COURSE CONTENT:

The course focuses on clever methods of organizing and accessing very large amounts of data to enable us to solve very large problems in a space efficient and time efficient manner. We will follow the τεχνη approach, used in CPSC 101 and 102, and focus on a large, semester-long problem from the visual domain. You will be given some elementary, ray-tracing code written in C. (You can use your own, from CPSC 102, if you prefer.) The goal will be to extend this code to a C++ ray-tracer that can ray-trace a scene such as that shown in Figure 1, which contains more than 250 million triangles. A straightforward extension of the supplied, elementary C code to ray-trace this scene would require months of execution time. You will be given 30 seconds.

MEETINGS:

- Monday, Wednesday, Friday, 10:10am - 11:00am, 119 McAdams
- Monday, 6:00pm - 7:50pm, 110E McAdams (Lab Section 1)
- Tuesday, 7:00pm - 8:50pm, 110E McAdams (Lab Section 2)
- Monday, 8:00pm - 9:50pm, 110E McAdams (Lab Section 3)
- Tuesday, 3:00pm - 4:50pm, 110E McAdams (Lab Section 4)

TEXT:


OFFICE HOURS:

- 3:30pm – 5:30pm MWF, 308 McAdams Hall (but come any time!)
ATTENDANCE:
1. Attendance is not required. There will be no roll call.
2. Substantial project information will be provided in class lectures.
3. No individual lectures will be given.

GRADES:
- 2 major programming projects (group projects), each counts 20%
- Exams:
  - Midterm counts 25%
  - Final counts 25%
- Lab participation counts 10%

The grading scale is:
A: 85 – 100
B: 70 – 84
C: 55 – 69
D: 40 – 54
F: 39 and below

Course Outline

<table>
<thead>
<tr>
<th>Class Periods</th>
<th>I. Orientation to 3D Geometry.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A. Points, Vectors.</td>
</tr>
<tr>
<td></td>
<td>B. Dot Products, Cross Products, and their Geometric Interpretations.</td>
</tr>
<tr>
<td></td>
<td>C. Rotations and Translations, and their Algebraic Formulations.</td>
</tr>
<tr>
<td></td>
<td>D. Ray/Surface Intersections - Sphere Example.</td>
</tr>
<tr>
<td></td>
<td>II. Elementary C++ Classes and Operator Overloading.</td>
</tr>
<tr>
<td></td>
<td>A. A Point Class and Arithmetic Thereon.</td>
</tr>
<tr>
<td></td>
<td>B. An Object Base Class, Virtual Methods, Protected Storage.</td>
</tr>
<tr>
<td></td>
<td>C. Derived Classes of Sphere and Triangle.</td>
</tr>
<tr>
<td></td>
<td>D. The Möller-Trombore Algorithm for Ray/Triangle Intersection.</td>
</tr>
<tr>
<td></td>
<td>E. A Second-Level Derived Class: Texture Triangle.</td>
</tr>
<tr>
<td></td>
<td>F. Handling Images in P7 (RGBA ppm) format.</td>
</tr>
<tr>
<td></td>
<td>III. Code Review of Distributed C Ray-Tracer</td>
</tr>
<tr>
<td></td>
<td>IV. Lists and Sorting</td>
</tr>
<tr>
<td></td>
<td>A. Queues, Stacks, Class Templates.</td>
</tr>
<tr>
<td></td>
<td>B. Selection Sort and Its Analysis - $O()$ Notation.</td>
</tr>
</tbody>
</table>


C. Heapsort.
D. A Linear-Time Algorithm to Find the Median.
E. Quicksort.

V. Midterm Exam 1

VI. Parsing Large Data Files 3
A. File Operations.
B. String Operators `strtok()`, `strcmp()`, `sscanf()`, `fgets()`.
C. The `.obj` File Format.

VII. Trees 10
A. Ordered Binary Search Trees and $O(\log(n))$ performance.
B. KD Trees for 3D Collections of Points.
C. Recursive Search of KD Trees.
D. Non-recursive, Stack-based Search of KD Trees.
E. KD Trees of Triangles and Bounding Structures.
F. AVL Trees and Their Performance.
G. Red/Black Trees and Their Performance.

VIII. Hashing 3
A. Direct Chaining.
B. Open Addressing.
C. Derivation of Average Performance.

IX. Dynamic Programming 3
A. Shortest Path in Layered Networks.
B. Optimal Disk Scheduling.