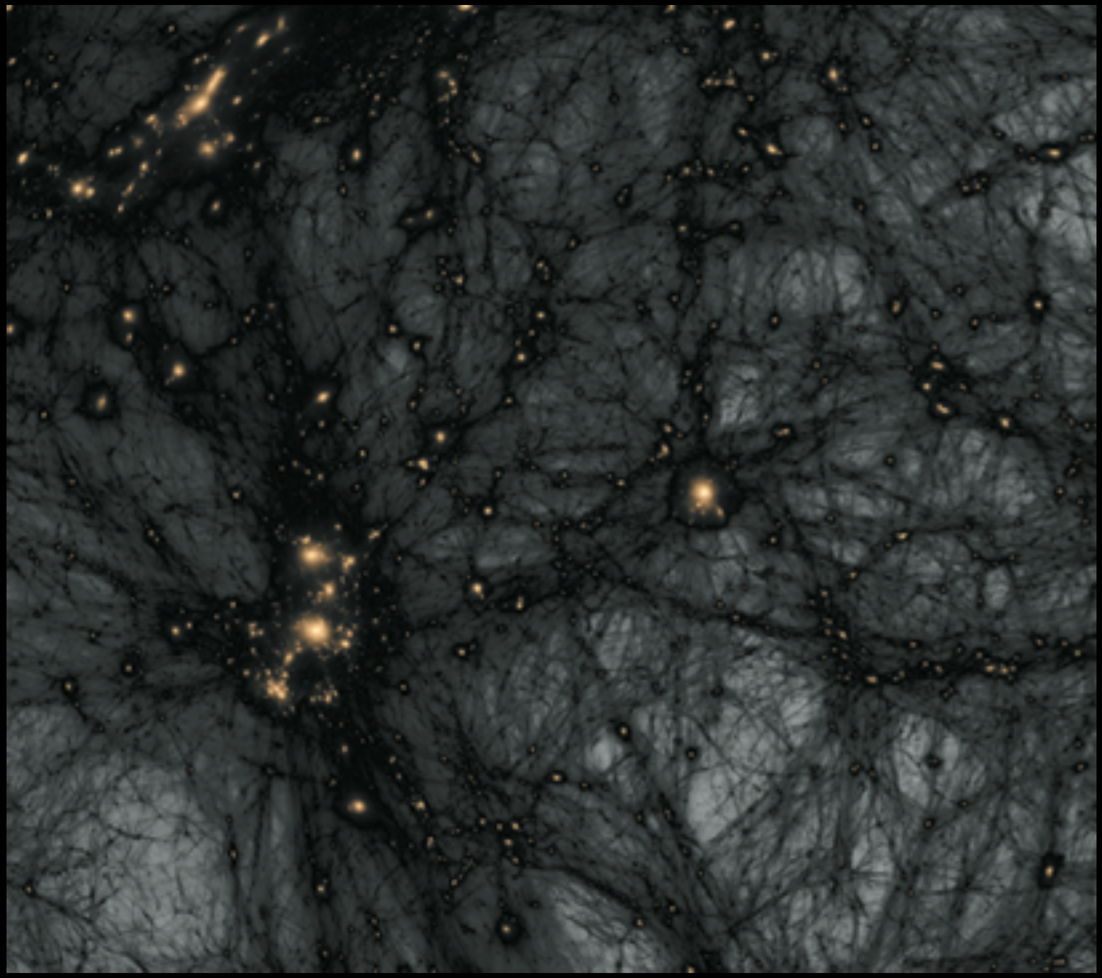
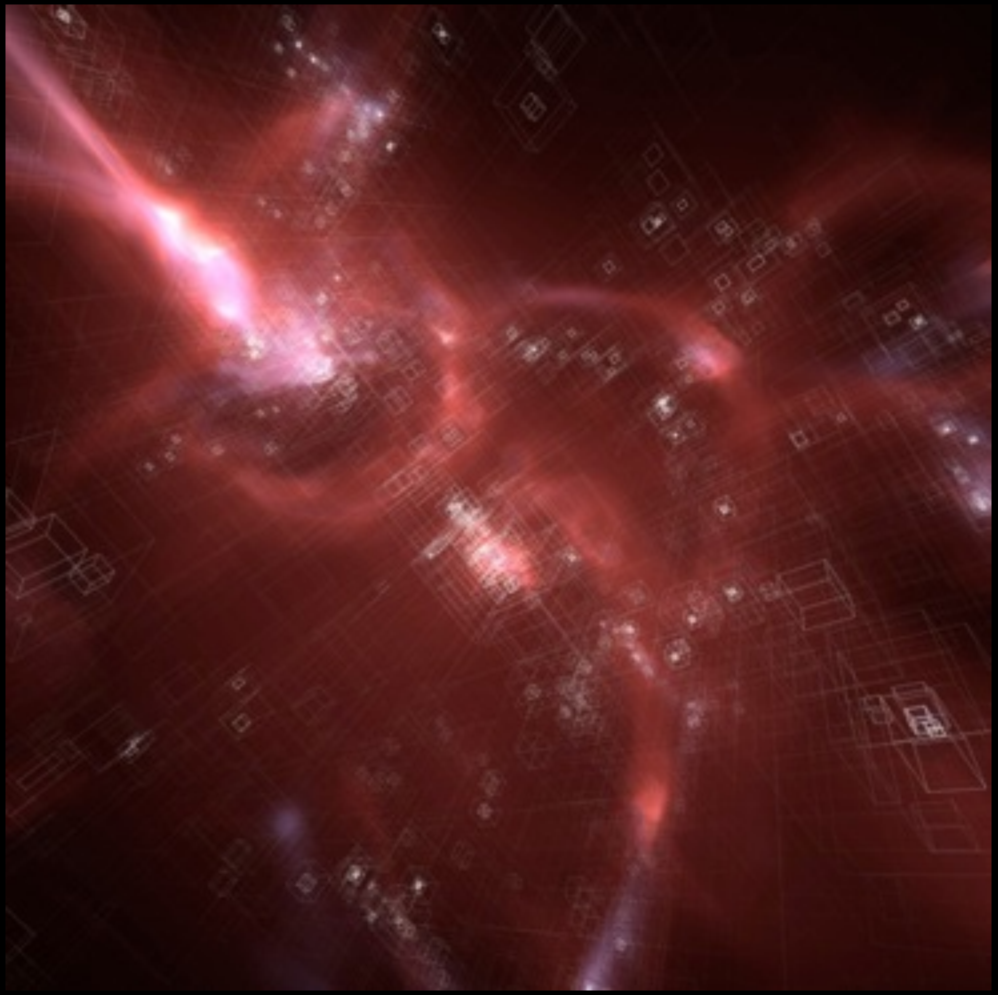
GPU-Based Visualization of AMR and N-Body Dark Matter Simulation Data





Ralf Kähler
(KIPAC/SLAC)

SLAC

HiPACC-Meeting 03/21/2014

KIPAC

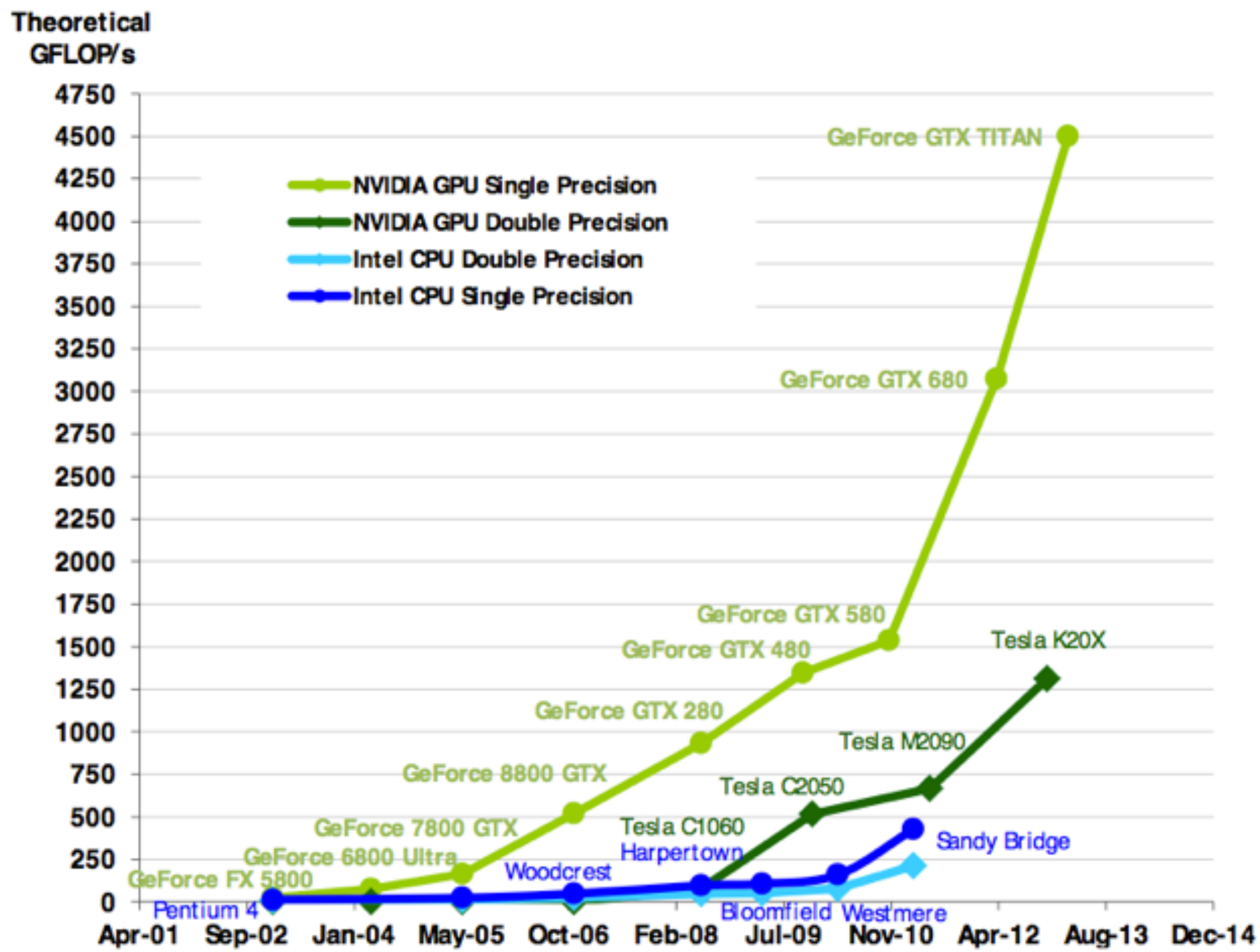


Figure 1 Floating-Point Operations per Second for the CPU and GPU

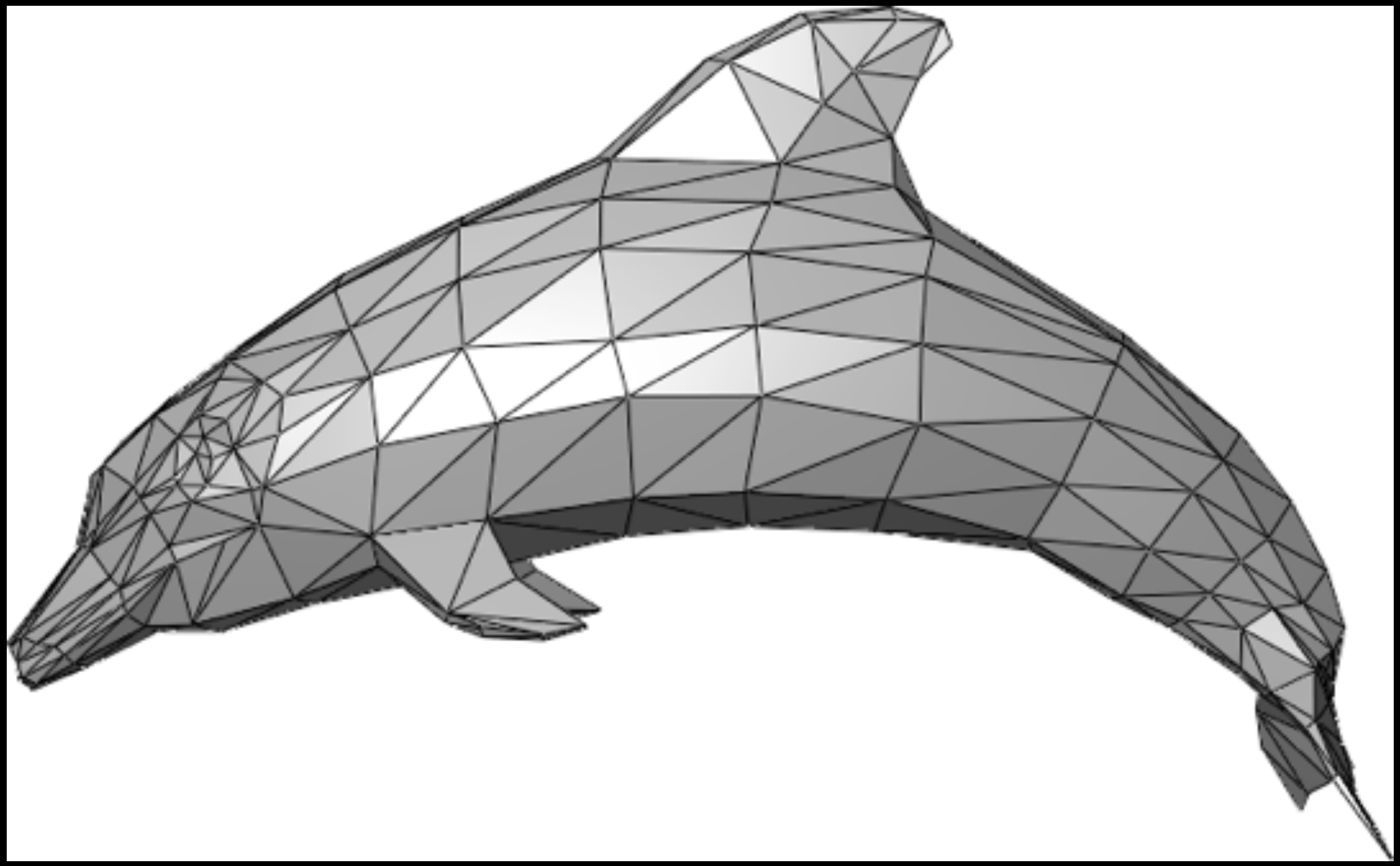
www.nvidia.com
CUDA C Programming Guide

Rasterization

COMPUTER GRAPHICS

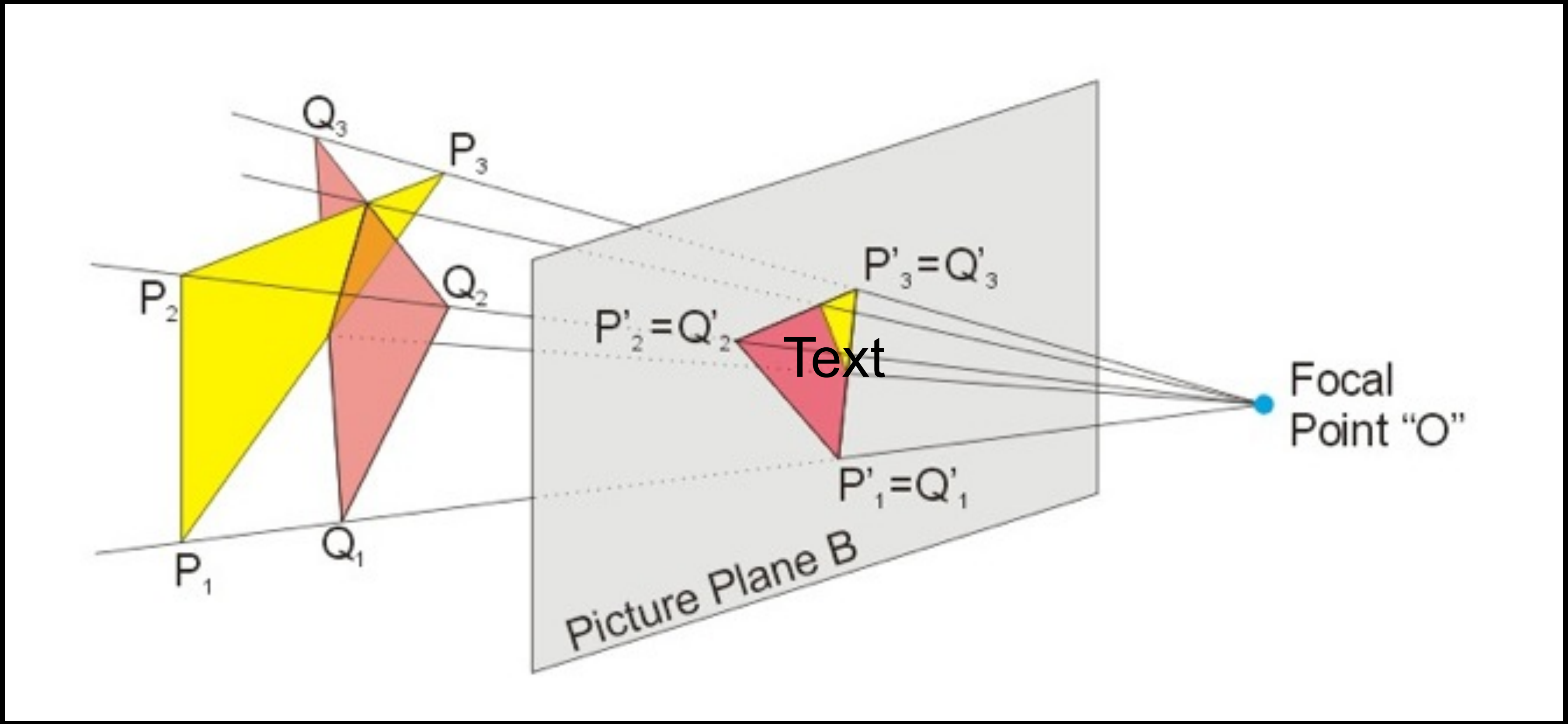
Assumption (for now):

Input object(s) represented as triangulated mesh



http://en.wikipedia.org/wiki/File:Dolphin_triangle_mesh.png

RASTERIZATION-BASED GRAPHICS



http://en.wikipedia.org/wiki/File:Perspective_Projection_Principle.jpg

Graphics Pipeline for Rasterization

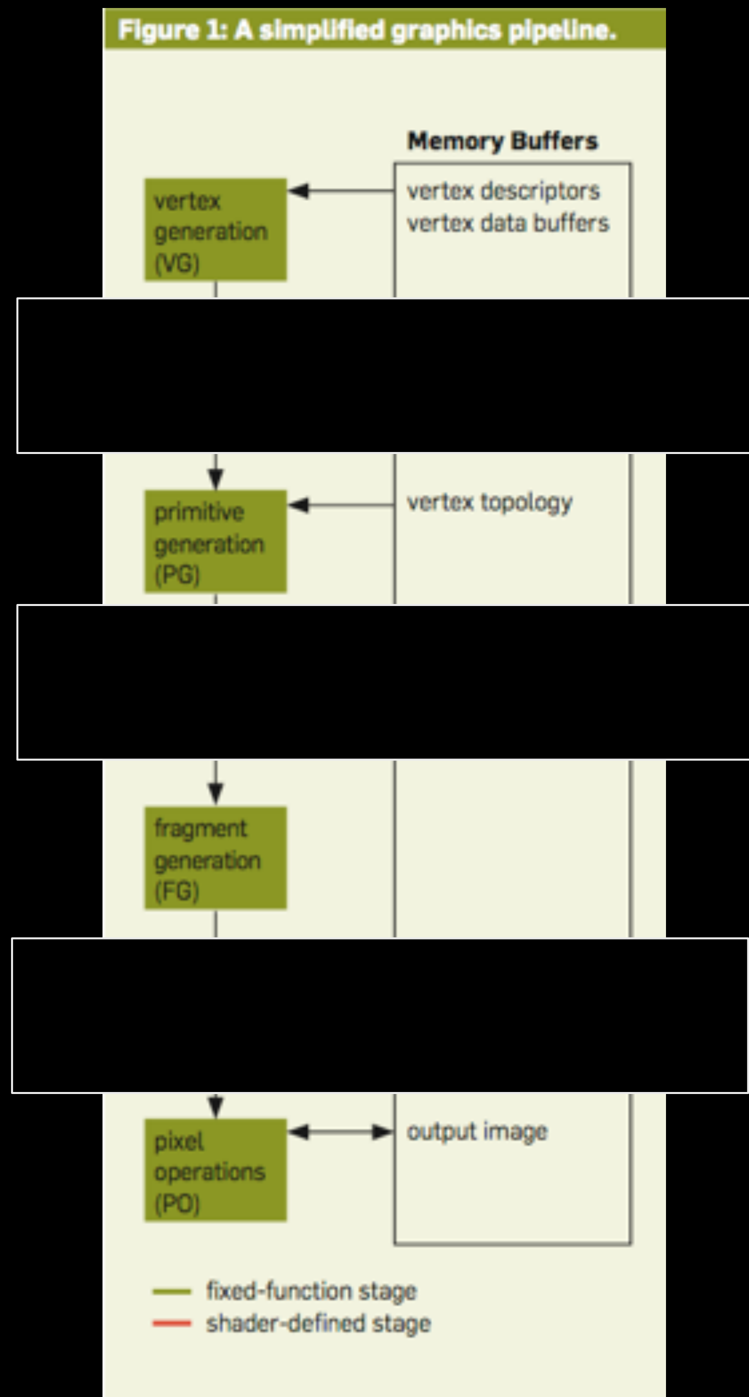


Image from: FATAHALIAN, "A closer look at GPUs"

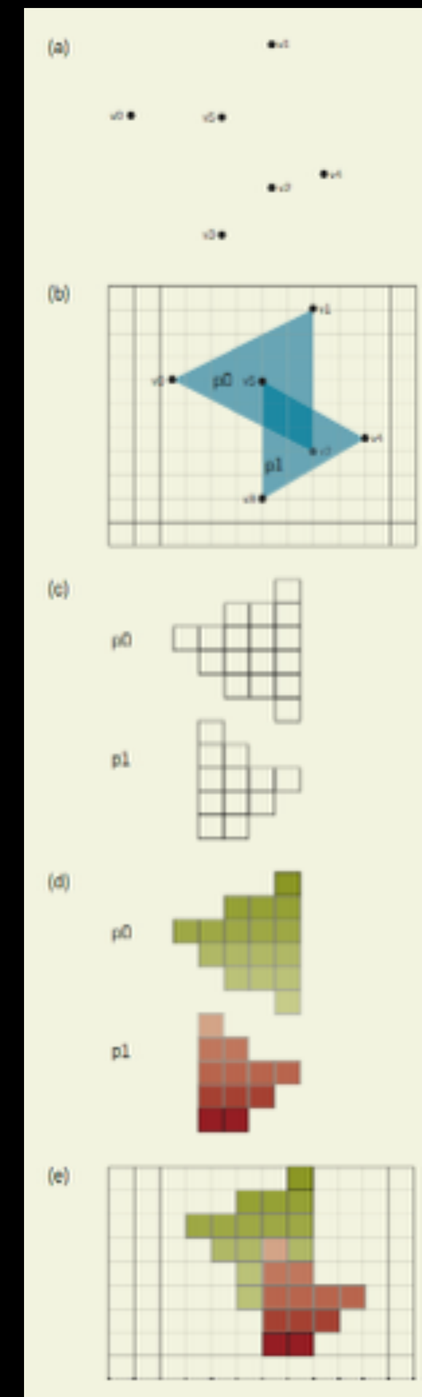


Image from: FATAHALIAN, "A closer look at GPUs"

