CS 563 Advanced Topics in Computer Graphics
Materials

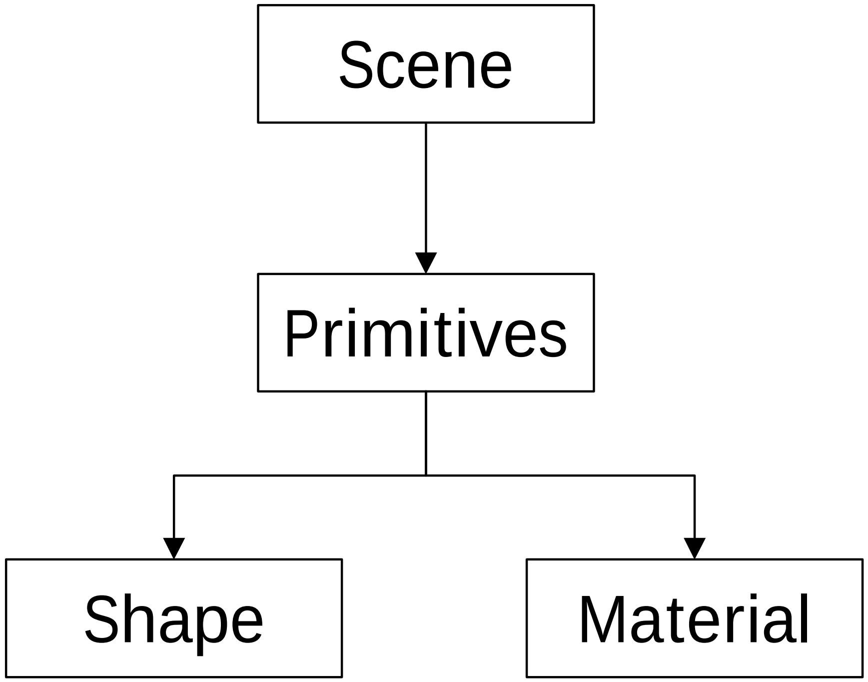
by Paulo Gonçalves de Barros

Summary

- Basic concepts
- BSDF
- Material interface and implementations
- Bump mapping

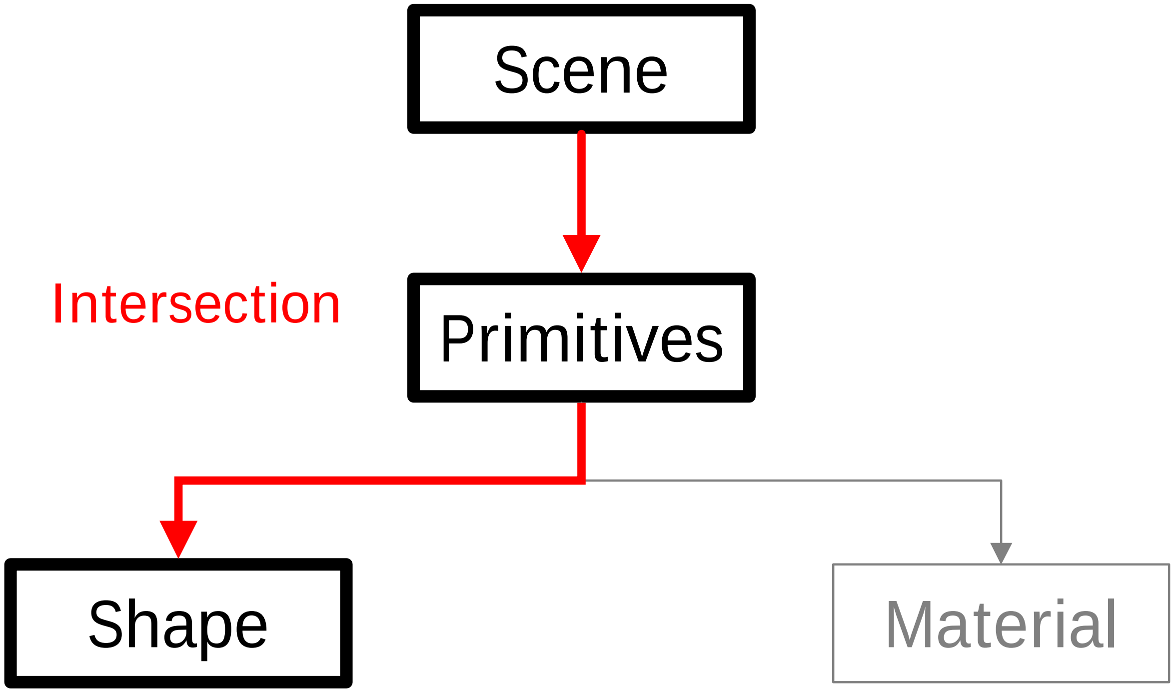
Basic concepts

- Scene structure



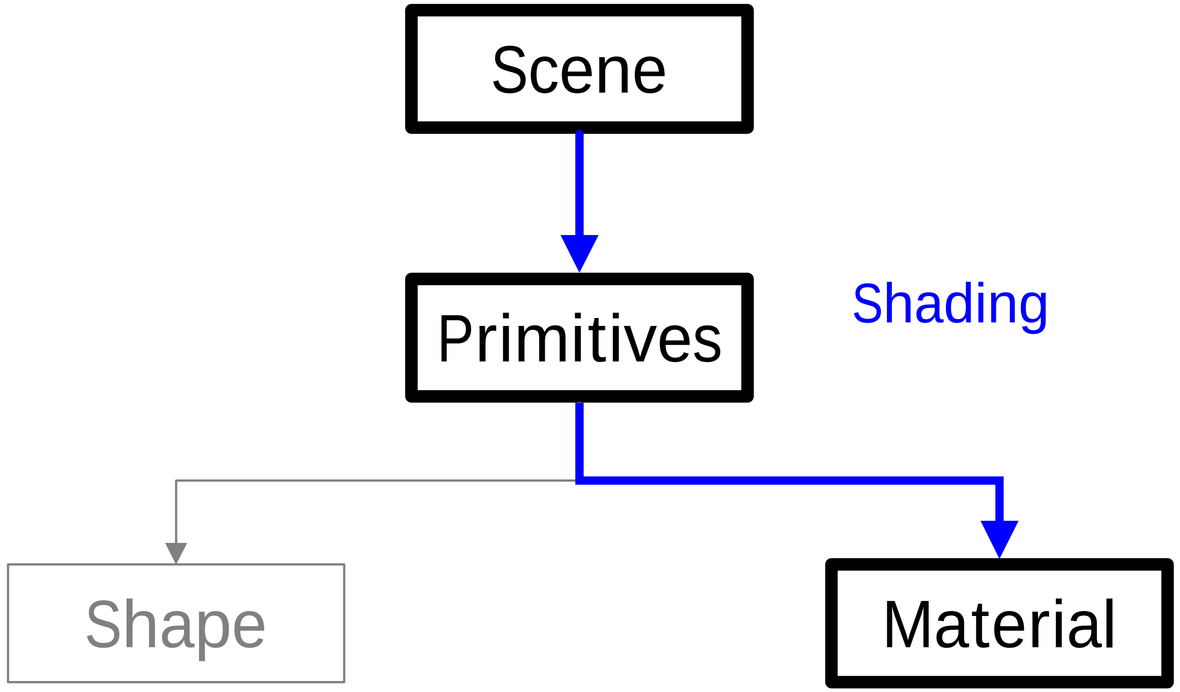
Basic concepts

- Calculating hits



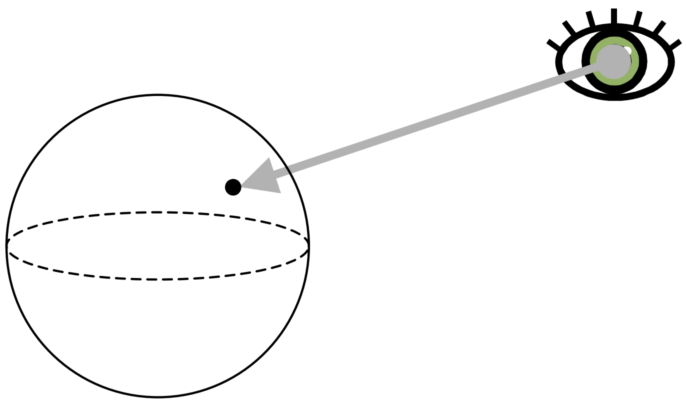
Basic concepts

- Obtaining ray colors



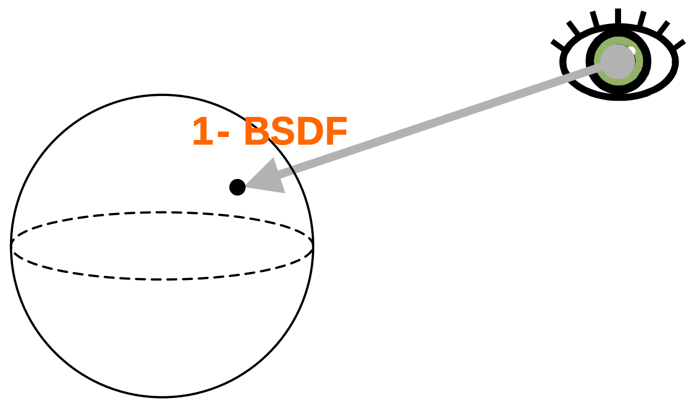
Basic concepts

- Obtaining ray colors



Basic concepts

- Obtaining ray colors

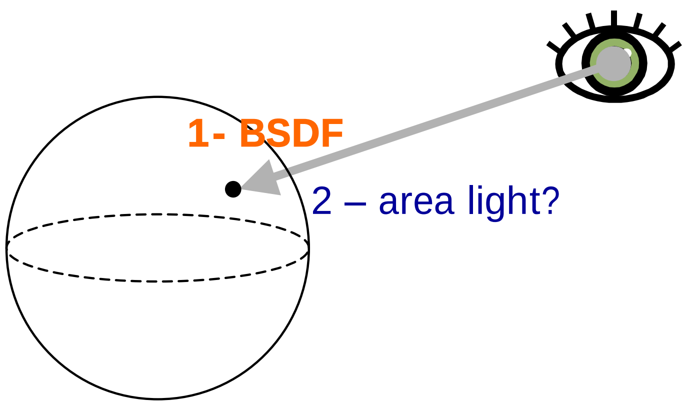


Whitted's

```
<Compute emitted and reflected light at ray intersection point>
    <Evaluate BSDF at hit point>
```

