CMT107 Visual Computing

VII.2 Freeform Surfaces

Xianfang Sun

School of Computer Science & Informatics
Cardiff University

Overview

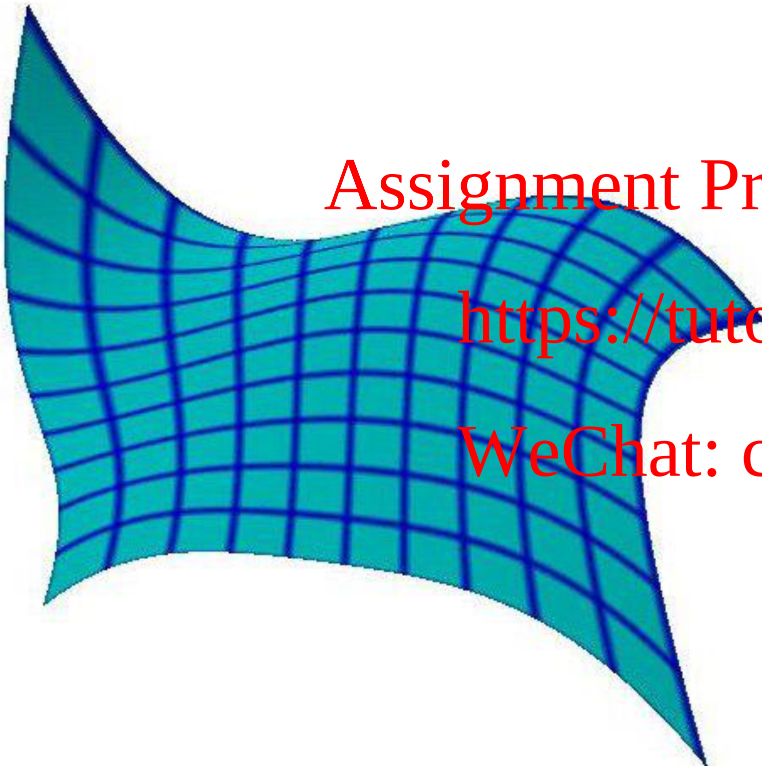
- Surface representations
- Parametric surfaces
- Piecewise polynomial surfaces
  - Tensor product splines
- Subdivision surfaces
  - Loop subdivision
  - Doo-Sabin subdivision
  - Catmull-Clark subdivision

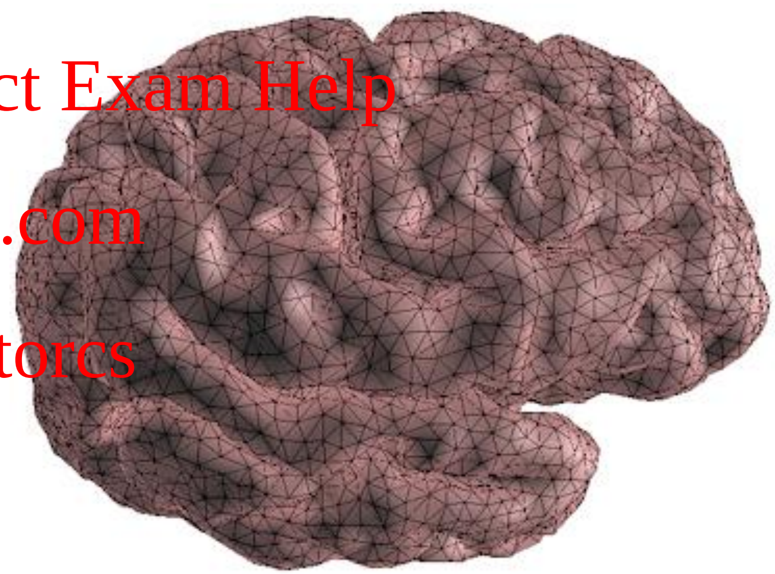
Surfaces

- We require general surface shapes (something better than polygonal meshes)
  - *Exact* boundary representation for some objects
  - *Create*, *edit* and *analyse* shapes

Explicit Surfaces

- A surface is a set of positions of a point moving with two degrees of freedom





- Explicit and implicit representation similar to curve
  - Explicit: $z = f(x, y)$ for $(x, y) \in R^2$

Implicit Surfaces

- Surface defined as solution of an equation system:

$$f(x, y, z) = 0$$

  - Usually one equation in 3D
- Example: linear equation (plane)

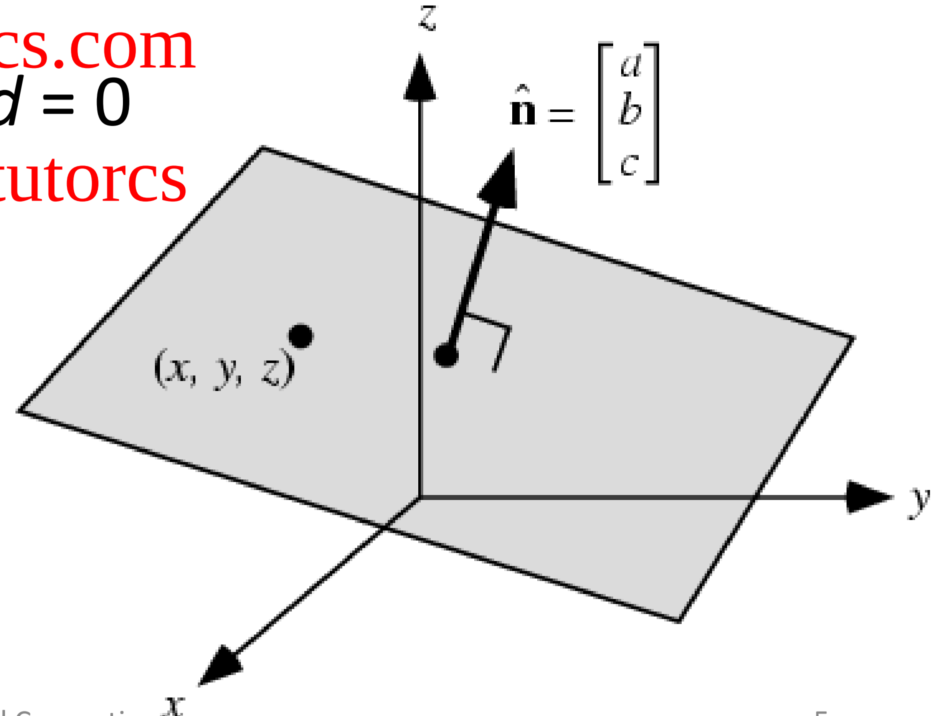
$$ax + by + cz + d = 0$$

  - Using vectors:

$$n^T\boldsymbol{x} + d = 0$$

  — n: unit plane normal

  — d: distance from origin



Implicit Quadrics

- Quadrics (quadratic surfaces)

$$ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + gx + hy + jz + k = 0$$

- Matrix representation:

$$\boldsymbol{x}^{\mathrm{T}} M \boldsymbol{x} + v^{\mathrm{T}} \boldsymbol{x} + s = 0$$

- Sphere / Ellipsoid:

$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} + \frac{z^2}{r_z^2} - 1 = 0$$

- Cylinder (elliptic):

$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} - 1 = 0$$

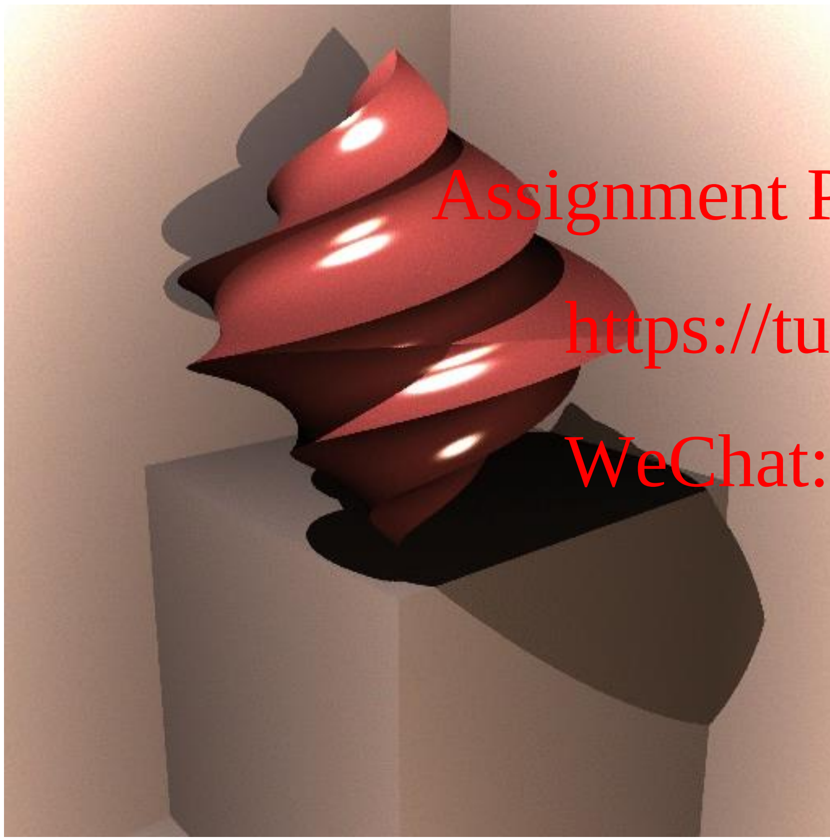
- Cone (elliptic):

