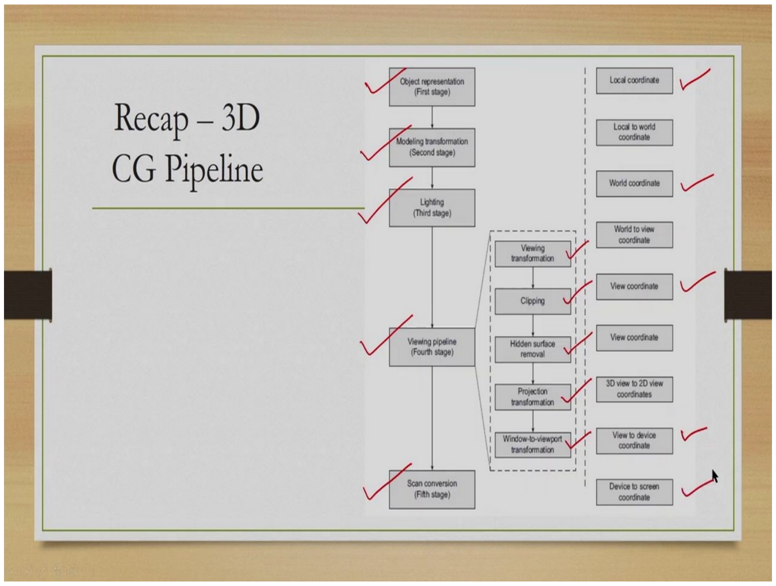
Computer Graphics

Professor. Doctor. Samit Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture No. 07
NPTEL-MOOCS L7

Hello and welcome to lecture number 7 in the course Computer Graphics so far we have covered 6 lectures and we are discussing the graphics pipeline. Before we go into today's topic, let us recap the pipeline quickly.

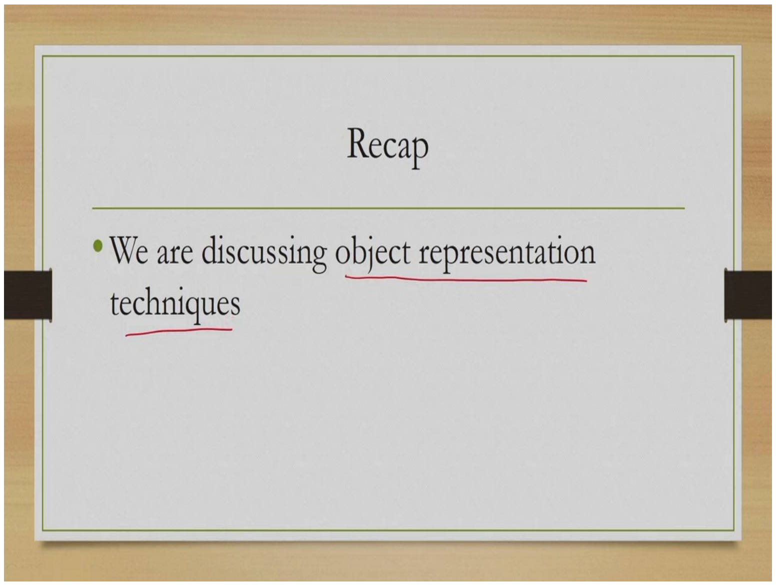
(Refer Slide Time: 0:49)


Recap – 3D CG Pipeline
Object representation (First stage)
Modeling transformation (Second stage)
Lighting (Third stage)
Viewing pipeline (Fourth stage)
Scan conversion (Fifth stage)
Viewing transformation
Clipping
Hidden surface removal
Projection transformation
Window-to-viewport transformation
Local coordinate
Local to world coordinate
World coordinate
World to view coordinate
View coordinate
View coordinate
3D view to 2D view coordinates
View to device coordinate
Device to screen coordinate

So, as we have already mentioned, there are 5 stages in the pipeline, first stage is object representation, second stage is modelling transformation, third stage is lighting or colouring. Fourth stage is a composition of sub stages and sub pipelines. This is called viewing pipeline, which consists of a sub pipeline which has 3 stages and 2 operations, viewing transformation, clipping, hidden surface removal, projection transformation. And window to viewport transformation. And the final stage of the pipeline is scan conversion or rendering.

And these stages take place in different coordinate systems, starting from local or object coordinate system transitioning through world coordinate, view coordinate, device coordinate to finally the screen coordinate system.

(Refer Slide Time: 2:04)

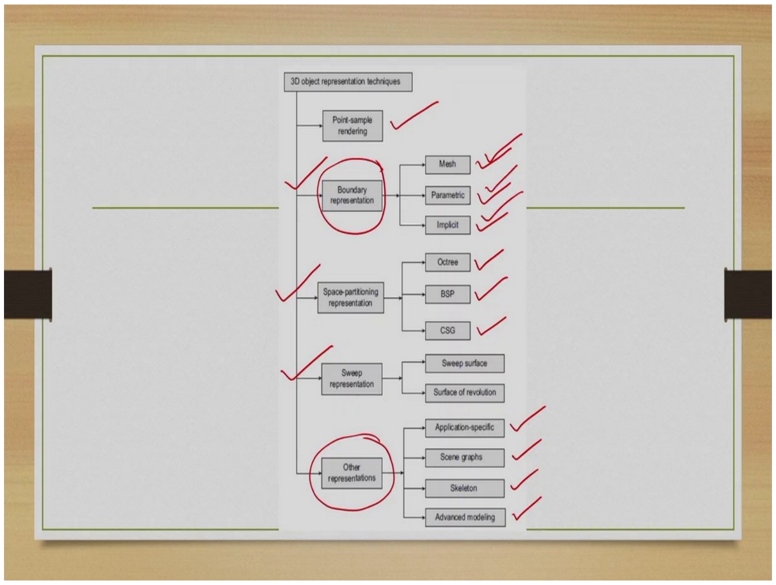


Recap

- We are discussing object representation techniques

Now, among these stages, we are currently discussing the first stage object representation techniques.

(Refer Slide Time: 2:15)

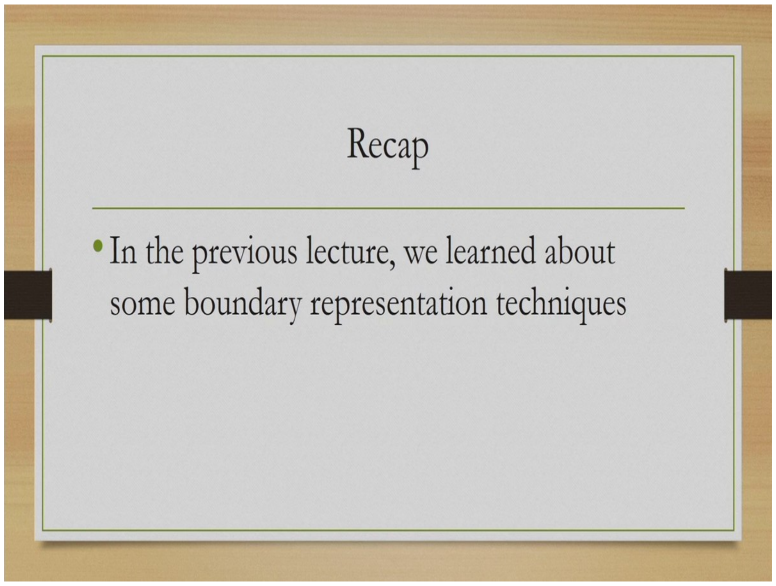


As we have already discussed in the object representation techniques there are broadly 5 categories, one is point sample rendering. The second one is boundary representation, then space partitioning, then sweep representation, and finally some other specific representation techniques which are application specific or referred to some advanced techniques such as scene graphs, skeletal model and other advanced modelling techniques such as fractal representation and particle systems.

In the boundary representation, their 3 broad group of techniques. One is mesh representation, one is parametric representation, and one is implicit representation. Similarly, in space partitioning representation, there are 3 broad techniques octree based representation BSP or binary space partitioning trees and CSG techniques. Now, among all these, we are currently discussing the boundary representation techniques and we will continue our discussion on this technique.

In the boundary representation techniques. In the last lecture, last couple of lectures, we have covered mesh representation and introduced the idea of parametric as well as implicit representation.

(Refer Slide Time: 3:50)

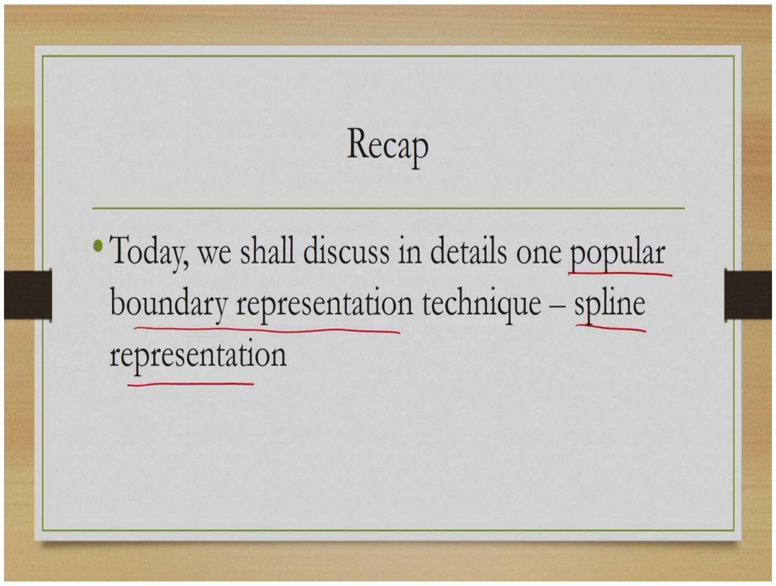


Recap

- In the previous lecture, we learned about some boundary representation techniques

These are some of the boundary representation techniques that we have introduced in the last lecture.

(Refer Slide Time: 3:58)

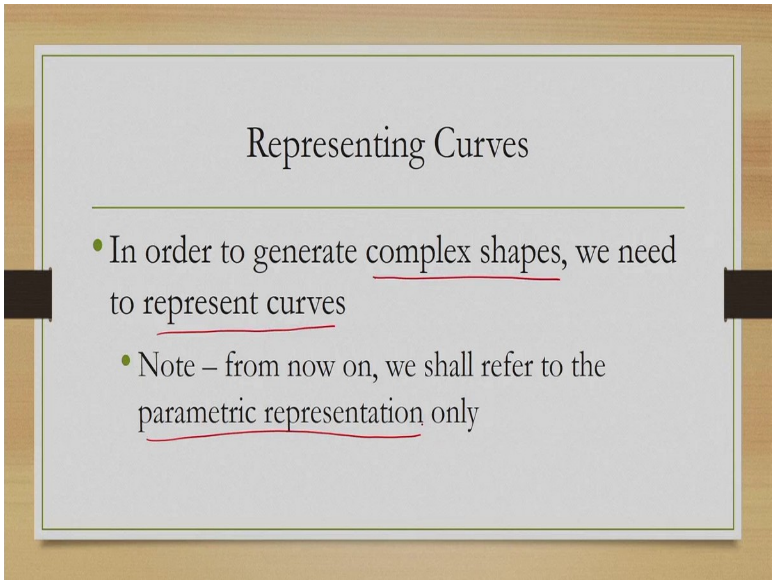


Recap

- Today, we shall discuss in details one popular boundary representation technique – spline representation

Today we will continue our discussion on boundary representation today we will focus on one specific and popular boundary representation technique, which is called spline representation.

(Refer Slide Time: 4:14)



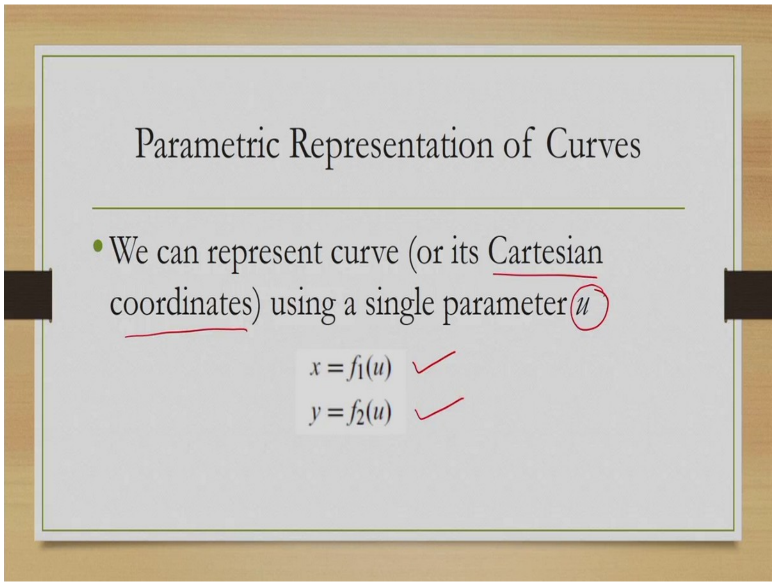
Representing Curves

- In order to generate complex shapes, we need to represent curves
  - Note – from now on, we shall refer to the parametric representation only

So, in order to understand the spline representation technique, we need to understand how we represent curve. Curve is very common, primitive shape which is required at many places to represent objects, particularly in the context of complex shapes we cannot avoid representing curves only with lines or points, it may not be possible to represent complex shapes, and we have to take into account curves.

To simplify our discussion will focus here only on parametric representation of curves, although earlier we have introduced both the types, namely parametric representation and implicit representation.

(Refer Slide Time: 5:10)



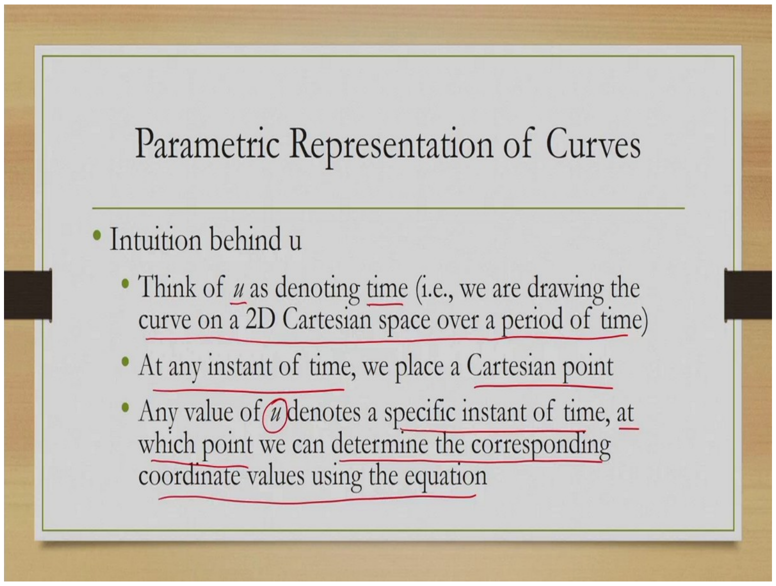
Parametric Representation of Curves

- We can represent curve (or its Cartesian coordinates) using a single parameter u

$$x = f_1(u) \checkmark$$
$$y = f_2(u) \checkmark$$

How we can representation curves in general using parametric form, we can use a single parameter u will denoted by u to represent curves or its Cartesian coordinates using these equations. This one is for representing the X coordinate. The other one is for representing the Y coordinate where X is a function of u and Y is another function of u. Let us try to understand the intuition behind this representation.

