CS 563 Advanced Topics in Computer Graphics
Image Based Rendering

by Suman Nadella

Outline

- What
- Why
- How
- Methods
- Applications
- References
- Conclusion

WHAT

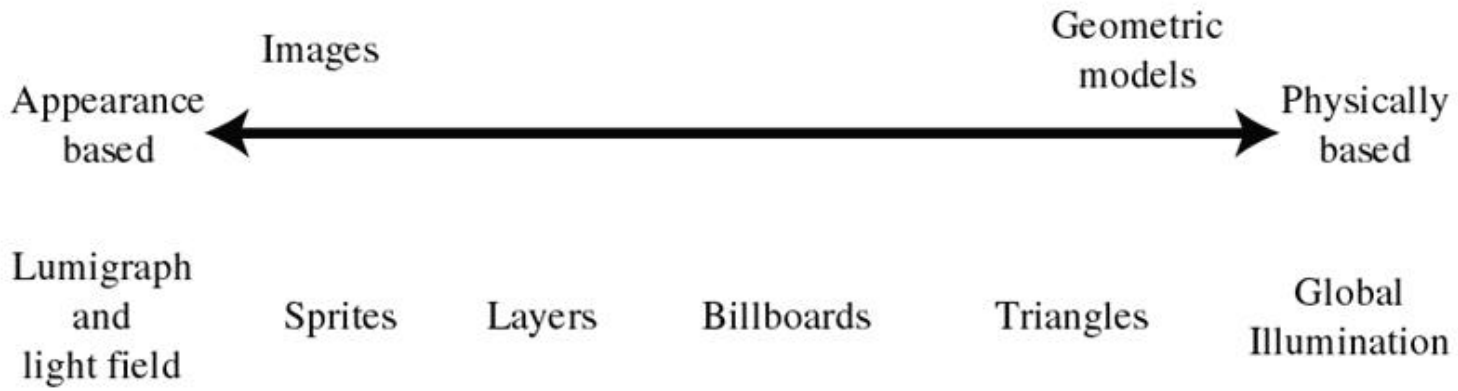
Properties

- Images are the primary source of Data
- Replace or enhance Polygon Models
- Complexity $\propto$ Number Of Pixels
- Precomputation capability

Driving Factors

- Make it as real as possible
- Fast, Faster ... Fastest (Fastestest?)
- Exploit LOD
- Industry trends
- FFF (Fire, Fog, Free forms)
- Give Computer Vision a chance !

HOW



- Pre calculate Images to isolate the scene complexity from rendering time
- Utilize the inherent coherence in the scene

Topics to Discuss

- Methods
  - Sprites
  - Billboards
  - Impostors
- Applications
  - Lens Flare and Bloom
  - Particle systems
  - Depth Sprites
  - Hierarchical Image Caching
  - Full Screen Billboarding
  - Skyboxes
  - Image Processing
  - Volume Rendering

Sprites

- Basic IBR – Image that moves around on the screen
- Classic Example – Mouse Cursor
- Rendered on a polygon
- Alpha channel
- No Warping / Projection
- Animation?

- Example

Layered Sprites

- Scene as a series of Layers
- Each layer has depth associated
- Render Back-to-front (avoid Z buffer)
- Zoom – easy to handle
- Camera movement perpendicular to the scene
- Change of view
- When to warp / When to recompute

Example

Chicken Crossing (Andrew Glassner , SIGGRAPH 96)

- 3D animated film rendered real-time
- Image Layers on Talisman Simulator

Example

Chicken Crossing (Andrew Glassner , SIGGRAPH 96)

- 3D animated film rendered real-time
- Image Layers on Talisman Simulator

Quick Note:

Interpenetrating Objects

Billboarding

- Orienting polygon based on View Direction
- Billboarding + Alpha + Animation = free forms

