3D animatsioon

3D (kolmemõõtmeline) ehk ruumiline animeerimine toimub põhimõtteliselt samadel alustel nagu 2D animatsioon, kasutatakse *keyframe*'e, *tweening*'ut, morfimist jne.

Kuigi esineb sarnasusi ja samasid termineid, esineb loomulikult ka palju erinevusi, isegi terminoloogias. Näiteks 3D terminoloogias on objekti liigutamine 3-mõõtmelises ruumis – *translation* ning objekti pööramine (*rotate*) – *transformation*.

Arusaadavatel põhjustel ei saa kasutada kihilist animeerimist (*cel animation*). Animatsiooni loomisel on vajalik hea ruumiline ettekujutus ning erinevalt 2D animeerimisest on tegemist väga töömahuka modelleerimistööga ning renderdamiseks kulub kümneid kuni sadu kordi rohkem aega.

3D modelleerimine

3D modelleerimine on ruumiliste objektide (3D mudelite) loomine tasapinnalistest (2D) objektidest (*loft*) või lihtsatest 3D objektidest ehk primitiividest nende kombineerimise (*Boolean operations*) ning deformeerimise teel.

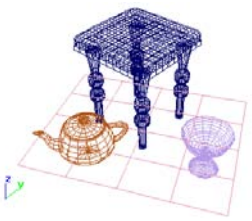
Mudelite kujutamiseks (*represent*) on kasutusel kaks võimalust:

- hulknurkne modelleerimine (*polygonal modeling*) – XYZ koordinaatidega määratakse hulk tippe (*vertices*), mis omavahel sirgetega ühendatakse. Nii moodustuvad hulknurkadest koosnevad kontuurid (*polygonal mesh*).
- NURBS modelleerimine (*Non Uniform Rational Basis Spline* ehk *Bézier Spline*) – juhtpunkte ette andes ja nende "kaalu" (*weight*) määrates moodustatakse kõverad (*curves*). Need kõverad juhinduvad etteantud punktidest kuid ei pea neid ilmtingimata interpoleerima (läbima). Juhtpunkti "kaalu" suurendamine tõmbab kõvera antud punktile lähemale. NURBS modelleerimist kasutatakse peamiselt orgaanilise modelleerimise (*organic modelling*) puhul.

Modelleerimise käigus luuakse objektid, mida hiljem 3D stseenis kasutatakse. Olemas on hulk erinevaid modelleerimistehnikaid, näiteks:

- *constructive solid geometry* – mille käigus luuakse keerukamaid objekte lihtsate objektide kombineerimise teel.
- *implicit surfaces* – mille käigus luuakse objekte kõveraid ja pindasid (*surfaces*) luues ja deformeerides.
- *subdivision surfaces* – mille käigus siledad pinnad luuakse jämedakoeliste hulknurkadest koosnevate kontuuride kaudu nende tahkusid iteratiivselt osadeks jagades.

Traatmudel ehk *wireframe*



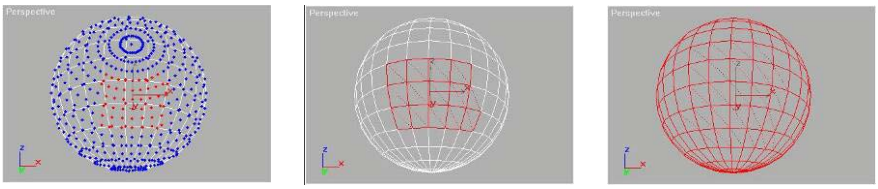
3D modelleerimise ja animeerimise käigus tehakse aja ning arvutuste mahu kokkuhoidmiseks suurem osa tööd stseeni traatmudeliga (*wireframe*). Sellise sõrestiku suurendamine/vähendamine, pööramine jms ei nõua arvutilt eriti suurt tööd, samas nõuab see animeerijalt üsnagi head ruumilist ettekujutust.
Traatmudeliga töötades puutume kokku järgmiste mõistetega:

Vertex, face, mesh

Vertex on tegelikult punkt ehk tipp, mille asukoha ruumis antud koordinaatsüsteemi suhtes määravad kolm arvu. *Vertex* defineeritaksegi sageli just selliste kolme arvu (koordinaadi) kogumina.
Niisugused punktid on 3D objektide ehituskivideks, nende abil määratakse pinnad (*face*) ja lõpuks terved võrgustikud (*mesh*).

