* Abstract
* The Abstract forms a review of your work
* It is a condensed version of the main body of your work
* Highlighting the major points covered
* Briefly describing the content and scope
* It is written after the report has been completed

Introduction

\*state the research question

\*history (current situation)

\*methods

\*where is used

Overview, then

Methods

Describe study design

Allow the reader to judge the validity of results in the context of methods used

Results

Present data, tables and figures that support results

Discussion

Answer questions posed in introduction

Support answer

Explain how answer/findings fits in the industry

Note surprising figures and inconsistencies

Notes:

illumination methods:

* raytracing and/or raycasting (raycasting is what is referred to as only "the first hit" in the process of raytracing)
* radiosity
  + The one type of radiosity form factor computation method that probably gets mentioned most is hemicubes

Real notes:

A lot of GI is done offline?

Approximating Dynamic Global Illumination in Image Space (first provided paper):

1.

Ambient occlusion is a popular illumination approximation.

There are many efficient methods for AO [reference some] and some of them are performed in screen space.

This paper shows that SSAO methods can facilitate data for other types of effects like directional shadows and indirect colour bleeding.

Does not depend on scene complexity

AO decouples visibility and illumination, allowing only for approximations.

Traditional SSAO ignores the directionality of light

2.

Ambient occlusion (AO) origins:

COOK, R. L., AND TORRANCE, K. E. 1981. A reﬂectance model for computer graphics. In SIGGRAPH ’81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, 307–316

ZHUKOV, S., IONES, A., AND KRONIN, G. 1998. An Ambient Light Illumination Model. In Eurographics Rendering Workshop 1998, 45–56

AO use in production:

LANDIS, H. 2002. RenderMan in Production. In ACM SIGGRAPH 2002 Course 16

AO is preferred because of its speed, simplicity and ease of implementation

AO Fields introduction

KONTKANEN, J., AND LAINE, S. 2005. Ambient Occlusion Fields. In Proceedings of ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games, 41–48

3.

Proposed method computes light transport in image space using a framebuffer with positions and normal:

SEGOVIA, B., IEHL, J.-C., MITANCHEY, R., AND P´EROCHE, B. 2006. Non-interleaved Deferred Shading of Interleaved Sample Patterns. In SIGGRAPH/Eurographics Graphics Hardware.

This outputs a framebuffer with illuminated pixels using two rendering passes: One for direct light and another one for indirect bounces.

Direct Lighting using DO:

For every pixel at 3D position P with normal n, the direct radiance Ldir is computed from N sampling directions ωi, uniformly distributed over the hemisphere, each covering a solid angle of ∆ω = 2π/N:

Each sample computes the product of incoming radiance Lin, visibility V and the diffuse BRDF ρ/π

To avoid the use of ray-tracing to compute the visibility V , we approximate occluders in screen space instead

In our approximate visibility test, all the sampling points below the surface of the nearby geometry are treated as occluders

In contrast to SSAO, we do not compute the illumination from all samples, but only from the visible directions

Including this directional information can improve the result by display coloured shadows, whereas SSAO simply displays a grey shadow at each location

Indirect Bounces:

The direct light stored in the previous pass framebuffer can be used by sampling occluders’ light bounce.

Introduction to Depth peeling

EVERITT, C., 2001. Introduction Interactive Order-Independent Transparency. NVidia Technical Report.

Depth peeling or different camera positions can be used to remove many problems with SSDO and give better sample set of the environment.

Emphasis on Screen Space technique and constant complexity

Deferred Radiance transfer volumes (Far Cry 3):

Developers thought that fully dynamic solutions were not a good trade-off between quality and performance

The technique uses precomputed probes that allow to relight the scene and approximate global illumination in real-time.

Much of the work if offloaded to the CPU

The deferred radiance transfer volumes contribute the ambient lighting in the image

The static objects contribute to the global illumination while dynamic objects receive global illumination but do not contribute to it.

Offline, we place probes in the world and precompute radiance transfer for them, a process which is also known as baking.

In-game, we relight the probes in real-time, whenever the lighting environment changes. We generate new irradiance values on the fly, which we insert into a number of volume textures. The GPU then uses these to shade everything in screen-space.

we automatically spawn new probes at good locations and make adjustments afterwards

To do so, we cast vertical rays every 4 meters along the ground plane. Every time the ray hits a static object or the terrain, we spawn a new probe. We also make sure to keep a minimal distance between hits

Since raycasting against the individual tree leaves will give very noisy result, we use instead the combined bounding box of the vegetation geometry. In order to not going over budget by spawning too many probes on top of the vegetation, we space the rays every 8 metres instead of 4.

We use a custom multicore raytracing solution for baking

On consoles, the probes store two things - low frequency precomputed radiance transfer (PRT for short) and sky visibility.

PRT allows us to capture the interaction of the surfaces in the scene, without knowing the lighting environment in advance.

more details about PRT, the SIGGRAPH 2002 work by  Sloan  et  al.  is  a  good  starting  point

Probes require complicated treatment, because they  are  really  “empty  space”  – that is, we have no idea of what the shading normal is going to be when baking.

Our solution is to compute PRT for a few directions on the phere. These directions are called  the  “transfer  basis”.  In  our   case, we have four of them - the first three point upwards, and the last direction points straight down.

We also store the sky visibility for the probe as a function on the sphere. You can think of this as a sort of very low frequency sky mask – the function has values of 1 for the directions where the sky is visible, and 0 otherwise. We use a single scalar for each of the 4 transfer basis directions.

