A Glance on Work of Jorge Stolfi in Computer Graphics

Prolegomenon

- Why we are here …
- How I got acquainted with Jorge

Salve Jorge !

Compugraphics 1991

COMPUGRAPHICS '91 - PANEL 3

COMPUTATIONAL GRAPHICS : THE EMERGING ALL-ENCOMPASSING GRAPHICS ENDEAVOR

Moderator: Harold P. Santo

Panelists

Vera Anand Clemson University, USA

Les Piegl University of South Florida, USA

Hellmuth Stachel

Jorge Stolfi DEC-SRC, USA

Villas de Sesimbra

Extending the Z-Buffer

Computer Graphics, Volume 24, Number 4, August 1990

Rendering CSG Models with a ZZ-Buffer

David Salesin* and Jorge Stolfi†

*Computer Science Department
Stanford University
Stanford, CA 94305

†DEC Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301

The ZZ-Buffer

- Main Goal
  - Acceleration Scheme for Ray Tracing
    - Stochastic and Distributed RT
    - General Surfaces (including CSG)

- Characteristics
  - Works on *Screen Space*
    (simpler scheme)
  - Optimizes *initial* and *final* rays only
    (the most important ones)

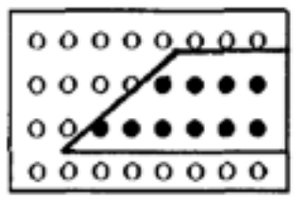
State of the Art

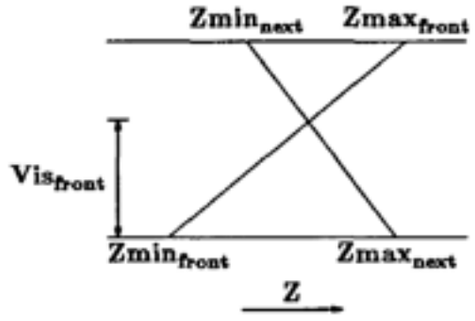
- The Cutting Edge in Rendering
  - Stochastic Ray Tracing
    - Anti-aliasing
    - Transparency
    - Depth of Field
    - Soft Shadows

