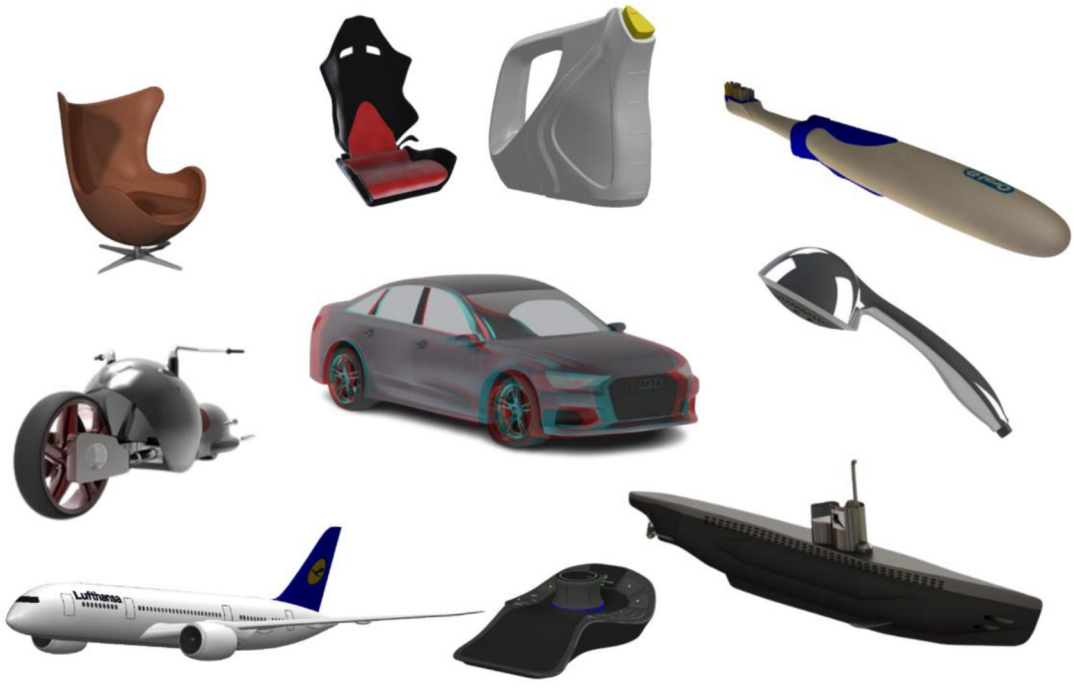
Univerzitetna založba
Univerze v Mariboru

3D modeliranje

Napredne tehnike in aplikacije v računalniško podprtem modeliranju



Avtor:

Miran Ulbin

Maribor, 2024

Univerza v Mariboru

Fakulteta za strojništvo

3D modeliranje

Napredne tehnike in aplikacije v računalniško podprtem modeliranju

Avtor

Miran Ulbin

November 2024

Naslov **3D modeliranje**
Title *3D Modeling*

Podnaslov **Napredne tehnike in aplikacije v računalniško podprtem modeliranju**
Subtitle *Advanced Techniques and Applications in Computer-Aided Design (CAD)*

Avtor Miran Ulbin
Author (Univerza v Mariboru, Fakulteta za strojništvo)

Recenzija Gregor Harih
Review (Univerza v Mariboru, Fakulteta za strojništvo)

Matej Borovinšek
(Univerza v Mariboru, Fakulteta za strojništvo)

Lektoriranje Špela Vidmar
Language editing (AMIDAS d.o.o.)

Tehnična urednika Marina Bajić
Technical editors (Univerza v Mariboru, Univerzitetna založba)

Jan Perša
(Univerza v Mariboru, Univerzitetna založba)

Oblikovanje ovitka Miran Ulbin
Cover designer (Univerza v Mariboru, Fakulteta za strojništvo)

Grafike na ovitku Miran Ulbin, 2024
Cover graphics

Grafične priloge Vsi viri so lastni, če ni navedeno drugače.
Graphic material Ulbin (avtor), 2024

Založnik **Univerza v Mariboru**
Published by **Univerzitetna založba**
Slomškov trg 15, 2000 Maribor, Slovenija
https://press.um.si, zalozba@um.si

Izdajatelj **Univerza v Mariboru**
Issued by **Fakulteta za strojništvo**
Smetanova ulica 17, 2000 Maribor, Slovenija
https://fs.um.si, fs@um.si

Izdaja Prva izdaja
Edition

Vrsta publiakcije E-knjiga
Publication type

Izdano Maribor, Slovenija, november 2024
Published at

Dostopno na https://press.um.si/index.php/ump/catalog/book/7fs24
Available at

CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

004.925(0.034.2)

ULBIN, Miran
3D modeliranje [Elektronski vir] : napredne tehnike in aplikacije v računalniško podprtem modeliranju / avtor Miran Ulbin. - E-publikacija

Način dostopa (URL): https://press.um.si/index.php/ump/catalog/book/7fs24
ISBN 978-961-286-930-4 (Pdf)
doi: 10.18690/um.fs.7.2024
COBISS.SI-ID 215433731

ISBN 978-961-286-930-4 (pdf)

DOI https://doi.org/10.18690/um.fs.7.2024

Cena Brezplačni izvod

Odgovorna oseba založnika Prof. dr. Zdravko Kačič
For publisher rektor Univerze v Mariboru

Citiranje Ulbin, M. (2024). *3D modeliranje: Napredne tehnike in aplikacije v računalniško podprtem*
Attribution *modeliranju.* Univerza v Mariboru, Univerzitetna založba. doi: 10.18690/um.fs.7.2024

Kazalo vsebine

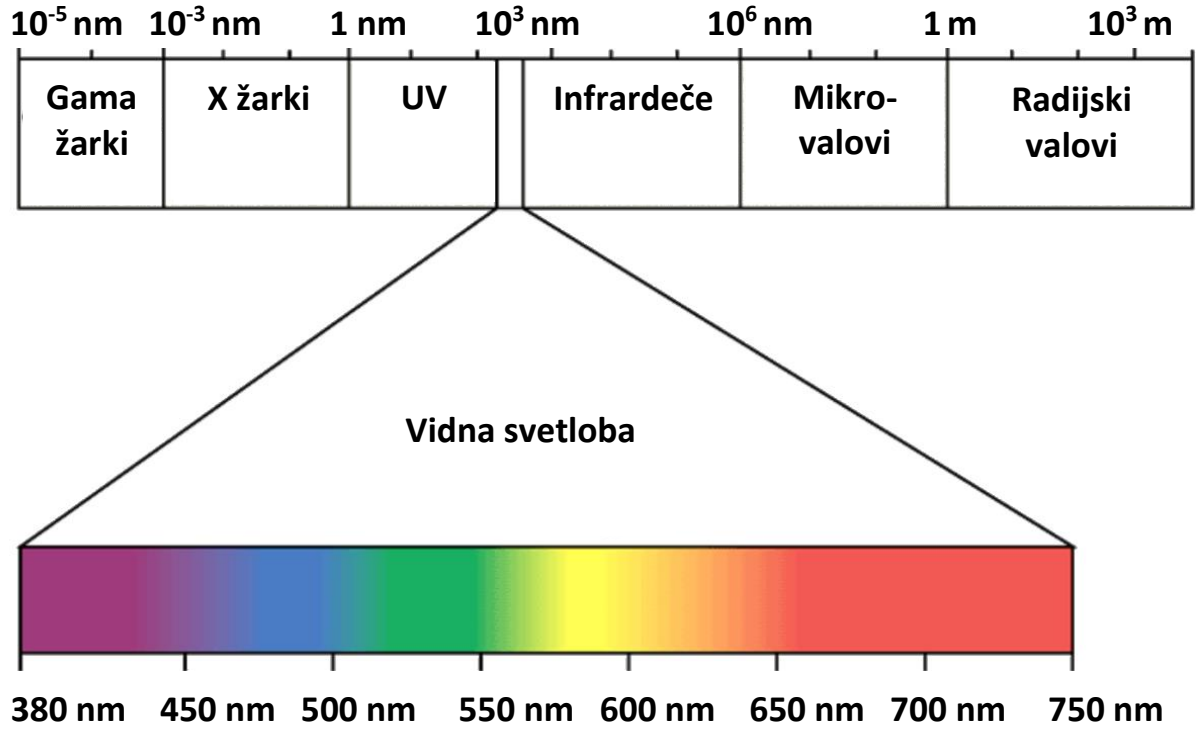
Uporabljene kratice

3D	3 Dimensional
3DXML	3 Dimensional eXtended Markup Language
ACIS	Alan Charles Ian System
AR	Augmented Reality
ASCII	American Standard Code for Information Interchange
AVI	Audio Video Interleave
B-rep	Boundary representation
CAD	Computer Aided Design
CATIA	Computer Aided Three-dimensional Interactive Application
CAVE	Cave Automatic Virtual Environment
CD	Compact Disc
CGI	Computer-Generated Imagery
CMYK	Cyan Magenta Yellow blacK
COLLADA	COLLAborative Design Activity
CSG	Constructive Solid Geometry
CT	Computed Tomography
DAE	Data Asset Exchange
DCI	Digital Cinema Initiative
DIC	Digital Image Correlation
DLP	Digital Light Processing
DVD	Digital Versatile Disc
DXF	Drawing eXchange Format
EOF	End Of File
FFmpeg	Fast Forward Moving Picture Experts Group
GKS	Graphical Kernel System
GPS	Global Positioning System
GPT	Generative Pre-training Transformer
HD	High Definition
HMD	Head Mounted Display
HTML	Hyper Text Markup Language
IBM	International Business Machines Corporation
ISO	International Organization for Standardization
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
LIDAR	Light Detection And Ranging
MIT	Massachusetts Institute of Technology
MKV	Matroska Video files
MOV	MOVie
MP4	Motion Picture Experts Group 4

MPEG	Motion Picture Experts Group
MRI	Magnetic Resonance Imaging
NASA	National Aeronautics and Space Administration
NTSC	National Television System Committee
NURBS	Non-Uniform Rational B-Spline
NX	NeXt Generation
OBJ	OBJect file format
OLED	Organic Light-Emitting Diode
OpenGL	Open Graphics Library
PAL	Phase-Alternating Line
PHIGS	Programmer's Hierarchical Interactive Graphics System
PLY	PoLYgon file format
PTC	Parametric Technology Corporation
QLED	Quantum Dot Light-Emitting Diode
QXGA	Quad EXtended Graphics Array
RGB	Red Green Blue
RGBA	Red Green Blue Alpha
RTX	Ray Tracing Texel eXtreme
SIGGRAPH	Special Interest Group on Computer Graphics and Interactive Techniques
STL	STereoLitography
SXGA	Super EXtended Graphics Array
TFLOPS	Trillion FLoating-point Operations per Second
UHD	Ultra High Definition
USAF SAGE	United States Air Force Semi-Automatic Ground Environment
USB-C	Universal Serial Bus C
UXGA	Ultra Extended Graphics Array
VGA	Video Graphics Array
VR	Virtual Reality
VRML	Virtual Reality Markup Language
VSDC	VideoSoftDevCom
WSXGA	Wide Super Extended Graphics Array
WUXGA	Wide Ultra Extended Graphics Array
WXGA	Wide Extended Graphics Array
XGA	EXtended Graphics Array
ZDA	Združene Države Amerike

1 Uvod

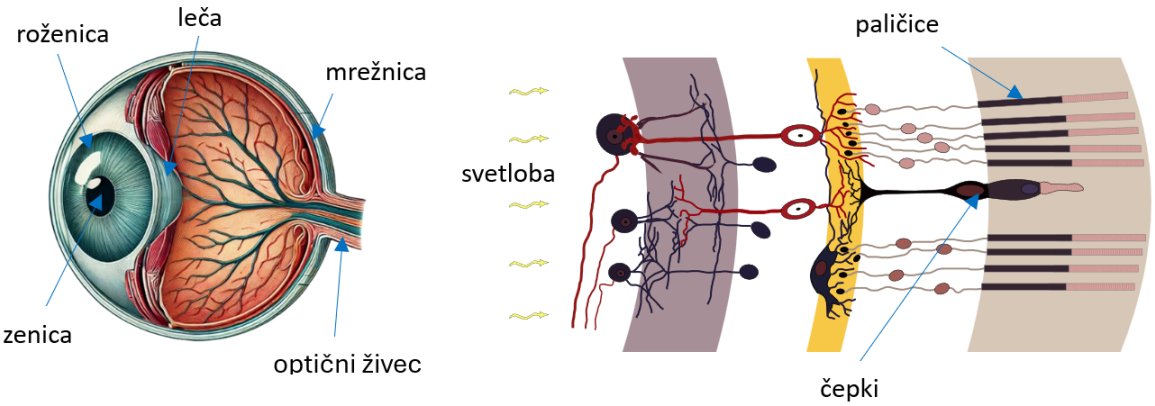
3D modeliranje je postopek, s katerim želimo modelirati realnost. Zato je treba najprej ugotoviti, kako jo sploh dojemamo. Glavni organ za dojemanje realnosti je oko, ki je senzor za vidno svetlobo. Oko omogoča vizualizacijo sveta z zajemanjem vidne svetlobe. Spekter vidne svetlobe je precej majhen, kot vidimo na sliki 1.1. To pomeni, da modeliramo relativno majhen del realnega sveta. To seveda precej olajša delo in zahteva neprimerno manj zahtevno strojno opremo, vendar se je treba že v tej točki zavedati, da bomo modelirali realnost z veliko napako.



Slika 1.1. Vidna svetloba

V nadaljevanju bomo pri modeliranju uporabili še veliko takšnih poenostavitev, zato bo 3D model le idealizirana podoba realnega sveta. Namen 3D modeliranja je izdelati takšen model, ki bo sprejemljiv za naše dojemanje sveta. Zato je treba poznati vse značilnosti našega dojemanja sveta.

Na sliki 1.2 sta prikazana sestava očesa in prerez mrežnice. Na sprednji strani očesa je zbirna leča, ki ustvari sliko na mrežnici. Zaradi narave leče je slika na mrežnici obrnjena na glavo, in ko se v možganih slika obdela, vidimo svet pravilno obrnjen. To je raziskoval že Descartes, ko je opazoval slike skozi secirane živalske oči. Kakšen vpliv ima obrnjena slika na percepcijo sveta, je prvi opisal Stratton leta 1896, ko je eksperimentiral z lečami, ki so sliko obrnile [1]. Ugotovil je, da se po nekaj dneh lahko privadimo na obrnjeno sliko, saj možgani naredijo nove povezave. Seveda je težko spremeniti tudi druge čute, ki so prilagojeni na »notranjo« sliko sveta, in taktilni čuti velikokrat zmotijo obrnjeno predstavitev. Zato je 3D model v vizualnem smislu dovolj lahko izdelati, dokler ne vključujemo drugih čutov. 3D model je tako brez težav predstavljen zrcalno, dokler je simetričen in nimamo dodatnih informacij o dejanskem izdelku. Če bo na primer zvok modeliran na napačni strani, bo iluzija hitro izgubljena. Dokler je glavnina modeliranja osredotočena na modeliranje vizualne oblike, je to nepomembno, vendar je pričakovati hiter razvoj vmesnikov za modeliranje drugih čutil, zato postaja vedno bolj pomembno modeliranje 3D sveta tako, kot ga doživljamo. [1]



Slika 1.2. Sestava očesa [2]

Na desni strani slike 1.2 je prikazan prerez mrežnice z biološkimi senzorji za zaznavanje svetlobe. V človeškem očesu je okoli 120 milijonov paličic in od 6 do 7 milijonov čepkov. Paličice zaznavajo samo črno-belo svetlobo, vendar so sposobne zaznati en sam foton [3]. Zanimivo je, da ima oko takšno občutljivost kljub dejstvu, da svetloba pada na čepke in paličice z napačne strani. Iz evolucijskih razlogov so ti organi obrnjeni nazaj, ker je bila včasih tam odsevna plast, ki omogoča boljše zaznavanje svetlobe. Takšno odsevno plast najdemo pri ribah in nočnih živalih (na primer psi, mačke itd.), ki zaradi tega še veliko bolje zaznavajo svetlobo. Ljudje smo v procesu evolucije to odsevno plast izgubili, vendar so senzorji še vedno obrnjeni nazaj.

Čepki na mrežnici so namenjeni za zaznavanje barvne svetlobe in v človeškem očesu so tri vrste čepkov za zaznavanje rdeče, modre in zelene svetlobe. Število različnih čepkov se razlikuje in v splošnem imajo ženske več čepkov za rdečo barvo ali celo dodatno vrsto čepka za specifično rdečo barvo. Posameznik lahko ima pomanjkanje čepkov določene barve, kar povzroča delno ali popolno barvno slepoto. Pomanjkanje senzorjev za določeno barvo lahko kompenziramo v 3D modelu z večjo prisotnostjo določenih nians in uporabnik lahko primeren barvni model izbere z ustreznim uporabniškim vmesnikom.

Glede na število čepkov bi lahko sklepali, da je barvna ločljivost človeškega očesa približno enaka HD sliki na televizorju, vendar oko ne deluje tako kot običajna kamera [4]. Oko zajema sliko večkrat na sekundo in slika je na določenih mestih veliko bolj natančna. Pred ogledalom ne vidimo premikov oči, ki jih vidimo, če se posnamemo s kamero [5]. Oko se ves čas premika in premiki bi bili moteči, zato se v času premika prikazuje stara slika. Če opazujemo sekundni kazalec na uri in premikamo oči sem in tja, bo kazalec zaostajal in pospeševal, ker bomo gledali stare posnetke. Frekvenca zajemanja slik se giblje od 15 Hz do 60 Hz. Za videoposnetke se zato običajno uporablja frekvenca okoli 25 Hz ali več.

Določeni deli slike so tudi modelirani. Na mestu, kjer pride očesni živec v oko, namreč ni slike, in ta predel se imenuje slepa pega. Vendar kljub temu vidimo celotno sliko in samo s posebnimi triki lahko določimo položaj slepe pege v vidnem polju [6]. Določeni deli slike so pri obdelavi v možganih izbrisani in tako običajno ne vidimo nosa, čeprav je ves čas v sliki. Velik del vidnega polja ni produkt dejanskega zajemanja slike, ampak je slika kreirana v možganih (na primer periferni vid). Periferni vid v očesu je tudi zajet le s paličicami in je zato črno-bel, barvna slika pa se nato kreira v možganih.

Že v 19. stoletju so se pojavile prve naprave, ki so omogočale posnetke in predvajanje slik. Fotografija, filmske plošče in filmski trak so omogočali posnetke realnega sveta zadnjih 200 let in šele zdaj se počasi umikajo sodobnim elektronskim napravam. Besedo televizija je prvi uporabil Constantin Perskyi leta 1900, ko je opisoval mehanske naprave za predvajanje slik, ki so temeljile na vrtečem disku z luknjicami, skozi katere so projicirali svetlobo (Nipkow disk).

V začetku 20. stoletja so se z razvojem televizije ukvarjali številni izumitelji, med njimi tudi baron Anton Codelli v Sloveniji, ki je razvil svoj model mehanske televizije. Elektronske televizije, ki so temeljile na katodni cevi, so se pojavile v začetku 20. stoletja in so nato dolgo dominirale na področju modeliranja realnosti. Vpliv standardov elektronske televizije je prisoten še danes, ko že zelo redko najdemo kakšno televizijo, ki dejansko vsebuje elektronsko oziroma katodno cev. Tako so v ZDA zasnovali standard televizije NTSC s 525 vrsticami, v Evropi pa smo prevzeli standard PAL s 625 vrsticami, ki so ga razvili v Sovjetski zvezi [7].

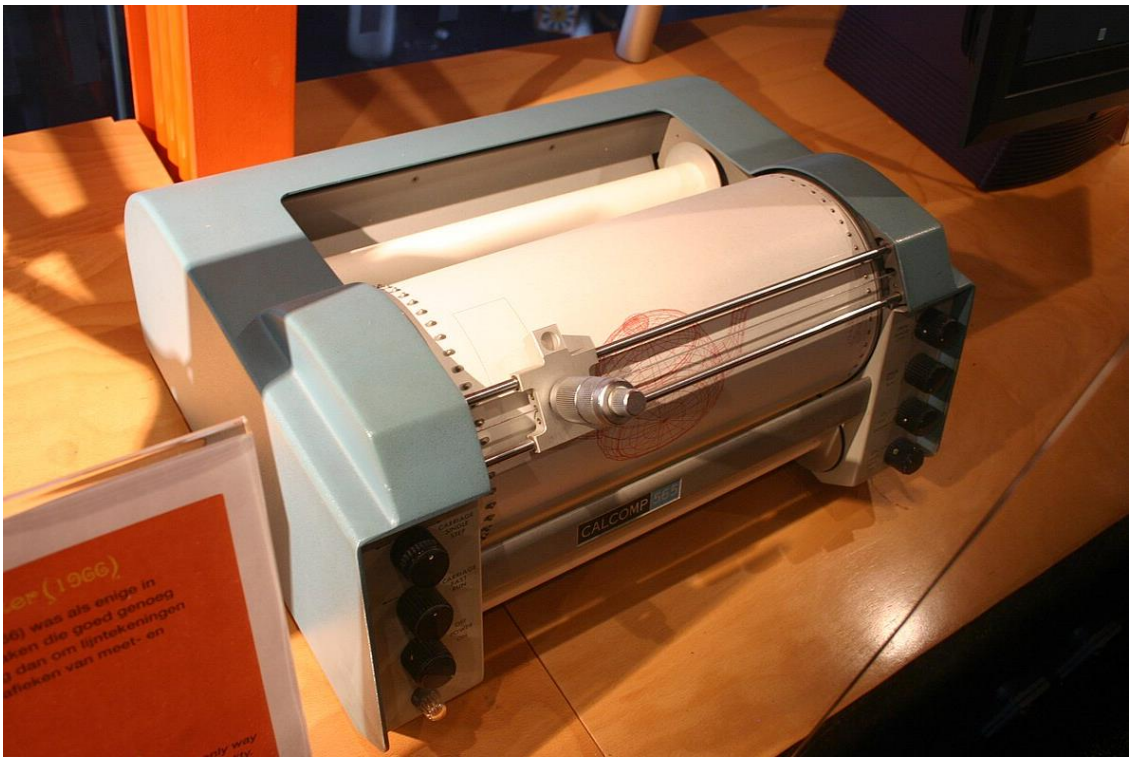
Po drugi svetovni vojni je bila hladna vojna gonilo razvoja računalništva in sistemov za prikazovanje slik. Prvi računalniški monitorji so bili razviti za potrebe obrambnega sistema ameriškega letalstva (USAF SAGE – Semi-Automatic Ground Environment [8]). V okviru sistema radarskih postaj za protizračno obrambo je bila razvita naprava, ki so jo imenovali računalniški prikazovalnik (angl. Computer Scope) in s katero so lahko opazovali radarske slike ter predvidevali pot sovražnega letala. Naprave so bile tudi interaktivne z uporabo optičnih naprav, ki so krmilile kazalec na zaslonu. Na sliki 1.3 je prikazana uporaba takšnih naprav v radarski postaji.

Slika 1.3. SAGE monitorji za nadzor zračnega prostora [9]
Credit Line: Courtesy of the Computer History Museum

Računalniško opremo je večinoma izdelovalo podjetje IBM [9], vendar je bilo veliko razvoja računalnikov in grafičnih sistemov opravljeno na MIT [10]. Na MIT je s to razvojno opremo delal Ivan Sutherland [11] in razvil prvi računalniški program za računalniško podprto konstruiranje, imenovan Sketchpad [12]. Sketchpad je imel velik vpliv na razvoj računalniške grafike, še večji vpliv pa je imel Sutherland neposredno z njegovim delom in delom njegovih doktorandov.

Izraz računalniška grafika (angl. Computer Graphics) je sicer prvi uporabil William Fetter [13]. Vendar je bil Fetter umetnik in je svoje ideje risal s kredo na tablo, uresničili pa so jih programerji. Izraz je nastal, ko so za podjetje Boeing izdelali računalniške programe, ki so izrisali grafiko [14].

Računalniška grafika v 60. letih 20. stoletja je temeljila na vektorski grafiki [15]. Način risanja grafike, kot ga poznamo pri risanju na papir ali tablo, temelji na risanju črt in likov. V računalništvu so črte določene s koordinato začetne točke in koordinato končne točke, zato so črte dejansko vektorji in grafika postane vektorska. Ko uporabimo ustrezne naprave, je mogoče vektorje narisati na površino, nato dobimo črte in like. V računalniškem smislu je ta grafika opisana s številkami, ki predstavljajo koordinate točk. Prve naprave za računalniško grafiko so bili risalniki s peresom, kot je risalnik na sliki 1.4. Koordinate točk so povezane z ukazi za dviganje in spuščanje peresa. Ko je pero spuščeno na površino, riše črte, ko je dvignjeno, se le premakne na nov začetni položaj. Na takšnem načelu temelji tudi programski jezik Logo [16], ki se pogosto uporablja za osnovno izobraževanje računalništva že v otroških vrtcih.



Slika 1.4. Risalnik Calcomp [17]

Risalnik na sliki 1.4 deluje tako, da boben premika papir v smeri y, vodoravna vodila pa premikajo pero v smeri x. Pero je lahko dvignjeno ali spuščeno. Tudi današnji risalniki so pogosto narejeni tako, da se papir premika v eni smeri, pero pa v drugi.

Namesto peresa je lahko tudi laser ali šobe za brizganje barve. Določene izvedbe risalnikov so izvedene tako, da je papir statičen in postavljen na vodoravno površino, vodila pa premikajo pero v smeri x in y. Po podobnem načelu delujejo tudi različni obdelovalni stroji, na primer laserski razrez, zato je jasno, da so bili računalniki zelo hitro uporabljeni tudi za krmiljenje obdelovalnih strojev. Tudi prvi monitorji na sliki 1.3 so delovali po enakem načelu in so risali linije po zaslonu. Takšni vektorski zasloni so se uporabljali še v osemdesetih letih dvajsetega stoletja. Vektorska grafika še danes ostaja v ozadju vseh programov računalniške grafike, kjer opisujemo geometrijo v odvisnosti od koordinat v prostoru. Večina današnje strojne opreme za prikazovanje grafike že dolgo ne deluje več po načelih vektorske oziroma analogne grafike.

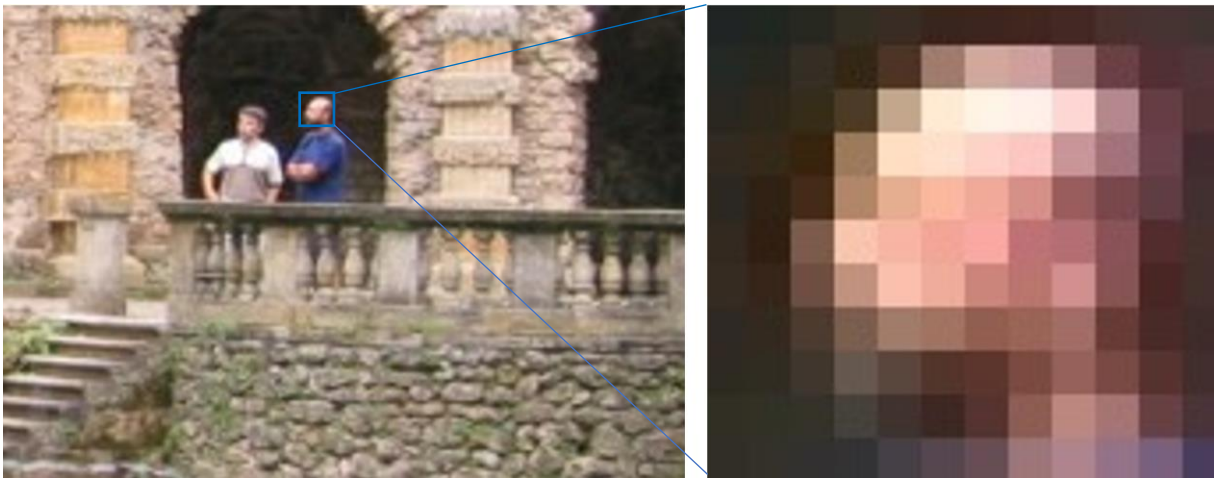
Razmah televizije, ki deluje po drugačnem načelu, je spremenil tudi naprave računalniške grafike. Televizijski standardi, NSTC in PAL ter podobni, temeljijo na risanju diskretnih pik na zaslonu. Čeprav so bili prvi televizijski aparati analogne naprave, je bil zaslon fizično razdeljen na pike. Te pike so bile postavljene v vrsticah in še danes PAL standard predpisuje 625 vrstic pik na zaslonu. Posamezno piko imenujemo piksel (angl. pixel – Picture Element), skupno število pikslov pa določa ločljivost zaslona. Grafiko, upodobljeno s piksli, imenujemo rastrska grafika [18].

Ločljivosti rastrske grafike so različne in so običajno navedene s številom pik v vodoravni in navpični smeri. Različne naprave imajo različne ločljivosti, pri katerih je tudi razmerje med širino in višino različno. Prvi monitorji so imeli pogosto razmerje med dolžino in višino enako 4:3, vendar se je izkazalo, da je za ljudi bolj ugodno razmerje, kjer je dolžina veliko večja kot višina monitorja. Zato je danes to razmerje pogosto 16:9 in v skrajnih primerih se z 21:9 približuje anamorfičnemu formatu 2,4:1 [19], ki se uporablja v kinematografiji.

Preglednica 1.1. Ločljivosti rastrskih monitorjev

Ime	Ločljivost
VGA (Video Graphics Array)	640 x 480
SVGA (Super VGA)	800 x 600
XGA (Extended Graphics Array)	1024 x 768
SXGA (Super XGA)	1280 x 1024
WXGA (Wide XGA)	1280 x 800
UXGA (Ultra XGA)	1600 x 1200
WSXGA+ (Wide SXGA plus)	1680 x 1050
WUXGA (Wide Ultra XGA)	1920 x 1200
QXGA (Quad XGA)	2048 x 1536
UHD-1 (4K)	3840 x 2160
DCI 4K	4096 x 2160
WHXGA	5120 x 3200
UHD-2 (8K)	7680 x 4320

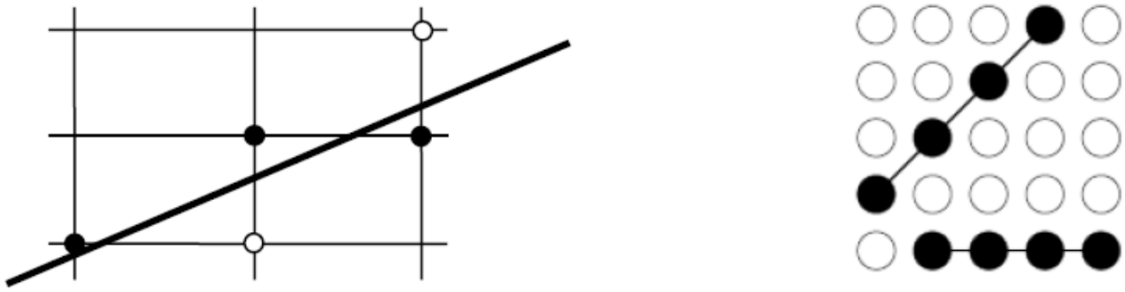
V preglednici 1.1 je prikazanih nekaj razmerij vodoravnih in navpičnih pikslov za monitorje. Ločljivosti so bile včasih tudi manjše, vendar je danes že zelo težko najti napravo, ki ima zaslon manjše ločljivosti, kot je VGA. Ker je število pik na napravah danes že zelo veliko, so pogosto označene kar s tisočicami (K – kilo), kjer je običajno navedeno število pikslov v vodoravni smeri kar s 4K, 8K, 16K, 32K itd. Večja kot je ločljivost oziroma več, kot je pikslov, boljša bi morala biti slika. Vendar je pomembno, kako velika je naprava. Pri ločljivosti 4K na majhni napravi, kot je na primer telefon, to pomeni izredno kakovostno sliko. Enaka ločljivost na zelo velikem zaslonu še komaj zadostuje. Težava nastane, ko so posamezni piksli na sliki preveliki, saj nam zaradi velikosti pikslov ne uspe več zaznati vsebine slike.



Slika 1.5. Velikost pikslov

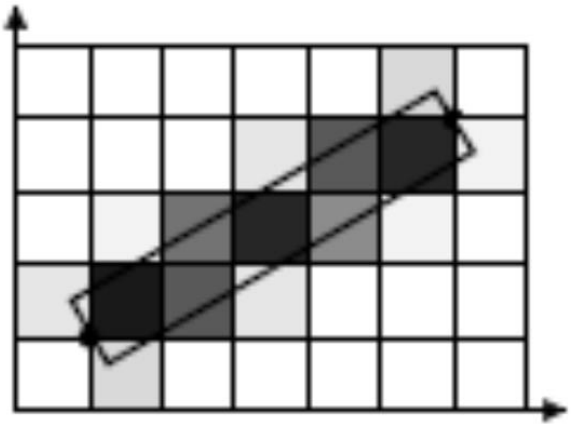
Če z levega dela slike 1.5 naredimo izrez in ga povečamo, ne bo več možno zaznati, da gre za osebo, čeprav je na levi strani to jasno. Za naprave računalniške grafike, pri katerih so oči zelo blizu površine, je zelo pomembno, da so piksli tako majhni, da ne vidimo posameznega piksla. Večje, kot so naprave, več pikslov mora biti na površini, da je posamezen piksel še dovolj majhen.

V primeru televizije ali fotografij je rastrska grafika zelo primerna, saj tudi oko zajema slike rastrsko. Težave nastanejo, ko želimo prikazati vektorsko računalniško grafiko na rastrski napravi. Z vektorsko grafiko smo narisali črto od začetne do končne koordinate. Na rastrski napravi ni nujno, da so na poti od začetne do končne točke res piksli, ki bodo to črto prikazali. Dokler gre za vodoravne in navpične črte, je to dokaj preprosto, medtem ko pri poševnih črtah takoj naletimo na težave, prikazane na sliki 1.6.



Slika 1.6. Vektorsko na rastrski grafiki

Običajno za pretvorbo med vektorsko in rastrsko grafiko skrbijo programi, vgrajeni v grafično kartico računalnika, in za linijo določijo premico med začetno in končno točko ter poiščejo najbližje piksle na poti. Vendar tudi to ne zadostuje, saj lahko dobimo nazobčano črto. Zato so potrebni številni programi, ki skrbijo, da je vektorska grafika lepo prikazana na rastrski napravi. Najpomembnejši postopek za glajenje črt (angl. anti-aliasing) je prikazan na sliki 1.7. Nekatera programska oprema omogoča uporabniku nastavitve, s katerimi lahko vpliva na izvedbo tega procesa in s tem vpliva na izgled grafike na rastrski napravi. Poleg glajenja so uporabljeni še nekateri drugi postopki, ki omogočajo bolj ali manj gladko predstavitev vektorske grafike na rastrski napravi (na primer povečanje ločljivosti). Nekatere naprave niso dovolj zmogljive in takrat izberemo hitrost prikazovanja na račun nekoliko slabše kakovosti pretvorbe v rastrsko grafiko.



Slika 1.7. Glajenje ali anti aliasing

Hitrost prikazovanja rastrske grafike je bila v času zaslonov s katodno cevjo omejena s hitrostjo osveževanja zaslona. Pri katodnih ceveh je elektronski žarek osvetljeval piksle vrstico za vrstico. Televizijski standardi, ki izvirajo iz obdobja druge svetovne vojne, so bili prilagojeni hitrosti takratnih naprav in so še dolgo osveževali le 25 slik na sekundo. Ideja je bila, da se zaslon osvežuje s frekvenco električnega toka, ki je v Evropi 50 Hz oziroma 60 Hz v ZDA. Vendar so naprave zmogle le polovico slike v enem ciklu, zato so najprej osvetlili 1., 3., 5. … vrstico in v naslednjem ciklu 2., 4., 6. … ter dosegli 25 slik na sekundo. Tak način se imenuje prepleteni (angl. Interlaced) način osveževanja, ki za računalniške monitorje nikakor ni primeren, saj zaznamo utripanje, ki je precej moteče. Zato so računalniški monitorji vedno uporabljali progresivni način osvetljevanja, kjer je bila v enem ciklu osvežena cela slika oziroma je bilo narisanih 50 oziroma 60 celotnih slik na sekundo. Vendar je bilo tudi to opazno, zato je bila težnja višja frekvenca osveževanja in v zadnjem obdobju obstoja so monitorji s katodno cevjo osveževali zaslon z več kot 100 Hz. Problem je praktično rešen z uvedbo zaslonov s tekočimi kristali, kjer ni potrebe po stalnem osveževanju zaslona, saj se spreminjajo samo piksli tam, kjer je bila slika spremenjena. Vendar to ne pomeni, da čas ni več pomemben. Ko se slika hitro spreminja (na primer film, računalniške igre), je treba spreminjati piksle vsaj z 90 Hz, in trenutno se tudi ta frekvenca povečuje na 120 Hz, 144 Hz ali več na sodobnih napravah. Čeprav človeško oko zajema običajno 25 slik na sekundo, se to zajemanje v določenih okoliščinah poveča. Zato mora biti umetno generirana grafika osveževana zelo hitro, da dobimo enak učinek, kot ga ima naravna svetloba.

V 20. stoletju so televizijski aparati in računalniški monitorji temeljili na načelu katodne cevi. Z izboljšavami v zaključku obdobja so bili ti zasloni zelo kakovostni in so dajali izvrstno sliko. Glavna težava teh naprav je bilo elektromagnetno sevanje, saj so za odklone žarka skrbeli elektromagneti, ki so hkrati porabljali veliko energije.

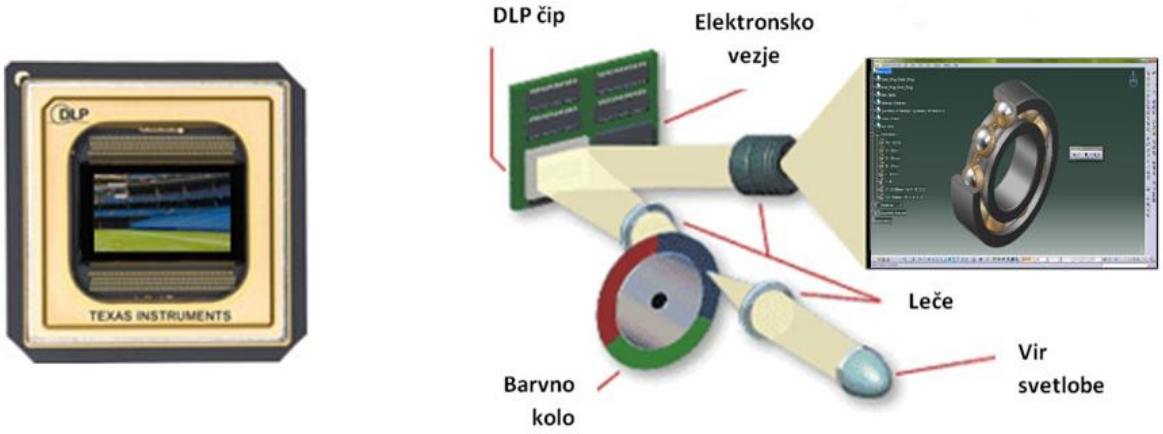
Tehnologija zaslonov s tekočimi kristali (angl. LCD Liquid Crystal Display) [20] je zahtevala kar nekaj razvoja in v prvih letih je bilo v proizvodnji LCD zaslonov več kot osemdeset odstotkov neuporabnih izdelkov (škarta). Sicer je princip LCD zaslonov relativno preprost. Vsak piksel je sestavljen iz več plasti, kjer je najpomembneje, da lahko z električnim signalom spremenimo polarizacijo LCD plasti tako, da enkrat prepušča svetlobo, drugič pa ne. Vir svetlobe je lahko katerokoli svetilo na zadnji strani zaslona.

Svetilnost LCD monitorjev so izboljšali, ko so namesto običajnega svetila dodali LED svetila, in takšni monitorji se imenujejo LED monitorji. To svetilo je lahko samo eno ali jih je več na različnih mestih zadnje strani monitorja (QLED). Čeprav QLED sugerira uporabo tehnologije kvantnih točk [21], je ta tehnologija majhnih diod, ki emitirajo svetlobo, še vedno v fazi razvoja in komercialne naprave to zgolj simulirajo z uporabo več svetil.

Vendar že obstaja tehnologija, kjer so posamezni piksli hkrati tudi vir svetlobe. Takšni monitorji se imenujejo OLED [22], kjer O v imenu označuje, da gre za organske elemente, ki oddajajo svetlobo, ko jih vzpodbudimo z električnim tokom. OLED zasloni delujejo brez dodatnega vira svetlobe in dajejo sliko z boljšim kontrastom kot LED LCD zasloni. OLED zaslone je mogoče izdelati tudi v obliki fleksibilnega filma, kar omogoča naprave, kjer se lahko zaslon zvija ali zlaga. Slabost je, da imajo OLED zasloni zaradi organske narave krajšo življenjsko dobo, poleg tega pa lahko pride do učinka zapečene slike na zaslonu, tako kot na zaslonih s katodno cevjo.

Zasloni z elementi MicroLED [23] so zelo perspektivni in kmalu si lahko obetamo naprave s to tehnologijo, ki daje še boljše rezultate kot OLED. V nasprotju z OLED osnovni svetilni elementi niso organski. Tako lahko kmalu pričakujemo zaslone, ki bodo še tanjši in bodo prikazovali večjo kakovost slike ob manjši porabi energije.

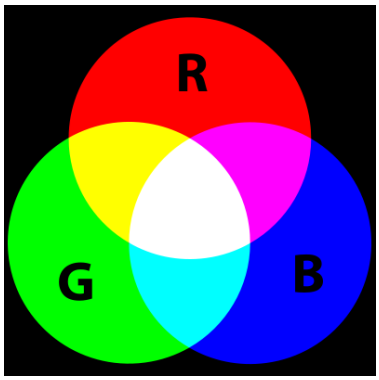
Sliko lahko z LCD tehnologijo tudi projiciramo na platno. Projektorji lahko imajo en LCD zaslon ali več zaslonov, ki generirajo sliko za posamezno barvo in nato projiciramo kompozitno sliko na platno. Pri takšnih projektorjih močna svetila za projiciranje zmanjšujejo življenjsko dobo LCD plasti in slika je vedno bolj medla. Zato so se danes precej uveljavili DLP projektorji (angl. Digital Light Processing) [24]. Pri DLP projektorjih sliko generira Texas Instruments Pico čip z mikroogledalci [25] (na sliki 1.8), kjer vsako ogledalce z vibriranjem odbije svetlobo proti platnu večkrat na sekundo. Če ni odboja proti platnu, je piksel neosvetljen in več, kot je odbojev proti platnu, bolj svetel je piksel. Za barve skrbi barvno kolo, ki se vrti med svetilom in ogledalci tako, da ogledalca odbijajo različne barve proti platnu. Tako je celotna slika praktično ustvarjena mehansko. Kljub temu takšni projektorji zmorejo prikazati več slik na sekundo kot LCD projektorji ob manjši porabi energije.



Slika 1.8. DLP projektor [25]

Tudi naprave za tiskanje slik na papir so danes vse rastrske. Za manjše formate slik lahko uporabljamo različne tehnologije tiskanja, od termalnih, laserskih do brizgalnih tiskalnikov. Za velike formate risb je najbolj racionalna uporaba brizgalnih (angl. InkJet) tiskalnikov. Risalniki, ki so risali s peresom na papir, so v večini primerov zgolj eksotične naprave za posebne namene (na primer simuliranje rokopisa).

Na sliki 1.8 vidimo barvno kolo, ki je sestavljeno iz treh segmentov za tri barve: rdečo, zeleno in modro. Tak barvni sistem imenujemo RGB (angl. Red, Green, Blue) ali seštevalni, saj nam seštevanje barv da belo barvo, kot je razvidno s slike 1.9.



Slika 1.9. RGB barvni sistem

Barva je določena s prispevkom posamezne barve, kar je lahko določeno z odstotki od 0 % do 100 % ali z decimalnimi števili od 0,0 do 1,0. Ker je prispevek barv zapisan z enim bytom, je zelo pogosto prispevek barve zapisan z vrednostjo byta. Vrednosti byta so lahko zapisane decimalno od 0 do 255 ali šestnajstiško od 00 do FF. V preglednici 1.2 je nekaj primerov zapisov barv v RGB barvnem sistemu.

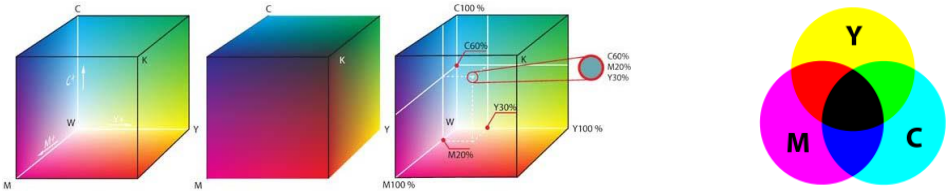
Preglednica 1.2. Primeri zapisov barv v RGB barvnem sistemu

Barva	Zapis barve v različnih oblikah
Bela	(100 %, 100 %, 100 %), (1.0, 1.0, 1.0), (255, 255, 255), #FFFFFF
Črna	(0 %, 0 %, 0 %), (0.0, 0.0, 0.0), (0, 0, 0), #000000
Rdeča	(100 %, 0 %, 0 %), (1.0, 0.0, 0.0), (255, 0, 0), #FF0000
Zelena	(0 %, 100 %, 0 %), (0.0, 1.0, 0.0), (0, 255, 0), #00FF00
Modra	(0 %, 0 %, 100 %), (0.0, 0.0, 1.0), (0, 0, 255), #0000FF
Vijoličasta	(100 %, 0 %, 100 %), (1.0, 0.0, 1.0), (255, 0, 255), #FF00FF
Rumena	(100 %, 100 %, 0 %), (1.0, 1.0, 0.0), (255, 255, 0), #FFFF00
Svetlo modra	(80 %, 90 %, 100 %), (0.8, 0.9, 1.0), (204, 229, 255), #CCE5FF
Siva	(50 %, 50 %, 50 %), (0.5, 0.5, 0.5), (128, 128, 128), #808080

S tako definiranim sistemom barv lahko zapišemo 16.777.216 barv, kar zadostuje za realistično prikazane barve, kot jih opazimo v naravi. Pri tem privzamemo, da je ozadje črno, kar je na napravah za prikazovanje precej težko zagotoviti. Monitorji so običajno sive barve, ko so izključeni. Tudi platno za projiciranje je pogosto belo, da odbija čim več svetlobe. Zato je na teh napravah težko zagotoviti čisto črno barvo. Poleg tega je s kombinacijo prispevkov rdeče, zelene in modre težko zagotoviti belo barvo. Zato je pogosto dodan še en byte, ki predstavlja svetlost posamezne pike. Pogosto se to imenuje kanal alfa (angl. Apha Channel) in pri tem v sistemu dodamo še en byte, ki naredi barvo zelo svetlo z vrednostjo 100 % ali temno z 0 %. Hkrati kanal alfa omogoča prosojnost, saj v primeru, da je vrednost manjša od 100 %, lahko kombiniramo več barv in tako prikažemo objekte, ki se vidijo skozi druge objekte. Barve so v sistemu običajno označene s črkami RGB (Red, Green, Blue), alfa kanal pa je označen s črko A.

Obstajajo različni sistemi zaporedja barv v odvisnosti od strojne opreme in programske opreme, na primer RGBA, ARGB, ABGR, BGRA [26]. Tako vsako barvo zapišemo s štirimi byti ali z eno 32-bitno besedo in pogosto je t. i. resnična barva (angl. true color) označena kot 32-bitna barva. Štirje byti so potem tudi zahtevan prostor v pomnilniku, in če želimo določiti, koliko prostora v računalniku zavzame slika, je treba ločljivost slike pomnožiti s štiri. Ločljivost UHD oziroma 4K iz tabele 1.1 zahteva približno 33 MB prostora v pomnilniku. Izraz megapiksel, ki se uporablja v fotografiji, označuje samo število pik in običajno je privzeto 32 bitov oziroma 4 byte za vsak piksel. Zato 1 megapiksel zahteva 4 MB pomnilnika.

RGB barvni sistem je prirejen za črno ozadje, v primeru tiska na belo ozadje, na primer na papir, je treba uporabiti drugačen barvni sistem, ki uporablja druge osnovne barve. Tak sistem je CMYK, prikazan na sliki 1.10, ki ga sestavljajo barve: sinje, škrlatna, rumena in črna (angl. Cyan, Magenta, Yellow, Black). Tak sistem imenujemo odštevalni, ker imamo na začetku združene vse barve v beli barvi ozadja, in ko dodajamo barve, jih pravzaprav odštevamo in dobimo na koncu črno barvo. Ker kombinacija sinje, škrlatne in rumene ni čisto črna barva, je treba dodati še črno barvo tako, kot je dodan alfa kanal v RGB barvnem sistemu za belo barvo in za prosojnost. Tudi v tem sistemu barve opišemo s kombinacijami, ki jih dobimo, ko združimo prispevke vseh štirih barv in vsak piksel tudi tukaj zahteva 32 bitov pomnilnika.



Slika 1.10. Barvni sistem CMYK [27]

Ta dva sistema sta danes najpogosteje v uporabi in na zaslonih uporabljamo RGBA, ko pa želimo sliko natisniti, jo je treba pretvoriti v sistem CMYK. Nekateri programi uporabljajo tudi drugačne zapise barv [28], ki jih srečamo precej redkeje kot zgoraj omenjena sistema.

Programska oprema računalniške grafike je bila dovolj preprosta v obdobju, ko je bila strojna oprema vektorska. Kljub temu pa je že takrat nastala prva grafična programska knjižnica Plot10, ki jo je izdelalo podjetje Tektronix za svojo strojno opremo. Namesto samostojnega kreiranja programa za računalniško grafiko je tako uporabnik uporabil programsko knjižnico, v kateri so bili podprogrami, ki so omogočali delo z računalniško grafiko. Namesto programiranja specifičnih ukazov v obliki binarnih kod, ki jih je bilo treba poslati na napravo, je programer uporabil specifičen podprogram. Tako ni bilo treba proučevati strojne opreme in iskati kode (zaporedja bytov), ki bo omogočala risanje črte na zaslonu, dovolj je bil klic podprograma za risanje črte. V knjižnici so bili zbrani vsi ukazi, ki so programerjem omogočali preprosto programiranje grafike na zaslonu. Z novo strojno opremo se je programska knjižnica dopolnjevala in podpirala različne terminale. S prihodom rastrske grafike so bile v knjižnico dodane vse nižjenivojske funkcije, ki so omogočale risanje tudi na rastrskih napravah.

Knjižnica Plot10 je bila namenjena terminalskemu delu, kjer ni bilo veliko interaktivne grafike. Delovne postaje, ki so imele neposredno povezavo z zaslonom tako kot današnji osebni računalniki, so omogočale bolj zahtevno uporabo računalniške grafike. Zato je bilo kar nekaj poskusov standardov za programske knjižnice za računalniško grafiko. Najprej je leta 1977 do 1998 nastajal GKS (angl. Graphical Kernel System), pozneje, od leta 1988 do 1997, še PHIGS (angl. Programmer's Hierarchical Interactive Graphics System). Obe knjižnici sta bili ISO standard, pri čemer je bil PHIGS zamenjava za GKS. Ideja je bila, da lahko napišemo univerzalni program s pomočjo standardne knjižnice, ki bo deloval na katerikoli napravi.

Poleg standardnih knjižnic so še vedno nastajale grafične knjižnice znotraj različnih podjetij in ena od teh je grafična knjižnica X Window System oziroma X11 [30] operacijskega sistema UNIX (pozneje tudi v LINUX). Razvoj te knjižnice se je začel leta 1984 in je v uporabi še danes. Iz imena izhaja, da knjižnica uporablja ekranska okna, ki so grafično prikazana na zaslonu. Zanimivo pri tem sistemu je tudi to, da lahko okno prikažemo tudi na oddaljenem računalniku in tako poganjamo program na enem računalniku, pri tem pa je rezultat prikazan na zaslonu drugega računalnika.

Druga takšna knjižnica OpenGL [31] je nastala leta 1992 v podjetju Silicon Graphics, ki je bilo takrat vodilni proizvajalec grafičnih delovnih postaj. Knjižnico so dali brezplačno na voljo vsem uporabnikom in kmalu je podpirala vso strojno opremo. Danes je na voljo tako za računalnike, opremljene z Linux operacijskimi sistemi, kot tiste z Windows sistemom, podpira pa tudi različne mobilne naprave z operacijskim sistemom Android in iOS.

Microsoft je leta 1995 v operacijskem sistemu Windows razvil svojo grafično knjižnico DirectX [32], na kateri temelji vsa grafika v tem operacijskem sistemu. Obe knjižnici, OpenGL in DirectX, podpirata številne ukaze za 2D in 3D grafiko, vendar je DirectX na voljo samo v operacijskem sistemu Windows. Zato so številni grafični programi zgrajeni s knjižnico OpenGL, saj jim to omogoča delo na bolj raznoliki strojni opremi. Veliko programske opreme za računalniško grafiko je že vključene v strojno opremo grafičnih procesorjev GPU (angl. Graphical Processing Unit), zato je delovanje grafičnih programov precej neodvisno od operacijskega sistema naprave.

Čeprav niti DirectX niti OpenGL nista mednarodna standarda, sta ti dve knjižnici uporabljani za izdelavo računalniške grafike namesto standardnih knjižnic GKS in PHIGS, ki sta utonili v pozabo. Tako DirectX kot OpenGL se danes še aktivno razvijata in dopolnjujeta ter tako omogočata kreiranje grafike na najsodobnejših napravah za prikazovanje računalniške grafike. V večini primerov sta obe knjižnici pogosto sestavni del operacijskega sistema na grafični računalniški opremi.

2 Zasnova modeliranja

Pri 3D modeliranju je treba nekje začeti. Včasih je zasnova skrita v ideji za izdelek ali je treba zasnovati izdelek glede na želene zahteve. Na voljo je kar nekaj načinov, kako pretvorimo to idejo v računalniški model. Najosnovnejši način je skica izdelka, ki so jo obvladali že pred 73.000 leti [33], kar velja za najstarejšo najdbo risanja na stene votlin. Risanje realističnih objektov v perspektivni projekciji je precej zahtevno in je trajalo kar nekaj časa, da je človeštvo osvojilo to tehniko [34]. Od takrat do danes smo se veliko naučili, vendar je osnovno orodje še vedno enako in z roko rišemo črte in kreiramo risbo. Seveda so se spremenili podlaga, pisalo in tehnika risanja. V zadnjem času je za skiciranje mogoče uporabiti tudi sodobna elektronska pomagala.

Drugi način pretvorbe ideje v obliko je ročno modeliranje izdelka. Najstarejši izdelki iz gline so bili izdelani na Kitajskem pred osemnajst tisoč leti, ker je glino dokaj lahko oblikovati in z vročino pretvoriti v trajno obliko. Seveda je tudi takšno ročno modeliranje še danes aktualno z glino in s podobnimi materiali.

Ročno izdelano skico ali ročno modeliran izdelek lahko pretvorimo v računalniško obliko. Pogosto je potem treba uporabiti še dodatna orodja za pretvorbo v natančen računalniški model izdelka. Mogoče je tudi direktno modelirati izdelek, vendar to vseeno pogosto zahteva izdelavo skic v programih za 3D modeliranje.

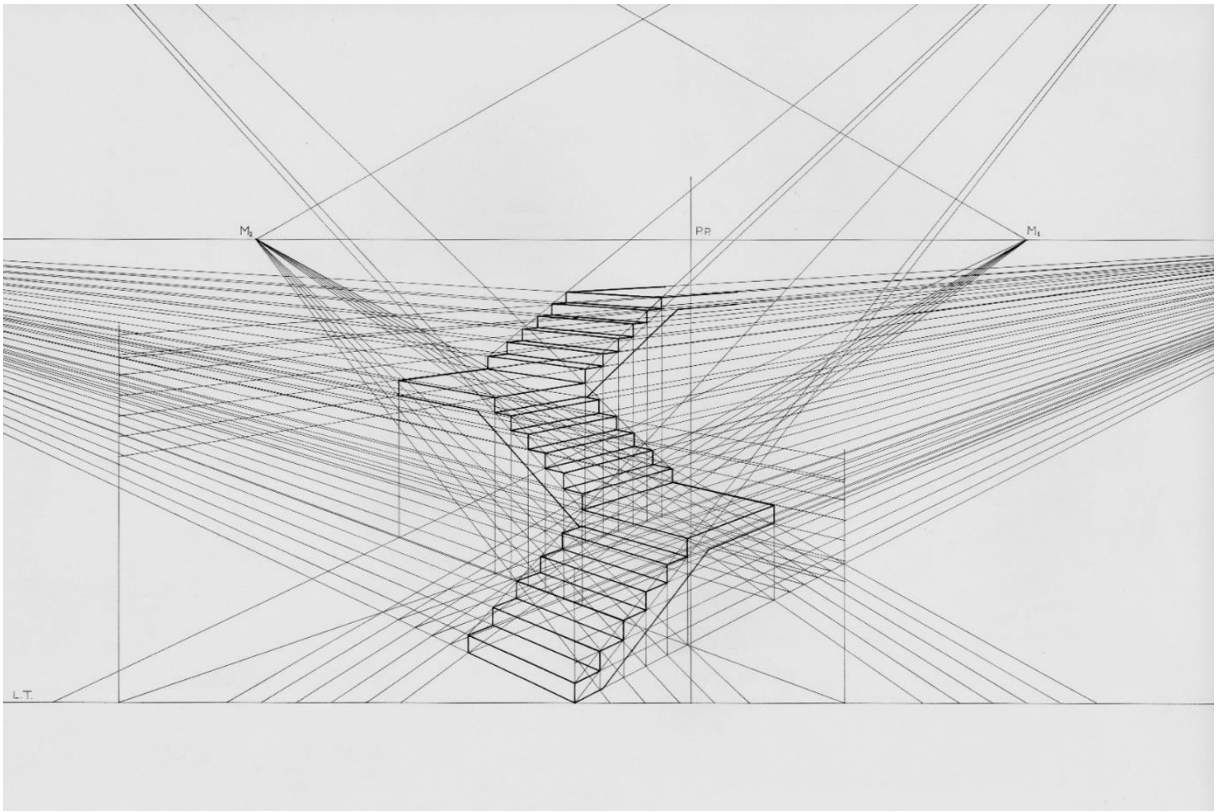
Prostoročno skiciranje pomeni samo uporabo svinčnika in papirja. Druga pomagala, na primer ravnilo, podaljšajo čas skiciranja in se jih raje izogibamo. Ravno črto je mogoče preprosto potegniti od začetne točke tako, da vlečemo pisalo proti cilju, ki ga imamo na očeh. Dokaj lahko je risati vodoravne in navpične črte, medtem ko je poševne črte lažje risati tako, da ustrezno obrnemo papir in spet rišemo vodoravno ali navpično. Krožnice rišemo s pomočjo vodoravnih in navpičnih srednjic. V določenih primerih si lahko pomagamo tako, da si predhodno označimo točke z razdaljo polmera od središča kroga na različnih točkah oboda.

Z elektronskimi pripomočki je prostoročno risanje še preprostejše, saj programska oprema prepozna geometrijske like, ki jih želimo narisati. Na računalniški tablici to deluje tako, da s peresom narišemo lik in na koncu pustimo pero na površini tablice. Posledično se lik samodejno popravi v idealno obliko [36]. Takšna programska oprema je vedno bolj pogosta, posebej na napravah z zaslonom na dotik, kjer je možno prostoročno skicirati.

Skice z miško predvidevajo uporabo vektorskih ukazov, kot je risanje linij, krožnih lokov ali krivulj, ki jim na določen način določimo parametre. Skicirka je danes sestavni del vsakega modelirnika, vendar redko omogoča dejansko prostoročno skiciranje. Zato je za izdelke, ki niso oblikovani z matematično določenimi oblikami, še vedno treba skicirati prostoročno. Zaradi zahtev po računalniški obdelavi izdelka je treba te prostoročne skice spremeniti v računalniški model.

Realistične slike so narisane v perspektivni projekciji, kjer so dimenzije objekta odvisne od njegove razdalje od gledišča. V naravi so objekti v daljavi seveda manjši kot tisti, ki so nam bližje. Tako je na sliki 2.1 hodnik z oddaljenostjo vedno ožji in vrata na hodniku so vedno manjša. Takšno perspektivno projekcijo imenujemo tudi enotočkovna perspektivna projekcija, saj se robovi v daljavi navidezno združijo v eni točki. Perspektivna projekcija v obliki skice je lahko tudi dvotočkovna ali večtočkovna [37]. Na sliki 2.2je prikazana večtočkovna perspektivna projekcija, ki se pogosto uporablja v arhitekturi.

Slika 2.1. Perspektivna slika

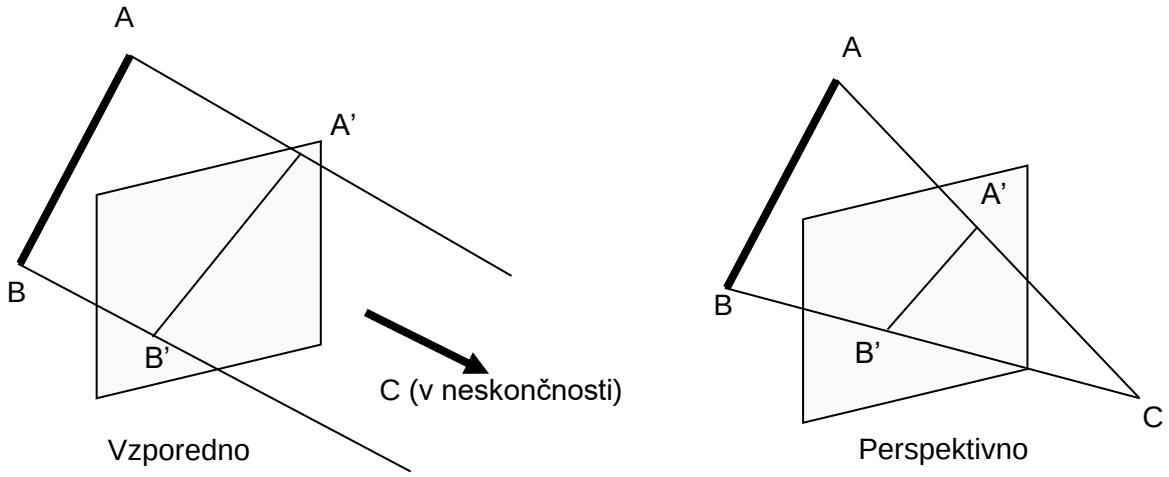


Slika 2.2. Večtočkovna perspektivna projekcija [37]

Perspektivno projekcijo lahko uporabljamo tudi pri kreiranju izdelkov tako, da navidezno povečamo ali zmanjšamo določen del izdelka. Primer takšnega perspektivnega oblikovanja je glavna ulica v Disneylandu na sliki 2.3, kjer je ulica oblikovana tako, da je na začetku dejansko široka, na koncu pa ozka. Tudi zgradbe so na začetku ulice visoke, na koncu pa majhne. Seveda so ustrezno skalirani tudi vsi elementi zgradb (okna, vrata itd.) Na tak način izgleda ulica daljša in grad na koncu ulice izgleda mnogo večji, dokler seveda ne pridemo do tja, ko vidimo, da sploh ni tako visok.

Slika 2.3. Glavna ulica v Disneylandu [38]

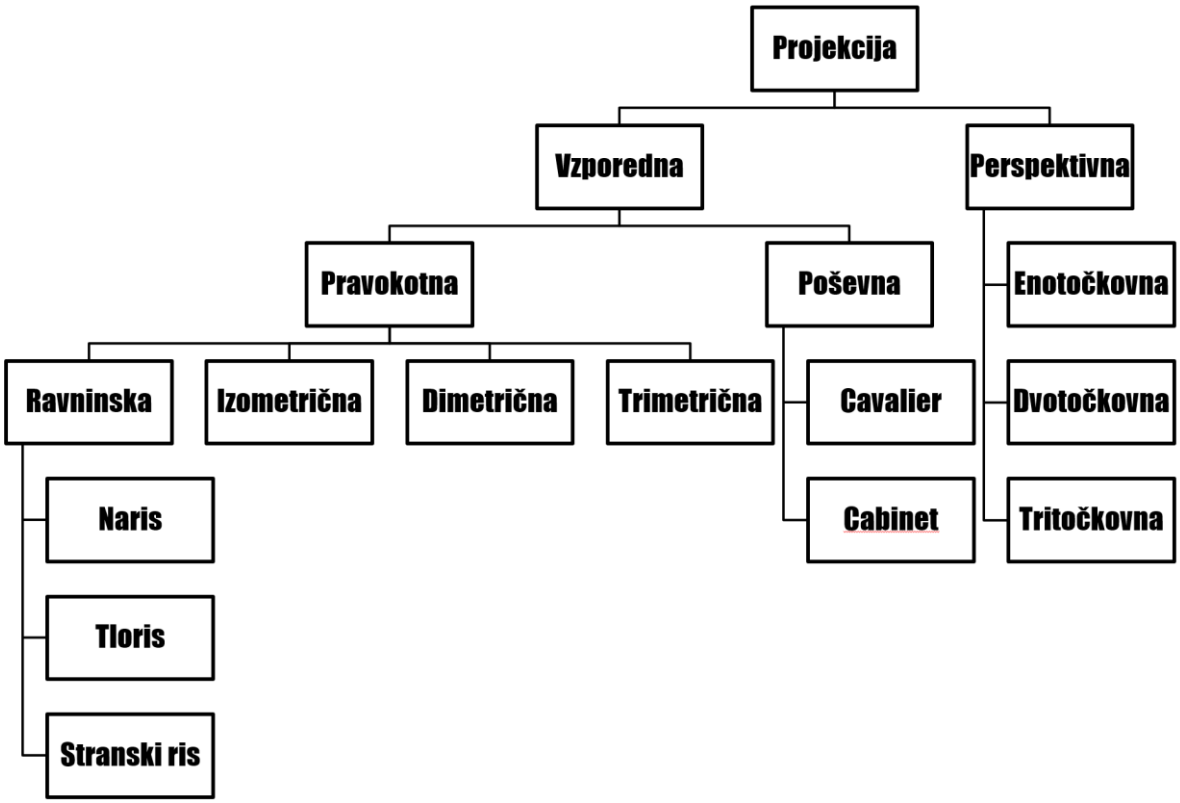
Perspektivna projekcija se pogosto uporablja za arhitekturne skice in za končno predstavitev virtualnih izdelkov, ko želimo realistično predstavitev izdelka. Za zasnovo inženirskih izdelkov in nadaljnjo obdelavo izdelka z računalniškimi orodji je perspektivna projekcija manj primerna. Veliko primerneje je uporabljati vzporedno projekcijo v ta namen. Na sliki 2.4 sta prikazani obe projekciji, ki se razlikujeta po poteku premic, ki oblikujejo predmet na projekcijski ravnini. Pri perspektivni projekciji se premice združijo v eni točki ali v več točkah, če gre za večtočkovno projekcijo. To točko imenujemo očišče (angl. vanishing point), premice v projekciji pa bežiščnice.



Slika 2.4. Vzporedna in perspektivna projekcija

Pri vzporednih projekcijah [39] so bežiščnice vzporedne, zato se očišče nahaja v neskončnosti. Posledica vzporednih bežiščnic je ohranjanje dimenzij objekta, ki so neodvisne od razdalje od očišča. Zagotavljanje enakosti dimenzije je mogoče v vseh smereh in potem govorimo o izometriji [40].

Pri vzporednih projekcijah ne moremo govoriti o večtočkovnih projekcijah, ker je očišče vedno v neskončnosti. Vseeno poznamo več različnih vzporednih projekcij, ki se razlikujejo po smeri projekcije glede na projekcijsko ravnino in po različnih skalirnih faktorjih v različnih smereh. Na sliki 2.5 je prikazana klasifikacija najbolj pogostih perspektivnih in vzporednih projekcij.

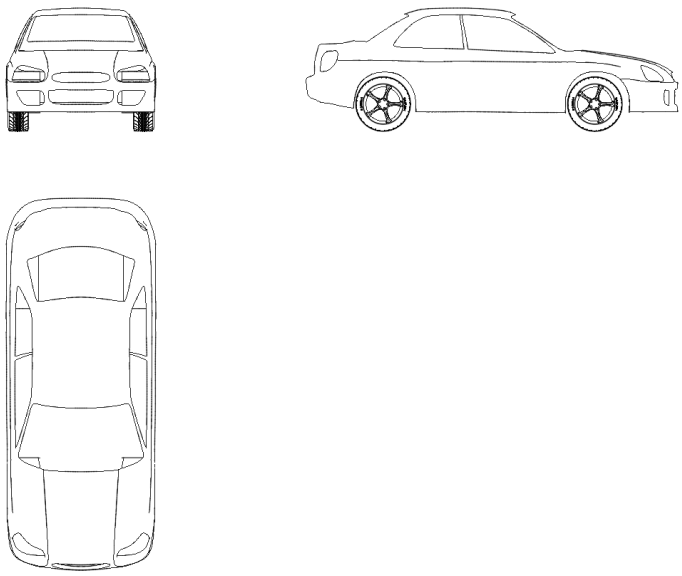


Slika 2.5. Različne vzporedne in perspektivne projekcije

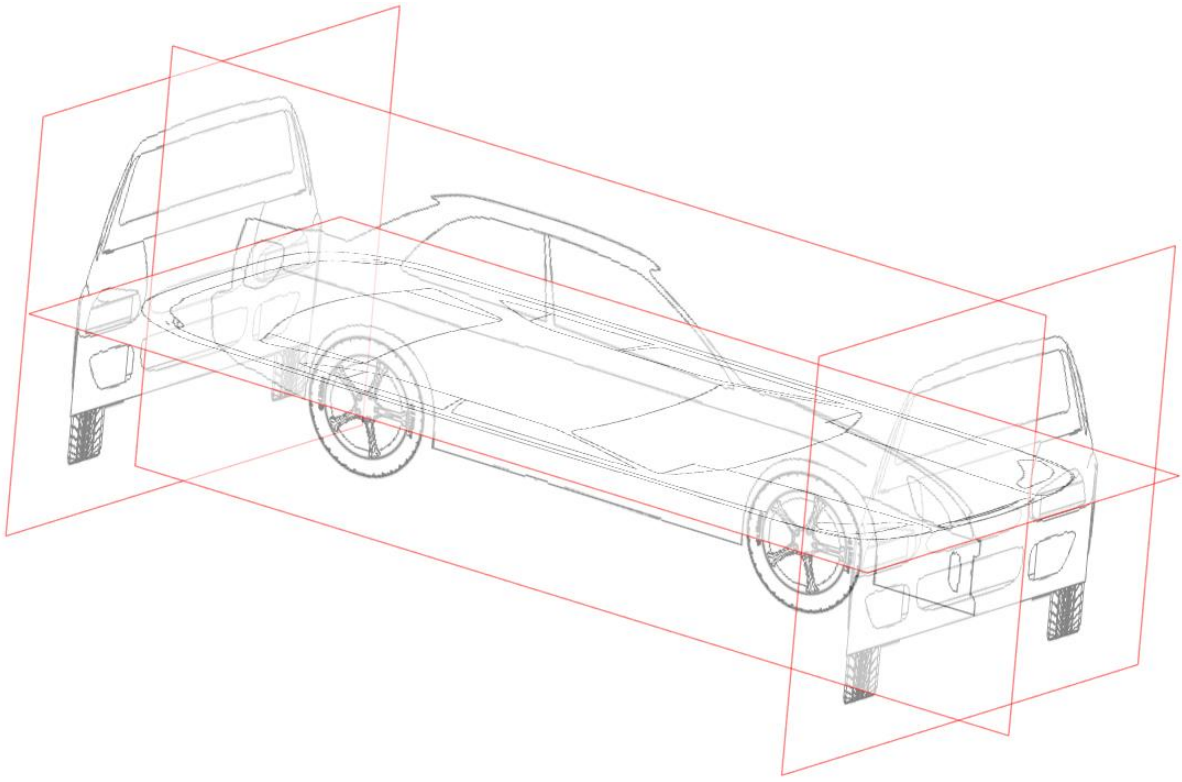
Največkrat se srečamo s pravokotnimi vzporednimi projekcijami. Poševne vzporedne projekcije se uporabljajo pri upodobitvah v različnih katalogih, vendar tako kot perspektivne projekcije niso primerne za uporabo skic z računalniškimi orodji, tudi vse pravokotne projekcije niso primerne in najraje uporabljamo izometrične projekcije. Najbolj so uporabne vzporedne pravokotne ravninske projekcije, ki jih imenujemo tudi ortogonalne projekcije. Na sliki 2.5 so prikazane samo tri ortogonalne projekcije, in sicer naris – pogled od spredaj, tloris – pogled od zgoraj in stranski ris – pogled od strani. Skice v ortogonalnih projekcijah so koristne, ker jih lahko v 3D modelirniku postavimo na koordinatne ravnine in jih uporabimo za kreiranje geometrije 3D modela.

Inženirji pogosto uporabljajo ortogonalne poglede za izdelavo skic, medtem ko oblikovalci in umetniki izdelek opazujejo manj tehnično in raje izdelajo skico v perspektivni ali v izometrični vzporedni projekciji. Če obstaja fizični model izdelka, je sicer mogoče narediti ortogonalne fotografije, vendar je potem boljša rešitev 3D skeniranje in rekonstrukcija izdelka v 3D modelirniku (kar tudi ne poteka vedno čisto gladko in brez težav). Zato so ortogonalne skice vedno najbolj zaželene za osnovne skice za 3D model.

Na sliki 2.6 je primer takšne skice, ki jo lahko uporabimo za osnovo pri gradnji 3D modela. Posamezne poglede je treba postaviti na koordinatne ravnine in jih ustrezno skalirati, kar omogoča gradnjo geometrije z risanjem preko skic v 3D modelirniku.



Slika 2.6. Ortogonalne skice



Slika 2.7. Postavitev skic na koordinatne ravnine

Na sliki 2.7 je prikazana postavitev ortogonalnih skic na koordinatne ravnine v 3D modelirniku. Rastrske skice so tako prikazane na ravninah v velikostih, ki omogočajo kreiranje geometrije izdelka. Zaradi skaliranja rastrskih slik so te slike običajno slabše kakovosti, vendar to ne vpliva na natančnost modela, ki je potem določena s postopkom konstruiranja izdelka. Skice služijo zgolj za osnovo in v nadaljevanju ne rekonstruiramo pikslov, ampak kreiramo novo vektorsko upodobitev izdelka v 3D prostoru.

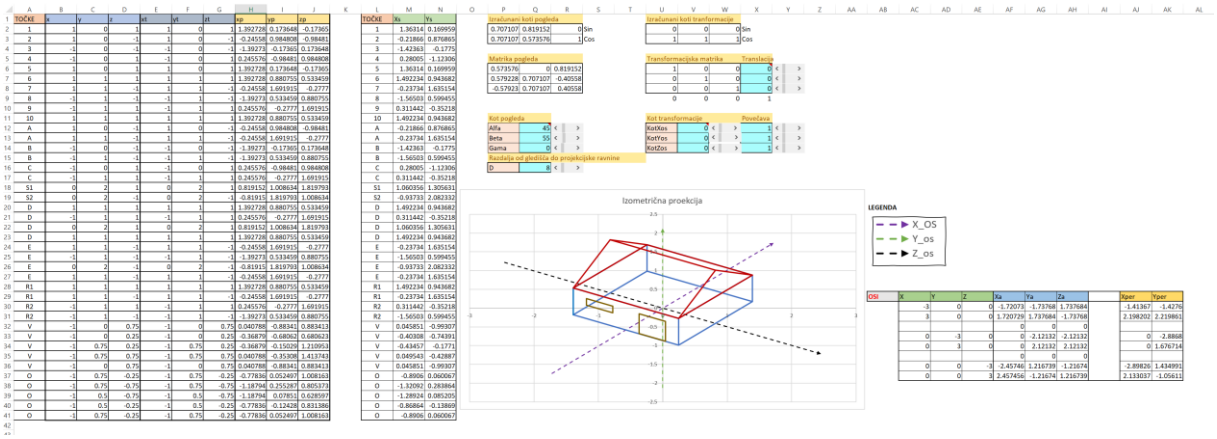
3 Modeliranje

Modeliranja s pomočjo računalnika se lahko lotimo na različne načine. Skico lahko zgolj pretvorimo v računalniški model. Izkaže se, da to ni vedno preprosto, saj želimo, da je model predstavljen v 3D, da lahko izdelamo simulacije modela ali pripravimo postopek za izdelavo izdelka. Zato je postopek modeliranja drugačen in v ta namen so bili razviti različni 3D modelirniki, ki omogočajo različne predstavitve modela v računalniku.

3.1 Žični model

Žični modeli so bili v uporabi veliko pred razvojem računalnikov. Velikokrat z žicami kreiramo modele molekul v kemiji ali zgradb v arhitekturi. Model je preprosto izdelati in predstavlja 3D model izdelka v stilizirani obliki. Velikokrat je mogoče tudi preveriti nosilnost izdelka in določiti mesta podpor ali ojačitev. Arhitekt Gaudi je za svojo katedralo Sagrada Familia v Barceloni izdelal žični model iz vrvic in uteži [41]. Ta model katedrale je obrnjen na glavo z namenom določanja oblike nosilnih elementov.

Žični model je mogoče narediti tudi na računalniku. Prvi je žične 3D modele na računalniku v začetku 60. let 20. stoletja izdelal Ivan Sutherland [11]. Uporabiti je treba ustrezno projekcijo in 3D model v obliki črt lahko narišemo na zaslonu. Tako je mogoče narediti žični modelirnik v programu MS Excel, prikazan na sliki 3.. V Excelu je narisan graf z ravnimi odseki med točkami tako, da so koordinate točk zapisane s tremi koordinatami in nato projicirane z uporabo vzporedne ali perspektivne projekcije.



Slika 3.1. Žični model v Excelu

Modelirnik na sliki 3., ki je bil izdelan za domačo nalogo pri predmetu 3D modeliranje, omogoča tudi geometrijske transformacije in model lahko premikamo, skaliramo in vrtimo v različnih smereh. V ta namen lahko uporabimo transformacijsko matriko reda 4 x 4, saj je operacija translacije zapisana v matriki v četrtem stolpcu. Zaradi tega morajo biti vektorji koordinat homogenizirani, kar pomeni, da dobijo četrti člen z vrednostjo 1. Translacija je določena s členi, ki določajo premik v smeri posamezne osi, kot je prikazano v enačbi 3.1. V enačbi 3.2 je prikazano skaliranje, ki je določeno s skalirnimi faktorji na diagonalnih členih matrike. Skalirni faktorji, večji od 1, omogočajo povečave, skalirni faktorji, manjši od 1, pa pomanjšave objekta. Rotacije okoli posameznih osi so določene s kotnimi funkcijami, razporejenimi v matriki. Enačba 3.3 določa rotacijo okoli *x* osi, enačba 3.4 je za rotacijo okoli osi *y*, rotacija okoli osi *z* pa je definirana v enačbi 3.5.

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$\boldsymbol{S} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$\mathbf{R}_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$\mathbf{R}_z = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Zgornje enačbe rotacij veljajo za Eulerjeve kote, kjer je *α* kot rotacije okoli osi *x*, *β* kot rotacije okoli osi *y* in *γ* kot rotacije okoli osi *z*. Če gre za tri rotacije, velja Eulerjev rotacijski teorem, in rezultat produkta dveh rotacijskih matrik je rotacijska matrika.

Matrične operacije za 3D rotacije sicer niso komutativne, zato je treba paziti na vrstni red operacij, vendar pri produktu treh matrik velja **ABC**=**A**(**BC**)=(**AB**)**C**. Zato lahko vse tri rotacije iz enačb 3.3, 3.4 in 3.5 zapišemo s produktom matrik **Rz**·**Ry**·**Rx,** zapisanim v enačbi 3.6. Nato pomnožimo najprej dve matriki in dobimo produkt dveh matrik v enačbi 3.7, produkt teh matrik pa je kompozitna rotacija, zapisana v enačbi 3.8

$$\begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$\begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & \sin\alpha\sin\beta & \cos\alpha\sin\beta & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$\begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & 0 \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & 0 \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

Če v enačbo 3.8 dodamo še člene za translacijo v četrti stolpec in skalirne faktorje na diagonalo, dobimo kompozitno transformacijsko matriko v enačbi 3.9. S takšno transformacijsko matriko lahko nato množimo homogenizirane koordinate točk modela.

