

作业 3: Screen Space Ray Tracing

GAMES202, 2021 年春季

教授: 闫令琪

计算机图形学与混合现实研讨会

GAMES: Graphics And Mixed Environment Seminar

发布日期为北京时间 2021 年 6 月 1 日 (星期二) 上午 10:00

截止时间为北京时间 2021 年 6 月 12 日 (星期六) 下午 8:00

注意:

- 任何更新或更正都将发布在论坛上，因此请偶尔检查一下。
 - 论坛链接 <http://games-cn.org/forums/forum/games202/>
 - 你必须独立完成自己的作业。
 - 你可以在论坛上发布帖子求助，但是发布问题之前，请仔细阅读本文档。
 - 在截止时间之前将你的作业提交到 SmartChair 上。
-

1 总览

本轮作业中，我们需要在一个光源为方向光，材质为漫反射 (Diffuse) 的场景中，完成屏幕空间下的全局光照效果（两次反射）。

为了在作业框架中实现上述效果，基于我们需要的信息不同我们会分三阶段着色，每个阶段都有相对应的任务。第一次着色负责计算 Shadow Map 所需的深度值并保存到贴图中。第二次着色负责计算屏幕空间中，每个像素对应的世界坐标系下位置、世界坐标系下法线、漫反射反射率和可见性信息并最终保存到对应贴图中。第三次着色基于之前得到的场景几何信息（像素对应的位置，法线），场景与光源的遮挡信息（光源坐标系的深度值），场景的材质信息（漫反射反射率），来计算两次反射的全局光照结果。

本轮作业的主要工作将集中在第三次着色中。

注意：一般情况下，我们应该在第二次着色时计算直接光照信息并保存下来，在之后计算间接光照时进行查询。不过，为了实现方便我们将这部分工作也转移到了第三次着色中完成。

2 实验流程

对于本次实验，有如下三个任务点需要依次完成：

- 实现对场景直接光照的着色（考虑阴影）。
- 实现屏幕空间下光线的求交 (SSR)。
- 实现对场景间接光照的着色。

如果你能确保每一步完成正确，最终我们可以得到质量不错的光照效果！

2.1 直接光照

在这一步里，你会开始熟悉如何使用提供的一些函数，以及 GBuffer 信息。你需要实现位于 `shaders/ssrShader/ssrFragment.gls` 文件中的 `EvalDiffuse(wi,`

`wo, uv)` 和 `EvalDirectionalLight(uv)` 函数，并在 `main` 函数中实现直接光照的效果。

以下的 `uv` 表示着色点的屏幕空间坐标，即值域为 $[0, 1]^2$ 。

`EvalDiffuse(wi, wo, uv)` 函数的返回值为 BSDF 的值。参数 `wi` 和 `wo` 为世界坐标系中的值，分别代表入射方向和出射方向，需要注意的是，这两个方向的起点都为着色点。在这一步中你会需要：漫反射率和法线信息。我们提供了 `GetGBufferDiffuse(uv)` 和 `GetGBufferNormalWorld(uv)` 来获取漫反射率和法线。这里需要注意，`GetGBufferDiffuse(uv)` 返回的值是在线性空间的，`GetGBufferNormalWorld(uv)` 得到的法线是在世界坐标系的。为了获取屏幕空间的坐标，我们提供了 `GetScreenCoordinate(posWorld)` 函数，它的参数为世界坐标系中的位置。

`EvalDirectionalLight(uv)` 函数的返回值为，着色点位于 `uv` 处得到的光源的辐射度，并且需要考虑遮挡关系 (Shadow Map)。在第二次着色的时候我们已经通过简单地比较深度得到了可见性信息，并保存到了贴图中。这里可以使用 `GetGBufferuShadow(uv)` 函数得到。你也可以修改 `src/shaders/gbufferShader/g-bufferFragment.glsl` 文件中的 `SimpleShadowMap(posWorld, bias)` 来实现其他阴影算法。

`main()` 函数中，第一行我们使用 `InitRand()` 函数来初始化随机数，得到随机数的状态 `s`。之后你可以调用 `Rand1(out s)` 和 `Rand2(out s)` 来得到类型为 `float` 和 `vec2` 的随机数。

