# B-splines

Andrzej Szymczak

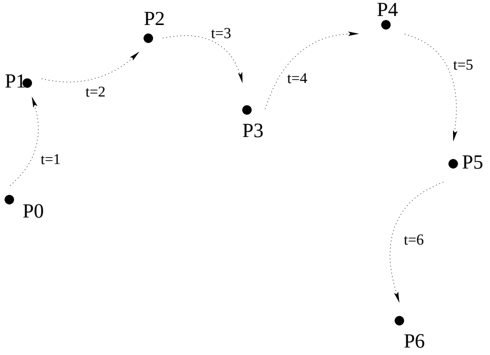
November 1, 2011

# B-splines

- Bezier curves don't have local control property (move a point and the entire curve changes)
- Bezier curves with complicated shape (many control points) have high degree
- Can join lower degree curves into longer ones, but then one has to make sure that the smoothness conditions at the join points are met
- B-splines are smooth and piecewise low degree for sequence of control points of any length
- There are several types of B-splines; here we'll cover uniform polynomial B-splines

# B-splines

- Take a sequence of control points $P_0, P_1, \ldots, P_{n-1}$.
- Zero-degree B-spline is a discontinuous 'curve' that jumps from one control point to the next every 1 unit of time
- $B_0(t) = P_{\lfloor t \rfloor}$



# B-splines

- Zero degree B-spline doesn't really deserve to be called a curve, but can be improved by filtering
- Simple box filter, or averaging over a moving unit time interval, is used to derive uniform B-splines
- Formula for $B_d$:

$$B_{d+1}(t) := \int_t^{t+1} B_d(s)ds$$

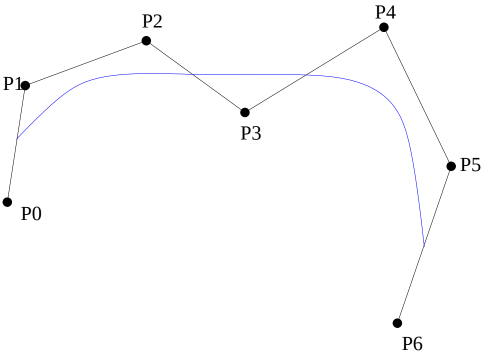
- $B_1$ is the piecewise linear (polygonal) curve connecting the consecutive control points
- By the Fundamental Theorem of Calculus $B_2$ is differentiable and the derivative is

$$B_2'(t) = B_1(t+1) - B_1(t)$$

- $B_2$'s derivative is piecewise linear so $B_2$ is piecewise quadratic

# B-splines

- black: $B_1$
- blue: $B_2$



# B-splines

- $B_d$'s domain: $[0, n-d]$
- $B_d$ is piecewise polynomial of degree $d$
- $B_d$ is $C^{d-1}$, i.e. has $i-1$ continuous derivatives
- Cubic B-spline ($B_3$) is the most popular: it has continuous acceleration (good for physics simulation)

# B-spline basis functions

- A B-spline of degree $d$ is defined by the following formula

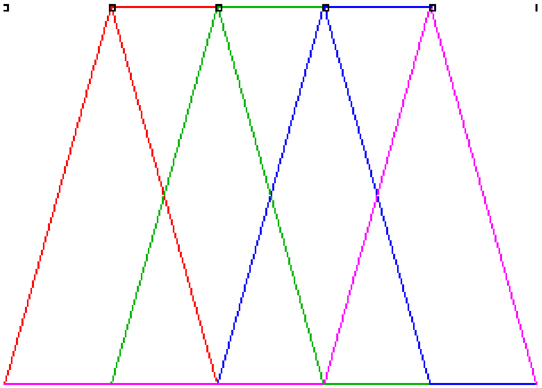
$$B_d(t) = \sum_{i=0}^{n-1} P_i \phi_i^{(d)}(t)$$

