REDAREA NON-FOTOREALISTA
TOON SHADING

Prof. univ. dr. ing. Florica Moldoveanu

Curs *Elemente de Grafică pe Calculator* – UPB, Automatică şi Calculatoare
2021-2022

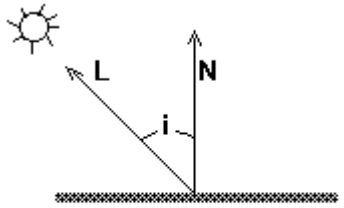
Afişarea în 2 nuanțe

Redare non-fotorealistă ← vs. → readare fotorealista, urmarita in Computer Graphics

- Tehnici şi stiluri de redare care crează imagini diferite de cele fotografice: inspirate din pictura, desen artistic, desene animate

Toon shading (cel shading)

- Tehnici de redare non-fotorealistă utilizate pentru a obține aparența de desen manual al unui obiect tri-dimensional.
- "**toon**" – personaj dintr-un film de desene animate (cart**oon** film).



Reflexia difuza

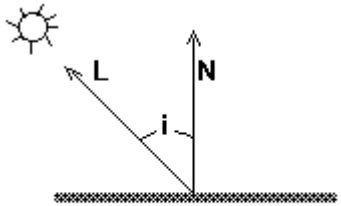
Se poate implementa în fragment shader sau în vertex shader.

Principiul "Toon shading":

- Reflexia difuză a luminii de la sursă este discretizată în 2 sau mai multe niveluri → într-un desen manual nu pot fi redate niveluri de reflexie într-un spectru continuu!
- Discretizarea se bazeaza pe modificarea produsului scalar (L•N)

Discretizarea reflexiei difuze în trepte

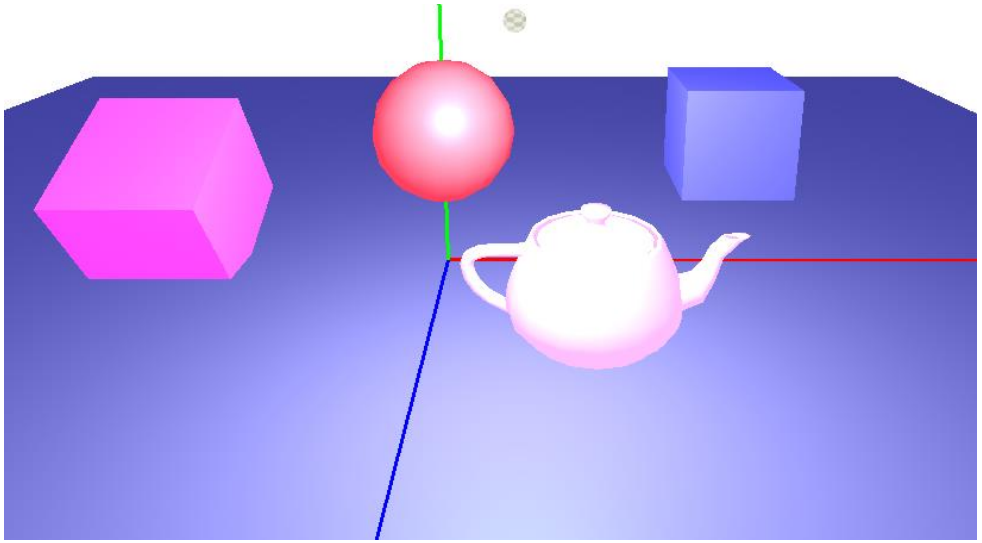
```
 NL = dot(N, L);

//discretizeaza reflexia difuza

////discretizare în 4 trepte /////////

            if(0 <= NL && NL <= 0.25) NL =0;

            if(0.25 <= NL && NL <= 0.5) NL =0.25;

            if(0.5 <= NL && NL <= 0.75) NL =0.5;

            if(0.75 <= NL && NL <= 1) NL =0.75;

 ////  în 2 trepte //////////

            if(0 <= NL && NL <= 0.5) NL =0;

            if(0.5 < NL && NL <= 1) NL =0.5;

vec3 diffuse_light = material_kd * max(NL,0);
```