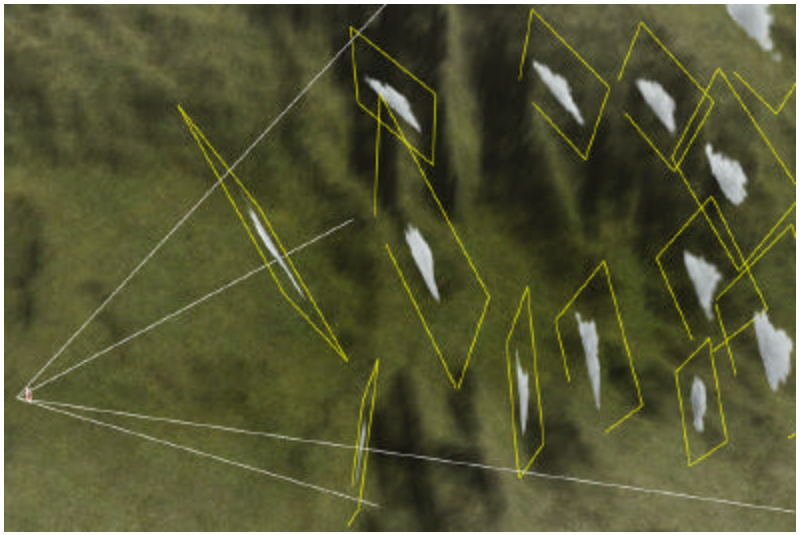
  (smoke, fire, explosions, clouds etc.)

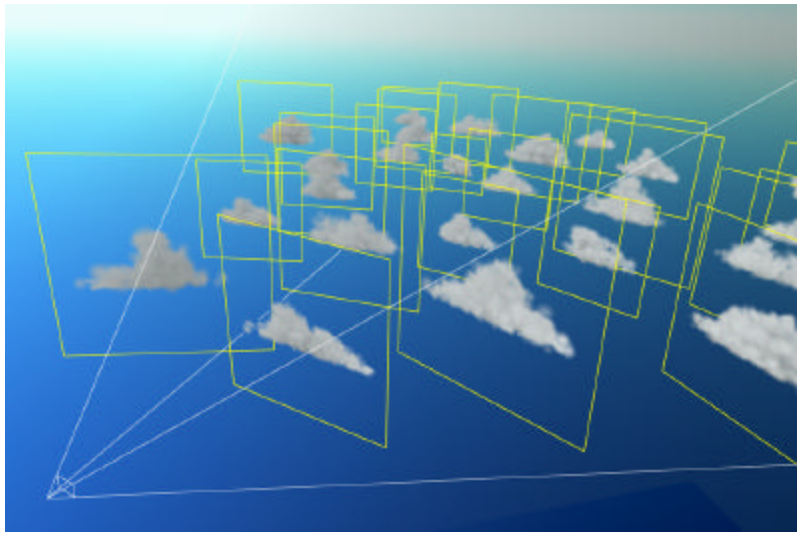
Lets talk about ...

- Basic Math
- Types of Billboards

Example

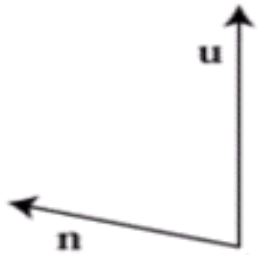
- Just to sustain interest ...
- Screen shots of billboards in action





Real time cloud rendering, Mark J. Harris

How to Billboard - 1



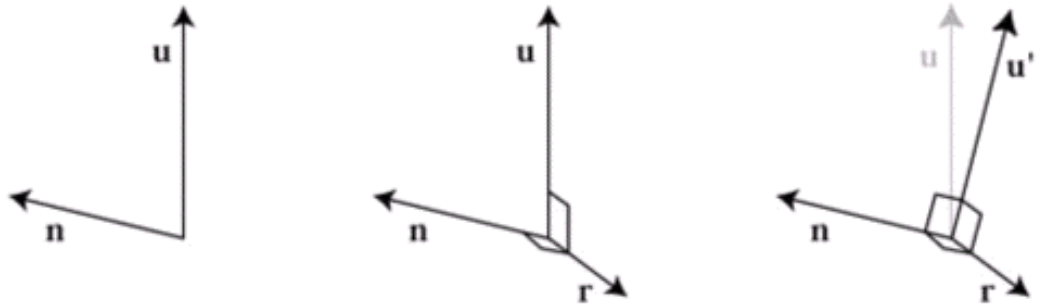
- Surface normal – n, Up direction – u

  Describe the orthonormal transforms

- Anchor Location (center)

  To establish its position in space

How to Billboard - 2



- u and n need not be perpendicular
- Fix one vector and find the perpendicular axes (say n in this case)
- Find r vector perpendicular to u and n (r = u x n)
- Now find u' perpendicular to both r and n (u' = n x r)
- Rotation Matrix M = (r, u', n)

Types of Billboards

- Differentiated based on which vector is kept constant
  - Screen Aligned Billboard
  - World Aligned Billboard
  - Axial Billboard

Screen Aligned Billboard

- Image always parallel to screen with constant up vector
- Surface normal = negation of view normal
- Up vector = camera's up vector
- Fixed n and u, thus fixed r
- Same for all billboards of this type

