Einführung in Visual Computing

object coordinates

world coordinates

camera coordinates

normalized device coordinates

pixel coordinates

Graphics Pipeline

Werner Purgathofer
Graphics Pipeline

- information is transformed to an image in successive steps
  - object and scene creation
  - definition of view (camera)
  - projection
  - rasterization

- this is called the **graphics pipeline**
  (or **viewing pipeline**, **transformation pipeline**, **rendering pipeline**, ...)

Werner Purgathofer
Transformation Pipeline

- object coordinates
- world coordinates
- camera coordinates
- normalized device coordinates
- pixel coordinates

Transformation steps:

- model transformation
- view transformation
- projection + homogenization
- viewport transformation
Rendering Pipeline (Technical Implementation)

scene objects in object space

transformed vertices in clip space

scene in normalized device coordinates

raster image in pixel coordinates

object capture/creation

modeling

viewing

projection

clipping + homogenization

viewport transformation

rasterization

shading

pixel stage ("fragment shader")

vertex stage ("vertex shader")

Werner Purgathofer
Einführung in Visual Computing

186.822

Graphics Pipeline

The End
arbitrary graph implemented with pointers
Reminder: Graphs and Trees

(binary tree implemented with pointers)

+ recursive evaluation
  “pre-order”: node-left-right 1-2-4-5-3-6-7
  “in-order”: left-node-right 4-2-5-1-6-3-7
  “post-order”: left-right-node 4-5-2-6-7-3-1
Einführung in Visual Computing

3D Object Representations
Object Representation in the Rendering Pipeline

scene objects in object space

transformed vertices in clip space

scene in normalized device coordinates

raster image in pixel coordinates

- object capture/creation
- modeling
- viewing
- projection
- clipping + homogenization
- viewport transformation
- rasterization
- shading

vertex stage ("vertex shader")

pixel stage ("fragment shader")
3D Object Representations

- graphics scenes contain
  - solid geometric objects
  - trees, flowers, clouds, rocks, water,...
- creation of models
  - surface ↔ interior models
  - explicit ↔ procedural models
  - heuristically ↔ physically based models
- representations
  - geometrical data structures
  - data structure organization
set of surface polygons enclose object interior

= **Boundary Representation**

("B-Rep")

equivalent example:

*machine part surface represented by triangles*
polygon tables (B-Rep lists)
- geometric and attribute tables
- vertex, edge, polygon tables
- consistency, completeness checks
Polygon Surfaces: Data Structure

**VERTEX TABLE**

<table>
<thead>
<tr>
<th>$V_1$</th>
<th>$x_1, y_1, z_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_2$</td>
<td>$x_2, y_2, z_2$</td>
</tr>
<tr>
<td>$V_3$</td>
<td>$x_3, y_3, z_3$</td>
</tr>
<tr>
<td>$V_4$</td>
<td>$x_4, y_4, z_4$</td>
</tr>
<tr>
<td>$V_5$</td>
<td>$x_5, y_5, z_5$</td>
</tr>
</tbody>
</table>

**EDGE TABLE**

<table>
<thead>
<tr>
<th>$E_1$</th>
<th>$V_1, V_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_2$</td>
<td>$V_2, V_3$</td>
</tr>
<tr>
<td>$E_3$</td>
<td>$V_3, V_1$</td>
</tr>
<tr>
<td>$E_4$</td>
<td>$V_3, V_4$</td>
</tr>
<tr>
<td>$E_5$</td>
<td>$V_4, V_5$</td>
</tr>
<tr>
<td>$E_6$</td>
<td>$V_5, V_1$</td>
</tr>
</tbody>
</table>

**POLYGON-SURFACE TABLE**

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$E_1, E_2, E_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_2$</td>
<td>$E_3, E_4, E_5, E_6$</td>
</tr>
</tbody>
</table>
Lists for B-Reps

surface list

edge list

vertex list

Werner Purgathofer
Reminder: Product of Vectors

\[ V_1 = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} \quad V_2 = \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix} \]

- **scalar product:**
  \[ V_1 \cdot V_2 = ? \]

- **cross product (vector product):**
  \[ V_1 \times V_2 = ? \]
Reminder: Product of Vectors

**Scalar product:**

\[
V_1 \cdot V_2 = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} \cdot \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix} = a_1 a_2 + b_1 b_2 + c_1 c_2
\]

\[
V_1 \cdot V_2 = |V_1||V_2| \cos \phi
\]

**Cross product (vector product):**

\[
V_1 \times V_2 = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} \times \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix} = \begin{pmatrix} b_1 c_2 - c_1 b_2 \\ c_1 a_2 - a_1 c_2 \\ a_1 b_2 - b_1 a_2 \end{pmatrix}
\]

\[
|V_1 \times V_2| = |V_1||V_2| \sin \phi
\]
Polygon Surfaces: Plane Equation

\[ Ax + By + Cz + D = 0 \]

- Plane parameters \( A, B, C, D \)
- Normal \( (A, B, C) \)

\( N = (1, 0, 0) \)

Werner Purgathofer
Front and Back Polygon Faces

- **back face** = polygon side that faces into the object interior
- **front face** = polygon side that faces outward
- **behind** a polygon plane = visible to the polygon back face
- **in front of** a polygon plane = visible to the polygon front face

[front = the side to which \((A,B,C)\) points]
Ax + By + Cz + D = 0 for points on the surface
< 0 for points behind
> 0 for points in front

if (1) right-handed coordinate system
(2) polygon points are ordered counterclockwise

\[ \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3 \text{ counterclockwise } \Rightarrow \text{ normal vector } \mathbf{N} = (\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_3 - \mathbf{V}_1) \]
Triangle Meshes

- most polygons are triangles
- **triangle mesh** = connected triangles
- **triangle-strip** = successive triangles
  (1 additional point per triangle)
Constructive Solid Geometry

- Constructive Solid Geometry (CSG)
  - boolean set operations on 3D objects
  - union, intersection, difference operation

combining 2 objects with a union operation, producing a single composite object
CSG: Different Set Operations
Every object is assembled from simple solids with set operations.

