Rendering 3D Scenes

Luiz Velho
IMPA

Outline

- Plenoptic Function / Light Fields
- Illumination
- Cameras
- Viewing Pipeline

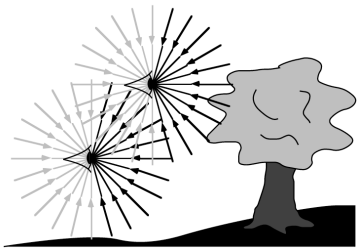
- Rendering Architectures

Plenoptic Function

Complete description of Visual Information in a 3D environment

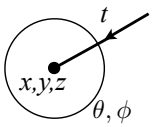
- $I_\lambda = P(x, y, z, \theta, \phi, t)$

  Holographic Image



- $P : \mathbb{R}^3 \times \mathbb{S}^2 \times \mathbb{R} \mapsto \mathcal{E}$

  6D Phase Space



OBS: No Explicit Geometry - Scene Modeling to the Rescue !

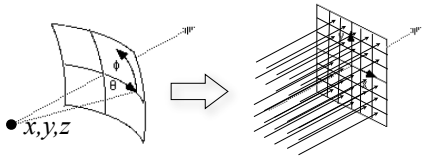
Light Field

A Slice of the Plenoptic Function

✳ Structured Sampling of P

- Point Sampling (Inverse Rendering)

  - Pinhole *Camera Model*

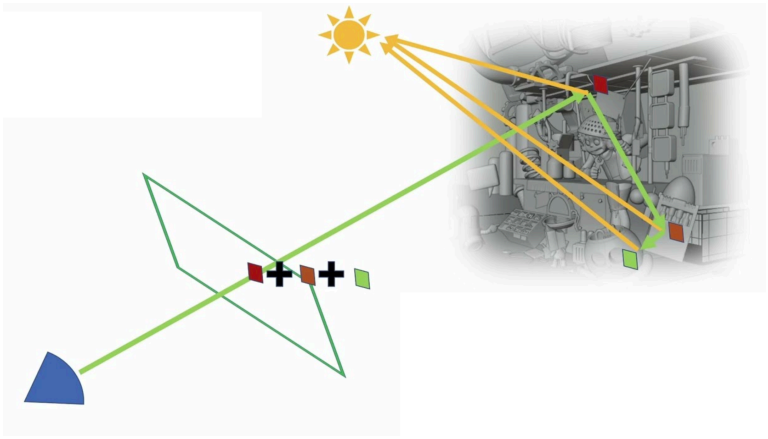


Ray Space

(x,y,z) Viewpoint / (θ, ϕ) Field of View (FOV)

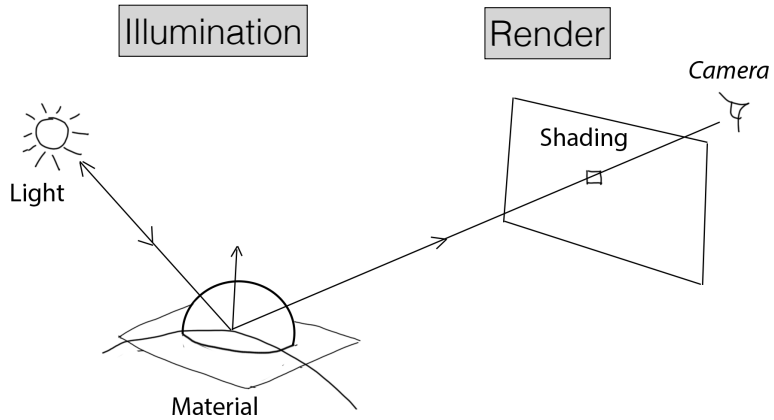
Synthesizing the Plenoptic Function

- Illumination
  - Light Sources
  - Materials
- Rendering
  - Shading



Conceptual Model

- **Data Driven Computation**



Illumination

Illumination

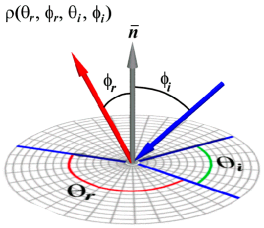
Study of Light Emission and Propagation

- Light
- Materials
- Transport

Local Illumination

Illumination Mechanisms

- Light Transport
  - Coherent 
  - Incoherent 
- Bidirectional Transport Function
  - *BRDF* - Reflection
  - *BTDF* - Transmission



