IR(3): Global Illumination

Alessandro Martinelli
alessandro.martinelli@unipv.it

27 November 2013

Illumination and Rendering:
- Illumination Principles
- Rendering Real-Time
- Global Illumination
IR(3): Global Illumination

Illumination and Rendering:
- Illumination Principles
- Rendering Real-Time
- **Global Illumination**
  - A Rendering Equation
  - Global Illumination Algorithms
A Rendering Equation (1/3)

Considering an entire 3D scenario:

Radiance Function

The Radiance Function \( L(X, Y) \) is the function giving the Radiance intensity on ray starting on a point \( X \) and hitting a point \( Y \), where \( X \) and \( Y \) are any point of the scenario, usually placed on some surface.
Considering an entire 3D scenario:

Radiance Function

The Radiance Function $L(X, Y)$ is the function giving the Radiance intensity on ray starting on a point $X$ and hitting a point $Y$, where $X$ and $Y$ are any point of the scenario, usually placed on some surface.

The Radiance Function tells what it can be seen from the point $Y$ when watching the point $X$. 
A Rendering Equation (1/3)

Considering an entire 3D scenario:

Radiance Function

The Radiance Function $L(X, Y)$ is the function giving the Radiance intensity on ray starting on a point $X$ and hitting a point $Y$, where $X$ and $Y$ are any point of the scenario, usually placed on some surface.

The Radiance Function tells what it can be seen from the point $Y$ when watching the point $X$.

Whole Scene BRDF

The Function $f(v_{XY}, l_{XZ}) = f(X, Y, Z)$ stays for a BRDF function in the point $X$, for each significant point on a scene, where $v_{XY}$ is the direction from the point $X$ to a point $Y$ and $l_{XZ}$ is the direction from the point $X$ to a point $Z$. 
A Rendering Equation (1/3)

Considering an entire 3D scenario:

Radiance Function

The Radiance Function \( L(X, Y) \) is the function giving the Radiance intensity on ray starting on a point \( X \) and hitting a point \( Y \), where \( X \) and \( Y \) are any point of the scenario, usually placed on some surface.

The Radiance Function tells what it can be seen from the point \( Y \) when watching the point \( X \)

Whole Scene BRDF

The Function \( f(v_{XY}, l_{XZ}) = f(X, Y, Z) \) stays for a BRDF function in the point \( X \), for each significant point on a scene, where \( v_{XY} \) is the direction from the point \( X \) to a point \( Y \) and \( l_{XZ} \) is the direction from the point \( X \) to a point \( Z \)

So, in every point of the scene, the BRDF will be:

\[
f(X, Y, Z) = \frac{dL(X, Y)}{dE_{\perp}(X, Z)}
\]
Given the BRDF Model:

\[ dL(X, Y) = f(X, Y, Z)dE_\perp(X, Z) \]  \hspace{1cm} (2)
A Rendering Equation (2/3)

Given the BRDF Model:

\[ dL(X, Y) = f(X, Y, Z)dE_{\perp}(X, Z) \]  \hspace{1cm} (2)

We should remember that:

\[ dE_{\perp}(X, Z) = \cos(\beta_{XZ})L(Z, X)d\omega \]  \hspace{1cm} (3)
Given the BRDF Model:

\[ dL(X, Y) = f(X, Y, Z)dE_\perp(X, Z) \]  \hspace{1cm} (2)

We should remember that:

\[ dE_\perp(X, Z) = \cos(\beta_{XZ})L(Z, X)d\omega \]  \hspace{1cm} (3)

Where \( \beta_{XZ} \) is the angle the Normal on the Surface in \( X \) will have with the direction going from \( X \) to \( Z \).
A Rendering Equation (2/3)

Given the BRDF Model:

\[ dL(X, Y) = f(X, Y, Z)dE_\perp(X, Z) \]  

We should remember that:

\[ dE_\perp(X, Z) = \cos(\beta_{XZ})L(Z, X)d\omega \]  

Where \( \beta_{XZ} \) is the angle the Normal on the Surface in \( X \) will have with the direction going from \( X \) to \( Z \)

So:

\[ dL(X, Y) = f(X, Y, Z)\cos(\beta_{XZ})L(Z, X)d\omega \]
A Rendering Equation (2/3)

Given the BRDF Model:

\[ dL(X, Y) = f(X, Y, Z)dE_\perp(X, Z) \]  \hspace{1cm} (2)

We should remember that:

\[ dE_\perp(X, Z) = \cos(\beta_{XZ})L(Z, X)d\omega \]  \hspace{1cm} (3)

Where \( \beta_{XZ} \) is the angle the Normal on the Surface in \( X \) will have with the direction going from \( X \) to \( Z \)

So:

\[ dL(X, Y) = f(X, Y, Z)\cos(\beta_{XZ})L(Z, X)d\omega \]  \hspace{1cm} (4)

