从光栅化到光线追踪

计算机图形学简介

光栅化

光线追踪

一些 RUST 特性

计算机图形学

“在计算机里除了文字与声音以外的一切”

几何	• 计算几何
渲染	• 光栅化 • 光线追踪
模拟	• 建模
……	

光栅化

渲染流水线

数据

- 顶点、材料、纹理

几何阶段

- 坐标变换、裁剪、屏幕映射

光栅化阶段

- 为每个图元计算覆盖了哪些像素
- 为这些像素计算它们的颜色

屏幕图像

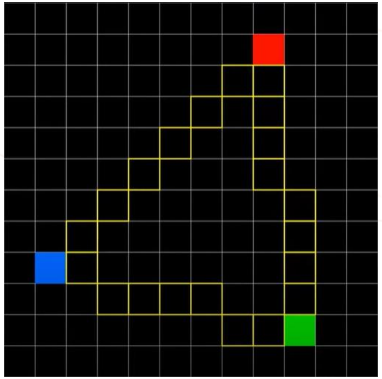
光栅化阶段

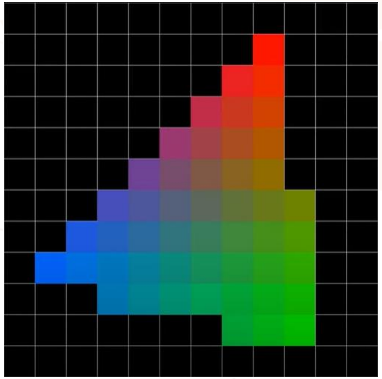
三角形设置

三角形遍历

片元着色器

逐片元操作





- 每个像素，深度、颜色、纹理采样
- 决定每个像素的最终颜色

光线追踪

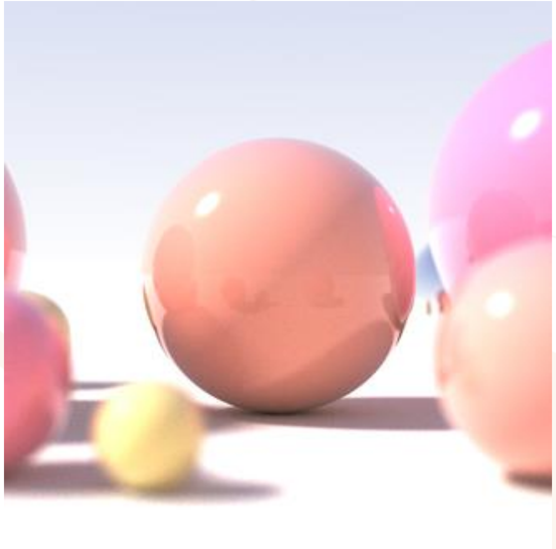
光线追踪

基本概念

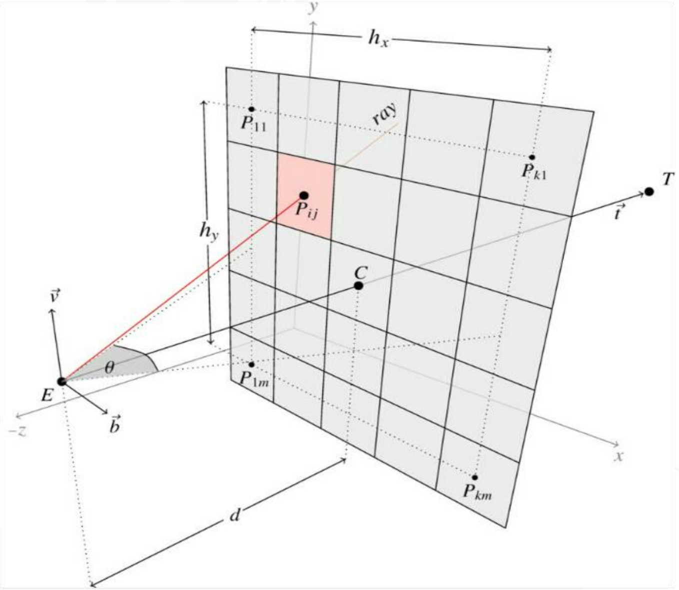
- 光线发出
- 反射与折射
- 材质

主要优化

- Bvh 树
- PDF
- ……



发射光线



y
h_x
ray
P_{11}
P_{k1}
P_{ij}
h_y
T
$\vec{t}$
C
$\vec{v}$
θ
E
P_{1m}
$\vec{b}$
-z
x
P_{km}
d