- Challenges in Surface Visualization
  - CSG Models

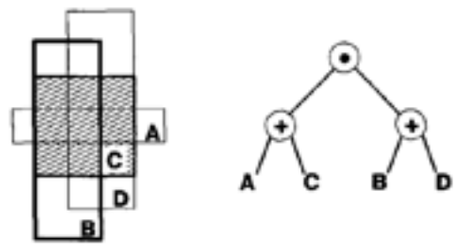
Previous Work

- A-Buffer for Rendering





- Active Zones for CSG



★ *Improvements*

- *Better Anti-aliasing / More Efficient*

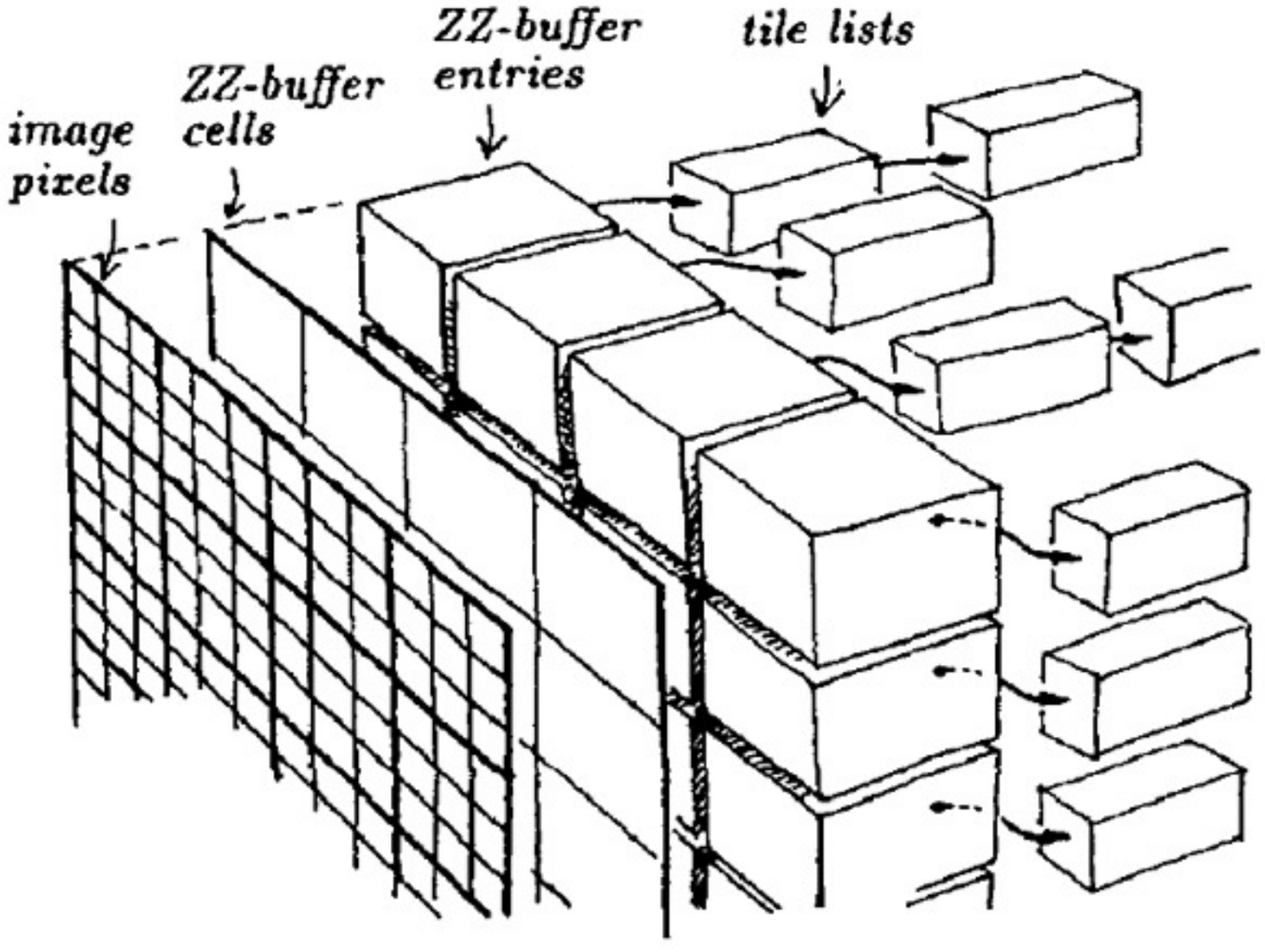
Major Strategies

- Efficient Indexing Scheme
  for Ray-Object Intersection

- Pixel Coverage Analysis
  to Detect Trivial Sampling

- Depth Bounds to
  Eliminate Invisible Objects

The ZZ-Buffer



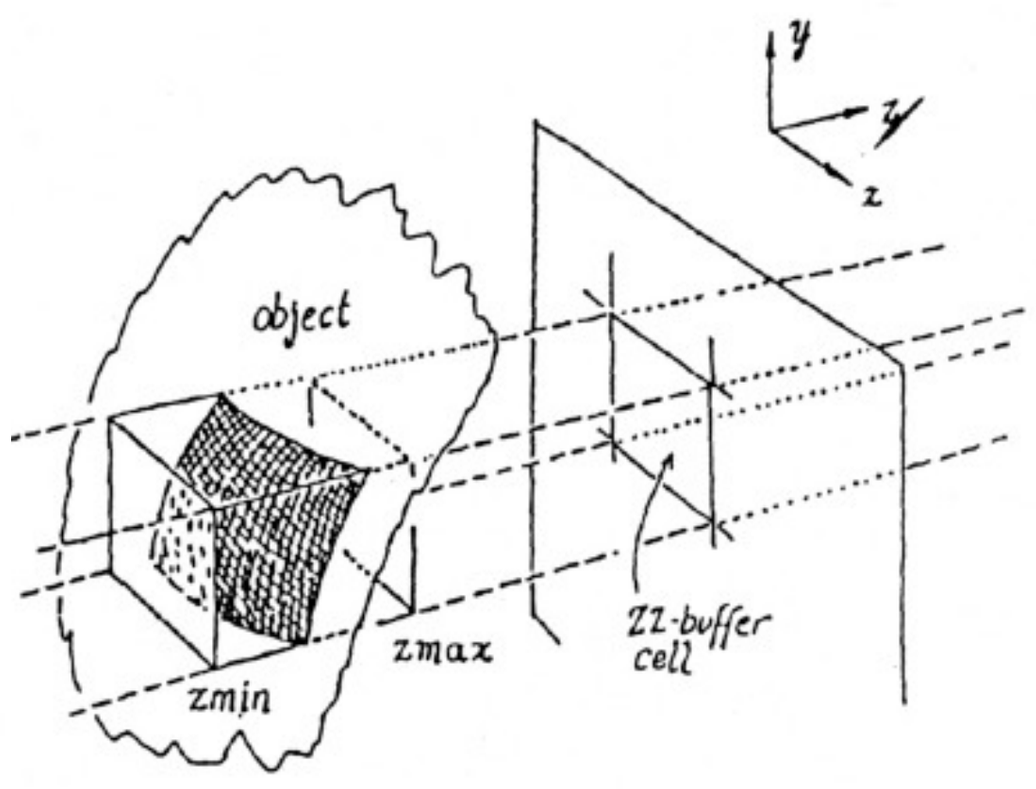
- The Big Picture

ZZ Tiles

- The Data Structure

```
type Tile = record
  obj:          pointer to Object
  zmin, zmax:   Coordinate
  opaque:       boolean
end record

type TileList = record
  first:   Tile
  rest:    pointer to TileList
end record
```