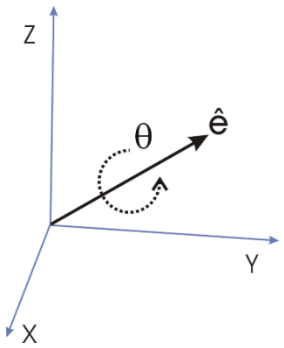
$$\begin{bmatrix} s_x(\cos\beta\cos\gamma) & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & d_x \\ \cos\beta\sin\gamma & s_y(\sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma) & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & d_y \\ -\sin\beta & \sin\alpha\cos\beta & s_z(\cos\alpha\cos\beta) & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

Poleg rotacijskih matrik z Eulerjevimi koti je mogoče uporabiti tudi Tait-Brynove kote [42], kjer je *α* kot rotacije okoli osi *z* (smer – angl. Yaw), *β* kot rotacije okoli osi *y* (Naklon – angl. Pitch) in kot *γ* rotacije okoli osi *x* (prevračanje – angl. Roll). Potem je tudi kompozitna rotacijska matrika nekoliko drugačna.

V računalniški grafiki se za rotacijo objektov pogosto uporabljajo tudi kvaternioni (angl. Quaternions) [43]. Teorijo je že leta 1843 razvil Rowan Hamilton in temelji na imaginarnih številih. V 3D prostoru imamo tri takšna števila i, j in k, za katere velja enačba 3.10.

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.10}$$

Krajevni vektorji točk, na primer (a_x, a_y, a_z), so potem zapisani v obliki $a_x i + a_y j + a_z k$.



Slika 3.2. Rotacija okoli Eulerjeve osi

Nato lahko rotacijo za kot θ okoli Eulerjeve osi na sliki 3.2zapišemo s kvaternioni. Os je definirana z enotskim vektorjem v enačbi 3.11.

$$\boldsymbol{u} = (u_x, u_y, u_y) = u_x i + u_y j + u_z k \tag{3.11}$$

Rotacijo nato zapišemo s kvaternionom q v enačbi 3.12.

$$\boldsymbol{q} = e^{\frac{\theta}{2}(u_x i + u_y j + u_z k)} = cos\frac{\theta}{2} + (u_x i + u_y j + u_z k) sin\frac{\theta}{2} \tag{3.12}$$

Poljubna točka p je zapisana v obliki enačbe 3.13.

$$\boldsymbol{p} = (p_x, p_y, p_y) = p_x i + p_y j + p_z k \tag{3.13}$$

Rotacijo točke p okoli osi u za kot θ izračunamo z uporabo enačbe 3.14.

$$\boldsymbol{p}' = qpq^{-1} \tag{3.14}$$

Pri tem je q^{-1} določen z enačbo 3.15.

$$\boldsymbol{q^{-1}} = e^{-\frac{\theta}{2}(u_x i + u_y j + u_z k)} = cos\frac{\theta}{2} - (u_x i + u_y j + u_z k) sin\frac{\theta}{2} \tag{3.15}$$

Kvaternioni nam omogočajo rotacijo okoli poljubne osi, vendar ne vključujejo skaliranja in translacije. Zato so kvaternioni bolj namenjeni za uporabo v programih računalniške grafike na ravni programiranja in končni uporabnik v modelirnikih pogosteje sreča transformacijsko matriko.

Poleg omenjenih transformacij modela se pri modeliranju pogosto srečamo tudi s transformacijo, ki jo imenujemo zrcaljenje. Zrcaljenje v prostoru je transformacija koordinat prek zrcalne ravnine. Zrcaljenje prek ravnine lahko definiramo z enačbo 3.16, če je ravnina definirana z enačbo 3.17. Če postavimo koordinatno izhodišče na zrcalno ravnino, lahko z namenom zrcaljenja samo zamenjamo predznak koordinatam modela. Tako imajo koordinate pozitivno vrednost na eni strani zrcalne ravnine in negativno vrednost na drugi strani.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1-2a^2 & -2ab & -2ac & -2ad \\ -2ab & 1-2b^2 & -\sin\alpha & -2bd \\ -2ac & -2bc & 1-2c^2 & -2cd \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.16}$$

$$ax + by + cz + d = 0 \tag{3.17}$$

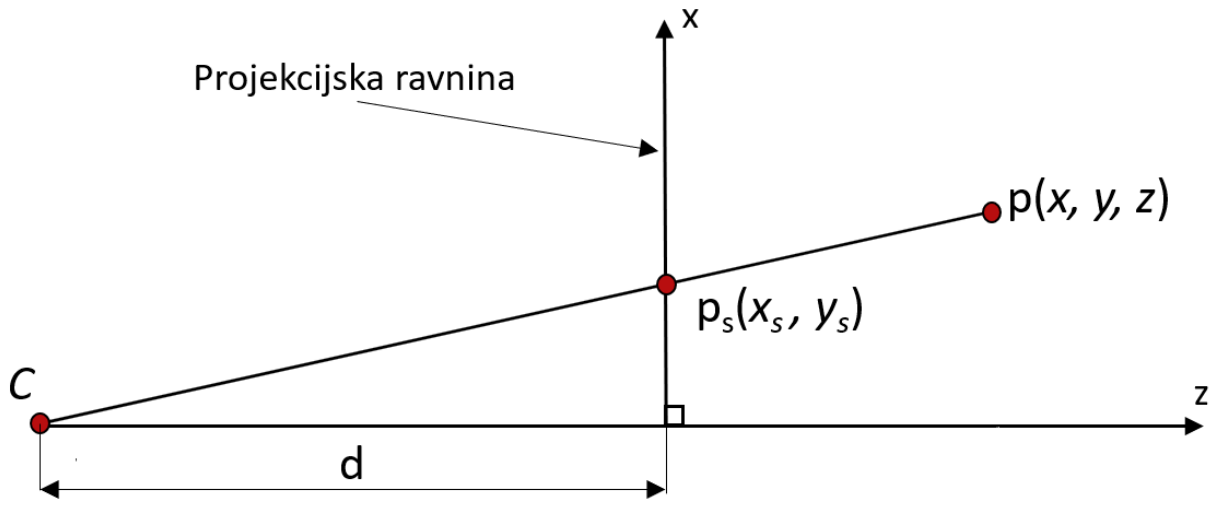
Geometrijske transformacije lahko uspešno uporabimo tudi za izračun projekcije točk na projekcijsko ravnino. To nam omogoča prikaz 3D objektov na ravnini računalniškega zaslona. V primeru vzporedne projekcije lahko uporabimo rotacijske matrike in sestavimo matriko pogleda, kot v enačbi 3.8. Če spremenimo vrstni red rotacij in določimo kombinacijo transformacije s produktom **Rx**·**Ry**·**Rz** v enačbi 3.18, je rezultat v enačbi 3.19 nekoliko drugačen, kot smo ga dobili v enačbi 3.8. V enačbah tudi nismo uporabili homogenizacije, ker pri pogledu ni treba upoštevati translacije. Nato za projekcijo uporabimo samo koordinate x in y, kot je zapisano v enačbi 3.20.

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.18}$$

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \sin\alpha\sin\beta\cos\gamma + \cos\alpha\sin\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\cos\gamma & -\sin\alpha\cos\beta \\ \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma & \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\beta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.19}$$

$$\begin{bmatrix} x_p \\ y_p \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} \tag{3.20}$$

Pri perspektivni projekciji je treba upoštevati razdaljo do očišča C na sliki 3.3 in koordinate točk izračunati tako, kot je prikazano v enačbah 3.21 in 3.22. V enačbi 3.21 je določena matrična oblika perspektivne projekcije, s katero pomnožimo homogenizirane vektorje točk.



Slika 3.3. Perspektivna projekcija

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ \frac{z}{f} \end{bmatrix} \tag{3.21}$$

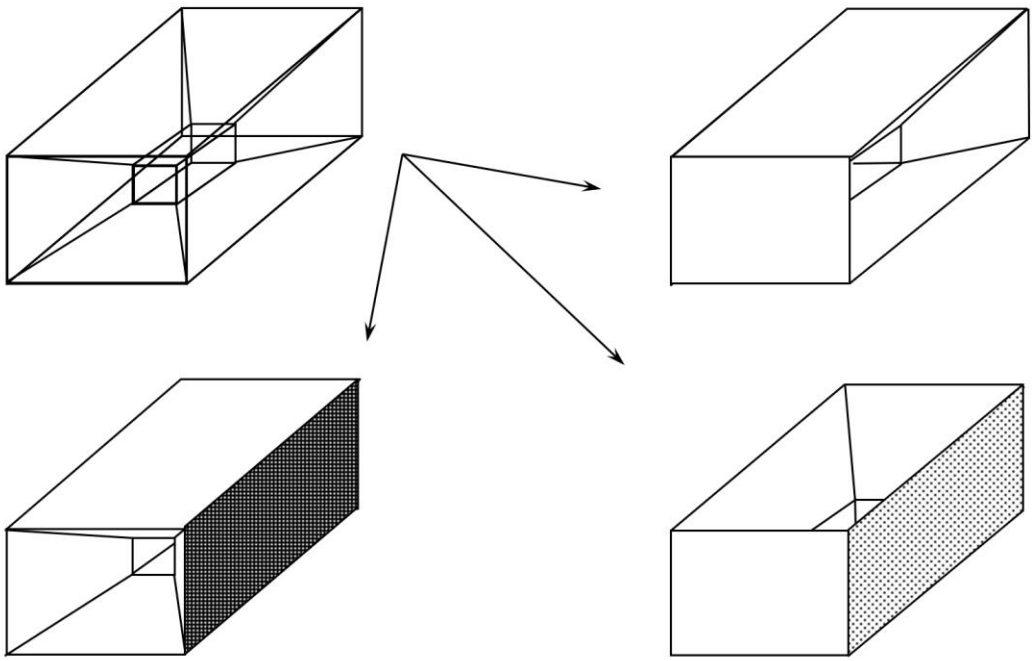
V enačbi 3.21 lahko uporabimo tudi rotacijske člene iz matrike v enačbi 3.19, da lahko prikažemo objekt iz različnih gledišč. Homogeniziran vektor lahko spremenimo v nehomogeniziran vektor tako, da vsako komponento vektorja delimo s četrto komponento vektorja in dobimo koordinato projicirane točke tako, kot je prikazano v enačbi 3.22.

$$\begin{bmatrix} x \\ y \\ 0 \\ \frac{z}{f} \end{bmatrix} :\frac{z}{f} = \begin{bmatrix} \frac{x}{\frac{z}{f}} \\ \frac{y}{\frac{z}{f}} \\ 0 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{x}{\frac{z}{f}} \\ \frac{y}{\frac{z}{f}} \end{bmatrix} \tag{3.22}$$

Takšna perspektivna projekcija je precej poenostavljena, medtem ko se v računalniški grafiki uporablja precej bolj kompleksno projiciranje, ki upošteva točko gledišča, kot pogleda in dve ravnini, ki definirata projekcijski prostor [44]. To velja tudi za vzporedno projekcijo in vedno je treba v splošnem določiti gledišče, volumen in orientacijo pogleda. Različne 3D projekcije so običajno definirane v programskih knjižnicah računalniške grafike (OpenGL, DirectX) in opisane v knjigah, ki obravnavajo računalniško grafiko [45].

Za enostavni žični model v Excelu, prikazan na sliki 3., opisane transformacije zadostujejo in omogočajo geometrijske transformacije in transformacije pogleda za prikaz v perspektivni in vzporedni projekciji. Tak modelirnik je dober primer za vajo uporabe geometrijskih transformacij in omogoča vizualizacijo enostavne geometrije.

Ko postane geometrija bolj kompleksna in želimo v modelu uporabiti tudi ploskve in poljubne površine, postane tak modelirnik precej manj uporaben. V žičnem modelirniku hitro pride do dvoumnosti in bi morali nekako določiti, kje so površine. S slike 3.4 je razvidno, da si lahko žični model razlagamo na več različnih načinov. Zato so žični modeli neuporabni, ko gre za kompleksne izdelke s številnimi površinami.



Slika 3.4. Dvoumnost žičnih modelov

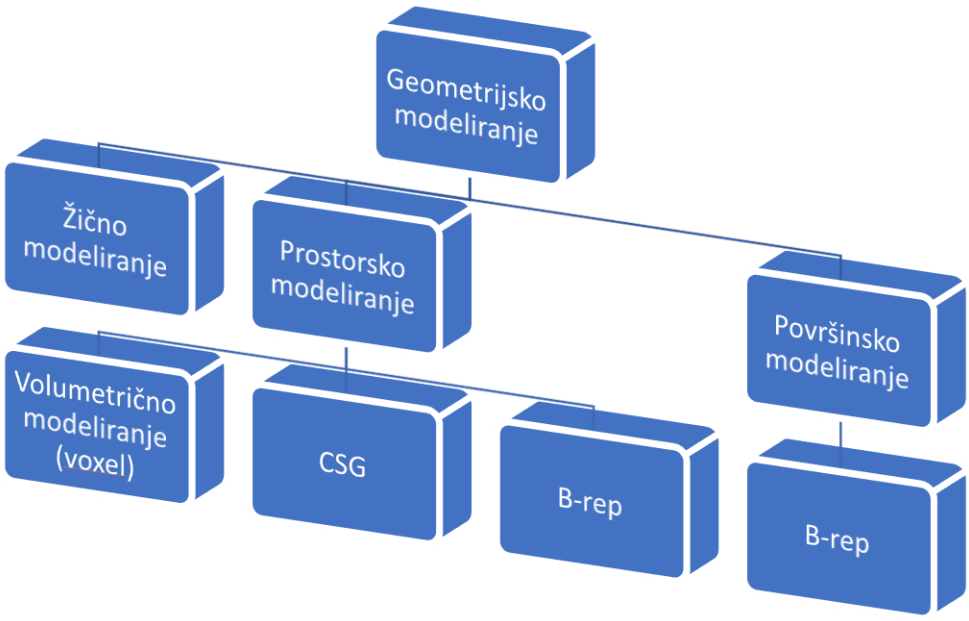
Vendar to ne pomeni, da žičnih modelirnikov danes ne uporabljamo. Za številne izdelke, na primer za jekleno konstrukcijo na sliki 3.5, je še danes bolj priporočljiva uporaba modelirnika, kjer sestavljamo linijske elemente v celovito konstrukcijo. Ti linijski elementi so lahko realizirani s palicami poljubnih prerezov in sodobni modelirniki lahko preklapljajo med linijskim in realističnim prikazovanjem izdelka, kjer so palice izrisane s površinami profilov, čeprav je interna predstavitev v obliki žičnega modela. Tudi simulacije takšnih modelov so potem veliko lažje, saj uporabljamo v numeričnih metodah linijske elemente namesto prostorskih ali površinskih. Za simulacijo takšnih linijskih elementov, kjer je ena dimenzija izrazito večja do drugih dveh, bi potrebovali veliko količino prostorskih elementov za simulacijo. Praktično bi vsaka enostavna simulacija zahtevala superračunalnik. Sodobni 3D modelirniki omogočajo tudi pretvorbo modela v žični model ali imajo vgrajene namenske žične modelirnike, s katerimi lahko takšne izdelke konstruiramo.

Slika 3.5. Primer jeklene konstrukcije

3.2 Prostorski in površinski modelirniki

Geometrijsko modeliranje omogoča tudi modeliranje prostorskih in površinskih modelov, ki ponujajo precej boljši način modeliranja realnih objektov. Geometrijsko modeliranje lahko v osnovi razdelimo na žično, prostorsko in površinsko modeliranje tako, kot je prikazano na sliki 3.6.

Najbolj zanimivo je področje prostorskega modeliranja, kjer obstaja več možnosti modeliranja. Zgodovinsko so bile ideje za različne načine modeliranja prisotne že zelo dolgo, vendar zaradi nezadostne strojne opreme niso bile uresničljive. Danes velik del modeliranja temelji na površinskih B-rep (angl. Boundary Representation) predstavitvah. Z razvojem strojne opreme, ki omogoča prostorsko renderiranje v realnem času, so danes znova precej atraktivne metode CSG (angl. Constructive Solid Geometry) in volumetričnega modeliranja.

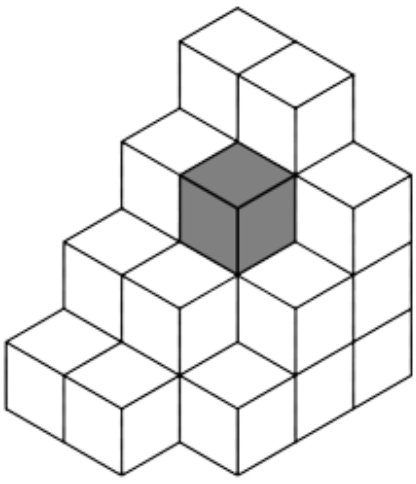


Slika 3.6. Klasifikacija modeliranja

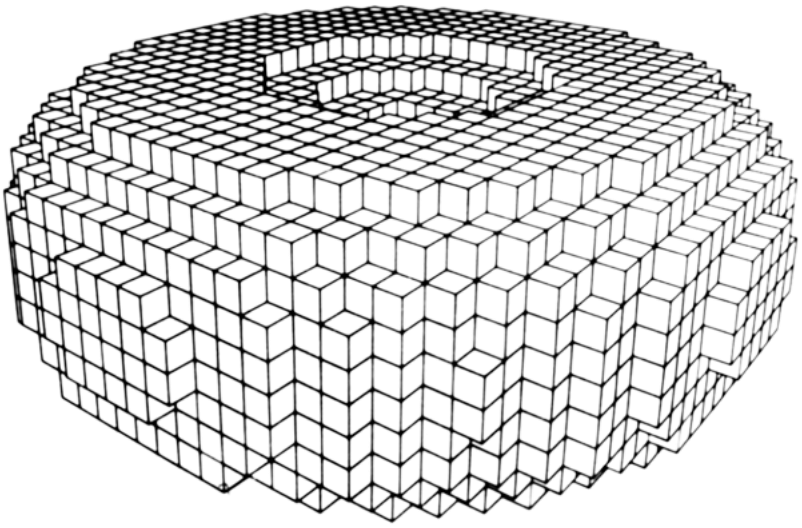
3.2.1 Volumetrični modelirniki

Osnovna ideja volumetričnega modelirnika je posnemanje realnega sveta, ki je sestavljen iz osnovnih delcev – atomov. Seveda je nemogoče modeliranje realnega sveta z delci v velikosti atomov. Tudi drugi gradniki, kot so molekule ali kristali, so še vedno veliko premajhni. Model, sestavljen iz tako ogromnega števila elementov, bi močno presegal zmogljivosti sedanjih računalnikov. Ideja je seveda primerna in tudi volumen lahko predstavimo tako, kot je v računalniku predstavljena rastrska slika. Rastrska slika je sestavljena iz pikslov (angl. pixel – Picture Element) in analogno lahko volumen predstavimo z voksli (angl. voxel – Volume Element). Termin voksel je bil najprej uporabljen v računalniški igri Commanche, kjer je Kyle Freeman razvil Voxel Space pogon za igro podjetja NovaLogic [46].

Osnovna ideja volumetričnega modeliranja je razdelitev volumna na volumske elemente oziroma voksle tako, kot je prikazano na sliki 3.7. Seveda velikost vokslov določa ločljivost modela, zato je težko zagotoviti gladkost površin in modeli imajo precej grobo površino, kar je razvidno s slike 3.8.



Slika 3.7. Voksel

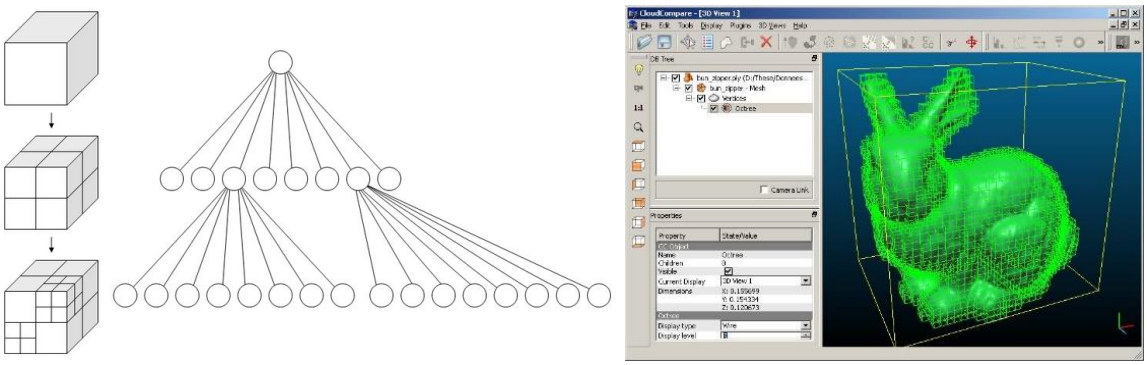


Slika 3.8. Model, predstavljen z voksli

Natančnost modela je določena z velikostjo vokslov. Pri skeniranju z računalniško tomografijo dobimo posnetke rezin, ki jih presvetlijo rentgenski žarki. Te rezine lahko nato združimo v volumetrični model tako, da je površina ene ploskve voksla enaka velikosti piksla skenirane rezine. Na desni strani slike 3.9 je prikazan volumetrični model por skenirane kroglice iz aluminijeve zlitine, prikazane na levi strani slike.

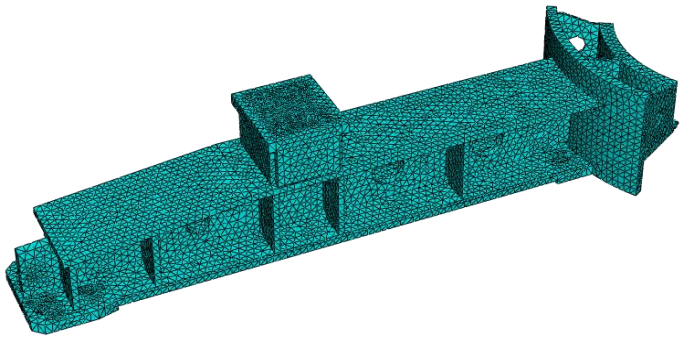
Slika 3.9. Volumetrični model por v kroglici aluminijeve zlitine

Skeniran model računalniške tomografije z ločljivostjo 1000 x 1000 x 1000 tako zahteva 1 GB pomnilnika, če v vsakem vokslu omogočamo samo 256 sivin (1Byte/voksel). Manipulacija z volumetričnimi modeli zato zahteva precej pomnilnika in zmogljive računalnike. V današnjem času, ko imamo osebne računalnike, ki imajo 64 GB ali 128 GB pomnilnika in zmogljive procesorje ter grafične kartice, to ni več problem. V preteklosti je bilo treba varčevati s pomnilnikom, zato so iskali boljše rešitve za volumetrični modelirnik. Takšna rešitev je razdelitev prostora na oktante (osem podprostorov), kjer potem opazujemo posamezen oktant. Vsak oktant potem spet razdelimo na osem oktantov, če je v prostoru posameznega oktanta prisotnih več materialov in praznin. To počnemo tako dolgo, dokler ni v posameznem oktantu samo enega materiala ali samo praznina. Struktura modela je nato zapisana v obliki osmiških dreves (angl. octree) [47] tako, kot je prikazano na sliki 3.10. Količina pomnilnika za shranjevanje modela je v tem primeru veliko manjša, vendar so voksli različno veliki.



Slika 3.10. Razdelitev prostora in drevesna struktura z oktalnimi drevesi [47, 48]

Volumetrični model z različno velikimi voksli predstavljajo tudi numerični modeli za simulacije z metodo končnih elementov. Tak model je prikazan na sliki 3.11, kjer je celoten model sestavljen iz različno velikih tetraedrov.



Slika 3.11. Numerični model za simulacijo po metodi končnih elementov

Kreiranje volumetričnih modelov iz skeniranih realnih objektov je dokaj preprosto, medtem ko je modeliranje volumetričnega modela na sliki 3.11 lahko precej zahtevno. Obstajajo nekateri modelirniki, ki omogočajo kreiranje 2D strukture elementov v ravnini in nato transformacije v prostor.

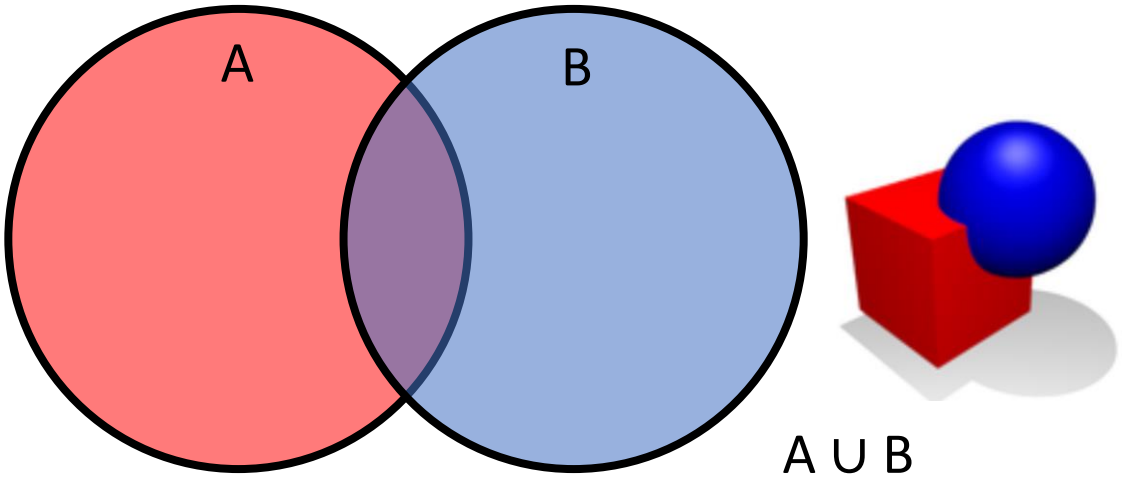
Drugi način modeliranja volumetričnega modela je postavljanje vokslov na ustrezna mesta v prostoru. Takšen način modeliranja lahko vidimo v računalniški igri Minecraft [49], ki je pravzaprav volumetričen modelirnik, namenjen za zabavo. Delo je pri tem precej podobno kreiranju slike v programu rastrske grafike, kjer spreminjamo barvo pikslov.

Volumetrični model moramo prikazati na ravninskem zaslonu, ker so volumetrični prikazovalniki (naprave za prikazovanje volumskih modelov) danes še precej redko dostopni. Prikazovanje volumetričnega modela je najenostavneje prikazati z uporabo renderiranja z metodo sledenja žarku (angl. raytracing), ki zahteva precej zmogljivo strojno opremo. V današnjem času so dosegljive grafične kartice, ki omogočajo renderiranje v realnem času (prikazovanje modela večkrat na sekundo). Seveda takšna strojna oprema še ni na voljo v vseh napravah (na primer mobilni telefoni), a vendar delujejo igre, kot je Minecraft. Nekateri volumetrični modelirniki, kot je Minecraft, zato voksle prikazujejo s poligoni, ki jih lahko prikazujejo tudi manj zmogljive grafične kartice. Pogosto tudi volumetrične modele, izdelane z računalniško tomografijo, pretvorimo v poligonske oziroma B-rep modele zaradi lažjega prikazovanja.

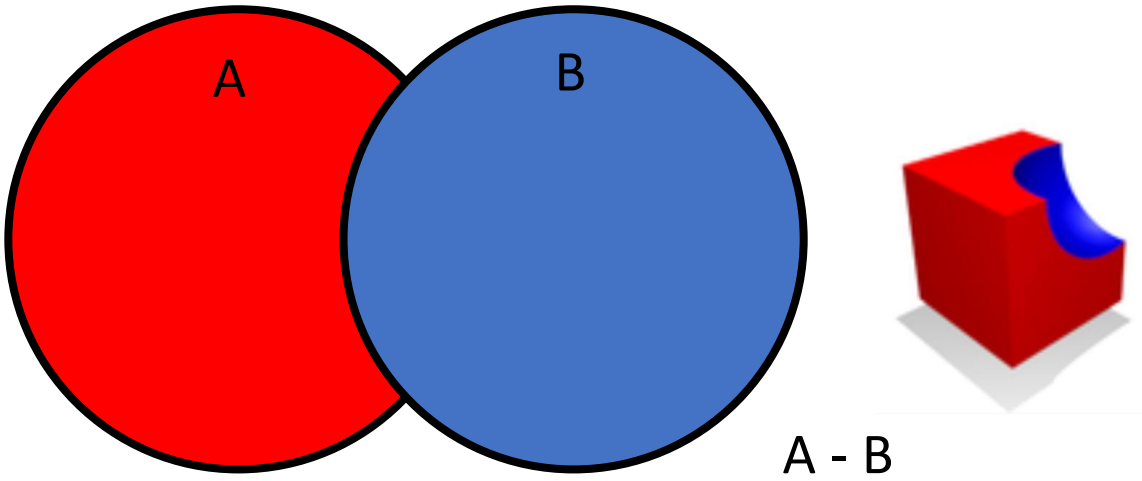
3.2.2 CSG modelirniki

Poleg sestavljanja majhnih volumskih elementov v volumetričnem modelirniku lahko poleg kvadrov sestavljamo tudi bolj kompleksna telesa, kot so sfere, kolobarji, konusi, prizme itd. Najprej so se pojavili modelirniki, kjer smo takšna telesa zgolj postavljali enega poleg drugega. Takšni modelirniki z golim postavljanjem osnovnih oblik (angl. pure primitive instancing) niso omogočali dovolj kreativnega modeliranja. Kmalu so se zato pojavili CSG modelirniki [50], ki so uporabljali iste osnovne oblike, vendar so omogočali kombiniranje oblik s pomočjo treh Boolovih operacij unija, razlika in presek.

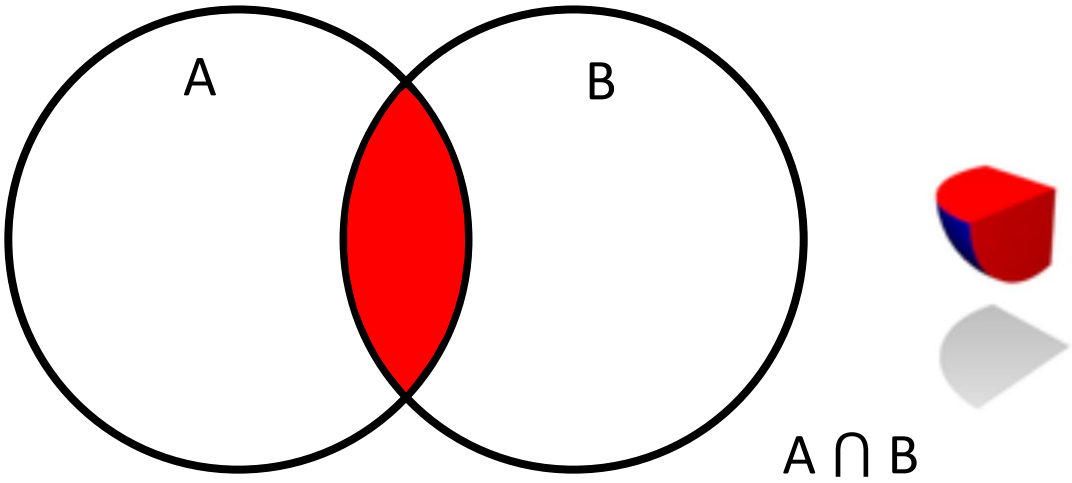
Unija deluje podobno kot pri postavljanju osnovnih oblik, le da lahko postavimo eno telo tudi znotraj drugega telesa in dobimo novo enotno telo tako, kot je prikazano na sliki 3.12. Na sliki 3.13 je prikazana operacijska razlika, na sliki 3.14 pa presek dveh teles. Rezultat vsake operacije je novo telo, ki ga lahko uporabimo v naslednji operaciji.



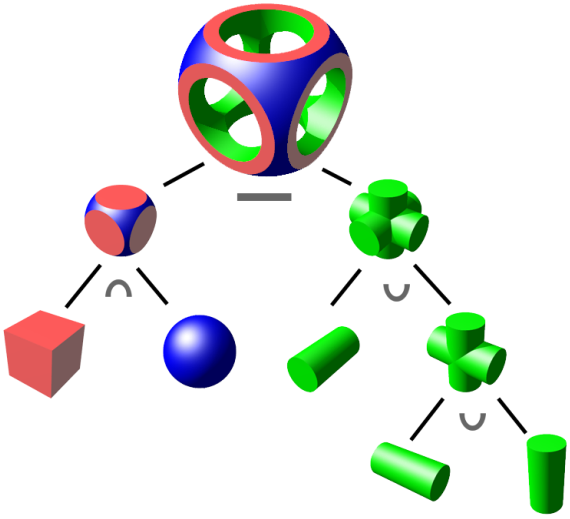
Slika 3.12. Unija dveh teles



Slika 3.13. Razlika dveh teles



Slika 3.14. Presek dveh teles



Slika 3.15. Zaporedje Boolovih operacij [51]

Tako lahko izdelamo kompleksen model z zaporedjem operacij, v katerih kombiniramo osnovne gradnike. Na sliki 3.15 je prikazan relativno kompleksen model, ki ga dobimo, ko od preseka kocke in krogle odštejemo unijo treh valjev. Jasno je, da lahko dimenzije osnovnih gradnikov poljubno določamo tako, da so možnosti za modeliranje praktično neomejene. Poleg tega takšno kombiniranje ukazov dovoljuje ogromno kreativnosti, saj lahko do želenega modela pridemo po različnih poteh. Sodobni modelirniki še danes pogosto vključujejo Boolove operacije v osnovni obliki, vendar so pogosto skrite v obliki drugačnih ukazov. Ko naredimo valjasto luknjo v prostorskem modelirniku, dejansko od modela odštejemo valj, vendar je uporabniški vmesnik v programih narejen tako, da je modeliranje bolj naravno in bolj prilagojeno razmišljanju uporabnika.

Modeliranje v CSG modelirniku temelji na logičnih osnovah in pogoj, da bodo operacije veljavne, je veljaven model po vsakem koraku operacije. Operacije vedno izvajamo na osnovnih gradnikih, ki morajo biti pravilna in zaprta telesa (angl. manifold). Nemogoče je zagotoviti, da bo operacija uspela, če bo osnovni gradnik odprto ali nepravilno telo (angl. nonmanifold). Zato v prostorskem modelirniku vedno testiramo, ali je telo še veljavno. Osnova za določanje veljavnosti je Eulerjeva karakteristika teles [52], ki določa, da preštejemo oglišča telesa, jim prištejemo število ploskev in odštejemo število robov.

Enačba 3.23, kjer V označuje oglišča (angl. vertex), E robove (angl. edge) in F ploskve (angl. face), za zaprta pravilna telesa rezultira v vrednosti 2.

$$V - E + F = 2 \tag{3.23}$$

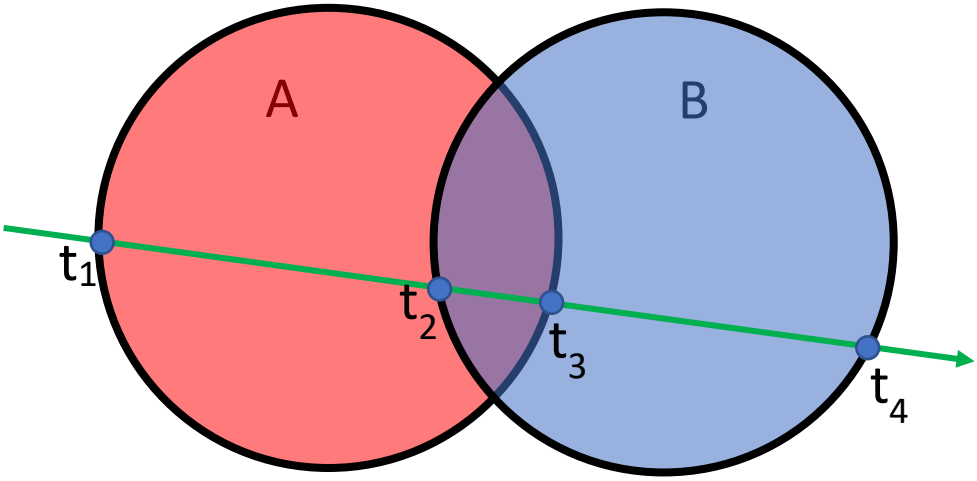
Telesa, ki niso zaprta, so različna od 2. Težava je v tem, da lahko v redkih primerih najdemo tudi odprta telesa, ki dajo rezultat 2. Zato je formula uporabna za ugotavljanje, ali gre za nepravilno oziroma odprto telo. Podobno velja za nekoliko izboljšano enačbo 3.24, ki se imenuje Euler-Poincarjeva karakteristika [53], kjer so V, E, F prevzeti iz enačbe 3.23. Dodano je število zank robov L (angl. loop), število školjk oziroma volumnov (školjka je polno telo in tudi praznina v telesu), označeno s S (angl. shell), ter število lukenj, ki predirajo telo, označeno z G (angl. genius).

$$V - E + F - (L - F) - 2(S - G) = 0 \tag{3.24}$$

Enačba 3.24 je enaka 0 za veljavna zaprta telesa, vendar se izkaže, da je enačba enaka 0 tudi za telo na sliki 3.16. V tem primeru je po operaciji ostala v telesu ploskev, kar seveda pomeni problem in vsaka nadaljnja operacija modeliranja je nemogoča. Zato modelirniki uporabljajo nekoliko bolj kompleksne načine preverjanja pravilnosti rezultata operacij modeliranja.

Slika 3.16. Nepravilno telo, za katerega dobimo 0 v enačbi 3.24

Rezultat modeliranja v CSG modelirniku je natančno določen, če operacije izvajamo s pravilnimi telesi in je rezultat operacije pravilno telo. To pa ne pomeni, da vidimo, kakšno telo je rezultat operacije. Za prikaz rezultata lahko uporabimo sledenje žarku tako, da upoštevamo rezultat Boolovih operacij. Na sliki 3.17 bo na površini točka t_1 in t_4, če bo operacija unija. Ko bomo upoštevali operacijo razlike A-B, bosta na površini točki t_1 in t_2. Ko bo rezultat operacije presek med A in B, bosta na površini točki t_2 in t_3.



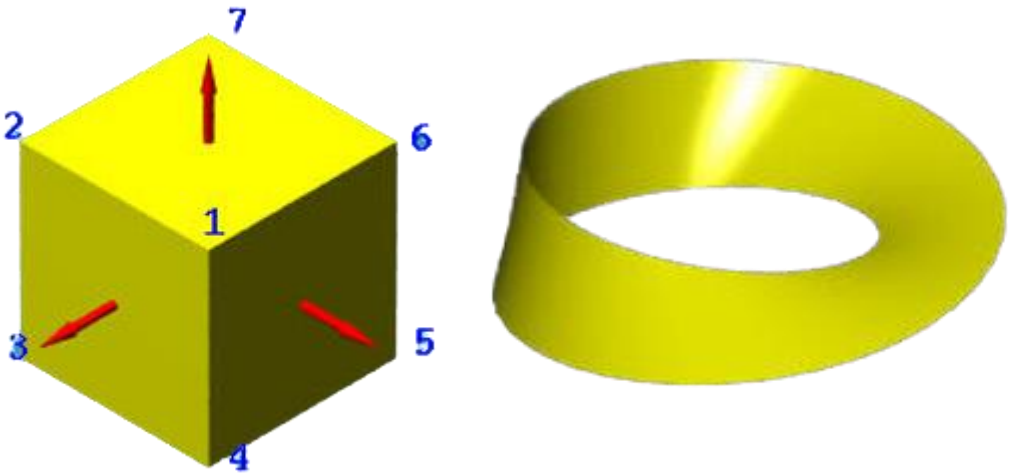
Slika 3.17. Sledenje žarku in Boolove operacije

Danes je mogoče realizirati CSG modelirnike, ki bi v vsakem trenutku renderirali model z metodo sledenja žarku. V preteklosti to ni bilo mogoče in renderiranje posamezne Boolove operacije je trajalo predolgo, da bi lahko tak modelirnik smiselno uporabljali. Zato je bilo treba po vsaki operaciji določiti površino v obliki B-rep tako, da se določijo poligoni, ki jih je potem mogoče precej preprosto prikazati. Zato je danes večina modelirnikov tipa B-rep, čeprav vključujejo tudi način modeliranja CSG.

3.2.3 B-rep prostorski in površinski modelirniki

Predstavitev modela s površinami B-rep (angl. boundary representation) [54] je razvil Ian Braid v okviru načrtovanja 3D modelirnika Build. Ta modelirnik je bil osnova za poznejše knjižnice Parasolid in ACIS, s katerima je izdelana večina sodobnih modelirnikov. Tudi modelirniki, ki niso osnovani na teh knjižnicah, uporabljajo enak princip in zato je danes velika večina modelirnikov tipa B-rep.

Osnovna ideja te predstavitve je v tem, da so vsa telesa predstavljena s površinami. Te so pogosto matematično zapisane v obliki parametričnih površin zaradi shranjevanja natančne geometrije. Tisto, kar uporabnik vidi na zaslonu, so poligoni, s katerimi so v končni fazi predstavljene površine. Kocko na sliki 3.18 tako predstavimo s šestimi poligoni v obliki kvadratov.



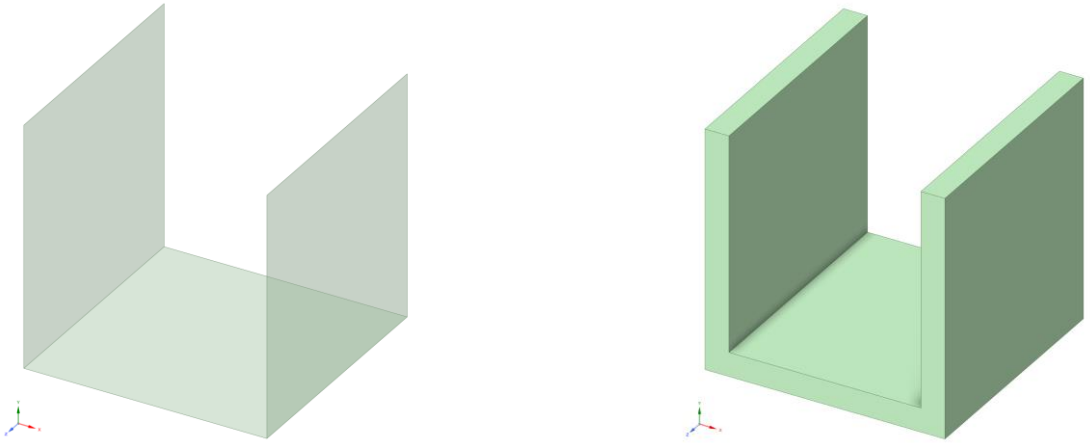
Slika 3.18. Predstavitev s površinami

Ni nujno, da so poligoni na kocki na sliki 3.18 vedno kvadrati. Precej bolj pogosto so poligoni trikotniki, ki jih grafične kartice najlažje izrišejo. Uporabnik robov teh poligonov ne vidi, zato je popolnoma vseeno, kakšne oblike so.

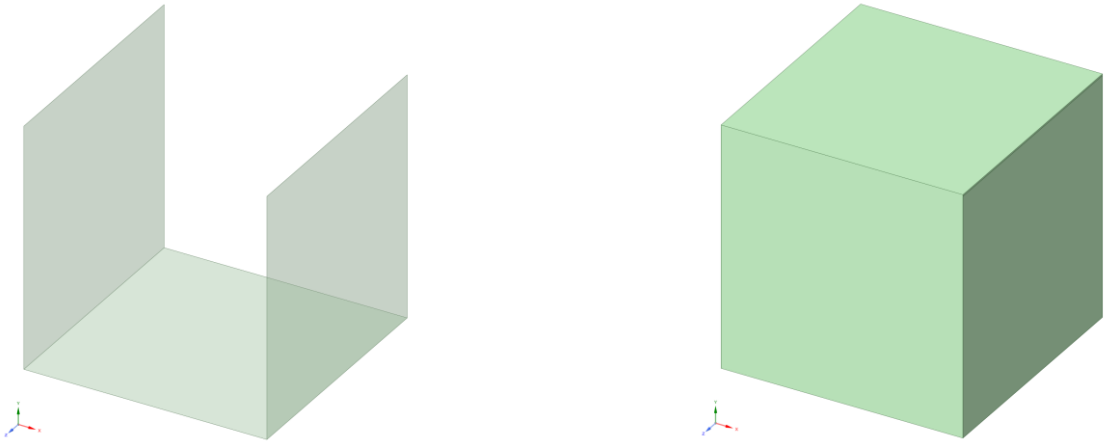
Predstavitev s površinami tudi nima težav z dilemo odprtega ali zaprtega telesa. Zato takšna predstavitev omogoča prikazovanje zaprtih teles, kot je na primer kocka, in odprtih površin, kot je površina na desni strani slike 3.18. Današnji modelirniki omogočajo površinske ukaze, s katerimi lahko izdelamo oblikovno zelo kompleksne površine. S temi ukazi naredimo raztege in rotacije v prostoru. Ko v prostor raztegnemo ploskev, dobimo polno telo, in ko v prostor raztegnemo krivuljo, dobimo odprto površino. Prostorski ukazi so lahko tudi bolj kompleksni in omogočajo kreiranje kompleksne prostorske geometrije, ki se lahko z vsakim dodatnim ukazom nadalje preoblikuje.

Odprte površine, ki jih lahko oblikujemo in prikažemo v modelirniku, v naravi ne obstajajo. Ko modeliramo te površine, so brez debeline, zato jih ni mogoče izdelati. Izdelamo jih lahko samo, če jih preoblikujemo v polno telo, in to lahko naredimo na dva načina. Površini lahko določimo debelino ali površine modeliramo tako, da zaprejo prostor. Na sliki 3.19 je prikazana pretvorba v površine z

debelino (angl. thick surface). Na sliki 3.20 so iste površine uporabljene za kreiranje prostorskega telesa z dodajanjem treh ploskev, ki zaprejo prostor.



Slika 3.19. Pretvorba v površine z debelino



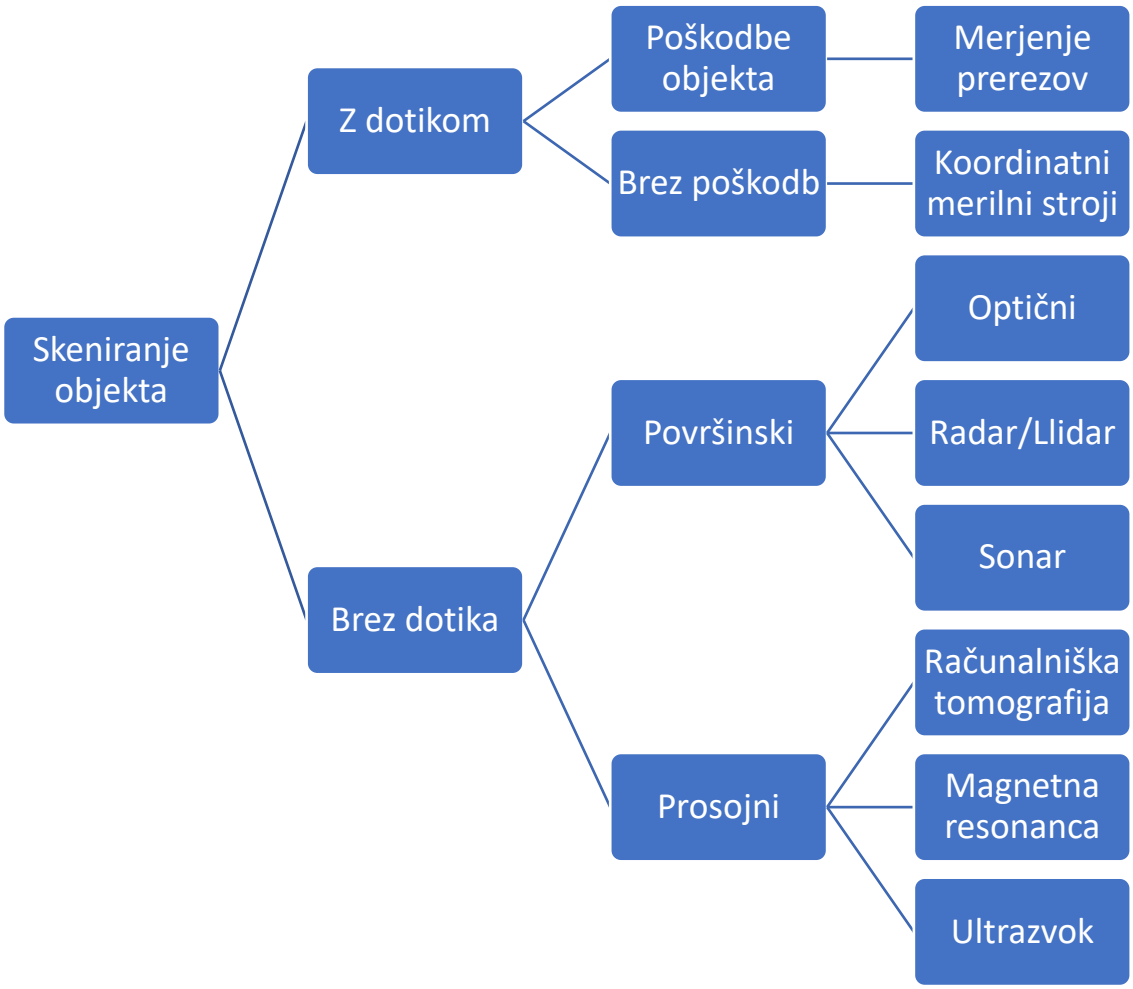
Slika 3.20. Pretvorba v prostorsko telo

4 Poligonske mreže

V modelirnikih B-rep je površina predstavljena s poligoni. Ker povezani poligoni spominjajo na ribiško mrežo, jih pogosto naslavljamo z imenom poligonska mreža. Takšne poligonske mreže dobimo po skeniranju površin ali jih kreiramo z modelirniki. Poligonski modelirniki so modelirniki B-rep, kjer je končni cilj izdelava modela, izdelanega s poligoni, saj bo model uporabljen samo za računalniške predstavitve v računalniških igrah ali videoposnetkih.

4.1 3D skeniranje

Danes je na voljo veliko možnosti za skeniranje realnih objektov. Najpreprostejša rešitev je ročno zajemanje podatkov o objektu, ki zahteva zgolj uporabo ročnih meril. Tako lahko le merimo izdelek in zapisujemo mere na skico. Seveda je veliko bolj natančno merjenje z namenskimi napravami, ki omogočajo zajem geometrije objekta. Na sliki 4.1 je prikazan pregled različnih metod za pridobivanje geometrije objekta. Metode, ki pridobivajo podatke z dotikom objekta, so precej zamudne in v primeru rezanja objekta je objekt uničen po postopku. Tudi merjenje s koordinatnimi stroji je precej zamudno in omogoča pridobivanje relativno majhnega števila točk površine.



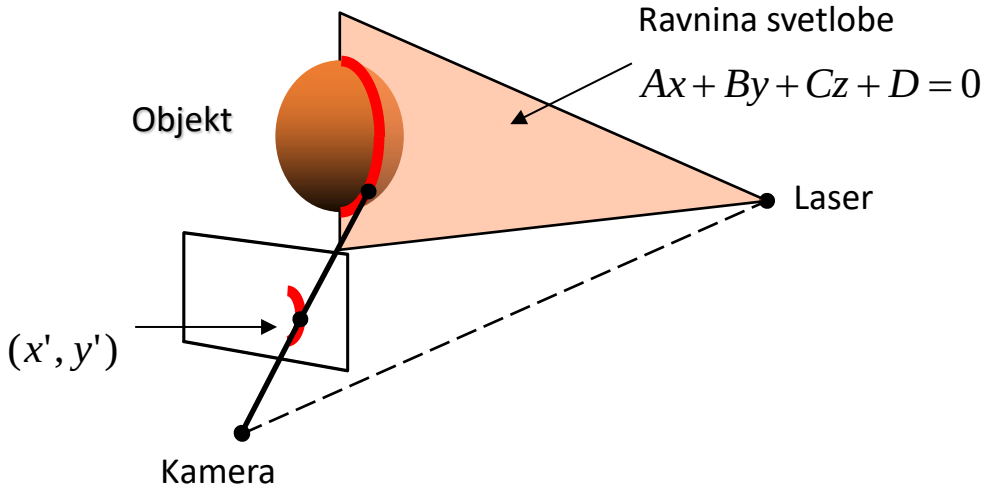
Slika 4.1. Načini za zajemanje geometrije objekta [55]

V nadaljevanju se bomo posvetili predvsem skeniranju objekta brez dotika, kjer bodo opisane nekatere metode za površinsko in prosojno skeniranje objektov.

4.1.1 Površinsko skeniranje

Optično skeniranje je najbolj običajen način za zajemanje geometrije objekta. Tudi na tem področju je več načinov skeniranja. Nekateri načini so aktivni, kar pomeni, da spreminjajo površino objekta skeniranja s projiciranjem žarkov ali vzorcev. Drugi načini skeniranja so pasivni in v ničemer ne posegajo na skenirano površino. Pasivni način je načeloma bolj zahteven, saj skeniranje objekta običajno zahteva prepoznavanje istega dela površine potem, ko se spremeni kot pogleda na objekt. Zato so najprej nastali aktivni načini skeniranja, kjer se z laserskim žarkom izbere del površine, ki jo nato izmerimo z zajemanjem s kamero.

Na sliki 4.2 je prikaza shema, na kateri z laserskim žarkom narišemo linijo na objekt, s kamero pa določamo koordinate točk na projicirani krivulji. Znana je ravnina, v kateri projiciramo laserski žarek. V kameri dobimo projekcijo slike v perspektivni projekciji, kjer je f goriščna razdalja leče. Če obrnemo enačbe perspektivne projekcije 3.22, dobimo enačbe 4.1. Iz presečišča premice žarka svetlobe in ravnine laserja lahko nato določimo razdaljo do točke na skenirani površini objekta.



Slika 4.2. Skeniranje z laserjem

$$x = x'z/f$$
$$y = y'z/f \tag{4.1}$$

$$z = \frac{-Df}{\mathrm{A}x' + \mathrm{B}y' + Cf} \tag{4.2}$$

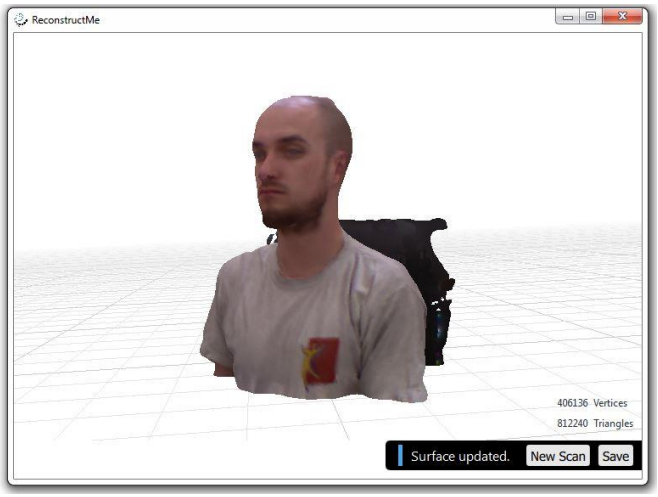
Ko premikamo laserski žarek po površini objekta, lahko določimo koordinate poljubne točke na površini. V ta namen je treba seveda vrteti objekt, zato so takšni skenerji pogosto opremljeni z vrtečo platformo, na katero postavimo objekt.

Obstajajo številne naprave za lasersko skeniranje in lasersko triangulacijo, s katerimi določamo točke na površini ali merimo razdaljo do površine. Namesto črte, ki jo ustvari laserski žarek, lahko to črto naredimo tudi kako drugače. Najbolj preprosto je narediti črto s senco palice na soncu. Namesto sence lahko projiciramo vzorec na površino [56]. Vzorci so lahko preprosti črno-beli za osnovno kalibracijo. Vzorci so lahko tudi bolj kompleksni in vsebujejo različno kodiranje za različne dele površin, da je identifikacija točke na površini lažja.

Najpreprostejša je uporaba takšnega skeniranja s projiciranjem svetlobe iz različnih smeri [57]. Pri tem lahko uporabimo aplikacijo na mobilnem telefonu, ki v temnem prostoru s svetlobo zaslona osvetli objekt iz štirih smeri in skonstruira 3D sliko objekta. Na sliki 4.3 je bil telefon uporabljan na obrazu, ki ga lahko po skeniranju preberemo v CAD modelirniku in mu dodamo poljubno teksturo. Takšni preprosti načini zahtevajo posebne pogoje osvetlitve in so manj natančni, zato v poplavi cenenih in natančnih skenerjev niso več uporabni.

Slika 4.3. Skeniranje z osvetlitvijo zaslona na mobilnem telefonu

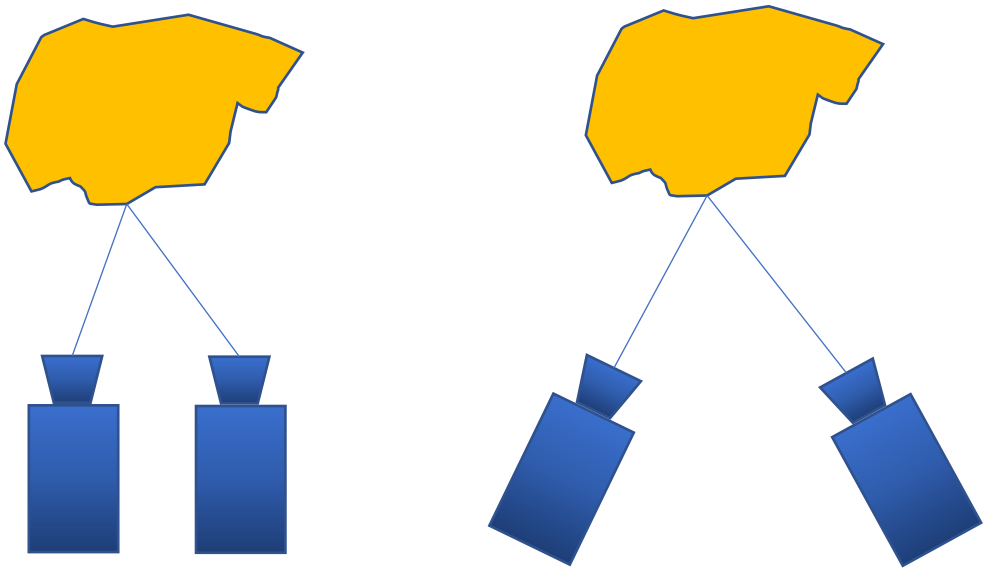
Skeniranje s projiciranjem lahko deluje tudi tako, da projiciramo z infrardečo svetlobo in za zajemanje uporabimo infrardečo kamero. Zelo razširjen je bil tak sistem v okviru igralne konzole Xbox, ki je vsebovala dodatek Kinect [58], ki je omogočal zajemaje gibov igralca. Podoben sistem uporabljajo spletne kamere, ki omogočajo prepoznavanje obraza za avtentifikacijo v operacijskem sistemu Windows s sistemom Hello ali na nekaterih mobilnih telefonih. Z ustrezno programsko opremo je mogoče tak sistem spremeniti v 3D skener [59]. Pri tem uporabimo Kinect sistem v povezavi z Windows osebnim računalnikom in lahko skeniramo celoten prostor, ki je v vidnem polju infrardeče kamere. Posledično dobimo 3D model poljubnega objekta, ki ga postavimo pred kamero. Na voljo je kar nekaj različne programske opreme za skeniranje s sistemom Kinect. Slika 4.4 je izdelana s programom RecostructMe [60], ki ima vgrajeno funkcijo za izdelavo 3D modela uporabnika.



Slika 4.4. Skeniranje s sistemom Kinect [60]

Najpogostejši optični skenerji danes delujejo po načelu paralakse [61]. Na tem načelu delujejo oči, s katerimi ocenimo razdaljo do objektov. Paralakso uporabljajo tudi v astronomiji, kjer so že v 19. stoletju izmerili razdaljo do zvezd. Takšno meritev opravimo v različnih obdobjih, ko je Zemlja enkrat na eni strani sonca in drugič na drugi strani. Isto zvezdo iz Zemlje vidimo pod različnimi koti in glede na razdaljo med obema položajema Zemlje in kotom lahko izračunamo razdaljo do zvezde. Enako načelo

lahko uporabimo za skeniranje objekta, kjer merimo razdaljo do točke na objektu. V ta namen uporabimo dve kameri, ki sta lahko postavljeni vzporedno ali pod kotom tako, kot je prikazano na sliki 4.5. Če poznamo razdaljo in kot med kamerama, lahko s trigonometrijo določimo razdaljo do točke.



Slika 4.5. Skeniranje z dvema kamerama

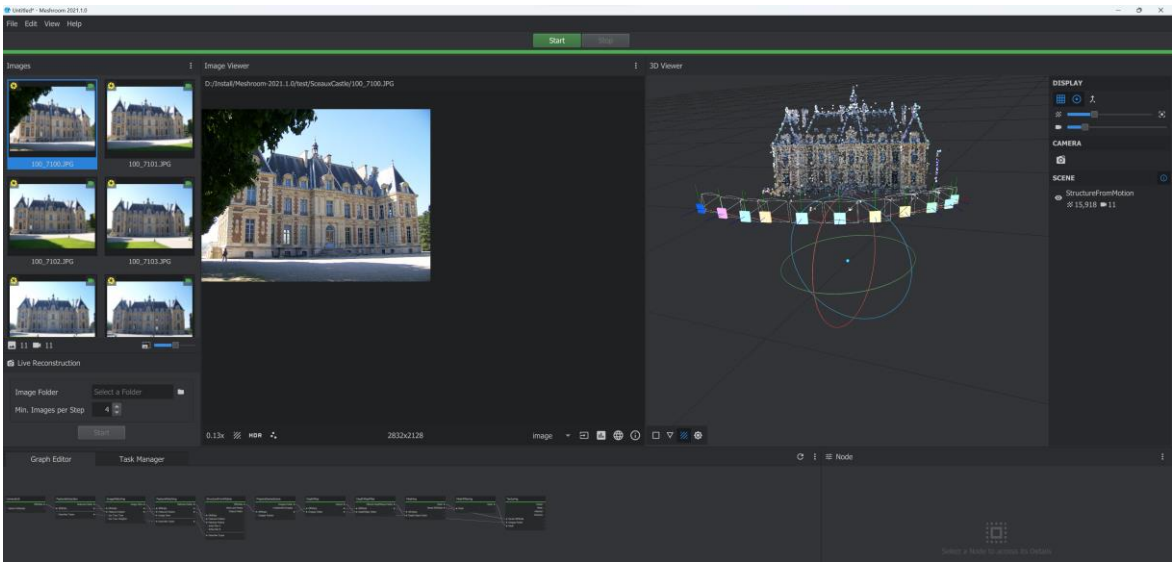
Glavna težava pri tem je določanje iste opazovane točke na objektu. V ta namen je treba narediti kalibracijo z metodo, ki identificira točko na površini. Nekateri skenerji uporabijo nalepke na površini, prikazane na sliki 4.6, za določanje točk na površini. Nalepke so koristne tudi za identifikacijo iste točke, ko objekt obrnemo in ga skeniramo z druge strani.

Slika 4.6. Nalepke na površini

Skeniranje poteka po površini okoli nalepke, na mestu nalepke površina običajno ni skenirana in jo je treba pozneje rekonstruirati. Najpreprosteje za uporabnika je uporaba skeniranja na izdelku, ki ga ni treba opremljati z nalepkami. Namesto tega sodobni 3D skenerji prepoznajo vzorec na površini objekta in tako identificirajo isto točko v obeh kamerah. V nadaljevanju lahko primerjajo tudi slike objekta, ki je obremenjen, in izračunajo pomike na površini objekta. Tak postopek se imenuje DIC (korelacija digitalnih slik – angl. Digital Image Corellation) [62]. Pri tej metodi je lahko težava gladka površina, na kateri ni mogoče prepoznati značilnih vzorcev na površini.

Podobno kot skeniranje s paralakso deluje postopek, ki se imenuje fotogrametrija [63], kjer 3D objekt skeniramo na podlagi fotografij objekta. Fotografije istega objekta so posnete iz različnih kotov, tako da je posneta celotna ali vsaj večji del površine objekta. Rezultat je precej odvisen od kakovosti

fotografij, ki morajo biti dobro osvetljene in posnete tako, da je mogoče najti iste točke na različnih fotografijah.

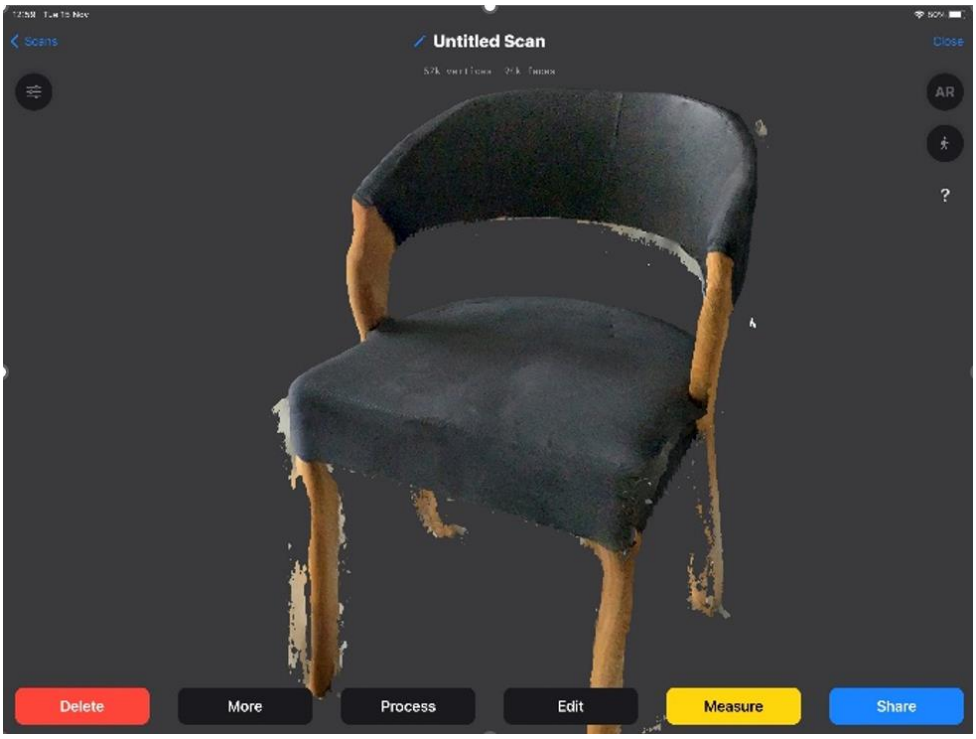


Slika 4.7. Rekonstrukcija iz posnetkov s programom Meshroom

Na sliki 4.7 je prikazana rekonstrukcija modela iz slik s prosto dostopnim programom Meshroom. Na levi strani slike so fotografije, posnete iz položajev, ki jih prikazujejo kvadratki na desni. V ozadju desnega okna je prikazan oblak točk, rekonstruiran iz fotografij. Kakovost dobljenega oblaka točk in hitrost rekonstrukcije je precej odvisna tudi od kakovosti programske opreme, ki mora poiskati enake vzorce pikslov na različnih slikah. Na spletni strani [63] so navedeni številni prosto dostopni in plačljivi programi za rekonstrukcijo modela na podlagi posnetih fotografij objekta.

V današnjem času se vedno bolj uveljavlja 3D skeniranje, ki temelji na postopku LIDAR [64]. Za razliko od optičnega skeniranja, kjer računamo razdaljo glede na geometrijo žarkov, LIDAR deluje podobno kot radar. Na površino pošljemo vir svetlobe in merimo čas, ko zaznamo odboj svetlobe od površine. Glede na čas odboja tako določimo razdaljo do površine z upoštevanjem hitrosti svetlobe. Vir svetlobe je pogosto laser LIDAR (angl. Laser Imaging, Detection And Ranging), vendar lahko uporabimo tudi druge vire svetlobe, zato L v kratici predstavlja angleško besedo za svetlobo LIDAR (angl. LIght Detection And Ranging). Natančnost določanja točk površine je odvisna od števila meritev, ki jih z LIDAR-jem opravimo. LIDAR uporabljajo tudi za skeniranje zemeljske površine, ko z letalom posnamejo pokrajino, ki jo letalo preleti. Tako zbrani podatki omogočajo izdelavo natančnih zemljevidov, ki se pogosto obnavljajo in jih uporabljamo v geodetske namene. V zadnjem času so te naprave dostopne tudi na mobilnih telefonih in tablicah, s katerimi lahko posnamemo prostor ali objekt v prostoru. Na sliki 4.8 je prikazan stol, skeniran s tablico Ipad pro, ki omogoča skeniranje s tehnologijo LIDAR. Ker ima tablica tudi običajno kamero, lahko program uporabi naravne barve objektov, ki jih potem prikaže na skeniranem modelu.

Po pretvorbi v enega od formatov za oblak točk teh informacij ni shranjenih in dobimo samo oblak točk brez tekstur. Tak model, prikazan na sliki 4.9, lahko preberemo s 3D pregledovalnikom v operacijskem sistemu Windows ali s katerimkoli programom, ki omogoča branje STL datotek. Seveda je na skeniranem modelu običajno veliko napak, ki jih odpravimo z bolj kakovostnim skeniranjem in s programi za obdelavo oblakov točk.



Slika 4.8. 3D skeniranje LIDAR na Ipad pro

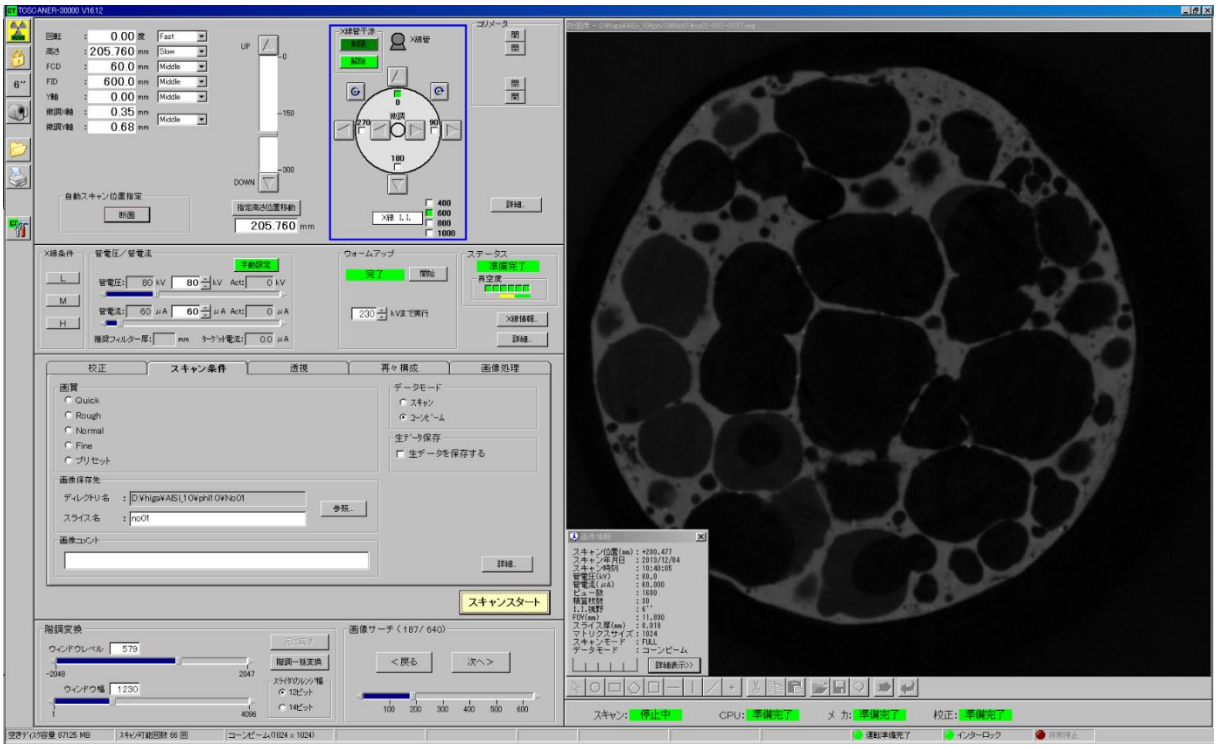
Slika 4.9. STL model skeniranega stola

Za skeniranje površin je že danes veliko možnosti in kmalu bo to še ena tehnologija, ki jo bomo imeli ves čas pri roki na mobilnem telefonu in bo 3D skeniranje objekta tako običajno, kot je danes na primer panoramsko fotografiranje. Namesto panoramske slike bomo dobili 3D model, ki ga bomo lahko pozneje opazovali iz različnih pogledov.

4.1.2 3D skeniranje pod površino

Najbolj znana metoda za zajemanje podatkov pod površino je slikanje z rentgenskimi oziroma X žarki. Pri tem je uporabljeno elektronsko zajemanje žarkov z računalniško obdelavo podatkov. Zato se tako skeniranje imenuje računalniška tomografija oziroma CT (angl. Computed Tomography) [65]. Pogosto najdemo tudi naprave, ki se imenujejo mikroCT ali nanoCT, kar označuje natančnost naprav, ki omogočajo skeniranje z ločljivostjo mikrometra ali nanometra. Alternativno lahko za skeniranje uporabimo tudi magnetno resonanco MRI (angl. Magnetic Resonance Imaging) [66], vendar je ta metoda primerna le za objekte, ki vsebujejo veliko vodika. Magnetna resonanca namreč z magnetnim poljem spreminja orientacijo vodikov atomov, ki jih lahko potem zaznamo v objektu skeniranja. Ker so organizmi večinoma sestavljeni iz vode, je uporaba MRI zelo pogosta v medicini, in predeli, ki vsebujejo veliko vodika, so na slikah bolj temni. Obstajajo še druge metode, kjer za preslikavo uporabljamo žarke različnih valovnih dolžin ali obsevanje s pozitroni ali ioni. V določenih primerih lahko pridobimo podatke pod površino tudi z ultrazvokom.

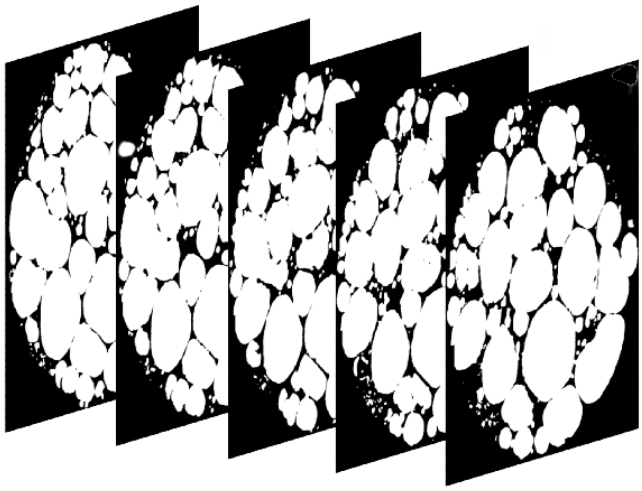
V vseh primerih namesto preslikave celotnega objekta snemamo posamezne rezine in rezultat je niz rastrskih slik. Piksli posameznih slik pri računalniški tomografiji določajo raven osvetlitve z X žarki. Popolnoma bel piksel pomeni, da žarek sploh ni prodrl skozi material na tem mestu. Črn piksel pa pomeni, da je žarek neovirano prispel do senzorja. Jakost žarkov mora biti zato prilagojena gostoti materiala, tako da za materiala z večjo gostoto uporabimo močnejše vire sevanja. Z ustrezno kalibracijo so potem praznine temne in materiali z različno gostoto obarvani z različnimi sivinami. Na sliki 4.10 je prikazana ena rezina kroglice s slike 3.9, skenirana z mikroCT skenerjem.



Slika 4.10. Ena rezina mikroCT skeniranja

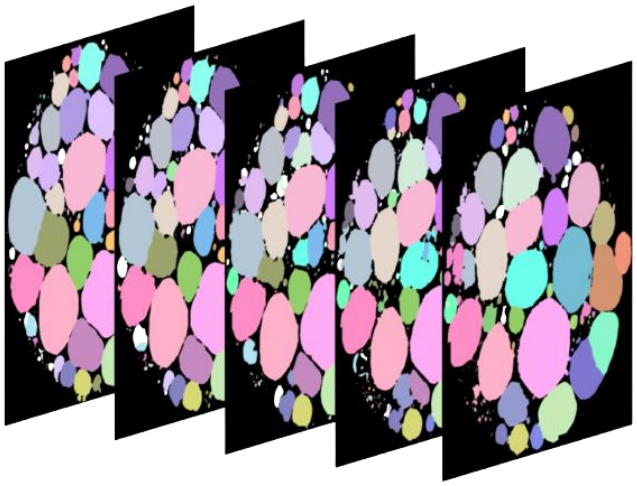
Število skeniranih rezin je odvisno od koraka skeniranja v smeri z. Običajno je na podlagi teh rezin narejen volumetrični model, zato je primerno, da je voksel enak pikslu v obliki kocke. To je težko zagotoviti med skeniranjem in pogosto se ustvarijo vmesne rezine z interpolacijo pikslov sosednjih rezin. Tako dobimo pri skeniranju pod površino oblak točk v obliki pikslov, ki so različno svetli in so urejeni v obliki niza rastrskih slik.

Delo po skeniranju pod površino se zato precej razlikuje od dela z običajnim oblakom točk in je precej bolj podobno delu z rastrskimi fotografijami, vendar mora programska oprema omogočati delo z nizi slik in z voksli. Obstaja nekaj prosto dostopnih programov, kot je na primer Fiji oziroma ImageJ [67], medtem ko je komercialna programska oprema precej draga. Skenirane slike je treba najprej obdelati z različnimi filtri, od katerih je najpomembnejši postopek segmentacije, ki določa mejo med materialom in praznino. Praznina namreč ni vedno čisto črna in material ima lahko več vrednosti sivine. Filter, ki je običajno imenovan prag (angl. threshold), določa vrednost sivine, pri kateri je meja med različnimi materiali. Piksli, ki so temnejši (imajo nižjo vrednost) od praga, so praznina, medtem ko svetlejši piksli določajo material. Mogoče je določiti več mejnih vrednosti in razlikovati med materiali različnih gostot. V enostavnem primeru, ko gre samo za en material, lahko slike tudi binariziramo in na podlagi mejne vrednosti določimo samo dve možnosti za barvo piksla, črno ali belo. Na sliki 4.11 je prikazan binariziran niz slik celične strukture, kjer nas zanimajo samo pore znotraj materiala. Če želimo analizirati pore, je treba uporabiti različne algoritme, in prvi korak zahteva, da prikažemo samo pore, zato so te prikazane z belo barvo. Material in okolica strukture predstavljajo črni piksli.



Slika 4.11. Binariziran niz slik

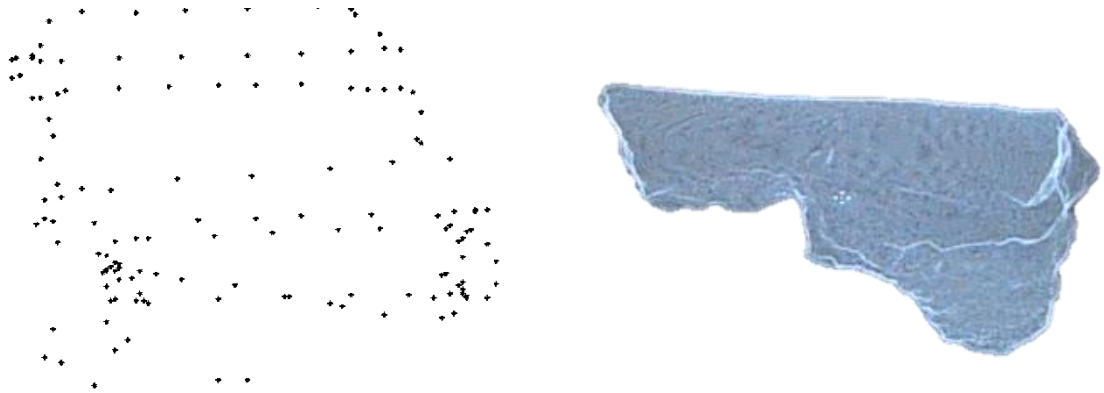
S posebnimi algoritmi potem pore med seboj ločimo oziroma segmentiramo tako, kot je prikazano na sliki 4.12, da ima vsaka pora svojo barvo. Pri tem je treba upoštevati, da se ista pora razteza čez več slik in ni dovolj analizirati samo ene slike, ampak je treba analizirati volumetrični model.