- *Uses: Text , Lens Flare etc*

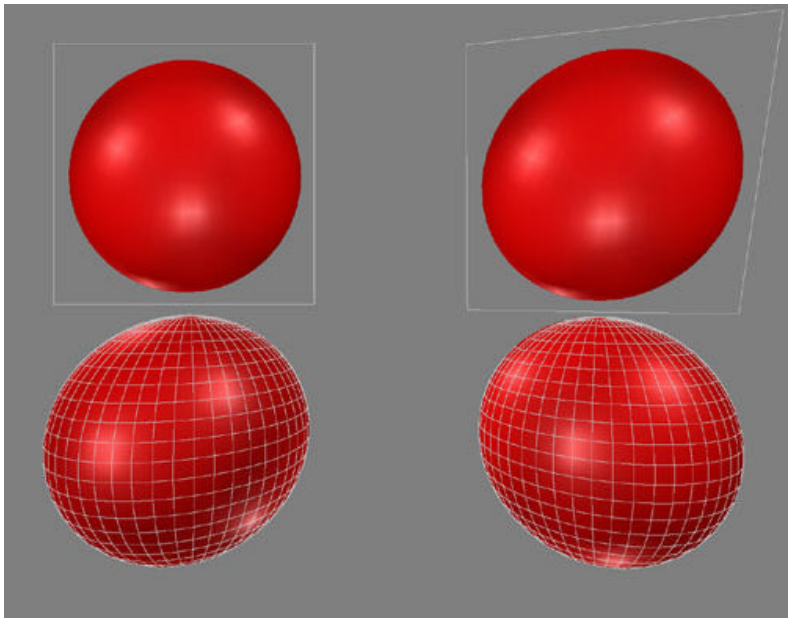
World Oriented Billboard

- Screen aligned works good for circular sprites
  - Up vector doesn't show any effect
- Orientation should be with respect to its world position rather than camera
- Use world up vector and same normal
- Same matrix again for all sprites...
- Good/Bad?

Perspective Projection...

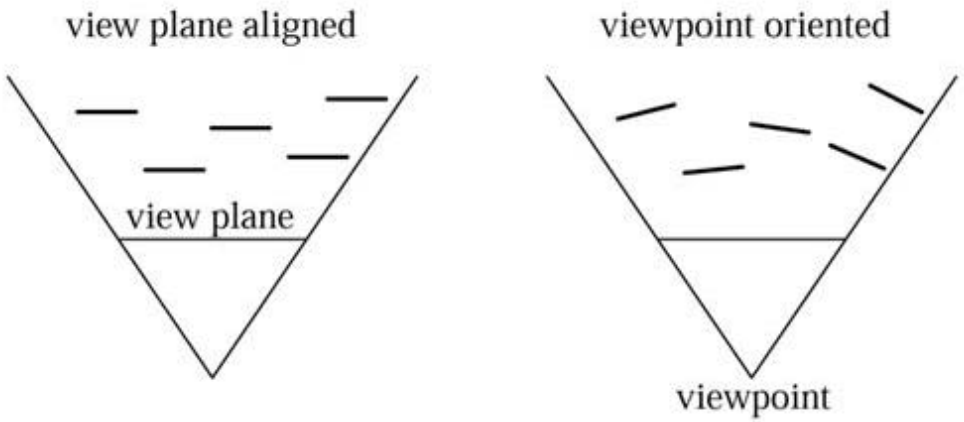
View Point Billboard - 1

- If camera's FOV does not match eye's FOV – warping
- Ignorable for small FOV / small Sprites , if not ...
- normal = vector (Billboard center to viewer's position)



A view of four spheres, with a wide field of view. The upper left is a billboard texture of a sphere, using view plane alignment. The upper right billboard is viewpoint oriented. The lower row shows two real spheres.