$$\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} - z^2 = 0$$

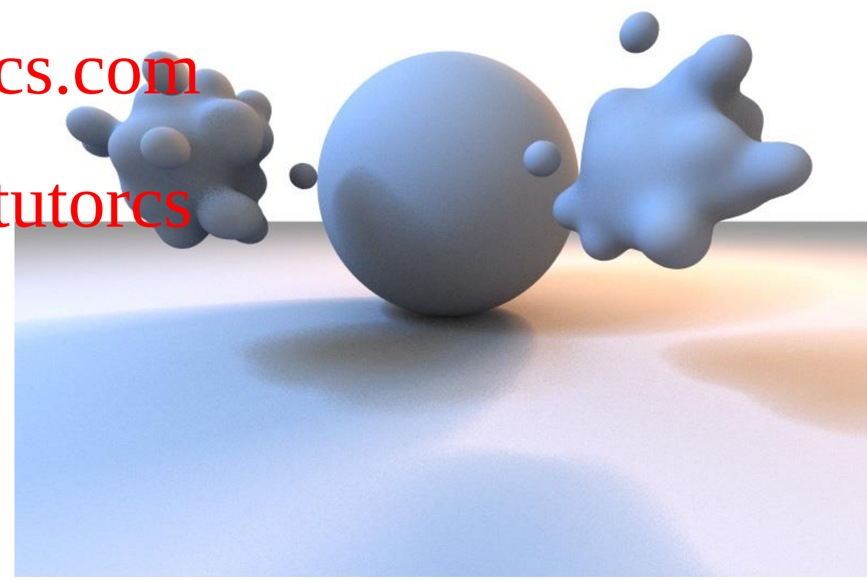
Properties of Implicit Surfaces

- Simple to test if point is on surface
- Simple to intersect two surfaces

- Hard to render
- Hard to describe complex shapes



Mathematical Functions / Sets



Blobby Models

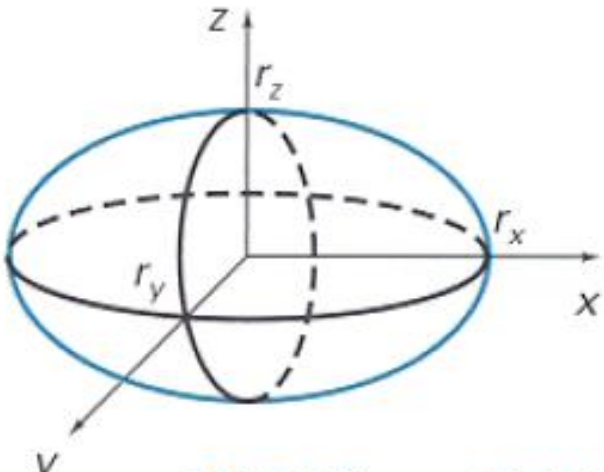
Parametric Surfaces

- Describe points on surface by *parametric functions*

$$\mathbf{s}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}$$

  - Maps 2D (u,v) parameter domain to 3D (x,y,z) model space

- Example: ellipsoid
  - $x(u, v) = r_x \cos u \cos v$
    $y(u, v) = r_y \cos u \sin v$
    $z(u, v) = r_z \sin u$
    $(u, v) \in [-\pi/2, \pi/2] \times [0, 2\pi]$



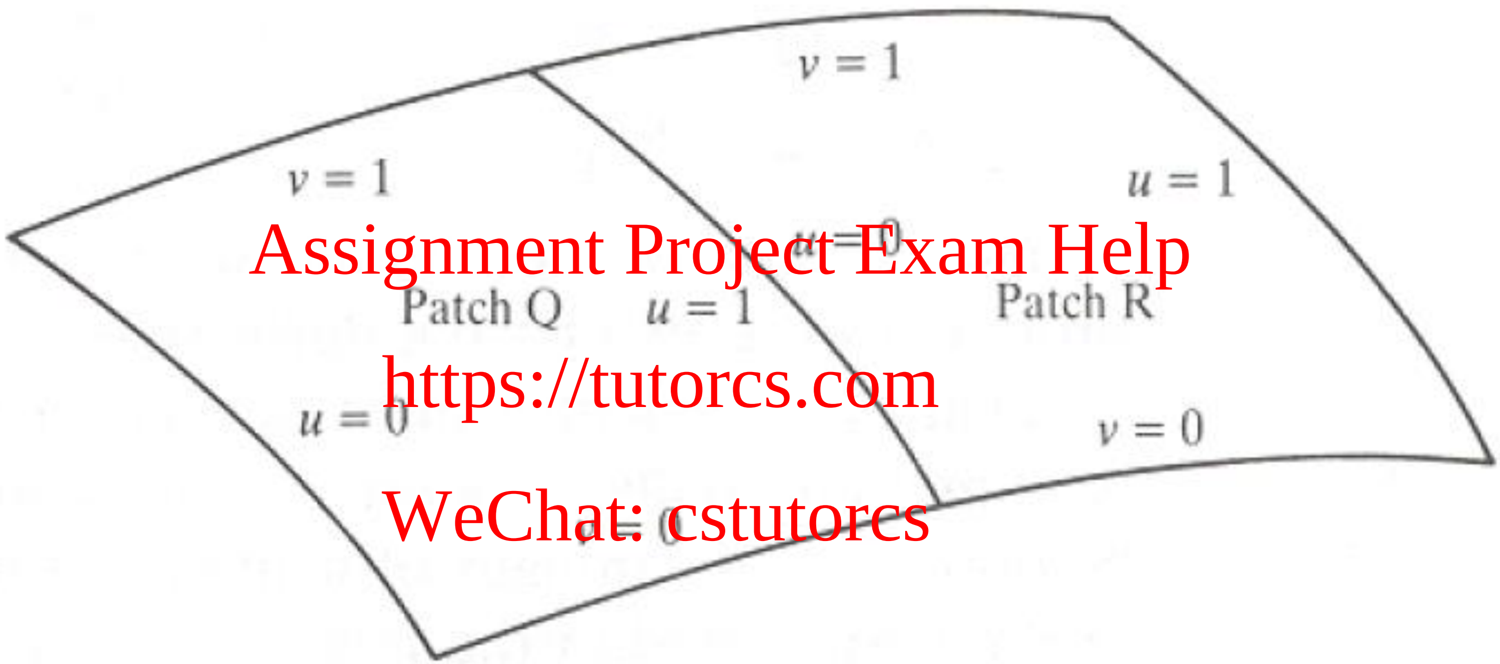
H&B Figure 10.10

Properties of Parametric Surfaces

- *Properties* similar to parametric curves
  - Simple to render points
  - Hard to test if point is on surface, compute intersections, etc.
- Hard to represent whole surface by single polynomial function
  - Use *piecewise polynomial surfaces*
  - Surface is cut into *patches*
  - *Smoothness* / *continuity* problem when joining patches

