1998

# Vector Geometry and Applications to Three-Dimensional Computer Graphics

Rory Morrison
*Edith Cowan University*

# Edith Cowan University

# Copyright Warning

# USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

# Vector Geometry and Applications to Three-Dimensional Computer Graphics

A Thesis Submitted to the
Faculty of Science and Technology
Edith Cowan University
Perth Western Australia

by Rory Morrison

in Partial Fulfilment of the
Requirements for the Degree of

**Bachelor of Science (Mathematics) Honours**

August 1998

## Table of Contents

## Abstract

The mathematics behind algorithms involved in generating three-dimensional images on a computer has stemmed from the analysis of the processes of sight and vision. These processes have been modeled to provide methods of visualising three-dimensional data sets. The applications of such visualisations are varied.

This project will study some of the mathematics that is used in three-dimensional graphics applications.

## Declaration

I certify that this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education; and that to the best of my knowledge and belief that it does not contain any material previously published or written by another person except where due reference is made in the text.

Signature

Date 21/8/98

## Acknowledgements

Many thanks go to my supervisor Geoff Comber, whose guidance, enthusiasm and encouragement, not to mention proof reading skills, went a long way to making this experience a rewarding one. Thanks also to Louise Claxton, who makes many of my other experiences rewarding too.

# 1. Introduction

## 1.1 Background and Significance

Computer graphics applications seek to generate images by processing data that is in part provided by a user. Frequently this data may represent three-dimensional spaces and may be thought of as a *problem world* or *scene*, which the user desires to exist as a visual communication.

In any visual communication of a given problem world, be it a photograph, artist's impression or computer generated image, it is the position from which the world is to be viewed that determines that communication. Typically, this viewing location, the *camera* or *eye*, is a point within the world, perhaps even within an object in that world. From this point the world is viewed through a *view plane* that facilitates the description of the third dimension in only two dimensions by means of projection.

Conventional theory dictates that it is the light within an environment that is received by the eye and interpreted as vision. Not surprisingly then, computer graphics applications aim to communicate visually by applying models simulating the behaviour of light and its interaction with the surfaces of an environment.

## 1.2 An Introduction to Ray Tracing

The modelling of light in an environment centres around the idea that a light source emits particles called photons. These carry energy that is perceived as a colour, should such a photon strike the eye. The path of a photon is assumed to be a straight line through space, and is referred to as a light ray. This may change direction as the light ray collides with or passes through different media in the environment. By identifying

all the light sources present, one could trace the light rays of all photons as they interact with an environment and identify those that are going to strike the eye, and hence generate an image based on the colours received. However, this method is an impractical solution to the image generation problem, since any one light source could emit millions of photons (per unit time), in all directions, filling the environment volume with rays, most of which would eventually have to be discarded for any given viewing location.

A more practical approach is found by identifying the potential paths that light could take to reach a given location (the eye). By placing a view plane in the problem world, a limited number of rays originating at the eye can be cast 'backwards' through this plane into the problem world. Determining the colour of such a ray involves examining the colour and the illumination of any surfaces that it strikes. This process is known as *ray tracing*. The theory behind ray tracing provides a general solution to the image generation problem, and will be examined in some detail in this project.

## 1.3 Significance of the Study

Generating computer graphics, which represent three-dimensional data, highlights the role of the computer as a principal tool in the visualisation process, the application of which can be seen in many areas, ranging from heavy industry to forms of entertainment. More recent developments in this field have centred on the extension and refinement of the application of existing principles. In particular, the rendering of non-visual data has seen much attention; computer images being used to aid understanding of data through visualisation.

## 1.4 Purpose of the Project

This project studies some of the mathematics behind the processes used to create images and effects that are now in evidence in many visually communicative media. It will consider contemporary three-dimensional graphics engines, both those designed for speed and interactivity, and those designed for detail and clarity; and it will present some of the mathematical principles involved.

## 2. Mathematical Framework and Preliminaries

The primary motivation for the following sections is to introduce the notation that will be employed in the rest of this project, and some of the mathematical concepts that are used to describe methods by which a problem world, or environment may be viewed.

### 2.1 Coordinate Spaces

Euclidean space is used to describe and understand the three dimensions of the world around us. We interpret size, length, angles and volumes according to Euclidean geometry, for it is the geometry that we are most familiar with (so much so that the word 'geometry' alone is commonly understood to refer to Euclidean geometry).

A point in Euclidean 3-space may be described by the coordinate triple $(x, y, z)$, where the coordinates represent distances of the point from the various coordinate planes.

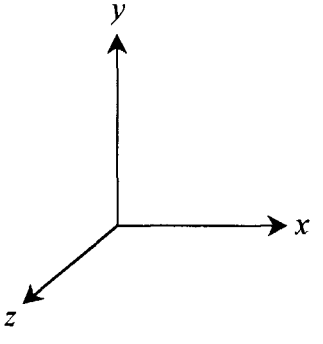



Figure 2.1.
Right-handed coordinate system.

This project will use a *right-handed* coordinate system. The choice of a right-handed or left-handed system is arbitrary as a simple transformation exists to switch between the systems:

$$\mathbf{T}_{RHS\leftrightarrow LHS} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

## 2.2 Vectors

This section introduces the notation that will be used to describe vectors, and summarises the vector operations that will be employed through the course of this project.

### 2.2.1 Notation.

The following alternative notation will be used to describe a vector:

$$\mathbf{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = [a, b, c]^{\mathrm{T}} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$$

where **i**, **j**, **k** are unit vectors in the directions of the principal axes. Although unconventional, the commas will sometimes be used in the horizontal matrix to assist readability.

### 2.2.2 Euclidean Norm of a Vector (Magnitude)

$$\|\mathbf{v}_0\| = \left\| [a_0, b_0, c_0]^{\mathrm{T}} \right\| = \sqrt{a_0^2 + b_0^2 + c_0^2}$$

### 2.2.3 Addition and Subtraction.

$$\mathbf{v}_0 \pm \mathbf{v}_1 = \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} \pm \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} = \begin{bmatrix} a_0 \pm a_1 \\ b_0 \pm b_1 \\ c_0 \pm c_1 \end{bmatrix} = (a_0 \pm a_1)\mathbf{i} + (b_0 \pm b_1)\mathbf{j} + (c_0 \pm c_1)\mathbf{k}$$

2.2.4 Scaling.

$$k\mathbf{v}_0 = k\begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} = \begin{bmatrix} ka_0 \\ kb_0 \\ kc_0 \end{bmatrix} = ka_0\mathbf{i} + kb_0\mathbf{j} + kc_0\mathbf{k}$$

2.2.5 Dot Product.

$$\mathbf{v}_0 \bullet \mathbf{v}_1 = \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} \bullet \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} = a_0a_1 + b_0b_1 + c_0c_1 = \|\mathbf{v}_0\|\|\mathbf{v}_1\|\cos\theta$$

where $\theta$ is the angle between $\mathbf{v}_0$ and $\mathbf{v}_1$ (If $\mathbf{v}_0 \bullet \mathbf{v}_1 = 0$ the vectors are said to be orthogonal).

2.2.6 Cross Product.

$$\mathbf{v}_0 \times \mathbf{v}_1 = -\mathbf{v}_1 \times \mathbf{v}_0 = \begin{vmatrix} a_0 & b_0 & c_0 \\ a_1 & b_1 & c_1 \\ \mathbf{i} & \mathbf{j} & \mathbf{k} \end{vmatrix} = (b_0c_1 - b_1c_0)\mathbf{i} + (a_1c_0 - a_0c_1)\mathbf{j} + (a_0b_1 - a_1b_0)\mathbf{k}$$

$$\text{and } \|\mathbf{v}_0 \times \mathbf{v}_1\| = \|\mathbf{v}_0\|\|\mathbf{v}_1\|\sin\theta$$

The cross product of any two vectors generates a vector that is orthogonal to both of those vectors. If we take the two vectors as lying on a plane, then the cross product defines the normal vector to the plane.

## 2.3 Homogeneous Coordinates

3-space exists as a subset of 4-space, which itself may be described by the coordinates $(X,Y,Z,W)$. As such, a point $(x,y,z)$ in 3-space is equivalent to the 4-tuple $(X,Y,Z,W)$ if and only if:

$$\frac{X}{W}=x,\ \frac{Y}{W}=y,\ \frac{Z}{W}=z.$$

It is convenient to choose the case for which $W=1$, allowing $(x,y,z)$ to be written *homogeneously* as $(x,y,z,1)$. Whilst this may seem to be of trivial notational value (we can already define a complete 3-dimensional problem world using only $(x,y,z)$), the use of homogeneous coordinates is useful in certain computations, which will be detailed below.

## 2.4 Elementary Euclidean Transformations.

We will be concerned with two types of transformations, those that transform the coordinate space relative to a fixed coordinate system, and the inverse transformations that transform the coordinate system itself to another. Generally, the matrix **M** will be used to denote transformations of the first type, and $\mathbf{M}^{-1}$ for those of the second type.

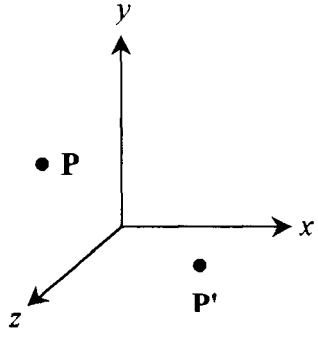



Figure 2.4.a.
Transformation of coordinate space by **M**.

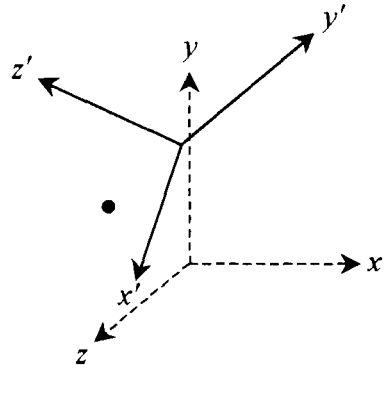



Figure 2.4.b.
Transformation of coordinate axes by $\mathbf{M}^{-1}$.

The matrix **M** describes a manipulation of a coordinate space relative to its own axes. Such transformations will be the standard for this project (as opposed to manipulation of coordinate axes with a fixed space), and the focus of the following discussion.

### 2.4.1 General Form of a Transformation.

All transformations in this project are stated in the premultiplicative form $\mathbf{P}' = \mathbf{MP}$, where **M** is the $4 \times 4$ transformation matrix, and **P** is the (augmented) column vector representing the points with homogeneous coordinates $(x, y, z, 1)$.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

i.e. $\mathbf{P}' = \mathbf{MP}$

### 2.4.2 Scaling

Scaling refers to the multiplication of each coordinate by some value:

$$x' = s_x x$$
$$y' = s_y y$$
$$z' = s_z z$$

This is more conveniently represented as the matrix transformation **S**.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

i.e. $\mathbf{P}' = \mathbf{SP}$

The inverse transformation (restoring the original coordinate values) is intuitively obvious. The scaling constants of the inverse matrix are simply the reciprocals of those in the original transformation matrix.

### 2.4.3 Rotation

The rotation of points and vectors in cartesian 3-space is readily handled by considering rotations about each axis separately, and then applying each transformation in turn to the object which is to be rotated.

#### 2.4.3.1 Rotation of a point / vector about the z axis.

In a right-handed coordinate system, a positive rotation about the $z$ axis rotates points on the $x$ axis towards the $y$ axis. The transformation matrix $\mathbf{R}_z$ is given:

$$\mathbf{R}_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

so that $\mathbf{P}' = \mathbf{R}_z\mathbf{P}$

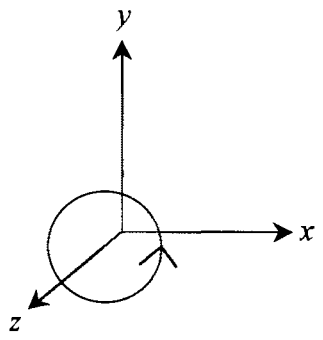



Figure 2.4.3.1. A positive rotation about the $z$ axis.

2.4.3.2 Rotation of a point / vector about the x axis.

In a right-handed coordinate system, a positive rotation about the $x$ axis rotates point on the $y$ axis towards the $z$ axis. The transformation matrix $\mathbf{R}_x$ is given:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

so that $\mathbf{P}' = \mathbf{R}_x\mathbf{P}$

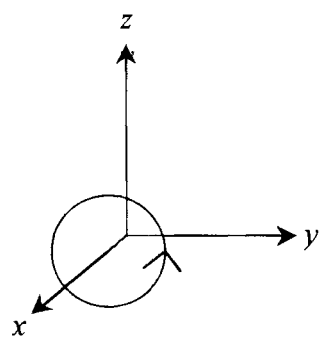



Figure 2.4.3.2. A positive rotation about the $x$ axis.

2.4.3.3 Rotation of a point / vector about the y axis.

In a right-handed coordinate system, a positive rotation about the $y$ axis rotates points on the $z$ axis towards the $x$ axis. The transformation matrix $\mathbf{R}_y$ is given:

$$\mathbf{R}_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

so that $\mathbf{P}' = \mathbf{R}_y\mathbf{P}$

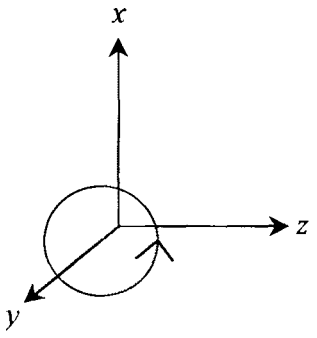



Figure 2.4.3.3. A positive rotation about the $y$ axis.

The inverse of a rotation of θ about an arbitrary axis is a rotation by -θ about the same axis. For example:

$$\mathbf{R}_y^{-1} = \begin{bmatrix} \cos(-\theta_y) & 0 & \sin(-\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-\theta_y) & 0 & \cos(-\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The reader will note that $\mathbf{R}_y^{-1} = \mathbf{R}_y^{\mathrm{T}}$. Matrices for which the inverse is equal to the transpose are known as orthogonal. In practical terms, the property allows an inverse to be found by exchanging row and column references. This can save on storage space and speed computations (a second matrix does not need to be computed, instead the order in which columns and rows are referenced is reversed ).

### 2.4.4 Translation

The translation of a point is defined by a shift in each of its coordinates:

$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

One of the primary reasons that homogeneous coordinates are used in computer graphics computations is they facilitate the representation of translations by matrix multiplication. The transformation matrix for a translation is:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

so that $\mathbf{P}' = \mathbf{TP}$

Again, the inverse matrix is intuitively obvious. To reverse the transformation applied by **T**, one would simply apply a translation of the same magnitude in the opposite direction.

### 2.4.5 Composite Transformations

Given the ability to rotate and translate a set of points or vectors, it is possible to position that set anywhere in Cartesian 3-space with the orientation (shape / relative position of points) of that set intact. Whilst one could perform the necessary transformations one step at a time, it is more compact to describe the whole *composite* transformation at once. Simply put, sequential transformation matrices may be multiplied together to provide a matrix that describes the total transformation. One must note that the ordering of this action is important, given the non-commutativity of matrices under multiplication. Whilst some composite transformations do not require such caution, the general form of a composite transformation is strictly ordered:

If the matrix transformations are applied in the order $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_j$, the composite transformation is defined:

$$\mathbf{M} = \mathbf{M}_j \ldots \mathbf{M}_2 \mathbf{M}_1$$

So the process of j transformations:

$$\begin{aligned} \mathbf{P}' &= \mathbf{M}_1 \mathbf{P} \\ \mathbf{P}'' &= \mathbf{M}_2 \mathbf{P}' \\ \mathbf{P}''' &= \mathbf{M}_3 \mathbf{P}'' \\ &\vdots \\ \mathbf{P}^{(i)} &= \mathbf{M}_i \mathbf{P}^{(i-1)} \end{aligned}$$

$$\text{reduces to} \quad \mathbf{P}^{(i)} = \mathbf{MP}$$

The inverse of this product is the product of the inverses of the original transformations in reverse order:

$$\mathbf{M}^{-1} = \mathbf{M}_1^{-1} \mathbf{M}_2^{-1} \ldots \mathbf{M}_j^{-1}$$

This follows from the matrix property that the inverse of a matrix product is the reversed product of the individual inverses: $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$. Moreover, it makes sense geometrically in that one would invert the individual transformations from last to first to find the original points / vectors that were transformed.

The use of composite transformations allows a number of more general transformations to be defined.

2.4.5.1 Scaling from an Arbitrary Point **d**.

The standard scaling matrix **S** represents a dilatation with centre at the origin. When we are concerned with a set of points, for example defining the vertices of an object in 3-space, we may need to scale from some point represented by $\mathbf{d} = [d_x, d_y, d_z, 1]^{\mathrm{T}}$, and

not the origin. This can be accomplished by first translating all points so that $\mathbf{d}' = [0,0,0,1]^T$. This transformation is:

$$\mathbf{T}_d = \begin{bmatrix} 1 & 0 & 0 & -d_x \\ 0 & 1 & 0 & -d_y \\ 0 & 0 & 1 & -d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The points can then be scaled according to the constants $s_x$, $s_y$, $s_z$. Applying the inverse translation $\mathbf{T}_d^{-1}$ to the scaled points positions them so that their only displacement was due to the scaling. The complete transformation can be written:

$$\mathbf{S_d} = \mathbf{T}_d^{-1}\mathbf{S}\mathbf{T}_d$$

$$= \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -d_x \\ 0 & 1 & 0 & -d_y \\ 0 & 0 & 1 & -d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{S_d} = \begin{bmatrix} s_x & 0 & 0 & d_x(1-s_x) \\ 0 & s_y & 0 & d_y(1-s_y) \\ 0 & 0 & s_z & d_z(1-s_z) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.4.5.2 Composite Rotation

The three rotation matrices, $\mathbf{R}_x, \mathbf{R}_y$, and $\mathbf{R}_z$, each perform a rotation about the named axis. A matrix **R** can be used to describe a general rotation of coordinate space about an arbitrary axis through the origin. The order in which the rotations are performed is a determining factor in the transformation (along with the size of the angles specified), and, as such, **R** has no set form and must be computed as a sequence of matrix multiplications each time it is used.

**R** has the general form:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Consider a rotation about an arbitrary axis through the origin:

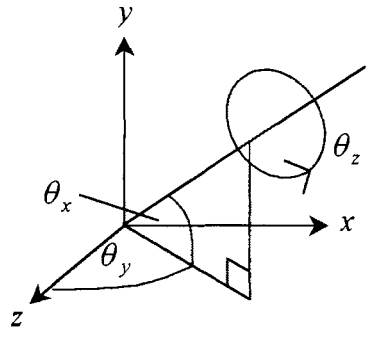



Figure 2.4.5.2. Rotation about an arbitrary axis.

The rotation of coordinate space about the arbitrary axis is obtained by a rotation about the $y$ axis, and then about the $x$ axis, and subsequently about the $z$ axis (on which now lies the arbitrary axis), before two rotations to restore the principal axes to their original position. That is $\mathbf{R} = \mathbf{R}_y\mathbf{R}_x\mathbf{R}_z\mathbf{R}_x^{\mathrm{T}}\mathbf{R}_y^{\mathrm{T}}$:

a rotation of $-\theta_y$ about the $y$ axis,

a rotation of $\theta_x$ about the $x$ axis,

a rotation of $\theta_z$ about the $z$ axis,

a rotation of $-\theta_x$ about the $x$ axis and finally,

a rotation of $\theta_y$ about the $y$ axis.

A worked example of this type of problem may be found in Appendix 1.

2.4.5.3 Rotating About an Axis through an Arbitrary Point **d**.

The arbitrary axis about which a rotation occurs need not pass through the origin. In much the same way as was done for the scaling matrix, a point **d** can be defined through which the axis of rotation passes, and then:

$$\mathbf{R_d} = \mathbf{T}_d^{-1}\mathbf{R}\mathbf{T}_d$$

$$\mathbf{R_d} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{R} \begin{bmatrix} 1 & 0 & 0 & -d_x \\ 0 & 1 & 0 & -d_y \\ 0 & 0 & 1 & -d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## 2.5 Defining Elements of a Scene.

The goal of a graphics application is to display an internally represented (i.e. stored in the computer) scene consisting of an array of objects. In this project the term *object* refers to a bounded volume, or some part of a surface. The following sections discuss the various treatments which may be applied to these objects in order to display them.

### 2.5.1 Object Space.

The concept of object space is a simple one. Rather than define each element of a scene in the coordinates of that scene (termed *world* coordinates, usually denoted $(x, y, z)$), each element is defined in its own coordinate space $(u_i, v_i, w_i)$. The object may then be scaled to the appropriate size, rotated to the desired orientation, and translated to the desired location through the use of the transformation matrices seen in the previous section.

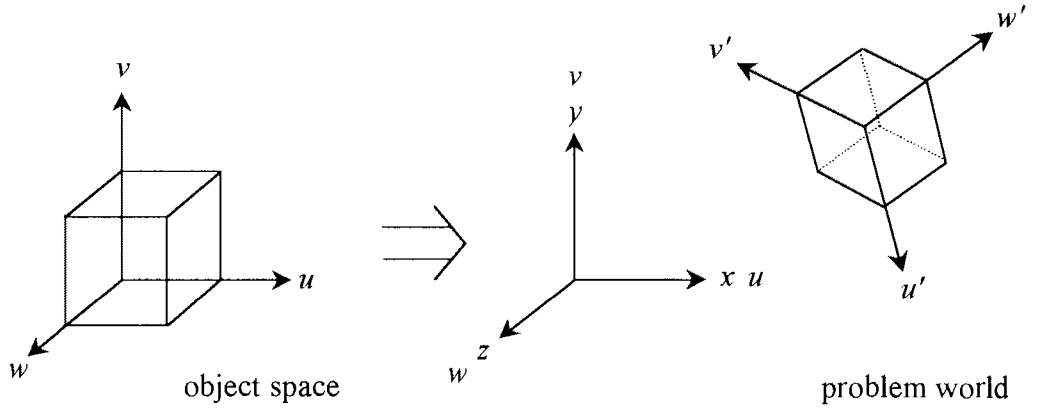



Figure 2.5.1. Object space.

### 2.5.2 Basic Mathematics of Object Definition

The processes to be detailed in this project require that the features or elements comprising a scene be represented mathematically. This mathematical modelling of such features involves describing their geometry in 3-space. Two broad categories of geometric model exist (Hanrahan, in Glassner, 1989): the models that are described by *classification* (i.e. those that are defined implicitly), and those that are described *enumeratively* (parametrically).

### 2.5.3 Implicit Definitions of Objects

The method of classification defines each element of a scene *implicitly*, according to a function of the coordinates of the three space in which it lies. The term implicit is used because the object itself is not described explicitly by the function, rather the regions of the three space in which it lies are identified as either being inside the object, outside the object, or lying on the boundary between these two classifications:

$$F(u,v,w)\begin{cases}<0 & \text{inside the object}\\ =0 & \text{on the surface of the object}\\ >0 & \text{outside of the object}\end{cases}$$

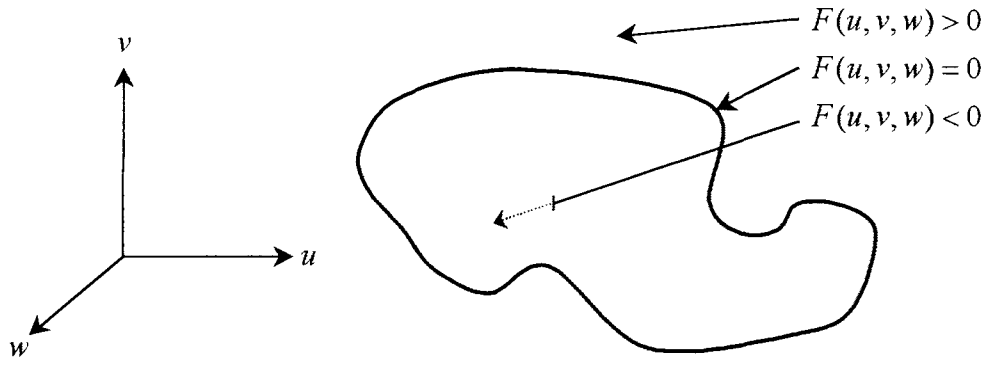



Figure 2.5.3. Object description using classification of coordinate space.

Whilst an implicit definition does not always lend itself to an intuitive visualisation of the surface it is ideal for use within a ray tracing algorithm, as all ray–surface intersections can be defined exactly by replacing each variable of $F$ with the equivalent parametrically defined coordinate of each particular ray, and then solving for the parameter, as follows.

Given the ray (in object space): $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$, for $t > 0$.

The parametric equations of the ray are:

$$u = u_0 + tu_t$$
$$v = v_0 + tv_t$$
$$w = w_0 + tw_t$$

The intersections of this ray with an object defined by $F(u,v,w) = 0$ can be found by solving the following equation for t, and substituting the returned value(s) back into the parametric equations above:

$$F(u_0 + tu_t, v_0 + tv_t, w_0 + tw_t) = 0$$

This equation will be abbreviated by $F^{\mathbf{r}}(t) = 0$, where $F^{\mathbf{r}}(t)$ is a function of coordinate values along ray **r**, which vary with $t$.

A more detailed discussion about the geometric interpretations of the solutions for this type of equation will follow in later sections, but generally the first positive intersection (i.e. corresponding to the value $\min(t : t > 0)$) is the important result.

### 2.5.4 Explicit Definitions of Objects

Typically elements of a scene that are defined explicitly are expressed as mappings from a set of parameters to a set of points in 3-space.

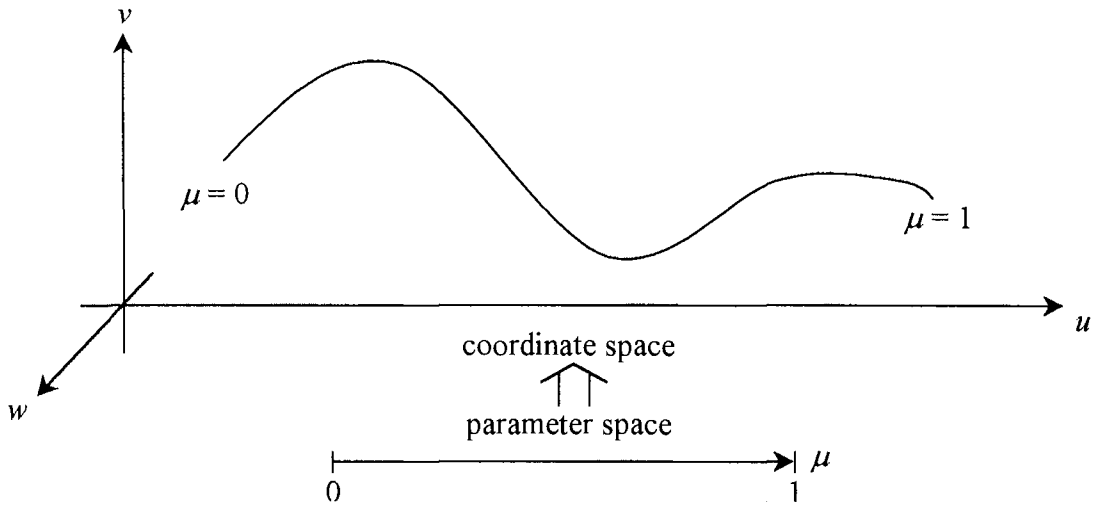



Figure 2.5.4. Diagrammatic representation of a parameter to point mapping.

To represent a surface, two parameters are required (a single parameter will describe a curve, three parameters will describe a volume). The coordinates of a point on such a

surface $S(\mu,\nu)$ are defined by $(u(\mu,\nu),v(\mu,\nu),w(\mu,\nu))$, where μ and ν are the two parameters.

The surface, then, is described as the parameters vary independently of each other through some specified domains. For example, a plane may be described explicitly as:

$$\mathbf{P} = \mu\mathbf{P}_\mu + \nu\mathbf{P}_\nu + \mathbf{P}_0$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mu \begin{bmatrix} u_\mu \\ v_\mu \\ w_\mu \end{bmatrix} + \nu \begin{bmatrix} u_\nu \\ v_\nu \\ w_\nu \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix} \qquad \text{for } \mu, \nu \in \mathbf{R}.$$

where $\mathbf{P}$ is a general point on the plane corresponding to the parameters $(\mu,\nu)$, $\mathbf{P}_\mu$ and $\mathbf{P}_\nu$ are independent vectors parallel to the plane, and $\mathbf{P}_0$ is some arbitrary fixed point on the plane.

The intersection of an arbitrary parametric surface such as $S(\mu,\nu)$ with a parametrically defined ray is not readily found. Mathematically, it is more convenient to treat the ray or line implicitly as the intersection of two planes, and then substitute the above parametric definition of each coordinate on the surface into each of the plane equations. Suppose $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ is represented as the intersection of the two planes:

$$a_1u + b_1v + c_1w + d_1 = 0 = \mathrm{F}_1(u,v,w)$$
$$a_2u + b_2v + c_2w + d_2 = 0 = \mathrm{F}_2(u,v,w)$$

The intersection of **r** with the surface $S(\mu,\nu)$ defined by $(u(\mu,\nu),v(\mu,\nu),w(\mu,\nu))$ may now be written:

$$F_1^S(\mu,\nu) = a_1u(\mu,\nu) + b_1v(\mu,\nu) + c_1w(\mu,\nu) + d_1 = 0$$
$$F_2^S(\mu,\nu) = a_2u(\mu,\nu) + b_2v(\mu,\nu) + c_2w(\mu,\nu) + d_2 = 0$$

where each $F^S(\mu,\nu)$ is a function of the coordinate values of surface $S$, which vary with $\mu$ and $\nu$.

The equations $F_1^S(\mu,\nu) = 0$ and $F_2^S(\mu,\nu) = 0$ may be solved to provide the parameter pair $(\mu,\nu)$ of any intersections that the ray makes with the surface. As the ray parameter $t$ is not used, the first intersection in terms of the direction of the travel of the ray is not immediately obvious in the event that more than one (real) intersection is made. The coordinates of each intersection, then, need to be compared to the origin of the ray to determine this initial intersection.

The conversion of a parametric, explicitly defined element to an implicit one is a process referred to as implicitization (sic). To find the parameters for a given point on an implicit surface, the reverse process, is referred to as inversion. Both techniques will be discussed in the ensuing descriptions of objects and with reference to their usage in computer graphics applications.

## 3. Ray Tracing and Ray Casting – Mathematical Discussion

Ray tracing identifies the potential paths that light could take about a scene (the problem world) to reach a given location (the eye). The ray tracing procedure may be described in broad terms by the following algorithm. The bracketed numbers refer to the section where the procedure is discussed.

*Given the problem world:* (3.1)
*select the centre of projection (eye) and the view window.* (3.2)
***for*** *each pixel in view window;*
*identify the ray from the centre of projection through that pixel.* (3.3, 3.4)
****for*** *each ray;*
***for*** *each object surface in the problem world;*
*test for the ray – surface intersection(s),* (3.5, 4)
***if*** *there is an intersection;*
*note the ray parameter(s).*
*identify the first intersection of the ray with the problem world,*
*identify the point of the first intersection,*
*compute the normal to the surface at this point of intersection.*
***for*** *each light source in the problem world; (5)*
*identify the ray from the point of intersection to the light,*
*test if the point is obscured from this light.*
***if*** *the surface is reflective;*
*cast a secondary ray in the direction of reflection and return to *.*
***if*** *the surface is transparent;*
*cast a secondary ray in the direction of refraction and return to *.*
*colour the pixel represented by the ray using illumination model and information received for reflection and/or transparency rays.*

The above algorithm is by no means an extensive representation of the ray tracing process, but serves as an introduction to the concepts. Further additions and extensions

will be noted if they have a substantial mathematical basis. The stated algorithm is also without reference to any form of optimisation techniques, which are essential if an implementation is to be rendered practical. Whilst the implementation of the algorithm is not a concern of this project, the structure described provides a framework for the discussion of the mathematics involved in generating images of three dimensional scenes.

## 3.1 Specification of the Problem World.

The method of construction of the problem world will vary greatly between applications, however the presence of some starting world coordinate system is an initial inclusion to all of these. If the application were for visualisation of existing (three dimensional or volumetric) data, then some coordinate system would already be specified by the sampling process. Whether or not this coordinate system would be used during the ray-tracing process is an arbitrary decision, as it is convenient to re-specify the problem world anyway in terms of the position and angle from which it is to be viewed before ray tracing from that orientation.

Choosing to discuss further points with reference to an object coordinate system $(u, v, w)$, we now consider the placement of objects, surfaces, and other data sets in the problem world, facilitated by the Euclidean transformations discussed in section 2.4. For each object, surface or data set $i$ to be placed in the problem world $(x, y, z)$ we can specify the required transformation by matrix $\mathbf{M}_i$:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{M}_i \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

where $\mathbf{M}_i = \mathbf{T}_i \mathbf{S}_i \mathbf{R}_i$ and

$$\mathbf{R}_i = \mathbf{T}_{d(r)}^{-1} \mathbf{R} \mathbf{T}_{d(r)}$$

$$\mathbf{S}_i = \mathbf{T}_{d(s)}^{-1} \begin{bmatrix} s_u & 0 & 0 & 0 \\ 0 & s_v & 0 & 0 \\ 0 & 0 & s_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{T}_{d(s)},$$

$$\mathbf{T}_i = \begin{bmatrix} 1 & 0 & 0 & t_u \\ 0 & 1 & 0 & t_v \\ 0 & 0 & 1 & t_w \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**R** is the composite rotation matrix for object *i* (see section 2.4.5.2). Both the scaling and rotation transformations have been specified for operations about a particular point **d**(s) and **d**(r) respectively (see section 2.4.5.1 and 2.4.5.3), although a careful specification of object coordinates when defining object space could obviate the need for some of these transformations.

We position the object *i* in the problem world by using $\mathbf{M}_i$ to transform the object space with respect to the coordinate axes *u*, *v* and *w*, and label the resulting coordinates $(x, y, z)$ instead of $(u, v, w)$.

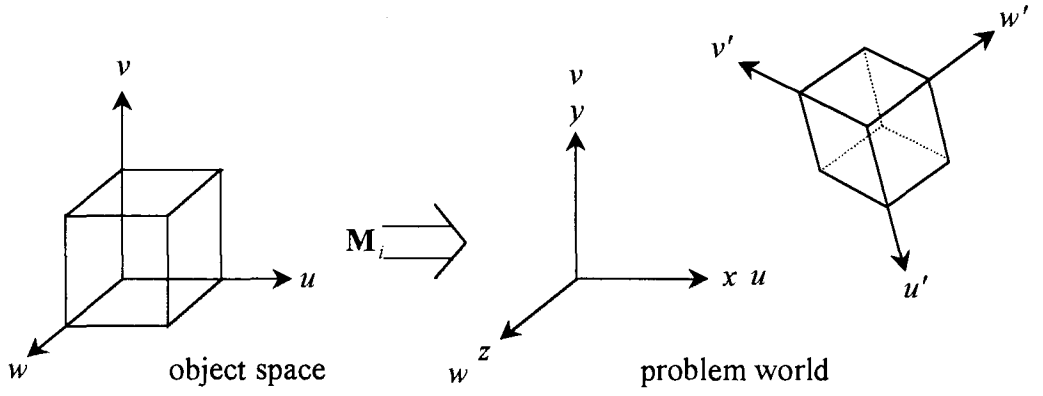



Figure 3.1. Positioning of object *i* in the problem world.

The specific ordering of the composition of **M** (a rotation followed by a scaling and then the translation to the desired location in the problem world) is a necessity not of a mathematical nature, but of a practical one for the programmer. The manual construction of a problem world through the manipulation of a number of objects and surfaces is simplified somewhat by defining relative scale and orientation (rotation) before the translation to the final location. The ordering of the scaling and rotation operations themselves is of less importance; it might be more precise when using some graphical interface to rotate a larger unscaled object than a minuscule one, but this is of no matter to the internal (representation on the computer) construction of the matrix **M**.

It is noted here that while we may consider the rotation and translation of an object not to affect its shape, scaling can produce alterations in shape. If the scaling constants $s_u$, $s_v$, and $s_w$ are not equal then the object can be distorted. For example, a sphere scaled with $s_u > s_v = s_w$ becomes an ellipsoid. These effects may be put to good use in extending the number of shapes available to the construction.

With the problem world in place we can begin to examine its representation in only two dimensions, that is on a view plane.

## 3.2 Specifying the Viewplane.

The viewplane is the plane through which the observer views the problem world. On this plane a view *window* is defined (which itself is often referred to as the viewplane, but will not be so in this project), which is a coordinate representation of the boundaries of the intended output. One can consider a rectangle of transparent material in front of them as an analogy to the role of this window.

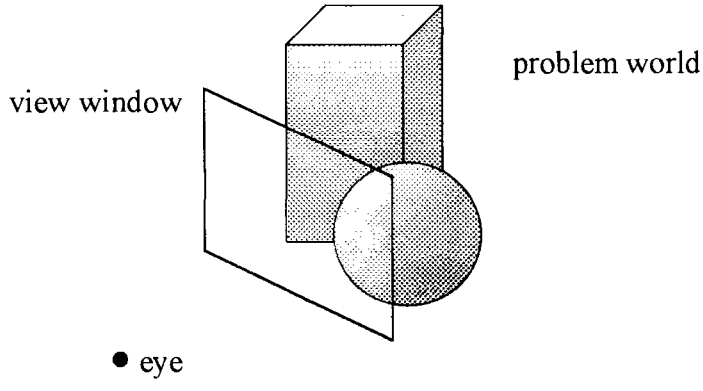



Figure 3.2. A view window in a problem world, illustrating the analogy drawn to a rectangle of transparent material.

The window is divided into a regular grid, the number of divisions determining the resolution of the final image. When making this choice of resolution consideration would be given to the resolution of the output (screen or printer), the desired level of detail in the image and the desired speed of the output.

### 3.2.1 Definition of Viewplane Space.

Typically the viewplane exists in its own coordinate space, with its normal running along one of the axes of that space. The position of the viewer is also indicated in this space. Most frequently the coordinates are labelled $(u, v, n)$, with the viewplane unit normal defined as $\mathbf{n} = [0, 0, -1]$ for a right-handed system (and $\mathbf{n} = [0, 0, 1]$ for a left-handed system). A typical view *system* construction would see the centre of the view window located at $(0, 0, -1)$ and bounding vertices $(-a, b, -1)$, $(a, b, -1)$, $(a, -b, -1)$ and $(-a, -b, -1)$, with the viewer location at $(0, 0, 0)$. Variations on this will be discussed in the subsequent sections.

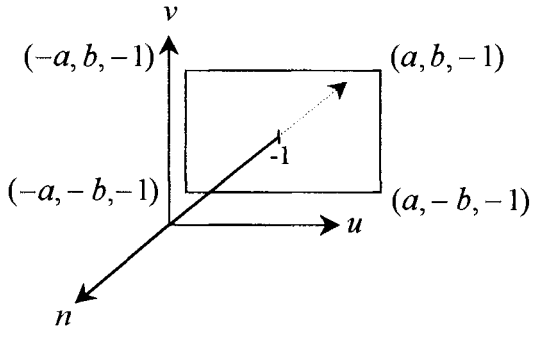



Figure 3.2.1.1. Specification of the view window.

Resolution is defined horizontally and vertically by the constants $g$ and $h$ (i.e. a $g \times h$ array of pixels), the central position (and name) of each pixel denoted $(i, j)$, where $0 \le i \le g - 1$, and $0 \le j \le h - 1$, for the $i, j \in \mathbb{N}$. Non-central positions on pixels such as pixel boundaries (used for the purpose of increasing the sampling of the problem world) can be referred to in terms of $(i, j)$ where $-0.5 \le i \le g - 0.5$, and $-0.5 \le j \le h - 0.5$, for $i, j \in \mathbb{R}$.

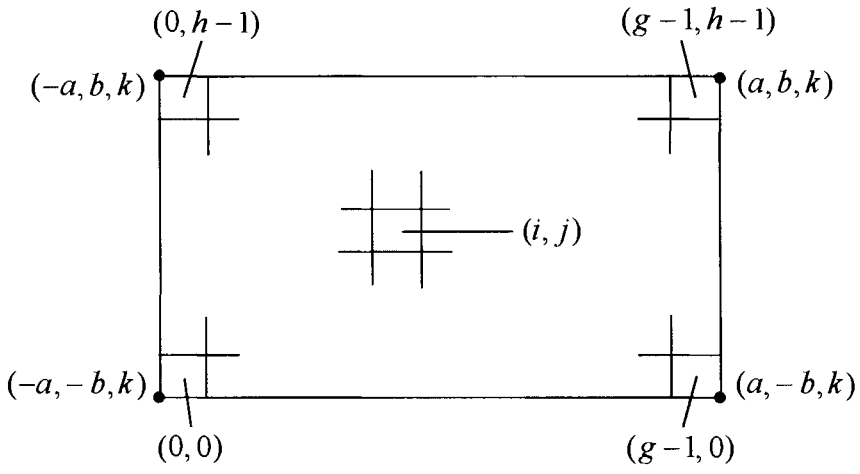



Figure 3.2.1.2 The pixels on the view window.

3.2.2 Pixel to Coordinate Mapping.

The relationship between the view system coordinates $(u, v, n)$ and the view window pixel references $(i, j)$ can be described by the one to one mapping $V : (i, j) \rightarrow (u, v, n)$. This mapping is defined by the resolution $(g \times h)$ and the size $(2a \times 2b)$ of the view window and by the orientation of the view plane in the view system. The following expresses this relationship. For a view window with centre $(0, 0, k)$ and bounding vertices $(-a, b, k)$, $(a, b, k)$, $(a, -b, k)$, $(-a, -b, k)$, describing a resolution of $g \times h$, it is easily shown that:

$$V: \quad \begin{aligned} u &= a\left(\frac{2i+1}{g} - 1\right) \\ v &= b\left(\frac{2j+1}{h} - 1\right) \\ n &= k \end{aligned} \qquad \begin{aligned} &-0.5 \le i \le (g - 0.5) \\ &-0.5 \le j \le (h - 0.5) \end{aligned} \quad \text{for } i, j \in \mathrm{R} \tag{3.2.2}$$

where $V$ generates $(u, v, n)$, the view system coordinates of the point on the view window pixel $(i, j)$ (window coordinates on a pixel $(i, j)$ vary for $[(i - 0.5), (i + 0.5)]$ and $[(j - 0.5), (j + 0.5)]$). A worked example of this mapping is provided in Appendix 1 (Example 3.2.2).

The view system may be placed into the problem world using a composite matrix transformation, $\mathbf{V} = \mathbf{T}_V \mathbf{S}_V \mathbf{R}_V$, according to the position and orientation from which the world is to be viewed. Having stated the above, the task of defining rays can begin. It is noted here that the inverse transformation $\mathbf{V}^{-1}$ applied to all world coordinates (i.e., all objects and surfaces would be associated with the matrix $\mathbf{V}^{-1}\mathbf{M}_i$, where $\mathbf{M}_i$ is object $i$'s transformation matrix) would effectively describe the problem world relative to the viewer.

## 3.3 Definition of the Rays.

The goal of the ray tracing application is to colour the pixels of the view window. This is accomplished by casting one or more rays through each pixel. The definition of each of these rays is presented as follows.

### 3.3.1 General Form of a Ray.

Explicit definition: $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$, where

$\mathbf{r}_0 = [x_0, y_0, z_0]^T$ is the vector indicating the point where the ray begins,

$\mathbf{r}_t = \frac{1}{\sqrt{x_t^2 + y_t^2 + z_t^2}}[x_t, y_t, z_t]^T$ is the unit vector in the direction of the ray,

$t$ is the parameter explicitly defining points along the ray.

The benefits of using a unit vector for $\mathbf{r}_t$ stem from an application of the dot product. Recall that $\mathbf{v}_0 \bullet \mathbf{v}_1 = \|\mathbf{v}_0\|\|\mathbf{v}_1\|\cos\theta$. If both $\mathbf{v}_0$ and $\mathbf{v}_1$ are unit vectors, then the cosine of the angle between the two is $\mathbf{v}_0 \bullet \mathbf{v}_1$. This simplification can save much

computational effort. Furthermore, if all rays are expressed in terms of unit vectors, then the parameter $t$ can be considered a uniform measure of the distance from the eye.

For $t: -\infty < t < \infty$, the formula for $\mathbf{r}$ describes a line. For the purposes of intersecting the ray with explicitly defined surfaces and objects, it is useful to have an implicit definition of this line. Such a definition is given in terms of the intersection of two planes, the implicitization of which we now explain.

The problem requires the definition of two planes on which the ray represented by vector $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ lies. Any point $\mathbf{p}$ on a plane can be defined by its relationship to coplanar point $\mathbf{p}_0$ and the normal vector $\mathbf{n}_{\text{plane}}$ by $(\mathbf{p} - \mathbf{p}_0) \bullet (\mathbf{n}_{\text{plane}}) = 0$. The point $\mathbf{r}_0$ may be substituted for $\mathbf{p}_0$ in this example. The normal vector to the plane may be found using the cross product of any two coplanar vectors. $\mathbf{r}_t$ describes one such vector, and since we seek any two planes that intersect along $\mathbf{r}_t$, the choice of the other vector (for each plane) is an arbitrary one. We will make use of the vectors normal to $\mathbf{r}_t$, which are infinite in number, but three of which are simply defined as follows.

Given $\mathbf{r}_t = [x_t, y_t, z_t]^{\mathrm{T}}$ three normal vectors are defined:

$$\mathbf{n}_k = [y_t, -x_t, 0]^{\mathrm{T}}$$

$$\mathbf{n}_j = [-z_t, 0, x_t]^{\mathrm{T}}$$

and

$$\mathbf{n}_i = [0, z_t, -y_t]^{\mathrm{T}}$$

All of these are clearly orthogonal to $\mathbf{r}_t$. The subscript of each normal refers to the axis to which it is also normal. Note that the opposites of each of these vectors could also be used, but would not provide any extra implicit definitions as –1 could be factored out, carried through the following equations, and divided away at the end.