Slika 4.12. Segmentiran niz slik

4.2 Oblaki točk in poligoni

Rezultat 3D skeniranja površine je običajno oblak točk. Točke za razliko od pikslov nimajo dimenzije in barve in so predstavljene samo s koordinato v prostoru. Kljub temu lahko razlikujejo različne gostote oblakov točk. V nekaterih oblakih točk so razdalje med točkami relativno velike, v drugih pa zelo majhne. Na levi strani slike 4.13 je skenirana pokrajina s skeniranjem z napravo GPS (angl. Global Positioning System). GPS naprave so danes praktično v vsakem mobilnem telefonu in z njimi lahko s pomočjo satelitov določimo trenutno lokacijo na površini Zemlje. Natančnost naprav je nekoliko omejena in brez posebne korekcije je natančnost naprav približno 5 m. Zato je nesmiselno snemanje površine zelo pogosto in rezultat snemanja so točke, ki so ločene z nekaj metri razdalje. To običajno zadostuje za izdelavo zemljevida, vendar so zanemarjene morebitne spremembe na površini med dvema točkama in se zato takšna pokrajina nekoliko razlikuje od dejanske pokrajine.



Slika 4.13. Gostota oblaka točk

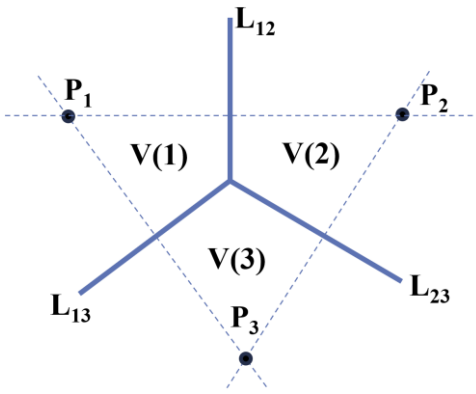
Na desni strani slike 4.13 je število točk tako veliko, da je praktično nemogoče ločiti posamezne točke, tudi če jih narišemo z najmanjšo velikostjo. Običajno dobimo takšno gostoto oblaka točk s kakovostnimi skenerji, ki določajo površino na mikroskopski ravni. Problem pri tako natančnem skeniranju je to, da na skeniranje površine vpliva hrapavost površine, kar onemogoča rekonstrukcijo gladkih površin.

Goste oblake točk običajno reduciramo, da je gostota manjša, redke oblake točk pa gladimo. Oba procesa se izvajata navadno šele potem, ko so točke v oblaku točk povezane s poligoni. Zato je vedno prvi korak obdelave oblaka točk teselacija oziroma kreiranje poligonov, ki povezujejo ločene točke. Descartes je v 17. stoletju poskušal vesolje modelirati s posameznimi točkami, ki so imele vplivno območje. To idejo je nadgradil Dirichlet sredi 19. stoletja, zato se ti poligoni včasih imenujejo tudi Dirichletove regije. Takšne regije so bile uporabljene v različne namene in najbolj znan je primer zdravnika Johna Snowa v Londonu, ki je tak diagram uporabil za lokaliziranje vzroka za epidemijo kolere. Vsak vodnjak v Londonu je bil ene točka in regija okoli vodnjaka je določala vplivno območje vodnjaka. Na podlagi pojavnosti bolezni je bilo mogoče odkriti, kateri vodni viri so kontaminirani.

V začetku 20. stoletja je ukrajinski matematik Georgij Voronij podrobno raziskal to področje za izdelavo poligonov iz poljubnih oblakov točk v n-prostoru. Zato imenujemo te poligone Voronijev diagram [68]. Ideja za Voronijev diagram je relativno preprosta, saj je treba le razdeliti prostor tako, da je v vsakem podprostoru ena točka. Razdelitev na podprostora lahko zapišemo z enačbo 4.3, ki določa presek polravnin, ki so povezane s to točko. Pri tem V(i) predstavlja Voronijevo regijo, ki je določena s presekom vseh polravnin H med sosednjimi točkami P v oblaku točk.

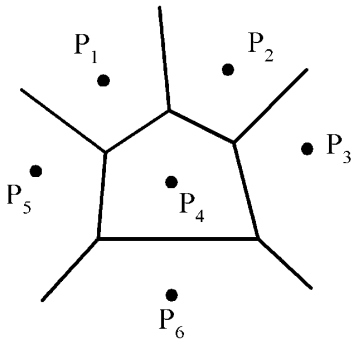
$$V(i) = \cap_{i \neq j} H(P_i, P_j) \tag{4.3}$$

V primeru na sliki 4.14 vidimo tri točke v prostoru, ki jih razdelimo z linijami tako, da je vsaka linija L pravokotna na premico, ki povezuje dve točki in razpolovi prostor med njima.



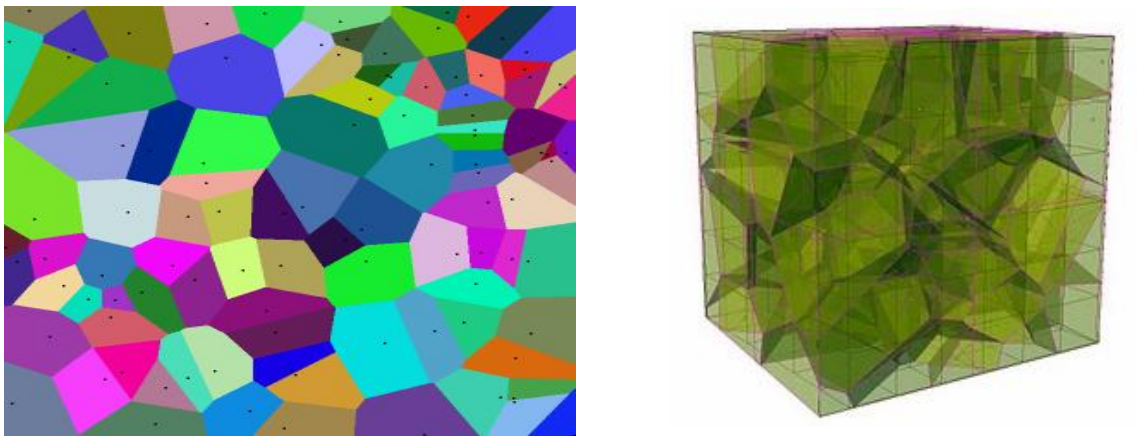
Slika 4.14. Voronijev diagram za tri točke

Dodatne točke generirajo dodatne linije, dokler ni celoten prostor okoli točke obdan s poligonom tako, kot je prikazano na sliki 4.15. Točke v Voronijevem diagramu se imenujejo Voronijeve točke in linije se imenujejo Voronijevi robovi.



Slika 4.15. Voronijev diagram

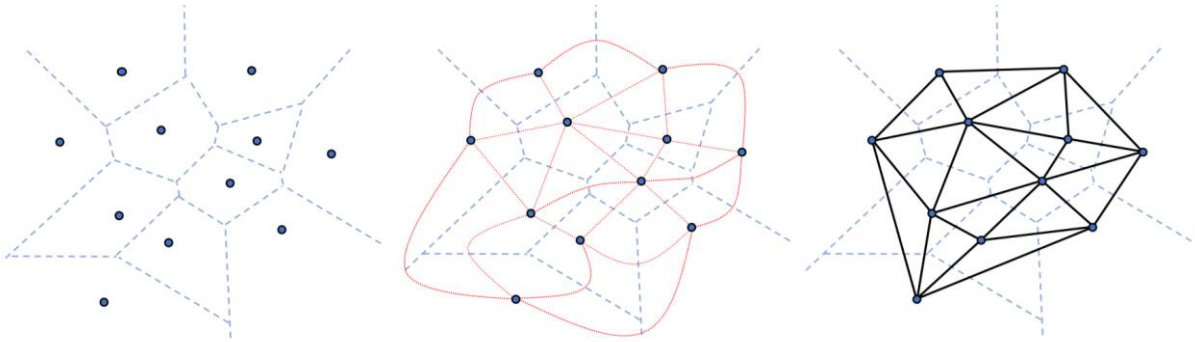
Voronijeve diagrame lahko prikažemo za površinske ali prostorske oblake točk, kot je razvidno s slike 4.16. Teoretično je mogoče izdelati Voronijeve diagrame za oblak točk v n-dimenzionalnem prostoru.



Slika 4.16. Voronijev diagram za površinske in prostorske oblake točk

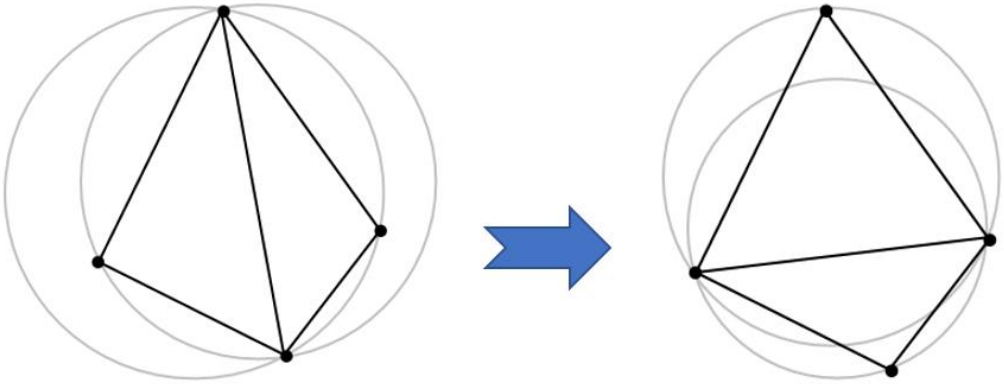
Če namesto točk v oblaku točk narišemo Voronijev diagram, lahko površino dejansko prikažemo in tako dobimo dimenzije objekta. Poligone lahko opremimo z ustrezno barvo ali teksturo in jo osenčimo v odvisnosti od svetila. Težava je v dejstvu, da poligon v Voronijevem diagramu sestavlja različno število robov, ki je odvisno od števila sosednjih točk. Za potrebe računalniške grafike je veliko lažje, če delamo z enakimi poligoni. Najenostavnejši poligoni so trikotniki in so zato tudi najbolj pogosto uporabljeni. Namesto teselacije, ki oblak točk spremeni v poligone, ki nimajo določenega števila robov, raje uporabimo triangulacijo, ki oblak točk spremeni v poligonsko mrežo, sestavljeno iz trikotnikov.

Najpogosteje za triangulacijo oblaka točk uporabimo Delonijevo triangulacijo [69]. Ruski matematik Boris Delone je bil doktorski študent Georgija Voronija, zato je njegova triangulacija nadaljevanje dela Voronija. Osnovna ideja je v povezavi točk prek Voronijevih robov, tako da dobimo trikotnike, kot je prikazano na sliki 4.17.



Slika 4.17. Triangulacija iz Voronijevega diagrama

Na tak način izdelani trikotniki niso vedno prave oblike, zato je Delone določil nekaj pravil za kreiranje trikotnikov. Osnovno pravilo za trikotnike je očrtan krog, ki mora vsebovati samo tri točke, ki so oglišča trikotnika. Če v očrtanem krogu najdemo več točk ali je vsota nasprotnih kotov večja od 180°, je treba spremeniti trikotnik, bodisi z zamenjavo robov ali z dodajanjem nove točke in kreiranjem dodatnih trikotnikov. Na levi strani slike 4.18 očrtana kroga vsebujeta štiri oglišča, zato je treba trikotnika spremeniti oziroma obrniti en rob tako, da dobimo trikotnika, kjer očrtana kroga vsebujeta samo tri oglišča, tako kot je prikazano na desni strani slike 4.18.



Slika 4.18. Pravilo veljavnega roba

Na spletu je danes veliko razvitih algoritmov za Voronijeve diagrame in Delonijevo triangulacijo, kot je na primer [70], kjer lahko poskusimo kreirati poligone z vnašanjem poljubnih točk. Poleg tega je na spletu mogoče najti programe v različnih programskih jezikih. Poleg pretvorbe oblakov točk v poligonske mreže se triangulacija uporablja tudi za pripravo numeričnih modelov v simulacijah, kjer je treba domeno razdeliti na manjše elemente.

4.3 Modeliranje s poligoni

Mreže poligonov oziroma trikotnikov večina CAD programov še vedno obravnava kot oblake točk. Kljub temu je na voljo cela vrsta modelirnikov, ki gradi model samo iz poligonov. Nekateri so namenjeni za urejanje mrež poligonov, ki so nastali iz oblakov točk po 3D skeniranju, medtem ko drugi začnejo modeliranje s poligoni brez začetnega oblaka točk.

Podobno, kot so najmanjši delci v fiziki atomi, ki so sestavljeni iz protonov, elektronov in nevtronov, lahko rečemo, da so poligoni najmanjši delci v računalniški grafiki. Analogno so poligoni sestavljeni iz stranic in oglišč. Stranice lahko obravnavamo kot robove in oglišča lahko enačimo s točkami v prostoru. Razlika je samo v tem, da ko obravnavamo rob, je lahko ta samostojen, ko gre za stranico, je ta vedno del poligona. Enako velja za točko in oglišče.

V različnih programih je lahko upravljanje s poligoni rešeno na različne načine. Najpogosteje najdemo zapise poligonov z geometrijskimi in topološkimi podatki. Geometrijski podatki predstavljajo koordinate točk v prostoru in s topološkimi podatki opišemo stranice in površino poligonov. Poleg teh osnovnih podatkov lahko za vsak poligon zapišemo še dodatne podatke, kot so barve oglišč, stranic in površine poligona. Poleg barve lahko predpišemo tudi normalo, teksturo, material, prosojnost itd.

Najenostavnejši zapis poligona je tak, kjer vsak trikotnik opišemo s seznamom koordinat tako, kot je prikazano v preglednici 2.1. V tem primeru je lahko posamezna točka zapisana večkrat oziroma redundantno. Redundanca je sicer v računalništvu nezaželena, saj se nepotrebno poveča količina pomnilnika, ki ga potrebujemo za zapis poligonov. Po drugi strani tak redundantni zapis omogoča hitrejše risanje elementov, saj ni treba iskati koordinat v seznamih koordinat.

Preglednica 2.1. Zapis trikotnikov s seznamom koordinat

	Oglišče 1			*Oglišče 2*			*Oglišče3*		
Trikotnik 1	X	Y	Z	X	Y	Z	X	Y	Z
Trikotnik2	X	Y	Z	X	Y	Z	X	Y	Z
Trikotnik 3	X	Y	Z	X	Y	Z	X	Y	Z

Seznami koordinat točk so lahko koristni tudi v druge namene in ne zgolj za risanje. Zato najpogosteje zapišemo trikotnike tako, da uporabimo seznam oglišč. V tem primeru imamo dva zapisa, in sicer seznam oglišč v preglednici 2.2 in seznam trikotnikov v preglednici 2.3, v katerem se sklicujemo na seznam oglišč. Dodamo lahko še več seznamov, kot je na primer seznam atributov v preglednici 2.4na katerega se lahko sklicujemo tako v seznamu oglišč kot v seznamu trikotnikov. Tako lahko povežemo številne sezname in uredimo podatke o poligonih v modelu.

Preglednica 2.2. Oglišča

Ogl.	*X*	Y	*Z*	*Atr*
1	X	Y	Z	a
2	X	Y	Z	b
3	X	Y	Z	c
4	X	Y	Z	d

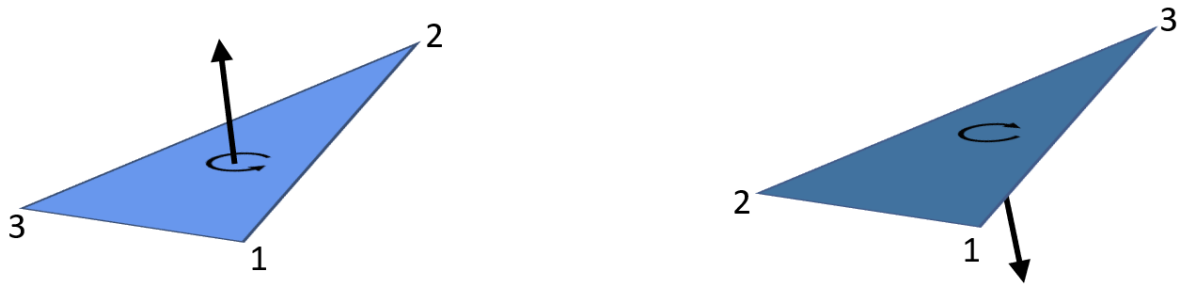
Preglednica 2.3. Trikotnik

Tri.	*X*	Y	*Z*	*Atr*
1	1	2	3	a
2	2	1	4	b
3	3	4	2	c

Preglednica 2.4. Atribut

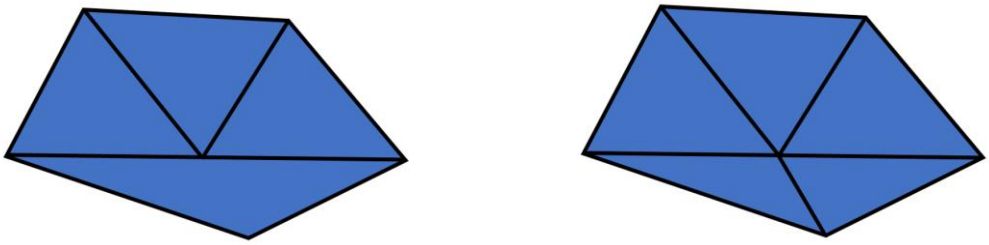
Atr.	*Barva*
a	rdeče
b	zeleno
c	modro
d	črno

Zaporedje oglišč v preglednici 2.3 določa orientacijo trikotnikov. Običajno lahko določimo normalo trikotnika z vektorskim produktom dveh povezanih stranic (na primer prve in zadnje stranice). Če zamenjamo vrstni red stranic, dobimo drugačen vektorski produkt, kot je prikazano na sliki 4.19.



Slika 4.19. Orientacija trikotnikov

Smer normale je pomembna predvsem zato, ker pri prostorskih objektih pričakujemo, da normala kaže iz objekta. Pri površinskih objektih so poligoni lahko različno osenčeni v odvisnosti od smeri normale. Zato je seveda smiselno, da so vsi poligoni, ki sestavljajo isto površino, orientirani v isti smeri. Poleg tega je treba upoštevati tudi določena pravila. Eno od takih pravil je, da morajo biti trikotniki vedno povezani z oglišči in ne smemo povezati dveh trikotnikov, kjer leži oglišče enega trikotnika na stranici drugega trikotnika. Mreža na levi strani slike 4.20 je zato nepravilna in je treba dodati trikotnike tako, kot je prikazano na desni strani slike 4.20.



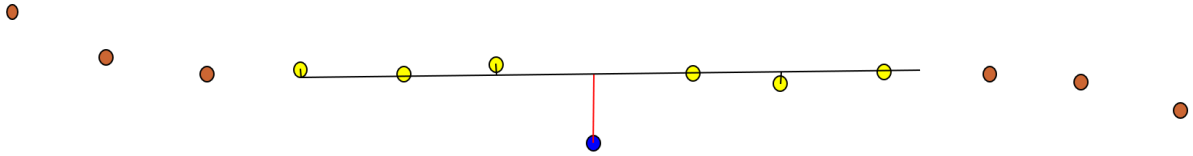
Slika 4.20. Pravilnost mreže trikotnikov

Poleg tega je treba upoštevati, da so mreže poligonov povezane in ne vsebujejo lukenj tam, kjer jih na dejanskem modelu ni. Pri 3D skeniranju se pogosto zgodi, da po triangulaciji dobimo mreže z luknjami in napakami, ki jih je treba popraviti. Pri skeniranju objektov, kjer uporabimo nalepke, del površine pod nalepko ne vsebuje oblaka točk in tam je na površini modela luknja, ki jo je treba zapolniti. Takšne popravke naredimo s programi za urejanje poligonskih mrež ali s poligonskimi modelirniki.

Poleg tega programi omogočajo decimacijo in glajenje. Decimacija je postopek, s katerim zmanjšamo število poligonov. Izraz decimacija prihaja iz časov rimske vojske, kjer so po izgubljeni bitki za kazen ubili desetino vojakov, da so se preostali v naslednjih bitkah bolj borili. Na podoben in nekoliko manj krut način v poligonski mreži zmanjšamo število poligonov. Tako dobimo novo poligonsko mrežo, ki vsebuje manj trikotnikov in je bolj groba. Decimacija omogoča reduciranje napak skeniranja in tako eliminiramo vpliv hrapavosti na obliko površine. Zmanjšanje števila poligonov je zelo pomembno predvsem v živih aplikacijah, kot so računalniške igre ali za potrebe razširjene realnosti (angl. Augmented Reality) [71]. V takšnih aplikacijah se morajo poligoni hitro izrisovati in veliko poligonov pomeni daljše čase risanja. Na sliki 4.21 je primer decimacije, kjer je na levi strani slike 224.126 poligonov, na desni pa samo 4933 poligonov. Tudi količina pomnilnika za shranjevanje poligonov je na desni stokrat manjša kot na levi. Zato je izris hitrejši ob sorazmerno dobri natančnosti modela. Decimacija običajno pomeni, da izbrišemo določeno število poligonov in ustrezno spremenimo topologijo preostalih poligonov. Včasih decimacijo izvedemo tudi tako, da izbrišemo ali združimo točke in rekonstruiramo mrežo poligonov. Vedno je rezultat nova mreža, ki vsebuje manj poligonov.

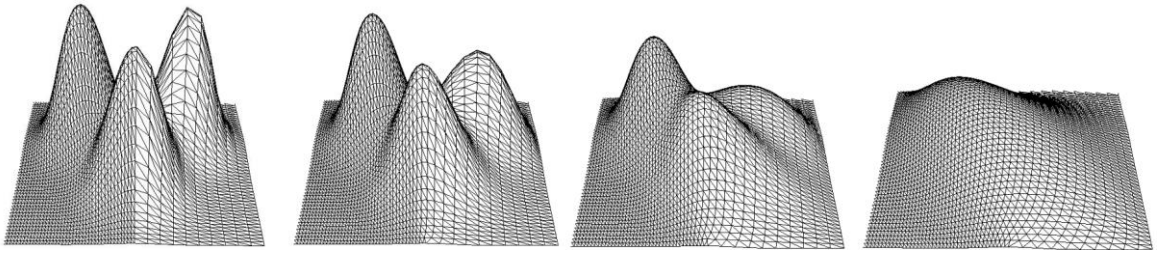
Slika 4.21. Decimacija poligonov [71]

Decimirana mreža poligonov na sliki 4.21 je optimirana tako, da model ohrani vse osnovne geometrijske značilnosti. Včasih lahko z decimacijo pretiravamo ali že osnovni model vsebuje premalo poligonov in model ima nato precej oglate robove. V takšnih primerih je treba uporabiti obraten postopek, ki bo pogosto rezultiral v večjem številu poligonov, ki bodo model zgladili. Glajenje si najlažje predstavljamo, če gre za točke v ravnini, kot so prikazane na sliki 4.22. V tem primeru lahko zgladimo krivuljo tako, da ustrezno premikamo točke. Za glajenje obstajajo številne metode, od katerih je najbolj pogosto uporabljena metoda drseča sredina (angl. moving average) [72].



Slika 4.22. Glajenje točk

Ko gre za glajenje poligonskih površin, ne zadostuje samo premikanje točk, ampak je treba tudi spreminjati mrežo poligonov. Na sliki 4.23 je prikazano glajenje poligonov, kjer je originalen model na levi strani sestavljen iz različne gostote poligonov. Z glajenjem modela se premikajo točke in kreirajo novi poligoni. Gladimo običajno postopoma v več korakih, zato je treba določiti, kako daleč želimo gladiti model, saj lahko v nekaj korakih z glajenjem točk na sliki 4.22 dobimo premico in na sliki 4.23 ravnino. Praviloma se uporabnik odloči, kakšno stopnjo glajenja bo uporabil za določen model. Več poligonov po glajenju zahteva tudi več pomnilnika in počasnejši izris poligonov. Za glajenje poligonov lahko uporabljamo različne metode in nekatere bodo v nadaljevanju tudi bolj podrobno predstavljene.



Slika 4.23. Glajenje poligonov

4.4 Računalniški zapisi poligonskih mrež

Model, ki je predstavljen v obliki poligonske mreže, je lahko zapisan v različnih oblikah. Zapis je navadno odvisen od računalniškega programa, s katerim je določena oblika zapisa. Nekatere oblike takšnih zapisov so postale tako popularne, da služijo za prenos modelov med različnimi programi. Zato večina programov vsebuje vmesnik, ki omogoča uvoz takšnih modelov. Tako praktično vsi CAD modelirniki omogočajo uvoz STL datotek, čeprav tak model potem obravnavajo kot za oblak točk.

STL

STL [73] je format, ki so ga razvili za proces 3D tiskanja, ki se imenuje stereolitografija v podjetju 3D Systems. Format zapisa je relativno preprost, saj je vsak poligon v datoteki zapisan z normalo poligona in s seznamom koordinat točk. Zapis omogoča modele s poljubnimi mnogokotniki, vendar je običajno najlažje določiti model z mrežo trikotnikov, zato je večina STL modelov predstavljenih z mrežo trikotnikov. Oblika zapisa je lahko v berljivem ASCII zapisu ali v binarni obliki z jasno definirano obliko zapisa. Zato je mogoče izdelati vmesnik, ki omogoča branje poljubne STL datoteke in hkrati je mogoče poljubno poligonsko mrežo zapisati v STL obliki. Čeprav je format datoteke nastal za namene 3D tiskanja, to ne pomeni, da lahko to datoteko kar pošljemo na tiskalnik. Format datoteke omogoča preprosto pretvorbo v navodila za specifičen tiskalnik za postopek tiskanja. Format v obliki STL je tako uporaben tako za stereolitografijo kot za različne druge tehnike 3D tiskanja, kot je tiskanje z ekstruzijo materiala, sintranjem, laminiranjem itd. V vsakem primeru je treba uporabiti dodatno programsko opremo, ki ji pogosto rečemo rezalnik (angl. slicer), ki pretvori STL model v obliko G-kode, kjer so zapisani ukazi za tiskalnik podobno, kot je to običajno za numerično krmiljene obdelovalne stroje. STL format je uporaben tudi za prenos 3D modelov med posameznimi modelirnimi programi in za računalniško predstavitev modela, saj lahko STL vedno prikažemo na zaslonu v poljubni projekciji. Pretvorba v CAD model ali v model za računalniške simulacije je malo bolj zahtevna, ker programi pogosto zahtevajo podatke o natančni geometriji modela. To pomeni, da želimo imeti podatek o koordinati poljubne točke na površini modela. Tega v STL ni, saj vsebuje samo podatke o koordinatah oglišč in vektorju normale. Mogoče je določiti točke znotraj enega poligona, vendar CAD programi želijo imeti matematičen zapis zveznih površin, zato pogosto iz oblaka točk, ki ga določajo oglišča trikotnikov, programi rekonstruirajo takšne površine. Nekateri programi takšno rekonstrukcijo omogočajo, drugi omogočajo samo prikaz STL modela, na podlagi katerega je treba potem še enkrat zmodelirati CAD model.Na sliki 4.24 je prikazan ASCII zapis STL datoteke, ki se navadno začne z ukazom solid, ki mu lahko damo poljubno ime. Nato ta model vsebuje poligone, opisane z ukazom facet in podatkom o normali. Namesto nx, ny in nz je treba vstaviti koordinate, ki določajo smer normale. Enako velja za koordinate oglišč, zapisane v seznamu za zanko (angl. loop) in za posamezno oglišče (angl. vertex).

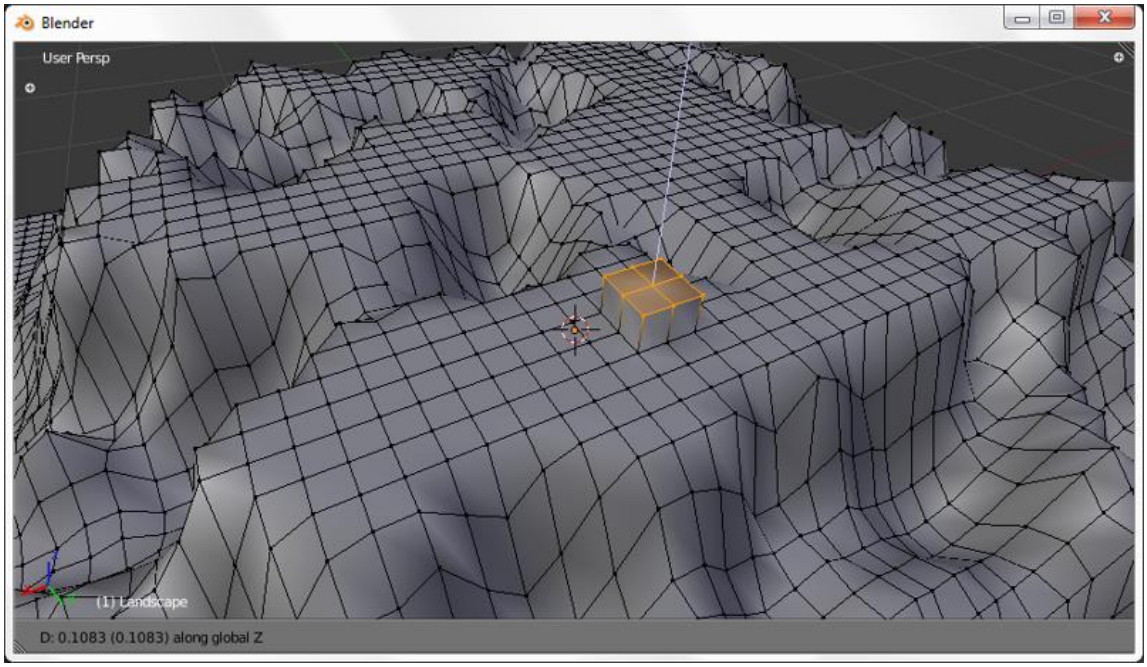
```
solid ime
    facet normal n_x n_y n_z
        outer loop
            vertex x_1 y_1 z_1
            vertex x_2 y_2 z_2
            vertex x_3 y_3 z_3
        endloop
    endfacet
endsolid ime
```

Slika 4.24. STL zapis

Na spletu je mogoče najti veliko modelov v obliki STL, posebej zaradi velikega razmaha 3D tiskanja. Nekateri modeli so brezplačni, veliko STL modelov pa je možno kupiti ali prodati na spletu.

Blender

Program Blender [74] je nastal kot oblikovalsko orodje za nizozemski animacijski studio. Podjetje NaN, ki je razvijalo program, je bankrotiralo leta 2002. Takrat je program postal odprtokoden in izvorna koda je postala javna, razvoj programa pa se v takem obsegu nadaljuje še danes. Zmogljivost in brezplačna uporaba sta pripomogli k temu, da ima program Blender veliko uporabnikov. Modeliranje v programu poteka vedno na ravni poligonov, pri čemer je mogoče uporabljati različne načine kreiranja in urejanja poligonov, kot je prikazano na sliki 4.25.

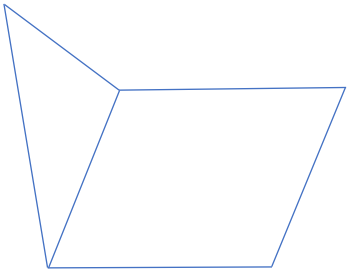


Slika 4.25. Urejanje mreže poligonov v programu Blender

Program vsebuje zahtevna orodja za kreiranje modelov in številne fizikalne modele za izdelavo simulacij. Posebej kakovostno je v programu izdelano renderiranje s številnimi vrstami osvetlitev in materialov. Renderiranje videoposnetkov je lahko časovno zelo zahtevno in presega zmogljivosti osebnih računalnikov. Zato je mogoče renderiranje s programom Blender opraviti tudi v računalniških gručah, ki vsebujejo zmogljive grafične kartice. Tako je mogoče izdelati renderiranje videoposnetkov v sprejemljivem času. Uporaba programa ni preprosta in zahteva kar nekaj časa za usposabljanje. Kljub temu je trenutno več kot milijon aktivnih uporabnikov programa. Posledično je na voljo veliko različnih modelov, ki jih ti uporabniki izdelajo. Datoteka z modelom programa blender ima končnico .blend. Čeprav so odprtokodni programi zelo popularni med uporabniki, ki nimajo denarja za nakup komercialne programske opreme, lahko program zasledimo tudi v organizacijah, kot je NASA. Veliko uporabe je mogoče zaslediti tudi v filmski in televizijski produkciji.

Autocad DXF

Autocad je program za izdelavo tehniške dokumentacije in nima veliko povezav s poligonskimi mrežami. Vendar so v programu zaradi potrebe po izmenjavi podatkov z drugimi programi razvili format DXF (angl. Drawing Exchange Format) [75]. Format je pretežno namenjen prenosu risb, narisanih s črtami, vendar je mogoče zapisati tudi poligone v prostoru. Zaradi enostavnosti in odprtosti zapisa je postal format zelo popularen. Čeprav format zapisa ni formalni standard, je zaradi široke uporabe postal dejanski standard za prenos 2D/3D modelov. Na sliki 4.26 je prikazan preprost 3D model z ustreznim zapisom v obliki DXF.



99	9	2	0	10
VISION3D DXF	$LINMAX	LAYER	3DFACE	-0.5
0	10	70	8	20
SECTION	1000.0	6	1	-0.5
2	20	0	62	30
HEADER	1000.0	LAYER	1	-0.5
9	0	2	10	11
$ACADVER	ENDSEC	1	-0.5	0.5
1	0	70	20	21
AC1006	SECTION	64	-0.5	-0.5
9	2	62	30	31
$INSBASE	TABLES	7	-0.5	-0.5
10	0	6	11	12
0.0	TABLE	CONTINUOUS	-0.5	0
20	2	0	21	22
0.0	LTYPE	ENDTAB	0.5	-0.5
30	70	0	31	32
0.0	1	TABLE	-0.5	0.5
9	0	2	12	13
$EXTMIN	LTYPE	STYLE	0.5	0
10	2	70	22	23
0.0	CONTINUOUS	0	0.5	-0.5
20	70	0	32	33
0.0	64	ENDTAB	-0.5	0.5
9	3	0	13	0
$EXTMAX	Solid line	ENDSEC	0.5	ENDSEC
10	72	0	23	0
1000.0	65	SECTION	-0.5	EOF
20	73	2	33	
1000.0	0	BLOCKS	-0.5	
9	40	0	0	
$LINMIN	0.000000	ENDSEC	3DFACE	
10	0	0	8	
0.0	ENDTAB	SECTION	1	
20	0	2	62	
0.0	TABLE	ENTITIES	1	

Slika 4.26. Zapis modela v DXF datoteki

Datoteka DXF je sestavljena iz ASCII zapisa, v katerem je v vsaki vrstici en ukaz ali podatek. Zapis na sliki 4.26 je prikazan v več stolpcih le zaradi prihranka prostora, v datoteki je vse skupaj v enem stolpcu in ukazi si navpično sledijo od 99 do EOF. Ukazi so dokaj razumljivi in ustrezno dokumentirani, zato je takšno obliko datoteke mogoče zlahka izdelati ali prebrati v številnih programih. Hkrati je tudi na spletu veliko risb in modelov zapisanih v obliki DXF, kar je rezultat dela velikega števila uporabnikov, ki so takšne datoteke izdelali.

Autodesk 3ds Max

Program 3ds Max [76], ki se je včasih imenoval tudi 3D Studio Max, je komercialni program, ki ima vse in več funkcionalnosti programa Blender. Verjetno je bolj smiselno trditi, da Blender poskuša posneti vse, kar ima 3ds Max. Modeliranje s poligoni je osnovna značilnost tega programa, ki je namenjen izdelavi virtualnih izdelkov, ki so potem uporabljeni v računalniških igrah ali filmih. V programu je mogoče izdelati vse od osnovnih modelov do končne renderirane animacije. Zato je to pogosto osnovno orodje za oblikovalce v industriji računalniških iger, filmov in televizijske produkcije. Modeli so zapisani v datotekah s končnico 3ds, ki jih lahko preberejo poleg 3ds Max programa tudi drugi programi. Podjetje Autodesk ima za izdelavo animacij na voljo še en program, in sicer program Maya [77], ki deluje tudi na računalnikih z operacijskimi sistemi Unix. Program Maya prav tako temelji na poligonskih mrežah in je bolj uporaben za zahtevne 3D animacije, vendar je tudi bolj zahteven za uporabo.

DAE – Collada

Datoteke s končnico .dae, imenovane COLLADA (angl. COLLAborative Design Activity) [78], so razvili v konzorciju Khronos pod pokroviteljstvom podjetja Sony. Format je postal ISO standard, zato je uporabljen v številnih programih (3ds Max, Blender, Cinema 4D, Maya, Modo, Solidworks, Sketchup itd.). Posebej pomembna je podpora v Sketchup in Google Earth, saj je tako mogoče modelirati zgradbe, ki so potem vidne na zemljevidih. Format je uporabljen tudi v igralnih pogonih (Unity, CryEngine itd.) in v različnih virtualnih svetovih, na primer Second Life. Zaradi vsega tega je format precej razširjen in pogosto uporabljen za predstavitev modelov, sestavljenih iz poligonskih mrež.

Wavefront OBJ

Format Wavefront obj [79] je nastal v podjetju Wavefront Techonlogies za potrebe programa Advanced Visualizer. S tem programom so izdelali računalniško grafiko za številne filme konec 20. stoletja, zato je postal ta format precej popularen. Program Advanced Visualizer je bil osnova za program Maya, ki je zdaj v lasti podjetja Autodesk. Zaradi velike razširjenosti in popularnosti formata se ta še danes pogosto uporablja za shranjevanje poligonskih mrež. Nekoliko nerodna je izbira ekstenzije .obj, saj lahko datoteke s takšno ekstenzijo vsebujejo tudi kakšne druge podatke o objektu. Pogosto so to tudi vmesne datoteke pri prevajanju računalniških programov.

PLY

Format PLY [80] so razvili na Univerzi v Stanfordu, kjer so za osnovo vzeli Wavefront OBJ datoteko in naredili svoj bolj odprt zapis za poligonske mreže. Zapis je lahko v binarni ali ASCII obliki in na voljo so tudi programske knjižnice, ki omogočajo preprosto branje in pisanje datotek. Zato je zlahka mogoče v poljuben program dodati podporo za ta zapis datotek. Posledično zapis podpirajo številni programi, v tem zapisu pa lahko najdemo tudi različne modele v izobilju.

Cinema 4D

Program Cinema 4D [81] je nastal iz programa za renderiranje za računalnik Amiga in je pozneje postal zmogljiv program za modeliranje, renderiranje in animacije poligonskih mrež. Uporabljen je v številnih filmih. Program dobro izkorišča delovanje strojne opreme in se pogosto navaja kot testni program za določanje zmogljivosti procesorjev in grafične opreme. Poleg uporabe v filmski industriji se pogosto uporablja v arhitekturi za predstavitev in sprehode skozi bodoče projekte. Čeprav gre za komercialen program, je na voljo veliko modelov v obliki datotek s končnico .c4d, ki jih je mogoče prebrati tudi v nekaterih drugih programih (na primer Adobe After Effects, Unity).

VRML

Format VRML (angl. Virtual Reality Modeling Language) [82] je nastal kot varianta opisnega jezika, kot je HTML (angl. Hyper Text Markup Language), s tem da namesto spletne strani s tem jezikom opišemo 3D model. Pri tem gre za model, sestavljen iz poligonske mreže, vendar so dodani tudi nekateri dodatni elementi, s katerimi lahko vplivamo na delovanje modela. Model v tej obliki je mogoče prikazati na spletni strani, vendar mora biti v brskalniku instaliran poseben vtičnik za prikazovanje VRML datotek. Uporaba je potem preprosta in uporabnik se lahko sprehaja po modelu in z različnimi stikali spreminja stanje modela. VRML je postal ISO standard in je bil zato precej uporabljan za kreiranje modelov. Vmesnik za zapis VRML modela je bilo dovolj preprosto vključiti v poljuben program.

X3D

Spletni brskalniki so precej napredovali od nastanka VRML jezika, zato je danes VRML že precej zastarel. Poskušali so ga nadomestiti z X3D jezikom, ki ni najbolj zaživel, saj je danes mogoče 3D modele prikazati v HTML obliki brez posebnega vtičnika.

3DXML

Zapis 3DXML [83] je precej podoben VRML oziroma X3D standardu, vendar so 3DXML obliko razvili v podjetju Dassault za potrebe prenosov 3D modelov, ki jih je bilo mogoče prikazati v prosto dostopnem programu 3DVIA Player. Podobno kot VRML je tudi ta format zastarel zaradi HTML 5 jezika.

HTML5

Z definicijo HTML5 jezika [84] je bilo mogoče v spletno stran vključiti različne aktivne elemente. Zato je odpadla potreba po različnih vtičnikih, ki so omogočali prikazovanje 3D modelov. 3D grafiko je mogoče programsko vstaviti v spletno stran. Zato danes ni več dovolj samo prikazati 3D modela na spletni strani, ampak imamo kar 3D modelirnike na spletnih straneh. Takšne prosto dostopne modelirnike je mogoče najti kar na spletu [85] in tudi velike programske hiše imajo na voljo spletne verzije svojih programov (na primer Dassault 3Dexperience, PTC Onshape). Zaradi tega so posebne datoteke, ki so omogočale prikazovanje 3D modelov na spletnih straneh s posebnimi vtičniki, postale odveč oziroma zastarele.

3MF

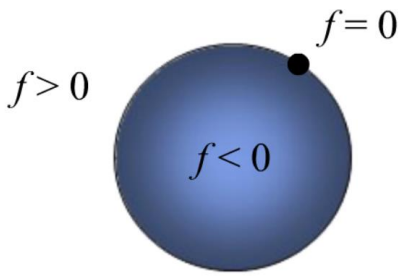
V zadnjih letih se je razvoj 3D modelov in 3D tiskanja zelo razmahnil, zato se je pokazala potreba po novem formatu za zapis poligonskih mrež. V ta namen je bil ustanovljen konzorcij, v katerem so številna podjetja, ki razvijajo 3D modelirnike (Autodesk, Dassault, Siemens, PTC itd.). Konzorcije je izdelal format za zapis 3D izdelave (angl. 3D Manufacturing Format) [86]. Vsa podjetja v konzorciju omogočajo izvoz tega formata iz svojih 3D modelirnikov. Poleg tega v konzorciju sodeluje tudi Microsoft, kot proizvajalec najbolj razširjenega operacijskega sistema in pisarniških programov. Datoteka s končnico .3mf je zapisana v obliki xml in vsebuje poleg zapisa 3D modela v obliki poligonov tudi teksture objektov. Tako je mogoče prenašati celoten 3D model v renderirani obliki in ga prikazati v različnih programih. Poleg namenskih programov za 3D modeliranje je prikazovanje omogočeno tudi v pisarniških programih, kot so na primer Microsoft Word, Excel, Powerpoint in Outlook. Glede na ime formata je glavni namen seveda 3D tiskanje, ki je lahko izvedeno z različnimi materiali v odvisnosti od izbranega materiala v 3D modelu.

5 Implicitne površine

Poligonske površine so zelo primerne za prikazovanje modelov na zaslonu. Tudi pri 3D tiskanju izdelka takšne mreže zadostujejo, saj je možno gladkost površin zagotoviti tudi z naknadno obdelavo površine. Če želimo natančno gladko geometrijo površine, so diskretni ploskovni poligoni moteči. Idealno želimo površino, kjer lahko med vsakima dvema točkama površine najdemo še eno točko. Z drugimi besedami to pomeni, da želimo zvezne površine, zapisane z matematičnimi funkcijami, ki omogočajo uporabo različnih numeričnih metod za urejanje modela v prostoru. Matematične funkcije so običajno zapisane implicitno, kar pomeni, da imamo na levi strani enačbe matematični izraz oziroma funkcijo z različnimi spremenljivkami, ki je enaka nič, kot je prikazano v enačbi 5.1.

$$f(x,y,z) = 0 \tag{5.1}$$

Implicitna površina je definirana s funkcijo, kjer je vrednost funkcije enaka 0 samo na površini, povsod drugje je vrednost funkcije različna od 0. Vrednost funkcije je lahko na eni strani površine večja od nič in na drugi strani manjša od nič. Pri tem sta pozitivna in negativna vrednost odvisni od same funkcije. Za sferično površino velja pravilo, kot je prikazano na sliki 5.1, vendar to ne velja za poljubno površino. Zato nas za implicitne površine zanimajo samo vrednosti tistih spremenljivk, ki določajo točko na površini in določajo vrednost funkcije enako 0.



Slika 5.1. Vrednost implicitne funkcije za sfero

Za potrebe računalniške grafike je treba določiti še tangente in normale v vsaki točki površine. Tangente so določene z odvodi funkcije v posamezni koordinatni smeri, normala pa je določena z gradientom funkcije tako, kot je prikazano v enačbi 5.2. Ker je površina zvezna, je mogoče določiti tangente in normale v vsaki točki površine.

$$\nabla f(x,y,z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right) \tag{5.2}$$

Najenostavnejša implicitna površina je ravnina, ki je redko uporabljena direktno v 3D modeliranju. Namesto tega se ravnine uporabljajo kot podporne površine, na katerih gradimo bolj kompleksno geometrijo. Ravnino lahko zapišemo z linearno enačbo 3.17 ter tangente in normale določimo z odvodi te enačbe.

Bolj privlačne so površine, ki jih lahko zapišemo z višjimi stopnjami polinoma, kot je sferična površina, določena z enačbo 5.3, kjer r označuje polmer sfere in elipsoid z enačbo 5.4, kjer r_x, r_y in r_z določajo polosi elipsoida.

$$x^2 + y^2 + z^2 - r^2 = 0 \tag{5.3}$$

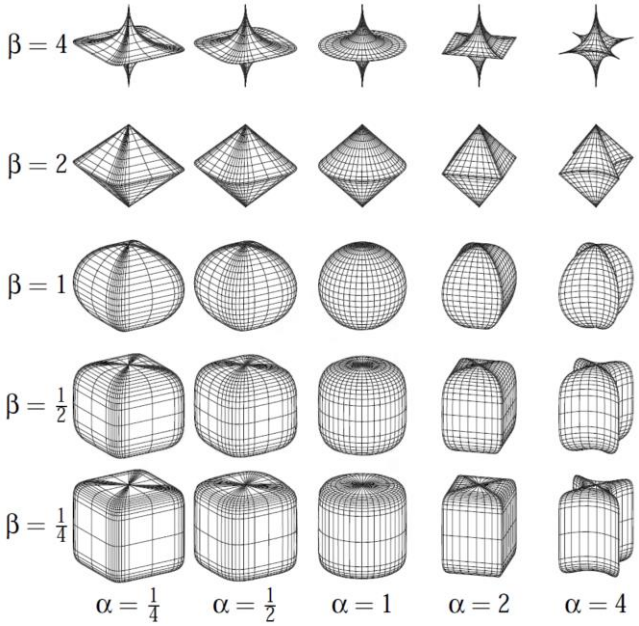
$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 - 1 = 0 \tag{5.4}$$

Implicitno enačbo lahko zapišemo tudi za bolj kompleksne površine, kot je na primer površina torusa v enačbi 5.5, kjer je c premer torusa in a premer cevi.

$$\left(\frac{c}{a}-\sqrt{\left(\frac{x}{a}\right)^2+\left(\frac{y}{a}\right)^2}\right)^2+\left(\frac{z}{a}\right)^2-1=0 \tag{5.5}$$

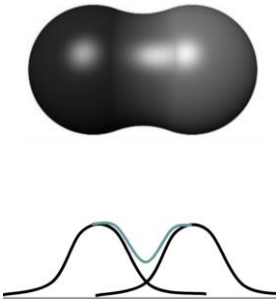
S spreminjanjem prametrov, polmerov pri sferi, polosi pri elipsoidu ali premerov pri torusu, lahko dobimo različne velikosti površine, vendar se oblika ne spremeni. Seveda je koristno, če lahko element spremenimo tudi oblikovno. Primer take implicitne površine je super elipsoid [87]. Super elipsoide lahko zapišemo z enačbo 5.6 [88] in prikažemo na sliki 5.2.

$$\left(x^{2/\alpha}+y^{2/\alpha}\right)^{\alpha/\beta}+z^{2/\alpha\beta}-1=0 \tag{5.6}$$



Slika 5.2. Oblike super elipsoidov v odvisnosti od parametrov [88]

Za potrebe predstavitve molekularnih struktur je Blinn leta 1982 predlagal implicitne strukture v obliki madežev (angl. Blobs) [89]. Madeži so lahko manj ali bolj razmazani in tako definirajo različne oblike. Matematično so madeži sestavljeni iz kombinacije Gaussovih krivulj, kot je prikazano na sliki 5.3.



Slika 5.3. Površina iz kombinacije Gaussovih krivulj

Blinn je implicitne površine zasnoval na podlagi gostote atomov vodika in posledično je enačba 5.7 relativno kompleksna. V enačbi 5.7 je radij določen z enačbo 5.8 in predstavlja razdaljo od središča posameznega atoma. Parametra a in b predstavljata standardno deviacijo in višino Gaussove krivulje. Parameter T je definiran v enačbi 5.9, B v enačbi 5.10 pa določa razmaz posameznega madeža.

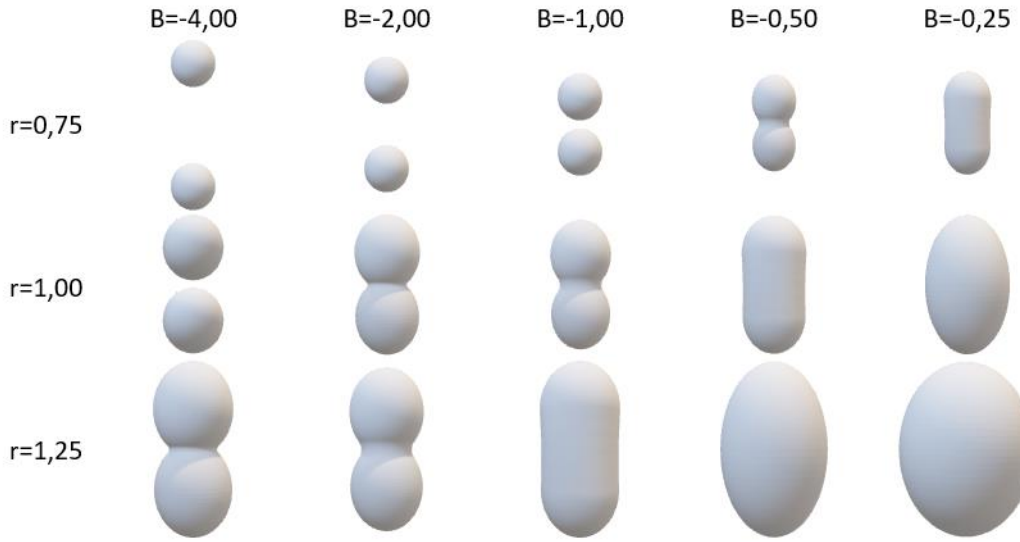
$$\sum_i b_i e^{-a_i r_i^2} - T = 0 \tag{5.7}$$

$$r = \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z)^2} \tag{5.8}$$

$$T = b_i e^{-a_i R_i^2} \tag{5.9}$$

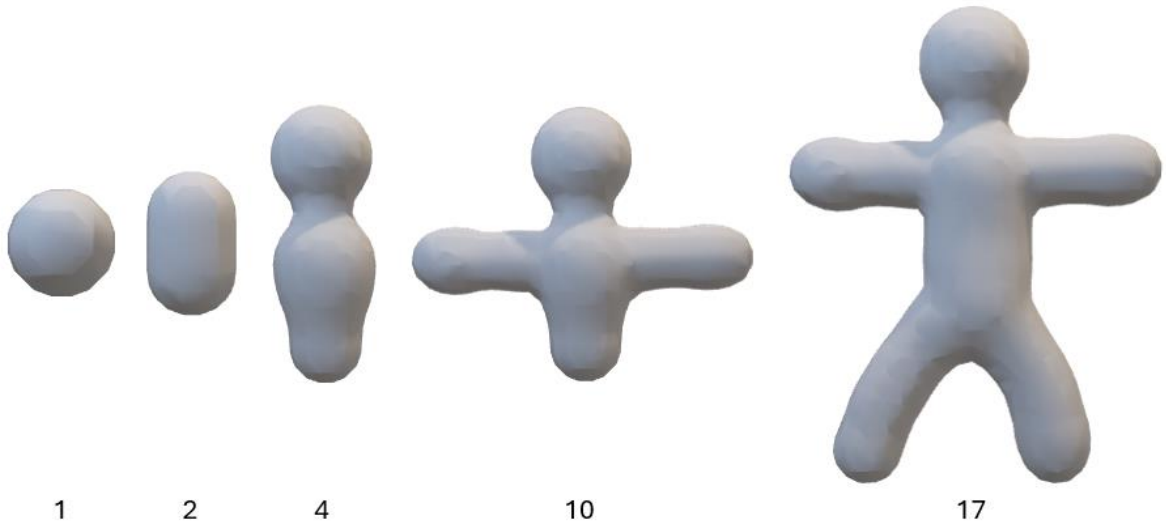
$$B_i = ln\left(\frac{T}{b_i}\right) \tag{5.10}$$

Na sliki 5.4 so prikazane različne oblike madežev v odvisnosti od vrednosti B in r.



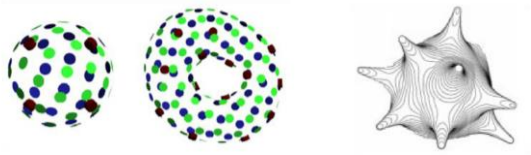
Slika 5.4. Vpliv parametrov na razmaz madežev [89]

Kakorkoli prilagajamo prametre super elipsoidov ali madežev, je nemogoče dobiti poljubno obliko površine. V ta namen kombiniramo implicitne površine z uporabo Boolove algebre podobno kot v CSG modelirniku. S prilagajanjem in sestavljanjem implicitnih površin lahko izdelamo poljubne izdelke [90]. Na sliki 5.5 je primer modeliranja skulpture v programu Blender z uporabo implicitnih površin imenovanih Metaballs. Številke pod modeli označujejo število uporabljenih implicitnih površin.



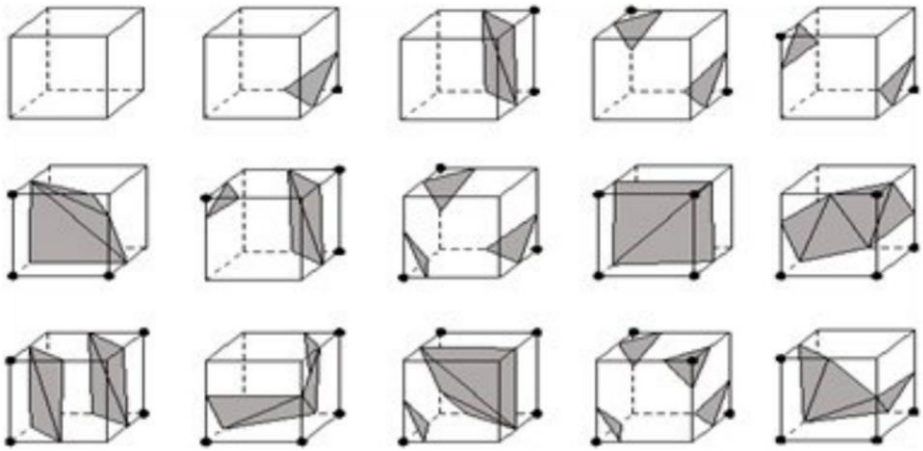
Slika 5.5. Modeliranje z implicitnimi površinami v programu Blender

Implicitne površine so predstavljene z implicitnimi enačbami in v primeru takšne kombinacije površin imamo podobne težave kot pri predstavitvi CSG modelov. Izdelek je dokaj preprosto predstaviti z metodo sledenja žarku, saj je treba le računati presečišča z implicitnimi površinami in upoštevati Boolovo algebro. Težava je v tem, da so bili računalniki še konec 20. stoletja zelo počasni in je takšno renderiranje lahko trajalo zelo dolgo. Drug način predstavitve implicitnih površin je risanje po površini, kjer najdemo določeno točko na površini in okoli nje pobarvamo nekaj površine. Tako dobimo na površini vzorce. Namesto tega lahko začnemo risati linije po površini in tako delno označimo površino. Primer takšnih predstavitev je na sliki 5.6.



Slika 5.6. Vzorci in krivulje na implicitni površini

Idealno je seveda, če lahko implicitne površine pretvorimo v poligone. V ta namen je uporaben algoritem »kock, ki korakajo« (angl. Marching Cubes) [91]. Algoritem temelji na volumetričnem modeliranju, kjer celoten prostor razdelimo na voksle v obliki kock. V vsakem vokslu nato primerjamo dejansko površino z vnaprej pripravljenimi vzorci poligonov in izberemo najbolj podoben vzorec. Na sliki 5.7 je prikazanih nekaj vzorcev, s katerimi primerjamo voksel prostora, kjer je želena površina.



Slika 5.7. Nekaj referenčnih vzorcev (15 od 256)

Marching Cubes je znan po svoji natančnosti in hitrosti izrisovanja 3D modelov. Algoritem je zmožen izrisovati zelo zahtevne modele s površinami z veliko krivuljami in neravninami, brez potrebe po velikih količinah računalniškega časa ali spomina. Tako je idealno orodje za aplikacije, kot so prikazovanje medicinskih slik, računalniška tomografija in računalniška rekonstrukcija.

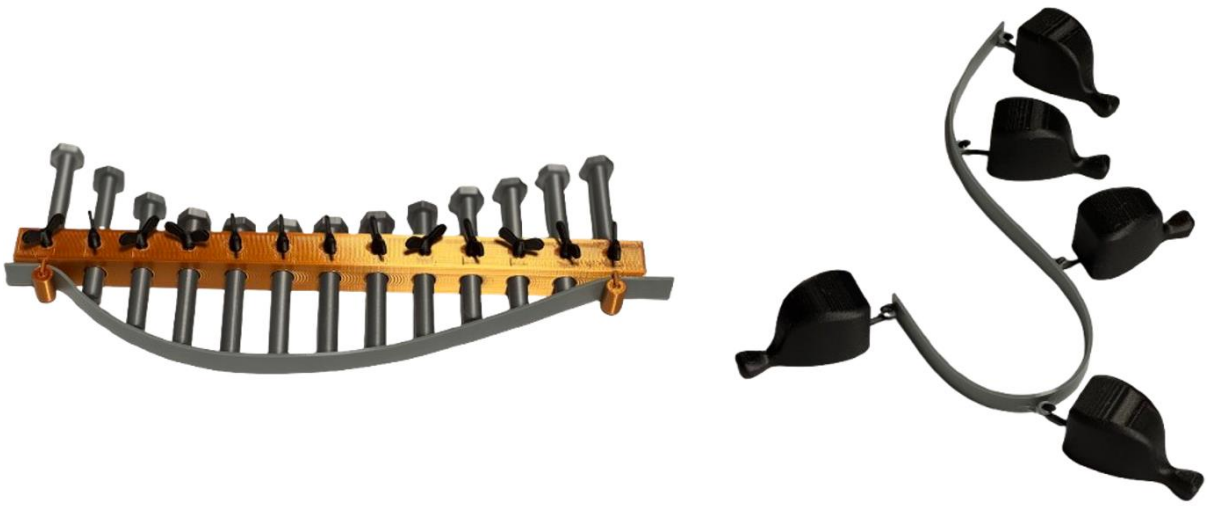
Kreiranje 3D modelov z implicitnimi modelirniki, ki so jih proučevali konec 20. stoletja, je precej zamrlo, ker ni bilo primerne metode za renderiranje površin. Tako je danes precej težko zaslediti modelirnik, ki deluje z implicitnimi površinami. Mogoče je najbolj uporabljan CSG modelirnik, ki se uporablja v računalniški igri Roblox [92], kjer se modelira z uporabo implicitnih teles, ki jih kombiniramo z Boolovo algebro. Dejanski modelirniki, ki bi delovali z Blinnovimi madeži, so bili razviti v raziskovalne namene in jih je praktično nemogoče uporabljati. V prihodnosti, ko bo strojna oprema za renderiranje s sledenjem žarkom v realnem času standardna oprema računalnikov, je mogoče pričakovati, da bodo takšni modelirniki znova zelo privlačni za uporabo.

6 Parametrične krivulje in površine

Implicitne površine so opisane s pomočjo matematičnih enačb, ki določajo, katere točke v prostoru spadajo na površino. Da bi izrisali implicitno površino, moramo najprej poiskati vse točke, ki so na površini, kar je lahko zahtevno in časovno potratno. Zato pogosto raje uporabljamo parametrične površine, ker jih je v računalništvu lažje uporabljati kot implicitne površine. Parametrične površine so opisane s pomočjo matematičnih funkcij, ki določajo njihovo obliko in položaj v prostoru. Ker so opisane s pomočjo funkcij, jih je lažje uporabljati in izrisovati, saj lahko računalnik zelo hitro uporabi formulo za določitev točke na površini. Parametrične površine vsebujejo parametrične krivulje in s spreminjanjem krivulj lahko spreminjamo tudi površine. Ker je matematična predstavitev krivulj bolj preprosta od površin, bodo v nadaljevanju najprej opisane različne parametrične krivulje in nato še parametrične površine.

6.1 Parametrične krivulje

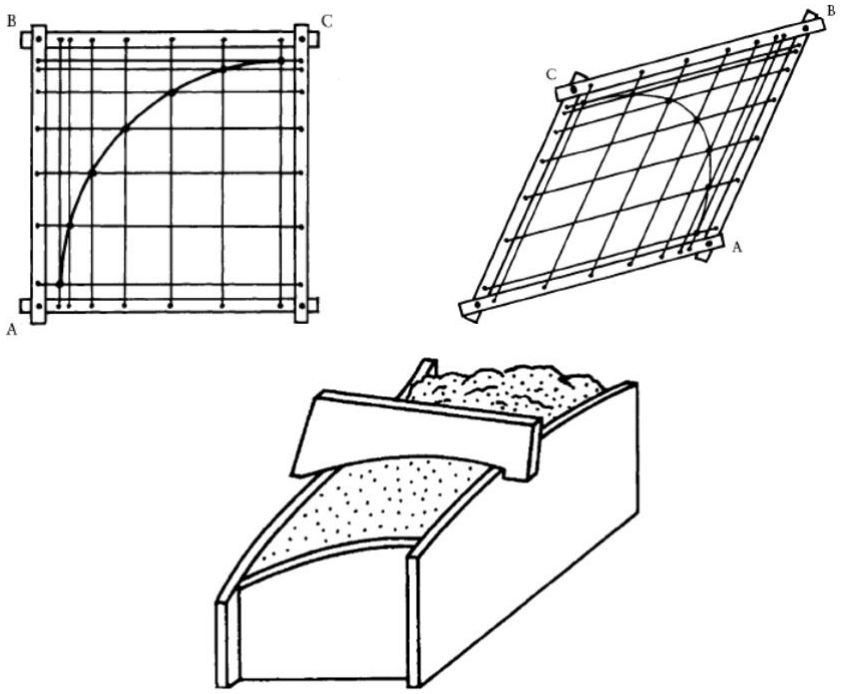
Uporaba krivulj za konstruiranje sega v antiko, ko so ljudje prvič začeli izdelovati dekorativne in funkcionalne predmete. Krivulje so bile pomembne tudi za konstruiranje ladij zaradi njihove sposobnosti, da se prilagodijo valovanju morja in da omogočijo bolj stabilno in varno plovbo. V srednjem veku so krivulje postale še pomembnejše pri konstruiranju ladij, saj so omogočile izdelavo bolj zmogljivih in hitrejših ladij, ki so lahko premagovale daljše razdalje in se soočale z močnejšimi valovi. Krivulje so bile uporabljene tudi za izdelavo stabilnejših in varnejših trupov ladij, ki so bili sposobni prenesti večje obremenitve. Obliko uporabljenih krivulj je bilo treba shraniti in pogosto so se v ta namen uporabljali modeli krivulj. Ti imajo kar nekaj pomanjkljivosti, saj se lahko modeli uničijo ali konkurenti ukradejo modele in tako pridobijo znanje. Zato so začeli uporabljati mehanske krivuljnike, ki so omogočali shranjevanje oblike. Replika takšnega spline krivuljnika, ki so ga uporabljali v 17. stoletja je prikazana na levi strani slike 6.1 in omogoča spreminjanje oblike s privijanjem vijakov, ki deformirajo palico v želeno obliko.



Slika 6.1. Spline naprave

Podobne krivuljnike prikazane na desni strani slike 6.1 najdemo še danes v konstruiranju in vijake so zamenjale uteži v obliki rac (angl. spline with ducks) [93]. S premikanjem uteži spremenimo obliko krivulje. To omogoča risanje poljubnih krivulj, vendar je manj primerno za shranjevanje oblik. Pri krivuljniku z vijaki je dovolj, da si zapomnimo ali zapišemo globino posameznega vijaka in s tem shranimo obliko.

V 20. stoletju so še vedno v industriji uporabljali krivuljnike, kot je krivuljnik na sliki 6.2, ki omogoča spremembo krivulj z deformiranjem okvirja. Tak krivuljnik so uporabljali v podjetju Renault v šestdesetih letih dvajsetega stoletja za izdelavo kalupov za izdelavo orodja za pločevinske dele izdelkov.

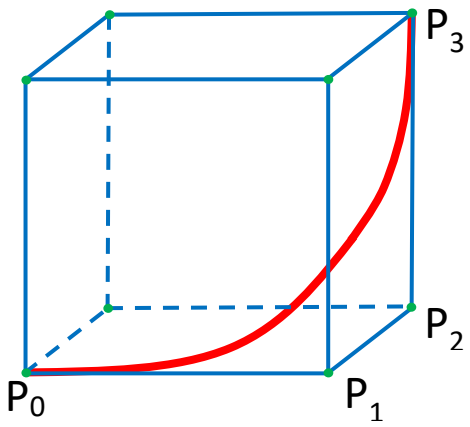


Slika 6.2. Krivuljnik z okvirjem in kalup za orodje v podjetju Renault

Z razvojem računalnikov je postalo zanimivo uporabljati matematične krivulje, ki jih lahko nato uporabimo v numerično krmiljenih strojih za izdelavo orodij. V letalski industriji so uporabljali stožnice (angl. Conics), ki so pravzaprav implicitne projekcijske krivulje. V računalništvu je bolj primerno za zapis krivulj uporabljati parametrične krivulje, kjer spreminjamo vrednost parametra in iz enačbe dobimo koordinate točke na krivulji.

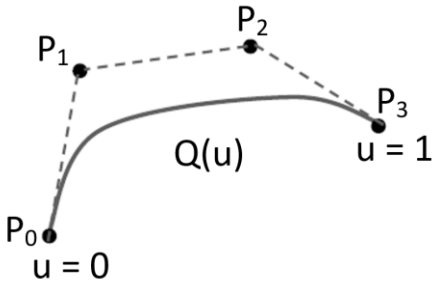
6.1.1 Bezierove krivulje

Bezierove krivulje [94] so prve parametrične krivulje, uporabljene v avtomobilski industriji. Zanimivo je, da jih je najprej uporabil francoski inženir De Casteljau, ki je delal za avtomobilsko podjetje Citroen v 60. letih 20. stoletja. Med svojim delom je razvil nov matematični model za opisovanje krivulj, imenovan algoritem De Casteljau [95], ki je omogočal bolj natančno in učinkovito izrisovanje zahtevnih krivulj v računalniški grafiki. Algoritem De Casteljau je bil zasnovan tako, da je omogočal izrisovanje krivulj z minimalnim številom podatkov, kar je bilo pomembno za izboljšanje učinkovitosti in hitrosti računalniških sistemov v tistem času. Ker so takrat v Citroenu dejansko uporabljali takšne krivulje za izdelavo orodij za avtomobile, je bila to poslovna skrivnost in De Casteljau tega ni nikjer objavil. Zato se te krivulje danes imenujejo Bezierove krivulje. Pierre Bézier je bil francoski inženir, ki je delal za podjetje Renault. V namen računalniškega modeliranja krivulj je uporabil kar krivuljnike, prikazane na sliki 6.2, in jih predelal v matematično obliko parametričnih enačb. Namesto na okvir je krivuljo namestil v dve oglišči paralelepipeda tako, kot je prikazano na sliki 6.3. Z deformacijo paralelepipeda se spreminja krivulja. Pri tem je začetna točka krivulje v oglišču P_0 in končna točka krivulje v oglišču P_3. Stranica med P_0 in P_1 določa tangento krivulje v točki P_0, stranica med P_2 in P_3 pa določa tangento krivulje v točki P_3.



Slika 6.3. Paralelepiped z včrtano krivuljo

Namesto celotnega paralelepipeda lahko narišemo samo krivuljo in štiri pomembna oglišča, ki jih imenujemo kontrolne točke krivulje. Takšna Bezierova krivulja je prikazana na sliki 6.4.



Slika 6.4. Bezierova krivulja s štirimi kontrolnimi točkami

Položaj krivulje v prostoru določajo kontrolne točke, medtem ko je oblika krivulje določena z Bernsteinovimi polinomi [96]. V primeru kubične krivulje obliko določajo štirje kubični Bernsteinovi polinomi, prikazani v enačbah od 6.1 do 6.4. Krivulja je nato kombinacija krivulj teh polinomov.

$$B_0(u) = (1-u)^3 \tag{6.1}$$

$$B_1(u) = 3u(1-u)^2 \tag{6.2}$$

$$B_2(u) = 3u^2(1-u) \tag{6.3}$$

$$B_3(u) = u^3 \tag{6.4}$$

Zato lahko enačbo Bezierove krivulje zapišemo v obliki enačbe 6.5, ki je vsota produktov koordinat kontrolnih točk in Bernsteinovih polinomov.

$$\boldsymbol{Q}(u) = \sum_{i=0}^{3} P_i B_i \tag{6.5}$$

Enačbo lahko tudi razvijemo in zapišemo v obliki enačbe 6.6, kjer vsako komponento točke izračunamo z enačbami 6.7, 6.8 in 6.9.

$$\boldsymbol{Q}(u) = \boldsymbol{P_0}(1-u)^3 + \boldsymbol{P_1}3u(1-u)^2 + \boldsymbol{P_2}3u^2(1-u) + \boldsymbol{P_3}u^3 \tag{6.6}$$

$$Q_x(u) = P_{0x}(1-u)^3 + P_{1x}3u(1-u)^2 + P_{2x}3u^2(1-u) + P_{3x}u^3 \tag{6.7}$$

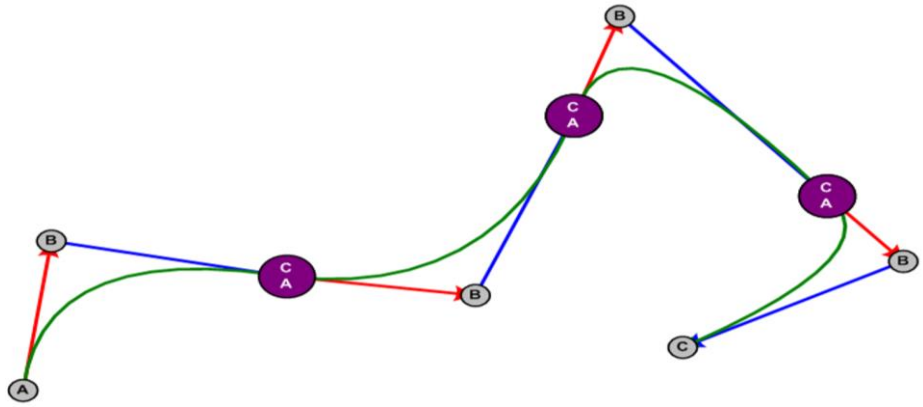
$$Q_y(u) = P_{0y}(1-u)^3 + P_{1y}3u(1-u)^2 + P_{2y}3u^2(1-u) + P_{3y}u^3 \tag{6.8}$$

$$Q_z(u) = P_{0z}(1-u)^3 + P_{1z}3u(1-u)^2 + P_{2z}3u^2(1-u) + P_{3z}u^3 \tag{6.9}$$

Enačbo 6.5 lahko zapišemo tudi v matrični obliki, kot je prikazano v enačbi 6.10.

$$\boldsymbol{Q}(u) = [u^3 \quad u^2 \quad u \quad 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{P_0} \\ \boldsymbol{P_1} \\ \boldsymbol{P_2} \\ \boldsymbol{P_3} \end{bmatrix} \tag{6.10}$$

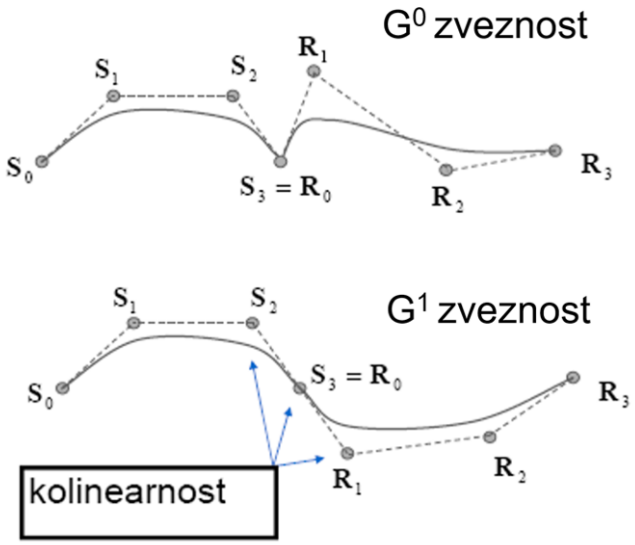
Z zgornjimi enačbami je mogoče določiti en odsek krivulje tako, da spreminjamo vrednosti parametra u od 0 do 1 s poljubnim korakom. Za vsako vrednost parametra u dobimo eno koordinato točke na krivulji. Ker je krivulja zvezna, je mogoče vedno določiti med dvema točkama še eno točko. Število točk na odseku krivulje je torej odvisno od ločljivosti, s katero razdelimo razpon med 0 in 1. Na podlagi tega lahko v Excelu preprosto narišemo en odsek krivulje tako, da določimo koordinato na krivulji tako, da povečujemo u od 0 s korakom 0,01 do vrednosti 1.



Slika 6.5. Več odsekov kvadratne Bezierove krivulje

Z enim odsekom krivulje lahko opišemo zgolj enostavno obliko krivulje, ki je lahko parabolična, če imamo samo kvadratno Bezierovo krivuljo, ali kubična, če je zapis Bezierove krivulje kubičen. Za bolj kompleksne oblike je treba sestavljati posamezne odseke krivulje tako, da dobimo želeno obliko. Na sliki 6.5je prikazana sestavljena krivulja, ki je sestavljena iz kvadratnih Bezierovih krivulj. Kvadratne Bezierove krivulje imajo na enem odseku samo tri kontrolne točke, krivulja pa je sestavljena iz treh kvadratnih Bernsteinovih polinomov.

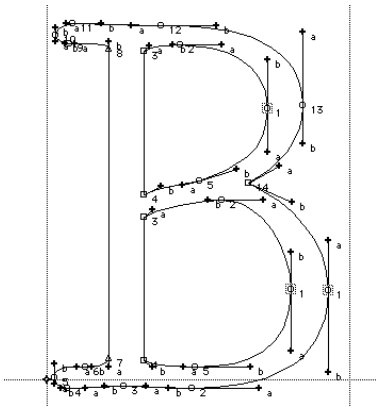
Pri sestavljanju odsekov Bezierovih krivulj postavimo začetno točko novega odseka v končno točko prejšnjega odseka. Pri tem se lahko zgodi, da izgubimo zveznost oziroma imamo samo zveznost G^0, kar pomeni, da se odseka stikata, vendar nimata enakih tangent v točki stika. Tangento dobimo tako, da določimo odvod funkcije v točki stika in enakost tangent pomeni, da je odvod funkcije prvega odseka enak odvodu funkcije drugega odseka na mestu stika. Ker smer tangente pri Bezierovih krivuljah na začetku in koncu odseka določajo kontrolne točke, velja pravilo, da morajo biti tri kontrolne točke pri spoju odsekov kolinearne.



Slika 6.6. Zveznost pri sestavljanju Bezierovih krivulj

Bezierove krivulje so tako zelo preproste za uporabo, vendar imajo kar nekaj pomanjkljivosti. Prva težava so kontrolne točke, ki ležijo na krivulji (začetna in končna točka odseka) in določajo obliko celotnega odseka. To pomeni, da ko premaknemo začetno kontrolno točko, ne spremenimo le položaja začetka krivulje, ampak se spremeni celotna krivulja. Ker želimo s krivuljami in posledično s površinami modelirati resnične izdelke, je včasih zaželeno, da lahko prilagodimo obliko samo enega dela krivulje. To pomeni, da želimo imeti lokalno kontrolo in prilagajati obliko vsakega dela krivulje. Tega Bezierove krivulje ne omogočajo, kar je posebna težava pri njihovi uporabi v namen 3D modeliranja. Druga še nekoliko bolj nenavadna težava je v tem, da s temi krivuljami ne moremo natančno opisati krožnih lokov, ampak lahko dobimo zgolj približek krožnega loka. Tretja težava je že omenjena težava z zagotavljanjem zveznosti, tako da moramo zagotoviti kolinearnost kontrolnih točk. Še nekaj dodatnih težav nastane pri modeliranju izdelkov z Bezierovimi površinami. Na začetku je bilo nekaj poskusov modeliranja s takšnimi površinami, ki so pokazali, da so te površine neuporabne za ta namen. V podjetju Renault temu tako niso namenili veliko pozornosti, saj je uprava menila, da bi takšne stvari verjetno že prej razvili v Združenih državah Amerike. Zato tudi njihovi modeli avtomobilov iz tistih časov ne kažejo posebnega razvoja v smeri 3D modeliranja in kompleksnih površin. Kljub temu so Bezierove krivulje danes najbolj uporabljane krivulje v računalništvu. Večina vsebine te strani je sestavljena iz Bezierovih krivulj.

Vse črke na računalnikih so danes sestavljene iz Bezierovih krivulj. V pionirskih časih računalnika so bile črke zapisane rastrsko s fiksnim številom pikslov in za večje črke je bila potrebna večja metrika pikslov. Zato so bile takrat črke na voljo le v nekaj različnih velikostih. Z razvojem laserskih tiskalnikov in grafičnih operacijskih sistemov je postalo to nesprejemljivo in je bilo treba razmišljati o vektorskem zapisu krivulj. Podjetje Adobe je leta 1984 predstavilo svoje Postscript fonte, ki so temeljili na kubičnih Bezierovih krivuljah. Zapis posamezne črke je vektorsko izdelan tako, kot je prikazano na sliki 6.7. Podatki o posameznih odsekih in kontrolnih točkah so zapisani v predpisani obliki in omogočajo poljubno skaliranje. Tako izdelane fonte je nato podjetje Adobe prodajalo drugim proizvajalcem tiskalnikov in grafičnih sistemov. Prvi operacijski sistem s Postscript fonti je bil Apple Macintosh leta 1984, nato pa je bil Postscript uporabljen tudi v Microsoft Windows 3.0 in številnih operacijskih sistemih, ki so temeljili na UNIX (SunOS, HP-UX, IBM-AIX, itd.).



