

Computing with Geometry as an Undergraduate Course: A Three-Year Experience

**John L. Lowther and Ching-Kuang Shene
Department of Computer Science
Michigan Technological University
Houghton, MI 49931-1295**

Partially supported by National Science Foundation

● What is this course all about?

We want to teach our students the way of handling geometric problems.

● Is this course necessary?

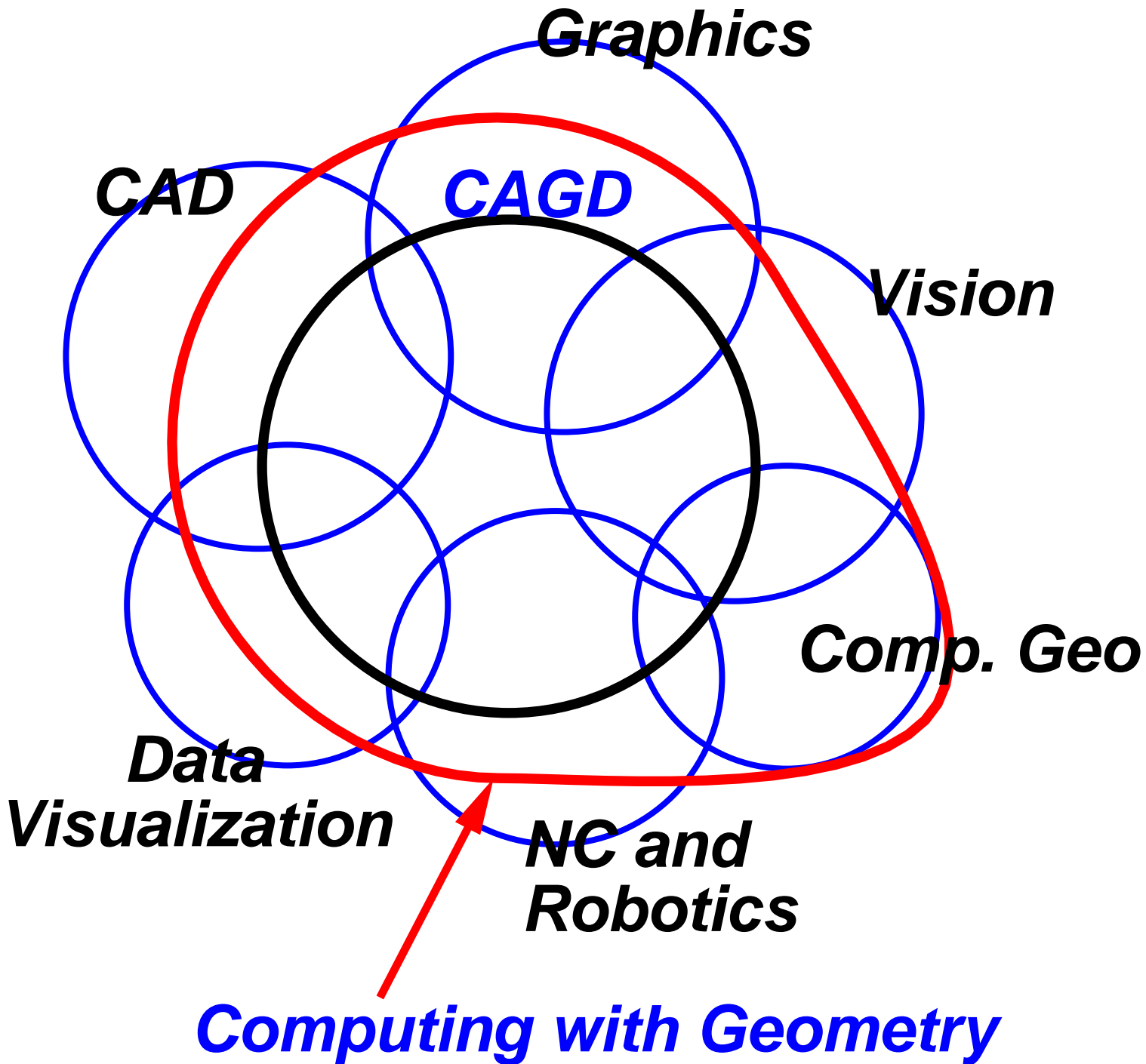
It depends. Many people oppose to this idea. But, as long as the demand is there, it is worth to provide our students with this important skill.

● How important is this course?

This is a *geometric* world. But, a typical CS student may not receive any geometric training.

Geometric materials are scattered throughout many courses *without* a coherent view.

This is NOT a Graphics Course



This course *does not* require graphics background

Nobody Teaches this Course!

There Is NO National Impact!!

- **Here are some information gathered from our course web site**

Site	Daily Avg
Course Info Page	10.91
Electronic Book	13.87
Curve User Guide	5.07
Surface User Guide	3.04
Download Visitors	4.96
Total Downloads	1090
CD-ROM Distributed	50+

- **We have 10–20 students per year**

- Educators and students *are* interested in these topics


Domain	%
CS,EECS,Math/CS...	27.47
CS alone	22.07
Science	4.67
Math alone	3.57
Engineering	7.78
Mech Eng alone	4.67
EDU general	11.36
Education Total	51.28
COM	25.09
Other	22.34
ORG/GOV	1.29

- "Other" may include many students as shown by their correspondences

Geographical Distribution

Area	%
North America	41
South America	2
Europe	41
Far East	9
Other areas	7

Top Three	%
USA	38.53
France	7.98
Germany	7.16

-  **We have no way to count visitors to our electronic book and user guides because our server does not keep track this information.**

- A quick Internet search reveals the following universities where our materials are being used:

Arizona State

DePaul University

Feng Chia University (Taiwan)

Ghent University (Belgium)

The Hong Kong U. of Sci. & Tech

INRIA Sophia–Antipolis (France)

Mechanical Engineering Nis (Yugos)

MIT

Ohio State

Taylor University

Technische Fachhochschule Berlin

University of Alaska

University of Alberta (Canada)

University of Magdeburg (Germany)

University of Patras (Greece)

Verona University (Italy)

Western Washington U.

- This search *does not* include courses *without* course web sites.

Too Much Mathematics

We are not in Math or Eng Dept

- Yes and No.
- With the help of our software tool **DesignMentor** we can take a hands–on and intuitive approach and do not use very much mathematics. Our electronic book shows this well.
- Calculus and elementary linear algebra *are* required. The former helps understand the characteristics of curves and surfaces and the latter provides a solid background of geometric transformations and affine and projective spaces.
- Thus, if students have good math background, we do more theory; otherwise, let them know more about algorithms and applications.

Design Merit

- Only covers the fundamentals of solids, curves, surfaces and their important algorithms and operations.
- It has to be intuitive and elementary, and takes a learning–by–doing approach. *Complex algebraic derivations and calculations are not our primary topics.*
- You cannot teach beginners top–down design because they do not know which way is up (C.A.R. Hoare). In a geometry class, let the students see the geometry *must* be the first step. So, we use our **DesignMentor** software tool.

The Theme of This Course

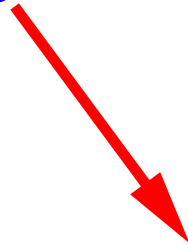
Geometry



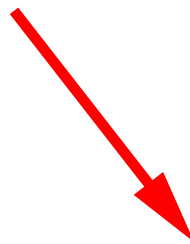
Representation



Algebra

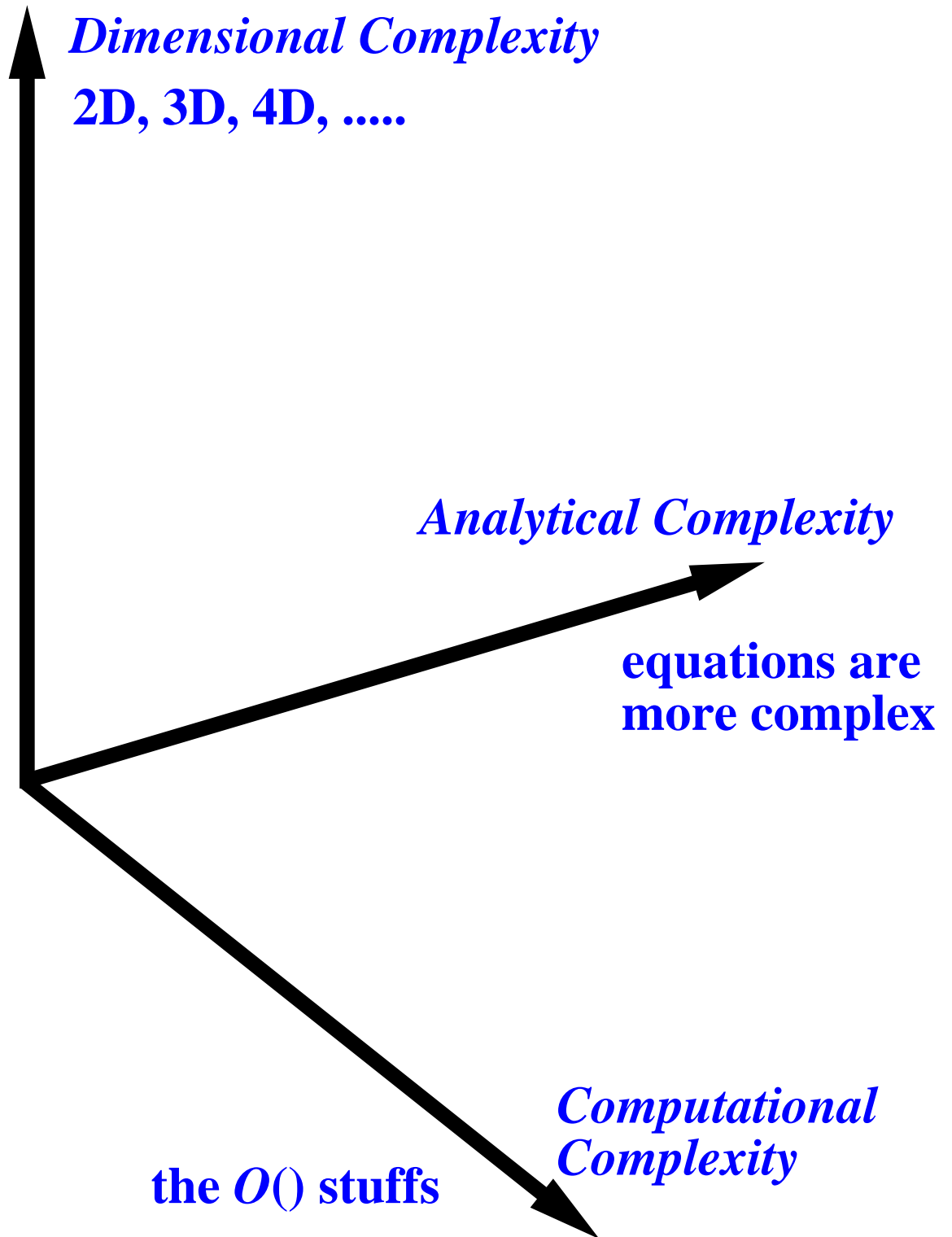


Algorithm



Program

Complexity of Geometric Problems



Algorithm Robustness

- Floating point computations cannot be exact. How do we use discrete entities to *represent* a continuum?
- What are the roots of $x^2 + 20000x + 1 = 0$ using single precision? **Answer:** 20000 and 0!
- Will the following five expressions deliver the same result?

