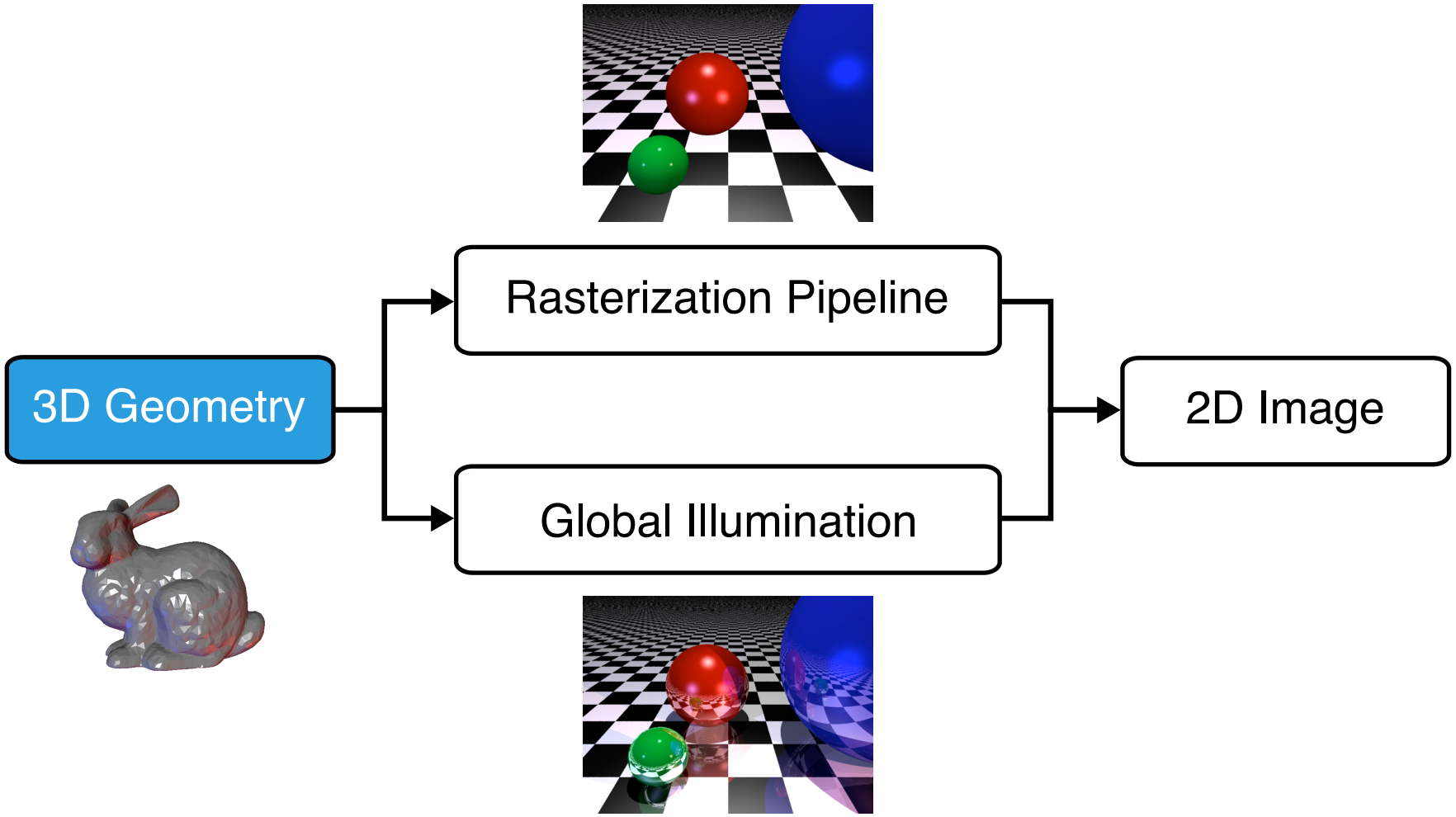
# Computer Graphics

## *Freeform Surfaces I*

Mark Pauly

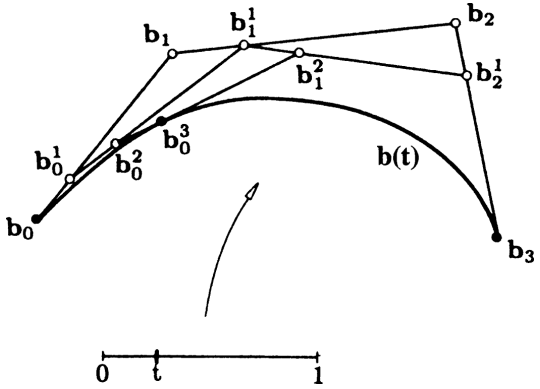
Geometric Computing Laboratory

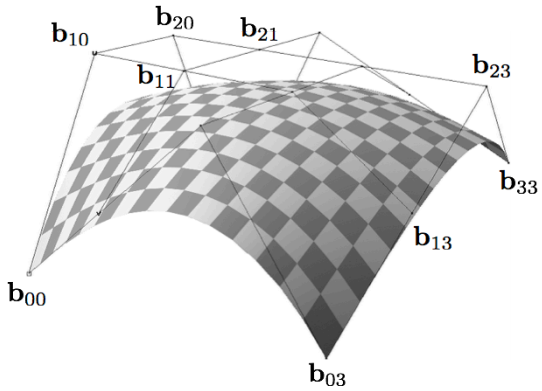
# Overview


Rasterization Pipeline
3D Geometry
2D Image
Global Illumination

# Overview

1. Freeform Curves
2. Freeform Surfaces
3. Freeform Deformation



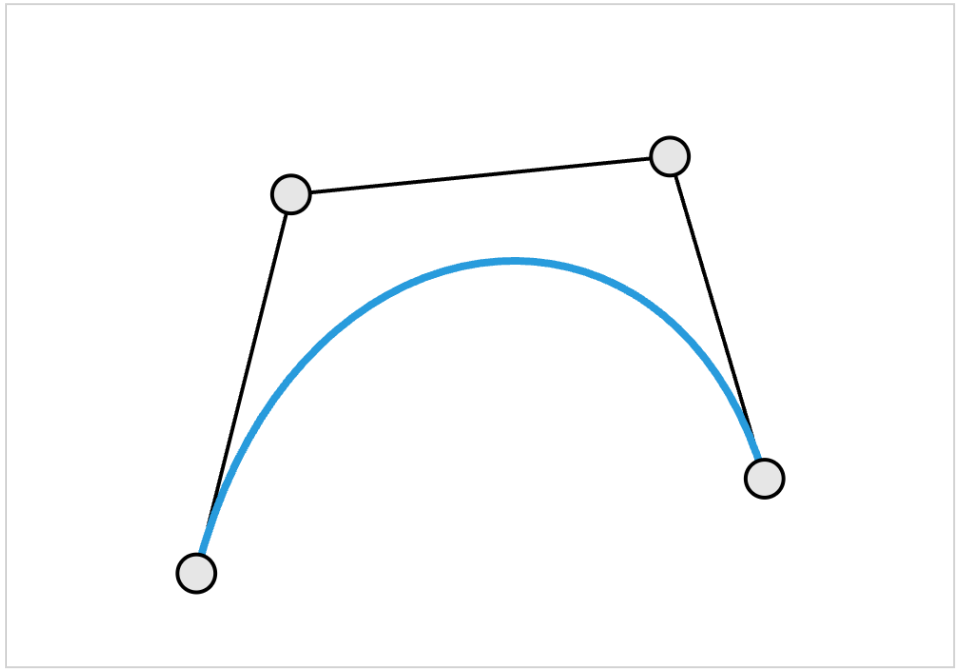



# Freeform Curves

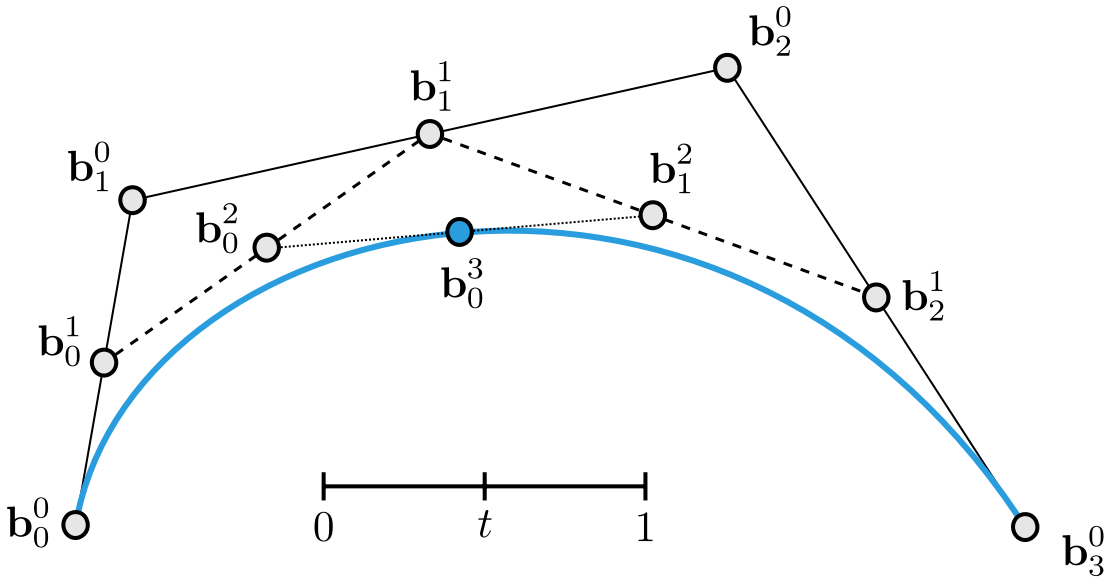
# Bezier Curves

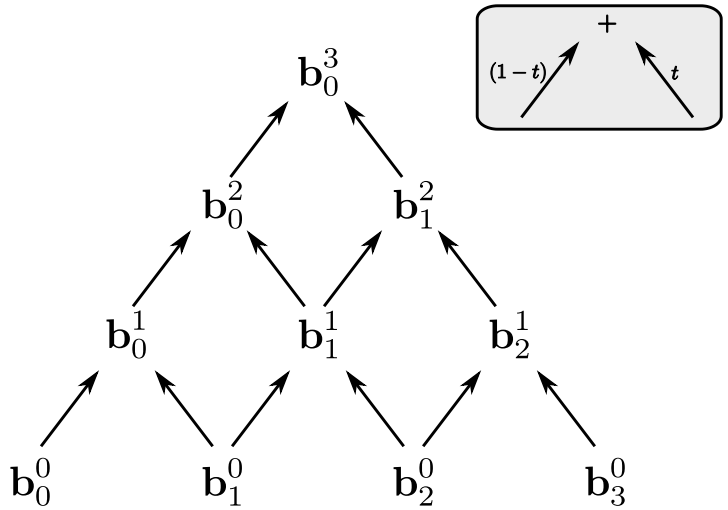
$$\mathbf{x}(t) = \sum_{i=0}^{n} \mathbf{b}_i \, B_i^n(t)$$

- Polynomial curve
- Affine combination
- Convex hull
- Pseudo-local control
- Endpoint interpolation