Study Topics

Plenoptic Function	→	Light Field (*)
Cameras	→	Images

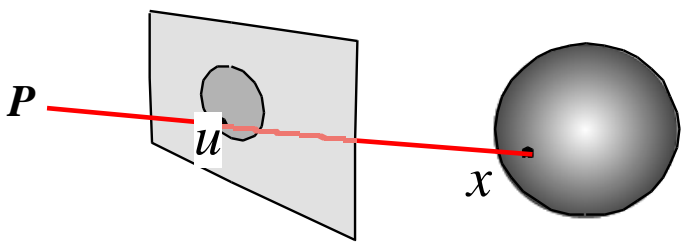
- Projective Geometry
- Camera Calibration

Projective Geometry
& Cameras

Mathematical Fundamentals

- Projective Geometry

$$2D \longleftrightarrow 3D$$

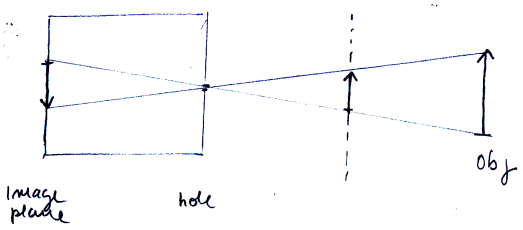


$$u = Px \qquad u \in \mathbb{R}^2, \quad x \in \mathbb{R}^3$$

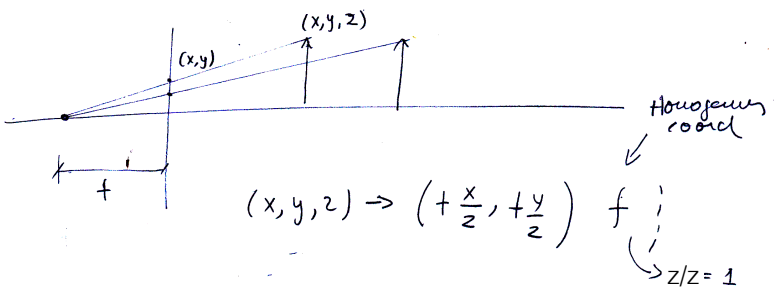
Projective Geometry

- Camera Transform
  - Projection (3D => 2D)
    - 2D Image $u \in I \subset \mathbb{R}^2$
    - 3D Space $x \subset \mathbb{R}^3$
    - Camera $u = Px$
- Types of Projection
  - Orthographic (Affine)
  - Perspective
  - Etc..

Pinhole Camera Model

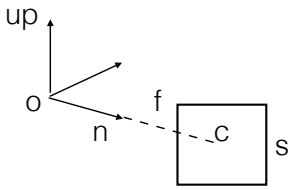


- Perspective Projection.



Vision / Graphics

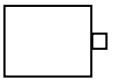
- Camera Model

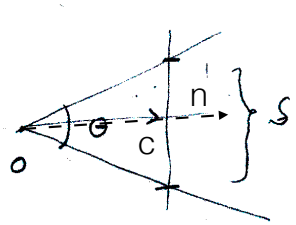


- Camera Parameters
  - Position: ***o***
  - Orientation: ***up , n***
  - Focal Distance: ***f***
  - Image Center: ***c***
  - Image Size: ***s***

