Rendering Engine Part A

hannes@ru.is
Rendering a 3D Scene

- Involves these basic steps
  - Describe Virtual Scene
  - Placing Virtual Camera (virtual light sensors)
  - Define Light Sources
  - Define Visual Properties of Surfaces
  - Solving the Rendering /Shading Equation
Goal of Rendering

• Photorealism vs. real-time performance

• A real-time engine has at most 33.3 ms to generate each image (for 30 FPS)
Describing a Scene

- Scenes are composed of Objects
  - Opaque
  - Transparent
  - Translucent

- Opaque objects can be rendered considering only the surface

- Real-time engines deal with opaque surfaces and approximate transparcy with alpha
Representing Surfaces

- Parametric Surface Equations
- Patches (Bézier, NURBS)
- Subdivision Surfaces
- Triangle Meshes
Triangle Meshes

• Triangles as a piece-wise linear approximation to a surface.

• Triangles the polygon of choice for real-time rendering because:
  – Simplest polygon
  – Always planar
  – Remain triangles under most transformations
  – Hardware for triangle rasterization
Tessellation

- The process of dividing a surface up into a collection of discrete polygons (triangles or quads)
- Triangulation is tessellation of a surface into triangles
- Ideally tessellate as much as needed for camera
Tessellation and LOD

- Approximate right density of tessellation with Level of Detail (LOD) versions of objects
Tessellation and Detail

• Dynamic Tessellation
  – Grid patterns e.g. for terrain
  – Fewer and fewer grid points used for tessellation further away from camera

• Progressive Meshes
  – Single high-resolution mesh
  – Automatically detessellated further away (collapsing edges)
Constructing Triangle Meshes

• Pick a Winding Order
  – Counterclockwise typical (but arbitrarily picked)
  – Determines which side is front and which is back

• Triangle Lists
  – List vertices in groups of three
Constructing Triangle Meshes

• Indexed Triangle Lists
  – Duplicated vertices waste memory and GPU cycles
  – Store vertices once in Vertex Buffer and use lightweight indices in an Index Buffer to define the triples of triangle vertices
Constructing Triangle Meshes

- Triangle Strips and Fans eliminate need for an index buffer while still reducing vertex duplication
- Predefined order of vertices
- Combined in a certain way to form triangles
Vertex Cache Optimization

• As vertices are processed by vertex shaders, they are cached for reuse.
• Strips and fans improve cache coherency.
• A vertex cache optimizer can manipulate other triangle meshes offline to optimize vertex reuse.
Model Space

• Positions of triangle vertices given relative to coordinate system called model space (also local or object space)

• The orientation of this coordinate system is arbitrary but typically aligned with a front, left (right), up direction
Instancing in World Space

- Mesh instances are positioned and oriented in a scene with respect to a world space coordinate system.
- The mesh’s vertices are converted from model space to world space using a model-to-world matrix.
- Special care has to be taken when converting normals from model to world space. Need to use inverse transpose matrix if scale and shear.
Visual Properties of Surfaces

• How does light interact with the surface?
  – Surface Normal
  – Diffuse Color
  – Shininess / Reflectivity
  – Roughness or Texture
  – Degree of Opacity or Transparency
  – Index of Refraction
Light and Color

- Light is electromagnetic radiation
- Color determined by intensity and wavelength
- Visible light in 740nm to 380nm range
- Beams can contain one or more wavelengths
Light-Object Interactions

• Many complex interactions with matter
• Governed by
  – Medium
  – Interface between media
• Surface is an interface between media
Light-Object Interaction

• What can light do?
  – Be absorbed
  – Be reflected
  – Be transmitted through an object and be refracted in the process
  – Be diffracted when passing through narrow openings (not usually modeled)
Light-Object Interaction

- Certain wavelengths absorbed by surface, others reflected
- Those not absorbed give off the perceived color
- Reflected can be diffuse (scattered equally in all directions) or specular (reflect directly or spread in narrow cone)
- Transmitted through volume, light can be scattered, partially absorbed or refracted
Color Spaces and Color Models

• Color model three dimensional because of three types of color sensors (cones) in our eyes, which are sensitive to different wavelengths of light

• Most commonly RGB
  – RGB888 uses 8 bits per channel
  – RGBA adds alpha channel
Vertex Attributes

• The simplest way to describe the visual properties of a surface is to specify them at discrete points on the surface.
• Vertices are such points.
• Visual properties in vertices are called vertex attributes.
Vertex Attributes

- Position vector
- Vertex normal
- Vertex tangent and bitangent (tangent space)
- Diffuse color
- Specular color
- Texture coordinates
- Skinning weights

Stored in different combinations in different vertex formats
Vertex Formats

• Vertex attributes are stored within data structures called vertex format
• Not all combinations of vertex attributes can be handled by all hardware, therefore these have to be compatible
• However, modern GPUs are capable of extracting the subset of attributes that they actually need
Attribute Interpolation

• Attributes at a triangle’s vertices are discretized approximations to the visual properties of the surface as a whole.