- $\phi_i^{(d)}$ for $i \in \{0, 1, \ldots, n-1\}$ are the *B-spline basis function*
- $\phi_i^{(d)}$ can be obtained from $\phi_0^{(d)}$ by shifting by $i$:

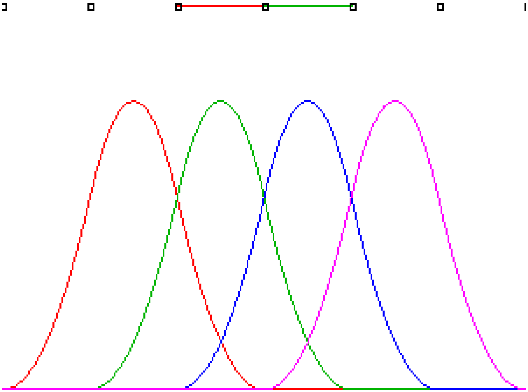
$$\phi_i^{(d)}(t) = \phi_0^{(d)}(t - i).$$

- The basis functions sum to 1 for every $t \in [0, n-d]$ (B-spline curve is a subset of the convex hull of the control points)

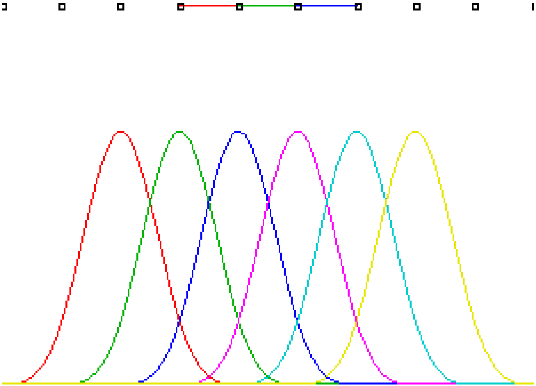
# B-spline basis functions: d=1



# B-spline basis functions: d=2



# B-spline basis functions: d=3



# B-spline subdivision

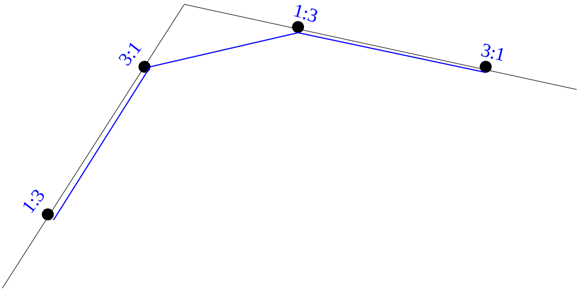
- An easy and fast procedure for drawing B-spline curves
- Based on two elementary operations on sequences of points
  - Doubling: $P_0 P_1 \dots P_{n-1} \longrightarrow P_0 P_0 P_1 P_1 P_2 P_2 \dots P_{n-1} P_{n-1}$
  - Averaging: $P_0 P_1 \dots P_{n-1} \longrightarrow \frac{P_0+P_1}{2}, \frac{P_1+P_2}{2} \dots \frac{P_{n-2}+P_{n-1}}{2}$
- Subdivision step for degree $d$ B-splines: double and do averaging $d$ times
- Drawing a B-spline: apply subdivision a few times starting from the control point sequence; connect the consecutive points in the resulting sequence
- One can also use explicit formulas for B-splines (see a good geometric modeling book), generate a number of points along the curve and connect them with lines

# B-spline subdivision: degree 1

- $P_0 P_1 \ldots P_{n-1} \longrightarrow P_0 P_0 P_1 P_1 \ldots P_{n-1} P_{n-1} \longrightarrow$
  $P_0, \frac{P_0+P_1}{2}, P_1, \frac{P_1+P_2}{2}, \ldots, \frac{P_{n-2}+Pn-1}{2}, P_{n-1}$
- Basically, each subdivision step inserts a midpoint between a consecutive pair of points in the input sequence
- Result: denser and denser set of points along the curve (degree 1 B-spline)

