Search

Developer Site Homepage

Developer News Homepage

Developer Login

Become a Registered Developer

Developer Tools

Documentation

DirectX

OpenGL

GPU Computing

Handheld

Events Calendar

Newsletter Sign-Up

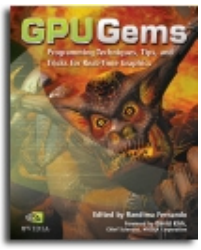
Drivers

Jobs (1)

Contact

Legal Information

Site Feedback



GPU Gems

GPU Gems is now available, right here, online. You can purchase a beautifully printed version of this book, and others in the series, at a 30% discount courtesy of InformIT and Addison-Wesley.

Please visit our Recent Documents page to see all the latest whitepapers and conference presentations that can help you with your projects.

Chapter 11. Shadow Map Antialiasing

Michael Bunnell
NVIDIA

Fabio Pellacini
Pixar Animation Studios

11.1 Introduction

Shadow mapping is the method of choice for creating shadows in high-end rendering for motion pictures and television. However, it has been problematic to use shadow mapping in real-time applications, such as video games, because of aliasing problems in the form of magnified *jaggies*. This chapter shows how to significantly reduce shadow map aliasing in a shader. It describes how to implement a simplified version of percentage-closer filtering that makes the most out of the GPU's shadow-mapping hardware to render soft-edged, antialiased shadows at real-time rates.

Shadow mapping involves projecting a shadow map on geometry and comparing the shadow map values with the light-view depth at each pixel. If the projection causes the shadow map to be magnified, aliasing in the form of large, unsightly jaggies will appear at shadow borders. Aliasing can usually be reduced by using higher-resolution shadow maps and increasing the shadow map resolution, using techniques such as *perspective shadow maps* (Stamminger and Drettakis 2002). However, using perspective shadow-mapping techniques and increasing shadow map resolution does not work when the light is traveling nearly parallel to the shadowed surface, because the magnification approaches infinity.

High-end rendering software solves the aliasing problem by using a technique called *percentage-closer filtering*.

11.2 Percentage-Closer Filtering

Unlike normal textures, shadow map textures cannot be prefiltered to remove aliasing. Instead, multiple shadow map

comparisons are made per pixel and averaged together. This technique is called percentage-closer filtering (PCF) because it calculates the percentage of the surface that is closer to the light and, therefore, not in shadow.

The original PCF algorithm, described in Reeves et al. 1987, called for mapping *the region to be shaded* into shadow map space and sampling that region stochastically (that is, randomly). The algorithm was first implemented using the REYES rendering engine, so *the region to be shaded* meant a four-sided micropolygon. Figure 11-1 shows an example of that implementation.

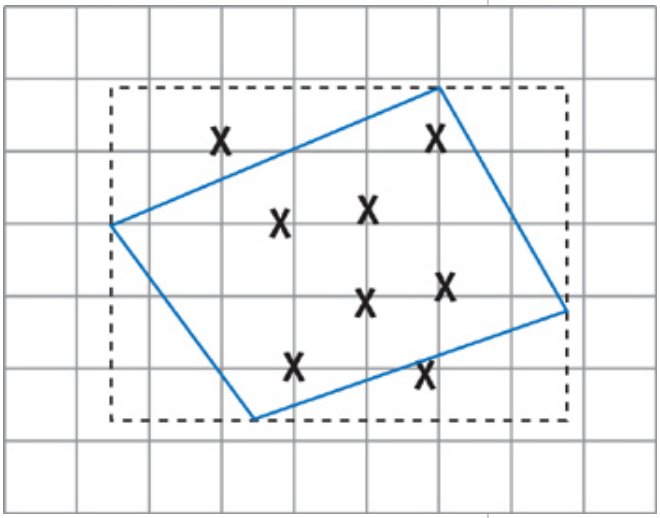


Figure 11-1 Percentage-Closer Filtering

In our implementation, we have changed the PCF algorithm slightly to make it easy and efficient to apply. Instead of calculating the region to be shaded in shadow map space, we simply use a 4x4-texel sample region everywhere. This region is large enough to significantly reduce aliasing, but not so large as to require huge numbers of samples or stochastic sampling techniques to achieve good results. Note that the sampling region is not aligned to texel boundaries. An aligned region would not achieve the antialiasing effect that we want.