Basic concepts

- Obtaining ray colors

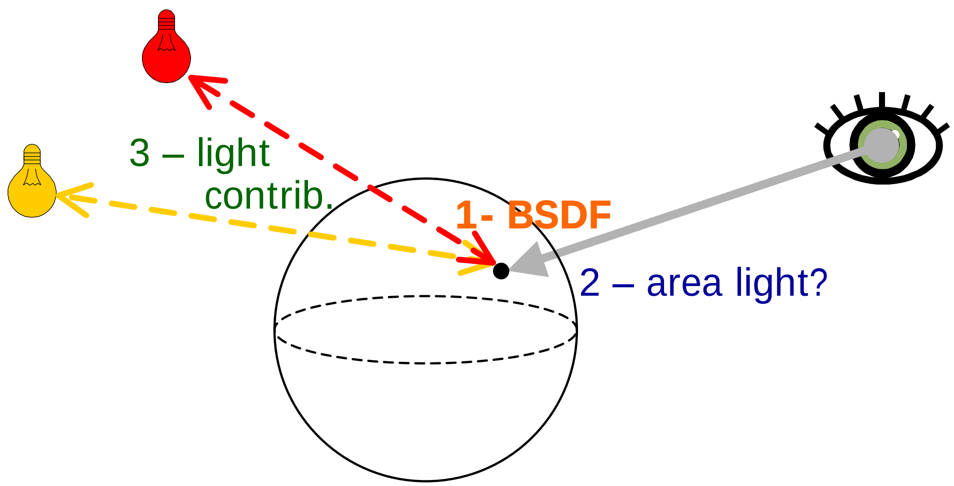


Whitted's

```
<Compute emitted and reflected light at ray intersection point>
    <Evaluate BSDF at hit point>
    <Initialize common variables for Whitted integrator>
    <Compute emitted light if ray hit an area light source>
```

Basic concepts

- Obtaining ray colors

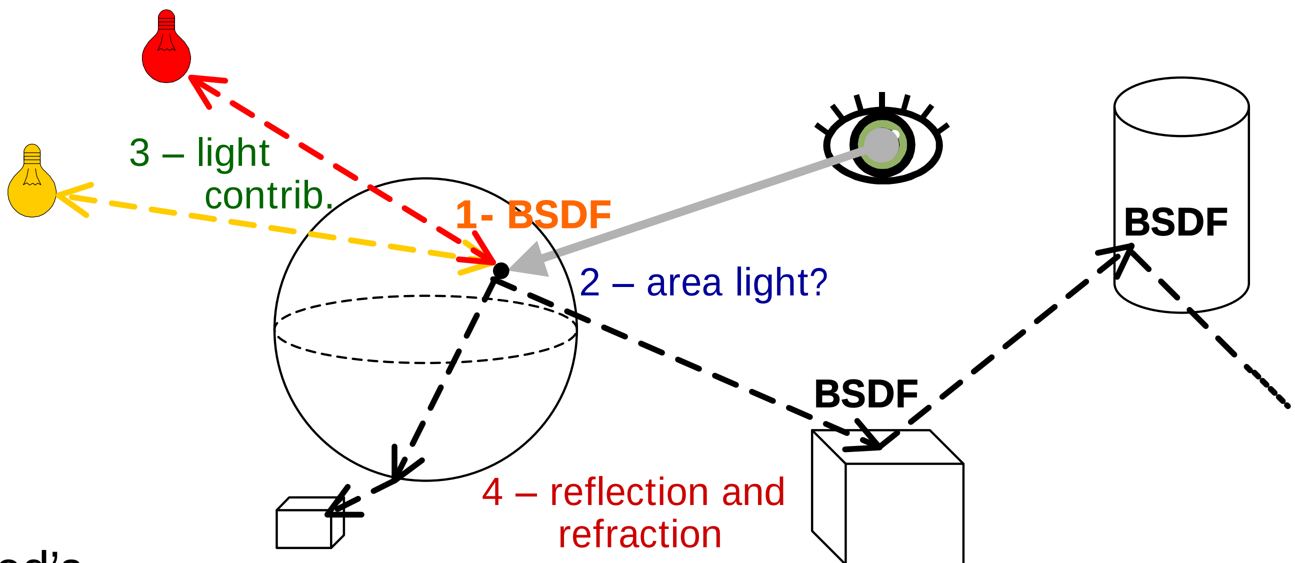


Whitted's

```
<Compute emitted and reflected light at ray intersection point>
    <Evaluate BSDF at hit point>
    <Initialize common variables for Whitted integrator>
    <Compute emitted light if ray hit an area light source>
    <Add contribution of each light source>
```

Basic concepts

- Obtaining ray colors

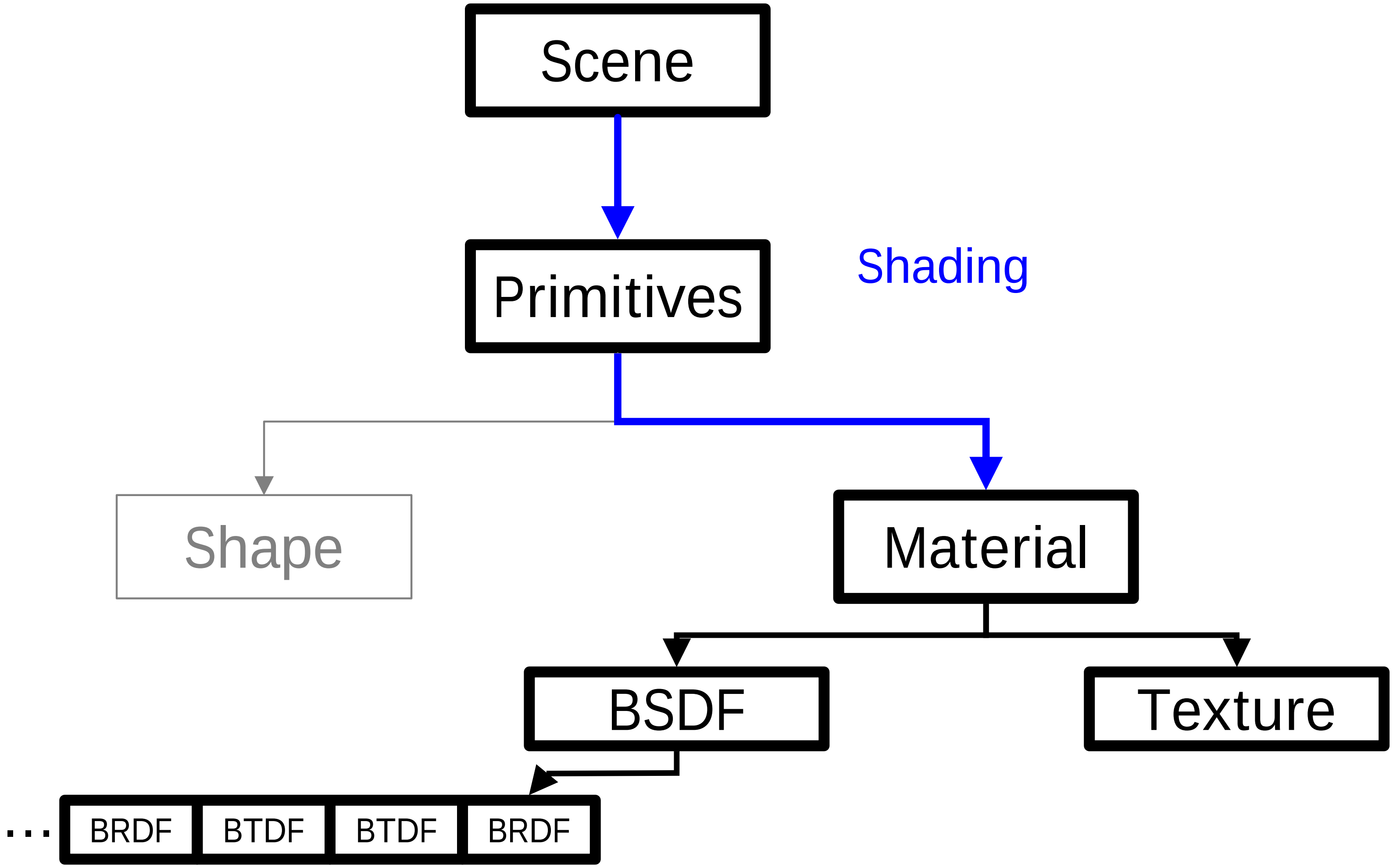


Whitted's

```
<Compute emitted and reflected light at ray intersection point>
    <Evaluate BSDF at hit point>
    <Initialize common variables for Whitted integrator>
    <Compute emitted light if ray hit an area light source>
    <Add contribution of each light source>
    if (rayDepth++ < maxDepth) {
          <Trace rays for specular reflection and refraction>
    }
    --rayDepth;
```

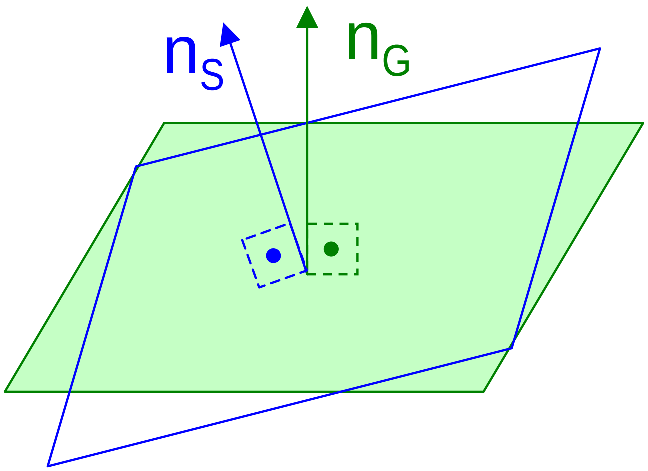
Basic concepts

- Obtaining ray colors



BSDF

- Parameters
  - DifferentialGeometry – shading dg
  - Geometric normal - n_G
  - Index of refraction
- Builds orthonormal coordinate system
  - Shading normal n_S



BSDF

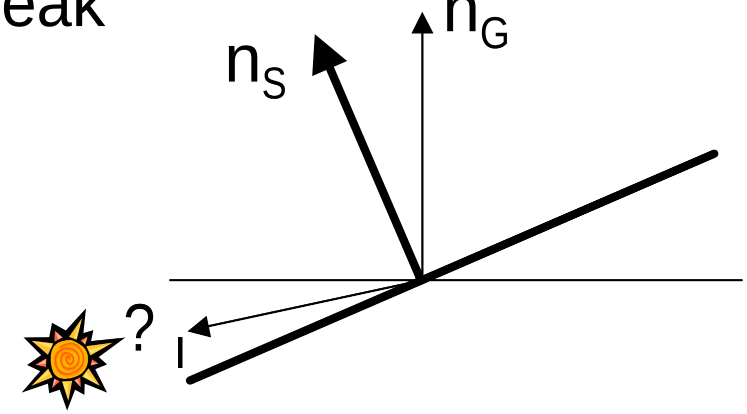
- Fixed maximum number of BxDFs
  - 8
  - Never needed more than that.
- Methods
  - Number of BxDFs Components
  - Normal equality
  - Coord. frames transformations