• When rendering the triangle, the interior as „seen“ through each pixel on the screen matters

• Simple linear interpolation of per-vertex attributes to determine per-pixel attributes is possible
Attribute Interpolation

- Gouraud Shading is an example of per-vertex color attributes interpolated to determine per-pixel color
Vertex Normals and Smoothing

• Simplest way to light a mesh is to calculate color per-vertex, using the vertex normal and direction to light source

• Normals can be made to point in such a way that the interpolation across the entire surface is smooth, resulting in rounded corners
Per-vertex Lighting Error

• Linear-interpolation of per-vertex data can cause visual errors

• For example when rendering specular highlights (the highlight calculated at a vertex gets spread out towards the other vertices)
Textures

• One way to provide per-pixel information is through texture maps
• Textures can contain color information, applied to the interiors of triangles of a mesh
• Textures can also contain other information for per-pixel calculation, e.g. normals
• Individual picture elements of textures are called texels (different from on-screen pixels)
Textures

• Typically texture sizes have to be powers of two and possibly even square
  – 512x512 or 1024x1024 are common

• Types of textures include
  – Diffuse maps (diffuse color)
  – Normal maps
  – Gloss maps
  – Environment maps
  – ...and just about anything we need for calculation
Texture Coordinates

• 2D textures need to be projected onto a 3D mesh so that values can be looked up for each interior triangle pixel
• Each texture exists in texture space (uv space), where coordinates range from (0,0) to (1,1)
• The projection is made possible by storing texture (u,v) coordinates with each vertex in the mesh, mapping each triangle onto a 2D triangle in the texture
Texture Addressing Modes

• Texture coordinates are permitted to be outside the [0, 1] range
• What happens outside of the range is determined by the texture addressing mode
  – Wrap
  – Mirror
  – Clamp
  – Border color
Texture Formats

- Texture bitmaps typically stored as
  - Targa (.tga)
  - Portable Network Graphics (.png)
  - Windows Bitmap (.bmp)
  - Tagged Image Format (.tif)

- Compressed textures also supported, e.g. DirectX supports DXT texture compression
Texel Density

- Texel Density is the ratio of texels to pixels
  - Imagine viewing the texels through a single pixel on the screen
- When low, we see the edges of the texel
- When high, image may band or swim, and memory is wasted
- Texel density of 1 would be ideal
Mipmapping

- We can approximate texel density of 1 with Mipmapping
- Mipmaps of a texture are lower-resolution versions, each half the height of previous one
- Graphics hardware picks the mip level that produces the best texel density
World Space Texel Density

• We can also measure the ratio of texels per world unit of measurement
• E.g. texels per meter or cm
• A low ratio may result in washed out environments
• Studios provide guidelines for this per game title
Texture Filtering

• When rendering a pixel of a textured triangle, the hardware samples the texture map by seeing where pixel falls in texture space
• Not usually a one-to-one mapping between texels and pixels
• May need to sample more than one texel and blend together \(\rightarrow\) texture filtering
Texture Filtering

- Most graphics cards support
  - Nearest neighbor: Just pick the closest texel and mipmap
  - Bilinear: Four surrounding texels blended in the nearest mipmap
  - Trilinear: Do bilinear filtering on two nearest mipmaps, then interpolate (less abrupt mip levels)
  - Anisotropic: Samples texels within trapezoidal regions based on view angle
Materials

- A material is a complete description of the visual properties of a mesh
  - Textures mapped onto surface
  - Shader programs used
  - Shader inputs
  - Etc.

- Vertex attributes typically not part of this, but mesh-material pairs often called render packets
Light Transport Models

- Mathematical models of light-surface and light-volume interactions
- Divided up into Local illumination models and Global illumination models
- Direct lighting is a simple local illumination model (phong lighting)
- Indirect lighting is an example of a global illumination model (ray tracing, radiosity)
The Phong Lighting Model

- Models light reflected from a surface as a sum of
  - Ambient term
  - Diffuse term
  - Specular term
Light Sources

• Approximations of real-world light sources
  – Static lighting
  – Ambient lights
  – Directional lights
  – Point lights
  – Spot lights
  – Area lights
  – Emissive lights