UNIVERZA V LJUBLJANI

Fakulteta za strojništvo

Generator mreže v prostorskih kontinuumih za numerične izračune

DIPLOMSKA NALOGA UNIVERZITETNEGA ŠTUDIJA

Matjaž Šubelj

Ljubljana, 2001

UNIVERZA V LJUBLJANI

Fakulteta za strojništvo

Generator mreže v prostorskih kontinuumih za numerične izračune

DIPLOMSKA NALOGA UNIVERZITETNEGA ŠTUDIJA

Matjaz Šubelj

Mentor: izr. prof. dr. Jože Duhovnik, dipl. ing.

Ljubljana, 2001

Zahvala

Zahvaljujem se staršem za potrpežljivost in finančno podporo med mojim študijem.

Zahvaljujem se tudi profesorju dr. Jožetu Duhovniku in vsem sodelavcem v laboratoriju LECAD za pomoč in nasvete v času mojega demonstratorstva v laboratoriju.

Glede na Zakon o avtorskih in sorodnih pravicah UL RS št. 21/1995 in Zakon o industrijski lastnini UL RS št. 13/1992, 13/1993, 27/1993, 34/1997 in 75/1997 velja še naslednje:

Diplomsko nalogo – arhivski izvod si je možno ogledati samo v prostorih knjižnice Fakultete za strojništvo v Ljubljani s pisnim dovoljenjem:

1. avtorja – diplomanta
   Matjaž Šubelj ________________

Če ni avtorjevega – diplomantovega podpisa, je diplomska naloga v knjižnici Fakultete za strojništvo v Ljubljani nedostopna za vpogled.

(U) Tek. Štev.: 5128 UDK 004.42

GENERATOR MREŽE V PROSTORSKIH KONTINUUMIH ZA NUMERIČNE IZRAČUNE

Matjaž Šubelj

Ključne besede: blok-strukturirane mreže
B-zlepki
trilinearna interpolacija
generator mrež
računalniška grafika
računalniško programiranje

Izvleček:

Osnovni opis blok-strukturiranih mrež. Definiranje geometrijske baze podatkov. Matematični zapis krivulj B-zlepkov, Coonsovih krp, Gordonovih površin in površinskih B-zlepkov. Matematični popis volumna s trilinearno interpolacijo. Izvedbe generatorjev mrežnih šablon. Opis, izvedba in uporaba interaktivnega grafičnega urejevalnika blok-strukturiranih mrež.

(U) No.: 5128 UDC 004.42

GRID GENERATOR FOR COMPLEX 3D DOMAINS

Matjaž Šubelj

Key words: block-structured grids
B-splines
trilinear interpolation
grid generator
computer graphics
computer programming

Abstract:

Description of block-structured grids. Definition of geometric database. Mathematical notation of B-splines, Coons patches, Gordon surfaces and surface B-splines. Mathematical representation of volume using trilinear interpolation. Implementation of grid template generators. Description, implementation and usage of interactive graphical editor for block-structured grids.

Kazalo

Poglavje 1

Uvod

1.1 Namen diplomske naloge

Numerična analiza toka fluida (ang. *Computational Fluid Dynamics* - CFD) zahteva diskretizacijo prostora, skozi katerega teče fluid. Prostor diskretiziramo z generiranjem mreže, ki prostor razdeli na končne elemente oziroma končne volumne. Kvaliteta mreže je ključnega pomena za uspešno analizo in njeno generiranje praviloma zahteva največ dela. Preostalo delo se deli na sam izračun, ki ga opravi računalnik, in vrednotenje rezultatov. Največji doprinos k produktivnosti nam torej prinese hitro generiranje kvalitetnih mrež.

Za analizo toka fluida so se izkazale kot najbolj primerne strukturirane mreže, ki razdelijo obravnavani računski prostor v poljubno število osnovnih gradnikov—v štirikotnike za 2D prostor in v heksaedre za 3D prostor. Gradniki so urejeni v $I \times J$ oziroma v $I \times J \times K$ polje. Strukturirane mreže so najpogosteje dobljene z razdelitvijo obravnavanega prostora na določeno število podprostorov, ki so nadalje razdeljeni na osnovne gradnike. Take mreže imenujemo blok-strukturirane mreže, njihova prednost pa je v tem, da s podprostori lažje prilagajamo gostoto in porazdelitev mreže kompleksni geometriji s čimer dobimo ugodnejšo porazdelitev računskih točk.

Namen te diplomske naloge je predstaviti problematiko generiranja blok-strukturiranih mrež v prostorskih kontinuumih in nadaljni razvoj skupine programskih orodij, ki so rezultat dosedanjega dela na tem področju.

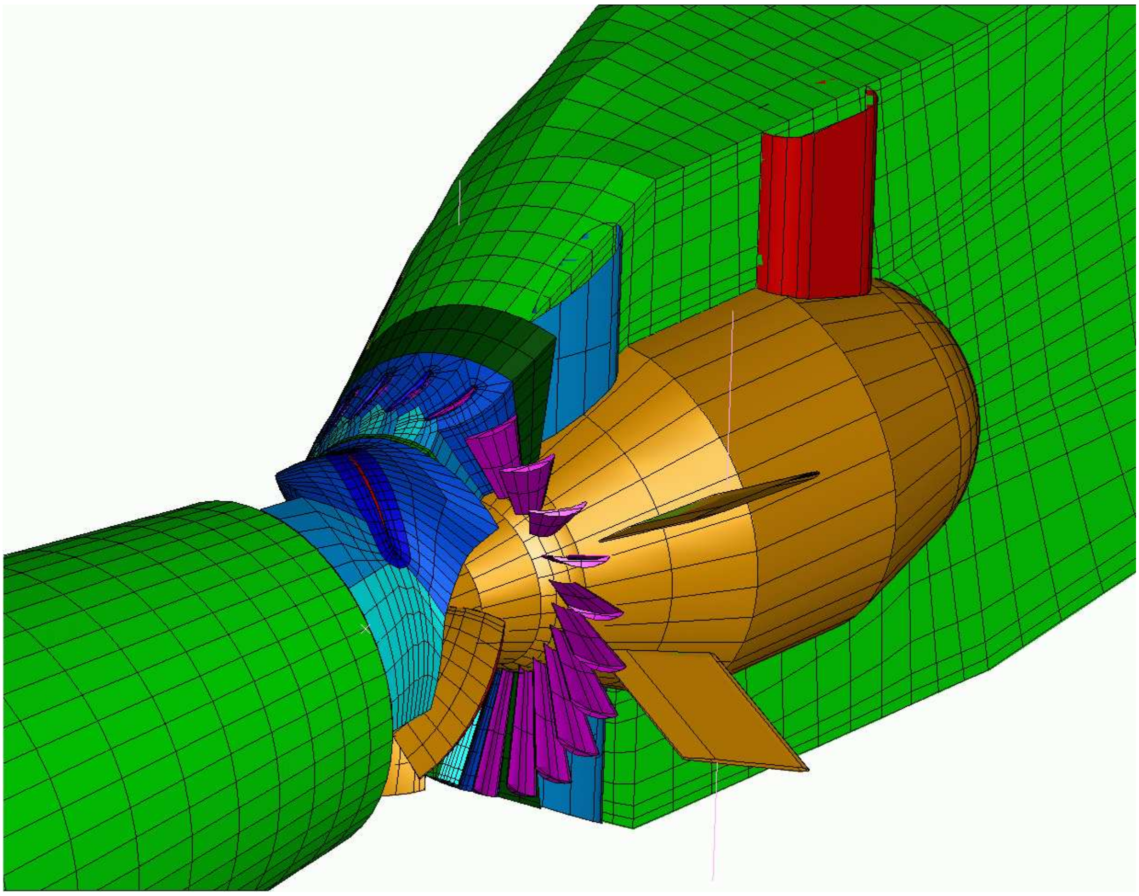
1.2 Izhodišča naloge

Pri razvoju skupine orodij za generiranje mrež v geometrijsko kompleksnih računskih prostorih sem sodeloval v okviru preddiplomskega seminarja. Rezultat razvoja je geometrijska baza podatkov, v kateri so shranjeni geometrijski in topološki podatki o mreži, generator mreže, ki uporablja transfinitno interpolacijo vozlišč znotraj območij, in interaktivni grafični urejevalnik mreže, s katerim lahko spreminjamo obliko podprostorov in porazdelitev vozlišč na robovih. Moja naloga pri razvoju orodij je bila izdelava grafičnega urejevalnika.

1.3 Cilji naloge

Glavni cilj diplomske naloge je povečanje funkcionalnosti grafičnega urejevalnika. Dodati mu je potrebno med drugim možnost ogleda generirane mreže, da se lahko takoj preveri kvaliteta elementov. Vgrajenemu tekstovnemu oknu je potrebno dodati ukazno vrstico, ki bo prepoznala nabor tipkanih ukazov s pripadajočimi parametri. Nabor ukazov naj bo tak, da bo možno spreminjati šablono mreže oziroma narediti novo šablono. Izdelati je potrebno tudi nekaj primerov mrežnih šablon.

Končni cilj razvoja omenjenih programskih orodij je hitra ocena kvalitete generirane mreže in po potrebi tudi njeno spremembo. Zaradi združenega prikazovalnika in generatorja mreže so posledice popravljene geometrije mreže vidne praktično takoj. Na ta način lahko v relativno kratkem času pripravimo kvalitetno mrežo za numerično analizo.



Slika 1.1: Diskretizacija pretočnega trakta cevne vodne turbine (Sava Vrhovo)

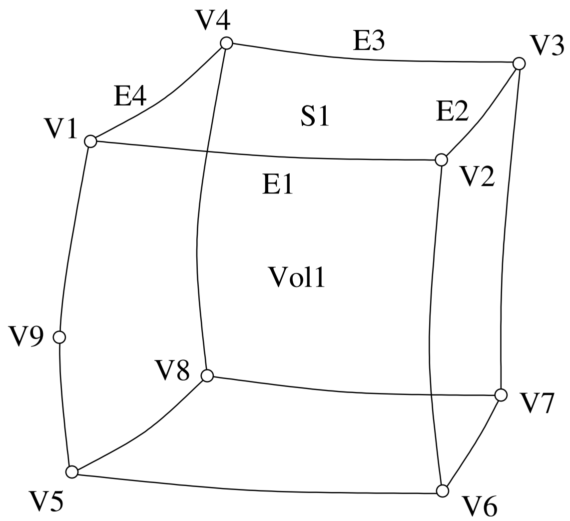
Poglavje 2

Geometrijska baza podatkov

2.1 Predstavitev baze

Da bi z grafičnim urejevalnikom lahko interaktivno popravljali mrežo, potrebujemo popoln geometrijski in topološki opis mreže. Ker imamo opravka z blok-strukturiranimi mrežami, potrebujemo tak opis za vsak podprostor računskega področja, nato pa še opis medsebojne povezave podprostorov. V ta namen je bila razvita geometrijska baza podatkov, ki skupaj s svojim naborom funkcij za interakcijo z mrežo zagotavlja topološko konsistentnost baze.

Baza loči med štirimi vrstami osnovnih gradnikov mreže, vsak tak gradnik pa nosi zapis lastnih topoloških in geometrijskih podatkov. Ti gradniki so vogal, rob, površina in volumen (slika 2.1).



Slika 2.1: Osnovni gradniki baze: *V*-vogal, *E*-rob, *S*-površina, *Vol*-volumen

Vogal Predstavlja vozlišče enega ali več robov, lahko pa predstavlja tudi neke vrste kontrolno točko roba, s katero lahko spreminjamo obliko roba[1]. Vogal vedno predsta-

[1] Ime "vogal" ni najbolj ustrezno, vendar naj "vozlišče" ostane rezervirano za uporabo v zvezi z mrežo.

vlja točko, ki leži neposredno na robu.

Rob Lahko predstavlja robove računskih podprostorov ali mejo računskega območja (npr. profil lopate vodilnika). Vsakemu robu, ki predstavlja rob računskega podprostora, lahko predpišemo lastno porazdelitev vozlišč mreže vzdolž robu.

Površina Podobno kot rob predstavlja meje računskih območij. Vsaka površina (z nekaterimi izjemami) je omejena s štirimi robovi in na njej lahko tvorimo ustrezno mrežo.

Volumen Predstavlja računske podprostore. Celotno računsko območje je sestavljeno iz enega ali več volumnov. Vsak volumen je omejen s šestimi površinami in v njem lahko generiramo ustrezno mrežo.

Ker je potrebno včasih generirati ”veliko” mrežo (npr. 100.000 elementov ali več), se generirana mreža ne shrani kot del baze, saj bi bil zapis baze na disku precej velik. Namesto tega shranijo posamezni gradniki vse podatke o mreži, ki so potrebni za njeno ponovno generiranje (npr. porazdelitev vozlišč po robovih, gostota vozlišč). Za potrebe analize se sproži generacija mreže, ta pa se potem izvozi v format, ki je razumljiv programu, ki analizo izvaja.

2.2 Topološka odvisnost gradnikov

V zgornjih odstavkih so že omenjene nekatere topološke značilnosti baze. Topološka odvisnost gradnikov je določena z njihovo usmerjenostjo, povezavami in geometrijskimi pogoji.

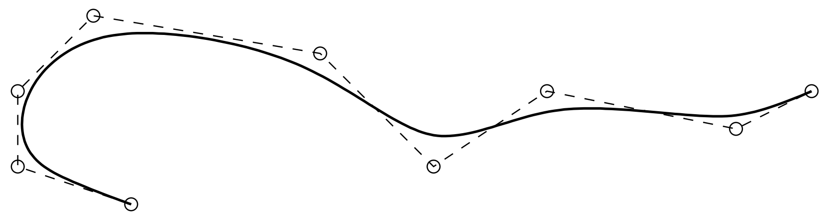
- Usmerjenost gradnika je med drugim pomembna za pravilno generiranje mreže. Razen vogala imajo vsi gradniki seznam topološko nižjih gradnikov iz katerih so sestavljeni. Vrstni red v tem seznamu določa usmerjenost gradnika. Usmerjenost, npr. roba, je določena z začetnim in končnim vogalom.

- Povezave med gradniki omogočajo konsistentnost baze. Vsak gradnik razen volumna ima enega ali več seznamov gradnikov, ki so od njega neposredno odvisni. Če npr. spremenimo število vozlišč mreže na nekem robu, se bo število vozlišč popravilo tudi na vseh ostalih robovih, ki s prvim tvorijo isto mrežo. Popis povezav v obe smeri omogoča hitro popravljanje baze, kar je pogoj za interaktivno delo.

- Geometrijski pogoji definirajo odvisnost med gradniki istega ali različnega topološkega ranga. Sem spadajo npr. zahteve, da nek vogal leži na nekem robu ali površini na določenem mestu, da je neka skupina vogalov periodična in podobno.

2.3 Geometrijski zapis robov

Zapis gradnikov v bazi je v kartezičnih koordinatah. Geometrijsko je rob predstavljen kot kubični B-zlepek (ang. *B-spline*). Zapis roba kot B-zlepka nam omogoča enostaven popis zapletenih krivulj, enostavno preoblikovanje roba, neodvisnost zapisa od prostorskih transformacij ipd.. Zaradi teh ugodnih lastnosti so B-zlepki in njihove izpeljanke NURBS najbolj uveljavljeni zapis krivulj pri današnjih geometrijskih modelirnikih.