Graphics Pipeline for Rasterization

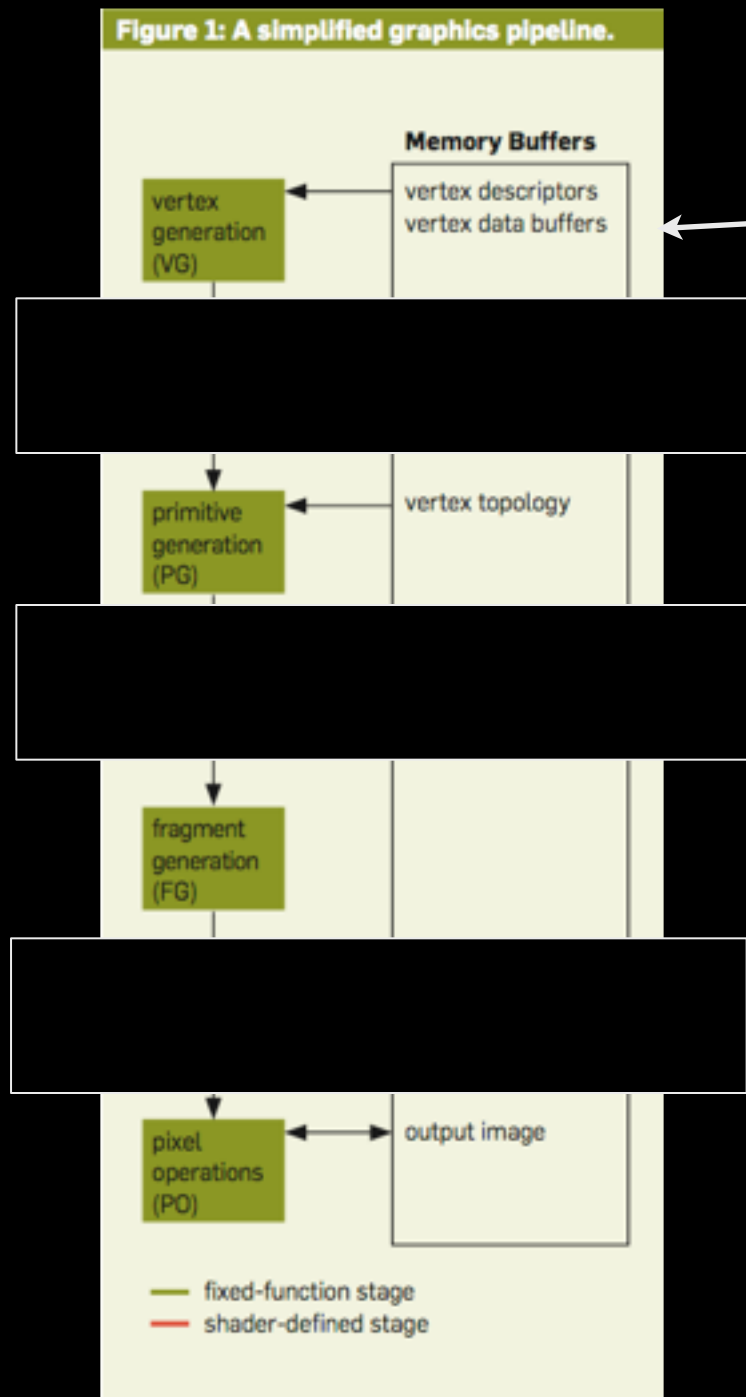


Image from: FATAHALIAN, “A closer look at GPUs”

Input mesh topology

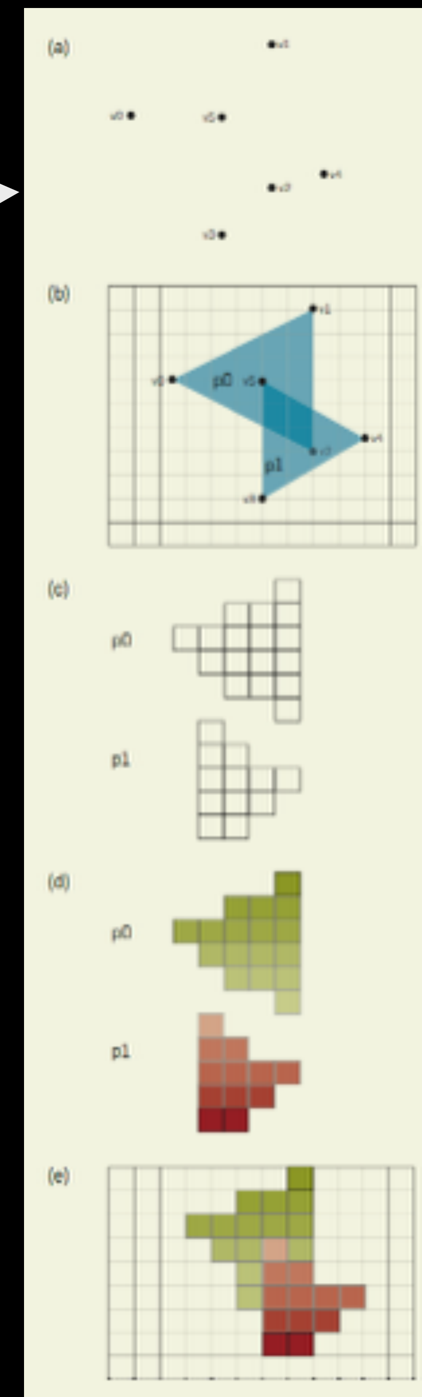


Image from: FATAHALIAN, “A closer look at GPUs”

Graphics Pipeline for Rasterization

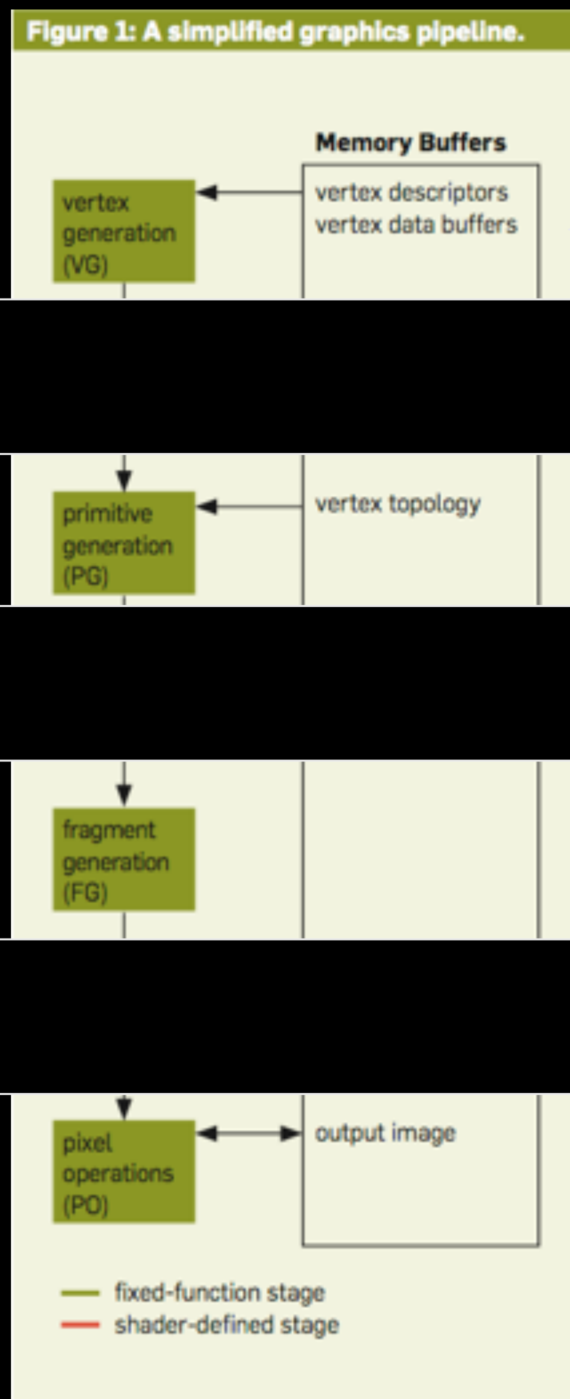


Image from: FATAHALIAN, "A closer look at GPUs"

Input mesh topology

Triangle Assembly
& Projection

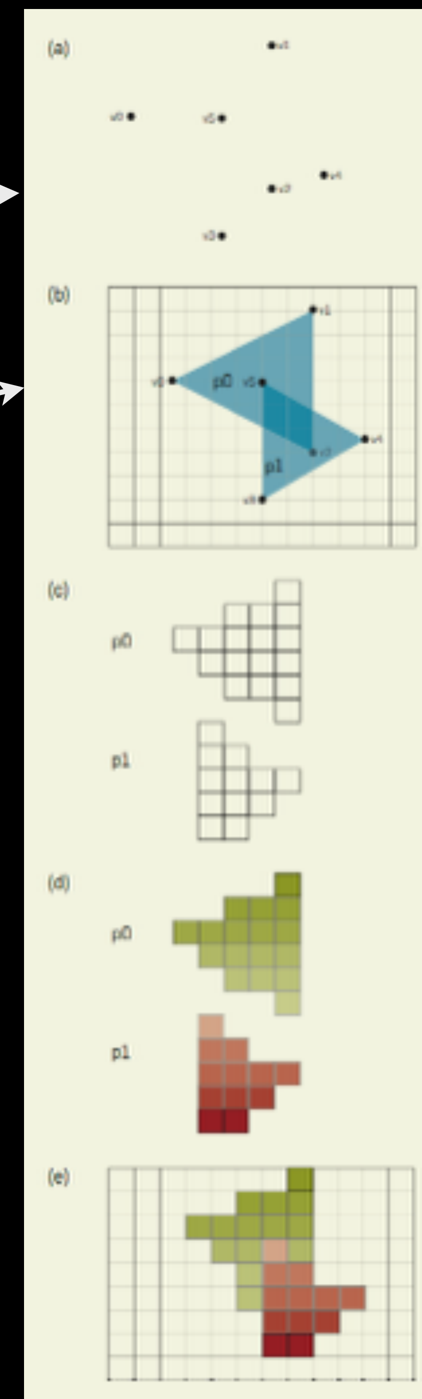


Image from: FATAHALIAN, "A closer look at GPUs"

Graphics Pipeline for Rasterization

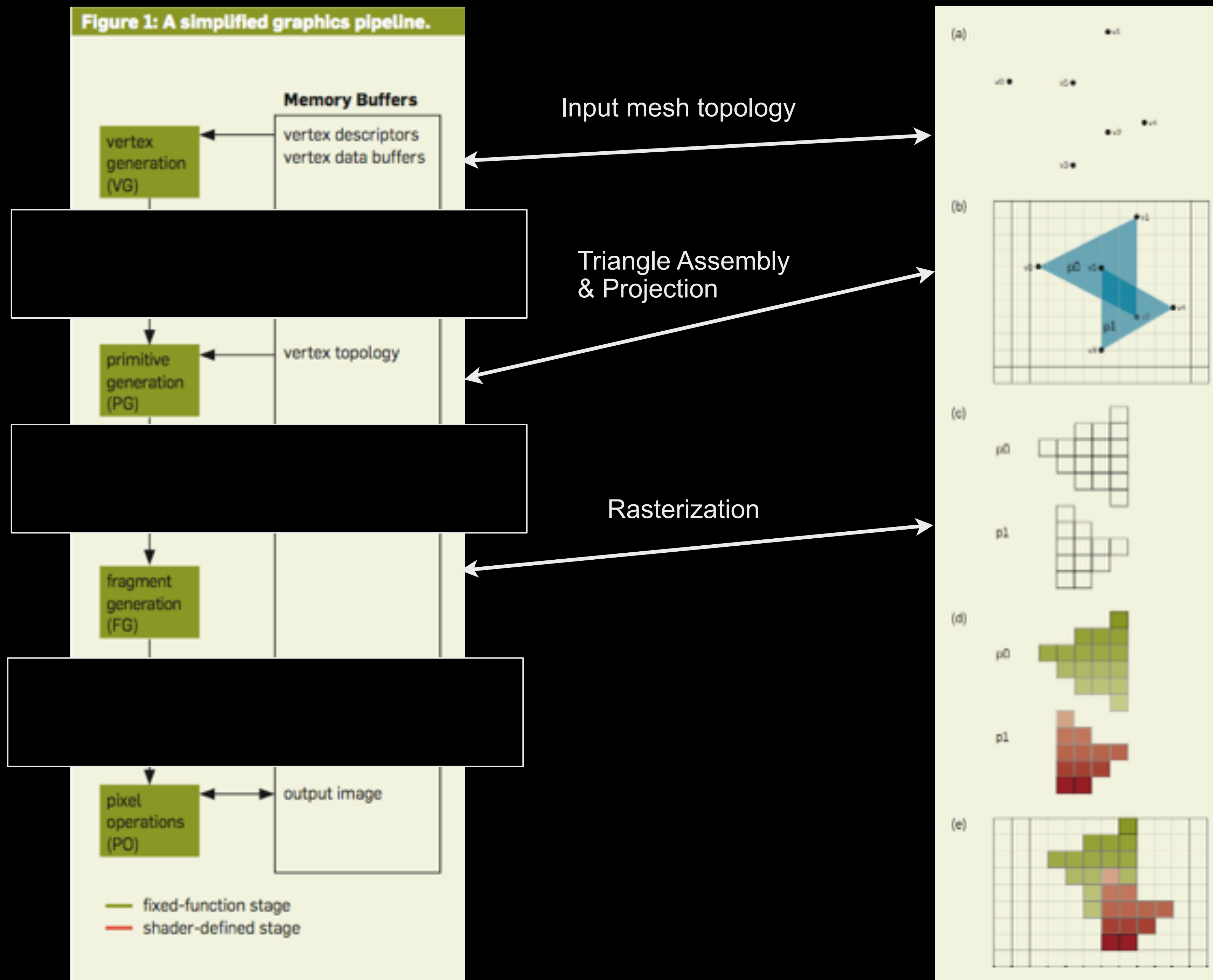


Image from: FATAHALIAN, “A closer look at GPUs”

Image from: FATAHALIAN, “A closer look at GPUs”

Graphics Pipeline for Rasterization

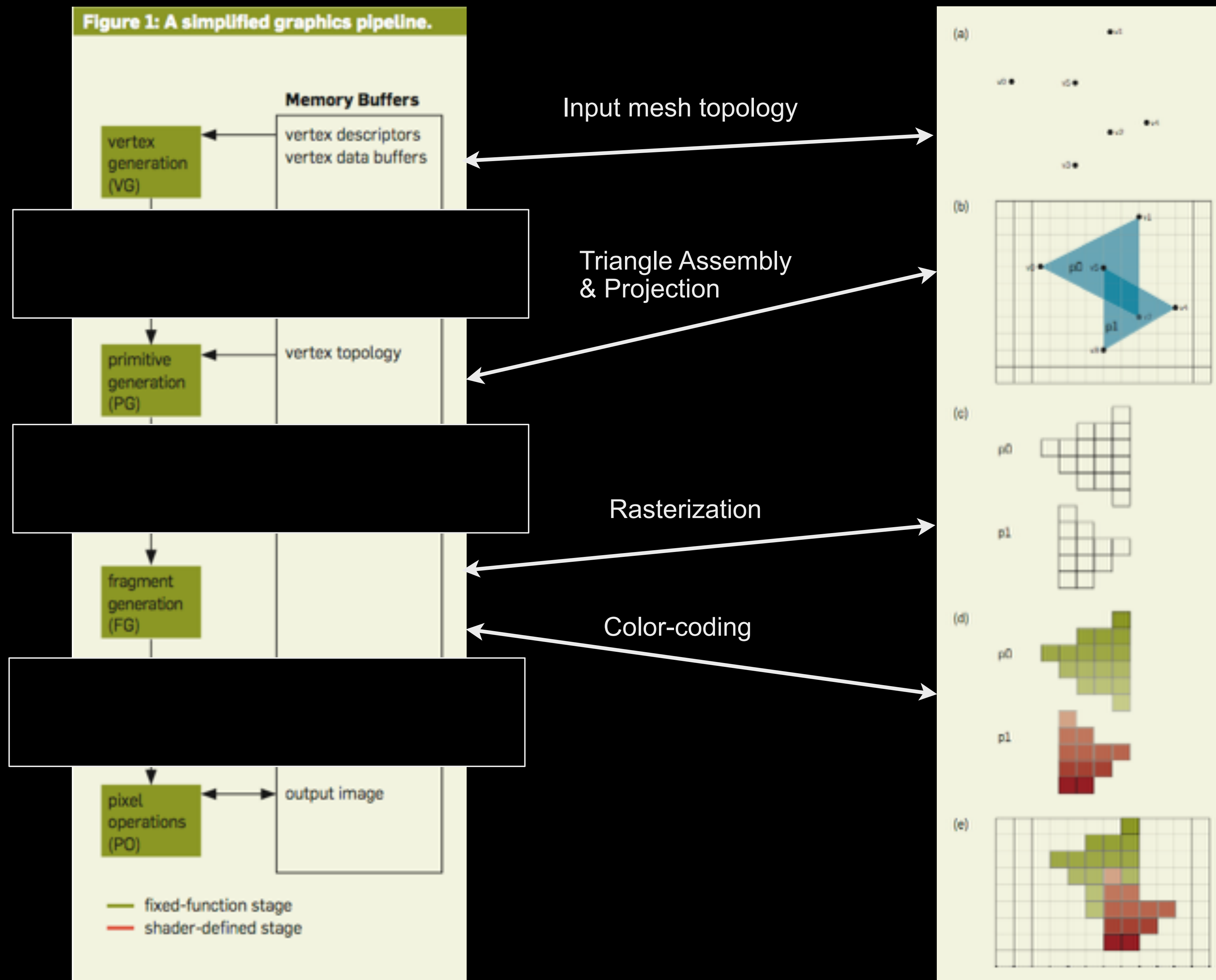


Image from: FATAHALIAN, “A closer look at GPUs”

Image from: FATAHALIAN, “A closer look at GPUs”

Graphics Pipeline for Rasterization

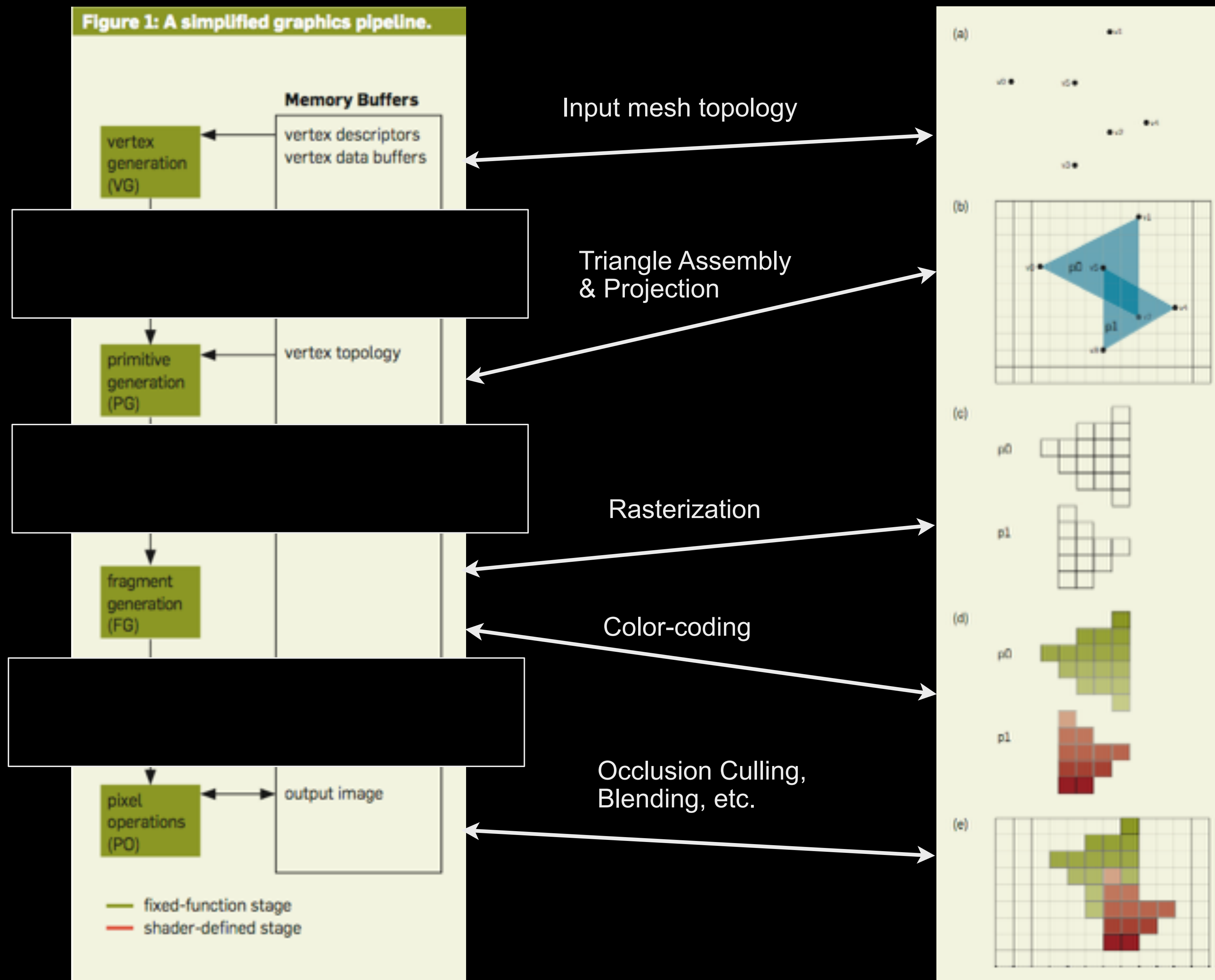


Image from: FATAHALIAN, "A closer look at GPUs"

Image from: FATAHALIAN, "A closer look at GPUs"

Programmable Graphics Pipeline

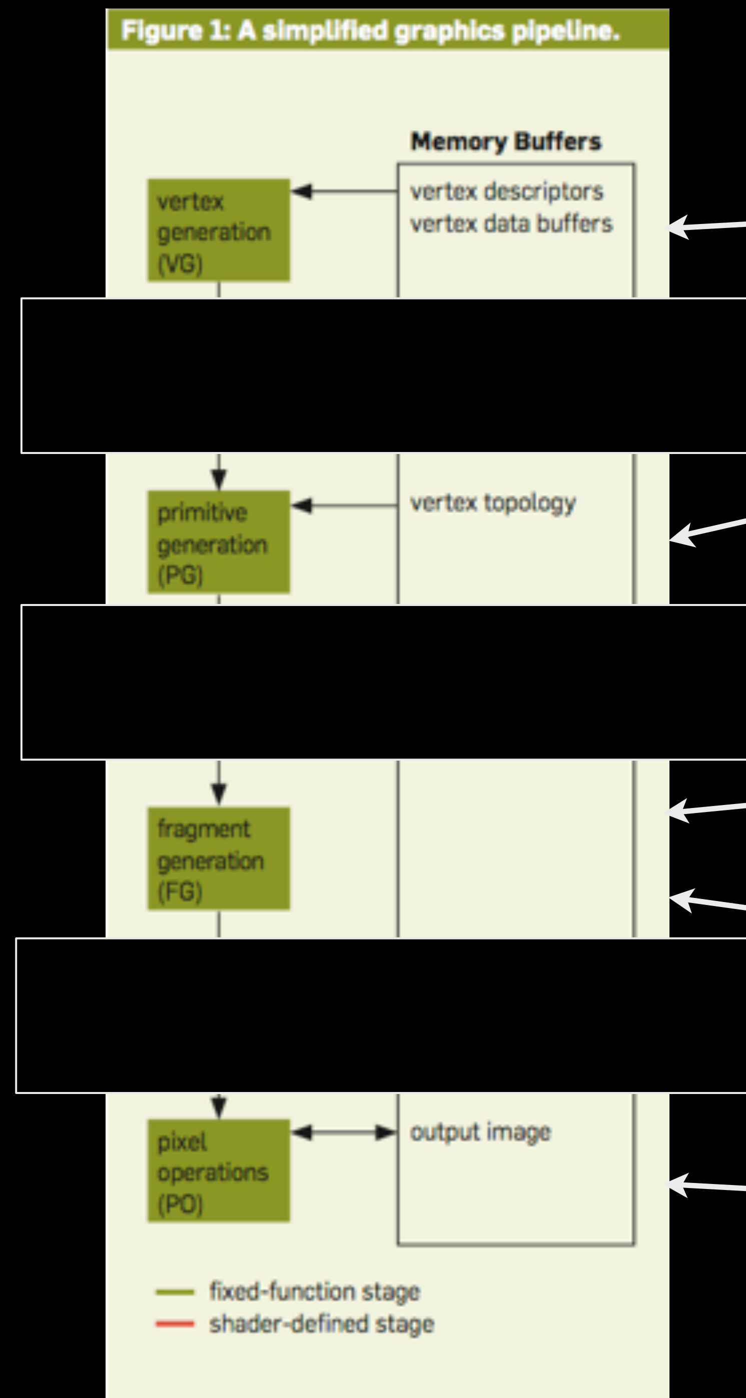


Image from: FATAHALIAN, "A closer look at GPUs"

Input mesh topology

Triangle Assembly & Projection

Pixel coverage

Texture lookup

Occlusion Culling, Blending, etc.