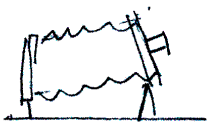
Field of View

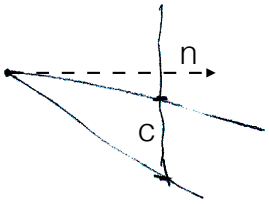
- Pinhole Camera





- View Camera





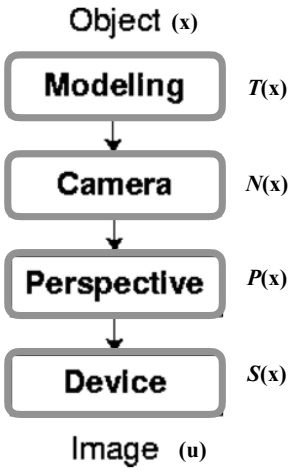
Viewing Transformations

Viewing Operations

- **Camera Transformation**
- **Clipping**
- **Projection**
- **Rasterisation**
- **Visibility**

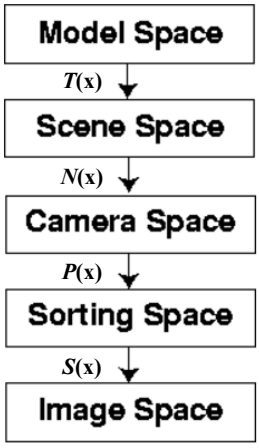
Viewing Pipeline

``Sequence of transformations mapping 3D objets to viewport''



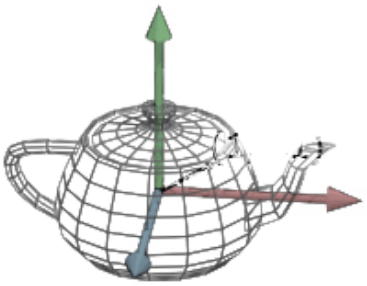
Reference Spaces

``Objects are transformed to coordinate systems where viewing operations are performed''



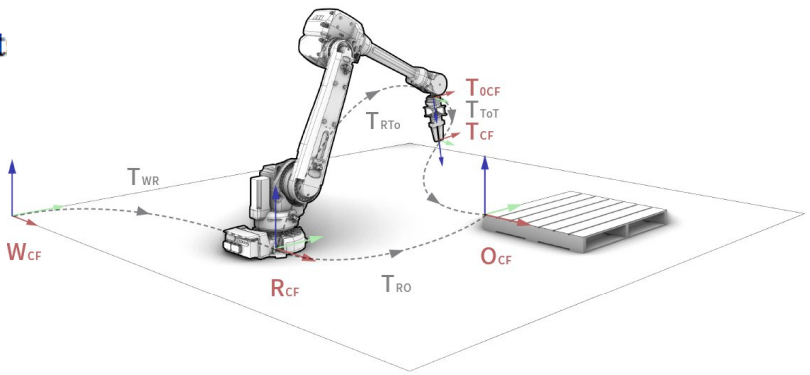
Model Space

- Characteristics
  - Object Coordinate System
  - Origin: Center of Object
  - Principal Axis (Object)
  - Normalized
- Operation
  - Modeling Ops



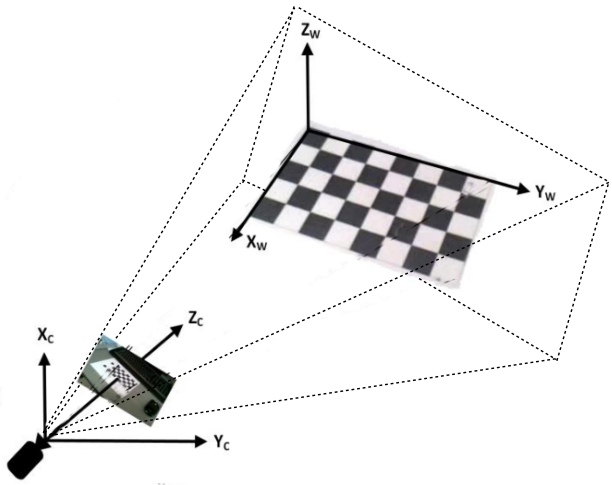
Scene Space

- Characteristics
  - Global Coordinate System
  - All Objects
  - Application Unit
- Operation
  - Shading



