NURBS AS THE MODELLING PRIMITIVE FOR RADIOSITY

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

By

George W. Leaver

Department of Computer Science

October 1996

Contents

List of figures **8**

Abstract **11**

Declaration **12**

Copyright & Ownership **13**

Dedication **14**

Acknowledgements **15**

1 Introduction **16**

1.1 NURBS in CAD and Computer Graphics 16

1.2 Illumination Models 17

1.3 Thesis Outline 20

2 Parametric Curves and Surfaces 22

2.1 Parametric Definitions 22

2.2 Power Basis Curves and Surfaces 24

2.3 Bézier Curves and Surfaces 25

2.3.1 Properties of Bézier Curves and Surfaces 26

2.4 B-spline Curves and Surfaces 28

2.4.1 Definition of B-spline Curves and Surfaces 29

2.4.2 Properties of B-spline Basis Functions 31

2.4.3 Properties of B-spline Curves and Surfaces 31

2.4.4 Derivatives of B-spline Curves and Surfaces 33

2.5 NURBS Curves and Surfaces 34

2.5.1 Homogeneous Coordinates 35

2.5.2 Definition of NURBS Curves and Surfaces 36

2.5.3 Properties of Rational B-Spline Basis Functions 38

2.5.4 Properties of NURBS Curves and Surfaces 39

2.5.5 Derivatives of NURBS Curves and Surfaces 40

2.5.6 Knot Insertion 41

2.6 Summary 42

3 Fundamentals of Radiosity **43**

3.1 Terminology and Physical Quantities 43

3.1.1 Local Coordinate Systems 43

3.1.2 Solid Angle 44

3.1.3 Energy and Power 45

3.1.4 Radiance 46

3.1.5 Irradiance 47

3.1.6 Radiosity 47

3.1.7 The BRDF 48

3.2 The Rendering Equation 50

3.2.1 The Radiosity Equation 53

3.3 Classical Radiosity Methods 54

3.4 Form Factors 56

3.4.1 Point-to-Point Form Factor 57

3.4.2 Point-to-Patch Form Factor 57

3.4.3 Patch-to-Patch Form Factor 58

3.4.4 Reciprocity Rules 58

3.4.5 Computing the Form Factor 59

3.5 Solution Methods 61

3.5.1 Jacobi Relaxation 62

3.5.2 Gauss-Seidel Relaxation 63

3.5.3 Progressive Refinement 64

3.6 Summary 65

4 Radiosity and Curved Surfaces 67

4.1 Previous Work 69

4.2 The Constant Radiosity Assumption 74

4.3 Higher Order Radiosity 76

4.3.1 Fredholm Integral Equations 77

4.3.2 The Galerkin Method 81

4.4 Galerkin Radiosity 82

4.4.1 Basis Sets 85

4.4.2 Numerical Quadrature 87

4.4.3 Shadow Masks 88

4.5 Summary 91

5 Implementation Issues 92

5.1 Describing Scenes 92

5.1.1 Additional Surface Parameters 93

5.1.2 NURBS Functions 94

5.2 Reflectance and Emittance Functions 94

5.3 Basis Functions and Quadrature 96

5.3.1 Legendre Polynomials 97

5.3.2 Numerical Quadrature 98

5.4 Solution Method 101

5.5 Ray Tracing NURBS 102

5.5.1 Bounding Box Intersection Tests 103

5.5.2 Surface Intersection Test 104

5.6 Rendering 105

5.7 Shadow Masks 106

5.8 Summary 107

6 Results and Conclusions 108

6.1 Results 108

6.2 Conclusions 116

6.3 Future Work 117

A Example MSDL File 120

Bibliography 125

List of Figures

2.1 The cubic Bernstein polynomials. 26

2.2 A cubic Bézier curve with its control polygon and control points (+). 27

2.3 Non-zero second-degree B-spline basis functions, with discontinuities at $u = 0.8$. The basis functions are defined on the knot vector $\{0, 0, 0, 0.2, 0.4, 0.6, 0.8, 0.8, 1, 1, 1\}$. 30

2.4 A cubic NURBS curve with its control polygon, control points (+) and points on the curve corresponding to knot values (o). The curve is defined on the knot vector $\{0, 0, 0, 0, 0.2, 0.4, 0.7, 0.9, 1, 1, 1, 1\}$. 37

2.5 A biquadratic NURBS surface with its control polygon and control points (+). The surface is defined on knot vectors $U = \{0, 0, 0, 0.25, 0.75, 1, 1, 1\}$ and $V = \{0, 0, 0, 0.25, 0.75, 1, 1, 1\}$. 38

3.1 A spherical coordinate system centred at a differential element $\mathbf{x}$. 44

3.2 The solid angle subtended by an object. 45

3.3 Terms used in the definition of the BRDF. 49

3.4 The relationship between three points $\mathbf{x}$, $\mathbf{x}'$ and $\mathbf{x}''$ in the environment. The radiance leaving $\mathbf{x}$, towards $\mathbf{x}''$, is considered in terms of the radiance emitted by $\mathbf{x}$ and radiance arriving at $\mathbf{x}$ from $\mathbf{x}'$. . 52

3.5 An ideal diffuse reflector reflects light equally in all directions, rather than in some direction dependent on the direction of incoming light. 53

3.6 Approximating the radiosity function with constant patch radiosities. 56

3.7 Pseudocode for progressive refinement in conventional radiosity. . 65

4.1 Displacing the hemisphere away from a patch (after [38]). 72

4.2 A constant basis function defined over the extent of a patch. . . . 76

5.1 A NURBS surface function used to specify varying reflectance or emittance over a surface. 95

5.2 Six unnormalised basis functions of a second degree basis set. . . . 98

5.3 Pseudocode for progressive refinement in Galerkin radiosity. . . . 101

6.1 Test scene *flat*. Two planar NURBS surface. 110

6.2 Test scene *flat(coloured)*. Two planar NURBS surfaces. The light source uses a NURBS surface function to specify non-constant emittance. 111

6.3 Emittance of light source in figure 6.2. 111

6.4 Test scene *box*. A traditional radiosity box. 112

6.5 Test scene *spike*. The centre control point of the floor in figure 6.4 has been raised, giving a spike in the floor. Shadow discontinuities are badly modelled resulting in Gibbs ringing artifacts. 112

6.6 Test scene *deformed*. The centre control points of other surfaces in figure 6.4 have been moved. Bright artifacts are present due to insufficient solution degrees being specified for the curved surfaces. 113

6.7 Test scene *depression*. The NURBS surface has a depression and curled corners. 113

6.8 Test scene *even*. The NURBS surface has an even parameterisation resulting in correctly distributed sample points. 114

6.9 Test scene *uneven*. The NURBS surface has an uneven parameterisation resulting in incorrectly distributed sample points. 114

Abstract

UNIVERSITY OF MANCHESTER

ABSTRACT OF THESIS submitted by **George W. Leaver** for the Degree of Master of Science and entitled **NURBS as the Modelling Primitive for Radiosity**

Month and Year of Submission: October 1996

The radiosity method has been the focus of much research to reduce the time and storage costs associated with generating realistic computer images. However, the majority of radiosity research is based on polygonal models of scenes, resulting in models with vast numbers of elements. **N**on **U**niform **R**ational **B**-**S**plines (NURBS) provide a mathematical basis for representing many forms of surfaces, including conics, planes and free-form surfaces. NURBS allow scenes of complex geometry to be modelled with little storage costs.

Radiosity methods capable of using NURBS surfaces directly, avoiding polygonalisation, are examined and the implementation of one such method is presented.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright & Ownership

Dedication

To my parents.

Acknowledgements

I would like to express my sincere thanks to my supervisor, Mr. W. T. Hewitt, for his guidance and support during my research and the writing of this thesis.

I would also like to thank other members of the Computer Graphics Unit for their help, including: John Irwin, Paul Lever, Mary McDerby, Helen Morphet and Martin Preston.

My friends, in particular those inhabiting the Vision Lab at strange hours of the night, for making this an enjoyable year.

My parents, for their support and encouragement throughout my years at university.

I would also like to acknowledge the Engineering and Physical Sciences Research Council for financial assistance.

Chapter 1

Introduction

Since its introduction three decades ago, a goal of computer graphics research has been to produce photo-realistic images. This goal is now being achieved, partly through advances in computer hardware, and partly through an increase in accuracy of illumination models. The boundaries between disciplines within computer graphics are becoming more blurred through such advances. For example, Virtual Reality (VR) systems are able to use illumination models that were considered too computationally expensive several years ago [2][51]. The modelling capabilities of Computer Aided Design (CAD) systems are also being combined with advances in image synthesis.

1.1 NURBS in CAD and Computer Graphics

Non **U**niform **R**ational **B**-**S**plines (NURBS) provide a powerful mathematical representation of curves and surfaces. Such forms have long been exploited in CAD systems. The ability to represent exactly many common shapes required in a design environment, such as conic sections, spheres, planes, free-form shapes and shapes exhibiting discontinuities makes NURBS an attractive modelling primitive. The computer aided manufacturing (CAM) and computer aided engineering (CAE) industry have made use of NURBS since the early 1980s. The Structural Dynamics Research Corporation used NURBS in their "I-deas/Geomod" geometric modelling system in 1983. NURBS have also been used by the Boeing company in aircraft design, using their "Tiger" system [45].

While NURBS have been accepted as the standard modelling primitive in CAD/CAM/CAE environments, they have not gained as much popularity in realistic image synthesis methods. Rather, the polygon has been the dominant modelling primitive in such methods.

1.2 Illumination Models

While geometric primitives such as NURBS may be used to model a scene, some method of rendering the scene is required. For realistic image synthesis we require that the rendering process produces an image comparable to the physical scene. Of course the model stored in a computer may be an imaginary scene, but the rendered image of this scene should lead a viewer to believe they are viewing a physical scene. The accuracy of the rendered image depends on, among other things, the accuracy of the illumination model used during the rendering process. An illumination model determines how light is distributed at a surface with respect to the incident energy arriving from light sources in the environment. This thesis is concerned with one such illumination model, namely *radiosity*. Early illumination models, such as the Gouraud [30] and Phong [44] models, considered light arriving at an object from point light sources only. These models are classed as *local* illumination models because they take no account of reflections from other *objects* in the environment. Light arriving from other objects is simply approximated by an "ambient" term and effects such as shadows are ignored. These early models are also empirical—the models were designed to produce visually acceptable results without considering the physical process behind the transmission of light.

To overcome some of the limitations of local illumination models, *global* illumination models were developed. Global illumination models consider other objects in the environment and so are capable of modelling more illumination effects than local models. The first global illumination model in computer graphics was given by Whitted [66]. His *ray tracing*[1] method modelled the specular reflection and transmission of light between objects, and also took account of shadows (although only sharp shadows were generated). The method was, however, based on ray optics rather than the physical process of light transmission and so *physically* realistic images were not produced. Despite its limitations, ray tracing remains an extremely popular method in the production of realistic images. The use of NURBS as the modelling primitive for ray tracing is discussed by Cook in [15].

While ray tracing was particularly effective at rendering scenes containing shiny, highly reflective objects, the interaction of light between diffuse surfaces was

[1]Ray tracing was first used by Appel [3] in 1968 for hidden surface removal. However, this discussion considers the form of ray tracing use in global illumination.

crudely approximated using the same technique as in local illumination models. The *radiosity* global illumination model was introduced independently by Goral [26] in 1984 and Nishita and Nikamae [41] in 1985 to consider the interaction of light between diffuse surfaces[2]. This work was developed from that of thermal engineers [40][18][36], and allowed effects such as colour bleeding[3] and soft shadows to be modelled. The radiosity method allows any surface in the environment to be defined as a light source or to *become* a light source, having been illuminated by other light sources. Hence surfaces may be illuminated through the direct transfer of light or indirect transfer of light—by light which reflects off other surfaces.

As with the thermal radiative methods on which the radiosity method was based, a key computation in the method is determining how much energy leaves one surface and arrives at another. This computation also contributes significantly to the solution time of the radiosity method [56]. In order to simplify the situation, polygons are typically used as the modelling primitive. This allows existing, polygon based algorithms to be utilised [12]. Today's graphics workstations also allow many polygon based calculations to be executed in hardware, providing significant efficiency improvements.

[2]Goral [26] is often cited as the originator of this work. However, Numan and Moore [43] considered radiant interchange among diffusely emitting and reflecting surfaces in 1982. Their work was primarily concerned with the form factor between partially obscured surface rather than rendering. They did suggests, though, that the method might be extended to illumination applications.

[3]Colour bleeding is the process in which a surface is illuminated by light reflecting off another surface, and so may appear to have the colour of the reflecting surface. Watt [65] discusses whether colour bleeding should be modelled at all, stating that the *colour constancy* of the human visual system ensures that the colour of an object is perceived as a permanent unchanging attribute. This thesis, however, is concerned with the radiosity method, rather than perceptual issues and so colour bleeding will be modelled as a consequence of using the radiosity method.

Polygons are not without their disadvantages. Modelling scenes using polygons often results in scene databases containing vast numbers of polygons, especially when curved surfaces are approximated by many small polygons. Examining scene files generated by the Computer Graphics Unit at the University of Manchester, spheres are typically approximated by 1,000 triangles. A room is modelled using approximately 100,000 quadrilaterals. Also, designing and constructing the scene using polygons is not very intuitive. A method of incorporating NURBS surfaces into the radiosity method is needed.

1.3 Thesis Outline

This thesis examines radiosity and curved surfaces. An implementation based on the method presented by Zatz [68][69] is given, using NURBS as the modelling primitive.

Chapter 2 gives an introduction to parametric surface representation. Simple representations, and their problems, are discussed before NURBS are introduced. Properties of, and operations on NURBS are discussed, for use in later chapters.

Chapter 3 focuses on the radiosity method. The physical process of the transmission of light is considered, resulting in an integral equation describing the distribution of light in an environment. A review of classical radiosity methods used to simplify the integral equation, resulting in polygon-based methods, is presented.

Chapter 4 looks at radiosity with curved surfaces. Alternative forms of classical radiosity, adapted to incorporate curved surfaces are discussed before *higher order* methods for solving the integral equation are presented. This results in *higher order radiosity*, which forms the basis of Zatz's method and the implementation in this thesis.

Chapter 5 presents the implementation details of the work in this thesis.

Chapter 6 is the concluding chapter, presenting results of the implementation, along with possible extensions to the method.

Chapter 2

Parametric Curves and Surfaces

This chapter introduces parametric curves and surfaces commonly used in computer aided design (CAD). The definitions and properties of various simple representations are introduced before NURBS, the parametric surfaces used in this thesis, are defined. Only a brief introduction to NURBS, relevant to this thesis, is made here. For more detailed discussions see [19][52]. Les Piegl and Wayne Tiller have produced a most thorough work on NURBS [48], detailing many aspects of NURBS.

2.1 Parametric Definitions

Parametric functions are one of the most common methods of representing curves and surfaces [48]. In general, the parametric description of a curve in 2D Euclidean space, $\mathbb{E}^2$, is given by the vector-valued function

$$\mathbf{C}(u) = (x(u), y(u)) \qquad a \leq u \leq b \tag{2.1}$$

where u is an independent variable defined on the interval $[a,b]$ and $x(u)$ and $y(u)$ are parametric functions defining the curve $\mathbf{C}(u)$. A surface is represented by a vector-valued function $\mathbf{S}(u,v)$ of two parameters defined over a region $\mathcal{R}$, i.e.,

$$\mathbf{S}(u,v) = (x(u,v), y(u,v), z(u,v)) \qquad (u,v) \in \mathcal{R}. \tag{2.2}$$

For example, a unit sphere centred at the origin is given by

$$x(u,v) = sin(u)cos(v)$$
$$y(u,v) = sin(u)sin(v)$$
$$z(u,v) = cos(u) \qquad 0 \leq u \leq \pi,\ 0 \leq v \leq 2\pi.$$

Using arbitrary parametric function, $x(u)$, $y(u,v), \ldots$, in (2.1) and (2.2) allows a wide variety of curves and surfaces to be defined. However, in a geometric modelling environment, curves and surfaces are usually described in terms of basis functions and basis function coefficients. Curves are then given by

$$\mathbf{C}(u) = (x(u), y(u)) = \sum_{i=0}^{n} \mathbf{a}_i f_i(u) \qquad a \leq u \leq b \tag{2.3}$$

where $\mathbf{a}_i$ are vectors of coefficients and $f_i(u)$ are basis functions. (2.3) is easily extended to surfaces to give the tensor product form[1] of surfaces. Tensor products use bivariate basis functions defined by products of univariate basis functions. Basis function coefficients are arranged in matrix form, giving

$$\mathbf{S}(u,v) = (x(u,v), y(u,v), z(u,v)) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{b}_{i,j} f_i(u) g_j(v) \qquad (u,v) \in \mathcal{R}. \tag{2.4}$$

Defining a curve or surface is now a matter of choosing the basis functions and their coefficients. A brief discussion of two simple classes of basis functions follows, before B-splines and NURBS are discussed in more detail.

[1]In [46], Piegl gives many other forms of surface representation.

2.2 Power Basis Curves and Surfaces

A class of functions which are efficiently processed by computer and require little storage space for their description are the polynomials. An nth degree polynomial curve, in $\mathbb{E}^3$, may be described in *power basis* form using the parametric functions

$$x(u) = \sum_{i=0}^{n} x_i u^i,$$
$$y(u) = \sum_{i=0}^{n} y_i u^i,$$
$$z(u) = \sum_{i=0}^{n} z_i u^i \qquad 0 \leq u \leq 1.$$

Using vector notation by writing $\mathbf{a}_i = (x_i, y_i, z_i)$, the curve is given by

$$\mathbf{C}(u) = \sum_{i=0}^{n} \mathbf{a}_i u^i \qquad 0 \leq u \leq 1. \tag{2.5}$$

Hence the basis functions in (2.5) are the polynomials u^i and the coefficients are the vectors $\mathbf{a}_i$. Using (2.4), the power basis surface is given by the tensor product

$$\mathbf{S}(u,v) = (x(u,v), y(u,v), z(u,v)) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{b}_{i,j} u^i v^j \qquad 0 \leq u, v \leq 1, \tag{2.6}$$

where $\mathbf{b}_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j})$. The interval over which parametric variables are defined is arbitrary. However, it is often normalised to $[0,1]$.

Despite the simplicity of the power basis form, it does have the following disadvantages which make it unsuitable for use in geometric design:

- The basis function coefficients $\mathbf{a}_i$ and $\mathbf{b}_{i,j}$ provide little geometric information and so it is unnatural for interactive shape design.
- Certain shapes (conics) cannot be represented exactly and so must be approximated.