Piecewise Polynomial Surface

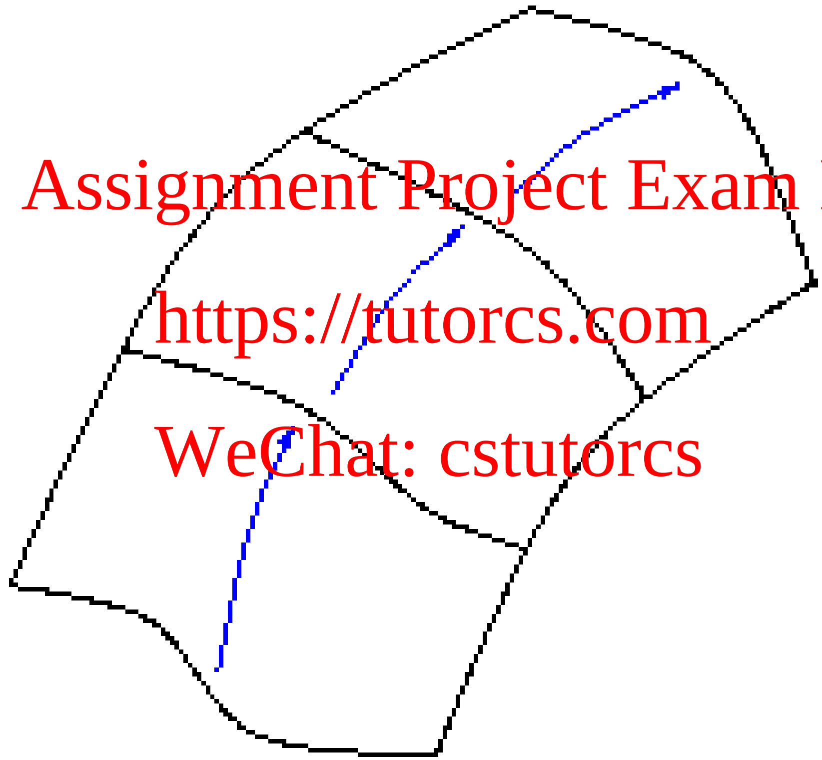
- *Spline surface*: piecewise polynomial surface patches



- Use spline curve approach with two degrees of freedom
  - Each patch is defined by a set of *control points*

Tensor Product

- Intuitively, a surface is a curve which *moves* through space while it changes its shape



- Mathematically this is the *tensor product* of two curves

Tensor Product Surfaces

- Surface patch as a curve moving through space
  - Assume this curve is at any time $v \in [0, 1]$ a Bézier curve

$$c^{v}(u) = \sum_{l=0}^{\alpha} P_l(v) B_l^{\alpha}(u)$$

  - The control points $P_l(v)$ lie on curves as well, assume these are also Bézier curves

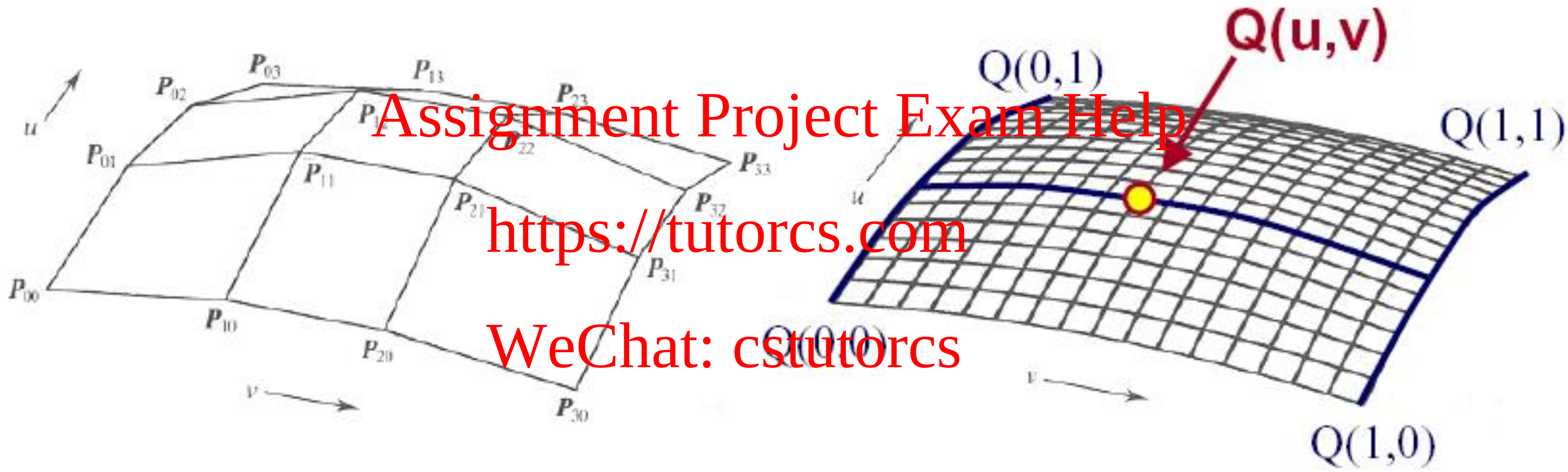
$$P_l(v) = \sum_{k=0}^{\beta} P_{l,k} B_k^{\beta}(v)$$

- Combing both gives the formula for a *Bézier surface patch*

$$Q(u, v) = \sum_{l=0}^{\alpha} \sum_{k=0}^{\beta} P_{l,k} B_l^{\alpha}(u) B_k^{\beta}(v)$$

Bézier Surface Patches

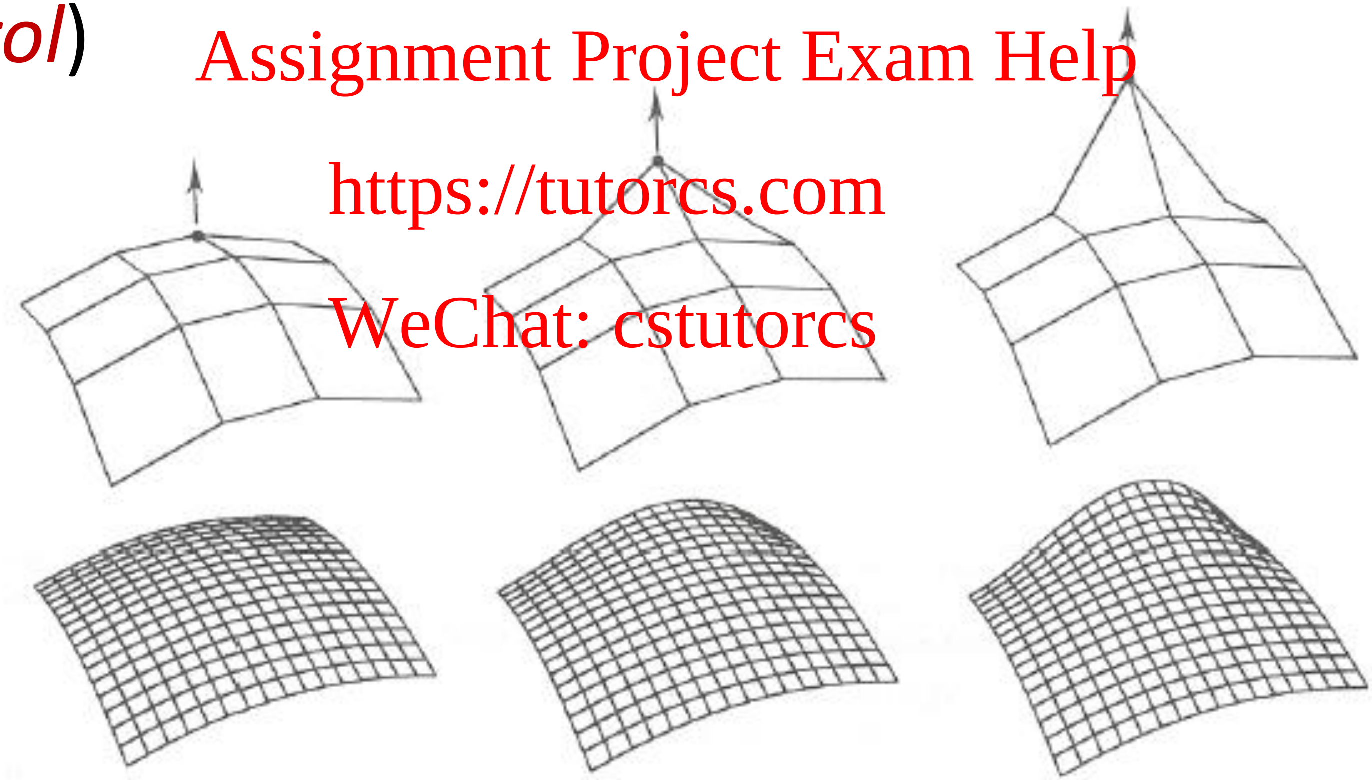
➢ Point $Q(u, v)$ on the patch is the tensor product of Bézier curves defined by the *control points* $P_{l,k}$



