Image-Based Rendering

V1.2

Anthony Steed

Based on slides from Celine Loscos (v1.0)

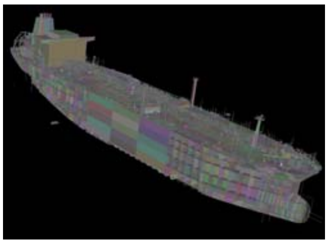
Goals

- Replacing geometry with images
  - For background geometry
  - For individual objects
  - Re-using previous images
- Defining the validity of an IBR
- Updating and replacing image

Overview

1. Motivation & Introduction
   - Examples
   - Classes of image-based rendering
2. Imposters
3. Crowd Models

Motivation



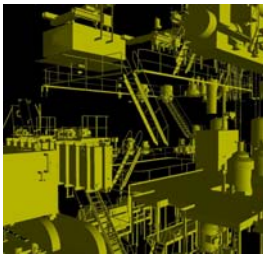
82 Million triangles – 126,000 objects

Modelling complex models require huge amounts of triangles

Conventional polygonal shading is too simplistic. The image doesn't look realistic

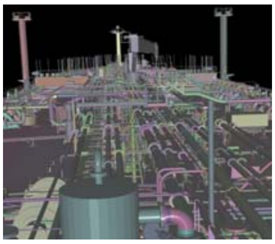
Data usually produced by CAD modelling or 3D scanning: very long and complex process

How many polygons is "enough"?



Distant geometry may even be smaller than a pixel

Millions of polygons to model which details?

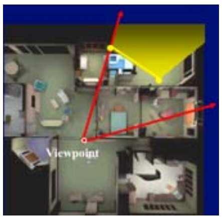


Uses of IBR

- Background or mid-ground geometry
- Individual objects (imposters)
- Re-use of previous frames (post-render warping)

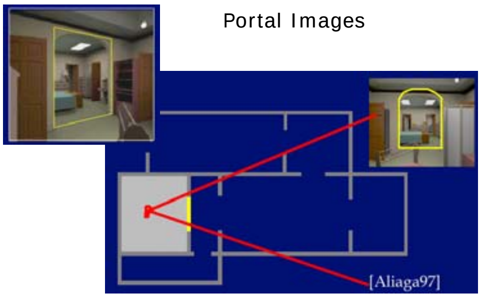
Background Geometry

Architectural walkthrough

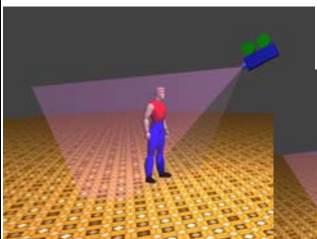


Background Geometry

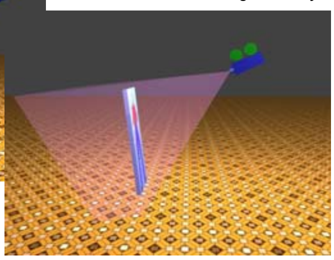
Portal Images



Impostors



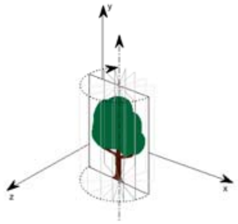
An impostor is an embedded 2D object that replaces only a subset of the scene geometry



The impostor image matches the one of the object from a specific viewpoint

The Simplest Impostor: the Billboard

- An object image is placed on a flat polygon inserted in the scene
- As viewpoint changes the polygon rotates to face the user
- Nice trick to render trees



Post-Render Warping(Mark et al.)

- Render conventional 3D graphics images slowly, on-the-fly
- Apply 3D image warping to generate in-between images quickly
- Use view prediction to guess future view and start rendering conventionally

Good things about IBR

- Model acquisition:
  - Images are relatively easy to acquire
  - Quality can be high and can have good sampling properties for very complex geometry
- Rendering complexity:
  - If you want photo-realistic output, start withphoto-realistic input
  - Dependent on resolution of images and screen, not on 3D geometry
  - Exploit frame coherence

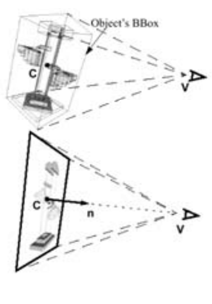
Problems of IBR

- Little hardware support
- It’s hard to have (things that are good about geometry!)
  - Dynamic scenes
  - Scene relighting
  - Depth information
  - Others (Specularity,…)

2. Impostors

- An impostor represents geometry as seen from a single viewpoint
- Due to image coherency, the same image can generally be reused for several frames
- When the viewpoint changes, the impostor image must be updated
- How much can the viewpoint move before we need to update the image?