- Algorithms for processing power basis polynomials are prone to round-off error and so are numerically unstable [16].

- Power basis forms are not closed under affine transformations. An affine transformation is composed of rotation/scaling/shearing/translation transformations. For a curve or surface to be closed under such transformations, applying the transformation to the basis function coefficients transforms the entire curve or surface with that transformation. Since this property does not hold for power basis forms, a curve or surface can only be transformed by evaluating each point on the curve or surface and then transforming each point.

2.3 Bézier Curves and Surfaces

Perhaps the greatest disadvantages of power basis curves and surfaces is their inability to be used intuitively in the design process and that they are not closed under affine transformations. Bézier curves alleviate these problems. An nth degree Bézier curve is defined by

$$\mathbf{C}(u) = \sum_{i=0}^{n} B_{i,n}(u)\mathbf{P}_i \qquad 0 \leq u \leq 1.$$

The basis functions $B_{i,n}$, are defined by the Bernstein polynomials

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i} \qquad 0 \leq u \leq 1. \tag{2.7}$$

Figure 2.1 shows the cubic Bernstein polynomials. The coefficients $\mathbf{P}_i$ now take on a more geometric meaning and are referred to as *control points*. Figure 2.2 shows how the control points relate to a cubic Bézier curve. A tensor-product

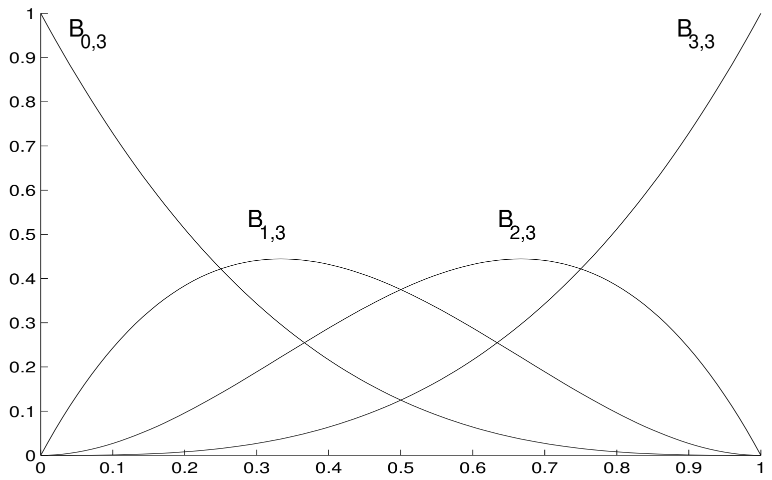

Figure 2.1: The cubic Bernstein polynomials.

Bézier surface is defined by

$$\mathbf{S}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u) B_{j,m}(v) \mathbf{P}_{i,j} \qquad 0 \le u,\ v \le 1$$

where $\mathbf{P}_{i,j}$ are arranged (topologically) in a bidirectional $n \times m$ net, and the basis functions are given by the product of the univariate Bernstein basis functions, $B_{i,n}(u)B_{j,m}(v)$.

2.3.1 Properties of Bézier Curves and Surfaces

Bézier curves and surfaces have the following geometric properties, as a result of analytic properties of the Bernstein basis functions.

- Interactivity: moving the control points gives gives an intuitive change in curve shape.
- Endpoint interpolation: the curve always passes through the first and last control points, i.e., $\mathbf{P}_0 = \mathbf{C}(0)$ and $\mathbf{P}_n = \mathbf{C}(1)$.

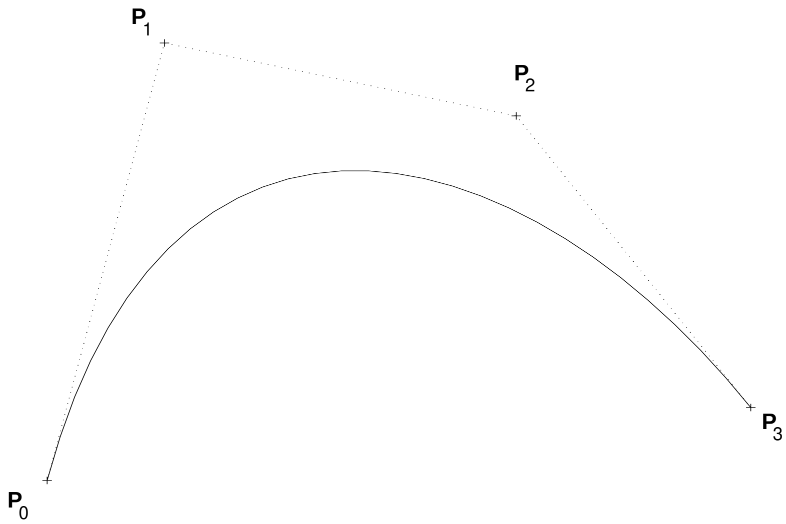

Figure 2.2: A cubic Bézier curve with its control polygon and control points (+).

- Endpoint tangents: the tangent directions to the curve at $\mathbf{C}(0)$ and $\mathbf{C}(1)$ are $\mathbf{P}_1 - \mathbf{P}_0$ and $\mathbf{P}_n - \mathbf{P}_{n-1}$ respectively. This allows certain endpoint conditions to be placed on curves.

- Convex hull property: a curve lies entirely within the convex hull of its defining control points.

- Variation diminishing property: no line (plane) intersects a 2D (3D) curve more times than it intersects the curve's control polygon. In [48], Piegl and Tiller describe this property as ensuring that a curve does not wiggle more than its control polygon.

- Closed under affine transformations: Unlike power basis forms, Bézier forms may be transformed by applying affine transformations only to the control points.

These properties will be discussed further in section 2.4.2 when B-splines are introduced.

Despite the advantages of Bézier curves and surfaces over the power basis form, the Bézier form is not without its problems:

- Bézier curves and surfaces exhibit no local control, i.e., moving one control point affects the entire curve. Local control is often desired in CAD applications.

- Complex shapes require Bézier forms defined using many control points, resulting in curves and surfaces of high degree. Evaluating points on such forms becomes inefficient and these forms are unstable in computational algorithms. Shape design becomes unintuitive when many control points are required because of the lack of local control.

2.4 B-spline Curves and Surfaces

The previous sections briefly discussed two simple forms of curves and surfaces. This section discusses non-rational B-spline[2] curves and surfaces, the non-rational form of NURBS.

Despite the interactivity of Bézier curves and surfaces, the lack of local control and the need for high degree basis functions and many control points to represent complex shapes limits the use of Bézier forms in CAD applications. These limitations may be overcome, to a certain extent, by joining several Bézier curves or surfaces. However, to ensure continuity between curves or surfaces, restrictions must be placed on the location of certain control points. B-spline curves and

[2] The prefix "non-rational" is usually dropped. A B-spline form is understood to be a non-rational B-spline form.

surfaces overcome these limitations by introducing piecewise polynomial curves with local control and the ability to control the continuity between sections of the curve.

2.4.1 Definition of B-spline Curves and Surfaces

A pth degree B-spline curve is defined by

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i \qquad a \leq u \leq b \tag{2.8}$$

where $\mathbf{P}_i$ are the control points in 3D Euclidean space $\mathbb{E}^3$, $N_{i,p}(u)$ are the pth degree normalised B-spline basis functions. Normalised B-spline basis functions are defined over a knot vector $U = \{u_0, \ldots, u_m\}$, a nondecreasing sequence of real numbers. The basis functions are given by the Cox-deBoor recurrence relation [7]

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) \\ &\quad + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u). \end{aligned} \tag{2.9}$$

Throughout this chapter it is assumed that the knot vector is of the form

$$U = \{\underbrace{a, \ldots, a}_{p+1}, u_{p+1}, \ldots, u_{m-p-1}, \underbrace{b, \ldots, b}_{p+1}\}. \tag{2.10}$$

In most applications, (2.10) has $a = 0$ and $b = 1$, and this will be the form assumed for the remainder of this chapter. However, as we shall see in section 4.4, we will require $a = -1,\ b = 1$. Note that (2.9) may yield 0/0 in which case the quotient is defined to be zero. Figure 2.3 shows the non-zero second-degree B-spline basis functions defined on the knot vector $\{0, 0, 0, 0.2, 0.4, 0.6, 0.8, 0.8, 1, 1, 1\}$.

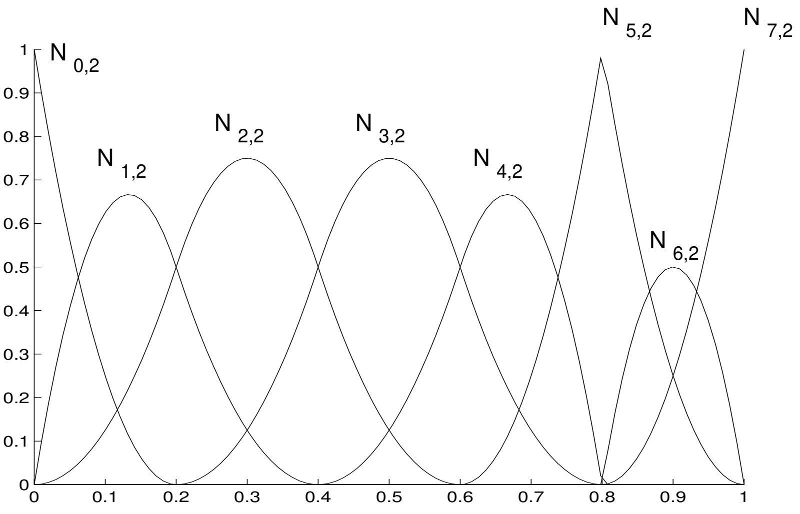

Figure 2.3: Non-zero second-degree B-spline basis functions, with discontinuities at $u = 0.8$. The basis functions are defined on the knot vector $\{0, 0, 0, 0.2, 0.4, 0.6, 0.8, 0.8, 1, 1, 1\}$.

Note the discontinuity in the basis functions $N_{4,2}$, $N_{5,2}$ and $N_{6,2}$ at the repeated knot $u = 0.8$.

A tensor-product B-spline surface is defined by

$$\mathbf{S}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) \mathbf{P}_{i,j} \tag{2.11}$$

where $\mathbf{P}_{i,j}$ are arranged in a topological control net in $\mathbb{E}^3$, and $N_{i,p}(u)$ and $N_{j,q}(v)$ are the normalised B-spline basis functions of degree p and q in the u and v directions respectively. $N_{i,p}(u)$ is defined over the knot vector

$$U = \{\underbrace{0, \ldots, 0}_{p+1}, u_{p+1}, \ldots, u_{r-p-1}, \underbrace{1, \ldots, 1}_{p+1}\},$$

and $N_{j,q}(v)$ is defined over the knot vector

$$V = \{\underbrace{0, \ldots, 0}_{q+1}, v_{q+1}, \ldots, v_{s-q-1}, \underbrace{1, \ldots, 1}_{q+1}\}.$$

For a curve $\mathbf{C}(u)$, the number of knots, m, the degree of the curve, p, and the number of control points, $n + 1$, are related by $m = p + n + 1$. For a surface $\mathbf{S}(u, v)$, this equation takes the form $r = p + n + 1$ and $s = q + m + 1$.

2.4.2 Properties of B-spline Basis Functions

B-spline basis functions have the following analytic properties:

- Non-negativity: $N_{i,p}(u) \geq 0$ for all i, p, and u. For tensor product bases, $N_{i,p}(u)N_{j,q}(v) \geq 0$ for all i, j, p, q, u, and v.

- Local support: $N_{i,p}(u) = 0$ for $u \notin [u_i, u_{i+p+1}]$. Also, $N_{i,p}(u)N_{j,q}(v) = 0$ for $(u, v) \notin [u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$.

- Partition of unity: $\sum_{i=0}^{n} N_{i,p}(u) = 1$ for all $u \in [0, 1]$. Also, $\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v) = 1$ for all $(u, v) \in [0, 1] \times [0, 1]$.

- Differentiability: $N_{i,p}(u)$ is infinitely continuously differentiable in an interior knot span $[u_i, u_{i+1})$. $N_{i,p}(u)$ is $p - k$ times continuously differentiable at an interior knot value, where k is the multiplicity of the knot. All partial derivatives of $N_{i,p}(u)N_{j,q}(v)$ exist in an interior rectangle formed by knot spans $[u_i, u_{i+1}) \times [v_j, v_{j+1})$. At an interior knot value u (v), $N_{i,p}(u)N_{j,q}(v)$ is $p - k$ $(q - k)$ times differentiable in the u (v) direction, where k is the multiplicity of the knot.

- Extrema: if $p > 0$, then $N_{i,p}(u)$ attains exactly one maximum value. If $p > 0$ and $q > 0$, then $N_{i,p}(u)N_{j,q}(v)$ attains exactly one maximum value.

2.4.3 Properties of B-spline Curves and Surfaces

The above analytic properties of the B-spline basis functions contribute to the useful geometric properties of B-spline curves and surfaces. They are:

- Strong convex hull property: the entire curve $\mathbf{C}(u)$ is contained in the convex hull of its control polygon. Also, if $u \in [u_i, u_{i+1})$ for $p \le i < n$, then $\mathbf{C}(u)$ is contained in the convex hull of the control points $\mathbf{P}_{i-p}, \ldots, \mathbf{P}_i$. The whole surface $\mathbf{S}(u,v)$ is contained in the convex hull of its control net. Also, if $(u,v) \in [u_{i_0}, u_{i_0+1}) \times [v_{j_0}, v_{j_0+1})$ for $p \le i_0 < n$ and $q \le j_0 < m$, then $\mathbf{S}(u,v)$ is contained in the convex hull formed by $\mathbf{P}_{i,j}$, $i_0 - p \le i \le i_0$ and $j_0 - q \le j \le j_0$. These results can be derived from the non-negativity, partition of unity and local support of B-spline basis functions.

- Endpoint interpolation: $\mathbf{C}(0) = \mathbf{P}_0$ and $\mathbf{C}(1) = \mathbf{P}_n$. Surfaces interpolate corner control points: $\mathbf{S}(0,0) = \mathbf{P}_{0,0}$, $\mathbf{S}(1,0) = \mathbf{P}_{n,0}$, $\mathbf{S}(0,1) = \mathbf{P}_{0,m}$, $\mathbf{S}(1,1) = \mathbf{P}_{n,m}$.

- Local shape modification: moving a control point $\mathbf{P}_i$ affects the curve $\mathbf{C}(u)$ only in knot spans $[u_i, u_{i+p+1})$. Moving a control point $\mathbf{P}_{i,j}$ affects the surface $\mathbf{S}(u,v)$ only in the rectangle $[u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$. These results can be derived from the locality of B-spline basis functions.

- Variation diminishing property: no line (plane) intersects a 2D (3D) curve more times than it intersects the curve's control polygon. There is no known generalisation of this property to B-spline surfaces.

- Closed under affine transformations: Affine transformations were described briefly in section 2.2. More formally, an affine transformation, $\Phi{:}\mathbb{E}^3 \to \mathbb{E}^3$, can be expressed as $\Phi\mathbf{r} = L\mathbf{r} + \mathbf{v}$, where L is a 3×3 matrix representing a linear transformation including rotation, scaling and shearing; $\mathbf{v}$ is a vector representing a translation; $\mathbf{r}$ is a point in $\mathbb{E}^3$. B-spline curves and surfaces are closed under affine transformations and so affine transformations may be applied to curves and surfaces by applying the transformation to the control points. This result can be derived from the partition of unity of B-spline basis functions. However, this property does not hold when non-linear transformations, such as perspective transformations, are applied.

- Same continuity/differentiability as B-spline basis functions: $\mathbf{C}(u)$ is a linear combination of B-spline basis functions and so the continuity and differentiability of $\mathbf{C}(u)$ is given by that of $N_{i,p}(u)$. $\mathbf{C}(u)$ is infinitely differentiable in an interior knot span $[u_i, u_{i+1})$, and $p-k$ times continuously differentiable at an interior knot of multiplicity k. Similarly, $\mathbf{S}(u,v)$ is infinitely differentiable in an interior rectangle formed by knot spans $[u_i, u_{i+1}) \times [v_j, v_{j+1})$, and $p-k$ $(q-k)$ times differentiable in the u (v) direction at an interior u (v) knot, where k is the multiplicity of the knot. The ability to restrict the continuity of curves and surfaces allows a designer to introduce discontinuities into the curve or surface. Hence both smooth and discontinuous surfaces (for example, the edge of a cube) may be modelled.

2.4.4 Derivatives of B-spline Curves and Surfaces

The kth derivative, $\mathbf{C}^{(k)}(u)$, of the curve $\mathbf{C}(u)$ is given by

$$\mathbf{C}^{(k)}(u) = \sum_{i=0}^{n-k} N_{i,p-k}(u)\mathbf{P}_i^{(k)} \tag{2.12}$$

where

$$\mathbf{P}_i^{(k)} = \begin{cases} \mathbf{P}_i & k = 0 \\ \frac{p-k+1}{u_{i+p+1}-u_{i+k}}(\mathbf{P}_{i+1}^{(k-1)} - \mathbf{P}_i^{(k-1)}) & k > 0 \end{cases}$$

and

$$U^{(k)} = \{\underbrace{0,\ldots,0}_{p-k+1}, u_{p+1},\ldots,u_{m-p-1},\underbrace{1,\ldots,1}_{p-k+1}\}.$$

From (2.12), $\mathbf{C}^{(k)}(u)$ is a B-spline curve of degree $p-k$.

All partial derivatives of a surface $\mathbf{S}(u,v)$ are given by

$$\frac{\partial^{k+l}}{\partial^k u \partial^l v}\mathbf{S}(u,v) = \sum_{i=0}^{n-k}\sum_{j=0}^{m-l} N_{i,p-k}(u)N_{j,q-l}(v)\mathbf{P}_{i,j}^{(k,l)} \tag{2.13}$$

where

$$\mathbf{P}_{i,j}^{(k,l)} = (q-l+1)\frac{\mathbf{P}_{i,j+1}^{(k,l-1)} - \mathbf{P}_{i,j}^{(k,l-1)}}{v_{j+q+1}-v_{j+1}}$$

with

$$U = \{\underbrace{0,\ldots,0}_{p-k+1}, u_{p+1},\ldots,u_{r-p-1},\underbrace{1,\ldots,1}_{p-k+1}\}$$

and

$$V = \{\underbrace{0,\ldots,0}_{q-l+1}, v_{q+1},\ldots,v_{s-q-1},\underbrace{1,\ldots,1}_{q-l+1}\}.$$

From (2.13), $\frac{\partial^{k+l}}{\partial^k u \partial^l v}\mathbf{S}(u,v)$ is a B-spline surfaces of degree $p-k$ in the u direction and degree $q-l$ in the v direction.

