MSc Computer Games and Entertainment

Maths & Graphics Unit 2011/12

Lecturer: Gareth Edwards

Order Independant Transparency

Order-Independent Transparency

First: Confusion

- **The terms A-Buffer and Accumulation-Buffer are used interchangably.**
- **But this is wrong – typically the A-Buffer refers ONLY to the implementation of the Accumulation-Buffer in REYES.**
- **The Accumulation-Buffers is used typically to composite multiple layers or images in a single result – NO DEPTH.....**
- **The A-Buffer term originated from REYES, in which Micro-Polygons, Bit masks and the Accumulation-Buffer are combined into a single order-independent rendering system.**

Accumulation-Buffer

- Hidden Surface Method
- Provides for Order-Independent Transparency
- Transparent objects that require alpha blending cannot be rendered on top in a Z-Buffer.
- Blending two or more normal's, depth or position values leads to wrong results.
- In other words deferred lighting of objects that need to be visible through each other is not easily possible because the data for the object that is visible through another object is lost

The A-buffer, an Antialiased Hidden Surface Method (1984)

- Loren Carpenter

- The A-buffer (anti-aliased, area-averaged, accumulation buffer) is a general hidden surface mechanism suited to medium scale virtual memory computers.

- It resolves visibility among an arbitrary collection of opaque, transparent, and intersecting objects.

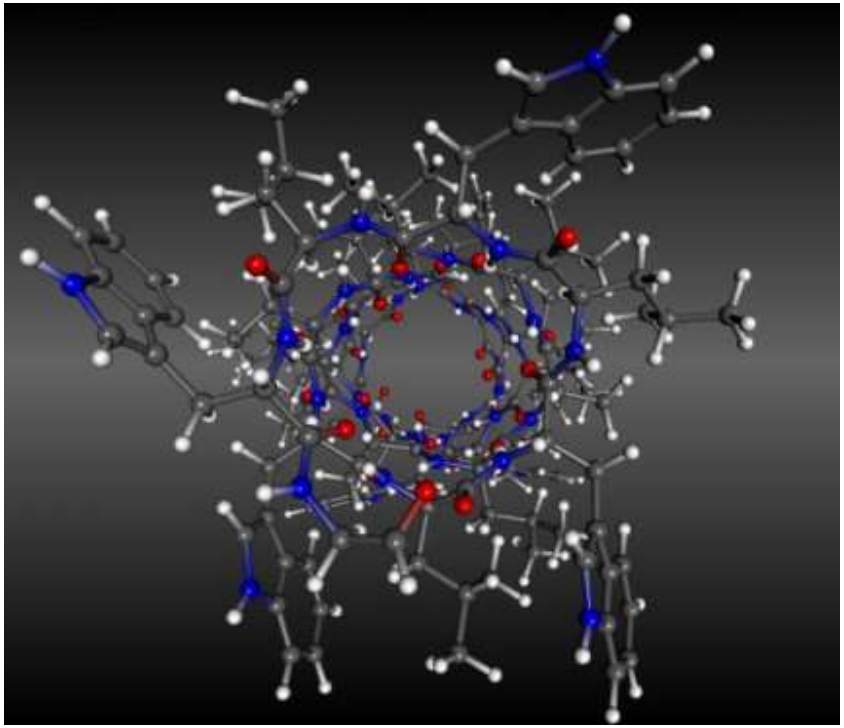
- Using an easy to compute Fourier window (box flter), it increases the effective image resolution many times over the Z-buffer, with a moderate increase in cost.

