Algoritmi sjenčanja u 3D prostoru

Novoselac, Paulina

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij računarstva, smjer programsko inženjerstvo

ALGORITMI SJENČANJA U 3D PROSTORU

Završni rad

Paulina Novoselac

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 13.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Paulina Novoselac
Studij, smjer:	Programsko inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4544, 27.07.2020.
OIB Pristupnika:	91466453934
Mentor:	doc. dr. sc. Anita Katić
Sumentor:	izv. prof. dr. sc. Časlav Livada
Sumentor iz tvrtke:	
Naslov završnog rada:	Algoritmi sjenčanja u 3D prostoru
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak završnog rad:	U radu je potrebno opisati problematiku sjenčanja u 3D prostoru te objasniti matematičku pozadinu koja stoji iza toga. U eksperimentalnom dijelu potrebno je na temelju matematičkih izraza napraviti vlastite algoritme sjenčanja za pojedine materijale te ih međusobno usporediti. Tema rezervirana za: Paulina Novoselac Sumentor: Časlav Livada
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	13.09.2023.
Datum potvrde ocjene od strane Odbora:	24.09.2023.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 25.09.2023.

Ime i prezime studenta:	Paulina Novoselac
Studij:	Programsko inženjerstvo
Mat. br. studenta, godina upisa:	R4544, 27.07.2020.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Algoritmi sjenčanja u 3D prostoru**

izrađen pod vodstvom mentora doc. dr. sc. Anita Katić

i sumentora izv. prof. dr. sc. Časlav Livada

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD 1
   1.1. Zadatak završnog rada 1
   1.2. Primjena 1
2. ALGORITMI SJENČANJA U 3D PROSTORU UZ KORIŠTENJE ALATA I TEHNOLOGIJA ZA NJIHOVU IZVEDBU 2
   2.1. OpenGL 3
   2.2. GLSL programski jezik 7
   2.3. C++ programski jezik 8
   2.4. GLEW 8
3. SJENČANJE U 3D PROSTORU 9
   3.1. Koncepti trodimenzionalnog prostora 9
   3.2. Geometrija u tri dimenzije 9
   3.3. Metode prikaza/projekcije u tri dimenzije 10
      3.3.1. Paralelna projekcija 10
      3.3.2. Perspektivna projekcija 10
   3.4. Definicija sjenčanja u kontekstu računalne grafike 11
   3.5. Modeliranje svjetlosti 11
      3.5.1. Izvori svjetlosti 12
      3.5.2. Ambijentalno sjenčanje 13
      3.5.3. Difuzno sjenčanje 13
      3.5.4. Sjajno sjenčanje 15
      3.5.5. Anizotropno sjenčanje 16
   3.6. Materijali objekata i tehnike prikaza u sceni 16
      3.6.1. Napredne tehnike sjenčanja za realistično prikazivanje materijala 18
      3.6.2. Sjene oblika 20
4. ALGORITMI SJENČANJA U 3D PROSTORU 21
   4.1. Uvod u matematički eksperimentalni dio algoritama sjenčanja 21
   4.2. Algoritam kontinuiranog sjenčanja 21
      4.2.1. Matematički eksperimentalni dio algoritma kontinuiranog sjenčanja 22
      4.2.2. Programski eksperimentalni dio algoritma kontinuiranog sjenčanja 23
   4.3. Algoritam Gouraudovog sjenčanja 28
      4.3.1. Matematički eksperimentalni dio algoritma Gouraudovog sjenčanja 29
      4.3.2. Programski eksperimentalni dio algoritma Gouraudovog sjenčanja 31
   4.4. Algoritam Phongovog sjenčanja 35
      4.4.1. Matematički eksperimentalni dio algoritma Phongovog sjenčanja 35
      4.4.2. Programski eksperimentalni dio algoritma Phongovog sjenčanja 36
   4.5. Razlike između flat i smooth sjenčanja 40

5. ZAKLJUČAK 41
SAŽETAK 46
ABSTRACT 47

1. UVOD

Tema završnog rada opisuje i daje uvid u problematiku sjenčanja u 3D prostoru. Obrada završnog rada temeljena je na matematičkoj pozadini sjenčanja, te na konkretnoj realizaciji uz pomoć *Open Graphics Library* odnosno OpenGL višeplatformske specifikacije čiji je cilj omogućavanje pisanja aplikacija za rad s 2D i 3D računalnom grafikom. Programski kod algoritama sjenčanja direktno je ovisan o matematičkoj pozadini, uključujući točke i vektore u prostoru, pravce te poligone na kojima se provodi već spomenuto sjenčanje. Algoritmi sjenčanja koji će se uspoređivati su konstantno sjenčanje, Gouraudovo sjenčanje, te Phongovo sjenčanje.

1.1. Zadatak završnog rada

U radu je potrebno opisati problematiku sjenčanja u 3D prostoru te objasniti matematičku pozadinu koja stoji iza toga. U eksperimentalnom dijelu potrebno je na temelju matematičkih izraza napraviti vlastite algoritme sjenčanja za pojedine materijale te ih međusobno usporediti.

1.2. Primjena

Algoritmi sjenčanja primjenjuju se u računalnoj grafici za prikaz boja i intenziteta površina objekata. Takva obilježja najčešće su primjenjiva u razvoju računalnih igara visokih razina realizma, te prikazu samostalnih objekata u sceni u međuovisnosti s izvorima svjetlosti.

2. ALGORITMI SJENČANJA U 3D PROSTORU UZ KORIŠTENJE ALATA I TEHNOLOGIJA ZA NJIHOVU IZVEDBU

Sjenčanje je glavna etapa u 3D računalnoj grafici koja omogućuje dvodimenzionalnu zaslonsku projekciju objekta kako bi izgledao stvarno ljudskoj vizualnoj percepciji [1]. Za generiranje realne 3D slike u boji na računalnoj grafičkoj opremi moraju se izvršiti četiri osnovna zadatka. Prvo, upotrijebiti matematičke metode za uspostavljanje potrebnog 3D geometrijskog opisa scene i unos istih u računalo. Ovaj dio se može napraviti pomoću sustava za 3D modeliranje. Geometrijski opis scene ima izravan utjecaj na ispravnost renderiranja. Odabrati razumno i učinkovito definiranje podataka iznimno je važno. Drugo, treba pretvoriti 3D geometrijski opis u 2D prikaz perspektive na sceni pomoću perspektivne projekcije. Treće, odrediti sve vizualne površine. To zahtijeva korištenje algoritma za uklanjanje skrivenih površina. Na kraju treba izračunati boju vizualnih površina [2]. Triangulacija pretvara zadani skup točaka u konzistentan poligonalni model (mrežu). Ova operacija dijeli ulazne podatke na jednostavne dijelove i obično generira vrhove, bridove i površine (koje predstavljaju analiziranu površinu) koji se susreću samo na zajedničkim rubovima. Metode konačnih elemenata koriste se za diskretizaciju izmjerene domene dijeljenjem na mnogo malih 'elemenata', obično trokuta ili četverokuta u dvije dimenzije i tetraedra u tri dimenzije. Optimalna triangulacija je definirana mjerenjem kutova, duljina rubova, visine ili površine elemenata, dok je pogreška aproksimacije konačnih elemenata obično povezana s minimalnim kutom elemenata. Postoji mnogo različitih algoritama sjenčanja, a najpoznatiji su kontinuirano sjenčanje i glatko sjenčanje. Ključna razlika između kontinuiranog i glatkog sjenčanja je u načinu na koji se koriste normale. Kontinuirano sjenčanje jednostavno svakom trokutu dodjeljuje vektor normale, a osvjetljenje se vrši pojedinačno na svakoj plohi. Za glatko sjenčanje, normale okolnih površina su prosječne i svakom vrhu je dodijeljena normala [3]. Programi pisani jezikom posebne namjene, poznatim kao programi za sjenčanje, opisuju položaj izvora svjetlosti i karakteristike emisije, svojstva boje i refleksije površina ili svojstva prijenosa atmosferskih medija. Konceptualno, ovi se programi izvršavaju za svaku točku objekata koji se prikazuju kako bi se proizvela konačna boja (i eventualno neprozirnost) prema zadanoj točki gledišta. Općenito, s konstrukcijama poznatim iz programskih jezika opće namjene kao što je C, uključujući petlje, uvjete i funkcije [4]. Gouraudovo sjenčanje radi na način pronalaženje vektora normale koji se odnosi na svaki vrh vidljive površine, izračunavajući boju piksela u vrhu i zatim linearno interpolirajući tu boju po vidljivoj površini. To rezultira prilično glatkom površinom koja koristi samo nešto veću količinu procesorske snage nego model

kontinuiranog sjenčanja. Istraživač po imenu Phong Bui-Tuong proširio je model sjenčanja Henryja Gourauda čineći sljedeći logični korak. Umjesto pronalaženja normala samo na vrhovima, Phong program za sjenčanje izračunava normalu za svaki piksel [5].

2.1. OpenGL

OpenGL (engl. *Open Graphics Library*)je definiran kao sučelje programske podrške za grafički dio sklopovlja. U svojoj svrsi, OpenGL je vrlo brza i lako prenosiva 3D grafička višeplatformska biblioteka/specifikacija. OpenGL nije programski jezik kao što su to C ili C++. Namijenjen je za korištenje uz hardver koji je dizajniran i optimiziran u svrhu prikaza i manipulacije 3D grafikom. Implementacija OpenGL-a je biblioteka koja kreira trodimenzionalnu sliku kao rezultat poziva i izvršavanja OpenGL funkcija ili je to driver za hardverske uređaje (najčešće grafičke kartice) no radi po istom principu. Implementacije na hardveru su puno brže od softverskih, samim time su češće korištene. OpenGL je proceduralni, a ne deskriptivni grafički API (engl. *Application Programming Interface*). Umjesto opisivanja scene i kako bi ista trebala izgledati, programer implementira korake za postizanje određenog izgleda i/ili efekta. Upravo ovi „koraci“ sadrže pozive mnogobrojnim OpenGL funkcijama i naredbama. Iste naredbe koriste se za crtanje grafičkih primitiva kao što su točke, linije i poligoni u već spomenute 3 dimenzije (visina, duljina, dubina). Osim toga, OpenGL podržava osvjetljavanje i osjenčavanje, dodavanje teksture, miješanje boja, prozirnost, ali i još mnoge efekte za postizanje željenog rezultata. Ono što OpenGL ne uključuje funkcije su za prikazivanje prozora i scene u prozoru, interakciju između korisnika ili ulazno/izlaznih datoteka. Upravo zato svako okruženje (Mac OS X ili Microsoft Windows) podržava vlastite funkcije u tu svrhu i zaduženo je za implementaciju nekog od načina prikaza upravljanja crtanja prozora na OpenGL. Softverska implementacija OpenGL-a preuzima grafičke zahtjeve iz aplikacije i konstruira/rasterizira obojenu sliku 3D grafike. Zatim istu sliku šalje za prikaz na monitoru. Hardverska implementacija OpenGL-a uobičajeno se realizira pomoću upravljačkog programa grafičke kartice. Ovdje se OpenGL pozivi prenose hardverskom upravljačkom programu, dok isti taj upravljački program izravno komunicira s hardverom grafičkog prikaza, a ne šalje svoj izlaz Windows GDI-u. GDI „graphics device interface“ je Windows API koji se poziva kada je izlaz potrebno prikazati na zaslonu, no sadržava i metode za pisanje teksta u prozoru, crtanje jednostavnih 2D linija i slično. Kompletan proces iscrtavanja, te detaljna objašnjenja za svaki od koraka prikazani su u nastavku.

1. SPECIFICIRANJE VRHOVA

Početak prikaza i obrade slike koristeći OpenGL počinje specifikacijom i određivanjem vrhova. Podaci vrhova poznati kao atributi vrhova zapisuju se kao objekt niza vrhova (engl. *Vertex Array Object* , VAO), spremaju se u jedan ili više objekata spremnika vrhova (engl. *Vertex Array Buffer*, VBO). Kako bi se razlikovala uloga VAO i VBO, VAO samo definira svojstva vrha, odnosno vrhova, dok VBO pohranjuje sve podatke navedene u VAO. Podaci o vrhovima koji se bilježe u VAO najčešće su: lokacija vrha u prostoru, koordinate teksture, te normala vrha. Bitno je zabilježiti indekse modela u spremnik elemenata jer se isti indeksi kasnije koriste za oblikovanje trokuta tokom procesa spajanja vrhova. U procesu spajanja vrhova iscrtavaju se primitivni oblici, no to su osim trokuta i točke, te linije, te se iscrtavaju u ovisnosti o ciljanom krajnjem objektu.

2. PROGRAM ZA SJENČANJE VRHOVA

Program za sjenčanje vrhova (engl. *vertex shader*) programski je dio koda pisan u OpenGL jeziku sjenčanja (engl. *OpenGL Shading Language*, GLSL) koji se prevodi na grafičku procesorsku jedinicu (engl. *Graphics Processing Unit*, GPU) prije izvršavanja. Svaki program za sjenčanje vrhova prima ulazne podatke za svaki vrh objekta, što bi značilo ako je u pitanju trokut, svaki program za sjenčanje vrhova primit će tri vrha kao ulazne podatke. Ulazni podaci o vrhovima su koordinate vrhova, boja vrhova, također normala vrhova neovisno o kakvom obliku se radi. Program za sjenčanje vrhova koristan je za izračun osvjetljenja, koji u konačnici služi za realistično osvjetljenje ciljanog objekta u 3D prostoru. Budući da program za sjenčanje vrhova direktno prima ulazne podatke iz podataka o vrhovima potrebno je specificirati ulazne varijable s metapodacima o lokaciji, kako bi se moglo upravljati atributima vrhova na centralnoj procesorskoj jedinici (engl. *Central Processing Unit*, CPU). Upravo to je obilježje programa za sjenčanje vrhova jer zahtijeva dodatnu specifikaciju za svoje ulazne podatke kako bi ga se kasnije moglo povezati s podacima o vrhovima. Program za sjenčanje vrhova najčešće ima i izlazni podatak, primjerice boju kojom će ciljani objekt biti obojan cijelom svojom površinom. Takav izlazni podatak potrebno je prenijeti idućem dijelu programskog koda koji se zove program za sjenčanje fragmenata o kojemu će biti riječi u nastavku. U obradi vrhova (engl. *Vertex Processing*) čiji je glavni predstavnik upravo program za sjenčanje vrhova koji je obavezan korak, mogu, ali ne moraju sudjelovati koraci popločavanja (engl. *Tessellation*) i program za sjenčanje geometrije (engl. *Geometry shader*). Cilj popločavanja je razlaganje primitivnog oblika na manje dijelove od kojih može biti sastavljen, dok je cilj programa za sjenčanje geometrije obrada ulaznog primitivnog oblika, što kao izlaznu vrijednost može imati nula ili više primitivnih ulaznih oblika.

3. NAKNADNA OBRADA VRHOVA

Naknadna obrada vrhova (engl. *Vertex Post-Processing*) slijedi neposredno nakon već navedene obrade vrhova pomoću programa za sjenčanje vrhova (i opcionalno popločavanja, te programa za sjenčanje geometrije). Početni korak u naknadnoj obradi vrhova je transformacija povratnih informacija (engl. *Transform Feedback*). Karakteristika ovog koraka je zabilješka o primitivnim oblicima generiranim u obradi vrhova, te pohrana istih u objekte spremnika (engl. *Buffer Objects*). Na taj način čuva se stanje objekta nakon transformiranja prikaza, također moguće je i opetovano slanje istih više puta ukoliko je potrebno. Idući korak u naknadnoj obradi vrhova je isijecanje (engl. *Clipping*). Ukoliko se primitivni oblik nalazi na granici između unutrašnje i vanjske strane volumena pogleda (engl. *Viewing Volume*), tada se provodi odvajanje na više primitivnih oblika kako bi cijeli primitivni oblik ostao u volumenu pogleda. Moguće je uvesti i korisnički definirane operacije isijecanja po vrhu, no to se može uvrstiti u zadnju etapu obrade vrhova. Bitna napomena je kako se pozicije vrhova transformiraju iz prostora isijecanja u prostor radnog prozora pomoću transformacije prikaza i perspektivne podijele.

4. SKLOP PRIMITIVA

Prije procesa rasterizacije, a nakon dosadašnjih obrada vrhova, korak sklopa primitiva (engl. *Primitive Assembly*), odnosno primitivnih oblika, zaslužan je za modeliranje vrhovima s ciljem stvaranja primitiva koji će se renderirati. Vrhovi se povezuju u ovisnosti o korisničkoj definiciji modeliranja istih, drugim riječima, vrhovi se povezuju u točke, linije ili najčešće trokute. Ukoliko korisnički odabir zahtijeva trokute kao rezultat, set od tri vrha tvorit će jedan trokut. Također, ako se u listi vrhova za generiranje trokuta nalazi više vrhova, svaka trojka tvorit će novi trokut. Krajnji cilj koraka sklopa primitiva je pretvaranje toka vrhova u sekvencu osnovnih primitiva, te je nužna prije prelaska u korak rasterizacije.

