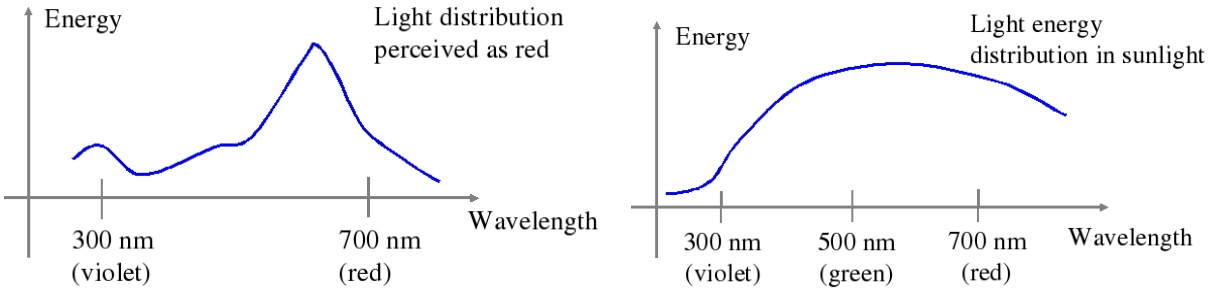
Lecture 7: Colour

The physical description of colour

Colour vision is a very complicated biological and psychological phenomenon. It can be described in many different ways, including by physics, by subjective observation, or by the tri-stimulus representation commonly used in computers. As we noted previously, light has a wavelength (λ), which can be measured accurately, and an energy or intensity. Lasers produce light of one particular wavelength, but generally visible light is a distribution of energy into a band of wavelengths. Figure 1(a) shows a distribution of light energy which we would perceive as red in colour, and figure 1(b) shows the the distribution of energy found in sunlight, which we perceive as light yellow. The wavelength range of the visible spectrum is approximately from 400 to 700 nano-meters.



(a) Red light

(b) Sunlight

Figure 1: Energy Distributions for light

In the eye we have just three distinct 'cone' cells for detecting light energy. These respond to a band of wavelength centred around red (600), green (560) and blue (440); respectively. The bands overlap, so, for example green light excites all three types. Typical responses of the cone cells are shown in figure 2 The consequence of this is that each type of cell may be excited similarly from very different energy distributions. Any distribution of wavelengths will be perceived by us as a single colour, but two entirely different distributions of intensities could be perceived as the same colour.

Subjective observation of colour

Two colours are said to be equivalent when an observer says that they are equivalent. Selecting three pure light sources (R, G, B) and mixing them together while varying their respective intensities, one may be able to create a large number of colours. Each colour satisfies the following linear combination: $X = r + g + b$ where r, g, b are the intensities of the red, green and blue light sources in the mix. One way to approach colour definition is through a matching experiment. For an unknown colour we can compare it to a mixture X, and adjust r, g and b until the mixture matches the unknown colour. As we shall see later, not all colours can be matched in this way. However, by adding one of the pure colours to an unknown, unmatchable colour, we can make a match. This is in effect subtracting a colour from the mix.

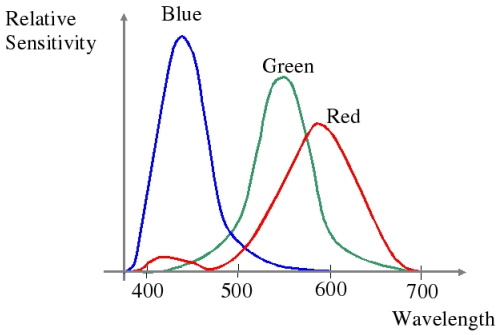


Figure 2: Typical responses of individual cone cells

$$X + r = g + b$$
or $$X + g = r + b$$
or $$X + b = r + g$$

The representation of colours as a mixture of three components is called the tri-stimulus representation, and is

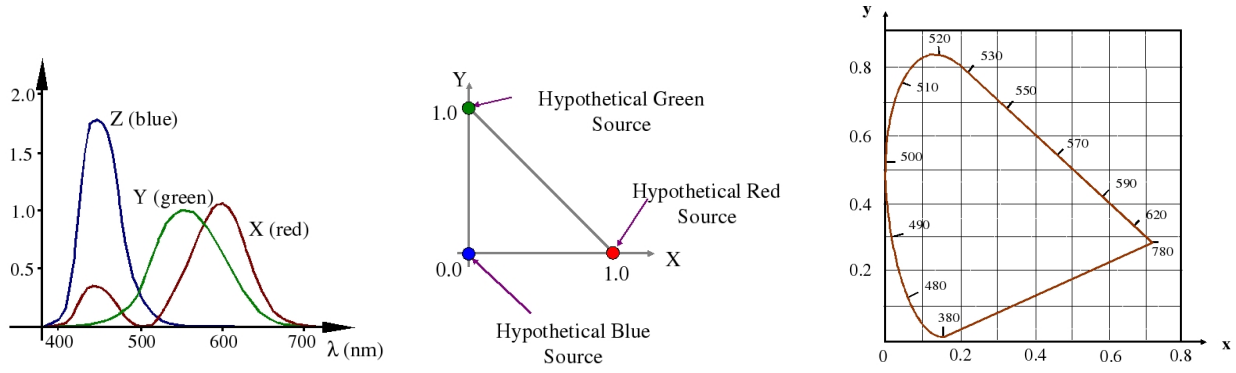
very commonly used in monitors and other active colour devices. The pure colours used are red green and blue, and are referred to as the additive primary colours.