2.5 NURBS Curves and Surfaces

The previous section introduced B-spline curves and surfaces, which were shown to have several advantages over Bézier and power basis forms. B-splines are not without their problems though, the main one being they are not closed under perspective transformations. The rational counterpart of B-splines are NURBS and are developed using *homogeneous coordinates*.

2.5.1 Homogeneous Coordinates

The homogeneous coordinate representation of an object in 3D space is a representation of that object in 4D space together with a projection from 4D space to 3D space [8]. Similarly, an object in 3D space may be elevated to 4D space using a projection from 3D space to 4D space. A rational curve in n dimensional space can be represented as a polynomial curve in $(n+1)$ dimensional space using homogeneous coordinates.

A point $\mathbf{P} = [x\ y\ z]^T$ in 3D Euclidean space, $\mathbb{E}^3$, may be elevated to 4D space by choosing a weight w and denoted by the homogeneous point

$$\mathbf{P}^w = \begin{cases} [wx\ wy\ wz\ w]^T & \text{if } w \neq 0 \\ [\text{x y z } 0]^T & \text{if } w = 0 \end{cases}$$

The point $\mathbf{P}$ is retrieved from $\mathbf{P}^w$ using a projective map H defined by

$$H\{[X\ Y\ Z\ W]^T\} = \begin{cases} [X/W\ Y/W\ Z/W]^T & \text{if } W \neq 0 \\ \text{direction}([X\ Y\ Z]^T) & \text{if } W = 0. \end{cases} \tag{2.14}$$

H maps $\mathbf{P}^w$ from the origin to the hyperplane $W = 1$. Hence,

$$\mathbf{P} = H\{\mathbf{P}^w\}$$

A particular advantage of using homogeneous coordinates $\mathbf{P}^w$ is that they are closed under *homogeneous* transformations, such as perspective transformations. Since both affine transformations (section 2.4.3) and perspective transformations may be expressed as homogeneous transformations in a 4×4 matrix, a series of affine and perspective transformations can be combined to give a single 4×4 matrix. This single matrix can then be applied to a homogeneous coordinate.

Homogeneous coordinates are now used to define NURBS.

2.5.2 Definition of NURBS Curves and Surfaces

A NURBS curve $\mathbf{C}(u)$, is the 3D projection of a 4D B-spline curve $\mathbf{C}^w(u)$, where

$$\mathbf{C}^w(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i^w \tag{2.15}$$

and

$$\mathbf{C}(u) = H\{\mathbf{C}^w(u)\} = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}. \tag{2.16}$$

$\mathbf{P}_i$ are the control points forming the control polygon; $\mathbf{P}_i^w = [w_i\mathbf{P}_i \ w_i]^T$ are the 4D homogeneous control points; w_i are the weights used to elevate $\mathbf{P}_i$ to $\mathbf{P}_i^w$; H is the perspective map given in (2.14). Other terms are given in (2.8). Rational B-spline basis functions are given by

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j} \tag{2.17}$$

and so (2.16) can be written as

$$\mathbf{C}(u) = \sum_{i=0}^{n} R_{i,p}(u)\mathbf{P}_i. \tag{2.18}$$

Figure 2.4 shows a cubic NURBS curve defined on the knot vector $\{0, 0, 0, 0, 0.2, 0.4, 0.7, 0.9, 1, 1, 1, 1\}$. A NURBS surface $\mathbf{S}(u, v)$, is the 3D projection of a 4D B-spline surface $\mathbf{S}^w(u, v)$, where

$$\mathbf{S}^w(u, v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u) N_{j,p}(v)\mathbf{P}_{i,j}^w \tag{2.19}$$

and

$$\mathbf{S}(u, v) = H\{\mathbf{S}^w(u, v)\} = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j}}. \tag{2.20}$$

$\mathbf{P}_{i,j}$ are the control points arranged in a topological net; $\mathbf{P}_{i,j}^w = [w_{i,j}\mathbf{P}_{i,j} \ w_{i,j}]^T$ are the 4D homogeneous control points; w_i are the weights used to elevate $\mathbf{P}_i$ to

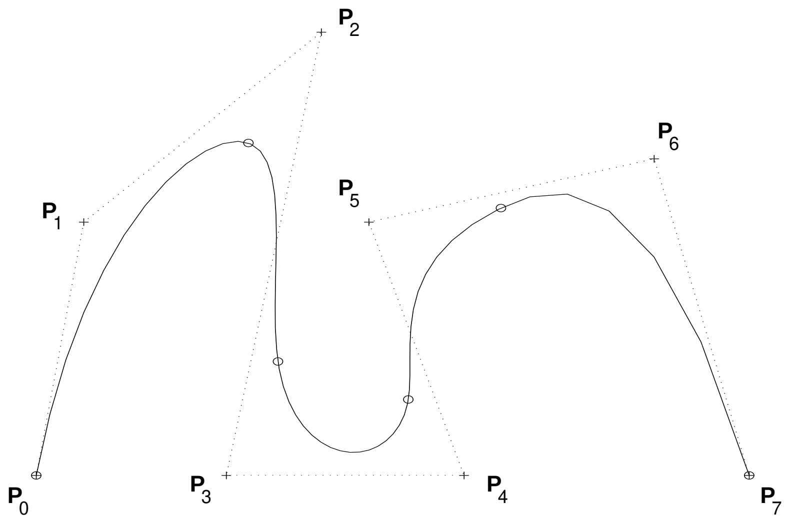

Figure 2.4: A cubic NURBS curve with its control polygon, control points (+) and points on the curve corresponding to knot values (o). The curve is defined on the knot vector $\{0, 0, 0, 0, 0.2, 0.4, 0.7, 0.9, 1, 1, 1, 1\}$.

$\mathbf{P}_i^w$. Other terms are given in (2.11). Bivariate rational B-spline basis functions are given by

$$R_{(i,p)(j,q)}(u,v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{l=0}^{n}\sum_{k=0}^{m} N_{l,p}(u)N_{k,q}(v)w_{l,k}} \tag{2.21}$$

and so (2.20) can be written as

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{(i,p),(j,q)}(u,v)\mathbf{P}_{i,j}. \tag{2.22}$$

Note that the NURBS surface in (2.22) is not a tensor product surface because the rational B-spline basis functions given in (2.21) are not simply the product of univariate basis functions (section 2.1). However, the 4D B-spline surface in (2.19) *is* a tensor product surface. Figure 2.5 shows a biquadratic NURBS surface defined on knot vectors $U = \{0, 0, 0, 0.25, 0.75, 1, 1, 1\}$ and $V = \{0, 0, 0, 0.25, 0.75, 1, 1, 1\}$.

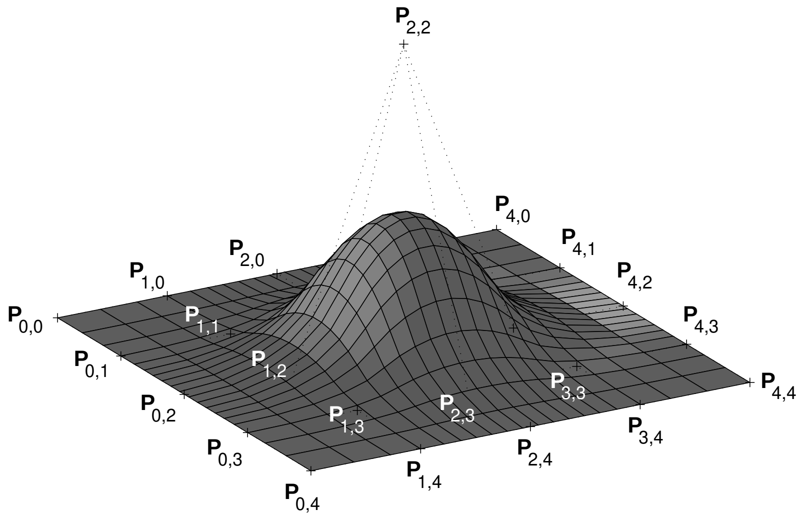

Figure 2.5: A biquadratic NURBS surface with its control polygon and control points (+). The surface is defined on knot vectors $U = \{0, 0, 0, 0.25, 0.75, 1, 1, 1\}$ and $V = \{0, 0, 0, 0.25, 0.75, 1, 1, 1\}$.

2.5.3 Properties of Rational B-Spline Basis Functions

Rational B-spline basis functions $R_{i,p}(u)$, given in (2.17), have the same analytic properties as B-spline basis functions, $N_{i,p}(u)$, described in section 2.4.2. However, non-negativity only holds for $w_i > 0$ for all i. Rational B-spline basis functions have the additional property that if all the weights in (2.16) and (2.20) are the same value (typically 1), then

$$R_{i,p}(u) = \begin{cases} B_{i,p}(u) & \text{if } T = \{\underbrace{0, \ldots, 0}_{p+1}, \underbrace{1, \ldots, 1}_{p+1}\} \\ N_{i,p}(u) & \text{otherwise} \end{cases}$$

where $B_{i,p}(u)$ are the Bernstein basis function given in (2.7). In fact, Bézier, rational Bézier and B-spline curves and surfaces are all special cases of NURBS curves and surfaces.

2.5.4 Properties of NURBS Curves and Surfaces

NURBS inherit all the geometric properties described in section 2.4.3 if all weights are positive. Using homogeneous coordinates, NURBS have the following additional properties [39]:

- Local shape control using weights as well as control points: NURBS exhibit the same local control as B-splines. However, modifying the weights of a control point also changes the shape of a NURBS locally, with the following properties:
  - Modifying w_i affects the curve $\mathbf{C}(u)$ only in the interval $u \in [u_i, u_{i+p+1})$.
  - If w_i increases (decreases), the point $\mathbf{C}(u)$ moves toward (away) from $\mathbf{P}_i$ and away from (toward) $\mathbf{P}_j$ $(j \neq i)$.
  - If $w_i \to \infty$, then
$$\mathbf{C}(u) = \begin{cases} \mathbf{P}_i & \text{if } u \in [u_i, u_{i+p+1}) \\ \mathbf{C}(u) & \text{otherwise.} \end{cases}$$
  - If $u \in [u_i, u_{i+p+1})$ is fixed and w_i varies between $0 < w_i < \infty$, then $\mathbf{C}(u)$ moves along a straight line segment between $\mathbf{C}(u)$ at $w_i = 0$ and the control point $\mathbf{P}_i$.

  Local shape control using weights is an important property in interactive design environments as it gives the designer another method of shape control;

- Exact representation of conics and quadrics: the use of rational basis functions allows NURBS curves to represent conic sections exactly. Since conic sections are among the most important curves in CAD [45], the ability to represent such shapes is an important property. NURBS also provide an exact representation of quadrics, both natural and general forms. The natural quadrics are the plane, cylinder, cone, and sphere. Hence NURBS provide a unified representaion of many common surfaces used in CAD applications. For a rigorous treatment of conics and quadrics see [48][45][39].

- Closed under homogeneous transformations: NURBS curves and surfaces are closed under homogeneous transformations. As stated in section 2.5.1, homogeneous transformations include affine and perspective transformations. This is an important property in computer graphics, which makes extensive use of affine and perspective transformations.

2.5.5 Derivatives of NURBS Curves and Surfaces

Since $\mathbf{C}^w(u)$ given in (2.15), and $\mathbf{S}^w(u,v)$ given in (2.19), are simply B-spline curves and surfaces, respectively, in 4D space, the kth derivative of a curve (2.12) and the $(k+l)$th partial derivatives of a surface (2.13) can be written in terms of homogeneous control points, i.e., for curves:

$$\mathbf{C}^{w(k)}(u) = \sum_{i=0}^{n-k} N_{i,p-k}(u)\mathbf{P}_i^{w(k)}, \tag{2.23}$$

and for surfaces

$$\frac{\partial^{k+l}}{\partial^k u \partial^l v}\mathbf{S}^w(u,v) = \sum_{i=0}^{n-k}\sum_{j=0}^{m-l} N_{i,p-k}(u)N_{j,q-l}(v)\mathbf{P}_{i,j}^{w\,(k,l)}. \tag{2.24}$$

Other terms are given in (2.12) and (2.13). However, the derivatives of NURBS, rather than the 4D B-spline representation of NURBS can also be defined. In [48], Piegl and Tiller give the equations for general order derivatives of NURBS curves and surfaces. This thesis requires only the first order derivatives:

Denote the numerator in (2.16) as $\mathbf{A}(u)$ and the denominator as $w(u)$. Then the first derivative of a NURBS curve is given by

$$\mathbf{C}' = \frac{w\mathbf{A}' - w'\mathbf{A}}{w^2} \tag{2.25}$$

Similarly, for surfaces denote the numerator in (2.20) as $\mathbf{A}(u, v)$ and the denominator as $w(u, v)$. Then

$$\mathbf{S}_u = \frac{w\mathbf{A}_u - w_u\mathbf{A}}{w^2} \tag{2.26}$$

$$\mathbf{S}_v = \frac{w\mathbf{A}_v - w_v\mathbf{A}}{w^2} \tag{2.27}$$

For simplicity, the (u) and (u, v) paramters have been omitted from (2.25), (2.26) and (2.27).

2.5.6 Knot Insertion

Knot insertion is one of several geometric algorithms used to modify NURBS curves and surfaces. It has several applications, including the addition of control points during interactive shape design, obtaining polygonal approximations to curves and surfaces, and the splitting of curves and surfaces. A rigorous study of knot insertion can be found in [48] and a good summary of the problem and its applications in [39]. This thesis uses knot insertion to subdivide a surface into four sub-surfaces. Essentially, new knots, corresponding to the parametric values at which the surface is to be subdivided, are inserted into the U and V knot vectors. New control points are found which define the sub-surfaces.

Given a NURBS surface of degree $p \times q$, as in (2.19), and interior parameter values $\overline{u}$ and $\overline{v}$, inserting $\overline{u}$ with multiplicity p and $\overline{v}$ with multiplicity q subdivides the surface into four NURBS sub-surfaces. These sub-surfaces may be subdivided further by inserting new knots into each surface, and so on. This form of surface subdivision will be used in section 5.5.

2.6 Summary

This chapter has presented a brief introduction to parametric shape representation, focusing on NURBS. NURBS have several properties which make them ideal as a modelling primitive. Their ability to represent exactly conic primitives, as well as free-form curves and surfaces, is of great benefit in a geometric modelling environment since it allows a unified representation of many primitives. Two methods of locally modifying the shape of a NURBS curve or surface make them particularly effective in an interactive design tool. Since NURBS are generalisations of B-splines and Bézier forms, algorithms developed for these forms can easily be extended to NURBS. NURBS are also closed under general homogeneous transformations which is of great use in computer graphics.

The next chapter will introduce radiosity, describing the general principles and fundamental algorithms.

Chapter 3

Fundamentals of Radiosity

This chapter introduces the radiosity method, giving nomenclature and associated units, and describes solution methods for polygonal environments. Radiosity is based on physical processes, originally thermal radiative transfer [40][18][36] and later adapted to illumination for use in computer graphics [26][41].

3.1 Terminology and Physical Quantities

To describe the radiosity method accurately, the following notation and physical quantities are required:

3.1.1 Local Coordinate Systems

When describing the location of an object or the direction, $\vec{\omega}$, of radiative transfer, relative to a point $\mathbf{x}$, a local coordinate system is defined at the point $\mathbf{x}$. A

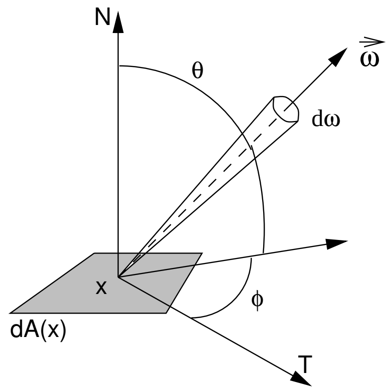

Figure 3.1: A spherical coordinate system centred at a differential element $\mathbf{x}$.

direction, $\vec{\omega}$, may then be specified by two angles (θ, ϕ). θ is the angle between the direction and the surface normal at $\mathbf{x}$. ϕ is the angle between the direction and an arbitrary surface tangent at $\mathbf{x}$. Figure 3.1 shows the spherical coordinate system at a differential element $\mathbf{x}$, of area $dA(\mathbf{x})$.

3.1.2 Solid Angle

The solid angle, ω, is the three-dimensional (3D) extension of the two-dimensional (2D) angle between two lines. Whereas the 2D angle measures the arc length of an object projected onto the radius of a unit circle, the solid angle measures the area of an object projected onto the surface of a unit sphere (figure 3.2). More precisely, the solid angle subtended by an object from a point $\mathbf{x}$ is given by the surface area of the radial projection of the object onto a unit sphere centred at point $\mathbf{x}$. Solid angles are measured in *steradians* (sr). Consider a differential element of area dA located a distance r from a point $\mathbf{x}$. Consider also a direction $\vec{\omega}$ from $\mathbf{x}$ passing through the centre of the differential element. The differential

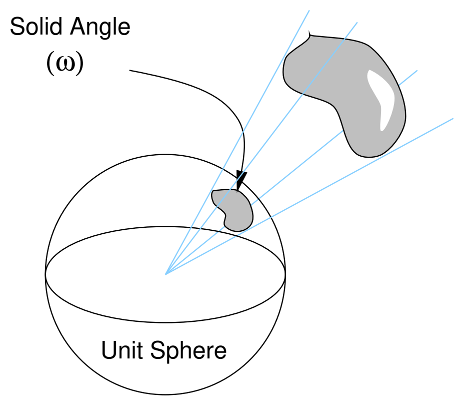

Figure 3.2: The solid angle subtended by an object.

solid angle subtended by such an element may be approximated by

$$d\omega \approx \frac{dA\ \cos\theta}{r^2}. \tag{3.1}$$

3.1.3 Energy and Power

When modelling the distribution of light in an environment, we are modelling how a form of electromagnetic radiation is distributed, since light is merely one form of electromagnetic radiation. The *radiant power*, of, in this case light, is given by radiant flux $\Phi = dQ/dt$, which is a measure of how *radiant energy* Q varies with time. Radiant power is measured in Joules per second (Js^{-1}) or, equivalently, Watts (W).

Radiant power is a useful quantity as we would certainly like to model the power[1] of light within an environment. Radiant power, however, tells us nothing about how light varies with surface area or direction of transport (or viewing position).

[1]The term power will be taken to mean radiant power.

Both of these factors are needed in a comprehensive model of the distribution of light in an environment. The following three quantities describe how radiant power varies with such quantities.

