上海交通大學
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

Outline

- Introduction
- Classification
- Global illumination effects
- Environmental effects
- Sensor effects

Introduction

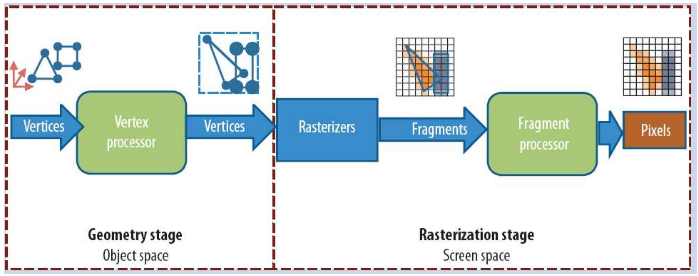
- Computer games:
  - —A virtual world simulating a real or imagined world
- Visual effects (VFXs) in computer games
  - —All graphics effects
  - —with special purposes
- Purposes:
  - —Simulate certain phenomena in our lives (e.g. lighting and shadowing, fire)
  - —Evoke an immersive illusion or a special emotional feeling (e.g. motion blur → sense of motion, fog → sense of mystery)

Examples of Game VFXs

From Big Screen to Small Ones

- **Movie VFXs:**
  - —Large screen with high resolution: e.g. 4096 x 2016
  - —Viewpoint predefined; non-interactive
  - —Computed offline on rendering farm
  - —Tuned in post production stage
- **Game VFXs:**
  - — Home screens (PC, TV): e.g. 1280 x 720
  - —Viewpoint changed interactively
  - —Computed in real-time (25-60 fps or 90-120 fps for VR) on graphics processor (GPU)
  - —Real-time is a must; with acceptable visual quality

GPU rendering pipelines



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

1896 1935 1987 2006

Classifications of Game VFXs

上海交通大学软件学院
数字艺术实验室
Digital ART LAB

Visual Effects Classifications

- **Phenomena:**
  - —Global illumination effects
  - —Environmental effects
  - —Sensor effects
- **Techniques:**
  - —Pre-computation
  - —Object space (vertex-based)
  - —Screen space (fragment-based) / postprocessing
- **May be mixing**
  - —E.g. explosion (particles + heat distortion)

Table 1. Classification of visual effects and techniques in computer games.

Computation stage	Global illumination effect	Environmental effect	Sensor effect	Flexibility	Computational complexity
Precomputation (offline)	Irradiance, shadow, color bleeding, precomputed radiance transfer (PRT), ambient occlusion (AO), irradiance volume	Skybox	None	Static scene	Scene complexity and level of detail
Object space (real-time)	Shadow, reflection, refraction, AO	Fog, sun shaft, smoke, fire, water, clouds, rain, snow, explosions	Motion blur	Dynamic scene	Scene complexity, level of detail, and screen resolution
Screen space (real time)	Refraction, color bleeding, SSAO, SSDO	Fog, sun shaft, rain, snow, heat distortion	Depth of field, motion blur, bloom, glare, afterimage, tone mapping, color correction, lens distortion, vignetting, chromatic abbreviation, film grain, night vision, thermal vision	Dynamic scene	Screen resolution only

Typical Visual Effects Workflow

- Visual effects artist
  - —conceive a visual effect
  - —Painting tool quickly draw a conceptual image
  - —Realize the effect using an existing tool (e.g. game editor plug-in module)
  - —Ask for a suitable tool
- Programmer
  - —Implement a tool and provide parameters for the artist to control
- Iteration
- Trends:
  - —programmer doing more work
  - —automate the production process

上海交通大学 Shanghai Jiao Tong University

1896 1935 1987 2006

Global illumination effects

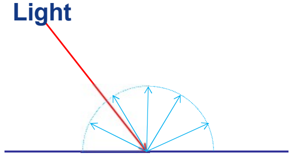
上海交通大学软件学院
数字艺术实验室
Digital ART LAB

Global illumination effects

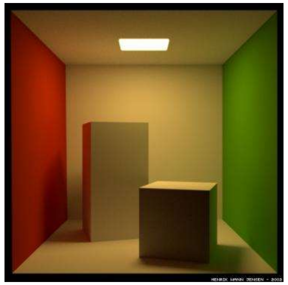
- direct illumination vs. indirect illumination
- Model indirect illumination effects and complex phenomenon
  - —reflection
  - —refraction
  - —color bleeding
  - —soft shadows
  - —…
- More complexity
  - —Light sources
  - —Materials
  - —Light transport paths

