**Realtime 3D Computer Graphics**

Virtual Reality

**Graphics**

---

**Computer graphics**

- 3D-Computer graphics (3D-CG) currently used for:
  - Simulators, VR, Games (real-time)
  - Design (CAD)
  - Entertainment (Movies), Art
  - Education

---

**Real-time 3D-CG**

Goals:
- Generating 3D worlds that visualize application domain specific data in >3 dimensions (maybe using 3D metaphors).
- Mapping 3D world to 2D pictures to simulate user-eye-specific views in interactive rates (stereoscopy, immersion).
- Implement user interaction with the virtual world.
- Operate in real-time, minimize lag, maximize simulation rate.

Methods:
- Computer graphics methods differ in:
  - primitives used and processed (e.g., vertex-, volume-, pixel-based, etc.).
  - lighting model and calculation used (local or global models).
  - direction of "lighting application", shading (scanline shader, raytracer, etc.).
  - Trigonometry and linear algebra.

- By 2005, the majority of VR-systems produces 2D images based on:
  - surface (vertex) models and (image) textures using:
  - a local Phong lighting model processed by a pipeline architecture (changing significantly just now from fixed to programmable pipeline).

---

**The 3D-pipeline**

The 3D-pipeline consists of several stages: application, geometry, model and view transformation, lighting, projection, clipping, screen mapping, rasterizer, visibility (z-buffer), shading, texturing.

---

**Spaces**

1. **Object Space**: Local objects’ frames of reference (aka Modeling Space).
2. **World Space**: Where the scene and viewing specification is made.
3. **Eyespace (Normalized Viewing Space)**: Eye point (COP) is at the origin looking down the Z axis.
4. **3D Image Space**: A 3D perspected space. Dimensions: -1:1 in x & y, 0:1 in Z. Where image space hidden surface algorithms work.
5. **Screen Space (2D)**: Coordinates 0:width, 0:height

---

**Example Image Space Viewing Algorithm Steps**

1. Create World Space Scene:
   - Set up view parameters
2. Calculate Backface, illumination, etc.
3. Calculate and Perform World to Eyespace Transformation
4. Calculate and Perform Perspective Transformation
5. Clip Objects
6. Perspective Divide
   - Can Perform Backface removal here
7. Map to 3D Screen Space:
   - Viewport Transformation
8. Perform Hidden Surface Removal
9. Draw to Screen (Scan Convert):
   - Shading, texture
Steps in the 3D pipeline

1. Application/Scene
   - Scene/Geometry database traversal
   - Movement of objects, and aiming and movement of view camera
   - Animated movement of object models
   - Description of the contents of the 3D world
   - Object Visibility Check including possible Oclusion Culling
   - Select Level of Detail (LOD)

2. Geometry
   - Transformations (rotation, translation, scaling)
   - Transform from Model Space to World Space (Direct3D)
   - Transform from World Space to View Space
   - View Projection
   - Rectangular Accept/Reject Culling
   - Back-Face Culling (can also be done in Screen Space), Lighting, Perspective Divide
   - Transform to Clip Space
   - Culling
   - Transform to Screen Space

3. Triangle Setup
   - Back-face Culling (or can be done in view space before lighting)
   - Slope/Delta Calculations
   - Scan-Line Conversion

4. Rendering / Rasterization
   - Shading
   - Texturing
   - Fog
   - Alpha Transparency Tests
   - Anti-aliasing (optional)
   - Display

5. Draw to Screen
   - Object Model / Face
   - Antialiasing (optional)
   - Depth Buffering
   - Alpha Translucency Tests
   - Shading
   - Scan-Line Conversion
   - Slope/Delta Calculations
   - Back-face Culling (or can be done in view space before lighting)
   - Occlusion Culling
6. Display
   - Rendered display
   - Scene/Geometry database traversal
   - Animated movement of object models
   - Designing the interaction.

VR-development tools

- Several software packages and APIs support the real-time 3D-CG relevant processing stages.
- Complex tools support additional simulation tasks (auditory, tactile feedback, …).
- For mature simulations, the overall task is separated into:
  - modeling the world
  - designing the interaction.
- Direct rendering, gif and simulation packages
  - OpenGL, Direct3D, SKS (IDV), VCL, X3D, VRML, HAVOC, …
- Scene graph based tools and APIs
  - VRML, X3D, OpenGL Performer, OpenGL, Open Inventor, OpenSG, PHIGS, …
- VR modeling toolkits
  - AVANGO, World toolkit, Massive1.3, Dive, Lighting, game engines, …

Scene Graphs

A Scene graph
- Is a data structure for 3D-CG:
  - DAG (Directed Acyclic Graph)
  - Tree (no multiple instances)
  - Organizes and controls the rendering of its constituent objects.
- Consists of:
  - Leaf nodes (geometry, lights, views)
  - Intermediate nodes (grouping, transformation, sensors, engines, materials).
  - A root node (base frame of reference).

Scene Graph Language with Field Routing: VRML/X3D

A application graph
- Can be either DAG or DCG (Directed Cyclic Graph).
- Organizes and controls data flow.
- Constitutes a data flow network:
  - Nodes (computational variables) are represented as fields.
  - Nodes are represented as field containers.
  - Fields of equal types can be interconnected by routes.
  - Changing field-values generates events processed:
    - Once per frame.
    - Once per frame.
  - Can support multiple field instances.

Application Graphs

More VRML later in this course!