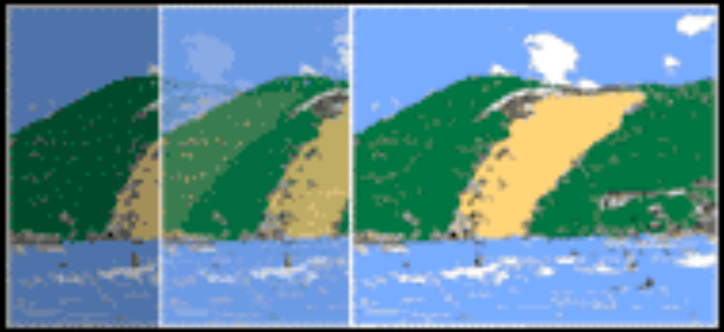
GPU-Based Volume Rendering of Unstructured Grids

Module 3:

Isosurface Techniques

João L. D. Comba

UFRGS



SIBGRAPI 2005

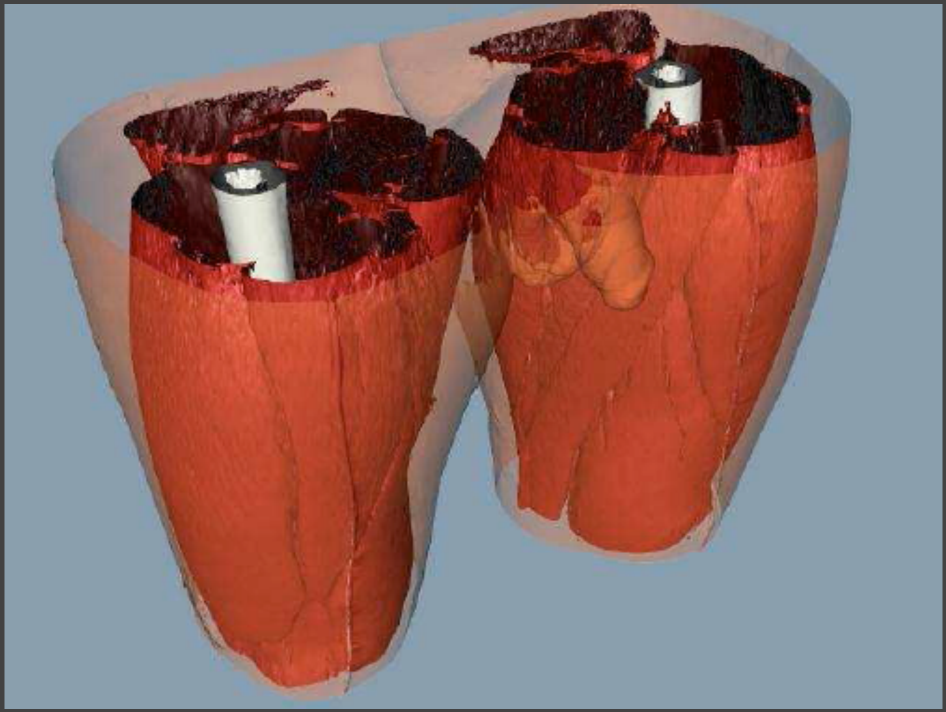
Natal - RN - Brazil

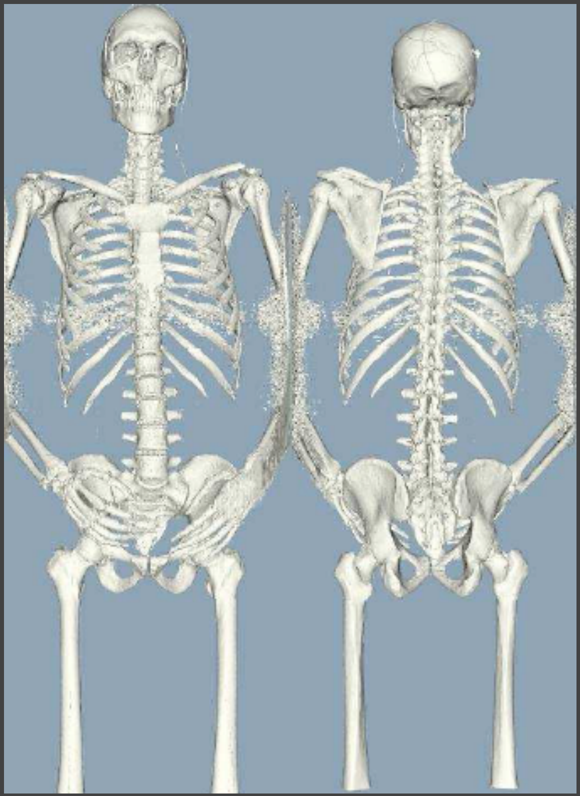
XVIII Brazilian Symposium on Computer Graphics and Image Processing

Isosurfaces

SIBGRAPI 2005

Isosurface: surface with the same scalar field



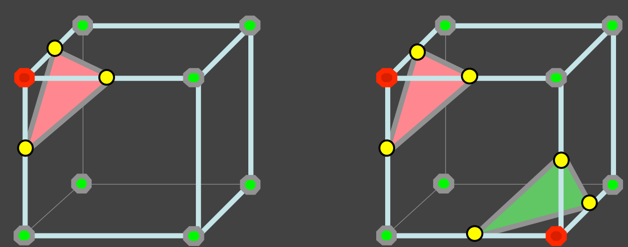


[Lorensen 95] Marching Through the Visible Man

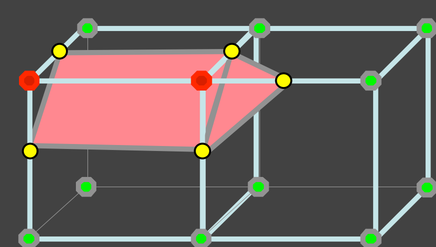
SIBGRAPI 2005

Marching Cubes

- Isosurface extraction from voxels [Lorensen 87]
  - Assumes Linear Interpolation between data
  - Corners are marked as inside/outside surface



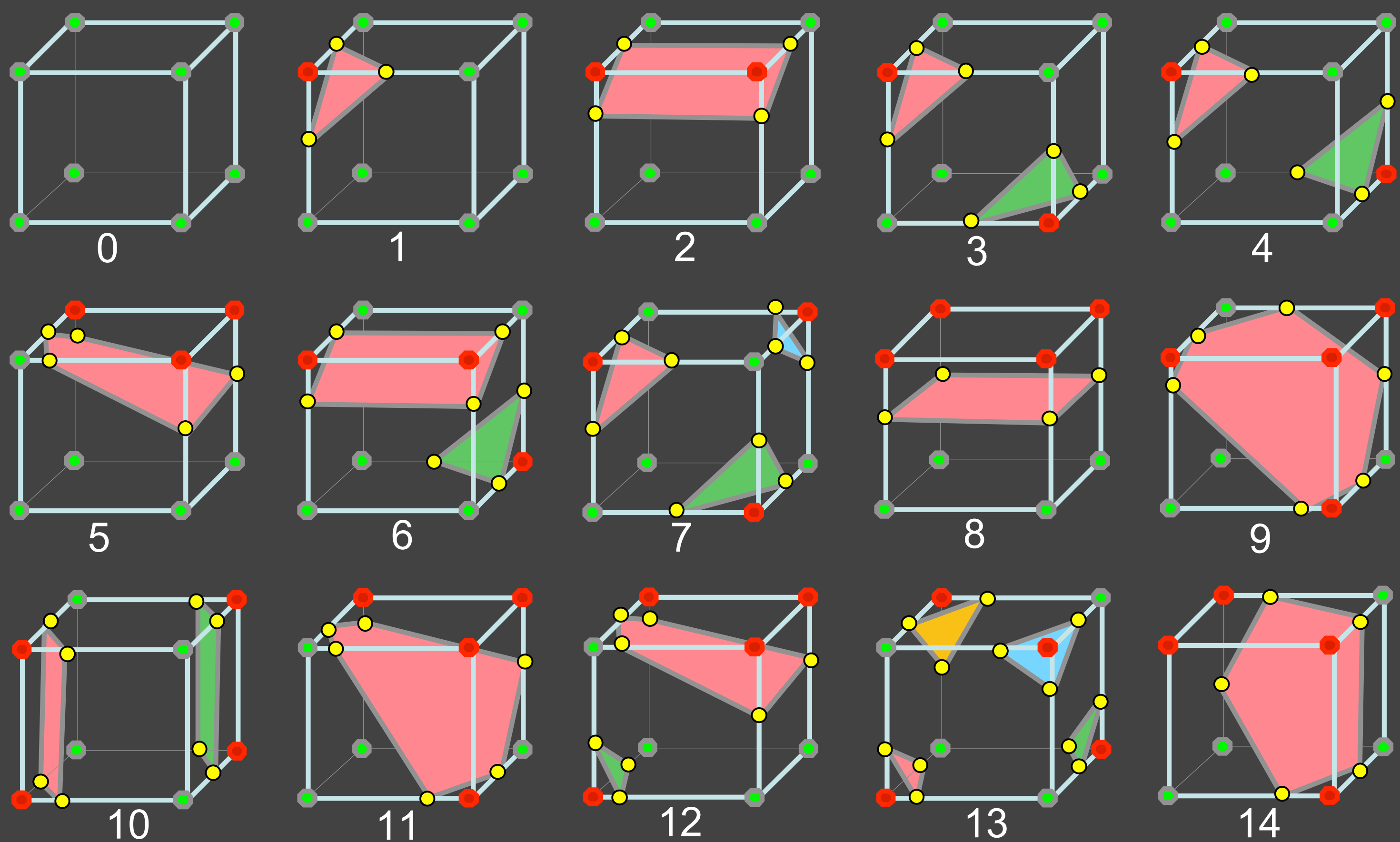
- Mesh extraction
  - Connect surface intersections



- Render Mesh using traditional techniques

SIBGRAPI 2005

Marching Cubes



Marching Cubes

- Create table with all configurations
  - Number of triangles
  - Connectivity
- For each voxel:
  - Identify configuration using table
  - Compute triangles and connect with adjacent voxels triangles

Marching Cubes

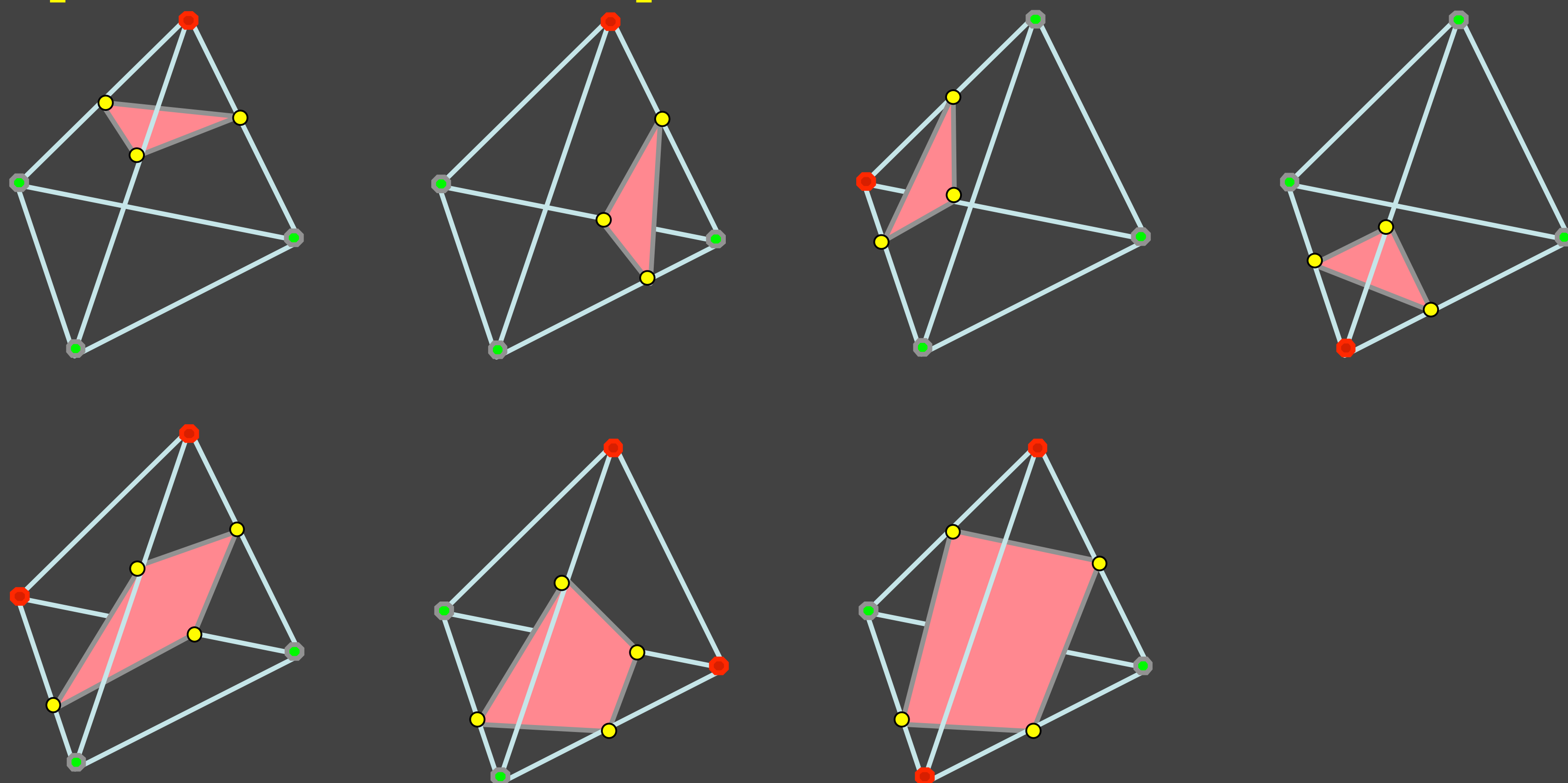
- Create table with all configurations
  - Number of triangles
  - Connectivity
- For each voxel:

Costly

  - Identify configuration using table
  - Compute triangles and connect with adjacent voxels triangles

Marching Tetrahedra

- [Doi an Koide 91]



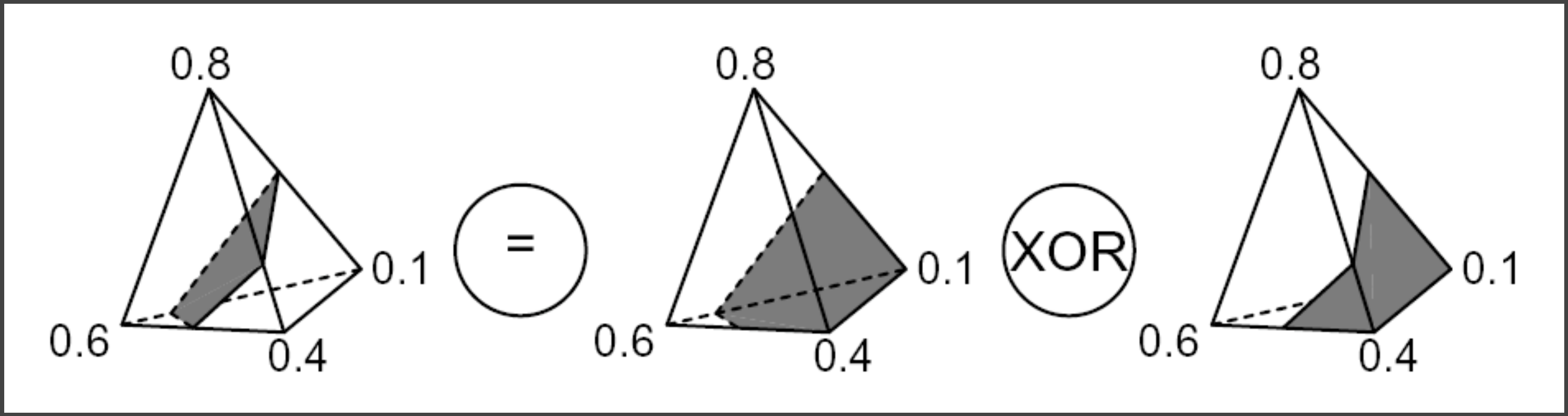
SIBGRAPI 2005

HW Accelerated Isosurface based on Cell Projection

- [Röttger et al] Vis 00

HW Accelerate Marching Cells Algorithm

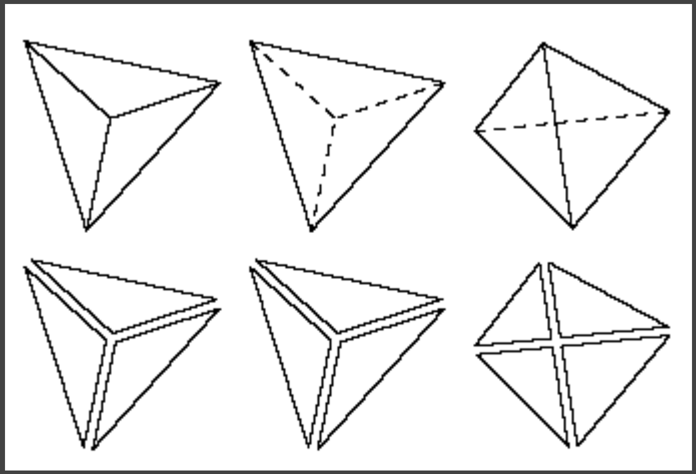
- Computing Isosurfaces using XOR [Westerman 98]