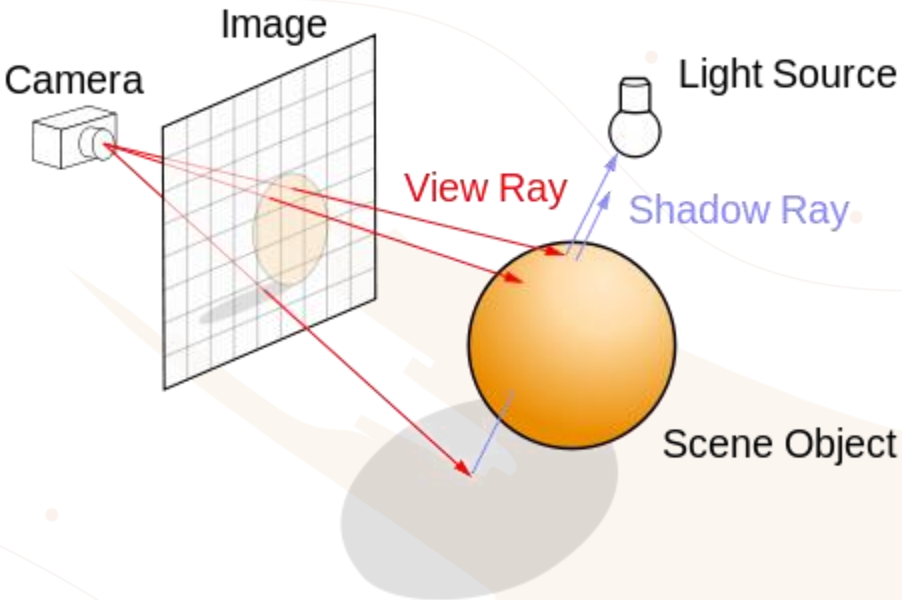


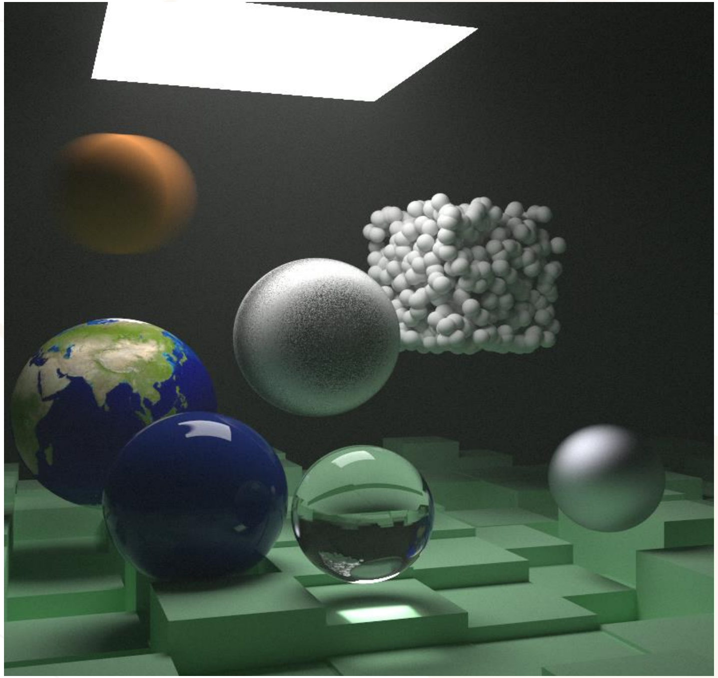
Image
Camera
Light Source
View Ray
Shadow Ray
Scene Object

