



HÖGSKOLAN
I SKÖVDE

Shaderprogrammering

Uppgift 3

Mjukvarushader
Bergslandskap RenderMan

Johan Lavén
d15johla@student.his.se

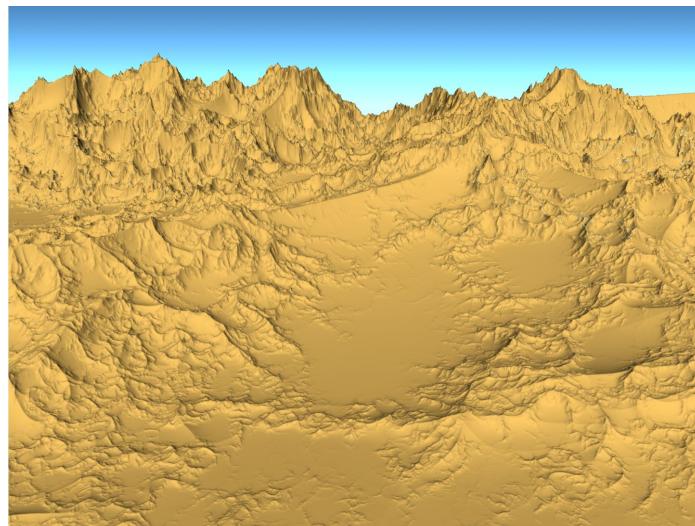
Introduktion	3
Designval och implementation	3
Resultat och Parametervariation	8
Referenser	10

Introduktion

Uppgift 3 gick ut på att med hjälp av en procedurell mjukvarushader skapa ett bergslandskap. Detta gjordes i RenderMan RSL och renderades med hjälp av Aqsis Renderer. Landskapet renderas genom två shaders, dessa är 1. bergslandskapet med sjö och färsgättning 2. himmel med moln och sol.

Designval och implementation

Det material som tillhandahölls för att påbörja uppgiften var en funktion som med hjälp av fraktalbrus skapar en formad ofärgad berggrund. Med hjälp av denna exempelkod och vår tidigare inlärda kunskap om RSL och RenderMan så ska vi manipulera och skriva om den fulla shadern för att skapa något som ser ut som ett bergslandskap med sjöar eller floder, snö, sten, gräsmark och slutligen ett lager dimma eller dis över berget.



Figur: Det resultat som det givna kodstycket generar.

```

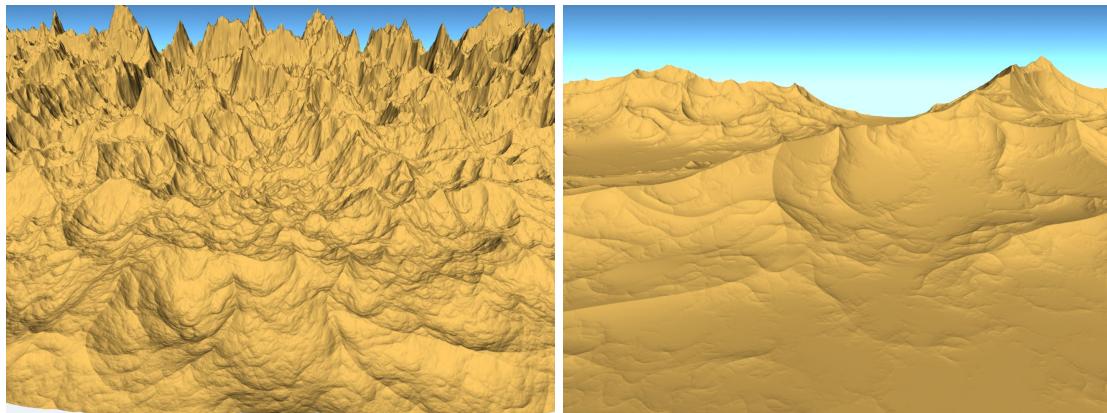
float RidgedMultifractal(point p; uniform float octaves, lacunarity, gain, H, sharpness,
threshold)
{
    float result, signal, weight, i, exponent;
    varying point PP=p;

    for( i=0; i<octaves; i += 1 ) {
        if ( i == 0 ) {
            signal = snoise( PP );
            if ( signal < 0.0 ) signal = -signal;
            signal = gain - signal;
            signal = pow( signal, sharpness );
            result = signal;
            weight = 1.0;
        }else{
            exponent = pow( lacunarity, (-i*H) );
            PP = PP * lacunarity;
            weight = signal * threshold;
            weight = clamp(weight,0,1);
            signal = snoise( PP );
            signal = abs(signal);
            signal = gain - signal;
            signal = pow( signal, sharpness );
            signal *= weight;
            result += signal * exponent;
        }
    }
    return(result);
}

```

Figur: Den givna funktionen för att skapa bergslandskapet procedurellt med fraktaler.

