上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

游戏设计与开发

Game Visual Effects (VFXs)

上海交通大学软件学院
数字艺术实验室
Digital ART LAB

Environmental Effects

- Natural fuzzy participating media or effects (smoke, fire, rain, cloud, explosion, …)
- Particle systems is the most widely used visual effects technique in game

Particles

- A set of particles to model, animate and render fuzzy effects (smoke, fire, cloud, rain, and etc.)
- E.g.: camera-aligned quadrilaterals (billboards)

Particle systems

- In today's game:
  - —commonly used for smoke, explosions
  - —particles (~10,000)
  - —point sprites with painted or pre-rendered textures
  - —simple lighting: normal mapping
  - —few interaction
- In tomorrow's game:
  - —similar to particle effects used in film
  - —particles (millions)
  - —physics simulation with artist control
  - —high quality shading and shadowing
  - —interaction with scene and characters

Evolution of particles

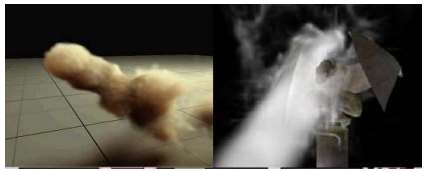
- Hard particle
- Soft particle
- Volumetric particle
- Interactive dynamic particles

Hard particle vs. Soft particle

- Hard particle: regular flat billboards
  - —seam between particle and other objects
- Soft particle: fragment shader
  - —sample from the depth buffer (GPU support)
  - —fading the particle if get closer to other geometry
  - —but add expenses (may down-sampling the depth buffer)

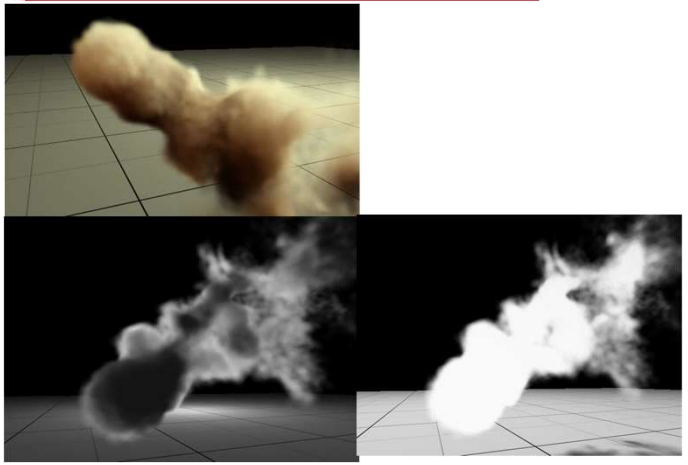
More Advanced Particles

- volumetric shadowed particle (with volume particle shadows)
- Interactive dynamic particles (e.g. interactive smoke particles with fluid dynamics)

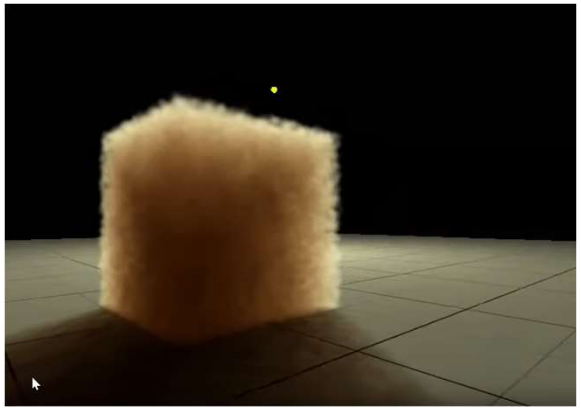


From Nvidia

Volumetric shadowed particle



Volumetric shadowed particle



Self-shadowing of particles

- maybe most important aspect of shading dense particles
- Important cues to density/shape of a cloud of particles
- Alternative Approaches (volume rendering)
  - View ray matching → sample → matching to light (large number of samples: very expensive)
  - Shadow volume + look up amount of light

  Above methods: voxelization is slow; need a lot of storage; particles confined to a fixed volume (often freely move in game)

  One solution: use a view-aligned 3D texture (video memory costs)

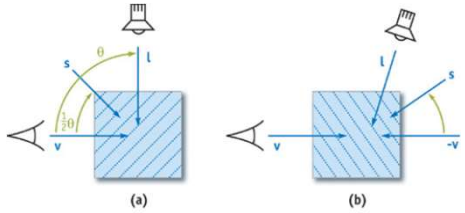
  Opacity shadow maps:

  -replace shadow volume with a set of 2D textures

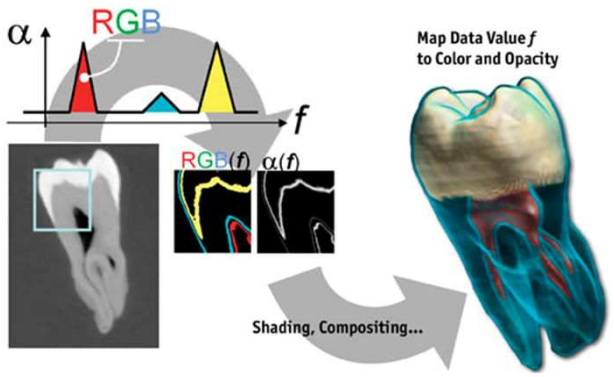
  -relatively large amount of storage, banding artifacts visible

Half-Angle Slice Rendering

- Extension of a volume rendering technique
- Key idea:
  - Calculate a half way vector between V and L
  - Render the volume as a series of slices perpendicular to this half-angle vector



Direct Volume Rendering



Direct Volume Rendering

- Ray casting vs. Texture-based methods
- Volume data is stored as
  - A stack of 2D texture slices or
  - A single 3D texture object
- Voxel: volume element, similar to pixel, texel
- Reconstruction
  - Each voxel corresponds to a location in data space and has one or more data values associated with it.
  - Values at intermediate locations are obtained by interpolating data at neighboring volume elements.

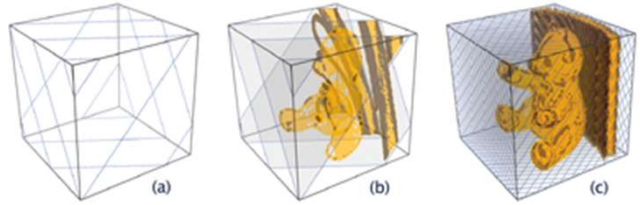
Optical model

- **Optical model:** describe how particles in the volume interact with light.
  - the most commonly used model assumes that the volume consists of particles that simultaneously emit and absorb light.
  - More complex models incorporate local illumination and volumetric shadows, and they account for light scattering effects.
- Optical parameters are
  - specified by the data values directly
  - or they are computed from applying one or more *transfer functions* to the data.

Transfer function

- The goal of the transfer function in visualization applications is to emphasize or *classify* features of interest in the data.
- Typically, transfer functions are implemented by *texture lookup tables*, though simple functions can also be computed in the fragment shader.

Volume Sampling and Compositing



emission-absorption model

Discrete Volume Rendering Equations

$$C = \sum_{i=1}^{n} C_i \prod_{j=1}^{i-1} (1 - A_j)$$

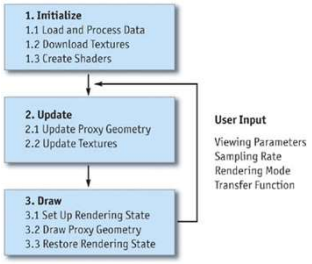
Opacity A: absorption
Color C (opacity-weighted): emission & absorption along the ray segment between i and i+1

$$A = 1 - \prod_{j=1}^{n} (1 - A_j)$$

Texture-based volume rendering

- Texture-based volume rendering techniques perform the sampling and compositing **steps by rendering a set of 2D geometric primitives inside the volume**
- Each primitive is assigned texture coordinates for sampling the volume texture.
- The ***proxy geometry*** is rasterized and blended into the frame buffer in **back-to-front or front-to-back order**.
- In the fragment shading stage, the interpolated texture coordinates are used for a data texture lookup.
- Next, the interpolated data values act as texture coordinates for a *dependent lookup* into the transfer function textures. Illumination techniques may modify the resulting color before it is sent to the compositing stage of the pipeline.

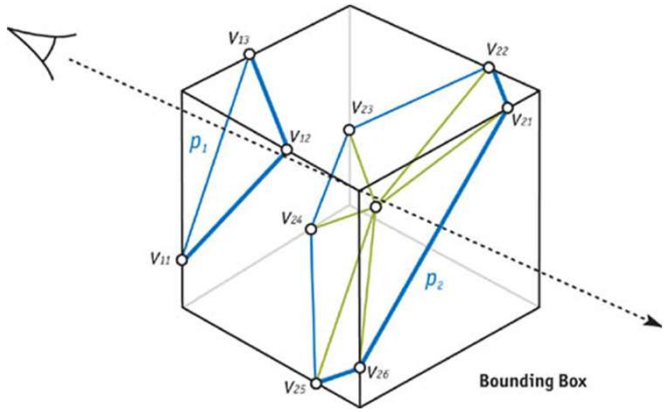
Texture-based Volume Rendering



Fragment Shader of a Simple Volume Renderer

```
void main(uniform float3 emissiveColor,
        uniform sampler3D dataTex,
        float3 texCoord : TEXCOORD0,
        float4 color : COLOR)
{
  float a = tex3D(texCoord, dataTex);  // Read 3D data texture
  color = a * emissiveColor;           // Multiply color by opacity
}
```