- *Order* of surface is given by order of curves α, β (e.g. bicubic: $\alpha = \beta = 3$)

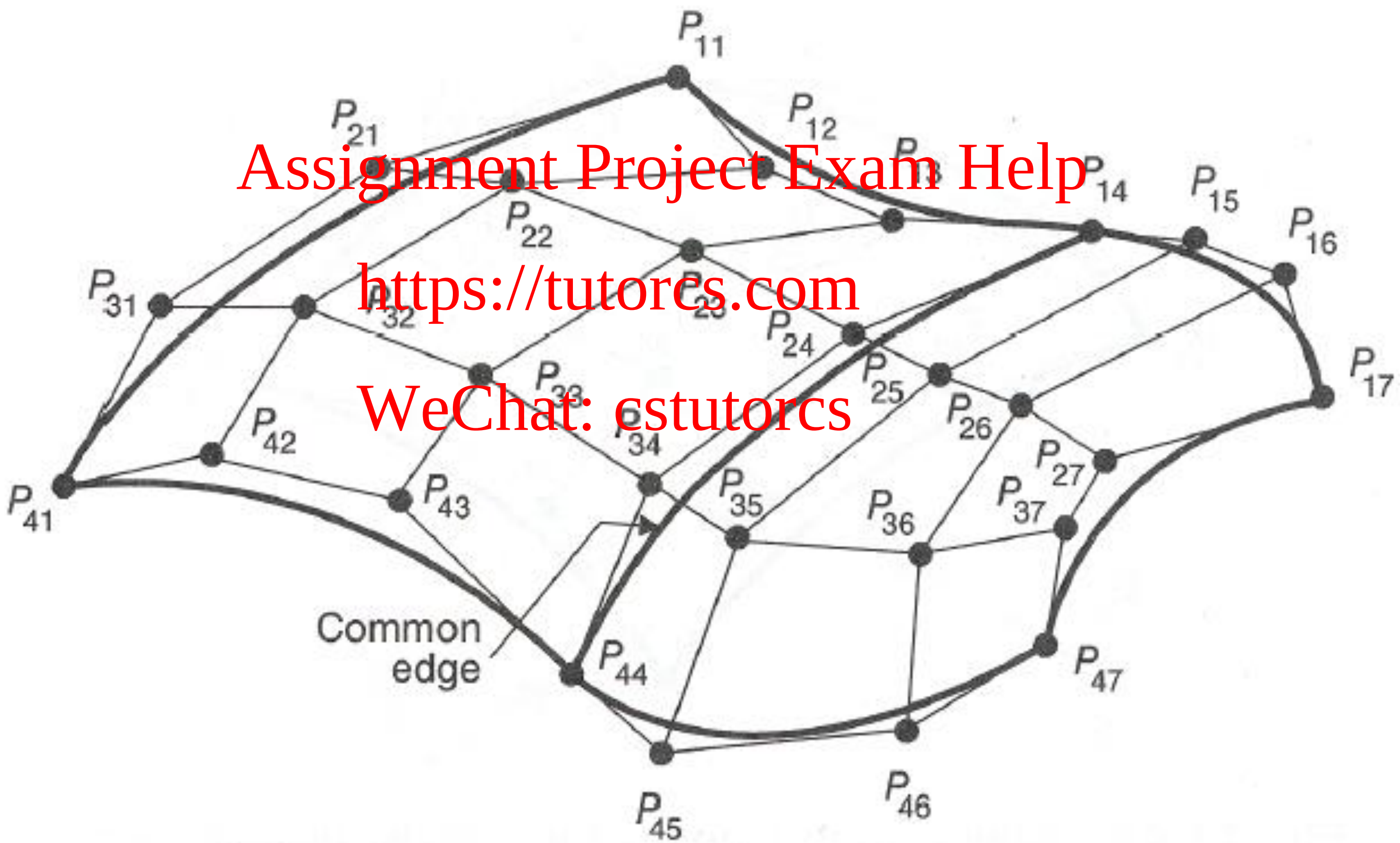
Properties of Bézier Patches

- *Interpolates* four corner control points
- Lies inside *convex hull* of control points
- Changing control points has only local effect (*local control*)



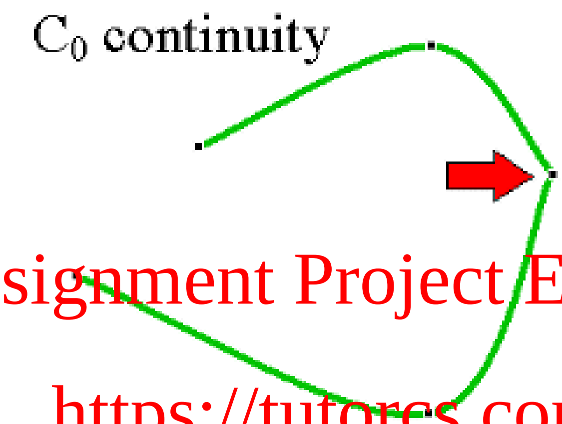
Smooth Bézier Surfaces

- *Continuity* / *smoothness* constraints similar to Bézier splines

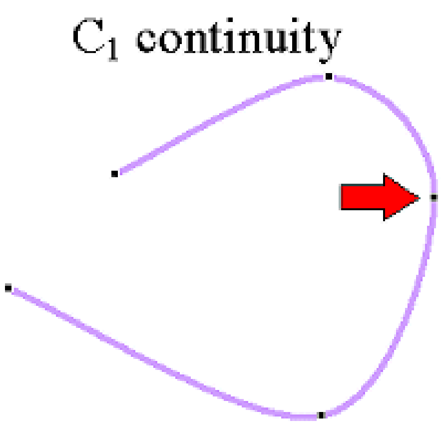


C^0 and C^1 Bézier Surfaces

- C^0 requires *aligning boundary curves*



- C^1 requires aligning boundary curves and derivatives



Drawing Bézier Surfaces

- Simple approach:

  loop through *uniformly* spaced increments of *u* and *v*

for (int $l = 0$; $l < l_{max}$; $++l$) {

  double $u = u_{min} + l * u_{step}$;

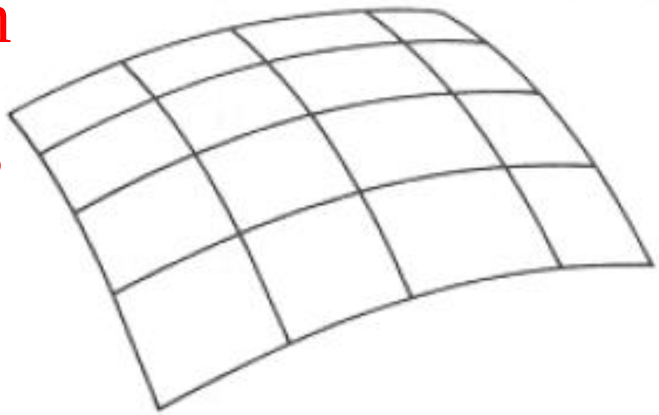
  for (int $k = 0$; $k < k_{max}$; $++k$) {

   double $v = v_{min} + k * v_{step}$;

   DrawQuadrilateral(...);

  }

}

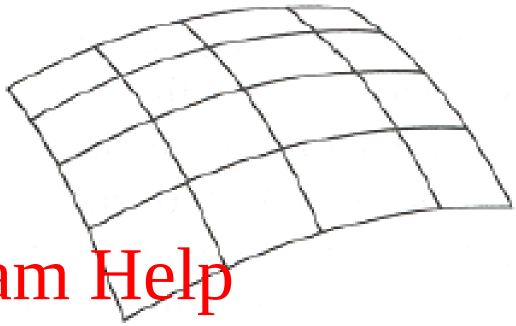


- Note, Bézier surfaces always have *quadrilateral* structure

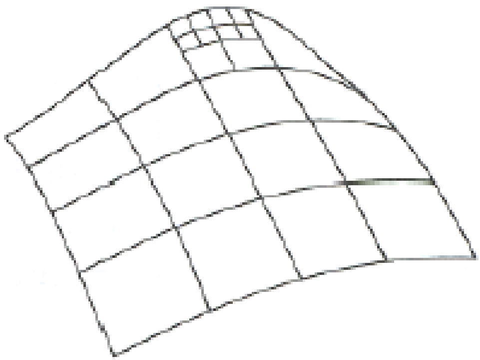
Drawing Bézier Surfaces

- Better approach:
  use *adaptive subdivision*

```
DrawSurface (surface) {
  if flat(surface, epsilon) {
    DrawQuadrilateral (surface);
  } else {
    SubdivideSurface (surface,...);
    DrawSurface (surfaceLL);
    DrawSurface (surfaceLR);
    DrawSurface (surfaceRL);
    DrawSurface (surfaceRR);
  }
}
```