$$x_{i+1} = (R + 1)x_i - R(x_i x_i)$$

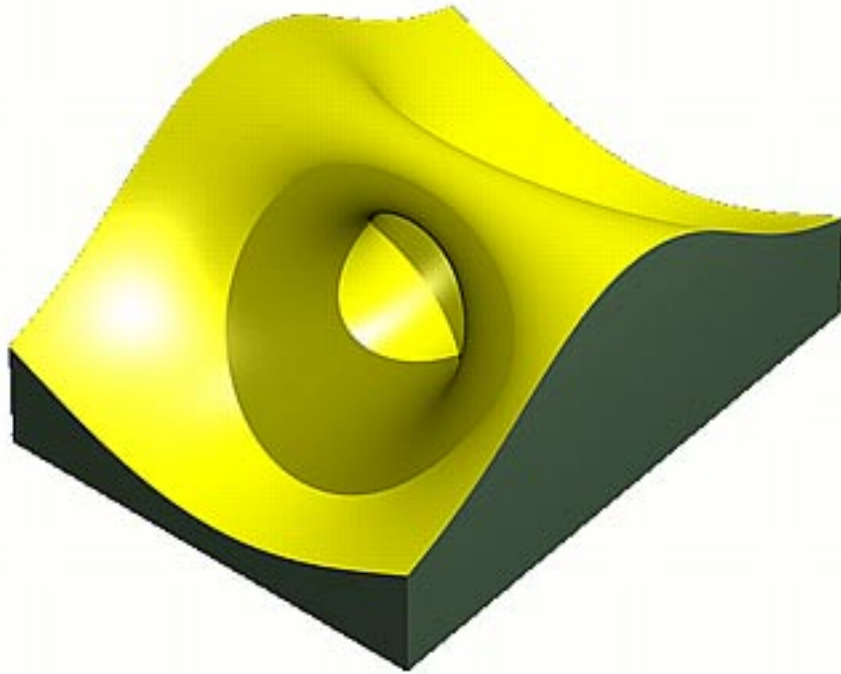
$$x_{i+1} = (R + 1)x_i - (Rx_i)x_i$$

$$x_{i+1} = ((R + 1) - Rx_i)x_i$$

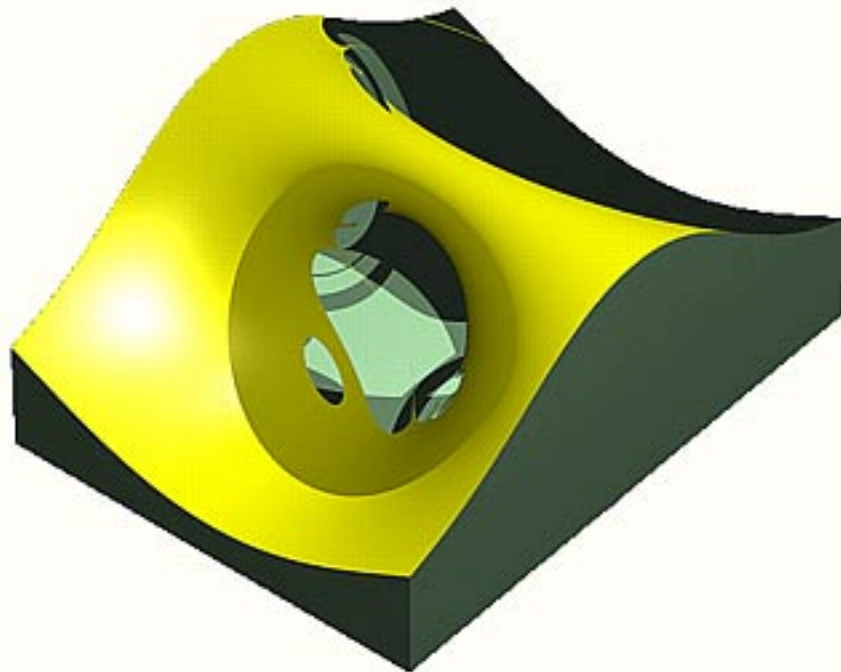
$$x_{i+1} = Rx_i + (1 - Rx_i)x_i$$

$$x_{i+1} = x_i + R(x_i - x_i x_i)$$

- No, it is a chaotic system!



Good Method



Poor Method

Course Contents

- **Course Overview**
- **Review of Geometric Concepts**
- **Object Representations**
- **Parametric Curves and Surfaces**
- **Bezier, B-spline and NURBS**
- **Implicit Curves and Surfaces**
- **Computational Geometry Issues**
- **Robustness Issues**
- **Interpolation and Approximation**

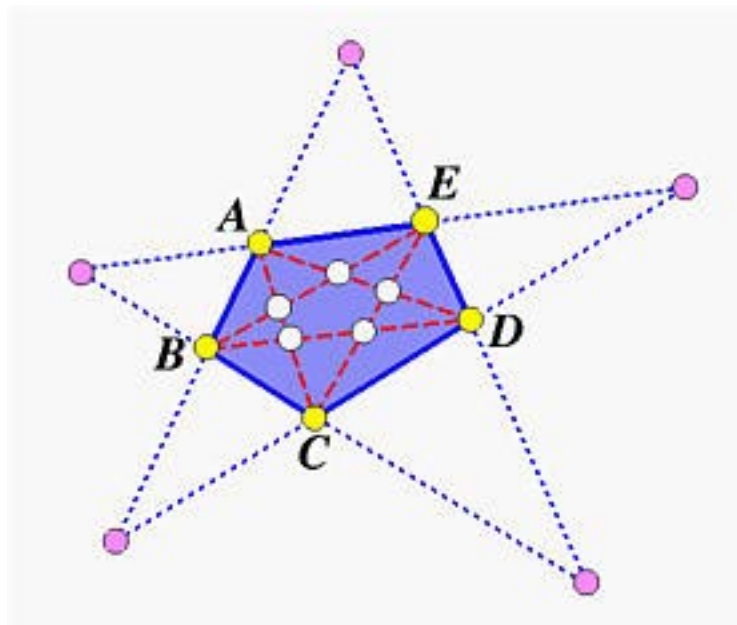
Course Overview

- **The theme of this course, complexity of geometric problems, and so on.**
- **The impact of finite precision arithmetic to geometric problems**
- **The failure of associative law and distributive law**
- **How to prevent the loss of significant digits**
- **Many computation examples**

A Taste of Implementation Failure

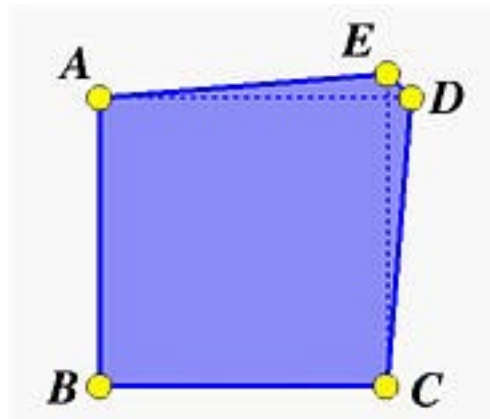
● Programming Exercise 1:

A pentagon can be shrunk (*going in*) or expanded (*going out*).



Are $out^2(in^2(P))$, $in^2(out^2(P))$, ... the same as P ?

$P \rightarrow$



the differences are 0.1, 0.01, 0.001, 0.0001, ...

Object Representations

● Wireframe and polyhedron models

● Boundary representations

Manifolds, the winged–edge data structure, the Euler–Poincare formula and Euler Operators (*e.g., make an edge and a vertex, make a face and an edge, make a shell and a hole, kill an edge and a vertex, kill a face and an edge, kill a shell and a hole and so on*).

Euler operators are used for constructing polyhedral solids.

● Constructive Solid Geometry

Interior, exterior and closure, regularized Boolean operators, and design examples.

Programming Exercise 2: Part I

● Winged–edge data structures

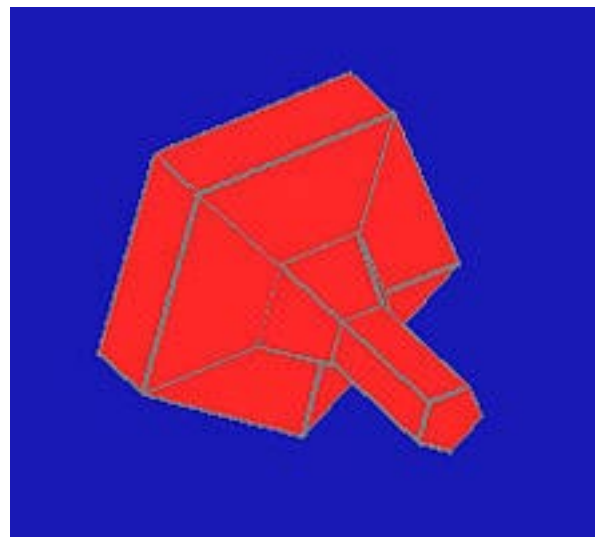
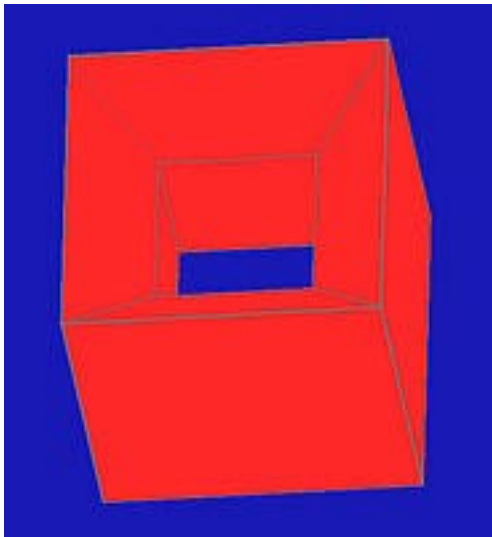
Perform queries such as

given a vertex, find all adjacent edges

given a facet, find all incident edges

given a vertex, find its link or star

Write a program that reads in a description of a solid with winged–edge data structure and display it. Students are required to supply some *interesting* designs.