Radiosity

Energy leaves surface per unit time per unit area (W/m^2)



Lambertian (diffuse) surface

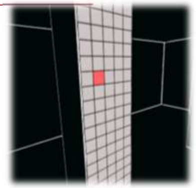


Color bleeding

Radiosity

Energy emitted + Energy reflected

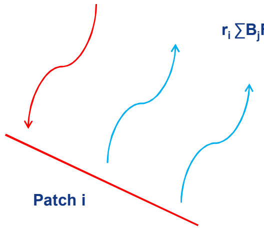
$$B_i = E_i + r_i \sum B_j F_{ij}$$



$\sum B_j F_{ij}$ energy from other patches

$r_i \sum B_j F_{ij}$ energy reflected by this patch

E_i energy emitted from this patch



Patch i

Radiosity Algorithm

- First developed in about 1950 in the engineering field of heat transfer.
- Refined for the problem of CG in 1984 by Cornell University researchers.

I - Incident
r - Reflectance
E - Emission
B - Radiosity

Segmentation
divide each surface into roughly equal sized patches

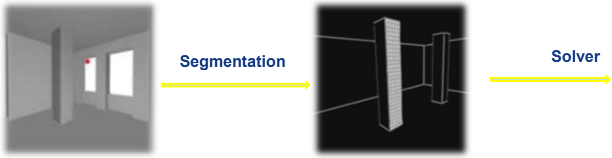
Initialise_patches
```
for each Patch in the scene
    if this patch is a light then
        patch.E = some amount of light
    else
        patch.E = black
    end if
    patch.B = patch.E
end Patch loop
```

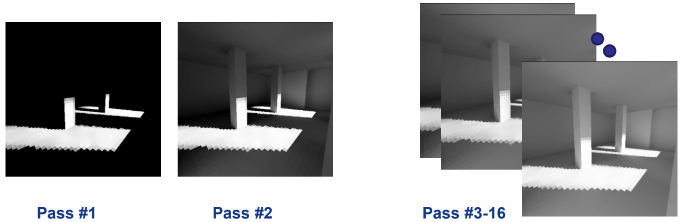
Passes_Loop

each patch collects light from the scene
```
for each Patch in the scene
    render the scene from the point of view of this patch
    patch.I = sum of incident light in rendering
end Patch loop
```

calculate radiosity for each patch
```
for each Patch in the scene
    patch.B = (I*r) + E
end Patch loop
```


Iterative Radiosity Solver

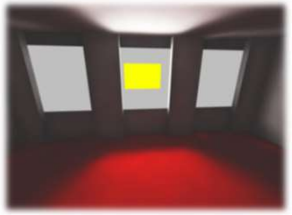




Pass #1 Pass #2 Pass #3-16

Light Map

- Radiosity is too slow for real time computation
- A light map is a 3D engine light data structure which contains the brightness of surfaces in a game.
- Light maps are pre-computed and used for static scenes/objects.

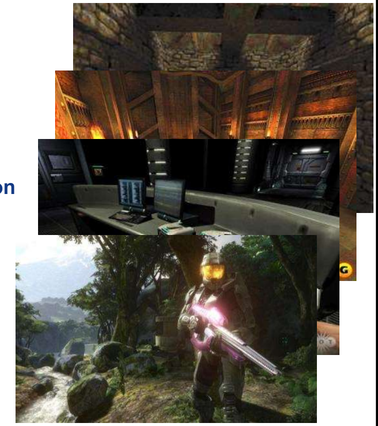


Light map tech (offline pre-lighting)

