CS7055 – Real-time Rendering

http://www.scss.tcd.ie/Michael.Manzke/index.php/mm-teaching/msc-taught/cs7055
CS7055 Real-time Rendering
http://www.scss.tcd.ie/Michael.Manzke/index.php/mm-teaching/msc-taught/cs7055

- Lecturer Details:
  - Dr. Michael Manzke
  - Location: Lloyd 0.14
  - Michael.Manzke@scss.tcd.ie

- Format:
  - 2 Lectures + 1 Lab per week
  - Assessed by Coursework
Interactive Photon Mapping

Some examples of advanced Real-time Rendering research at TCD.

Interactive Global illumination using CUDA: Raytraced scene runs at 70+fps and Photon Mapping at 5fps

Images by Bartosz Fabianowski, GV2
Interactive Photon Mapping

Some examples of advanced Real-time Rendering research at TCD.

Interactive Global illumination using CUDA: Raytraced scene runs at 70+fps and Photon Mapping at 5fps

Sponza Atrium

- 256k photons emitted
- two bounces of indirect lighting

Images by Bartosz Fabianowski, GV2

Video: http://www.scss.tcd.ie/~fabianob/egsr2009.avi
Non-photorealistic Rendering (NPR)

Real-time rendering for aesthetics instead of realism. Image space abstraction and 3D edge rendering are used to create painterly effects and shading styles.

Images by Niall Redmond, GV2. Models of Trinity courtesy of GV2 Metropolis project.
Illustrative Visualisation

Rendering for clarity on the GPU. GPU Image Shaders used for different types of edge detection (above). Pixel and vertex shaders are used in combination with 3D Line Drawing and cool-to-warm Gooch shading for volume rendering of the Visible Human dataset.

Images by Niall Redmond, and Andrew Corcoran, GV2
## Tentative Lecture Plan

<table>
<thead>
<tr>
<th></th>
<th>Lecture Topic</th>
<th>Lab Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>LAB: BASIC 3D GRAPHICS [LECTURE &amp; LAB]</td>
</tr>
<tr>
<td>2</td>
<td>SHADER PROGRAMMING &amp; GLSL</td>
<td>LAB: BASIC ILLUMINATION MODELS [IET LAB]</td>
</tr>
<tr>
<td>3</td>
<td>ADVANCED ILLUMINATION MODELS</td>
<td>LAB: ADVANCED ILLUMINATION [IET LAB]</td>
</tr>
<tr>
<td>4</td>
<td>TRANSMITTANCE MODELS</td>
<td>LAB: TRANSMITTANCE EFFECTS [IET LAB]</td>
</tr>
<tr>
<td>5</td>
<td>SURFACE MAPPING</td>
<td>LAB: NORMAL MAPPING [IET LAB]</td>
</tr>
<tr>
<td>6</td>
<td>MULTI-PASS RENDERING</td>
<td>LAB: (NORMAL MAPPING DEMOS) [IET LAB]</td>
</tr>
<tr>
<td></td>
<td><strong>READING WEEK</strong> [23 FEB – 27 FEB 2015]</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>SHADOWS</td>
<td>GAME TECH TALKS</td>
</tr>
<tr>
<td>9</td>
<td>NON-PHOTOREALISTIC RENDERING</td>
<td>GAME TECH TALKS</td>
</tr>
<tr>
<td>10</td>
<td>VOLUME RENDERING</td>
<td>GAME TECH TALKS</td>
</tr>
<tr>
<td>11</td>
<td>GLOBAL ILLUMINATION</td>
<td>GAME TECH TALKS</td>
</tr>
<tr>
<td>12</td>
<td>LEVEL OF DETAIL</td>
<td></td>
</tr>
</tbody>
</table>
Assessment

- Assessed by Coursework Only
- Passing grade is 50%*
- Breakdown of assessment
  - ~35% Weekly lab assignments
  - ~15% Game Technology Presentation
  - ~50% Final Project