3.1.4 Radiance

Radiance, L, is defined as radiant power travelling from a point, per unit projected area perpendicular to the direction of transport, per unit solid angle, and is given by

$$L(\mathbf{x}, \vec{\omega}) = \frac{1}{\cos\theta} \frac{d^2\Phi}{dA\, d\omega},$$

where $\vec{\omega}$ is the direction of transport from a point $\mathbf{x}$. Radiance is measured in Watts per metre squared per steradian ($Wm^{-2}sr^{-1}$) and describes how radiant power varies with viewing angle and surface area.

An important property of radiance is that the radiance in a particular direction of transport remains constant as it propagates along the direction of transport[2]. This property ensures that the radiance *incident* at a point $\mathbf{x}$ from a direction $\vec{\omega}_{in}$, is equal to the radiance *leaving* a point $\mathbf{x}'$ in a direction $\vec{\omega}_{in}$ to the point $\mathbf{x}$, i.e.

$$L_{in}(\mathbf{x}, \vec{\omega}_{in}) = L_{out}(\mathbf{x}', \vec{\omega}_{in}). \tag{3.2}$$

[2]Provided there is no loss due to absorption or scattering. This thesis assumes an environment is defined in a vacuum as so no absorption or scattering takes place.

3.1.5 Irradiance

Irradiance, E, describes how radiant power varies as a function of surface area only. More precisely, irradiance is defined as the the total radiant power *incident* on a surface, per unit surface area, and is defined in terms of incident radiant power, Φ_{in}:

$$E = \frac{d\Phi_{in}}{dA}.$$

Irradiance is measured in Watts per metre squared (Wm^{-2}) and can be related to incident radiance, L_{in}, by integrating over the unit hemisphere, Ω:

$$E(\mathbf{x}) = \int_{\Omega} L_{in}(\mathbf{x}, \vec{\omega}) \cos\theta \, d\omega.$$

3.1.6 Radiosity

Radiosity, B, is similar to irradiance. Whereas irradiance measure the total radiant power *incident* on a surface, radiosity measures the total radiant power *leaving* a surface, per unit surface area. Thus, radiosity[3] is defined in terms of outgoing radiant power, Φ_{out}:

$$B = \frac{d\Phi_{out}}{dA}.$$

Radiosity is also measured in Watts per metre squared (Wm^{-2}) and can be related to outgoing radiance, L_{out}, by integrating over the unit hemisphere, Ω:

$$B(\mathbf{x}) = \int_{\Omega} L_{out}(\mathbf{x}, \vec{\omega}) \cos\theta \, d\omega. \tag{3.3}$$

We now have three measures to characterise the distribution of light in an environment. Irradiance describes how power varies with area and direction of

[3]The correct term for radiosity is radiant exitance. However, radiosity has been adopted by the computer graphics community as so will be used in this thesis.

transport, i.e., it gives the radiant power at a point, per projected unit area, per unit solid angle. The dependence on direction can be removed to give irradiance, for incident power, and radiosity, for outgoing power. Irradiance (radiosity) give the total incident (outgoing) power per unit area at a point.

A mechanism is now required that allows the reflective properties of a surface to be described. Reflection is the process by which light incident on a surface leaves the surface from the same side. In this discussion surfaces are assumed to be opaque and so transmission may be ignored. Note also that, so far, all definitions have assumed light is monochromatic, i.e., of a single wavelength. However, all radiometric quantities may be sampled at discrete wavelengths when colour images are required[4]. For notational convenience, any dependence on wavelength will be omitted and equations are presented as if for monochromatic light only. Describing the reflective properties of a surface is achieved using a Bidirectional Reflectance Distribution Function (BRDF).

3.1.7 The BRDF

The bidirectional reflectance distribution function gives a measure of the directional distribution of reflected light from a surface. The BRDF, denoted by ρ_{bd}, is defined to be the ratio of reflected, or outgoing, radiance at a point $\mathbf{x}$, in a direction $\vec{\omega}_{out}$, to the irradiance at $\mathbf{x}$, from the incident direction $\vec{\omega}_{in}$, i.e.

$$\rho_{bd}(\mathbf{x}, \vec{\omega}_{out}, \vec{\omega}_{in}) = \frac{dL_{out}(\mathbf{x}, \vec{\omega}_{out})}{L_{in}(\mathbf{x}, \vec{\omega}_{in}) \cos\theta_{in}\, d\omega_{in}} \tag{3.4}$$

[4]This discrete sampling precludes the modelling of fluorescence, in which light undergoes a change of wavelength before being emitted, but this is not under consideration in this thesis.

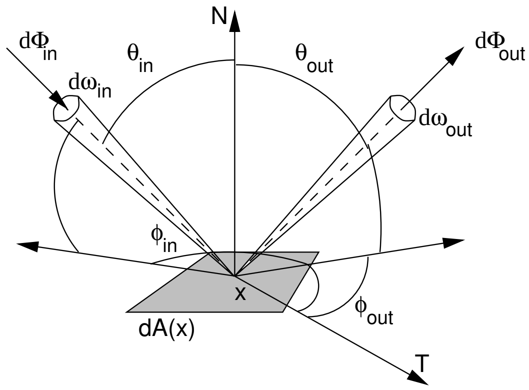

Figure 3.3: Terms used in the definition of the BRDF.

Figure 3.3 shows terms used in the definition of the BRDF. The BRDF is measured in *inverse steradians* (sr^{-1}) and can take any value from 0 to infinity. A more useful measure of reflectance is the *directional hemispherical reflectance* function ρ_{dh}. This is defined to be the fraction of incident power per unit area that is reflected by a surface in all directions, i.e., the amount of light scattered into the entire hemisphere from a single incident direction, and is given by

$$\rho_{dh}(\mathbf{x}, \vec{\omega}_{in}) = \int_{\Omega} \rho_{bd}(\mathbf{x}, \vec{\omega}_{out}, \vec{\omega}_{in}) \cos\theta_{out}\, d\omega_{out}, \tag{3.5}$$

where Ω is the unit hemisphere. The directional hemispherical reflectance is dimensionless and takes values from 0 to 1 and is commonly referred to as simply the *reflectance* of a surface [14].

Note that the BRDF satisfies the *Helmholtz reciprocity rule*, which states that interchanging the incident and outgoing directions will not affect the fraction of light being reflected, i.e.,

$$\rho_{bd}(\mathbf{x}, \vec{\omega}_{out}, \vec{\omega}_{in}) = \rho_{bd}(\mathbf{x}, \vec{\omega}_{in}, \vec{\omega}_{out}). \tag{3.6}$$

The BRDF allows very general reflectance distributions to be described. As noted by Gatenby [22], many formulations have been given for the BRDF, some of which as discussed in [21].

3.2 The Rendering Equation

Using the previously defined physical quantities and the BRDF of a surface, the distribution of light in an environment, or scene S, can now be determined. This results in the *rendering equation* or *global illumination equation*. The rendering equation was first presented in the context of computer graphics by Kajiya [37], using slightly different notation to that given here. This equation specifies the total, unknown, radiance, L_{out} leaving a surface point $\mathbf{x}$ in a direction $\vec{\omega}_{out}$ in terms of the radiance emitted by the surface point, L_e, plus the radiance reflected by the surface point L_r, due to incident radiance from other surfaces, i.e.,

$$L_{out}(\mathbf{x}, \vec{\omega}_{out}) = L_e(\mathbf{x}, \vec{\omega}_{out}) + L_r(\mathbf{x}, \vec{\omega}_{out}). \tag{3.7}$$

For closed environments, (3.7) shows that energy is conserved in the environment—energy is absorbed by a surface or reflected or both. The total energy in the environment remains constant. Recall (3.4), which gives the BRDF in terms of incident radiance and outgoing irradiance. Rearranging (3.4) and integrating over the hemisphere of incoming directions, Ω_{in}, an expression for the reflected radiance due to incoming light from *all* directions is given by the *reflectance equation*:

$$L_r(\mathbf{x}, \vec{\omega}_{out}) = \int_{\Omega_{in}} \rho_{bd}(\mathbf{x}, \vec{\omega}_{out}, \vec{\omega}_{in}) L_{in}(\mathbf{x}, \vec{\omega}_{in}) \cos\theta_{in}\, d\omega_{in}.$$

(3.7) can now be written

$$L_{out}(\mathbf{x}, \vec{\omega}_{out}) \;=\; L_e(\mathbf{x}, \vec{\omega}_{out})$$

$$+ \int_{\Omega_{in}} \rho_{bd}(\mathbf{x}, \vec{\omega}_{out}, \vec{\omega}_{in}) L_{in}(\mathbf{x}, \vec{\omega}_{in}) \cos\theta_{in}\, d\omega_{in}. \qquad (3.8)$$

A further change can simplify (3.8) by removing the dependence on incident radiance. Rather than considering incident radiance at a point $\mathbf{x}$, (3.8) can be written in terms of outgoing radiance from a point $\mathbf{x}'$, using (3.2). While at first this change seems introduce another variable, $\mathbf{x}'$, it allows (3.8) to be written as an integral equation of only one unknown function, L_{out}.

The differential solid angle, $d\omega_{in}$ may also be written in terms of $\mathbf{x}'$ using the approximation given in (3.1)

$$d\omega_{in} = \frac{\cos\theta'\, dA(\mathbf{x}')}{\|\mathbf{x} - \mathbf{x}'\|^2}.$$

By introducing the point $\mathbf{x}'$, radiance at $\mathbf{x}$ is found by considering radiance *leaving* all points $\mathbf{x}'$ visible to $\mathbf{x}$, travelling toward $\mathbf{x}$. Thus, the integration limits in (3.8) must now select all $\mathbf{x}'$ visible to $\mathbf{x}$, rather than all directions in Ω_{in}. Determining all $\mathbf{x}'$ visible to $\mathbf{x}$ is a non-trivial task and so a simpler range of integration is used. The integrand is evaluated for *all* $\mathbf{x}'$ in the environment S, and a visibility function $V(\mathbf{x}, \mathbf{x}')$ is introduced. $V(\mathbf{x}, \mathbf{x}')$ is defined by

$$V(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & \mathbf{x} \text{ and } \mathbf{x}' \text{ are mutually visible} \\ 0 & \text{otherwise.} \end{cases}$$

Hence, (3.8) can now be written

$$L_{out}(\mathbf{x}, \vec{\omega}_{out}) = L_e(\mathbf{x}, \vec{\omega}_{out}) +$$
$$\int_{\mathbf{x}' \in S} \rho_{bd}(\mathbf{x}, \vec{\omega}_{out}, \vec{\omega}_{in}) L_{out}(\mathbf{x}', \vec{\omega}_{in}) V(\mathbf{x}, \mathbf{x}') \frac{\cos\theta_{in} \cos\theta'}{\|\mathbf{x} - \mathbf{x}'\|^2}\, dA(\mathbf{x}'). \qquad (3.9)$$

The radiance leaving any point in the scene, in any direction, is now expressed in terms of the radiance leaving every other point in the scene.

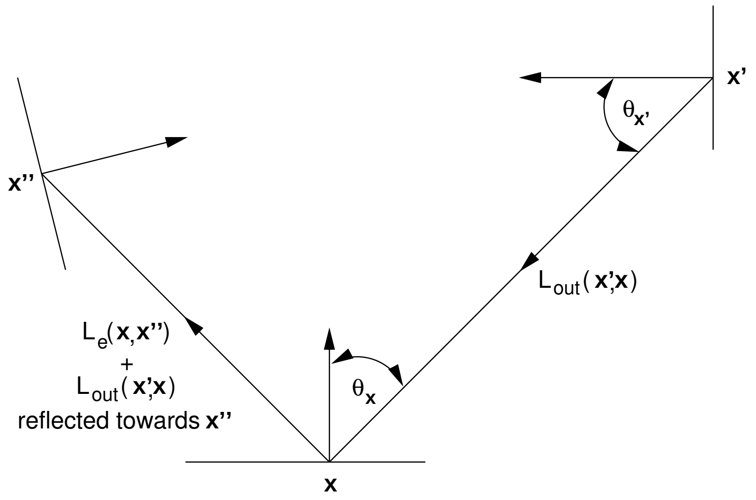

Figure 3.4: The relationship between three points $\mathbf{x}$, $\mathbf{x}'$ and $\mathbf{x}''$ in the environment. The radiance leaving $\mathbf{x}$, towards $\mathbf{x}''$, is considered in terms of the radiance emitted by $\mathbf{x}$ and radiance arriving at $\mathbf{x}$ from $\mathbf{x}'$.

Note that (3.9) determines radiance leaving a point $\mathbf{x}$ in some *direction*, $\vec{\omega}_{out}$. In practice, it is more convenient to specify the outgoing direction in terms of a third point $\mathbf{x}''$, i.e., radiance leaving $\mathbf{x}$ toward $\mathbf{x}''$. The direction is then given by $\mathbf{x}'' - \mathbf{x}$. To avoid this cumbersome notation, L and ρ_{bd} will simply be written in terms of two and three points respectively. Modifying the notation for the angle θ in a similar fashion (figure 3.4), the rendering equation becomes

$$L_{out}(\mathbf{x},\mathbf{x}'') = L_e(\mathbf{x},\mathbf{x}'') + \int_{\mathbf{x}'\in S} \rho(\mathbf{x},\mathbf{x}'',\mathbf{x}')L_{out}(\mathbf{x}',\mathbf{x})V(\mathbf{x},\mathbf{x}')\frac{\cos\theta_{\mathbf{x}}\cos\theta_{\mathbf{x}'}}{\|\mathbf{x}-\mathbf{x}'\|^2}\,dA(\mathbf{x}') \tag{3.10}$$

(3.10) is referred to as the *rendering equation.* The rendering equation is an integral equation[5] that, in general, cannot be solved analytically, except for a few specific cases. However, by assuming that surfaces in an environment are ideal diffuse reflectors, (3.10) can by simplified. This will lead to a formulation of the rendering equation often referred to as the *Radiosity Equation.*

[5] See section 4.3.

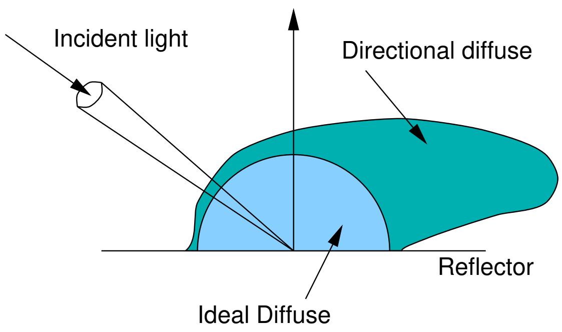

Figure 3.5: An ideal diffuse reflector reflects light equally in all directions, rather than in some direction dependent on the direction of incoming light.

3.2.1 The Radiosity Equation

The radiosity equation [26] is derived from the rendering equation by assuming that all surfaces in the environment are ideal diffuse reflectors. An ideal diffuse reflector exhibits Lambertian diffuse reflectance. For Lambertian diffuse reflection, light is reflected equally in all directions [25] (figure 3.5). The BRDF, (3.4), then, has no dependence on outgoing direction, $\vec{\omega}_{out}$. Also, by the reciprocity rule of the BRDF, given in (3.6), the BRDF must have no dependence on the incoming direction, $\vec{\omega}_{in}$. Hence the BRDF is given by

$$\rho_{bd}(\mathbf{x}, \vec{\omega}_{out}, \vec{\omega}_{in}) = \rho(\mathbf{x}).$$

Recall the directional hemispherical reflectance function, ρ_{dh}, (3.5). Since the BRDF now has no dependence on direction, it can be moved out of the integral to give

$$\begin{aligned} \rho_{dh}(\mathbf{x}, \vec{\omega}_{in}) &= \rho(\mathbf{x}) \int_{\Omega} \cos\theta_{out} \, d\omega_{out} \\ &= \pi \rho(\mathbf{x}). \end{aligned}$$

The rendering equation, (3.10) may now be written in terms of the directional hemispherical reflectance function. Any dependence on incident and outgoing direction is also removed from all terms in the rendering equation, when ideal diffuse reflectors are considered, to give

$$L_{out}(\mathbf{x}) = L_e(\mathbf{x}) + \frac{\rho(\mathbf{x})}{\pi} \int_{\mathbf{x}' \in S} L_{out}(\mathbf{x}') V(\mathbf{x}, \mathbf{x}') \frac{\cos\theta_{\mathbf{x}} \cos\theta_{\mathbf{x}'}}{\|\mathbf{x} - \mathbf{x}'\|^2} \, dA(\mathbf{x}'). \qquad (3.11)$$

Recall that radiance describes power as a function of surface area and direction of transport (section 3.1.4) and that radiosity describes power as a function of surface area only (section 3.1.6). Now that directional dependence has been removed, radiosity may be used in place of radiance. Equation (3.3) gives radiosity in terms of radiance, i.e.

$$B(\mathbf{x}) = \int_{\Omega} L(\mathbf{x}, \vec{\omega}) \cos\theta \, d\omega.$$

Since L no longer depends on $\vec{\omega}$, L may be moved out of the integral to give

$$\begin{aligned} B(\mathbf{x}) &= L(\mathbf{x}) \int_{\Omega} \cos\theta \, d\omega \\ &= \pi L(\mathbf{x}). \end{aligned}$$

Performing a similar integration for all terms in (3.11) (and multiplying both sides of the equation by π) gives

$$B_{out}(\mathbf{x}) = B_e(\mathbf{x}) + \rho(\mathbf{x}) \int_{\mathbf{x}' \in S} B_{out}(\mathbf{x}') V(\mathbf{x}, \mathbf{x}') \frac{\cos\theta_{\mathbf{x}} \cos\theta_{\mathbf{x}'}}{\pi \|\mathbf{x} - \mathbf{x}'\|^2} \, dA(\mathbf{x}'). \qquad (3.12)$$

(3.12) represents the *radiosity equation* [14], for which numerous solution methods exist. The next sections discusses several *classical radiosity methods* for simplifying (3.12) further.

