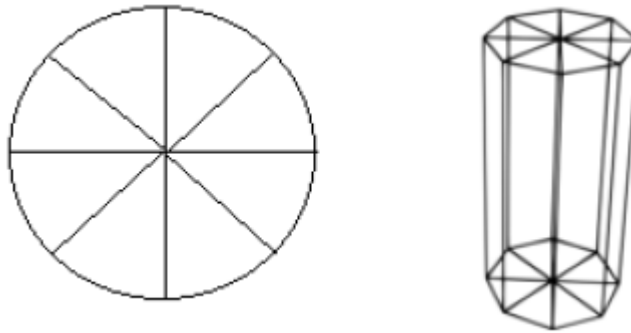
4.3. Projekcinis tinklelis

Projekcinį tinklelį galima generuoti norint 3D vaizdą perteikti į 2D. Viena iš pavyzdžių būtų piramidė, apibrėžta kaip trikampis, su nuorodomis į trijų dimensijų vaizdą. Kitaip tariant, taškas vedą į

eilę sektorių, o sektorius į sugeneruotą tinklėlio grupę. Naudojant šį metodą jo tikslumas tiesiogiai priklauso nuo vartotojo sugebėjimų pritaikyti jam rekomenduojamą tinklėlį.

Dirbant su 3D modeliu ir naudojant 2D tinklėlį, vartotojo tikslas yra kuo tiksliau atkartoti modelį, kurio pagalba būtų galima sugeneruoti baigtinį 3D tinklėlį. Sluoksnių skaičius ir jų pozicijos (pvz. mazgų koordinatės išilgai rekomendacinės linijos) gali būti perduodamos netiesiogiai (kai duotas linijos diskretizavimas) arba tiesiogiai (naudojantis besitempiančia funkcija). Tada rekomenduojamas tinklėlis sugeneruojamas palei duotą kryptį ir sukuria norimą skaičių trijų dimensijų elementų sluoksnių tarp apatinės ir viršutinės ribos. Remiantis dviejų dimensijų elementais (trikampis, keturkampis) sugeneruojami trijų dimensijų elementai (tetraedrai, prizmės) (Y., 2009) (Pascal Frey, Paul Louis George, 2000)

Pagrindinis šio metodo trūkumas yra tas, kad galimi išsigimę elementai, pavyzdžiui atvejuose, kai srities riba sutampa su ašies riba.



22 pav. Dimensijų projekcija

Šiame pavyzdyje parodytas projekcinio tinklėlio generavimo pavyzdys. Dėl paprastumo vaizduojamas tik vienas generavimo sluoksnis.

4.4. Plintančiojo fronto metodas

Plintančio fronto daugiaviršūnio tinklėlio algoritmas yra vienas galingiausių „auginimo“ metodų. Šis metodas yra paremtas paviršiaus auginimu judinat jo ribines kraštinės kreives, kol visa objekto geometrija ir topologija yra apdirbama.

Klasikiniai plintančio fronto požiūriai prasideda nuo daugiaviršūnio tinklėlio srities ribų diskreditacijos į kampų rinkinį 2D ar trikampių elementų rinkinį 3D. Kaip galima suprasti iš metodo pavadinimo, šis metodas susideda iš tinklėlio kūrimo paeiliui – elementas po elemento. Kai yra sukuriamas nors vienas elementas, jam yra sukuriami nauji taškai, kurie yra sujungiami su prieš tai jau sukurtais elementais, tai yra pereinama visą dar nesudarytą daugiaviršūnio tinklėlio erdvę, tikrinant visą tuščios erdvės frontą. (Medeiros, Esdras, Velho, Luiz, Lopes, Hélio, 2003). Šis procesas tęsiasi iki tol,

kol frontas palieka tuščias – tai yra kai erdvė yra visiškai suskaidyta į daugiaviršūnius tinklelius. Frontas sugeba atskirti jau sudarytus daugiaviršūnius tinklelius ir vis dar tuščias erdves, tuo metu trianguliuojamo objekto erdvėje. Frontas gali turėti kelis regionus, kuriuose esantys elementai yra sujungiami (Pascal Frey, Paul Louis George, 2000).

Viena iš pagrindinių plintančio fronto ypatybių yra vidinių taškų kriterijus. Naudojantis šiuo metodu turėtume gauti tokį rezultatą, kad gauti elementai turėtų tenkinti dydžio ir formos kriterijus. Toks dydžio kriterijus ir atitinkamai elementų dydžio paskirstymo funkcija iš anksto apibūdina norimą elementų dydį ir formą.

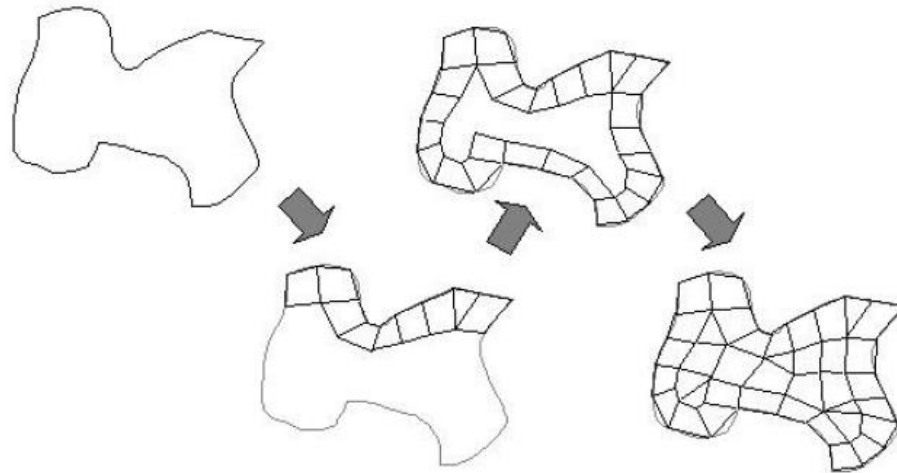
Šių metodų pagrindinis privalumas yra jų euristinė tinklelių generavimo strategija, kuri leidžia sugeneruoti aukštos kokybės ir tinkamai surūšiuotus tinklelius. Dar vienas šio metodo pliusas yra tai, kad ne kaip kiti automatiniai metodai, šis metodas išsaugo kraštinių vientisumą, kadangi figūros kraštinių diskreditacija yra jos pradinis frontas (Pascal Frey, Paul Louis George, 2000).

Plintančio fronto metodą galima suskirstyti į kelis žingsnius. Kiekviename žingsnyje yra pasirenkama detalė ir sukuriamą naują optimali ašis, kuri įterpiama į daugiaviršūnį tinklelį, jei ji suformuoja geros formos tinklelį. Plintančio fronto žingsnius galima apibendrinti šiais žingsniais (Pascal Frey, Paul Louis George, 2000):

1. Žingsnis: Parengtinis apibrėžimas
 - Daugiaviršūnio tinklelio figūros ribos duomenų nuskaitymas (pvz. ribos tinklelis T)
 - Elementų dydžio paskirstymo funkcijos apibūdinimas (reglamentuotas tinklelio generavimas) arba geriausias tokios funkcijos įmanomas sprendimas.
2. Žingsnis: Fronto F inicializavimas su T
3. Žingsnis: Fronto F analizė (tol kol F nėra tuščias):
 - Pasirinkti fronto elementą f (pagrįstą apibūdinimo kriterijumi)
 - Rasti geriausią taškų poziciją P_{opt} šiam elementui
 - Rasti, ar taškas P egzistuoja dabartiniame tinklelyje, kuris turėtų būti panaudotas pirmiau P_{opt} . Jei toks taškas egzistuoja, P laikome P_{opt}
 - Suformuojame elementą K su f ir P_{opt}
 - Patikriname, ar elementas K kertasi su kuriuo nors tinklelio elementu. Jei kertasi, pasirenkam naują tašką P ir grįžtame į 3 d punktą
4. Žingsnis: Atnaujiname frontą ir esamą tinklelį:
 - Pašaliname elementą f iš fronto F . O F elementai naudojami iš K

- Pridedame elementus iš naujai sukurto fronto K
 - Atnaujiname esamą tinklą T
5. Žingsnis: Jei frontas F nėra tuščias grįžtame į 3 žingsnį
 6. Žingsnis: Tinklo optimizavimas

Šiame paveiksle galima pamatyti kaip formuojamas tinklas remiantis žingsniais:



23 pav. Plintančio fronto tinklo generavimas

Kaip matome iš 24 paveiksliuko, tinklas pradedamas formuoti nuo kraštų, kol yra užpildoma visa forma.

5. Atvaizduojamų formų formatų klasifikacija

Informacija ir žinios apie inžinerinę proceso dalį yra kaupiamos jos vizualiaame atvaizdavime. Naudojant tradicinius dizaino modelius, apibrėžiant inžinerinę informaciją būtų naudojami brėžiniai bei prie jų esanti papildoma informacija, tačiau šių dienų technologijos leidžia CAM/CAD informaciją saugoti elektronine forma. Šiuolaikinė kompiuterinė technologija skiriasi nuo seniau populiarios 2D formato sistemos. Dabartinės sistemos informacija leidžia užrašyti į kietus kūnus, tačiau šios sistemos gali būti atvaizduojamos įvairiais formatais. Taip apsunkinant bendradarbiavimą tarp skirtingų sistemų. Bendras duomenų formatas leistų pagerinti skirtingų aplinkų bendradarbiavimą ir užtikrintų procesų plėtrą.