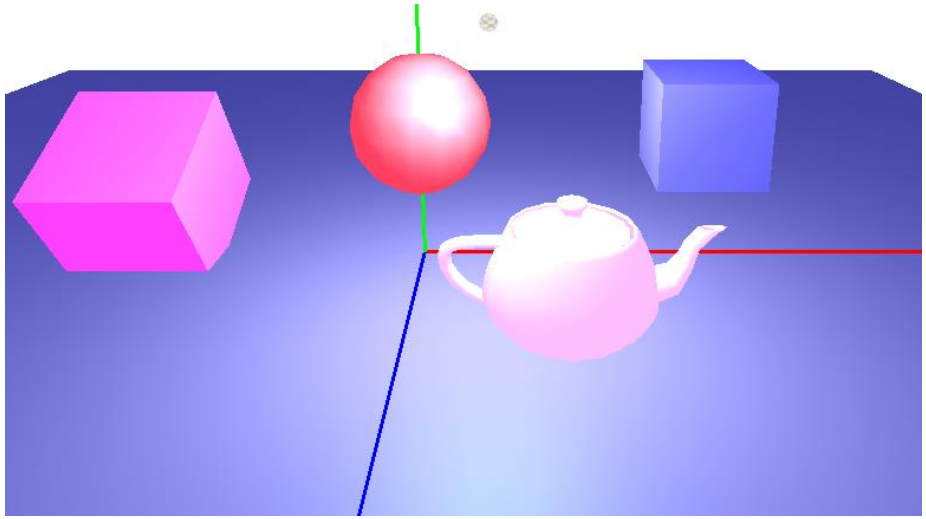
Reflexia difuza

Exemplu: Implementare in Vertex shader

```
layout(location = 0) in vec3 v_position;
layout(location = 1) in vec3 v_normal;
uniform int diffuse_toon; //uniform ..matrici de transformare, pozitie sursa, observator, constante pt calcul reflexie
out vec3 color;
void main()
{ vec3 world_position = (Model * vec4(v_position, 1)).xyz;
  vec3 N = normalize(mat3(Model) * v_normal);
  vec3 L = normalize(light_position - world_position);
  vec3 V = normalize(eye_position - world_position);
  vec3 H = normalize(L + V);
  vec3 ambient_light = material_ka * ambient_intensity;
  float NL = dot(N, L);
* if (diffuse_toon==2 || diffuse_tone == 4) discretizeaza reflexia difuza modificand NL
 vec3 diffuse_light = material_kd * max(NL,0);  //kdλ*max((Lui • Nu),0
 vec3 specular_light = vec3(0);
if ( NL > 0)
    specular_light = material_ks * pow(max(dot(N, H), 0), material_shininess);  //  fat*ksλ*max((Nu•Hu), 0)n
float factorAtenuare = 1 / (kc + kl * distance(light_position, world_position) + kq * pow(distance(light_position,
world_position), 2));
color = material_ke + ambient_light + factorAtenuare * light_intensity * (diffuse_light + specular_light);
gl_Position = Projection * View * vec4(world_position, 1.0);
}
```

