CMSC 427: Chapter 15
Modeling

Reading: Chapts 15 and 16 in Shirley.
Overview:
- Constructive solid geometry (CSG)
- Polyhedral models
- Implicit surfaces: quadrics, superquadrics, blobby models.
- Parametric surfaces: Bézier curves and surfaces, B-splines, NURBS.

Modeling

Goals of Modeling:
Represent 3-d objects efficiently for easy design, visualization, and modification.

No limit to what can be modeled:
Natural objects: Trees, flowers, rocks, water, fire, smoke, clouds.
Human and animals: Skeletal structure, skin, hair, facial expressions.
Architecture: Walls, doors, windows, furniture, pipes, railing.
Non-geometric Elements: Lighting, textures, surface materials.

Geometric Shape Representations:
Volumetric Representations: Constructive solid geometry (CSG).
Boundary representations: explicit, implicit, and parametric.
Procedural, particle systems, and physically-based models.
Overview

- Constructive solid geometry (CSG)
- Polyhedral models
- Implicit surfaces: quadrics, superquadrics, blobby models
- Parametric surfaces

Constructive Solid Geometry

Constructive Solid Geometry (CSG):
- Used originally for the design of manufactured objects that are created by machining operations: boring, drilling, milling.
- Defines complex 3-d objects in terms of primitive objects: rectangular blocks, slabs, cylinders, spheres, and cones.
- Primitives can be transformed: translation, scaling, rotation.
- Primitives can be combined through set operations: union, intersection, difference.

Rendering with CSG:
- In order to render, first convert to a boundary representation (B-Rep).
- A powerful method for design but B-Rep conversion to B-Rep is costly.

Image source: Scientific Instrument Services, Inc.
Constructive Solid Geometry

**CSG Example:**

![Diagram showing construction of a final object from block-1, block-2, cylinder-1, and a difference operation]

Overview

- Constructive solid geometry (CSG)
- Polyhedral models
- Implicit surfaces: quadrics, superquadrics, blobby models
- Parametric surfaces
Boundary Representations (B-Reps)

**Boundary Representations:** Also known as B-Reps, represent 3-dimensional objects by their 2-dimensional surface boundaries.

**Polyhedral Models:** Represent an object's surface as a collection of flat polygonal faces.

**Curved Surface Models:** (to be discussed further below)
- **Explicit:** \( z = f(x, y) \).
- **Implicit:** \( f(x, y, z) = 0 \).
- **Parametric:** \( p(u, v) = (x(u, v), y(u, v), z(u, v)) \).

**Which is best?** As we shall see, each method has its strengths and weaknesses.

---

**Polyhedral Models**

**Polyhedral Models:** Represent the boundary as a mesh of flat polygonal faces.

**Elements:** vertices, edges, faces.

**Topology:** In order to reason about the object, it is desirable to store information about how these elements are connected together.

**Example:** Each...
- **Vertex:** Points to one of its incident edges.
- **Face:** Points to one of its incident edges.
- **Edge:** Points to:
  - its two incident faces,
  - its two incident vertices, and
  - the next edges in CW and/or CCW order about each of its two incident faces.

Image source: Ohio Supercomputing Center
**Euler-Poincaré Formula**

**Euler-Poincaré Formula:** The mathematician Leonard Euler proved that the numbers vertices \(v\), edges \(e\), and faces \(f\) are related to each other.

\[ v - e + f = 2 - 2g, \]

where \(g\) denotes the number of handles of the surface, called its *genus*. (This was generalized by Poincaré to higher dimensions.)

**Example:**

- \(v = 7\)
- \(e = 9\)
- \(f = 4\) (includes the external face)
- \(g = 0\) (a planar object)

\[ 7 - 9 + 4 = 2 - 2 \cdot 0 \]

**Polyhedral Models: Issues**

**Polyhedral Rendering:**

- Complex objects are represented as a large mesh of simple polygons (typically triangles or quadrangles).
- These meshes are rendered as a number of OpenGL triangle strips or quad strips.

**Triangles are easy to render:**

- Edges are straight (linear).
- Interiors are flat: allows fast incremental scan-conversion.
- Triangle visibility is constant (no self-occlusions).
- Normals can be used to convey flat or smooth shading.
Polyhedral Models: Issues

Strengths of Polyhedral Models:
- **Flexible**: Can model arbitrarily complex surfaces.
- **Highly Local**: Local changes can be made without affecting the rest of the model.

