HÖGSKOLAN
I SKÖVDE

Shaderprogrammering

Uppgift 2

Mjukvarurendering Galax

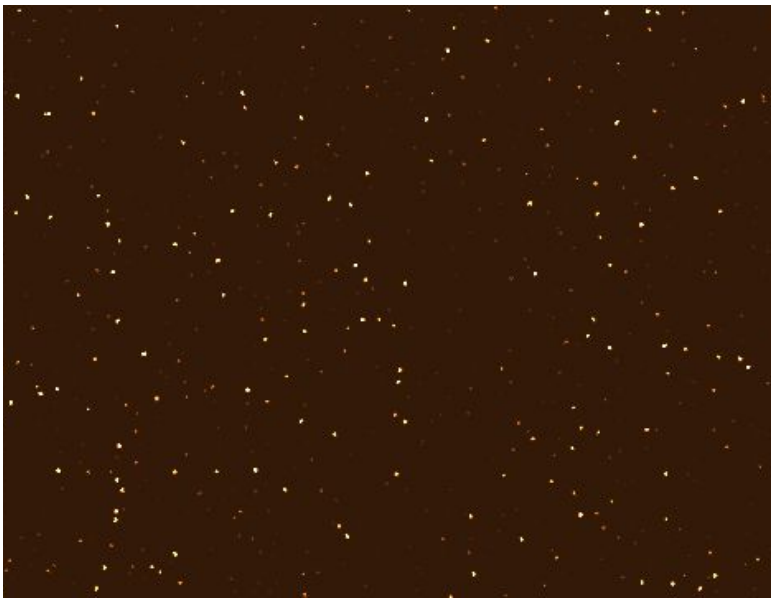
Johan Lavén
d15johla@student.his.se

Deluppgift 1:

I uppgift 1 var målet att skapa en stjärnhimmel med kontrast mot bakgrunden genom att använda cellbrus. Cellbrus är en en funktion som delar upp geometriska primitiver i rutmönster. Inom vardera av dessa rutmönster så returneras ett värde baserat på en pseudo-slumpad position. Med dessa värden så baserar vi färgerna hos vardera pixel med hjälp av en smoothstep för att göra övergångarna mer abrupta än vad de varit ifall det direkta värdet från funktionen används. I och med att tröskelvärdena baserat på cellbrus-funktionen sätts så nära takvärdet kan vi ge en effekt av stjärnor utspridda över en himmel.

```
voronoi_f1f2_2d (ss, tt, f1, 0,0,0,0,0); // f1 recives a value between 0 and 1

…

f1 = smoothstep(0.01, 0.05, f1);
```

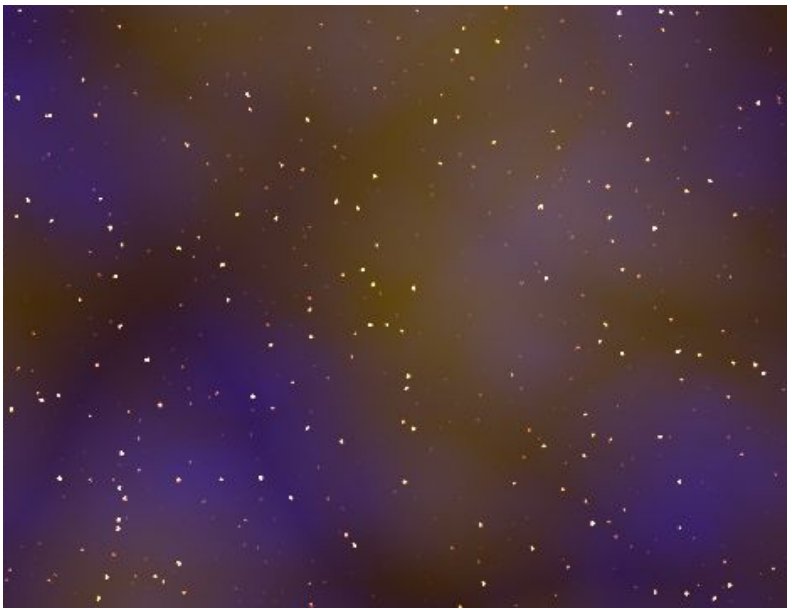
Kod som illustrerar användandet av mod samt step.