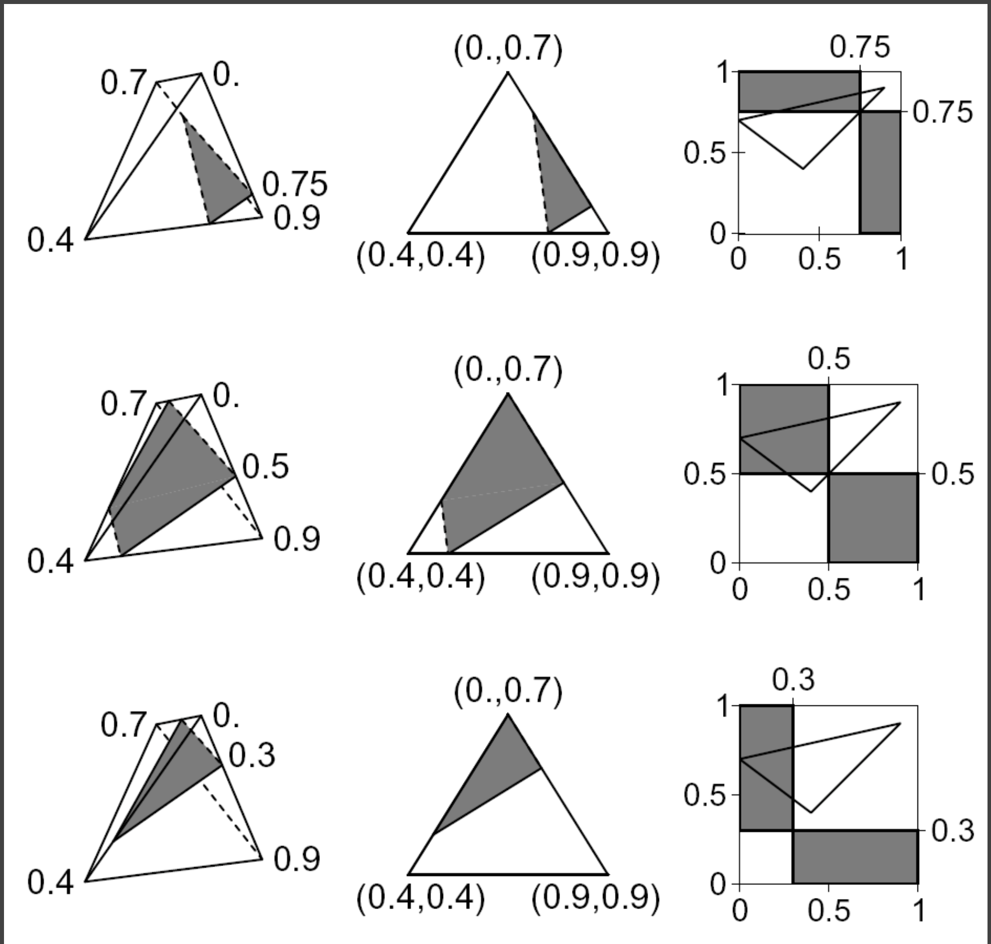


Back faces

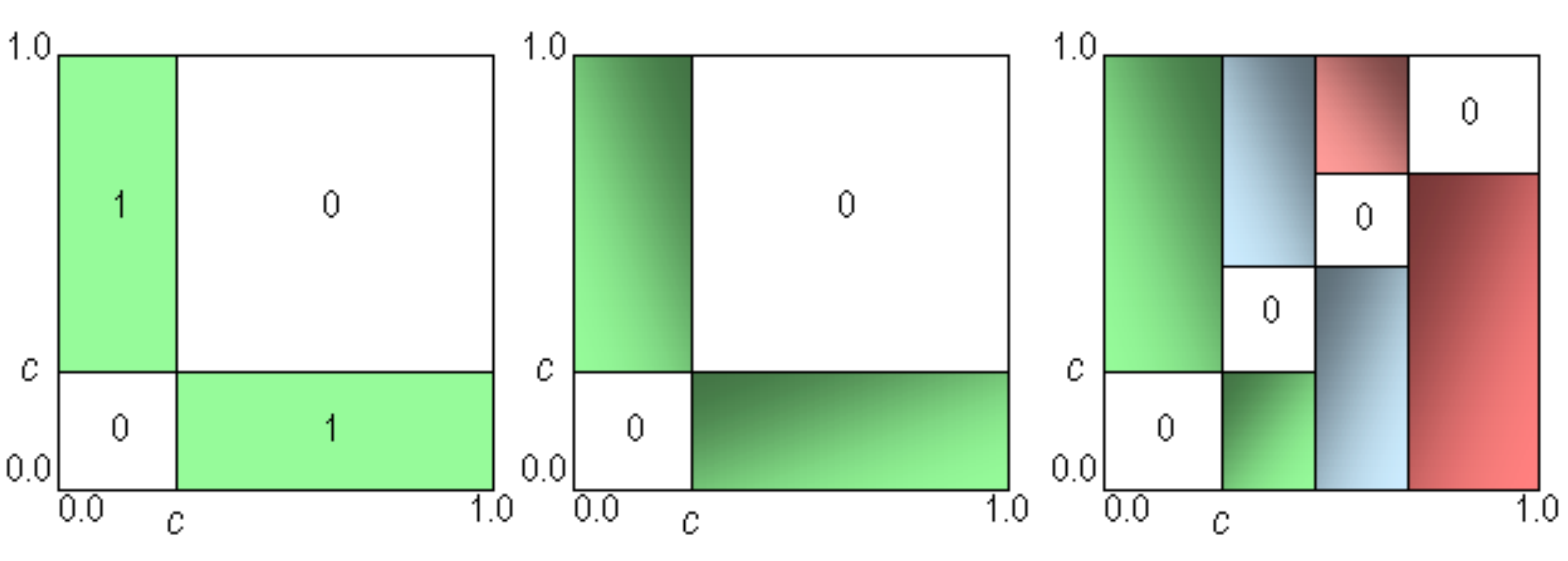
Front faces

Projected Tetra with flat-shaded isosurfaces



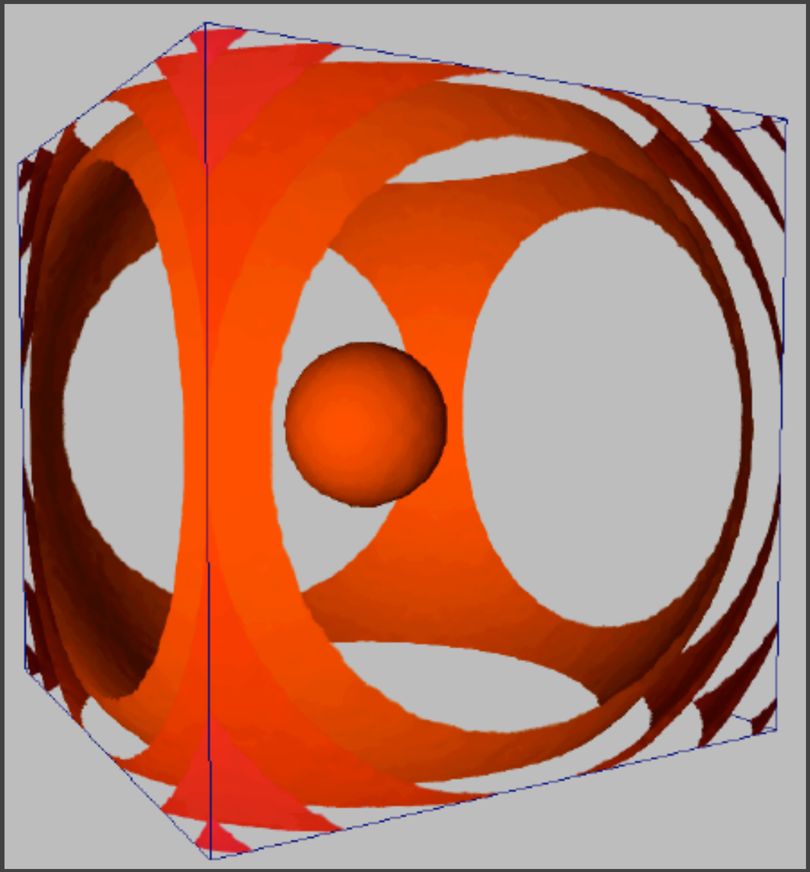


Smoothly Shaded and Multiple Isosurfaces

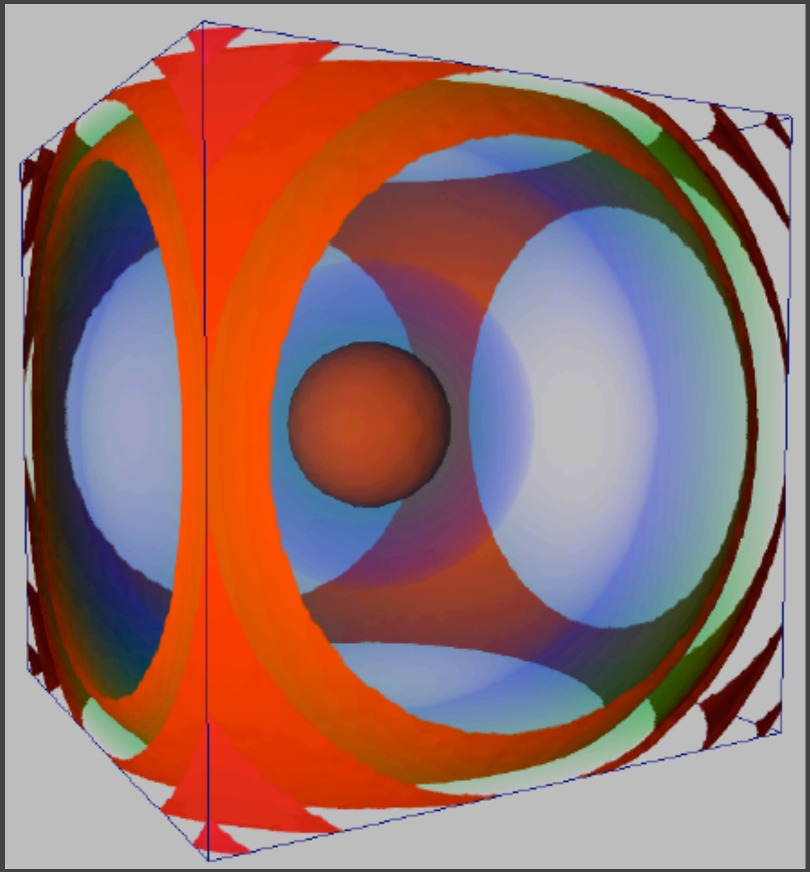


SIBGRAPI 2005

Results

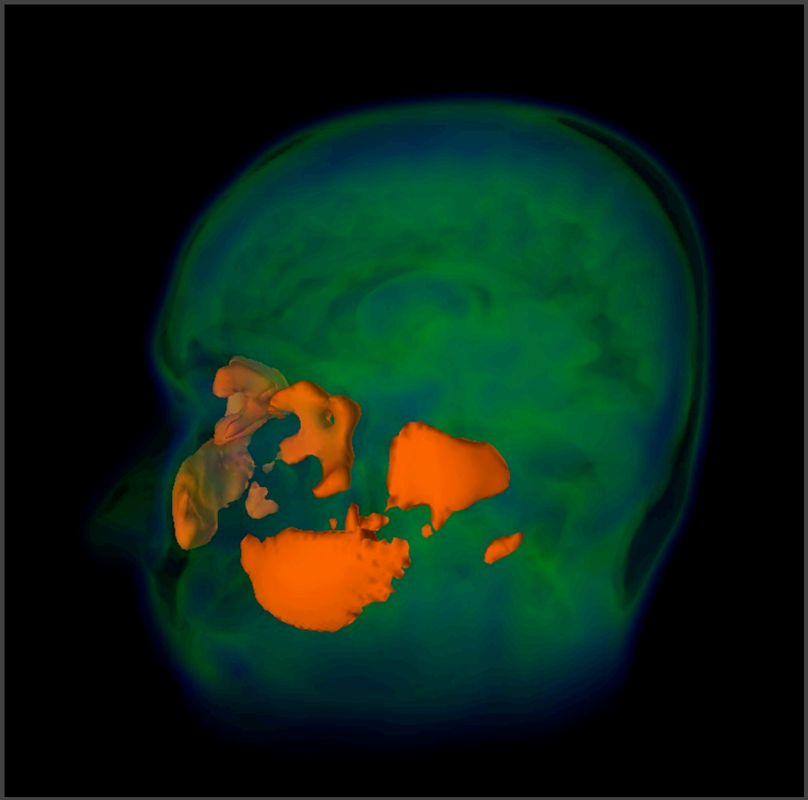


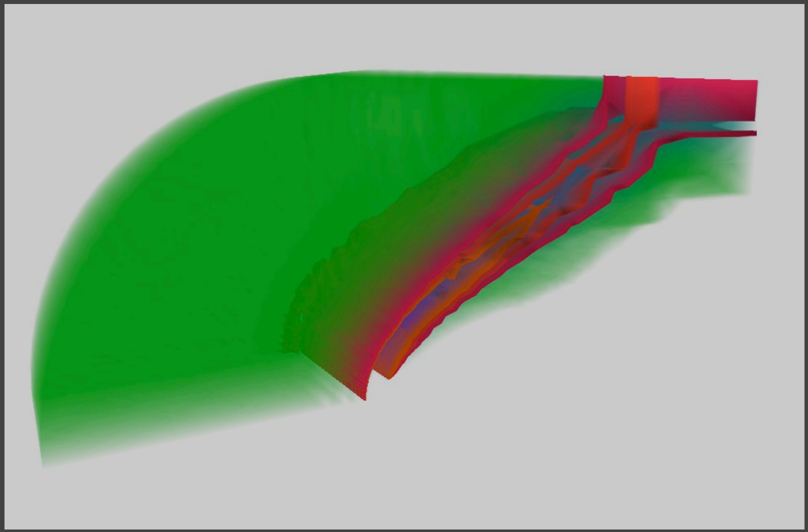
Mulitple Isosurfaces



Smoothly-Shaded Isosurfaces

Results

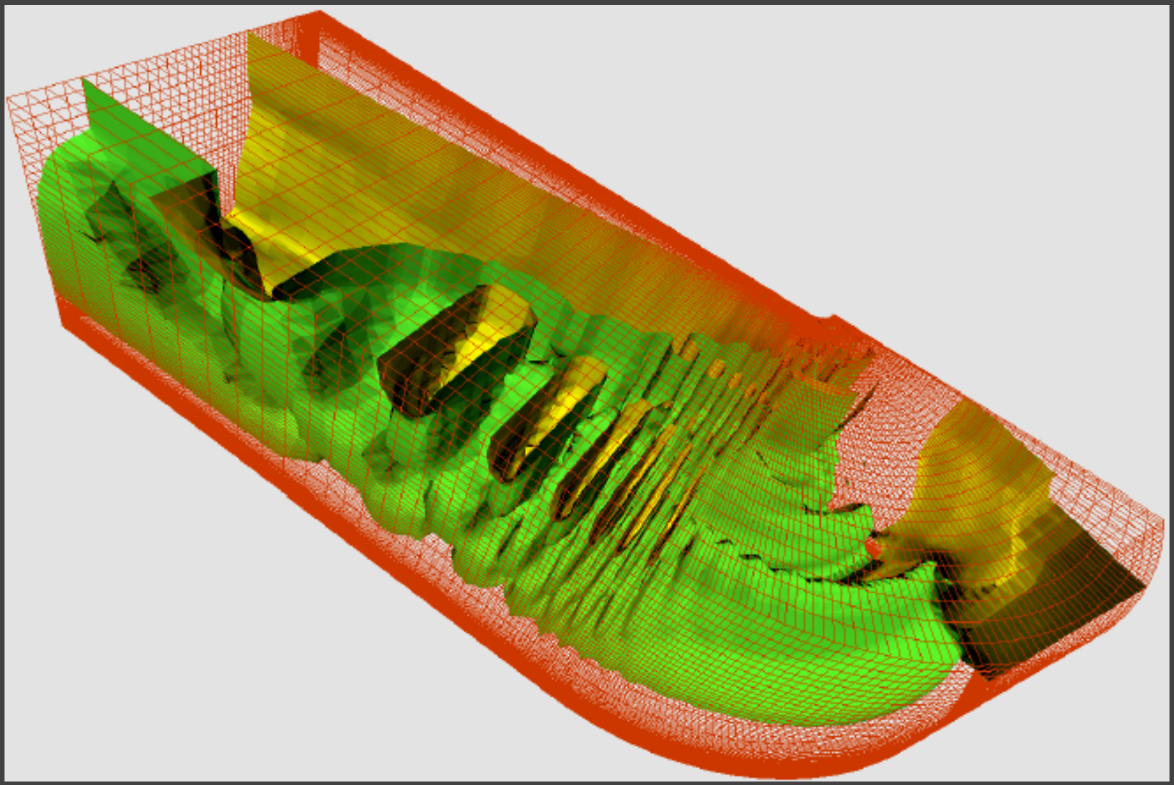




Isosurface Computation using Vertex Programs

SIBGRAPI 2005

- [Pascucci 2004] Sym. Vis 2004

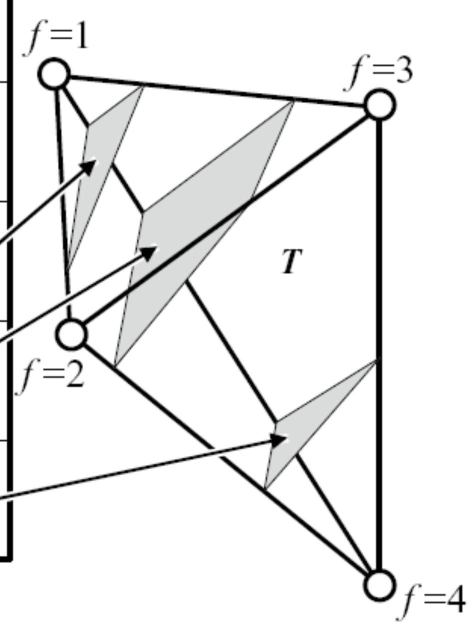


Isosurface Computation using Vertex Programs

SIBGRAPI **2005**

- Possible Isosurfaces by Linear Interpolation

isosurface isovalue	geometric primitive
$f = 0$	empty
$f = 1.8$	triangle
$f = 2.5$	quad
$f = 3.7$	triangle



Isosurface Computation using Vertex Programs

SIBGRAPI 2005

- Rendering Step:

```
set_global_parameters();
set_isovalue();
glBegin(GL_QUADS);          // Start drawing quads
for i=0 to num_tets do:
    set_tet_parameters(i);  // Store vertices in registers
    glVertex2b(0,0);        // Run program four times
    glVertex2b(1,1);        //   with v[OPOS].x set
    glVertex2b(2,2);        //   successively to 0,1,2,3.
    glVertex2b(3,3);
glEnd();                    // Stop drawing quads
```

SIBGRAPI 2005

Isosurface Computation using Vertex Programs

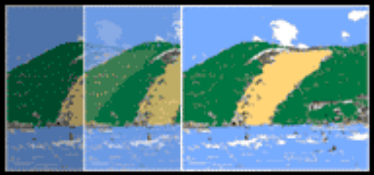
- Vertex Program Registers:

Edge/Vertex Table

Const. Reg.	Edge Selection	V0	V1	V2	V3	Vertex Selection
70	E0 start	1	0	0	0	V0
71	E0 end	0	1	0	0	V1
72	E1 start	0	0	1	0	V2
73	E1 end	0	0	0	1	V3
74	E2 start	1	0	0	0	
75	E2 end	0	0	0	1	
76	E3 start	0	1	0	0	
77	E3 end	0	0	1	0	
78	E4 start	0	1	0	0	
79	E4 end	0	0	0	1	
80	E5 start	1	0	0	0	
81	E5 end	0	0	1	0	

Isosurface Intersection Table.

