Scan-Line Algorithm

Lecture 8

```
Scan-Line
    sort objects by y, for all y {
        sort objects by x, for all x {
            compare z
        }
    }
```

One of the earliest algorithms for image generation.
1967-1974

This algorithm creates and image by processing the scene data on a line-by-line basis from the top to the bottom of the frame buffer.

Generally fast because:
1. Visibility decisions can be made in a 2-D subset of the 3-D scene
2. Coherency properties can be used to speed processing

1

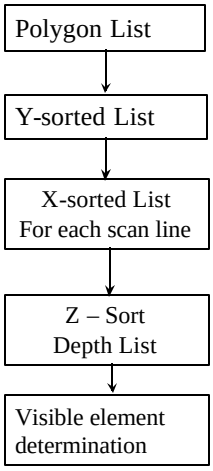
Scan-Line Algorithm

Lecture 8

- Use a Window that is one scan line high (constant Y value)
- Polygons intersections can be considered as a collection of line segments
- Visibility testing is then done on these line segments
- This reduces the 3-D problem to a 2-D problem

Typical Steps

- **Sort polygons with one y bucket per scan line**
- **For all active polygons on this scan line find and sort the x end points**
- **On a pixel by pixel basis sort out which line segment is closest**
- **Find the color for each pixel**
- **Move to the next scan line**
- **Update x end points for all polygons still in y bucket**
- **Add new x end points for new polygons.**



2

Scan-Line Algorithm

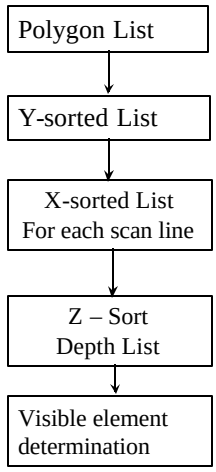
Lecture 8

The greatest variation in scan line algorithms occurs in the z sort or depth list

Worst case, a z-depth list must be generated for each pixel

If in the x sorted line list none of the objects cross each other, visibility can be found for large sections of the scan line.

If in the x sorted list lines do cross each others span, lists can be generated so that again a large number of pixels can be handle in one segment.



3

Painter's Algorithm

Lecture 8

```
Painter's
    sort objects by z, for all objects {
        for all covered pixels(x,y) {
            paint
        }
    }
```

Another of the earliest algorithms for image generation.
1969-1972

It can be thought of as painting, or filling, with opaque paint, where closer objects are painted over farther ones.

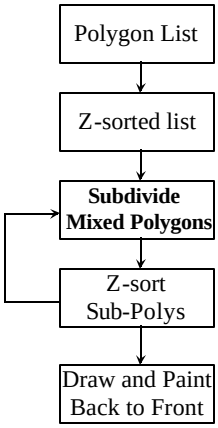
This algorithm solves the visible object problem by painting. It is simple but is not always the most efficient.

4

Painter's Algorithm

Lecture 8

- Start by sorting all objects by depth from front to back.
- Find all objects that have mixed Z.
- Subdivided these objects.
- Resort the sub objects by Z
- Repeat the above three steps until all object have clean Z values
- Start at the back and draw and fill all objects into the frame buffer.



5

Painter's Algorithm

Lecture 8

Z sorting is the key to the Painter's Algorithm.

- Simple and straight forward sort/subdivide.
- Binary-Space-Partitioning (BSP) tree.
- 2 1/2 D with *a priori* valid order (Compositing)
- Particle systems (Very simple to sort)

6

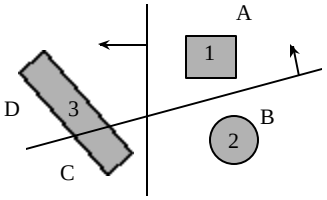
Painter's Algorithm BSP

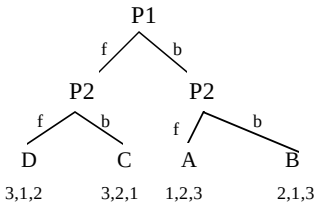
Lecture 8

Extremely efficient method for calculating the visibility relationships among a static group of polygons.

It is a trade off between a time and space intensive preprocessing step against a linear display step.

Well suited for applications in which the viewpoint changes, but were the objects do not.