View Point Billboard - 2



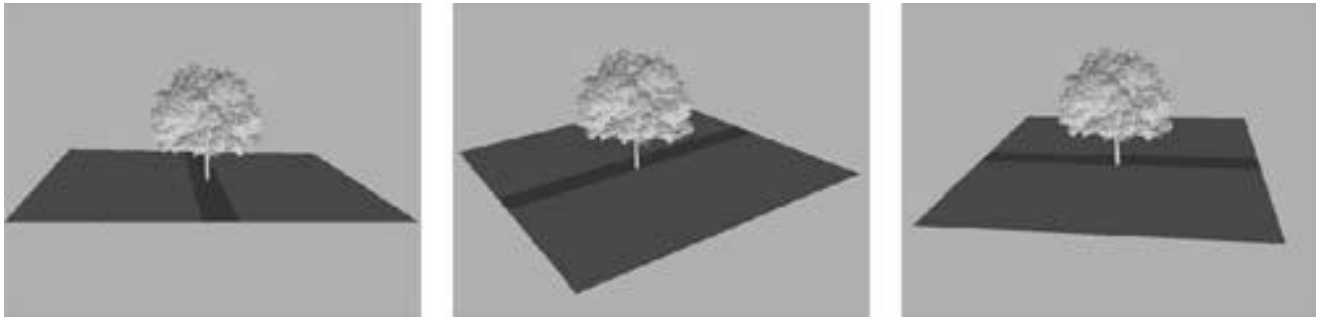
- Distortion shows up in view point orientation
- Looks similar to how real images get distorted
- Good for impostors

Axial Billboard - 1

- Cylindrical Symmetry (trees, laser beams etc)
- Does not face straight-on towards viewer
- Rotate around some world space axis and align to face user as much as possible
- Up vector is fixed, and view point direction is the adjustable vector

Axial Billboard - 2

- Tree example
- Single billboard v/s solid surface tree
- Up vector along tree trunk



- What if ...
- See it from top, will look like a cardboard cutout

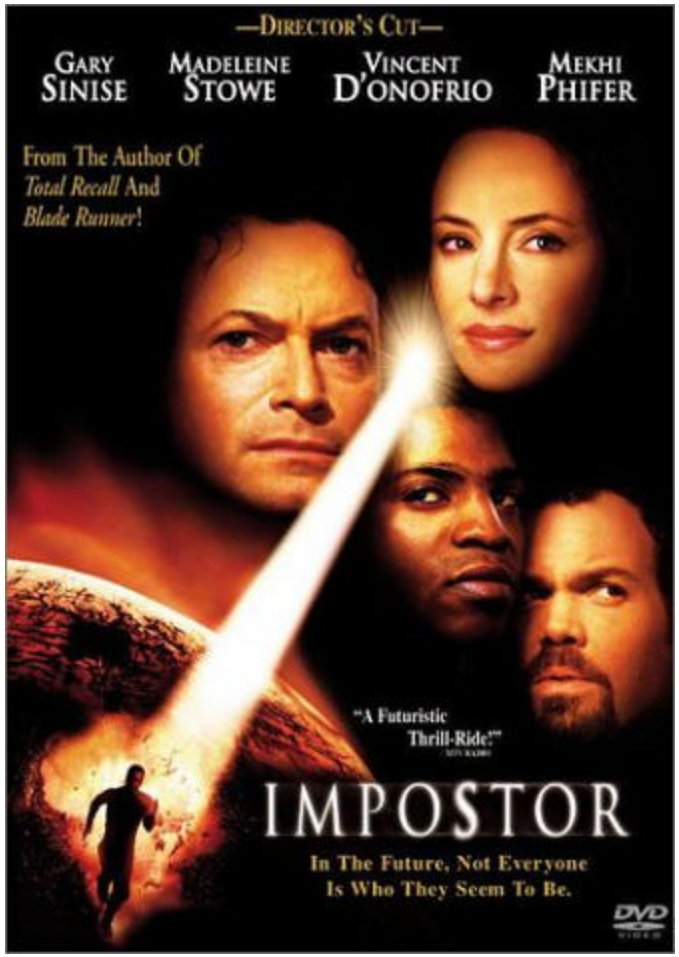
All in One

http://www.cs.unc.edu/~andrewz/twa/screenshots.html

All in One

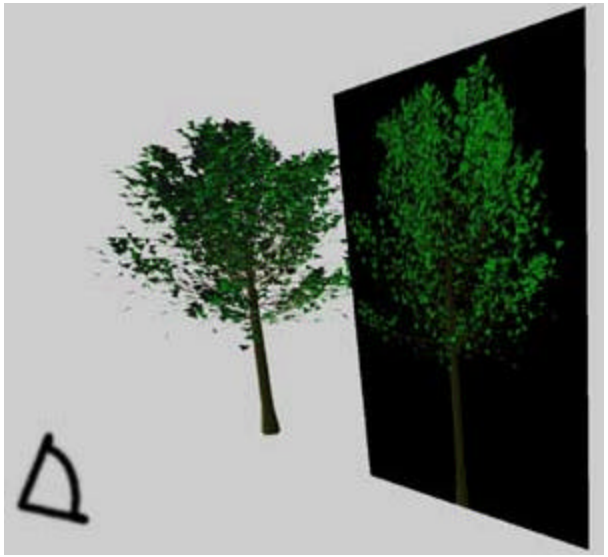
http://www.cs.unc.edu/~andrewz/twa/screenshots.html

Impostors


—Director's Cut—
Gary Sinise
Madeleine Stowe
Vincent D'Onofrio
Mekhi Phifer
From The Author Of
Total Recall And
Blade Runner!
"A Futuristic Thrill-Ride!"
IMPOSTOR
In The Future, Not Everyone
Is Who They Seem To Be.
DVD

Impostors

- Well, that was an impostor of an impostor !
- The real one...



