sample函数说明

输入：`dir`：光线的方向

1：计算球面坐标（经纬角）

```
vec3 d = dir.normalized();
float phi = atan2(d.z, d.x);   // [-π, π]，水平角度（经度）
float theta = acos(d.y);       // [0, π]，竖直角度（纬度）
```

- `phi` 是绕地球转一圈的方向（绕 y 轴）
- `theta` 是你从北极看向赤道的角度（从 y=1 向下看）

2：转换为 UV 坐标（映射到 [0,1]）

```
// 把 [-π, π] 映射到 [0, 1]
float u = (phi + M_PI) / (2 * M_PI);
// 把 [0, π] 映射到 [0, 1]
float v = theta / M_PI;
```

这将球面坐标转换为图像上的二维坐标。

3：映射到像素坐标

```
int x = std::min(int(u * width), width - 1);
int y = std::min(int(v * height), height - 1);
```

- 把 `u, v` 转为图像的整数像素位置
- 加上 `min()` 防止越界

4：读取 RGB 像素值

```
int idx = (y * width + x) * 3;
return vec3{
    image_data[idx]     / 255.f,
```

```
    image_data[idx + 1] / 255.f,
    image_data[idx + 2] / 255.f
};
```

- 每个像素占 3 字节（RGB）
- 除以 `255.f` 将 `uint8_t` 色值标准化到 `[0.0, 1.0]`

返回该方向对应的背景颜色

可以通过这段代码根据任意方向光线返回对应颜色，实现背景。