Programming Exercise 2: Part II

Constructive Solid Geometry

Design a scene using set union, intersection and difference. Students are given a scene and an object to start with.



MTU

CSE



Background Information

- **This is a 300–level course. Most students are juniors, some are seniors, and only a few are sophomores.**
- **Course Prerequisites**
C/C++, data structures, calculus and linear algebra
- **Teaching Environment**
This course is taught in a computer equipped classroom. Course materials are all web–based. Software tools can be accessed from every workstation. Thus, we can take a hands–on approach.
- **Software Tools**
DesignMentor and a raytracer

Parametric Curves

● Parametric Curves

● Moving Triad

Tangent vector, normal vector, bi-normal vector, curvature, and curvature sphere

● Continuity Issues

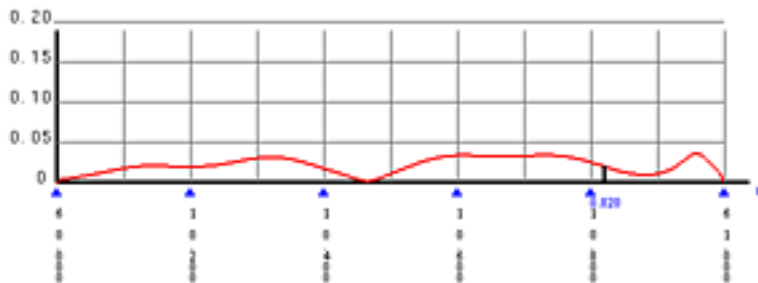
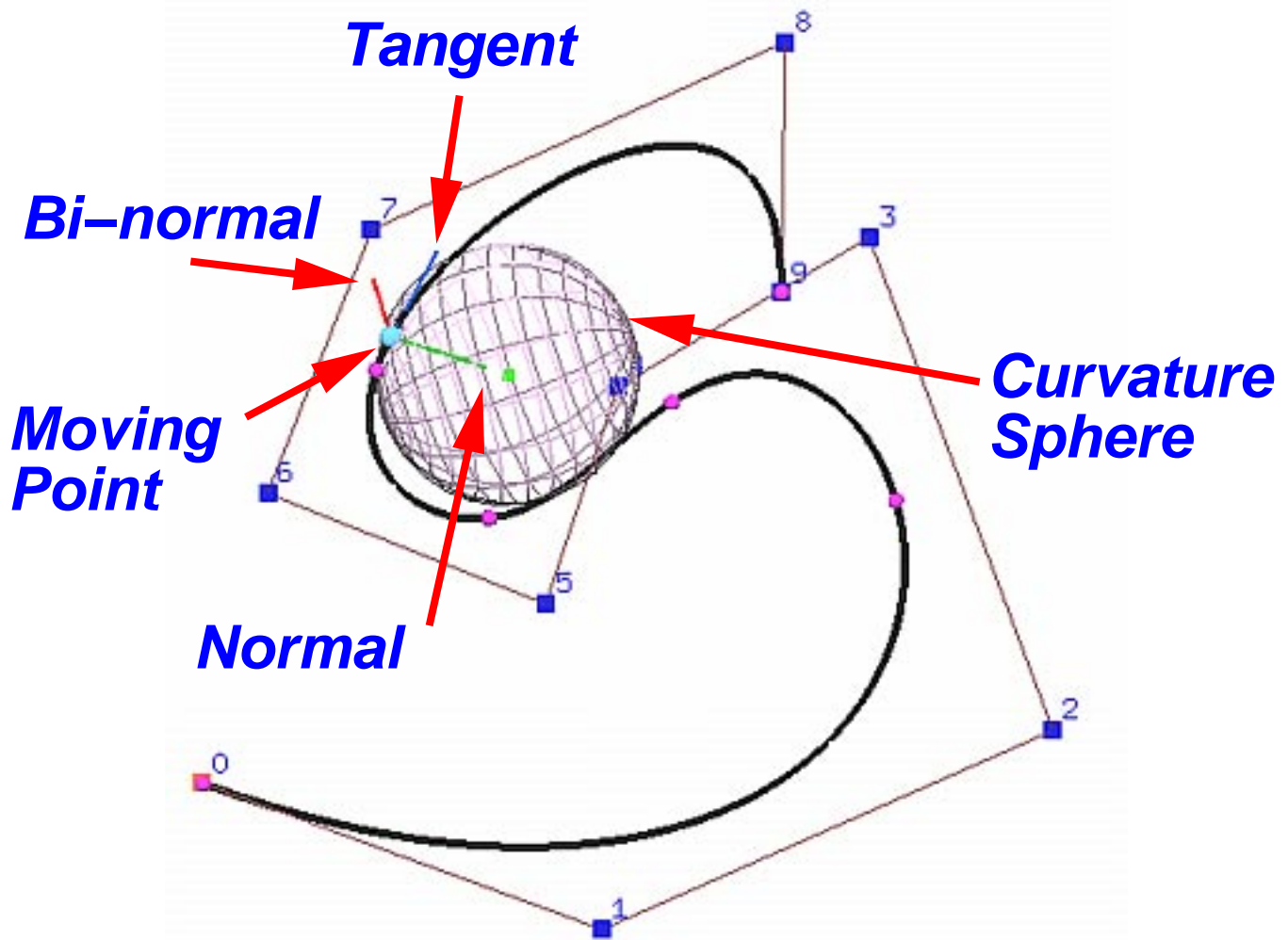
C^k – and G^k – continuity, curvature continuity

● Rational Curves

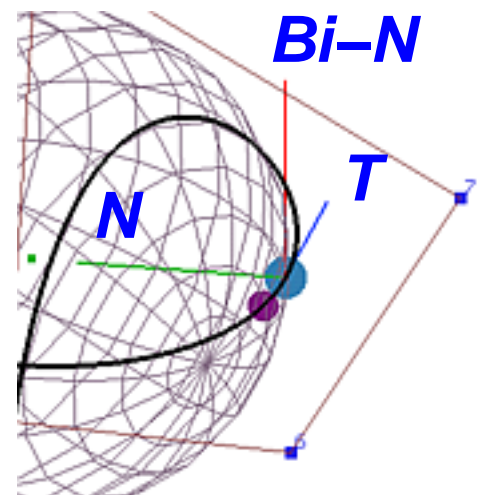
Why are circles, ellipses, and hyperbolas cannot be represented by polynomials. This opens up a window for the need of NURBS

Uniformization Theorems. "Parametric \rightarrow implicit" is always possible. Only rational curves can have parametric forms.

Curve Tracing with DesignMentor



Curvature Plot



Riding on the Curve

Bezier, B-Spline and NURBS

● Bezier Curves (about 3 hours)

Fundamentals: construction, partition of unity, convex hull property, affine invariance, and variation diminishing property

● De Casteljau's Algorithm

This is the algorithm for computing a point on a Bezier curve given a parameter

● Advanced Topics

Curve Subdivision: Dividing a Bezier curve into two Bezier sub-curves

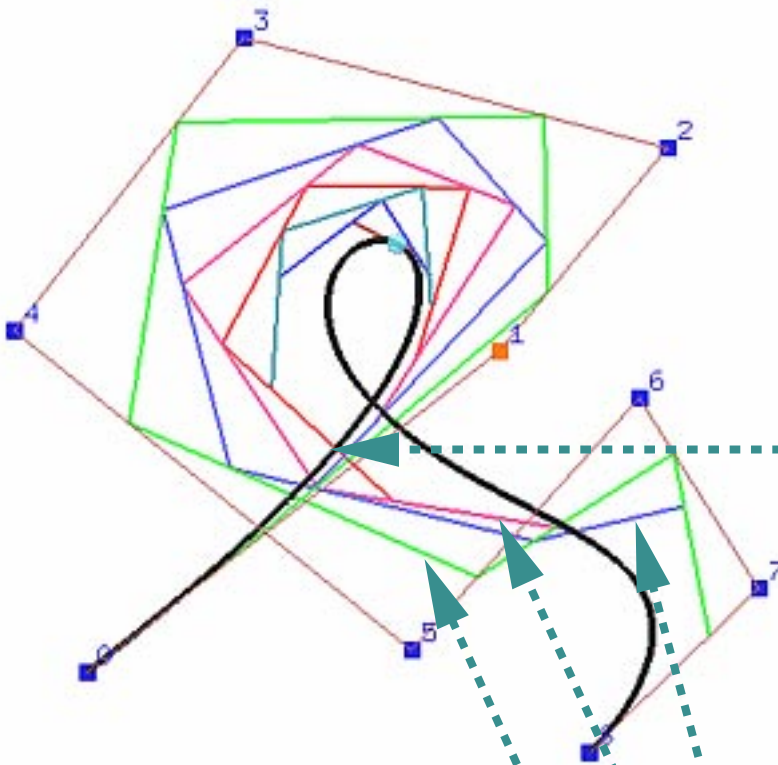
Degree Elevation: Increasing the degree of a Bezier curve by one *without* changing the shape of the curve

● Other Important Topics

First derivative (Hodograph), second and higher derivative. The concept of *corner cutting*.

