1996-11-27 Bo Nordin (bosse@cb.uu.se)

Lecture notes, computer graphics (F13) (Foley 9.1, 9.2)

Curves introduction

Representation of curves and surfaces
Curves WHY?
Complex-shaped objects occur in the real world.
They can be modelled as plane-faced polyhedra, wireframes
Problems: bulky, intractable, "unreal"

Man-made objects are well suited to modelling with mathematical curves/surfaces (CAD technique even allows the user to "create" objects from models).
Characters, data plots, CG art, virtual reality

Natural objects often need other modelling techniques -> fractals

Not only must the representation be tractable to computers, but also to the user (at least a mapping from computer store to user interface must be possible).

Two ways to introduce curves: *either* from circles, ellipses *or* from other 3D representations
I have already talked about circles

Let's introduce curves/surfaces from other 3D object representations

Surface modeling (today) vs. *solid modeling*

Polygon meshes (9.1)

Boundary representations of 3D objects: set of surface polygons (speeds up rendering, everything is described by linear equations).
A polygon representation for a polyhedron precisely defines the surface,
but for other objects, surfaces are *tessellated* (*tiled*) to produce a *polygon-mesh* approx.

triangle strips, *quadri-lateral mesh*
triangulation - the process of transforming a polygon to triangles.
triangles always lie in one plane!!

Polygon mesh is a collection of edges, vertices, and polygons connected such that each edge is shared by at most two polygons. Polygons are closed
Vertices usually assumed enumerated counter-clockwise -> define inside/outside

Representation
explicit: P = ((x_1,y_1,z_1), ... (x_n,y_n,z_n)) - efficient for a single polygon
written twice, no notion of which edges are the same, not space efficient

pointers to a vertex list (SPHIGS) each vertex stored once, a polygon is defined by a list of indices (+vertices stored once, -still hard to find polygons that share an edge
V = (V_1..V_N) P_1=(1,2,4) P_2 = (4,2,3)

pointers to an edge list: vertex list V and edge list E, V=(V_1...V_4), E_1=(V_1,V_2,P_1,P_2)
P_1 = (E_1, E_4, E_5)

Plane equations:
We frequently need to know the plane equation. (sometimes known as a result of the construction). Otherwise recall $Ax + By + Cz + D = 0$ (A,B, and C define the normal to the plane)

We often need to distinguish between the two sides of the surface that enclose the object. If vertices specified in counter-clockwise direction when viewing the outer side of the plane in a right-handed coordinate system, the dir of the normal vector is from inside to outside.

Foley pg. 326-327 gives an algorithm for calculation of the plane equation coeff's given n vertices of a polygon that is approximately planar.

Another strategy is to use 3 vertices (another reason why triangles are ok):
$A = y_1(z_2-z_3) + y_2(z_3-z_1) + y_3(z_1-z_2)$
$B = z_1(x_2-x_3) + z_2(x_3-x_1) + z_3(x_1-x_2)$
$C = x_1(y_2-y_3) + x_2(y_3-y_1) + x_3(y_1-y_2)$
$D = -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1-y_1z_3) - x_3(y_1z_2 - y_2z_1)$

Normal vector is (A,B,C)

Or: $\mathbf{N} = (\mathbf{V}_2-\mathbf{V}_1)*(\mathbf{V}_3-\mathbf{V}_1)$ (* is the *cross-product* (see Foley 5.1.6), not the dot product)

Curves (9.2)

OK, this is not surface modeling *but* surfaces are generalizaitons of curves
Polylines, polygons are 1st-degree, piecewise approximations (large # of coords, interactive-manipulations hard).
Explicit functions: y = f(x), z = g(x) BAD! 1/ no multiple y for one x (no circles, ellipses)
2/ not rotationally invariant, 3/ describing curves with vertical tangents (infinite slope)
Implicit function: f(x,y,z) = 0 ex: $x^2 + y^2 = 1$ constraints for half-circles (>=0) hard to join

parametric representation: x = x(t), y = y(t), z = z(t)
piecewise polynomial curve <-- piecewise linear
cubic polynomials common - lower, too inflexible, higher - too flexible
We save the curve as a # of control points
Interpolation goes through the control points
Approximation: does not have to go through the control points