Det är genom att manipulera invärdena till *RidgedMultifractal()*-funktionen som olika former och karaktärer till bergskedjan kan ges. För att få parametrar som såg bra ut ihop så experimenteras det mycket. Målet var att få något som såg ut som en slät bergskedja med inte så vassa klippor, detta för att få en naturlig look på hur bekanta nordiska fjäll ser ut.



Figurer: Väster: Experimenterade parametrar som inte var vad som eftersöktes.
Höger: Det resultat som valdes i slutändan.

```

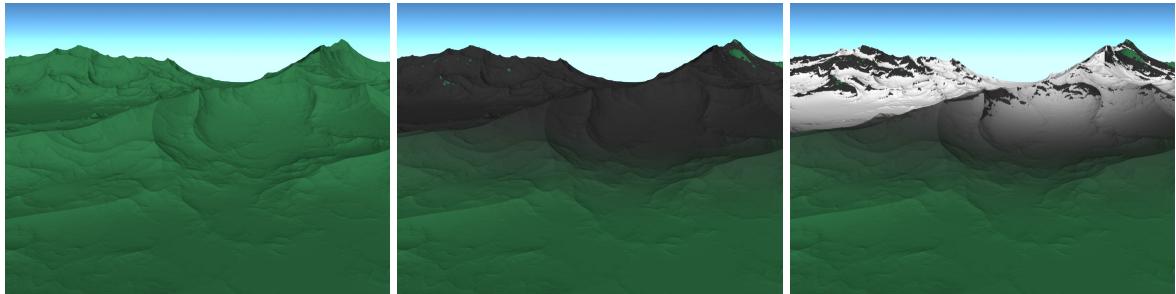
float magnitude = (RidgedMultifractal((P+32)*0.002, 7, 2.5, 0.9, 1.2, 5.8)*120);

```

Figur: Valda inparametrar för generering av terräng.

Det första målet var att färglägga alla delar av berget för sig, dessa inkluderar:

- Grön färg som representerar skog och gräs med brus för att ge lite textur, bruset kan appliceras i form av annan färg eller displacement, i detta fall varierar färgerna.
- Mörk sten på branter av bergstopparna.
- Vit snö en bit upp på berget, denna ska brytas mot tidigare nämnd mörk färg när lutningen blir för brant.
- Ljus beige eller guldig färg som ska föreställa sand där berget bryter mot vattnet.
- Blått vatten med brus i form av färg eller displacement. Här kommer vattnet få brus i form av färg för att efterlikna skummande vågor.
- En grå dimma eller dis som blir mer tydlig ju längre bort från kameran pixeln är.



Figurer: Tre olika stadier av färgläggning. *Vänster*: gräsets grundfärg. *Mitten*: Stenens grundfärg. *Höger*: Snö som ritas ut istället för bergets egna färg baserat på normalens lutning.

```

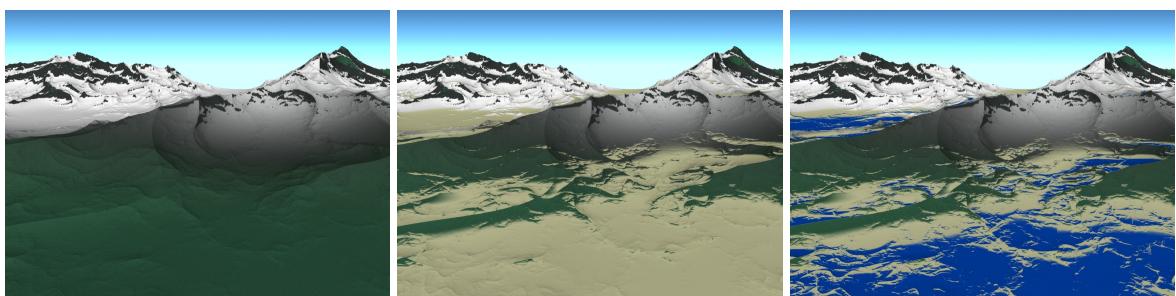
float peaksel = ycomp(normalize(N));
peaksel = step(0.2, peaksel);
float trans = smoothstep(0.6, 0.9, v);
color mountainPeakColor = (0.2,0.2,0.2);
outCol = mix(outCol, mountainPeakColor, peaksel*(1-trans));

//Snow on mountain
float snowsel = ycomp(normalize(N));
snowsel = step(0.5, snowsel);
float transS = smoothstep(0.3, 0.7, v);
color snowC = (1.0,1.0,1.0)

outCol = mix(outCol, snowC, snowsel*(1-transS));

```

Figur: Metod för att rita ut de grå områdena och hur snö appliceras med relation till normalens vinkel. Samma metoder används senare för sanden och vattnet.



