GigaVoxels: Voxels Come Into Play
Cyril Crassin

To cite this version:
inria-00526646

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

Crytek Conference 26/11/09

Cyril Crassin, Fabrice Neyret,
INRIA Rhône-Alpes & Grenoble Univ.

Sylvain Lefebvre, Elmar Eisemann, Miguel Sainz
INRIA Sophia-Antipolis Saarland Univ./MPI NVIDIA Corporation
A (very) brief history of voxels

- Rings a bell?

Comanche (Novalogic)

Outcast (Appeal Software)
Voxel Engines in Special effects

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

The Day After Tomorrow, Digital Domain

XXX, 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 dependant 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
Voxel sculpting

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

Graphic illustration showing:
- Node pool
- Linear Memory
- CUDA 3D Array (Texture)

Brick pool

GPU
Rendering

GPU

Sparse Voxel Octree

Voxel Ray-Tracer

Output image

Data usage+requests

Structure updates

GPU Cache Manager

Voxel data fetch

CPU

On-demand Data Producer

CPU Data Store
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
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

Ray source pixel

Ray cone footprint

Interpolated voxel samples

Camera

Rendered image

Sample footprint

Integrated pixel Color+Alpha

Full Mipmap pyramid

S0

S1

S2

S3

S3_1

S3_0

S2_0

S1_1

S1_0

Image plane

One pixel footprint

Linearly interpolated sample
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

Sparse Voxel Octree -> Voxel Ray-Tracer

Voxel Ray-Tracer -> GPU Cache Manager

GPU Cache Manager -> CPU Data Store

Data usage+requests

Structure updates

Output image
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

Data request

No Data

Data requests

(Constant value)

(Max opacity)

(Node not reached)

Node pool

Brick pool

1 2 3 4 5

Pass 1

Wrong LOD

Pass 2

Data request

Pass 4

(Node not reached)
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

Voxel Ray-Tracer
- Request batch

Node Pool
- Nodes write

Cache Mgr
- Voxel bricks write

Brick Pool

Data Provider
- Procedural Generation
- Zero-copy fetch

Request handler

CPU Data Store
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

Octree/Bricks Pool

Cache Elements *(Node Tile/Brick)*

Usage sorted nodes addresses

New elements mask

Stream compaction

Concatenate

New data
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
Voxels generation

- Voxels generation
- GPU
- Sparse Voxel Octree
- Voxel Ray-Tracer
- Output image
- Data usage+requests
- Structure updates
- GPU Cache Manager
- Procedural data
- Mesh data
- GPU Producer
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
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
THANK YOU
FOR YOUR ATTENTION