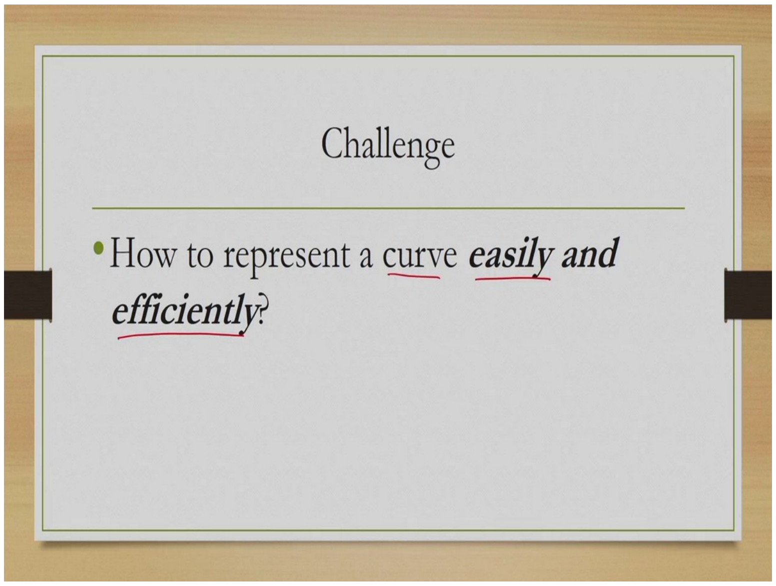
(Refer Slide Time: 5:47)


Parametric Representation of Curves

- Intuition behind u
  - Think of *u* as denoting time (i.e., we are drawing the curve on a 2D Cartesian space over a period of time)
  - At any instant of time, we place a Cartesian point
  - Any value of *u* denotes a specific instant of time, at which point we can determine the corresponding coordinate values using the equation

We can assume that u is denoting time. We can think of it in this way, we are drawing the curve on a 2D Cartesian space over a period of time now at an instant of time, we place a Cartesian point. Then we can say that at that point of time the Cartesian point is that, in other words the Cartesian point is characterized by the instant of time, which is u. So, essentially u denotes specific instant of time, at which point we can determine the corresponding coordinate values using the equation. This is the simple intuition behind the idea of parametric representation of a curve.

(Refer Slide Time: 6:51)

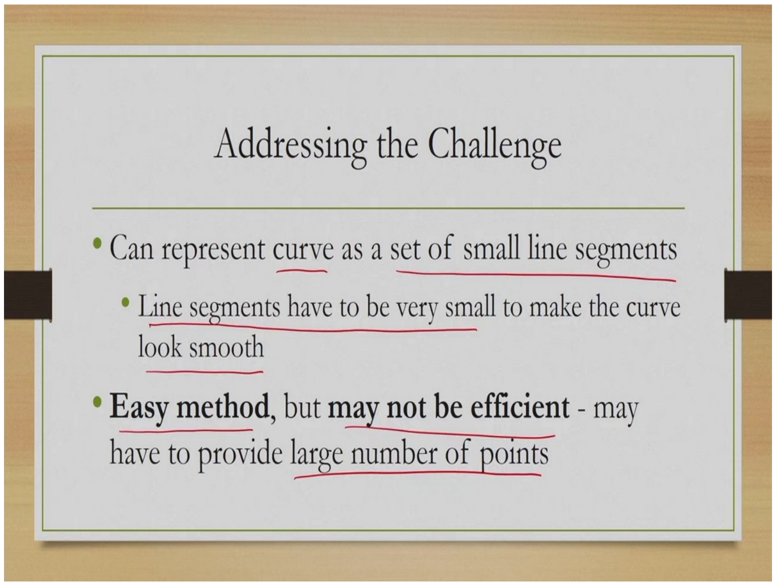


Challenge

- How to represent a curve ***easily and efficiently***?

So, that is about understanding how to representation curve parametrically. Now, our objective is to represent the curve easily and efficiently. Let us elaborate on this a little bit more.

(Refer Slide Time: 7:07)

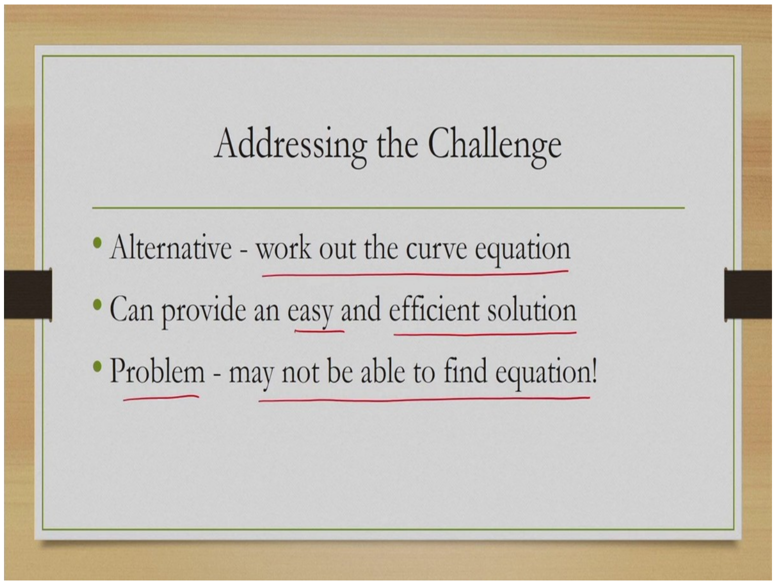


Addressing the Challenge

- Can represent curve as a set of small line segments
  - Line segments have to be very small to make the curve look smooth
- **Easy method**, but **may not be efficient** - may have to provide large number of points

As we all know, we can approximate a curve in terms of a set of small line segments, of course here the segments have to be very small to make the curve look smooth, otherwise the curve make it a jagged appearance. Now, clearly this is very easy, intuitive but may not be efficient. We may have to provide a large number of points to draw small lines judgments.

(Refer Slide Time: 7:49)

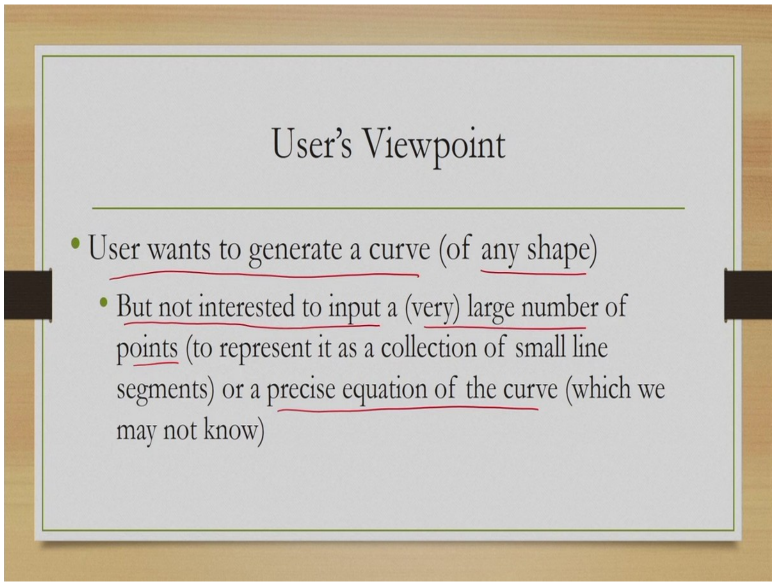


Addressing the Challenge

- Alternative - work out the curve equation
- Can provide an easy and efficient solution
- Problem - may not be able to find equation!

There may be another alternative. We can work out the curve equation and apply the equation to find out any point on the curve. So, this clearly is better than specifying manually large number of points to approximate the curve in the form of a set of line segments. So, clearly this is easy and may turn out to be efficient also. But the problem here is that for many curves we may not be able to find the equation itself. It is very difficult for any arbitrarily set curve to find out the curve question.

(Refer Slide Time: 8:38)



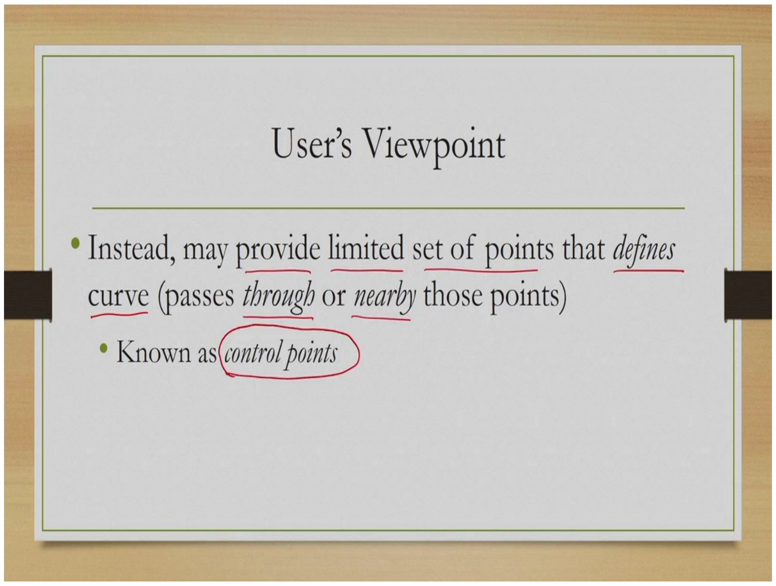
User's Viewpoint

- User wants to generate a curve (of any shape)
  - But not interested to input a (very) large number of points (to represent it as a collection of small line segments) or a precise equation of the curve (which we may not know)

So, let us try to understand these problems from that point of view of a user, what the user thinks. And what are the problems that are user faces. Now user wants to generate a curve of any arbitrary shape. If we are trying to represent the curve in the form of a large number of small line segments, then user has to input a very large number of those points through which the line segments can be generated. Clearly, no user would be interested to input a very large number of such points.

On the other hand, for a user. It may be difficult or even impossible to find out a precise equation of the curve. So, therefore in both the approaches user is not going to be benefited.

(Refer Slide Time: 9:48)



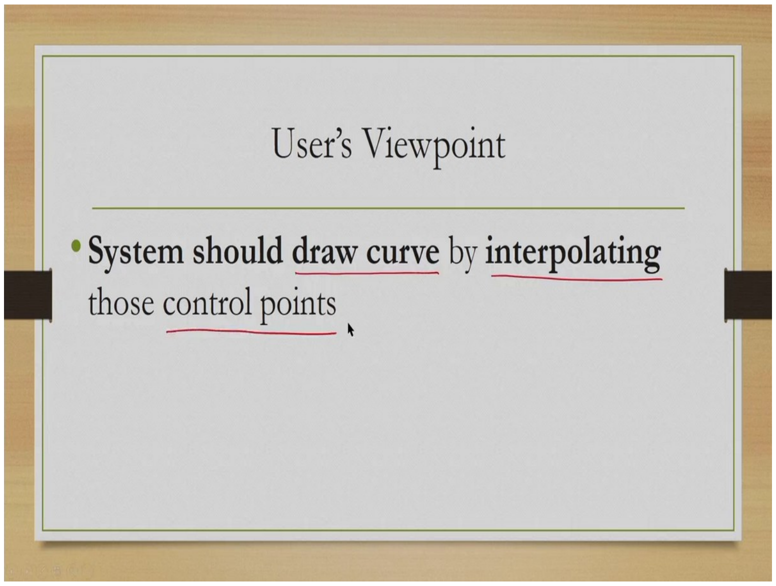
User's Viewpoint

- Instead, may provide limited set of points that *defines* curve (passes *through* or *nearby* those points)
  - Known as *control points*

Ideally, what a user should do or what a user wants to do, user wants to provide a limited set of points. Now, these points define the curve. So, essentially user is not providing all possible line segments to approximate the curve or providing a precise equation to find out points on the curve. Instead, user is providing a small or limited set of points which defines the curve. In other words, these points a chosen such that the curve passes through or nearby those points, these points are also known as control points.

So, the alternative to the user is to provide a small set of control points instead of providing large set of points through which line segments can be drawn or give a precise curve equation. So, user has provided set of control points.

(Refer Slide Time: 11:05)

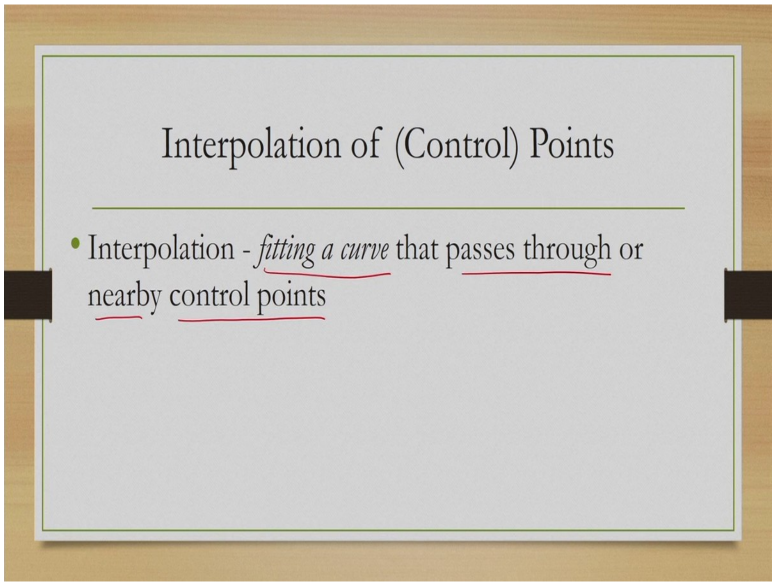


User's Viewpoint

- **System should draw curve** by **interpolating** those control points

And the user expects the system to draw the curve by interpolation by interpolating those control points. So, let us try to briefly understand what is the idea of interpolation, many of you or maybe all of you may already know what is interpolation, but there is no harm in refreshing our knowledge.

(Refer Slide Time: 11:31)

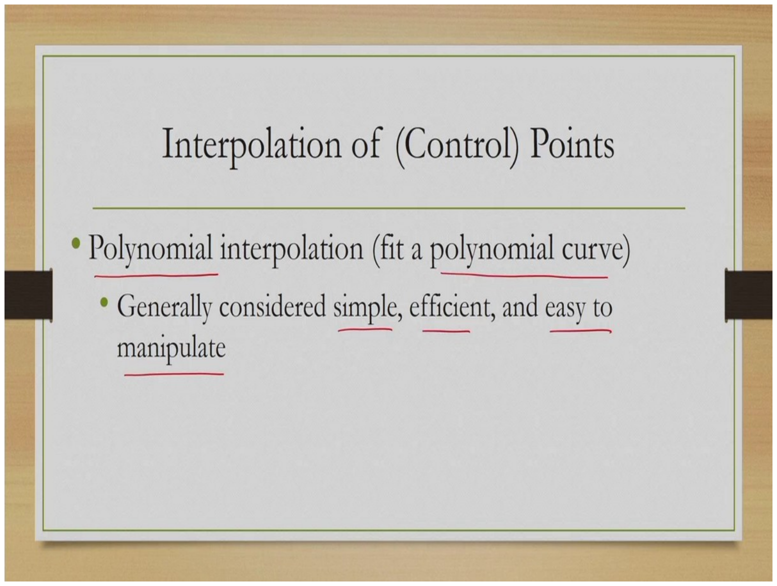


Interpolation of (Control) Points

- Interpolation - *fitting a curve* that passes through or nearby control points

So essentially, when we talk of interpolation, what we mean, we essentially mean by interpolation fitting of a curve that passes through or nearby the set of points provided or the control points.