Dabar rinkoje esančias formas pateikiančius formatus galima suskirstyti į tris pagrindines kategorijas: „Native CAD“, „Neutral“, „Lightweight“ (V. Quintana, 2012).

5.1.,„Native CAD“

Dauguma programinės įrangos kūrėjų dažniausiai sukuria savo patentuotus failų formatus, kuriuos atidaryti ir naudoti galima tik jų kurta programine įranga. Tokio tipo formatas yra vadinamas „Native CAD“. Tokį formatą paprastai apibūdina nuosavybės teisės, tai yra todėl, kad įrangos kūrėjas nori apsaugoti savo kurtą produktą nuo dubliavimo. Nors dalis informacijos yra pasiekama, tačiau dažniausiai nėra tikslios formato specifikacijos bei dokumentacijos, kuri leistų tinkamai paruošti formatą kitoms kompiuterinės grafikos sistemoms. Dėl šios priežasties yra matomas noras pabėgti nuo „Native CAD“ formato, tačiau nors šis formatas ir turi šiuos trūkumus, didelė dalis jo vis dar yra naudojama dabartinėje inžinerijoje. Pagrindiniai „Native CAD“ formato trūkumai yra šie:

- Programinės įrangos teisių reglamentai
- Programinės įrangos senėjimas
- Dideli failų dydžiai.
- Riboti gebėjimai palaikyti vizualizavimo ir manipuliavimo reikalavimus tolesniems procesams ir vartotojams.

Nepaisant minėtų trūkumų, vienas iš pagrindinių „Native CAD“ privalumų yra gebėjimas išsaugoti konkrečius inžinerinių duomenų aspektus, išlaikant išsamią objekto informaciją vėlesniam naudojimui.

5.2.,„Neutral“

Tai formatas, kuris leidžia skirtingoms CAD programoms apsieiti informacija bei išlaikyti modelio vientisumą. Svarbus „Neutral“ formatų sėkmės veiksnys yra tikslumas, informatyvumas, komunikacija ir visų dalyvių bendradarbiavimas. Įmonių pagrindinis tikslas yra surasti efektyvius metodus, leisiančius kontroliuoti, platinti ir išlaikyti formos aprašymą per visą modelio gyvenimo ciklą. Todėl „Neutral“ formatas yra vienas iš formatų, padedančių išspręsti šiuos iššūkius. „Neutral“ formatas yra paremtas tarptautiniais standartais, todėl gali perduoti dideles geometrines reprezentacijas.

- Privalumai: gebėjimas išlaikyti aiškią geometriją, palaikyti tolesnį 3D modelių suderinamumą.
- Trūkumai: mažas apsaugos lygis, ilgas laikas prisitaikyti prie naujų savybių CAD programinėje įrangoje, didelis failų dydis.

Pagrindiniai šio formato atstovai yra: „*Standard for the Exchange of Product model data*“ (STEP) ir „*Initial Graphics Exchange Specification*“ (IGES), VDAFS, STL, VRML.

5.2.1. „Initial Graphics Exchange Specification“ (IGES)

IGES yra „Neutral“ formatas, apibūdinantis produkto duomenų atvaizdavimą naudojant kietakūnį (angl. *Solid*) modeliavimą kartu su B - reprezentacijos struktūra. Šis formatas gali būti pritaikytas dideliame sričių spektrui, tarp kurių yra elektra, elektrinės ir mechanikos inžinerija. IGES pateikia formatą, kuriuo naudojantis galima perkelti duomenis tarp skirtingų CAD sistemų. Norint atlikti mainus tarp skirtingų sistemų, IGES turi atlikti dviejų lygių informacijos apdorojimą (Bhandarkar Mangesh, 1999).

- Išankstinių procesorių, kurie priima kai kuriuos CAD duomenis, kaip aprašyta sistemoje, ir konvertuoja jį į IGES formatą;
- Po procesoriaus, kuris konvertuoja duomenis iš IGES atgal į kai kuriuos CAD sistemos formatus;

IGES formate esama duomenų, kurie naudojami apibūdinti, ir reprezentuoti produktą apibrėžiančias savybes bei informaciją. Ši informacija ir savybės yra patalpinamos ir saugomos nepriklausomai nuo aplinkos. Toks informacijos išsaugojimo būdas padidina duomenų mainų galimybes skirtingose CAD sistemose. Iš to galime susidaryti išvadą, kad fundamentiniai informacijos vienetai IGES yra subjektai. Produkto duomenų aprašymas pateikiamas geometriniais, negeometriniais ir topologiniais subjektais. Geometriniai subjektai suteikia duomenis apie objekto fizinę formą, į kurią yra įtraukiami:

- Taškai

- Išlenkimai
- Paviršiai
- Pilnaviduriai kūnai
- Ryšiai, kurie sudaro panašių subjektų rinkinių krūvą.

Negeometriniai objektai apibrėžia objekto peržiūros perspektyvą anotacijomis ir matmenimis. Šiuose objektuose taip pat gali būti įtraukti brėžiniai, tekstai, vaizdai, žymėjimai, konsultantų ir vadovų aprašymai. Topologija apibūdina ryšius tarp primitivių geometrinių objektų, tokiu kaip kraštinės, viršūnės, kilpos ir t.t.

IGES failą sudaro 80 stulpelių eilutės, kurios yra sugrupuotos į 5 ar 6 skyrius. Kiekvienas skyrius turi nuosavą kodą, kuris yra pavaizduotas 1 lentelėje.

1 lentelė. IGES failų skyriai (Kennicott, 2001)

Sekcijos pavadinimas	Stulpelio 73 raidžių kodas
Pradžia	S
Globalus	G
Rodyklės įrašas	D
Parametrizuoti duomenys	P
Nutrukti	T

IGES duomenis gali būti gaunami arba ASCII arba dvejetainiu formatu. Bendra failų struktūra atvaizduojama 2 lentelėje.

2 lentelė. IGES failų struktūra (Kennicott, 2001)

1	8	9	16	17	24	25	32	33	40	41	48	49	56	57	64	65	72	73	80
Pradžios skyrius - tai žmogui suprantamas failo prologas																		S0000001	
Turi daugiau negu vieną eilutę																		S0000002	
:																		S0000003	
:																		...	
1-72 eilutėje naudojami ASCII ženklai																		S000000N	
Globalioji sekcija - siunčiamos sistemos ir failo informacija.																		G0000001	
Čia yra eilė eilučių, reikalingų laikyti parametrinius laukus atskirtus su																		G0000002	
:																		G0000003	
:																		...	
1-72 stulpeliuose aprašomi parametriniai ribotuvai ir nutraukiamas vienu įrašo ribotuvu																		G000000N	

Rodyklės įvedimo skyrius – kiekvienam objektui yra viena pora eilučių.		D0000001
Rodyklės įrašų laukai 1-9 devyniuose 8 stulpelių plotuose		D0000002
Rodyklės įvedimo laukai 10-18 devyniuose 8 stulpelių plotuose		
Parametrų duomenų sekcija - reikšmės ir parametrų ribotuvai, užbaigti vienu įrašų ribotuvu, 1-64 stulpeliuose; 65 skiltis nenaudojama	DE atgalinė rodyklė	P0000001 P0000002
S0000020 G0000003 D0000500 P0000261	Pabaigos skirsnis - įrašų skaičius ankstesnėms sekcijose; 33–72 stulpeliai nenaudojami	T0000001

Kodas, kuris identifikuoja skyrius, yra pateikiama kiekvienoje eilutėje 73 stulpeliuose. Stulpeliuose 74-80 saugoma kiekvienos sekcijos eilutės numeris. Šios sekos numerį sudaro 7 skaičiai, pradedant nuo 1, kurie nuosekliai padidėja po vieną, su pirmąja vieta ar nuli, kuris yra užpildomas pagal eilutės numerį. Stulpeliai nuo 1 – 72 talpina duomenis, būdingus šiam objektui.

Naudojantis lentelėje pateiktais duomenimis, galima susidaryti vaizdą, kaip atrodys galutinis failas. Nors iš pradžių ši informacija ir atrodo padrika, bet giliau patyrinęjus turimą informaciją viskas tampa aiškiau.

5.2.1.1.IGES failų sekcijų gilesnis aptarimas

„Flag“ skyrius

Šis skyrius yra neprivalomas ir naudojamas norint nurodyti, ar failas yra dvejetainis, ar naudojamas suspaustas ASCII formatas.

Pradžios skyrius

Pradžios skyriuje turėtų būti pateikiama žmogui suprantama informacija, į kurią būtų įtraukta inžinieriaus pateikta informacija apie modelio sudarymą bei kiekvienos detalės aprašymas. Šiame skyriuje turėtų būti pateiktas bent vienas įrašas norint, kad programa atpažintų failą kaip tinkama skaityti. Kiekvienas įrašas, esantis šiame skyriuje, yra užvadinas S raide, kuri yra patalpinama 73 stulpelyje ir jos eilės numeris yra stulpelių intervale 74 – 80. 1-73 stulpeliuose nėra jokių specialių formato taisyklių, kad būtų supaprastintas vartotojo duomenų įvedimas. Šiame skyriuje yra įtraukiama tokia informacija kaip pavadinimai, siunčiančios ir priimančios CAD sistemos, taip pat trumpas keičiamo produkto aprašymas.