Slika 2.2: Primer krivulje B-zlepka in njenih kontrolnih točk

Krivuja B-zlepkov (slika 2.2) je definirana kot zaporedje točk, ki jih dobimo, če enačbo 2.1 izračunamo za vse vrednosti parametra u.

$$\boldsymbol{P}(u) = \sum_{i=1}^{n} N_{i,k}(u)\boldsymbol{P}_i \tag{2.1}$$

Označbe v zgornji enačbi pomenijo:

- $\boldsymbol{P}(u)$ - točka na B-zlepku
- $N_{i,k}(u)$ - osnovna funkcija B-zlepka
- $\boldsymbol{P}_i$ - i-ta kontrolna točka

Osnovna funkcija B-zlepka $N_{i,k}$ je definirana s pomočjo Cox-de-Boor-ove rekurzivne enačbe (Watt [6], str. 124):

$$N_{i,1}(u) = \begin{cases} 1 & t_i \leq u < t_{i+1} \\ 0 & \text{v drugih primerih} \end{cases} \tag{2.2}$$

$$N_{i,k}(u) = \frac{(u - t_i)\, N_{i,k-1}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u)\, N_{i+1,k-1}(u)}{t_{i+k} - t_{i+1}} \tag{2.3}$$

kjer je:

- (k-1) - stopnja osnovne funkcije
- t_i - element vozliščnega vektorja
- u - parameter krivulje
- n - število kontrolnih točk

Vrednosti elementov vozliščnega vektorja za odprto krivuljo prikazujejo enačbe 2.4. Vozliščni vektor ima $n + k$ elementov. S tem se zagotovi, da gre B-zlepek skozi začetno in končno kontrolno točko. Prva in zadnja vrednost se v vozliščnem vektorju ponovita k-krat:

$$\begin{array}{ll} t_i = 0 & i < k \\ t_i = i - k + 1 & k \leq i \leq n \\ t_i = n - k + 2 & i > n \end{array} \tag{2.4}$$

Pri izračunu enačb 2.2 in 2.3 moramo upoštevati naslednje omejitve:

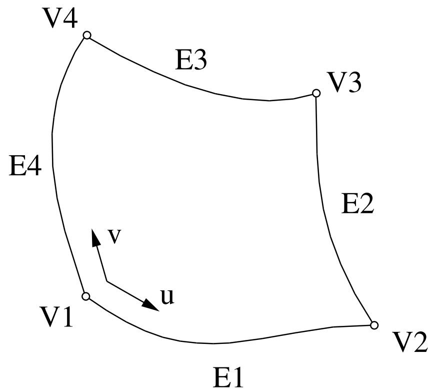
$$\begin{array}{l} \frac{0}{0} = 1 \\ 0 \leq i \leq n + k \\ 0 \leq u \leq n - k + 2 \end{array} \tag{2.5}$$

2.4 Geometrijski zapis površin

Geometrijska predstavitev površin je odvisna od vrste robov, ki jih omejujejo. Če so ti robovi B-zlepki, potem je tudi površina sama B-zlepek in točke na površini se izračunajo glede na definicijo površinskega B-zlepka. Zapis s površinskimi B-zlepki nam da enako ugodne lastnosti kot prej omenjeni zapis robov. V ostalih primerih je površina definirana kot bilinearna interpolacija zunanjih robov—Coons-ova krpa. Ponekod imamo na voljo tudi vmesne krivulje, ki ležijo na površini. Če so tudi te shranjene v bazi, jih lahko pri interpolaciji upoštevamo—dobimo t.i. Gordonove površine.

2.4.1 Bilinearna interpolacija: Coonsove krpe

Kadar imamo na voljo podatke o robnih krivuljah, ki omejujejo neko ploskev, ne poznamo pa poteka površine med krivuljami, moramo za popis površine uporabiti interpolacijo. Preproste interpolacije dobimo s polinomi, od katerih je najpreprostejša linearna interpolacija.



Slika 2.3: Površina omejena s štirimi krivuljami

Kadar je površina omejena s štirimi robnimi krivuljami, interpoliramo najprej med dvema nasprotnima krivuljama, nato pa še med štirimi točkami v katerih se robne krivulje dotikajo (slika 2.3). Dobimo tri površine, ki so razpete med krivuljama oz. točkami. Če sedaj seštejemo prvi dve in odštejemo tretjo, dobimo t.i. Coons-ovo krpo, ki predstavlja bilinearno interpolacijo površine med štirimi robnimi krivuljami.

Površina, razpeta med krivulji $\boldsymbol{E}_1$ in $\boldsymbol{E}_3$, je:

$$\boldsymbol{P}_1(u,v) = (1-v)\,\boldsymbol{E}_1(u) + v\,\boldsymbol{E}_3(u) \tag{2.6}$$

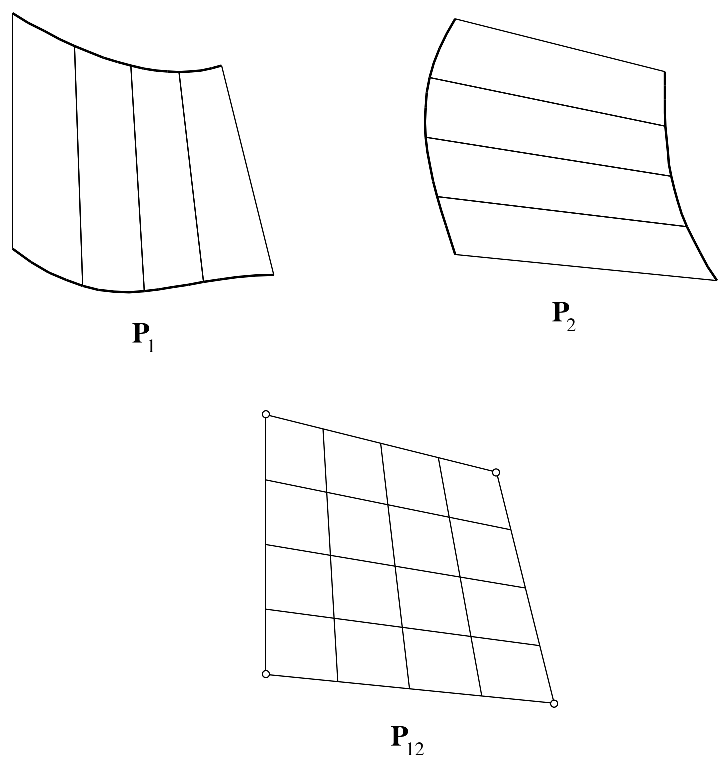
Za krivulji $\boldsymbol{E}_2$ in $\boldsymbol{E}_4$ velja analogno:

$$\boldsymbol{P}_2(u,v) = (1-u)\,\boldsymbol{E}_4(v) + u\,\boldsymbol{E}_2(v) \tag{2.7}$$

Površina, razpeta med točkami $\boldsymbol{V}_1$, $\boldsymbol{V}_2$, $\boldsymbol{V}_3$, $\boldsymbol{V}_4$, pa je:

$$\boldsymbol{P}_{12}(u,v) = [1-u, u]\begin{bmatrix}\boldsymbol{V}_1, \boldsymbol{V}_4\\ \boldsymbol{V}_2, \boldsymbol{V}_3\end{bmatrix}\begin{bmatrix}1-v\\ v\end{bmatrix} \tag{2.8}$$

Vse tri površine so prikazane na sliki 2.4.



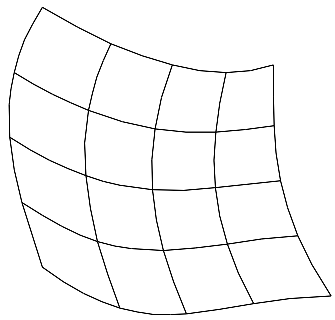
Slika 2.4: Površine dobljene s posamičnimi linearnimi interpolacijami

Coonsova krpa je torej:

$$\boldsymbol{P} = \boldsymbol{P}_1 + \boldsymbol{P}_2 - \boldsymbol{P}_{12}$$

$$
\begin{aligned}
\boldsymbol{P}(u,v) &= [1-v,v]\begin{bmatrix}\boldsymbol{E}_1(u)\\ \boldsymbol{E}_3(u)\end{bmatrix}\\
&\quad + [1-u,u]\begin{bmatrix}\boldsymbol{E}_4(v)\\ \boldsymbol{E}_2(v)\end{bmatrix}\\
&\quad - [1-u,u]\begin{bmatrix}\boldsymbol{V}_1,\boldsymbol{V}_4\\ \boldsymbol{V}_2,\boldsymbol{V}_3\end{bmatrix}\begin{bmatrix}1-v\\ v\end{bmatrix}
\end{aligned}
\tag{2.9}
$$

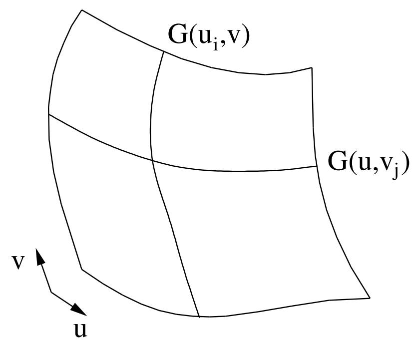
Končna površina je prikazana na sliki 2.5.



Slika 2.5: Površina dobljena z bilinearno interpolacijo med robnimi krivuljami

2.4.2 Bilinearna interpolacija: Gordonove površine

Gordonove površine so posplošitev Coonsovih krp. Pogosto nam modeliranje površine iz samo robnih krivulj ne zadostuje, imamo pa na voljo mrežo izoparametričnih krivulj $\boldsymbol{G}(u_i,v)$, $i=0,\dots,m$ in $\boldsymbol{G}(u,v_j)$, $j=0,\dots,n$, ki opisujejo površino (slika 2.6). Če linearno interpoliramo med vsemi temi krivuljami, dobimo t.i. Gordonovo površino.



Slika 2.6: Gordonova površina je bilinearna interpolacija med ortogonalno mrežo krivulj

Površina, ki interpolira krivulje $\boldsymbol{G}(u_i,v)$, je:

$$
\boldsymbol{P}_1(u,v) = \sum_{i=0}^{m} \boldsymbol{G}(u_i,v)\, L_i^m(u) \tag{2.10}
$$

Pri tem je $L_i^m(u)$ Lagrange-ev polinom stopnje m:

$$L_i^m(u) = \frac{\prod_{\substack{j=0\\j\neq i}}^{m}(u-u_j)}{\prod_{\substack{j=0\\j\neq i}}^{m}(u_i-u_j)} \tag{2.11}$$

Analogno je površina, ki interpolira krivulje $\boldsymbol{G}(u, v_j)$:

$$\boldsymbol{P}_2(u,v) = \sum_{j=0}^{n} \boldsymbol{G}(u,v_j)\, L_j^n(v) \tag{2.12}$$

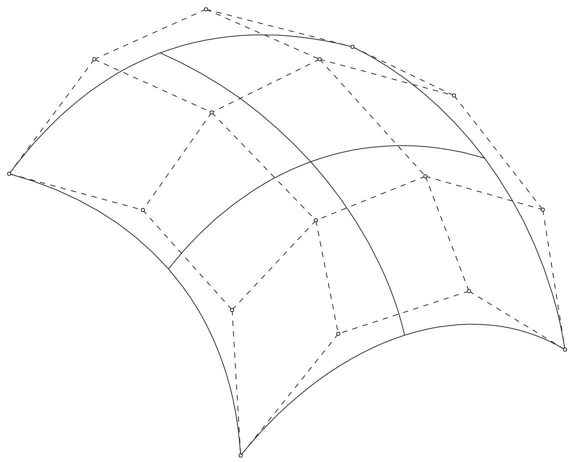
Površina, ki interpolira med presečišči krivulj, je:

$$\boldsymbol{P}_{12}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \boldsymbol{G}(u_i,v_j)\, L_i^m(u) L_j^n(v) \tag{2.13}$$

Gordonovo površino dobimo zopet kot:

$$\boldsymbol{P} = \boldsymbol{P}_1 + \boldsymbol{P}_2 - \boldsymbol{P}_{12} \tag{2.14}$$

2.4.3 Površinski B-zlepek



Slika 2.7: Primer površinskega B-zlepka in njegovih kontrolnih točk

Površinski B-zlepek (slika 2.7) lahko zapišemo kot tenzorski produkt dveh krivulj B-zlepkov:

$$\boldsymbol{P}(u,v) = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,k}(u) N_{j,l}(v) \boldsymbol{P}_{i,j} \tag{2.15}$$

Površina $\boldsymbol{P}(u, v)$ je definirana podobno kot krivulja B-zlepka z:

- neodvisnima parametroma u in v
- dvema vozliščnima vektorjema velikosti $n + k$ in $m + l$
- dvema stopnjema k (smer u) in l (smer v) osnovnih krivulj $N_{i,k}(u)$ in $N_{j,l}(v)$
- tabelo kontrolnih točk, definirano s $\boldsymbol{P}_{i,j}$, velikosti $n \times m$

2.5 Glavni gradniki baze

2.5.1 Vogali

Vogal je topološko najnižji gradnik in je edini, ki ga je možno v grafičnem urejevalniku interaktivno premikati oz. spreminjati njegove koordinate. Ker interakcija poteka v 3D prostoru, je potrebna neka dodatna informacija, da lahko iz zaslonskih koordinat izračunamo dejanske prostorske koordinate. Ta informacija je podana v obliki geometrijskega pogoja, ki pove na katerem robu ali površini vogal leži. Premikanje vogala je omejeno le na ta rob oz. površino. Pogoj spada med ostale že omenjene topološke odvisnosti, ki skrbijo za konsistentnost baze.

Vsak vogal v bazi je definiran na naslednji način:

Vogal {

- *ime*
- *3D kartezična koordinata*
- *seznam odvisnih robov*
- *seznam odvisnih površin*
- *seznam odvisnih volumnov*
- *seznam geometrijskih pogojev*
- *vogal je viden*
- *vogal je možno premikati*

}

2.5.2 Robovi

Rob ima lahko poleg začetnega in končnega vogala tudi več "kontrolnih" vogalov. Ti ležijo na robu med začetnim in končnim vogalom in skupaj z njima predstavljajo kontrolne točke roba s katerimi lahko spreminjamo njegovo obliko. Njihov vpliv na obliko roba je podoben vplivu kontrolnih točk B-zlepka, le da ležijo na samem robu. Seznam vmesnih kontrolnih vogalov je podan kot seznam odvisnih vogalov. Geometrijski pogoj roba nam pove za kakšen tip roba gre (raven rob, B-zlepek) in ali leži na katerem drugem gradniku (površini, drugemu robu).

Vsak rob v bazi je tako definiran kot:

Rob {

- *ime*
- *indeks začetnega in končnega vogala*
- *polje točk na robu*
- *geometrijski pogoj*
- *definicija B-zlepka*

- *porazdelitev vozlišč mreže*
- *seznam odvisnih vogalov*
- *seznam odvisnih površin*
- *seznam odvisnih volumnov*
- *rob je viden*
- *rob je periodičen*

}

2.5.3 Površine

Površina je najnižji topološki gradnik na katerem se lahko generira mreža. Porazdelitev in gostota mreže je odvisna od porazdelitve vozlišč po robovih, ki površino omejujejo. Površine, ki predstavljajo meje celotnega računskega območja, so označene kot zunanje površine. Na teh površinah se generira in prikaže mreža, ko v grafičnem urejevalniku sprožimo predogled mreže.

Površina v bazi je definirana kot:

Površina {

- *ime*
- *seznam vogalov*
- *seznam robov*
- *polje točk na površini*
- *definicija površinskega B-zlepka*
- *seznam Gordonovih krivulj*
- *polje vozlišč mreže*
- *seznam odvisnih volumnov*
- *površina je vidna*
- *površina je zunanja*

}

2.5.4 Volumni

Volumen je najvišji topološki gradnik baze in predstavlja računske podprostore. Njegova struktura je zelo enostavna, saj se uporablja samo kot nosilec mreže.

Volumen je definiran kot:

Volumen {

- *ime*
- *seznam vogalov*

- *seznam robov*
- *seznam površin*
- *polje vozlišč mreže*

}

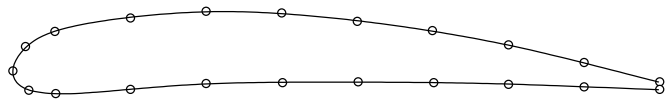
2.6 Generiranje baze

Generiranje geometrijske baze podatkov je bilo razvito v programskem jeziku C, ki poleg hitrega izvajanja omogoča tudi definiranje knjižnice funkcij. Zaradi kompleksnosti baze so morale biti istočasno razvite tudi funkcije za spreminjanje oz. upravljanje baze, ki imajo vgrajene mehanizme za ohranjanje konsistentnosti baze. Baza se generira v glavnem preko teh funkcij. Generator baze predstavlja določen tip mrežne šablone, po kateri razdeli osnovno geometrijsko domeno v poddomene. Za vsak tip mrežne šablone obstaja drug generator baze. Generatorji baze in mrežne šablone so podrobneje opisani v poglavju 3 na strani 24.

Razpon nalog, ki jih opravljajo funkcije za generiranje baze, je zelo širok. Obsega od najbolj osnovnih, recimo normiranje vektorja ali branje datotek, do zahtevnejših, kot je npr. iskanje najbližje točke na površini. Nekatere ključne funkcije so razdelane v naslednjih podpoglavjih.