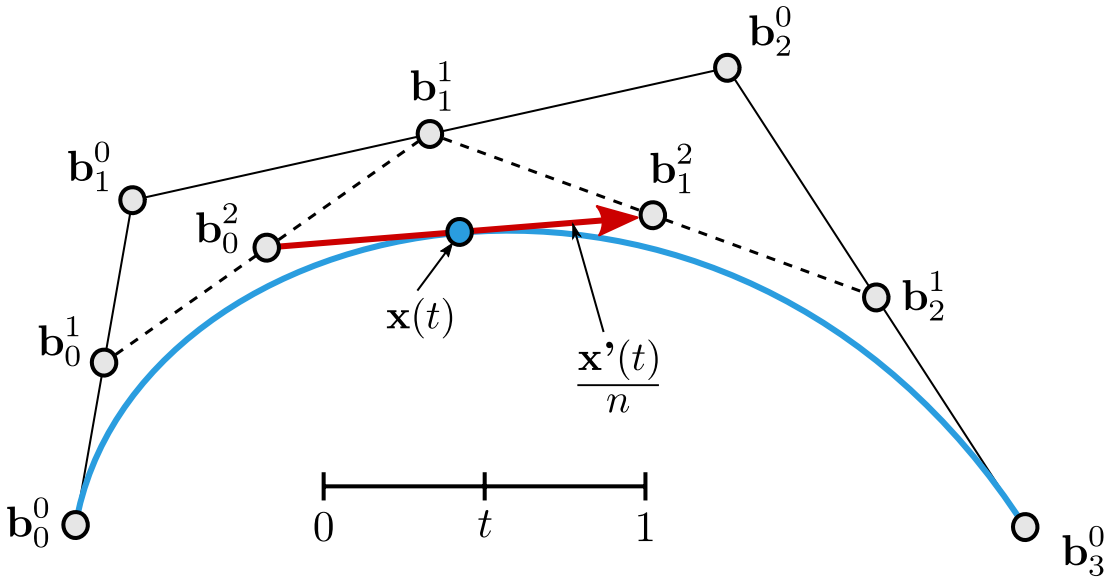


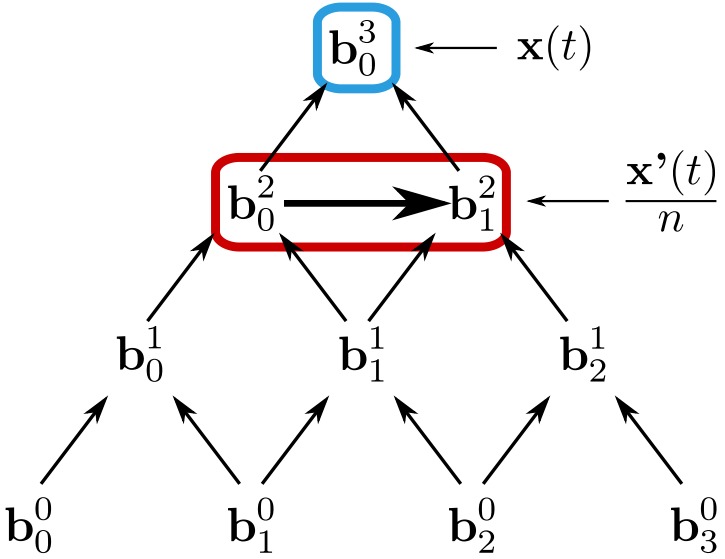
# de Casteljau Algorithm



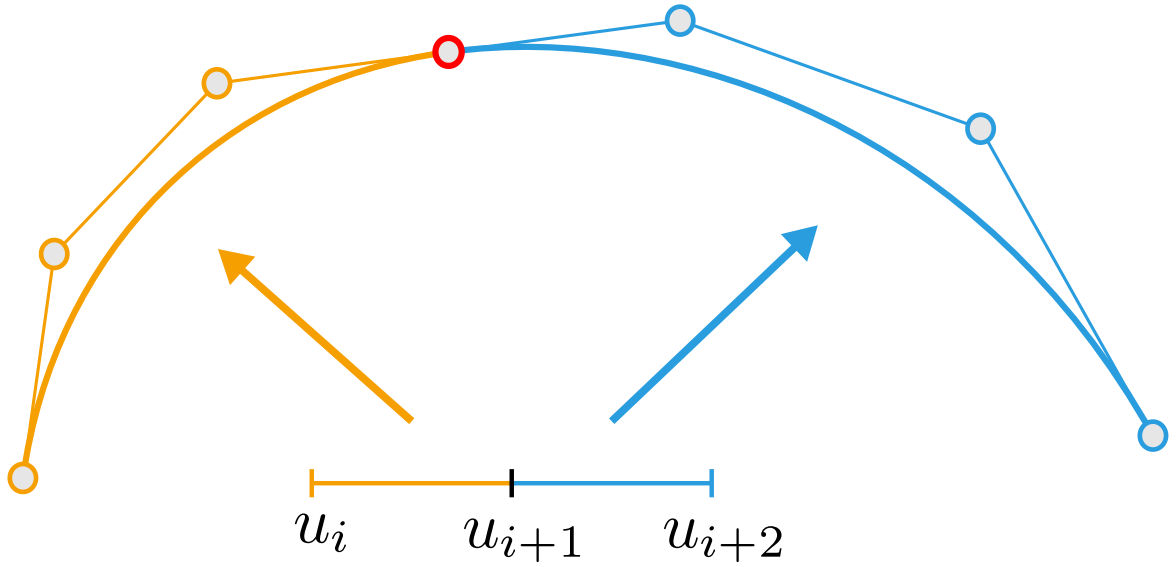



# Derivative of a Bezier Curve



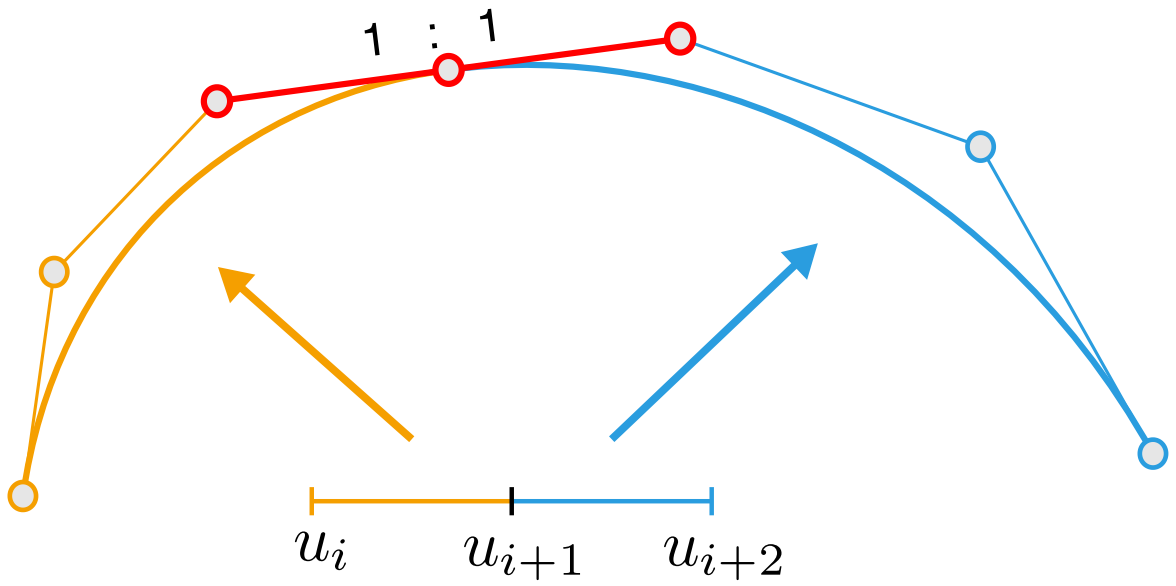



# Piecewise Bezier Curves



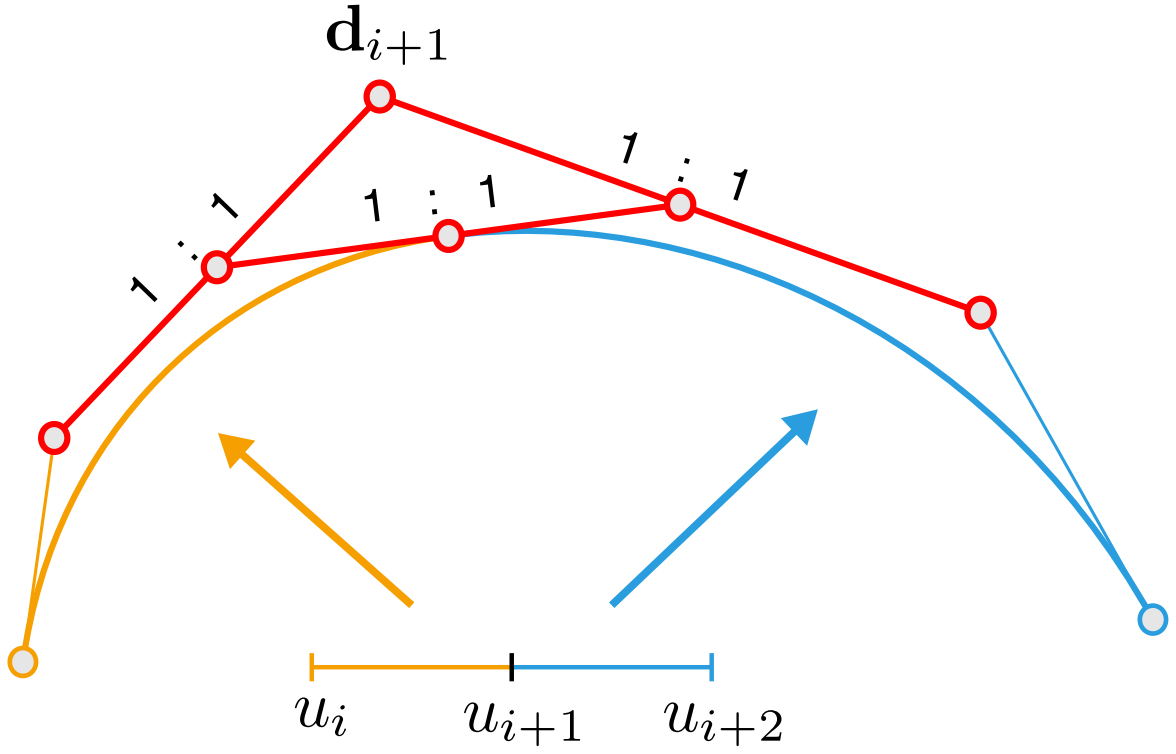

$C^0$ continuity: $\mathbf{a}_n = \mathbf{b}_0$

# Piecewise Bezier Curves




$C^1$ continuity: $\mathbf{a}_n = \mathbf{b}_0 = \frac{1}{2}(\mathbf{a}_{n-1} + \mathbf{b}_1)$

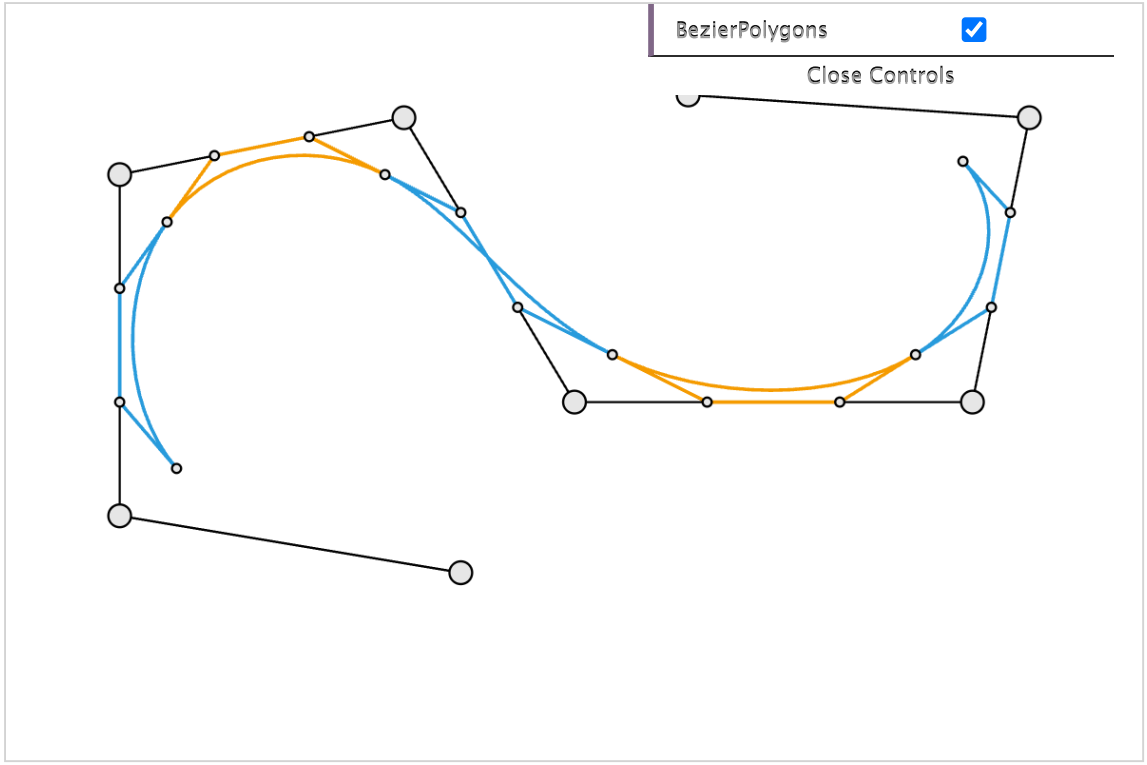
# Piecewise Bezier Curves




$C^2$ continuity: *A-frame* construction

# Bezier Curves vs. B-Spline Curves

- Disadvantages of Bezier curves
  - Adding more DoFs raises degree
  - High-degree polynomials tend to oscillate
  - Only pseudo-local control
- Splines overcome these problems
  - Low-degree curve segments avoid oscillations
  - Connect segments with $C^{n-1}$ smoothness
  - Allows for local modifications
  - No more endpoint interpolation



# Freeform Surfaces

# Parametric Surfaces

- Parametric surface $\mathbf{x}(u, v)$:

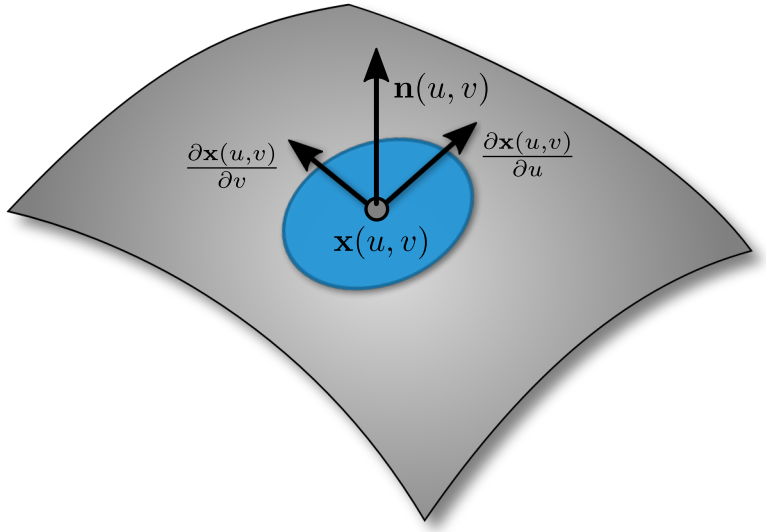
$$\mathbf{x}(u, v) \;=\; (x(u, v),\, y(u, v),\, z(u, v))^{\mathsf{T}}$$

- Tangents in $u$ and $v$ direction

$$\frac{\partial}{\partial u}\mathbf{x}(u, v)\,, \qquad \frac{\partial}{\partial v}\mathbf{x}(u, v)$$

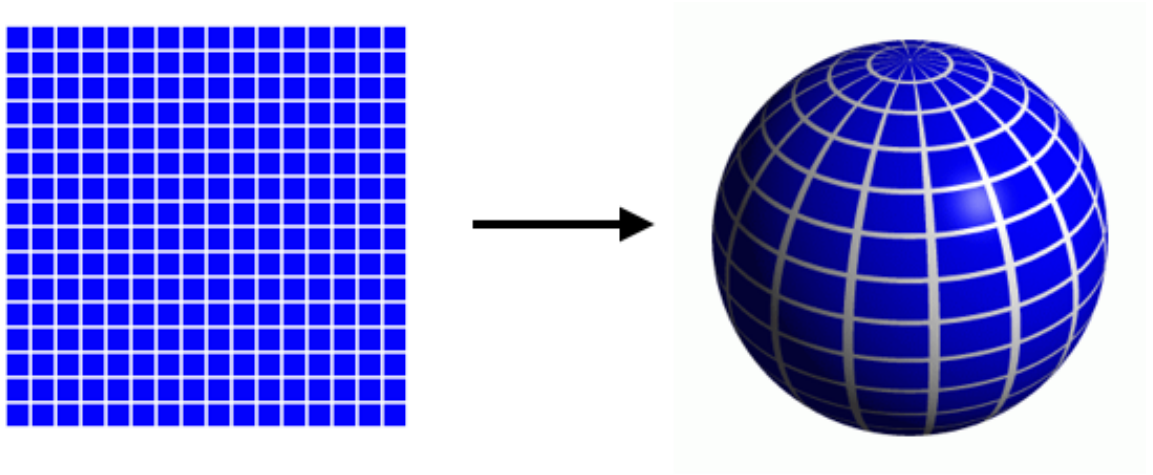
- Normal vector as the normalized cross-product of the two tangent vectors

$$\mathbf{n}(u, v) = \frac{\frac{\partial}{\partial u}\mathbf{x}(u, v) \times \frac{\partial}{\partial v}\mathbf{x}(u, v)}{\left\| \frac{\partial}{\partial u}\mathbf{x}(u, v) \times \frac{\partial}{\partial v}\mathbf{x}(u, v) \right\|}$$



# Example: Sphere

$$\mathbf{f}\colon [0, 2\pi] \times [0, \pi] \to \mathbb{R}^3 \quad \text{mit} \quad \mathbf{f}(u, v) = \begin{pmatrix} r\cos(u)\sin(v) \\ r\sin(u)\sin(v) \\ r\cos(v) \end{pmatrix}$$

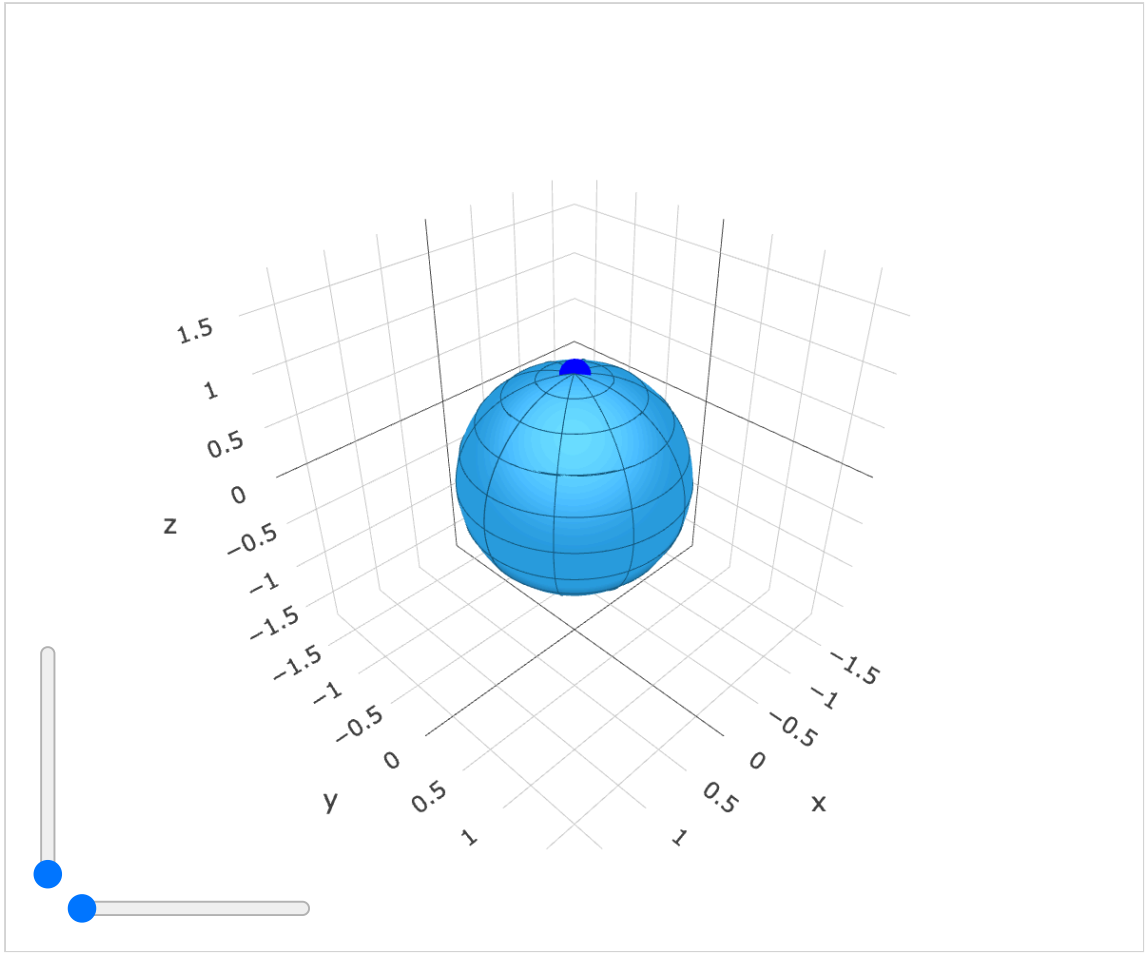


# Example: Sphere

$$\mathbf{f}(u,v) = \begin{pmatrix} \cos(u)\sin(v) \\ \sin(u)\sin(v) \\ \cos(v) \end{pmatrix}$$

