GigaVoxels: Voxels Come Into Play

Cyril Crassin

To cite this version:

HAL Id: inria-00526646
https://inria.hal.science/inria-00526646
Submitted on 15 Nov 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
GIGAVOXELS: Voxels Come Into Play

Cyril Crassin, Fabrice Neyret, Sylvain Lefebvre, Elmar Eisemann, Miguel Sainz

INRIA Rhône-Alpes & Grenoble Univ., INRIA Sophia-Antipolis, Saarland Univ./MPI, NVIDIA Corporation
A (very) brief history of voxels

- Rings a bell?

Comanche (Novalogic)

Outcast (Appeal Software)

Voxel grid illustration courtesy of "Real-Time Volume Graphics"
Voxel Engines in Special effects

- Natural representation
  - Fluid, smoke, scans, ...
- Volumetric phenomena
  - Semi-transparency
- Unified rendering representation
  - Particles, meshes, fluids...

XXX, Digital Domain

The Day After Tomorrow, Digital Domain

Lord of the Rings, Digital Domain
Voxels in video games?

- **Renewed interest**
  - ID Software
    - John Carmack, Jon Olick (Siggraph 08)
    - Sparse Voxel Octree ray-casting
  - Crytek
    - Cevat Yerli
  - ...

- **Two goals:**
  - Content generation
  - Rendering
Why bother with voxels?

- Exploding number of triangles
  - Costly to transform & rasterize
  - Inefficient raster of small triangles on current generation GPUs

- Geometric LOD ill-defined
  - Eg. Progressive Meshes
  - Lot of manual intervention for the artist
Why bother with voxels?

- Filtering is an issue
  - Needs massive multi-sampling
  - Multi-sampling is expensive

*The Mummy 3, Digital Domain/Rhythm&Hues*
Why bother with voxels?

- Unified Geometry + Texture representation
- Avg space occupancy/density information
- Avg color information
Why bother with voxels?

- Filtering is well defined
  - LOD = Mip-Mapping
    - Similarly to 2D textures

- Unique multi-scale representation
  - No additional authoring

- Structured representation
  - Convenient to traverse & edit
  - Efficient to render
    - -> Ray-casting
How to exploit them?

Main problems:

- How to render voxels quickly on the GPU?
  - How to exploit these properties?
- Memory is a key issue!
  - E.g. $4096^3 \times RGBA8 = 256 \text{ GB!!!}$
  - Transfer CPU $\leftrightarrow$ GPU expensive
GigaVoxels

- **Goal:** **Real-time** exploration of very large voxel scenes

- Full GPU rendering pipeline
  - Ray-tracing based approach
  - Fully scalable: **Infinite resolution**

- **Publications:**
  - I3D2009 paper [CNLE09]
  - Siggraph 2009 Talk
  - GPU Pro (ShaderX 8) Book Chapter
Key ideas

- Rendering only dependent on what is visible
  - Ray-tracing approach

- Load only needed data, at the needed resolution
  - Occlusion + LOD
  - Ray-guided streaming

- Reuse loaded data as much as possible
  - GPU cache mechanism
GigaVoxels CUDA pipeline

**GPU**
- Sparse Voxel Octree
- Voxels Ray-Tracer
- Data usage + requests
- Structure updates
- GPU Cache Manager

**CPU**
- Mesh data
- GPU Producer
- Procedural data
- Mass storage
Voxel sculpting

- Direct voxel sculpting
  - 3D-Coat
    - Like ZBrush
- Generate a lot of details
  - 5-20 FPS
Data Structure

GPU

Voxel Ray-Tracer

Output image

Data usage+requests

Sparse Voxel Octree

GPU Cache Manager

Voxel data fetch

Mass storage

CPU Data Store

CPU
Sparse Voxel MipMap Pyramid

Data structure

**Generalized Octree**
- Empty space compaction

**Bricks of voxels**
- Linked by octree nodes
- Store opacity, color, normal,…

Tower model courtesy of Erklaerbar, made with 3DCoat
Octree of Voxel Bricks

- One child pointer
- Compact structure
- Cache efficient
Rendering

GPU

Voxel Ray-Tracer

Output image

Data usage+requests

Structure updates

Sparse Voxel Octree

GPU Cache Manager

On-demand Data Producer

CPU Data Store

On-demand Data Producer

Voxel data fetch

Voxel data fetch

CPU

On-demand Data Producer

GPU Cache Manager

Output image

Data usage+requests

Structure updates

Sparse Voxel Octree

Voxel Ray-Tracer
Hierarchical Volume Ray-Casting

- Render semi-transparent materials
  - Participating medias
- Emission/Absorption model for each ray
  - Accumulate Color intensity + Alpha
  - Front-to-back
    - Stop when opaque
Hierarchical Volume Ray-Casting