3.3 Classical Radiosity Methods

Despite the various simplifications presented in the previous sections, (3.12) is still an integral equation that cannot, in general, be solved analytically. There are many numerical methods available for solving such equations [4][17][35]. The approach taken in [26], and in much of the work that followed, is to divide each surface in the environment into a number of smaller surfaces, or *patches*. The integral over all points in the environment, in (3.12), is now broken into N integrals, each corresponding to a discrete patch P_j:

$$B_{out}(\mathbf{x}) = B_e(\mathbf{x}) + \rho(\mathbf{x}) \sum_{j=1}^{N} \int_{\mathbf{x}' \in P_j} B_{out}(\mathbf{x}') V(\mathbf{x}, \mathbf{x}') \frac{\cos\theta_{\mathbf{x}} \cos\theta_{\mathbf{x}'}}{\pi \|\mathbf{x} - \mathbf{x}'\|^2} \, dA(\mathbf{x}'). \quad (3.13)$$

The radiosity across each patch is then considered to be constant. This is equivalent to approximating the radiosity function with a constant basis function over each patch (see section 4.3 for more details). Figure 3.6(i) shows a continuous radiosity function and figure 3.6(ii) shows the approximation of this function using constant radiosity defined over each patch. The reflectance function, ρ, is also assumed to be constant over each patch. The constant radiosity, B_i, over the ith patch, P_i, is given by an area-weighted average of the pointwise radiosities over the patch using [56]:

$$B_i = \frac{1}{A_i} \int_{\mathbf{x} \in P_i} B(\mathbf{x}) \, d\mathbf{x},$$

where A_i is the area of ith patch. Applying a similar integration to each radiosity term in (3.13), the *classical radiosity equation* may be written:

$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j \frac{1}{A_i} \int_{\mathbf{x} \in P_i} \int_{\mathbf{x}' \in P_j} V(\mathbf{x}, \mathbf{x}') \frac{\cos\theta_{\mathbf{x}} \cos\theta_{\mathbf{x}'}}{\pi \|\mathbf{x} - \mathbf{x}'\|^2} \, dA(\mathbf{x}') \, dA(\mathbf{x}),$$

Making a change of notation, E_i is the radiosity being emitted by the ith patch (for example, due to incandescent or blackbody processes [25]); ρ_i is the constant reflectance of the ith patch; B_j is the constant radiosity over the jth patch P_j and N is the number of patches in the environment.

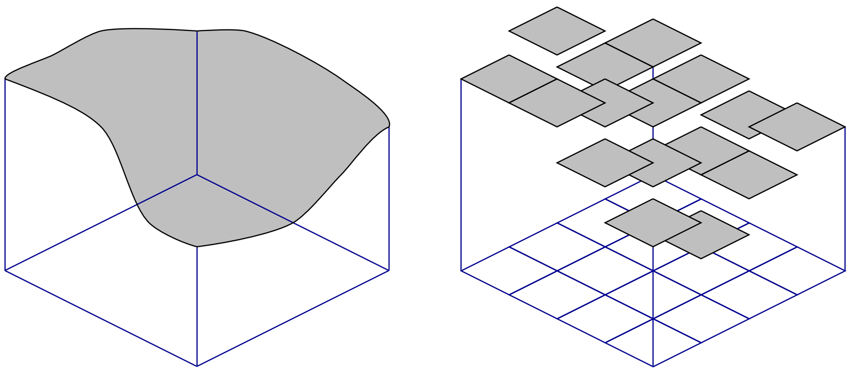

Figure 3.6: Approximating the radiosity function with constant patch radiosities.

Introducing the notation

$$F_{ij} = \frac{1}{A_i} \int_{\mathbf{x} \in P_i} \int_{\mathbf{x}' \in P_j} V(\mathbf{x}, \mathbf{x}') \frac{\cos \theta_{\mathbf{x}} \cos \theta_{\mathbf{x}'}}{\pi \|\mathbf{x} - \mathbf{x}'\|^2} \, dA(\mathbf{x}') \, dA(\mathbf{x}), \tag{3.14}$$

to define the *form factor* between two patches P_i and P_j, the radiosity equation is given by

$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j F_{ij}. \tag{3.15}$$

The form factor given in (3.14) is a purely geometric term giving the proportion of power leaving patch P_i that is received by patch P_j [26]. Once F_{ij} are known for all patches i and j in the environment, (3.15) can be solved. Computing form factors, though, is often the most time consuming part of a radiosity solution [56].

3.4 Form Factors

The form factor depends only on the shape and relative location of surfaces in a scene. This fact is particularly useful when the radiosity equation, given in

(3.15), is sampled at various wavelengths[6]. Since the form factor is independent of wavelength, form factors only need to be determined once and can then be reused for each spectral sample.

3.4.1 Point-to-Point Form Factor

Recall (3.14), which gave the form factor between two patches, P_i and P_j. As such, (3.14) is referred to as the *patch-to-patch* form factor. The integrand of (3.14) gives the form factor between two differential areas. Thus, the *point-to-point* form factor between two differential areas $dA(\mathbf{x})$ and $dA(\mathbf{x}')$ is given by

$$F_{dA(\mathbf{x})dA(\mathbf{x}')} = \frac{\cos\theta_{\mathbf{x}}\cos\theta_{\mathbf{x}'}}{\pi\|\mathbf{x}-\mathbf{x}'\|^2}dA(\mathbf{x}'). \tag{3.16}$$

Note that it is assumed that the two differential areas are mutually visible, i.e., $V(\mathbf{x},\mathbf{x}') = 1$.

3.4.2 Point-to-Patch Form Factor

Enlarging $\mathbf{x}'$ so that it forms a finite element, or patch, P_j, the point-to-point form factor can be integrated over all $\mathbf{x}'$ on patch P_j. The mutual visibility of two differential areas must now be considered. If a point $\mathbf{x}'$ on P_j is not visible to $\mathbf{x}$, $\mathbf{x}'$ does not contribute to the form factor between $\mathbf{x}$ and P_j. Thus, the *point-to-patch* form factor between differential element $\mathbf{x}$ and finite element P_j is given by

$$F_{dA(\mathbf{x})P_j} = \int_{\mathbf{x}'\in P_j} V(\mathbf{x},\mathbf{x}')\frac{\cos\theta_{\mathbf{x}}\cos\theta_{\mathbf{x}'}}{\pi\|\mathbf{x}-\mathbf{x}'\|^2}dA(\mathbf{x}'). \tag{3.17}$$

[6] For example, red, green and blue samples are usually taken for the production of colour images.

3.4.3 Patch-to-Patch Form Factor

Enlarging $\mathbf{x}$ to form the finite element P_i and integrating over every point on P_i results in the *patch-to-patch* form factor given in equation (3.14). Writing the integration step explicitly, the patch-to-patch form factor can be written concisely

$$F_{ij} = \frac{1}{A_i} \int_{\mathbf{x} \in P_i} F_{dA(\mathbf{x})P_j} \, dA(\mathbf{x}). \tag{3.18}$$

3.4.4 Reciprocity Rules

The expressions giving form factors ((3.16), (3.17), (3.18)) are symmetric except for an area factor. The area factor depends on the type of form factor being defined. Multiplying by the area factor results in the following reciprocity rules

$$\begin{aligned} dA(\mathbf{x}) F_{dA(\mathbf{x}) dA(\mathbf{x}')} &= dA(\mathbf{x}') F_{dA(\mathbf{x}') dA(\mathbf{x})} \\ dA(\mathbf{x}) F_{dA(\mathbf{x}) P_j} &= A_j F_{P_j dA(\mathbf{x})} \\ A_i F_{ij} &= A_j F_{ji} \end{aligned} \tag{3.19}$$

Reciprocity rules allow unknown form factors to be determined from known form factors. (3.19) also explains the seemingly erroneous F_{ij} term in (3.15). Multiplying (3.15) by A_i gives

$$A_i B_i = A_i E_i + \rho_i \sum_{j=1}^{N} B_j A_i F_{ij}.$$

Now, using the reciprocity rule in (3.19), we obtain

$$A_i B_i = A_i E_i + \rho_i \sum_{j=1}^{N} B_j A_j F_{ji}, \tag{3.20}$$

which contains the expected form factor F_{ji}. Note that (3.20) is defined in terms of *power*, not *radiosity*.

3.4.5 Computing the Form Factor

In general, closed form expressions for form factors exist for only simple geometries [54]. As such, numerical methods must be used to approximate the form factor. A brief discussion of such methods is presented here.

Recall (3.18) which gave the patch-to-patch form factor in terms of the point-to-patch form factor. The integration step required when enlarging the differential point can be computed numerically by approximating each differential point by a smaller finite element, e_k. If n such elements are used, each of area A_i/n, the patch-to-patch form factor is given by

$$\begin{aligned} F_{ij} &\approx \frac{1}{A_i}\sum_{k=1}^{n}\frac{A_i}{n}F_{e_kP_j} \\ &= \frac{1}{n}\sum_{k=1}^{n}F_{e_kP_j}, \end{aligned}$$

where $F_{e_kP_j}$ is the approximate point-to-patch form factor between the element e_k and the patch P_j.

Contour integration was used by Baum *et. al.* [6], following [36], to compute exactly $F_{dA(\mathbf{x})P_j}$, given in (3.17). A closed-form expression was given for the point-to-patch form factor. Contour integration must also be applied to a polygon which is completely unoccluded when viewed from $\mathbf{x}$ [25]. Goral [26] also used contour integration in the original paper on radiosity for environments consisting of unoccluded polygons.

Cohen and Greenberg [12] introduced the *hemicube* to compute form factors, using a projection method. Essentially, a hemicube is placed over a differential point. Each face of the hemicube is divided into a number of smaller elements by tiling each face with a regular grid. Form factors from the point to each hemicube element can then be precomputed. The form factor from the point to all finite patches in the environment is then approximated by projecting the finite patches onto the hemicube and determining the number of hemicube elements covered by the projection. Other projection surfaces have been proposed, such as hemispheres [21] and planes [55].

Wallace *et. al.* [63] introduced a variation on the point-to-patch form factor by calculating the *point-to-disc* form factor, with the disk having the same area as the patch. The point-to-disc form factor is one of the form factors with a known analytic solution. However, this approximation is only valid when the separation distance between the point and the patch is large, relative to the size of the patch. Radiosity was evaluated at patch vertices, rather than at a point located at the centre of a patch. Previous methods extrapolated this radiosity to the patch vertices and then Gouraud shaded [29] the patches to avoid a faceted appearance. Both the extrapolation step and the shading step introduce error into the radiosity across a patch. Calculating the radiosity at the patch vertices directly, removes the extrapolation step.

Each method for calculating the form factor has its advantages and disadvantages. The hemicube method has been widely adopted because of its simplicity and the ability to perform many of the calculations required, when planar polygons are used as the modelling primitive, in hardware [25]. However, the hemicube method suffers from aliasing caused by the fixed sampling of the hemicube. Ray traced form factors have the advantage that they can support complex geometries. However, they are typically slower than projection methods [63]. For a more detailed discussion of form factor calculation, the reader is referred to [36][25][56][14][21].

3.5 Solution Methods

As stated in section 3.3, once all form factors F_{ij} have been computed, the radiosity equation, given in (3.15) and repeated here for convenience, may be solved.

$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j F_{ij}, \tag{3.21}$$

Equation (3.21) expresses a system of N linear equations with N unknown variables, B_i, one for each patch in the environment [26]. Rearranging (3.21) to a more convenient form gives

$$B_i - \rho_i \sum_{j=1}^{N} B_j F_{ij} = E_i,$$

or, in tableau form,

$$\begin{bmatrix} (1-\rho_1)F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & (1-\rho_2)F_{22} & \dots & -\rho_2 F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \dots & (1-\rho_N)F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix}.$$

Using matrix notation gives

$$\mathbf{MB} = \mathbf{E}, \tag{3.22}$$

where $M_{ij} = \delta_{ij} - \rho_i F_{ij}$ are the elements of $\mathbf{M}$ and δ_{ij} is the Kronecker Delta function defined by

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.23}$$

The system of equations in (3.22) may be solved once the vectors and matrices have been constructed. This involves obtaining the emittance vector, $\mathbf{E}$, computing the matrix of form factors, $\mathbf{M}$, and then inverting matrix $\mathbf{M}$ to give

$$\mathbf{B} = \mathbf{M}^{-1}\mathbf{E}. \tag{3.24}$$

Note that before any images are generated, all entries in $\mathbf{M}$ must be calculated. Thus, this solution method is called *full matrix* (FM) radiosity.

Matrix $\mathbf{M}$ may be inverted using, for example, Gaussian techniques. Such methods are $\mathcal{O}(N^3)$ in complexity and so their use becomes prohibitive for even the simplest of scenes [14]. For this reason *iterative* techniques have become popular in radiosity computations. Iterative techniques are guaranteed to converge because $\mathbf{M}$ is diagonally dominant [27]. The iterative methods often used in radiosity calculations are referred to as *relaxation* methods [14]. Two examples of relation methods are *Jacobi* relaxation and *Gauss-Seidel* relaxation.

3.5.1 Jacobi Relaxation

The simplest relaxation solution method, Jacobi relaxation, uses an initial guess vector, $B^{(0)}$, and proceeds to refine this guess, iterating toward a converged solution. For radiosity calculations, a suitable initial guess vector is given by the vector of emittance values, i.e.

$$B_i^{(0)} = E_i \qquad 1 \leq i \leq N.$$

At each iteration ($m > 0$), a new vector of radiosity values, $B^{(m)}$, is calculated using the update rule [56]

$$B_i^{(m)} = -\sum_{j=1, j\neq i}^{N} \frac{M_{ij}}{M_{ii}} B_j^{(m-1)} + \frac{E_i}{M_{ii}} \qquad 1 \leq i \leq N.$$

For most practical applications, when planar polygons are used as the modelling primitive for each patch, $M_{ii} = 1$ because $F_{ii} = 0$.

3.5.2 Gauss-Seidel Relaxation

Gauss-Seidel relaxation is very similar to Jacobi relaxation except that values already computed during the current iteration are used in the current solution vector. The update rule is given by [56]

$$B_i^{(m)} = -\sum_{j=1}^{i-1} \frac{M_{ij}}{M_{ii}} B_j^{(m)} - \sum_{j=i+1}^{N} \frac{M_{ij}}{M_{ii}} B_j^{(m-1)} + \frac{E_i}{M_{ii}} \qquad 1 \leq i \leq N.$$

Gauss-Seidel relaxation has the advantage over Jacobi relaxation that a single solution vector can be used for all iterations and so its entries can be updated in place [56]. Gauss-Seidel relaxation also has a greater asymptotic convergence rate than Jacobi relaxation [56]. A physical interpretation can be made of the Gauss-Seidel method. At each iteration, light is gathered from all other patches in the environment by the current patch.

Both Jacobi and Gauss-Seidel relaxation methods are terminated when some criterion is met. This criterion may be when a predefined number of iterations is reached or when the radiosity transported during a particular iteration falls below some threshold. Such a threshold may be a certain percentage of the total energy in the environment.

3.5.3 Progressive Refinement

Despite the simplicity of the full matrix methods, they have a significant drawback. The requirement that the matrix of form factors be precomputed can lead to significant start-up costs when scenes containing large numbers of patches are to be solved for. Also, the order in which patches are chosen during the relaxation process is not based on how much they will contribute to the final solution. Hence, it may take many iterations before all the patches which are significantly illuminated are solved for.

In 1988, Cohen *et. al.* proposed *progressive refinement* (PR) radiosity as a solution to these problems [11]. Progressive refinement uses a variation of Southwell relaxation [27] to *shoot* light into the environment. This method requires only one row of the form factor matrix $\mathbf{M}$ to be computed at each iteration. The significant advantage of the method is that the patch which will shoot the most light is chosen at each iteration step. Each patch has a radiosity value B_i and an unshot radiosity value ΔB_i. The patch which will shoot light is chosen by determining which patch has the greatest unshot energy—the unshot radiosity of the patch multiplied by the patch's area. The unshot radiosity of the shooting patch is used to update the radiosity and the unshot radiosity of all other patches in the environment. Once a patch has shot its unshot radiosity, the unshot radiosity value becomes zero. Of course, other shooting patches may update this zero value if they shoot light at the patch during a later iteration of the algorithm. Figure 3.7 summarises the progressive refinement approach. After one iteration of the algorithm, all patches illuminated by one particularly bright light source will contain a certain amount of radiosity. The scene may be rendered to display this

```
for ( i = 1...N ) {
    B_i = E_i;
    ΔB_i = E_i;
}
while ( not converged ) {
    choose i such that ΔB_i * A_i is greatest;
    for ( j = 1...N ) {
        Δrad = ΔB_i * ρ_j F_ji;
        ΔB_j = ΔB_j + Δrad;
        B_j = B_j + Δrad;
    }
    ΔB_i = 0;
    render scene using B_i for each patch i;
}
```

Figure 3.7: Pseudocode for progressive refinement in conventional radiosity.

partial result. As the algorithm progresses, the scene will become progressively brighter as other patches are chosen to shoot light. A fully converged solution will still take as long to converge as FM methods. However, because useful partial results may be displayed, the PR algorithm may be stopped when the result is *visually* acceptable. Choosing the shooting patch based on how much light it will shoot ensures that visual acceptability is likely to be reached much sooner than with other iterative methods. Also, early termination of the iterative solution can mean than many form factors are never evaluated.

3.6 Summary

This chapter has presented some of the fundamental equations and algorithms associated with the radiosity method. By considering the physical process of radiative transport, the rendering equation was defined. In order to simplify the problem of solving the rendering equation, certain assumptions were made about the nature of surfaces in an environment. Surfaces were assumed to be perfect diffuse reflectors and radiosity across a patch was assumed to be constant. This led to the classical radiosity equation, which was expressed as a system of linear equations which were solved using a number of simple iterative techniques.

There are many aspects of the radiosity method not covered here. Glassner has produced a rigorous study of radiosity in his work on image synthesis [25]. Other comprehensive surveys can be found in [56][14].

The next chapter will discuss the use of curved surfaces in the radiosity method. Higher order radiosity will also be introduced which will provide the method implemented in this thesis.

Chapter 4

Radiosity and Curved Surfaces

Much of the research into radiosity has concentrated on reducing the computational expense of the method. As such, planar polygons have been used as the modelling primitive in radiosity systems. Three factors account for this dependence on the polygon:

- Classical radiosity methods approximate the varying radiosity function over a surface using smaller patches of constant radiosity (section 3.3). Subdivision of a surface can be performed efficiently if planar polygons are used to represent both the original surface and the resulting patches.

- Form factor calculations make a significant contribution to the solution time in a radiosity system (section 3.4.5). Such calculations may be simplified using polygons. The polygon support provided by today's graphics hardware can also be used to reduce form factor computation time [25].

- Polygons are the common denominator—any geometry may be approximated by polygons and polygons may be rendered quickly using hardware rendering [56].

This presents us immediately with one method of incorporating curved surfaces into a radiosity system. Curved surfaces may be tessellated and approximated by planar polygons, typically triangles [1]. While this method allows curved surfaces to be used in any standard polygon-based radiosity system, it does have severe drawbacks. The number of polygons generated by tessellating a surface is often large, resulting in high memory costs [64]. Each polygon generated by the tessellation process becomes a surface in the environment, with which all other surfaces must interact. This clearly precludes the use of full matrix radiosity methods and even PR radiosity will be evaluating many form factors. Hierarchical Radiosity (HR) [32][33] may be used to overcome the problems caused by environments consisting of many polygons, by considering interactions between groups of polygons. However, HR also has high start-up costs caused by an *initial linking* phase to construct a hierarchy [23].