2.6.1 Interpolacija podanih točk s kubičnim B-zlepkom

Osnovni podatki za generiranje baze so običajno množice točk v prostoru, ki predstavljajo neko karakteristično obliko. Pri turbinskih strojih je tak zapis tipičen za profil lopate turbine. Ker potrebujemo popoln opis profila lopate, moramo skozi množico točk napeljati krivuljo, ki se mora obnašati čim bolj naravno, t.j. biti mora gladka (slika 2.8). V praksi pogosto potrebujemo interpolacijo skozi točke, ki so podane v prostoru.



Slika 2.8: Interpolacija podanih točk z gladko krivuljo

Število podanih točk je lahko zelo različno (za profil lopate npr. od 20 do 100), zato se v praksi najbolje obnese interpolacija z zlepki. Zaradi svojih ugodnih lastnosti smo v našem primeru izbrali kubični B-zlepek, ki smo ga zapisali kot:

$$\boldsymbol{P}(u) = \sum_{i=1}^{n} N_{i,k}(u)\boldsymbol{P}_i \tag{2.16}$$

Pri tem je u parameter krivulje (običajno v normiranem intervalu [0,1]), $\boldsymbol{P}_i$ so kontrolne točke zlepka, $N_{i,k}$ pa osnovne (bazne) funkcije po enačbah 2.2 in 2.3. Podane naj bodo točke $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_L$, katerim naj ustrezajo vrednosti parametra $u_0, \ldots, u_L$. Problem interpolacije bo rešen, ko bomo določili kontrolne točke B-zlepka $\boldsymbol{P}_{-1}, \ldots, \boldsymbol{P}_{L+1}$ tako, da bo $\boldsymbol{P}(u_i) = \boldsymbol{x}_i$. Izkaže se, da je problem enolično rešljiv le v primeru, ko izberemo še nekatere dodatne pogoje:

1. Določiti je potrebno vrednosti $u_0, \ldots, u_L$, ki v splošnem niso podane. Izberemo, npr., da je vrednost parametra kar normirana razdalja med točkami.

2. Začetna in končna kontrolna točka naj sovpadata z začetno in končno interpolacijsko točko: $\boldsymbol{P}_{-1} = \boldsymbol{x}_0$ in $\boldsymbol{P}_{L+1} = \boldsymbol{x}_L$.

3. Tangenta na začetku in koncu krivulje v splošnem prav tako ni podana, zato ji predpišemo, npr., smer tangente na parabolo skozi zadnje tri točke (t.i. Besselov robni pogoj)

Po izpeljavi (Farin [1], str. 122) se rešitev problema prevede na sistem enačb:

$$\begin{bmatrix} 1 & & & & & & \\ \alpha_1 & \beta_1 & \gamma_1 & & & & \\ & & & \ddots & & & \\ & & & & \alpha_{L-1} & \beta_{L-1} & \gamma_{L-1} \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_0 \\ \boldsymbol{P}_1 \\ \vdots \\ \boldsymbol{P}_{L-1} \\ \boldsymbol{P}_L \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_0 \\ \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_{L-1} \\ \boldsymbol{r}_L \end{bmatrix} \tag{2.17}$$

Koeficienti so:

$$\begin{aligned} \alpha_i &= \frac{(\Delta_i)^2}{\Delta_{i-2} + \Delta_{i-1} + \Delta_i} \\ \beta_i &= \frac{\Delta_i(\Delta_{i-2} + \Delta_{i-1})}{\Delta_{i-2} + \Delta_{i-1} + \Delta_i} + \frac{\Delta_{i-1}(\Delta_i + \Delta_{i+1})}{\Delta_{i-1} + \Delta_i + \Delta_{i+1}} \\ \gamma_i &= \frac{(\Delta_{i-1})^2}{\Delta_{i-1} + \Delta_i + \Delta_{i+1}} \end{aligned} \tag{2.18}$$

pri čemer je $\Delta_i = \Delta u_i$. Elementi na desni strani so:

$$\boldsymbol{r}_i = (\Delta_{i-1} + \Delta_i)\, \boldsymbol{x}_i \tag{2.19}$$

in Besselova robna pogoja (Farin [1], str. 127):

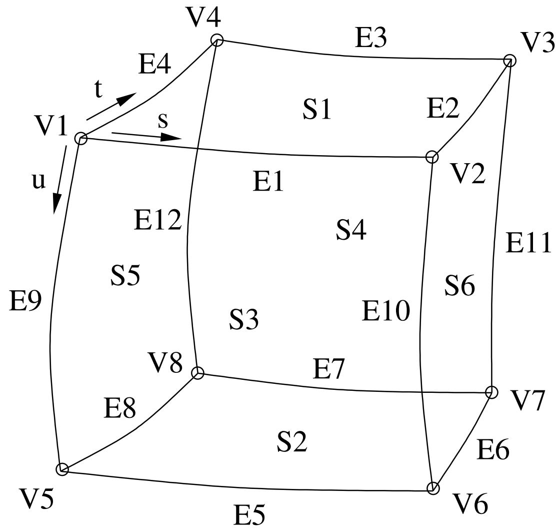
$$\boldsymbol{r}_0 = -\frac{2(2\Delta_0 + \Delta_1)}{\Delta_0 \beta_1}\, \boldsymbol{x}_0 + \frac{\beta_1}{2\Delta_0 \Delta_1}\, \boldsymbol{x}_1 - \frac{2\Delta_0}{\Delta_1 \beta_1}\, \boldsymbol{x}_2 \tag{2.20}$$

$$\boldsymbol{r}_L = -\frac{2\Delta_{L-1}}{\Delta_{L-2}\beta_{L-1}}\, \boldsymbol{x}_{L-2} - \frac{\beta_{L-1}}{2\Delta_{L-2}\Delta_{L-1}}\, \boldsymbol{x}_{L-1} + \frac{2(2\Delta_{L-1} + \Delta_{L-2})}{\Delta_{L-1}\beta_{L-1}}\, \boldsymbol{x}_L \tag{2.21}$$

Sistem enačb 2.17 je tridiagonalen, kar nam omogoča uporabo enostavnega in hitrega algoritma za njegovo reševanje. Število operacij je pri tem proporcionalno številu kontrolnih točk.

2.6.2 Zapis volumna z interpolacijo med robnimi površinami

V poglavju 2.4.1 smo izpeljali enačbo za parametrično iskanje točke na površini z bilinearno interpolacijo med robovi površine. Če enačbo 2.9 razširimo na tri dimenzije, lahko z njo določimo točko v volumnu, pri čemer linearno interpoliramo med robnimi površinami volumna. S tem dobimo trilinearno ali t.i. *transfinitno* interpolacijo.



Slika 2.9: Topologija in parametrizacija volumna

Pri razširitvi enačbe 2.9 na tri dimenzije uporabimo isti splošen recept kot v 2D primeru, pri čemer upoštevamo topološki dvig sodelujočih elementov: vsota linearnih interpolacij med nasprotnimi pari površin minus interpolacija med štirimi nasprotnimi robovi minus interpolacija med vogali.

$$\boldsymbol{P} = \boldsymbol{P}_1 + \boldsymbol{P}_2 + \boldsymbol{P}_3 - \boldsymbol{P}_{12} - \boldsymbol{P}_{13} - \boldsymbol{P}_{23} - \boldsymbol{P}_{123} \tag{2.22}$$

$$\boldsymbol{P}_1(s,t,u) = (1-s)\,\boldsymbol{S}_5(t,u) + s\,\boldsymbol{S}_6(t,u) \tag{2.23}$$

$$\boldsymbol{P}_{12}(s,t,u) = [1-t, t] \begin{bmatrix} \boldsymbol{E}_9(u), \boldsymbol{E}_{10}(u) \\ \boldsymbol{E}_{12}(u), \boldsymbol{E}_{11}(u) \end{bmatrix} \begin{bmatrix} 1-s \\ s \end{bmatrix} \tag{2.24}$$

$$\begin{aligned} \boldsymbol{P}_{123}(s,t,u) \;&=\; (1-u)\;[1-s, s] \begin{bmatrix} \boldsymbol{V}_1, \boldsymbol{V}_4 \\ \boldsymbol{V}_2, \boldsymbol{V}_3 \end{bmatrix} \begin{bmatrix} 1-t \\ t \end{bmatrix} \\ &\quad - u\;[1-s, s] \begin{bmatrix} \boldsymbol{V}_5, \boldsymbol{V}_8 \\ \boldsymbol{V}_6, \boldsymbol{V}_7 \end{bmatrix} \begin{bmatrix} 1-t \\ t \end{bmatrix} \end{aligned} \tag{2.25}$$

Točko v volumnu lahko sedaj izračunamo glede na poljubne parametre s, t in u, pri čemer manjkajoče člene v enačbi 2.22 razvijemo analogno po enačbah 2.23 in 2.24.

Tak postopek uporablja funkcija, ki generira mrežo v volumnu. Pri tem se računajo vozlišča mreže, ki so pravzaprav točke v prostoru. Vrednosti parametrov s, t, u določa porazdelitev vozlišč, ki jo ima predpisan vsak rob. Vozlišča na površinah volumna se računajo glede na geometrijski zapis površine, pri čemer se zopet upoštevajo parametri porazdelitve po mejnih robovih.

2.7 Upravljanje z bazo

Zaradi kompleksnih relacij med gradniki baze so bile skupaj z bazo napisane tudi funkcije za upravljanje baze. Uporaba teh funkcij zagotavlja konsistentnost baze v vsakem trenutku. Grafični urejevalnik spreminja bazo samo preko teh funkcij, bralni dostop do baze pa je neposreden. Funkcije, ki jih urejevalnik uporablja za upravljanje baze, so naslednje:

1. Branje in pisanje baze iz/v datoteko: baza se shrani v datoteko v ASCII formatu; shranijo se tudi podatki o porazdelitvi in gostoti vozlišč mreže, ne pa tudi vozlišča sama.

2. Premikanje vogala: funkcija popravi vse podatke v bazi, ki so odvisni od položaja vogala; vhodni podatek je približna projekcija novih koordinat na geometrijo.

3. Spreminjanje porazdelitve vozlišč mreže vzdolž roba: potrebne spremembe se avtomatično prenesejo na vse odvisne robove.

4. Generiranje mreže na posamezni površini: funkcija se kliče za posamezne površine pri vsakem predogledu mreže.

5. Generiranje mreže v posameznem volumnu: funkcija se izvede za vsak računski podprostor pred izvozom mreže v zunanji format.

6. Izvoz mreže v zunanji format: za potrebe analize se mreža izvozi v format, kot ga določa program za analizo; trenutno sta implementirana izvoza v SDRC I-DEAS univerzalni format in ICCM COMET format, oba tipa ASCII.

Poglavje 3

Generator geometrijske baze podatkov

3.1 Predstavitev

Blok-strukturirane mreže so se izkazale kot dober kompromis med možnostjo obravnave kompleksne geometrije in možnostjo uporabe najrazličnejših že obstoječih programov za numerične analize. Aplikacija blok-strukturiranih mrež na kompleksne geometrijske domene nam da popis začetne domene z geometrijsko enostavnejšimi poddomenami—bloki. Proces generiranja mreže se razdeli na vsak blok, povezave med mrežami posameznih blokov pa so določene z blokovno matriko domene. Blokovno omejeno generiranje mreže nam ponuja veliko kontrolo nad kvaliteto mreže v smislu gostote, ortogonalnosti in popačenosti mreže. Blok-strukturirane mreže izkazujejo tudi nekatere prednosti pri implementaciji vzporednega računanja.

Če se omejimo na numerične analize toka fluida, lahko večino obravnavanih gometrijskih domen tipiziramo. Tipične geometrijske domene so npr. območja okrog krila letala, območja okrog lopatic turbin, območja okrog kaskad lopatic vodilnika, obtekanje teles določenih oblik (trupi letal, ladij), sesalne cevi, spiralna ohišja ipd..

Za vsako od tipiziranih geometrijskih oblik lahko pripravimo šablono, po kateri kompleksno obliko razdelimo na enostavnejše bloke. Tak pristop k generiranju blok-strukturiranih mrež nam poleg enostavnosti omogoča veliko prilagodljivost in dobro kvaliteto začetne mreže.

3.2 Izvedba generatorja baze

Pojem generator baze predstavlja skupino ukazov, ki se izvedejo v določenem vrstnem redu in katerih rezultat je zgrajena geometrijska baza podatkov za določen tip geometrijske domene. Ukazi se interpretirajo kot klici funkcij, ki operirajo s podatki in gradijo bazo. Nabor teh funkcij je preddefiniran in je sestavni del baze. Vrsta in zaporedje ukazov določa tip obravnavane geometrijske domene—šablono, po kateri generator baze razdeli domeno na bloke. Poleg razdelitve začetne domene lahko šablona določa tudi začetno porazdelitev in gostoto mreže v posameznih blokih.

Generator baze je lahko izveden na naslednje načine:

- kot samostojen program, napisan v jeziku C,

- kot datoteka z makro ukazi, ki jih interpretira grafični urejevalnik oz. samostojni interpreter,
- kot zaporedje makro ukazov, tipkanih v tekstovnem vmesniku grafičnega urejevalnika.

Očitno je, da je zadnji način primeren le za najenostavnejše primere, uporaben pa je za dodajanje lastnosti oz. popravljanje baze. Nekatere druge značilnosti naštetih načinov izvedb prikazuje tabela 3.1.

izvedba	zahtevnost problemov	popravljanje šablone	hitrost generiranja	zahtevnost uporabe
C program	vse	težko[1]	hitro	srednja
makro datoteka	enostavna, srednja	enostavno	srednje	enostavna
tipkanje ukazov	enostavna	težko[2]	počasno	težka

Tabela 3.1: Značilnosti izvedb generatorja geometrijske baze podatkov

Najmočnejši način izvedbe je samostojen C program, saj nam omogoča definiranje najrazličnejših novih operacij nad podatki in s tem razširitev preddefiniranega nabora funkcij. Ker se funkcije kličejo neposredno, je čas generiranja baze zelo kratek. Slabost tega načina je potrebno znanje programskega jezika C in v primeru, da nimamo na voljo izvorne kode, tudi ne moremo popravljati že obstoječe šablone. Program zaženemo v direktoriju, v katerem se nahajajo datoteke s podatki, program pa generira datoteko z zapisom geometrijske baze podatkov.

Datoteka z makro ukazi je enostavnejši način izvedbe generatorja baze, vendar smo pri tem omejeni le na nabor ukazov, ki jih zna grafični urejevalnik interpretirati. V primeru kompleksne geometrijske domene utegne biti datoteka tudi precej dolga. Ker je datoteka v ASCII formatu, je popravljanje obstoječih šablon enostavno. Ta način je tudi najprimernejši za hitro prototipno izdelavo šablon. Bazo generiramo tako, da v ukazni vrstici grafičnega urejevalnika vnesemo ukaz za interpretiranje makro datotek, in mu podamo ime datoteke. Generirana baza se nato prikaže v urejevalniku.

Tipkanje ukazov v ukazni vrstici urejevalnika je osnovni način izvajanja makro datotek, saj so le-te samo skupek ukazov, ki se izvajajo vrstico za vrstico. Izdelava šablone na ta način pa je zamudno in težko opravilo in se zato ne priporoča.

3.2.1 Nabor ukazov za generiranje šablon

Makro ukazi predstavljajo prirejen nabor funkcij knjižnice baze. Trenutno je nabor omejen na funkcije za osnovno generiranje baze. Gradniki generirani z makro ukazi imajo privzeta imena, sestavljena iz karakteristične oznake in indeksa gradnika. Karakteristične oznake so **V**, **E**, **S** in **Vol** za vogal, rob, površino in volumen. Sintaksa makro ukazov je:

ukaz ***argument1 argument2*** ...

[1]Ni možno, če nimamo izvorne kode.

[2]Ker je proces enosmeren, v nekaterih primerih popravilo sploh ni možno.