Const. Reg.	Edge 0	Edge 1	Edge 2	Edge 3	Interp. case
40	70	70	70	70	0
41	74	80	78	78	1
42	80	76	80	80	10
43	74	80	76	78	11
44	70	76	78	78	100
45	70	76	80	74	101
46	70	80	80	78	110
47	70	80	74	74	111
48	80	70	74	74	1000
49	70	78	80	80	1001
50	70	74	80	76	1010
51	76	70	78	78	1011
52	74	78	76	80	1100
53	76	80	80	80	1101
54	80	74	78	78	1110
55	70	70	70	70	1111



SIBGRAPI 2005

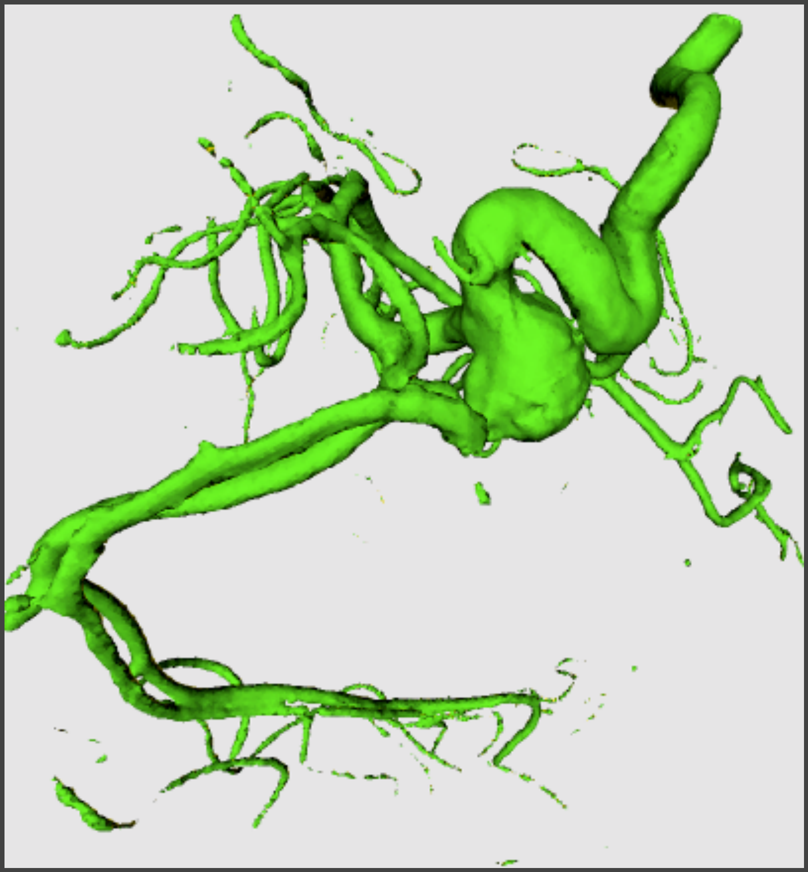
Isosurface Computation using Vertex Programs

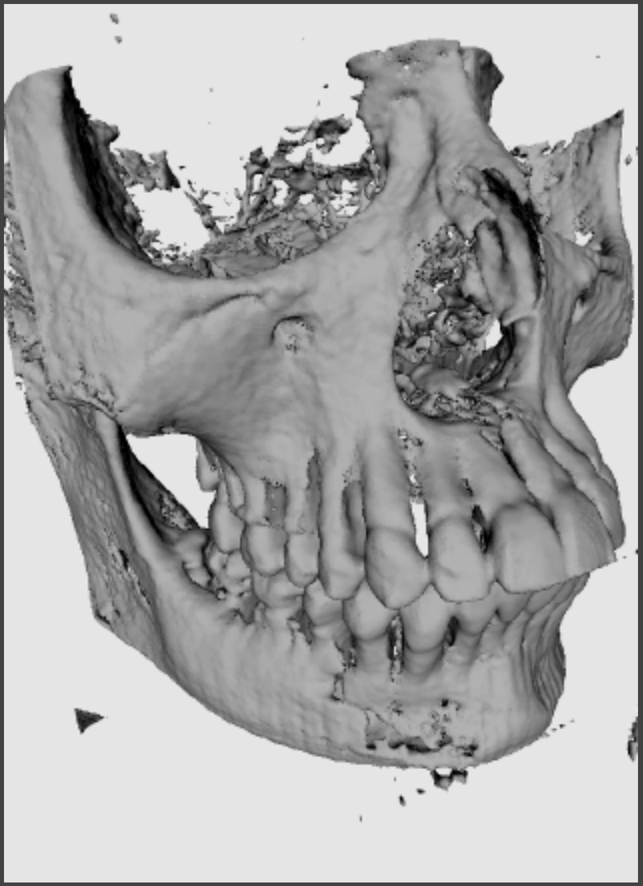
- Vertex Program Registers:

Edge/Vertex Table						
Const. Reg.	Edge Selection	V0	V1	V2	V3	Vertex Selection
70	E0 start	1	0	0	0	V0
71	E0 end	0	1	0	0	V1
72	E1 start	0	0	1	0	V2
73	E1 end	0	0	0	1	V3
74	E2 start	1	0	0	0	
75	E2 end	0	0	0	1	
76	E3 start	0	1	0	0	
77	E3 end	0	0	1	0	
78	E4 start	0	1	0	0	
79	E4 end	0	0	0	1	
80	E5 start	1	0	0	0	
81	E5 end	0	0	1	0	

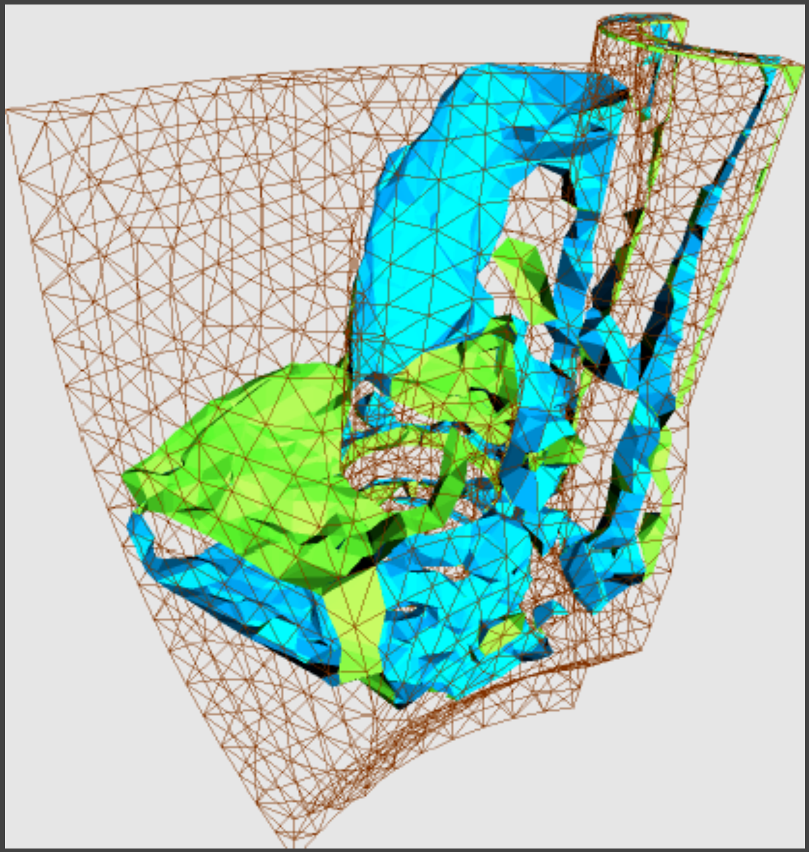
Isosurface Intersection Table.					
Const.	Edge				Interp.
Reg.	0	1	2	3	case
40	70	70	70	70	0
41	74	80	78	78	1
42	80	76	80	80	10
43	74	80	76	78	11
44	70	76	78	78	100
45	70	76	80	74	101
46	70	80	80	78	110
47	70	80	74	74	111
48	80	70	74	74	1000
49	70	78	80	80	1001
50	70	74	80	76	1010
51	76	70	78	78	1011
52	74	78	76	80	1100
53	76	80	80	80	1101
54	80	74	78	78	1110
55	70	70	70	70	1111

Results



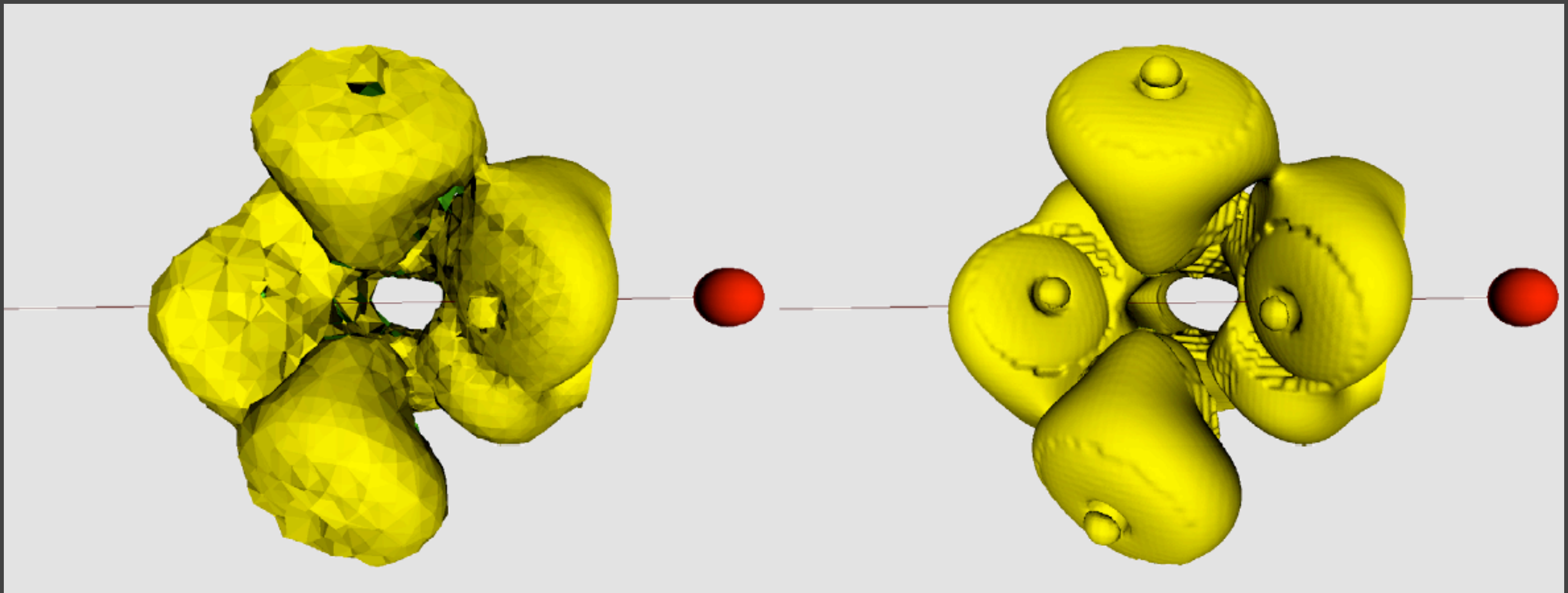


Results



Results

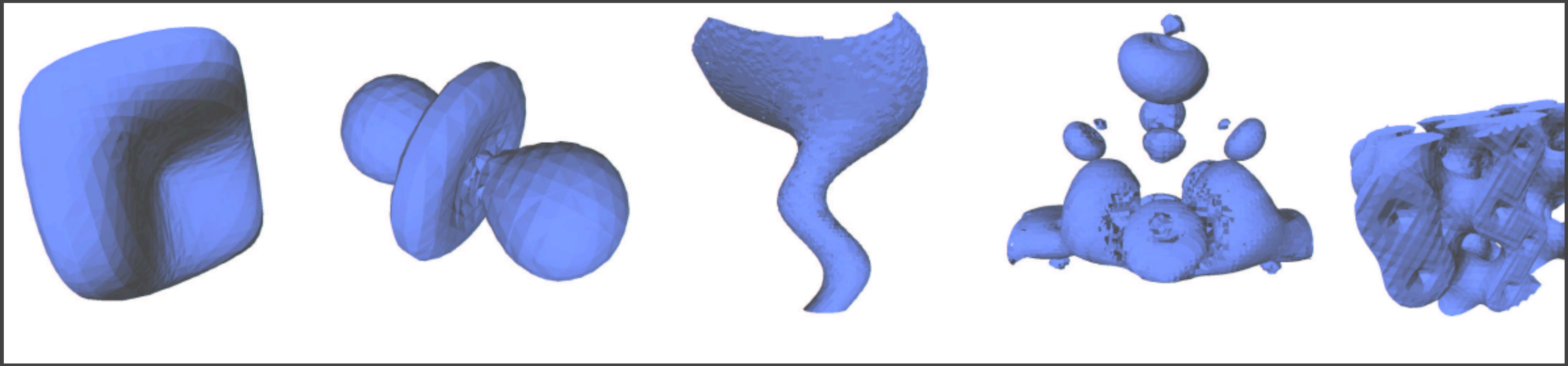
- View-dependent refinement



Isosurface Computation using Fragment Programs

SIBGRAPI 2005

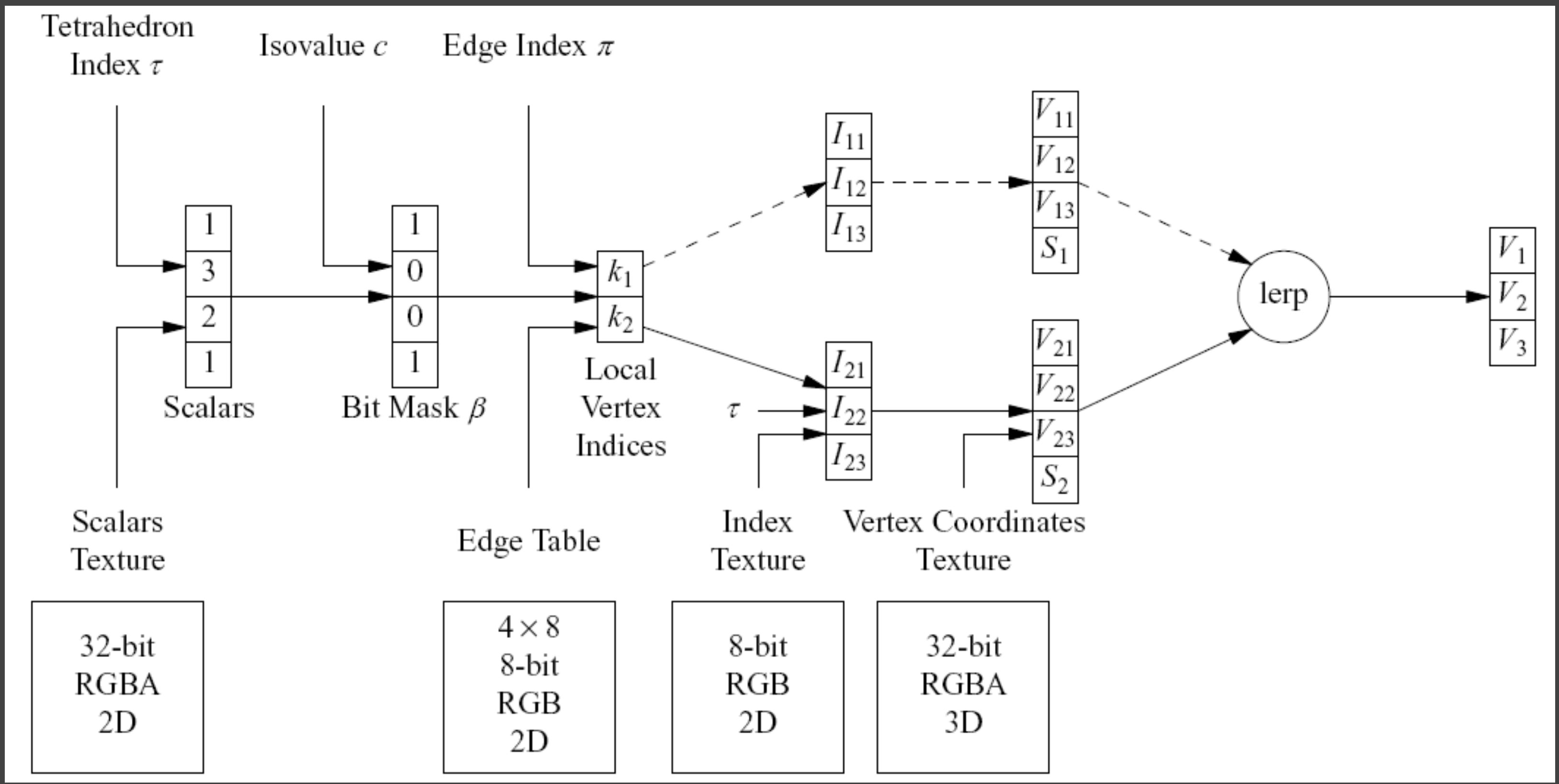
- [Klein et al 2004] Pacific Graphics



Isosurface Computation using Fragment Programs

SIBGRAPI 2005

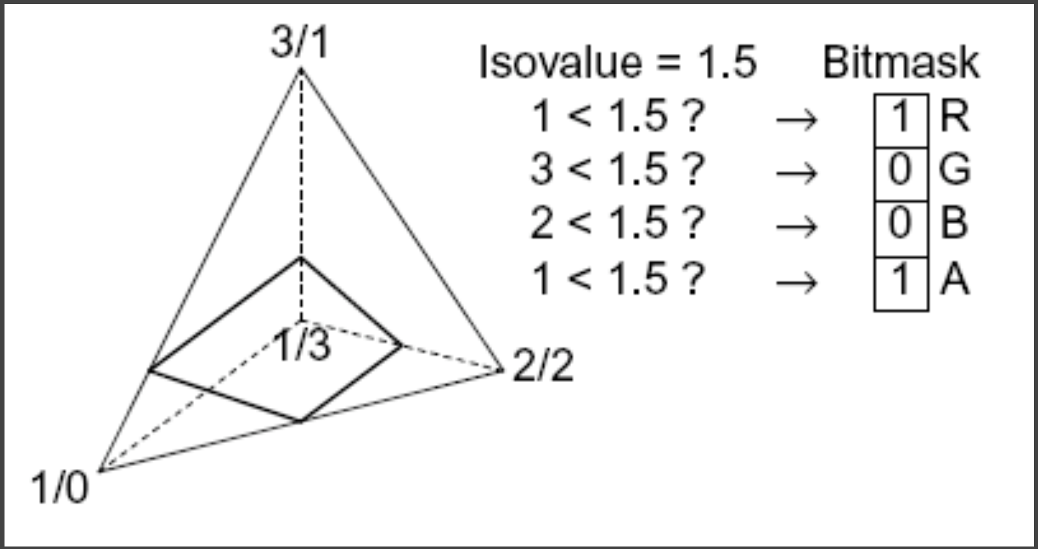
- System Overview



Isosurface Computation using Fragment Programs

SIBGRAPI 2005

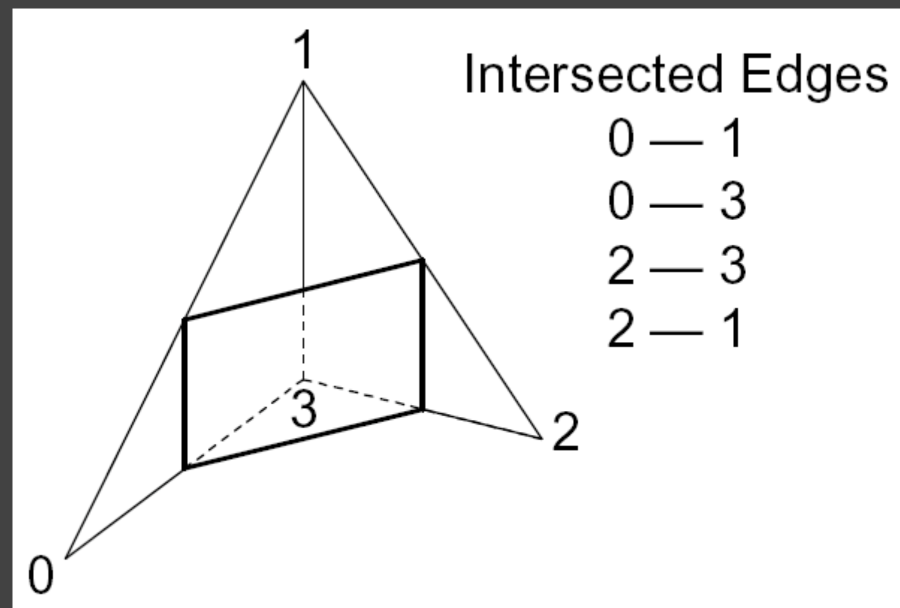
- Tetrahedron Classification



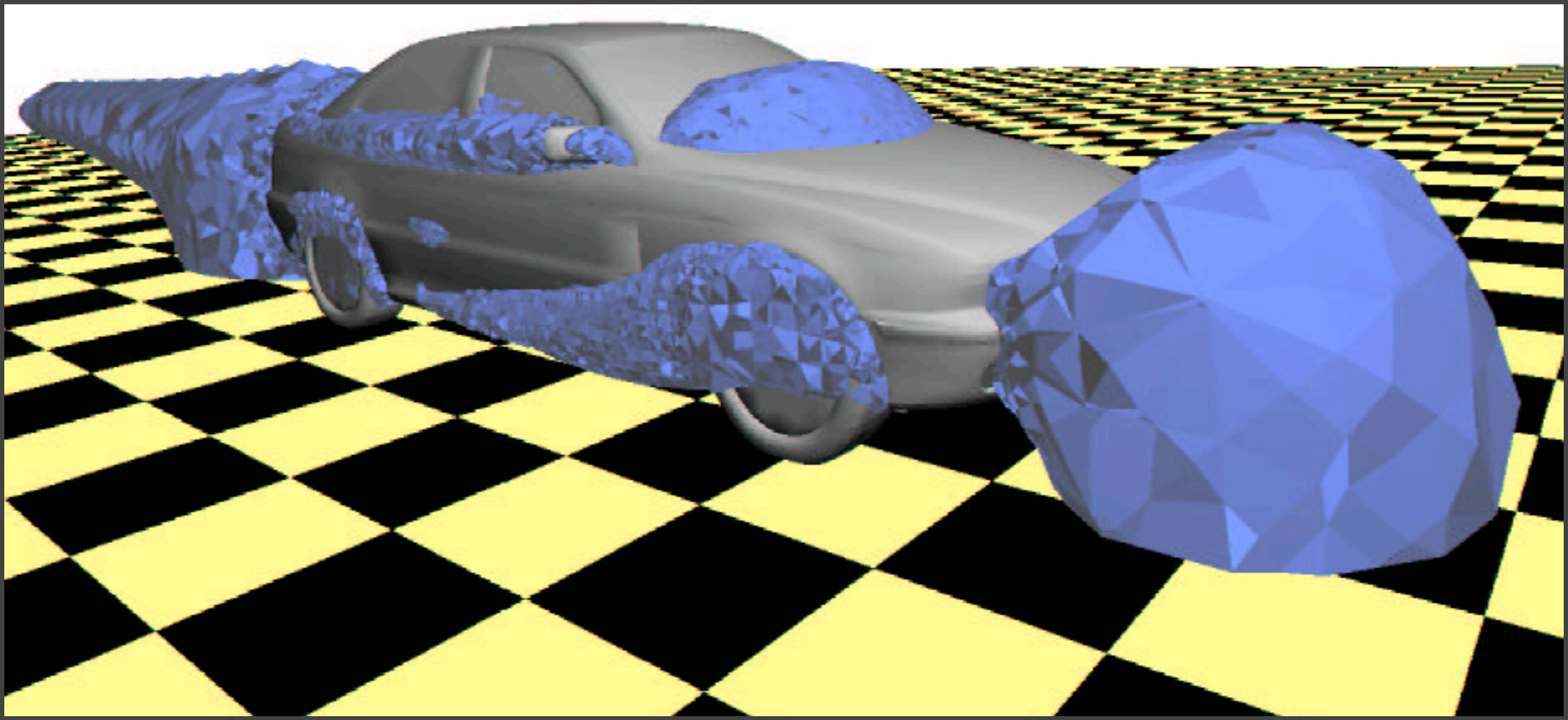
Isosurface Computation using Fragment Programs