Hardware shaders work on pixels, not on micropolygons, so matching the original implementation would involve transforming a four-sided polygon representing a screen pixel into shadow map space to calculate the sample region. Our implementation uses a fixed-size sample region instead. A fixed-size region lets us skip a complicated transformation and allows us to calculate a precise shadow percentage instead of an approximate one using stochastic sampling. See Figure 11-2.

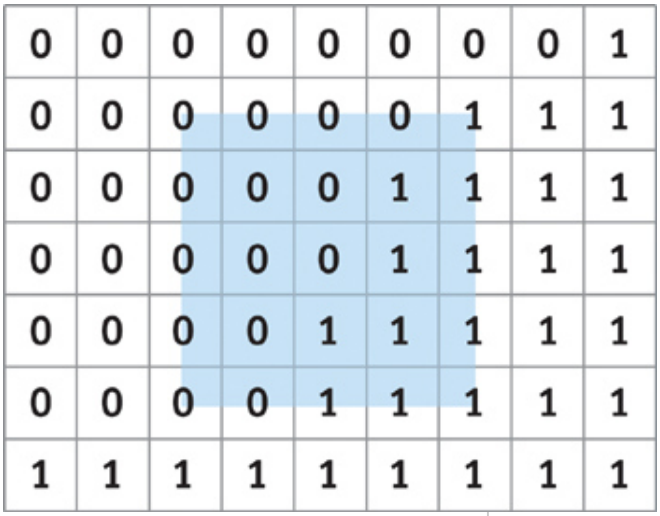


Figure 11-2 Sampling an Area of 4x4 Texels

11.3 A Brute-Force Implementation

NVIDIA GPUs have built-in percentage-closer filtering for shadow map sampling. The hardware does four depth compares and uses the fractional part of the texture coordinate to bilinearly interpolate the shadow value. The shadow result is the percentage that a texel-size sample area is in shadow. See Figure 11-3. A single texel-size sample region is not big enough to effectively remove aliasing, but the region can be increased to a 4x4 texel size by averaging 16 shadow compare values. The offsets for *x* and *y* are -1.5, -0.5, 0.5, and 1.5 for samples one texel unit apart.

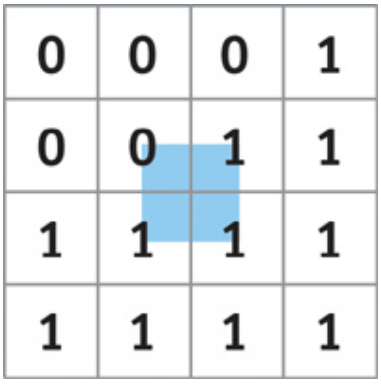


Figure 11-3 Using the Hardware

The following function can be used to do a projected texture map read with an offset given in texel units. The variable `texmapscale` is a `float2` containing 1/width and 1/height of the shadow map.

```
float3 offset_lookup(sampler2D map,
                     float4 loc,
                     float2 offset)
{
return tex2Dproj(map, float4(loc.xy + offset * texmapscale * loc.w,
                             loc.z, loc.w));
}
```

We can implement the 16-sample version in a fragment program as follows:

```
float sum = 0;
float x, y;

for (y = -1.5; y <= 1.5; y += 1.0)
  for (x = -1.5; x <= 1.5; x += 1.0)
    sum += offset_lookup(shadowmap, shadowCoord, float2(x, y));

shadowCoeff = sum / 16.0;
```

11.4 Using Fewer Samples

The performance of the brute-force method is better than one might expect. Many of the texture fetches are in the texture cache because they are guaranteed to be close to one another. However, if we change the sampling pattern per pixel, we can attain similar results with only four samples per pixel.

The four-sample technique produces results similar to those created by dithering black-and-white data to render a grayscale image. The sample region size remains the same, but we use only four of the 16 samples per pixel. The set of four samples varies depending on the screen location. Figure 11-4 shows the sampling pattern used in the four-sample version of the shader to pick four out of 16 possible sample locations per pixel.