The Radiance Function is the solution of the very well known Rendering Equation (Kajiya 1986):

\[ L(X, Y) = \int_\Omega f(X, Y, Z)\cos(\beta_{XZ})L(Z, X)d\omega \]  \hspace{1cm} (5)

Where \( \Omega \) is the maximum solid angle \([0, 4\pi]\)
A Rendering Equation (3/3)
A Rendering Equation (3/3)

The Rendering Equation:

- Is a *differential equation*
The Rendering Equation:

- Is a *differential equation*
- Is used to evaluate *The Radiance Function*...
A Rendering Equation (3/3)

The Rendering Equation:

- Is a *differential equation*
- Is used to evaluate **The Radiance Function**...
- Which is on both the side of the equation!!!
The Rendering Equation:

- Is a *differential equation*
- Is used to evaluate **The Radiance Function**...
- Which is on both the side of the equation!!!
- With few exceptions, it will not have an analytical solution, and it will require numeric analysis.
Global Illumination

With **Global Illumination** we mean the definition of algorithms used to find an approximation for the **Radiance Function** solving the Rendering Equation.
Global Illumination

With **Global Illumination** we mean the definition of algorithms used to find an approximation for the **Radiance Function** solving the Rendering Equation. This algorithms must be generic, that is, they must work on every scene!

Some famous algorithms:

- Ray-Tracing (*treated at the course*)
- Generic Path-Tracing (*treated at the course*)
- Radiosity (*treated at the course*)
- Photon Mapping
- Precomputed Radiance Transfer
- Irradiance Maps
- etc.
IR(3): Global Illumination

Illumination and Rendering:
- Illumination Principles
- Rendering Real-Time
- **Global Illumination**
  - A Rendering Equation
  - *Global Illumination Algorithms*
Ray Casting is a Rasterization Algorithm alternative to the one we have already seen on the Rendering Pipeline:
Ray Casting is a Rasterization Algorithm alternative to the one we have already seen on the Rendering Pipeline:

- It is given a Point of View (the Eye of the Viewer)
Ray Casting is a Rasterization Algorithm alternative to the one we have already seen on the Rendering Pipeline:

- It is given a Point of View (the Eye of the Viewer)
- It is given a model about how the image to be drawn is placed in the space. Pixels are rectangles in the 3D space.
Ray Casting is a Rasterization Algorithm alternative to the one we have already seen on the Rendering Pipeline:

- It is given a Point of View (the Eye of the Viewer)
- It is given a model about how the image to be drawn is placed in the space. Pixels are rectangles in the 3D space.
- It is given an entire 3D scenario
Ray Casting is a Rasterization Algorithm alternative to the one we have already seen on the Rendering Pipeline:

- It is given a Point of View (the Eye of the Viewer)
- It is given a model about how the image to be drawn is placed in the space. Pixels are rectangles in the 3D space.
- It is given an entire 3D scenario
- It is evaluated the ray starting from the Point of View and passing through the center of each Pixel
Ray Casting is a **Rasterization Algorithm** alternative to the one we have already seen on the Rendering Pipeline:

- It is given a Point of View (the Eye of the Viewer)
- It is given a model about how the image to be drawn is placed in the space. Pixels are rectangles in the 3D space.
- It is given an entire 3D scenario
- It is evaluated the ray starting from the Point of View and passing through the center of each Pixel
- It is found the nearest of all the possible intersections with object in the scene

The intersection stays for what can be seen from the point of view through each of the pixels on the image. The color of the object on the intersection point will be the color of the pixel.
• Ray Tracing is based on Ray Casting.
Ray Tracing (1/2)

- Ray Tracing is based on Ray Casting.
- With a Ray-Casting software library, it is possible to evaluate the intersections ray-object.
Ray Tracing (1/2)

- Ray Tracing is based on Ray Casting.
- With a Ray-Casting software library, it is possible to evaluate the intersections ray-object.
- On that intersection it is possible to evaluate alternatives rays.
Ray Tracing (1/2)

- Ray Tracing is based on Ray Casting.
- With a Ray-Casting software library, it is possible to evaluate the intersections ray-object.
- On that intersection it is possible to evaluate alternatives rays.
  - The Reflected Ray
  - The Refracted Ray
  - The Ray going to some Point-Light, to verify if there are objects in the path, so to understand if the intersection point is inside a shadow.
Ray Tracing (1/2)

- Ray Tracing is based on Ray Casting.
- With a Ray-Casting software library, it is possible to evaluate the intersections ray-object.
- On that intersection it is possible to evaluate alternatives rays.
  - The Reflected Ray
  - The Refracted Ray
  - The Ray going to some Point-Light, to verify if there are objects in the path, so to understand if the intersection point is inside a shadow.