Stjärnhimmel genererad med Cellbrus.

Deluppgift 2

För att inte stjärnhimlen ska kännas så tråkig och monoton ska procedurellt genererade moln appliceras ovanpå stjärnorna för att ge en effekt av en eller flera nebulosor. Detta med hjälp av en av två funktioner som kallas fBm() eller turbulence() som beräknar en textur med hjälp av fraktaler som är självliknande eller självupprepande mönster, ett så kallat *Fraktalbrus* eller "fractal noise". Värdet returnerat fraktalbruset används för att multiplicera in en vald färg för molnet, denna metod upprepas med olika parametervärden för att skapa flertalet olika nebulosor.



Ovan figur illustrerar tillägget av de kosmiska molnen ovanpå stjärnorna

```
float cloud0 = turbulence(pp/100, 6, 1.6, 0.5);

float cloud1 = turbulence((pp+37)/100, 8, 1.6, 0.2);

color blueCloud = blue*cloud0;

color yellowCloud = yellow * cloud1 * 0.3;
```

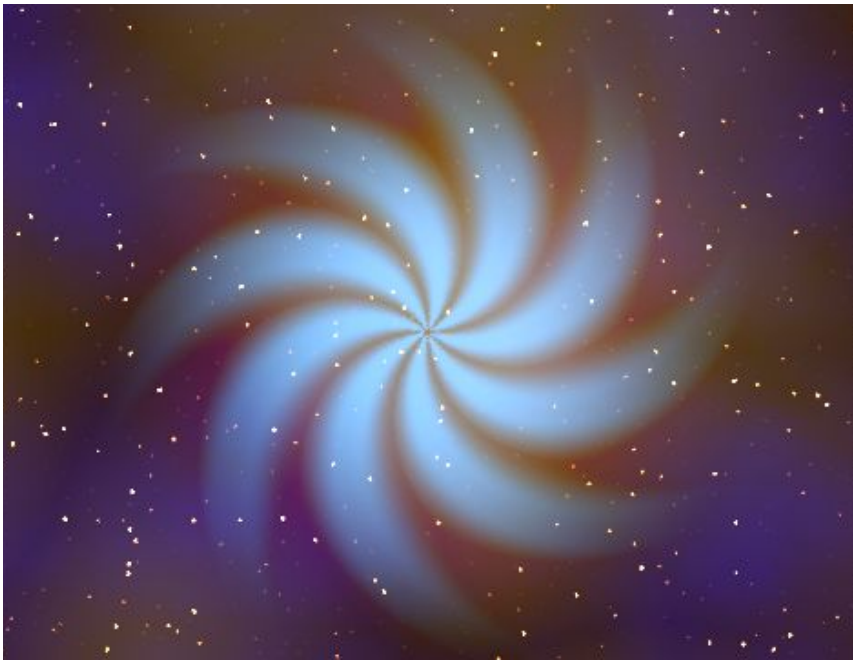
Kod för användande av en fraktalbrusfunktion.

Deluppgift 3

Framför stjärnhimlen skulle en galax procedurellt genereras. Detta med hjälp av att nyttja ett polärt ett polärt koordinatsystem i syfte av att skapa roterande armar hos en galax. Utöver själva spiralmönstret så använda smothstep()-funktionen för att på ett mjukt sätt tona in och ut armarna av galaxen samt att ytterligare en smoothstep används för att tona ut armarna baserat på distansen från centrum.