View-Aligned Slicing with Two Sampling Planes



Over and Under Operators

- Back-to-Front Compositing (the Over Operator)

$$\hat{C}_i = C_i + (1 - A_i)\hat{C}_{i+1},$$

$$\hat{A}_i = A_i + (1 - A_i)\hat{A}_{i+1},$$

- Front-to-Back Compositing (the Under Operator)

$$\hat{C}_i = (1 - \hat{A}_{i-1})C_i + \hat{C}_{i-1},$$

$$\hat{A}_i = (1 - \hat{A}_{i-1})A_i + \hat{A}_{i-1},$$

Volumetric Light

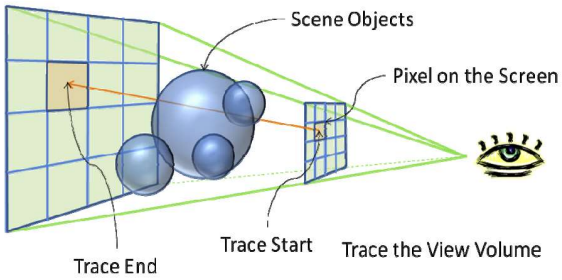
Volumetric Light

- Optical Principle
  - —Participating media: light scattering
  - —Occluding matters: leaves, buildings, clouds…
  - —Strong contrast of light and dark area

- Approximation in game
  - —Visual effect vs. physically correct simulation

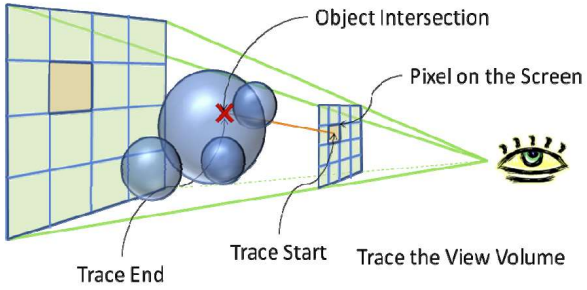
Volumetric Light

- from Nvidia White Paper “Volume Light” 2008



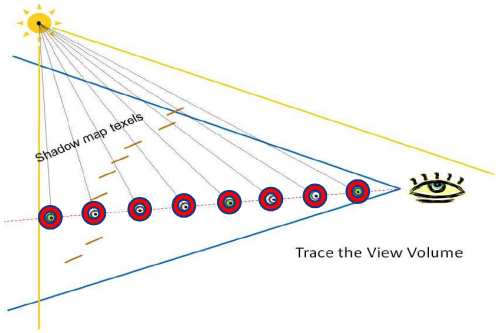
Volumetric Light

- Check scene depth buffer : find Trace End



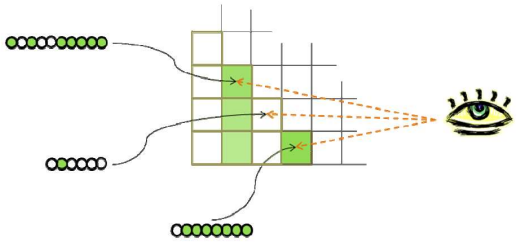
Volumetric Light

- light space shadow map comparison: determine lit samples



Volumetric Light

- Accumulate lit samples: calculate final intensity



Volumetric Light

- Optimization 1 (downsample: 1/4): smaller resolution to reduce computation cost
- Problem: aliasing

Volumetric Light

- Sobel filter: for edge detection

1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
1	-2	0	2	1

Gx

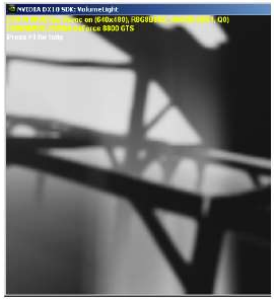
Gy

1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

5x5 filter kernels. Edge direction is defined as float2(Gx, Gy).