Globalusis skyrius

Šiame skyriuje yra patalpinami duomenys reikalingi „pre-processor“ ir „post-processor“ operacijų palaikymui. Kiekvienas įrašas, esantis šiame skyriuje, identifikuojamas pagal S raidę 73 stulpelyje ir yra sekoje tarp 74 – 80 stulpelio.

Katalogų įvedimo skyrius

Kataloginio įrašo sudarymo skyriuje siekiama pateikti failo indeksą ir jame išsaugoti kiekvieno objekto atributų informaciją. Įrašų eiliškumas dabartiniame skyriuje yra savavališkas. Kai kuriuose katalogo įrašo laukuose gali būti išsaugoma arba atributo vertė, arba rodyklės, kiekviena tokia objekto vertė gali turėti vieną ar daugiau tokių verčių. Jeigu laukuose pateikiama teigiama reikšmė, šis laukas yra traktuojama kaip atributas, o jeigu vertė yra neigiama, tai parodo, kad absoliuti vertė katalogo įrašė yra rodyklė ir objektas turi vieną ar daugiau atributo reikšmių. Kaip parodyta 3 lentelėje, atributas 1 ir atributas 11 savyje laiko išsaugotus numerius, 2 atributas yra rodyklės įrašas, kuris yra pateikiamas „Parametrų duomenų skyriuje“, kuris yra aprašomas žemiau.

3 lentelė. Katalogų įvedimo skyriaus subjekto požymiai

1 8	9 16	17 24	25 32	33 40	41 48	49 56	57 64	65 72	73 80
(1) Objekto tipo numeris #	(2) Paramet- riniai duome- nys →	(3) Struk- tūra #, →	(4) Linijos šrifto modelis #, →	(5) Lygis #, →	(6) Vaizdas 0, →	(7) Trans- forma- cijos matrica 0, →	(8) Etikečių rodymo Asoc. 0, →	(9) Statuso Nr. #	(10) Sekos Nr. D #
(11) Objekto tipo numeris #	(12) Linijos storio Nr. #	(13) Spalvos Nr. #, →	(14) Para- metro linijų skaičius #	(15) Formos Nr.	(16) (17) Rezervuota		(18) Obj. Etiketė	(19) Obj. Indekso Nr. #	(20) Sekos Nr. D# +1

Terminų sąrašas iš lentelės:

(n) – Lauko numeris n

- Sveikas skaičius

→ - rodyklė

→, # - sveikas skaičius arba rodyklė

0, → - nulis arba rodyklė

4 Lentelėje nurodomi pagrindiniai IGES objektai, kurias naudojantis yra aprašomas kiekvienas objektas sukurtas naudojantis kompiuterinio modeliavimo sistema.

4 lentelė. Pagrindinių IGES failų objektų sąrašas

Objekto numeris	Objekto tipas
110	Linija
100	Apskritimo arka
124	Transformacijos matrica
120	Paviršių apsisukimai
116	Taškai
123	Kryptis
190	Plokščias paviršius
502	Viršūnių sąrašas
404	Kraštų sąrašas
508	Kilpa
510	Paviršius
514	Kiautas
192	Dešinys apskrito cilindro paviršius

Parametrinių duomenų skyrius

Šis failų skyrius saugo parametrinius duomenis, susijusius su kiekvienu objektu. Parametriniai duomenys yra sukuriama laisvu stiliumi, kurio pirmas laukas visada yra objekto tipo numeris. Objekto tipo numeris ir parametrų ribojimas turi po vieną indeksą iš kiekvieno objekto, esančio keičiamajame faile. Laisvai suformuotos parametrinės eilutės baigiasi 64 stulpelyje. 65 stulpelyje yra patalpinta vietos charakteristika. Nuo 64 iki 72 stulpelio visi parametrai turi sekos numerį, kuris yra šio objekto, objekto katalogo pirmos eilutės numeris. 73 stulpelis iš visų eilučių, esančių parametrų duomenyse, turi raide P ir stulpeliai nuo 74 iki 80 turi seką numerių, kurie yra parodyti 5 lentelėje.

5 lentelė. Parametrinių duomenų sekcija (Kennicott, 2001)

1 64	66 72	73 80
Objekto tipo numeris, po kurio seka parametro ribotuvai, po kurio yra parametrai, kurie yra atskirti parametrų ribotuvais	DE rodyklė	P0000001
Parametrai atskirti parametrų ribotuvais, po kurių yra įrašyti ribotuvai.	DE rodyklė	P0000001

Pabaigos skyrius

Šiame skyriuje yra apibrėžiama tikrai viena eilutė, kuri yra failo pabaiga. Čia yra pateikiama tam tikra informacijos suvestinė pvz. kiek eilučių yra kiekviename skyriuje.

5.2.1.2.IGES Trūkumai

IGES yra neutralus formatas, kuris leidžia produkto duomenų apsikeitimą, todėl jis turi aiškų pranašumą prieš integruotus failų vertėjus. Nors jis yra vartojamas ir pritaikomas daugelyje industrijos šakų, IGES turi ir keletą trūkumų. Vienas iš jų yra, kad šis formatas neturi oficialaus duomenų modelio, dėl kurio kyla neaiškumai kai kuriais atvejais. Antra, kai buvo matoma 80 stulpelių faile, ten yra daug informacijos, kurią yra sunku suprasti dėl sudėtingos struktūros. Trečia, jeigu IGES faile atsiranda sintaksės klaida, ją yra sunku surasti ir ištaisyti. Taip pat klaidos gali atsirasti dėl pasikeitimų, kurie yra padaromi faile.

IGES suteikia tik duomenis iš 3D modelių ir brėžinių. Dabartiniuose darbuose interesai yra paremti kaip tik tokio tipo informacija, tačiau proceso planavimo srityje detalės gamybos vaizdų atvaizdavimas turės didesnę paklausą. Dėl šios priežasties skirtingoms inžinerinėms sritims, kurios keisis vienodais IGES duomenimis, bus sudėtinga, kadangi reikalingus duomenis galės išgauti tik individai, kuriems jie bus reikalingi, bet ne visi kompanijoje esantys specialistai. Taip pat IGES nesuteikia duomenų, susijusių su gyvenimo ciklu, todėl ją yra sunku plėtoti toliau dėl trūkstančių integracijos išteklių (P. Bhandarkar Mangesh, 2000)

5.2.2. Gaminio modelio duomenų keitimo standartas (STEP)

Gaminio modelio duomenų keitimo standartas, arba STEP, suteikia galimybę sukurti detalės duomenų aprašymą kartu su mechanizmais, kurie leidžia detalės duomenų mainus. Tokie mainai vyksta tarp skirtingų kompiuterinių sistemų, kuriose yra aprašomas pilnas produkto gyvenimo ciklas: dizainas, produkcija, naudojimas, priežiūra. Duomenys ir detalės, kurie yra gaunami šio proceso metu, yra

naudingi tolimesniuose žingsniuose. Esama įvairių kompiuterinių sistemų terpių, kurios gali būti įtrauktos į šiuos procesus, kurie gali būti netgi ir skirtingose geografinėse plokštumose. Norint, kad duomenys būtų galima naudoti skirtingose kompanijose, informacija turi būti pateikta formoje, kuri turi jau nustatytus standartus, kurie leistų sklandų pasikeitimą tarp skirtingų kompiuterinių aplinkų.

5.2.2.1. STEP formato apžvalga

STEP formatas susideda iš serijos detalių, kurios yra importuojamos ir yra pateikiamos atskirai. Šios detalės patenka į vieną iš išvardintų laukų: apibūdinimo metodas, priskirti resursai, pritaikymo protokolai, abstrakčios bandymų vietos, pritaikymo metodai bei funkcionalumo bandymai.

Pagrindinis STEP formato tikslas yra realizuoti produkto informacijos aprašymo galimybę visuose produkto gyvavimo lygiuose nepriklausomai nuo sistemos. Tačiau, norint pasiekti šiuos užsibrėžtus tikslus, reikia laiko. Labiausiai apčiuopiamas STEP privalumas vartotojams šiandien yra galimybė keistis dizaino duomenimis kaip kietais modeliais ir tvirtų modelių rinkiniais.

STEP aprašymo metodas yra ne tik tinkamas „Neutral“ failų mainų metodui, bet ir norint pasidalinti įgyvendinimo ir produkto duomenų bazėmis ir archyvais. Vienas pagrindinių STEP failo formato tikslų yra pastatyti integruotą produkto informacijos duomenų bazę, kuri yra pasiekama ir naudinga visiems resursams, kurie yra reikalingi norint palaikyti produkto gyvavimo ciklą (P. Bhandarkar Mangesh, 2000).

5.2.2.2. STEP formato panaudojimo aplinka

STEP formato specifikacijoje yra „Application Suites“, kurie kreipiasi į pagrindines aplikacijos sritis. Toliau pateiktuose rinkiniuose, priešingai nei vieninteliuose taikomųjų programų protokoluose, yra naudojami įvairūs taikomųjų programų protokolai. Tokių protokolų pavyzdžiais gali būti laivų statymo rinkiniai, elektromechanikos rinkiniai, elektrinių rinkiniai, sistemų inžinierių rinkiniai bei produkto gyvenimo ciklo priežiūros ir gamybos rinkiniai (SCRA, 2006).