Weaknesses of polyhedral models:
- **Space inefficient**: Curved geometries require lots of triangles.
- **Limited Resolution**: Unlike exact representations, zooming in closer results in loss of smoothness.
- **Highly local**: Global shape design and shape modification are complex since influence is completely local.
- **Complexity of basic operations**: For example, inside/outside tests are difficult for non-convex objects.

Image Source: BOSS International, Inc.

Overview

- Constructive solid geometry (CSG)
- Polyhedral models
- **Implicit surfaces**: quadrics, superquadrics, blobby models
- Parametric surfaces
**Explicit and Implicit Surface Representations**

**Explicit Representation:**
- Represents one coordinate (say, the z-coordinate) as a function of the other two (say, x and y).
- **Example:** The upper half of a unit sphere:
  \[ z = \sqrt{x^2 + y^2} \]
- Very limited, since only one z value for any (x, y) pair.

**Implicit Representation:**
- Expresses a surface as the set of points that satisfy an equation:
  \[ f(x, y, z) = 0 \]
- Subdivides space into inside/outside regions based on whether
  \[ f(x, y, z) < 0 \text{ or } > 0 \]. These regions need not be connected.
- **Isosurface:** The set of points for which \( f(x, y, z) \) has the same fixed value. (Need not be zero.)

**Implicit Surface Representation**

\[ f(x, y, z) = 0 \text{ (isosurface)} \]

\[ f(x) < 0 \]

\[ f(x) > 0 \]
Implicit Surfaces: Issues

Advantages:

- **Smooth**: Surfaces are smooth and (depending on representation) easy to compute derivatives.
- **Membership**: Each to determine whether a point lies inside or outside by simply evaluating the function.
- **Full resolution**: Since representation is exact, can zoom in arbitrarily close.

Disadvantages:

- **Hard to control**: Very difficult to determine the function that produces a desired general shape. (However, see discussion of blobs and metaballs below for shape approximation.)
- **Hard to render**: Before rendering (using, say, OpenGL) need to convert to a mesh of polygon surface meshes.

Quadrics

**Quadric**: An implicit surface defined by an algebraic equation of degree 2. (Recall that the degree is maximum of the sums of the variable exponents in each term.)

**Examples**: (see next slide)

- \( x^2 + y^2 + z^2 - 1 = 0 \) Sphere/Ellipsoid
- \( x^2 + y^2 - z^2 - 1 = 0 \) Hyperboloid of one sheet
- \( x^2 - y^2 - z^2 - 1 = 0 \) Hyperboloid of two sheets
- \( x^2 + y^2 - z^2 = 0 \) Elliptic cone
- \( x^2 + y^2 - z = 0 \) Elliptic paraboloid
- \( -x^2 + y^2 - z = 0 \) Hyperbolic paraboloid

In general, \( x, y, z \) above can be replaced with \( x/a, y/b, z/c \) for arbitrary positive constants \( a, b, \) and \( c \).
Quadrics

**Ellipsoid**
\[(x/a)^2 + (y/b)^2 + (z/c)^2 = 1\]

**Hyperboloid**
\[(x/a)^2 + (y/b)^2 - (z/c)^2 = 1\]

\[-(x/a)^2 - (y/b)^2 + (z/c)^2 = 1\]

**Elliptic Cone**
\[(x/a)^2 + (y/b)^2 - (z/c)^2 = 0\]

**Elliptic Paraboloid**
\[(x/a)^2 + (y/b)^2 - (z/c)^2 = 0\]

**Hyperbolic Paraboloid**
\[-(x/a)^2 - (y/b)^2 - (z/c)^2 = 0\]

---

Quadrics - Representation

**Quadric General Form:**  
\[ax^2 + by^2 + cz^2 + 2fyz + 2gzx + 2hxy + 2px + 2qy + 2rz + d = 0.\]

**Matrix Representation:** Although this looks messy, there is a simple representation in *matrix form*, which is easy to verify is equivalent to the above.
- Let \(p = [x, y, z, 1]^T\) be the homogeneous column vector for a point.
- We can express the above equation as \(p^TQp = 0\), using the following symmetric 4x4 matrix \(Q\),

\[
\begin{bmatrix}
    a & h & g & p \\
    h & b & f & q \\
    g & f & c & r \\
    p & q & r & d
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    z \\
    1
\end{bmatrix} = 0
\]
Superquadrics

Superquadrics:
  - A generalization of quadrics developed by Alan Barr, 1981.
  - Enhance flexibility by introducing two additional parameters, s and t, which modify the quadric exponent.
  - Example:
    - Quadric Ellipsoid: \((x/a)^2 + (y/b)^2 + (z/c)^2 = 1\)
    - Superquadric Ellipsoid: \([ (x/a)^{2s} + (y/b)^{2t} ]^{1/t} + (z/c)^{2/t} = 1\)
   Observe that when \(s = t = 1\), the superquadric and the quadric are identical.
  - Similarly one can define superquadric hyperboloids, superquadric paraboloids, and so on.