The Standard Additive Linear System (CIE)

Matching allows us to determine an rgb representation of any colour, even though the scales include negative numbers. The scales used depend on the wavelengths of the sources for matching and are not standard. In order to create a manageable standard system, a committee of scientists devised the CIE Chromaticity Diagram, in which colours are specified by a linear combination of three light sources normalised to the positive range [0..1] to avoid any negative values. The three principle colours would normally require a three dimensional diagram, but by using further normalisation a two dimensional diagram is produced. Suppose a colour is defined, as above, by: $C = r + g + b$. If we divide each component of this equation with $(r + g + b)$ and get a normalised set of coordinates:

$x = r/(r + g + b)$ the normalised red component,
$y = g/(r + g + b)$ the normalised green component,
$z = b/(r + g + b)$ the normalised blue component,
and $x + y + z = 1$

So, knowing two of the three normalised colour components (x, y) the third one can be determined, for example $z = 1 - x - y$.



(a) Standard Oberver response

(b) Normalised space for colour definition

(c) The CIE Diagram

Figure 3: Defining the CIE Diagram

The process by which the CIE diagram was defined is illustrated in in figure 3. First a standard observer response was defined. This takes into account not only the typical responses of the three types of cone cell, shown in figure 2, but also the density of the cells in the foveal region of the eye. This standard response is shown in figure 3(a) and represents the typical overall cone cell responses to the light arriving in a small solid angel at the eye. Any wavelength in figure 3(a) can now be associated with a tri-stimulus value defined by the relative magnitudes of the responses of the three types of cell. For example at the wavelength 580 we can read off the graph that blue is zero and red and greeen are approximately equal at 0.8. The three values are normlised so that they sum to 1 and plotted in a normalised cartesian space which is illustrated by figure 3(b). The resulting wavelengths plotted in this space are shown on the CIE diagram of figure 3(c). All visible colours can be defined in this way, though only their normalised values are shown on colour prints of the diagram.

The pure visible colours are distributed on the edge of the horseshoe shaped curve with their physical wavelength running from 380 to 780 nm. At top and bottom ends of the visible spectrum, where the cell responses fall off, the wavelengths converge very fast towards a vanishing point. For example the wavelength 700 nm is virtually coincident with the 780 nm point in figure 3(c). We can see that just below 500 nm the red component is zero and the blue and green are approximately equal which would correspond to the colour cyan. The straight line from 780 to 540 on the right hand side represents the condition that $x + y = 1$ which means that the blue component is zero. The straight line at the bottom, joining 380 to 780 is a combination of red and blue, which has no equivalent single wavelength colour. It is called the magenta line. All visible colours are

contained within the horseshoe, which must be convex, since all colours are a mixture of pure wavelengths. The usefulness of this diagram is that it expresses a linear additive system and the addition of colour components can be done graphically.

Graphical Manipulation of Colour Parameters

Some definitions are shown on the CIE diagram of figure 4. The white point, $x = y = z = 1/3$, has equal strength of all three basic components. Pure colours, around the edge of the horseshoe, are called pure or fully saturated colours. If we take a pure colour, say 488 nm on the diagram, the line connecting this point and the white point contains all the possible colours which can be achieved by adding white to this pure colour. If we extend this line and on the other side it hits the curve at another pure colour point (598 nm in this case), which is called the complement because mixing the two makes white.

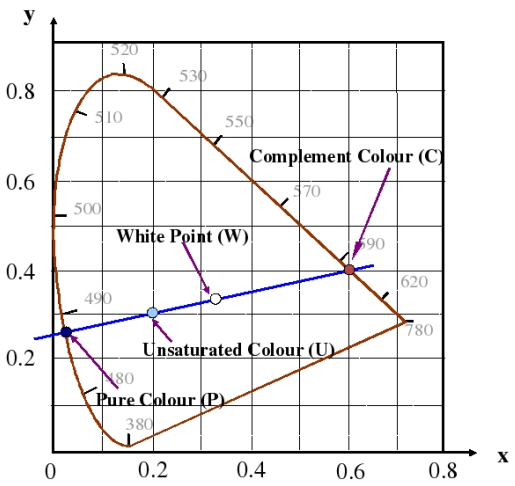


Figure 4: Calculations on the CIE Diagram

Saturation, or the degree of purity, of a given colour point can also be determined graphically. The white point is completely unsaturated (has no colour). The ratio of distances between the colour and the white point and the pure colour and the white point gives us saturation:

$$saturation = \frac{|W - U|}{|W - P|}$$

The saturation is always taken as 1 on the horseshoe (fully saturated) and zero at the white point. If we take any colour on the diagram, for example [0.2, 0.3, 0.5] shown in figure 4, we can measure this ratio, which in this case is approximately equal to 0.4. This tells us the white content of the point. We can also determine a pure colour by considering the line through the point and the white point. From the slope of the line we can calculate that it cuts the y axis at [0, 0.25, 0.75]. This point is not in the visible region of the CIE diagram. The nearest pure colour will be seen to be around [0.04, .26, .736]. The complement of this colour can be found by determining the colour which when added to it will produce white. This will be on the z=0 line, so the blue component is zero, and so we have that [0.04, .26, 0.736] + [r,g,0] = [k,k,k] from the blue component we can find that k = .736, whence 0.04 + r = 0.736, so r = 0.696 and g = 0.476. If we normalise these points we get [0.6, 0.4, 0.0] as shown on the diagram. Notice that the normalisation on the diagram gives a weighting to the colour coordinates. Since on the diagram the line joining a colour and its complement always goes through the white point, it is always possible to find a linear combination of the two that produces white. It does not mean that adding the co-ordinates and normalising produces white.

Subtractive Colour representation

Paints and printing ink absorb colour reflecting only part of the incident light. For this reason they are called subtractive primaries The three principal colours of paints are magenta (purple), cyan (pale blue) and yellow. These three can be created by combining the additive colour primaries. Magenta is red and blue, cyan is blue and green, and yellow is red and green. Black is created by a blend of the three subtractive primaries, but is usually also used in colour printing for convenience. It reflects none of the incident light. White paint (or paper) reflects all the incident light and so does not change its colour. Black and white paints can be blended to produce different shades of grey. With grey and red paint you may make brown, and a milliard other colours which have no equivalents in the preceding systems. Paints are much richer than colours produced by red/green/blue lights.

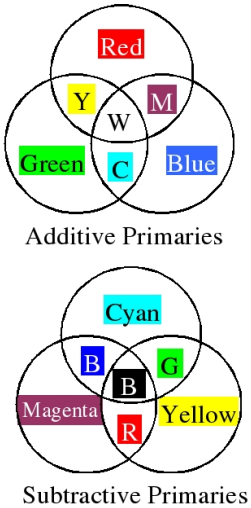


Figure 5: The Primary Colours

Figure 5 indicates how the different, pure basic colours are generated by the additive (monitor) and subtractive (printer) colour systems. Each circle represents one basic additive (R,G,B), or subtractive (M,C,Y) colour. The other regions indicate pure

colours generated by two basic colour addition or subtraction, respectively. The centre colour (three addition or subtraction) in the additive system is white, while it is black in the subtractive system (all colours are absorbed). The two systems show additive or subtractive complementary colours in the same region; i.e., red + cyan light makes white while red + cyan paint makes black.

Practical Colours

Approximately 128 different hues may be distinguished by the human eye. For each hue around 20 to 30 different saturation values may be seen as different colours. The eye is capable of distinguishing between 60 and 100 different brightness levels. If we multiply these three numbers, we get approximately 350,000 different colours. 24 bits provides us over 1.6 million colour shades, however, the eye is very sensitive to differences and under some circumstances small differences can be detected by it.

When a transformation is necessary from the CIE standard colour chart to the colour produced on a specific display device, we must express the primary colours of the display device by x and y quantities. Good quality monitors will be calibrated for the CIE colour chart. For example, a colour CRT monitor may have the following primary colour sources:

	x	y	z
Red	0.628	0.346	0.026
Green	0.268	0.588	0.144
Blue	0.150	0.07	0.780

These three points define a triangle on the CIE chromaticity diagram, as shown in figure 6. Only the points inside this triangle may be reproduced by the display device.

Colour Spaces and Transformation of Colours

There are many different ways of representing colour, and often they can be expressed as linear combinations of each other. For example, the transformation from computer colour intensities R,G,B (expressed in the range from 0.0 to 1.0) may be done by the matrix:

$$[x, y, z] = \begin{bmatrix} 0.628 & 0.268 & 0.150 \\ 0.346 & 0.588 & 0.07 \\ 0.026 & 0.144 & 0.780 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (1)$$

The pure red on the monitor has representation of [1.0, 0, 0] which then produces the correct CIE values of [0.628, 0.346, 0.026] for the monitor. In order to find out how to represent a known CIE colour on the display, we have to invert the matrix and then right multiply it with the vector [x,y,z].

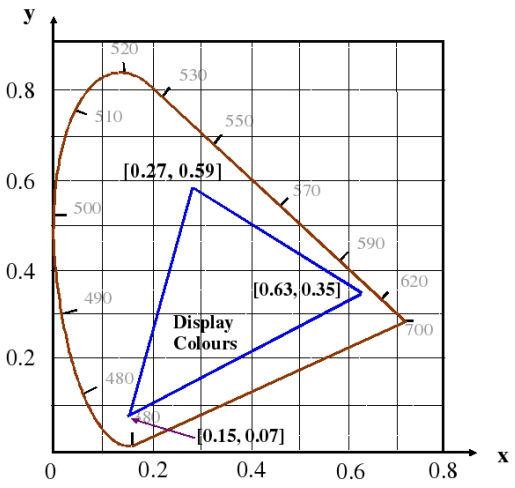


Figure 6: Display Colours

One useful colour space is called the HSV (Hue, Saturation and Value) space. It is of interest since it is thought to correspond more closely to our perception of colour than the RGB space we have been discussing. Like the RGB system, a colour is represented by three values which can be calculated from the r g b values as follows:

$V = max(r, g, b)$

$S = (max(r, g, b) - min(r, g, b))/max(r, g, b)$

Hue (which is an angle between 0 and 360^o) is best described procedurally:

if $(r = g = b)$	Hue is undefined, the colour is black, white or grey.
if $(r > b)$ and $(g > b)$	Hue $= 120 * (g - b)/((r - b) + (g - b))$
if $(g > r)$ and $(b > r)$	Hue $= 120 + 120 * (b - r)/((g - r) + (b - r))$
if $(r > g)$ and $(b > g)$	Hue$= 240 + 120 * (r - g)/((r - g) + (b - g))$

There are other, very similar, perceptual colour space called HSL (lightness) and HSI (intensity) which have slightly different definitions fo saturation and intensity. Note that, every colour in the RGB system is made

up of a white component whose magnitude is the minimum out of r g and b, and a pure colour defined by the ratio of the two non-zero colours when the white component has been subtracted (Figure 7). Note that the pure colour is not a coherent wavelength as in the CIE diagram. The saturation is the proportion of pure colour. As in the RGB system colours may be adjusted by changing any of the three values. However, with the HSI system the three variables each correspond to properties of colour we can easily understand, and thus they are easier to use to adjust colour palettes and balance colour images.

Alpha Channels

Colour representations in computer systems sometimes use four components to represent each colour. The first three are R G and B and the fourth or alpha value is simply an attenuation of the intensity. This does not add anything further to the representation, but allows greater flexibility in representing colours and can avoid truncation errors at low intensities. It can also be used for providing special features such as masking certain parts of an image.

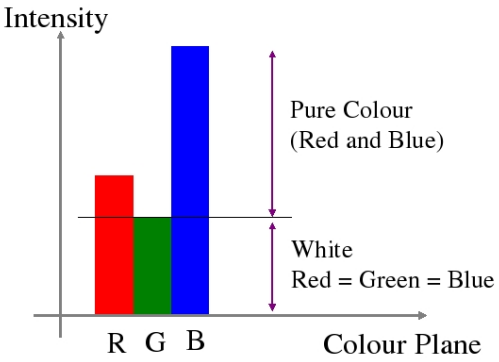


Figure 7: A tri-Stimulus Colour

Lecture 8: Introduction to Spline Curves

Splines are used in graphics to represent smooth curves and surfaces. They use a small set of control points (knots) and a function that generates a curve through those points. This allows the creation of complex smooth shapes without the need for mainpulating many short line segments or polygons at the cost of a little extra computation time when the objects of a scene are being designed. We will start with a simple, but not very useful spline. Taking the equation $y = f(x)$, we can express f as a polynomial function, say:

$$y = a_2x^2 + a_1x + a_0$$

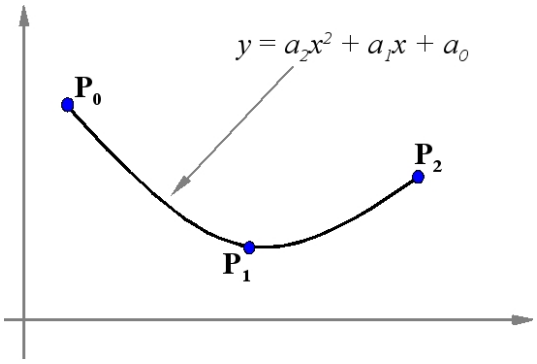


Figure 1: A non-parametric spline

If we now take any three points $[x_0, y_0]$, $[x_1, y_1]$ and $[x_2, y_2]$, we can substitute then into the equation to get three simultaneous equations which we can solve for the unknowns a_2, a_1 and a_0. We now have the equation of a curve interpolating the three points. It is of course a parabola, or parabolic spline. Notice that we don't have any control over the curve. There is only one parabola that will fit the data as shown in figure 1.

Parametric Splines

We can improve our choice by the simple expedient of using a parametric spline. Let us consider first a quadratic polynomial spline written in vector notation as:

$$\mathbf{P} = \mathbf{a_2}\mu^2 + \mathbf{a_1}\mu + \mathbf{a_0} \tag{1}$$

where $\mathbf{a_0}, \mathbf{a_1}$ and $\mathbf{a_2}$ are constant vectors whose values determine the shape of the spline. For two dimensional curves we now therefore have six unknowns (rather than the three previously). We can use these extra degrees of freedom to control the shape of the curve.