BSDF

- Problems with shading normals

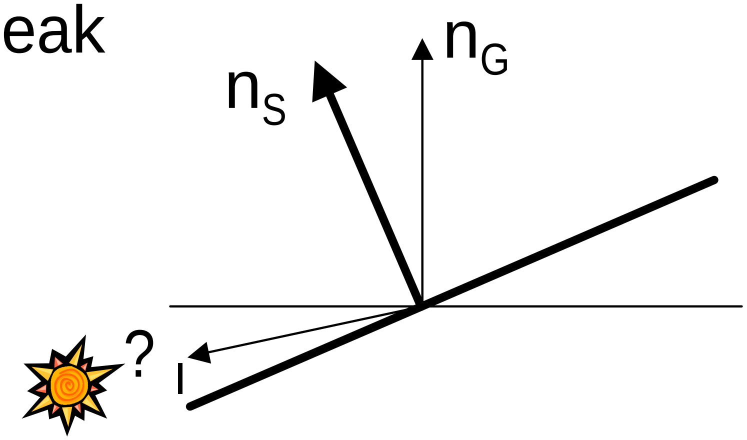
- Problems with shading normals

  - Light leak

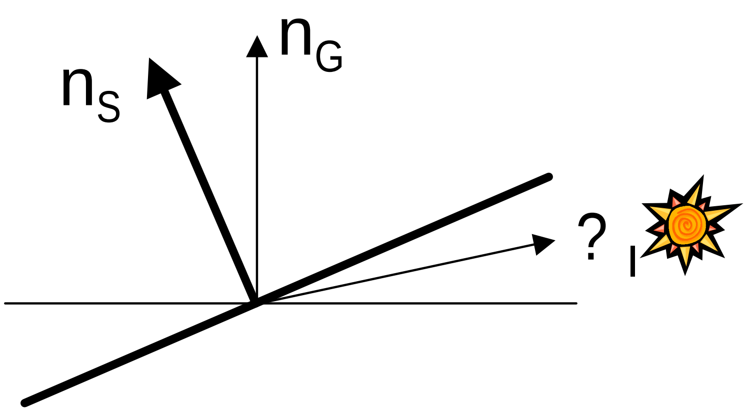


BSDF

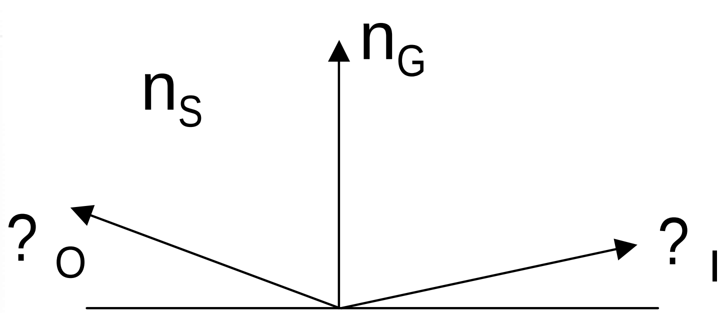
- Problems with shading normals
  - Light leak



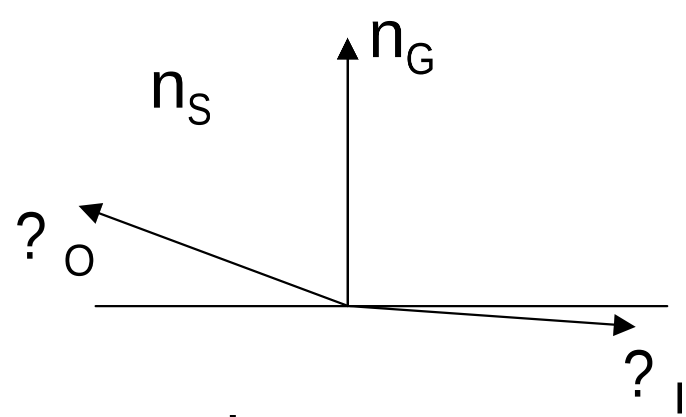
  - Black spots



- Solution
  - BRDF or BTDF?
    - Use geometric normal;

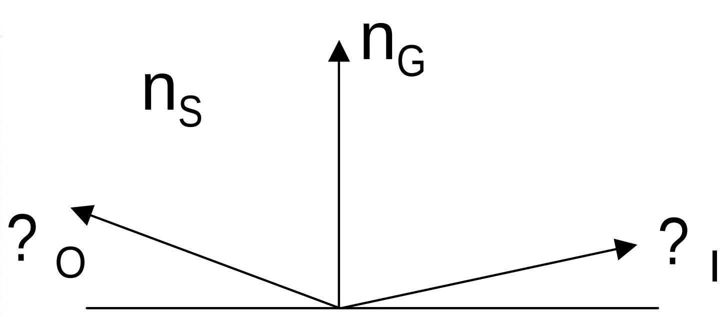


Evaluate BRDFs

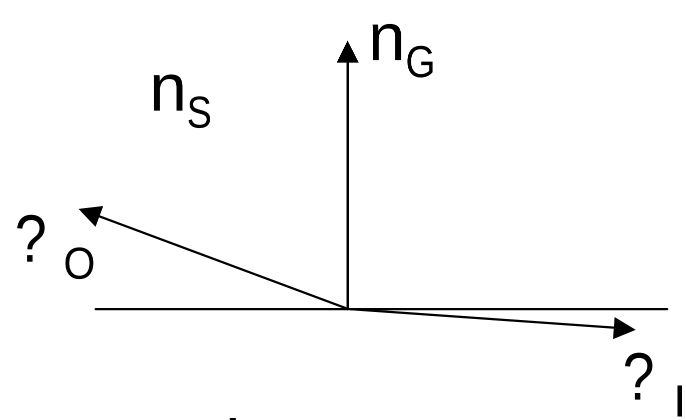


Evaluate BTDFs

- Solution
  - BRDF or BTDF?
    - Use geometric normal;



Evaluate BRDFs

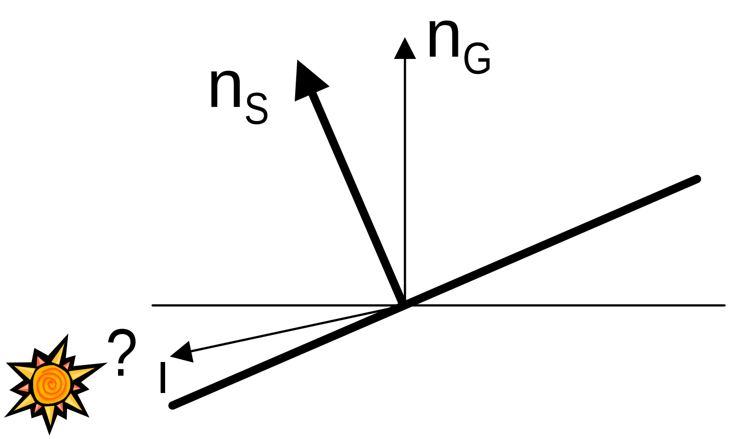


Evaluate BTDFs

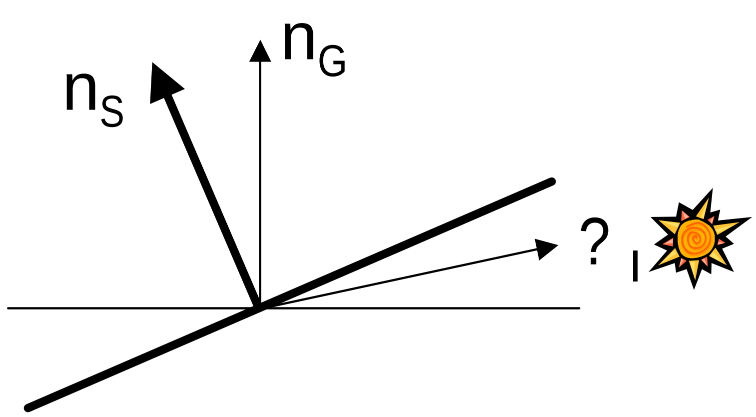
  - Scattering equation evaluation
    - Use shading normal

BSDF

- Solution



- Light leaks avoided
- Only BTDFs are considered



- Black spots avoided
- Only BRDFs are considered

BSDF

- Memmory management
  - Many BSDFs created during single ray cast
  - Performance issues with dynamic allocation

BSDF

- Memmory management
  - Many BSDFs created during single ray cast
  - Performance issues with dynamic allocation

  - How to avoid this?

BSDF

- Memmory management
  - Many BSDFs created during single ray cast
  - Performance issues with dynamic allocation

  - How to avoid this?
  - Previous memory allocation

BSDF

- Memmory management
  - Many BSDFs created during single ray cast
  - Performance issues with dynamic allocation

  - How to avoid this?
  - Previous memory allocation

  - For every single ray hit?

BSDF

- Memmory management
  - Many BSDFs created during single ray cast
  - Performance issues with dynamic allocation

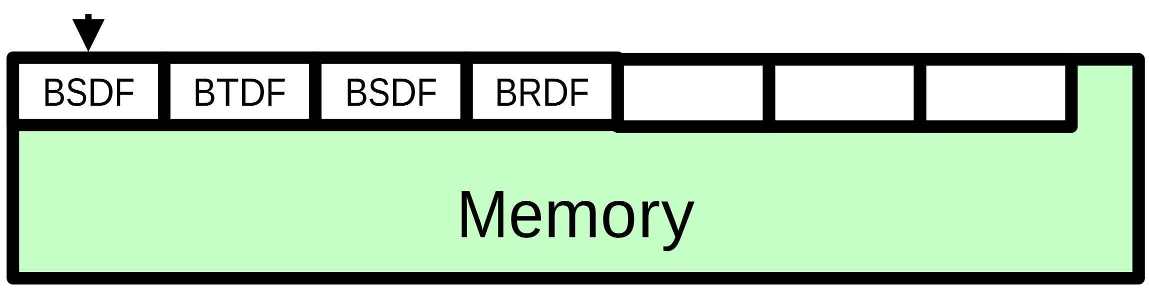
  - How to avoid this?
  - Previous memory allocation

  - For every single ray hit?
  - Reuse memory

BSDF

- Memory Arena
  - Static chunk of memory
  - All BxDFs for a ray are sequentialy saved there
  - Used and recycled at every ray tracing

```
Class BSDF {
        static Memory Arena
.
.
.
}
```



Materials

- GetBSDF method
  - Parameters
    - dgGeom – actual Differential Geometry
    - dgShading – perturbed shading geometry
  - Returns final shading geometry for point
    - BRDF
    - BTDF