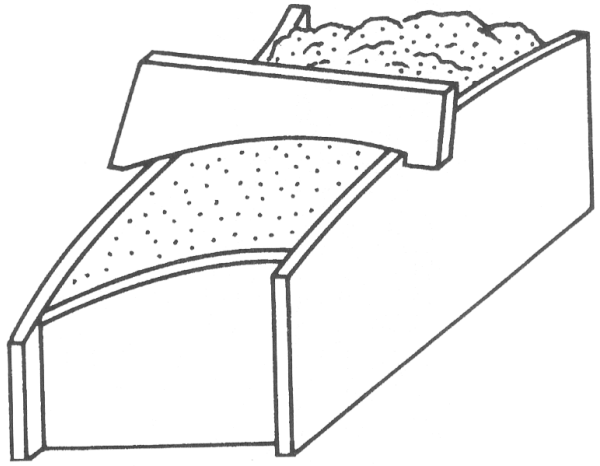
$$\frac{\partial \mathbf{f}}{\partial u}(u,v) = \begin{pmatrix} -\sin(u)\sin(v) \\ \cos(u)\sin(v) \\ 0 \end{pmatrix}$$

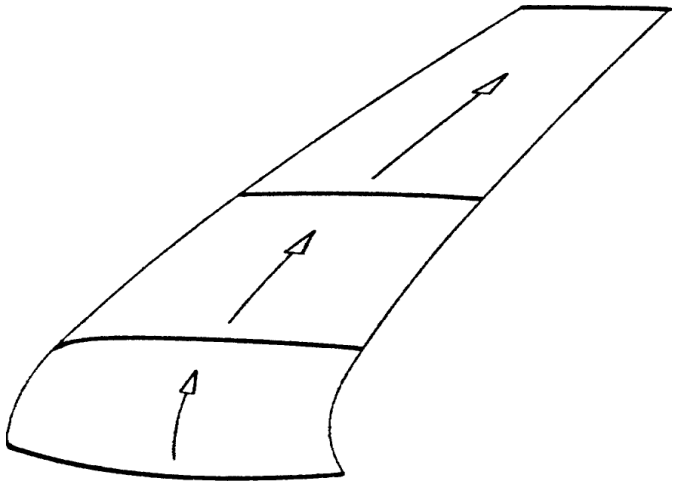
$$\frac{\partial \mathbf{f}}{\partial v}(u,v) = \begin{pmatrix} \cos(u)\cos(v) \\ \sin(u)\cos(v) \\ -\sin(v) \end{pmatrix}$$



# Tensor Product Surfaces

- Surface is defined by a curve moving through space
- The curve itself might deform as it moves



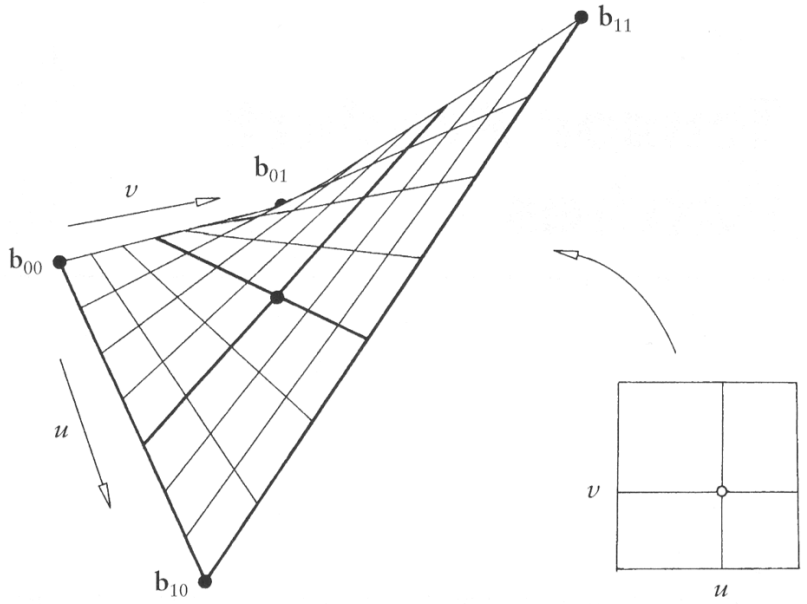


# Bilinear Interpolation

- Simplest case is a bilinear patch:

$$\mathbf{x}(u,v) = \begin{pmatrix} 1-u, & u \end{pmatrix} \begin{pmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} \\ \mathbf{b}_{10} & \mathbf{b}_{11} \end{pmatrix} \begin{pmatrix} 1-v \\ v \end{pmatrix}$$

- Isoparametric curves are straight lines



# Bilinear Patch

```
# four control points
b00 = vector([0.0,  0.0,  0.5])
b01 = vector([0.0,  1.0,  0.0])
b10 = vector([1.0, -0.2, -0.5])
b11 = vector([1.0,  1.2,  0.7])

# function for bilinear interpolation
var('u,v')
def x(u,v):
    return (1-v)*((1-u)*b00 + u*b01) + v*((1-u)*b10 + u*b11)

# plot it
patch = parametric_plot3d( x(u,v), (u,0,1), (v,0,1), color='lightgrey', viewer='threejs')

# add isolines
@interact
def _(U=(0.0,1.0), V=(0.0, 1.0)):
    iso_v = parametric_plot3d( x(u,V), (u,0,1), color='red',   thickness=10, viewer='threejs'
    iso_u = parametric_plot3d( x(U,v), (v,0,1), color='green', thickness=10, viewer='threejs'
    show( patch + iso_u + iso_v )
```

Evaluate

Language: Sage

# Tensor Product Bezier Surfaces

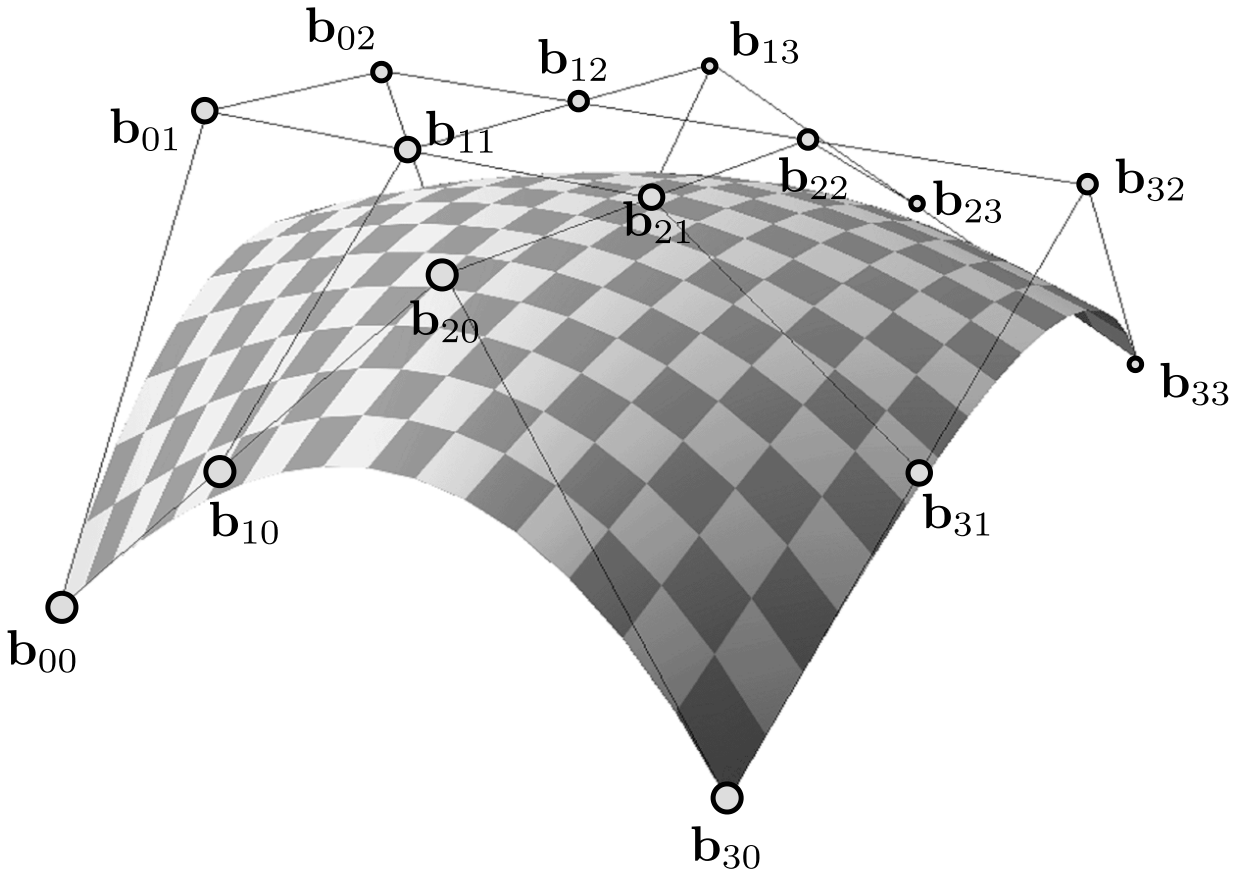
- Let a Bezier curve be given as $\mathbf{x}(u) = \sum_{i=0}^{m} \mathbf{b}_i B_i^m(u)$
- Control points $\mathbf{b}_i$ move on Bezier curves $\mathbf{b}_i(v)$

