mental ray physical subsurface scattering tutorials

Document version 1.05
Dec 14, 2004

Table of Contents

Chapter 1

Physical subsurface scattering, milk tutorial

1.1 Before you begin

In order to render correctly the scene files included with the milk tutorial you must point to the required files on your disk/local network:

- point to the correct shader libraries and declaration files: `physics`, `base` and `subsurface`
- point to the file texture `wood.iff`

Please modify all the scene files in your local folder

Once all shader libraries are linked and included correctly you're ready to render the .mi file with the `ray` command:

- `ray -v on -x on -imgpipe 1 filename.mi | imf_disp -`

Where:

- `-v` on means progress messages verbosity
- `-x` on means colored warning/error messages
- `-imgpipe 1 ... | imf_disp -` pipes the output to mental ray image viewer

While rendering image will automatically show up tile-by-tile in `imf_disp`

For more options invoke `ray` help: `ray -h`

1.2 Intro

This document describes, in an easy-to-follow tutorial form, how to use the mental images subsurface scattering shader 'misss_physical' with mental ray standalone in order to simulate the appearance of a highly scattering material (in the case of this tutorial, milk).

Note that mental ray comes with shader documentation, which serves as a complete reference for the physical subsurface scattering shaders used in this tutorial. The subsurface scattering shader documentation is located in the last section following the base, physical, and contour shader documentation sections.

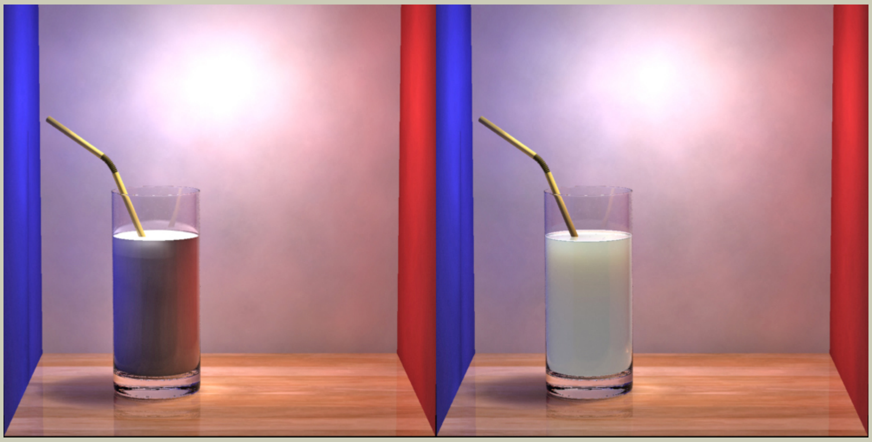
In order to be application independent the tutorial was made for mental ray standalone although the shader works on all OEM application which run mental ray core version 3.3 or later.

You are required to have:

- mental ray standalone 3.3 or later
- the misss_physical shader, from subsurface.DSO library and subsurface.mi declaration file
- public mental images 'base' and 'physics' shader libraries

Shown below are two images:

1. in the first image the milk is rendered with the DGS shader (dgs_material, dgs_material_photon)
2. the second image was rendered using the mental images subsurface scattering shader (misss_physical)


Left: No subsurface scattering vs. Right: Subsurface scattering

The intermediate steps taken to arrive at Image.2 from Image.1 are the body of this tutorial.

1.3 Options block, scene database and shader assignment

The options block requires :

- trace on [1]
- caustics on
- globillum on

A brief background on what the scene database consist of:

- a glass, a revolved closed NURBS surface
- the milk, another revolved closed surface. It derives from a sub-section of the glass curve, then extended to create the milk top surface, including a bit of surface tension border. Milk is then shrunk by1% to avoid inter penetration (the scale pivot is centered on the milk surface prior scaling down). It is important to model correctly a single closed surface in order for photons to be stored in a definite volume.
- a straw object, another revolved closed NURBS surface (then deformed)
- a cornell box object, a poly cube object
- a camera, focal length 110 mm

[1] *raytracing* must be on when using misss_physical* shaders, this differs from the misss_fast* shaders which work fine also with the *scanline* algorithm

- a pointlight, standing 1mm below the ceiling of the box

Please keep in mind that the entire scene was modeled in centimeter units of length.

Shaders assignment:

- glass: dielectric_material and dielectric_material_photon shaders
- milk in Image.1: dgs_material and dgs_material_photon shaders
- milk in Image.2: misss_physical material and photon shader
- straw: dgs_material and dgs_material_photon shaders
- box:
  - white ceiling and back wall, blue wall, red wall: dgs_material and dgs_material_photon shaders
  - floor: textured dgs_material and dgs_material_photon shaders
- light: mib_physical_light
  - 250000 caustics photons
  - 250000 GI photons
  - intensity value 3500

So, the very same setup was used to produce both images, but instead of dgs_material and dgs_material_photon, the subsurface scattering shader misss_physical, has been used in both the material and photon shaders slots for the milk object in Image.2 (remember to lightlink the pointlight to misss_physical).

Let's now see how to fine tune the misss_physical shader on a step-by-step basis. The steps used to go from Image.1a (see below) to Image.2 illustrate the process of subsurface scattering parameter selection and variation. In the following images, only the milk material has changed, unless explicitly stated.

1.4 Step 1 - optimized scene with non-tweaked SSS parameters

Let's substitute the dgs_material and dgs_material_photon shader with the misss_physical shader:

- apply misss_physical both as material and photon shader [2]

[2] alternatively you can use the misss_physical_phen applied only to the material port: it will automatically setup the connection to the photon one

- remember to specify the light in the light_array

These three were the only 'connections' needed, I use the term connection because they represent links in the shading graph. From now on we are going to tweak 'only' shader parameters.

The following image - Image.2b - was rendered using the following table of parameters of misss_physical, this is the scene you are going to use as a starting point for the tutorial.

The scene features also lower quality parameters such as less photons and lower antialiasing which will allow faster rendertime while keeping the effect a good preview.