Face on kolme või enama omavahel seotud punkti (*vertex*) poolt moodustatud väike kolmnurkne pind ehk tahk. Enamus 3D programme kasutab kolmnurkseid tahkusid kuid mõned kasutavad ka nelinurkseid.
Iga sellise tahuga on seotud veel ka mõiste normaal (*normal*), mis määrab, kumb pool tahust on objekti sisepinnal ja kumb välispinnal.
Sageli nimetatakse tahkusid ka polügoonideks (*polygon*).

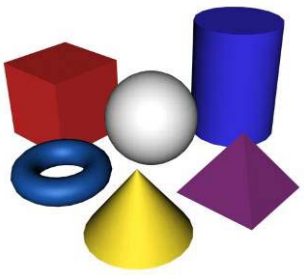
Mesh ehk võrgustik on tahkude kogumik, mis kirjeldab objekti. Objekt võib olla sfäär, tool, auto või mistahes muu asi.



vertex *face* *mesh*

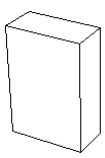
Primitiivid

Enamus objekte meid ümbritsevas keskkonnas koosnevad lihtsatest objektidest. Praktiliselt kõik 3D programmid pakuvad kasutajale hulka lihtsaid objekte ehk primitiive (*primitives*), milledest saab koostada keerukamaid objekte. Primitiivide hulka kuuluvad tavaliselt sfäär (*sphere*), risttahukas (*box*), silinder (*cylinder*), koonus (*cone*), püramiid (*pyramid*) ja rõngas (*torus*). Mitmed programmid pakuvad lisaks veel teisigi primitiive nagu näiteks toru (*tube*) või poolkera (*hemisphere*).

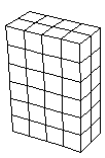


Leidub programme, mis pakuvad primitiivide seas välja ka selliseid huvitavaid objekte nagu näiteks teekann (3DMax), põhjuseks asjaolu, et teekann (*Utah teapot*) oli esimene kompleksne 3D mudel, mis loodi 1975. aastal Martin Newell'i poolt Utah'i Ülikoolis ja on muutunud üheks 3D etalon-objektiks.

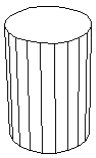
Primitiividel saab määrata ka segmentide (*segments*) arvu ehk mitmeks osaks primitiivi pinnad jagatakse.



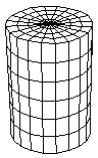
Risttahukas, kõik küljed 1 segmendiga



Sama risttahukas, küljed 2, 4 ja 6 segmendiga



Silinder, 1 kõrguse ja 1 kaane segment



Silinder, 12 kõrguse ja 3 kaane segmenti

Segmentide arv on oluline mitmesuguste operatsioonide, nagu väänamised jms, rakendamisel objektidele. Mida rohkem segmente, seda sujuvamalt saab objekti moonutada. Samas kasvatab segmentide hulk ka failimahtu.

Loogikatehted objektidega

Primitiivide ja teiste objektide kombineerimiseks kasutatakse loogikatehteid (*Boolean operations*) mida on kolm peamist tüüpi:

- liitmine (*addition* või ka *union*), mille korral on tulemuseks kahe esialgse objekti summa, objektid oleks nagu kokku sulatatud;
- lahutamine (*subtraction*), mille korral on tulemuseks teise objekti kujuline auk esimeses objektis, esimesest objektist eemaldatakse nende ühisosa;
- lõige (*intersection*), mille korral on tulemuseks objektide ühisosa.

2 objekti

liidetud objektid

objektide lahutamise üks tulemus

objektide lahutamise teine tulemus

objektide ühisosa

Objektide deformeerimine

Nii primitiive kui ka keerulisemaid objekte saab deformeerida mitmete vahenditega, mida mõnes programmis nimetatakse modifitseerijateks (*modifiers*) ja teistes objekti operatsioonideks (*object operations*). Igal deformatsioonil on omad parameetrid. Paremad programmid võimaldavad määrata ka objekti piirkonna, mille ulatuses deformatsioon toimub (*upper limit* ja *lower limit*).