Reflected and Refracted Rays can add other objects and may be subdivided again.
Ray Tracing (1/2)

- Ray Tracing is based on Ray Casting.
- With a Ray-Casting software library, it is possible to evaluate the intersections ray-object.
- On that intersection it is possible to evaluate alternatives rays.
  - The Reflected Ray
  - The Refracted Ray
  - The Ray going to some Point-Light, to verify if there are objects in the path, so to understand if the intersection point is inside a shadow.

Reflected and Refracted Rays can add other objects and may be subdivided again.
With Ray Tracing we are able to approximate some elements of the Radiance Function, when BRDF are Lambertian, or when they take into account of simple Surface Reflections, but it is less useful on complex BRDF taking into account (for example) micro faces models.
images generated with the Open Source software PoV-Ray (PoV:Persistence of Vision) www.povray.org
Monte-Carlo Integration

Monte-Carlo Integration is a numeric solution which can be used in many situations to evaluate integrals, which is based upon the use of samples placed in random positions.
Monte-Carlo Integration and Path-Tracing (1/2)

Monte-Carlo Integration

Monte-Carlo Integration is a numeric solution which can be used in many situations to evaluate integrals, which is based upon the use of samples placed in random positions.

Example (Monte-Carlo Integration)

- It is given a function $f(x)$, and it is necessary to evaluate the integral of $f(x)$ in $[x_A, x_B]$. 
Monte-Carlo Integration

Monte-Carlo Integration is a numeric solution which can be used in many situations to evaluate integrals, which is based upon the use of samples placed in random positions.

Example (Monte-Carlo Integration)

- It is given a function $f(x)$, and it is necessary to evaluate the integral of $f(x)$ in $[x_A, x_B]$.
- It is extracted a vector of $N$ random values $x_i$, so that $x_A \leq x_1, x_2 \ldots x_N \leq x_B$.
Monte-Carlo Integration

**Monte-Carlo Integration** is a numeric solution which can be used in many situations to evaluate integrals, which is based upon the use of samples placed in random positions.

**Example (Monte-Carlo Integration)**

- It is given a function \( f(x) \), and it is necessary to evaluate the integral of \( f(x) \) in \([x_A, x_B]\).
- It is extracted a vector of \( N \) random values \( x_i \), so that \( x_A \leq x_1, x_2 \ldots x_N \leq x_B \)
- It is possible to evaluate the integral like:

\[
\int_{x_A}^{x_B} f(x) \, dx \cong \left( \sum_{i=1}^{N} f(x_i) \right) \frac{(x_B - x_A)}{N}
\]  

(6)
Monte-Carlo Integration and Path-Tracing (1/2)

Monte-Carlo Integration

Monte-Carlo Integration is a numeric solution which can be used in many situations to evaluate integrals, which is based upon the use of samples placed in random positions.

Example (Monte-Carlo Integration)

- It is given a function $f(x)$, and it is necessary to evaluate the integral of $f(x)$ in $[x_A, x_B]$.
- It is extracted a vector of $N$ random values $x_i$, so that $x_A \leq x_1, x_2 \ldots x_N \leq x_B$.
- It is possible to evaluate the integral like:

$$
\int_{x_A}^{x_B} f(x) \, dx \cong \left( \sum_{1}^{N} f(x_i) \right) \frac{(x_B - x_A)}{N} \tag{6}
$$

Why should we use Monte-Carlo Integration?
Monte-Carlo Integration and Path-Tracing (1/2)

Monte-Carlo Integration

Monte-Carlo Integration is a numeric solution which can be used in many situations to evaluate integrals, which is based upon the use of samples placed in random positions.

Example (Monte-Carlo Integration)

- It is given a function \( f(x) \), and it is necessary to evaluate the integral of \( f(x) \) in \([x_A, x_B]\).
- It is extracted a vector of \( N \) random values \( x_i \), so that \( x_A \leq x_1, x_2 \ldots x_N \leq x_B \)
- It is possible to evaluate the integral like:

\[
\int_{x_A}^{x_B} f(x) \, dx \approx \left( \sum_{1}^{N} f(x_i) \right) \frac{(x_B - x_A)}{N} \quad (6)
\]

Why should we use Monte-Carlo Integration?

- Because there are situations in which the random Integrator will give a better result compared to fixed-step solutions.
Monte-Carlo Integration and Path-Tracing (1/2)

Monte-Carlo Integration

Monte-Carlo Integration is a numeric solution which can be used in many situations to evaluate integrals, which is based upon the use of samples placed in random positions.

Example (Monte-Carlo Integration)

- It is given a function $f(x)$, and it is necessary to evaluate the integral of $f(x)$ in $[x_A, x_B]$.
- It is extracted a vector of $N$ random values $x_i$, so that $x_A \leq x_1, x_2 \ldots x_N \leq x_B$
- It is possible to evaluate the integral like:

$$\int_{x_A}^{x_B} f(x) \, dx \approx \left( \sum_{1}^{N} f(x_i) \right) \frac{(x_B - x_A)}{N} \quad (6)$$

Why should we use Monte-Carlo Integration?