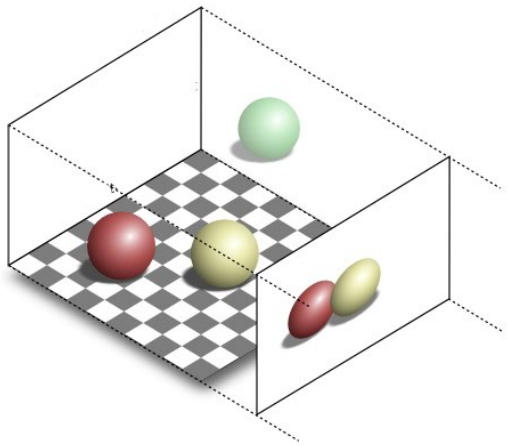
Camera Space

- Characteristics
  - Camera Coordinate System
  - XY Plane: Image Plane
  - Z Axis: View Direction
  - Normalized Fustrum
- Operation
  - Clipping



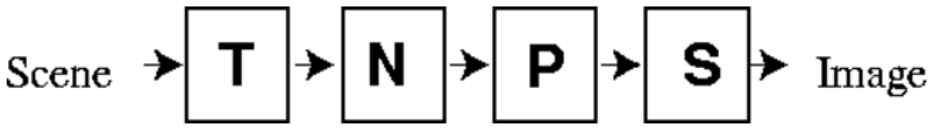
Sorting Space

- Characteristics
  - Sorting Coordinate System
  - Viewpoint at Infinity
  - Preserve Linear Structures
- Operation
  - Visibility Computation

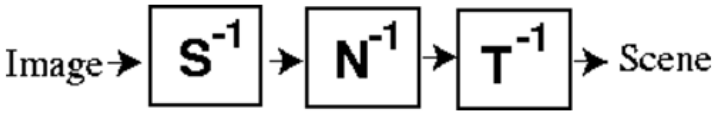


Viewing Pipeline

- Parametric Models



- Implicit Models



P^{-1}

Object Centered Pipeline

- **Projection + Raster**

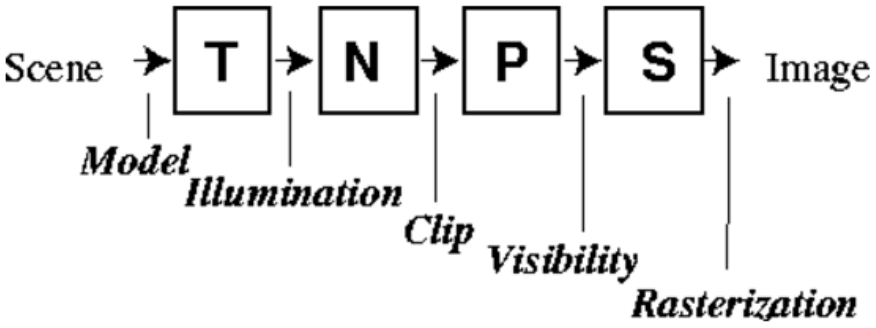
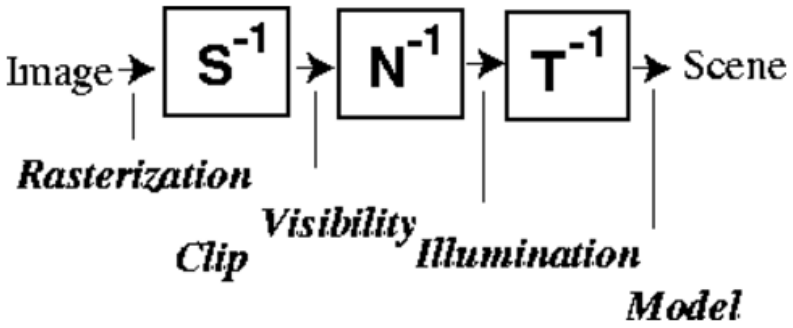