- Assume static relationship of light and scene object
  - —the GI light transport will be constant for static scenes
  - —pre-computation of light transport result and store at per-vertex or per-texel (exitance value, irradiance value or irradiance direction)
    - View-independent effects: (such as Lambertian diffuse reflection) : store exitance value
    - View-dependent effects (such as normal map, mirror reflection) : store irradiance value and direction, and calculate at run time
- dynamic object are treated in additional

Evolution of Light map techniques

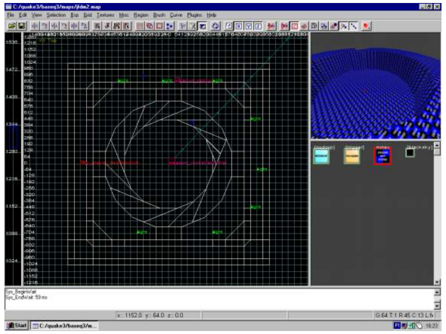
- Light maps
  - —1996, Quake:
    - static light map
  - —1999, Quake III:
    - per-vertex lighting
    - hardware acceleration
  - —2004, Doom 3:
    - per-pixel lighting
    - dynamic shadows
  - —2007, Halo 3:
    - dynamic GI
    - Spherical harmonic light maps



[Quake III Arena]

- Tech Info
  - Publisher: Activision
  - Developer: id Software
  - Genre: Sci-Fi First-Person Shooter
  - Release Date: Dec 2, 1999
- Minimum System Requirements
  - System: PII 233 or equivalent
  - RAM: 64 MB
  - Video Memory: 8 MB
  - Hard Drive Space: 70 MB
- Official Site
  - www.quakelive.com

[Quake III Arena]

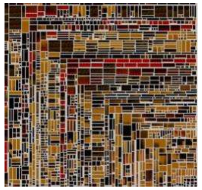


Pre-processing

Q3Radiant

Input:
1. Static scene
2. Static light

Output:
1. Light map
2. Parameterizations



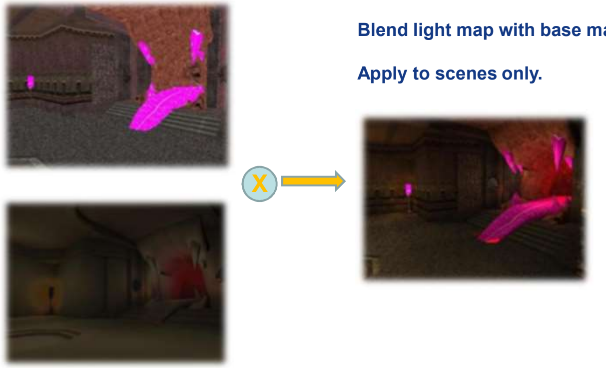
Digital ART Lab,SJTU

[Quake III Arena]

Rendering

Blend light map with base map.

Apply to scenes only.



Digital ART Lab,SJTU

[Quake III Arena]

Vertex Lighting

Light map

Digital ART Lab,SJTU

Ambient Occlusion (AO)

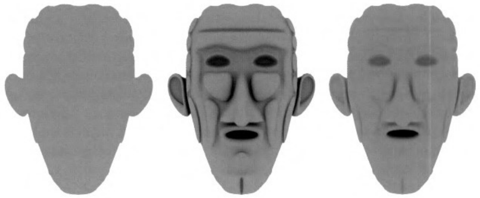
- Shadowing of ambient light
- Use the probability of being occluded by near objects to calculate AO factors
- AO factors do not consider light direction, only an approximation of ambient light shadowing
- Techniques:
  - —Pre-processing as in light map (for static object)
  - —Dynamic object (may use object-space method)
  - —Screen-based AO

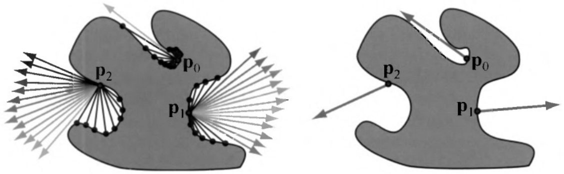
Digital ART Lab,SJTU

Ambient Occlusion (AO)