5. RASTERIZACIJA

Rasterizacija (engl. *Rasterization*) je proces pretvaranja prethodno dobivenih primitiva iz koraka sklop primitiva u piksele koji će se prikazivati na zaslonu. Bitno je odrediti i orijentaciju poligona (radi jednostavnosti bit će govora o trokutu), drugim riječima, odrediti je li trokut okrenut prema prednjem ili stražnjem dijelu. Takav proces određivanja naziva se odbacivanje lica (engl. *Face culling*), pomoću kojega se mogu odbaciti svi trokuti čija orijentacija ne zadovoljava uvjet. Nadalje, svaki trokut koji je pravilno orijentiran dalje se obrađuje. Obrada se nastavlja određivanjem piksela koji su njegov sastavni dio. Piksel koji se smatra kao sastavni dio pravilno orijentiranog trokuta je onaj čije je središte unutar granica trokuta. Rasterizacija uključuje i

interpolaciju. Sama interpolacija bazira se na računanju vrijednosti između već poznatih vrijednosti s obzirom na matematički model. Cilj interpolacije u sklopu rasterizacije je izračun atributa za svaki pojedini piksel, na temelju vrijednosti atributa vrhova i udaljenosti istog piksela od svakog položaja vrha. Za svaki piksel, vrijednost se svakog atributa interpolira uz uvjet da postoji točka s koordinatama središta piksela, baricentičke koordinate koje određuju utjecaj svakog vrha na istu točku unutar trokuta, te početno poznate atribute vrhova. Svaki piksel generiran rasterizacijom prelazi u sljedeći korak, odnosno prolazi kroz program za sjenčanje fragmenata.

6. PROGRAM ZA SJENČANJE FRAGMENATA

Program za sjenčanje fragmenata (engl. *fragment shader*) dio je programskog koda koji je zaslužan za obradu fragmenata generiranih rasterizacijom, i koji kao krajnji rezultat daje set boja i vrijednost dubine. Za svaki piksel koji obuhvaća primitivni oblik, stvara se takozvani fragment. Svaki fragment sadržava interpolirane vrijednosti za svaki vektor, a njihov izvor je upravo izlazna vrijednost koja se dobiva izvođenjem programa za sjenčanje vrhova, te se prenosi programu za sjenčanje fragmenata. Izlazna vrijednost nakon provođenja programa za sjenčanje fragmenata predstavlja boju kojom je obojan svaki fragment predanog mu primitivnog oblika. Nakon provođenja naredbi programa za sjenčanje fragmenata, izlazna vrijednost, odnosno boja, može biti obrađena u nastavku samog procesa renderiranja u OpenGL-u.

7. OPERACIJE PO UZORKU

Operacije po uzorku (engl. *Per-Sample Operations*) podrazumijevaju testiranja hoće li fragmenti biti obrađeni i ažurirani ili će biti odbačeni jer je testiranje rezultiralo negativnim izlazom. Testove u nastavku najčešće omogućava korisnik prema svojim potrebama. Test vlasništva piksela (engl. *Pixel ownership test*) rezultira negativnim ishodom ukoliko piksel sadrži nedefinirane vrijednosti, također ako je aktivan drugi prozor koji nadilazi GL prozor. Test škara (engl. *Scissor Test*) rezultira negativnim ishodom ukoliko piksel određenog fragmenta izlazi izvan granica ciljanog pravokutnika zaslona. Test dubine (engl. *Depth Test*) rezultira negativnim ishodom ukoliko su pojedini fragmenti renderirani pogrešnim redoslijedom, drugim riječima, u međuspremnik dubine (engl. *Depth Buffer*) spremljene su udaljenosti od točke gledišta k fragmentu kako bi se znalo koji je najbliži fragment. Test ima negativni rezultat ukoliko su pojedini bliži fragmenti renderirani kao udaljeni, što na konačnom prikazu ne odgovara vrijednostima fragmenata i udaljenostima definiranima u međuspremniku dubina. Višeuzorkovanje (engl. *Multisampling*) je proces u kojemu je svaki piksel na rubu poligona uzorkovan više puta. Takvim procesom određuje se koji piksel pripada unutrašnjosti trokuta, te koji je na rubu. Što je više

podudaranja, veća je sigurnost da se piksel nalazi unutar trokuta. Za piksele koji su uzorkovanjem definirani kao rubni, zaglađivanje nazubljenosti rubova je proces koji upravo takve piksele čini prozirnijima kako bi se zagladile granice trokuta. Uzorkovanje je inače dio procesa rasterizacije, no zaglađivanje nazubljenosti rubova je svakako dio operacija po uzorku. Miješanje (engl. *Blending*) je proces miješanja boja, gdje je čest slučaj promjena prozirnosti (primjena nove alpha vrijednosti zaslužne za razinu prozirnosti). Na kraju svih navedenih testiranja, ako su ista omogućena korisničkim nastojanjem, podaci o fragmentima zapisuju se u spremnik okvira (engl. *Framebuffer*) i iscrtavaju se.

2.2. GLSL programski jezik

OpenGL jezik za sjenčanje ili GLSL (engl. *OpenGL Shading Language*) programski je jezik visoke razine. Programiranje GPU-a u takvom jeziku, umjesto u asemblerskom kodu znači kompaktniji i čitljiviji kod, uz to povećava i produktivnost. Izgledom i pravilima sličan je programskom jeziku C, no GLSL podržava i druge ugrađene tipove podataka i funkcija koje su neizmjerno bitne za pisanje i interakciju programa za sjenčanje vrhova i programa za sjenčanje fragmenata. Specifična obilježja GLSL-a su podržavanja novih tipova podataka, što uključuje vektore s jednom, dvije, tri ili četiri komponente za svaku od tipova podataka kao što su bool, int, uint, float i double. Također, podržava i float, te double tipove unutar matrica koje su također definirane. Tipovi podataka (uključujući vektore i matrice) mogu imati i pohrambene modifikatore koji utječu na njihovo ponašanje. Od značaja su „in“, „out“, „uniform“, „buffer“ modifikatori. „In“ modifikator koristi se za ulazne podatke kao što su atributi vektora (za programe za sjenčanje vrhova). „Out“ modifikator definira izlazne podatke primjerice transformirane koordinate iz programa za sjenčanje vrhova. „Uniform“ modifikator zaslužan je za određivanje vrijednosti varijable od strane aplikacije, ali prije izvršenja koda programa za sjenčanje. Naknadne promjene nisu moguće jer se uniformne varijable prenose u svim etapama koda programa za sjenčanje, te se samim time moraju deklarirati kao globalne varijable. Bitno je naglasiti, da program za sjenčanje ne može pisati u uniformnu varijablu niti mijenjati njezinu vrijednost i sadržaj. „Buffer“ modifikator vrlo je sličan uniformnom, no glavna razlika je što program za sjenčanje može mijenjati vrijednosti i sadržaj tako označene varijable. Najčešće se koristi kako bi se označio dio memorije kao memorijski buffer/spremnik koji se prenosi kroz programe za sjenčanje u samom programskom kodu. Sam GLSL podržava i uključuje matematičke operacije, postavljanje teksture, upravljanje osvjetljenjem i sjenama, post—procesiranje, interpolaciju i to su jedne od mnogih dodatnih funkcija. Podržava više različitih vrsta programa za sjenčanje, o kojima je bilo riječi.

2.3. C++ programski jezik

Programski jezik C++ višenamjenski je programski jezik. Činjenica da se uglavnom prevodi u izvorni strojni kod čini ga neizostavnim dijelom za sustave koji očekuju visoku izvedbu, kao što su to operacije i računanja u 3D grafici. Budući da je OpenGL biblioteka temeljena na programskom jeziku C, C++ izvrstan je izbor jer je uz obilježja C poprimio i obilježja objektno orijentiranih jezika, što omogućuje bolju manipulaciju i obradu podataka. Direktna poveznica između OpenGL, GLSL i C++ jezika je ta što koristeći C++, programer piše kod koji se izvršava na CPU-u i uključuje OpenGL pozive. Samim time zadatak koji obavlja međudjelovanje C++ i OpenGL-a je instalacija GLSL napisanog koda na GPU.

2.4. GLEW

GLEW (engl. *OpenGL Extension Wrangler Library*) je biblioteka otvorenog koda (engl. *Open-source*) čija svrha je učitavanje C/C++ proširenja. GLEW omogućava efikasne mehanizme određivanja koja OpenGL ekstenzija (proširenje) je podržana na ciljanoj platformi za vrijeme izvođenja.

3. SJENČANJE U 3D PROSTORU

3.1. Koncepti trodimenzionalnog prostora

Kako bi se postigla realistična slika, scena ili model, isti moraju biti prikazani u 3D grafici, jer se na taj način postiže dojam realizma, u odnosu na 2D, kao što su primjerice grafovi, mape i slično. Kako bi se proizveo realistični prikaz oblika ili modela, isti mora biti prikazan u sceni koordinatnog sustava pomoću transformacijskih i projekcijskih procesa koje preoblikuju trodimenzionalni oblik u prikaz koordinatnog sustava na dvodimenzionalnom uređaju. Također je potrebno odrediti vidljivi dio scene kako bi objekt bio selektivno prikazan u ovisnosti o položaju unutar scene. Takav prikaz moguće je ostvariti iscrtavanjem. Dakle, iscrtavanje kao proces uzima geometrijski opis trodimenzionalnog promatranog objekta i modificira ga u sliku objekta prikazanu na zaslonu. Postizanje vjerodostojnog izgleda trodimenzionalnog objekta vrši se uz pomoć raznih efekata. Jedan od efekata svakako je perspektiva u kojoj skrivamo dijelove objekta koje ljudsko oko ne bi vidjelo iz perspektive gledišta u kojoj se trenutno nalazi, primjerice, ako je promatrani objekt kocka, dakle trodimenzionalni objekt promatranja, vidljive će biti samo one plohe na kojima je usmjereno gledište, dok će se ostale plohe izostaviti, kao i u stvarnom svijetu. Važnu ulogu predstavlja i obojenje objekta, naime, ako se za cijeli objekt iskoristi identična boja i nijansa boje, plohe objekta i njegovi oštri bridovi gube značaj, jer je očekivano kako samo određene stranice trebaju biti jednako obojene, dok druge trebaju biti nijansirane u odnosu na položaj svjetlosti, perspektivu gledišta i ostale faktore.

3.2. Geometrija u tri dimenzije

Budući da je potrebno prikazati 3D objekt u 2D sceni, Kartezijev koordinatni sustav, uz $x - os$ i $y - os$, koje redom predstavljaju širinu i dužinu objekta, uvodi se dodatna $z - os$ za prikaz dubine. Z os određuje položaj objekta u odnosu na udaljenost od gledišta, odnosno, ako se objekt nalazi u pozitivnom dijelu $z - osi$, tada smatramo da je isti bliži gledištu. Vrijedi i obrnuto, ako je objekt u negativnom dijelu $z - osi$, tada smatramo da je on dalji gledištu. Svakako je nužna prilagodba širine i duljine, odnosno ovisnost o x i y osi iz razloga što su bliži objekti širi i dulji, dok su oni udaljeni, uži i kraći.

3.3. Metode prikaza/projekcije u tri dimenzije

Nakon modeliranja, odnosno eventualnog rotiranja, skaliranja i translatiranja, koje se izvršavaju na objektu, cilj projekcije je prikaz istih rezultata u 2D sceni. Projekcijske transformacije zaslužne su za prikaz krajnjeg izgleda scene nakon već spomenutog, modeliranja. Najvažnije i najpotrebnije projekcije su ortografska i perspektivna projekcija.

3.3.1. Paralelna projekcija

Paralelna projekcija zaslužna je za prikaz objekata, odnosno poligona na zaslonu na način da izuzima pojam udaljenosti objekta. Linije i poligoni prikazani su na 2D zaslonu koristeći samo paralelne linije, što bi značilo, iako je objekt udaljen od točke gledišta, bit će prikazan identičnom veličinom kao da je u neposrednoj blizini gledišta. Ovakva projekcija najčešće se koristi za iscrtavanje dvodimenzionalnih slika kao što su nacrti ili dvodimenzionalne grafike, kao primjerice teksta. Paralelna projekcija često se koristi i u 3D renderiranju kada je dubina renderiranja vrlo mala u odnosu na udaljenost točke gledišta. U nastavku je prikazan efekt paralelne projekcije ukoliko se ista koristi za prikaz 3D objekta. Uočljivo je kako na modelu, konkretno, kuće, njezin udaljeniji dio nije prikazan u pozadini, u odnosu na prednji dio kuće.

Slika 3.1*: Prikaz paralelne projekcije modela kuće* [6]

3.3.2. Perspektivna projekcija

Perspektivna projekcija prikazuje scenu onakvu kakvom bi ju doživjeli u stvarnom životu. Prepoznatljiva je po tome što se udaljeni objekti prikazuju manjima od onih u neposrednoj blizini gledišta. Dobar primjer ovakve perspektive su tračnice upravo iz razloga što su paralelne u stvarnom životu, no uslijed uvođenja perspektivne projekcije gledišta, iste se sužavaju sve više u daljinu. OpenGL omogućava nam upravo primjenu ovakve projekcije. Koristi se za iscrtavanje

scene koje sadržavaju objekte koji zahtijevaju sužavanje s obzirom na udaljenost. U nastavku je prikazan model kuće u perspektivnoj projekciji. Za razliku od paralelne projekcije, perspektivna projekcija udaljeniji dio kuće prikazuje u području gdje bi isti bio očekivan u stvarnom svijetu. Odnosno, udaljeniji dio kuće prikazan je smanjen u odnosu na bliži dio kuće.

Slika 3.2*: Prikaz perspektivne projekcije modela kuće* [6]

3.4. Definicija sjenčanja u kontekstu računalne grafike

Sjenčanje, u kontekstu računalne grafike, je kao proces dio renderiranja u kojemu dolazi do obojenja objekata prikazanih u 3D sceni. Sjenčanje se bavi proizvođenjem ili simulacijom boje objekata, jednog ili više, promatranih s jedne točke gledišta. Dakako, točka gledišta igra važnu ulogu u projekciji objekata jer sama pojavnost objekta ovisi o kutu i udaljenosti od točke gledišta, pa samim time ovisnost se prenosi i na boju, te osvjetljenje istog objekta. Kako je to slučaj i u stvarnosti, izvan konteksta računalne grafike, svjetlost i osvjetljenje objekata vrlo je važno za doživljaj boje i obilježja promatranog oblika. Sjenčanje se oslanja na zakone prirode, kako bi simulacija u računalnoj grafici nalikovala stvarnom svijetu.

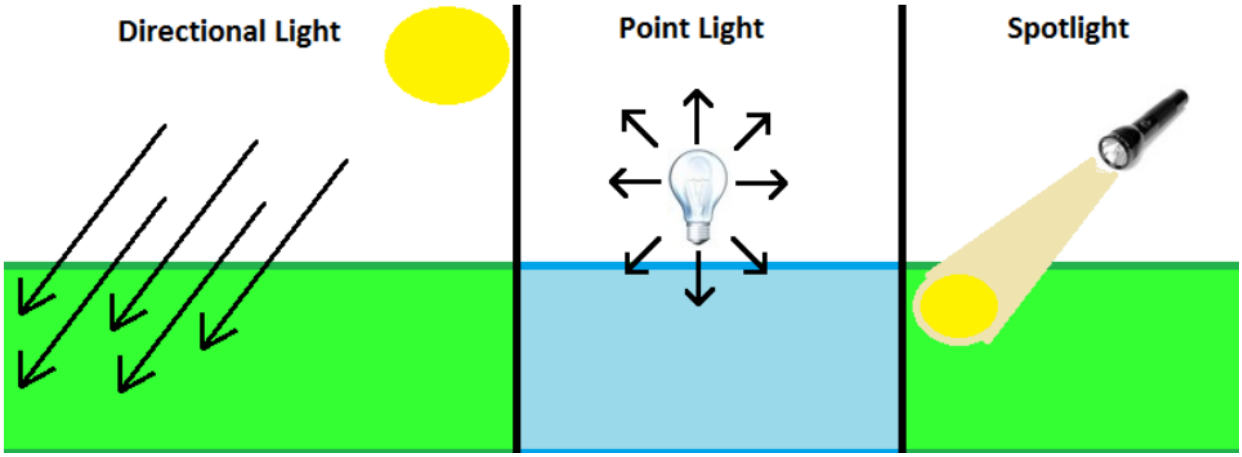
3.5. Modeliranje svjetlosti

Budući da je osvjetljenje vrlo bitna tehnika u računalnoj grafici, postupak modeliranja svjetlosti sastoji se od niza etapa. Počevši od odabira izvora svjetlosti, materijala od kojeg je osvjetljeni predmet načinjen, normale površine objekta u koju je svjetlost usmjerena, te samog konačnog izgleda trodimenzionalnog objekta u sceni. Bez osvjetljenja, objekti izgledaju kao da su načinjeni od plastike, upravo iz tog razloga, za dobivanje realističnijeg prikaza moraju se definirati materijalna svojstva, svojstva osvjetljenja, pa tako i sveobuhvatna svjetlost sa svojim parametrima

koji se odnose na ambijentalno ili dvostrano osvjetljenje. Osvjetljenje je usko vezano uz sjenčanje u OpenGL-u jer se ono realizira ili po-vrhu (engl. *Per-Vertex*) ili po-fragmentu (engl. *Per-Fragment*), a moguća sjenčanja su difuzno, sjajno i ambijentalno sjenčanje koja će biti opisana u nastavku. OpenGL također podržava različite tipove izvora svjetlosti: lokalni (točkasti) izvor svjetlosti, spotlight izvor svjetlosti, te udaljeni izvor svjetlosti o čemu će također biti više riječi u nastavku.