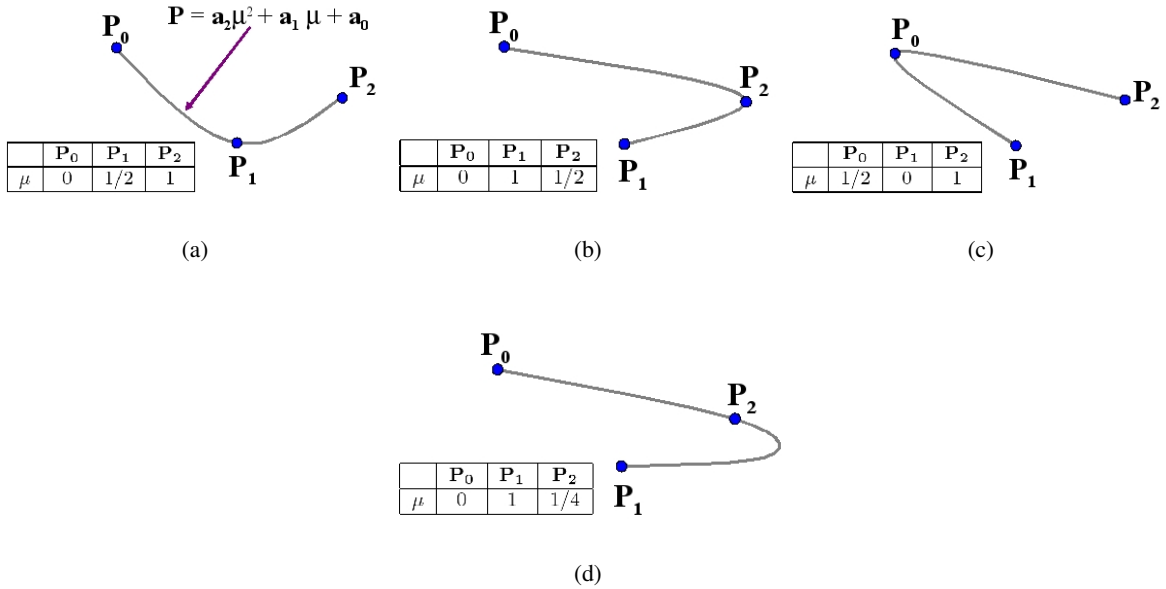


Figure 2: Possibilities using parametric splines

We will use the convention that $0 \leq \mu \leq 1$ over the range of interest. Hence at $\mu = 0$ the curve is at the first point to be interpolated, and at $\mu = 1$ it is at the last. Now consider interpolating the three points as before ($\mathbf{P_0} = [x_0, y_0], \mathbf{P_1} = [x_1, y_1]$ and $\mathbf{P_2} = [x_2, y_2]$). When $\mu = 0$ the curve passes through the first point, say

$\mathbf{P_0}$, and so, substituting $\mu = 0$ into equation 1 we can write $\mathbf{P_0} = \mathbf{a_0}$. Similarly, when $\mu = 1$ the point passes through the last point, say $\mathbf{P_2}$, and this gives us the equation:

$$\mathbf{P_2} = \mathbf{a_2} + \mathbf{a_1} + \mathbf{a_0},$$

substituting for $\mathbf{a_0}$ we get

$$\mathbf{P_2} - \mathbf{P_0} = \mathbf{a_2} + \mathbf{a_1}. \tag{2}$$

We have now met all our conditions except that the curve shall pass through $\mathbf{P_1}$. We can choose μ anywhere in the range $0 < \mu < 1$, and get a third equation to solve for the curve parameters. In other words we can now pick one of a family of curves interpolating the three points, by selecting the value of μ at $\mathbf{P_1}$. Choosing $\mu = 1/4$ we get

$$\mathbf{P_2} - \mathbf{P_0} = \mathbf{a_2}/16 + \mathbf{a_1}/4. \tag{3}$$

We can now solve equations 2 and 3 for the values of $\mathbf{a_1}$ and $\mathbf{a_2}$ and draw the curve as shown in Figure 2(d). Further possible curves using the same three points and parametric equation are shown in Figures 2(a), 2(b) and 2(c).

SplinePatches

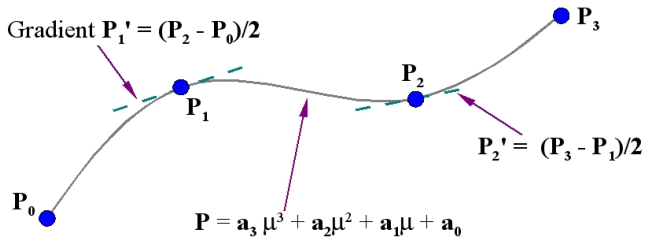


Figure 3: A simple way to join patches

Although we have gained more freedom by using the parametric form, we do not have any intuitive way of using it. That is to say, we have no simple way to choose μ values for each point to get the type of interpolating spline we want. Moreover, we still face the problem of having to use ever higher degrees of polynomials for higher numbers of points. To overcome these difficulties we introduce a method based on spline patches that allows simple intuitive spline construction. We define a different curve between each pair of adjacent knots, as shown in Figure 3. This is most commonly done by using a cubic spline for each patch, rather than the quadratic splines formulated above. The reason for choosing a cubic form is so that we can join the patches smoothly together. The equation for a parmetric cubic spline patch has four unknowns, $\mathbf{a_0}, \mathbf{a_1}, \mathbf{a_2}$ and $\mathbf{a_3}$:

$$\mathbf{P} = \mathbf{a_3}\mu^3 + \mathbf{a_2}\mu^2 + \mathbf{a_1}\mu + \mathbf{a_0} \tag{4}$$