$$\mathbf{b}_i(v) = \sum_{j=0}^{n} \mathbf{b}_{i,j} B_j^n(v)$$

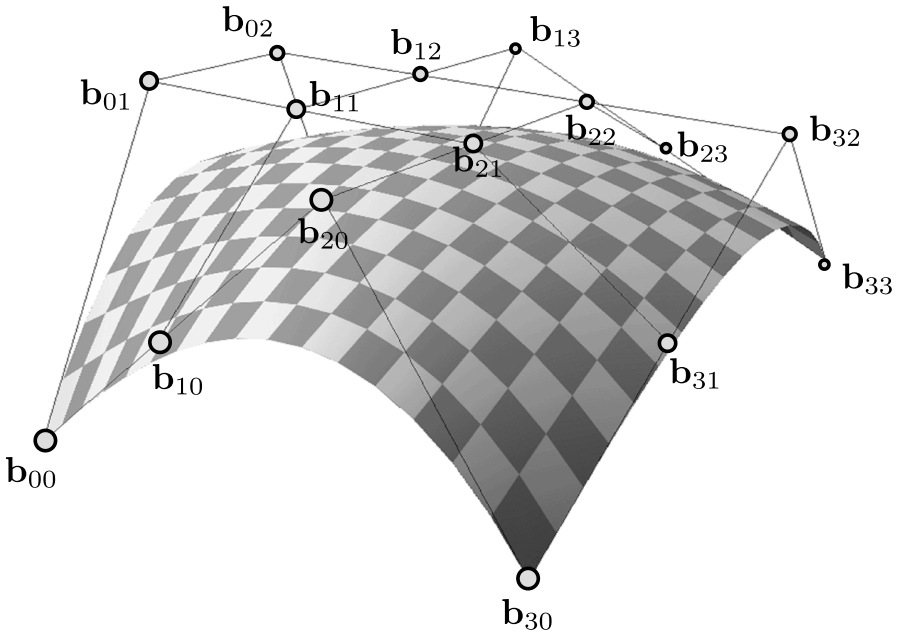
- This defines a tensor product Bezier patch

$$\mathbf{x}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{b}_{i,j} B_i^m(u) B_j^n(v)$$

# Example: Bicubic Bezier Patch



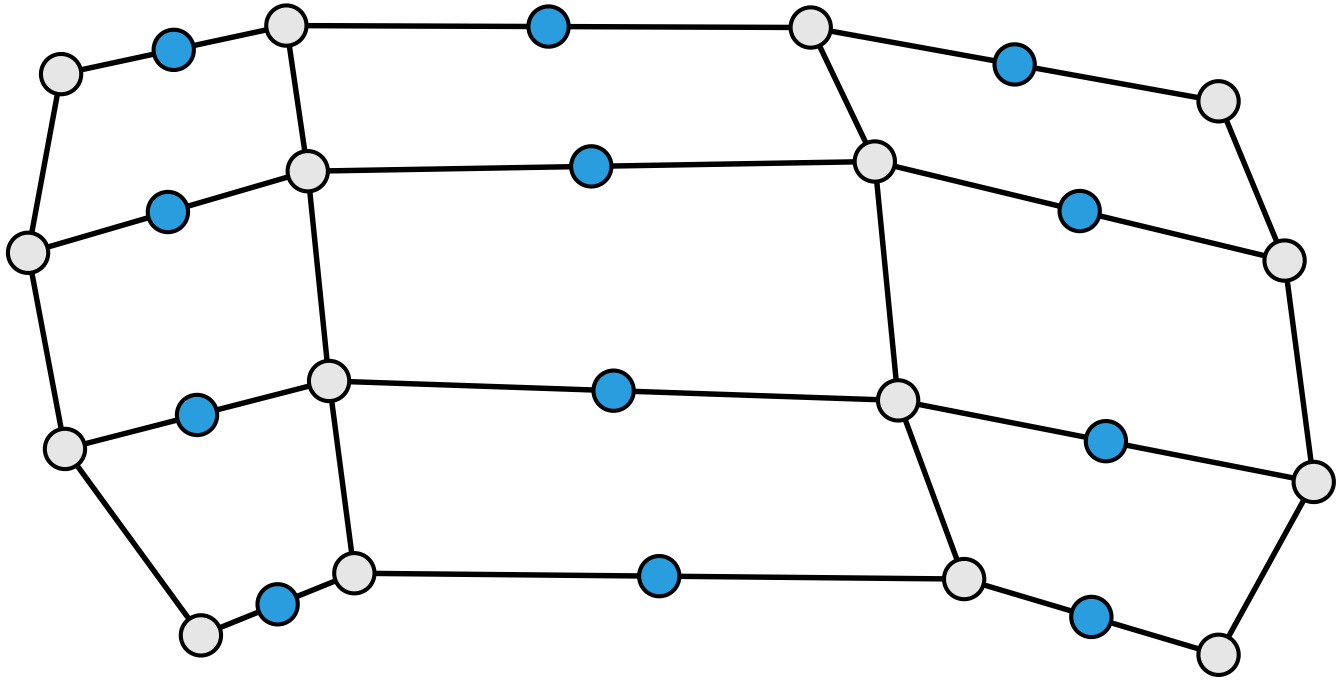
# Properties of Bezier Patches




- $u$- and $v$-isolines are Bezier curves on the surface
- $\mathbf{x}(u, v)$ is affine combination of control points
- Surface lies in convex hull of control points
- Surface interpolates corners of control grid
- Surface interpolates boundary curves

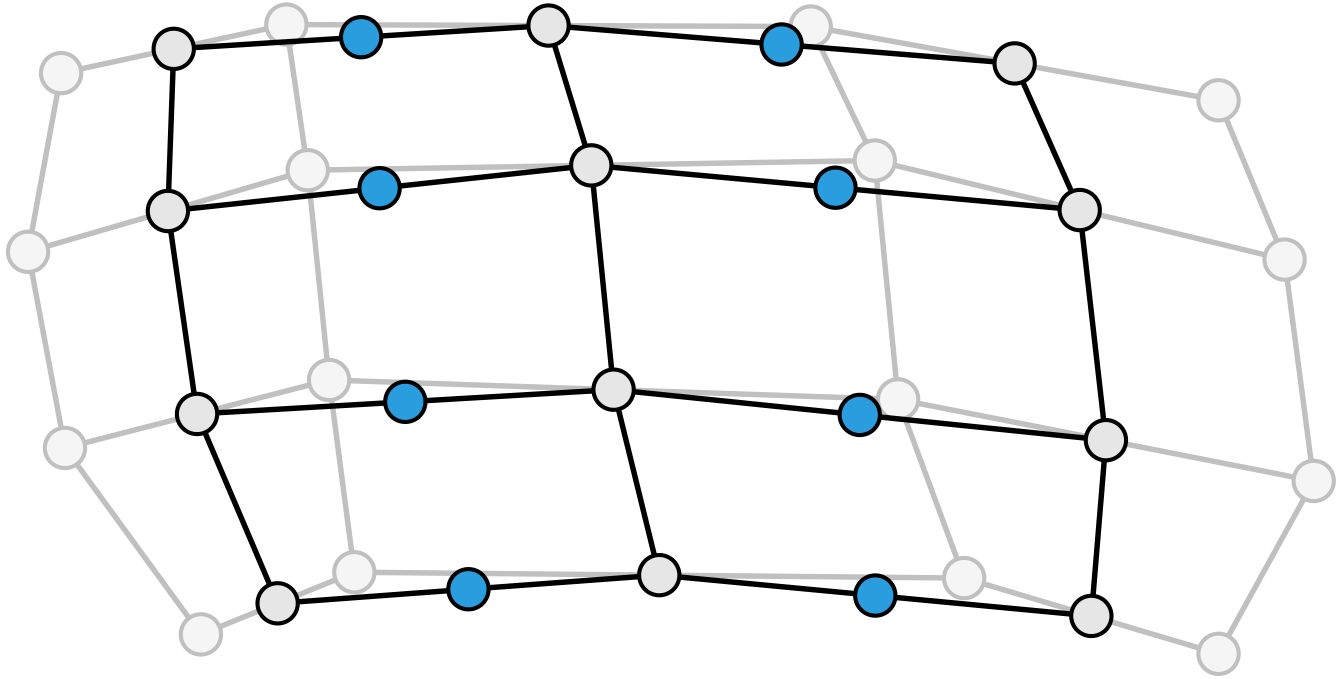
# de Casteljau Algorithm

- Apply de Casteljau with parameter $u$ for each row of control points.