The cross product of each of these vectors with the ray direction vector $\mathbf{r}_t$ defines a vector normal to a plane on which $\mathbf{r}_t$ lies, thus $\mathbf{n}_{\text{plane } k} = \mathbf{r}_t \times \mathbf{n}_k$, $\mathbf{n}_{\text{plane } j} = \mathbf{r}_t \times \mathbf{n}_j$, and $\mathbf{n}_{\text{plane } i} = \mathbf{r}_t \times \mathbf{n}_i$. These planes are described by all vectors $\mathbf{p}$ for which $\mathbf{p} \bullet \mathbf{n}_{\text{plane}} = 0$, where $\mathbf{p}$ itself can be represented as the vector difference between *any* point $[x, y, z]^{\mathrm{T}}$ on that plane and a given point on the plane, in this case $\mathbf{r}_0 = [x_0, y_0, z_0]^{\mathrm{T}}$. So we have, for the plane containing $\mathbf{n}_k$:

$$
\begin{aligned}
[x - x_0, y - y_0, z - z_0]^{\mathrm{T}} \bullet \mathbf{n}_{\text{plane } k} &= 0 \\
[x - x_0, y - y_0, z - z_0]^{\mathrm{T}} \bullet (\mathbf{r}_t \times \mathbf{n}_k) &= 0 \\
[x - x_0, y - y_0, z - z_0]^{\mathrm{T}} \bullet ([x_t, y_t, z_t]^{\mathrm{T}} \times [y_t, -x_t, 0]^{\mathrm{T}}) &= 0 \\
[x - x_0, y - y_0, z - z_0]^{\mathrm{T}} \bullet [x_t z_t, y_t z_t, -x_t^2 - y_t^2]^{\mathrm{T}} &= 0 \\
[(x - x_0)(x_t z_t) + (y - y_0)(y_t z_t) + (z - z_0)(-x_t^2 - y_t^2)] &= 0
\end{aligned}
$$

$$
(x_t z_t)x + (y_t z_t)y + (-x_t^2 - y_t^2)z + (-x_0 x_t z_t - y_0 y_t z_t + z_0 x_t^2 + z_0 y_t^2) = 0 \qquad (3.3.1.1)
$$

This has the form $ax + by + cz + d = 0$, the implicit form of a plane. By similar expansions:

$$
\begin{aligned}
[x - x_0, y - y_0, z - z_0]^{\mathrm{T}} \bullet \mathbf{n}_{\text{plane } j} &= 0 \\
[x - x_0, y - y_0, z - z_0]^{\mathrm{T}} \bullet ([x_t, y_t, z_t]^{\mathrm{T}} \times [-z_t, 0, x_t]^{\mathrm{T}}) &= 0 \\
[x - x_0, y - y_0, z - z_0]^{\mathrm{T}} \bullet [x_t y_t, -x_t^2 - z_t^2, y_t z_t]^{\mathrm{T}} &= 0
\end{aligned}
$$

$$
(x_t y_t)x + (-x_t^2 - z_t^2)y + (y_t z_t)z + (-x_0 x_t y_t + y_0 x_t^2 + y_0 z_t^2 - z_0 y_t z_t) = 0 \qquad (3.3.1.2)
$$

and

$$[x-x_0, y-y_0, z-z_0]^{\mathrm{T}} \bullet \mathbf{n}_{\text{plane } i} = 0$$

$$[x-x_0, y-y_0, z-z_0]^{\mathrm{T}} \bullet ([x_t, y_t, z_t]^{\mathrm{T}} \times [0, z_t, -y_t]^{\mathrm{T}}) = 0$$

$$[x-x_0, y-y_0, z-z_0]^{\mathrm{T}} \bullet [-y_t^2 - z_t^2, x_t y_t, x_t z_t]^{\mathrm{T}} = 0$$

$$(-y_t^2 - z_t^2)x + (x_t y_t)y + (x_t z_t)z + (x_0 y_t^2 + x_0 z_t^2 - y_0 x_t y_t - z_0 x_t z_t) = 0 \qquad (3.3.1.3)$$

The set of points that satisfy any two of the equations (3.3.1.1), (3.3.1.2) or (3.3.1.3) lie on the line describing the path of the ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$. See Example 3.3.1 in Appendix 1 for a worked example.

### 3.3.2 Labelling of the Rays.

Now that the general form of a ray in three-space has been identified, the form of a ray in the ray tracing problem can be stated. The set of *primary* rays passing through the view window is labelled according to an intersecting pixel $(i, j)$: e.g $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ becomes $\mathbf{r}^{i,j} = \mathbf{r}_0^{i,j} + t\mathbf{r}_t^{i,j}$. As such $\mathbf{r}_t^{i,j}$ is defined as the (normalised) direction vector from $\mathbf{r}_0^{i,j}$ to the view system coordinates $(u, v, n)$ corresponding to the pixel $(i, j)$. Recalling the mapping $V : (i, j) \rightarrow (u, v, n)$ (3.2.2), we can express $\mathbf{r}_t^{i,j}$ in the form:

$$\mathbf{r}_t^{i,j} = \frac{[V_u(i,j), V_v(i,j), V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0^{i,j}}{\left\| [V_u(i,j), V_v(i,j), V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0^{i,j} \right\|},$$

giving us the general ray:

$$\mathbf{r}^{i,j} = \mathbf{r}_0^{i,j} + t\frac{[V_u(i,j), V_v(i,j), V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0^{i,j}}{\left\| [V_u(i,j), V_v(i,j), V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0^{i,j} \right\|} \qquad (3.3.2)$$

The ray is, then, defined by its point of origin and the pixel it passes through, according to some mapping V. Using the equations of (3.2.2), the above equation (3.3.2) can be expanded to:

$$\mathbf{r}^{i,j} = \mathbf{r}_0^{i,j} + \frac{t\left[\left[a\left(\frac{2i+1}{g}-1\right), b\left(\frac{2j+1}{h}-1\right), k\right] - \mathbf{r}_0^{i,j}\right]}{\left\|\left[\left[a\left(\frac{2i+1}{g}-1\right), b\left(\frac{2j+1}{h}-1\right), k\right] - \mathbf{r}_0^{i,j}\right]\right\|}$$

Implicit definitions can be obtained in a manner similar to that demonstrated in the previous section.

Armed with the above definition of a ray in general terms, the discussion will now turn to specific ray equations and their applications.

## 3.4 Projection Methods.

The process of creating a two-dimensional image based on three-dimensional data is referred to as projection. The rays employed in a ray tracing algorithm describe the path of projection of a finite number of points in the problem world onto the view plane (the rays are referred to as *projectors*). By giving the sets of rays different paths a number of different types of projections may be obtained.

### 3.4.1 Parallel Projection.

Parallel projection refers to the use of projectors that are all parallel. As such, there is no single observer location when dealing with a parallel projection, rather we assume that all rays pass through the view window parallel to one another and, hence, all start at different points, defined for each by the pixel coordinates. The typical (orthographic)

parallel projection has rays that are in the same direction as the normal to the view window, that is $\mathbf{r}_t = [0, 0, -1]^{\mathrm{T}}$, which begin from points on the *uv* plane with *u* and *v* equivalent to those on the view window.

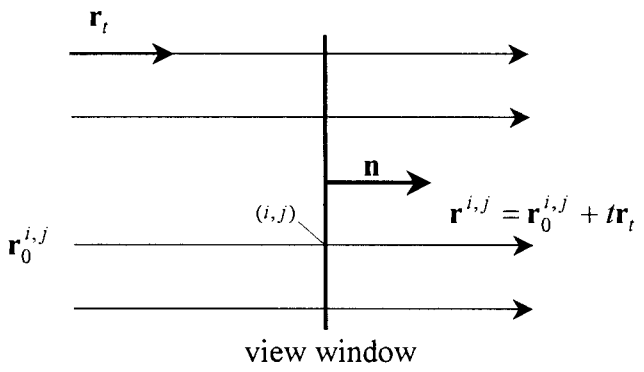



Figure 3.4.1.1. Rays and view window of orthographic projection.

The equation for these rays may be written explicitly:

$$\begin{aligned}
\mathbf{r}^{i,j} &= \mathbf{r}_0^{i,j} + t\frac{\mathbf{r}_t^{i,j}}{\left\|\mathbf{r}_t^{i,j}\right\|} \\
&= \mathbf{r}_0^{i,j} + t\frac{[0,0,-1]^{\mathrm{T}}}{1} \\
&= [a\left(\frac{2i+1}{g} - 1\right), b\left(\frac{2j+1}{h} - 1\right), 0]^{\mathrm{T}} + t[0,0,-1]^{\mathrm{T}} \\
\mathbf{r}^{i,j} &= [a\left(\frac{2i+1}{g} - 1\right), b\left(\frac{2j+1}{h} - 1\right), -t]^{\mathrm{T}}
\end{aligned}$$

The implicit definition is intuitive; the rays are described by the intersection of the planes:

$$u = a\left(\frac{2i+1}{g} - 1\right) \text{ and } v = b\left(\frac{2j+1}{h} - 1\right)$$

Parallel projections can be used to obtain side, front and back elevation views of an object / problem world as well as plan (top) views, through particular placements of the viewing system in that world.

An oblique projection can be performed by displacing the points of origin of rays defined under an orthogonal projection, whilst still passing the rays through their defining pixels. As such, the projectors remain parallel, but this direction of projection is now defined in the more general sense $\mathbf{r}_t = [V_u(i,j), V_v(i,j), V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0^{i,j}$ by any one $(i,j)$. Denoting the displacement by $[c,d,e]^{\mathrm{T}}$, the new vectors can be stated:

$$
\begin{aligned}
\mathbf{r}^{i,j} &= \mathbf{r}_0^{i,j} + t\frac{\mathbf{r}_t^{i,j}}{\left\|\mathbf{r}_t^{i,j}\right\|} \\
&= \mathbf{r}_0^{i,j} + t\frac{[V_u(i,j), V_v(i,j), V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0^{i,j}}{\left\|[V_u(i,j), V_v(i,j), V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0^{i,j}\right\|} \\
&= [a\left(\frac{2i+1}{g} - 1\right) + c, b\left(\frac{2j+1}{h} - 1\right) + d, e]^{\mathrm{T}} \\
&\quad + \frac{t\left[[a\left(\frac{2i+1}{g} - 1\right), b\left(\frac{2j+1}{h} - 1\right), k]^{\mathrm{T}} - [a\left(\frac{2i+1}{g} - 1\right) + c, b\left(\frac{2j+1}{h} - 1\right) + d, e]^{\mathrm{T}}\right]}{\left\|\left[[a\left(\frac{2i+1}{g} - 1\right), b\left(\frac{2j+1}{h} - 1\right), k]^{\mathrm{T}} - [a\left(\frac{2i+1}{g} - 1\right) + c, b\left(\frac{2j+1}{h} - 1\right) + d, e]^{\mathrm{T}}\right]\right\|} \\
\mathbf{r}^{i,j} &= [a\left(\frac{2i+1}{g} - 1\right) + c, b\left(\frac{2j+1}{h} - 1\right) + d, e]^{\mathrm{T}} + \frac{t}{\sqrt{c^2 + d^2 + (k-e)^2}}[-c, -d, k-e]^{\mathrm{T}}
\end{aligned}
$$

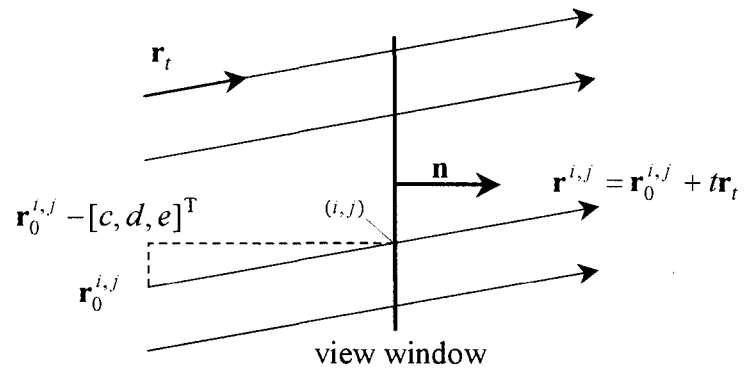



Figure 3.4.1.2. Rays and view window of oblique projection.

Note that the direction of these projectors is independent of the pixels through which they pass, and is simply a function of the displacement of the origins. Implicit formulations may be found by applying two of the equations (3.3.1.1), (3.3.1.2) or (3.3.1.3), as identified in section 3.3.1.

Traditional ray tracing does not usually feature parallel projections, simply because the typical goal has been to attain visual realism, which is not provided for by parallel methods. However parallel projection fits well within the framework of ray tracing and so has been included here. An example may be found in Appendix 1 (Example 3.4.1).

### 3.4.2 Perspective Projections.

The perspective projection is the typical means by which ray tracing is accomplished. Mimicking the process of vision (though only to a certain extent, as it is better thought of as mimicking a pinhole camera model (Glassner. 1989, p. 2)), only one ray origin is defined per image, named the centre of perspectivity, or the eye. Casting rays outward through each pixel on the view window from this point generates the perspective effect

by which we gauge relative distance from our eyes. Typically this initial point, $\mathbf{r}_0$, is placed at the origin $(0,0,0)$. The directions of the rays, then, are defined in terms of the pixels through which they each pass:

$$\begin{aligned}
\mathbf{r}^{i,j} &= \mathbf{r}_0^{i,j} + t\frac{\mathbf{r}_t^{i,j}}{\left\|\mathbf{r}_t^{i,j}\right\|} = \mathbf{r}_0 + t\frac{\mathbf{r}_t^{i,j}}{\left\|\mathbf{r}_t^{i,j}\right\|} \\
&= \mathbf{r}_0 + t\frac{[V_u(i,j),V_v(i,j),V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0}{\left\|[V_u(i,j),V_v(i,j),V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0\right\|} \\
&= [0,0,0]^{\mathrm{T}} \\
&\quad + \frac{t}{\sqrt{a^2\left(\frac{2i+1}{g}-1\right)^2 + b^2\left(\frac{2j+1}{h}-1\right)^2 + k^2}}\left[[a\left(\frac{2i+1}{g}-1\right), b\left(\frac{2j+1}{h}-1\right), k]^{\mathrm{T}} - [0,0,0]^{\mathrm{T}}\right] \\
\mathbf{r}^{i,j} &= \frac{t}{\sqrt{a^2\left(\frac{2i+1}{g}-1\right)^2 + b^2\left(\frac{2j+1}{h}-1\right)^2 + k^2}}[a\left(\frac{2i+1}{g}-1\right), b\left(\frac{2j+1}{h}-1\right), k]^{\mathrm{T}}
\end{aligned}$$

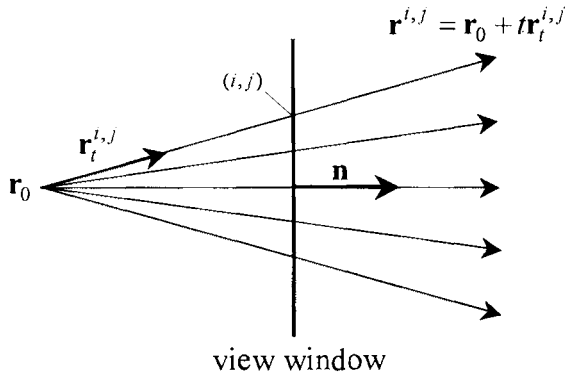



Figure 3.4.2.1. Rays and view window of perspective projection.

Placing $\mathbf{r}_0$ at a point other than the origin (for the viewing system with the view window centred at $(0,0,k)$) will generate various effects, which may be desirable in the presentation of the image.

Moving the centre of perspectivity to $(c,d,e)$, the following ray definitions apply:

$$\mathbf{r}^{i,j} = \mathbf{r}_0 + t\frac{\mathbf{r}_t^{i,j}}{\left\|\mathbf{r}_t^{i,j}\right\|}$$

$$= \mathbf{r}_0 + t\frac{[V_u(i,j),V_v(i,j),V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0}{\left\|[V_u(i,j),V_v(i,j),V_n(i,j)]^{\mathrm{T}} - \mathbf{r}_0\right\|}$$

$$\mathbf{r}^{i,j} = [c,d,e]^{\mathrm{T}}$$

$$+\frac{t}{\sqrt{\left[a\left(\frac{2i+1}{g}-1\right)-c\right]^2+\left[b\left(\frac{2j+1}{h}-1\right)-d\right]^2+(k-e)^2}}[a\left(\frac{2i+1}{g}-1\right)-c, b\left(\frac{2j+1}{h}-1\right)-d, k-e]^{\mathrm{T}}$$

Again, the implicit form of the above may be found using two of the equations (3.3.1.1), (3.3.1.2) or (3.3.1.3).

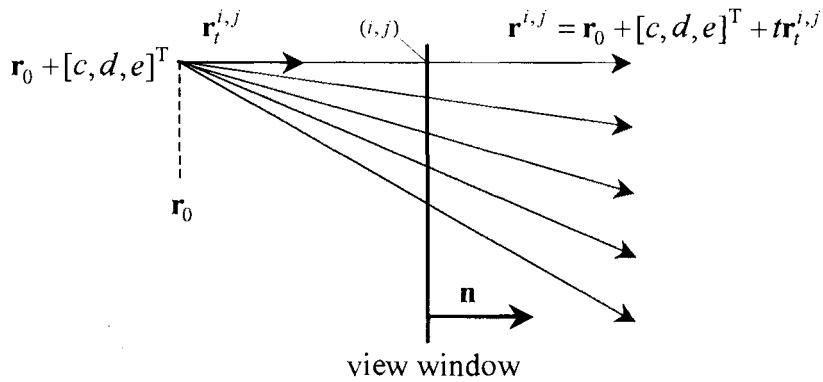



Figure 3.4.2.2. Rays and view window of alternative perspective projection.

Appendix 1 includes an example of a perspective projection (Example 3.4.2).

### 3.4.3 Other Projection Methods.

There are other methods of projection that may be adapted for use in computer graphics algorithms. The parallel and perspective methods are classed as *linear* projections (because of the behaviour of the rays). *Non-linear* projections may be used to mimic various camera lens types, such as *fisheye* and *omnimax*. Such variations can be achieved by making the parameter $t$ a function of pixel position $(i, j)$, defining ray origins according to $(i, j)$ but not ray direction (for example, rather than one point as the eye, a circle through which various rays pass is defined), or by defining curved paths for rays.

## 3.5 Ray – Surface Intersections.

Ray tracing is principally concerned with ray – surface intersections, as rays are cast out into the problem world to return information about the various surfaces that they strike. Given that there are two broad methods to define an object, through classification (implicitly) or through enumeration (explicitly), there are two broad types of ray - surface intersection. For an implicitly defined surface $F(u, v, w) = 0$, an explicitly defined ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ is used to replace the coordinates in $F$ with parametric expressions in $t$. Alternatively, an explicit surface $(u(\mu,\nu), v(\mu,\nu), w(\mu,\nu))$ is intersected with an implicitly defined ray (the intersection of two implicitly defined planes).

Intersection testing and computation for a general ray tracing scene is processor intensive. In any given scene, a ray needs to be tested against all surfaces for a potential intersection (the use of object hierarchies to restrict these searches is not within the scope and aim of this project). Once all objects have been tested, then and only then

may the first intersection of that 'backwards' ray with the problem world be identified. Mathematically, there is little one can do to enhance the process beyond placing a primitive bounding volume about a particular surface and testing for an intersection with that. If no intersection is returned then a potentially complex surface-ray equation is not required to be solved.

### 3.5.1 Rays defined in object spaces.

The ray-surface intersection computation generally takes place in object coordinate space. This may seem an odd choice after defining in Section 3.2.1 a matrix $\mathbf{M}_i = \mathbf{T}_i\mathbf{S}_i\mathbf{R}_i$ to transform every object or surface $i$ into some position and orientation in the problem world, but a ray is merely a line in three-space, and is transformed relatively simply from its vector form to any object or surface space by the inverse object transformation matrix:

$$\mathbf{M}_i^{-1} = \mathbf{R}_i^{-1}\mathbf{S}_i^{-1}\mathbf{T}_i^{-1} \qquad (3.5.1.1)$$

Furthermore, recall from Section 3.2.2 that the ray is defined in view system space, which is related to the problem world according to some transformation:

$$\mathbf{V} = \mathbf{T}_\mathbf{V}\mathbf{S}_\mathbf{V}\mathbf{R}_\mathbf{V}\,. \qquad (3.5.1.2)$$

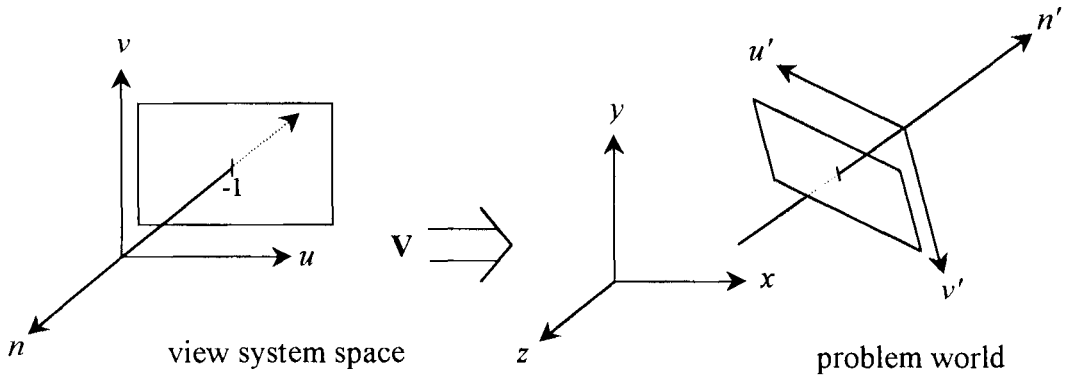



Figure 3.5.1. The placement of the view system in the problem world using transformation **V**.

To work in object space the ray first needs to be expressed in the problem world coordinates, and so is first transformed by $\mathbf{V} = \mathbf{T}_V \mathbf{S}_V \mathbf{R}_V$. It must be noted here that, because of the definition of the ray, caution must be exercised during this transformation. Whilst the origin component of the ray $\mathbf{r}_0$ is transformed, the direction component $\mathbf{r}_t$ should not be subjected to a translation transformation (the direction of a vector should be invariant under translation). Thus, for any ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$, using (3.5.1.1) and (3.5.1.2) yeilds:

$$\begin{aligned}\mathbf{r}_0^{(i)} &= \mathbf{R}_i^{-1}\mathbf{S}_i^{-1}\mathbf{T}_i^{-1}\mathbf{T}_V\mathbf{S}_V\mathbf{R}_V\mathbf{r}_0 \\ &= \mathbf{M}_i^{-1}\mathbf{V}\mathbf{r}_0\end{aligned}$$

and

$$\begin{aligned}\mathbf{r}_t^{(i)} &= \mathbf{R}_i^{-1}\mathbf{S}_i^{-1}\mathbf{S}_V\mathbf{R}_V\mathbf{r}_t \\ &= \mathbf{R}_i^{-1}\mathbf{S}_i^{-1}\mathbf{T}_i^{-1}\mathbf{T}_i\mathbf{T}_V^{-1}\mathbf{T}_V\mathbf{S}_V\mathbf{R}_V\mathbf{r}_t \\ &= \mathbf{M}_i^{-1}\mathbf{T}_i\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t\end{aligned}$$

so that

$$\mathbf{r}^{(i)} = \mathbf{M}_i^{-1}\mathbf{V}\mathbf{r}_0 + t\mathbf{M}_i^{-1}\mathbf{T}_i\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t \qquad \text{where } \mathbf{r}^{(i)} \text{ is the ray in object } i \text{ space.}$$

Example 3.5.1 in Appendix 1 illustrates a transformation matrix **V** for a particular positioning of the viewplane in the problem world.

### 3.5.2 Interpretation of intersections.

Using $\mathbf{r}^{(i)}$ explicitly to find any ray-surface intersections will provide solutions in terms of parameter $t$. Finding the intersection of an explicitly defined surface with an implicitly defined ray provides the parameter pair(s) $(\mu,\nu)_i$, which corresponds to a point in space on the ray (and surface). Given one coordinate value from such a point, the parameter $t$ can be computed. The following is an interpretation of the possible values of $t$.

| | |
|---|---|
| Positive solutions: | Intersections in the path of the ray. |
| Negative solutions: | Intersections 'behind' the origin of the ray. |
| $\min(t : t > 0)$: | The nearest intersection to the origin of the ray. |
| One solution: | The intersection occurs on a point of tangency. |
| Complex solution: | The ray does not intersect the given object. |

For a ray tracing application the solution corresponding to the least positive value of $t$ is of the most significance. It represents the first point on that surface which the ray would hit, assuming there are no objects in front of it. Testing for all objects we can state the first intersection of the ray with the problem world (at $t^*$) as follows:

$$t^* = \min(t_i : t_i > 0) \quad \text{for all objects } i.$$

Having found the nearest point of intersection (assuming that there was an intersection) of a ray with the elements in the problem world, the ray tracing procedure can begin the process of implementing optical effects and this will be detailed in Chapter 5. Chapter 4

examines many of the objects and surfaces that are frequently rendered using ray tracers, beginning with primitive objects and progressing through to parametric representations of surfaces.

## 4. Representations of Surfaces.

This section will elaborate on the general representation of surfaces implicitly ($F(x,y,z)=0$) and explicitly ($S(\mu,\nu)$) by considering a number of particular surface and object definitions. The following section re-iterates the requirements that a ray tracing algorithm requires of such objects, and provides a guide to the content of subsequent sections.

### 4.0 Guide to the Treatment of Surfaces.

#### 4.0.1 General Discussion of the Particular Surface.

Any special properties or usage of a particular type of surface will be noted.

#### 4.0.2 The Mathematical Representation of the Surface.

As stated before, there are two broad types of surface representation, implicit and explicit. Whilst the choice of representation is an arbitrary one from a mathematical perspective, when both are readily definable, a programmer would choose the representation more suitable for coding – with issues such as storage capacity, speed of use and accuracy of results (once coded) influencing their choice. For some surfaces though, the representation will be limited to either implicit or explicit only, as the other definition may be impractical, even mathematically.

Recall that an implicit surface is a definition by *classification*, the function $F(x,y,z)$ describing some region in space for which $F(x,y,z)=0$ is the boundary or surface; where $F(x,y,z)<0$ refers to the interior; and $F(x,y,z)>0$ the points $(x,y,z)$ exterior to this region ($F$ is also known as a *point-membership classification* function). For

example, the sphere $u^2+v^2+w^2=1$ is such an example as it can be represented by $u^2+v^2+w^2-1=0$. Implicit definitions are generally used in computer graphics applications to describe simpler, 'regular' surfaces.

Recall too that $S(\mu,\nu):(u(\mu,\nu),v(\mu,\nu),w(\mu,\nu))$ is an *enumerative* description, or explicit representation, of a surface. For example, the same sphere can be described parametrically:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \sqrt{1-\mu^2}\cos\nu \\ \mu \\ \sqrt{1-\mu^2}\sin\nu \end{bmatrix} \qquad \text{for } -1 \le \mu \le 1 \text{ and } 0 \le \nu \le 2\pi$$

Explicit definitions exist for many of the implicit functions that will be discussed below, and they (the former) will be included in the discussion as the explicit definition may on occasion require pre-computation of some quantities useful to a graphics program (for example, the surface normal).

### 4.0.3 The Representation of the Surface Normal.

The surface normal, the vector pointing in the direction in which the surface is considered to be 'facing', is readily defined for an implicit surface. Given $F(x,y,z)$, the normal vector to the surface defined by $F(x,y,z)=0$ is given by the vector:

$$\mathbf{n}^{\mathrm{T}} = [\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}]$$

The computation of a normal vector to a parametric surface is more involved. Such a vector is expressed as the cross product (section 2.2.6) of two vectors in the plane that is tangential to the surface at the point for which the normal is required. These vectors may be described using the first partial derivatives of the parametric function $S(\mu,\nu):(u(\mu,\nu),v(\mu,\nu),w(\mu,\nu))$ with respect to each parameter $(\mu,\nu)$. So the normal vector can be stated:

$$\mathbf{n}=[\frac{\partial u(\mu,\nu)}{\partial \mu},\frac{\partial v(\mu,\nu)}{\partial \mu},\frac{\partial w(\mu,\nu)}{\partial \mu}]^{\mathrm{T}}\times[\frac{\partial u(\mu,\nu)}{\partial \nu},\frac{\partial v(\mu,\nu)}{\partial \nu},\frac{\partial w(\mu,\nu)}{\partial \nu}]^{\mathrm{T}}$$

The normal vector is used extensively in the shading and illumination of surfaces in the problem world, and its usage will be discussed more thoroughly in Chapter 5, which deals specifically with that area of the image generation process.

A preliminary note is made here about the direction of the normal. The cross product is such that for any vectors $\mathbf{p}$ and $\mathbf{q}$, $\mathbf{p}\times\mathbf{q}=-\mathbf{q}\times\mathbf{p}$. Given that the choice of tangential plane vector order is an arbitrary one (it could be considered standard to use first the derivative with respect to the first parameter alphabetically speaking), the resulting normal vector really has two directions that may be considered, one opposite to the other. The choice of which to use is based on the use of the normal. For the purpose of shading and illumination, the normal of a surface at a point of intersection with a ray is said to be the vector through points with the same function classification (i.e. either $F<0$ or $F>0$) as the ray had 'just' before it intersected. In other words if the ray came from outside of the object, then the normal used in shading computations points to the outside or away from the object. Conversely, a ray that is passing through the

interior of an object to make a particular intersection defines a normal pointing inwards. A simple algorithm exists, which can be used generally, to identify the normal to be used:

*if* $\mathbf{r}_t \bullet \mathbf{n} > 0$ *then;*

$\mathbf{n} = (-1) * \mathbf{n}$ *: reverse the direction of the normal* (fig. 4.0.3 b).

***else*** *the direction of the normal is correct* (fig. 4.0.3 a).

where $\mathbf{r}_t$ is the direction of the ray **r** and **n** is the normal.

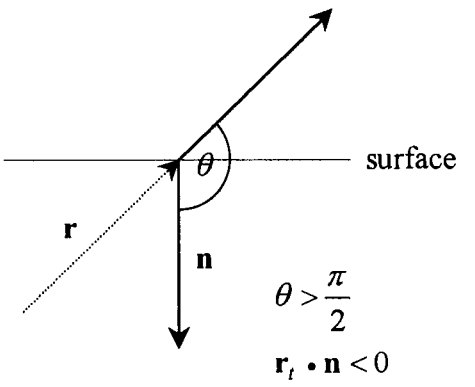



Figure 4.0.3 a.

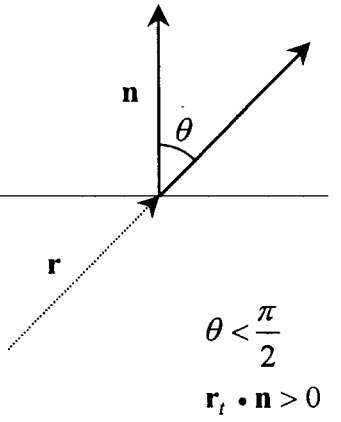



Figure 4.0.3 b.

4.0.4 The Intersection of a Ray with the Surface.

The discussion of each surface will provide a simplification of the equations used to intersect a ray with that surface. Recall that the general form of a ray intersection with an implicit surface is as follows.

For an implicit surface: $F(u, v, w) = 0$,

and the explicitly defined ray: $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t \Rightarrow \begin{cases} u = u_0 + tu_t \\ v = v_0 + tv_t \\ w = w_0 + tw_t \end{cases}$

then the intersection is given: $F(u_0 + tu_t, v_0 + tv_t, w_0 + tw_t) = 0$

which we abbreviate to $F^{\mathbf{r}}(t) = 0$, where $F^{\mathbf{r}}(t)$ is a function of coordinate values along ray **r**, which vary with $t$. The ray equation can be solved for $t$, thus providing the position of any intersections.

When rays are cast at parametric surfaces, the intersections are obtained as follows.

Treating the ray implicitly as the intersection of two planes

$\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ becomes $\begin{aligned} A_1u + B_1v + C_1w + D_1 &= 0 = \mathrm{F}_1(u,v,w) \\ A_2u + B_2v + C_2w + D_2 &= 0 = \mathrm{F}_2(u,v,w) \end{aligned}$

using methods detailed in section 3.3.1.

The intersection with the surface $S(\mu,\nu)$ defined by $(u(\mu,\nu), v(\mu,\nu), w(\mu,\nu))$ may be written:

$$\begin{aligned} \mathrm{F}_1^S(\mu,\nu) &= A_1u(\mu,\nu) + B_1v(\mu,\nu) + C_1w(\mu,\nu) + D_1 = 0 \\ \mathrm{F}_2^S(\mu,\nu) &= A_2u(\mu,\nu) + B_2v(\mu,\nu) + C_2w(\mu,\nu) + D_2 = 0 \end{aligned}$$

where each $F^S(\mu,\nu)$ is a function of the surface parameters $\mu$ and $\nu$. These equations may be solved to provide the parameter pair $(\mu,\nu)$ of any intersections of the ray with the surface.

Note that the general form of a ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ will be used in most instances, in spite of specific forms already defined for particular projection methods (section 3.4), which may have common usage. The reasoning behind this is that not all rays that are tested in a ray tracing algorithm originate at the eye and pass through the view window. For example, secondary rays cast from points of intersection are based on the direction of the surface normal and the intersecting ray.

### 4.0.5 Behaviour of Surfaces under Transformations.

Whilst in many cases it will be possible to apply some transformation to a given surface directly, this project will not consider such treatments for any surfaces to be ray traced. Rather, the intended transformation matrix is inverted and applied to any rays against which this object is to be tested, and the computations are performed in the object space. It is convenient to test for intersections in this manner because the rays are more simply transformed than the surfaces. Still, the principle of the above is the same as that of transforming a surface and comparing it to a ray that is in its original form. This would provide an avenue for predicting and checking results.

## 4.1 Quadric Surfaces

Many so called *geometric primitives*, as referred to in the computer graphics literature, fall into the family of quadric surfaces. Planes, spheres, cylinders and cones, amongst others, are quadric surfaces. The general implicit equation for such a surface is:

$$F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k = 0$$

This can be expressed in the following matrix form:

$$F(x, y, z) = \mathbf{P}^{\mathrm{T}}\mathbf{Q}\mathbf{P} = 0$$

$$\text{where: } \mathbf{P} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ and } \mathbf{Q} = \begin{bmatrix} a & d & f & g \\ d & b & e & h \\ f & e & c & j \\ g & h & j & k \end{bmatrix}$$

This family of surfaces possesses many properties that make the surfaces attractive for use in graphics applications (Foley, et al. 1994, pg. 357). The two main properties that make quadrics particularly popular in ray tracing are the ease of computation of the surface normal, and the fact that an intersection with a ray can be found by simply solving a quadratic equation.

Given a ray of the form $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$, the intersection with the quadric $F$ can be written:

$$F(x_0 + tx_t, y_0 + ty_t, z_0 + tz_t) = \mathbf{r}^{\mathrm{T}}\mathbf{Q}\mathbf{r} = 0$$

$$\begin{aligned} \mathbf{r}^{\mathrm{T}}\mathbf{Q}\mathbf{r} &= [\mathbf{r}_0 + t\mathbf{r}_t]^{\mathrm{T}}\mathbf{Q}[\mathbf{r}_0 + t\mathbf{r}_t] \\ &= \mathbf{r}_0^{\mathrm{T}}\mathbf{Q}\mathbf{r}_0 + t\mathbf{r}_0^{\mathrm{T}}\mathbf{Q}\mathbf{r}_t + t\mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_0 + t^2\mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_t \\ &= \mathbf{r}_0^{\mathrm{T}}\mathbf{Q}\mathbf{r}_0 + 2t\mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_0 + t^2\mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_t \quad \text{as } \mathbf{r}_0^{\mathrm{T}}\mathbf{Q}\mathbf{r}_t = \mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_0 \end{aligned}$$

so that $\mathbf{r}_0^{\mathrm{T}}\mathbf{Q}\mathbf{r}_0 + 2t\mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_0 + t^2\mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_t = 0$

This equation is a quadratic in $t$, and hence can be solved using the quadratic formula to give:

$$t = \frac{-\mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_0 \pm \sqrt{(\mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_0)^2 - (\mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_t)(\mathbf{r}_0^{\mathrm{T}}\mathbf{Q}\mathbf{r}_0)}}{\mathbf{r}_t^{\mathrm{T}}\mathbf{Q}\mathbf{r}_t}$$

The normal to a quadric surface is found through taking the first derivative of $F$ with respect to each variable:

$$\mathbf{n} = [\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}]^{\mathrm{T}}$$

$$\text{so if } F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k,$$

$$\text{then } \mathbf{n} = \begin{bmatrix} 2ax + 2dy + 2fz + 2g \\ 2by + 2dx + 2ez + 2h \\ 2cz + 2ey + 2fx + 2j \end{bmatrix}$$

To conclude the summary of the quadric surfaces, it is noted here that a quadric surface can be transformed by the composite transformation (4 × 4) matrix **M** according to the following (Foley, et al. 1994, pg 357):

$$F_{\mathbf{M}}'(x, y, z) = \mathbf{P}^{\mathrm{T}}(\mathbf{M}^{-1})^{\mathrm{T}}\mathbf{Q}\mathbf{M}^{-1}\mathbf{P} = 0$$

This property would facilitate placing quadric surfaces directly into the problem world, and even into view space. However, as stated before, the use of transformations will be kept to the manipulation of the ray equations.

### 4.1.1 Planes

The implicit definition of a plane can be written using the quadratic form:

$$F(x, y, z) = \mathbf{P}^{\mathrm{T}}\mathbf{Q}\mathbf{P} = 0, \text{ where } \mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & g \\ 0 & 0 & 0 & h \\ 0 & 0 & 0 & j \\ g & h & j & k \end{bmatrix} \text{ and } \mathbf{P} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$2gx + 2hy + 2jz + k = 0$$

The normal to this plane is given by $\mathbf{n} = [2g, 2h, 2j]^{\mathrm{T}}$. The unit normal is given by

$$\hat{\mathbf{n}} = \frac{1}{\sqrt{g^2 + h^2 + j^2}} [g, h, j]^{\mathrm{T}}$$

Should a parametric representation of the above plane be sought, one can refer to the following. For any point $(x, y, z)$ on the plane $2gx + 2hy + 2jz + k = 0$, a position vector $\mathbf{P} = [x, y, z]^{\mathrm{T}}$ can be expressed such that:

$$\mathbf{P} = \mathbf{P}_0 + \mu \mathbf{P}_\mu + \nu \mathbf{P}_\nu$$

$\mathbf{P}_0$ is some arbitrary point on the plane from which all $\mathbf{P}$ can be described by moving in the directions of $\mathbf{P}_\mu$ and $\mathbf{P}_\nu$. $\mathbf{P}_0$ may be defined as a point on one of the principal axes, by setting two variables to zero and solving for the remaining one: e.g., on the $z$ axis $\mathbf{P}_0 = [0, 0, -\frac{k}{2j}]^{\mathrm{T}}$. $\mathbf{P}_\mu$ and $\mathbf{P}_\nu$ may be any two vectors orthogonal to $\mathbf{n} = [2g, 2h, 2j]^{\mathrm{T}}$, for which $\mathbf{P}_\mu \neq k\mathbf{P}_\nu$. For example, $\mathbf{P}_\mu$ and $\mathbf{P}_\nu$ could be chosen from the three vectors $[h, -g, 0]^{\mathrm{T}}$, $[-j, 0, g]^{\mathrm{T}}$, $[0, j, -h]^{\mathrm{T}}$.

To illustrate, a parametric representation for the plane $2x + 4y - 3z + 2 = 0$ is

$$\mathbf{P} = [0, 0, \frac{2}{3}]^{\mathrm{T}} + \mu[2, -1, 0]^{\mathrm{T}} + \nu[\frac{3}{2}, 0, 1]^{\mathrm{T}}, \text{ for } \mu, \nu \in \mathrm{R}.$$

The solution to a ray-plane intersection is obtained without the need for the quadratic formula. The implicit form of the plane is:

$$2gx + 2hy + 2jz + k = 0$$

When we substitute the ray equation into this we get:

$$2g(x_0 + tx_t) + 2h(y_0 + ty_t) + 2j(z_0 + tz_t) + k = 0$$

$$2(gx_0 + hy_0 + jz_0) + k = -2t(gx_t + hy_t + jz_t)$$

$$t = -\frac{gx_0 + hy_0 + jz_0 + k/2}{gx_t + hy_t + jz_t}$$

$$\text{i.e. } t = \frac{-(\mathbf{n} \bullet \mathbf{r}_0 + k/2)}{\mathbf{n} \bullet \mathbf{r}_t}$$

There is no solution if the ray is parallel to the plane (and hence orthogonal to the plane's normal), as the denominator in the expression for $t$ is zero.

### 4.1.2 Faceted Surfaces.

A method of surface construction is to specify a set of bounded planes or *facets* that together describe the desired surface. For example, the surface of a cube could be described by a set of six facets. The positioning of facets relative to each other is best accomplished by identifying the vertices of the surface, and then noting the vertices common to each facet. These facets, each represented by a set of vertices, may be treated as bounded planes and used in image generation techniques.

It is convenient to subdivide each facet into triangles to simplify its representation mathematically. A triangle can be described parametrically much in the same way a plane was in the previous section, except that constraints are applied to the parameters.

Consider the triangle defined by position vectors $\mathbf{P}_0$, $\mathbf{P}_1$ and $\mathbf{P}_2$ ($[x_0, y_0, z_0]^T$, $[x_1, y_1, z_1]^T$, $[x_2, y_2, z_2]^T$ respectively). Each of these vectors is placed with its tail at the origin of the local coordinate space, and hence 'points' to a single

coordinate (a corner of the triangle). For any point $(x, y, z)$ on this triangle, a position vector $\mathbf{P} = [x, y, z]^T$ may be expressed:

$$\mathbf{P} = \mathbf{P}_0 + \mu\mathbf{P}_\mu + \nu\mathbf{P}_\nu$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + \mu \begin{bmatrix} x_\mu \\ y_\mu \\ z_\mu \end{bmatrix} + \nu \begin{bmatrix} x_\nu \\ y_\nu \\ z_\nu \end{bmatrix}$$

where $\mathbf{P}_\mu = \mathbf{P}_1 - \mathbf{P}_0$, $\mathbf{P}_\nu = \mathbf{P}_2 - \mathbf{P}_0$, and $\mu \geq 0, \nu \geq 0, \mu + \nu \leq 1$. A normal to this triangular facet is $\mathbf{n} = (\mathbf{P}_1 - \mathbf{P}_0) \times (\mathbf{P}_2 - \mathbf{P}_0)$.

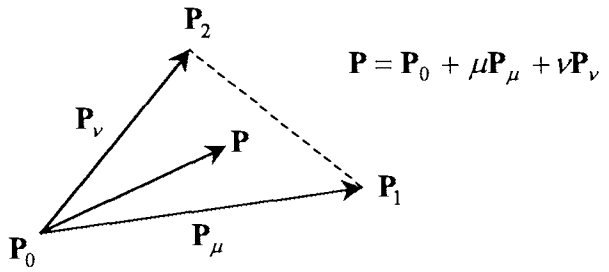



Figure 4.1.2.1. Representation of a Triangular Facet. Note that for simplicity, the position vectors of the corners are not drawn.

When testing for an intersection of a ray with a facet, the ray can be implicitized into the intersection of two planes (as discussed in section 3.3.1) and then the parametric facet coordinates substituted into the two equations. A solution (if any) would be in terms of $\mu$ and $\nu$, which could then be directly compared to the bounding conditions stated above to see whether the ray intersected the facet.

Alternatively, the facet plane could be represented implicitly as $Ax + By + Cz + D = 0$, where $A$, $B$ and $C$, respectively, are the $x$, $y$ and $z$ components of $\mathbf{n}$ and $D$ is found by $D = -Ax - By - Cz$ and using the coordinates of any one of the facet's vertices. The

intersection would be found with an explicitly defined ray, in terms of the ray parameter $t$, and hence the intersection coordinates would have to be computed and substituted into the parametric equation for the facet to determine whether the intersection was within the bounds of the facet. Examples 4.1.2, 4.1.2.1 and 4.1.2.2 in Appendix 1 provide worked examples of the intersection of a ray with a facet.

In some instances, the use of quadrilaterals may be considered as an alternative to triangles. A point on the quadrilateral $\mathbf{P_0P_1P_2P_3}$, labelled either clockwise or anticlockwise, with vertices $(x_0, y_0, z_0)$, $(x_1, y_1, z_1)$, $(x_2, y_2, z_2)$, $(x_3, y_3, z_3)$, has position vector $\mathbf{P} = [x, y, z]^{\mathrm{T}}$:

$$\mathbf{P} = \mathbf{P}_0 + \mu[(1-\nu)(\mathbf{P}_1 - \mathbf{P}_0) + \nu(\mathbf{P}_2 - \mathbf{P}_3)] + \nu(\mathbf{P}_3 - \mathbf{P}_0)$$

where $0 \le \mu \le 1$, and $0 \le \nu \le 1$ (Burger & Gillies, 1989. pg. 412).

The normal to this surface is $\mathbf{n} = (\mathbf{P}_1 - \mathbf{P}_0) \times (\mathbf{P}_2 - \mathbf{P}_0)$ or the cross product of vectors joining any two of the vertices. Computations to find intersections with rays are carried out as with triangles, although are more complicated because a mixed term with the product of the facet parameters $\mu$ and $\nu$ is involved.

### 4.1.3 Spheres.

The equation of a sphere with radius 1 and centred at the origin may be written in quadratic form:

$$F(x, y, z) = \mathbf{P}^{\mathrm{T}}\mathbf{Q}\mathbf{P} = 0$$

where $\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$ and $\mathbf{P} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

which is equivalent to

$$x^2 + y^2 + z^2 - 1 = 0$$

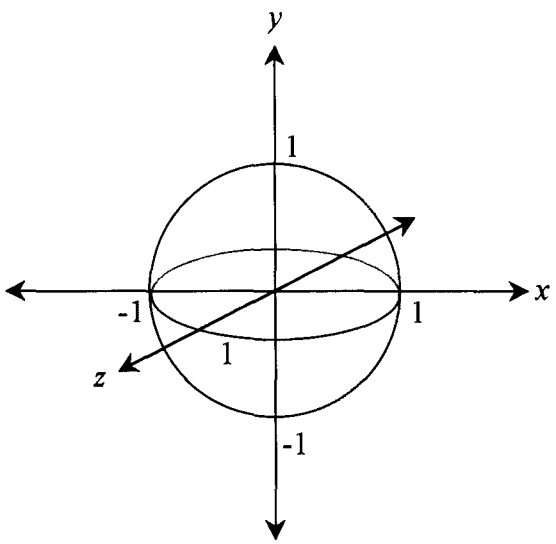



Figure 4.1.3. The sphere $x^2 + y^2 + z^2 - 1 = 0$.

The normal to this surface, given by $\mathbf{n} = [\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}]^{\mathrm{T}}$, is $\mathbf{n} = [2x, 2y, 2z]^{\mathrm{T}}$. The unit normal is $\hat{\mathbf{n}} = [x, y, z]^{\mathrm{T}}$, as expected for a point on a unit sphere (the normal is the vector from the origin to that point).

The intersection of the sphere $x^2 + y^2 + z^2 - 1 = 0$ with the ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ (already assumed to have already been transformed by the inverse of the sphere's own transformation matrix $\mathbf{M}_{\text{sphere}}$) may be found by solving the quadratic equation:

$$(x_0 + tx_t)^2 + (y_0 + ty_t)^2 + (z_0 + tz_t)^2 - 1 = 0$$
$$t^2(x_t^2 + y_t^2 + z_t^2) + 2t(x_0x_t + y_0y_t + z_0z_t) + (x_0^2 + y_0^2 + z_0^2) - 1 = 0$$

Two solutions for $t$ would indicate that the ray passes through the sphere (see Example 4.1.3, Appendix 1). The ray is said to touch the sphere tangentially when $t$ has only one solution, and typically this occurrence is ignored by the ray tracing algorithm. No solutions are returned when the ray does not intersect the surface.

Spheres are members of the subset of quadrics named the *quadrics of revolution*. The general form of the equations for these surfaces is $ax^2 + by^2 + cz^2 + 2jz + k = 0$, for $a = b$. The cross section made by the plane $z = 0$ is a circle of the form $x^2 + y^2 + k/a = 0$. Cylinders, cones and paraboloids are other examples of this type of surface. The following sections refer to examples where $a = b = 1$, on the understanding that a scaling matrix **S** (section 2.4.2) could be applied subsequent to their definition to provide shapes where $a$ and $b$ have values other than 1.

### 4.1.4 Cylinders.

The equation of an infinite cylinder with radius 1 and centred about the z axis may be written in general quadratic form:

$$F(x, y, z) = \mathbf{P}^{\mathrm{T}}\mathbf{Q}\mathbf{P} = 0$$

where $\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$ and $\mathbf{P} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

which is equivalent to

$$x^2 + y^2 - 1 = 0 \tag{4.1.4}$$

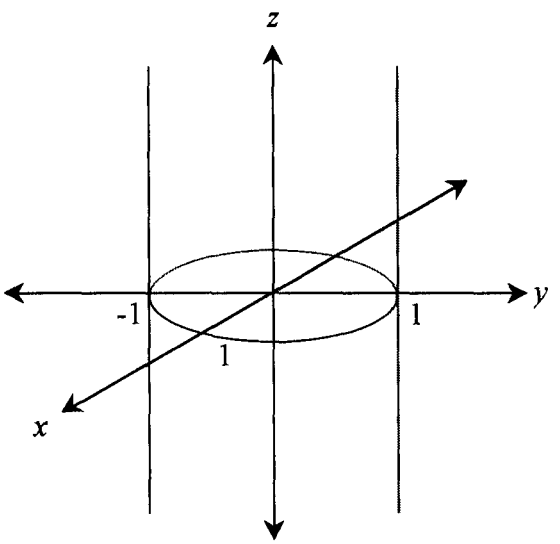



Figure 4.1.4. The cylinder $x^2 + y^2 - 1 = 0$.