Another disadvantage of the tessellation method is that tessellation is based on the geometry of a curved surface, rather than the distribution of light across the surface [68]. Regions of small curvature may be approximated by a small number of polygons. However, if this region has a rapidly varying distribution of light across it, the radiosity function will not be approximated well. Of course, such regions may be split further into a number of patches but this only adds to the number of surfaces in the environment.

Goral [26] suggested that the radiosity method should be extended to handle curved surfaces. This chapter begins with a review of previous research concerning radiosity and curved surfaces.

4.1 Previous Work

Wallace *et. al.* [63] proposed a ray tracing algorithm to overcome the aliasing problems of the hemicube. This method was discussed in the context of form factor calculation in section 3.4.5. While surfaces were tessellated for shading and display, ray casting used during form factor calculation used the original surface geometry. This allowed more accurate form factors to be calculated. Storage costs associated with meshing, during the radiosity solution, were avoided by generating uniformly distributed sample points on a surface, at which radiosity was evaluated. The sampling pattern across a surface could be refined depending on the distance from a vertex to the source patch or the source patch's radiosity. While Wallace's technique avoided aliasing caused by the fixed sampling of the hemicube, it has some disadvantages. Radiosity is computed at patch vertices and so the radiosity of a patch (required when a patch becomes a source from which light is shot) must be approximated as a weighted average of vertex radiosities. Ray casting approaches are also typically slower than hemicube methods [56].

An alternative representation of the radiosity function across a surface was presented by Heckbert [34]. Radiosity is stored in an *adaptive radiosity texture*, or *rex*. A rex records the pattern of light, shadow and colour bleeding on a surface. This representation eliminates the need for a complex data structure used to store a mesh of polygonal elements. Radiosity is associated with a surface point using a mapping between a surface's geometry and its texture map. A three-pass, bidirectional ray tracing scheme was proposed by Heckbert which allowed both diffuse and specular effects to be accounted for. The interaction of a photon with a surface, along its path from a light (L) to the eye (E), is characterised as either specular (S) or diffuse (D). Each path can then be labelled with a string formed from the regular expression $L(D|S)^*E$. Conventional ray tracing simulates $LDS^*E \,|\, LS^*E$ paths and conventional radiosity simulates LD^*E paths. Heckbert's implementation simulated paths given by LS^*DS^*E. Monte Carlo ray tracing was used to sample the radiosity function. A quadtree was used to store rex points, allowing adaptive sampling. However, the mapping between rex points and surface geometry is not always straightforward [14]. Heckbert also reported that excessive amounts of memory were required to store the quadtree because a node in the quadtree is stored for every light ray. During rendering, rex values may be interpolated to reduce the number of rex elements required on a surface. This method, though, does present an effective way of decoupling surface geometry and the radiosity function across the surface.

Vedel [62] also used a sampling data structure, similar to the rex. The radiosity function was reconstructed from an array of weighted samples. By computing the radiosity gradient, an elliptical filter, perpendicular to the radiosity gradient, was used to determine the weight that should be applied to a sample during reconstruction. The elliptical filter is formed by stretching the weight function along the direction of smallest change in the radiosity gradient, and squeezing it along the direction of greatest change. This allows the filter to apply appropriate weights along shadow boundaries, for example. Vedel also uses a ray casting technique to compute form factors. Sample points are constructed on a regular grid over the surface, although the surface is never actually meshed during the radiosity solution process. Rendering, though, does require surfaces to be meshed so that radiosity function sample values may be mapped to the surface geometry.

The previous two methods reconstructed the radiosity function from a set of sample values. The particular methods used made it possible to sample the radiosity function without explicitly meshing the surfaces. Sample points were stored in various data structures. It was the use of ray tracing to evaluate form factors that allowed various geometries to be used, not just polygonal patches. The following methods were proposed specifically with curved surfaces in mind.

Kok *et. al.* [38] used a two-pass method to calculate diffuse and specular effects for Bézier patches. A preprocess was used to calculate the diffuse effects. Regularly distributed sample points were located on each surface and form factors were computed between sample points. Adaptive subdivision of the grid of sample points was used in cases where the radiosity gradient between sample points exceeded some threshold. Form factors were calculated by shooting visibility rays into the environment. Visibility rays were distributed over a unit hemisphere by subdividing the hemisphere into j circles, parallel to the base of the hemisphere. Each ray has a weight factor, given by

$$W_{jk} = \frac{\cos(j\Delta\theta)}{\sum_{i=0}^{k} N_i \cos(i\Delta\theta)} \qquad 0 \le k \le N_j - 1,\ 0 \le j \le k,$$

where k is the size of the frustum formed between a certain number (k) of circles; N_j is the number of rays distributed around the circumference of circle j; $\Delta\theta$ is the constant spacing of the circles. A point-to-patch form factor is evaluated by summing the weights of rays that intersect other patches. The postprocess

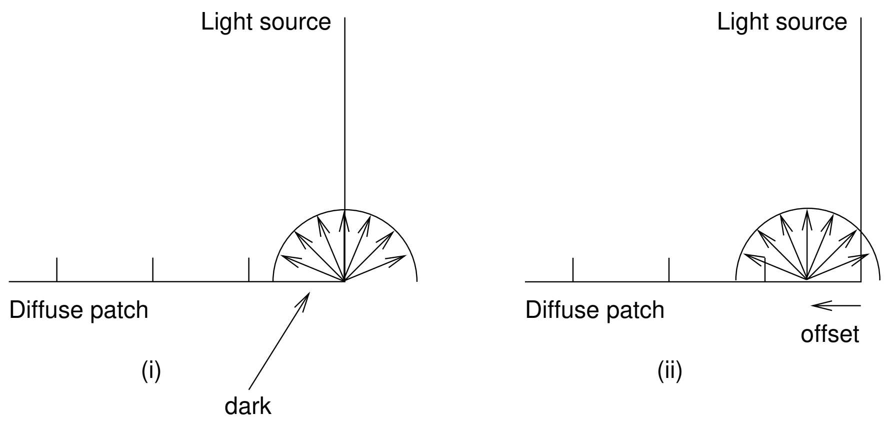

Figure 4.1: Displacing the hemisphere away from a patch (after [38]).

renders the scene using ray tracing to obtain the visible points in a scene. Sample points are interpolated to give the radiosity value across a patch. The postprocess also determines specular effects by tracing a frustum of rays from the intersection point whenever a specular patch is intersected. Note that, in this method, the corner points of patches were selected as sample points. This creates a problem with intersecting patches, as shown in figure 4.1(i). The form factor between the sample point and the light source will be zero because no rays will intersect the light source. Hence, the corner of the receiving patch will be dark. This artifact was removed by displacing the sample point away from the corner of the receiving patch, as shown in figure 4.1(ii). Results reported by Kok *et. al.* state that for a scene consisting of fourteen Bézier surfaces, over 15 million bounding box tests were required during the intersection tests used to evaluate form factors. With such a high number of intersections required, it was proposed that Bézier/ray intersection tests be performed in hardware.

A tessellation approach was used by Bao and Peng [5] to incorporate curved surfaces into the radiosity method. Surfaces were subdivided into triangular patches and radiosity was calculated at the patch vertices. The radiosity method was also extended to account for specular effects in non-diffuse environments. In this method, however, radiosity was assumed to vary across each surface patch (as opposed to being constant in conventional radiosity—section 3.3). A bilinear interpolation scheme was incorporated into the radiosity equations themselves, as well as during the rendering process. Radiosity was calculated at patch vertices. Thus, the radiosity equation, for the lth vertex of the ith patch on surface S_i, is given by

$$B(i_l) = E_i + \rho_i \sum_{k=1}^{N} \sum_{m=1}^{3} B(k_m) F_{S_k i_l, m} \qquad 1 \leq i \leq N,\ l = 1, 2, 3.$$

The patch-to-patch form factors $F_{S_k i_l, m}$ are given by

$$\begin{aligned} F_{S_k i_l, 1} &= \int_{S_k} u F_{i_l, dS_k} \\ F_{S_k i_l, 2} &= \int_{S_k} v F_{i_l, dS_k} \\ F_{S_k i_l, 3} &= \int_{S_k} (1 - u - v) F_{i_l, dS_k}, \end{aligned}$$

with F_{i_l, dS_k} giving the point-to-point form factor between vertex i_l and differential surface element dS_k. While this method avoided the inaccuracies caused by assuming that radiosity is constant across patches, surfaces were still tessellated, and so the problem of tessellating a surface based on geometry *and* the distribution of light across the surface still exists. Bao *et. al.* noted that this problem should be studied further.

The two researchers that produced some of the earliest work on radiosity [41] also published work concerning curved surfaces and radiosity [42], in the same volume as Bao *et. al.* [5]. The area sampling approach proposed by Nishita and Nakamae used a pyramid tracing scheme to evaluate form factors. In a method similar to that given by Wallace [63], surfaces are divided into elements[1]. Elements are further subdivided into sub-elements for use in form factor evaluation. This subdivision of elements into sub-elements is adaptive so that the level of accuracy of form factor calculation can be controlled. The form factor of an element is calculated by multiplying the form factor of all its sub-elements by a shadow ratio. The shadow ratio is given by an intersection test between each surface patch and a pyramid formed between the calculation vertex and the four vertices of an occluding subelement. Bézier clipping is used to perform this intersection test. The form factor of a sub-element is calculated using a projection method onto the base of a hemisphere.

4.2 The Constant Radiosity Assumption

The previous section discussed existing techniques used to incorporate curved surfaces into the radiosity method. Whilst each method has been shown to be effective, they suffer from various problems. As stated in the introduction to this chapter, methods which tessellate curved surfaces suffer from excessive memory usage and tessellation is rarely based on the distribution of radiosity across a surface. Ray tracing methods have been described, which eliminate the need for meshes of polygons to be stored. However, ray tracing methods usually require a large number of sample points to accurately sample the radiosity function [37].

A source of inaccuracy in conventional radiosity is the assumption that radiosity is constant across a patch [59]. Tampieri and Lischinski refer to this as the

[1]Nishita *et. al.* used the term "patch" to mean a Bézier surface and the term "element" to mean the surface produced as a result of subdividing a patch.

Constant Radiosity Assumption (CRA) [59]. They note that radiosity is only constant across an element which is in total shadow and so the CRA is rarely satisfied. The CRA also leads to error during the rendering phase of the radiosity method [68]. As noted in section 3.4.5, vertex radiosity values are interpolated across a patch during rendering. For consistency, constant radiosity should be assumed during rendering if it is assumed during the solution process. However, making such an assumption would lead to a faceted appearance of a rendered solution unless surfaces were subdivided into pixel sized patches. It should be clear by now that this is an unreasonable solution to the problem.

Having looked at conventional radiosity techniques, what can be said about a method that will allow NURBS to be incorporated into the radiosity method?

- Tessellation should be avoided.
- If ray tracing is to be used, the number of ray/surface intersection tests should not be excessive.
- The CRA should not be made.
- Interpolating between sample points on a surface should be avoided.

The next section presents a radiosity method which satisfies these constraints and so allows NURBS to be used as the modelling primitive.

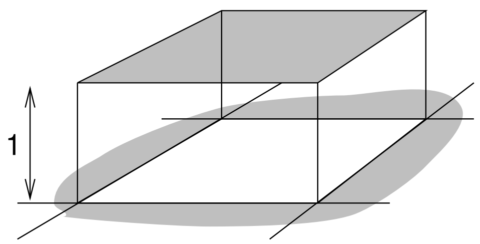

Figure 4.2: A constant basis function defined over the extent of a patch.

4.3 Higher Order Radiosity

As stated in section 3.3, one method of solving the radiosity equation, given in (3.12), is the finite element approach, as used in the classical radiosity method. The finite element method approximates a complicated function by a linear sum of simpler *basis functions* [14]. In the case of classical radiosity, the basis functions, $N_i(\mathbf{x})$, are simply constant functions defined over the extent of a patch, i.e.,

$$N_i(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in P_i \\ 0 & \mathbf{x} \notin P_i \end{cases} \tag{4.1}$$

Figure 4.2 shows such a basis function. The radiosity function, $B(\mathbf{x})$, across a surface is then approximated by a weighted sum of basis functions:

$$B(\mathbf{x}) \approx \sum_{i=1}^{n} B_i N_i(\mathbf{x}).$$

B_i are the weights used to scale the basis functions, i gives the index of a particular patch on a surface and n is the total number of patches defined on a surface. Clearly, using the constant basis function given in (4.1), the radiosity at a point $\mathbf{x}$, contained in the ith patch, is given by a scalar multiple of the basis function defined *only* on that patch. This is the essence of classical radiosity. The constant basis functions have *local support* over the surface, i.e., they are non-zero over a single patch only. All that remains is to determine the weights B_i. If the radiosity function were known, we could simply sample the radiosity function at $\mathbf{x}$ to obtain the weight. Clearly though, if the radiosity function were known, we would not be approximating it with basis functions in the first place. Since the radiosity function is not known, weights are calculated in such a way that the error between the approximated function and the actual function is minimised, in some sense.

The ability to represent an unknown function using scaled basis functions is at the heart of the method implemented by the author in this thesis. Essentially, meshing can be avoided by using basis functions that have *global* support over an entire surface. Clearly, constant basis functions defined over an entire surface will not allow a varying radiosity function be be represented (this is the very reason surfaces were split into patches). Thus, *higher order* basis functions must be used. The method to be described is the Galerkin Radiosity method, first suggested in the context of radiosity, in the plane, by Heckbert [35] and later extended to three-dimensional radiosity by Zatz [68][69].

Before Galerkin radiosity is discussed, the Galerkin method in the context of integral equations in general is presented [17]. This begins with a classification of the integral equation found in the radiosity method.

4.3.1 Fredholm Integral Equations

The radiosity equation given in (3.12) is an example of a *linear inhomogeneous singular Fredholm integral equation of the second kind* [17]. Such an equation is of the form

$$x(s) = y(s) + \int_a^b K(s,u)x(u)\,du. \tag{4.2}$$

x is the unknown function for which the equation must be solved. y and K are both known functions. Function y is often called the *driving term* and K is referred to as the *kernel*. The equation is linear because the integrand is simply the product of an unknown function and a known function. It is inhomogeneous because the driving term is non-zero. The singular nature of the equation can be attributed to the $1/\|\mathbf{x}-\mathbf{x}'\|^2$ term in (3.12). Some theory concerning such integral equations is now given. Other texts on the subject of integral equations ([17][4] and also [25]) present this material using univariate functions and state that it can be generalised to functions of two or more variables. This thesis requires that the integral equation be defined using bivariate functions (in preparation for NURBS). As such, bivariate functions will be used in the theory given below. The two parameters are simply a means of specifying some point in the environment. Thus, (4.2) becomes

$$x(s,t) = y(s,t) + \int_a^b \int_c^d K(s,t,u,v)x(u,v)\,dudv, \tag{4.3}$$

where bivariate functions are defined over the domain $[a,b]\times[c,d]$. However, the remainder of this thesis will assume that $c = a$ and $d = b$, i.e., bivariate functions are defined over the domain $[a,b]\times[a,b]$.

For notational convenience, the *kernel integral operator* $\mathcal{K}$ is introduced and defined by

$$(\mathcal{K}x)(s,t) = \int_a^b \int_a^b K(s,t,u,v)x(u,v)\,dudv.$$

Equation (4.3) may now be written

$$x = y + \mathcal{K}x. \tag{4.4}$$

Note that the dependence of x, y and $(\mathcal{K}x)$ on parameters s, t, u and v is taken as being implicit. To describe a method of solving this equation, the following concepts are required:

- The function space $\mathcal{L}^2(a,b)$ is defined by the set of all functions $x(s,t)$ that satisfy

$$\int_a^b \int_a^b |x(s,t)|^2 ds dt < \infty.$$

  If a function $x(s,t)$ is in the function space $\mathcal{L}^2(a,b)$ then $x(s,t)$ is called an $\mathcal{L}^2$ function. It is assumed that all functions x and y are $\mathcal{L}^2$ functions in this thesis. The kernel function is assumed to satisfy

$$\int_a^b \int_a^b \int_a^b \int_a^b |K(s,t,u,v)|^2 ds dt du dv < \infty.$$

- The L_2 *norm* of an $\mathcal{L}^2$ function is defined by

$$\|x\|_2 = \left\{ \int_a^b \int_a^b |x(s,t)|^2 \, ds dt \right\}^{1/2}.$$

  Note that the subscript 2 will be omitted from the norm $\|\cdot\|_2$ and is taken to be implicit.

- The norm $\|\mathcal{K}\|$ of the operator $\mathcal{K}$ is defined by

$$\|\mathcal{K}\| = \sup_{x \in \mathcal{L}^2, x \neq 0} \frac{\|\mathcal{K}x\|}{\|x\|}.$$

  $\mathcal{K}$ is defined as an operator which maps the $\mathcal{L}^2$ function x onto the $\mathcal{L}^2$ function $\mathcal{K}x$.

- The *inner product* $<x,y>_{\mathcal{W}}$ between two *real* $\mathcal{L}^2$ functions x and y, with respect to a weight function $\mathcal{W}$, is defined by

$$< x, y >_{\mathcal{W}} = \int_a^b \int_a^b x(s,t) y(s,t) \mathcal{W}(s,t) \, ds dt. \tag{4.5}$$

With these definitions it can be stated, without proof given here, that in general, solutions of (4.4) exist in infinite-dimensional *Hilbert Space* [67]. Given these mathematical tools we can now take steps to solve the integral equation given in (4.4).

Note that one solution of the equation is given by rearranging the equation and expanding it into an infinite Neumann series [17]. Rearranging gives

$$x = (\mathcal{I} - \mathcal{K})^{-1}y,$$

where $\mathcal{I}$ is the identity operator. If $\|\mathcal{K}\| < 1$ then

$$\begin{aligned} x &= y + \mathcal{K}y + \mathcal{K}^2 y + \cdots \\ &= y + \sum_{i=1}^{\infty} \mathcal{K}^i y. \end{aligned} \tag{4.6}$$

Delves [17] showed that if indeed $\|\mathcal{K}\| < 1$ then (4.6) is an exact solution of (4.4). Kajiya [37] also noted that the Neumann series expansion corresponds to the iterative solutions of full matrix radiosity (section 3.5). Each term in (4.6) corresponds to an iteration in the solution process. It was also noted that applying the operator $\mathcal{K}$ is equivalent to one bounce of light in the environment. Since the process of reflection cannot reflect more light than was incident, it can be concluded that, in fact $\|\mathcal{K}\| < 1$. However, as Zatz points out, in practice this form of analytic application of the kernel operator $\mathcal{K}$ can only be performed once, in a limited number of cases and so a numerical solution method must be found [68].

