## Drugi path tracing algoritmi in zakaj jih uporablamo(bidirectional pa to)

Osnovni path tracing počasi konvergira pri veliko okluzijah (luč je zakrita).  
  
**Bidirectional Path Tracing** – Pošiljamo žarke iz luči in kamere hkrati in jih povežemo med seboj. Pare poti izberemo z večkratnim pomembnostnim vzorčenjem(Multiple importance sampling).   
  
**Metropolis Light Transport** – Žarke, ki so veliko prispevajo končni sliki ponovno uporabimo z rahlimi variacijami poti. Uporabljamo BPT (Metropolis sampling – imamo vrednost, jo mutiramo, izračunamo vrjetnost sprejetja – če funkcija narašča, in vrednost sprejmemo ali pa ne)  
  
**Photon mapping** – Problemi pri SDS (specular-diffuse-specular) pri normalnih pristopih. Pri photon mappingu delamo ray tracing iz luči, si zapomnemo kje so fotoni prišli do difuzne površine – shranimo v Photon map. Nato delamo PT, ko pridemo do difuzne površine interpoliramo vrednosti fotonov.

## Kaj so L-sistemi

Gramatike, ki jih lahko uporabljamo za modeliranje bioloških sistemov. Lahko so deterministični ali stohastični. Uporabljamo jih za generiranje rastlin, ali mest(ulic, cest, parcel, …).

## Resevanje diferencialnih enacb pri simulaciji delcev (ODE)

**Eksplicitne metode**:

**Eulerjeva metoda** – Začnemo v točki x in se v majhnem časovnem koraku h pomaknemo naprej. V vsakem časovnem intervalu na podlagi trenutne hitrosti izračunamo pozicijo v naslednjem časovnem koraku. (nenatančno)  
  
**Midpoint metoda** – Zančemo v x, se pomaknemo za pol intervala in v tej točki pogledamo hitrost. Na podlagi te hitrosti se premaknemo za celoten časovni interval.  
  
**Runge-kutta** – Izračunamo odvod v nekaj korakih. Q-stage p-order RK izračuna vrednost odvoda q-krat. 1st order 1 stage – eulerjeva metoda, 2nd order 2 stage – midpoint metoda.

**Implicitne metode** – nimajo problema s stabilnostjo. Da izračunamo naslednjo vrednost moramo vedeti odvod v naslednji vrednosti (optimizacijski problem). Počasno a natančno.

## PBR algoritmi

**Metanje žarkov (Ray tracing)** – **Ray casting:** simulira svetlobne žarke. Iz kamere pošiljamo žarke, ki zadanejo predmete. Izberemo najbližje presečišče žarka in predmeta. Nato iz presečišča žarka in predmeta proti luči pošljemo žarek. Če žarek na poti sreča predmet je žarek v senci, drugače ni.  
**Ray tracing** rekurzivno sledi žarkom po prvem odboju, če je element spekluaren (transmisija, odboj). Na koncu izračunamo svetlost. Tu nimamo difuznih odbojev (razen v luč).

**Scattering equation**:  
**BRDF** - Opisuje koliko svetlobe se iz ωi odbije iz površine v smeri ω0 – Kakšno je razmerje med sevalnostjo odbite svetlobe in osvetlitvijo točke. **BTDF** – Opisuje Koliko svetlobe se prenese skozi material glede na vhodno in izhodno smer. **BSDF -**  združena BRDF in BTDF.  
**Enačba razprševanja** (Scattering equation)- opisuje kakšna je sevalnost žarka glede na prispevke svetlobe iz vseh smeri v ciljni točki. Integriramo na podlagi monte carlo metode.

## mikropovrsinski materijali - Oren Nayar pa kaj vpliva na nagnjenost posameznih ploskec

Predpostavljajo, da je površina sestavljena iz ploščic usmerjenih v neko smer. Posamezna ploščica je lahko difuzna ali spekularna.  
**Oren Nayar** – difuzni model, definiran z gausovo porazdelitvijo kotov med ploščico in normalo. Določimo standardno dviacijo kotov z parametrom σ. Parameter σ nam določa kako “hrapava” je površina (manjši kot bolj gladka je površina).