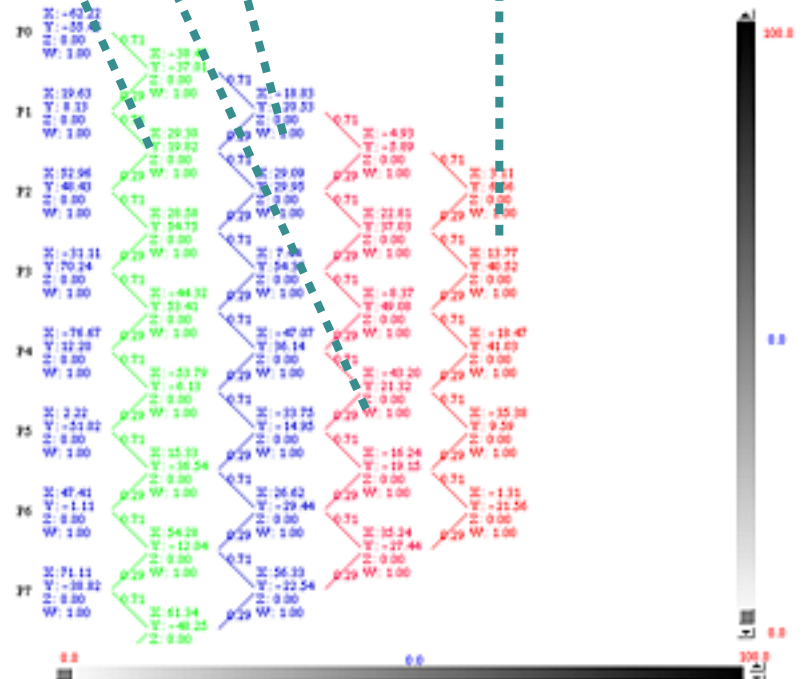
De Casteljau's Algorithm with DesignMentor

de Casteljau's Algorithm and Intermediate Computation

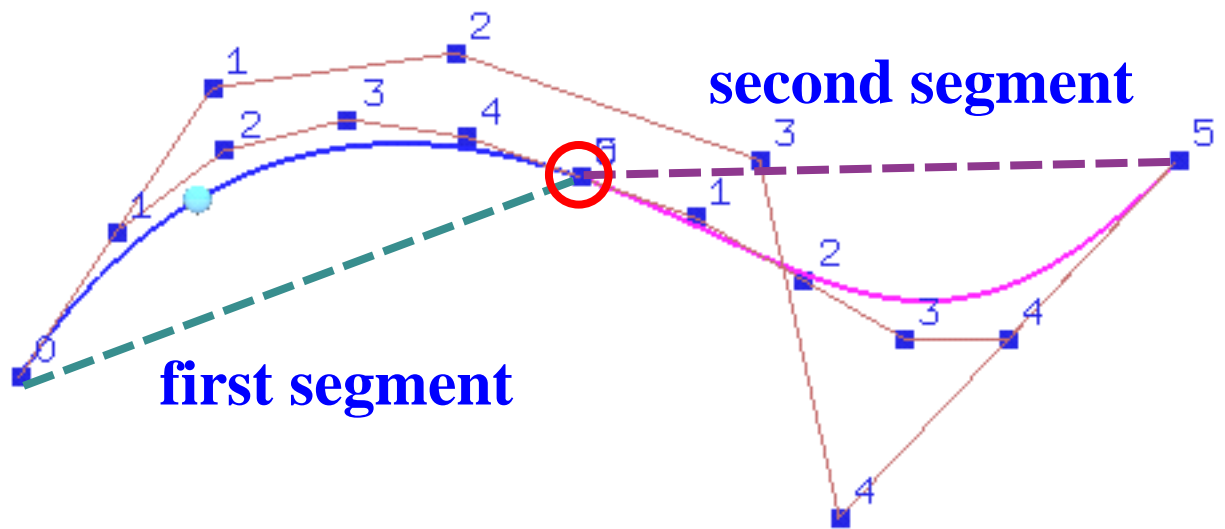
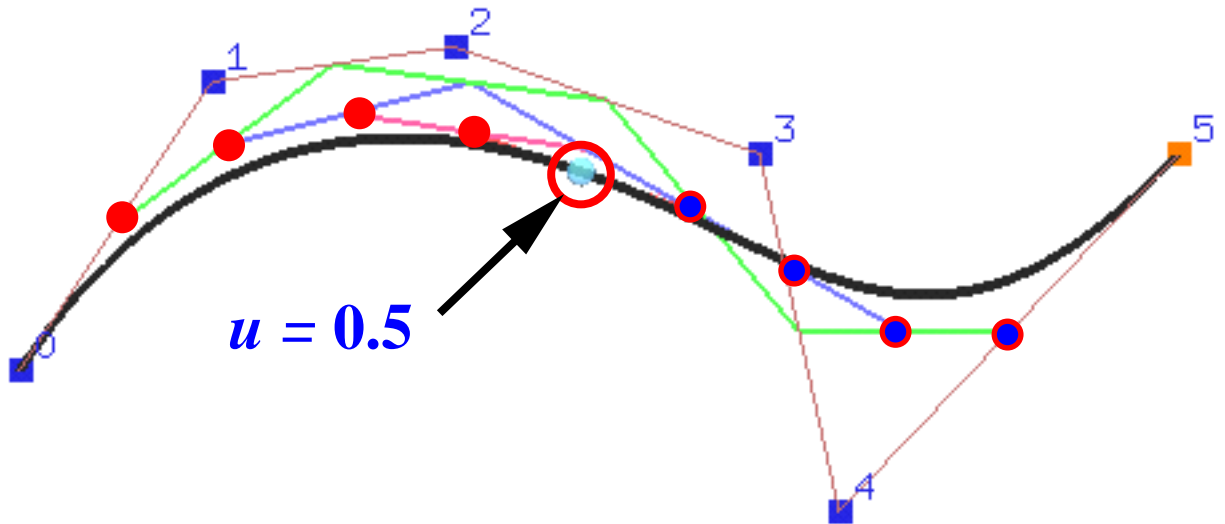


Note the color relation

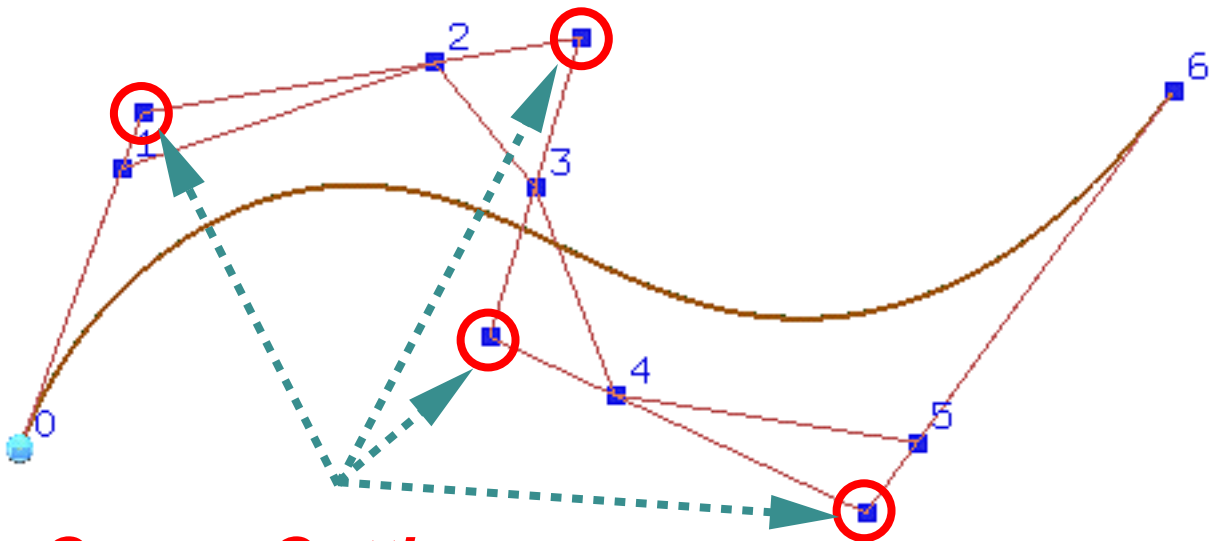
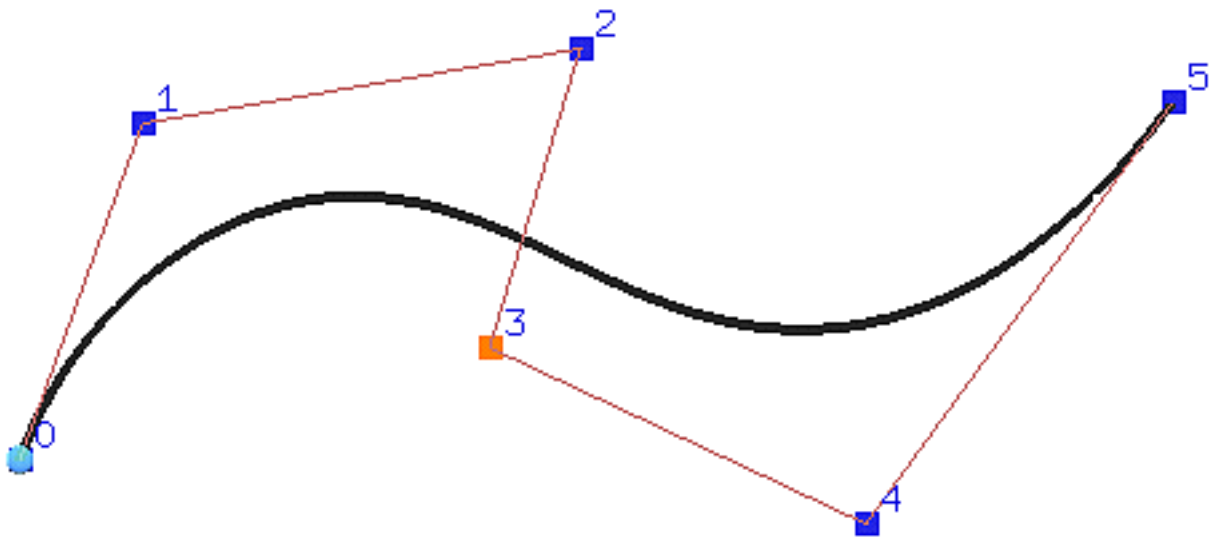
Stepwise Computation