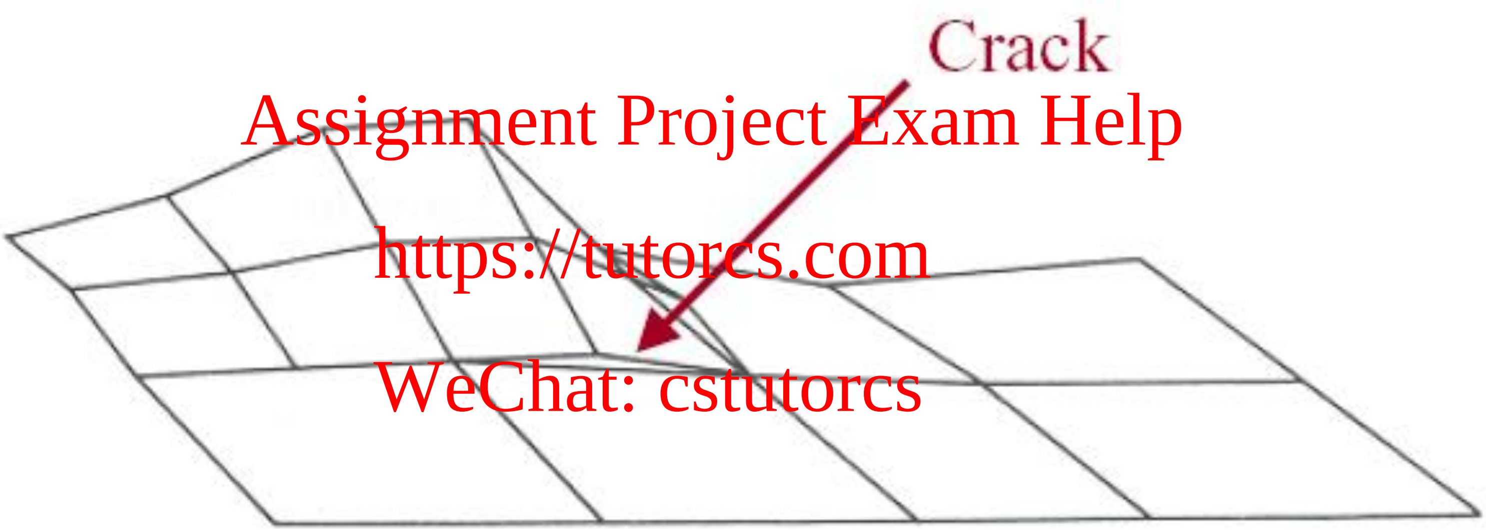
Uniform subdivision



Adaptive subdivision

Drawing Bézier Surfaces

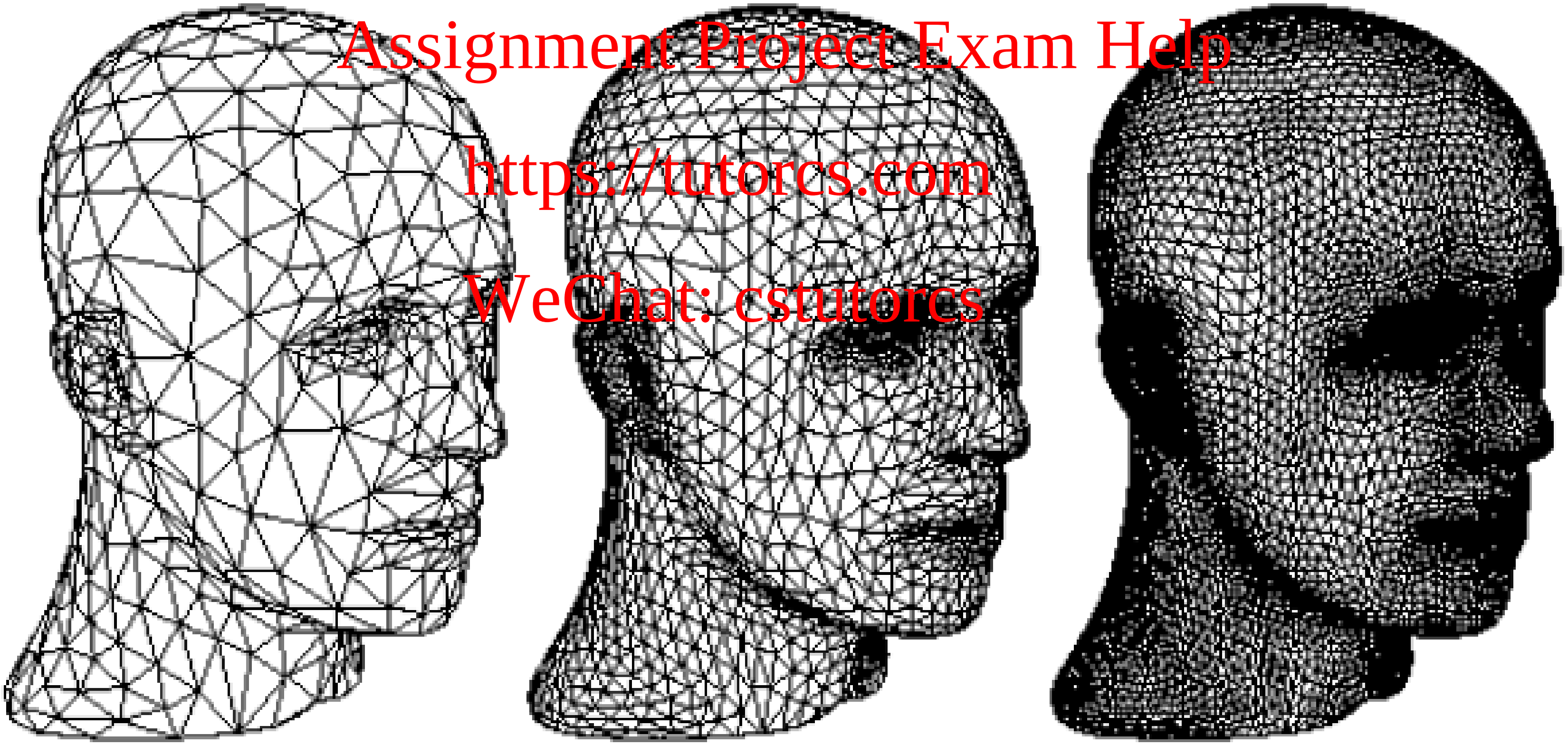
- Problem of adaptive subdivision:
  - *Cracks* at boundaries between patches at different subdivision levels



  - Avoid cracks by *adding extra vertices* and *triangulating* quadrilaterals with neighbours at finer level