Deformeerimiseks on tavaliselt järgmised vahendid:

- Painutus (*bend*) võimaldab objekti soovitud nurga all kõveraks painutada. Tavaliselt saab määrata, mitme nurgakraadi võrra painutatakse (*angle*) ja painutuse suunda (*direction*).

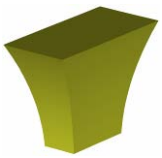
Risttahukas enne deformeerimist

Risttahukas painutatud 90°, kuna kõrguse segmente on vaid 4, siis on deformatsioon nurgeline

Sama risttahukas sama paindega, kõrguse segmente 25

- Ahendamine/laiendamine (*taper*) muudab objekti ühest otsast kitsamaks või laiemaks. Määrata saab deformatsiooni ulatust (*amount*). Paremad programmid võimaldavad määrata ka ahenemis/laienemisjoone kõverust (*curve*).

laiendamine ilma lisaparameetriteta



laiendamine negatiivse kõverusega

laiendamine piiranguga ülevalt ja alt (*upper limit* ja *lower limit*)

- Kallutamine (*skew*) laseb objekti kallutada etteantud ulatuses (*amount*) ja suunas (*direction*).

kallutamine ilma lisaparameetriteta

kallutamine piiranguga ülevalt (*upper limit*)

- Vääne (*twist*) laseb objekti väänata kruvi laadseks. Ette saab anda nurgakraadid (*angle*) ja kalde (*bias*).

vääne ilma lisaparameetriteta

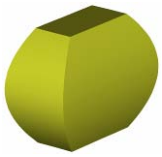
vääne kaldega

vääne piiranguga ülevalt ja alt (*upper limit* ja *lower limit*)

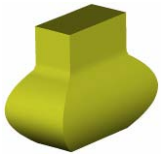
- Venitus (*stretch*) venitab või surub objekti kokku nagu oleks see kummist. Määrata saab venituse ulatust (*stretch*) ja venitusega/kokkusurumisega kaasnevat kitsenemist/laienemist (*amplify*).

venitus

venitus piiranguga ülalt (*upper limit*)



kokku surumine



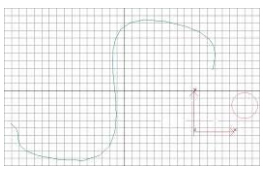
kokku surumine piiranguga ülalt (*upper limit*)

Lofting

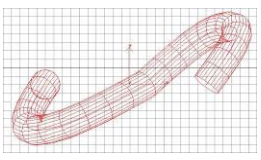
Lofting on tegevus, mille käigus kahemõõtmelised hulknurgad (*2D polygons*) venitatakse kolmemõõtmelisteks. *Loft*'i asemel öeldakse sageli ka *skinning*. Traditsiooniline *lofting* näeb välja nagu kanga venitamine sõrestikule. Tüüpilised objektid, mida *loft* operatsiooni abil luuakse on näiteks paat, lennuki tiib ning surfilaud.
Loft'i korral saab määrata mitmeid parameetreid, näiteks:

- *subdivisions* määrab, mitmest osast koosneb loodav pind kahe joone vahel etteantud sõrestikus ehk kui sujuva tulemuse saame;
- *interpolation* määrab, kas etteantud sõrestiku joonte vahele luuakse sirgjooned (*linear*) või kumerad.
- *caps* määrab, kas loodav objekt tuleb seest õõnes või täidetud.