Makro ukazi in njihovi argumenti so naslednji:

initgdb

Pripravi bazo za vnos podatkov. Ta ukaz mora nastopiti prvi, če želimo genrireti novo bazo.

settol ***tol***

Nastavi toleranco pri projekcijah točk na površine ali robove. Velikost tolerance je odvisna od podatkov, v splošnem naj bo enaka natančnosti podatkov. Toleranca ostane v veljavi do naslednjega klica.

tol	toleranca

vert ***vi x z y***

Generira nov vogal.

vi	indeks vogala
x y z	koordinate vogala

vertoe ***vi ei s***

Generira nov vogal in ga pripne na rob na dolžini *s*.

vi	indeks vogala
ei	indeks roba
s	parameter roba [0–1]

edge0 ***ei vi1 vi2 npoe filename on_si***

Iz datoteke prebere točke, jih interpolira z B-zlepkom in rob projecira na površino. Indeksa vogalov sta lahko -1, kar pomeni, da se ne upoštevata.

ei	indeks roba
vi1 vi2	indeks začetnega in končnega vogala
npoe	število točk na robu (diskretizacija)
filename	ime datoteke s točkami
on_si	indeks površine na katero se projecija

edge1 ***ei vi1 vi2 npoe on_si***

Generira nov rob, ki leži na površini ***on_si*** in poteka med vogaloma ***vi1*** in ***vi2***. Vogala morata ležati na isti površini.

ei	indeks roba
vi1 vi2	indeks začetnega in končnega vogala
npoe	število točk na robu
on_si	indeks površine na kateri leži rob

edge2 ***ei vi1 vi2 npoe***

Generira raven rob med vogaloma.

ei	indeks roba
vi1 vi2	indeks začetnega in končnega vogala
npoe	število točk na robu

edge3 ***ei vi1 vi2 npoei vi3 vi4 . . .***

Generira rob kot B-zlepek med vogali. Če sta podana samo prvi in končni vogal, je rob raven.

ei	indeks roba
vi1 vi2	indeks začetnega in končnega vogala
npoei	število točk narobu med dvema vogaloma
vi3 vi4 . . .	indeksi vogalov na robu od ***vi1*** do ***vi2*** (neobvezno)

edge4 ***ei vi1 vi2 npoe on_ei***

Generira nov rob, ki leži na robu ***on_ei*** in poteka med vogaloma ***vi1*** in ***vi2***. Vogala morata ležati na istem robu.

ei	indeks roba
vi1 vi2	indeks začetnega in končnega vogala
npoe	število točk na robu
on_ei	indeks roba na katerem leži rob

surfi ***si vi1 vi2 vi3 vi4 ei1 ei2 ei3 ei4***

Generira površino kot bilinearno interpolacijo med robovi. Vogali so vozlišča robov in določajo orientacijo površine.

si	indeks površine
vi1 . . . vi4	indeksi vogalov površine
ei1 . . . ei4	indeksi mejnih robov površine

surfbi ***si vi1 vi2 vi3 vi4***

Generira površino kot površinski B-zlepek, ki je razpet med podanimi vogali.

si	indeks površine
vi1 . . . vi4	indeksi vogalov površine

surfb ***si vi1 vi2 vi3 vi4 ei1 ei2 ei3 ei4 filename***

Generira površino kot površinski B-zlepek, katerega definicijo prebere iz datoteke. Indeksi vogalov in robov se ne upoštevajo, če imajo vrednost -1.

si	indeks površine
vi1 . . . vi4	indeksi vogalov površine
ei1 . . . ei4	indeksi mejnih robov površine
filename	ime datoteke z definicijo površinskega B-zlepka

vol ***vi vi1 . . . vi8 ei1 . . . ei8 si1 . . . si6***

Generira volumen.

vi	indeks volumna
vi1 . . . vi8	indeksi vogalov volumna
ei1 . . . ei8	indeksi mejnih robov volumna
si1 . . . si6	indeksi mejnih površin volumna

att2edge ***vi ei***

Projecira in pripne vogal ***vi*** na rob ***ei***.

vi	indeks vogala
ei	indeks roba

att2surf ***vi si***

Projecira in pripne vogal ***vi*** na površino ***si***.

vi	indeks vogala
si	indeks površine

coupvert ***vi on_i tvi ton_i type***

Poveže premikanje vogala ***tvi*** z vogalom ***vi***.

vi	indeks vogala, ki ga premikamo
on_i	indeks gradnika po katerem se premika ***vi***
tvi	indeks vogala, ki se premika zraven
ton_i	indeks gradnika po katerem se premika ***tvi***
type	Način povezave:

- **0** povezano po površini v smeri s
- **1** povezano po površini v smeri t

finddep

Poišče vse geometrijske in topološke odvisnosti med gradniki in jih zapiše v bazo.

chbias ***ei ne dtype ef depth***

Spremeni število in porazdelitev vozlišč vzdolž roba. ***depth*** določa prenos parametrov na odvisne robove. Pred tem je potrebno uporabiti ukaz **finddep**.

ei	indeks roba
ne	število elementov mreže
dtype	tip porazdelitve:

- **0** enakomerno
- **1** zgoščevanje proti koncu
- **2** zgoščevanje proti sredini
- **3** zgoščevanje od sredine proti koncu

ef	koeficient ekspanzije porazdelitve
depth	prenos parametrov na odvisne robove:

- **0** brez prenosa
- **1** na vse odvisne robove
- **2** prenos števila elementov na vse odvisne robove

verton, vertoff ***vi***

Vklapljanje in izklapljanje prikaza vogala.

vi	indeks vogala

edgeon, edgeoff ***ei***

Vklapljanje in izklapljanje prikaza roba.

ei	indeks roba

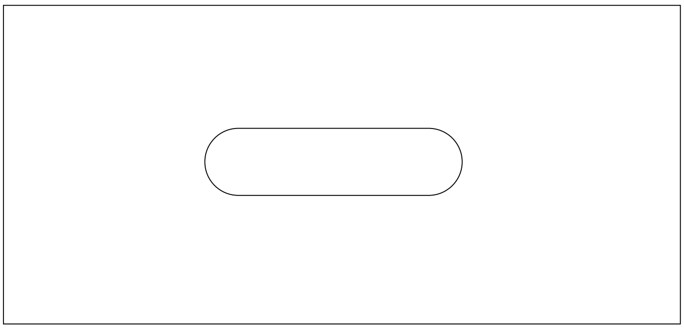
surfon, surfoff ***si***

Vklapljanje in izklapljanje prikaza površine.

si	indeks površine

3.2.2 Primer izdelave enostavne šablone

Za primer vzemimo izdelavo šablone z makro ukazi, ki jih bomo zapisali v datoteko. Predpostavimo, da imamo ravninsko računsko domeno, kot jo kaže slika 3.1. Dimenzije izberemo, npr. zunanje dimenzije 2x1 m in dimenzije ovala 0.8x0.2 m, izhodišče koordinatnega sistema pa v spodnjem levem kotu.



Slika 3.1: Ravninska računska domena z ovalno oviro v sredini

Zamislimo si neko razdelitev domene in jo poskušamo doseči z uporabo zgornjih makro ukazov. Ukaze pišemo v datoteko, ki se mora začeti z vrstico:

```
# TGMacro File 0.1
```

Ta določa tip in verzijo makro datoteke. Znak '#' na začetku ostalih vrstic določa komentar; interpreter preskoči celotno vrstico. Vsak ukaz z argumenti pišemo v svojo vrstico, pri čemer so ukaz in argumenti ločeni z vsaj enim presledkom ali tabulatorjem. Imena datotek zapišemo med dvojne narekovaje. Vsebina makro datoteke je tako lahko naslednja:

```
# TGMacro File 0.1
# simple example

initgdb
settol 0.001

# vogali nosilne povrsine
vert 0 0 0 0
vert 1 2 0 0
vert 2 2 1 0
vert 3 0 1 0

# nosilna povrsina
surfbi 0 0 1 2 3
surfoff 0

# nosilni robovi
edge0 0 -1-1 "points.dat" 0
```

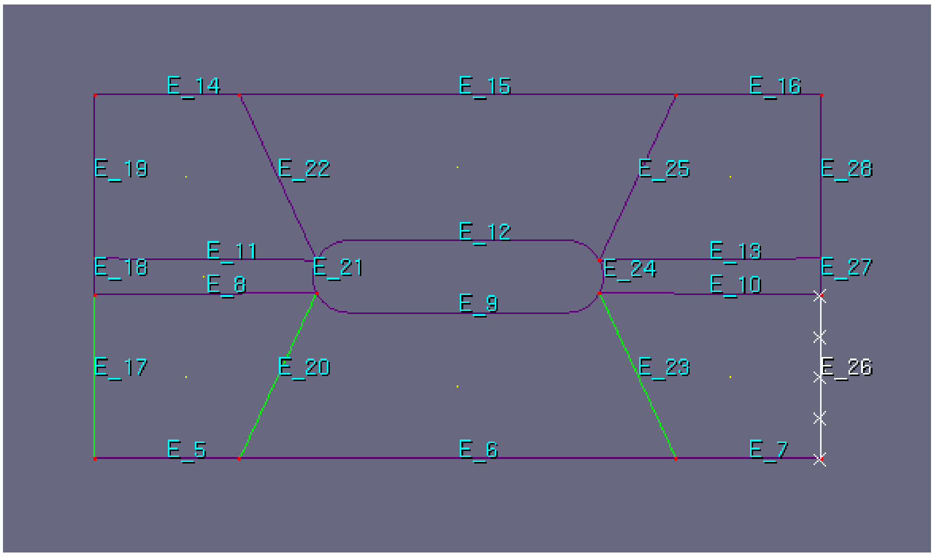
```
edge2 1 0 1 10
edge2 2 1 2 10
edge2 3 2 3 10
edge2 4 3 0 10

edgeoff 0
edgeoff 1
edgeoff 2
edgeoff 3
edgeoff 4

# vidni gradniki baze
vertoe 4 1 0.2
vertoe 5 1 0.8
vertoe 6 2 0.4
vertoe 7 2 0.6
.
.
.
surfi 6 10 15 9 3 11 22 14 19
surfi 7 15 14 8 9 12 25 15 22
surfi 8 14 7 2 8 13 28 16 25

coupvert 4 0 9 0 3
coupvert 5 0 8 0 3

finddep

chbias 9 10 2 1.5 1
chbias 12 10 2 1.5 1
```

Shranimo jo kot `simple.tgm`. Makro datoteka in datoteka s točkami `points.dat` sta izpisani v dodatku A. Makro datoteko interpretiramo v grafičnem urejevalniku z naslednjim ukazom: `readmacro simple.tgm`. V urejevalniku se prikaže model na sliki 3.2.

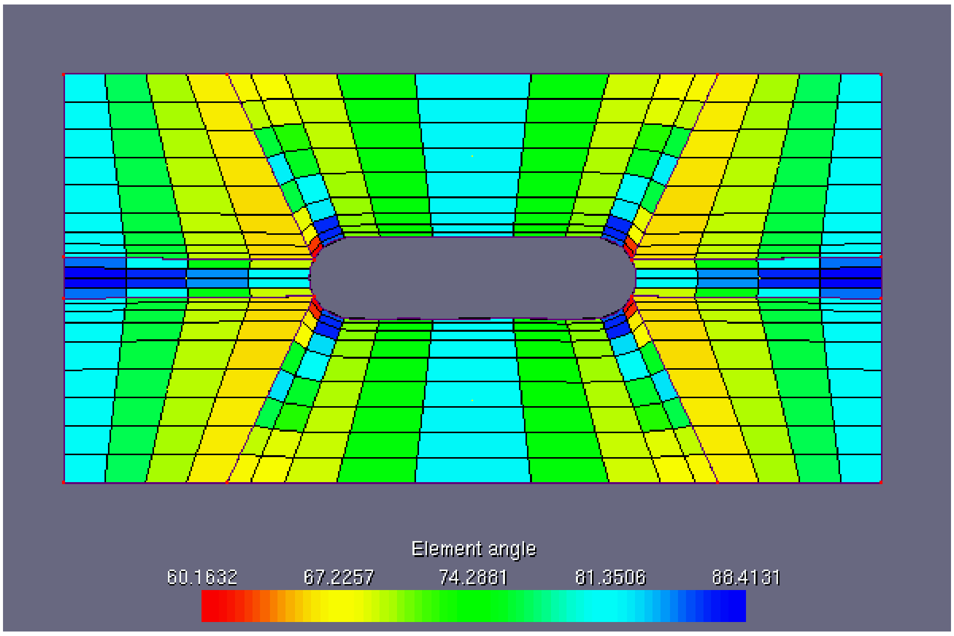
Topološke povezave med robovi nam omogočajo, da s spremembo privzete enakomerne porazdelitve vozlišč mreže na samo dveh robovih (*E_26* in *E_28*) dosežemo porazdelitev mreže na sliki 3.3. Z geometrijskimi povezavami pa bi lahko dosegli popolno simetrično razdelitev domene. V našem primeru sta geometrijsko povezana samo dva para vogalov na zgornjem in spodnjem robu, ogliščni vogali domene pa so nepremični.

3.3 Primeri mrežnih šablon

Do sedaj so bile izdelane tri mrežne šablone, ki so primerne za oblopatična območja turbinskih strojev. Te šablone so *gg_2d*, *gg_turbo1* in *gg_turbo2*. Izdelane so bile kot samostojni C programi. Prva je namenjena ravninskim domenam, drugi dve pa prostorskim.



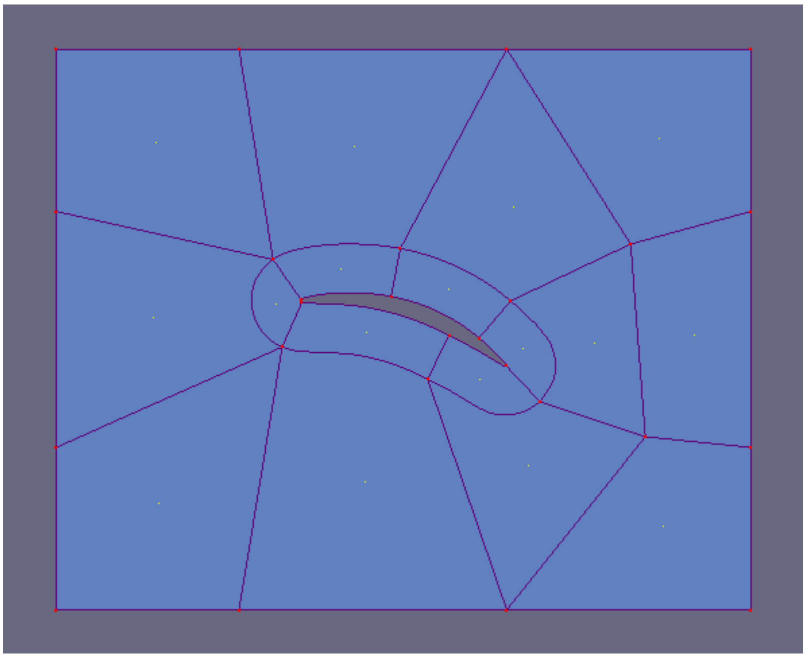
Slika 3.2: Razdelitev enostavne domene generirane z interpretiranjem makro datoteke; izbran je rob *E_26* (bel, z vozlišči), zeleni robovi so njegovi odvisni robovi



Slika 3.3: Porazdelitev mreže, ki jo dosežemo s spremembo porazdelitve na robovih *E_26* in *E_28* (slika 3.2)

3.3.1 2D-oblopatično območje

Šablona *gg_2d* je namenjena ravninskim domenam okrog ene turbinske lopatice. Okrog lopatice generira mrežo, ki je primerna za široko območje naklona lopatice. Model računske domene po šabloni *gg_2d* je prikazan na sliki 3.4, mreža pa je prikazana na sliki 3.5.