2.2 Screen Space Ray Tracing

在这一步里，你需要实现位于 `shaders/ssrShader/ssrFragment.glsl` 文件中的 `bool RayMarch(ori, dir, out hitPos)` 函数。

`RayMarch` 函数的返回值为是否相交，当相交的时候你需要将参数 `hitPos` 设置为交点。参数 `ori` 和 `dir` 为世界坐标系中的值，分别代表光线的起点和方向，其中方向向量为单位向量。

2.3 间接光照

在这一步里，你需要在 `main` 函数中实现间接光照，即利用 SSR 找到的交点计算第二次反射的辐射度。计算间接光照我们使用蒙特卡洛方法求解渲染方程，步骤如下面的伪代码所示，其中 $position_0$ 表示直接光照的着色点， $position_1$ 表示间接光照的着色点。

Algorithm 1 间接光照

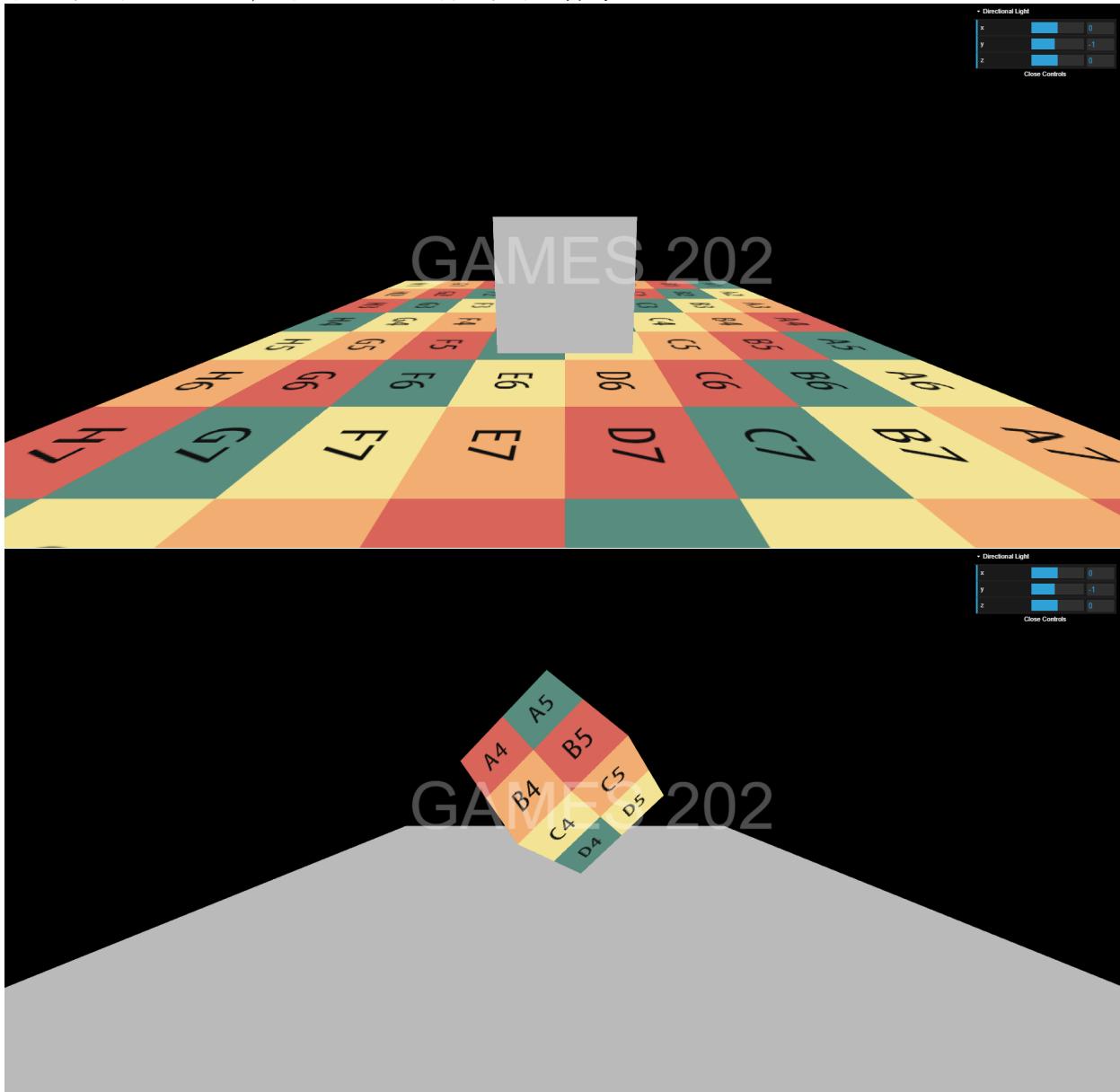
```
1:  $L_{indirect} \leftarrow 0$ 
2: for  $i = 0$  to SAMPLE_NUM do
3:    $direction \leftarrow SampleDirection()$ 
4:    $hit, position_1 \leftarrow Intersection()$ 
5:   if hit then
6:      $L \leftarrow \frac{EvaluateBSDF(position_0)}{PdfBSDF} * EvaluateBSDF(position_1) * EvaluateLight(position_1)$ 
7:      $L_{indirect} \leftarrow L_{indirect} + L$ 
8:   end if
9: end for
10:  $L_{indirect} \leftarrow \frac{L_{indirect}}{\text{SAMPLE\_NUM}}$ 
```