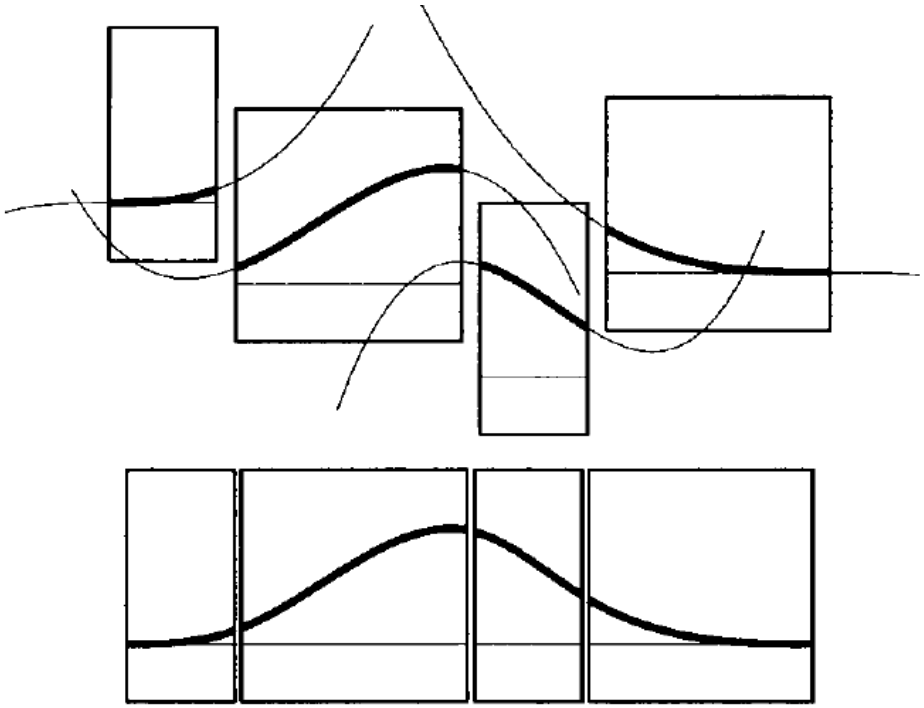
Slika 6.7. Vektorski zapis Postscript črke

Ker je bilo treba za Postscript plačati licenčnino in je bil font nekoliko zahteven za izrisovanje in skaliranje, so se v podjetju Apple odločili za izdelavo svojih fontov, ki so jih imenovali TrueType in so jih leta 1991 vključili v operacijski sistem Mac System 7 in tiskalnik Apple LaserWriter. Za večjo razširjenost teh pisavo so TrueType brezplačno dali na voljo tudi Microsoftu, ki jih je leta 1992 vključil v Windows 3.1. Pozneje so bili ti fonti uporabljeni tudi v drugih operacijskih sistemih (na primer Linux). Truetype fonti prav tako temeljijo na Bezierovih krivuljah, vendar so odseki Bezierovih krivulj izdelani z Bernsteinovimi polinomi druge stopnje. Zato je oblika nekoliko manj zaobljena, saj so krivulje le kvadratne namesto kubičnih v Postscript fontih. Posledično so TrueType fonti hitreje narisani in skalirani, vendar nekoliko manj lepi. Zaradi tega je izpis besedila veliko lepši, če besedilo spremenimo v Postscript obliko. Simbolični primer razlike med Postscript in TrueType fonti je prikazan na sliki 6.8.

Slika 6.8. Primerjava med Postscript (levo) in TrueType (desno) pisavo

6.1.2 B-spline krivulje

Carl-Wilhel Reinhold de Boor je bil rojen v Vzhodni Nemčiji, vendar je emigriral v Zahodno Nemčijo in nato v Združene države Amerike, ko je izvedel, da se mu ni uspelo vpisati na študij kemije v Berlinu (zaradi slabih ocen matematike). Na Harvardu je uspešno zaključil študij matematike in se nato zaposlil v podjetju General Motors. Tam so orodja za avtomobilske karoserije izdelovali tako, da so izdelali leseni model površine in ga potem kopirali z obdelovalnim strojem. Po leseni površini se je premikalo tipalo in enake gibe je potem izvajalo frezalo. DeBoor je vedel, da bi bil postopek veliko lažji, če bi lahko površino opisali matematično in bi na podlagi tega izdelali program za frezalni stroj. Zato se je začel ukvarjati s problemom matematičnega zapisa takšne površine. Ko se je zaposlil na Univerzi Winsconsin-Madison, kjer je sodeloval z Isaacom Jacobom Schoenbergom, si je zamislil podobne krivulje, kot so Bezierove krivulje. Namesto kombinacije Bernsteinovih polinomov je bila ta krivulja sestavljena iz zlepkov posameznih delov krivulj na podlagi Bernsteinovih polinov, tako kot je prikazano na sliki 6.9. Te krivulje so dobile ime B-spline [97], kjer je črka B oznaka za osnovne krivulje (angl. Basis spline).



Slika 6.9. Zlepljena B-spline krivulja

DeBoor je uporabil te krivulje in izdelal DeBoorov algoritem, ki je omogočil preprosto konstruiranje teh krivulj. Pri tem je določil tudi notacijo za zapis enačb, ki je bila uporabljena v prejšnjem poglavju za zapis Bezierovih krivulj. V splošnem so B-spline krivulje opisane z enačbo 6.11. V enačbi je u prameter, od katerega je odvisna lokacija na krivulji in je običajno definiran od 0 do 1. Število kontrolnih točk, ki geometrijsko določajo krivuljo, je n+1. Stopnja polinoma baznih funkcij je določena z vrednostjo k-1 (funkcije so kubične, ko je k=4). N so bazne funkcije, ki so določene z rekurzivnimi funkcijami, prikazanimi v enačbah 6.12 in 6.13. Rekurzivno pomeni, da v funkciji lahko funkcija kliče samo sebe.

$$\boldsymbol{Q}(u) = \sum_{i=1}^{n+1} \boldsymbol{P_i} N_{i,k}\,(u) \tag{6.11}$$

$$N_{i,0}(u) = \begin{cases} 1 \text{ če } u_i \le u \ < u_{i+1} \\ 0 \text{ če } zgornji\ pogoj\ ni\ izpolnjen \end{cases} \tag{6.12}$$

$$N_{i,k}(u) = \frac{(u-u_i)}{u_{i+k-1}-u_i} N_{i,k-1}(u) + \frac{(u_{i+k}-u)}{u_{i+k}-u_{i+1}} N_{i+1,k-1}(u) \tag{6.13}$$

Če vstavimo k=4, dobimo za štiri kontrolne točke štiri enačbe 6.14, 6.15, 6.16, 6.17 za bazne funkcije.

$$N_1(u) = \frac{1}{6}(1-u)^3 = \frac{1}{6}(-1u^3 + 3u^2 - 3u + 1) \tag{6.14}$$

$$N_2(u) = \frac{1}{6}3u(1-u)^2 = \frac{1}{6}(3u^3 - 6u^2 + 3u + 0) \tag{6.15}$$

$$N_3(u) = \frac{1}{6}3u(1-u^2) = \frac{1}{6}(-3u^3 + 0u^2 + 3u + 0) \tag{6.16}$$

$$N_4(u) = \frac{1}{6}u(u^2 + 4u + 1) = \frac{1}{6}(1u^3 + 4u^2 + 1u + 0) \tag{6.17}$$

Zgornje enačbe lahko uporabimo v matrični obliki in B-spline zapišemo z enačbo 6.18. V tej enačbi je predvideno, da so kontrolne točke razporejene enakomerno.

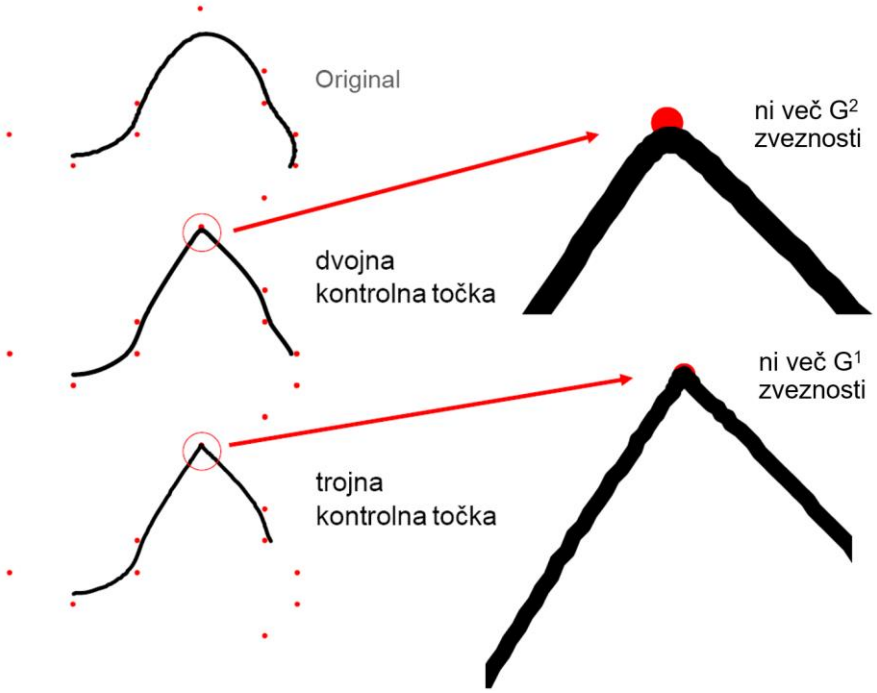
$$\boldsymbol{Q}(u) = [u^3 \quad u^2 \quad u \quad 1]\frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{P_0} \\ \boldsymbol{P_1} \\ \boldsymbol{P_2} \\ \boldsymbol{P_3} \end{bmatrix} \tag{6.18}$$

V splošni obliki je namesto tega uporaben DeBoorov algoritem v Python programskem jeziku, prikazan na sliki 6.10, ki izračuna koordinate točk tako, kot ga je zapisal ChatGPT [99].

```
def deboor(knots, P, degree, u):
    """
    Izračuna točko na B-zlepku v parametru u z uporabo algoritma De Boor.
    :param knots: Vezni vektor B-zlepka.
    :param P: Kontrolne točke B-zlepka.
    :param degree: Stopnja B-zlepka.
    :param u: Parameter vrednosti za izračun točke na B-zlepku
    :return: Točka na B-zlepku v parametru u.
    """
    # Poišči indeks prve nenule bazne funkcije
    span = find_span(knots, degree, u)

    # Inicializiraj De Boor točke s kontrolnimi točkami v trenutnem span
    de_boor_pts = P[span-degree:span+1]

    # Za vse stopnje algoritma De Boor
    for k in range(1, degree+1):
        # Ustvari prazno seznam za shranjevanje naslednjih De Boor točk
        next_level = []

        # Za vse De Boor točke tekoče stopnje
        for i in range(span-degree+k, span-k+1):
            # Izračunaj naslednjo De Boor točko z rekurzivno formulo
            alpha = (u - knots[i]) / (knots[i+degree-k+1] - knots[i])
            next_pt = (1.0 - alpha) * de_boor_pts[i-span+degree] + alpha *
de_boor_pts[i-span+degree+1]
            next_level.append(next_pt)

        # Posodobi De Boor točke za naslednjo stopnjo
        de_boor_pts = next_level

    # Vrni zadnjo De Boor točko, ki je točka na B-zlepku v parametru u
    return de_boor_pts[0]
```

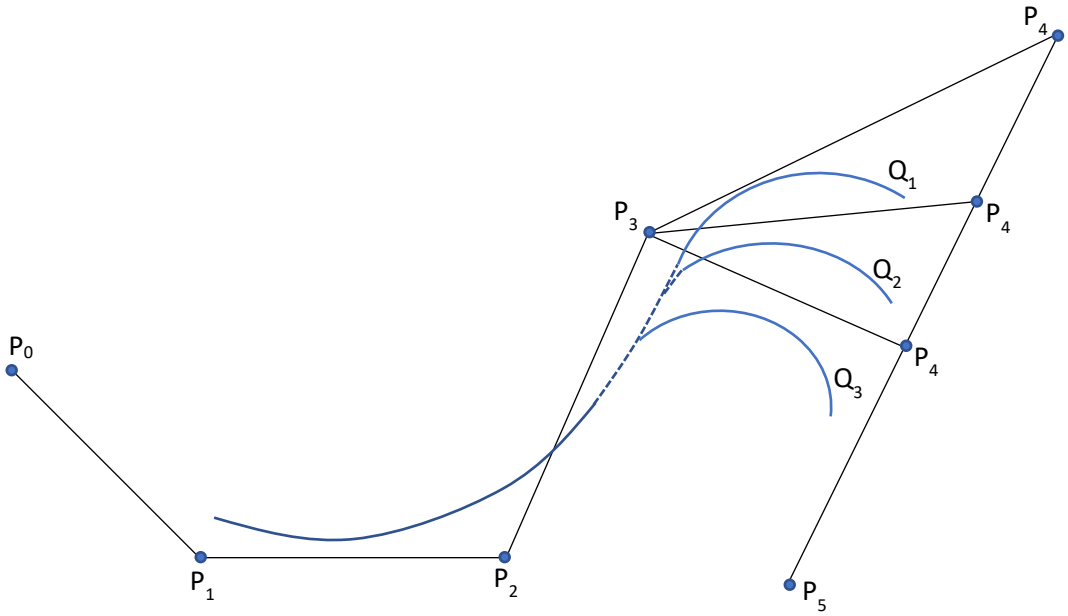
Slika 6.10. DeBoorov algoritem za B-spline krivuljo [99]

V takšni splošni obliki lahko imamo več kontrolnih točk postavljenih v isto točko v prostoru. Tako lahko krivuljo prilagajamo dejanski obliki, vendar dvojna točka pomeni tudi, da zmanjšamo zveznost krivulje tako, kot je prikazano na sliki 6.11.



Slika 6.11. Vpliv neenakomerno razporejenih kontrolnih točk

Poleg tega ima krivulja tudi lokalno kontrolo, kar pomeni, da se s premikom kontrolne točke spremeni samo del krivulje v bližini te kontrolne točke. Tako lahko prilagajamo posamezne dele krivulje želeni obliki. Pri Bezierovi krivulji se s premikom kontrolne točke spremeni celotna oblika odseka krivulje, medtem ko se pri B-spline krivulji spremeni samo del krivulje, kot je prikazano na sliki 6.12.



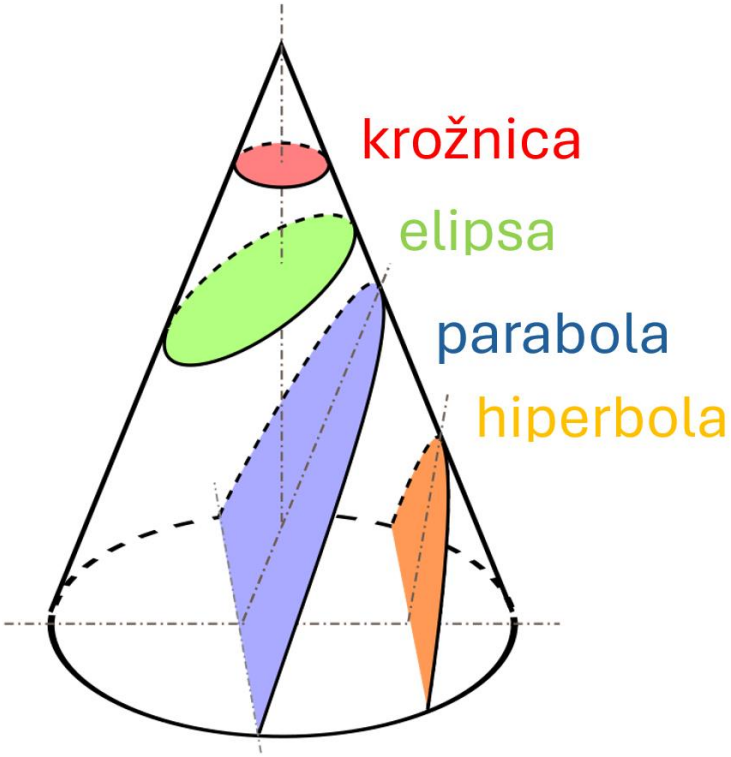
Slika 6.12. Lokalna kontrola B-spline krivulj

Pri premiku kontrolnih točk in lokalni spremembi geometrije krivulje se ne spremeni zveznost krivulje. V splošnem predstavitev B-spline krivulje ni možno zgolj opisati tako kot Bezierove krivulje, vendar so razviti algoritmi za določanje točk krivulje izjemno učinkoviti. Takšna fleksibilnost B-spline krivulj je omogočala izdelavo B-spline površin, ki jih je bilo mogoče koristno uporabiti za numerično modeliranje

površin v 3D modelirnikih. Krivulje so manj uporabne za interpolacijo geometrije, zato je treba upoštevati, da vedno dobimo le aproksimacijo površine. V modelirnikih to ni moteče, saj geometrijo ustvarjamo, težko pa je s takšnimi krivuljami natančno opisati realno skenirano površino.

6.1.3 NURBS krivulje

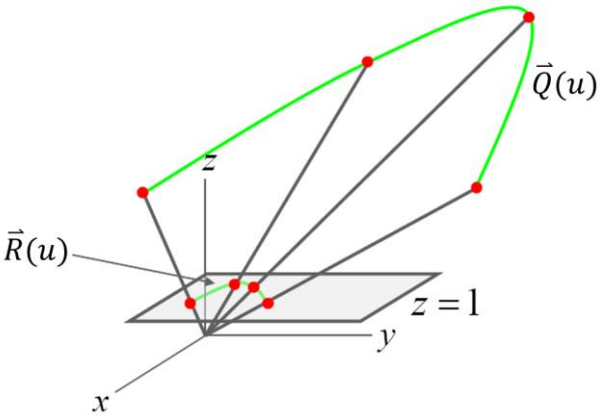
Čeprav B-spline krivulje dobro izpolnjujejo zahteve v računalniški grafiki in 3D modeliranju, se je izkazalo, da so NURBS krivulje [100] še boljše in predstavljajo takšen zapis krivulj, s katerimi je mogoče modelirati vse preostale krivulje. NURBS (angl. Non-Uniform Rational B-Spline) je kratica, ki pove, da gre za B-spline krivulje z neenakomerno razporeditvijo kontrolnih točk, ki so zapisane z racionalno enačbo (enačba v obliki ulomka). Do takšne rešitve so prišli z združitvijo koncepta B-spline krivulj in stožnic, ki so jih uporabljali v letalski industriji. Stožnice [101] so bile pogosto kreirane tako, da so uporabili presečno ravnino, s katero so prerezali obliko in kot rezultat dobili krivuljo na ravnini, kot je prikazano na sliki 6.13.



Slika 6.13. Stožnice [102]

Podoben rezultat lahko dobimo, če projiciramo krivuljo na površino. Pri projekciji 3D krivulje $\boldsymbol{Q}(u)$ na površino dobimo 2D krivuljo $\boldsymbol{R}(u)$ na sliki 6.14, ki jo lahko zapišemo z enačbo 6.19. Enačba je racionalne oblike, ker smo jo delili z ravnino z=1 in posledično koordinate x in y delimo s koordinato z. Na tak način bi lahko dobili le 2D racionalne krivulje. Čeprav si je nemogoče predstavljati 4D krivulje, je matematično mogoče zapisati 4D krivuljo, ki jo projiciramo v 3D prostor.

$$\boldsymbol{R}(u) = \left(\frac{x(u)}{z(u)}, \frac{y(u)}{z(u)}, 1\right) \tag{6.19}$$

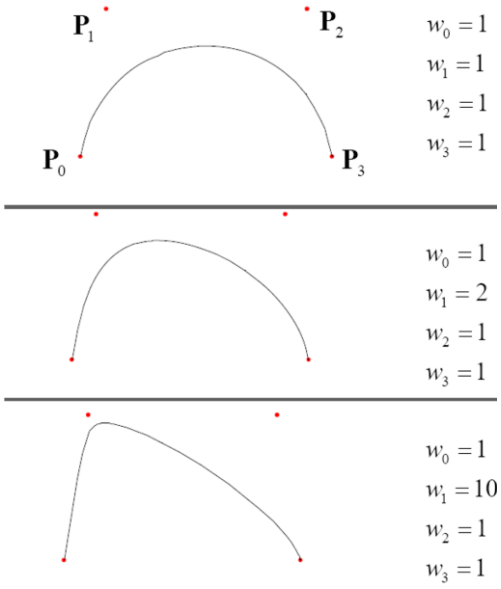


Slika 6.14. Projekcija 3D krivulje na ravnino

Pri 4D krivuljah dodamo vsaki kontrolni točki še eno koordinato, ki jo imenujemo w. Ta koordinata w predstavlja utež, s katero kontroliramo projekcijo krivulje. Tako lahko projiciramo Bezierovo krivuljo, zapisano z enačbo 6.5, definirano v 4D prostoru z dodatnimi koordinatami w. Po projiciranju dobimo racionalno krivuljo, zapisano v enačbi 6.20.

$$\boldsymbol{Q}(u) = \frac{\sum w_i \boldsymbol{P_i} B_i(u)}{\sum w_i B_i(u)} \tag{6.20}$$

Takšne racionalne Bezierove krivulje imajo veliko boljše lastnosti kot običajne Bezierove krivulje. Z njimi lahko na primer opišemo krožni lok, česar ni mogoče z običajno Bezierovo krivuljo. Poleg tega te krivulje omogočajo tudi lokalno kontrolo, in če uporabljamo uteži, lahko spreminjamo krivuljo tako, kot je prikazano na sliki 6.15.



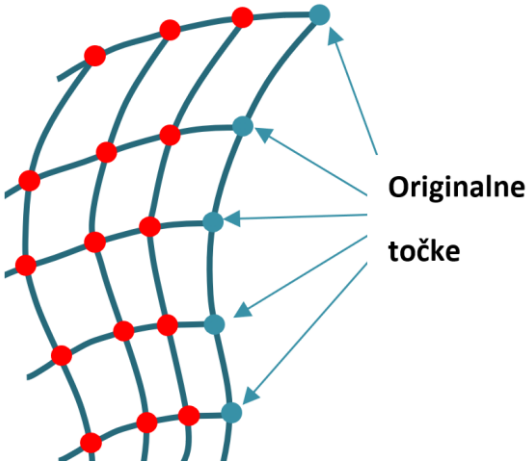
Slika 6.15. Lokalna kontrola racionalne Bezierove krivulje

Zato je krivulja še veliko boljša, če spremenimo B-spline krivuljo v racionalno krivuljo. To naredimo tako, da uporabimo enačbo 6.11 in jo projiciramo iz 4D prostora v 3D prostor, da dobimo NURBS krivuljo zapisano v enačbi 6.21.

$$\boldsymbol{Q}(u) = \frac{\sum w_i \boldsymbol{P_i} N_{i,k}(u)}{\sum w_i N_{i,k}(u)} \tag{6.21}$$

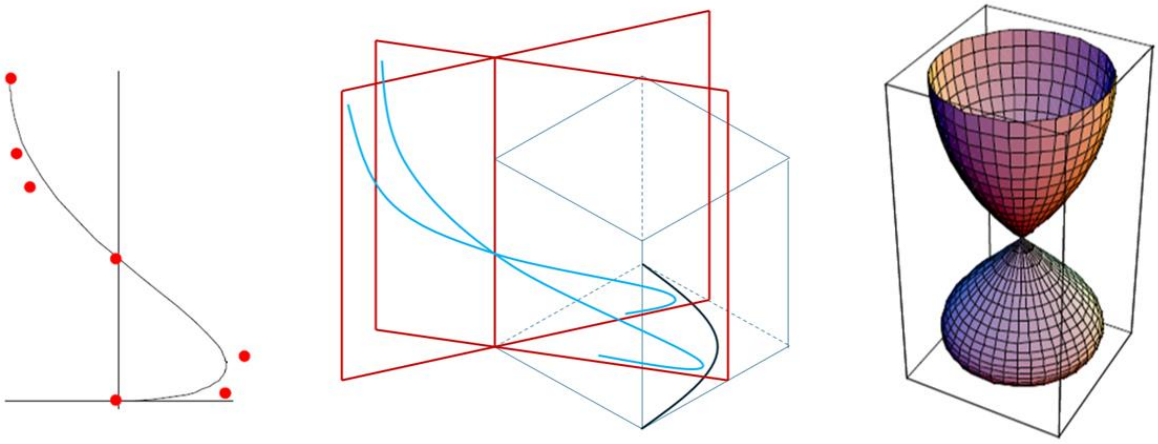
6.2 Parametrične površine

Krivulje je dokaj preprosto uporabiti za kreiranje površin. Najenostavneje je uporabiti geometrijsko transformacijo in obstoječo krivuljo raztegnemo ali zavrtimo v prostor z določenim korakom. V vsakem koraku na krivulji določimo točke, ki jih nato povežemo v poligone, kot je prikazano na sliki 6.16. Tako ustvarjene poligone, ki predstavljajo površino, imenujemo krpice (angl. Patch), ki so lahko štirikotniki ali trikotniki.



Slika 6.16. Transformacija krivulje

Krivulja, ki jo uporabimo za transformacijo, je lahko parametrična krivulja iz prejšnjega poglavja. Na sliki 6.17 je prikazana transformacija dveh pravokotno postavljenih Bezierovih krivulj, iz katerih potem kreiramo površino z generiranjem točk na krivuljah in povezovanje teh točk v poligone. Obe pravokotno postavljeni krivulji sta zvezni, kar pomeni, da lahko na krivuljah določimo neskončno število točk. Več točk pomeni več poligonov in s tem bolj gladke površine. Hkrati je mogoče spreminjati takšno površino, saj s premikom kontrolnih točk originalne krivulje dobimo drugačno površino. Ko spreminjamo gladkost krivulje, hkrati spremenimo gladkost površine. Če želimo oster prehod na površini, naredimo tak prehod na krivulji.

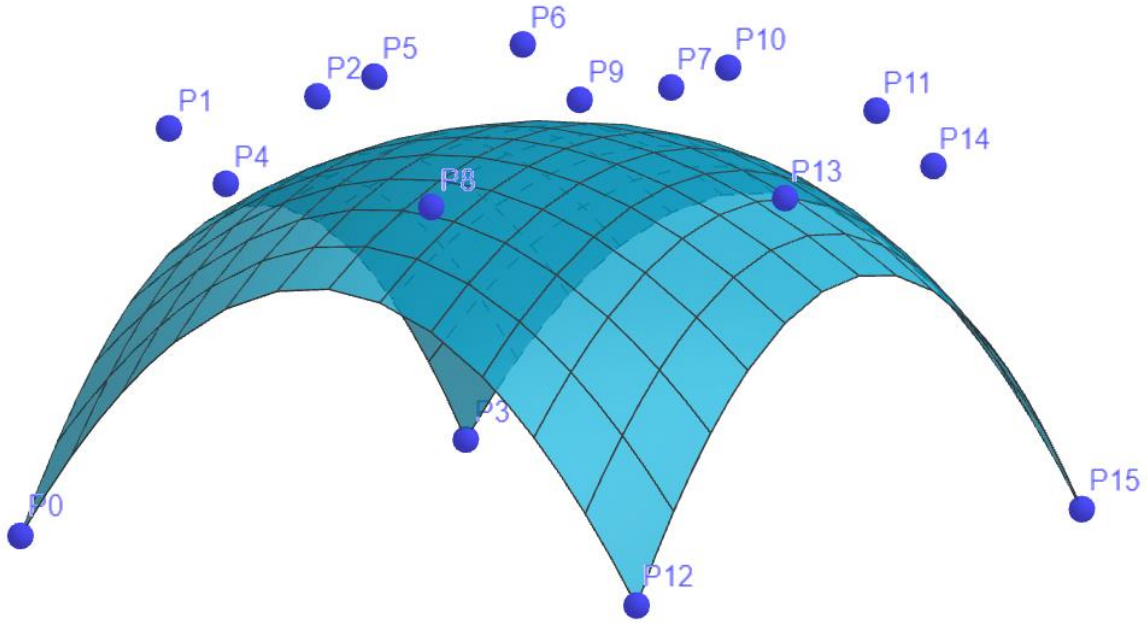


Slika 6.17. Transformacija Bezierove krivulje

Seveda takim površinam težko rečemo parametrične površine, čeprav je krivulja definirana s parametrično krivuljo. Zato je cilj kombinacije krivulj na tak način, da bo mogoče celotno površino opisati s parametrično enačbo. Šele potem bo mogoče določiti koordinate točk na poljubnem mestu površine.

6.2.1 Bezierove površine

Bezierove površine [103] kreiramo tako, da povežemo dve Bezierovi krivulji v površino. Ena krivulja je lahko definirana s parametrom u, druga pa s parametrom v, pri čemer oba parametra spreminjamo na intervalu med 0 in 1. Tako dobimo en odsek kubične Bezierove površine, prikazane na sliki 6.18, ki jo določa 16 kontrolnih točk, označenih s P1 do P15.



Slika 6.18. Bezierova površina

Tudi enačba Bezierove površine je definirana sorazmerno preprosto s produktom dveh krivulj in je prikazana v enačbi 6.22.

$$\boldsymbol{Q}(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} \boldsymbol{P_{i,j}} B_i(u) B_j(v) \tag{6.22}$$

Bolj pregledna je morda enačba v matrični obliki, zapisana z enačbo 6.23, kjer sta **B** in **P** zapisani v matrični obliki v enačbi 6.24.

$$\boldsymbol{Q}(u,v) = [u^3 \quad u^2 \quad u \quad 1]\mathbf{B_z}\mathbf{P}\mathbf{B_Z^T}\begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix} \tag{6.23}$$

$$\mathbf{B}_z = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} \boldsymbol{P_0} & \boldsymbol{P_1} & \boldsymbol{P_2} & \boldsymbol{P_3} \\ \boldsymbol{P_4} & \boldsymbol{P_5} & \boldsymbol{P_6} & \boldsymbol{P_7} \\ \boldsymbol{P_8} & \boldsymbol{P_9} & \boldsymbol{P_{10}} & \boldsymbol{P_{11}} \\ \boldsymbol{P_{12}} & \boldsymbol{P_{13}} & \boldsymbol{P_{14}} & \boldsymbol{P_{15}} \end{bmatrix} \tag{6.24}$$

Zato lahko enačbo razvijemo in jo v obliki enačbe 6.25 vstavimo v poljuben program za prikazovanje površin. Dodati je treba kontrolne točke in razpon parametrov u in v in iz enačbe 6.25 lahko izračunamo koordinato poljubne točke na površini.

$$\begin{aligned} \boldsymbol{Q}(u,v) = {} & (1-\mathrm{u})^3\,(1-\mathrm{v})^3\boldsymbol{P_0} + (1-\mathrm{u})^3 3\mathrm{v}\,(1-\mathrm{v})^2\boldsymbol{P_1} + (1-\mathrm{u})^3 3v^2(1-\mathrm{v})\boldsymbol{P_2} + \\ & +(1-\mathrm{u})^3 v^3\,\boldsymbol{P_3} + 3\mathrm{u}\,(1-\mathrm{u})^2(1-\mathrm{v})^3\boldsymbol{P_4} + 3\mathrm{u}\,(1-\mathrm{u})^2 3\mathrm{v}\,(1-\mathrm{v})^2\boldsymbol{P_5} + \\ & +3\mathrm{u}\,(1-\mathrm{u})^2 3\,v^2(1-\mathrm{v})\boldsymbol{P_6} + 3\,\mathrm{u}\,(1-\mathrm{u})^2\,v^3\boldsymbol{P_7} + 3u^2\,(1-\mathrm{u})(1-\mathrm{v})^3\boldsymbol{P_8} + \\ & +3u^2(1-\mathrm{u})3\mathrm{v}(1-\mathrm{v})^2\boldsymbol{P_9} + 3u^2(1-\mathrm{u})3v^2(1-\mathrm{v})\boldsymbol{P_{10}} + 3u^2(1-\mathrm{u})v^3\boldsymbol{P_{11}} + \\ & +u^3(1-\mathrm{v})^3\boldsymbol{P_{12}} + u^3 3\,\mathrm{v}\,(1-\mathrm{v})^2\,\boldsymbol{P_{13}} + u^3 3\,v^2(1-\mathrm{v})\,\boldsymbol{P_{14}} + u^3\,v^3\,\boldsymbol{P_{15}} \end{aligned} \tag{6.25}$$

Z enačbo Bezierove površine lahko določimo, s poljubno gostoto, oglišča poligonov, s katerimi nato izrišemo površino. Za površine v računalniški grafiki je pomembno, da lahko v vsaki točki površine določimo tudi tangento in normalo površine. Ker je površina zvezna, lahko to naredimo tako, da enačbo 6.22 odvajamo po parametrih u in v. Odvod enačbe 6.23 po parametru u je prikazan v enačbi 6.26, odvod po parametru v pa v enačbi 6.27. Odvoda predstavljata tangenti površine v različnih smereh in ju lahko določimo v vsaki točki površine.

$$\boldsymbol{Q}_u(u,v) = [3u^2 \quad 2u \quad 1 \quad 0]\mathbf{B_z P B_Z^T}\begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix} \tag{6.26}$$

$$\boldsymbol{Q}_v(u,v) = [u^3 \quad u^2 \quad u \quad 1]\mathbf{B_z P B_Z^T}\begin{bmatrix} 3v^2 \\ 2v \\ 1 \\ 0 \end{bmatrix} \tag{6.27}$$

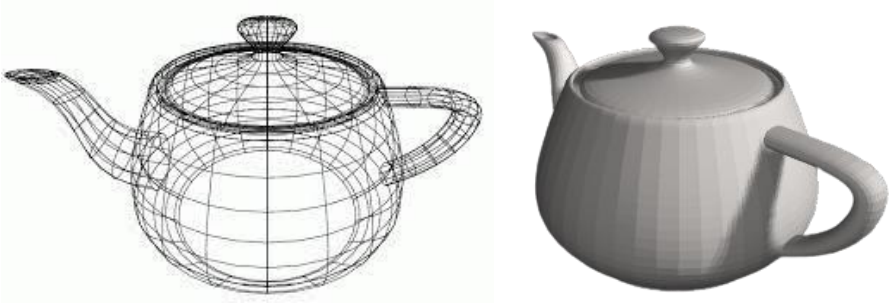
Z vektorskim produktom tangent lahko določimo normale v vsaki točki površine z enačbo 6.28.

$$\boldsymbol{N}(u,v) = \boldsymbol{Q}_u(u,v) \times \boldsymbol{Q}_v(u,v) \tag{6.28}$$

Z mešanim odvodom v enačbi 6.29 lahko določimo zasuk površine v katerikoli točki površine. Z zasukom lahko kontroliramo lokalno deformacijo in orientacijo površine. Tako bo zasuk na površini valja vedno enak 0, ker je normalni vektor vedno pravokoten na os valja.

$$\boldsymbol{Q}_{uv}(u,v) = [3u^2 \quad 2u \quad 1 \quad 0]\mathbf{B_z P B_Z^T}\begin{bmatrix} 3v^2 \\ 2v \\ 1 \\ 0 \end{bmatrix} \tag{6.29}$$

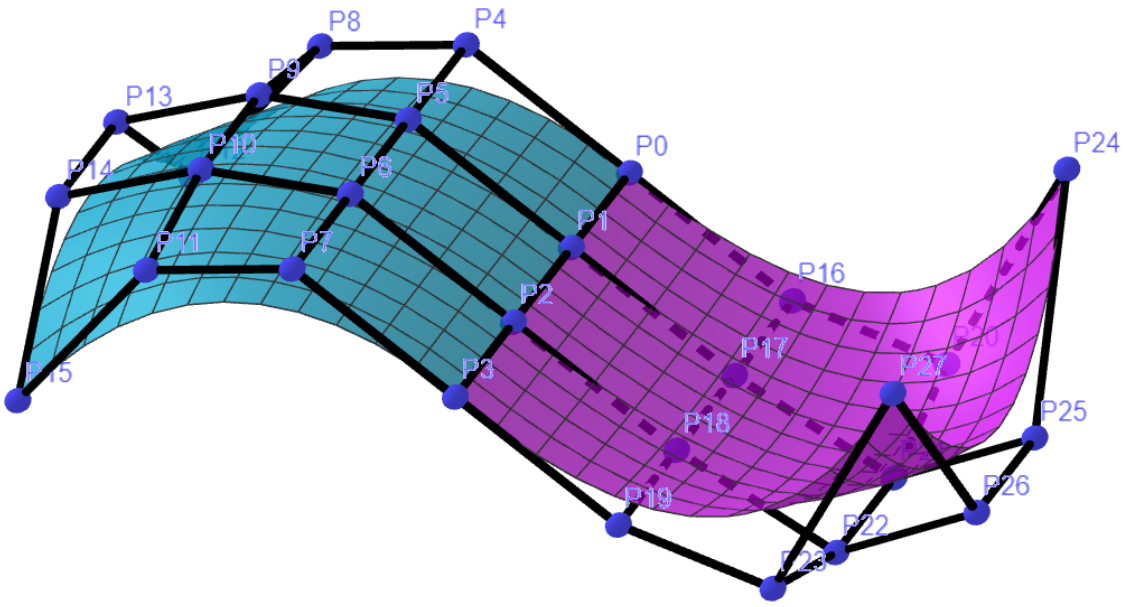
S tako jasno določenimi matematičnimi površinami je bilo mogoče v 70. letih 20. stoletja prikazovati takšne površine in jih uporabiti za prve površinske modelirnike.



Slika 6.19. Modeliranje z Bezierovimi površinami

Na sliki 6.19 je model, ki ga je izdelal Martin Newell na Univerzi Utah v okviru doktorske naloge. Za izdelavo je uporabil Bezierove površine in za model izbral domači čajnik. Čajnik je potem postal vzorčni model za uporabo novejših tehnik računalniške grafike, ker je v svojem delu objavil geometrijske podatke modela in so ga lahko uporabljali številni raziskovalci. Bezierove površine se niso izkazale za najboljšo izbiro za modeliranje izdelkov. Čajnik tako ni imel dna. Ročaj in izlivnik nista bila ustrezno spojena, saj površine niso omogočale takšnega prilagajanja, da bi jih bilo mogoče spojiti brez vrzeli. Številne pomanjkljivosti Bezierovih površin izvirajo že iz krivulj, saj nimamo lokalne kontrole in s površinami lahko opišemo le približek sfere. Tudi pri spajanju površin je treba skrbeti za ohranjanje zveznosti na prehodu med odseki površin.

Pi spoju dveh odsekov Bezierovih površin je treba skrbeti, da so točke na spoju površin koplanarne. Na sliki 6.20 morajo biti točke P4-P7 in P16-P19 na enaki ravnini s točkami P0-P3 na spoju površin. Le tako bodo tangente na spoju enake na obeh odsekih površin.



Slika 6.20. Zveznost dveh odsekov Bezierovih površin

S premikanjem kontrolnih točk lahko zagotovimo tudi koplanarnost točk, vendar premik vsake kontrolne točke spremeni celotno površino. Tudi zato so Beziereve površine manj primerne za 3D modeliranje.

6.2.2 NURBS površine

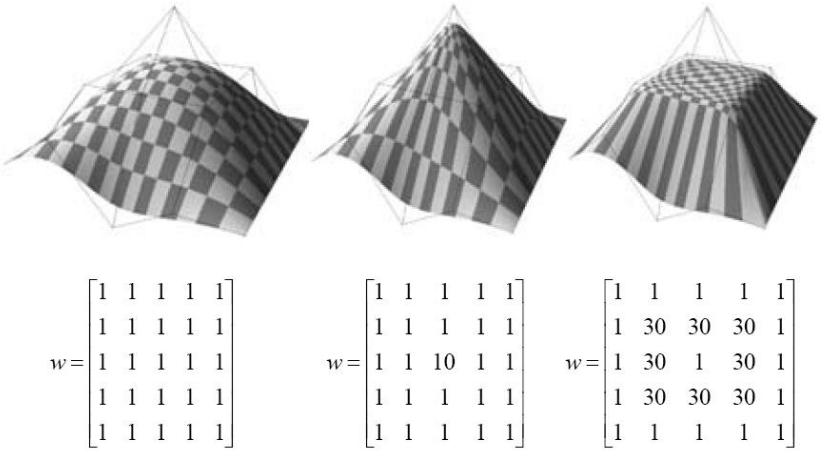
B-spline površine, ki temeljijo na kombinaciji B-spline krivulj, so se izkazale za veliko bolj primerne za modeliranje. Posebej uporabne so površine, ki temeljijo na racionalnih B-spline površinah, kjer so kontrolne točke neenakomerno razporejene. Takšne NURBS površine [100] lahko zapišemo z enačbo 6.30, kjer so $N_{i,p}(u)$ in $N_{i,q}(v)$ bazične funkcije, w_i uteži v kontrolnih točkah in $\boldsymbol{P_i}$ koordinate kontrolnih točk.

$$\boldsymbol{Q}(u,v) = \frac{\sum N_{i,p}(u) N_{i,q}(v) w_i \boldsymbol{P_i}}{\sum N_{i,p}(u) N_{i,q}(v) w_i} \tag{6.30}$$

Težava s to enačbo je v tem, da je ni tako preprosto spremeniti v enačbo, da bi bila zgolj odvisna od parametrov in kontrolnih točk, kot je enačba za Bezierovo površino 6.25. Zato je površina običajno določena z enačbo, zapisano v obliki računalniškega programa po DeBoorovem algoritmu. Osnovni postopek za določanje NURBS površine je naslednji:

- Definicija kontrolnih točk in uteži NURBS površine.
- Definicija vozliščnih vektorjev za površino v smeri u in v.
- Programska določitev baznih funkcij za parametra u in v z uporabo vozliščnih vektorjev in stopnjo polinoma površine.
- Množenje vsake kontrolne točke z njeno ustrezno utežjo in ustrezno bazno funkcijo.
- Seštevek prispevkov množenja določa koordinato točke na površini v odvisnosti od u in v.

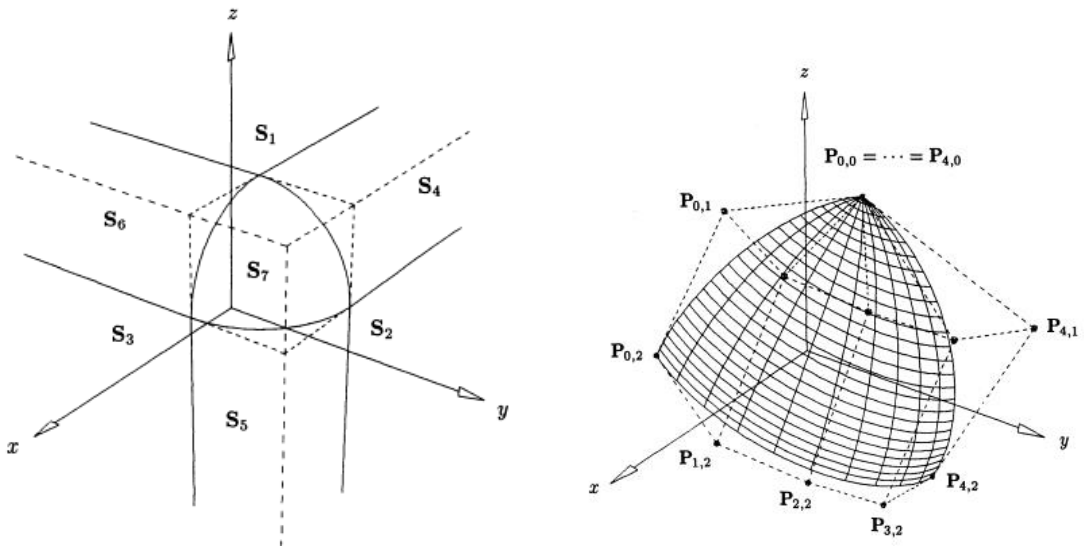
S spreminjanjem koordinat kontrolnih točk in pripadajočih uteži lahko prilagajamo obliko površine. V CAD programu so modeli zgrajeni tako, da se za vsako površino določijo kontrolne točke in uteži tako, da dobimo natančno obliko želenega izdelka. Navadno tega ni mogoče narediti samo z enim odsekom površine, ampak je površina zgrajena iz več odsekov, ki so med seboj zvezno povezani. Zveznost povezav je veliko lažje zagotoviti, saj imajo NURBS površine lokalno kontrolo in se s spremembo ene kontrolne točke ne spremeni celotna površina. Stopnja polinoma NURBS površine v CAD programih je določena interno v programu in se običajno giblje med 3 in 5. Stopnja polinoma določa, kako hitro se krivulja spreminja v različnih območjih površine in koliko kontrolnih točk je potrebno za njeno določanje. Stopnja 3 pomeni, da se krivulja spreminja počasi in potrebuje manj kontrolnih točk, kar omogoča hitrejše izrisovanje površine in manjšo potrebo po spominu. Stopnja 5 pa pomeni, da se krivulja spreminja hitreje in potrebuje več kontrolnih točk, kar omogoča bolj natančen izris površine, vendar zahteva več pomnilnika. Obstajajo tudi CAD programi, ki omogočajo dinamično spreminjanje stopnje polinoma, kar omogoča večjo natančnost v nekaterih območjih površine in hitrejši izris v drugih območjih.



$$w=\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix} \quad w=\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&10&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix} \quad w=\begin{bmatrix}1&1&1&1&1\\1&30&30&30&1\\1&30&1&30&1\\1&30&30&30&1\\1&1&1&1&1\end{bmatrix}$$

Slika 6.21. Prilaganje oblike NURBS površine z utežmi

Na sliki 6.21 je prikazano prilagajanje površine samo s spremembo velikosti uteži. Kreiranje površine, ki je rezultat treh zaokrožitev na vogalu kocke, je prikazano na sliki 6.22.



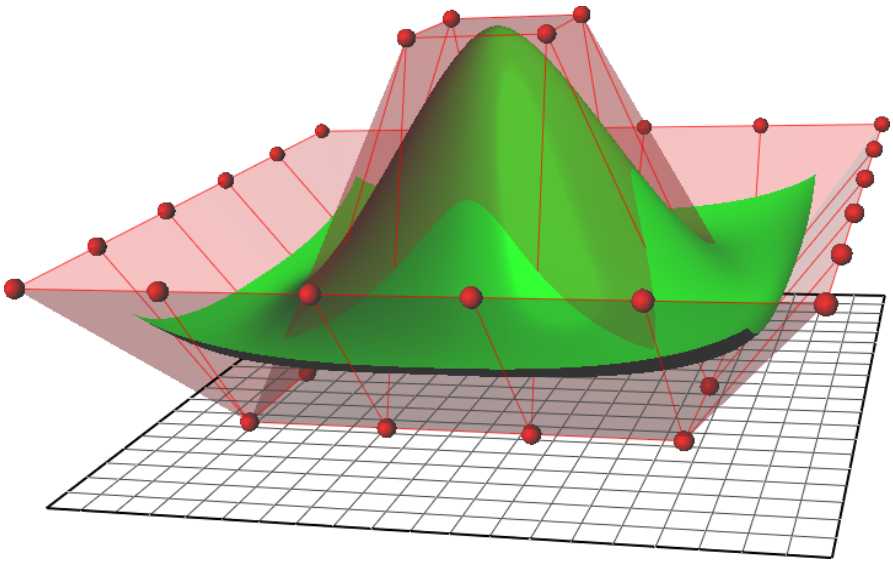
Slika 6.22. Kreiranje NURBS površine med modeliranjem

6.2.3 Renderiranje parametričnih površin

Površina parametrične površine je natančno določena in je po celotni površini zvezna. To pomeni, da lahko v katerikoli točki določimo tangente in normalo. Normala v vsaki točki je pomembna zato, ker lahko potem površino obarvamo v odvisnosti od položaja vira svetlobe. Večja težava je v tem, kaj obarvati. Zadostovalo bi obarvati samo posamezne piksle, ki jim določimo barvo z metodo sledenja žarku. Pri tej metodi, ki bo opisana v poglavju 11.3, je treba določiti presečišče s površino in na mestu presečišča določiti barvo piksla. Kot je omenjeno v prejšnjem poglavju, imamo pri NURBS parametričnih površinah najmanj kubične oblike površin ali površine z višjimi stopnjami polinoma. Določanje presečišča v vsaki točki površine je lahko zato časovno precej zahtevno in ne omogoča renderiranja v realnem času. Z razvojem grafične strojne opreme je mogoče, da bo v prihodnosti računanje presečišč tako hitro, da bo mogoče neposredno renderirati parametrične površine.

V današnjem času je zato vedno treba pretvoriti parametrične površine v poligonske površine. Že na začetku poglavja 6.2 je zapisano, da se površine predstavljajo s krpicami oziroma s poligoni, ki so kreirani na površini. Velikost teh poligonov je lahko poljubna, vendar ni velike potrebe po kreiranju zelo malih poligonov, saj z renderiranjem dobimo zelo gladke predstavitve izdelkov tudi, če uporabimo sorazmerno velike poligone. Ker so površine določene parametrično, je mogoče sorazmerno enostavno določiti mrežo poligonov v odvisnosti od parametrov u in v. V sodobnih modelirnikih je model lahko kakovostno renderiran tudi med modeliranjem. S tehniko sledenja žarkom v realnem času bodo kmalu simulirani tudi učinki refleksije in loma svetlobe v realnem času. To velja seveda ob predpostavki, da je parametrični model predstavljen s poligonsko mrežo, ki omogoča preprosto računanje presečišč v vsaki točki poligonske površine.

V CAD modelirnikih imamo zato vedno podvojeno strukturo. Model je interno shranjen in upravljan z natančno geometrijo NURBS parametričnih površin. Za vizualizacijo je treba matematični model opremiti še s predstavitvijo v obliki poligonske mreže. V vsakem koraku modeliranja sta določena interna natančna geometrija in poligonska geometrija modela. V določenih operacijah modeliranja se lahko zgodi, da program zaradi geometrijskih razlogov ali zaradi pomanjkljivosti programske opreme ne zmore najti primernega matematičnega zapisa za uporabnikov ukaz. Zato v takih primerih ni mogoče izdelati internega modela in posledično tudi poligonske mreže in uporabnik mora storiti korak nazaj in uporabiti drugačen ukaz. Na sliki 6.23 je predstavljena renderirana površina NURBS s prikazano interno strukturo kontrolnih točk.



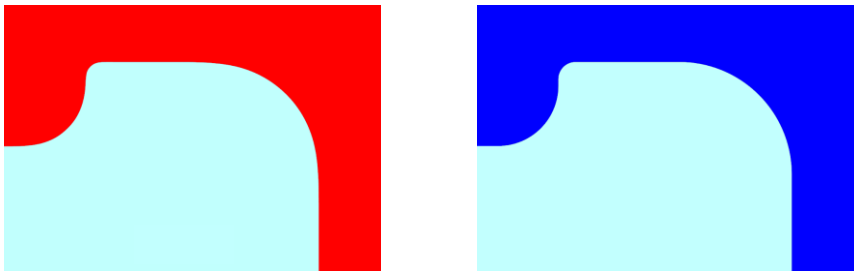
Slika 6.23. Površina NURBS [104]

7 Glajenje površin

Gladkost implicitnih in parametričnih površin, obravnavanih v prejšnjih poglavjih, je definirana s stopnjo polinoma, s katero so površine opisane. Kljub temu se srečujemo s problemom gladkosti na prehodih med eno in drugo površino ali med posameznimi odseki površin. Veliko večja težava je gladkost poligonskih površin, saj je odvisna tako od števila kot tudi od postavitve poligonov. Ker so skoraj vse površine pri renderiranju predstavljene s poligoni, je ta težava praktično neodvisna od načina modeliranja. Zato bodo v tem poglavju najprej predstavljeni vrednotenje, opazovanje in popravljanje gladkosti površin. Nato pa bo predstavljena dodatna metoda za modeliranje površin, ki temelji na glajenju z deljenjem površin (angl. subdivision surface). V okviru teh metod so nastali številni algoritmi, ki omogočajo modeliranje površin na nekoliko drugačen, manj matematičen način.

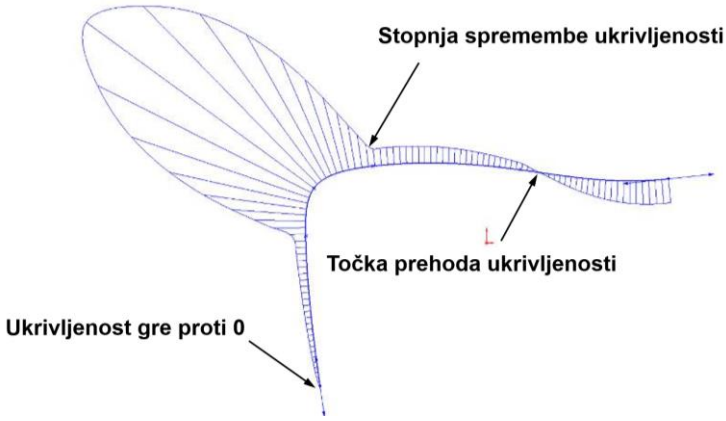
7.1 Gladkost površin

Gladkost površin [105] je definirana z zveznostjo površin. Čeprav so določene površine izdelane z zaokrožitvami, lahko včasih s prostim očesom opazimo, da površina ni gladka. Včasih so razlike zelo majhne, vendar so vseeno precej pomembne. Na sliki 7.1 je na levi strani oblika zaslona mobilnega telefona Apple in na desni zaslon, kjer je vogal zaslona le zaokrožen. Razliko opazimo le z zelo natančnim opazovanjem, če bi na primer sliki izrezali in ju položili eno na drugo ali opazovali računalniško animacijo prekrivanja slik.



Slika 7.1. Apple zaslon na levi in zaslon z zaokrožitvijo na desni

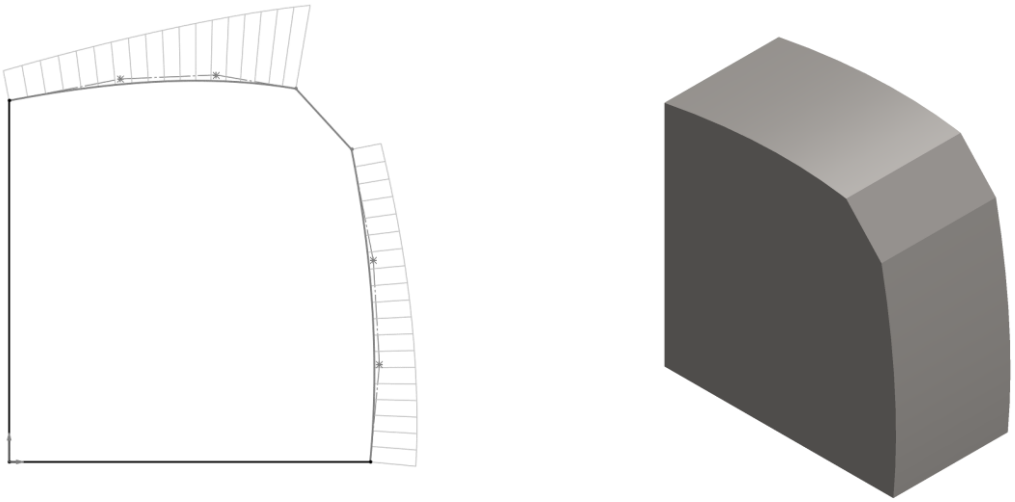
V določenih primerih, ko imamo samo zaokrožitev, lahko na izdelku opazimo rob ali ga otipamo. Zato je pri oblikovanju izdelkov velik poudarek na gladkosti robov in površin. Zelo koristna so različna orodja za spremljanje ukrivljenosti. Na sliki 7.2 je prikazano orodje, ki nam vektorsko ponazori, kako se spreminja ukrivljenost krivulje. Lahko vidimo stopnjo ukrivljenosti, ki je ponazorjena z velikostjo normalnih vektorjev, hkrati pa lahko opazujemo spremembo ukrivljenosti. Posebej so zanimivi prehodi med različnimi stopnjami ukrivljenosti, ki morajo tudi biti čim bolj gladki.



Slika 7.2. Spremljanje ukrivljenosti krivulje

G^0 zveznost

V določenih primerih se dve površini stikata, vendar prehod ni gladek. Površini sta spojeni in povezani z robom. Tak primer dobimo, če odrežemo vogal (na primer z ukazom champfer). Seveda so tako povezane površine na vseh oglatih telesih. Ploskve kvadra so na robovih povezane z G^0 zveznostjo. Oznaka 0 označuje odvod funkcije in ničti odvod pomeni, da je v točkah spoja le vrednost funkcije enaka. Na mestu spoja ima vsaka površina različni tangenti in v matematičnem smislu ni zvezna.

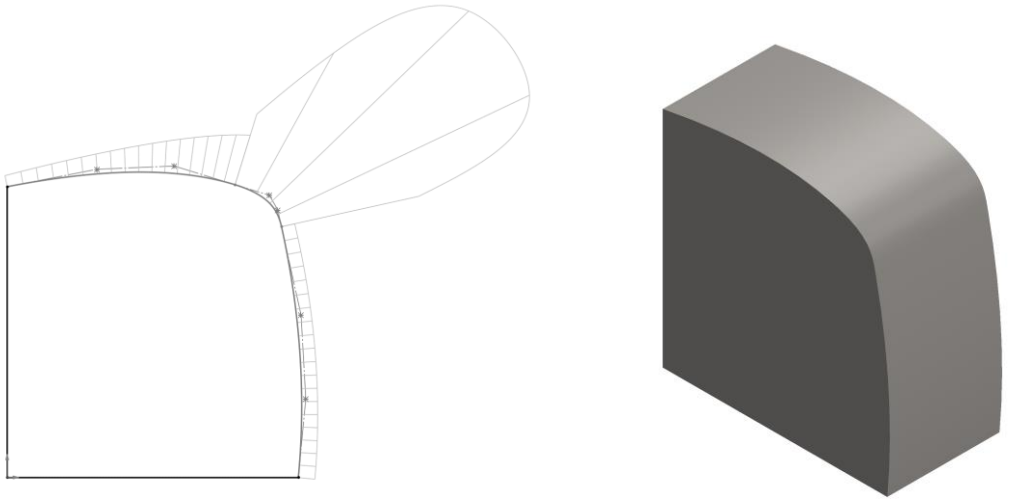


Slika 7.3. Zveznost G^0

Na sliki 7.3 je prikazan del, kjer se dve površini stikata in si delita skupni rob. Na levi strani slike se vidi ukrivljenost posameznih površin, kjer velikost normalnih vektorjev določa ukrivljenost površine. Sprememba ukrivljenosti na skupnem robu je trenutna, saj ravni odsek sploh ni ukrivljen. Jasno je, da prehod v matematičnem smislu ni zvezen (ni mogoče odvajati povezanih odsekov krivulje), čeprav se ta spoj imenuje G^0 zveznost.

G^1 zveznost

Če se površini stikata in imata na mestu spoja enako tangento, sta površini G^1 zvezni na prehodu površin in imamo tangenten prehod, če namesto skupnega roba uporabimo zaokrožitev (na primer ukaz fillet ali dodamo krivuljo v skicirki) ter na prehodu uporabimo tangentno omejitev.

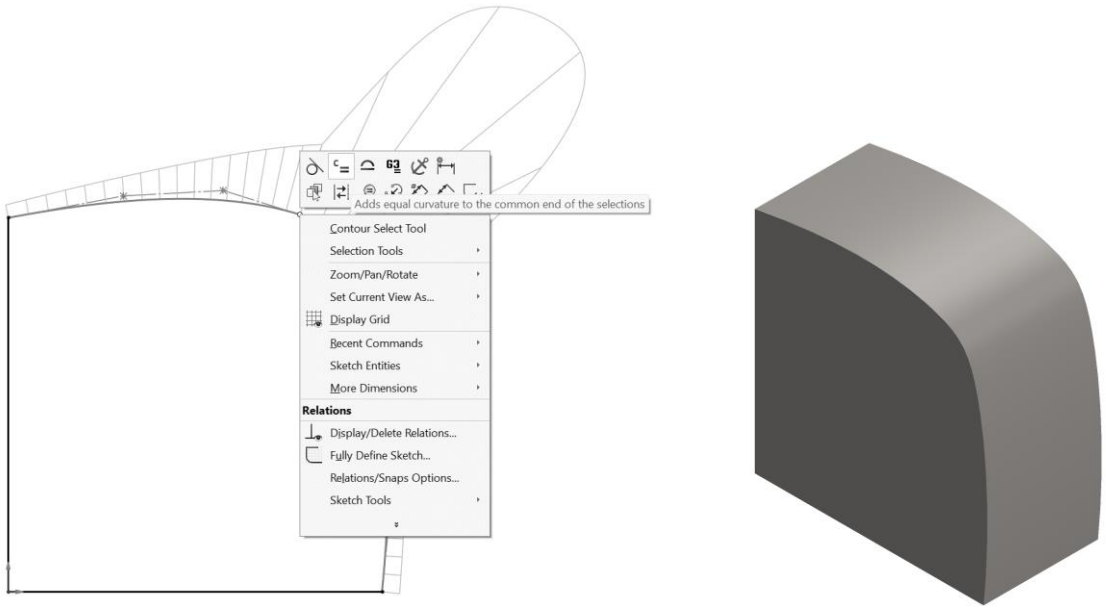


Slika 7.4. Zveznost G^1

Tangenten prehod pomeni, da sta tangenti enaki, kot je prikazano na sliki 7.4 med zgornjo površino in zaokrožitvijo ter med zaokrožitvijo in sprednjo površino. Na levi strani slike 7.4 se vidi, da je ukrivljenost zaokrožitve veliko večja kot na obeh sosednjih površinah. Čeprav imamo na prehodu G^1 zveznost, je sprememba ukrivljenosti trenutna in sprememba ukrivljenosti ni zvezna. To pomeni, da drugi odvod funkcij ni enak in imamo samo G^1 zveznost. Pogosto takšna zveznost ne zadostuje, če želimo gladek prehod med površinami. To je razvidno iz senčenega modela na sliki 7.4, kjer je prehod med površinami viden kot rob. Na izdelanih izdelkih se tak rob tudi čuti na otip, zato je zaželena večja zveznost prehoda med površinami.

G^2 zveznost

Na izdelkih, izdelanih z G^2 zveznostjo, prehoda med površinami ni več možno zaznati in površina je popolnoma gladka, saj izenačimo ukrivljenost na prehodu. V matematičnem smislu to pomeni, da je na prehodu drugi odvod funkcije enak.

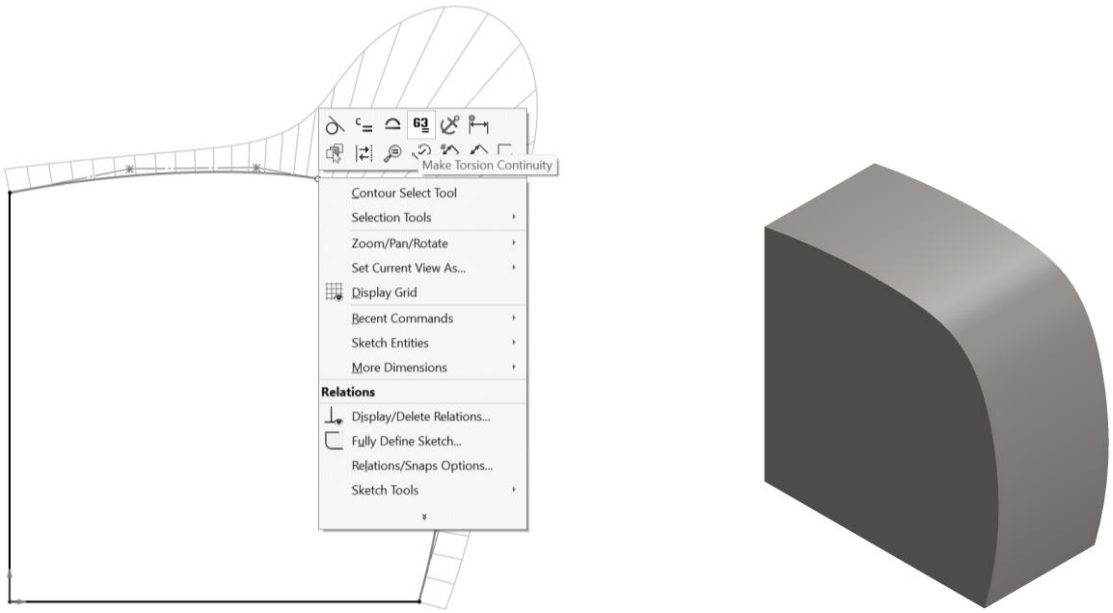


Slika 7.5. Zveznost G^2

Tak prehod izdelamo tako, da tangentni omejitvi dodamo še zahtevo, da je ukrivljenost na obeh odsekih enaka. V programu Solidworks v ta namen uporabimo ukaz »c=«, kot je razvidno iz leve strani slike 7.5, kjer se vidi, da je sprememba ukrivljenosti zdaj približno linearna. Ukrivljenost lahko izenačimo le tako, da se spremeni oblika obeh krivulj. Zato je nemogoče narediti prehod z G^2 zveznostjo med ravno in ukrivljeno površino. Zveznost G^2 se skoraj ne opazi na običajnih izdelkih, razen če gre za gladke odbojne površine (na primer karoserija avtomobila), ko se ti prehodi lahko opazijo. Zato je vrhunske izdelke treba izdelati s še večjo stopnjo zveznosti.

G^3 zveznost

Če želimo G^3 zveznost, je treba na prehodu med površinami zagotoviti takšno obliko, da je na prehodu še tretji odvod funkcije enak. To pomeni, da je prehod tangenten in še ukrivljenost ni zgolj izenačena, ampak se med površinama gladko spreminja. Na sliki 7.6 je prikazana G3 površina, na kateri ni mogoče opaziti prehoda med površinami. Na levi strani slike je prikazana ukrivljenost površin, s katerih je razviden gladek prehod med eno in drugo ukrivljenostjo. V programu Solidworks to naredimo z omejitvijo »G3≡«.

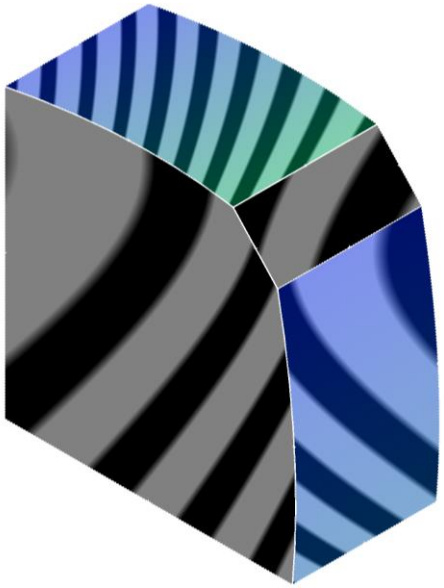


Slika 7.6. Zveznost G^3

Zveznost G^3 je najvišji cilj oblikovanja površin, zato so tako oblikovani vrhunski oblikovalski izdelki. V napravah Apple so celo zaslonske ikone oblikovane z zveznostjo G^3 in ne zgolj fizični izdelki. Oblikovanje površin s takšnimi prehodi seveda zahteva precej dodatnega dela pri oblikovanju površin, posebej ko gre za oblikovno kompleksne izdelke.

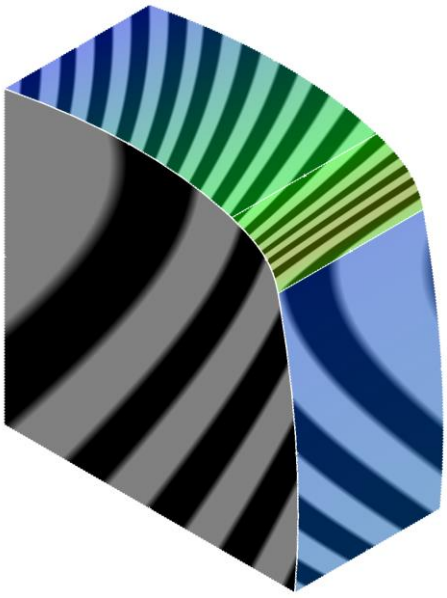
7.2 Orodja za spremljanje gladkosti površin

Spreminjanje gladkosti površin poteka tako, da spremenimo gladkost krivulj, ki določajo površino. Pri modeliranju je težko opazovati gladkost posameznih krivulj, zato uporabljamo orodja, ki nam omogočajo pregled gladkosti površin. Najpogosteje uporabljeno orodje se imenuje zebra, ker površino prikažemo s črno-belim vzorcem, ki je podoben programu zebre. Ta vzorec nam pokaže potek zveznosti po površini. Na sliki 7.7 so prikazane površine z zebra vzorcem in hkrati so površine pobarvane v odvisnosti od ukrivljenosti.



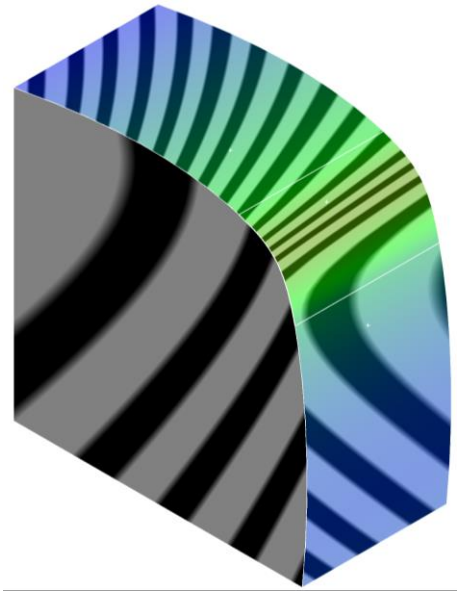
Slika 7.7. Zebra vzorec za zveznost G^0

Pri G^0 zveznosti je razvidno, da se zebra vzorec ne nadaljuje pri spoju ukrivljene in ravne površine, kar pomeni, da nimamo zveznosti v matematičnem pomenu besede. Zveznost G^0 pomeni le, da se površini stikata in seveda imamo izrazit rob na prehodu. Na sliki 7.8 imamo tangenten prehod in zveznost G^1 med površinami. Vzorec se nadaljuje z ene površine na drugo, čeprav je ukrivljenost različna, kar lahko sklepamo iz barve površin. Kljub tangentni zveznosti vzorec ne poteka gladko čez prehod, ampak je zlomljen ali zvit, kar je posebej opazno na prehodu na spodnjo površino. Tak prehod med površinami je opazen tudi na dejanskem izdelku.



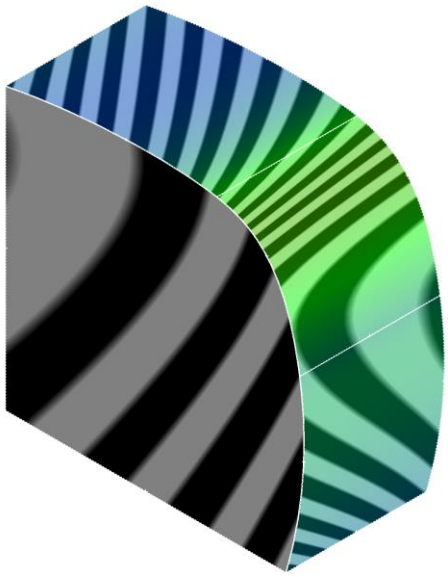
Slika 7.8. Zebra vzorec za zveznost G^1

Zveznost G^2 je prikazana na sliki 7.9 in proge zebra vzorca se zvezno nadaljujejo z ene površine na drugo brez zloma. Pri G^2 zveznosti je ukrivljenost na prehodu enaka, zato tam ni nenadne spremembe barve na sliki. Sprememba ukrivljenosti je še vedno prisotna in jo lahko ugotovimo iz barve, precej težko bi to določili samo na podlagi zebra vzorca.



Slika 7.9. Zebra vzorec za zveznost G^2

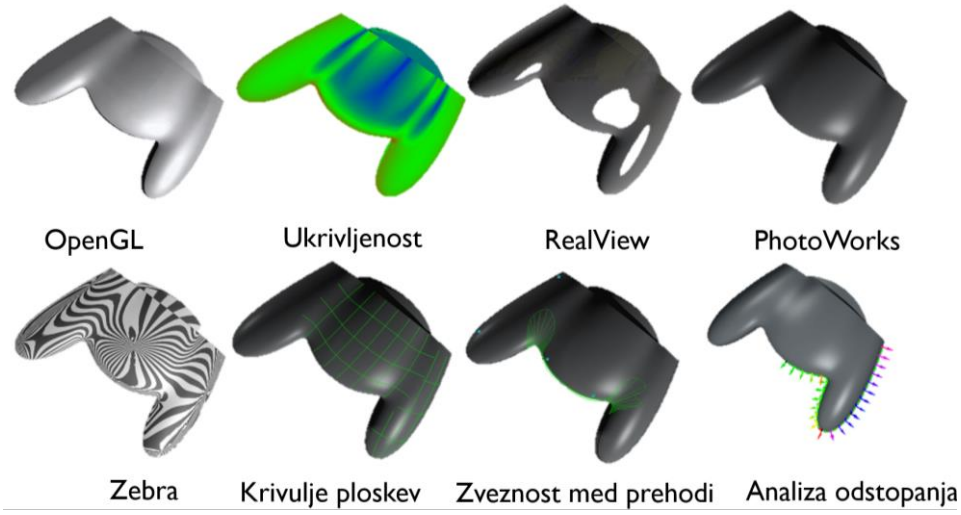
Šele ko imamo G^3 zveznost na sliki 7.10, se tudi barva površine bolj gladko preliva, kar kaže na bolj gladko spremembo ukrivljenosti površin.



Slika 7.10. Zebra vzorec za zveznost G^3

Samo z zebra vzorcem je razliko med G^2 in G^3 zveznostjo skoraj nemogoče opaziti. Tudi sicer G^2 zveznost zadostuje za večino izdelkov. Le tam, kjer je zahteva po izjemno oblikovani površini, se odločimo za G^3 zveznost. Zebra vzorec je izjemno uporaben, ko gre za popravljanje gladkosti površin, kjer so moteči G^0 ali G^1 prehodi.

Čeprav je zebra vzorec precej pogosto orodje v 3D modelirnikih, obstajajo tudi druga orodja, ki nam pomagajo najti nepravilnosti površin. V številnih programih je dovolj, če površino obarvamo z zrcalno površino, kjer je ob primernem renderiranju mogoče najti različne napake površin. Na sliki 7.11 je prikazanih nekaj orodij, ki jih uporabljajo različni programi. Običajno je v vsakem programu za 3D modeliranje v ta namen na voljo vsaj eno od teh orodij, včasih je lahko v naboru tudi več različnih orodij. Zelo pogosto imamo na voljo orodje za prikazovanje zebra vzorcev in način za prikazovanje ukrivljenosti površin. Pri urejanju krivulj pogosto najdemo grafe v obliki glavnika, s katerimi ugotavljamo velikost ukrivljenosti in zveznost na prehodih.

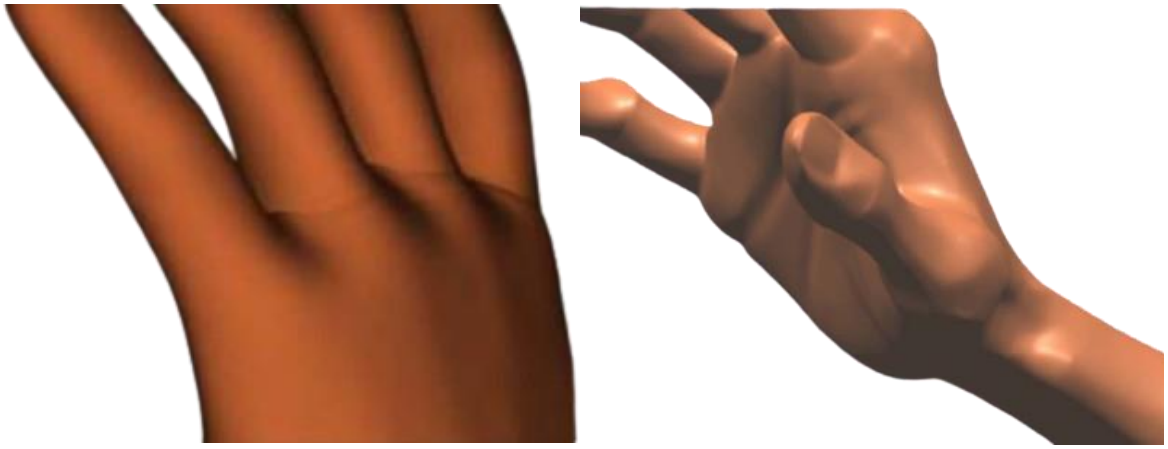


Slika 7.11. Orodja za ugotavljanje nepravilnosti na površinah

7.3 Deljenje površin

Deljenje površin (angl. Subdivision surface) [106] je postopek glajenja površin. Običajno deluje na poligonskih mrežah, kjer obstoječe poligone zamenjamo z drugimi tako, da dobimo bolj gladko površino. Izdelki, izdelani iz NURBS površin, so včasih preveč matematični in nerealistično zaobljeni. Najprej je postalo to problematično v risankah, kjer so želeli animirati bolj realistične like. V podjetju Pixar so izdelovali celovečerne risanke, kot so Toy Story, Cars itd. z uporabo NURBS. Prvi kratki film, narejen z deljenjem površin, je bil Geri's game [107], pozneje pa so bili risani filmi (na primer Toy Story 2) pretežno narejeni z uporabo deljenja površin. Na sliki 7.12 je prikazana razlika med modeli rok, izdelanimi z NURBS površinami na levi strani slike in z delitvijo površin na desni strani slike. Površine na desni so bolj realistične, a še vedno gladke. Z napredkom metod deljenja površin so se začele uporabljati tudi v 3D modelirnikih, ki temeljijo na NURBS površinah, saj je mogoče model, izdelan z deljenjem površin, pretvoriti v NURBS.

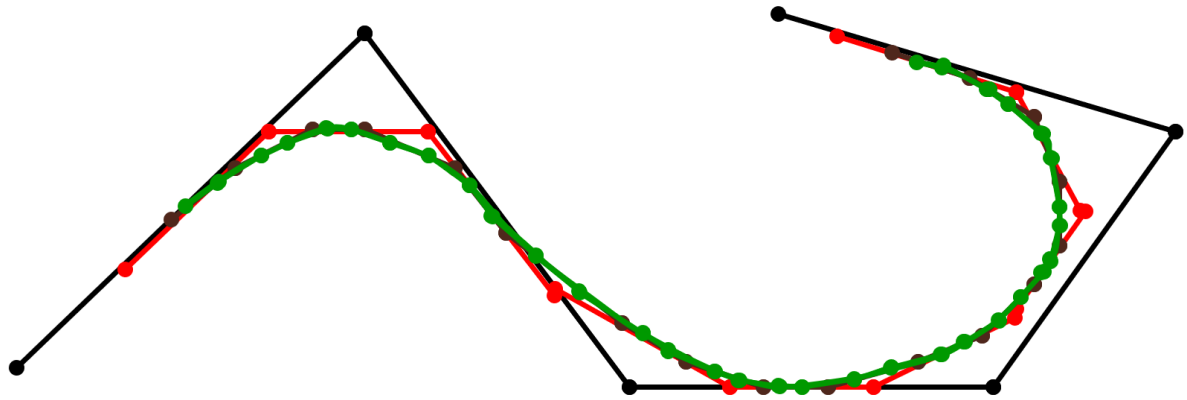
Woody (Toy Story) Geri (Geri's Game)



Slika 7.12. Primerjava med NURBS in deljenjem površin

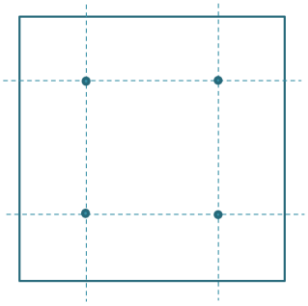
7.3.1 Doo-Sabin

Daniel Doo in Malcolm Sabin sta leta 1978 zasnovala algoritem Doo-Sabin [108] za deljenje površin, ki deluje v splošnem tako, da zgladi robove in vogale. Algoritem temelji na metodi glajenja krivulj, ki jo je najprej izdelal Georges de Rham leta 1947, ko za to še ni bilo prave uporabe. Leta 1974 je to ponovno oživil George Chaikin. Tudi Paul de Casteljau je že leta 1959 kreiral Bezierove krivulje tako, da je delil odseke na tretjinah odsekov. Bezierovo krivuljo lahko dobimo z deljenjem, če z daljicami povežemo štiri kontrolne točke Bezierove krivulje in na vsaki daljici kreiramo novo točko na tretjini odseka ter te odseke povežemo in nadaljujejo z novo delitvijo novih daljic. Z algoritmom Chaikina namesto tega daljice razdelimo na ¼ in ¾ odsekov in te nove točke povežemo v nov poligon, ki ga nato ponovno delimo, dokler ne dobimo gladke krivulje. Na sliki 7.13 je prikazana takšna delitev, kjer iz osnovnih ravnih odsekov dobimo gladko krivuljo. Krivulja je sicer krajša od osnovnih odsekov in je precej podobna B-spline krivulji. Krivulja odreže vse vogale in podobno nato deluje tudi deljenje površin z algoritmom Doo-Sabin.



