Rendering Pipeline

Erickson Nascimento
erickson@dcc.ufmg.br
www.dcc.ufmg.br/~erickson
UFMG

1

Computer Graphics Challenges

- Suppose we can create a precise computer representation of the 3D world
- Questions:
  - How can we generate realistic 2D?
  - What are the best way to model 3D world?
  - How to render such model?

UFMG

2

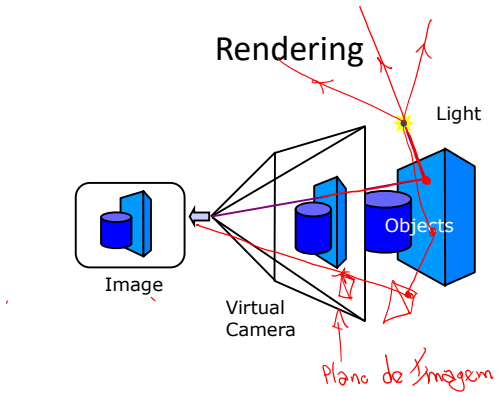
Rendering

- Synthesizing an image given
  - objects
  - virtual cameras
  - light sources
- Techniques
  - rasterization
  - ray casting

UFMG

3

Rendering



4

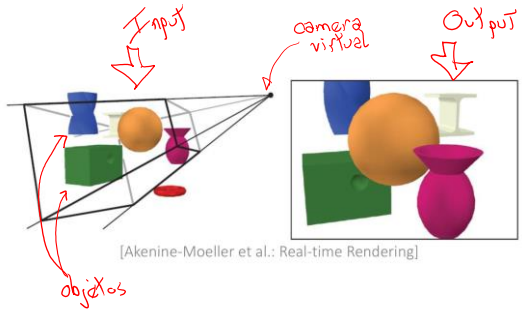
I/O in the rendering pipeline

- Input:
  - Geometric model (normal, color, texture coordinates, pose)
  - Lighting model (direction, position, intensity)
  - Virtual Camera (focal length, pose, etc.)
  - Raster viewport (pixel size, etc.)
- Output:
  - Colors/intesities (image)

UFMG

5

I/O in the rendering pipeline



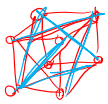
[Akenine-Moeller et al.: Real-time Rendering]

UFMG

6

Object representation

- 3D objects can be represented by **primitives** like
  - vertices (i.e., points)
  - lines
  - polygons (e.g., triangles)
- Primitives are processed to generate a 2D image



UFMG

7

Object representation

- List of vertices and attributes
  - X, Y, Z (3D coordinates)
  - R, G, B (color)
  - u, v (texture coordinates)
- Geometric information
  - Positions, normals, curvature
- List of triangles
  - How are the triangles connected (Topological information) ?

UFMG

8

Manifold

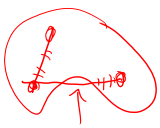
- Manifold
  - Every edge has exactly two incident triangles
- Manifold with boundaries
  - Every edge has one or two incident triangles

UFMG

9

Why do we use triangles?

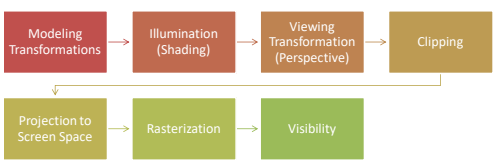
- Minimal planar primitives
- Piecewise linear representation
- Easy to implement in hardware
- Easy to interpolate attributes
  - Convex Linear Interpolation



UFMG

10

Big picture



11

Rendering Pipeline

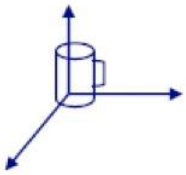
- Input
  - Soup of 3D triangles
  - Camera and illumination configuration
- Output
  - 2D image
- Each triangle is sent through it in the pipeline

UFMG

12

Modeling Transformations

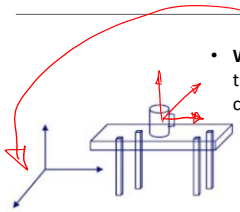
- **Object space**: 3D models defined in their own coordinate system object space
  - Designing objects and not worrying about scene
  - Library of objects