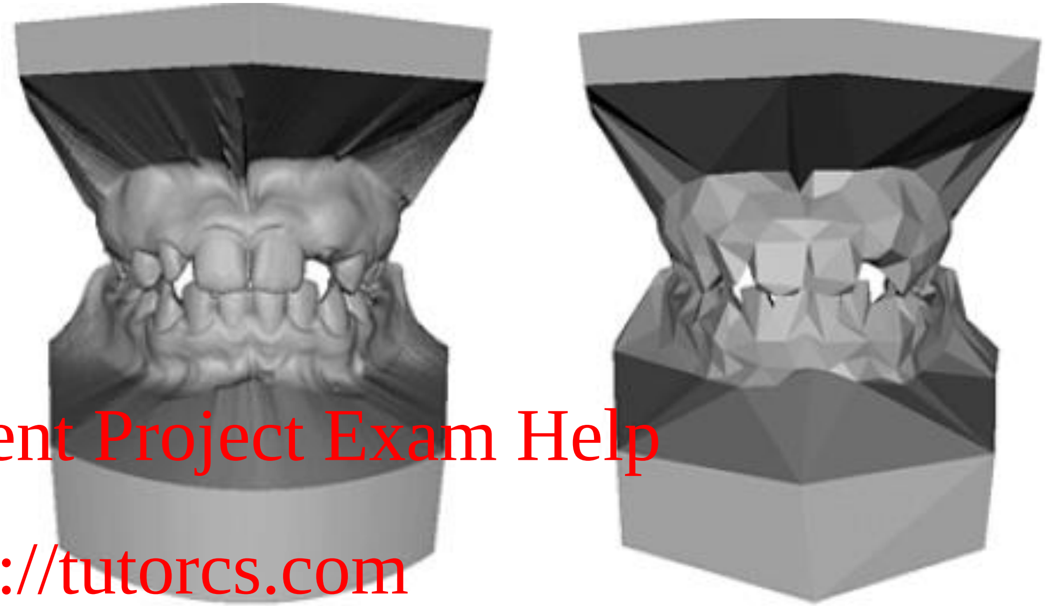
Subdivision Surfaces

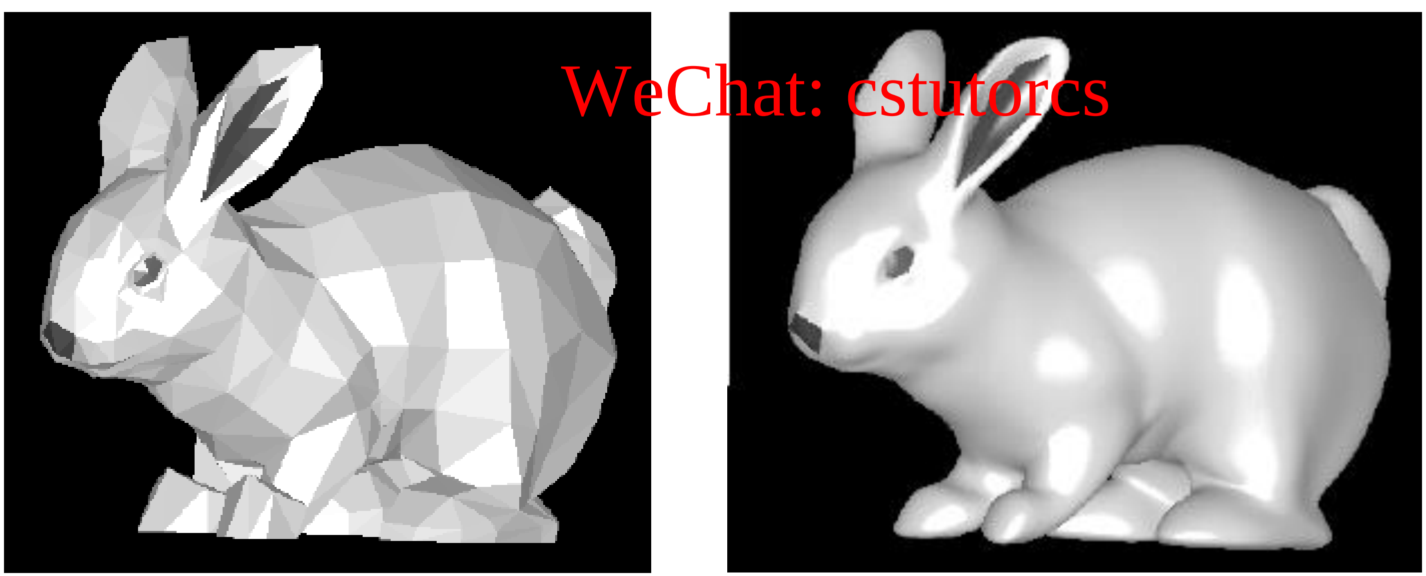
- Idea of *subdivision surfaces*
  - Define a smooth surface as the limit of a sequence of successive refinements of a mesh



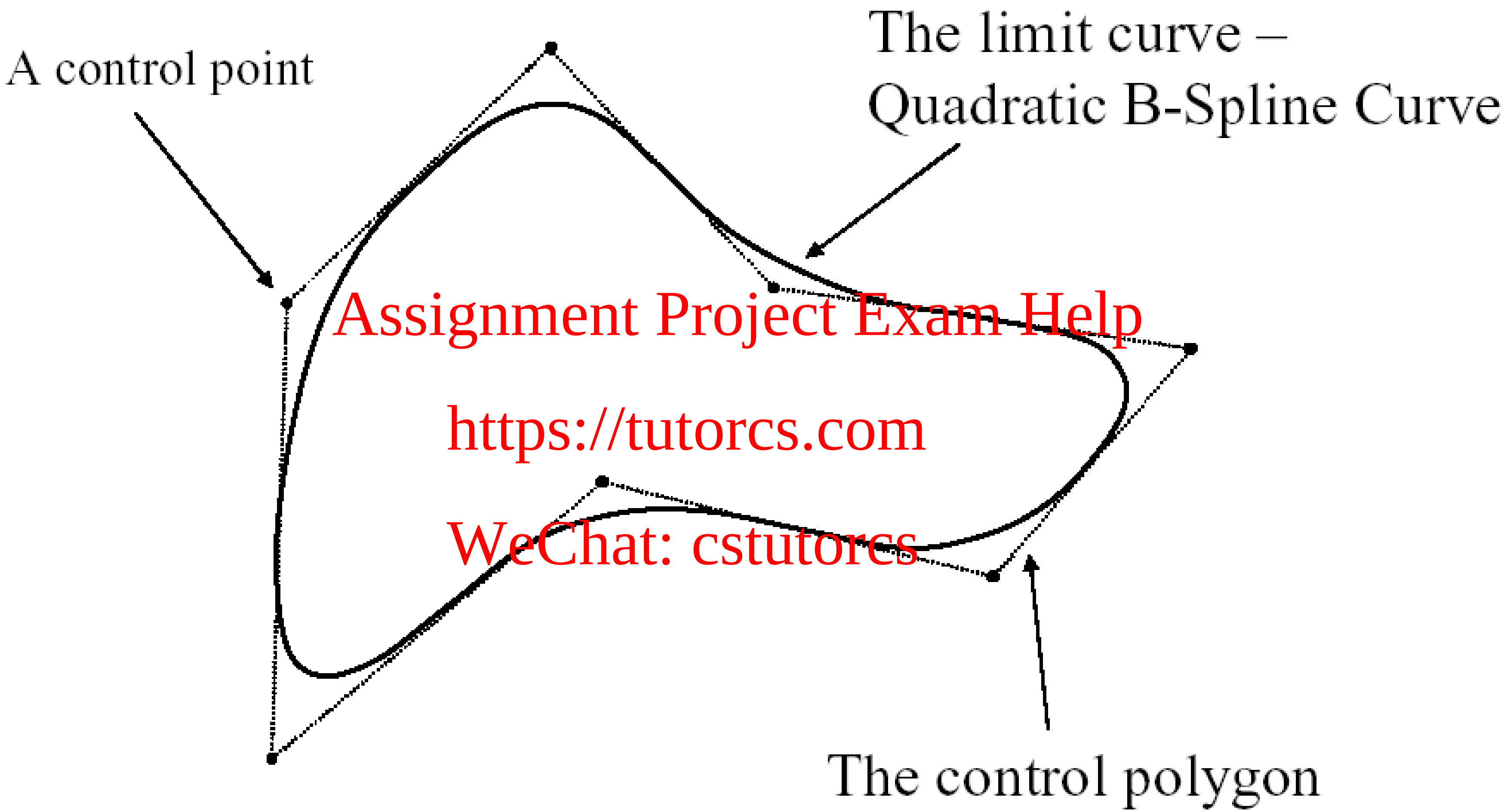
Why Subdivision?