光线如何击中物体

- HittableList

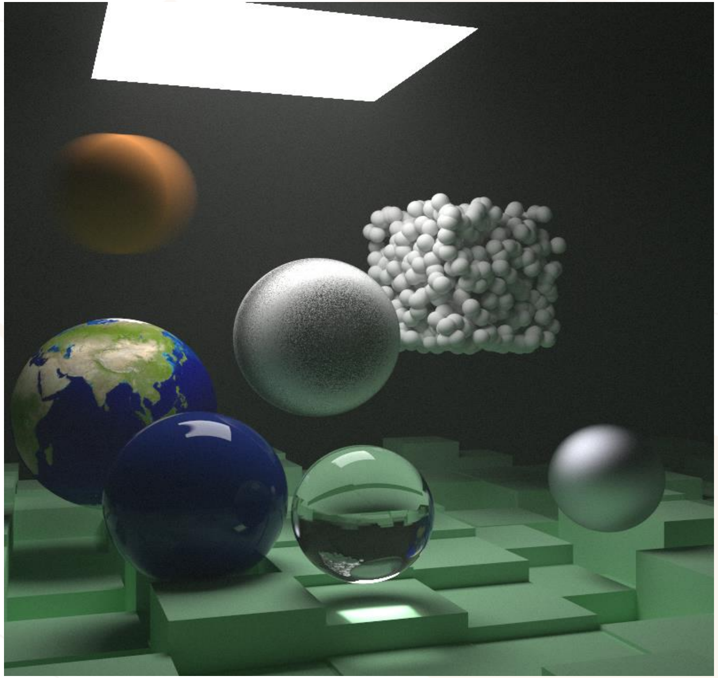
遍历每个物体

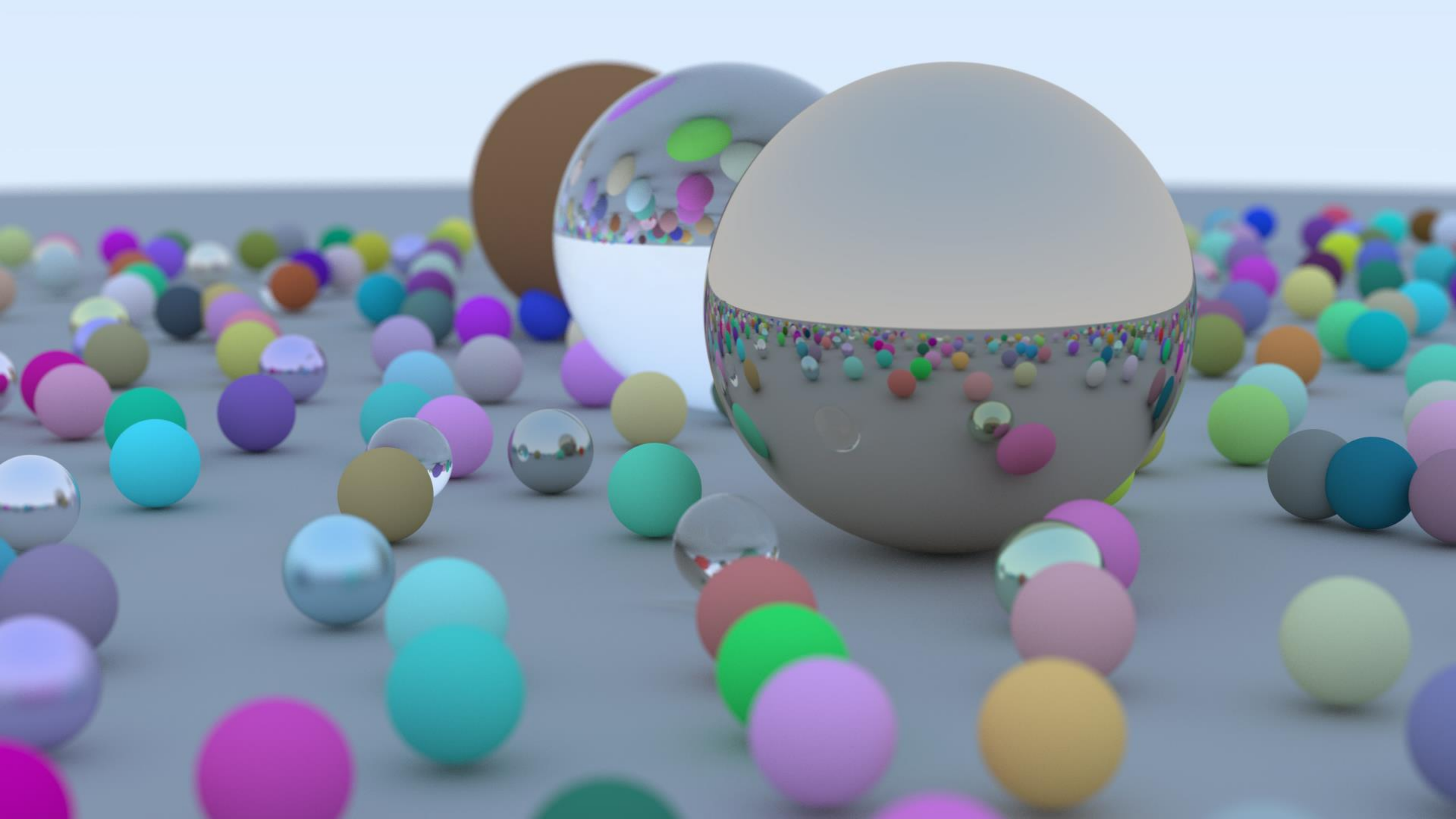
暴力判断是否相交



当光线击中一点

- 根据物体形状算出法线
- 根据物体材质计算出反射/折射光
  - Lambertian 朗伯体
  - Metal 金属
  - Dielectric 电介质
  - Isotropic 各向同性（雾）
  - DiffuseLight 光源





光栅化与光线追踪的区别

精密计算、补光补影
来达到理想效果

人工渲染

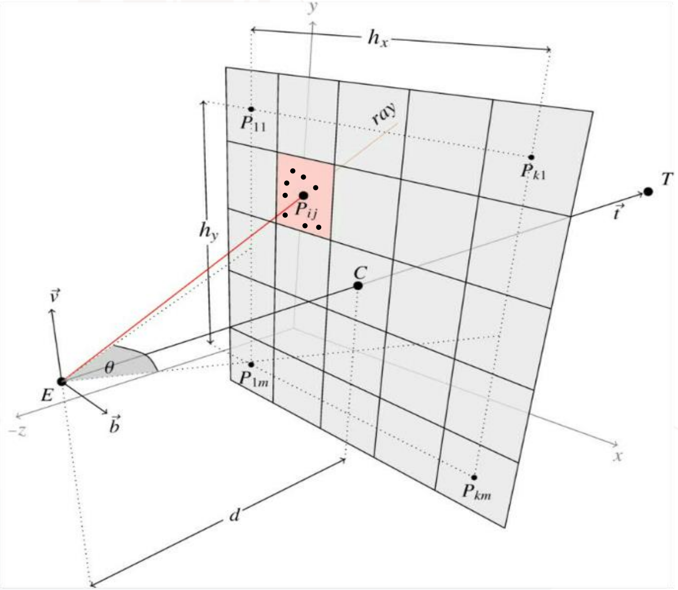
or

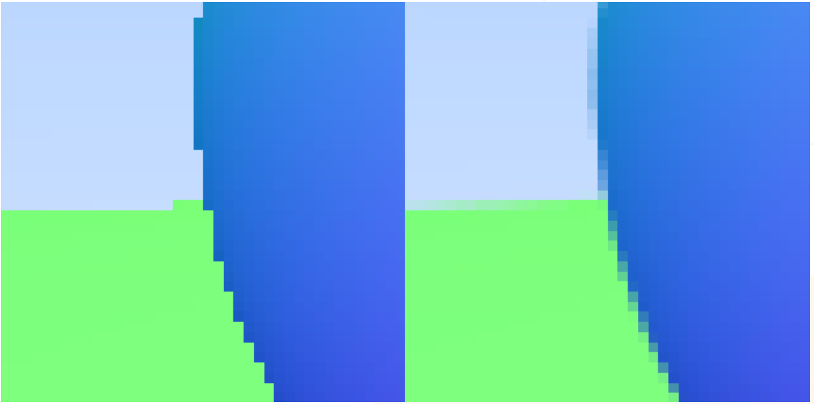
计算机渲染

“让光飞一会儿”