The normal to this surface, given by $\mathbf{n} = [\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}]^{\mathrm{T}}$, is $\mathbf{n} = [2x, 2y, 0]^{\mathrm{T}}$. The unit normal, $\hat{\mathbf{n}} = [x, y, 0]^{\mathrm{T}}$, is as expected for a point on a the surface of an unbounded cylinder (the normal is the vector from the corresponding point on the z axis, with position vector $[0,0,z]^{\mathrm{T}}$, to the point indicated by $[x, y, z]^{\mathrm{T}}$).

The intersection of a cylinder $x^2 + y^2 - 1 = 0$ with a parametrically defined ray can be found via substitution:

$$(x_0 + tx_t)^2 + (y_0 + ty_t)^2 - 1 = 0$$
$$t^2(x_t^2 + y_t^2) + 2t(x_0x_t + y_0y_t) + (x_0^2 + y_0^2) - 1 = 0$$

This equation can be solved using the quadratic formula. Solutions for $t$ can be interpreted in a similar manner to the case for the sphere: two solutions indicating a ray passing through the cylinder, one solution for a ray that tangentially touches the surface and no solutions for a ray that has no intersection with the cylinder.

It would be unusual to use many infinite cylinders in the construction of the problem world, rather each cylinder would be of a certain length and perhaps *capped* with two circular discs at the ends. We may define a finite cylinder, by the following:

$$x^2 + y^2 - 1 = 0$$
$$0 \le z \le 1$$

The choice to limit $z$ in this manner is an arbitrary one, and relies on the scaling component in the transformation matrix $\mathbf{M}_t$ to achieve varying degrees of width and height (which would be reflected in the components of $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ before the computation of the intersection). Under these limits, the solutions providing the intersections of the cylinder with a ray may take on new meaning. Two intersections with $z$ components out of the domain $0 \le z \le 1$ would mean that the ray does not come into contact with the finite cylinder, whilst two intersections with the infinite cylinder,

one of which is in the stated domain, would indicate that the ray enters or exits through one of the capping surfaces. Example 4.1.4 illustrates this numerically (Appendix 1).

### 4.1.5 Cones.

The equation of an infinite cone with semi-vertical angle $\pi/4$ is:

$$F(x, y, z) = \mathbf{P}^{\mathrm{T}}\mathbf{Q}\mathbf{P} = 0$$

where $$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{P} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

which is equivalent to $$x^2 + y^2 - z^2 = 0$$

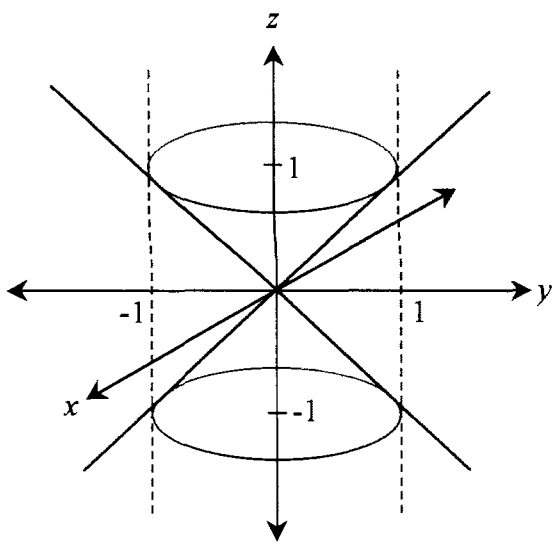



Figure 4.1.5.1. The cone $x^2 + y^2 - z^2 = 0$.

The normal to this surface, given by $\mathbf{n} = [\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}]^{\mathrm{T}}$, is $\mathbf{n} = [2x, 2y, -2z]^{\mathrm{T}}$. The unit normal is $\hat{\mathbf{n}} = \frac{1}{\sqrt{x^2 + y^2 + z^2}}[x, y, -z]^{\mathrm{T}}$.

The coordinates of a parametrically defined ray may be substituted into the equation of a cone to provide its points of intersection (if any) with that surface (see Example 4.1.5 in Appendix 1):

using $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$, the expression for the cone $x^2 + y^2 - z^2 = 0$ becomes:

$$(x_0 + tx_t)^2 + (y_0 + ty_t)^2 - (z_0 + tz_t)^2 = 0$$

$$t^2(x_t^2 + y_t^2 - z_t^2) + 2t(x_0x_t + y_0y_t - z_0z_t) + (x_0^2 + y_0^2 - z_0^2) = 0$$

As with cylinders, an infinite cone may be unsuitable in the construction of a problem world, and so it may be bounded as required. Bounding the cone at $z = 0$ provides the familiar pointed end (at which the normal vector **n** is undefined), and at $z = \pm 1$ a 'bottom' to the cone. Choosing to restrict both $z$ boundaries to values higher (or lower) than 0 provides a truncated cone, without the pointed end, as illustrated below.

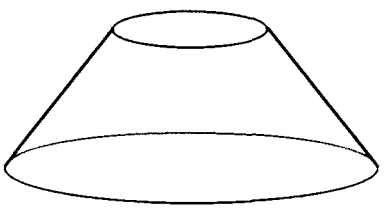


Figure 4.1.5.2. A truncated cone (or 'bucket').

### 4.1.6 Paraboloids and Hyperboloids

Consider the paraboloid:

$$x^2 + y^2 - z = 0$$

which may be written in the quadratic form:

$$F(x,y,z)=\mathbf{P}^{\mathrm{T}}\mathbf{Q}\mathbf{P}=0,\ \text{where}\ \mathbf{Q}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&-\frac{1}{2}\\0&0&-\frac{1}{2}&0\end{bmatrix}\ \text{and}\ \mathbf{P}=\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

The normal to this paraboloid is $\mathbf{n}=[2x,2y,-1]^{\mathrm{T}}$. The intersection of this surface with a ray is found as follows:

$$\begin{aligned}x^2+y^2-z&=0\\(x_0+tx_t)^2+(y_0+ty_t)^2-(z_0+tz_t)&=0\\t^2(x_t^2+y_t^2)+t(2x_0x_t+2y_0y_t-z_t)+(x_0^2+y_0^2-z_0)&=0\end{aligned}$$

A circular hyperboloid may be expressed in quadratic form; such as

$$F(x,y,z)=\mathbf{P}^{\mathrm{T}}\mathbf{Q}\mathbf{P}=0$$

where $\mathbf{Q}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&-1&0\\0&0&0&\mp1\end{bmatrix}$ and $\mathbf{P}=\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$

which is equivalent to

$$x^2+y^2-z^2\mp1=0 \qquad (4.1.6.1)$$

A hyperboloid of one sheet, without discontinuities, is represented by the equation

$$x^2+y^2-z^2-1=0 \qquad (4.1.6.2)$$

whereas $x^2+y^2-z^2+1=0$ refers to a discontinuous hyperboloid, which is not defined between the planes $z=1$ and $z=-1$. The normal to the surface defined by (4.1.6.1) is $\mathbf{n}=[2x,2y,-2z]^{\mathrm{T}}$.

The intersection of this surface with a ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ is found as follows:

$$x^2 + y^2 + z \mp 1 = 0$$
$$(x_0 + tx_t)^2 + (y_0 + ty_t)^2 + (z_0 + tz_t) \mp 1 = 0$$
$$t^2(x_t^2 + y_t^2) + t(2x_0x_t + 2y_0y_t + z_t) + (x_0^2 + y_0^2 + z_0) \mp 1 = 0$$

$t^2(x_t^2 + y_t^2) + t(2x_0x_t + 2y_0y_t + z_t) + (x_0^2 + y_0^2 + z_0) - 1 = 0$, for one sheet.
$t^2(x_t^2 + y_t^2) + t(2x_0x_t + 2y_0y_t + z_t) + (x_0^2 + y_0^2 + z_0) + 1 = 0$, for two sheets.

A paraboloid or hyperboloid may be bounded in much the same way as described for a cone or cylinder.

## 4.2 Tori.

A torus, or *doughnut ring*, is generated by rotating a circle about a line which lies in the plane of the circle, but does not intersect it. The implicit equation for a torus is based on the radius $r$ of the circle rotated and on the radius $s$ of the revolution (which is taken to be the distance from the centre of the circle to the axis of revolution). By describing a second circle that is the mirror image of the first in the given line a cross section of the torus is defined. For example, if the two circles lie on the $xz$ plane, they are described by: $(x-s)^2 + z^2 = r^2$ and $(x+s)^2 + z^2 = r^2$. The equation for the cross section is obtained by multiplying the two equations together (once each is rearranged in the form $F(x,z) = 0$):

$$((x-s)^2 + z^2 - r^2)((x+s)^2 + z^2 - r^2) = 0$$

which expands to:

$$(x^4 - 2s^2x^2 + s^4) + (z^2 - r^2)(2x^2 + 2s^2) + (z^4 - 2r^2z^2 + r^4) = 0 \qquad (4.2.1)$$

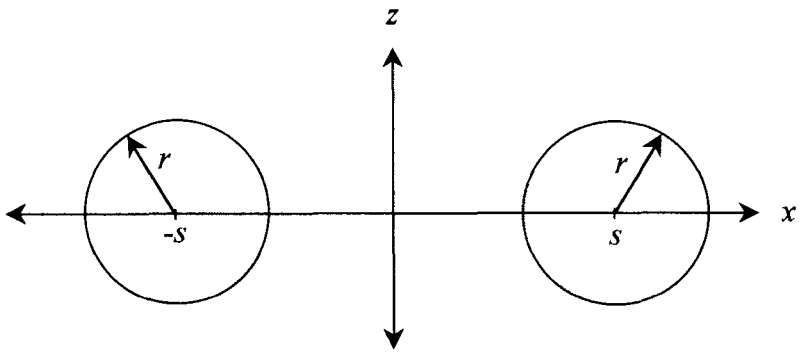



Figure 4.2. Cross section of a torus.

Equation (4.2.1) implicitly describes the points on the two circles on the *xz* plane. The torus is obtained by the rotating of the circles about the *z* axis. The equation of the torus is found by replacing $x^2$ in (4.2.1) with $(x^2 + y^2)$, yielding:

$$(x^2 + y^2)^2 - 2s^2(x^2 + y^2) + s^4 + 2(z^2 - r^2)(x^2 + y^2 + s^2) + (z^4 - 2r^2z^2 + r^4) = 0$$

which can be simplified:

$$(x^2 + y^2 + z^2)^2 - 2(r^2 + s^2)(x^2 + y^2 + z^2) + 4s^2z^2 + (r^2 - s^2)^2 = 0 \qquad (4.2.2)$$

Equation (4.2.2) represents a torus with radius of revolution *s* (measured to the centre of the band) and of band radius *r*. As one would expect, the equation is a quartic, indicating that it can have up to four intersections with any one line. Intersecting the ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ with the torus provides an equation in terms of parameter *t* of the form:

$$F^{\mathbf{r}}(t) = at^4 + bt^3 + ct^2 + dt + e = 0,$$

The normal to the surface at any point may be found by partial differentiation of the implicit formula (4.2.2):

$$\mathbf{n} = \begin{bmatrix} 4x(x^2 + y^2 + z^2 - r^2 - s^2) \\ 4y(x^2 + y^2 + z^2 - r^2 - s^2) \\ 4z(x^2 + y^2 + z^2 - r^2 + s^2) \end{bmatrix}$$

See Appendix 1 for an example of a ray-torus intersection problem with surface normal computation (Example 4.2).

## 4.3 Composite Surfaces of Revolution.

The above method for generating the torus by the revolution of the cross sections about an axis can be applied generally to other curves defined in the same plane (the *uw* plane). Suppose these curves are labelled $1, 2, ..., j$. By rearranging the equation of each in the form $F_i(u, w) = 0$, we can form the equation implicitly representing all points on the composite cross section:

$$F(u, w) = \prod_{i=1}^{j} F_i(u, w) = 0$$

The representation for the surface of revolution (the revolution of the composite boundaries about the *w* axis) is then given by substitution of $\sqrt{u^2 + v^2}$ for $u$:

$$\prod_{i=1}^{j} F_i(\sqrt{u^2 + v^2}, w) = 0$$

For example, consider the construction of an infinite cylinder of radius 1, with its axis along the *z* axis. On the *xz* plane the two edges of this can be described by the lines

$x=-1$ and $x=1$, or equivalently: $x+1=0$ and $x-1=0$. The composite function is then:

$$F(x,z)=\prod_{i=1}^{2}F_i(x,z)=0$$

$$(x+1)(x-1)=0$$
$$x^2-1=0$$

The cylinder is defined:

$$\prod_{i=1}^{2}F_i(\sqrt{x^2+y^2},z)=0$$

$$(\sqrt{x^2+y^2})^2-1=0$$
$$x^2+y^2-1=0$$

which agrees with equation (4.1.4).

Similarly consider the construction of a single sheet hyperboloid. On the $xz$ plane the cross section can be described by $x^2=z^2+1$, and by $-x^2=z^2+1$. The composite function is then:

$$(x-\sqrt{z^2-1})(x+\sqrt{z^2+1})=0$$
$$x^2-(z^2+1)=0$$

The hyperboloid is defined:

$$\prod_{i=1}^{2}F_i(\sqrt{x^2+y^2},z)=0$$

$$(\sqrt{x^2+y^2})^2-(z^2+1)=0$$
$$x^2+y^2-z^2-1=0$$

which agrees with (4.1.6).

These surfaces may be combined once they have been defined in the above form, or alternatively the same effect can be achieved from the combined set of component curves on the *uw* plane. Consider the following bounded cylinder of radius one, capped by a paraboloid touching the origin and by the plane $z = -1$.

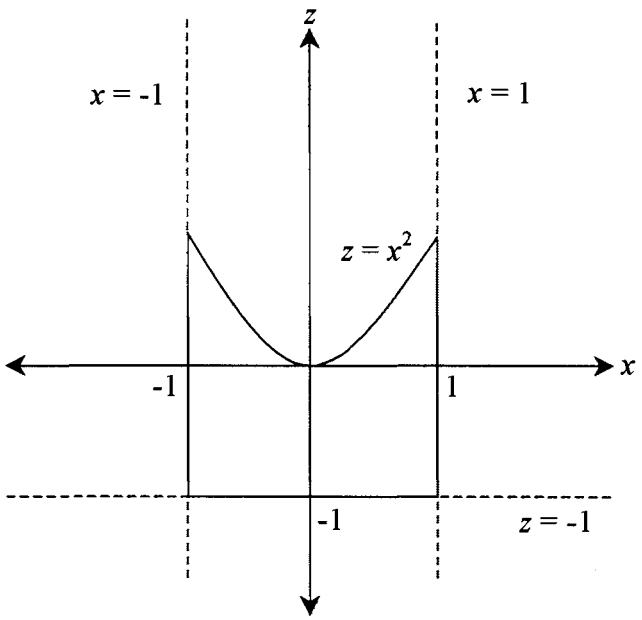



Figure 4.3.1. Composite Surface of Revolution.

The composite surface consists of the paraboloid $x^2 + y^2 - z = 0$; the cylinder $x^2 + y^2 - 1 = 0$; and the plane $z + 1 = 0$. The expression for the surface as a whole can be written:

$$(x^2 + y^2 - z)(x^2 + y^2 - 1)(z + 1) = 0 \qquad (4.3.1)$$

The same result is achieved when the curves $z = x^2$, $z = -1$, $x = 1$, and $x = -1$ are combined, and then $\sqrt{x^2 + y^2}$ substituted for $x$.

Substituting the ray coordinate equations into (4.3.1) yields an equation from which parameter values of $t$ can be obtained for points of intersection. The surface normal may be obtained by partial differentiation.

When generating surfaces with the above method, it is important to note the practical implications of the required computations. The equations generated for such a surface can be of too high an order to allow straightforward computation of intersections with a ray (bearing in mind that in all likelihood, many rays will need to be tested for intersections with the surface) hence making ray tracing computationally expensive. Should the complexity of the calculations become too great, one can limit the definition of the composite surface, and treat it as separate surfaces of lower orders.

A bounding volume may be defined in addition to, or as an alternative to, the above strategy, against which rays are tested prior to intersection with the full surface. For example, the infinite cylinder used in the above bounded the entire surface. Rays that did not intersect with the cylinder would certainly not intersect the surface in question, and so could be deleted from further computations. Furthermore, all rays with two intersections with the cylinder, having $z$ components of either both greater than one or both less than minus one, could be removed, a test which is simply performed.

## 4.4 Swept Surfaces.

Whereas surfaces of revolution were generated by revolving an arbitrary curve about an axis, *swept* surfaces, in their simplest forms, are described by translating a planar curve along an axis. For example, a cylinder was generated by the complete revolution of a straight line about an axis running parallel to it. The same form may also be described

by *sweeping* along that axis a circle lying on the plane perpendicular to the axis. More complex shapes can be obtained through sweeping a curve, or volume, along some arbitrary path in three-space, described by a number of parameters. The following sections, however, will look at the surfaces generated by sweeping a planar curve through some distance along the axis perpendicular to the plane, and more particularly, the intersection of such surfaces with a ray in their object space.

### 4.4.1 Cylindrical Sweeps.

A *cylindrical* sweep (named so because the method can be used to describe a cylinder) simply augments a third variable to the definition of a planar curve, which facilitates the definition of the curve on an infinite number of planes parallel to that on which the curve was described. For example, the coordinate pair $(u(s), w(s))$ representing the parametric curve on the *uw* plane becomes the triple $(u(s), k, w(s))$, where $a \le k \le b$ for some $a$ and $b$.

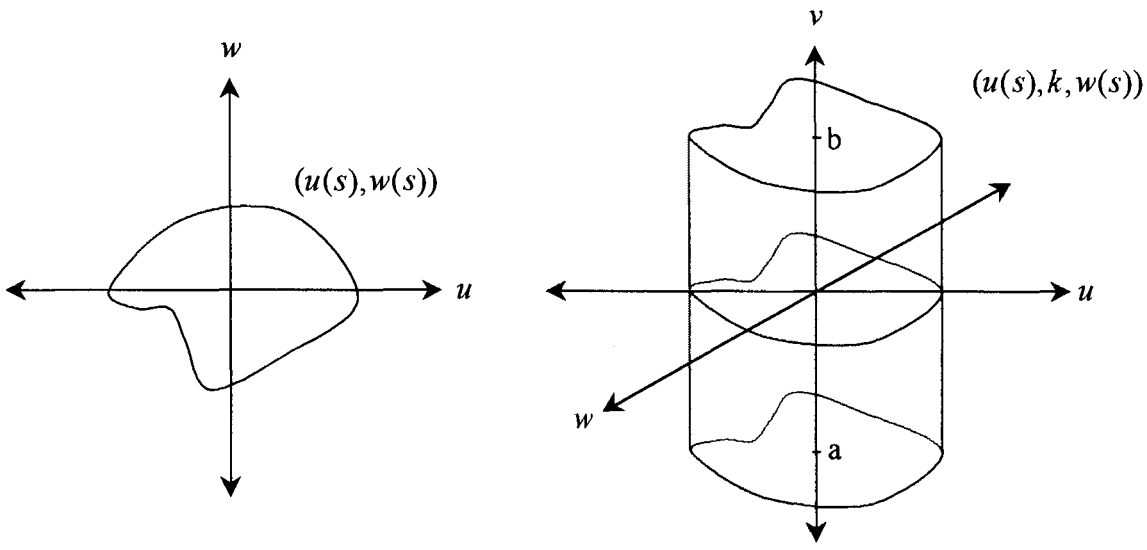



Figure 4.4.1. A cylindrical sweep for $a \le k \le b$ of a parametric curve.

When computing the ray-surface intersection in object space, the ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ can be projected on to the plane of the curve (for example the *uw* plane) by omitting the ray coordinate corresponding to the axis of the sweep (*v*). If the ray is required in its implicit form, it is described by the single two-dimensional line equation:

$$w_t u - u_t w + (u_0 w_t - u_0 w_t) = 0$$

Once the intersection(s), if any, in 2-space is found, the value of the third (omitted) coordinate is obtained for this intersection via the ray parameter *t*. By comparing the value of the third coordinate to the range of the sweep ($a \le k \le b$) we can determine whether the ray actually intersects with the swept surface, or whether it passes above or below.

The normal to the surface is parallel to the plane of the curve (i.e., has a value of 0 for the component representing a direction parallel to the sweep axis), and is defined by taking the first partial derivatives of the curve. For example, for a curve swept along the *v* axis:

$$\mathbf{n} = \left[ \frac{\partial u(s)}{\partial s}, 0, \frac{\partial w(s)}{\partial s} \right]$$

A swept surface may be bounded by planes at either end, much in the same way the surfaces of revolution were previously.

4.4.2 Cone Sweeps.

A *cone* sweep is similar to a cylinder sweep, but scales the curve for which it is defined in proportion to the value of the variable which it augments. For example, the coordinate pair $(u(s), w(s))$ representing the parametric curve on the *uw* plane becomes the triple $(|k|u(s), k, |k|w(s))$, where $a \le k \le b$.

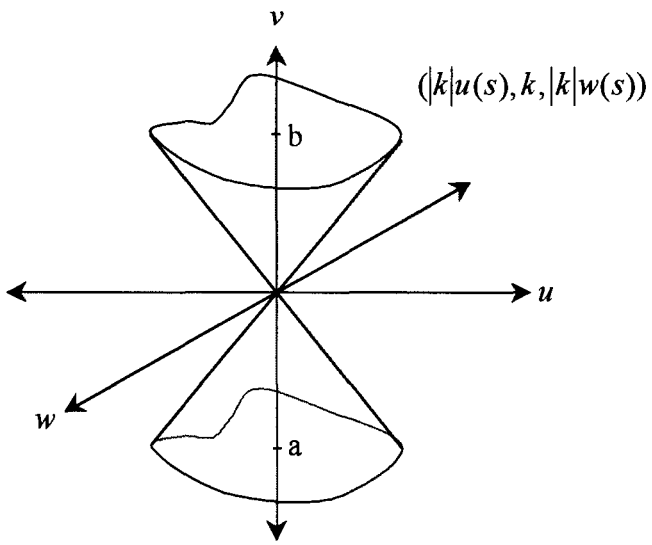
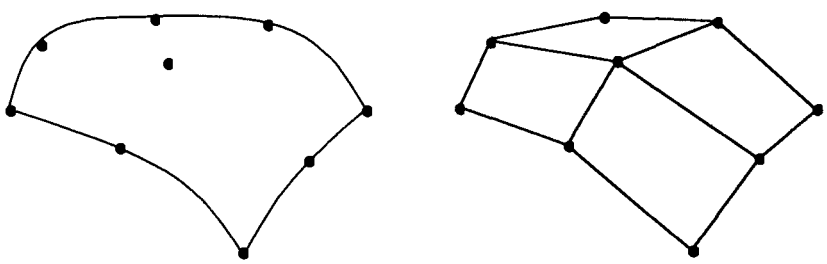



Figure 4.4.2. A cone sweep for $a \le k \le b$.

The intersection with the ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$ may be found in a manner depending on the form of the original curve. Once an intersection is found it may then be compared to the range of the sweep $a \le k \le b$ to decide if the ray passes above, below or through the swept surface.

## 4.5 Parametric Curves.

Consider a smooth object, sampled or approximated, and stored as a set of points defined as vertices describing a polygon mesh. Many applications in the design field require that from such a set of points a smooth curve be drawn, as the output often is to model the real world, in which many objects and surfaces are smooth. This section will discuss methods that enable such a curve to be defined, one that either *interpolates* (passes through) the given data, or provides some reasonable approximation to the given data.



Smooth surface, with sampled points.

Polygon mesh, using points as vertices.

Figure 4.5. Polygon mesh approximation to a smooth surface.

The reader is reminded that whilst this project seeks to examine some of the mathematics behind such techniques, the issue of the practicality of any implementation is of key concern to the extent of the discussion of a technique. The common goal is to generate smooth, user-definable curves and surfaces, but this is not irrespective of the speed and the usefulness of the method employed to do so. In particular, techniques that interpolate the given points are generally computationally expensive and in some instances subject to unwanted effects in between points; thus extended discussion will be reserved for more suitable methods, beginning with two-dimensional curves.

A preliminary note is made here with regard to use of examples in the following discussion. Because these curves and surfaces will be described explicitly, examples that would generate a point alone or a series of points would have little illustrative value. Typically, one would need to evaluate the expression describing the curve or surface at many different values of the parameter to begin understanding the nature of the shape being created, and within this discussion there is little room for such lengthy procedures. Instead the reader is referred to Appendix 2 for information regarding the Microsoft Excel workbooks on the disks included with this report. These workbooks examine some of the parametric curves and surfaces that are to be discussed, and allow manipulation of their defining features. Appendix 2b includes some output from these programs for a more speedy, if less dynamic reference.

### 4.5.1 Piecewise Linear Curves.

A very rudimentary approximation of a smooth curve is through the piecewise use of line segments between the points $(x_i, y_i)$ sampled from it. A function can be written, providing $n$ line segments joining $n+1$ points:

$$P(x) = \begin{cases} \left(\frac{y_0 - y_1}{x_0 - x_1}\right)x + y_0 - \left(\frac{y_0 - y_1}{x_0 - x_1}\right)x_0, & x_0 \le x \le x_1 \\ \left(\frac{y_1 - y_2}{x_1 - x_2}\right)x + y_1 - \left(\frac{y_1 - y_2}{x_1 - x_2}\right)x_1, & x_1 \le x \le x_2 \\ \vdots & \\ \left(\frac{y_{n-1} - y_n}{x_{n-1} - x_n}\right)x + y_{n-1} - \left(\frac{y_{n-1} - y_n}{x_{n-1} - x_n}\right)x_{n-1}, & x_{n-1} \le x \le x_n \end{cases}$$

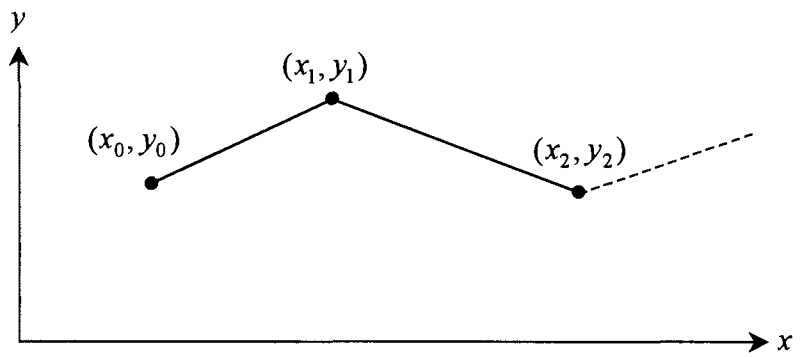



Figure 4.5.1. Piecewise linear interpolation.

We may choose to write this parametrically as:

$$x(t) = \begin{cases} (x_1 - x_0)t_1 + x_0, & 0 \le t_1 \le 1 \\ (x_2 - x_1)t_2 + x_1, & 0 \le t_2 \le 1 \\ \vdots & \\ (x_n - x_{n-1})t_n + x_{n-1}, & 0 \le t_n \le 1 \end{cases} \quad \text{and} \quad y(t) = \begin{cases} (y_1 - y_0)t_1 + y_0, & 0 \le t_1 \le 1 \\ (y_2 - y_1)t_2 + y_1, & 0 \le t_2 \le 1 \\ \vdots & \\ (y_n - y_{n-1})t_n + y_{n-1}, & 0 \le t_n \le 1 \end{cases}$$

This may be more concisely represented by:

$$\mathbf{Q}_i(t_i) = \begin{bmatrix} x(t_i) \\ y(t_i) \end{bmatrix} = \begin{bmatrix} (x_i - x_{i-1}) & x_{i-1} \\ (y_i - y_{i-1}) & y_{i-1} \end{bmatrix} \begin{bmatrix} t_i \\ 1 \end{bmatrix}$$

where $0 \le t_i \le 1$, and $i = 1,\ldots, n$. Note that the parameters $t_i$ could be replaced by the solitary $t$, defining each curve segment at the same time.

Generally, as the number of sample points is increased, so too does the accuracy of the approximation. However this requires more storage space and moreover the method quickly becomes unwieldy and inefficient. Furthermore, while adjacent line segments are joined, adjacent slopes are generally not equal, limiting the extent to which we can accept the method as one that provides a suitable approximation of a smooth, non-linear curve. By increasing the order of the functions defining the curve segments, and forcing

the equality of slopes at the joins, acceptable approximations suitable for use in computer graphics applications can be obtained.

### 4.5.2 Continuity between Curve Segments.

Before any further discussion about the use of curve segments, a notational point is made regarding the continuity of the functions that will be dealt with. Consider the parametric curve segment $\mathbf{Q}_i(t)$ of degree $n$:

$$\mathbf{Q}_i(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \mathbf{C}_i \begin{bmatrix} t^n & \dots & t & 1 \end{bmatrix}^{\mathrm{T}}$$

$\mathbf{C}_i$ is a $2 \times (n+1)$ matrix, describing the coefficients of the functions for curve segment $i$. If $\mathbf{Q}_i(1) = \mathbf{Q}_{i+1}(0)$, that is the curve segments join at point $(x_i, y_i)$, then the curve is said to have *geometric continuity* (Foley et al. 1994, p. 330). This is denoted $G^0$. Furthermore, if $\mathbf{Q}_i'(1) = h\mathbf{Q}_{i+1}'(0)$ then the curve is said to have $G^1$ continuity. This implies that the tangent vectors of each curve at the join are scalar multiples of each other, indicating that the curve is smooth (at least geometrically speaking) at that join. Generally, a curve is said to be $G^n$ continuous at the join between segments $i$ and $i+1$ if $\mathbf{Q}_i^{(n)}(1) = h\mathbf{Q}_{i+1}^{(n)}(0)$.

As a side note, should the curve segments be such that $\mathbf{Q}_i^{(n)}(1) = \mathbf{Q}_{i+1}^{(n)}(0)$ at join $i$ then at this point on the curve is said to have $C^n$ parametric continuity. This is useful for the purposes of animation, where the curve may describe a path of movement, to ensure smooth movement between frames. Clearly, $C^n$ implies $G^n$ continuity (for the case where $h = 1$).

4.5.3 The de Casteljau algorithm.

Consider the four points $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$, and $\mathbf{P}_3$. The piecewise linear interpolation is given by:

$$\mathbf{Q}_i(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} (x_i - x_{i-1}) & x_{i-1} \\ (y_i - y_{i-1}) & y_{i-1} \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}, \text{ where } i = 1, 2, 3.$$

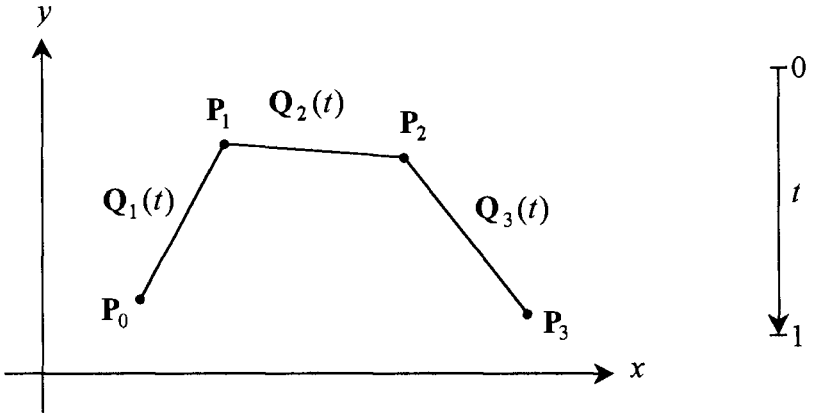



Figure 4.5.3.1 Piecewise linear interpolation of four points.

The de Casteljau algorithm (formulated by Paul de Casteljau in 1959) provides a method for defining a smooth curve which interpolates end points, and which tends towards the intermediate points. Consider a second piecewise linear interpolation performed on the points $(x_1(t), y_1(t))$, $(x_2(t), y_2(t))$, and $(x_3(t), y_3(t))$. The resulting curves may be written (by expanding the notation such that a superscript refers to the level of linear interpolation):

$$\mathbf{Q}_1^{(2)}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} (x_2^{(1)}(t) - x_1^{(1)}(t)) & x_1^{(1)}(t) \\ (y_2^{(1)}(t) - y_1^{(1)}(t)) & y_1^{(1)}(t) \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}$$

and:

$$\mathbf{Q}_2^{(2)}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} (x_3^{(1)}(t) - x_2^{(1)}(t)) & x_2^{(1)}(t) \\ (y_3^{(1)}(t) - y_2^{(1)}(t)) & y_2^{(1)}(t) \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}$$

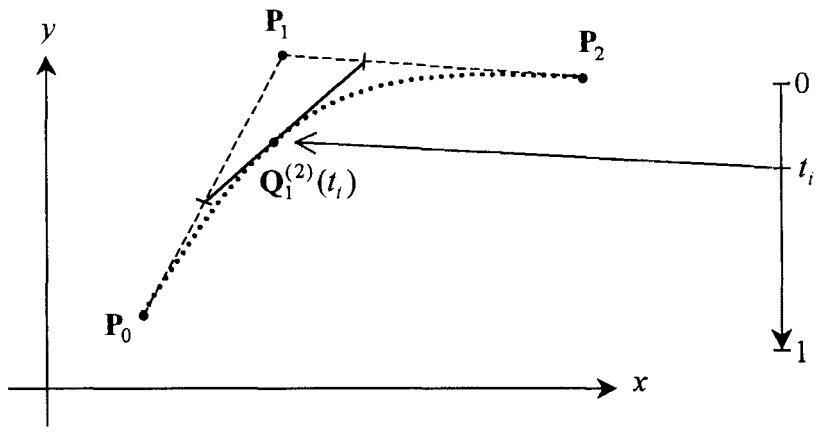



Figure 4.5.3.2 Detail of second linear interpolation.

A third linear interpolation may be performed on the points generated from the second interpolation (and so on if more points are available). The recursive nature of this process may be expressed as follows:

$$\mathbf{Q}_i^{(r)}(t)=\begin{bmatrix} x_i^{(r)}(t) \\ y_i^{(r)}(t) \end{bmatrix}=\begin{bmatrix} (x_{i+1}^{(r-1)}(t)-x_i^{(r-1)}(t)) & x_i^{(r-1)}(t) \\ (y_{i+1}^{(r-1)}(t)-y_i^{(r-1)}(t)) & y_i^{(r-1)}(t) \end{bmatrix}\begin{bmatrix} t \\ 1 \end{bmatrix}, \text{ for } \begin{matrix} r=1,\ldots,n \\ i=1,\ldots,n-r+1 \end{matrix}$$

or, more succinctly,

$$x_i^{(r)}(t)=(1-t)x_i^{(r-1)}(t)+tx_{i+1}^{(r-1)}(t)$$
$$y_i^{(r)}(t)=(1-t)y_i^{(r-1)}(t)+ty_{i+1}^{(r-1)}(t)$$

i.e. $\mathbf{Q}_i^{(r)}(t)=(1-t)\mathbf{Q}_i^{(r-1)}(t)+t\mathbf{Q}_{i+1}^{(r-1)}(t)$, where $\mathbf{Q}_i^{(0)}(t)=\mathbf{P}_{i-1}=[x_{i-1},y_{i-1}]^{\mathrm{T}}$

The above notational abbreviation will be utilised throughout the rest of this discussion.

The de Casteljau algorithm may be used to describe a smooth curve of degree $n$ for $n+1$ points. However, for the purpose of computer graphics, and in particular, applications running at speed, functions of high degree are infrequently used, given the computational expenses they incur. Such costs must be weighed against the usefulness

of the function for, as we have seen, a function of degree $n=1$ provides little more than a piecewise linear approximation to the curve that might be sought. The computer graphics literature suggests that a curve of degree $n=3$, $\mathbf{Q}_i^{(3)}(t)$ for example, is the most suitable compromise (Foley et al. 1994, p. 329). Cubic curves will be examined throughout this section in some detail given they are the ones most commonly used.

### 4.5.4 Bézier Curves.

The cubic Bézier curve is a popular choice of parametrically defined curve, and is used extensively in CAD applications. Each Bézier curve segment may be specified using four *control points* $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2,$ and $\mathbf{P}_3$, two denoting the start and finish of the curve segment, and a further two, each of which lie on a line tangential to the curve at the its end points. The cubic Bézier curve, which will be denoted $\mathbf{Q}_\mathbf{B}(t)$, may be generated using the de Casteljau algorithm:

$$\mathbf{Q}_i^{(r)}(t) = (1-t)\mathbf{Q}_i^{(r-1)}(t) + t\mathbf{Q}_{i+1}^{(r-1)}(t), \text{ where } \mathbf{Q}_i^{(0)}(t) = \mathbf{P}_{i-1}$$

$$\begin{aligned}
\mathbf{Q}_\mathbf{B}(t) &= \mathbf{Q}_i^{(3)}(t) \\
&= (1-t)\mathbf{Q}_i^{(2)}(t) + t\mathbf{Q}_{i+1}^{(2)}(t) \\
&= (1-t)((1-t)\mathbf{Q}_i^{(1)}(t) + t\mathbf{Q}_{i+1}^{(1)}(t)) + t((1-t)\mathbf{Q}_{i+1}^{(1)}(t) + t\mathbf{Q}_{i+2}^{(1)}(t)) \\
&= (1-t)^2\mathbf{Q}_i^{(1)}(t) + 2t(1-t)\mathbf{Q}_{i+1}^{(1)}(t) + t^2\mathbf{Q}_{i+2}^{(1)}(t) \\
&= (1-t)^3\mathbf{Q}_i^{(0)}(t) + 3t(1-t)^2\mathbf{Q}_{i+1}^{(0)}(t) + 3t^2(1-t)\mathbf{Q}_{i+2}^{(0)}(t) + t^3\mathbf{Q}_{i+3}^{(0)}(t) \\
\mathbf{Q}_\mathbf{B}(t) &= (1-t)^3\mathbf{P}_0 + 3t(1-t)^2\mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3\mathbf{P}_3
\end{aligned}$$

Using matrix notation we obtain:

$$\mathbf{Q}_B(t) = \begin{bmatrix}\mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3\end{bmatrix}\begin{bmatrix}(1-t)^3 \\ 3t(1-t)^2 \\ 3t^2(1-t) \\ t^3\end{bmatrix}$$

$$= \begin{bmatrix}\mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}t^3 \\ t^2 \\ t \\ 1\end{bmatrix}$$

$$\mathbf{Q}_B(t) = \mathbf{G}_B\mathbf{M}_B\begin{bmatrix}t^3 & t^2 & t & 1\end{bmatrix}^T$$

where $\mathbf{G}$ is referred to as the geometry matrix, and contains the control points for the particular curve segment, while $\mathbf{M}$ is referred to as a basis matrix for the particular type of curve (in this case $\mathbf{M}_B$ is a Bézier curve basis matrix). This notation is adopted from Foley et al. (1994, p. 331).

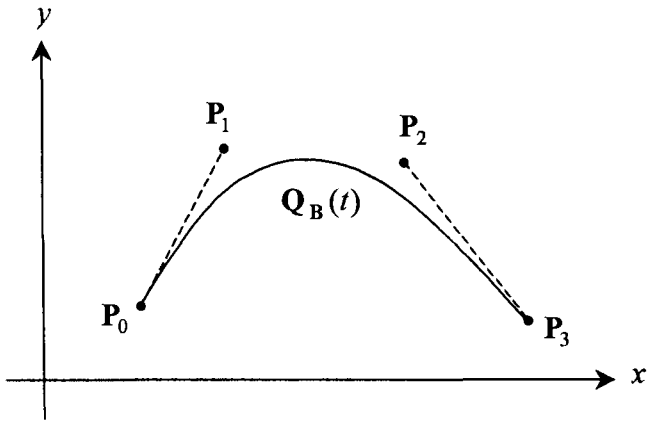



Figure 4.5.4.1 Cubic Bézier curve.

The product $\mathbf{M}_B\begin{bmatrix}t^3 & t^2 & t & 1\end{bmatrix}^T$ provides the weights of the points of $\mathbf{G}_B$, often referred to as the *blending functions*. By plotting the values of each polynomial in $\mathbf{B}_B(t) = \mathbf{M}_B\begin{bmatrix}t^3 & t^2 & t & 1\end{bmatrix}^T$, we can gain an insight into the behaviour of the curve

$\mathbf{Q}_B(t)$ before plotting it. It is noted that the blending functions $B_B(t)$ of the Bézier curve are called the Bernstein polynomials, and they have the general form:

$$B(n,j,t) = \binom{n}{j} t^j (1-t)^{n-j}, \text{ for } j = 0,1,\ldots, n$$

These polynomials have properties that makes the use of Bézier curves attractive in computer graphics applications. Primarily, for any given $n$, the sum of all $n$th degree Bernstein polynomials is one. This is proved as follows (Farin, 1988. pg. 38):

$$\sum_{j=0}^{n} B(n,j,t) = \sum_{j=0}^{n} \binom{n}{j} t^j (1-t)^{n-j} = (t + (1-t))^n = 1$$

Secondly, each polynomial is non-negative. As such, we can think of these polynomials as providing a weighted average of coordinates of the points to which they are applied. Bézier curves, then, may be considered to always lie within a *convex hull*, defined by the control points. A curve is said to have the *convex hull property* if it is contained entirely with the convex boundary of its control points. Foley et al. liken this boundary to a 'rubber band' stretched around the points: any 'interior' points are not considered part of this boundary (1994, p. 338). Mathematically it could be defined as the boundary of the union of all triangles defined by triples of the control points. Whilst the three-dimensional comparison will be made later when dealing with Bézier surfaces, it is clear already that this property would be useful in defining a bounding volume in ray tracing applications.

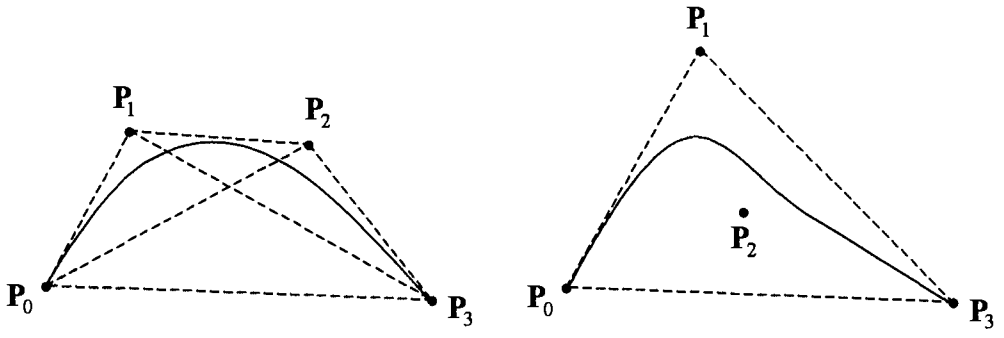



Figure 4.5.4.2. The convex hull property. The union of triangles principle is exemplified on the left, whilst the 'rubber band' effect is demonstrated in the diagram on the right.

Bézier curve segments are joined by allocating points $\mathbf{P}_{k-3}, \mathbf{P}_{k-2}, \mathbf{P}_{k-1}, \mathbf{P}_k$ for some $k$ to a curve segment $\mathbf{Q}_{\mathbf{B}i-1}(t)$, and points $\mathbf{P}_k, \mathbf{P}_{k+1}, \mathbf{P}_{k+2}, \mathbf{P}_{k+3}$ to the following curve segment $\mathbf{Q}_{\mathbf{B}_i}(t)$. $G^0$ continuity is automatically assured between the curves since, by definition, they interpolate start and end points. To attain $G^1$ continuity at the join $\mathbf{P}_k$, the points $\mathbf{P}_{k-1}, \mathbf{P}_k, \mathbf{P}_{k+1}$ must be collinear, or algebraically: $(\mathbf{P}_{k-1} - \mathbf{P}_k) = h(\mathbf{P}_k - \mathbf{P}_{k+1})$ for some $h$. If $h=1$, $C^1$ continuity is attained. By joining curve segments in this manner, much scope is provided for design and application. In general, a piecewise defined cubic Bézier curve $\mathbf{Q}_{\mathbf{B}}(t)$ of $s$ segments requires $3s+1$ control points, and interpolates points $\mathbf{P}_k$, where $k = 0, 3, 6, ..., s$.

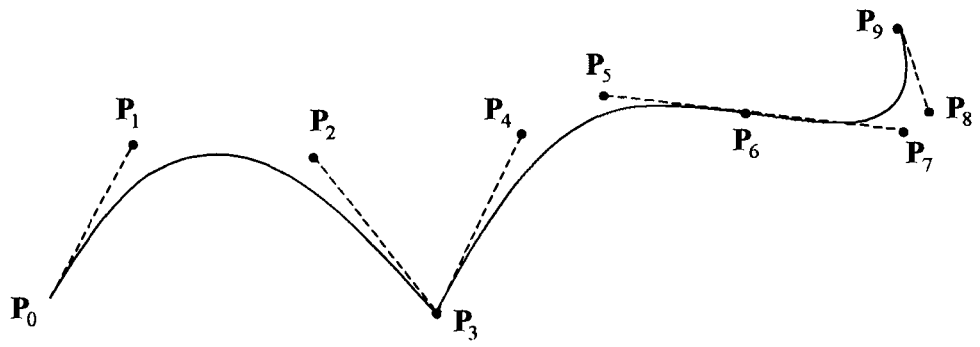



Figure 4.5.4.3. Bézier curve segments joined with: $G^0$ continuity at $\mathbf{P}_3$, and $G^1$ continuity at $\mathbf{P}_6$.

4.5.5 Hermite Curves.

The intermediate control points, $\mathbf{P}_{k+1}$ and $\mathbf{P}_{k+2}$ of a Bézier curve starting at $\mathbf{P}_k$ specify the tangent to the curve at its start and end points, $\mathbf{P}_k$ and $\mathbf{P}_{k+3}$, respectively. The Hermite curve is based on the same principle, but rather than explicitly state the intermediate control points, they are inferred by the slope of the tangents that they make. The following matrix formulation applies:

$$\mathbf{Q}_{\mathrm{H}}(t) = \begin{bmatrix} \mathbf{P}_0 & \mathbf{P}_3 & \mathbf{H}_0 & \mathbf{H}_3 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

$$\mathbf{Q}_{\mathrm{H}}(t) = \mathbf{G}_{\mathrm{H}} \mathbf{M}_{\mathrm{H}} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^{\mathrm{T}}$$

where $\mathbf{H}_0 = \begin{bmatrix} h_{0x} \\ h_{0y} \end{bmatrix}$ and $\mathbf{H}_3 = \begin{bmatrix} h_{3x} \\ h_{3y} \end{bmatrix}$ in which $h_{0x}$ and $h_{3x}$ are the $x$ components and $h_{0y}$ and $h_{3y}$ are the y components of the slopes of the tangent to $\mathbf{Q}_{\mathrm{H}}(t)$ at $\mathbf{P}_0$ and $\mathbf{P}_3$

respectively. As with the case of the Bézier curves, the blending functions given by $\mathbf{B}_{\mathrm{H}}(t) = \mathbf{M}_{\mathrm{H}}\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^{\mathrm{T}}$ may be referred to in order to anticipate the behaviour of the curve.