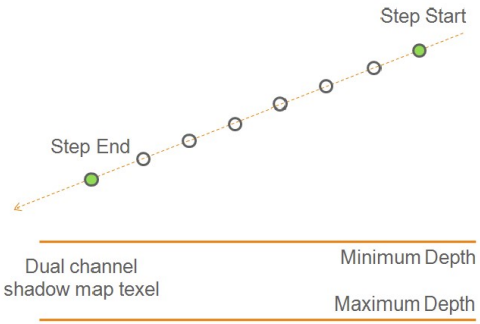
Volumetric Light

- Edge blurring



Volumetric Light

- Optimization 2 (variable sampling step sizes): faster tracing

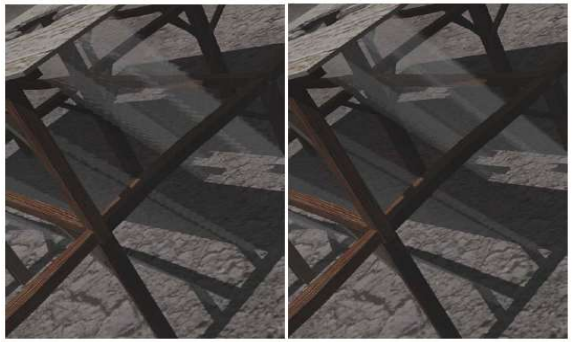


Volumetric Light

- Optimization 3 (increase light intensity non-uniformly for holes): better visual quality
- Techniques:
  - Search for discontinuities in shadow map
  - Locate holes

Volumetric Light

- Optimization 4 (jittering initial sampling positions): solve banding artifacts due to regular sampling



Regular sampling(left) and the same scene with a jittering applied(right).

Volumetric Light

- Final result

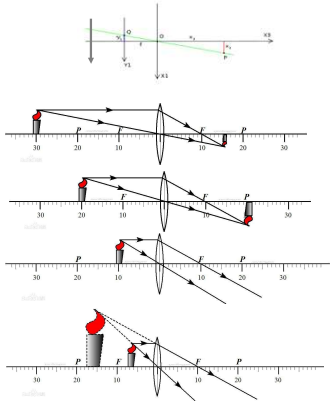
Sensor Effects

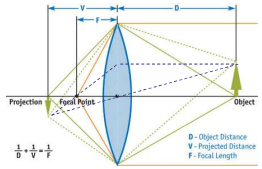
Sensor Effects

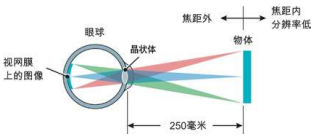
- Pinhole camera model: sharp, perfect images
- Real camera and our eyes: optical and mechanical limitations
- Simulate cameras and human visual system

- Often implemented as screen-space postprocessing
- Many postprocessing sensor effects:
  - Lens distortion, vignetting, chromatic abbreviation, after-imaging, film grain, tone adjustment, night and thermal vision …

Principles of Pinhole, Lens, Eyes



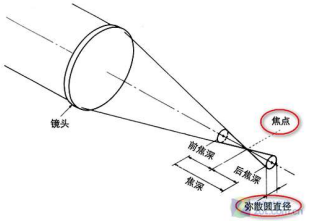


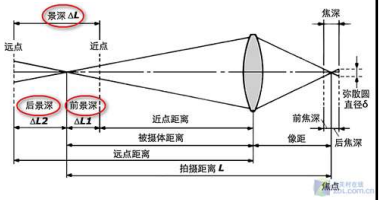


Depth of Field

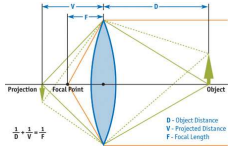
Depth of Field (DoF)

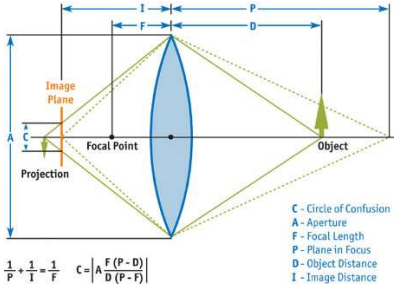
- Ideal pinhole camera: sharp image
- Real lens system:
  - Sharp at focus distance
  - otherwise form a circle of confusion (CoC)
  - depth of field: the portion of scene in focus
- DoF in game: sense of aiming at a certain target



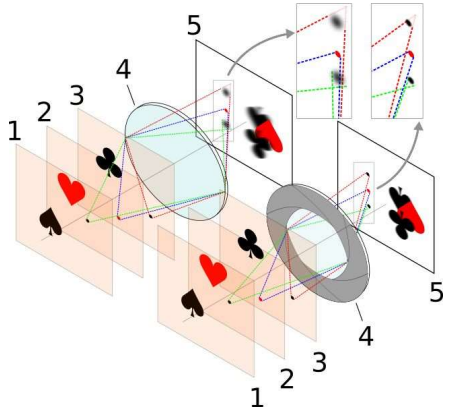


Circle of Confusion (CoC)





