

# Procedurellt Landskap

Linköpings Universitet  
Procedurella Metoder för Bilder  
TNM084

Caroline Gard, carga635

January 17, 2018

## Abstract

Denna rapport presenterar ett projekt utvecklat i WebGL med Javascriptramverket Three.js. Rapporten sammanfattar även den generella bakgrunden till metoder för att generera procedurella bilder. Denna rapport beskriver ett projekt utvecklat i kursen Procedurella Metoder för Bilder (TNM084) på Linköpings Universitet, och behandlar även strategi, implementation och resultat.

## 1 Introduktion

Procedurella metoder för att generera texturer är ett kraftfullt verktyg inom datorgrafik och handlar om att skapa mönster och strukturer över ytor med matematiska funktioner istället för användning av bildbaserade texturer. Procedurella brusfunktioner är vanliga att använda för att generera slumpartade variationer och komplexitet över ytan. Fördelarna med procedurellt generade bilder är att sådana metoder är mer flexibla, kompakta och tillåter abstraktion och komplexitet, till skillnad från bildbaserade metoder som är mer minneskrävande, mer begränsade för förändring av texturer och har begränsad upplösning.

## 2 Mål

Målet med projektet var att skapa en vy över ett landskap med procedurellt genererade berg, en sjö och övriga naturobjekt genom användning av shaders. Målet var även att implementera flat shading för att uppnå en polygonliknande stil på scenen. Projektet är utvecklat i WebGL och använder sig av programbiblioteket Three.js för en enklare uppsättning av canvas och objekthantering. Inspiration till projektet var från hemsidan <http://www.thehybridforest.com/>.

## 3 Metod

Projektet är utvecklat i WebGL som är ett Javascript API som gör det möjligt att utveckla datorgrafik i 2D och 3D i webbläsaren. Samtliga texturer i scenen är skapade med *ShaderMaterial* inom Three.js med tillhörande manuellt skrivna vertex- och fragmentsshaders. Vertexshadern behandlar vertexpunkterna för objektet och dess position, medan fragmentshadern behandlar färgvärdet och djupvärdet för punkten[1]. De shaders utvecklade för projektet är skrivna i shadingspråket GLSL och javascript, medan de brusfunktioner som används är skrivna av Ian McEwan på Ashima Arts.

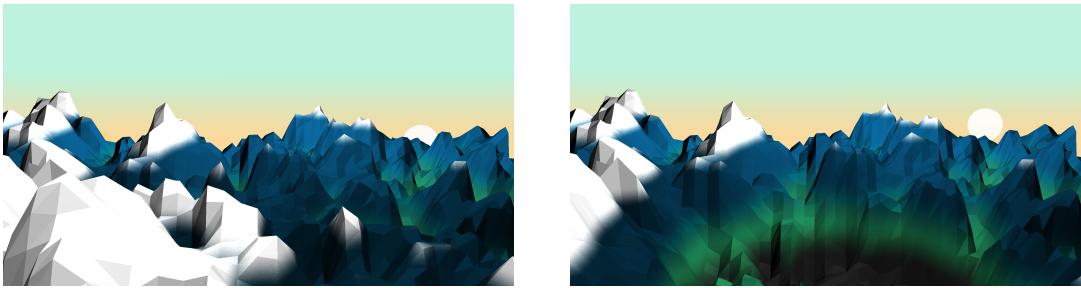
De brusfunktioner använda i projektet är Simplex-brus och Worley-brus. Simplex-brus är en utvecklad brusfunktion av föregångaren Perlin Noise och är användbar för texturer såsom terräng, vågor, eld och moln. Både Simplex- och Perlin-brus returnerar gradienten för varje punkt i intervallet mellan svart och vitt. Dessa brusfunktioner är skapade för att vara kontrollerbara och generera samma resultat, men samtidigt upplevas som slumpad[2]. Worley-brus har en annan struktur till utseendet och kan kortfattat beskrivas som att det baseras på att skapa slumpyrör och sedan avgöra avståndet mellan de närmaste grannpunkterna för att skapa ett cell-likt mönster[3].