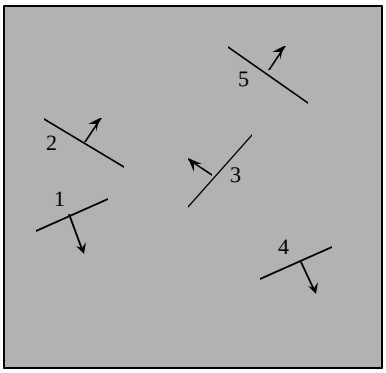


7

Painter's Algorithm BSP

Lecture 8

Example

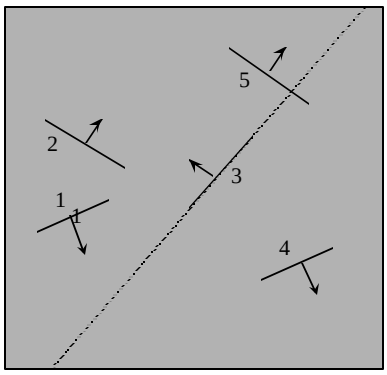


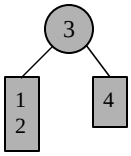
8

Painter's Algorithm BSP

Lecture 8

Example



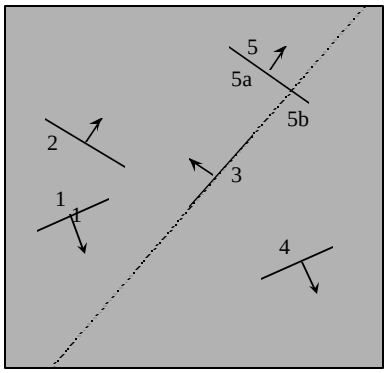


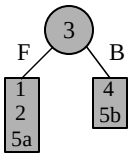
9

Painter's Algorithm BSP

Lecture 8

Example



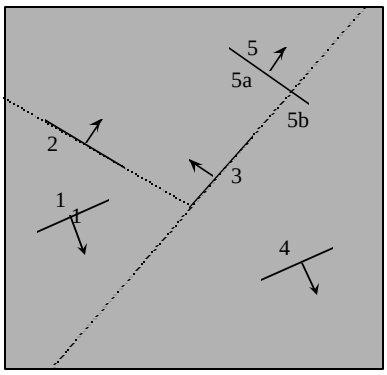


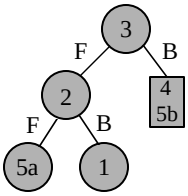
10

Painter's Algorithm
BSP

Lecture 8

Example



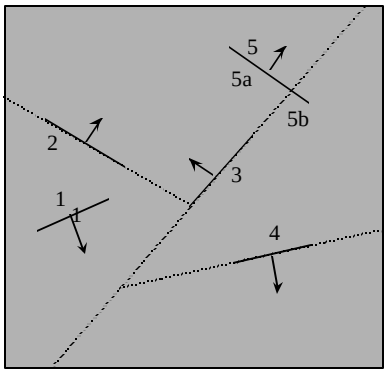


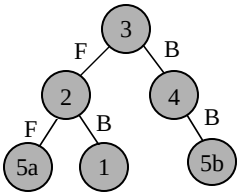
11

Painter's Algorithm
BSP

Lecture 8

Example



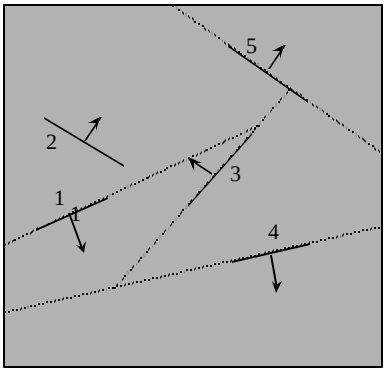


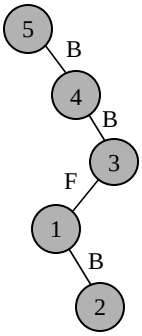
12

Painter's Algorithm
BSP

Lecture 8

Example II





13

Painter's Algorithm

Lecture 8

The "simplest" versions of the painters algorithm must do a shading calculation for every pixel of every object.

Some shading calculations can be avoided by painting some selected closest objects and then paint around them for the back objects.

The above can have serious problems because of requiring global order which can be very expensive to calculate.

With many Painter's algorithms aliasing can be a real problem. Using the alpha channel and compositing can help.

14

Painter's Algorithm

Lecture 8

Particle systems lend them selves to the Painter's Algorithm

Partical systems are made of small dynamic objects that can be created, extinguished, or moved. They are generally small and look like points which contain color and transparency. If they do have shape it can change with time along with color and material type.

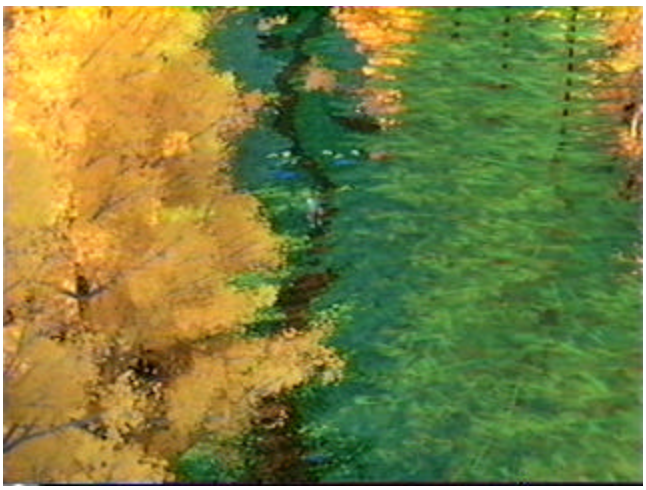
Being small they are easily sorted into fixed bins in space with little to no chance for z overlap.