- Level of Detail
- Compression
- Smoothing





Cutting Corners – Curves



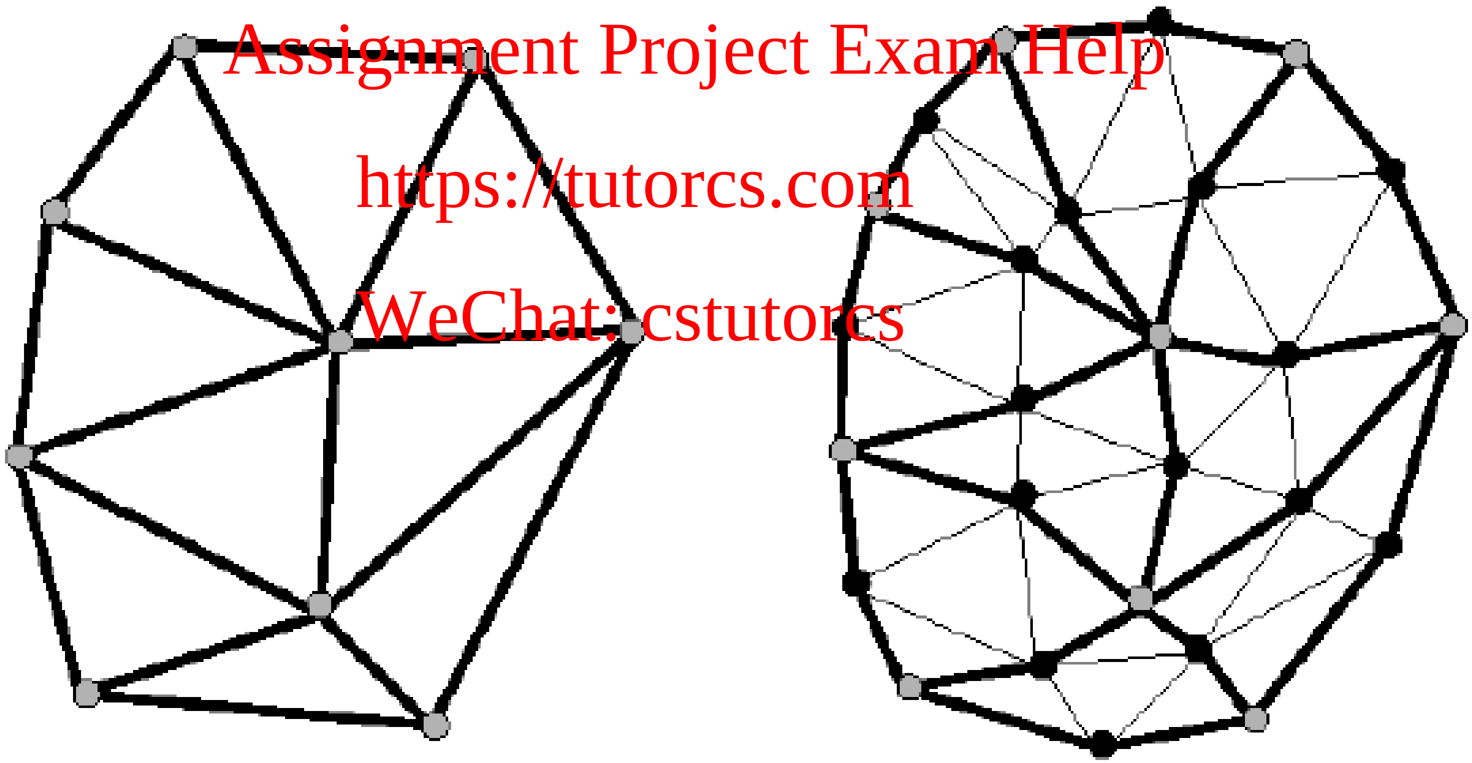
Center for Graphics and Geometric Computing, Technion

Surface Subdivision

- Start with a *control mesh*
- Per iteration construct *refined* mesh by inserting vertices
- Mesh sequence should converge to a *limit surface*
- Subdivision scheme defined by two elements
  - Generate *topology* of the new mesh
  - Compute vertex locations in new mesh
    - *Vertex point*: new location of old vertex
    - *Edge point*: location of new vertex on old edge
    - *Face point*: location of new vertex on old face

Loop Subdivision

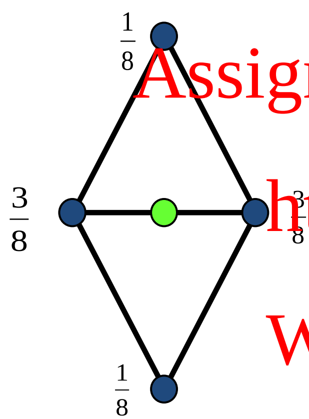
- Loop subdivision scheme:
  - Refine each triangle into 4 triangles by splitting each edge and connecting new vertices



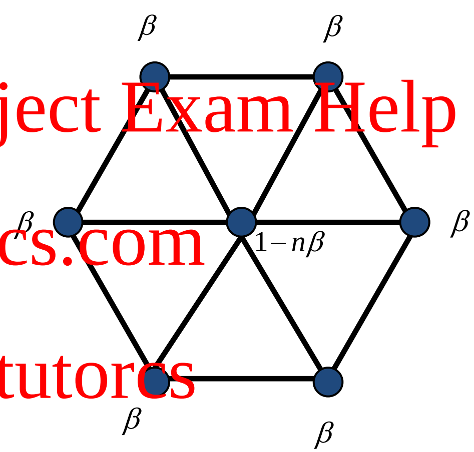
Loop Subdivision

- Computing locations of new vertices
  - *Weighted average* of original vertices in neighbourhood

Edge point



Vertex point



$$\beta = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{2}{8}\cos\left(\frac{2\pi}{n}\right)\right)^2\right)$$

- No face points

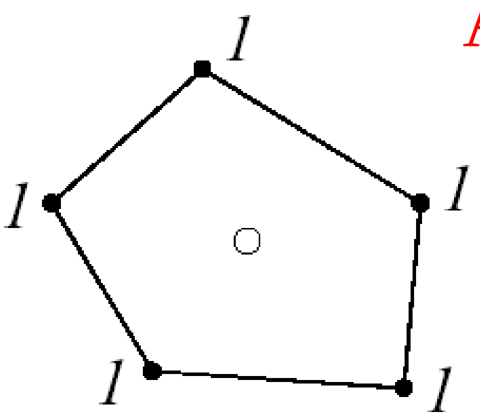
Catmull-Clark Subdivision

- Mesh is the control mesh of a *B-Spline surface*
  - Refined mesh is also a control mesh of a B-Spline Surface
- Incremental construction
  - Calculate face points
  - Calculate edge points using face points
  - Calculate vertex points using face and edge points
  - Connect vertices

Catmull-Clark Subdivision

Step 1

First, all the face points are calculated

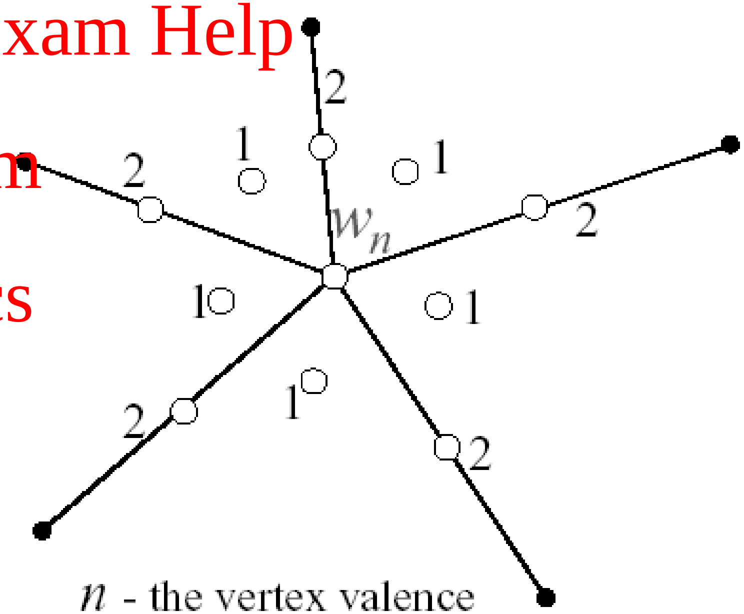


Step 2

Then the edge points are calculated using the values of the face points and the original vertices

Step 3

Last, the vertex points are calculated using the values of the face and edge points and the original vertex