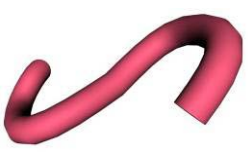
Lofting võib kolmemõõtmelise objekti saada ka mingit tasapinnalist kujundit etteantud joont (*path*) mööda ruumis joonistades. Kasutades mitut erinevat tasapinnalist kujundit võib saada erinevaid, küllaltki keerukaid objekte.



joon ja kujund



loftitud objekti traatmudel

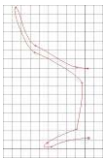


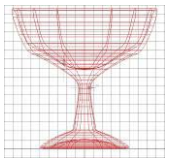
renderdatud tulemus

tulemus kaht erinevat kujundit kasutades

Lathe

Lathe on operatsioon, mille käigus pööratakse kõverjoont ümber vertikaaltelje ja saadakse ruumiline objekt. *Lathe* tähendab tõlkes "treipingil töötlema". Muutes kõverjoone asukohta või orientatsiooni telje suhtes, võib tulemuseks saada palju erinevaid tulemusi. Tüüpilised *lathe* abil loodavad 3D objektid on igasugused vaasid, pokaalid, pudelid jms.

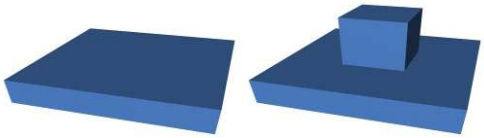

etteantud joon


sõrestik peale *lathe*'i


renderdatud tulemus

Tavaliselt pööratakse joont ümber telje 360°, kuid vajaduse korral saab ka vähem pöörata.
Mõnedes programmides loetakse *lathe* üheks *loft*'i alamoperatsiooniks.

Extrude



Extrude on tõlkes "välja suruma". *Extrude* korral luuakse ruumiline objekt tasapinnalist kujundit mööda mingit etteantud teed korduvalt uuesti joonistades. Etteantud tee võib olla sirge või kõverjoon. *Extrude*'i võib kasutada nii uute objektide loomisel kui ka olemasolevate objektide geomeetria muutmisel surudes võrgustikus (*mesh*) pindasid (*face*) objekti pinnast sisse või väljapoole.
Mõnedes programmides loetakse *extrude* üheks *loft*'i alamoperatsiooniks.

Tekstuurid

Selleks, et renderdamise tulemusena objektidele täiesti realistlikku välimust anda, kasutatakse *surface map*'i ehk tekstuuri (*texture*), mis tegelikult on lihtsalt üks pilt, mis mähitakse ümber objekti. Näiteks võib seina kujutavale objektile asetada telliskividest laotud seina pildi.

Vahetevahel ei anna lihtne tekstuuri kasutamine piisavalt head tulemust. Sellisel juhul kasutatakse lisavõimalusi nagu näiteks *bump map*, *opacity map* vms.


Lihtsalt värvitud


Puidu tekstuur

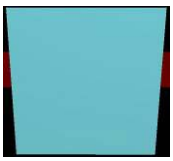

Telliskiviseina tekstuur


Foto

Mitmed 3D programmid kasutavad tekstuuri kohta ka nimetust materjal (*material*).

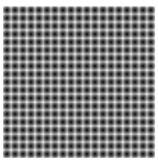
Läbipaistvus

Läbipaistvus (*transparency* vahel ka *translucency*) muudab objekti läbipaistvaks. Tavaliselt saab läbipaistvust seada protsentuaalselt.
Sageli saab kasutada ka *transparency map*'i mille puhul antakse ette pilt, mille abil luuakse ebaühtlane läbipaistvus. Reeglina vastab pildi valgetele osadele täielikult läbipaistmatu osa ja mustale täielikult läbipaistvale.



läbipaistmatu

50% läbipaistev



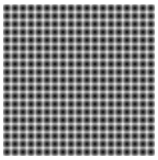
transparency map'iks kasutatud pilt

läbipaistvus *transparency map*'iga

Bump Map

Bump map on meetod, mille abil kantakse objektile tekstuur või reljeef ilma aluseks oleva mudeli geomeetriat muutmata. Tekstuurina kasutatava pildi halltoonides (*grayscale*) väärtustega pikselid tõlgendatakse objekti vastavatele pikselitele rakendatuna pinna kõrgemate ja madalamate osadena. Heledamad pikselid vastavad kõrgematele ja tumedad madalamatele osadele kuid *bump map*'i võib rakendada ka negatiivina (st vastupidi. Seega võib valge värvus tekstuuril tähistada pinna kõrgeimat või vastupidi madalaimat osa.

