Shaderprogrammering

Uppgift 5

Hårdvarushader
Tangetrymd och parallax

Johan Lavén
d15johla@student.his.se

Introduktion

Målet med denna shader är att efterlikna en textur av murbruk inklusive reflektioner i murbruket som sträv yta där stenarna befinner sig. Tekannan ska ha en diffus belysning baserat på ljuskälla, det ska finnas reflektioner i murbruket baserat på en fördefinierad textur samt att det ska finnas en sträv texturliknande effekt baserat på en normalmap.

Designval och implementation

För att få så hög kvallité som möjligt på belysning och texturer så valde jag att skriva största delen av shadern i fragmentprogrammet (*Pixelshadern*).

Jag valde att lägga reflektionen hos tekannan baserat på två faktorer, den första är en fresnelreflektion och det andra är genom en reliefmap som talar om vart murbruket ska reflektera som mest. Fresnelreflektionen är menad att ge en övergripande reflektion av scenens belysning i utkanterna av objektet från kamerans perspektiv, det vill säga att om kameravektorn pekar rakt mot modellen så ska speglingar synas vid sidorna och över/underkant av modellen. Reflektionen som baseras på reliefmapen fungerar på så sätt att en färdig textur talar om vart kannans textur ska lysa upp, i detta fall samspelar den texturen med resterande murbrukstexturer.

Figurer:
Tekannan illustrerad med Fresnelreflektion(*Höger*) och utan (*vänster*)

Sättet som jag valde för att slå ihop de olika typerna av reflektionseffekter var genom addition. Då valet gjordes att addera ihop de olika reflektionerna så fanns det de delar av resultatet som blev onaturligt ljust, därför delades dessutom slutresultatet på två för att ge en lugnare intensitet.

```
float fresnelF = smoothstep(fresnelStart, fresnelEnd, dot(E, N));
reflection *= reflection;
```

Figur: Kod som skapar fresenlreflektionen och slår ihop den med den tidigare beräknade reflektionen.

Tekannan ska även ha en reflektion baserat på en *cubemap* som illustrerar scenen tekannan befinner sig i. Hur mycket som ska synas av den reflekterade cubemapen

```
vec3 reflection = textureCube(cubeMap, R).xyz;
reflection = mix(reflection*texture2D(reliefMap, newCoords).xyz, reflection,
reflectionInMurskarv);
```

Figur: Ovan kod visar sammanslagning av reflektionen från cubemapen och hur mycket den ska begränsas till insidan av murbrukstexturen.

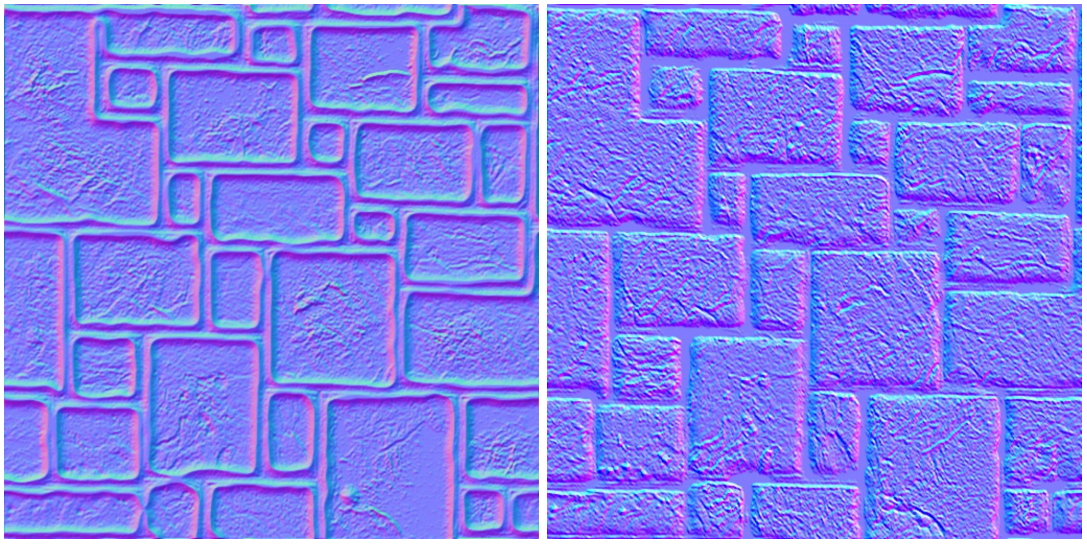
Figur: Illustration av cubemapsreflektion på objektet när variabelns som kontrollerar texturen för objektet dragits ner.

För och nackdelar med reflektioner

Skillnaden på resultat när det gäller en reflektion baserat på cubemaps kontra en vanlig spekulär belysning är detaljrikedomen. Ett objekt som ska ha detaljer i form av reflektioner kommer att behöva göra det på två sätt, Antingen genom en fördefinierad cubemap som innehåller information om vad som ska reflekteras på ytan, detta anses vara en relativt billig form av reflektion då den inte behöver beräkna omgivningen i realtid. Med denna metod framstår dock en del problem, Bland annat ifall objektet med reflektionerna inte är statiskt i scenen, För oavsett om du flyttar objektet till en annan position så kommer reflektionerna fortfarande att spegla den tidigare fördefinierade miljön baserat på cubemapen. Ifall effekten av reflektion är efterfrågat men att det ska uppdateras finns det en metod som kallas för *Light probes*. I boken *OpenGL shading Language(Rost, Licea-Kane and Ginsburg, 2013)* beskrivs en Light Probe som en apparat eller ett system som fångar en högdynamisk bildrepresentation av ljuset i alla riktningar. Detta system kan fungera i realtid men kostar mycket att beräkna i jämförelse med traditionella cubemaps och andra klassiska belysningsmetoder, av den anledningen så används därför denna metod sparsamt i praktiken.

Bumpmap analys och jämförelse

I det givna arkivet fanns två givna normalmaps, även känt som bumpmaps, för att ge texturen lite djup. Denna metod kallas *Bump Mapping*. Bump mapping är en renderingsteknik som simulerar utseendet av bucklor, skrynklor eller andra ojämnheter genom att förskjuta ytnormalen innan beräkning av ljuseffekter. Båda bumpmap-texturerna beskriver hur normalen ska förskjutas på samma ställen men med olika sätt.



Figurer: De två givna normalmapsen sida vid sida
(Vänster: vass, Höger: mjuk)

Figurer: Resultatet när normalen beräknas på de olika normalmap-texturerna.

På exemplet med tekannan som modell så anser jag att den lite mjukare bumpmapen (den till höger av figurerna) ger en mer realistisk effekt. Detta då murbruket mellan mosaiken ser mer realistisk ut med de mjukare övergångarna. Om jag i det vänstra exemplet skulle begränsa upplysningen och reflektionerna i murbruket så tror jag att effekten skulle kunna se någorlunda bättre ut.

Referenser

Rost, R., Licea-Kane, B. and Ginsburg, D. (2013). *OpenGL shading language*. 3rd ed. Upper Saddle River, NJ [u.a.]: Addison-Wesley, pp.693-694.