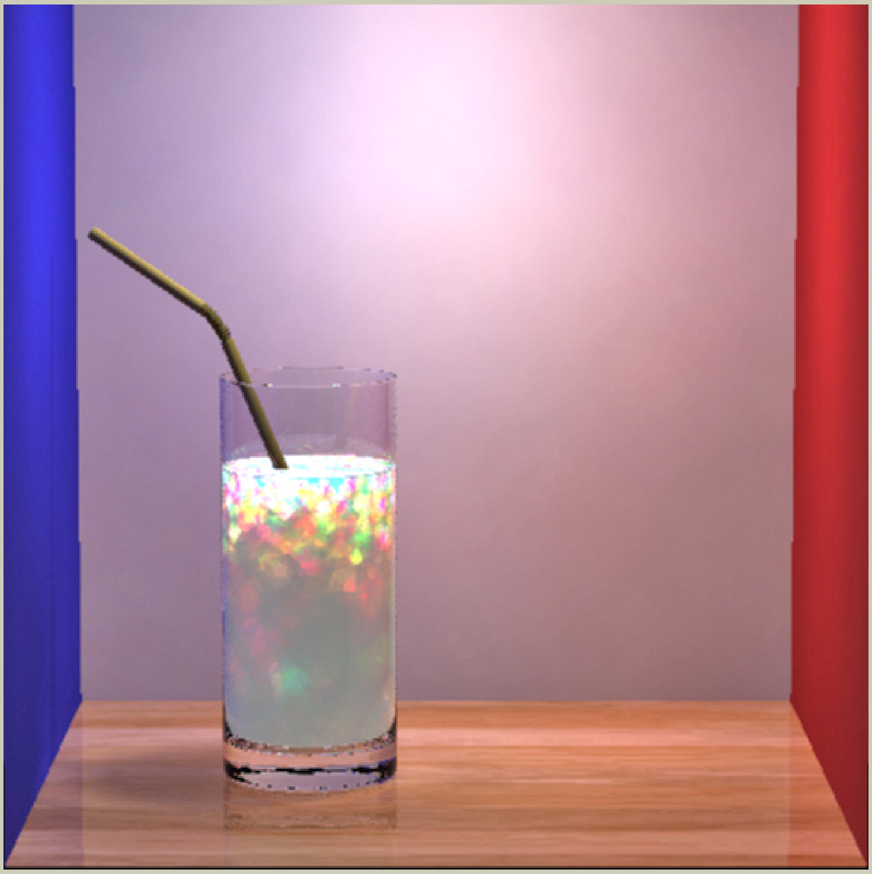


Image 1a

```
shader "mix_subsurf1"
"misss_physical" (
    "material" 0.5 0.5 0.5 1.,
    "transmission" 0.5 0.5 0.5 1.,
    "ior" 1.3,
    "absorption_coeff" 0.0014 0.0025 0.0142,
    "scattering_coeff" 0.7 1.22 1.9,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
    "depth" 10.,
    "max_samples" 1,
    "max_photons" 10,
    "max_radius" 10.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
        "pointLight1"
    ]
    )

options "miDefaultOptions"
    caustic 0
    caustic filter box 1.1
    caustic accuracy 1000 1
    globillum on
    globillum 0
    globillum accuracy 1000 8.
```

These settings are ”not good on purpose”, when you'll have some more experience with the misss_physical shader, you'll know what to use prior rendering.

Here we have specified an average grey color for material and transmission, while milk's ior and both scattering and absorption coefficients are derived experimentally.

An important parameter is scale_conversion: using the correct units of length measurement is critical for subsurface scattering. The scattering and absorption coefficients are experimentally derived and are commonly available for many substances. In this case, the coefficients are in units of inverse millimeter[3]. The other parameter related to size, max_radius, must be in similar units. Since the scene was modeled in centimeters, the scale_conversion is set to 10 to compensate, i.e., there are 10 millimeters per centimeter.

We are assuming milk is a forward scattering material, hence we are using scattering_anisotropy = 0.75 while we are using low values for depth, max_samples, max_photons and max_radius as a starting point.

We are also reducing caustic accuracy in the 'miDefaultOptions' scene options block.

[3]The units for the scattering coefficient us and the absorption coefficient ua are inverse length, such that the product of the coefficient and a distance L is dimensionless, where L is a photon's path length of travel without interaction through a medium. The probability P of a photon transmission of length L without absorption or scattering is $P = e^-(ua + us)L$

1.5 Step 2 - photon scattering

Image.1b illustrates two useful diagnostics.

First the colored spots are bundles of 10 photons sampled from within a radius of one millimeter (as you can see max_radius = 1, its values are always mm). These photons are not the ones in the caustic/GI photon map, but are the ones returned in the environment by the misss_physical shader. Using small max_photons and max_radius shows how photons are distributed. There areless photons on the bottom which is expected since the light shining from the top doesn't hit there and especially bottom-left of the glass, which is to be expected as it is in shadow.

Second, the white spots are bundles of 1 caustic photons (this time from the photon map) sampled from within a radius of one millimeter (the reflective floor is also generating caustics - this could be avoided marking the floor as caustic/GI receiver only).

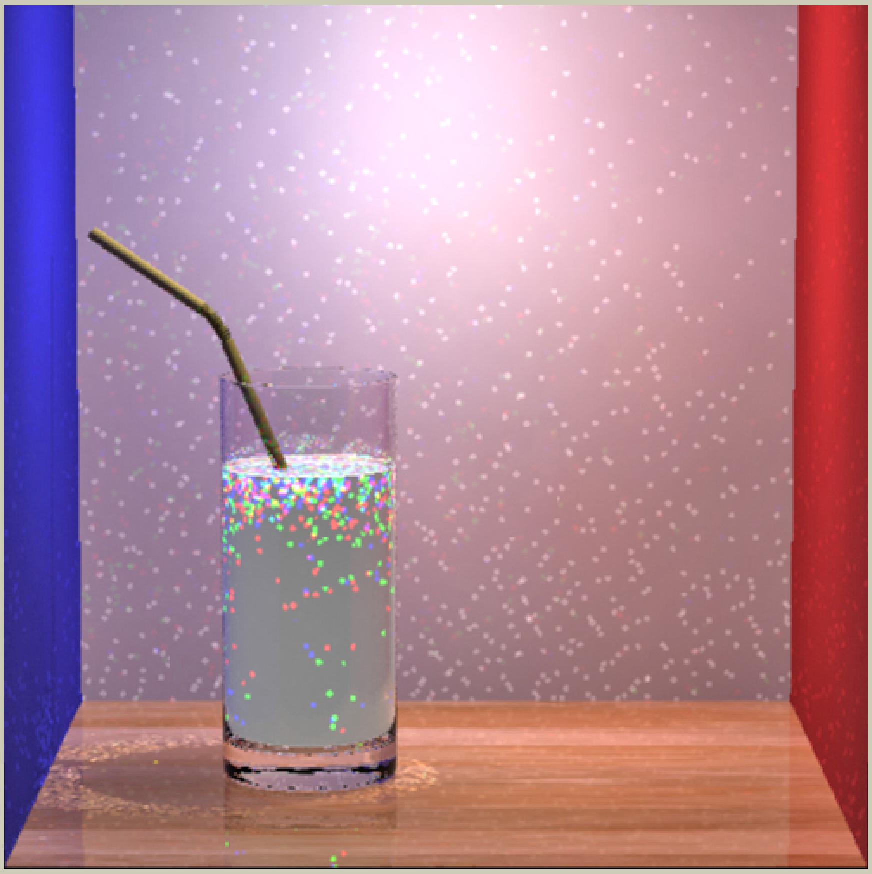


Image 1b

```
shader "mix_subsurf1"
"misss_physical" (
    "material" 0.5 0.5 0.5 1.,
    "transmission" 0.5 0.5 0.5 1.,
    "ior" 1.3,
    "absorption_coeff" 0.0014 0.0025 0.0142,
    "scattering_coeff" 0.7 1.22 1.9,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
    "depth" 10.,
    "max_samples" 1,
    "max_photons" 10,
==> "max_radius" 1.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
        "pointLight1"
    ]
    )

options "miDefaultOptions"
    caustic 0
    caustic filter box 1.1
==> caustic accuracy 1 0.1
    globillum on
    globillum 0
    globillum accuracy 1000 8.

light "pointLightShape1"
    = "physical_light1"
    origin 0. 0. 0.
    energy 10000. 10000. 10000.
    exponent 2.
    caustic photons 50000
    globillum photons 50000
    end light
```

We can immediately draw some conclusions:

Since the photons are widely distributed, having some more of them would improve the scattering simulation and the caustics effect generated by the glass. You can actually avoid the floor (and since it's a single object the whole cornell box) to cast caustics: by default objects do cast+receive caustics and GI, you can switch to receive only by changing the flag number from 3 to 2:

```
instance "pCube1" "pCubeShape1"
    material ["lambert2SG","lambert3SG","lambert4SG","lambert9SG"]
==> caustic 2
    globillum 3
    transform
        0.0416667 0. 0. 0.
        0. 0.0416667 0. 0.
        0. 0. 0.0416667 0.
        0. -0.5 0. 1.
    end instance
```

Although this is not really necessary. Do it if you like. Lets increase number of photons now.

1.6 Step 3 - Increasing photon density

In Image.1c both the number of caustics and GI photons in the light source is increased to 100000. One can see better caustics through the glass and a much better distribution of photons in the milk. misss_physical correctly scatters photons based on thewavelength dependentabsorption and scattering.