Denna del av rapporten beskriver mer detaljerat de shaders använda för de olika texturerna i det implementerade projektet och sammanfattar val av metoder och implementation, och inkluderar även delresultat i form av bilder.

### 3.1 Berg

Bergstexturen utgår från ett plan där varje punkt på planet förflyttas i höjdled längs normalen genom att i vertexshadern behandlar vertexpunkterna genom simplex-brus för tredimensionella objekt. Simplex-brus används för bergstrukturen för att erhålla berg med olika höjd genom scenen. För att skapa en intressantare variation på bergen, kombinerades flera typer av samma brus med olika storlek, genom att för varje bruspåslag förändra input-datan exponentiellt.

För att skapa nedsjunkning i bergen för en kontrollerad placering av sjön, skapades en cirkulär gradient som är mörk i centrum och övergår gradvis till vitt mot kanterna. Detta adderades sedan med brusfunktionen för att erhålla sjön på önskad plats i scenen, se Figur 1.



(a) Berg utan vattenhål.

(b) Berg med vattenhål för placering av sjön

Figure 1: Figur (a) och (b) visar före och efter då gropen för sjön har adderats med brusfunktionen för bergen.

Färgtexturen för bergen beror på bergets höjd. Färgtexturen varierar mellan fyra färger som motsvarar snö, berg, gräs och sjöbotten, och utförs i fragmentshadern. Dessa färgvariationer kombineras med de inbyggda GLSL-funktionerna *smoothstep* och *mix* för att avgöra vart färgförändringen ska ske och bestämma den linjära interpolationen mellan de olika färgerna. Bergets blåa färger varierar från mörk till ljus beroende på avståndet från kameran för att skapa en känsla för djup i scenen. Simplex-brus användes vid övergången från snö till berg för att göra övergången mindre skarp.

### 3.2 Sjöyta

Liknande texturen för bergen, är vattnet baserat på simplex-brus för både vertex- och fragmentshadern, där vattenytan är i grunden uppbyggd av ett plan. Den största skillnaden är att brusfunktionen som används vid vattenberäkningarna är fyrdimensionell, alltså inkluderar även en input-variabel för tid, så att vattenytan kan variera över tid. Vattenytan varierar i två färger baserat på brusfunktionen och är något transparant, se Figur 2 och 3.