**Zrcalni mikropovršinski modeli** – Imamo normalno porazdelitev half-vektorjev odbojev(Blinn-Phong), v obzir moramo vzeti tudi senčenje in maskiranje(odbiti žarki se zadanejo nazaj v material) kjer računamo samo en odboj(Townbridge-Reitz). Za računanje uporabimo tudi fresnelove enačbe.

## Generiranje višinskih map - Perlinov šum pa kako deluje

Uporabljamo jih za generacije terena, kjer sivina piksla predstavlja višino terena.  
**Diamond-Square**: enostaven, med sosednjimi točkami na sredino dodamo novo točko, ki je povprečje štirih in nek naključen odmik. Na koncu moramo izvesti še glajenje (3x3 glajenje).

**Perlinov šum** – lahko generiramo poljubno dimenzionalne šumne funkcije. Izberemo si število točk in frekvenco šuma. V vsaki točki naključno določimo gradient. Nato interpoliramo končno funkcijo glede na gradiente. Funkcija ima vedno v izbranih točkah vrednost 0 in naključen gradient. Za vsako točko, ki jo vzorčimo izračunamo razdaljo do najbližjih točk mreže in izračunamo gradient v točki s skalarnim produktom. Nato interpoliramo skozi vse točke(bilinearno, kubično).  
Funkcije lahko med sabo tudi kombiniramo, tako, da višamo frekvence vzorčenja, da dobimo oktave. Nato jih združimo skupaj.

## Signed distance fields (kaj je to, kje uporabljamo, metode ki smo jih predstavil)

Implicitna predstavitev za definiranje površin. Je skalarno polje, kjer vsaka vrednost v polju predstavlja razdaljo do površine(0 na površini, 0> v notranjosti, 0< v zunanjosti). Za vsako točko najdemo najbližjo točko iz oblaka točk, Izračunamo skalarni produkt z normalo oblaka točk, da dobimo vrednost. Uporabljamo jo pri marching cubes.

Marching cubes – Algoritem za izdelovanje isopovršin – pretvorba skalarnega polja v poligone. Imamo množico 15 pravil kako generiramo poligone glede na konfiguracijo vokslov. Vsako koniguracijo lahko zakodiramo v 8bitno število. Za vsak kot trikotnika interpoliramo vrednosti glede na sosdenje točke. Normale vokslov izračunamo z razliko vrednosti sosednjih vokslov. Za vsako ogljišče trikotnika lahko izračunamo normale. Algoritem je izboljšan tako, da se doda 14 novih konfiguracij.

Lahko jih sestavimo tudi na drugačne načine: npr. z RBF rekonstrukcijo ali Poissonovo rekonstrukcijo.

## Kinematika (kaj je to, forward/inverse pa metode za inverse)

**Kinematika** – animacija skeleta  
**Forward kinematics** – Animiramo tako, da rotiramo sklepe ali premikamo osnovno vozlišče. Pozicijo posameznega dela skeleta dobimo tako, da začnemo na poziciji osnovnega(starševskega) vozlišča in z translacijami kosti in rotacijami sklepov dobimo pozicijo – Sestavljanje transormacij od starševskega vozlišča do poljubnega sklepa.

**Backward kinematics** – Upoštevamo to, da so kosti fiksne dolžine. Samo s premikanjem zadnjega člena v verigi določimo obliko celotne verige členov. Pri dveh členih trivialno(trikotniško pravilo), pri večih optimizacijski problem:

**Inverz Jakobijeve matrike** – definiramo funkcijo glede na odvode kotov in končne pozicije zadnje točke, rešujemo iterativno v majhnih korakih.  
**Cyclic coordinate descent** – za vsak sklep narišemo premico iz trenutnega sklepa do točke in se zarotiramo, da premico pokrijemo. Ponavljamo, dokler ne pridemo do ciljne točke.

## Volumetricno upodabljanje (pojavi - emisija/absorbcija/sipanje, beerov zakon, kako se rešuje enačba)

Ko imamo neke delce v prostoru(nismo v vakumu)-megla, oblaki, …  
Pojavi: -**Sipanje ven**: Zmanjšanje sevalnosti-svetloba se siplje ven iz medija  
 -**Sipanje notri**: Povečanje sevalnosti – svetloba se siplje v medij  
 -**Absorbcija**: Sevalnost se zmanjša ko gre čez medij  
 -**Emisija**: Sevalnost se poveča ko gre čez medij  
**Fazna funkcija** - verjetnostna porazdelitev za sipanje v neki smeri- lahko se siplje notri ali ven  
Dobimo enačba širjenja svetlobe, ki upošteva vse te količine, vrne sevalnost. Podobno kot BRDF za volumne.  
**Extinction:** Absorbcija+Sipanje ven(koliko svetlobe se izgubi v volumnu) – **Beerov zakon**-sevalnost se manjša eksponentno.