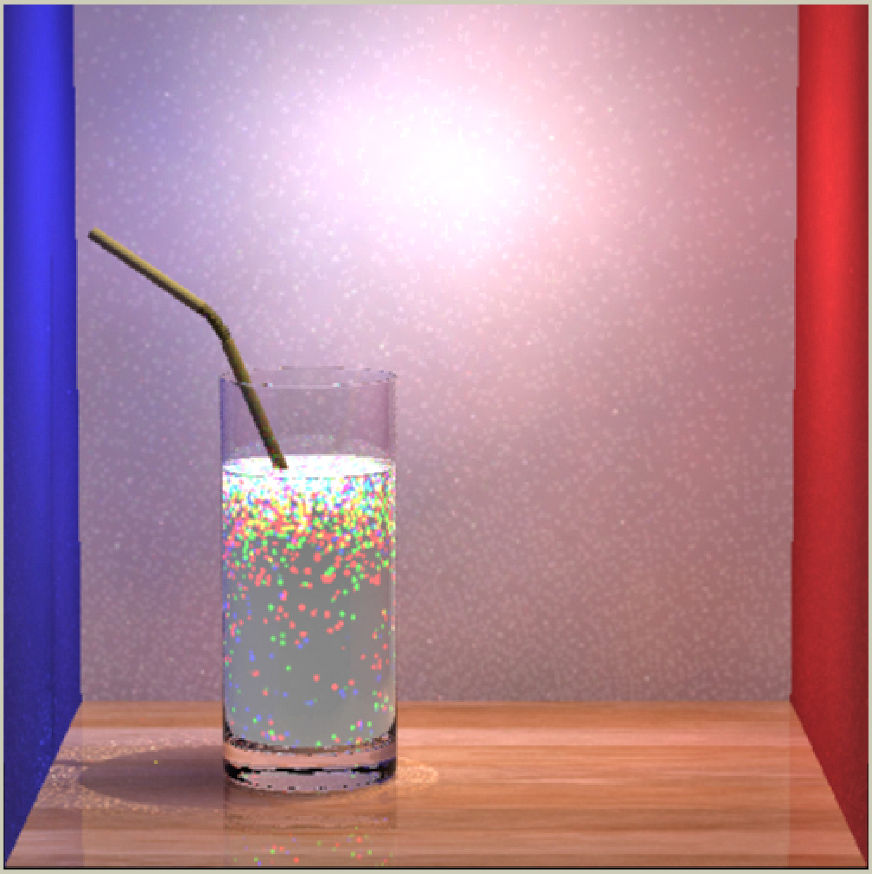


Image 1c

```
shader "mix_subsurf1"
"misss_physical" (
    "material" 0.5 0.5 0.5 1.,
    "transmission" 0.5 0.5 0.5 1.,
    "ior" 1.3,
    "absorption_coeff" 0.0014 0.0025 0.0142,
    "scattering_coeff" 0.7 1.22 1.9,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
    "depth" 10.,
    "max_samples" 1,
    "max_photons" 10,
    "max_radius" 1.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
        "pointLight1"
    ]
    )

options "miDefaultOptions"

caustic 0
    caustic filter box 1.1
    caustic accuracy 1 0.1
    globillum on
    globillum 0
    globillum accuracy 1000 8.

light "pointLightShape1"
    = "physical_light1"
    origin 0. 0. 0.
    energy 10000. 10000. 10000.
    exponent 2.
==> caustic photons 100000
==> globillum photons 100000
    end light
```

Again we can draw some conclusions:

The number of photons seem to be sufficient, now we can concentrate on their averaging by increasing max_radius and max_photons. The max_radius is the size of the spherical volume where we lookup photons. The max_photons is the maximum number of photons to lookup within the max_radius.

Similarly, for the caustics we will restore the previous values for caustic accuracy and radius.

We are going to see that shader statistics becomes useful at this point.

1.7 Step 4 - Forcing photon averaging

Here we'll do a first attempt by increasing max_radius to 2, and max_photons to 1000, this means we are going to lookup 1000 photons in a 3d sphere of radius 2mm (in the kd-tree). This is obviously a small radius, the milk section is 40 mm, but we do this on purpose to see what statistics will report.

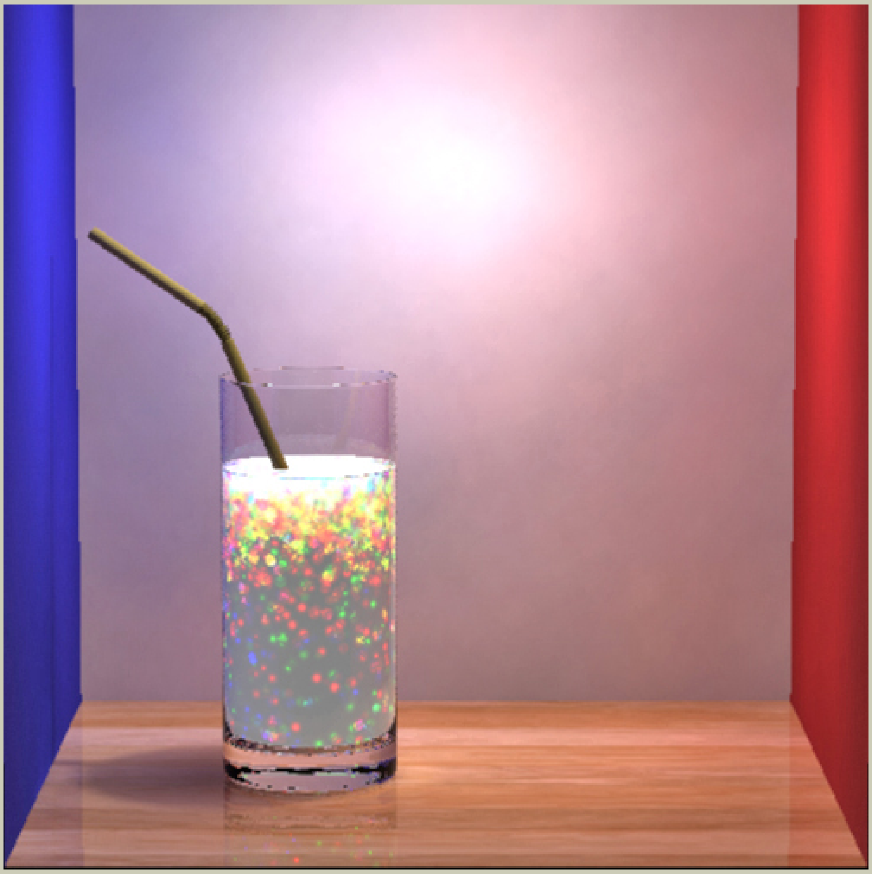


Image 1d

```
shader "mix_subsurf1"
"misss_physical" (
    "material" 0.5 0.5 0.5 1.,
    "transmission" 0.5 0.5 0.5 1.,
    "ior" 1.3,
    "absorption_coeff" 0.0014 0.0025 0.0142,
    "scattering_coeff" 0.7 1.22 1.9,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
    "depth" 10.,
    "max_samples" 1,
==> "max_photons" 1000,
==> "max_radius" 2.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
        "pointLight1"
    ]
    )

options "miDefaultOptions"
    caustic 0
    caustic filter box 1.1
    caustic accuracy 1000 1
    globillum on
    globillum 0
    globillum accuracy 1000 8.
```

Here's a portion of the messages produced by the miss_physical shader

```
    PHEN 0.0 info : "mix_subsurf1" multi-scatter photons per sample:
    PHEN 0.0 info : samples: 55632
    PHEN 0.0 info : minimum: 0.0000
    PHEN 0.0 info : maximum: 97.0000
    PHEN 0.0 info : mean: 6.9235
    PHEN 0.0 info : variance: 140.8517
```

Observe that the maximum photon count per sample is 97, the average is only 7 and the variance is 140. This indicates that the radius for sampling photons is too small. On average, only 7 photons contribute to the sample calculation. Increasing max_radius will allow the shader to find additional photons to improve the accuracy of each sample, thus producing a smoother image.