objekt lihtsalt värvitud



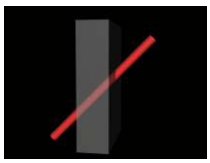
bump map'iks kasutatud pilt

objekt *bump map* materjaliga

Valguse murdumine

Kui on tegemist läbipaistva materjaliga, siis tuleb tähelepanu pöörata ka sellisele nähtusele nagu valguse murdumine (*refraction*). Valgus liigub erinevates keskkondades erineva kiirusega, näiteks teemandis liigub valgus üle kahe korra aeglasemalt kui õhus. Mida suurem on valguse kiiruse erinevus kahes keskkonnas, seda suurem on murdumisefekt kahe keskkonna vahelisel piiril.
Murdumisefekti iseloomustamiseks erinevate ainete puhul kasutatakse murdumisnäitajat ehk valguse murdumise indeksit IOR (*index of refraction*), mis arvutatakse jagades valguse kiiruse vaakumis valguse kiirusega vaadeldavas aines. Tavaliselt esitatakse murdumisnäitaja väärtused kollase valguse jaoks, mis asub nähtava spektri keskel.

Kui murdumisnäitaja on väärtusega 1 (nagu õhul), siis läbipaistva objekti taga olevad objektid paistavad moonutusteta. Väärtuse 1,5 puhul paistab tugev moonutus. Kui murdumisnäitaja on pisut väiksem kui 1, siis objekt peegeldab oma servadel (nagu seebimull). Mida tihedam objekt, seda suurem murdumisnäitaja.



läbipaistval objektil IOR=1

läbipaistval objektil IOR=1,6

Tüüpiliste materjalide/keskkondade valguse murdumisnäitajad (IOR) on järgmised (kasvavas järjekorras):

materjal	IOR
vaakum	1,0
heelium	1,000036
vesinik (gaas)	1,00014
hapnik	1,000276
argoon	1,000281
õhk	1,0002926

materjal	IOR
vedel vesinik	1,0974
süsihappegaas (vedel)	1,2
vedel hapnik	1,221
jää	1,309
vesi (100° C)	1,31819
alkohol	1,329

materjal	IOR
vesi (35° C)	1,33157
vesi (20° C)	1,33335
Atsetoon	1,36
etanool	1,36
Etüül alkohol	1,36
30% suhkrulahus	1,38

materjal	IOR
opaal	1,45
plastik	1,46
sulanud kvarts	1,46
glütseriin	1,473
80% suhkrulahus	1,49
pleksiklaas	1,5
benseen	1,501
kroonklaas (*crown glass*)	1,52
pärlmutter	1,53
keedusool	1,544

materjal	IOR
merevaik	1,546
polüstüreen	1,55
polüstüreen	1,55
smaragd	1,57
akvamariin	1,577
lasuriit	1,61
sinine topaas	1,61
ametüst	1,611
roosa topaas	1,62
topaas	1,62

materjal	IOR
valge topaas	1,63
kvarts	1,644
flintklaas (*flint glass*)	1,65
safiir	1,76
rubiin	1,77
safiir	1,77
teemant	2,419
klaas	1,5 kuni 1,7
joodi kristall	3,34

NB! Läbipaistvate materjalide kasutamisel peab 3D modelleerija kasutama õigeid murdumisnäitajaid!

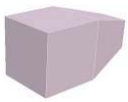
Üsna hea selgituse selle nähtuse kohta leiab aadressil:
http://www.ps.missouri.edu/rickspage/refract/refraction.html

Block modeling

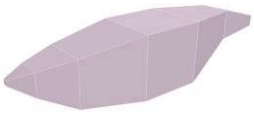
Block modeling on modelleerimistehnika, mille puhul võetakse aluseks mõni primitiiv ning luuakse sellest sobiv 3D objekt selleks tahkusid välja surudes (*extrude*), välja surutud tahke nihutades ning nende suurust muutes (*scale*), tahkude kuju muutes (selleks tahkude servi või punkte selekteerides ja nihutades).

Järgnevalt näide vaala modelleerimise kohta!

