Last time

☐ Modeling Techniques
Today

- More modeling techniques
- Splines
- Homework 5 available, due 12/06, via email
- Pick a slot for grading Project 2
  - [https://docs.google.com/spreadsheets/d/1e4VXnY2I Pp7Dbk9_Pb8jcBmpKfE_35EN--MTIbp8LSc/edit?usp=sharing_eid&ts=56556945](https://docs.google.com/spreadsheets/d/1e4VXnY2I Pp7Dbk9_Pb8jcBmpKfE_35EN--MTIbp8LSc/edit?usp=sharing_eid&ts=56556945)
Implicit Functions

- Some surfaces can be represented as the vanishing points of functions (defined over 3D space)
  - Places where a function $f(x,y,z)=0$
- Some objects are easy to represent this way
  - Spheres, ellipses, and similar
  - More generally, quadratic surfaces:
    $$ax^2 + bx + cy^2 + dy + ez^2 + fz + g = 0$$
  - Shapes depends on all the parameters $a,b,c,d,e,f,g$
Blobs and Metaballs

- Define the location of some points, \( p_i \).
- For each point, define a function on the distance to a given point, \( D(x, p_i) \).
- Sum these functions up, and use them as an implicit function.
- Often, use Gaussian functions of distance, or other forms.
  - Various results are called blobs or metaballs.
Example with Blobs

Rendered with POVray. Not everything is a blob, but the characters are.
Blob Math

- Implicit equation: \( f(x, y, z) = w_0 + \sum_{i=1}^{n_{blobs}} w_i g_i(x, y, z) = 0 \)

- The \( w_i \) are weights - just numbers

- The \( g_i \) are functions, one common choice is:

  \[ g_i(x) = e^{-\frac{\|x-c_i\|^2}{\sigma_i}} \]

- \( c_i \) and \( \sigma_i \) are parameters
Rendering Implicit Surfaces

- Some methods can render them directly
  - Raytracing - find intersections with Newton’s method

- For polygonal renderer, must convert to polygons
  - *Marching Cubes* algorithm
  - Also used for finding *iso-surfaces, or level sets*, in volume data
Implicit Surfaces Summary

- **Advantages:**
  - Good for organic looking shapes eg human body
  - Good for extracting surfaces from volume representations, such as water surfaces in fluid simulation
  - Easy inside/outside testing

- **Disadvantages:**
  - Difficult to render
  - Difficult to control when animating
What are Parametric Curves?

- Define a parameter space
  - 1D for curves: $t$
  - 2D for surfaces: $(s,t)$

- Define a mapping from parameter space to 3D points
  - A function that takes parameter values and gives back 3D points

- The result is a parametric curve or surface
Why Parametric Curves?

- Parametric curves are intended to provide the generality of polygon meshes but with fewer parameters for smooth surfaces
  - Polygon meshes have as many parameters as there are vertices (at least)
- Fewer parameters make it faster to create a curve, and easier to edit an existing curve
- Normal vectors and texture coordinates can be easily defined everywhere
- Parametric curves are easier to animate than polygon meshes
Parametric Curves

- We have seen the parametric form for a line:

\[x = (1 - t)x_0 + tx_1\]
\[y = (1 - t)y_0 + ty_1\]
\[z = (1 - t)z_0 + tz_1\]

- Note that x, y and z are each given by an equation that involves:
  - The parameter \(t\)
  - Some user specified control points, \(x_0\) and \(x_1\)

- This is an example of a parametric curve
Basis Functions (first sighting)

- A line is the sum of two functions multiplied by vectors:
  \[
  \begin{bmatrix}
  x \\
  y \\
  z
  \end{bmatrix}
  = (1-t) \begin{bmatrix}
  x_0 \\
  y_0 \\
  z_0
  \end{bmatrix} + t \begin{bmatrix}
  x_1 \\
  y_1 \\
  z_1
  \end{bmatrix}
  \]

- A linear combination of basis functions
  - \( t \) and \( 1-t \) are the basis functions

- The weights are called control points
  - \((x_0,y_0,z_0)\) and \((x_1,y_1,z_1)\) are the control points
  - They control the shape and position of the curve
A spline is a parametric curve defined by control points

- The term spline dates from engineering drawing, where a spline was a piece of flexible wood used to draw smooth curves
- The control points are adjusted by the user to control the shape of the curve

A Hermite spline is a curve for which the user provides:

- The endpoints of the curve
- The parametric derivatives of the curve at the endpoints (tangents with length)
- The parametric derivatives are $dx/dt, dy/dt, dz/dt$
- That is enough to define a cubic Hermite spline
Spline

Control Point Interpretation

Start Tangent:

\[ \frac{dx}{dt} \bigg|_0 \]

End Tangent:

\[ \frac{dx}{dt} \bigg|_1 \]

Start Point: \( X_0 \)

End Point: \( X_1 \)
Hermite Spline (2)

☐ Say the user provides \( x_0, x_1, \frac{dx_0}{dt}, \frac{dx_1}{dt} \), \( t=0 \), \( t=1 \),\( d \), \( c \), \( b \), \( a \)

☐ A cubic spline has degree 3, and is of the form:

\[
x = at^3 + bt^2 + ct + d
\]

- For some constants \( a, b, c \) and \( d \) derived from the control points, but how?

☐ We have constraints:

- The curve must pass through \( x_0 \) when \( t=0 \)
- The derivative must be \( x'_0 \) when \( t=0 \)
- The curve must pass through \( x_1 \) when \( t=1 \)
- The derivative must be \( x'_1 \) when \( t=1 \)
Hermite Spline (3)

- Solving for the unknowns gives:
  
  \[
  a = -2x_1 + 2x_0 + x'_1 + x'_0 \\
  b = 3x_1 - 3x_0 - x'_1 - 2x'_0 \\
  c = x'_0 \\
  d = x_0 
  \]

- Rearranging gives:

  \[
  x = x_1(-2t^3 + 3t^2) + x_0(2t^3 - 3t^2 + 1) + x'_1(t^3 - t^2) + x'_0(t^3 - 2t^2 + t)
  \]

or

\[
 x = \begin{bmatrix}
 x_1 & x_0 & x'_1 & x'_0 \\
 \end{bmatrix} \begin{bmatrix}
 -2 & 3 & 0 & 0 \\
 2 & -3 & 0 & 1 \\
 1 & -1 & 0 & 0 \\
 1 & -2 & 1 & 0 \\
 \end{bmatrix} \begin{bmatrix}
 t^3 \\
 t^2 \\
 t \\
 1 \\
 \end{bmatrix}
\]
A point on a Hermite curve is obtained by multiplying each control point by some function and summing.
Splines in 2D and 3D

For higher dimensions, define the control points in higher dimensions (that is, as vectors)

\[
\begin{bmatrix}
    x \\
    y \\
    z
\end{bmatrix}
= 
\begin{bmatrix}
    x_1 & x_0 & x'_1 & x'_0 \\
    y_1 & y_0 & y'_1 & y'_0 \\
    z_1 & z_0 & z'_1 & z'_0
\end{bmatrix}
\begin{bmatrix}
    -2 & 3 & 0 & 0 \\
    2 & -3 & 0 & 1 \\
    1 & -1 & 0 & 0 \\
    1 & -2 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    t^3 \\
    t^2 \\
    t \\
    1
\end{bmatrix}
\]
Beziers Curves (1)

- Different choices of basis functions give different curves
  - Choice of basis determines how the control points influence the curve
  - In Hermite case, two control points define endpoints, and two more define parametric derivatives
- For Bezier curves, two control points define endpoints, and two control the tangents at the endpoints in a geometric way
Control Point Interpretation

Point along start tangent

Start Point

Point along end Tangent

End Point
Beziers Curves (2)

- The user supplies $d+1$ control points, $p_i$
- Write the curve as:
  $$x(t) = \sum_{i=0}^{d} p_i B_i^d(t) \quad B_i^d(t) = \binom{d}{i} t^i (1-t)^{d-i}$$
- The functions $B_i^d$ are the Bernstein polynomials of degree $d$
- This equation can be written as a matrix equation also
  - There is a matrix to take Hermite control points to Bezier control points
Beziers Basis Functions for $d=3$
Bezier Curves of Varying Degree
Bezier Curve Properties

- The first and last control points are interpolated.
- The tangent to the curve at the first control point is along the line joining the first and second control points.
- The tangent at the last control point is along the line joining the second last and last control points.
- The curve lies entirely within the convex hull of its control points.
  - The Bernstein polynomials (the basis functions) sum to 1 and are everywhere positive.
- They can be rendered in many ways.
  - E.g.: Convert to line segments with a subdivision algorithm.
Rendering Bezier Curves (1)

- Evaluate the curve at a fixed set of parameter values and join the points with straight lines

