CS 536
Computer Graphics
Intro to Curves
Week 1, Lecture 2
David Breen, William Regli and Maxim Peysakhov
Department of Computer Science
Drexel University

Outline
• Math review
• Introduction to 2D curves
• Functional representations
• Parametric cubic curves
• Introduction to Bézier curves

Geometric Preliminaries
• Affine Geometry
  – Scalars + Points + Vectors and their ops
• Euclidean Geometry
  – Affine Geometry lacks angles, distance
  – New op: Inner/Dot product, which gives
    • Length, distance, normalization
    • Angle, Orthogonality, Orthogonal projection
• Projective Geometry

Affine Geometry
• Affine Operations:
  • Affine Combinations:
    \[ \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n \]

Mathematical Preliminaries
• Vector: an n-tuple of real numbers
• Vector Operations
  • Vector addition: \[ u + v = w \]
  • Commutative
  • Identity element: \( 0 \)
  • Scalar multiplication: \[ c \cdot v \]
• Note: Vectors and Points are different
  • Can not add points
  • Can find the vector between two points

Linear Combinations & Dot Products
• A linear combination of the vectors
  \[ v_1, v_2, \ldots, v_n \]
  is any vector of the form
  \[ \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n \]
  where \( \alpha_i \) is a real number (i.e. a scalar)
• Dot Product:
  \[ u \cdot v = \sum_{k=1}^{m} u_k v_k \]
  a real value \( w = w_1 + w_2 + \ldots + w_n \) written as \( u \cdot v \)
Fun with Dot Products

• Euclidian Distance from \((x,y)\) to \((0,0)\)
  \[
  \sqrt{x^2 + y^2}
  \]
  in general:
  \[
  \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}
  \]
  which is just:
  \[
  \sqrt{x \cdot x}
  \]
• This is also the length of vector \(\mathbf{v}\):
  \[
  \|\mathbf{v}\| \quad \text{or} \quad |\mathbf{v}|
  \]
• Normalization of a vector: \(\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}\)
• Orthogonal vectors: \(\hat{\mathbf{u}} \cdot \hat{\mathbf{v}} = 0\)

Projections & Angles

• Angle between vectors \(\theta\)
  \[
  \hat{\mathbf{u}} \cdot \hat{\mathbf{v}} = \|\mathbf{u}\|\|\mathbf{v}\|\cos(\theta)
  \]
  \[
  \theta = \arccos\left(\frac{\hat{\mathbf{u}} \cdot \hat{\mathbf{v}}}{\|\mathbf{u}\|\|\mathbf{v}\|}\right)
  \]
• Projection of vectors
  \[
  \hat{\mathbf{u}}_1 = \frac{(\hat{\mathbf{u}} \cdot \hat{\mathbf{v}})}{(\hat{\mathbf{v}} \cdot \hat{\mathbf{v}})} \hat{\mathbf{v}}
  \]
  \[
  \hat{\mathbf{u}}_2 = \mathbf{u} - \hat{\mathbf{u}}_1
  \]

Matrices and Matrix Operators

• A \(n\)-dimensional vector:
  \[
  \begin{bmatrix}
  c_1 \\
  c_2 \\
  \vdots \\
  c_n
  \end{bmatrix}
  \]
• Matrix Operations:
  - Addition/Subtraction
  - Identity
  - Multiplication
    - \(A\times B\)
  - Matrix Multiplication
  - Implementation Issues:
    Where does the index start?
    (0 or 1, it’s up to you…)
• Identity Matrix:
  \[
  I = \begin{bmatrix}
  1 & 0 & \cdots & 0 \\
  0 & 1 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & 1
  \end{bmatrix}
  \]
  \[
  A \times I = A
  \]
  \[
  I \times A = A
  \]

Matrix Multiplication

• \([C] = [A][B]\)
• Sum over rows & columns
• Recall: matrix multiplication is not commutative
• Identity Matrix:
  \[
  I = \sum_{i,j} a_{ij}b_{ij}
  \]
  \[
  \begin{bmatrix}
  a_{11} & a_{21} & \cdots & a_{n1} \\
  a_{12} & a_{22} & \cdots & a_{n2} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{1n} & a_{2n} & \cdots & a_{nn}
  \end{bmatrix}
  \]