The Algorithm

- Two Phases

1. Tiling Phase:
   Preprocess the Scene into the ZZ-Buffer

   – Screen Space Indexing

   – Visibility Coherency

2. Rendering Phase:
   Compute Pixel Values

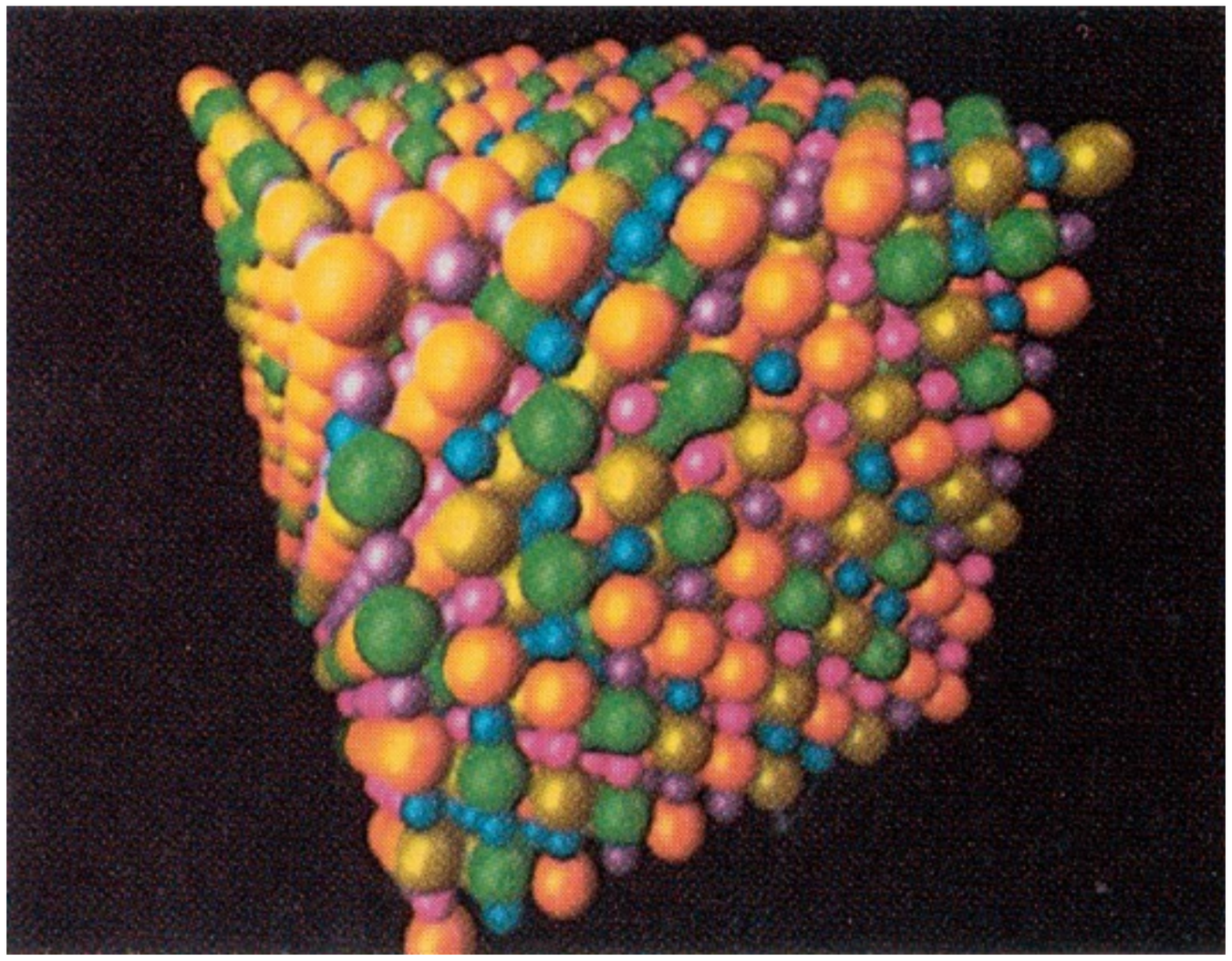
   – Stochastic Ray Tracing

   – Shading and Filtering

Basic Tiling

```
procedure AddTile takes
  e:   var Entry
  new: Tile
begin
  if not e.opaque or new.zmin ≤ e.zmax then
    { The new object may be visible: }
    if opaque and e.zmin > new.zmax then
      { The new object blocks all the old ones: }
      reclaim(e.tilelist)
      e.tilelist ← alloc TileList[new, nil]
      e.zmin ← new.zmin
      e.zmax ← new.zmax
      e.opaque ← new.opaque
    else
      { Add object to list and update entry: }
      e.tilelist ← alloc TileList[new, e.tilelist]
      e.zmin ← min(e.zmin, new.zmin)
      if e.opaque and new.opaque then
        e.zmax ← min(e.zmax, new.zmax)
      elseif not e.opaque and new.opaque then
        e.zmax ← new.zmax
      elseif not e.opaque then
        e.zmax ← max(e.zmax, new.zmax)
      endif
      e.opaque ← e.opaque or new.opaque
    endif
  endif
end procedure
```

