Announcements

- Second assignment due Thursday, 2:30p.
- Thursday: pearlescent surfaces. *CIS*.
- Added paper for *Pearlescent Surfaces*.
- Tuesday: total degree surfaces, wavelet-based representations. *CIS*.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

The second assignment is a bit harder than the first one. Start now — Wednesday night will definitely be too late to finish on time. This time, I will not be lenient with extensions.

Are Lucas Pereira and Robert Williamson here?

All color figures from two of the papers are on the WWW. I brought all the papers to class to see the pictures.

The added paper is being handed out today. Leftover copies will be left on my desk. It is a simple two-pager discussing an efficient implementation of the algorithm presented in Peercy's paper.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Today

- Rational curves:
  - Example.
  - Control vectors.
  - Degree raising.
  - NURBS curves.

- Surfaces:
  - Parametric surfaces.
  - Tensor product patches.
  - Tensor product splines.
  - NURBS surfaces.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Rational Curves: Example

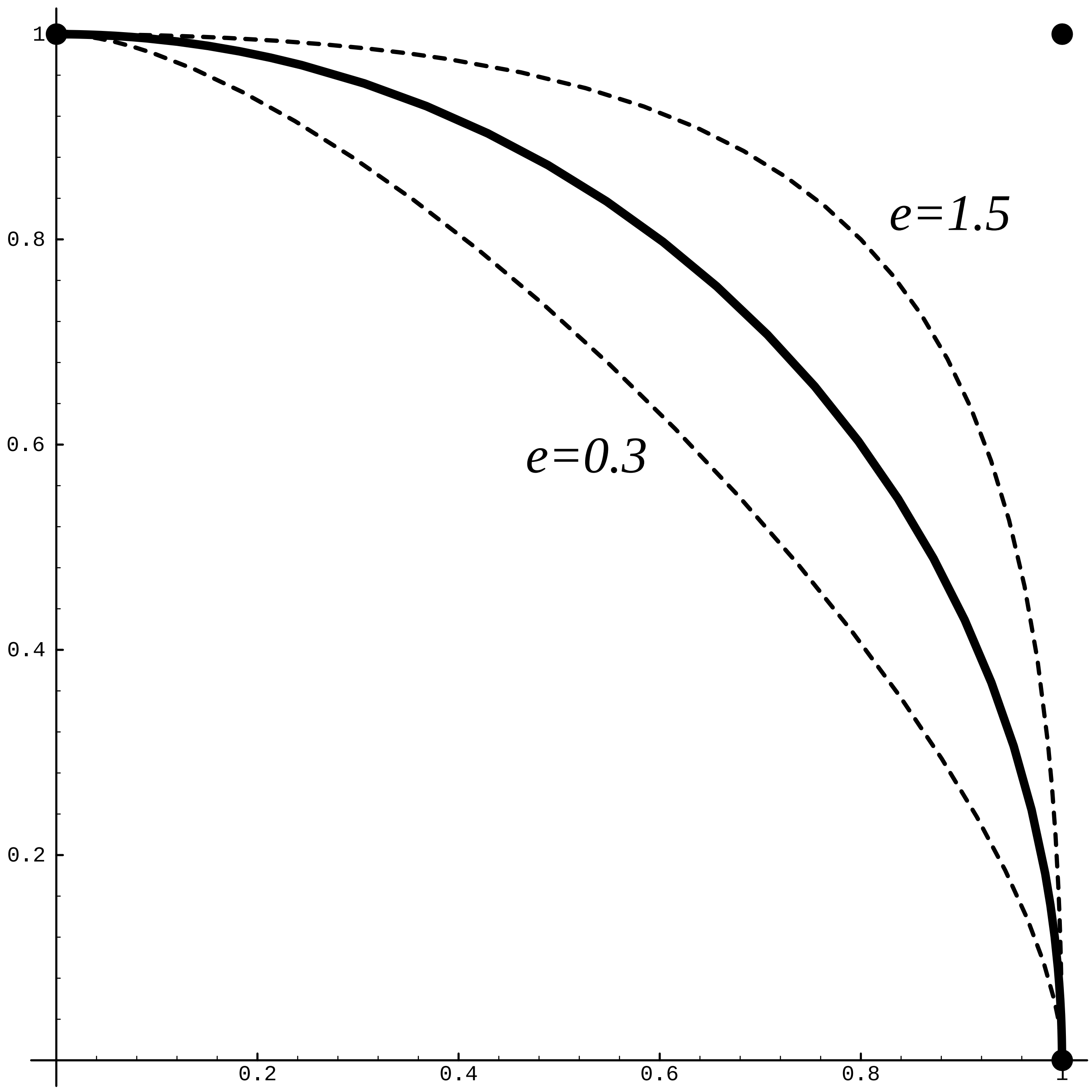
Let

$$\begin{aligned} w_1 &= 1 & (x_1, y_1) &= (1, 0) \\ w_2 &= e & (x_2, y_2) &= (1, 1) \\ w_3 &= 1 & (x_3, y_3) &= (0, 1). \end{aligned}$$

Find e so that curve is a circle.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Rational Curves: Example



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

We know that the curve has to be a conic section. And we know that it interpolates $(1, 0)$ and $(0, 1)$, and is tangent to these points with tangential directions perpendicular to the axes. So, there is only one choice for our desired circle — the unit circle centered at the origin and passing through

$$(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}).$$

The shoulder point of the rational curve appears at

$$\frac{\bar{\mathbf{b}}_1 B_1(0.5) + e\bar{\mathbf{b}}_2 B_2(0.5) + \bar{\mathbf{b}}_3 B_3(0.5)}{B_1(0.5) + eB_2(0.5) + B_3(0.5)} =$$
$$0.25 \frac{\bar{\mathbf{b}}_1 + 2e\bar{\mathbf{b}}_2 + \bar{\mathbf{b}}_3}{0.5 + 0.5e} =$$
$$\frac{0.5}{1+e}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} + 2e\begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right] =$$
$$\frac{0.5}{1+e}\begin{pmatrix} 1+2e \\ 1+2e \end{pmatrix}.$$

This is the only point on the rational curve whose x and y coordinates are equal (for finite t). This fact can be inferred via algebraic manipulations. Alternatively, and more intuitively, we can argue as follows: the given Bézier points only define conics symmetric about the line $x = y$. As all conics intersect an arbitrary line at most twice, our rational curve intersects $x = y$ at most twice: at $t = 0.5$, the only intersection for finite t occurs.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

To finish up, we observe that, for the curve to match the unit circle, we need e such that the shoulder point appears at

$$(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}).$$

This condition is equivalent to

$$\frac{0.5 + e}{1 + e} = \frac{1}{\sqrt{2}},$$

which is satisfied when

$$e = \frac{1}{\sqrt{2}}.$$

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Control Vectors

If $w_i = 0$, we should not think of control point

$$w_i = 0 \qquad (x_i, y_i)$$

as a point-at-infinity. Why? Because

- $x_i = y_i = 0$ is allowed, and
- different multiples of (x_i, y_i) yield different shapes.

Instead, call it *control vector*.

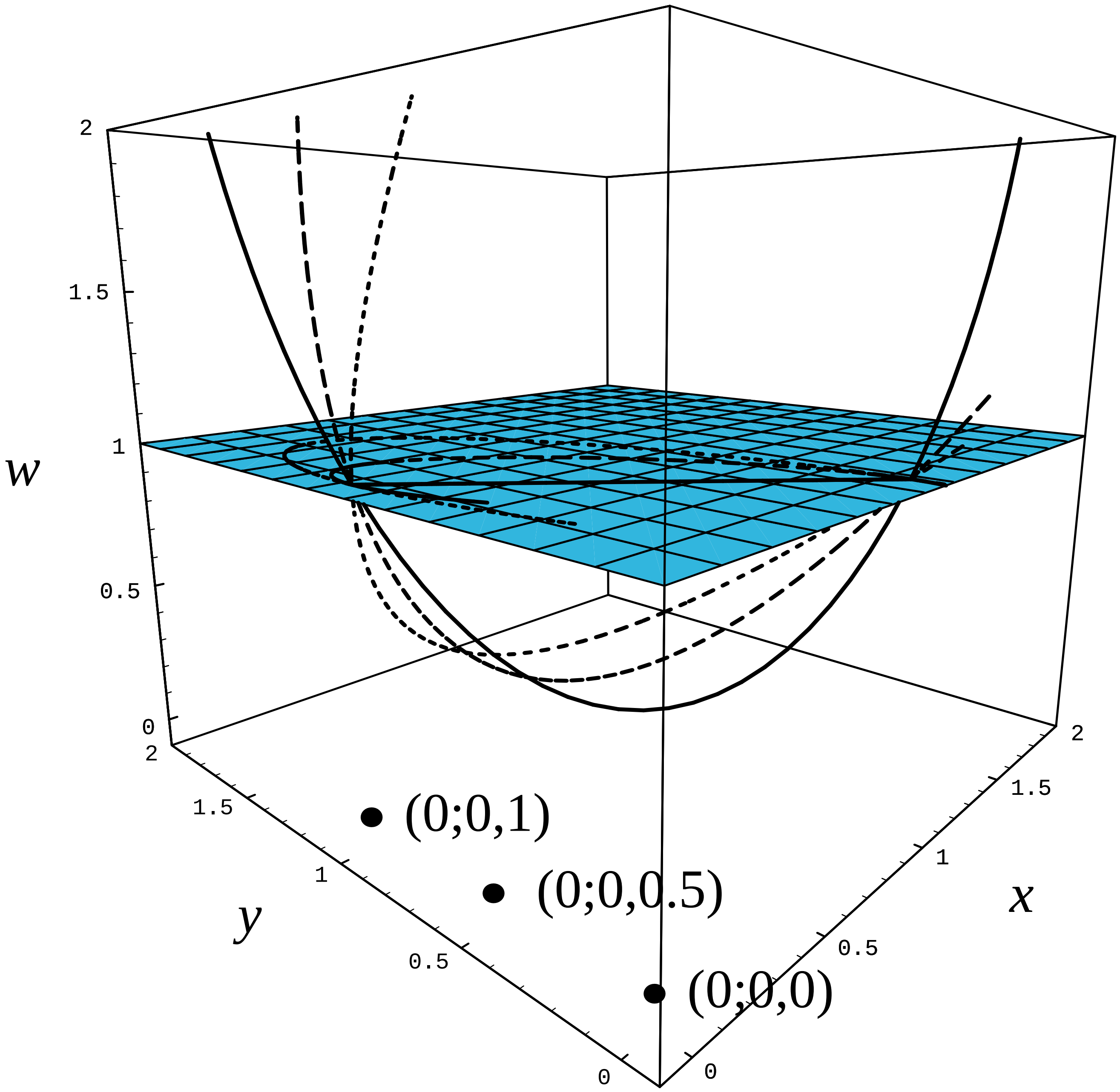
Find curve points using projective form

$$\mathbf{p}(t) = \mathbf{b}_1 B_1(t) + \mathbf{b}_2 B_2(t) + \mathbf{b}_3 B_3(t),$$