在采样光线方向的时候，你可以均匀采样上半球，或者按 \cos 值加权来采样上半球，我们也提供了相应的 `SampleHemisphereUniform(inout s, out pdf)` 和 `SampleHemisphereCos(inout s, out pdf)`。他们会返回一个局部坐标系的位置，参数 `pdf` 是采样的概率，参数 `s` 是你需要传入的随机数状态。这里我们也提供了 `LocalBasis(n, out b1, out b2)` 函数，通过传入的世界坐标系的法线 `n`，建立局部坐标系，返回两个切线向量。你可以通过他们将采样得到的局部坐标系的方向变换到世界坐标系中。

3 场景

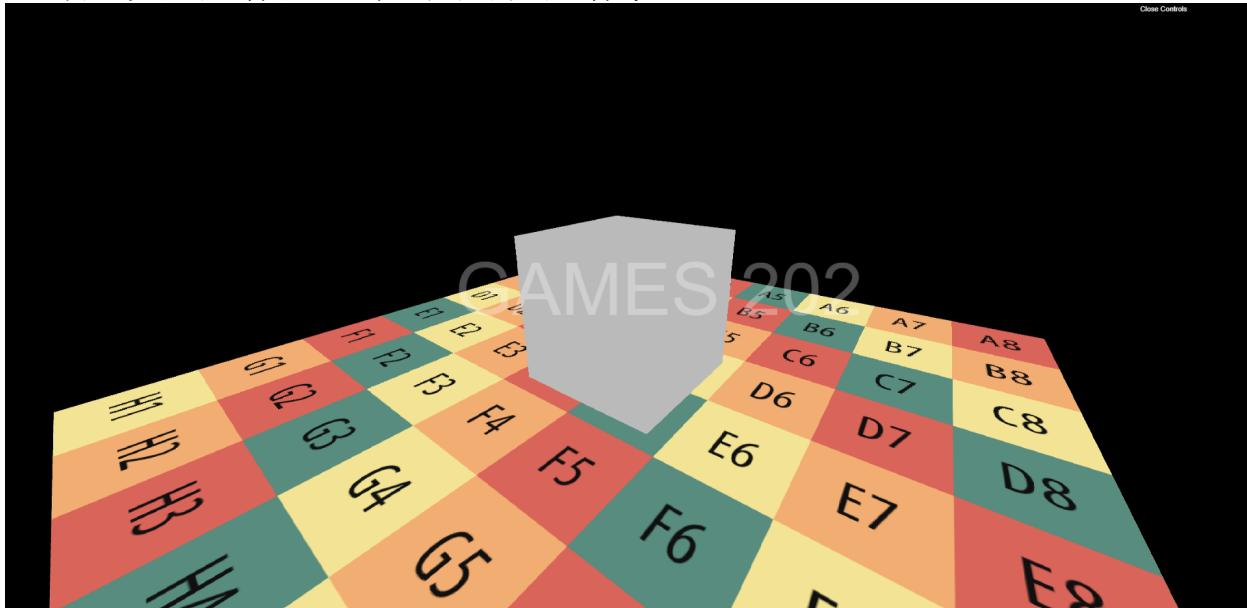
我们提供 3 个场景，两个简单场景方便调试，一个山洞场景用于展示你最终的结果。你可以在 `src/engine.js` 中使用 `loadGLTF()` 函数来加载不同场景。我们也在 `src/engine.js` 中提供了不同场景的光源和相机参数，切换场景时这些参数也要切换。