- Edge Table

	π							
β^T	0		1		2		3	
(0,0,0,0)	0	0	0	0	0	0	0	0
(0,0,0,1)	3	0	3	1	3	2	3	2
(0,0,1,0)	2	3	2	0	2	1	2	1
(0,1,0,0)	1	2	1	3	1	0	1	0
(1,0,0,0)	0	1	0	2	0	3	0	3
(0,0,1,1)	0	2	0	3	1	3	1	2
(0,1,0,1)	0	1	0	3	2	3	2	1
(0,1,1,0)	1	0	1	3	2	3	2	0

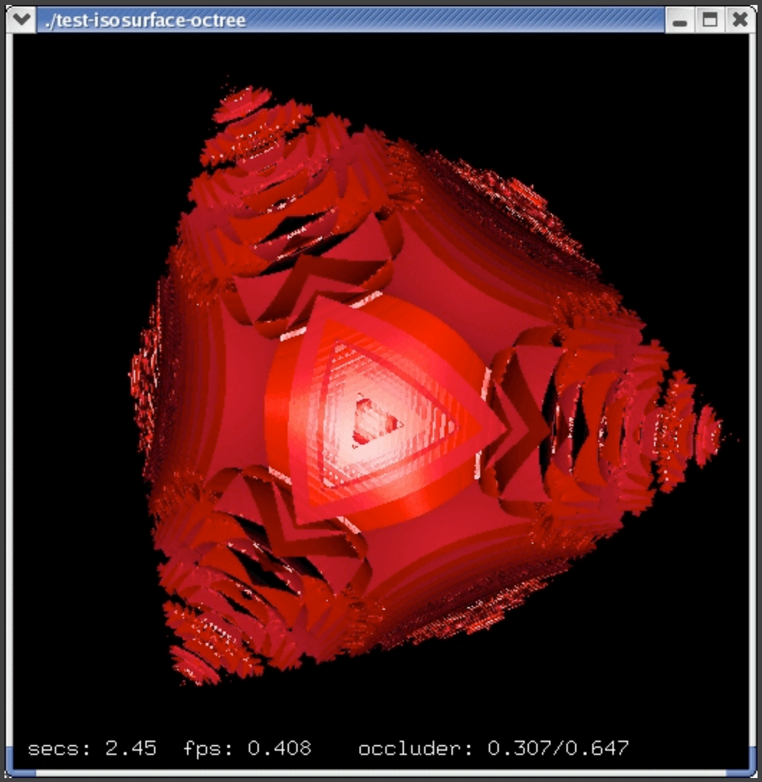


Isosurface Computation using Fragment Programs



Implicit Occluders

- [Pesco et al] VolVis 2004



SIBGRAPI 2005

Implicit Occluder

Motivation:

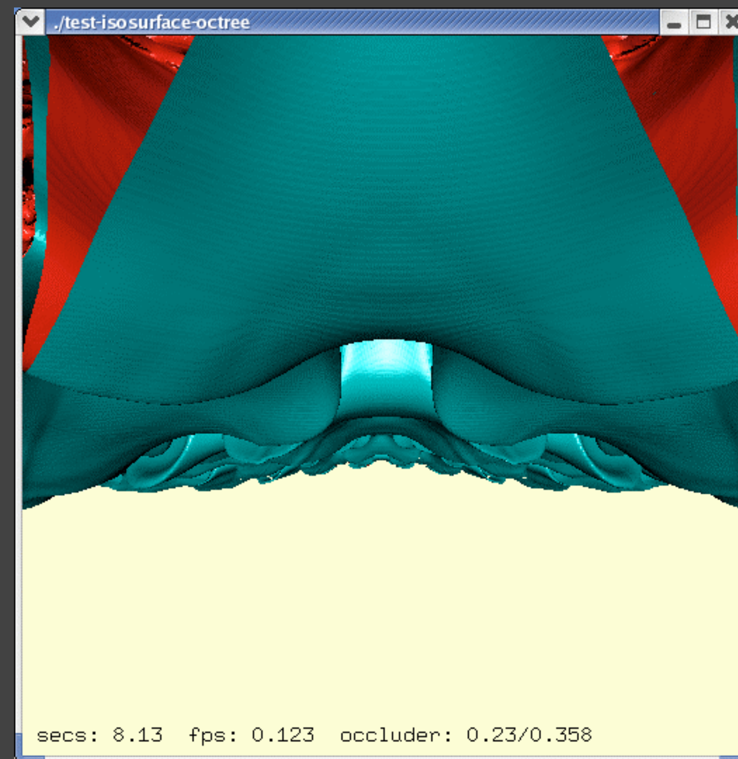
- Exploration of large data sets using isosurfaces

Challenges:

- Isosurfaces can be larger than the original volume
- Often too large to keep in memory

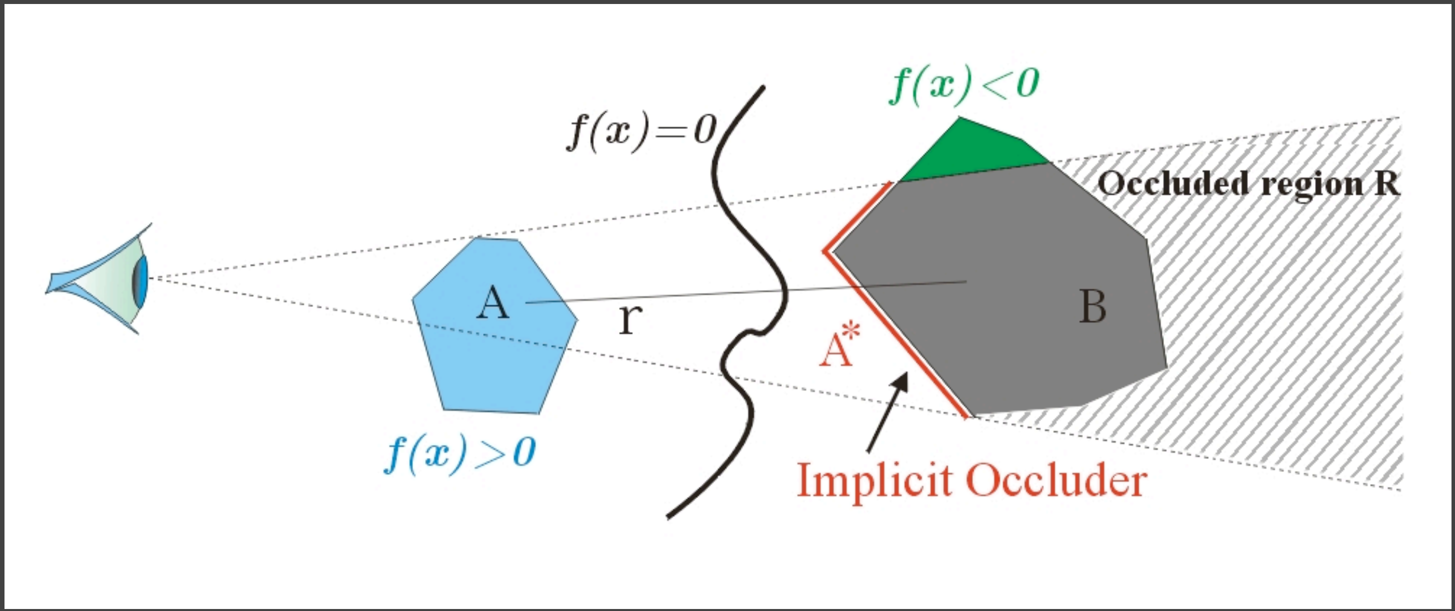
Goal:

- Speed up the computation and rendering of opaque isosurfaces by performing visibility computations



PPM Data Set: 1536 x 1536 x 512
Isovalue = 127
Total number of faces = 89 Million

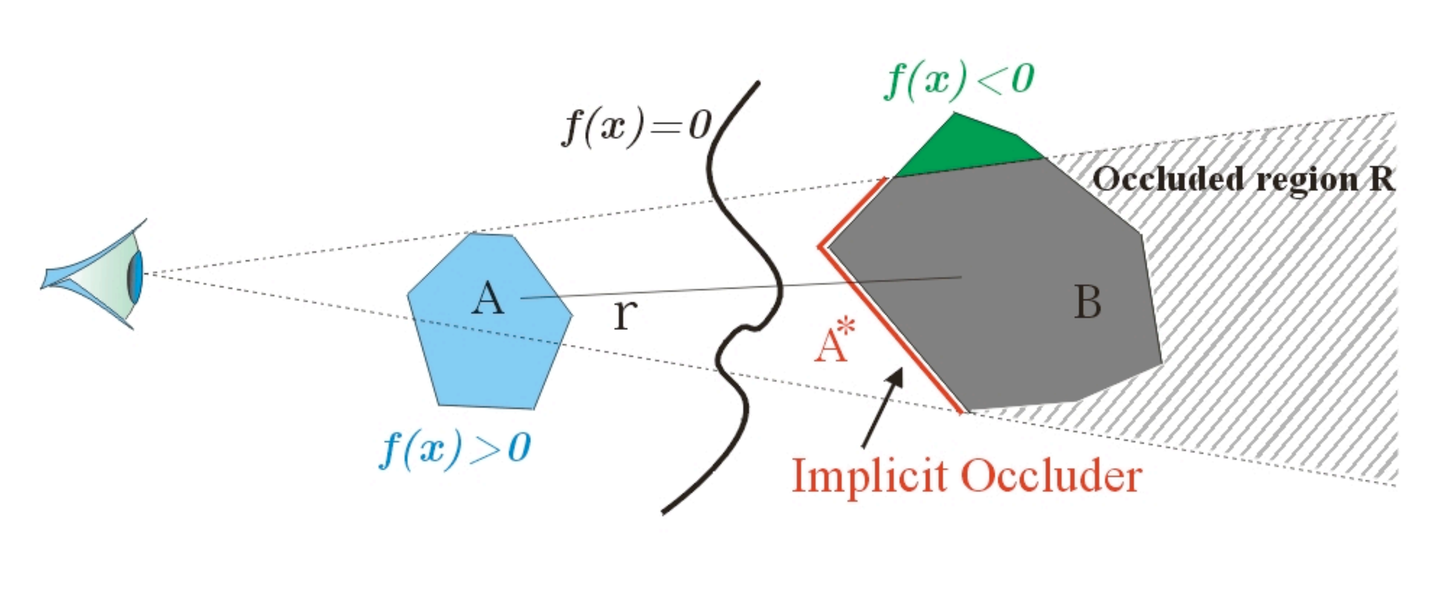
Implicit Occluder



- f is a continuous scalar field defined on a convex domain D.
- region A: f(x) assume only positive value.
- region B: f(x) assume only negative value.
- a viewpoint and central projection A* of region A onto the boundary of B

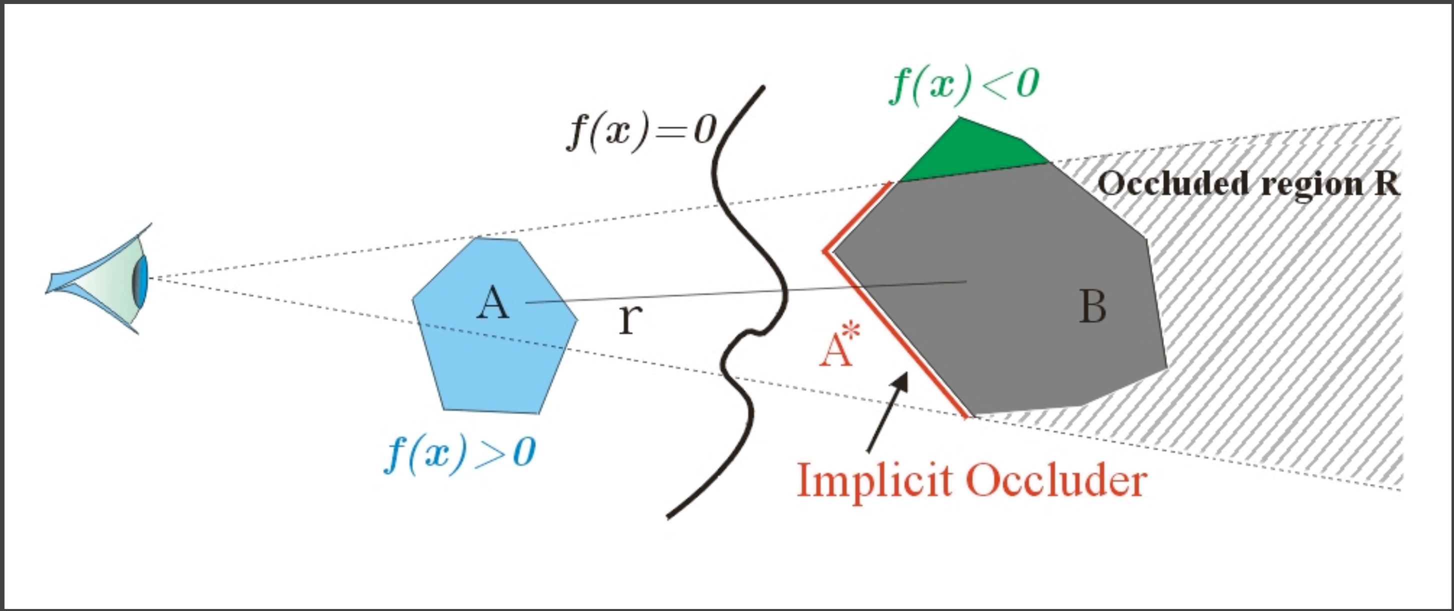
Implicit Occluder

SIBGRAPI 2005



- Any segment r connecting the current viewpoint with A* must also intersect the isosurface $f^{-1}(0)$.
- Therefore, region R behind A* is completely occluded and can be used as an occluder in place of $f^{-1}(0)$.

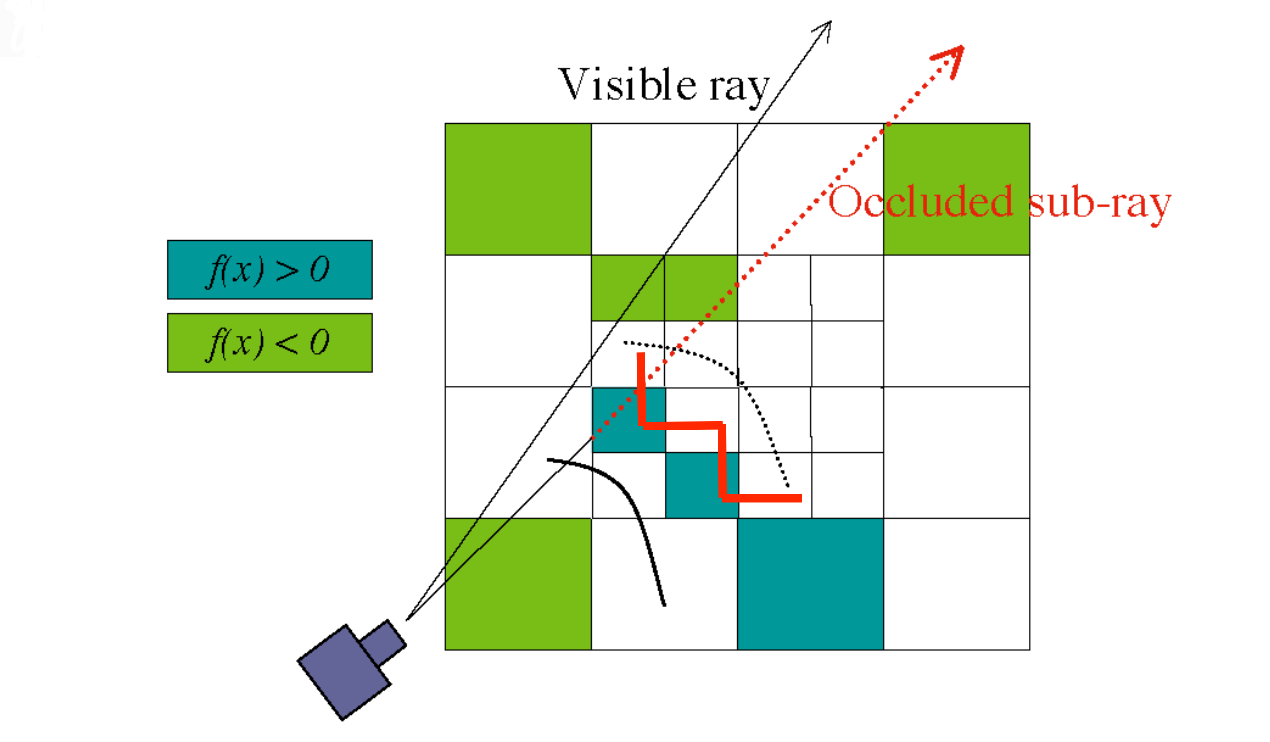
Implicit Occluder



- Exploits the continuity of f to determine conservative visibility bounds implicitly, i.e. without computing the isosurface $f^{-1}(0)$.
- Implicit Occluders are generated based on the change in sign of f

Implicit Occluders

- Use an octree with per node min-max information
- Find the closest occurrence of sign change



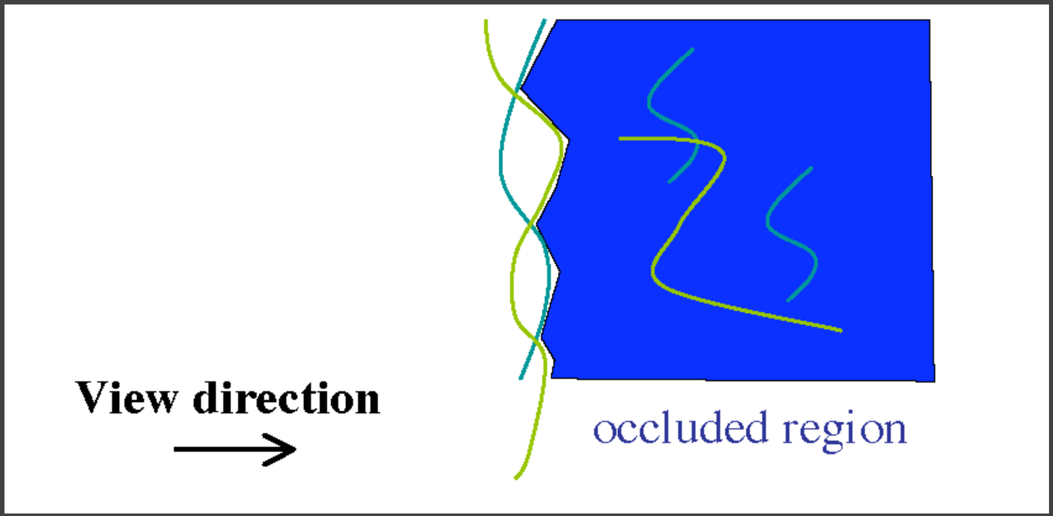
Implicit Occluders

Algorithm can be divided into three main parts:

1. Build screen-space per-pixel occluders
2. For the remaining (visible) nodes, compute and render the isosurface.
3. Use hardware occlusion-culling queries to prune invisible parts of the octree.