where the $\mathbf{b}_i$'s are *control sites*. Then draw by projecting on $w = 1$ plane.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Control Vectors: Example



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Recall that, when we classified the degree 2 rational curves, we got the line (the *really* narrow, degenerate ellipse) using $e = 0$. That is, our second Bézier point was $(0; 0, 0)$!

Here we are replicating the line, whose second Bézier point was $(0; 0, 0)$, and we add two more curves in dashed and dotted line styles: their second Bézier points are, respectively, $(0; 0, 0.5)$, and $(0; 0, 1)$. The three resulting projections are very different from each other; hence, varying y in the second Bézier point $(0; 0, y)$ produces very different rational curves.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Degree Raising

Important difference from Bézier curves: there is no unique way to degree-raise a rational curve.

Algorithm for degree raising:

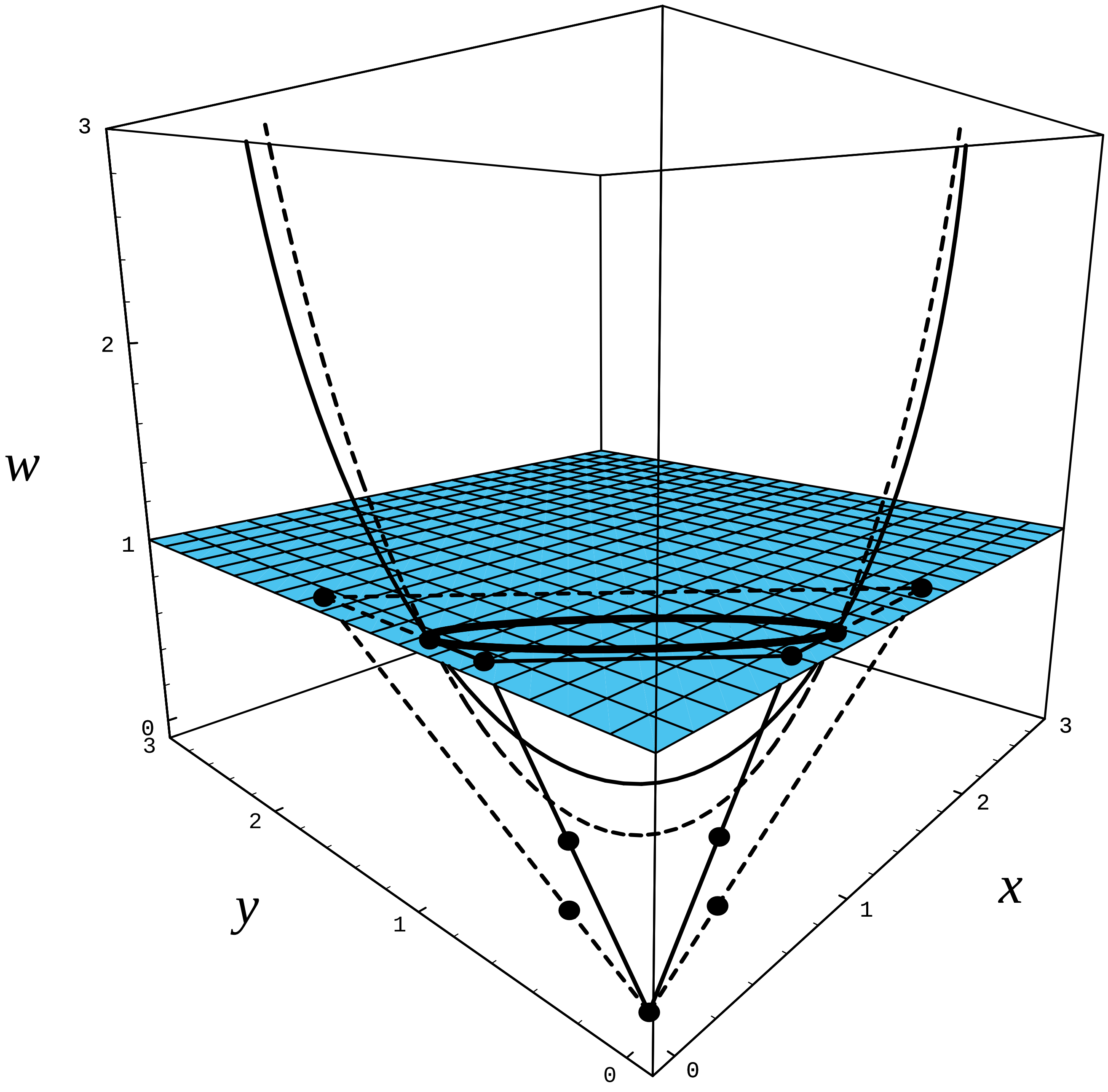
1. degree raise generating Bézier curve, and then

2. project its Bézier points to get Bézier points of degree-raised rational curve.

Result will be *different* if we reparameterize the rational curve before the first step.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Degree Raising: Example



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Here we are replicating the two alternate parameterizations of the ellipse for $e = \pm 0.5$. The two alternate generating parabolas are drawn in solid and dashed line styles. We have raised the degree of both parabolas to obtain their cubic Bézier points, shown on the plot. Projecting these points back onto the $w = 1$ plane yields two different control polygons (and associated Bézier points) for the same rational curve.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

NURBS Curves

Splines whose segments are rational curves. Or, projections of generating B-splines.

Non-Uniform: knots need not be equidistant.

Rational: segments are rational curves.

B-Splines: *spline:* segments join with some guaranteed continuity; *B-spline:* shape specified using de Boor points.

Example: planar NURBS curves are projections of generating 3D B-splines on $w = 1$ plane. They are specified by de Boor points which are the projections of the generating B-spline's de Boor points.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

If the segments of the generating B-spline join with C^n continuity, so will the NURBS segments. The reason is that the projection of a particle moving in 3D space is subject to no more discontinuities than the movement of the original particle. This is not always self-evident. If the generating B-spline crosses the $w = 0$ plane, then the situation is tricky as the rational curve may go to infinity. However, this is a rare case, and does not concern us since it can be safely eliminated if we allow only positive weights in the NURBS de Boor points (which is equivalent to forcing the generating B-spline to lie above the $w = 0$ plane). Indeed, most, if not all, commercial modelers, disallow negative weights.

A situation which does not concern us when designing NURBS is illustrated by the following example: consider a joint of a 3D B-spline with C^2 continuity between the two segments s_1 and s_2 meeting at the joint. The projection of this curve is a rational curve with C^2 continuity at the projected joint, as we discussed above. Now take the first of the two adjacent segments of the rational curve, $\bar{s}_1$, and scale its weights by the same number. This scaling did not affect at all the continuity at the joint of the rational spline. However, the generating curve of the new rational spline is a 3D B-spline whose two adjacent segments s'_1 (the 3D version of $\bar{s}_1$) and s_2 are, in general, fully discontinuous. In summary, while C^2 continuity of the generating B-spline implies C^2 continuity of the rational B-spline, the inverse is not always true.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

As Bézier curves are a special case of rational curves, B-splines are a special case of NURBS: setting the weight equal to unity for all the NURBS de Boor points yields a regular B-spline. The reason is that, if all the weights are one, the generating B-spline defining the NURBS lies fully on the $w = 1$ plane and is thus the NURBS itself.

A quick note on the naming of splines. A spline is a piecewise curve (or surface, as we will see later today), whose segments join with some continuity. A B-spline is a spline whose shape is defined by de Boor points. Other ways to define the shape of splines is by interpolation, i.e. the user specifying a set of points through which a spline must pass; such splines are called *cardinal* or *Catmull-Rom* splines. The term *control points* always refers to the user-specified points which define the spline's shape. Finally, while NURBS are B-splines, the term *B-spline* usually refers to piecewise polynomials curves (or surfaces).

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

NURBS Curves: Motivation

Why use NURBS?

- Retain all good properties of B-splines.
- Projection invariant.
- Can perfectly represent conics.
- Weights add flexibility in shape specification, refinement.

Why not?