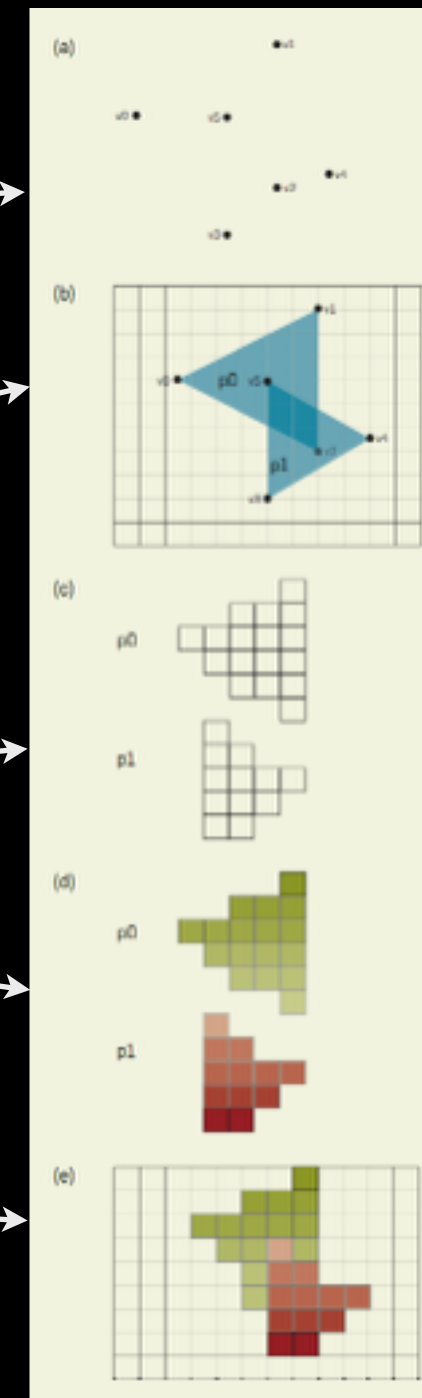


Image from: FATAHALIAN, "A closer look at GPUs"

Programmable Graphics Pipeline

Vertex Shader:
- operates on input vertices
- change size, color, position

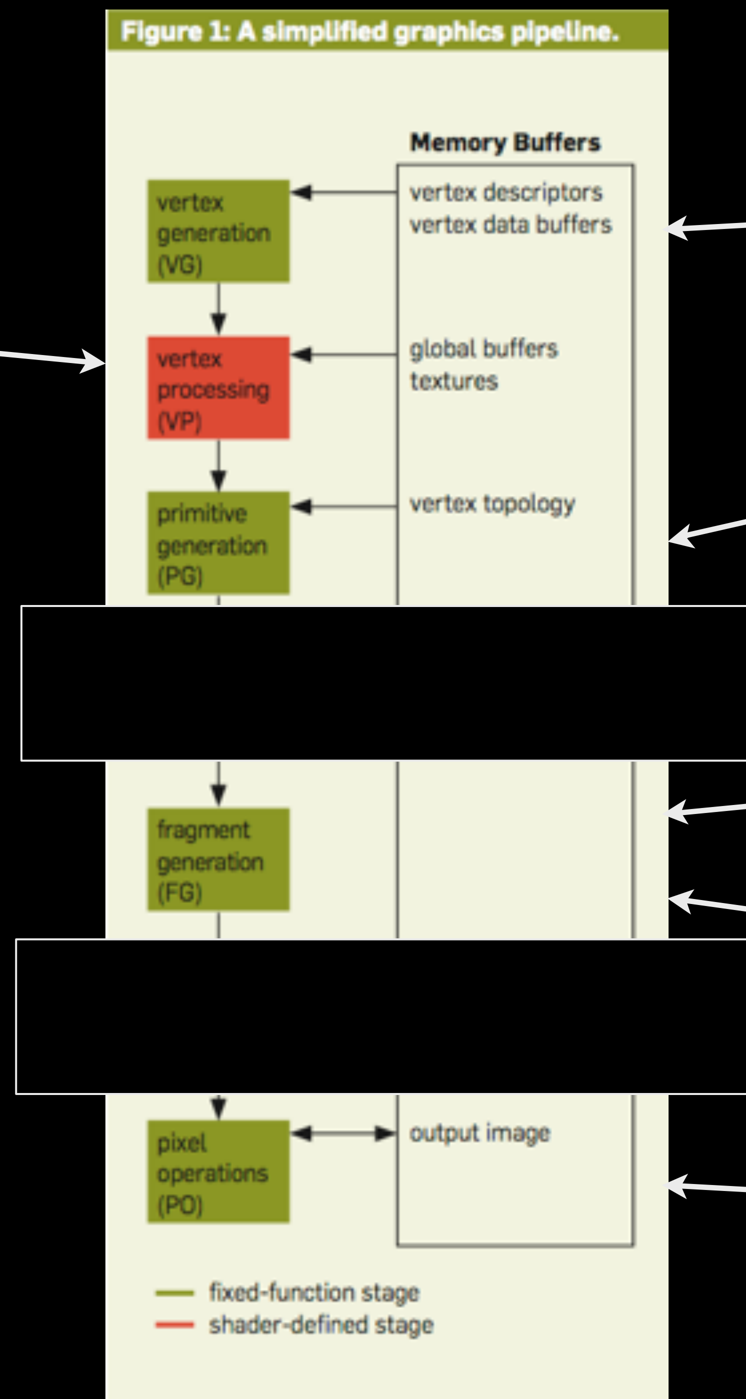


Image from: FATAHALIAN, “A closer look at GPUs”

Input mesh topology

Triangle Assembly & Projection

Pixel coverage

Texture lookup

Occlusion Culling, Blending, etc.

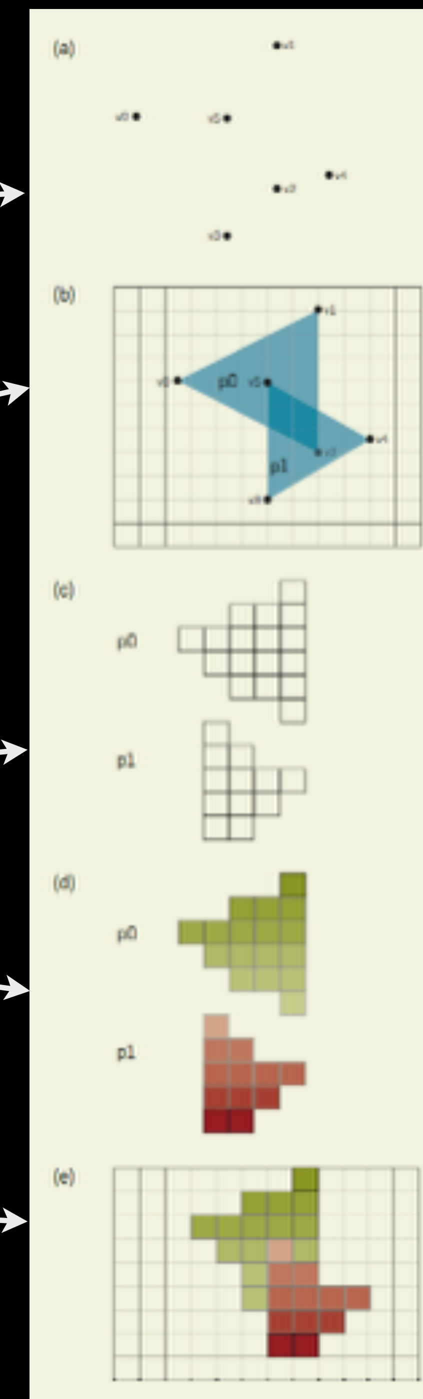


Image from: FATAHALIAN, “A closer look at GPUs”

Programmable Graphics Pipeline

Vertex Shader:
- operates on input vertices
- change size, color, position

Geometry & Tessellation Shader
- destroy or add primitives

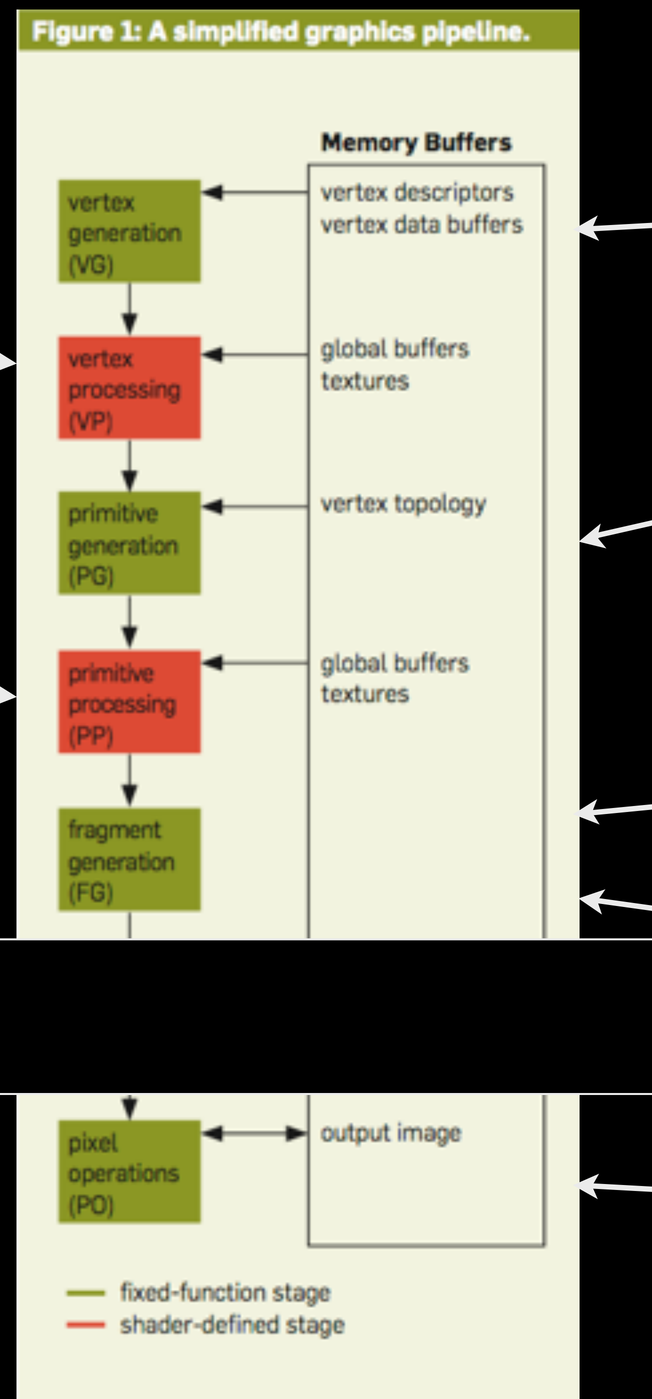


Image from: FATAHALIAN, “A closer look at GPUs”

Input mesh topology

Triangle Assembly
& Projection

Pixel coverage

Texture lookup

Occlusion Culling,
Blending, etc.

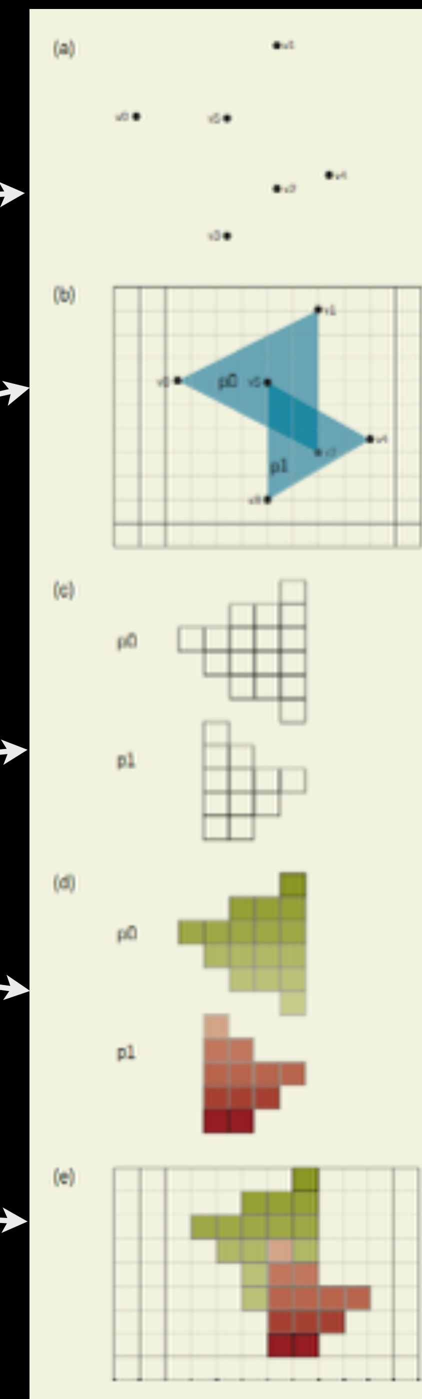


Image from: FATAHALIAN, “A closer look at GPUs”

Programmable Graphics Pipeline

Vertex Shader:
- operates on input vertices
- change size, color, position

Geometry & Tessellation Shader
- destroy or add primitives

Fragment/Pixel Shader:
- color-code pixels
- discard pixels

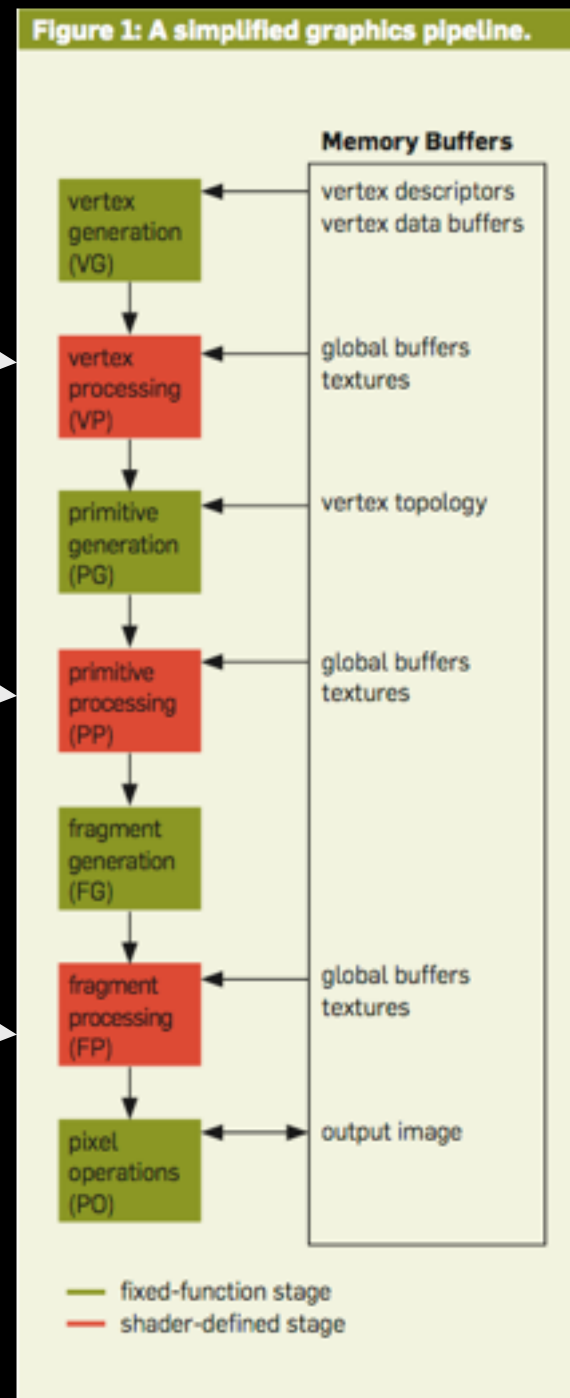


Image from: FATAHALIAN, "A closer look at GPUs"

Input mesh topology

Triangle Assembly & Projection

Pixel coverage

Texture lookup

Occlusion Culling, Blending, etc.

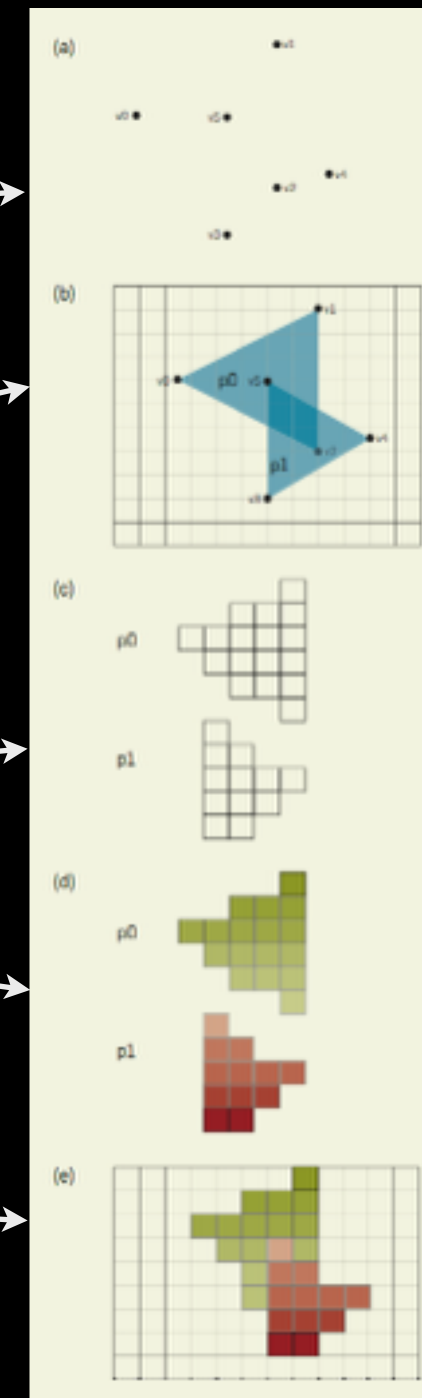


Image from: FATAHALIAN, "A closer look at GPUs"

Graphics Hardware

Efficient hardware units

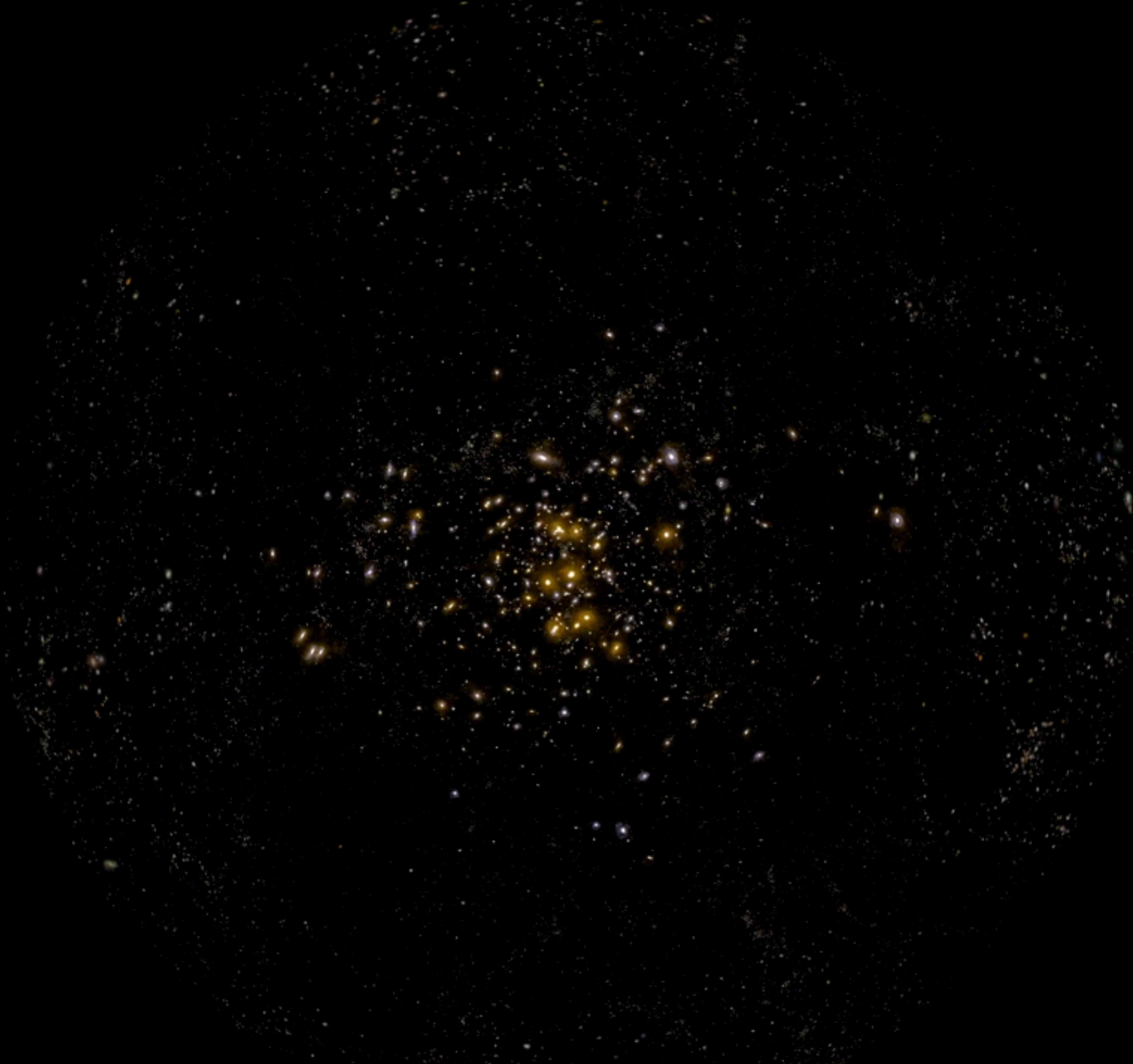
texture mapping

rasterization
pixel operation (blending)
visibility testing

} **not available in CUDA/OpenCL**

Data parallelism (SIMD)

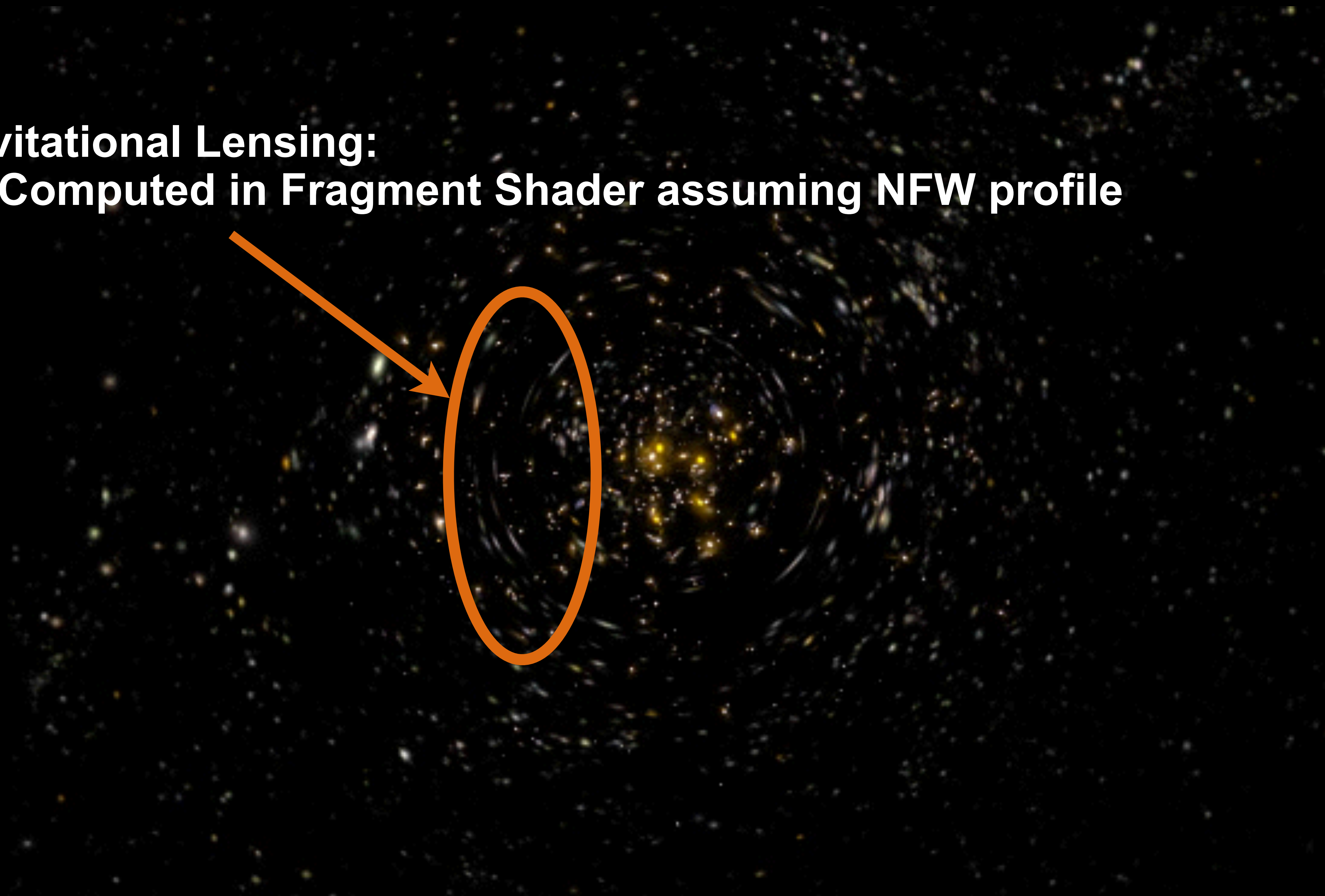
- triangles and pixels processed in parallel
-> massively parallel architectures with thousands of cores



AMNH Planetarium Show: “Dark Universe” (2013), narrated by Neil deGrasse Tyson,
Gravitational Lensing Scene (Kaehler, Emmart, Abel)