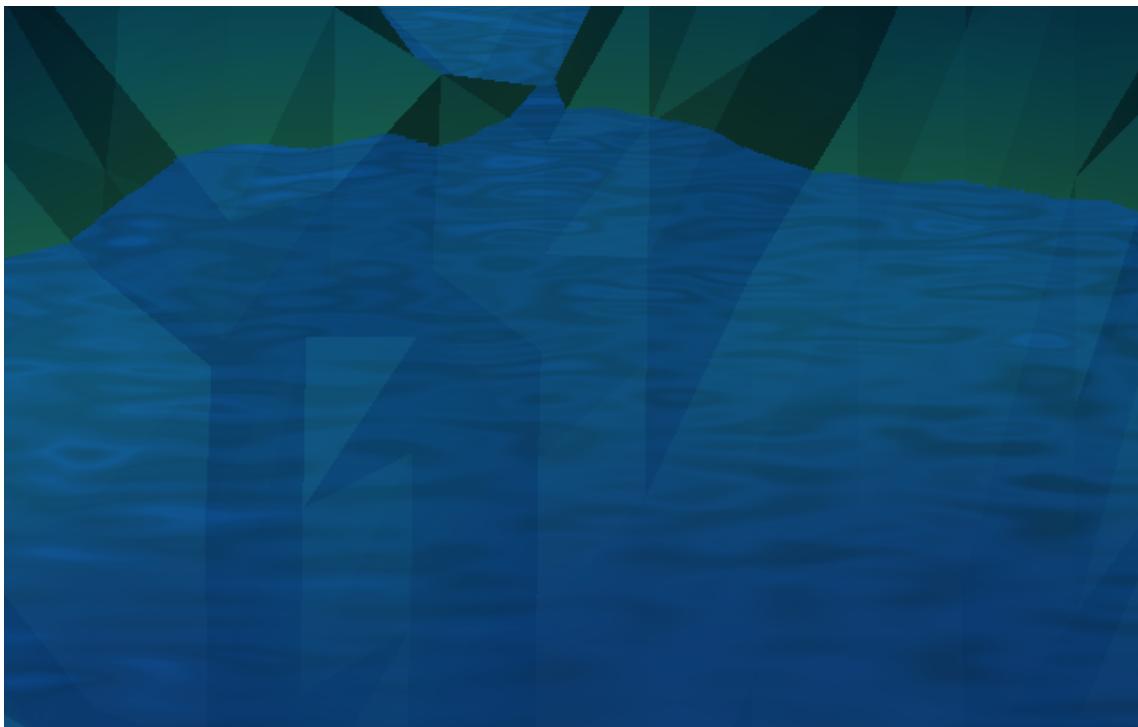
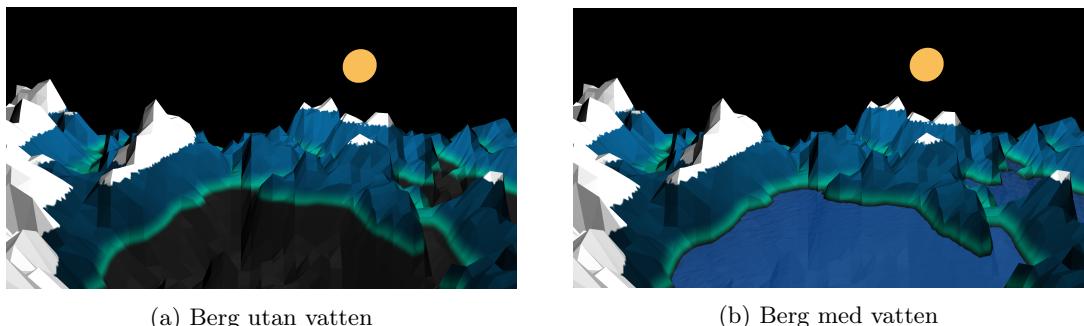


Figure 2: Närbild på vattentexturen



(a) Berg utan vatten

(b) Berg med vatten

Figure 3: Figur (a) och (b) visar ett tidigare stadio då vatten lades till i projektet.

### 3.3 Himmel

Scenen består även av himmel, en sol och en måne. Solen och månen cirkulerar kontinuerligt i xy-led bakom bergen och bestämmer tid för dygnet i scenen. Himlen består av ett upprätt plan som varierar mellan tre typer beroende på tid på dygnet; dag, natt och ett läge som representerar både soluppgång och solnedgång. Samtliga tider på dygnet består av en gradientsskillnad mellan två färger. Himlen som representerar natt har även stjärnor som är skapade med en kombination av simplex-brus och Worley-brus, se Figur 4. Dessa syns som vita prickar med olika transparens med till synes slumpad placering. Solens och månens bana är skapat med sinus- och cosinusfunktioner och samma funktioner avgör tiden för när en form av himmel ska övergå till en annan.



(a) Natthimmel utan stjärnor.

(b) Natthimmel med stjärnor.

Figure 4: Figur (a) och (b) visar före och efter då brusfunktionen för sjärnor applicerats på himlen.

### 3.4 Beräkning av normaler och ljus

Normalen i varje punkt är viktig bland annat för beräkning av ljus. Det inbyggda materialet *Shader-Material* i Three.js, ger direkt tillgång till normalen för tillhörande vertexpunkt i vertexshadern. Problemet är att när vertexpunkten förflyttas med brusfunktionen, blir denna normal felaktig, därmed behövs en ny normal för den förflyttade vertexpunkten beräknas. Detta kan utföras genom att använda GLSL-funktionerna  $dFdx$  och  $dFdy$ , där den partiella derivatan för respektive riktning erhålls för den gällande vertexpunkten. Kryssprodukten av dessa två derivator ger den korrekta normalen för vertexpunkten.