Impostors Made Easy – William Damon, Intel

No Impostors

With Impostors

Impostors

- Billboard created on fly
- Render a complex object into image texture
- Mapped onto Billboard
- Few instances of Object / frames before update

- Why create them?

Impostors - 1

- Fast to draw
- Closely resemble the object
- Reuse for several viewpoints located close together
- Best for static and distant objects
  - Movement of object diminishes with distance from viewer
- Overcome low LOD constraints, since a high quality imposter can be created

- How to make them?

Impostors - 2

- Off-screen buffer
- Initialize alpha channel to 0.0 (transparent)
- Set to 1.0 when object is present
- Render the object with viewer looking at the center of bounding box
- Size of Impostor's Quad = bounding box
- New versions – render directly to texture (RGBa)
- Once rendered, normal points towards viewer (viewpoint oriented)
- Forsyth – project texture onto bounding box

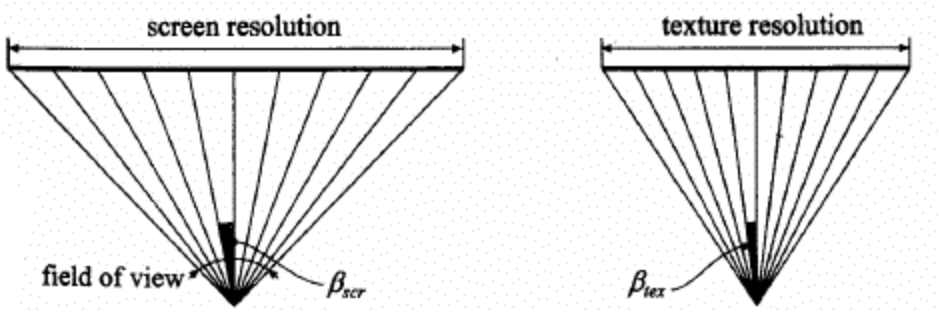
Impostors - 3

- Texture resolution need not exceed screen resolution
- texres = screenres * objsize/(2 * distance * tan(fov/2))

- When can this go wrong?
- Error > Threshold
  - Resolution
  - Point of view

- Lifetime of Impostor – how to find it...

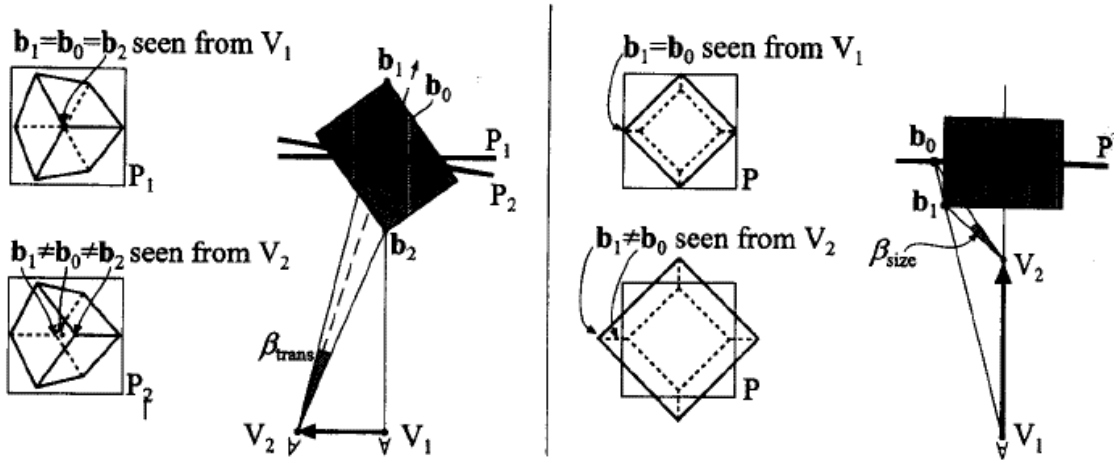
Resolution



- ßscr = angle of pixel (fov/screenres)
- ßtex = angle of texture (fov/texres)
- When ßtex > ßscr , recompute

