Announcements

- Homework assignment #4 due Friday, Oct 29
- Problem #2 not due until Friday, Nov 5
- Two options for problem #4 (extra credit)
Lecture Overview

- Scene Graphs & Hierarchies
  - Performance Optimization
- Curves
  - Introduction
  - Polynomial curves
Performance Optimization

- Level-of-detail techniques
  - Use lower quality for distant (small) objects
- Culling
  - Quickly discard invisible parts of the scene
- Scene graph compilation
  - Efficient use of low-level API
  - Avoid state changes in rendering pipeline
  - Render objects with similar properties (geometry, shaders, materials) in batches
Level-of-Detail Techniques

- Don’t draw objects smaller than a threshold
  - Popping artifacts
- Replace objects by impostors
  - Textured planes representing the objects

Dynamic impostor generation

Original vs. impostor
Level-of-Detail Techniques

- Adapt triangle count to projected size

With bump mapping

Without bump mapping
Culling

- **Occlusion culling**
  - Discard objects that are within view frustum, but hidden behind other objects

- **View frustum culling**
  - Discard objects outside view frustum

- Essential for interactive performance with large scenes
Occlusion Culling

- **Cell-based occlusion culling**
  - Divide scene into cells
  - Determine *potentially visible set* (PVS) for each cell
  - Discard all cells not in PVS

- **Two main variants**
  - Precomputation using binary space partitioning (BSP) trees
  - Portal algorithms

- **Specialized algorithms for different types of geometry**
  - Indoor scenes
  - Terrain
View Frustum Culling

- Frustum defined by 6 planes
- Each plane divides space into “outside”, “inside”
- Check each object against each plane
  - Outside, inside, intersecting
- If “outside” all planes
  - Outside the frustum
- If “inside” all planes
  - Inside the frustum
- Else partly inside and partly out
- Efficiency
Bounding Volumes

- Simple shape that completely encloses an object
- Generally a box or sphere
- We use spheres
  - Easiest to work with
  - Though hard to get tight fits
- Intersect bounding volume with view frustum, instead of full geometry
Distance to Plane

- A plane is described by a point $p$ on the plane and a unit normal $n$.
- Find the (perpendicular) distance from point $x$ to the plane.
Distance to Plane

- The distance is the length of the projection of $x-p$ onto $n$

$$dist = (x - p) \cdot \tilde{n}$$
Distance to Plane

- The distance has a sign
  - positive on the side of the plane the normal points to
  - negative on the opposite side
  - zero exactly on the plane

- Divides 3D space into two infinite half-spaces

\[ dist(x) = \left( x - p \right) \cdot \hat{n} \]
Distance to Plane

- **Simplification**
  
  \[
  \text{dist}(x) = (x - p) \cdot n
  = x \cdot n - p \cdot n
  \]
  
  \[
  \text{dist}(x) = x \cdot n - d, \quad d = pn
  \]
  
  - \(d\) is independent of \(x\)
  - \(d\) is distance from the origin to the plane
  - We can represent a plane with just \(d\) and \(n\)
Frustum With Signed Planes

- Normal of each plane points outside
  - “outside” means positive distance
  - “inside” means negative distance
Test Sphere and Plane

- For sphere with radius $r$ and origin $x$, test the distance to the origin, and see if it is beyond the radius

- Three cases:
  - $dist(x) > r$
    - completely above
  - $dist(x) < -r$
    - completely below
  - $-r < dist(x) < r$
    - intersects

Positive

Negative
Culling Summary

- Precompute the normal \( \mathbf{n} \) and value \( d \) for each of the six planes.
- Given a sphere with center \( \mathbf{x} \) and radius \( r \)
- For each plane:
  - if \( \text{dist}(\mathbf{x}) > r \): sphere is outside! (no need to continue loop)
  - add 1 to count if \( \text{dist}(\mathbf{x}) < -r \)
- If we made it through the loop, check the count:
  - if the count is 6, the sphere is completely inside
  - otherwise the sphere intersects the frustum
  - (can use a flag instead of a count)
Culling Groups of Objects

- Want to be able to cull the whole group quickly
- But if the group is partly in and partly out, want to be able to cull individual objects
Hierarchical Bounding Volumes

- Given hierarchy of objects
- Bounding volume of each node encloses the bounding volumes of all its children
- Start by testing the outermost bounding volume
  - If it is entirely outside, don’t draw the group at all
  - If it is entirely inside, draw the whole group
Hierarchical Culling

- If the bounding volume is partly inside and partly outside
  - Test each child’s bounding volume individually
  - If the child is in, draw it; if it’s out cull it; if it’s partly in and partly out, recurse.
  - If recursion reaches a leaf node, draw it normally
Lecture Overview

- Scene Graphs & Hierarchies
  - Performance Optimization
- Curves
  - Introduction
  - Polynomial curves
Modeling

- Creating 3D objects
- How to construct complex surfaces?
- Goal
  - Specify objects with control points
  - Objects should be visually pleasing (smooth)
- Start with curves, then generalize to surfaces

- Next: What can curves be used for?
Curves

- Surface of revolution (homework project!)
Curves

- Extruded/swept surfaces
Curves

- **Animation**
  - Provide a “track” for objects
  - Use as camera path
Curves

- Specify parameter values over time
Curves

- Can be generalized to surface patches
Curve Representation

- Specify every point along a curve?
  - Hard to get precise, smooth results
  - Too much data, difficult to work with

- Specify a curve using a small number of “control points”
  - Known as a *spline curve* or just *spline*
Spline: Definition