UFMG

13

Modeling Transformations



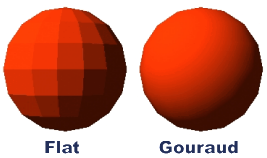
- **World space**: transforms orient the models within a common coordinate frame

UFMG

14

Illumination (Shading)

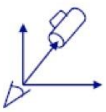
- Vertices shaded according to material properties, surface properties (normal) and light sources
- Local lighting model
  - Diffuse
  - Ambient
  - Phong



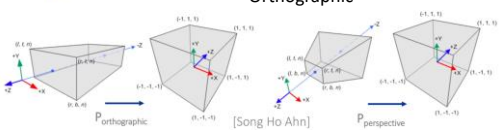
UFMG

15

Viewing Transformation



- Project world space to **eye space**
- Projection models
  - Perspective
  - Orthographic

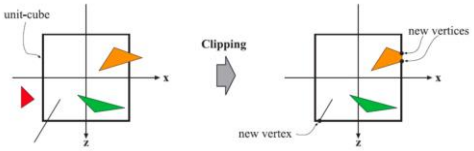


UFMG

16

Clipping

- Portions of the object outside the view volume (view frustum) are removed

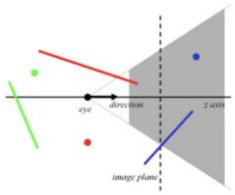


UFMG

17

Clipping

- Eliminate portions of objects outside the viewing frustum
- Avoid degeneracies
  - Don't draw stuff behind the eye
  - Avoid division by 0 and overflow
- Efficiency
  - Do not waste time on objects outside the image boundary

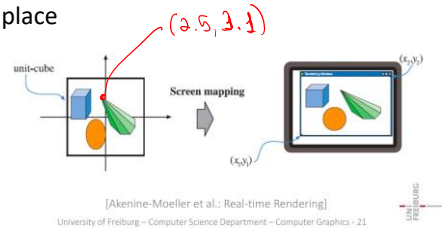


UFMG

18

Projection to Screen Space

- The objects are projected to the 2D image place



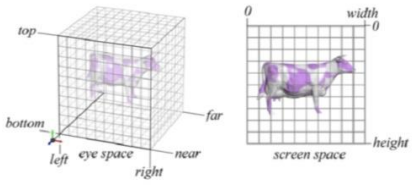
[Akenine-Moeller et al.: Real-time Rendering]

University of Freiburg – Computer Science Department – Computer Graphics - 21

UFMG

19

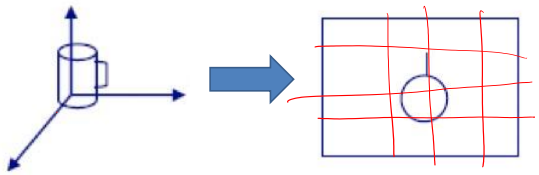
Projection to Screen Space



UFMG

20

From object space to screen space

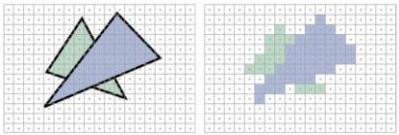


UFMG

21

Rasterization

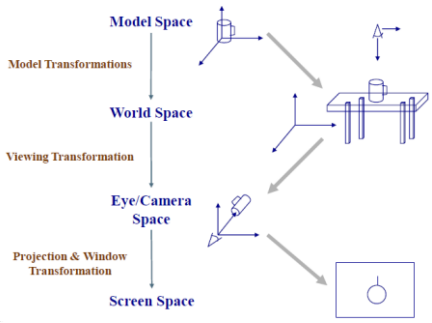
- Scan Conversion
  - Interpolate values (color, depth, etc.) and draw the objects
  - Our primitives are continuous, but the screen is discrete



UFMG

22

Rendering Pipeline



UFMG

23