Point of View

- Translation of viewer
  - ßtrans = angle between extreme points of bounding box
- Movement towards the impostor
  - ßsize = angle of extreme points projection on impostor plane



ßtrans > ßscr or ßsize > ßscr , recompute

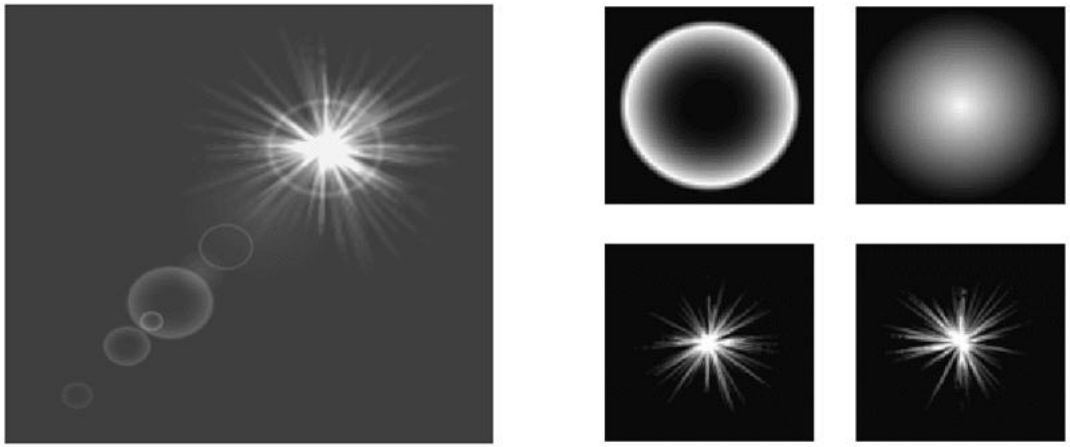
Cloud Impostors

Mark Harris (UNC Chapel Hill) - Real Time Cloud Rendering

DEMO

Lens Flare and Bloom - 1

- Caused by lens of eye/camera when directed at light
- Halo – refraction of light by lens
- Ciliary Corona – Density fluctuations of lens
- Bloom – Scattering in lens, glow around light



Halo, Bloom, Ciliary Corona – top to bottom

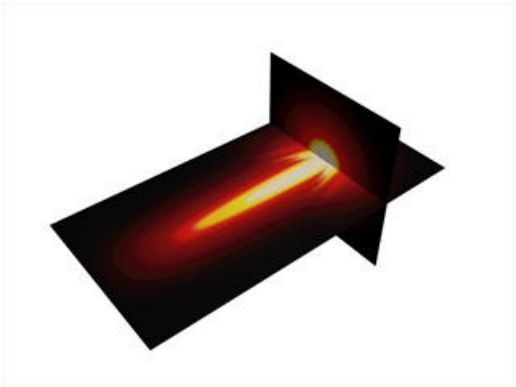
http://www.gamedev.net/reference/articles/article940.asp