(Refer Slide Time: 11:49)

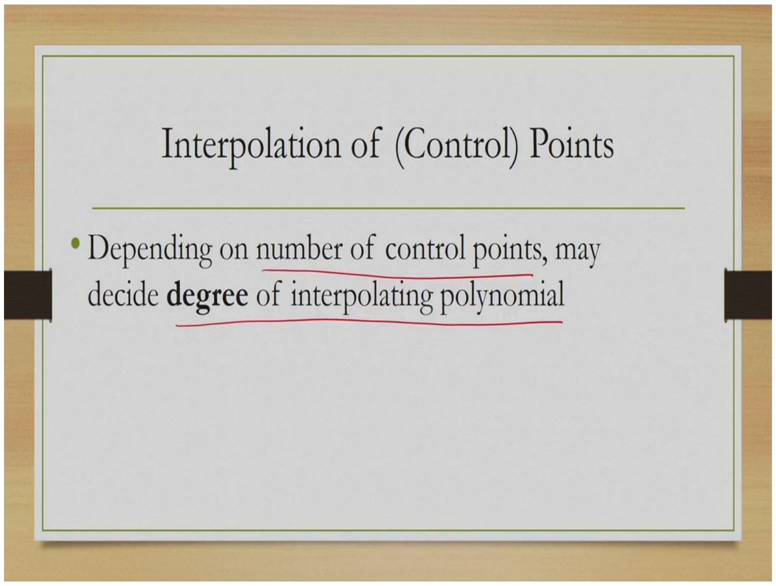


Interpolation of (Control) Points

- Polynomial interpolation (fit a polynomial curve)
  - Generally considered simple, efficient, and easy to manipulate

One form of interpolation is polynomial interpolation. In this interpolation what we do, we try to fit a polynomial curve through the given set of control points. Now, polynomial interpolation is very popular because it is generally considered that such interpolations are simple, efficient and easy to manipulate. So, we will focus here on polynomial interpolation.

(Refer Slide Time: 12:27)

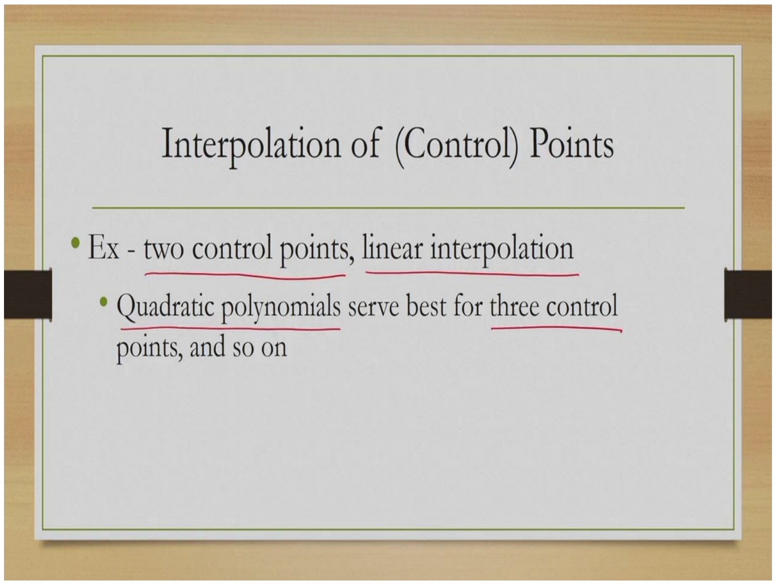


Interpolation of (Control) Points

- Depending on number of control points, may decide **degree** of interpolating polynomial

Now, depending on the number of control points, the degree of the interpolating polynomial is decided. So, when we talk a polynomial interpolation, one concern is what should be the degree of the polynomial now that can be decided based on the number of control points provided.

(Refer Slide Time: 12:58)

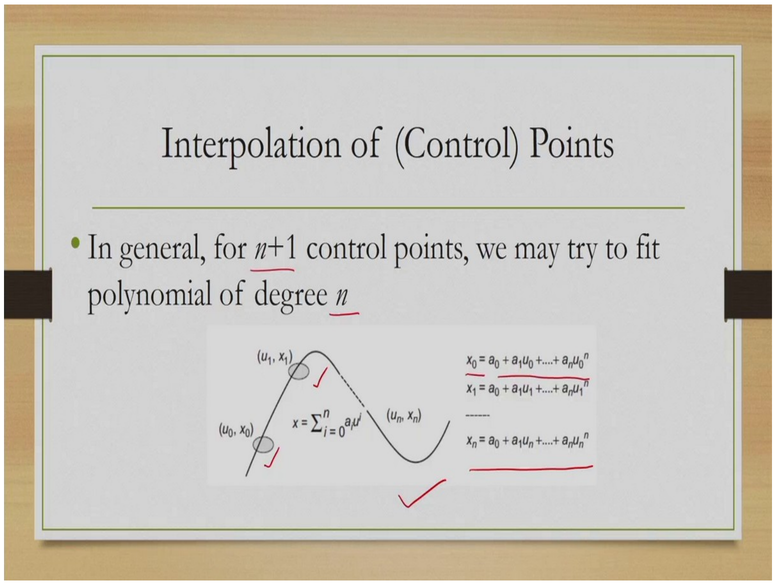


Interpolation of (Control) Points

- Ex - two control points, linear interpolation
  - Quadratic polynomials serve best for three control points, and so on

Let us take an example, suppose we are given 2 control points in such a situation, it is advisable to go for linear interpolation rather than any other higher form of interpolation because we have only two control points. Similarly, if there are 3 control points, then we can go for quadratic polynomials. There are 4 control points which use the degree accordingly and so on.

(Refer Slide Time: 13:32)



Interpolation of (Control) Points

- In general, for $\underline{n+1}$ control points, we may try to fit polynomial of degree $\underline{n}$

(u_1, x_1) ✓
(u_0, x_0) ✓
$x = \sum_{i=0}^{n} a_i u^i$
(u_n, x_n)
✓

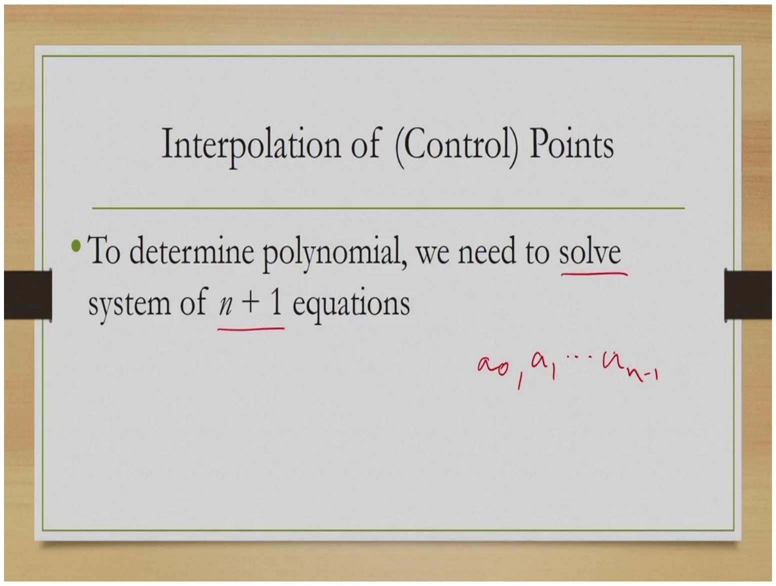
$$x_0 = a_0 + a_1 u_0 + \ldots + a_n u_0^n$$
$$x_1 = a_0 + a_1 u_1 + \ldots + a_n u_1^n$$

$$x_n = a_0 + a_1 u_n + \ldots + a_n u_n^n$$

Therefore, we can say that in general for n+1 control points, we may try to fit polynomial of degree n which is pictorially depicted here in this figure, we are given these control points through which we are trying to fit a curve. And if the number of control points is n+1, then the curve that we should work with or the polynomial that we should work with should have degree n ideally. Note at the system of equations that we have mentioned here.

This is for X coordinate, similarly for Y coordinate, we can have a similar set of systems. Now since they are n control points given we have n x coordinate values for each of these coordinates. We have 1 equation of the curve in terms of the parameter and so for the n number of control points, we have n number of equations.

(Refer Slide Time: 14:49)



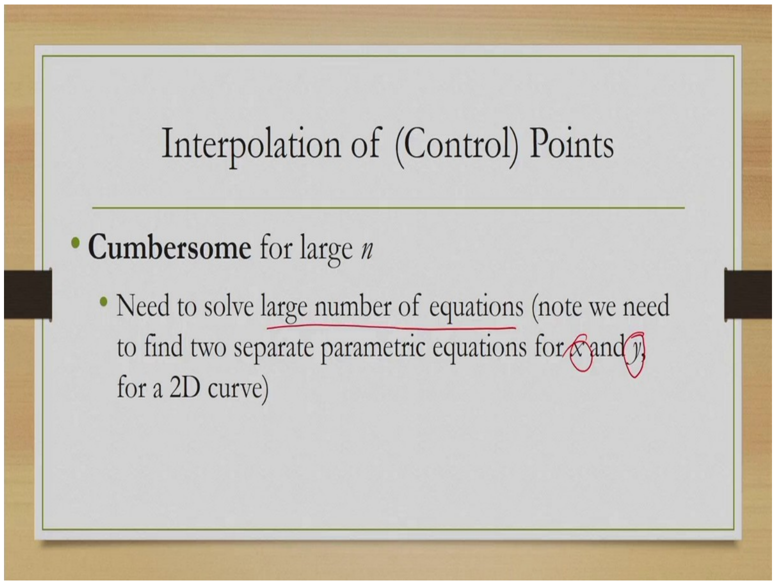
Interpolation of (Control) Points

- To determine polynomial, we need to solve system of $n + 1$ equations

$a_0, a_1 \cdots a_{n-1}$

Now, in those are equations there are constant terms, those are the coefficients like a_0, a_1 to a_{n-1}. If we decide these coordinates, then we can define the polynomial. So, to get the values of this coordinate these coefficients what we need to do, we need to solve the set of equations. The n plus one equations that we have seen earlier. If we solve this, then we will get these values of the coefficients which defines the polynomial.

(Refer Slide Time: 15:36)

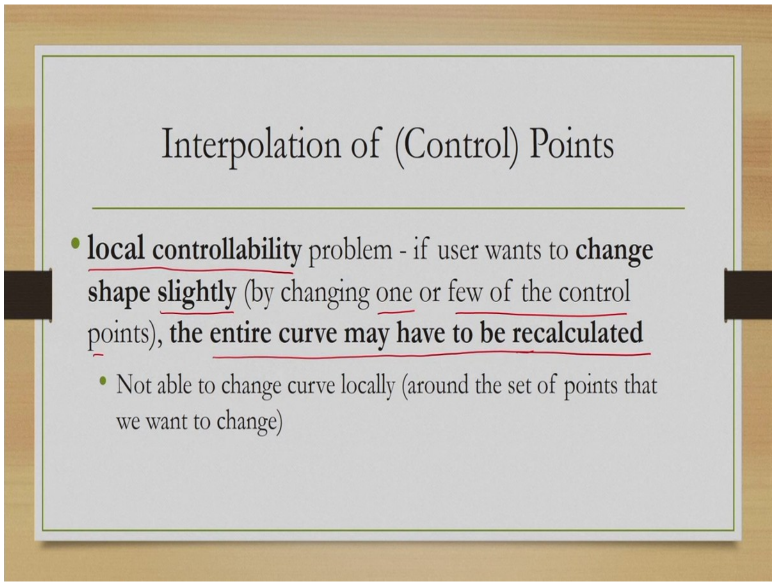


Interpolation of (Control) Points

- **Cumbersome** for large n
  - Need to solve large number of equations (note we need to find two separate parametric equations for x and y, for a 2D curve)

But there is one problem, if we have a very large n if we have many control points, a large number of n. Then we need to solve a very large number of equations, which is not easy. On top of it we need to keep in mind that there are two separate sets of equations, one for X and one for Y. So, we need to solve actually two sets of equations rather than one and for large and, this becomes very cumbersome to do.

(Refer Slide Time: 16:24)



Interpolation of (Control) Points

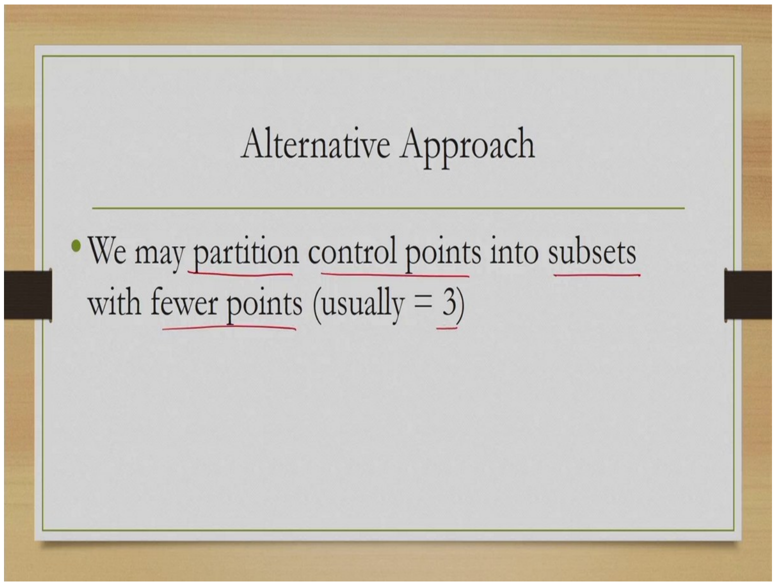
- **local controllability** problem - if user wants to **change shape slightly** (by changing one or few of the control points), **the entire curve may have to be recalculated**
  - Not able to change curve locally (around the set of points that we want to change)

Along with that, there is one more problem, which is called local controllability issue. Suppose you or the user wants to change the shape slightly. So, with the polynomial equation will get a curve which represents a shape.

Now, I want to change it slightly. Then ideally, what should I do? A change of one or few of the control points to denote the small change. But if we go for polynomial interpolation, then to get the new curve, we may have to recalculate the entire thing again. So, entire curve may have to be recalculated. Which is, of course not a good thing because we have changed a few points and ideally we should be able to restrict our pre calculations effort to those few points only, but instead we have to solve the entire set of equations again, which is not an efficient approach.

So, this problem is known as local controllability, where we are unable to control local changes locally. We have to control local changes through global recalculation of the curve. Now, in order to address these issues, there is another solution which we will discuss.