Curve Subdivision with DesignMentor



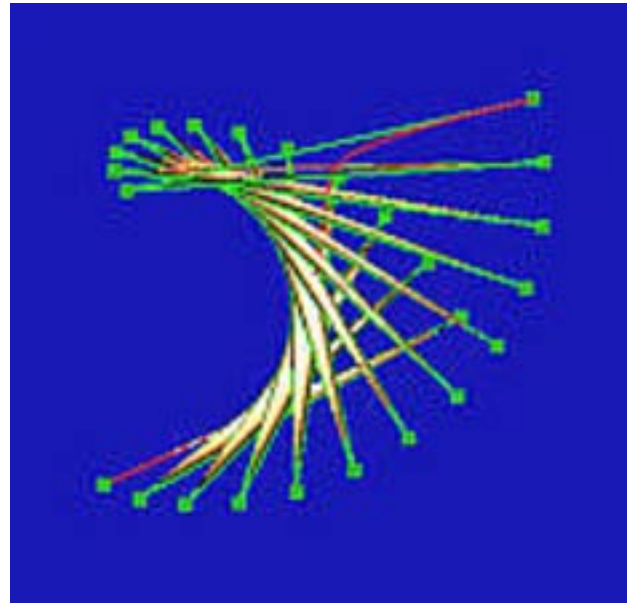
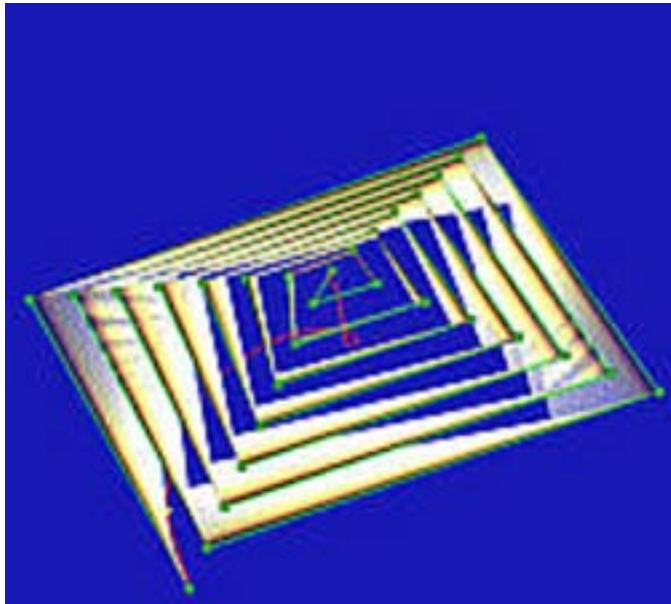
Degree Elevation with DesignMentor



Corner Cutting

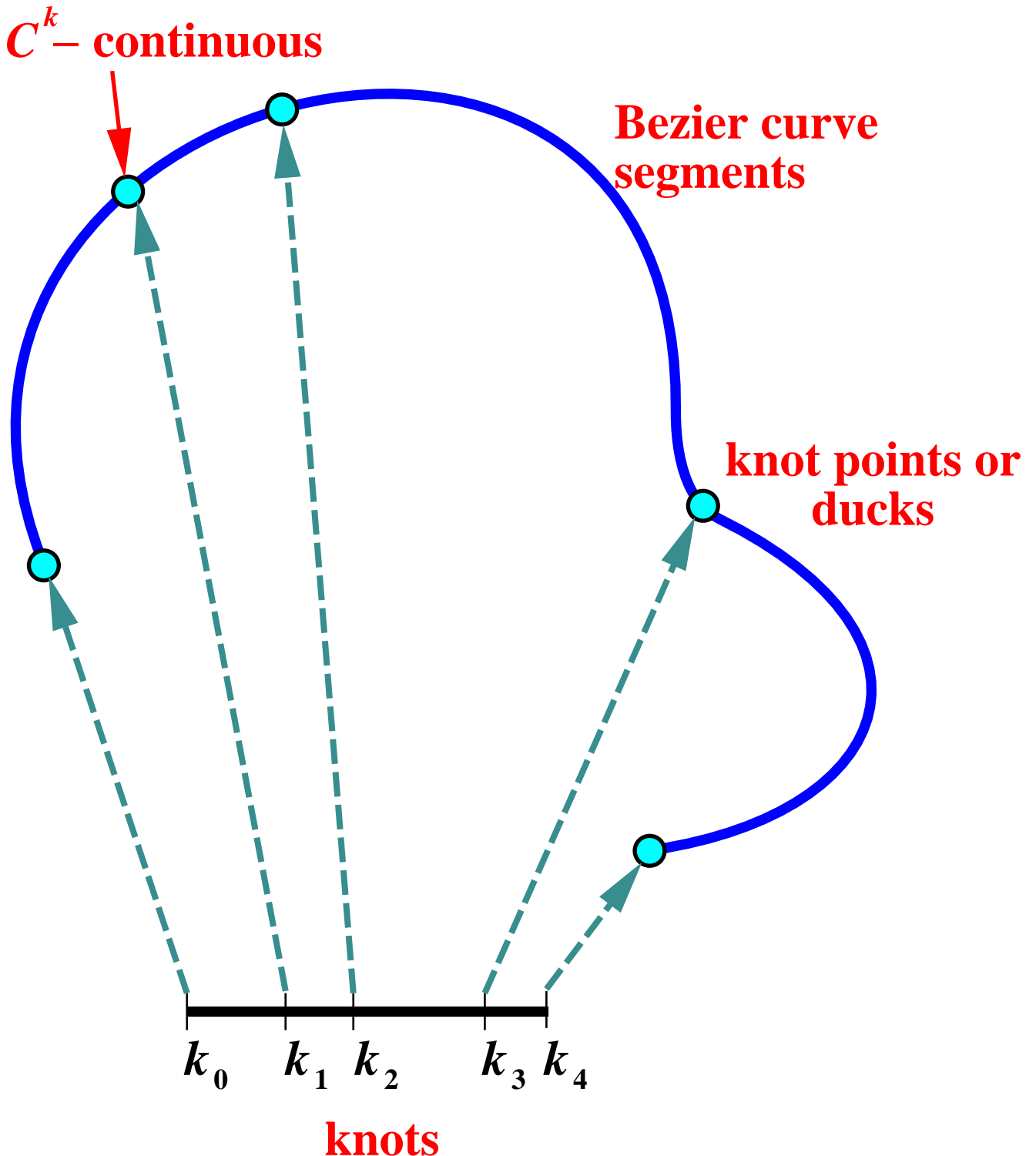
Programming Exercise 3

- Implement de Casteljau's algorithm and use it to trace a Bezier curve. Also design some interesting curves.



● B-spline Curves (about 6 hours)

Motivation: a B-spline curve is the union of a number of Bezier curves joining together with C^k -continuity.



● B-spline Fundamentals

Construction, partition of unity, *strong convex hull* property, *local modification* property, affine invariance and variation diminishing

● Advanced Topics

Knot Insertion: Inserting a new knot *without* changing the shape of the B-spline curve

Curve Subdivision: Dividing a B-spline curve into two B-spline sub-curves

Degree Elevation: Increasing the degree of a B-spline curve by one *without* changing the shape of the B-spline curve

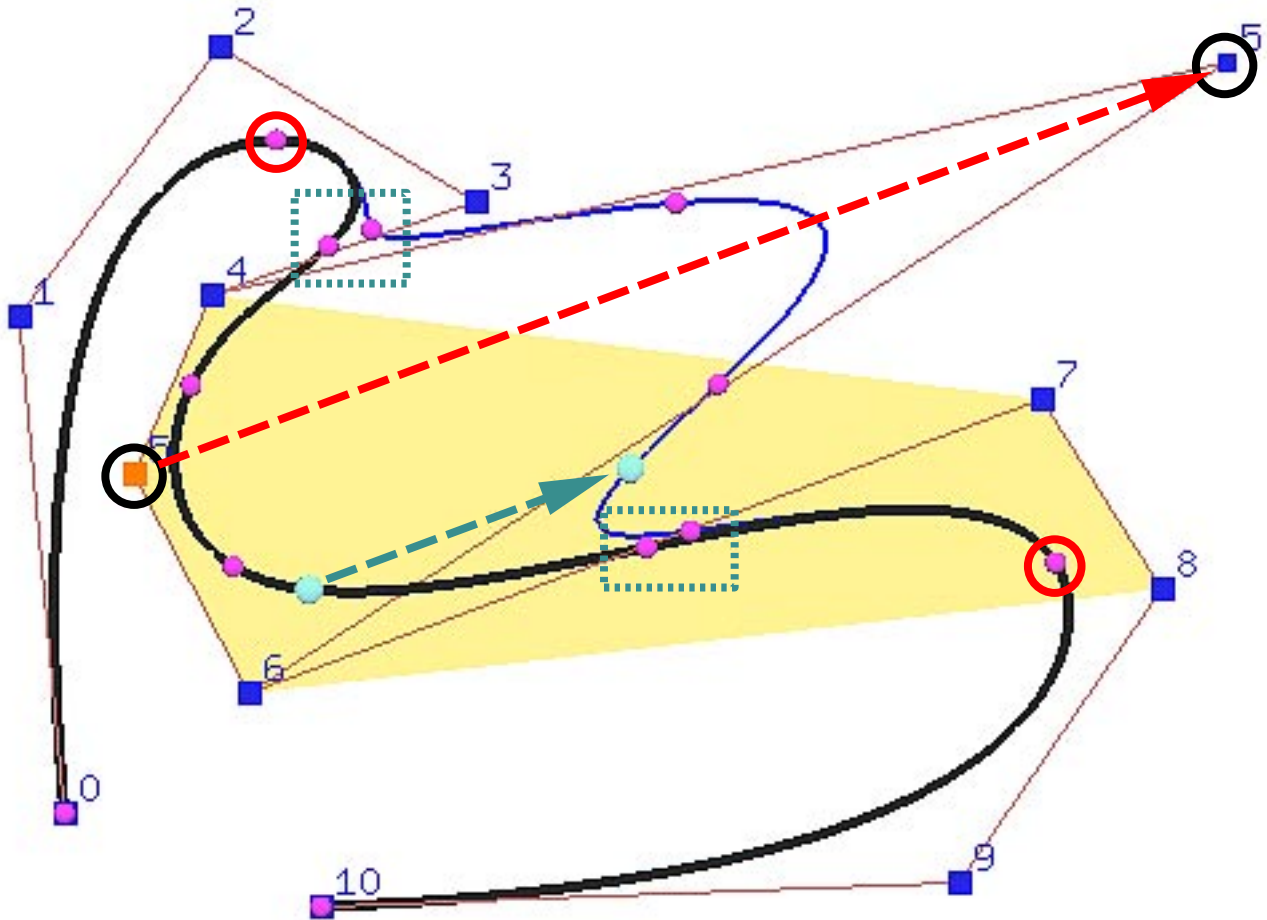
● De Boor's Algorithm

Computing a point on a B-spline curve given a parameter. **This extends de Casteljau's algorithm to B-spline curves.** De Boor's algorithm is implemented by repeated knot insertion.