Examples of superquadric ellipsoids.

Image Courtesy Montiel, Aguado, Zaluska, University of Surrey, UK
Blobs and Metaballs

Blobs:
- Another type of implicit function surface model.
- Also known as metaballs, blobby models, soft objects.
- Useful for modeling objects made up of soft rounded parts, such as muscles for humans and animals.
- Based on combining many simple implicit functions into one complex implicit function.

Intuition:
- Consider a point \( p = (p_x, p_y, p_z) \), and any function \( f \) that decreases as a function of the distance \( r \) of point \( (x, y, z) \) from \( p \).
- Let:
  \[
  r = \sqrt{(x-p_x)^2 + (y-p_y)^2 + (z-p_z)^2}
  \]

Examples: (let \( a \) and \( b \) be parameters selected by the user)

\[
\begin{align*}
  f(x,y,z) &= \frac{a}{r^2} \\
  f(x,y,z) &= e^{-a r^2} \\
  f(x,y,z) &= \begin{cases} 
    a(1 - 3(r/b)^2) & 0 \leq r \leq b/3 \\
    (3/2)a(1-r/b)^2 & b/3 \leq r \leq b \\
    0 & b \leq r
  \end{cases}
\end{align*}
\]

The choice of \( f \) affects blob shape.

Image source: http://www.programmersheaven.com/2/Fast-metaballs
Blobs and Metaballs

Intuition:
- Given number of points, sum of these functions for each such point. Like a density or potential function.
- Next, consider isosurfaces (like contour lines in a terrain map) that arise at various values of the density function.

Intuition:
- Selecting an arbitrary threshold value $T$ uniquely determines one of these isosurfaces. The result is our blobby object.
- Final implicit function: $f(x, y, z) = \sum c_k f_k(r) - T = 0$, where $f_k$ is the density function for $p_k$, and $c_k$ is an additional parameter.
Blobs and Metaballs

Example:

Muscular structure was created using Metareyes plug-in and hundreds of metaballs.

Image source: Ryan Geiss, www.geisswerks.com

Image courtesy Spencer Arts
Creating Implicit Models from Scanned Data

3-d range scanners produce point data, from which a blobby model can be generated.

Points with normals (typical output of range scanners).

\[ f(x,y,z) > 0 \text{ inside} \]
\[ f(x,y,z) < 0 \text{ outside} \]
\[ f(x,y,z) = 0 \]

Overview

- Constructive solid geometry (CSG)
- Polyhedral models
- Implicit surfaces: quadrics, superquadrics, blobby models
- Parametric surfaces
Parametric Surface Models

**Parametric Model:**
- Represents coordinates as functions of parameters.
- Parametric curve:
  \[ x(u) = \ldots \quad y(u) = \ldots \quad z(u) = \ldots \quad \text{where } u_0 \leq u \leq u_1. \]
- Parametric surface:
  \[ x(u, v) = \ldots \quad y(u, v) = \ldots \quad z(u, v) = \ldots \quad \text{where } u_0 \leq u \leq u_1, \quad v_0 \leq v \leq v_1. \]

**Example:**
- Recall that a sphere of radius \( r \) centered at the origin can be expressed parametrically in spherical coordinates:
  \[ z(\theta, \varphi) = r \cdot \cos \varphi \]
  \[ x(\theta, \varphi) = r \cdot \cos \theta \sin \varphi \]
  \[ y(\theta, \varphi) = r \cdot \sin \theta \sin \varphi \]
  where \( 0 \leq \theta \leq 2\pi \) and \( 0 \leq \varphi \leq \pi \).

**Exercise:** How would you generalize this to an ellipse?

---

**Advantages:**
- Easy to enumerate points on the curve/surface, by enumerating \( u \) and \( v \) values and evaluating.
- Easy to render: Just subdivide the surface into quadrilateral or triangular patches, which can be passed to OpenGL.