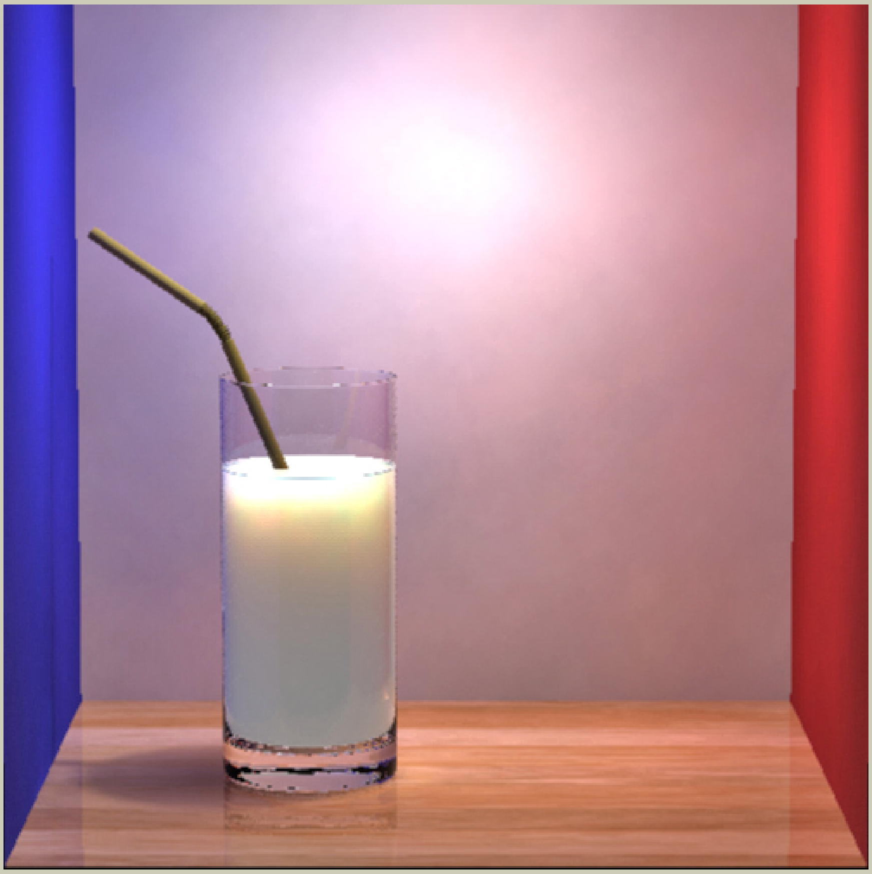


Image 1e

```
shader "mix_subsurf1"
"misss_physical" (
    "material" 0.5 0.5 0.5 1.,
    "transmission" 0.5 0.5 0.5 1.,
    "ior" 1.3,
    "absorption_coeff" 0.0014 0.0025 0.0142,
    "scattering_coeff" 0.7 1.22 1.9,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
    "depth" 10.,
    "max_samples" 1,
    "max_photons" 1000,
==> "max_radius" 40.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
    "pointLight1"
    ]
    )
```

A smoother image is obtained by increasing max_radius to 40.

```
PHEN 0.0 info : "mix_subsurf1" multi-scatter photons per sample:
PHEN 0.0 info : samples: 24616
PHEN 0.0 info : minimum: 1000.0000
PHEN 0.0 info : maximum: 1000.0000
PHEN 0.0 info : mean: 1000.0000
PHEN 0.0 info : variance: 0.0000
```

Now we basically lookup for the right number of photons in the right radius. Here I jumped to a good value because I knew the size of my SSS object, that's why modeling units are important.

Generally, once you have a decent photon distribution, if more than doubling the max_radius does not significantly change the average number of photons per sample nor the variance, it means that there are no photons to lookup anymore. In the case you still have an unsmooth image, means try increasing the number of photons cast.

On the other hand the image is now so bright that the rim and straw shadows on the milk are completely washed out. Also, the visibility of the straw in the zone where it enters/exits the milk is a bit low.

1.8 Step 5 - control brightness

The shader automatically handles energy balance, as you can see from the shader statistics at the end of a rendering process, but it is always possible to fine tune brightness.

On the input side, either the material and transmission color or the light's photon energy may be reduced and in general plays an important role:

- The 'material' of misss_physical is just a way to add a value. You can use 'material' to add a constant value everywhere.
- You can use instead transmission to get less influence from the scattering.

You can therefore choose a base value that gives a certain amount of color everywhere and then take only a part of the sss effect using a smaller transmission. This can 'clamp' the range of sss values in a nice way.

A note for the tech users: if you do *not* connect a material shader, reflectivity and refractivity is determined by Fresnel, which controls such effects depending on ior and the angle (viewer/normal). Fresnel is always activated, therefore you cannot deactivate but you can add a second reflection layer and use ior = 1.0 to get a minimum impact of fresnel.

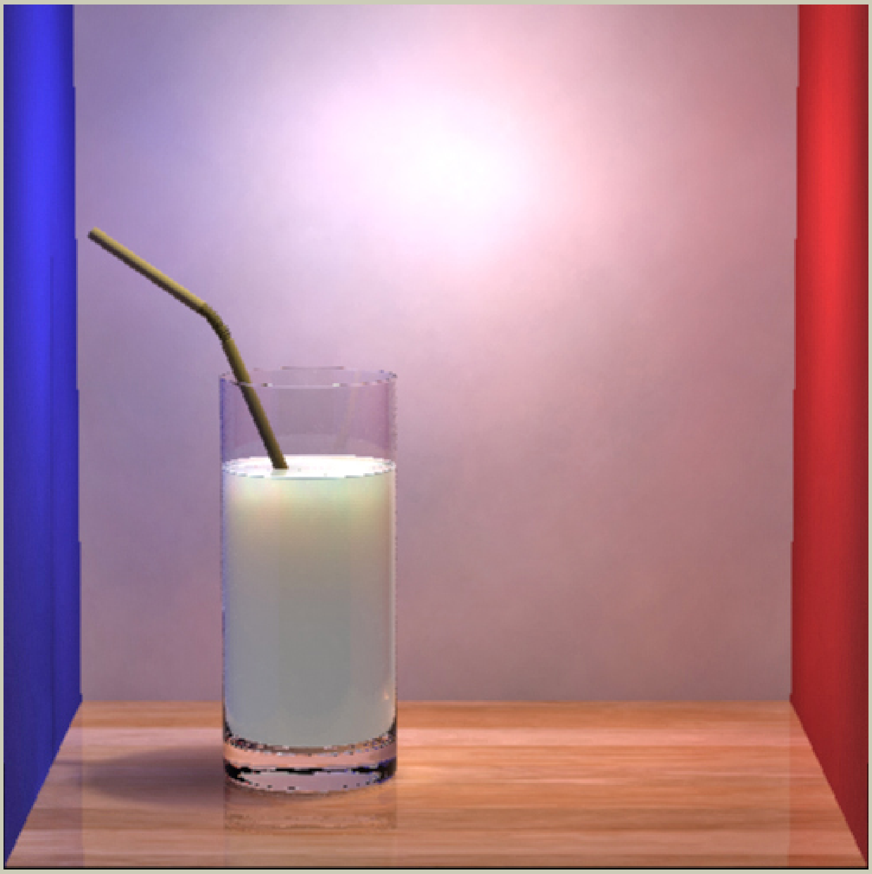


Image 1f

```
shader "mix_subsurf1"
"misss_physical" (
    "material" 0.5 0.5 0.5 1.,
==> "transmission" 0.25 0.25 0.25 1.,
    "ior" 1.3,
    "absorption_coeff" 0.0014 0.0025 0.0142,
    "scattering_coeff" 0.7 1.22 1.9,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
    "depth" 10.,
    "max_samples" 1,
    "max_photons" 1000,
    "max_radius" 40.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
        "pointLight1"
    ]
    )
```

1.9 Step 6 - Final quality and better visualize caustics

In this last step we are going to push the numbers a bit in order to reach the quality of Image.2:

- the number of caustics and GI photons in the light source is both increased to 250000,
- the photon lookups increased to 4000
- radius lookup is increased to 50 mm
- max_samples increase to 10.