We use the extra degrees of freedom provided by the cubic equation to set the gradient at either end of the patch, and thus make it join seamlessly to its neighbours. Looking at Figure 3, we see that this can conveniently be done by taking the difference of the coordinates on either side of the knot in question. This however is not the only way of setting this gradient, as we will see later. If we differentiate the curve equation we get:

$$\mathbf{P'} = 3\mathbf{a_3}\mu^2 + 2\mathbf{a_2}\mu + \mathbf{a_1} \tag{5}$$

Now consider the two ends of a spline patch between $\mathbf{P_i}$ and $\mathbf{P_{i+1}}$, where $\mu = 0$ or $\mu = 1$. We can find the positions and gradients at each end by substituting $\mu = 0$ and $\mu = 1$ into equations 4 and 5 which gives:

$\mathbf{P_i} = \mathbf{a_0}$
$\mathbf{P'_i} = \mathbf{a_1}$
$\mathbf{P_{i+1}} = \mathbf{a_3} + \mathbf{a_2} + \mathbf{a_1} + \mathbf{a_0}$
$\mathbf{P'_{i+1}} = \mathbf{3a_3} + \mathbf{2a_2} + \mathbf{a_1}$

We can write this system of equations in matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \\ \mathbf{a_2} \\ \mathbf{a_3} \end{bmatrix} = \begin{bmatrix} \mathbf{P_i} \\ \mathbf{P'_i} \\ \mathbf{P_{i+1}} \\ \mathbf{P'_{i+1}} \end{bmatrix} \tag{6}$$

and since the $\mathbf{a}$ values are unknown and the $\mathbf{P}$ and $\mathbf{P}'$ known, we need to invert the matrix to solve for the parameters of the spline patch.

$$\begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \\ \mathbf{a_2} \\ \mathbf{a_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & -3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P_i} \\ \mathbf{P'_i} \\ \mathbf{P_{i+1}} \\ \mathbf{P'_{i+1}} \end{bmatrix} \tag{7}$$

Notice that we have vector quantities, so, this formulation represents eight equations for the 2D case, and twelve for the 3D case. The matrix is the same for each dimension. For any given set of knots, this cubic patch method gives a stable, practical solution.

Bezier Curves

One of the simplest ways of approximating a curve was made popular by the French mathematician Pierre Bezier in the context of car body design. It was based on a mathematical formulation by another French mathematician Paul de Casteljau. A typical Bezier curve is shown in Figure 4. Here four knots $\mathbf{P_0}$, $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$ are shown. The gradient at each end of the Bezier curve is the same as the gradient of the line joining the first two knots, thus: $\mathbf{P'_0} = k(\mathbf{P_1} - \mathbf{P_0})$, where k is the number of knots - 1. This is an important property as it allows us to join Bezier patches together smoothly. Computation of the Bezier Curve may be done in two ways. The first uses a recursive algorithm based on a method of Casteljau. The idea is illustrated by Figure 5. For a given value of μ, say 1/2 we first construct the points on the lines [$\mathbf{P_{0,0}}$, $\mathbf{P_{0,1}}$] [$\mathbf{P_{0,1}}$, $\mathbf{P_{0,2}}$] and [$\mathbf{P_{0,2}}$, $\mathbf{P_{0,3}}$] for the chosen value of μ. These are labelled as the first set of constructed points, $\mathbf{P_{1,0}}$, $\mathbf{P_{1,1}}$ and $\mathbf{P_{1,2}}$. The new points are joined up and the same procedure is followed to construct the second set of points $\mathbf{P_{2,0}}$ and $\mathbf{P_{2,1}}$. The process is repeated to find the point $\mathbf{P_{3,0}}$. This is a point on the Bezier curve. As μ varies from 0 to 1 the locus of $\mathbf{P_{3,0}}$ traces out the Bezier Curve. Using a functional pseudocode which allows us such liberties as scalar and vector multiplications and typed functions, this algorithm can be written very simply:

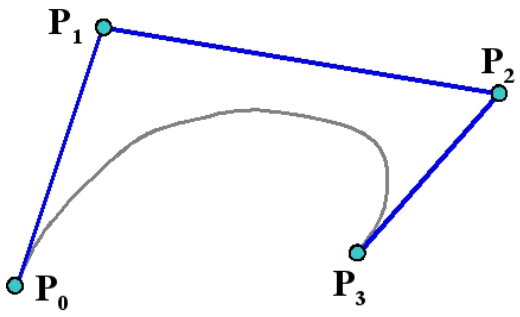


Figure 4: A typical Bezier Curve

```
Point Casteljau (knots P[] ; int N, int r, float μ)
begin
    if (r == 1)
    then Casteljau = μ*P[N+1] + (1-μ)*P[N]
    else Casteljau = μ*Casteljau(P,N+1,r-1,μ) + (1-μ)*Casteljau(P,N,r-1,μ)
end
```

and the curve can be drawn with:

```
for j=1 to L
begin
    locus=Casteljau(Knotarray,0,N,j/L)
    drawto(locus.x,locus.y)
end
```

Note that here, and for all subsequent treatment of splines we will use a set of N+1 knots, labelled 0 to N.