Impostors



Idea proposed independently by Schaufler and Shade in1996

Algorithm

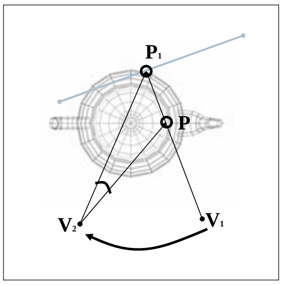
- •Select a subset of the model
- •Create image of the subset
- •Replace subset with image

Advantages

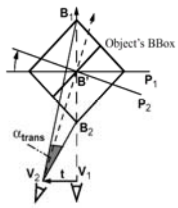
- •Rendering time independent from geometric complexity
- •Exploit rendering coherence: the same image can be used for several frame

First metric- Shade

- Points on the object are projected into the image
- When the viewpoint moves, angular discrepancy in points position appears
- An error angle can be calculated and used to limit the amount of error introduced



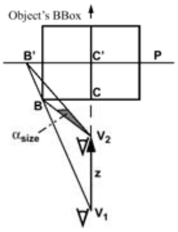
Second Metric –Schaufler(I)



Consideration of two worst case:

1) Angular discrepancy due to translation of the viewpoint parallel to the impostor plane

Second Metric –Schaufler(II)



2) Viewpoint moving towards the object

USE IMPOSTOR if:

$$(\alpha_{trans} < \alpha_{screen}) \text{ and } (\alpha_{size} < \alpha_{screen})$$

where $\alpha_{screen} = \dfrac{\text{field of view}}{\text{screen resolution}}$

(α_{screen} is the angle subtended by a pixel at the viewpoint)

How to improve validity?

Artefacts arise due to the planar nature of the impostor.No motion parallax

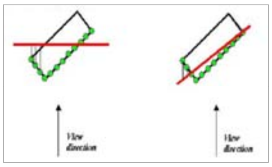
To reduce artefacts we can store more information about the impostor (depth, multiple layers...)

Regeneration of the impostor image from a previous one: Image Warping

Choosing the best impostor plane

Choosing the best impostor plane

- **Errors proportional to the distance from the projection plane**
- **Best impostor plane orientation depends on the sample**

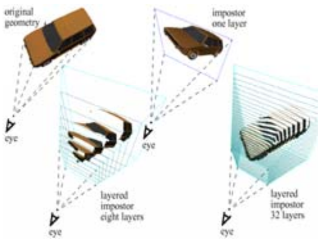


3D warping using depth information

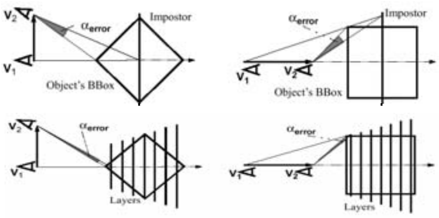
- a depth value is associated to each pixel
- 3D warping is possible
- Holes appear were data is missing. Can be attenuated warping multiple images or using interpolation
- No hardware support on conventional graphics pipelines

Hardware assisted image warping

- Images stored as RGBA texture
- Alpha channel store the object's depth map (8BIT)
- Using Alpha Testing we can select different "slices" of the impostor
- Layers are rendered one in front to the other, to approximate the original depth of the object

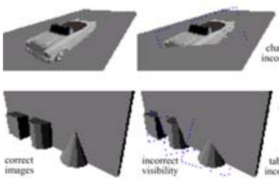


Single VS Layered



- Layered impostors are a better approximation
- "Hardware accelerated"
- Fill-rate expensive
- Number of layers used may be varied with the distance