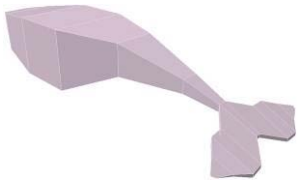
aluseks võetud primtiiv



on alustatud pindade välja surumist



vaala kere hakkab kuju võtma



valminud on saba

on alustatud uimedega

vaala kere kõik osad onn valmis

kandiline kere on silutud (*smooth*)

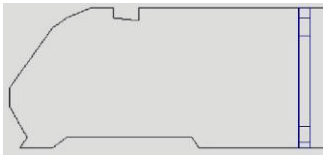
objektile on rakendatud vajalikud pinnatekstuurid

Ristlõike järgi modelleerimine

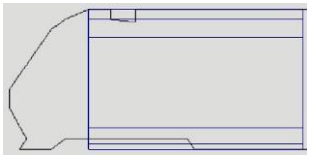
Ristlõike järgi modelleerimise (*Cross section modeling*) on modelleerimistehnika, mille korral antakse ette objekti ristlõige ja seda venitatakse ning moonutatakse vastavalt vajadusele. Seda tehnikat kasutatakse selliste objektide korral, mille kuju pikitelge mööda palju ei varieeru.

Tavaliselt kasutatakse eeskujuna stseeni taustaks paigutatud reaalse objekti fotot.

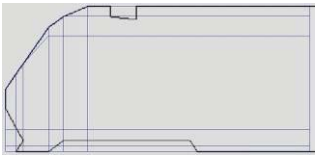
objekti ristlõige



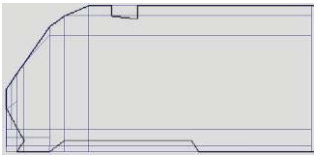
ristlõige paigutatud taustajoonisele

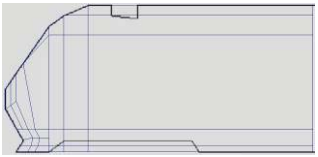


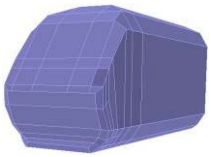
ristlõike venitamise algus



objekti ülevalt poolt valmis


objekti alumisele poolele rohkem detaile


objekt valmis


tulemus

Skeletiline animeerimine

Skeletiline animeerimine (*skeletal animation*) on meetod, mille käigus pannakse mitmetest ühendatud sektsioonidest koosnev mehaaniline objekt liikuma, näiteks inimene vms.

Reaalses maailmas koosnevad paljud mehaanilised objektid konkreetsetest sektsioonidest, mis on omavahel liitekohtadega (*joints*) ühendatud. Näidetena võib tuua inimese ja loomad, auto vedrustuse, robotid jms. Need sektsioonid võivad ümber oma telje pöörata (*pivot*), painduda. Inimese ja loomade puhul on selliseks sektsioonide kogumiks skelett, sektsioonideks on siis luud (*bone*), ühenduskohtadeks (*joints*) on liigesed.

Kinematics

Liikumist kalkuleeritakse kahel moel:

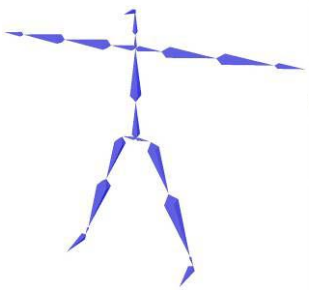
- *Forward kinematics* – ühendatud struktuuri lõpppunkti asukoht kalkuleeritakse struktuuri alguspunkti ning kõigi sektsioonide pöörde- ja käändenurkade järgi ehk ühenduskohtade (*joints*) asukohtade järgi. Näiteks andes ette inimese küünarvarre liikumise (küünarvarre kaldenurgad õlaliigese suhtes), käsivarre liikumise (käsivarre kaldenurgad küünarnuki suhtes), peopesa asendi randmeliigese suhtes jne, saame lõpuks sõrmeotste asukohad. Seega arvutatakse lõpppunkti asukoht alguspunkti suhtes ja see on suhteliselt lihtne tegevus.
- *Inverse kinematics* – ühendatud struktuuri lõpppunkti liigutamise järel arvutatakse välja kõigi ühenduskohtade asukohad, mis on vajalikud soovitud tulemuse saamiseks. Näiteks liigutades peopesa mingile kohale ruumis, seatakse randmeliiges, küünarnukk jms vajalikele kohtadele. Tegemist on keeruka ülesandega, sest tegemist võib olla mitmete võimalike lahendustega.