● Other Topics

Derivative computation

Shape Editing with DesignMentor



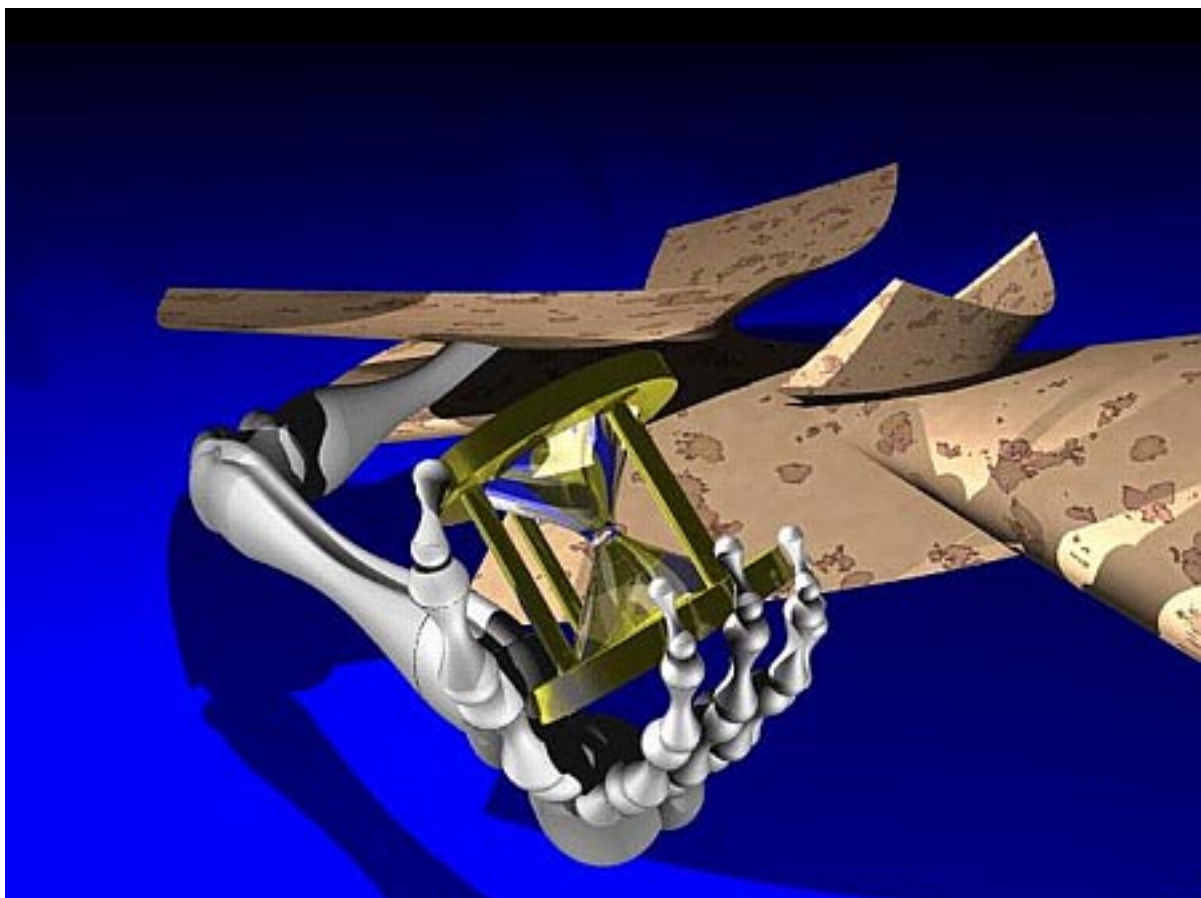
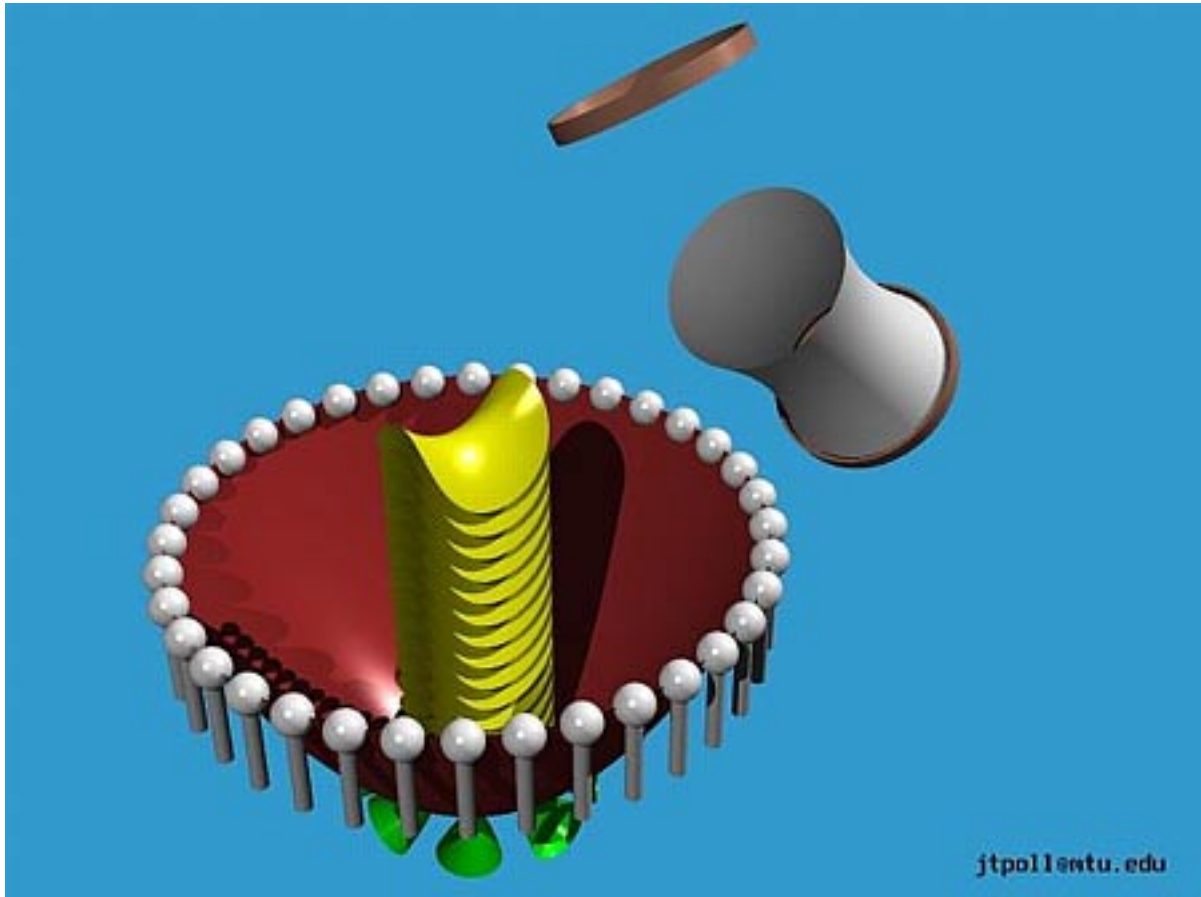
The local modification property guarantees that *only part of the curve* will be affected when a control point changes its position.

In fact, position change is *parallel* to the movement of the control point.

Programming Exercise 4

- Implement de Boor's algorithm with repeated knot insertion and use it to trace a B-spline curve
- Familiarize all quadric surfaces and use *all* of them to construct a scene using whatever technique you like (*e.g.*, constructive solid geometry). Quadric surfaces are in implicit form.

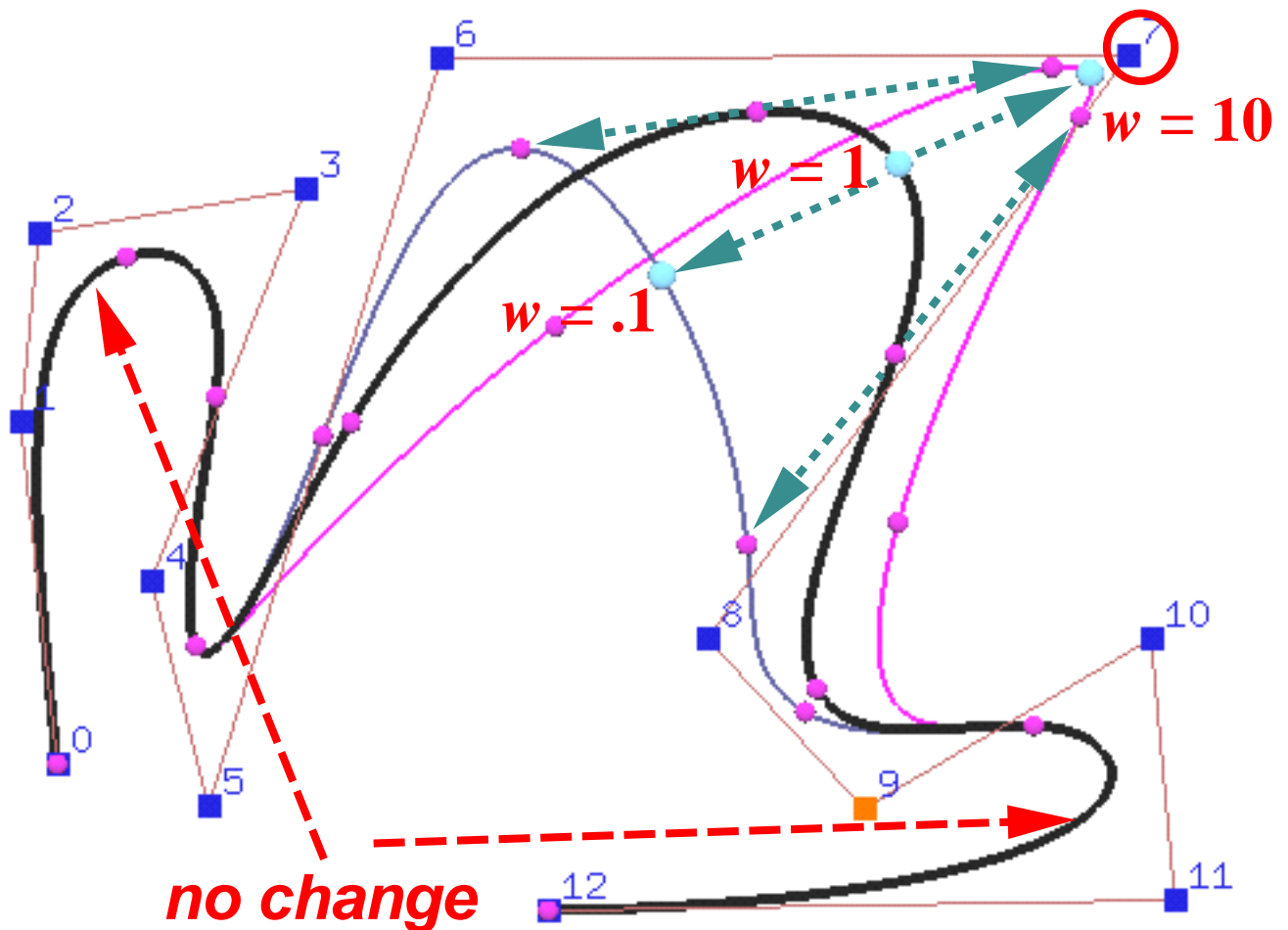
Quadric Surface Modeling



NURBS Curves

- A 3D B-spline curve using homogeneous coordinate is a 4D B-spline curve.
- Projecting this 4D B-spline back to 3D with $w = 1$, the 4th coordinate, yields a NURBS curve.
- Since a NURBS curve is rational, circles, ellipses and hyperbolas can be represented.
- Most non-metric related properties of B-spline translate to NURBS (e.g., de Boor's algorithm, strong convex hull, local modification, and so on)
- Hence, you know NURBS if you know B-spline well.