## Imamo tocke v prostoru, kako lahko iz njih rekonstruiramo neko geometrijo

Direktno -Triangulacija: Pretvorimo mrežo točk v poligone:  
 -Osnovna metoda: povečemo točke, ki so vidne med seboj, če ne sekajo obstoječe poti.  
 -Delaunay triangulacija: maksimiziramo najmanjši notranji kot - Za vsak trikotnik očrtamo krog in pogledamo, če je kakšna točka notri. Če je, rob trikotnika zamenjamo. (Optimizacija z velikim trikotnikom kjer inkrementalno dodajamo robove in menjamo robove.)

Implicitno – Signed distance fields, marching cubes.

## Scattering equation

**Enačba razprševanja** (Scattering equation)- opisuje kakšna je sevalnost žarka glede na prispevke svetlobe iz vseh smeri v ciljni točki. Integriramo na podlagi monte carlo metode. Vsebuje BRDF, BSDF, BTDF, BSSRDF…

## Polje sil - Opisi kaj je. Ali lahko vrednost delca vpliva na polje sil

V vsaki točki prostora je kot vektor definirana sila, ki vpliva na delce , ki se nahajajo v tej točki. Vrednost delca ne vpliva na polje sil(razen če imajo delci gravitacijsko polje).

## Transfer funckije pri vokslih

Kako skalarno vrednost voksla pretvorimo v barvo in prosojnost(emisija in absorbcija). Te funkcije določimo sami. Za vsako vrednost skalarja določimo barvo in absorbcijo. Če hočemo poudariti prehode med deli, lahko iz 1 dimenzionalnih prenosnih funkcij upoštevamo tudi gradiente v neki točki. Lahko upoštevamo tudi 2. odvod in tako naprej.

## Kako vzorčimo funkcijo glede na verjetnost

Naredimo seštevanje verjetnosti po korakih, dokler v zadnjem koraku ne seštejemo vseh. Nato izberemo enotno naključno funkcijo, katere vrednost bo določila, katero vrednost bomo izbrali.

## Generiranje mest

**L-sistemi:** Vhod je gostota prebivalstva, višinska slika, prepovedane lokacije, izhod je sistem cest in avtocest, parcele, geometrija stavb.  
Ceste-globalno grejo v smeri gostote prebivalstva, lokalno se izogibajo ovir.  
Parcele-generirajo se med cestami in se rekurzivno razdeli na manjše dele, ki bodo stavbe.

**CGA shape**: Gramatika, kjer iterativno menjamo dele z novimi deli po nekih pravilih. Uporabljamo jih za generiranje stavb.

## problemi pri solidih

**Z uporabo vokslov**: Visoka kvaliteta zahteva veliko vokslov  
**Z uporabo CSG(Constructive Solid Geometry)**: Omejeno število primitivov, težko modeliranje kompleksnih predmetov, veliko dreves je lahko isti solid, male spremembe mnogo vplivajo na rezultat  
**Z uporabo B-rep (Boundary representations)**: načeloma nima težav

## odstranjevanje šuma v point cloudih

Skupaj z oceno normal na točkah lahko tudi odstranimo šum. Normale iščemo tako, da iščemo normalo dolžine 1, ki ima najmanjšo vsoto razdalje ravnine do drugih točk. To lahko posplošimo na problem iskanja lastnih vrednosti kovariančne matrike. Inčemo najmanjšo lastno vrednost kov. matrike. Šum odstranimo tako, da gledamo lastne vrednosti kov. matrike. Če je razmerje vrednosti zadnje l.v. in predzadnje l.v. večje od nekega praga, točko odstranimo.