- Slower drawing (rational interpolations).
- Extra flexibility might be overkill — Occam: keep UI as complex as task demands and no more.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Comments on why NURBS are desirable:

- For example, the convex hull property is still valid if all the weights have the same sign: a NURBS segment is fully contained in the convex hull of those de Boor points which were used to draw the segment. The reason for the positivity condition on the weights can be intuitively understood by mentally visualizing the projection operation: if some weights have different signs, the generating B-spline may cross the $w = 0$ plane and thus the NURBS may contain points-at-infinity which no convex hull can contain.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

- Projective invariance is similar to affine invariance. It means that, in order to draw the result of applying a projective transformation to a NURBS, we can

  1. apply this transform to the de Boor points of the NURBS, and
  2. draw a new NURBS whose de Boor points are the transformed de Boor points of the original NURBS.

  This new NURBS will be identical to the result of transforming the original one, point by point.

  This property follows from the affine invariance of the generating B-spline: a projective map of the Euclidean plane is an affine (in particular, a linear) map of the projective plane, which, as we've seen above, is algebraically equivalent to 3D space — the space where the generating B-spline resides.

  Projective invariance is really important in rendering NURBS, since the viewing transformation required to render a scene is a projective map. That is, we only have to pipe the few de Boor points through the graphics pipeline in order to render a NURBS; we need not pipe through every single point on the curve.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

If B-spline curves are a special case of NURBS, why doesn't it follow that B-splines are projectively invariant? The reason is that the result of projectively transforming a B-spline curve is an arbitrary NURBS, not just a B-spline. Hence, unless our hardware can draw NURBS, B-splines have to be transformed point-by-point during projective transformations.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

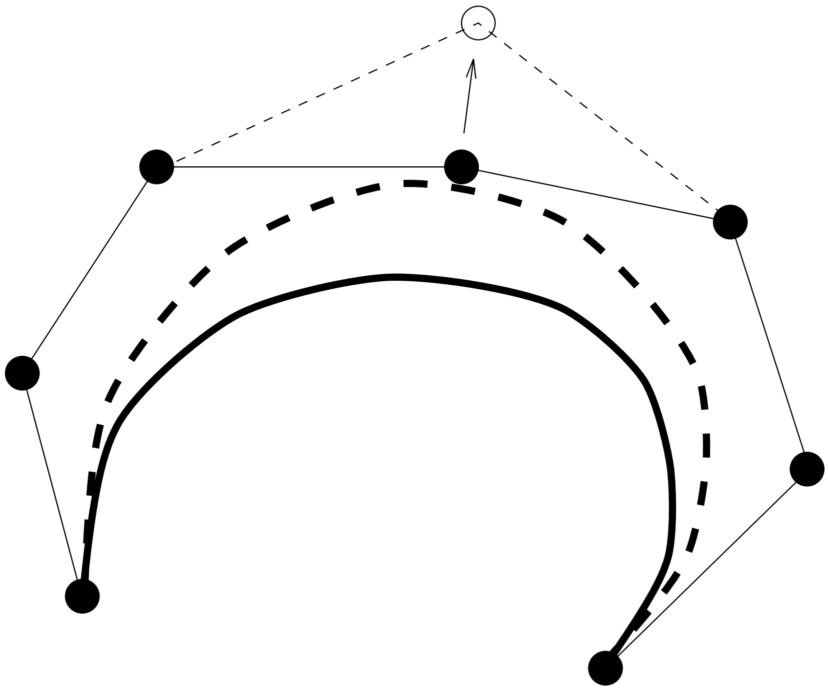
- Conics are very common in CAD (e.g. lenses have spherical surfaces, and the top view of a spiral staircase is a circle), and they need to be precisely drawn (not just approximated) for many applications, such as manufacturing or architectural design. A drawing package using B-spline curves needs to define separate primitives for circles, ellipses, and other conics; however, a NURBS-based CAD tools can represent conics using the just NURBS primitive. Such uniformity in representation is of tremendous importance is designing intuitive user interfaces, easing code development (less special cases), providing generalized operations (which apply to all primitives), etc.

- The next two slides illustrate the effect of weights in designing (i.e. specifying shapes) using NURBS. In terms of shape refinement of B-splines, we have so far discussed knot insertion and degree raising as methods of adding flexibility to a B-spline. An alternate method for B-spline shape refinement is to add weights to the de Boor points. That is, to refine the shape of a B-spline, we now have the option of turning it into a NURBS by adding unity weights to all the de Boor points. Then, by changing these weights, we can effect subtle local changes easily. The primary drawback is that our hardware or modeling software might not support NURBS; also, like degree raising, turning a B-spline into a NURBS is a not a local operation.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

NURBS Curves: Flexibility

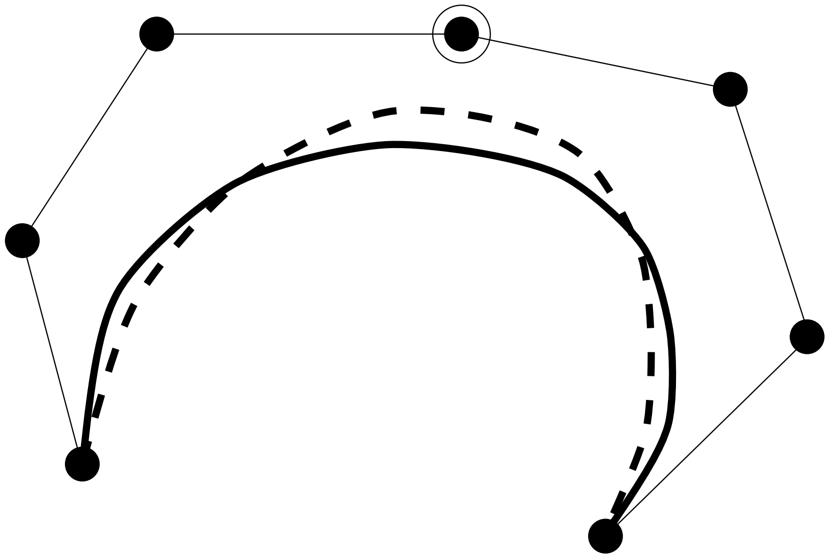
Moving the control point:



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

NURBS Curves: Flexibility

Increasing the weight:



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

These slides illustrate the added flexibility rational curves offer the designer: moving a control point has a different effect on the curve than increasing its weight.

In the drawings, the solid line style is used for the original NURBS, while the dashed line style is used for the modified NURBS. The point whose properties (position or weight) are modified is shown as a hollow circle after the modification, the radius representing the relative change in the point's weight.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Parametric Surfaces

Surface given by

$$\mathbf{p}(u, v)$$

where

- u, v are *parameters*; uv space is *domain space*;

- as u, v vary, $\mathbf{p}(u, v)$ traces a *surface* in 3D.

For

$$u_0 \leq u \leq u_1 \quad \text{and} \quad v_0 \leq v \leq v_1,$$

$\mathbf{p}(u, v)$ maps *domain rectangle* to *surface patch*, the *image* of the rectangle. Similarly for triangles.

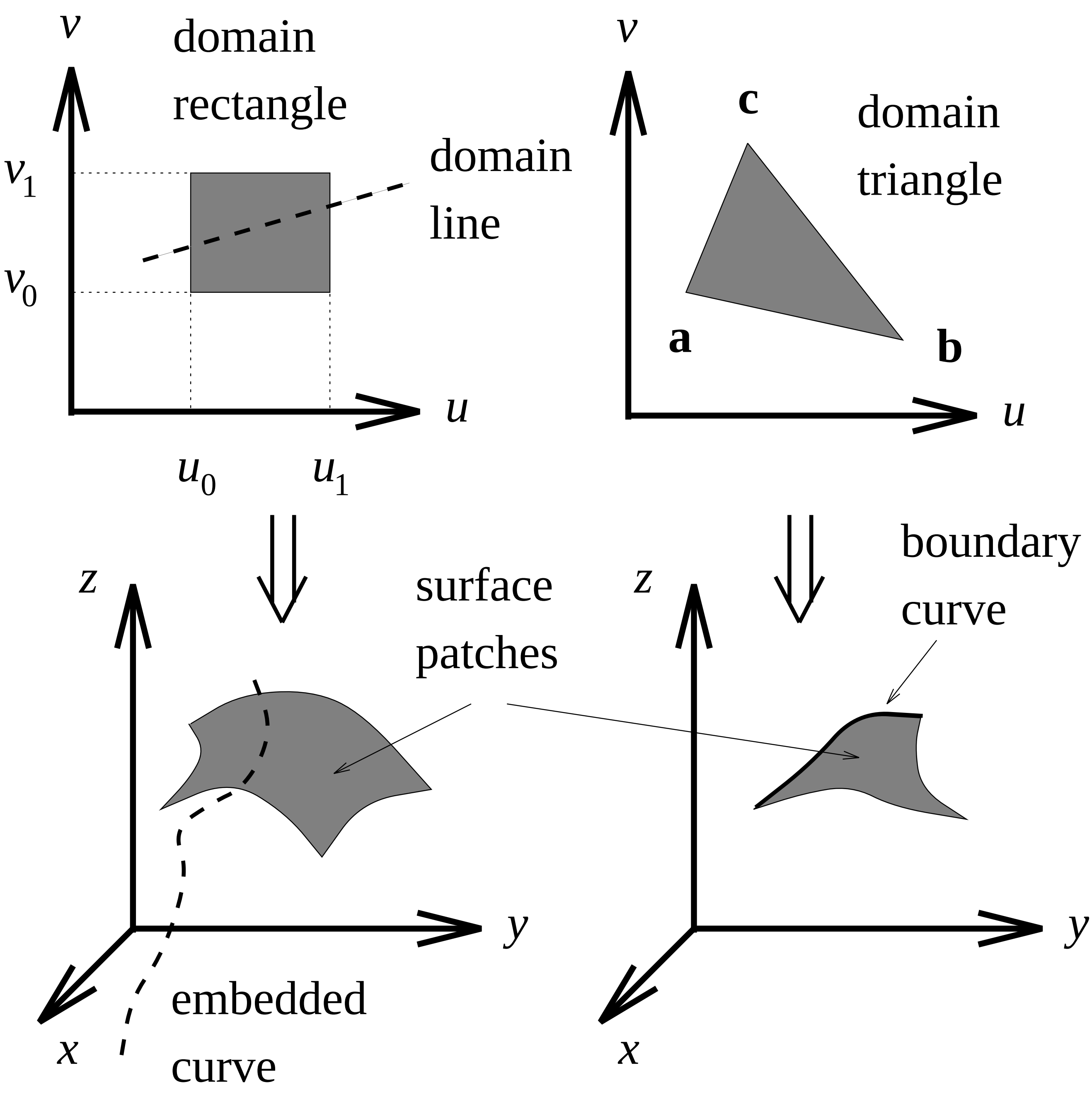
Domain lines map to curves *embedded* in surface. If domain lines are edges of domain rectangle/triangle, their image is *boundary curve*.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