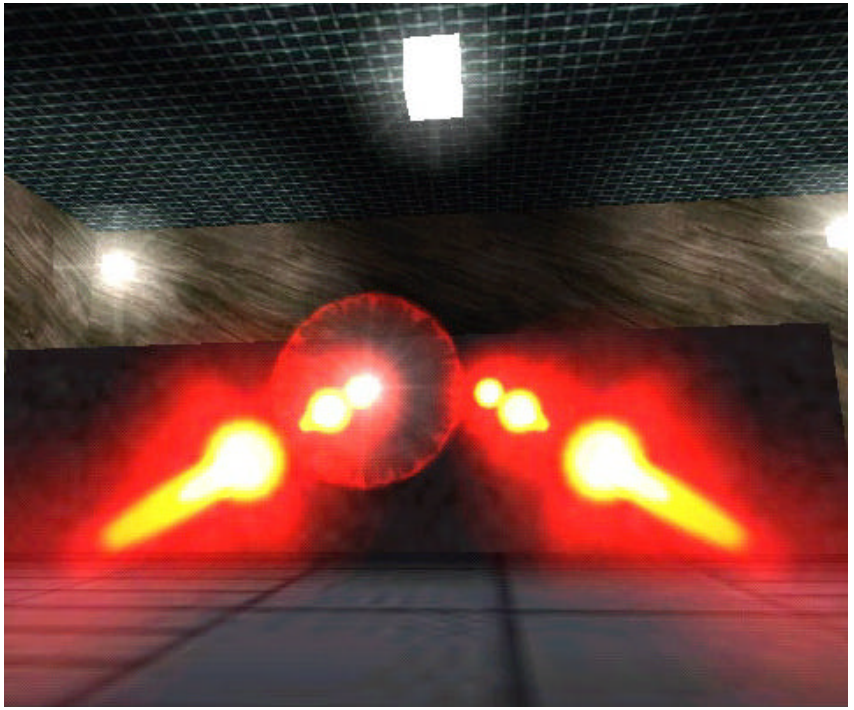
Lens Flare and Bloom - 2

- Use set of textures for glare effects
- Each texture is bill boarded
- Alpha map – how much to blend
- Can be given colors for corona
- Overlap all of them !
- Animate – create sparkle

Example

- Lasers/blasters

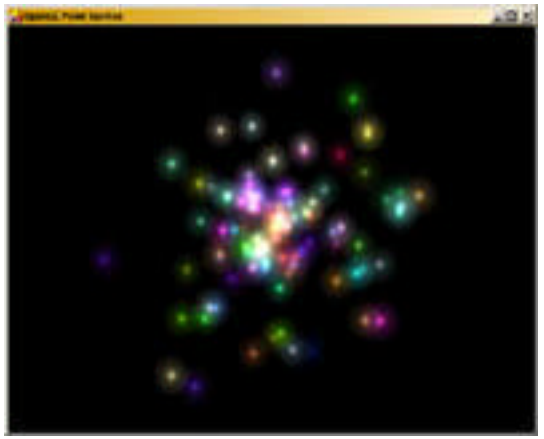




Greg Dunham, 2002

Particle Systems

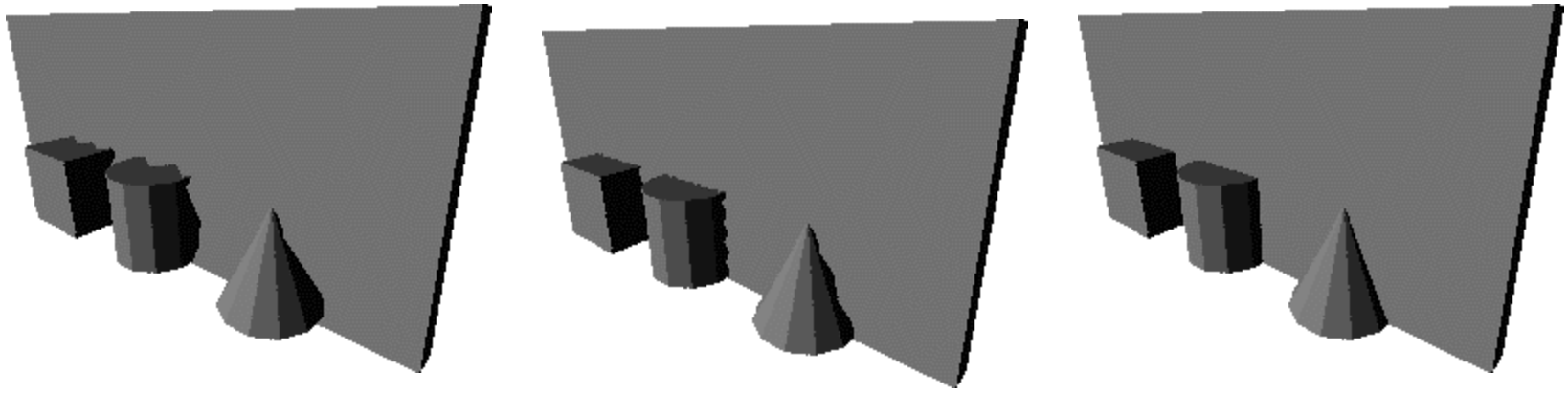
- Set of separate small objects set into motion using an algorithm
- Simulating Fire, smoke, explosions, water flows, trees, galaxies ...
- Method of animation – not rendering
- Representation – Points, lines ...
- Can be billboards too
- DirectX – point sprite primitive
- Example (Gamedev.net)



www.codesampler.com/source

Depth Sprite aka Nailboard

- Give depth to image !
- RGB? - ? is depth parameter
- Depth deviation from the sprite to actual depth of the geometry
- Accuracy varies with number of bits used to represent ?