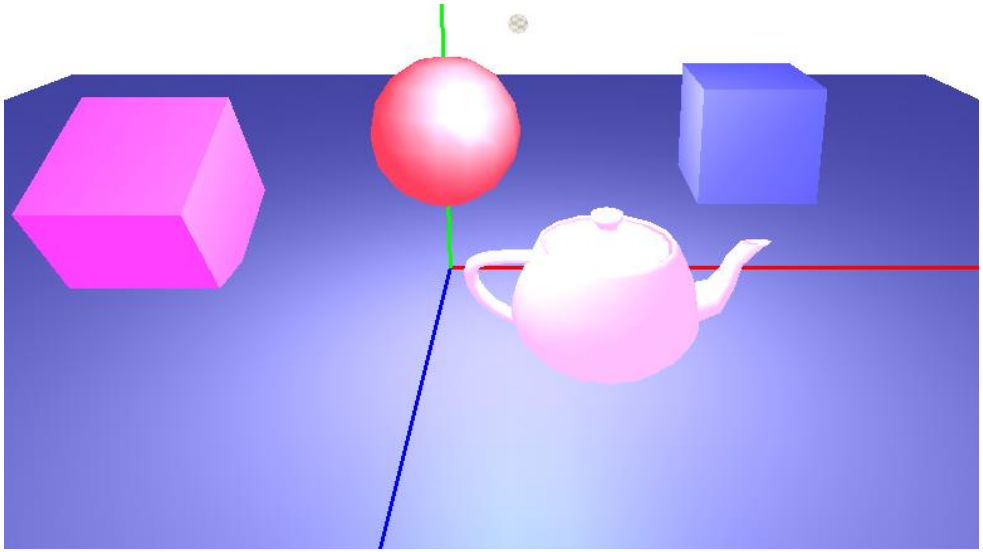


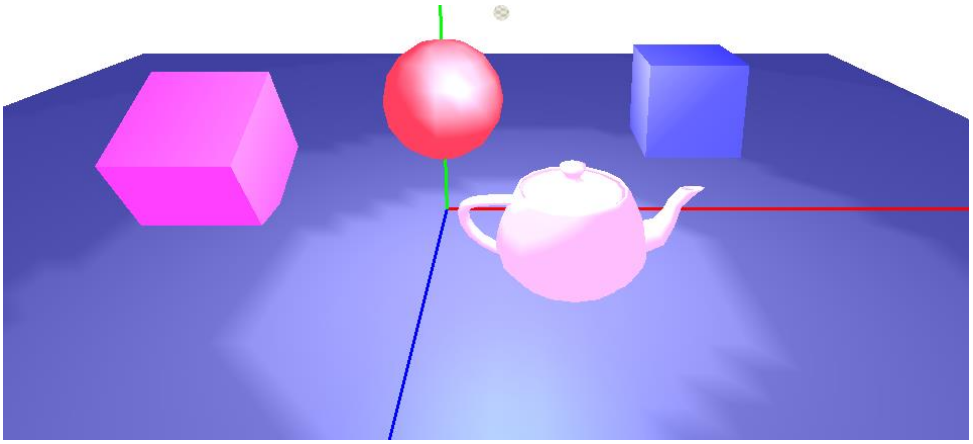
Phong shading



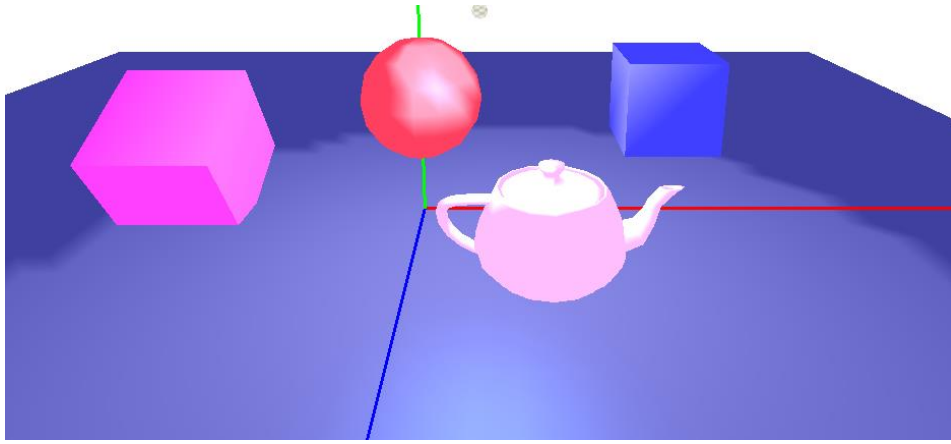
Gouraud shading

Gouraud

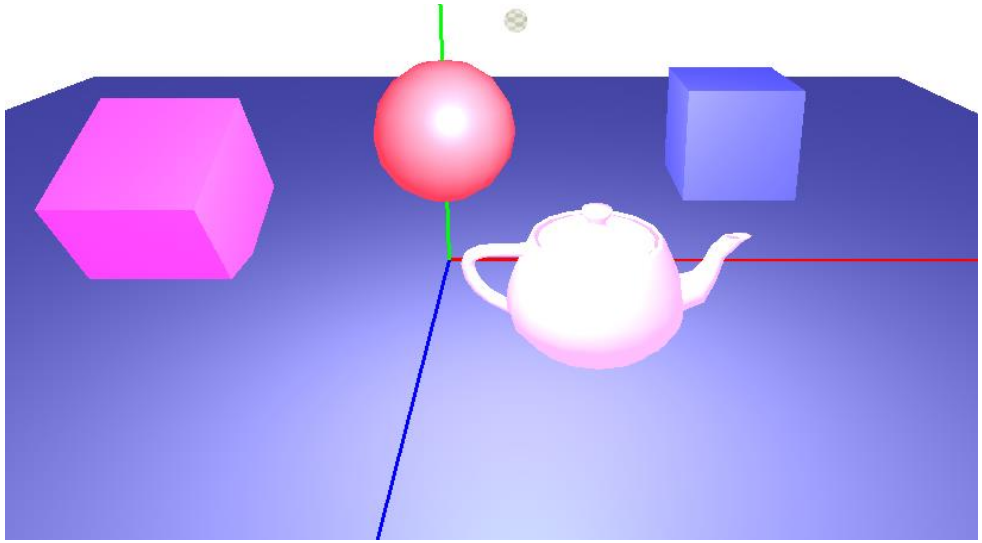