We will now embark on a broad overview of surface modeling. We will cover the topic more lightly than rational curves, drawing on intuition much more than on formal proofs.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Parametric Surfaces



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

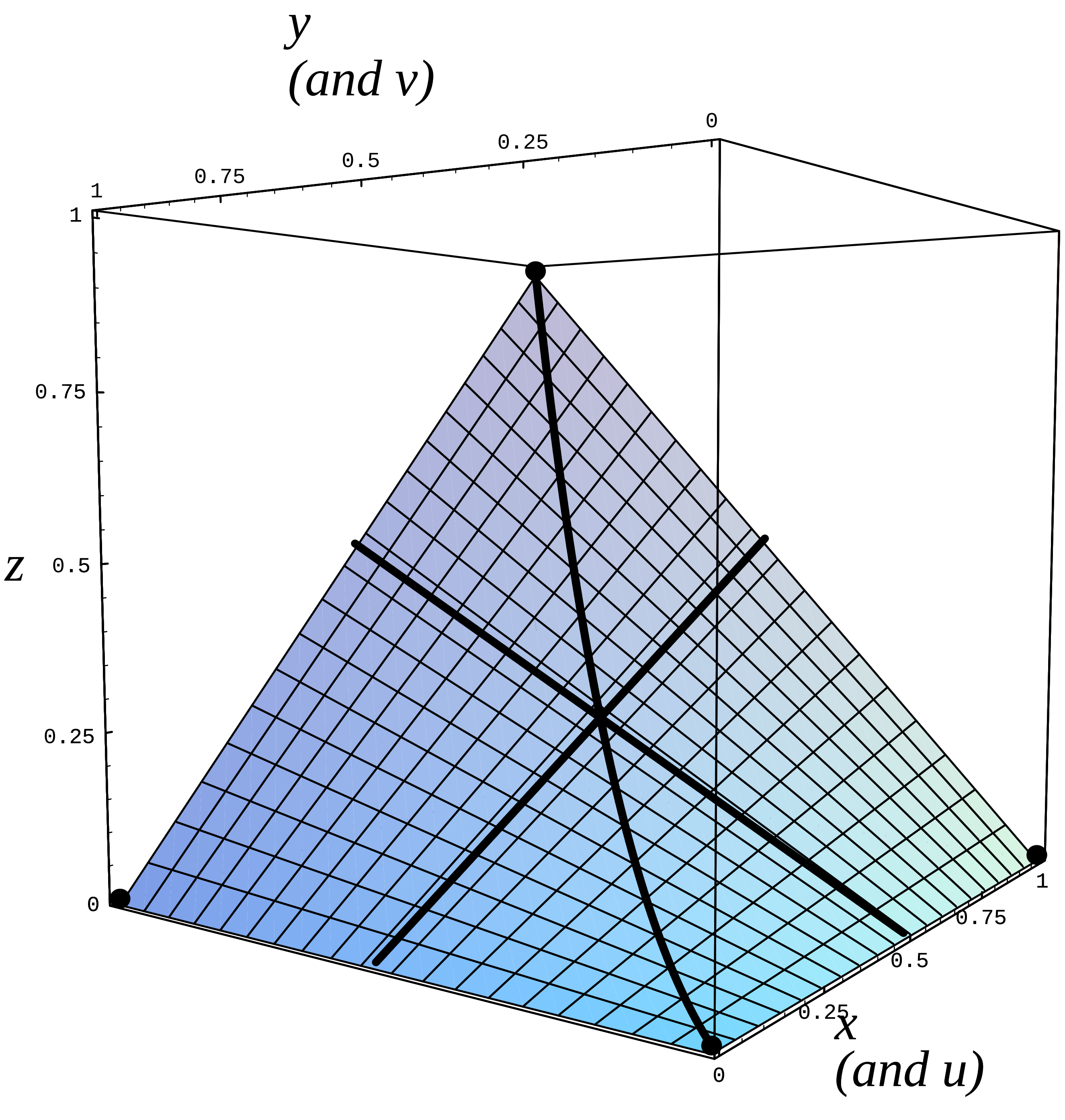
Tensor Product Patches: Ruled

Let

$$
\begin{aligned}
\mathbf{p}(u,v) &= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}(1-u)(1-v) + \\
&\quad \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}u(1-v) + \\
&\quad \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}(1-u)v + \\
&\quad \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}uv \\
&= \begin{pmatrix} u \\ v \\ uv \end{pmatrix}.
\end{aligned}
$$

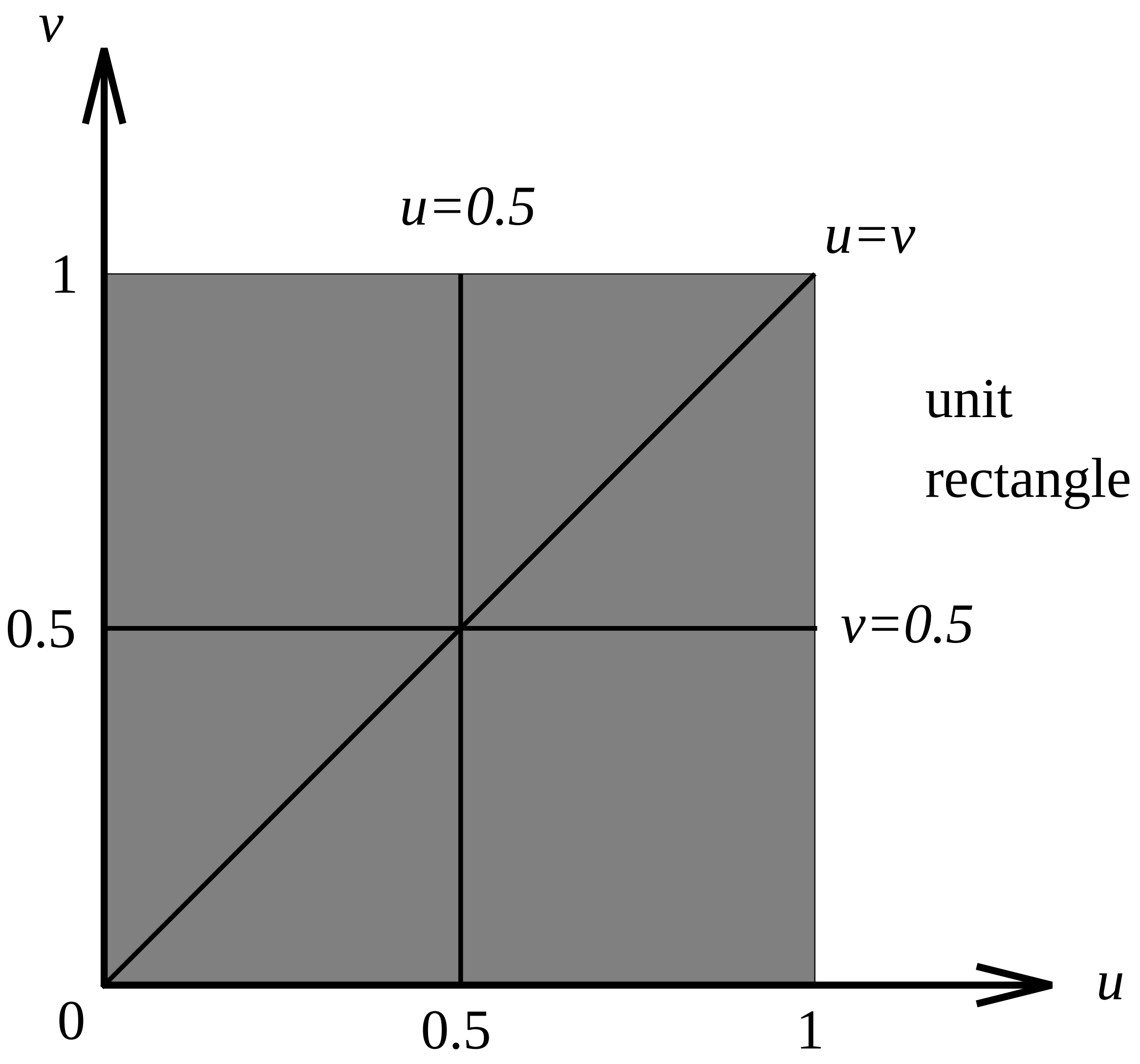
© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Ruled



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Ruled



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

The first plot shows (in part) the patch of $\mathbf{p}(u, v)$ corresponding to the *unit* domain rectangle

$$0 \leq u \leq 1 \quad \text{and} \quad 0 \leq v \leq 1.$$

In this plot, the points

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

are highlighted as solid disks.