Data structure: binary tree.

Recursive evaluation.
Operations with CSG Trees

- **transformations**
  - multiplication of all transformation matrices with the matrix of this transformation

- **combinations**
  - generate a new node with the desired operator and link the operands as subtrees to it

\[ A \text{ op } B: \]

\[ \begin{array}{c}
\text{op} \\
A \\
\text{op} \\
B
\end{array} \]
Rendering of CSG Trees

- transform into B-Rep and use normal hidden surface algorithm
- or
- render directly with ray casting (or with ray tracing)
Ray-Casting Method (1)

- line-of-sight of each pixel is intersected with all surfaces
- take closest intersected surface
Ray-Casting Method (2)

- based on geometric optics, tracing paths of light rays
- backward tracing of light rays
- suitable for complex, curved surfaces
- special case of ray-tracing algorithms
- efficient ray-surface intersection techniques necessary
  - intersection point
  - normal vector
Ray-Casting Methods for CSG (1)

- visibility processing

Pixel plane

Ray
Ray-Casting Methods for CSG (2)

- determining surface limits

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Union</td>
<td>{A, D}</td>
</tr>
<tr>
<td>Intersection</td>
<td>{C, B}</td>
</tr>
<tr>
<td>Difference</td>
<td>{A, C}</td>
</tr>
</tbody>
</table>
Ray-Casting Methods for CSG (3)

- volume determination

\[ V_{ij} \approx A_{ij} \cdot \Delta z_{ij} \]

\[ V \approx \sum V_{ij} \]
Properties of CSG

- advantages
  - exact representation
  - low memory cost
  - combinations and transformations trivial

- disadvantages
  - rendering effort is high
Quadtrees

- hierarchical enumeration of objects
- in 2D: quadtree
  - hierarchical subdivision until a region is homogeneous

```
+--------+--------+
| 0 1    | 2 3    |
+--------+--------+
| 0       | 1       |
| 2       | 3       |
```

region of a 2-dim. space

data elements in the representative quadtree node
Quadtrees

- area with $2^n$ by $2^n$ pixels ⇒ quadtree with $n$ levels
- storage efficiency
Quadtrees

- area with $2^n$ by $2^n$ pixels $\Rightarrow$ quadtree with $n$ levels
- storage efficiency
quadtree representation for a region containing one foreground-color pixel on a solid background
suitable for representing (2D) images
Octree

= extension to 3D

regular space subdivision:

- simple (empty or uniform) $\Rightarrow$ leaf node
- complex (other cases) $\Rightarrow$ divide further
Octrees

- octree divides 3D cube into octants
- volume elements (voxels)
- set operations easy on octrees
- geometric transformations difficult on octrees

region of a 3-dim. space

data elements in the representative octree node
Octree Simple Example

G (WWWWSG (WWWWWWWS) SS)
Operations with Octrees

- **transformations**
  - very complicated except for a few special cases, e.g. rotation by 90°, mirroring at a subdivision plane, scalation by $2^n$

- **combinations**
  - very simple:
    - if A or B homogeneous $\Rightarrow$ simple rules
    - else combine recursively all 8 octants of A and B
Properties of Octrees

- **advantages**
  - combinations very simple
  - fast rendering
  - spatial search possible

- **disadvantages**
  - inexact representation
  - low image quality
  - restricted transformations
  - high memory cost
Other 3D Object Representations

- BSP trees
- fractal geometry methods
- shape grammars, procedural models
- particle systems
- physically based modeling
- visualization of data sets
- ...
Scene Graphs

- object-oriented data structure
  - directed acyclic graph
- describes logical and/or spatial relationship of scene objects
- describes groups of (groups of ...) objects
- no exact definition
- used in most graphics systems, e.g.
  - OpenSceneGraph
  - VRML
  - VRML
  - X3D ...
Scene Graph Example

sun

world

house

walls

doors

roof

windows

car

body

engine

wheels

transf. transf. transf. transf.

wheel

tree

© 123RF
Scene Graph Example
Scene Graph Example

Scene Graph:

rootNode

Node

Geometry
Scene Graph Example

Star
  Rotation
    Wobble!
      Planet 1
        Rotation
          Moon A
          Moon B
      Rotation
        Planet 2
          Rotation
            Moon C
            Moon D

© Garret Foster
Scene Graph Example
Scene Graph Example

- robot
  - body
    - bodyTransform
    - bronze
    - bodyCylinder
    - leftLeg
    - rightLeg
    - headTransform
    - silver
    - headSphere
  - leftTransform
  - rightTransform
  - leg
    - thigh
    - calf
    - foot
    - calfTransform
    - footTransform

© Francois Faure
Scene Graph Example