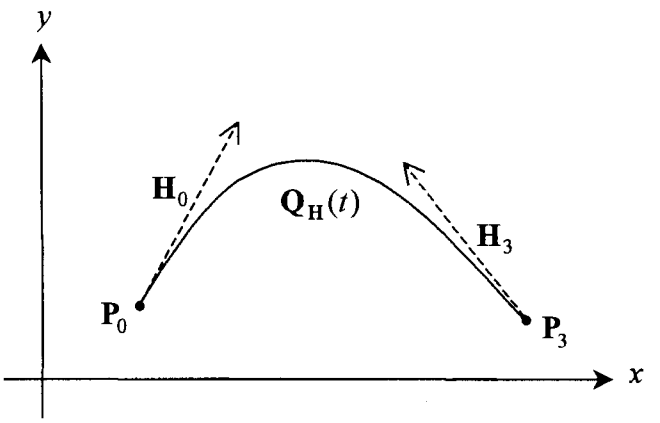



Figure 4.5.5. Hermite curve.

Having previously detailed the mechanics of the Bézier curve, the specification of the Hermite curve using explicit control points is a logical extension of the discussion of cubic parametric curves. By allocating *phantom* intermediate control points $\mathbf{P}_1' = \mathbf{P}_0 + \mathbf{H}_0/j$ and $\mathbf{P}_2' = \mathbf{P}_3 - \mathbf{H}_3/k$, we can compare the two types of curve. Restating $\mathbf{Q}_{\mathrm{H}}(t)$:

$$\mathbf{Q}_{\mathrm{H}}(t) = \begin{bmatrix} \mathbf{P}_0 & \mathbf{P}_3 & \mathbf{H}_0 & \mathbf{H}_3 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

The effect of $\mathbf{H}_0$ and $\mathbf{H}_3$ is equivalent to using the pair of phantom intermediate points $\mathbf{P}_1' = \mathbf{P}_0 + \mathbf{H}_0/j$ and $\mathbf{P}_2' = \mathbf{P}_3 - \mathbf{H}_3/k$:

$$\mathbf{Q}_{\mathbf{H}}^{*}(t)=\begin{bmatrix}\mathbf{P}_0 & \mathbf{P}_1' & \mathbf{P}_2' & \mathbf{P}_3\end{bmatrix}\begin{bmatrix}1 & 0 & -j & 0\\ 0 & 0 & j & 0\\ 0 & 0 & 0 & -k\\ 0 & 1 & 0 & k\end{bmatrix}\begin{bmatrix}2 & -3 & 0 & 1\\ -2 & 3 & 0 & 0\\ 1 & -2 & 1 & 0\\ 1 & -1 & 0 & 0\end{bmatrix}\begin{bmatrix}t^3\\ t^2\\ t\\ 1\end{bmatrix}$$

which we write as

$$\mathbf{Q}_{\mathbf{H}}^{*}(t)=\mathbf{G}_{\mathbf{H}}^{*}(j,k)\mathbf{M}_{\mathbf{H}}\begin{bmatrix}t^3 & t^2 & t & 1\end{bmatrix}^{\mathrm{T}} \tag{4.5.5}$$

The variables $j$ and $k$ represent the magnitude of the tangent vectors at the start and end points respectively. It may be shown by a simple matrix multiplication that for $j=k=3$, $\mathbf{Q}_{\mathbf{H}}^{*}(t)=\mathbf{Q}_{\mathbf{B}}(t)$.

We can adopt (4.5.5) to define a modified Bézier curve $\mathbf{Q}_{\mathbf{B}}^{*}(t)$:

$$\mathbf{Q}_{\mathbf{B}}^{*}(t)=\begin{bmatrix}\mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3\end{bmatrix}\begin{bmatrix}-j+2 & 2j-3 & -j & 1\\ j & -2j & j & 0\\ -k & k & 0 & 0\\ k-2 & -k+3 & 0 & 0\end{bmatrix}\begin{bmatrix}t^3\\ t^2\\ t\\ 1\end{bmatrix}$$

which we may write

$$\mathbf{Q}_{\mathbf{B}}^{*}(t)=\mathbf{G}_{\mathbf{B}}\mathbf{M}_{\mathbf{B}}^{*}(j,k)\begin{bmatrix}t^3 & t^2 & t & 1\end{bmatrix}^{\mathrm{T}}$$

When the curve is drawn, the values of $j$ and $k$ determine the 'speed' at which the curve leaves the end points in the direction of the intermediate control points (i.e. the magnitude of the tangent to the curve). The convex hull property applies to the points $\mathbf{P}_0$, $\mathbf{P}_1^{*}=\mathbf{P}_0+j(\mathbf{P}_1-\mathbf{P}_0)/3$, $\mathbf{P}_2^{*}=\mathbf{P}_3-k(\mathbf{P}_3-\mathbf{P}_2)/3$, and $\mathbf{P}_3$.

### 4.5.6 Uniform Cubic B-Splines.

The problem of approximating a curve, given only a number of sample points from it, is one that could be solved using a so called natural cubic spline which interpolates those points and ensures parametric continuity up to $C^2$. However, this approach offers little of the flexibility of Bézier curves since arbitrary alteration of points requires much re-computation. Uniform cubic B-splines however, do exhibit such *local control*, at the expense of the interpolation of any of the data.

Cubic B-splines differ from the previous piecewise curves that we have discussed in that, rather than the curve being made up of segments $\mathbf{Q}_{\mathbf{BS}_i}(t)$ each relying on four distinct control points $\mathbf{P}_{3(i-1)}, \mathbf{P}_{3(i-1)+1}, \mathbf{P}_{3(i-1)+2}, \mathbf{P}_{3(i-1)+3}$, the curve is defined progressively by segments defined by consecutive quadruples of control points. This is more formally written:

$$\mathbf{G}_{\mathbf{BS}_i} = \begin{bmatrix} \mathbf{P}_{i-1} & \mathbf{P}_i & \mathbf{P}_{i+1} & \mathbf{P}_{i+2} \end{bmatrix}$$

A cubic B-spline with $s-2$ segments needs $s+1$ control points $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_s$. Each curve segment $\mathbf{Q}_{\mathbf{BS}_i}(t)$ is defined for the parameter $t$, where $t_{i-1} \le t < t_i$, with $t_0 = 0$ and $t_i - t_{i-1} = 1$ (this is a *uniform* parameter interval and so these splines are often referred to as such).

The matrix equation for a curve segment of a uniform B-spline is given, for $i = 1,\ldots, s-2$:

$$\mathbf{Q}_{\mathbf{BS}_i}(t) = \begin{bmatrix}\mathbf{P}_{i-1} & \mathbf{P}_i & \mathbf{P}_{i+1} & \mathbf{P}_{i+2}\end{bmatrix}\frac{1}{6}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 0 & 4\\ -3 & 3 & 3 & 1\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}(t-t_{i-1})^3\\ (t-t_{i-1})^2\\ t-t_{i-1}\\ 1\end{bmatrix}$$

which we can rewrite as

$$\begin{aligned}\mathbf{Q}_{\mathbf{BS}_i}(t) &= \mathbf{G}_{\mathbf{BS}_i}\mathbf{M}_{\mathbf{BS}}\begin{bmatrix}(t-t_{i-1})^3, & (t-t_{i-1})^2, & t-t_{i-1}, & 1\end{bmatrix}^{\mathrm{T}}\\ &= \mathbf{G}_{\mathbf{BS}_i}\mathbf{B}_{\mathbf{BS}}(t-t_{i-1})\end{aligned}$$

Much can be understood about the behaviour of these curves by examining the blending functions. By abbreviating $(t-t_{i-1})$ with $t$ (a notation that will see much use in the upcoming discussion), the cubic B-spline blending functions are expressed as follows:

$$\begin{aligned}\mathbf{B}_{\mathbf{BS}}(t) &= \mathbf{M}_{\mathbf{BS}}\begin{bmatrix}t^3 & t^2 & t & 1\end{bmatrix}^{\mathrm{T}}\\ &= \frac{1}{6}\begin{bmatrix}(1-t)^3, & 3t^3-6t^2+4, & -3t^3+3t^2+3t+1, & t^3\end{bmatrix}^{\mathrm{T}} \text{ for } 0 \le t \le 1\end{aligned}$$

As for the Bézier curves, the blending functions are non-negative in the domain $0 \le t < 1$, and sum to one. Therefore the convex hull property applies to cubic uniform B-spline segments. Similarly, we can also glean information about the continuity of these curves at the joins or *knots* (the literature refers to the points $\mathbf{Q}_{\mathbf{BS}_i}(0)$, for $i = 1,\ldots, s-1$, as knots) by examination of the blending functions. For example, to demonstrate $C^2$ continuity, we need the second derivative of $\mathbf{Q}_{\mathbf{BS}}(t)$, and hence the

second derivative of $\mathbf{B}_{\mathbf{BS}}(t)$ which is $\mathbf{B}_{\mathbf{BS}}^{(2)}(t) = [1-t, \quad 3t-2, \quad -3t+1, \quad t]^{\mathrm{T}}$. We can therefore write:

$$\begin{aligned}\mathbf{G}_{\mathbf{BS}i}\mathbf{B}_{\mathbf{BS}}^{(2)}(1) &= [\mathbf{P}_{i-1} \quad \mathbf{P}_i \quad \mathbf{P}_{i+1} \quad \mathbf{P}_{i+2}][0 \quad 1 \quad -2 \quad 1]^{\mathrm{T}} \\ &= \mathbf{P}_i - 2\mathbf{P}_{i+1} + \mathbf{P}_{i+2}\end{aligned}$$

and

$$\begin{aligned}\mathbf{G}_{\mathbf{BS}i+1}\mathbf{B}_{\mathbf{BS}}^{(2)}(0) &= [\mathbf{P}_i \quad \mathbf{P}_{i+1} \quad \mathbf{P}_{i+2} \quad \mathbf{P}_{i+3}][1 \quad -2 \quad 1 \quad 0]^{\mathrm{T}} \\ &= \mathbf{P}_i - 2\mathbf{P}_{i+1} + \mathbf{P}_{i+2}\end{aligned}$$

from which it follows that

$$\mathbf{Q}_{\mathbf{BS}i}^{(2)}(1) = \mathbf{Q}_{\mathbf{BS}i+1}^{(2)}(0)$$

indicating $C^2$ continuity.

$C^1$ and $C^0$ continuities can be demonstrated in a similar manner. The continuities $C^2$, $C^1$ and $C^0$ are automatic across all knots, unlike that for the Bézier curves, where the careful (collinear) positioning of all control points in each segment was necessary.

The following sections look at some extensions to the uniform B-spline.

#### 4.5.6.1 β-Splines: Tensioning a spline.

By using modified blending functions the behaviour of a spline near its control points can be controlled more precisely. Such splines are known as β-splines, and much research has been carried out in this particular area (Burger, 1989. p. 268). The blending function presented here is adapted from a β-spline given by Barsky (1983, from Burger, 1989. p. 269), and features a tension parameter $\tau$ that allows the curve to be pulled towards the control points:

$$\mathbf{B}_\beta(t,\tau)=\frac{1}{12+\tau}\begin{bmatrix} 2(1-t)^3 \\ (8+\tau)-3(4+\tau)t^2+2(3+\tau)t^3 \\ 2+6t+3(2+\tau)t^2-2(3+\tau)t^3 \\ 2t^3 \end{bmatrix}, \text{ for } \tau \geq 0.$$

The corresponding basis matrix $\mathbf{M}_\beta$ is written:

$$\mathbf{M}_\beta(\tau)=\frac{1}{12+\tau}\begin{bmatrix} -2 & 6 & -6 & 2 \\ 2(3+\tau) & -3(4+\tau) & 0 & 8+\tau \\ -2(3+\tau) & 3(2+\tau) & 6 & 2 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

At $\tau=0$ there is no tension, and $\mathbf{M}_\beta(0)$ reduces to $\mathbf{M_{BS}}$. Usage of $\mathbf{M}_\beta(\tau)$ is the same as $\mathbf{M_{BS}}$, i.e. $\mathbf{Q}_{\beta_i}(t)=\mathbf{G}_{\beta_i}\mathbf{M}_\beta(\tau)\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^T$.

4.5.6.2 Controlling End Points of a B-Spline.

Since a B-spline does not usually interpolate the control points, the end points of such a curve are prone to unsatisfactory positioning, and this is of particular concern in a design application where precision is the goal. However, a number of techniques exist to curb the sometimes erratic nature of the extremes of a B-spline.

By duplicating the first and last control points $\mathbf{P}_0$ and $\mathbf{P}_s$ by defining $\mathbf{P}_{-1}=\mathbf{P}_0$ and $\mathbf{P}_{s+1}=\mathbf{P}_s$ respectively, the curve is extended towards them with an extra curve segment at each end. Furthermore, the slope of these segments approaches that of the line extending from each of the end points to the corresponding nearest non-duplicate point ($\mathbf{P}_1$ and $\mathbf{P}_{s-1}$). For the discussion we will add a subscript $x$ to the

blending function notation so that $B_{\mathbf{BS}x}$ denotes the weighting applied to a particular control point $x$:

$$\mathbf{Q}_{\mathbf{BS}i}(t) = B_{\mathbf{BS}i-1}(t)\mathbf{P}_{i-1} + B_{\mathbf{BS}i}(t)\mathbf{P}_i + B_{\mathbf{BS}i+1}(t)\mathbf{P}_{i+1} + B_{\mathbf{BS}i+2}(t)\mathbf{P}_{i+2} \qquad (4.5.6.2)$$

where
$$\begin{aligned} B_{\mathbf{BS}i-1}(t) &= (1-t)^3 \\ B_{\mathbf{BS}i}(t) &= 3t^3 - 6t^2 + 4 \\ B_{\mathbf{BS}i+1}(t) &= -3t^3 + 3t^2 + 3t + 1 \\ B_{\mathbf{BS}i+2}(t) &= t^3 \end{aligned} \quad \text{for } 0 \le t \le 1.$$

Now, the curve segments generated by the duplication of the end points ($i = 0$ and $i = s$) can be written using the fact that $\mathbf{P}_{-1} = \mathbf{P}_0$ and $\mathbf{P}_{s+1} = \mathbf{P}_s$:

$$\mathbf{Q}_{\mathbf{BS}0}(t) = (B_{\mathbf{BS}-1}(t) + B_{\mathbf{BS}0}(t))\mathbf{P}_0 + B_{\mathbf{BS}1}(t)\mathbf{P}_1 + B_{\mathbf{BS}2}(t)\mathbf{P}_2$$

and

$$\mathbf{Q}_{\mathbf{BS}s-1}(t) = B_{\mathbf{BS}s-2}(t)\mathbf{P}_{s-2} + B_{\mathbf{BS}s-1}(t)\mathbf{P}_{s-1} + (B_{\mathbf{BS}s}(t) + B_{\mathbf{BS}s+1}(t))\mathbf{P}_s$$

When the end points $\mathbf{P}_0$ and $\mathbf{P}_s$ are triplicated by adding yet another pair of additional points, the new curve segments $\mathbf{Q}_{\mathbf{BS}-1}(t)$ and $\mathbf{Q}_{\mathbf{BS}s}(t)$ interpolate the respective end points:

The curve segment $\mathbf{Q}_{\mathbf{BS}-1}(t)$ is defined by using (4.5.6.2) for $\mathbf{P}_{-2} = \mathbf{P}_{-1} = \mathbf{P}_0$ and $\mathbf{P}_1$:

$$\mathbf{Q}_{\mathbf{BS}-1}(t) = (B_{\mathbf{BS}-2}(t) + B_{\mathbf{BS}-1}(t) + B_{\mathbf{BS}0}(t))\mathbf{P}_0 + B_{\mathbf{BS}1}(t)\mathbf{P}_1$$

from which it follows that

$$\mathbf{Q}_{\mathbf{BS}-1}(0) = (1/6 + 2/3 + 1/6)\mathbf{P}_0 + (0)\mathbf{P}_1 = \mathbf{P}_0$$

Similarly, using (4.5.6.2) the curve segment $\mathbf{Q}_{\mathbf{BS}\,s}(t)$ is defined for $\mathbf{P}_s = \mathbf{P}_{s+1} = \mathbf{P}_{s+2}$ and $\mathbf{P}_{s-1}$:

$$\mathbf{Q}_{\mathbf{BS}\,s}(t) = B_{\mathbf{BS}\,s-1}(t)\mathbf{P}_{s-1} + (B_{\mathbf{BS}\,s}(t) + B_{\mathbf{BS}\,s+1}(t) + B_{\mathbf{BS}\,s+2}(t))\mathbf{P}_s$$

from which it follows that

$$\mathbf{Q}_{\mathbf{BS}\,s}(1) = (0)\mathbf{P}_{s-1} + (1/6 + 2/3 + 1/6)\mathbf{P}_s = \mathbf{P}_s$$

These techniques can be applied anywhere along the curve, but reduce the level of continuity for each replication performed.

Another method by which the end points can be treated is through the specification of phantom end points, which constrain the curve to pass through $\mathbf{P}_0$ and $\mathbf{P}_s$. Labelling these phantom points $\mathbf{P}^*_{-1}$ and $\mathbf{P}^*_{s+1}$ the following conditions hold at the respective parameter values:

$$\mathbf{Q}_{\mathbf{BS}\,0}(0) = 1/6\,\mathbf{P}^*_{-1} + 2/3\,\mathbf{P}_0 + 1/6\,\mathbf{P}_1 + \mathbf{P}_2(0) = \mathbf{P}_0$$

$$\mathbf{P}^*_{-1} = 2\mathbf{P}_0 - \mathbf{P}_1$$

and similarly:

$$\mathbf{Q}_{\mathbf{BS}\,s-1}(1) = \mathbf{P}_{s-2}(0) + 1/6\,\mathbf{P}_{s-1} + 2/3\,\mathbf{P}_s + 1/6\,\mathbf{P}^*_{s+1} = \mathbf{P}_0$$

$$\mathbf{P}^*_{s+1} = 2\mathbf{P}_s - \mathbf{P}_{s-1}$$

Whilst the B-spline does not pass through the phantom points, it does begin and end at the desired locations. Furthermore, the slope at the end of these curves follows the gradient of the line joining the first two and the last two control points.

4.5.6.3 Interpolation Using a Cubic B-Spline.

A B-spline may interpolate a set of control points $\mathbf{P}_i$ (for $i = 0,\ldots,s$), by describing a set of $s+3$ phantom points that blend at the ends of each curve segment to each of those control points. Note that there are now $s$ curve segments (numbered 0 to $s-1$; one between successive pairs of control points). For the phantom points denoted by $\mathbf{P}_i^*$ ($i = -1,\ldots,s+1$) this can be written:

$$\mathbf{Q}_{\mathbf{BS}_i}(t) = B_{\mathbf{BS}_{i-1}}(t)\mathbf{P}_{i-1}^* + B_{\mathbf{BS}_i}(t)\mathbf{P}_i^* + B_{\mathbf{BS}_{i+1}}(t)\mathbf{P}_{i+1}^* + B_{\mathbf{BS}_{i+2}}(t)\mathbf{P}_{i+2}^*, \text{ for } i = 0,\ldots, s-1.$$

$\mathbf{Q}_{\mathbf{BS}_i}(t)$ interpolates the control points:

$$\begin{aligned} \mathbf{Q}_{\mathbf{BS}_0}(0) &= \mathbf{P}_0 \\ \mathbf{Q}_{\mathbf{BS}_i}(1) &= \mathbf{Q}_{\mathbf{BS}_{i+1}}(0) = \mathbf{P}_{i+1}, \text{ for } i = 0,\ldots, s-2, \\ \mathbf{Q}_{\mathbf{BS}_{s-1}}(1) &= \mathbf{P}_s \end{aligned}$$

and so

$$\mathbf{P}_i = (1/6)\mathbf{P}_{i-1}^* + (2/3)\mathbf{P}_i^* + (1/6)\mathbf{P}_{i+1}^*, \text{ for } i = 0,\ldots, s$$

In defining the end phantom points $\mathbf{P}_{-1}^*$ and $\mathbf{P}_{s+1}^*$ there is an opportunity to force the gradient of the curve segments at the end points by taking the first derivative of $\mathbf{Q}_{\mathbf{BS}_i}(t)$. The phantom points $\mathbf{P}_{-1}^*$ and $\mathbf{P}_{s+1}^*$ are defined in relation to user-selected gradients $\mathbf{m}_0$ and $\mathbf{m}_s$ as follows:

$$\begin{aligned}\mathbf{m}_0 &= \mathbf{Q}_{\mathrm{BS}0}'(0) \\ &= B_{\mathrm{BS}-1}'(0)\mathbf{P}_{-1}^* + B_{\mathrm{BS}0}'(0)\mathbf{P}_0^* + B_{\mathrm{BS}1}'(0)\mathbf{P}_1^* + B_{\mathrm{BS}2}'(0)\mathbf{P}_2^* \\ &= -3\mathbf{P}_{-1}^* + 3\mathbf{P}_1^*\end{aligned}$$

and

$$\begin{aligned}\mathbf{m}_s &= \mathbf{Q}_{\mathrm{BS}s-1}'(1) \\ &= B_{\mathrm{BS}s-2}'(1)\mathbf{P}_{s-2}^* + B_{\mathrm{BS}s-1}'(1)\mathbf{P}_{s-1}^* + B_{\mathrm{BS}s}'(1)\mathbf{P}_s^* + B_{\mathrm{BS}s+1}'(1)\mathbf{P}_{s+1}^* \\ &= -3\mathbf{P}_{s-1}^* + 3\mathbf{P}_{s+1}^*\end{aligned}$$

In summary there are $s+3$ unknowns, which are the phantom control points $\mathbf{P}_{-1}^*$ to $\mathbf{P}_{s+1}^*$, and there are $s+3$ equations, of which $s+1$ are described by the relation $\mathbf{P}_i = (1/6)\mathbf{P}_{i-1}^* + (2/3)\mathbf{P}_i^* + (1/6)\mathbf{P}_{i+1}^*$, for $i = 0,\ldots, s$. The other two equations are those concerning the gradient at the end points as described above. The system of equations can be solved to determine the phantom control points that will provide a cubic B-spline curve that interpolates the given control points:

$$\begin{bmatrix} -3 & 0 & 3 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 1/6 & 2/3 & 1/6 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 1/6 & 2/3 & 1/6 & \ldots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1/6 & 2/3 & 1/6 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1/6 & 2/3 & 1/6 \\ 0 & 0 & 0 & 0 & \ldots & 0 & -3 & 0 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{-1}^* \\ \mathbf{P}_0^* \\ \mathbf{P}_1^* \\ \vdots \\ \mathbf{P}_{s-1}^* \\ \mathbf{P}_s^* \\ \mathbf{P}_{s+1}^* \end{bmatrix} = \begin{bmatrix} \mathbf{m}_0 \\ \mathbf{P}_0 \\ \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_{s-1} \\ \mathbf{P}_s \\ \mathbf{m}_s \end{bmatrix}$$

$$\mathbf{A}\mathbf{G}^* = \mathbf{G}$$

$$\mathbf{G}^* = \mathbf{A}^{-1}\mathbf{G}$$

in which $\mathbf{A}$ is of order $(s+3)\times(s+3)$.

In implementing this method, the pre-computation of the inverse of **A** may be problematic in that it restricts the number of control points available for use. Preferably, some general iterative method of solution such as Gaussian elimination would be used instead.

Each interpolating curve segment may be drawn according to:

$$\mathbf{Q}_{\mathbf{BS}_i}(t) = \mathbf{G}^*_{\mathbf{BS}_i}\mathbf{B}_{\mathbf{BS}_i}, \text{ for } i = 0,\ldots, s-1$$

where $\mathbf{G}^*_{\mathbf{BS}_i}$ is the row vector of the $i$th set of four consecutive elements from $\mathbf{G}^*$, i.e.

$$\mathbf{G}^*_{\mathbf{BS}_i} = \begin{bmatrix} \mathbf{P}^*_{i-1} & \mathbf{P}^*_i & \mathbf{P}^*_{i+1} & \mathbf{P}^*_{i+2} \end{bmatrix}$$

It is of note that the matrix **A** is such that, should a particular control point $\mathbf{P}_i$ not need to be interpolated, row $i$ in **A** can be adjusted so that all entries are 0 other than a 1 on the main diagonal. The curve resulting from the phantom points generated by this matrix no longer interpolates control point $i$. Similarly, should only one control point $j$ need to be interpolated, then all rows other than row $j$ have all entries 0 other than a 1 on the main diagonal.

### 4.5.7 Non-Uniform B-Splines.

Previously, the uniform B-splines discussed were subject to the condition $t_i - t_{i-1} = 1$, for all segments $i = 1,\ldots, s-2$. This was advantageous in that the entire curve was defined using the matrix $\mathbf{B}_{\mathbf{BS}}(t) = \mathbf{M}_{\mathbf{BS}}\left[(t-t_{i-1})^3, \ (t-t_{i-1})^2, \ t-t_{i-1}, \ 1\right]^{\mathrm{T}}$ for the blending functions. However, relaxing this condition brings an increased generality to the formulation can make the spline easier to manipulate. The expression of a non-uniform cubic B-spline is:

$$\mathbf{Q}_{\mathrm{BS}\,i}(t) = \mathbf{P}_{i-1}\mathbf{B}_{i-1,4}(t) + \mathbf{P}_i\mathbf{B}_{i,4}(t) + \mathbf{P}_{i+1}\mathbf{B}_{i+1,4}(t) + \mathbf{P}_{i+2}\mathbf{B}_{i+2,4}(t)$$

with recursively defined blending functions:

$$\mathbf{B}_{i,1}(t) = \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise,} \end{cases}$$

$$\mathbf{B}_{i,2}(t) = \frac{t - t_i}{t_{i+1} - t_i}\mathbf{B}_{i,1}(t) + \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}}\mathbf{B}_{i+1,1}(t)$$

$$\mathbf{B}_{i,3}(t) = \frac{t - t_i}{t_{i+2} - t_i}\mathbf{B}_{i,2}(t) + \frac{t_{i+3} - t}{t_{i+3} - t_{i+1}}\mathbf{B}_{i+1,2}(t)$$

$$\mathbf{B}_{i,4}(t) = \frac{t - t_i}{t_{i+3} - t_i}\mathbf{B}_{i,3}(t) + \frac{t_{i+4} - t}{t_{i+4} - t_{i+1}}\mathbf{B}_{i+1,3}(t)$$

Without any constraint on the parameter interval $t_i \le t \le t_{i+1}$, it is possible to define a number of knots, or joins, for the same value of $t_i$. An interpolation is thus achieved, yet without the necessity of incurring a strictly linear function on either side of the knots which was the case for uniform B-splines (without phantom control points). Furthermore, extra knots and control points can be added easily to reshape the curve.

## 4.6 Parametric Surfaces.

A parametric surface may be thought of as a span of parametric curves which vary from each other according to some second parameter, and at the same time a span of parametric curves defined by that second parameter, varying from each other according to the first parameter. Based on the parametric curves discussed previously, the *bi-cubic* surfaces generated by Bézier and B-spline curves will now be examined.

4.6.1 Properties of the Parametric Bi-Cubic Surface.

The general form of a parametric bi-cubic surface is given by:

$$\mathbf{S}(t,u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathbf{M}^{\mathrm{T}} \mathbf{G} \mathbf{M} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^{\mathrm{T}},$$

or if $\mathbf{S}(t,u) = [x(t,u), y(t,u), z(t,u)]^{\mathrm{T}}$

$$\begin{aligned} x(t,u) &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathbf{M}^{\mathrm{T}} \mathbf{G}_x \mathbf{M} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^{\mathrm{T}} \\ y(t,u) &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathbf{M}^{\mathrm{T}} \mathbf{G}_y \mathbf{M} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^{\mathrm{T}}, \quad 0 \le t,u \le 1. \\ z(t,u) &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathbf{M}^{\mathrm{T}} \mathbf{G}_z \mathbf{M} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^{\mathrm{T}} \end{aligned}$$

where: **G** is the geometry matrix, which contains the control points in each co-ordinate,

**M** is the basis matrix for the particular curve upon which the surface is based.

The normal to such a surface is simply expressed, given that the vectors tangential to the surface are readily found through a first partial derivative with respect to each of the parameters involved. The cross product then yields the normal vector as a bi-quintic expression (Foley et al. 1994, p. 355):

$$\mathbf{n} = \frac{\partial \mathbf{S}(t,u)}{\partial t} \times \frac{\partial \mathbf{S}(t,u)}{\partial u}$$

$$\mathbf{n} = \begin{bmatrix} \dfrac{\partial x(t,u)}{\partial t} & \dfrac{\partial y(t,u)}{\partial t} & \dfrac{\partial z(t,u)}{\partial t} \\ \dfrac{\partial x(t,u)}{\partial u} & \dfrac{\partial y(t,u)}{\partial u} & \dfrac{\partial z(t,u)}{\partial u} \\ \mathbf{i} & \mathbf{j} & \mathbf{k} \end{bmatrix}$$

An intersection of a patch with a ray may be defined when the ray is expressed implicitly as the intersection of two planes (see section 3.3.1). The alternative would be to implicitize the bi-cubic patch and intersect it with a parametrically defined ray. However, Hanrahan (in Glassner, 1989) states that a bi-cubic patch can lead to an

implicit surface of degree 18, and so the possibly simpler (in that it is simpler to implicitize a ray than a patch) problem is presented here.

Given the intersection of two planes representing the ray:

$$a_1 x + b_1 y + c_1 z + d_1 = 0$$
$$a_2 x + b_2 y + c_2 z + d_2 = 0$$

The surface's coordinates can then be substituted in:

$$a_1 x(t,u) + b_1 y(t,u) + c_1 z(t,u) + d_1 = 0$$
$$a_2 x(t,u) + b_2 y(t,u) + c_2 z(t,u) + d_2 = 0$$

Given the relatively high degree of the polynomials, one would seek to facilitate bounding volume algorithms prior to pursuing a solution of the above. Fortunately, as hinted at earlier, parametric bi-cubic surfaces such as the Bézier and B-spline definitions satisfy the convex hull property, from which a bounding volume is readily identified. In the two-dimensional case, the 'rubber band' analogy was drawn to illustrate this. A similar analogy exists for the surface representations, whereby one can imagine stretching a balloon over the control points to ascertain the points that form the convex hull, and hence define the facets to test against all incoming rays.

### 4.6.2 Bézier Surfaces.

The Bézier surface is represented as follows:

$$\mathbf{S_B}(t,u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathbf{M_B}^{\mathrm{T}} \mathbf{G_B} \mathbf{M_B} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^{\mathrm{T}}, \text{ for } 0 \le t,u \le 1$$

$$\text{where } \mathbf{M_B} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{G_B} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} & \mathbf{P}_{14} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} & \mathbf{P}_{24} \\ \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} & \mathbf{P}_{34} \\ \mathbf{P}_{41} & \mathbf{P}_{42} & \mathbf{P}_{43} & \mathbf{P}_{44} \end{bmatrix}$$

$\mathbf{G_B}$, the geometry matrix for the Bézier surface, is an arrangement of 16 control points that can be thought of as describing two 'horizontal' Bézier curves ($\mathbf{P}_{11}$ to $\mathbf{P}_{14}$ and $\mathbf{P}_{41}$ to $\mathbf{P}_{44}$), and two 'vertical' Bézier curves ($\mathbf{P}_{11}$ to $\mathbf{P}_{41}$ and $\mathbf{P}_{14}$ to $\mathbf{P}_{44}$). As such, the corner entries are all interpolated by the surface.

Bézier surfaces can be joined in a manner similar to their two dimensional counterparts. The literature (for example, Burger, 1989. p. 268) refers to these separate surfaces to be joined as *patches*. $G^0$ continuity is attained simply by making the four control points along the edge of the join equal. $G^1$ continuity is achieved when the four triplets of points along an edge (i.e., the edge points and those either side of the edge points) each exhibit collinearity. $C^1$ continuity is attainable, but perhaps not desirable given the restrictions it places on points. To underline that observation, consider a patch that is joined on all four sides to other patches. In this case every single element of $\mathbf{G_B}$ is influencing the continuity of at least one of the joins of that patch.

### 4.6.3 B-Spline Surfaces.

A B-spline patch can be written:

$$\mathbf{S_{BS}}(t,u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathbf{M_{BS}}^{\mathrm{T}} \mathbf{G_{BS}} \mathbf{M_{BS}} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^{\mathrm{T}}, \text{ for } 0 \le t,u \le 1$$

$$\text{where } \mathbf{M_B} = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

Note that the effect of the geometry matrix $\mathbf{G_{BS}}$ is very different from that of the Bézier form. Rather than defining a regular 'patchwork' of control points, representing the individual patches, any size matrix of points will suffice (providing it is at least $4 \times 4$). The B-spline surface may then take each particular $4 \times 4$ matrix, and without any continuity conditions other than an avoidance of duplicate control points, generate a patch which is automatically joined to the adjacent patches.

The reader is reminded that examples of curves and surfaces described from Section 4.5.4 through to this point are presented in Appendix 2b. Moreover they are encouraged to try the Excel workbooks included with this thesis, which were used to generate the images presented.

## 5. Illumination Models and Shading of Surfaces.

The process of illumination is based in part on the physical behaviour of light in an environment. However, some illumination models mimic physical laws more closely than others. Therefore, the following methods should be considered to be approximations of the applicable physical laws, simply because they have been developed to provide desirable image quality rather than strictly obey the laws of optics.

### 5.1 Model Notation.

An illumination model specifies the factors that determine a surface's colour at a given point (Foley et al. 1994, p. 477). To begin to implement such a model, the term *colour* needs to be quantified in some manner. For the purposes of this project, colour will be described in terms the *intensity*, $I_\lambda$, of light in possibly a number of *wavelengths*, each denoted $\lambda$, and with no further detail. The colour of a point on a surface, then, may be described as the net perception of all $I_\lambda$ for all light that makes its way from that point to the eye.

The specification of light sources within the environment takes one of two forms. We may define a *point* light source, $L$, from which light of specified wavelength(s) $\lambda$ and intensity $I_{\lambda L}$ is emitted, by its location in the problem world $(x_L, y_L, z_L)$. From any point $(x, y, z)$ in the problem world, the unit vector $\hat{\mathbf{L}}$ indicates the direction to this light source:

$$\hat{\mathbf{L}} = \frac{\begin{bmatrix} x_L - x, & y_L - y, & z_L - z \end{bmatrix}^{\mathrm{T}}}{\left\| \begin{bmatrix} x_L - x, & y_L - y, & z_L - z \end{bmatrix}^{\mathrm{T}} \right\|}$$

By placing a point source at infinity, a *directional* light source is defined. The vector $\hat{\mathbf{L}} = [x_L, y_L, z_L]^T / \left\| [x_L, y_L, z_L]^T \right\|$ denotes the direction opposite to that of light rays from this infinite source.

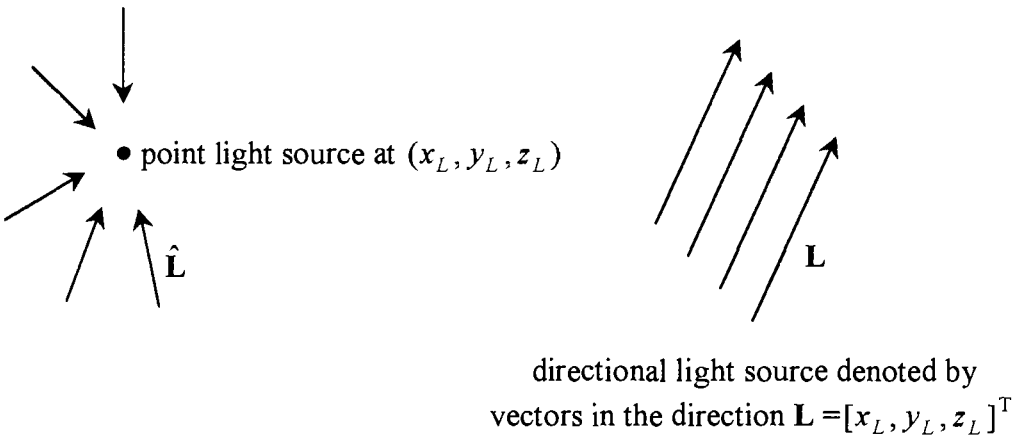



Figure 5.1. Depiction of point and directional light sources.

To evaluate $I_\lambda$ at a point on a surface, the behaviour of light at that point is studied, bearing in mind that the only light that will travel towards the eye is to be considered. Glassner (1989, p. 130) identifies "four mechanisms of light transport" that can be used. The geometry of these will be detailed in the following section to facilitate the discussion of the behaviour of light upon striking a surface and its treatment in a ray tracing algorithm.

## 5.2 Mechanisms of Light Transport.

### 5.2.1 Specular Reflection.

Specular reflection is observed when looking at a shiny surface. A perfect specular reflector would have a mirror-like quality, where each of the *incident* (incoming) light rays is reflected in a single direction. In the typical instance though, light rays are

reflected in different directions by the surface with varying degrees of intensity depending on the quality of the reflector. Whilst this typical case is difficult to apply because of the inordinate number of post-primary rays that may be cast, approximations to it exist based on the example of a perfect specular reflector.

The direction of reflected light (from a perfect specular reflector) can be found as follows (refer to Figure 5.2.1):

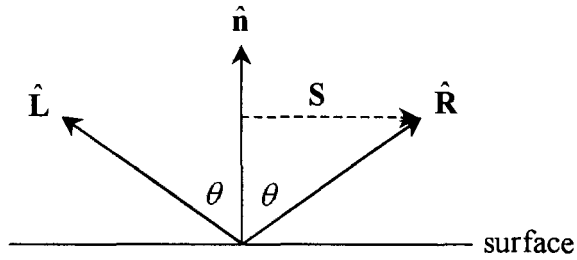



Figure 5.2.1. Perfect specular reflection.

$$\text{Let } \mathbf{S} = \hat{\mathbf{n}}\cos\theta - \hat{\mathbf{L}}$$

$$\text{and let unit vector } \hat{\mathbf{R}} = \hat{\mathbf{n}}\cos\theta + \mathbf{S}$$

$$\text{then } \hat{\mathbf{R}} = 2\hat{\mathbf{n}}\cos\theta - \hat{\mathbf{L}},$$

Now since $\hat{\mathbf{n}}$ and $\hat{\mathbf{L}}$ are unit vectors, $\cos\theta = \hat{\mathbf{n}} \bullet \hat{\mathbf{L}}$ and so:

$$\hat{\mathbf{R}} = 2\hat{\mathbf{n}}(\hat{\mathbf{n}} \bullet \hat{\mathbf{L}}) - \hat{\mathbf{L}}. \qquad (5.2.1)$$

An worked example of the above can be found in Appendix 1. The computations regarding the intensity of light specularly reflected, $I_\lambda^S$, will be discussed in the section detailing an illumination model Section 5.4.

### 5.2.2 Diffuse Reflection.

Matt surfaces such as chalk provide the best examples of diffuse reflectors. Diffusely reflected light is reflected away from the surface equally in all directions.

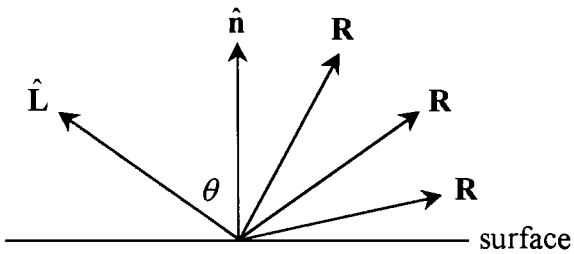



Figure 5.2.2. Perfect diffuse reflection.

The only mathematical principle that need be stated here (without delving too far into surface physics), is that the intensity of the light diffusely reflected, $I_\lambda^D$, is taken to be proportional to the cosine of the angle between the vector $\hat{\mathbf{L}}$ and the outward facing surface normal $\hat{\mathbf{n}}$:

$$I_\lambda^D \propto I_{\lambda L}(\hat{\mathbf{n}} \bullet \hat{\mathbf{L}})$$

### 5.2.3 Specular Transmission.

Transparent surfaces are those which exhibit specular transmission. The surface represents the boundary between two media through which light travels at different speeds. According to the theory presented in much of the literature, a phenomenon known as *refraction*, or the bending of the path of the light, occurs at such a boundary. Consider the following:

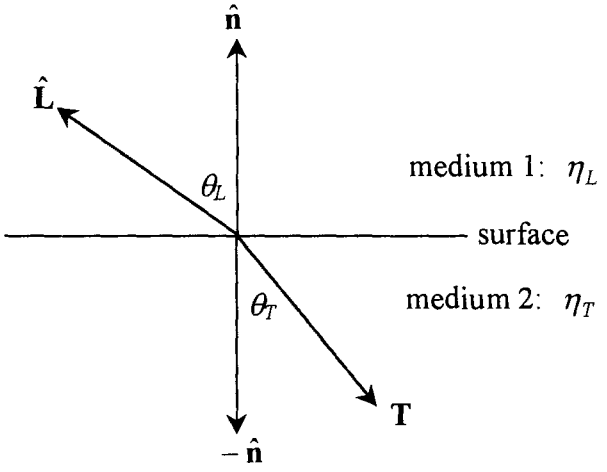



Figure 5.2.3. Specular transmission. $\hat{\mathbf{L}}$ points towards the light source and **T** gives the direction of the transmitted light.

Given that $\eta_L$ is the *index of refraction* of medium 1 (the speed of light through this medium relative to the speed of light through a vacuum) and that $\eta_T$ is the *index of refraction* of medium 2, then the relationship between $\theta_L$ and $\theta_T$ can be written:

$$\frac{\sin\theta_T}{\sin\theta_L} = \frac{\eta_L}{\eta_T} = \eta_{LT}$$

This is known as *Snell's Law*. The vector **T**, representing the direction of transmission, will need to be computed in terms of the vectors $\hat{\mathbf{n}}$ and $\hat{\mathbf{L}}$, and the indices of refraction for the media on either side of the surface. Consider the following diagram:

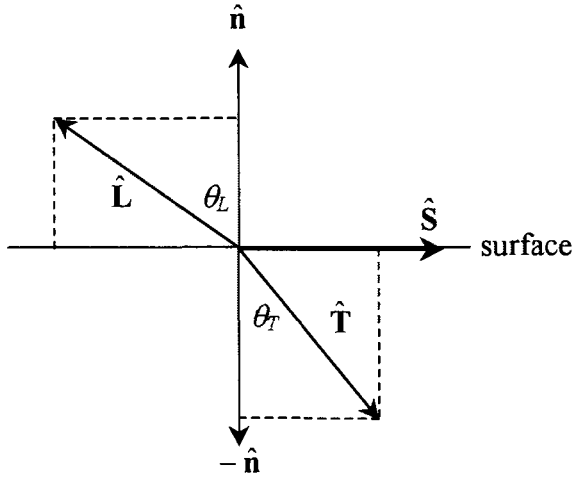



Figure 5.2.3.1. Perfect specular transmission (refraction).

The unit vector $\hat{\mathbf{T}}$ may be expressed as:

$$\hat{\mathbf{T}} = \hat{\mathbf{S}} \sin\theta_T - \hat{\mathbf{n}} \cos\theta_T$$

where $\hat{\mathbf{S}}$ is the unit vector on the surface and in the same plane as $\hat{\mathbf{n}}$ and $\hat{\mathbf{L}}$. $\hat{\mathbf{L}}$ may be written:

$$\hat{\mathbf{L}} = -\hat{\mathbf{S}} \sin\theta_L + \hat{\mathbf{n}} \cos\theta_L$$

which gives:

$$\hat{\mathbf{S}} = \frac{\hat{\mathbf{n}} \cos\theta_L - \hat{\mathbf{L}}}{\sin\theta_L}$$

and so:

$$\hat{\mathbf{T}} = \frac{\sin\theta_T}{\sin\theta_L}(\hat{\mathbf{n}} \cos\theta_L - \hat{\mathbf{L}}) - \hat{\mathbf{n}} \cos\theta_T$$

Rearranging and replacing $\frac{\sin\theta_T}{\sin\theta_L}$ with $\eta_{LT}$ :

$$\hat{\mathbf{T}} = \hat{\mathbf{n}}[\eta_{LT} \cos\theta_L - \cos\theta_T] - \eta_{LT} \hat{\mathbf{L}}$$

Given that $\hat{\mathbf{n}}$ and $\hat{\mathbf{L}}$ are unit vectors, $\cos\theta_L$ and $\cos\theta_T$ can be written as follows:

$$\cos\theta_L = \hat{\mathbf{n}} \bullet \hat{\mathbf{L}}.$$
$$\cos\theta_T = \sqrt{1 - \sin^2\theta_T}$$
$$= \sqrt{1 - \eta_{LT}^2 (1 - \cos^2\theta_L)}$$
$$= \sqrt{1 - \eta_{LT}^2 (1 - (\hat{\mathbf{n}} \bullet \hat{\mathbf{L}})^2)}$$

This provides us with the final expression for **T**:

$$\hat{\mathbf{T}} = \hat{\mathbf{n}}[\eta_{LT} (\hat{\mathbf{n}} \bullet \hat{\mathbf{L}}) - \sqrt{1 - \eta_{LT}^2 (1 - (\hat{\mathbf{n}} \bullet \hat{\mathbf{L}})^2)}\ ] - \eta_{LT} \hat{\mathbf{L}} \qquad (5.2.3)$$

See Example 5.2.3 in Appendix 1 for a numerical illustration of the above.

A phenomenon known as *total internal reflection* may occur when light passing through one medium reaches the boundary with a second medium having a smaller refractive index. Rather than continue at a refracted angle in the next medium, the light is reflected against the boundary between the two media and continues in the original medium.

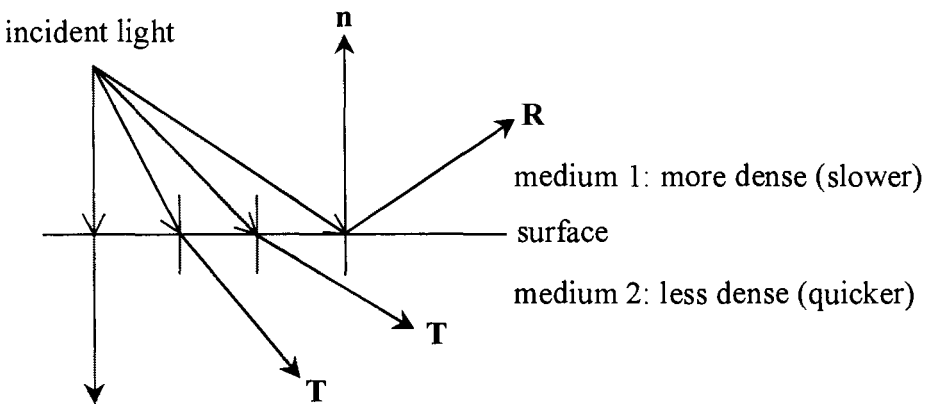



Figure 5.2.3.2. Total internal reflection.

Should equation 5.2.3 yield a complex value for $\hat{\mathbf{T}}$ then total internal reflection is considered to occur. The *criti□al angle*, the angle of incidence of a light ray with the surface below which total internal reflection occurs, may be found by simplifying $1-\eta_{LT}^2(1-(\hat{\mathbf{n}}\bullet\hat{\mathbf{L}})^2)=0$, which gives: $\sin\theta_L=\eta_T/\eta_L$, which is at least reasonable, as $\eta_T/\eta_L<1$ given that $\eta_L>\eta_T$ (light is travelling slower on the side of the light source).

Again, the intensity of light specularly transmitted, $I_\lambda^T$, will be discussed in Section 5.4.

5.2.4 Diffuse Transmission.

A *translucent* material, that allows light to pass through, but colours and scatters it along the way (making objects behind the material appear indistinct), exhibits diffuse transmission. Perfect diffuse transmission would scatter light evenly in all directions as it passed through.