Materials

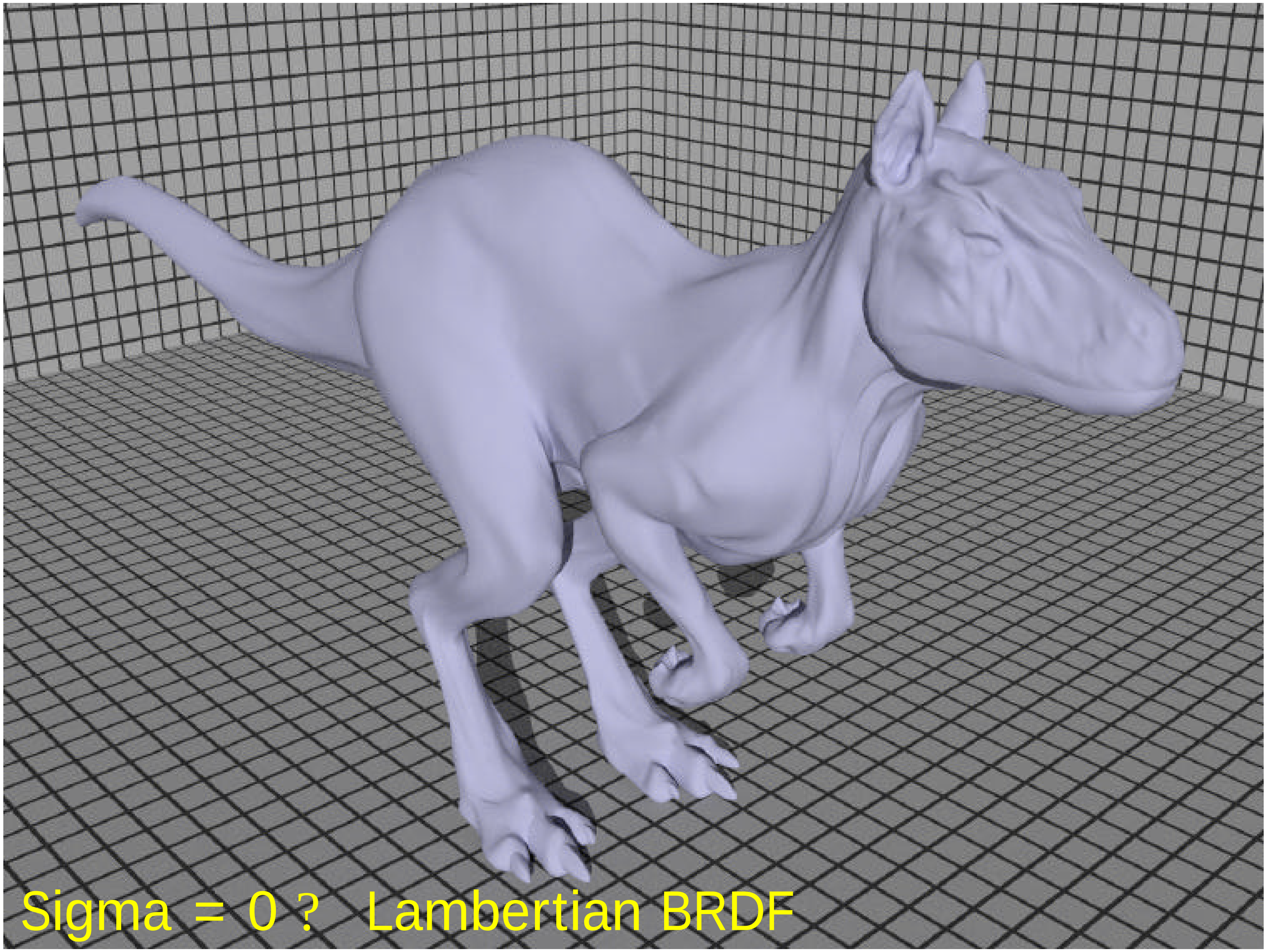
- Create access to BSDF in Intersection class
  - Intersection:: GetBSDF
    - dg.ComputeDifferentials (ray)
    - Primitive->getBSDF
    - Material->getBSDF

Materials

- Matte
  - Purely diffuse surface
  - Parameters
    - Spectral diffuse reflection – *Kd*
    - Scalar roughness value – *sigma*
    - Optional scalar texture – *bumpMap*

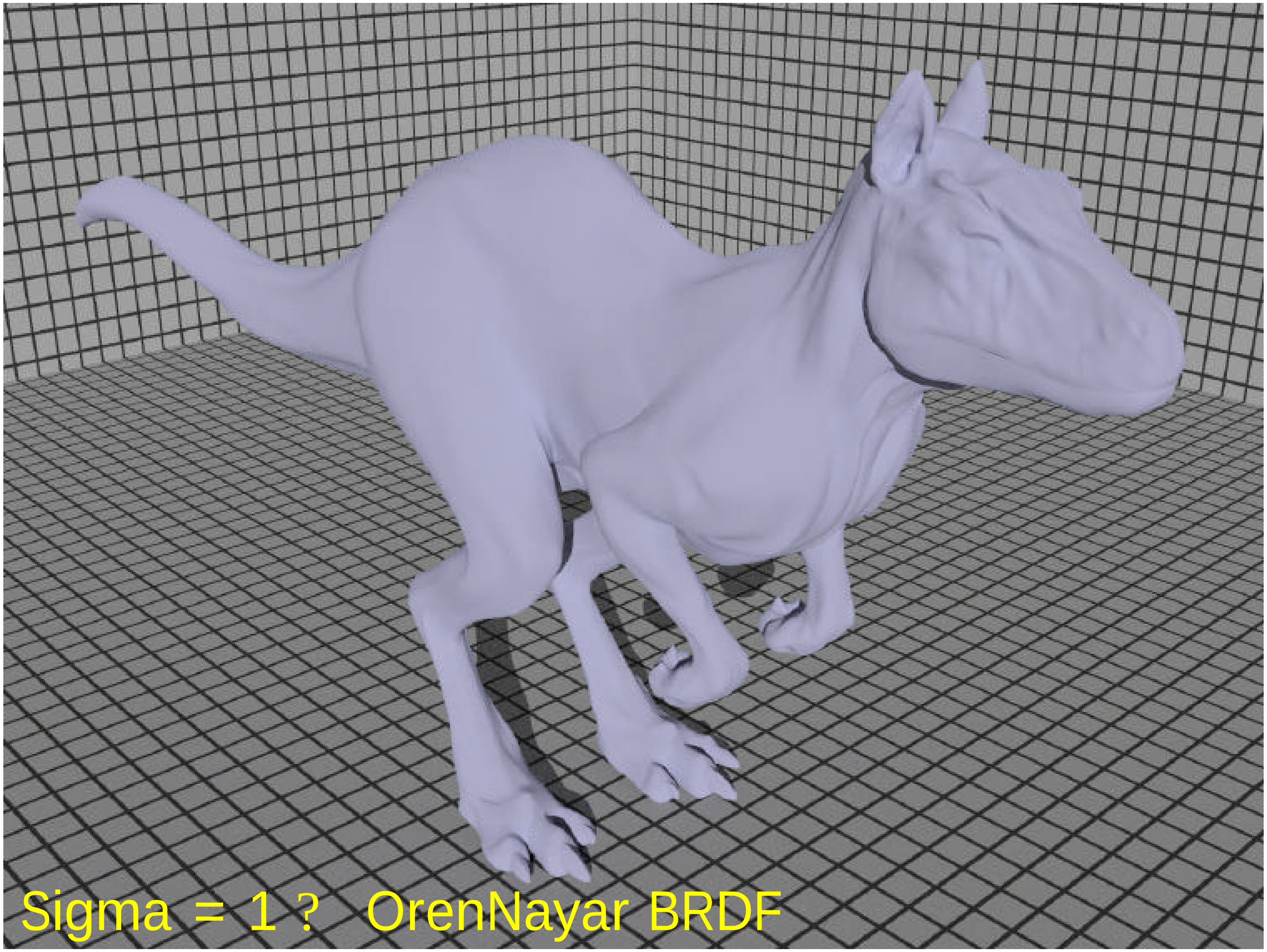
Materials

- Sigma variation example



Materials

- Sigma variation example



Materials

- Matte getBSDF method

```
BSDF *Matte::GetBSDF(const DifferentialGeometry &dgGeom,
        const DifferentialGeometry &dgShading) const {
   // Allocate _BSDF_, possibly doing bump-mapping with _bumpMap_
   DifferentialGeometry dgs;













   return bsdf;
}
```

Materials

- Matte getBSDF method

```
BSDF *Matte::GetBSDF(const DifferentialGeometry &dgGeom,
        const DifferentialGeometry &dgShading) const {
   // Allocate _BSDF_, possibly doing bump-mapping with _bumpMap_
   DifferentialGeometry dgs;

   if (bumpMap)
        Bump(bumpMap, dgGeom, dgShading, &dgs);
   else
        dgs = dgShading;

   return bsdf;
}
```

Calculates shading normal based on bump map

- Matte getBSDF method

```
BSDF *Matte::GetBSDF(const DifferentialGeometry &dgGeom,
        const DifferentialGeometry &dgShading) const {
    // Allocate _BSDF_, possibly doing bump-mapping with _bumpMap_
    DifferentialGeometry dgs;

    if (bumpMap)
        Bump(bumpMap, dgGeom, dgShading, &dgs);
    else
        dgs = dgShading;

    BSDF *bsdf = BSDF_ALLOC(BSDF)(dgs, dgGeom.nn);
```

Calculates shading normal based on bump map

Allocates the BSDF

```
    return bsdf;
}
```

- Matte getBSDF method

```
BSDF *Matte::GetBSDF(const DifferentialGeometry &dgGeom,
        const DifferentialGeometry &dgShading) const {
    // Allocate _BSDF_, possibly doing bump-mapping with _bumpMap_
    DifferentialGeometry dgs;

    if (bumpMap)
        Bump(bumpMap, dgGeom, dgShading, &dgs);
    else
        dgs = dgShading;

    BSDF *bsdf = BSDF_ALLOC(BSDF)(dgs, dgGeom.nn);

    Spectrum r = Kd->Evaluate(dgs).Clamp();

    return bsdf;
}
```

Calculates shading normal based on bump map

Allocates the BSDF

Texture evaluation; Obtention of reflection and roughness coefficients.

- Matte getBSDF method

```
BSDF *Matte::GetBSDF(const DifferentialGeometry &dgGeom,
        const DifferentialGeometry &dgShading) const {
    // Allocate _BSDF_, possibly doing bump-mapping with _bumpMap_
    DifferentialGeometry dgs;

    if (bumpMap)
        Bump(bumpMap, dgGeom, dgShading, &dgs);
    else
        dgs = dgShading;

    BSDF *bsdf = BSDF_ALLOC(BSDF)(dgs, dgGeom.nn);

    Spectrum r = Kd->Evaluate(dgs).Clamp();

    float sig = Clamp(sigma->Evaluate(dgs), 0.f, 90.f);
    if (sig == 0.)
        bsdf->Add(BSDF_ALLOC(Lambertian)(r));
    else
        bsdf->Add(BSDF_ALLOC(OrenNayar)(r, sig));

    return bsdf;
}
```

Calculates shading normal based on bump map

Allocates the BSDF

Texture evaluation; Obtention of reflection and roughness coefficients.