Image Centered Pipeline

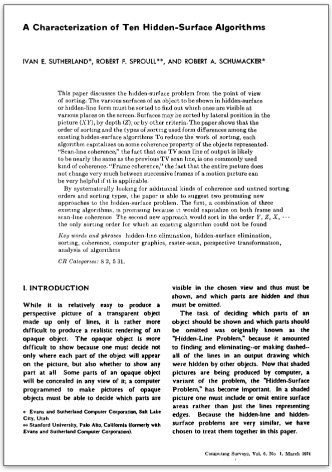
- **Ray Casting**



Rendering Architectures

A Bit of History

- From Visibility to Rasterization



A Characterization of Ten Hidden-Surface Algorithms

IVAN E. SUTHERLAND*, ROBERT F. SPROULL**, AND ROBERT A. SCHUMACKER*

This paper discusses the hidden-surface problem from the point of view of sorting. The various surfaces of an object to be shown in hidden-surface or hidden-line form must be sorted to find out which ones are visible at various places on the screen. Surfaces may be sorted by lateral position in the picture (XY), by depth (Z), or by other criteria. The paper shows that the order of sorting and the types of sorting used form differences among the existing hidden-surface algorithms To reduce the work of sorting, each algorithm capitalizes on some coherence property of the objects represented. "Scan-line coherence," the fact that one TV scan line of output is likely to be nearly the same as the previous TV scan line, is one commonly used kind of coherence. "Frame coherence," the fact that the entire picture does not change very much between successive frames of a motion picture can be very helpful if it is applicable.

By systematically looking for additional kinds of coherence and untried sorting orders and sorting types, the paper is able to suggest two promising new approaches to the hidden-surface problem. The first, a combination of three existing algorithms, is promising because it would capitalize on both frame and scan-line coherence The second new approach would sort in the order $Y, Z, X, \cdots$ the only sorting order for which an existing algorithm could not be found

Key words and phrases hidden-line elimination, hidden-surface elimination, sorting, coherence, computer graphics, raster-scan, perspective transformation, analysis of algorithms

CR Categories: 8 2, 5 31.

I. INTRODUCTION

While it is relatively easy to produce a perspective picture of a transparent object made up only of lines, it is rather more difficult to produce a realistic rendering of an opaque object. The opaque object is more difficult to show because one must decide not only where each part of the object will appear on the picture, but also whether to show any part at all Some parts of an opaque object will be concealed in any view of it; a computer programmed to make pictures of opaque objects must be able to decide which parts are visible in the chosen view and thus must be shown, and which parts are hidden and thus must be omitted.

The task of deciding which parts of an object should be shown and which parts should be omitted was originally known as the "Hidden-Line Problem," because it amounted to finding and eliminating--or making dashed--all of the lines in an output drawing which were hidden by other objects. Now that shaded pictures are being produced by computer, a variant of the problem, the "Hidden-Surface Problem," has become important. In a shaded picture one must include or omit entire surface areas rather than just the lines representing edges. Because the hidden-line and hidden-surface problems are very similar, we have chosen to treat them together in this paper.

* Evans and Sutherland Computer Corporation, Salt Lake City, Utah
** Stanford University, Palo Alto, California (formerly with Evans and Sutherland Computer Corporation).