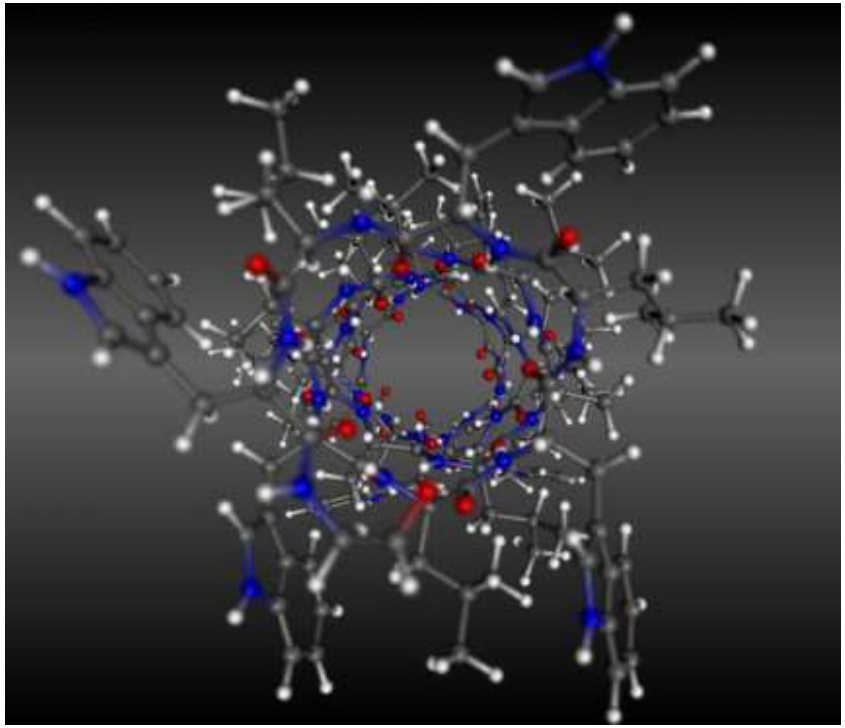
- The A-buffer is incorporated into the REYES 3-D rendering system at Lucasfilm and was used successfully in the "Genesis Demo " sequence in Star Trek 2.

Accumulation-Buffer

- The accumulation buffer can be used to accumulate a series of images generated in the colour buffer. Finally, the result is copied back into the colour buffer for viewing. Because a scene has to be rendered several times it takes longer to create an image, but the quality of the output is improved significantly.

- Depth of Field.....



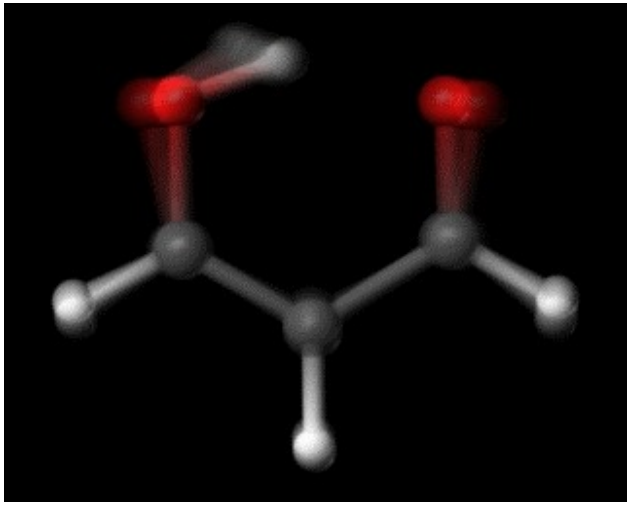


Accumulation-Buffer

- The scene is rendered continuously. Between each rendering step the positions of the atoms are changed.

- After the image of a scene is added to the accumulation buffer the contents of the accumulation buffer is normalized and copied to the colour buffer for viewing.