The second plot shows (in part) the unit domain rectangle in gray.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Ruled

Can interpret formula for $\mathbf{p}(u, v)$ in two equivalent ways; surface is traced either by

- line segment with moving vertices at

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}(1-u)+\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}u, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}(1-u)+\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}u,$$

  where u is time and interpolation in v gives points along segment; or by

- line segment with moving vertices at

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}(1-v)+\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}v, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}(1-v)+\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}v,$$

  where v is time and interpolation in u gives points along segment.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

The two alternate views come from the two different, symmetric ways we can rewrite the expression defining $\mathbf{p}(u,v)$:

$$\begin{aligned}\mathbf{p}(u,v) &= (1-v)\left[\begin{pmatrix}0\\0\\0\end{pmatrix}(1-u)+\begin{pmatrix}1\\0\\0\end{pmatrix}u\right]+\\ &\qquad v\left[\begin{pmatrix}0\\1\\0\end{pmatrix}(1-u)+\begin{pmatrix}1\\1\\1\end{pmatrix}u\right],\end{aligned}$$

or

$$\begin{aligned}\mathbf{p}(u,v) &= (1-u)\left[\begin{pmatrix}0\\0\\0\end{pmatrix}(1-v)+\begin{pmatrix}0\\1\\0\end{pmatrix}v\right]+\\ &\qquad u\left[\begin{pmatrix}1\\0\\0\end{pmatrix}(1-v)+\begin{pmatrix}1\\1\\1\end{pmatrix}v\right].\end{aligned}$$

The moving curve, a line segment for this particular surface, is called the *tracing curve*.

In the previous 3D plot, observe the two orthogonal families of aligned line segments: each family corresponds to one of the above alternate views of $\mathbf{p}(u,v)$.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Ruled

$$\mathbf{p}(u, v) = \begin{pmatrix} u \\ v \\ uv \end{pmatrix}$$

is line for constant u or constant v (*ruled surface*).

Image of domain line $u = v = t$ is

$$\mathbf{p}(t) = \begin{pmatrix} t \\ t \\ t^2 \end{pmatrix},$$

a parabola.

So domain lines not parallel to axes in domain space do not always map to embedded *lines*.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

There are two ways to see that $\mathbf{p}(u, v)$ is a line for constant u or v. Focusing on constant u, one way is to use the given expression for $\mathbf{p}(u, v)$ and immediately conclude that

$$\mathbf{p}(u_0, v) = \begin{pmatrix} u_0 \\ v \\ u_0 v \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ u_0 \end{pmatrix} v + \begin{pmatrix} u_0 \\ 0 \\ 0 \end{pmatrix}.$$

The other way is to make use of our earlier remark: keeping u constant means that the tracing line with vertices at

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}(1-u) + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}(1-u) + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} u$$

is frozen in time.

When we use 0 or 1 for the constant value of u or v we obtain the edges of the domain rectangle. Hence, the images of these lines are the boundary curves of the patch. In the previous drawings, these line segments appear at the four edges of the unit rectangle and the patch.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Also in the drawings, we have highlighted three domain lines and their images:

- two domain lines for which one of u or v is kept constant at 0.5, and their images (which are lines, as well), and

- the domain line $u = v$, whose image is the depicted embedded parabola.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Bicubic

Bicubic tensor product patch: image of unit domain rectangle with

$$\mathbf{p}(u, v) = \sum_{i=1}^{4} \sum_{j=1}^{4} \mathbf{b}_{i,j} B_i(u) B_j(v),$$

where

- the $\mathbf{b}_{i,j}$'s are the *Bézier points* of the patch, forming the *control net*, and

- the $B_i(u)$'s and $B_j(v)$'s are the *Bernstein polynomials* for cubic Bézier curves.

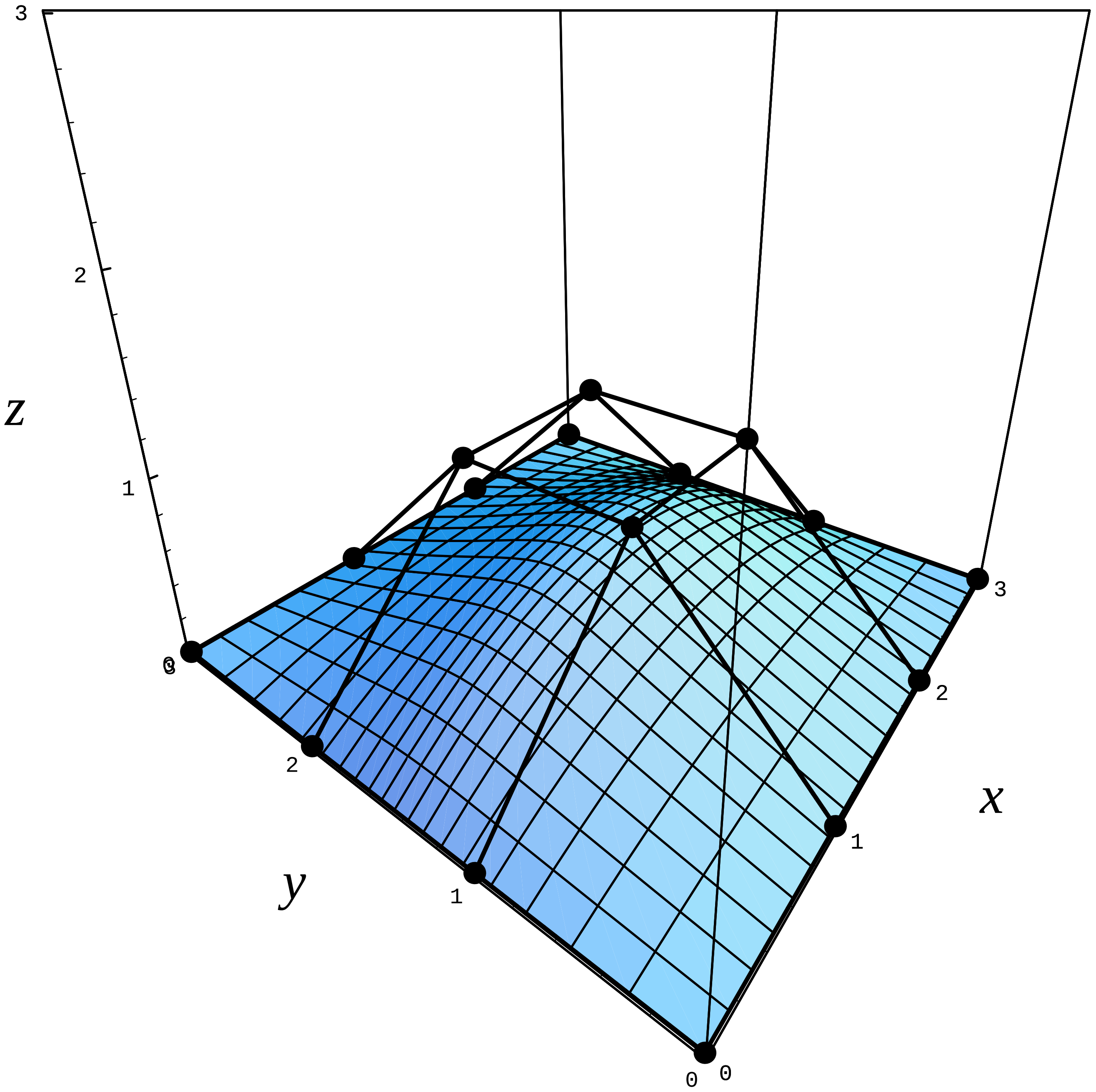
© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

We are now going to generalize ruled surfaces to bicubic patches. Ruled surfaces are also called bi-affine patches since the tracing curve is an affine curve (a line).

Of course, just as in the case of Bézier curves, we need not restrict ourselves to the unit domain rectangle: by modifying the basis polynomials, we can define bicubic tensor product patches over arbitrary domain rectangles, just as the domain of Bézier curves need not be the interval $[0, 1]$ on the real line.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Bicubic



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

In this plot, the Bézier points and control net of the patch are highlighted. We have placed the Bézier points in a very regular pattern in order to simplify the diagram; in general, they can be located anywhere in space.

Also, we have aligned the x axis of 3D space with the u axis of the domain space, and the y axis with the v axis. This alignment makes it easier to follow the result of keeping one of u or v constant while varying the other.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Bicubic

Can see patch as tracing cubic Bézier curve with moving Bézier points

$$\mathbf{p}(u,v) = \sum_{i=1}^{4} \left[\sum_{j=1}^{4} \mathbf{b}_{i,j} B_j(v) \right] B_i(u),$$

where v is time and interpolation in u gives points on curve. Bézier points (in brackets) also move along cubic Bézier curves.