Step 1: Build screen-space per-pixel occluders

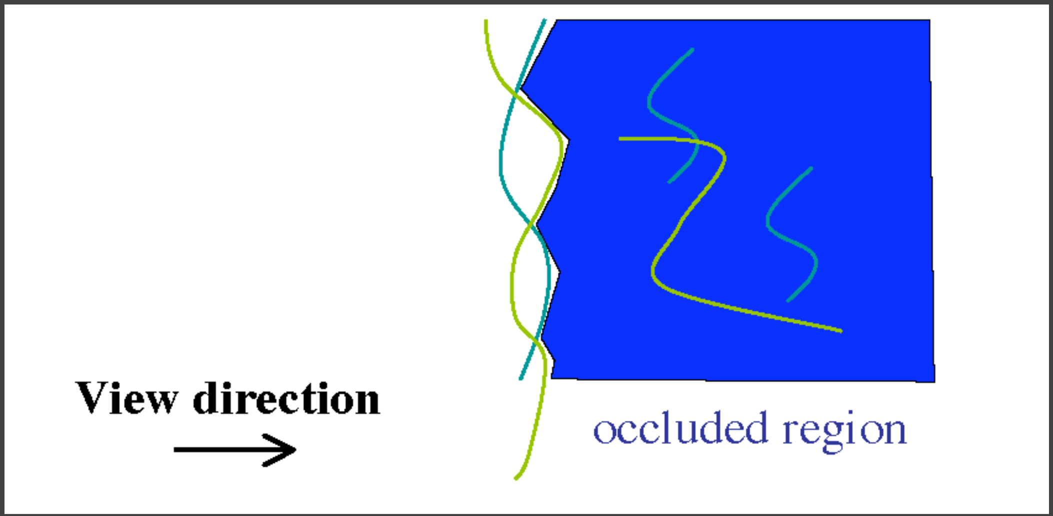
Finding regions of screen-space overlap between nodes that are above and below the isosurface value.



Step 1: Build screen-space per-pixel occluders

SIBGRAPI 2005

Finding regions of screen-space overlap between nodes that are above and below the isosurface value.



Two-pass strategy:

- all nodes with negative function value are drawn in the first pass
- the nodes with positive function value are drawn in the second pass

SIBGRAPI 2005

Step 1: Build screen-space per-pixel occluders

glClearDepth(1); **Clear Buffers** glClearStencil(0);

Depth Buffer

Front to Back

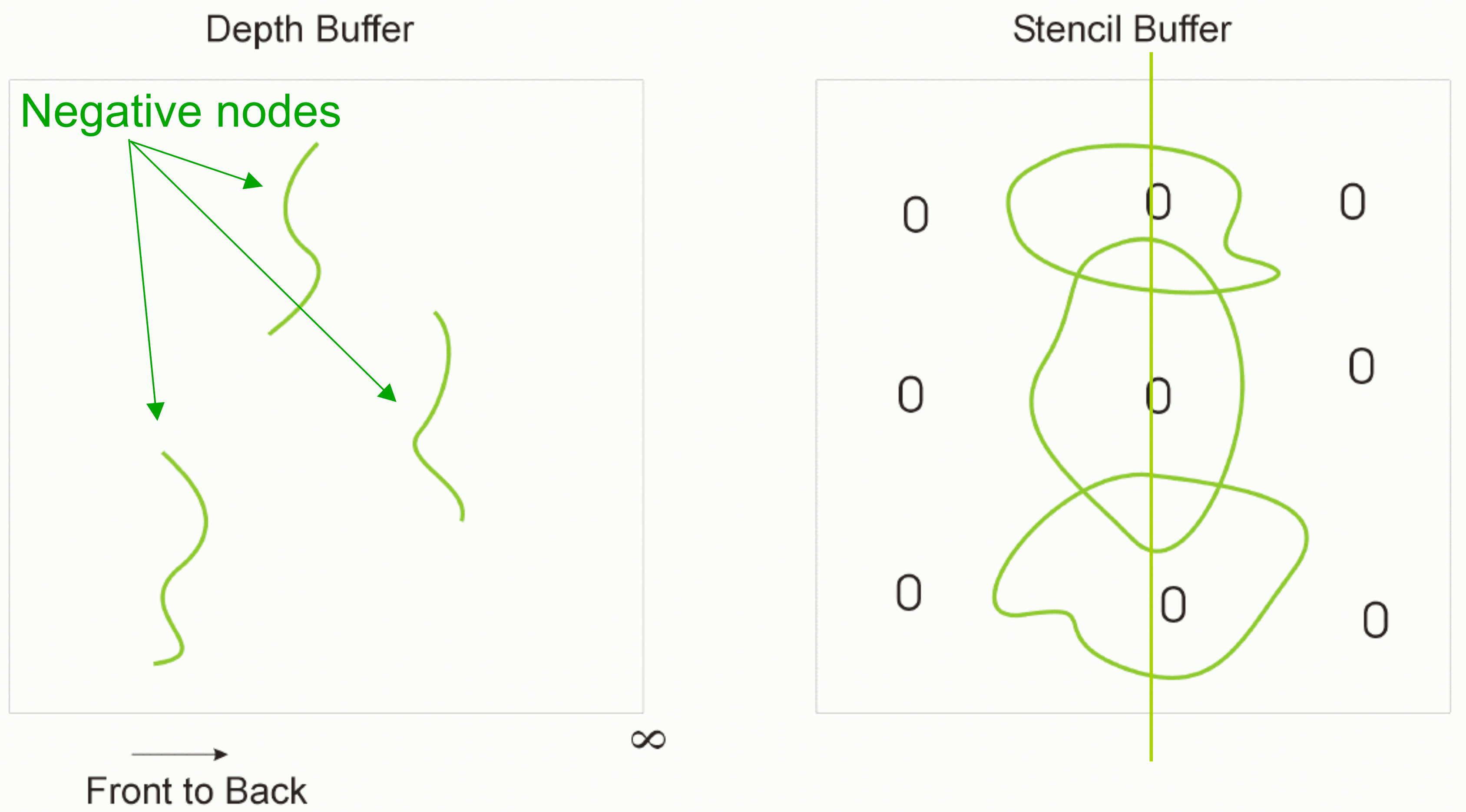
Stencil Buffer

0	0	0
0	0	0
0	0	0

Step 1: Build screen-space per-pixel occluders

SIBGRAPI 2005

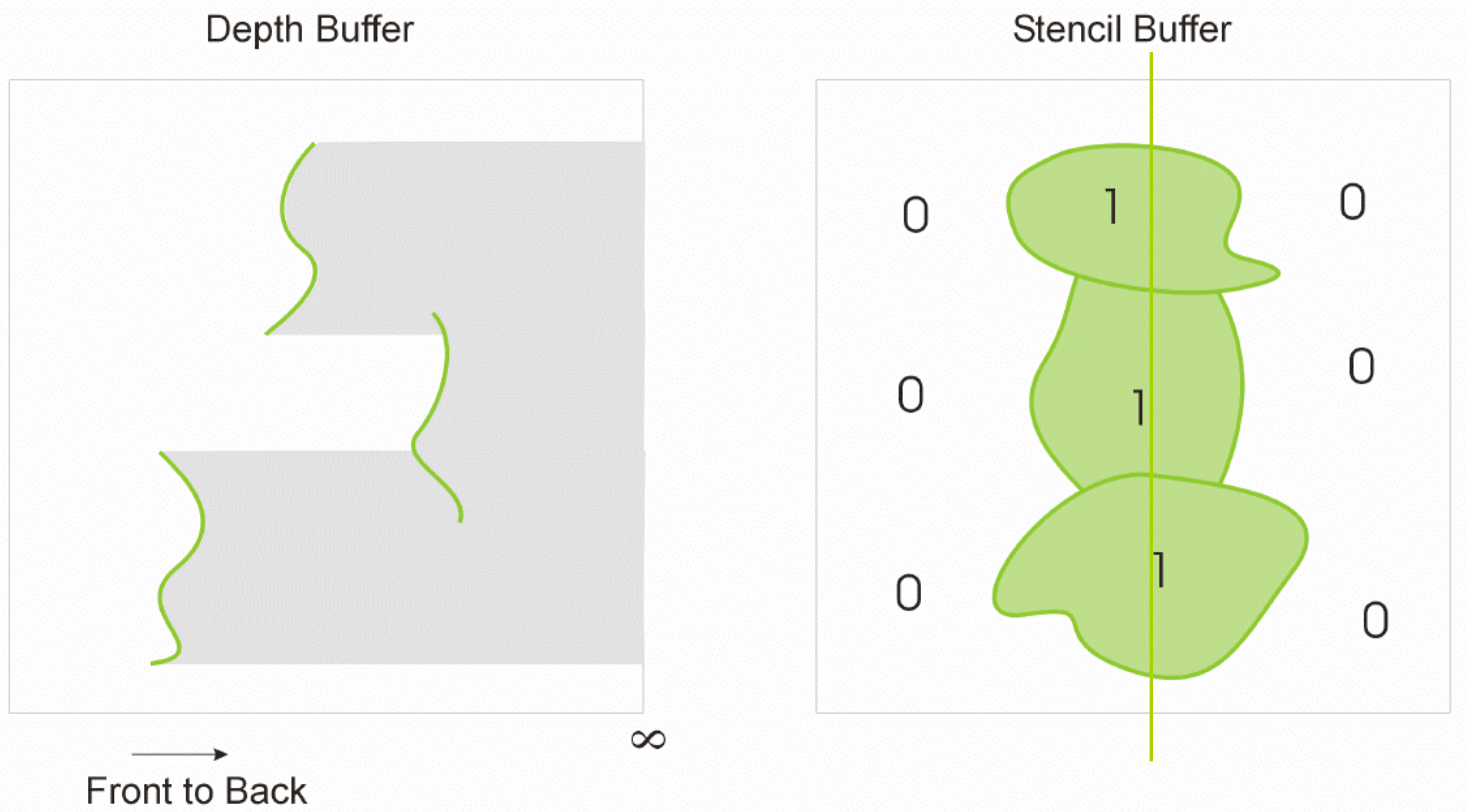
First render pass: Nodes with negative function value are drawn.



SIBGRAPI 2005

Step 1: Build screen-space per-pixel occluders

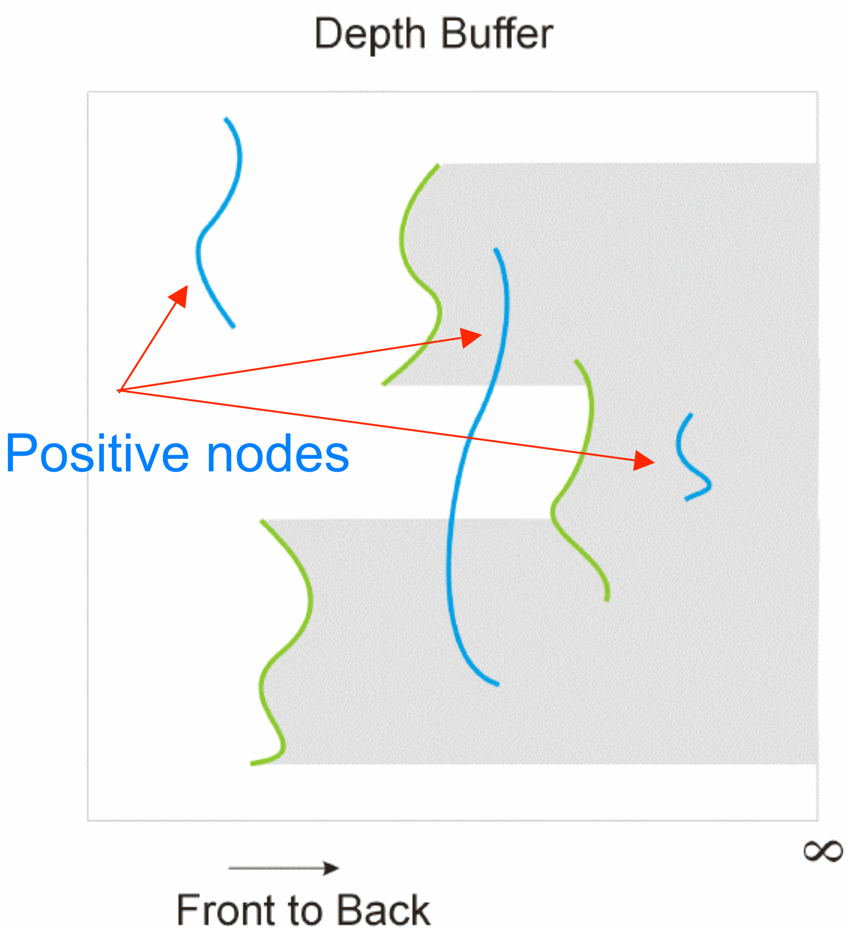
First render pass: Nodes with negative function value are drawn.

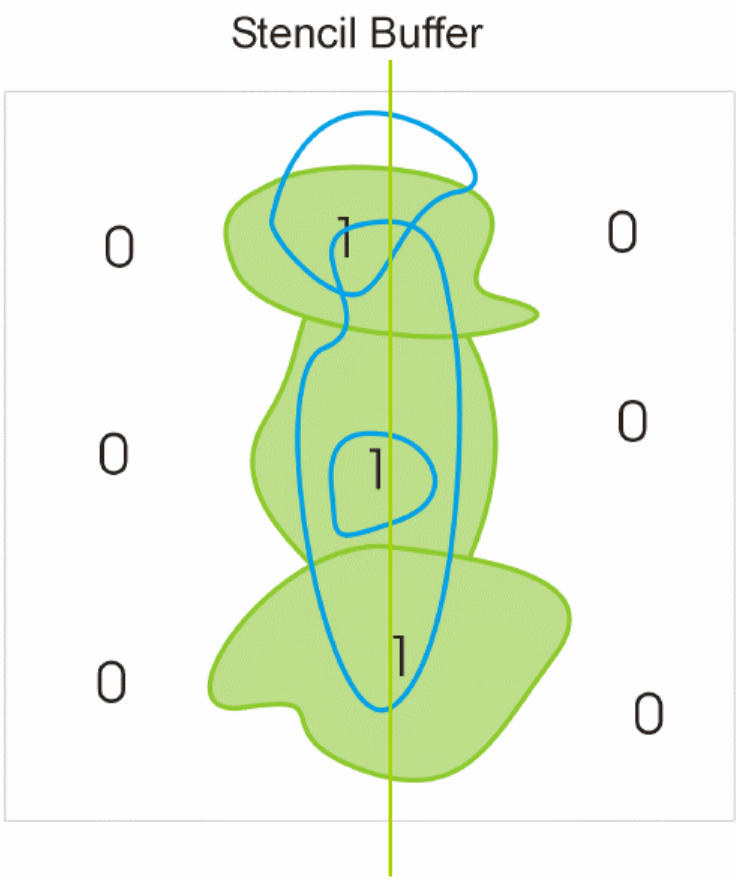


Step 1: Build screen-space per-pixel occluders

SIBGRAPI 2005

Second render pass: Nodes with positive function value are drawn, set Depth Buffer to GL_GREATER