4.3.2 The Galerkin Method

The Galerkin method[2] [17][4] is a *projection method* which projects the solution of the integral equation (4.4) from one function space to another. As stated in the previous section, the solution of the integral equation lies in an infinite dimensional space (Hilbert space). To be able to compute a solution, the exact solution must be projected into a finite space. Recall from the discussion at the start of section 4.3 that a function may be represented as a linear combination of basis functions. Thus, the exact solution is given by

$$x(s,t) = \sum_{i=1}^{\infty} x_i N_i(s,t),$$

where x_i are the weights. Defining a *projection operator* $\mathcal{P}_n$ which projects functions from the infinite dimensional space to the finite dimensional space, the approximated representation of the solution is given by

$$x_n(s,t) = \mathcal{P}_n x(s,t) = \sum_{i=1}^{n} x_i N_i(s,t). \tag{4.7}$$

To measure, in some sense, the "closeness" of this approximation to the actual solution, a *residual function*, r is defined by

$$r = x_n - \mathcal{K} x_n - y.$$

The Galerkin method requires that the "closeness" of the approximation should be such that the residual is *orthogonal* to all basis functions spanning the finite space into which the exact solution is projected [56], i.e.,

$$< N_j, r >= 0 \qquad \forall j. \tag{4.8}$$

[2]Also called the *Ritz-Galerkin method* or the *Method of Moments.*

The weight function used by the inner product is identically 1 in this case. The basis functions are also defined to be *orthonormal* [4]. This simplifies the equations to be presented later. Basis functions N_i and N_j are orthonormal if

$$< N_j, N_i >= \delta_{ij},$$

where δ_{ij} is the Kronecker Delta function defined by (3.23). Expanding r in (4.8) gives

$$\begin{aligned} 0 &= < N_j, x_n - \mathcal{K}x_n - y > \\ &= < N_j, \sum_{i=1}^{n} x_i N_i - \sum_{i=1}^{n} x_i \mathcal{K} N_i - y > \\ &= \sum_{i=1}^{n} x_i < N_j, N_i > - \sum_{i=1}^{n} x_i < N_j, \mathcal{K} N_i > - < N_j, y > \end{aligned} \tag{4.9}$$

All that is required now is to solve this equation to determine the weights x_i. Both y and $\mathcal{K}N_i$ are defined in terms of known functions. The basis functions are to be chosen and so they are also known. (4.5) defines the inner product in terms of integrals (of known functions) which may be evaluated numerically [57].

Having solved (4.9) to obtain the weights x_i, the unknown function $x(s,t)$ can be approximated by evaluating (4.7).

4.4 Galerkin Radiosity

Having given the theory behind the Galerkin method for solving integral equations, a description of how the method can be used to obtain the distribution of light in a scene is now given. The following section summarises the method developed by Zatz [68].

Note that the previous discussion cited the radiosity equation given by (3.12). This equation integrated over every point in the environment. We now consider the form of the radiosity equation given by (3.13) but with some significant changes:

- Surfaces within the environment are not subdivided into patches. Only entire surfaces are considered.
- Radiosity, emittance and reflectance, need not be considered constant across a surface.
- Terms in the equation are given as functions of two parametric variables rather than of one point, $\mathbf{x}$, in the environment. The parametric variables specify a particular point on a surface with $(s,t) \in [-1,1] \times [-1,1]$ (section 2.1).

Hence, the radiosity equation is given by

$$B_i(s,t) = E_i(s,t) + \sum_{j=1}^{N} \int_{-1}^{1}\int_{-1}^{1} K_{ij}(s,t,u,v) B_j(u,v)\,du dv, \tag{4.10}$$

where $B_i(s,t)$ is the radiosity on the ith surface at a point given by (s,t); $E_i(s,t)$ is similarly the emittance; $K_{ij}(s,t,u,v)$ is the kernel function given by

$$K_{ij}(s,t,u,v) = \rho_i(s,t) F_{ij}(s,t,u,v) V_{ij}(s,t,u,v). \tag{4.11}$$

In this case, F_{ij} is the *point-to-point* form factor between point (s,t) on surface i and point (u,v) on surface j. The point-to-point form factor was given in section 3.4.1. Written parametrically, the form factor becomes

$$F_{ij}(s,t,u,v) = dA_j(u,v) \cdot \\ \frac{(\hat{\mathbf{n}}_i(s,t) \cdot (\mathbf{x}_j(u,v) - \mathbf{x}_i(s,t)))(\hat{\mathbf{n}}_j(u,v) \cdot (\mathbf{x}_i(s,t) - \mathbf{x}_j(u,v)))}{\pi \|\mathbf{x}_i(s,t) - \mathbf{x}_j(u,v)\|^4},$$

where $\hat{\mathbf{n}}_i(s,t)$ is the unit normal at the point given by (s,t), $\mathbf{x}_i(s,t)$ is the actual point in $\mathbb{E}^3$ given by (s,t) and $dA_j(u,v)$ is the differential area of the point given by (u,v) on surface j.

The Galerkin method was applied by Zatz [68] as follows:

A basis set of two-dimensional basis functions $N_k(s,t)$, is defined for each surface i in the environment:

$$\{N_k(s,t) | k = 1, \ldots, n_i\},$$

where n_i is the number of basis functions in the basis set. Each surface in the environment may have a different number of basis functions. Given the radiosity function $B_i(s,t)$ over the surface i, define coefficients B_i^k by

$$B_i^k =< B_i(s,t), N_k(s,t) >_{\mathcal{W}} . \tag{4.12}$$

The radiosity function is then projected into the basis set, i.e., the original function's best approximation in the basis set $\{N_k(s,t)\}$ is given by

$$B_i(s,t) \approx \sum_{k=1}^{n_i} B_i^k N_k(s,t). \tag{4.13}$$

At first sight this does not seem to be of much help since we require the original function to evaluate the coefficients. This paradox will be resolved though. Proceeding, replace $B_j(u,v)$ in (4.10) with the expansion given in (4.13) (using an index l instead of k):

$$B_i(s,t) = E_i(s,t) + \sum_{j=1}^{N} \sum_{l=1}^{n_j} B_j^l \int_{-1}^{1} \int_{-1}^{1} K_{ij}(s,t,u,v) N_l(u,v) \, dudv. \tag{4.14}$$

Note that the B_j^l coefficient is simply a constant so can be moved out of the integral. Taking the inner product of all terms in (4.14) with a basis function

$N_i(s,t)$ gives

$$< B_i(s,t), N_i(s,t) >_{\mathcal{W}} = < E_i(s,t), N_i(s,t) >_{\mathcal{W}} +$$
$$\sum_{j=1}^{N} \sum_{l=1}^{n_j} B_j^l < \int_{-1}^{1}\int_{-1}^{1} K_{ij}(s,t,u,v) N_l(u,v)\, du dv, N_k(s,t) >_{\mathcal{W}} .$$

But by (4.12) this gives

$$B_i^k = E_i^k + \sum_{j=1}^{N} \sum_{l=1}^{n_j} B_j^l < \int_{-1}^{1}\int_{-1}^{1} K_{ij}(s,t,u,v) N_l(u,v)\, du dv, N_k(s,t) >_{\mathcal{W}} . \quad (4.15)$$

The remaining inner product contains only known functions and so can be computed. Denoting this inner product by K_{ij}^{kl} (which is referred to as the *kernel matrix*), (4.15) is written concisely

$$B_i^k = E_i^k + \sum_{j,l} B_j^l K_{ij}^{kl}. \quad (4.16)$$

There is now a strong similarity between this form of the radiosity equation and that given in (3.15). The difference between the two forms is that the coefficients given here refer to some function representing part of the distribution of radiosity across a surface, rather than simply a constant value [68]. The paradox concerning the use of the radiosity function to obtain the B_i^k coefficients has been resolved. The B_i^k coefficients are found by solving (4.16) using any of the iterative methods presented in section 3.5. The radiosity function is then reconstructed using these coefficients in (4.13). Prior to an iterative solution, the E_i^K coefficients must be computed. During the iterative solution, the K_{ij}^{kl} coefficients will be computed. Both coefficients can be computed because they involve only known functions.

4.4.1 Basis Sets

To calculate coefficients B_i^k, E_i^k and K_{ij}^{kl}, a basis set is required. As stated above, this is a set of two-dimensional basis functions. A 2D basis function is the tensor product of a two 1D basis function. A *solution degree* is specified for each surface. This represents the highest degree that a 2D basis function may have in the set. The degree of a particular 2D basis function is given by the sum of the degrees of the 1D basis functions in the tensor product.

The method proposed by Zatz used normalised[3] Legendre polynomials and normalised Jacobi polynomials [49] as 1D basis functions. The majority of basis sets were constructed using tensor products of Legendre polynomials. However, recall that the rendering equation is singular because of the $1/\|\mathbf{x}-\mathbf{x}'\|^2$ term in the kernel due to the form factor. When two surfaces meet at a common edge, two points $\mathbf{x}$ and $\mathbf{x}'$ approach each other, and so the kernel function approaches infinity. To remove this singularity, the as yet unspecified weight function $\mathcal{W}$ was utilised. The inner product between two functions was given in (4.5) as

$$< x, y >_{\mathcal{W}} = \int_a^b \int_a^b x(s,t) y(s,t) \mathcal{W}(s,t)\, ds dt,$$

Legendre polynomials are formed when the weight function is given by $\mathcal{W}(s,t) = 1$. Jacobi polynomials, denoted $P^{(2,2)}$, use a weight function

$$\mathcal{W}(s,t) = (1-s^2)^2(1-t^2)^2.$$

The zero of order 2 in the weight function will cancel the pole of order 2 in the kernel. Zatz noted that the Jacobi polynomials are not a good candidate for basis sets when the surfaces being considered do not involve a singularity. As such a hybrid method was used whereby the Legendre polynomials were used for "non-singular transfers" and Jacobi polynomials were used for "singular transfers". The coefficients produced by the Jacobi polynomials were then converted back

[3]Normalising polynomials ensures the polynomials are orthonormal. Unless otherwise stated, it will be assumed all polynomials are normalised.

into coefficients of Legendre polynomials to allow these coefficients to be used in later non-singular transfers.

4.4.2 Numerical Quadrature

Evaluating the coefficients in (4.16) is a matter of evaluating the 2D integrals given by the inner products, and the 4D integral given by the kernel matrix. Gaussian quadrature [17] is used to evaluate these integrals. Gaussian quadrature evaluates an integral of a function by summing weighted function samples. The points at which to sample the function are called *quadrature points*[4]. Hence, for a list of quadrature points $\{\xi_i | i = 1, \ldots, N\}$ and weights $\{w_i | i = 1, \ldots, N\}$, a 1D function $f(x)$ is integrated using

$$\int_a^b f(x)\mathcal{W}(x)dx \approx \sum_{i=1}^{N} f(\xi_i)w_i, \tag{4.17}$$

where $\mathcal{W}$ is a weight function to be specified. Gaussian quadrature rules have the property that they can integrate exactly, 1D polynomial integrands up to degree $2N - 1$ using N sample points. Integrals of higher dimensions, e.g. the inner products and the kernel matrix coefficients, can be evaluated using tensor products of single dimensional rules [57]. Note that the weight function $\mathcal{W}$ in (4.17) has yet to be specified. The weight function determines the quadrature points and weights computed. For $\mathcal{W} = 1$ a Gauss-Legendre rule is formed. Similarly, by using the same weight function used by the Jacobi basis set, a Gauss-Jacobi rule is formed. Methods for computing quadrature points and weights are summarised by Zatz [68] and discussed in [58], which also contains comprehensive tables of quadrature points and weights. Delves [17] notes that computing quadrature

[4]Also called *abscissae* or *nodes*.

points and weights is relatively expensive and so, in practice tables of such data are precomputed and stored.

Recall that in the Galerkin radiosity method, a radiosity function is projected into a basis set by taking the inner product of the radiosity function with a basis function. Consider a 1D radiosity function and 1D basis functions: If the basis function is of degree N, the radiosity function is being approximated by a function also of degree N and so the resulting integrand, when evaluating the inner product, will be of degree $2N$. Therefore the quadrature rule must use $N+1$ quadrature points and weights to evaluate the integral accurately. Hence, using 2D radiosity functions and basis functions, over emitting surfaces and receiving surfaces, the Galerkin radiosity method is $\mathcal{O}((N+1)^4)$ in complexity.

More information concerning quadrature may be found in [58][17][68]. With regard to this thesis, the point is that the Galerkin radiosity equation in (4.16) requires certain integrals to be evaluated. These integrals *can* be evaluated using numerical quadrature and the quadrature points used in the evaluation give the sample points on a NURBS surface. Increasing the solution degree of a surface increases the number of sample points on that surface, thus increasing the accuracy of solution for that surface.

4.4.3 Shadow Masks

Modelling shadows caused by occlusion can be problematic in a global illumination method [68]. Constant radiosity methods have used adaptive meshing in an attempt to capture the *discontinuities* in the radiosity function caused by shadows [13]. Adaptive subdivision allows the size of the patches in a mesh across a surface to vary. Many smaller patches maybe used in regions where shadow edges are found. Larger patches are used in regions that have almost constant radiosity. Discontinuity meshing also captures shadow detail, aligning mesh elements with shadow discontinuities by considering the geometric intersections of all surfaces [14][22]. The Galerkin method under consideration in this thesis avoids meshing and so an alternative method must be used. As Zatz states, the simplest method of accounting for shadows is to ignore them. A discontinuous radiosity function is then approximated by basis functions in the usual way. Unfortunately, low order basis functions cannot model accurately a discontinuous function, and so Gibbs ringing artifacts, in the form of rippled shadows, are present in images. Increasing the degree of the basis set defined on a surface reduces the intensity of the ripples, but they never completely disappear. The proposed solution, by Zatz, is to used a *shadow mask*. A shadow mask can be thought of as a texture map used to capture significant shadow information. Shadow masks are defined between receiving surfaces and emitting surfaces. A particular receiving surface may have several shadow masks defined across it, one for each emitting surface which is expected to cause shadows. The shadow mask function between a receiving surface i and an emitting surface j is given by

$$M_{ij}(s,t) = \frac{\int_{-1}^{1}\int_{-1}^{1} V_{ij}(s,t,u,v)\,du dv}{\int_{-1}^{1}\int_{-1}^{1} du dv},$$

where the denominator normalises M to the range $[0,1]$. Shadow masks are calculated using multiple point-to-point view tests between the receiver and the emitter. Shadow masks can be precomputed and stored in a suitable data structure for the sampling pattern used. Zatz used a regular grid of points across surfaces and used bilinear interpolation to evaluate the mask function at non-sample points. Shadow masks were incorporated into the radiosity equation by allowing emitting surfaces to interact with either the shadow mask on a surface, if the mask is defined for that emitter, or with the standard basis set across the surface. In the case of an emitter interacting with a shadow mask, the standard visibility test used when evaluating the kernel (4.11) is ignored. If an emitter interacts with the standard basis set across a surface then the usual visibility test is required. When a shadow masked surface is selected to emit radiosity, the shadow mask is used to weight the value of the radiosity function at a particular point. Hence shadow masks are not only used during the rendering stage as texture maps, but also during the solution stage to modify how radiosity is emitted by a particular surface.

Zatz reported that shadow masks only increment the number of surfaces in an environment and so have performance advantages over shadow meshing techniques, which can significantly increase the number of surfaces. However, shadow masks have significant drawbacks. The shadow mask is effectively resorting back to techniques used in constant radiosity. The sampling of a surface's occlusion with respect to an emitter, on a regular grid of sample points is a form of meshing that the higher order method was used to eliminate. Using interpolation to evaluate the mask at non-sample points also introduces the Constant Radiosity Assumption into the higher order method.

4.5 Summary

This chapter has presented a review of early techniques used to incorporate curved surfaces into the radiosity method. Meshing can be avoided using higher order techniques such as Galerkin radiosity. The Galerkin method solves the radiosity integral equation by projecting radiosity functions into finite basis sets. The integrals used to perform this projection are evaluated using numerical quadrature techniques. The resulting radiosity *coefficients* form a system of equations that are solved using the iterative methods of conventional radiosity. A particular advantage of the Galerkin method is that having obtained the radiosity coefficients, the radiosity function can be evaluated at any parametric point and so inappropriate interpolation during the rendering phase of conventional radiosity is avoided. The use of parametric variables to describe the radiosity function allows NURBS to be used as the modelling primitive.

Despite these advantages, there are particular disadvantages. Using shadow masks to overcome the basis functions' inability to capture shadow discontinuities is a "hack" which simply resorts back to the constant radiosity assumption. The Galerkin method may also be compute-intensive. The $\mathcal{O}((N+1)^4)$ cost of the method means it does not offer any speed improvements over conventional radiosity. Zatz reports that solution times tend to be the same for both methods. However, it is also reported that the Galerkin method uses significantly less memory [68].

Implementation details of this work are now given.

Chapter 5

Implementation Issues

This chapter discusses implementation details of Galerkin radiosity, the method implemented in this thesis to incorporate NURBS in to the radiosity method.

5.1 Describing Scenes

Now that a method for obtaining the radiosity at any point in an environment is available, we require some means of describing the environment. The Manchester Scene Description Language (MSDL) [50] has been used in this implementation. MSDL allows scenes to be described using a variety of modelling primitives, such as polygons, spheres and NURBS, although only NURBS have been used by the author. Affine transformations may be applied to objects to scale/rotate/shift objects within the scene. The ability to apply such transformations to scene objects is a useful property of a modelling system. Objects may be defined in one coordinate system which may be a particularly convenient coordinate system for the object's definition. The object may then be instantiated in another coordinate system (for example, the world coordinate system) by applying affine transformations to the object.

5.1.1 Additional Surface Parameters

As stated in section 4.4.1, the accuracy of the approximation of the radiosity function across a surface is specified by a solution degree for that surface. To allow this information to be specified in the MSDL description of a scene, MSDL's *verbatim blocks* must be utilised. Verbatim blocks are a means of including arbitrary text as part of an object's description. MSDL passes the text through to the calling application without making any attempt to understand the text. The calling application must extract any necessary information from the text in a suitable manner.

Verbatim blocks are also used to specify two-sided surfaces in this implementation. When planar polygons are used as the modelling primitive, it can be assumed that a particular polygon cannot transfer light to itself—it cannot "see itself." Using NURBS allows the possibility that a curved surface may be able to transfer light not only to itself but also to its rear side. As such, surfaces can be designated as single sided or double sided. Clearly, radiosity coefficients must be maintained separately for each side of a surface. However, there will be certain circumstances in which a surface requires only one side. For example, the planer wall of a room. Calculating radiosity coefficients for the unseen[1] side becomes

[1]Unseen from any other point in the environment, not just from an observer's location.

unnecessary. Hence, specifying a surface as either one-sided or two-sided provides a convenient mechanism to model such surfaces.