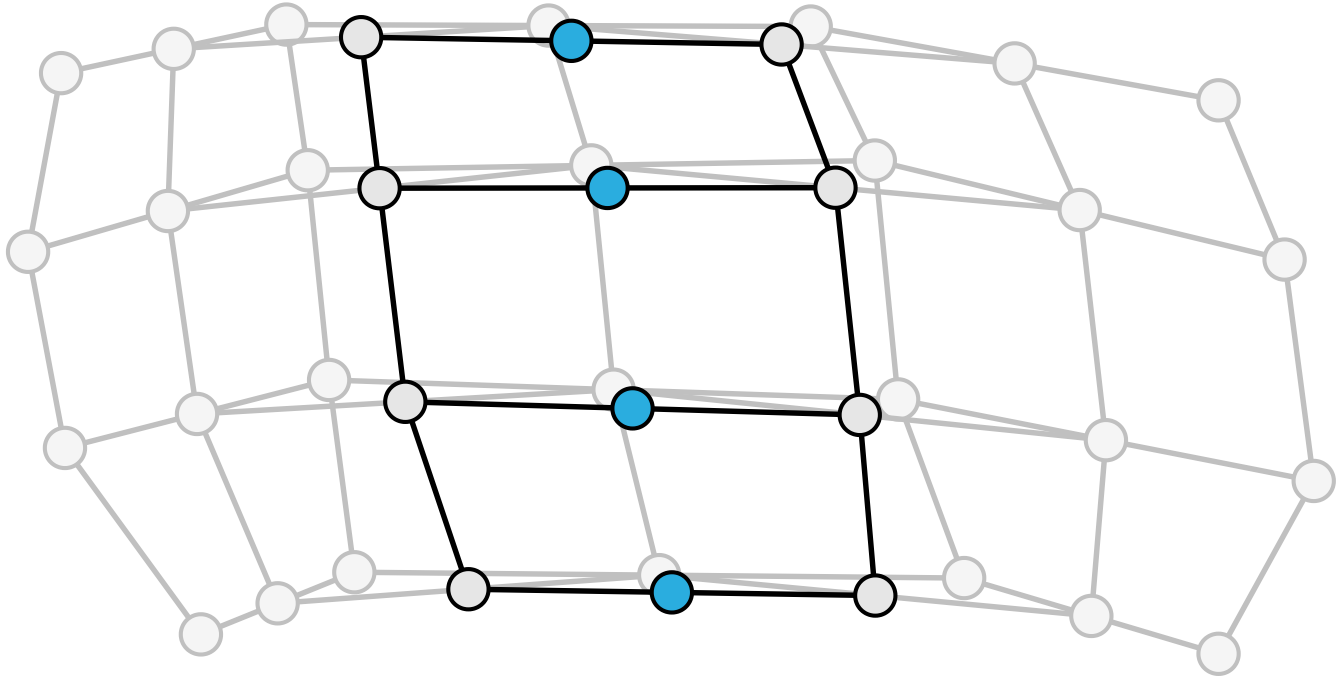
# de Casteljau Algorithm

- Apply de Casteljau with parameter $u$ for each row of control points.



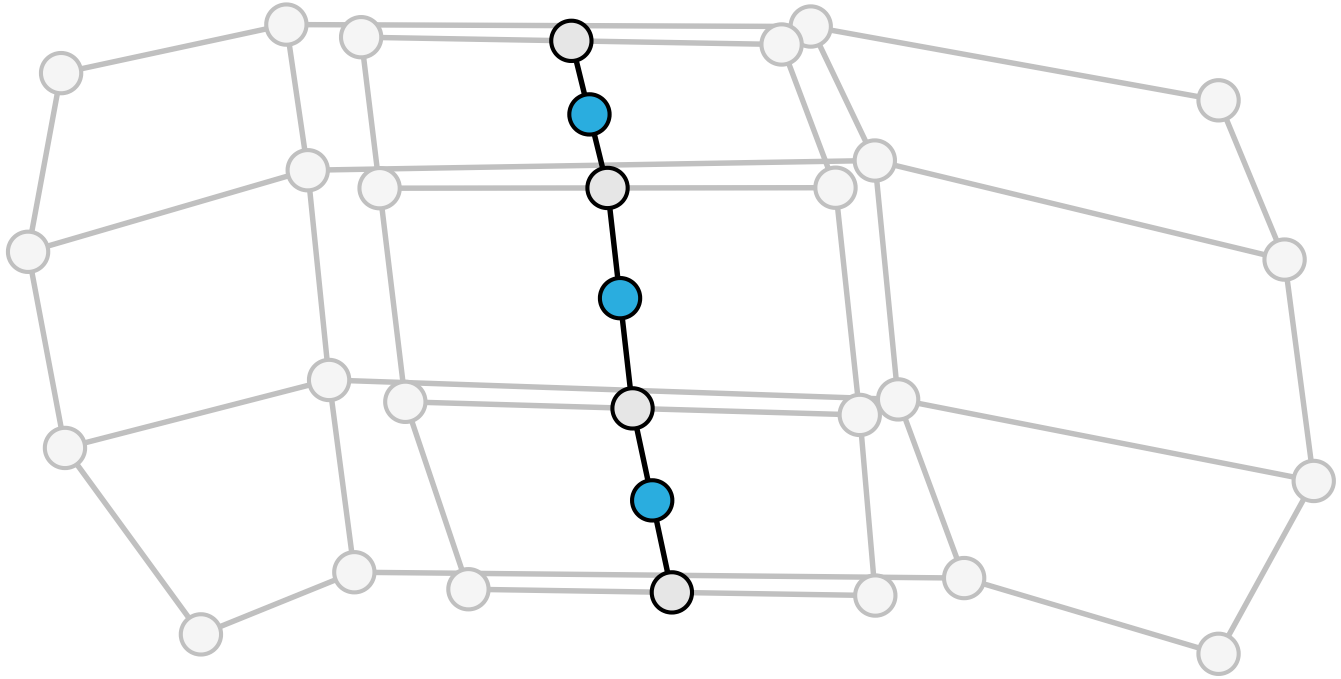
# de Casteljau Algorithm

- Apply de Casteljau with parameter $u$ for each row of control points.



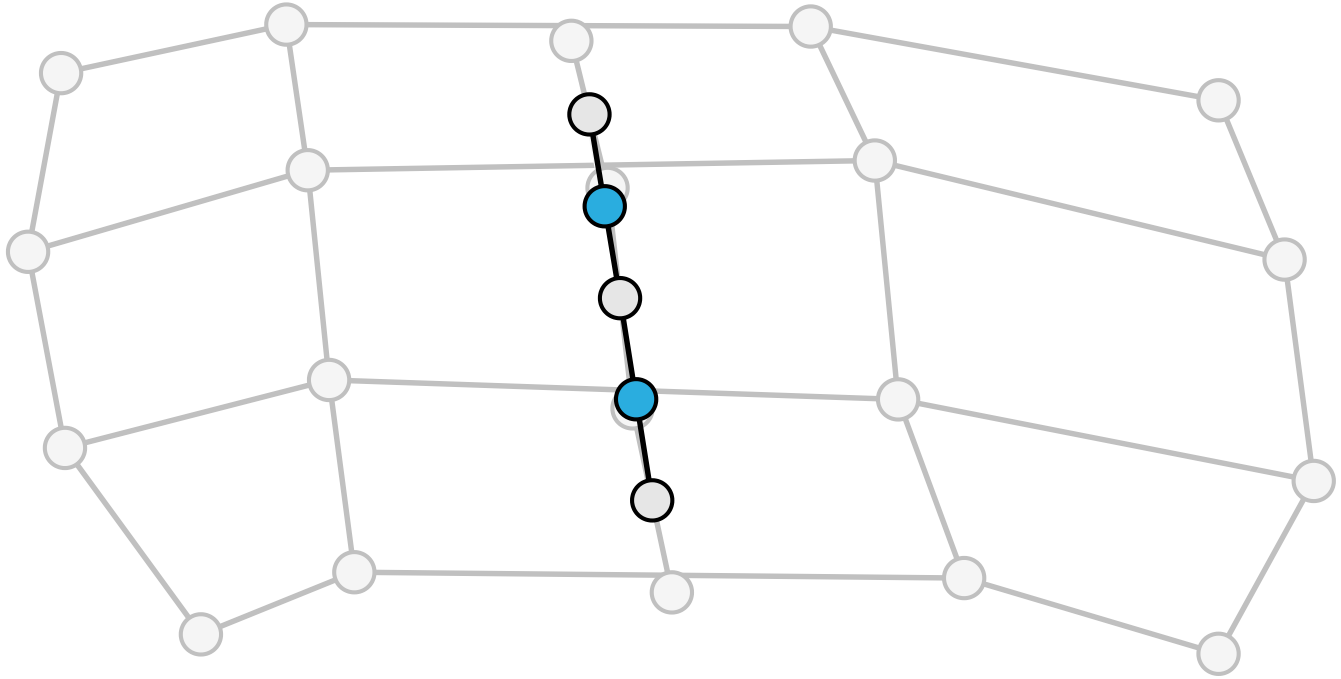
# de Casteljau Algorithm

- Apply de Casteljau with parameter $v$ to remaining column.