For PC, we also bake local radiance transfer. Using that, we can approximate global illumination from dynamic light sources

The implementation is based on the SIGGRAPH 2005 paper by Kristensen et al.

In FC3, the lighting from the sky and the sun is driven by a couple of artist-authored gradients.

Once  we’ve  relit  the  probes,  we  insert  the  resulting  irradiance   values into volume maps which follow the camera. The approach was inspired by CryEngine’s light propagation volumes

There’s  two  instances  where  we  need  to  update  the  volume   textures. First is when the time of day has changed sufficiently and  we’ve  relit  the  probes.  The  second  is  much  more  frequent,   when the camera position has changed, and the volume map has moved more than one texel in world space.

In the second case, rather than flushing the entire volume map, our strategy is re-use the data from previous frames. Only  slices  that  don’t  have  existing  data  are  copied  into  the   volume textures. In order to avoid shifting the other slices to their corresponding location in the volume, instead we use wrapping.

This illustration shows how we do that – when the camera moves in X, we update the correct Y slice, when it moves in Y we update the correct X slice and so on. Here we also keep track of the planes that were updated, so we can know at what offset the volume textures should wrap around.

In our game we have a few materials that are not lit deferred – things like transparent objects, particles. In this case, we compute the closest probe on the CPU and upload it as pixel shader constant, then do the ambient in forward

Far Cry 3 has lots of indoors locations too. The indoors and outdoors probe have drastically different intensities, which leads to the so-called probe bleeding. In this case we see that the interior is incorrectly lit by the outdoors probes, and it picks up the colour of the sky which makes it too blue.

To prevent this, we have a solution based on closed 3D volumes that our artists can place inside the problem buildings

Reference uses:

References to parts/aspects of the topic

References to hardware requirements, market share/penetration; references to research on those features

Cyril Crassin, Neyret Fabrice, Sainz Miguel, Green Simon, and Eisemann Elmar, "Interactive indirect illumination using voxel cone tracing," Computer Graphics Forum, vol. 30, pp. 921-1930, September 2011.

Gabriel Telles O'Neill, Lee Won-Sook, and William Jeff, Haptic-Based 3D Carving Simulator

Dennis Bautembach, Animated sparse voxel octrees, 2011, Phd Thesis.

Aaron Lefohn, Shubhabrata Sengupta, Joe Kniss, Robert Strzodka, and John D Owens, "Dynamic adaptive shadow maps on graphics hardware," ACM SIGGRAPH, p. 13, 2005

Technique first introductions/ first papers

Jon Olick, "Current and Next Generation Parallelism in Games," in SIGGRAPH, 2008

Cyril Crassin, Fabrice Neyret, Sylvain Lefebvre, and Elmar Eisemann, "GigaVoxels : Ray-Guided Streaming for Efficient and Detailed Voxel Rendering," in ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 2009, p. Febuary

Samuli Laine and Tero Karras, "Efficient sparse voxel octrees," Visualization and Computer Graphics, vol. 11, no. 8, pp. 1048-1059, 2011

Aaron M Knoll, Ingo Wald, and Charles D. Hansen, "Coherent multiresolution isosurface ray tracing," The Visual Computer, vol. 25, no. 3, pp. 209-225, 2009

Audun Wilhelmsen, Efficient Ray Tracing of Sparse Voxel Octrees on an FPGA, 2012, Phd Thesis.

Ray tracing was first proposed back in 1968 by Arthur Appel:

Arthur Appel, "Some techniques for shading machine renderings of solids," in Proceedings of the April 30--May 2, 1968, spring joint computer conference, 1968, pp. 37-45

it revoles around the concept of shooting rays from the camera into the world/volumetric data and returning the colour of the voxel hit by the ray. This however was quickly expanded in 1977 by Turner Whittard [20] who presented the idea of ray tracing. Ray tracing is very similar to ray casting, except whenever the casted rays hit a voxel instead of returning that colour value they instead shoot off multiple additional rays to provide additional information such as reflection, refraction and shadowing

[20] Turner Whitted, "An improved illumination model for shaded display," ACM SIGGRAPH Computer Graphics, vol. 13, no. 2, 1979

Offline ray tracing use:

Steven G Parker et al., "Optix: a general purpose ray tracing engine," ACM Transactions on Graphics (TOG), vol. 29, no. 4, p. 66, 2010

Cone Tracing (high quality offline renders):

David B Kirk, "The simulation of natural features using cone tracing," The Visual Computer, vol. 3, no. 2, pp. 6371, 1987

Vertex based transformation was first proposed by William E Lorensen in 1987 [30]

William E. Lorensen and E. Cline Harvey, "Marching cubes: A high resolution 3D surface construction algorithm," ACM Siggraph Computer Graphics, vol. 21, no. 4, pp. 163-169, 1987

Addressed issues in other papers

Abstract:

* Historical grounding on changes that have been happening
* Dominating methodology
* Present topic
* Present aimed work

Introduction

* Historical layout
* Introducing background knowledge or area (selected and opposing views) / introducing topic
* Report aims

More detailed description/going into the topics

Compare to other methods

Discuss Methods

Important issues/limitations to discuss

List historical origin of techniques in discussions!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

(in the voxel paper)

There are well established off-line solutions for accu-

rate global-illumination computation such as path trac-

ing [Kaj86],