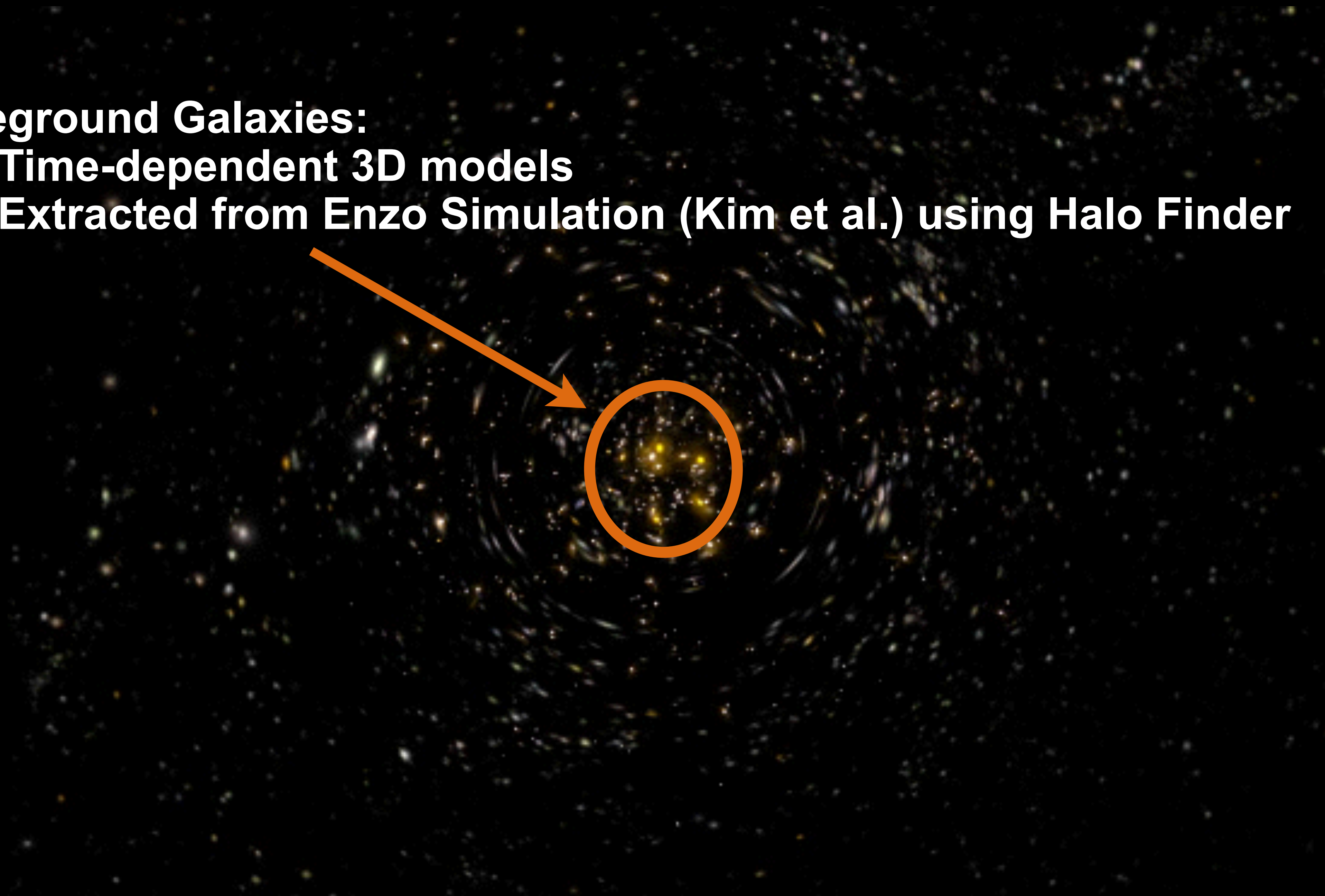
Background Galaxies:

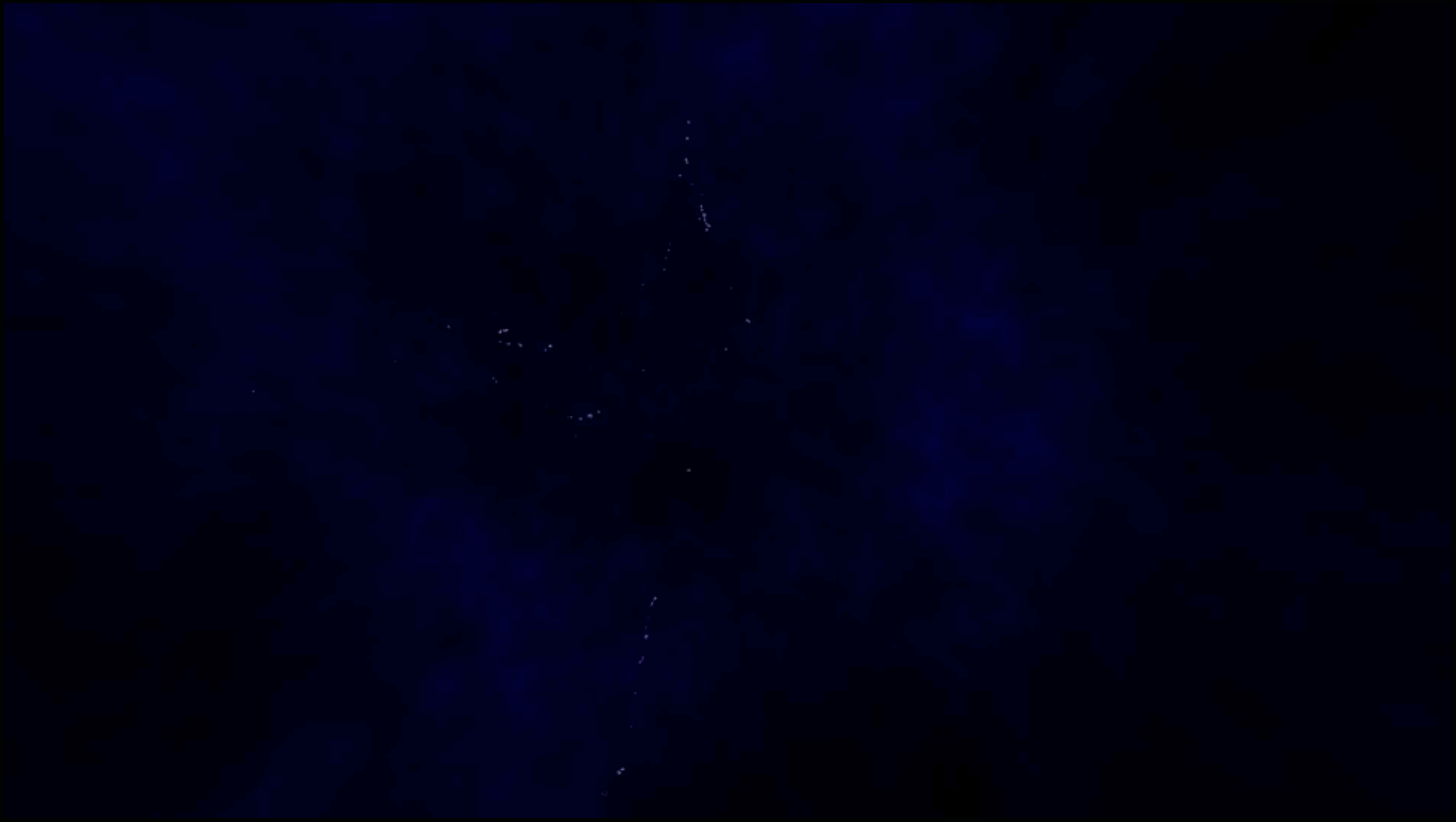
Halos from Millenium XXL Simulation (Angulo et al.)


Gravitational Lensing:
Computed in Fragment Shader assuming NFW profile

Foreground Galaxies:

- **Time-dependent 3D models**
- **Extracted from Enzo Simulation (Kim et al.) using Halo Finder**





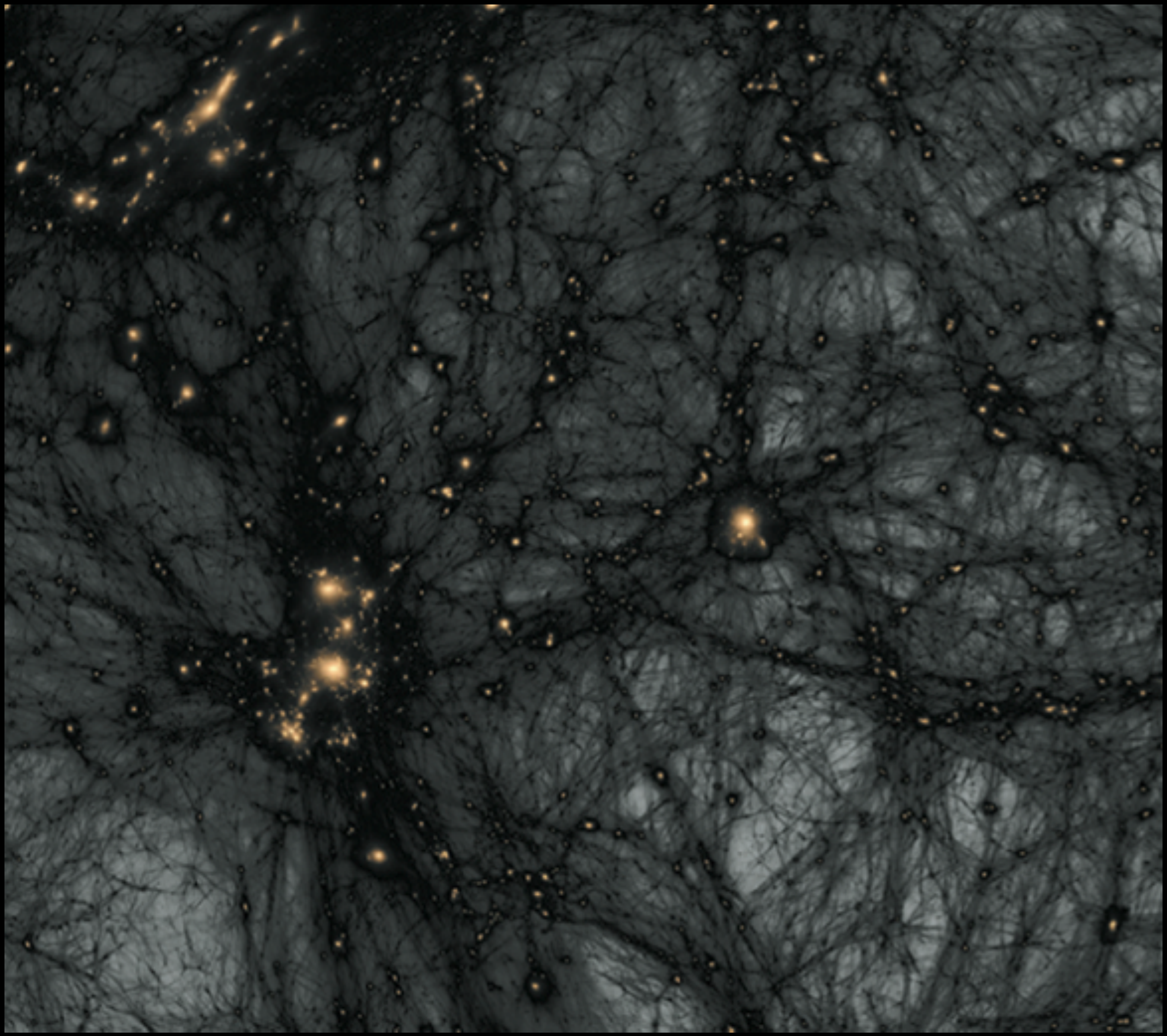
Rendering for American Museum of Natural History Show “The Big Bang”, Narrated by Liam Neeson
Kaehler, Kim, Abel (Stanford/SLAC)

LENSING SCENE

rendered with 100 MPixel resolution

more than 1 TByte of image data

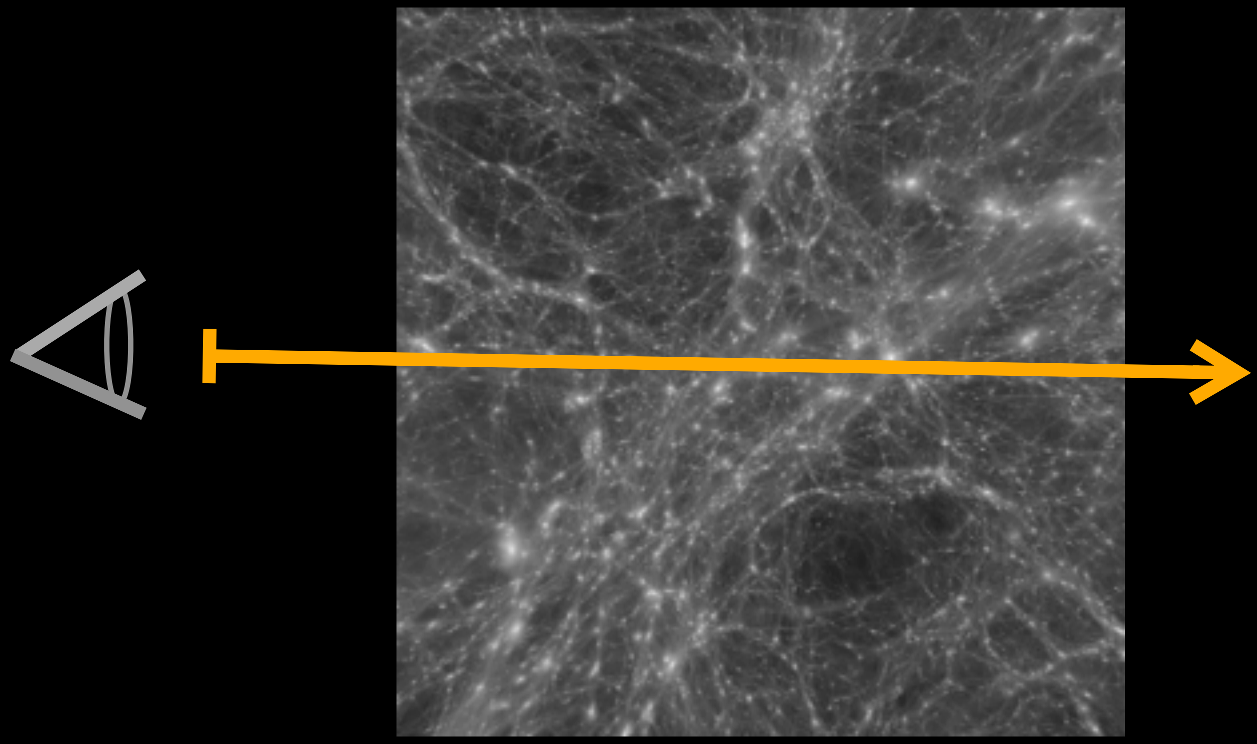
VISUALIZATION OF N-BODY DARK MATTER SIMULATIONS USING RASTERIZATION GRAPHICS



DENSITY PROJECTIONS

Integrated density along line-of-sight

$$\rho_{proj} = \int \rho(x)dx$$



DENSITY PROJECTIONS

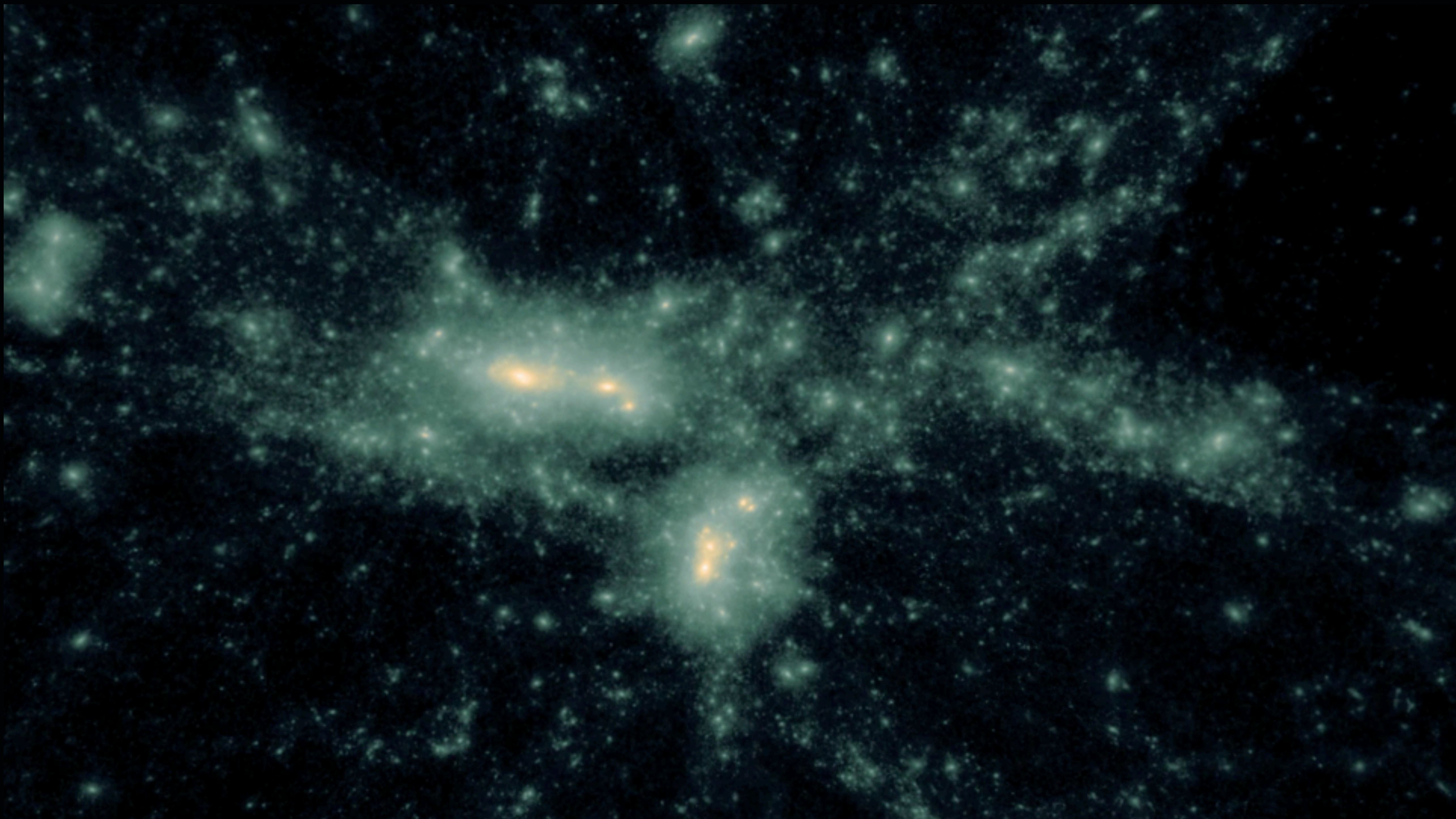
Estimation of Dark Matter Densities between tracers ?

DENSITY PROJECTIONS

Kernel smoothing (SPH)

- density around tracer estimated by volume of sphere of n-nearest neighbors
- smoothed by kernel profile centered at particle’s position
- Box filter, Gaussian, Cubic-splines, ...

see e.g. Monaghan [1998], Fraedrich et al. [2009]

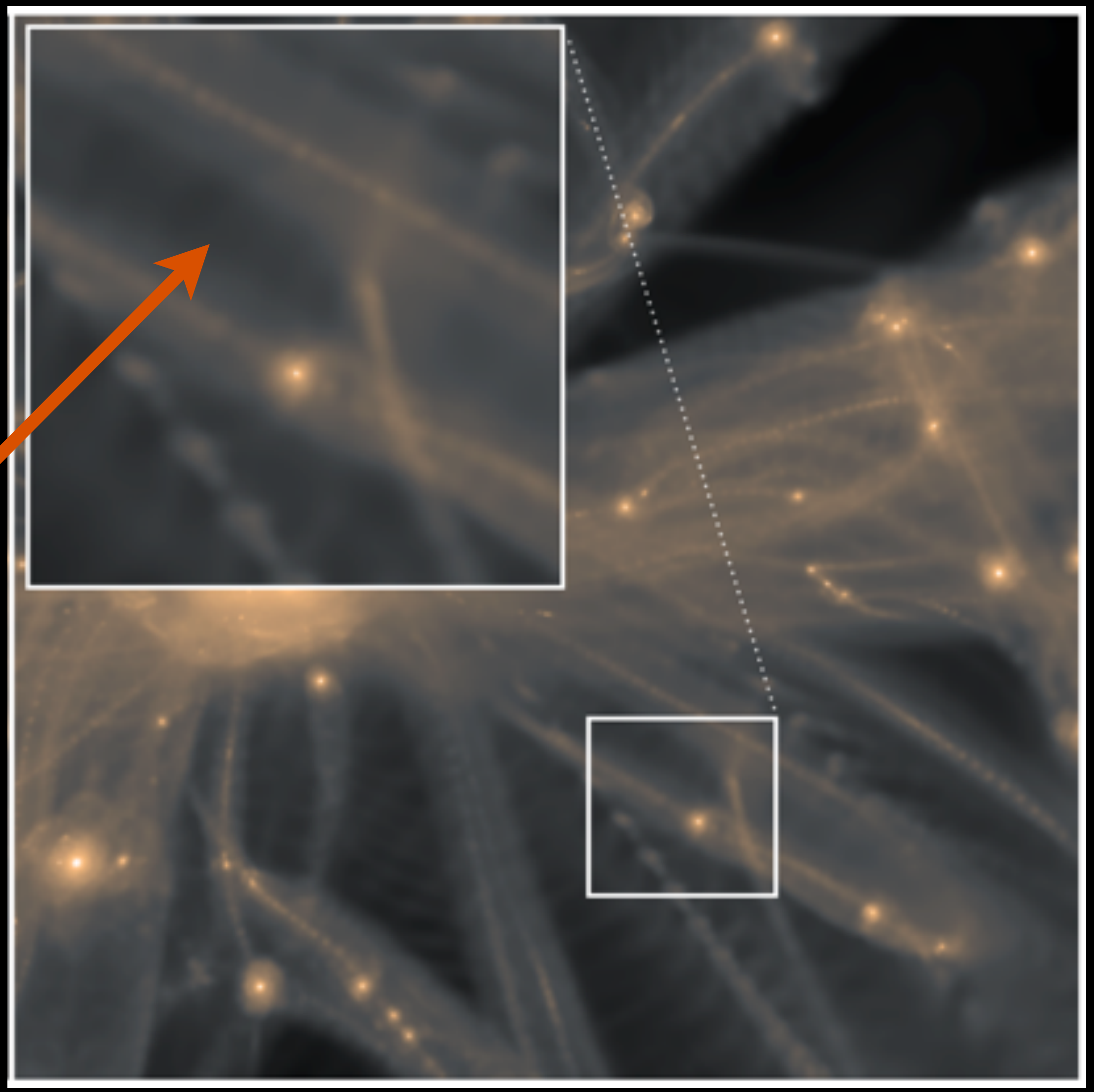


Simulation data: Hao-Yi Wu (Stanford), Oliver Hahn (Stanford), Risa Wechsler (Stanford)

DENSITY PROJECTIONS

Problems

to much smoothing

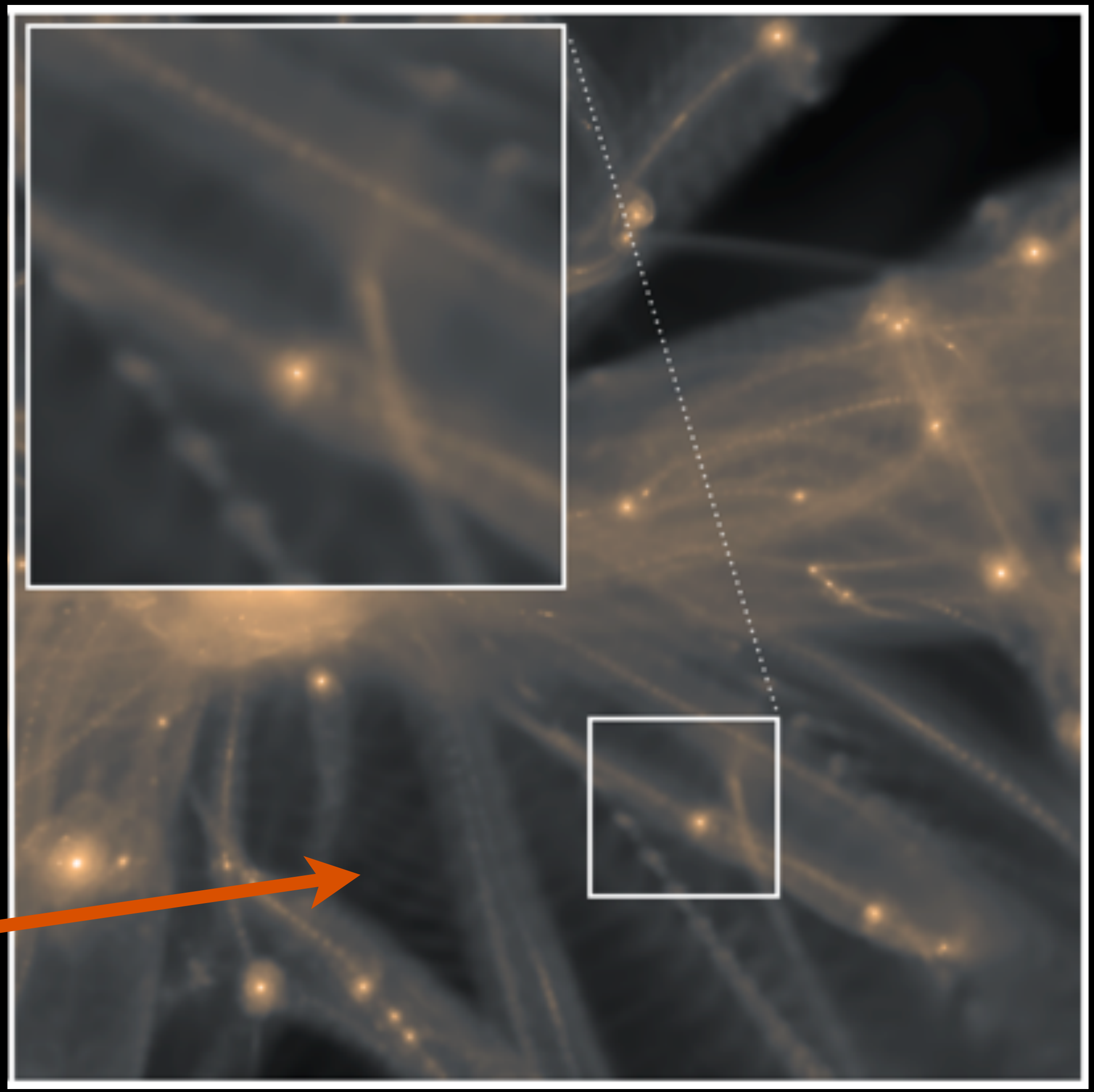


Kaehler, Hahn, Abel [2012]

