Whitepaper:
The Monte-Carlo Solver in ChiTech

August, 2019

Jan Vermaak

Rev 1.00

1 Basic transport

The fundamental transport of a particle in a medium, where the interaction probability is given by σ_t in units $[cm^{-1}]$, can be described as follows. Given a flux of particles, ϕ, in units $[cm^{-2}s^{-1}]$ the interaction rate per unit volume is given by

$$RR = \sigma_t \phi$$

This also represents the rate of change of ϕ as

$$\frac{d\phi}{dx} = -\sigma_t \phi$$

This forms a first order differential equation which we can multiply with the integration factor $e^{\sigma_t x}$

$$\begin{aligned}
e^{\sigma_t x}\frac{d\phi}{dx} + \sigma_t e^{\sigma_t x}\phi &= 0 \\
\therefore \frac{d}{dx}\left(e^{\sigma_t x}\phi\right) &= 0 \\
\int_0^L \frac{d}{dx}\left(e^{\sigma_t x}\phi\right).dx &= 0 \\
\left(e^{\sigma_t x}\phi\right)_0^L &= 0 \\
e^{\sigma_t L}\phi_L - \phi_0 &= 0 \\
\therefore \frac{\phi_L}{\phi_0} &= e^{-\sigma_t L}
\end{aligned}$$

The amount of interactions in distance L is then given by $\phi_0 - \phi_L$, and the probability of interacting, p_i within the distance L is given by

$$\begin{aligned}
p_i = \frac{\phi_0 - \phi_L}{\phi_0} &= 1 - \frac{\phi_L}{\phi_0} \\
\therefore p_i &= 1 - e^{-\sigma_t L}
\end{aligned}$$

This form of the interaction probability is a curve which starts at zero and assymptotically tends to unity. Additionally, this equation can be inverted by solving for L

$$\begin{aligned}
e^{-\sigma_t L} &= 1 - p_i \\
-\sigma_t L &= ln(1-p_i) \\
\therefore L &= -\frac{ln(1-p_i)}{\sigma_t} \qquad (1.1)
\end{aligned}$$

This equation can now be sampled in a given material by randomly sampling $p_i \in [0,1]$ which will produce the distance-to-interaction, d_i. We now denote any random number $\theta \in [0,1]$ and write the base equation for sampling the distance-to-interaction

$$d_i = -\frac{ln(1-\theta)}{\sigma_t} \tag{1.2}$$

Consider the multi-zone geometry in 1 below. For practical purposes it is important to be able to resample the distance-to-interaction from one zone to another. The question is however if the resampling of a multi-zone geometrical arrangement is equivalent to a single large zone. It can be proved statistically that the two sampling techniques are equivalent.

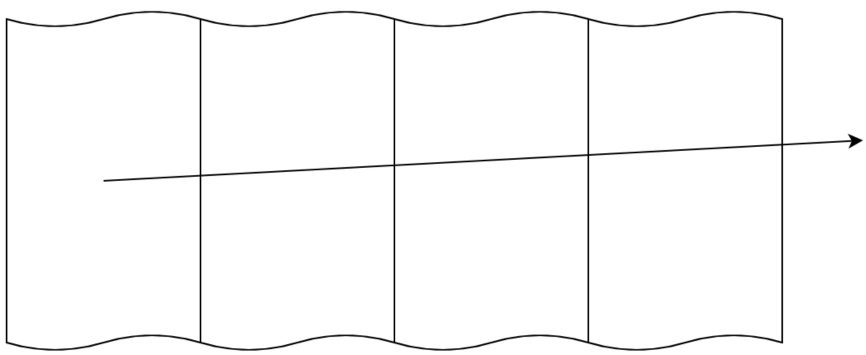

Figure 1: Multi-zone geometry.

The next quantity to compute is the distance-to-surface, d_s. This process we will call **ray-tracing**.