Allocates new BRDF according to sigma and Adds it to final BSDF

Materials

- Plastic
  - Mixture of diffuse and glossy surface
  - Parameters
    - Spectral diffuse reflection – *Kd*
    - Glossy specular reflection – *Ks*
    - Scalar roughness value – *roughness*
      - *Size of specular highlight*
    - Optional scalar texture – *bumpMap*

- Plastic getBSDF method

```
BSDF *Plastic::GetBSDF(const DifferentialGeometry &dgGeom,
        const DifferentialGeometry &dgShading) const {
    DifferentialGeometry dgs;
    if (bumpMap)
        Bump(bumpMap, dgGeom, dgShading, &dgs);
    else
        dgs = dgShading;

    BSDF *bsdf = BSDF_ALLOC(BSDF)(dgs, dgGeom.nn);

    Spectrum kd = Kd->Evaluate(dgs).Clamp();
    BxDF *diff = BSDF_ALLOC(Lambertian)(kd);
    Fresnel *fresnel =
    BSDF_ALLOC(FresnelDielectric)(1.5f, 1.f);
    bsdf->Add(diff);

    Spectrum ks = Ks->Evaluate(dgs).Clamp();
    float rough = roughness->Evaluate(dgs);
    BxDF *spec = BSDF_ALLOC(Microfacet)(ks, fresnel,
    BSDF_ALLOC(Blinn)(1.f / rough));
    bsdf->Add(spec);

    return bsdf;
}
```

Calculates shading normal based on bump map

Allocates the BSDF

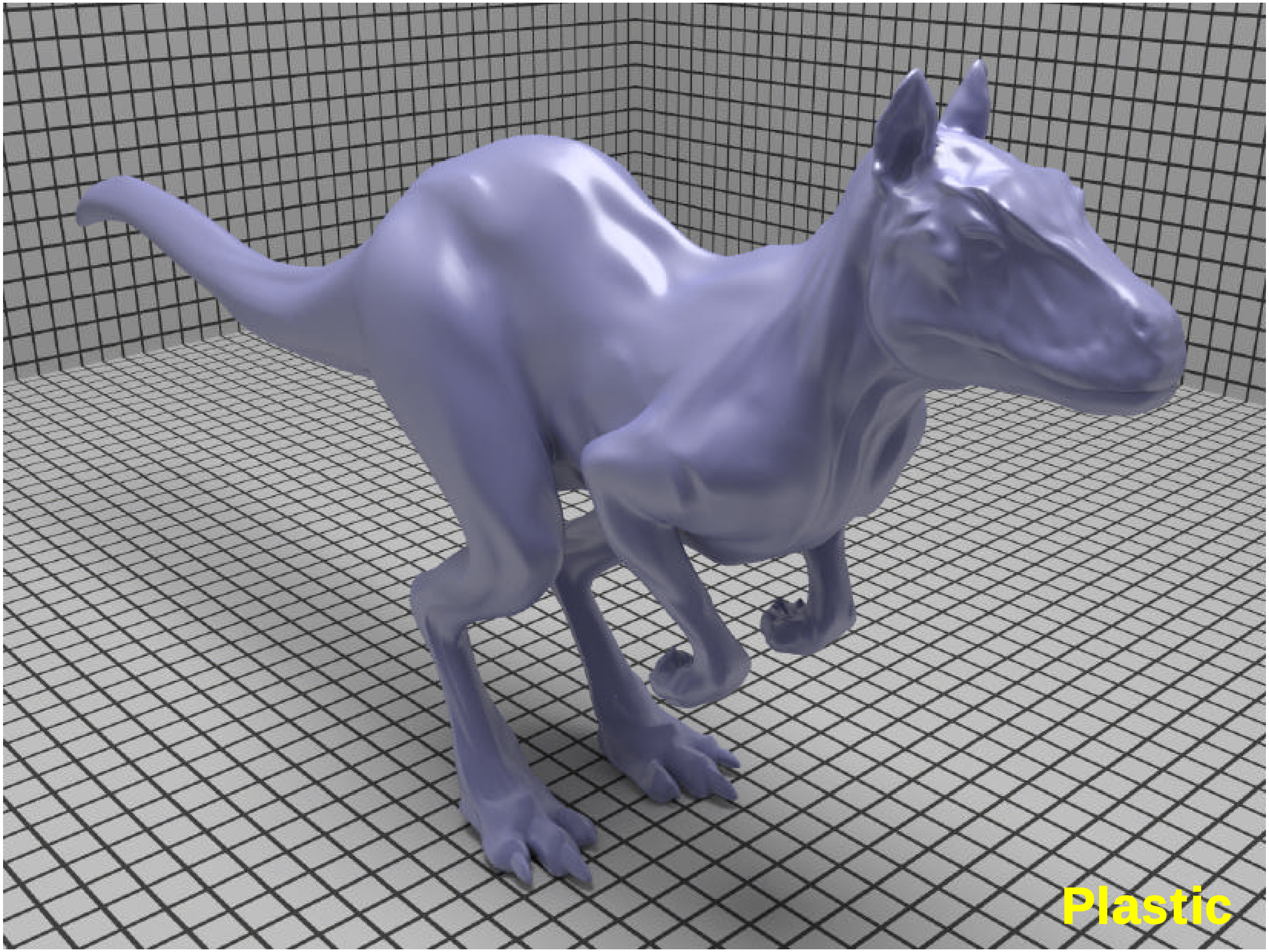
Texture reflection evaluation;

Allocates reflection BRDF according to sigma and Adds it to final BSDF

Texture glossy evaluation;