**Disadvantages:**
- Representing complex objects requires putting many patches together. Guaranteeing smoothness at patch boundaries can be tricky.
- Sidedness test is harder: We can test whether a point lies inside or outside an implicit surface by testing the sign of \( f(x, y, z) \). Harder for parametric surfaces.
Parametric Continuity

Continuity:
- When two parametric surface patches are joined to make a larger surface, and important issue is whether they connect smoothly at the joint.

Parametric Continuity:
- A curve is $C^k$ continuous at a joining point $p$ if the first $k$ derivatives (in $u$) of the two functions exist and are equal at $p$.

- This can be generalized to surfaces using partial derivatives.
- Note that parametric continuity depends on the way the curve is parameterized, and is not a pure geometric property.

Interpolation vs. Approximation

Control Points:
- Parametric curves and surfaces are often defined in terms of a set of control points. User's can adjust shape by moving the points.

Interpolation:
- The curve/surface passes through the control points.

Approximation:
- The curve/surface passes close to the control points.

Issues:
- Although interpolation would seem the better approach from a user's perspective (more exact surface control), the requirement of interpolation can result in greater variation (wiggling) in the curve.
Parametric Polynomial Curves

Parametric Polynomial Curve: “Nicest” parametric representation.
- The parametric functions are polynomials of degree $d$ in the parameters.
  \[ x(u) = \sum_{i=0}^{d} a_i u^i \quad y(u) = \sum_{i=0}^{d} b_i u^i \]

Example: $x(u) = 7u^3 + 3u^2 - 4u + 6$, $y(u) = u^3 - 4u^2 + 6$ is a parametric polynomial curve of degree 3, a cubic parametric polynomial.

Why polynomials are nice: They are easy to evaluate and easy to differentiate.

Horner’s Rule: Can be computed efficiently with $d$ multiplications and $d$ additions/subtractions: $x(u) = (((7u + 3)u - 4)u + 6)$.

Matrix Representation: A convenient symbolic form.
Can express $x(u) = 7u^3 + 3u^2 - 4u + 6$ as: $x(u) = \begin{bmatrix} 7 \\ 3 \\ 4 \\ 6 \end{bmatrix}$

Bézier Curves

Bézier Curves:
- Developed in the 1960’s by Bézier at Renault and de Casteljau at Citroën.
- Given $n$ control points, this is a parametric polynomial curve of degree $n-1$.
- An elegant method for defining a curve that is defined by a sequence of control points.
- Has a number of nice geometric properties.
- Generalizes readily to surfaces.
Iterated Interpolation: The basic idea behind Bézier curves.
- Given two points \( p_0 \) and \( p_1 \), how can we define a parametric curve \( p(u) \) between them?
- Easy. Just use linear interpolation:
  \[
p(u) = (1-u)p_0 + up_1 \quad \text{for } 0 \leq u \leq 1.
  \]
- Each point on this line segment is the convex combination (weighted average) of the two control points.

Taking this one step further:
- Given three points \( p_0, p_1, \) and \( p_2 \), how can we define a parametric curve \( p(u) \) between them?
- de Casteljau’s idea:
  - Interpolate between \( p_0 \) and \( p_1 \). Call the result \( p_{01}(u) \).
  - Interpolate between \( p_1 \) and \( p_2 \). Call the result \( p_{12}(u) \).
  
  \[
  p_{01}(u) = (1-u)p_0 + up_1 \\
  p_{12}(u) = (1-u)p_1 + up_2
  \]
  - Finally, interpolate between \( p_{01}(u) \) and \( p_{12}(u) \).
  
  \[
  p(u) = (1-u)p_{01}(u) + up_{12}(u) \\
  = (1-u)((1-u)p_0 + up_1) + u((1-u)p_1 + up_2) \\
  = (1-u)^2p_0 + 2(1-u)up_1 + u^2p_2
  \]
Blending:
- The result is the Bézier curve of degree 2:
  \[ p(u) = (1-u)^2 p_0 + 2(1-u)u p_1 + u^2 p_2. \]
- We can express \( p(u) \) as a parametric blending of the control points \( p_0, p_1, \) and \( p_2, \)
  \[ p(u) = b_{02}(u) p_0 + b_{12}(u) p_1 + b_{22}(u) p_2. \]
  by the Bézier blending functions (of degree 2):
  \[ b_{02}(u) = (1-u)^2, \quad b_{12}(u) = 2(1-u)u, \quad b_{22}(u) = u^2. \]

Convex Hull Property:
- Observe that the blending functions sum to 1 for any \( u \) and are nonnegative if \( 0 \leq u \leq 1. \) That is, \( p(u) \) is a convex combination of the control points.
- A Bézier curve lies within the convex hull of its control points.