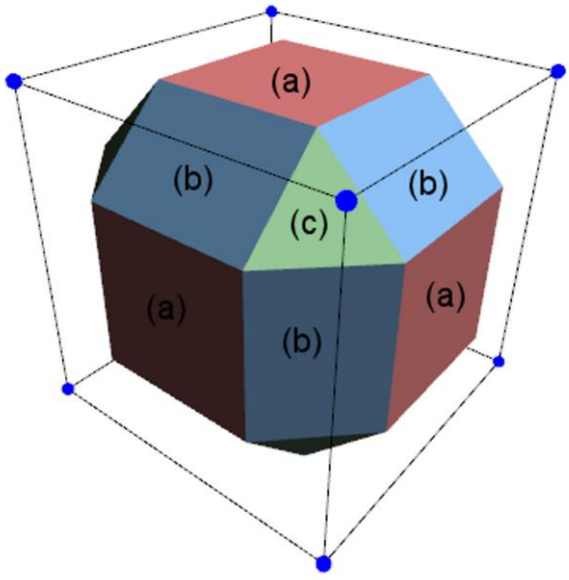
Slika 7.13. Delitev krivulje z algoritmom Chaikina

Algoritem Doo-Sabin deli vse poligone na ¼ in ¾ tako, kot je prikazano na sliki 7.14. Če povežemo te točke, dobimo samo manjši poligon. Pravo deljenje površin dobimo, ko razdelimo več povezanih poligonov in točke povežemo med seboj v novo poligonsko mrežo.



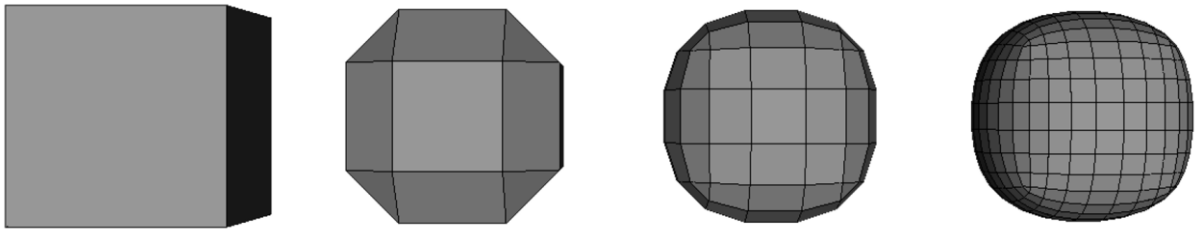
Slika 7.14. Delitev poligona z algoritmom Doo-Sabin

To lahko najlepše vidimo, če za deljenje izberemo kocko in jo delimo z algoritmom Doo-Sabin tako, kot je prikazano na sliki 7.15. Lepo se vidi, da z algoritmom porežemo vse vogale in robove in dobimo novo telo, ki je sicer manjše, vendar nekoliko manj oglato.



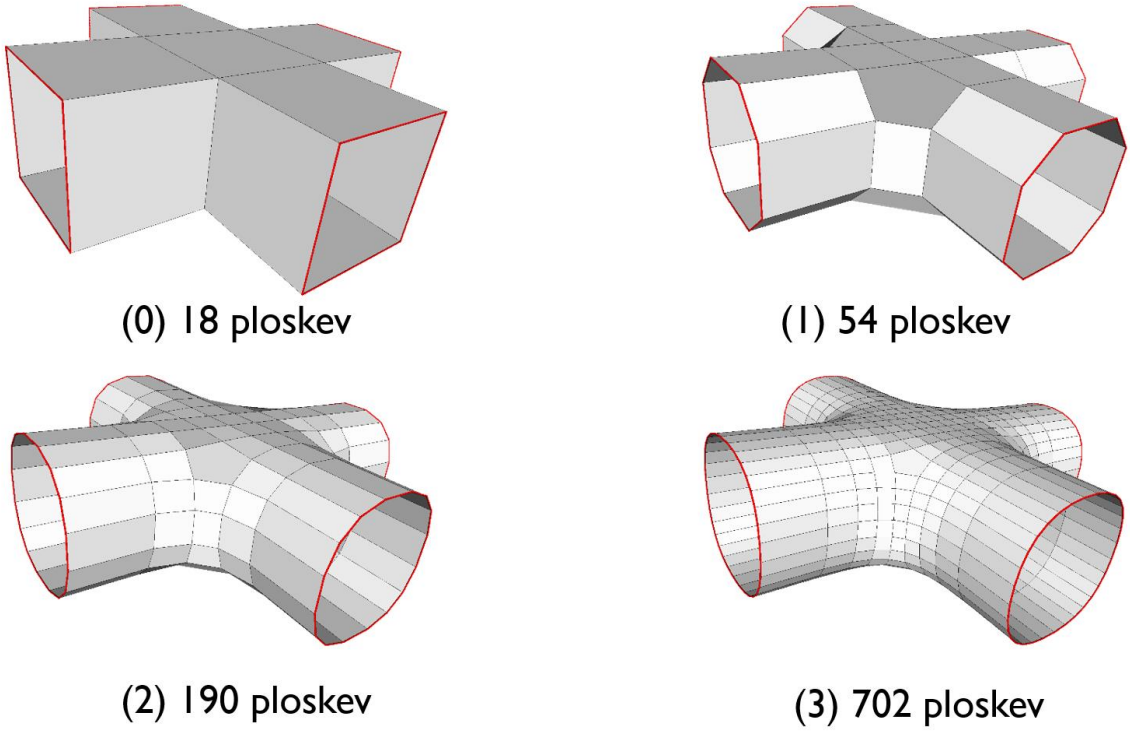
Slika 7.15. Delitev kocke z algoritmom Doo-Sabin

Glavna težava algoritma je to, da je poligon c na sliki 7.15 trikotnik in tako imamo v mreži poligone različnih oblik. Kljub temu je postal ta algoritem kmalu zelo popularen, saj lahko z nadaljnjimi delitvami hitro dobimo zelo gladko površino. To vidimo s slike 7.16, kjer v nekaj korakih dobimo sferično obliko iz kocke.



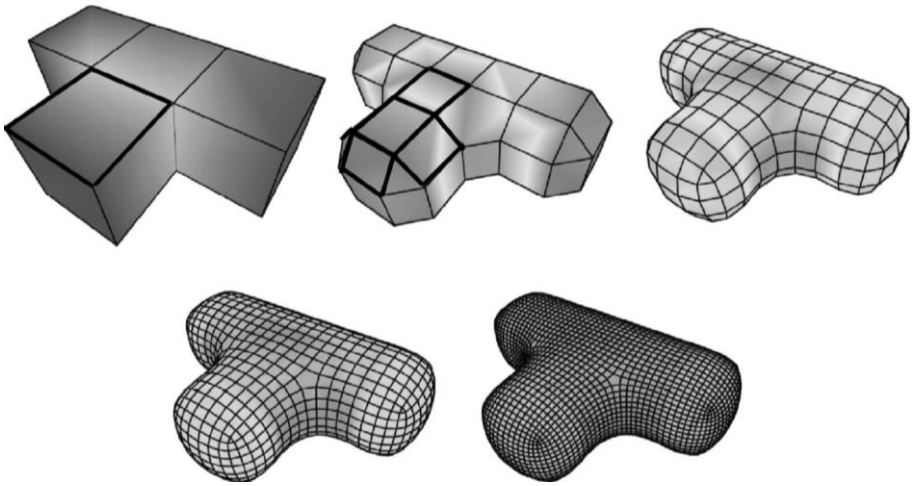
Slika 7.16. Delitev kocke v sferično obliko z algoritmom Doo-Sabin

Na tak način lahko dobimo za rezultat zvezne površine in Doo-Sabin omogoča kreiranje površin s tangentno G^1 zveznostjo površin. Nova geometrija odstopa od prvotne, zato je modeliranje takšnih površin narejeno tako, da najprej kreiramo geometrijo osnovne kletke, ki jo nato zgladimo do končne oblike. Na sliki 7.17 je prikazana transformacija osnovne kletke do končnega izdelka. Število poligonov se seveda pri končnem izdelku poveča, vendar je oblika precej bolj gladka.



Slika 7.17. Delitev cevi z algoritmom Doo-Sabin

Končni zglajen model je precej manjši od osnovne kletke, vendar lahko model vedno skaliramo, da dobimo želeno velikost. Za gradnjo večjega modela je treba razširiti kletko. Pri modeliranju z deljenjem površin nas običajno ne zanima, kako so zgrajeni poligoni, ampak je treba le kreirati, deliti, širiti in skalirati osnovno kletko, da dobimo želeni rezultat. Modeliranje je tako precej bolj podobno modeliranju skulpture iz gline kot matematičnim operacijam s krivuljami in površinami. Zato je tak način modeliranja izjemno popularen pri oblikovalcih in animatorjih, kjer natančne dimenzije izdelka niso tako pomembne.

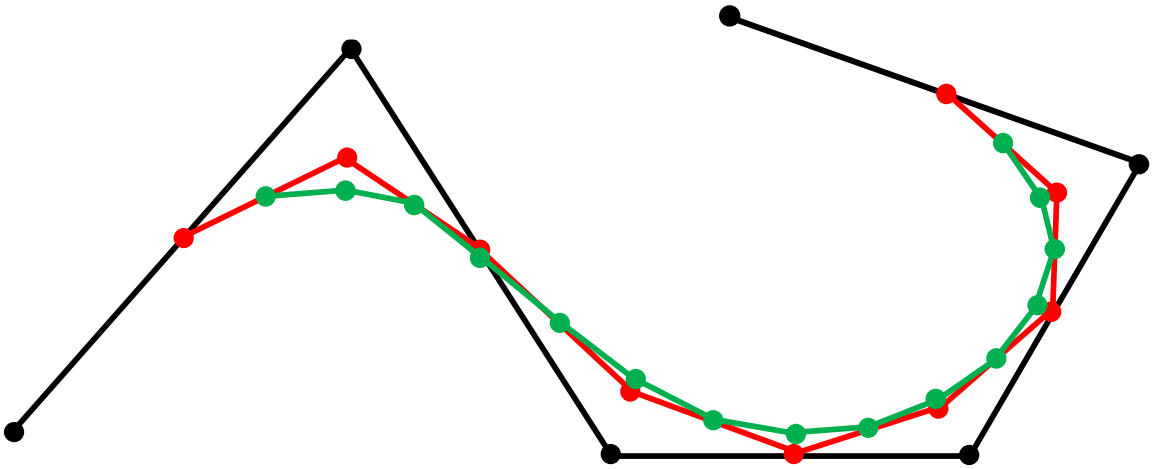


Slika 7.18. Referenčni model, glajen z algoritmom Doo-Sabin

Na sliki 7.18 je referenčni model, ki bo pozneje glajen tudi z drugimi algoritmi, s katerimi dobimo nekoliko drugačne rešitve za enak model.

7.3.2 Catmull-Clark

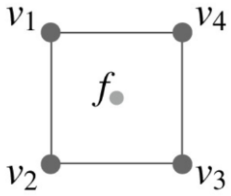
Edwin Catmull in Jim Clark sta leta 1978 razvila svoj Catmull-Clark [109] algoritem za deljenje površin, pri katerem dobimo po deljenju samo štirikotnike. Poleg tega je gladkost površin za stopnjo višja od Doo-Sabin in z algoritmom Catmull-Clark lahko izdelamo površine z zveznostjo G^2. Njun cilj je bil prvotno razviti algoritem za kreiranje B-spline krivulj iz poljubne geometrije. Na sliki 7.19 je prikazano glajenje krivulje z algoritmom Catmull-Clark. Pri tem algoritmu razdelimo vse odseke tako, da dodamo točke na sredini odseka. Nato dodamo še ogliščne točke (angl. Vertex points) v aritmetični sredini med obema polovicama odsekov in vmesno prvotno točko na koncu odseka. Nato povežemo nove točke in ponavljamo postopek, dokler ne dosežemo želene gladkosti.



Slika 7.19. Glajenje krivulje Catmull-Clark

Podobno deluje algoritem za poligone, kjer je dodatno definirana še točka na ploskvi. Tako določamo novo geometrijo na tak način, da določimo točko na ploskvi z enačbo 7.1 glede na sliko 7.20.

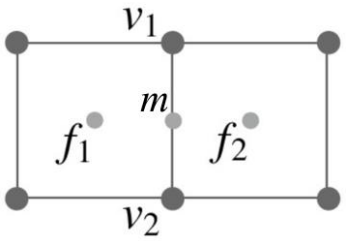
$$f = \frac{v_1+v_2+v_3+v_4}{4} \tag{7.1}$$



Slika 7.20. Točka na ploskvi

Točko na robu določimo z enačbo 7.2 glede na obstoječe točke na sliki 7.21.

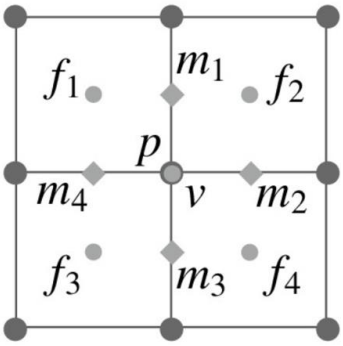
$$m = \frac{v_1+v_2+f_1+f_2}{4} \tag{7.2}$$



Slika 7.21. Točka na robu

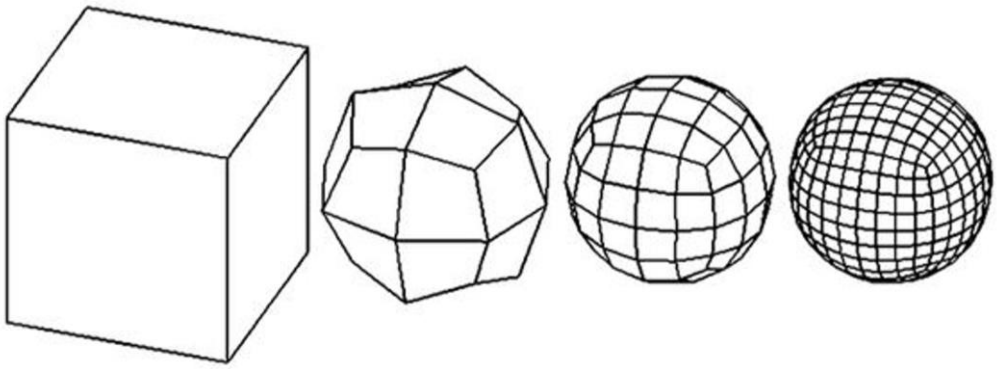
Nato lahko določimo še ogliščno točko z enačbo 7.3 glede na obstoječe točke na sliki 7.22.

$$v = \frac{f_1+f_2+f_3+f_4+2(m_1+m_2+m_3+m_4)+4p}{16} \tag{7.3}$$

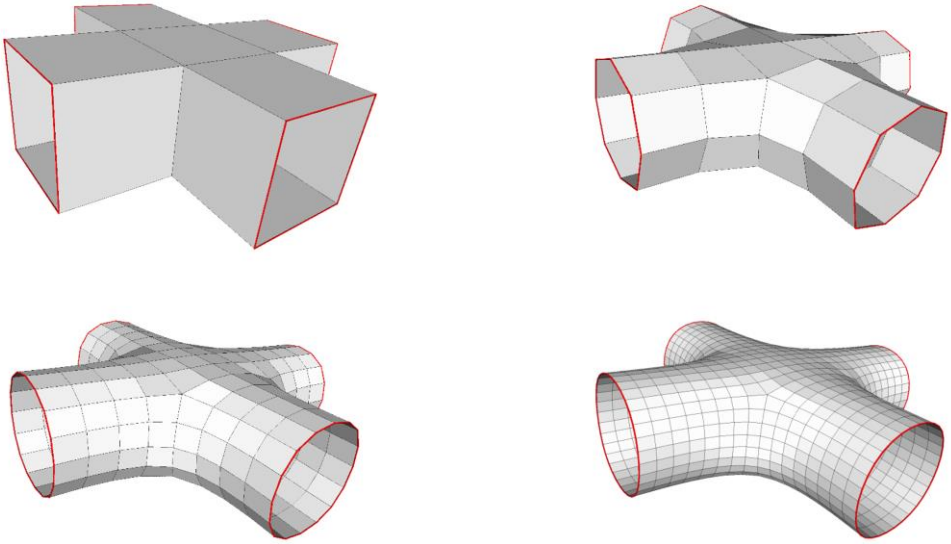


Slika 7.22. Ogliščna točka

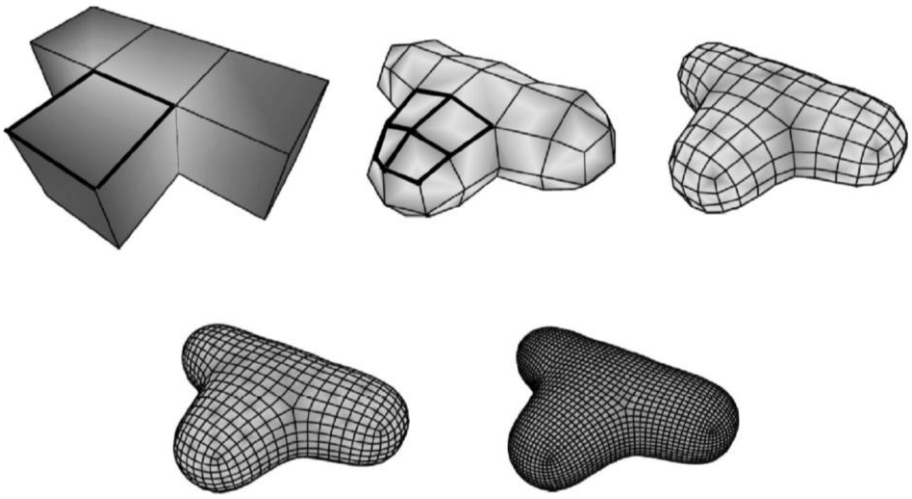
Tako dobimo delitev površin, ki je boljša od Doo-Sabin, ker nima trikotnih poligonov in vseeno zelo hitro dobimo gladko obliko. Na sliki 7.23 je prehod od kocke do sfere, kjer je nova površina v prvem koraku še precej oglata, vendar že v tretjem koraku dobimo precej gladko obliko. Tudi pri cevnem spoju na sliki 7.24 so v končni obliki vsi poligoni štirikotniki in veliko bolj enakomerni. Zaradi tega je algoritem Catmull-Clark največkrat uporabljen za glajenje površin.



Slika 7.23. Prehod od kocke do sfere z algoritmom Catmull-Clark



Slika 7.24. Cevni spoj z algoritmom Catmull-Clark



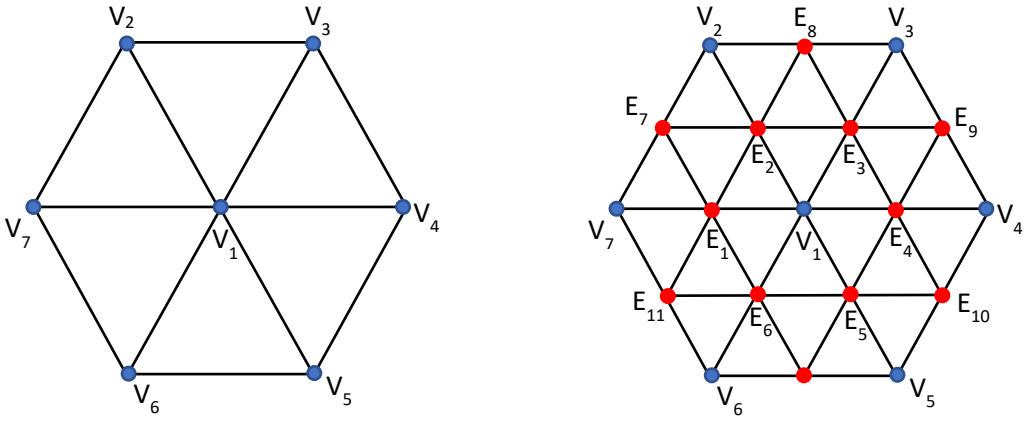
Slika 7.25. Referenčni model, glajen z algoritmom Catmull-Clark

Na sliki 7.24 je cevni spoj, kjer se vidi razlika v primerjavi z algoritmom Doo-Sabin predvsem na mestu spoja cevi. Pri modelu na sliki 7.25 je opazno, da so vsi poligoni pravokotniki in ne najdemo trikotnikov ali petkotnikov, kot jih vidimo na sliki 7.18.

7.3.3 Loop

Slabost Doo-Sabin glajenja površin je to, da pri glajenju površin poleg štirikotnikov dobimo različne mnogokotnike. Problem je v tem, da so na različnih mestih poligoni različnih oblik. S trikotniki ni nič narobe, ravno obratno, v računalniški grafiki je uporaba trikotnikov še pogostejša kot uporaba štirikotnikov. Zato je leta 1987 Charles Loop razvil algoritem za deljenje površin, imenovan Loop [110], ki deluje tako, da so vsi poligoni v poligonski mreži trikotniki.

Loop glajenje površin deluje tako, da obstoječo trikotniško mrežo razdelimo na manjše trikotnike. Vsak trikotnik razdelimo tako, da mu vrišemo nov trikotnik, ki ima oglišča na sredini stranic obstoječega trikotnika. Na tak način vsak trikotnik razdelimo na štiri manjše trikotnike tako, kot je prikazano na sliki 7.26.



Slika 7.26. Delitev trikotniške mreže

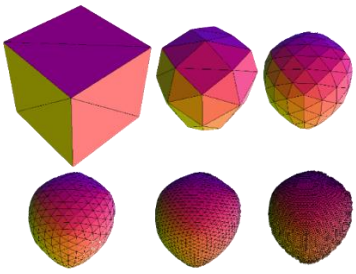
Pri delitvi ločimo med starimi oglišči, ki so na sliki 7.26 označena s črko V, in novimi oglišči, ki so označena s črko E. Glajenje izvedemo tako, da premaknemo vsa oglišča V tako, da dobimo bolj gladko površino. V ta namen spremenimo koordinate oglišča V z enačbo 7.4. V enačbi 7.4 E_o predstavlja vsa nova oglišča v okolici obstoječega oglišča V.

$$V_i = \frac{3}{8}\frac{\sum E_o}{n} + \frac{5}{8}V_i \tag{7.4}$$

Za primer oglišča V_1 lahko določimo novo pozicijo z enačbo 7.5, kjer je V_1 nova pozicija in V_{1o} stara pozicija oglišča V_1. Na tak način seveda obstoječa oglišča premaknemo na povprečno višino novo kreiranih oglišč.

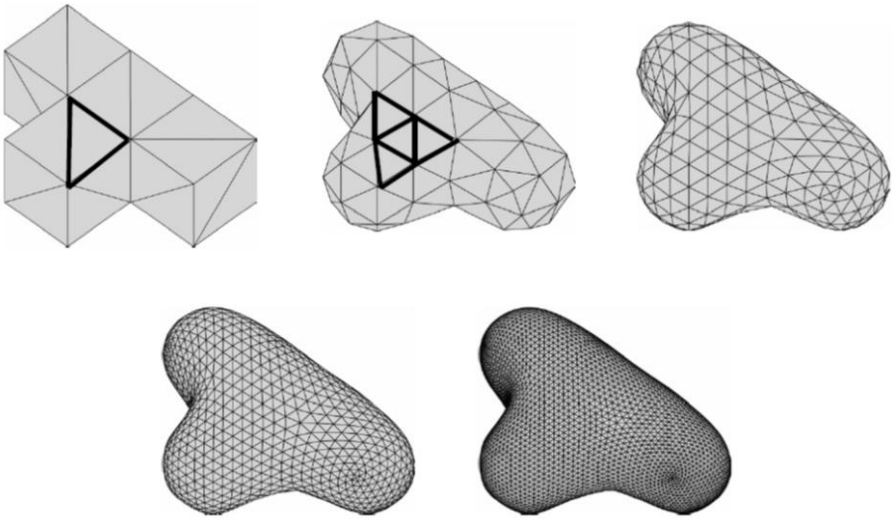
$$V_1 = \frac{3}{8}\frac{E_1+E_2+E_3+E_4+E_5+E_6}{6} + \frac{5}{8}V_{1o} \tag{7.5}$$

V primeru 3D površine postane ta z vsakim korakom bolj gladka. Na sliki 7.27 je prikazano glajenje Loop za primer kocke.



Slika 7.27. Glajenje kocke z metodo Loop

V nasprotju z drugimi metodami je pri Loopu mogoče zaznati vpliv trikotniške oblike, ki končno obliko naredi nekoliko bolj oglato. Kljub temu Loop glajenje še vedno temelji na aproksimacijskem pristopu in končni izdelek nima geometrijskih značilnosti osnovne geometrije. To je razvidno tudi s slike 7.28, kjer je nazorno pokazan način glajenja Loop in je očitno končni izdelek močno skrčen. Sliko 7.28 lahko primerjamo tudi s slikama 7.18 in 7.25. Vidimo, da je model, glajen z Loop glajenjem, sestavljen iz trikotnikov, ki so vsi približno enako veliki. Predstavitev s trikotniki je posebej primerna za uporabo s sodobnimi grafičnimi karticami, ki so optimirane za delovanje s trikotniki.

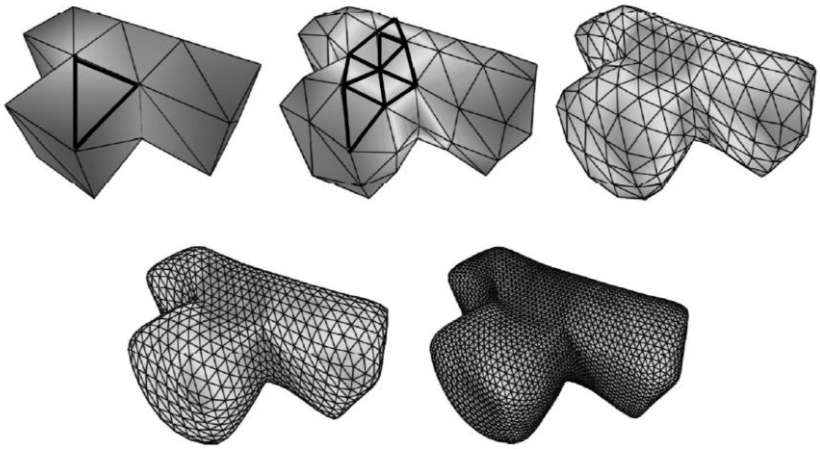


Slika 7.28. Loop glajenje

Aproksimacija glajenja je sprejemljiva, če nam začetni model služi le kot gradbeni oder za končni izdelek. Zato se je takšna tehnika modeliranja izdelkov močno uveljavila v podjetju Pixar, kjer je bil Loop zaposlen. Pozneje je postal tak način modeliranja vedno bolj razširjen tudi zunaj industrije risank in v zadnjem času je precej popularen tudi na področju računalniško podprtega konstruiranja.

7.3.4 Interpolacijske metode glajenja

Aproksimacijske metode glajenja so primerne za oblikovanje izdelka. V primeru rekonstrukcije izdelka na podlagi 3D skeniranja je pomembno, da ohranimo osnovne gabarite izdelka. Zato so zanimive metode, pri katerih model gladimo, vendar ohranimo originalno velikost.



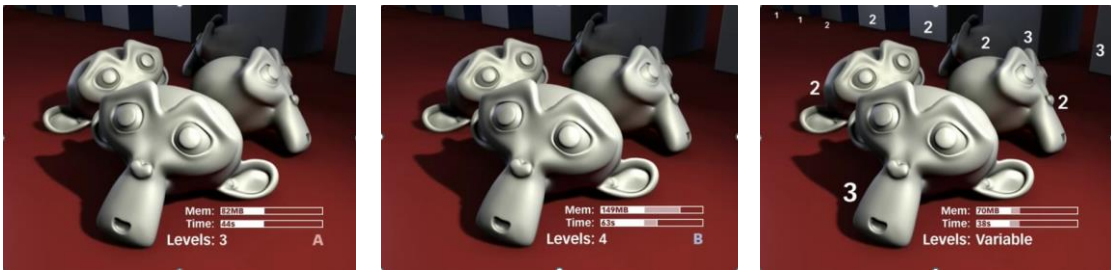
Slika 7.29. Butterfly glajenje

Prva takšna metoda Butterfly [111], ki so jo razvili v Pixarju leta 1990, temelji na glajenju Loop in zato deluje s trikotnimi poligoni. Na sliki 7.29 je prikazana referenčna geometrija z glajenjem Butterfly. Izkazalo se je, da je v primeru neenakomerne trikotniške mreže težko izdelati glajenje Butterfly, zato so bile razvite tudi interpolacijske sheme za poligonske mreže s pravokotniki. Takšni sta shema Modified Butterfly razvita leta 1996 [112] in interpolacijska metoda, ki jo je razvil Kobbelt leta 1996 [113].

7.3.5 Praktična uporaba glajenja

Za CAD je bila razvita posebna metoda glajenja, imenovana T-spline [104]. Metodo je razvil Thomas W. Sederberg leta 2003 in kmalu je postala precej popularna posebej zato, ker je bila mogoča pretvorba med T-spline in NURBS predstavitvijo površin. Leta 2011 je podjetje, ki je tržilo T-spline, kupilo Autodesk in od takrat je metoda podlaga za proste površine v programski opremi Autodesk. S predstavitvijo T-spline lahko površino opišemo s precej manjšim številom kontrolnih točk, kot z NURBS. Tako kot vse metode za glajenje površin je T-spline glajenje veliko bolj primerno kot modeliranje z NURBS površinami, ko gre za naravne modele. Ker so sodobni izdelki oblikovani s kompleksnimi površinami, pogosto presegajo zmogljivosti modeliranja z NURBS parametričnimi površinami. V drugih programih, kjer so uporabljali T-spline (npr. Rhinoceros), so morali plačati licenčnino Autodesku. To predstavlja težavo, zato danes večina CAD programov uporablja izvedbo glajenja Catmull-Clark, imenovano Sub-D. Način modeliranja s prostimi površinami bo opisan v poglavju 9.

Glajenje površin je izredno pomembno tudi za aplikacije, ki aktivno renderirajo sceno v živo, v odvisnosti od gledišča. Takšne aplikacije so računalniške igre, ki postajajo v zadnjem času izredno realistične. Kljub veliki zmogljivosti računalnikov ni smiselno pretirano natančno modelirati celotne scene. Objekte, ki so v ospredju, je smiselno izrisati z veliko natančnostjo, objekti v daljavi so lahko izdelani manj natančno z večjimi poligoni. Značilnost glajenja površin je tudi v tem, da lahko lokalno gladimo površine. Zato rečemo, da imamo na različnih delih površin različne stopnje glajenja. Na sliki 7.30 so prikazane različne stopnje glajenja scene. Na levih dveh slikah A in B je stopnja glajenja enotna za celotno sceno. Vidi se tudi, da ni bistvene razlike v kakovosti površin med stopnjo 3 in 4, vendar glajenje z višjo stopnjo zahteva več pomnilnika in daljše čase renderiranja, saj imamo na površini več poligonov. Najbolj ugodno je glajenje s spremenljivo stopnjo glajenja, prikazano na desni strani slike 7.30, kjer so objekti bližje gledišču izdelani s kakovostjo 3, bolj oddaljeni pa z 2 ali 1. Takšno glajenje je najhitrejše in zahteva najmanj pomnilnika, čeprav ne vidimo posebnih razlik v kakovosti slike.



Slika 7.30. Različne stopnje glajenja [116]

Glajenje površin je del grafičnih knjižnic OpenGL in DirectX že od leta 2008 in spreminjanje scene z različnimi stopnjami glajenja je izvedeno na ravni strojne opreme. To pomeni, da je takšno glajenje zelo hitro in se izvaja v živo med delovanjem grafične aplikacije.

8 Generativne površine

V tem poglavju bodo opisani glavni načini generativnega modeliranja površin. Generativno v tem primeru pomeni, da uporabimo poseben ukaz, ki generira površino. Pri tem bodo opisani tudi vsi podporni elementi, ki jih moramo uporabiti pri takšnem načinu modeliranja. Generativni način modeliranja površin je običajno postopkoven in moramo najprej modelirati osnovno geometrijo, ki jo nato uporabimo v kompleksnem 3D ukazu. V nasprotju s prostim modeliranjem, ki bo opisano v naslednjem poglavju, je treba ukaz načrtovati vnaprej. Pogosto je treba dobro razmisliti, kako bomo izdelek modelirali. Na primer, če je treba modelirati simetričen izdelek, modeliramo samo polovico in nato naredimo preslikavo. Zato je koristno, da pred začetkom modeliranja naredimo načrt modeliranja na podlagi skice izdelka.

8.1 Žične strukture

Običajno se modeliranja površin lotimo tako, da začnemo z modeliranjem žičnih struktur. Pri tem gre v osnovi za enodimenzionalne elemente, kot so točke linije in krivulje. Čeprav so konceptualno enodimenzionalne strukture, jih seveda lahko oblikujemo v prostoru.

Točka

Najpreprostejši element, kot je točka, lahko predstavimo s prostorskimi koordinatami. V CAD modelirnikih navadno začnemo risati skice, kar tudi vse točke postavi na ravnino skice. Zato imajo točke na 2D ravnini skice samo dve koordinati. V nekaterih CAD programih lahko vedno vnesemo 3D koordinate točke in tako naredimo točko neodvisno od ravnine skice. V drugih programih lahko vnašamo 3D koordinate šele, ko uporabimo 3D skico. Tako na primer je treba v Solidworksu izbrati 3D skicirko in z ukazom točka klikniti kamorkoli na površino zaslona. Nato lahko spremenimo lastnosti točke in vnesemo želene koordinate. Ustvarimo lahko poljuben oblak točk v prostoru. Seveda so na voljo tudi drugi načini kreiranja točk. Točko lahko ustvarimo s povezavo z drugimi strukturami modelirnika. Točko v prostoru lahko dobimo na naslednje načine:

- Vnos koordinat točke
- Točka na liniji
- Točka na krivulji
- Točka na oddaljenosti od krivulje
- Točka na presečišču linij ali krivulj
- Točka na projekciji krivulje
- Točka na ravnini ali na oddaljenosti od ravnine
- Točka na površini ali oddaljenosti od površine
- Točka v središču kroga
- Točka na tangenti kroga

Verjetno lahko najdemo še vsaj nekaj dodatnih načinov za kreiranje točk. Včasih so te točke skrite in uporabljene v interne namene modelirnika, vendar jih lahko pogosto aktiviramo in jih tako naredimo dostopne za uporabnika.

Linija

Linije (daljice ali premice) so naslednji element, ki ga uporabljamo za generiranje žične strukture. Običajno so linije prvi element, s katerim se lotimo ustvarjanja skice na skicirni ravnini. Seveda je mogoče ustvariti tudi prostorsko linijo tako, da povežemo dve točki v prostoru. V nekaterih programih, kot je na primer Solidworks, je treba v ta namen uporabiti 3D skicirko.

Linija je lahko daljica, če definiramo začetek in konec, medtem ko neskončno linijo kreiramo samo v eni točki in določimo le smer premice. Ni nujno, da za kreiranje linije uporabimo točke neposredno, ampak jo lahko naredimo tudi z uporabo drugih gradnikov. Linija je lahko gradnik modela ali samo konstrukcijski element, ki služi za konstrukcijo preostalih elementov, medtem ko se njena geometrija ne uporablja direktno za model. Takšni konstrukcijski elementi so tudi vizualno drugačni od gradnikov, tako da so predstavljeni z drugo barvo, debelino ali tipom črte. Tako lahko linijo kreiramo na naslednje načine:

- Od točke do točke
- Vodoravno, navpično ali pod kotom skozi točko
- Srednjica
- Normalno na krivuljo
- Tangentno na krivuljo
- Normalno na površino
- Tangentno na površino

Tudi za linije velja, da zgoraj niso naštete vse variante, saj lahko kombiniramo različne načine kreiranja linij in tako kreativno ustvarjamo geometrijo modela. Linije lahko povezujemo v like z zaporednim kreiranjem linij. Ti liki so lahko odprti ali zaprti in pri zaprtih likih je zadnja točka zadnje linije enaka prvi točki prve linije. Nekateri liki so že oblikovani in tako lahko direktno kreiramo štirikotnike različnih oblik od pravokotnikov do paralelogramov z različnimi načini kreiranja.

Krožni loki

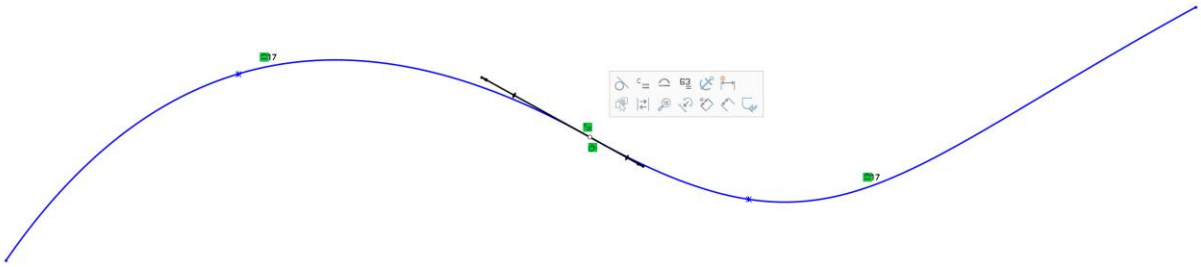
Običajen način za krožne loke ali krožnice poteka tako, da definiramo središče kroga in radij. Pogosto se zgodi, da je lažje določiti točke na krožnici kot središče kroga. Če na primer rekonstruiramo obstoječo geometrijo, je zelo težko določiti središče luknje na fizičnem izdelku. Zato je generiranje krožnic ali krožnih lokov s tremi točkami tudi zelo popularno. Naredimo lahko celotno krožnico ali samo del krožnice. Nekaj različnih načinov kreiranja krožnic in krožnih lokov je navedenih spodaj:

- Središče in radij
- Središče in točka
- Dve točki in radij
- Tri točke
- Tangenti in radij
- Tangenti in točka
- Tri tangente

Tako kot pri predhodnih elementih je možnih še več variant in tudi ta geometrija je lahko direktna podlaga za model ali podana samo v obliki konstrukcijskih oblik, ki služijo za nadaljnje generiranje žične strukture.

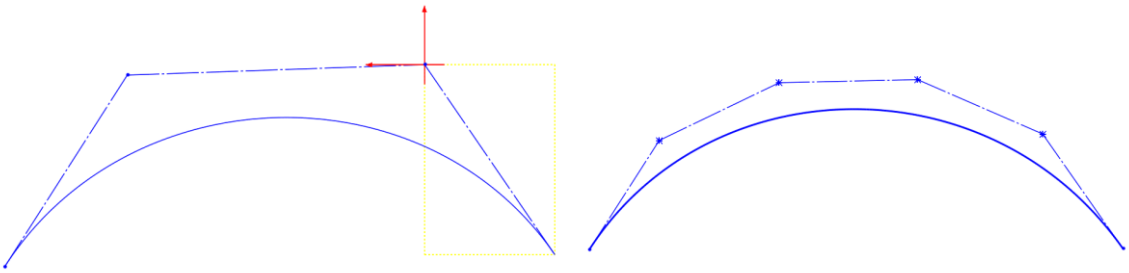
Krivulje

Tudi krivulje lahko narišemo v različnih oblikah in na različne načine. Najbolj običajen način je tak, kjer interpoliramo krivuljo skozi obstoječe točke, skozi katere je nato modelirana NURBS krivulja. Obliko krivulje lahko nato spreminjamo tako, da premikamo položaj točk ali spreminjamo smer tangente v posamezni točki krivulje. Različne odseke krivulje lahko povezujemo. Pri povezavi odsekov krivulj lahko definiramo zveznost na spoju, ki je lahko G^0, G^1, G^2, ali G^3. Na sliki 8.1 sta prikazana dva odseka NURBS krivulje, povezana z G^2 zveznostjo, kar je označeno s simbolom c= (enaka ukrivljenost).



Slika 8.1. Dva odseka krivulje NURBS

Seveda lahko modeliramo tudi določene parametrične krivulje, na primer Bezier ali B-spline z različnimi stopnjami polinoma. V tem primeru izberemo vrsto krivulje, stopnjo polinoma in nato določimo položaje kontrolnih točk. Izbiramo lahko tudi med prikazovanjem običajne ali racionalne oblike krivulje. Takšne krivulje lahko nato spreminjamo tako, da spreminjamo položaj kontrolnih točk ali stopnjo polinoma krivulje. Prav tako lahko urejamo načine povezave med odseki z različnimi stopnjami zveznosti te povezave. Na levi strani slike 8.2 je prikazano kreiranje kubične Bezierove krivulje, ki jo lahko spremenimo v poljubno polinomsko obliko. Na desni strani slike je ista krivulja s polinomom stopnje 5. Pri tem so samodejno kreirane dodatne kontrolne točke.



Slika 8.2. Bezierova krivulja z različnimi stopnjami polinoma

Možni so še drugi načini kreiranja parametričnih krivulj. Tako je mogoče definirati krivuljo s parametrično enačbo in jo nato uporabiti v CAD modelu.

Zaokroževanje in povezovanje krivulj

Krivulje lahko povezujemo med seboj z linijami, krožnimi loki ali krivuljami. Pri tem veljajo enaka pravila kot pri povezovanju odsekov krivulj. Na spoju lahko definiramo zveznost v obliki G^0, G^1, G^2, ali G^3. Pri tem je seveda pomembno, kakšni so odseki in kakšno spremembo oblike dopuščamo. Najtežje je modelirati krivulje z G^3 zveznostjo, saj to pomeni, da mora biti tretji odvod funkcij obeh odsekov enak. V tem primeru morata oba odseka imeti dovolj visoko stopnjo polinoma.

Robne krivulje

Za podporno geometrijo za generacijo površin lahko uporabimo tudi krivulje, ki predstavljajo rob obstoječe površine. V tem primeru je dovolj zgolj izbrati rob, ki je lahko linija, krožni lok ali krivulja. Poseben primer so presečne krivulje, ki nastanejo na preseku dveh obstoječih površin.

Projekcijske krivulje

Krivulje, ki smo jih generirali v prostoru, lahko projiciramo na obstoječo površino in tako dobimo novo krivuljo na površini. Prav tako lahko krivulje modeliramo tako, da izbiramo točke na površini. Rezultat je podoben projekcijski krivulji, vendar je na ta način krivulja na površini kreirana neposredno.

Posebne krivulje

Nekatere bolj zahtevne krivulje imajo posebno definicijo s precej podobnimi ukazi, kot veljajo za površine. Najbolj popularna posebna krivulja je vijačnica, kjer je običajno treba definirati presek, korak, smer vijačnice in spremembe koraka ali radija. Na sliki 8.3 je površina, ki jo naredimo z vijačnico in krožnim osnovnim profilom. Vijačnico lahko uporabimo tudi za izrezovanje materiala in tako lahko kreiramo različne oblike notranjih ali zunanjih navojev. To je smiselno le v primeru 3D tiska, saj se navoji običajno ne izrišejo v tehniški dokumentaciji. Namesto tega lahko v CAD programih označimo površine tako, da vsebujejo navoje, ki so potem v tehniški dokumentaciji pravilno narisani. Posebne krivulje lahko naredimo tudi skozi referenčne točke obstoječe geometrije, z uporabo koordinat x, y, z ali s sestavljanjem odsekov krivulj.

Slika 8.3. Površina, izdelana z vijačnico

Ravnine

Ravnine so sicer površine, vendar se v glavnem uporabljajo kot podporna geometrija. Šele ko naredimo ploskev kot površino, lahko to ploskev uporabimo kot geometrijo v prostoru. Ravnino lahko uporabljamo kot žično strukturo, ki je podlaga za veliko drugih geometrijskih ukazov. Pogosto namreč kreiramo podporno 2D geometrijo na ravninah. Običajno začnemo z osnovnimi koordinatnimi ravninami, kmalu to več ne zadošča in je treba kreirati nove ravnine na želenih mestih modela. Matematično lahko ravnino kreiramo z enačbo ravnine, vendar je najpogostejši način kreiranja ravnine odmik od obstoječe ravnine. Sicer ravnine kreiramo na naslednje načine:

- Enačba ravnine
- Na oddaljenosti od obstoječe ravnine
- Tri točke
- Dve liniji
- Linija in točka
- Ravninska krivulja
- Tangenta na površino v izbrani točki
- Normala na površino v izbrani točki
- Pod kotom od ravnine v določeni točki

Seveda je mogoče najti še kak način za kreiranje ravnine, ki jo potem uporabimo za 2D skico. Ravnine lahko uporabljamo tudi v druge načine modeliranja, saj lahko ravnino uporabimo tudi za neposredno modeliranje. Tako lahko ravnine uporabljamo v 3D ukazih za referenčne objekte ali z njimi izvajamo operacije, kot je na primer rezanje površin.

8.2 Generiranje 3D površin

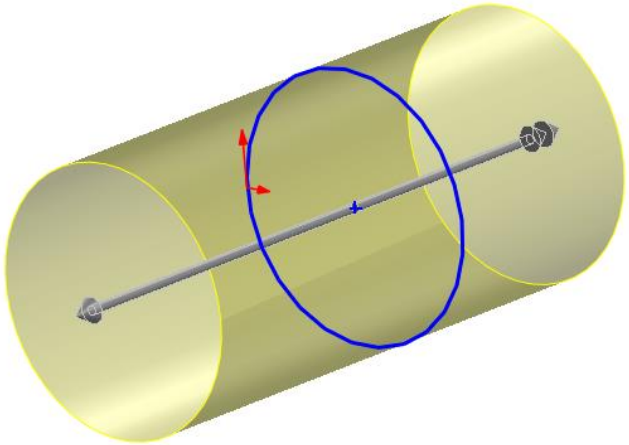
V veliko primerih poteka modeliranje površin na tak način, da najprej kreiramo podporno geometrijo in nato uporabimo 3D ukaz. V določenih primerih lahko tudi najprej začnemo s 3D ukazom in nato v dodatnih korakih ustvarimo podporno geometrijo.

8.2.1 Vlek

Vlek ali izvlek [117] je prvi ukaz, ki ga lahko uporabimo za kreiranje 3D površin. Ukaz ima lahko različna imena v različnih programih in najpogosteje naletimo na izraz »Extrude« in v določenih primerih »Boss«. Ta ukaz omogoča kreiranje površine tako, da poljubno obliko izvlečemo v določeni smeri v prostor. Oblika je običajno definirana v obliki 2D ali 3D skice v obliki žične strukture. Skica je sestavljena iz linij in krivulj in vseeno je, ali je tako izdelan profil zaprt ali odprt. Pri prostorskem modeliranju mora biti profil zaprt, in ko uporabimo vlek, dobimo polno »solid« telo. Pri površinskem modeliranju dobimo z vlekom površino, in če je profil zaprt, dobimo cevaste površine. Na sliki 8.4 sta prikazani površini generirani z ukazom vlek. Za površino na levi je bil uporabljen zaprt profil v obliki krožnice, za desno sliko pa odprt profil v obliki krivulje. V obeh primerih je rezultat 3D ukaza površina, ki jo lahko prikažemo na zaslonu, vendar je treba vedeti, da ta površina nima debeline in jo je fizično nemogoče izdelati.

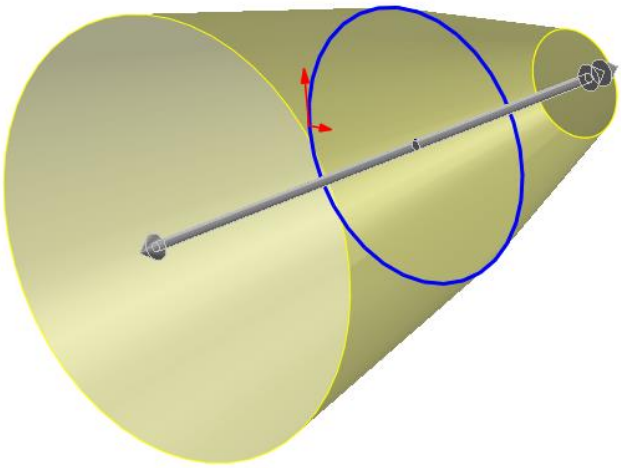
Slika 8.4. Površine, izdelane z izvlekom

Vlek lahko izvedemo v določeni smeri vektorja, ki je običajno normalna na površino profila. Vlek lahko izvedemo v eni smeri ali obeh smereh glede na izbran profil, kot je prikazano na sliki 8.5.



Slika 8.5. Vlek v dveh smereh

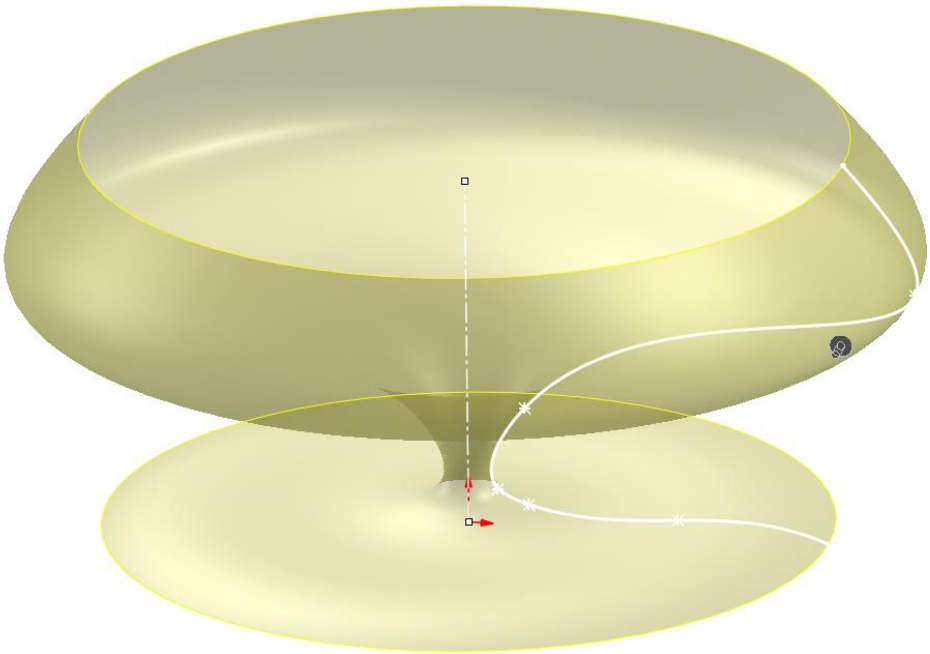
Pri vleku lahko površino kreiramo pod kotom z ukazom »Draft«, kjer se lahko površina oži ali širi. Z dodatno možnostjo lahko dodamo še površino na zaključku, kot nekakšno pokrivalo. Pri tem imamo še vedno samo površine, dokler ne zapremo celotnega telesa s površinami in telo pretvorimo v polno telo (angl. Solid). Na sliki 8.6 je spremenjen vlek s slike 8.5 tako, da imamo na sprednji strani razširitev in na zadnji strani zožitev. Pri tem je na zadnji strani dodana še površina, ki cev zapre z zadnje strani. Enako lahko seveda izvedemo vlek odprtega profila, kot je prikazano na desni strani slike 8.4.



Slika 8.6. Vlek v dveh smereh s spremembo kota površine

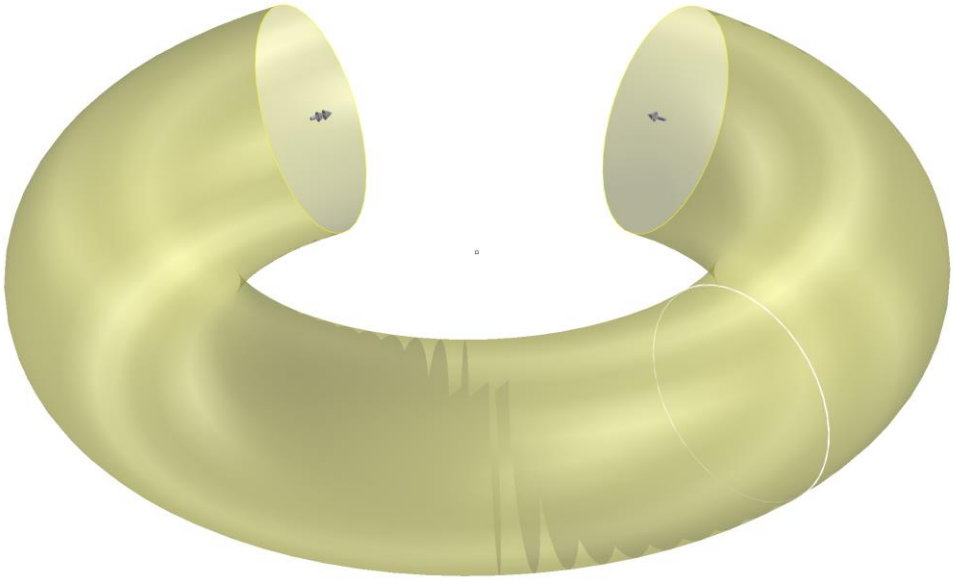
8.2.2 Vrtenje

Vrtenje (angl. Revolve) [118] je vleku podoben 3D ukaz, s tem da gre pri tem za vlek v polarnem koordinatnem sistemu. Zato je treba poleg osnovnega profila za ta ukaz določiti še os vrtenja, ki določa koordinatno izhodišče, okoli katerega z vrtenjem izvlečemo osnovni profil v površino. Na sliki 8.7 je prikazana površina, ki jo dobimo tako, da okoli navpične osi zavrtimo izbrano krivuljo.



Slika 8.7. Površina z vrtenjem osnovnega profila okoli osi

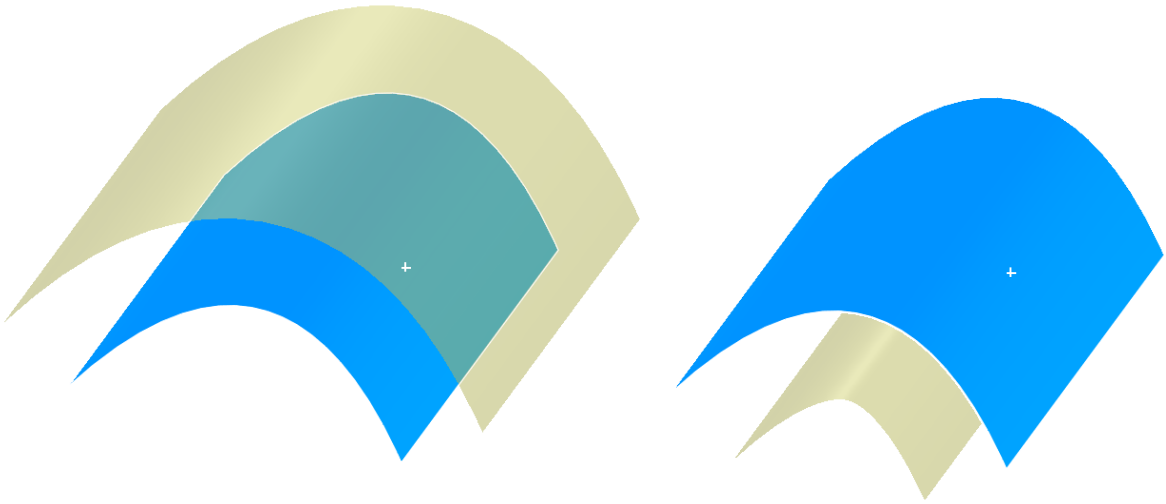
Vrtenje profila lahko obsega cel krog oziroma 360 stopinj ali izberemo določen kot vrtenja. Profil lahko vrtimo v eno ali v obe strani, če ne gre za vrtenje po celotnem obodu. Vrtimo lahko odprte profile tako, kot je prikazano na sliki 8.7, ali zaprte profile, kot je prikazano na sliki 8.8, kjer je krožnica zavrtena za različna kota v obe smeri. V obeh primerih je rezultat vrtenja površina, ki je brez debeline. Na sliki 8.8 je jasno, da gre za cev, vendar ima tudi model na sliki 8.7 luknjo na sredini. Z vrtenjem dobimo relativno preprosto oblikovno zelo kompleksne 3D oblike, zato je vrtenje precej popularen ukaz v modeliranju površin.



Slika 8.8. Površina z vrtenjem zaprtega profila okoli osi

8.2.3 Odmik

Odmik [119] je ukaz, s katerim generiramo odmično površino (angl. Offset Surface). Ukaz je relativno preprost, saj je treba le izbrati obstoječo površino in razdaljo, na kateri želimo novo površino. Na sliki 8.9 je prikazan ukaz Odmik, s katerim je na levi strani narejena nova površina nad obstoječo površino, na desni strani pa je nova površina pod obstoječo površino.

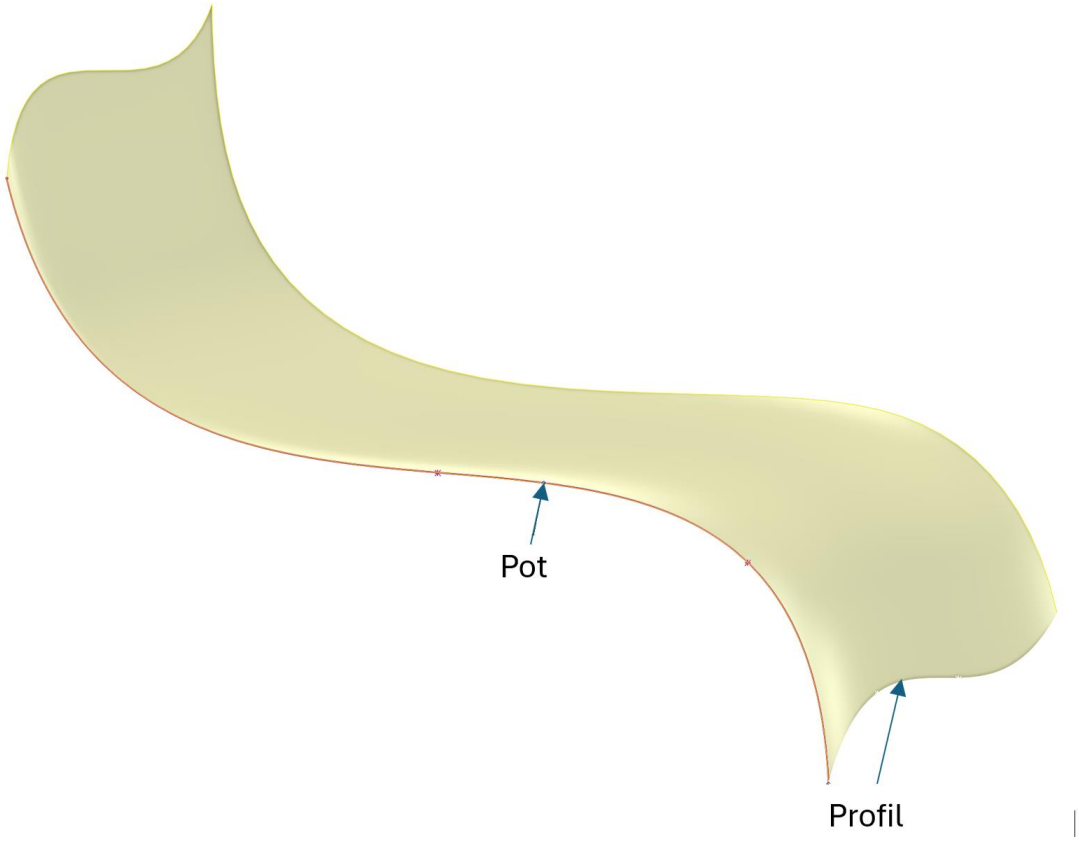


Slika 8.9. Površina z odmikom od obstoječe površine

Uporaba ukaza je relativno preprosta in ukaz je posebej uporaben, če želimo s površinami določiti stene izdelka. Posebej zanimiva je uporaba ukaza Odmik, ko ga uporabimo na sestavljenih površinah, saj namesto množice malih površin dobimo samo eno površino. Zato tak ukaz lahko uporabimo v namen zamenjave množice povezanih površin z eno samo površino. To naredimo tako, da izvedemo odmik sestavljene površine za določeno razdaljo ter nato še en odmik tako ustvarjene površine v nasprotni smeri. Potem dobimo novo površino na istem mestu, kot je sestavljena površina, ki je v nadaljnjih korakih skrita in je bila uporabljena kot podporna geometrija za končno izdelano površino.

8.2.4 Premik

Premik (angl. Sweep) [120] je najbolj razširjen ukaz v površinskem modeliranju. V splošnem je ukaz precej podoben vleku, vendar namesto vlečenja v določeni smeri izvedemo premik v smeri druge krivulje. Druga krivulja tako določa pot premika. Rezultati ukaza so veliko bolj raznoliki kot pri vleku, saj je površina ukrivljena zaradi definicije osnovnega profila in hkrati zaradi definicije vodilne krivulje. Na sliki 8.10 je prikazana površina, kjer krivuljo profila spodaj desno premaknemo po poti, ki jo opisuje krivulja na levi strani. Rezultat premika je površina na sliki. Pri tem se je treba zavedati, da v primeru, ko zamenjamo obe krivulji, običajno ne dobimo enake površine, saj imamo potem drugačen profil, ki ga vlečemo po drugi poti.

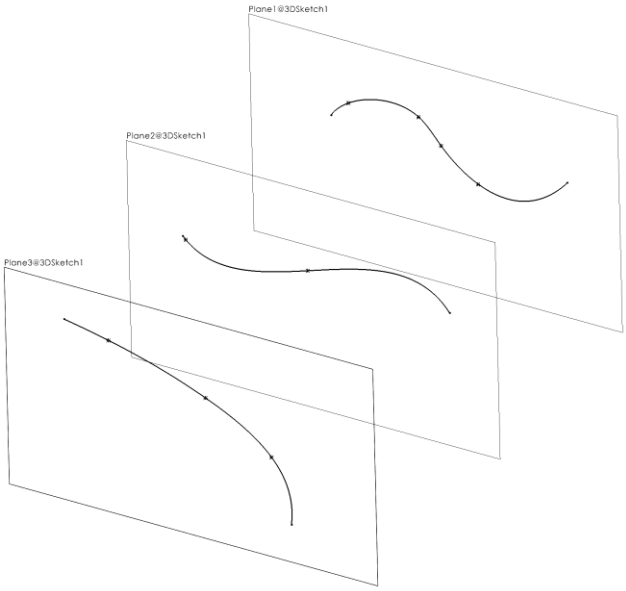


Slika 8.10. Površina s premikom profila po drugi krivulji

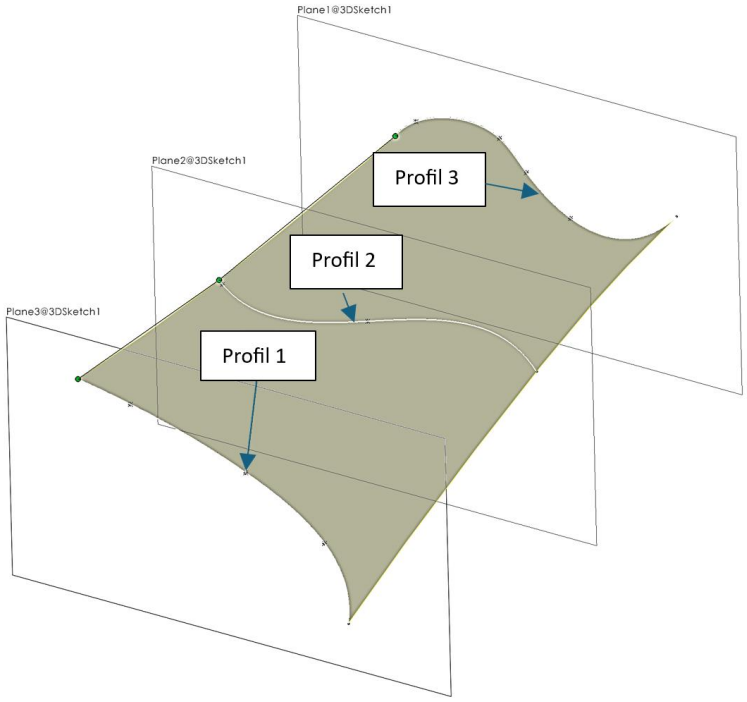
V osnovi je ukaz dokaj preprost in omogoča kreativno generiranje površin. Ukaz je mogoče uporabiti s številnimi možnostmi, ki omogočajo generiranje še bolj zahtevnih površin. Pri ukazu je mogoče določiti več vodilnih krivulj, določati različne orientacije in zasuke profila po poti ter določiti različne načine zaključka v odvisnosti od tangentosti ali poti površine. To omogoča različne načine povezovanja in glajenja posameznih odsekov površin. Zato pogosto v povezavi s tem ukazom uporabljamo tudi različna orodja za spremljanje zveznosti in ukrivljenosti površin.

8.2.5 Ovoj

Generiranje površine z ovojem [121] ima številna imena v angleščini (angl. Loft, Skin, Multisection Surface). Najbolj razširjeno je ime Loft, ki se uporablja tudi v programu Solidworks. Ukaz deluje tako, da s krivuljami določimo profile, ki jih nato »oblečemo« s površino. To je tudi razlog, da se ukaz včasih imenuje z angleško besedo koža in profili predstavljajo skelet modela. Najenostavneje uporabimo ukaz tako, da izdelamo skice profilov. Vsaka skica je na drugi ravnini ali je narisana s 3D profilom. Primer profilov na treh ravninah je prikazan na sliki 8.11, na sliki 8.12 so ti profili oblečeni s površino, ki je rezultat ukaza ovoj.



Slika 8.11. Odprti profili na različnih ravninah

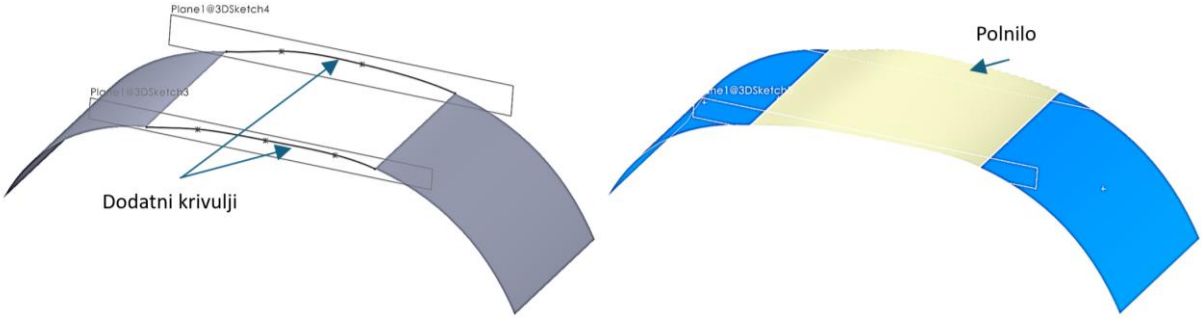


Slika 8.12. Ovojna (angl. Loft) površina

To je le najpreprostejša oblika ukaza, kjer je oblika površine določena le s profili. Seveda je mogoče oblikovati površino tudi med profili, kjer običajno definiramo potek površine z vodilnimi krivuljami. Uporaba ukaza je precej preprosta, zato je ukaz pogosto uporabljen pri modeliranju. Težavo povzročajo le pretirane spremembe oblik profilov, zaradi katerih je površino nemogoče izdelati, ker se lahko površina seka sama s sabo. Druga težava je zaključevanje površin v konične oblike, kjer so profili vedno manjši. Zato je treba paziti pri izdelavi profilov in zaključevanje površin na koncih izvesti s kakšnim drugim ukazom.

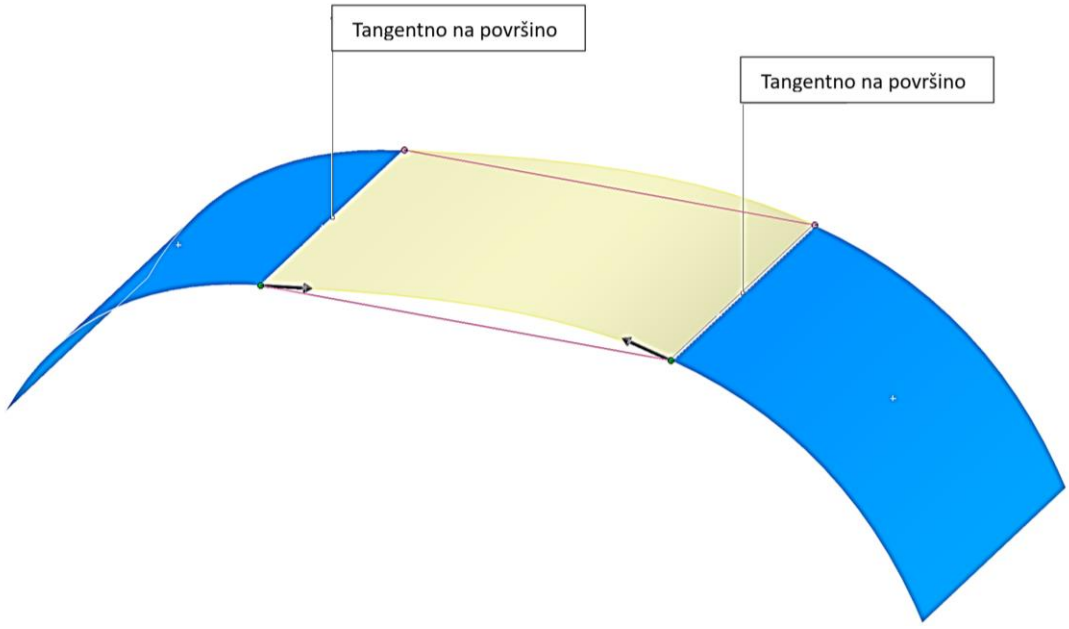
8.2.6 Polnilo in prehod

Ukaza Polnilo (angl. Fill) [122] in Prehod (angl. Blend, Boundary Surface) [123] uporabljamo za spajanje površin, ki se ne stikajo ali se le delno stikajo. Ukaz običajno zahteva izbor krivulj, ki določajo površino. Te krivulje morajo določati zanko, ki jo zapolnimo s površino. Te krivulje lahko narišemo na skici ali 3D skici in krivulje zapolnimo s polnilom. Tudi med obstoječimi površinami lahko narišemo povezovalne krivulje in potem obstoječe robove obstoječih površin in dodatne krivulje zapolnimo s površinami. Na sliki 8.13 sta narisani dodatni krivulji med dvema obstoječima površinama. Nato vse štiri krivulje definirajo novo površino, ki zapolni površino med obstoječima površinama.



Slika 8.13. Polnilo (angl. Fill) površina

V določenih primerih želimo le povezati dve ločeni površini in ne želimo kreirati dodatnih krivulj, ki bi predstavljale zaprto zanko. V tem primeru običajno govorimo o ukazu prehod oziroma Blend, kot je to prikazano na primeru na sliki 8.14, kjer naredimo površino med obstoječima modrima površinama tako, da izberemo le končna dva robova in način povezave, ki je na sliki 8.14 tangenten. Na prehodih lahko izenačimo tudi ukrivljenost in s puščicami na sliki določimo vpliv prehoda na povezovalno površino.



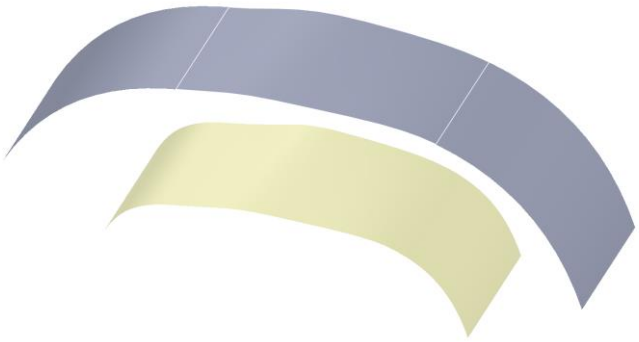
Slika 8.14. Prehod (angl. Blend) površina

V različnih programih imata oba ukaza lahko tudi nekoliko drugačna imena. V programu Solidworks je polnilo imenovano »Filled Surface«, za ukaz prehod se uporablja »Boundary Surface«. Prehod lahko naredimo tudi z ukazom Loft, če želimo prehod nekoliko natančneje definirati z vodilnimi krivuljami.

8.3 Sestavljanje in rezanje površin

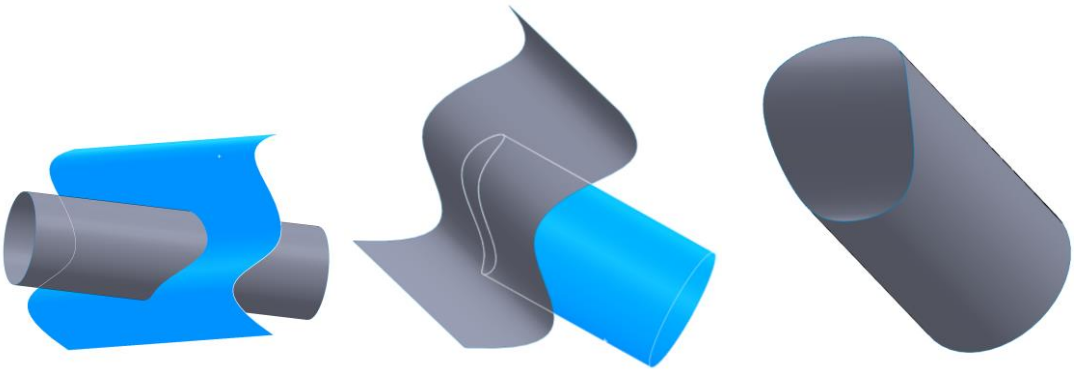
Kreiranje površin pogosto ne zadostuje za uspešno gradnjo modela. Velikokrat je treba del površine odrezati. Po drugi strani je včasih model izdelan iz številnih malih površin, ki jih je pogosto zelo koristno združiti. V ta namen uporabljamo ukaze za urejanje površin.

Orodje za sestavljanje površin ima v različnih programih različna imena (angl. Knit, Join) [124]. Rezultat sestavljanja površin je enotna površina, ki je enaka, kot so izvorne površine, le da zdaj predstavlja eno značilko v strukturi modela. V določenih primerih se lahko med posameznimi površinami, ki jih želimo sestaviti, pojavijo vrzeli. To je mogoče rešiti s povečanjem tolerance med površinami, vendar se lahko zgodi, da je vrzel prevelika. V takšnem primeru bo model težko zapreti in iz površin narediti polno (solid) telo. V takšnih primerih je treba vrzel zapolniti ali najprej del površine odrezati in nato ustvariti novo površino. V splošnem je ukaz za združevanje površin relativno preprost in večina težav je v programski opremi, ki ji ne uspe ustvariti matematične podlage za tako nastalo novo površino. Na sliki 8.15 uporabimo površine s slike 8.13 in jih sestavimo v enotno površino. V modelu ta sestava ni takoj očitna (vidi se le v drevesni strukturi) in šele, ko to sestavljeno površino uporabimo v novem ukazu, je očitno, da gre za novo enotno površino. Na sliki 8.15 smo uporabili ukaz odmik in na novo ustvarjeni površini ni več sledi, da je originalna površina sestavljena iz treh površin.



Slika 8.15. Sestavljanje (angl. Knit) in odmik površine

Orodje za rezanje površin je običajno označeno z imeni (angl. Trim, Cut) [125] in poteka tako, da uporabimo eno površino kot rezilo, s katerim prerežemo drugo površino. Na mestu reza nastane krivulja, ki razdeli površino na dva dela. V ukazu se lahko odločimo, kateri del površine ohranimo ali katerega izbrišemo. Na sliki 8.16 imamo dve površini, ki smo ju dobili z vlekom osnovnih profilov. V ukazu za rezanje najprej določimo modro valovito površino za rezilo in odstranimo zgornji del »cevi«. Nato v naslednjem ukazu izberemo cev za rezilo in obdržimo samo del valovite površine, ki zapira cev. Na desni strani slike 8.16 je nato končni izdelek površine cevi, ki jo zapira valovita površina.



Slika 8.16. Rezanje (angl. Trim) površin

Takšno generiranje površin z različnimi ukazi tudi s sestavljanjem in rezanjem površin določa končni izdelek. Pri tem je treba vedno paziti, da ne dobimo ostrih robov tam, kjer jih ne želimo, in da so površine sestavljene tako, da med njimi ni vrzeli. Na koncu modeliranja je želeno, da je končno telo zaprto in lahko s površin dobimo polno telo. Zato pri takšnem generativnem modeliranju vedno uporabljamo tudi orodja za preverjanje gladkosti površin, ki so predstavljena v poglavju 7.2. Pretvorba v polno telo poteka tako, da zapremo prostor s površinami. Včasih spremenimo površine v polno telo tako, da površinam določimo debelino in dobimo t. i. debele površine (angl. Thick Surface). Nato lahko modeliranje nadaljujemo tudi z ukazi prostorskega modeliranja. To v nekaterih CAD programih zahteva spremembo modula, s katerim delamo. V večini programov sta ta dva načina prostorskega in površinskega modeliranja precej prepletena in govorimo o hibridnem modeliranju. Hibridno modeliranje bo opisano v poglavju 10.

8.4 Modelirniki za generativne površine

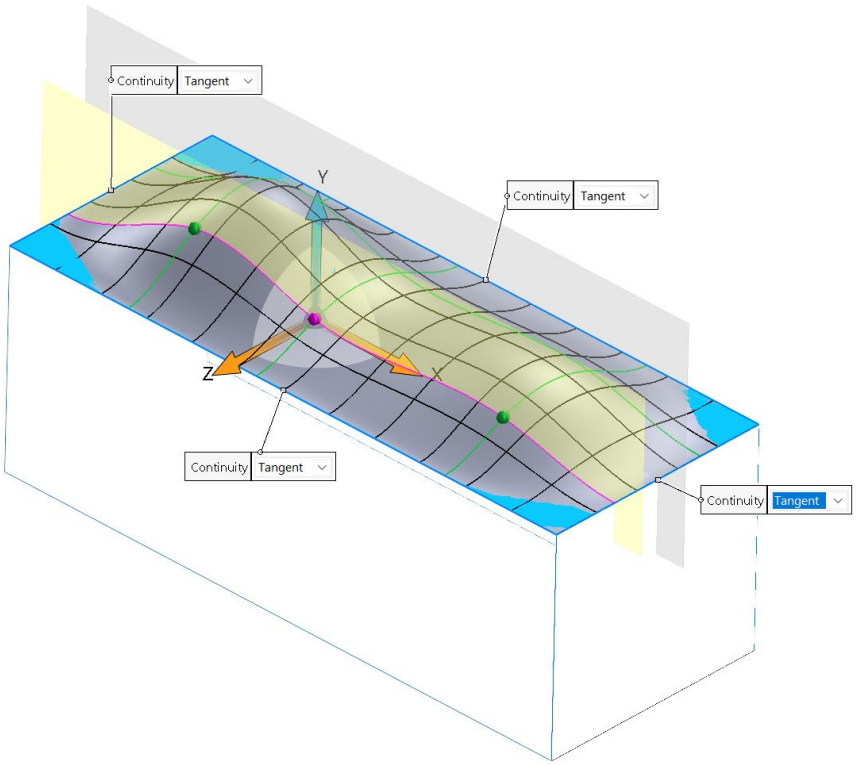
Večina sodobnih CAD modelirnikov vključuje tudi modeliranje generativnih površin. Dolgo je veljal program CATIA [126] proizvajalca Dassault Systemes za najzmogljivejši program za površinsko modeliranje. Danes je težko sklepati, kateri program je boljši od drugega, saj precej hitro programi prevzemajo različne funkcionalnosti. Program Catia je precej razširjen v evropski avtomobilski industriji, medtem ko v ZDA avtomobilska industrija (razen Ford) uporablja program NX [127]. NX je v lasti podjetja Siemens in zato si program utira pot tudi v evropsko avtomobilsko industrijo (na primer Daimler). Tudi drugi CAD modelirniki, kot sta Creo [128] proizvajalca PTC in Inventor [129] proizvajalca Autodesk, so vrhunska izbira za modeliranje generativnih površin. Precej so se uveljavili tudi številni manjši programi, ki so včasih cenejša alternativa istega proizvajalca za zgoraj omenjene programe ali dovolj zmogljiv poceni površinski modelirnik. Tako lahko med zelo uspešne programe umestimo Solidworks [130], Solidedge [131], Onshape [132], Fusion 360 [133] in Rhinoceros [134]. Poleg tega so na voljo še številni drugi programi, ki tukaj niso omenjeni. Večina programov je seveda plačljivih in le Onshape in Fusion 360 sta do neke mere uporabna brezplačno. Razlike v zmogljivosti programov seveda obstajajo in praviloma so dražji programi tudi bolj zmogljivi. Za začetnike je ta razlika skoraj neopazna in se kaže šele pri kreiranju geometrijsko zelo zapletenih površin, kjer manj zmogljivi modelirniki odpovejo ali kreirajo napačno površino. Vsi našteti modelirniki omogočajo uporabo ukazov za modeliranje generativnih površin, ki so opisani v tem poglavju.