As with the case for diffuse reflection, the intensity of the light diffusely transmitted, $I_\lambda^{DT}$, is taken to be proportional to the cosine of the angle between the vector $\hat{\mathbf{L}}$ and the outward facing surface normal $\hat{\mathbf{n}}$.

5.3 Application of Optical Models to Ray Tracing.

Recall that the goal of the illumination model in the ray tracing algorithm is to colour the pixel through which a particular primary ray was cast into the problem world. Thus the previous expressions for the behaviour of light when it strikes a surface need to be restated and put into the context of the ray tracing algorithm, where only information

regarding light that is eventually received by the eye (i.e., travelling along a primary ray) is of concern.

We may consider two sources of colour information for a particular point on a surface. Primarily, it is the presence of light sources in the problem world that will cast this point in varying degrees of light or shadow, depending on the positions of these sources relative to the point on the surface. Should this same point be on a surface that facilitates either perfect specular reflection or perfect specular transmission, or both, then we can consider the other positions in the problem world as secondary providers of colour information. A single *secondary* ray can be cast in the direction of reflection or transmission to indicate the direction in which this position lies from the point on the surface. This secondary ray may be followed in a similar manner to the primary ray that *spawned* it.

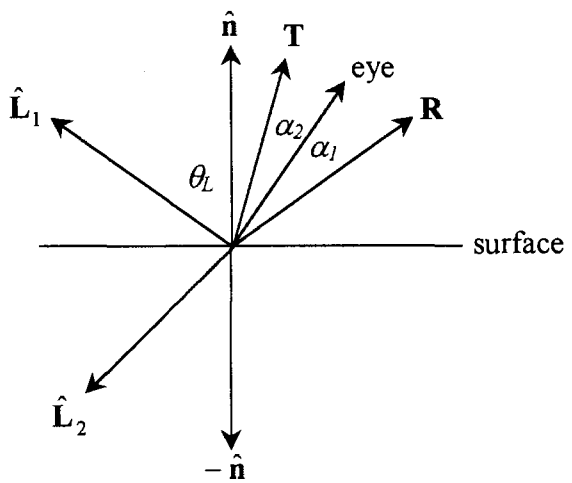



Figure 5.3.1. Illumination of point by light sources.

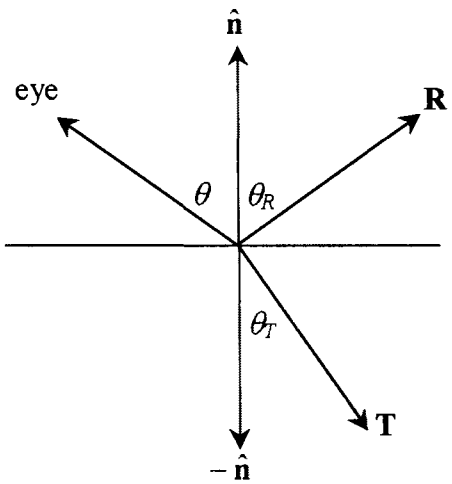



Figure 5.3.2. Casting of secondary rays for a perfect specular surface.

### 5.3.1 Illumination of a Surface at a Point of Intersection.

The extent to which each light source in the problem world lights a particular point is a function of both the angle of incidence a ray cast from that point strikes the surface, and the wavelength of the light. The following will provide expressions for the angle of incidence in terms of the mathematics already identified in the ray tracing process (for the four modes of light transport). These will be determined with reference to the position of the viewer. Note that computations will take place in problem world space, and so the surface normal vector $\hat{\mathbf{n}}$, previously defined in object space, is transformed by matrix $\mathbf{T}_i^{-1}\mathbf{M}_i$. Similarly, the direction in which the eye (or origin of the spawned ray) lies is expressed by the vector $-\mathbf{T}_V^{-1}\mathbf{Vr}_t$ (see section 3.5.1).

#### 5.3.1.1 Illumination by Specular Reflection.

A light source (pointed to by vector $\hat{\mathbf{L}}$) is not typically located in a manner that allows the perfect specular reflection to be seen from the viewing position. As such, perfect specular reflection is of limited use in an illumination model.

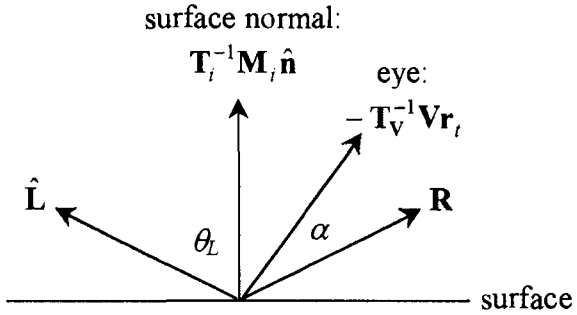



Figure 5.3.1.1. Specular reflection in problem world space.

In the Phong illumination model an imperfect specular reflector is approximated using the angle $\alpha$ between the vector pointing towards the viewer and the reflected vector $\mathbf{R}$. This model assumes that specular reflectance is at a maximum when $\alpha = 0$, and decreases sharply as $\alpha$ increases, according to $\cos^n \alpha$, where $n$ is some number specific to the particular surface. This will be used in Section 5.4, but first we need the following:

Using equation (5.2.1) with unit vector $\hat{\mathbf{m}} = \dfrac{\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}}{\left\|\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}\right\|}$ instead of $\hat{\mathbf{n}}$, $\hat{\mathbf{R}}$ can be stated:

$$\hat{\mathbf{R}} = 2\hat{\mathbf{m}}(\hat{\mathbf{m}} \bullet \hat{\mathbf{L}}) - \hat{\mathbf{L}}$$

The cosine of $\alpha$ can now be written:

$$\cos\alpha = -\frac{\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t}{\left\|\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t\right\|} \bullet \hat{\mathbf{R}}$$

$$= -2\frac{\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t}{\left\|\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t\right\|} \bullet \left( \frac{\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}}{\left\|\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}\right\|} \left( \left( \frac{\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}}{\left\|\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}\right\|} \right) \bullet \hat{\mathbf{L}} \right) - \hat{\mathbf{L}} \right)$$

Example 5.3.1.1 in Appendix 1 illustrates this, for the simplified case without object space / view system transformations.

It may also be noted that the cosine of the angle of incidence of the light, $\theta_L$, is written:

$$\cos\theta_L = \frac{\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}}{\left\|\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}\right\|} \bullet \hat{\mathbf{L}}$$

5.3.1.2 Illumination by Specular Transmission.

In much the same way as illumination by perfect specular reflection was limited to one view point only, so too is illumination by perfect specular transmission. Similarly, imperfect specular transmission is approximated by finding the angle $\alpha$ between the actual and ideal incident rays.

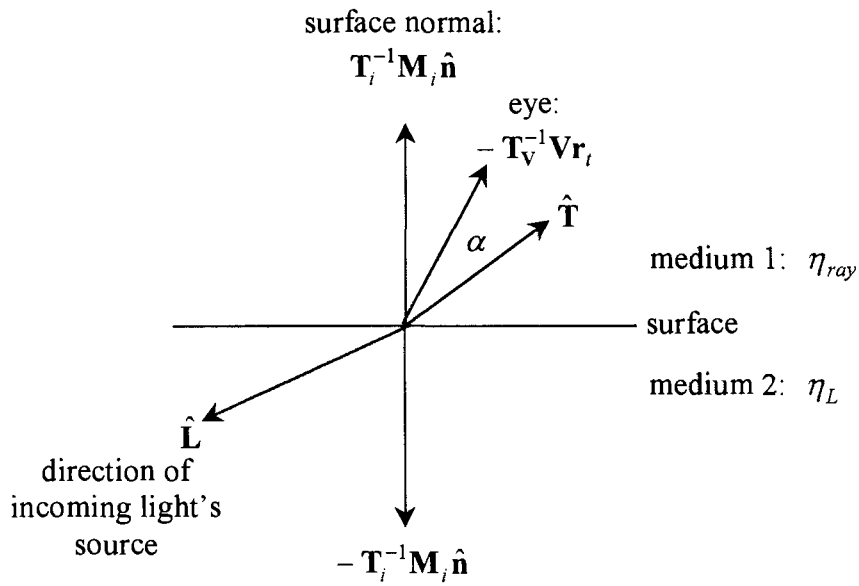



Figure 5.3.1.2. Specular transmission in problem world space.

Using $\hat{\mathbf{m}} = -\dfrac{\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}}{\left\|\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}\right\|}$ in place of $\hat{\mathbf{n}}$, equation (5.2.3) gives the formula for $\hat{\mathbf{T}}$ as:

$$\hat{\mathbf{T}} = \hat{\mathbf{m}}\left[\eta_{L,ray}(\hat{\mathbf{m}} \bullet \hat{\mathbf{L}}) - \sqrt{1 - \eta_{L,ray}^2(1 - (\hat{\mathbf{m}} \bullet \hat{\mathbf{L}})^2)}\ \right] - \eta_{L,ray}\hat{\mathbf{L}},$$

where $\eta_{L,ray}$ is the ratio of the indices of refraction $\eta_L/\eta_{ray}$, with the numerator referring to the medium through which the light passes to reach the surface.

Note that $\mathbf{T}_i^{-1}$ and $\mathbf{T}_V^{-1}$ are the inverse translation matrices for the object space and view system space, and are not related to $\hat{\mathbf{T}}$, which is the vector representing the direction that incoming that would follow if it were to be perfectly specularly transmitted.

With an expression for $\hat{\mathbf{T}}$ in hand, and bearing in mind that the angle made by $\hat{\mathbf{L}}$ with the surface may cause total internal reflection, the cosine of $\alpha$, for use in the Phong illumination model, can now be written using $\cos\alpha = (-\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t) \bullet \hat{\mathbf{T}} / \left\| \mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t \right\|$.

5.3.1.3 Illumination by Diffuse Reflection and Diffuse Transmission.

Diffuse light is reflected and transmitted in all directions (which includes the direction of the viewer) with an equal intensity, this intensity being proportional to the cosine of the angle of incidence:

$$I_\lambda^D \propto I_{\lambda L} \frac{\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}}{\left\| \mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}} \right\|} \bullet \hat{\mathbf{L}}$$

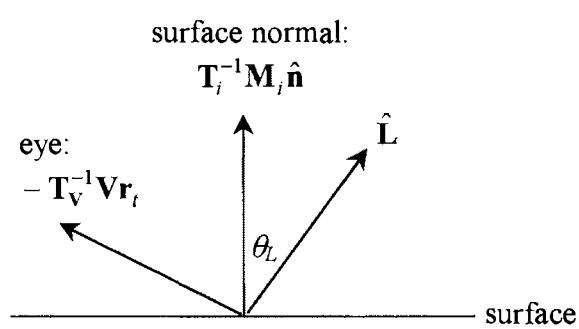



Figure 5.2.2. Diffuse reflection in problem world space.

5.3.2 Recursive Ray Tracing.

Should the surface at a point of intersection with a primary ray have either perfect specular reflection or perfect specular transmission qualities, then a secondary ray is cast to receive information about the part of problem world that can be seen as a result of the reflection or refraction. Should the secondary ray encounter a similar surface, then a tertiary ray is cast, hence the term *recursive* ray tracing. Subsequent rays are treated in the same way as rays before them, up until some arbitrary threshold for the number of rays, or accumulated distance of travel by the light they represent, is reached.

5.3.2.1 Definition of a Specularly Reflected Ray.

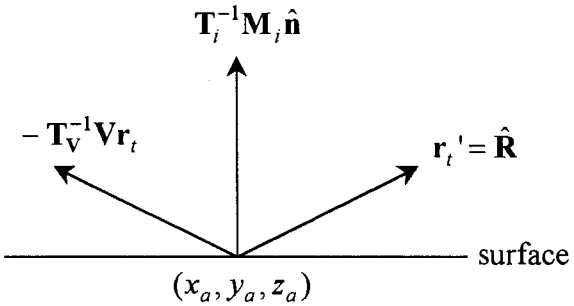



Figure 5.3.2.1. Casting of a subsequent ray due to specular reflection.

Given the ray - surface intersection at $(x_a, y_a, z_a)$, the subsequent ray cast to receive information about the problem world specularly reflected to this point is defined (in problem world space):

$$\mathbf{r}' = [x_a, y_a, z_a]^T + t\mathbf{r}_t'$$

We may obtain an expression for $\mathbf{r}_t'$ from (5.2.1) by replacing

$$\hat{\mathbf{n}} \text{ by } \hat{\mathbf{m}} = \frac{\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}}{\left\|\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}\right\|}, \text{ and } \hat{\mathbf{L}} \text{ by } \hat{\mathbf{k}} = \frac{-\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t}{\left\|\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t\right\|}$$

which gives

$$\mathbf{r}_t' = \hat{\mathbf{R}} = 2\hat{\mathbf{m}}(\hat{\mathbf{m}} \bullet \hat{\mathbf{k}}) - \hat{\mathbf{k}} .$$

See Appendix 1, Example 5.3.2.1 for an application of the above equation.

5.3.2.2 Definition of a Specularly Transmitted (Refracted) Ray.

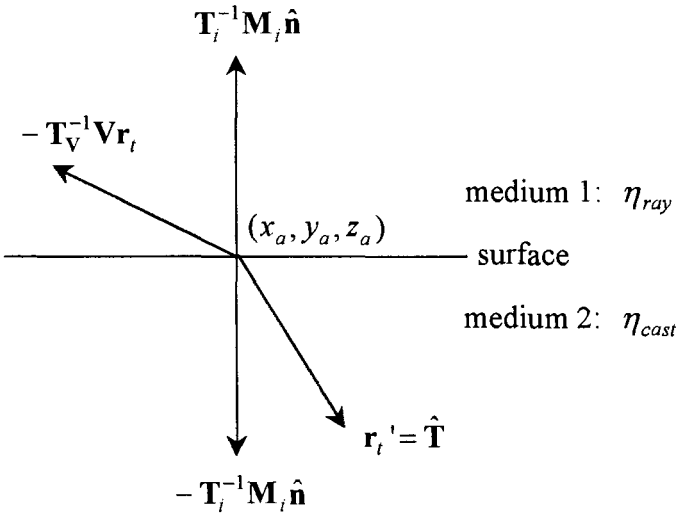



Figure 5.3.2.2. Casting of a subsequent ray due to specular transmission.

A subsequent ray cast to receive information about the problem world specularly transmitted to a point of intersection $(x_a, y_a, z_a)$ is defined (in problem world space):

$$\mathbf{r}' = [x_a, y_a, z_a]^{\mathrm{T}} + t\mathbf{r}_t'$$

We may obtain an expression for $\mathbf{r}_t'$ from (5.2.3) by replacing

$$\hat{\mathbf{n}} \text{ by } \hat{\mathbf{m}} = \frac{\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}}{\left\|\mathbf{T}_i^{-1}\mathbf{M}_i\hat{\mathbf{n}}\right\|}, \text{ and } \hat{\mathbf{L}} \text{ by } \hat{\mathbf{k}} = \frac{-\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t}{\left\|\mathbf{T}_V^{-1}\mathbf{V}\mathbf{r}_t\right\|}$$

which gives

$$\mathbf{r}_t' = \hat{\mathbf{T}} = \hat{\mathbf{m}}\left[\eta_{ray,cast}(\hat{\mathbf{m}} \bullet \hat{\mathbf{k}}) - \sqrt{1 - \eta_{ray,cast}^2(1 - (\hat{\mathbf{m}} \bullet \hat{\mathbf{k}})^2)}\right] + \eta_{ray,cast}\hat{\mathbf{k}}$$

in which $\eta_{ray,cast}$ is the ratio of the indices of refraction $\eta_{ray}/\eta_{cast}$, with the numerator referring to the medium through which the light passes to reach the surface. Example 5.3.2.2. in Appendix 1 illustrates the definition of such a ray.

## 5.4 Illumination Model.

What follows is an example of an illumination model that may be used to generate the colour information for a particular point on a surface. The mathematical content behind it is quite thin, beyond the already indicated references to the geometry of the behaviour of light rays. As such, it is included purely for the sake of completeness.

The illumination equation describes the net intensity of light of a particular wavelength that is evident from the position of the eye. The factors that contribute to this include the properties (with respect to facilitating optical mechanisms for light of certain wavelengths) of the surface at which this value is determined; the intensity, colour (wavelength), and angle of incidence of light illuminating the point; and the angle from which the surface is viewed.

The notation used is, in part, from Foley et al. (1994, pp. 478-489).

### 5.4.1 Surface Characteristics.

In this basic model we may identify two variables that influence the illumination of a surface. A constant $k$ representing the surface's ability to facilitate the mechanics of light transport at that point $(0 \leq k \leq 1)$ is used in conjunction with $O_\lambda$, the surface's *colour* component of wavelength $\lambda$, to determine the amount of light of that

wavelength which enters the calculations. The intensity of illumination then, is based on these and the intensity and direction of incident light:

$$I_\lambda = I_{\lambda L} k O_\lambda F(\theta)$$

A more adaptable model is described by considering that a surface has independent variables representing its ability to reflect light, both diffusely and specularly, and transmit light (for which the diffuse case will be addressed separately):

$$I_\lambda = I_\lambda^D + I_\lambda^S + I_\lambda^T$$

$$I_\lambda = I_{\lambda L}\left[\left[k_d O_{d\lambda}(\hat{\mathbf{n}} \bullet \hat{\mathbf{L}})\right] + O_{s\lambda}\begin{cases} k_s \cos^n \alpha_s, \text{if source of } \hat{\mathbf{L}} \text{ is 'in front of surface'} \\ k_t \cos^n \alpha_t, \text{if source of } \hat{\mathbf{L}} \text{ is 'behind the surface'} \end{cases}\right] \quad (5.4.1)$$

where $I_{\lambda L}$ is the light source intensity. Note that the diffuse reflection component is proportional to the angle of incidence of the light, and that the intensity of specular reflection and transmission components relies on the Phong illumination model, which considers $\alpha$, the angle between the direction of perfect specular transport and the direction from which the surface is viewed.

### 5.4.2 Light Source Attenuation.

Light source attenuation describes the decrease in light energy when a light source is moved away from a surface. The expression is a function of the distance $d_L$ between the source and the point that is being illuminated:

$$f_{att\,L} = \min\left(\frac{1}{c_3 d_L^2 + c_2 d_L + c_1}, 1\right)$$

where $c_3, c_2, \text{and } c_1$ are constants associated with the light source. Applied to illumination equation 5.4.1 we obtain:

$$I_\lambda = f_{att\,L} I_{\lambda L} \left[ \left[ k_d O_{d\lambda} (\hat{\mathbf{n}} \bullet \hat{\mathbf{L}}) \right] + O_{s\lambda} \begin{cases} k_s \cos^n \alpha_s, & \hat{\mathbf{L}} \text{ in front} \\ k_t \cos^n \alpha_t, & \hat{\mathbf{L}} \text{ behind} \end{cases} \right]$$

5.4.3 Multiple Light Sources and Shadows.

Multiple light sources are simply accounted for by the summation of the illumination equations attributable to each for the given point on the surface. However, some sources may be occluded from that particular point, and as such should not contribute to the overall equation. Multiple light sources may be modelled as follows:

$$I_\lambda = \sum_{l=1}^{m} S_l f_{att\,l} I_{\lambda l} \left[ \left[ k_d O_{d\lambda} (\hat{\mathbf{n}} \bullet \hat{\mathbf{L}}_l) \right] + O_{s\lambda} \begin{cases} k_s \cos^n \alpha_s, & \hat{\mathbf{L}}_l \text{ in front} \\ k_t \cos^n \alpha_t, & \hat{\mathbf{L}}_l \text{ behind} \end{cases} \right]$$

where $m$ is the number of light sources illuminating the scene, and $S_l$ is the shadow term, defined as:

$$S_l = \begin{cases} 0, \text{if light source } l \text{ is obscured} \\ 1, \text{if light source } l \text{ is not obscured} \end{cases}$$

5.4.4 Ambient Lighting.

Ambient light is typically considered to be a result of general diffuse interactions not described by the model, including diffusely transmitted light from light sources that have been defined, and light from sources external to the problem world (for example, the sky). Surfaces are illuminated by ambient light of intensity $I_\lambda^A$ according to the term $k_a$ $(0 \le k_a \le 1)$ and their diffuse colour $O_{d\lambda}$. The 'complete' illumination equation can now be described:

$$I_\lambda = I_\lambda^A k_a O_{d\lambda} + \sum_{l=1}^{m} S_l f_{att\,l} I_{\lambda l} \left[ \left[ k_d O_{d\lambda} (\hat{\mathbf{n}} \bullet \hat{\mathbf{L}}_l) \right] + O_{s\lambda} \begin{cases} k_s \cos^n \alpha_s, & \hat{\mathbf{L}}_l \text{ in front} \\ k_t \cos^n \alpha_t, & \hat{\mathbf{L}}_l \text{ behind} \end{cases} \right]$$

## 5.5 Surface Detail.

In the previous discussion each surface was identified as having a single colour parameter (for each mode of light transport) that was uniform across its entirety. This section will look at a number of mappings that can add detail to the appearance of a surface.

### 5.5.1 Texture Mapping.

A *texture map* is usually, but not necessarily, a two-dimensional, rectangular image that is to be placed on some surface in the problem world. An example would be a digitised photograph, stored as a matrix of coloured pixels and indexed by the parameters $\mu_t$ and $\nu_t$, to be placed on some facet representing a picture in a frame in the problem world.

For the purposes of ray tracing the texture mapping must be invertible: i.e., given a point of ray intersection on a textured surface $(x, y, z)$, the corresponding texture space parameters $\mu_t$ and $\nu_t$ can be found. The following describes the mapping from a rectangular texture space to an arbitrary quadrilateral $\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2\mathbf{P}_3$.

The position vector $\mathbf{P} = [x, y, z]^T$ of a point on the quadrilateral corresponding to texture parameters $\mu_t$ and $\nu_t$ is given by a modified version of the equation given on page 55.

$$\mathbf{P} = \mathbf{P}_0 + \mu_t[(1-\nu_t)(\mathbf{P}_1 - \mathbf{P}_0) + \nu_t(\mathbf{P}_2 - \mathbf{P}_3)] + \nu_t(\mathbf{P}_3 - \mathbf{P}_0) \qquad (5.5.1.1)$$

where $0 \le \mu_t \le 1$, and $0 \le \nu_t \le 1$.

Note that $\mathbf{P}$ is given in terms of a displacement from the first corner $\mathbf{P}_0$. This displacement is labelled $\mathbf{p}$, where $\mathbf{p} = \mathbf{P} - \mathbf{P}_0$, which gives:

$$\mathbf{p} = \mu_t[(1-\nu_t)(\mathbf{P}_1 - \mathbf{P}_0) + \nu_t(\mathbf{P}_2 - \mathbf{P}_3)] + \nu_t(\mathbf{P}_3 - \mathbf{P}_0) \qquad (5.5.1.2)$$

where

$$\mathbf{p} = [p_x, p_y, p_z]^{\mathrm{T}}$$

The corresponding texture space parameters can be identified by solving any two of the three equations represented by equation (5.5.1.2). Representing the vector differences $(\mathbf{P}_1 - \mathbf{P}_0)$, $(\mathbf{P}_3 - \mathbf{P}_0)$ and $(\mathbf{P}_2 - \mathbf{P}_3)$ by **a**, **b**, and **d** respectively, equation (5.5.1.2) is simplified:

$$\begin{aligned}\mathbf{p} &= \mu_t[(1-\nu_t)\mathbf{a} + \nu_t\mathbf{d}] + \nu_t\mathbf{b} \\ &= \mu_t\nu_t(\mathbf{d} - \mathbf{a}) + \mu_t\mathbf{a} + \nu_t\mathbf{b}\end{aligned}$$

Solving the equations $p_x$ and $p_y$ for $\mu_t$ and $\nu_t$, the general inversion is obtained:

$$\mu_t = \frac{-a_2 \pm \sqrt{a_2^2 - 4a_1a_3}}{2a_1}, \quad 0 \le \mu_t \le 1$$

where $a_1 = \left(-a_x(d_y - a_y) + a_y(d_x - a_x)\right)$

$$a_2 = \left(a_yb_x - a_xb_y + p_x(d_y - a_y) + p_y(d_x - a_x)\right)$$

$$a_3 = p_xb_y - p_yb_x$$

and $\nu_t = \dfrac{p_x - \mu_t a_x}{\mu_t(d_x - a_x) + b_x}$

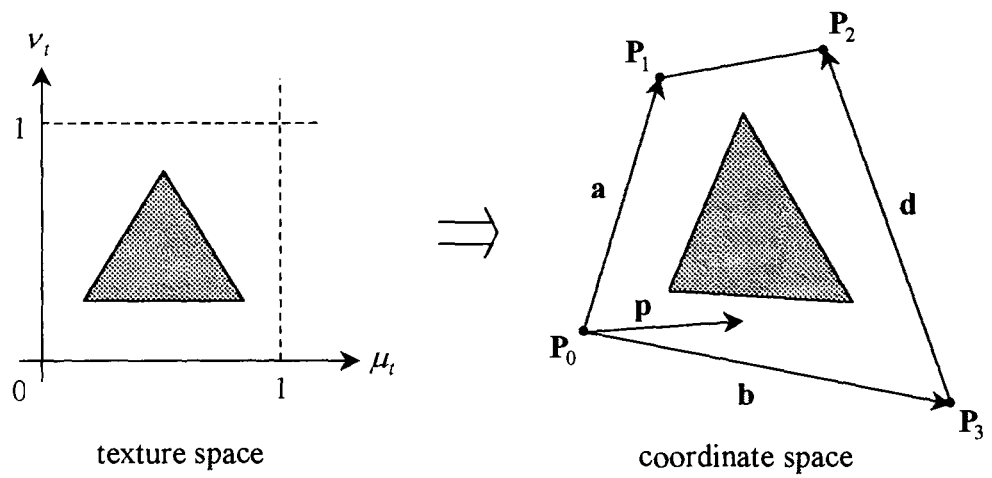



Figure 5.5.1.1. Texture Mapping to a quadrilateral.

As one would expect, the case for which $\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2\mathbf{P}_3$ is a parallelogram simplifies the computations somewhat $(\mathbf{a} = \mathbf{d})$:

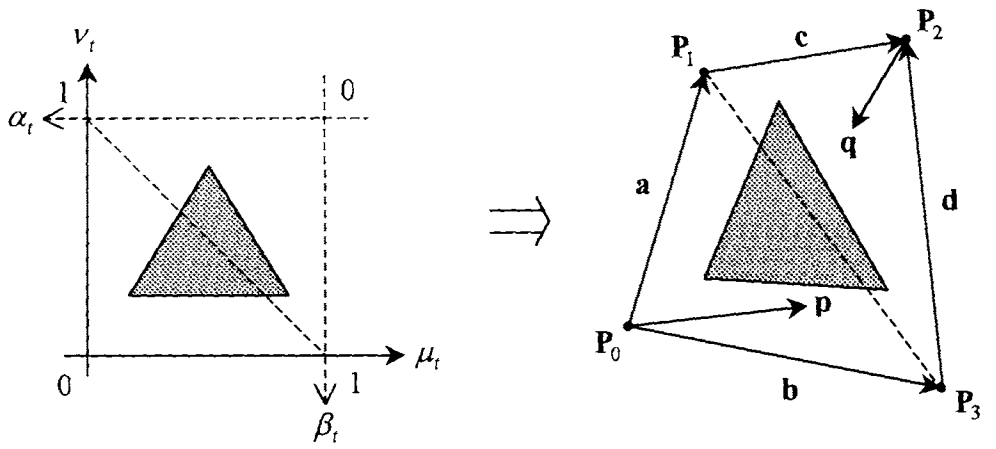
$$\mathbf{p} = \mu_t \mathbf{a} + v_t \mathbf{b}$$

The general inversion is stated:

$$\mu_t = \frac{p_x b_y - p_y b_x}{a_y b_x - a_x b_y}$$

$$v_t = \frac{p_x - \mu_t a_x}{b_x}$$

As with the specification of faceted surfaces, matters are simplified somewhat when triangles are used:




texture space coordinate space

For triangles there are two mappings:

$$\mathbf{p} = v_t\mathbf{a} + \mu_t\mathbf{b} \text{ for } \mu_t, v_t \geq 0, \text{and } 0 \leq \mu_t + v_t \leq 1,$$

$$\mathbf{q} = -\alpha_t\mathbf{c} - \beta_t\mathbf{d} \text{ for } \alpha_t, \beta_t \geq 0, \text{and } 0 \leq \alpha_t + \beta_t \leq 1.$$

where $\mathbf{p}$ is the displacement from $\mathbf{P}_0$ and $\mathbf{q}$ is the displacement from $\mathbf{P}_2$. Both of these formulations are simply invertible.

Textures can be mapped to any parametrically defined surface. For example, a mapping to a bi-cubic parametric patch is trivially done through substitution of texture space parameters for the surface parameters. A mapping to a cylinder would most likely assign texture space parameter $v_t$ to the height of the cylinder, and $\mu_t$ to the angle of revolution about the cylinder's axis.

5.5.1.2 Bump Mapping.

Bump mapping is similar to texture mapping in that detail is added to an otherwise uniform surface through a mapping to coordinate space. However, rather than map an array of colours to the surface, an array of surface normal transformations are mapped that provide variations in the illumination of the surface. For example, bump mapping can be used to simulate the dimples on a golf ball by altering the surface normal of a sphere in a regular patterned manner. Whilst the technique brings little to the discussion of mathematics behind image generation, it is a useful short cut that is commonly applied.

## 5.6 Approximation of a Smooth Surface through Phong Shading.

Consider a triangular facet, with normal $\hat{\mathbf{n}}$, that is defined by vertices which were sampled from a smooth object. If it is anticipated that a number of rays will intersect with the facet, the surface normal may be adjusted at the points of intersection, which will in turn mean that the illumination of the approximated surface is closer to that of the original form. Phong (1975) provides a method to accomplish this.

Phong shading linearly interpolates a surface normal according to the (normalised) vector sum of the normal vectors of that surface and the adjacent surfaces at each vertex (care must be taken not to sum vertices at a vertex running along a deliberately discontinuous edge, otherwise unwanted smoothing may occur). More formally the surface normal is defined as follows.

For the triangular facet $\mathbf{P}_0\ \mathbf{P}_1\ \mathbf{P}_2$ the approximated surface normal $\mathbf{n}^*$ at the point $\mathbf{P} = \mathbf{P}_0 + \mu(\mathbf{P}_1 - \mathbf{P}_0) + \nu(\mathbf{P}_2 - \mathbf{P}_0)$ is given by:

$$\mathbf{n}^* = \hat{\mathbf{n}}_0 + \mu(\hat{\mathbf{n}}_1 - \hat{\mathbf{n}}_0) + \nu(\hat{\mathbf{n}}_2 - \hat{\mathbf{n}}_0)$$

where $\hat{\mathbf{n}}_i$ is the unit vector in the direction of $\mathbf{n}_i$, the vector sum of the unit normals to the facets (that we are interested in) meeting at vertex *i*.

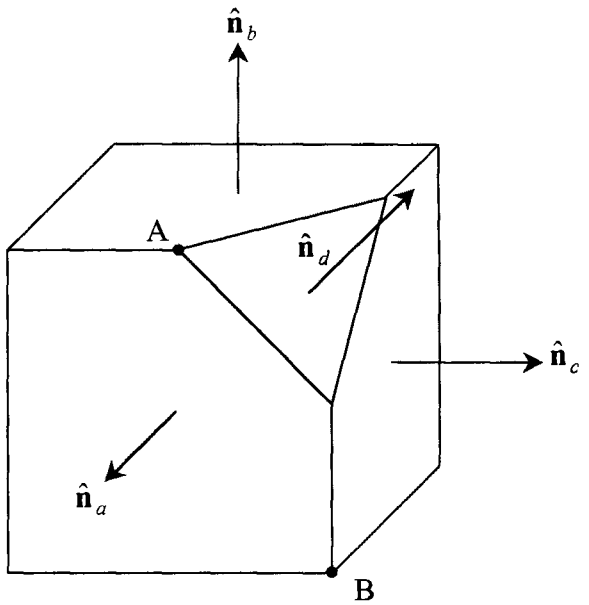



Figure 5.6. A cube with a 'rounded' edge.

Consider figure 5.6, where a corner of a cube has been removed in an attempt to model some sort of curvature. Close to vertex A, one would assume that the Phong shading method would use the 'average normal' vector $\hat{\mathbf{n}}_A$, which has the same direction as $\mathbf{n}_A = \hat{\mathbf{n}}_a + \hat{\mathbf{n}}_b + \hat{\mathbf{n}}_d$. However, close to vertex B, where rounding is not wanted (a straight edge is desired), an average normal vector would not be designated, instead shading on each facet occurring irrespective of the adjacent facets.

## 6. Other Methods of Image Generation.

A primary inefficiency of the ray tracing algorithm discussed in Chapter 3 is the arbitrary casting of (at least) one ray through each pixel on the view window. The method does not consider the construction of the problem world and would be improved if the direction of each ray cast could be determined by the presence of detail in the problem world. By considering the positions of vertices and edges in the problem world we identify not only a means (albeit an over-simplified one, given the depth of research into the area) of improving the efficiency of the ray tracing algorithm, but also a general method of two-dimensional image generation.

In Section 3.4 a number of projection methods were discussed in the definition of the general form of a ray, without any reference to the problem world into which they were to be cast. In this section, the special cases of these projection methods will be considered in which, depending on the method used, rays passing through a number of vertices in the problem world are cast in order to describe the appropriate image on the view window.

The resulting image consists of a set of points on the view window, which are then joined by computer algorithm to define edges, and in turn used to identify facets that can be shaded. The removal of vertices, edges and facets that are hidden or out of view from the consideration of the image generation process is a task that can occur before or during shading of surfaces. There is much literature describing such image generation processes, and with an almost exclusively computer programming orientation (especially with regards to the hidden point / line / surface removal steps) the associated

algorithms will not be discussed in any further detail by this project, except where there is suitable mathematical content.

## 6.0 Review of Notation.

The following discussion will look at different types of projection. But first we briefly remind the reader of some of the notation used in the definition of a view system as detailed in Section 3.2. The form in which a ray is expressed (Sections 3.3 to 3.4) is also reviewed.

The axes of the view system are labelled $(u, v, n)$ and view system space is positioned in the problem world by the transformation $\mathbf{V}$. Hence, when working in view system space, object coordinates need to be first transformed into problem world coordinates and then into view system coordinates: this composite transformation is represented by the matrix product $\mathbf{V}^{-1}\mathbf{M}_i$ (for object $i$).

The view plane is defined in view system space $(u, v, n)$ by $n = k$, with normal vector $\hat{\mathbf{n}} = [0, 0, -1]^{\mathrm{T}}$, and the view window as the rectangle with bounding vertices $(-a, b, k)$, $(a, b, k)$, $(a, -b, k)$ and $(-a, -b, k)$. The 'observer' is typically located at $(0, 0, 0)$, and 'looks' through a view window lying on the plane $k = -1$. For the purposes of the following sections, a point on the view window will not be referred to by a point on the pixel $(i, j)$ as previously, but rather the point $(A, B, k)$ on the plane of the window itself will be used. As such, $-a \le A \le a$ and $-b \le B \le b$ define the points on the view window. The equation $\mathbf{r}^{A,B} = \mathbf{r}_0^{A,B} + t\mathbf{r}_t^{A,B}$ describes the ray that passes

through the view window at the point $(A,B,k)$, where $\mathbf{r}_0^{A,B}$ and $\mathbf{r}_t^{A,B}$ are defined according to the particular projection method to be used.

As stated throughout Chapter 3, ray traced projection is implemented by tracing the paths of an arbitrary number of rays through the view window into the problem world. Now consider that the points of interest in the problem world are already known, and we wish to examine their *projection* on to the view plane, more particularly any projections on to the view window, which also lies in the problem world. The following sections discuss identifying the point $(A,B,k)$ on the view window for the ray $\mathbf{r}^{A,B}$ that passes through an already identified point in the problem world.

## 6.1 Parallel Projection.

A parallel projection is defined by a set of rays that have parallel direction vectors, i.e. have the same vector $\mathbf{r}_t$ in each of their equations.

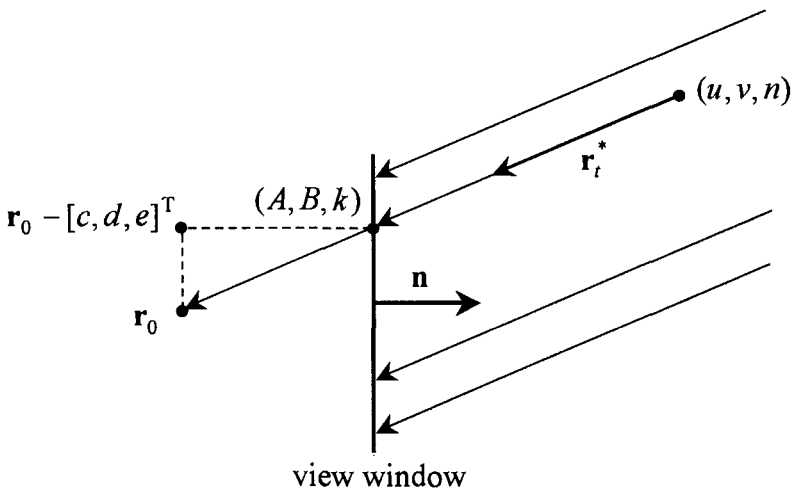



Figure 6.1. Rays and view window of a parallel projection.

Recall that a parallel projection is not restricted to the direction of the normal vector to the view window. This direction is denoted from the point $(u,v,n)$ in the problem world as $\mathbf{r}_t^* = [c,d,e-k]^{\mathrm{T}}$, where the vector $[c,d,e]^{\mathrm{T}}$ represents the displacement of the origin of a ray that would have passed perpendicularly through the view plane (the view plane is typically located with centre $(0,0,-k)$, or by position vector $[0,0,-k]^{\mathrm{T}}$, hence the vector addition in $\mathbf{r}_t^*$ ).

So for any point $(u,v,n)$ in the view space representation of the problem world, the following equations describe the projection by ray $\mathbf{r}^{A,B}$ to the point $(A,B,k)$ on the view window:

$$\begin{bmatrix} u \\ v \\ n \end{bmatrix} = \begin{bmatrix} A \\ B \\ k \end{bmatrix} + t \begin{bmatrix} -c \\ -d \\ k-e \end{bmatrix}, \text{ so: } \quad \begin{aligned} A &= u + tc && (6.1.1) \\ B &= v + td && (6.1.2) \\ k &= n + t(e-k) && (6.1.3) \end{aligned}$$

Equation (6.1.3) provides the solution for $t$ when the ray passes through the view plane:

$$t = \frac{k-n}{e-k} \qquad (6.1.4)$$

Using result (6.1.4) in (6.1.1) and (6.1.2) we obtain expressions for $A$ and $B$, which lead to the matrix representation, **L**, of the parallel projection operation:

$$\begin{bmatrix} A \\ B \\ k \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\frac{c}{e-k} & \frac{ck}{e-k} \\ 0 & 1 & -\frac{d}{e-k} & \frac{dk}{e-k} \\ 0 & 0 & 0 & k \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ n \\ 1 \end{bmatrix}$$

$$[A,B,k,1]^{\mathrm{T}} = \mathbf{L}\,[u,v,n,1]^{\mathrm{T}}$$

For example, consider the trivial case where the view plane is located at $n = -1$, and the origin of the rays has not been displaced, that is the projection is perpendicular to the view plane, or $c = d = e = 0$. **L** then reduces to:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

which means that under this projection $[A, B, -1, 1]^T = [u, v, -1, 1]^T$, indicating that the $n$ term of a point in the (view system representation of) problem world is simply replaced with $k = -1$, the $n$ term of the view plane.

6.2 Perspective Projection.

Ray traced perspective projection is implemented by tracing the paths of rays originating from the eye, which is usually denoted at the origin of the view system, through the view window into the problem world. To find the position $(A, B, k)$ on the view window passed through by such a vector intersecting point $(u, v, n)$, a comparison of similar triangles may be used.

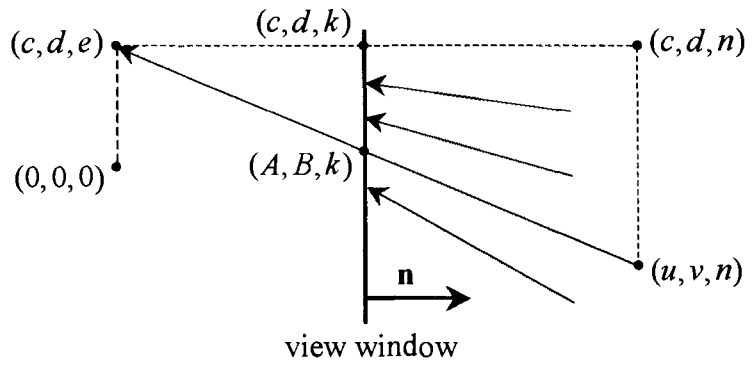



Figure 6.2. Rays and view window of a perspective projection.

Note that this describes the general perspective projection case, where the eye has been translated to $(c,d,e)$. What is required is an expression for $(A,B,k)$ in terms of $(u,v,n)$. By similar triangles (comparison of perpendicular sides):

$$\frac{k-e}{n-e}=\frac{A-c}{u-c} \quad \text{and} \quad \frac{k-e}{n-e}=\frac{B-d}{v-d}$$

Rearranging these in terms of the required variable:

$$A=\frac{(k-e)(u-c)}{(n-e)}+c \quad \text{and} \quad B=\frac{(k-e)(v-d)}{(n-e)}+d \qquad (6.2.1)$$

By factoring each right hand side for $\frac{1}{n-e}$, the perspective projection may be expressed in matrix form, **R**:

$$\begin{bmatrix} A \\ B \\ k \\ W \end{bmatrix} = \frac{1}{n-e} \begin{bmatrix} k-e & 0 & c & -ck \\ 0 & k-e & d & -dk \\ 0 & 0 & k & -ek \\ 0 & 0 & 1 & -e \end{bmatrix} \begin{bmatrix} u \\ v \\ n \\ 1 \end{bmatrix}$$

$$[A,B,k,1]^{\mathrm{T}} = \frac{1}{n-e}\mathbf{R}[u,v,n,1]^{\mathrm{T}}$$

Further understanding of the above comes through the consideration of a matrix $\mathbf{R}^*$ defined by equation (6.2.1) but without the factoring out the expression $1/(n-e)$. In such a case, the fourth entry of the resulting coordinates $[A^*, B^*, k^*, W]^T$ might not necessarily be equal to one. Thus, to complete the projection, the coordinates need to be reverted to the form $[A, B, k, 1]^T$, by dividing through by W:

$$[A, B, k, 1]^T = \frac{1}{W}[A^*, B^*, k^*, W]^T = \frac{1}{n-e}\mathbf{R}^*[u, v, n, 1]^T$$

6.3 Implications of Projecting on to the View Plane.

When projecting on to the view plane a number of difficulties emerge that are a result of attempting to state explicitly the points of interest in a scene. Hidden surface and line detection is a key area of concern, with mathematical concepts such as identifying volumes of occlusion behind foreground objects, or viewing volumes that represent the total space in the problem world that could be seen through the view window.

Whereas the ray tracing method of sampling the problem world for incoming light rays relied solely on line – surface intersection computations, the above methods require that if a surface – surface intersection is present and suspected to be visible, it must be computed and eventually be represented explicitly as a set of points.

6.4 Gouraud Shading.

Once the visible surface algorithms have specified the extent to which each surface is visible, the process of shading can begin. In the most basic shading routines the entire

surface is illuminated using its normal, which produces images with a faceted appearance.

Since each projected vertex is from some object in the problem world, the colour and illumination of the surface at each vertex may be computed. The normal at each vertex, $\mathbf{n}_i$, is dependent on the adjoining surfaces' (intended) continuity to the particular surface being shaded. From Section 5.6:

$$\mathbf{n}_i = \sum_{j=1}^{k} \hat{\mathbf{n}}(j),$$

where $k$ is the number of facets, with unit normals $\hat{\mathbf{n}}(j)$, each cornered by vertex $i$, that are considered contiguous to the facet in question. Using the computed normal, use can be made of the illumination models of Section 5 to provide an intensity $I_{\lambda i}$ at each vertex.

Gouraud shading (Gouraud, 1971) linearly interpolates the intensities of the vertices about a (triangular) facet. The approximated intensity of light on the surface of a triangular facet $\mathbf{P}_0$ $\mathbf{P}_1$ $\mathbf{P}_2$ as a result of this technique at the point $\mathbf{P} = \mathbf{P}_0 + \mu(\mathbf{P}_1 - \mathbf{P}_0) + \nu(\mathbf{P}_2 - \mathbf{P}_0)$ is given by:

$$I_{\lambda G} = I_{\lambda 0} + \mu(I_{\lambda 1} - I_{\lambda 0}) + \nu(I_{\lambda 2} - I_{\lambda 0})$$

where $\mu, \nu \geq 0$ and $0 \leq \mu + \nu \leq 1$.

It is noted that the Phong illumination model is equally well suited to this task, although takes more time as the illumination model is run through for each interpolation.

## 7. Conclusion.

It was noted in the introduction that conventional theory dictates that it is the light within an environment that is received by the eye and interpreted as vision. As such, computer graphics applications that aim to communicate visually some digitally stored environment apply models simulating the behaviour and interaction of light with an environment. The ray tracing algorithm was cited as an implementation of some of these light behaviour models.

This thesis presented an outline of some the mathematical concepts that find application in the ray tracing algorithm, one solution to the image generation problem. In particular, vector geometry was applied to the understanding that incoming light rays from an environment provide the visual information necessary to render the scene in two-dimensions. This provided a mathematical framework (the view system) through which an arbitrary problem world could be viewed.

Whilst the ray tracing algorithm provides an elegant means of image generation, it is a costly computational procedure. As alluded to in Section 6, the scope for the optimisation of the basic technique lies with identifying that the procedure essentially samples the problem world before it. Thus, statistical methods have found much use in the practical implementation of ray tracing (Glassner, 1989. p. 24). The special case was presented (Section 6.2) where only the vertices of interest are sampled from the problem world, using a perspective projection transformation. The reader familiar with computer graphics will identify this as the core of the method by which the real time rendering of graphics is accomplished.

The content of this thesis could be understood to be representative of a fair proportion of elementary image generation concepts, certainly all of which have been many times considered and presented in more detail than shown here. As such this should be considered to provide at best a mathematical discussion of some computer graphics topics that are perhaps usually presented with only their implementation in mind.

## 8. References.

Burger, P., & Gillies, D. (1989). Interactive Computer Graphics. Wokingham, England: Addison-Wesley.

Phong, Bui-Tuong (1975) Illumination for Computer Generated Images. Communications of the ACM, 18(6), 311-317.

Farin, G. (1990). Curves and Surfaces for Computer Aided Geometric Design. Boston: Academic Press.

Foley, J. D., van Dam, A., Feiner, S. K., et al. (1994). Introduction to Computer Graphics. Reading, Mass.: Addison-Wesley.

Glassner, A. S. (1989). An Introduction to Ray Tracing. London: Academic Press.

Gouraud, H. (1971). Computer Display of Curved Surfaces. IEEE Transactions on Computers, 20, 623-628.

### 8.1 Bibliography.

Anton, H. (1991). Elementary Linear Algebra, 6th ed. New York: John Wiley

Farin, G. E. (1995). NURB curves and surfaces: from projective geometry to practical use. Wellesley, Mass.: A.K. Peters.

Foley, J.D., & van Dam, A. (1982). Fundamentals of Interactive Computer Graphics. Reading, Mass.: Addison-Wesley.

Foley, J. D., van Dam, A., Feiner, S. K., et al. (1990). Computer Graphics: Principles and Practice. Reading, Mass.: Addison-Wesley.

Gerald, C. F., & Wheatley, P. O. (1994). Applied Numerical Analysis, 5th ed. Reading, Mass.: Addison-Wesley.