Allocates glossy BRDF according to sigma and Adds it to final BSDF

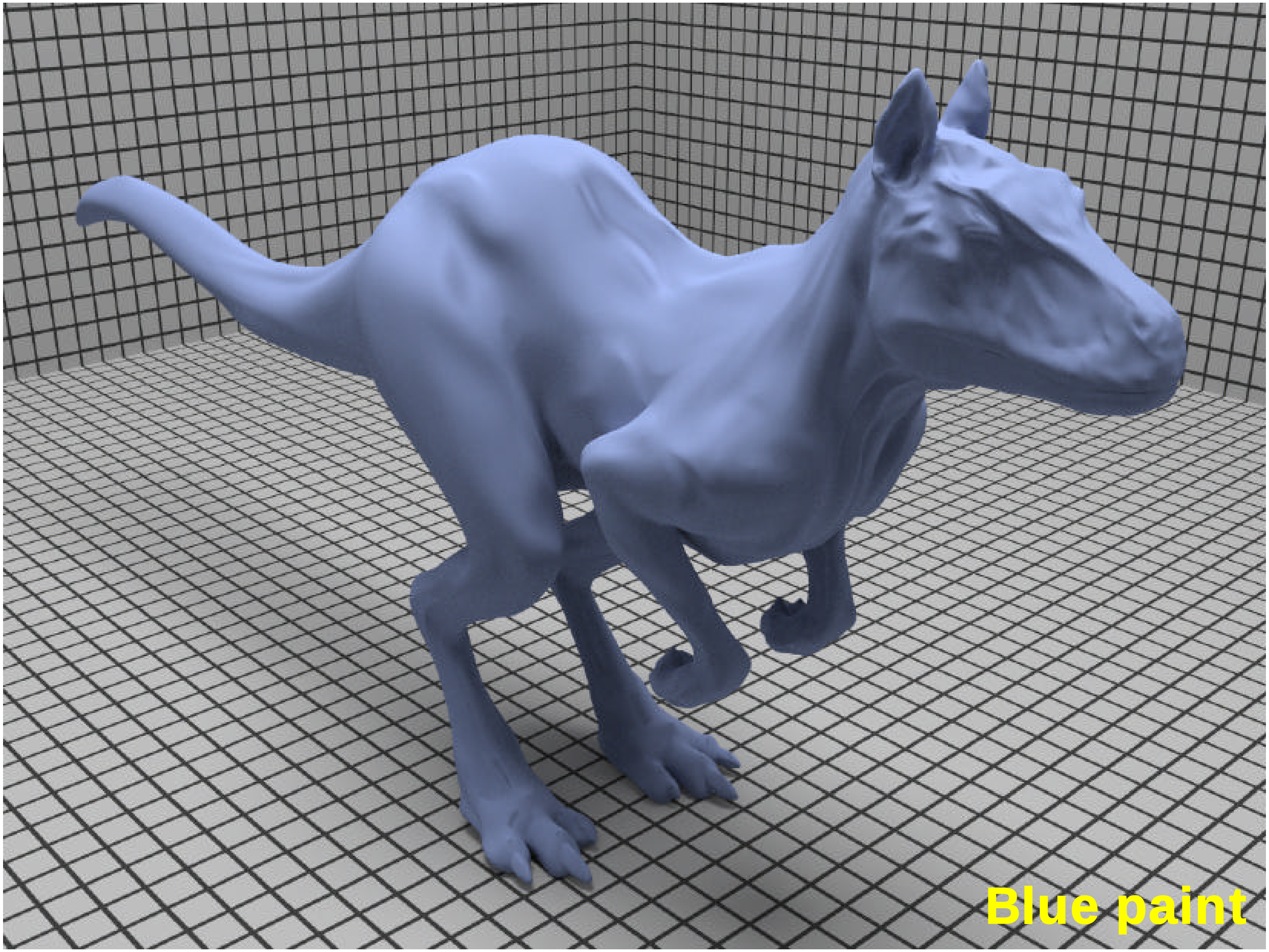
Materials



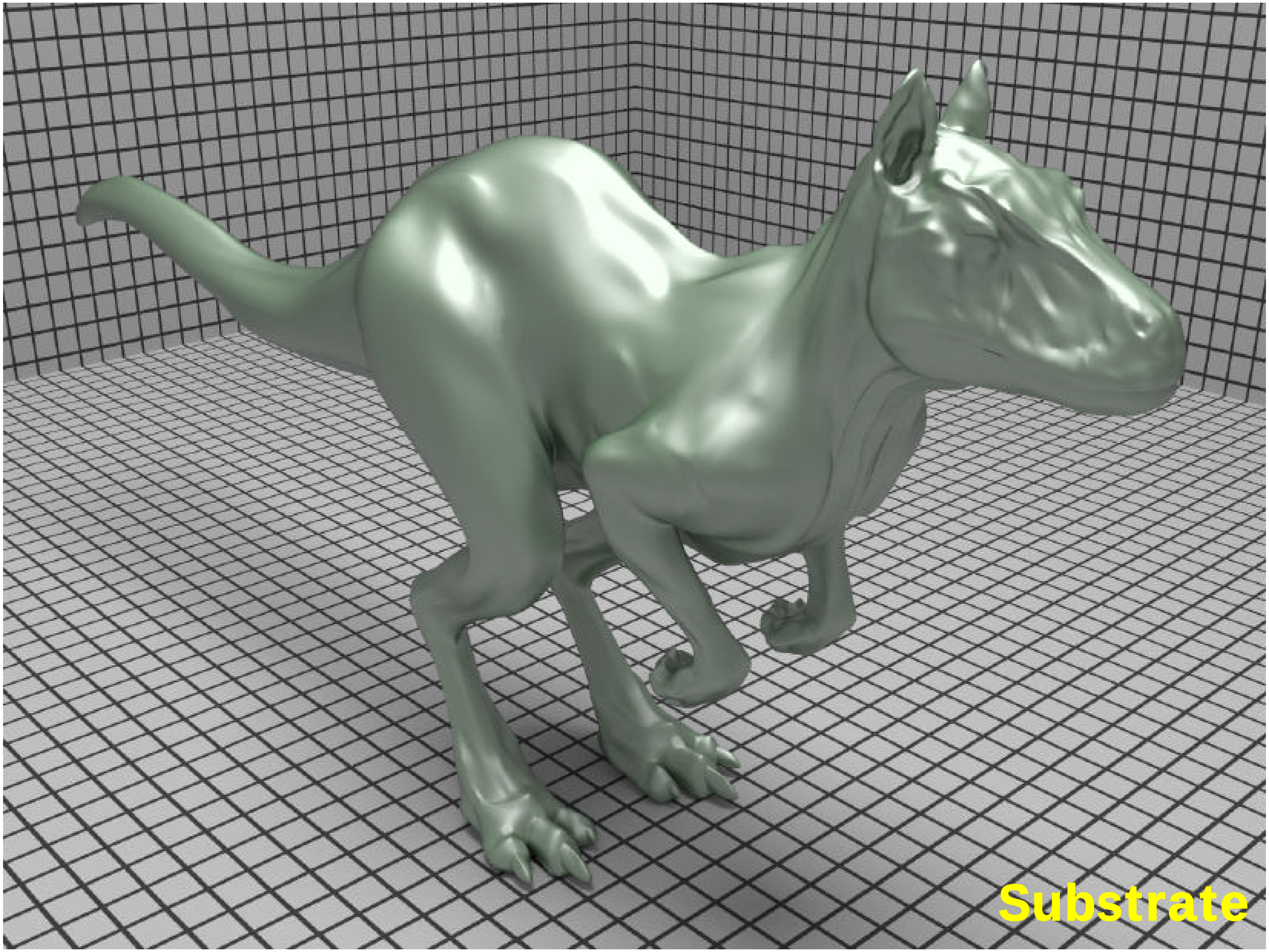
Materials

- Other materials
  - Translucent
  - Mirror
  - Glass
  - ShinyMetal
  - Substrate
  - Clay
  - Felt
  - Primer
  - Skin
  - BluePaint
  - Uber