9 Površine prostega sloga

Generativno modeliranje zahteva precejšen razmislek o obliki izdelka, zato je zelo uporabno, če je oblikovno izdelek že predhodno določen. To pomeni, da je bila oblikovana skica in nato generiramo površine izdelka glede na predlogo. V določenih drugih primerih je zaželeno, da direktno oblikujemo izdelek in površine prosto oblikujemo. Oblikovalci so pogosto poleg skice še fizično izdelali izdelek iz gline ali lesa, ker je bilo površino teh materialov lažje preoblikovati. Zato je želeno, da tudi virtualni izdelek oblikujemo kot glino in površine prosto preoblikujemo. V pomanjkanju programskih orodij so se pojavile tudi haptične naprave [135], ki omogočajo takšno virtualno modeliranje. Seveda je podobno mogoče upravljati tudi naprave virtualne resničnosti. V vsakem primeru mora takšno prosto modeliranje omogočati tudi programska oprema. Zato se v sodobnih CAD modelirnikih tudi vedno bolj uveljavljajo metode prostega oblikovanja površin.

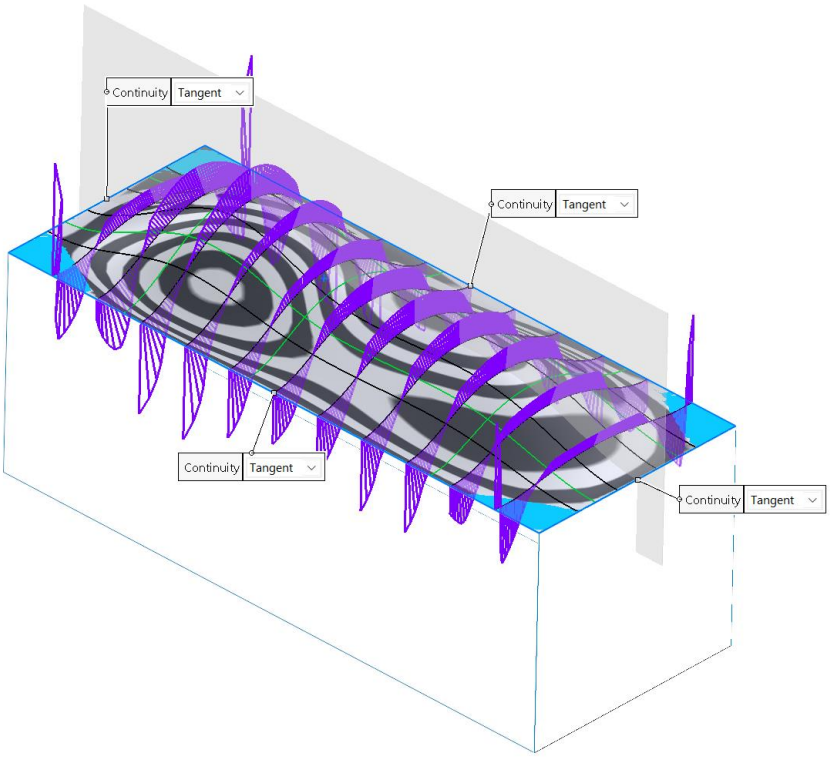
9.1 Prosto oblikovanje površin

Nadgradnja generativnega modeliranja površin je prosto oblikovanje površin NURBS [136], ki je danes že standardna funkcija vsakega CAD modelirnika. Ukaz ima različna imena, ki pogosto označujejo besedo prosto (angl. Freeform, Freestyle, Net Surface). Pri tem lahko površino na določenih mestih »zgrabimo« in jo oblikujemo po želji. Navadno je takšna površina diskretizirana z mrežo linij ali krivulj in razdeljena na posamezne poligone, s katerimi lahko oblikujemo površino. Na površini so pogosto tudi točke, ki omogočajo takšno manipulacijo. Na sliki 9.1 je prikazana prosto oblikovana površina v programu Solidworks, kjer izberemo obstoječo ravninsko površino in jo spremenimo v prosto površino. Takšni površini lahko določimo različno gostoto mreže, simetrijo v različnih smereh. Nato dodamo krivulje in točke, ki jih lahko izberemo in oblikujemo v različnih smereh. Površino lahko preoblikujemo v posameznih točkah ali v poligonih, ki jih tvorijo kreirane krivulje na površini. Na robovih površine lahko določimo način prehoda na sosednjo površino, ki je lahko samo stik ali tangenten ali pa se nadaljuje z enako ukrivljenostjo (G^0, G^1, G^2).



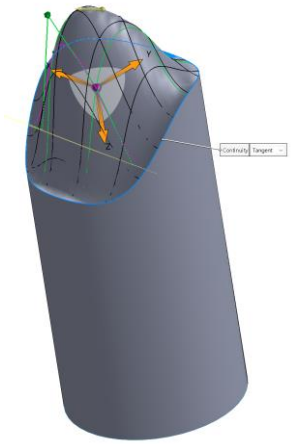
Slika 9.1. Proste površine (angl. Freeform) v Solidworksu

Na tako oblikovani površini lahko prikažemo tudi orodja za spremljanje kakovosti površin, da opazujemo zveznost ali ukrivljenost površin. Na sliki 9.2 je prikazano orodje zebra ter grafi za vektorsko spremljanje ukrivljenosti površin na prosti površini.



Slika 9.2. Kakovost površin na prosti površini (angl. Freeform) v Solidworksu

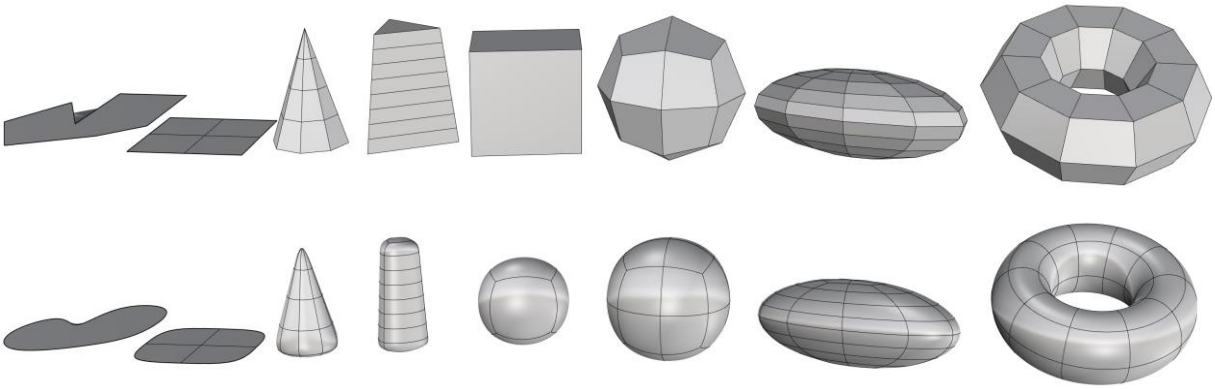
Orodja za urejanje prostih površin omogočajo spreminjanje oblike površine NURBS samo s premiki miške. Tako lahko površino oblikujemo kot glino, vendar model še vedno ostane zapisan s površinami NURBS in ga lahko uporabimo v drugih ukazih modelirnika. Po drugi strani smo omejeni na spreminjanje generativnih površin in zato takšno modeliranje ni v celoti prosto. Oblikovanje prostih površin ni omejeno zgolj na ploske površine, ampak lahko tako preoblikujemo vsako površino v modelirniku. Na sliki 9.3 je primer preoblikovanja površine, ki smo jo dobili z rezanjem površin na sliki 8.16.



Slika 9.3. Prosto oblikovanje poljubne površine v Solidworksu

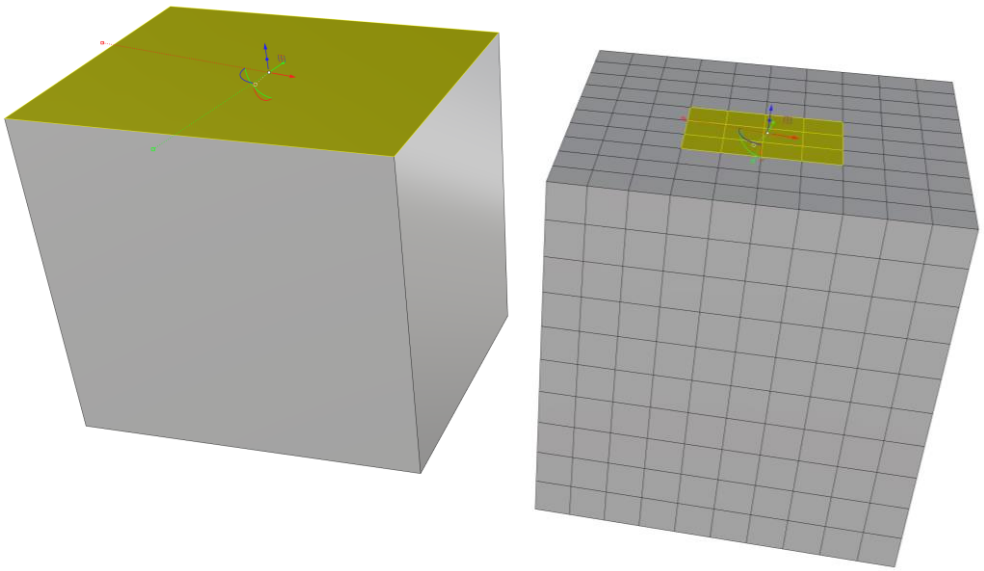
9.2 Modeliranje z deljenjem površin

Veliko modelirnikov v zadnjem času ponuja modeliranje z deljenjem površin (angl. Subdivision Surface) oziroma SubD [137]. Podjetje Autodesk je razvilo metodo, ki se imenuje T-splines, imenovano po značilnih zvezah krivulj v obliki črke T. V izogib licenciranju je večina preostalih ponudnikov programske opreme raje razvila svoj način za glajenje površin ali uporabljajo odprtokodno knjižnico podjetja Pixar, ki temelji na deljenju površin Catmull-Clark. Vsi takšni načini modeliranja so si precej podobni. Pri modeliranju najprej izberemo osnovno predstavitev za začetek modeliranja. Ta predstavitev je lahko kvader, valj, sfera, elipsoid, konus, torus itd. Samodejno se izvede glajenje te predstavitve in v vsakem koraku modeliranja imamo na voljo osnovno grobo predstavitev telesa in glajeno predstavitev telesa. Nekateri modelirniki hkrati prikazujejo obe predstavitvi, drugi omogočajo preklapljanje med obema predstavitvama. Na sliki 9.4 so prikazane osnovne predstavitve za modeliranje SubD. Prvi dve predstavitvi na levi strani sta ploskovni, preostale predstavitve so telesa. Na zgornji strani slike so grobe predstavitve teles, pod njimi so glajene predstavitve.



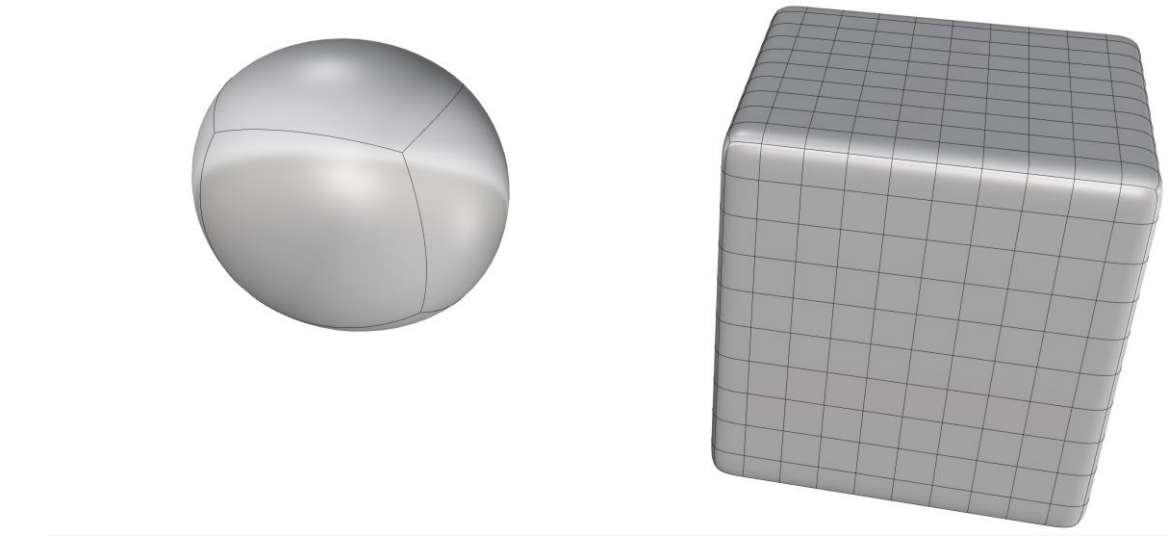
Slika 9.4. SubD osnovna telesa v programu Rhinoceros

Število poligonov osnovnih predstavitev je mogoče povečati, če želimo modelirati bolj detajlno. Na začetku je priporočljivo uporabiti manj poligonov in jih pozneje po potrebi namnožiti. Na sliki 9.5 sta dve osnovni predstavitvi z različno gostoto poligonov. Če želimo spreminjati zgornji del telesa na desni, je treba izbrati vse poligone zgornji strani, sicer spremenimo samo del te ploskve.



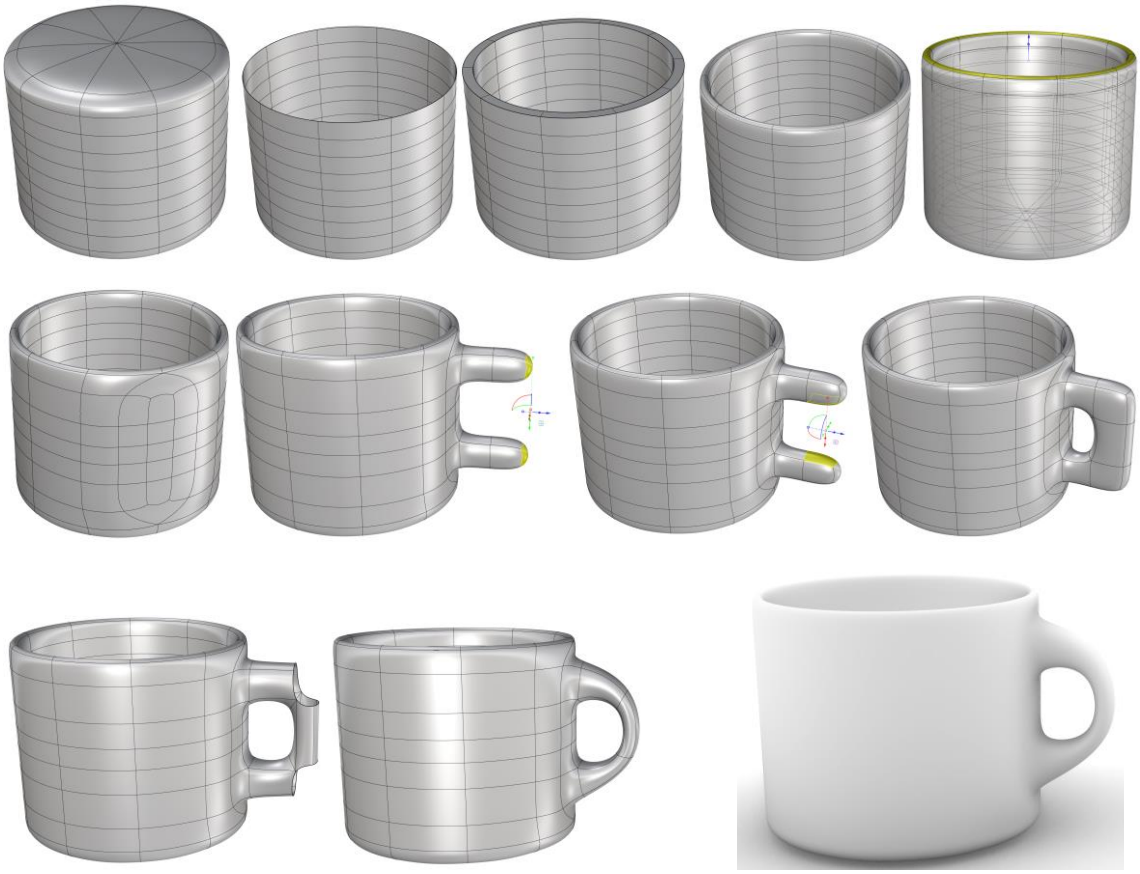
Slika 9.5. SubD gostejša mreža poligonov Rhinoceros

Gostota poligonov na osnovni predstavitvi seveda vpliva tudi na obliko glajenega telesa. Na sliki 9.6 je glajena predstavitev teles s slike 9.5. Če je na vsaki ploskvi kvadra samo en poligon, dobimo sferično predstavitev telesa, pri večjem številu poligonov je telo še vedno kvader, le robovi in oglišča so zaobljeni.



Slika 9.6. SubD glajena oblika teles s slike 9.5

Modeliranje poteka na tak način, da posamezne ploskve grobe ali glajene predstavitve preoblikujemo v želeno obliko.



Slika 9.7. SubD modeliranje preprostega izdelka

Na sliki 9.7 je prikazano modeliranje preprostega izdelka. Na začetku je bilo izbrano valjasto polno telo, kjer so bili v naslednjem koraku odstranjeni poligoni na zgornji strani. Rezultat je odprta površina valjaste predstavitve, ki jo v naslednjem koraku spet spremenimo v polno telo z uporabo ukaza odmične površine. Na zgornji površini dobimo ostre robove, ki jih pretvorimo v zaobljene robove in površino lahko poljubno povišamo. Za dodajanje ročaja na lončku dodamo nove robove na mreži, s tem da predhodno zbrišemo nekaj robov na pregosti mreži. Gostoto mreže lahko tako globalno in lokalno poljubno spreminjamo in brišemo ali dodajamo posamezne robove. V nadaljevanju izberemo dva poligona ter izvlečemo zgornji in spodnji del ročaja. Izvlek vmes prekinemo, da dobimo dva nivoja poligonov in na drugem nivoju izberemo spodnji in zgornji poligon, ki ju nato spojimo. Velikost nivojev lahko pozneje spreminjamo do želene debeline v različnih koordinatnih smereh. Nato na zgornjem in spodnjem vogalu ročaja preprosto izbrišemo poligone, da jih nato povežemo v zaobljeno obliko. Čeprav modeliranje poteka z manipulacijo robov in poligonov, je rezultat površinski model, ki ga zlahka tudi spremenimo v polno telo. Razlog je v tem, da je gladka predstavitev po definiciji vedno G^2 zvezna, zato je vedno mogoče pretvoriti te površine v površine NURBS, s katerimi lahko v nadaljevanju uporabimo preostale površinske ali prostorske ukaze.

Tak način modeliranja se precej razlikuje od generativnega modeliranja in je bolj naraven za oblikovalce. Tudi oblike izdelkov so bolj naravne in manj matematične. Generativne ukaze lahko vseeno uporabljamo v kombinaciji prostega modeliranja. Na začetku lahko uporabimo generativno modeliranje za kreiranje krivulj, ki nato služijo za izdelavo osnovne geometrije. Tudi med modeliranjem lahko uporabimo ukaze, kot je na primer odmik, uporabljen v zgornjem primeru. Ko končamo takšno modeliranje, lahko robne krivulje tako izdelanih površin uporabimo za generativne ukaze, kot je na primer premik.

9.3 Modelirniki za modeliranje prostih površin

Večina CAD modelirnikov, omenjenih v poglavju 8, vsaj v določeni meri podpira prosto modeliranje. Modeliranje površin z uporabo algoritmov glajenja površin podpirajo le določeni programi. Autodesk Inventor [129] in Fusion 360 [133] uporabljata za prosto modeliranje zaščiteno tehnologijo, imenovano T-splines. Tehnologijo T-splines Autodesk uporablja tudi v modelirniku Maya [77]. Preostali proizvajalci uporabljajo bodisi Catmul-Clark ali kakšno svojo izvedbo glajenja površin. Tako je v programih Catia [126], NX [127] in Creo [128] seveda tudi deljenje površin na vrhunski ravni. V programu Rhinoceros [134] so v preteklosti uporabljali T-splines, v novejših verzijah so prešli na nekoliko spremenjeno tehnologijo SubD. V nekaterih programih, kot je Onshape [132], je prosto modeliranje z glajenjem površin na voljo v obliki dodanega vtičnika. V Solidworksu za zdaj še ni mogoče uporabljati glajenja površin, vendar je mogoče to uporabiti v 3Dexperience modulu xShape [139]. Prosto modeliranje je zelo primerno za programe v navidezni resničnosti, kjer je najbolj znan program Gravity Sketch [140], ki omogoča modeliranje površin z zamahi rok ali modeliranje SubD površin.

10 Hibridno modeliranje

Cilj 3D modeliranja je prostorski model izdelka, ki ga lahko uporabimo za nadaljnjo uporabo za simulacije in nato za dejansko izdelavo. Zato v zaključni fazi površinskega modeliranja vedno pridemo do oblikovanja izdelka s prostorskim modelirnikom. Izkaže se, da smo tudi pri prostorskem modeliranju pogosto prisiljeni v uporabo površinskih ukazov. Kombiniranje ukazov prostorskega in površinskega modeliranja imenujemo hibridno modeliranje. Za nekatere CAD programe je hibridno modeliranje naravni način delovanja, medtem ko pri nekaterih programih lahko vključimo ali izključimo hibridno modeliranje. Če je hibridno modeliranje izključeno, je treba vsak rezultat površinskega modeliranja naprej pretvoriti v prostorski model.

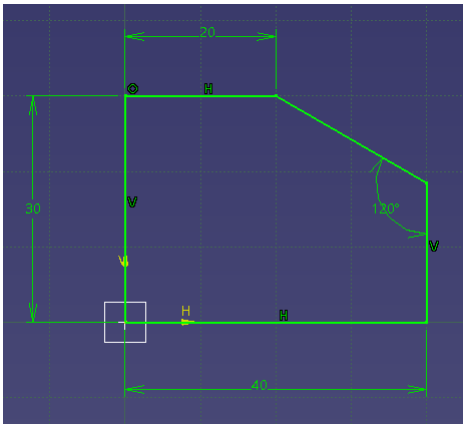
10.1 Prostorsko modeliranje

S prostorskim modeliranjem modeliramo zaprto (angl. Manifold) telo. Modeliranje običajno poteka tako, da na ravnino narišemo osnovni profil, ki ga s 3D ukazom spremenimo v telo. Prvotni modelirniki so to počeli tako, da je bilo treba vse geometrijske veličine fiksirati oziroma so imeli konstantne in nesprejemljive dimenzije. Mnogo večjo fleksibilnost modeliranja je omogočilo parametrično modeliranje, pri katerem so dimenzije in oblike telesa spremenljive. V zgodnjem obdobju modelirnikov so bili 3D ukazi podobni ukazom proizvodnega strojništva in obdelovalnih strojev, kar je bilo pozneje zamenjano z dodajanjem značilk modela. Tak način, ki temelji na značilkah (angl. Feature Based Modelling), uporablja precej drugačno filozofijo. Namesto da na primer izrežemo luknjo, v model dodamo luknjo, kot dodatno značilko modela. To je seveda v nasprotju s proizvodnim razmišljanjem, vendar je tak način modeliranja danes prevladal.

10.1.1 Skicirka z omejitvami

Osnovno orodje prostorskega modeliranja je skicirka, s katero narišemo osnovno obliko oziroma skico telesa na ravnino. Prva skicirka na računalniku je program Sketchpad [12], ki je deloval podobno kot sodobne skicirke. Pozneje so v prvih modelirniki uporabljali orodja za 2D risanje profilov, ki niso delovala kot skicirka, ampak kot risarko orodje za 2D geometrijo, kjer so bile dimenzije natančno določene. Orodje ima ime skicirka, ker profil ne predstavlja končne oblike in ga lahko pozneje spreminjamo in prilagajamo. To omogoča vgrajeno parametrično modeliranje, kjer so vse geometrijske in dimenzijske omejitve (angl. Constraints) spremenljive. Prvo takšno parametrično skicirko so skupaj s parametričnim modeliranjem predstavili v podjetju PTC v programu ProEngineer leta 1987 [141]. V skicirki porabnik profila ne riše natančno, ampak zgolj skicira približno obliko. V naslednjem koraku dodaja omejitve tako, da geometrijsko omeji določene elemente skice. Tako lahko določimo, da je črta vodoravna ali navpična ali vzporedna ali pravokotna na drugo črto oziroma tangentna na krožnico itd. Podobno kot s prej navedenimi geometrijskimi omejitvami lahko skico omejimo tudi z dimenzijskimi omejitvami. To pomeni, da določimo dolžino črte, razdaljo od drugega elementa, naklonski kot glede na drugi element itd. Ko vnesemo ravno dovolj omejitev, dobimo polno določeno skico (angl. Constrainded Sketch), kjer so določene vse spremenljivke v profilu. Če nato dodamo še eno omejitev, dobimo predimenzionirano skico in za uspešno nadaljevanje je treba kakšno omejitev izbrisati. V programih običajno barva črt na skici določa, ali gre za poddimenzionirano, predimenzionirano ali polno določeno skico. Vedno je zaželena določena skica, vendar to ne pomeni, da skice ni več mogoče spreminjati. Kadarkoli lahko kakšno omejitev zamenjamo z drugo, vendar tako, da spet dobimo določeno skico profila, s katerim nato gradimo telo. Skicirke se razlikujejo po obliki in uporabniškem vmesniku, vendar je večina elementov precej podobnih v različnih programih. Mogoče je še največje odstopanje v programu Catia, kjer je skicirka nekako ločena od modelirnika in deluje v samostojnem oknu. Kljub temu se tudi v programu Catia način dela s skicirko ne razlikuje veliko od drugih modelirnikov, kjer je skicirka le prikazana na ravnini v prostorskem modelirniku.

Na sliki 10.1 je prikazana polno določena skica v programu CATIA. Na sliki so na ozadju vidne črte mreže (angl. Grid), s katero si lahko pomagamo pri skiciranju in jih je mogoče vključiti in izključiti ali spremeniti njihovo gostoto. Pri kreiranju skice je treba paziti na vrstni red določanja omejitev, saj se lahko hitro zgodi, da se pri dodajanju omejitev oblika skice spremeni in popači. To je posebej pogosto, ko je prvotna skica narisana v dimenzijah, ki se precej razlikujejo od končnih dimenzij profila. Zato izdelava profila v skicirki lahko zahteva veliko več časa kot poznejša pretvorba v telo, ki se običajno izvede z enim ukazom.

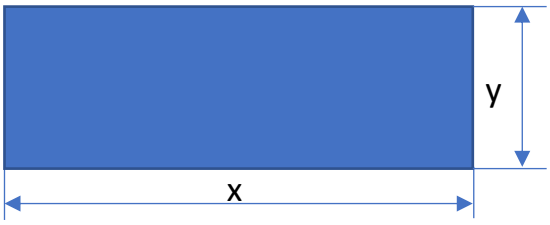


Slika 10.1. Skicirka v programu CATIA

Tudi ko je izdelano prostorsko telo, se je treba pogosto vrniti v skicirko in popraviti profil. Lahko izkoristimo še eno značilnost parametričnega modeliranja in namesto konstantnih vrednosti omejitev v skicirki dodamo spremenljive vrednosti v obliki parametrov, ki jim pozneje lahko določimo ali spreminjamo vrednosti.

10.1.2 Parametrično modeliranje

Parameter je grška sestavljenka »para« in »metron« in direkten prevod bi bil »poleg-mera«. Pri parametričnem modeliranju lahko v skicirki namesto konstantnih vrednosti zapišemo spremenljivke tako, kot je prikazano na sliki 10.2. Parametrično modeliranje ima zasnove že pred računalniško ero, saj lahko tudi Gaudijeve žične modele, prikazane na sliki **Napaka! Vira sklicevanja ni bilo mogoče najti.**, obravnavamo kot parametrične modele [142]. V računalniškem smislu se je parametrično modeliranje začelo s programom ProEngineer leta 1987 in zaradi velike popularnosti je danes parametrično modeliranje prisotno skoraj v vseh CAD modelirnikih. Osnovna ideja je v tem, da lahko spremenljivkam pozneje kadarkoli spremenimo vrednost in dobimo drugačno velikost in obliko profila.



Slika 10.2. Dimenzije, zapisane s spremenljivko

Spremenljivkam lahko določimo konstantne vrednosti ali določimo relacijo med parametri. Tako lahko na sliki 10.2 določimo $x = 10$ mm in $y = x / 2$. Na tak način bo ne glede na velikost profila oblika ostala vedno enaka. V tem preprostem primeru lahko tudi ugotovimo, da se oba parametra nekoliko razlikujeta.

Parameter *x* je določen z vrednostjo, ki jo mora določiti uporabnik in ni odvisna od drugih parametrov. Zato takšnim parametrom rečemo neodvisni parametri. Parameter *y* je po drugi strani popolnoma odvisen od parametra *x*, saj se bo s spremembo vrednosti parametra *x* samodejno spremenila vrednost parametra *y*. Zato rečemo, da so takšni parametri odvisni parametri. V modelu je zaželeno, da je čim manj neodvisnih parametrov in so vsi drugi parametri odvisni. Če so določene dimenzije v parametričnem modelu konstantne oziroma zapisane s številko namesto s parametrom, lahko pride do težav, ko se preostale dimenzije modela spreminjajo. V skrajnem primeru potem ni mogoče več rekonstruirati modela. Zato pogosto določimo le omejeno število neodvisnih parametrov, za katere uporabljeni parametri zagotavljajo veljavnost modela. V ta namen lahko uporabimo konstrukcijsko preglednico, v kateri so navedeni vsi neodvisni parametri.

Na sliki 10.3 je prikazan primer konstrukcijske preglednice v Solidworksu za parametrični model gorskega kolesa. V CAD programih je običajno mogoče urejati konstrukcijske preglednice tudi zunaj CAD programa. Najpogosteje v ta namen uporabljamo Microsoft Excel, seveda je to mogoče tudi vgraditi v kakšen zunanji program, ki omogoča izgradnjo CAD modela po meri.

	$VALUE@Hraz@equations	$VALUE@Ltero@equations	$VALUE@Bram@equations	$VALUE@Dkol@equations	$VALUE@Bkol@equations	$VALUE@RPR@equations
S	=770mm	=1150mm	=350mm	=26	=2	=400
M	=815mm	=1200mm	=400mm	=27.5	=2.1	=400
L	=861mm	=1250mm	=450mm	=27.5	=2.2	=500
XL	=907mm	=1300mm	=500mm	=29	=2.3	=500

Slika 10.3. Konstrukcijska preglednica v Solidworksu [143]

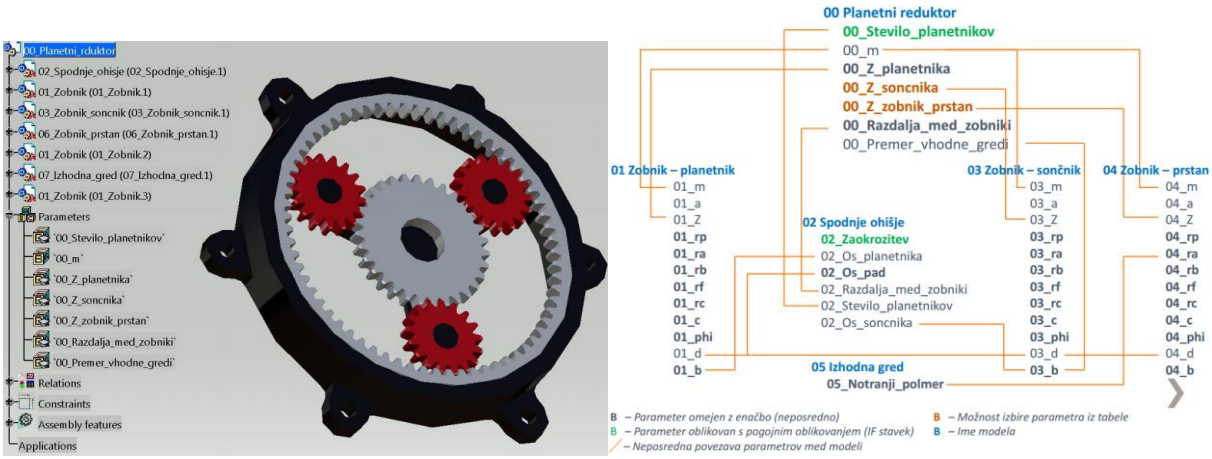
Za določanje odvisnih parametrov lahko uporabljamo enačbe z običajnimi matematičnimi operatorji ter številnimi matematičnimi funkcijami in konstantami. Zato lahko odvisni parameter določa zelo zapletena matematična enačba. V določenih primerih tudi zelo kompleksne enačbe ne zadoščajo, saj je velikost določenega parametra nemogoče neposredno izračunati, ampak je treba parameter določiti glede na določene pogoje. V takih primerih se poslužujemo logične funkcije »če« (angl. if). V programu Solidworks je ta funkcija imenovana »IIF«. Funkcija je posebej uporabna, ker lahko v stavek funkcije vgradimo novo funkcijo »IIF« in dobimo kompleksne vgnezdene odločitvene stavke. Primer takšnega stavka je prikazan na sliki 10.4.

```
h = IIF("W" <= 3254000, IIF("W" <= 1944000, IIF("W" <= 1572000, IIF("W"
<= 1434000, IIF("W" <= 1224000,368,375),380), IIF("W" <
= 1760000,387,393)), IIF("W" <= 2641000, IIF("W" <
=  2380000, IIF("W" <= 2125000,399,407),416), IIF("W" <
= 2938000,425, IIF("W" <= 3254000, 435,446)))), IIF("W" <
= 4994000, IIF("W" <= 4284000, IIF("W" <= 3947000, IIF("W"
<=  3625000,446,455),465), IIF("W" <
= 4634000,474,483)), IIF("W" <= 6938000, IIF("W" <
=  6203000, IIF("W" <= 5552000,498,514),531), IIF("W" <
= 7739000,550, IIF("W" <= 8645000, 569, IIF("W" <
= 9712000,580,600))))))
```

Slika 10.4. Vgnezden IIF stavek v Solidworksu [144]

Poleg tega parametrično modeliranje omogoča še številne druge načine za avtomatizacijo prostorskega modela. Tako lahko vgradimo različna opozorila ali semaforje, določimo različna pravila ali zakone konstruiranja in vse skupaj povežemo z makro programi. Na tak način je mogoče CAD program popolnoma prilagoditi končnemu uporabniku. Modeliranje specifičnih izdelkov je tako dosegljivo tudi za uporabnike, ki niso vešči modeliranja. V skrajnem primeru takšne avtomatizacije si lahko kupec izdelka sam kreira končno obliko in dimenzije želenega izdelka.

Parametrično modeliranje je precej bolj zahtevno za konstruktorja. Zahteva dober razmislek na začetku in načrtovanje modeliranja. Posebej zahtevno je, ko je treba narediti odvisnost parametrov v sestavi. Na sliki 10.5 je prikazan parametrični model planetnega gonila na levi strani slike, na desni strani slike pa je shema načrtovanja parametričnega modela. Načrtovanje parametrov na začetku olajša modeliranje, saj je brez načrtovanja treba narediti veliko sprememb v poznejših fazah modeliranja, ko ugotovimo določeno odvisnost med posameznimi deli v sestavi.

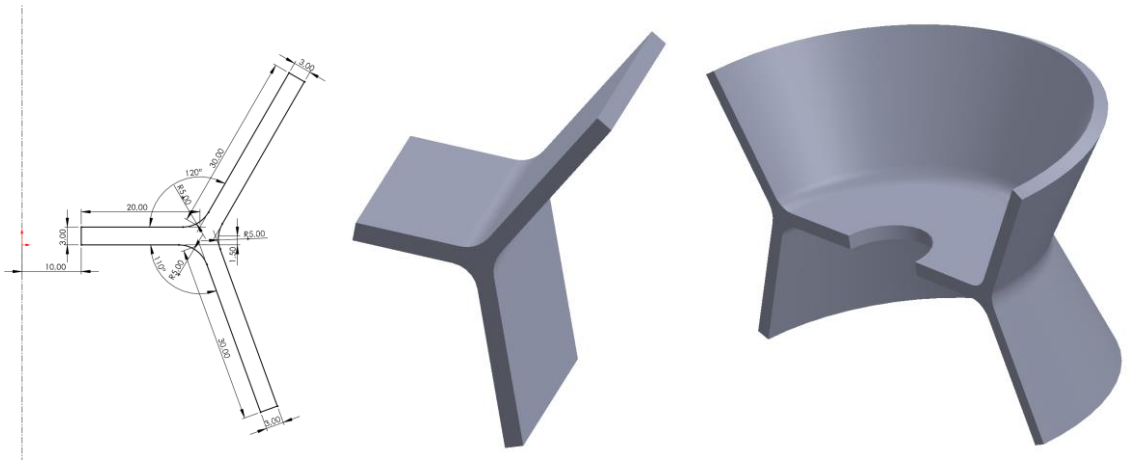


Slika 10.5. Parametrični model planetnega gonila

Ko je model modeliran parametrično, so poznejše spremembe modela veliko lažje, saj je treba le spremeniti določen parameter in dobimo drugo varianto izdelka. Poleg tega so določene dimenzije izdelka izračunane in zapisane v enačbah. To pomeni, da je shranjeno v berljivi obliki tudi znanje o izdelku, kar je zelo koristno za konstruktorja, ki modificira obstoječi izdelek drugega konstruktorja.

10.1.3 Prostorski ukazi in značilnosti

Prostorski ukazi so precej podobni ukazom v površinskem modelirniku, pri čemer je tukaj prostorski model rezultat modeliranja. To pomeni, da ima model volumen in ob določitvi materiala oziroma gostote je mogoče določiti tudi maso izdelka. Na sliki 10.6je prikazan primer dveh prostorskih ukazov z enako skico v Solidworks. Profil na levi strani slike lahko raztegnemo v prostor ali zavrtimo okoli osi.



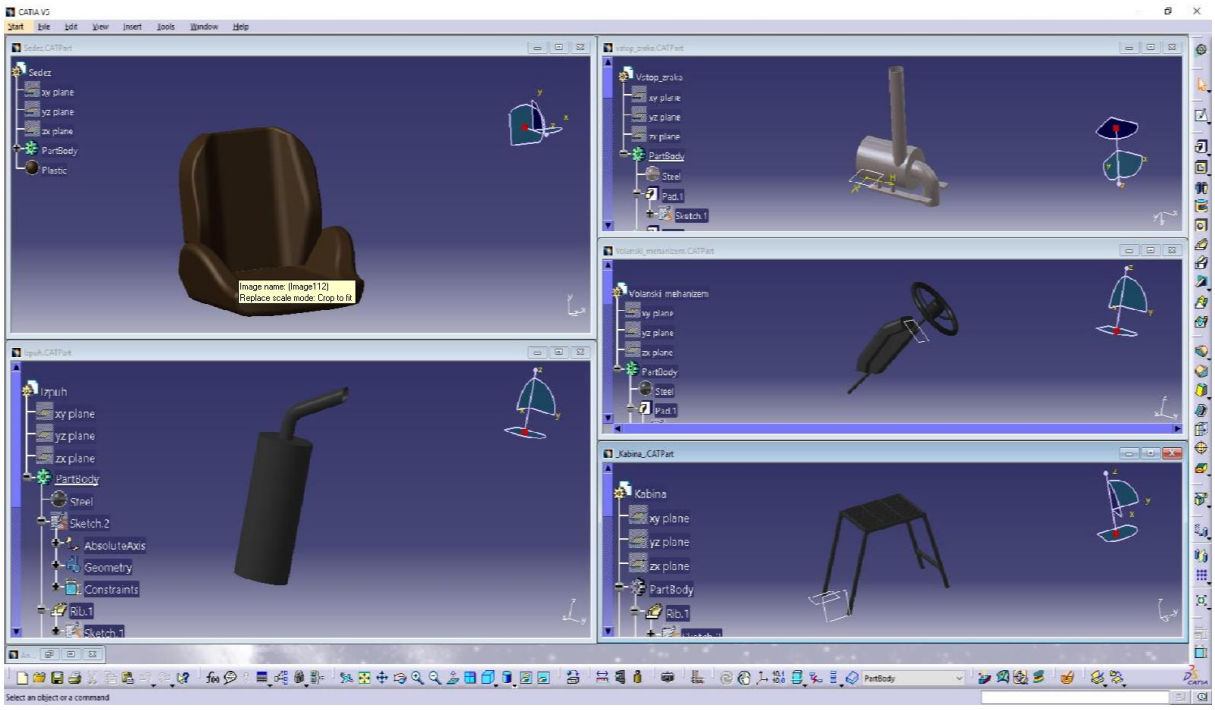
Slika 10.6. Prostorski ukazi v Solidworksu

Različnih prostorskih ukazov sicer ni veliko, zato je treba izdelek modelirati s kombinacijami ukazov. Sodobni modelirniki, ki temeljijo na parametričnem modeliranju, uporabljajo za modeliranje običajno pristop, ki se imenuje »modeliranje z značilkami« (angl. Feature Based Modelling) [145].

Pri modeliranju na tak način gradimo model z dodajanjem značilk (angl. Feature) osnovnemu modelu. Vsak del modela je določena značilka, ki dodaja neko lastnost oziroma značilnost modelu. Tako lahko osnovnemu modelu, ki ga naredimo s 3D ukazom (npr. izvlek ali zasuk), dodamo luknje, zaokrožene ali odrezane robove itd. Značilke so lahko tudi bolj kompleksne značilnosti modela. Za potrebe simulacije lahko izdelamo mrežo končnih elementov za prostorski model, ki je tako kot običajen zaokrožen rob samo še ena značilnost modela. Posledično so vse značilnosti modela v odvisnosti. Ko spremenimo dimenzijo modela in s tem spremenimo rob, se hkrati spremeni tudi zaokrožitev roba in na enak način se prilagodi mreža končnih elementov, saj je le značilka v modelu. Glavne značilnosti modeliranja z značilkami so:

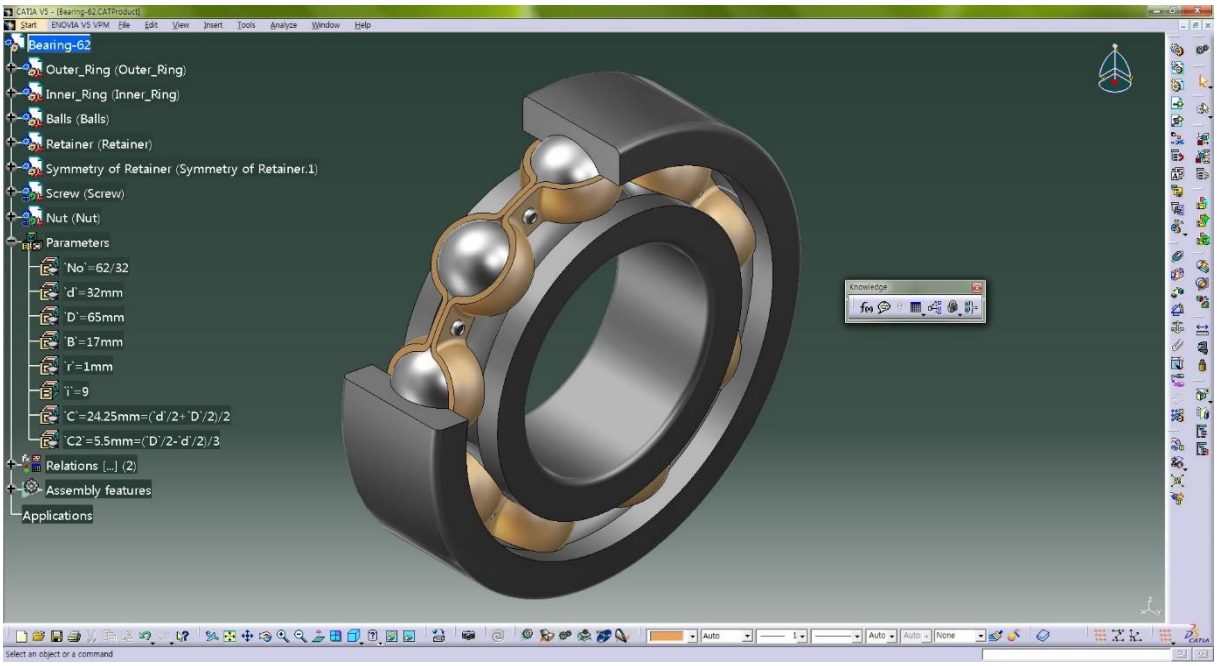
- Prilagodljivost: Modeli se lahko preprosto prilagajajo z nastavljanjem parametrov značilk, kar omogoča hitre spremembe in iteracije pri oblikovanju.
- Ohranjanje oblikovalskega namena: Parametrični odnosi med značilkami ohranjajo izvorno oblikovalsko namero, kar zagotavlja doslednost in nadzor pri spremembah.
- Ponovna uporabnost: Značilke se lahko ponovno uporabijo v več modelih, kar povečuje učinkovitost in produktivnost pri oblikovalskih procesih.
- Parametrični odnosi: Značilke se lahko povezujejo med seboj prek odvisnosti, omejitev ali sklicev, kar omogoča kompleksne odnose in dinamično obnašanje.
- Sledenje zgodovini: Zaporedje značilk in njihovi parametri tvorijo zgodovino modela, ki se lahko uporablja za pregled, analizo in razumevanje gradnje modela.

Praktično model upravljamo z značilkami v drevesni strukturi modela. Drevesno strukturo značilk, skupaj z modeliranjem z značilkami, so najprej predstavili v programu Pro/Engineer v začetku 80-ih let 20. stoletja. Pozneje so to prevzeli praktično vsi modelirniki. Izvedba drevesne strukture je odvisna od specifičnega programa in v drevesni strukturi sestave najdemo posamezne dele sestave, ki se nato razvejajo na značilke, ki oblikujejo kos. Pogosto so tudi parametri in zakoni, ki oblikujejo izdelek, sestavni del drevesne strukture izdelka. Na sliki 10.8 so prikazana okna s kosi in poleg vsakega kosa je drevesna struktura z značilkami, s katerimi je kos definiran.



Slika 10.7. Deli z značilkami v programu CATIA V5

Na sliki 10.8 je prikazana sestava ležaja, kjer so poleg delov v drevesni strukturi tudi parametri, s katerimi je mogoče parametrično spreminjati ležaj.



Slika 10.8. Drevesna struktura v programu CATIA V5

10.2 Direktno modeliranje

Direktno prostorsko modeliranje [146] je precej podobno prostemu modeliranju površin. Namesto da bi uporabili vnaprej definirane ukaze za kompleksne prostorske ukaze, za večino operacij uporabimo samo dva preprosta ukaza. Eden od teh ukazov je premik geometrije, kjer z miško »primemo« in premaknemo geometrijski element. Drugi ukaz pa je vlek, kjer z miško »primemo« in izvlečemo določen geometrijski element. Pri tem seveda ni vseeno, za kakšen geometrijski element gre. Če imamo 2D skico, jo z ukazom vlek lahko izvlečemo v prostor, kot z ukazom vlek pri modeliranju z značilkami. Podobno se ukaz obnaša, ko »primemo« površino 3D telesa. Ko je geometrijski element rob 3D telesa, z vlekom naredimo zaokrožitev namesto roba. Glavna razlika je v tem, da pri direktnem modeliranju nimamo shranjene zgodovine. Ko smo izvlekli 2D skico, ta skica ni značilka modela, ki bi jo lahko popravili, ampak je preprosto postala del celotne geometrije. Obliko lahko naknadno spreminjamo s ponovnim modeliranjem. Čeprav je direktno modeliranje na izgled privlačno in prosto oblikujemo izdelek, je za realen izdelek treba določiti natančne mere geometrije. Med modeliranjem je to seveda mogoče določiti, pozneje je mere precej težje spreminjati.

Prednosti direktnega modeliranja:

- Enostavnost in intuitivnost: Direktno modeliranje omogoča hitro in intuitivno oblikovanje, saj oblikovalci lahko neposredno premikajo, vlečejo, obračajo in spreminjajo geometrijo brez kompleksnega nastavljanja parametrov.
- Hitra iteracija in prilagodljivost: Spremembe v modelu se lahko hitro in preprosto izvedejo, kar omogoča hitrejšo iteracijo in prilagajanje oblikovalskim spremembam.
- Neodvisnost od zgodovine urejanja: Pri direktnem modeliranju ni treba slediti zaporedju urejanja ali odvisnosti značilk, kar omogoča večjo svobodo pri oblikovanju in urejanju modela.
- Boljša prilagoditev uvoženim modelom: Direktno modeliranje je učinkovito pri prilagajanju in urejanju uvoženih 3D modelov, saj se lahko hitro popravijo morebitne napake ali optimizirajo oblikovalski elementi.

Slabosti direktnega modeliranja:

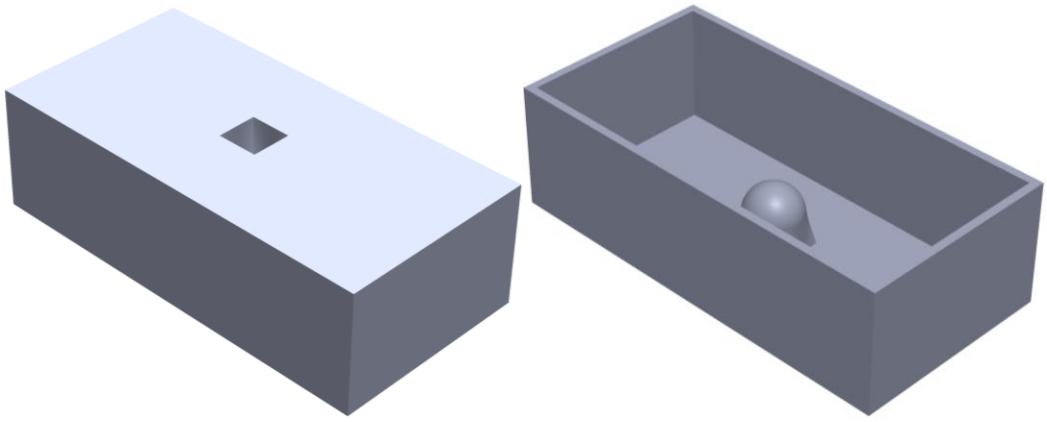
- Pomanjkanje parametričnosti: Brez parametričnih značilk in zgodovine urejanja je težje slediti spremembam modela, kar lahko otežuje oblikovalski proces in prilagajanje modela v prihodnosti.
- Omejen nadzor nad oblikovalskimi nameni: Zaradi pomanjkanja parametričnih povezav je težje ohranjati in nadzorovati izvorno oblikovalsko namero, kar lahko vpliva na doslednost in natančnost modela.
- Omejena ponovna uporabnost: Direktno modeliranje pogosto ustvari specifične geometrije, ki morda niso preprosto ponovno uporabne v drugih projektih ali kontekstih.
- Težave pri kompleksnih modelih: Pri zelo kompleksnih modelih, kjer je treba slediti kompleksnim odvisnostim in parametričnim relacijam, se lahko direktno modeliranje izkaže za manj primerno, saj lahko postane težko obvladljivo in težje sledljivo.

Zaradi teh slabosti tudi v direktnih modelirnikih pogosto najdemo parametre, s katerimi lahko model opremimo in pozneje spreminjamo. Pogosto se ukazi direktnega modeliranja shranjujejo v obliki programske kode in tako dobimo zgodovino modeliranja. S spreminjanjem te programske kode lahko pozneje spreminjamo tudi model, vendar je to precej bolj togo, kot je pri modeliranju z značilkami. Mogoče je najbolj značilen primer direktnega modelirnika program SpaceClaim [147], ki ga najdemo v okviru programskega paketa Ansys. Tudi SpaceClaim omogoča ob direktnem modeliranju beleženje zgodovine in delo s parametri. Podoben način modeliranja uporablja tudi program Shapr3D [148], ki deluje tudi na grafičnih tablicah iPad.

V sodobnih modelirnikih z značilkami danes pogosto najdemo tudi ukaze za direktno modeliranje. Pogosto se to imenuje »synchronous modelling«, kjer lahko direktno spreminjamo model in rezultat je nato shranjen v obliki značilke.

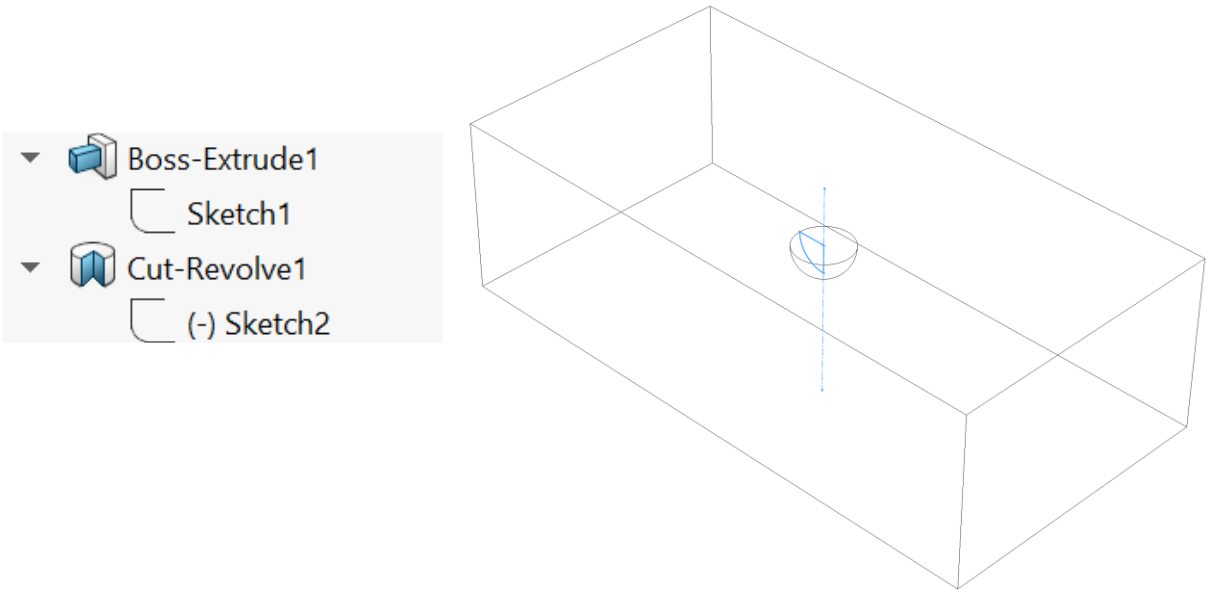
10.3 Hibridno modeliranje

Po definiciji je hibridno modeliranje [149] kombinacija različnih vrst modeliranja od volumetričnega, poligonskega, prostorskega do površinskega modeliranja. V nekaterih primerih je tudi kombinacija direktnega modeliranja in modeliranja z značilkami že neke vrste hibridno modeliranje, čeprav gre pri tej kombinaciji še vedno za različne ukaze prostorskega modeliranja. Termin hibridno modeliranje se najpogosteje uporablja za kombinacijo različnih vrst prostorskega modeliranja s površinskim modeliranjem. Pri takšnem modeliranju je treba paziti, kateri način modeliranja izberemo. Na preprostem primeru na sliki 10.9 lahko vidimo razliko med modeliranjem s čistim prostorskim modeliranjem in hibridnim modeliranjem v programu Solidworks.



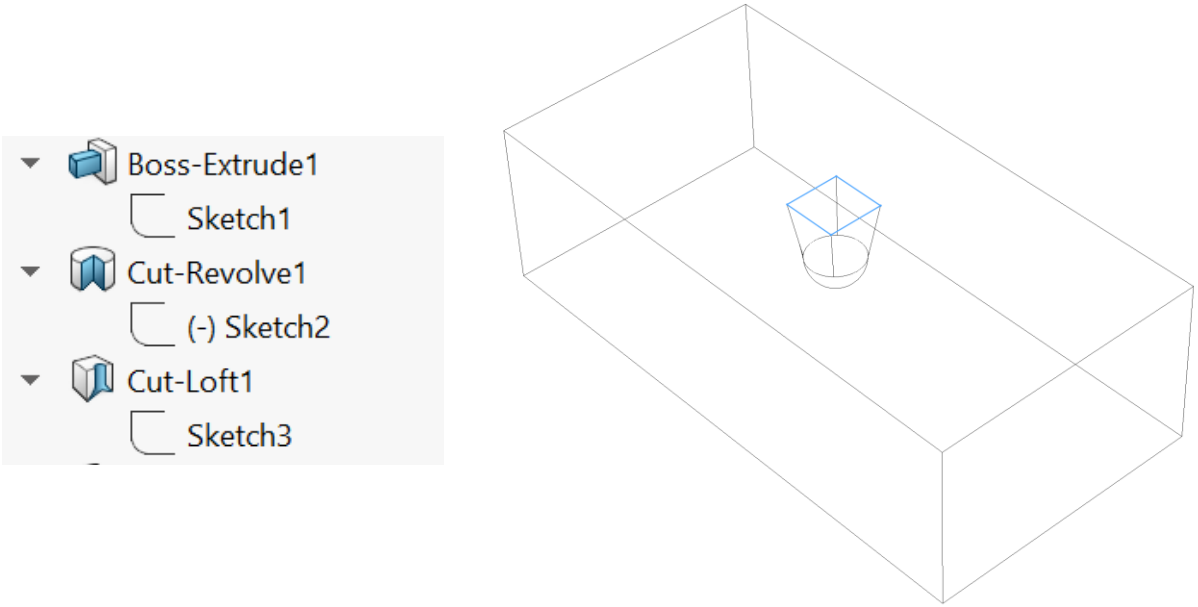
Slika 10.9. Primer modela z zgornje in spodnje strani

V tem primeru modela želimo na običajno telo v obliki kvadra narediti izrez, ki ima na zgornji strani kvadratno obliko, spodaj pa se zaključi sferično. Ta model lahko zmodeliramo v Solidworks samo z uporabo prostorskih ukazov tako, da najprej naredimo prostorsko telo kvader z ukazom Boss-Extrude, kjer izvlečemo pravokotni profil v višino. V nadaljevanju uporabimo prostorski ukaz Cut-Revolve in iz kvadra izrežemo sferično obliko na spodnji strani tako, kot je prikazano na sliki 10.10.



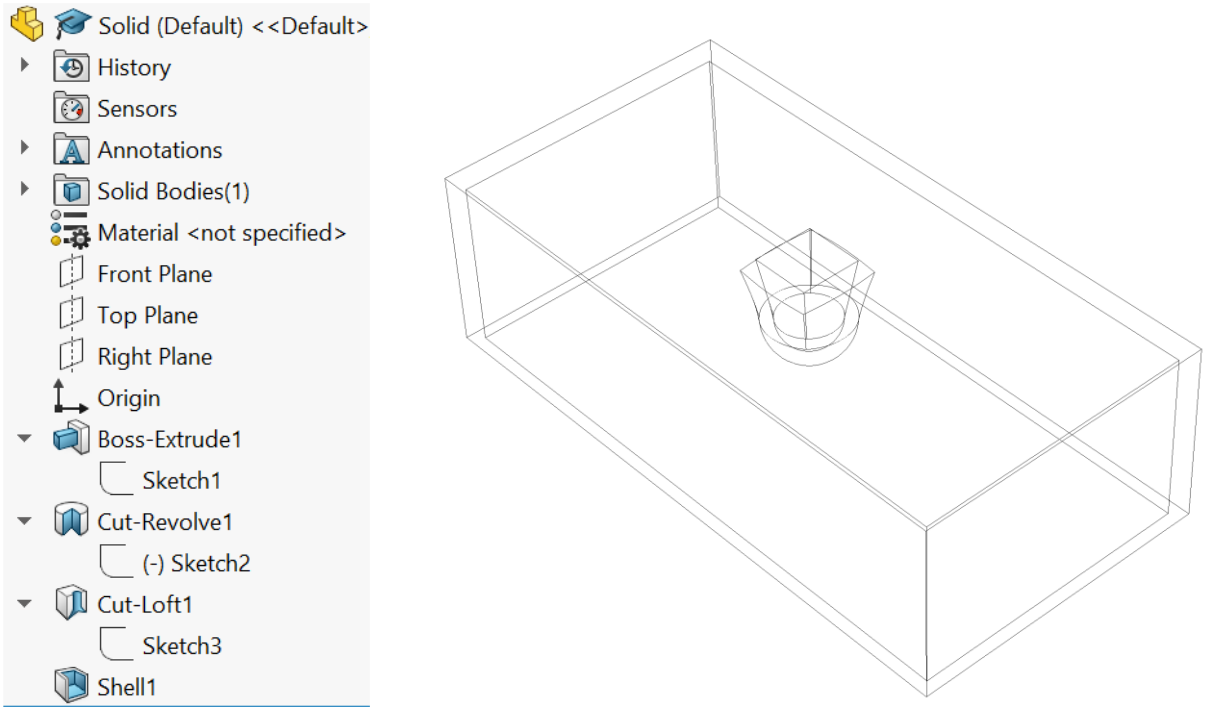
Slika 10.10. Sferični izrez na spodnji strani

Nato na zgornji površini kvadra narišemo kvadratni profil izreza. Z ukazom Cut-Loft povežemo zgornji kvadratni profil s spodnjim krožnim profilom in vse skupaj izrežemo iz modela tako, kot je prikazano na sliki 10.11.



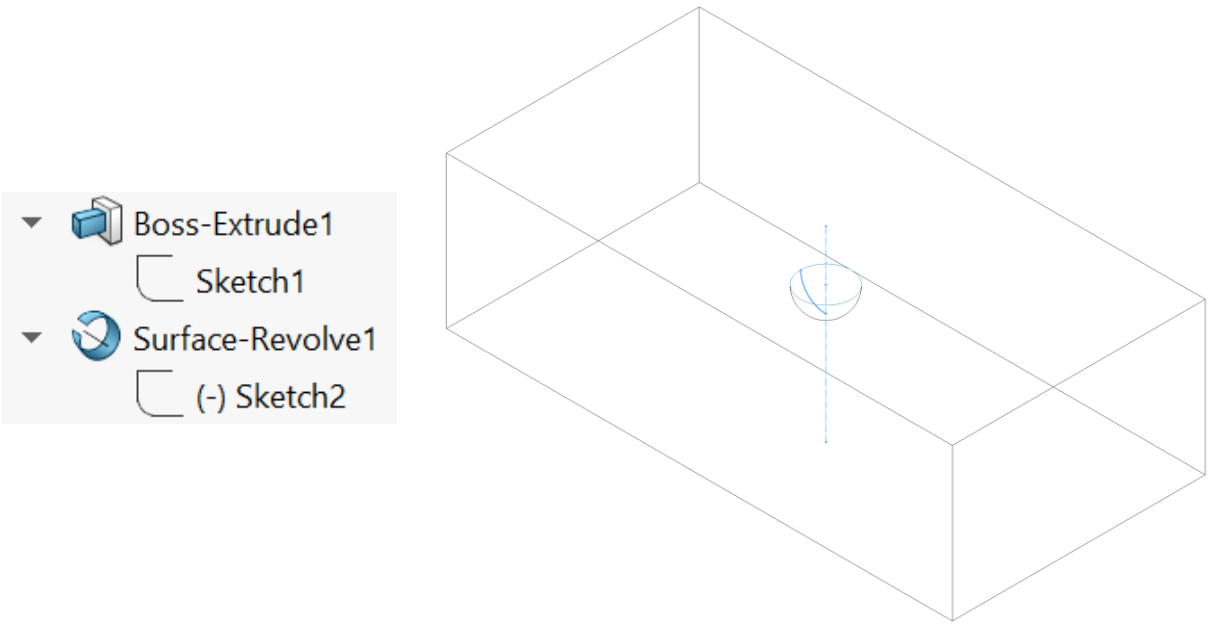
Slika 10.11. Izrez z uporabo ukaza Cut-Loft

Na koncu samo izvedemo ukaz Shell z debelino stene 2 mm in odstranitvijo spodnje ploskve kvadra in dobimo rezultat na sliki 10.12.



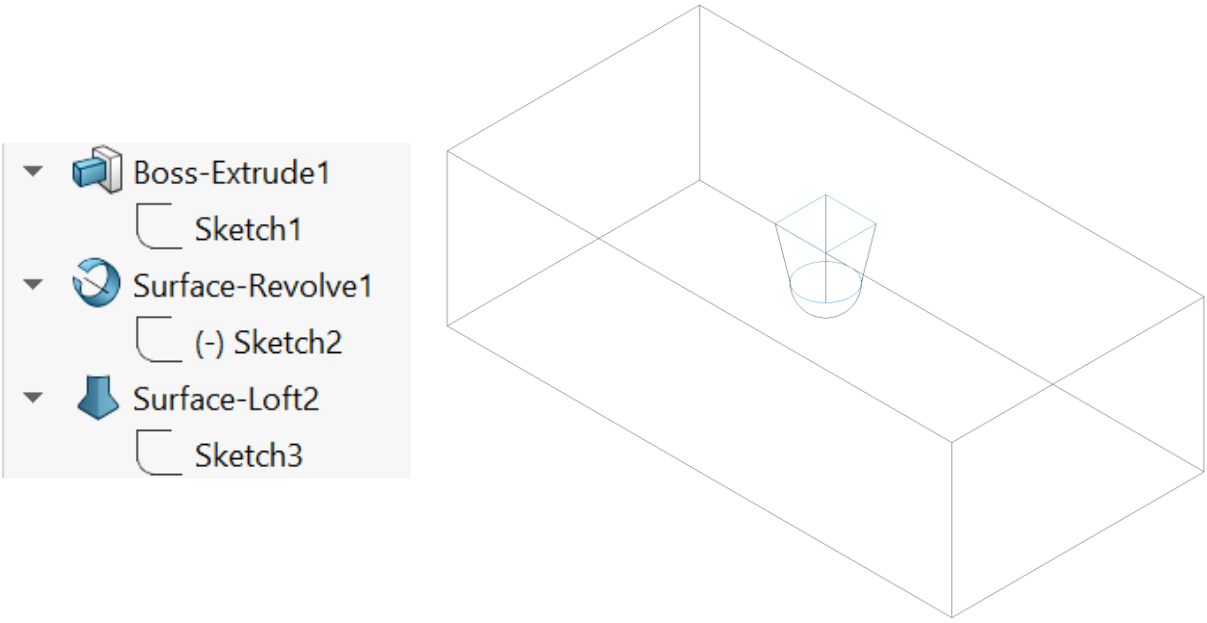
Slika 10.12. Končni izdelek s čistim prostorskim modeliranjem

Na nekoliko drugačen način pridemo do enakega izdelka, če uporabimo hibridno modeliranje, ko uporabljamo tako prostorske kot površinske ukaze. Prvi korak je enak in s prostorskim modeliranjem izdelamo kvader. Nato namesto ukazov za prostorsko rezanje uporabimo podobne ukaze, s katerimi kreiramo površine. Najprej naredimo sferično površino na spodnji strani izreza z ukazom Surface-Revolve. Tako kot pri prostorskem ukazu tudi tukaj naredimo os rotacije in krožni lok, ki ga zavrtimo okoli osi. Rezultat je prikazan na sliki 10.13.



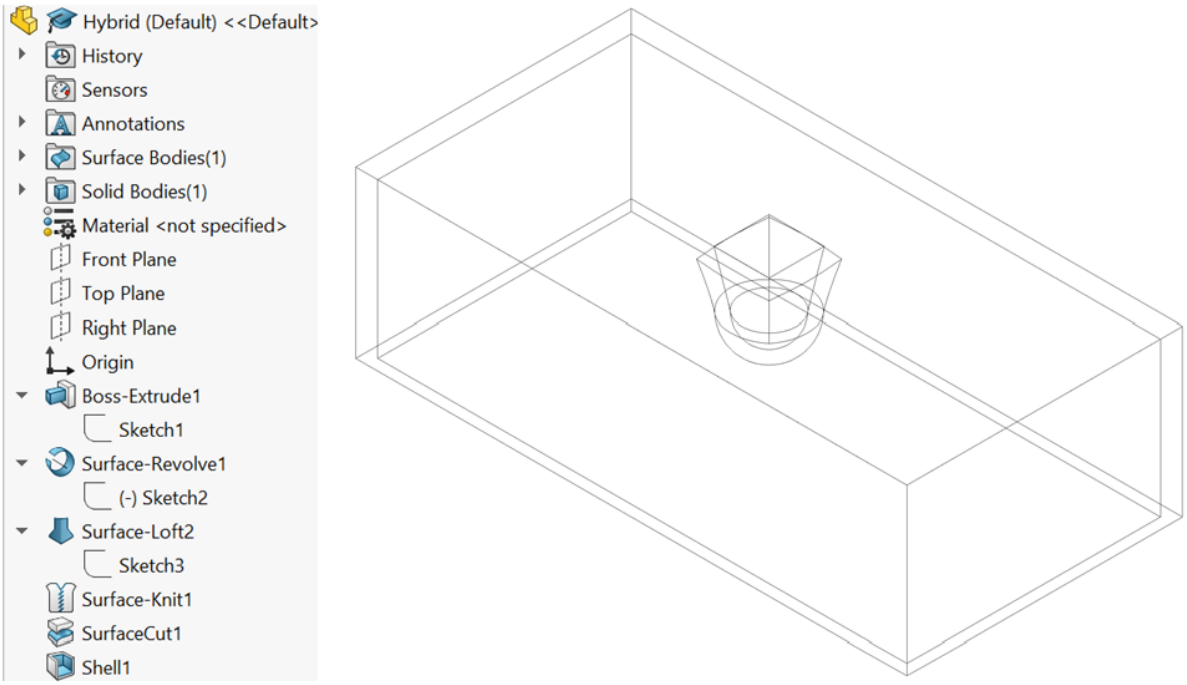
Slika 10.13. Sferična površina na spodnji strani izreza

V nadaljevanju uporabimo ukaz za kreiranje površine Surface-Loft med kvadratnim profilom na zgornji strani in krožnim profilom na spodnji strani tako, kot je prikazano na sliki 10.14.



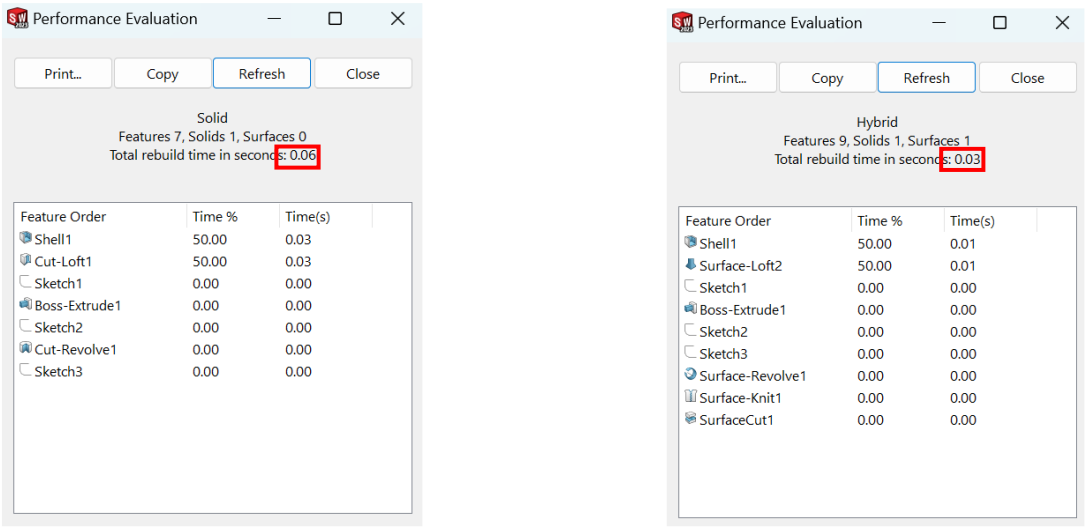
Slika 10.14. Površina, izdelana z ukazom Surface-Loft

Obe površini izreza povežemo z ukazom Surface-Knit in naredimo izrez iz kvadra z ukazom Surface-Cut. Ko na koncu spet naredimo lupino z ukazom Shell, dobimo enak prostorski model, kot je prikazan na sliki 10.12. Na sliki 10.15 je prikazan končni izdelek, izdelan s hibridnim modeliranjem, kjer se razlikuje le drevesna struktura, saj vsebuje površinske ukaze.



Slika 10.15. Končni izdelek, izdelan s hibridnim modeliranjem

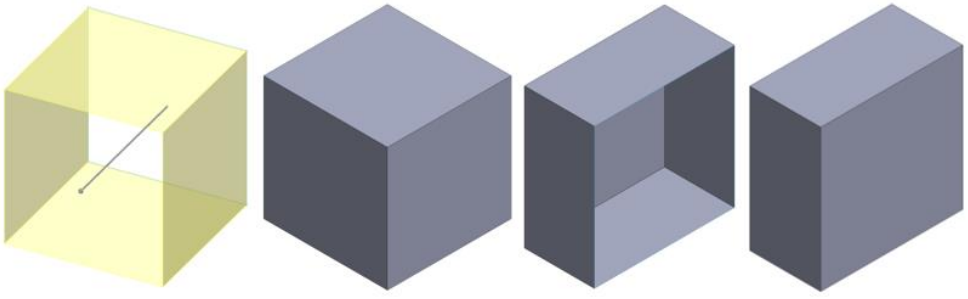
Če primerjamo oba načina modeliranja, je na prvi pogled prostorski način bolj učinkovit, saj vsebuje manj ukazov, ker ni bilo treba povezovati obeh izrezov in ni bilo treba uporabiti dodatnega ukaza za izrezovanje. Vendar je treba upoštevati tudi hitrost posameznih ukazov v modelirniku. Na sliki 10.16 je prikazana statistika generiranja modela, izdelanega s prostorskimi in hibridnimi ukazi. To statistiko v Solidworksu prikažemo z ukazom Tools/Evaluate/Performance Evaluation.



Slika 10.16. Statistika hitrosti izvajanja posameznega ukaza v Solidworksu

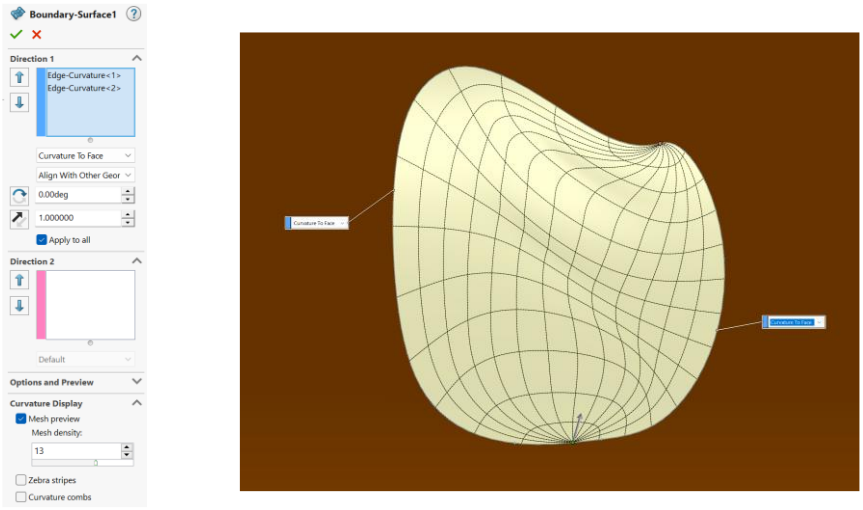
Na sliki 10.16 je na levi strani slike statistika izvajanja čistih prostorskih ukazov, na desni strani pa je statistika hibridnega modeliranja. Vidimo, da je hitrost regeneracije modela z uporabo hibridnega modeliranja dvakrat hitrejša od prostorskega modeliranja. Absolutno so časi izredno majhni in so prikazani v stotinkah sekunde. S kompleksnostjo modela narašča tudi pomembnost izbire načina modeliranja. Stokrat večji model bi porabil kar 3 sekunde več za regeneracijo, kar pa ni več zanemarljivo. V praksi uporabljamo še veliko večje modele in takrat je izredno pomembno, da izkoristimo prednosti hibridnega modeliranja. Ko primerjamo enakovredna ukaza, vidimo, da je že površinski ukaz Loft trikrat hitrejši od prostorskega ukaza Loft. Zanimivo je tudi, da je enak prostorski ukaz Shell hitrejši, če smo predhodno uporabili površinske ukaze. Ni nujno, da so vsi površinski ukazi hitrejši od prostorskih, zato je pri modeliranju kompleksnih modelov treba spremljati statistiko in modelirati z učinkovito regeneracijo modela [150].

Glavni razlog za hitrejšo regeneracijo hibridnih modelov je dejstvo, da vsi današnji modelirniki temeljijo na t. i. B-rep predstavitvi in interno vedno delajo s površinami ali ploskvami (oziroma ravninskimi površinami). Kocka je na primer predstavljena s ploskvami v prostoru. Na levi strani slike 10.17 je prikazano, kaj dobimo, če kvadratni profil raztegnemo s površinami. V nadaljevanju samo dodamo ploskev zadaj in spredaj in dobimo zaprto telo, ki ga lahko pretvorimo v prostorsko telo. Dokler ne naredimo pretvorbe, imamo samo kocko, ki je votla. Če uporabimo ravnino in z njo prerežemo kocko na polovico, dobimo za rezultat votlo polovico kocke, prikazano na tretjem modelu slike 10.17. Če v ukazu Knit-Surface označimo, da želimo prostorsko telo, bo pa tudi znotraj kocke material. Te prostorske kocke sicer ne moremo prerezati z ravnino, vendar lahko uporabimo prostorski ukaz Cut-Extrude na podoben način – na sredini kocke odrežemo polovico kocke, ki je na skrajni desni strani slike zdaj polna.



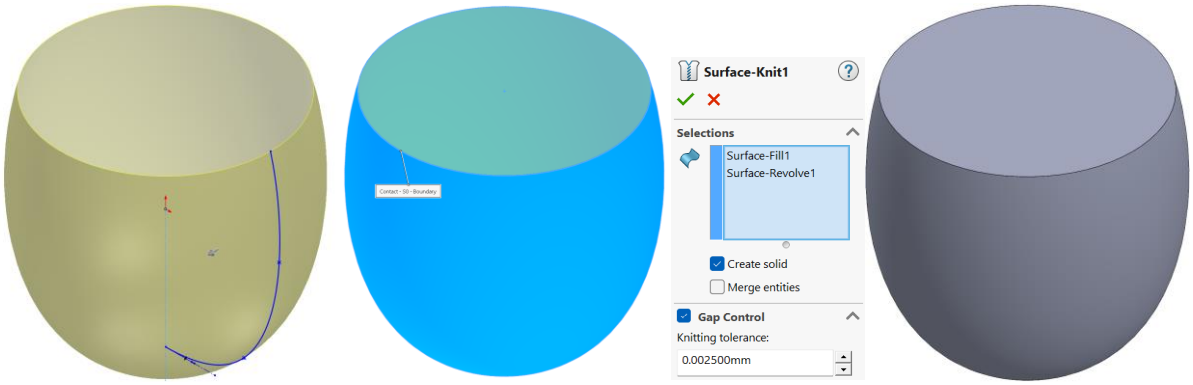
Slika 10.17. Predstavitev kocke s ploskvami

V modelirnikih je model v celoti predstavljen s površinami. To ne pomeni, da so vse površine ploskovne oziroma poligonske. Kadar modeliramo oblike, ki jih ni mogoče predstaviti s ploskovnimi površinami, uporabljamo parametrične površine NURBS za predstavitev površin. Naloga modelirnika je generiranje takšnih površin glede na zahteve uporabnika. V določenih primerih je to dokaj preprosto, ko gre za ploskovne ali rotacijske oblike. V splošnem je določanje površin precej zahtevno in pogosto zmožnost kreiranja kakovostnih površin določa kakovost modelirnika. Na sliki 10.18 je primer ovojne površine s prikazano mrežo, ki prikazuje kreiranje parametrične površine med dvema robovoma na luknji ukrivljene površine.