Matrix Determinants

• A single real number
• Computed recursively
  \[
  \det(A) = \sum_{j=1}^{n} A_{ij}(-1)^{i+j} M_{ij}
  \]
• Example:
  \[
  \det\begin{bmatrix}
  a & b \\
  c & d
  \end{bmatrix} = ad - bc
  \]
• Uses:
  - Find vector orthogonal to two other vectors
  - Determine the plane of a polygon

Cross Product

• Given two non-parallel vectors, \(A\) and \(B\)
• \(A \times B\) calculates third vector \(C\) that is orthogonal to \(A\) and \(B\)
• \(A \times B = (a_1b_2 - a_2b_1, a_2b_3 - a_3b_2, a_3b_1 - a_1b_3)\)

\[
A \times B = \begin{bmatrix}
  \hat{x} & \hat{y} & \hat{z} \\
  a_1 & a_2 & a_3 \\
  b_1 & b_2 & b_3
  \end{bmatrix}
\]

\[
\begin{bmatrix}
  \hat{x} & \hat{y} & \hat{z} \\
  a_1 & a_2 & a_3 \\
  b_1 & b_2 & b_3
  \end{bmatrix}
\]
Matrix Transpose & Inverse

- **Matrix Transpose:**
  Swap rows and cols:  
  \[ A = \begin{bmatrix} 2 \\ 8 \end{bmatrix} \quad A^T = \begin{bmatrix} 2 & 8 \end{bmatrix} \]
- **Facts about the transpose:**
  \( (A^T)^T = A \)  
  \( (A + B)^T = A^T + B^T \)  
  \( (cA)^T = cA^T \)
- **Matrix Inverse:**
  Given \( A \), find \( B \) such that  
  \( AB = BA = I \)  
  \( B = A^{-1} \)  
  (only defined for square matrices)

Derivatives of Polynomials

- \( f(x) = \alpha x^n \)
- \( \frac{df(x)}{dx} = \alpha nx^{n-1} \)
- \( f(x) = 5x^3 \)
- \( \frac{df(x)}{dx} = 15x^2 \)

Partial Derivatives of Polynomials

- \( f(x, y) = \alpha x^n y^m \)
- \( \frac{df(x, y)}{dx} = \alpha nx^{n-1} y^m \)
- \( f(x, y) = 5x^3 y^4 \)
- \( \frac{df(x, y)}{dx} = 15x^2 y^4 \)

Example Application: Font Design and Display

- Curved objects are everywhere
- There is always need for:
  - mathematical fidelity
  - high precision
  - artistic freedom and flexibility
  - physical realism

Example Application: Graphic Design and Arts

- Pic courtesy of G. Fair & ASU

![Example Image](http://www.pilot3d.com)
Example Application: Tool Path Generation

Example Application: Motion Planning

Functional Representations

• **Explicit Functions:**
  - representing one variable with another
  - fine if \( \exists \) only one \( x \) value for each \( y \) value
  - Problem: what if I have a sphere?

\[
z = \pm \sqrt{r^2 - x^2 - y^2}
\]

• Multiple values .... (not used in graphics)

Functional Representations

• **Implicit Functions:**
  - curves/surfaces represented as “the zeros”
  - good for rep. of \( (n-1)D \) objects in \( nD \) space
  - Sphere example: \( x^2 + y^2 + z^2 = r^2 \)
  - What class of function?
    - polynomial: linear combo of integer powers of \( x,y,z \)
    - algebraic curves & surfaces: rep’d by implicit polynomial functions
    - polynomial degree: total sum of powers, i.e. polynomial of degree 6 : \( x^6 + y^2 + z^2 - r^2 = 0 \)

• **Parametric Functions:**
  - 2D/3D curve: two functions of one parameter
    \( (x(u), y(u)) \)
  - 3D surface: three functions of two parameters
    \( (x(u,v), y(u,v), z(u,v)) \)

  - Example: Sphere
    \( x(\theta, \phi) = \cos \phi \cos \theta \)
    \( y(\theta, \phi) = \cos \phi \sin \theta \)
    \( z(\theta, \phi) = \sin \phi \)

