Shaderprogrammering

Uppgift 7

Hårdvarushader

Screen Space

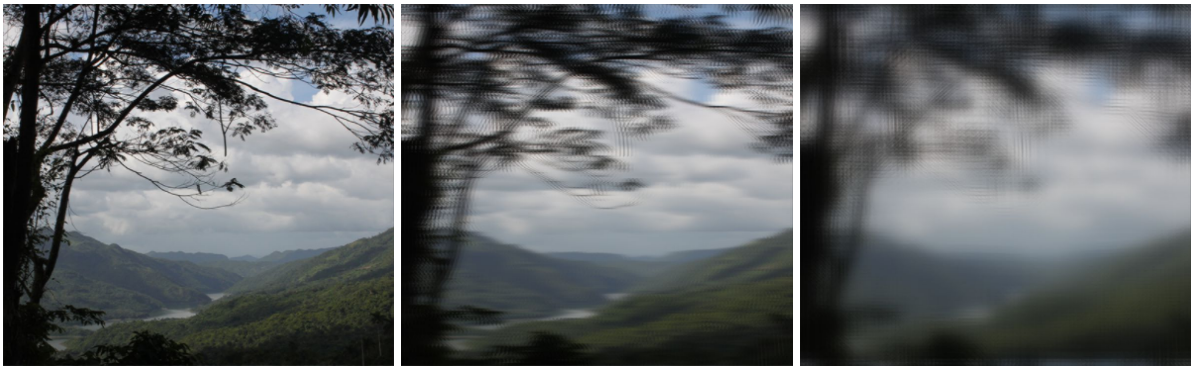
Johan Lavén
d15johla@student.his.se

Introduktion

Målet med *screen space shading* är att skapa estetiska effekter som kan jämföras med kameraeffekter. Tanken är att ingen av dessa modifikationer görs i form av belysning eller skuggning i en 3D-scen. Utan att alla effekter appliceras direkt på pixlarna av slutresultatet. Dessa typer av shaders är vanliga i applikationer som till exempel instagram och snapchat.

Designval och implementation

Genom att arbeta med fragmentshadern i tre olika pass så möjliggörs tekniken att stapla effekter ovanpå varandra på ett sätt som påminner om lager i ett bildredigeringsprogram. Det innebär att de effekter som beräknas i pass 2 läggs ovanpå resultatet av samtliga effekter från pass 1 och så vidare. Till exempel som blur- och gloweffekter som vanligtvis beräknas genom att ta samplingsvärden från runtliggande pixlar i alla riktningar. Men om samma effekt önskas uppnå kan samplingarna göras i två pass där det sköter samplingarna i x- och y-led separat istället för en samplingsmatris som som gör alla samplingar i båda leden samtidigt. Stora O-notationen för denna algoritm i två pass är O(2n) jämfört mot O(n^2) hos matrissamplingen.



Figur: Orginalbild till vänster jämfört mot bild med applicerad blureffekt i x-led och bild med sampling i båda riktningar..

Denna effekt uppnås med en filter-kernel som innehåller 3-dimensionella vektorer där z-elementet representerar vikten från varje samplingspunkt. Samma samplingsmetod går används vid glow-effekter, där enda skillnaden är rgb-värdena från varje sampling multipliceras med sig själva.

```
kernel[0] = vec3( 0.0, 0.0, 0.266);
kernel[1] = vec3( 0.0, 1.0, 0.213);
…
kernel[10]= vec3( 0.0,-5.0, 0.001);
vec3 blurCol;
for(int i = 0; i < 11; i++){
   blurCol += texture2D(Pass1, texCoord + blurPow*kernel[i].xy).xyz * kernel[i].z;
}
```

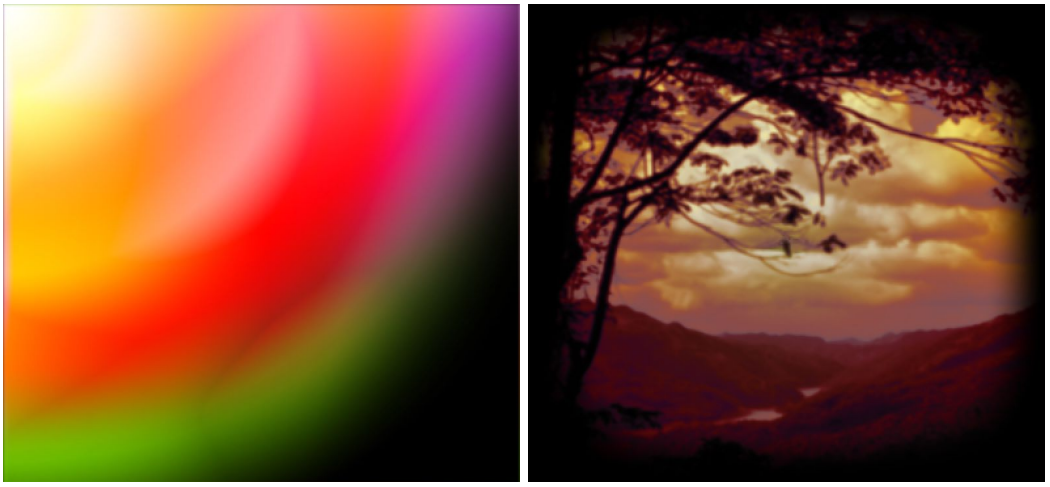
Figur: Kodrepresentation av blur-effekten i y-led.