Depth of field

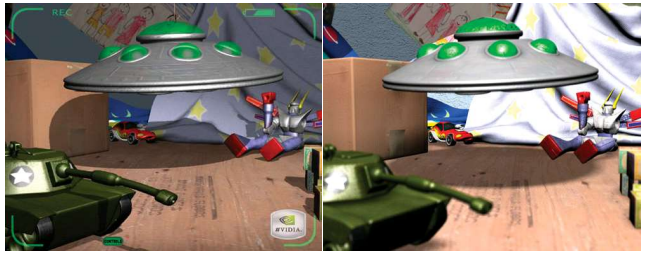


Lens Factors

	Aperture Size	Exposure	Depth of Field
f/1.4	Very large	Lets in a lot of light	Very thin
f/2.0	Large	Half as much light as f/1.4	Thin
f/2.8	Large	Half as much light as f/2	Thin
f/4.0	Moderate	Half as much light as f/2.8	Moderately thin
f/5.6	Moderate	Half as much light as f/4	Moderate
f/8.0	Moderate	Half as much light as f/5.6	Moderately large
f/11.0	Small	Half as much light as f/8	Large
f/16.0	Small	Half as much light as f/11	Large
f/22.0	Very small	Half as much light as f/16	Very large

DoF Techniques

- Distributed Ray-traced DoF
- Accumulation-Buffer DoF (from multiple cameras)
- Layered DoF
- Forward-mapped z-buffer techniques
- Reverse-mapped z-buffer techniques

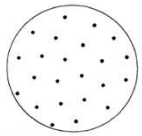


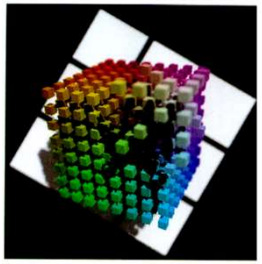
Ray-Traced Depth of Field

- Casting rays across the lens
- High quality; degrades nicely to noise
- slow

Accumulation-Buffer Depth of Field

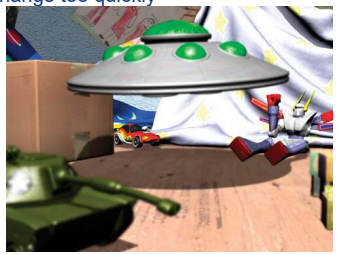
- approximate ray-tracing DoF using a z-buffer + an accumulation buffer
- render the scene multiple times from different locations through the lens
- Multi-passes
- E.g. 50 passes for a 8 pixel radius CoC (roughly 200 pixels)
- ghosting (less passes or banding (integer)





Layered Depth of Field

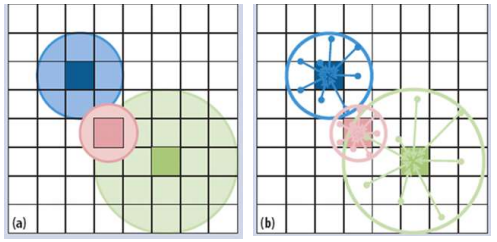
- objects sorted in layers with non-overlapping depth
- each layer can be blurred, then composited
  - —not technically correct (unnoticed mostly)
  - —not good for object span large depth ranges
  - —blur change too quickly



Depth of field

- **Postprocessing: use depth buffer to compute CoC**
  - **a) forward mapping (spreads to CoC)**
  - **b) or reverse mapping (averaging surrounding pixels)**
    - **more efficient in current GPU fragment shader**
    - **but not support Bokeh: the shape of CoC**

(Capcom's Lost Planet: Bokeh with forward mapping DoF using Geometry Shader)



Example Result

- **Forward mapping vs. Reverse mapping**

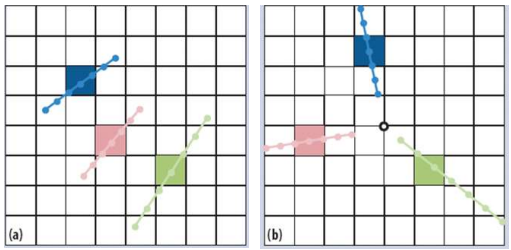
Motion Blur and Radial Blur

- **Integral of lighting over the exposure period**
- **A sense of motion in game**