- **Wikipedia:**
  - Term comes from flexible spline devices used by shipbuilders and draftsmen to draw smooth shapes
  - Spline consists of a long strip fixed in position at a number of points that relaxes to form a smooth curve passing through those points
Interpolating Splines

- Curve goes through all control points
- Seems most intuitive
- Surprisingly, not usually the best choice
- Hard to predict behavior
  - Overshoots, wiggles
- Hard to get “nice-looking” curves
Approximating Splines

- Curve is “influenced” by control points

- Various types & techniques
- Most common: polynomial functions
  - Bézier spline
  - B-spline (generalization of Bézier spline)
  - NURBS (Non Uniform Rational Basis Spline)

- In this lecture: focus on Bézier splines
Mathematical Definition

- A vector valued function of one variable $\mathbf{x}(t)$
  - Given $t$, compute a 3D point $\mathbf{x}=(x, y, z)$
  - May interpret as three functions $x(t)$, $y(t)$, $z(t)$
  - “Moving a point along the curve”
Tangent Vector

- Derivative $\mathbf{x}'(t) = \frac{d\mathbf{x}}{dt} = (x'(t), y'(t), z'(t))$
- A vector that points in the direction of movement
- Length corresponds to speed
Lecture Overview

- Scene Graphs & Hierarchies
  - Performance Optimization
- Curves
  - Introduction
  - Polynomial curves
Polynomial Functions

- **Linear:** \( f(t) = at + b \)
  (1\text{st} order)

- **Quadratic:** \( f(t) = at^2 + bt + c \)
  (2\text{nd} order)

- **Cubic:** \( f(t) = at^3 + bt^2 + ct + d \)
  (3\text{rd} order)
Polynomial Curves

- **Linear** \( \mathbf{x}(t) = \mathbf{a}t + \mathbf{b} \)

  \( \mathbf{x} = (x, y, z), \mathbf{a} = (a_x, a_y, a_z), \mathbf{b} = (b_x, b_y, b_z) \)

- **Evaluated as:**
  
  \[
  \begin{align*}
  x(t) &= a_xt + b_x \\
  y(t) &= a_yt + b_y \\
  z(t) &= a_zt + b_z
  \end{align*}
  \]
Polynomial Curves

- **Quadratic:** \( x(t) = at^2 + bt + c \) (2\(^{\text{nd}}\) order)

- **Cubic:** \( x(t) = at^3 + bt^2 + ct + d \) (3\(^{\text{rd}}\) order)

- We usually define the curve for \( 0 \leq t \leq 1 \)
Control Points

- Polynomial coefficients $a, b, c, d$ can be interpreted as control points
  - Remember: $a, b, c, d$ have $x, y, z$ components each
- Unfortunately, they don’t intuitively describe the shape of the curve
- Main objective of curve representation is to come up with intuitive control points
Control Points

- How many control points?
  - Two points define a line (1\textsuperscript{st} order)
  - Three points define a quadratic curve (2\textsuperscript{nd} order)
  - Four points define a cubic curve (3\textsuperscript{rd} order)
  - $k+1$ points define a $k$-order curve

- Let’s start with a line...
First Order Curve

- Based on linear interpolation (LERP)
  - Weighted average between two values
  - “Value” could be a number, vector, color, …
- Interpolate between points \( p_0 \) and \( p_1 \) with parameter \( t \)
  - Defines a “curve” that is straight (first-order spline)
  - \( t=0 \) corresponds to \( p_0 \)
  - \( t=1 \) corresponds to \( p_1 \)
  - \( t=0.5 \) corresponds to midpoint

\[
x(t) = Lerp(t, p_0, p_1) = (1-t)p_0 + t \cdot p_1
\]
Linear Interpolation

- Three different ways to write it
  - All equivalent
  - Different properties become apparent

1. **Weighted sum of the control points**
   \[ x(t) = p_0(1 - t) + p_1 t \]

2. **Polynomial in \( t \)**
   \[ x(t) = (p_1 - p_0)t + p_0 \]

3. **Matrix form**
   \[
   x(t) = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}
   \]
Weighted Average

\[ x(t) = (1 - t)p_0 + t p_1 \]

\[ = B_0(t) p_0 + B_1(t) p_1, \text{ where } B_0(t) = 1 - t \text{ and } B_1(t) = t \]

- Weights are a function of \( t \)
  - Sum is always 1, for any value of \( t \)
  - Also known as *blending functions*
Linear Polynomial

\[ x(t) = (p_1 - p_0) t + p_0 \]

- Curve is based at point \( p_0 \)
- Add the vector, scaled by \( t \)
Matrix Form

\[ x(t) = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix} = GBT \]

- **Geometry matrix**
  \[ G = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \]

- **Geometric basis**
  \[ B = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \]

- **Polynomial basis**
  \[ T = \begin{bmatrix} t \\ 1 \end{bmatrix} \]

- **In components**
  \[ x(t) = \begin{bmatrix} p_{0x} & p_{1x} \\ p_{0y} & p_{1y} \\ p_{0z} & p_{1z} \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix} \]
Tangent

- For a straight line, the tangent is constant
  \[ x'(t) = p_1 - p_0 \]

- Weighted average
  \[ x'(t) = (-1)p_0 + (+1)p_1 \]

- Polynomial
  \[ x'(t) = 0t + (p_1 - p_0) \]

- Matrix form
  \[ x'(t) = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]
Lissajous Curves

- Live demo: [http://ibiblio.org/e-notes/Lis/Lissa.htm](http://ibiblio.org/e-notes/Lis/Lissa.htm)

[Image of a Lissajous Curve]

Next Lecture

- Bezier curves
- Curves with multiple segments
- Extension to surfaces