Figurer: Fortsättning av färgläggning. *Vänster*: Vanligt brus appliceras mot det gröna för att se ut som gräs eller skog, detta genom RSL noise(). *Mitten*: Sand som ska ritas över med vattnet till stor del. *Höger*: Vatten är tillagd men ser dåligt ut.

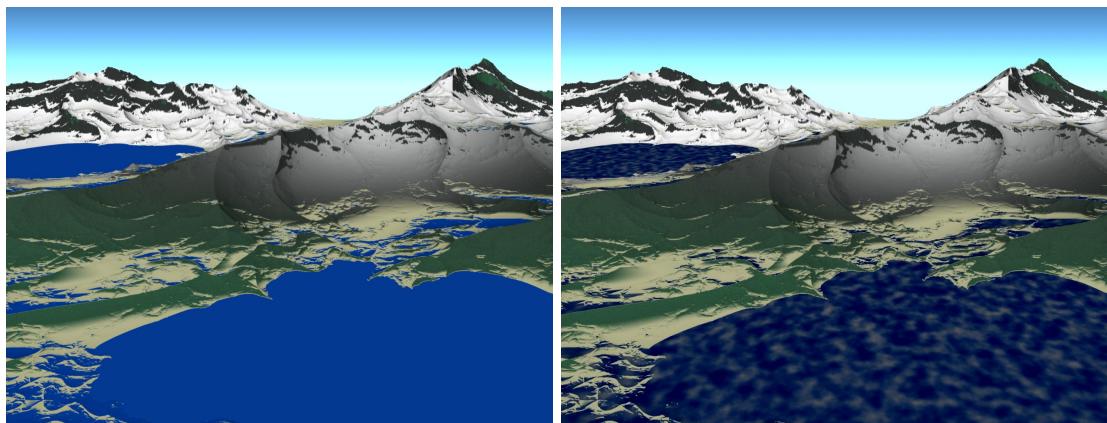
```

float grassNoise = noise(P);
outCol = mix(outCol, grassColor * grassNoise * grassFreq, outCol);

```

Figur: Hur bruset multiplikativt läggs på till gräsets grundfärg.

För att åtgärda problemet med vattnet så clamps magnituden som returneras från RidgedMultifractal(). Detta resulterar i en helt slät vattenytan. Då lösningen att strypa höjden ger en onaturligt slät enfärgad yta så appliceras sedan vågeffekter på vattnet med hjälp av ett fraktalbrus som påverkar färgen.



Figurer: Vänster: Vatten på slät yta. Höger: Vågor i form av fraktalbrus på färgen.

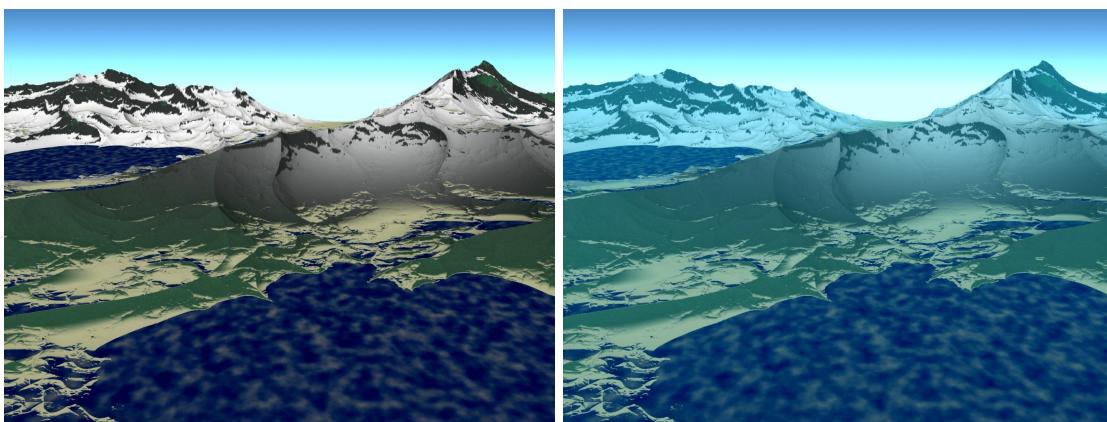
```

magnitude = clamp(magnitude, waterLevel, 1000);

```

Figur: Begränsar bergslandskapet så att vattnet ska bli platt. *waterLevel* är en float.