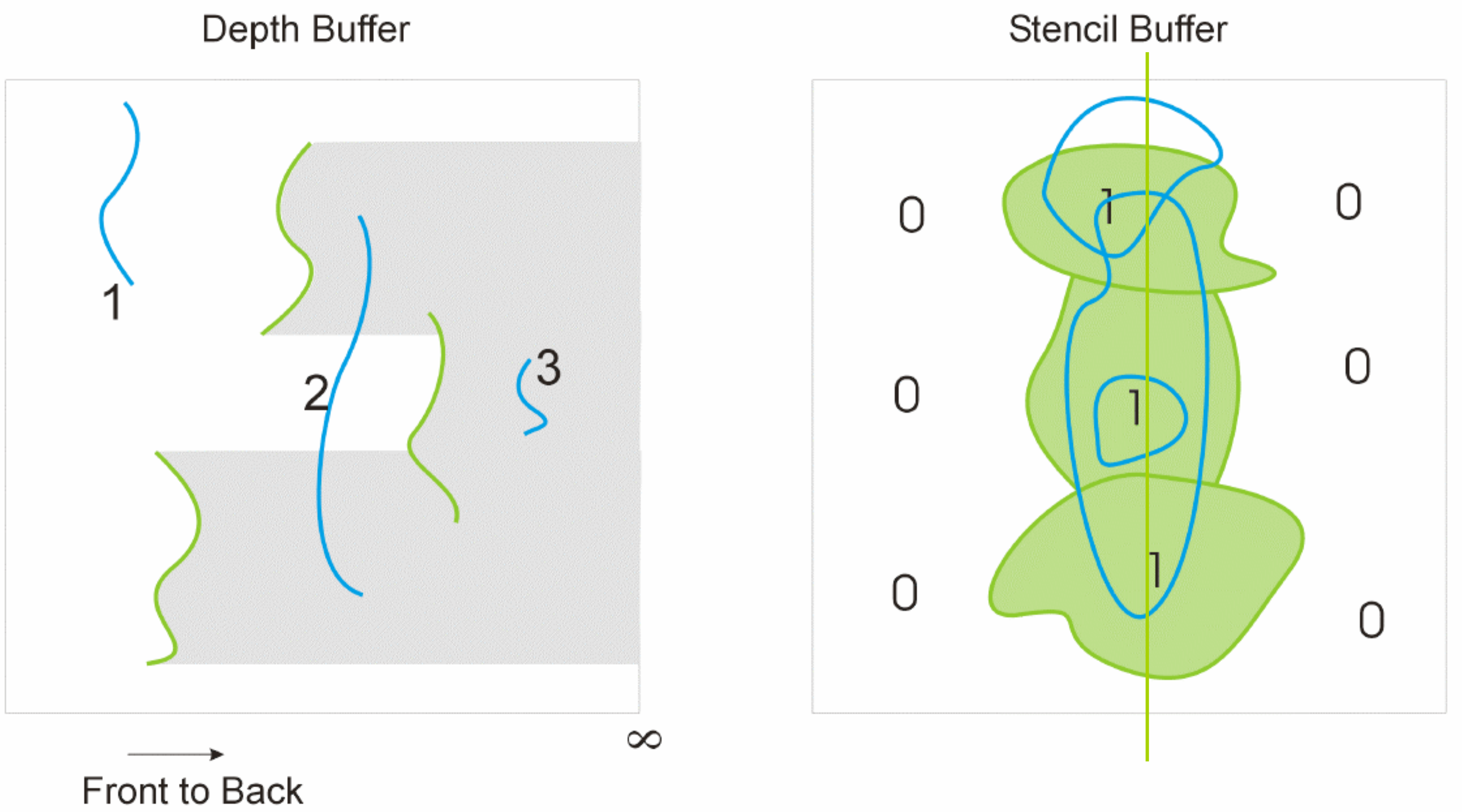




Step 1: Build screen-space per-pixel occluders

SIBGRAPI 2005

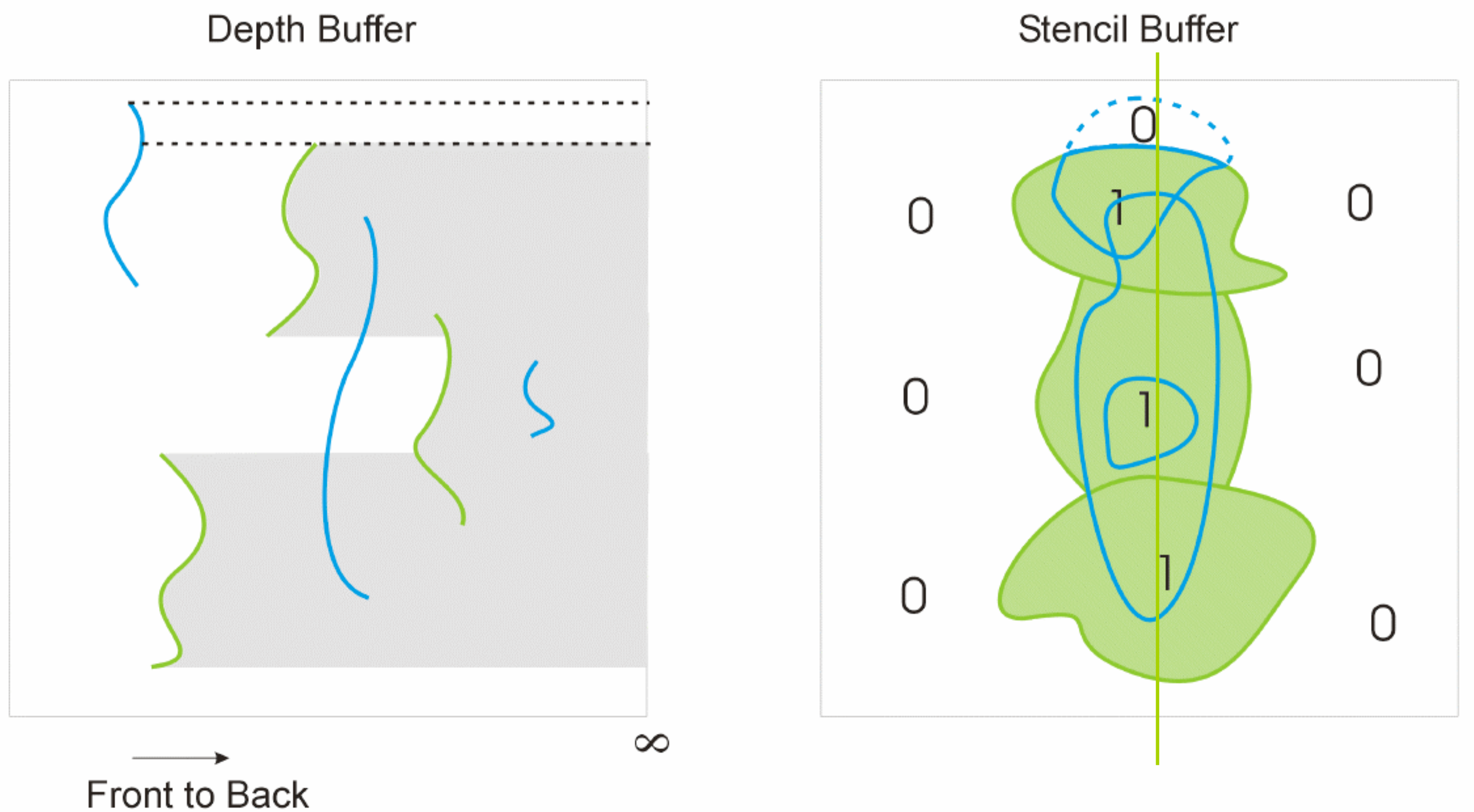
Second render pass: The second pass is performed front-to-back.
Sets correct depth for the complete I.Occluders.



SIBGRAPI 2005

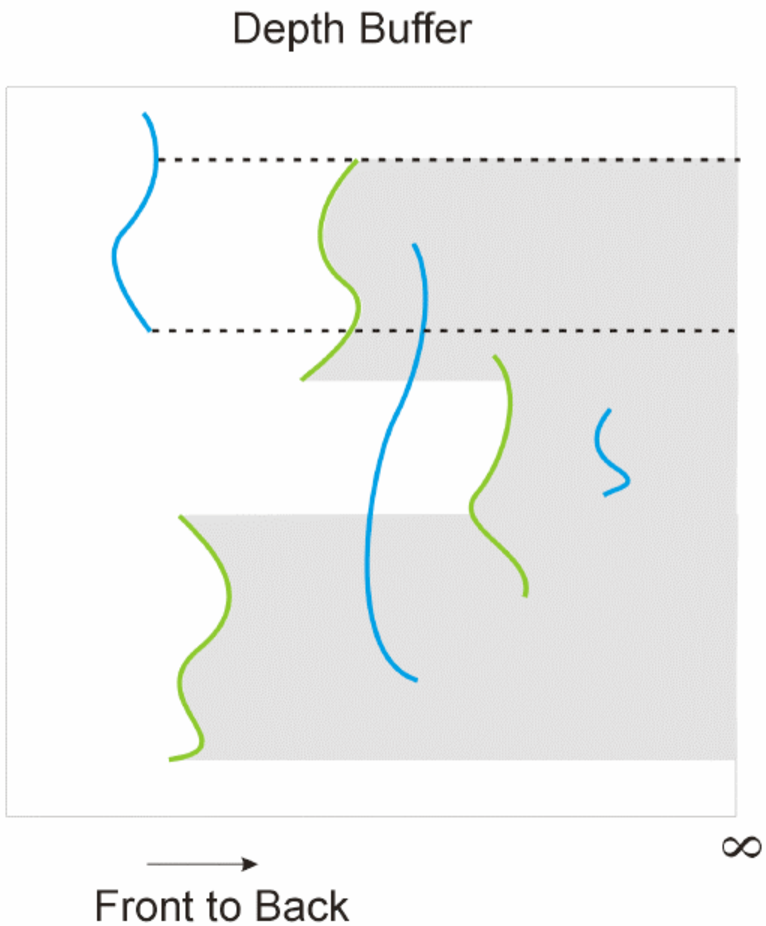
Step 1: Build screen-space per-pixel occluders

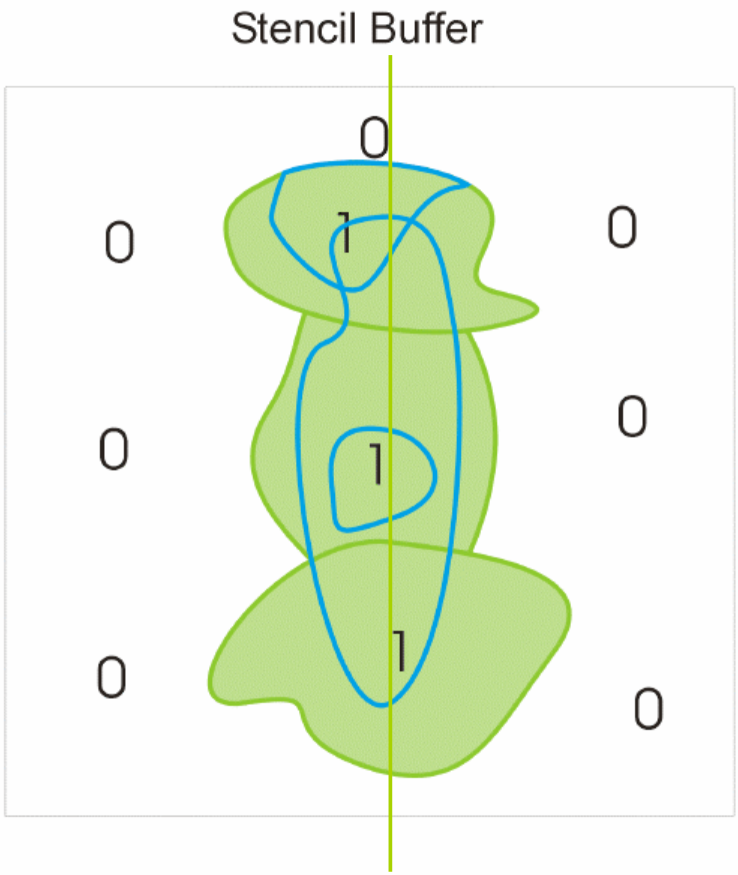
The stencil buffer is used to determine which pixels have been covered in the first pass.



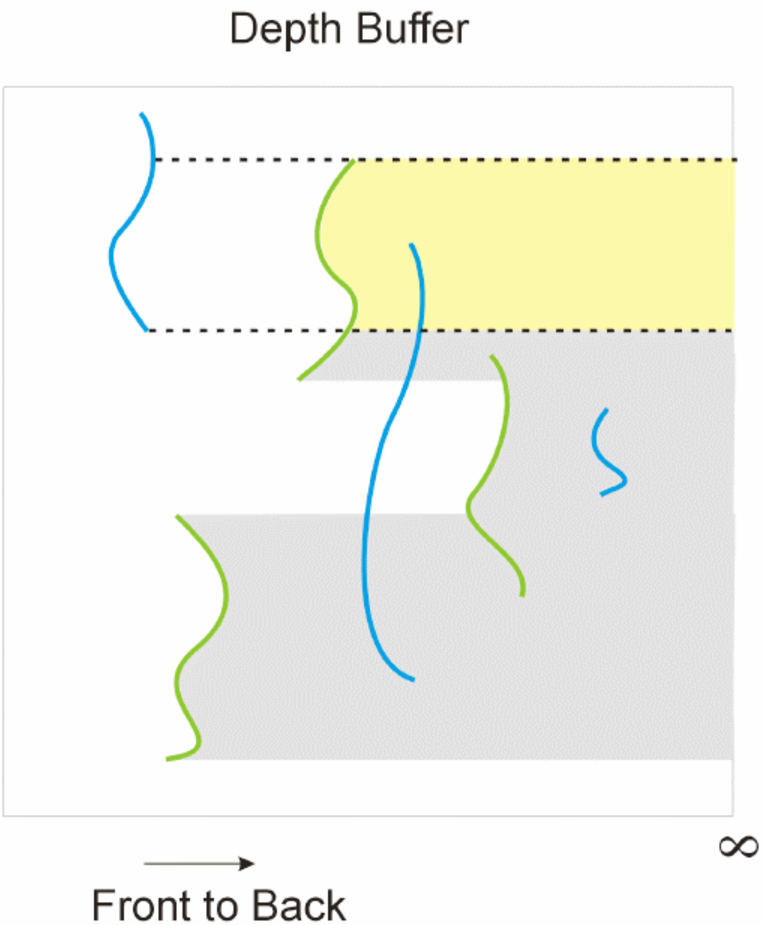
Step 1: Build screen-space per-pixel occluders

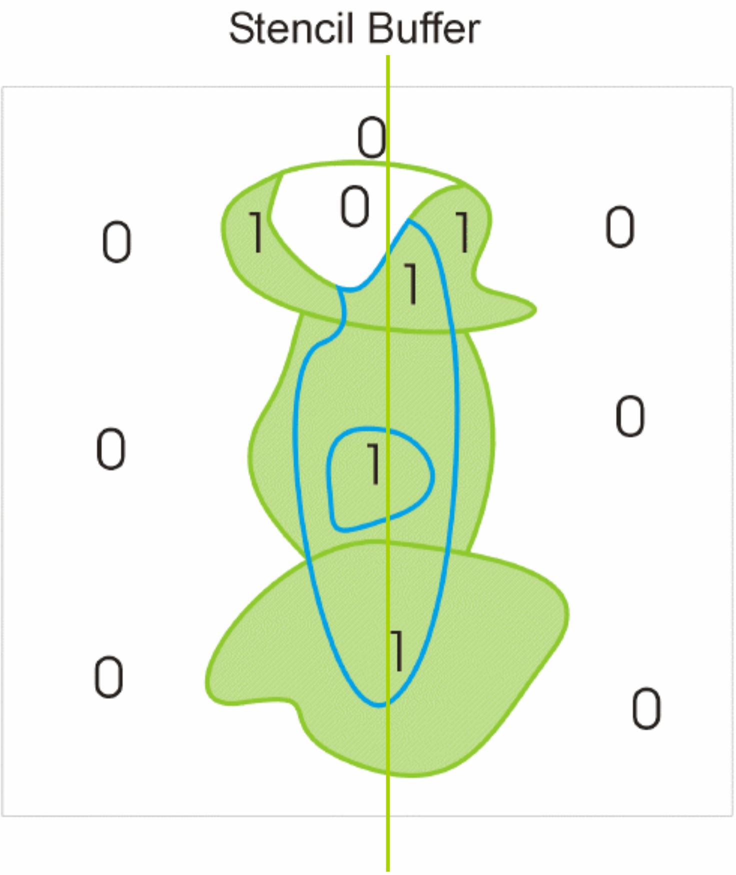
SIBGRAPI 2005





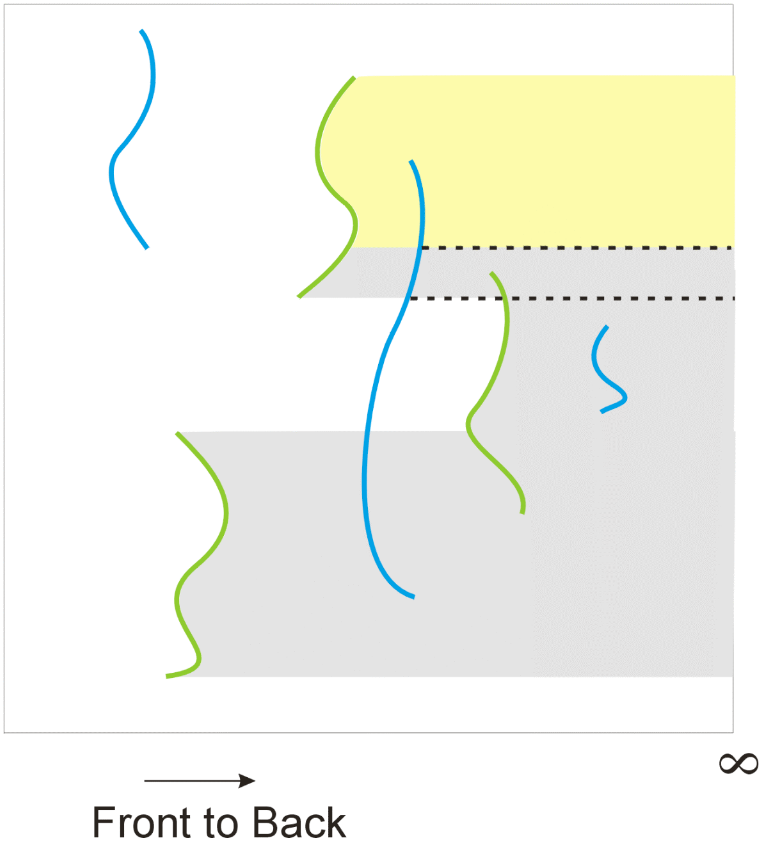
Step 1: Build screen-space per-pixel occluders