STEP formato duomenų pateikimo galimybės vis didėja, toks didėjimas lemia jo pripažinimą pramonėje. Šis formatas vis labiau pripažįstamas tokiose industrijose kaip automobiliai, gynyba, erdvėlaiviai bei laivininkystė. Kompanijos ir perpardavėjai pradėjo STEP formatą atpažinti kaip priemonę, skirtą apibrėžti ir saugoti informaciją apie gaminį. Taip pat STEP formato populiarumą remia orlaivių bei automobilių pramonė. Paskutiniu metu STEP formatą naudojančių programų kiekiai auga kiekvienais metais (Bhandarkar Mangesh, 1999)

5.3. „Lightweight representations“

„Lightweight representations“, tai detalus modelio formatas, kuriame trūksta tradicinio CAD modeliams būdingo detalumo. Svarbiausios šį formatą atstovaujančios charakteristikos yra mažas failo dydis, kuris yra gaunamas suspaudimo technika, nepriklausomybė nuo aplikacijų ir platformų, atvira prieiga ir delikačios informacijos apsaugos palaikymas. Taip pat 3D anotacijų skaitymas bei atvaizdavimas.

Privalumai: nedidelis failo dydis, kuris sumažina saugyklų bei tinklo reikalavimus, naujos galimybės (kietų paviršių interpretacija, 3D anotacijos ir t.t.), duomenų šifravimo mechanizmai, nepriklausomybė nuo CAD sistemos.

Trūkumai: Performatavimo reikalavimai, reprezentuojamos formos duomenų netekimai.

5.4.„Tessellated“

„Tessellated“, tai modeliavimo formatų grupė, kuri yra skirta 3D spausdinimui, animacijos kūrimui ar žaidimams. Dažniausiai šiai grupei atstovauja STL ir OBJ formatai. Šie formatai atiduoda informaciją išsklotinėmis arba briaunomis ir nesineša su savimi B - reprezentacijos modeliams būdingos CAD informacijos, tokios kaip svoris ar tūris.

5.4.1. STL formato apžvalga

Tarkime, tikslus STL failų rašymo būdas skiriasi tarp sistemų, tačiau perduodami duomenys yra tokie artimi duomenims, kurie taip pat yra naudojami grafikos perteikimo procese, todėl šie gauti duomenys taip pat gali būti naudojami ir STL failų kūrime. (Szirmay-Kalos L, Ma’rton G, Dobos B, Horva’th T, Risztics P, Kova’cs E., 1995)

Grafikai naudojamas pagrindinis trikampių algoritmas susideda iš dviejų pagrindinių etapų: (i) kraštų padalijimo ir (ii) paviršių trianguliacijos. Pirmame žingsnyje visi išlenkti kraštai yra padalinti į pakankamai mažus segmentus, kurie gali būti apytiksliai nusakomi naudojantis tiesiomis linijomis. Paviršiaus trianguliacija susideda iš dviejų pakopų:

1. Apdoroti kiekvieną paviršių, padalijant paviršius ties bet kuriomis įgaubtomis viršūnėmis;
2. Kai lieka tik išgaubti paviršiai, tada juos galima suskaidyti į trikampius sujungiant tinkamas viršūnes.

Pirmas dalykas, į kurį reikia atkreipti dėmesį kalbant apie šio tipo paprastus algoritmus, yra tai, kad artimiausios viršūnės kriterijus nebūtinai veda į reguliarių lygiakraštį trikampį. Įgaubta viršūnė yra sujungta su artimiausia jo viršūne, o tai reiškia, kad forma išgaubto paviršiaus (kuris ilgainiui tampa

trianguluotu) nekontroliuojamas. Nesunkiai lygiašoniai trikampiai gali susiaurėti, užuot turėję taisyklingas lygiakraštes formas, ypač ant pavienių lenktų paviršių, tokių kaip cilindrai ar kūgiai, kurios yra sudarytos iš briaunų su mažu stygos aukščiu. Netvarkingos formos trikampiai gali sukelti geometrinių problemų (tarpų ar atitinkančių viršūnių) apdorojant vėliau.

STL byla parašyta nuskaitant visus briaunotus paviršius, esančius modelyje, ir užrašant briaunos normalias ir viršūnių pozicijas aplink taško padėtį. Bendrasis STL formato failas yra užrašomas taip:

```
solid <name>facet normal <real> <real> <real>
outer loop
vertex <real> <real> <real>
vertex <real> <real> <real>
vertex <real> <real> <real>
endloop
endfacet
facet normal <real> <real> <real>
outer loop
vertex <real> <real> <real>
vertex <real> <real> <real>
vertex <real> <real> <real>
endloop
endfacet
endsolid
```

Tai yra gana paprastas formatas. Nors jis ir turi potencialą apibrėžti paviršius, turinčius daug kraštinių, tačiau standartinis STL gali atpažinti tik trikampes briaunas.

Yra keturi pagrindiniai STL failo formato trūkumai:

1. Formato dubliavimas;
2. Trūksta išsamaus geometrinio aprašymo (neįmanoma tiksliai atkurti formos);
3. Nėra tiksliai apibrėžtas aproksimacijos metodas;
4. Nėra technologinė informacija (tokia kaip paviršiaus apdaila), nes trūksta technologijos šiai informacijai išgauti

Dėl pirmojo trūkumo susidaro dideli STL failai, kurie sulėtina gaminimą. Dėl antrojo trūkumo gali būti nesuderinamumas tarp aspektų ir kyla pavojus, kad gali būti nesuderintos viršūnės ir neteisinga topologija. Dėl trečiojo trūkumo gali būti iškraipomi objektai. Ketvirtasis trūkumas neleidžia greitai integruoti CAD modeliavimo sistemas.

Šis formatas pateikia perteklinius duomenis, nes normalės aspektą galima nustatyti dviem skirtingais būdais. Viršūnės yra išdėstytos prieš laikrodžio rodyklę aplink briauną, tačiau tai lemia briaunos normalią, taigi, briaunos STL formato eilutėje nurodytas normalės yra nereikalingos, nebent briaunos yra labai mažos, tokiu atveju iš viršūnės padėčių apskaičiuota normalė yra netiksli.

Kitas trūkumas yra aspektų nesuderinamumas. Informacijos apie viršūnių padėtis nėra, vadinasi, norint sukurti modelį, reikia ieškoti atitinkamų viršūnių. Tai gali užtrukti daug laiko ir pavojinga, nes

geometrinių taškų padėtis iš esmės yra netiksli, todėl sutapimui patikrinti reikia naudoti nuokrypį. Jei STL faile yra daug mažų aspektų, kyla pavojus, kad viršūnės nesutampa, taigi gali būti sukurta neteisinga topologija.

Aproksimacijos metodas nėra tiksliai apibrėžtas, ir atrodo, kad numatytoji reikšmė yra ta pati kaip grafikos, kuriai būdingi skirtingi poreikiai ir skirtingi reikalavimai, lyginant su greitojo prototipų sudarymo reikalavimais. Tai reiškia, kad apytikslis vaizdas gali būti objekto viduje arba išorėje, atsižvelgiant į tai, ar paviršius yra išgaubtas, ar įgaubtas. Tai sukelia objekto iškraipymą.

Galiausiai, technologinė informacija negali būti pridėta, nes aspekto ir paviršiaus santykis nėra apibrėžtas, todėl neįmanoma apytiksliai nustatyti objekto paviršiaus.

5.5. Apibendrinimas

„Native“ CAD reprezentacija atrodo sudėtinga ir formatai dažniausiai yra nuo ko nors priklausomi. Šio formato specifikacijos yra neprieinamos ir tai prieštarauja efektyviai duomenų mainų idėjai. „Lightweight representations“ trūkumas yra duomenų praradimas, kuris nėra tinkamas sprendimas šiame darbe. Todėl darbe naudosime IGES formatą duomenų išgavimui bei jų grąžinimui. IGES formatas leidžia reikiamą duomenų išgavimą iš „SketchUp“ programos. Gauti duomenys leidžia sklandžius duomenų mainus tarp „SketchUp“ ir „Inventor“ programų.

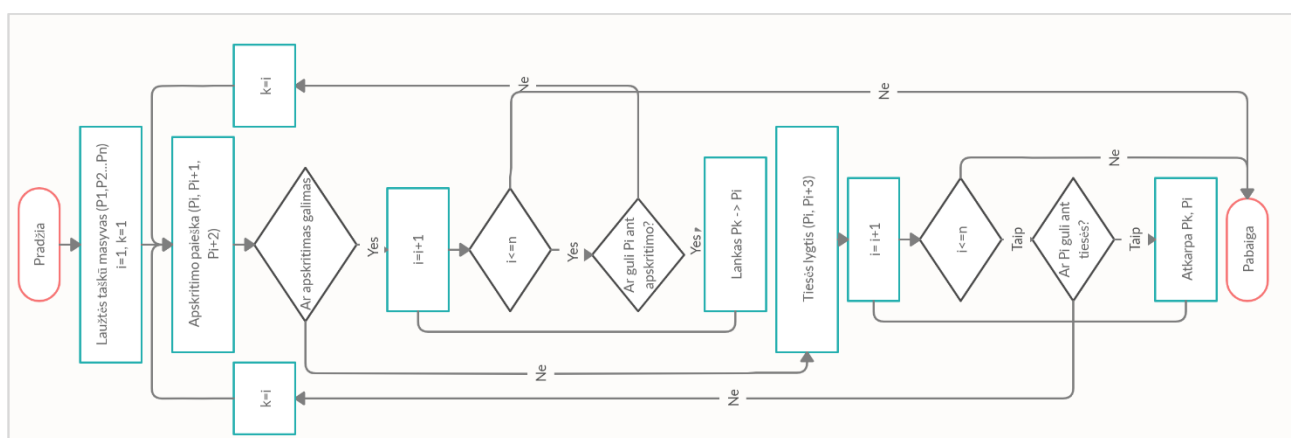
6. Daugiaviršūnių tinklų aproksimacijos pagrindai ir taikymas

Mūsų uždavinys yra konvertuoti daugiaviršūnių tinklų išreikštus modelius į B - reprezentacijos formą. Šiame darbe apsiribosime cilindrinų ir plokščių paviršių konvertavimu. Pirmiausiai išnagrinėsime aproksimacijos uždavinį 2D plokštumoje, kuris yra reikalingas B - reprezentacijos briaunų sudarymui, o principus vėliau pernešime į 3D erdvę plokštumų ir cilindrinų paviršių paieškai.