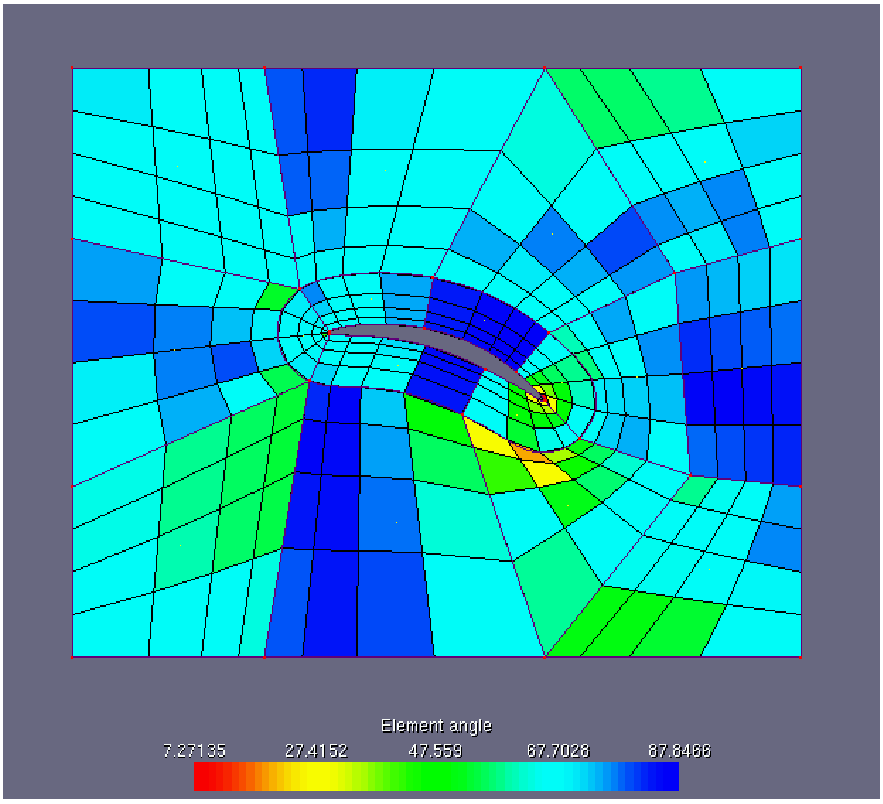
Slika 3.4: Model 2D-oblopatičnega območja generiran z mrežno šablono *gg_2d*

3.3.2 3D-oblopatično območje

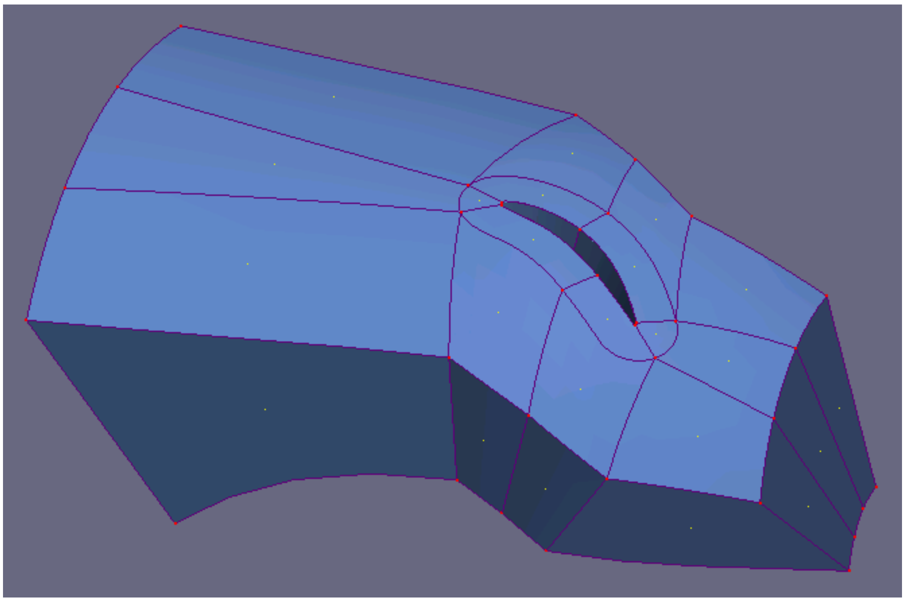
Z ravninskimi domenami se pri analizi turbinskih strojev srečujemo redko. Šablona *gg_turbo1* je namenjena prostorskim domenam okrog lopatic turbin, pri čemer se upošteva tudi periodičnost domene po obodu turbine. Generirana mreža je enostavna in zato primerna za majhne naklone lopatic. Razdelitev računske domene po tej šabloni je prikazana na sliki 3.6, mreža pa na sliki 3.7.

3.3.3 3D-oblopatično območje z velikim vstopnim kotom

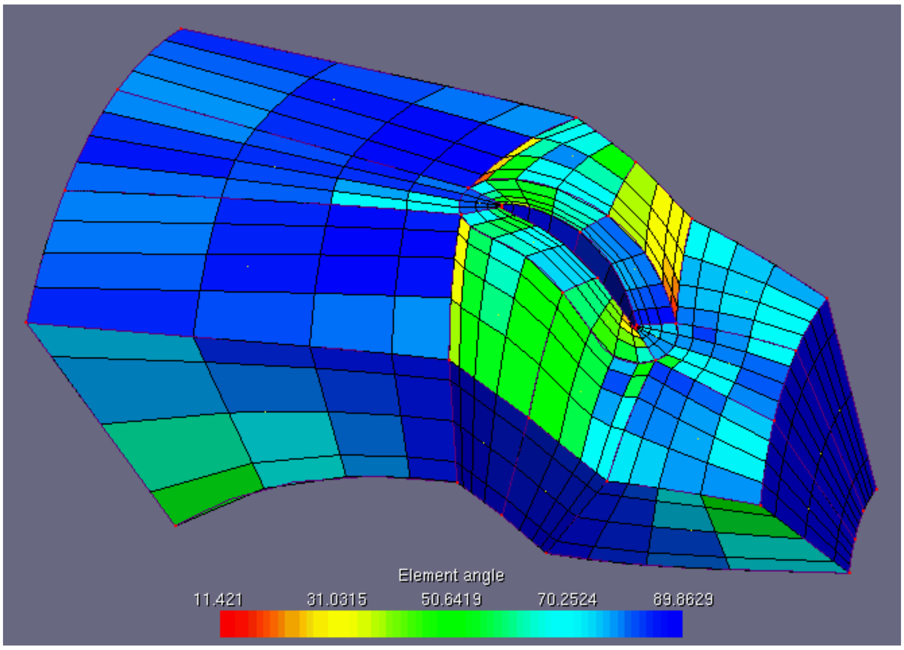
Šablona *gg_turbo2* je modifikacija šablone *gg_turbo1*. Mreža generirana s to šablono je manj občutljiva na spreminjanje naklona lopatice. Lahko jo uporabimo za analizo turbin, ki imajo močno zaprte lopatice, npr. gonilnika vodne turbine. Lahko jo uporabimo tudi kot generator mreže pri analizi zapiranja vodilnika ali podobno, kjer se analiza izvaja avtomatizirano za diskretne vrednosti naklona lopatic. Model računske domene, kot ga generira šablona *gg_turbo2* je prikazan na sliki 3.8. Generirana mreža je prikazana na sliki 3.9.



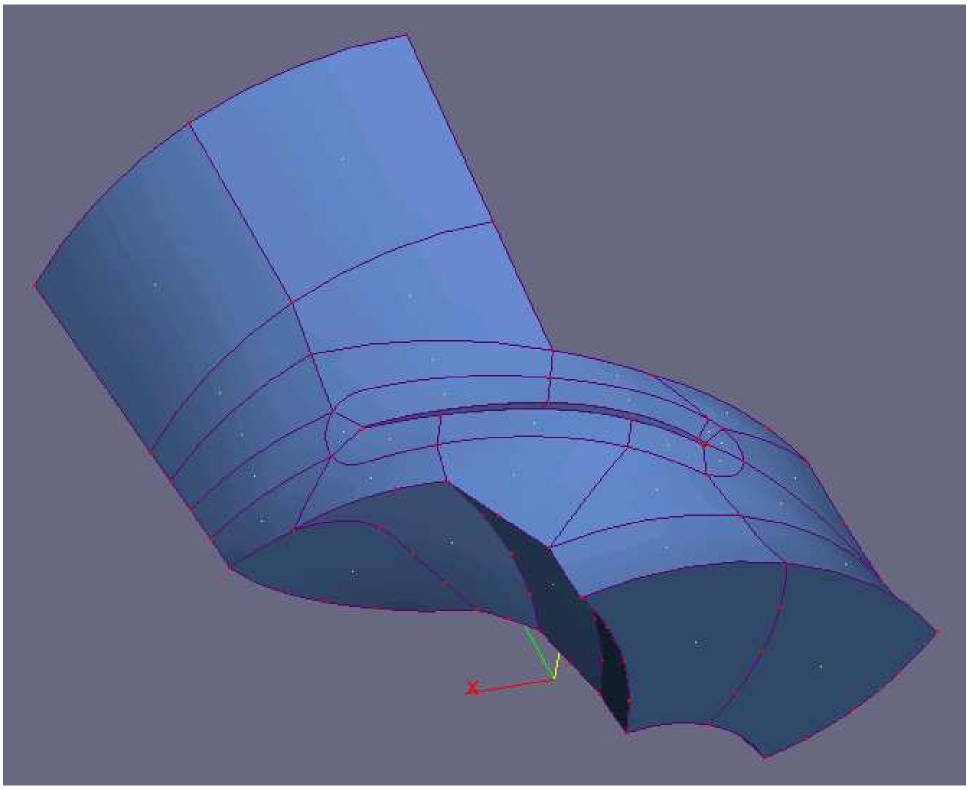
Slika 3.5: Predogled mreže 2D-oblopatičnega območja generirane z mrežno šablono *gg_2d*



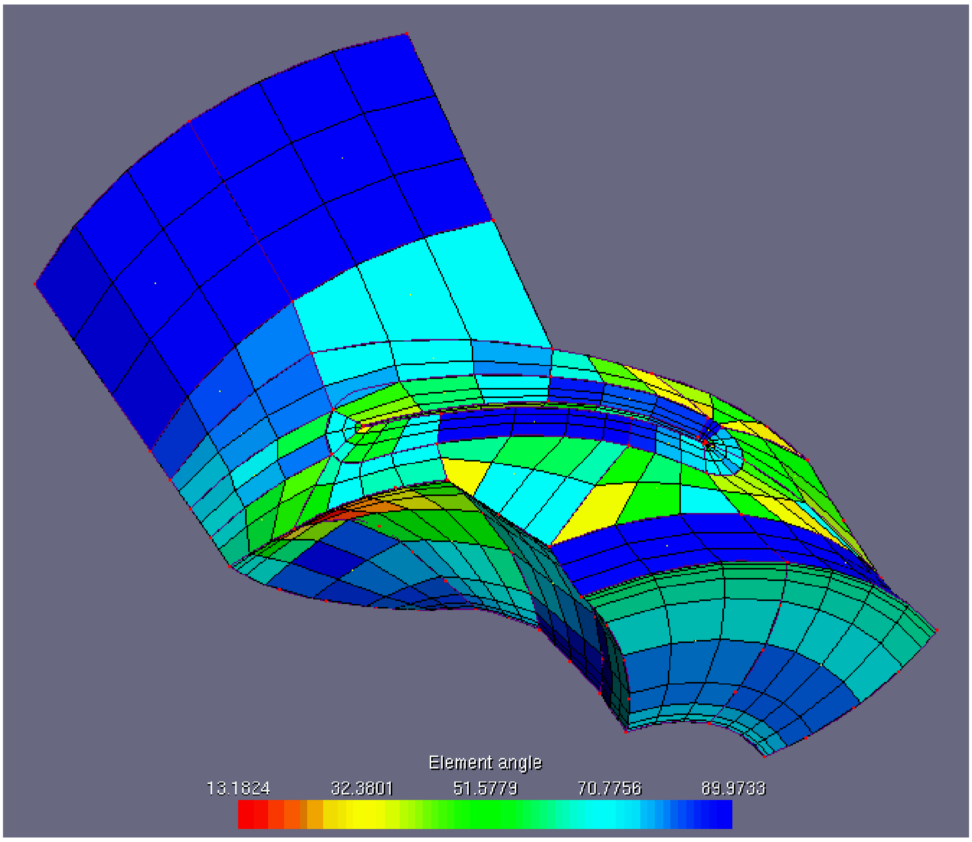
Slika 3.6: Razdelitev domene 3D-oblopatičnega območja po šabloni *gg_turbo1*



Slika 3.7: Predogled mreže 3D-oblopatičnega območja generirane z mrežno šablono *gg_turbo1*



Slika 3.8: Senčen prikaz 3D-oblopatičnega območja generiranega z mrežno šablono *gg_turbo2*



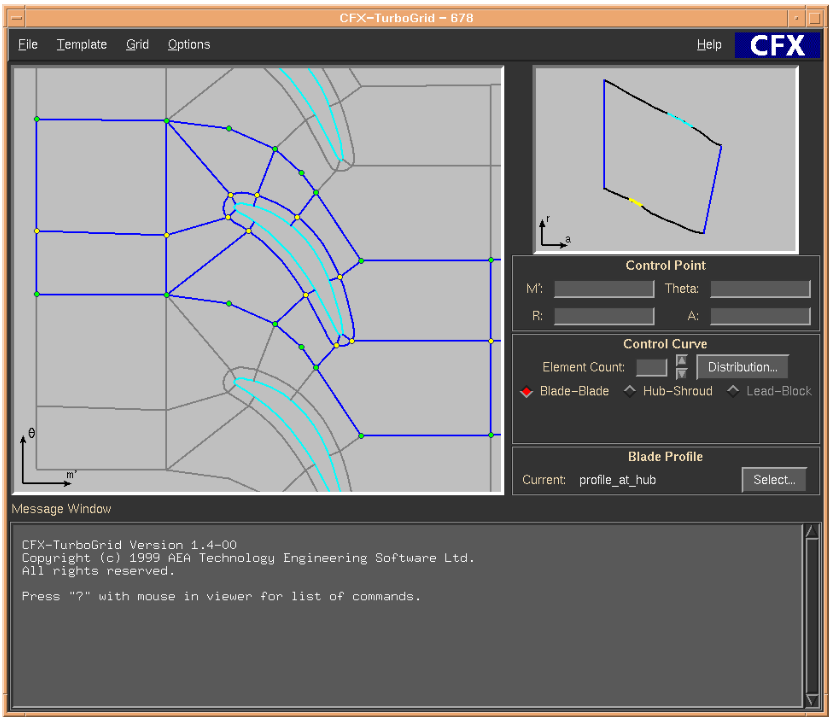
Slika 3.9: Predogled mreže 3D-oblopatičnega območja generirane z mrežno šablono *gg_turbo2*

Poglavje 4

Grafični urejevalnik *GridEdit*

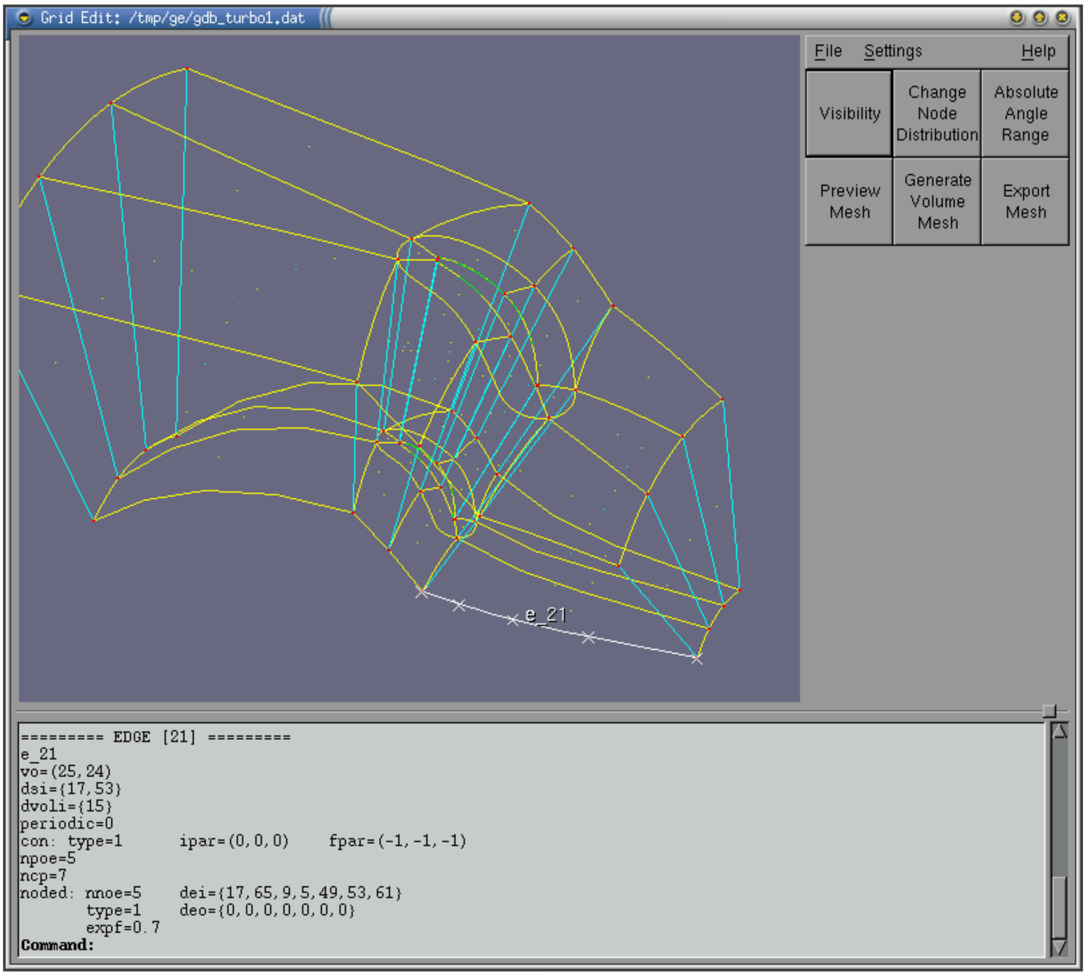
4.1 Predstavitev urejevalnika

Potreba po interaktivnem grafičnem urejevalniku se je pokazala pri uporabi komercialnega produkta *AEA Turbogrid* za generiranje mrež oblopatičnih območij v turbostrojih (slika 4.1). Turbogrid ima kljub bogatemu izboru nastavljivih parametrov kar nekaj pomankljivosti, ena od glavnih je urejanje podobmočij mreženja samo v 2D prikazu, t.j. v določeni presečni ravnini. Ker se spremembe v eni ravnini ne odražajo na ostalih, je potrebno "ciklirati" med ravninami, da je sprememba zvezna. Poleg tega so območja praviloma zavita v prostoru in projekcija na ravnino popači dejansko obliko.