From Ubisoft EndWar

Motion Blur and Radial Blur

- Averaging multiple rendering results (too slow)
- Post-processing in game: average sample points
  - —the direction and length of the line segment depends on
    - a) the velocity vector in that pixel (motion blur)
    - b) or the relative position from the screen center (radial blur)



Motion blue example

- GTA5

High Dynamic Range

- Allow brightest and darkest details together
- First used in Valve's Half-Life: Lost Coast (2005)

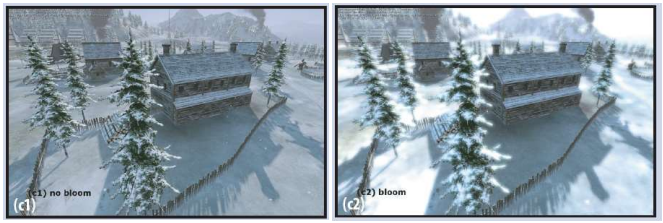
From Ubisoft's Endwar

LDR vs. HDR

- Low dynamic range (LDR) model:
  - —8 bits/per pixel color channel
  - —out-of-range values are clamped during computation
  - —precision is lost
- High dynamic range (HDR) imaging and lighting:
  - —more bits for each color channel as floating-point
  - —higher precision
  - —Implemented on GPU with floating-point texture format and sufficient instruction count

Bloom

- **HDR enables effects such as *bloom (glow around a bright light: e.g. snow in sun) or glare effects***
  - **first extract the bright pixels**
  - **then overlay a blurring filter**



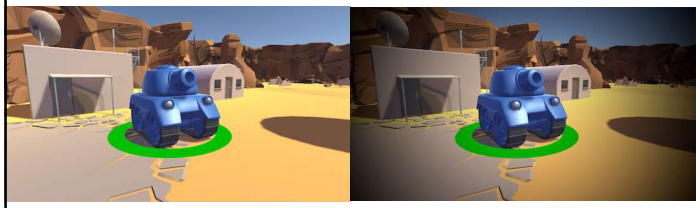
Vignetting

- Edges of an image darker than the center
- Causes: mechanical, optical, pixel, natural

in CV: image vignetting correction
In CG: image vignetting synthesizing

Vignetting

- **Vignette** effect darkens the edges of an image, leaving the center of the image brighter.
- In photography, this effect is caused by thick or stacked filters, secondary lenses, and improper lens hoods.
- For artistic effect: draw focus to the center of an image



Chromatic aberration

- Simulate low quality/old cameras
- Multi-colored blurring around the edges of objects, marked by a red/green/blue separation
- Artistic effects: camera impact, or intoxication effects

Film grain

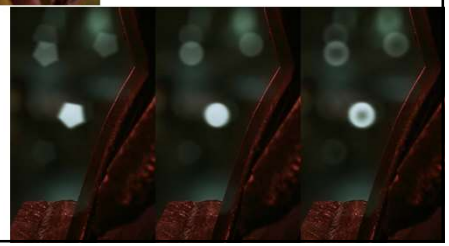
Film grain

- Emulating the apparent imperfections of film
- Camera's film give the image a coarse, unprocessed effect
- a coherent gradient noise (in Unity)
- Camera effect or historical story in games, or horror-themed games

Film grain

- Film grain On (Left), Off (right)

Bokeh



Lens flare

Video

- Ubi Game VFX video

Further VFXs: More Physically based

- Physically based rendering
- Simulation of physical lens

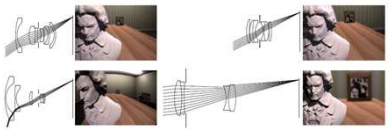
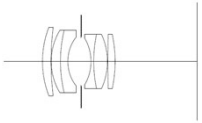
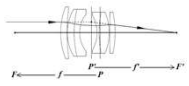


Figure 8: Four views of the same scene taken with a 16mm fisheye lens (bottom left), 35mm wide-angle lens (top left), 50mm double-Gauss lens (top right), and a 250mm telephoto lens (bottom right). A profile view of the lens system used to take each image is shown on the left. As with physical lenses, perspective is compressed with long focal lengths and expanded with short focal lengths. The fisheye image shows the lens' signature barrel distortion.





radius	thick	n_d	V-no	ap
58.950	7.520	1.670	47.1	50.4
169.660	0.240			50.4
38.550	8.050	1.670	47.1	46.0
81.540	6.550	1.699	30.1	46.0
25.500	11.410			36.0
	9.000			34.2
-28.990	2.360	1.603	38.0	34.0
81.540	12.130	1.658	57.3	40.0
-40.770	0.380			40.0
874.130	6.440	1.717	48.0	40.0
-79.460	72.228			40.0