- Volume ray-casting
  [Sch05, CB04, LHN05a, Olick08, GMAG08, CNLE09]

- One big CUDA kernel
  - One thread per ray

- Octree traversal
  - KD-restart algorithm [FS05]
  - Ray-driven LOD

- Bricks marching
  - Regular sampling into the 3D texture
Volume Ray-Casting

Tree Descent

1

2

3

Skip Node

4

Brick Marching

5

6

7

8

9

Brick Marching

Ray traversal

Per-ray LOD evaluation
Rendering costs
Volume MipMapping mechanism

Problem: LOD uses discrete downsampled levels

- Popping + Aliasing
- Same as bilinear only for 2D textures

- Geometry is texture 😊
  - Uses pre-integrated LOD !
  - No need of multi-sampling (eg. MSAA)
Cone tracing

- Camera
- Rendered image
- Ray source pixel
- Ray cone footprint
- Interpolated voxel samples

Sample footprint
- Sample footprint
- Linearly interpolated sample

Integrated pixel
- Color + Alpha

Full MipMap pyramid
- S0
- S1
- S2
- S3
Shading computation

- Standard Blinn-Phong illumination
  - Per sample

- Normal information
  - On-the-fly gradient with finite differences
  - Stored normal information

- Deferred for opaque objects
Data Management

GPU

Sparse Voxel Octree

Voxel Ray-Tracer

Output image

Data usage+requests

Structure updates

GPU Cache Manager

GPU Producer

CPU Data Store

CPU Data Store
GPU Caches

- Data management made through a cache mechanism
  - Used for both the **node pool** and **brick pool**
  - Allows full scalability

- Rely on the octree to address elements
  - The node pool is addressing itself!
  - No page table

- Data requests generated by the ray-tracing
  - Node subdivision
  - Brick loading
Incremental octree update

- Progressive loading

Pass 1

Pass 2

Pass 4

Wrong LOD

Wrong LOD

Data request

Data requests

Data request

Data request

(LoD OK)

(Constant value)

(Max opacity)

(Node not reached)

No Data

Node pool

Brick pool

1 2 3 4 5
Ray-based visibility & requests

- Minimum amount of data is loaded
- Progressive refinement
  - Always ensure interactivity
- Fully compatible with secondary rays and exotic rays paths
  - Reflections, refractions, shadows, curved rays, …
Cache requests handling

- Entirely handled on the GPU
Cache strategy

- Least Recently Used (LRU) strategy
  - Older elements replaced first

- Sorted usage list maintained for each cache on the GPU.

- Usage info provided by the ray-tracer

- Maintained as a data-parallel process

- Used when new elements have to be inserted
SVMP caches

- LRU (Least Recently Used)
  - Track elements usage
  - Maintain list with least used in front

Cache Elements (*Node Tile/Brick*)

- Usage sorted nodes addresses
  - Oldest
  - Newest
- New elements
- New data
- Stream compaction
- Concatenate

Octree/Bricks Pool
Global cache characteristics

- Driven by ray-tracing

- Fully managed on the GPU
  - Zero CPU intervention apart kernel launches.
  - Leads to fully on-chip structure management and building

- More efficient when large amount of updates
APPLICATIONS
Voxel data synthesis

- Instantiation

- Recursivity
  - Infinite details
Free voxel objects instancing

- BVH structure ray-casting
  - Cooperative ray packet traversal [GPSS07]
  - Shared stack
- WA-Buffer
  - Deferred compositing
Procedural noise

- On-the-fly mesh voxelization
  - Distance field

- Procedural noise
Cool Blurry Effects

- Going further with 3D MipMapping
  - Full pre-integrated versions of objects

- Idea: Implements blurry effects very efficiently
  - Without multi-sampling
  - Tuning the mipmap level

- Soft shadows
- Depth of field
- Glossy reflections…
Soft shadows

- Secondary rays
  - When ray hit object surface
- MipMap level chosen to approximate light source cone
  - Resulting integrated opacity
- Fully compatible with the cache
Depth-Of-Field

- Similarly for depth-of-field...
  - MipMap leveld based on circle-of-confusion size

Illustration courtesy of GPU Gems
Ambient occlusion

- Uses one filtered sample
  - Covers the surrounding region

*Without AO*

*With AO*
Future work direction

- **Animation**
  - Yes, this can be efficiently animated!
  - Volume deformation (skinning)

- **Improved visibility integration**

- **Filtering**
  - Shading/Normals
  - Isotropic pre-integration
    - Two walls problems
Many thanks go to ...

- Digisens Corporation
- Rhone-Alpes Explora’doc program
- Cluster of Excellence on Multimodal Computing and Interaction (M2CI)
- 3D-Coat and Rick Sarasin
- Erklaerbar
Any questions?

THANK YOU FOR YOUR ATTENTION