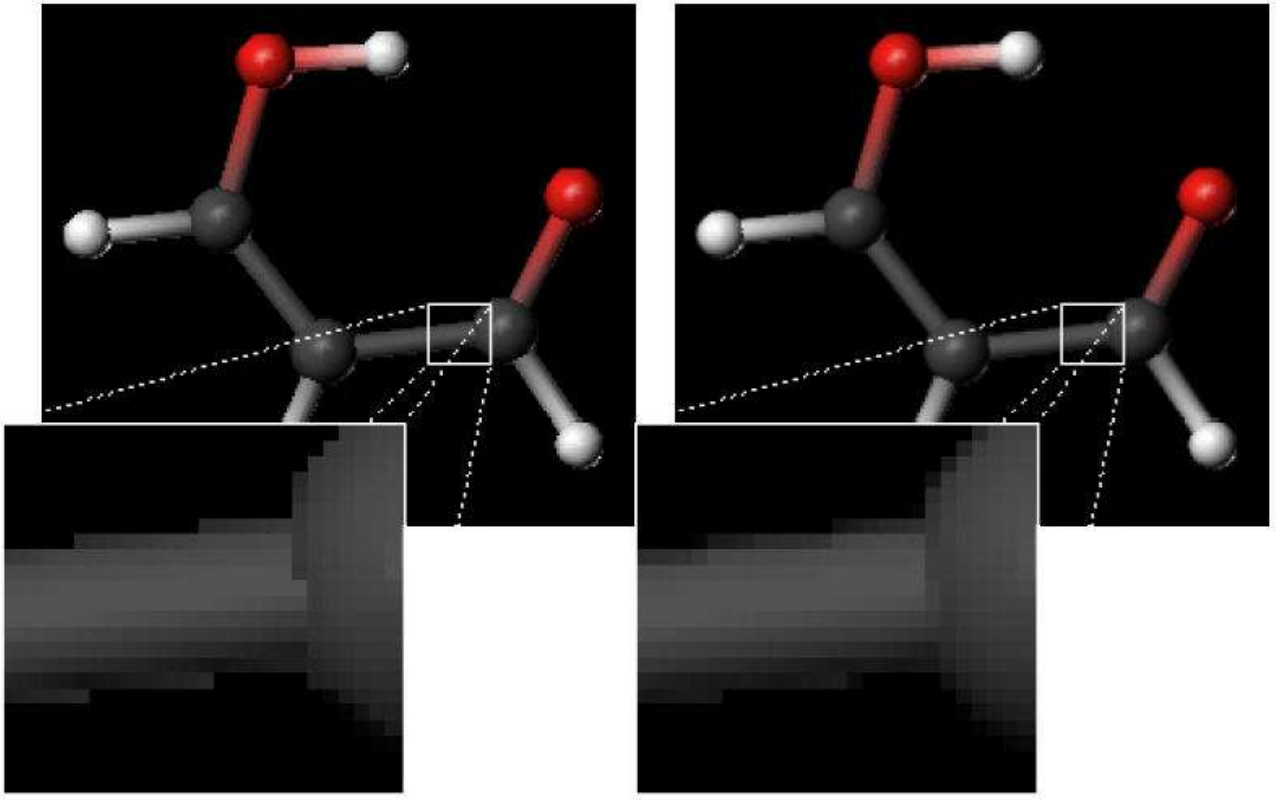
- The rendering speed is only reduced by a factor of 2 or less for scenes with more details, because for each frame only one image is stored and one image is loaded.



Accumulation-Buffer

- The right picture was created by accumulating 24 images as described in the Depth-Of-Field section.

- This time, the position of the eye is held fixed and the model is moved in fractions of a pixel around the original position.



What is a Buffered Image?

- A buffered image is a type of image whose pixels can be modified. For example, you can draw on a buffered image and then draw the resulting buffered image on the screen or save it to a file.

- A buffered image supports many formats for storing pixels.

- Although a buffered image of any format can be drawn on the screen, it is best to choose a format that is the most compatible with the screen to allow efficient drawing

What is difference between A-buffer algorithm Z-buffer algorithm?

- A Z-buffer is a raster buffer that stores colour and depth information at each pixel.
- The "z" in the title refers to the "z" plane in 3D space, which is traditionally thought of as the "depth" dimension.
- The Z-buffer initialises each pixel to the default colour and an infinite depth.
- During the rendering process, when a colour is written to a pixel, it first compares the current depth of the colour in the pixel.
- If the new colour is closer than the current colour but closer than the clip plane (which is typically zero), the colour is written and the depth updated.
- In that sense, it's similar to the painter's algorithm, where the closer object covers the further object.

A-Buffer adds anti-aliasing...

