

Physically based Rendering

FABIAN MEISTER

Matrikelnummer: 298 5828

E-Mail: meisterfa68426@th-nuernberg.de

Technische Hochschule Nürnberg Georg Simon Ohm

1 Einleitung

Der Begriff Physically Based Rendering/Shading (kurz PBR) beschreibt einen Überbegriff, welcher verschiedene Rendering Methoden und Techniken umfasst. Diese basieren auf physikalischen Theorien und Prinzipien, welche darauf ausgerichtet sind, die Wechselwirkung zwischen Licht und Materie so korrekt wie möglich zu modellieren. Zu diesen physikalischen Gegebenheiten zählt beispielweise die Energieerhaltung innerhalb eines Systems. (vgl. S.133 David Wolf und DeFries)

Das PBR ist dennoch keine physikalisch korrekte Simulation des Lichtes, da es Approximations-funktion verwendet, um den Arbeitsaufwand und somit die gesamte Berechnungsdauer zu verringern. Aus diesem Grund wird es Physically Based (zu dt. physikalisch basierendes) Rendering genannt und nicht Physically Rendering. (DeFries)

2 Physikalische Grundlagen der Radiometrie

3 Reflectance Equation

3.1 Bidirectional Reflective Distribution Function

3.2 Lambert Diffuse BRDF

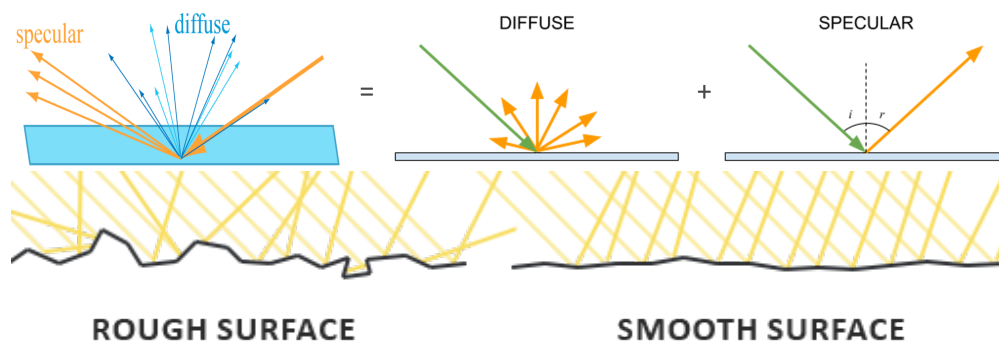
3.3 Cook-Torrance specular BRDF

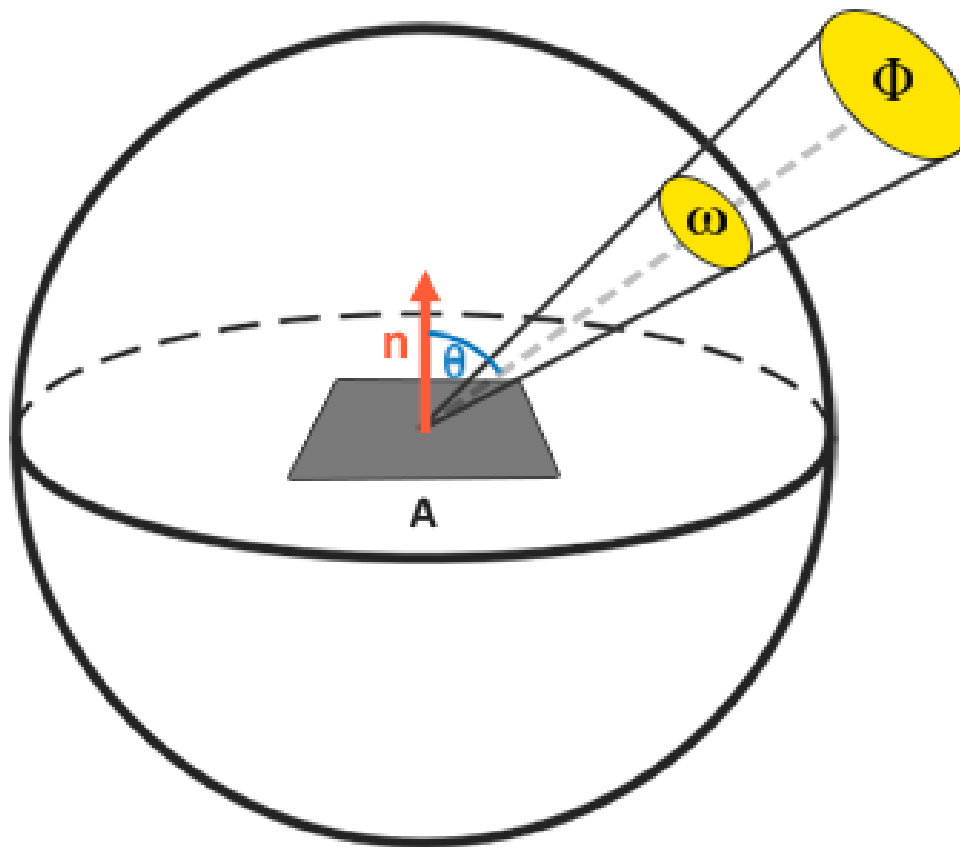
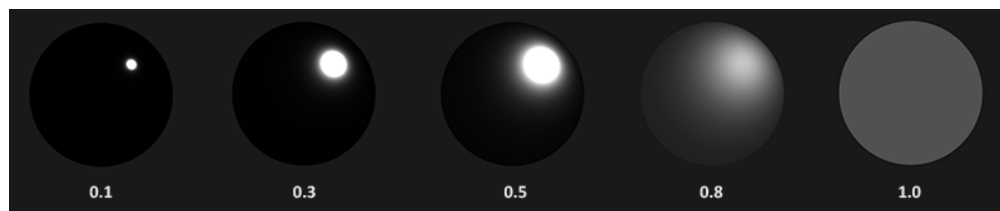
3.3.1 Microfacet Surface Model

