Assignment 4: Shading

Marie Elkjær Rødsgaard - dck495
Department of Computer Science

November 29, 2022

Contents

1 Introduktion

Denne opgave handler om at forstå og implementere phong shading af trekanter. I denne opgave vil vi gå over hvad phong shading er og hvad de tre led til at lave phong shading er: ambient, diffuse og specular refleksioner.

2 Teori

Phong shading er en simpel måde at modelere lys reflekteret på en overflade. Phong shading består som nævnt før af tre led: ambient, diffuse og specular refleksioner. Ambient refleksion er refleksioner der ikke er direkte på en flade men reflekteret over flere flader. Dette kan man også kaldet 'inter-refleksion'. Denne model vi bruger til phongs refleksion er en simpel model og tager ikke højde for at lys kan komme mange steder fra. Vi har en direkte belysning som vi modelerer. Vi kan skrive dette op som

$$R_{ambient} = I_a K_a; \quad k_a \in [0, 1]$$

Her er I_a en lysintensitet og k_a er en refleksionskoefficient mellem 0-1.

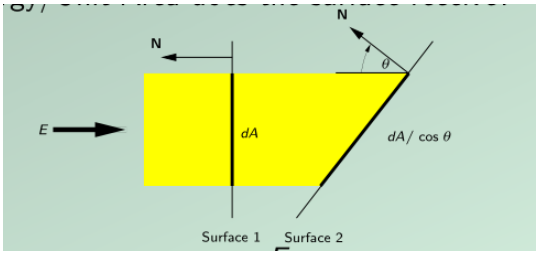

Figure 1: diffuse reflektion visualisation

Diffuse refleksion er når lys bliver reflekteret lige meget i alle retninger når det rammer en mat flade, dette kaldes også lambertian refleksion. Figure 1 ovenover viser en lysstråle med energy E som rammer en flade dA. dA er et meget lille areal, der står vinkelret på lyset som også kan ses på normalen, denne flade kaldes 'surface 1'. Vi har også en anden flade på figuren 'surface 2' som er surface 1 bare vippet og har θ vinkel som forskel. Arealet på den

vippede flade er større. Det vil sige at energien er mere intens på surface 1. Intensiteten for de to flader kan vi nu skrive op som

$$I = \frac{E}{dA}$$

$$I_\theta = \frac{E\cos\theta}{dA} = I\cos\theta$$

Når vi kigger på fladen fra øjet af, kan vi finde ud af hvor meget lys der så kommer fra fladen og op i øjet. Her har vi en vinkel Φ som er vinklen mellem normal vektoren og view vektoren. Lamberts lov siger at hvis vi har noget energi som bliver udsendt fra vores flade, så har vi en vis energimængde, der bliver udsendt vinkelret på fladen som svarer til normalens retning. Den kalder vi E_N. Hvis vi nu kigger i retning V og vil vide hvor meget lys, som bliver udsendt kan vi bruge Lamberts lov. Lamberts lov siger at lyset udsendt i retning V er lig lyset, der bliver udsendt i retning N ganget med $\cos\Phi$. Det kan skrives op som
Brightness: $E_V = E_N\cos\Phi$ (Lambert's lov)
Projekteret areal: $dA_V = dA\cos\Phi$
Intensitet: $I_V = \frac{E_V}{dA_V} = \frac{E_N\cos\Phi}{dA\cos\Phi} = \frac{E_N}{dA} = I_N$