Blending

Another way to view a Bezier curve is to think of it as a blend of its knots. In the simplest case, if we apply the Casteljau algorithm to the degenerate case of two knots we get the parametric line equation:

$$\mathbf{P} = \mu\mathbf{P_{0,1}} + (1 - \mu)\mathbf{P_{0,0}}$$

We can think of this as linearly blending the two points to produce a third. The parameter μ may be thought of as measuring the distance along the line. Like the curve constructed using the Casteljau algorithm, most spline formulations consist of a blend of the positions of the knots. For more than two points the blend is expressed by the iterative formulation of the Bezier Curve:

$$\mathbf{P}(\mu) = \sum_{i=0}^{N} \mathbf{P_i} W(N, i, \mu)$$

where $W(N, i, \mu)$ is called the Bernstein blending function:

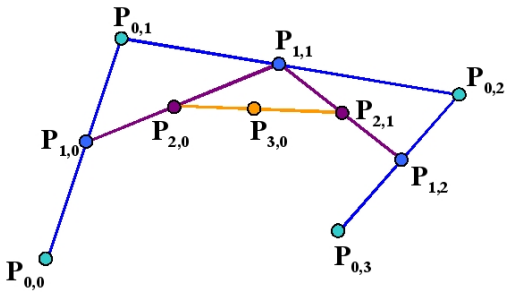


Figure 5: Using Casteljau's construction to draw a Bezier curve

$$W(N, I, \mu) = \binom{N}{i} \mu^i (i - \mu)^{N-i}$$

$$\binom{N}{i} = \frac{N!}{(N-i)!i!}$$

As before we are using a parameter μ to determine the distance along the curve, and it can be easily verified that when μ is 0 or 1 the spline interpolates the end points, $\mathbf{P(0)} = \mathbf{P_0}$ and $\mathbf{P(1)} = \mathbf{P_N}$, and that when $0 < \mu < 1$ then $\mathbf{P}(\mu)$ is a blend of all the knots $\mathbf{P_i}$.

The iterative equation for the Bezier curve can be computed slightly more efficiently in terms of space than the recursive form, though for most applications this is not likely to be significant, since it is rare to use Bezier curves for more than a few points. The iterative solution, though less elegant, generalises to surface construction more easily, and so tends to be used in preference.

Characteristics of Bezier Curves

As previously mentioned, Bezier curves have their end gradient clamped to the slope of the end line segments, and, beyond the ends they blend the positions of all the points. Since Bezier Curves are a blend of all their control points there is little local control over a part of the curve. Figure 6 shows how a large number of control points tends to be ineffectual with Bezier curves. Moving the intermediate points has little effect, and it is not possible to create a curve which wiggles with any degree of complexity. This problem can be offset to some degree by pieceing together a number of sections, as we did for the cubic patches.

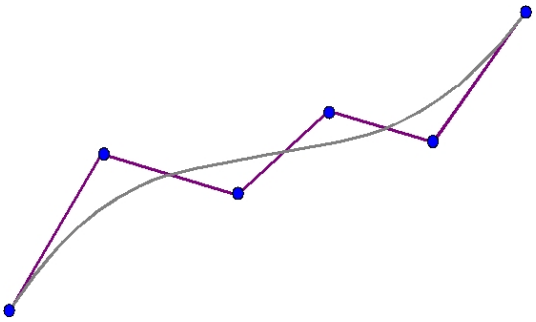


Figure 6: The lack of local detail in Bezier curves

Relation between Bezier Curves and Cubic Patches

We noted previously that the order of the Bezier curve is one less than the number of knots, so for four knots we have a cubic spline. This will be verified by expanding the iterative form of the Bezier curve:

$$\mathbf{P}(\mu) = \sum_{i=0}^{3} \mathbf{P_i} W(3, i, \mu)$$

$$\mathbf{P}(\mu) = \mathbf{P_0}(1-\mu)^3 + 3\mathbf{P_1}\mu(1-\mu)^2 + 3\mathbf{P_2}\mu^2(1-\mu) + \mathbf{P_3}\mu^3$$

If we multiply out the brackets are collect the terms we get:

$$\mathbf{P}(\mu) = \mathbf{a_3}\mu^3 + \mathbf{a_2}\mu^2 + \mathbf{a_1}\mu + \mathbf{a_0}$$

where
$$\mathbf{a_3} = \mathbf{P_3} - 3\mathbf{P_2} + 3\mathbf{P_1} - \mathbf{P_0}$$
$$\mathbf{a_2} = 3\mathbf{P_2} - 6\mathbf{P_1} + 3\mathbf{P_0}$$
$$\mathbf{a_1} = 3\mathbf{P_1} - 3\mathbf{P_0}$$
$$\mathbf{a_0} = \mathbf{P_0}$$

So, we see that the four point Bezier curve is just a cubic spline patch between $\mathbf{P_0}$ and $\mathbf{P_3}$, with two shape control points, $\mathbf{P_1}$ and $\mathbf{P_2}$, which are manipulated by the designer.