DENSITY PROJECTIONS

Problems

noise in under-dense regions



Kaehler, Hahn, Abel [2012]

TESSELLATION

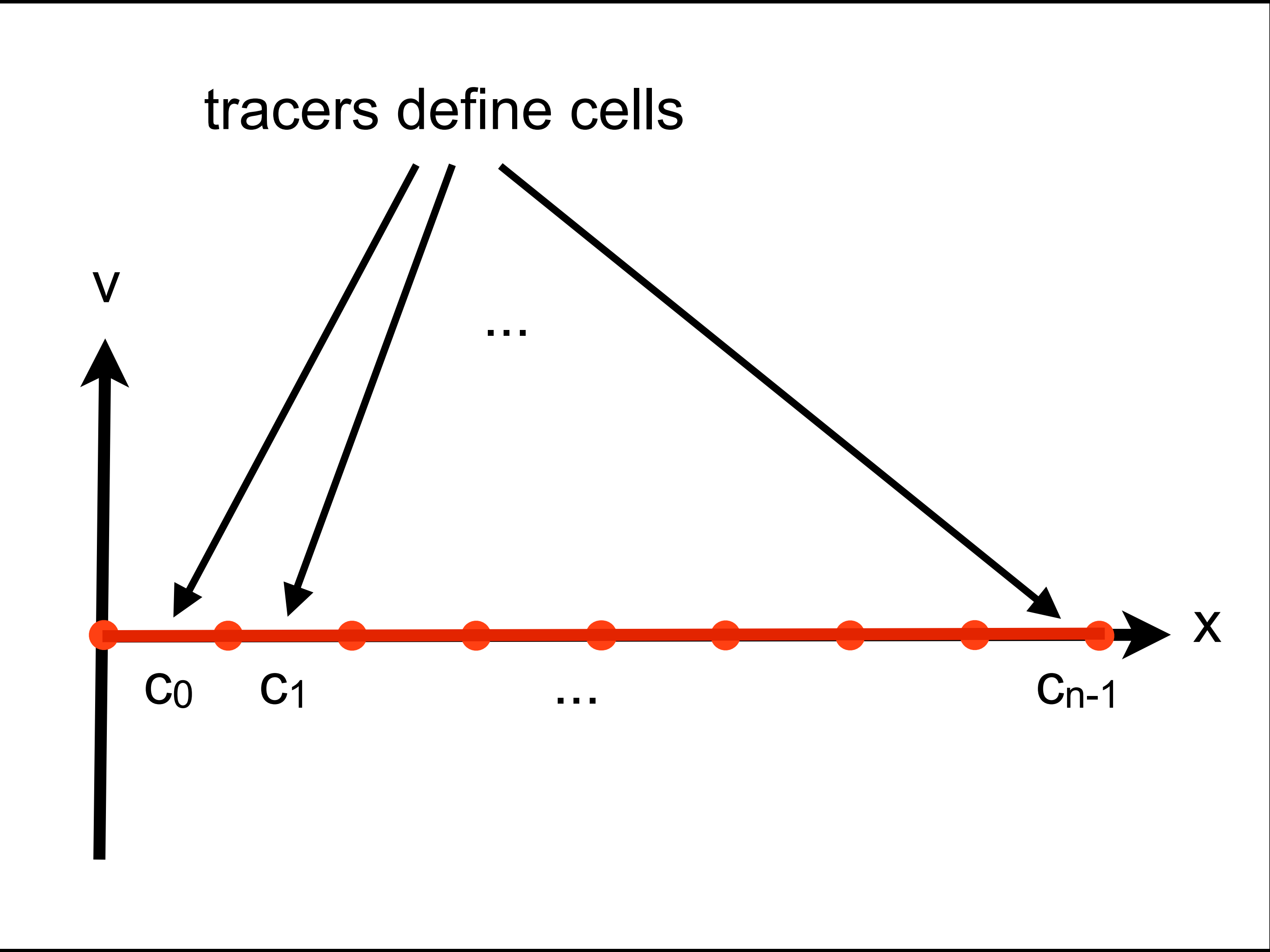
Tessellation of Dark Matter Sheet in Phase-Space

Abel, Hahn, Kaehler [2011]

Initial Conditions:

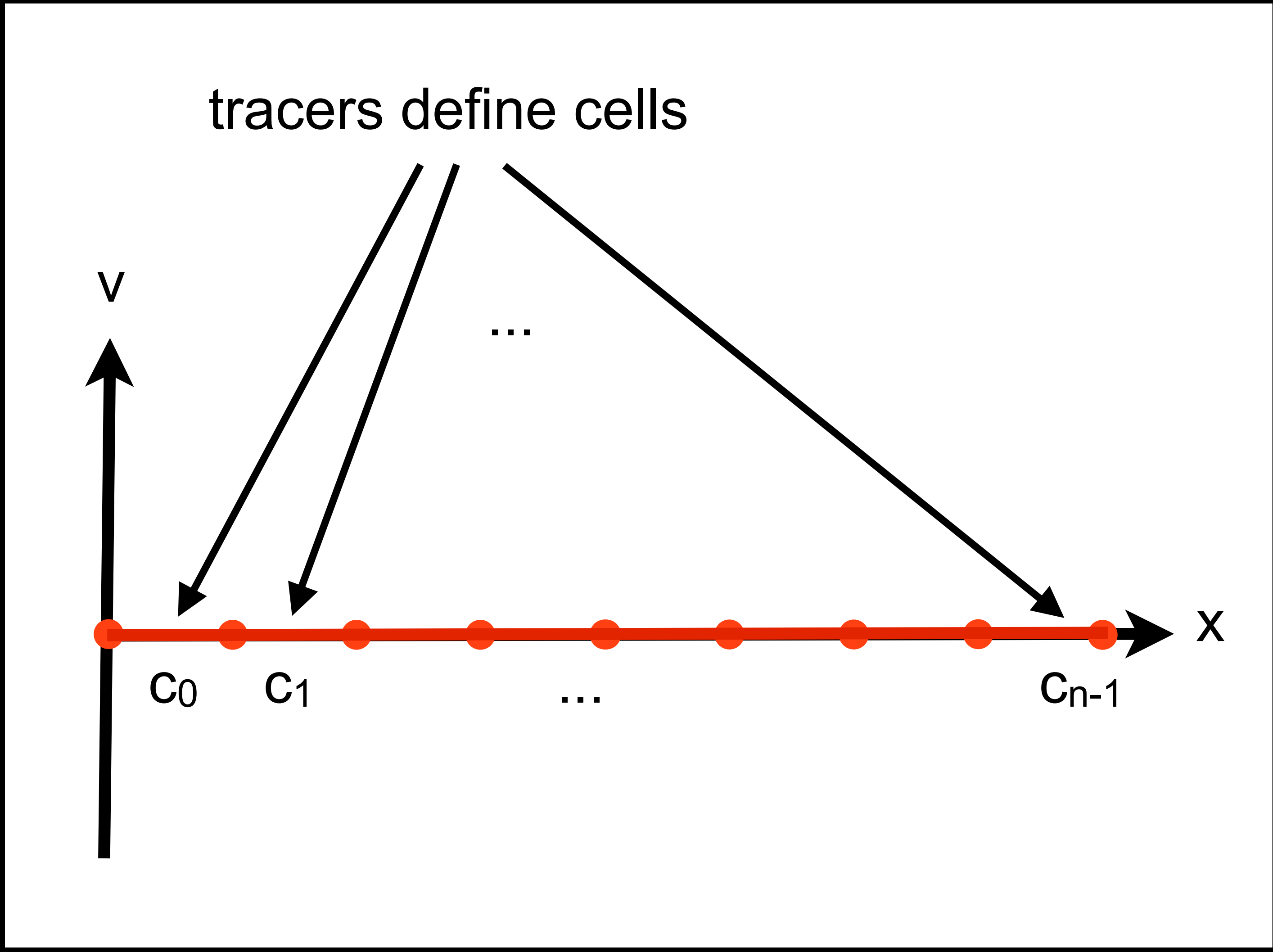
Tracers placed on regular grid

Pairs of tracers define cells



TESSELLATION

Assumption: same amount of mass per cell

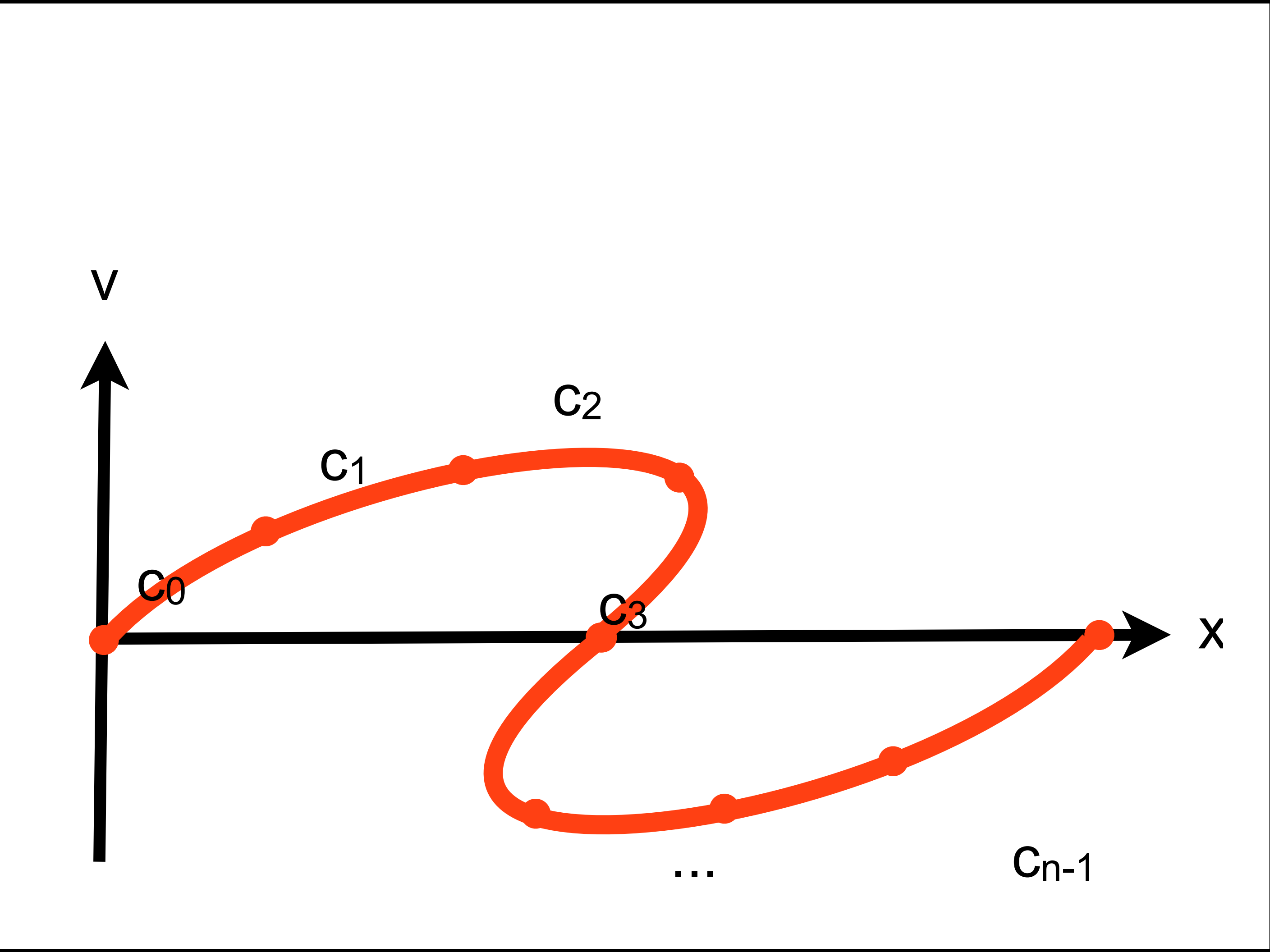


TESSELLATION

Connectivity constant over time

Motion of tracers deforms cells

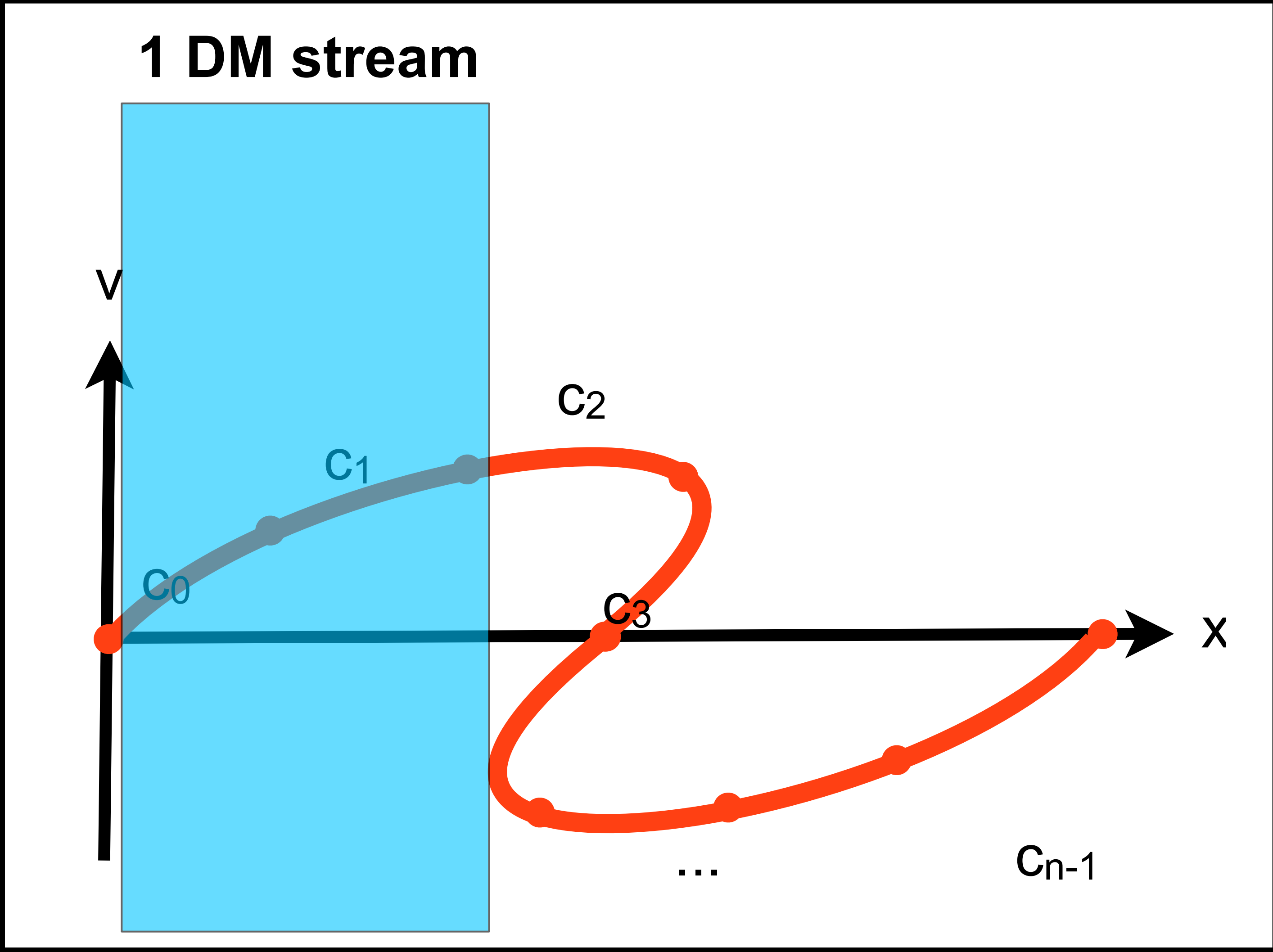
Densities change over time



TESSELLATION

Mass density computation:

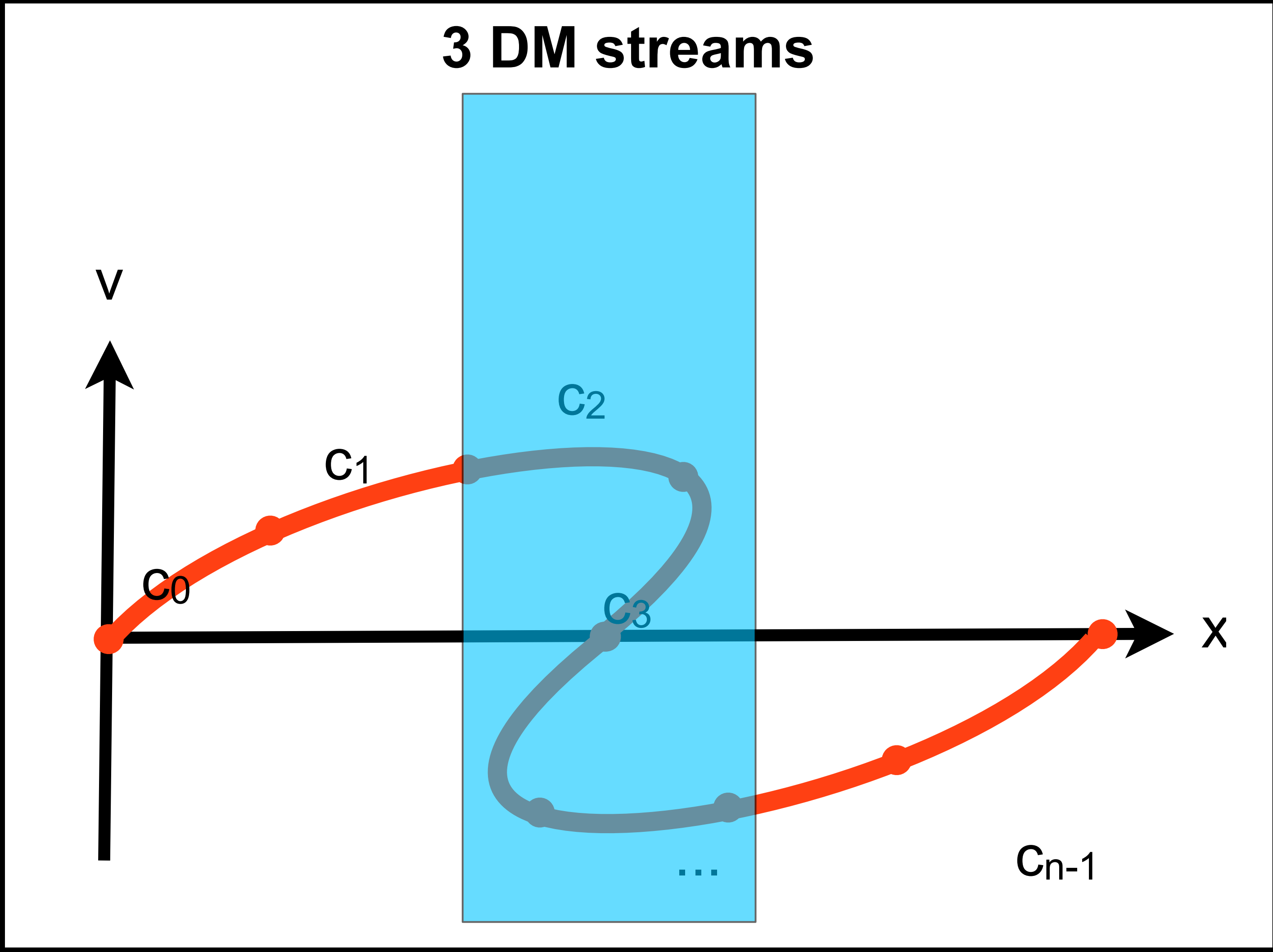
project cells into position space



TESSELLATION

Mass density computation:

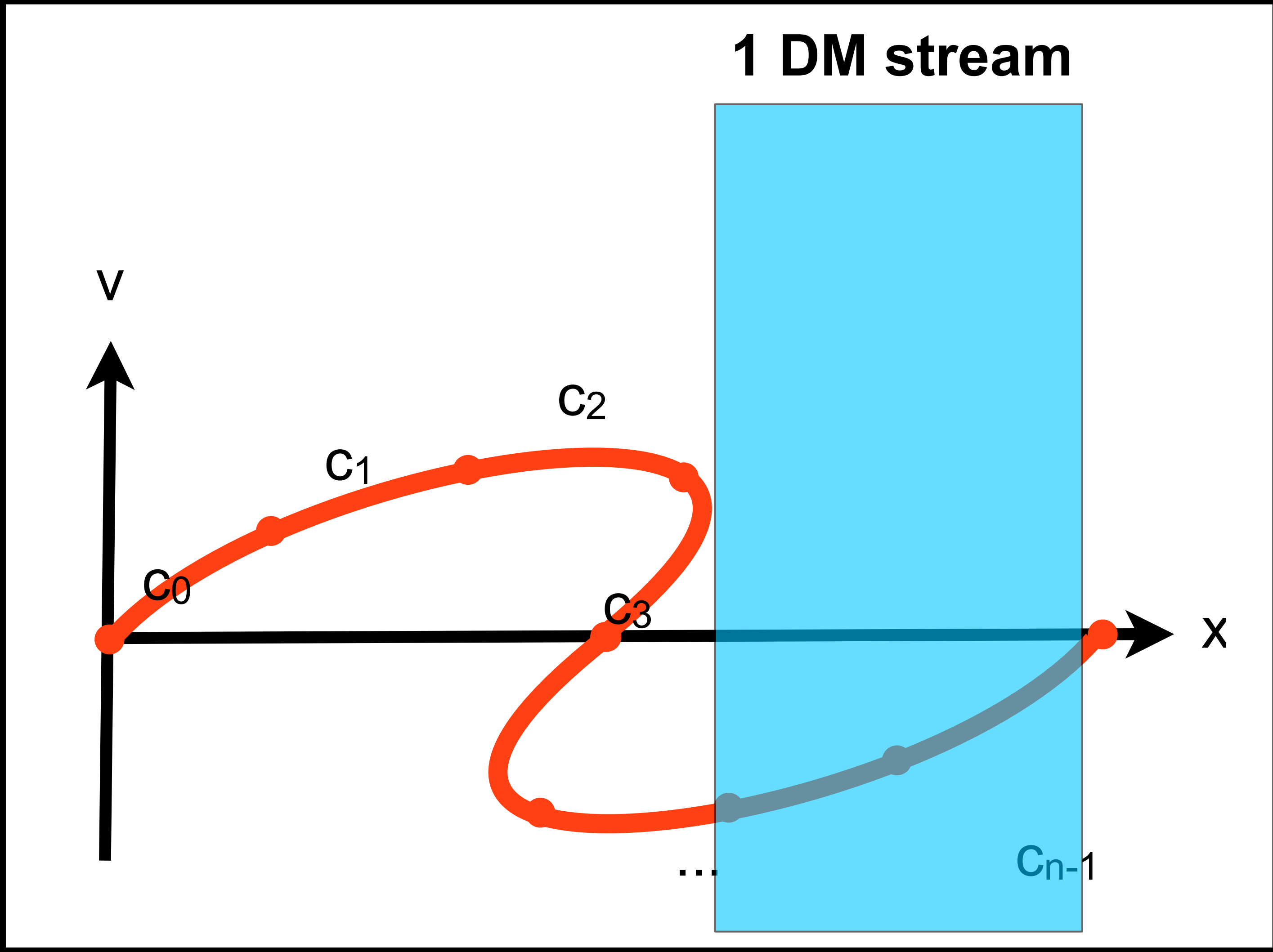
project cells into position space



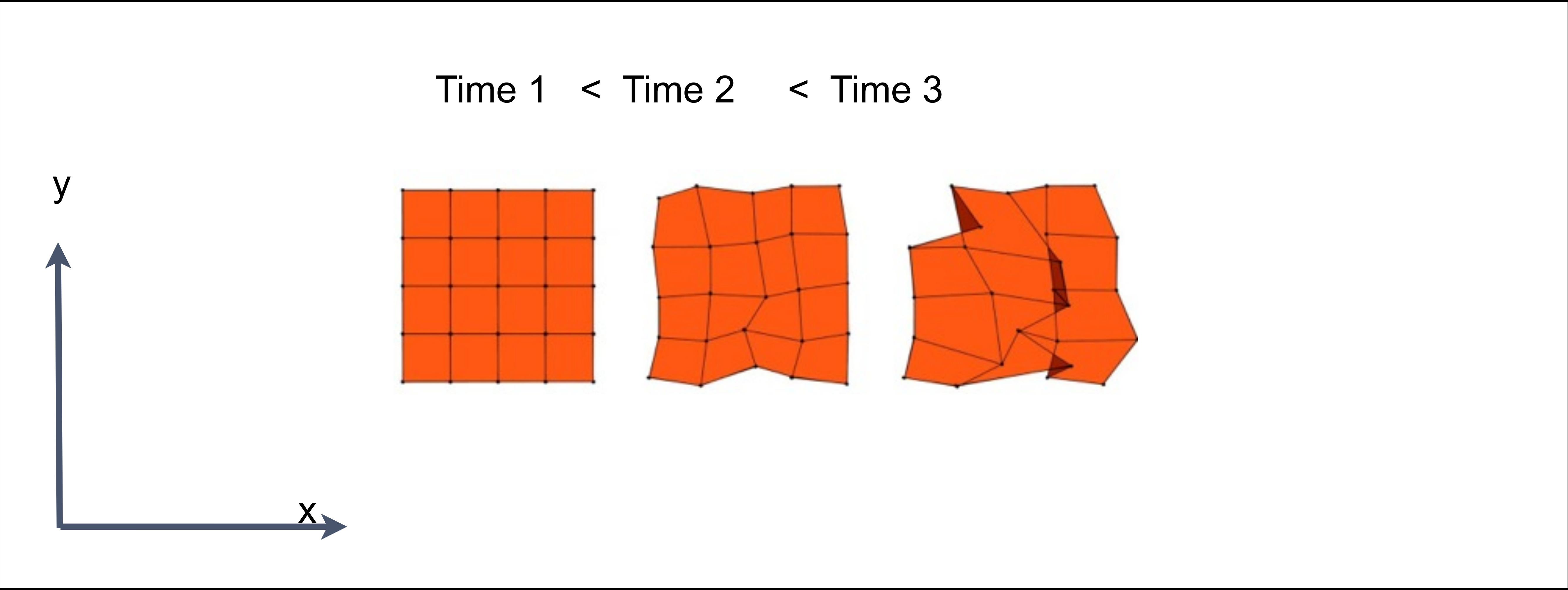
TESSELLATION

Mass density computation:

project cells into position space



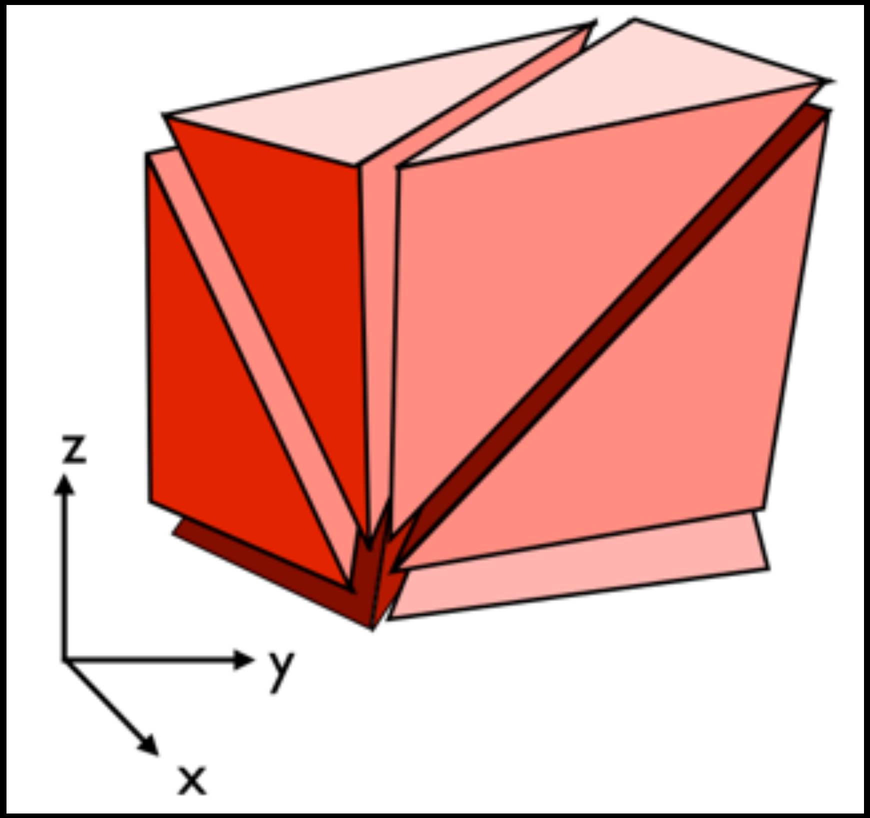
TESSELLATION



TESSELLATION

Subdivision of cubical cells by tetrahedra

always convex



Freudenthal Triangulation[1942]

6 Tetrahedra per Cube

Tetrahedra defined by triangles, so let’s use rasterization graphics

PROBLEMS

Large number of tetrahedral elements

512^3 tracer particles ⇒ 8×10^8 tetrahedra

PROBLEMS

Large number of tetrahedral elements

512^3 tracer particles ⇒ 8×10^8 tetrahedra

Memory requirements for 512^3 tracer particles

positions: 1.5 GBytes

densities: 3 GBytes

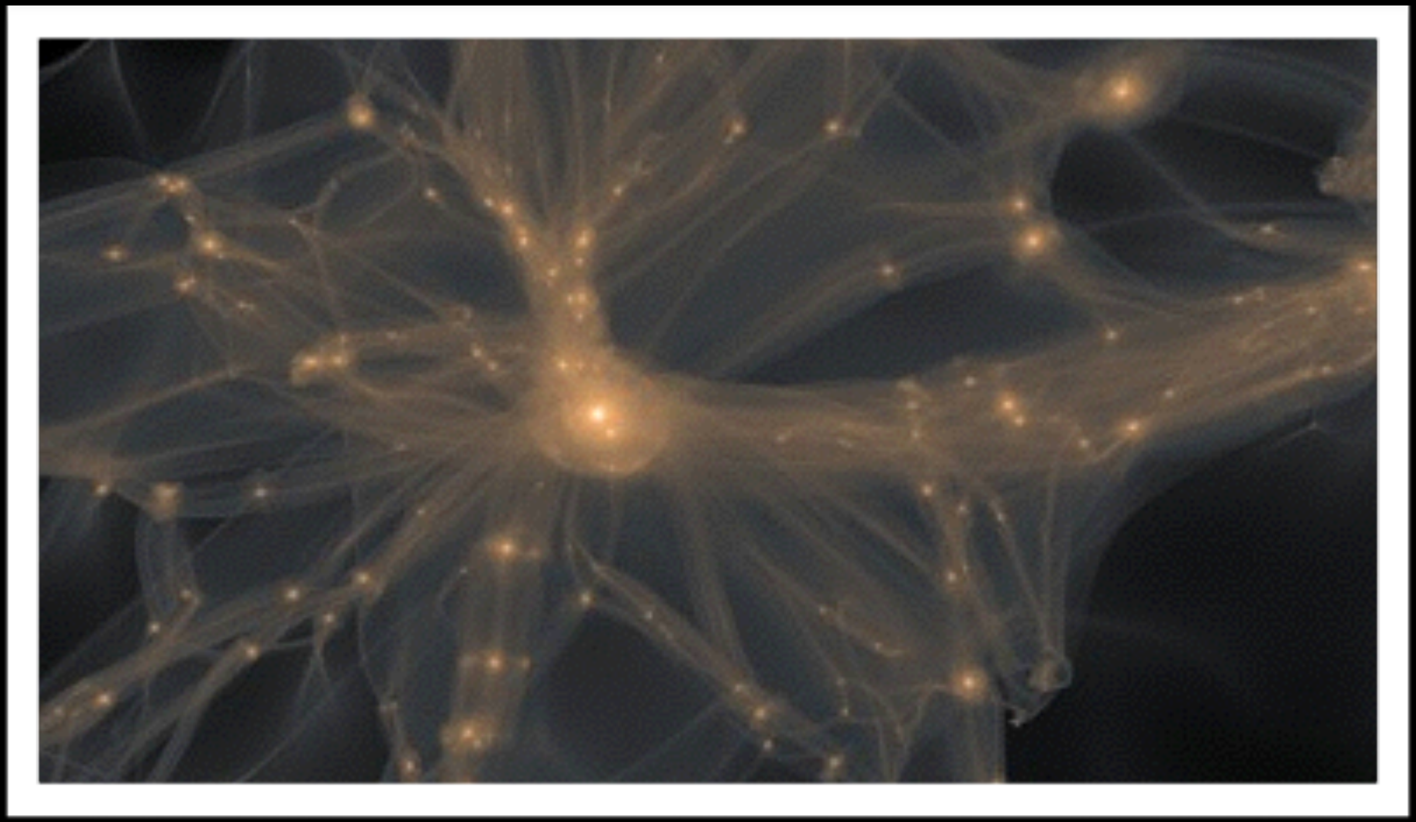
connectivity: ~17 GBytes

GPU-BASED DENSITY PROJECTION APPROACH

Features

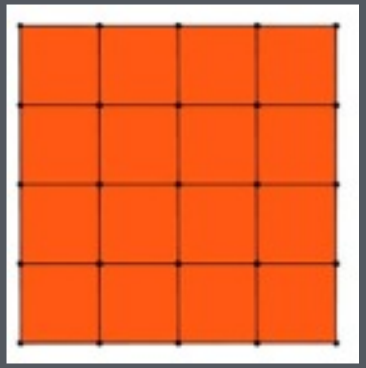
- no-connectivity information transferred between CPU and GPU
- no preprocessing:
  - geometries of tets constructed on-the-fly on GPU
  - densities of tets computed on-the-fly on GPU

Kaehler, Hahn, Abel [2012]



GPU PROCESSING

3D TEXTURE



- Tracer positions stored in 3D RGB texture
- Texel coordinate based on position at initial time

GPU

VERTEX SHADER

- one instance per cell (instanced rendering)
- samples cell’s vertices

GEOMETRY SHADER

- computes volume & density of tetrahedra
- 6 invocations per vertex shader instance

DATA STORAGE

Memory requirements for 512^3 tracer particles

positions: 1.5 GBytes

densities: 3 GBytes

connectivity: ~17 GBytes

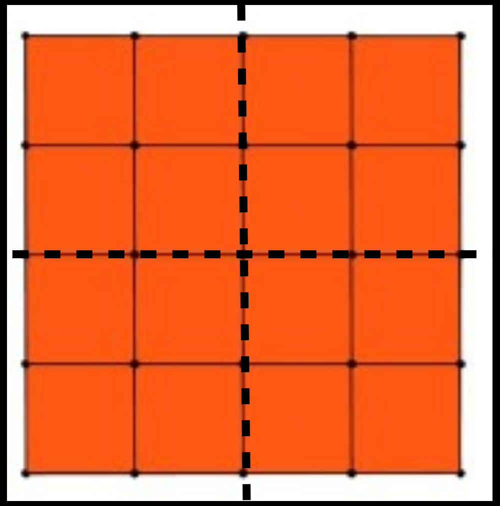
DISTRIBUTED RENDERING

Texture split in ‘bricks’

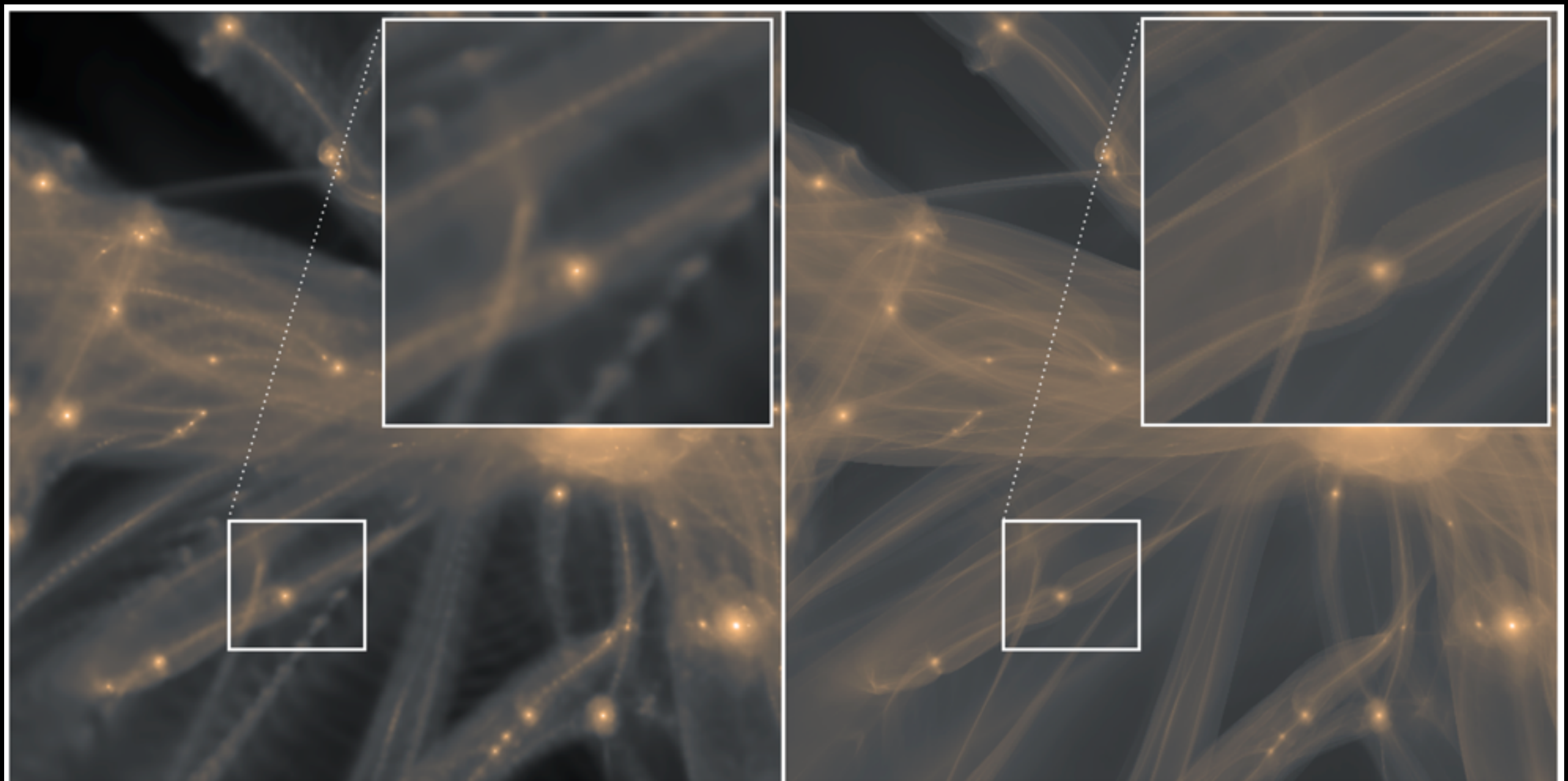
Bricks share data layer on faces

Bricks rendered on separate node

Compositing of partial results



RESULTS

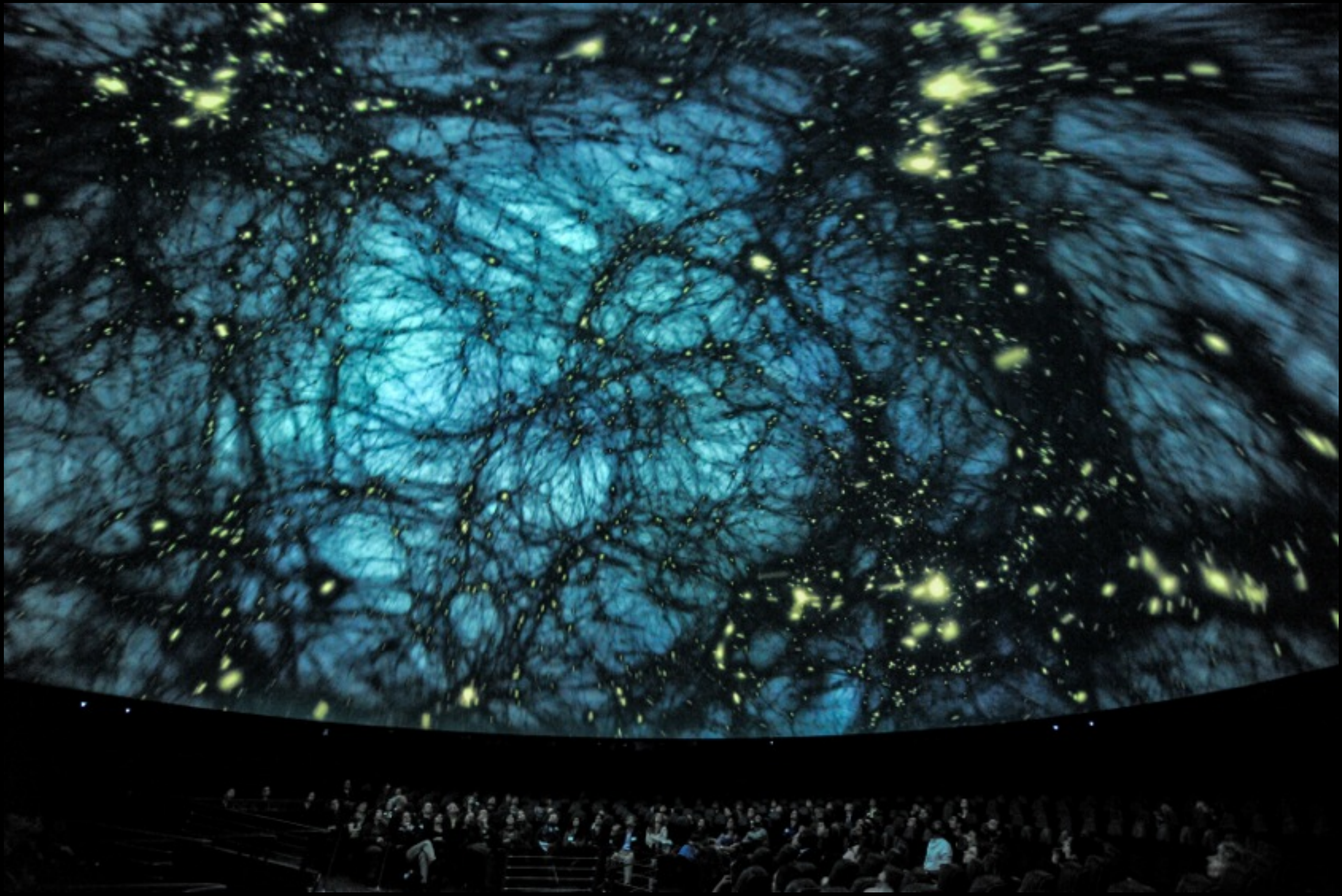


SPH kernel smoothing

Phase-Space Tessellation:
cell-projection approach

Kaehler, Hahn, Abel [2012]

“Dark Universe” Planetarium Show



Courtesy of D. Finnin, American Museum of Natural History

LARGE-SCALE-STRUCTURE SCENE

Simulation:

2300 time steps

768^3 tracers (256^3 run replicated 27 times)

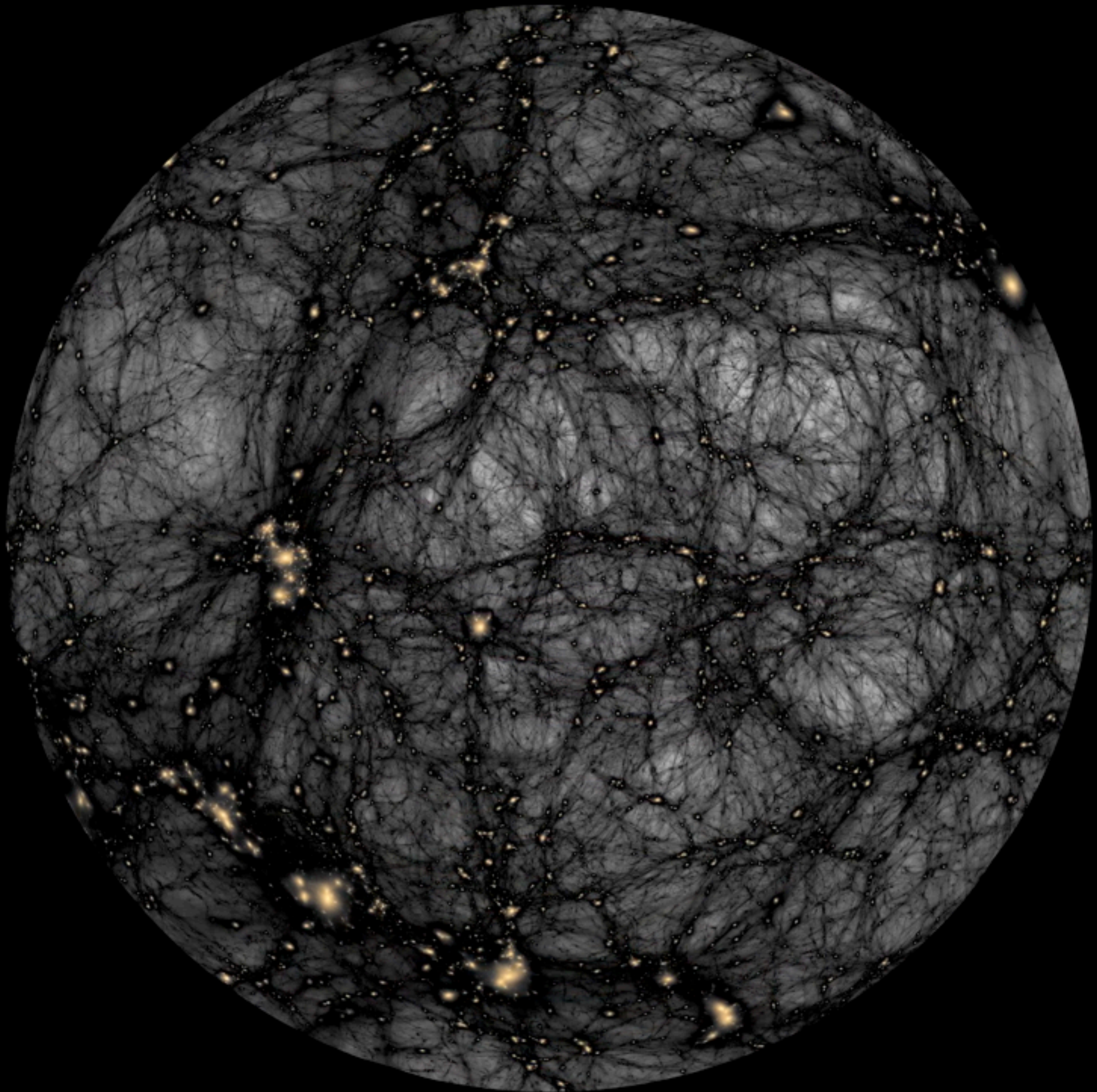
Rendering:

16 billion triangles per snapshot

24 MPixel resolution

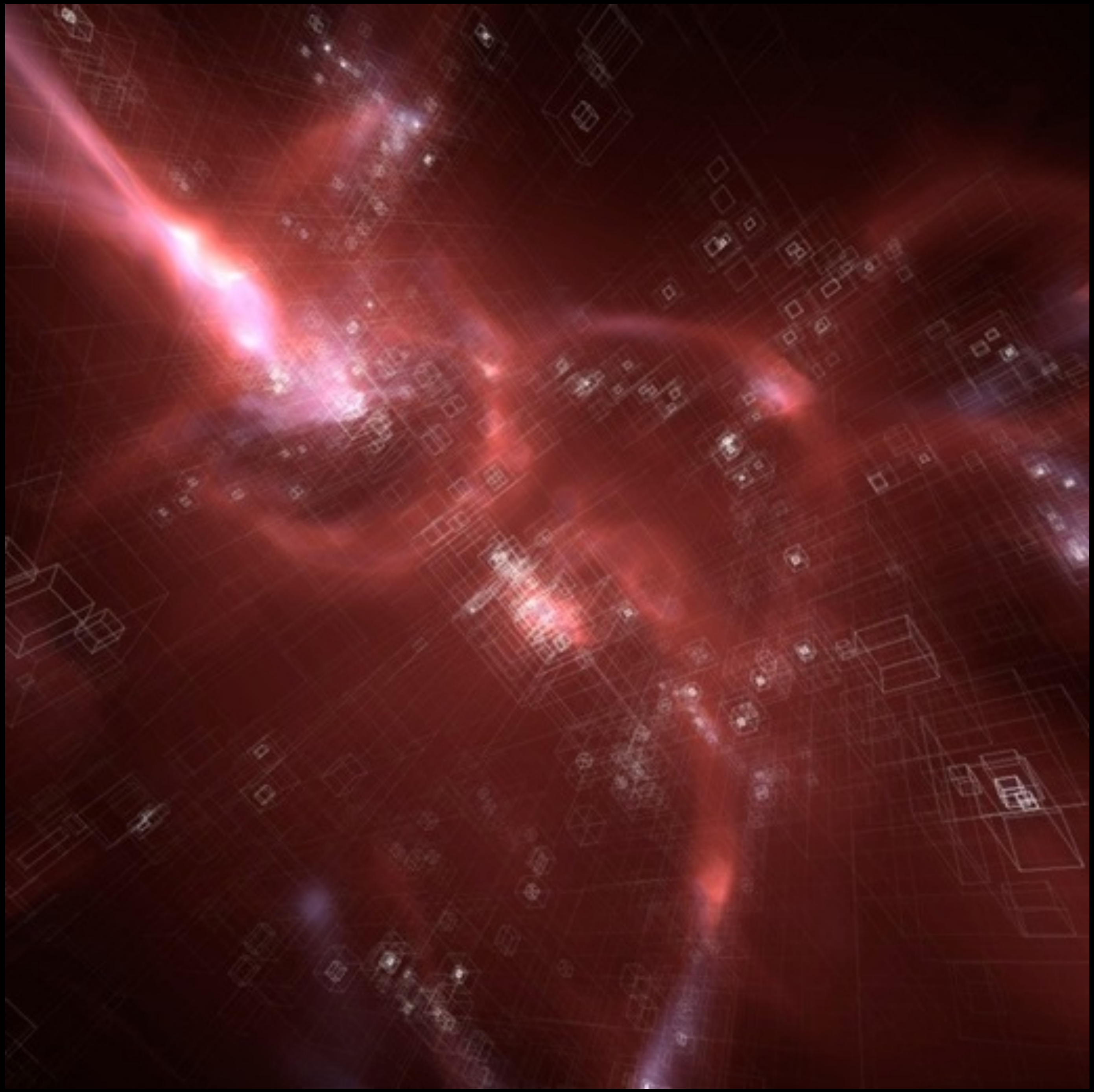
Hardware:

Single Workstation with Nvidia Quadro 4000 card



Gravitational Lensing Scene for AMNH Planetarium Show: “Dark Universe” (2013) , (Kaehler, Emmart, Abel, Hahn)

GPU-Based Direct Volume Rendering of Adaptive Mesh Refinement Data



DIRECT VOLUME RENDERING

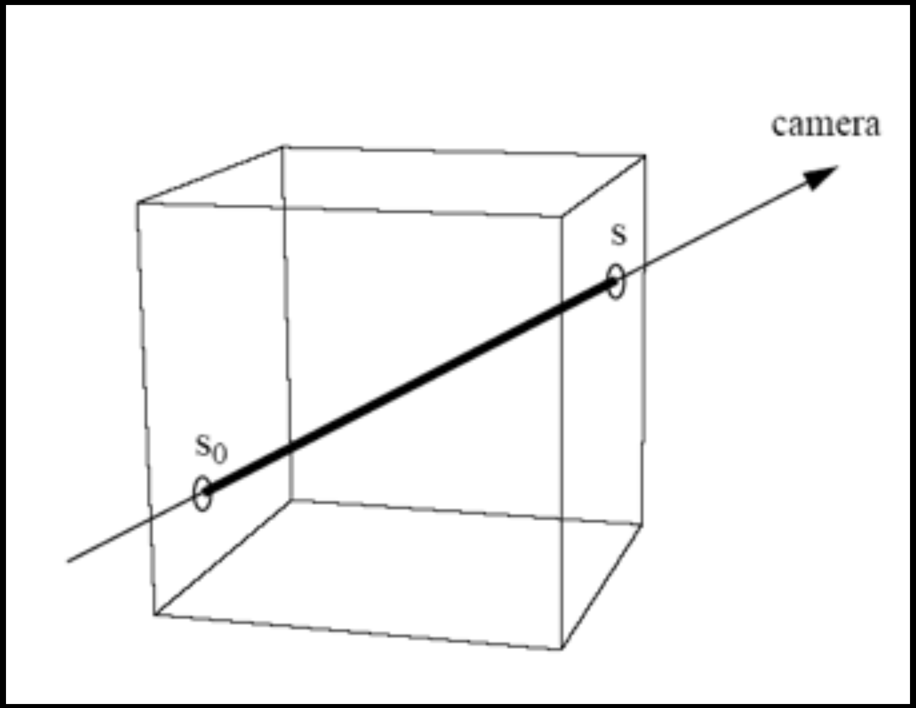
Assign emission and absorption coefficient to data samples

Display resulting light intensity in image plane

—> Solve radiation transfer equation

$$\frac{\partial}{\partial s} I(\mathbf{x}, \mathbf{n}, \nu) = -\kappa(\mathbf{x}, \mathbf{n}, \nu) I(\mathbf{x}, \mathbf{n}, \nu) + q(\mathbf{x}, \mathbf{n}, \nu)$$

Assumption: no scattering



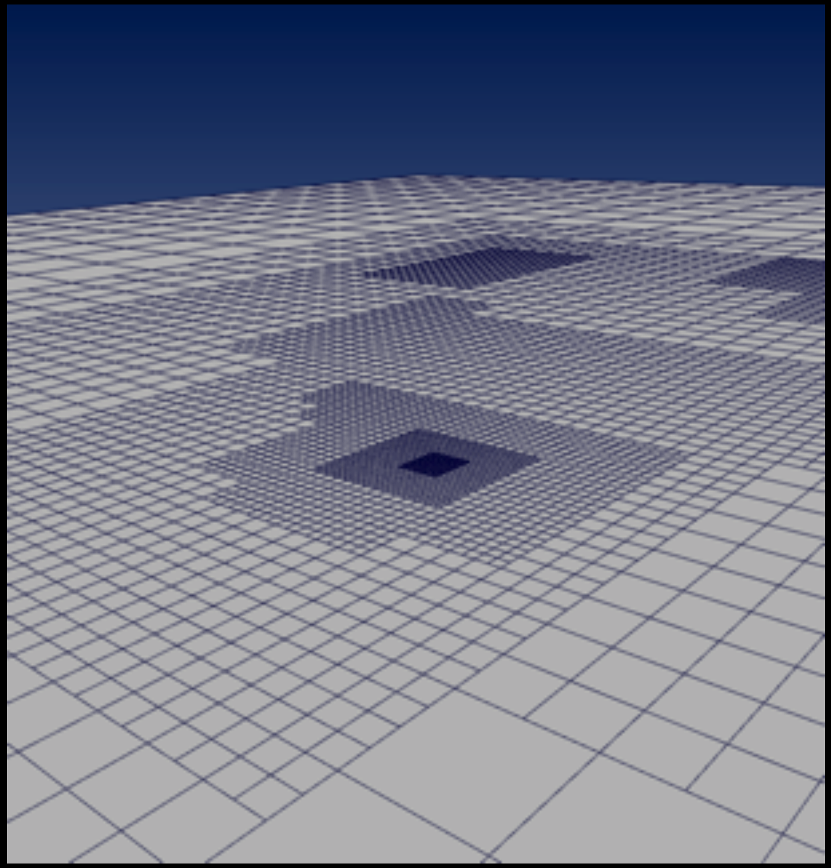
Simplified Transfer Equation

$$I(s) = I(s_0)e^{-\tau(s_0,s)} + \int_{s_0}^{s} q(s')e^{-\tau(s',s)}ds'$$

$$\tau(s_0, s_1) := \int_{s_0}^{s_1} \kappa(t)dt$$

STRUCTURED AMR

Refined regions overlap coarse ones

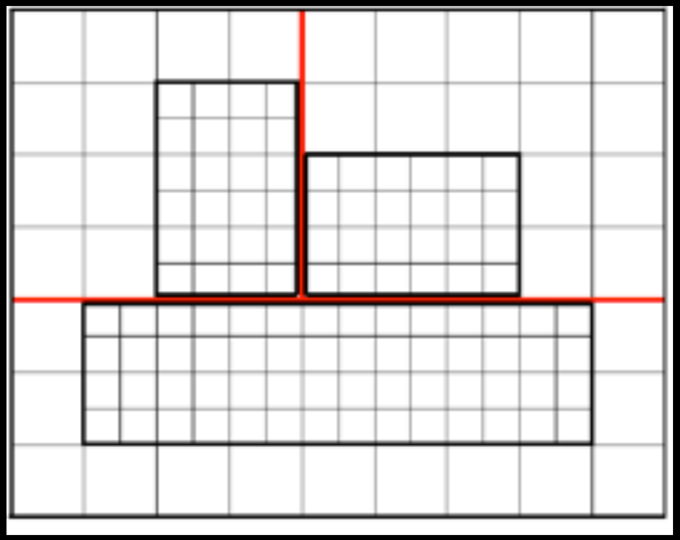


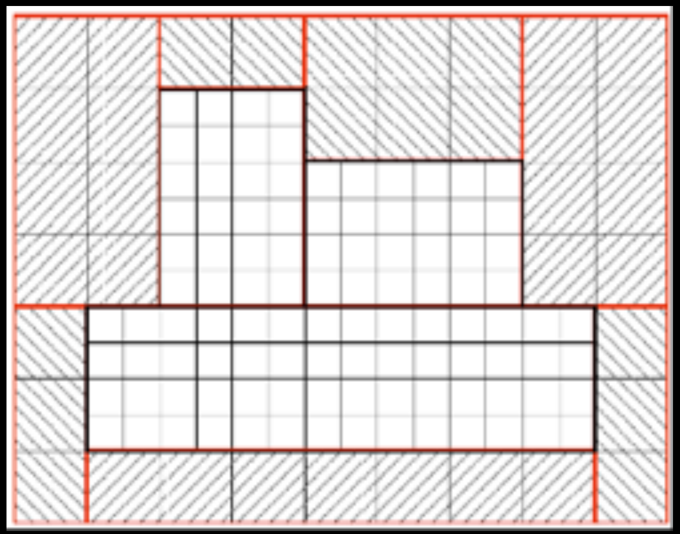
Problem of overlapping regions

Decomposition of data domain

- Adaptive kD-tree
- Nodes consist of non-overlapping boxes
- No visibility cycles

Supports front-to-back / back-to-front traversal





Kaehler, Hege [2002]

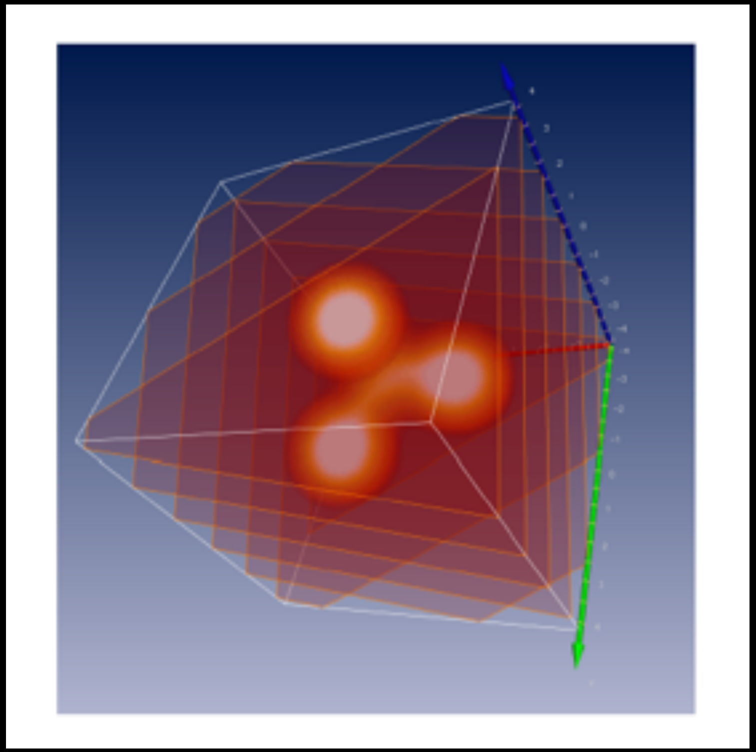
“Slice-Based” Direct Volume Rendering

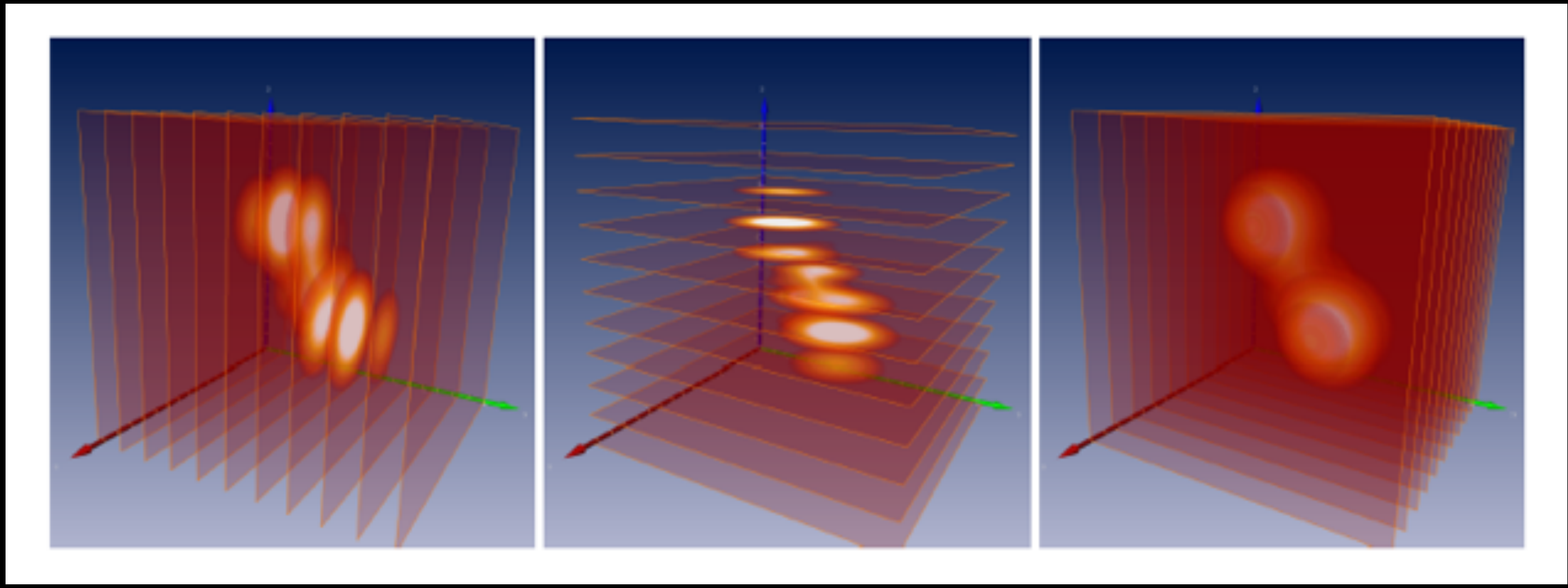
Exploit GPU support for filtering and blending

Traverse kD-tree on CPU

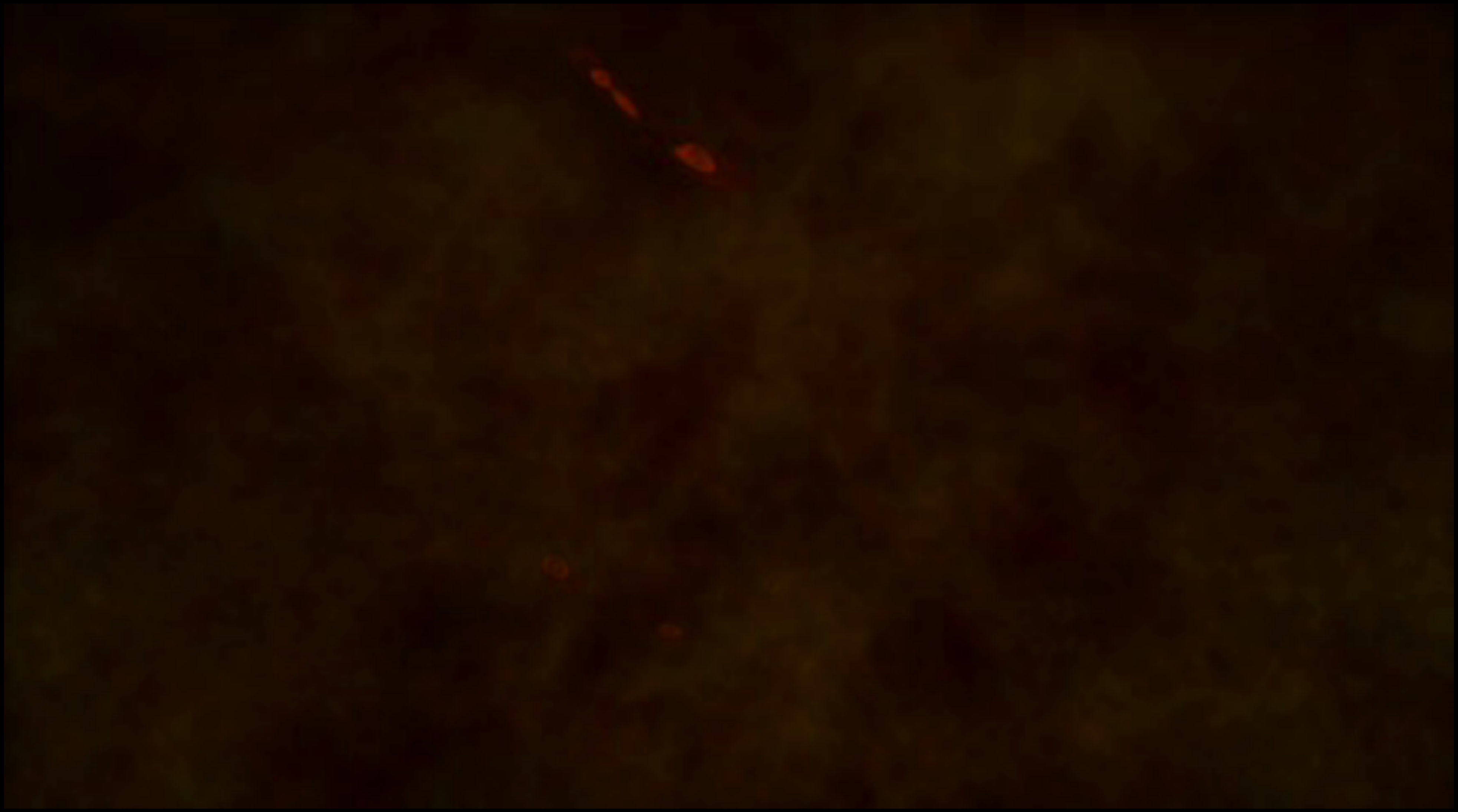
For each node at sufficient resolution:

- Upload data block as 3D texture
- Extraction of slices using texture hardware
- Blending into frame buffer





Kaehler, Hege [2002]



Animation for Discovery Channel Television Show: “The Unfolding Universe” (2002)
Visualization: R. Kaehler (KIPAC), D. Cox (NCSA), R. Patterson (NCSA), S.Levy (NCSA)
Numerical Simulation: T. Abel (KIPAC)

	CPU	GPU
Texture-Based	• ray/geometry setup • traversal of kD-tree • LOD selection • access to whole domain • data I/O	• data sampling • blending

GPU-BASED RAYCASTING

Traverse kD-tree on CPU

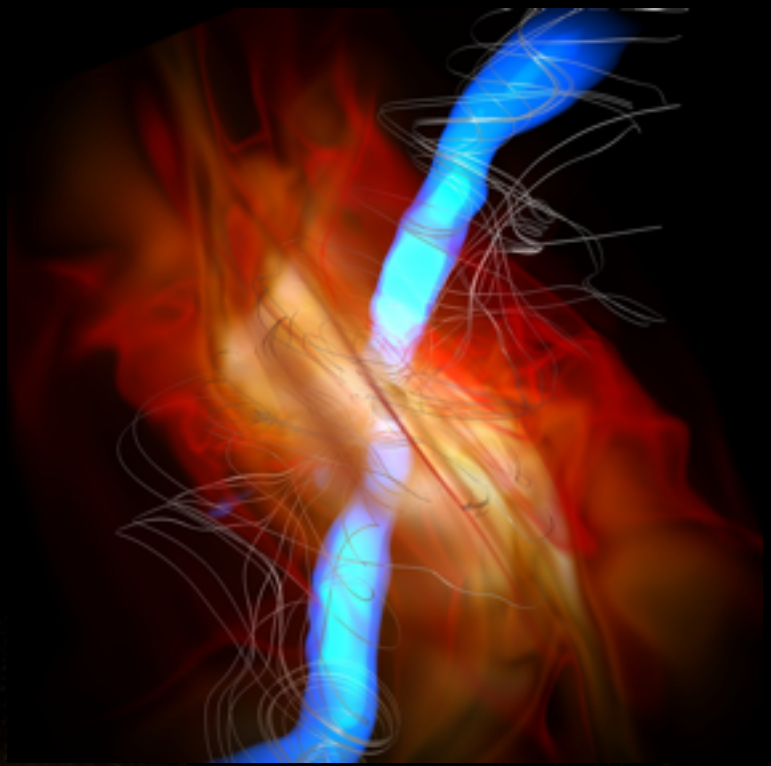
For each node at sufficient resolution:

- Render front faces of domain
- Instance of fragment shader for each covered pixel
  - Compute ray-direction
  - Sampling and color mapping

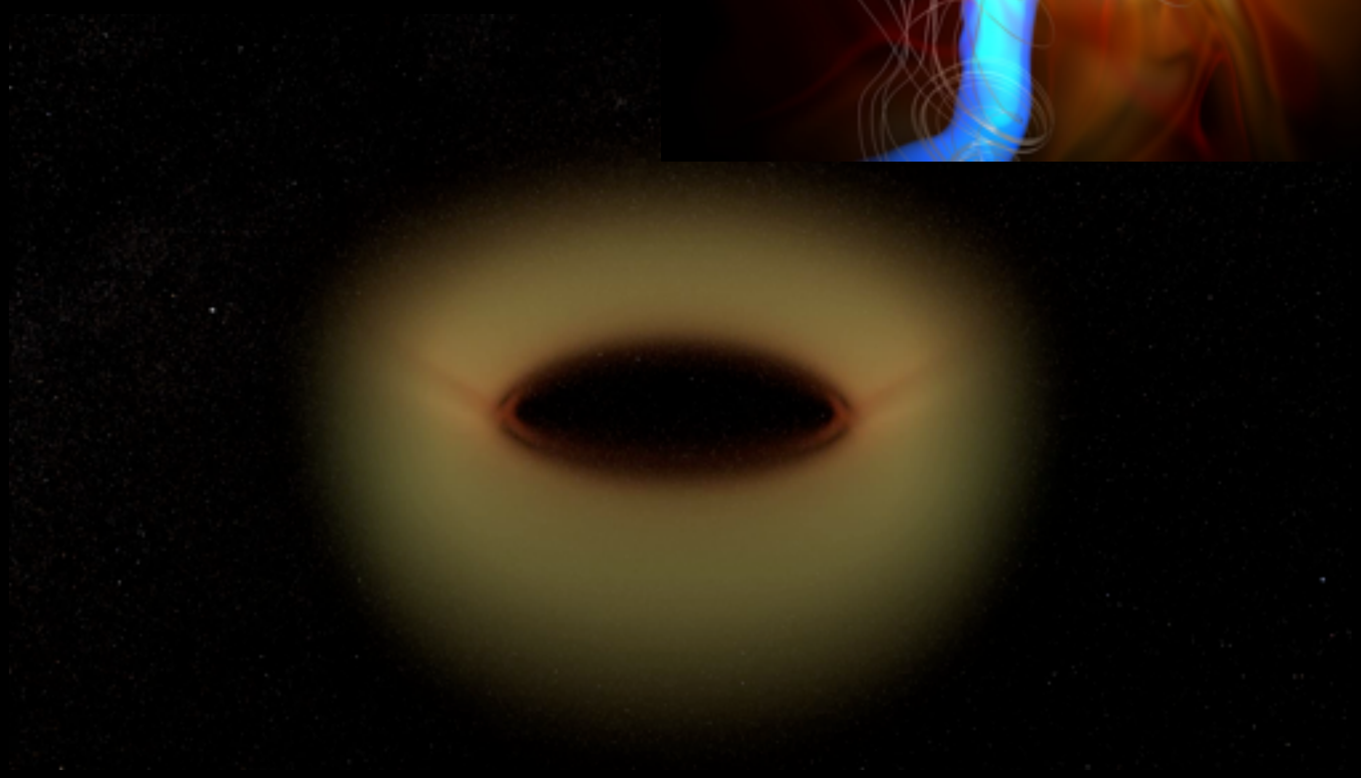
Display resulting pixel intensities

Kaehler, Wise, Abel, Hege [2008]

Examples



Simulation: John Wise (Georgia Tech), Tom Abel (KIPAC)



Simulation: Jonathan McKinney (UMD), Alexander Tchekhovsky (Princeton) and Roger Blandford (Stanford)

Simulation: Marcelo Alvarez (CITA), Tom Abel (KIPAC/Stanford)

AMR Volume Rendering on KIPAC's GPU-CLUSTER

4 MAC OS nodes

two NVIDIA 6800 cards each

12 rendering instances in parallel

	CPU	GPU
Texture-Based	• ray/geometry setup • traversal of kD-tree • LOD selection • access to whole domain • data I/O	• data sampling • blending
GPU-Raycasting I	• traversal of kD-tree • LOD selection • access to whole data domain • data I/O	• ray setup • data sampling • blending