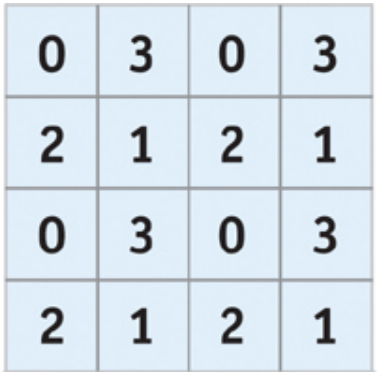


Figure 11-4 The Sampling Pattern Used for the Four-Sample Version of the Shader

We can implement the four-sample version as follows:

```
offset = (float)(frac(position.xy * 0.5) > 0.25);  // mod
offset.y += offset.x;  // y ^= x in floating point

   if (offset.y > 1.1)
  offset.y = 0;
shadowCoeff = (offset_lookup(shadowmap, sCoord, offset +
                             float2(-1.5, 0.5)) +
               offset_lookup(shadowmap, sCoord, offset +
                             float2(0.5, 0.5)) +
               offset_lookup(shadowmap, sCoord, offset +
                             float2(-1.5, -1.5)) +
               offset_lookup(shadowmap, sCoord, offset +
                             float2(0.5, -1.5)) ) * 0.25;
```

11.5 Why It Works

How can we antialias shadows with a fixed-size sample region even though texture projection can greatly magnify the shadow map? The answer is simple: When a texture is magnified, the texture map samples are close to each other for adjacent pixels. If the samples are close to each other and the sampled area is relatively large, then there can be only a very small difference between the shadow values, because the sample areas overlap a lot. Figure 11-5 shows how sample areas for adjacent pixels overlap when the shadow map is magnified.

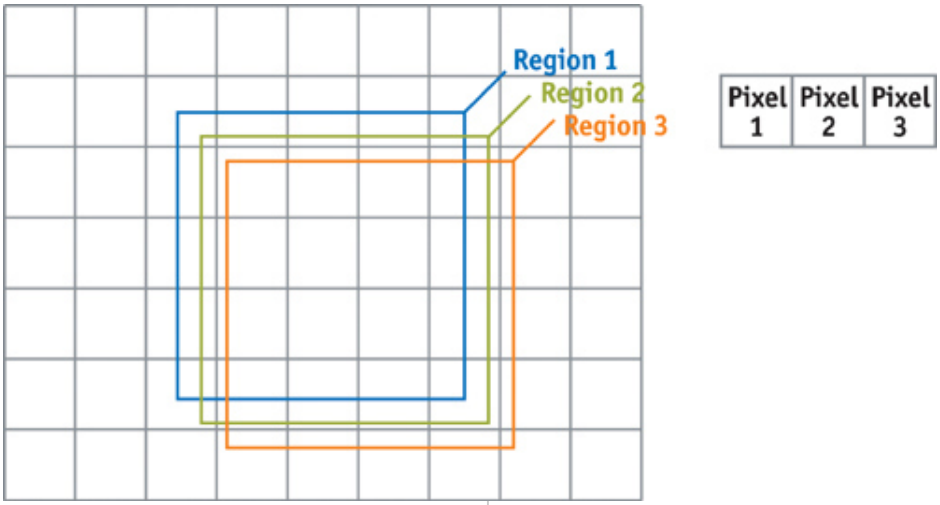


Figure 11-5 Overlapping Sampling Regions for Adjacent Pixels

The more the shadow map is magnified, the smaller the difference between adjacent pixels, and the smoother the transition between shadowed and unshadowed regions. The hardware calculates that shadow percentage with eight bits of precision, so even in the case of extreme magnification and high-contrast shadows, there will always be a smooth shadow transition without banding. If the shadow regions are very close to each other, the shadow value will differ only by the least significant bit for eight-bits-per-component output. This is illustrated in Figures 11-6, 11-7, and 11-8, which show shadows for a ninja model with 1, 4, and 16 samples, respectively. Figure 11-9 shows a magnification of the ninja's thumb shadow in each of the three cases. Notice the vastly improved shadow quality in the 16-sample case.