2 Raytracing utilities

2.1 Intersection of a line and a plane chi_mesh::CheckPlaneLineIntersect

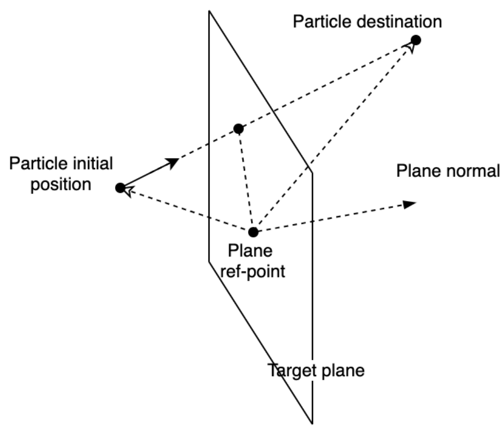

Figure 2: Graphical representation of intersection of a line with a plane.

Given a particle's location, $\bar{r}_i$, and direction, $\hat{\Omega}$, we can create a 3D line of the form

$$\bar{r} = \bar{r}_i + d\hat{\Omega}$$
$$(x, y, z) = (x_i, y_i, z_i) + d(\Omega_x, \Omega_y, \Omega_z)$$

where d is the only unknown parameter required to define $\bar{r} = (x, y, z)$. Alternatively we can be supplied by another point on the line, $\bar{p}_f$ and then define a weight, $w \in [0, 1]$, which can define a **line segment**

$$\bar{r} = w\bar{r}_i + (w - 1)\bar{r}_f$$
$$(x, y, z) = w(x_i, y_i, z_i) + (w - 1)(x_f, y_f, z_f)$$

For the equation of a plane we need a refence point, $\bar{p}_0$, and a normal, $\hat{n}$, after which we can form a vector from point $\bar{p}_0$ to $\bar{r}_p = (x, y, z)$. The plane is then defined by the relationship that the dot-product of this vector with the normal is zero,

$$\hat{n} \bullet (\bar{r}_p - \bar{p}_0) = 0$$
$$n_x(x - p_{0x}) + n_y(y - p_{0y}) + n_z(z - p_{0z}) = 0.$$

A very simple way to simultaneously determine whether the line intersects the plane and where the intersection is, is to use the segment definition of the line and compute two vectors; one from $\bar{p}_0$ to $\bar{r}_i$, and another from $\bar{p}_0$ to $\bar{r}_f$

$$\vec{v}_i = \bar{r}_i - \bar{p}_0 \quad \text{and} \quad \vec{v}_f = \bar{r}_f - \bar{p}_0$$

after which we compute the projection of these vectors along the normal of the plane by taking the dot-products

$$D_i = \hat{n} \bullet \vec{v}_i \quad \text{and} \quad D_f = \hat{n} \bullet \vec{v}_f.$$

Since we use the same normal for both computations the line is intersecting the plane only if the signs of the dot-products is not equal and w can then be computed as

$$\text{if} \quad \text{sgn}(D_i) \neq \text{sgn}(D_f)$$

$$w = \frac{|D_i|}{|D_i| + |D_f|}$$

Implementation

```
bool chi_mesh::
CheckPlaneLineIntersect(chi_mesh::Normal plane_normal,
                        chi_mesh::Vector plane_point,
                        chi_mesh::Vector line_point_0,
                        chi_mesh::Vector line_point_1,
                        chi_mesh::Vector& intersection_point,
                        std::pair<double,double>& weights)
{
  chi_mesh::Vector v0 = line_point_0 - plane_point;
  chi_mesh::Vector v1 = line_point_1 - plane_point;

  double dotp_0 = plane_normal.Dot(v0);
  double dotp_1 = plane_normal.Dot(v1);

  bool sense_0 = (dotp_0 >= 0.0);
  bool sense_1 = (dotp_1 >= 0.0);

  if (sense_0 != sense_1)
  {
    double dotp_total = std::fabs(dotp_0) + std::fabs(dotp_1);
    weights.first = (std::fabs(dotp_0)/dotp_total);
    weights.second = 1.0 - weights.first;
    intersection_point =
      line_point_0*weights.second +
      line_point_1*weights.first;

    return true;
  }

  return false;
}
```

References

[1] *Blender - a 3D modelling and rendering package*, Blender Online Community, Blender Foundation, Blender Institute, Amsterdam, 2018

[2] Cheng et al, *Delaunay Mesh Generation*, Chapman & Hall/CRC Computer & Information Science Series, 2013