Screen Space Classification for Efficient Deferred Shading

Neil Hutchinson, Balor Knight, Matthew Ritchie, George Parrish, Jeremy Moore

Disney Interactive Studios

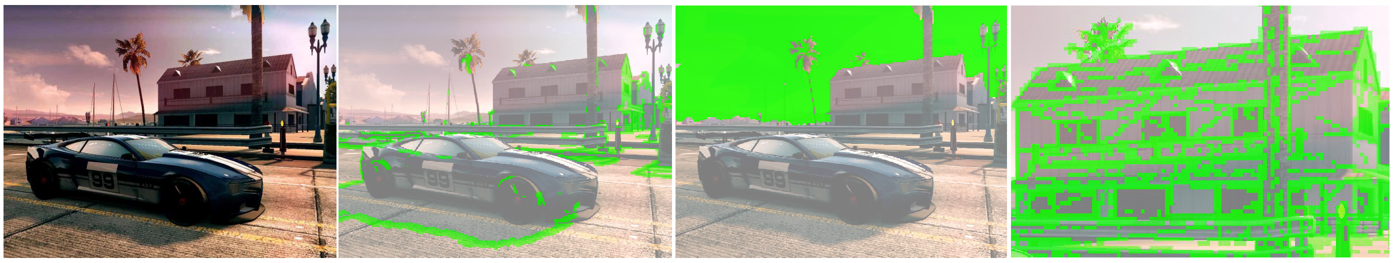

Figure 1: (a) Final rendered image, (b) Soft shadow classification, (c) Sky classification, (d) MSAA edge classification zoomed to show tile structure.

Introduction

Deferred shading is an increasingly popular technique for video game rendering. In the standard implementation a geometry pass writes depths, normals and other properties to a geometry buffer (G-buffer) before a lighting pass is applied as a screen space operation. Deferred shading is often combined with deferred shadowing where the occlusion values due to one or more lights are gathered in a screen space shadow buffer. The universal application of complex shaders to the entire screen during the shadow and light passes of these techniques can contribute to poor performance. A more optimal approach would take different shading paths for different parts of the scene. For example we would prefer to only apply expensive shadow filtering to known shadow edges. However this typically involves the use of dynamic branches within shaders which can lead to poor performance on current video game shading hardware.

[Swoboda 2009] proposes using the Synergistic Processing Unit (SPU) of Sony's PlayStation® 3 to categorize areas of the screen with the aim of increasing rendering efficiency in post processing effects such as depth of field. [Moore, Jefferies 2009] propose using screen space buffers to classify the screen as a way to optimize MSAA deferred shading and deferred shadow rendering. We extend these works to provide a general framework for screen classification. We introduce a method for decomposing the screen into tiles and define a number of useful tile classification criteria. We show how this approach can be used to reduce the complexity of the lighting pass in deferred shading and to optimize a further range of rendering and post processing effects.

We describe how our approach has been implemented in *Split/Second*, a major racing game developed at Disney's *Black Rock Studio* We provide details of the implementation for the Microsoft Xbox® 360 and Sony PlayStation® 3 hardware platforms.

Our Approach

In our approach we divide the screen into tiles. For each tile we aim to apply lighting shaders that contain the minimum locally required complexity. To achieve this we first apply a full screen classification pass which determines which components of our lighting calculation are required in each screen tile. This classification pass can be carried out using either the GPU or the CPU or a combination of the two. We use a palette of screen attributes from which the classification pass selects a combination for each tile. In our example case the attributes include but are not limited to: "sky", "MSAA edge", "sun-facing", and "soft shadow edge". The presence of these attributes in any tile can be determined by analyzing the G-buffer and the screen space deferred shadow mask.

For each tile the collected screen attributes are combined to create a shader ID. This determines the shader that will be used to process the tile during the lighting pass. The maximum number of shaders used in the lighting pass is 2^n where n is the total number of screen attributes. For large n combinatorial complexity can be avoided by grouping our screen attributes and defining more than one pass per tile. In our example we use 8 screen attributes so that the shader ID fits conveniently into an 8-bit render target channel. The shaders themselves can be generated by pre-processing an uber-shader during shader compilation.

Once the shader ID for each tile has been calculated we generate an index buffer with which to submit a single draw call for each shader found in our scene. In our implementation this index buffer generation phase is carried out on the CPU and synchronized with the GPU submission and rendering.

The optimum tile size to trade off average tile complexity against vertex cost can vary according to hardware and scene complexity. For *Split/Second* we found that a tile size of 4x4 pixels gave a good balance. Vertex processing cost is further reduced by a simple aggregation of adjacent tiles with the same shader ID.

We have adapted the technique for our different target hardware platforms. On the PlayStation® 3 we move work away from the GPU by carrying out part of the classification pass on the SPU. The SPU is also used to generate the final index buffers. On the Xbox® 360 we make use of the Procedural Synthesis feature (XPS) to optimally synchronize the CPU submission of draw calls with the GPU.

References

SWOBODA, M. 2009. Deferred Lighting and Post Processing on PlayStation®3. *Game Developers Conference 2009.*

MOORE, J., JEFFERIES, D. 2009. Rendering Techniques in Split/Second. *Advanced Real-Time Rendering in 3D Graphics and Games – SIGGRAPH 2009.*

Copyright is held by the author / owner(s).
SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007