(Refer Slide Time: 18:27)

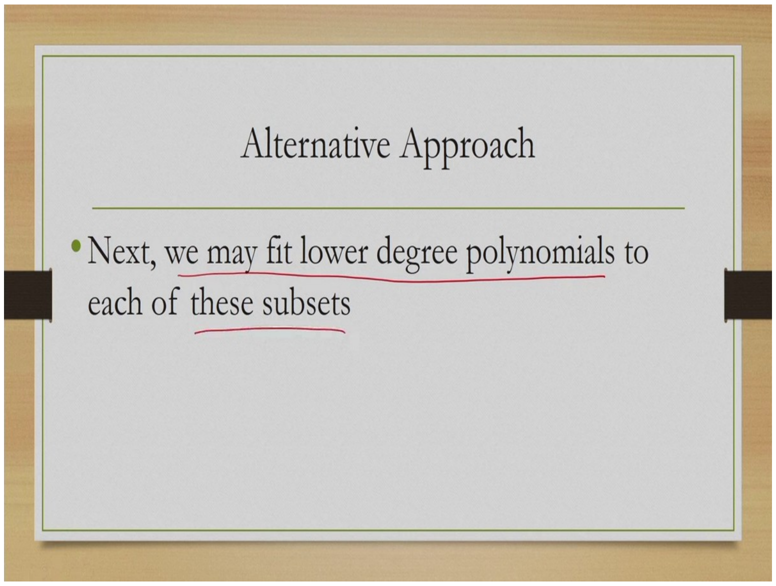


Alternative Approach

- We may partition control points into subsets with fewer points (usually = 3)

Now, what is this alternative approach? Suppose we are given again n plus one control points irrespective of the value of n, we may partition the entire set into subsets with fewer points. Typically, these fewer points at 3. So, given a set of n plus one points, we may like to have subsets where each subset contains three control points.

(Refer Slide Time: 19:00)

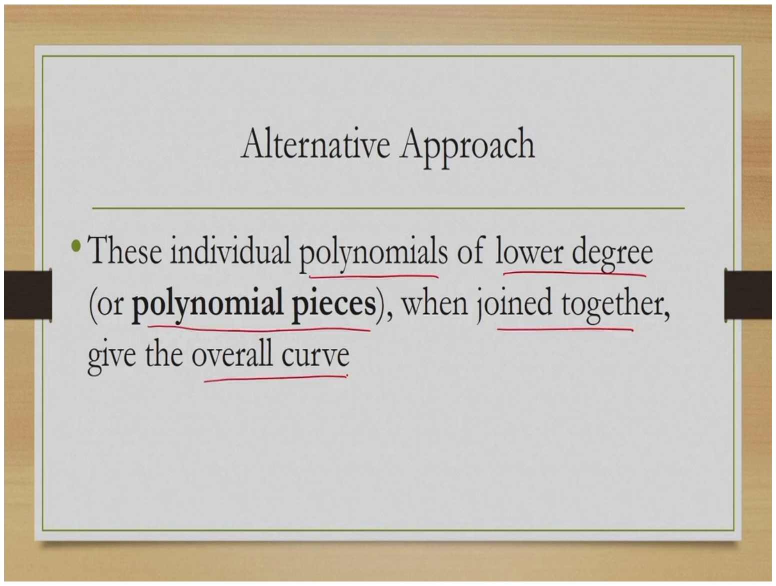


Alternative Approach

- Next, we may fit lower degree polynomials to each of these subsets

Now for each of these subsets we may fit lower degree polynomials. In this case, the degree 2 polynomials for each of the subsets.

(Refer Slide Time: 19:17)



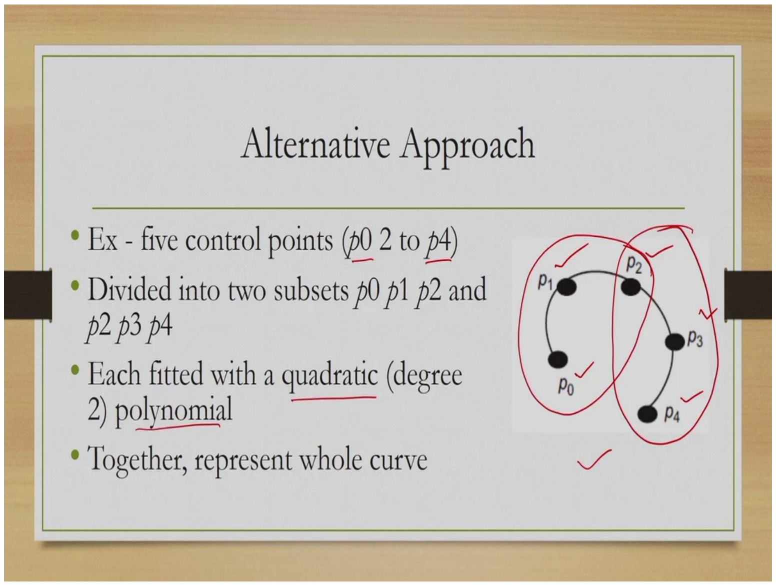
Alternative Approach

- These individual polynomials of lower degree (or **polynomial pieces**), when joined together, give the overall curve

And then these individual polynomials of lower degree, which are also called polynomial pieces, when they join together, they give the overall curved. So, the idea is very simple. You are given a large number of control points, but it is not necessary to fit a single polynomial curve using the entire set of control points. Instead, what we do, we divide the entire set of control points into subsets of smaller numbers. Each subset contains very few control points.

Typical value used is three and for each of these subsets we fit or interpolate a smaller degree polynomial. And these polynomials, when they join together, they give the overall curved. So, this individual polynomials are also known as polynomial pieces. So, the entire curve we are representing in terms of polynomial pieces.

(Refer Slide Time: 20:23)



Alternative Approach

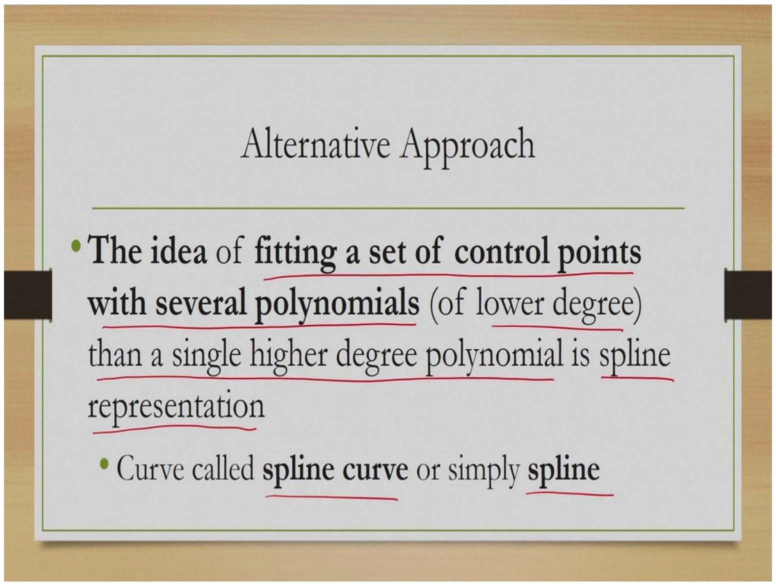
- Ex - five control points (*p*0 2 to *p*4)
- Divided into two subsets *p*0 *p*1 *p*2 and *p*2 *p*3 *p*4
- Each fitted with a quadratic (degree 2) polynomial
- Together, represent whole curve

p_1
p_2
p_3
p_0
p_4

Let us take an example, consider this figure here. There are 5 control points p0 to p4 as you can see, p0, p1, p2, p4 p3 and p4. Now, these 5 points need not be used to draw a single polynomial. Which in this case would be of degree 4 instead what we can do, we can subdivide the curves or the set of control points into subsets. Like the two subsets shown here in one subset, we have three control points p0, p1, p2 another subset we have another 3 control points p2, p3, p4

For each of these subsets, we draw a quadratic or degree 2 polynomial and then when they join together, we get the overall interpolated curve. That is the basic idea.

(Refer Slide Time: 21:40)

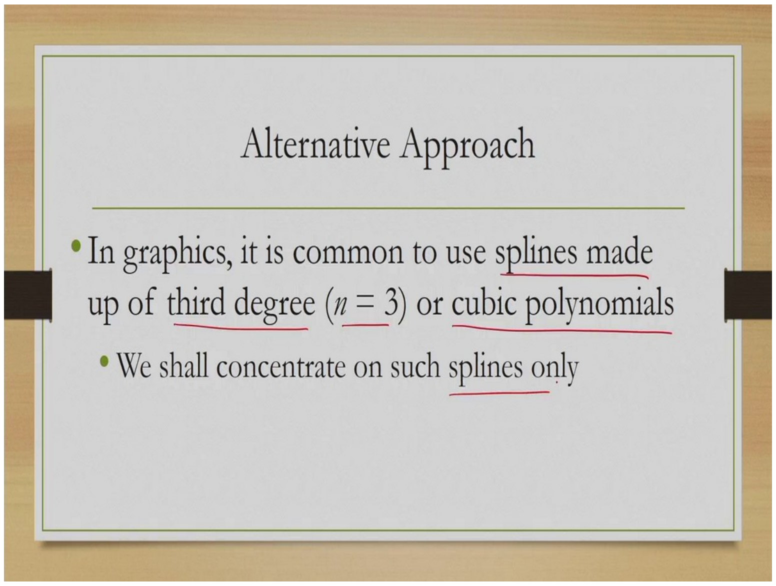


Alternative Approach

- **The idea** of **fitting a set of control points with several polynomials** (of lower degree) than a single higher degree polynomial is spline representation
  - Curve called **spline curve** or simply **spline**

Now this idea of fitting a set of control points with several polynomials of lower degree than a single higher degree polynomial is known as spline representation. So, when we talk of spline representation, we are essentially referring to the fact that there is a set of control points, but we are not interpolating the entire set with a single polynomial curve. Instead, we are representing it in terms of several polynomial pieces. Now the entire carve is called spine curve simply spline. This is a very popular curve representation technique used in computer graphics.

(Refer Slide Time: 22:32)

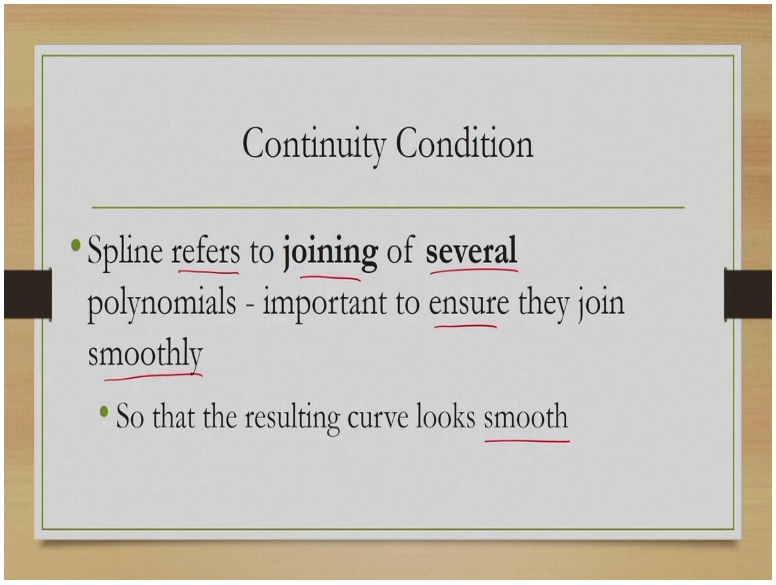


Alternative Approach

- In graphics, it is common to use splines made up of third degree ($n = 3$) or cubic polynomials
  - We shall concentrate on such splines only

In graphics, it is very common to use splines made of third degree or n = 3 polynomials, also known as cubic polynomials. In our subsequent discussion, we concentrate. We will concentrate on these polynomials only and corresponding splines only. There is one important thing in spline representation that we have to keep in mind that is called continuity condition.

(Refer Slide Time: 23:10)

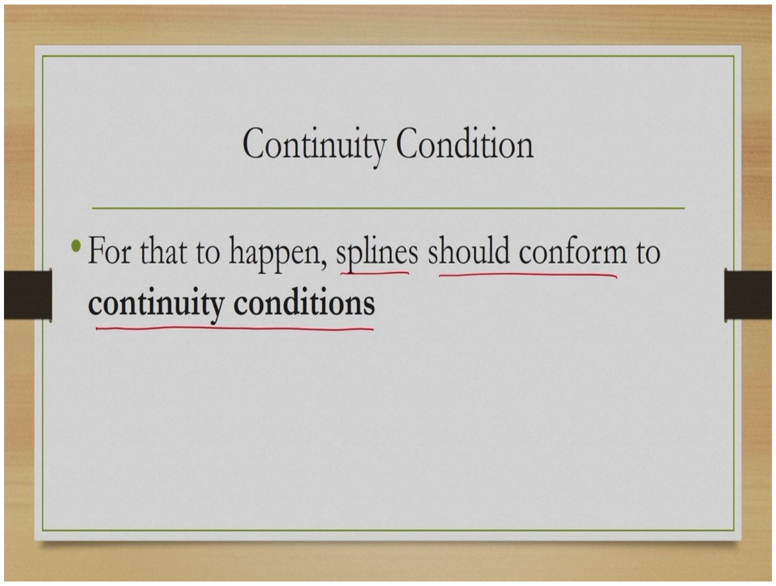


Continuity Condition

- Spline refers to **joining** of **several** polynomials - important to ensure they join smoothly
  - So that the resulting curve looks smooth

Now splines as we have discussed, refers to joining of several polynomials. So, clearly it is important to ensure that they joint smoothly. To make the resulting curve look smooth.

(Refer Slide Time: 23:32)

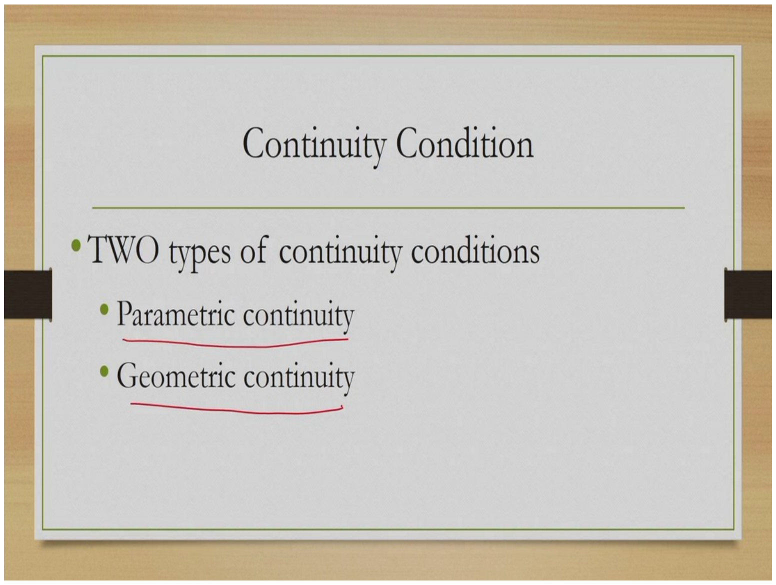


Continuity Condition

- For that to happen, splines should conform to **continuity conditions**

Now, how to ensure that? In order to ensure that to happen, splines must conform to what is known as continuity condition.