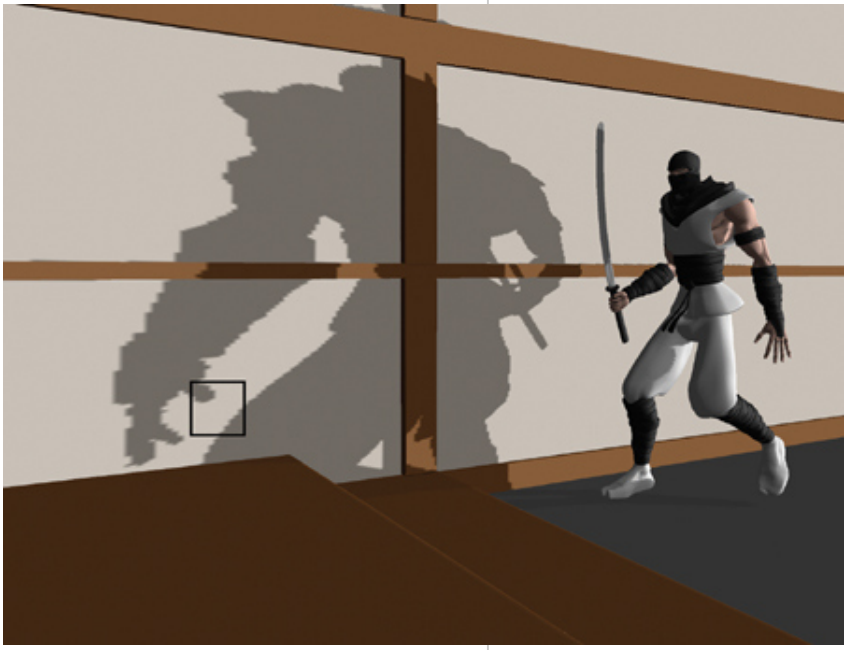


Figure 11-6 Ninja Shadow with One Sample per Pixel

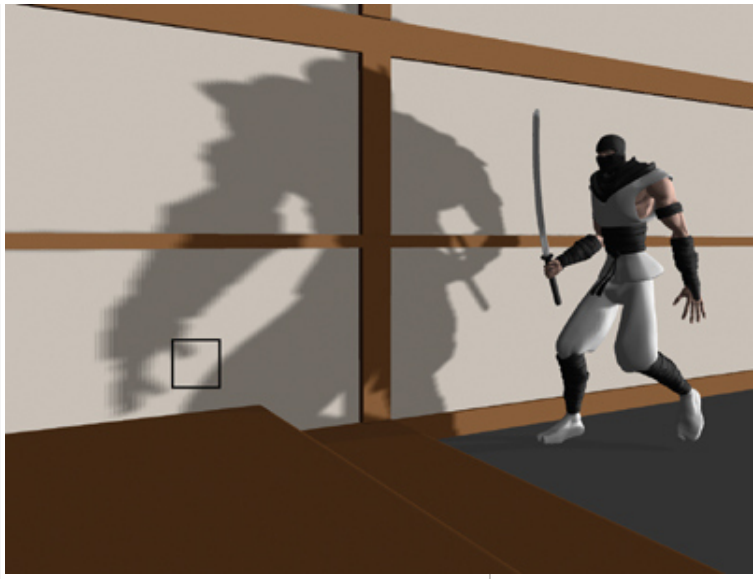


Figure 11-7 Ninja Shadow with Four Dithered Samples per Pixel

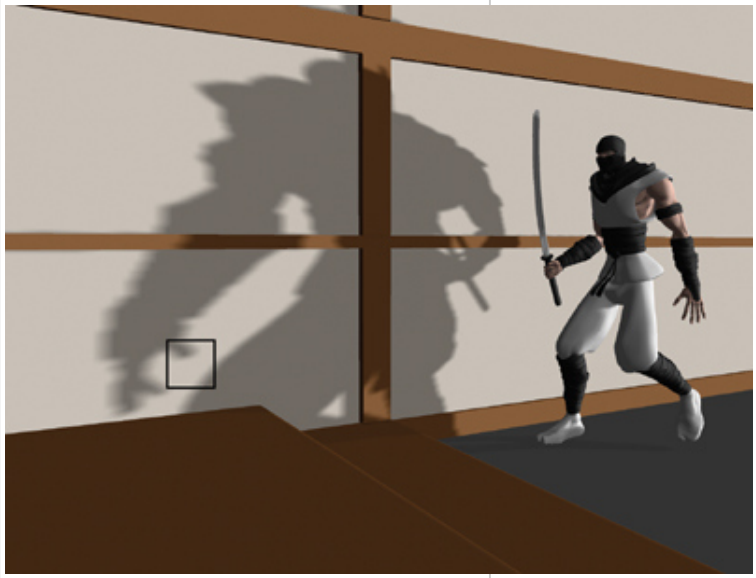


Figure 11-8 Ninja Shadow with Sixteen Samples per Pixel

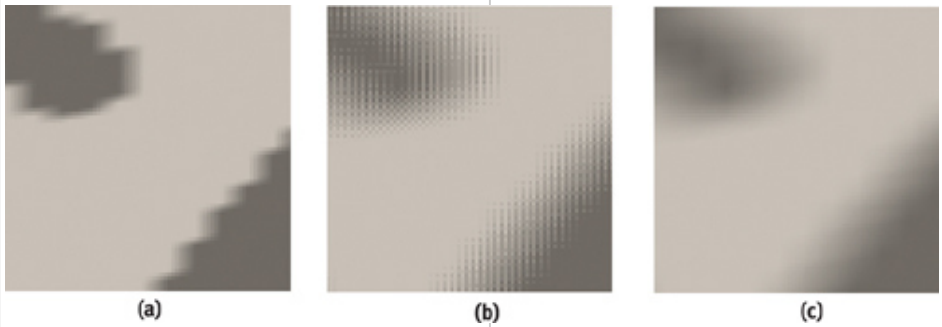


Figure 11-9 The Shadows Magnified

11.6 Conclusion

Shadow mapping is a popular method for rendering shadows, but it suffers from aliasing artifacts. We can greatly reduce shadow map aliasing by averaging multiple shadow map values. If we take advantage of the GPU's shadow-mapping hardware and use clever sampling techniques, we can render soft-edged, antialiased shadows at high frame rates.

11.7 References

Fernando, Randima, and Mark Kilgard. 2003. *The Cg Tutorial*. Addison-Wesley. This introduction to the Cg language has a good section on shadow mapping.

Reeves, W. T., D. H. Salesin, and P. L. Cook. 1987. "Rendering Antialiased Shadows with Depth Maps." *Computer Graphics* 21(4) (Proceedings of SIGGRAPH 87).

Stamminger, Marc, and George Drettakis. 2002. "Perspective Shadow Maps." In *Proceedings of SIGGRAPH 2002*, pp. 557–562.

Copyright

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Addison-Wesley was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers discounts on this book when ordered in quantity for bulk purchases and special sales. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact:

International Sales
international@pearsoned.com

Visit Addison-Wesley on the Web:
www.awprofessional.com

Library of Congress Control Number:
2004100582

GeForce™ and NVIDIA Quadro® are trademarks or registered trademarks of NVIDIA Corporation.
RenderMan® is a registered trademark of Pixar Animation Studios.
"Shadow Map Antialiasing" © 2003 NVIDIA Corporation and Pixar Animation Studios.
"Cinematic Lighting" © 2003 Pixar Animation Studios.
Dawn images © 2002 NVIDIA Corporation. Vulcan images © 2003 NVIDIA Corporation.

Copyright © 2004 by NVIDIA Corporation.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher. Printed in the United States of America. Published simultaneously in Canada.

For information on obtaining permission for use of material from this work, please submit a written request to:

Pearson Education, Inc.
Rights and Contracts Department
One Lake Street
Upper Saddle River, NJ 07458

Text printed on recycled and acid-free paper.

5 6 7 8 9 10 QWT 09 08 07

5th Printing September 2007