CMSC 335
COMPUTER GRAPHICS

LECTURE 14

• TRIANGLE MESHES
• BOUNDING VOLUME HIERARCHIES
• SPACE PARTITIONING
• TILING
GEOMETRIC DATA STRUCTURES

• There are many data structures useful in computer graphics
  • Efficient storage of meshes
  • Efficient storage of scenes
  • Efficient structures for collision detection
  • Efficient structures for rendering order
  • Etc!
TRIANGLE MESHES
TRIANGLE MESHES

- We have seen only a few ways to store triangle meshes
  - For example – interleaved vertex data in which every consecutive three vertices defines a triangle
- However, what if we needed triangle adjacency information or edge information?
  - Very relevant in applications that reason about or modify geometry (Maya/Blender)
MESH TOPOLOGY

• Typically in geometric algorithms, a surface is required to be **manifold**, loosely it should be "watertight"
  • Every edge needs to be shared by exactly two triangles
  • Every vertex has a single, complete loop of triangles around it
• Can be relaxed to include boundary information
• Further, zero area triangles present problems in algorithms
Many ways to store meshes with various tradeoffs

- As **triangles** – three vectors per triangle ($9n_t$ memory)
- **Indexed meshes** in which indices of vertices form a triangle – one vector per vertex and three integers per triangle ($3n_v + 3n_t$ memory)

But what about connectivity?
TRIANGLE CONNECTIVITY

• In geometric algorithms, we need to quickly \(O(1)\)-time) answer things like:
  • What are adjacent triangles to a vertex/edge/triangle
  • What are adjacent triangles/edges to a vertex

• Can explicitly store a graph of adjacencies (\textit{triangle-neighbor structure}), also called the \textit{dual} of a mesh (needs about \(16n_t\) memory)
EDGE CENTRIC STRUCTURES

• Can be more clever (and efficient) if stored in an edge centric way
  • **Winged-edge structure** — stores vertices and triangles associated with each edge
  • **Half-edge structure** — provides an order to the winged-edge structure so that only half of the information needs to be stored
SCENE GRAPHS
SCENE GRAPHS

- Scene graphs provide a hierarchical representation and transformations for all objects within a scene
  - Really a forest
  - Parent transformations are applied to children objects (from the left).
  - A preorder traversal is appropriate for drawing scene
BOUNDING VOLUME HIERARCHIES
We have discussed bounding boxes and the consideration for their collision tests before.

- Similarly we have discussed spheres.
- It's just more equations with rays, but a bit easier than Barycentric Coordinates (all math omitted).

Bounding boxes can be axis aligned or oriented.
Typically, a tree data structure will be used to store a hierarchy.

- Leaves are objects (or single/groups of triangles)
- Internal nodes are groups.
- Every node has a bounding volume.
- A basic version would be binary
- Balanced trees are ideal for performance
BOUNDING VOLUME HIERARCHIES

- For a collision check (e.g., with a ray) you test with the root box. If it collides check with left and right box. Recurse only on boxes that are hit, all the way down the tree to find which object is hit.

- To encourage balance, during creation sort objects along major axis of distribution and split objects roughly evenly.
  - Alternatively have subtrees roughly contain the same amount of space.
UNIFORM SPATIAL SUBDIVISION

• Divide space by evenly spaced boxes (grid)
• Every point belongs to a single node (whereas objects may span many nodes)
• Grid can be traverse during collision checks in a similar fashion to line drawing
ADAPTIVE SPATIAL SUBDIVISION

- **Quadtrees** and **octrees** allow for an adaptive subdivision in which space is divided into 4 or 8 even sections if it contains an object (all the way down to points)
  - Recursive construction and traversal algorithms for collisions
You can also subdivide space using axis-aligned planes into a binary tree data structure.

**KD-trees** are a variant of this data structure that are very common.
**BINARY SPACE PARTITIONING FOR VISIBILITY**

- In this context, non-axis aligned planes are chosen, e.g., planes defined by triangles in the scene
  - Similar construction as all the others
- This allows for an automatic rendering order based on location of viewing (also allows for hidden surface removal)
TILING
TILING

• Grids (multi-dimensional arrays) are exceptionally common to graphics
  • Think of voxels (Minecraft)
  • Think of texels
  • Space partitioning schemes

• What is most efficient way to store this?
  • Surely in a compact manner
TILING

- However, because adjacent rows/columns usually need to be accessed together, a linear packaging of data is not always efficient with respect to a cache hierarchy.

- Instead, tiling blocks of the array together can be very useful, i.e., a two level indexing (by blocks and then by element within each block).
  - Indexing this structure uses clever modular arithmetic.