6.1. 2D Poligoninės kreivės konvertavimas į tiesių ir lankų rinkinį

Pirmiausia aptarkime, kaip laužtės taškų rinkinyje atrasti mums reikiamas formas. Bandymus pradėsime nuo 2D plokštumos ir pereisime į 3D. Norint atrasti tiesę taškų rinkinyje mums užtenka žinoti 2 taškus- pradžios ir pabaigos. Norėdami rasti tiesę naudosime tiesės lygtį (1). Naudodamas šią lygtį atliksiu eksperimentą „AutoCad“ programos pagalba. Naudodamas „Autolisp“ programavimo kalbą sukursiu programą, kuri laužtės taškų rinkinyje išskirstys taškus, į priklausančius tiesės atkarpoms, ir taškus, priklausančius lankiniams segmentams.

Norėdamas išspręsti šį uždavinį, sukūriau tokį veikimo algoritmą:



24 pav. Kreivės konvertavimas į tiesių ir lankų rinkinį algoritmo schema

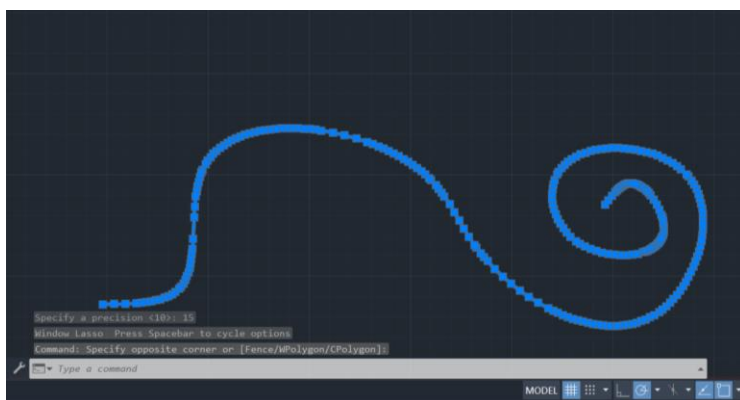
Kaip matome iš schemos (žr. 25 pav.) programa pradeda nuo laužtės taškų masyvo sudarymo. Indeksui i priskiriame reikšmę (i=1), o k indeksas bus lygus i. Tuomet pradeda apkritimo paiešką. Pasirenkame pirmuosius tris taškus P_i , P_{i+1} , P_{i+n} laužtės taškų masyve. Jeigu iš šių taškų gauname apkritimą, tuomet imame $i=i+1$ ir jeigu $i \leq n$ užduodame, ar P_i guli ant apkritimo. Jeigu atsakymas taip, tuomet darome lanką nuo taško P_k iki P_i . Šį algoritmą kartojame, kol yra patenkinamos apkritimo sąlygos. Jeigu iš pasirinktų taškų negalime sudaryti apkritimo, pereiname prie tiesės lygties. Kai patikriname visus taškus ir nebegalime sudaryti nei tiesės, nei apkritimo, viską pradame iš naujo.

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \quad (1)$$

Šioje lygtyje x_1, x_2, x, y, y_1, y_2 yra taškų koordinatės

Surasti apkritimą yra sudėtingiau, kadangi jam aprašyti reikia 3 taškų. Turime laužtės taškų rinkinį (žr. 26 pav.). Kaip matyti iš paveiksluko, tokius taškus greičiausia būtų sujungti linijomis, tačiau šis būdas nėra pats geriausias, nes šį rinkinį sujungus lankais, žymiai sumažėja duomenų kiekis. Norint tai padaryti, reikia paimti tris taškus ir jiems surasti apskritimo parametrus. Žinodami tris apskritimo koordinatas (x_1, y_1) , (x_2, y_2) ir (x_3, y_3) , galime surasti apskritimo lygtį ir tada apskaičiuoti apskritimo centrą ir spindulį. Apskritimo lygtis bendrąja forma yra:

$$x^2 + y^2 + 2gx + 2fy + c = 0 \quad (2)$$



25 pav. Taškų rinkinio pavyzdys

Kai mes jau žinome, kad šie trys taškai yra ant apskritimo, jie patenkins apskritimo lygtį ir, sudėję juos į bendrąją lygtį, gausime tris lygtis su trimis kintamaisiais g, f ir c . Ir toliau sprenddami šias lygtis galime gauti apskritimo centro koordinatas. Naudodami bendrą apskritimo lygtį (2) galime gauti g, f ir c reikšmes.

Pradėsime nuo trijų apskritimo taškų įstatymo į formulę (2):

$$x_1^2 + y_1^2 + 2gx_1 + 2fy_1 + c = 0 \quad (3)$$

$$x_2^2 + y_2^2 + 2gx_2 + 2fy_2 + c = 0 \quad (4)$$

$$x_3^2 + y_3^2 + 2gx_3 + 2fy_3 + c = 0 \quad (5)$$

Iš (4) formulės gauname:

$$2gx_1 = -x_1^2 - y_1^2 - 2fy_1 - c \quad (6)$$

$$c = -x_1^2 - y_1^2 - 2fy_1 - 2gx_1 \quad (7)$$

Iš (6) formulės gauname:

$$2fy_3 = -x_3^2 - y_3^2 - 2gx_3 - c \quad (8)$$

Iš formulės (5) atimame formulę (4) ir gauname:

$$2g(x_1 - x_2) = (x_2^2 - x_1^2) + (y_2^2 - y_1^2) + 2f(y_2 - y_1) \quad (9)$$

Toliau įstatome formulę (8) į formulę (9) ir gauname:

$$2fy_3 = -x_3^2 - y_3^2 - 2gx_3 + x_1^2 + y_1^2 + 2fy_1 + 2gx_1 \quad (10)$$

Toliau statome $2g$ reikšmę iš (10) formulės į (11) ir gausime:

$$2f = \frac{(x_1^2 - x_3^2)(x_1 - x_2) + (y_1^2 - y_3^2)(x_1 - x_2) + (x_2^2 - x_1^2)(x_1 - x_3) + (y_2^2 - y_1^2)(x_1 - x_3)}{(y_3 - y_1)(x_1 - x_2) - (y_2 - y_1)(x_1 - x_3)} \quad (11)$$

Naudodami tą patį metodą galime gauti ir g . Gavę apkritimo centrus ir įstatę juos į formulę (3), gausime apkritimo spindulį.

Atlikę skaičiavimus mes sužinome šio apskritimo centrą bei spindulį. Šių dviejų charakteristikų užtenka, kad galėtumėme tikrinti kitus taškus, ar jie guli ant šio apskritimo. Kiti taškai gali neviseškai atitikti lygties sąlygas, todėl galima įvesti toleranciją ir tikrinti taško priklausymą duotam apskritimui. Žinodami apkritimo spindulį galime tikrinti kitus taškus. Žiūrime kitą tašką ir matuojame atstumą iki jo nuo jau žinomo apkritimo centro. Jeigu atstumas atitinka jau žinomą spindulį, vadinasi šis taškas vis dar patenka į mūsų apskritimą. Norėdami užtikrinti, kad turėtume kuo mažiau duomenų, mes šiems taškams pritaikome ir toleranciją.

Žinodami apkritimo lygtis, galime parašyti programą, kuri leis nustatyti, ar taškai priklauso tiesei ar apskritimui. Naudodamasis „Autolisp“ programavimo kalba, parašiau programą apskritimui ir tiesei nustatyti (žr. 27 pav.). Programos pradžioje apsirašome visus kintamuosius, kurių mums reikės norint nubrėžti apkritimą per 3 taškus. Toliau randame kiekvieno taško X ir Y koordinates. Susirašome kitus kintamuosius, kurie yra formulėje. Kode esantis kintamasis naudojamas ne tik apkritimo formulėje, bet ir pasako, ar taškas priklauso tiesei ar apskritimui. Tolesnėje kodo dalyje apskaičiuojame centro koordinates, kurios padės mums sujunti taškus, esančius taškų debesyje.

