Chapter2 What is a Ray?

第二章 什么是一条光线？

2.0 摘要 ABSTRACT

本章定义了一条*光线*（ray），展示了如何使用*光线间隔*（ray intervals），并演示了如何使用 DXR（Windows 图形学 API）追踪指定的光线。

2.1 光线的数学描述 MATHEMATICAL DESCRIPTION OF A RAY

对于光线追踪，一个重要的可计算结构是三维光线。在数学和光线追踪中，光线通常指的是三维射线，一条射线通常被指定为一条直线上的一个区间。三维空间的直线没有类似于直线在二维空间的隐式方程表示（y = mx + b），因此通常使用参数形式（式 2-1）。在本章中，所有的线、点和向量都假定为 3 维。

一条参数化的线段可以表示为点 A 和 B 的加权平均值。

$$P(t) = (1 - t) * A + t * B \qquad (2\text{-}1)$$

在编程时，我们可以将此公式视为一个函数 P(t)，它以实数 t 作为输入并返回一个点 P。对于直线，参数可以取任意实数值，即 $t \in [-\infty, +\infty]$，点 P 随着 t 的变化沿直线不断移动，如图 2-1 所示。

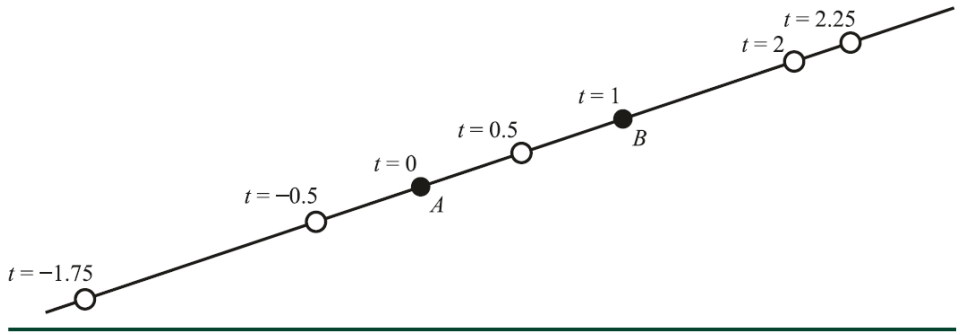

图 2-1 不同的 t 值如何在直线上给出不同的点

为了实现这个功能，我们需要一种方法来表示点 A 和 B。通常使用笛卡尔坐标（在 API 和编程语言中，这种表示通常称为 vec3 或 float3，包含三个实数 x、y 和 z）。同一条线可以用沿线的任意两个不同的 A、B 点来表示，但是选择不同的 A、B 点会相应的改变位置的 t 值。

更通用的做法是使用通常使用一个点和一个方向向量来表示一条线。如图 2-2 所示，可以选择（B – A）向量作为光线方向 d，我们的点 A 作为光线原点 O，可得公式 2-2：

$$\mathrm{P(t)} = O + \mathrm{t} * d \tag{2-2}$$

注意，t 为标量，d 为向量。

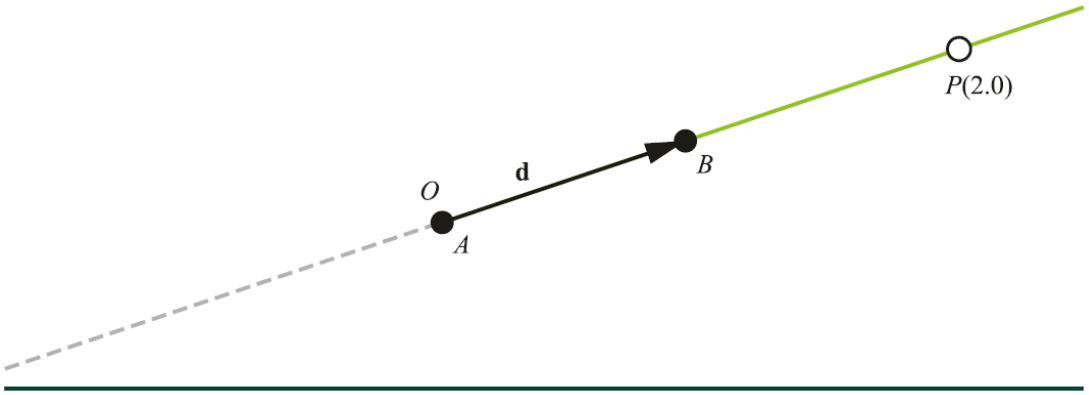

图 2-2 一条射线 P(t) = O + td

上图射线由原点 O 和射线方向 d 描述，在这种情况下为 d = B - A。我们通常只对正向的点感兴趣（根据光线的物理性质），即找到的点位于原点的前面（t >0）。

为了方便计算（比如计算向量之间的余弦），通常将方向向量归一化为单位向量 ˆd，归一化方向向量的一个好处是任何两个点 t 值的差异就是点之间的实际距离，即：

$$|| \ P(t1) - P(t2) \ || \ = \ |t1 - t2| \qquad (2\text{-}3)$$

2.2 光线间隔 RAY INTERVALS

公式 2-2 将光线描绘成一条半无限长的线。但是在光线追踪中，光线通常带有一个额外的区间：*交点*（intersection）有意义的 t 值范围。通常，我们将此区间指定为两个值，tmin 和 tmax， t 应当在[tmin, tmax]区间内。换句话说，如果在 t 处找到一个相交点（光线场景/物体的相交点），但 t < tmin 或 t > tmax，则不会报告该相交点。见图 2-3。