(Refer Slide Time: 23:50)

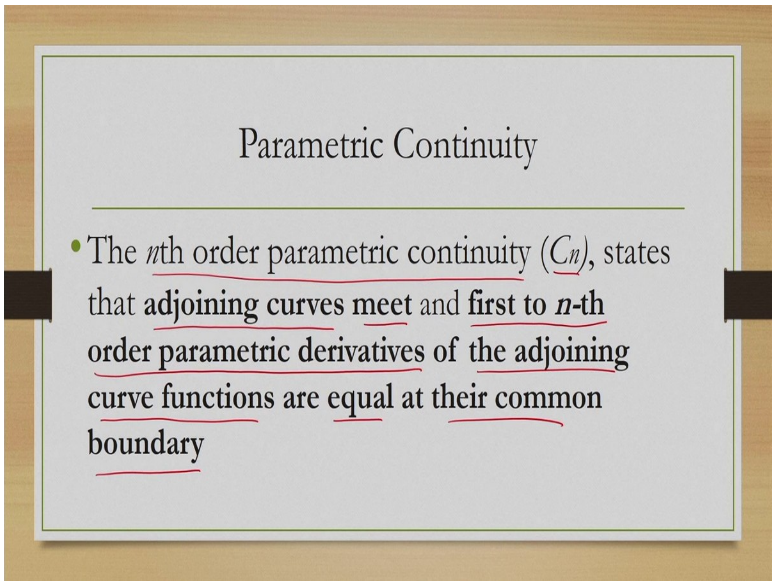


Continuity Condition

- TWO types of continuity conditions
  - Parametric continuity
  - Geometric continuity

There are several such conditions broadly, they are of two types, one is parametric continuity condition and the other one is geometric continuity conditions.

(Refer Slide Time: 24:02)

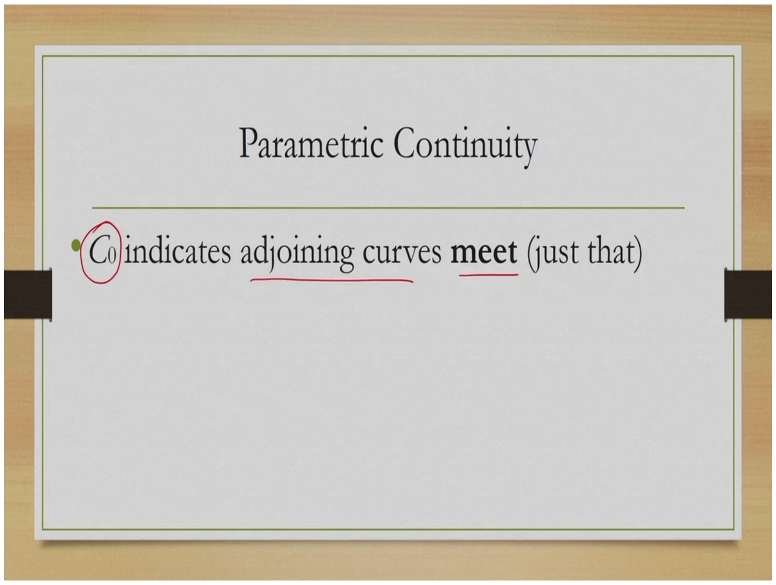


Parametric Continuity

- The *n*th order parametric continuity (*Cn*), states that **adjoining curves meet** and **first to *n*-th order parametric derivatives** of **the adjoining curve functions are equal at their common boundary**

So, in general, the nth order parametric continuity condition denoted by Cn states that adjoining curves meet and first to the nth order parametric derivatives of the adjoining curve functions are equal at their common boundary that is the general definition. Now, let us see what they refer to in simple terms.

(Refer Slide Time: 24:40)

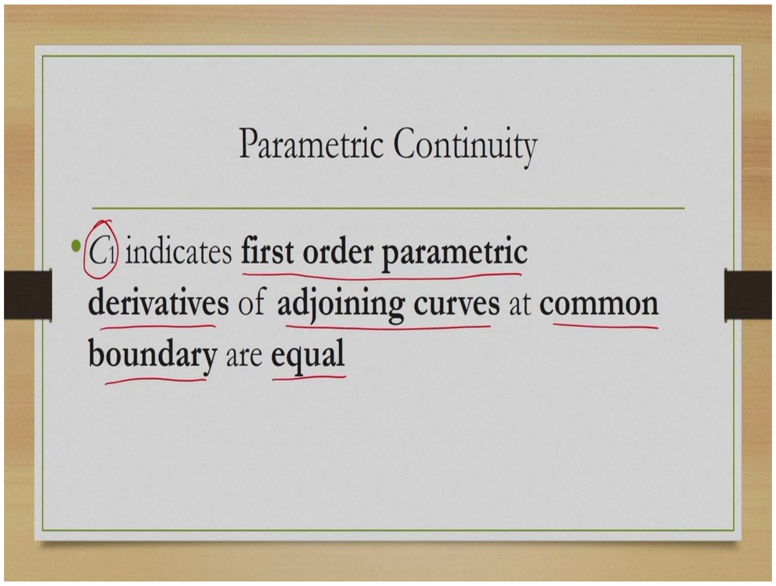


Parametric Continuity

- C_0 indicates adjoining curves **meet** (just that)

So, the first parametric continuity condition is C0, the zeroth order condition, which simply states that the adjoining curve meet. It is just that the simple condition.

(Refer Slide Time: 25:00)

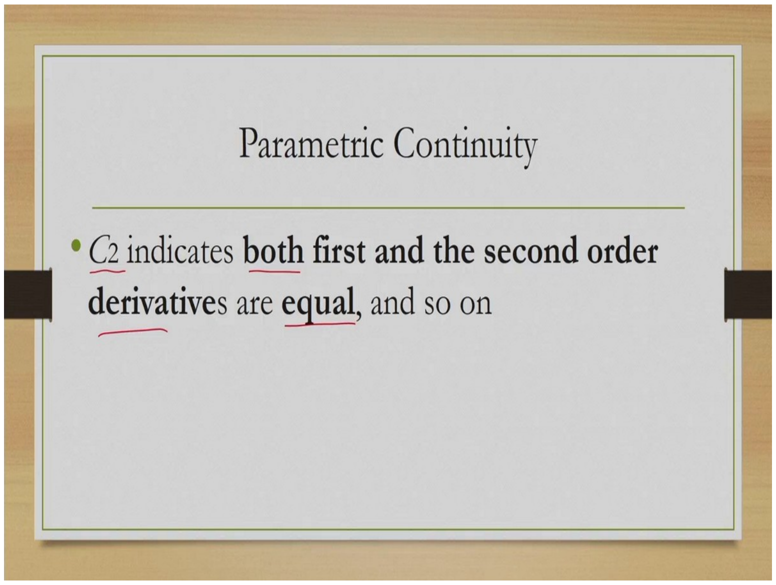


Parametric Continuity

- C_1 indicates **first order parametric derivatives** of **adjoining curves** at **common boundary** are **equal**

Now, the first order parametric condition C1 indicates that the first order derivatives of adjoining curves at common boundary are equal. So, essentially it tells that at the common boundary, we have to ensure that the first order parametric derivative. That means the derivative with respect to the parameter u of the curve should be equal.

(Refer Slide Time: 25:40)

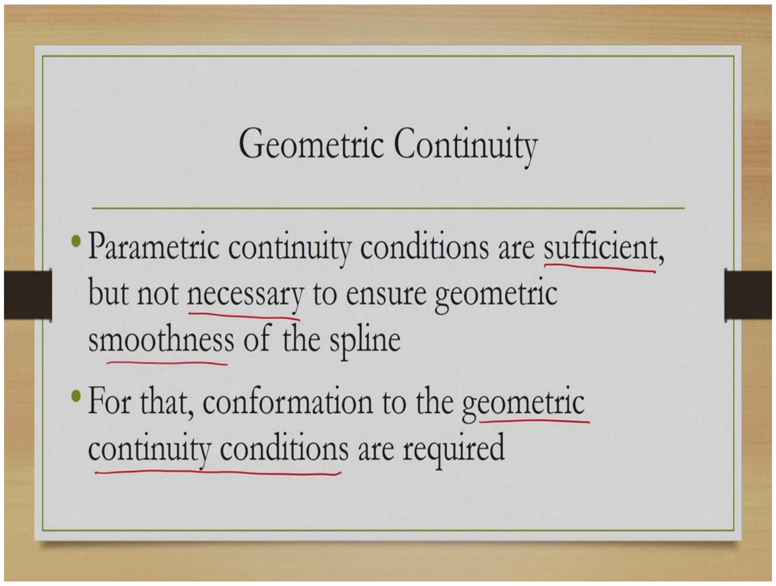


Parametric Continuity

- C2 indicates **both first and the second order derivatives** are **equal**, and so on

In a similar way C2 indicates that both the first and the second order derivatives are equal at the common boundary. And in this way, we can go on. But since in graphics we mostly focus on third degree polynomials so, we are mostly concerned with these continuity conditions up to C2.

(Refer Slide Time: 26:09)

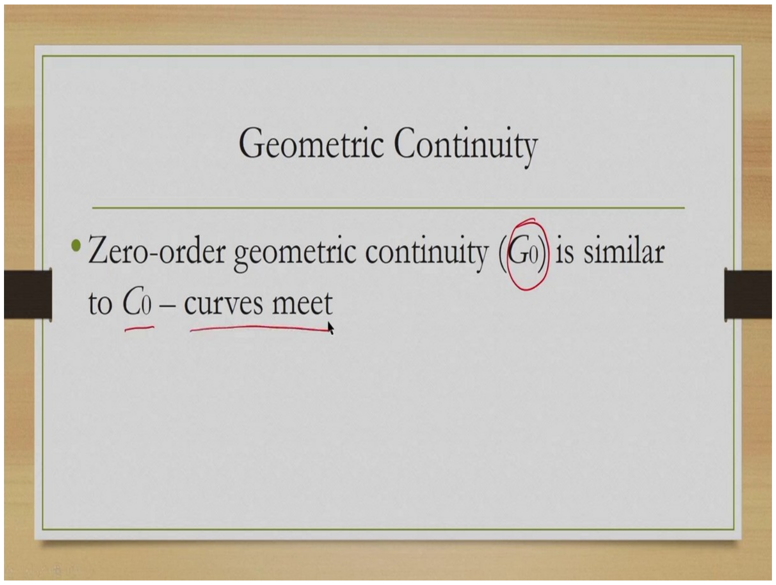


Geometric Continuity

- Parametric continuity conditions are sufficient, but not necessary to ensure geometric smoothness of the spline
- For that, conformation to the geometric continuity conditions are required

Now, this parametric continuity conditions are sufficient, but not necessary to ensure geometric smoothness of the spline. For that, what we need is to conform to the other set of continuity conditions called geometric continuity conditions. Now what are those?

(Refer Slide Time: 26:34)

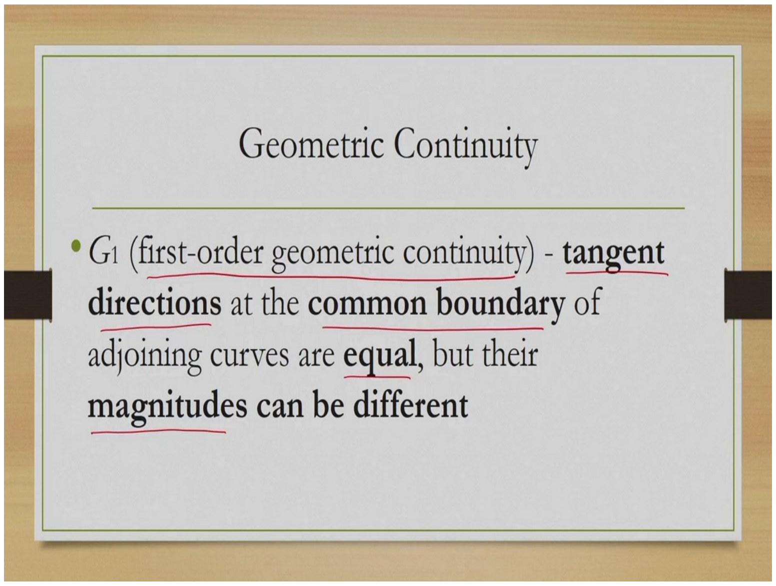


Geometric Continuity

- Zero-order geometric continuity (G_0) is similar to C_0 – curves meet

The 0 order condition is denoted by G0. This is the zeroth order condition which is similar to C0, which simply states that the curves must meet.

(Refer Slide Time: 26:50)

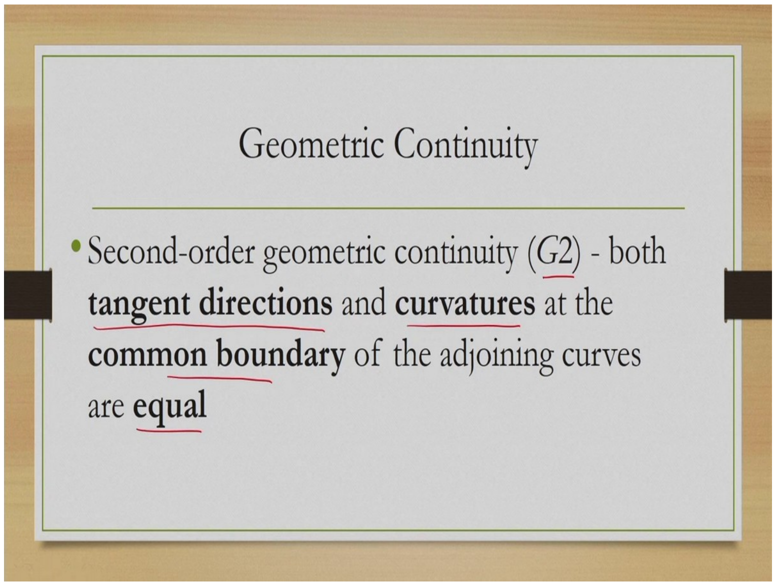


Geometric Continuity

- G_1 (first-order geometric continuity) - **tangent directions** at the **common boundary** of adjoining curves are **equal**, but their **magnitudes can be different**

Similarly, G1 or the first order geometric continuity condition tells that the tangent directions at the common boundary should be equal, although they are magnitudes can be different so that directions must be equal but magnitudes can vary at the boundary that is the G1 or first-order geometric continuity condition.

