MODELING AND HIERARCHY
Introduction

- Models are abstractions of the world—both of the real world in which we live and of virtual worlds that we create with computers.
- We are all familiar with mathematical models that are used in all areas of science and engineering.
- These models use equations to model the physical phenomena that we wish to study.
- In computer science, we use abstract data types to model organizations of objects; in computer graphics, we model our worlds with geometric objects.
Our first concern is how we can store a model that may include many sophisticated objects.

There are two immediate issues:

(1) how we define an object more complex than the ones we have dealt with until now.

(2) how we represent a collection of these objects.
Modeling

- Modeling primitives
  - Polygon
  - Sphere, ellipsoid, torus, superquadric
  - Surfaces of revolutions, smoothed polygons
  - Particles
  - Skin & bones

- Approaches to modeling complex shapes
  - Tools such as extrude, revolve, loft, split, stitch, blend
  - Constructive solid geometry (CSG)
  - Hierarchy; kinematic joints
Modeling

- **Primitives**: polygons, subdivision surfaces, NURBS, particles, fluids, bones, procedural
- **Tools**: Construction, bending, twisting, shear, sweeping, stitching, blending, etc.
- **Surfacing**: bump mapping, cloth, skin, hair, fur, muscle
- **Animation**: inverse kinematics, pose blending, motion capture, keyframe animation, editable motion trails
- **Rendering**: interactive and photo-realistic, compositing, stereoscopic
Primitives

- Most common primitive is the polygon
- Number of polygons: tradeoff between render time and model accuracy
Spline Curves

- Linear spline
- Cardinal spline
- B-spline
- Bezier curve
- NURBS (non-uniform rational b-spline)
Mesh
Mesh deformations

STARR

RANDOM

ILIA

MANDELBROT
Sweep

- Sweep a shape over a path to form a generalized cylinder
Revolution

- Revolve a shape around an axis to create an object with rotational symmetry
Extrusion

- Extrude: grow a 2D shape in the third dimension
- Shape is created with a (1D) b-spline curves
- Hole was created by subtracting a cylinder
Subdivision Surfaces

- Can set level of polygon subdivision
Particles
Algorithmic Primitives

Algorithms for trees, mountains, grass, fur, lightning, fire, …
Collada

- Collaborative Design Activity: interchange file format for 3D applications
- Managed by Khronos Group (same as WebGL)
- XML-based
- Supported by a wide variety of applications including 3ds Max, Maya, Blender, Lightwave, Unity, Unreal
- Three.js support is there but buggy
- Version 1.4: skinning, morphing, animation
- Version 1.5: kinematics, asset geolocation, level of detail
Hierarchical models

- When animation is desired, objects may have parts that move with respect to each other
  - Object represented as hierarchy
  - Often there are joints with motion constraints
  - E.g. represent wheels of car as sub-objects with rotational motion (car moves $2\pi r$ per rotation)
  - Done with “parenting”
DAG models

- Could use tree to represent object
- Actually, a DAG (directed acyclic graph) is better: can re-use objects
- Note that each arrow needs a separate modeling transform
- In object-oriented graphics, also need motion constraints with each arrow
Example: Robot

- Traverse DAG using DFS (or BFS)
- Push and pop matrices along the way (e.g. left-child right-sibling)
- The figure shows a boxlike representation of a humanoid that might be used for a robot model or in a virtual reality application
If we take the torso as the root element, we can represent this figure with the tree shown in Figure below.
Our second approach is to use a standard tree data structure to represent our hierarchy and then to render it with a traversal algorithm that is independent of the model. We use a **left-child, right-sibling** structure.
The models that we developed for our two examples the robot arm and the figure are articulated: The models consist of rigid parts connected by joints.

We can make such models change their positions in time animate them by altering the values of a small set of parameters.

Hierarchical models allow us to model the compound motions incorporating the physical relationships among the parts of the model.

What we have not discussed is how to alter the parameters over time so as to achieve the desired motion.
Suppose you want the robot to pick up a can of oil to drink. How?

You could set the joint positions at each moment in the animation (kinematics).
Inverse Kinematics

- You can’t just invert the joint transformations
- Joint settings aren’t even necessarily unique for a hand position!
- **Inverse kinematics**: figure out from the hand position where the joints should be set.
Using Inverse Kinematics

- Specify joint constraints and priorities
- Move end effector (or object pose)
- Let the system figure out joint positions
- Lightwave
Keyframe Animation

- In traditional *key frame animation* the animator draws several important frames, and helpers do the “inbetweening” or “tweening”
- Computer animation is also key-frame based
- At key frames, animator positions objects and lights, sets parameters, etc.
- The system interpolates parameter values linearly or along a curve
- To get from one object pose to the next, inverse kinematics determine joint motions
Motion Capture

- More realistic motion sequences can be generated by Motion Capture
- Attach joint position indicators to real actors
- Record live action
Algorithms for Visibility

- Question: How can you do hidden-surface removal without a Z-buffer, or speed it up with a Z-buffer?
- Is there some data structure that could be computed in advance that would help?
BSP Trees

- We can use trees to describe the world object space and encapsulate the spatial relationships among groups of objects.

- These relationships can lead to fast methods of **visibility testing** to determine which objects might be seen by a camera, thus avoiding processing all objects with tests such as the z-buffer algorithm.

- These techniques have become very important in real-time animations for computer games.
BSP-tree

- The painter’s algorithm for hidden surface removal works by drawing all faces, from back to front.
- How to get a listing of the faces in back-to-front order?
- Put them into a binary tree and traverse the tree (but in what order?)
Right is “front” of polygon; left is “back”

In and Out nodes show regions of space inside or outside the object
BSP-tree Summary

- Returns polygons not necessarily in sorted order, but in an order that is correct for back-to-front rendering
- Widely used when Z-buffer hardware may not be available (e.g. game engines)
- Guarantees back-to-front rendering for alpha blending
- Works well (linear-time traversals) in the number of split polygons
- [And we hope the number of polygons doesn’t grow too much through splitting]
Handling Large Spatial Data Sets

- Example application: image-based rendering
  - Suppose you have many digital images of a scene, with depth information for pixels
  - How to find efficiently the points that are in front?

- Other applications:
  - Speeding up ray-tracing with many objects
  - Rendering contours of 3D volumetric data such as MRI scans
Quadtree

- Quadtree: divide space into four quadrants. Mark as Empty, Full, or Partially full.
- Recursively subdivide partially full regions
- Saves much time, space over 2D pixel data!
Quadtree Structure
Octrees

- Generalize to cutting up a cube into 8 sub-cubes, each of which may be E, F, or P (and subdivided)
- Much more efficient than a 3D array of cells for 3D volumetric data
Quadtree Algorithms

- How would you
  - render a quadtree shape?
  - find the intersection of a ray with a quadtree shape?
  - Take the union of two quadtrees?
  - Intersection?
  - Find the neighbors of a cell?
Applications of Octrees

- Contour finding in MRI data
- 3D scanning and rendering
- Efficient ray tracing
- Intersection, collision testing