3.5.1. Izvori svjetlosti

Ambijentalno osvjetljenje (engl. *Ambient Lights*) prvi je oblik izvora svjetlosti koje je poznato po osvjetljavanju cijelog prostora, primjerice cijele prostorije. Ovakvo osvjetljenje raspršuje se ravnomjerno u prostoru, samim time se koristi kao element općeg osvjetljenja scene. Jedan od konkretnih izvora svjetlosti je točkasti izvor svjetlosti (engl. *Point Light Source*). Karakteristika je točkastog izvora svjetlosti ravnomjerno emitiranje svjetlosti iz jedne točke u svim smjerovima, primjerice kao što je to slučaj emitiranja svjetlosti sijalice. Usmjereni izvor svjetlosti (engl. *Spotlight*), iako može biti konstruiran kao točkasti izvor svjetlosti sužavanjem kuta emitiranja, karakterističan je upravo po usmjerenju i kutu pod kojim se svjetlost raspršuje. Konkretan primjer spotlight izvora svjetlosti je reflektor čija je zadaća osvjetljenje konkretnog objekta u prostoru prema kojemu je otvoren kut i usmjerenje svjetlosti. Udaljeni izvor svjetlosti (engl. *Distant Light Source*) je svaki izvor svjetlosti čija udaljenost je značajna u odnosu na površinu i/ili objekt koji osvjetljava. Smjer osvjetljenja uniforman za cijelu površinu. Konkretan primjer takvog osvjetljenja je Sunce. U nastavku su prikazana tri najčešća izvora svjetlosti korištena u OpenGL implementacijama, pri čemu je direktni izvor svjetlosti (engl. *Directional Light*) sinonim za prethodno objašnjen udaljeni izvor svjetlosti, konkretno, Sunce.



Slika 3.3: *Izvori svjetlosti (s lijeva na desno: direktan/udaljen izvor svjetlosti, točkasti izvor svjetlosti, usmjereni izvor svjetlosti)* [7]

OpenGL podržava ovakva četiri oblika izvora svjetlosti i barem osam izvora svjetlosti po programu. Svaki izvor svjetlosti mora biti omogućen i specificiran pojedinačno. Funkcije koje to ostvaruju su *glLightfv(source, parameter, pointer_to_array),* te *glLightf(source, parameter, value).*

3.5.2. Ambijentalno sjenčanje

Ambijentalno sjenčanje jedno je od jednostavnijih oblika sjenčanja jer ne uzima u obzir poziciju ili rotaciju izvora svjetlosti u sceni. Pruža opće osvjetljenje trodimenzionalnih objekata u sceni, bez obzira kakvi su i ima li ostalih izvora svjetlosti usmjerenih ka istim objektima. Računanje ambijentalnog sjenčanja temelji se na koeficijentu ambijentalnog sjenčanja, koje predstavlja boju, te intenzitet ambijentalnog sjenčanja, koji je konstantan i jednak za svaku točku površine. Množenje koeficijenta i intenziteta za rezultat ima ambijentalno sjenčanje scene. Ovakvo sjenčanje nije naklonjeno realističnom prikazu jer će modeli biti jednako osvijetljeni na svim svojim plohama, bez obzira na orijentaciju u prostoru.

U nastavku je prikazan konkretan primjer ukoliko je objekt osjenčan samo ambijentalnim osvjetljenjem. Uočljivo je kako objekt nema nikakve bridove osim obruba, dok je cijela unutrašnjost osjenčana istim intenzitetom, dakle i istom bojom bez ikakvih prijelaza.

Slika 3.4: *Ambijentalno sjenčanje* [8]

3.5.3. Difuzno sjenčanje

Difuzno sjenčanje kao glavnu uočljivu karakteristiku ima statično osvjetljenje objekta. Drugim riječima, ako postoji izvor svjetlosti, usmjeren prema specifičnoj točki na površini objekta, svjetlost će se reflektirati ravnomjerno u svim smjerovima. Difuzno sjenčanje ovisi o usmjerenosti

izvora svjetlosti i karakteristikama materijala od kojih je model načinjen. Važno je za napomenuti, kako se difuzno sjenčanje objekta ne mijenja u odnosu na promjeru smjera gledišta, tj. bit će osvjetljena samo ona točka objekta u koju je usmjeren izvor svjetlosti. S obzirom na navedene karakteristike, računanje difuznog sjenčanja umnožak je koeficijenta difuzne refleksije, intenziteta difuzne svjetlosti uz skalarni produkt normale promatrane površine i vektora osvjetljenja iz izvora svjetlosti. Skalarni produkt normale promatrane površine i vektora izvora svjetlosti u ekstremima ima određenu maksimalnu pozitivnu vrijednost osvjetljenja promatrane točke u slučaju kada se izvor svjetlosti nalazi točno na normali površine. U drugom ekstremnom slučaju, ako se izvor svjetlosti nalazi u ravnini promatrane površine, okomito na normalu, osvjetljenja nema i konačan rezultat je 0. Slučaj u kojemu je osvjetljenje pozitivno, no nije ekstrem, uvjetovano je kutom pod kojim se nalazi izvor svjetlosti. Uvjet je takav da se izvor svjetlosti nalazi pod proizvoljnim kutom između normale površine i same površine, izostavivši kut normale od 90° u odnosu na pripadnu joj površinu s 0°. Ovakav skalarni produkt, točno je zapisati u obliku kosinusa kuta normale i vektora izvora svjetlosti. Lambertov kosinusni zakon govori kako je jakost osvjetljenja površine proporcionalna kosinusu kuta što ga upadne zrake izvora svjetlosti zatvaraju s normalom površine. Realističnost prikaza oblika postignuta je u osnovnoj mjeri zbog prethodno navedenog obilježja ovisnosti izričito o kutu upadnih zraka izvora svjetlosti i normale površine, dok se u obzir ne uzima položaj gledišta, koji bi bio ključan za postizanje većeg nivoa realizma.

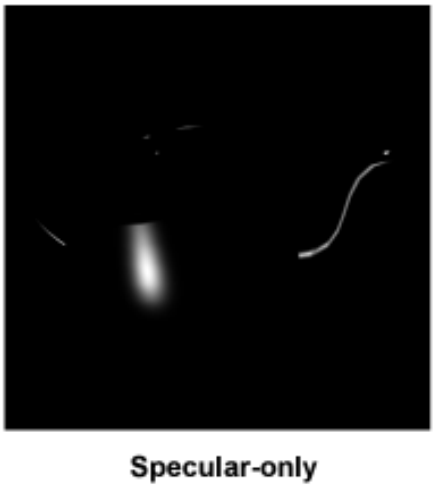
U nastavku je prikazano tijelo osjenčano isključivo difuznim sjenčanjem. Za razliku od ambijentalnog sjenčanja, primjenom difuznog sjenčanja vidljiva je površina objekta, zakrivljenost pojedinih ploha koje ga sačinjavaju, također i oštrijih bridova koji upotpunjuju sveukupni dojam realističnog prikaza objekta. Difuzno sjenčanje prikazuje materijal objekta kao matiran, bez eventualnog odsjaja, neovisno o materijalu.

Slika 3.5: *Difuzno sjenčanje* [8]

3.5.4. Sjajno sjenčanje

Sjajno (engl. *Specular*) sjenčanje opravdava svoj naziv po posljedicama koje ima na površinu modela na koju je primijenjeno. U stvarnosti, kada u sjajni materijal usmjerimo, primjerice, točkasti izvor svjetlosti, takav materijal oponašat će refleksiju navedenog izvora svjetlosti. Računski izvod sjajnog sjenčanja umnožak je koeficijenta sjajne refleksije, intenziteta svjetlosti sjajne refleksije, te skalarnog produkta vektora gledišta i vektora idealne refleksije potenciranog na konstantu odsjaja. Vektor idealne refleksije proučava se u odnosu na kut upadnih zraka izvora svjetlosti do normale površine, te reflektirane zrake koja se odbija od površinu. Vektor idealne refleksije je onaj vektor čiji je kut između upadne zrake svjetlosti u odnosu na normalu, identičan kutu reflektirane zrake u odnosu na istu normalu. Intenzitet i položaj sjajnog osvjetljenja promijenit će se ukoliko se položaj gledišta nalazi pod kutom odstupanja od reflektirane zrake. Materijal od kojeg je načinjen model također ima značaj u kolikoj mjeri će se reflektirati svjetlost. Glavna razlika u odnosu na difuzno sjenčanje je uzimanje u obzir položaja gledišta zbog kojeg se u konačnici mijenja intenzitet, položaj i oblik osvjetljenja modela kojeg promatramo, dok se kod difuznog sjenčanja osvjetljava samo ona točka koja je prvotno definirana i odabrana, bez obzira na položaj gledišta.

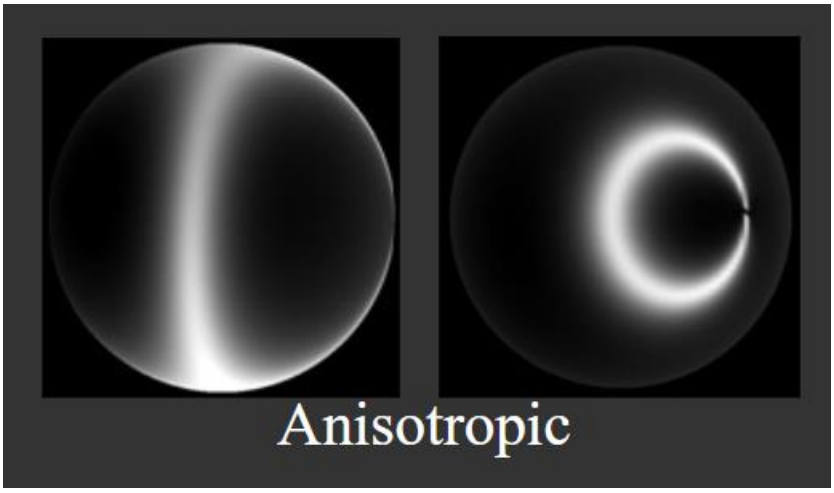
U nastavku je priložena slika koja demonstrira sjajno sjenčanje objekta. Primjetno pokazuje sjaj materijala, no uz sjaj, ostale plohe i prvotna boja objekta nisu vidljive. Za konačni prikaz oblika potrebno je uračunati i prethodno opisane oblike sjenčanja, odnosno ambijentalno i difuzno sjenčanje.



Slika 3.6: *Sjajno sjenčanje* [8]

3.5.5. Anizotropno sjenčanje

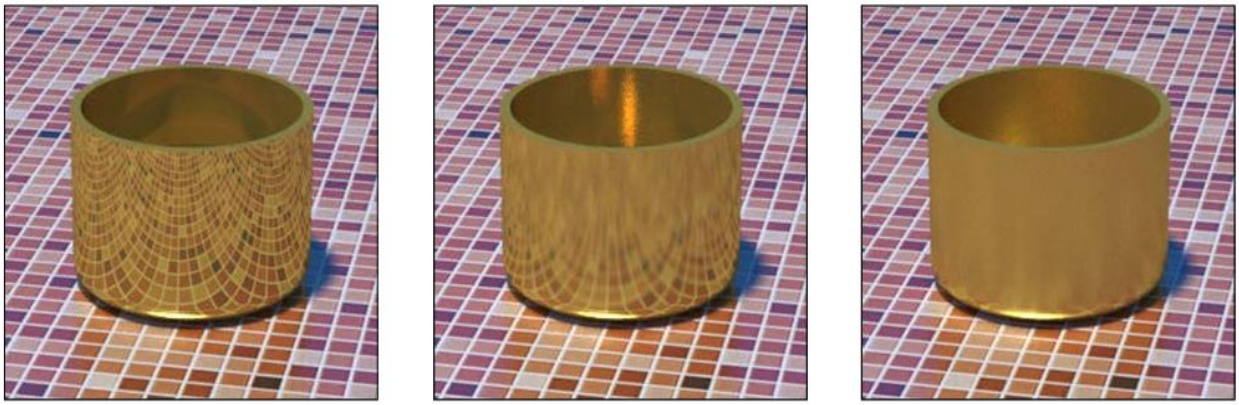
Anizotropno sjenčanje primjenjuje se za simulaciju materijala koji posjeduju anizotropna svojstva. Anizotropna svojstva materijala uključuju refleksiju svjetlosti na različite načine u različitim smjerovima. Stvarni primjer takvih materijala su kosa, brušeni metali i slični materijali koji dijele ovakva svojstva refleksije svjetlosti. Budući da anizotropni materijali pokazuju različita svojstva refleksije u različitim smjerovima, u izračun anizotropnog sjenčanja moraju se uvesti smjerovi svojstva materijala.



Slika 3.7: *Anizotropno sjenčanje* [9]

3.6. Materijali objekata i tehnike prikaza u sceni

Karakteristike materijala od kojega je objekt načinjen, proučavaju se s obzirom na površinsku reakciju prilikom djelovanja različitih oblika svjetlosti. S obzirom na fizička svojstva materijala, određene površine objekata mogu reflektirati, prenositi ili apsorbirati različite valne duljine svjetlosti. Sjajni materijal pokazuje koncentriranu, drugim riječima gotovo zrcalnu sjajnu (spekularnu) refleksiju. Materijali grube prirode pokazuju svojstva raširene sjajne (spekularne) refleksije, upravo zato ne posjeduju svojstvo odsjaja. Sjajna refleksija kod metala prikazuje svojstvo boje metala, umjesto boje izvora svjetlosti kako je to inače slučaj. Materijali poput brušenih metala, vinila, imaju sitne nabore, te nemaju ista svojstva sjajne refleksije, odnosno reflektiraju svjetlost na različite načine u različitim smjerovima. Prikaz materijala u računalnoj grafici teži aproksimaciji stvarnih materijala. U nastavku slijede tri primjera istog modela, no različitih materijala. Uočljive razlike vide se u području reflektiranja svjetlosti krenuvši od savršeno glatke površine, ka površini grubljeg materijala.

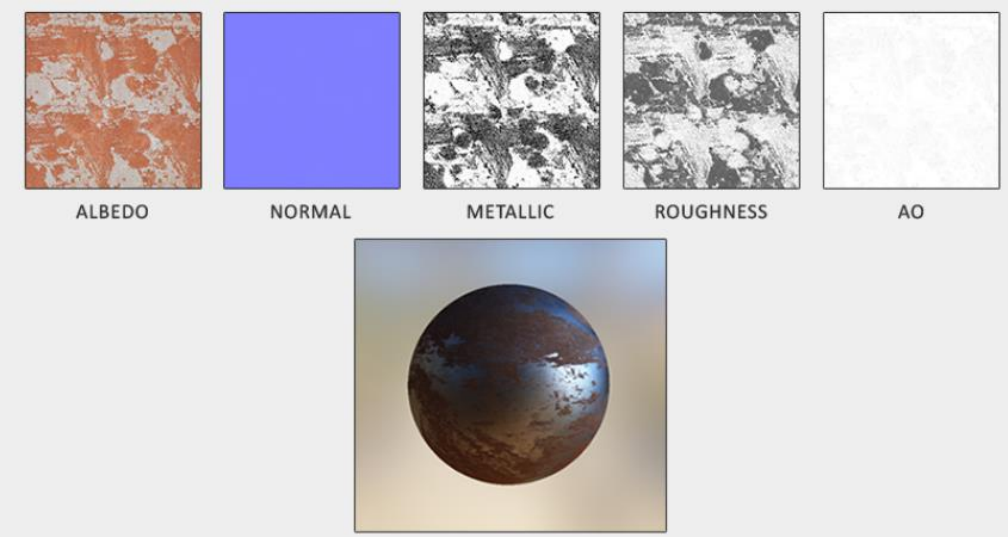


Slika 3.8. *Materijali modela(s lijeva na desno: idealna refleksija, refleksija srednje grubog materijala, refleksija grubog materijala)* [10]

Refleksija svjetlosti, kako je upravo objašnjeno, definira doživljaj promatranog materijala i jasne razdiobe među njima. Najčešće korišteni model aproksimacije svojstava materijala u 3D računalnoj grafici naziva se Phongov reflektivni model. Phongov model konstruiran je upotrebom tri različite refleksijske komponente. Uključuje komponentu ambijentalne refleksije, komponentu difuzne refleksije koja uključuje osvjetljenje bez obzira na položaj gledišta, te komponentu sjajne refleksije zaslužnu za prikaz odsjaja materijala ovisno o kutu gledišta i odstupanja od maksimalnog intenziteta refleksije. Konačan rezultat zbroj je ovih komponenti koje se mogu modificirati u ovisnosti o svojstvima fizičkih materijala koje je potrebno prikazati. Poboljšani model refleksije nastao na temelju Phongovog modela zove se Blinn-Phong model. Blinn-Phong model nastao je s obzirom na nedostatke uočene u Phongovom reflektivnom modelu, odnosno u rezultatu sjajnog sjenčanja kao njegove komponente. Ukoliko kut između gledišta i refleksije, po računu sjajnog sjenčanja, prelazi 90°, kosinus funkcija će rezultirati negativnim ishodom, što u konačnici znači da je sjajno sjenčanje nepostojeće. Ovakav slučaj riješen je uspostavom Blinn-Phong modela u kojemu se umjesto vektora refleksije u dijelu sjajnog sjenčanja, koristi vektor polu-udaljenosti (halfway vector). Vektor polu-udaljenosti nalazi se točno između vektora gledišta i smjera osvjetljenja. Korištenjem takvog vektora izbjegnut je problem kuta većeg od 90° jer na ovaj način, kut između vektora gledišta i normale promatrane površine nikad neće prijeći 90° bez obzira na položaj gledišta (uz uvjet da se izvor svjetlosti nalazi iznad površine). Blin-Phong model refleksije kao glavnu razliku između Phong modela refleksije podrazumijeva izračun kuta između vektora polu-udaljenosti i normale površine, dok je to kod Phongovog modela bio izračun kuta između vektora gledišta i refleksije.