- The A-buffer uses the same algorithm for handling depth, but adds anti-aliasing AND the facility for order independent transparency.
- Each pixel contains a set of sub-pixels.
- During the write operation, the values are accumulated at the sub-pixel level.
- For the final pixel read, the final colour is the sum of all the sub-pixels.
- The algorithm was originally developed by Loren Carpenter at Pixar for the RenderMan renderer.

Micro-polygons

- The positions of the sub-pixels in each pixel are randomly selected in space and time, which allows smooth blurring of moving objects.

- RenderMan dices geometry down to micro-polygons (polygons approximately the size of a pixel), and then performs a coverage test to determine if a sub-pixel is covered by a micro-polygon.

- However, this approach doesn't work with a more "typical" renderer, since they typically deal with points, which unlike micro-polygons, have no surface area.

Micro-polygons

- A common adaption of this algorithm is the accumulation technique, which renders an image multiple times, randomly jittering (moving) the position of the eye point by some small amount.

- The result of each rendering is accumulated and averaged into single buffer.

- This approach is made practical with a hardware accelerated renderer such as OpenGL.

- However, this approach is probably better thought of as super-sampling rather than an A-buffer.

Accurate Image Generation and Interactive Image Editing with the A-buffer

- The A-Buffer is alive and well!
- New paper by Wing Hung Lau, Neil Wiseman
- In this paper, they suggest a way of increasing the sub-pixel resolution by storing the bitmask index rather than the bitmask.
- This allows much more accurate images to be generated while at the same time, minimising memory usage. It also allows zooming to reveal more information.
- They also suggest an enhancement to the A-buffer by allowing the creation of dynamic objects. These dynamic objects can then be edited (deleted, moved, etc.) interactively as image modification and assembly is going on.

Deferred shading using the G-Buffer

- In computer graphics, deferred shading is a three dimensional shading technique in which the result of a shading algorithm is calculated by dividing it into smaller parts that are written to intermediate buffer storage (called G-buffer) to be combined later, instead of immediately writing the shader result to the colour frame buffer.

- Implementations on modern hardware tend to use multiple render targets (MRT) to avoid redundant vertex transformations.

- Usually once all the needed buffers are built they are then read (usually as input textures) into a shading algorithm (for example a lighting equation) and combined to produce the final result.

- In this way the computation and memory bandwidth required to shade a scene is reduced to those visible portions, thereby reducing the shaded depth complexity.

Disadvantages (1)

- The inability to handle transparency within the algorithm, although this problem is a generic one in Z-buffered scenes and it tends to be handled by delaying and sorting the rendering of transparent portions of the scene.

- Depth peeling can be used to achieve order-independent transparency in deferred rendering, but at the cost of additional batches and g-buffer size.

- Modern hardware, supporting DirectX 10 and later, is often capable of performing batches fast enough to maintain interactive frame rates.

- When order-independent transparency is desired (commonly for consumer applications) deferred shading is no less effective than forward shading using the same technique.

Disadvantages (2)

- The difficulty with using multiple materials. It's possible to use many different materials, but it requires more data to be stored in the G-buffer, which is already quite large and eats up a large amount of the memory bandwidth

Disadvantages (3)

- One more rather important disadvantage is that, due to separating the lighting stage from the geometric stage, hardware anti-aliasing does not produce correct results any more.

- Although the first pass used when rendering the basic properties (diffuse, normal etc.) can use anti-aliasing, it's not until full lighting has been applied that anti-alias is needed.

The Future???

- **I predict a variant of the original REYES system.**
- **Created using a technology such as Direct Compute**

DirectCompute

- Microsoft DirectCompute is an application programming interface (API) that supports general-purpose computing on graphics processing units on Microsoft Windows Vista and Windows 7.

- DirectCompute is part of the Microsoft DirectX collection of APIs and was initially released with the DirectX 11 API but runs on both DirectX 10 and DirectX 11 graphics processing units.

- The DirectCompute architecture shares a range of computational interfaces with its competitors - the Khronos Group's Open Computing Language (OpenCL) and NVIDIA's Compute Unified Device Architecture (CUDA).

END