You do know what the first three changes will do, as for max_samples this will affect better sampling.

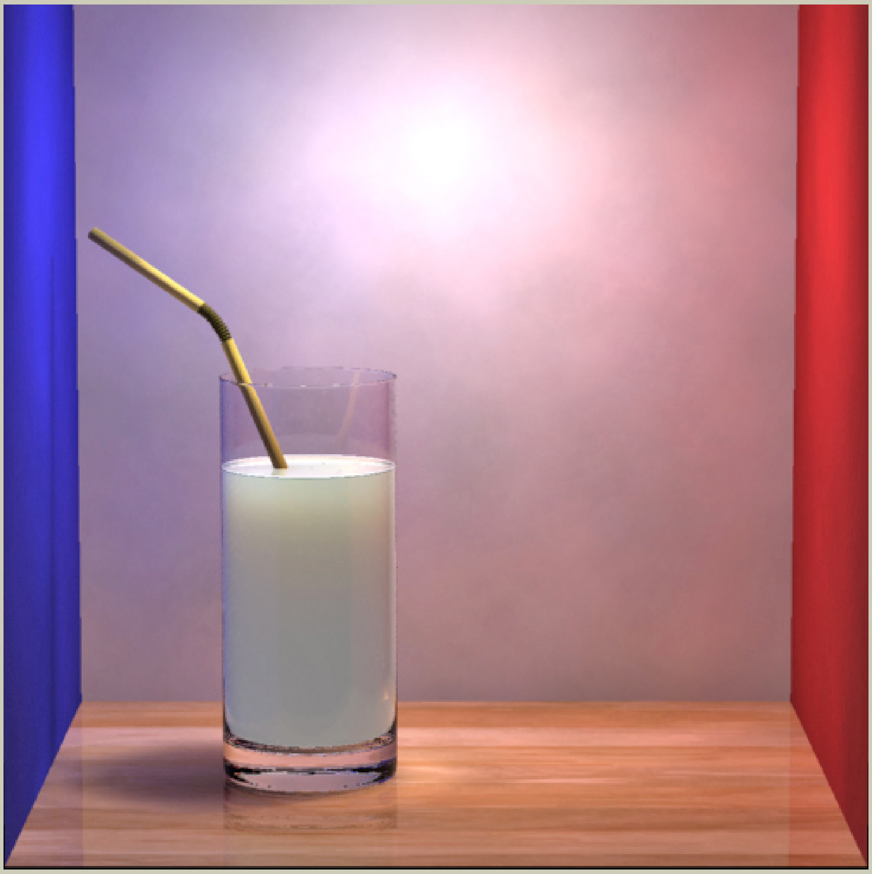


Image 2

```
shader "mix_subsurf1"
"misss_physical" (
    "material" 0.5 0.5 0.5 1.,
==> "transmission" 0.2 0.2 0.2 1.,
    "ior" 1.3,
    "absorption_coeff" 0.0014 0.0025 0.0142,
    "scattering_coeff" 0.7 1.22 1.9,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
==> "depth" 12.,
==> "max_samples" 10,
==> "max_photons" 4000,
==> "max_radius" 50.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
    "pointLight1"
    ]
    )

light "pointLightShape1"
    = "physical_light1"
    origin 0. 0. 0.
    energy 10000. 10000. 10000.
    exponent 2.
==> caustic photons 250000
==> globillum photons 250000
end light
```

Notice that the straw shadow caused by absence of GI photons becomes more evident. Notice also how nicely the end tip of the straw emerges from the milk at the bottom right of the glass.

In order to better see the caustics you can check out the scene used to render the banner of this tutorial, the glass of milk is now in a black box without front and rear walls, the light energy is increased and consequently the transmission color is reduced to avoid a glowing effect.

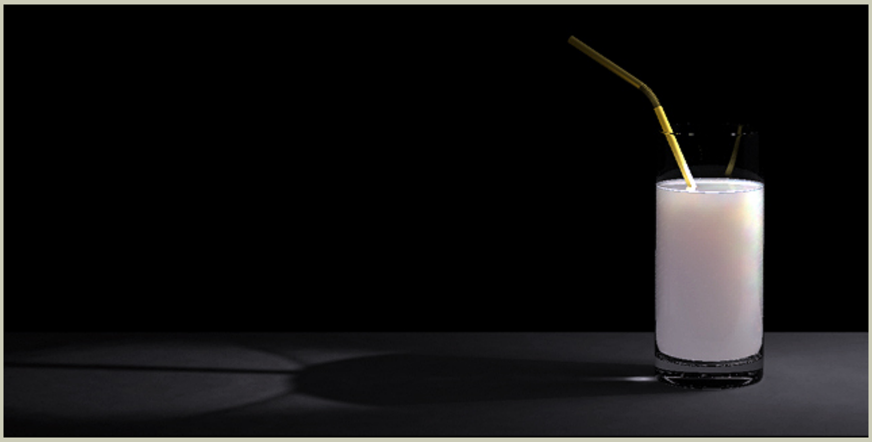


Image 3

Thank you for reading through!

Chapter 2

Physical subsurface scattering, dragon tutorial

2.1 Before you begin

In order to render correctly the scene files included with the dragon tutorial you must point to the required files on your disk/local network:

- point to the correct shader libraries and declaration files: `physics`, `base` and `subsurface`
- point to the file textures `wood.iff`, `riverscene.jpg`
- point to the external geometry file `dragon.mi` (you'll find it zipped, remember to uncompress)

Please modify all the scene files in your local folder.

Once all shader libraries are linked and included correctly you're ready to render the .mi file with the `ray` command:

- `ray -v on -x on -imgpipe 1 filename.mi | imf_disp -`

Where:

- `-v` on means progress messages verbosity
- `-x` on means colored warning/error messages
- `-imgpipe 1 ... | imf_disp -` pipes the output to mental ray image viewer

While rendering image will automatically show up tile-by-tile in `imf_disp`

For more options invoke `ray` help: `ray -h`

2.2 Intro

This document describes, in an easy-to-follow tutorial form, how to use the mental images subsurface scattering shader 'misss_physical' with mental ray standalone in order to simulate the appearance of a highly scattering material (in the case of this tutorial, a dragon made of jade).

Note that mental ray comes with shader documentation, which serves as a complete reference for the physical subsurface scattering shaders used in this tutorial. The subsurface scattering shader documentation is located in the last section following the base, physical, and contour shader documentation sections.