Slika 4.1: Komercialen program za mreženje oblopatičnih območij turbostrojev *AEA Turbogrid*

Omenjene in druge pomanjkljivosti v zvezi z grafično predstavitvijo smo poskusili odpraviti v našem urejevalniku. Urejanje podprostorov v 3D je bolj pregledno in hitrejše, prav tako ni problemov s popačenjem oblike. V povezavi s hitrim predogledom površinske mreže nam daje natančno predstavo o vplivu sprememb na kvaliteto mreže. V veliko pomoč je tudi prikaz porazdelitve vozlišč mreže po robu ali površini, saj nam omogoča takojšnje odkrivanje napak pri določanju porazdelitve. Urejevalnik je bil nadgrajen tudi s tekstovnim vmesnikom z ukazno vrstico, ki je uporaben zlasti pri generiranju geometrijske baze podatkov, omogoča pa tudi prilagoditev mrežne šablone dejanskemu problemu.



Slika 4.2: Grafični urejevalnik *Grid Edit*

4.2 Funkcionalnost urejevalnika

Da bi bil grafični urejevalnik lahko v pomoč uporabniku pri pripravi računske mreže, smo določili naslednje zahteve, ki jim mora zadostiti:

- uporabniški vmesnik z grafičnim oknom, gumbi za proženje akcij in tekstovnim oknom,
- dinamično spreminjanje pogleda v grafičnem oknu (translacija, skaliranje, rotacija),
- premikanje vogalov po robu ali površini s "povleci in spusti",
- izpis podatkov v tekstovno okno o izbranem (kliknjenem) gradniku,
- okno za vnos parametrov porazdelitve vozlišč po izbranem robu,
- prikaz geometrije v senčenem načinu,
- okno za vklapljanje in izklapljanje prikaza različnih elementov v grafičnem oknu, senčenega načina ipd.,
- predogled mreže na izbrani površini ali volumnu,
- predogled mreže celotnega računskega območja naenkrat,
- prikaz kvalitete mreže z barvno skalo,
- ukazna vrstica v tekstovnem vmesniku, ki prepoznava nabor tipkanih ukazov s spremljajočimi parametri.

Poleg teh funkcionalnih zahtev je zaželeno, da je izvorna koda urejevalnika prenosljiva med različnimi operacijskimi sistemi in različnimi strojnimi platformami. Konkretno to pomeni, da lahko urejevalnik poganjamo tako na osebnih računalnikih z MS Windows kot tudi na Unix delovnih postajah. Prav tako je zaželeno, da so morebitne programske knjižnice, ki jih urejevalnik uporablja, pod prosto licenco.

4.3 Izvedba urejevalnika

Grafična aplikacija v splošnem uporablja neko grafično knjižnico, ki ji s svojim naborom funkcij omogoča risanje v okno. Najbolj razširjena taka knjižnica je OpenGL (Open Graphics Library), ki je bila v začetku namenjena samo profesionalnim delovnim postajam, sedaj pa je našla pot tudi v osebne računalnike. Neposredna uporaba te knjižnice je precej zahtevna, zato so se pojavile objektne knjižnice, ki omogočajo hiter in enostaven razvoj programov, v svojem grafičnem jedru pa uporabljajo OpenGL. Ena takih knjižnic je VTK (Visualization ToolKit), ki je namenjena razvoju zahtevnih grafičnih aplikacij za vizualizacijo in obdelavo slik.

Aplikacija, ki ima grafični uporabniški vmesnik[1] (ang. *GUI - Graphical User Interface*), med drugim uporablja neko GUI knjižnico, ki omogoča enostavno uporabo elementov kot so gumbi, menuji, drsniki ipd., s katerimi zgradimo zunanji videz aplikacije. GUI knjižnic je veliko, med bolj priljubljenimi pa je GTK+.

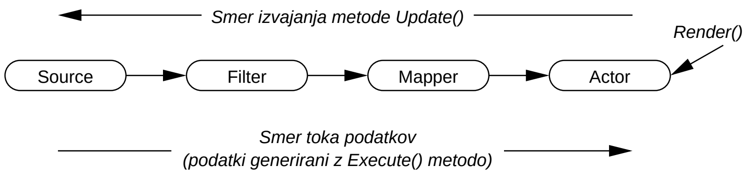
[1] *grafični* v tem primeru pomeni, da aplikacija teče v okenskem okolju—za razliko od tekstovnega vmesnika

Ker je naš urejevalnik grafična aplikacija z grafičnim uporabniškim vmesnikom, uporablja obe vrsti knjižnic in sicer že omenjeni VTK in GTK+. Prva je napisana v programskem jeziku C++, druga pa v jeziku C. Oba jezika je sicer možno uporabljati v istem programu, vendar pa obstaja tudi C++ "ovitek" (ang. *wrapper*) knjižnice GTK+: GTK−−. Slednji omogoča izrabo vseh prednosti jezika C++, ne da bi pri tem morali napisati celo knjižnico znova. Izvorna koda urejevalnika je bila tako v celoti napisana v jeziku C++, ki je poleg jezika C najbolj razširjen programski jezik za pisanje aplikacij. Obe knjižnici, VTK in GTK−−, podpirata tako MS Windows kot Unix sisteme in sta obe pod prosto licenco. VTK ima lastno prosto licenco, GTK−− (in GTK+) pa je pod GNU LGPL verzija 2 (GNU Library General Public License), ki je ena od osnovnih licenc prostega programja.

4.3.1 Grafična knjižnica VTK

VTK je objektna knjižnica zgrajena okrog knjižnice OpenGL. Namenjena je hitremu razvoju zahtevnih grafičnih aplikacij za obdelavo in vizualizacijo podatkov. Kljub svoji obsežnosti in kompleksnosti je njena uporaba relativno enostavna.

V osnovi imamo na voljo nekaj tipov gradnikov, ki jih med seboj povežemo v *grafični cevovod* (slika 4.3). Ti osnovni tipi so *Source*, *Filter*, *Mapper* in *Actor* (izvor, filter, projektor, igralec). Med seboj si podajajo podatkovne objekte in jih povežemo na način `object2->SetInput( object1->GetOutput() )`, torej preko vhodov in izhodov objekta. Grafični cevovod se prične s Source objekti. Ti priskrbijo podatke, ki jih vizualiziramo, tako da jih generirajo ali pa jih preberejo iz datoteke. Filter objekti podatke prečistijo, poenostavijo, jim kaj dodajo ali odvzamejo in nam na ta način omogočijo obdelavo podatkov. Če jih ne potrebujemo lahko Filter objekte tudi kar izpustimo iz grafičnega cevovoda. Mapper objekti (med drugim) predstavljajo geometrijo, ki ponazarja podatke. Pri Actor objektih se cevovod zaključi. Actor objekti združujejo informacije o transformacijah in videzu predmeta, ki ga prikazujejo.



Slika 4.3: Grafični cevovod knjižnice VTK in njegovo izvajanje

Običajno vzpostavimo grafični cevovod za vsak predmet, ki ga prikazujemo. Cevovodi si lahko delijo objekte, lahko jih združujemo ali vejimo. Način izvajanja cevovoda je naslednji: vsak objekt, skozi katerega potujejo podatki, ima notranji zapis zadnjega časa izvajanja in čas zadnje spremembe objekta. Preden se podatki prikažejo na zaslonu, vsak objekt preveri ali se je kaj spremenil od zadnjega časa izvajanja. Če se je, potem podatke še enkrat procesira. Npr., nekemu filtru smo vključili drug način obdelave podatkov. Pred naslednjo osvežitvijo slike na zaslonu, se bo filter ponovno izvedel, prav tako pa tudi vsi objekti do konca cevovoda, ker bodo vsi dobili nove podatke. V tem primeru se edino

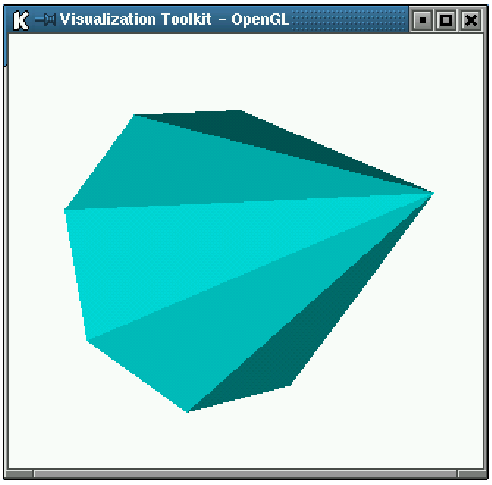
Source objektu ne bo potrebno ponovno izvajati, ker se je sprememba zgodila v cevovodu za njim.

Oglejmo si kratek primer uporabe objektov knjižnice:

```
int main() {
  // ustvarimo risarja, risarsko okno in
  // interaktorja za uporabnika
  vtkRenderer *ren = vtkRenderer::New();
  vtkRenderWindow *renWindow = vtkRenderWindow::New();
    renWindow->AddRenderer( ren );
  vtkRenderWindowInteractor *iren =
                        vtkRenderWindowInteractor::New();
    iren->SetRenderWindow( renWindow );

  // ustvarimo igralca z geometrijo osemstrane piramide
  vtkConeSource *cone = vtkConeSource::New();
    cone->SetResolution( 8 );
  vtkPolyDataMapper *coneMapper = vtkPolyDataMapper::New();
    coneMapper->SetInput( cone->GetOutput() );
  vtkActor *coneActor = vtkActor::New();
    coneActor->SetMapper( coneMapper );
    coneActor->GetProperty()->SetColor( 0,1,1 );

  // risarju povemo katerega igralca naj riše
  ren->AddActor( coneActor );
  ren->SetBackground( 1,1,1 ); // bela barva ozadja

  // uporabniku omogočimo interakcijo z igralcem
  iren->Start();

  // počistimo za seboj
  ren->Delete();
  renWindow->Delete();
  iren->Delete();
  cone->Delete();
  coneMapper->Delete();
  coneActor->Delete();
  return 0;
}
```

Na sredini je lepo vidna zgradba grafičnega cevovoda. V njem ni nobenega Filter objekta, ker ga za prikaz enostavne piramide ne potrebujemo. Ko program prevedemo in poženemo, se nam odpre okno na sliki 4.4.

Z miško lahko piramido interaktivno rotiramo, prestavljamo in povečujemo ali pomanjšujemo. Kakšno interakcijo in na kakšen se bo le-ta izvajala, določa tip interaktorja, ki ga uporabimo v programu. Naš grafični urejevalnik ima podobno zasnovo kot zgornji primer, le da je grafični cevovod precej bolj kompleksen.



Slika 4.4: Primer uporabe VTK knjižnice–osemstrana piramida

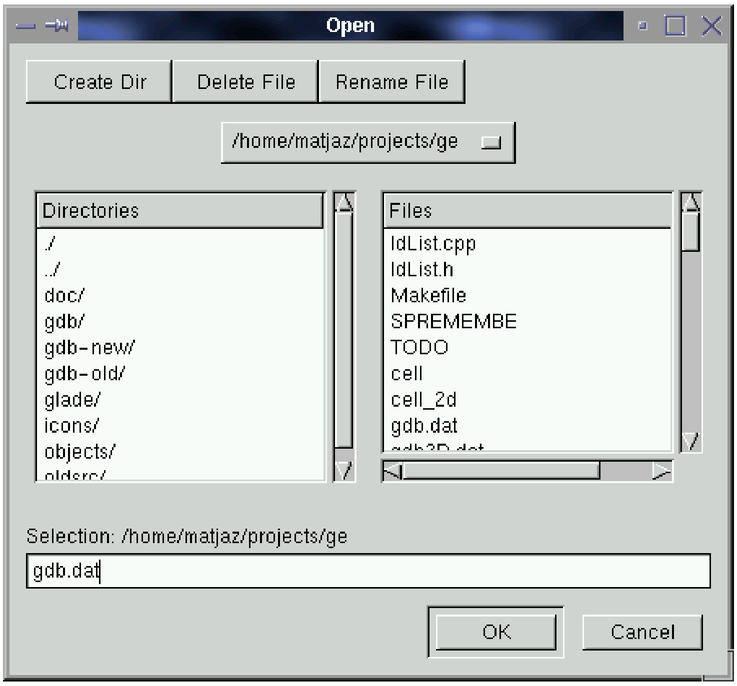
4.3.2 GUI knjižnica GTK−−

GTK−− knjižnica počasi postaja čedalje bolj priljubljena med razvijalci uporabniških vmesnikov. Njeni dobri strani sta velika prilagodljivost in odličen t.i. *signal/slot* (oddajnik/sprejemnik) mehanizem. Slednji omogoča komunikacijo med najrazličnejšimi objekti in funkcijami. Na vsak slot lahko pripeljemo enega ali več signalov, ki so posledica uporabnikovih akcij. Če, npr., kliknemo na nek gumb, potem bo gumb oddal signal, ki bo morebitnemu sprejemniku signala povedal, da je bil gumb pritisnjen. Sprejemnik lahko nato primerno odgovori na uporabnikov klik. Vsakemu objektu, ki ga na novo deklariramo, lahko definiramo poljubno mnogo signalov in/ali slotov. Slaba stran te knjižnice pa je pomanjkljiva dokumentacija.

GUI knjižnice so zbirke objektov, kot so gumbi, drsniki, menuji, dialogi ipd.. Z njimi lahko relativno hitro sestavimo poljuben uporabniški vmesnik. Za primer si poglejmo uporabo objekta `FileDialog`, ki vpraša uporabnika po imenu datoteke:

```
void odpri_datoteko()
{
  string ime_dat; // ime datoteke

  ime_dat  = FileDialog::get_filename( this, "Open", "gdb.dat" );

  if (ime_dat.empty())
    {
    // uporabnik je pritisnil ''Cancel'' ali ''Esc''
    return;
    }

  // pokličemo funkcijo, ki datoteko prebere
  odpri( ime_dat );
}
```

Ko v programu pokličemo zgornjo funkcijo, se nam odpre okno na sliki 4.5. Z miško lahko sedaj izberemo datoteko, ki jo želimo odpreti.



Slika 4.5: Primer uporabe GTK−− knjižnice—FileDialog

4.4 Uporaba urejevalnika

Pri zagonu urejevalnik samodejno naloži datoteko z geometrijsko bazo podatkov `gdb.dat` in jo prikaže. Če datoteke s tem imenom ni v trenutnem imeniku, urejevalnik prikaže samo koordinatno izhodišče, datoteko z bazo pa lahko kasneje naložimo preko menuja *File – Open*.

