INTRODUCTION TO
COMPUTER GRAPHICS AND
INTERACTION

RASTERISED RENDERING

Christopher Peters
HPCViz, KTH Royal Institute of Technology, Sweden
chpeters@kth.se
http://kth.academia.edu/ChristopherEdwardPeters

Based on DGI12 notes by Carl Henrik Ek
The Rendering Equation

\[ L_o(x, \omega_o, \lambda, t) = L_e(x, \omega_o, \lambda, t) + \int_{\Omega} f_r(x, \omega_i, \omega_o, \lambda, t) L_i(x, \omega_i, \lambda, t) (\omega_i \cdot n) \, d\omega_i \]

Describes:
Total amount of light emitted from a point \( x \) along a specific viewing direction

Given:
- Incoming light function
- BRDF
- Hemisphere containing all \( \omega_i \)

Basis:
- Law of conservation of energy
Global Illumination

Ray tracing:
– Good for specular
– Bad for diffuse

Radiosity:
– Good for diffuse
– Bad for specular

Hybrid techniques

Christopher Peters
chpeters@kth.se
Caustics

- Curved regions of bright reflected or refracted light
Sub-surface scattering

- Light bouncing around inside material before exiting
Ray tracing

Pixel order rendering technique
- Trace at least one ray through each image pixel
- Maintains primitives in geometric scene model
- Queries this for each ray
- Determine which primitive is visible for each pixel

Geometry queries can have high cost
Rasterisation

Scanline: object order based

Fragments
  - Data for single pixel
  - Frame buffer

Handle occlusion using depth buffer
  - Later details (more specifically, fragments) overwrite earlier ones if closer to camera

Shade based on vertices and interpolate
  - See lighting and shading lecture
Rasterisation

Process of converting geometry into a raster image (series of pixels)
- 3D scene converted into 2D image
- Polygons
  - ...composed of triangles
  - ...composed of vertices
Rasterisation

Rasteriser takes stream of vertices
Project them onto the 2D surface of the screen
Fill in the interior of the 2D triangles

Core concepts:
- Geometry and transformations
- Projection
- Clipping
- Scanline Conversion
Geometry Transformations

Matrix multiplication
Translation, scaling, rotation, projection
Familiar?

---

Christopher Peters  
DH2323 Rasterised Rendering I  
chpeters@kth.se
Transformation Stack

*Stack* of transforms (i.e. matrices)
- Push and pop
Position stream of input vertices
Incoming vertices transformed according to the transformation stack
Remember: local coordinate marker idea
Projection

Remove depth
  – Convert 3D geometry to flat 2D representation
  – Do so for each vertex of each polygon

Orthographic projection
  – Simply remove z coordinate
  – Viewing volume is a cube

Perspective projection
  – Single point of projection (focal point)
  – Viewing volume is a pyramid
Parallel Projections

axonometric

orthographic

oblique
Orthographic Projection

From OpenGL Programming Guide
Perspective Projection

From OpenGL Programming Guide

Christopher Peters
DH2323 Rasterised Rendering I
chpeters@kth.se
Example

Perspective projection (P)  Orthographic projection (O)

Nicolas P. Rougier, ERSF Code camp
Projection
Camera Specification

Parameters

Centre of projection (COP)
Field of view (FOV)
Projection direction
Up direction
Outcomes

Large FOV

Small FOV
Clipping

Projected locations may be outside the viewing window
Truncate triangles to fit them inside the viewing area
– e.g. Sutherland-Hodgeman algorithm

Real-time Rendering, Akenine-Moller, Haines and Hoffman
Scan Conversion

Fill interior of triangles in image plane
Use *scanline fill algorithm* to fill polygons
Framebuffer
Line Drawing

A line usually defined as infinitely thin

How to display using pixels?

Fixed and finite area

Choose pixels that best represent the line

Different algorithms, providing different results:

- mid-point
- neighbourhood
- weighted area
Bresenham's line algorithm

dx = x_end - x_start
dy = y_end - y_start
d = 2 \times dy - dx
x = x_start
y = y_start
while x < x_end
  if d <= 0 then
    d = d + (2 \times dy)
    x = x + 1
  else
    d = d + 2 \times (dy - dx)
    x = x + 1
    y = y + 1
  endif
  SetPixel(x, y)
endwhile
Polygon Filling

Fill surface
Triangles
Interpolation
Compute edges
Backface Culling

Objects within the view-frustum may have polygons pointing away from the viewer
Not visible
*Back-faces*

The process is known as *back-face culling*
Backface Culling

To eliminate back-faces:

For each polygon in the scene {
    Take its normal vector
    Take the view direction vector
    Use the dot product to find the angle between normal and view direction
    If the angle is LESS than 90 degrees, then the polygon is culled
}
Visible Surface Determination

Painter's algorithm
Sort polygons relative to the viewpoint
Render those polygons that are nearer the viewpoint after those polygons that are further away from the viewpoint

Problems?
Visible Surface Determination

Painter's algorithm
Sort polygons relative to the viewpoint
Render those polygons that are nearer the viewpoint \textit{after} those polygons that are further away from the viewpoint

Problems?
Depth Buffer

Image-space visibility algorithm
Buffer is 2D array, one element per pixel
Compute depth of each generated pixel
Overwrite depth buffer value if new value is nearer to camera than previous
Non-linear, Z-fighting

A simple three-dimensional scene
Z-buffer representation
Note: BSP Trees

Used by Quake, Quake 2, etc
World-space visibility algorithm
Visibility calculations on a *per polygon* basis
Split polygons
Compare with Z-buffer algorithm
Texturing

Bitmap (2D image) applied to a triangle
Each triangle associated with an image
Each vertex associated with a texture coordinate \((u,v)\)
For each rendered pixel:

- Lookup corresponding texture element (texel)
- Interpolation
Labs

Animation track, lab 3 posted this Friday
Labs

Animation track, lab 3 posted this Friday

You should be finishing Lab 2 soon
Any major problems?

Suggested steps:

Fork (continue working on problem)
If (help session happening soon)
    Goto help session and ask (question)
Else
    Post (specific details, KTH Social)
    Wait until (reply or help session)

Try not to crash in meanwhile…
(Physical) Labs Session

• Friday 29th April, 12.30-2p.m, Visualisation (VIC) Studio

See previous KTH Social posts for directions

• Purposes:
  – Ask questions/get help if in process of completing a lab task
  – Obtain feedback if you have work-in-progress
    • Code / report
    • Documentation
Submission dates

Submission date for all labs:
On or before **May 20th**
Through Bilda *(will open by end of this week – look out for the notification email)*

Submission date for all projects**:
On or before **May 31st**
Through Bilda *(will open by end of this week)*

**Also upload your project specifications to Bilda**