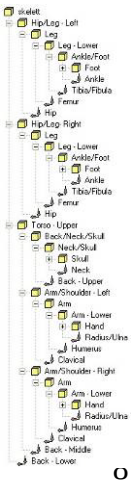
Elusolendite animeerimine

Elusolendite (inimesed, loomad) puhul kasutatakse modelleerimisel ja animeerimisel kolme meetodit: luustiku, kihilise konstruktsiooni ja raamistiku (*framework*).

Luustik

Luustiku (*bones*) meetodi puhul modelleeritakse vaid olendi välimus. Lihaseid ja muid sisemisi struktuure ei modelleerita. Luuakse ühest võrestikust (*mesh*) väline kest (*skin*) ja see ühendatakse luustikuga (*skeleton*).





Luustik luuakse luude (*bones*) hierarhiana, milles on defineeritud luude omavahelised seosed. Lõpuks seotakse omavahel luud ja võrestiku punktid (*points* ehk *vertices*) selliselt, et luu liigutamisel võrestik vastavalt deformeerub. Enamus võrestiku tahkusid (*polygon* ehk *face*) liiguvad just ühe kindla luu suhtes. Ühenduskohtade (liigeste) läheduses liiguvad tahud naaberluude liikumisel

vahepealselt, sellisel moel painutatakse liigese painutamisel võrestikku.

Suurem osa 3D animatsiooniprogramme kasutab seda meetodit.

Kihiline konstruktsioon

Kihiline konstruktsioon (*layered construction*) – mille puhul modelleeritakse ka keha sisemised struktuurid ning tulemus on täpsem. Mudelil on luustik ja väline kest nagu luustiku meetodi puhul kuid lisaks veel vahepealne kiht.

Luustik on hingede või kuulliigenditega (*ball joints*) ühendatud jäikadest kujunditest. Luud ei pea olema ehtsa välimusega, neid võib modelleerida sirglõikudena. Jäik luustik kaetakse deformeeritavate kujunditega lihaste, rasva ja kõõluste modelleerimiseks. Kõige peale genereeritakse väline "nahk" (*skin*).

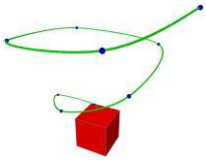
Osakeste animeerimine

Osakeste animeerimine (*particle animation*) on meetod, mille abil luuakse suitsu, tuld, plahvatusi, vedelikke, lumesadu jne. Osakeste süsteem on üksteisest sõltumatute punktide süsteem, mida animeeritakse vastavalt etteantud reeglitele.

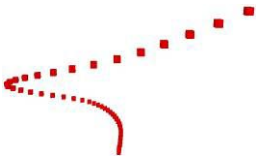
Kõige tavalisemad osakeste atribuudid, mida saab määrata ja animeerida, on: asukoht (*position*), kiirus (*velocity*), energia (*energy*), värvus (*color*) ja selle juurde käiv läbipaistvus (*transparency*) ning eluiga (*lifetime*). Neile atribuutidele saab määrata väärtuseid osakese sünnihetkeks ja nende väärtuste muutumise reegleid osakese eluajaks.

Osakeste kohta määratakse nende elutsükkel (*life cycle*), millistele reeglite põhjal määratakse tekkeaeg ning eluaeg. Määratakse osakeste allikas (*emitter*), koht kus osakesed tekivad. Allikas määrab ka osakeste algparameetrid.

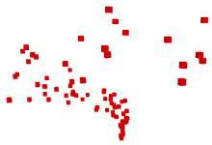
Enamasti lasevad 3D programmid määrata objekti (allikas), mille eeskujul osakesi luuakse. Objekti saab panna etteantud rajale (*path*) liikuma ning siis tekib animatsioon, mille käigus osakesi kiiratakse ajas muutuvast kohast mööda etteantud rada.