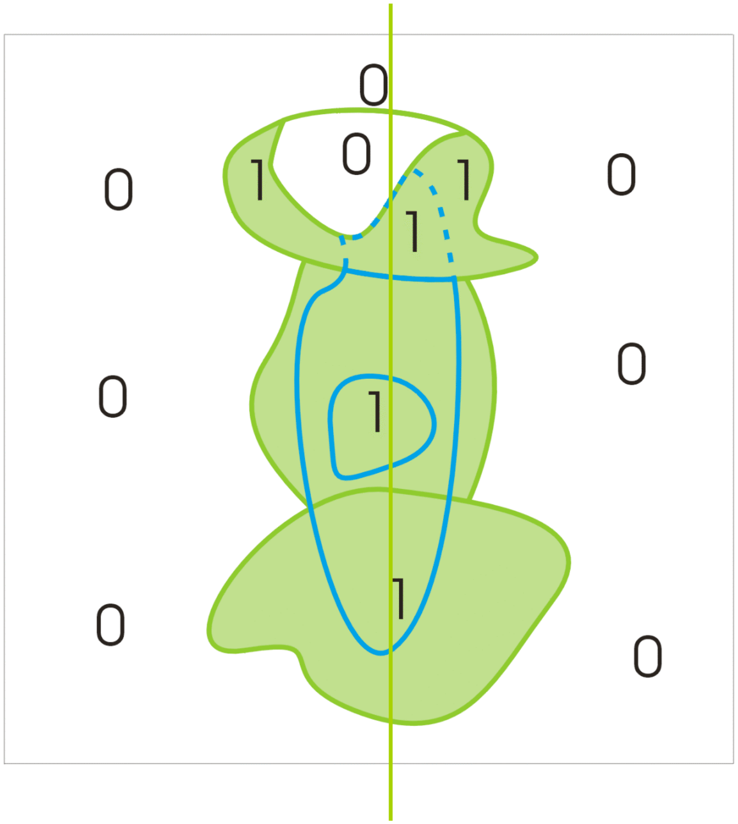


Step 1: Build screen-space per-pixel occluders

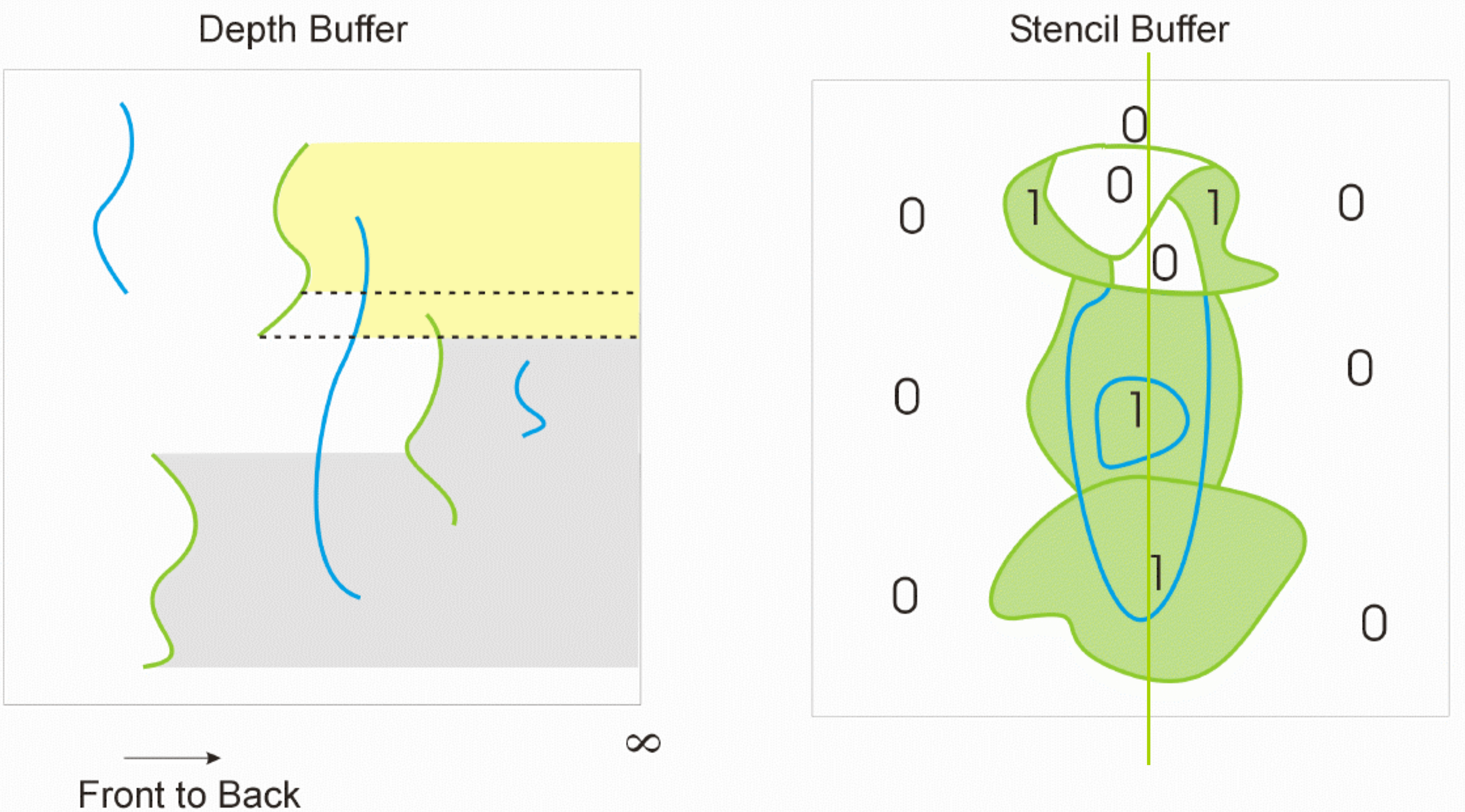
Depth Buffer



Stencil Buffer

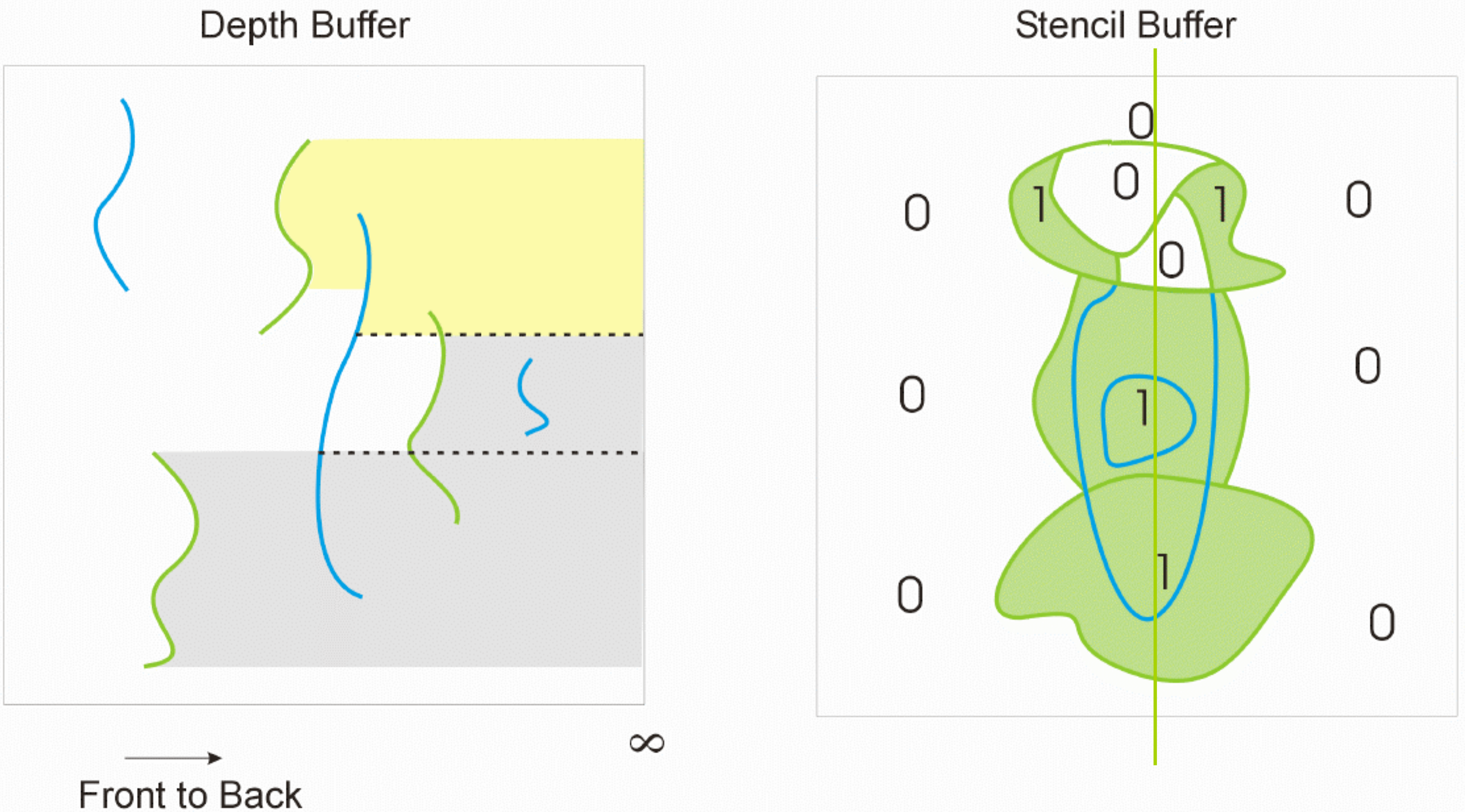


Step 1: Build screen-space per-pixel occluders



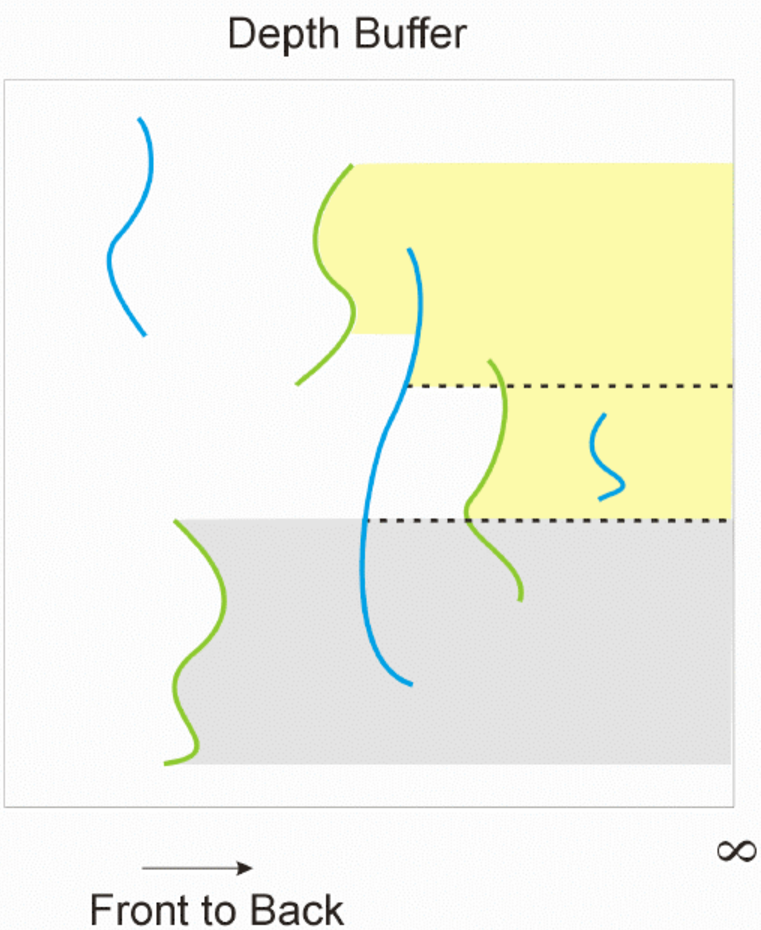
SIBGRAPI 2005

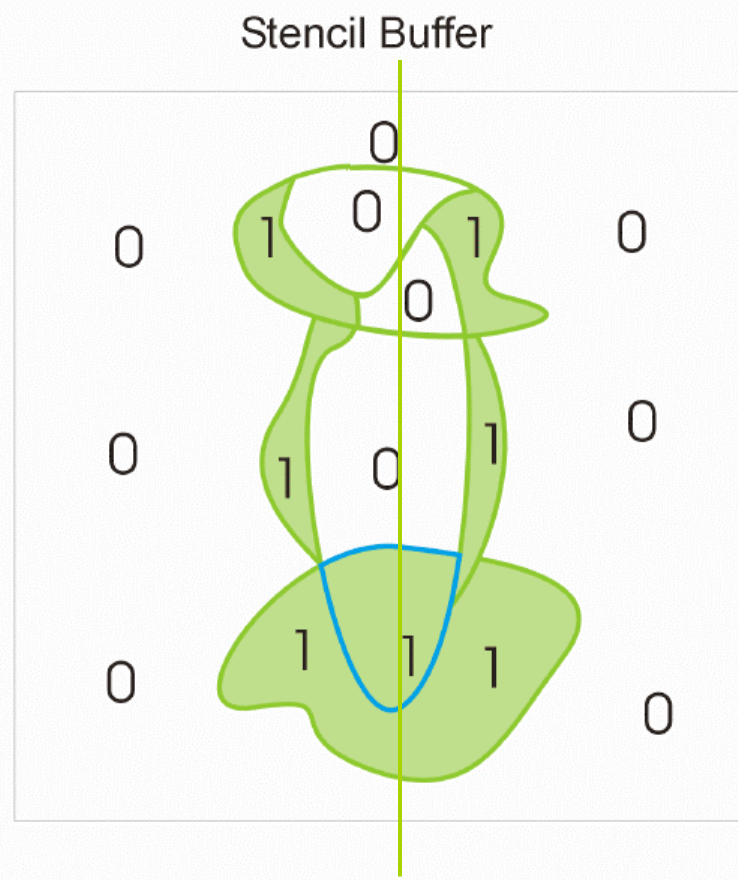
Step 1: Build screen-space per-pixel occluders



SIBGRAPI 2005

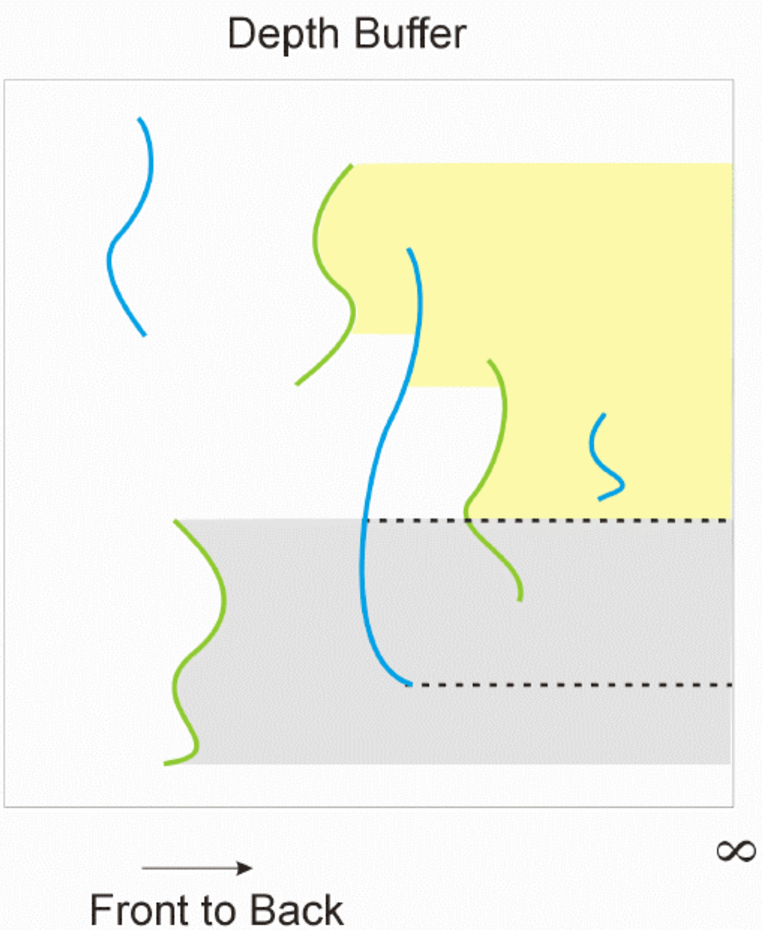
Step 1: Build screen-space per-pixel occluders

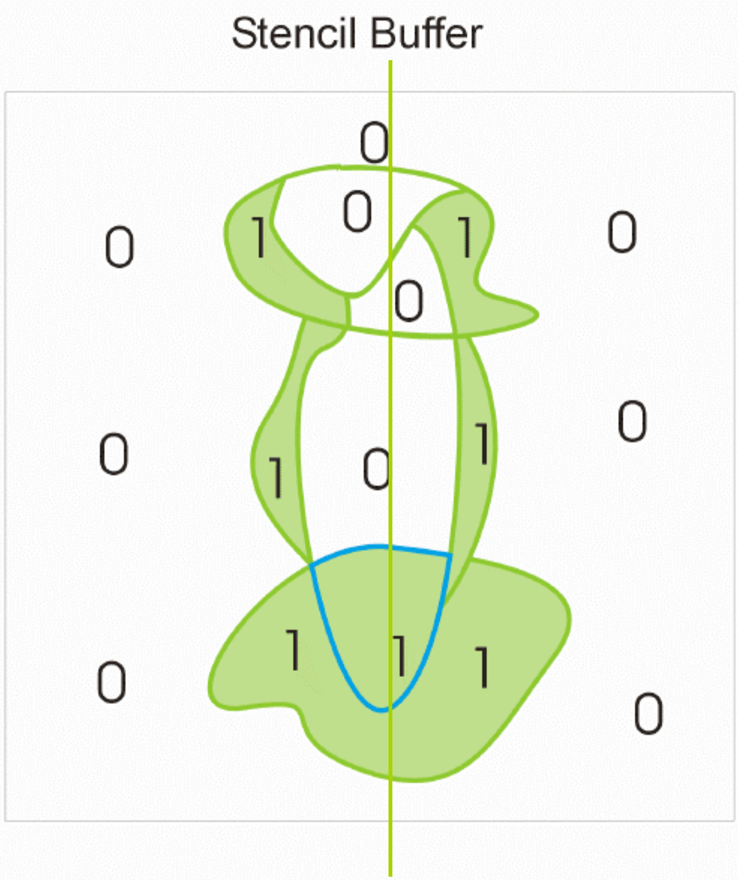




Step 1: Build screen-space per-pixel occluders

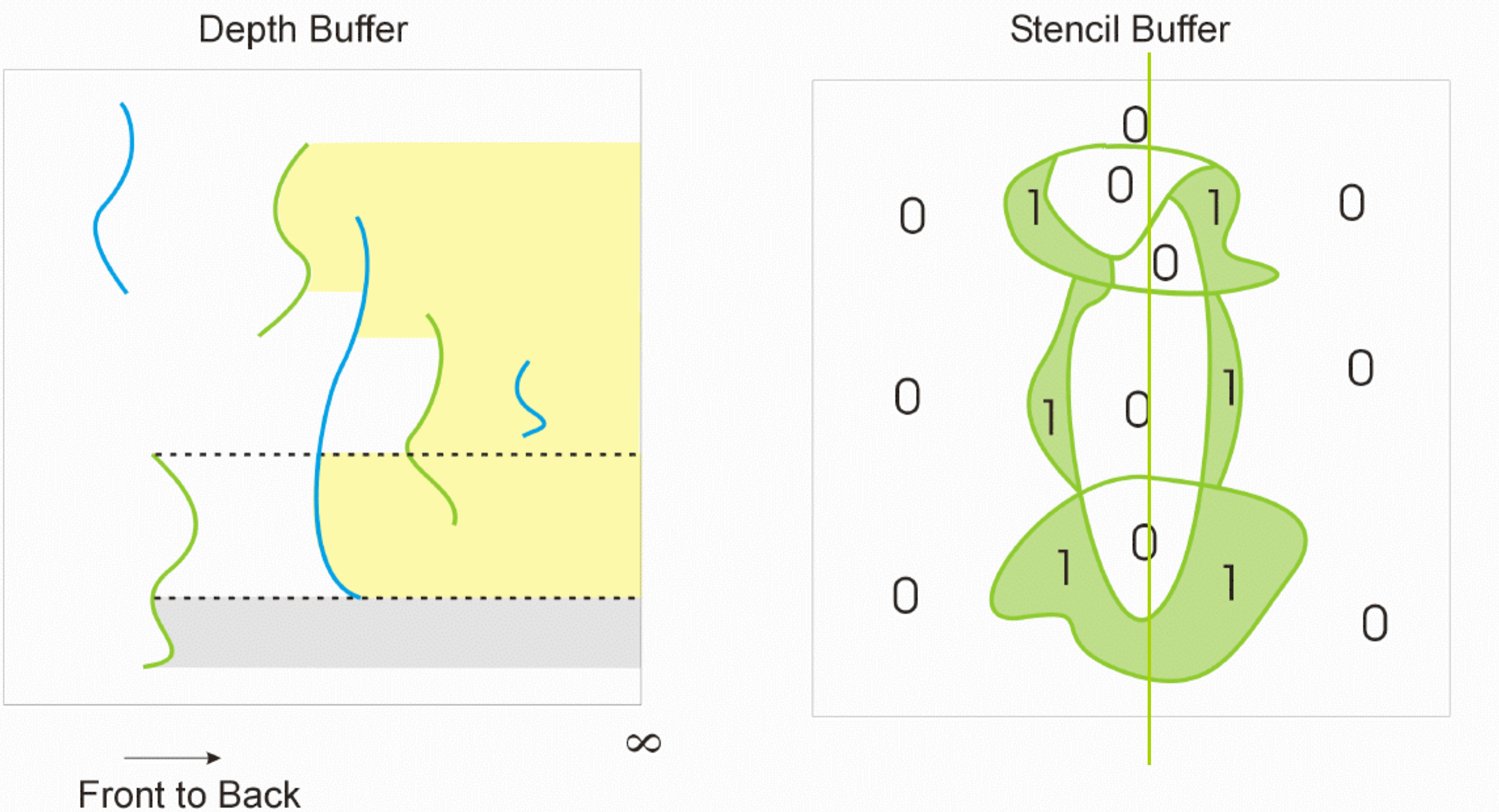
SIBGRAPI 2005





SIBGRAPI 2005

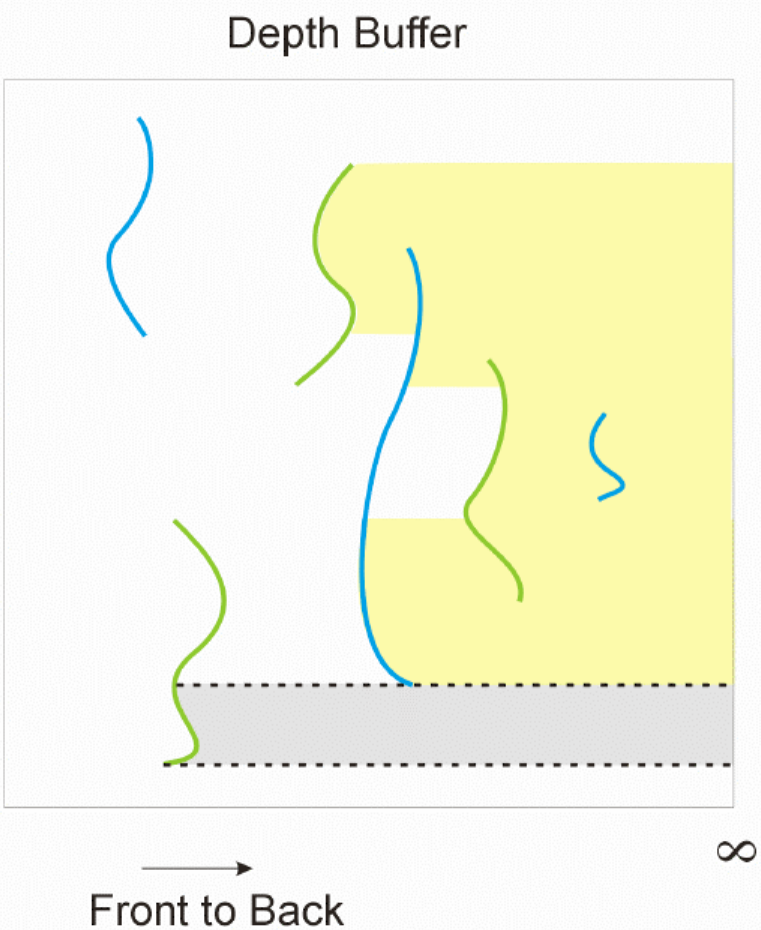
Step 1: Build screen-space per-pixel occluders

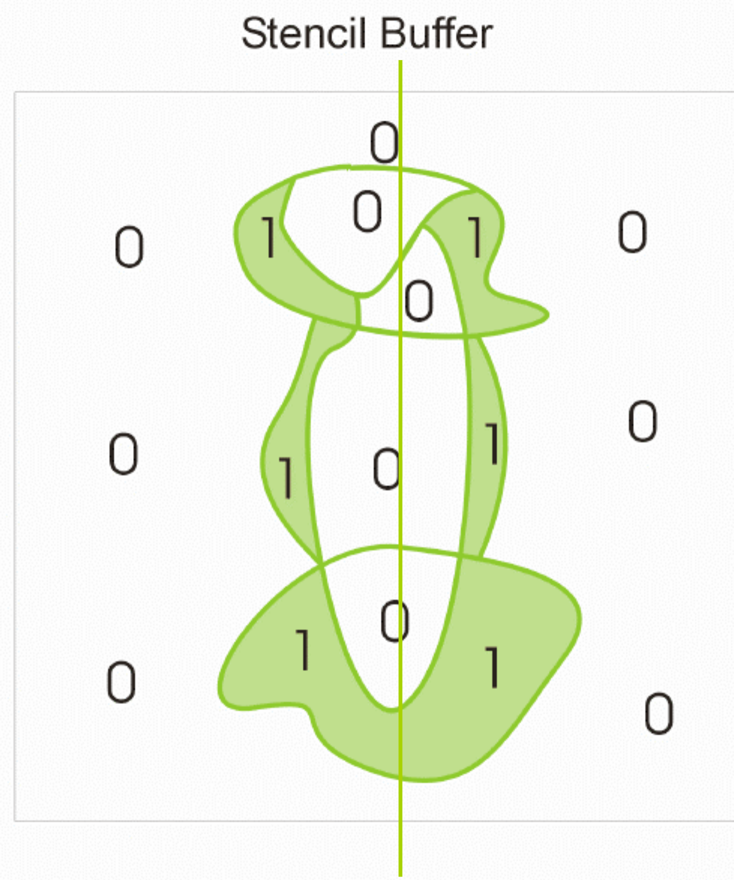


Step 1: Build screen-space per-pixel occluders

SIBGRAPI 2005

The depth of the Implicit Occluders that remain incomplete after the second pass is finally set back to infinity.