Det sista som ska appliceras på själva berget innan himlen målas om är en dis. Denna dimma ska påverka topparna av berget och ska baseras på djupvektorn från kameran till punkten i fråga. Diset är skapat efter en metod som nämns i boken *Advanced RenderMan, Creating CGI for Motion Pictures* (1999:s315).



Figur: jämförelse på landskapet med och utan dis.

```

float d = 1-exp(-length(I)/dist);
...
Ci = mix(Ci, color(0,0.8,1) ,d);
Oi = mix(Oi, color(1,1,1) , d);

```

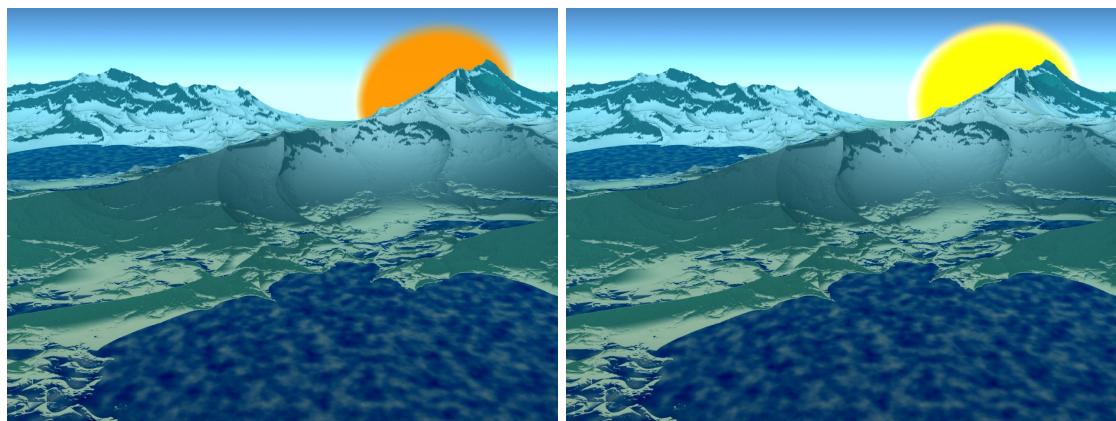
Figur: Implementation av bris. *dist* är en float med ett högt värde på 1400 i föregående exempel.

Bergslandskapet ansågs vid detta skede klart och arbetet på shadern som ritar ut himlen procedurellt skulle skapas. Målet med himlen var att efterlikna en soluppgång med en sol som stiger och tunna vintermoln som ska passa bra med diset.

Därför är saker som ska finnas i shadern:

- Procedurellt genererade moln.
- En mindre men färgstark sol.
- En passande ljus färg på himlen som ska skina igenom molnen.

Det första som skapades var solen och för att betona att det är en soluppgång så ritas den ut till stor del dold av bergslandskapet. Solen ritas även ut till höger för att betona en soluppgång i öst.



Figurer: Vänster: Solen utritad på önskad plats. Höger: Samma sol med lite extra glow.

```

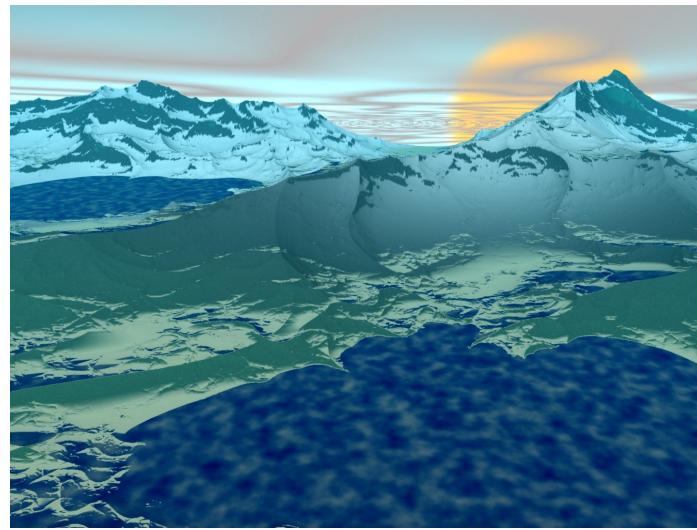
point sunCenter = (0, 0, 0);
point sunEdge = ((u-0.67), (v-0.5)*0.25, 0);
point bottomV = (0,v,0);

float dist1 = distance(sunCenter, sunEdge)*sunSize; //higher Value is smaller sun

dist1 = smoothstep(0.3, 0.4, dist1);
dist1 = clamp(dist1, 0.01, 1.0);