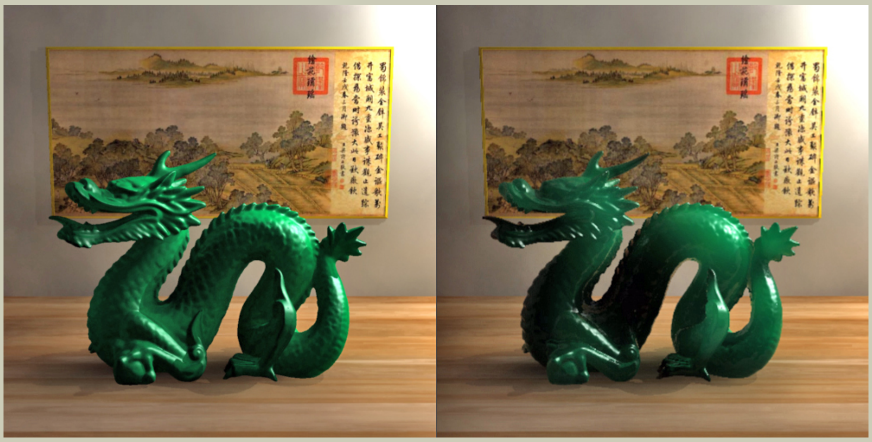
In order to be application independent the tutorial was made for mental ray standalone although the shader works on all OEM application which run mental ray core version 3.3 or later.

You are required to have:

- mental ray standalone 3.3 or later
- the misss_physical shader, from subsurface.DSO library and subsurface.mi declaration file
- public mental images 'base' and 'physics' shader libraries

Shown below are two images:

1. in the first image the dragon is rendered with the mib_illum_phong and mib_photon_basic, respectively material and photon shaders
2. the second image was rendered using the mental images subsurface scattering shader (misss_physical)



Left: No subsurface scattering vs. Right: Subsurface scattering

The intermediate steps taken to arrive at Image.2 from Image.1 are the body of this tutorial.

2.3 Options block, scene database and shader assignment

The options block requires :

- raytracing on
- caustics on
- globillum on

A brief background on what the scene database consist of:

- a placeholder for a polygonal mesh dragon geometry (courtesy from the Stanford 3DscanRep, it consist of roughly 1 million polygons). The placeholder holds the place for the real dragon geometry which is contained in a separate .mi file called dragon.mi (the geometry file).

  In the .mi the scene files the geometry is called in this way:

```
object "dragonShape"
    visible on
    trace on
    shadow on
    tagged
    box -7.929822 0.160171 -3.690562
        7.069097 10.73411 3.016582
==> file "yourpath\dragon.mi"
end object
```

While in the geometry file it looks like:

```
==> incremental object "dragonShape"
    visible on
    shadow on
    tagged
==> box -7.929822 0.160171 -3.690562
        7.069097 10.73411 3.016582
    group
        # point vectors
        2.34453 0.441679 -3.68691
        ...
        ...
        ...
        0 2614239 2614240 2614241
    end group
end object
```

This is shown just to let you know how it is done, you don't need to modify the dragon.mi file, just to know its location on your local disk and specify it on the tutorial scene files.

Geometry is then parsed at rendertime by calling the file at the right moment, the shader will be applied correctly since the objects share the same name.

There are various reason you want to parse geometry at rendertime which are not the subject of this tutorial, however the particular reason to have a setup like this one is that I could edit the scene files quickly without loading in the text editor many MB of polygon data.

- a poly box is surrounding the scene, representing the environment
- a poly plane and a NURBS beveled surface represents respectively the painting and its frame
- a camera, focal length 55 mm
- a pointlight, standing on the upper-right-far corner of the box

Shaders assignment:

- dragon in Image.1: mib_illum_phong and mib_photon_basic, respectively material and photon shaders
- dragon in Image.2: misss_physical, applied both as material and photon shader [1]
- polygonal box:
  - white ceiling, back wall, side and left wall: dgs_material and dgs_material_photon shaders

[1] alternatively you can use the misss_physical_phen applied only to the material port: it will automatically setup the connection to the photon one

  - floor: texture mapping network made of base shaders with dgs_material and dgs_material_photon shaders. The texture is a wood floor

- poly plane with the mapped painting: dgs_material and dgs_photon on a texture mapping network made of base shaders: the texture is a Chinese painting called "The river scene"

- NURBS beveled surface which is basically the painting's frame: another dgs_material, dgs_material_ photon material and photon shader couple

- light: mib_physical_light
  - Image.1: no caustics photons (caustic off), 100000 GI photons
  - Image.2: 100000 caustic photons, 100000 GI photons
  - intensity value 5000

So, almost the very same setup was used to produce both images, but instead of mib_illum_phong and mib_photon_basic, the subsurface scattering shader misss_physical, has been used in both the material and photon shaders slots for the dragon object in Image.2 (remember to lightlink the light to mi_sssphysical).

Let's now see how to fine tune the misss_physical shader on a step-by-step basis. The steps used to go from Image.1a (see below) to Image.2 illustrate the process of subsurface scattering parameter selection and variation. In the following images, only the jade material has changed, unless explicitly stated.

2.4 Step 1 - photon distribution

Starting from the setup of Image.1, let's substitute the mib_illum_phong and dgs_material_photon shader with the misss_physical shader:

- apply miss_physical both as material and photon shader
- remember to specify the light in the light_array

Then we need to turn on caustics in the options block (caustics photons couldn't be stored in the previous scene since all objects were diffusive, now that the misss_physical is attached to the dragon placeholder, caustics photons will be stored for the SSS computation):

- caustic on

The following image - Image.1a - was rendered using the following table of parameters of misss_physical, this is the scene you are going to use as a starting point for the tutorial.

The scene features also lower quality parameters such as less photons (c, GI 50000) and lower antialiasing values (-2, 0) which will allow faster render times while keeping the effect a good preview.

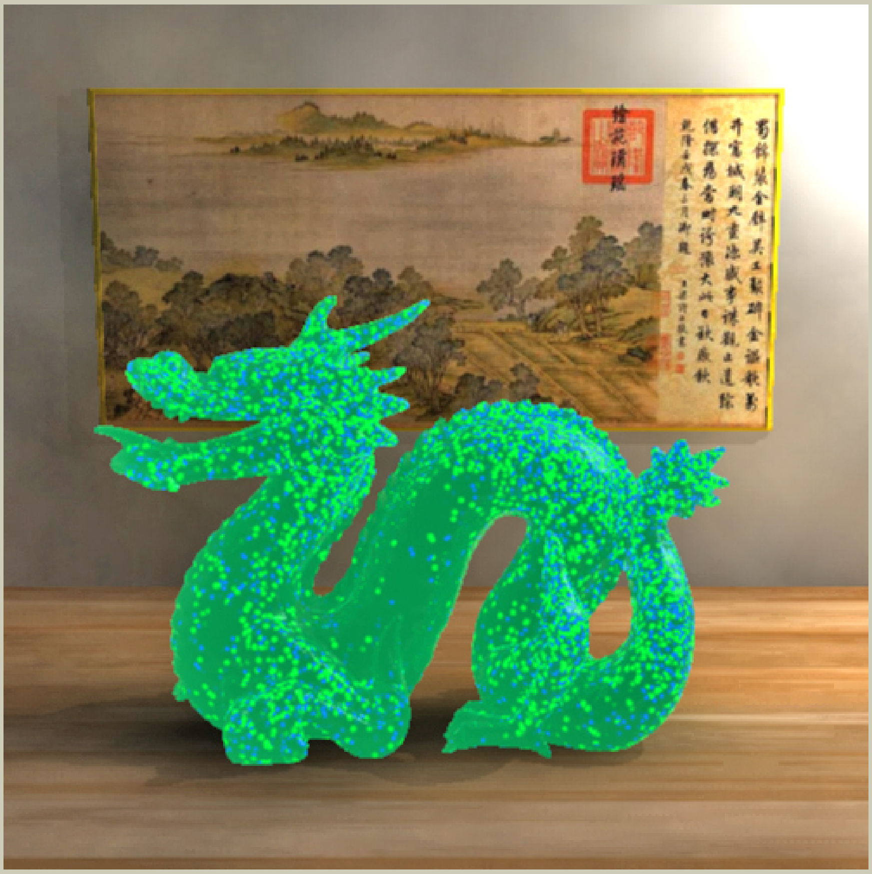


Image 1a

```
shader "dragon1_mix_subsurf1"
"misss_physical" (
    "material" 0. 0.5 0.25 1.,
    "transmission" 0. 0.5 0.25 1.,
    "ior" 1.5,
    "absorption_coeff" 0.001 0.001 0.00213,
    "scattering_coeff" 0.657 0.786 0.9,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
    "depth" 1.,
    "max_samples" 1,
    "max_photons" 1,
    "max_radius" 1.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
        "pointLight2"
    ]
    )
```

As you can see there are more photons on the right and on the top part of the dragon as it is expected for the light position.