We can also show that the Casteljau construction results in the creation of a cubic spline patch. To do this we expand the the recursion backwards:

$$\begin{aligned}
\mathbf{P_{3,0}} &= \mu\mathbf{P_{2,1}} + (1-\mu)\mathbf{P_{2,0}} \\
&= \mu[\mu\mathbf{P_{1,2}} + (1-\mu)\mathbf{P_{1,1}}] + (1-\mu)[\mu\mathbf{P_{1,1}} + (1-\mu)\mathbf{P_{1,0}}] \\
&= \mu^2\mathbf{P_{1,2}} + \mu(1-\mu)\mathbf{P_{1,1}} + (1-\mu)^2\mathbf{P_{1,0}} \\
&= \mu^2[\mu\mathbf{P_{0,3}} + (1-\mu)\mathbf{P_{0,2}}] + \mu(1-\mu)[\mu\mathbf{P_{0,2}} + (1-\mu)\mathbf{P_{0,1}}] \\
&\quad +(1-\mu)^2[\mu\mathbf{P_{0,1}} + (1-\mu)\mathbf{P_{0,0}}]
\end{aligned}$$

if we drop the first subscript, which indicated the construction level of the Casteljau algorithm we get:

$$\begin{aligned}
\mathbf{P}(\mu) &= \mu^2[\mu\mathbf{P_3} + (1-\mu)\mathbf{P_2}] + \mu(1-\mu)[\mu\mathbf{P_2} + (1-\mu)\mathbf{P_1}] \\
&\quad +(1-\mu)^2[\mu\mathbf{P_1} + (1-\mu)\mathbf{P_0}] \\
&= \mu^3\mathbf{P_3} + \mu^2(1-\mu)\mathbf{P_3} + \mu(1-\mu)^2\mathbf{P_1} + (1-\mu)^3\mathbf{P_0}
\end{aligned}$$

which is the same as the blending formulation.

So we can think of a four point Bezier curve as a cubic spline patch with shape control. The curve goes through points $\mathbf{P_0}$ and $\mathbf{P_3}$ and by picking up points $\mathbf{P_1}$ and $\mathbf{P_2}$ with a mouse and moving them we can control the shape as desired.

The Gradients at the spline ends

To determine the gradient at any point of a parametric spline curve we differentiate it with respect to the parameter. Thus if:

$$\mathbf{P}(\mu) = \mathbf{P_0}(1-\mu)^3 + 3\mathbf{P_1}\mu(1-\mu)^2 + 3\mathbf{P_2}\mu^2(1-\mu) + \mathbf{P_3}\mu^3$$

we differentiate to get:

$$\mathbf{P}'(\mu) = -3\mathbf{P_0}(1-\mu)^2 + 3\mathbf{P_1}(1-\mu)^2 - 6\mathbf{P_1}\mu(1-\mu) + 6\mathbf{P_2}\mu(1-\mu) - 3\mathbf{P_2}\mu^2 + 3\mathbf{P_3}\mu^2$$

grouping the terms

$$\mathbf{P}'(\mu) = (3\mathbf{P_1} - 3\mathbf{P_0})(1-\mu)^2 + (6\mathbf{P_2} - 6\mathbf{P_1})\mu(1-\mu) + (3\mathbf{P_3} - 3\mathbf{P_2})\mu^2$$

So at the start, where $\mu = 0$, the gradient is $3\mathbf{P_1} - 3\mathbf{P_0}$ and at the end, where $\mu = 1$, the gradient is $3\mathbf{P_3} - 3\mathbf{P_2}$. This confirms that the curve is tangential to $\mathbf{P_1} - \mathbf{P_0}$ at the start and $\mathbf{P_3} - \mathbf{P_2}$ at the end.

Tutorial 4: Shading

A graphics scene is to made up of a set of triangles. When one of the triangles is in the standard viewing system (viewpoint at the origin) it has vertex coordinates:

Vertex	Coordinates
$\mathbf{P}_1$	(-10, 20, 30)
$\mathbf{P}_2$	(15, 25, 25)
$\mathbf{P}_3$	(5, -20, 50)

Assume that the triangle is visible from the viewpoint.

1. Find the outer normal vector of the surface.

2. The scene is lit by a single light source which is located at position (-2, -40, -50). Assuming that only diffuse lighting is being used, find the brightest point on the triangle.

3. If the triangle is to be drawn using interpolation shading, which will be the brightest point? Assume that the incident light at each point of the triangle is a constant (no inverse square attenuation of the light).

4. Would the result be different if the inverse square law was taken into account?

5. The triangle is part of a bigger surface. A fourth point $\mathbf{P}_4$ at (-25, 25, 40) forms another two triangles. One is with $\mathbf{P}_1$ and $\mathbf{P}_2$, and the other with $\mathbf{P}_1$ and $\mathbf{P}_3$. There are no other faces that meet at $\mathbf{P}_1$.

What is the unit normal vector at $\mathbf{P}_1$ that would be used for Gouraud shading or Phong shading.