3.6.1. Napredne tehnike sjenčanja za realistično prikazivanje materijala

Budući da je cilj iscrtavanja (engl. *rendering*) u računalnoj grafici postizanje izgleda scene i oblika što sličnije onima u stvarnom svijetu, postoje bolje tehnike od Phongovog i Blinn-Phongovog modela refleksije svjetlosti scene i oblika. Jedna od najpoznatijih tehnika, odnosno skupa tehnika, je fizički bazirano iscrtavanje (engl. *Physically Based Rendering*, PBR) kratica će se koristiti u nastavku. PBR orijentirano je aproksimaciji fizičkih svojstava materijala i svjetla u sceni kako bi sveukupni doživljaj dostigao što veću razinu realizma. Jedna od najznačajnijih prednosti ovakvog pristupa je izgled materijala. Materijali se prikazuju točno bez obzira na osvjetljenje u kojemu se nalaze, što nije slučaj pri korištenju tehnika poput Phong i Blinn-Phong. U nastavku se nalazi ilustracija koja u prvom redu predstavlja materijalna svojstva i teksture koje se mogu koristiti za PBR prikaz željenog modela. Pojedine su teksture opcionalne u ovisnosti o željenom rezultatu, no vidljivo je kako prikazana sfera bitno odudara realističnom komponentom, uzevši za usporedbu Phong i Blinn-Phong tehniku.



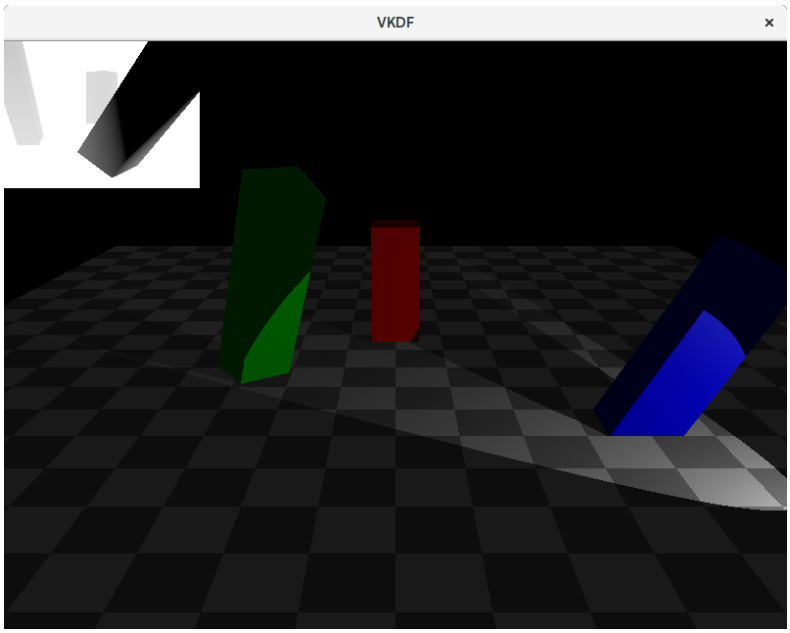
Slika 3.8. *PBR skup tehnika prikazan na primjeru sfere* [11]

Za PBR modele osvjetljenja, koji su smatrani fizički baziranim sjenčanjem, moraju nužno vrijediti sljedeća tri uvjeta. Kako bi se model smatrao fizički baziranim, mora se temeljiti na mikrofacetnom modelu površine (engl. *Microfacet Surface Model*), mora težiti očuvanju energije te koristiti fizički baziranu dvosmjerno reflektivnu distribucijsku funkciju (engl. *Bidirectional Reflective Distribution Function,* BRDF), u nastavku će se koristiti kratica. Mikrofacetni model temelji se na teoriji postojanja mikrofaceta, odnosno savršeno reflektirajućih ogledalaca, na svakoj površini mikroskopski gledajući. Efekt koji se postiže korištenjem mikrofaceta, pogotovo ako se u obzir uzima sjajno sjenčanje, odnosno sjajna refleksija, je sljedeći. Hrapavije površine raspršit

će upadne zrake izvora svjetlosti u vrlo različitim smjerovima što rezultira raširenijom sjajnom refleksijom, kako bi to izgledalo i u stvarnom svijetu. Glatke površine reagiraju suprotno. Glatke će površine reflektirati upadne zrake izvora svjetlosti u približno istom smjeru, što rezultira manjom i izraženijom sjajnom refleksijom. Uveden je parametar grubosti materijala. Što je parametar veći, sjajna refleksija je raširenija i slabija, te vrijedi i obratno. Zakon očuvanja energije, jedan je od najbitnijih prirodnih zakona. Budući da je PBR orijentiran ka prirodnim zakonima, ovaj zakon očuvanja energije mora vrijediti i za osvjetljenje, odnosno za refleksiju i refrakciju svjetlosti. Zakon očuvanja energije, nadovezuje se na prethodno objašnjenu teoriju mikrofaceta. Uzmimo za primjer hrapave materijale s najraširenijom sjajnom refleksijom. Prirodni zakon nalaže kako će intenzitet svjetlosti opasti. Upravo to je zakon očuvanja energije, u protivnom bi najhrapaviji materijali prikazivali sjajnu refleksiju u najintenzivnijem obliku, odnosno emitirali bi više energije, kršeći zakon očuvanja energije. Zrake se svjetlosti nakon što dođu u susret s površinom, djelomično emitiraju, što upravo predstavlja dio sjajnog sjenčanja, dok se djelomično apsorbiraju, što predstavlja dio difuznog sjenčanja. Ovakav pristup funkcionira prigodno za materijale poput voska, kože i sl., dok za metalne materijale vrijedi sljedeće. Iako metalne površine prate principe refleksije i refrakcije, lomljena se svjetlost apsorbira bez raspršenja što dovodi do prikaza samo reflektirane svjetlosti. Iz istog razloga, metali i površine načinjene od istih, ne prikazuju difuzno obojenje. Budući da se svjetlost koja je reflektirana ne može više apsorbirati, one se međusobno isključuju. Jednadžba refleksije osigurava da će očuvanje energije biti postojano, odnosno da difuzno i sjajna refleksija nikada neće premašiti vrijednost upadnog svjetla izvora svjetlosti na promatranu površinu. BDRF je funkcija dvosmjerne reflektivne distribucije koja za ulaz uzima niz parametara. Parametri su ulazni smjer svjetlosti, izlazni smjer (u smjeru gledišta), normalu površine, te parametar hrapavosti površine, opisan u dijelu mikrofaceta. Upotreba BDRF-a u prvom je redu aproksimacija količine kojoj pojedinačne svjetlosne zrake doprinose krajnjoj reflektiranoj svjetlosti neprozirne površine materijala s obzirom na njegova svojstva. BDRF za savršeno glatke površine, primjerice ogledalo, za rezultat daje 1 samo za ulaznu zraku čiji je upadni kut jednak kutu reflektirane izlazne zrake. BDRF se pridržava zakona očuvanja energije i teorije mikrofaceta, prema tome se i smatra tehnikom realističnog prikaza materijala, za razliku od Blinn-Phong modela koji djeluje po sličnom principu, ali se ne pridržava zakona očuvanja energije. Najpoznatija BDRF izvedba naziva se Cook-Torrance BRDF koja omogućuje realistični prikaz kako metalnih materijala, tako i materijala s hrapavom površinom

3.6.2. Sjene oblika

Sjene su jedan od temelja prikaza realističnih oblika jer daju vidljiv prikaz trodimenzionalnosti oblika i prostora u kojemu se isti nalaze. Također, zaslužene su za bolju orijentaciju u prostoru, prikaz odnosa između oblika i površine, te oblika međusobno. Daju bolji uvid u dubinu scene i objekata u njoj. Budući da je prikaz sjena objekata relativno zahtijevan za prikazati u 3D računalnoj grafici, postoji nekoliko jednostavnih tehnika prikaza. Tehnika mapiranja sjena (engl. *Shadow mapping*) postupak je u kojoj se koristi tekstura, mapa sjena (engl. *Shadow Map*) kao mehanizam pohrane informacija o dubini vidljive scene iz perspektive izvora svjetlosti. Nakon početnog renderiranja scene iz perspektive izvora svjetlosti, stvara se referenca za provjeru sjena, odnosno mapa sjena. Ponovnim iscrtavanjem scene uspoređuju se pikseli od kojih je scena tvorena, rezultat ponovnog renderiranja pokazuje je li promatrani piksel u sjeni ili izvan nje, odnosno je li osvjetljen ili nije. Piksel se smatra lociranim u sjeni ako je dubina piksela iz scene veća od one u mapi sjena. Navedeni postupak koristan je kada su sjene generirane samo iz smjera izvora svjetlosti. Kada je izvor svjetlosti točkasti, primjerice žarulja ili svijeća, koristi se tehnika točkastih sjena (engl. *Point shadows*). U nastavku je prikazana vizualizacija tehnike mapiranja sjena.



Slika 3.10. *Vizualizacija tehnike mapiranja sjena* [12]

4. ALGORITMI SJENČANJA U 3D PROSTORU

4.1. Uvod u matematički eksperimentalni dio algoritama sjenčanja

Koordinatni sustav uređeni je par čvrste točke O u prostoru i baze vektorskog prostora. Ako govorimo o desnom Kartezijevom koordinatnom sustavu uređeni par je $(O;\ i, j, k)$. Budući da će se za svaki trokut, odnosno svaki poligon, trebati izračunati normala vektora, baza vektorskog prostora upravo je ortonormirana baza. Norma vektora može se izračunati jedino u ortonormiranoj bazi, što će biti neophodno za izračun već spomenutih normala. Trokut je jednoznačno određen trima svojim vrhovima. Nazovimo ih A, B i C. Svaki od vrhova nalazi se u prostoru koordinatnog sustava s x, y i z koordinatama. Kako bi tri vrha tvorila trokut, vektori koji ih povezuju ne smiju biti kolinearni, odnosno ne smiju imati isti smjer. Također, vrhovi moraju biti međusobno različiti, te konačno svaki od vrhova trokuta mora biti povezan s ostalima kako bi se tvorio trokut. Polazišna točka u sjenčanju svakog poligona, odnosno objekata načinjenih od poligona je određivanje normale za pojedini poligon, na individualne načine u ovisnosti o algoritmu sjenčanja koji se koristi. Općenito, normala površine objekta koji se promatra okomiti je vektor na istu površinu. Konkretno ju računamo na temelju geometrije oblika koji će posljedično biti osjenčan. Radi jednostavnosti, praktični primjeri pokazuju se na najjednostavnijem poligonu, odnosno trokutu. Također, normalu je potrebno normirati, drugim riječima, skalirati na vektor duljine 1. Svrha normiranja je zadržavanje istog smjera, no promjena duljine na jediničnu jer će u koracima implementacije algoritama sjenčanja biti bitno omogućiti pravilan izračun osvjetljenja i drugih operacija koje ovise o smjeru normale. Normiranje se najčešće provodi računanjem Euklidske (L2) norme koja dijeli svaki od skalara koji množi komponente početnog vektora normale.

4.2. Algoritam kontinuiranog sjenčanja

Algoritam kontinuiranog sjenčanja, najjednostavnija je metoda sjenčanja, u kojoj je za svaki poligon, od kojih je objekt izgrađen, odabran jedan ključan vrh ili najbolje središte istog. Nakon odabira takvog vrha ili točke središta, računa se intenzitet svjetlosti nakon odabranog modela osvjetljenja, te se cijeli poligon, posljedično, ispunjava kopijom iste te vrijednosti. Algoritam kontinuiranog sjenčanja, po svojoj svrsi, dobro radi s oblicima kao što su piramide, no ako je u pitanju objekt s više stranica i/ili objekt s zaobljenom površinom, ovakav pristup je neefikasan jer je prikaz sam po sebi nerealističan. Reprezentacija modela sa zaobljenom površinom, primjerice stošca može se prikazati kao višestranična piramida. Povećanjem broja stranica, odnosno povećanjem broja inicijalnih trokuta ili poligona, svaki od njih bit će obojan

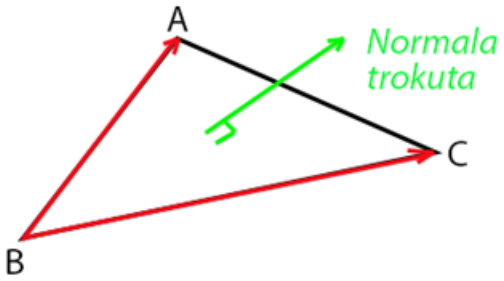
svojom bojom, kako je prethodno objašnjeno. Veći broj stranica, posljedično rezultira većim brojem prikazanih nijansi boje, samim time aproksimacija stošca u obliku višestranične piramide izgleda realističnije. Budući da je riječ o pokušaju realističnog prikaza, povećanjem broja bojanih trokuta povećana je realističnost, no aproksimacija je i dalje očigledan problem kad su u pitanju modeli zaobljena oblika. Problem je, također, u zauzimanju memorije prilikom dodavanja stranica i eventualnoj promjeni položaja kamere/točke gledišta jer je približavanjem i direktnim pogledom na mrežu stranica, uočljiva aproksimacija zaobljene površine koristeći mnoštvo stranica.

4.2.1. Matematički eksperimentalni dio algoritma kontinuiranog sjenčanja

Matematički eksperimentalni dio algoritma kontinuiranog, odnosno flat sjenčanja započinje određivanjem koordinata vrhova trokuta. Napomena, radi jednostavnosti objašnjenja koristi se najjednostavniji poligon, odnosno trokut. Vrhove trokuta nazovimo A, B, C. Svaki vrh posjeduje svoje koordinate u prostoru.

$$A = (x_a, y_a, z_a,), B = (x_b, y_b, z_b), C = (x_c, y_c, z_c) \tag{4-1}$$

Zelenom bojom prikazana je normala trokuta. Vektori $\overrightarrow{BA}$ i $\overrightarrow{BC}$ označeni su crvenom bojom, te se koriste za izračun pripadne normale.