Reminder: an aggregate of 50% is required across all modules in order to proceed to MSc
Recommended Reading
N.B. The course is not directly based on any one of these texts but these are relevant

Various papers and talks from Siggraph, GDC, ACM, Eurographics, i3D, NPAR. Should be available online.
Game Tech Talks
15% of Module Assessment

1. **Choose** a video game or commercial interactive application (of any era) which excelled in its use of 3D graphics

2. **Research** the game. Find related screenshots, presentations, technical articles

3. **Prepare a 15-minute presentation** (With Slides)
   a. Introduction to the game: *history, details of platforms, developer etc.*
   b. Reason for choosing the game: *what aspect of 3d did it excel in*
   c. Technical overview: *overview of the tech behind the 3d features*
   d. Examples of the technology: *movies, pictures*
   e. Conclusion: *impact, did future games improve upon it, could you improve on it*

4. **Present** your work. Strict 15 mins for presentation & 5 mins for Q&A.
The Rendering Pipeline
The processes involved in representing, storing and manipulating graphical information and displaying this on visual output devices.
The Rendering Equation

Representation of light transport in the scene, describing the amount of light going from any point \( x \) to another point \( x' \):

\[
I(x, x') = g(x, x')[\varepsilon(x, x') + \int_{s} \rho(x, x', x'')I(x', x'')dx'']
\]

[Kajiya 1986]

- **I(x, x')**: intensity of light passing from \( x \) to \( x' \)
- **(two point transport)**
- **g(x, x')**: 
  \[
  g(x, x') = \begin{cases} 
  0 & \text{if } x \text{ and } x' \text{ are not mutually visible} \\
  1/r^2 & \text{where } r = \frac{\|xx'\|}{1} 
  \end{cases}
  \]
  Attenuation (distance from \( x \) to \( x' \))
- **(geometry factor)**
- **\( e(x, x') \)**: intensity of light emitted by \( x \) and passing to \( x' \)
- **\( r(x, x', x'') \)**: bi-directional reflectance scaling factor for light passing from \( x'' \) to \( x \) by reflecting off \( x' \)
- **\( S \)**: all surfaces in the scene
Local vs Global Illumination

**Local**
Illumination depends on local object & light sources only

**Global**
Illumination at a point can depend on any other point in the scene
Local Illumination

Ambient Light + Diffuse reflection + Specular Highlight =

Texture + Lighting =

Texture Lighting
Global Illumination
Gran Turismo for PlayStation 1997

Gran Turismo 2 for PS/PSOne 1999

Gran Turismo 3, PlayStation 2 2002

Gran Turismo 4, PlayStation 2 2004
Interactive Graphics Pipeline

The graphics pipeline [done for every frame] converts a scene description into pixels. Functionally divided into 3 high level stages:

- **Application Stage**
  - Geometry setup
  - Model and View Transform
  - Lighting
  - Projection
  - Clipping
  - Screen Mapping

- **Geometry Stage**
  - Triangle Setup
  - Triangle Traversal
  - Pixel Shading
  - Merging

- **Rasterizer Stage**
Scene Description

Geometry (Typically Polygonal Models)
Surface Information (textures, BRDF*, etc)

Lighting
Motion Information
Camera

*Bidirectional Reflectance Distribution Function (BRDF)
Application Stage

- Generally run on the CPU – under full control of the developer
- Key Stage: Geometry Setup
- Perform calculations required to create full geometric description of current scene state:
  - Evaluate all animations
    - Collect physics update information
    - Update character poses
    - Update procedural animations
  - Generate any procedural geometry
    - Skin Characters
    - Procedural content generation
    - Tessellate higher order surfaces
  - Build geometry data
    - Vertex Buffers, VBOs etc
- Returns rendering primitives.
Transform

- Most geometry is defined in local space
  - E.g. Centred on the origin
  - Need to position relative to the world

- Scene Graph
  - A hierarchy of relative transformations
  - Approach used by many game engines
    - Gamebryo, OSG etc.
  - Alternative is a flat scene database