En korrekt beräknad normal är viktig vid ljusberäkningarna av texturerna. Scenen i det implementerade projektet, inkluderar bakgrundsbelysning men även ljus som följer solen respektive månen. Ljuset från solen och månen, påverkar texturen för bergen och beräknas i fragmentshadern, se Figur 5. Detta görs genom att beräkna riktningen för ljuset mot den aktuella vertexpunkten, för att sedan beräkna skalärprodukten mellan ljusets riktning och normalen för vertexpunkten. Resultatet tolkas som ljusintensitet och multiplicera med bakgrundsbelysningen och färgen för punkten.



(a) Måne framför berg

(b) Månen bakom berg

Figure 5: Figur (a) och (b) visar hur ljuset från månen påverkar berget. Liknande gäller även solljuset.

## 4 Resultat

Nedan visas resultatet av projektet i helhet för de olika tiderna på dygnet.



Figure 6: Dag.

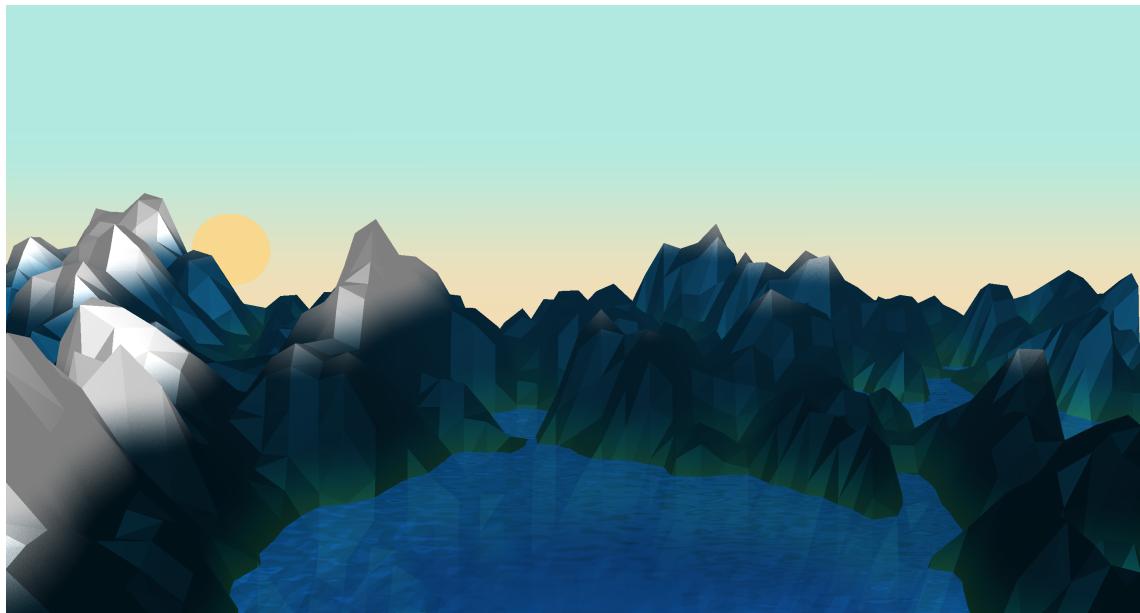


Figure 7: Solnedgång.