Glassner, A. S., ED. (1990). Graphics Gems. San Diego, CA: Academic Press.

Hall, R. (1989). Illumination and Colour in Computer Generated Imagery. New York: Springer-Verlag.

Hearn, D., & Baker, M.P. (1996). Computer Graphics (2nd Ed.). New Jersey: Prentice Hall.

Hoggar, S. G. (1993). Mathematics for Computer Graphics. Cambridge: Cambridge University Press.

Newman, W. M., & Sproull, R. F. (1978). Principles of Interactive Computer Graphics 2nd ed. New York: McGraw-Hill.

Penna, M. A., & Patterson, R. (1986) Projective Geometry and its applications to Computer Graphics New Jersey: Prentice-Hall.

Rogers, D. F., (1985). Procedural Elements for Computer Graphics. New York: McGraw-Hill.

Salmon & Slater (1987). Computer Graphics Systems and Concepts. Wokingham, England: Addison-Wesley.

Wylie, C. R. (1970). Introduction to Projective Geometry. New York: McGraw-Hill.

Young, J. W. (1971). Projective Geometry. Chicago: Open Court for The Mathematical Association of America.

Zobrist, G. W., & Sabharwal, C. (1992). Progress in Computer Graphics Volume 1. Norwood, NJ: Ablex Publishing Co.

## 8.2 Some Other Relevant Material.

Karsten, Isakovic (1997). The 3D Engines List[online]. Available WWW:
http://cg.cs.tu-berlin.de/~ki/engines.html

Provides a list of 3-d engines for home computers compiled by Isakovic Karsten from Technical University of Berlin. - for applications both commercial and domestic.

Hammersleys, Tom (1997). Graphics Coding Page[online]. Available WWW:
http://www.users.globalnet.co.uk/~tomh/

Explains some of the mathematics behind some problems commonly faced by amateur graphics programmers.

Holten-Lund, Hans (1997). Bookmarks[online]. Available WWW:
http://www.id.dtu.dk/~hahl/hotlist.html

A comprehensive list of links to many graphical resources on the internet.

http://www.siggraph.org/

ACM SIGGRAPH (Special Interest Group on Computer Graphics)
"SIGGRAPH is about the exchange of ideas among researchers and technology developers through our conferences, publications to advance the technology of computer graphics and interactive techniques."

Spencer, Stephen (1996) ACM SIGGRAPH Online Bibliography Database[online].
Available WWW: http://www.siggraph.org/publications/bibliography/bibliography.html

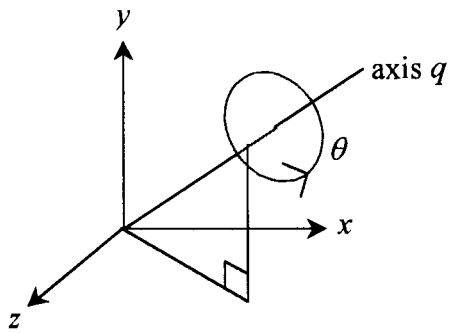
ACM SIGGRAPH Online Bibliography Database.
This online bibliography database is a collection of over fourteen thousand unique computer graphics references.

# APPENDIX 1

Worked Examples.

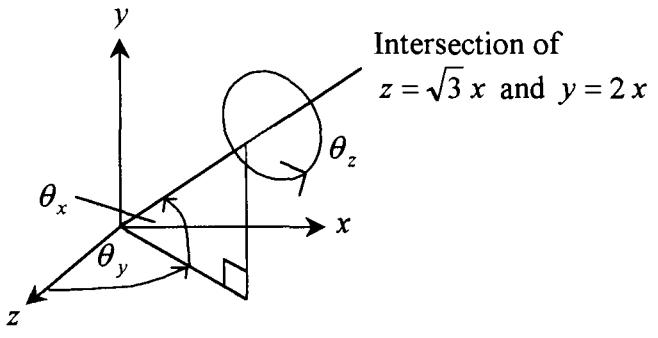
**Example 2.4.5.2:** Rotation about an arbitrary axis running through the origin.

Consider an anticlockwise rotation of $\theta$ about an arbitrary axis $q$ that runs through the origin:




This example identifies the matrix transformation of the form $\mathbf{R} = \mathbf{R}_y\mathbf{R}_x\mathbf{R}_z\mathbf{R}_x^{\mathrm{T}}\mathbf{R}_y^{\mathrm{T}}$ that performs the above rotation.

Suppose that the line that is the intersection of the planes $z = \sqrt{3}\,x$ and $y = 2\,x$ lies on the axis $q$. This axis may now be described in terms of some rotation of space about the $y$ axis (by $\theta_y$) and then about the $x$ axis (by $\theta_x$), which transforms a line running along the $z$ axis into the areforementioned line along $q$. Furthermore, the anticlockwise rotation of $\theta_z$ about the $z$ axis prior to the above rotations facilitates the anticlockwise rotation of $\theta$ about $q$, if $\theta_z = \theta$.




Given that the line is the intersection of the planes $z = \sqrt{3}\,x$ and $y = 2x$, the angles $\theta_y$ and $\theta_x$ can be obtained. Note that the rotation about the $x$ axis is in a clockwise direction (or from the $z$ axis to the $y$ axis), and so the angle $\theta_x$ is negative.

$$\tan\theta_y = \frac{x}{z} = \frac{x}{\sqrt{3}\,x} = \frac{1}{\sqrt{3}} \qquad \text{so } \theta_y = \frac{\pi}{6}$$

$$\tan(-\theta_x) = \frac{y}{\sqrt{x^2 + z^2}} = \frac{2x}{\sqrt{x^2 + (\sqrt{3}\,x)^2}} = 1 \qquad \text{so } \theta_x = -\frac{\pi}{4}$$

Note that only the solutions in the first octant (in each case) were stated, simply because these are the angles relevant to the problem. Now, the rotation matrices $\mathbf{R}_y$ and $\mathbf{R}_x$ can be determined, and subsequently their transposes $\mathbf{R}_y^{\mathrm{T}}$ and $\mathbf{R}_x^{\mathrm{T}}$:

$$\mathbf{R}_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 & 0 \\ 0 & 1 & 0 & 0 \\ -1/2 & 0 & \sqrt{3}/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & -\sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now, suppose that an anti-clockwise rotation of $\theta = \pi/2$ about $q$ is sought. Then matrix $\mathbf{R}_z$ is written, using $\theta_z = \pi/2$:

$$\mathbf{R}_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hence the composite transformation **R**, that rotates coordinate space by $\theta = \pi/2$ about the axis $q$, on which the intersection of the planes $z = \sqrt{3}\,x$ and $y = 2\,x$ lies, is given:

$$\mathbf{R} = \mathbf{R}_y \mathbf{R}_x \mathbf{R}_z \mathbf{R}_x^{\mathrm{T}} \mathbf{R}_y^{\mathrm{T}}$$

$$\approx \begin{bmatrix} 0.1250 & -0.3624 & 0.9236 & 0 \\ 0.8624 & 0.5000 & 0.0795 & 0 \\ -0.4906 & 0.7866 & 0.3750 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

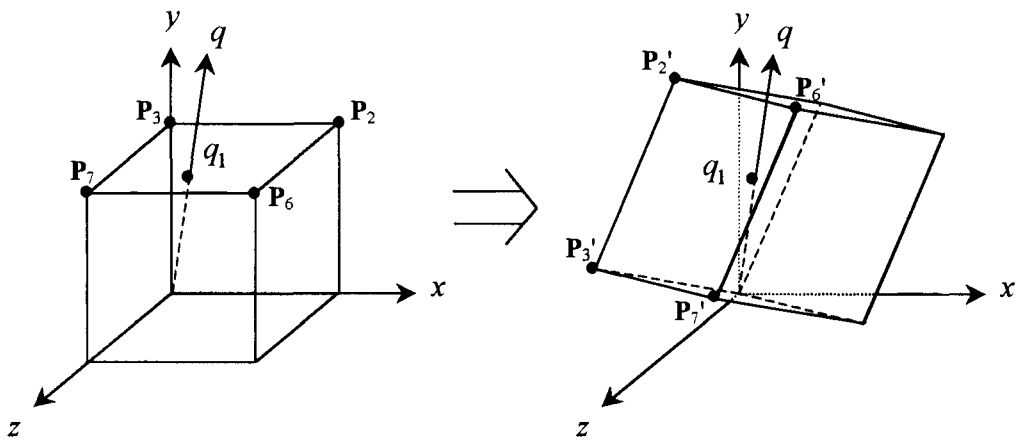
The cube with homogeneous vertices (0, 0, 0, 1), (1, 0, 0, 1), (1, 1, 0, 1), (0, 1, 0, 1), (0, 0, 1, 1), (1, 0, 1, 1), (1, 1, 1, 1) and (0, 1, 1, 1), represented by the vectors $\mathbf{P}_0$ to $\mathbf{P}_7$, is transformed by the above transformation (augmenting the vertex vectors):

$$\begin{bmatrix} \mathbf{P}_0' & \mathbf{P}_1' & \mathbf{P}_2' & \mathbf{P}_3' & \mathbf{P}_4' & \mathbf{P}_5' & \mathbf{P}_6' & \mathbf{P}_7' \end{bmatrix} = \mathbf{R} \begin{bmatrix} \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3 & \mathbf{P}_4 & \mathbf{P}_5 & \mathbf{P}_6 & \mathbf{P}_7 \end{bmatrix}$$

This yields the following set of transformed vertices, represented in $(x, y, z)$ coordinates:

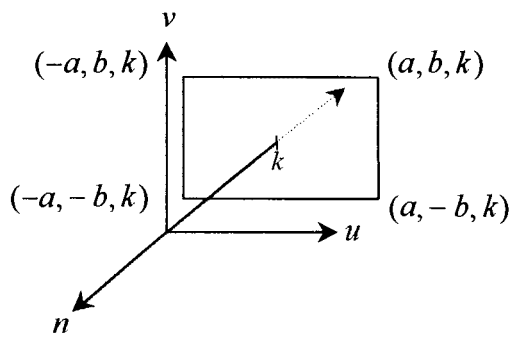
| Rotated vertices | P0' | P1' | P2' | P3' | P4' | P5' | P6' | P7' |
|---|---|---|---|---|---|---|---|---|
| $x$ | 0.0000 | 0.1250 | -0.2374 | -0.3624 | 0.9236 | 1.0486 | 0.6862 | 0.5612 |
| $y$ | 0.0000 | 0.8624 | 1.3624 | 0.5000 | 0.0795 | 0.9418 | 1.4418 | 0.5795 |
| $z$ | 0.0000 | -0.4906 | 0.2960 | 0.7866 | 0.3750 | -0.1156 | 0.6710 | 1.1616 |

Diagrammatically:



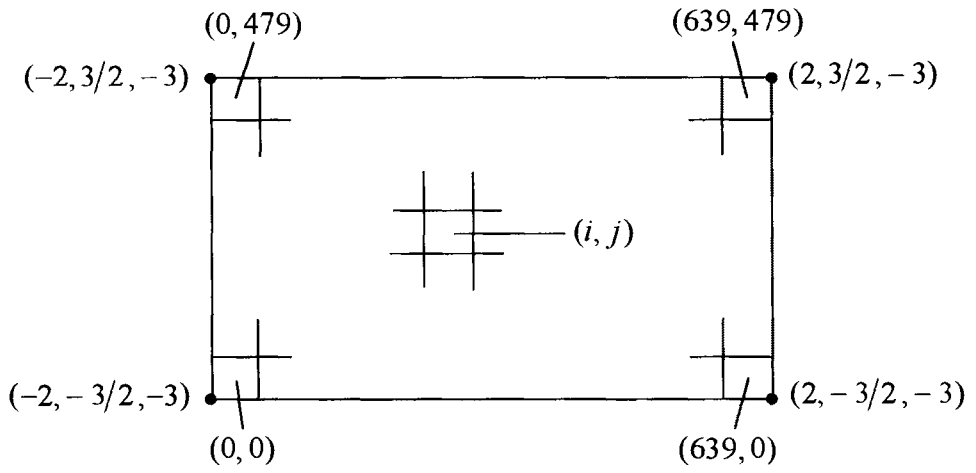

where $q_1 = [\frac{1}{2}, 1, \frac{\sqrt{3}}{2}]^T$ is a point on both the surface of the cube and the axis $q$ about which the cube is rotated.

**Example 3.2.1:** Specification of a View Plane and View Window, for use in further computations.




Working in view system space $(u, v, n)$, the view plane that will be featured in the following examples is $n = k = -3$. The view window will be described by $a = 2$ and $b = 3/2$, that is it has the vertices $(-2, 3/2, -3)$, $(2, 3/2, -3)$, $(2, -3/2, -3)$ and $(-2, -3/2, -3)$. The viewer is located at $(0, 0, 0)$, unless otherwise stated.

Resolution is defined horizontally and vertically by the constants $g = 640$ and $h = 480$. (i.e. an $640 \times 480$ array of pixels), with the central position of each pixel denoted $(i, j)$, where $0 \le i \le 639$, and $0 \le j \le 479$, for $i, j \in \mathrm{Z}$. Non-central positions on pixels such as pixel boundaries are referred to in terms of $(i, j)$ where $i, j \in \mathrm{R}$. Thus, the pixels on the view window can be shown:




**Example 3.2.2:** Pixel to Coordinate Mapping.

The relationship between the view system coordinates $(u, v, n)$ and the view window pixel references $(i, j)$ can be described by the one to one mapping $V : (i, j) \rightarrow (u, v, n)$, as follows.

For a view window with centre $(0, 0, k)$ and bounding vertices $(-a, b, k)$, $(a, b, k)$, $(a, -b, k)$, $(-a, -b, k)$, describing a resolution of $(g \times h)$:

$$V: \quad \begin{aligned} u &= a\left(\frac{2i+1}{g} - 1\right) \\ v &= b\left(\frac{2j+1}{h} - 1\right) \\ n &= k \end{aligned} \qquad \begin{aligned} -0.5 \le i \le (g - 0.5) \\ -0.5 \le j \le (h - 0.5) \end{aligned}$$

Thus, our mapping is stated:

$$V: \quad \begin{aligned} u &= 2\left(\frac{2i+1}{640} - 1\right) \\ v &= \frac{3}{2}\left(\frac{2j+1}{480} - 1\right) \\ n &= -3 \end{aligned} \qquad \begin{aligned} -0.5 \le i \le 639.5 \\ -0.5 \le j \le 479.5 \end{aligned}$$

For example, the view system coordinates of the centre of the corner pixel (639, 479) can be written:

$$u = 2\left(\frac{2(639)+1}{640} - 1\right) = \frac{639}{320}$$
$$v = \frac{3}{2}\left(\frac{2(479)+1}{480} - 1\right) = \frac{479}{320}$$
$$n = -3$$

so

$$V : (639, 479) \rightarrow (\frac{639}{320}, \frac{479}{320}, -3)$$

Similarly for the bottom left corner of the pixel $(0, 0)$, i.e. $i = -0.5$ and $j = -0.5$, the view system coordinates are:

$$u = 2\left(\frac{2(-0.5)+1}{640} - 1\right) = -2$$
$$v = \frac{3}{2}\left(\frac{2(-0.5)+1}{480} - 1\right) = -\frac{3}{2}$$
$$n = -3$$

so

$$V : (-0.5, -0.5) \rightarrow (-2, -\frac{3}{2}, -3)$$

**Example 3.3.1:** The implicitization of a parametric ray.

The trivial case of the implicitization of the parametric ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$, where:

$$\mathbf{r}_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \text{ and } \mathbf{r}_t = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

will be demonstrated in this example. For a non-trivial example, see Example 4.1.2. The set of points that describe the intersection of any two of the following three equations ((3.3.1.1), (3.3.1.2) or (3.3.1.3)) lie on a line describing the path of the ray $\mathbf{r} = \mathbf{r}_0 + t\mathbf{r}_t$:

$$(x_t z_t)x + (y_t z_t)y + (-x_t^2 - y_t^2)z + (-x_0 x_t z_t - y_0 y_t z_t + z_0 x_t^2 + z_0 y_t^2) = 0 \qquad (3.3.1.1)$$

$$(x_t y_t)x + (-x_t^2 - z_t^2)y + (y_t z_t)z + (-x_0 x_t y_t + y_0 x_t^2 + y_0 z_t^2 - z_0 y_t z_t) = 0 \qquad (3.3.1.2)$$

$$(-y_t^2 - z_t^2)x + (x_t y_t)y + (x_t z_t)z + (x_0 y_t^2 + x_0 z_t^2 - y_0 x_t y_t - z_0 x_t z_t) = 0 \qquad (3.3.1.3)$$

Substituting in the values of the variables as indicated by the ray **r**, into all three equations for the purpose of the example, yields:

$$(0)x + (0)y + (0)z + (0) = 0 \qquad (3.3.1.a)$$

$$(0)x + (-1)y + (0)z + (-0 + 0 + 1 - 0) = 0$$
$$y = 1 \qquad (3.3.1.b)$$

$$(-1)x + (0)y + (0)z + (0 + 1 - 0 - 0) = 0$$
$$x = 1 \qquad (3.3.1.c)$$

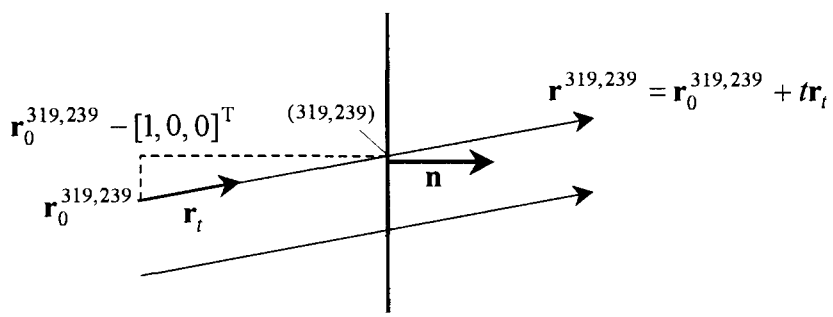
(3.3.1.a) provides no additional information, and so the implicit form of the ray **r** is the intersection of the planes $x=1$ and $y=1$, which is a result obtainable by simple inspection of **r**.

**Example 3.4.1:** General Parallel Projection.

The following illustrates an oblique projection, performed by displacing the points of origin of rays defined under an orthogonal projection. Denoting the displacement by $[c,d,e]^{\mathrm{T}}$, the equation for a ray $\mathbf{r}^{i,j}$, passing through pixel $(i,j)$ is written:

$$\mathbf{r}^{i,j} = [a\left(\frac{2i+1}{g}-1\right)+c, b\left(\frac{2j+1}{h}-1\right)+d, e]^{\mathrm{T}} + \frac{t}{\sqrt{c^2+d^2+(k-e)^2}}[-c,-d,k-e]^{\mathrm{T}}$$

Using the view system described in Example 3.2.1 ($a=2$, $b=3/2$, $k=-3$, $g=640$ and $h=480$) a parallel projection ray passing through the pixel (319, 239) will be defined for the displacement $[1, 0, 0]^{\mathrm{T}}$.




view window

$$\mathbf{r}^{319,239} = [2\left(\frac{2(319)+1}{640}-1\right)+1, \frac{3}{2}\left(\frac{2(239)+1}{480}-1\right), 0]^{\mathrm{T}} + \frac{t}{\sqrt{1+(-3)^2}}[-1, 0, -3]^{\mathrm{T}}$$

$$= \left[\frac{319}{320}, -\frac{1}{320}, 0\right]^{\mathrm{T}} + \frac{t}{\sqrt{10}}[-1, 0, -3]^{\mathrm{T}}$$

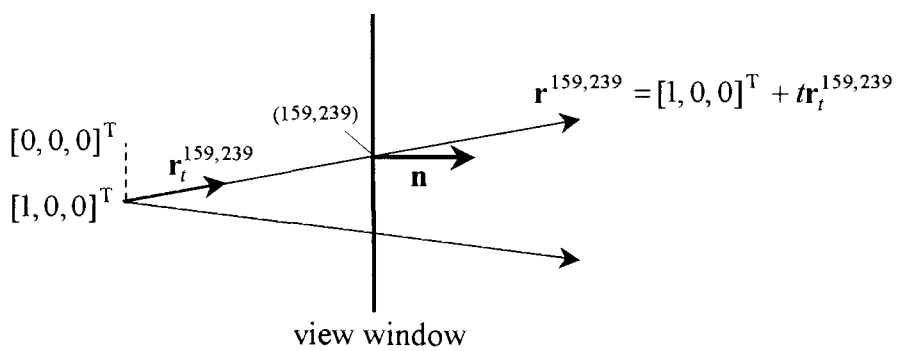
Note that the direction of this ray is simply a function of the displacement of its origin, and not its pixel position.

**Example 3.4.2:** General Perspective Projection

A perspective projection with centre of perspectivity or eye displaced by $[c, d, e]^{\mathrm{T}}$ has the equation for a ray $\mathbf{r}^{i,j}$, passing through pixel $(i, j)$ as follows:

$$\mathbf{r}^{i,j} = [c, d, e]^{\mathrm{T}} + \frac{t}{\sqrt{\left[a\left(\frac{2i+1}{g}-1\right)-c\right]^2 + \left[b\left(\frac{2j+1}{h}-1\right)-d\right]^2 + (k-e)^2}}[a\left(\frac{2i+1}{g}-1\right)-c, b\left(\frac{2j+1}{h}-1\right)-d, k-e]^{\mathrm{T}}$$

Using the view system described in Example 3.2.1 ($a = 2$, $b = 3/2$, $k = -3$, $g = 640$ and $h = 480$) a perspective projection ray that passes through the pixel (159, 239) will be defined for the displacement $[1, 0, 0]^{\mathrm{T}}$.

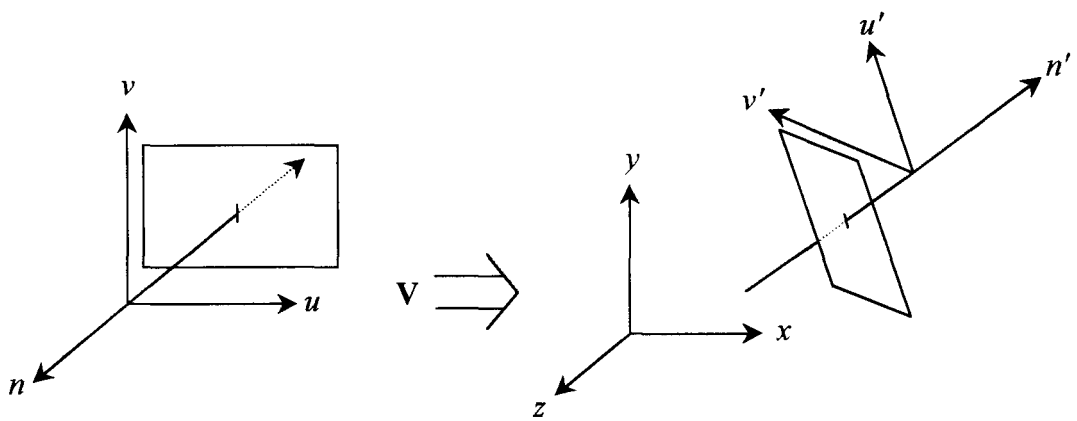


$$\mathbf{r}^{159,239} = [1,0,0]^{\mathrm{T}} + \frac{t}{\sqrt{\frac{641^2+1}{320}+9}}[-\frac{641}{320},-\frac{1}{320},-3]^{\mathrm{T}}$$
$$\approx [1.0\times10^{0},0,0]^{\mathrm{T}} + t[-5.553\times10^{-1},-8.663\times10^{-4},-8.316\times10^{-1}]^{\mathrm{T}}$$

Note that scientific notation was used because of the small $v$ component of the ray.

**Example 3.5.1:** Position of the View System in the Problem World.

To use a ray defined in view system space for the purpose of testing for intersections with the problem world and determining illuminations, it needs to be expressed in the problem world coordinate system $(x, y, z)$ by using the transformation $\mathbf{V} = \mathbf{T}_{\mathbf{V}}\mathbf{S}_{\mathbf{V}}\mathbf{R}_{\mathbf{V}}$.




Of course, this transformation determines what part of the problem world is to be viewed. For this example, consider that the (unscaled) viewer or eye is to be located at the problem world coordinates $(10,10,10)$, and is 'looking' straight at the origin of the problem world space . The transformation **V** that orients the view system as described can be determined as follows:

First, a rotation of $\theta_v = \pi/4$ of view system space about the $v$ axis,

then a rotation of $\theta_u = -\pi/4$ of view system space about the $u$ axis,

and finally a translation of view system space by +10 along each axis.

The corresponding matrix **V** is found by matrix multiplication, noting that scaling matrix $\mathbf{S_V}$ is an identity matrix as no scaling occurs:

$$\mathbf{V} = \mathbf{T_V}\mathbf{S_V}(\mathbf{R}_v\mathbf{R}_u)$$

$$= \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left( \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-\pi/4) & -\sin(-\pi/4) & 0 \\ 0 & \sin(-\pi/4) & \cos(-\pi/4) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\pi/4 & 0 & \sin\pi/4 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\pi/4 & 0 & \cos\pi/4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)$$

$$= \begin{bmatrix} \sqrt{2}/2 & 0 & \sqrt{2}/2 & 10 \\ -1/2 & \sqrt{2}/2 & 1/2 & 10 \\ -1/2 & \sqrt{2}/2 & 1/2 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{i.e.} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2}/2 & 0 & \sqrt{2}/2 & 10 \\ -1/2 & \sqrt{2}/2 & 1/2 & 10 \\ -1/2 & \sqrt{2}/2 & 1/2 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ n \\ 1 \end{bmatrix}$$

The view can be rotated about the chosen axis on which the viewplane normal lies by including a rotation about the $n$ axis in view system space before translating (and scaling).

**Example 4.1.2:** Triangular Facet.

Consider the triangular facet, a bounded plane cornered by vertices indicated by position vectors $\mathbf{P}_0 = [1,0,0]^T$, $\mathbf{P}_1 = [0,2,0]^T$ and $\mathbf{P}_2 = [0,0,1]^T$. The position vector $\mathbf{P}$ to any point on this facet can be defined parametrically as follows:

$$\mathbf{P} = \mathbf{P}_0 + \mu\mathbf{P}_\mu + \nu\mathbf{P}_\nu$$

where

$$\mathbf{P}_\mu = \mathbf{P}_1 - \mathbf{P}_0 = [-1,2,0]^T \text{ and}$$
$$\mathbf{P}_\nu = \mathbf{P}_2 - \mathbf{P}_0 = [-1,0,1]^T$$

for $\mu, \nu \geq 0$ and $\mu + \nu \leq 1$.

The normal to this parametrically defined surface is obtained using the cross product of any two linearly independent vectors running parallel to its plane. Two such vectors are $\mathbf{P}_\mu$ and $\mathbf{P}_\nu$:

$$\begin{aligned}\mathbf{n} &= \mathbf{P}_\mu \times \mathbf{P}_\nu \\ &= [-1,2,0]^T \times [-1,0,1]^T \\ &= [2,1,2]^T\end{aligned}$$

Determining the intersection of a planar facet with a ray can be approached in two ways, either by treating the facet parametrically and the ray implicitly, or the facet implicitly and the ray parametrically. The following will demonstrate both methods.

**Example 4.1.2.1:** Intersection of a parametrically defined facet with a ray.

Referring to the facet $\mathbf{P} = \mathbf{P}_0 + \mu[-1,2,0]^T + \nu[-1,01]^T$ where $\mu, \nu \geq 0$ and $\mu + \nu \leq 1$, we can state the individual coordinate equations in $(x, y, z)$ terms:

$$x = 1 - \mu - \nu \qquad (4.1.2.a)$$
$$y = 2\mu \qquad (4.1.2.b)$$
$$z = \nu \qquad (4.1.2.c)$$

Now consider the parametrically defined ray $\mathbf{r} = [5,5,3]^T + t[-2,-2,-1]^T$. In order to find the intersection with the parametrically defined facet, the ray needs to be implicitized. Using equations (3.3.1.1) and (3.3.1.2) from the body of the report, the implicitization follows:

$$(x_t z_t)x + (y_t z_t)y + (-x_t^2 - y_t^2)z + (-x_0 x_t z_t - y_0 y_t z_t + z_0 x_t^2 + z_0 y_t^2) = 0 \qquad (3.3.1.1)$$

$$2x + 2y - 8z + (-10 - 10 + 12 + 12) = 0$$
$$x + y - 4z + 2 = 0 \qquad (4.1.2.d)$$

$$(x_t y_t)x + (-x_t^2 - z_t^2)y + (y_t z_t)z + (-x_0 x_t y_t + y_0 x_t^2 + y_0 z_t^2 - z_0 y_t z_t) = 0 \qquad (3.3.1.2)$$

$$4x - 5y + 2z + (-20 + 20 + 5 + -6) = 0$$
$$4x - 5y + 2z - 1 = 0 \qquad (4.1.2.e)$$

The implicit form of the line along the ray, then, is represented implicitly by the intersection of the planes described by (4.1.2.d) and (4.1.2.e). Substituting (4.1.2.a)-(4.1.2.c) into these plane equations, the problem is put into the terms of the two parameters $\mu$ and $\nu$:

$$(1 - \mu - \nu) + (2\mu) - 4(\nu) + 2 = 0$$
$$\mu - 5\nu + 3 = 0 \qquad (4.1.2.f)$$

and

$$\begin{aligned} 4(1-\mu-\nu)-5(2\mu)+2(\nu)-1&=0 \\ -14\mu-2\nu+3&=0 \end{aligned} \qquad (4.1.2.g)$$

Rearranging (4.1.2.f) to get $\mu = 5\nu - 3$ and substituting this result into (4.1.2.g), we get $\nu = 5/8$ and subsequently $\mu = 1/8$. Thus the point of intersection can be found by evaluating $\mathbf{P} = \mathbf{P}_0 + \mu[-1,2,0]^{\mathrm{T}} + \nu[-1,0\,1]^{\mathrm{T}}$ for these parameter values, which yields:

$$\mathbf{P} = \begin{bmatrix} 1/4 \\ 1/4 \\ 5/8 \end{bmatrix}$$

To find the ray parameter value of this intersection, it is simply a matter of solving the given parametric ray equation for the above point of intersection:

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 5/8 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \\ 3 \end{bmatrix} + t \begin{bmatrix} -2 \\ -2 \\ -1 \end{bmatrix}$$

The solution is provided when $t = 19/8$.

**Example 4.1.2.2:** Intersection of a implicitly defined facet with a ray.

Referring again to the facet $\mathbf{P} = \mathbf{P}_0 + \mu[-1,2,0]^{\mathrm{T}} + \nu[-1,0,1]^{\mathrm{T}}$ where $\mu, \nu \geq 0$ and $\mu + \nu \leq 1$, and noting that the surface normal is $\mathbf{n} = [2,1,2]^{\mathrm{T}}$, the implicit form of the plane of the facet is readily found. In general terms, the plane represented implicitly by $Ax + By + Cz + D = 0$ has a surface normal $\mathbf{n} = [A,B,C]^{\mathrm{T}}$. Thus, we have in this particular instance:

$$2x + y + 2z + D = 0$$

The remaining term $D$ is found by substituting a known point from the facet or plane into equation (4.1.2.h). For example, the vertex $\mathbf{P}_1 = [0, 2, 0]^T$ provides us with $D = -2$. Thus the implicit representation of the plane on which the facet $\mathbf{P}_0\ \mathbf{P}_1\ \mathbf{P}_2$ lies is stated:

$$2x + y + 2z - 2 = 0 \qquad (4.2.1.h)$$

The intersection with the parametrically defined ray $\mathbf{r} = [5, 5, 3]^T + t[-2, -2, -1]^T$ is provided when the ray coordinates are substituted into (4.2.1.h)

$$x = 5 - 2t$$
$$y = 5 - 2t$$
$$z = 3 - t$$

which gives:

$$2(5 - 2t) + (5 - 2t) + 2(3 - t) - 2 = 0$$
$$t = \frac{19}{8}$$

which is identical to that obtained for the same ray in Example 4.1.2.1.

**Example 4.1.3:** Intersection of a Parametric Ray with a Sphere.

The unit sphere is defined in $(x, y, z)$ space by the equation $x^2 + y^2 + z^2 - 1 = 0$. To test for any intersection of a parametrically defined ray with this sphere, one may substitute the ray coordinates into that equation to obtain an expression in terms of the ray parameter. The general case is stated:

$$F^{\mathbf{r}}(t) = (x_0 + tx_t)^2 + (y_0 + ty_t)^2 + (z_0 + tz_t)^2 - 1 = 0$$
$$t^2(x_t^2 + y_t^2 + z_t^2) + 2t(x_0x_t + y_0y_t + z_0z_t) + (x_0^2 + y_0^2 + z_0^2) - 1 = 0$$

For the ray $\mathbf{r} = [5,5,3]^{\mathrm{T}} + t[-2,-2,-1]^{\mathrm{T}}$ this becomes $9t^2 - 46t + 58 = 0$. There are two solutions:

$t = \dfrac{23 + \sqrt{7}}{9} \approx 2.8495$, providing the intersection at $[-0.6991, -0.6991, 0.1505]^{\mathrm{T}}$, and

$t = \dfrac{23 - \sqrt{7}}{9} \approx 2.2616$, providing the intersection at $[0.4768, 0.4768, 0.7384]^{\mathrm{T}}$.

The solution for which $t \approx 2.2616$ occurs first in terms of travel of the ray, and would be the intersection considered by a ray tracing algorithm. The unit normal at this point is simply $\hat{\mathbf{n}} = [x, y, z]^{\mathrm{T}} = [0.4768, 0.4768, 0.7384]^{\mathrm{T}}$.

**Example 4.1.4:** Intersection of a Parametric Ray with a Cylinder.

Consider the cylinder defined by the equation $x^2 + y^2 - 1 = 0$. To test for any intersection of a parametrically defined ray with this surface, one may substitute the ray coordinates into the equation to obtain an expression in terms of the ray parameter. The general case is stated:

$$F^{\mathbf{r}}(t) = (x_0 + tx_t)^2 + (y_0 + ty_t)^2 - 1 = 0$$
$$t^2(x_t^2 + y_t^2) + 2t(x_0x_t + y_0y_t) + (x_0^2 + y_0^2) - 1 = 0$$

For the ray $\mathbf{r} = [5,5,3]^T + t[-2,-2,-1]^T$ this becomes $9t^2 - 46t + 58 = 0$. There are two solutions for the infinite cylinder:

$t = \frac{10+\sqrt{2}}{4} \approx 2.8536$, providing the intersection at $[-0.7071, -0.7071, 0.1464]^T$, and

$t = \frac{10-\sqrt{2}}{4} \approx 2.1464$, providing the intersection at $[0.7071, 0.7071, 0.8536]^T$.

The solution for which $t \approx 2.1464$ is the solution occurring first in terms of travel of the ray, and would be the intersection considered by a ray tracing algorithm. However consider the finite cylinder, identical to this except that the variable $z$ is bounded within the range [0,0.5]. The ray no longer strikes the cylinder at this 'first point', rather it intersects the bounding plane $z = 0.5$ at $[0,0,0.5]^T$ when $t = 2.5$, and passes through the shape, exiting at the original point of intersection which has a $z$ value within the limits set ($z = 0.1464$).

**Example 4.1.5:** Intersection of a Parametric Ray with a Cone.

The equation in $t$:

$$F^{\mathbf{r}}(t) = t^2(x_t^2 + y_t^2 - z_t^2) + 2t(x_0x_t + y_0y_t - z_0z_t) + (x_0^2 + y_0^2 - z_0^2) - 1 = 0$$

denotes the parametric ray coordinate substitution into the equation of an infinite cone with semi-vertical angle of $\pi/4$ whose equation is $x^2 + y^2 - z^2 = 0$. For the ray $\mathbf{r} = [5,5,3]^T + t[-2,-2,-1]^T$ this becomes $7t^2 - 34t + 41 = 0$.

The solutions are simply obtained:

$t = \dfrac{17+\sqrt{2}}{7} \approx 2.6306$, providing the intersection at $[-0.2612,-0.2612,0.3694]^T$, and

$t = \dfrac{17-\sqrt{2}}{7} \approx 2.2265$, providing the intersection at $[0.5469,0.5469,0.7735]^T$.

The intersection at $[0.5469,0.5469,0.7735]^T$ is the solution occurring first in terms of travel of the ray. The surface normal at this point is given by $\mathbf{n} = [x,y,-z]^T = [0.5469,0.5469,-0.7735]^T$, which provides the unit normal $\hat{\mathbf{n}} = [0.4571,0.4571,-0.6465]^T$.

**Example 4.2:** Intersection of a Parametric Ray with a Torus.

The following equation is of a torus centred about the origin, with radius of revolution $s$ and cross-sectional radius $r$.

$$(x^2 + y^2 + z^2)^2 - 2(r^2 + s^2)(x^2 + y^2 + z^2) + 4s^2z^2 + (r^2 - s^2)^2 = 0 \qquad (4.2.a)$$

Again using the ray $\mathbf{r} = [5,5,3]^T + t[-2,-2,-1]^T$, the following illustrates the intersection of this ray with the implicitly defined surface. Consider then, the torus described when $r = 1$ and $s = 2$. (4.2.a) becomes:

$$(x^2 + y^2 + z^2)^2 - 10(x^2 + y^2 + z^2) + 16z^2 + 9 = 0 \qquad (4.2.b)$$

Now substituting in to (4.2.b) the following:

$$x = 5 - 2t$$
$$y = 5 - 2t$$
$$z = 3 - t$$

we obtain

$$F^r(t) = 81t^4 - 828t^3 + 3104t^2 - 5064t + 3044 = 0 \qquad (4.2.c)$$

To solve for the intersections, of which four are possible in the general case, one may use an iterative method such as Newton's Method, or attempt to use geometry to intersect the ray with the cross-section of the torus. Equation (4.2.c) has 2 solutions, $t = 2.8571$ and $t = 3.5111$, which return points of intersection at $[-0.7144,-0.7144,0.1428]^T$ and $[-2.0220,-2.0220,-0.5111]^T$ respectively.

The surface normal is given by the partial derivative, with respect to each coordinate, of (4.2.b):

$$\mathbf{n} = \begin{bmatrix} 4x(x^2 + y^2 + z^2 - 5) \\ 4y(x^2 + y^2 + z^2 - 5) \\ 4z(x^2 + y^2 + z^2 + 3) \end{bmatrix}$$

The surface unit normal at the 'first' point of intersection, $[-0.7144, -0.7144, 0.1428]^T$, is then $\mathbf{n} = [11.3125, 11.3125, 2.3086]^T$, which gives the unit normal at this point to be $\hat{\mathbf{n}} = [0.6999, 0.6999, 0.1428]^T$.

**Example 5:** Positioning of a point light source over a surface (for the purposes of following examples regarding illumination topics)

This series of examples illustrating some of the concepts discussed in the body of this project (in Chapter 5) is based on the following:

The surface to be illuminated is the plane $y = 1$ in problem world space (object space transformations will be disregarded in this example). The point at which the illumination will be examined is $\mathbf{P}_0 = [1,1,1]^T$, at which the unit normal is denoted as facing in the direction of the positive $y$ axis, that is $\hat{\mathbf{n}} = [0,1,0]^T$. Let the surface also be the boundary between two media with different refractive indices. Medium A, for which $y > 1$, has the refractive index $\eta_A = 1$. Medium B, for which $y < 1$, has the refractive index $\eta_B = 4/3$.

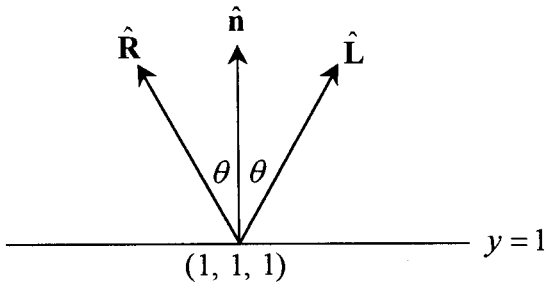
The point light source is located at $(x_L, y_L, z_L) = (1 + 2\sqrt{2}, 7, 3)$. Thus from $\mathbf{P}_0$, the unit vector indicating the direction in which this source lies is

$$\mathbf{L} = [\frac{\sqrt{2}}{2\sqrt{3}}, \frac{\sqrt{3}}{2}, \frac{1}{2\sqrt{3}}]^T$$

A final consideration is that the viewer of the illumination of the point $\mathbf{P}_0$ is located at $(-2, 4, -2)$. Thus, the unit vector representing the direction of the viewer from the point of interest is

$$\hat{\mathbf{k}} = [-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}]^T$$

**Example 5.2.1:** Determining the direction of perfectly specularly reflected light.



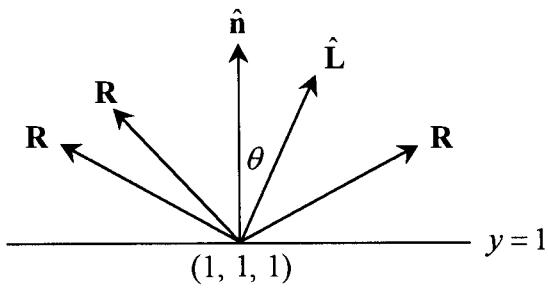

Using equation (5.2.1), the unit vector $\hat{\mathbf{R}}$ with direction of that of perfectly specularly reflected light can be computed:

$$\hat{\mathbf{R}} = 2\hat{\mathbf{n}}(\hat{\mathbf{n}} \bullet \hat{\mathbf{L}}) - \hat{\mathbf{L}} \tag{5.2.1}$$

$$= 2(\frac{\sqrt{3}}{2})[0,1,0]^{\mathrm{T}} - [\frac{\sqrt{2}}{2\sqrt{3}}, \frac{\sqrt{3}}{2}, \frac{1}{2\sqrt{3}}]^{\mathrm{T}}$$

$$= [-\frac{\sqrt{2}}{2\sqrt{3}}, \frac{\sqrt{3}}{2}, -\frac{1}{2\sqrt{3}}]^{\mathrm{T}}$$

**Example 5.2.2:** Diffuse reflection (and transmission)

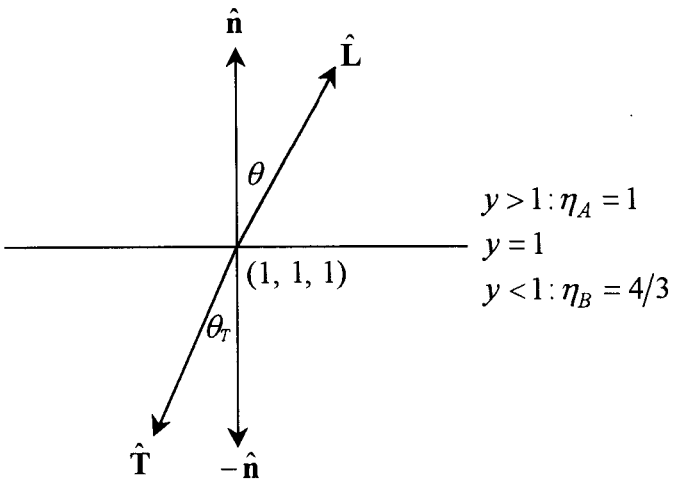



Diffuse interactions with a surface cause light to be scattered with uniform intensity in all directions (including those directions through the medium in which the particular light source is not located, should the surface facilitate transmission of light). The intensity at which the light is scattered is taken to be proportional to the cosine of the angle of incidence $\theta$. Thus, in this case

$$I_\lambda^D \propto I_{\lambda L}(\hat{\mathbf{n}} \bullet \hat{\mathbf{L}})$$
$$\propto \frac{\sqrt{3}}{2} I_{\lambda L}$$

Given the simplicity of the above statement one would not want to cast too much weight on any interpretation of it, but effectively the model suggests that the intensity of diffuse light that any viewer may see from this point due to this particular light source is about 86.6% of what the maximum intensity could be (as $\hat{\mathbf{n}} \bullet \hat{\mathbf{L}} \leq 1$).

**Example 5.2.3:** Determining the direction of perfectly specularly transmitted light.




Using equation (5.2.3), the unit vector $\hat{\mathbf{T}}$ with direction of that of perfect specularly reflected light can be computed:

$$\hat{\mathbf{T}} = \hat{\mathbf{n}}[\eta_{AB}(\hat{\mathbf{n}} \bullet \hat{\mathbf{L}}) - \sqrt{1 - \eta_{AB}^2(1 - (\hat{\mathbf{n}} \bullet \hat{\mathbf{L}})^2)}\ ] - \eta_{AB}\hat{\mathbf{L}} \qquad (5.2.3)$$

where $\eta_{AB} = \frac{\eta_A}{\eta_B} = \frac{3}{4}$.

$$\hat{\mathbf{T}} = [0,1,0]^{\mathrm{T}}\left(\frac{3}{4}\times\frac{\sqrt{3}}{2} - \sqrt{1-\frac{9}{16}\left(1-\left(\frac{\sqrt{3}}{2}\right)^2\right)}\right) - \frac{3}{4}[\frac{\sqrt{2}}{2\sqrt{3}}, \frac{\sqrt{3}}{2}, \frac{1}{2\sqrt{3}}]^{\mathrm{T}}$$
$$= [0, \frac{3\sqrt{3}-\sqrt{55}}{8}, 0]^{\mathrm{T}} - [\frac{\sqrt{2}\sqrt{3}}{8}, \frac{3\sqrt{3}}{8}, \frac{\sqrt{3}}{8}]^{\mathrm{T}}$$
$$= [-\frac{\sqrt{2}\sqrt{3}}{8}, -\frac{\sqrt{5}\sqrt{11}}{8}, -\frac{\sqrt{3}}{8}]^{\mathrm{T}}$$

The angle the vector $\hat{\mathbf{T}}$ makes with the negative surface normal is provided by the equation

$$\cos\theta_T = \hat{\mathbf{T}} \bullet (-\hat{\mathbf{n}}) = \hat{\mathbf{T}} \bullet \hat{\mathbf{n}}$$

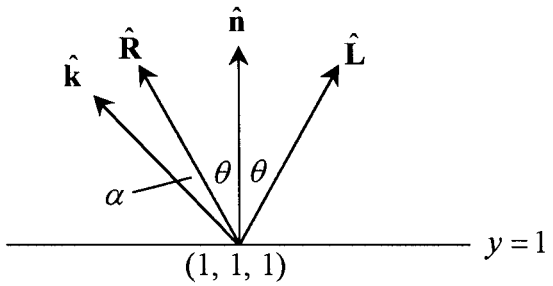
which provides us with $\theta_T = \cos^{-1}\left(\frac{\sqrt{5}\sqrt{11}}{8}\right) = \cos^{-1}\left(\sqrt{1-\frac{9}{64}}\right)$.

The sine of this angle is $\sqrt{9/64} = 3/8$. Note that this is in agreement with Snell's Law, that states:

$$\frac{\sin\theta_T}{\sin\theta} = \frac{\eta_A}{\eta_B}$$

Recall that the angle of incidence was $\theta = \pi/6$, as its cosine was $\sqrt{3}/2$. Thus the sine of this angle is $\sin\theta = 1/2$. From this and the fact that $\eta_A = 1$ and $\eta_B = 4/3$, Snell's Law confirms that the angle $\hat{\mathbf{T}}$ makes with the opposite to the surface normal has a sine of 3/8.

**Example 5.3.1.1:** Determining the angle between the direction of perfect specular reflection and the direction of the viewer.