I shadern så skapas även några klassiska grafiska effekter där två av dessa är scanlines och en vinjett. Båda dessa effekter har uppgiften att mörka ut delar av skärmen. Scanline-effekten

simulerar effekten av en gammal CRT-tv som skannar över bildskärmen och vignette av en gammaldags bild med rundade utkanter.

Figurer: Vänster: Applicerad Vignette
Höger: Scanlines som rör sig uppåt på skärmen.

Bilden ska även kunna färgjusteras med hjälp av bland annat ett colorizer-filter, ett sådant filter innebär att en separat textur med fördefinierade färger som ska multipliceras in i bilden baserat på dess x- och y-kordinater.



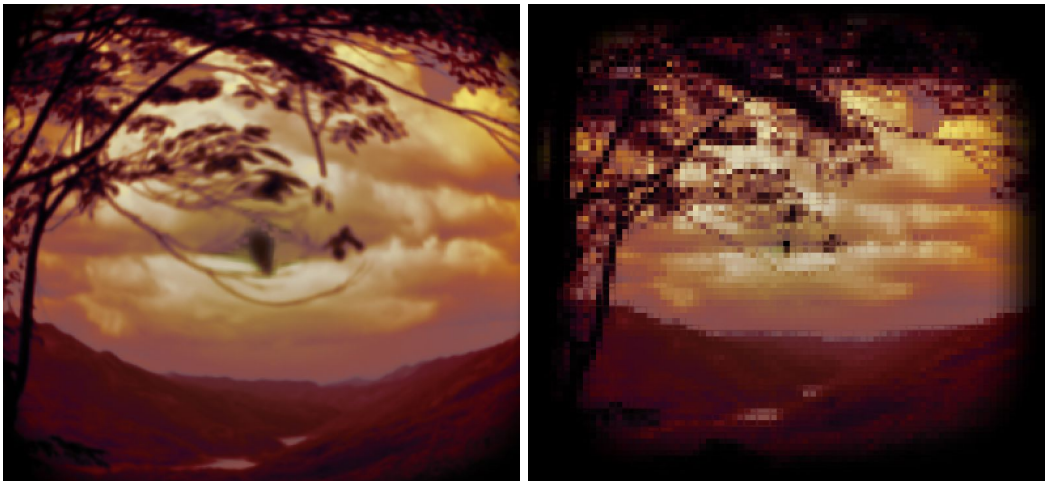
Figurer: Colorizertexturen som används för att bestämma färgen. Samt resultatet till höger.

```
vec2 texC;
texC.x=1.0-clamp(blurCol.x,0.01,0.99);
texC.y=1.0-(distance(vec2(0.5,0.5),texCoord )*1.0) * ColorizerDist;
outCol *= texture2D(colorizer, texC).xyz;
```

Figur: Sampling samt sammanslagning av colorizern och utdatans färgvärden.

I det sista passet läggs följande effekter ovanpå resultatet från det tidigare passet:

- Fisheye-effekt (Lense *Buldge*)
- Pixelering
- Sepiatoning
- Animerade bruseffekter



Figurer: Vänster: Lense Buldge, Höger: Pixelering

Figurer: Vänster: Sepiatoning, Höger: animerat Skärmburs

Det sista som ska göras är att det ska finnas en *lens flare* i bilden. Målet är att få samtliga effekter i pass 3 att också påverka denna lens flare så därför gjordes beslutet att mixa in den i pass 2. efter att blureffekten beräknats. Detta ger då en illusion att landskapet är i bakgrunden och att lens flaren hamnar direkt på linsen i fokus.

Här gjordes även beslutet att det ska finnas två sätt att applicera effekten, och att det ska vara enkelt att välja mellan dem. De två sammanslagningsoperationerna som används för att implementera effekten är:

1. Snittvärdet från de två texturerna
2. Differensen mellan de två texturerna

```
vec4 lensFlareTex = texture2D(lensFlare, texCoord);
vec4 colA = vec4(blurCol,1.0);
vec4 colB = vec4(blurCol,1.0);
colA.xyz = (lensFlareTex.xyz + blurCol)/2; //Avrage
colB.xyz = abs(blurCol.xyz-lensFlareTex.xyz); //Diff
blurCol = mix(colA, colB, mergeMecanic).xyz;
```

Figur: Kod för sammanslagning och val mellan olika metoder.

Figurer: Vänster visar resultatet av sammanslagningsmetoden snitt och höger visar resultatet av differentiell sammanslagning.

För och nackdelar med flerpassrendering

En fördel med att använda flerpassrendering är att effekter kan på ett smidigt sätt appliceras uppå varandra. När ett Pass renderats klart så läggs utdatan från det passet i en framebuffer. Den datan går att plocka upp av nästkommande pass för att använda som en textur. På så vis kan effekter som är enklast att applicera under samplingsstadet smidigt utföras. Flerpassrendering låser även upp möjligheten till andra renderingsmetoder jämfört mot den klassiska "*forward rendering*". En av dessa metoder är den så kallade *deferred rendering* som först beräknar vilken geometri som ska synas i scenen innan andra effekter som belysning beräknas.

Referenser

Rost, R., Licea-Kane, B. and Ginsburg, D. (2013). *OpenGL shading language*. 3rd ed. Upper Saddle River, NJ [u.a.]: Addison-Wesley, pp.258-261.