Slika 4.1. *Normala trokuta*

Normala trokuta računa se iz potrebe kasnijeg izračuna intenziteta osvjetljenja trokuta. Vektor

$$\overrightarrow{BA} = (\mathrm{x_a} - \mathrm{x_b})\,\vec{\imath} + (\mathrm{y_a} - \mathrm{y_b})\,\vec{\jmath} + (\mathrm{z_a} - \mathrm{z_b})\,\vec{k} \tag{4-2}$$

Vektor

$$\overrightarrow{BC} = (\mathrm{x_c} - \mathrm{x_b})\,\vec{\imath} + (\mathrm{y_c} - \mathrm{y_b})\,\vec{\jmath} + (\mathrm{z_c} - \mathrm{z_b})\,\vec{k} \tag{4-3}$$

Za izračun normale koristi se vektorski produkt vektora $\overrightarrow{BA}$ i $\overrightarrow{BC}$. Vektorski produkt vektora $\overrightarrow{BA}$ i $\overrightarrow{BC}$ računa se:

$$\vec{N} = \overrightarrow{BA} \times \overrightarrow{BC} = \begin{vmatrix} \vec{\imath} & \vec{\jmath} & \vec{k} \\ x_a - x_b & y_a - y_b & z_a - z_b \\ x_c - x_b & y_c - y_b & z_c - z_b \end{vmatrix} =$$

$$= \begin{vmatrix} y_a - y_b & z_a - z_b \\ y_c - y_b & z_c - z_b) \end{vmatrix} \vec{\imath} - \begin{vmatrix} x_a - x_b & z_a - z_b \\ x_c - x_b & z_c - z_b \end{vmatrix} \vec{\jmath} + \begin{vmatrix} x_a - x_b & y_a - y_b \\ x_c - x_b & y_c - y_b \end{vmatrix} \vec{k} \quad (4\text{-}4)$$

Nadalje, nužno je definirati izvor svjetlosti za dani trokut. Intenzitet osvjetljenja označimo s I. Faktor intenziteta osvjetljenja, označimo ga s I_{faktor}, računa se skalarnim produktom prethodno izračunate normale trokuta $\vec{N}$ i definiranog smjera izvora svjetlosti. Smjer izvora svjetlosti označimo s $\vec{L}$.

$$I_{faktor} = \vec{N} \cdot \vec{L} = |\vec{N}| \cdot |\vec{L}| cos\varphi \quad (4\text{-}5)$$

pri čemu je

$$|\vec{N}| = \sqrt{{n_x}^2 + {n_y}^2 + {n_z}^2} \quad (4\text{-}6)$$

$$|\vec{L}| = \sqrt{{l_x}^2 + {l_y}^2 + {l_z}^2} \quad (4\text{-}7)$$

$$cos\varphi = \frac{\vec{N} \cdot \vec{L}}{|\vec{N}| \cdot |\vec{L}|} = \frac{n_x l_x + n_y l_y + n_z l_z}{\sqrt{{n_x}^2 + {n_y}^2 + {n_z}^2} \cdot \sqrt{{l_x}^2 + {l_y}^2 + {l_z}^2}} \quad (4\text{-}8)$$

Dobiveni I_{faktor} koristi se pri umnošku sa prethodno zadanim intenzitetom osvjetljenja trokuta, označimo ga s I. Intenzitet osvjetljenja I zapravo je boja koja se primjenjuje na trokut, te umjesto koordinata u prostoru, komponente ovog vektora su r, g i b, oznake za definiranje boje r(Red), g(Green), b(blue). Zadnja formulacija zaslužna za konačno sjenčanje trokuta u željenoj, izračunatoj boji je:

$$I_{konačno} = I \cdot I_{faktor} \quad (4\text{-}9)$$

4.2.2. Programski eksperimentalni dio algoritma kontinuiranog sjenčanja

Program za sjenčanje vrhova kontinuiranog sjenčanja

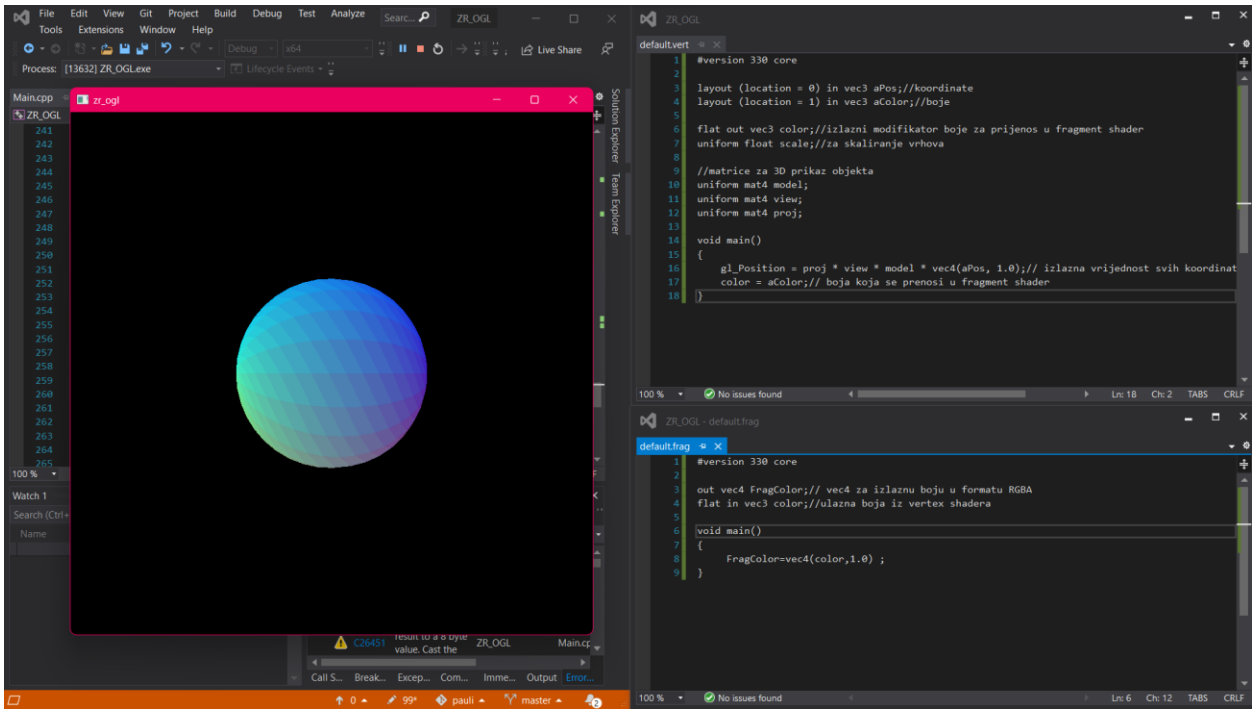
```
#version 330 core

layout (location = 0) in vec3 aPos;//koordinate
layout (location = 1) in vec3 aColor;//boje
```

```
out vec3 color;//izlazni modifikator boje za prijenos u fragment shader
uniform float scale;//za skaliranje vrhova

//matrice za 3D prikaz objekta
uniform mat4 model;
uniform mat4 view;
uniform mat4 proj;

void main()
{
	gl_Position = proj * view * model * vec4(aPos, 1.0);// izlazna
vrijednost svih koordinata
	color = aColor;// boja koja se prenosi u fragment shader
}
```

Program za sjenčanje fragmenata kontinuiranog sjenčanja

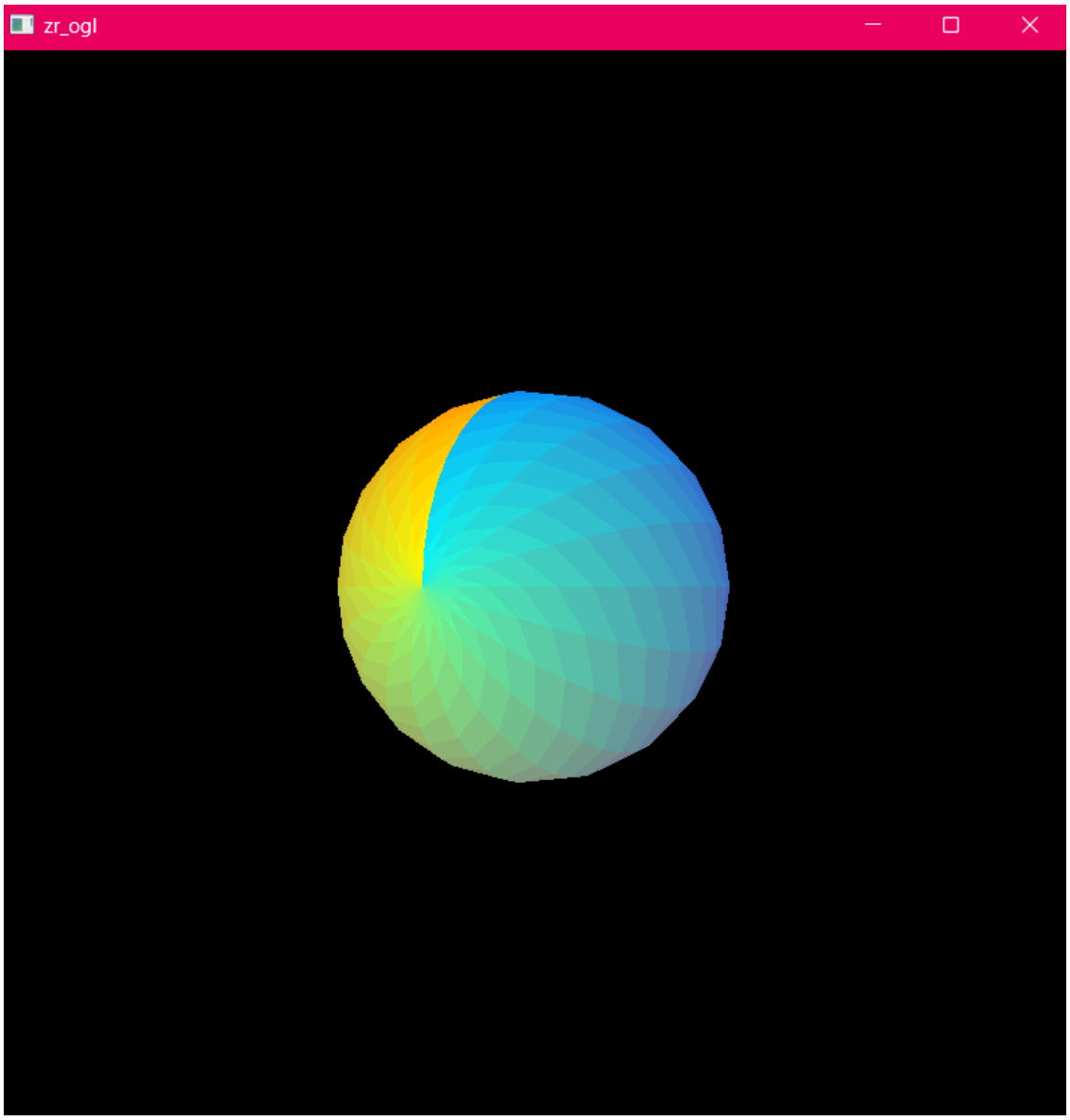
```
#version 330 core

out vec4 FragColor;// vec4 za izlaznu boju u formatu RGBA
in vec3 color;//ulazna boja iz vertex shadera

void main()
{
	 FragColor=vec4(color,1.0) ;
}
```

Slika zaslona u nastavku prikazuje implementaciju prethodna dva dijela programa za kontinuirano sjenčanje. Program za sjenčanje vrhova kontinuiranog sjenčanja i program za sjenčanje fragmenata kontinuiranog sjenčanja u glavnom dijelu programa „Main.cpp“ pozvani su kao parametri konstruktoru kompletnog programa za sjenčanje, odnosno glavnog programa za sjenčanje „Shader“.



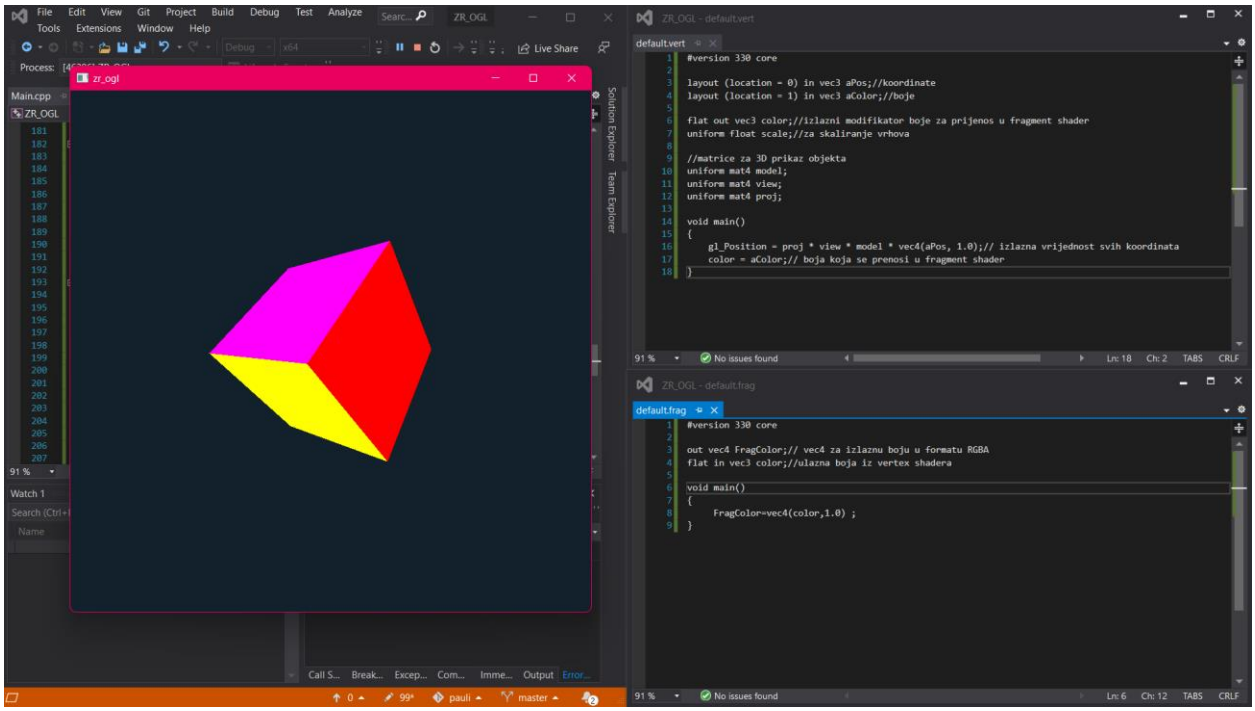
Slika 4.2: *Prikaz kontinuiranog sjenčanja, uz pripadne programe za sjenčanje vrhova i fragmenata, te model sfere osjenčan kontinuiranim sjenčanjem*

Program za sjenčanje vrhova kontinuiranog sjenčanja koristi atribut lokacije kako bi isti atribut upućivao na lokaciju vrhova na kojima će se ostvariti proces sjenčanja. Lokacija je u konačnici dobivena kao rezultat pomnoženih matrica za transformaciju lokalnog prostora u prostor „svijeta“, matrice za transformaciju pozicije iz prostora svijeta u koordinatni sustav kamere, te na kraju matrice koja transformira poziciju koordinatnog sustava kamere u prostor potreban za projekciju. Te matrice redom se nazivaju *model, view, proj,* koje su i uobičajeni nazivi jer predstavljaju svoju konkretnu radnju. Koordinate, konačno su, x, y i z koordinata iz *„aPos“*, kao što i ime govori to je atribut pozicije, uz dodatak 1.0 vrijednosti koja je potrebna pri množenju matrica. Naziv vektora boje koji se šalje programu za sjenčanje fragmenata nužno mora biti isti kao onaj unutar samog koda programa za sjenčanje fragmenata. Razlika je jedino u modifikatorima, odnosno, u programu za sjenčanje vrhova koristi se „out“ modifikator čija je svrha prepoznavanje boje kao izlaznog vektora koji će se prenijeti u program za sjenčanje fragmenata. Tome slično, „in“ modifikator u programu za sjenčanje fragmenata služi za definiranje boje kao ulaznog vektora, kojemu će se u konačnici dodijeliti još jedan parametar, odnosno konačna će boja imati vrijednosti r, g i b za definiranje boje, te vrijednost alpha za definiranje njezine prozirnosti. U sintezi programa za sjenčanje vrhova i programa za sjenčanje fragmenata, konstruira se jedinstveni program za flat sjenčanje. Rezultat njegove primjene prikazan je u nastavku. Ovakav oblik sjenčanja najjednostavniji je, također je najmanje realističan što se vidi iz prikazanog.



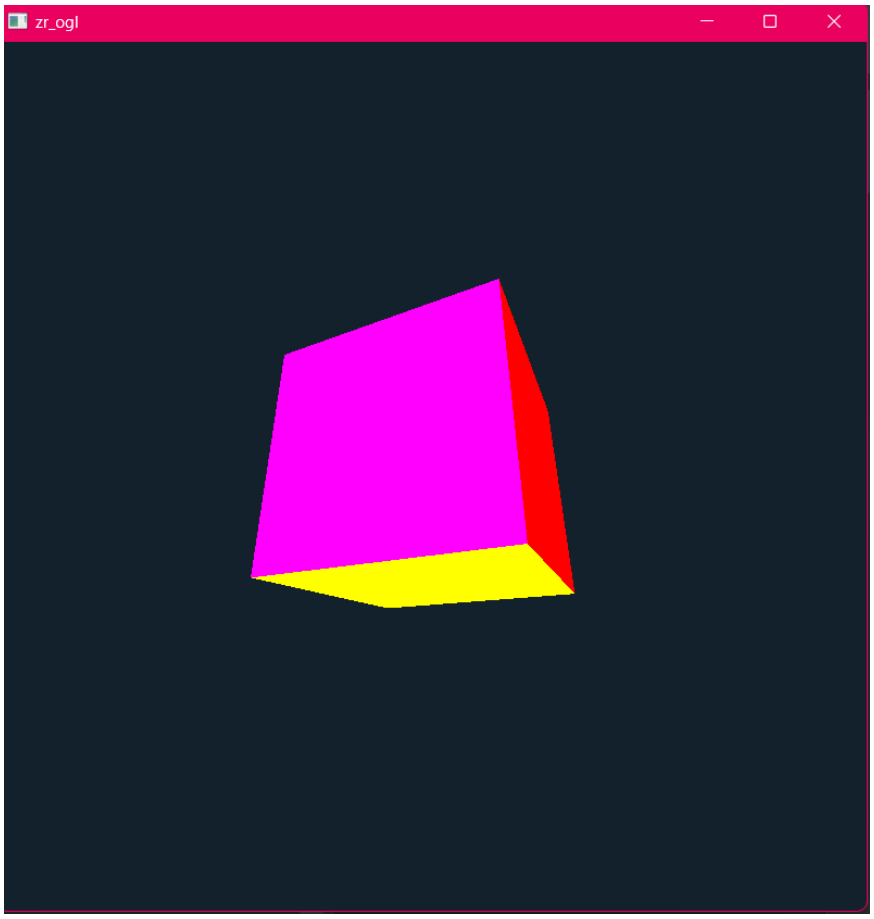
Slika 4.3: *Uvećani prikaz modela sfere osjenčanog kontinuiranim sjenčanjem*

Sfera je vidno osjenčana kako bi bio osjenčan oblik koji nema zaobljena svojstva, primjerice kocka, prikazana u nastavku. Razlog tomu je korištenje algoritma sjenčanja primjenjivog za bojanje i prikaz ravnih površina, kao što je to slučaj kod svih stranica kocke. Problem nastaje upravo u sjenčanju zaobljenih oblika jer kontinuirano sjenčanje ne može pratiti zaobljenost oblika i simulirati glatke prijelaze. Budući da je sfera kao model napravljena od primitivnih oblika, odnosno trokuta, flat sjenčanje primjenjuje se po svakom primitivnom obliku zasebno što rezultira nerealističnom prikazu obojenja sfere. Flat sjenčanje dobro funkcionira za sjenčanje kocke prikazane u nastavku.



