
# Shaderprogrammering

## Uppgift 1

## Mjukvarurendering Badboll

Johan Lavén
d15johla@student.his.se

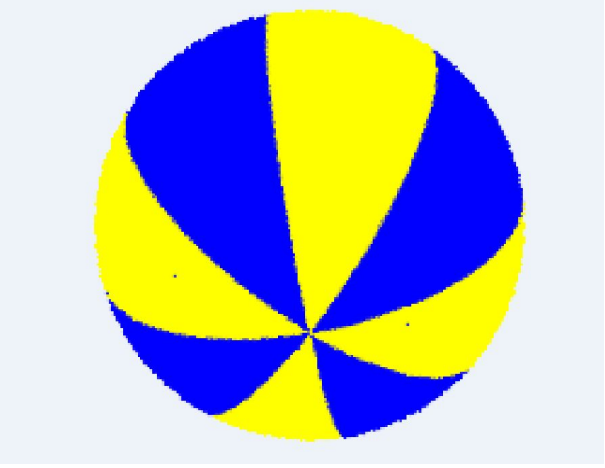
# Deluppgift 1:

Det första steget till att skapa en shader som illustrerar texturerna av en badboll på en sfär var att dela upp sfärens yta i sektioner av klyftor för att möjliggöra användandet av flera färger.

Detta gjordes genom att introducera oss till rendermans RSL-funktion step(min x,y) som returnerar tröskelvärdet 0 då värdet y är mindre än x och returnerar 1 då y är större än givet värde x. En annan funktion som krävdes att vi kände till för att möjliggöra uppdelandet av sfärens yta var den givna RSL-funktionen mod(x, y) som returnerar divisionsresten av x/y. För att välja mellan två olika färger så användes åter en given funktion mix(color x, color y, z) där den returvärdet är en blandning av färgerna x och y baserat på värdet z mellan 0 och 1. Genom att sätta respektive färg till gul och blå och ge z värdet från step-funktionen så kan vi enkelt avgöra de två färgerna som bollen ska delas upp i.

```
float f = mod(u*freq, 2.0);

float g = f;



f = step(0.25, f);

g = step(1.25, g);

f = f-g;

color yr = mix(yellow, blue, f);
```

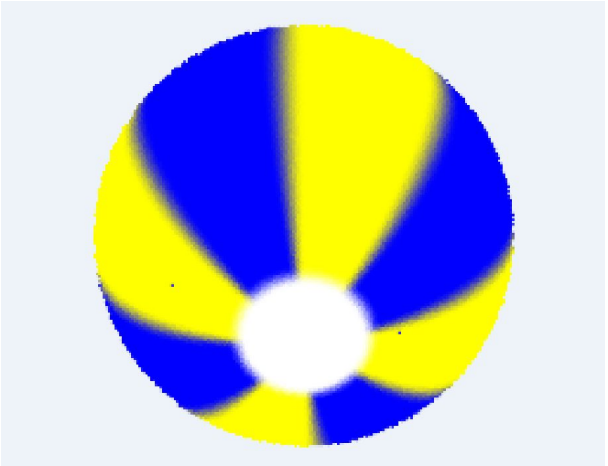
**Figur 1** Kod som illustrerar användandet av mod samt step.



**Figur 1** Badboll i två färger

# Deluppgift 2

För att ge bollen en textur utan dessa skarpa kanter så ersattes den givna step-funktionerna till en liknande funktion kallad smoothstep. Funktionen smootstep tar istället in 3 parametrar där de första två anger vart graderingen från 0 till 1 ska påbörjas och avslutas. Det ger då att returvärdet av funktionen alltid är mellan 0 och 1. Utöver denna ändring så lades ytterligare ett färgfält till på sfärens södra pol för att efterlikna utseendes hos en klassisk badboll, även här hanterades övergången mellan färgerna med hjälp av smoothstep och mix-funktionerna inbyggda i renderingsspråket.

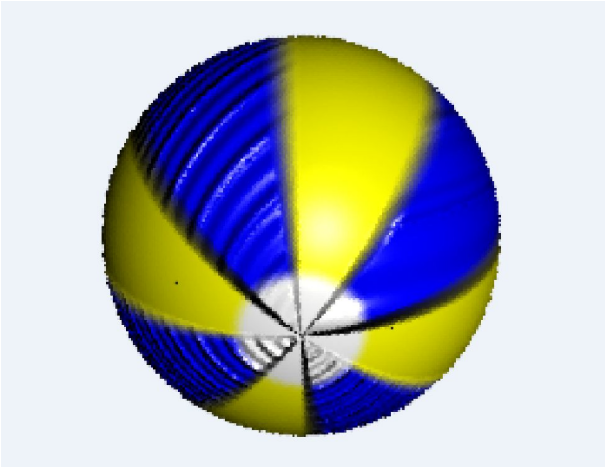


```
...
        f = smoothstep(0.0, 0.25, f);

        g = smoothstep(1.0, 1.25, g);

...

        b = smoothstep(0.75, 1.0, b);

        color yr = mix(yellow, blue, f);

        yr = mix(white, yr, b);
```

# Deluppgift 3

Det sista som skulle justeras på badbollen var att de blå klyftorna av bodbollen ska framstå som insjunkna i sfären samt att de ska någon typ av textur baserat på den inbyggda noise-funktionen. Noise-funktionen “noise(type x)” kan returnera en punkt, float, color eller vector baserat på vad man ska använda returvärdet till. Genom att applicera noise-funktionen på en given punkt som på badbollen så kan ytan få en speciell och eftertraktad textur som efterliknar vågor eller bulor på normalen. Den sista effekten som applicerades mot sfären var en ljussättning med direkt ljus (diffuse) samt speglat ljus (specular) som gav ett slutresultat som efterliknade en blank yta med texturer på de blåa fälten av badbollen.

```
…
        myPoint += f*noise(mod(u*v*20, 0.5)*nFreq)*N*bumpdepth;
…
        color dcolor = yr*diffuse(n);
        color scolor = speccolor*specular(n, V, lightIntensity);
```



# Diskussions

I boken *Advanced RenderMan, Creating CGI for Motion Pictures* (1999:243-245) diskuteras hur procedurella shaders kombineras med färdiga texturer för att konstnären eller programmeraren ska på ett mer kontrollerbart sätt uppnå det resultat som söks. Det kunde som i deras exempel vara att på en enkel textur används som indata genom att måla olika områden i varierande nyanser av en gråskala för att specificera för shadern hur mycket rost som ska genereras på de olika positionerna. Dessa metoder är inget som användes i uppgiften för att färglägga och texturera badbollen men skulle kunnat ha använts till att specificera vart de olika klyftorna skulle delas upp istället för att göra det matematiskt. En annat exempel som tas upp i bokens kapitel är att färdiga texturer nästan bara exlusivt används när det gäller att skapa karaktärer där konstnären behöver full kontroll i alla lägen.

# Referenser

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