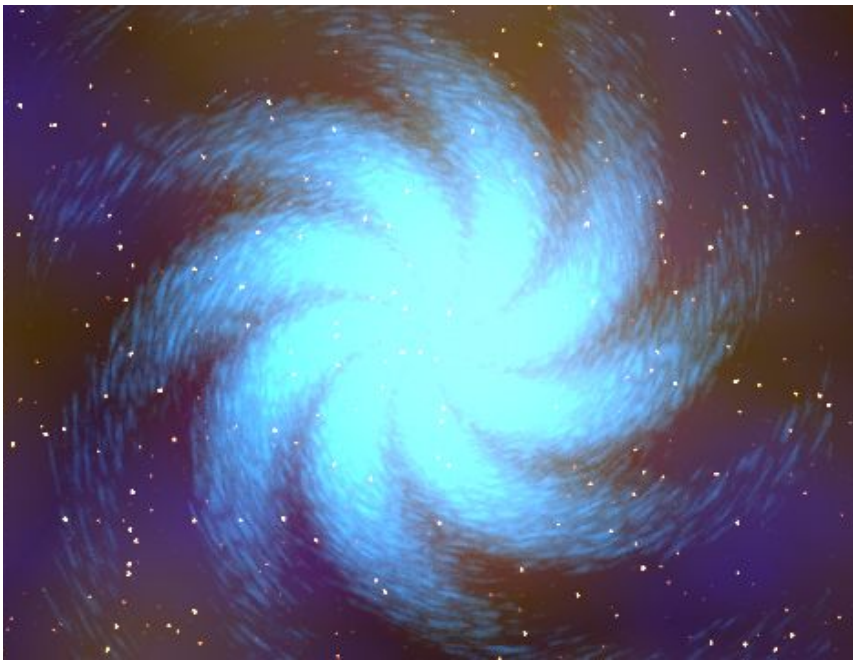
```
float rays = smoothstep(0.0, 0.7,mod(atan(uSpin,vSpin)*2+dist,1.57));

float rays2 = smoothstep(1.0, 1.7, mod(atan(uSpin,vSpin)*2+dist, 1.57));

rays = rays-rays2;

float distTone = smoothstep(0.5, 3.0, dist);

rays = rays - distTone;
```

Kod för att tona in och ut i relation till rotation



Figur illustrerar vad ovan kod resulterar i.

Nästa steg i att skapa en realistisk galax är att skapa en ljusare centrum av galaxen samt att skapa brus innuti armarna för att ge effekten av att galaxen i helhet är uppbyggd av mindre solsystem och som roterar runt ett lokalt centrum. Metoden för att skapa en den efterlängtade effekten bygger på samma funktioner som nebulosorna i stjärnhimlen, dvs turbolense(). Se nedan bild för att se effekten av dessa tillägg.



Bilden ovan illustrerar hur fraktalbruset samt det ljusare centrumet ger ett mer realistiskt utseende på galaxen i helhet.

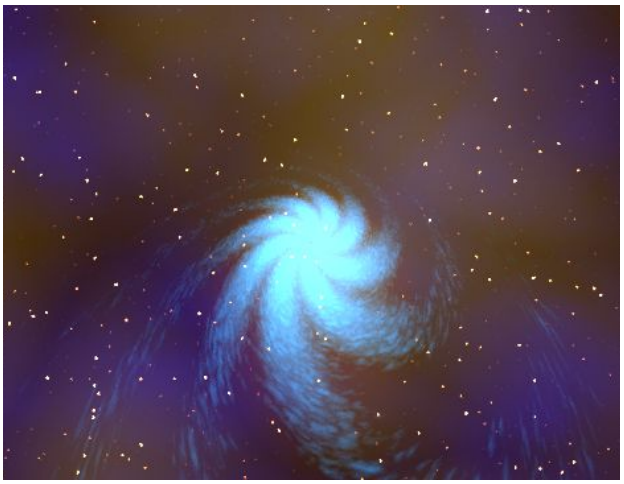
Det sista som saknas är att skapa ett djupperspektiv på galaxen så att det ger en illusion av att galaxen ligger på ett plan i rymden som inte är riktat rakt mot åskådaren. Detta ges genom att förvränga u och v.