Slika 4.4: *Prikaz kontinuiranog sjenčanja, uz pripadne programe za sjenčanje vrhova i fragmenata, te model kocke osjenčan kontinuiranim sjenčanjem*

Identično kontinuirano sjenčanje, odnosno program za sjenčanje vrhova i program za sjenčanje fragmenata primijenjeni su na dva potpuno različita oblika. Važna je stavka prilikom prikaza algoritama sjenčanja odabrati oblik koji će najbolje prikazati sjenčanje. Iz tog razloga prikazane su i sfera i kocka kako bi bili vidljive posljedice krivo odabranog oblika, u ovom slučaju, sfere. Rotacija iste kocke prikazana je u nastavku.



Slika 4.5: *Uvećani prikaz modela kocke osjenčanog kontinuiranim sjenčanjem*

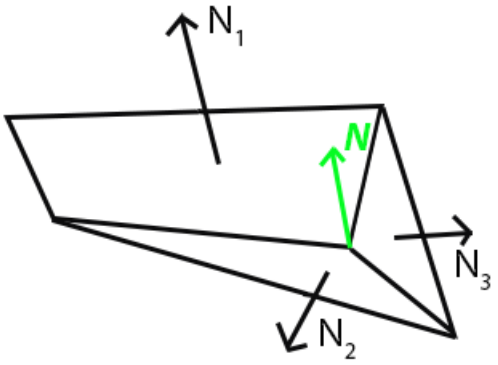
4.3. Algoritam Gouraudovog sjenčanja

Nakon što je odabran model osvjetljenja koji je potreban za izračun intenziteta osvjetljenja, glavna razlika u odnosu na algoritam konstantnog sjenčanja je što se izračunava intenzitet za svaki vrh poligona od kojega je sačinjen model. Budući da su poligoni najčešće trokuti, znači da se intenzitet osvjetljenja računa za njegova 3 pripadajuća vrha. Ako za primjer uzmemo Phongov model osvjetljenja, gdje difuzna komponenta svake točke ovisi o vektoru svjetla i vektoru normale, te u konačnici njihovom skalarnom produktu. Kao što je već bila riječ o aproksimaciji modela koristeći niz poligona za prikaz njegova plašta, upravo se sjenčanje modela svodi na sjenčanje poligona. Kada je to razlučeno, za svaki od pripadajućih poligona računa se normala vrhova. Na temelju svih izračunatih normala, računa se srednja normala. Srednja normala bit će zaslužna za izračun intenziteta, a računa se kao aritmetička sredina normala svih poligona koji dijele isti vrh, odnosno svih susjednih poligona koji u jednoj zajedničkoj točci imaju zajedničku srednju normalu. Nakon izračuna intenziteta svjetlosti u svim vrhovima poligona, započinje proces sjenčanja. Već izračunati intenziteti svjetlosti za svaki vrh najčešće se međusobno razlikuju unutar poligona.

Interpolacijom se dobivaju vrijednosti intenziteta u za svaki brid poligona, potom i za svaku točku krenuvši od lijevih ka desnim bridovima poligona. Takva interpolacija se još naziva i bilinearna interpolacija.

4.3.1. Matematički eksperimentalni dio algoritma Gouraudovog sjenčanja

Uzmimo za primjer tijelo dano u nastavku.



Slika 4.6: *Srednja normala poligona*

Vidljivo je kako je tijelo sačinjeno od 3 poligona s pripadajućim normalama $\overrightarrow{N_1}$, $\overrightarrow{N_2}$, $\overrightarrow{N_3}$. Normale $\overrightarrow{N_2}$, $\overrightarrow{N_3}$izračunate su pomoću formule u nastavku:

$$\vec{N}_i = \vec{a}_i \text{ x } \vec{b}_i = \begin{vmatrix} \vec{\imath} & \vec{\jmath} & \vec{k} \\ a_{ix} & a_{iy} & a_{iz} \\ b_{ix} & b_{iy} & b_{iz} \end{vmatrix} = \vec{\imath} \begin{vmatrix} a_{iy} & a_{iz} \\ b_{iy} & b_{iz} \end{vmatrix} - \vec{\jmath} \begin{vmatrix} a_{ix} & a_{iz} \\ b_{ix} & b_{iz} \end{vmatrix} + \vec{k} \begin{vmatrix} a_{ix} & a_{iy} \\ b_{ix} & b_{iy} \end{vmatrix} \quad (4\text{-}10)$$

$$\text{za } i = 2{,}3$$

Prikazana dva od tri poligona su trokuti koji imaju vrhove A, B i C s pripadajućim koordinatama. Vektori $\vec{a}$ i $\vec{b}$ iz (4-10) vektori su određeni dvama vrhovima pripadnog im trokuta, pri čemu im je jedan vrh zajednički. Jedan od tri poligona je četverokut, te se srednja normala četverokuta $\overrightarrow{N_1}$ računa rastavom četverokuta na dva trokuta.

$$\overrightarrow{N_1} = \frac{\overrightarrow{N_{11}} + \overrightarrow{N_{12}}}{2} \quad (4\text{-}11)$$

pri čemu su $\overrightarrow{N_{11}}$ i $\overrightarrow{N_{12}}$ normale trokuta na koje je četverokut rastavljen, a računaju se prema izrazu (4-10) nakon čega se dijeli s 2, odnosno brojem trokuta u rastavu. Konačno se dobije srednja normala četverokuta $\overrightarrow{N_1}$.

Na temelju svih normala poligona koji dijele zajednički vrh prema primjeru Slika 1, računa se srednja normala vrha tijela prema izrazu:

$$\vec{N} = \frac{1}{n}\sum_{i=1}^{n} \vec{N}_i \tag{4-12}$$

Nakon što je poznata srednja normala poligona $\vec{N}$ i definiran smjer svjetlosti izvora svjetlosti $\vec{L}$, potrebno je za svaki od vrhova izračunati intenzitet. Intenzitet osvjetljenja I računa se po Phongovom modelu čiji je ukupan utjecaj linearna kombinacija komponente ambijentalnog, difuznog i sjajnog sjenčanja opisanih u prethodnim poglavljima. Formulacija sljedeća: Phongov model osvjetljenja

$$I = I_g + I_d + I_s \tag{4-13}$$

$$I_g = I_a \cdot k_a \tag{4-14}$$

pri čemu jeI_a vrijednost svjetlosti koja će biti apsorbirana i reflektirana, ovisno o koeficijentuk_a.

$$I_d = I_i \cdot k_d \cdot (\vec{L} \cdot \vec{N}), \tag{4-15}$$

pri čemu je I_i intenzitet izvora svjetlosti, koeficijent k_d zadužen je za prikaz refleksije, a ovisi o valnoj duljini svjetlosti upadne zrake, $\vec{L}$ vektor ka izvoru svjetlosti iz točke promatranja, $\vec{N}$ vektor normale

$$I_s = I_i \cdot k_s \cdot \left(\vec{R} \cdot \vec{V}\right)^n \tag{4-16}$$

pri čemu je I_i intenzitet izvora svjetlosti, koeficijent $k_s$prikazuje ovisnost o materijalu, oznaka n je koeficijent (skalar) korišten za opisivanje grubоće površine i pripadnih joj reflektirajućih svojstava, $\vec{R}$ vektor je reflektirane zrake, dok je $\vec{V}$ vektor ka smjeru gledišta. Prethodno opisani intenzitet osvjetljenja računa se za svaki vrh pojedinačno. Ukoliko je poligon četverokut, intenzitet I računa se za četiri vrha, te se dobivaju intenziteti $I_1, I_2, I_3, te\ I_4$. Na taj način svaki od vrhova poligona ima vlastiti intenzitet osvjetljenja. Intenziteti se ponajprije interpoliraju po bridovima, te se tako dobivaju poznate vrijednosti intenziteta svake točke svakog pojedinog brida poligona. Nakon tog koraka, interpoliraju se intenziteti krenuvši od lijevih ka desnim bridovima. Ovakve interpolacije izvode se postupkom DDA (Digital Differential Analyzer) za generiranje točaka po linijama koje će povezivati svaki vrh poligona, te se zatim vrši interpolacija intenziteta između točaka kako bi se postigla glatka tranzicija osvjetljavanja poligona. Intenziteti $I_1, I_2, I_3, te\ I_4$ imaju pripadne koordinate $I_1(x_1, y_1),\ I_2(x_2, y_2),\ I_3(x_3, y_3),\ I_4(x_4, y_4)$. Ukoliko postoji točka $I_a(x_a, y_a)$ na lijevom bridu poligona, odnosno na bridu kojeg povezuju vrhovi I_1 i I_4, te postoji točka $I_b(x_b, y_b)$ na desnom bridu poligona, odnosno bridu kojeg povezuju I_1 i I_2, moguće je izračunati

intenzitet $I_{sr}(x_{sr}, y_{sr})$ koji se nalazi na liniji koji povezuje točke I_a i I_b bilinearnom interpolacijom (linerana interpolacija po $y\,osi$, zatim interpolacija interpoliranih vrijednosti duž $x\,osi$). Formulacija je sljedeća:

Intenzitet I_a za točku (x_a, y_{sr}) dobiva se interpolacijom između I_1i I_4:

$$\mathrm{I_a} = \frac{1}{y_4 - y_1}\big(I_1(y_4 - y_{sr}) + I_4(y_{sr} - y_1)\big) \tag{4-17}$$

Intenzitet I_b za točku (x_b, y_{sr}) dobiva se interpolacijom između I_1 i I_2:

$$\mathrm{I_b} = \frac{1}{y_2 - y_1}\big(I_1(y_2 - y_{sr}) + I_2(y_{sr} - y_1)\big) \tag{4-18}$$

Konačno, interpolacijom intenziteta I_a i I_b dobije se intenzitet točke (x_{sr}, y_{sr}):

$$\mathrm{I_{sr}} = \frac{1}{x_b - x_a}\big(I_a(x_b - x_{sr}) + I_b(x_{sr} - x_a)\big) \tag{4-19}$$

Prethodni postupak ponavlja se za svaki piksel unutar poligona.

4.3.2. Programski eksperimentalni dio algoritma Gouraudovog sjenčanja

Program za sjenčanje vrhova Gouraudovog sjenčanja

```
#version 330 core
layout(location=0) in vec3 inPosition;  // normale vrhova
layout(location=2) in vec3 inNormal;  // normale vrhova

out vec3 FragColor; // izlazna boja za fragment shader
out vec3 FragPos;  // izlazna pozicija vrhova za racunanje spekularnog
osvjetljenja
out vec3 Normal;  // izlazna normala za racunanje spekularnog osvjetljenja

uniform mat4 model; // Model matrica
uniform mat4 view; // View matrica
uniform mat4 proj; // Projekcijska matrica
uniform vec3 lightPosition; // pozicija svjetlosti

uniform vec3 ambientColor; // Ambijentna boja
uniform vec3 diffuseColor; // Difuzna boj
uniform vec3 specularColor; // Spekularna boja
uniform float shininess;  // Faktor sjaja

void main(){
```

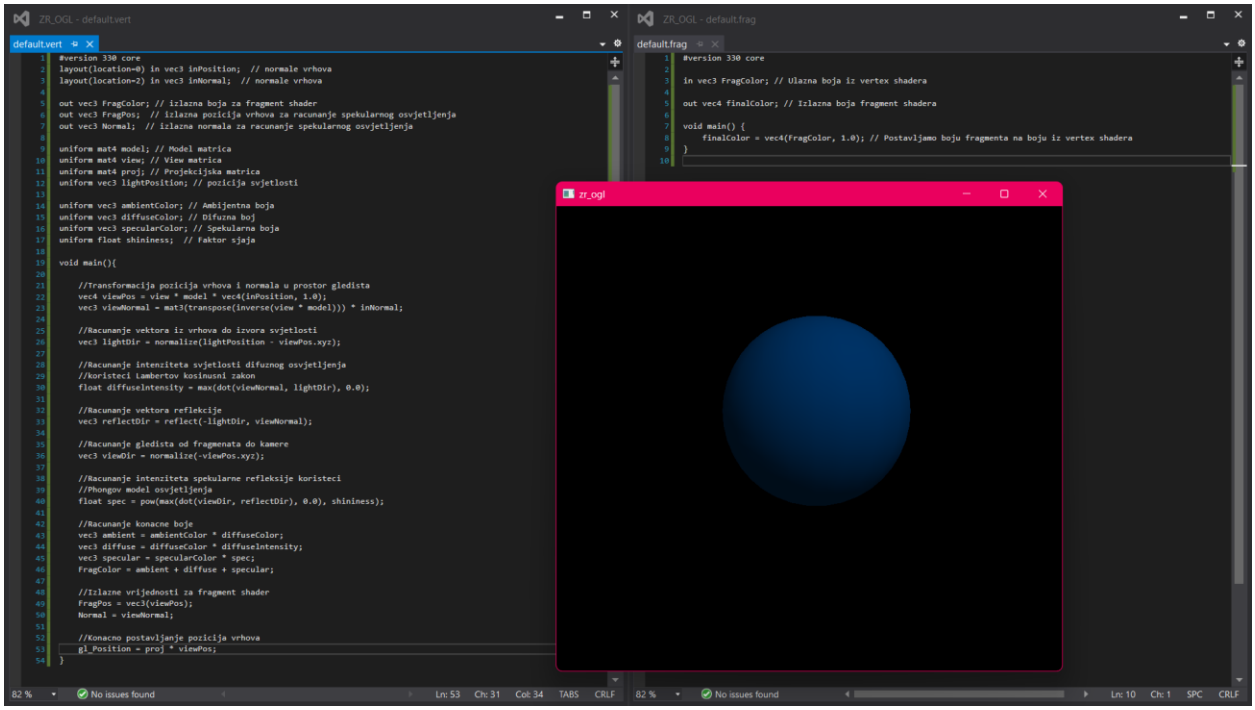
```
    //Transformacija pozicija vrhova i normala u prostor gledista
    vec4 viewPos = view * model * vec4(inPosition, 1.0);
    vec3 viewNormal = mat3(transpose(inverse(view * model))) * inNormal;

    //Racunanje vektora iz vrhova do izvora svjetlosti
    vec3 lightDir = normalize(lightPosition - viewPos.xyz);

    //Racunanje intenziteta svjetlosti difuznog osvjetljenja
    //koristeci Lambertov kosinusni zakon
    float diffuseIntensity = max(dot(viewNormal, lightDir), 0.0);

    //Racunanje vektora reflekcije
    vec3 reflectDir = reflect(-lightDir, viewNormal);

    //Racunanje gledista od fragmenata do kamere
    vec3 viewDir = normalize(-viewPos.xyz);

    //Racunanje intenziteta spekularne refleksije koristeci
    //Phongov model osvjetljenja
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), shininess);

    //Racunanje konacne boje
    vec3 ambient = ambientColor * diffuseColor;
    vec3 diffuse = diffuseColor * diffuseIntensity;
    vec3 specular = specularColor * spec;
    FragColor = ambient + diffuse + specular;

    //Izlazne vrijednosti za fragment shader
    FragPos = vec3(viewPos);
    Normal = viewNormal;

    //Konacno postavljanje pozicija vrhova
    gl_Position = proj * viewPos;
}
```

Program za sjenčanje fragmenata Gouraudovog sjenčanja

```
#version 330 core

in vec3 FragColor; // Ulazna boja iz vertex shadera
out vec4 finalColor; // Izlazna boja fragment shadera
```