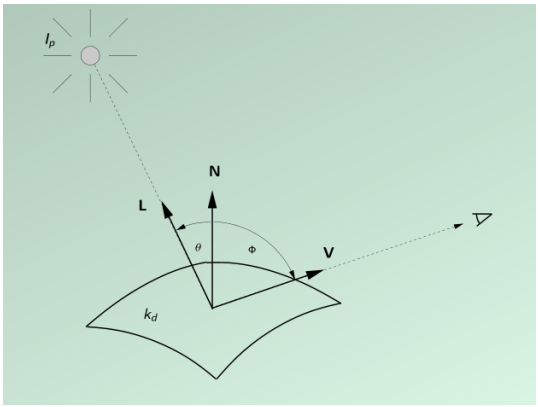

Figure 2: Lamberts lov fra øjet

Vores $R_{diffuse}$ kan vi nu skrive op som

$$R_{diffuse} = I_p k_d cos_\theta = I_p k_d(N \cdot Ł)$$

Figure 2 herover viser hvordan de forskellige variabler er illustreret. Hvis vi lader vores vektore her være enhedsvektore så gælder der at $\cos\theta$ svare til prikproduktet mellem N og L.

Specular refleksion kan ses når lys rammer skinnende overflader, så som et spejl. Man kan også tænke på et æble der har et punkt hvor det nærmest er hvidt fra lyset og resten af æblet har diffuse refleksion. Lyset, der er fremhævet vil nu flytte sig hvis man flytter sit hoved. Dette er fordi at det punkterne reflektere lys ulige på fladen. Hvis vi have et perfekt spejl ville lyset kun blive reflekteret i en retning R, som ville være L spejlet med vores normal N. Da vi ikke har et perfekt spejl bliver lyset reflekteret over et større sted som aftager jo længere væk man flytter sig med en vinkel α fra R. $R_{specular}$ kan skrives op som

$$I_{specular} = I_p k_S \cos^n \alpha = I_p k_s (R \cdot V)^n$$

Igen er vores vektorer enhedsvektore så vi bare kan finde prikproduktet. Vi skal nu finde ud af hvordan vi finder vores R vektor. Da R er spejlet af L vektoren betyder det at deres vinkler er ens. Vi kender vores L og N vektorer. Vi tager først vores L vektorer og projekterer hen på vores normalvektor N. Så kan vi finde $N\cos\theta$ som er en vektor på vores normalvektor.

$$S = N\cos\theta - L$$

$$R = N\cos\theta + S = N\cos\theta + N\cos\theta - L$$

$$R = 2N\cos\theta - L$$

$$R = 2N(N \cdot L) - L$$

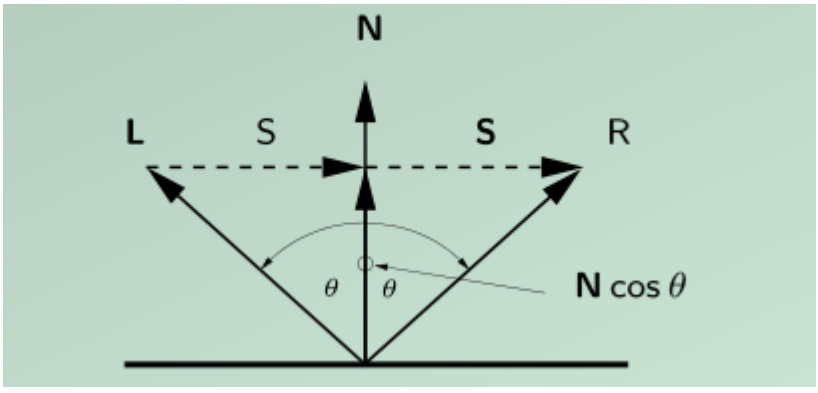

Figure 3: hvordan vi finder vores R vektor.

Nu har vi fundet ud af hvordan vi beregner de tre led som phong reflektionsmodellen består af. Vi adderer dem sammen og så får vi

$$I_{phong} = I_{ambient} + I_{diffuse} + I_{specular}$$

$$I_{phong} = I_a k_a + I_p k_d (N \cdot L) + I_p k_s (R \cdot V)^n$$

I en del af ledene er det nødvendigt at udregne et prikprodukt. Hvis dette prikprodukt skulle gå hen og blive negativt ville det betyde at vores lys er negativt. Vi vil nu, givet et punkt P med samme normal n, transformere P med en matrix M. Vores matrix M er en general homogen transformationsmatrix. Så transformeres vores punkt P således