- Advantage: Very simple

- Disadvantages:
  - Expensive to evaluate the curve at many points
  - No easy way of knowing how fine to sample points, and maybe sampling rate must be different along the curve
  - No easy way to adapt. In particular, it is hard to measure the deviation of a line segment from the exact curve
Rendering Bezier Curves (2)

- Recall that a Bezier curve lies entirely within the convex hull of its control vertices
- If the control vertices are nearly collinear, then the convex hull is a good approximation to the curve
- Also, a cubic Bezier curve can be subdivided into two shorter curves that exactly cover the original
- This suggests an algorithm:
  - Keep breaking the curve into sub-curves
  - Stop when the control points of each sub-curve are nearly collinear
  - Draw the control polygon - the polygon formed by the control points
Invariance

- *Translational invariance* means that translating the control points and then evaluating the curve is the same as evaluating and then translating the curve.

- *Rotational invariance* means that rotating the control points and then evaluating the curve is the same as evaluating and then rotating the curve.

- These properties are essential for parametric curves used in graphics.

- It is easy to prove that Bezier curves, Hermite curves and everything else we will study are translation and rotation invariant.

- Some forms of curves, *rational splines*, are also *perspective invariant*:
  - Can do perspective transform of control points and *then* evaluate the curve.
Longer Curves

- A single cubic Bezier or Hermite curve can only capture a small class of curves
- One solution is to raise the degree
  - Allows more control, at the expense of more control points and higher degree polynomials
  - Control is not local, one control point influences entire curve
- Alternate, most common solution is to join pieces of cubic curve together into piecewise cubic curves
  - Total curve can be broken into pieces, each of which is cubic
  - Local control: Each control point only influences a limited part of the curve
  - Interaction and design is much easier
Pieewise Bezier Curve

$P_{0,0}$ $P_{0,1}$ $P_{0,2}$ $P_{0,3}$ $P_{1,0}$ $P_{1,1}$ $P_{1,2}$ $P_{1,3}$
When two curves are joined, we typically want some degree of continuity across the boundary (the knot):

- $C^0$, “C-zero”, point-wise continuous, curves share the same point where they join
- $C^1$, “C-one”, continuous derivatives, curves share the same parametric derivatives where they join
- $C^2$, “C-two”, continuous second derivatives, curves share the same parametric second derivatives where they join
- Higher orders possible
Bezizer Continuity

Disclaimer: PowerPoint curves are not Bezier curves, they are interpolating piecewise quadratic curves! This diagram is an approximation.
Sketch of Proof for $C^1$

Bezier curve equation:

$$\mathbf{x} = \mathbf{x}_0 (1-t)^3 + \mathbf{x}_1 3t(1-t)^2 + \mathbf{x}_2 3t^2 (1-t) + \mathbf{x}_3 t^3$$

$$= \mathbf{x}_0 (1-3t + 3t^2 - t^3) + \mathbf{x}_1 3(t - 2t^2 + t^3) + \mathbf{x}_2 3(t^2 - t^3) + \mathbf{x}_3 t^3$$

Parametric derivative:

$$\frac{d\mathbf{x}}{dt} = \mathbf{x}_0 (-3 + 6t - 3t^2) + \mathbf{x}_1 3(1-4t + 3t^2) + \mathbf{x}_2 3(2t - 3t^2) + \mathbf{x}_3 3t^2$$

Evaluated at endpoint of curve (note proves tangent property):

$$\left. \frac{d\mathbf{x}}{dt} \right|_{t=0} = -3\mathbf{x}_0 + 3\mathbf{x}_1 = 3(\mathbf{x}_1 - \mathbf{x}_0)$$

$$\left. \frac{d\mathbf{x}}{dt} \right|_{t=1} = -3\mathbf{x}_2 + 3\mathbf{x}_3 = 3(\mathbf{x}_3 - \mathbf{x}_2)$$
Proof (cont)

$P_{0,0}, P_{0,1}, P_{0,2}, J, P_{1,1}, P_{1,2}, P_{1,3}$

$C^1$ requires equal parametric derivatives:

$$3(P_{1,1} - J) = 3(J - P_{0,2})$$

$$P_{1,1} - J = J - P_{0,2}$$
Geometric Continuity

- Derivative continuity is important for animation
  - If an object moves along the curve with constant parametric speed, there should be no sudden jump at the knots