- Shadowing of ambient light $L_i(\mathbf{l}) = L_A$
  - —the softest shadow
  - —ambient light lacks any directional variation (so without AO, objects appear flat)

$$E(\mathbf{p}, \mathbf{n}) = \int_{\Omega} L_A \cos\theta_i d\omega_i = \pi L_A$$





- Use the probability of being occluded by near objects to calculate AO factors (visibility *v(p,l)*)
- AO factors do not consider light direction, only an approximation of ambient light shadowing

$$E(\mathbf{p}, \mathbf{n}) = L_A \int_{\Omega} v(\mathbf{p}, \mathbf{l}) \cos\theta_i d\omega_i$$

- AO factor K_A (ambient visibility: value [0,1])

$$k_A(\mathbf{p}) = \frac{1}{\pi} \int_{\Omega} v(\mathbf{p}, \mathbf{l}) \cos\theta_i d\omega_i.$$

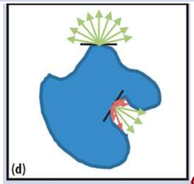
$$E(\mathbf{p}, \mathbf{n}) = k_A(\mathbf{p}) \pi L_A$$

- Techniques:
  - —Pre-processing as in light map (for static object)
  - —Dynamic object (may use object-space method)
  - —Screen-based AO

Ambient Occlusion

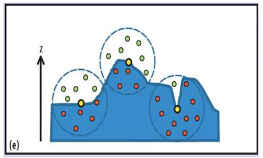
- *From Ubisoft's Endwar (AO map)*

- Object space methods:
  - —dependent on scene complexity



SSAO

- Screen space ambient occlusion
  - —independent of scene complexity
  - —depth buffer as a height field
  - —samples neighborhood points and computes the number of points passing the depth test.

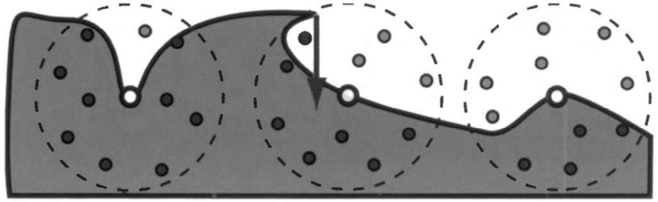


- Crytek's SSAO (Crysis, 2007)
  - —Only 16 texture fetches per pixel
  - —Use random rotated kernel
  - —make the noise high-frequency
  - —Post-blur step
  - —Edge-preserving blur

Digital ART Lab,SJTU

Crytek's SSAO

- The algorithm
  - —is executed purely on GPU and implemented as a pixel shader
  - —analyzing the scene depth buffer which is stored in a texture.
  - —For every pixel on the screen, the pixel shader samples the depth values around the current pixel and tries to compute the amount of occlusion from each of the sampled points.
  - —In its simplest implementation, the occlusion factor depends only on the depth difference between sampled point and current point.



Digital ART Lab,SJTU

Optimization

- a brute force method
  - —require about 200 texture reads per pixel for good visual quality.
  - —not acceptable for real-time rendering on GPU
- In order to get high quality results with far fewer reads
  - —sampling is performed using a randomly-rotated kernel
  - —The kernel orientation is repeated every *N* screen pixels in order to have only high-frequency noise in the final picture.

Digital ART Lab,SJTU

Optimization

- In the end, this high frequency noise is greatly removed by a *NxN* post-process blurring step taking into account depth discontinuities
  - —using methods such as comparing adjacent normals and depths.
- Such a solution allows
  - —a reduction in the number of depth samples per pixel to about 16 or less while maintaining a high quality result,
  - —and allows the use of SSAO in real-time applications like computer games.

Digital ART Lab,SJTU

SSAO

- **Advantages:**
  - Independent from scene complexity
  - no data pre-processing, no loading time
  - dynamic scenes
  - consistent way for every pixel
  - no CPU usage – completely on GPU
  - easily integrated into any modern graphics pipeline
- **Disadvantages:**
  - local and often view-dependent (dependent on adjacent texel depths)
  - Blurring noise may cause bleeding at object edges

SSAO in Crysis