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = P \mapsto \hat{P} = MP = \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \hat{w} \end{pmatrix}$$

Vi har nu vores punkt P som ligger i vores plan Π hvor $\Pi^T P = 0$. Hvis Punktet P er transformeret med matricen M så vi har $\hat{P} = MP$. Så skal vi også tranformere vores plan Π. Dette kan vi gøre ved nu at have en ukendt matrix Q hvor vi siger

$$\hat{\Pi} = Q\Pi$$

Vi kan nu finde ud af hvad Q skal være ved at bruge betingelsen for at $\hat{P}$ skal ligger i den planen $\hat{\Pi}$.

$$\hat{\Pi}^T \hat{P} = 0$$

$$(Q\Pi)^T (MP) = 0$$

$$\Pi^T Q M P = 0$$

Vi ved nu at $\Pi^T P = 0$. Så vi skal nu have at $Q^T M = I$. For at få enhedsmatricen skal $Q = (M^{-1})^T$. Så har vi at $\hat{P}$ ligger i $\hat{\Pi}$. Og hvis normalen n også skal tranformeres gøres det også med $(M^{-1})^T$ ligesom for Π.

3 Implementation

For at implementere phong shading har vi kigget på to filer `vertexttransform.vert` og `phong.frag`. I vores `vertexttransform.vert` har vi vores `CTM` matrix som er vores `current transformation matrix` fra sidste opgave. Den variabel bruger vi til at finde vores nye positioner. Dette gør vi ved at sige gl_Position = CTM * vec4(Vertex, 1.0f);.

I `phong.frag` programmet laver vi vores phong refleksion. `phong.frag` har variablerne `AmbientLightColor` `LightPosition`, `LightColor`, `EyePosition`, `AmbientColor, DiffuseColor, SpecularColor,` og `Shininess`. Disse variabler bruger vi så til at finde vores ambient, diffuse og specular refleksioner. Vi finder først ambient og ligger det til vores color. Ambient finder vi som nævnt i teori afsnittet, med en `AmbientLightColor` ganget med `AmbientColor`. For at finde diffuse og specular skal vi bruge vektorene, N, L, V og R. Alle vektorene finder vi og derefter normalisere vi dem. Nu kan vi så finde diffuse og specular med formlerne nævnt i teori afsnittet. Inden vi regner dem ud har de hver et tjek over at deres prikprodukt er positivt. Hvis de ikke er positive ligger vi dem ikke til color. Til sidst sætter vi `FragColor = vec4(color, 1.0f);` så den bliver tegnet.

I hovedprogrammet kalder nu vores to programmer `vertexttransform.vert` og `phong.frag` ved at lave et gpu program. Derefter gemmes de to i variablen `triangleshaderID` som har lavet et shader program af gpu programmet. `triangleshaderID` bruger vi så til at få alle locationerne på variablerne vi skal bruge i vores `vertexttransform.vert` og `phong.frag`. I et while loop tildeler vi så værdier til vores uniforme variabler, som også vil opdateres der. Efter det tegner vi så vores trekanter med shading i linjen `glDrawArrays(GL_TRIANGLES, 0, NVertices);`.

4 Test

Det implementerede program for at lave phong shading bruger de tre led ambient, diffuse og specular. Hver af de led kan ses enkeltvis på figure 4. På figure 5 kan vi så se phong shading på midten af en trekant og det ser rigtigt ud. Figure 6 viser to andre trekanter hvor der også er phong shading på og det ser stadig rigtigt ud. Der er cases hvor lyset ville være bag ved trekanten

så ville vi ikke kunne se vores diffuse og specular på billede.