Performance

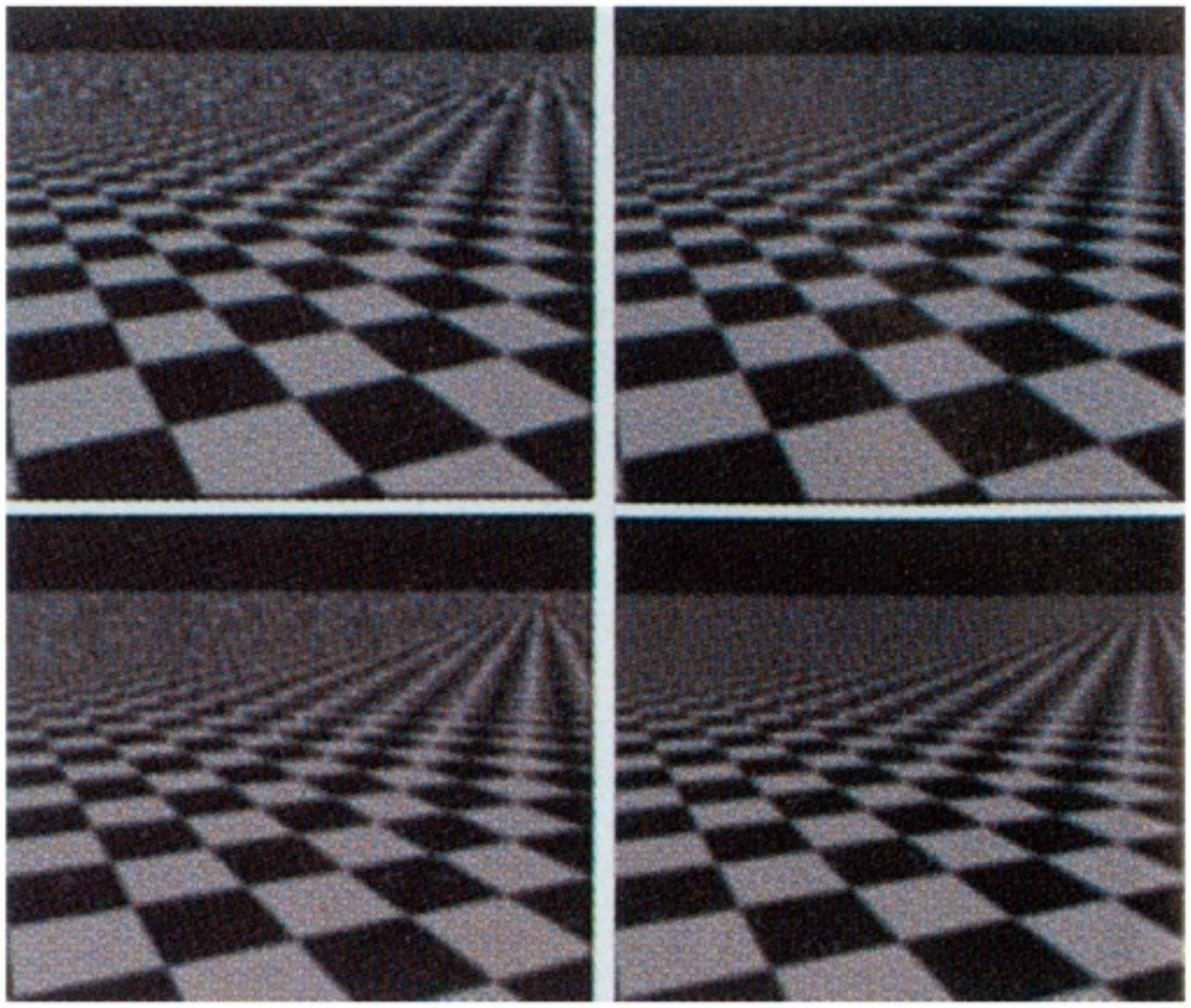


- Cost = Preprocess + Visibility + Shading

Anti-Aliasing

4 Samples

36 Samples



Regular

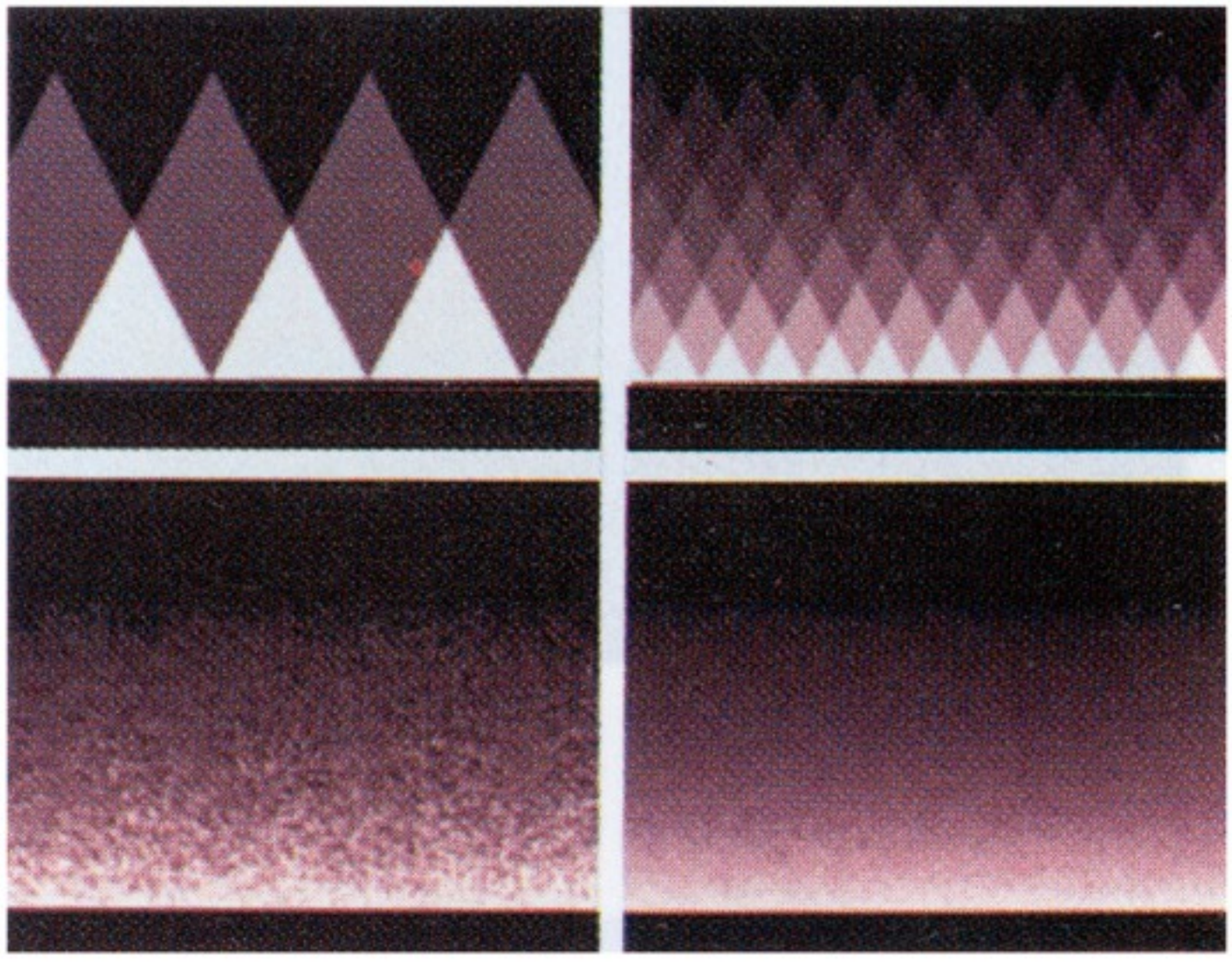
Jittered

- Stochastic Ray Tracing

Comparison

4 Samples

36 Samples



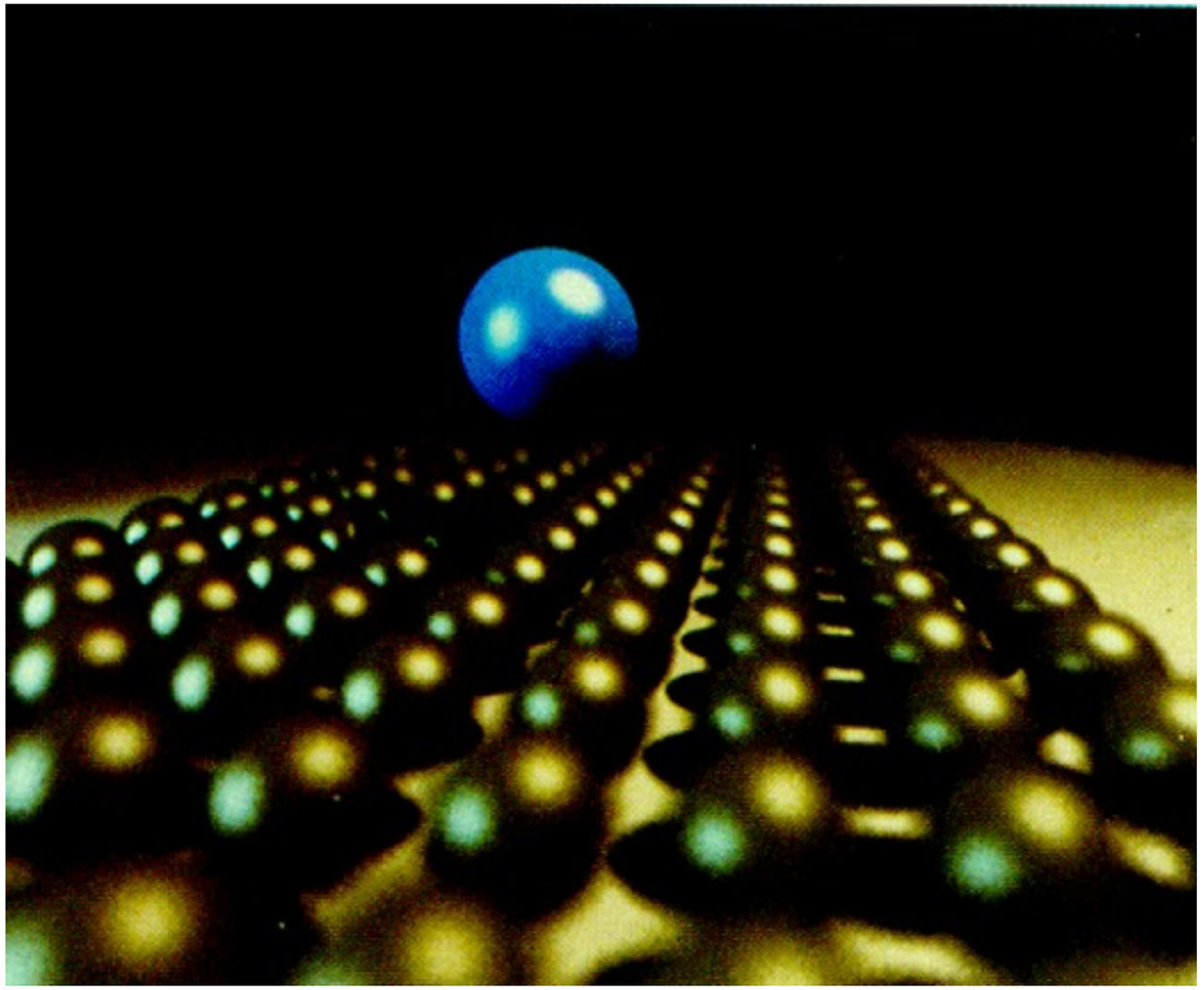