抗锯齿优化

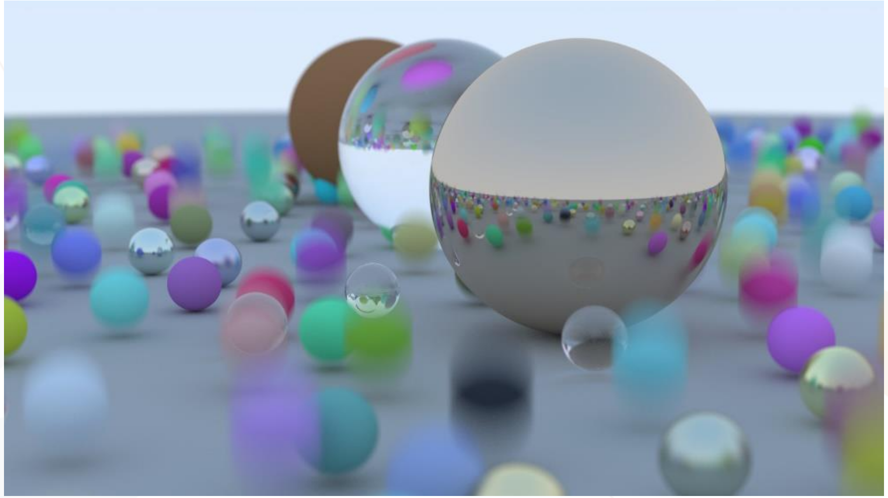
- 同一像素采样多次，所得的结果取平均值





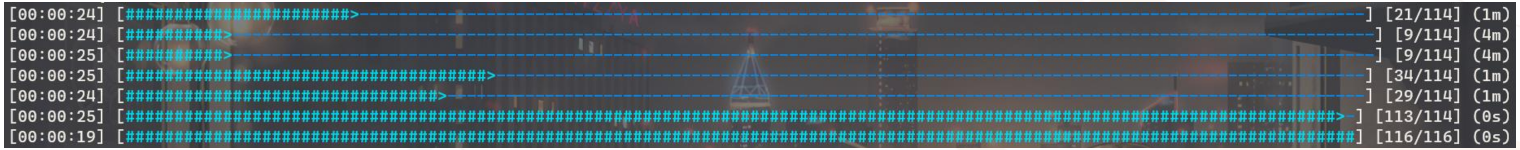
运动效果

- 增加运动物体类，不同时间位于不同位置
- 随机设定照相机采样时间 t，采样当前时刻的物体位置

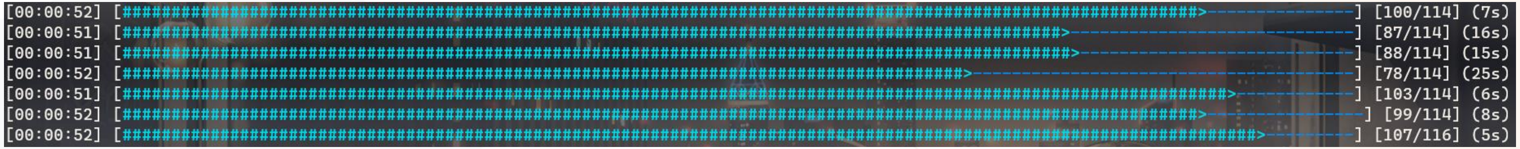


多线程优化

- 多线程支持
- 不同像素的渲染是独立的
- 将你的世界复制 N 次
- 以行为单位，分给不同的线程去渲染

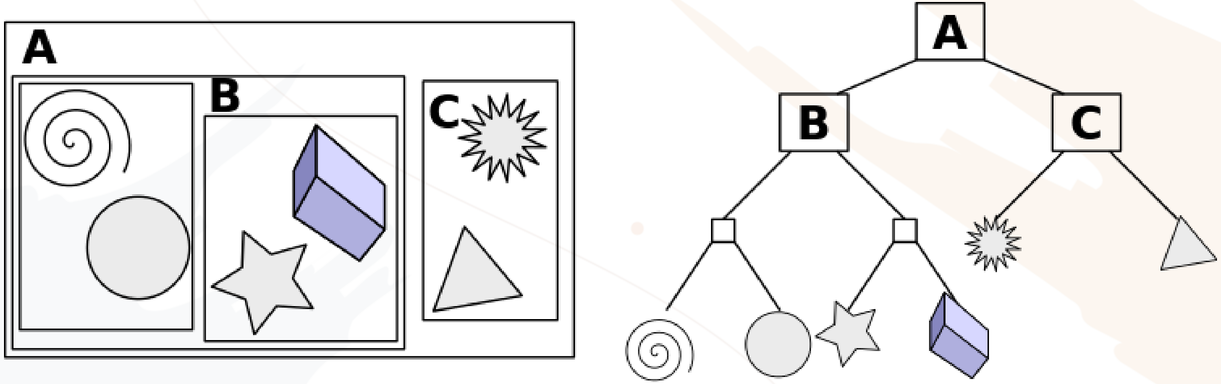


- 以行为单位，将不同的行**随机**分给不同的线程去渲染



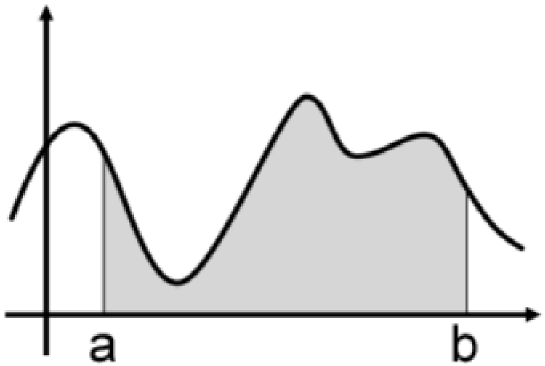
BVH Tree

- 将物体用 bounding box 包起来
- 用分治的思想建树，父节点的范围是所有子节点范围的并
- 当父节点未被击中时，它的所有子节点也一定不会被击中



蒙特卡洛积分

- 蒙特卡洛积分

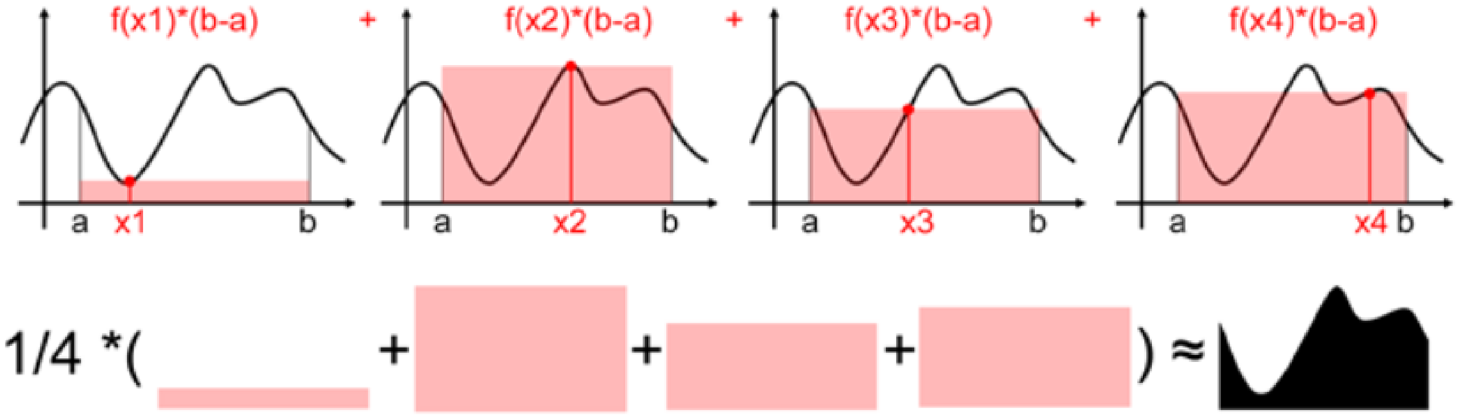


被积函数 $f(x)$，积分区间 $[a, b]$

求其积分

蒙特卡洛积分

蒙特卡洛方法：采样 4 次并求平均值



N 次采样： $< F^N > = (b-a)\frac{1}{N}\sum_{i=0}^{N-1} f(Xi)$

蒙特卡洛积分

- 收敛太慢?
- 利用 概率密度函数

生成满足概率密度函数分布的随机数 Xi

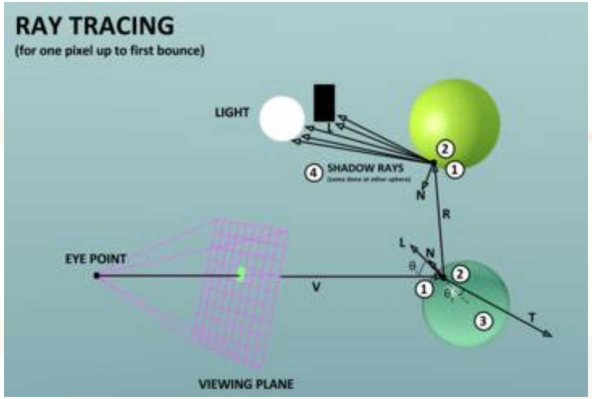
N 次采样:
$$< F^N > = \frac{1}{N}\sum_{i=0}^{N-1}\frac{f(Xi)}{pdf(Xi)}$$