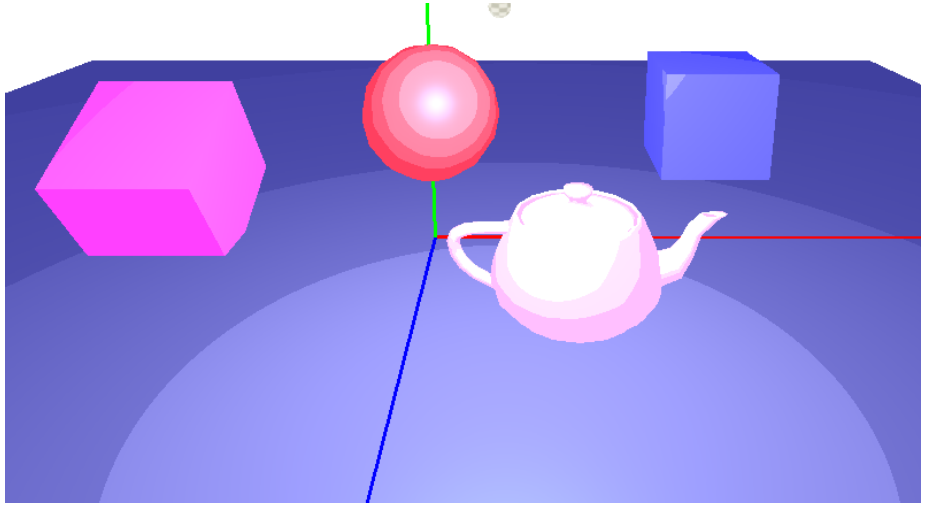
Gouraud – iluminare difuză în 4 trepte



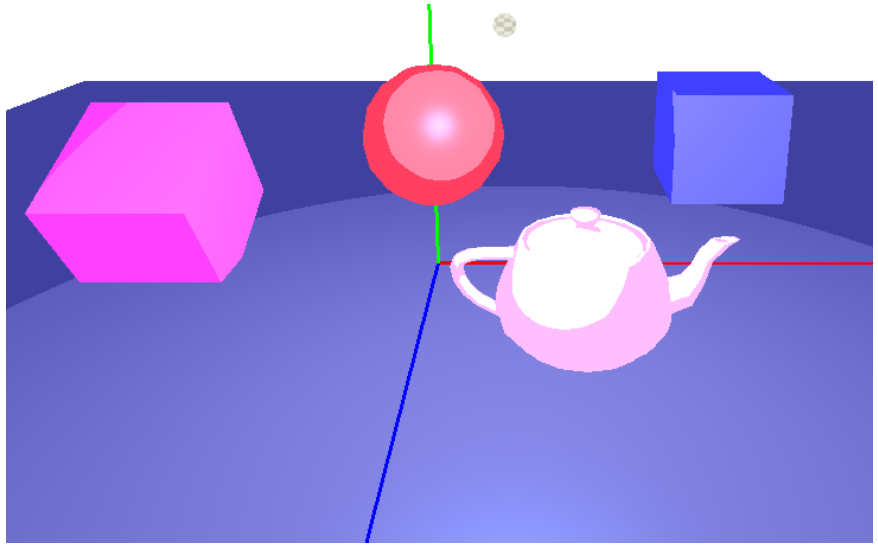
Gouraud – iluminare difuză în 2 trepte



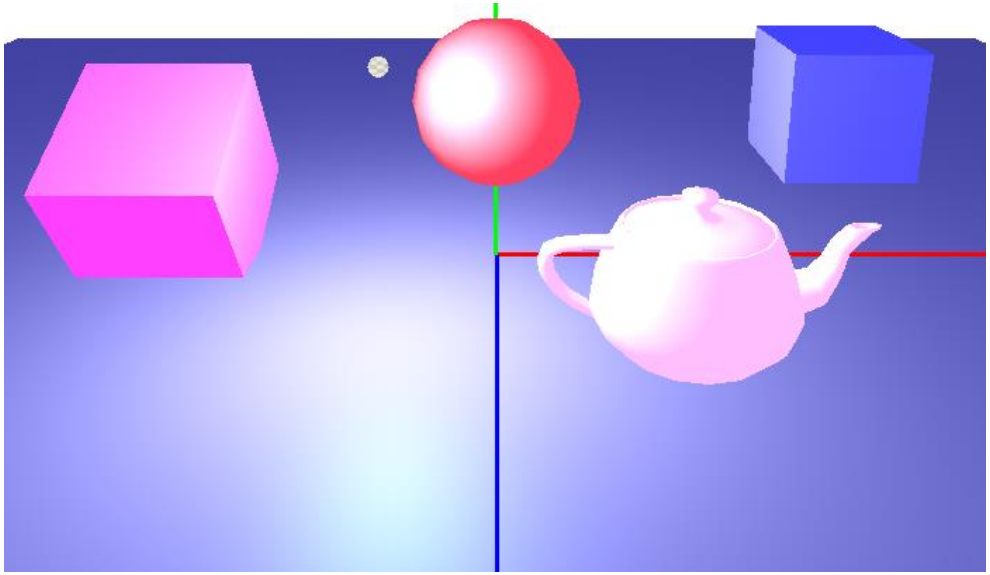
Phong



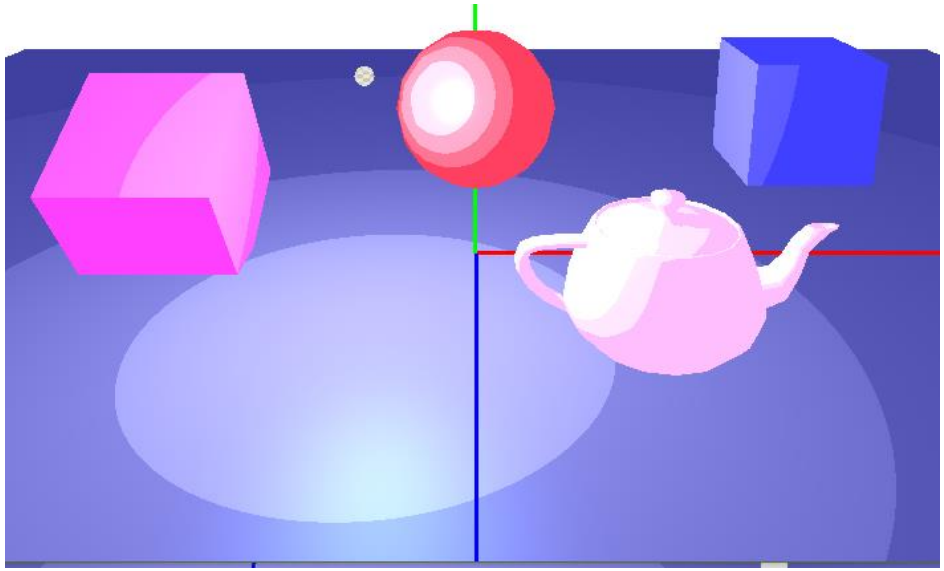