A-Buffer

ZZ-Buffer

- "comb" of 200 triangles (100 x 1.1 pixels)

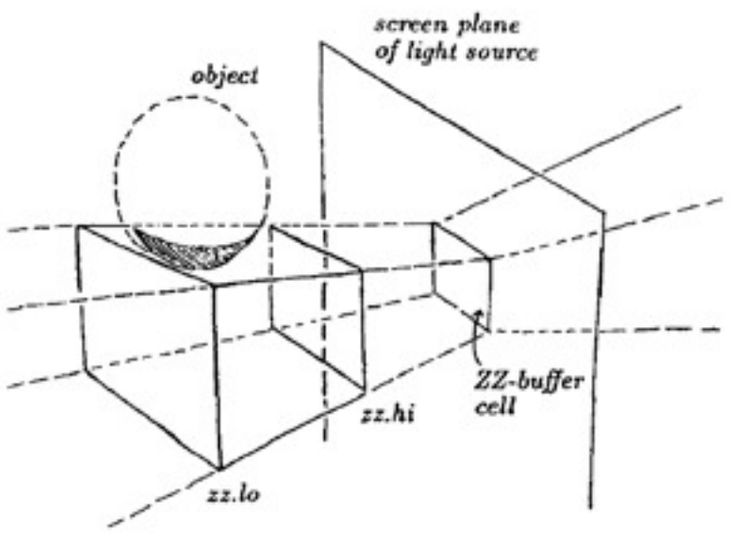
Rendering Effects

- Camera Depth of Field

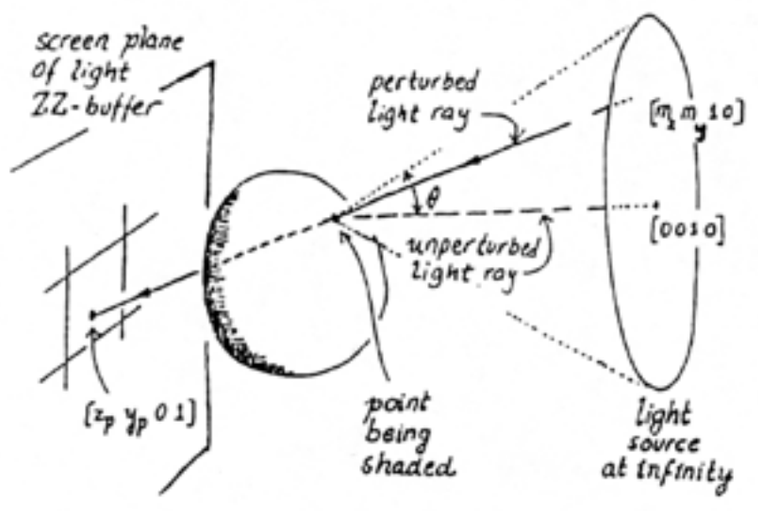


Extending to Shadows

- *Additional ZZ-Buffer per Light Source*
  - Two Small Changes:

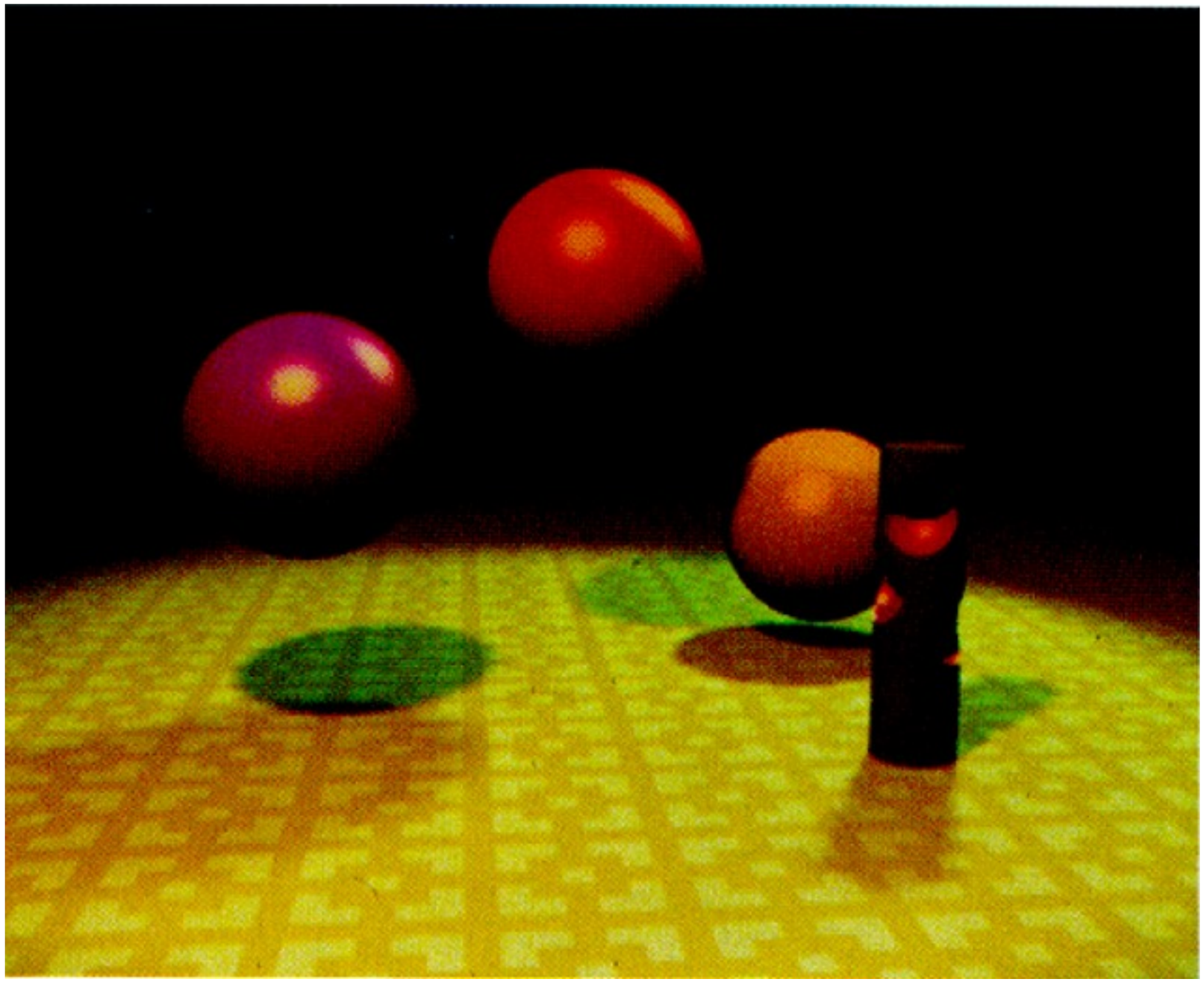


Computing Tiles for Penumbrae



Ray Tracing Area Light

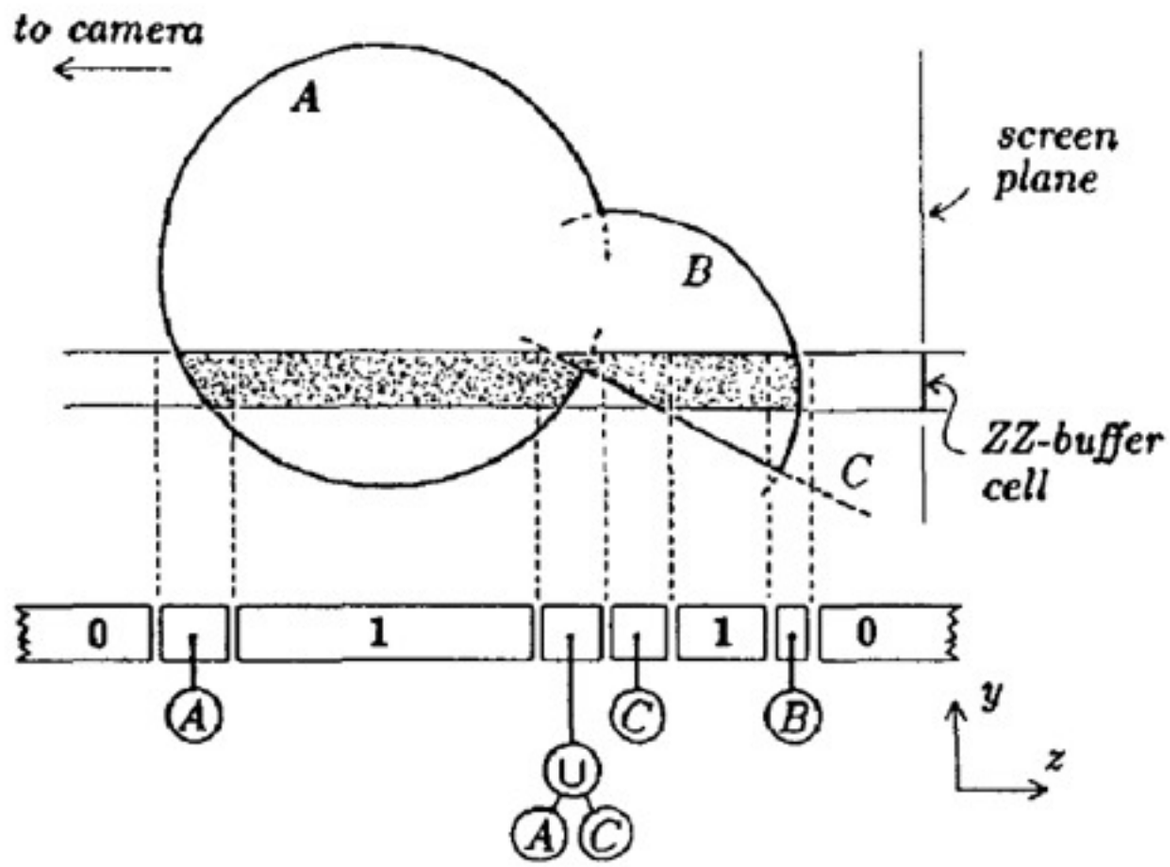
Soft Shadows



Extending to CSG

- CSG (sub)-Expressions

```
type Tile = record
   zz:    Interval
   expr:  CSGTree
   flags: TileFlags
end record
```