- 求函数积分值时，选取恰当的概率密度函数（PDF），让结果更快收敛

概率密度函数优化

- 采样数少的情况下减少噪点?
- 利用 概率密度函数

- 噪点的本质是发射出的光线没有击中光源
- 利用 PDF，提高像光源发射光线的概率



边缘检测

- 用水平方向和竖直方向上的边缘信息，在进行边缘检测时，对每个像素分别进行一次卷积计算，得到两个方向上的梯度值 G_x 和 G_y，获得整体梯度

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

- 梯度越大，越有可能是边缘点一个是基于深度计算的梯度，另一个是基于颜色计算的梯度

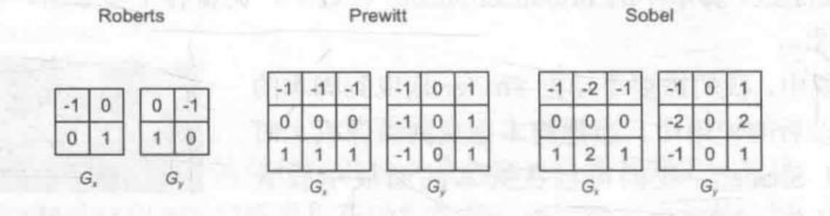
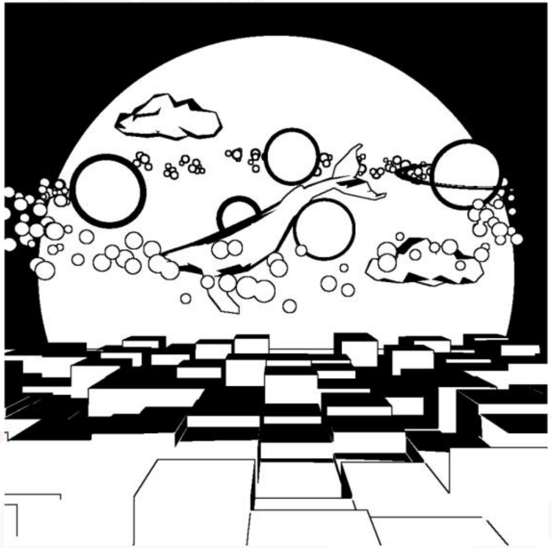
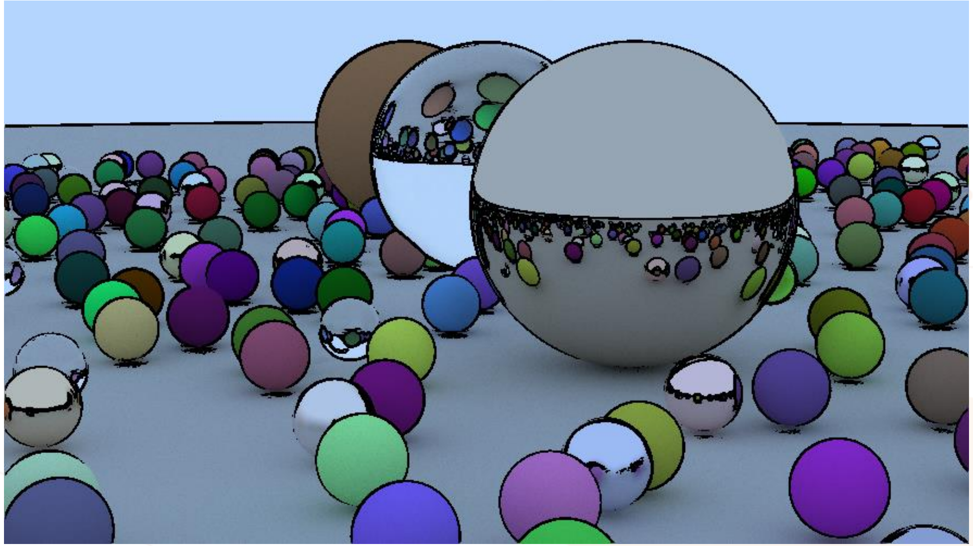


图 12.5　3 种常见的边缘检测算子

边缘检测

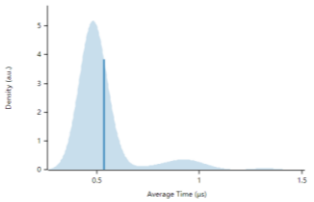


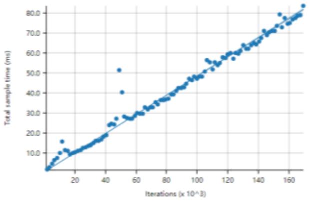


benchmark

- 向图像中心位置(width/2, height/2)发射光线100次得到的benchmark

Ray test





Additional Statistics:

	Lower bound	Estimate	Upper bound
Slope	478.62 ns	482.93 ns	488.32 ns
R^2	0.5392939	0.5423378	0.5375748
Mean	507.46 ns	534.99 ns	566.81 ns
Std. Dev.	101.04 ns	153.62 ns	199.02 ns
Median	474.94 ns	478.16 ns	485.31 ns
MAD	11.318 ns	16.254 ns	22.683 ns

Additional Plots:

- Typical
- Mean
- Std. Dev.
- Median
- MAD
- Slope

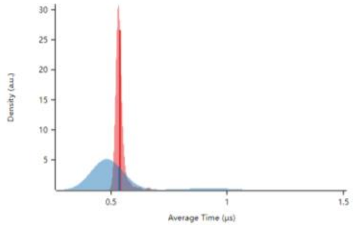
Understanding this report:

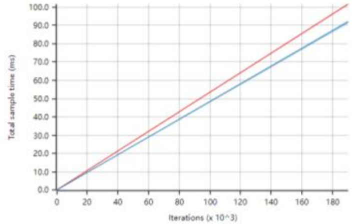
The plot on the left displays the average time per iteration for this benchmark. The shaded region shows the estimated probability of an iteration taking a certain amount of time, while the line shows the mean. Click on the plot for a larger view showing the outliers.