Weight Change with DesignMentor



Increasing weight pulls the curve toward the selected control point. Decreasing weight pushes the curve away from the selected control point.

Bezier, B-spline, NURBS Surfaces

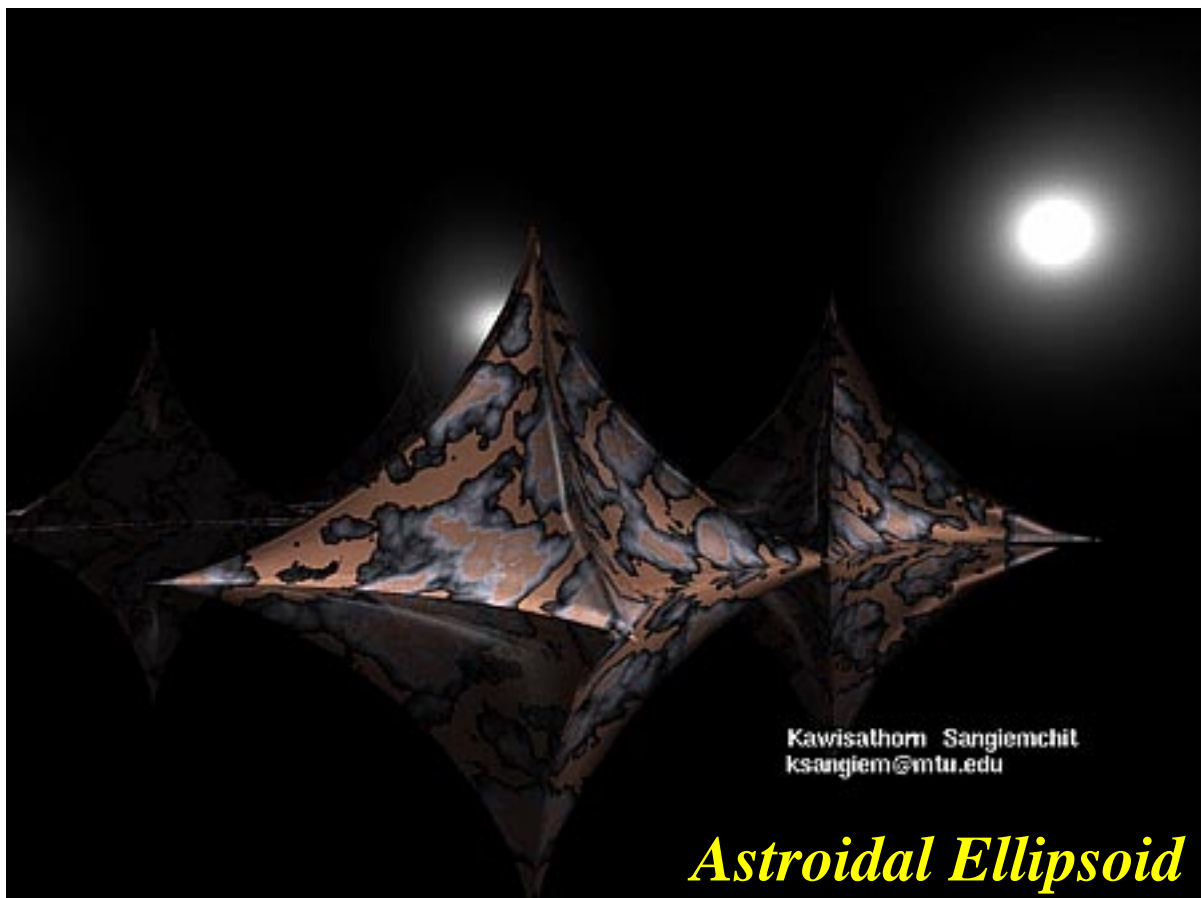
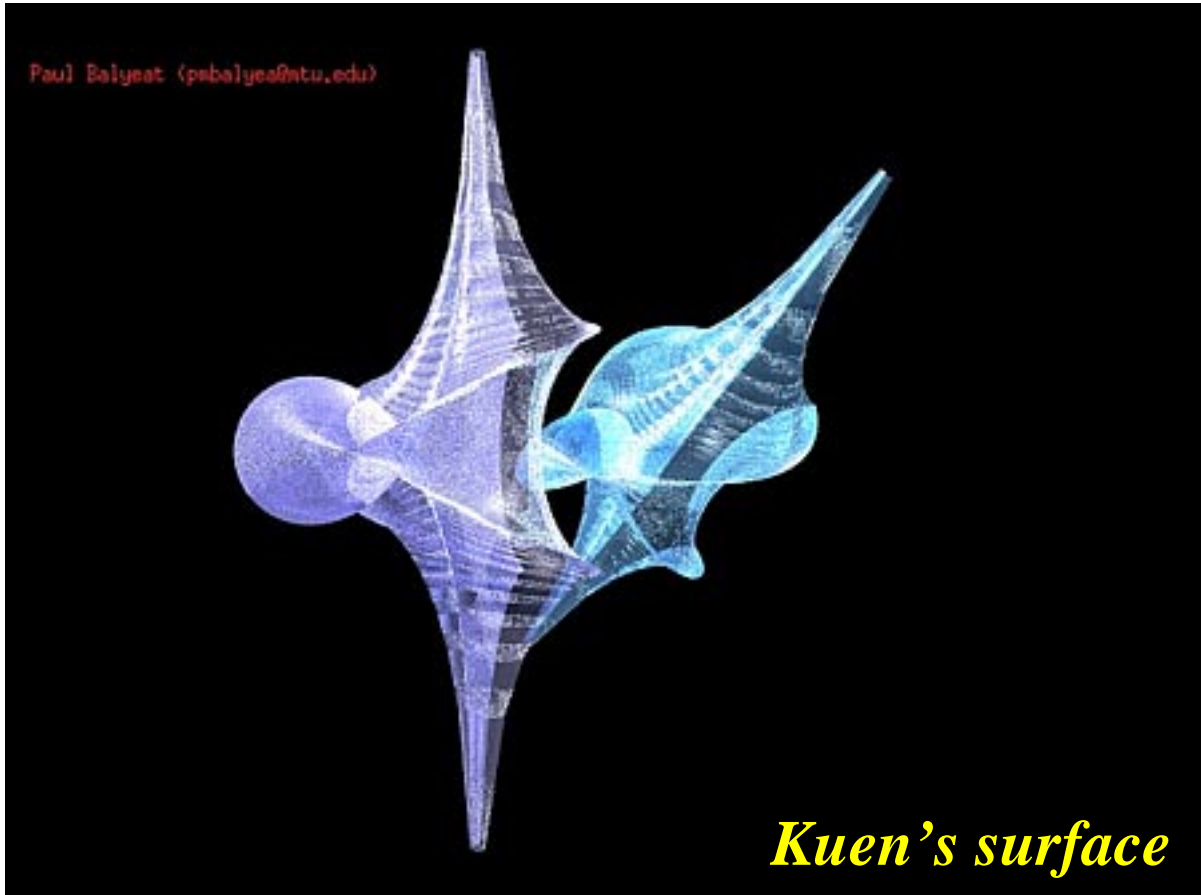
- A Bezier, B-spline and NURBS surface is a tensor product surface and is the *product* of *two* curves.
- Surfaces are defined by a grid and have two sets of parameters, two sets of knots and so on.
- De Casteljau's and de Boor's algorithms can easily be generalized for surfaces.
- Other algorithms such as knot insertion are applied to one set of parameters.

Programming Exercise 5

- Implement the 3D de Boor's algorithm and use it to trace a B-spline surface. **Note that adding a weight to each point making a B-spline surface a NURBS one. In OpenGL, this is the *only* difference.**
- Triangulate the domain of a parametric surface and display it. Then, use this surface to design an interesting scene.

Students are *only* given a set of parametric equations. They do not know what the surface looks like.

Parametric Surface Triangulation



Interpolation and Approximation

● Interpolation

Find a B-spline curve (*resp.*, surface) that contains all data points (*resp.*, data point grid)

● Approximation

Find a B-spline curve (*resp.*, surface) that can follow the shape of the data points (*resp.*, data point grid) as close as possible.

● This topic is difficult and requires a linear equation solver. Most students managed to get it. Some performed extremely well.