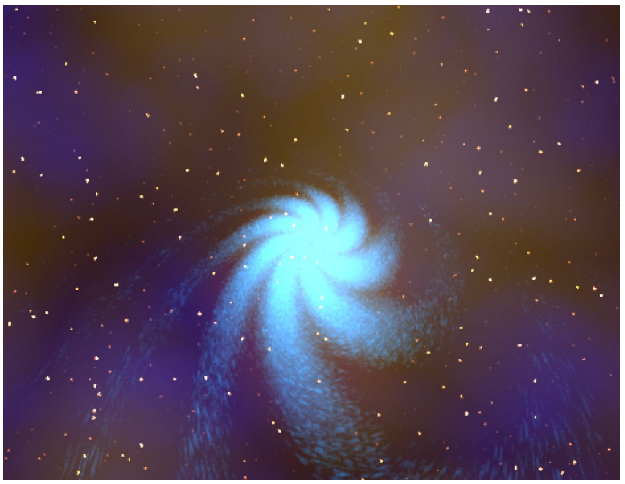
Bilden visar på effekten av att modifiera perspektivet av galaxen.

```
float vc=(v-0.1)*1.5;

float uSpin = ((u-0.5)*spinMultiplier)/(1.0-vc);

float vSpin = ((v-0.5)*spinMultiplier)/(1.0-vc);
```

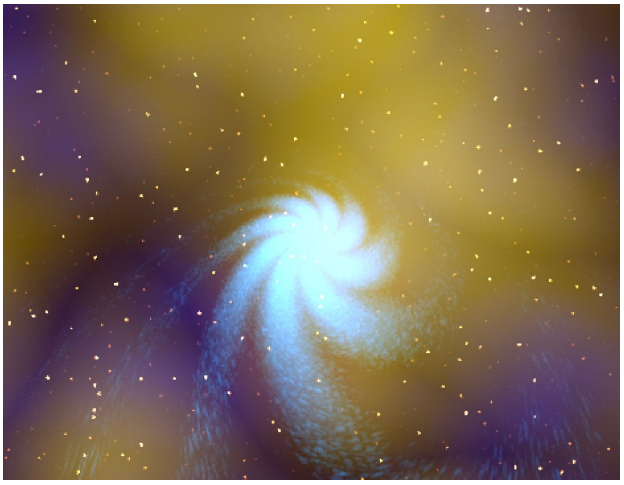
Koden som används för att modifiera perspektivet.

Modifiering av slutparametrar





Bilden till vänster visar på slutresultatet med alla de parametrar jag valt.
Bilden till höger illustrerar effekten av dubblerar frekvensen för interna moln i galaxen.



Bilden till vänster visar på när jag höjer färgintensiteten hos den gula nebulosan. Notera att molnens form och frekvens står sig.
Bilden till höger visar resultatet när jag multiplicerar alla invärden hos fraktalbrus-funktionen med 2. Detta ger högre gain, octaves och lacunarity.

Diskussion

Vikningseffekter(*aliasing*) är ett fenomen som uppstår när insamlandet av mätdata(*sampling*) har lägre frekvens än det som ska mätas, även kallat Nyquistteoremet efter forskaren Harry Nyqvist. Fenomenet uppenbarar sig i grafik på så sätt att kanter med hög kontrast får artefakter på grund av dessa mätvärden, ofta kallat "*jaggies*" i stillbilder eller "*the crawlies"* i animerade bilder *Real time rendering* (2008:s117). I boken *Advanced RenderMan, Creating CGI for Motion Pictures* (1999:s263-280) beskrivs just vikningseffekter som en dålig rekonstruktion av kamerans ursprungsfunktion. Den stora nackdelen gällande vikningseffekter i samband med helt procedurella shaders är att kontrollen för just kanthantering kräver mer arbete jämfört mot grafik skapat av en konstnär. Man kan dock gå runt problemet genom att ta in mer mätdata *n* från den ursprungliga funktionen, känt som *super sampling*. Det är en lösning som fungerar till viss mån då kostnaden i tid och datorkraft multipliceras med *n* nya mätpunkter och felvärdet minskar bara med kvadratroten av *n.* Av den anledningen har mer effektiva former av skydd mot vikningseffekter (*antialiasing*) utvecklats. Några av dessa är kända som *Analytical Antialiasing* och *Frequency Clamping Antialiasing.*

Referenser

Apodaca, A.A. & Gritz, L., 1999. *Advanced RenderMan: creating CGI for motion pictures*, San Francisco: Kaufmann.

Möller, T., Haines, E. & Hoffman, N.,2008. *Real-time rendering third edition*, Wellesley, MA: Peters.