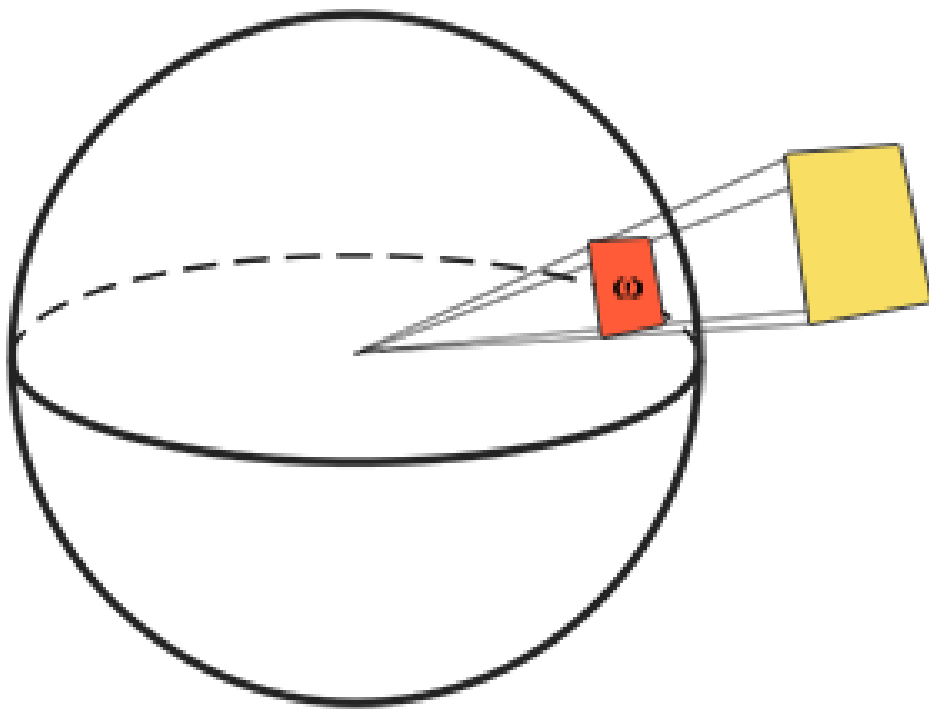
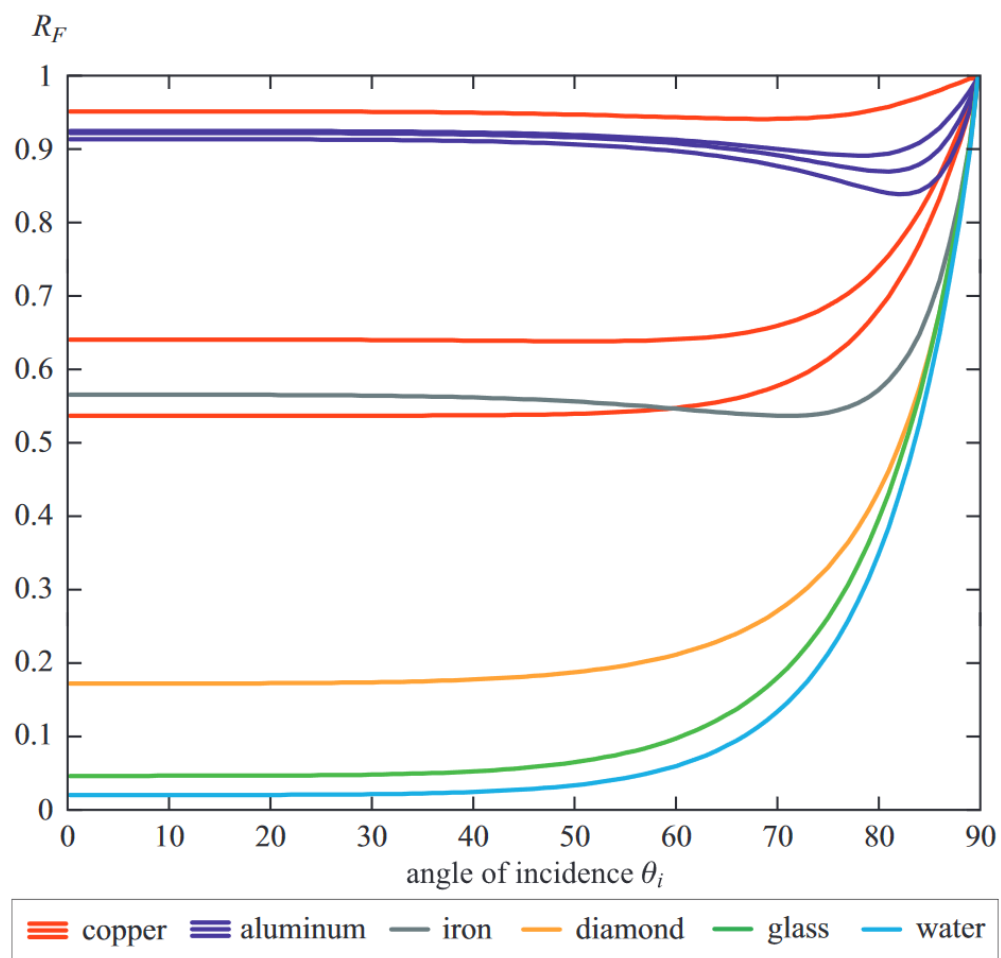
3.3.2 Normal distribution function