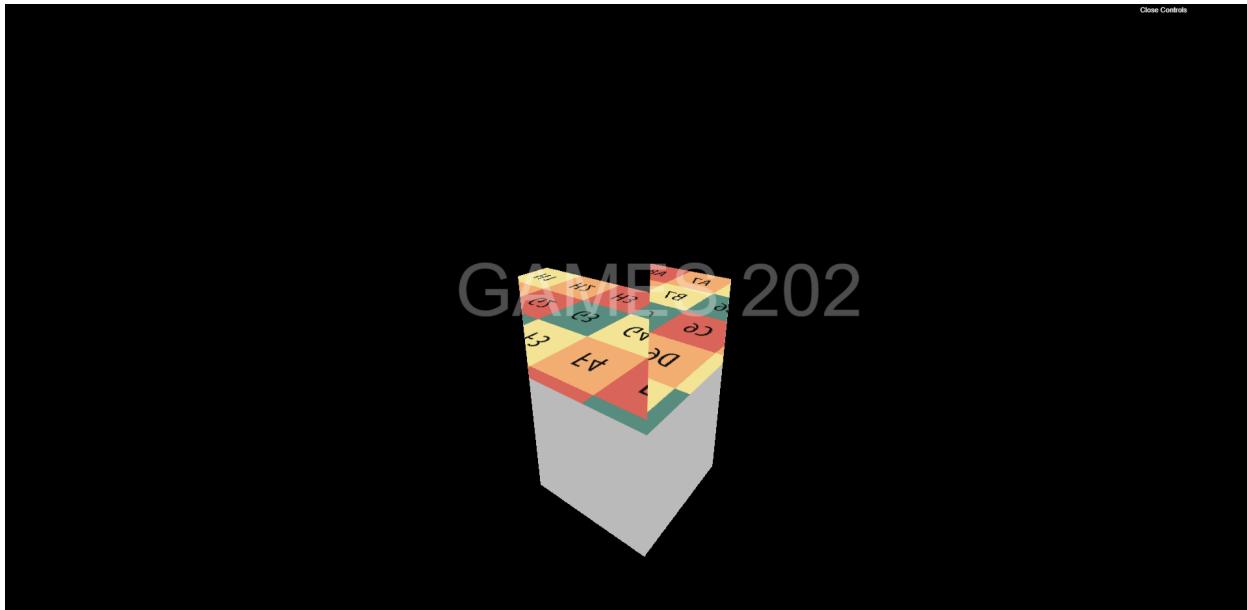
场景如下图所示，下图展示的是漫反射率。





为验证你的 Screen Space Ray Tracing 实现是否正确，你可以在第一个场景中简单实现一个镜面反射来观察结果，如下图所示。第一张图片是漫反射率，第二张图片是镜面反射后查询交点的漫反射率。





最终的场景会接近如下效果，第一张是直接光照，第二张是直接光照和间接光照。







4 评分与提交

提交时需要删除 /lib, /assets 等所有与你实现代码无关的文件夹，并在建立的 /images 文件夹中保存运行截图。

截图需要展示 3 个场景的直接光照，间接光照效果。同时在 README.md 中简要描述本轮工作，尤其是完成了作业的哪些部分。如果完成的任务中包含了提高部分请注明改动了哪些代码、文件以方便助教同学批改。

4.1 评分

- [5 分] 实现直接光照。
- [15 分] 实现 Screen Space Ray Tracing。
- [10 分] 实现间接光照。
- [10 分] Bonus 1: 实现 Mipmap 优化的 Screen Space Ray Tracing。
- [-3 分] 惩罚分数:

未删除 /lib, /assets, assignment3.pdf 等与代码无关的文件夹。

未按格式建立 /images, 缺少结果图片。

4.2 提交

作业提交使用的平台为 Smartchair 平台，地址为<http://www.smartchair.org/GAMES202/>。