Image Courtesy of Pat Hanrahan(1995)
"A realistic camera model for computer graphics"

Further VFXs: More interaction

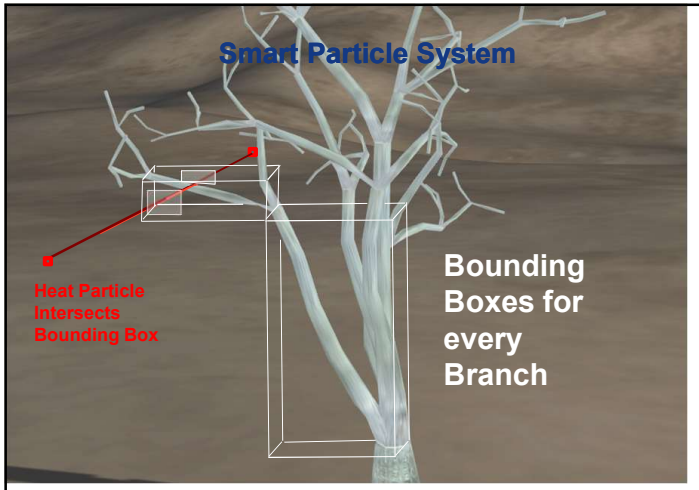
- Combine visual effects with game physics
- Enable new type of game play
  - E.g. fire propagation in Ubisoft's FarCry 2

Smart Particle System

Series of Particle Engines made up of two parts

- Fire Particles
  - Visible, what we think of as particles
- Heat Emitter Particles
  - Invisible, physical in nature

Smart Particle System



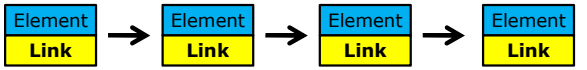
Procedural Fire

- Trees catch on fire spreading from branch to branch and then tree to tree.
- Smart Particle Systems use heat particles generated by each fire and the fire spreads procedurally.
- Configurable Flames and Heat particle behavior.
  - Density, Velocity, direction, etc.

Further VFXs: new GPU support

- DirectX 11 hardware opens the door to many new visual efffects
- E.g. per pixel linked lists allow for:
  Order-independent Transparency, Indirect Shadows, Ray Tracing of dynamic scenes, custom blending, Advanced Depth of Field, etc.
- *E.g. Hardware tesselletion* allow for cinema quality models with less space
- *E.g. Direct Compute* allow for more advanced post-processing functions

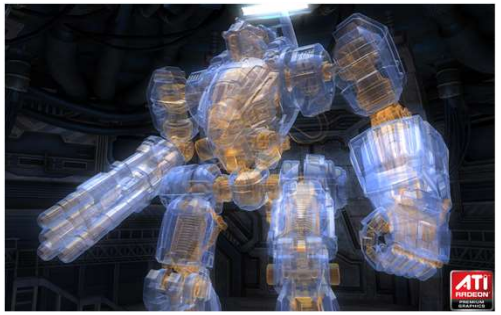
Per-pixel Linked List

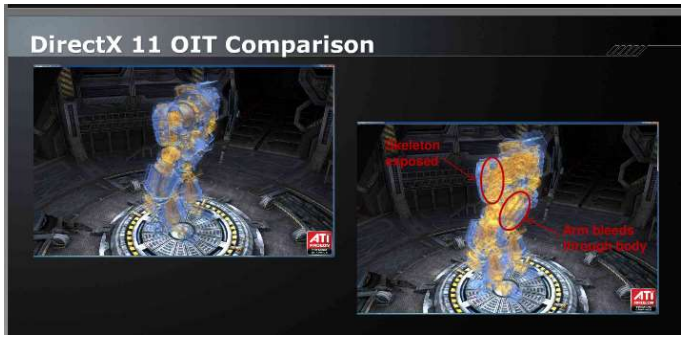


- Very hard to implement efficiently with previous real-time graphics APIs
- DX11 allows efficient creation and parsing of linked lists
- Per-pixel linked lists
  - A collection of linked lists enumerating all pixels belonging to the same screen position

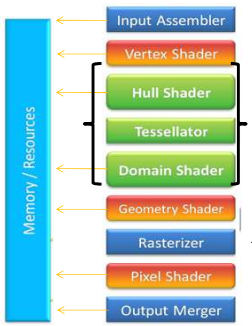
OIT result

- Enable order independent transparency (OIT)
  - Using *DirectX 11 atomic operation* and *append buffers*