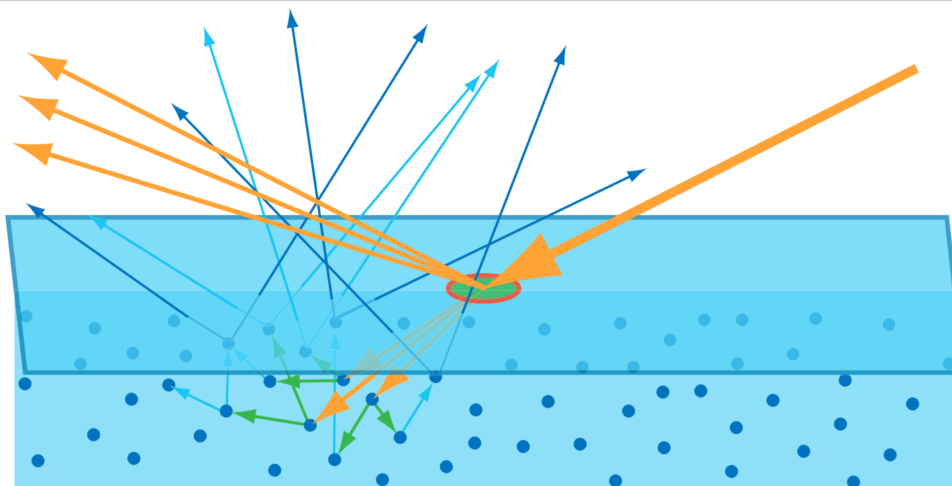
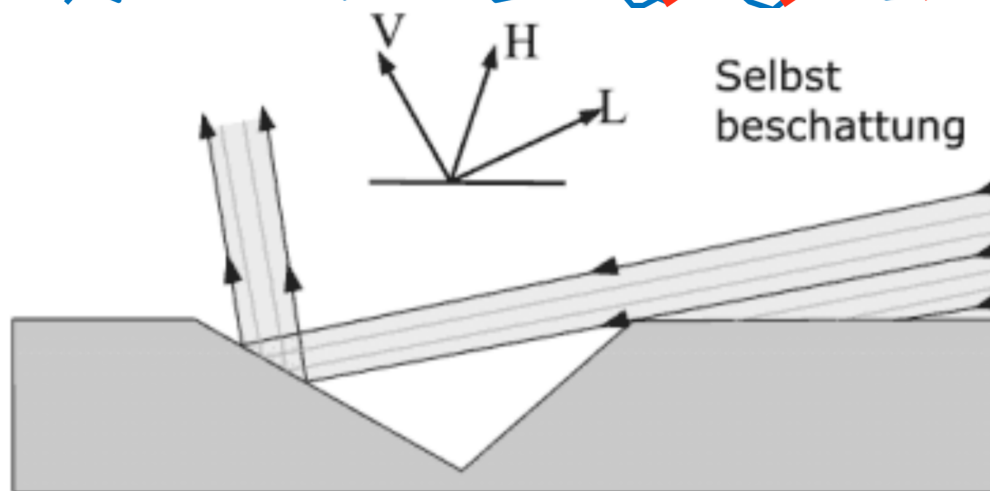
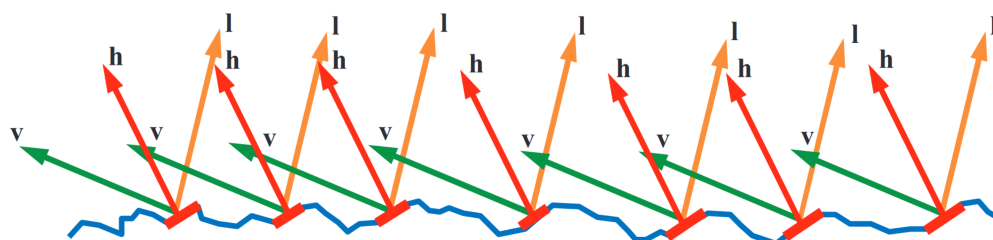
3.3.3 Geometry function

3.3.4 Fresnel equation









```

float geomSmith( float dotProd ) {
    float k = (Material.Rough + 1.0) * (Material.Rough + 1.0) / 8.0;
    float denom = dotProd * (1 - k) + k;
    return 1.0 / denom;
}

vec3 F = schlickFresnel(hDotV);
vec3 kD = mix((vec3(1.0) - F), vec3(0.0), Material.Metal);
(kD * diffuseBrdf / PI + specBrdf) * lightIntensity * nDotL

if (meshNumber == 1){
    model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f,
        1.0f, 0.0f));
    setMatrices();
    mesh1->render();
}