*: may be subject to culling, view dependent
Viewing

- Creating the camera view
  - Transform all geometry into the camera space (eye space)
  - Camera at origin looking down Z
Lighting (Vertex Shading)

- Generally performed in world-space
  - Light positions defined relative to the world
- Lighting at this stage implies per-vertex lighting
  - Light intensities calculated at vertices of geometry only
  - Later interpolated to compute lighting at interior points
- Lighting results from this stage are stored with the vertices
Projection & Clipping

- Project scene onto view volume  \( x_c = Px_e \)
  - Orthographic / perspective etc.
  - Objects are converted to normalized device co-ords. Frustum mapped to unit cube

- Primitives partially inside the unit cube need clipping
Screen Mapping

- Clipped primitives inside the view volume are transformed to 2D Screen coordinates
**Rasterize**

- All geometric primitives (triangles) must be converted to pixels – or rasterized
  - Colour determined by lighting / textures – usually interpolated
  - Depth determined from previous transformation pipeline
  - At least 2 buffers maintained: C (color) and Z (depth)

- Many different raster operations (ROP)
  - AA, HDR, Gamma etc
Triangle Setup
- differentials/data for the triangles surface are computed
- used for scan conversion and interpolation

Triangle Traversal
- scan conversion
- fragments generated for each pixel covered by a primitive
- each fragment properties (e.g., Depth) are generated using data interpolated among verts

Pixel Shading
- per-pixel operations performed here by programmable GPU cores e.g., Texturing
- extremely configurable

Merging
- combine fragments into the colour buffer.
- Includes visibility ops z-buffer, alpha channel, stencil.
- Typically not programmable but highly configurable.
Real-time Graphics

- Perceptual Optimisation
  - Image fidelity: manage complexity of illumination/shading, limit dynamics
  - Dimension reduction
  - Discretization
  - Level of detail techniques: manage geometric detail, adaptive graphics
  - Time-critical & progressive techniques
  - Post-process and abstraction: hide bad details

- Performance Optimisation
  - Graphics hardware
  - Parallel rendering
  - Algorithmic optimisation
    - exploit coherency + re-use
  - Culling
  - Pre-computation
### Tentative Lecture Plan

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>LAB: BASIC 3D GRAPHICS :: 3</td>
</tr>
<tr>
<td>2</td>
<td>SHADER PROGRAMMING &amp; GLSL</td>
<td>LAB: BASIC ILLUMINATION MODELS :: 8</td>
</tr>
<tr>
<td>3</td>
<td>ADVANCED ILLUMINATION MODELS</td>
<td>LAB: ADVANCED ILLUMINATION :: 8</td>
</tr>
<tr>
<td>4</td>
<td>TRANSMITTANCE MODELS</td>
<td>LAB: TRANSMITTANCE EFFECTS :: 8</td>
</tr>
<tr>
<td>5</td>
<td>SURFACE MAPPING</td>
<td>LAB: NORMAL MAPPING:: 8</td>
</tr>
<tr>
<td>6</td>
<td>MULTI-PASS RENDERING</td>
<td>LAB: (NORMAL MAPPING DEMOS) :: 8</td>
</tr>
</tbody>
</table>

**READING WEEK [25 FEB – 01 MAR 2013]**

<table>
<thead>
<tr>
<th>8</th>
<th>SHADOWS</th>
<th>GAME TECH TALKS :: 14</th>
<th>SHADOW RENDERING</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>NON-PHOTOREALISTIC RENDERING</td>
<td>GAME TECH TALKS</td>
<td>REAL-TIME NPR</td>
</tr>
<tr>
<td>10</td>
<td>VOLUME RENDERING</td>
<td>GAME TECH TALKS</td>
<td>OPENGL ES</td>
</tr>
<tr>
<td>11</td>
<td>GLOBAL ILLUMINATION</td>
<td>GAME TECH TALKS</td>
<td>OPENCL</td>
</tr>
<tr>
<td>12</td>
<td>LEVEL OF DETAIL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>