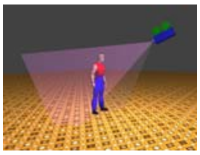
Limits of the Impostors

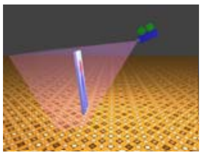


Incorrect visibility due to the lack of depth information

Warping a singe image can produce "holes" where data is missing

3. Crowd using impostors

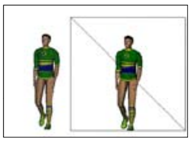


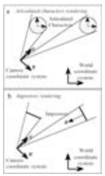


Not really conventional impostors:

- **Replaced geometry is animated!**
- **>10,000 independent impostors**

Computing impostors on the fly





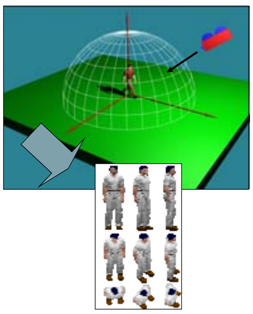
Aubel,Boulic,Thalmann (1998)

Texture as a cache memory

Single impostor:
multiple resolutions used
(128x128->32x32)

Multiple impostors:
Trying to reduce redrawing at a minimum

Precomputing impostors



- **A discrete set of images are taken from around the 3D models (32x8) and for each frame of animation**
- **At run time for each avatar the best sample extracted and projected on impostor plane**

Static or Dynamic impostors?

Pros / **Cons**

Static

Pros:
- Very fast
- Hard-core optimizations at preprocessing
- No 2-ways BUS traffic

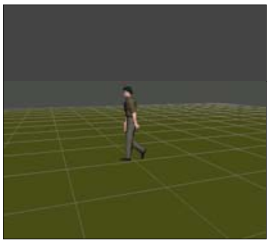
Cons:
- Need a lot of texture memory

Dynamic

Pros:
- Texture memory acts just as a (smaller) cache
- No preprocessing
- Transparent to the artist

Cons:
- High BUS traffic
- No much time for “clever tricks”
- Slower

How many samples are “enough”?



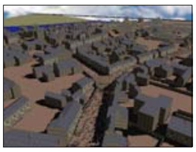
A practical example:

- 32x8 samples
- Average sample size 128*32 pixels
- 18 frames of animation
- With memory management & texture compression 256k/frame in texture memory

“Impostors are unflexible”

- Using multi-pass rendering to control impostor colors
- RGB stores shading info only
- Alpha testing used to select single sub-regions
- 16 independent regions with texture compression

Real-time crowd rendering



10,000 Impostor@20Hz PentiumIII-800Mhz
NVIDIA GeForce 2 32Mb

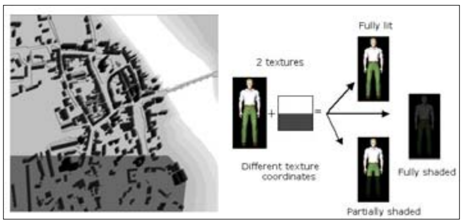
Tecchia, Chrysanthou, Loscos (2001)

Impostor Shading

Modulating ambient lighting on each impostor can improve realism

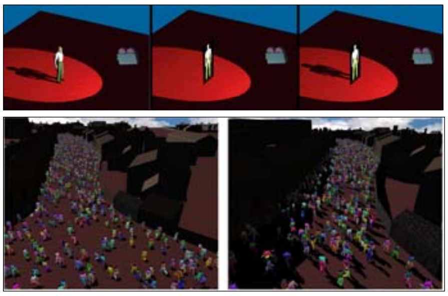
Shadows

Even more tricks are possible with shadow volumes and multi-texture



Loscos, Tecchia, Chrysanthou, (2001)

Impostors (fake) shadow



Impostors & shadowmaps

•Shadow-maps can be used to cast impostors shadows onto the environment

•Only perspective-correct shadow-maps really suitable

•Only one pass for shadow-map computation

•NO self shadowing

Conclusions

- Image-based rendering has some definite uses
  - Replacing backgrounds
  - Providing very dense changing models
- IBR exploits image coherency between frames
- However, introduces artefacts and, as other acceleration techniques, needs careful use in real situations