Why do we implement lighting in the Pixel Shader?

Asked 9 years, 4 months ago Modified 6 years, 5 months ago

Viewed 1k times

1

I am reading Introduction to 3D Game Programing with DirectX 11 by Frank D. Luna, and can't seem to understand why do we implement lighting in Pixel Shader? I would be grateful if you could send me some reference pages on the subject.

Thank you.

directx hlsl lighting pixel-shader

Share

Improve this question

Follow

asked Aug 1, 2015 at 7:18

mkorunoski

71 ● 2 ● 8

1 Answer

Sorted by: Highest score (default)

9

Lighting can be done many ways. There are hundreds of SIGGRAPH papers on the topic.

For games, there are a few common approaches (or more often, games will employ a mixture of these approaches)

- **Static lighting or lightmaps**: Lighting is computed offline, usually with a global-illumination solver, and the results are baked into textures. These lightmaps are blended with the base diffuse textures at runtime to create the sense of sophisticated shadows and subtle lighting, but none of it actually changes. The great thing about lightmaps is that you can capture very interesting and sophisticated lighting techniques that are very expensive to compute and then 'replay' them very inexpensively. The limitation is that you can't move the lights, although there are techniques for layering a limited number of dynamic lights on-top.
- **Deferred lighting**: In this approach, the scene is rendered many times to encode information into offscreen textures, then additional passes are made to compute the final image. Here often there is one rendering pass per light in the scene. See deferred shading. The good thing about deferred shading is that it is very easy to make the renderer scale with art-driven content without as many hard limits--you can just do more passes for more lights for example which are simply additive. The problem with deferred shading is that each pass tends to do little computation, and the many passes really push hard on the memory bandwidth of modern GPUs which have a lot more compute power than bandwidth.

- **Per-face Forward lighting**: This is commonly known as flat shading. Here the lighting is performed once per triangle/polygon using a face-normal. On modern GPUs, this is usually done on the programmable vertex shader but could also use a geometry shader to compute the per-face normal rather than having to replicate it in vertices. The result is not very realistic, but very cheap to draw since the color is constant per face. This is really only used if you are going for a "Tron look" or some other non-photorealistic rendering technique.
- **Vertex Forward lighting**: This is classic lighting where the light computation is performed per vertex with a per-vertex normal. The colors at each vertex are then interpolated across the face of the triangle/polygon (Gouraud shading). This lighting is cheap, and on modern GPUs would be done in the vertex shader, but the result can be too smooth for many complex materials, and any specular highlights tend to get blurred or missed.
- **Per-pixel Forward lighting**: This is the heart of your question: Here the lighting is computed once per pixel. This can be something like classic Phong or Blinn/Phong shading where the normal is interpolated between the vertices or normal maps where a second texture provides the normal information for the surface. In a modern GPU, this is done in the pixel shader and can provide much more surface information, better specular highlights, roughness, etc. at the expensive of more pixel

shader computation. On modern GPUs, they tend to have a lot of compute power relative to the memory bandwidth, so per-pixel lighting is very affordable compared to the old days. In fact, Physically Based Rendering techniques are quite popular in modern games and these tend to have very long and complex pixel shaders combining data from 6 to 8 textures for every pixel on every surface in the scene.

That's a really rough survey and as I said there's a ton of books, articles, and background on this topic.

The short answer to your question is: *because we can!*

Share Improve this answer Follow

edited Jul 13, 2018 at 5:36

answered Aug 1, 2015 at 8:04

Chuck Walbourn

40.9k ● 2 ● 67 ● 90