One would expect that a perfect specularly reflected or transmitted light ray seldom travels in a direction straight towards the viewer. Rather than model the imperfect specular reflectance of the surface, the contribution of the light source to the illumination of a surface at a point is taken to be proportional to some function of the angle $\alpha$ between the 'perfect' ray and the vector pointing towards the viewer. For our example:

$$\begin{aligned}\cos\alpha &= \hat{\mathbf{R}} \bullet \hat{\mathbf{k}} \\ &= [-\frac{\sqrt{2}}{2\sqrt{3}}, \frac{\sqrt{3}}{2}, -\frac{1}{2\sqrt{3}}]^{\mathrm{T}} \bullet [-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}]^{\mathrm{T}} \\ &= \frac{4+\sqrt{2}}{6}\end{aligned}$$

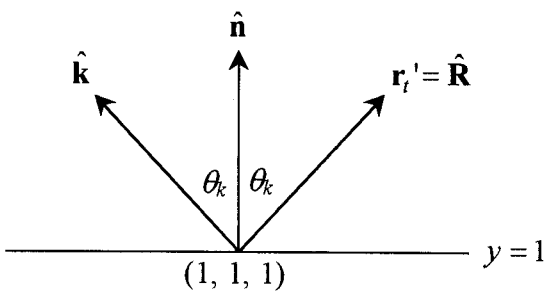
The function $f(\alpha, n) = \cos^n \alpha$ provides a suitable means of approximating the proportion of the original light source intensity reflected (or transmitted) towards the viewer (Phong, in Foley et al. 1994, p. 485). The variable $n$ distinguishes between different types of surfaces, with lower values denoting surfaces that scatter light to a greater extent than those with higher values of $n$. For our example then, consider the following proportions returned by $f(\alpha, n)$ for the angle

$$\alpha = \cos^{-1}\left(\frac{4+\sqrt{2}}{6}\right) \approx 0.4456^{R} \text{ or } 25.53^{\circ}.$$

| $n$ | $\cos^{n}(0.4456)$ |
|---|---|
| 1 | 0.9024 |
| 2 | 0.8143 |
| 8 | 0.4396 |
| 100 | 0.0003 |

For example when $n = 8$, the model suggests that 43.96% of the maximum intensity of light from the particular light source is specularly reflected off the surface in the direction towards the viewer.

**Example 5.3.2.1:** Determining the direction of a perfect specular reflection of a ray cast from the viewer.




Using equation (5.2.1), the unit vector $\hat{\mathbf{R}}$ may be found, the direction in which a secondary ray $\mathbf{r}'$ (representing a perfect specular reflection seen by the viewer) travels.

$$\hat{\mathbf{R}} = 2\hat{\mathbf{n}}(\hat{\mathbf{n}} \bullet \hat{\mathbf{k}}) - \hat{\mathbf{k}} \tag{5.2.1}$$

$$= 2(\frac{1}{\sqrt{3}})[0,1,0]^{\mathrm{T}} - [-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}]^{\mathrm{T}}$$

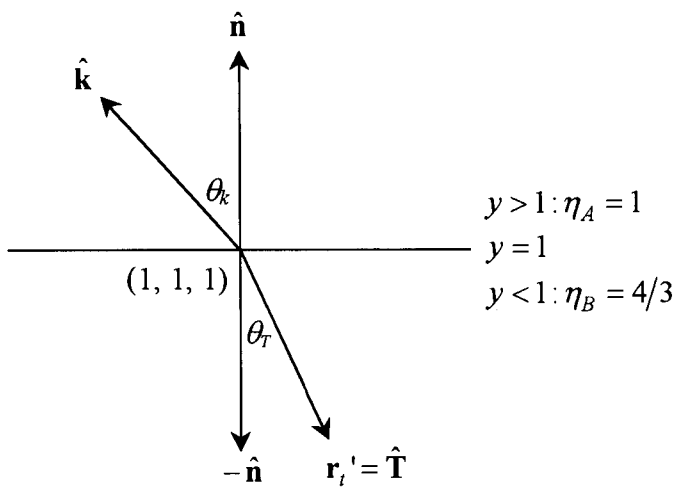
$$= [\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}]^{\mathrm{T}}$$

The secondary ray **r'** can be stated as

$$\mathbf{r}' = \mathbf{r}_0' + t\mathbf{r}_t'$$

$$\mathbf{r}' = [1,1,1]^{\mathrm{T}} + t[\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}]^{\mathrm{T}}$$

**Example 5.3.2.2:** Determining the direction of perfect specular transmission of a ray cast from the viewer.




Using equation (5.2.3), the unit vector $\hat{\mathbf{T}}$ may be found, the direction in which a secondary ray **r'** (representing a perfect specular transmission seen by the viewer) travels.

$$\hat{\mathbf{T}} = \hat{\mathbf{n}}[\eta_{AB}(\hat{\mathbf{n}} \bullet \hat{\mathbf{k}}) - \sqrt{1 - \eta_{AB}^2(1 - (\hat{\mathbf{n}} \bullet \hat{\mathbf{k}})^2)}\ ] - \eta_{AB}\hat{\mathbf{k}} \tag{5.2.3}$$

where $\eta_{AB} = \frac{\eta_A}{\eta_B} = \frac{3}{4}$.

$$\hat{\mathbf{T}} = [0,1,0]^{\mathrm{T}}\left(\frac{3}{4}\times\frac{1}{\sqrt{3}} - \sqrt{1-\frac{9}{16}\left(1-\left(\frac{1}{\sqrt{3}}\right)^2\right)}\right) - \frac{3}{4}[-\frac{1}{\sqrt{3}},\frac{1}{\sqrt{3}},-\frac{1}{\sqrt{3}}]^{\mathrm{T}}$$

$$= [0,\frac{\sqrt{3}-2\sqrt{5/2}}{4},0]^{\mathrm{T}} - [\frac{\sqrt{3}}{4},\frac{\sqrt{3}}{4},\frac{\sqrt{3}}{4}]^{\mathrm{T}}$$

$$= [\frac{\sqrt{3}}{4},-\frac{\sqrt{2}\sqrt{5}}{4},\frac{\sqrt{3}}{4}]^{\mathrm{T}}$$

So the secondary ray **r'** can be stated as

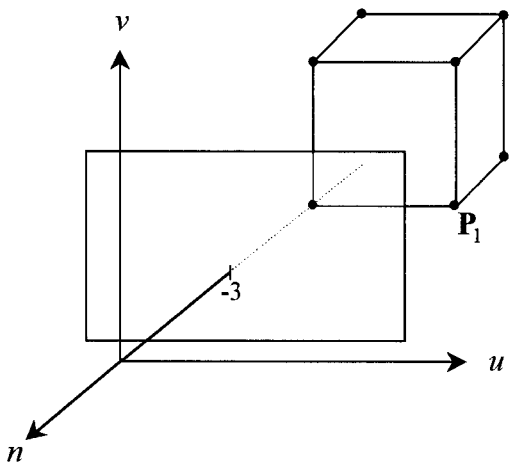
$$\mathbf{r}' = \mathbf{r}_0' + t\mathbf{r}_t'$$

$$\mathbf{r}' = [1,1,1]^{\mathrm{T}} + t[\frac{\sqrt{3}}{4},-\frac{\sqrt{2}\sqrt{5}}{4},\frac{\sqrt{3}}{4}]^{\mathrm{T}}$$

**Example 6:** Projections of a cube.

The following examples will project the vertices of a cube located in view system space $(u, v, n)$ on to the view plane. Theses vertices are:

| | P0 | P1 | P2 | P3 | P4 | P5 | P6 | P7 |
|---|---|---|---|---|---|---|---|---|
| $u$ | 0 | 3 | 3 | 0 | 0 | 3 | 3 | 0 |
| $v$ | 0 | 0 | 3 | 3 | 0 | 0 | 3 | 3 |
| $n$ | -6 | -6 | -6 | -6 | -9 | -9 | -9 | -9 |




**Example 6.1:** Parallel projection.

Using the following transformation, the set of vertices of the cube will be projected back on to the view plane, which is defined for $n = -3$. The projectors will not be orthogonal to the view plane, rather they will be defined by the displacement $[c, d, e]^{\mathrm{T}}$ of the origin of a vector passing through the centre of the view window $(0, 0, -1)$. In this instance, the displacement will be $[1, 1, 0]^{\mathrm{T}}$.

Using the matrix **L**

$$\mathbf{L}=\begin{bmatrix} 1 & 0 & -\dfrac{c}{e-k} & \dfrac{ck}{e-k} \\ 0 & 1 & -\dfrac{d}{e-k} & \dfrac{dk}{e-k} \\ 0 & 0 & 0 & k \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

for $c=1$, $d=1$, $e=0$ and $k=-3$, the transformation to view window coordinates $(A,B)$ is written

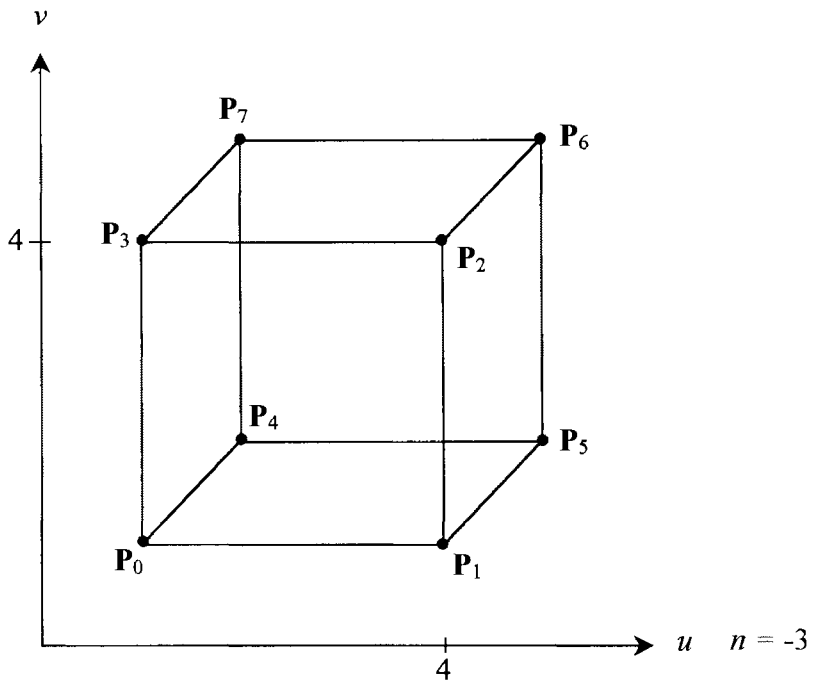
$$[A,B,k,1]^{\mathrm{T}}=\mathbf{L}[u,v,n,1]^{\mathrm{T}}$$

$$\begin{bmatrix} A \\ B \\ k \\ 1 \end{bmatrix}=\begin{bmatrix} 1 & 0 & -\dfrac{1}{3} & -1 \\ 0 & 1 & -\dfrac{1}{3} & -1 \\ 0 & 0 & 0 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} u \\ v \\ n \\ 1 \end{bmatrix}$$

which yields:

| | P0 | P1 | P2 | P3 | P4 | P5 | P6 | P7 |
|---|---|---|---|---|---|---|---|---|
| $A$ | 1 | 4 | 4 | 1 | 2 | 5 | 5 | 2 |
| $B$ | 1 | 1 | 4 | 4 | 2 | 2 | 5 | 5 |
| $k$ | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |

Plotting the above in $uv$ space, and joining the vertices on each edge by a line shows that the transformation provides an oblique image.




**Example 6.2:** Perspective Projection.

Using a perspective transformation, the set of vertices of the cube will be projected on to the view plane, which is defined for $n=-3$. The viewer has been displaced from the origin using $[c,d,e]^{\mathrm{T}}=[4,1,0]^{\mathrm{T}}$.

Using the matrix **R**

$$\mathbf{R}=\frac{1}{n-e}\begin{bmatrix} k-e & 0 & c & -ck \\ 0 & k-e & d & -dk \\ 0 & 0 & k & -ek \\ 0 & 0 & 1 & -e \end{bmatrix}$$

for $c=4$, $d=1$, $e=0$ and $k=-3$, the transformation to view window coordinates $(A,B)$ is written

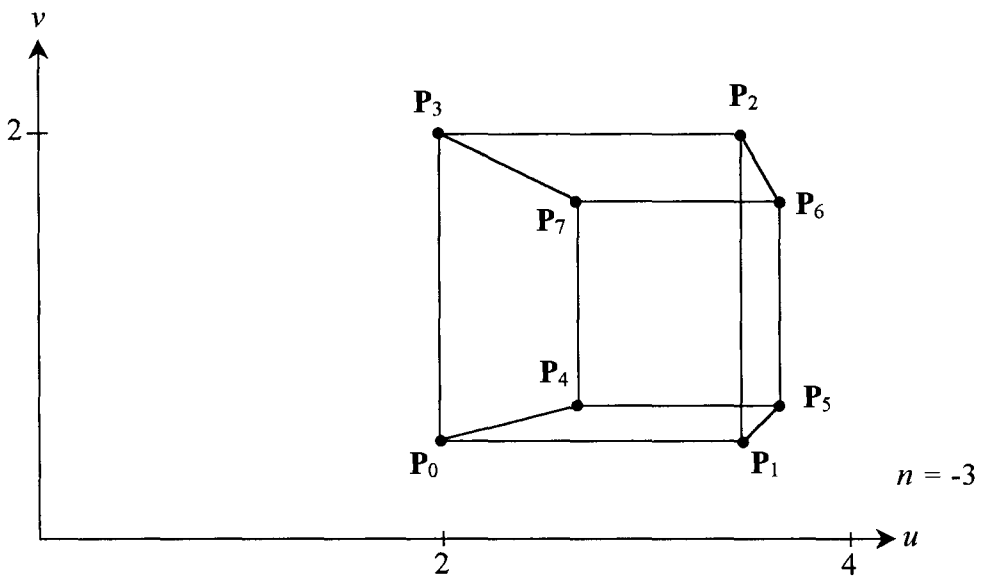
$$[A, B, k, 1]^{\mathrm{T}} = \mathbf{R}[u, v, n, 1]^{\mathrm{T}}$$

$$\begin{bmatrix} A \\ B \\ k \\ 1 \end{bmatrix} = \frac{1}{n} \begin{bmatrix} -3 & 0 & 4 & 12 \\ 0 & -3 & 1 & 3 \\ 0 & 0 & -3 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ n \\ 1 \end{bmatrix}$$

which yields:

| | P0 | P1 | P2 | P3 | P4 | P5 | P6 | P7 |
|---|---|---|---|---|---|---|---|---|
| *A* | 2 | 3.5 | 3.5 | 2 | 2.67 | 3.67 | 3.67 | 2.67 |
| *B* | 0.5 | 0.5 | 2 | 2 | 0.67 | 0.67 | 1.67 | 1.67 |
| *k* | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |

Plotting the above in *uv* space, and joining the vertices on each edge by a line shows that the transformation provides a perspective image in two dimensions.



# APPENDIX 2

Guide to Excel Routines for Parametric Curves and Surfaces.

## Guide to Excel Routines for Parametric Curves and Surfaces.

Included with this thesis are a number of Microsoft Excel Workbooks that illustrate the parametric curves and surfaces detailed in Sections 4.5 and 4.6. The included files are in Excel version 5.0 / Windows 95 format, and were created on a Pentium 100 with 32Mb RAM, using Excel 97. Macros are used to automate a number of features, included the selection of the type of curve and the entry of control points for pre-defined examples. Disabling the macros before opening a file does not render the workbook useless (curves will still be drawn for control points given manually) but may detract from the presentation of the concepts.

To use the files simply copy them from the floppy disks included with this thesis into the directory of choice on a suitable drive (e.g. C:\TEMP), and open them using a version of Excel no less than 5.0. Alternatively, from Excel open the files from the floppy disks and save them to an alternative drive.

Included Files:

**2d-Curves.xls**

**2d-Bezier.xls**

**2d-B-spline.xls**

**3d-Bezier.xls**

**3d-B-spline.xls**

## Note on Using Charts Interactively.

In many instances, it is preferable to manipulate the control points visually, rather than adjust an entry in the appropriate cell. On some of the charts featured in the included software, the control points have been plotted in addition to the curve which they define, in order to facilitate the direct manipulation of the shape of the curve. Charts in Excel are such that data plotted on them can be adjusted directly by using the mouse. However if the data plotted comes from cells containing functions, each adjustment will be queried with respect to what variable the user would like to adjust to achieve the desired change.

To use Excel charts interactively:

1. Select the chart on which you wish to alter data (control points) by performing a single left mouse button click when the mouse pointer is over some part of the chart.

2. Now carefully place the mouse pointer over some part of the series plot particular to the data point (control point). That is, if the control points are joined by lines, you can place the mouse pointer over a part of that line. Otherwise, just place the mouse pointer over a control point. Perform a SINGLE left click.

3. Pause. Clicking again too soon will bring up a dialogue box allowing you to adjust the specification of the chart in some way. If this occurs click on 'cancel' in the dialogue box and continue.

4. The data series should be selected (Excel highlights plotted points, but does not highlight any joining lines). Position the mouse pointer over the control point you wish to adjust and left click ONCE.

5. Pause. Should you left click again too soon a dialogue box may come up again. Should this happen, left click 'cancel' on the dialogue box and redo step 4.

6. After about a half a second after the single click, a number of things should happen. Most noticeably, if the mouse pointer is still over the control point selected, then the pointer changes from the default arrow to a four headed 'movement arrow'. If the mouse was moved off the control point after step 4, the default arrow should still be showing. Positioning the pointer over the selected point again causes the pointer to change to the four-headed arrow. Secondly, the only data that is highlighted is the selected point, by a large background square, and the 'previous' point in the series, by a smaller background square.

7. When the four headed arrow is showing (i.e., the pointer is over the selected control point) the user can *press and hold* the left mouse button to pick up the control point. By moving the mouse, the point can be adjusted either horizontally or vertically. Diagonal movement is permitted, but the only change registered is the maximum of the horizontal and vertical displacement. For example, moving the point to a location up and to the left of its original placement, but more up than left, will only register a change in terms of the vertical axis.

8. Letting go of the left button will drop the point at its new location, and Excel will adjust the corresponding cell accordingly. Should the cell contain a formula, then the spreadsheet will query the user as to which variable (cell) they wish to adjust to achieve their desired value.

9. Once dropped, the same point can be picked up again straight away by returning to step 7. If another point is to be adjusted, return to step 4, but note that one does not need to pause (in step 5) to get the movement cursor.

10. Once done moving the points, return to the normal worksheet function by left clicking on a location other than the chart on that sheet. This action will ensure that macro buttons (that become inactive when editing a chart on that sheet) are ready to use. A test for this is to see whether a small, pointing hand appears when the mouse pointer is passed over the macro button.

**Further Notes On Using Chart Interactively.**

It may occur that a number of control points that the user is wishing to adjust interactively are too close together or even on top of each other, preventing accurate manipulation using only the method described above. There are a number of options available.

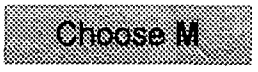
i. The user can use the 'zoom in' function to obtain a clearer view for those points close together.

ii. The user can use the left and right cursor keys to cycle through the particular data series. This is ideal when control points are stacked on each other. By selecting a point in the same series with the mouse in less congested area, the user can move along the series by pressing the left or right keys until they reach the point of congestion. Since with each press of a key they move either up or down one position in the series, any particular point is easily highlighted. Then all that needs to be done is to place the mouse pointer over the highlighted point, which should cause it to change to the 'movement arrow', and move the point as before.

**Notes on Using the file "2d-Curves.xls".**

This workbook generates a single, cubic parametric curve segment, based on four control points. The parametric curve types featured are Bézier, modified Bézier (Hermite), B-spline and β-spline. The workbook serves as a comparison between these types of curve segment, but does not illustrate the joining of such segments. The following is a brief description of the options available on important sheets of this workbook, and a reference to the appropriate sections of the body of this report..

Sheet: TYPE OF CURVE




Here the user can choose between the four types of curve:

Bézier (Section 4.5.4)

Modified Bézier or Hermite, for parameters $j$ and $k$ (Section 4.5.5)

B-spline (Section 4.5.6)

β-spline, for the tension parameter $t$ or $\tau$ (Section 4.5.6.1)

Note that when selecting a basis matrix **M** that uses a parameter value, adjusting the parameter value will alter the curve as soon as the new value is 'entered'. This involves actually pressing enter on the keyboard, as merely adjusting the number and leaving the cursor in that cell has no effect.

The blending functions for each type of curve (Section 4.5.4) can be viewed by clicking on the appropriate button on this sheet.

Sheet: CONTROL POINTS

Modify G

There are two example sets of control points on this sheet. The control points can be user defined by manual entry in to the appropriate cells, or by using the chart provided interactively.

Sheet: Q(t)

View Q(t)

This sheet displays the plot of the curve defined by $\mathbf{Q}(t) = \mathbf{GMT}$. It is computed for one hundred parameter values of $t$, all of which are regularly spaced. Note that the control points on the plot are interactive.

**Notes on Using the file "2d-Bezier.xls".**

This workbook generates a cubic parametric curve using three Bézier curve segments. The convex hull property is demonstrated for each segment, and through the use of control point manipulation, continuities at join points are illustrated. The following is a brief description of the options available on important sheets of this workbook, and a reference to the appropriate sections of the body of this report..

Sheet: TYPE OF CURVE


Here the user can choose between two types of curve:

Bézier (Section 4.5.4)

Modified Bézier or Hermite, for parameters $j$ and $k$ (Section 4.5.5)

Note that when selecting the basis matrix **M** for the modified Bézier curve, adjusting the parameter value will alter the curve as soon as the new value is 'entered'. This involves actually pressing enter on the keyboard, as merely adjusting the number and leaving the cursor in that cell has no effect. Also note that for $j = k = 3$ an unmodified Bézier curve is plotted.

The blending functions for each type of curve (Section 4.5.4) can be viewed by clicking on the appropriate button on this sheet.

Sheet: CONTROL POINTS

Modify G

This sheet allows the user to alter the nine control points for the three Bézier curve segments. Note that the first control points of the second and third segments are deliberately hidden, to ensure that the equality to the last control point of the preceding segment in each case is not removed. There are two example sets of control points on this sheet. Furthermore, the user can adjust the control points either by manual entry to the appropriate cells, or by using the chart provided.

Two further options are provided. The first, labelled "Linearise: Adjust Join Points" automatically recomputes control points $\mathbf{P}_3$ and $\mathbf{P}_6$ as the mid points of the line segments $\mathbf{P}_2\mathbf{P}_4$ and $\mathbf{P}_5\mathbf{P}_7$ respectively. Thus collinearity between is assured across these joins and the curve becomes smooth. The second option, "Make Loop", simply makes $\mathbf{P}_0$ and $\mathbf{P}_9$ equal, and makes $\mathbf{P}_1\mathbf{P}_8$ collinear.

Finally, the convex hulls of each segment can be viewed by clicking on the following box: View Convex Hull

Sheet: Q(t)

View Q(t)

This sheet displays the plot of the curve defined by $\mathbf{Q}(t) = \mathbf{G}\,\mathbf{M}\,\mathbf{T}$. It is computed for one hundred parameter values of $t$, all of which are regularly spaced. Note that the control points on the plot are interactive.

**Notes on Using the file "2d-B-spline.xls".**

This workbook generates a cubic parametric curve using seven uniform cubic B-spline curve segments. By far the most involved workbook, it covers examples of point replication to increase control at the cost of reduced continuity, the use of phantom end points to interpolate the first and last control points, the use of phantom control points to achieve interpolation of control points and the use of β-splines featuring a tension parameter.

Sheet: TYPE OF CURVE


Here the user can choose between two types of curve:

B-spline (Section 4.5.6)

β-spline, for the tension parameter $t$ or $\tau$ (Section 4.5.6.1)

Note that when selecting the basis matrix **M** for the β-spline, adjusting the parameter value will alter the curve as soon as the new value is entered, which involves actually pressing enter on the keyboard, as merely adjusting the number and leaving the cursor in that cell has no effect. Also note that for $\tau = t = 0$ there is no tension and a B-spline curve is plotted. For $t > 0$ tension is applied (the curve tends towards the control points). For $-12 < t < 0$, the curve moves away from the control points (more erratically as $t$ tends toward -12). It is recommended that the user does not enter values between $-11$ and $-13$, to avoid overflow errors (the plot becomes quite useless for these values).

Sheet: CONTROL POINTS

Modify G

This sheet allows the user to alter the ten control points for the seven spline curve segments. Three examples are provided to illustrate the flexibility of this particular type of curve. Furthermore, the user can adjust the control points either by manual entry in to the appropriate cells, or by using the chart provided. The following discusses the numerous features available on this sheet.

Controlling the End Points of a Spline (Section 4.5.6.2)

i. Duplication and Triplication of End Points.

Duplicate End Points

Triplicate End Points

By clicking on these buttons, a macro makes either the first and second, and ninth and tenth control points equal, or the first, second and third, and the eighth, ninth and tenth control points equal. Note that this method differs slightly from that described in the body of the report as we are not adding additional points, but rather redefining the end points and utilising the original end points for the purposes of either duplication or triplication. Still, the effect is the same – the curve is pulled towards the (new) end points when duplicated and it interpolates it when triplicated. Note that the user can manually triplicate any point using the table, to check that the curve becomes only $C^0$ continuous.

ii. The Use of Phantom End Points. Enable Disable

The theory behind the use of phantom end points is that by introducing an extra point at each end of the curve, the subsequent extra curve segments can be made to interpolate the original end points. The implementation of phantom end points provided by this workbook does not introduce any new points, instead it sacrifices the original end points and so provides interpolation for the second and ninth points, $\mathbf{P}_1$ and $\mathbf{P}_8$. Unlike the implementation of the replication measures identified previously, the phantom end points can be left 'on' (enabled) so to speak, so that the curve can be modified and still maintain the interpolation at the points $\mathbf{P}_1$ and $\mathbf{P}_8$, until the option is turned 'off' (disabled). Note that if the user attempts to interactively alter the phantom end points they will be queried as to what cell they want to adjust to do so, as the location of these phantom points is a function of the subsequent / previous two control points:

$$\mathbf{P}_0^* = 2\mathbf{P}_1 - \mathbf{P}_2 \text{ and } \mathbf{P}_9^* = 2\mathbf{P}_8 - \mathbf{P}_7$$

Interpolation Using a Cubic B-spline (Section 4.5.6.3) Interpolate Control Points

Using the following matrix, a set of phantom control points can be identified that, when used to plot a B-spline (not a β-spline) provide a curve that interpolates the intended control points.

$$\begin{bmatrix} -3 & 0 & 3 & 0 & \dots & 0 & 0 & 0 & 0 \\ 1/6 & 2/3 & 1/6 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 1/6 & 2/3 & 1/6 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1/6 & 2/3 & 1/6 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1/6 & 2/3 & 1/6 \\ 0 & 0 & 0 & 0 & \dots & 0 & -3 & 0 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{-1}^{*} \\ \mathbf{P}_{0}^{*} \\ \mathbf{P}_{1}^{*} \\ \vdots \\ \mathbf{P}_{s-1}^{*} \\ \mathbf{P}_{s}^{*} \\ \mathbf{P}_{s+1}^{*} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_0 \\ \mathbf{P}_0 \\ \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_{s-1} \\ \mathbf{P}_s \\ \mathbf{m}_s \end{bmatrix}$$

$$\mathbf{AG}^{*} = \mathbf{G}$$

$$\mathbf{G}^{*} = \mathbf{A}^{-1}\mathbf{G}$$

There are a number of things to note here. Primarily this is a ten by ten matrix requiring inversion (in this case the workbook plots for ten phantom control points, not ten control points). Excel can perform the computation for us, but a Gaussian elimination is provided on the sheet INTERP. GAUSSIAN ELIMINATION anyway. Secondly, there are two gradient requirements for the ends of the curve, $\mathbf{m}_1$ and $\mathbf{m}_8$. Rather than query the user further, these are automatically set to be the slopes of the lines between $\mathbf{P}_1$ and $\mathbf{P}_2$, and $\mathbf{P}_7$ and $\mathbf{P}_8$.

Upon pressing the button "Interpolate Control Points", the user is presented with the sheet INTERPOLATION CONTROL, which is similar to the previous sheet for control points except that there is a set of phantom points in addition to the given control points, and a panel of toggles. These toggles enable the user to indicate whether or not the control point listed above it should be interpolated. An option is also included to interpolate all or none (should the macro fail / be disabled, the user can go to the sheet INTERP. GAUSSIAN ELIMINATION and adjust the matrix **A** for each particular control point $\mathbf{P}_i$. Should $\mathbf{P}_i$ not need to be interpolated, row $i$ in **A** can be adjusted so that all entries are 0 other than a 1 on the main diagonal. For interpolation of $\mathbf{P}_i$, the

main diagonal element of row *i* should be 2/3, whilst the adjacent entries to the left and right of this should be 1/6).

Sheet: Q(t) and INTERP. Q(t) View Q(t) View Q(t)*

Again, the curve is plotted on these sheets. It should also be pointed out that the interpolation plot is interactive, which in effect enables the user to have very direct control over the shape of the curve.

**Notes on Using the files "3d-Bezier.xls" and "3d-B-spline".**

The transition from two-dimensional parametric cubic curves to three-dimensional parametric bi-cubic surfaces necessitates a simpler form of demonstration, partly due to the limitations of the plotting capabilities of Excel (we will only be able to plot surfaces where control just one coordinate, inspite of having the data available to control all three coordinates) and partly due to the fact that visually manipulating three dimensional control points is difficult with a two dimensional control device (the mouse).

The Bézier surface workbook, **3d-Bezier.xls**, features three Bézier surface patches joined in a row, each patch defined by 16 three-dimensional coordinates. The relevant section in the body of this project is Section 4.6.2. Control points can be manipulated across all three patches by looking at the same row number in each geometry matrix simultaneously.

The B-spline surface workbook, **3d-B-spline.xls**, features nine surface patches defined by a six by six coordinate array. Since continuity is automatic across the entire surface, the user can enter coordinates arbitrarily and produce a smooth surface (except when points are replicated). Refer to Section 4.6.3. of the project body for more information.

The graphic output provided is limited in that variations in only one coordinate at a time are shown. The effect is similar to defining each coordinate triple over a plane divided into a regular grid governing $x$ and $z$ coordinates say, while variable $y$ is freely defined.

# APPENDIX 2A

Data used as Examples in Excel Routines for Parametric Curves and Surfaces.

Below is the data stored as examples in each of the workbooks.

**2d-Bezier.xls**

Eg. 1

| | P0 | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 |
|---|---|---|---|---|---|---|---|---|---|---|
| x | 5.25 | 1.25 | 4 | 3 | 1.5 | 5.5 | 6.5 | 7.5 | 10.5 | 5.25 |
| y | 4 | 3 | 2 | 1.38 | 0.75 | 3 | 0.63 | 0 | 1.25 | 2.25 |

Eg. 2

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| x | 3 | 1.25 | 1.25 | 3 | 3 | 5 | 7.5 | 7.5 | 11 | 11 |
| y | 3 | 3 | 2 | 2 | 0.75 | 3 | 3 | 1.25 | 1.25 | 2 |

**2d-B-spline.xls**

Eg. 1

| | P0 | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 |
|---|---|---|---|---|---|---|---|---|---|---|
| x | 1 | 1 | 2 | 3 | 6 | 2 | 5 | 6 | 7 | 7 |
| y | 6 | 4 | 3 | 4 | 7 | 7 | 4 | 3 | 4 | 6 |

Eg. 2

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| x | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| y | 1 | 3 | 4 | 5 | 2 | 4 | 3 | 1 | 2 | 6 |

Eg. 3

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| x | 3 | 1 | 4 | 2 | 5 | 3 | 6 | 4 | 7 | 5 |
| y | 1 | 4 | 2 | 5 | 3 | 6 | 4 | 7 | 5 | 8 |

**3d-Bezier.xls**

*z* values

| | $y=0$ | $y=1$ | $y=2$ | $y=3$ | $y=4$ | $y=5$ | $y=6$ | $y=7$ | $y=8$ | $y=9$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $x=0$ | 0 | 3 | 3 | 2 | 0 | 3 | 4 | 5 | 5 | 3 |
| $x=1$ | 1 | 3 | 3 | 2 | 1 | 1 | 2 | 3 | 3 | 1 |
| $x=2$ | 1 | 3 | 3 | 2 | 1 | 1 | 2 | 3 | 3 | 1 |
| $x=3$ | 3 | 5 | 5 | 4 | 3 | 0 | 2 | 3 | 3 | 8 |

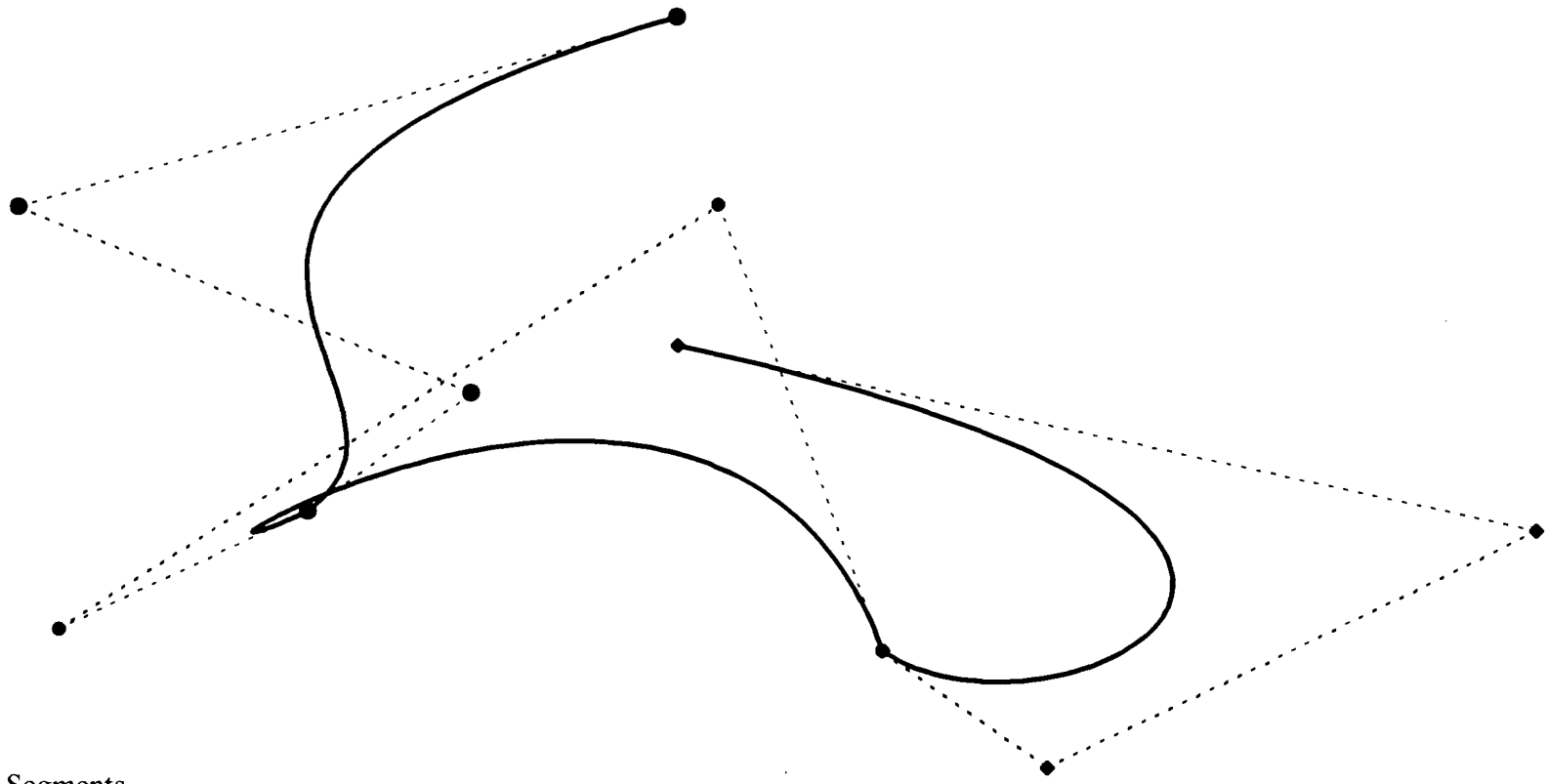
**3d-B-spline.xls**

$z$ values

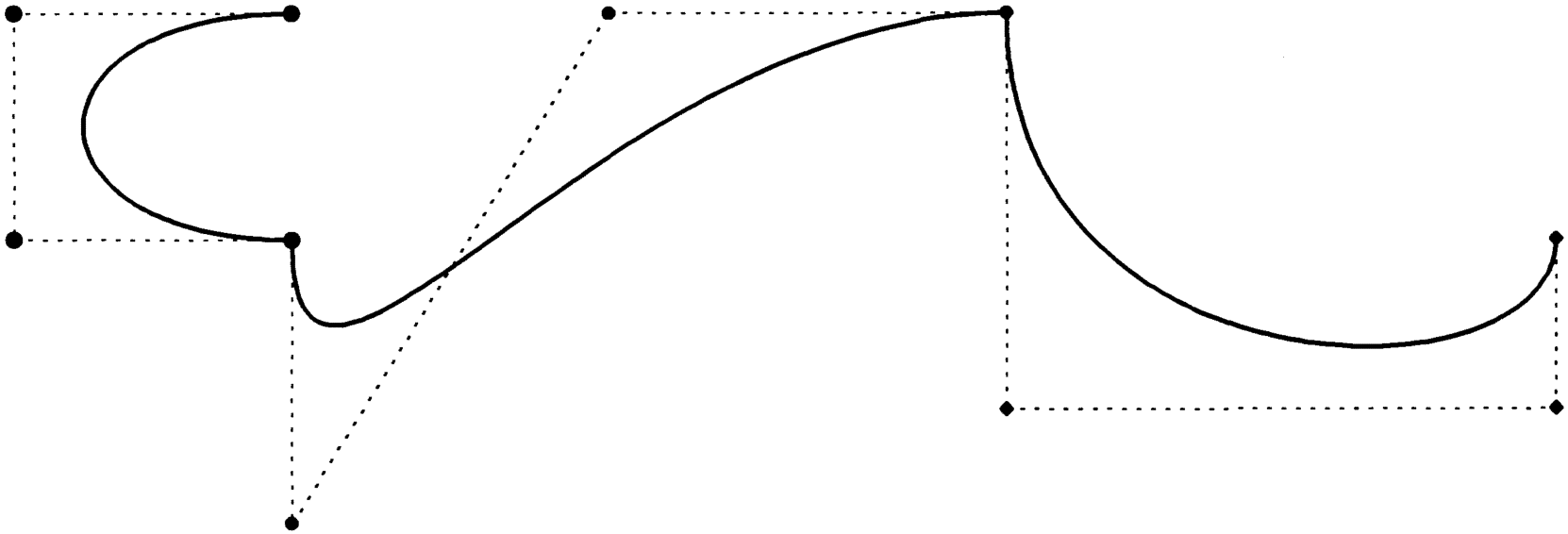
| | $y = 0$ | $y = 1$ | $y = 2$ | $y = 3$ | $y = 4$ | $y = 5$ |
|---|---|---|---|---|---|---|
| $x = 0$ | 2 | 2 | 4 | 4 | 4 | 0 |
| $x = 1$ | 2 | 4 | 2 | 2 | 4 | 4 |
| $x = 2$ | 4 | 2 | 6 | 2 | 2 | 4 |
| $x = 3$ | 4 | 2 | 2 | 2 | 2 | 4 |
| $x = 4$ | 4 | 2 | 2 | 2 | 3 | 2 |
| $x = 5$ | 0 | 4 | 4 | 2 | 2 | 3 |

# APPENDIX 2B

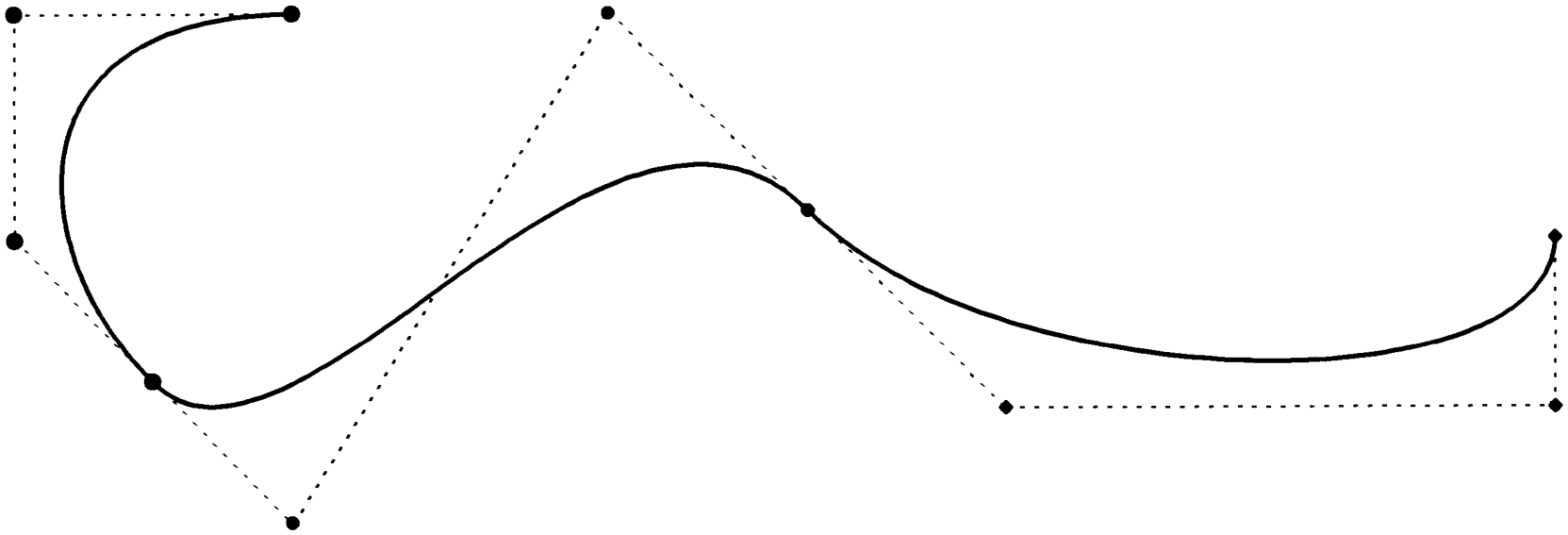
Plots of Examples in Excel Routines for Parametric Curves and Surfaces.



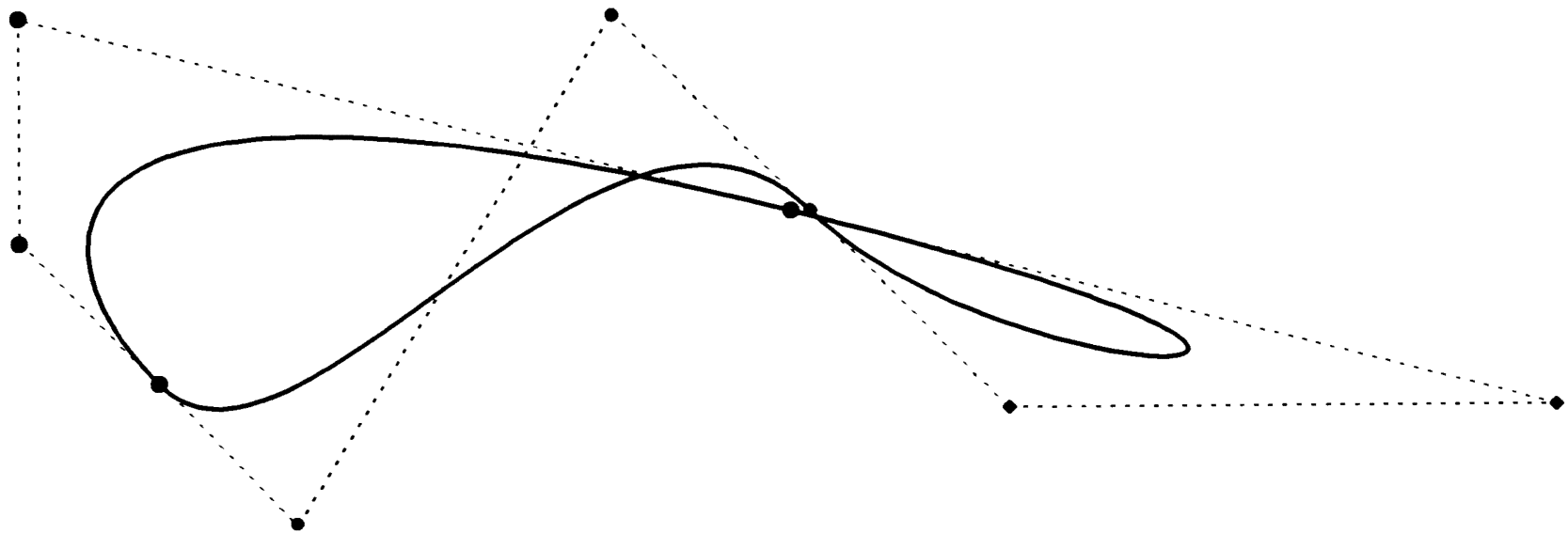
Bezier Curve Segments.
Example 1.



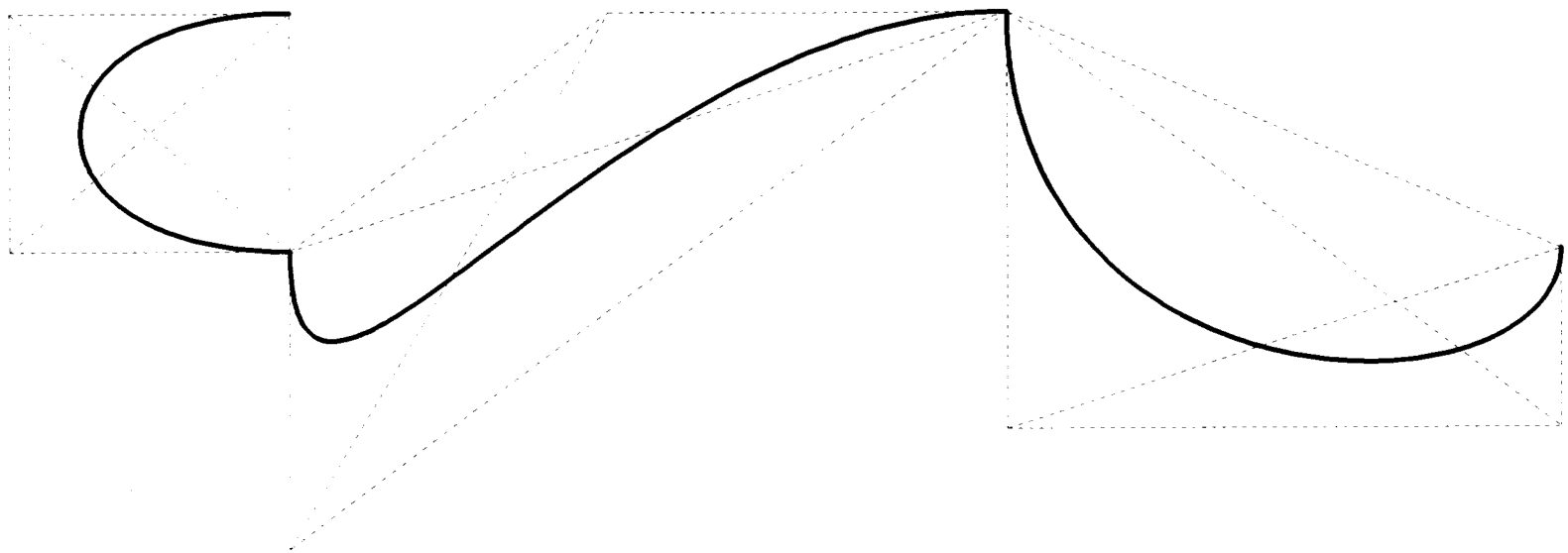
Bezier Curve Segments.
Example 2.