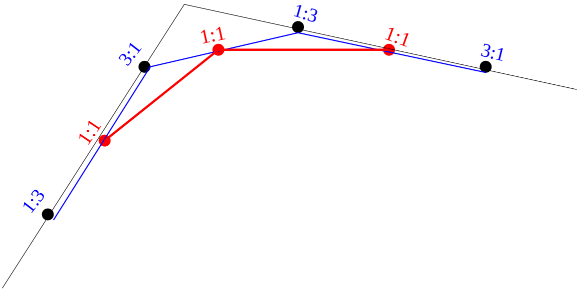
# B-spline subdivision: degree 2

- $P_0 P_1 \ldots P_{n-1} \longrightarrow$
  $\frac{3P_0+P_1}{4}, \frac{P_0+3P_1}{4}, \frac{3P_1+P_2}{4}, \frac{P_1+3P_2}{4}, \ldots, \frac{3P_{n-1}+P_n}{4}, \frac{P_{n-1}+3P_n}{4}$
- Corner cutting (here: as a way to make the curve smoother)
- Result: denser and denser set of points along the curve (degree 2 B-spline)
- Black: input; blue: output (lines connect consecutive points)



# B-spline subdivision: degree 3

- $P_0P_1 \ldots P_{n-1} \longrightarrow$
  $\frac{P_0+P_1}{2}, \frac{P_0+6P_1+P_2}{8}, \frac{P_1+P_2}{2}, \frac{P_1+6P_2+P_3}{8}, \ldots, \frac{P_{n-2}+6P_{n-1}+P_n}{8}, \frac{P_{n-1}+P_n}{2}$
- More aggressive corner cutting (in a way, twice per step)
- Result: denser and denser set of points along the curve (degree 3 B-spline)
- Black: input; red: output (lines connect consecutive points)



# Other subdivision schemes

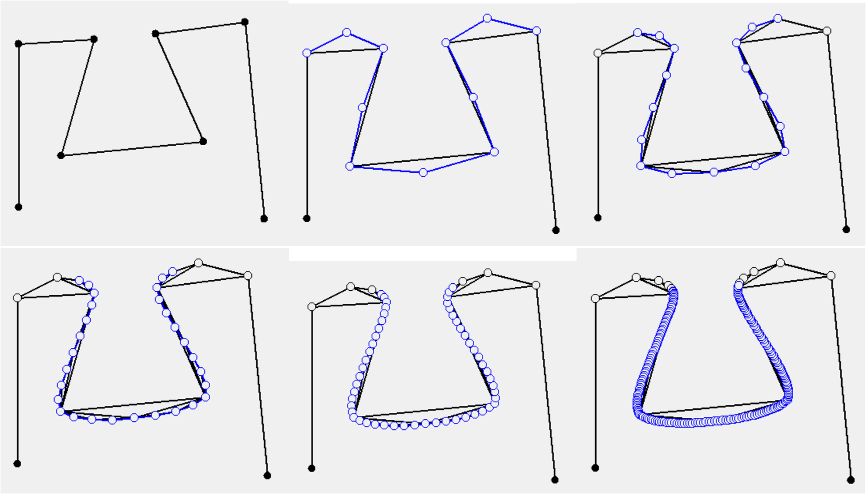
Andrzej Szymczak

November 1, 2011

# 4-point rule

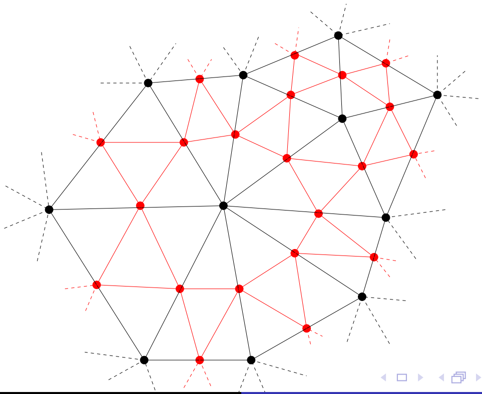
- Wouldn't it be nice to have a way to draw a smooth curve that passes through the control points?
- This is what 4-point rule does; however, it is not a piecewise polynomial curve; in fact, there is no simple analytical expression for the curve!
- Subdivision scheme: insert the point $\frac{-P_{i-1}+9P_i+9P_{i+1}-P_{i+2}}{16}$ between $P_i$ and $P_{i+1}$
- The limit curve starts at the third control point and ends at the third last; it passes through all control points between them