- Because there are situations in which the random Integrator will give a better result compared to fixed-step solutions.
- ... like in our situation
Path-Tracing

- It generalizes the idea of Ray-Tracing.
Path-Tracing

- It generalizes the idea of Ray-Tracing.
- Basically: the most general solution known when BRDFs are of any type.
Path-Tracing

- It generalizes the idea of Ray-Tracing.
- Basically: the **most general solution known** when BRDFs are of **any type**.
- It is based upon an approximation of the Rendering Equation using Monte-Carlo Integration.
Path-Tracing

- It generalizes the idea of Ray-Tracing.
- Basically: the **most general solution known** when BRDFs are of any type.
- It is based upon an approximation of the Rendering Equation using Monte-Carlo Integration.

How should we use Monte-Carlo Integration on the Rendering Equation?
Monte-Carlo Integral and Path-Tracing (2/2)

Path-Tracing

- It generalizes the idea of Ray-Tracing.
- Basically: the most general solution known when BRDFs are of any type.
- It is based upon an approximation of the Rendering Equation using Monte-Carlo Integration

How should we use Monte-Carlo Integration on the Rendering Equation?

\[ L(\mathbf{X}, \mathbf{Y}) = \int_{\Omega} f(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) \cos(\beta_{XZ}) L(\mathbf{Z}, \mathbf{X}) d\omega \]

... when it is necessary to evaluate. When it is necessary to evaluate these integrals, they are approximated by sending rays to a very large number (1000, 10000) of possible, random, points.
The **Radiosity Algorithm** is applied under the assumption all the surfaces have only a diffuse component.
The **Radiosity Algorithm** is applied under the assumption all the surfaces have only a diffuse component. In this situation:

\[ L = \frac{c_{\text{diff}} E}{\pi} \]  

where \( E \) is the total irradiance falling on a surface, and \( c_{\text{diff}} \) is the coefficient of the diffuse reflection, \( L \) is the Radianza which goes to any direction from that surface point.
The *Radiosity Algorithm* is applied under the assumption all the surfaces have only a diffuse component. In this situation:

\[ L = \frac{c_{\text{diff}} E}{\pi} \]  

(7)

Where \( E \) is the total irradiance falling on a surface, and \( c_{\text{diff}} \) is the coefficient of the diffuse reflection, \( L \) is the Radianza which goes to any direction from that surface point.

Let’s suppose a scene is built with a finite set of \( N \) triangles:

- To each couple of polygons it is related a factor \( G_{ij} \), called geometric factor, defining how much of the Radiance exiting from the triangle \( i \) will go to the triangle \( j \)
- Instead of evaluating the *Radiance Function*, it is evaluated the Radiosity exiting from each triangle.
The **Radiosity Algorithm** is applied under the assumption all the surfaces have only a diffuse component. In this situations:

\[
L = \frac{c_{\text{diff}} E}{\pi}
\]  

(7)

Where \( E \) is the total irradiance falling on a surface, and \( c_{\text{diff}} \) is the coefficient of the diffuse reflection, \( L \) is the Radianza which goes to any direction from that surface point.

Let’s suppose a scene is built with a finite set of \( N \) triangles:

- To each couple of polygons it is related a factor \( G_{ij} \), called geometric factor, defining how much of the Radiance exiting from the triangle \( i \) will go to the triangle \( j \)
- Instead of evaluating the **Radiance Function**, it is evaluated the Radiosity exiting from each triangle.

With this considerations **the rendering equation** ....
The **Radiosity Algorithm** is applied under the assumption all the surfaces have only a diffuse component. In this situations:

\[
L = \frac{c_{\text{diff}} E}{\pi}
\]  

(7)

Where \( E \) is the total irradiance falling on a surface, and \( c_{\text{diff}} \) is the coefficient of the diffuse reflection, \( L \) is the radianza which goes to any direction from that surface point.

Let’s suppose a scene is built with a finite set of \( N \) triangles:

- To each couple of polygons it is related a factor \( G_{ij} \), called geometric factor, defining how much of the radiance exiting from the triangle \( i \) will go to the triangle \( j \)
- Instead of evaluating the **Radiance Function**, it is evaluated the radiosity exiting from each triangle.

With this considerations the **rendering equation** .... becomes a **linear system with N equations** (each one defining the behavior of light on some triangle) for **N unknown variables** (the value of radiosity of each polygon).
The Radiosity Algorithm is able to evaluate con extreme precision the Rendering Equation when BRDF are purely diffuse, but it is not suitable with more complex BRDF.

*images generated with the Open Source Software PoV-Ray (PoV:Persistence of Vision) www.povray.org*