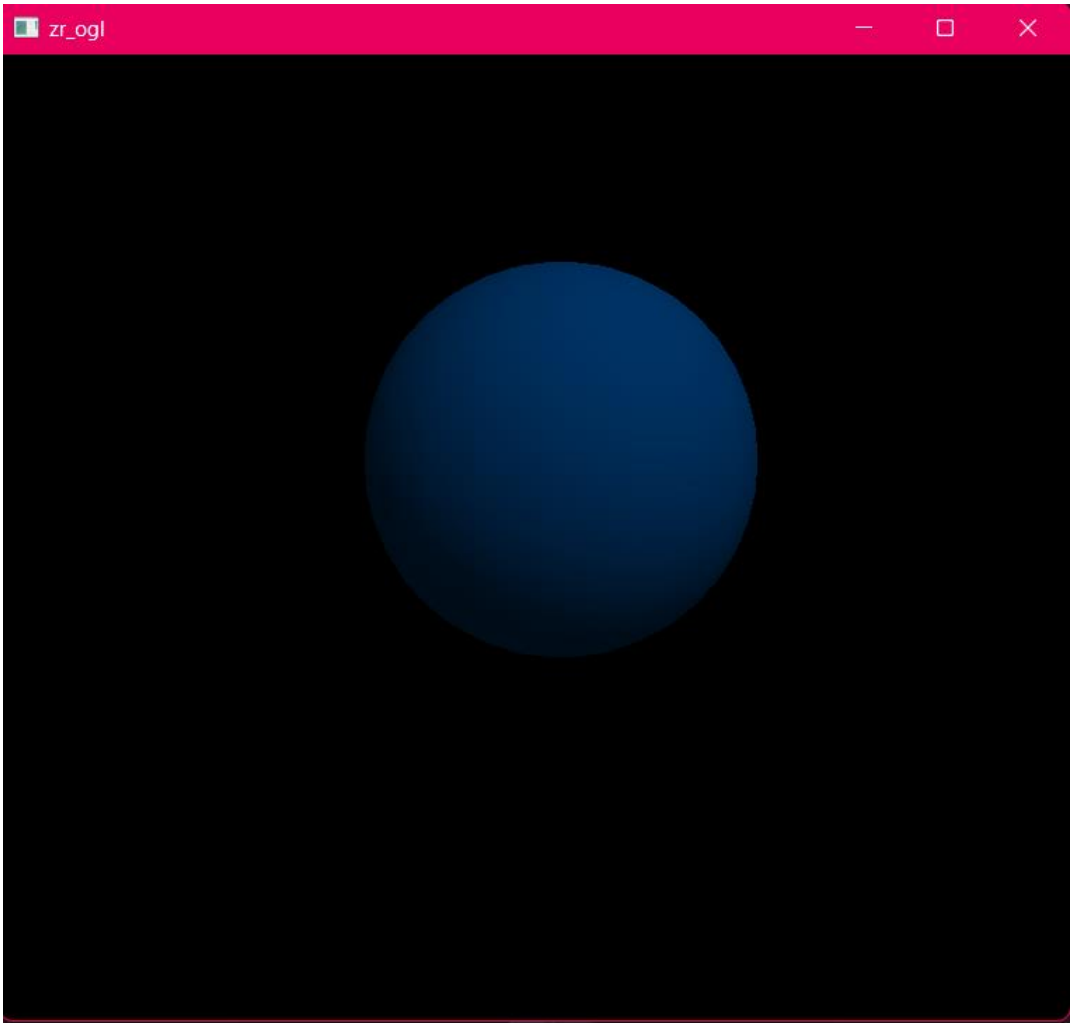
```
void main() {
    finalColor = vec4(FragColor, 1.0); // postavljanje boje fragmenta
na boju iz vertex shadera
}
```

Procedura sjenčanja algoritmom Gouraudovog sjenčanja počinje izvedbom programa za sjenčanje vrhova, konkretno krenuvši s transformacijom pozicija vrhova i normala u prostoru gledišta. Navedena transformacija ostvaruje se umnoškom matrica modela, pogleda te u konačnici ulaznih pozicija vrhova od kojih je načinjena sfera na koju se primjenjuje Gouraudovo sjenčanje. Provodi se izračun vektora čije usmjerenje je prema izvoru svjetlosti krenuvši od vrha, također računa se intenzitet difuznog osvjetljenja pomoću Lambertovog kosinusnog zakona već opisan u radu. Korištenje Lambertovog kosinusnog zakona u ovom slučaju potrebno je za izračun koliko svjetlosti će vrh poprimiti iz izvora svjetlosti. Korišten je također i Phongov model osvjetljenja, također opisan u radu. Pomoću njega uvodi se prividni odsjaj na površini. Kako bi se dobila konačna boja, nužno je upotrijebiti prethodno izračunate kombinacije ambijentalne, difuzne i spekularne boje. Nakon izračuna konačne boje, potrebno je postaviti poziciju vrha kako bi se ista prikazala na ekranu nakon primjene transformacije modela, pogleda, te projekcije. Program za sjenčanje fragmenata Gouraudovog sjenčanja kao ulazni podatak prima izlaznu boju iz programa za sjenčanje vrhova. Konačna boja primjenjuje se na željenom obliku koji se sjenča. U nastavku je prikazan prethodno objašnjen programski kod, uz izlazni rezultat sjenčanja primijenjen na sferi.



Slika 4.7: *Prikaz Gouraudovog sjenčanja, uz pripadne programe za sjenčanje vrhova i fragmenata, te model sfere osjenčan Gouraudovim sjenčanjem*

Jasno je vidljiva razlika između algoritma kontinuiranog sjenčanja i algoritma Gouraudovog sjenčanja. Sfera poprima oblik klasične sfere, odnosno, prijelazi sada izgledaju zaglađeno, dok to nije bio slučaj prilikom korištenja algoritma kontinuiranog sjenčanja. Tomu je razlog računanje osvjetljenja i boje za svaki vrh objekta, te interpolacija istih vrijednosti između vrhovima što rezultira primjetnu glatku tranziciju površinske boje. Već je bilo riječi o odabiru pravilnog oblika na kojemu će se provesti sjenčanje. Budući da se Gouraudovo sjenčanje koristi za osjenčavanje glatkih i zaobljenih površina, logičan izbor oblika bila je sfera. Uvećani rezultat osjenčane sfere prikazan je u nastavku.



Slika 4.8: *Uvećani prikaz modela sfere osjenčanog Gouraudovim sjenčanjem*

4.4. Algoritam Phongovog sjenčanja

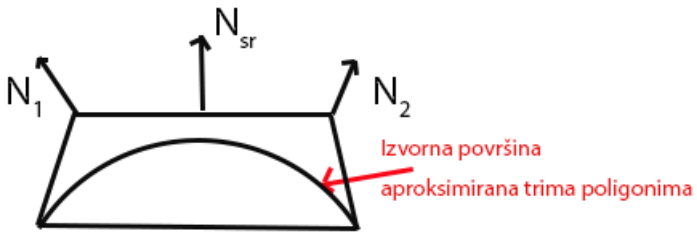
U odnosu na algoritam konstantnog sjenčanja i algoritam Gouraudovog sjenčanja, algoritam Phongovog sjenčanja zahtjevniji je za računalo, no daje bolje rezultate od oba navedena. Algoritam Phongovog sjenčanja također koristi bilinearnu interpolaciju, no u ovom slučaju, umjesto interpolacije intenziteta vrhova i bridova, interpoliraju se normale. Cilj interpolacije normala je što će svaki piksel na temelju izračunate normale, ovakvim postupkom, imati i izračunat prikladni intenzitet svjetlosti. Zbog ovakvih detaljnih izračuna za svaki piksel, može se reći da Phongovo sjenčanje najbolje prati zakrivljenost površine iako je predstavljena aproksimacijom niza poligona.

4.4.1. Matematički eksperimentalni dio algoritma Phongovog sjenčanja

Postupak izračuna algoritma Phongovog sjenčanja kreće s računanjem normala vrhova prema izrazu:

$$\vec{N} = \frac{1}{n}\sum_{i=1}^{n} \vec{N}_i \tag{4-20}$$

Normale koje se računaju prikazane su slikom u nastavku (radi jednostavnosti računa i zaključaka, zakrivljenost prate samo tri poligona, dok bi ih inače bilo puno više za bolje praćenje krivulje izvorne površine). Vidljivo je kako su zajedničke normale $\overrightarrow{N_1}$ i $\overrightarrow{N_2}$. Ukoliko $\overrightarrow{N_{sr}}$ računamo bilinearnom interpolacijom, prividni dojam je kako izračunata normala slijedi zakrivljenost promatrane površine, prema tome je i algoritam sjenčanja realističniji od Gouraudovog algoritma sjenčanja poligona.



Slika 4.9: *Aproksimacija zakrivljene površine*

Idući dio postupka koristi bilinearnu interpolaciju, no umjesto računanja intenziteta za danu koordinatu, računa se normala za danu koordinatu točke. Krajnja $\overrightarrow{N_{sr}}$ normala računa se za svaki pojedinačni piksel unutar poligona.

Normala N_a za točku (x_a, y_{sr}) dobiva se interpolacijom između $\overrightarrow{N_1}$ i $\overrightarrow{N_4}$:

$$\overrightarrow{N_a} = \frac{1}{y_4 - y_1}\left(\overrightarrow{N_1}(y_4 - y_{sr}) + \overrightarrow{N_4}(y_{sr} - y_1)\right) \tag{4-21}$$

Normala N_b za točku (x_b, y_{sr}) dobiva se interpolacijom između $\overrightarrow{N_1}$ i $\overrightarrow{N_2}$:

$$\overrightarrow{N_b} = \frac{1}{y_2 - y_1}\left(\overrightarrow{N_1}(y_2 - y_{sr}) + \overrightarrow{N_2}(y_{sr} - y_1)\right) \tag{4-22}$$

Konačno, interpolacijom normala $\overrightarrow{N_a}$ i $\overrightarrow{N_b}$ dobije se normala točke (x_{sr}, y_{sr}):

$$\overrightarrow{N_{sr}} = \frac{1}{x_b - x_a}\left(\overrightarrow{N_a}(x_b - x_{sr}) + \overrightarrow{N_b}(x_{sr} - x_a)\right) \tag{4-23}$$

4.4.2. Programski eksperimentalni dio algoritma Phongovog sjenčanja

Program za sjenčanje vrhova Phongovog sjenčanja

```
#version 330 core
layout(location = 0) in vec3 inPosition;  // Pozicija vrha trokuta
layout(location = 2) in vec3 inNormal;    // Normala vrha trokuta

out vec3 FragPos;    // Pozicija fragmenta
out vec3 Normal;     // Normala fragmenta

uniform mat4 model;     // Model matrica
uniform mat4 view;      // Matrica pogleda
uniform mat4 proj;      // Matrica projekcije
uniform vec3 lightPosition;  // Pozicija svjetla

void main()
{
    //  pozicija vrha i normala
    vec4 worldPosition = model * vec4(inPosition, 1.0);
    FragPos = worldPosition.xyz;
    Normal = normalize(mat3(transpose(inverse(model))) * inNormal);

    //  pozicija vrha u koordinatnom prostoru kamere
    vec4 viewPosition = view * worldPosition;
```

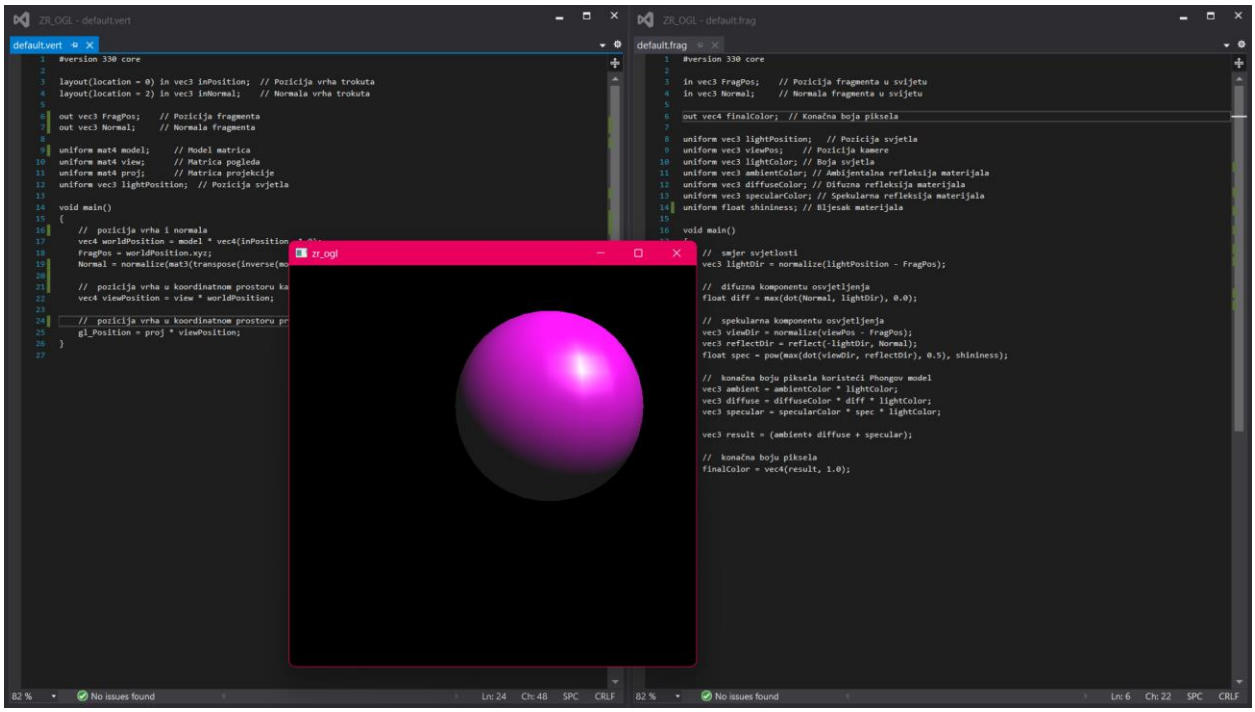
```
    //  pozicija vrha u koordinatnom prostoru projekcije
    gl_Position = proj * viewPosition;
}
```

Program za sjenčanje fragmenata Phongovog sjenčanja

```
#version 330 core

in vec3 FragPos;    // Pozicija fragmenta u svijetu
in vec3 Normal;     // Normala fragmenta u svijetu

out vec4 finalColor;  // Konačna boja piksela

uniform vec3 lightPosition;   // Pozicija svjetla
uniform vec3 viewPos;    // Pozicija kamere
uniform vec3 lightColor; // Boja svjetla
uniform vec3 ambientColor; // Ambijentalna refleksija materijala
uniform vec3 diffuseColor; // Difuzna refleksija materijala
uniform vec3 specularColor; // Spekularna refleksija materijala
uniform float shininess; // Bljesak materijala

void main()
{
    //  smjer svjetlosti
    vec3 lightDir = normalize(lightPosition - FragPos);
    //  difuzna komponentu osvjetljenja
    float diff = max(dot(Normal, lightDir), 0.0);

    //  spekularna komponentu osvjetljenja
    vec3 viewDir = normalize(viewPos - FragPos);
    vec3 reflectDir = reflect(-lightDir, Normal);
    float spec = pow(max(dot(viewDir, reflectDir), 0.5), shininess);

    //  konačna boju piksela koristeći Phongov model
    vec3 ambient = ambientColor * lightColor;
    vec3 diffuse = diffuseColor * diff * lightColor;
    vec3 specular = specularColor * spec * lightColor;
```

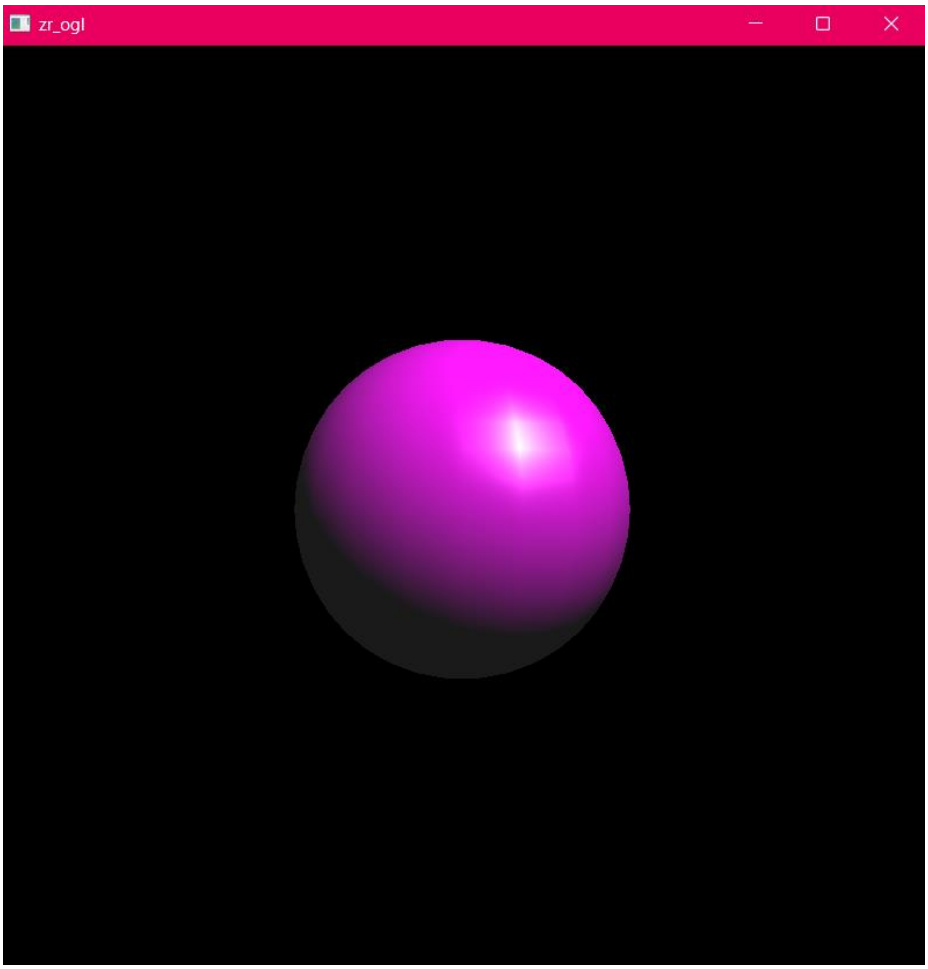
```
    vec3 result = (ambient+ diffuse + specular);

    //  konačna boja piksela
    finalColor = vec4(result, 1.0);
}
```

Prilikom početka izvršavanja programa za sjenčanje vrhova Phongovog sjenčanja deklarirani su ulazni podaci o pozicijama vrhova, te normalama. Također, deklarirane su izlazne vrijednosti, FragPos i Normal koje koriste out modifikator kako bi se označile kao izlazne vrijednosti, u budućnosti prenesene programu za sjenčanje fragmenata Phongovog sjenčanja. Uniformne matrice definirane u glavnom dijelu programa služe za transformacije pogleda i svjetlosti na sljedeći način: računa se transformacija pozicije vrha iz trenutnog prostora u prostor svijeta, pozicija fragmenta također se prilagođava u prostor svijeta, normale se normaliziraju (to je nužan korak kako ne bi došlo do neočekivanih rezultata), te se također prilagođavaju za prostor svijeta. Na kraju, pozicija vrha transformira se u prostor projekcije. Glavna zadaća idućeg dijela Phongovog sjenčanja, odnosno programa za sjenčanje fragmenata, konstruiranje je konačne boje koja će biti primijenjena na odabrani model, koji je ponovno sfera. Definira se izlazna varijabla boje, uniformni vektori definirani su u glavnom dijelu programa, a to su pozicija svjetlosti, pozicija pogleda, boja svjetla, ambijentalna, difuzna, te spekularna komponenta boje, te odsjaj. U glavnoj funkciji računa se smjer svjetla krenuvši od pozicije svjetla ka fragmentu. Konačna ambijentalna, difuzna i spekularna refleksija računaju se pomoću Phongovog modela osvjetljenja, te se rezultat prikazuje kao kombinacija istih. Objašnjeni postupak prikazan je sljedećom fotografijom zaslona na kojemu se nalaze programi za sjenčanje vrhova i fragmenata Phongovog sjenčanja, te njihova primjena na sferi.