# 4-point rule



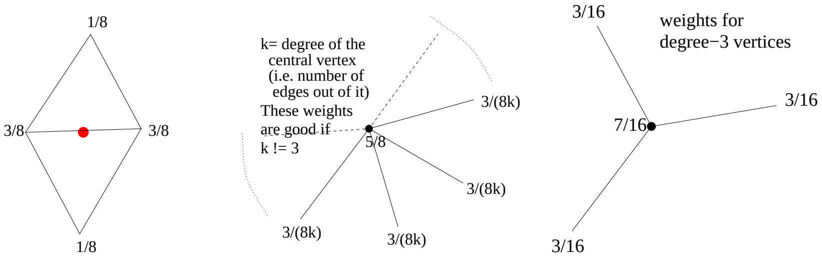
# Surface subdivision

- General scheme
  - Subdivide polygons into smaller ones
  - Move vertices around to smooth the mesh
- For triangle meshes, one can apply 1 : 4 subdivision (split each triangle into 4); splitting lines (for the black mesh) are shown in red below

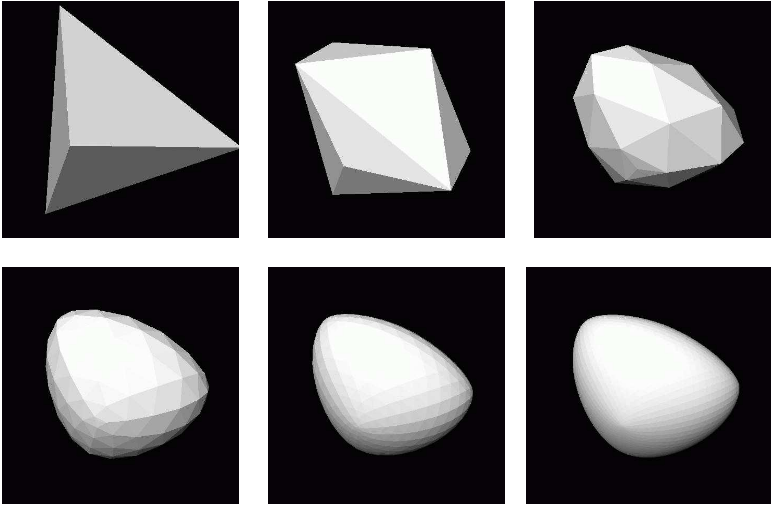


# Loop subdivision

- Two types of vertices
  - edge vertices (at the original mesh edges)
  - old vertices: at vertices of original mesh
- Old vertices move to a weighted average of neighbors in the original mesh
- Edge vertices move to a weighted average of vertices of the two triangles (in the original mesh) incident to the edge
- Weights are shown below



# Tetrahedron



# More examples (Beethoven, Johnny Hendrics)

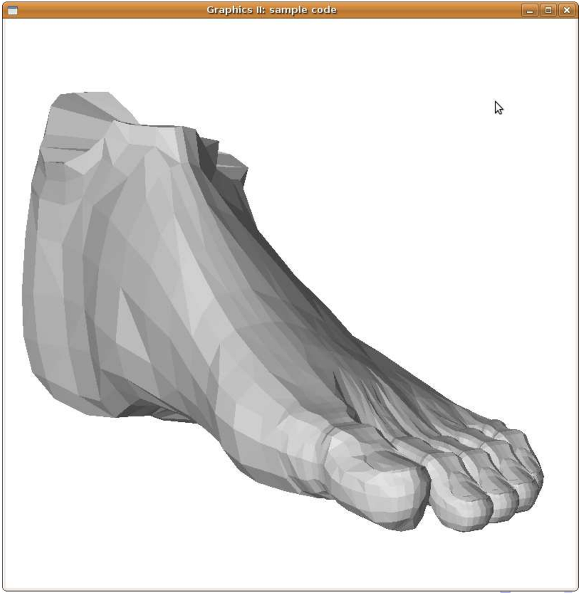
# More examples (Beethoven, Johnny Hendrics)