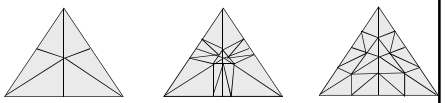




Hardware Tessellation

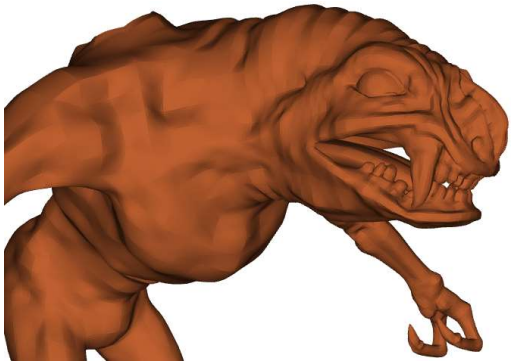


- Hardware subdivision: add more geometric details by computing ; less space

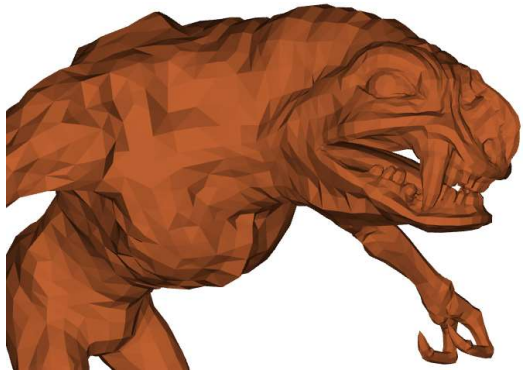


Level 1.0 Level 1.5 Level 3.0

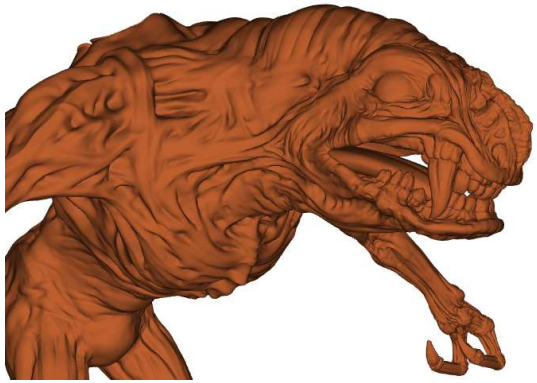
Coarse Model (from Metro 2033)

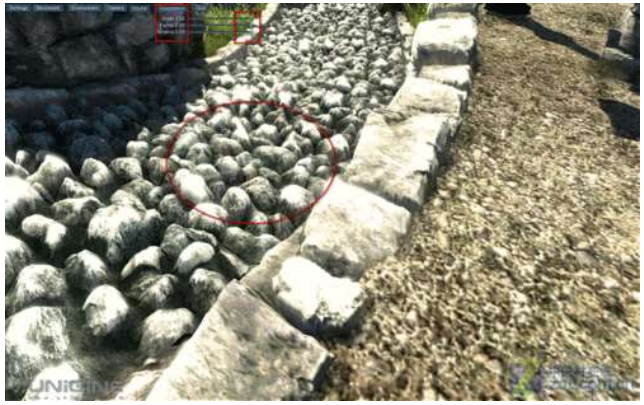


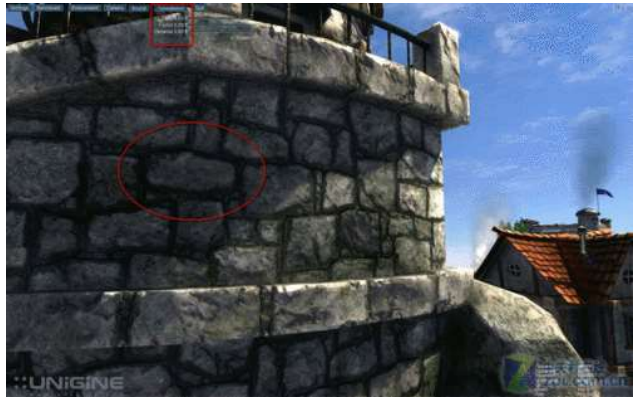
HW subdivision



Apply Displacement







Depth of Field

- Using novel techniques with DirectCompute 11: more accurate and more efficient post processing

Wrap Up

- Movie VFXs vs. Game VFXs
- People: designer, programmer and artists
- Tech horsepower: GPU
- Techniques: pre-compute, object-space, screen-space
- Phenomena (examples):
  - GI effects (light map, SSAO)
  - Env effects (particles, volumetric light)
  - Sensor effects (depth of field, motion blur, HDR)
- Trends:
  - Physically based rendering
  - visual + physics interaction
  - more powerful GPU functions enable better effects