Aluseks võetud objekt ja tema liikumisrada



particle efekt ilma osakeste rajalt kõrvalekaldeta



particle efekt osakeste rajalt kõrvalekaldumisega

particle efekt suitsujoa imiteerimiseks

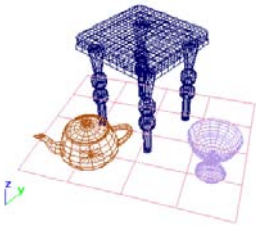
3D *Rendering*

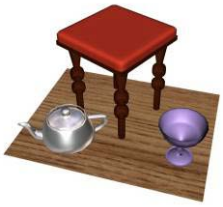
Enamus 3D graafikaprogramme ei suuda reaalajas kaadreid õigete värvide, tekstuuride, valguse ja varjudega valmis joonistada. selle asemel kasutatakse võrgustikku (*mesh*), mis on objektide jämedaks kujutiseks. Kui võrgustikuga rahule jäädakse, siis renderdatakse pilt terviklikult valmis.
Seoses arvutustehnika arenguga muutub renderdamine järjest vähem aeganõudvamaks ning järjest enam programme toetavad reaalajas renderdamist.

Wireframe rendering

Kõige algelisem viis võrgustiku põhjal õige pildi renderdamiseks on nn *wireframe rendering*, mille käigus värvitakse võrgustiku iga sektsioon kaetakse tekstuuri keskmise värvusega.

Tegelikult pole see renderdamise definitsiooni jälgides isegi õige renderdamine. Selline meetod on kasulik eriefektide ja kiire renderdamise (*quick rendering*) jaoks.





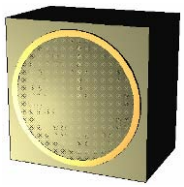
Et tööd kiirendada, kasutavad mitmed programmid protsessi *hidden line removal*, mille käigus jäetakse renderdamata võrgustiku osad, mis valmis objektil välja ei paista.

Flat Shading

Flat Shading on kiireim renderdamismeetod. Arvuti kalkuleerib objekti jaoks keskmise värvuse ning katab siis objekti kõik pinnad (*faces*) lisades värvile musta või valget, et simuleerida varjutust. Objektid näevad välja lamedad, pisut kandilised ja ebarealistlikud, tekstuure ei kasutata.
Suurem osa kaasaegseid programme suudab seda meetodit kasutada reaalajas.

Gourad

Gourad meetod on *Flat Shading*'ust pisut arenenum. Arvutatakse välja värvid objekti kõikide pindade (*face*) servadel (*vertices*) ning seejärel värvitakse kõik pinnad selle värvi järkjärgulise üleminekuga (*gradient*) heledama või tumedama tooni suunas. See meetod pole oluliselt aeglasem kui *Flat Shading* ja ka seda võib rakendada reaalajas.



Phong

Phong meetodi korral uuritakse objekti nähtava osa iga pikselit ning püütakse määrata tema õiget värvi. Võetakse arvesse valgust ja tekstuure. Tulemuse saamine ei võta liiga palju aega ning on küllaltki hea.
Paljud odavamad programmid paremat tulemust anda ei suudagi, jaosvara (*shareware*) programmide seas on see sageli ainus kasutatav meetod.

Ray Tracing

Ray Tracing on kõige parema kvaliteediga ning ka kõige rohkem aega nõudev renderdamise meetod. Selle meetodi korral renderdatakse kogu info; varjud, valgus, peegeldused, läbipaistvused jms.
Ray Tracing meetodi korral arvestatakse ka valguse füüsikaliste omadustega, mida teised meetodid ei tee. Näiteks võetakse arvesse valguse murdumist erinevates keskkondades, pliiats veeklaasis näib murdunud, täpselt nagu reaalses elus.

Radiosity

Radiosity on uus, hiljuti väljatöötatud renderdamismeetod. Selle meetodi puhul vaadatakse üle kõigi stseenis nähtaval olevate objektide omavahelised seosed ja alles pärast seda renderdatakse pilt.
Näiteks muudab valguse peegeldumine erksavärviliselt objektilt ümbritsevad heledad objektid õrnalt sama tooni. Punase palli ümber muutuvad valged seinad roosakaks. Sellist tulemust *Ray Tracing* ei anna, sest selle meetodi puhul ei peegeldata valguskiiri mattidelt objektidelt.