Slika 10.18. Mreža parametrične površine v programu Solidworks

Na koncu modeliranja vedno želimo prostorski model, saj ta predstavlja dejansko predstavitev resničnega izdelka. Zato je treba kreirati prostorsko telo tudi v primeru, ko začnemo model izdelovati samo s površinskim modeliranjem. To naredimo tako, da najprej zapremo površine in jih nato povežemo med seboj ter označimo, da naj bo rezultat operacije prostorsko oziroma solid telo. V primeru na sliki 10.19 je to dokaj preprosto, saj zgolj zapremo zgornji del vrtenine in vse skupaj z ukazom Surface-Knit pretvorimo v prostorsko telo.



Slika 10.19. Zapiranje površin v prostorsko telo v programu Solidworks

Zapiranje površin in pretvorba v prostorsko telo ni vedno tako preprosta. Veliko težav predstavljajo vrzeli med površinami, ki onemogočajo zapiranje. Zato je včasih bolj racionalno delati hibridno in že po nekaj korakih pretvoriti model iz površinskega v prostorsko telo. Po drugi strani lahko to predstavlja težavo saj se lahko čas regeneracije zaradi tega poveča. Težava se še poveča, ko oblikujemo kompleksne oblike površin, kjer je treba skrbeti za gladke prehode med površinami. Zato hibridno modeliranje zahteva veliko spretnosti in izkušenj uporabnika.

11 Renderiranje

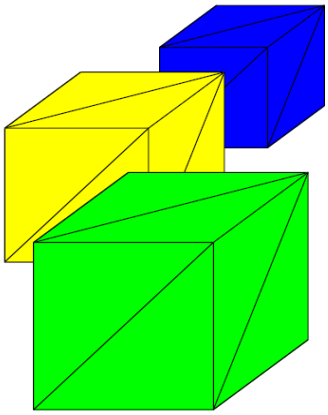
Renderiranje oziroma upodabljanje [151] je postopek, ko kreiramo sliko računalniškega 3D modela na zaslonu. Danes to pomeni upodabljanje v obliki rastrske grafike, saj so vse izhodne naprave od zaslonov, projektorjev do tiskalnikov in risalnikov rastrske in prikazujejo sliko, sestavljeno iz posameznih slikovnih elementov – pikslov. Čeprav so računalniki postali zelo hitri, je renderiranje zelo zahteven proces in običajno zahteva dodatno strojno opremo za kreiranje slike. Včasih je ta strojna oprema sicer vgrajena kar v procesor računalnika, vendar za potrebe programov za računalniško podprto konstruiranje to ne zadostuje. Zato ima delovna postaja za konstruiranje običajno zelo zmogljivo grafično kartico, ki vsebuje tako strojno kot programsko opremo za vse postopke renderiranja, ki bodo obravnavani v tem poglavju. Renderiranje namreč ni en homogen proces, ampak je treba uporabiti številne postopke, ki omogočajo kakovostno sliko modela na zaslonu. 3D model je v računalniku shranjen v celoti, vendar bo na zaslonu prikazan samo izrez, ki ga je izbral uporabnik, s površinami, ki jih uporabnik vidi iz svojega gledišča. Pri tem je želeno, da bodo te površine prikazane tako, kot je dejanski izdelek v naravi. Optimalna slika modela se v ničemer ne sme razlikovati od digitalne fotografije dejanskega izdelka.

11.1 Odstranjevanje skritih ploskev in robov

Prvi korak pri renderiranju je običajno določanje vidnega prostora objekta. Glede na izbrano projekcijo lahko določimo meje vidnega polja levo, desno, spodaj in zgoraj in poleg tega še vidno globino, kar pomeni mejo za blizu in daleč. To naredimo z operacijo izreza (angl. clipping) [152], kjer izločimo vso geometrijo zunaj tega prostora. Običajno gre pri tem za poligone, ki jih v celoti izločimo, v določenih primerih je mogoče tudi odrezati del poligona ali algoritem prilagodimo tako, da se pikselacija (postopek pretvorbe vektorske grafike v rastrsko) izvaja samo znotraj tega prostora za prikazovanje. Osnovni namen te operacije je zmanjševanje števila poligonov, s katerimi je treba izvajati nadaljnje operacije.

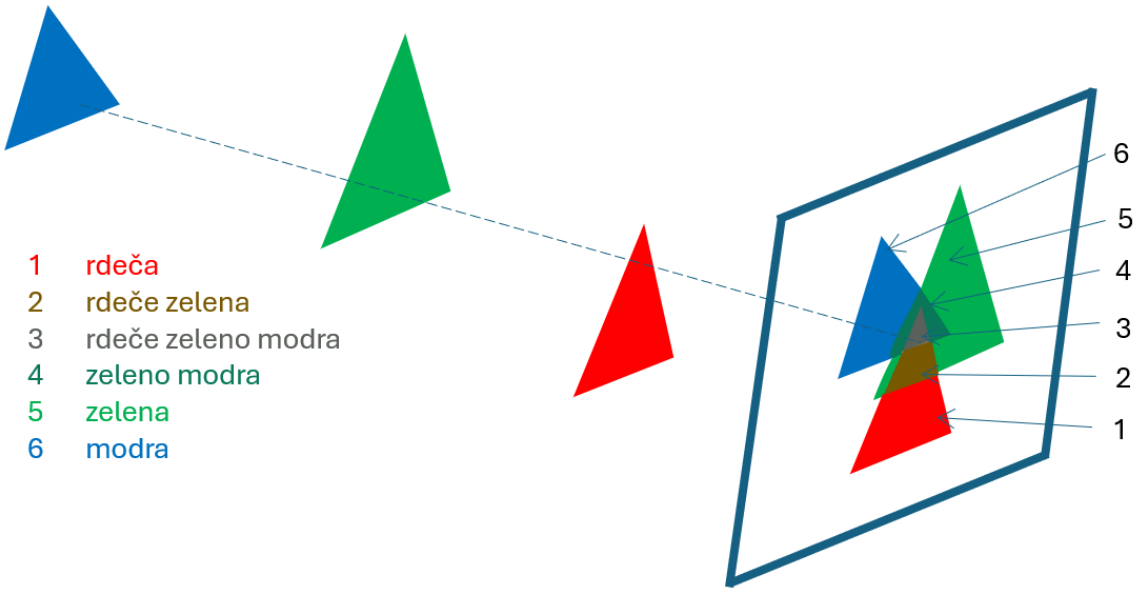
Naslednja operacija izločanja poligonov se nanaša na izločanje poligonov glede na gledišče. Iz gledišča zagotovo ne bomo videli zadnje strani objekta oziroma ploskev, ki so obrnjene vstran od gledišča. Takšno izločanje (angl. culling) je običajno povezano z izločanjem glede na normalo [153] tako, da izločimo vse poligone, pri katerih normala kaže vstran od gledišča. Pri površinskih modelirnikih je mogoče, da je v vidnem polju tako sprednja kot zadnja stran površine in v takšnih primerih isti filter omogoča različno obarvanje oziroma senčenje površin na eni in drugi strani.

Oba predhodna postopka običajno zmanjšata število poligonov, ki jih je treba prikazati na sceni. Težava pri tem je, da je treba poligone prikazati tako, kot so vidni v naravi. V računalniškem modelu so tudi po odstranitvi številnih poligonov še vedno na sceni poligoni, ki bodo delno ali v celoti skriti za sprednjimi poligoni. Problem odstranjevanja skritih ploskev oziroma robov je bil znan že v 60. letih 20. stoletja, ko je problem odstranjevanja skritih robov predstavil Sutherland [11] kot enega od nerešljivih problemov v računalniški grafiki. Takrat so bili v uporabi žični 3D modeli in v tem primeru je treba prikazati samo tiste robove, ki so vidni, oziroma določati presečišča robov in prikazati samo tiste dele robov, ki so vidni. Problem je bil takrat res nerešljiv, zaradi velikega števila presečišč med robovi, zaradi narave takrat razvitih algoritmov, ki so bili zelo neučinkoviti in slabe zmogljivosti računalnikov. Problem so uspešno rešili šele leta 1988 z uporabo učinkovitih algoritmov in uporabe vzporednih procesorjev [154]. Danes je seveda mogoče prikazovati tudi žični model z odstranjevanjem skritih robov. Žične modelirnike so kmalu zamenjali poligoni in odstranjevanje skritih ploskev z uporabo poligonov je na prvi pogled celo nekoliko lažje. Dovolj je, da poligone rišemo od zadaj naprej in vsak sprednji poligon s svojo površino prekrije zadnjega. Tak algoritem se imenuje slikarski algoritem (angl. Painter's algorithm) [155], ki je prikazan na sliki 11.1, kjer so najprej izrisani modri poligoni, nato rumeni in na koncu zeleni.



Slika 11.1. Slikarski algoritem

V splošnem seveda to ne deluje tako preprosto, saj se pogosto zgodi, da ni tako enostavne ločnice med posameznimi elementi. Lahko pride tudi do cikličnega prekrivanja poligonov, ko je en poligon pred drugim poligonom, drugi poligon je pred tretjim poligonom in tretji poligon je pred prvim poligonom. Zato so razvili algoritme z delitvenimi drevesi, kjer je bil vsak poligon v svojem prostoru (listu na delitvenem drevesu). Med temi algoritmi [156] je bil zelo popularen Warnockov algoritem, ki je v primeru cikličnega prekrivanja poligone razdelil in vsak del poligona umestil v svoj podprostor. Ultimativni algoritem za odstranjevanje skritih robov, imenovan Z-buffer [157], je glede na Wikipedijo predlagal Strasser [158], čeprav je v istem letu v svoji disertaciji to predlagal tudi Edwin Catmull [159]. Catmull, ki smo ga že spoznali pri deljenju površin Catmull-Clark, je precej pogosteje omenjen kot izumitelj tega algoritma. Večji problem je bila praktična uporaba algoritma, ki predvideva dodaten pomnilnik za shranjevanje koordinat z za piksle poligonov. Algoritem je pravzaprav slikarski algoritem, ki namesto poligonov razvršča piksle po globini. Na koncu je prikazan le tisti piksel, ki je najbližje gledišču. Leta 1974, ko je bil algoritem predlagan, je bil pomnilnik zelo dragocen, zato je trajalo kar nekaj časa do praktične uporabe. Druga težava je numerična ločljivost, saj se lahko zgodi, da zaradi zaokroževanja pride do numerične napake in je dejansko prikazan piksel, ki bi moral biti skrit. To lahko rešimo z večjo natančnostjo shranjene številke, zato so danes pomnilniki za Z-buffer narejeni tako, da je številka zapisana vsaj z 32 biti.



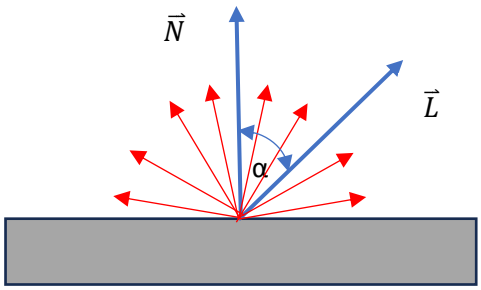
Slika 11.2. Z-buffer algoritem

Določene površine so lahko tudi prozorne in v takšnih primerih je treba prikazati tudi piksle v ozadju [160]. V ta namen se uporablja zapis barv RGBA, kjer črka A določa prosojnost, in tako omogočimo prikaz pikslov, ki so zakriti. Na sliki 11.2 je prikazan način uporabe algoritma Z-buffer. Ko gre za transparentne površine, je treba uporabiti zapis barv RGBA, sicer bodo na površini prikazane barve, ki so označene s številkami od 1 do 6. Odstranjevanje skritih površin je v obliki Z-buffer skupaj s pomnilnikom danes del grafične kartice in tako za odstranjevanje skritih površin v celoti skrbi grafična kartica računalnika.

11.2 Senčenje površin

Izraz senčenje prihaja iz slikarstva, kjer je senčenje ključnega pomena pri ustvarjanju oblik, definiranju svetlobe in sence ter pri podajanju različnih materialov in tekstur. Pravilno senčenje lahko poudari obliko predmeta, mu doda dimenzijo in mu pomaga k bolj realističnemu videzu. Enak namen ima senčenje površin v računalniški grafiki [161]. Senčenje površin je odvisno od materiala površine in svetlobe, ki pada na to površino. Svetloba ima običajno vir v svetlobnem telesu, ki je lahko različne intenzivnosti in barve, poleg tega je lahko virov svetlobe več in so razporejeni kjerkoli v prostoru. Takšni direktni viri svetlobe lahko povzročajo tudi učinek zrcalnega odboja na površini. Večino svetlobe direktnih virov svetlobe lahko obravnavamo kot vzporedne žarke svetlobe, ki se razpršijo po prostoru oziroma površini. Takšni razpršeni svetlobi rečemo difuzna svetloba in pri stiku s površino opazujemo difuzni odboj svetlobe. Poleg direktne svetlobe, ki izvira iz svetlobnega vira, je treba upoštevati tudi indirektni vir svetlobe, ki izvira iz odbojev svetlobe od sten in drugih objektov. Takšni indirektni svetlobi rečemo ambientna svetloba, ker je težko določiti natančen izvor svetlobe in se nanaša na splošno osvetlitev prostora. Ta svetloba je običajno konstantna in ne pade iz ene smeri. Ambientna svetloba ne ustvarja močnih kontrastov ali ostrih senc, temveč zagotavlja splošno svetlobo, ki je enakomerno prisotna v okolju.

Senčenje površin je mogoče izvesti dovolj preprosto z uporabo enostavnih enačb, ki določajo odboj svetlobe. Za difuzno svetlobo upoštevamo Lambertianov zakon odboja [162], kjer upoštevamo smer vira svetlobe **L** in normalo **N** površine, na katero padajo žarki, tako kot je prikazano na sliki. Skalarni produkt vektorjev **L** in **N** določa intenzivnost difuznega odboja. Pri tem je treba vedeti, da se žarki v točki odboja pri difuznem odboju odbijejo v vse smeri enakomerno in pri tem gledišče ni pomembno za izračun.



Slika 11.3. Difuzen odboj svetlobe

Pri tem ni pomembna samo smer proti viru svetlobe, ampak tudi jakost točkovnega vira svetlobe, ki jo označimo z I_p. Ker jakost svetlobe pada z oddaljenostjo, lahko to opredelimo s faktorjem f_{att}. Faktor f_{att} je odvisen od oddaljenosti površine od vira svetlobe, določene z d_L, in ga izračunano z enačbo 11.1.

$$f_{att} = \frac{1}{(d_L)^2} \tag{11.1}$$

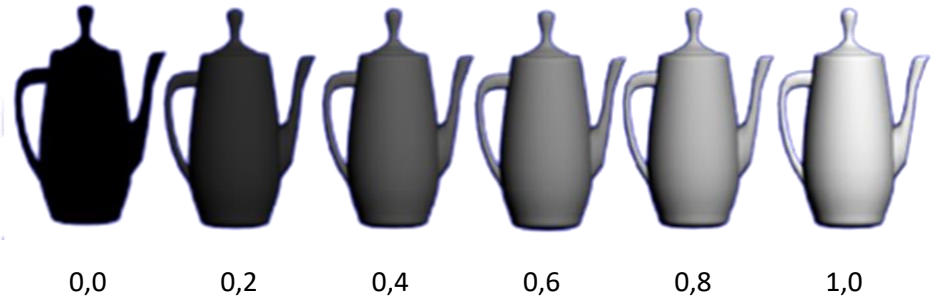
Intenzivnost difuznega odboja svetlobe lahko potem izračunamo z enačbo 11.2, kjer I_p predstavlja jakost svetila in k_d materialni faktor difuznega odboja, ki bo opisan v nadaljevanju.

$$I = f_{att} \cdot I_p \cdot k_d \cdot (\boldsymbol{N} \cdot \boldsymbol{L}) \tag{11.2}$$

Skalarni produkt lahko izračunamo tudi s produktom dolžin vektorjev in kosinusom kota α med vektorjema. Zato lahko privzamemo enotska vektorja N in L ter namesto skalarnega produkta zapišemo kosinus kota. Tako lahko jakost difuznega odboja zapišemo z enačbo 11.3.

$$I = f_{att} \cdot I_p \cdot k_d \cdot \cos\alpha \tag{11.3}$$

V enačbah 11.2 in 11.3 je tudi faktor k_d, ki je materialni faktor difuznega odboja svetlobe. Odboj svetlobe je namreč zelo odvisen od materiala površine, saj nekatere površine odbijajo svetlobo bolj kot druge. Ta faktor lahko določimo eksperimentalno za različne materiale. Na sliki 11.4 je primer vrednosti faktorja k_d in njegov učinek na intenzivnost odboja svetlobe.



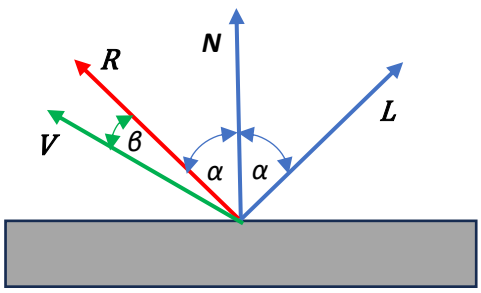
0,0 0,2 0,4 0,6 0,8 1,0

Slika 11.4. Materialni faktor difuznega odboja svetlobe k_d

Poleg difuzne svetlobe je treba upoštevati tudi ambientno svetlobo, kjer je vpadni kot svetlobe nepomemben, saj ta prihaja od povsod. Določiti je treba intenzivnost ambientne svetlobe I_a in podobno kot pri difuznem odboju faktor ambientnega odboja k_a, ki ima tako kot k_d vrednosti med 0 in 1 in je določen z eksperimenti. Nato lahko sestavimo jakost ambientnega in difuznega odboja ter jakost odboja svetlobe zapišemo z enačbo 11.4.

$$I = I_a \cdot k_a + f_{att} \cdot I_p \cdot k_d \cdot \cos\alpha \tag{11.4}$$

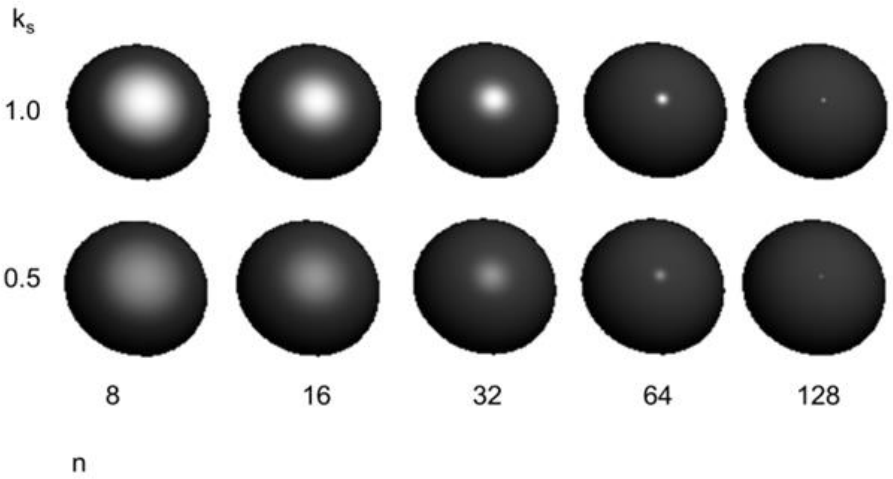
Direktna osvetlitev na površini pogosto povzroča odsev svetila. Simuliranje takšnega zrcalnega odboja (angl. Specular highlight) [163] je odvisno od vira svetlobe in gledišča. Na sliki 11.5 je normala površine določena z vektorjem **N**, smer proti viru svetlobe pa z vektorjem **L,** tako kot pri difuznem odboju. Privzamemo, da se pri zrcalnem odboju svetloba odbije z enakim kotom α, kot je padla na površino, vektor odboja pa je označen z **R**. Gledišče je označeno z vektorjem **V,** med odbojnim vektorjem in vektorjem gledišča pa je kot β.



Slika 11.5. Zrcalen odboj svetlobe

Tudi v tem primeru sta oblika in jakost odboja odvisna od materiala površine, kjer jakost odboja svetlobe na površini določa faktor k_s, prikazan na navpični osi slike 11.6. Od materiala ni odvisna samo jakost odboja svetlobe, ampak tudi velikost zrcalnega odboja svetlobe, ki je določena s potenco *n*, ki je prikazana na vodoravni koordinatni osi slike 11.6. Primer obeh materialnih podatkov lahko vidimo na sliki 11.6 in seveda je treba te podatke določiti za vsak material eksperimentalno. Nato lahko zapišemo zrcalni odboj v obliki enačbe 11.5 oziroma z enačbo 11.6, kjer skalarni produkt vektorjev zamenjamo s kosinusom kota med njima.

$$I_s = I_p \cdot k_s \cdot (\boldsymbol{R} \cdot \boldsymbol{V})^n \tag{11.5}$$



Slika 11.6. Materialni podatki za zrcalen odboj svetlobe

$$I_s = I_p \cdot k_s \cdot \cos^n \beta \tag{11.6}$$

Celoten odboj svetlobe lahko nato zapišemo z enačbo 11.7, kjer je upoštevan ambienten, difuzen in zrcalen odboj svetlobe.

$$I = I_a \cdot k_a + I_p \cdot (f_{att} \cdot k_d \cdot \cos\alpha + k_s \cdot \cos^n \beta) \tag{11.7}$$

V enačbi 11.7 ni nobene informacije o barvi površine, ampak le o jakosti odboja svetlobe za izbrano točko površine. Če želimo določiti barvo v določeni točki površine, je treba enačbo uporabiti za vsako od osnovnih barv. Pri tem je treba določiti še barvo svetila in barvo površin. Zato za svetila uporabimo jakost svetila v določeni barvi in enako velja za ambientno barvo, ki je odvisna od barve okolice (na primer sten prostora). Barvo površine označimo s podatkom *O*, ki za vsako barvo pove, v kolikšni meri določene barve se odbije od površine. Tako dobimo enačbo 11.8 za jakost odboja rdeče barve, enačbo 11.9 za jakost odboja zelene barve in enačbo 11.10 za jakost odboja modre barve. Pri tem vidimo, da je zrcalni odboj večinoma odvisen od barve svetila in ne od barve površine, na kateri se odboj pojavi.

$$I_R = I_{aR} \cdot k_a \cdot O_{aR} + I_{pR} \cdot (f_{att} \cdot k_d \cdot O_{dR} \cdot \cos\alpha + k_s \cdot \cos^n \beta) \tag{11.8}$$

$$I_G = I_{aG} \cdot k_a \cdot O_{aG} + I_{pG} \cdot (f_{att} \cdot k_d \cdot O_{dG} \cdot \cos\alpha + k_s \cdot \cos^n \beta) \tag{11.9}$$

$$I_B = I_{aB} \cdot k_a \cdot O_{aB} + I_{pB} \cdot (f_{att} \cdot k_d \cdot O_{dB} \cdot \cos\alpha + k_s \cdot \cos^n \beta) \tag{11.10}$$

Te enačbe so precej splošne in različni modeli morda ne določajo odboja tako natančno, medtem ko drugi, bolj kompleksni modeli, uporabljajo še kompleksnejše modele barvnega odboja [163]. Običajno imamo na voljo podobne tri enačbe za barvni sistem RGB, vendar se modeli še bolj razlikujejo po tem, kje te enačbe izračunavamo.

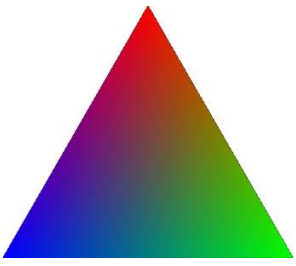
Poligonsko senčenje

Najenostavnejši model senčenja je poligonsko senčenje, pri katerem enačbe računamo za celoten poligon. Glede na dejstvo, da je poligon ravninski objekt, v enačbah upoštevamo normalo poligona in z izračunano barvo pobarvamo poligon tako, kot je prikazano na sliki 11.7. Hitro opazimo, da je v tem primeru nemogoče učinkovito simulirati zrcalni odboj, saj mora cel poligon odsevati svetilo. Še danes je ob veliki zmogljivosti računalnikov težko zagotoviti tako male poligone na modelu, da bi s poligonskim senčenjem dobili kakovostno sliko. Pri poligonskem senčenju je vsak poligon narisan z drugačno barvo, zato bodo tudi ob uporabi zelo malih poligonov še vedno na sliki vidne meje poligonov. To je seveda danes neuporabno za senčenje 3D modelov.

Slika 11.7. Poligonsko senčenje sfere

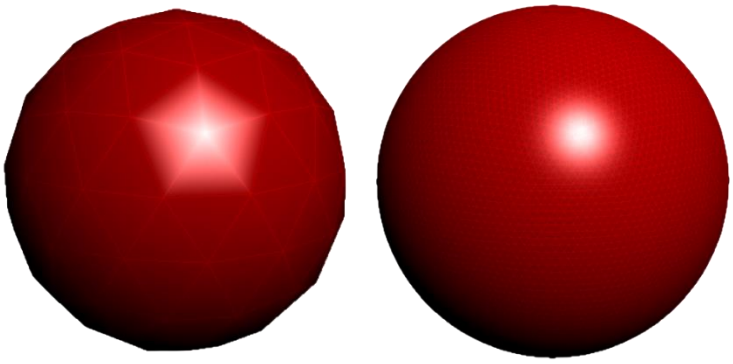
Gouraudovo senčenje [164**]**

Henri Gouraud je leta 1971 v okviru doktorske disertacije na Univerzi v Utahu izdelal tehniko senčenja, ki jo danes imenujemo Gouraudovo senčenje. Osnovna ideja je, da za točko računanja enačb (11.8, 11.11 in 11.10) izberemo oglišče poligonov. Če je poligon trikotnik, določimo normalo v vsakem oglišču trikotnika. Ta normala ni odvisna samo od enega poligona, ampak izračunamo srednjo vrednost normale glede na vse poligone, ki si delijo to oglišče. Zato v vsakem oglišču poligona dobimo drugačno jakost svetlobe oziroma drugačno barvo senčenja, čeprav je poligon ploskoven element. Tako dobljene barve nato interpoliramo po celotni površini poligona. Če uporabimo osnovne barve za senčenje trikotnika v vsakem oglišču, dobimo rezultat, prikazan na sliki 11.8.



Slika 11.8. Interpolacija barve po površini glede na rezultate v oglišču

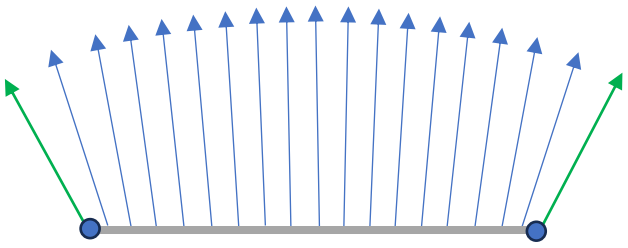
Na sliki 11.9 je prikazana sfera, senčena z Gouraudovim algoritmom. Ker se enačbe računajo v ogliščih, je zrcalni odboj vedno prikazan v oglišču in je odvisen od velikosti poligonov. Na sliki 11.9 vidimo razliko senčenja, če imamo velike poligone na levi strani in drobne poligone na desni strani. Za kakovostno simuliranje zrcalnega odboja je treba modelirati z velikim številom poligonov, ki si jih težko privoščimo za prikazovanje kompleksnega 3D modela.



Slika 11.9. Gouraudovo senčenje sfere

Kljub temu je Gouraudovo modeliranje veliko bolj kakovostno kot poligonsko senčenje in kljub lepšemu prikazu ni veliko bolj zahtevno od poligonskega senčenja. Zato so modelirniki še v 90. letih 20. stoletja uporabljali Gouraudov način senčenja za prikazovanje modela med modeliranjem.

Phongovo senčenje [165**]**
Leta 1975 je tudi Bui Tuong Phong na Univerzi v Utahu izdelal doktorsko disertacijo. V okviru svoje disertacije je predstavil novo tehniko senčenja površin, ki jo danes imenujemo Phongovo senčenje. Osnovna ideja tega senčenja je v tem, da svetlobnih enačb ne računamo le v ogliščih poligona, ampak v vsakem pikslu slike. Pri tem je treba določiti smer normale v vsaki točki slike in to je izvedeno tako, da se smer normale interpolira med oglišči poligona. Grafično je takšna interpolacija podobna tej na sliki 11.10, kjer normale interpoliramo po robu med oglišcema.



Slika 11.10. Interpoliranje normal med oglišči

Kadar imamo trikotni poligon z oglišči A, B in C, lahko interpolacijo normale za poljubno točko P določimo z enačbo 11.12, kjer so $\boldsymbol{N_A}$, $\boldsymbol{N_B}$ in $\boldsymbol{N_C}$ normalni vektorji v oglišcih trikotnika, w_{AP}, w_{BP} in w_{CP} pa so uteži za poljubno točko P.

$$\boldsymbol{N_P} = w_{AP} \cdot \boldsymbol{N_A} + w_{BP} \cdot \boldsymbol{N_B} + w_{CP} \cdot \boldsymbol{N_C} \tag{11.12}$$

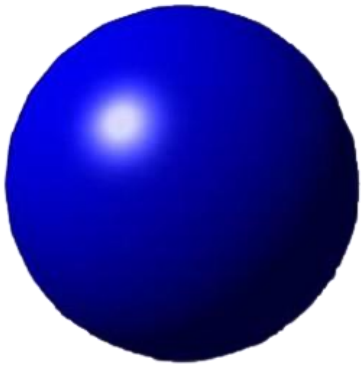
Uteži *w* so odvisne od lokacije točke na poligonu in oddaljenosti točke od oglišč. Izračunamo jih z enačbami 11.13, 11.14 in 11.15, kjer so d_{AP}, d_{BP} in d_{CP} razdalje od točke P do ustreznega oglišča trikotnika.

$$w_{AP} = \frac{d_{AP}}{(d_{AP}+d_{BP}+d_{CP})} \tag{11.13}$$

$$w_{BP} = \frac{d_{BP}}{(d_{AP}+d_{BP}+d_{CP})} \tag{11.14}$$

$$w_{CP} = \frac{d_{CP}}{(d_{AP}+d_{BP}+d_{CP})} \tag{11.15}$$

Ko so določene normale, je treba v vsakem pikslu izračunati barvo piksla z uporabo enačb 11.8, 11.9 in 11.10. To predstavlja precejšnje število izračunov in zahtevnost senčenja ni več odvisna samo od števila poligonov, ampak tudi od ločljivosti končne slike. Leta 1975 je bila uporaba takšnega senčenja precej omejena z zmogljivostjo računalniške strojne opreme. Na sliki 11.11 je prikazana sfera, senčena s Phongovim senčenjem. Pri tem gre za enak model z enakimi poligoni, kot je sfera na sliki 11.7. Vidimo, da je sfera, senčena s Phongovim senčenjem, veliko lepša in grobost poligonov sploh ni opazna. Ravne odseke poligonov vidimo le na robovih sfere. Tudi zrcalni odboj je zelo kakovostno renderiran in ni odvisen od oglišča kot pri Gouraudovem senčenju. Središče zrcalnega odboja je lahko na poljubnem mestu v odvisnosti od položaja svetila.

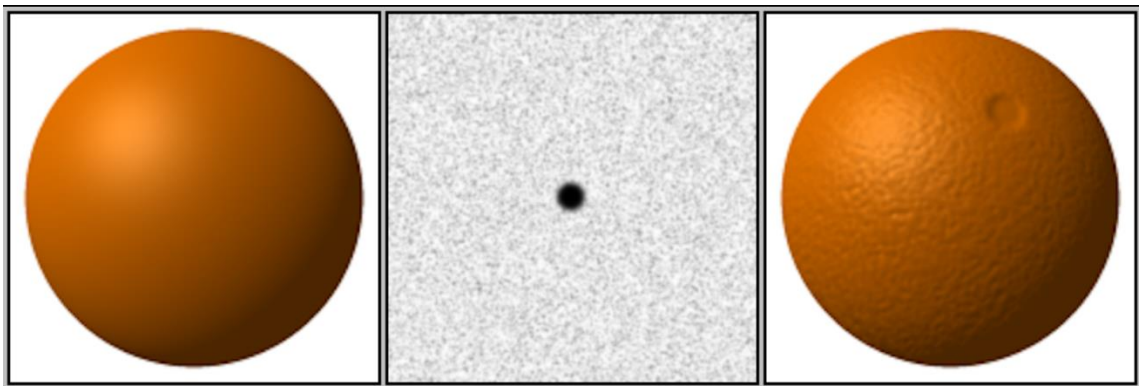


Slika 11.11. Phongovo senčenje sfere

Glede na Moorov zakon, ko se zmogljivost računalnikov podvoji vsaki dve leti, je strojna oprema postala dovolj zmogljiva in danes Phongovo senčenje deluje v realnem času že na manj zmogljivi strojni opremi (na primer mobilni telefoni in tablice). Pri običajnem modeliranju, ko z miško ali s prsti vrtimo model sem in tja, se scena ves čas senči s Phongovim senčenjem in le še redko imamo na voljo izbiro Gouraudovo senčenje. Za popolno realistično upodabljanje modelov Phongovo senčenje ne zadostuje in je treba izbrati natančnejši način senčenja površin, saj je zanemarjenih kar nekaj značilnosti površine (na primer hrapavost, lom svetlobe itd.).

Preslikava neravnin [166]

Phongovo senčenje je dovolj kakovostno senčenje za hitro prikazovanje prostorskih modelov v okviru računalniško podprtega konstruiranja, kjer so modeli v večini primerov del nežive narave. Phongovo senčenje žive narave je precej nerealistično, saj so v naravi površine redko popolnoma gladke. Zato so za potrebe računalniških vizualizacij razvili metodo senčenja s preslikavo neravnin (angl. Bump mapping), ki naredi površine bolj naravne. V ta namen se kreira površina neravnin, ki odbijajo svetlobo v različnih smereh. Namesto da interpoliramo smer normal med oglišči poligona, je smer normal odvisna od zemljevida neravnin. Na sliki 11.12 vidimo na levi strani osnovno geometrijo objekta, na sredini pa načrt neravnin. Končno senčenje je prikazano na desni strani slike 11.12, kjer dobimo realistično predstavitev objekta s preslikavo načrta neravnin na osnovni geometriji. Pri tem je število poligonov lahko sorazmerno malo, tako kot pri Phongovem senčenju.



Slika 11.12. Senčenje s preslikavo neravnin [166]

Takšno renderiranje se še danes uporablja v računalniških igrah, kjer strojna oprema ne zmore bolj zahtevnega renderiranja s sledenjem žarkom. Na tak način so liki ob sorazmerno preprostem načinu renderiranja prikazani zelo realistično. Kot rečeno je to manj uporabno za računalniško podprto konstruiranje, razen če bi želeli senčiti naravne materiale. Za končno renderiranje izdelka zato raje uporabljamo senčenje s sledenjem žarku, opisano v nadaljevanju.

11.3 Realistično renderiranje izdelkov

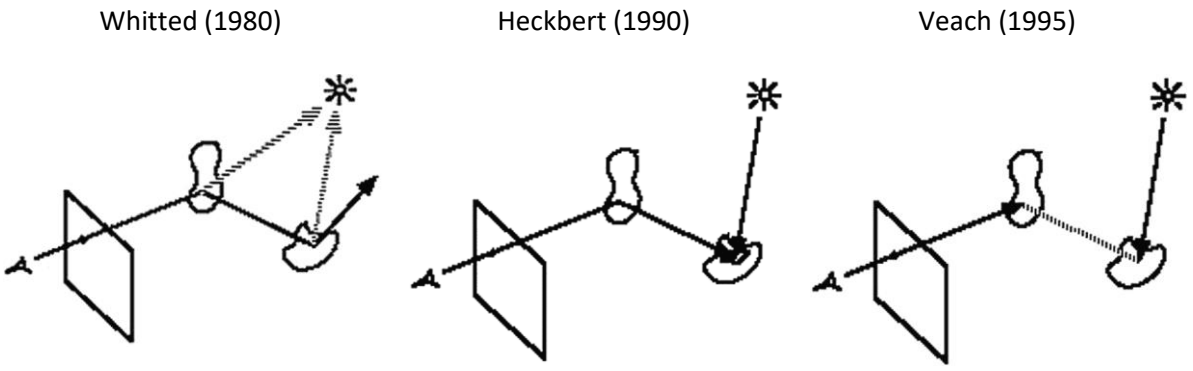
Zgoraj opisane metode senčenja lahko precej lepo simulirajo odboj svetlobe od različnih površin. Ti načini senčenja precej dobro delujejo za kovinske materiale, kjer svetloba ne prodre pod površino objekta. Za kovinske materiale je značilno, da vidna svetloba nima dovolj energije za preboj površine in elektroni v kovinskih materialih absorbirajo ali odbijejo vidno svetlobo. V naravi in v industrijskih izdelkih pa včasih uporabimo materiale, ki niso kovinski. Najbolj značilen tak material je steklo. Elektroni v steklu ne morejo absorbirati vidne svetlobe, saj je energija vidne svetlobe prevelika za elektrone stekla. To ne velja za vso svetlobo, saj na primer ultravijolična svetloba kljub še večji energiji povzroči resonanco elektronov in steklo takšno svetlobo odbije. Svetloba z višjo energijo, na primer rentgenski žarki, lahko prehaja tudi skozi kovine. Zato se pri renderiranju običajno ukvarjamo s simuliranjem vidne svetlobe in lahko za različne materiale določimo stopnjo prosojnosti za vidno svetlobo. Pri prehajanju svetlobe skozi material pride zaradi spremembe hitrosti svetlobe v različnih materialih do loma svetlobe in tudi ta učinek je treba simulirati, če želimo realistično upodobiti objekt na sceni. Poleg vpoja, odboja in loma svetlobe se v naravi pojavi tudi pojav fluorescenčne svetlobe. Fluorescenčna svetloba se pojavi, ko material reagira s svetlobo svetlobnega vira in odda svetlobo pod drugim kotom in v drugi barvi. Ta pojav le redko simuliramo z metodami računalniške grafike, vendar če želimo realistično sceno, je treba upoštevati tudi to. Za simulacijo svetlobe z namenom renderiranja je bilo razvitih kar nekaj metod, ki bodo na kratko predstavljene v nadaljevanju.

Sledenje žarkom (angl. Raytracing) [167]

Ideja sledenja svetlobnim žarkom na sceni je že zelo stara in jo je prvi uporabil slikar Albrecht Dürer v šestnajstem stoletju. Celotno sceno je razdelil na okenca in opazoval, kaj se skozi posamezno okence vidi. Analogno lahko seveda sceno narišemo z računalnikom, kjer pogledamo oziramo pošljemo žarek na sceno skozi vsak piksel slike. Ta piksel prikazuje tisti delec scene, ki ga je zadel žarek. V začetku 70. let 20. stoletja je bilo kar nekaj poskusov ustvarjanja slike na tak način. Izkazalo se je, da ni dovolj zgolj uporabiti enega žarka od piksla do površine. Zato je leta 1978 Turner Whitted predstavil posnetek renderirane scene z algoritmom sledenja žarku z rekurzivnimi žarki. Na [168] je opis s filmom, ki ga je Whitted pripravil za konferenco Siggraph, na kateri je predstavil svoj algoritem.

Whitted je v svojem algoritmu poslal žarek skozi vsak piksel na sceno, in ko je našel presečišče žarka z objektom na sceni, je kreiral nove žarke in za vsak žarek uporabil rekurzivno funkcijo. Pri rekurziji v programskem jeziku v določeni funkciji izvedemo isto funkcijo (funkcija kliče samo sebe). Pri tem je uporabljen sklad (angl. stack), ki omogoča shranjevanje stanja. Ko po rekurzivnih klicih pridemo do konca, ki je v primeru algoritma sledenja žarku vir svetlobe, se z rezultatom vračamo nazaj do prvega klica. Tako z uporabo svetlobnih enačb podobno kot pri Phongovem senčenju dobimo končno barvo piksla. V nasprotju s Phongovim senčenjem lahko z algoritmom sledenja žarku simuliramo tudi lom svetlobe, sence zaradi drugih objektov, odsev objektov okolice na površini itd. To pomeni, da lahko renderiramo sceno tako, kot je izdelek, prikazan v naravi. To seveda ni zastonj in algoritem zahteva izračun velikega števila presečišč in svetlobnih enačb. Čeprav je Whitted uporabil enostavne površine (ravnine in sfere), s katerimi je računanje presečišč dokaj preprosto, je renderiranje trajalo precej časa (merjeno v tednih), zato je imel precejšno časovno stisko za oddajo prispevka na konferenco. Čeprav je algoritem deloval odlično v primerjavi s senčenjem v tistem času, si je bilo težko zamišljati, da bo takšno senčenje sploh kdaj mogoče v realnem času. Za vsako barvo piksla slike bi potreboval takratni superračunalnik Cray, da bi lahko renderiral 60 slik na sekundo. To pomeni vsaj tri superračunalnike za en piksel slike. To je bil razlog, da je bila tehnika sledenja žarku nekaj časa prisotna samo v akademskih krogih, nato so jo začeli počasi uporabljati za produkcijo s poenostavitvami in dolgimi časi renderiranja. Zato je izdelava filmov, v katerih so bile računalniško generirane scene, včasih trajala dolga leta. Po štiridesetih letih od razvoja algoritma je danes na voljo strojna oprema (grafične kartice), ki renderira sceno v realnem času. To pomeni, da naredijo vsaj 60 slik na sekundo v visoki ločljivosti. Postopek je precej podoben, kot ga je predlagal Whitted pred 40 leti, saj izračune za vsak piksel dela en ali več računalniških jeder. Pri tem so današnja jedra približno tisočkrat hitrejša od takratnih superračunalnikov.

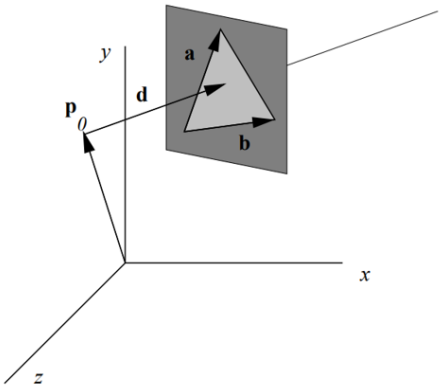
Kljub veliki hitrosti smo še vedno omejeni z geometrijo objektov, zato algoritem sledenja žarku deluje sorazmerno dobro za poligonske površine ali enostavne implicitne površine (na primer sfere). Whitted je uporabil sledenje žarku tako, da je začel žarek na mestu gledišča, ga poslal na sceno, sledil žarkom in na koncu dodal še žarke do svetila. Pozneje so metodo prilagodili tako, da je bila začetna točka žarka na svetilu in so opazovali pot žarkov do gledišča. Tretja možnost je kombinacija obeh prej navedenih načinov tako, da se žarki začnejo tako na gledišču kot na svetilu. V vseh primerih je pomembno opazovati presečišča žarkov z objekti na sceni. Na sliki 11.13 so prikazani trije načini sledenja žarku. Whitted je uporabil žarke od gledišča do svetila, Heckbert od svetila do gledišča in Veach kombinacijo obeh poti. Ne glede na to, iz katere smeri pošljemo žarek, je treba določiti presečišče žarka z objektom na sceni. Matematično to pomeni, da moramo določiti presečišče premice žarka s površino na sceni.



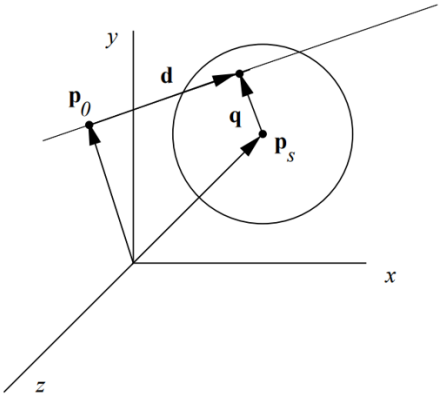
Slika 11.13. Tipi metod sledenja žarku [169]

Na sliki 11.14 sta prikazani dve vrsti enostavnih presečišč žarka. Na levi strani je presečišče žarka s poligonom, ki ga preprosto določimo s presečiščem premice in ravnine poligona. Še bolj pomembno je, da lahko iz računanja izločimo vse poligone, ki jih žarek zgreši. V ta namen je kreiran pravokotnik mej poligona, in če je žarek zunaj teh mej, je računanje presečišča nepotrebno. Kadar je treba določiti presečišče, je tudi to določanje zelo preprosto. Nato sledi še določanje svetlobnih enačb na mestu presečišča. Podobno velja za sfero, ki jo lahko opišemo z implicitno enačbo in podobno lahko določimo, ali bo žarek zadel sfero in nato precej preprosto določimo presečišče žarka s sfero. Zaradi tega je veliko začetnih predstavitev renderiranja s sledenjem žarkom vključevalo sfere in poligone.

Presečišče žarka s poligonom



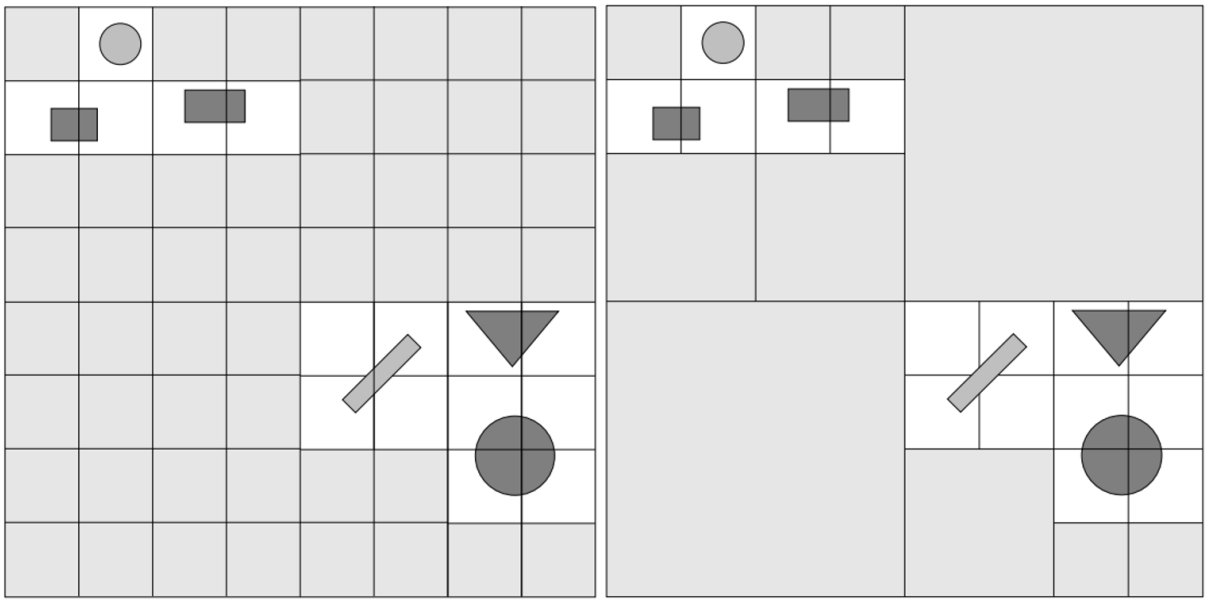
Presečišče žarka s sfero



Slika 11.14. Enostavna presečišča pri metodi sledenja žarku

Tudi danes se glede tega ni spremenilo veliko, saj je računanje presečišč s parametričnimi površinami, ki so lahko opisane s kubičnimi polinomi (ali celo s polinomi višjega reda), še vedno precej zahtevno. Zato bo vsaj še nekaj let metoda sledenja žarkov v realnem času zahtevala poligonske površine. Za računalniške igre to ni velika težava, saj so scene običajno zgrajene s poligonskimi mrežami. Za prostorske in površinske modelirnike to pomeni, da bodo programi še dolgo morali kreirati poligonsko mrežo površin pred prikazovanjem na zaslonu. Z razvojem grafične strojne opreme je mogoče pričakovati, da bo v naslednjih desetletjih mogoče renderirati tudi površine z višjo stopnjo polinomov v realnem času. Današnja strojna oprema je tako zmogljiva, da lahko renderira scene visoke ločljivosti v zelo kratkem času. Za računalniške igre je treba izdelati vsaj šestdeset slik na sekundo, zato so postale grafične kartice zelo zmogljivi računalniki. Strojna oprema se razvija z velikimi koraki in danes so takšne grafične kartice, ki omogočajo renderiranje s sledenjem žarkom v realnem času, vgrajene tudi v mobilne telefone.

Razlog za veliko hitrost renderiranja je po eni strani dejstvo, da se zmogljivost računalnikov podvoji vsaki dve leti. Zmogljivost računalnika v modulu Apollo, s katerim so poleteli na Luno, je danes primerljiva s čipom, vgrajenim v USB-C kabel. Drugi razlog v hitrosti renderiranja je v tem, da lahko celotno sceno razdelimo na posamezne domene tako, kot je prikazano na sliki 11.15. Za vsako domeno lahko uporabimo en procesor oziroma računalniško jedro in celotna scena se obdeluje vzporedno z velikim številom jeder. Grafična kartica Nvidia GeForce RTX 4090 tako deluje s 16.384 jedri. To pomeni, da lahko sceno razdelimo na 16.384 domen. Če imamo 4K zaslon z ločljivostjo 3840 x 2160 pikslov, mora vsako jedro računati presečišča za 506 pikslov, kar pomeni domeno velikosti približno 22 x 23 pikslov. To je še vedno veliko žarkov, vendar je hitrost računanja nepredstavljivo hitra, saj kartica zmore 82 bilijonov računskih operacij v sekundi (82 TFLOPS). Vsako jedro lahko izvede 5 milijard operacij na sekundo. Vsa jedra na kartici komunicirajo med seboj, saj je včasih treba izmenjati podatke, ko prehaja žarek iz ene domene v drugo. Ker je vse skupaj v istem čipu, je tudi ta hitrost komunikacije izredno velika.



Slika 11.15. Razdelitev scene na domene

Renderiranje z metodo sledenja žarkom daje izredno dobre rezultate. Na sliki 11.16je prikazana primerjava z renderiranim modelom in fotografijo 3D miške. Model je bil izdelan v okviru seminarske naloge pri predmetu 3D modeliranje na Fakulteti za strojništvo, Univerze v Mariboru. Pri tem modelu sicer ne gre za transparentne ali polirane površine, kjer lahko z metodo sledenju žarkov simuliramo tudi učinke loma svetlobe in zrcaljenja okolice na površini izdelka.

Renderiran model Fotografija miške

Slika 11.16. Primerjava med renderiranim modelom in fotografijo

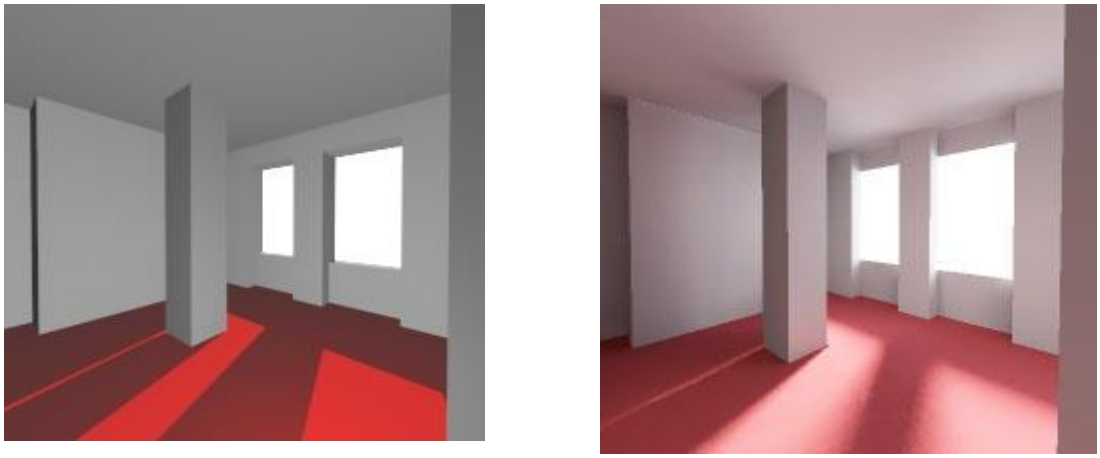
Sodobni programi za renderiranje omogočajo prikaz površin, ki so enake kot v naravi. Pri tem je veliko truda vloženega v naravno modeliranje fluidov in površin, ki niso gladke (na primer dlaka pri živalih). Posledično postajajo risani filmi, kjer nastopajo živali, bolj podobni dokumentarnim posnetkom. V filmski industriji je postalo lažje modelirati naravo kot posneti naravo, saj če želimo v filmu prikazati orkan na morju, je veliko težje izdelati kakovosten posnetek v težkih razmerah kot modelirati in renderirati sceno na računalniku.

Metoda sledenja žarkom je postala že običajno orodje, ki je vključeno v vsak CAD program, s katerim lahko zelo hitro izdelamo kakovostno sliko modela. Če uporabimo posebne programe za renderiranje, imamo na voljo še večjo paleto materialov, tipov in tehnik osvetlitev ter različnih dodatkov, ki jih lahko uporabimo za kreiranje scene, na kateri predstavimo izdelek. Pogosto ne zadošča več samo ena slika, ampak je za učinkovito predstavitev izdelka treba izdelati videoposnetek.

Sevalnost (angl. Radiocity) [170**]**

Metoda sledenju žarkom je omejena s številom žarkov, ki jih uporabimo na sceni. V naravi so fotoni zelo majhni in nemogoče bi bilo simulirati gibanje posameznih fotonov, da bi dobili še kakovostnejšo sliko scene. Kljub temu obstajajo tudi poskusi s simuliranjem fotonov v metodi, imenovani sledenje fotonov (angl. Photon Mapping) [171]. Sevalnost oziroma radiosity je metoda, s katero lahko metodo sledenju žarkom izboljšamo tako, da namesto sledenja posameznim fotonom simuliramo sevanje površin in sledimo energiji sevanja med posameznimi površinami na sceni. Rezultat je slika, kjer so prehodi veliko mehkejši in namesto trdih senc objektov dobimo bolj mehke sence objektov. Razlika je lepo vidna na enostavni sceni na sliki 11.17, kjer je na levi strani scena renderirana z metodo sledenja žarkom, na desni strani pa je scena renderirana s sevalno metodo.

Sledenje žarkom Sevalnost

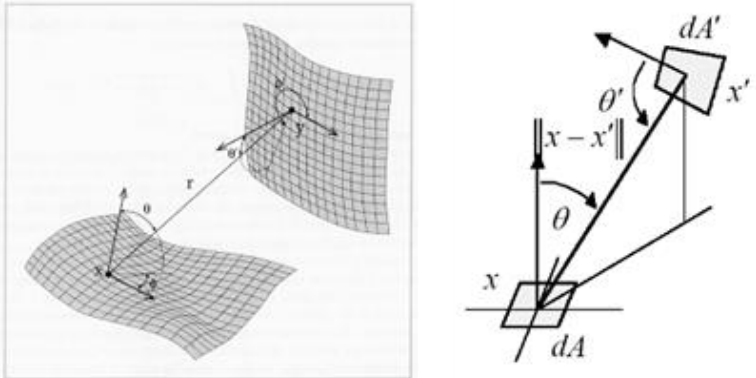


Slika 11.17. Primerjava med sledenjem žarkom in sevalnostjo

Ideja sevalnosti je reševanje sevalnih enačb za vsako površino na sceni. Sevalna enačba je zapisana v enačbi 11.16, kjer z B_i izračunamo sevanje površine, ki je vsota emisivnosti površine E_i in vsote prispevkov sosednjih površin. Prispevek sosednje površine je določen s sevanjem površine B_i in oblikovnega faktorja F_{ij}. Vsota prispevkov sosednjih površin je pomnožena z absorbcijskim faktorjem ρ_i, ki določa stopnjo absorbcije površine.

$$B_i = E_i + \rho_i \sum B_j F_{ij} \tag{11.16}$$

V zgornji enačbi je najtežje določiti faktor oblike površine F_{ij}, saj je treba v tem faktorju upoštevati medsebojni položaj obeh površin tako, kot je prikazano na sliki 11.18.



Slika 11.18. Določanje faktorja oblike površine

Prvi korak pri renderiranju s sevalnostjo zahteva, da vse površine na sceni spremenimo v mrežo površinskih končnih elementov, za katere poznamo integralske rešitve diferencialnih enačb. Ker je scena običajno modelirana s parametričnimi površinami ali s poligoni, ni težko izdelati mreže končnih elementov za poljubno sceno, ki jo želimo renderirati. Površine na sceni vsebujejo tudi materialne parametre, ki omogočajo izračune sevalnosti. Ko določimo vire svetlobe, je mogoče kreirati sistem linearnih enačb, s katerimi je definirana medsebojna odvisnost površin. Rezultat dobimo z rešitvijo tega sistema enačb. To pomeni, da je takšno renderiranje pravzaprav numerična simulacija, ki je lahko računsko precej intenzivna. Posebej zato, ker je treba določene detajle na sceni diskretizirati z zelo majhnimi končnimi elementi, da dobimo kakovostne rezultate. Glede na vedno večje strojne zmogljivosti je mogoče pričakovati, da bo tudi s to metodo kmalu mogoče renderirati scene v realnem času.

11.4 Uporaba renderiranja v praksi

Enostaven 3D model lahko danes upodobimo v realistični obliki na relativno preprost način. Skoraj vsak CAD modelirnik vsebuje orodje za realistično upodobitev, kjer je mogoče to narediti s samo nekaj kliki. Nekoliko bolj zahtevno je renderiranje izdelka v naravnem okolju, saj zgolj uporaba slike za ozadje običajno ne deluje najbolj prepričljivo. Pogosto se namreč zgodi, da je že zelo težavno izbrati enako osvetlitev modela, da bo vir svetlobe skladen s tistim na sliki. Prav tako bi morali okoliški objekti odsevati na površini izdelka. Zato renderiranje pogosto zahteva kreiranje celotne scene, na katero dodamo vse dodatne elemente v 3D obliki. V ta namen so določeni programi razvili posebna orodja, s katerimi oblikujemo celotno sceno, v katero umestimo želen izdelek.

Najbolj znano komercialno orodje za renderiranje je Renderman [172], ki ga razvijajo v podjetju Pixar in je zato vodilno orodje v filmski industriji. Seveda je to orodje primerno tudi za renderiranje CAD izdelkov, saj je mogoče uvoziti modele v številnih formatih. Zaradi zahtevnosti renderiranja videoposnetkov programska oprema zahteva zmogljivo strojno opremo in uporabimo lahko tudi t. i. render farme, kjer sliko hkrati renderira veliko povezanih grafičnih kartic. Zato program deluje na različnih računalniških platformah.

Zelo popularen program za renderiranje je Keyshot [173], saj je delo s programom zelo preprosto. Posebej privlačno je dejstvo, da lahko v Keyshot uvozimo neposredno tudi modele iz programov Solidworks, NX, Rhinoceros in še nekaterih drugih modelirnikov. Običajno programi za renderiranje zahtevajo zapis modela v kakšni prenosljivi poligonski obliki. Podobno popularen je tudi program Vray [174], tudi zaradi neposredne vključitve v program Rhinoceros. Vsi zgoraj našteti programi so plačljivi in licence se plačujejo mesečno ter v odvisnosti od uporabe strojne opreme. Na voljo so tudi odprtokodni oziroma prosto dostopni programi, med katerimi je najbolj znan program Povray [175]. Delo s tem programom je precej kompleksno in zahteva veliko izkušenj. Precej popularen za renderiranje je tudi prosto dostopen program Blender [74], ki omogoča zelo kakovostno renderiranje slik in videoposnetkov. Odprtokodnih in plačljivih programov za renderiranje je še veliko več in zgoraj so našteti samo nekateri najbolj znani.

12 Animacije

Včasih sta za predstavitev izdelka zadoščali slika in tehnična dokumentacija izdelka. Danes je treba skoraj vedno narediti predstavitev izdelka v delovanju ali vsaj videoizdelka tako, da ga lahko pogledamo iz vseh smeri. Za pogled iz vseh strani je izdelek pogosto predstavljen na spletnih straneh v obliki 3D modela, ki ga lahko uporabnik sam vrti. Ko pa gre za nekoliko bolj kompleksen izdelek in je treba pokazati tudi njegove funkcije, je to nekoliko težje izvedljivo, zato se običajno izdela renderiran videoposnetek izdelka. Pri tem imamo na voljo različne načine za izdelavo videoposnetka in koristno je upoštevati tudi bogate izkušnje, ki jih ima za izdelavo videoposnetkov filmska industrija.

12.1 Animacija ali simulacija

Čeprav so poskusi kreiranja gibajočih slik stari že nekaj tisoč let, so šele okoli leta 1800 odkrili fenomen vztrajnosti vida, kjer lahko zaporedje sličic dojamemo kot gibanje. Najbolj preprost primer tega fenomena smo nekateri spoznali že v otroštvu, ko na dva zaporedna lista v beležnici narišemo nekoliko spremenjeno sceno. Tako lahko na prvi list papirja narišemo figuro kovača z dvignjenim kladivom, na drugi pa s kladivom na nakovalu. Ko ta dva lista hitro menjujemo, lahko to dojamemo kot animacijo kovača, ki udarja po nakovalu. Namesto dveh sličic lahko gibanje predstavimo z večjim številom sličic, na katere narišemo vse vmesne korake. Tako lahko v beležnici z dovolj listi narišemo lepo tekoč risani film, za katerega je treba uporabiti nekoliko spretnosti, da dovolj hitro spuščamo posamezne liste. V devetnajstem stoletju so izdelali kar nekaj različnih mehanskih naprav za prikazovanje gibajočih slik. Nekatere naprave so imele slike prikazane na listih, ki so se dovolj hitro vrteli, nato so naredili mehanske naprave s premikajočimi figurami na sceni, ki se je vrtela. Takšna naprava se imenuje Zoetrope [176] in je še danes aktualna za kreiranje ponavljajočih scen. Pri tej napravi, ki jo lahko opazujemo na video posnetku [177], je bilo treba natančno določiti vrtilno hitrost, da dobimo želeno ponavljajoče gibanje na sceni. Danes lahko to brez težav naredimo z elektromotorjem, ki mu prilagajamo vrtilno hitrost, dokler ne dosežemo želenega učinka.

Po izumu fotografije in filmskih trakov so se gibajoče slike projicirale na platno in po prvih dokumentarnih in igranih filmih, kjer so s kamero zajemali slike v gibanju, so se kmalu pojavili animirani filmi. Osnovna ideja je bila enaka kot pri omenjeni animaciji z beležnico. Na vsako sličico filma je bil posnet delček gibanja in pri projiciranju te slike zaživijo. Kmalu so ugotovili, da ni treba risati znova cele slike, ampak so bili posamezni elementi slike narisani na prozorne folije v nivojih. Tako je ozadje lahko ostalo enako in objekti so se premikali po sceni v skladu s scenarijem.

Ti izumi, povezani z animiranimi filmi, so si sledili z velikim zamahom v prvi polovici dvajsetega stoletja. Čeprav še niso obstajale barvne fotografije, so že leta 1920 projicirali barvni film tako, da so pobarvali sličice na filmu. Kmalu so sliki dodali zvok in hitro so začeli delati z dejanskim barvnim filmom. Tudi tehnologija snemanja se je razvijala in izdelani so bili kompleksni stroji za premikanje kamere, da je bilo možno ustvariti učinek premikov v globino. Večino razvoja na področju animiranih filmov gre pripisati podjetju Walt Disney Company [178].

V osemdesetih letih dvajsetega stoletja se je začela izdelava animiranih filmov z uporabo računalnika. Pionirsko delo gre pripisati Johnu Lasseterju [179], ki je kariero začel pri Walt Disneyju in se nato preselil v LucasFilm in potem v Pixar. V tistem času je bil LucasFilm del podjetja Walt Disney. Nato je Steve Jobs kupil oddelek za grafiko podjetja LucasFilm in ustanovil podjetje Pixar. Danes je Pixar spet del podjetja Walt Disney. V Pixarju je Lasseter izdelal kratek risani film Luxo Jr. z uporabo računalnika. Film je bil predstavljen na konferenci računalniške grafike SIGGRAPH leta 1986, ki je bil nominiran za oskarja v kategoriji kratkih animiranih filmov ter tako postal prvi CGI (angl. Computer Generated Imagery) [180] film z nominacijo za oskarja.

Animirani filmi, izdelani s pomočjo računalnika, so bili v začetku izdelani tako, da je bila narisana vsaka sličica filma. Pri tem sličice niso bile v celoti izrisane, ampak so se izdelali poligonski modeli, ki so bili potem računalniško renderirani. Za to renderiranje je skrbel Edwin Catmull (znan tudi po Z-buffer, Catmull-Clark). Sence objektov so bile samodejno generirane in ne zgolj narisane. Tudi površine objektov so bile računalniško senčene, namesto da bi bile ročno pobarvane. Zaradi skromne strojne opreme v tistih časih je izdelava filma, dolgega vsega dve minuti, trajala izredno dolgo in zgodilo se je, da sta Lasseter in Catmull včasih kar prespala v studiu v spalnih vrečah. Kreiranje slik, kjer računalnik samodejno ustvari sceno, nekoliko zamegli mejo med animacijo in simulacijo. Ko govorimo o računalniških simulacijah, poskušamo s preračuni izračunati delovanje fizikalnega sistema in potem lahko rezultate tudi vizualiziramo. Pri računalniško generiranih filmih je sprva ta simulacija morda res pomenila zgolj kakšno računanje za določanje barve površine in določanja senc objekta. Toda že v tem prvem filmu so računali oziroma simulirali gibanje kablov lučk.

V današnjem času so računalniško kreirani filmi skoraj v celoti simulacije, saj se simulirajo gibi objektov in obnašanje različnih trdin, tekočin in plinov. Zato je težko narediti ostro ločnico in določenemu videoposnetku reči animacija ali simulacija. Pogosto uporabljamo kar oba izraza in tudi v simulacijskih programih izdelavi videoposnetkov rečemo animacija, čeprav gre dejansko za rezultate simulacije. Kreiranje animiranih posnetkov lahko po stopnji avtomatizacije ločimo na tri tehnike: animacija posnetkov, animacija s ključnimi posnetki in animacija s simulacijo gibanja.

12.2 Animacija posnetkov

Animacija posnetkov (angl. Stop-motion animation) [181] je tehnika, kjer izdelamo vsako sličico animacije. To pomeni, da je treba izdelati najmanj 25 sličic za eno sekundo animacije, da bo gibanje gladko. Vsaka sličica prikazuje časovno ustavljeno dogajanje v delčku sekunde. Takšno animacijo lahko naredimo tudi brez računalnika s fotografskim aparatom tako, da slikamo posamezno narisano sličico ali poljuben objekt v prostoru. Naslednjo sliko naredimo, ko objekt malenkostno spremenimo ali ga premaknemo v prostoru in v primeru narisanih slik narišemo nekoliko spremenjeno sliko. Premiki ali spremembe morajo biti dovolj majhni, da ustrezajo kratkemu časovnemu koraku. Seveda lahko takšne animacije izdelamo tudi z uporabo računalnika tako, da slike prikazujemo na zaslonu. Vnaprej narisane sličice lahko prikazujemo s poljubno hitrostjo na zaslonu in na tak način predvajamo animacijo.

Lahko izdelamo tudi računalniški program, ki sproti riše grafiko in jo prikazuje. Vendar se ne sme zgoditi, da bi gledalec videl proces risanja, zato v računalniški grafiki pogosto uporabimo dvojni ali trojni grafični pomnilnik (angl. Double Buffer, Tripple Buffer). Tak podvojen pomnilnik omogoča risanje grafičnih elementov na skriti pomnilnik in prikazovanje slike na zaslonu iz podvojenega pomnilnika. Takoj, ko se risanje zaključi, se oba pomnilnika zamenjata in skriti pomnilnik postane viden, prejšnji vidni pomnilnik pa postane nov skriti pomnilnik. Preklop med pomnilnikoma je zelo hiter in skoraj trenuten, zato ga človeško oko ne zazna. Pogosto v računalniških programih najdemo možnost, s katero lahko izbiramo med enojnim, dvojnim, trojnim ali celo štirikratnim pomnilnikom. Tak pomnilnik ni namenjen samo predvajanju animacij, ampak se uporablja tudi pri normalnem delu z računalniško grafiko, kjer bi bilo moteče, če bi se grafični elementi izrisovali na zaslonu. Zato lahko brez težav vrtimo 3D model po zaslonu in se scena gladko renderira.

Tehnika animacije posnetkov zahteva kar nekaj znanja s področja animacije. Ustvarjalec si mora zamisliti in določiti vsak posnetek animacije posebej. Zato je treba vnaprej izdelati scenarij in določiti vse gibe in spremembe objektov. Nato je treba tudi preračunati gibanje tako, da glede na število sličic ne bo prehitro ali prepočasno, saj bo gledalec to zelo hitro opazil.

Ko gre za sestavljene objekte ali like v animaciji, jih je dobro razdeliti na posamezne dele. Pri likih so to lahko udi, oči, usta itd., da jih lahko med kreiranjem posnetkov premikamo neodvisno, kot so to včasih počeli s prozornimi folijami pri animiranih filmih. V 3D modeliranju je običajno sestav izdelka narejen iz posameznih neodvisnih delov, ki jih lahko prav tako premikamo neodvisno za potrebe animacije.

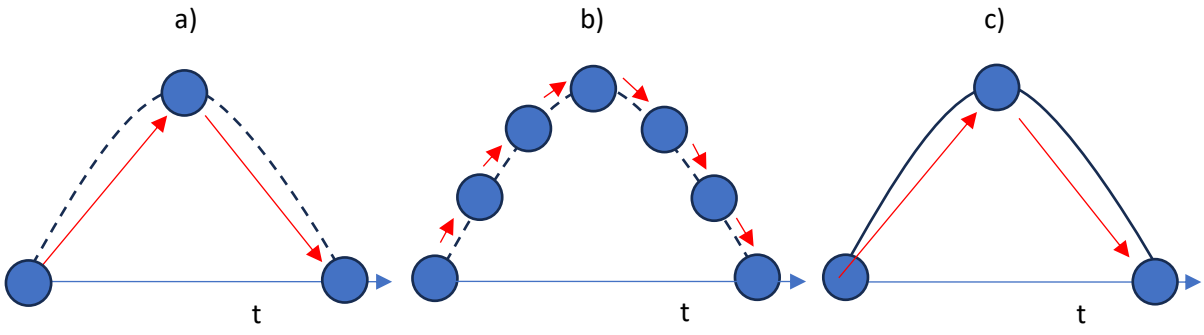
Za ustvarjanje posebnih učinkov animacije je treba dodati tudi nekoliko boj zahtevne postopke animiranja, kjer s posnetki ustvarimo iluzijo fizikalnih pojavov. Tako lahko manipuliramo s časom, elastičnostjo ali togostjo delov tako, da objekte namerno preoblikujemo, raztegnemo ali skrčimo. Seveda to zahteva veliko znanja in izkušenj z animacijami in proučevanjem naravnih pojavov. Poleg tega je vse skupaj še precej časovno zahtevno, saj je treba izdelati veliko takšnih posnetkov.

Po drugi strani je izdelava animacije iz posameznih posnetkov z uporabo računalnika izredno preprosta. V poljubnem programu za urejanje rastrske grafike je treba zgolj vsak posnetek shraniti kot en nivo slike in potem lahko takšno animacijo shranimo v obliki animirane gif datoteke [182]. Sicer lahko množico slik shranimo tudi v obliki videoposnetka, za kar imamo na voljo številne formate za zapis videa.

12.3 Animacija s ključnimi posnetki

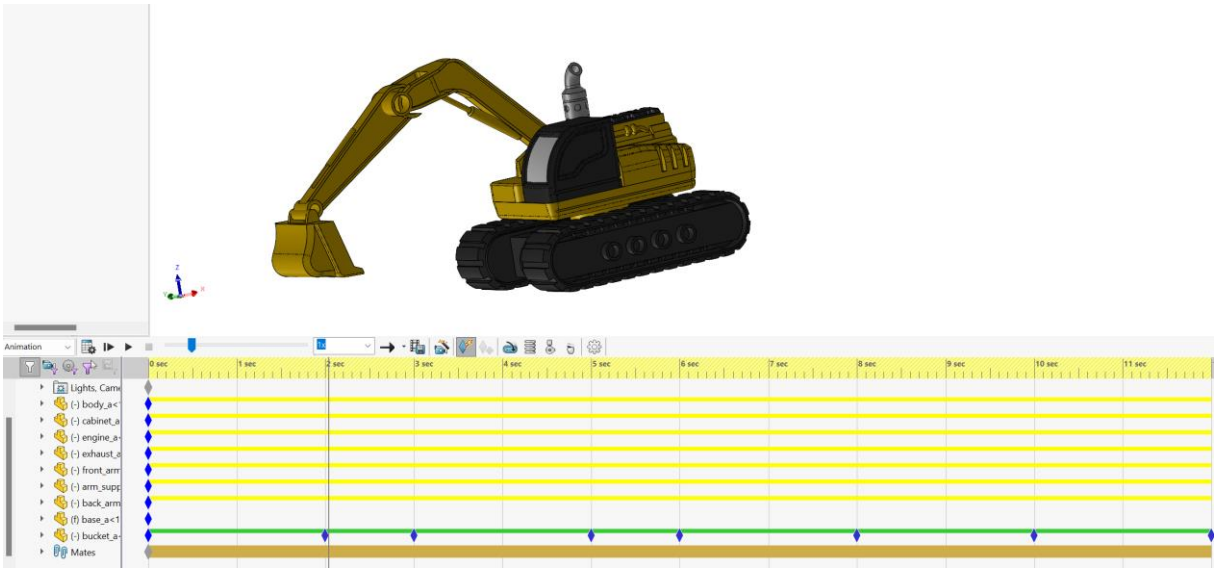
Dokaj zamudno je animirati prav vsak posnetek za vse delčke sekunde videa. Zato se pogosto srečamo s kreiranjem animacij s ključnimi posnetki (angl. Keyframe animation) [183]. Animacijo s ključnimi posnetki je izumil Nestor Burtnyk, ko je delal za televizijo Kanade. Ideja izhaja iz filmskega traku, kjer posamezni sliki oziroma posnetku rečemo okvir (angl. frame) in tako lahko določimo ključne posnetke, med katerimi so spremembe posnetkov zelo majhne. Nato je možno s pomočjo računalnika med ključnimi posnetki umetno kreirati te vmesne posnetke. Na tak način lahko animator nariše samo ključna posnetka animiranih likov v dveh različnih položajih in nato se z računalnikom generirajo vse vmesne sličice. Pri tem je treba uporabiti interpolacijo, da določimo položaje vseh vmesnih pikslov. Interpolacija med ključnimi posnetki je lahko linearna ali v obliki višje rednih krivulj [184].

V primeru linearne interpolacije je treba v določenih primerih narediti veliko več ključnih posnetkov. Na sliki 12. vidimo primer animiranja leta krogle. Če uporabimo samo tri ključne posnetke in linearno interpolacijo na sliki 12.a, bo let krogle prikazan v obliki trikotnika namesto parabole. Zato je treba narediti dodatne ključne posnetke, kot je prikazano na sliki 12.b. V primeru parabolične interpolacije na sliki 12.c zadostujejo samo trije ključni posnetki.



Slika 12.1. Linearna in parabolična interpolacija

Animacija s ključnimi posnetki je še posebej zanimiva pri prikazovanju CAD modelov v gibanju. Namesto da bi premikali posamezne dele sestave za drobne korake, ki jih del naredi v petindvajsetini sekunde, postavimo sestavo v ključne položaje. Vsi vmesni položaji so nato izračunani z interpolacijo med ključnimi položaji. Tako lahko v programu Solidworks izdelamo analizo gibanja sestave tako, da po določenem scenariju postavimo dele sestave v določen položaj s pomočjo časovnice, ki je prikazana na spodnjem delu slike 12.2. Vsak sestavni del ima na časovnici svojo linijo in znotraj analize gibanja lahko uporabnik določi začetne pogoje, sile in momente ter analizira hitrosti, pospeške ali trke. Nato lahko tako izdelano animacijo izvozimo v primeren format za prikazovanje animacije v videoprikazovalniku.

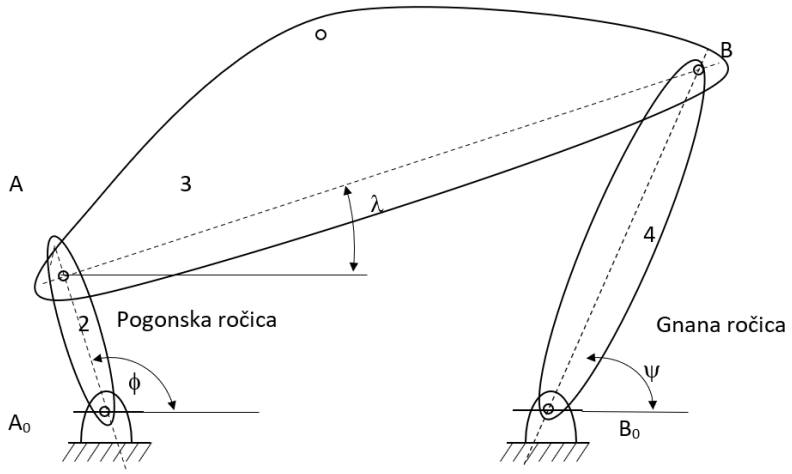


Slika 12.2. Analiza gibanja v programu Solidworks

Takšen način izdelave animacije nam delo precej olajša, saj nam ni treba skrbeti za vse podrobnosti gibanja. Zato se lahko bolj posvetimo kakovostni izdelavi predstavitve, kjer sta zelo pomembna tudi začetek in konec animacije. Na začetku je pomembno, da gledalca pripravimo na dogajanje. Tako je primerno narediti uvod v gibanje in se glavnina gibanja začne dogajati šele po tem uvodu. Prav tako se mora gibanje na koncu zaključiti v nekakšnem izteku. Tako se izogibamo posnetkom delovanja izdelka, kjer se gibanje začne in konča hipno. Velikokrat je zaželeno narediti neskončen video. To pomeni, da se animacija začne in konča z enakim ključnim posnetkom. Izdelek je na koncu v enakem položaju, kot na začetku zato, da lahko potem posnetek zavrtimo v zanki in dobimo neskončno ponavljajoče gibanje izdelka.

12.4 Animacija s simulacijo gibanja

Čeprav je mogoče pri animaciji s ključnimi posnetki določiti nekatere robne pogoje, je treba gibanje določiti ročno. V določenih primerih je želeno, da celotno gibanje v animaciji določimo s simulacijo gibanja (angl. Motion Capture) [185]. Izraz »motion capture« pomeni zajemanje gibanja, ker je posebej za naravno gibanje precej težko izdelati ustrezne simulacijske modele, zato si pomagajo s snemanjem gibanja. Z metodami kinematike je namreč mogoče določiti gibanje objektov na podlagi začetnih podatkov, če je mehanizem določen in matematično rešljiv. Tako je treba za mehanizem na sliki 12.3 ugotoviti, koliko je neznank in koliko kinematičnih enačb lahko določimo za mehanizem. Če je število neznank in enačb enako, je mehanizem rešljiv, sicer je nedoločen ali predoločen in nimamo matematične rešitve za mehanizem. Kadar ni direktne matematične rešitve mehanizma, lahko tak problem rešujemo tudi z različnimi numeričnimi metodami, s katerimi lahko simuliramo delovanje mehanizmov.



Slika 12.3. Štirizgibni mehanizem

Tako lahko za mehanizme določimo vse odvisne pomike, hitrosti in pospeške. Izračunamo lahko tudi sile, ko podamo mase elementov. Masa posameznih delov sestave je določena z volumnom in gostoto materiala. Ko izvedemo simulacijo z uporabo sil in geometrijo sestave, lahko določimo tudi pomike, ki so posledica teh sil. Podobno lahko v nadaljevanju določimo tudi specifične deformacije in napetosti v delih mehanizma. Na ta način lahko določimo še mesta razpok in posamezni deli se lahko tudi odlomijo. Tako lahko izdelamo video iz dinamične simulacije, kjer so za vsak časovni korak izračunane deformacije in napetosti. Tako je vsak časovni korak ena 3D scena, iz katere lahko naredimo posnetek iz poljubnega gledišča. Z zaporednim kreiranjem scen lahko izdelamo 3D video posnetek, znotraj katerega si lahko ogledujemo video iz različnih zornih kotov.