4.4.1 Spreminjanje pogleda in izbiranje

Pogled na model računskega prostora lahko interaktivno spreminjamo s premikanjem miške in hkratnim držanjem tipk *F1* do *F3*. Pri tem se pogled spreminja proporcionalno premiku miške. Pogled spreminjajo naslednje tipke in kombinacije:

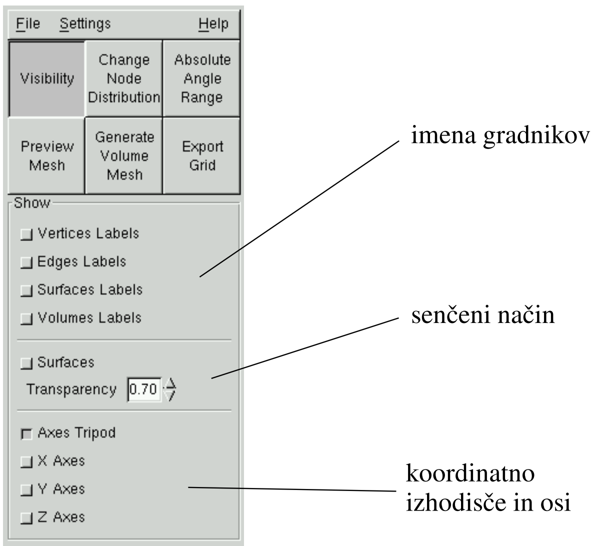
- Tipka **F1** in premik miške: skaliranje
- Tipka **F2** in premik miške: translacija
- Tipka **F3** in premik miške: rotacija
- Tipka **F5**: pogled na ravnino $x - y$
- Tipka **F6**: pogled na ravnino $x - z$

- Tipka **F7**: pogled na ravnino $y - z$
- **Levi gumb** miške: izbiranje gradnikov modela.
- Tipka **r**: resetiranje pogleda, poišče center in popravi skaliranje tako, da so vsi igralci vidni

Vogal ali rob izberemo tako, da nanj kliknemo. Izpiše se njegovo ime, gradnik pa se obarva belo. Če smo izbrali rob, se na njem prikaže tudi porazdelitev vozlišč mreže. Površino ali volumen izberemo s klikom na rumeno oz. svetlo modro piko. ki leži približno v geometrijskem središču gradnika. Pri izbiri gradnika se v tekstovni vmesnik izpišejo njegovi podatki. Pri izbiri vogala ali roba se označijo tudi vsi odvisni vogali oz. robovi. Gradnik odznačimo tako, da kliknemo na ozadje.

4.4.2 Določanje vidnosti elementov

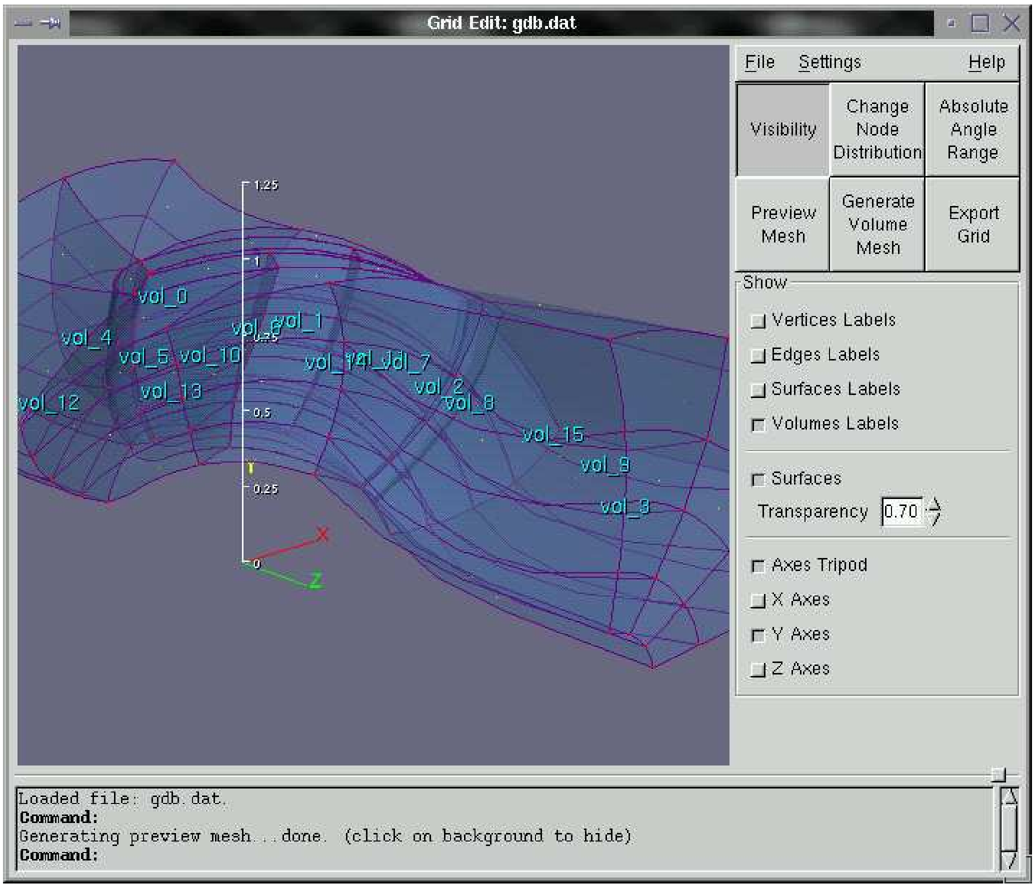
Vidnost elementov grafičnega prikaza določamo v oknu, ki se prikaže ob pritisku na gumb **Visibility**. Okno je prikazano na sliki 4.6 s privzetimi izbirami, ki ustrezajo sliki 4.2. Vgreznjen gumb ob izbiri pomeni, da je element viden. V senčenem načinu lahko izbiramo tudi prosojnost senčenih površin. Na sliki 4.7 je prikazan primer z nekaterimi vklopljenimi elementi.



Slika 4.6: Določanje vidnosti elementov grafičnega prikaza

4.4.3 Predogled mreže

Kadar spreminjamo porazdelitev vozlišč mreže, lahko na hiter in enostaven način ugotovimo, kakšen vpliv imajo spremembe na kvaliteto mreže. To lahko storimo na več načinov:



Slika 4.7: Model v senčenem načinu z vklopljenim prikazom imen volumnov in Y osjo.

- izberemo površino: prikaže se mreža na izbrani površini,
- izberemo volumen: prikaže se mreža na vseh površinah volumna naenkrat,
- zaporedoma izberemo več površin in/ali volumnov pri čemer držimo pritisnjeno tipko *Ctrl* (ponekod *Control*): prikaže se mreža na vsakem od izbranih gradnikov; tako si lahko ogledamo, npr., potek mreže po delu računske domene (slika 4.8),
- kliknemo na gumb *Preview Mesh*: prikaže se mreža na vseh zunanjih površinah računske domene naenkrat.

Pri vsakem predogledu mreže se v spodnjem delu grafičnega okna prikaže barvna lestvica, ki poteka od rdeče do modre barve in ponazarja kvaliteto elementov mreže. Trenutno se kvaliteta mreže ocenjuje po velikosti najmanjšega kota vsakega od elementov mreže. Vsak element mreže je obarvan glede na svoj najmanjši kot, ki se meri v stopinjah. V splošnem se element mreže smatra kot dober, če ima najmanjši kot vrednost med 20 in 90 stopinjami. Pri elementih z najmanjšim kotom manj kot 20 stopinj, lahko pride do poslabšanja natančnosti izračuna. Rdeča barva predstavlja torej slabe elemente. Z gumbom *Absolute Angle Range* preklapljamo med relativno in absolutno lestvico. Dolžina relativne lestvice se sproti prilagaja obsegu najmanjših kotov elementov prikazane mreže in je privzet način prikaza. Absolutna lestvica ima konstanten obseg od 20 do 90 stopinj. Z relativno lestvico lahko opazujemo vpliv sprememb na lokalnem nivoju, z absolutno pa splošno kvaliteto mreže. Na sliki 4.8 je prikazan detajl mreže obarvan z absolutno barvno lestvico. Predogled celotne mreže z relativno barvno lestvico je prikazan na slikah 3.7 in 3.9.

Poudariti je potrebno, da se v splošnem kvaliteta mreže ocenjuje po več kriterijih, npr. po številu elementov, pravilni porazdelitvi vozlišč, razmerju stranic elementov ipd., in ne samo po najmanjšem kotu elementa.

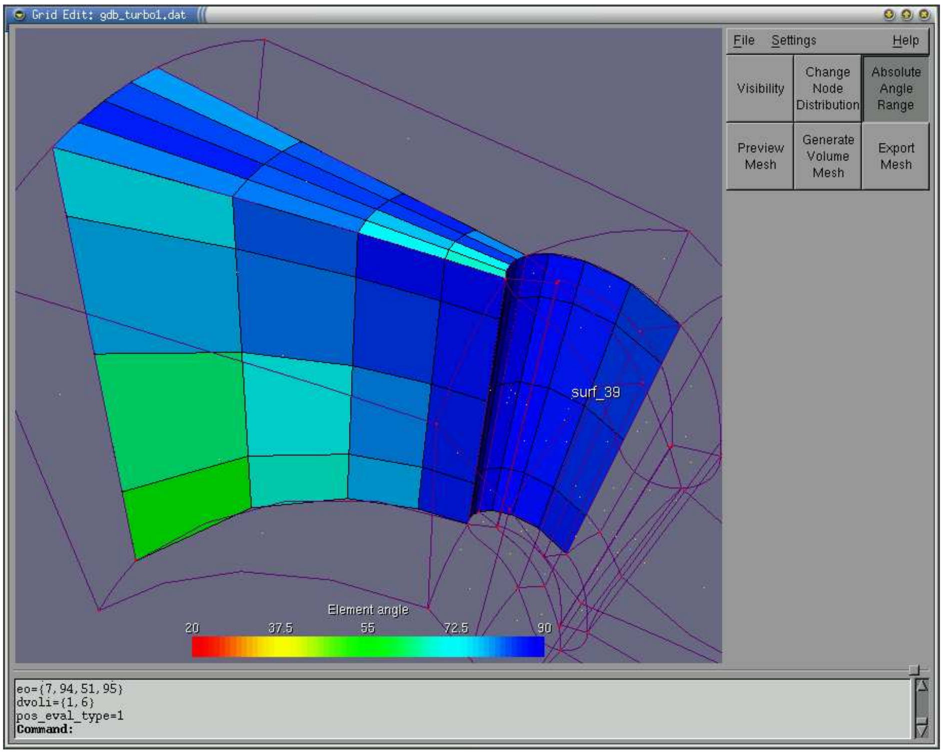
Zaradi hitrejšega odziva se pri predogledu mreže vedno izračunajo samo vozlišča po površinah. Za generiranje elementov tudi po volumnu moramo pritisniti gumb *Generate Volume Mesh*, ki izračuna vozlišča po celotni domeni. Prikaz mreže in lestvice izklopimo s klikom na ozadje ali z izbiro vogala ali roba.

4.4.4 Spreminjanje porazdelitve vozlišč

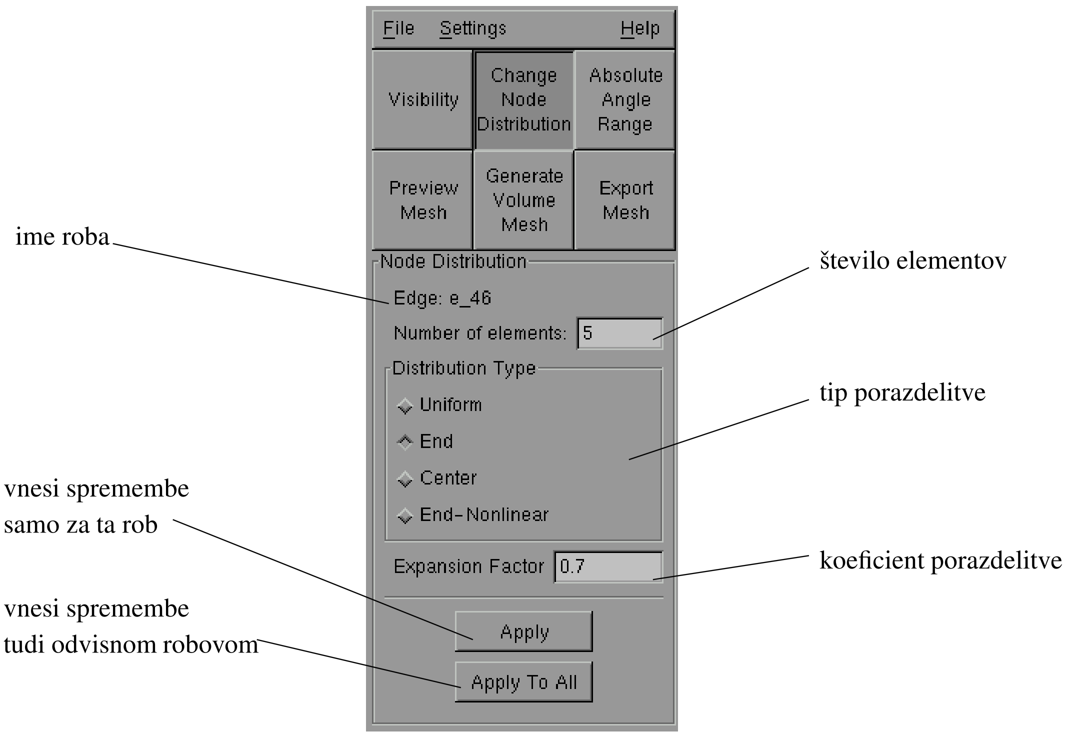
Z gumbom *Change Node Distribution* odpremo okno za spreminjanje števila in porazdelitve vozlišč mreže po robu (slika 4.9). Vrednosti nastavitev v oknu vedno odražajo porazdelitev trenutno izbranega roba, katerega ime je izpisano na vrhu okna. Z gumbom *Apply* se spremembe porazdelitve zapišejo v bazo samo za izbran rob, z gumbom *Apply To All* pa tudi za vse njegove odvisne robove. Sprememba števila vozlišč se vedno odrazi tudi na vseh odvisnih robovih. Sprememba baze se takoj odrazi tudi v grafičnem oknu. Vozlišča se na izbranem robu prikažejo z belimi križci (slika 3.2).

4.4.5 Premikanje vogalov

Kot smo že omenili, lahko spreminjamo obliko blokov mreže s premikanjem vogalov. Vogal premaknemo tako, da ga najprej označimo, nato pa s pritisnjenim levim gumbom

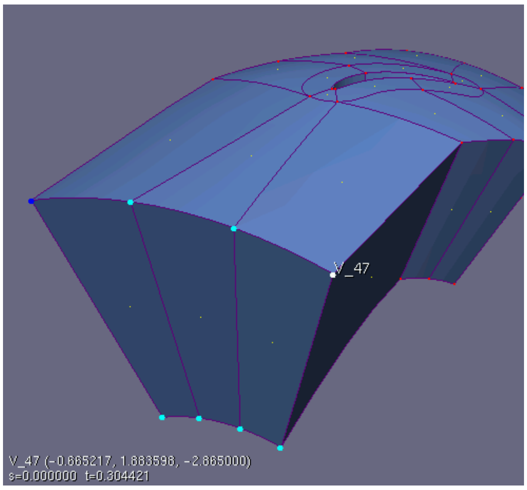


Slika 4.8: Sestavljen predogled mreže volumna in površine, prikazan z absolutno barvno lestvico

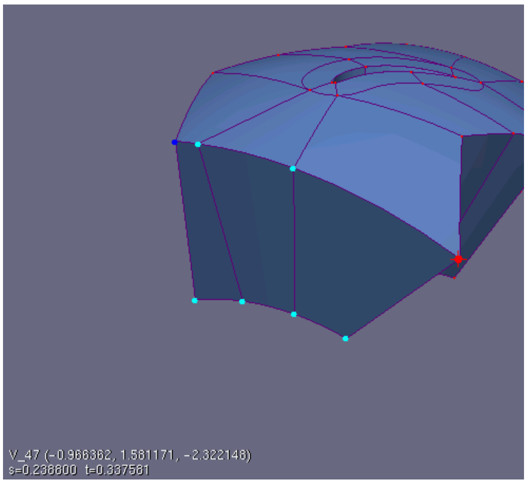


Slika 4.9: Določanje porazdelitve vozlišč mreže po izbranem robu

miške prestavimo drugam (t.j. povleci in spusti). Med premikanjem je vogal označen z rdečim krogcem in križem, v levi spodnji kot grafičnega okna pa se sproti izpisujejo kartezične in parametrične koordinate. Ker gre načeloma za drugo 3D lokacijo, kazalec miške pa lahko prestavimo le na 2D zaslonu, je 3D lokacija definirana s projekcijo zaslonske točke na 3D geometrijo. Pri premikanju se upoštevajo vsi geometrijski pogoji vogala, hkrati se upoštevajo tudi vse njegove topološke odvisnosti. Izbrani vogal je obarvan belo, njegovi odvisni vogali pa so obarvani glede na tip topološke odvisnosti. Geometrija se v grafičnem oknu popravlja v realnem času. Sliki 4.10 in 4.11 prikazujeta primer premikanja vogala *V_47* na katerega so vezani vsi ostali vogali na čelni površini. Temno moder vogal ima več topoloških povezav z vogalom *V_47* (sledi mu tako po s kot po t koordinati).