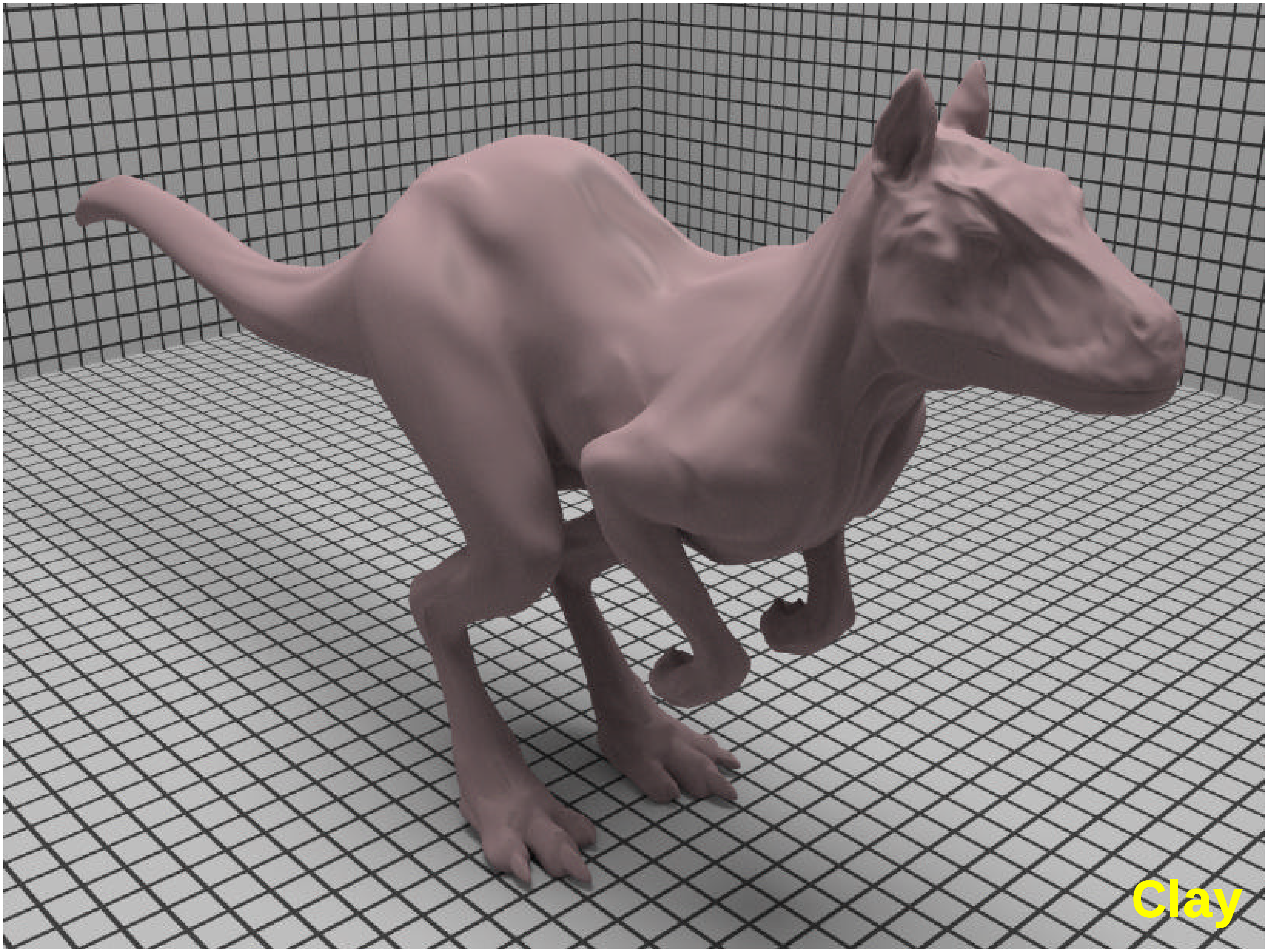
Materials



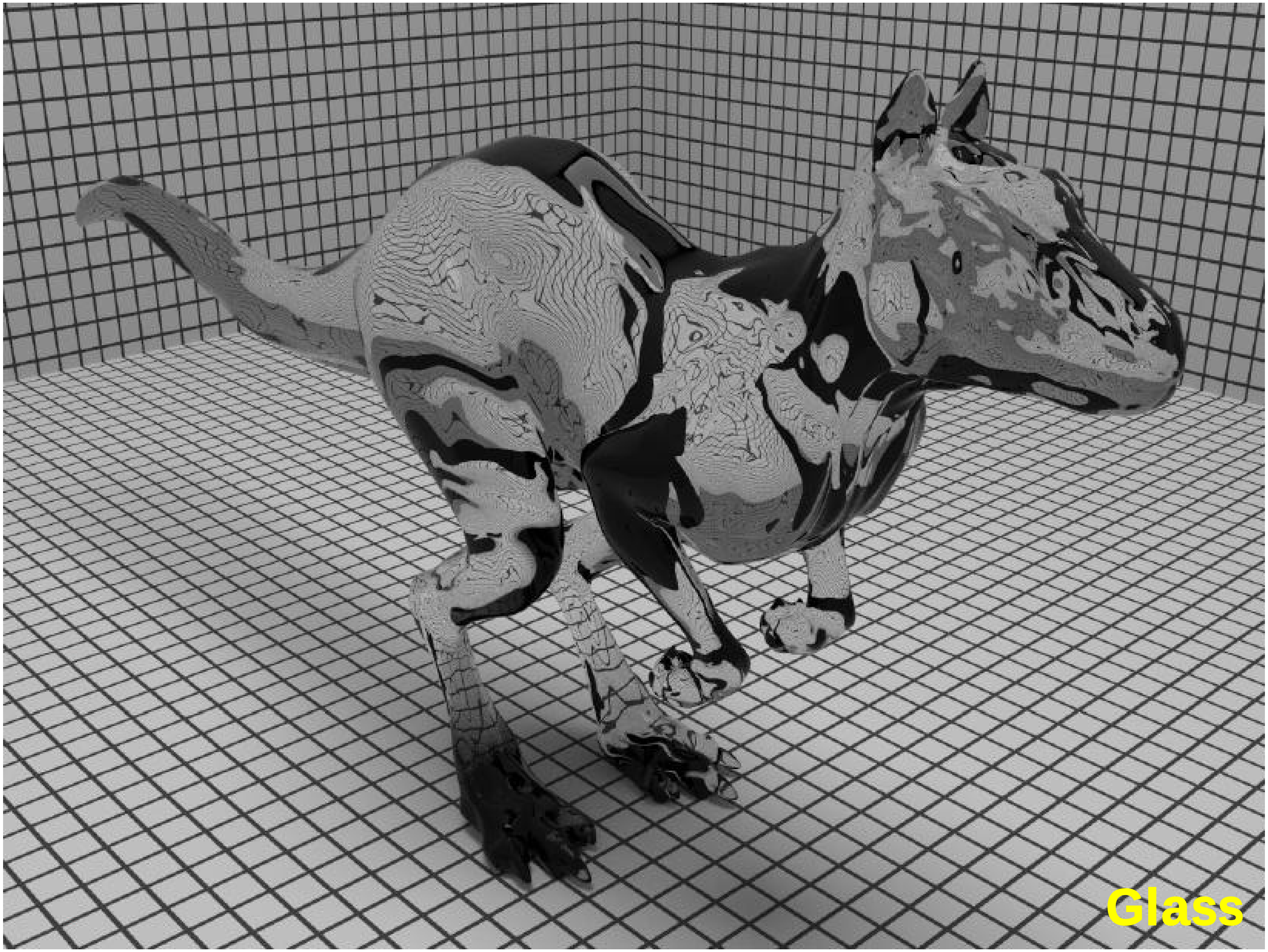
Materials



Materials



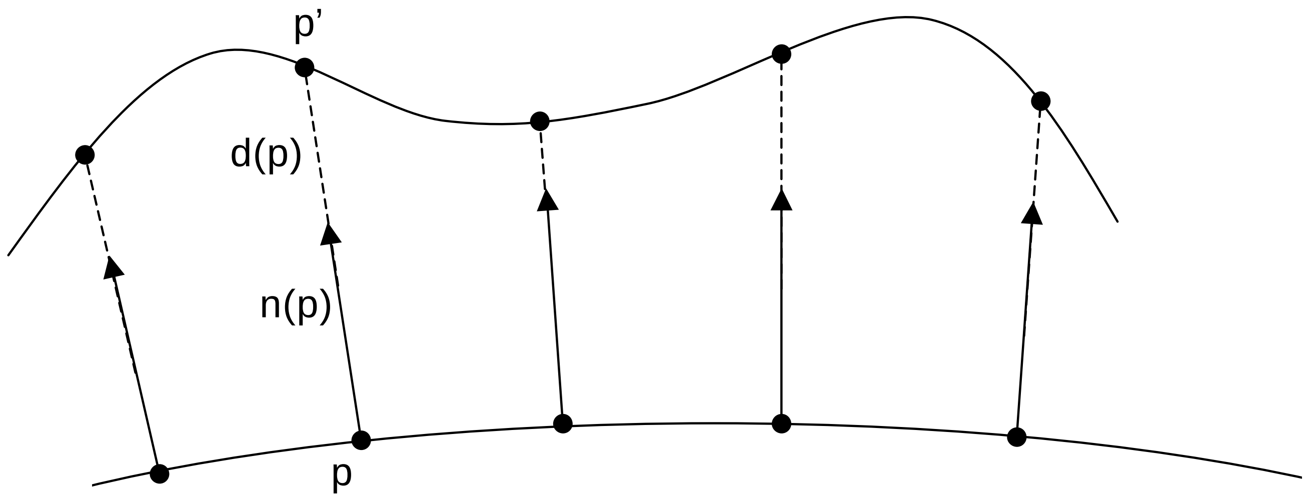
Materials



Bump Mapping

- Displacement simulation to points

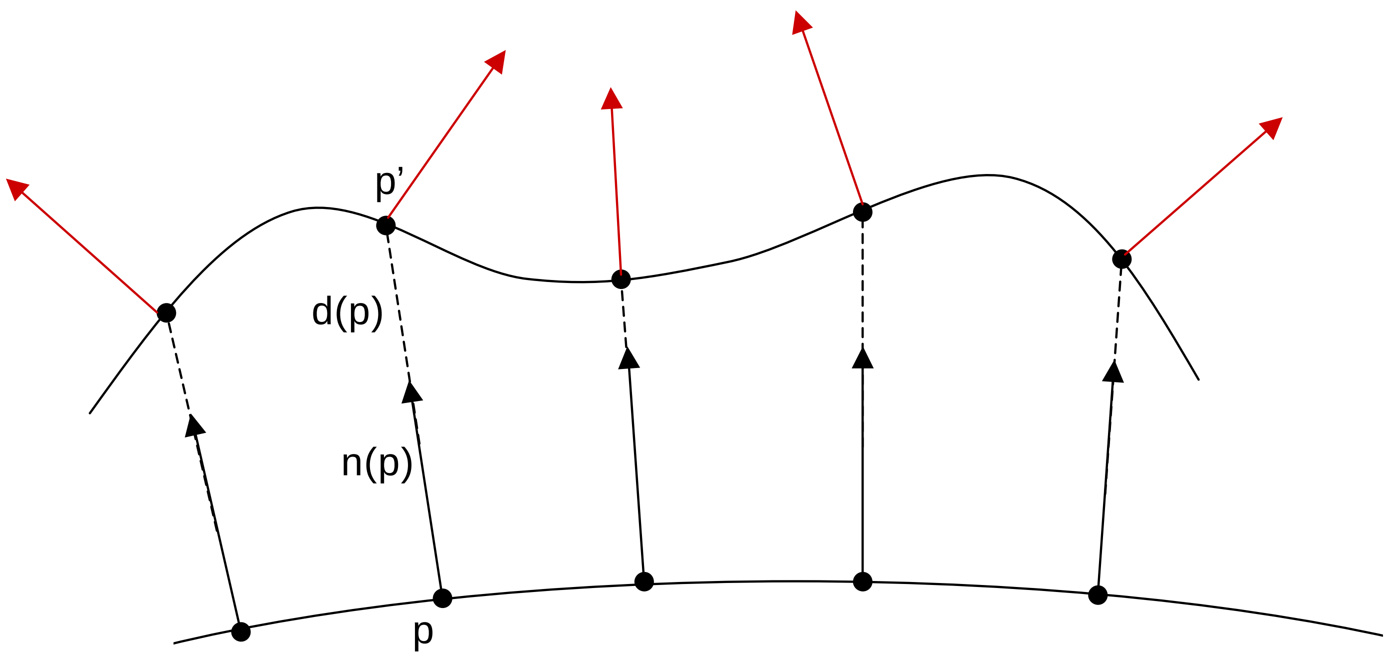
$$p'(u,v) = p(u,v) + d(u,v)n(u,v)$$



Bump Mapping

- Displacement simulation to points

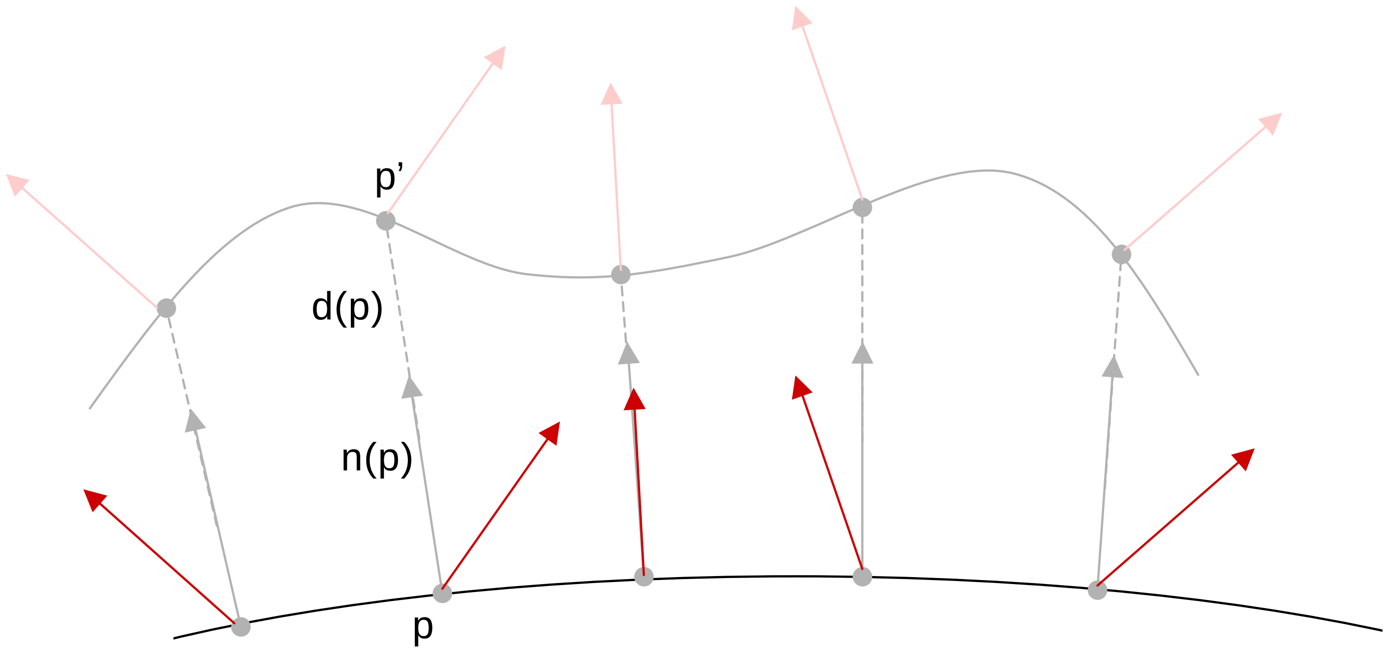
$$p'(u,v) = p(u,v) + d(u,v)n(u,v)$$



Bump Mapping

- Displacement simulation to points

$$p'(u,v) = p(u,v) + d(u,v)n(u,v)$$



Bump Mapping

$$n = \frac{\partial p}{\partial u} \times \frac{\partial p}{\partial v}$$

- Change the partial derivatives of p to change normal for p'

$$p'(u,v) = p(u,v) + d(u,v)n(u,v)$$

- Derivation using the product and chain rules

$$\frac{\partial p'}{\partial u} = \frac{\partial p}{\partial u} + \frac{\partial d}{\partial u} n + d \frac{\partial n}{\partial u}$$

Product rule

- D[u*v] = u*D[v] +v*D[u]

Chain rule

- $h(t) = f(g(t))$.
- $dh(t) = \mathrm{d}f(g(t))\mathrm{d}g(t)$.

Bump Mapping

- By the definition of partial derivative

$$\frac{\partial d(u,v)}{\partial u} = \lim_{\Delta u \to 0} \frac{d(u+\Delta u, v) - d(u,v)}{\Delta u}$$

- For small ?u, we have that

$$\frac{\partial p'}{\partial u} = \frac{\partial p}{\partial u} + \frac{d(u+\Delta u, v) - d(u,v)}{\Delta u} n + d \frac{\partial n}{\partial u}$$

Bump Mapping

- By the definition of partial derivative

$$\frac{\partial d(u,v)}{\partial u} = \lim_{\Delta u \to 0} \frac{d(u+\Delta u, v) - d(u,v)}{\Delta u}$$