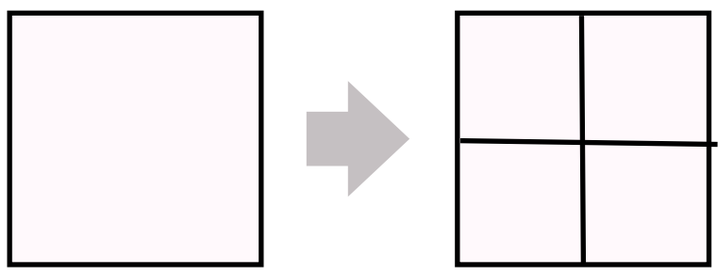
Screen-Space Subdivision

- Refine Tile List

1. Start with a Single ZZ-Cell covering the Image
2. Recursive Subdivision until ZZ-Cell is *simple*

(a) Split Cell

(b) Recompute Tile Lists



✳ *Warnock Algorithm*

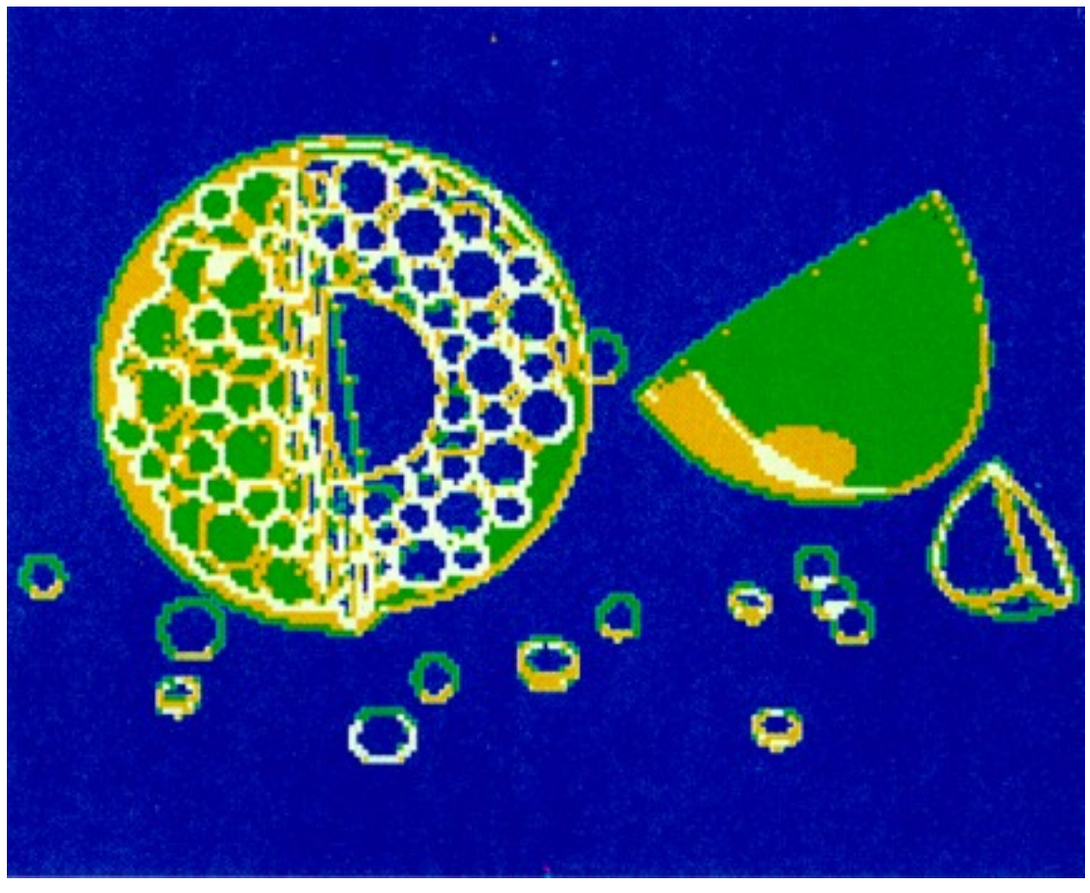
CSG Tiling

```
procedure RefineTileList
takes
   oldList: TileList
   newCell:Rectangle
returns TileList
begin
   newList ← NIL
   for each oldTile in oldList do
      auxList ← ComputeTiles (oldTile.expr , newCell )
      for each auxTile in auxList do
         if auxTile.zz ∩ oldTile.zz ≠ φ then
            newTile ← ClipTile (auxTile , oldTile.zz )
            newList ← Append (newList , newTile )
         end if
      end for
   end for
   return newList
end procedure
```

CSG Test

- Model: *500 spheres inside a translucent shell*

ZZ-Buffer (214x160 cells)