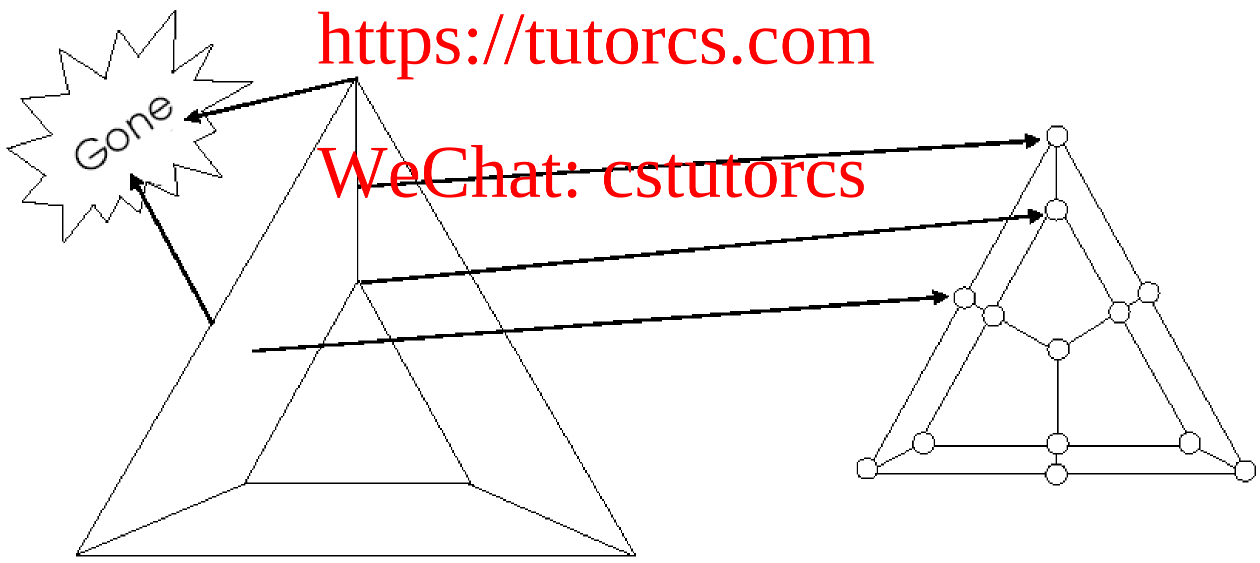


n - the vertex valence

Center for Graphics and Geometric Computing, Technion

Catmull-Clark Subdivision

- Connecting new vertices:
  - Connect each new face point to edge points of the edges defining the old face
  - Connect each new vertex point to new edge points of all old edges incident on the old vertex point



Center for Graphics and Geometric Computing, Technion

Catmull-Clark Subdivision

- Face Point

$$f = \frac{1}{m}\sum_{i=1}^{m} p_i$$

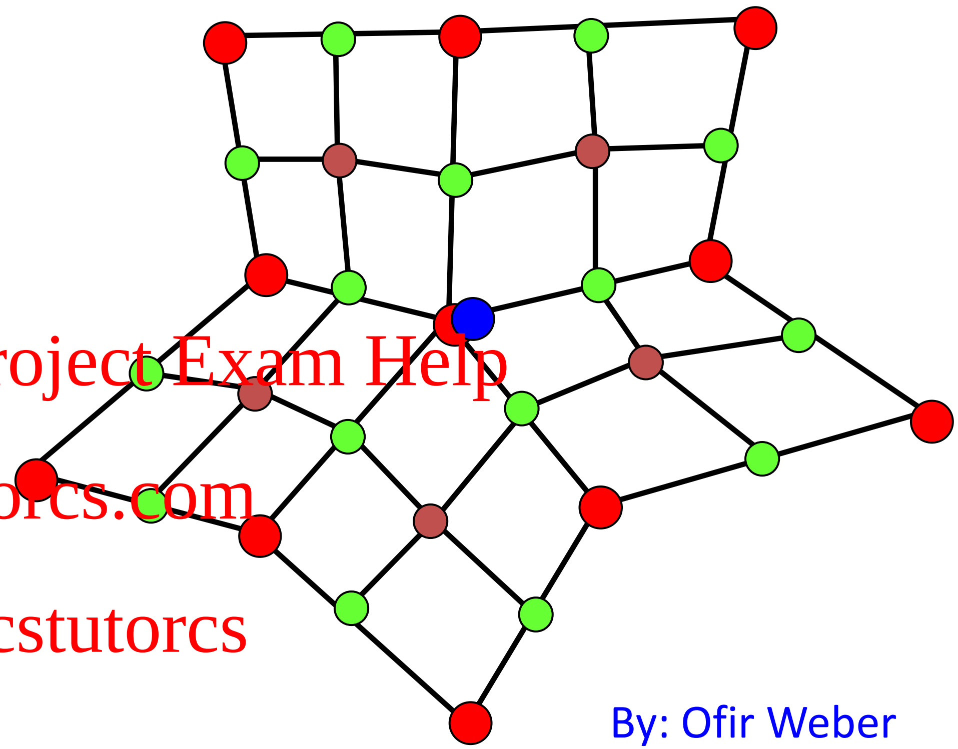
- Edge Point

$$e = \frac{p_1 + p_2 + f_1 + f_2}{4}$$

- Vertex Point

$$v = \frac{Q}{n} + \frac{2R}{n} + \frac{p\,(n-3)}{n}$$

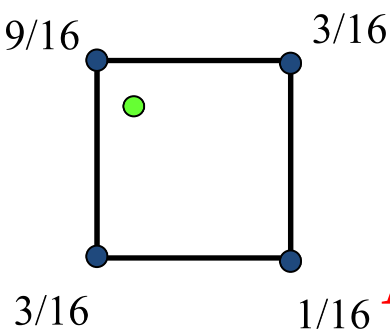
$$\boldsymbol{v} = \frac{1}{\boldsymbol{n}^2}\sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \boldsymbol{f}_{\boldsymbol{i}} + \frac{1}{\boldsymbol{n}^2}\sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \boldsymbol{p}_{\boldsymbol{i}} + \frac{\boldsymbol{n}-2}{\boldsymbol{n}}\boldsymbol{p}$$

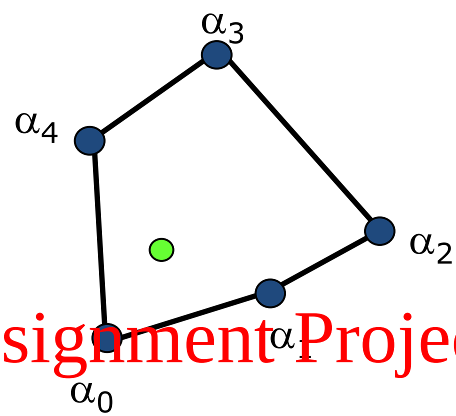


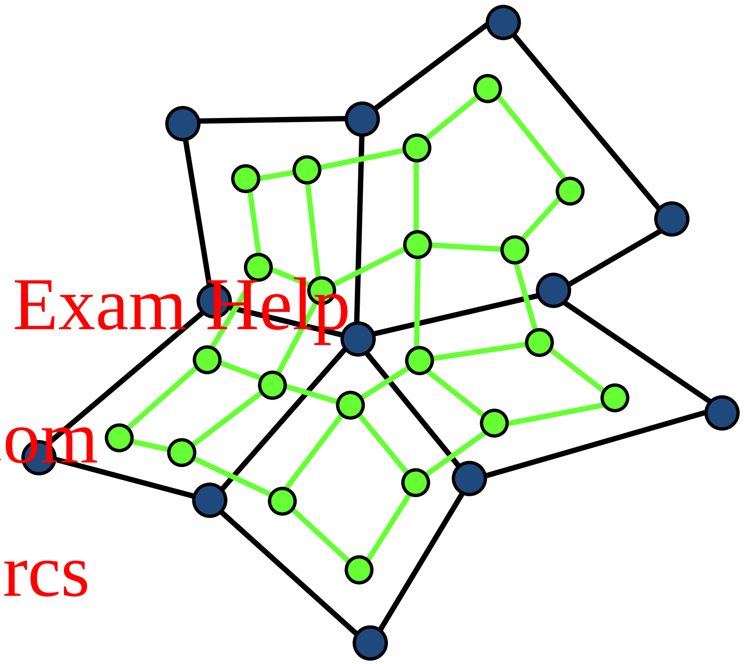
By: Ofir Weber

- Q – Average of face points
- R – Average of midpoints
- p – old vertex

Masks:







$$\alpha_i = \frac{\delta_{i,0}}{4} + \frac{3 + 2\cos\left(2i\pi / n\right)}{4n}$$

$$p = \sum_{i=0}^{n-1} \alpha_i p_i$$

Properties of Subdivision Surfaces

- *Advantages*
  - Simple methods for describing complex surfaces
  - Multi-resolution evaluation and manipulation
  - Arbitrary topology of control mesh (not only quadrilaterals)
  - Limit surface is smooth
- *Disadvantages*
  - No obvious parametrisation
  - Hard to find intersections

Summary

- What are parametric surfaces? What are their advantages and disadvantages?
- What are spline surfaces? What are their advantages and disadvantages? What is the major problem when defining surfaces “piecewise”?
- What is the principle of a tensor product surface? What are Bézier surfaces? What conditions do the control points of C^0/C^1 continuous Bézier surfaces have to fulfil?
- What is the principle of subdivision surfaces? What are their advantages / disadvantages?
- How do Loop, Catmull-Clark, Doo-Sabin subdivision schemes work?