Generalizing: to 4 control points we have
- The Bézier curve of degree 3:
  \[ p(u) = (1-u)^3 p_0 + 3(1-u)^2 u p_1 + 3(1-u)u^2 p_2 + u^3 p_3. \]
- We can express \( p(u) \) as a parametric blending of the control points.
  \[ p(u) = b_{03}(u) p_0 + b_{13}(u) p_1 + b_{23}(u) p_2 + b_{33}(u) p_3. \]
  by the Bézier blending functions (of degree 3):
  \[ b_{03}(u) = (1-u)^3, \quad b_{13}(u) = 3(1-u)^2 u, \quad b_{23}(u) = 3(1-u)u^2, \quad b_{33}(u) = u^3. \]
**Bézier Curves: Four Control Points**

Bézier Curve for 4 control points $P_0, P_1, P_2, P_3$.

Bézier blending functions of degree 3.

$P(u) = \sum_{j=0}^{d} b_{jd}(u) p_j$

where $b_{jd}(u)$ is the $j^{th}$ Bézier blending function of degree $d$.

**Bézier Curves: The General Case**

**Generalization:**
- A Bézier curve of degree $d$ over $d+1$ control points $p_0, p_1, \ldots, p_d$ is defined as follows:

$P(u) = \sum_{j=0}^{d} b_{jd}(u) p_j$

where $b_{jd}(u)$ is the $j^{th}$ Bézier blending function of degree $d$.

**Generalizing the Blending Functions: The Berstein Polynomials**

**Degree 0:** $b_{00}(u) = 1$

**Degree 1:** $b_{01}(u) = 1 - u$, $b_{11}(u) = u$

**Degree 2:** $b_{02}(u) = (1-u)^2$, $b_{12}(u) = 2(1-u)u$, $b_{22}(u) = u^2$

**Degree 3:** $b_{03}(u) = (1-u)^3$, $b_{13}(u) = 3(1-u)^2u$, $b_{23}(u) = 3(1-u)u^2$, $b_{33}(u) = u^3$

... (the coefficients follow Pascal's triangle)

**Degree $d$:** $b_{jd}(u) = \binom{d}{j} (1-u)^{d-j} u^j$ where $\binom{d}{j} = \frac{d!}{j!(d-j)!}$

Image source: http://www.ursoswald.ch

Image source: MIT
Bézier Curves: Examples

Examples with:
(a) 3 points
(b) 4 points
(c) 4 points
(d) 4 points
(e) 5 points

Image source: Hearn and Baker

Bézier Curves: Properties

Nice Properties of Bézier curves:
Convex Hull Property: A Bézier curve lies within the convex hull of its control points.
Starting and Ending: A Bézier curve starts and ends at the first and last control points $p_0$ and $p_d$.
Tangencies at Endpoint: A Bézier curve is tangent with the first segment $p_0p_1$ at $u = 0$, and it is tangent with its last segment $p_{d-1}p_d$ at $u = 1$.
Variation Diminishing: If a line intersects the control polygon k times, then it intersects the Bézier curve no more than k times. Thus, the Bézier curve cannot be more "wiggly" than the control polygon.
Bézier Curves: Examples

Joining Bézier Curves:
Align the two tangents at the joining point (that is, the last and first edges of the control polygon) to achieve at least \( C^1 \) (slope) continuity.

Example: Two curves are joined at \( p_0' = p_2 \). The control points \( p_1, p_2, \) and \( p_1' \) are collinear.

From Curves to Surfaces

We know how to define a Bézier curve. Can we generalize this to generate Bézier surfaces?