(Refer Slide Time: 27:23)

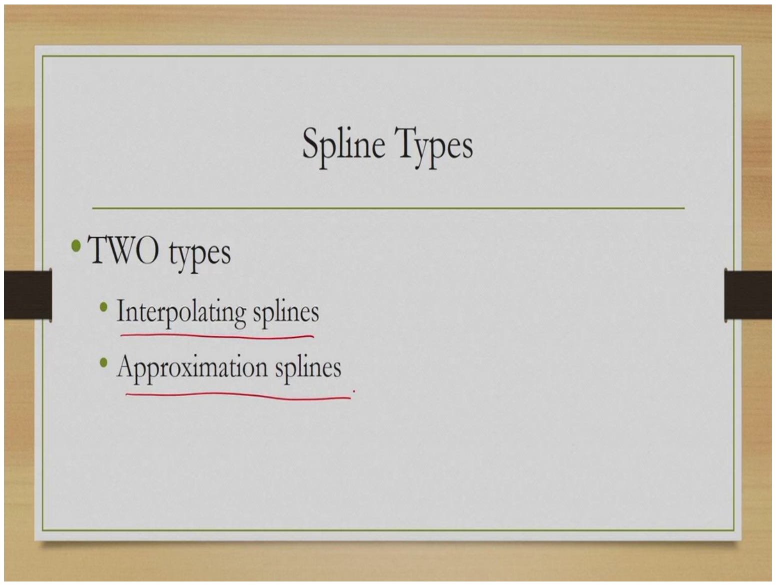


Geometric Continuity

- Second-order geometric continuity ($G2$) - both **tangent directions** and **curvatures** at the **common boundary** of the adjoining curves are **equal**

Second-order condition or G2 indicates that both tangent direction and curvatures at the common boundary of the adjoining curves should be equal. Again, we can go on like this up to any order, but since we are mostly concerned with cubic polynomials, up to G2 should be sufficient for our understanding. So, that is one basic knowledge that we should have about splines that is, if we want to represent any curve as splines, that means in terms of smaller polynomial pieces, we should ensure that the curves conforms to the continuity conditions, parametric and geometric continuity conditions.

(Refer Slide Time: 28:27)

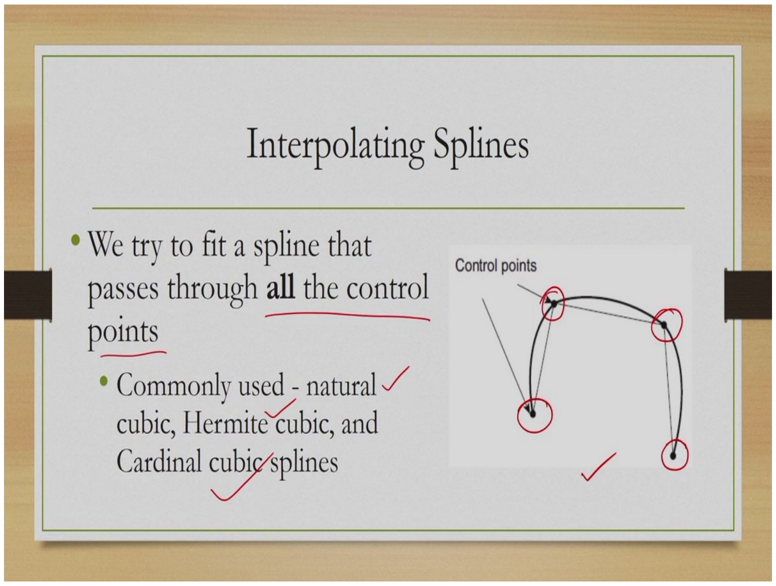


Spline Types

- TWO types
  - Interpolating splines
  - Approximation splines

Now let us try to see what are the different types of Spline representations that we can use. There are broadly two types. One is interpolating splines. Other one is approximation splines.

(Refer Slide Time: 28:46)



Interpolating Splines

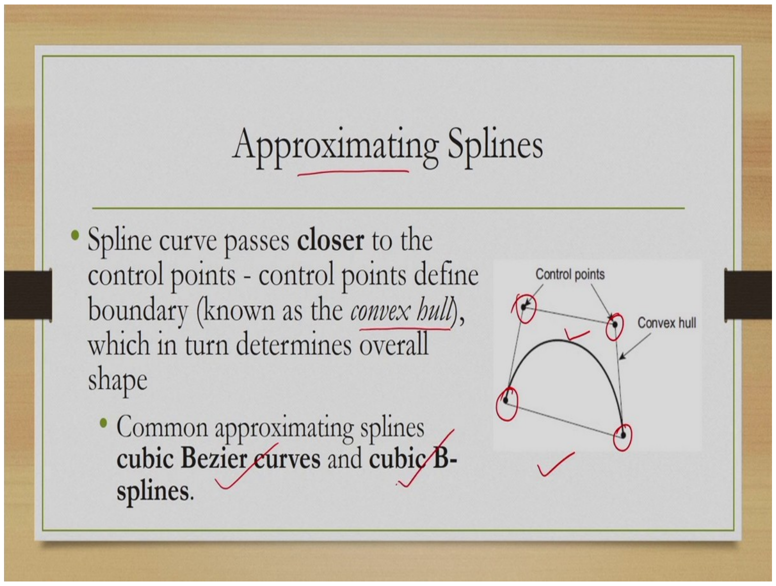
- We try to fit a spline that passes through **all** the control points
  - Commonly used - natural cubic, Hermite cubic, and Cardinal cubic splines

Control points

Now, in case of interpolating splines, what we want? We essentially try to fit the curve such that it passes through all the control points. So, essentially we are given a set of control points and we are representing the curve in the form of splines, in a way such that the polynomial pieces of the spline passes through all the control points as shown in this figure.

Now, the commonly used interpolating splines in computer graphics are natural cubic splines, hermite cubic splines, and Cardinal cubic splines. So, we will discuss about these splines in details later.

(Refer Slide Time: 29:46)



Approximating Splines

- Spline curve passes **closer** to the control points - control points define boundary (known as the *convex hull*), which in turn determines overall shape
  - Common approximating splines **cubic Bezier curves** and **cubic B-splines**.

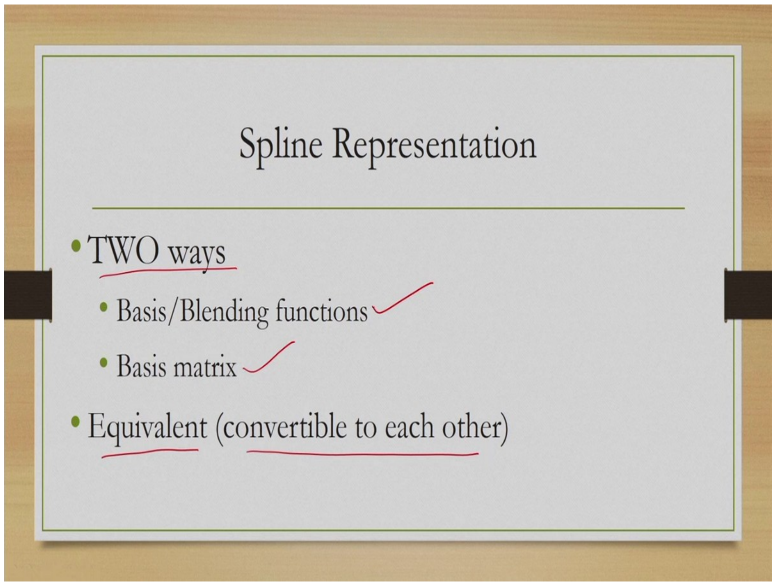
Control points
Convex hull

The other type of Spline curves are called approximating splines here. Control points are used to define a boundary or convex hull, the spline itself does not pass through all the control points. Instead, it is restricted within the boundary defined by the control points. Take the same example here we have 4 control points. But here, the curve is not passing through the 4 control points, unlike earlier in case of interpolating splines, what is happening here is that these control points are defining a bonding region, a boundary which is popularly called convex hull, and the spline lies within this boundary.

In other words, the Spline shape is determined by the convection. Now, there are a few common and popular splines approximating splines used in applications, namely the Cubic bezier curves and the Cubic B splines, again will discuss about those later. So, that is the basic idea of spline. What it is and what makes them good for representing any curve.

So, what it is it is essentially representing a complex shape in terms of smaller, manageable, lower degree polynomials or polynomial pieces. And it is able to represent the Curves smoothly because splines are supposed to conform to continuity conditions. Now, let us try to understand how we represent spline. This is same as knowing how to represent the objects which are represented by splines.

(Refer Slide Time: 32:08)

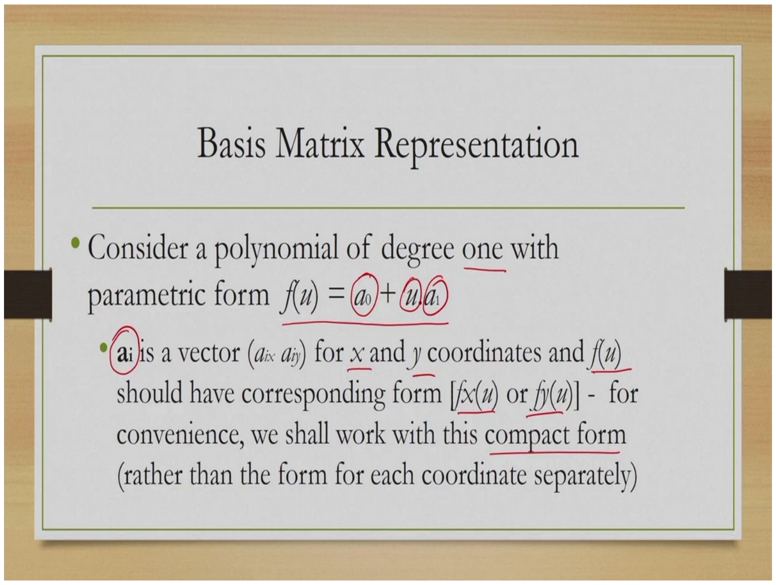


Spline Representation

- TWO ways
  - Basis/Blending functions
  - Basis matrix
- Equivalent (convertible to each other)

How can we represent splines? There are two ways, broadly one is basis or blending function based representation. Other one is basis metrics based representation. And these two are equivalent. Of course, that is quite obvious and one can be converted to the other and vice versa.

(Refer Slide Time: 32:34)



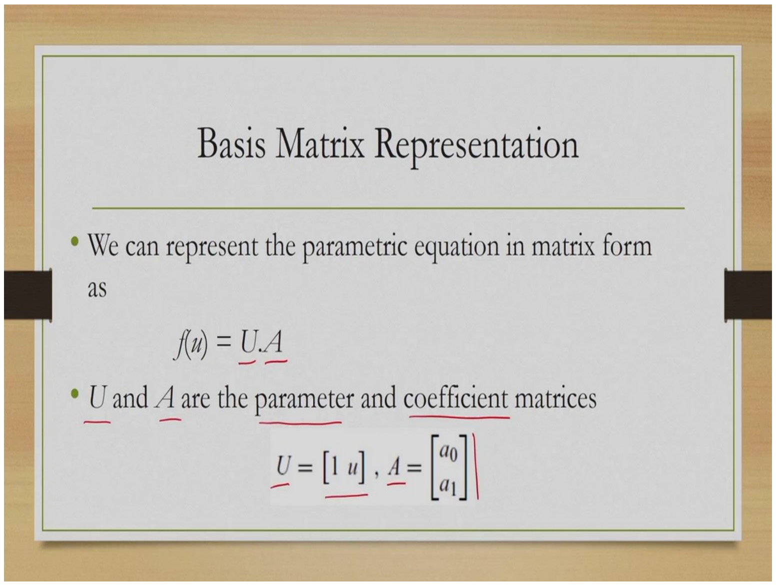
Basis Matrix Representation

- Consider a polynomial of degree one with parametric form $f(u) = a_0 + u a_1$
  - a_i is a vector (a_{ix} a_{iy}) for x and y coordinates and $f(u)$ should have corresponding form [$fx(u)$ or $fy(u)$] - for convenience, we shall work with this compact form (rather than the form for each coordinate separately)

Let us take some examples to understand the representation so will start with basic metrics, representation of splines. And we will start with a simple example. Consider a polynomial of degree one that is a linear polynomial, which in the parametric form we can represent as a f u equal to a0 plus ua1. Now a0, a1 are coefficients. And u is the parameter we must keep in mind here that this is a compact representation.

ai like a0, a1 actually represents vectors comprising of two components, one each for the corresponding coordinates. So, a0 actually has $a0_x$, $a0_y$ values separate for x and y coordinates. Similarly, fu should have corresponding expressions, namely fx_u and fy_u. However for simplicity we will work with this compact form rather than the expanded form.

(Refer Slide Time: 34:00)



Basis Matrix Representation

- We can represent the parametric equation in matrix form as

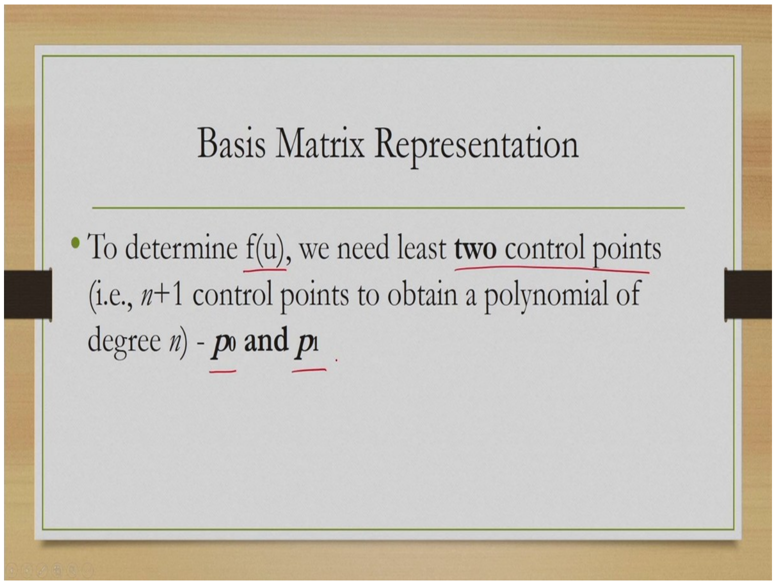
$$f(u) = \underline{U}.\underline{A}$$

- $\underline{U}$ and $\underline{A}$ are the parameter and coefficient matrices

$$\underline{U} = \underline{[1 \; u]}, \; \underline{A} = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$$

Now, this parametric equation we can represent in the form of matrix U.A. So, this is a dot product of two matrices, U and A. U is the parameter metrics and A is the coefficient metrics. Where U is denoted in the form of this vector 1, u and the metrics A is denoted in this column vector form. Having the two coefficients a0, a1 in our example.

(Refer Slide Time: 34:46)

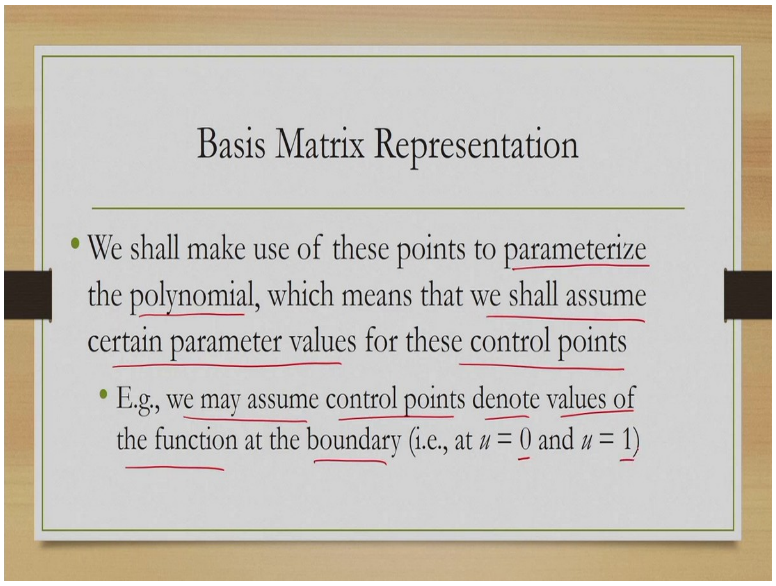


Basis Matrix Representation

- To determine f(u), we need least **two** control points (i.e., n+1 control points to obtain a polynomial of degree n) - p_0 **and** p_1

Now, since this is a polynomial of degree 1, so we need at least two control points to determine f. Let us denote those two by p0 and p1.