```

```

    }
    else if (meshNumber == 2){
        ...
    }
    else if (meshNumber == 3){
        ...
    }
}

if (glfwGetInputMode(window, GLFW_CURSOR) == GLFW_CURSOR_DISABLED)

while( ! glfwWindowShouldClose(window) && !glfwGetKey(window,
    GLFW_KEY_ESCAPE) ) {
    keypress = "";
    [...]
    processKeypress(window, keypress);
    scene->update(float(glfwGetTime()), keypress);
    [...]
}

front.x = sin(glm::radians(Yaw)) * cos(glm::radians(Pitch));
front.y = sin(glm::radians(Pitch));
front.z = ((-1) * cos(glm::radians(Yaw))) * cos(glm::radians(Pitch))
    ;

void processKeypress(GLFWwindow* window, std::string& keypress)
{
    if (glfwGetKey(window, GLFW_KEY_1) == GLFW_PRESS)
        keypress = "gold";
    else if (glfwGetKey(window, GLFW_KEY_2) == GLFW_PRESS)
        keypress = "copper";
    [...]
    else if (glfwGetKey(window, GLFW_KEY_C) == GLFW_PRESS)
        animate = true;
}

void ScenePbr::processKeyboardInput(std::string& keypress, float
    deltaT)
{
    if (keypress == "forward")
        camera.ProcessKeyboard(FORWARD, deltaT);
    else if (keypress == "backward")
        camera.ProcessKeyboard(BACKWARD, deltaT);
    [...]
}

glfwGetCursorPos(window, &xpos, &ypos);
scene->updateMouseMove(xpos, ypos);

if (objMaterial == "gold")
{
    // Gold
    drawSpot(glm::vec3(0.0f, 0.0f, 1.5f), metalRough, 1, glm::vec3(1,
        0.71f, 0.29f));
}
[...]
else if (objMaterial == "copper")
{
    // Copper

```

```

    drawSpot(glm::vec3(0.0f, 0.0f, 1.5f), metalRough, 1, glm::vec3
      (0.95f, 0.64f, 0.54f));
  }

```

$$\begin{aligned}
\Phi &= \frac{\Delta Q}{\Delta t} \quad \omega = \frac{A}{r^2} \quad L = \frac{\Delta \Phi \cdot}{\Delta \Omega \cdot \Delta A \cdot \cos \varepsilon} \quad E = \frac{\Delta E}{\Delta A} \quad E = \int_{\Omega} L \cdot \cos \varepsilon \, d\omega \quad L_o(p, \omega_o) = \\
\int_{\Omega} f_r(p, w_i, W_o) L_i(p, w_i) n \cdot w_i \, dw_i \quad f_r(p, w_i, w_o) &= \frac{\Delta L_o(p, \omega_o)}{\Delta E_i(p, w_i)} \quad f_{Lambert} = \frac{c}{\pi} \quad f_{CookTorrance} = \\
\frac{D \cdot G \cdot F}{4 \cdot (w_0 \cdot n) (w_i \cdot n)} \quad D_{GGXTR}(n, h, \alpha) &= \frac{\alpha^2}{\pi \cdot \left((n \cdot h)^2 \cdot (\alpha^2 - 1) + 1 \right)^2} \quad G_{SchlickGGX}(n, v, k) = \\
\frac{n \cdot v}{(n \cdot v) \cdot (1 - k) + k} \quad k_{direct} &= \frac{(\alpha + 1)^2}{8} \quad G_{Smith}(n, v, l, k) = G_{sub}(n, v, k) \cdot G_{sub}(n, l, k) \\
F_{Schlick}(h, v, F_0) &= F_0 + (1 - F_0) \cdot (1 - (h \cdot v))^5 \\
L_0(p, \omega_0) &= \int_{\Omega} \left(k_d \cdot \frac{c}{\pi} + k_s \frac{D \cdot G \cdot F}{4 \cdot (w_i \cdot n) (\omega_0 \cdot n)} \right) \cdot L_i(p, w_i) \cdot n \cdot \omega_i \, d\omega_i
\end{aligned}$$

Literatur