Phong – iluminare difuză în 4 trepte



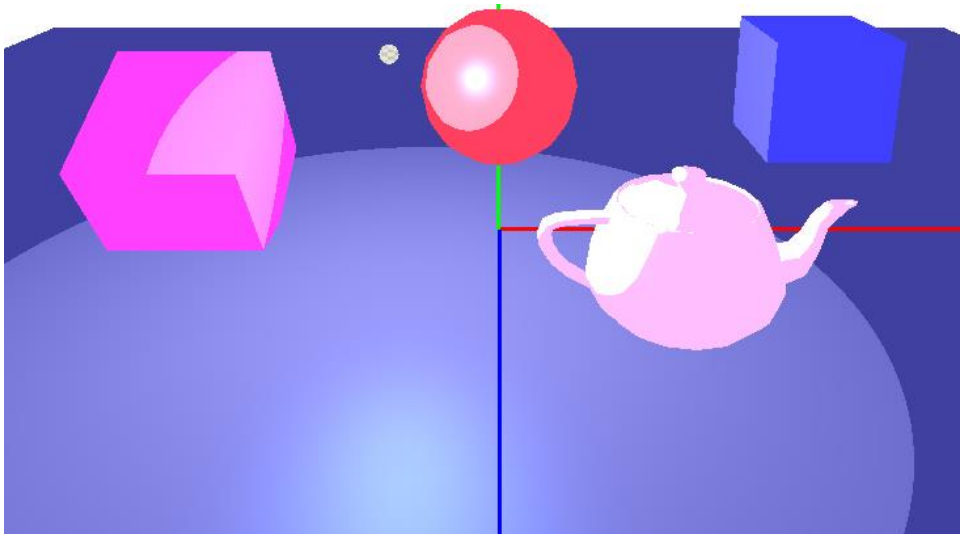
Phong – iluminare difuză în 2 trepte



Phong



Phong – iluminare difuză în 4 trepte



Phong – iluminare difuză în 2 trepte