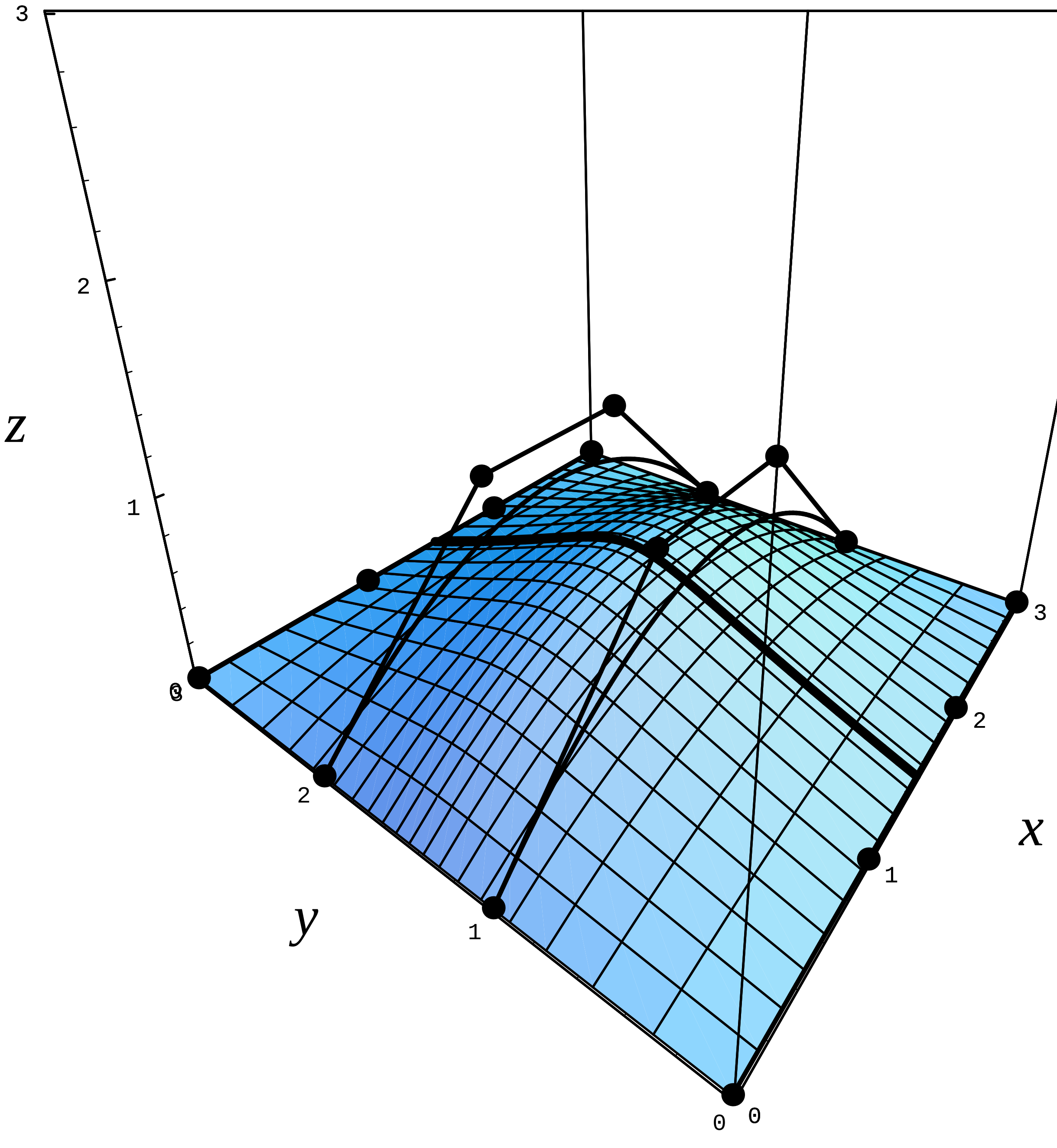
Or, tracing cubic Bézier curve with moving Bézier points

$$\mathbf{p}(u,v) = \sum_{j=1}^{4} \left[\sum_{i=1}^{4} \mathbf{b}_{i,j} B_i(u) \right] B_j(v),$$

where u is time and interpolation in v gives points on curve. Bézier points (in brackets) also move along cubic Bézier curves.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Bicubic



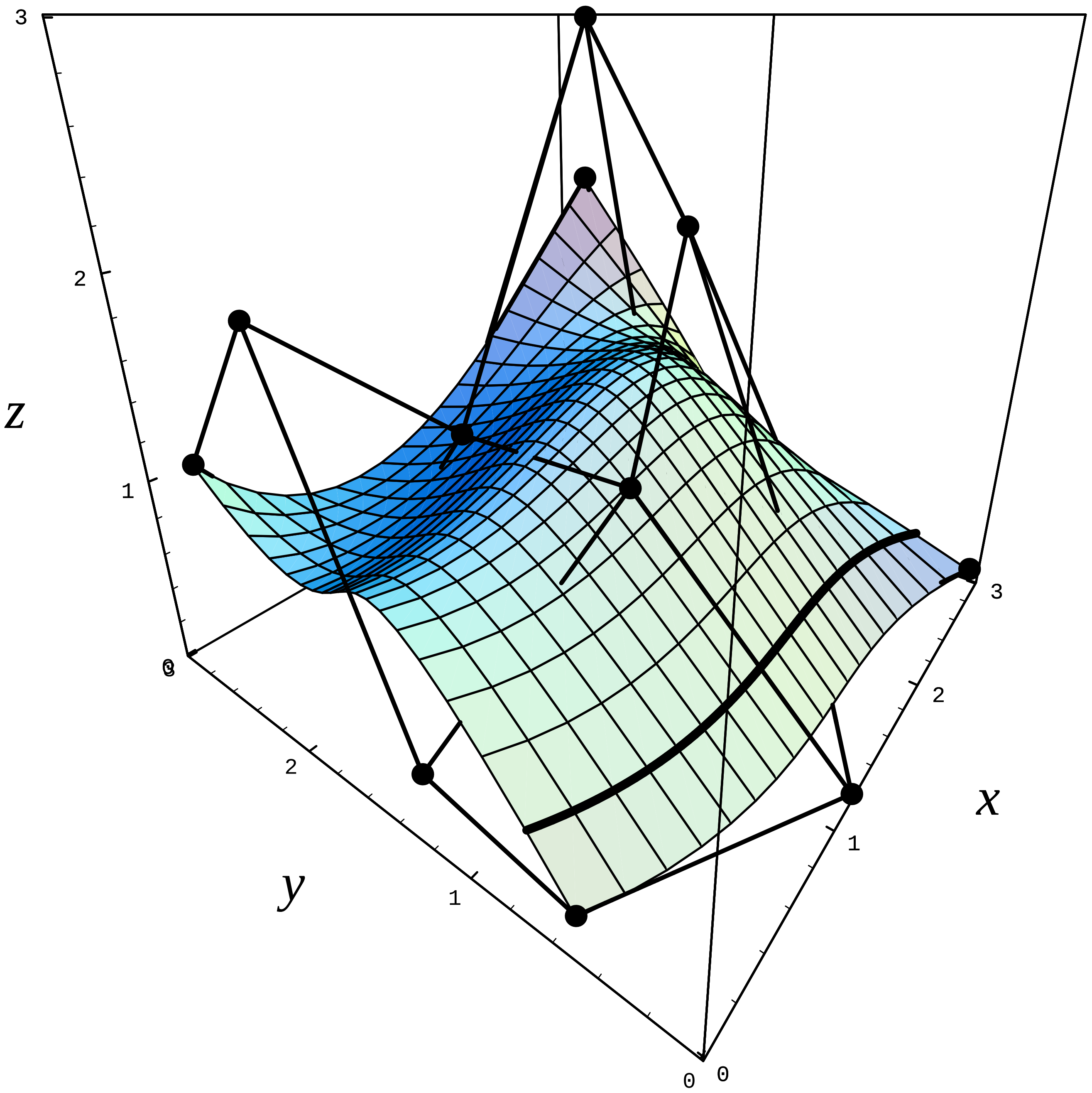
© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

In this plot, we have adopted the view that u is time and interpolation in v gives points on the tracing curve. We have drawn the Bézier points of the patch, the four paths on which the Bézier points of the tracing curve travel, as well as a snapshot of the tracing curve itself for $u = 0.5$.

Due to the regular placement of the Bézier points, one might get the impression that domain lines parallel to the domain axes map to planar curves. This need not be the case at all, as the following slide illustrates.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Bicubic



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

This is just a plot of a bicubic tensor product patch whose Bézier points are placed in a somewhat irregular pattern; still, we maintained some alignment between the x and y axes of 3D space and the u and v axes of the domain space, respectively. The highlighted curve is the image of a domain line parallel to the u domain axis; that is, v is constant along the domain line. Clearly, the embedded curve is not a planar one. Moreover, this surface patch, unlike our previous example, has a fairly complex and asymmetric shape.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