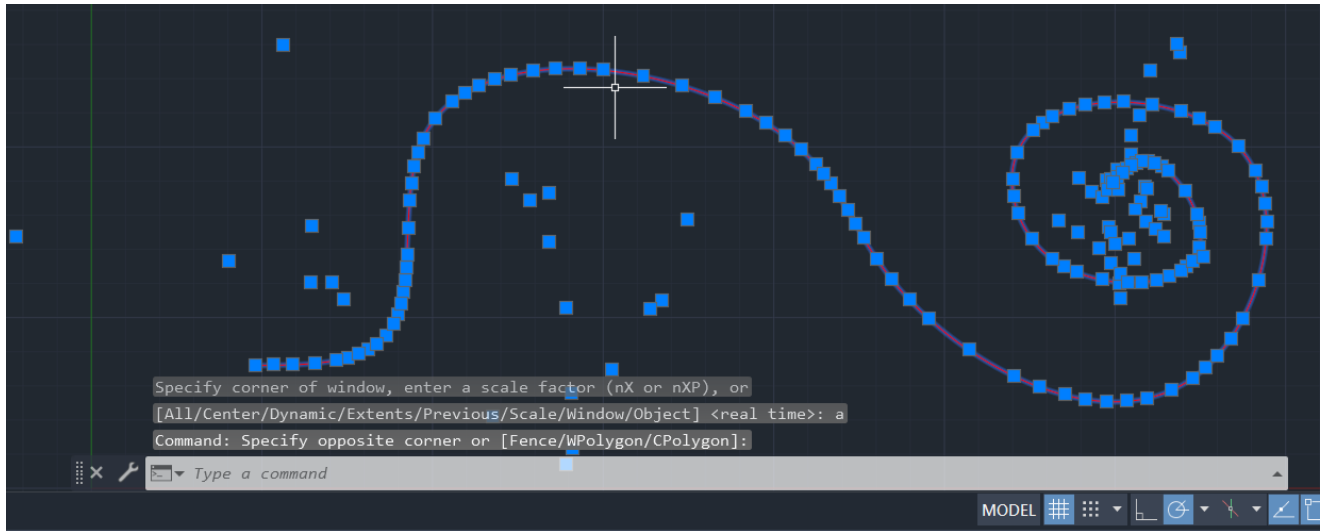
```

1 (defun ARC_ThroughThreePoint (Pt1 Pt2 Pt3 / X1 X2 X3 Y1 Y2 Y3 A B C D E F G X0 Y0 CenterPoint 3DStartPoint RadiusVector R)
2   (setq X1 (car Pt1)
3     X2 (car Pt2)
4     X3 (car Pt3)
5     Y1 (cadr Pt1)
6     Y2 (cadr Pt2)
7     Y3 (cadr Pt3)
8     A (- X2 X1)
9     B (- Y2 Y1)
10    C (- X3 X1)
11    D (- Y3 Y1)
12    E (+ (* A (+ X1 X2)) (* B (+ Y1 Y2)))
13    F (+ (* C (+ X1 X3)) (* D (+ Y1 Y3)))
14    G (* 2.0 (- (* A (- Y3 Y2)) (* B (- X3 X2)))) ;jeigu G=0 tada tiese
15  )
16  (if (= G 0.0)
17    (list Pt1 Pt3 -1.0); gauname tiese
18    (progn
19      (setq X0 (/ (- (* D E) (* B F)) G)
20        Y0 (/ (- (* A F) (* C E)) G)
21        CenterPoint (list X0 Y0 0.0)
22        3DStartPoint (append Pt1 '(0.0))
23        RadiusVector (SubtractVector CenterPoint 3DStartPoint)
24        R (VLength RadiusVector)
25      )
26    )
27    (Print G)
28    (if (> R 20000.0) (setq R (- R))); jeigu numanomas maksimalus radiusas virsija tam tikra riba pereinama i tiese
29    (list X0 Y0 R)
30  )
31 )
32 )

```

26 pav. Programa apskritimui per 3 taškus rasti

Pritaikęs savo parašytą programą taškų kolekcijai (žr. 26 pav.) gauname (žr. 28 pav.).

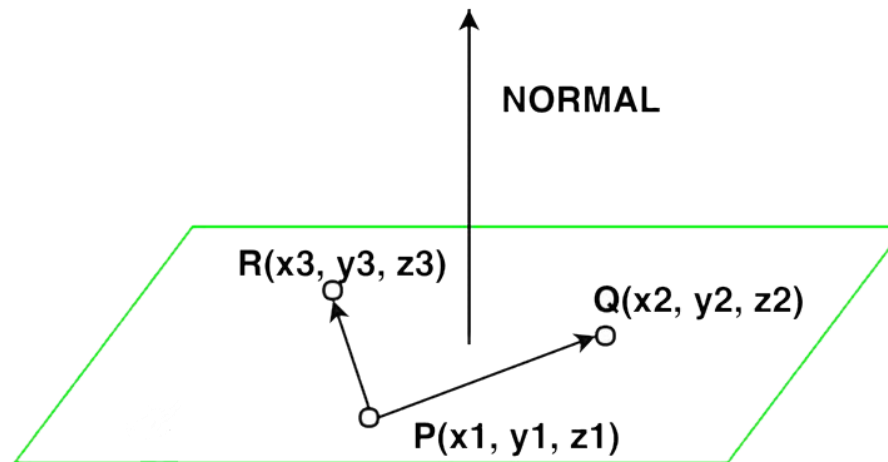


27 pav. Informacijos rinkinys po programos pritaikymo

Iš paveiksluko matome kaip sumažėjo duomenų kiekis palyginus su tuo ką turėjome pradžioje.

6.2. 3D Daugiaviršūnių tinklų konvertavimas į plokštumą

Minimalus taškų rinkinys 3D formų nustatymui yra gana lengvai suprantamas, 2 taškai nurodo liniją, 3 taškai nurodo plokštumą. Todėl 3D erdves ir pradėsime nuo plokštumos. Matematikoje plokštuma yra plokščias dvimatis paviršius, besitęsiantis į begalybę. Kaip minėtą prieš tai, surasti plokštumai reikia 3 taškų. Trys taškai (P,Q,R) gali apibūdinti du skirtingus vektorius PQ ir PR. Kadangi abu vektoriai yra plokštumoje, jų kryžminę sandaugą galima naudoti kaip plokštumos normalę (žr. 29 pav.). Norint rasti plokštumą, mums pirmiausia reikia apsibrėžti kintamuosius. Pirmiausia mums reikia apskaičiuoti vektorius, tuomet rasti kryžminę dviejų vektorių sandaugą.



28 pav. Plokštuma per 3 taškus

Norėdamas rasti plokštumos formulę, naudosisiu šią lygtį:

$$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0 \quad (12)$$

Šioje lygtyje kintamieji (x_0, y_0, z_0) vieno iš trijų taškų plokštumoje koordinatės. (a, b, c) yra vektoriaus koordinatės, statmenos plokštumai. Kaip matome iš paveiksliuko (žr. 30 pav.), mes kol kas vektorių neturime, bet galime gauti juos pasinaudoję dviem iš trijų taškų. Mūsų pirmo vektoriaus $\overrightarrow{PR}$ lygtis atrodo šitaip:

$$\overrightarrow{PR} = \langle (x_1 - x_3), (y_1 - y_3), (z_1 - z_3) \rangle \quad (13)$$

Antrojo vektoriaus $\overrightarrow{PQ}$ lygtis:

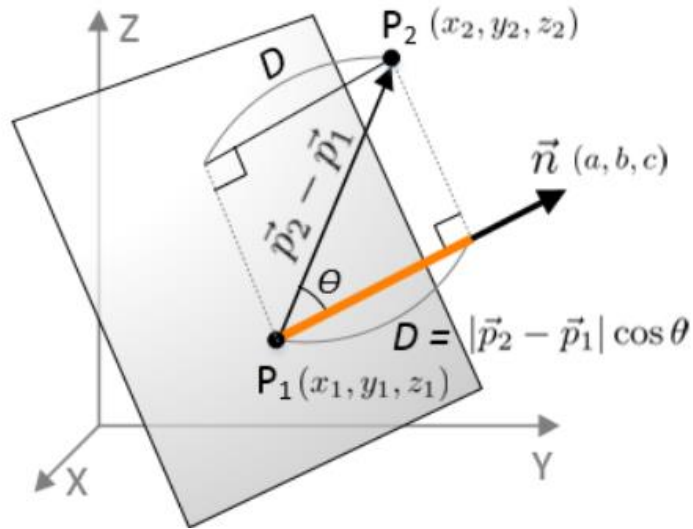
$$\overrightarrow{PQ} = \langle (x_1 - x_2), (y_1 - y_2), (z_1 - z_2) \rangle \quad (14)$$

Norėdami surasti $\langle a, b, c \rangle$ koordinatės, turime surasti kryžminę vektorių $\overrightarrow{PQ}$ ir $\overrightarrow{PR}$ sandaugą.

$$\begin{aligned} \overrightarrow{PQ} \times \overrightarrow{PR} &= \begin{pmatrix} i & j & k \\ (x_1 - x_3) & (y_1 - y_3) & (z_1 - z_3) \\ (x_1 - x_2) & (y_1 - y_2) & (z_1 - z_2) \end{pmatrix} \\ &= \begin{vmatrix} (y_1 - y_3) & (z_1 - z_3) \\ (y_1 - y_2) & (z_1 - z_2) \end{vmatrix} \vec{i} - \begin{vmatrix} (x_1 - x_3) & (z_1 - z_3) \\ (x_1 - x_2) & (z_1 - z_2) \end{vmatrix} \vec{j} + \begin{vmatrix} (x_1 - x_3) & (y_1 - y_3) \\ (x_1 - x_2) & (y_1 - y_2) \end{vmatrix} \vec{k} \\ &= \vec{i}((y_1 - y_3)(z_1 - z_2) - (y_1 - y_2)(z_1 - z_3)) \\ &\quad - \vec{j}((x_1 - x_3)(z_1 - z_2) - (x_1 - x_2)(z_1 - z_3)) \\ &\quad + \vec{k}((x_1 - x_3)(y_1 - y_2) - (x_1 - x_2)(y_1 - y_3)) \end{aligned} \quad (15)$$

Išskaičiavę vektorius $(\vec{i}, \vec{j}, \vec{k})$, randame (a, b, c) koordinatės, jas įstatę į plokštumos lygtį (12) kartu su vienu iš pasirinktų taškų gauname plokštumos lygtį per užduotus tris taškus.