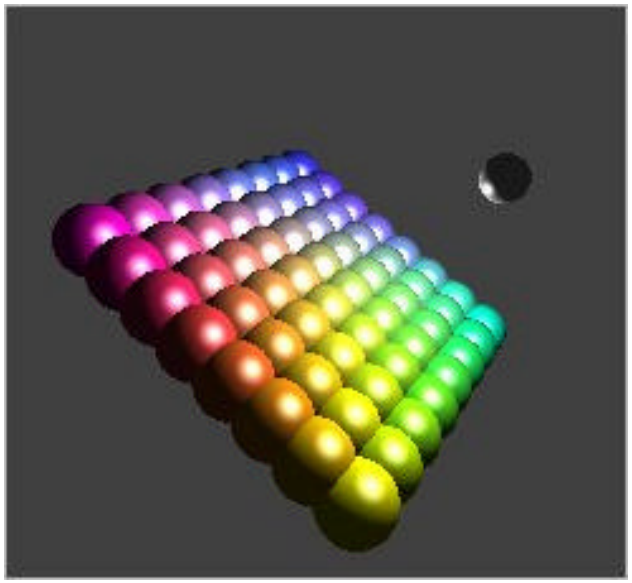


2 bits 4 bits 8 bits

http://zeus.gup.uni-linz.ac.at/~gs/research/nailbord/

Another Example

- Another example for Depth Sprite (NVIDIA)



Hierarchical Image Caching

- Impostors arranged in hierarchy
- Partition scene into boxes – impostor per box
- Parent impostor created from its children
- Minimize dividing plane/object intersections
- Balance the tree

- Walkthrough without Imposters
- Walkthrough with Imposters

http://zeus.gup.uni-linz.ac.at/~gs/research/icache/

Full Screen Billboarding

- Foreground image – goggle view , flash effects etc...
- Background image – environment

Skyboxes

- Environment map of surroundings
- Cube large enough to enclose all objects in the scene
- Far away objects (star fields, sky) - static
- Resolution – texel per screen pixel
- One face covers 90 degrees FOV
- Hide the seams ! (Overlap)
- texres = screenres / tan(fov/2)
- Example

Image Processing

- Render a scene as a texture/image
- Map to a Quadrilateral (texel/pixel)
- Use Pixel Shader to sample it more than once
- Combine !
- Example
  - Blur using a 3x3 grid
  - Nine texture coordinate pairs
  - Each offset by one texel as needed
  - Weigh and sum and output

Image Processing

- Render a scene as a texture/image
- Map to a Quadrilateral (texel/pixel)
- Use Pixel Shader to sample it more than once
- Combine !
- Example
  - Blur using a 3x3 grid
  - Nine texture coordinate pairs
  - Each offset by one texel as needed
  - Weigh and sum and output

Quick Note: Advanced Image Processing with DirectX 9 Pixel Shaders (ATI) – lists Prof. Mike Gennert

Volume Rendering

- Rendering Voxels (CT/MRI)
- Methods
  - Voxel Data is set of 2D image slices (Lacroute & Levoy)
  - Splatting – Voxel represented by alpha blended circular object (splat) , that drops of in opacity at fringes
  - Volume slices as textured Quads (OpenGL Volumizer API)

References

- Chapter 8, *"Real-Time Rendering"*, Second Edition, 2002
  http://www.realtimerendering.com
- Sprite Example
  http://wally.cs.iupui.edu/n341-client/gamelib20/examples/sprite_example.html
- Chicken Crossing
  http://www.glassner.com/andrew/media/chicken/chicken.htm
- IBR Resources
  http://www-2.cs.cmu.edu/%7Eph/869/www/misc.html
- Real Time Cloud Rendering
  http://www.markmark.net/clouds/
- Beam Runner Hyper Game Screenshots
  http://www.cs.unc.edu/~andrewz/twa/screenshots.html
- Impostors Made Easy – William Damon, Intel
  http://www.intel.com/cd/ids/developer/asmo-na/eng/technologies/tools/20219.htm

References

- Text Book Excerpts
  http://www.gamedev.net/reference/articles/article940.asp
- Laser Beams
  http://barney.gonzaga.edu/~gulax/dunham.html
- Particle Systems
  http://www.codesampler.com/source
- Gamedev Particle Systems demo
  http://www.gamedev.net/reference/programming/features/pointspritevb/
- Depth Sprites
  http://zeus.gup.uni-linz.ac.at/~gs/research/nailbord/
- Hierarchical Image Caching
  http://zeus.gup.uni-linz.ac.at/~gs/research/icache/
- Skybox Demo
  http://www.morrowland.com/apron/tut_gl.php
- Image Processing – DirectX Pixel Shaders
  http://www.ati.com/developer/shaderx/ShaderX2_AdvancedImageProcessing.pdf

Conclusion

Questions / Comments / Suggestions