	CPU	GPU
Texture-Based	• ray/geometry setup • traversal of kD-tree • LOD selection • access to whole domain • data I/O	• data sampling • blending
GPU-Raycasting I	• traversal of kD-tree • LOD selection • access to whole data domain • data I/O	• ray setup • data sampling • blending
GPU-Raycasting II	• data I/O	• traversal of kD-tree • LOD selection • access to whole data domain • ray setup • data sampling • blending

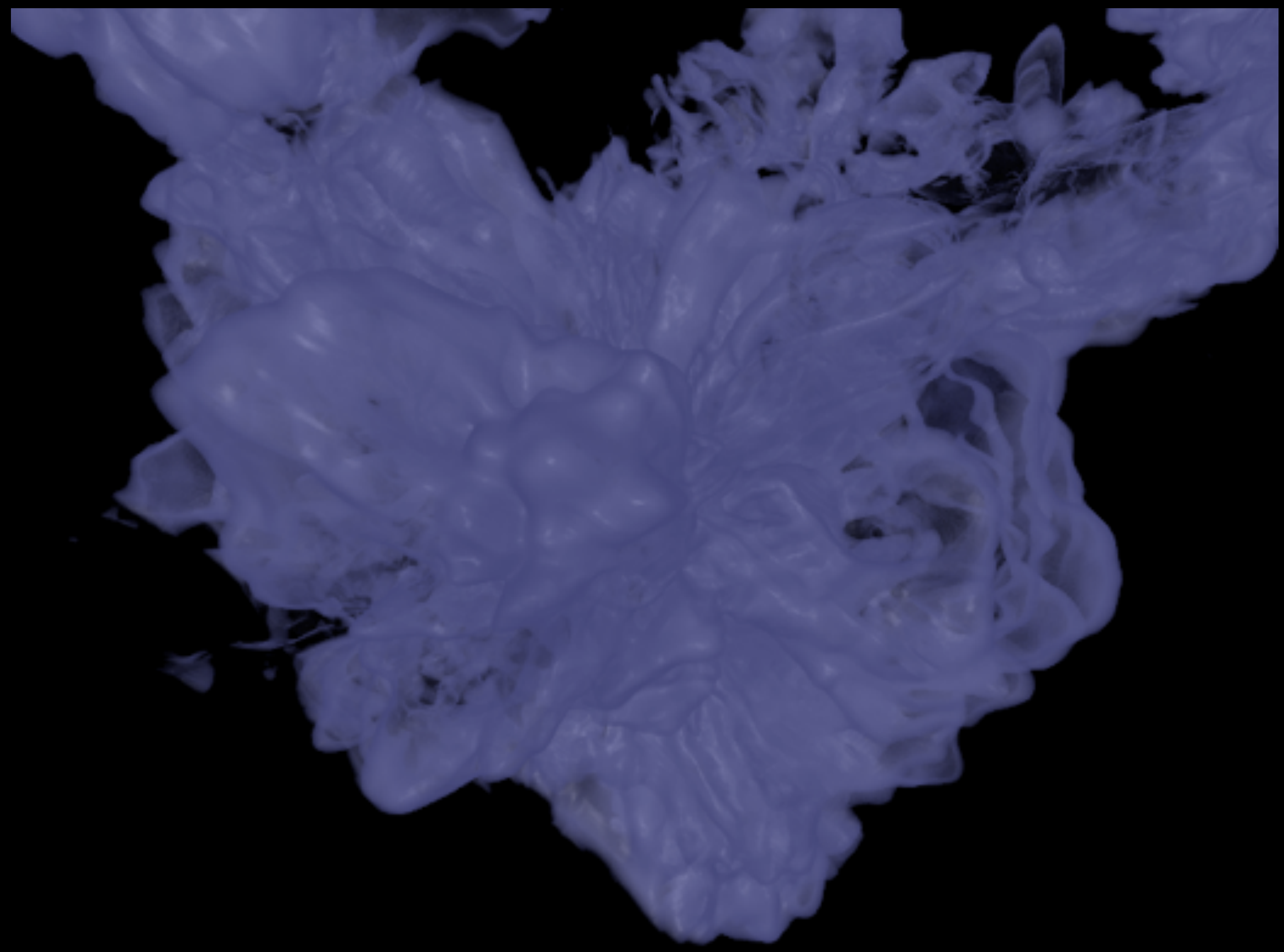
GPU-RAYCASTING FOR AMR DATA

complete hierarchy accessible in fragment shader

(not just single subgrids)

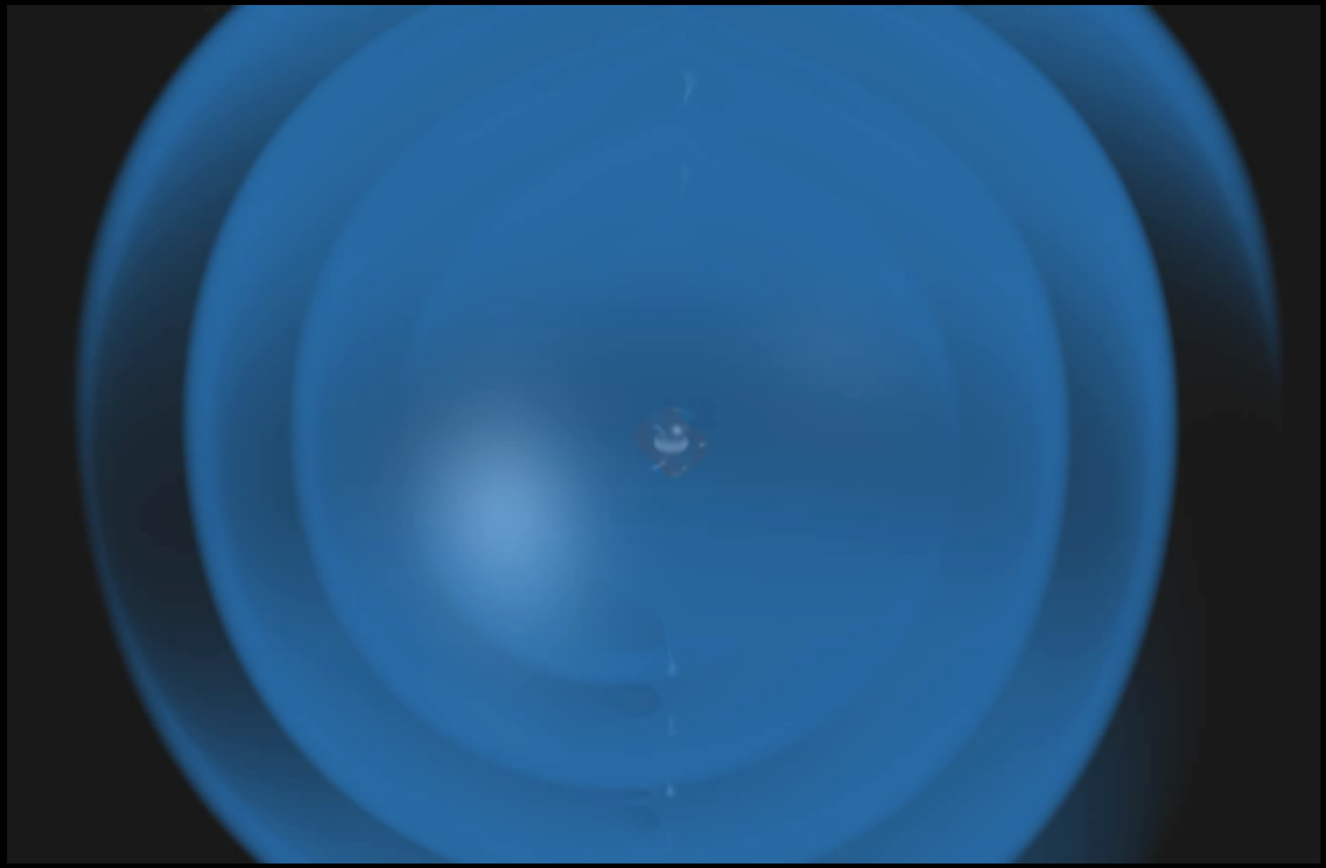
allows for advanced shading techniques

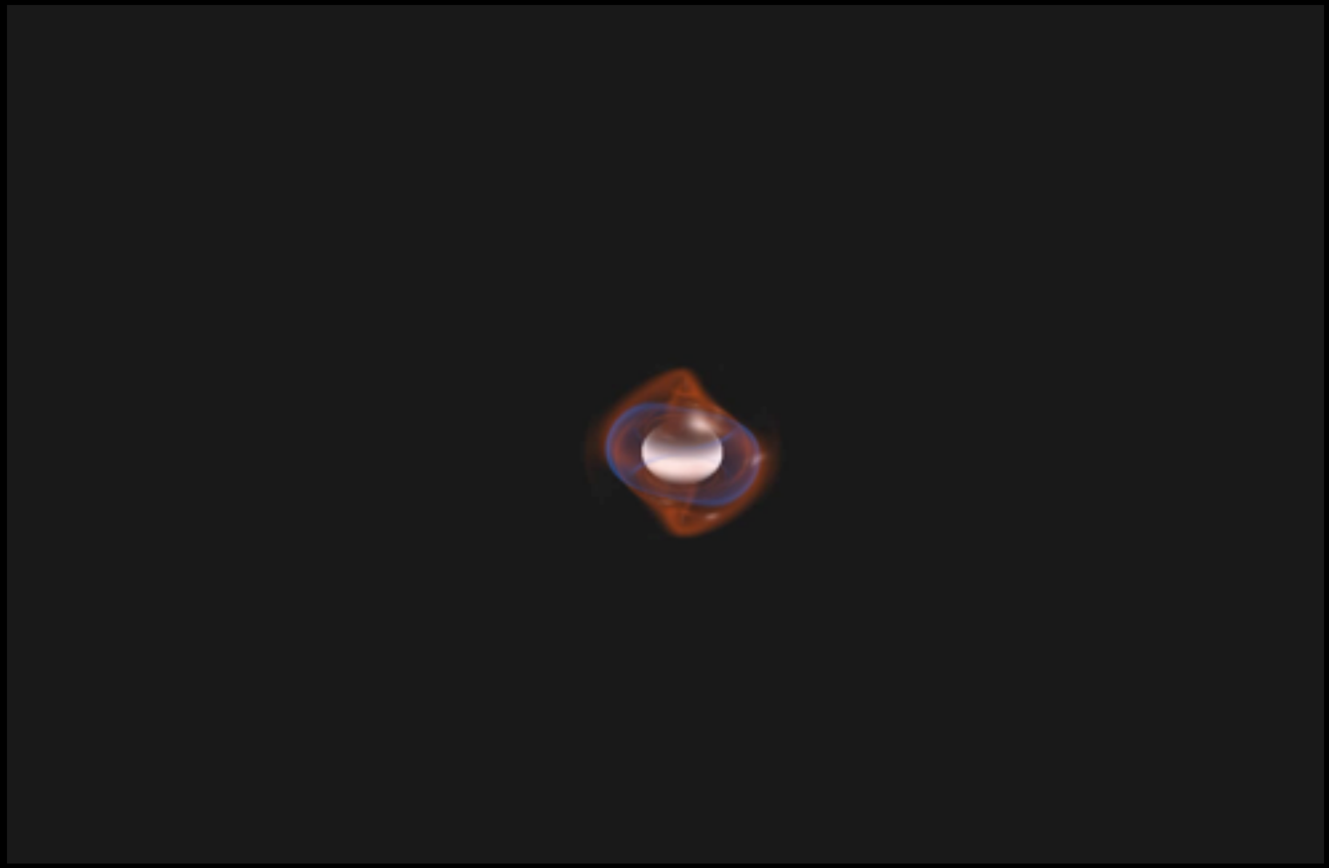
GRADIENT-BASED SHADING



On-the-fly Gradient-Computation
- reduced GPU memory requirements

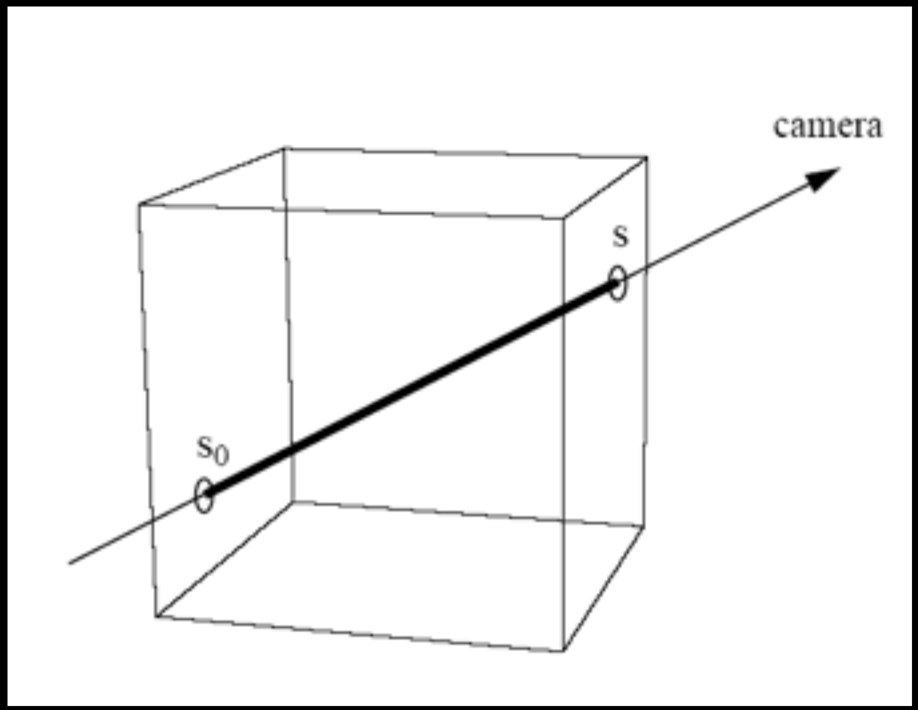
On-the-fly Gradient-Computation



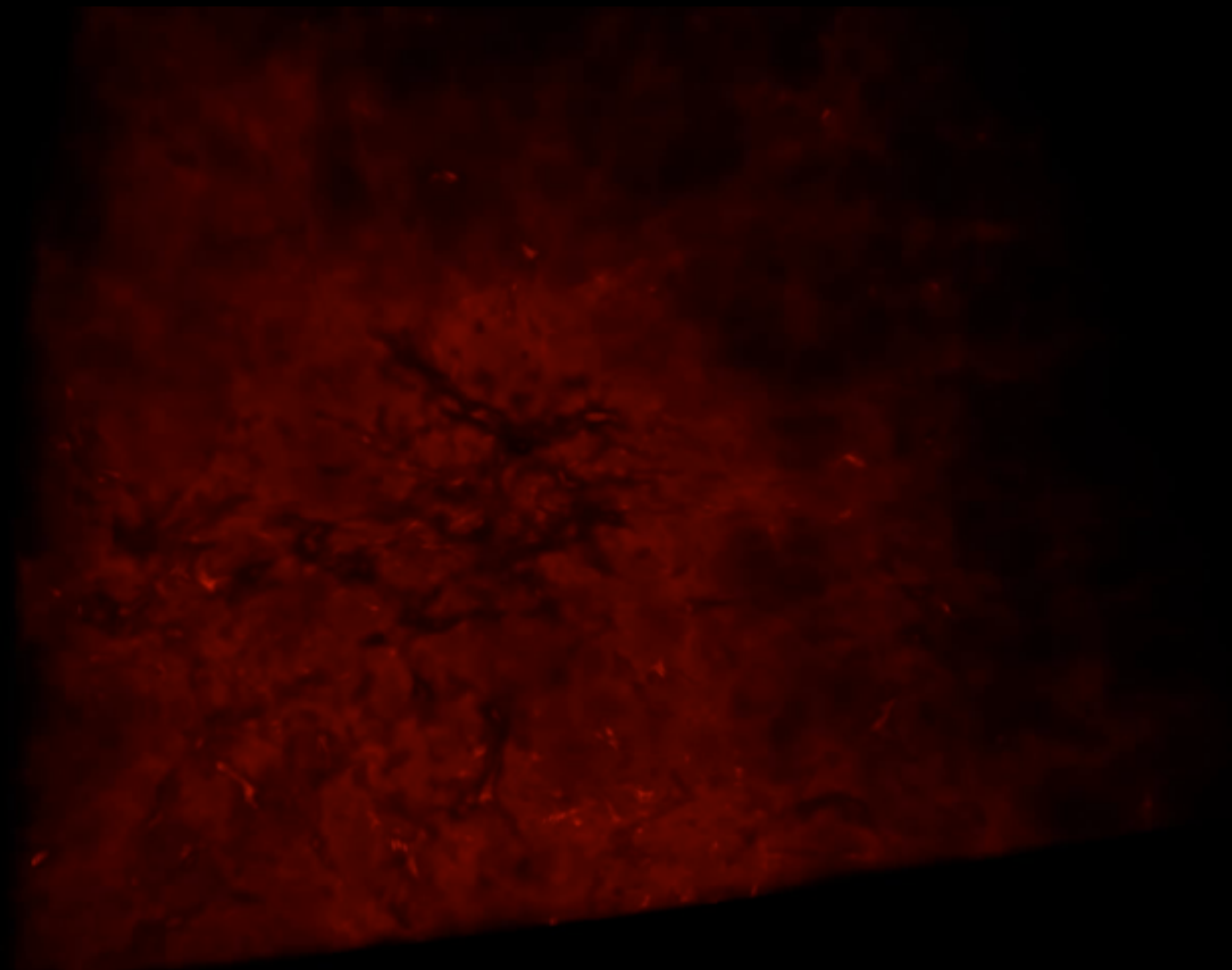


Simulation: William East (KIPAC)

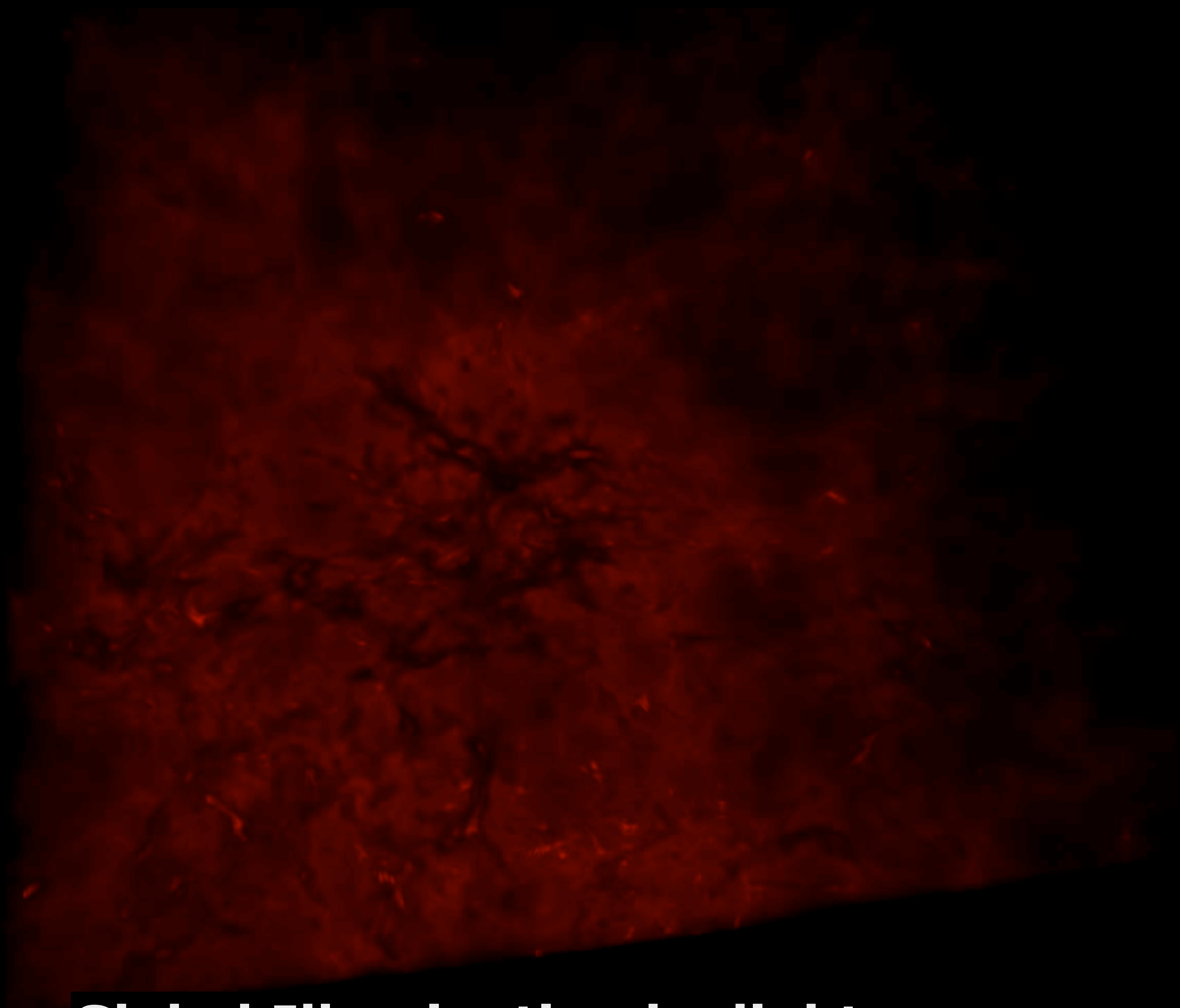
Assumption: no scattering



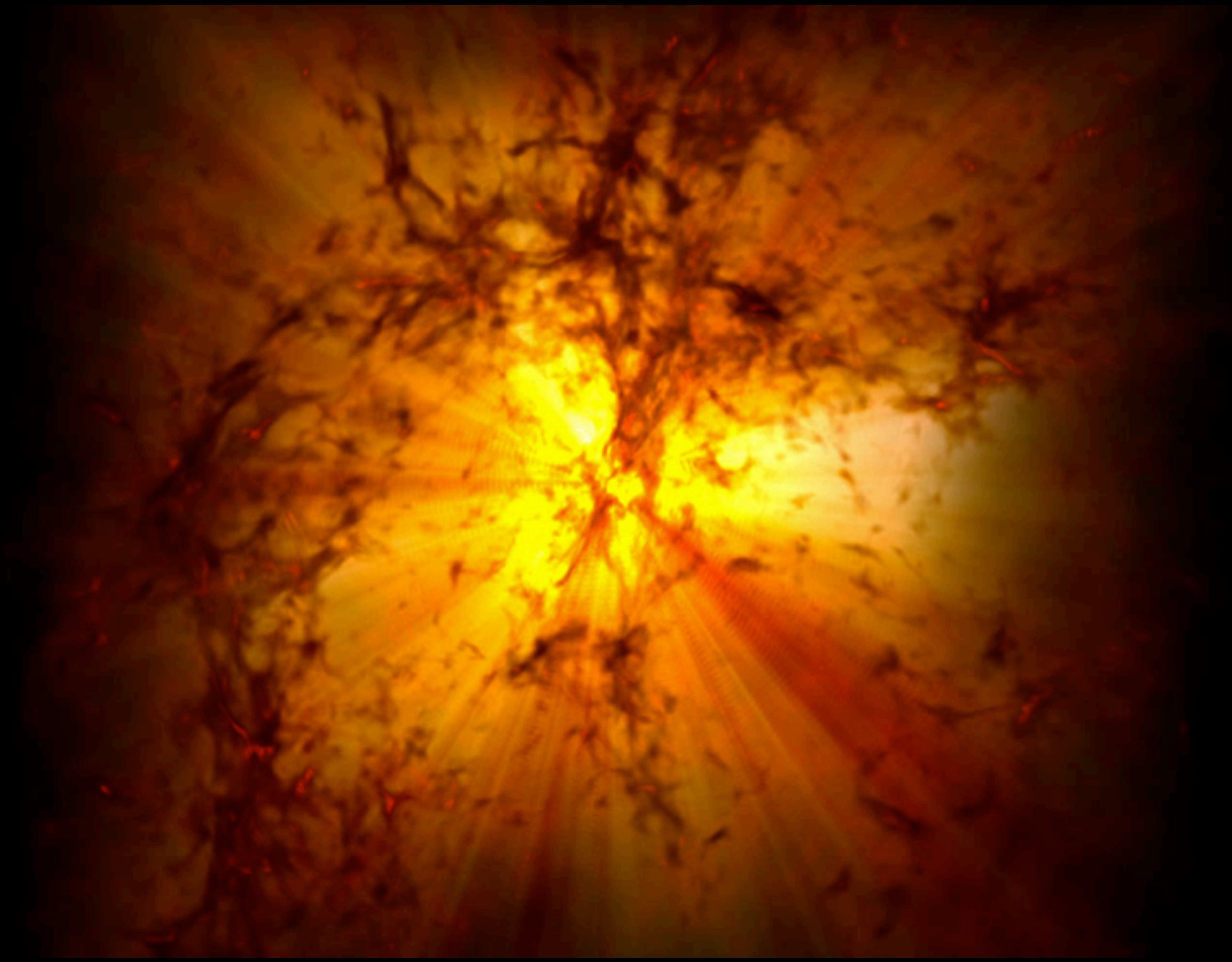
GLOBAL ILLUMINATION FOR AMR DATA



One Snapshot from Enzo Simulation by Wise and Abel.
100,000 subgrids
1 billion cells



Global Illumination by light source computed on-the-fly in fragment shader



Thanks for your attention !