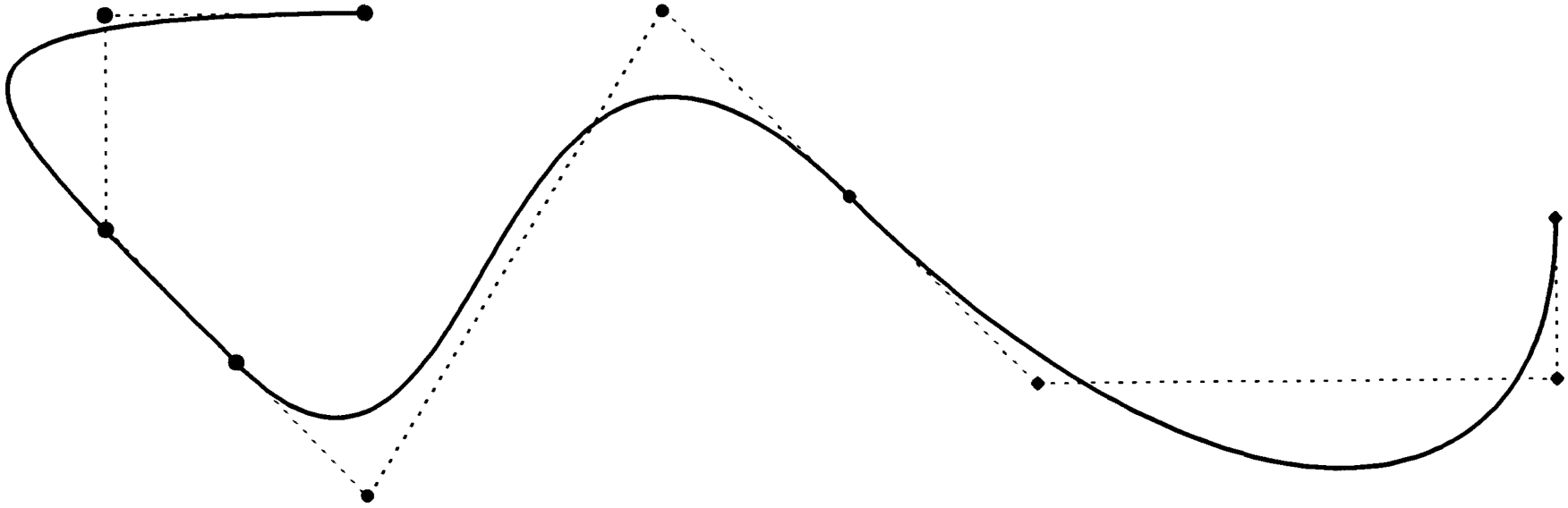
Bezier Curve Segments.
Example 2 data:
'Linearised' Control Points.



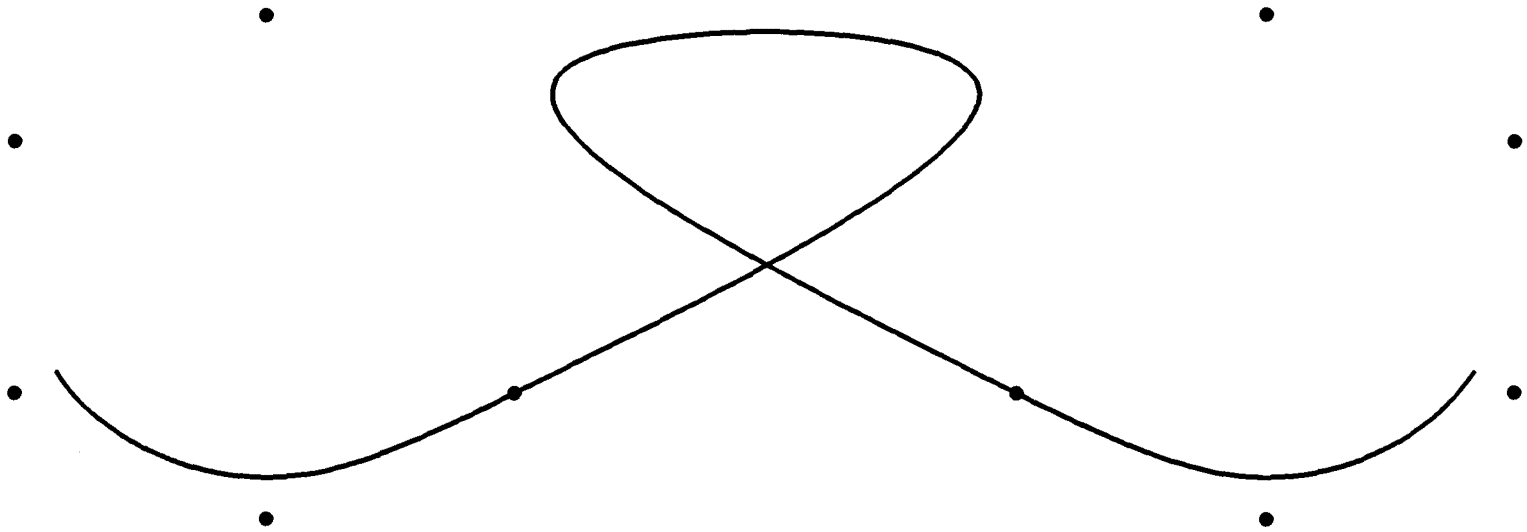
Bezier Curve Segments.
Example 2 data:
'Linearised' and Looped Control Points.



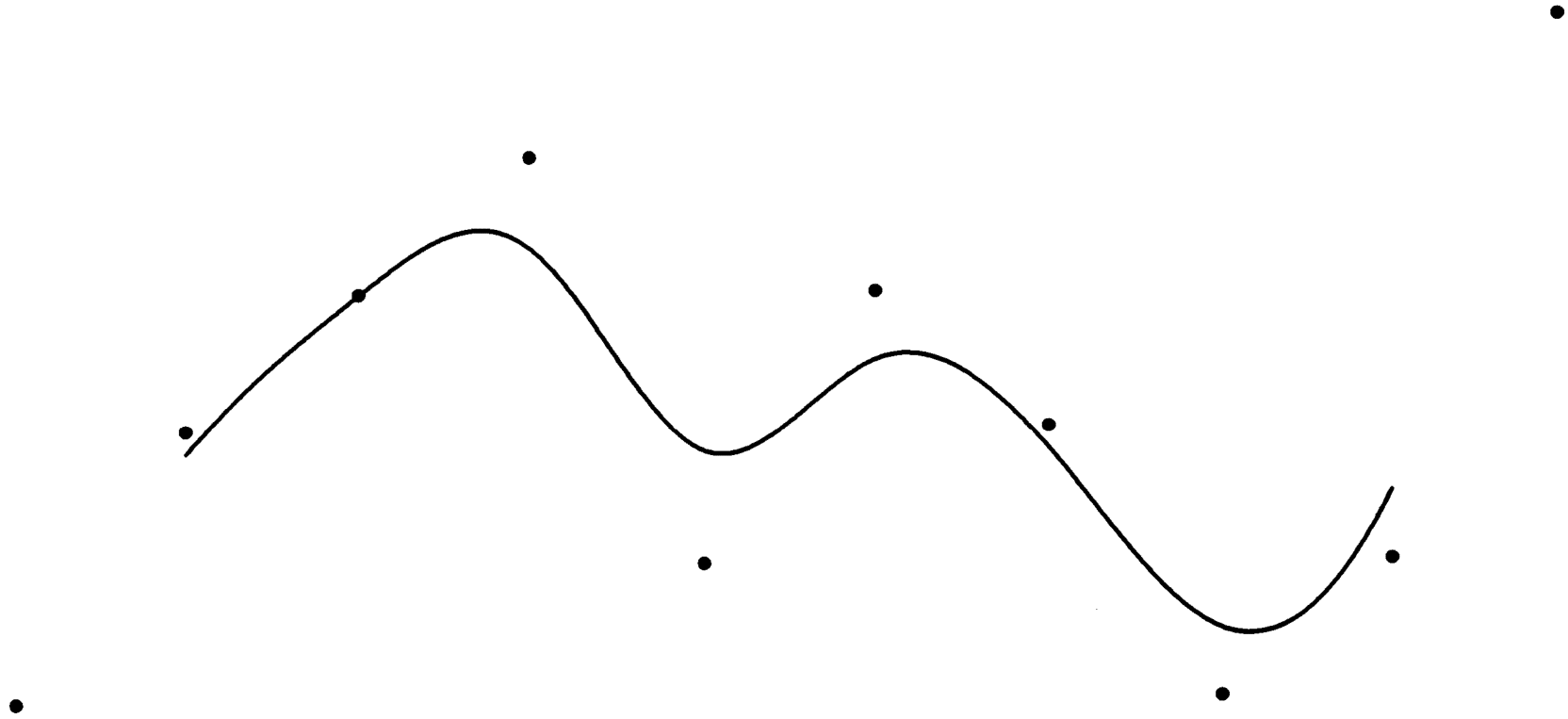
Bezier Curve Segments.
Example 2 data:
Convex Hulls of Segments (union of triangles)



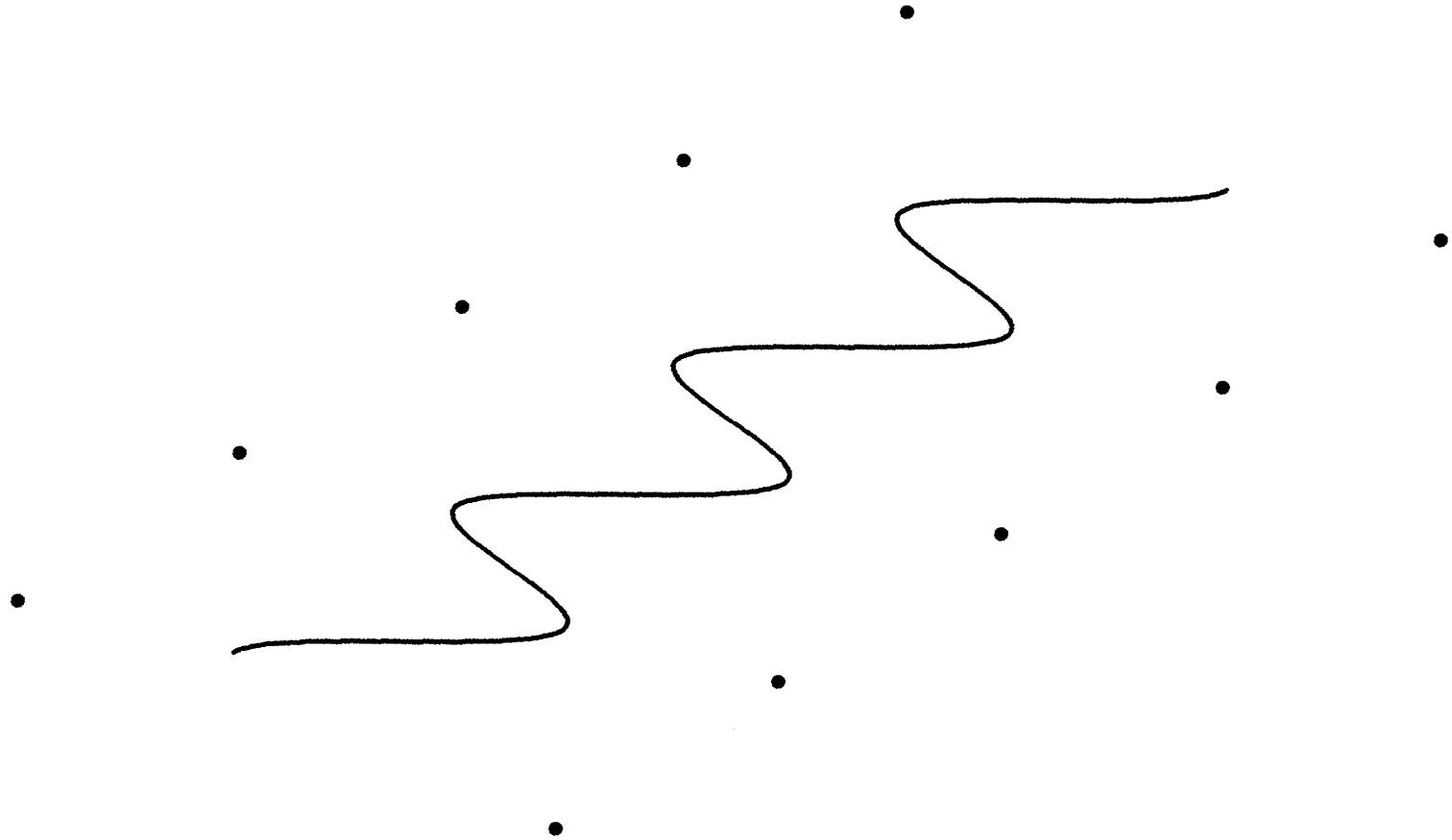
Modified Bezier Curve Segments.
$j = 6$ and $k = 6$
Example 2 data:
'Linearised'



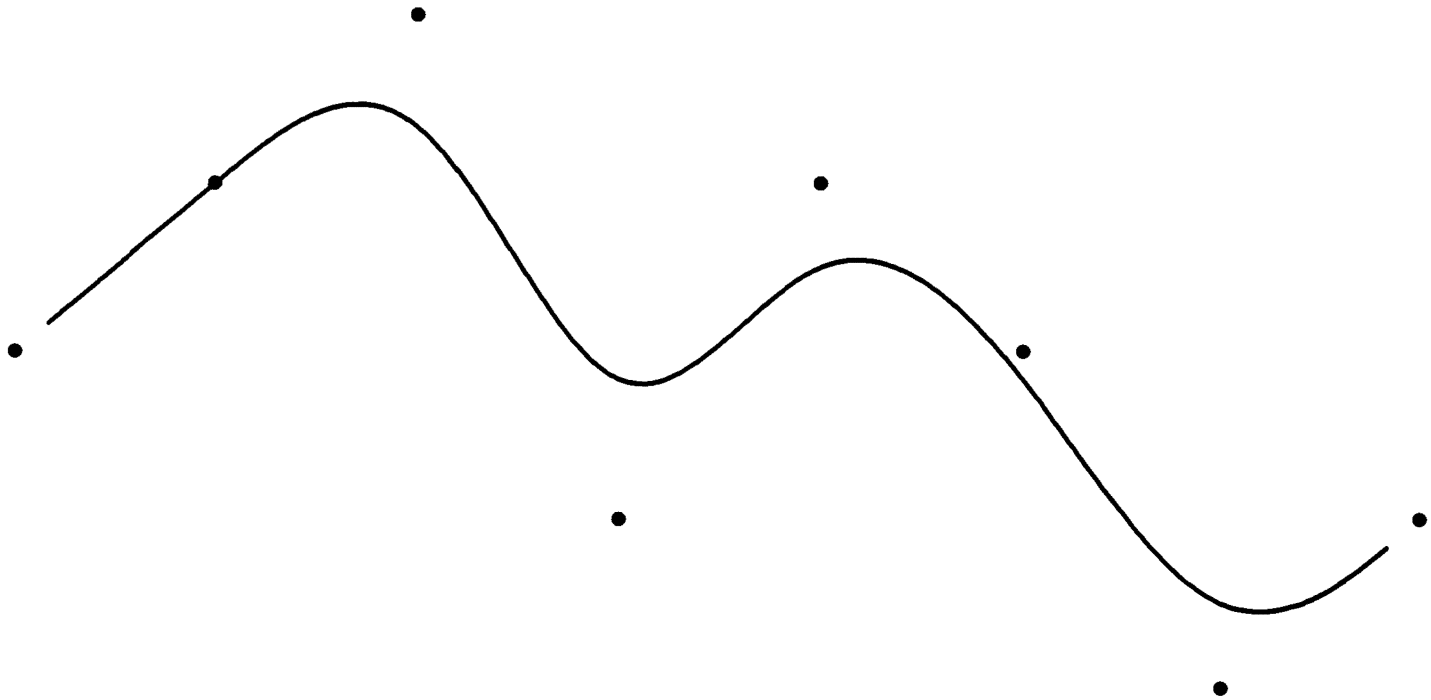
B-spline Curve Segments.
Example 1.



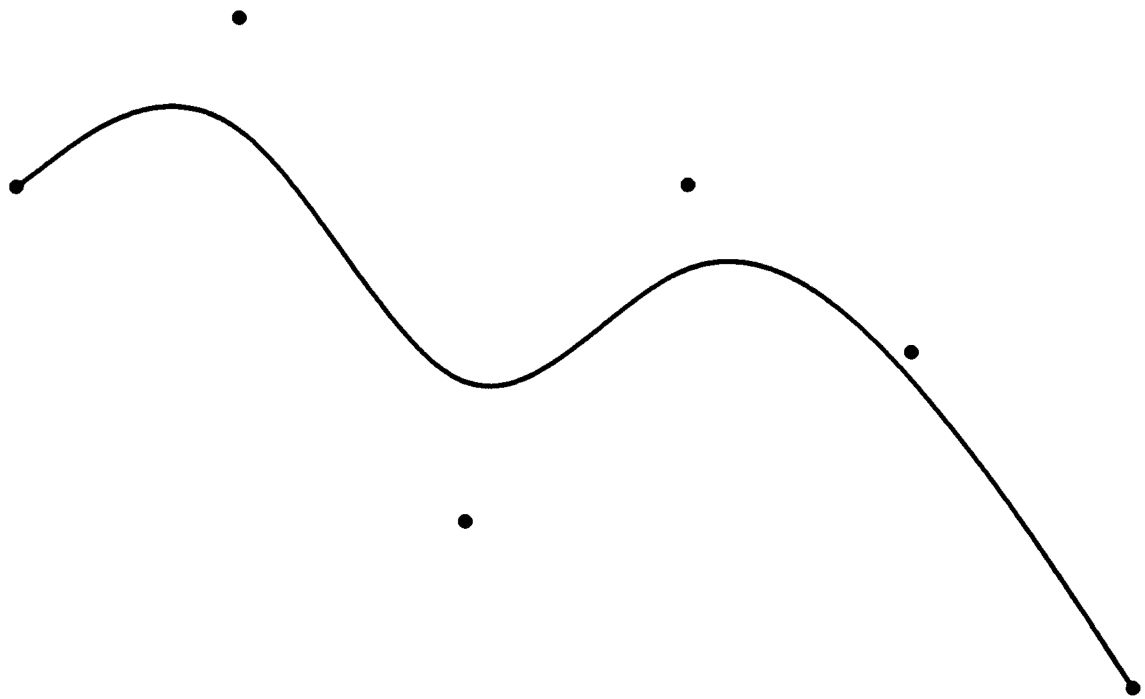
B-spline Curve Segments.
Example 2.



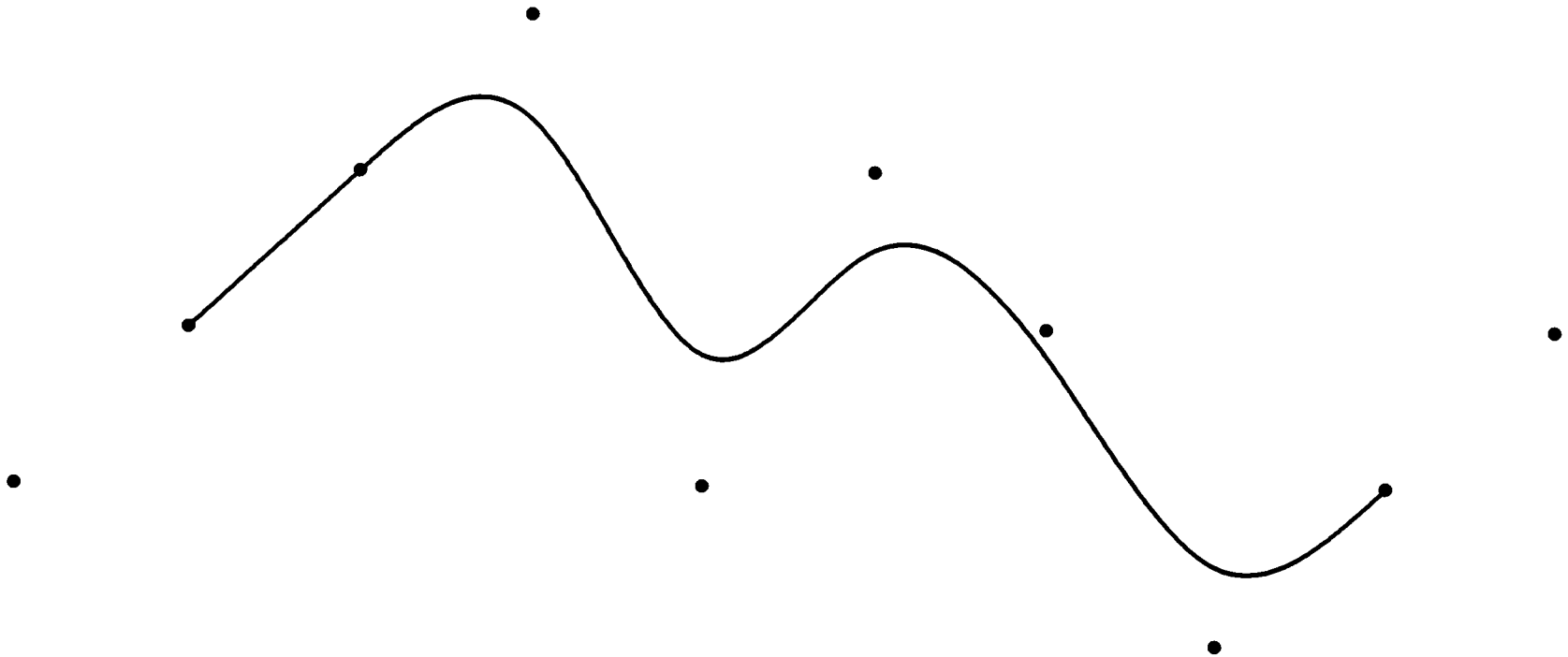
B-spline Curve Segments.
Example 3.



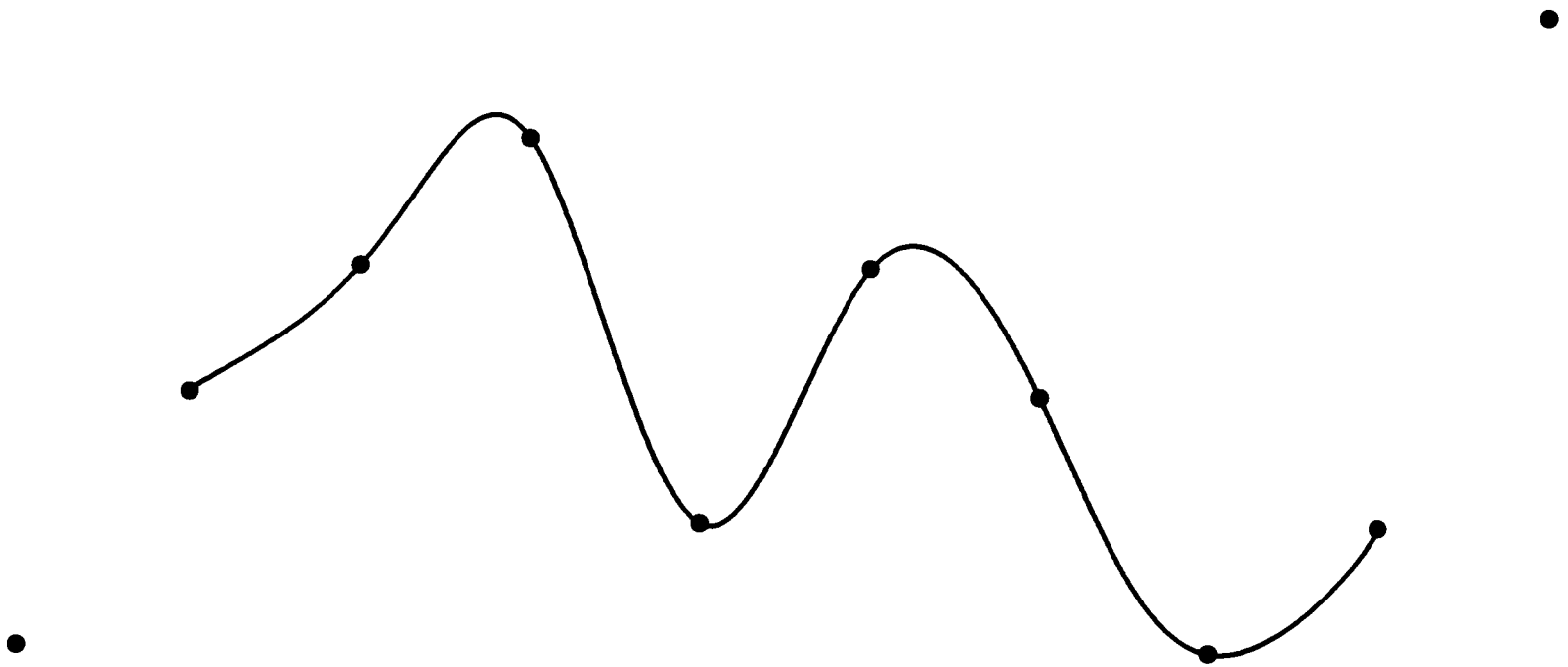
B-spline Curve Segments.
Example 2 data:
Duplicate End Points.



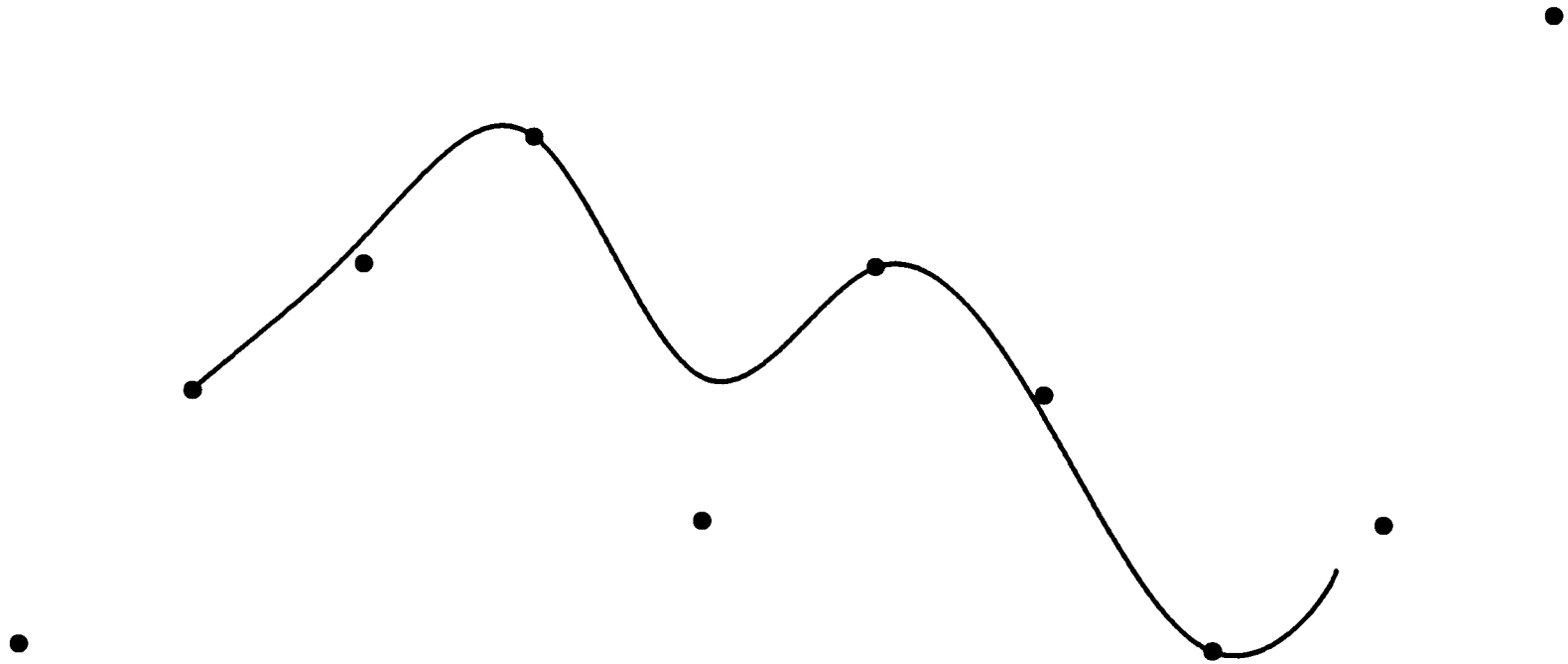
B-spline Curve Segments.
Example 2 data:
Triplicate End Points.



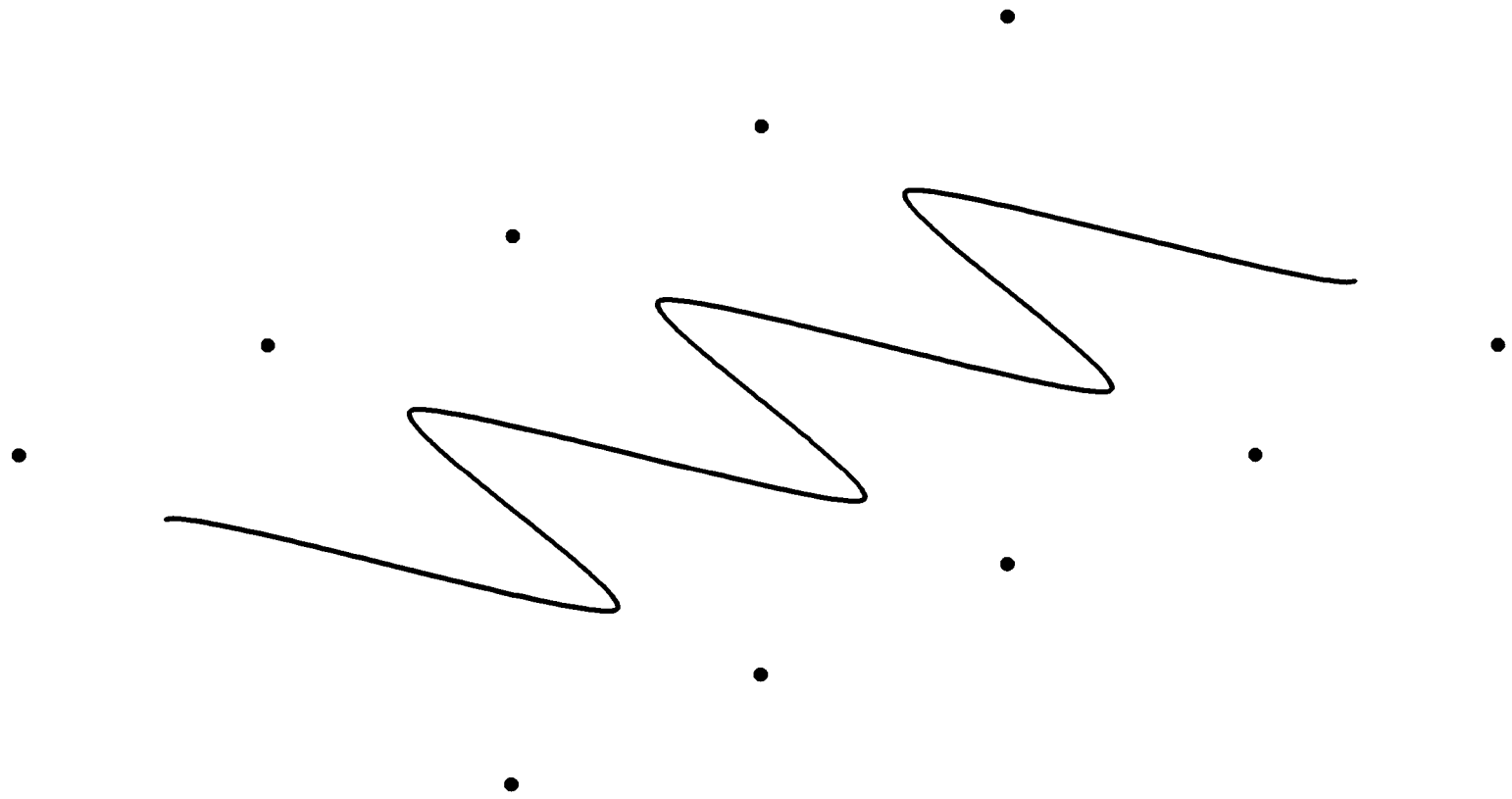
B-spline Curve Segments.
Example 2 data:
Phantom End Points.



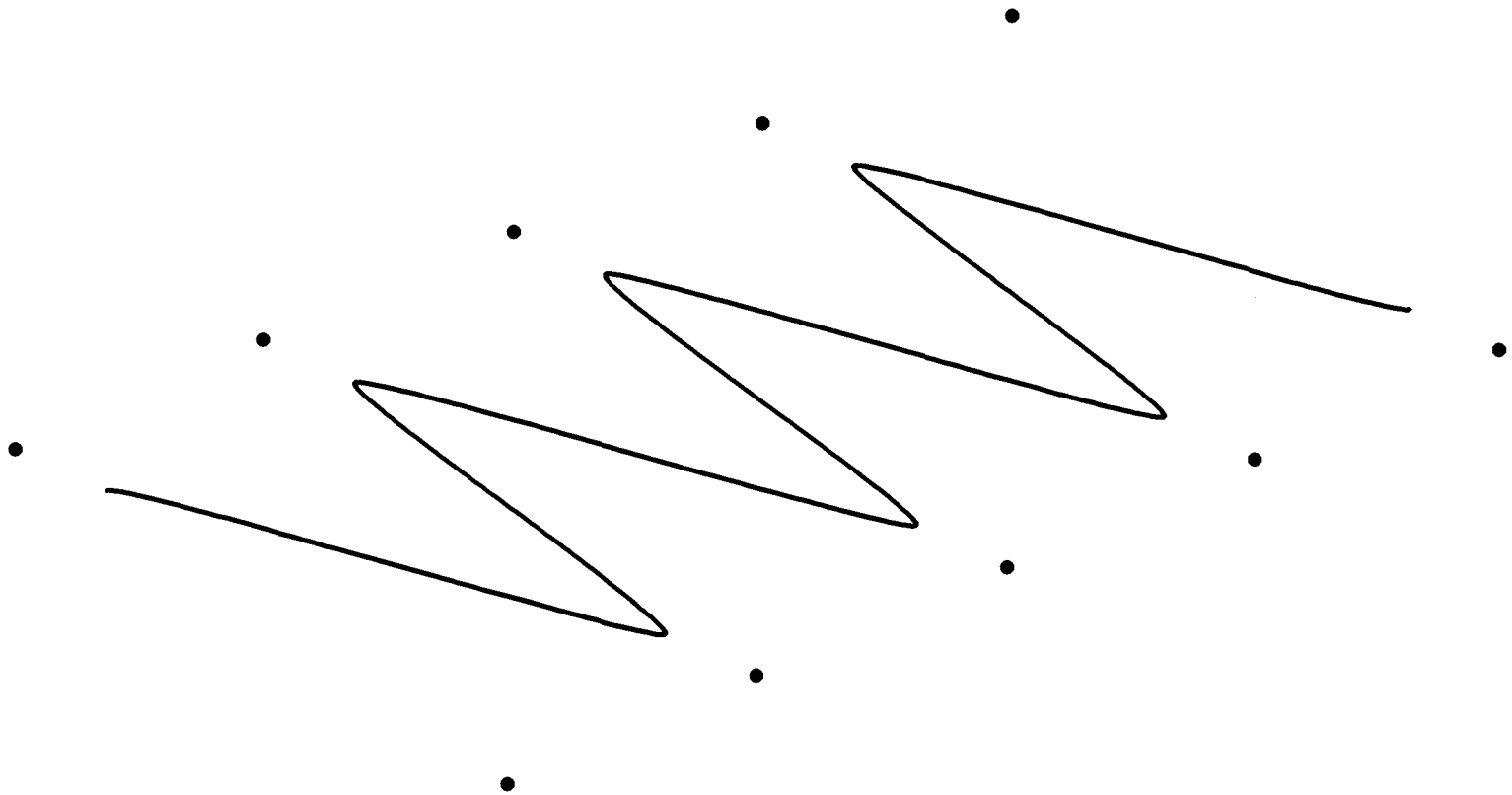
B-spline Curve Segments.
Example 2 data:
Interpolation using Phantom Points.



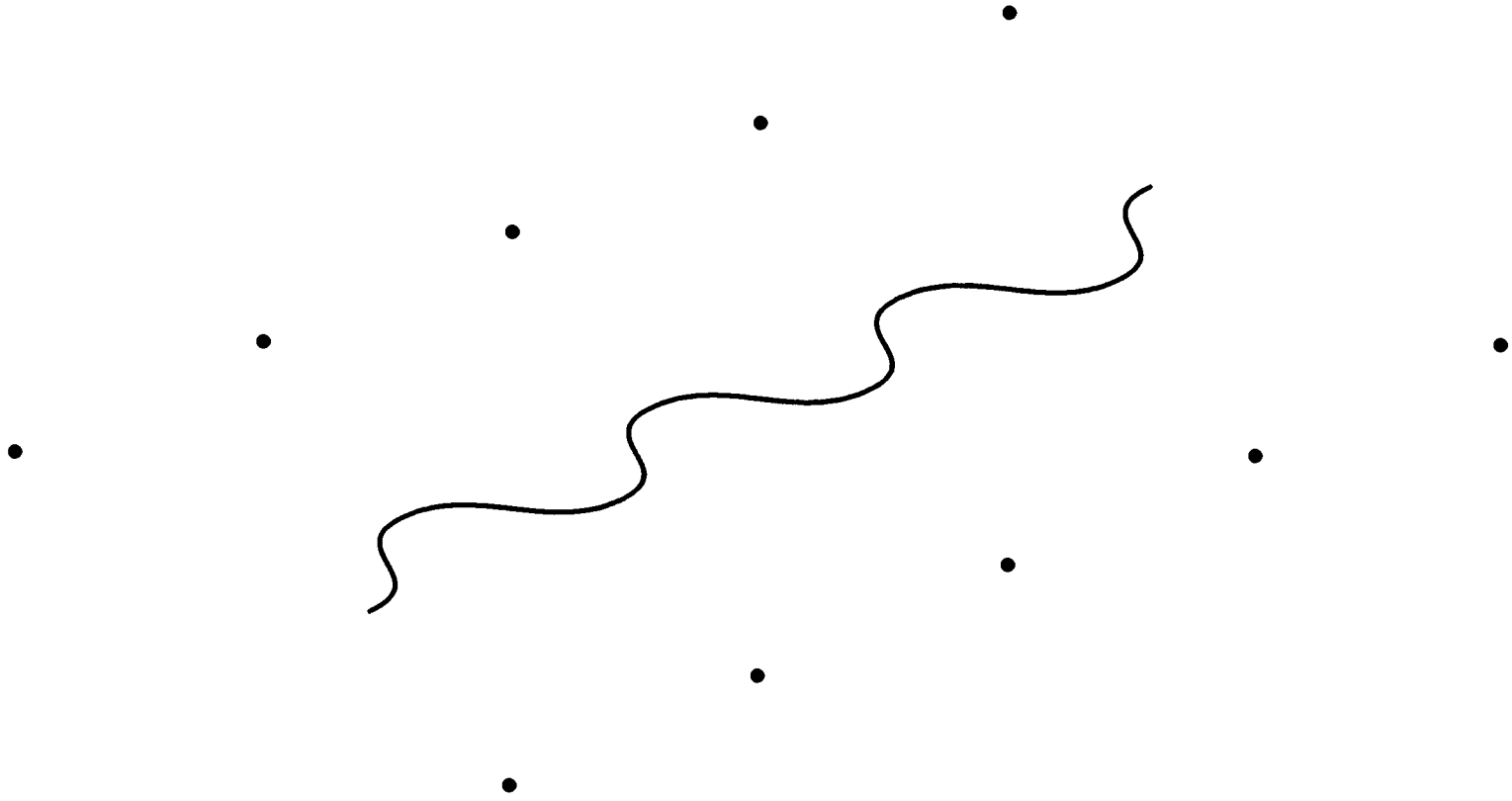
B-spline Curve Segments.
Example 2 data:
Interpolation of control points with odd labels,
using Phantom Points.



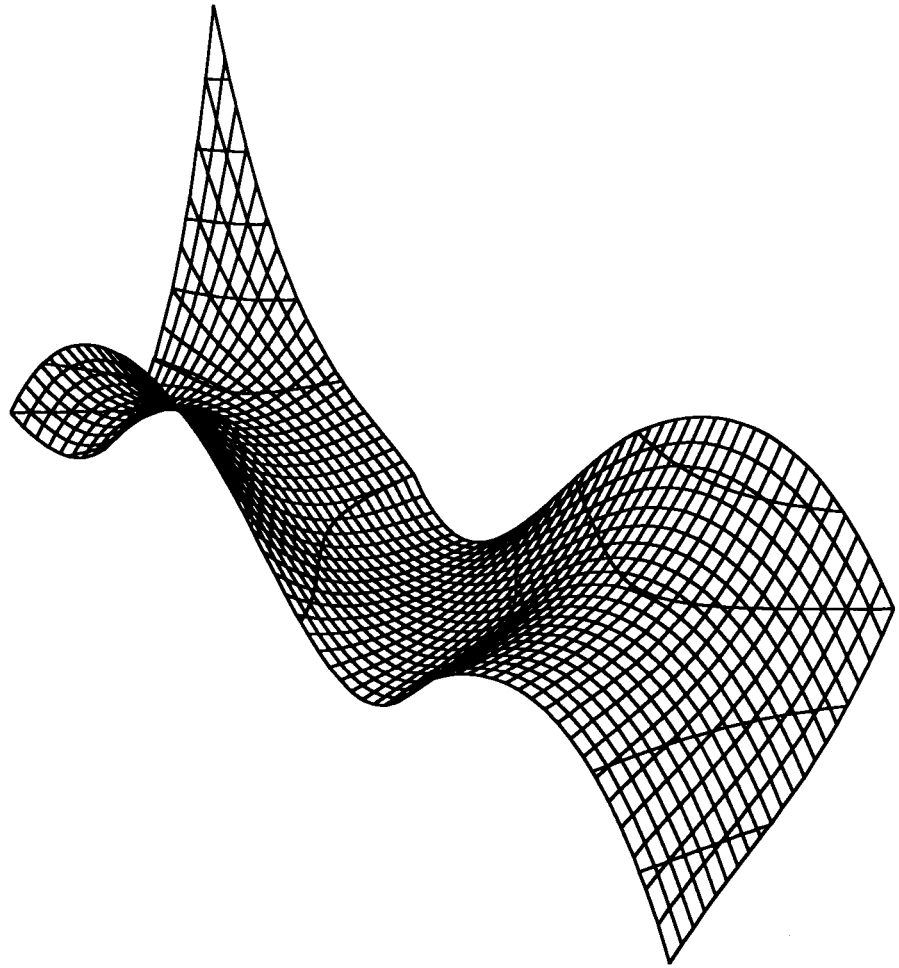
Beta-spline Curve Segments.
Example 3 data:
Tension = 5.



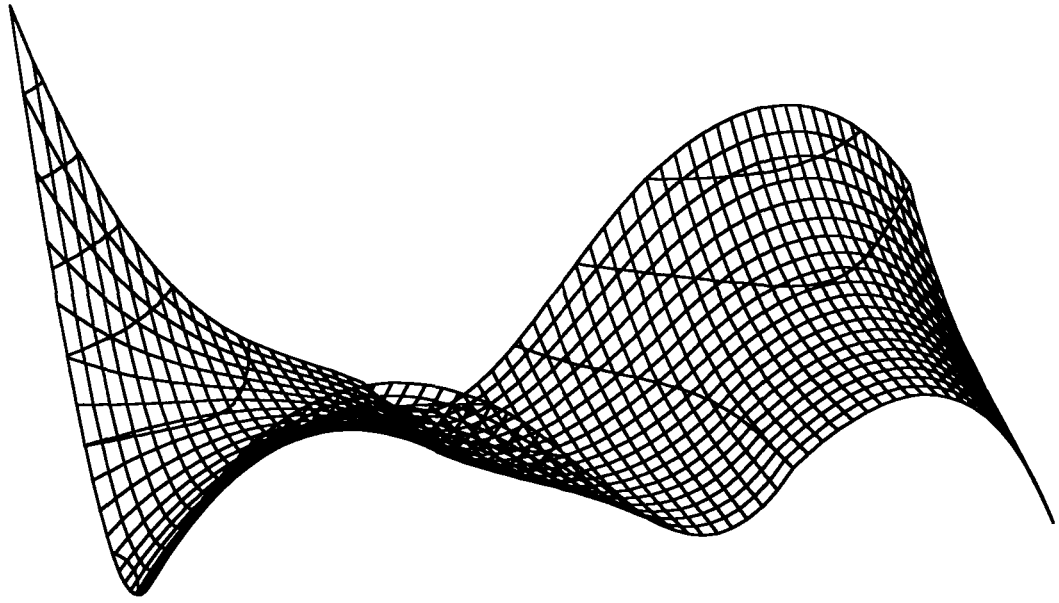
Beta-spline Curve Segments.
Example 3 data:
Tension = 15.



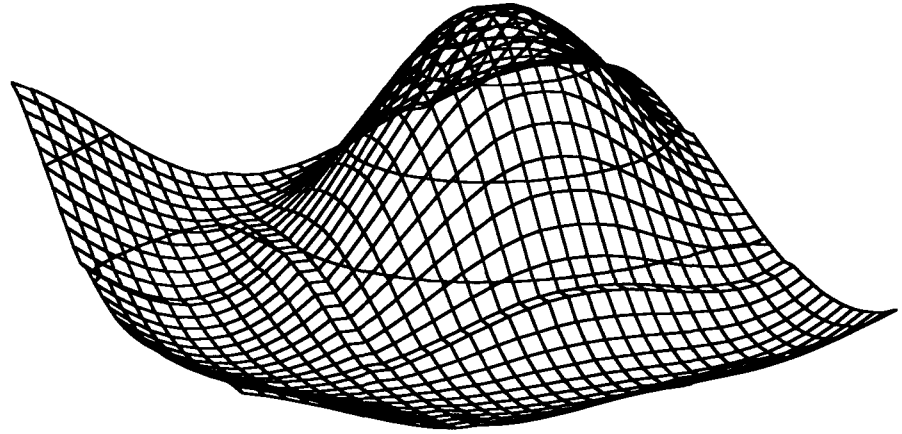
Beta-spline Curve Segments.
Example 3 data:
Tension = -5.



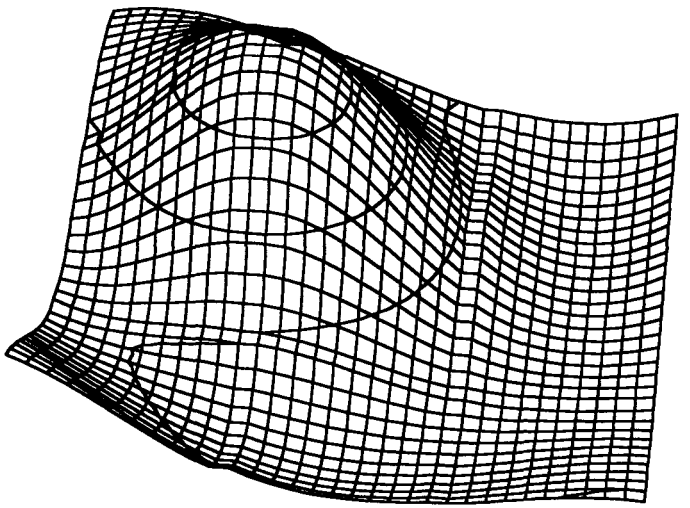
Bi-Cubic Parametric Bezier Surface.
z values over regular grid.
(view 1)



Bi-Cubic Parametric Bezier Surface.
z values over regular grid.
(view 2)



Bi-Cubic Parametric B-Spline Surface.
z values over regular grid.
(view 1)



Bi-Cubic Parametric B-Spline Surface.
z values over regular grid.
(view 2)