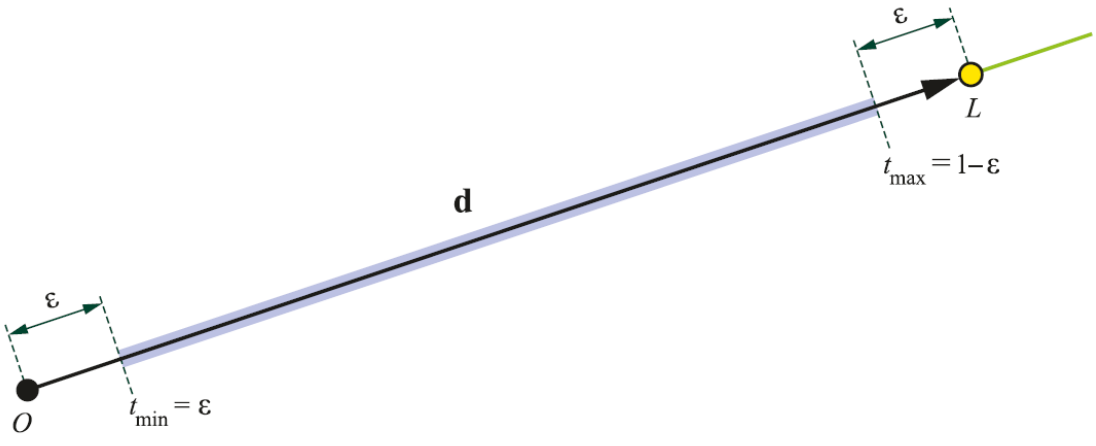

图 2-2 光线间隔示例

在上图示例中，L 处有一个光源，我们只想搜索 O 和 L 之间的相交点。光线间隔 [tmin, tmax] 用于将 t 值的交点搜索范围限制为 [tmin, tmax]。通过将光线间隔设置为 [ε, 1 - ε] 来实现为避免精度问题，，在此图中以浅蓝色显示间隔。

设定 tmax 的目的是，在某些情况下，对超出 tmax 的求交点是无意义的（比如在某点 p 向光源发出 shadow ray 判断 P 是否在阴影中，只有 P 点和光源点之间的交点是有意义的）。假设光源位置为 L，以一条原点为 O = P、非归一化方向矢量 d = L - P、tmin = 0 和 tmax = 1 的阴影射线为例。如果交点的 t 属于 [0, 1] ，则光线与遮挡光线的几何体相交。在实践中，通常设置 tmin = ε 和 tmax = 1 - ε，ε 为极小数。这种调整有助于避免由于数值精度问题而导致的*自相交*（*self-intersections*）；使用浮点表示，P 所在的表面可能与我们的射线相交于一个小的非零值 t。对于非点光源，光源的基元不应阻挡住 shadow ray，因此我们使用 tmax = 1 - ε 较小光线间隔。如果是理想数学，这个问题可以忽略，找（0，1）区间的交点即可。由于浮点精度有限的，使用 ε 因子是一个常见的解决方案。有关如何避免自相交的更多信息，请参阅第 6 章。

若使用归一化的方向向量 d，tmin = ε, tmax = length − ε，此时 length = ‖L − P‖，P 到 L 的距离。注意，ε 的取值与使用非归一化向量 d 时不同，因为 t 的数值范围改变了。

一些渲染器对所有或部分光线方向使用单位长度向量。这样可以通过与其他单位向量的点积进行快速计算余弦，并且可增加代码可

读性。如前所述，单位长度方向向量意味着光线的参数 t 可以解释为距离。

然而，对于实例化（图形学技术，几何体的顶点数据只保存一份通过变换矩阵渲染出多个位置不同的此几何体）的几何图形可以使用每个实例的变换矩阵来表示。在对光线与物体进行求交需要将光线转换到物体空间中，此转换会改变方向向量 d 的长度。此时转换后的光线方向是未归一化的。此外，归一化会消耗一点性能并且可能是不必要的，就像阴影光线一样。

由于上述原因，对于是否使用单位方向向量不给出建议。

2.3 DXR 中的光线

本节介绍 DirectX Raytracing 中光线的定义。在 DXR 中，光线由以下结构定义：

```
1.  struct RayDesc
2.  {
3.  float3 Origin;
4.  float TMin;
5.  float3 Direction;
6.  float TMax;
7.  };
```

要在 DXR 中使用 TraceRay() 函数跟踪光线，需要使用 RayDesc。 RayDesc::Origin 设置为我们射线的原点 O，RayDesc::Direction 设置为方向 d，并且 t 间隔（RayDesc::TMin 和 RayDesc::TMax）也必须初始化。例如，对

于一条眼睛射线 (RayDesc eyeRay)，我们设置 eyeRay.TMin = 0.0 和 eyeRay.TMax = MAX，这表明我们对位于原点前面的所有交点感兴趣。

2.4 结论

本章展示了光线在光线追踪器中的典型定义和使用方式，并以 DXR API 的光线定义为例。其他光线追踪系统，例如 OptiX 和 Vulkan 光线追踪扩展，有细微的变化。例如，OptiX 明确定义了一种光线类型，例如阴影光线。这些系统还有其他的共同点，例如光线 payload 的设计。payload 是一种数据结构，用户可以定义它来使光线携带附加信息以及可以由单独的着色器或模块访问和编辑。此类数据是特定于应用程序的。重点是，在每个定义了光线的渲染系统中，您都会找到射线的原点、方向和间隔。

参考文献：见原文

unofficial_RayTracingGems_v1.4.pdf (realtimerendering.com)