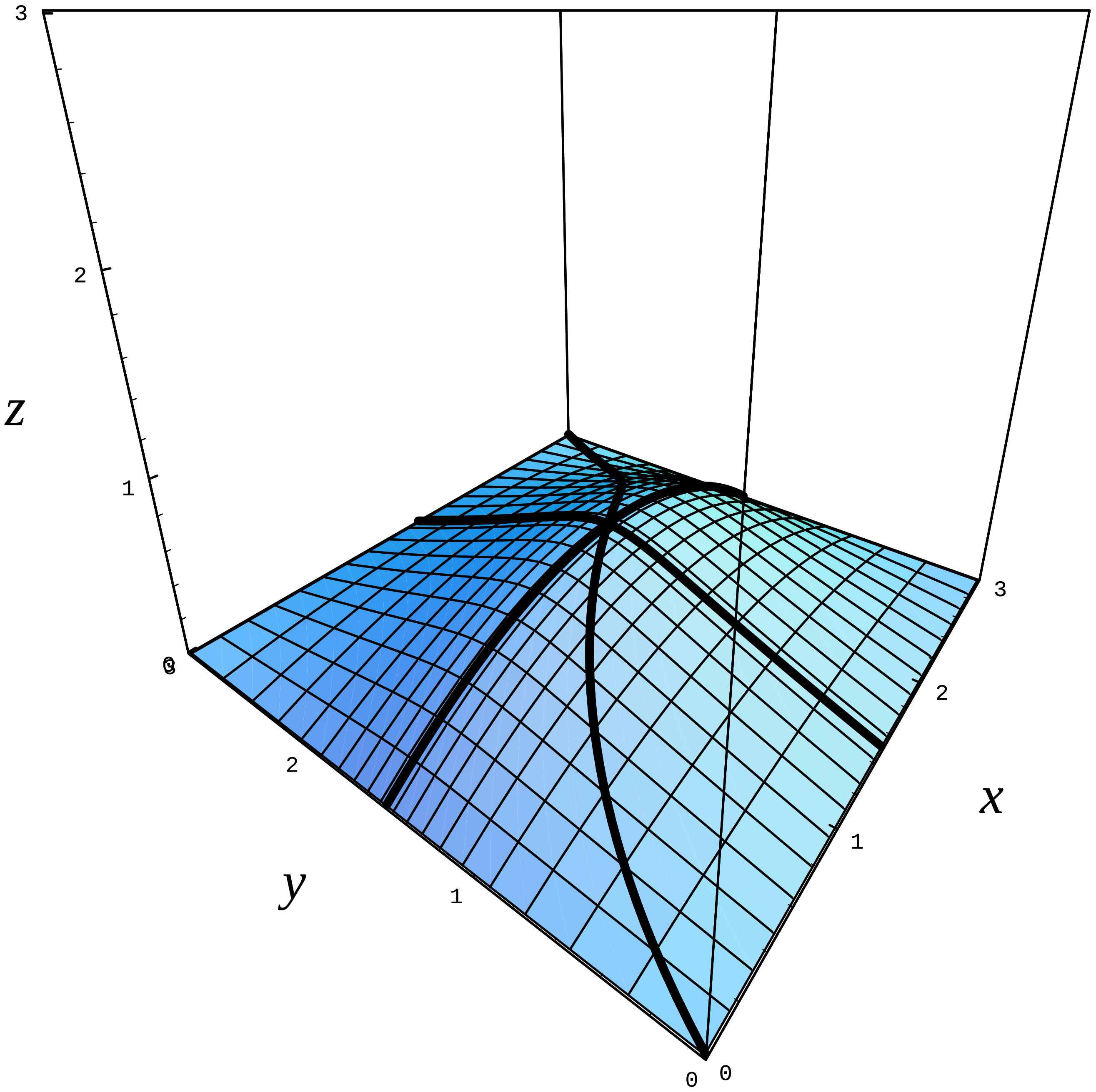
Tensor Product Patches: Bicubic

$\mathbf{p}(u, v)$ is cubic Bézier curve for constant u or constant v.

But domain lines not parallel to axes in domain space do not always map to embedded *cubic* Bézier curves.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Bicubic



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

In this drawing, we have highlighted three embedded curves on the patch:

- two cubic Bézier curves for which one of u or v is kept constant at 0.5, and

- the image of the line $u = v$, which is not a cubic Bézier curve; admittedly, it is hard for the inexperienced reader to see that the shown curve is not a cubic one.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: General

General tensor product patch: image of unit domain rectangle with

$$\mathbf{p}(u, v) = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{b}_{i,j} B_i(u) \mathcal{B}_j(v),$$

where

- the $\mathbf{b}_{i,j}$'s are the Bézier points of the patch, forming the control net of the patch,
- the $B_i(u)$'s are the Bernstein polynomials for Bézier curves of degree $m - 1$, and
- the $\mathcal{B}_j(v)$'s are the Bernstein polynomials for Bézier curves of degree $n - 1$.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

We have generalized bicubic patches to patches of arbitrary degree; moreover, the degree may be different along each of the axes of the domain space.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Properties

Properties follow from Bézier curves:

- Affine invariance.
- Convex hull.
- And many more. . .

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Algorithms

Algorithms derived by applying curve algorithms along each axis of domain space independently:

- Degree elevation.
- Subdivision.
- Polarization.
- And many more. . .

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

An example of generalizing an algorithm follows...

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Polarization

Polar form of biquadratic

$$\mathbf{p}(u,v) = \begin{pmatrix} uv \\ u^2 + v^2 \\ u + v + 1 \end{pmatrix}$$

is

$$\hat{\mathbf{p}}(u_1,u_2;v_1,v_2) = \begin{pmatrix} \frac{u_1+u_2}{2}\frac{v_1+v_2}{2} \\ u_1u_2 + v_1v_2 \\ \frac{u_1+u_2}{2} + \frac{v_1+v_2}{2} + 1 \end{pmatrix}$$

by polarizing u, v separately (i.e. as if other parameter was a constant).

Can draw patch by doing polar interpolations since polar form is affine w.r.t. each of u_1, u_2, v_1, v_2 separately.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Trimming

To make a hole, must define patches all around it and join them. Takes lots of patches to go around wide hole.

Engineer's hack: patches may have holes. How?

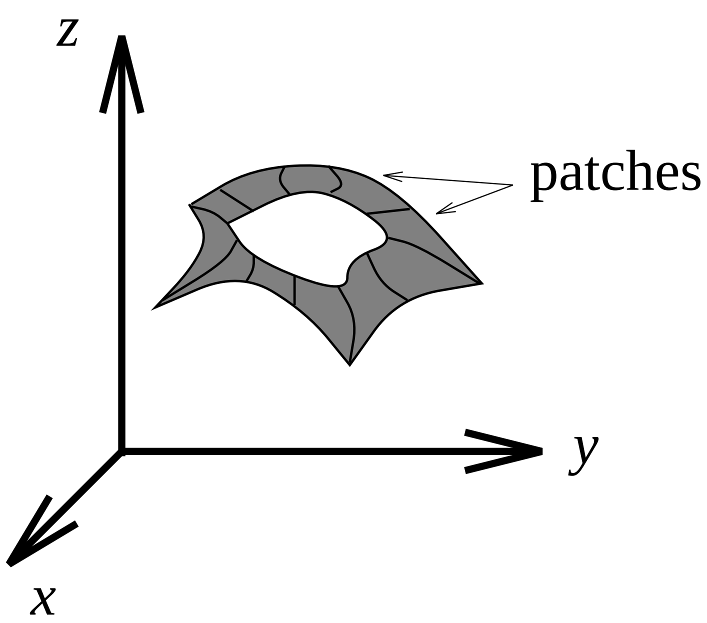
1. User marks part of domain rectangle as invalid (e.g. interior of some user-defined 2D Bézier curve in domain space).

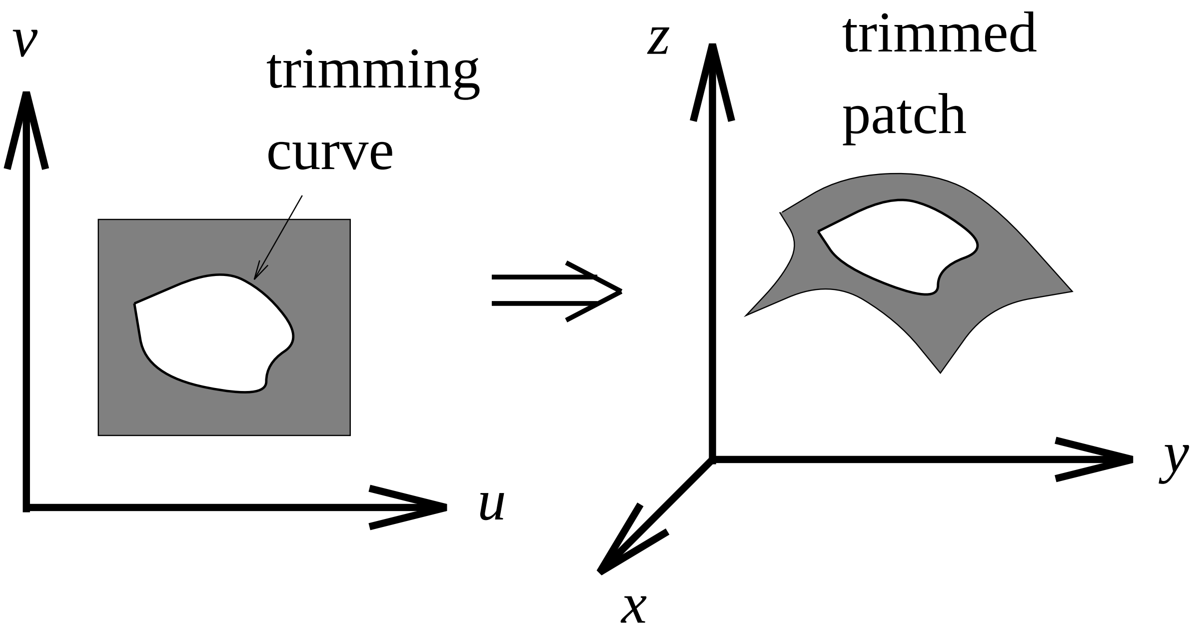
2. Rendering algorithm does not draw patch section with corresponding parameter values in invalid region.

Patches with holes are *trimmed patches*. 2D curve(s) defining hole are *trimming curves*.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Patches: Trimming





© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

On top is shown a collection of patches tiled around a hole. On the bottom, we have managed to draw the same shape, i.e. a rectangular surface with a hole, using a single trimmed patch: the diagram on the left shows the domain space and the trimming curve, while the diagram on the right shows the trimmed patch itself.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Splines

Join patches with some parametric continuity along patch boundaries. Continuity guaranteed by following theorem:

Two adjacent patches are C^n continuous along their common boundary if all rows of control points define C^n continuous Bézier curves.