Computing Surveys, Vol. 6, No 1, March 1974

1974

Computer Graphics, Volume 23, Number 3, July 1989

A Characterization of Ten Rasterization Techniques

Nader Gharachorloo
Satish Gupta
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598

Robert F. Sproull
Ivan E. Sutherland
Sutherland, Sproull, and Associates, Inc.
4516 Henry Street, Pittsburgh, PA 15213

Abstract

With widespread use of raster scan displays and the ever-increasing desire for faster interactivity, higher image complexity, and higher resolution in displayed images, several techniques have been proposed for rasterizing primitive graphical objects. This paper characterizes the performance of these techniques and shows how they evolve for more complex images on higher resolution displays. This characterization will not only show the strengths and deficiencies of existing rasterization techniques, but will also reveal new architectures for future raster graphics systems.

Introduction

The raster-scan display is now the most popular computer output device because of the rapid decrease in the cost of semiconductor memories and the low cost of raster-scan CRTs. Such displays are now used in a wide range of applications, from home computers to engineering design to flight simulation.

Raster-scan displays are most commonly driven by a frame buffer, a memory that stores the color value for every picture element on the screen and refreshes the display continuously. The process of generating these picture element values from a geometric description of the image is known as rasterization. The principal strength of the frame buffer is that it can show an arbitrary image on the display with an arbitrary number of primitive objects, subject only to the limit of spatial and intensity resolution of the display device. The weakness of a frame buffer is that a great many bits must be changed to change the picture. Because of their large size, frame buffers are usually implemented using the highest density, and hence the slowest, memory chips. The difficulty of accessing pixel data rapidly in such frame-buffer memories can be overcome by several different techniques. This paper will characterize ten such techniques.

These techniques will be characterized along three aspects of the display user's desires: interactivity, image complexity, and resolution. *Interactivity* is determined by how rapidly the image is updated and hence how rapidly the user can interact with his application. *Image complexity* is characterized by the number of the primitive objects (shaded triangles, vectors, characters) displayed. *Resolution* refers strictly to the number of pixels on the display device and will provide a measure of how these techniques are likely to evolve as resolution increases. Examining the full spectrum of rasterization techniques will help predict the evolution of future graphics architectures.

Background

Figure 1 shows a typical rasterization system based on the frame buffer. The host computer generates graphical primitives, the rasterization processor converts these primitives into pixel values and stores them in the frame buffer, which is accessed continuously to refresh the screen. Any rasterization system must be able to receive graphical primitives, rasterize these primitives, and continuously refresh the raster display. Given that all rasterization systems must satisfy these common input and output requirements, the only fundamental difference among rendering systems is the frame-buffer memory organization and the rasterization processor to match it.

Before examining the details of different rasterization techniques, it is necessary to have an understanding of the primitives being rasterized, the sources of parallelism in the rendering problem, and the organization of the rasterization processor.

Rasterization Primitives

We have chosen three primitives for characterization: Gouraud-shaded three-dimensional triangles, two-dimensional vectors, and characters. With the advent of windowing systems, it is desirable that the same rasterization system be capable of efficiently rendering all three types of primitives on the same screen. Higher-level curve and surface primitives can be broken down into these basic primitives, i.e. polylines and triangular meshes.

Three-dimensional triangles are represented by three vertices, each of which is described by x,y,z screen coordinates and an r,g,b color triplet. The triangle is rasterized by linearly interpolating the depth and color values for each pixel inside the triangle and conditionally updating the frame buffer and the z-buffer values only if the triangle is visible [25]. Most systems use 8 or 24 bits per pixel for storing color values and 16 or more bits per pixel for storing depth information.

Given P triangles of average pixel area A, a total of $P \times A$ pixels must be accessed to render the scene. Assuming the

©1989 ACM-0-89791-312-4/89/007/0355 $00.75

355

1989

Main Algorithms

- ✦ Projection + Rasterization
  - Object Space
    - Pre-Compute Order of Pieces
    - (Visibility First)
- ✦ Ray Casting
  - Image Space
    - Ordering at Each Ray
    - (Visibility Last)

Looking Ahead

- **Sistemas Gráficos 3D**

  - Descrição
  - Conteúdo
  - Software

  - Notas de Aula
  - Metodologia

  - Padrões Gráficos