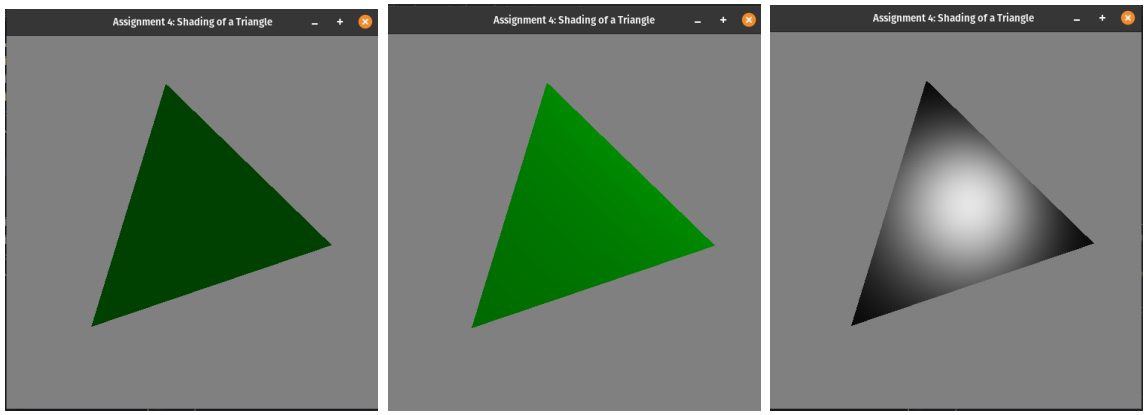

Figure 4: Her ses ambient, diffuse og specular reflektioner hver for sig

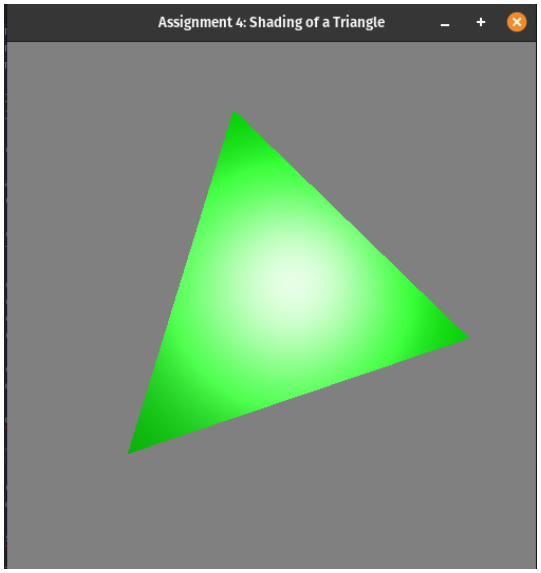

Figure 5: Phong shading i midten af en trekant

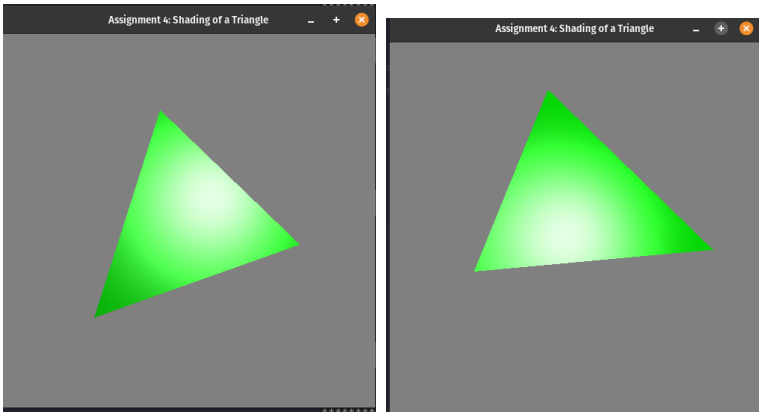

Figure 6: Andre trekanter med phong shading

5 Konklusion

Vi har nu lavet en implementation af phong shading efter først at have gennemgået hvordan de tre led af phongs reflektionsmodel og hvordan de sammen kan laves til phong shading. Vi har lavet test for at tjekke at programmet virker på forskellige trekanter og om de enkelte led gør som ønsket. Efter testene kan vi se at programmet opfylder vores ønskede krav.

References