- For other applications, *tangent continuity* might be enough
  - Requires that the tangents point in the same direction
  - Referred to as \( G^1 \) *geometric continuity*
  - Curves *could* be made \( C^1 \) with a re-parameterization: \( u = f(t) \)
  - The geometric version of \( C^2 \) is \( G^2 \), based on curves having the same radius of curvature across the knot

- What is the tangent continuity constraint for a Bezier curve?
Bezizer Geometric Continuity

\[(P_{1,1} - J) = k(J - P_{0,2}) \quad \text{for some } k\]
Bezier Curve/Surface Problems

- To make a long continuous curve with Bezier segments requires using many segments
  - Same for large surface
- Maintaining continuity requires constraints on the control point positions
  - The user cannot arbitrarily move control vertices and automatically maintain continuity
  - The constraints must be explicitly maintained
  - It is not intuitive to have control points that are not free
B-splines

- B-splines automatically take care of continuity, with exactly one control vertex per curve segment
- Many types of B-splines: degree may be different (linear, quadratic, cubic, etc) and they may be uniform or non-uniform
  - We will only look closely at uniform B-splines
- With uniform B-splines, continuity is always one degree lower than the degree of each curve piece
  - Linear B-splines have $C^0$ continuity, cubic have $C^2$, etc
Uniform Cubic B-spline on [0,1)

- Four control points are required to define the curve for $0 \leq t < 1$ ($t$ is the parameter)
  - Not surprising for a cubic curve with 4 degrees of freedom
- The equation looks just like a Bezier curve, but with different basis functions
  - Also called *blending functions* - they describe how to blend the control points to make the curve (see Shirley book ch. 15.6)

$$x(t) = \sum_{i=0}^{3} P_i B_{i,4}(t)$$

$$= P_0 \frac{1}{6}(1 - 3t + 3t^2 - t^3) + P_1 \frac{1}{6}(4 - 6t^2 + 3t^3) + P_2 \frac{1}{6}(1 + 3t + 3t^2 - 3t^3) + P_3 \frac{1}{6}(t^3)$$
Basis Functions on \([0,1)\)

- Does the curve interpolate its endpoints?
- Does it lie inside its convex hull?

\[
x(t) = P_0 \frac{1}{6} \left(1 - 3t + 3t^2 - t^3\right) + P_1 \frac{1}{6_1} \left(4 - 6t^2 + 3t^3\right) + P_2 \frac{1}{6} \left(1 + 3t + 3t^2 - 3t^3\right) + P_3 \frac{1}{6} (t^3)
\]
Uniform Cubic B-spline on [0,1)

- The blending functions sum to one, and are positive everywhere
  - The curve lies inside its convex hull
- The curve does not interpolate its endpoints
  - Requires hacks or non-uniform B-splines
- There is also a matrix form for the curve:

\[
x(t) = \frac{1}{6} \begin{bmatrix} P_0 & P_1 & P_2 & P_3 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}
\]
Demo
Uniform B-spline at Arbitrary t

- The interval from an integer parameter value \( v \) to \( v+1 \) is essentially the same as the interval from 0 to 1
  - The parameter value is offset by \( v \)
  - A different set of control points is needed

- To evaluate a uniform cubic B-spline at an arbitrary parameter value \( t \):
  - Find the greatest integer less than or equal to \( t \): \( v = \text{floor}(t) \)
  - Evaluate:
    \[
    X(t) = \sum_{i=0}^{3} P_{i+v} B_{i,4}(t-v)
    \]

- Valid parameter range: \( 0 \leq t < n-3 \), where \( n \) is the number of control points
Loops

- To create a loop, use control points from the start of the curve when computing values at the end of the curve:

\[ X(t) = \sum_{i=0}^{3} P_{(i+v) \mod n} B_{i,4}(t - v) \]

- Any parameter value is now valid
  - Although for numerical reasons it is sensible to keep it within a small multiple of \( n \)
Demo
B-splines and Interpolation, Continuity

☐ Uniform B-splines do not interpolate control points, unless:
  ■ You repeat a control point three times
  ■ But then all derivatives also vanish (=0) at that point
☐ To align tangents, use double control vertices
  ■ Then tangent aligns similar to Bezier curve
☐ Uniform B-splines are automatically $C^2$
How to Choose a Spline

- Hermite curves are good for single segments where you know the parametric derivative or want easy control of it.
- Bezier curves are good for single segments or patches where a user controls the points.
- B-splines are good for large continuous curves and surfaces.
Next Time

☐ Raytracing