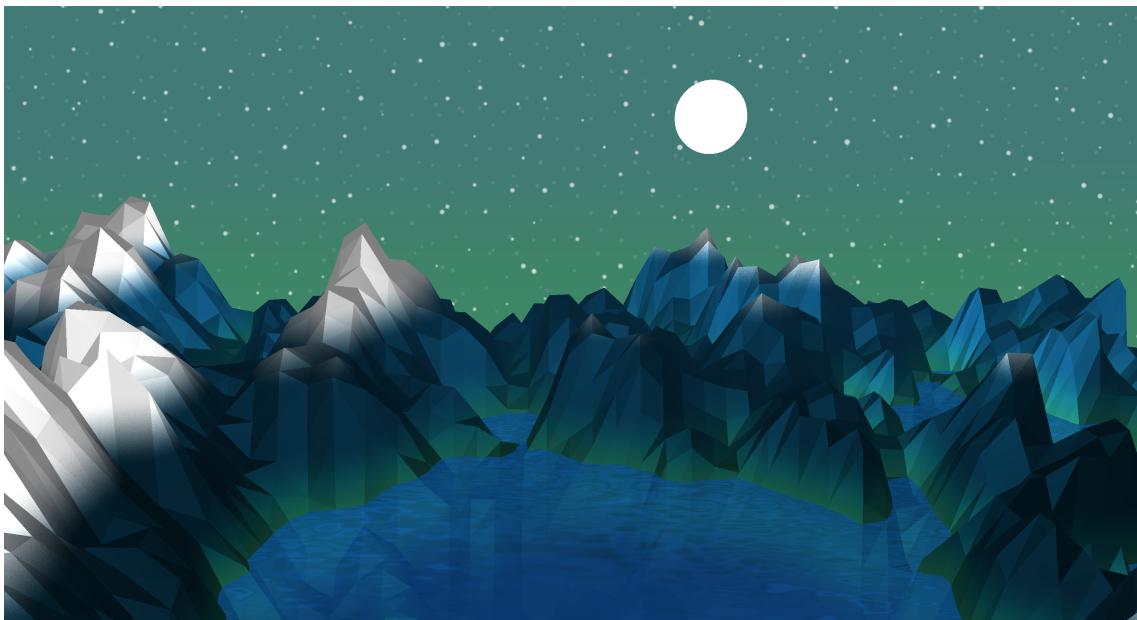


Figure 8: Natt.

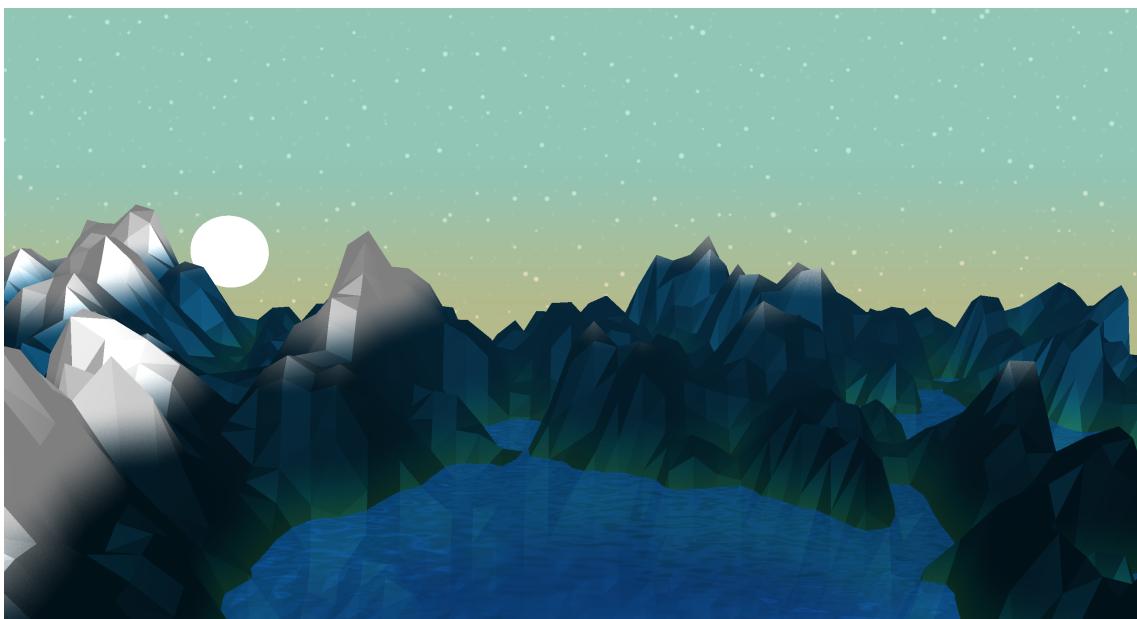


Figure 9: Natt mot soluppgång.

## References

- [1] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, S. Worley, *Texturing and Modeling, a Procedural Approach*, Third edition, 2003
- [2] S. Gustavson, *Simplex noise demystified*, Linköping University, 2005
- [3] P. Gonzales Vivo, J. Lowe, *The Book of Shaders*, <https://thebookofshaders.com/12/>, 2015