  - Note: rep. not algebraic, but is parametric

• Which is best??
  - It depends on the application
  - Implicit is good for
    - computing ray/surface intersection
    - point inclusion (inside/outside test)
    - mass & volume properties
  - Parametric is good for
    - subdivision, faceting for rendering
    - Surface & area properties
    - popular in graphics
Issues in Specifying/Designing Curves/Surfaces
• Note: the internal mathematical representation can be very complex
  – high degree polynomials
  – hard to see how parameters relate to shape
• How do we deal with this complexity?
  – Use curve control points and either
    • Interpolate
    • Approximate

Points to Curves
• The Lagrangian interpolating polynomial
  – \( n+1 \) points, the unique polynomial of degree \( n \)
  – curve wiggles thru each control point
  – Issue: not good if you want smooth or flat curves
• Approximation of control points
  – points are weights that tug on the curve or surface

Warning, Warning, Warning: Pending Notation Abuse
• \( t \) and \( u \) are used interchangeably as a parameterization variable for functions
• Why?
  – \( t \) historically is “time”, certain parametric functions can describe “change over time” (e.g. motion of a camera, physics models)
  – \( u \) comes from the 3D world, i.e. where two variables define a B-spline surface
    • \( u \) and \( v \) are the variables for defining a surface
    • Choice of \( t \) or \( u \) depends on the text/reference

Parametric Curves
• General rep:
  \[ x = x(t), \quad y = y(t) \]
• Properties:
  – individual functions are single-valued
  – approximations are done with piecewise polynomial curves
  – Each segment is given by two cubic polynomials \((x,y)\) in parameter \( t \)
  – Concise representation

Cubic Parametric Curves
• Balance between
  – Complexity
  – Control
  – Wiggles
  – Amount of computation
  – Non-planar

Parametric Curves
• Cubic Polynomials that define a parametric curve segment
  \[ Q(t) = [x(t) \quad y(t) \quad z(t)]^T \]
  \( t \) to be \( 0 \leq t \leq 1 \).
  are of the form
  \[
  \begin{align*}
  x(t) &= a_xt^3 + b_xt^2 + c_xt + d_x \\
  y(t) &= a_yt^3 + b_yt^2 + c_yt + d_y \\
  z(t) &= a_zt^3 + b_zt^2 + c_zt + d_z
  \end{align*}
  \]
  \( 0 \leq t \leq 1. \)
Parametric Curves

- If coefficients are represented as a matrix:
  \[
  C = \begin{bmatrix}
  a_0 & b_0 & c_0 & d_0 \\
  a_1 & b_1 & c_1 & d_1 \\
  a_2 & b_2 & c_2 & d_2 \\
  a_3 & b_3 & c_3 & d_3
  \end{bmatrix}
  \]
  and
  \[
  T = \begin{bmatrix}
  t^3 & t^2 & t & 1
  \end{bmatrix}^T
  \]
  then:
  \[
  Q(t) = [x(t) \ y(t) \ z(t)] = C \cdot T
  \]

Parametric Curves

- \( Q(t) \) can be defined with four constraints:
  - Rewrite the coefficient matrix \( C = G \cdot M \)
    where \( M \) is a 4x4 basis matrix, and \( G \) is a four-element constraint matrix (geometry matrix).
  - Expanding \( Q(t) = G \cdot M \cdot T \) gives:
    \[
    Q(t) = \begin{bmatrix}
    x(t) \\
    y(t) \\
    z(t)
    \end{bmatrix} = \begin{bmatrix}
    m_{11} & m_{12} & m_{13} & m_{14} \\
    m_{21} & m_{22} & m_{23} & m_{24} \\
    m_{31} & m_{32} & m_{33} & m_{34}
    \end{bmatrix} \begin{bmatrix}
    t^3 \\
    t^2 \\
    t \\
    1
    \end{bmatrix}
    \]

  \( Q(t) \) is a weighted sum of the columns of the geometry matrix, each of which represents a point or vector in 3-space.