The plot on the right shows the linear regression calculated from the measurements. Each point represents a sample, though here it shows the total time for the sample rather than time per iteration. The line is the line of best fit for these measurements.

See the documentation for more details on the additional statistics.

Change Since Previous Benchmark





Additional Statistics:

	Lower bound	Estimate	Upper bound	
Change in time	-5.8761%	-0.6489%	+5.2814%	(p = 0.82 > 0.05)

No change in performance detected.

Additional Plots:

- Change in mean
- Change in median
- T-Test

Understanding this report:

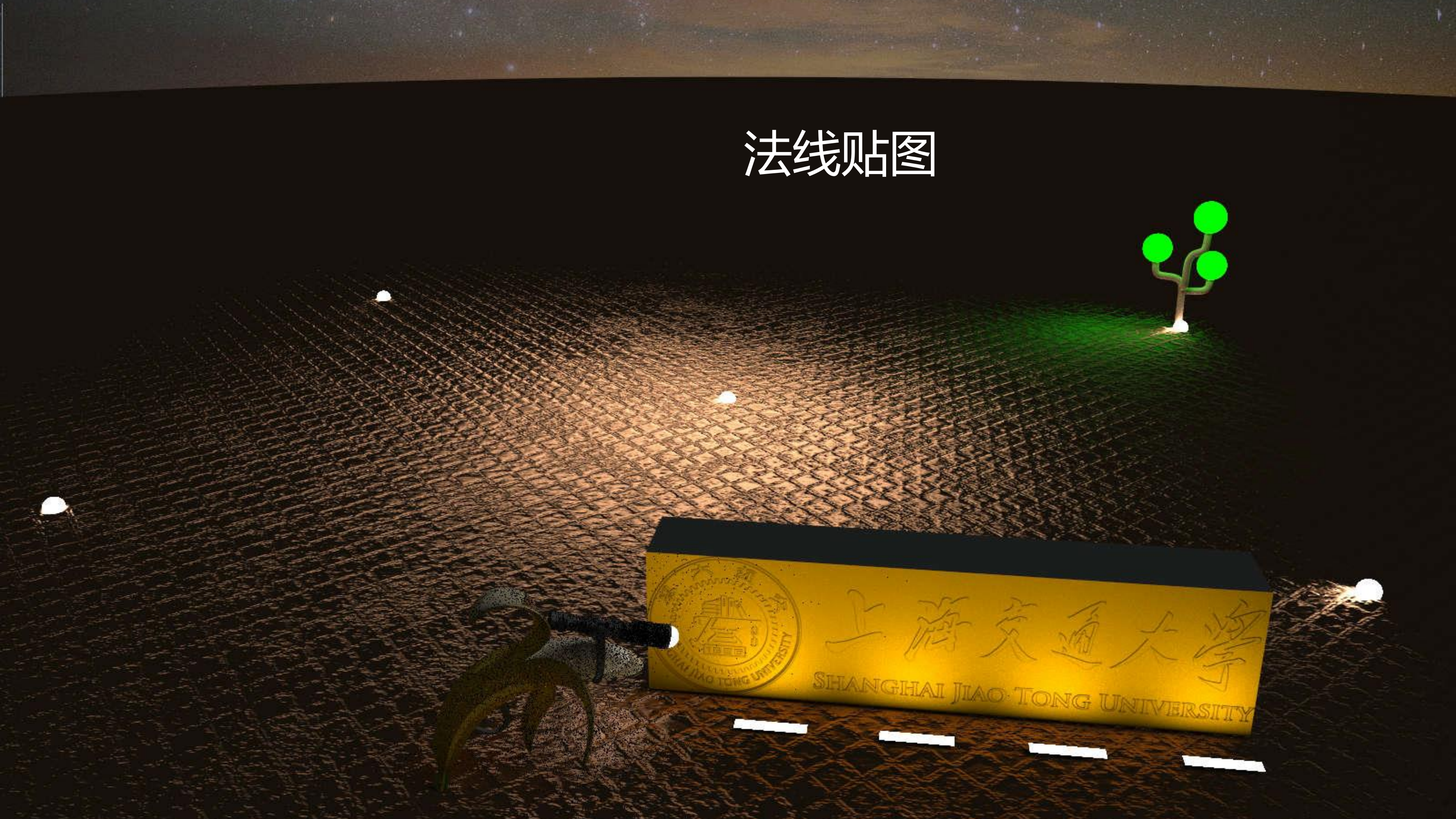
The plot on the left shows the probability of the function taking a certain amount of time. The red curve represents the saved measurements from the last time this benchmark was run, while the blue curve shows the measurements from this run. The lines represent the mean time per iteration. Click on the plot for a larger view.

The plot on the right shows the two regressions. Again, the red line represents the previous measurement while the blue line shows the current measurement.