Some consideration on the parameters:

The absorption and scattering coefficient for Jade are derived trough trial and error, I have to thank "Rui Wang" for suggesting such values:

- "absorption_coeff" 0.001 0.001 0.00213,
- "scattering_coeff" 0.657 0.786 0.9,

Scale conversion is set to 10 because the scene is modeled in centimeters hence 10mm = 1 cm.

Setting max_photons and max_radius will produce this diagnostic/photon density image, useful to understand where photons are stored. We are going to increase photon number to 100000 (both for c and GI) since a bit more density is required.

We're not going to tune the shader step by step as in the milk tutorial, so the next image will be already tuned, we're going instead to show ho to use the misss_physical as an utility node piping in a material shader. We can say that the color is too bright and obviously we need to increase and smooth the photons averaging.

Read the milk tutorial for a deeper understanding of the tuning process.

2.5 Step 2 - tuned scene

In the following image we have tuned the shader to get a good result:

- increased c and GI photons to 100000
- we've shuttled down the material color to black since the dragon was too bright, now the scattering effect is more evident.
- Then the smoothing of photons averaging is handled by the higher max_photons and max_radius.
- We've increased the samples to get better quality and increased depth to 10 scattering events, actually 1 MFP (mean free path) = 1 scattering event.

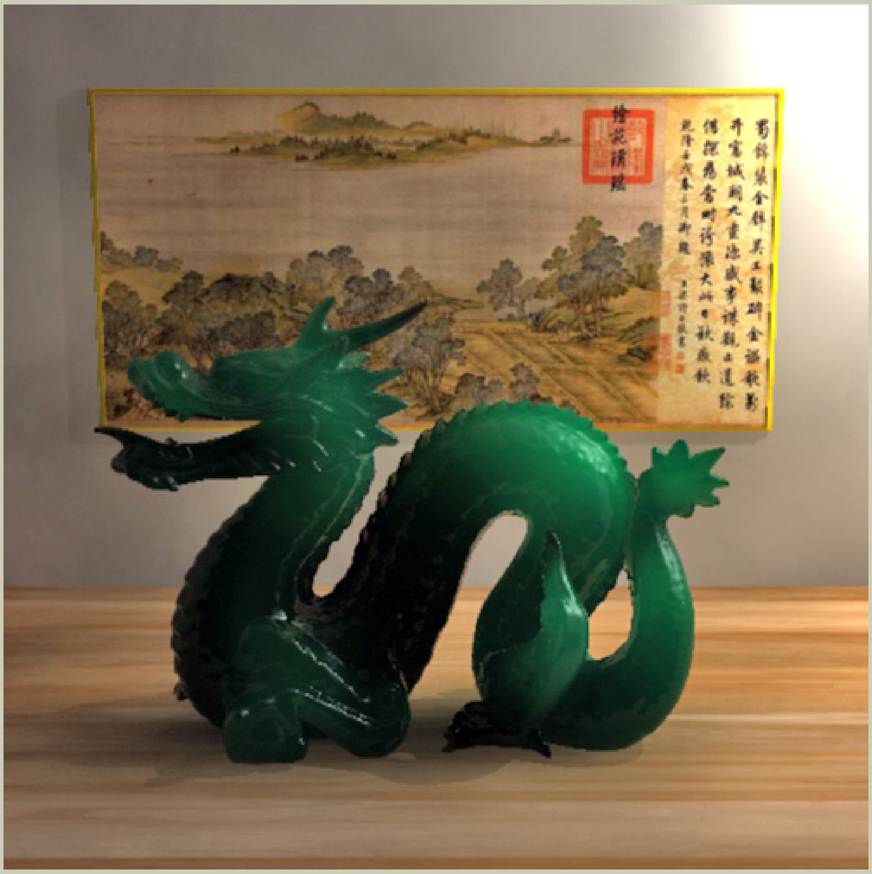


Image 1b

```
shader "dragon1_mix_subsurf1"
"misss_physical" (
==> "material" 0. 0. 0. 1.,
    "transmission" 0. 0.5 0.25 1.,
    "ior" 1.5,
    "absorption_coeff" 0.001 0.001 0.00213,
    "scattering_coeff" 0.657 0.786 0.9,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
==> "depth" 10.,
==> "max_samples" 10,
==> "max_photons" 2000,
==> "max_radius" 50.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
        "pointLight2"
    ]
    )
```

The look is now more satisfying, maybe a bit too dark, we could increase the material to an V-HSV value of 0.25 and transmission to V 0.75 to get an intermediate effect from step one and this step. We can do this later though, what we are going to fine tune are the specular effects on the dragon. Imagine we have a dragon made of polished Jade: how we are going to get this effect? Simple, read the answer on step 3.

2.6 Step 3 - specular effects

In order to get a polished jade look, we want the dragon to appear a bit more specular and to reflect the surroundings on itself: what we need to do is to pipe onto the material color a material shader which handles specularity, such the illumination model of mib_illum_phong (remembering to lightlink the light also to mib_illum_phong).

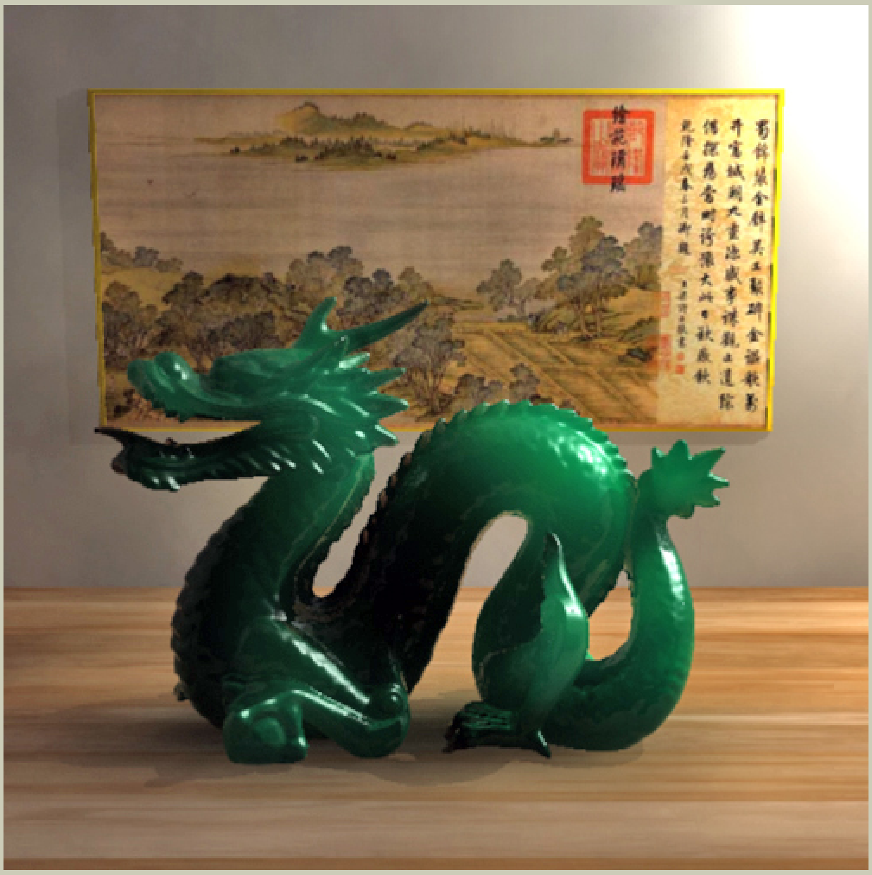


Image 1c

```
shader "dragon1_mix_subsurf1"
```

```
"misss_physical" (
==> "material" = "dragon1_mib_illum_phong1",
==> "transmission" 0. 0.6 0.3 1.,
    "ior" 1.5,
    "absorption_coeff" 0.001 0.001 0.00213,
    "scattering_coeff" 0.657 0.786 0.9,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
    "depth" 10.,
    "max_samples" 10,
    "max_photons" 2000,
    "max_radius" 50.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
        "pointLight2"
    ]
    )

shader "dragon1_mib_illum_phong1"
"mib_illum_phong" (
    "ambience" 0. 0. 0. 1.,
    "ambient" 0. 0. 0. 1.,
==> "diffuse" 0. 0.125 0.0625 1.,
==> "specular" 1. 1. 1. 1.,
==> "exponent" 50.,
    "mode" 0,
    "lights" [
        "pointLight2"
    ]
    )
```