$T = [t^3\ t^2\ t\ 1]$ $Q(t) = [x(t)\ y\{t\}\ z(t)]^T = C * T$
where x(t),for instance, is $x(t) = a_xt^3 + b_xt^2 + c_xt + d_x$ ($t < [0,1]$) (x,y,z are independent!)
We want a smooth transition:
Continuity between curve segments (parametric, i.e. in t):
joined - C^0 (zero-order parametric continuity) (DRAW examples Foley 331,, B&H: 319
derivative equal - C^1 (first-order parametric continuity)
2nd derivative equal - C^2 (second-order parametric continuity)
geometric continuity conditions
zero-order geometric == zero-order parametric
first-order geometric == derivatives proportional

A curve segment Q(t) is defined by constraints in endpoints, tangent vectors, and continuity.
Each cubic polynomial has four unknowns, so we need four constraints

Hermite: 2 endpoints and 2 tangent vectors
Bezier: 2 endpoints and two points that control the endpoint tangent vectors
B-spline: four control points

rewrite C = G * M where M is a 4x4 *basis matrix* and G is 4-element matrix of constraints, called the *geometry matrix*.

$$Q(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} g_{1x} & g_{2x} & g_{3x} & g_{4x} \\ g_{1y} & g_{2y} & g_{3y} & g_{4y} \\ g_{1z} & g_{2z} & g_{3z} & g_{4z} \end{pmatrix} \begin{pmatrix} m_{11} & m_{21} & m_{31} & m_{41} \\ m_{12} & m_{22} & m_{32} & m_{42} \\ m_{13} & m_{23} & m_{33} & m_{43} \\ m_{14} & m_{24} & m_{34} & m_{44} \end{pmatrix} \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

The curve is a weighted sum of the elements of the geometry matrix
The *blending functions* B are given by B = M * T.
Blending functions should sum to unity.
Note the similarity to a piecewise linear approximation (only 2 then):

$x(t) = g_{1x}(1-t) + g_{2x}(t)$
$y(t) = g_{1y}(1-t) + g_{2y}(t)$
$z(t) = g_{1z}(1-t) + g_{2z}(t)$

When viewed this way parametric cubics are just a generalization of straight-line approximations.
Hermite curves:

Interpolating (two points P_1 and P_4, and two tangent vectors, R_1 and R_4)
Finding the Hermite basis matrix M_H and the Hermite geometry vector G_H
For x:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x = C_x * T = G_{Hx} * M_H * T = G_{Hx} * M_H [\, t^3\ t^2\ t\ 1]^T$$

$$x(0) = P_{1x} = G_{Hx} * M_H [0\ 0\ 0\ 1]^T \quad x(1) = P_{4x} = G_{Hx} * M_H [\, 1\ 1\ 1\ 1]^T$$

$$x'(t) = G_{Hx} * M_H [\, 3t^2\ 2t\ 1\ 0]^T \text{ and thus}$$

$$x'(0) = R_{1x} = G_{Hx} * M_H [\, 0\ 0\ 1\ 0]^T \text{ and } x'(1) = R_{4x} = G_{Hx} * M_H [\, 3\ 2\ 1\ 0]^T$$

or in matrix form:

$$[P_{1x}\ P_{4x}\ R_{1x}\ R_{4x}] = G_{Hx} = G_{Hx} * M_H * \begin{pmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

$$M_H = \begin{pmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix}$$ (inverse of the matrix)

Use in $x(t) = G_{Hx} * M_H * T$ to find x(t) based on the geometry vector
Similarly for y(t) and z(t):

$$Q(t) = [\, x(t)\ y(t)\ z(t)\,]^T = G_H * M_H * T \quad (G_H \text{ is } [\, P1\ P4\ R1\ R4])$$

Expanding the product:

$$Q(t) = G_H * M_H * T = G_H * B_H =$$

$$(2t^3 - 3t^2 + 1)\,P_1 + (-2t^3 + 3t^2)P_4 + (t^3 - 2t^2 + t)R_1 + (t^3 - t^2)R_4$$