## kako vzorčimo pri physics based rendering

**Enakomerno vzorčenje**: V vsaki smeri je ista verjetnost za vzorčenje žarka.  
**Kosinusno uteženo vzorčenje**: Več vzorcev damo proti vrhu pol-krogle(ker je osvetljenost povezana z kosinusom vpadnega kota)-dobro za difuzne materiale  
Sami lahko določimo vzorčenje glede na BDRF, ki je najboljši.  
**Vzorčenje luči**: Moramo znati vzorčiti točke na luči(za diskasto luč vzorčimo po kotu in razdalji).  
**Večkratno pomembnostno vzorčenje(Multiple importance sampling)**: Kombiniramo vzorčenje iz luči in iz BDRF. Nekajkrat vzorčimo iz luči, nekajkrat iz BDRF in nato utežimo rezultate.

## Orientacija pri animacijah (Euler angles, Quaternions)

Interpolacije rotacij lahko delamo na 2 načina: Z Eulerjevimi koti, Z kvarternioni(quarternions)  
Če interpoliramo med eulerjevimi koti, lahko pride do kardanske zapore(gimbal lock).  
Problem kardanske zapore lahko rešimo z uporabo kvarternionov pri animaciji rotacije. Pri kravternionih moramo uporabiti sferično linearno interpolacijo, da so kvarternioni enakomerno oddaljeni med seboj.

## BSDF

Je kombinacija BRDF in BTDF, da lahko žarki skozi material gredo, ali pa se od njega odbijejo. Večina materialov je takšnih. Npr. steklo.

## RBF

Radialna bazna funkcija(radial basis function): funkcije čigar vrednosti so odvisne od razdalje točke x do središča c. Uporabljamo jih za boljše konstruiranje SDF(signed distance fields), ker ploskve niso več ravne na vsaki točki, ampak so rezultati mehkejši.

## Luč in zrcalni odboj (ali mormo vzorčit po luči al po telesu)

Pri zrcalnem odboju je bolje vzorčiti po telesu, saj bo do iste točke na luči vodila samo ena smer(vektor).

## Rigid telesa

Toga telesa se ne deformirajo in imajo nek volumen. Pri togih telesih moramo poleg hitrosti gibanja upoštevati tudi to, da se telo lahko vrti(računamo rotacijo in kotno hitrost). Vrtenje lahko računamo glede na masno središče. Položaj lahko izračunamo iz masnega središča. Da lahko opišemo spremembe kotne hitrosti, lahko za izračun sil simuliramo , da je telo narejeno iz delcev in za vsakega izračunamo silo in jih seštejemo med seboj.  
**Vztrajnostni tenzor**-Kako je masa razporejena relativno na masno središče.  
**Navor**-kako so soli razporejene po telesu

## Fresnelova enačba

Fresnelova enačba nam pove količino svetlobe, ki se odbije od površine materiala. Upoštevati moramo 3 materiale: Dielektriki(ne prevajajo el. Energije, del svetlobe se prepusti), prevodniki(odbije večino svetlobe, svetloba se absorbira), polprevodniki.  
**Schlickova aproksimacija** – Največkrat jo uporabljamo namesto fresnelovih enačb, ker nam ni mar za polarizacijo.

## B-rep + Eulerjeve operacije

Eulerjeve operacije – Operacije, ki ohranjajo veljavnost polnih teles. Moramo zagotoviti, da bojo operacije vedno ohranjale formulo za polna telesa. Make n Kill d, 99 operacij.

## Path tracing - Kako se računa pixle(filtri, aliasing ...)

**Vzorčenje piksla:**  
- Regularno: Vsak piksel v isti točki, pripelje do prekrivanja(aliasinga)  
- Naključno: Vmes so luknje, vzorci so neenakomerno porazdeljeni.  
- Jittering: Piksel razdelimo na nekaj podpodročij in v vsakem podpodročju naključno vzorčimo.  
Lahko tudi generiramo vzorce z nizkimi dirkrepancami: Van der Corput(obračanje števil npr 6=110, f(6)=0.0112= 3/8, lahko delamo več-dimenzionalno, tako, da izberemo več baz).  
**Rekonstrukcija slike:**  
- box filter(povprečimo vrednosti znotraj »škatle« piksla)-pride do aliasinga  
- uteženo vzorčenje(piksel je utežena vsota vzorcev iz okolice)-filtrirne funkcije:trikotna, Gaussova, Mitchellova, Lanczos, …

## Voxli

Regularna mreža elementov v treh dimenzijah. Voxel-volume element. Voksli lahko hranijo podatke o objektu. Voksle dobimo iz naprav za zajem prostora; ponavadi vrednost voksla predstavlja gostoto materiala. Voksli lahko predstavljajo 3d predmet(hrani vrednost poln, prazen). Voksle lahko uporabljamo za implicitne modele s konstrukcijo SDF(Signed distance fields). Voksli so točke na regularni mreži.