```

Figur: Placering av solen samt dess storlek hanteras av ovanstående kodexempel.



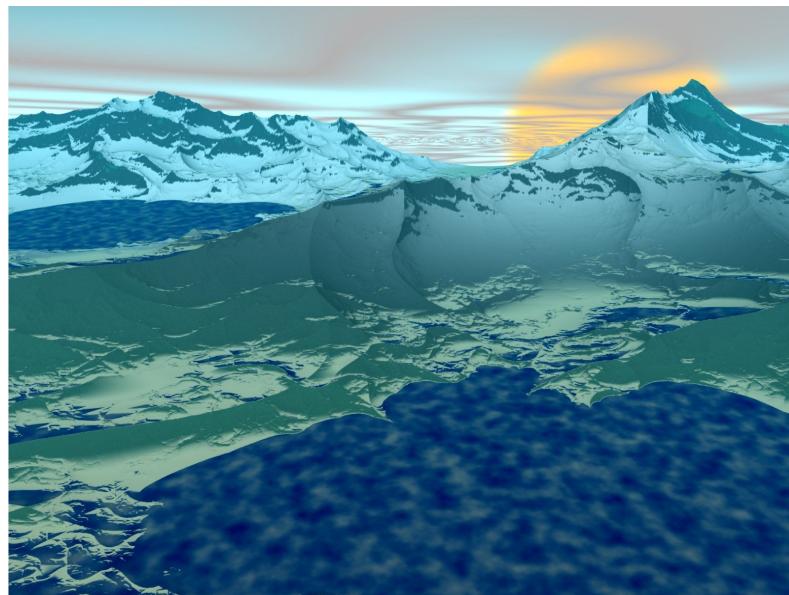
Figur: Det sista som lades till var ett lager moln som baseras på ett fraktalbrus.

```
float vc=(v+0.1)*1.7;
float uCord=((u-0.67)*1.3)/(1.0-vc);
float vCord=((vc-0.6)/(1.0-vc));
point pp=point(uCord,vCord,0);
float clouds = abs(fBm(pp+seed,3,2,0.2));

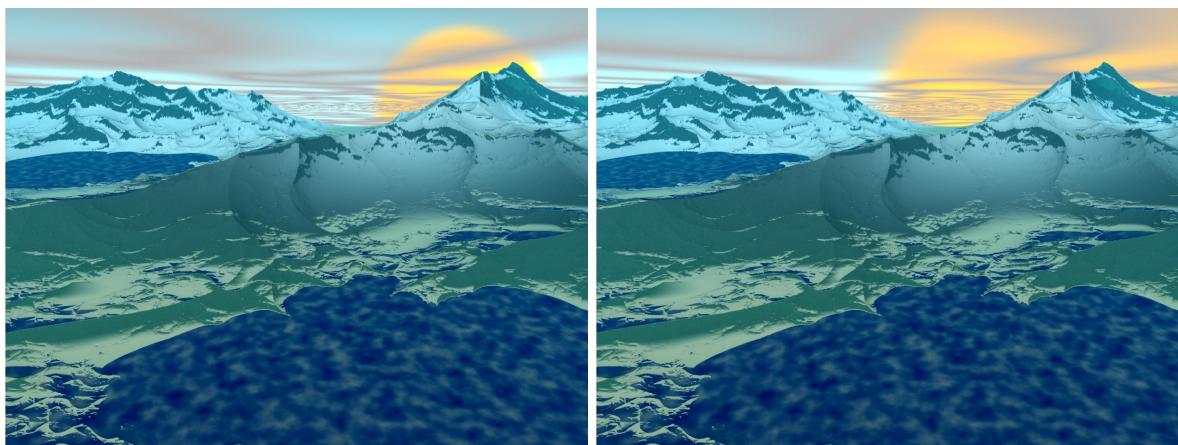
outcol = mix(orange, outcol, dist1) + orange*(1-dist1);
Ci = outcol * clouds + (0.7, 0.7, 0.7) * (1-clouds);
```

Figur: Skapande av moln med fBm (övre stycket) samt sammanslagning av sol, moln och himmel.