Blending functions: at t =0 only P_1 is nonzero and at t = 1 only P_4 is nonzero

Figure 9.11 - 9.13 page 334-335

Drawing curves:
Evaluate the equations at n successive values of t separated by a step size.
Horner's scheme $f(t) = at^3 + bt^2 + ct + d = ((at+b)t+c)t+d$

Cubic curves are linear combinations of the four elements of the geom, vector we can transform the geometry vector and then use it to generate the transformed curve.
I.e: the curves are invariant under translation, scaling, and translation.
Not under perspective transformation, though,

Bezier curves:
History: Renault ca 1970
Specifies the endpoints and the endpoint vectors by 2 intermediate points not on the curve the starting and ending vectors are determined by the vectors P_1P_2 and P_3P_4.

$R_1 = Q'(0) = 3(P_2 - P_1)$ and $R_4 = Q'(1) = 3(P_4 - P_3)$

Bezier geometry matrix is $G_B = [\,P_1\ P_2\ P_3\ P_4]$
M_{HB} that defines $G_H = G_B * M_{HB}$

$$GH = [\,P1\ P4\ R1\ R4] = [\,P1\ P2\ P3\ P4] \begin{pmatrix} 1 & 0 & -3 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \end{pmatrix} = G_B * M_{HB}$$

To *Bezier basis matrix* M_B

$$Q(t) = G_H * M_H * T = (G_B * M_{HB}) * M_H * T = G_B(M_{HB} * M_H) * T = G_B * M_B * T$$

$$M_B = M_{HB} * M_H = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

and $Q(t) = G_B * M_B * T = (1-t)^3P_1 + 3t(1-t)^2P_2 + 3t^2(1-t)P_3 + t^3P_4$

Figure 9.7 pg 338 show joining of 2 curve segments $(P_3 - P_4) = k(P_4 - P_5)$, $k > 0$

Sum to unity and everywhere non-negative -> Q(t) is a weighted average ->
Each curve segment is contained in the convex hull of the four control points.
Can be used for clipping, for instance

Uniform non-rational B-Splines
Ferguson, Boeing ca 1960
Spline (ducks et al)
Consists of curve segments whose polynomial coeff's depend on just a few control points (local control)
We describe the process in terms of all segments at once
Approximate m+1 control points P_0 - P_m (m >= 3) with a curve consisting of m-2 cubic polynomial curve segments $Q_3 ... Q_m$.
If m == 3 there is a single curve segment Q_3 defined on $t_3 <= t < t_4$.

For each i >= 4 there is a *knot point* between Q_{i-1} and Q_i.
Uniform means that the knot are spaced at equal intervals of the t parameter
Each of the m-2 curve segments of a B-spline curve is defined by 4 of the m+1 control pnts.
The geometry matrix:

$$[\ P_{i-3}\ P_{i-2}\ P_{i-1}\ P_i]$$

The first segment Q_3 is defined by P_0 - P_3 over the par range $t_3 = 0$ to $t_4 = 1$.
The 2nd Q_3 defined by P_1 - P_4 over the parameter range $t_4 = 1$, $t_5 = 2$.
In general Qi begins somewhere near P_{i-2} and ends near P_{i-1}.

Blending functions for each curve segment are exactly the same

Multiple control points:
C^0, C^1, and C^2 continuous.
more control
But double control points:
triple control points: Q_i becomes a straight line (C^2 but only G^0)
convex hull (example page 346)
non-uniform
interval between successive knot points are no longer uniform
blending functions differ for each interval
+ may reduce continuity to C^0

multiple knot points (overcomes some of the problems with multiple control points)

Nonuniform rational cubic polynomials:
x(t) = X(t)/W(t), y(t) = Y(t)/W(t), z(t) = Z(t)/W(t)
X(t) -- W(t) are cubic polynomial curves in homogenuous coordinates.
Invariant under perspective transformation (for non-rational curves we must compute each point and the do the perspective projection)

Non-uniform rational B-splines (NURBS) - most general

Quadric surfaces, superquadrics
1/ Described by 2nd-degree equations (spheres, ellipsoids, toruses)
2/ If we add parameters to the equations: superquadrics (ex. superellipse)

Comparison (9.2.8)
- ease of interactive manipulation
- degree of continuity.
- speed of computation

one can convert between representation

table page 350