## Bidirectional path tracing, zakaj se uproablja, kaj je prednost pred enosmernim

Uporablja se kot izboljšanje običajnega path tracinga, kjer žarke hkrati pošiljamo iz kamere in iz luči, ter vmesne poti združimo z uporabo večkratnega pomembnostnega vzorčenja. Prednost pred enosmernim je, da algoritem hitreje konvergira.

## Ruska ruleta

Matematično pravilna izboljšava path tracinga, kjer neko pot zanemarimo, stohastično, glede na svetlost te poti. Ruska ruleta zanemari temne poti z večjo verjetnostjo kot svetle.

## Path tracing pri voxlih

Za direktno vizualizacijo vokslov lahko uporabimo:  
- Splatting(metanje kep): voksli so proecirani na sliko direktno. »splati« imajo vnaprej določeno obliko. Hiter, nenatančen  
- Ray casting(metanje žarkov): skozi material pošljemo žarke, ter vzorčimo žarek skozi volumen. Vzorčenje je lahko enotno ali prilagodljivo. Na vsaki točki vzorčenja interpoliramo vrednost točke, glede na sosednje voksle. Metode za to so najbližji sosed, trilinearna interpolacija(še vedno hitra). Nato vzorce združimo skupaj. Za to obstaja več načinov:  
 - Prvi vzorec-dobimo isopovršino  
 - Povprečje – dobimo x-ray sliko  
 - Maksimalna vrednost – dobimo Maximum intensity projection  
 - Accumulate – Vrednosti vzdolž žarka skupaj kombiniramo in sestavimo sliko, na kateri lahko vidimo več stvari. To delamo tako, da za vsak vzorec predpostavimo, da ima barvo in transparenco in nato združujemo barve skupaj glede na te enote. 2 načina: back-to-front, front-to-back.

## Generiranje pokrain z erozijo

Ko generiramo pokrajno, lahko generiramo tudi vire vode(dejanski vir ali dež). Nato v vsaki točki pogledamo koliko vode in koliko sedimanta imamo. Voda gre vedno v smeri, kjer je nivo vode manjši. Za erozijo pa moramo izračunati še kapaciteto sedimenta, ki ga lahko voda v neki točki nosi. Ta je odvisna od kota. S temi pravili lahko simuliramo erozijo.

## Monte carlo interpolacija

Če vemo, kako izračunati neko funkcijo, se lahko zbrižamo integralu te funkcije tako, da naključno generiramo vrednosti v območju integrala. Ko je teh vzorcev dovolj, lahko iz teh vzorcev tvorimo prostornino pod krivuljo.

## Monte carlo pri PBR, za kaj ga v bistvu uporabljamo

Monte carlo pri PBR uporabljamo za računanje integrala Scattering equationa, ker eksplicitna rešitev za to ni možna.

## Space colonization

Algoritem za generiranje dreves ipd. Definiramo točke atrakcije, kamor se bodo veje gibale in definiramo prvo točko drevesa. Nato točke tekmujejo za prostor, točke so privlačene v smeri najbližje točke atrakcije. Točke atrakcije v radiju »kill distance« so uničene. Na koncu dodamo cilindre, generiramo liste, …

## Splines

Načini modeliranja kompleksnejših krivulj. Krivulje razdelimo na delčke imenovani »splini«, ki so med seboj povezani, oz. imajo različne kontinuitete. Po navadi hočemo, da je kontinuiteta C2, kjer to pomeni, da imata sosednja splina enak prvi in drugi odvod, ter sta povezani. Ta metoda je veliko bolj računsko izvedljiva, kot pa če bi imeli polinom visokega ranga.

## Facial animations – parametrization

Parametrizacija je način animacije obraza, kjer je animacija obraza neodvisna od modela, integracija in animacija je lahka, ter v realnem času. Animacije so tudi neodvisne od topologije modela. Poznamo več sistemov za parametrizacijo: FACS, MPEG4, Minimal perception action.