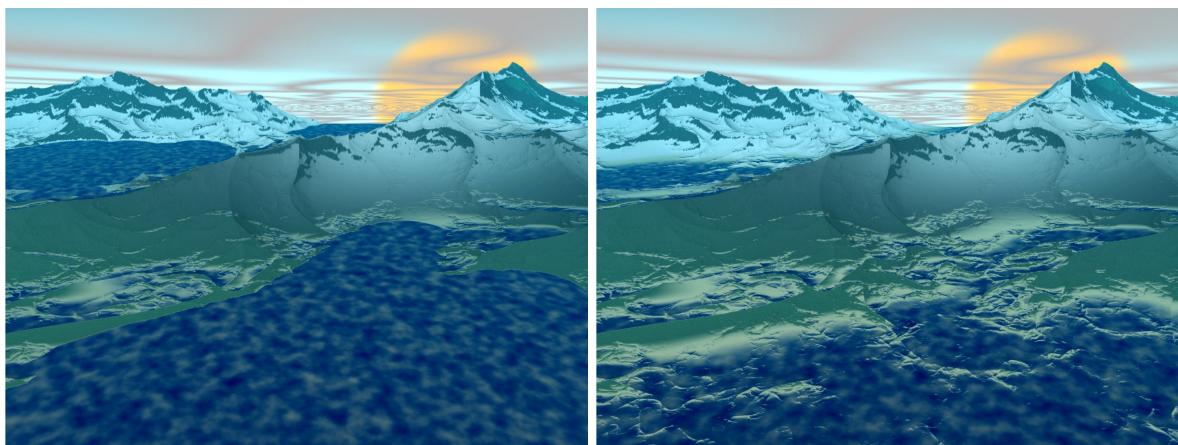
Resultat och Parametervariation



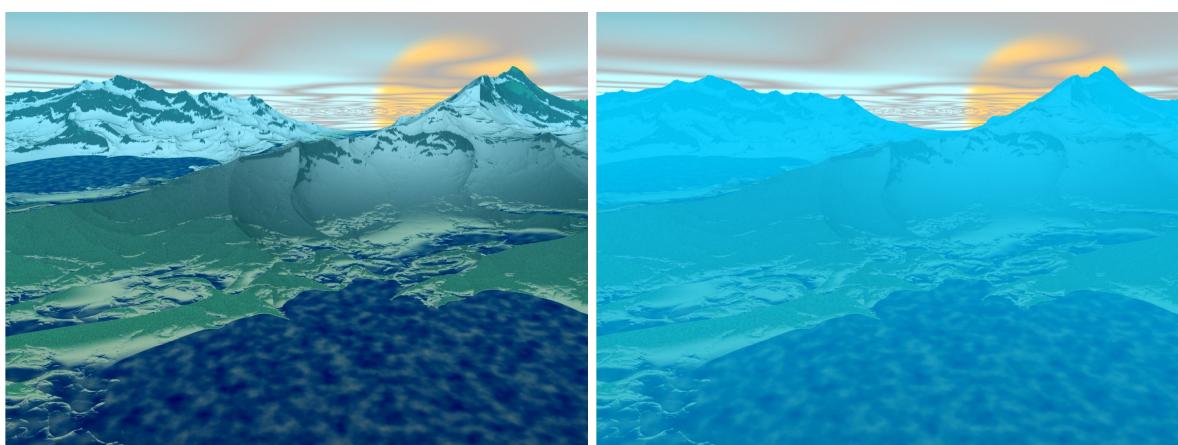
Figur: Slutresultatet så som jag valde att ha samtliga parametrar.



Figurer: Vänster: Ändrad seed till molnenas fraktalbrus (offset på punkten P).
Höger: Storleken på solen



Figur: Vänster: Höjd vattennivå, För att detta ska funka måste smoothstep-parametrarna som styr vart vattenfärgen börjar och slutar vara separat från clamp-värdet som styr i vilken höjd vattnet planar av.
Höger: Vattenytan planas inte längre av, men färgen övergår mot vattnets färg på samma höjd som i slutresultatet.



Figurer: vänster: Frekvensen av brus i gräset är högre än vanligt Höger: Inparametern som styr avståndet på diseffekten delat på fem jämfört mot slutresultatet.

Referenser

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