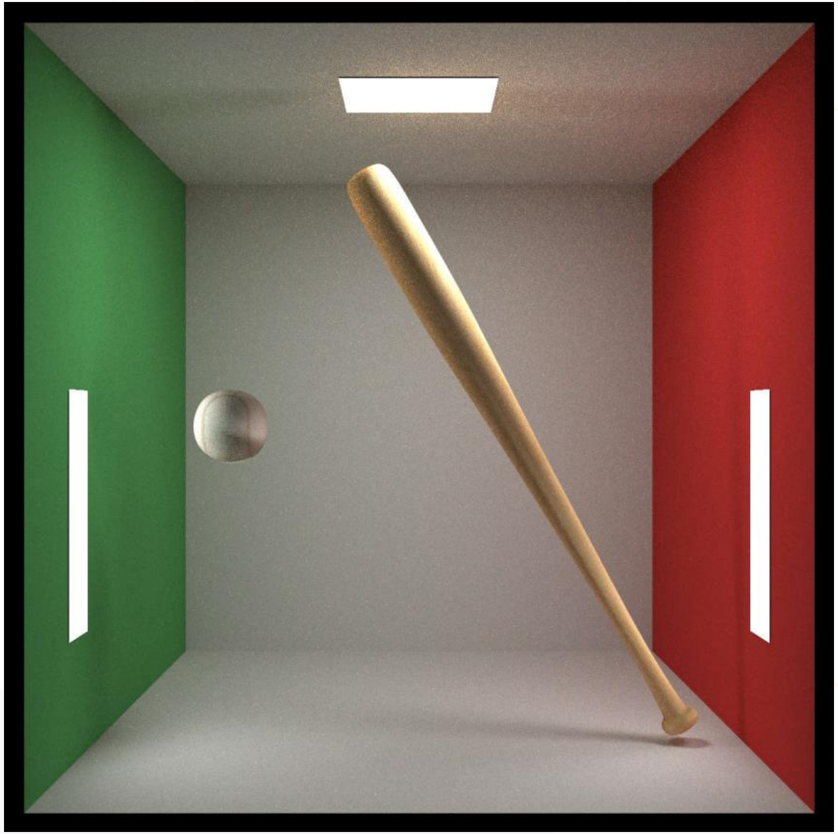
See the documentation for more details on the additional statistics.

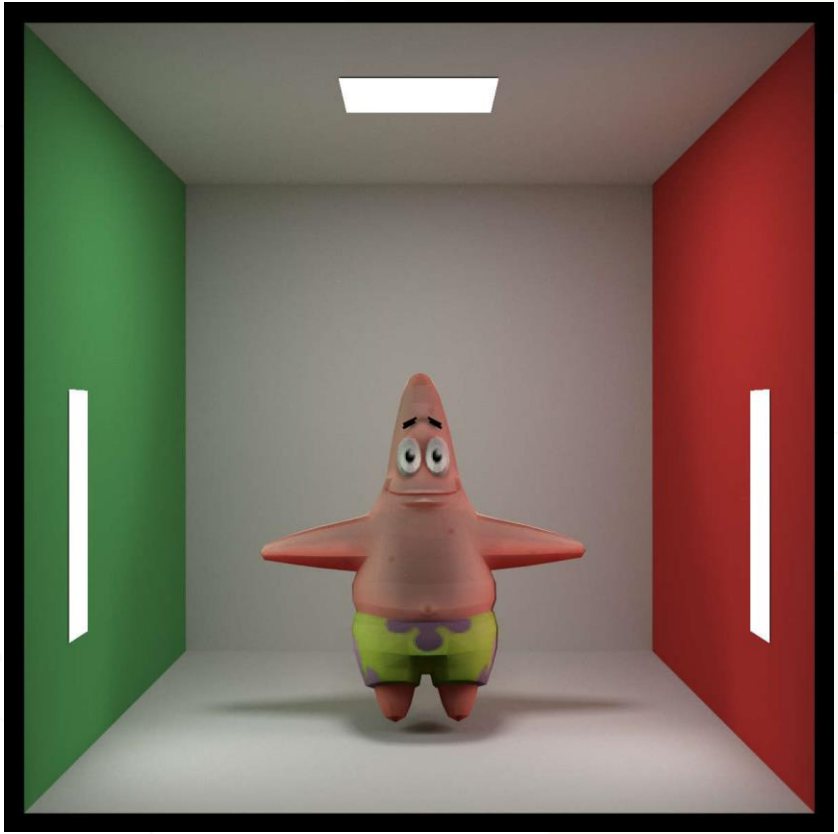
- 光线发射时间-密度分布
- 随着迭代次数增加，总共需要的时间

法线贴图



双线性插值





RUST 特性

➢ 生命周期

```rust
fn main() {
  let a;                  // -------------+-- a start
  {                       //              |
    let b = 5;            // -+-- b start |
    a = &b;               //  |           |
  }                       // -+-- b over  |
  println!("a: {}", a);   //              |
}                         // -------------+-- a over

```

RUST 特性

- 生命周期参数

```rust
fn max_num<'a>(x: &'a i32, y: &'a i32) -> &'a i32 {
    if x > y {
        return &x;
    } else {
        return &y;
    }
}
```