(Refer Slide Time: 35:06)

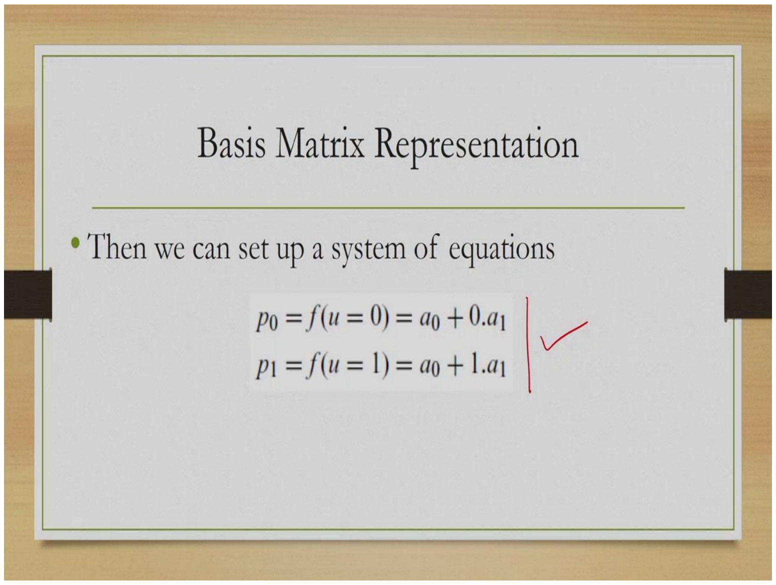


Basis Matrix Representation

- We shall make use of these points to parameterize the polynomial, which means that we shall assume certain parameter values for these control points
  - E.g., we may assume control points denote values of the function at the boundary (i.e., at u = 0 and u = 1)

Now, these points we will use to parameterize the polynomial, in other words, we shall assume that certain parameter values and therefore these control points, for example, we may assume the control points denote values of the function at the boundary where we can define the boundary as the points where the parameter values text of value 0 and 1.

(Refer Slide Time: 35:48)



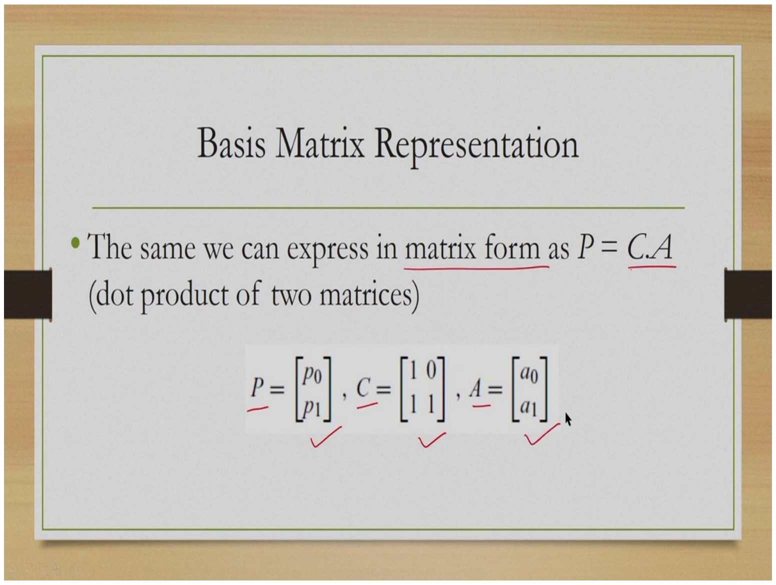
Basis Matrix Representation

- Then we can set up a system of equations

$$p_0 = f(u=0) = a_0 + 0.a_1$$
$$p_1 = f(u=1) = a_0 + 1.a_1$$

If that is the case, then we can set up our system of equations as shown here, two equation. One for p0, one for p1 with the parameter value fixed. Now, by solving these equations we can get the coefficients.

(Refer Slide Time: 36:17)



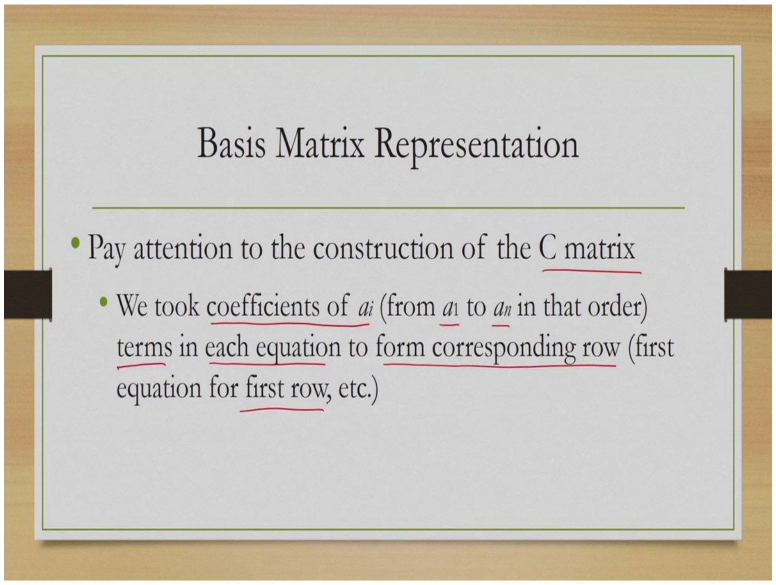
Basis Matrix Representation

- The same we can express in matrix form as $P = C.A$ (dot product of two matrices)

$$P = \begin{bmatrix} p_0 \\ p_1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, A = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$$

However, if we look closely then we can see that the same system of equation we can represent in the form of matrices. Now, what is this matrix representation we can represent it as being able to C.A. Where p is defined as a column vector C is defined as another column vector and A is defined as yet another column vector.

(Refer Slide Time: 36:56)

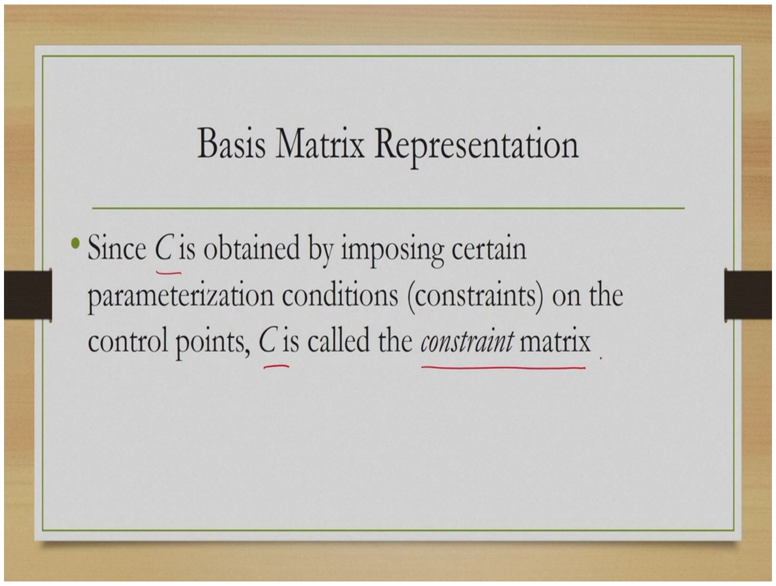


Basis Matrix Representation

- Pay attention to the construction of the C matrix
  - We took coefficients of a_i (from a_1 to a_n in that order) terms in each equation to form corresponding row (first equation for first row, etc.)

So, how we constructed the C matrix, we took the coefficients of a_i. That means from a_1 to a_n in that order. Those terms in each equation to from the corresponding row of the C matrix. So, first equation we took for the first row and so on.

(Refer Slide Time: 37:30)

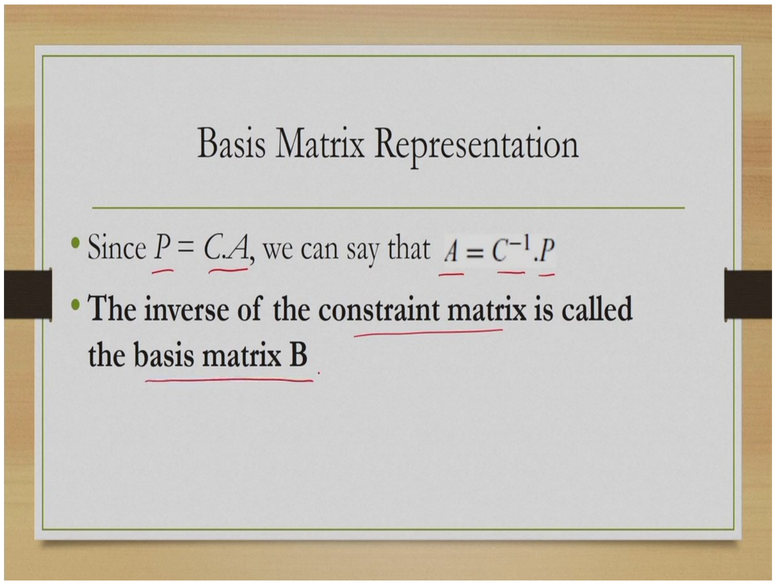


Basis Matrix Representation

- Since C is obtained by imposing certain parameterization conditions (constraints) on the control points, C is called the *constraint* matrix

In other words, we imposed certain constraints. Parameterization conditions as constraints to obtain C. Accordingly, C is called the constraint matrix.

(Refer Slide Time: 37:53)

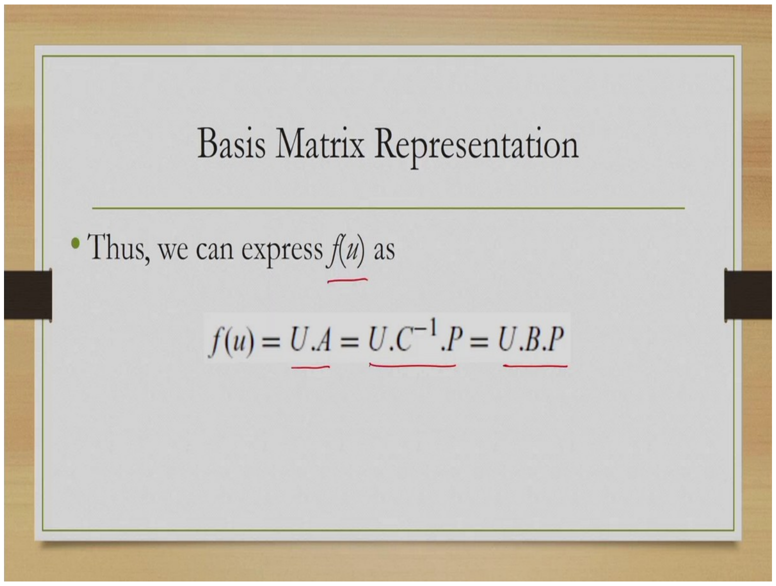


Basis Matrix Representation

- Since $P = C.A$, we can say that $A = C^{-1}.P$
- **The inverse of the constraint matrix is called the basis matrix B**

Now we know P equal to C.A so we can say that A equal to C^{-1}.P. Now, this inverse of the constant matrix is called basis matrix.

(Refer Slide Time: 38:13)



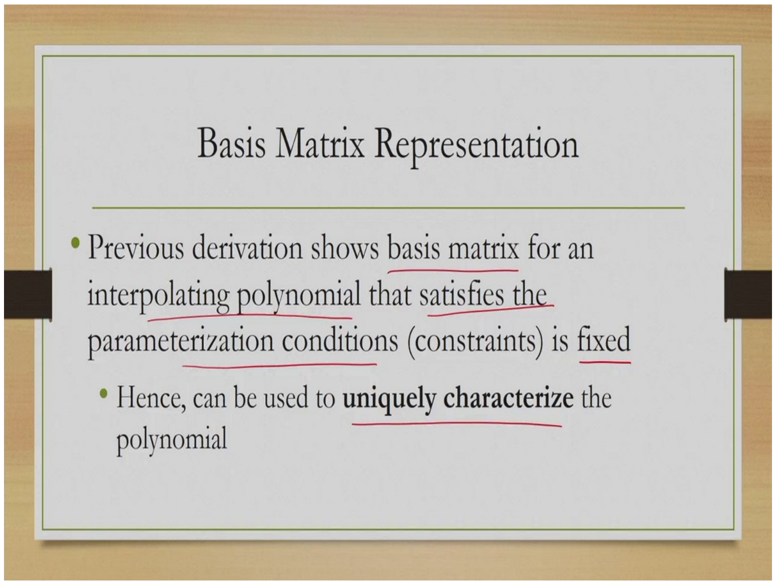
Basis Matrix Representation

- Thus, we can express $f(u)$ as

$$f(u) = \underline{U.A} = \underline{U.C^{-1}.P} = \underline{U.B.P}$$

So, we can represent f as U.A which can be expended, as U.C-1.P or UBP. So, this is the way to represent f in terms of matrix multiplication.

(Refer Slide Time: 38:37)

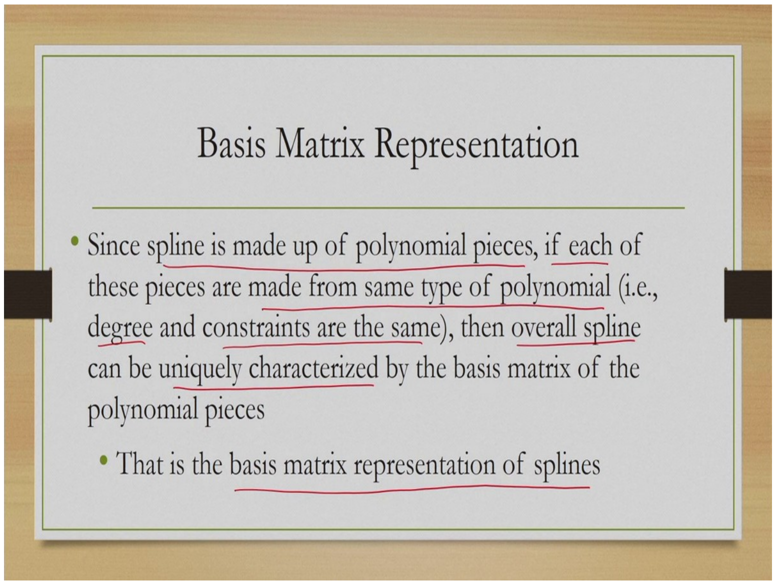


Basis Matrix Representation

- Previous derivation shows basis matrix for an interpolating polynomial that satisfies the parameterization conditions (constraints) is fixed
  - Hence, can be used to **uniquely characterize** the polynomial

Now, one thing we have to note here is that the basis matrix. For an interpolating polynomial that satisfies the parameterization conditions is fixed. In other words, the matrix or the basis matrix uniquely characterizes the polynomial. So, if we use the basis matrix B instead of the polynomial equation, then this is as good as representing the polynomial because B is fixed for the particular polynomial.

