Rapport i Avancerad Datorgrafik

# Uppgift 1

## Kod

surface enkelshader(

color purple = (0.5, 0.0, 0.6); // Purple color

color orange = (0.6, 0.2, 0.0); // Orange color

color white = (1.0, 1.0, 1.0); // White color

float frequency = (4.0); // Frequency of meridians (stripes) in the sphere

float diffuseStrength = (0.9); // Value defining the intensity of the light being shed on the object

float roughness = (0.1); // The level of irregularity

float bumpheight = (0.8); // The depth of the texture bumps

float f = (0.1); // The interval (size) of the white area

)

{

float segments = mod((s \* frequency), 1.0); // Iterate frequencies through the meridians (s) to create sphere segments

float alignedMeridian = mod((s \* frequency + 0.125), 1.0); // Additional frequency used to correctly align with the meridians

// Smoothly transition from 0.25 units until 0.5 units at each frequency

// First value indicates the end of solid color (from 0.0)

// Second value indicates end of smoothed color, i.e. when that color ceases

float f1 = smoothstep(0.25, 0.5, segments);

float f2 = smoothstep(0.75, 1.0, segments);

float f3 = smoothstep((f - 0.065), f, v);

// Calculate a new normal by altering the global view point and create a noise effect coordinated using bumpheight

P += step(0.5, alignedMeridian) \* (noise(P \* frequency) \* N \* bumpheight);

N = calculatenormal(P);

// Calculate the normal of the shape

normal normalValue = normalize(N);

// Create a base color mix constisting of orange and purple, smoothly blending them using the f1 and f2 smoothstep intervals

// Create a new color mix, adding white to the base mix amd switching/blending between the interval of the third parameter f3

color baseColor = mix(orange, purple, f1-f2);

color whitedBaseColor = mix (white, baseColor, f3);

// Diffuse the color to a defined intesity by multiplying it with a diffuse function

color diffusedColor = mix (whitedBaseColor, whitedBaseColor \* diffuse(normalValue), diffuseStrength);

// Rendering instruction

Ci = diffusedColor;

Oi = Os;

}

## Reflektion

## Procedurella Shaders kontra Färdiga Texturer