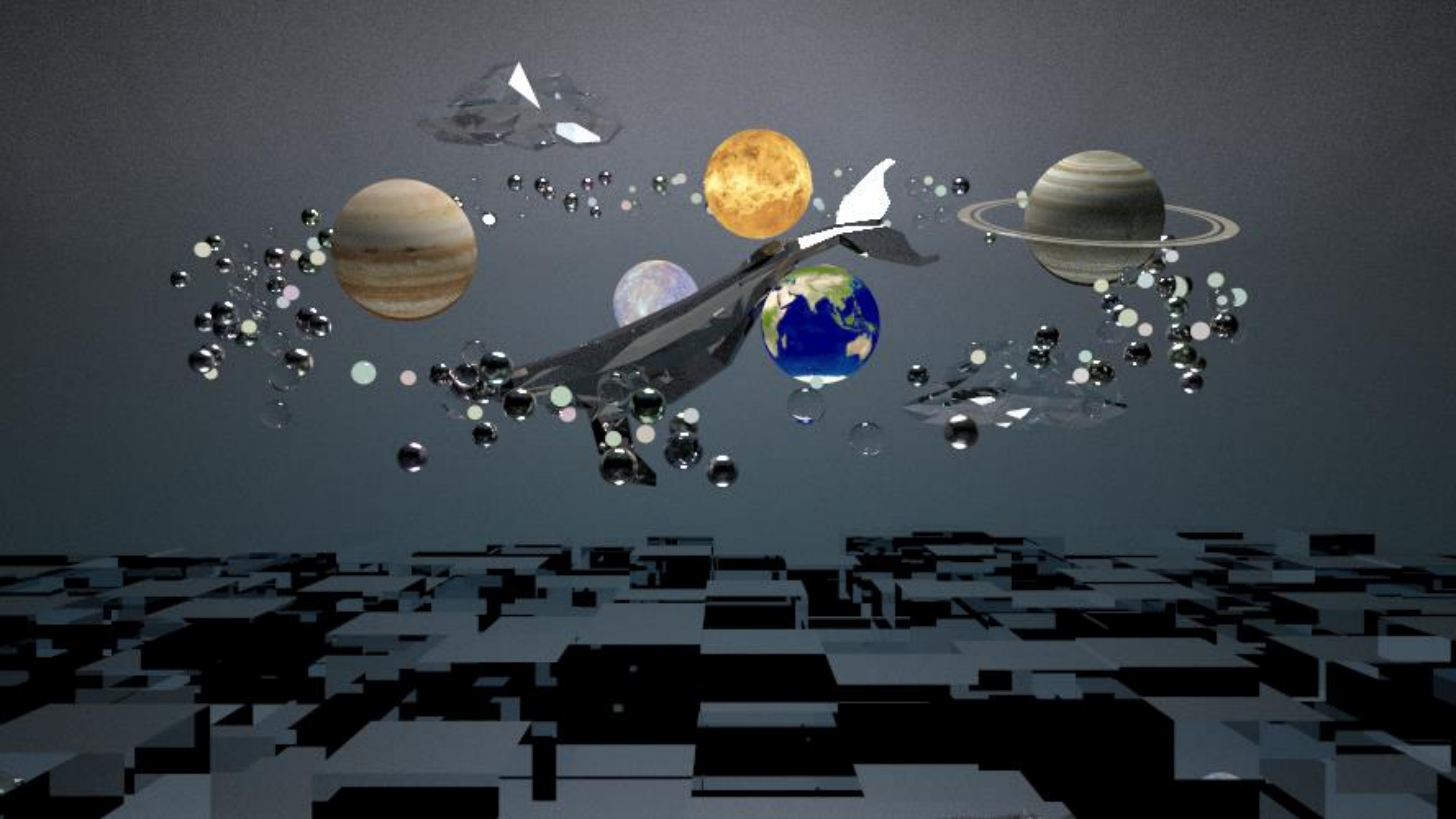
RUST 特性

➢ 生命周期参数

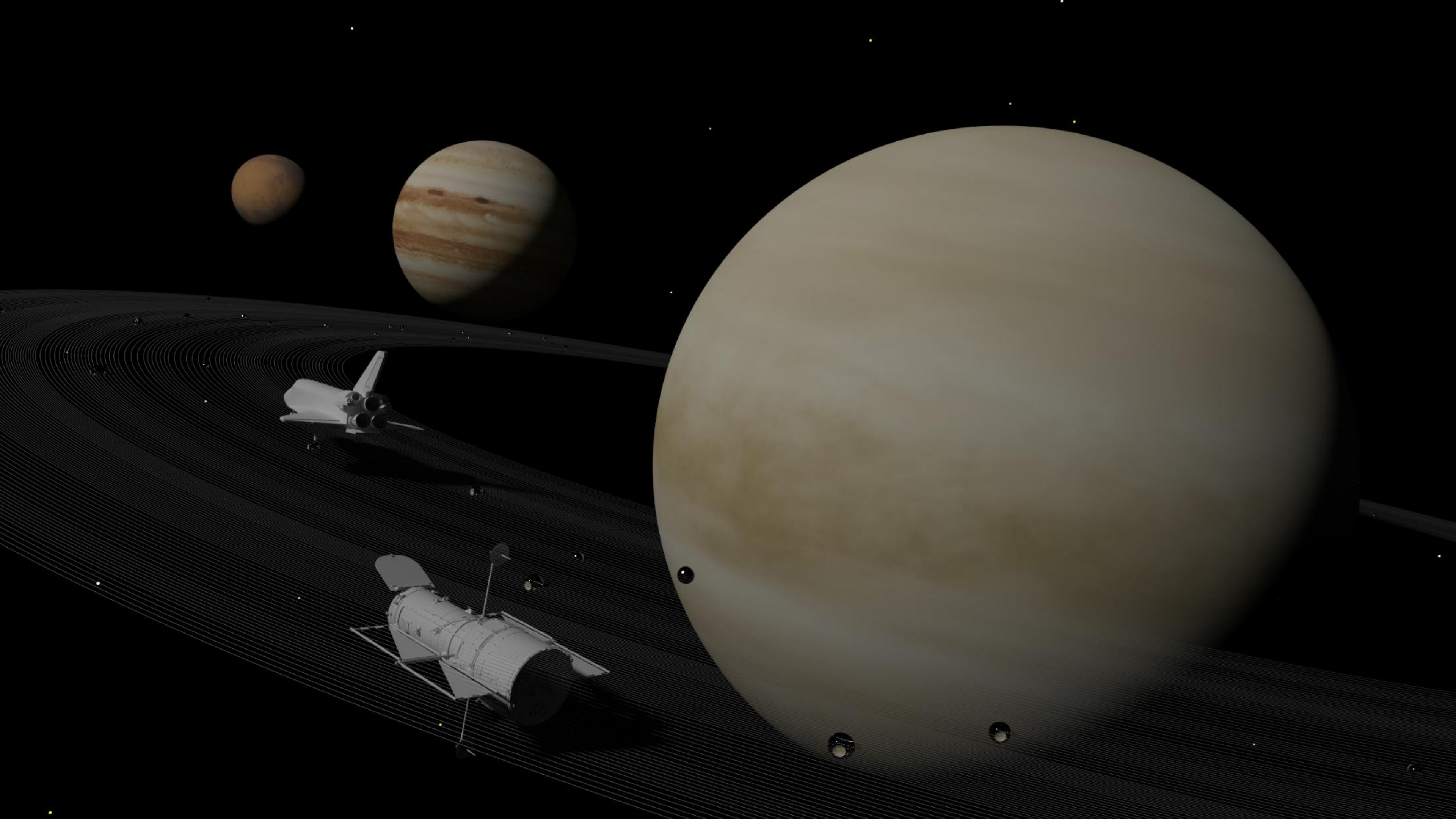
```rust
struct Foo<'a> {
  v: &'a i32
}

```

RUST 特性

- Trait 特性 而非继承树
- 严格的编译检查
- 不显示声明就没有的特性
- 让错误无处可逃




DAFTPU




AC 021
RAY T ACER
oduced by hnyls2002
SHANGHAI JIAO TONG UNIVERSITY

THANKS FOR LISTENING