Kadangi jau žinome plokštumos lygtį, galime patikrinti ir kitus taškus, kurie yra erdvėje, ar jie priklauso mūsų plokštumai. Tai darysime matuodami, kiek yra nutolęs 4 taškas plokštumoje. Trumpiausias atstumas nuo erdvėje esančio taško P_2 iki plokštumos gali būti apskaičiuotas pagal dviejų vektorių n ir $(\vec{p}_2 - \vec{p}_1)$ taškinę sandaugą. Tai daroma projektuojant vektorių $(\vec{p}_2 - \vec{p}_1)$ į normalės vektorių n , esantį plokštumoje (žr. 30 pav.).



29 pav. Atstumas tarp taško ir plokštumos

Atstumas D tarp plokštumos $ax + by + cz + d = 0$ ir taško P_2 iš to mes gauname formulę:

$$D = |\vec{p}_2 - \vec{p}_1| \cos \theta = \frac{(\vec{p}_2 - \vec{p}_1) \cdot \vec{n}}{|\vec{n}|} = \frac{(\vec{p}_2 - \vec{p}_1) \cdot \vec{n}}{\sqrt{a^2 + b^2 + c^2}} \quad (16)$$

Aukščiau pateiktos lygties $(\vec{p}_2 - \vec{p}_1) \cdot \vec{n}$ skaitiklio dalis išplečiama:

$$\begin{aligned} (\vec{p}_2 - \vec{p}_1) \cdot \vec{n} &= (a, b, c)((x_2 - x_1), (y_2 - y_1), (z_2 - z_1)) \\ &= ax_2 + by_2 + cz_2 - (ax_1 + by_1 + cz_1) \end{aligned} \quad (17)$$

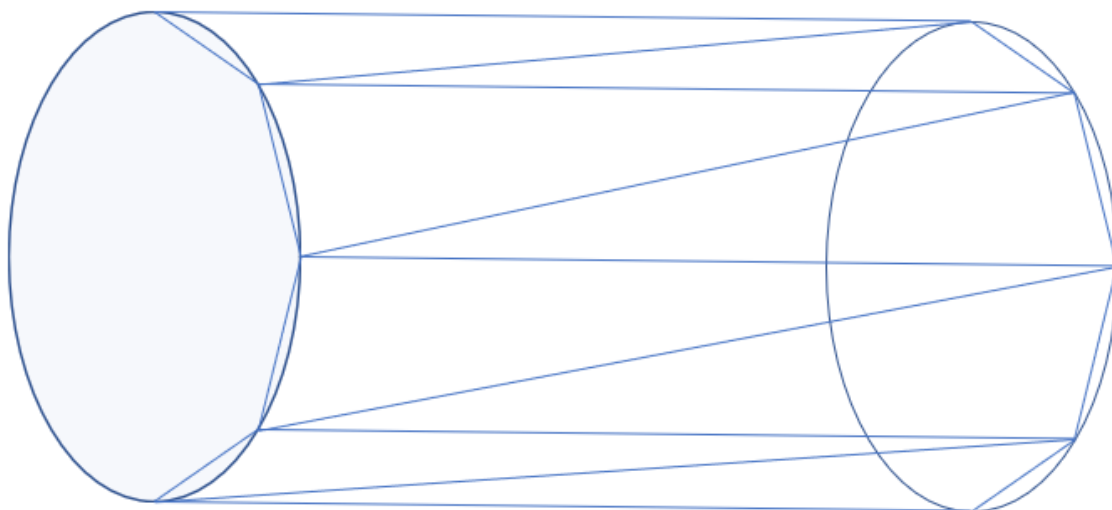
Galiausiai mes pritaikome ją ankstesnei lygčiai, kad užpildytume atstumo formulę:

$$D = \frac{ax_2 + by_2 + cz_2 - (ax_1 + by_1 + cz_1)}{\sqrt{a^2 + b^2 + c^2}} \quad (18)$$

Pritaikę šią formulę ir jai priskyre toleranciją, galime surasti visus taškus, kurie priklauso plokštumai.

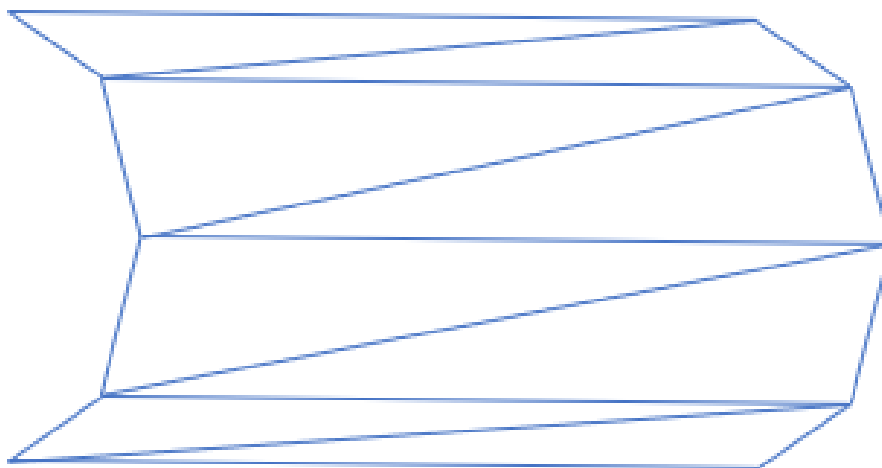
6.3.3D Daugiaviršūnių tinklų konvertavimas į cilindrą

Turime cilindrą, kuris yra išreikštas daugiaviršūniu tinklu (žr. 31 pav.). Kaip matome, cilindras yra sudarytas iš trikampių segmentų, kurie turi savo taškus, kurių mums reikia norint surasti cilindrą.



30 pav. Cilindras išreikštas daugiaviršūniu tinklu

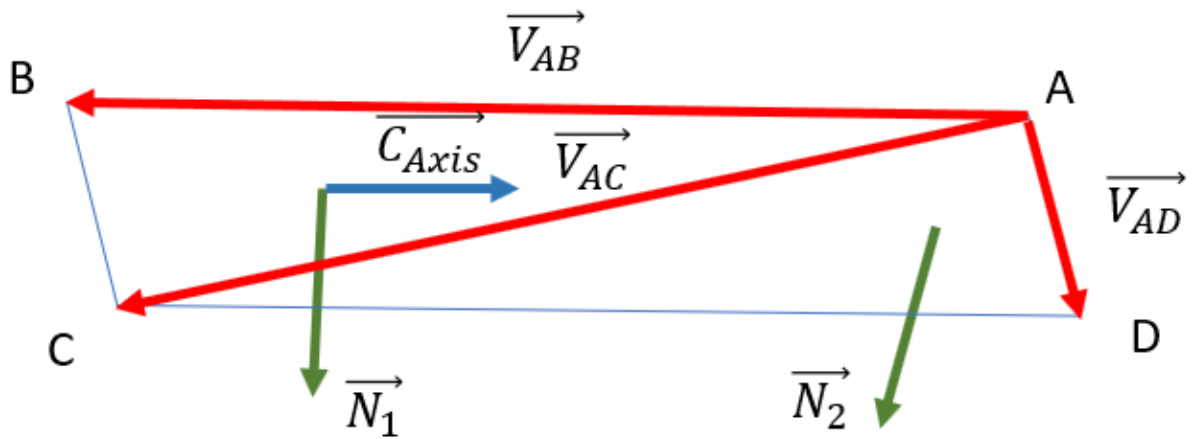
Kai pašaliname cilindą, gauname segmentus, kurie ir sudaro cilindą (žr. 32 pav.)



