Local image-based lighting with parallax-corrected cubemaps

Lagarde Sébastien (**Lagardese@hotmail.fr**) & Antoine Zanuttini (**lezanu@gmail.com)**
DONTNOD Entertainment Paris, France

Figure 1. *Left: Render of default cubemap. Middle: Lighting artists Cubemap tools. Right: Render of parallax-corrected cubemap (256x256x6).*

1. Introduction

Image-based lighting (IBL) is typically used for distant lighting represented by an infinite environment map. This technique has been used by many games. Games divide their scenes into several sectors and associate a cubemap (the environment mapping of choice due to its graphic hardware performance) to each of them. The cubemap of the sector where the camera is located is then used to light objects [1]. The problem with this approach is that it cannot accurately represent local reflections on specular and glossy objects.. Our game requires more accurate local reflections, which implies that a local image-based lighting technique must be used. Previous local image-based lighting approaches, such as the Half life 2 approach [2] consist of assigning an individual cubemap to each objects. These approaches suffer from lighting seams and parallax issues [3]. We introduce a new approach which avoids these artifacts while still preserving extremely high performance on current console generation hardware (PS3/XBOX360).

2. Lighting seams

In order to achieve seamless lighting transitions between objects we adopt the single-cubemap approach of the "infinite cubemap" technique; the same cubemap is applied to all objects of the scene. However in our case, this single cubemap is the result of blending several local cubemaps. Contrary to an infinite cubemap, each local cubemap has a position and must be generated in the game engine (in an offline preprocess) to accurately represent the local scene reflection. During the game's runtime, each frame all local cubemap influences which overlap the camera (or the player) are selected to be blended together to create the single cubemap which will be applied to the scene. We develop algorithms for calculating the blending weights for each selected local cubemaps and for blend them appropriately. The blend step is entirely done on the GPU and is very efficient even for current console generation hardware.

Using a single cubemap (generated from local cubemaps) for all visible objects, introduces noticeable artifacts for distant reflective objects. To fight this problem we use the local lighting information available at the distant object locations to adapt the cubemap lighting. We also develop some automatic shader level of detail to smoothly disable cubemap lighting with distance.

3. Parallax issue

A cubemap only defines accurate reflections at the location where it was generated. Furthermore, a cubemap represents an infinite box reflection resulting in parallax issues when used (reflected objects do not appear at the right positions). This is particularly obvious on planar objects. We develop a new technique to fix the parallax issue for planar objects (Figure 1 shows walls with a highly specular planar floor to illustrate the technique). A set of tools have been developed for lighting artists to help them to define the rectangular area that a local cubemap must represent and the provided information is then used at blend time to correctly adjust planar reflections. Coupled with our GPU blending algorithm we are able to blend several parallax-corrected cubemaps efficiently on current console hardware (Around 0.28ms for 4 parallax-corrected 128x128x6 cubemaps on PS3). Figure 2 shows another sample of parallax-corrected cubemap use on a planar object.

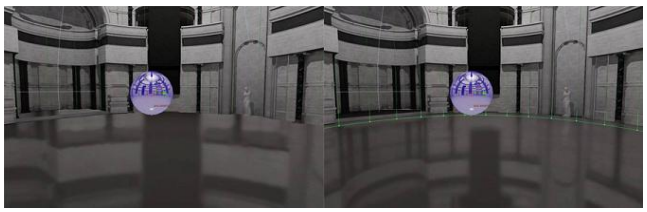

Figure 2. *Left: Cubemap. Right: Parallax-corrected cubemap.*

4. Conclusions

The advantage of our system is to provide an efficient way to simulate local reflections with seamless transitions between objects. It requires some tools to ease the work of lighting artists as well as some setup time.

References

[1] Van der leeuw, The playstation 3 SPUs in the real world. http://www.slideshare.net/guerrillagames/the-playstation3s-spus-in-the-real-world-a-killzone-2-case-study-9886224
[2] McTaggarts, Half-Life 2 Valve Source Shading http://www2.ati.com/developer/gdc/D3DTutorial10_Half-Life2_Shading.pdf
[3]Valve wiki http://developer.valvesoftware.com/wiki/Cubemaps

SIGGRAPH 2012, Los Angeles, California, August 5 – 9, 2012.
ISBN 978-1-4503-1435-0/12/0008