Einführung in Visual Computing

Werner Purgathofer

object coordinates

world coordinates

camera coordinates

normalized device coordinates

pixel coordinates

Graphics Pipeline
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

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

vertex stage ("vertex shader")

pixel stage ("fragment shader")
arbitrary graph implemented with pointers
Reminder: Graphs and Trees

binary tree 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”)

example:

machine part surface

represented by triangles
Polygon Surfaces (2)

- polygon tables (B-Rep lists)
  - geometric and attribute tables
  - vertex, edge, polygon tables
  - consistency, completeness checks
Polygon Surfaces: Data Structure

VERTEX TABLE

V₁:  x₁, y₁, z₁
V₂:  x₂, y₂, z₂
V₃:  x₃, y₃, z₃
V₄:  x₄, y₄, z₄
V₅:  x₅, y₅, z₅

EDGE TABLE

E₁:  V₁, V₂
E₂:  V₂, V₃
E₃:  V₃, V₁
E₄:  V₃, V₄
E₅:  V₄, V₅
E₆:  V₅, V₁

POLYGON TABLE

S₁:  E₁, E₂, E₃, E₆
S₂:  E₃, E₄, E₅, E₆
Lists for B-Reps

surface list

vertex list

edge list
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 \((A, B, C)\)

Example:

\[ x - 1 = 0 \]
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]
Front and Back Polygon Faces

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

\[ V_1, V_2, V_3 \text{ counterclockwise } \Rightarrow \]

normal vector \[ N = (V_2 - V_1) \times (V_3 - V_1) \]
Triangle Meshes

- most polygons are triangles
- *triangle mesh* = connected triangles

- *triangle-strip* = successive triangles
  (1 additional point per triangle)
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
CSG: Different Set Operations

\[ \cap, \cup, \neg \]
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

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

```
    op
   / \        
  /   \       
 A     B
```

Werner Purgathofer
Rendering of CSG Trees

- transform into B-Rep and use normal hidden surface algorithm
  or
- render directly with ray casting (or with ray tracing)
- 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
- visibility processing
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>
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

Region of a 2-dim. space

Data elements in the representative quadtree node

Quadrant 0
Quadrant 1
Quadrant 2
Quadrant 3
Quadtrees

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

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

- Quadtree with 2 by 2 pixels for a quadtree containing one foreground-color pixel on a solid background
- Storage efficiency

```
  0 1 2 3
  0 1 2 3
  0 1 2 3
  0 1 2 3
```

```
  0 1 2 3
  0 1 2 3
  0 1 2 3
  0 1 2 3
```

```
  0 1 2 3
  0 1 2 3
  0 1 2 3
  0 1 2 3
```
suitable for representing (2D) images
Octree

= extension to 3D

regular space subdivision:
  • simple (empty or uniform) ⇒ leaf node
  • complex (other cases) ⇒ 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
linearization: \( G(\text{WWWWSG(wwwwwwWWWWS)}SS) \)
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
  - X3D ...
Scene Graph Example

- sun
- world
- car
- tree
- house
- walls
- roof
- body
- engine
- wheels
- door
- windows
- transf.
- transf.
- transf.
- transf.
- wheel
Scene Graph Example

Scene Graph:

- rootNode
  - Node
    - Geometry
      - House
      - Character
      - People
      - Motorcycle

© jmonkeyengine.org
Scene Graph Example

Star

(Wobble!) Rotation

Planet 1

Rotation

Planet 2 Rotation

Moon C Moon D

Moon A Moon B

© Garret Foster
Scene Graph Example
Scene Graph Example

robot

body

bodyCylinder

leftLeg

rightLeg

head

headTransform

headSphere

bodyTransform

bronze

leftTransform

rightTransform

leg

thigh

calf(Transform)

calf

foot(Transform)

foot
Scene Graph Example

VirtualUniverse Object

Locale Object

BranchGroup Nodes

TransformGroup Nodes

Shape3D Node

ViewPlatform Object

View

Other Objects

Appearance

Geometry

Behavior Node

User Code and Data

B

T

S

BG

BG

© S. Teichmann
Einführung in Visual Computing

186.822

3D Object Representations

The End