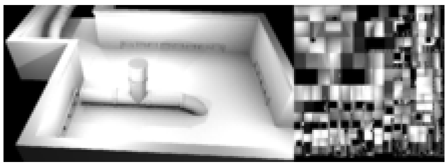
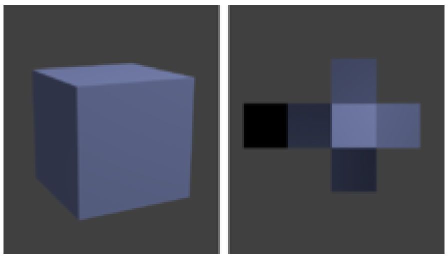
WIKIPEDIA

Lightmap

A **lightmap** is a data structure used in **lightmapping**, a form of surface caching in which the brightness of surfaces in a virtual scene is pre-calculated and stored in texture maps for later use. Lightmaps are most commonly applied to static objects in applications that use real-time 3D computer graphics, such as video games, in order to provide lighting effects such as global illumination at a relatively low computational cost.


A complex scene with corresponding lightmap (shown on the right).


Cube with a simple lightmap (shown on the right).

Contents

History

John Carmack's Quake was the first computer game to use lightmaps to augment rendering.[1] Before lightmaps were invented, realtime applications relied purely on Gouraud shading to interpolate vertex lighting for surfaces. This only allowed low frequency lighting information, and could create clipping artifacts close to the camera without perspective-correct interpolation. Discontinuity meshing was sometimes used especially with radiosity solutions to adaptively improve the resolution of vertex lighting information, however the additional cost in primitive setup for realtime rasterization was generally prohibitive. Quake's software rasterizer used surface caching to apply lighting calculations in texture space once when polygons initially appear within the viewing frustum (effectively creating temporary 'lit' versions of the currently visible textures as the viewer negotiated the scene).

As consumer 3d graphics hardware capable of multitexturing, light-mapping became more popular, and engines began to combine light-maps in real time as a secondary multiply-blend texture layer.

Limitations

Lightmaps are composed of lumels[2] (Lumination elements), analogous to texels in Texture Mapping. Smaller lumels yield a higher resolution lightmap, providing finer lighting detail at the price of reduced performance and increased memory usage. For example, a lightmap scale of 4 lumels per world unit would give a lower quality than a scale of 16 lumels per world unit. Thus, in using the technique, level designers and 3d artists often have to make a compromise between performance and quality; if high resolution lightmaps are used too frequently then the application may consume excessive system resources, negatively affecting performance.

Lightmap resolution and scaling may also be limited by the amount of disk storage space, bandwidth/download time, or texture memory available to the application. Some implementations attempt to pack multiple lightmaps together in a process known as *atlasing*[3] to help circumvent these limitations.

Lightmap resolution and scale are two different things. The resolution is the area, in pixels, available for storing one or more surface's lightmaps. The number of individual surfaces that can fit on a lightmap is determined by the scale. Lower scale values mean higher quality and more space taken on a lightmap. Higher scale values mean lower quality and less space taken. A surface can have a lightmap that has the same area, so a 1:1 ratio, or smaller, so the lightmap is stretched to fit.

Lightmaps in games are usually colored texture maps, or per vertex colors. They are usually flat, without information about the light's direction, whilst some game engines use multiple lightmaps to provide approximate directional information to combine with normal-maps. Lightmaps may also store separate precalculated components of lighting information for semi-dynamic lighting with shaders, such as ambient-occlusion & sunlight shadowing.

Creation

When creating lightmaps, any lighting model may be used, because the lighting is entirely precomputed and real-time performance is not always a necessity. A variety of techniques including ambient occlusion, direct lighting with sampled shadow edges, and full radiosity[4] bounce light solutions are typically used. Modern 3D packages include specific plugins for applying light-map UV-coordinates, atlas-ing multiple surfaces into single texture sheets, and rendering the maps themselves. Alternatively game engine pipelines may include custom lightmap creation tools. An additional consideration is the use of compressed DXT textures which are subject to blocking artifacts – individual surfaces must not collide on 4x4 texel chunks for best results.

In all cases, soft shadows for static geometry are possible if simple occlusion tests (such as basic ray-tracing) are used to determine which lumels are visible to the light. However, the actual softness of the shadows is determined by how the engine interpolates the lumel data across a surface, and can result in a pixelated look if the lumels are too large. See texture filtering.

Lightmaps can also be calculated in real-time[5] for good quality colored lighting effects that are not prone to the defects of Gouraud shading, although shadow creation must still be done using another method such as stencil shadow volumes or shadow mapping, as real-time ray-tracing is still too slow to perform on modern hardware in most 3D engines.

Photon mapping can be used to calculate global illumination for light maps.

Alternatives

Vertex lighting

In **vertex lighting**, lighting information is computed per vertex and stored in vertex color attributes. The two techniques may be combined, e.g. vertex color values stored for high detail meshes, whilst light maps are only used for coarser geometry.

Discontinuity mapping

In **discontinuity mapping**, the scene may be further subdivided and clipped along major changes in light and dark to better define shadows.

See also

- Environment map
- SSAO
- Shader
- Texture mapping
- Baking (computer graphics)

References

1. Abrash, Michael. "Quake's Lighting Model: Surface Caching" (http://www.bluesnews.com/abrash/chap68.shtml). *www.bluesnews.com*. Retrieved 2015-09-07.
2. Channa, Keshav (July 21, 2003). "flipcode - Light Mapping - Theory and Implementation" (http://www.flipcode.com/archives/Light_Mapping_Theory_and_Implementation.shtml). *www.flipcode.com*. Retrieved 2015-09-07.
3. "Texture Atlasing Whitepaper" (http://http.download.nvidia.com/developer/NVTextureSuite/Atlas_Tools/Texture_Atlas_Whitepaper.pdf) (PDF). *nvidia.com*. NVIDIA. 2004-07-07. Retrieved 2015-09-07.
4. Jason Mitchell, Gary McTaggart, Chris Green, Shading in Valve's Source Engine (https://steamcdn-a.akamaihd.net/apps/valve/2006/SIGGRAPH06_Course_ShadingInValvesSourceEngine.pdf). (PDF) Retrieved Jun. 07, 2019.
5. November 16, 2003. Dynamic Lightmaps in OpenGL (http://joshbeam.com/articles/dynamic_lightmaps_in_opengl/). Joshbeam.com Retrieved Jul. 07, 2014.