We have adjusted transmission, to see the scattering effect just a bit more, and material: we have now a shader in place of a RGBA color, the shader instance is "dragon1_mib_illum_phong1" which is the mib_illum_phong we were referring to.

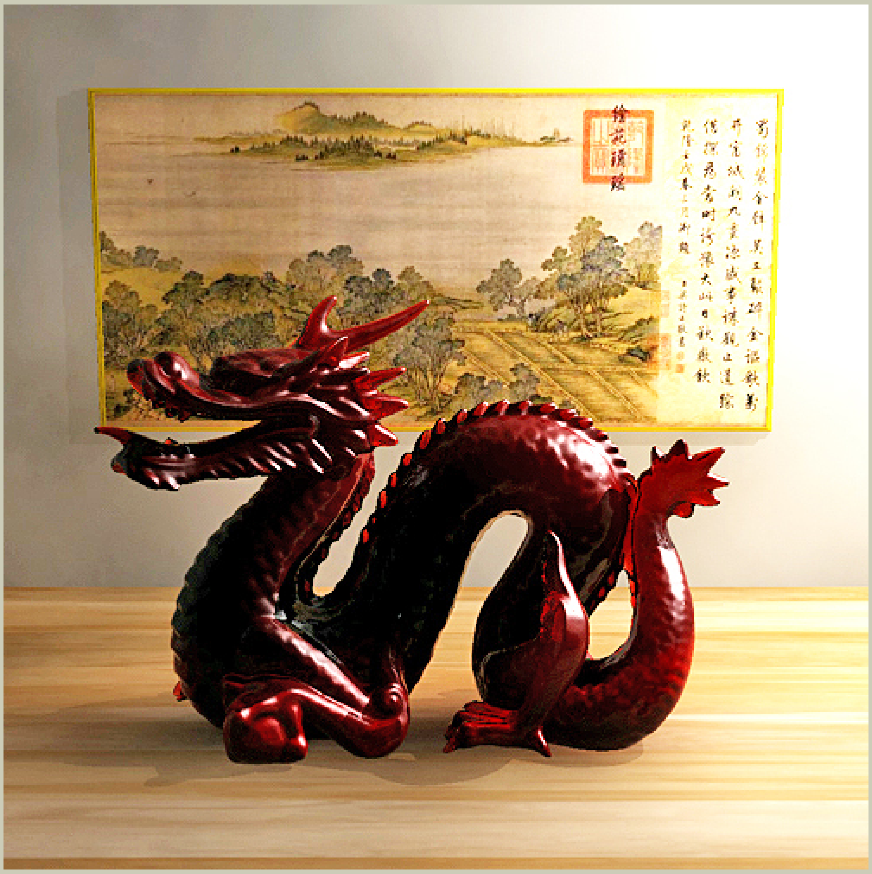
In order to control specularity we now have to tweak its specular color and exponent parameters (the higher the exponent, the crisper the specular areas). Diffuse now will take the original material color of the previous example, which in this case was increased from black to a dark green (V = 0.125).

In order to have the dragon reflect the environment you then need to add a sample compositing shader such as mi_reflect, and weight the effect trough the utility mib_color_mix. I will leave this as a further step for users who want to experiment.

Anyway, all OEM applications do have shaders which handles reflections inside their phong shaders and not recurring an utility shader.

2.7 EXTRA Step - Papa tomato says: CATCH UP!

Please pass me the “Fox-Force-Five” joke, I don’t think Quentin Tarantino will personally use this shader, anyway, here we’ll turn jade into ketchup, an useful example to show how important are the coefficients and the material and transmission color, also the light intensity has changed.



A dragon made of ketchup

```
shader "dragon1_mix_subsurf1"
"misss_physical" (
    "material" = "dragon1_mib_illum_phong1",
    "transmission" 0.5 0.0 0.0 1.,
    "ior" 1.5,
==> "absorption_coeff" 0.061 0.97 1.45,
==> "scattering_coeff" 0.18 0.07 0.03,
    "scale_conversion" 10.,
    "scattering_anisotropy" 0.75,
    "depth" 10.,
    "max_samples" 10,
    "max_photons" 2000,
    "max_radius" 50.,
    "approx_diffusion" on,
    "approx_single_scatter" on,
    "approx_multiple_scatter" on,
    "lights" [
        "pointLight2"
    ]
    )

shader "dragon1_mib_illum_phong1"
"mib_illum_phong" (
    "ambience" 0. 0. 0. 1.,
    "ambient" 0. 0. 0. 1.,
==> "diffuse" 0.25 0.0 0.0 1.,
    "specular" 1. 1. 1. 1.,
    "exponent" 27.,
    "mode" 0,
    "lights" [
        "pointLight2"
    ]
    )
```

Following here some practical notes on the shader parameters of the misss_physical. Please note the comments and refer to the shader documentation for detailed explanation of each attribute.

type	name	comments for users	unit
color	material	set a color or a color shader (eg: 3dtexture) or an illumination model (eg: mib_illum_phong). The effect is additive.	
color	transmission	set a color or a color shader (eg: 3dtexture), it controls the intensity of the SSS effect, acting as a multiplier	
scalar	ior	range is 0.5 to 10, depends on material	
vector	absorption_coeff	find and use experimental values, no need to be normalized. Experimentally derived absorption_coeff for skim milk is: R 0.0014, G 0.0025, B 0.0142	inverse length
vector	scattering_coeff	find and use experimental values, no need to be normalized. Experimentally derived scattering_coeff for skim milk is: R 0.070, G 1.22, B 1.90	inverse length

scalar	scale_conversion	use 10 if cm was the modeling unit size: 10 mm = 1 cm	
scalar	scattering_anisotropy	also called "g", range is -1 to 1, it's the degree of scattering direction, negative is backside, positive is forward scattering (0 = isotropic); depends from material	
scalar	depth	this acts as a multiplier of the MFP (mean free path) it defines the depth of the shallow layer. A value of 10 means 10 units of MFP, which is different for the RGB components. Check shader statistics.	MFP units
integer	max_samples	start from 1 and go up to 32 if you need better sampling quality	
integer	max_photons	max number of photons to lookup in the max_radius. Start with 500, then increase up to thousands... generally you should get a good result with 1000 lookups.	
scalar	max_radius	the unit here is millimeters, start with 10 and increase, see feedback, depends on your SSS object dimension	mm
boolean	approx_diffusion	should be on, but you can experiment	
boolean	approx_single_scatter	should be on, but you can experiment	
boolean	approx_multiple_scatter	should be on, but you can experiment	
array light	lights	light(s) you want to contribute to subsurface scattering calculations	

Thank you for reading through!