Parametric Curves

- Multiplying out \( Q(t) = G \cdot M \cdot T \) gives
  \[
  x(t) = m_{11} t^3 + m_{12} t^2 + m_{13} t + m_{14}
  \]
  \[
  y(t) = m_{21} t^3 + m_{22} t^2 + m_{23} t + m_{24}
  \]
  \[
  z(t) = m_{31} t^3 + m_{32} t^2 + m_{33} t + m_{34}
  \]
  (i.e. just weighted sums of the elements)
  - The weights are cubic polynomials in \( t \) (called the blending functions, \( B = M \cdot T \)), \( Q(t) = G \cdot B \)
  - \( M \) and \( G \) matrices vary by curve
    - Hermite, Bézier, spline, etc.

Some Types of Curves

- **Hermite**
  - Defined by two end points and two tangent vectors
- **Bézier**
  - Two end points plus two control points for the tangent vectors

Splines

- **Basis Splines**
  - Defined with 4 control points
- **Uniform, nonrational B-splines**
- **Nonuniform, nonrational B-splines**
- **Nonuniform, rational B-splines [NURBS]**

Convex Hulls

- The smallest convex container of a set of points
- Both practically and theoretically useful in a number of applications

Bézier Curves

- Pierre Bézier @ Renault ~1960
- Basic idea
  - Four points
    - Start point \( P_0 \)
    - End point \( P_3 \)
    - Tangent at \( P_3 \) and \( P_0 \)
    - Tangent at \( P_2 \) and \( P_1 \)
Bézier Curves

An Example:
- **Geometry matrix** is 
  \[ G_B = [p_1 \ p_2 \ p_3 \ p_4] \]
  where \( p_i \) are control points for the curve
- **Basis Matrix** is
  \[
  M_B = \begin{bmatrix}
  -1 & 1 & 0 & 0 \\
  3 & -3 & 1 & 0 \\
  -3 & 3 & 0 & 0 \\
  1 & 0 & 0 & 0
  \end{bmatrix}
  \]

Bézier Curves

- The general representation of a Bézier curve is
  \[ Q(t) = G_B \cdot M_B \cdot T \]
  where
  \[ G_B - \text{Bézier Geometry Matrix} \]
  \[ M_B - \text{Bézier Basis Matrix} \]
  which is (multiplying out):
  \[ Q(t) = (1-t)^3 p_1 + 3(1-t)^2 t p_2 + 3(1-t) t^2 p_3 + t^3 p_4 \]

Bernstein Polynomials (1911)

- The general form for the \( i \)-th Bernstein polynomial for a degree \( k \) Bézier curve is
  \[ b_{ik}(u) = \binom{k}{i} (1-u)^{k-i} u^i \]
- Some properties of BPs
  - Invariant under transformations
  - Form a partition of unity, i.e. summing to 1
  - Low degree BPs can be written as high degree BPs
  - BP derivatives are linear combo of BPs
  - Form a basis for space of polynomials w/ deg \( k \)
Bézier Curves and the Bernstein Polynomials

- The four cubic Bernstein polynomials
- \( B_B = M_B \cdot T \)

Observe:
- at \( t=0 \), only \( B_{B_1} \) is >0
  - curve interpolates \( P_1 \)
- at \( t=1 \), only \( B_{B_4} \) is >0
  - curve interpolates \( P_4 \)

General Form of Bézier Curve

\[
Q(u) = \sum_{i=0}^{k} P_{i+1} \binom{k}{i} (1 - u)^{k-i} u^i
\]

Control points: \( P_0, P_1, \ldots, P_{k+1}; \quad 0 \leq u \leq 1 \)

Produces a point on curve \( Q \) at parameter value \( u \)

Properties of Bézier Curves

- \( k+1 \) control points defines a single curve of degree \( k \)
- Affine invariance
- Invariance under affine parameter transformations
- Convex hull property
  - curve lies completely within convex hull of control points
- Endpoint interpolation
- Intuitive for design
  - curve mimics the control polygon

Issues with Bézier Curves

- Creating complex curves requires many control points
  - potentially a very high-degree polynomial with many wiggles
- Bézier blending functions have global support over the whole curve
  - move just one point, change whole curve
- Improved Idea: link \( C^1 \) lots of low degree (cubic) Bézier curves end-to-end

Programming Assignment 1

- Process command-line arguments
- Read in 3D control points
- Iterate through parameter space by \( du \)
  - for loop should use integers!
- At each \( u \) value evaluate Bézier curve formula to produce a sequence of 3D points
- Output points by printing them to standard out as a polyline and control points as spheres in Open Inventor