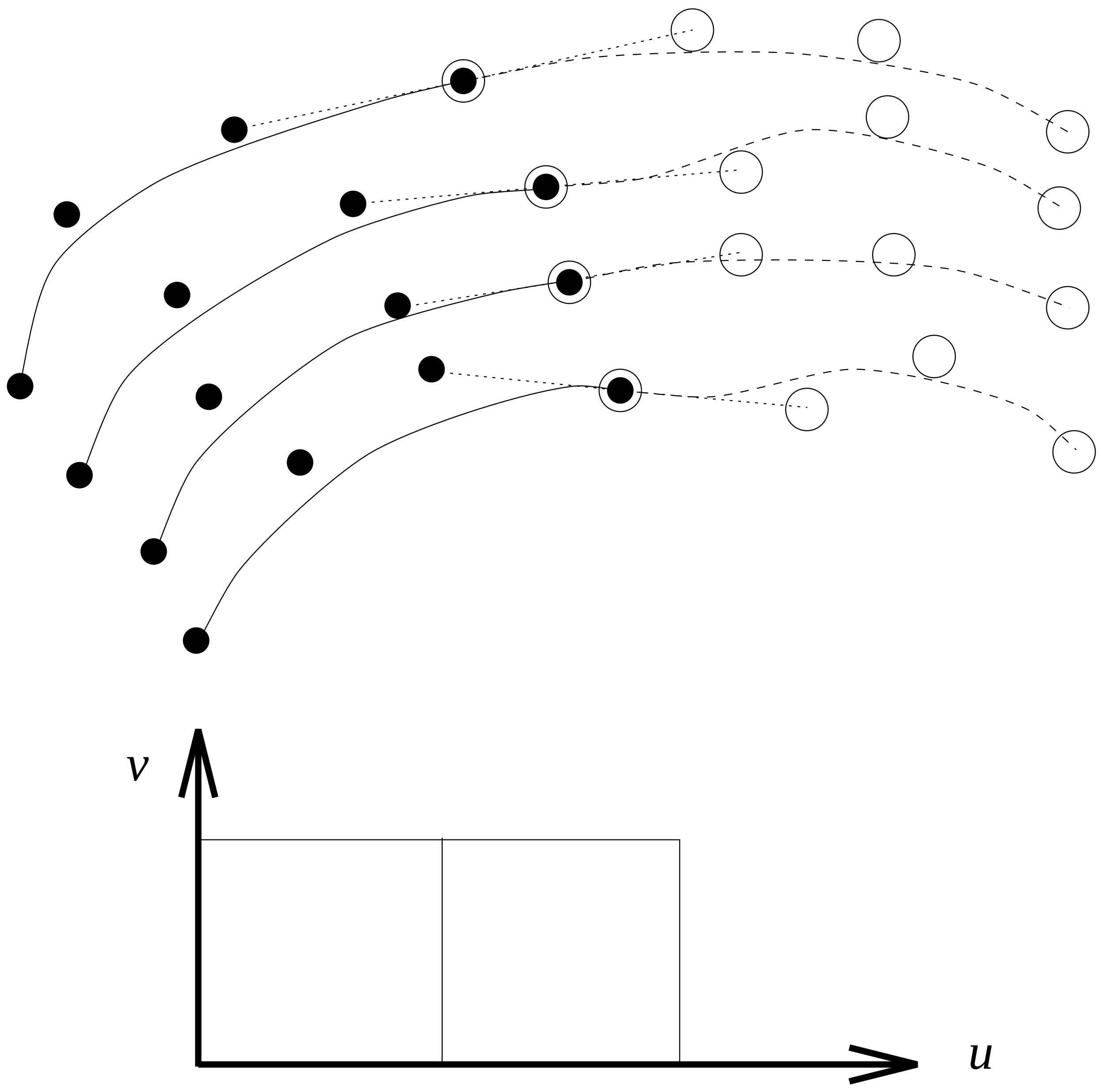
© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor product splines are also called *B-spline surfaces*, as their shape is usually specified using de Boor points.

The example shown on the next slide illustrates what we mean by “rows of control points”.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Splines: Example

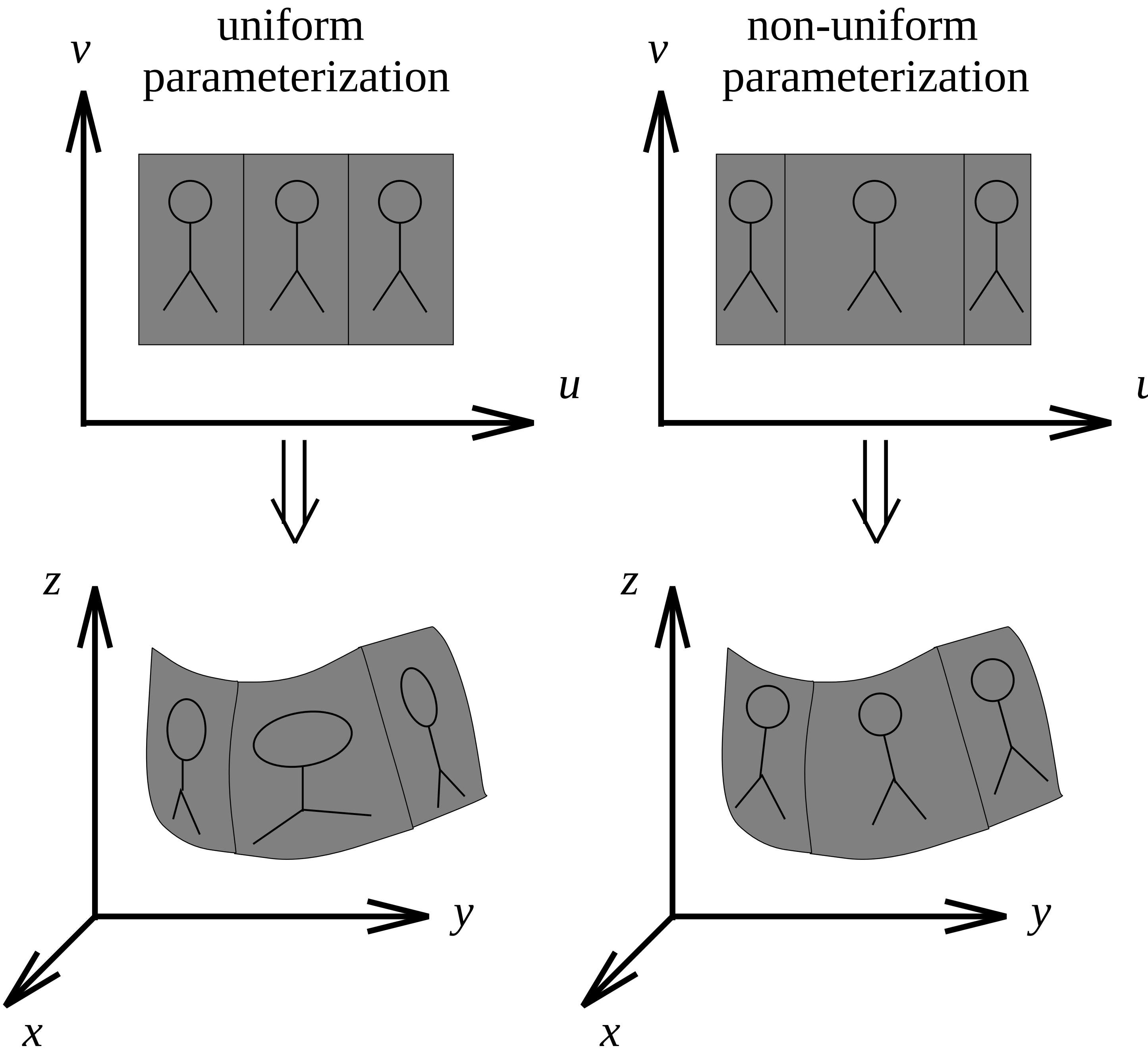


© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

This diagram outlines two adjacent bicubic patches meeting with C^1 continuity. As the lower diagram suggests, the domain rectangles of these patches are adjacent in the u direction. The upper diagram shows the Bézier points of the two patches, using solid disks for the first patch and hollow circles for the second. Also shown are the Bézier curves for the four rows of constant v; solid line style is used for the first patch, and dashed line style is used for the second patch. Finally, C^1 continuity is graphically verified for all four rows of adjacent Bézier curves. From the C^1 continuity of these curves, we conclude that the patches are C^1-continuous along the full extent of their shared boundary.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Tensor Product Splines: Knot Spacing



© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Non-uniform knot spacing is very important for surfaces. In the case of curves, we were primarily concerned with the shape of the curve. However, in the case of surfaces, we are very much interested in their parameterization, since the u and v parameters are frequently used as texture coordinates during texture mapping. Having control over the knot spacing allows the animator not only to tweak the shape of a surface, but, more importantly, to evenly apply a texture map on it. By an "even" application, we mean that large patches correspond to large domain rectangles and are thus covered by large segments of the texture map (as in the example on the right column of the slide); in other words, the texture does not undergo extreme stretching or compression p (as in the example on the left column of the slide).

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

NURBS Surfaces

Define

- either as 4D tensor product spline projected into 3D, or

- as tracer of 3D NURBS curve with control points defined by other 3D NURBS curves.

Tensor product splines are special case of NURBS surfaces.

Usefulness: can represent exactly

- quadric surfaces (cylinders, cones, spheres, ellipsoids, paraboloids, hyperboloids), and

- surfaces of revolution.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Strictly (mathematically) speaking, a tensor product spline is any surface defined by a tracing spline curve whose control points are defined by other curves of the same type. However, the term *tensor product spline* is most often reserved for surfaces produced by B-spline curves.

Quadrics are the 3D analogue of 2D conic sections. They are very common in solid modeling, as well as ray-tracing, since it's very easy to test for intersections against them, find normals, make inclusion/exclusion tests, etc.

An intuitive explanation for the fact that surfaces of revolution can only be described by NURBS surfaces — but not by tensor product surfaces — is that the circular symmetry of rotation requires the exact representation of conics (the circle, in particular).

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

References

- Böhm: sections 29, 34–36, 38.
- Seidel: pages 42–43.
- Ramshaw: section 15.
- Farin: pages 246-248, 259-262, 267–269, 271-277, 292, 309–313, 316–317.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/

Böhm, Seidel, and Ramshaw are in the course reader, while Farin is on reserve at the MCS library.

© 1995 Apostolos Lerios
Stanford University
http://graphics.stanford.edu/~tolis/