Slika 4.10: *Prikaz Phongovog sjenčanja, uz pripadne programe za sjenčanje vrhova i fragmenata, te model sfere osjenčan Phongovim sjenčanjem*

Vidljiva razlika u odnosu na kontinuirano sjenčanje, također i na Gouraudovo sjenčanje je svakako stupanj realizma. Phongovo sjenčanje, svakako je najkompleksniji oblik sjenčanja jer se primjenjuje na normalu svakog vrha, odnosno interpolacija normale i pozicije fragmenta iz programa za sjenčanje vrhova Phongovog sjenčanja koriste se u programu za sjenčanje fragmenata za izračunavanje konačne boje i osvjetljenja za svaki fragment, što nije bio slučaj ni u kontinuiranom sjenčanju, niti u Gouraudovom sjenčanju. Dakako, realizmu doprinose i uključenje svih oblika refleksija, odnosno ambijentalna, difuzna te spekularna reflekcija u kombinaciji. Ambijentalna komponenta, kao i kod Gouraudovog sjenčanja stvara efekt mekog i ravnomjernog osvjetljenja po cijeloj sferi, te doprinosi zaobljenom izgledu. Razlog tomu je što je ambijentalna komponenta zadužena za ravnomjerno osvjetljenje, bez obzira na položaj izvora svjetlosti. Difuzna komponenta pruža osvjetljenje na dijelovima u kojima svjetlost direktno pogađa površinu sfere, čineći tu površinu svjetlijom. Cijelom izgledu i realističnom dojmu doprinosi spekularna komponenta zadužena za dojam sjaja i glatke teksture sferne površine. Sve komponente primijenjene su na sferi prikazanoj u nastavku.



Slika 4.11: *Uvećani prikaz modela sfere osjenčanog Phongovim sjenčanjem*

4.5. Razlike između flat i smooth sjenčanja

Različitosti između flat i smooth sjenčanja vrlo su očigledna, prvenstveno iz perspektive realističnog renderiranja. Flat sjenčanje, drugim riječima, konstantno/kontinuirano sjenčanje jedna je od jednostavnijih tehnika sjenčanja. Poligoni u prikazanom modelu posjeduju istu boju, bez glatkih prijelaza između istih. Boja poligona računa se samo jednom, po vrhu ili po središtu istog, te se sjenča kao ravna ploha, ne uzimajući u obzir eventualnu zaobljenost. Glavni nedostatak flat sjenčanja je manjak realizma, budući da nema vidljivih glatkih prijelaza između obojenja, aproksimacija zaobljenosti ploha je slaba, te se gubi detaljnost potrebna za prikaz realistične scene. Smooth ili glatko sjenčanje tehnika je u kojoj se vrši interpolacija svjetlosti piksela od kojih je poligon tvoren. Interpolacijom svjetlosti model poprima obilježja zaglađenosti. Implementacija glatkog sjenčanja računalno je zahtjevnija od kontinuiranog sjenčanja, no pruža dojam realističnosti promatrane scene i objekata u istoj. Nedostatak je potencijalna smanjena brzina renderiranja u ovisnosti broja i veličine objekata u sceni.

5. ZAKLJUČAK

Algoritmi sjenčanja u 3D prostoru sastavni su dio operacija u 3D grafici. Algoritmi sjenčanja opisani u ovom radu, algoritam kontinuiranog sjenčanja, algoritam Gouraudovog sjenčanja, te algoritam Phongovog sjenčanja. Isti algoritmi jedni su od najbazičnijih algoritama sjenčanja koji prikazuju funkcionalnosti prijenosa boje i utjecaja osvjetljenja na prikaze realističnih, ali i onih manje realističnih oblika i modela u sceni. Njihovo projektiranje i uspostavljanje provedeno je korištenjem OpenGL-a, 3D grafičke višeplatformska biblioteke/specifikacije. OpenGL nije programski jezik, te se u ovom radu, uz njegovo korištenje, ostatak koda pisao programskim jezikom C++. Za pravilno konstruiranje prethodno navedenih algoritama, nužno je bilo pratiti proces OpenGL proces iscrtavanja, te su ključni različiti dijelovi, kao što su program za sjenčanje fragmenata, te program za sjenčanje vrhova, prikazani u radu. Konstruiranje prikladnog osvjetljenja, također je imalo veliki značaj u samom izvođenju algoritama, te je utjecalo na krajnji rezultat svakog sjenčanja. Algoritam kontinuiranog sjenčanja, kako je i matematički potkrijepljeno, kao polazišnu točku boje uzima jednu boju unutar primitiva, u svim slučajevima, primitiv je bio trokut. Takva boja primijenjena je na cijelom obliku. Algoritam kontinuiranog sjenčanja zato je prikladan za prikazivanje oblika i modela s ravnim plohama, kao što je to primjer kocka, dok ne funkcionira za zaobljene oblike kao što je to sfera. Iduća dva algoritma sjenčanja, algoritam Gouraudovog sjenčanja, te algoritam Phongovog sjenčanja, kompleksniji su te su primjenjivi najviše za modele i oblike zaobljenih površina. Razlika između algoritma Gouraudovog sjenčanja, te algoritma Phongovog sjenčanja je interpolacija, odnosno izvršavanje iste. Kod Gouraudovog sjenčanja, boja se interpolira između vrhova, dok se kod Phongovog sjenčanja boja interpolira između normala tih istih vrhova čineći u konačnici oblik realističnijim.

LITERATURA

[1] B.-C. Shih, Y.-H. Yeh i C.-Y. Lee, »An Area Efficient Architecture For 3d Graphics Shading,« IEEE, Los Angeles, 2002..

[2] S. Dewen i W. Hongxia, »Realistic Real-time Rendering of 3D Terrain Scenes,« IEEE, Hangzhou, 2009..

[3] R. Fabio, »FROM POINT CLOUD TO SURFACE: THE MODELING AND VISUALIZATION PROBLEM,« Swiss Federal Institute of Technology, Zurich, 2003..

[4] M. S. Peercy, M. Olano, J. Airey i J. P. Ungar, »Interactive Multi-Pass Programmable Shading,« ACM Press/Addison-Wesley Publishing Co., New York, 2000..

[5] J. S. Joon , S. E. Hui i Y. M. Chan, »Understanding the Technicalities of Photorealistic 3D Environments to Support Cinematography and Composition for Film and Animation,« IEEE, Penang, 2008..

[6] T. Thormahlen, »Graphics Programming Cameras: Parallel Projection,« Phillips Universitat Marburg, 2021.. [Mrežno]. Available: https://www.mathematik.uni-marburg.de/~thormae/lectures/graphics1/graphics_6_2_eng_web.html#1.

[7] A. Vu, »Light Setting in OpenGL,« 2021.. [Mrežno]. Available: https://blogs.oregonstate.edu/learnfromscratch/2021/10/13/light-setting-in-opengl/.

[8] M. Bailey, »Using Shaders for Lighting, Oregon State University Computer Graphics,« 5 Prosinac 2022.. [Mrežno]. Available: https://web.engr.oregonstate.edu/~mjb/cs557/Handouts/lighting.1pp.pdf. [Pokušaj pristupa 25 Lipanj 2023.].

[9] M. Reed, »Computer Graphics (Spring 2008) COMS 4160, Lecture 20: Illumination and Shading 2,« Department of Computer Science Columbia University, New York, 2008..

[10] G. Papaioannou, »COMPUTER GRAPHICS COURSE: Materials and Appearance: An Introduction,« AUEB Computer Graphics Group, 2014.. [Mrežno]. Available: https://eclass.aueb.gr/modules/document/file.php/INF142/%CE%A0%CE%B1%CF%81

%CE%B1%CE%B4%CF%8C%CF%83%CE%B5%CE%B9%CF%82/ComputerGraphics-Materials_and_Appearance.pdf.

[11] J. de Vries, »PBR Theory,« 2014.. [Mrežno]. Available: https://learnopengl.com/PBR/Theory.

[12] I. Toral, »Developer Log Working with lights and shadows – Part II: The shadow map,« 2017.. [Mrežno]. Available: https://blogs.igalia.com/itoral/2017/07/30/working-with-lights-and-shadows-part-ii-the-shadow-map/.

[13] V. Gordon i J. Clevenger, Computer Graphics Programming in OpenGL with C++, Dulles, Virginia: Mercury Learning and Information, 2021..

[14] »Portal: OpenGL Concepts,« 15 Rujan 2017.. [Mrežno]. Available: https://www.khronos.org/opengl/wiki/Portal:OpenGL_Concepts . [Pokušaj pristupa 26 Lipanj 2023.].

[15] J. de Vries, »OpenGL,« 2014.. [Mrežno]. Available: https://learnopengl.com/Getting-started/OpenGL. [Pokušaj pristupa 25. Lipanj 2023.].

[16] M. Bailey, »Introduction to using the OpenGL Shading Language (GLSL),« 22 Kolovoz 2022.. [Mrežno]. Available: https://web.engr.oregonstate.edu/~mjb/cs550/PDFs/Shaders.1pp.pdf. [Pokušaj pristupa 25. Lipanj 2023.].

[17] R. J. Rost, OpenGL® Shading Language, Second Edition, Stoughton: Addison Wesley Professional, 2006..

[18] M. Čupić i Ž. Mihajlović, »Interaktivna računalna grafika kroz primjere u OpenGL-u,« 8 Ožujak 2018.. [Mrežno]. Available: http://java.zemris.fer.hr/nastava/irg/knjiga-0.1.2018-03-08.pdf. [Pokušaj pristupa 25 Lipanj 2023.].

[19] R. S. Wright Jr., B. Lipchak i N. Haemel, OpenGL SuperBible, Fourth Edition, Boston: Addison-Wesley, 2007..

[20] B. Johnson, »Rendering Overview,« Department of Architecture, University of Washington, Travanj 2014. [Mrežno]. Available:

http://courses.washington.edu/arch481/1.Tapestry%20Reader/4.Rendering/0.default.html. [Pokušaj pristupa 25. Lipanj 2023.].

[21] »Rendering Pipeline Overview,« The Khronos Group Inc., 7 Studeni 2022.. [Mrežno]. Available: https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview. [Pokušaj pristupa 25. Lipanj 2023.].

[22] »Vertex Shader,« The Khronos Group Inc., 10 Studeni 2017.. [Mrežno]. Available: https://www.khronos.org/opengl/wiki/Vertex_Shader. [Pokušaj pristupa 25. Lipanj 2023.].

[23] »Vertex Post-Processing,« The Khronos Group Inc., 11 Veljača 2021.. [Mrežno]. Available: https://www.khronos.org/opengl/wiki/Vertex_Post-Processing. [Pokušaj pristupa 25. Lipanj 2023.].

[24] »Interpolation,« [Mrežno]. Available: https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/interpolation/introduction.html. [Pokušaj pristupa 25 Lipanj 2023.].

[25] »Fragment Shader,« The Khronos Gorup Inc., 25 Studeni 2020.. [Mrežno]. Available: https://www.khronos.org/opengl/wiki/Fragment_Shader. [Pokušaj pristupa 25. Lipanj 2023.].

[26] »Per-Sample Processing,« The Khronos Group Inc., 3 Travanj 2019.. [Mrežno]. Available: https://www.khronos.org/opengl/wiki/Per-Sample_Processing. [Pokušaj pristupa 25. Lipanj 2023.].

[27] F. Dunn i I. Parberry, 3D Math Primer for Graphics and Game Development, Second Edition, Boca Raton: Taylor and Francis Group, LLC, 2011..

[28] A. P. Godse i D. D. A. Godse, Computer Graphics and Multimedia, Pune: Technical Publications, 2021..

[29] »What is Shading: Light-Matter interaction,« [Mrežno]. Available: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/what-is-shading-light-matter-interaction.html. [Pokušaj pristupa 25. Lipanj 2023.].

[30] »Diffuse and Lambertian Shading,« [Mrežno]. Available: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/diffuse-lambertian-shading.html. [Pokušaj pristupa 25 Lipanj 2023.].

[31] R. Tunnel, J. Jaggo i M. Luik, »Shading and Lighting,« University of Tartu, [Mrežno]. Available: https://cglearn.codelight.eu/pub/computer-graphics/shading-and-lighting. [Pokušaj pristupa 25. Lipanj 2023.].

[32] J. de Vries, »Basic Lighting,« 2014.. [Mrežno]. Available: https://learnopengl.com/Lighting/Basic-Lighting.

[33] J. de Vries, »Materials,« 2014. [Mrežno]. Available: https://learnopengl.com/Lighting/Materials. [Pokušaj pristupa 25. Lipanj 2023.].

[34] J. de Vries, »Advanced Lighting,« 2014.. [Mrežno]. Available: https://learnopengl.com/Advanced-Lighting/Advanced-Lighting. [Pokušaj pristupa 25. Lipanj 2023.].

[35] J. de Vries, »Shadow Mapping,« 2014.. [Mrežno]. Available: https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping. [Pokušaj pristupa 25. Lipanj 2023.].

[36] J. de Vries, »Point Shadows,« 2014.. [Mrežno]. Available: https://learnopengl.com/Advanced-Lighting/Shadows/Point-Shadows. [Pokušaj pristupa 25 Lipanj 2023.].

[37] J. F. Hughes, A. Van Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. K. Feiner i K. Akeley, Computer Graphics Principles and Practice, Third Edition, Boston: Addison-Wesley, 2014..

[38] P. Novoselac, »Završni rad- Algoritmi sjenčanja u 3D prostoru,« 2023.. [Mrežno]. Available: https://gitlab.com/paulina.novoselac/zavrsni-rad-algoritmi-sjencanja-u-3d-prostoru.

SAŽETAK

Glavni problem proučavanja u ovom završnom radu temelji se na osnovnim principima sjenčanja u 3D prostoru. Upotrebljavaju se znanja iz područja linearne algebre, njihova implementacija u dvodimenzionalnom i trodimenzionalnom prostoru, te općenite transformacije i definicije nužne za stvaranje algoritama sjenčanja. Algoritmi sjenčanja direktno su ovisni o obliku, odnosno tijelu, na koji su primijenjeni, na njegov položaj u prostoru scene, te osvjetljenja koja upotpunjuju doživljaj realizma. Konstantno, Gouraudovo i Phongovo sjenčanje osnovni su principi sjenčanja u računalnoj grafici, tako reći, preteče svih zahtjevnijih algoritama sjenčanja. Problem je riješen uz pomoć OpenGL-a, GLSL-a, C++-a uz međudjelovanje s osnovnim principima prostora i transformacija linearne algebre. OpenGL omogućio je definiranje programa za sjenčanje vrhova i programa za sjenčanje fragmenata, definiranje oblika proučavanja i generalno postavljanje svjetla i scene. C++ programski jezik zaslužan je za implementiranje svojstava OpenGL-a, pa i njega samoga.

Ključne riječi: algoritam, sjenčanje, 3D, OpenGL, program za sjenčanje

ABSTRACT

SHADING ALGORITHMS IN 3D SPACE

The main problem studied in this final paper is based on the basic principles of shading in 3D space. Knowledge from the field of linear algebra, their implementation in two-dimensional and three-dimensional space, and general transformations and definitions necessary for creating shading algorithms are used. Shading algorithms are directly dependent on the shape, that is, the body, to which they are applied, on its position in the space of the scene, and the lighting that completes the experience of realism. Flat, Gouraud and Phong shading are the basic principles of shading in computer graphics, so to speak, the forerunners of all more demanding shading algorithms. The problem was solved with the help of OpenGL, GLSL, C++ with interaction of the basic space principles and transformations of linear algebra. OpenGL made it possible to define vertex shaders and fragment shaders, define studied shapes and generally set up lights and scenes. The C++ programming language is responsible for implementing the properties of OpenGL, including itself.

Keywords: algorithm, shading, 3D, OpenGL, shader