# More examples (Beethoven, Johnny Hendrics)

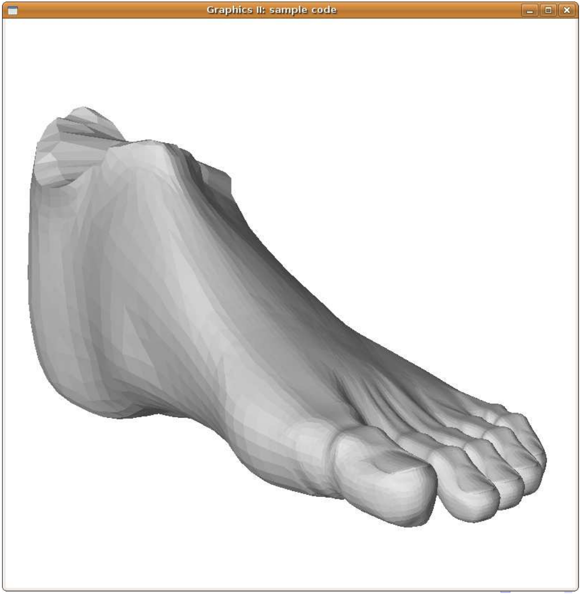
# More examples (Beethoven, Johnny Hendrics)

# More examples (Beethoven, Johnny Hendrics)

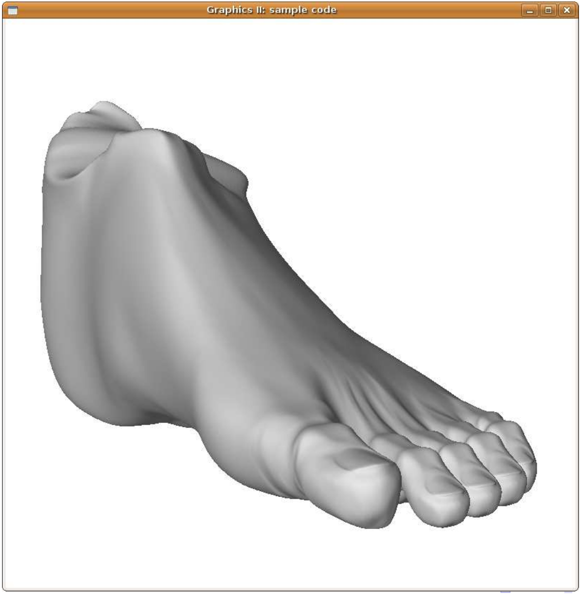
# More examples (foot, Scott Wiedemann)



# More examples (foot, Scott Wiedemann)

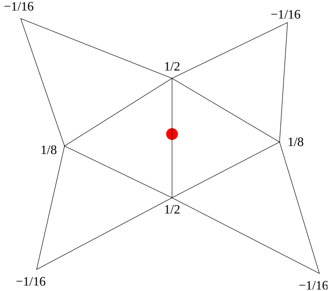


# More examples (foot, Scott Wiedemann)



# Butterfly subdivision

- Leave the old vertices in place
- Interpolating scheme: limit surface passes through vertices of the control mesh
- Move the edge vertices using the weights shown below (note that some of them are negative!)
- Generally, not as good properties as the Loop subdivision (but some issues can be fixed by manipulating the weights)



# Interactive demos

- Surface subdivision (does not look like Loop or butterfly)
- B-spline curve