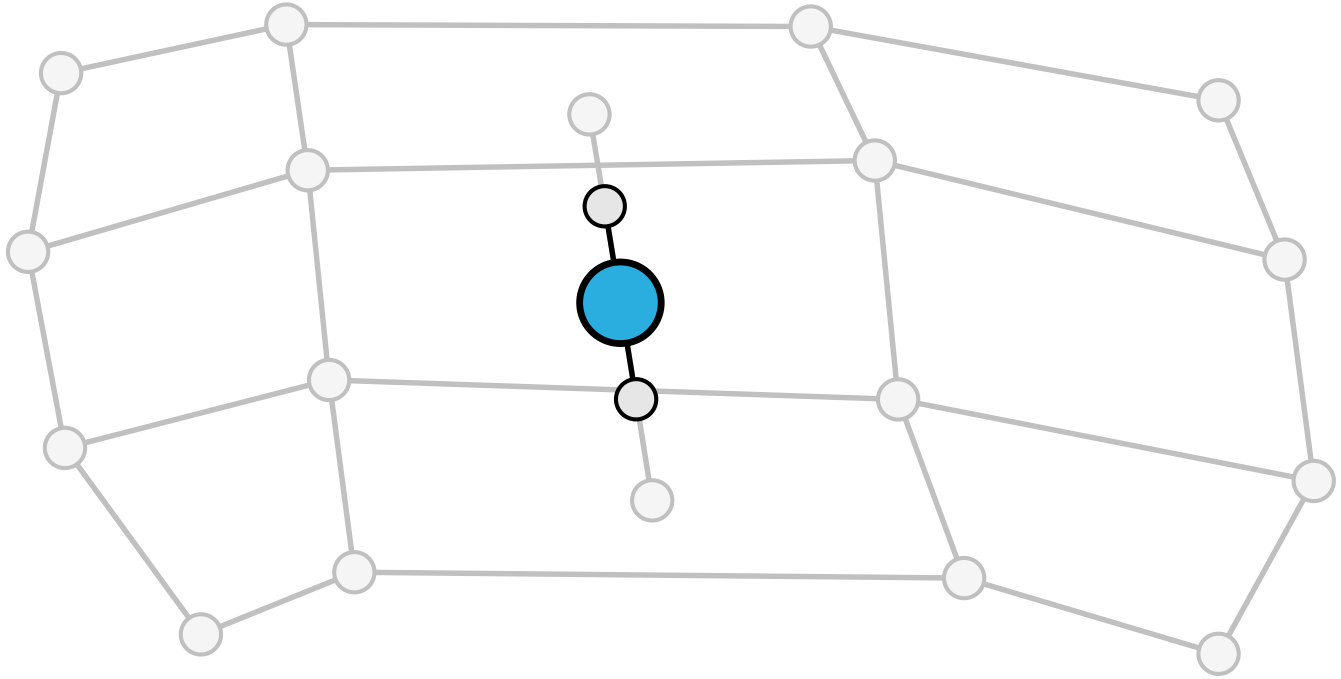
# de Casteljau Algorithm

- Apply de Casteljau with parameter $v$ to remaining column.



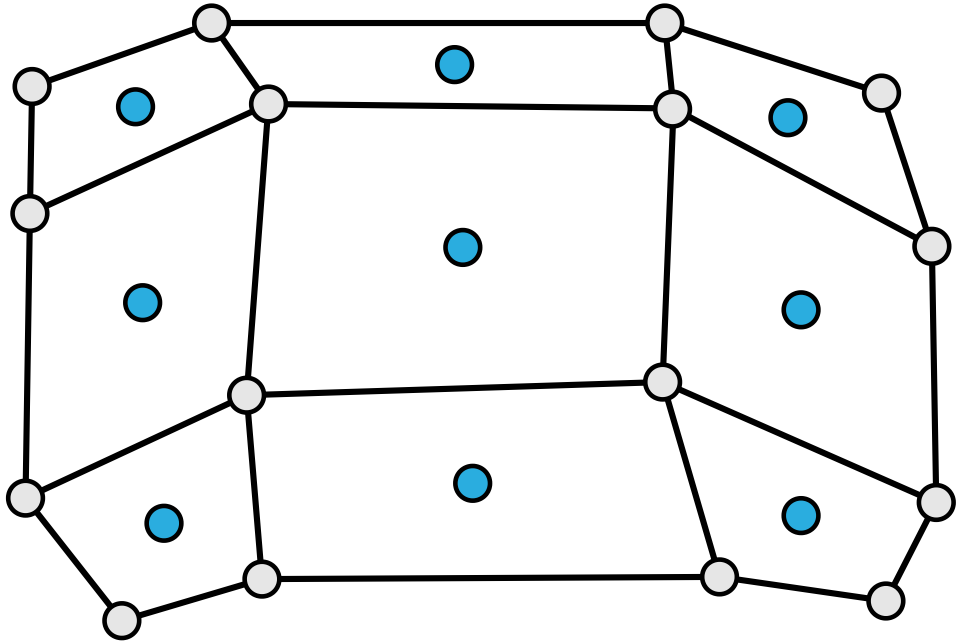
# de Casteljau Algorithm

- Apply de Casteljau with parameter $v$ to remaining column.



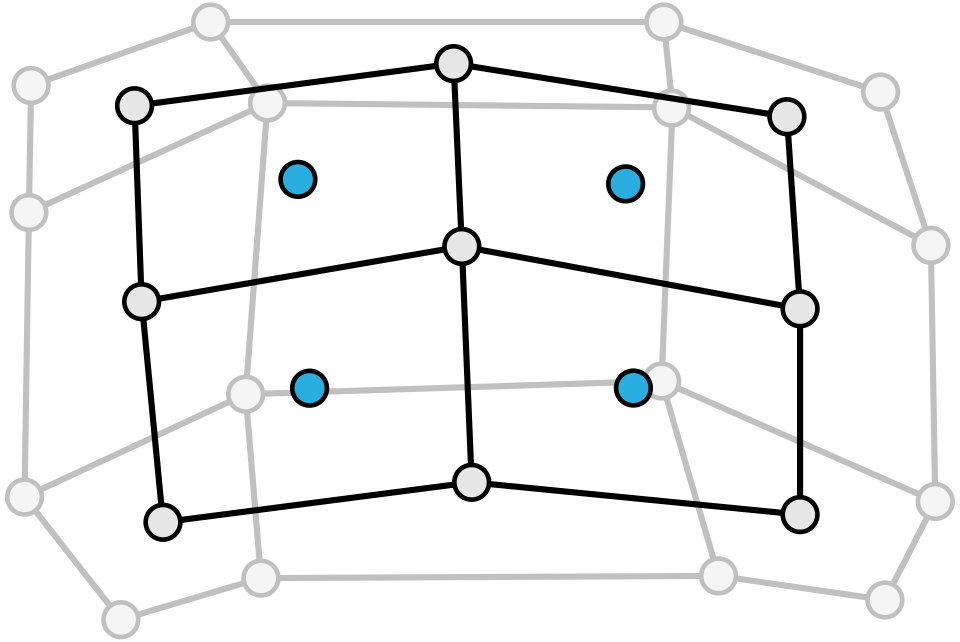
# de Casteljau Algorithm for $n = m$

- Construct 2D pyramid using repeated bilinear interpolation.



# de Casteljau Algorithm for $n = m$

- Construct 2D pyramid using repeated bilinear interpolation.



# de Casteljau Algorithm for $n = m$

- Construct 2D pyramid using repeated bilinear interpolation.