Tensor Product Construction: (of a Bézier surfaces of degree \( d \))
- We have two parameters \( u \) and \( v \).
- Rather than \( d+1 \) control points, let us assume that we have a \((d+1) \times (d+1)\) mesh of control points, \( p_{ij} \), \( 0 \leq i, j \leq d \).
- For each fixed \( i \), we can define the Bézier curve for \( p_0, p_1, \ldots, p_d \), the result is
  \[
P_i(u) = \sum_{0 \leq j \leq d} b_j(u)p_{ij}
  \]
- We can then apply the blending process to \( p_0(u), p_1(u), \ldots, p_d(u) \) using the parameter \( v \). The result is a Bézier surface:

\[
P(u,v) = \sum_{0 \leq i \leq d} b_i(v)p_i(u) = \sum_{0 \leq i \leq d} \sum_{0 \leq j \leq d} b_i(v)b_j(u)p_{ij}
\]
Bézier Surfaces

Joining Bézier Surfaces:

Align the two tangents at the joining points (that is, the rightmost and leftmost edges of the seam where the control polygon meet) to achieve at least $C^1$ slope continuity.

B-Splines

Shortcomings of Bézier Curves/Surfaces:

High Degree: To fit many points, the polynomial degree must be very high. Joining curves of low degree requires effort from the designer to maintain tangency at joints.

Global Support: Every control point affects the entire curve. Harder for a designer to alter the curve locally.

B-Splines: remedy this problem

Similar structure: Like Bézier curves and surfaces, B-Spline curves and surfaces are based on control points and blending functions.

Local Control: Each blending function is non-zero over a small interval of parameter space. Each control point only affects a local region of the curve or surface.
**B-Splines**

**B-Spline Blending Functions:**
- Each blending function is non-zero over a small interval.
- What sort of function does this?

As with Bézier curves, this is done by repeated blending, but the functions are based on a more complex recursive formula, called the Cox-de Boor formulas (which we will skip).

**Example**: Quadratic B-spline blending function

\[
B_{2,2}(u) = \begin{cases} 
\frac{1}{2}u^2 & \text{for } 0 \leq u < 1 \\
\frac{1}{2}u(2-u) + \frac{1}{2}(u-1)(3-u) & \text{for } 1 \leq u < 2 \\
\frac{1}{2}(3-u)^2 & \text{for } 2 \leq u < 3
\end{cases}
\]

Image source: Hearn and Baker
B-Spline Generalizations

Uniform/Non-uniform B-Splines:
- The set of knot values is collectively known as the knot vector.
- If the knot values are equi-spaced (in parameter space), the curve is a uniform B-spline.
- Otherwise it is a non-uniform B-spline.
- Varying the distribution of knot values causes the curve to "bunch up" near control points where the knot values are closer together, thus increases their influence.
- This provides the designer with greater flexibility.

B-Spline Surfaces:
- The same tensor-product construction can be used to generate B-spline surfaces.
  - 2-dimensional mesh of control points.
  - First blend among columns \((u)\), then blend among rows \((v)\).

Rational Functions:
- A rational function is the ratio of two polynomials \(P(u)/Q(u)\).
- Rational functions provide greater flexibility over polynomials.
- For example, there is no polynomial parametric curve for a simple circular arc (\(\)):
  - \((\cos \theta, \sin \theta)\) is not a polynomial.
  - \((x, \sqrt{(1-x^2)})\) is not a polynomial.
- ...but a circle can be parameterized as a rational function:
  - \((x(u), y(u)) = ((1-u^2)/(1+u^2), 2u/(1+u^2))\) is a rational function.
To see this, verify that \(x(u)^2 + y(u)^2 = 1\).
**B-Spline Generalizations: NURBS**

Rationals seem to be more complex:
- Evaluation, derivatives, scan-conversion are more complex.

**Trick for creating rational functions from polynomials:**
- Express the previous circle parametric representation using **homogeneous coordinates** we have:
  \[ x(u), y(u), w(u) = [(1-u^2), 2u, (1+u^2)] \]
  This is a polynomial. (!)

**Non-Uniform Rational B-Splines (NURBS):**
- Express your 3-d control points in **homogeneous coordinates**, and compute a (4-dimensional) **non-uniform B-spline**.
- After applying **perspective normalization**, you have a rational function in dimension 3.
- Perspective normalization comes **practically for free** on modern graphics systems.
- NURBS are **widely used** in the CAD industry.

Chapter 15, Slide 51
Copyright © D. M. Mount and A. Varshney

---

**NURBS**

Images source www.rhino3d.com:
Arc de Triomphe is by Mariani Louis
Jaguar is by Alexey Popov

Chapter 15, Slide 52
Copyright © D. M. Mount and A. Varshney
Summary

Summary:
- Constructive solid geometry (CSG)
- Polyhedral models
- Implicit surfaces: quadrics, superquadrics, blobby models.
- Parametric surfaces: Bézier curves and surfaces, B-splines, NURBS.