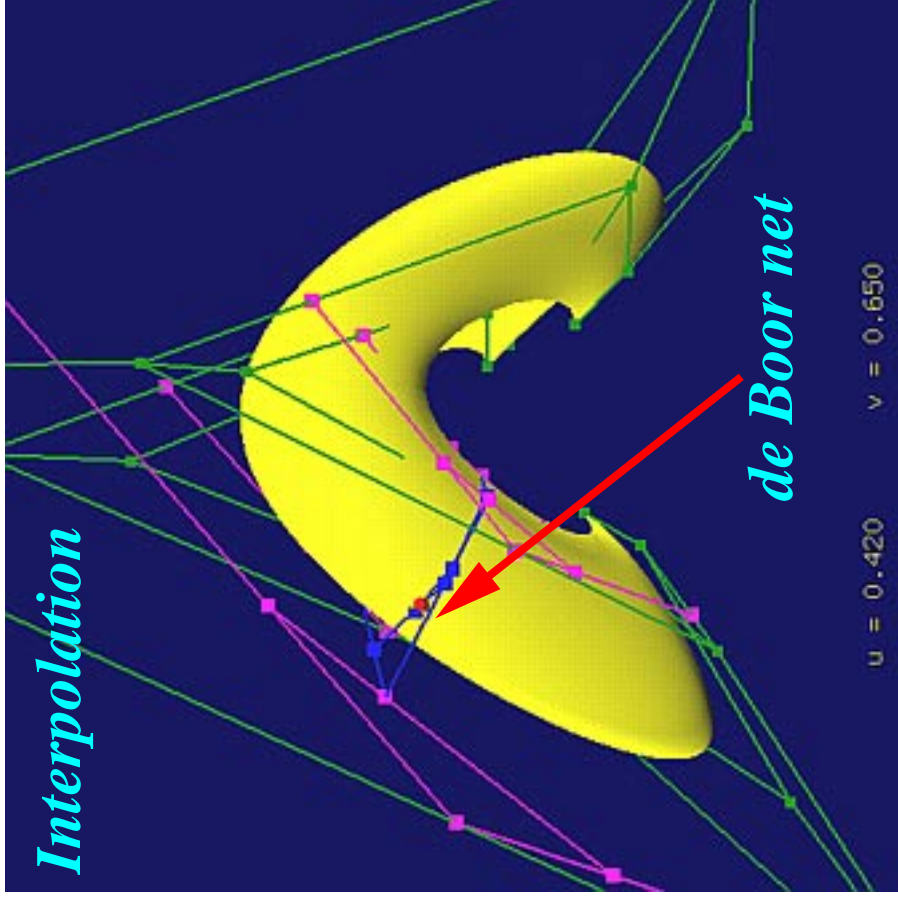
● This topic is also very useful in graphics (*e.g.*, animation path generation) and in other areas (*e.g.*, reverse-engineering)

Programming Exercise 6

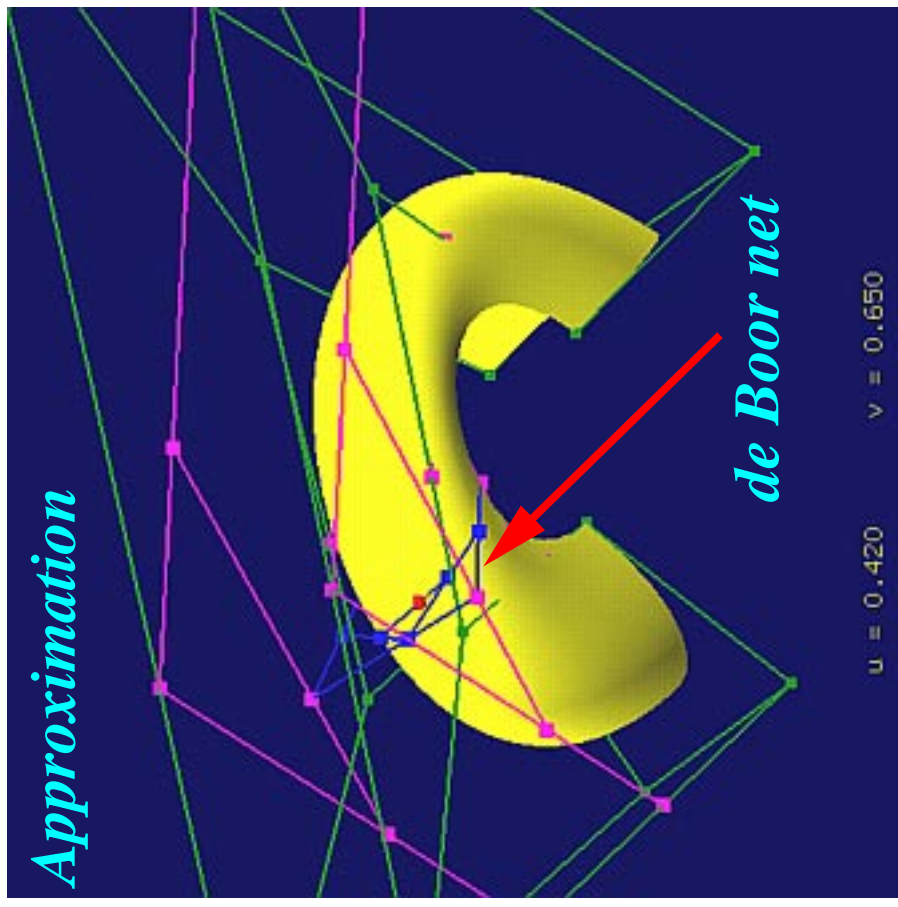
- Design a program that can interpolate and approximate an input data grid using B-spline surfaces and different parameter selection methods.
- Render an algebraic surface and use it to design an interesting scene. Students only receive an equation and do not know what the surface looks like.

This surface has a free parameter. Students should explore the relation between this parameter and the shape of this surface.

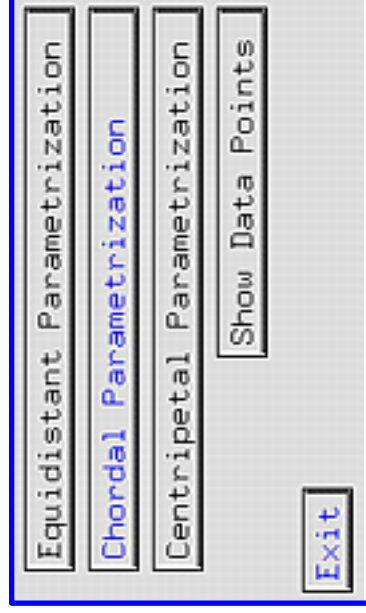
Interpolation



Approximation



*parameter selection
methods*

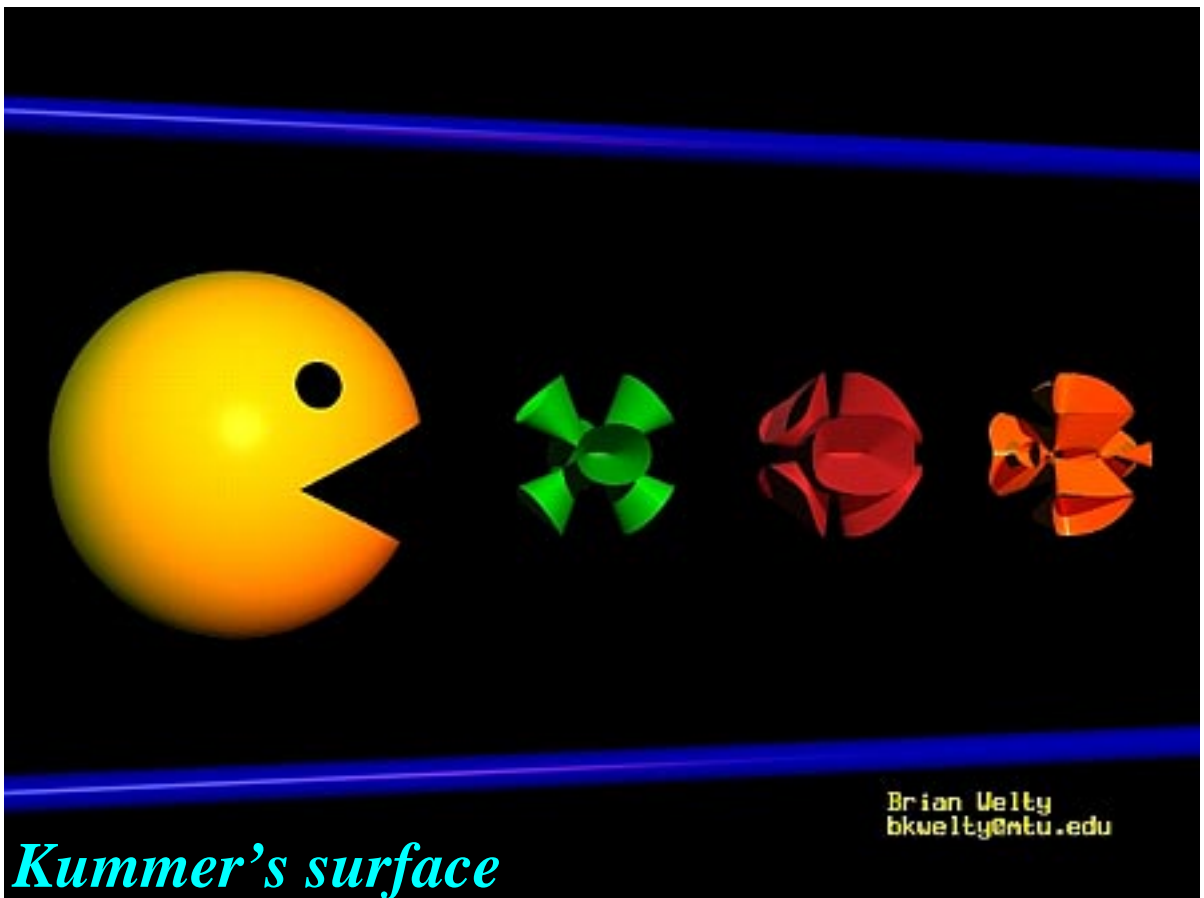


Algebraic Surface Modeling

Ernie Fessenden ewfessen@mtu.edu



Kummer's surface



Brian Uelty
bkuelty@mtu.edu

Kummer's surface

Course Evaluation

- Since this is an elective course, students in this course just love it.
- The difference between a pre-test and a post-test shows that students do learn a lot. Compared with their exam papers and exercises, this is a fair assessment.
- Almost all students prefer this intuitive, elementary and learning-by-doing approach.
- Since students receive a mini environment to work with, they *do not* have to know about rendering. In fact, 40% of them do not have graphics background.
- A detailed behavior-based study will be published elsewhere.

Conclusion

- **We believe we should provide our students with some geometry skills.**
- **Based on our experience, this course is a success, especially that there are so many off-campus visitors and software downloads.**
- **We still need more materials for instructors and students to make choices. Good candidates include the blossoming principle, scattered data interpolation/approximation, multiresolution modeling, geometric compression and so on.**
- **All materials are available at**

<http://www.cs.mtu.edu/~shene>