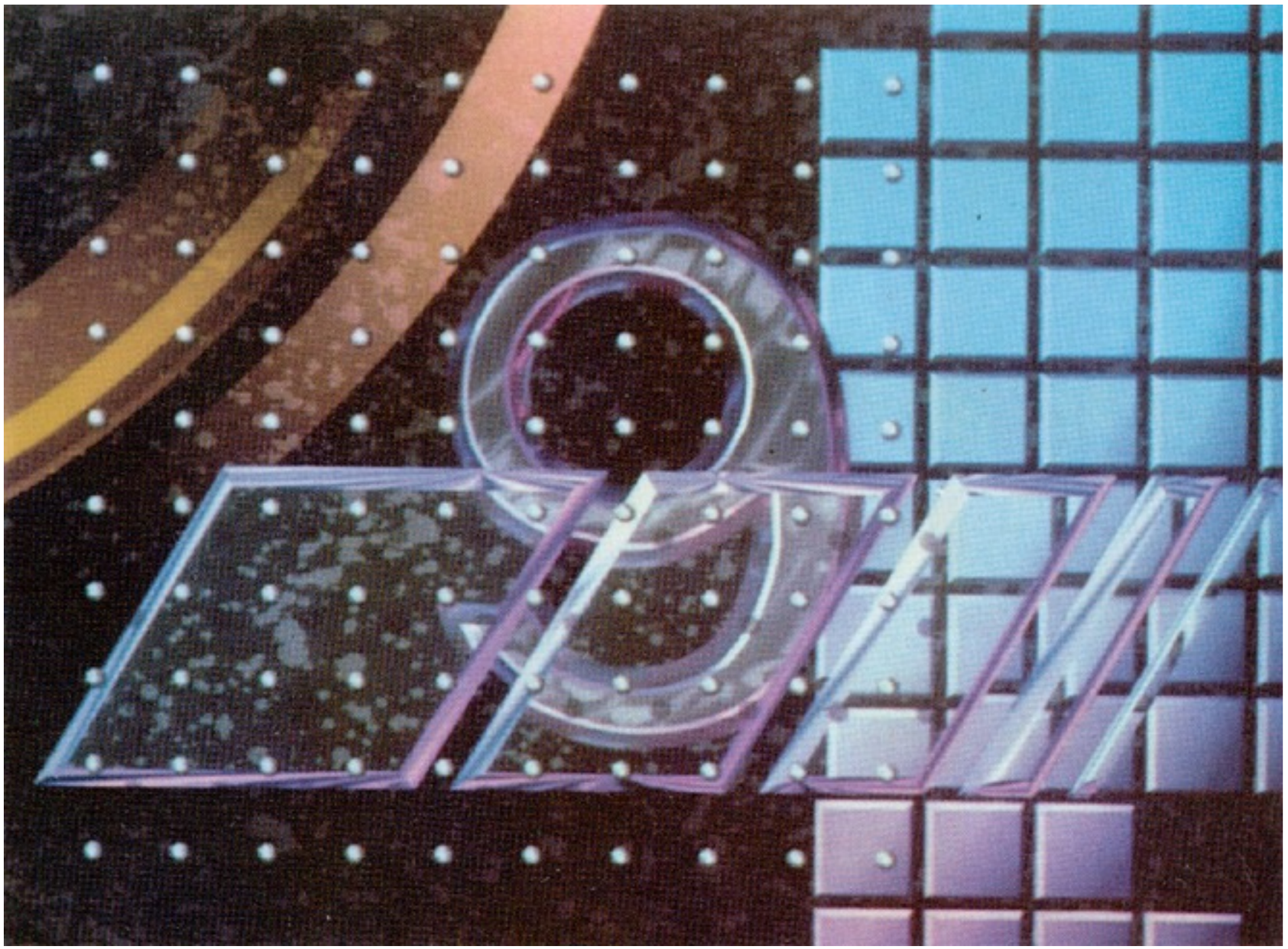
Number of Entries: 1 2 3 >

Implementations

- DEC Systems Research Center, Palo Alto
  - Experimental Software
- Sogitec, Paris
  - Production System
  - Commercial and Artistic Animations

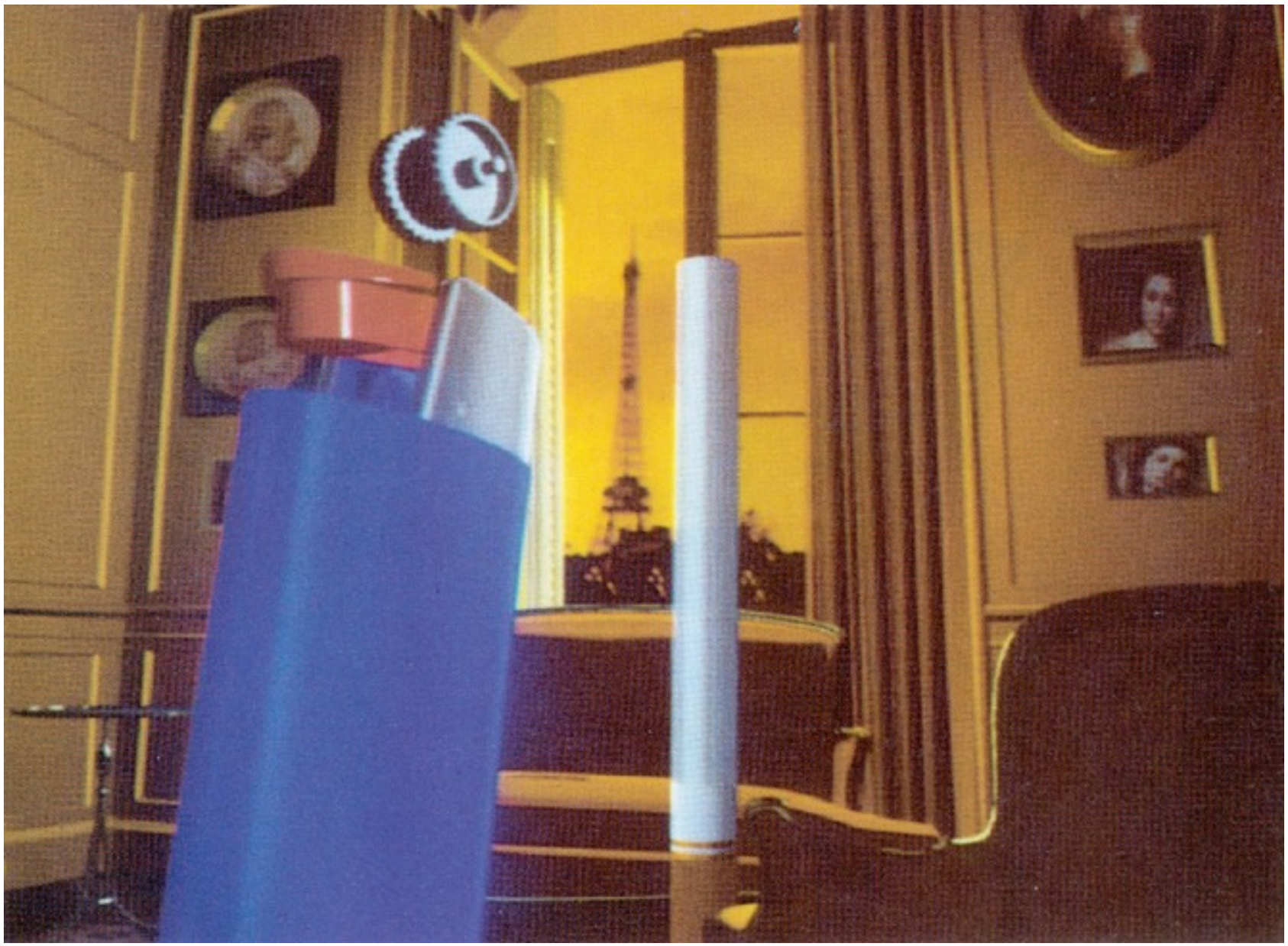
✳ *Many Licenses to Studios & Academia*

Countdown



- Sogitec

Jumpin' Jacques Splash



- Film and Video show, SIGGRAPH 1988