Takšne simulacije so lahko izredno zahtevne in se lahko izvajajo tudi nekaj dni na superračunalnikih. Zato so pogosto za potrebe računalniške grafike te simulacije poenostavljene, da dajejo dovolj hitro dovolj natančne rezultate. Tako lahko danes v okviru računalniške grafike izvajamo simulacije, ki simulirajo trke, mehanizme, tekočine ali pline. Takšni programi se izvajajo v okviru programske opreme grafičnih kartic, ki se običajno imenuje fizika, ker simuliramo naravne pojave, za katere veljajo fizikalni zakoni. Velika hitrost strojne opreme omogoča izvajanje takšnih programov v realnem času, kar se navadno izkorišča v okviru računalniških iger. V določenih primerih so simulacije še vedno preveč zahtevne in potem se lahko dejansko lotimo zajemanja gibanja.

Zajemanje gibanja je dejansko postopek, ki je precej podoben postopkom pri animaciji posnetkov. Tukaj s kamero snemamo gibanje v naravi in potem rezultate tega snemanja uporabimo za izdelavo animacije. Pri snemanju je ustvarjeno veliko podatkov, ki jih reduciramo zgolj na nekaj časovnih točk in na ključne posnetke. Tako na primer igralca opremijo z lučkami na različnih delih telesa in te lučke potem predstavljajo diskretne točke v prostoru, za katere opravljamo meritve položaja in hitrosti v določenem časovnem koraku. Te podatke nato preslikamo na lik v animiranem filmu in določimo gibanje tega umetnega lika na podlagi realnega gibanja. Tako dobimo veliko bolj naravno gibanje v animiranem filmu. Ti rezultati zajemanja gibanja omogočajo tudi izdelavo simulacij gibanja, ki jih lahko nato spreminjamo in izboljšujemo. Zato je danes zelo težko ločiti med igranim in animiranim filmom. V sodobnih filmih igrajo tudi že umrli igralci, velikokrat se je mogoče tudi izogniti tveganim prizorom, za katere potrebujemo kaskaderje.

Za simulacije izdelkov je mogoče uporabiti podobne tehnike, saj vidimo veliko videoposnetkov, ki prikazujejo napredne robote v gibanju in so pogosto zgolj računalniška animacija z zajemanjem gibanja igralcev. Uporaba umetne inteligence omogoča tudi simuliranje gibanja z uporabo strojnega učenja.

Zato je danes mogoče zaslediti veliko lažnih posnetkov, ki so precej prepričljivi, vendar so zgolj z umetno inteligenco kreirane animacije. Za zdaj so vsaj deloma avtorji ideje ali delov posnetka še vedno ljudje, vendar lahko kmalu pričakujemo animirani film, ki bo v celoti od zasnove do produkcije izdelan brez avtorja.

12.5 Programi in datoteke za zapis videa

Glede na razširjenost operacijskega sistema Windows je večina videoposnetkov v obliki AVI (angl. Audio Video Interleave), ki jo je razvil Microsoft leta 1992 [186]. Surovi videoposnetki v obliki AVI so le zapored shranjene slike in posledično zasedajo veliko pomnilniškega prostora. Ker je bila količina prostora na medijih za zapis videa (CD, DVD) omejena, so razvili različne algoritme za stiskanje videoposnetkov. Že leta 1988 so zasnovali skupino MPEG (angl. Moving Picture Experts Group) [187], ki skrbi za standarde na področju stiskanja video- in avdioposnetkov. Ti standardi so običajno označeni s kratico MPEG, ki ji sledijo različne številke ali znaki. Trenutno najbolj razširjen standard je MPEG-4, čeprav je na voljo že najnovejša verzija standarda MPEG-5. Datoteka s stisnjenim avdio-video posnetkom po standardih MPEG ima običajno oznako MP4 [188]. Poleg AVI in MP4 oblike datotek so bile precej razširjene tudi datoteke v obliki MOV (angl. Quicktime Movie), ki so jih razvili v podjetju Apple, vendar jih je danes skoraj popolnoma zamenjala oblika MP4. Seveda je poleg uradnih formatov datotek priznanih proizvajalcev strojne in programske opreme še cela množica odprtokodnih zapisov datotek. Med temi je mogoče najbolj znan zapis datoteke, imenovan Matroska s končnico MKV [189], ki je prosto dostopen format, primerljiv z zapisom MP4.

Datoteka z videom je lahko v različnih oblikah shranjena in predvajanja s posebnimi programi, ki jih imenujemo kodeki (angl. Codecs) [190]. Najpogosteje uporabljeni kodeki danes temeljijo na MPEG oblikah in so znani po oznakah H.264 in H.265. Takšna programska oprema omogoča stiskanje videoposnetkov, kar se imenuje kodiranje (angl. encode), obraten postopek pa je dekodiranje (angl. decode). Predvajanje videa, kodiranega z določenim kodekom, je mogoče samo na napravah, kjer je nameščena programska oprema za dekodiranje videa. Stiskanje podatkov s kodeki je asimetrično, kar pomeni, da je postopek stiskanja v komprimirano obliko precej bolj zahteven kot postopek predvajanja videa. Zato se pogosto zgodi, da je kreiranje videoposnetka precej računsko zahtevno in pogosto zahteva posebno strojno opremo v obliki zmogljive grafične kartice. Pregled različnih formatov za zapis videoposnetkov je mogoče najti na spletu v obliki preglednic [191], kjer je mogoče primerjati formate zapisov in najpogosteje uporabljenih kodekov.

Ker so v uporabi številni kodeki in vrsta različnih datotek za zapis videoposnetkov, so na voljo tudi številni programi, ki omogočajo pretvorbo videoposnetkov. Nekatera orodja so namenjena izključno za pretvorbo videoposnetkov. Najbolj znano takšno programsko orodje je prosto dostopna orodjarna, imenovana FFmpeg [192], ki prvotno deluje v obliki pisanja ukazov v ukazni vrstici operacijskega sistema. To orodje je zelo učinkovito in danes je mogoče s pomočjo programov umetne inteligence preprosto poiskati pravilen ukaz za želeno operacijo. Izdelava videoposnetka seveda zahteva veliko več, zato je pogosto treba uporabiti programe, ki poleg pretvorbe omogočajo tudi montažo. Glavna naloga takšnih programov je urejanje videoposnetkov (rezanje, lepljenje, dodajanje različnih učinkov in zvoka) ter nato zapis končnega izdelka v poljubnem formatu. Za urejanje videoposnetkov so na voljo številna komercialna programska orodja, npr. Adobe Premiere Pro, DaVinci Resolve, Final Cut Pro, Avid Media Composer, Vegas Pro. Skoraj vsi programi ponujajo preizkusne verzije programov, na voljo pa so tudi nekateri odprtokodni ali prosto dostopni programi, na primer OpenShot, Shortcut, Kdenlive in VSDC Free Video Editor. Večina programov je zelo podobna po zmogljivostih, spodaj je opisanih le nekaj najbolj popularnih orodij:

Adobe Premiere Pro [193]

Verjetno najzmogljivejši program za urejanje videoposnetkov je Adobe Premiere Pro. Program je plačljiv, vendar ni na voljo v obliki enkratnega nakupa, saj podjetje Adobe ponuja programsko opremo v obliki mesečne ali letne naročnine. To pomeni, da lahko vedno uporabljamo zadnjo verzijo programa, ki omogoča urejanje videoposnetkov s številnimi funkcijami možnosti montaže, barvne korekcije in posebnih učinkov. V orodjih Adobe je vedno več sistemov umetne inteligence, ki močno olajšajo delo, saj omogočajo preproste načine montaže in kreiranja umetnih scen. To precej olajša tudi posebno popravljanje videa z dodajanjem novih elementov, kar je bilo včasih izvedljivo le z dolgotrajnim popravljanjem posameznih posnetkov. Program ponuja široko podporo za različne izhodne formate v različnih ločljivostih, vključno s stereoskopskimi zapisi, in pri tem podpira zmogljivo strojno grafično opremo.

DaVinci Resolve [194]

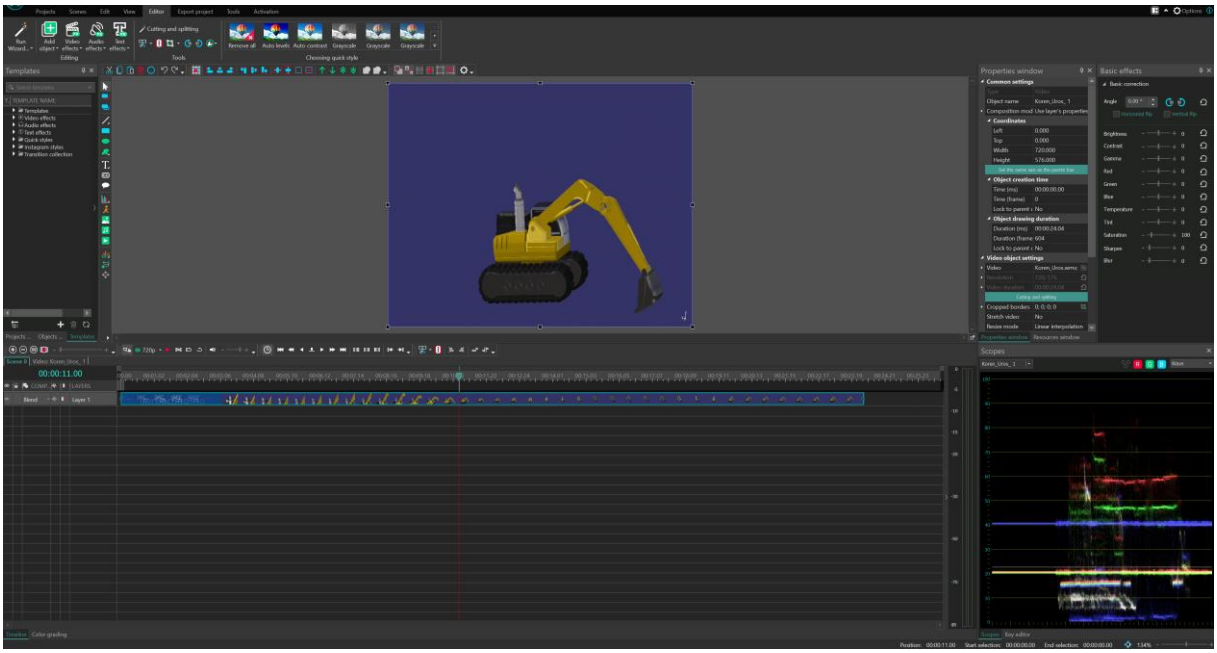
Zelo težko je narediti ločnico med posameznimi programi in razlik v funkcijah med programom DaVinciResolve in Adobe Premiere Pro je zelo malo. Morda ima DaVinciResolve celo več funkcij in prav tako pospešeno uvajajo sisteme umetne inteligence v program. V nasprotju z Adobe Premiere Pro je mogoče program kupiti in se izogniti periodičnim plačilom. Izbira programskega orodja je seveda povezana z izkušnjami in pričakovanji uporabnikov in manj z dejanskimi prednostmi in slabostmi posameznega programa.

Final Cut Pro [195]

Tudi ta program omogoča profesionalno obdelavo videoposnetkov, vendar je na voljo samo na računalnikih podjetja Apple. To dejstvo ni prevelika ovira, ker oblikovalci večinoma uporabljajo Apple računalnike. Zato je za ta segment uporabnikov program Final Cut Pro naravna izbira. Tudi ta program omogoča podobno funkcionalnost kot prejšnja dva programa. Morda je uporabniško nekoliko bolj prijazen in je tako kot DaVinciResolve na voljo ob enkratnem plačilu. Program je nekoliko težje uporabljati v skupinskih projektih, saj je namenjen le za uporabo enemu uporabniku.

VSDC Free Video Editor [196]

Med prosto dostopnimi programi za urejanje videoposnetkov je vredno omeniti program VSDC, ki tudi v prosto dostopni verziji omogoča številne možnosti, ki jih ponujajo komercialna orodja. Tudi za VSDC je na voljo plačljiva verzija, ki omogoča sledenje gibanju, kreiranje sekvenc s pomočjo umetne inteligence, stabilizacijo videa in ekstenzivno uporabo strojne grafične opreme. V obliki plačljive verzije je program po funkcionalnosti enak zgoraj omenjenim komercialnim programom. Vendar osnovne funkcije, ki so na voljo v prosti verziji, zadoščajo za uporabo programa za uporabnike, ki se z videoposnetki ne ukvarjajo profesionalno. Za pripravo videoposnetkov v namene promocije izdelka na osnovi 3D modela ta program popolnoma zadošča.



Slika 12.4. Okno programa VSDC

13 Virtualna resničnost

Virtualna resničnost (angl. Virtual Reality) [197] je že desetletja področje različnih raziskav, vendar si le s težavo utira pot do uporabnikov. Računalniško generirane 3D modele lahko dojemamo samo z različnimi pripomočki, ki nam pričarajo iluzijo realnega izdelka. Najprej se je ta težnja pokazala v umetnosti, ko so v renesansi odkrili perspektivno projekcijo, ki je veliko bolj realistično prikazala prostorsko sceno na platnu ali freski. Od takrat se ni veliko spremenilo, čeprav so že v devetnajstem stoletju odkrili stereoskopske učinke. Ves čas je to ostalo nekje v ozadju, zanimivo le za manjše skupine ljudi. V 60. letih dvajsetega stoletja so posneli veliko stereoskopskih filmov in tudi to se ni prijelo. Med letoma 2010 in 2013 je bilo obdobje, ko so proizvajalci zasuli trg s t. i. 3D televizorji, ki so omogočali stereoskopsko predvajanje z uporabo pasivnih ali aktivnih očal. Vendar tudi takrat to ni zaživelo in danes je skoraj nemogoče najti tak televizor. Mobilni telefoni omogočajo sisteme razširjene resničnosti (angl. Augmented Reality) [198], ki je v zadnjem času precej popularna. Razširjena resničnost je vedno bolj povezana z virtualno resničnostjo in ta dva sistema se danes zelo prepletata. Najnovejši sistemi, kot so Microsoftov Hololens, Meta Quest 3 in Apple Vision Pro, omogočajo kombinacijo VR in AR in to imenujejo z različnimi termini, na primer povezana resničnost (angl. connected reality), mešana resničnost (angl. mixed reality) in razširjena resničnost v širšem smislu (angl. extended reality). Različni izrazi, ki jih je včasih že zelo težko prevesti v slovenščino, so posledica konkurence na trgu, kjer se različni proizvajalci borijo za svoj delež trga. Glede na vlaganja v to področje je mogoče pričakovati, da bo v nekaj letih ali desetletjih to področje toliko zaživelo, da bo dobilo precej večjo vlogo tudi na področju 3D modeliranja. Zato je dobro poznati tudi to področje, ki bo pregledno povzeto v nadaljevanju.

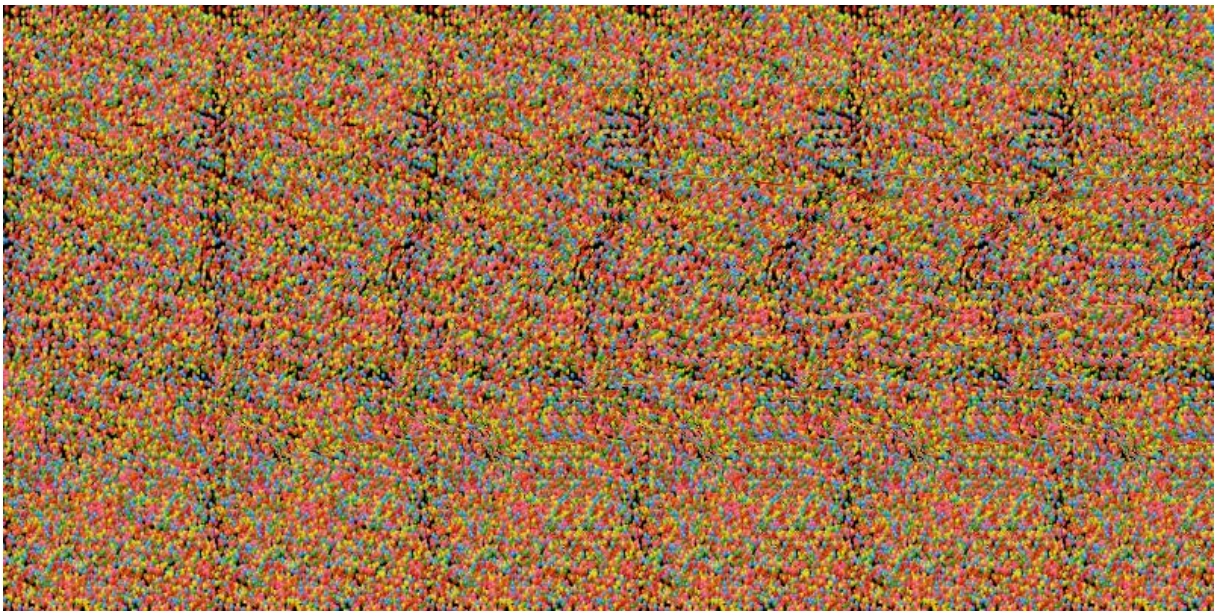
13.1 Pasivna vizualizacija

Pasivna vizualizacija navidezne resničnosti pomeni, da nismo opremljeni z aktivnimi napravami. Če za gledanje navidezne resničnosti uporabljamo očala, to pomeni, da so to pasivna očala brez dodatnega napajanja. Najlažje je, če sploh nimamo nobenega pripomočka in lahko opazujemo navidezno resničnost. Vseeno je treba nekako preslepiti čutila in jasno je, da perspektivne projekcije na zaslonu ne doživljamo kot vključenost v 3D prostor. Kljub temu lahko z določenimi triki pričaramo iluzijo prostora. Najlažje to naredimo s premikanjem objekta, saj s tem spremenimo gledišče. To na nek način ustvari iluzijo prostora, saj v naravi lahko tudi samo z enim očesom ocenimo razdaljo do objekta. Če zapremo eno oko in pogledamo svoj palec na roki, vidimo samo 2D sliko palca, ko ga zavrtimo, dobimo takoj 3D pogled na objekt pred očesom. Podobno se zgodi, ko zavrtimo model na zaslonu. Zato pogosto pri 3D modeliranju uporabljamo vrtenje objektov, da si lažje predstavljamo objekt v prostoru. V ta namen se pri modeliranju uporabljajo naprave, ki jim rečemo 3D miške (slika 11.17). Več slik objekta v različnih položajih lahko tudi natisnemo s tehniko, ki se imenuje Lentikularno tiskanje (angl. Lenticular printing) [199]. Izraz izhaja iz latinske besede leticus, ki pomeni majhen list, saj pri tem postopku uporabljamo majhne kroglice v obliki lističev, ki delujejo kot leče in svetlobo razpršijo v različnih smereh. Posledično lahko vidimo različne slike, če pogledamo na lentikularni tisk iz različnih smeri. Tako natisnjene slike pogosto srečamo v obliki otroških igrač, vendar so poskušali narediti številne naprave, ki izkoriščajo ta fenomen že od prvega patenta v devetnajstem stoletju. Najbolj dovršena takšna naprava je bil televizor Philips Inition, ki je bil na voljo v omenjenih letih razmaha 3D televizorjev. Omogočal je gledanje prostorske slike tako, da je televizor v vsakem pikslu kreiral po dve različni sliki z lentikularno tehniko in jih hitro menjaval. Tako je bilo mogoče spremljati 3D sceno brez pripomočkov. Cena televizorja je bila visoka in ni zmogla konkurirati proizvodom, ki so enako ali boljšo kakovost slike ponujali z uporabo očal, vendar z neprimerno nižjo ceno. Zato je danes težko najti podatke o lentikularnih teleivizorjih, saj so razvoj skoraj popolnoma opustili.

Podobno lahko izdelamo tudi druge animirane gif slike, ki nam zaradi spremembe gledišča z lentikularno tehniko dajo občutek opazovanja 3D scene. Na internetu je mogoče najti številne takšne animirane gif datoteke, ki z lentikularno tehniko prikazujejo 3D scene [200].

Poleg tega se v aktivnih aplikacijah, kot so računalniške igre, poslužujejo še drugačnih pristopov, kjer ustvarijo različne atmosferske ali naravne učinke. Tako vidimo pogosto različne meglice, dim, ali premikajočo naravo (drevesa, trava itd.), kar naredi sceno veliko bolj realistično in omogoča »potopitev« v navidezni svet do določene mere. Takoj ko pogledamo mimo okvirja zaslona, ta iluzija izpuhti. To lahko preprečimo tako, da uporabnika postavimo v prostor, kjer so okoli njega povsod zasloni, ki prikazujejo navidezni svet. V zabaviščnih parkih ali v namenskih kino dvoranah imajo takšne prostore, kjer je povsod okoli gledalca zaslon. Tako gledalec nima referenčne točke in je potopljen v svet, in dokler je projekcija dovolj daleč od gledalca, ima iluzijo, da je dejansko v prostoru. Tak učinek lahko danes dosežemo s kamerami, ki snemajo s kotom 360 stopinj, kar pomeni, da je posneto vse v sferi okoli kamere in tako lahko tudi predvajamo celotne posnetke prostora gledalca, ki ima potem občutek, da je dejansko tam. Iluzijo izgubimo, ko pridemo do stene oziroma do platna tako, kot se zgodi junaku v filmu Truman show. Še prej iluzija izgine, ko spremenimo gledišče, saj se scena ne spreminja v odvisnosti od gledišča. Tudi objekti na platnu niso identični realnim 3D objektom, saj nimajo globine.

Globino dosežemo tako, da simuliramo naravni vid, kjer pri gledanju uporabljamo obe očesi in vsako oko pošlje v možgane nekoliko drugačno sliko, ki se potem zlije v 3D predstavitev objekta. Takšno predstavitev imenujemo stereoskopska predstavitev, kjer imamo za vsako oko kreirano svojo sliko. Nato je treba poskrbeti, da eno oko vidi samo eno sliko. Posebna izvedba takšnih stereoskopskih slik uporablja trik, kjer skrijemo sliko za posamezno oko v barvnem vzorcu. Nato je treba najti točko gledišča in oči usmeriti vzporedno ali jih prekrižati, da se pokaže stereoskopska slika. Zato takšno predstavitev lahko imenujemo stereoskopske slike z napenjanjem oči. Na sliki 13. je prikazana slika z vzorcem, kjer lahko vidimo stereoskopsko predstavitev morskega psa, če napenjamo oči, dokler se ne prikaže stereoskopska slika. Na spletnih straneh lahko najdemo veliko takšnih slik, ki se tudi prodajajo na spletu in v knjižnih oblikah. Seveda je takšno gledanje bolj namenjeno zanimivosti kot resni stereoskopski predstavitvi objektov.



Slika 13.1. Stereoskopska slika z napenjanjem oči [201]

Za kakovostnejši prikaz je treba sliko za posamezno oko skriti nekoliko drugače. Že v 19. stoletju so odkrili način prisilnega gledanja slik s posameznim očesom. V ta namen se je najprej uporabljal stereoskop, ki ga je prvi izdelal Charles Wheatstone [202]. Izboljšan stereoskop je bila daljnogledu podobna naprava, ki služi za gledanje dveh slik, ki sta posneti z dvojno kamero ali z zamaknjenim gledališčem. Na sliki 13.2 je primer takšnega stereoskopa.

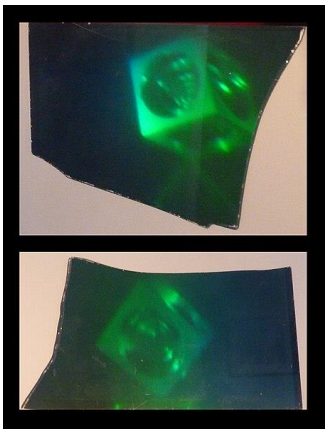
Slika 13.2. Stereoskop [203]

Stereoskop je primeren za opazovanje ene statične slike, poleg tega je naprava precej okorna in relativno nerodna za uporabo. Zato so se prav tako že v 19. stoletju domislili bolj praktičnega načina za posredovanje določene slike v določeno oko. Ta način se imenuje anaglyph [204], po grški besedi za izrezovanje, saj lahko potem vidimo izrezano ali reliefno sliko. Ta tehnika deluje tako, da uporabimo očala s komplementarnima filtroma. Običajno imamo očala z rdečim in modrim steklom. Enaka filtra nato uporabimo na dveh projektorjih, s katerima enako sceno projiciramo na isto platno. Filtra omogočata, da vsako oko vidi samo eno od obeh slik, ki sta projicirani na platno. Možgani nato sliki povežejo in vidimo izrezljano (anaglyph) oziroma stereoskopsko sliko scene. Tako lahko opazujemo posamezne slike ali filme. Očala so pasivna in ne zahtevajo dodatnega napajanja in tehnika je tako preprosta, da se zdaj uporablja že več kot 130 let in je še vedno glavni vir stereoskopskih slik. Tako lahko tak način projiciranja vidimo v sodobnih 3D kinodvoranah, na televiziji in na internetu. V tej tehniki je posneta tudi površina Marsa in tako lahko stereoskopsko opazujemo površino drugega planeta. Seveda je mogoče tudi poljuben 3D model pretvoriti v anaglyph sliko. Na sliki 13.3 so anaglyph očala in slika modela, narejenega v programu Solidworks. Iz modela lahko naredimo dve sliki v različnih pogledih, ki jih nato obdelamo s filtroma v programih za urejanje rastrske grafike. Veliko modelirnikov omogoča tudi direkten izvoz anaglyph slike iz renderirane slike modela. V tem primeru ni treba predhodno izdelovati različnih pogledov modela, ampak se celoten postopek izvede z uporabniškim vmesnikom za izdelavo anaglyph slik.

Slika 13.3. Anaglyph očala in stereoskopska slika

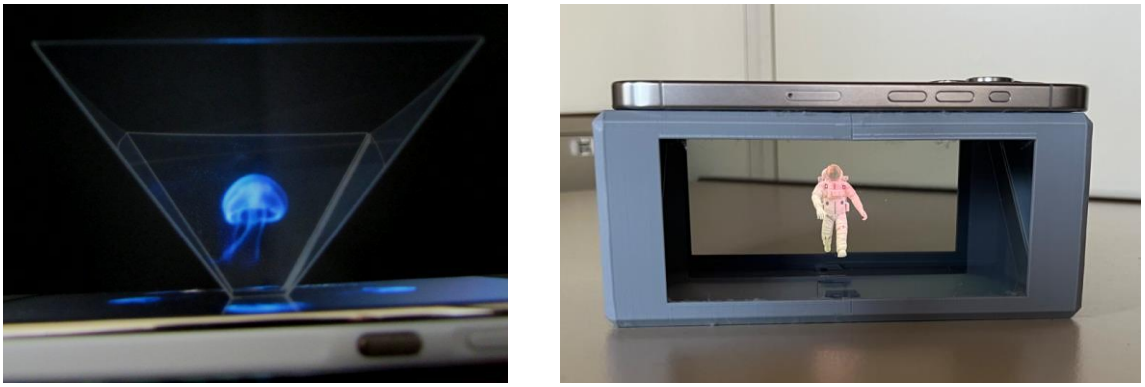
Glavna slabost tehnike anaglyph je dejstvo, da uporabljamo barvne filtre za skrivanje slik. Zato je opazovanje anaglyph posnetkov nekoliko čudno brez uporabe očal. Poleg tega so tudi barve na posnetkih nekoliko okrnjene. Namesto barvnih filtrov lahko uporabimo tudi nekoliko drugačne filtre, ki omogočajo ohranitev barv. Pri takšni izvedbi imamo namesto barvnih filtrov filtra, ki različno polarizirata svetlobo [205]. Podobno kot pri anaglyph tudi tukaj uporabimo enaka filtra za obdelavo pogledov in rezultat je enak oziroma boljši, saj so barve na sliki pravilne. Ker imamo kombinacijo dveh slik na isti površini, je še vedno slika neprimerna za gledanje brez očal. Takšen sistem s polarizacijskimi filtri se uporablja v 3D kino dvoranah in pri nekaterih 3D televizijah in monitorjih, ki so delovali s pasivnimi očali. Običajno so televizorji in monitorji delovali tako, da je vsaka druga rastrska vrstica drugače polarizirana. Zato je bila stereoskopska slika prikazana s polovično ločljivostjo.

Seveda bi bila stereoskopska predstavitev še boljša brez uporabe očal, tako da bi dejansko virtualni model projicirali v prostor. V znanstveni fantastiki je takšno projiciranje modela v prostor imenovano hologramska predstavitev, čeprav so resnični holografi precej manj prepričljivi in za zdaj še ni mogoče kakovostno projicirati modela v prostor tako, kot je vidno v nekaterih filmih. Hologram in holografijo [206] v resničnem svetu je izumil Dennis Gabor, ki je zato leta 1971 prejel Nobelovo nagrado. Holografijo je razvil v 40. letih 20. stoletja pri razvoju elektronskega mikroskopa. Pravi razmah je holografija dosegla šele z razvojem laserjev. Holografija omogoča snemanje interferenčnega vzorca svetlobe na fotografski film. Namesto slike posnamemo svetlobno polje odbojev in pri tem uporabimo ogledala, da dobimo slike iz različnih zornih kotov. Zato lahko objekt na filmu opazujemo tako, da je viden iz poljubnega zornega kota. Še bolj osupljivo je dejstvo, da lahko izrežemo poljuben del filma in je na njem še vedno shranjena slika celotnega modela, kot je prikazano na sliki 13.4.



Slika 13.4. Hologram [207]

Nekoliko težje je tako posnet 3D objekt projicirati v prostor. Naprave, ki to do neke mere omogočajo, so precej drage in omejeno uporabne. Veliko lažje je narediti iluzijo hologramа. V ta namen se pogosto uporablja tehnika, imenovana Pepper Ghost [208], kjer projiciramo sliko na prozorno odbojno folijo. Tako so bile ustvarjene iluzije hologramov umrlih umetnikov, ki so »v živo« nastopali na odru. Takšno miniaturno navidezno hologramsko projekcijo lahko naredimo tudi z mobilnim telefonom. Na sliki 13.5 je na levi strani prikazan način projiciranja z invertirano piramido iz transparentne folije. Na desni strani pa je primer škatle, izdelane s 3D tiskalnikom, v katero vstavimo prozorno steklo z odbojno folijo, ki odseva sliko zaslona mobilnega telefona [210]. Na enak način delujejo zgoraj omenjene odrske predstave, kjer so seveda odbojne površine ustrezno večje, gledalci pa dobijo iluzijo umeščenosti objekta v prostor. Pri invertirani piramidi lahko objekt opazujemo iz poljubne strani, vendar je na zaslonu več slik oziroma ena slika za vsako površino piramide.



Slika 13.5. »Pepper ghost« učinek z mobilnim telefonom [209]

13.2 Aktivna vizualizacija

Najprej so termin aktivna očala uporabljali pri 3D televizorjih, kjer so dosegli boljšo kakovost vizualizacije tako, da so namesto filtrov v očalih preprosto zatemnili eno oko, ko je bila na zaslonu slika za drugo oko. Nato se je s prikazom druge slike zatemnilo drugo oko. Takšna aktivna 3D očala so namesto stekel vsebovala preprost LCD zaslon, ki je kot žaluzije deloval tako, da se lahko steklo odpre ali zapre. Očala morajo biti sinhronizirana s sistemom za prikazovanje tako, da vsako oko vidi svojo sliko. Ker očala nimajo nobenega filtra in je celotna slika zaslona namenjena enemu očesu, vidimo sliko v polni ločljivosti z vsemi barvami. Zato je kakovost stereoskopske slike na 3D televizorju z aktivnimi očali veliko boljša kot pri televizorjih s pasivnimi očali. Seveda je takšna očala mogoče uporabiti na računalniškem monitorju ali projektorju s pogojem, da naprave zmorejo prikazovanje vsaj 120 slik na sekundo. Pri nižji frekvenci bi bilo mogoče zaznati utripanje slike, pri takšnih hitrostih pa je slika stabilna. Aktivna očala vsebujejo elektroniko in baterijo za napajanje, zato so posledično precej težja in dražja od pasivnih očal. Na sliki 13.6 so prikazana takšna preklopna očala, ki so povezana z grafično kartico računalnika in tako sinhronizirano prikazujejo slike za vsako oko posebej.

Slika 13.6. NVidia 3D očala Vision 2

Aktivna vizualizacija pa ne pomeni nujno samo očal, ki so aktivna. Pomembno je tudi, da se scena prilagaja uporabniku. Ko gledamo skozi okno stanovanja, se lahko premikamo z ene do druge strani okna in opazujemo okolico na drugi strani okna iz različnih zornih kotov. Učinek ni enak, če imamo na zaslonu posnetek okna, saj se slika na zaslonu ne prilagaja našemu pogledu. To so hitro ugotovili pri kreiranju sistema, imenovanega CAVE (angl. Cave Automatic Virtual Environment) [211], ki so ga naredili leta 1992 na Univerzi v Illinois, v Chicagu. CAVE je soba, kjer so vse stene projekcijska platna, na katera se projicira stereoskopska slika. Zato mora uporabnik v sobi uporabljati bodisi pasivna ali aktivna očala. Vendar bi se iluzija virtualne resničnosti razblinila, če se projicirane slike ne bi prilagajale uporabniku. Zato v CAVE uporabljajo sistem za sledenje uporabnika. Najpogosteje so senzorji za sledenje nameščeni kar na očala, saj je pomembno slediti le premikom glave. Zato takšnim sistemom rečemo tudi sistemi za sledenje glave (angl. Head tracking). Cene prvih CAVE sistemov so bile v višini več milijonov dolarjev, saj sta delovanje in hitro osveževanje slike zahtevala vrhunske projektorje in zelo zmogljivo računalniško opremo. Danes je zaradi večje zmogljivosti strojne opreme to mogoče narediti veliko ceneje. Cena je precej odvisna od velikosti virtualnega prostora in ločljivosti projekcije. Tako so v avstrijskem Gradcu naredili CAVE za približno 40.000 evrov, medtem ko je pred leti podjetje za izdelavo kuhinj Miele izdelalo svoj CAVE, ki je še vedno stalo 1,6 milijona evrov. Na sliki 13.7 je CAVE, ki ga v različnih izvedbah na spletu ponuja podjetje Visbox [212]. CAVE sistemi so v glavnem namenjeni promociji izdelkov in za vključenost uporabnika v virtualni prostor.

Slika 13.7. CAVE [213]

Ker je v CAVE še vedno treba nositi očala, je danes veliko bolj smiselno uporabiti kar očala, ki direktno kreirajo sliko za vsako oko posebej. Takšna očala, kjer so zasloni direktno na glavi, so znana po oznaki HMD (angl. Head Mounted Display). Očala HMD so danes glavno gonilo razvoja virtualne resničnosti. Prve verzije takšnih očal so razvijali za potrebe treninga pilotov in astronavtov in so bila zelo draga.

Poleg tega so vedno zahtevala dodaten računalnik za kreiranje slike, kjer je bil računalnik z žico povezan z očali. V zadnjih letih je postalo to veliko bolj cenovno dostopno, saj so mobilni računalniki postali dovolj zmogljivi in očala lahko delujejo samostojno. Primer takšnih VR očal je Meta Quest 3 [214], ki je prikazan na sliki 13.8. Ta očala ponujajo ločljivost 2064 x 2208 za vsako oko in vsebujejo zmogljiv procesor za delovanje aplikacij in generiranje slike. Očala še vedno omogočajo žično ali brezžično povezavo z računalnikom za bolj zahtevne aplikacije, vendar večina aplikacij deluje samostojno brez dodatnega računalnika. Očala skrbijo tudi za sledenje premikov glave uporabnika in kreirajo sliko glede na pogled. Pri tem je bilo treba rešiti težavo, ki je običajno v sistemih CAVE ni, in sicer dejstvo, da mora slika ostati pokončna tudi, če uporabnik nagne glavo na stran. Če namreč kamero postavimo postrani, bomo dobili poševno sliko, medtem ko v realnosti pri nagibu glave na stran slika v možganih ostane pokončna. Zraven VR očal spadajo tudi kontrolniki, ki omogočajo interakcijo z objekti v virtualnem svetu, da lahko objekt primemo, spustimo, aktiviramo itd. Kontrolniki navadno delujejo skupaj s kamerami na očalih. Zato je omogočeno tudi prepoznavanje rok in uporabnik lahko interakcijo z objekti v virtualnem svetu izvede kar z rokami.

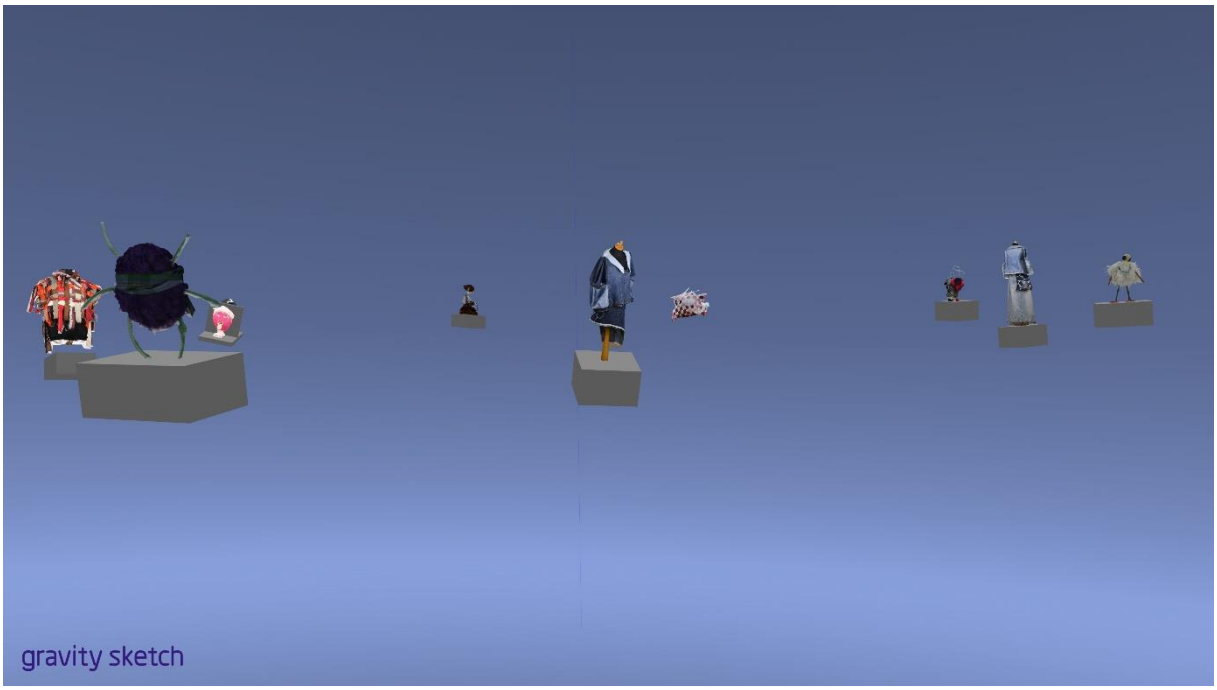
Slika 13.8. Meta Quest 3

VR očala omogočajo popolno avdio-vizualno vključitev v prostor z omejeno interakcijo z objekti. Interakcija je omejena, ker nimamo taktilnega občutka, ko nekaj primemo v prostoru. Poleg tega še pogosto manjka tudi sledenje celotnemu telesu in nogam, čeprav obstajajo določene rešitve, ki omogočajo tudi to. Tako lahko imamo posebne platforme za hojo in spremljanje telesa ter posebne čevlje za hojo. Obstajajo tudi VR obleke, ki ponujajo določen taktilni učinek, vendar so vse rešitve še v razvoju. Očala so za zdaj še precej težka, še največja težava pa je, da se mora uporabnik navaditi na delo v VR, saj začetniki pogosto čutijo slabost, podobno slabosti pri vožnji z avtomobilom. Zaradi tega in zaradi omejenih zmogljivosti baterij je delo v VR časovno omejeno na največ nekaj ur. Kljub temu lahko pričakujemo v naslednjih letih še boljše naprave za virtualno resničnost glede na velika vlaganja v razvoj te tehnologije s strani podjetij, kot so Meta, Apple in Microsoft.

Večinoma so VR očala danes namenjena za zabavno industrijo, vendar jih je mogoče uporabiti tudi za kreativno delo na področju 3D modeliranja. Z modelirniki lahko ustvarjamo modele v prostoru v »naravni« velikosti. Precej popularen program za površinsko modeliranje v virtualni resničnosti je program Gravity Sketch [140], ki omogoča tudi skupinsko delo. Tako lahko več ljudi vstopi v skupni virtualni prostor, kjer skupaj modelirajo izdelek [215]. Pri tem se seveda lahko pogovarjajo, modelirajo in pišejo sporočila.

3D modeliranje v virtualni resničnosti je tako kot celotna tehnologija šele v povojih, vendar je tak način modeliranja izredno popularen med industrijskimi oblikovalci, saj oblikovalec lahko opazuje svoj izdelek v nastajanju brez omejitve 2D površine zaslona.

Na sliki 13.9 so v programu Gravity Sketch prikazani izdelki oblikovalcev izdelani v okviru Design Week 2023 na Fakulteti za strojništvo, Univerze v Mariboru.



Slika 13.9. Modeli v VR programu Gravity Sketch

14 Sklep

3D modeliranje je v 60 letih razvoja ustvarilo številne poti in možnosti kreativnega ustvarjanja modelov izdelkov. V posameznih poglavjih tega učbenika so opisana teoretična ozadja in uporaba različnih tehnik modeliranja. Navedeni so tudi številni viri, ki so v večini primerov spletne strani, s katerih je mogoče pridobiti še bolj poglobljeno znanje s tega področja.

Vsak učbenik je seveda omejen s časom pisanja, sploh kadar gre za tehnologije, ki se še vedno razvijajo. Zato je rok uporabe učbenika omejen in je treba slediti času ter spremljati vedno nove dosežke. V zadnjem poglavju je sicer opisana virtualna resničnost, vendar ne vemo, ali bo razvoj šel v to smer ali se bo pojavila kakšna nova pot. V zgodovini so bili takšni prelomni dogodki tudi na področju 3D modeliranja, ko so v 90. letih 20. stoletja propadala velika podjetja, ki niso zmogla prehoda od delovnih postaj do osebnih računalnikov. Zdaj se večina modelirnikov seli na oblak, vendar je možno, da bo kmalu sledil prehod nazaj na mobilne naprave.

Zato naj ta učbenik služi le kot osnova za razumevanje osnovnih pojmov. Treba se je prilagajati izzivom, ki jih bo prinesla uporaba sodobnih tehnik 3D modeliranja. Glede na paleto različnih tehnik 3D modeliranja je treba tudi izbrati primerno orodje. Za razvoj industrijskih izdelkov so seveda najbolj primerni modelirniki, ki omogočajo generativno modeliranje. Vendar so danes številni izdelki izdelani s tehniko 3D tiskanja, kjer zadostuje modeliranje s prostimi površinami ali zgolj poligonsko modeliranje. Zato je pomembna izbira primernega modelirnika in tudi pri tej odločitvi bo ta učbenik nekoliko v pomoč.

Literatura

[1] Stratton, G. M. (1896). Some preliminary experiments on vision without inversion of the retinal image. Psychological Review, 3(6), 611–617. https://doi.org/10.1037/h0072918

[2] Cajal S. R. Y., Histologie Du Système Nerveux de lHomme et Des Vertébrés, Maloine, Paris, 1911, Retina diagram, Wikipedia.org, CC BY-SA 3.0 [splet], dosegljivo: https://commons.wikimedia.org/wiki/File:Retina-diagram.svg [datum dostopa: 7. 11. 2024].

[3] Tinsley, J., Molodtsov, M. (2016), Prevedel, R. et al. Direct detection of a single photon by humans. Nat Commun 7, 12172. https://doi.org/10.1038/ncomms12172

[4] Clark N. R. (2005). Notes on the Resolution and Other Details of the Human Eye, [splet], dosegljivo: https://clarkvision.com/imagedetail/eye-resolution.html [datum dostopa: 19. 2. 2024].

[5] Mathot S. (2015). Can you see while your eyes move? [splet], dosegljivo: https://www.cogsci.nl/blog/miscellaneous/242-can-you-see-while-your-eyes-move [datum dostopa: 19. 2. 2024].

[6] Seltman W. (2020) How to Find the Blind Spot in Your Eye, [splet], dosegljivo: https://www.webmd.com/eye-health/eye-blind-spot [datum dostopa: 19. 2. 2024].

[7] History of television. [splet], dosegljivo: https://en.wikipedia.org/wiki/History_of_television [datum dostopa: 19. 2. 2024].

[8] Semi-Automatic Ground Environment [splet], dosegljivo: https://en.wikipedia.org/wiki/Semi-Automatic_Ground_Environment [datum dostopa: 19. 2. 2024].

[9] Computer Scope v sistemu USAF SAGE [splet], dosegljivo: https://www.computerhistory.org/revolution/input-output/14/346 [datum dostopa: 19. 2. 2024].

[10] MIT – Massachusetts Institute of Technology [splet], dosegljivo: https://www.mit.edu/ [datum dostopa: 19. 2. 2024].

[11] Ivan Sutherland [splet], dosegljivo: https://en.wikipedia.org/wiki/Ivan_Sutherland [datum dostopa: 19. 2. 2024].

[12] Sketchpad [splet], dosegljivo: https://en.wikipedia.org/wiki/Sketchpad [datum dostopa: 19. 2. 2024].

[13] William Fetter [splet], dosegljivo: https://en.wikipedia.org/wiki/William_Fetter [datum dostopa: 19. 2. 2024].

[14] Boeing Man [splet], dosegljivo: https://secure.boeingimages.com/archive/William-Fetter's-Boeing-Man-2F3XC5YCZNC.html [datum dostopa: 19. 2. 2024].

[15] Vektorska grafika [splet], dosegljivo: https://en.wikipedia.org/wiki/Vector_graphics [datum dostopa: 19. 2. 2024].

[16] Logo [splet], dosegljivo: https://en.wikipedia.org/wiki/Logo_(programming_language) [datum dostopa: 19. 2. 2024].

[17] Ryan Somma, Calcomp drum plotter, Wikipedia.org, CC BY-SA 2.0 [splet], dosegljivo: https://en.wikipedia.org/wiki/Calcomp_plotter [datum dostopa: 19. 2. 2024].

[18] Rasterska grafika [splet], dosegljivo: https://en.wikipedia.org/wiki/Raster_graphics [datum dostopa: 19. 2. 2024].

[19] Anamorfični format [splet], dosegljivo: https://en.wikipedia.org/wiki/Anamorphic_format [datum dostopa: 19. 2. 2024].

[20] LCD zaslon [splet], dosegljivo: https://sl.wikipedia.org/wiki/LCD-zaslon [datum dostopa: 19. 2. 2024].

[21] Kvantna točka https://en.wikipedia.org/wiki/Quantum_dot_display [datum dostopa: 19. 2. 2024].

[22] OLED [splet], dosegljivo: https://en.wikipedia.org/wiki/OLED [datum dostopa: 19. 2. 2024].

[23] MicroLED [splet], dosegljivo: https://en.wikipedia.org/wiki/MicroLED [datum dostopa: 19. 2. 2024].

[24] DLP [splet], dosegljivo: https://sl.wikipedia.org/wiki/Digital_light_processing [datum dostopa: 19. 2. 2024].

[25] TI Pico [splet], dosegljivo: https://www.ti.com/dlp-chip/display-and-projection/pico-chipsets/getting-started.html [datum dostopa: 19. 2. 2024].

[26] RGBA barvni sistem [splet], dosegljivo: https://en.wikipedia.org/wiki/RGBA_color_model [datum dostopa: 19. 2. 2024].

[27] Horst Frank, CMYK, Wikipedia.org, CC BY-SA 3.0, [splet], dosegljivo: https://commons.wikimedia.org/wiki/File:CMYK_farbwuerfel.jpg [datum dostopa: 19. 2. 2024].

[28] Barvni modeli [splet], dosegljivo: https://en.wikipedia.org/wiki/Color_model [datum dostopa: 19. 2. 2024].

[29] Tektronix Plot10 [splet], dosegljivo: https://www.computerhistory.org/collections/catalog/102646117 [datum dostopa: 19. 2. 2024].

[30] X Window System [splet], dosegljivo: https://en.wikipedia.org/wiki/X_Window_System [datum dostopa: 19. 2. 2024].

[31] OpenGL [splet], dosegljivo: https://en.wikipedia.org/wiki/OpenGL [datum dostopa: 19. 2. 2024].

[32] DirectX [splet], dosegljivo: https://en.wikipedia.org/wiki/DirectX [datum dostopa: 19. 2. 2024].

[33] Oldest known drawing [splet] dosegljivo: https://www.cnet.com/science/oldest-known-drawing-from-73000-years-ago-looks-like-a-hashtag/ [datum dostopa: 19. 2. 2024].

[34] The History of Perspective [splet] dosegljivo: http://www.essentialvermeer.com/technique/perspective/history.html [datum dostopa: 19. 2. 2024].

[35] Free Hand Sketching; Its Methods, Instruments, Principles. [splet] dosegljivo: https://civilseek.com/free-hand-sketching/ [datum dostopa: 19. 2. 2024].

[36] How to Draw Perfect Shapes on iPhone and iPad [splet], dosegljivo: https://hitechglitz.com/how-to-draw-perfect-shapes-on-iphone-and-ipad/ [datum dostopa: 19. 2. 2024].

[37] Luciano Testoni, Staircase perspective, 1995, Wikipedia.org, [splet], dosegljivo: https://en.wikipedia.org/wiki/Perspective_(graphical) [datum dostopa: 19. 2. 2024].

[38] Almondox, 77700 Chessy, France (Disneyland Paris main street), 2014, Wikipedia.org, CC BY-A 3.0 [splet], dosegljivo: https://commons.wikimedia.org/wiki/File:77700_Chessy,_France_-_panoramio_(10).jpg [datum dostopa: 19. 2. 2024].

[39] Parallel projection [splet] dosegljivo: https://en.wikipedia.org/wiki/Parallel_projection [datum dostopa: 19. 2. 2024].

[40] Isometric projection [splet], dosegljivo: https://en.wikipedia.org/wiki/Isometric_projection [datum dostopa: 19. 2. 2024].

[41] La Sagrada Familia [splet], dosegljivo: https://99percentinvisible.org/episode/la-sagrada-familia-2/ [datum dostopa: 19. 2. 2024].

[42] Rotacijske matrike [splet], dosegljivo: https://en.wikipedia.org/wiki/Rotation_matrix [datum dostopa: 19. 2. 2024].

[43] Quaternioni [splet], dosegljivo: https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation [datum dostopa: 19. 2. 2024].

[44] 3D projection [splet], dosegljivo: https://jsantell.com/3d-projection/ [datum dostopa: 19. 2. 2024].

[45] Marschner S., Shirley P. (2015). Fundamentals of Computer Graphics 4th Edition, A K Peters/CRC Press, https://doi.org/10.1201/9781315372198 .

[46] NovaLogic [splet], dosegljivo: https://en.wikipedia.org/wiki/NovaLogic [datum dostopa: 19. 2. 2024].

[47] White Timberwolf, Octree, 2010, Wikipedia.org, CC BY-A 3.0 [splet], dosegljivo: https://en.wikipedia.org/wiki/Octree [datum dostopa: 19. 2. 2024].

[48] CloudCompare octree, 2015, Gnu Free Documentation License 1.2 [splet], dosegljivo: https://www.cloudcompare.org/doc/wiki/index.php/CloudCompare_octree [datum dostopa: 11. 11. 2024].

[49] Minecraft [splet], dosegljivo: https://www.minecraft.net/en-us [datum dostopa: 19. 2. 2024].

[50] Constructive solid geometry [splet], dosegljivo: https://en.wikipedia.org/wiki/Constructive_solid_geometry [datum dostopa: 19. 2. 2024].

[51] Pandakekok9, 2016, CSG tree, Wikipedia.org, CC BY-A 3.0 [splet], dosegljivo: https://en.wikipedia.org/wiki/Constructive_solid_geometry#/media/File:Csg_tree.png [datum dostopa: 19. 2. 2024].

[52] Euler characteristic [splet], dosegljivo: https://en.wikipedia.org/wiki/Euler_characteristic [datum dostopa: 19. 2. 2024].

[53] Euler-Poincare formula [splet], dosegljivo: https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/model/euler.html [datum dostopa: 19. 2. 2024].

[54] Boundary representation [splet], dosegljivo: https://en.wikipedia.org/wiki/Boundary_representation [datum dostopa: 19. 2. 2024].

[55] Bellocchio, F., Borghese, N. A., Ferrari, S., Piuri, V. (2012). 3D Surface Reconstruction: Multi-Scale Hierarchical Approaches. Springer Science & Business Media, 2012. https://doi.org/10.1007/978-1-4614-5632-2_7

[56] 3D Scanning for Personal 3D Printing [splet], dosegljivo: http://mesh.brown.edu/desktop3dscan/ch5-structured.html [datum dostopa: 19. 2. 2024].

[57] Trimensional [splet], dosegljivo: http://www.trimensional.com/ [datum dostopa: 19. 2. 2024].

[58] Kinect [splet], dosegljivo: https://en.wikipedia.org/wiki/Kinect [datum dostopa: 19. 2. 2024].

[59] Kinect 3D scanner [splet], dosegljivo: https://all3dp.com/2/kinect-3d-scanner-easy-beginner-tutorial/ [datum dostopa: 19. 2. 2024].

[60] ReconstructMe [splet], dosegljivo: https://www.reconstructme.net/reconstructme-selfies-displayed-on-3d-screen/ [datum dostopa: 19. 2. 2024].

[61] Parallax [splet], dosegljivo: https://en.wikipedia.org/wiki/Parallax [datum dostopa: 19. 2. 2024].

[62] Digital image corellation and tracking [splet], dosegljivo: https://en.wikipedia.org/wiki/Digital_image_correlation_and_tracking [datum dostopa: 19. 2. 2024].

[63] Fotogrametrija [splet], dosegljivo: https://all3dp.com/1/best-photogrammetry-software/ [datum dostopa: 19. 2. 2024].

[64] LIDAR [splet], dosegljivo: https://en.wikipedia.org/wiki/Lidar [datum dostopa: 19. 2. 2024].

[65] CT scan [splet], dosegljivo: https://en.wikipedia.org/wiki/CT_scan [datum dostopa: 19. 2. 2024].

[66] Magnetic resonance imaging [splet], dosegljivo: https://en.wikipedia.org/wiki/Magnetic_resonance_imaging [datum dostopa: 19. 2. 2024].

[67] Fiji [splet], dosegljivo: https://imagej.net/software/fiji/ [datum dostopa: 19. 2. 2024].

[68] Voronoi diagram [splet], dosegljivo: https://en.wikipedia.org/wiki/Voronoi_diagram [datum dostopa: 19. 2. 2024].

[69] Delaunay triangulation [splet], dosegljivo: https://cartography-playground.gitlab.io/playgrounds/triangulation-delaunay-voronoi-diagram/ [datum dostopa: 19. 2. 2024].

[70] Cartography Playground [splet], dosegljivo: https://cartography-playground.gitlab.io/playgrounds/triangulation-delaunay-voronoi-diagram/ [datum dostopa: 19. 2. 2024].

[71] Decimacija [splet], dosegljivo: https://www.vntana.com/blog/what-is-mesh-decimation-and-why-is-it-vital-for-ar/ [datum dostopa: 19. 2. 2024].

[72] Smoothing [splet], dosegljivo: https://en.wikipedia.org/wiki/Smoothing [datum dostopa: 19. 2. 2024].

[73] STL [splet], dosegljivo: https://en.wikipedia.org/wiki/STL_(file_format) [datum dostopa: 19. 2. 2024].

[74] Blender (software) [splet], dosegljivo: https://en.wikipedia.org/wiki/Blender_(software) [datum dostopa: 19. 2. 2024].

[75] Autocad DXF [splet], dosegljivo: https://en.wikipedia.org/wiki/AutoCAD_DXF [datum dostopa: 19. 2. 2024].

[76] Autodesk 3D Studio Max [splet], dosegljivo: https://en.wikipedia.org/wiki/Autodesk_3ds_Max [datum dostopa: 19. 2. 2024].

[77] Autodesk Maya [splet], dosegljivo: https://www.autodesk.com/products/maya/overview?term=1-YEAR&tab=subscription [datum dostopa: 19. 2. 2024].

[78] DAE Collada [splet], dosegljivo: https://en.wikipedia.org/wiki/COLLADA [datum dostopa: 19. 2. 2024].

[79] Wavefront OBJ [splet], dosegljivo: https://en.wikipedia.org/wiki/Wavefront_.obj_file [datum dostopa: 19. 2. 2024].

[80] PLY (file format) [splet], dosegljivo: https://en.wikipedia.org/wiki/PLY_(file_format) [datum dostopa: 19. 2. 2024].

[81] Cinema 4D [splet], dosegljivo: https://en.wikipedia.org/wiki/Cinema_4D [datum dostopa: 19. 2. 2024].

[82] VRML [splet], dosegljivo: https://en.wikipedia.org/wiki/VRML [datum dostopa: 19. 2. 2024].

[83] 3DXML [splet], dosegljivo: https://en.wikipedia.org/wiki/3DXML [datum dostopa: 19. 2. 2024].

[84] HTML5 [splet], dosegljivo: https://en.wikipedia.org/wiki/HTML5 [datum dostopa: 19. 2. 2024].

[85] Web based 3D modelling [splet], dosegljivo: https://manufactur3dmag.com/popular-free-online-web-based-3d-modelling-software/ [datum dostopa: 19. 2. 2024].

[86] 3D Manufacturing Format [splet], dosegljivo: https://en.wikipedia.org/wiki/3D_Manufacturing_Format [datum dostopa: 19. 2. 2024].

[87] Superelipsoid [splet], dosegljivo: https://en.wikipedia.org/wiki/Superellipsoid [datum dostopa: 19. 2. 2024].

[88] Kindlmann G. (2004). Superquadrtic Tensor Glyphs. Joint EUROGRAPHICS – IEEE TCVG Symposium on Visualization (2004) 147–154. http://dx.doi.org/10.2312/VisSym/VisSym04/147-154.

[89] Blinn J. F.(1982). A generalisation of algebraic surface drawing, ACM Trans. Graph, 1(3) (1982) 235–256.

[90] Muraki S: Volumetric shape description of range data using "Blobby Model". Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91), July–August 1991, Las Vegas, Nev, USA 227–235.

[91] Marching Cubes [splet], dosegljivo: https://en.wikipedia.org/wiki/Marching_cubes [datum dostopa: 19. 2. 2024].

[92] Roblox solid modelling [splet] https://create.roblox.com/docs/parts/solid-modeling, dosegljivo: [datum dostopa: 19. 2. 2024].

[93] Spline weights [splet], dosegljivo: https://edsonmarine.com/products/home-accessories/spline-weights/ [datum dostopa: 19. 8. 2024].

[94] Bézier curve [splet], dosegljivo: https://en.wikipedia.org/wiki/B%C3%A9zier_curve [datum dostopa: 19. 2. 2024].

[95] De Casteljau [splet], dosegljivo: https://en.wikipedia.org/wiki/De_Casteljau%27s_algorithm [datum dostopa: 19. 2. 2024].

[96] Bernstein polynomial [splet] dosegljivo: https://en.wikipedia.org/wiki/Bernstein_polynomial [datum dostopa: 19. 2. 2024].

[97] B-spline [splet], dosegljivo: https://en.wikipedia.org/wiki/B-spline [datum dostopa: 19. 2. 2024].

[98] Carl R. de Boor [splet], dosegljivo: https://en.wikipedia.org/wiki/Carl_R._de_Boor [datum dostopa: 19. 2. 2024].

[99] ChatGPT. (2021). OpenAI, dosegljivo: https://openai.com/chatgpt [datum dostopa: 19. 2. 2024].

[100] NURBS krivulje [splet], dosegljivo: https://en.wikipedia.org/wiki/Non-uniform_rational_B-spline [datum dostopa: 19. 2. 2024].

[101] Stožnice [splet], dosegljivo: https://en.wikipedia.org/wiki/Conic_section [datum dostopa: 19. 2. 2024].

[102] Rillke, Conics sections, Wikipedia.org, CC BY-A 4.0 [splet], dosegljivo: https://en.wikipedia.org/wiki/File:Conic_Sections.svg#filelinks [datum dostopa: 14. 11. 2024].

[103] Bezier surface [splet], dosegljivo: https://en.wikipedia.org/wiki/B%C3%A9zier_surface [datum dostopa: 19. 2. 2024].

[104] Chrschn, NURBS surface, Wikipedia.org, Public domain [splet], dosegljivo: https://commons.wikimedia.org/wiki/File:NURBS_surface.png [datum dostopa: 19. 2. 2024].

[105] Zveznost površin G0, G1, G2, G3 (Autodesk Alias) [splet], dosegljivo: https://www.aliasworkbench.com/theoryBuilders/TB3_continuity1.htm [datum dostopa: 19. 2. 2024].

[106] Subdivision surface [splet], dosegljivo: https://en.wikipedia.org/wiki/Subdivision_surface [datum dostopa: 19. 2. 2024].

[107] Geri's game [splet], dosegljivo: https://en.wikipedia.org/wiki/Geri%27s_Game [datum dostopa: 19. 2. 2024].

[108] Doo-Sabin [splet], dosegljivo: https://en.wikipedia.org/wiki/Doo%E2%80%93Sabin_subdivision_surface [datum dostopa: 19. 2. 2024].

[109] Catmull-Clark algoritm [splet], dosegljivo: https://en.wikipedia.org/wiki/Catmull%E2%80%93Clark_subdivision_surface [datum dostopa: 19. 2. 2024].

[110] Loop [splet], dosegljivo: https://en.wikipedia.org/wiki/Loop_subdivision_surface [datum dostopa: 19. 2. 2024].

[111] Dyn N., Levine D., Gregory J. A. (1990). A butterfly subdivision scheme for surface interpolation with tension control, ACM Trans. Graph, 9(2) (1990) 160–169 https://doi.org/10.1145/78956.78958.

[112] Zorin, Denis; Schr¨oder, Peter; Sweldens, Wim (1996). *Interpolating Subdivision for Meshes with Arbitrary Topology* (https://cims.nyu.edu/gcl/papers/zorin1996ism.pdf). Department of Computer Science, California Institute of Technology, Pasadena, CA 91125.

[113] Kobbelt glajenje [splet], dosegljivo: https://diglib.eg.org/bitstream/handle/10.2312/egt19981028/Part1.pdf?sequence=1&isAllowed=y [datum dostopa: 19. 2. 2024].

[114] T-spline [splet], dosegljivo: https://en.wikipedia.org/wiki/T-spline [datum dostopa: 19. 2. 2024].

[115] Sangali G., Definition and theory of T-splines for IGA, University of Pavia [splet], dosegljivo: https://indico.ictp.it/event/a12191/session/35/contribution/20/material/0/0.pdf [datum dostopa: 19. 2. 2024].

[116] Subdivision levels [splet], dosegljivo: https://www.youtube.com/watch?v=ckOTl2GcS-E [datum dostopa: 19. 2. 2024].

[117] Surface Extrudes [splet], dosegljivo: https://help.solidworks.com/2021/english/SolidWorks/sldworks/c_Surface_Extrudes.htm?id=007491b06e844014a26d660914917635#Pg0 [datum dostopa: 19. 2. 2024].

[118] Revolved Surface [splet], dosegljivo: https://help.solidworks.com/2021/english/SolidWorks/sldworks/t_Revolved_Surface.htm [datum dostopa: 19. 2. 2024].

[119] Offset Surface [splet], dosegljivo: https://help.solidworks.com/2021/english/SolidWorks/sldworks/t_offset_surface.htm [datum dostopa: 19. 2. 2024].

[120] Swept Surface [splet], dosegljivo: https://help.solidworks.com/2021/english/SolidWorks/sldworks/t_Swept_Surfaces.htm [datum dostopa: 19. 2. 2024].

[121] Lofted surface [splet], dosegljivo: https://help.solidworks.com/2022/english/solidworks/sldworks/t_lofted_surface.htm [datum dostopa: 19. 2. 2024].

[122] Filled Surface [splet], dosegljivo: https://help.solidworks.com/2022/english/SolidWorks/sldworks/c_Filled_Surface.htm [datum dostopa: 19. 2. 2024].

[123] Boundary surface [splet], dosegljivo: https://help.solidworks.com/2022/english/SolidWorks/sldworks/c_Boundary_Surface_Concept_surfaces.htm [datum dostopa: 19. 2. 2024].

[124] Knit Surface [splet], dosegljivo: https://help.solidworks.com/2021/english/SolidWorks/sldworks/hidd_dve_knit_surface.htm [datum dostopa: 19. 2. 2024].

[125] Trim Surface [splet], dosegljivo: https://help.solidworks.com/2021/english/SolidWorks/sldworks/hidd_feat_trim_ref_surface.htm [datum dostopa: 19. 2. 2024].

[126] CATIA [splet], dosegljivo: https://www.3ds.com/products/catia [datum dostopa: 19. 2. 2024].

[127] NX [splet], dosegljivo: https://plm.sw.siemens.com/en-US/nx/ [datum dostopa: 19. 2. 2024].

[128] Creo [splet], dosegljivo: https://www.ptc.com/en/products/creo [datum dostopa: 19. 2. 2024].

[129] Inventor [splet], dosegljivo: https://www.autodesk.com/products/inventor/overview?term=1-YEAR&tab=subscription [datum dostopa: 19. 2. 2024].

[130] Solidworks [splet], dosegljivo: https://www.solidworks.com/ [datum dostopa: 19. 2. 2024].

[131] Solidedge [splet], dosegljivo: https://solidedge.siemens.com/en/ [datum dostopa: 19. 2. 2024].

[132] Onshape [splet], dosegljivo: https://www.onshape.com/en/ [datum dostopa: 19. 2. 2024].

[133] Fusion 360 [splet], dosegljivo: https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR&tab=subscription [datum dostopa: 19. 2. 2024].

[134] Rhinoceros [splet], dosegljivo: https://www.rhino3d.com/ [datum dostopa: 19. 2. 2024].

[135] Haptične naprave [splet], dosegljivo: [datum dostopa: 19. 2. 2024].

[136] Solidworks Freeform [splet], dosegljivo: https://www.youtube.com/watch?v=fuOtGyuuKnQ [datum dostopa: 19. 2. 2024].

[137] SubD surface [splet], dosegljivo: https://www.rhino3d.com/features/subd/ [datum dostopa: 19. 2. 2024].

[138] SubD mouse [splet], dosegljivo: https://wiki.mcneel.com/7/tutorials/subd/mouse [datum dostopa: 19. 2. 2024].

[139] 3DExperience xShape [splet], dosegljivo: https://www.solidworks.com/media/3d-sculptor-getting-started-xshape [datum dostopa: 19. 2. 2024].

[140] Gravity Sketch [splet], dosegljivo https://www.gravitysketch.com/ [datum dostopa: 19. 2. 2024].

[141] History of CAD – Parametric Technology Corporation [splet], dosegljivo: https://www.shapr3d.com/history-of-cad/parametric-technology-corporation [datum dostopa: 19. 2. 2024].

[142] History of parametric [splet], dosegljivo: https://www.danieldavis.com/a-history-of-parametric/ [datum dostopa: 19. 2. 2024].

[143] Jurij Gričnik, Parametrično modeliranje mostnega žerjava, diplomsko delo, Univerza v Mariboru, Fakulteta za strojništvo, 2020.

[144] Jernej Kojzek, Parametrično modeliranje gorskega kolesa, diplomsko delo, Univerza v Mariboru, Fakulteta za strojništvo, 2023.

[145] Emilio M. Sanfilippo, Feature-based product modelling: an ontological approach, International Journal of Computer Integrated Manufacturing, 31:11, 1097-1110, DOI: https://doi.org/10.1080/0951192X.2018.1497814 , 2018.

[146] Direct modelling [splet], dosegljivo: https://www.shapr3d.com/blog/how-does-direct-modeling-actually-work [datum dostopa: 19. 2. 2024].

[147] SpaceClaim [splet], dosegljivo: https://www.ansys.com/products/3d-design/ansys-spaceclaim [datum dostopa: 19. 2. 2024].

[148] Shapr3D [splet], dosegljivo: https://www.shapr3d.com/ [datum dostopa: 19. 2. 2024].

[149] Glossary – Hybrid Modeling [splet], dosegljivo: https://www.spatial.com/resources/glossary/what-is-hybrid-modeling [datum dostopa: 19. 2. 2024].

[150] Matt Lombard, SolidWorks Surfacing and Complex Shape Modeling Bible, Wiley, 2008.

[151] Renderiranje [splet], dosegljivo: https://en.wikipedia.org/wiki/Rendering_(computer_graphics) [datum dostopa: 19. 2. 2024].

[152] Clipping (computer graphics) [splet], dosegljivo: https://en.wikipedia.org/wiki/Clipping_(computer_graphics) [datum dostopa: 19. 2. 2024].

[153] Back-face culling [splet], dosegljivo: https://en.wikipedia.org/wiki/Back-face_culling [datum dostopa: 19. 2. 2024].

[154] Hidden-line removal [splet], dosegljivo: https://en.wikipedia.org/wiki/Hidden-line_removal [datum dostopa: 19. 2. 2024].

[155] Painter's algorithm [splet], dosegljivo: https://en.wikipedia.org/wiki/Painter%27s_algorithm [datum dostopa: 19. 2. 2024].

[156] Hidden-surface determination [splet], dosegljivo: https://en.wikipedia.org/wiki/Hidden-surface_determination [datum dostopa: 19. 2. 2024].

[157] Z-buffering [splet], dosegljivo: https://en.wikipedia.org/wiki/Z-buffering [datum dostopa: 19. 2. 2024].

[158] Wolfgang Strasser [splet], dosegljivo: https://cgvr.cs.uni-bremen.de/miscellaneous/strasser/Wolfgang_Strasser_Schnelle_Kurven-_und_Flaechendarstellung_auf_grafischen_Sichtgeraeten.pdf [datum dostopa: 19. 2. 2024].

[159] Edwin Earl Catmull [splet], dosegljivo: https://static1.1.sqspcdn.com/static/f/552576/6419248/1270507173137/catmull_thesis.pdf [datum dostopa: 19. 2. 2024].

[160] Z-buffer delovanje [splet], dosegljivo: https://www.geeksforgeeks.org/z-buffer-depth-buffer-method/ [datum dostopa: 19. 2. 2024].

[161] List of common shading algorithms [splet], dosegljivo: https://en.wikipedia.org/wiki/List_of_common_shading_algorithms [datum dostopa: 19. 2. 2024].

[162] Lambertian reflectance [splet], dosegljivo: https://en.wikipedia.org/wiki/Lambertian_reflectance [datum dostopa: 19. 2. 2024].

[163] Specular highlight [splet], dosegljivo: https://en.wikipedia.org/wiki/Specular_highlight [datum dostopa: 19. 2. 2024].

[164] Gouraud shading [splet], dosegljivo: https://en.wikipedia.org/wiki/Gouraud_shading [datum dostopa: 19. 2. 2024].

[165] Phong reflection model [splet], dosegljivo: https://en.wikipedia.org/wiki/Phong_reflection_model [datum dostopa: 19. 2. 2024].

[166] Bump mapping [splet], dosegljivo: https://en.wikipedia.org/wiki/Bump_mapping [datum dostopa: 19. 2. 2024].

[167] Ray tracing [splet], dosegljivo: https://en.wikipedia.org/wiki/Ray_tracing_(graphics) [datum dostopa: 19. 2. 2024].

[168] Whitted T. J., A Ray-Tracing Pioneer Explains How He Stumbled into Global Illumination [splet], dosegljivo: https://blogs.nvidia.com/blog/ray-tracing-global-illumination-turner-whitted/ [datum dostopa: 19. 2. 2024].

[169] Tipi sledenja žarku [splet], dosegljivo: https://cs.stanford.edu/people/eroberts/courses/soco/projects/1997-98/ray-tracing/types.html [datum dostopa: 19. 2. 2024].

[170] Radiosity [splet], dosegljivo: https://en.wikipedia.org/wiki/Radiosity_(computer_graphics) [datum dostopa: 19. 2. 2024].

[171] Photon mapping [splet], dosegljivo: https://en.wikipedia.org/wiki/Photon_mapping [datum dostopa: 19. 2. 2024].

[172] Renderman Pixar [splet], dosegljivo: https://renderman.pixar.com/ [datum dostopa: 19. 2. 2024].

[173] Keyshot [splet], dosegljivo: https://www.keyshot.com/ [datum dostopa: 19. 2. 2024].

[174] Vray [splet], dosegljivo: https://www.chaos.com/ [datum dostopa: 19. 2. 2024].

[175] Povray [splet], dosegljivo: https://www.povray.org/ [datum dostopa: 19. 2. 2024].

[176] Zoetrope [splet], dosegljivo: https://en.wikipedia.org/wiki/Zoetrope [datum dostopa: 19. 2. 2024].

[177] Zoetrope video [splet], dosegljivo: https://www.youtube.com/watch?v=3-rPn0a56WE [datum dostopa: 19. 2. 2024].

[178] Walt Disney Company [splet], dosegljivo: https://en.wikipedia.org/wiki/The_Walt_Disney_Company [datum dostopa: 19. 2. 2024].

[179] John Lasseter [splet], dosegljivo: https://en.wikipedia.org/wiki/John_Lasseter [datum dostopa: 19. 2. 2024].

[180] CGI [splet], dosegljivo: https://en.wikipedia.org/wiki/Computer-generated_imagery [datum dostopa: 19. 2. 2024].

[181] Stop motion [splet], dosegljivo: https://en.wikipedia.org/wiki/Stop_motion [datum dostopa: 19. 2. 2024].

[182] GIF [splet], dosegljivo: https://en.wikipedia.org/wiki/GIF [datum dostopa: 19. 2. 2024].

[183] Key frame [splet], dosegljivo: https://en.wikipedia.org/wiki/Key_frame [datum dostopa: 19. 2. 2024].

[184] Keyframe interpolation [splet], dosegljivo: https://helpx.adobe.com/after-effects/using/keyframe-interpolation.html [datum dostopa: 19. 2. 2024].

[185] Motion capture [splet], dosegljivo: https://en.wikipedia.org/wiki/Motion_capture [datum dostopa: 19. 2. 2024].

[186] AVI video zapis [splet], dosegljivo: https://en.wikipedia.org/wiki/Audio_Video_Interleave [datum dostopa: 19. 2. 2024].

[187] MPEG [splet], dosegljivo: https://en.wikipedia.org/wiki/Moving_Picture_Experts_Group [datum dostopa: 19. 2. 2024].

[188] MP4 [splet], dosegljivo: https://en.wikipedia.org/wiki/MP4_file_format [datum dostopa: 19. 2. 2024].

[189] Matroska [splet], dosegljivo: https://en.wikipedia.org/wiki/Matroska [datum dostopa: 19. 2. 2024].

[190] Video codec [splet], dosegljivo: https://en.wikipedia.org/wiki/Video_codec [datum dostopa: 19. 2. 2024].

[191] Videoformati in kodeki [splet], dosegljivo: https://en.wikipedia.org/wiki/Comparison_of_video_container_formats [datum dostopa: 19. 2. 2024].

[192] FFmpeg [splet], dosegljivo: https://en.wikipedia.org/wiki/FFmpeg [datum dostopa: 19. 2. 2024].

[193] Adobe Premiere Pro [splet], dosegljivo: https://www.adobe.com/si/products/premiere.html [datum dostopa: 19. 2. 2024].

[194] DaVinci Resolve [splet], dosegljivo: https://www.blackmagicdesign.com/products/davinciresolve [datum dostopa: 19. 2. 2024].

[195] Final Cut Pro [splet], dosegljivo: https://www.apple.com/final-cut-pro/ [datum dostopa: 19. 2. 2024].

[196] VSDC Free Video Editor [splet], dosegljivo: https://www.videosoftdev.com/ [datum dostopa: 19. 2. 2024].

[197] Virtual reality [splet], dosegljivo: https://en.wikipedia.org/wiki/Virtual_reality [datum dostopa: 19. 2. 2024].

[198] Augmented reality [splet], dosegljivo: https://en.wikipedia.org/wiki/Augmented_reality [datum dostopa: 19. 2. 2024].

[199] Lenticular printing [splet], dosegljivo: https://en.wikipedia.org/wiki/Lenticular_printing [datum dostopa: 19. 2. 2024].

[200] 3D gif [splet], dosegljivo: https://giphy.com/explore/lenticular-3d [datum dostopa: 19. 2. 2024].

[201] Stereoskopska slika z napenjanjem oči [splet], dosegljivo: https://en.wikipedia.org/wiki/Autostereogram#/media/File:Stereogram_Tut_Random_Dot_Shark.png [datum dostopa: 19. 8. 2024].

[202] Stereoskop [splet], dosegljivo: https://en.wikipedia.org/wiki/Stereoscope [datum dostopa: 19. 2. 2024].

[203] Davepape, Holmes Stereoscope, Wikipedia.org, CC0 1.0 [splet], dosegljivo: https://commons.wikimedia.org/wiki/File:Holmes_stereoscope.jpg [datum dostopa: 19. 2. 2024].

[204] Anaglyph 3d [splet], dosegljivo: https://en.wikipedia.org/wiki/Anaglyph_3D [datum dostopa: 19. 2. 2024].

[205] Pasivna stereoskopija [splet], dosegljivo: https://www.expertreviews.co.uk/tvs-entertainment/8074/3d-tv-explained-passive-active-glasses-and-glasses-free-3d [datum dostopa: 19. 2. 2024].

[206] Holografija [splet], dosegljivo: https://en.wikipedia.org/wiki/Holography [datum dostopa: 19. 2. 2024].

[207] Epzcaw, Broken hologram, Wikipedia.org, CC-BY-SA-3.0 [splet], dosegljivo: https://sl.wikipedia.org/wiki/Holografija#/media/Slika:Broken_hologram.jpg [datum dostopa: 19. 8. 2024].

[208] Pepper Ghost [splet], dosegljivo: https://en.wikipedia.org/wiki/Pepper%27s_ghost [datum dostopa: 19. 2. 2024].

[209] Karthikch98, Pyramid holographic 3D holographic projection phone projector 3D holographic projection 3D mobile phone naked eye 3D pyramid, Wikipedia.org, CC-BY-SA-4.0 [splet], dosegljivo: https://commons.wikimedia.org/wiki/File:Pyramid_holographic_3D_holographic_projection_phone_projector_3D_holographic_projection_3D_mobile_phone_naked_eye_3D_pyramid.jpg [datum dostopa: 19. 2. 2024].

[210] Widescreen holographic box for smartphones [splet], dosegljivo: https://cults3d.com/en/3d-model/gadget/widescreen-holographic-box-for-smartphones [datum dostopa: 19. 2. 2024].

[211] CAVE [splet], dosegljivo: https://en.wikipedia.org/wiki/Cave_automatic_virtual_environment [datum dostopa: 19. 2. 2024].

[212] CAVE VisBox [splet], dosegljivo: http://www.visbox.com/products/cave/viscube-c4/ [datum dostopa: 19. 2. 2024].

[213] Davepape, CAVE Crayoland, Wikipedia.org. PD [splet], dosegljivo: https://commons.wikimedia.org/wiki/File:CAVE_Crayoland.jpg [datum dostopa: 19. 8. 2024].

[214] Meta Quest 3 [splet], dosegljivo: https://www.meta.com/quest/quest-3/ [datum dostopa: 19. 2. 2024].

[215] Team work Gravity Sketch [splet], dosegljivo: https://www.yankodesign.com/2020/07/20/Sgravity-sketch-launches-collaborative-tool-to-let-remote-teams-design-together-in-virtual-reality-wfh/ [datum dostopa: 19. 2. 2024].

3D modeliranje

Napredne tehnike in aplikacije v računalniško podprtem modeliranju

MIRAN ULBIN

Povzetek Skripta s področja 3D modeliranja je namenjena kot študijski pripomoček pri izvedbi predavanj predmetov 3D modeliranje in Računalniško oblikovanje. Vsebuje razlago večjega dela učne snovi, ki jo morajo študentje pri teh predmetih osvojiti, in je skladna z učnim načrtom omenjenih predmetov.

Ključne besede: • 3D modeliranje • računalniško podprto konstruiranje • površinsko modeliranje • vizualizacija • računalniška grafika

NASLOV AVTORJA: dr. Miran Ulbin, Univerza v Mariboru, Fakulteta za strojništvo, Maribor, Slovenija,

DOI https://doi.org/10.18690/um.fs.7.2024 ISBN 978-961-286-930-4 (pdf)

Dostopno na: http://press.um.si

Univerza v Mariboru

Fakulteta za strojništvo