31 pav. Segmentai, sudarantys cilindą

Iš šios daugybės segmentų mes pasirenkame 2, kurie yra šalia vienas kito. Šiems segmentas prigiriame taškus A, B, C, ir D (žr. 33 pav.), kurie išreiškia turimą tinklą. Kadangi mes turime taškų koordinates, mes galime apsiskaičiuoti vektorius $\vec{V}_{AB}, \vec{V}_{AC}, \vec{V}_{AD}$.

$$\vec{V}_{AB} = < (x_1 - x_2), (y_1 - y_2), (z_1 - z_2) > \quad (19)$$



32 pav. Pasirinkti segmentai

Kai jau turime suskaičiuotus vektorius, galime surasti normalės vektorius plokštumoms ABC ir ACD. Norint gauti šias normales mus reikia sudauginti vektorių $\vec{V}_{AB}$ iš vektorių $\vec{V}_{AC}$. Sandaugai atlikti naudosime formulę (15).

$$\vec{N}_1 = \vec{V}_{AB} \times \vec{V}_{AC} \quad (20)$$

Kai jau turime abiejų plokštumų vektorius, juos sudauginę gauname cilindro ašies vektorių kryptį. Norint patikrinti, ar šie vektoriai tikrai priklauso cilindriui, mums reikia palyginti šiuos vektorius. Jei vektoriai yra lygiagretūs, vadinasi šie vektoriai nepriklauso cilindriui, o yra vienoje plokštumoje. Norint patikrinti, ar vektoriai yra lygiagretūs, turime juos normalizuoti, tai yra privesti šių vektorių ilgį prie 1.

$$\vec{C}_{AXIS} = \vec{N}_1 \times \vec{N}_2 \quad (21)$$

Vektoriaus normalizaciją pradėsime nuo vektorių ilgio skaičiavimo:

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (22)$$

Kai jau turime vektorių ilgį, galime jį normalizuoti. $\vec{N}_1$ yra vektorių x, y ir z koordinatės, o $|\vec{a}|$ yra vektorių ilgis.

$$u = \frac{\vec{N}_1}{|\vec{a}|} \quad (23)$$

Kadangi mes jau žinome, kad $\vec{C}_{AXIS}$ priklauso cilindriui, mes galime surasti apkritimą, kuris priklauso šiam cilindriui. Norėdami tai padaryti, pastatome plokštumą, kuri eina per vieną iš tinklo kraštinių vidurio taškų. Norėdami atrasti šios kraštinės vidurio tašką, pasinaudosime šia formule:

$$M = A + \vec{V}_{AB} * 0,5 \quad (24)$$

Čia A yra taško koordinatė, esanti ant tinklo. Kai jau turime taško koordinatę ant tinklo ir žinome cilindro vektorių kryptį, galime per šiuos taškus nubrėžti plokštumą. Norėdami nubrėžti plokštumą

pirmiausia turime rasti plokštumos lygtį. Pirmiausia mums reikia rasti vektorių, kuris jungia M tašką ir N tašką. N yra taškas plokštumoje. Norėdami rasti šį vektorių, naudosime šią formulę:

$$\vec{p} - \vec{p}_1 = (x - x_1), (y - y_1), (z - z_1) \quad (25)$$

Kadangi vektoriaus $\vec{p} - \vec{p}_1$ ir normalės vektoriaus normalusis $\vec{C}_{AXIS}$ yra statmeni vienas kitam, dviejų vektorių sandauga turėtų būti lygi 0.

$$\vec{C}_{AXIS} \cdot (\vec{p} - \vec{p}_1) = 0 \quad (26)$$

Šis normalės vektorius ir vektoriaus sandauga plokštumoje tampa plokštumos lygtimi. Apskaičiavę sandaugą, gauname:

$$(a, b, c) \cdot (x - x_1), (y - y_1), (z - z_1) = 0 \quad (27)$$

$$a(x - x_1) + b(y - y_1) + c(z - z_1) = 0 \quad (28)$$

$$ax + by + cz - (ax_1 + by_1 + cz_1) = 0 \quad (29)$$

Norėdami supaprastinti plokštumos lygtį (29), galime pastovias reikšmes pakeiti į $d = -(ax_1 + by_1 + cz_1)$ ir taip gauti supaprastintą plokštumos lygtį $ax + by + cz + d = 0$.

Turėdami plokštumos formulę, mes galime nubrėžti plokštumą. Kai jau turime plokštumą, galime surasti dar 2 taškus, kurie guli ant jos. Sukirsdami atkarpas AC ir CD su plokštuma, gauname dar 2 taškus N ir O. Kadangi plokštumoje mes turime tris taškus M, N ir O, jų pagalba, taip pat naudojantis algoritmu, esančiu paveikslėlyje, mes galime surasti apkritimą (žr. 26 pav.). Kai randame apkritimą, jo dėka mes turime cilindro centrą bei jo spindulį. Kai jau susitvarkome su šiais dviem segmentais, prie jų prijungiame kitus segmentus ir plokštumoje atvaizduojame kitus taškus iš cilindro tinklelio. Ir taip tikriname, ar šie taškai vis dar priklauso mūsų apkritimui.

Išvados ir rekomendacijos

1. Buvo apžvelgtos ir išanalizuotos dvi skirtingos CAD modelių užrašymo struktūros, ribojančiosios geometrijos ir daugiaviršūnio tinklo modelių užrašymo struktūros.
2. Darbo realizavimui buvo pasirinktas IGES formatas. To priežastimi tapo šio formato viešai publikuojamas aprašymas ir tai, kad jis yra tipinis B-reprezentacijos formato atstovas.
3. CAD sistema, generuojančia daugiaviršūnio tinklo modelius, pasirinkta „SketchUp“ programa. Tai nulėmė jos populiarumas, prieinamumas (yra nemokama versija). Ji patogi ir gerai dokumentuota platforma.
4. Darbo metu buvo sukurti algoritmai, kaip iš daugiaviršūnio tinklo lopinio informacijos rekonstruoti B-reprezentacijos lopinio kraštines į tiesėmis ir lankais išreikštas briaunas, patį lopinio paviršių atpažinti kaip plokštumos arba cilindrinio paviršiaus fragmentą.
5. Šie algoritmai buvo pritaikyti programos prototipe. Jie sugeba „SketchUp“ programoje sukurtus paprastus modelius eksportuoti į IGES formatą. Vėliau toks failas gali būti importuojamas į bet kurią B-reprezentaciją vartojančią programą, kuri turi IGES importo galimybę. Bandymams buvo pasirinkta „Autodesk Inventor“ programa.
6. Prototipo trūkumai ir galimos vystymo kryptys yra šios: kol kas galima importuoti tik paprastos formos kūnus, kurių kraštines sudaro plokščios 2D figūros., apsiribojama tik plokščių ir cilindrinio paviršių atpažinimu. Ateityje reiktų įtraukti sferos, kūgio, toro formos paviršius ir sugebėti rekonstruoti netaisyklingos formos NURBS paviršius. Reiktų praplėsti kraštinių formų užrašymą 3D kreivėmis.



Literatūros sąrašas

1. AutoDesk. (2020 m. Sausio 5 d.). Nuskaityta iš Autodesk: <https://www.autodesk.com/products/eagle/blog/complete-history-failed-ecad-mcad-exchange-file-formats/>
2. Bhandarkar Mangesh, R. N. (1999). *STEP product information models in Agile manufacturing*.
3. Blazek, J. (2015 m. 01 05 d.). *Computational Fluid Dynamics: Principles and Applications*. Butterworth-Heinemann. Nuskaityta iš wikipedija: http://en.wikipedia.org/wiki/Delaunay_triangulation
4. Chakravorty, D. (2020 m. January 16 d.). *all3dp*. Nuskaityta iš all3dp: <https://all3dp.com/what-is-stl-file-format-extension-3d-printing/>
5. Kennicott, P. R. (2001). *Initial Graphics Exchange Specification IGES 6.0*.
6. Kutz, M. (2002). *Handbook of Materials Selection*. CRC Press.
7. Mark Meyer, Mathieu Desbrun, Peter Schroder, and Alan H. Barr. (2003). *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds*.
8. Matoušek, J. (2002). *Lectures in Discrete Geometry*.
9. Medeiros, Esdras, Velho, Luiz, Lopes, Hélio. (2003). A Topological Framework for Advancing Front Triangulation. *16th Brazilian Symposium on Computer Graphics and Image Processing*, (p. 45-51). Brazil.
10. Merriam-Webster, Incorporated. (1828). Paimta 2019 m. 04 25 d. iš Merriam-Webster: <https://www.merriam-webster.com/>
11. P. Bhandarkar Mangesh, D. B. (2000). *Migrating from IGES to STEP: one to one translation of IGES drawing to STEP drafting data* (T. 41).
12. Pascal Frey, Paul Louis George. (2000). Mesh Generation Application To Finite Elements. Esantis *Mesh Generation*. ISTE Publishing Company.
13. Peterson, S. (2020 m. 01 05 d.). *COMPUTING CONSTRAINED DELAUNAY TRIANGULATIONS*. Nuskaityta iš COMPUTING CONSTRAINED DELAUNAY TRIANGULATIONS: http://www.geom.uiuc.edu/~samuelp/del_project.html
14. Powers, S. (2009). *Practical RDF*. O'Reilly Media.
15. Ruiter, M. d. (1988). *Advance in Computer Graphics III*. Springer-Verlag Berlin Heidelberg.
16. SCRA. (2006). *STEP Application Handbook ISO 10303 version 3*.
17. Smith, C. (2006). *On Vertex-Vertex Systems and Their Use in*.
18. Stroud, I. (2006). *Boundary Representation Modelling Techniques*.

19. Szirmay-Kalos L, Ma'rton G, Dobos B, Horva'th T, Risztics P, Kova'cs E. (1995). *Theory of three-dimensional computer graphics*. Budapest: Akademiai Kiado.
20. TransMagic. (2020 m. 01 5 d.). Nuskaityta iš TransMagic: <https://transmagic.com/iges/>
21. V. Quintana, R. L. (2012). *Re-engineering the Engineering Change Management process for a drawing-less environment* (T. 63(1)).
22. Y., Y. Y. (2009). Pattern Recognition Letters. *ELSEVIER*.