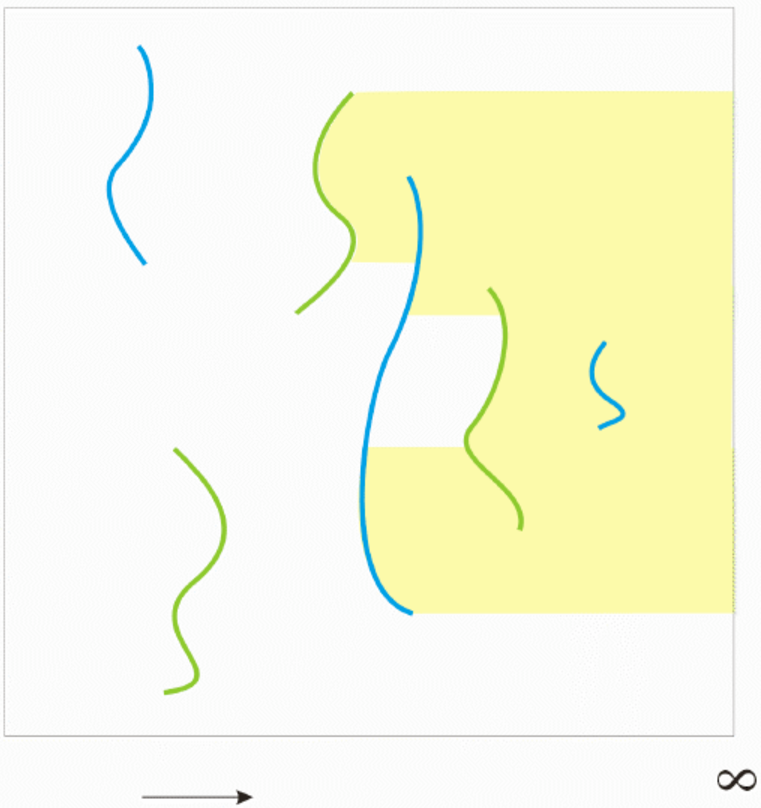




SIBGRAPI 2005

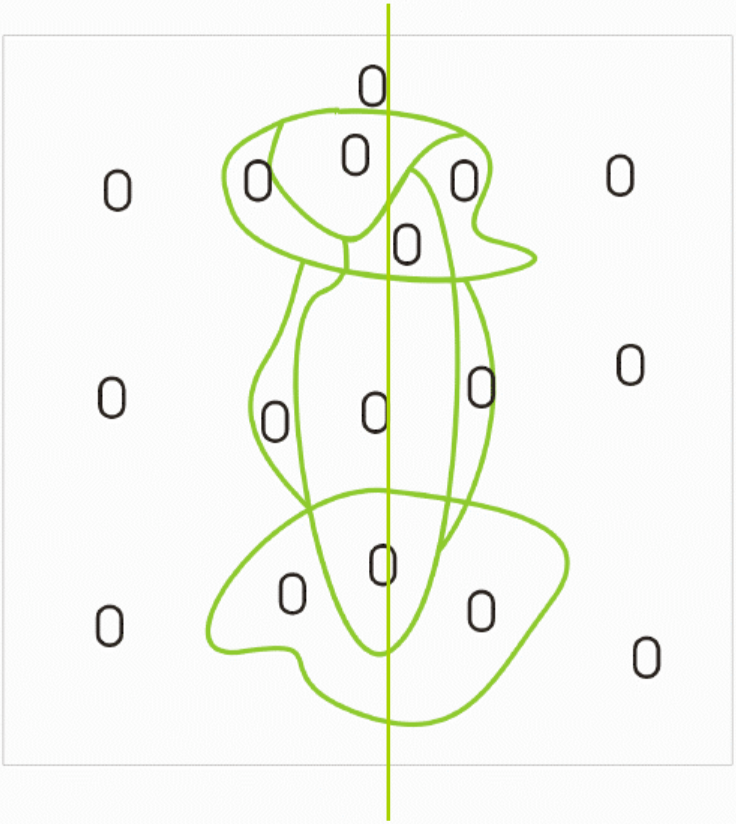
Step 1: Build screen-space per-pixel occluders

Depth Buffer



Front to Back

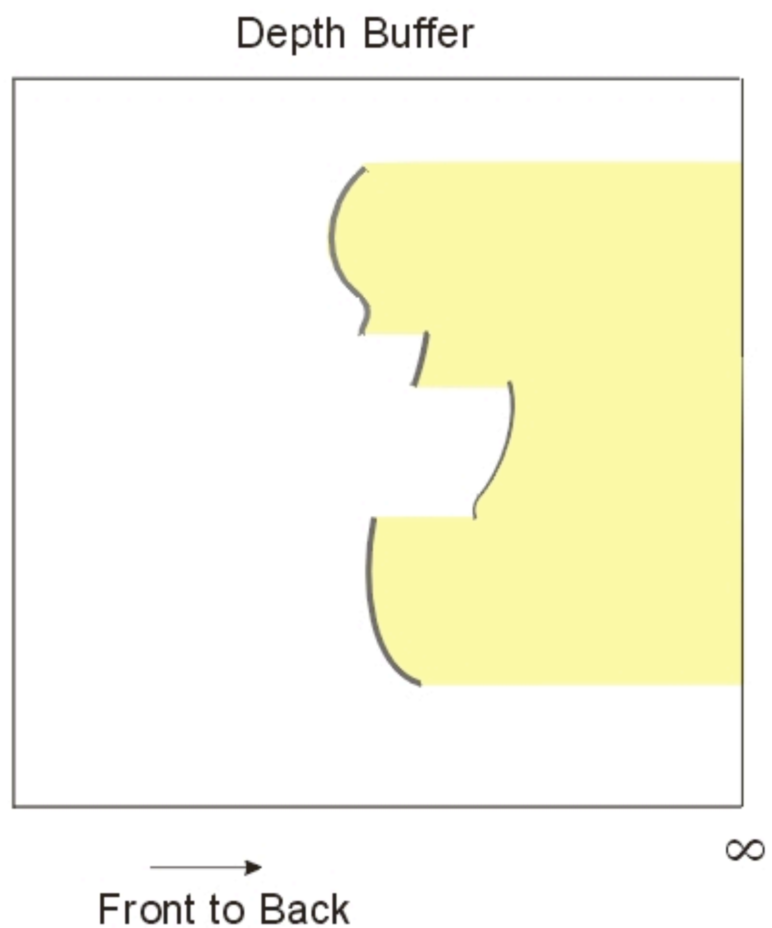
Stencil Buffer



Step 1: Build screen-space per-pixel occluders

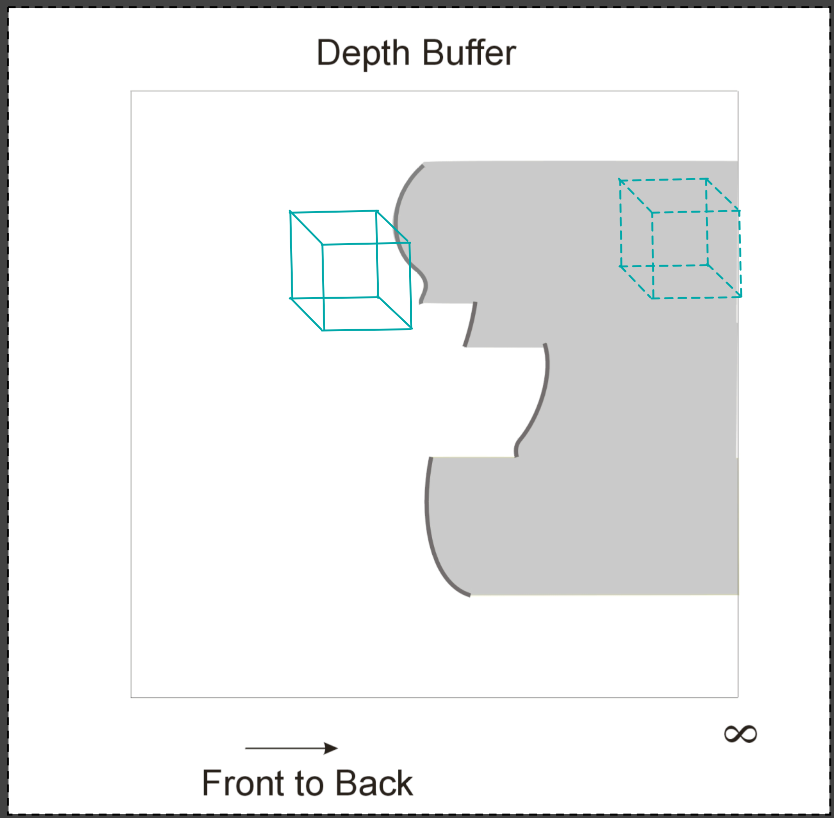
SIBGRAPI 2005

- The yellow region corresponds to the Implicit Occluder.



Step 2

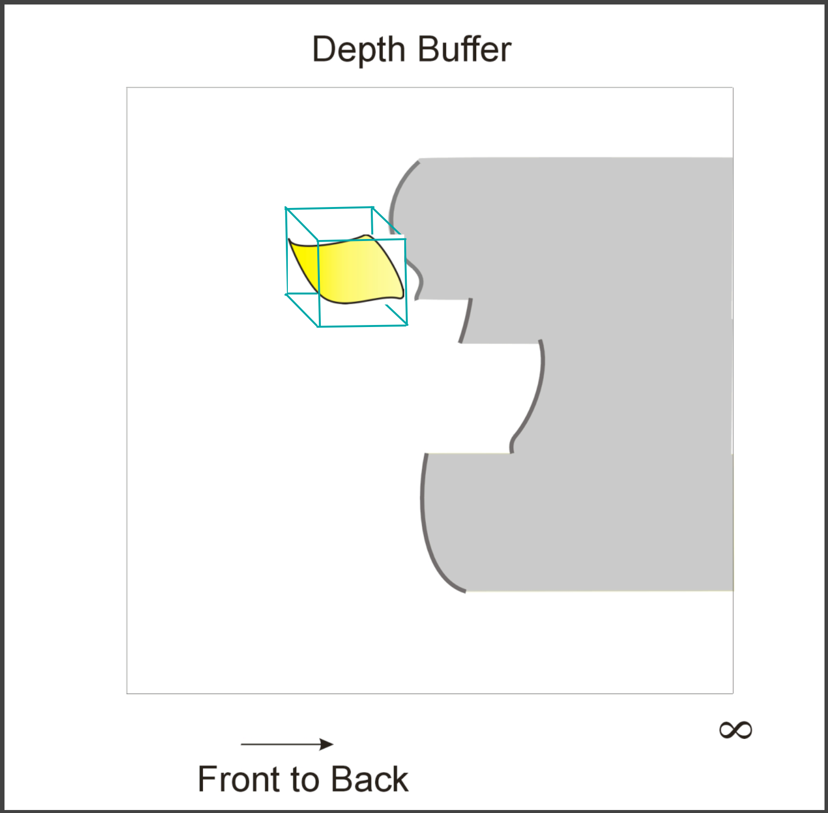
To test if a given node of the octree is visible or not, check whether the bounding box of the node is visible.



Step 3

SIBGRAPI 2005

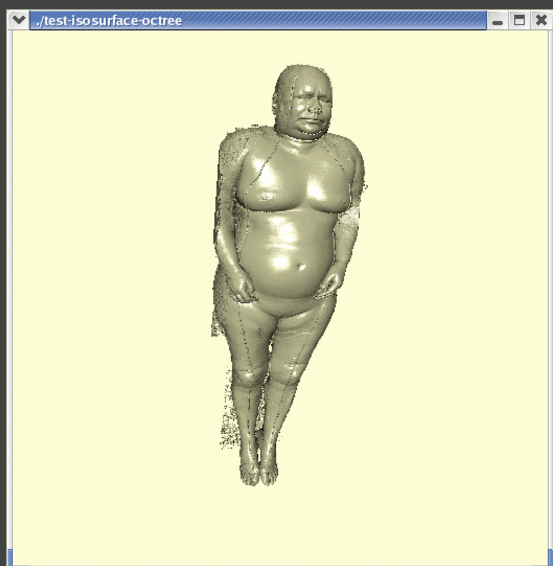
If the node is determined to be visible, then the isosurface contained in the node is computed and rendered.



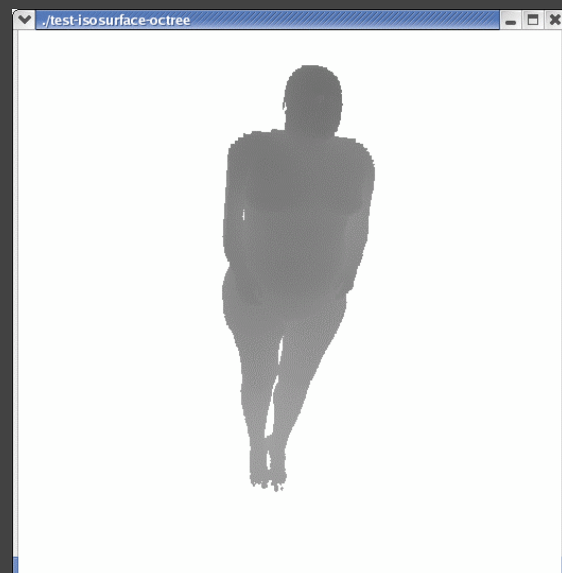
Implicit Occluder Results

SIBGRAPI 2005

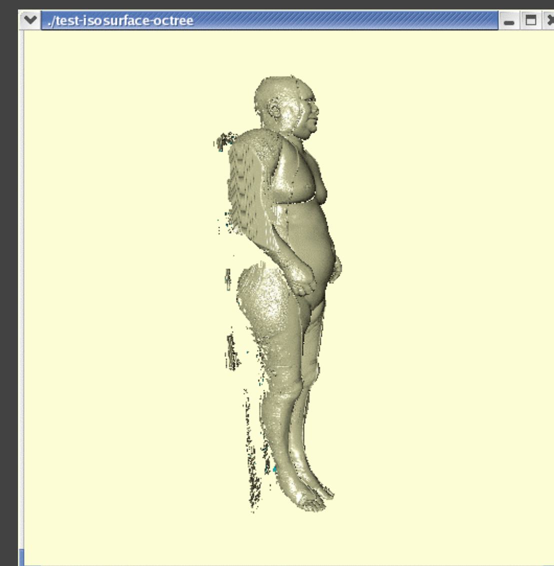
Example



Visible Woman



Implicit Occluder

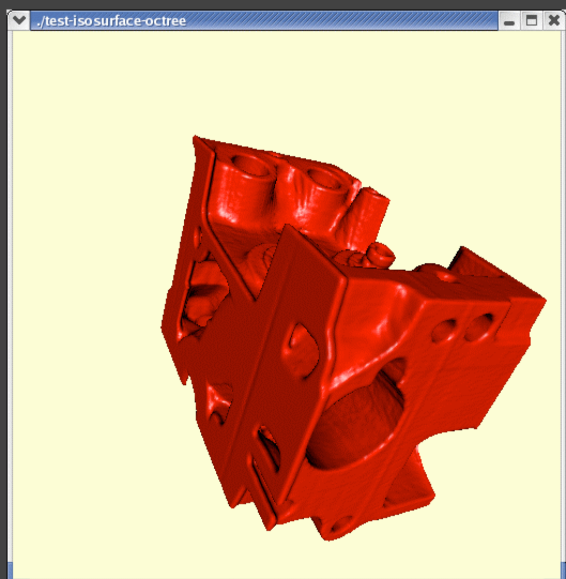


Occluded faces in this example.

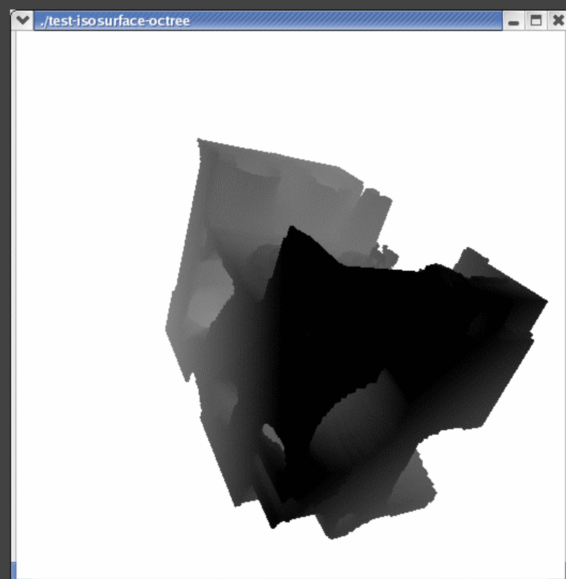
SIBGRAPI 2005

Implicit Occluder Results

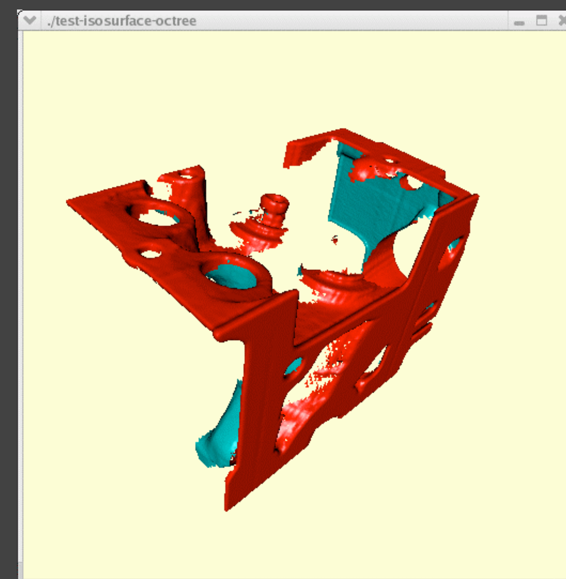
Example



Engine



Implicit Occluder



Occluded faces in this example.