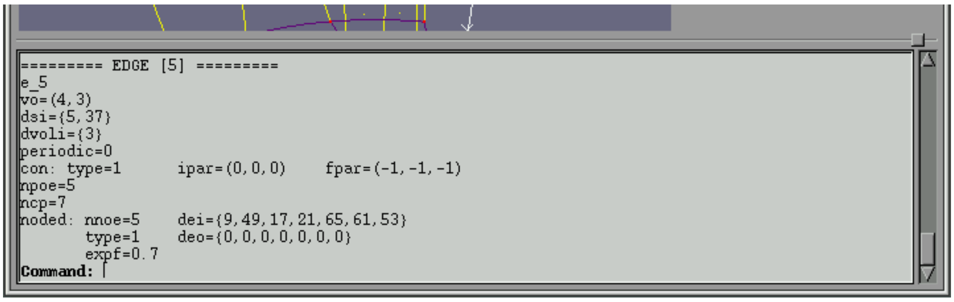
Slika 4.10: Izbrani vogal *V_47* in njegovi odvisni vogali. Koordinati s in t izbranega vogala predstavljata parametra zgornje površine, po kateri se vogal lahko premika.



Slika 4.11: Premikanje vogala s slike 4.10

4.4.6 Tekstovno okno

V tekstovni vmesnik se pri izbiranju gradnikov izpisujejo vsi njegovi glavni podatki. To so poleg imena in indeksa v bazi predvsem geometrijski pogoji in topološke odvisnosti. Pri robu se izpišejo tudi parametri porazdelitve vozlišč, pri površini in volumnu pa tudi podatki o mreži. Trenutno se podatki o gradnikih izpisujejo v obliki primerni za razhroščevanje baze ter urejevalnika in zato še niso v veliko pomoč običajnemu uporabniku. Poleg podatkov o izbranih gradnikih se v okno izpisujejo tudi nekatera obvestila v zvezi z generiranjem mrež, shranjevanjem baze ipd.. Primer izpisa informacij o izbranem robu je prikazan na sliki 4.12.



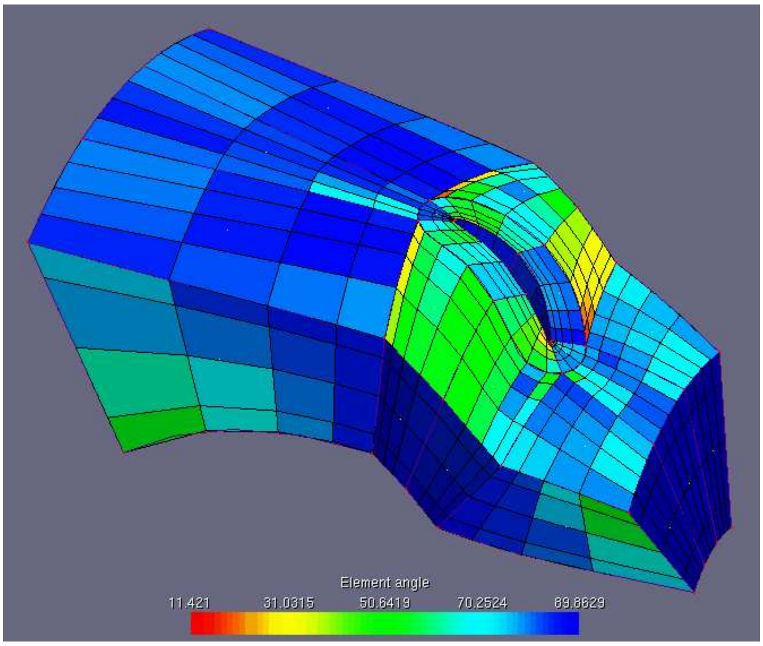
Slika 4.12: Primer izpisa podatkov o gradniku. Spodnja vrstica deluje kot ukazna vrstica.

Spodnja vrstica tekstovnega okna deluje kot ukazna vrstica, v katero lahko tipkamo ukaze. Uporablja se v glavnem za generiranje ali popravljanje mrežne šablone oz. baze. Ukazi, ki jih urejevalnik pozna, so opisani v poglavju 3.2.1 na strani 25.

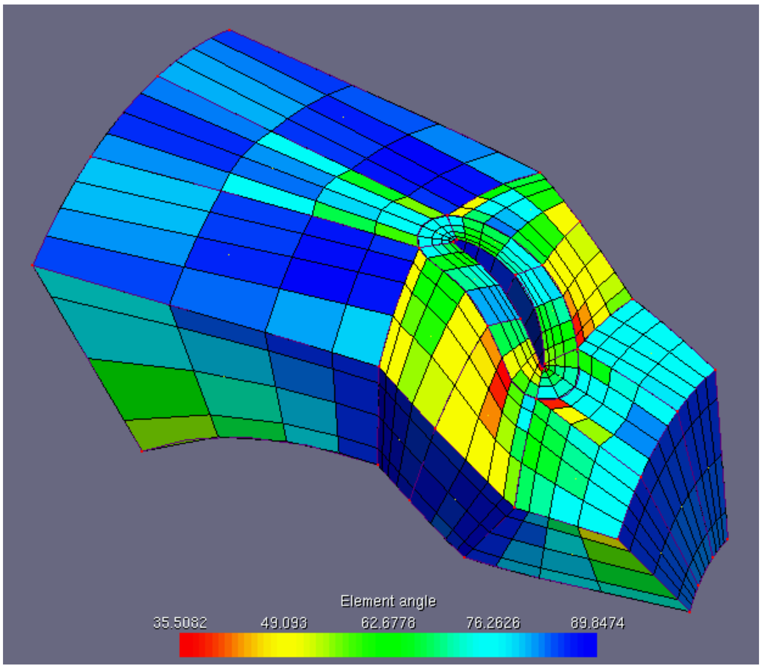
4.5 Primer uporabe

Z mrežnimi šablonami lahko dosežemo zelo dobro začetno mrežo. Vendar pa je praktično nemogoče izdelati šablono, ki bi generirala samo dobre elemente in bila pri tem univerzalna. Šablona *gg_turbo1*, ki je opisana v poglavju 3.3.2, je tipičen primer. V splošnem generira dobro mrežo, vendar z nekaj izredno slabimi elementi. Na sliki 3.7 vidimo, da je ima najslabši element kot okrog 11 stopinj. Z nekaj dobro premišljenimi premiki vogalov lahko mrežo popravimo tako, da znaša najmanjši kot približno 35 stopinj in jo s tem bistveno izboljšamo. Kvaliteto mreže lahko še naprej izboljšujemo s spreminjanjem porazdelitve vozlišč, za kar pa so potrebna določena znanja in izkušnje.

Da bi lahko uporabili šablono kot generator mreže za avtomatizirano izvajanje analiz z diskretnimi spremembami geometrije modela (npr. zapiranje vodilnika turbine), mora le-ta generirati mrežo brez slabih elementov. To lahko dosežemo s prirejanjem ustrezne šablone za konkreten primer. Pri tem lahko urejevalnik uporabljamo samo kot pregledovalnik mreže ali pa kot orodje za interaktivno spreminjanje baze in s tem ugotavljanje potrebnih popravkov šablone. Slednje lahko potem zapišemo v makro datoteko, ki jo zaženemo po vsakem generiranju baze.



Slika 4.13: Predogled mreže, kot jo generira šablona *gg_turbo1*



Slika 4.14: V primerjavi s sliko 4.13 smo z nekaj spremembami odpravili slabe elemente

Poglavje 5

Zaključek

Glavni cilj diplomske naloge je bil povečanje funkcionalnosti grafičnega urejevalnika. Dodan mu je bil predogled generirane mreže, barvanje elementov mreže glede na njihov najmanjši kot, razširjen je bil nabor nastavljivih parametrov v oknu *Change Node Distribution*, največje spremembe pa je bil deležen tekstovni vmesnik. Vmesniku je bila dodana ukazna vrstica, ki prepoznava nabor tipkanih ukazov z argumenti. Slednji se prevedejo v klice funkcij geometrijske baze podatkov, ki operirajo nad gradniki baze. Z njimi lahko izdelujemo nove ali pa popravljamo oz. prilagajamo že obstoječe mrežne šablone. Namesto tipkanja ukazov v ukazno vrstico je bil razvit princip makro datoteke, ki nam omogoča večkratno uporabo in enostavno popravljanje mrežne šablone. Izdelani so bili štirje primeri mrežnih šablon, od najenostavnejše v obliki makro datoteke do najzahtevnejše v obliki C programa. S predvidenim samostojnim interpreterjem bo možno uporabiti makro datoteke tudi brez izvajanja v grafičnem urejevalniku, torej kot samostojen generator baze.

Predmet diplomske naloge je bila tudi poglobitev v algoritem mreženja na osnovi eliptične interpolacije. Izkazalo se je, da je ta tema preobsežna za tako zastavljeno diplomsko nalogo, saj v času, ki sem ga imel na voljo, nisem prišel do uporabnih rezultatov. Problem se je začel že pri iskanju literature, saj v slovenskih knjižnicah ni moč dobiti niti enega izvoda knjige *Numerical Grid Generation, Foundations and Applications*, avtorjev J.F. Thompson in drugih, ki je praktično obvezna literatura na področju mreženja. Dobil sem sicer starejši izvod, ki je objavljen na internetu, vendar za resnejše delo ni najbolj primeren (strani niso oštevilčene, slike so slabe ipd.).

Za res udobno delo z mrežami je nujen nadaljni razvoj grafičnega urejevalnika in geometrijske baze podatkov. Najbolj potrebno je dodati preverjanje kakovosti elementov volumske mreže, razširiti kriterije ocenjevanja elementov (ortogonalnost, razmerje stranic ipd.) in dodati histogram kvalitete elementov. S slednjim bi dobili oceno kvalitete elementov celotne mreže in ne samo najboljšega in najslabšega elementa.

Nabor makro ukazov za generiranje baze oz. mrežne šablone je v sedanjem stanju sicer zadosten, vendar bomo hitro naleteli na primer, kjer bomo morali operirati tudi z najosnovnejšimi elementi, t.j. točkami in krivuljami. Dobro bi bilo torej dodati tudi nekatere ukaze, ki kličejo ustrezne nizkonivojske funkcije baze.

V poročilu o diplomski nalogi sem se izogibal podrobnejšega pojasnjevanja implementacije predstavljenih lastnosti grafičnega urejevalnika, saj bi poročilo postalo preobsežno, implementacija pa bi bila najverjetneje jasna samo programerjem. Izjemi sta samo poglavja 4.3.1 in 4.3.2, ki predstavita princip uporabe objektnega programiranja za razvoj aplikacij kot je naš grafični urejevalnik. Oba primera kode bi morala biti razumljiva vsakemu, ki zna vsaj malo angleščine.

Literatura

[1] Gerald E. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, Inc., četrta izdaja, 1996.

[2] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison Wesley Publishing Company, Inc., druga izdaja, 1994.

[3] John R. Levine, Tony Mason, in Doug Brown. *Lex & Yacc*. O'Reilly & Associates, Inc., druge izdaja, 1992.

[4] Michael E. Mortenson. *Geometric Modeling*. John Wiley & Sons, Inc., prva izdaja, 1985.

[5] Anthony Porter. *The Best C/C++ Tips Ever*. Osborne McGraw-Hill, 1993.

[6] Allan H. Watt. *Fundamentals of Three-Dimensional Computer Graphics*. Addison Wesley Publishing Company, Inc., prva izdaja, 1989.

Izjava

Diplomsko nalogo sem samostojno izdelal pod vodstvom mentorja izr. prof. dr. Jožeta Duhovnika.

Matjaž Šubelj

Ljubljana, dne 12.3.2001

Dodatek A

Generiranje baze z makro datoteko

A.1 Makro datoteka *simple.tgm*

Primer mrežne šablone v poglavju 3.2.2 je izdelan z makro datoteko `simple.tgm`, Makro datoteke zaženemo v grafičnem urejevalniku z ukazom `readmacro` *`datoteka`*. Vsebina datoteke `simple.tgm` je naslednja:

```
# TGMacro File 0.1
# simple example

initgdb
settol 0.001

# vogali nosilne povrsine
vert 0 0 0 0
vert 1 2 0 0
vert 2 2 1 0
vert 3 0 1 0

# nosilna povrsina
surfbi 0 0 1 2 3
surfoff 0

# nosilni robovi
edge0 0 -1-1 "points.dat" 0
edge2 1 0 1 10
edge2 2 1 2 10
edge2 3 2 3 10
edge2 4 3 0 10

edgeoff 0
edgeoff 1
edgeoff 2
edgeoff 3
```

```
edgeoff 4

# vidni gradniki baze
vertoe 4 1 0.2
vertoe 5 1 0.8
vertoe 6 2 0.4
vertoe 7 2 0.6
vertoe 8 3 0.2
vertoe 9 3 0.8
vertoe 10 4 0.4
vertoe 11 4 0.6

att2surf 4 0
att2surf 5 0
att2surf 8 0
att2surf 9 0

vert 12 0.5 0.4 0
vert 13 1.5 0.4 0
vert 14 1.5 0.6 0
vert 15 0.5 0.6 0

att2edge 12 0
att2edge 13 0
att2edge 14 0
att2edge 15 0

edge2 5 0 4 10
edge2 6 4 5 10
edge2 7 5 1 10
edge2 8 11 12 10
edge4 9 12 13 30 0
edge2 10 13 6 10
edge2 11 10 15 10
edge4 12 15 14 30 0
edge2 13 14 7 10
edge2 14 3 9 10
edge2 15 9 8 10
edge2 16 8 2 10

edge2 17 0 11 10
edge2 18 11 10 10
edge2 19 10 3 10
edge2 20 4 12 10
edge4 21 12 15 30 0
edge2 22 15 9 10
edge2 23 5 13 10
edge4 24 13 14 30 0
```

```
edge2 25 14 8 10
edge2 26 1 6 10
edge2 27 6 7 10
edge2 28 7 2 10

surfi 1 0 4 12 11 5 20 8 17
surfi 2 4 5 13 12 6 23 9 20
surfi 3 5 1 6 13 7 26 10 23
surfi 4 11 12 15 10 8 21 11 18
surfi 5 6 7 14 13 27 13 24 10
surfi 6 10 15 9 3 11 22 14 19
surfi 7 15 14 8 9 12 25 15 22
surfi 8 14 7 2 8 13 28 16 25

coupvert 4 0 9 0 3
coupvert 5 0 8 0 3

finddep

chbias 9 10 2 1.5 1
chbias 12 10 2 1.5 1
```

A.2 Datoteka s podatki

Podatki v datotekah so praviloma kartezične koordinate točk v prostoru. Koordinate ene točke so podane v eni vrstici v zaporedju *x y z*. Med posameznimi koordinatami mora biti vsaj en presledek ali tabulator. Znak # tudi tu označuje komentirano vrstico. Datoteka `points.dat`, ki je bila uporabljena v primeru v poglavju 3.2.2 je naslednja:

```
# x            y
1.4            0.5
1.396592583    0.525881905
1.38660254     0.55
1.370710678    0.570710678
1.35           0.58660254
1.325881905    0.596592583
1.3            0.6
1.2            0.6
1.1            0.6
1.0            0.6
0.9            0.6
0.8            0.6
0.7            0.6
0.674118095    0.596592583
0.65           0.58660254
0.629289322    0.570710678
```

0.61339746	0.55
0.603407417	0.525881905
0.6	0.5
0.603407417	0.474118095
0.61339746	0.45
0.629289322	0.429289322
0.65	0.41339746
0.674118095	0.403407417
0.7	0.4
0.8	0.4
0.9	0.4
1.0	0.4
1.2	0.4
1.3	0.4
1.325881905	0.403407417
1.35	0.41339746
1.370710678	0.429289322
1.38660254	0.45
1.396592583	0.474118095
1.4	0.5