Pseudo random location and motion tend to cause uncorrelated obscurations suitable for antialiased compositing.

15

Painter's Algorithm

Lecture 8



16

Painter's Algorithm

Lecture 8

17

Z-Buffer

Lecture 8

```
Z-Buffer
    for all objects {
        for all covered pixels {
            compare z
        }
    }
```

Of all the image rendering algorithms the Z-Buffer or Depth Buffer is one of the simplest.

For each pixel in the display buffer record the depth value of the object in the scene. Start with the nearest object in each pixel and determine the shading values.

This algorithm lends it self to easy anti-aliasing

18

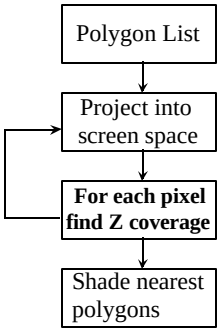
Z-Buffer

Lecture 8

Development of the Z-Buffer is attributed to Ed Catmull in 1974.

Many additions have been add including

- •Alpha buffers
- •Jittered Samples
- •Weighted averages anti-aliasing
- •Bi-cubic patch subdivision
- •Depth maps
- •Shadows



19

Z-Buffer

Lecture 8

Two Papers that are a must for understanding Z-buffer rendering systems.

"The Reyes Image Rendering Architecture"

"The A-Buffer, an Antialiased Hidden Surface Method"

20

Reyes Rendering System

Lecture 8

Design Principles:

- **Natural Coordinates**
- **Vectorization**
- **Common Representation**
- **Locality**
  - **Geometric**
  - **Texture**
- **Linearity**
- **Large Models**
- **Back Door**
- **Texture Maps**

21

The Alpha Buffer

Lecture 8

The Alpha Buffer is

- **Antialiased**
- **Area-averaged**
- **accumulation Buffer**

Resolves visibility of an arbitrary collection of opaque and transparent surfaces.

Increases image resolution many times over the standard Z-Buffer

22

Alpha & Z-Buffer

Lecture 8

Intensity

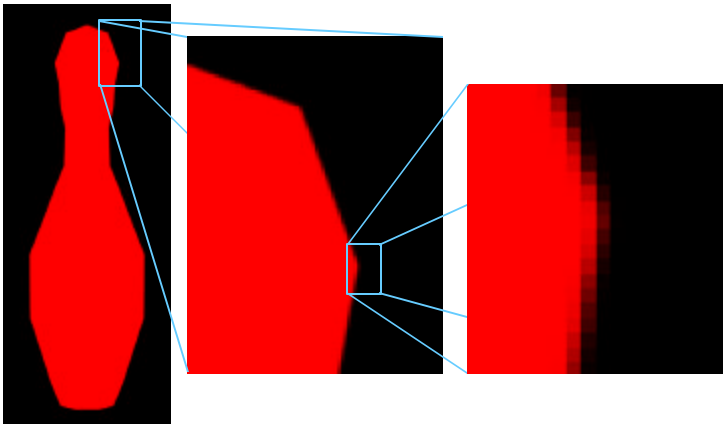
Depth Buffer

Alpha

23

Alpha

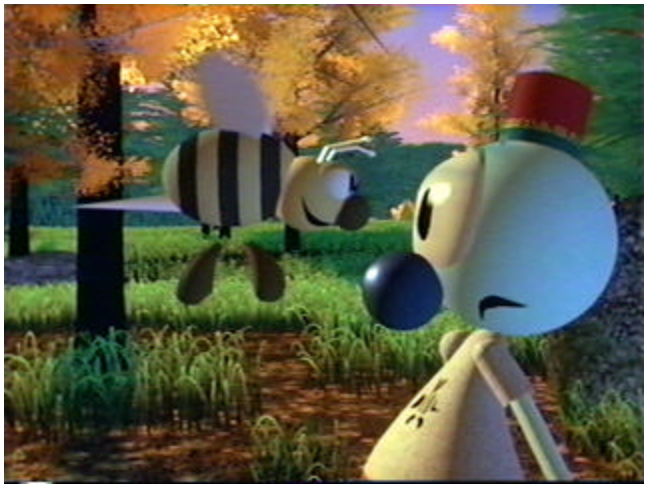
Lecture 8



24

Compositing Example

Lecture 8



25

Compositing Example

Lecture 8

26

Compositing Example

Lecture 8

27

Compositing Example

Lecture 8

28

Compositing Example

Lecture 8

29

Compositing

Lecture 8

Important Papers:

“Compositing Digital Images”

“Compositing 3-D Rendered Images

“Compositing - Theory”

“Composting - Practice”

30