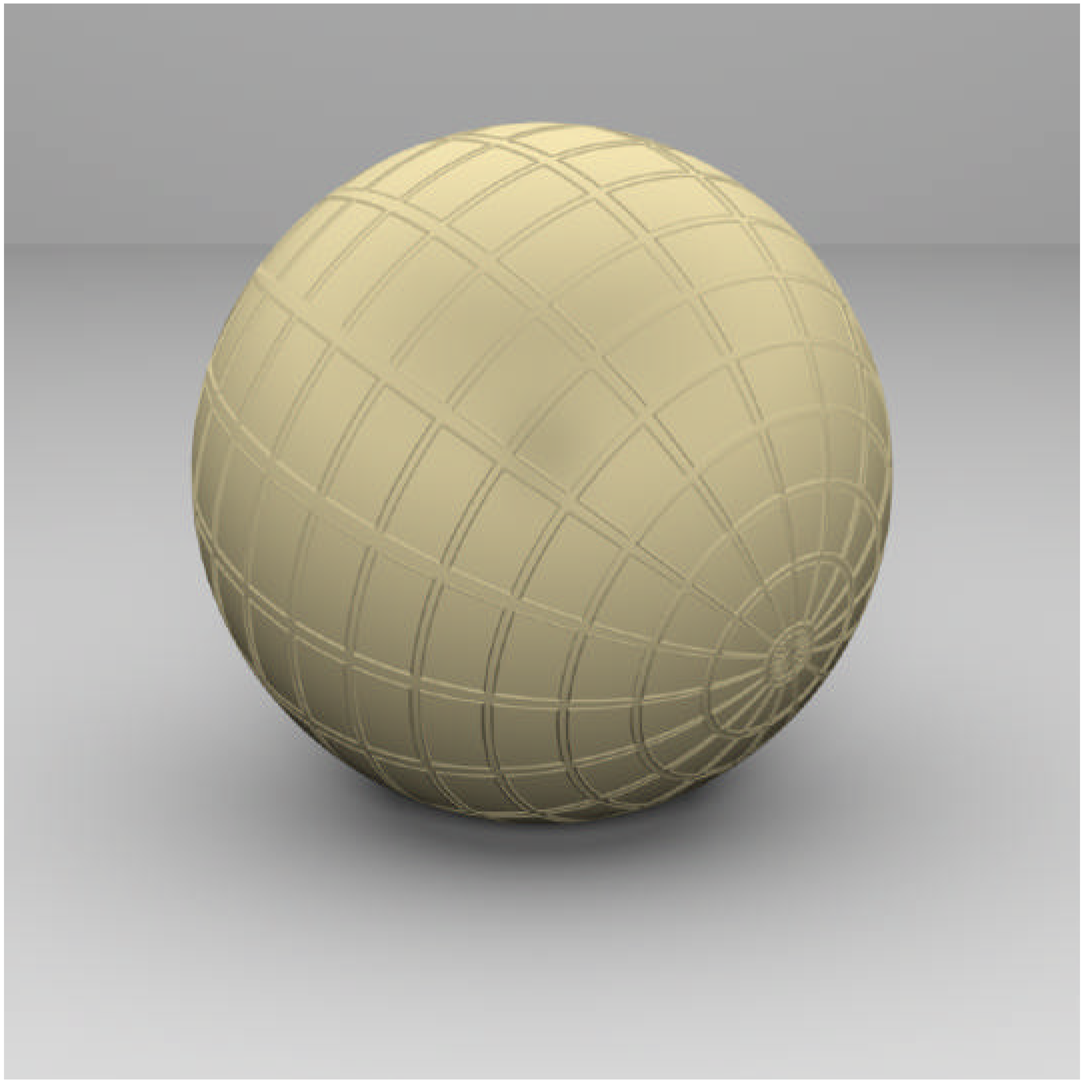
- For small ?u, we have that

$$\frac{\partial p'}{\partial u} = \frac{\partial p}{\partial u} + \frac{d(u+\Delta u, v) - d(u,v)}{\Delta u} n + d\frac{\partial n}{\partial u}$$

$$\frac{\partial p'}{\partial v} = \frac{\partial p}{\partial v} + \frac{d(u, v+\Delta v) - d(u,v)}{\Delta v} n + d\frac{\partial n}{\partial v}$$

Bump Mapping

- Effect



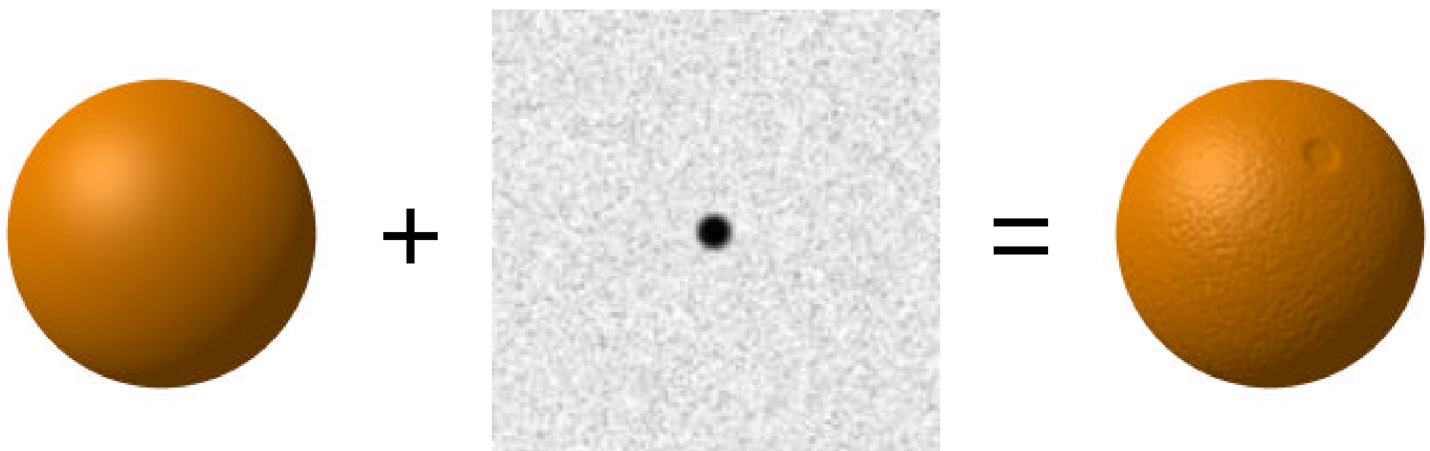
Bump Mapping

- Effect

Bum Map

Bump Mapping

- Effect
  - http://en.wikipedia.org/wiki/Bump_mapping

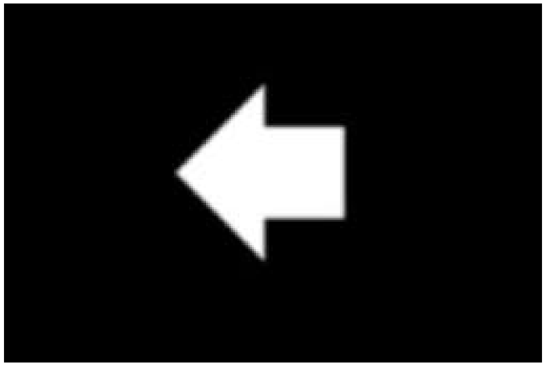


Bump Mapping

- Advantages
  - Nice depth effects
  - Easy to implement
  - Reasonably fast performance
- Disadvantages
  - No real *p'* is created
  - Does not affect objects surface
    - Does not affect shadow casting process
    - Does not affect objects edges visualization

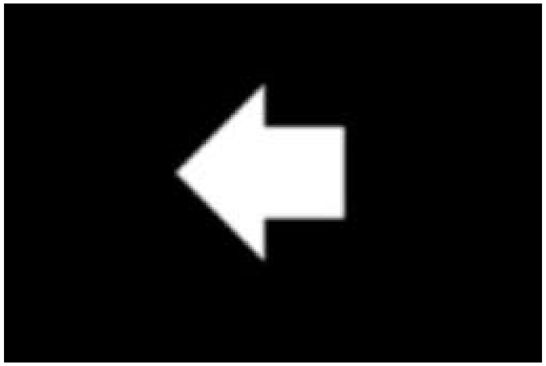
Bump Mapping

- Effect
  - http://www.sanedraw.com/LEARN/OVERVIEW/OV150MAC/INDEX.HTM



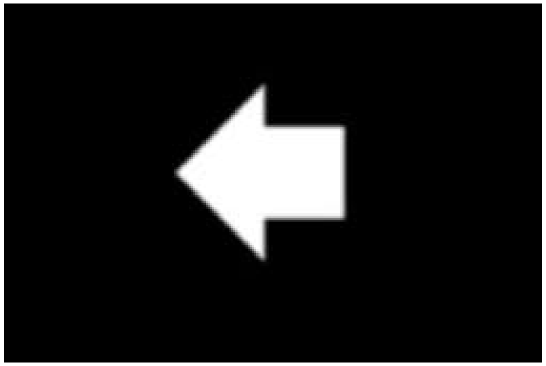
Bump Mapping

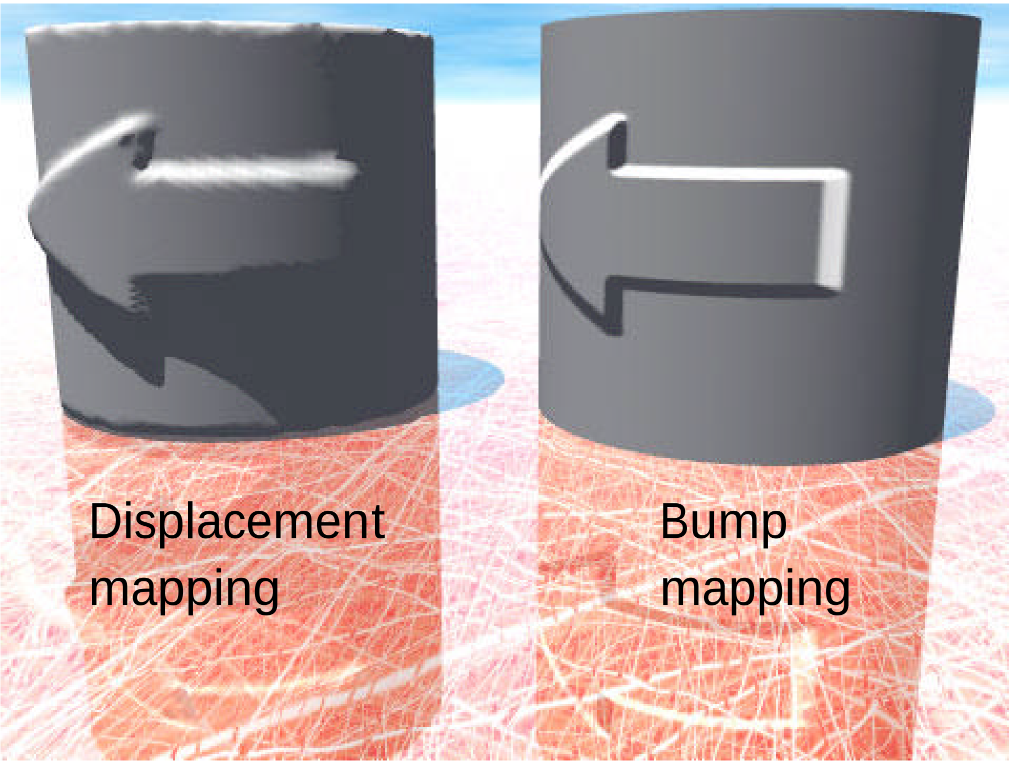
- Effect
  - http://www.sanedraw.com/LEARN/OVERVIEW/OV150MAC/INDEX.HTM



Bump Mapping

- Effect
  - http://www.sanedraw.com/LEARN/OVERVIEW/OV150MAC/INDEX.HTM





?

CS 563 Advanced Topics in Computer Graphics
Materials

by Paulo Gonçalves de Barros