(Refer Slide Time: 39:19)



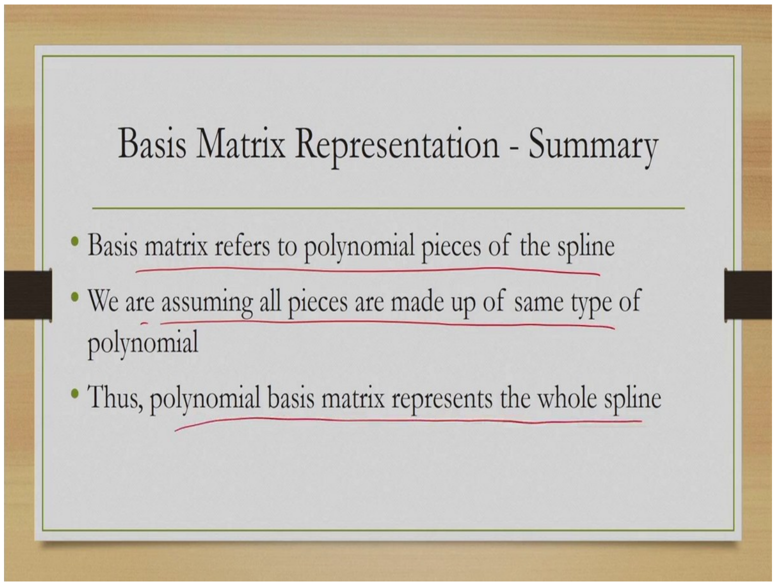
Basis Matrix Representation

- Since spline is made up of polynomial pieces, if each of these pieces are made from same type of polynomial (i.e., degree and constraints are the same), then overall spline can be uniquely characterized by the basis matrix of the polynomial pieces
  - That is the basis matrix representation of splines

Now, we know that spline is made up of polynomial pieces. Now, if each piece is made of the same type of polynomial, that means the degree and the constraints are the same. So, then overall, Spline can be uniquely characterized by each piece. And since already we have mentioned that a polynomial piece can be characterized by the basis matrix, then the basis matrix can also be used to uniquely characterize the entire spline. So, when we are representing the spline, we can simply represent it in terms of the basis matrix.

Now that is the basis matrix representation of spline. So, to recap given a polynomial, we can have a unique basis matrix for that polynomial under certain constraints. So, the basis matrix is suitable to represent the polynomial. Now the same polynomial pieces are used to represent a spline. So, for each polynomial piece, we have the same metrics so we can use a single basis matrix to represent the overall Spline, because the basis matrix will tell us that particular polynomial pieces are used to represent the spline. This is the basis matrix representation of splines.

(Refer Slide Time: 41:04)

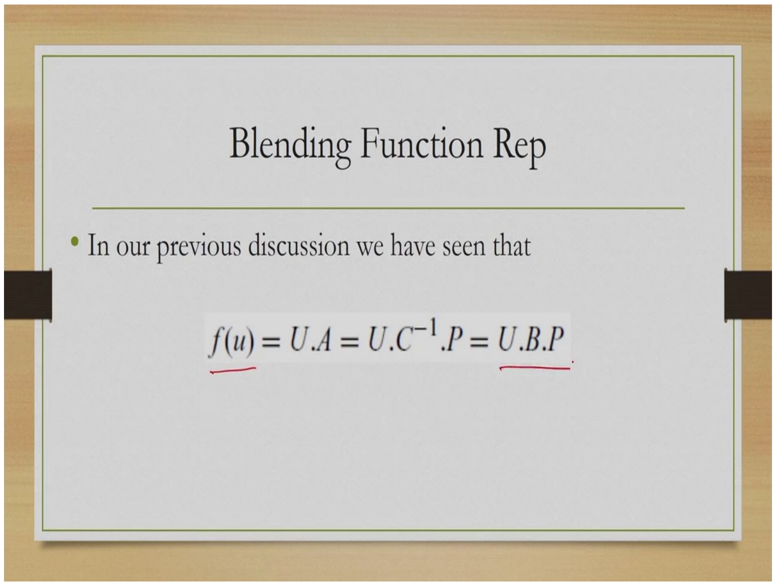


Basis Matrix Representation - Summary

- Basis matrix refers to polynomial pieces of the spline
- We are assuming all pieces are made up of same type of polynomial
- Thus, polynomial basis matrix represents the whole spline

This explanation we just mentioned, so basis matrix refers to polynomial pieces of this spline, we are assuming all pieces are made up of same polynomial. So, polynomial basis matrix represents the whole spline. Now, let us focus attention to the other type of spline representation, namely the blending function representation.

(Refer Slide Time: 41:33)



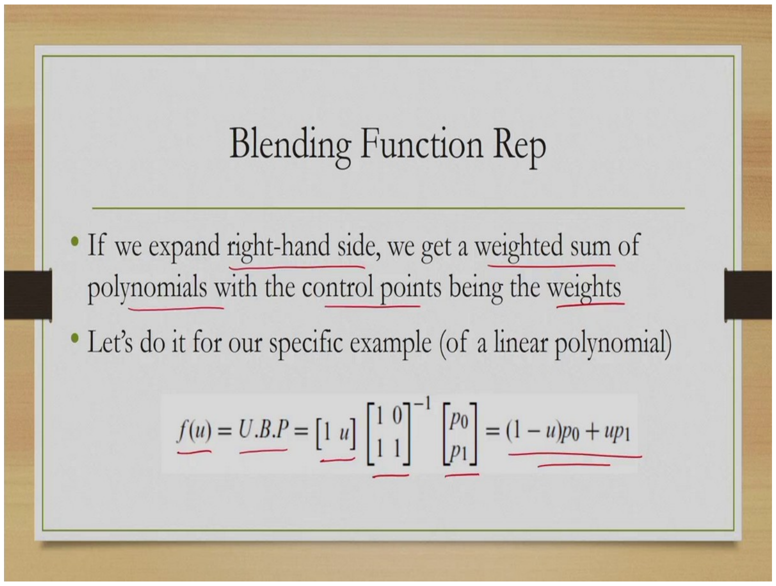
Blending Function Rep

- In our previous discussion we have seen that

$$\underline{f(u)} = U.A = U.C^{-1}.P = \underline{U.B.P}$$

Now, earlier, we have seen that we can representation f in terms of basis matrix, like U.B.P.

(Refer Slide Time: 41:43)



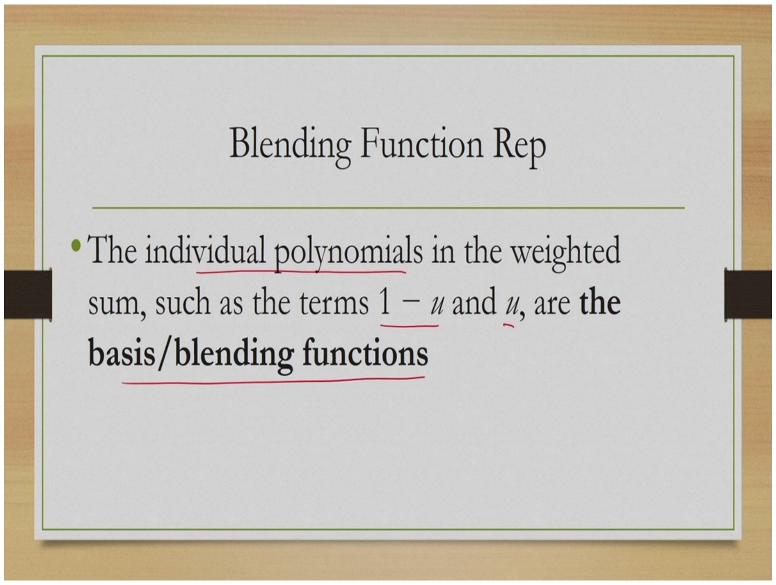
Blending Function Rep

- If we expand right-hand side, we get a weighted sum of polynomials with the control points being the weights
- Let's do it for our specific example (of a linear polynomial)

$$f(u) = U.B.P = \begin{bmatrix} 1 & u \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} p_0 \\ p_1 \end{bmatrix} = (1-u)p_0 + up_1$$

Now, if we expand right hand side, we get weighted sum of polynomials with the control points being the weights. So, in our example let us derive it and see what happens. So, in our example we have a polynomial of degree 1 and we have the matrices in this from u is this one. B, is this Matrix and C is the control point metrics. Now if we expand, we will get this equation in terms of the control points.

(Refer Slide Time: 42:37)

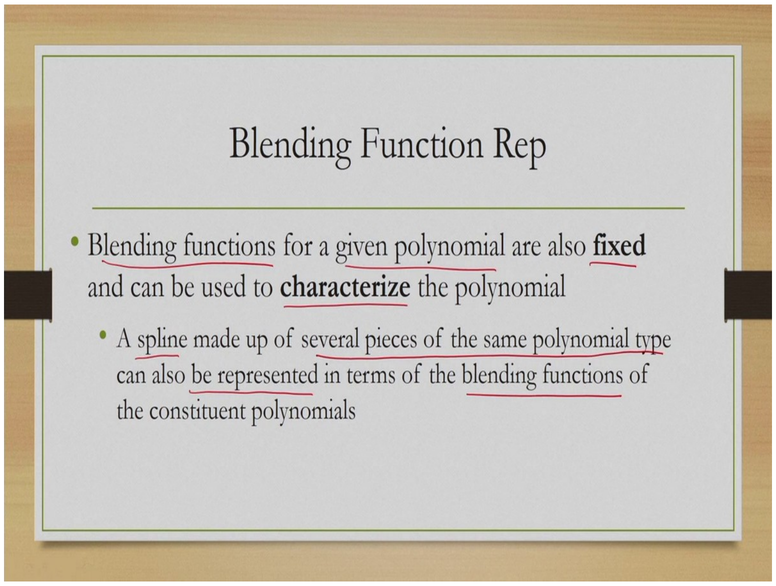


Blending Function Rep

- The individual polynomials in the weighted sum, such as the terms $1 - u$ and u, are **the basis/blending functions**

Now, the individual polynomials in the weighted sum, such as the term 1-u and u are the blending functions. So, the overall function is represented as a weighted sum of polynomials. And these individual polynomials are called the basis function or the blending functions.

(Refer Slide Time: 43:03)

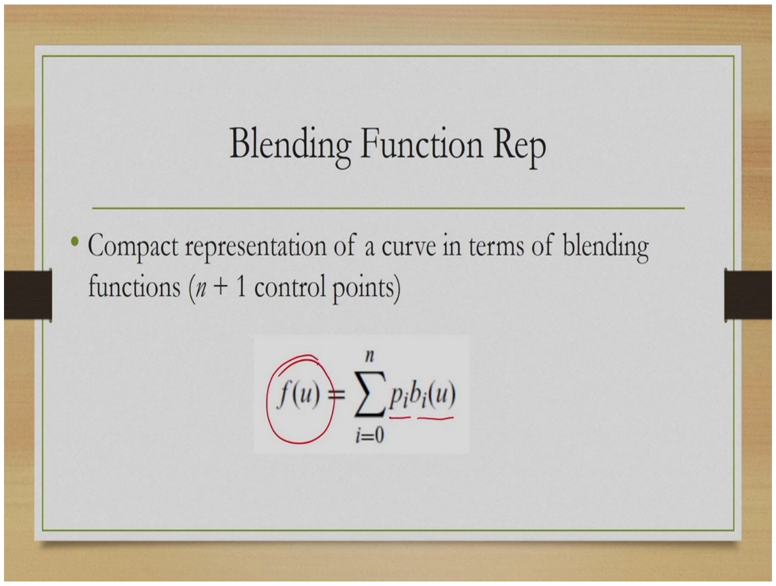


Blending Function Rep

- Blending functions for a given polynomial are also **fixed** and can be used to **characterize** the polynomial
  - A spline made up of several pieces of the same polynomial type can also be represented in terms of the blending functions of the constituent polynomials

Now, for a given polynomial, the blending functions are also fixed so we can use them to characterize the polynomial. So, for a given polynomial with constraints, the functions that can be used to represent it are fixed. So, this blending function set can be used to characterize the polynomial so we can apply the same logic here. Spline made up of several pieces of the same polynomial type. Therefore can also be represented in terms of the blending functions since they are uniquely characterizing the constituent polynomial pieces.

(Refer Slide Time: 43:52)



Blending Function Rep

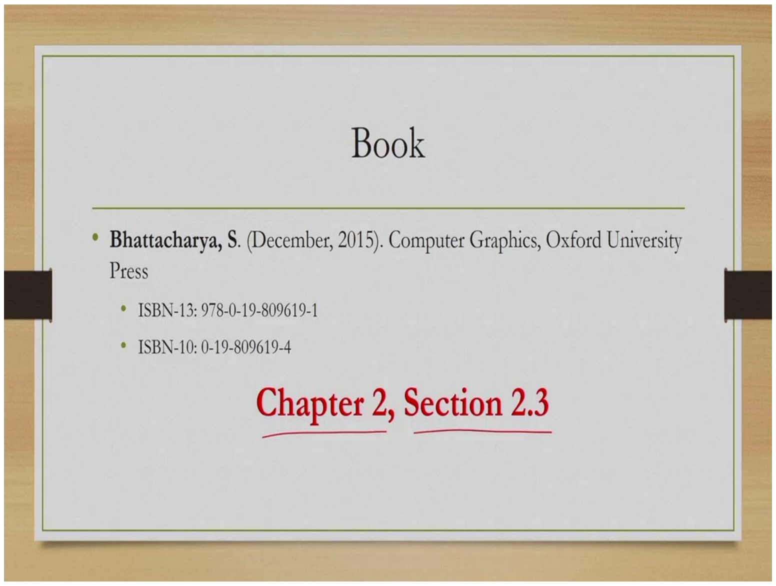
- Compact representation of a curve in terms of blending functions ($n + 1$ control points)

$$f(u) = \sum_{i=0}^{n} p_i b_i(u)$$

So, in a compact form we can represent a Spline or the curve f in this way where pi is the i-th control point and bi is the blending function. So, to recap today we have got introduced to the basic idea of splines, which is essentially representing a curve in terms of constituent lower degree polynomial pieces. Then we discussed the continuity conditions to ensure that splines give us smooth curves. We also discussed the broad types of splines and the way the splines can be represented in the form of basis matrices or blending functions.

In the next lecture will take up detailed discussion on the various types of splines that we have mentioned, namely the interpolating splines and the approximating splines. We will also learn in the next lecture about the use of splines to represent surfaces in computer graphics.

(Refer Slide Time: 45:15)



Book

- **Bhattacharya, S**. (December, 2015). Computer Graphics, Oxford University Press
  - ISBN-13: 978-0-19-809619-1
  - ISBN-10: 0-19-809619-4

Chapter 2, Section 2.3

Whatever I have discussed today can be found in this book. You are advised to go through Chapter 2, Section 2.3 to learn about these topics in more detail. See you in the next lecture till then thank you and goodbye.