5.1.2 NURBS Functions

While MSDL allows NURBS to be described in a scene, they must still be represented internally in a convenient form. The author has used the *Nlib* NURBS library [47] to store and manipulate NURBS. The Nlib library is an object based library providing implementations of many of the algorithms described in [48]. During the reading of a scene, the MSDL specification of a NURBS surface is converted to the Nlib specification. Any transformation matrix, composed in MSDL to apply affine transforms to an object, is also converted to Nlib's matrix format. This allows easy application of the transformation matrix to the NURBS surface using Nlib functions. Before the radiosity solution can be computed for an environment, each knot vector of every NURBS surface must be scaled to the range $[-1, 1]$. Again, Nlib provides a convenient mechanism to perform this scaling. Note that such a scaling does not alter the geometric properties of a NURBS surface in any way.

5.2 Reflectance and Emittance Functions

As stated in section 4.4, reflectance and emittance need not be considered constant across a surface. This implementation uses NURBS *surface functions* to specify reflectance and emittance as varying functions. A surface function can be thought of as a NURBS surface where control points are specified using scalar

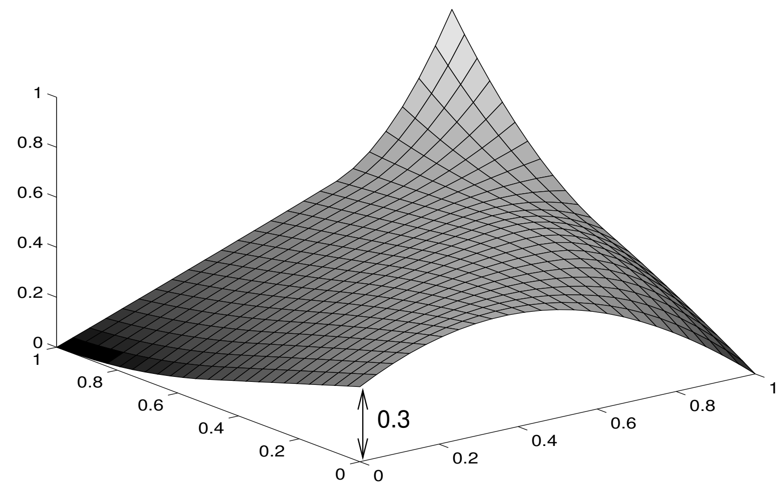

Figure 5.1: A NURBS surface function used to specify varying reflectance or emittance over a surface.

values only. The surface function is then defined on a regular grid of these scalar values. Figure 5.1 shows a surface function defined on the knot vector $U = \{0.0, 0.0, 0.0, 0.5, 1.0, 1.0, 1.0\}$ and $V = \{0.0, 0.0, 0.0, 0.7, 1.0, 1.0, 1.0\}$. The scalar "control points" are given by

$$\begin{array}{cccc} 0.3 & 0.6 & 0.2 & 0.0 \\ 0.2 & 0.5 & 0.5 & 0.2 \\ 0.0 & 0.3 & 0.5 & 0.5 \\ 0.0 & 0.2 & 0.5 & 1.0 \end{array}$$

Using a surface function for each spectral sample allows the reflectance or emittance to be specified. Of course, such functions must be projected into the basis set for a particular surface before a radiosity solution can be found. Hence, a rapidly varying, or discontinuous function, will not be approximated well using Legendre Polynomials. However, surface functions provide a convenient mechanism for specifying non-constant reflectance and emittance.

MSDL verbatim blocks are again used to allow the scene designer to utilise such functions. A keyword is used to indicate whether the emittance and reflectance of a particular surface is to be given as a surface function. If so, then the surface function descriptions are read from the verbatim block. Otherwise, constant values are read. The reflectance surface function should be in the range $[0, 1]$ but the emittance function may take any positive value.

Appendix A gives an example MSDL file containing surface functions.

5.3 Basis Functions and Quadrature

The basis functions used in this implementation are 2D Legendre polynomials. As discussed in section 4.4.1, Zatz used the weight function associated with Jacobi polynomials to remove the singularity in the kernel function when two surfaces met at common edges. However, this method of removing the singularity can only be used when the singularity is known to occur at the *edges* of adjacent surfaces, i.e. along isoparametric lines specified by the parametric values $(s, t, u, v) \in \{-1, 1\}$. When two surfaces met along isoparametric lines other than at the edges of the surfaces, the shooting surface was temporarily meshed for the transfer of light to the receiving surface. As noted in [14], using a particular weight function to remove singularities does not, in practice, avoid a possible division by zero. The weight function simply goes to zero as fast as the $1/\|\mathbf{x} - \mathbf{x}'\|^2$ term in the kernel. This implementation ignores the singularity in the kernel, noting that Gaussian quadrature nodes for Gauss-Legendre rules never lie on the edges of surfaces. This is because the quadrature nodes are interior to the range of integration for a particular integral [58]. It is noted, however, that the Galerkin radiosity method should be extended to detect and remove singularities occurring anywhere on a surface. Ideally, the method should detect the intersection of two NURBS surface and extract the curve of intersection. A weight function can then be constructed to remove the singularities occurring along this curve. However, computing such a weight function would be extremely compute-intensive, not to mention mathematically challenging.

5.3.1 Legendre Polynomials

One dimensional Legendre polynomials $P_n(x)$ are constructed using the three-point recurrence relation [20]

$$(n+1)P_{n+1}(x) = (2n+1)P_n(x) - nP_{n-1}(x),$$

where n is the degree of the Legendre polynomial. Normalisation of the polynomials is achieved using a scaling of $(n+1/2)^{1/2}$. A basis set of two-dimensional Legendre polynomials is represented by two lists, one for each dimension. The list elements give the degrees of Legendre polynomials required for each basis function. The particular degrees p, q of Legendre polynomials $P_p(s)$ and $P_q(t)$, in an nth degree basis set, are given by $\{(p,q) \mid p+q \leq n \; \forall \, p,q \in [0,n]\}$. Hence each list will contain $(n+1)(n+2)/2$ elements. A basis function $N_k(s,t)$ is evaluated at (s,t) by evaluating the required one-dimensional Legendre polynomials $P_p(s)$ and $P_q(t)$ and taking the product of the results. Note that the subscript k is simply an index into the 2D basis set whereas the subscripts p and q give the degree of the 1D Legendre polynomials. Figure 5.2 shows the six unnormalised basis functions of a second degree basis set. Every surface in the environment has associated with it the two lists of Legendre polynomial degrees, determined

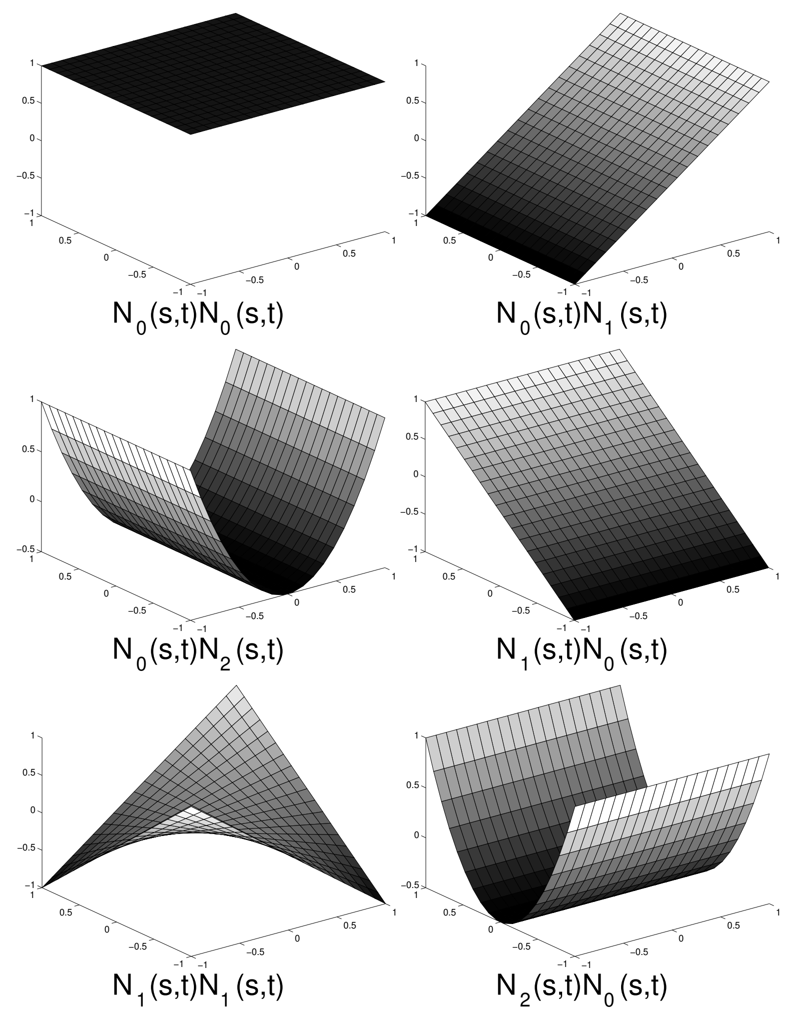

Figure 5.2: Six unnormalised basis functions of a second degree basis set.

using the solution degree of the surface.

5.3.2 Numerical Quadrature

Galerkin coefficients are computed by evaluating integrals using Gaussian Quadrature (section 4.4.2). Since Legendre polynomials are used to construct the basis functions, Gauss-Legendre quadrature rules are used to evaluate integrals containing these basis functions. For example, the emittance coefficients are computed

Degrees	Points	Weights
0,1	0.00000	2.00000
2,3	0.57735	1.00000
	-0.57735	1.00000
4,5	0.77460	0.55556
	0.00000	0.88889
	-0.77460	0.55556
6,7	0.86114	0.34785
	0.33998	0.65215
	-0.33998	0.65215
	-0.86114	0.34785
8,9	0.90618	0.23693
	0.53847	0.47863
	0.00000	0.56889
	-0.53847	0.47863
	-0.90618	0.23693

Table 5.1: Gaussian quadrature points and weights used to integrate functions of various degrees (from [58]).

using

$$\begin{aligned} E_i^k &= < E_i(s,t), N_k(s,t) > \\ &= \int_{-1}^{1}\int_{-1}^{1} E_i(s,t)N_k(s,t)\, ds dt \\ &\approx \sum_{\alpha}^{n_\alpha}\sum_{\beta}^{n_\beta} E_i(\xi_\alpha,\xi_\beta)N_k(\xi_\alpha,\xi_\beta)w_\alpha w_\beta, \end{aligned}$$

where ξ_α and ξ_β are the quadrature points required to integrate the functions in each dimension; w_α and w_β are the weights associated with each quadrature point; n_α and n_β are the number of quadrature points and weights used in such an integration; The quadrature points and weights are obtained using a lookup table of quadrature points and weights required to integrate a function of a specified degree. Table 5.1 shows the quadrature points and weights for functions of various degrees.

The Kernel matrix given in section 4.4 is computed similarly:

$$\begin{aligned}
K_{ij}^{kl} &= < \int_{-1}^{1}\int_{-1}^{1} K_{ij}(s,t,u,v)N_l(u,v)\,dudv, N_k(s,t) > \\
&= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} K_{ij}(s,t,u,v)N_l(u,v)N_k(s,t)n_l(s,t)\,dudvdsdt \\
&\approx \sum_{\alpha}^{n_k}\sum_{\beta}^{n_k}\sum_{\gamma}^{n_l}\sum_{\delta}^{n_l} K_{ij}(\xi_\alpha,\xi_\beta,\xi_\gamma,\xi_\delta)N_k(\xi_\alpha,\xi_\beta)N_l(\xi_\gamma,\xi_\delta)w_\alpha w_\beta w_\gamma w_\delta.
\end{aligned}$$

In conventional radiosity, radiosity values are typically weighted by the area of the patch from which the radiosity function was sampled. For example, the progressive refinement solution method described in section 3.5.3 uses a patch's energy (radiosity multiplied by patch area) to determine the shooting order. Galerkin radiosity weights sample points using *area/basis weights*. The area/basis weight is given by the inner product of the differential area of a sample point on a NURBS surface with a basis function:

$$\begin{aligned}
A_i^k &= < dA_i(s,t), N_k(s,t) > \\
&= \int_{-1}^{1}\int_{-1}^{1} dA_i(s,t)N_k(s,t)\,dsdt \\
&\approx \sum_{\alpha}^{n_\alpha}\sum_{\beta}^{n_\beta} dA_i(\xi_\alpha,\xi_\beta)N_k(\xi_\alpha,\xi_\beta)w_\alpha w_\beta.
\end{aligned}$$

The differential area of a point on a NURBS surface is given by

$$dA_i(s,t) = \left\| \frac{\partial \mathbf{x}_i(s,t)}{\partial s} \times \frac{\partial \mathbf{x}_i(s,t)}{\partial t} \right\|,$$

where partial derivatives of the surface at point $\mathbf{x}$ are obtained using (2.26) and (2.27).

```
for ( i = 1...N )
    for ( k = 1...n_i ) {
        E_i^k =< E_i(s,t), N_k(s,t) >;
        B_i^k = E_i^k;
        ΔB_i^k = E_i^k;
        A_i^k =< dA_i(s,t), |N_k(s,t)| >;
    }
while ( not converged ) {
    choose i,k such that |ΔB_i^k| * A_i^k is greatest;
    for ( j = 1...N )
        for ( l = 1...n_j ) {
            K_ji^lk =< ∫_{-1}^{1} ∫_{-1}^{1} K_ji(s,t,u,v) N_k(u,v) dudv, N_l(s,t) >;
            Δrad = ΔB_i^k * K_ji^lk;
            ΔB_j^l = ΔB_j^l + Δrad;
            B_j^l = B_j^l + Δrad;
        }
        ΔB_i^k = 0;
        render scene using B_i^k for each surface i;
}
```

Figure 5.3: Pseudocode for progressive refinement in Galerkin radiosity.

5.4 Solution Method

The linear system of equations given by (4.16) is solved using progressive refinement (PR) radiosity. The PR method was described in section 3.5.3. For higher order radiosity several modifications are made.

As with conventional PR, the surface from which light is shot into the environment is chosen by determining which surface will shoot the most light, i.e. which has the most unshot energy. However, for higher order radiosity, the particular basis function of a surface must be chosen. Zatz suggests using the *magnitude* of the radiosity coefficients when determining the particular surface and basis function to shoot from. This is because the radiosity coefficients may be negative, due to the shape of a particular basis function. Figure 5.3 gives the PR algorithm for Galerkin radiosity (using a different form to that given in [68]). Notice the similarity between the pseudocode given for Galerkin radiosity and that given for conventional radiosity in figure 3.7. However, extra loops are required to consider each basis function of surfaces during the shooting and receiving stages.

Previous sections have described how to evaluate the inner products occurring in the pseudocode. Computing K_{ji}^{lk} coefficients requires the kernel function K_{ji} to be evaluated. This function is described in section 4.4. Recall that the kernel function contains the visibility function $V_{ij}(s,t,u,v)$, used to determine whether two surface points are mutually visible. The implementation of this function is described in the following section.

5.5 Ray Tracing NURBS

Evaluating the visibility function $V_{ij}(s,t,u,v)$ requires a form of ray tracing. A visibility ray defined between two known surface points specified by parametric points (s,t) and (u,v). This ray is intersected with all surfaces in the environment. If an intersection occurs with a surface, and the intersection point lies between the two known points, the two known points are occluded. Note that there are two differences between this scheme and conventional ray tracing: the end point of the ray is known and the "nearest hit" is not required. These facts mean that intersection points occurring beyond the visibility ray can be ignored and that as soon as an occluding intersection has been found, the search for a nearer intersection point need not occur. In conventional ray tracing, this procedure is referred to as *shadow testing* [64].

While the simpler case of visibility tracing can reduce the computation time required to intersect a ray with a NURBS surface, computing a ray/NURBS intersection remains compute-intensive. The method implemented in this thesis is based on the subdivision scheme presented in [15]. There are two stages to the algorithm:

5.5.1 Bounding Box Intersection Tests

NURBS surfaces are enclosed in bounding boxes and a visibility ray is intersected with each bounding box [31]. If an intersection occurs, the surface enclosed in the bounding box is subdivided into four sub-surfaces (section 2.5.6). Each sub-surface is enclosed in a bounding box and the ray/box intersection test is repeated. Note that, intersection with a bounding box does not imply intersection with the enclosed surface. However, if a ray misses the bounding box, it will also miss the enclosed surface. Hence, only tests in which the bounding box is missed are terminated at this stage. Subdividing surfaces in this way creates a hierarchy of bounding boxes. This hierarchy of bounding boxes is traversed until a predefined depth is reached. Any leaf nodes which may yield an intersection with their enclosed surface are tested using a further subdivision test based on a scheme given by Veen [61], described below. To reduce the number of surface subdivisions required when performing bounding box tests, the hierarchy of bounding boxes[2] associated with each surface is precomputed to a given depth. Sub-surfaces are only required at the leaf nodes of the hierarchy. All other nodes simply contain the bounding box that would enclose a sub-surface at that level in the hierarchy.

[2] Specifically a quadtree of bounding boxes.

5.5.2 Surface Intersection Test

Having found a set of sub-surfaces (leaf nodes in the bounding box hierarchy) which may yield an intersection, Veen's subdivision test is applied. Veen subdivided NURBS surfaces into Bézier patches before performing any intersection tests. The method implemented in this thesis does not subdivide NURBS surfaces into Bézier patches. Such a subdivision would result in greater memory usage which the use of NURBS was to alleviate. However, Veen's intersection test is still valid.

Veen uses a novel approach to test for ray/surface intersection. The test determines whether the ray intersects the surface's control polygon. An affine transformation is constructed which maps the visibility ray to the x axis, with the ray's origin corresponding to the world coordinate system origin. The transformation is then applied to the surface's control polygon. Testing the ray against the control polygon is now simply a matter of determining whether the transformed control polygon intersects the x axis. Such an intersection occurs if

$$(\exists y_{i,j} = 0 \vee (\exists y_{i,j} < 0 \wedge \exists y_{i,j} > 0)) \wedge (\exists z_{i,j} = 0 \vee (\exists z_{i,j} < 0 \wedge \exists z_{i,j} > 0)), \quad (5.1)$$

where $y_{i,j}$ and $z_{i,j}$ are the y and z ordinates respectively of the control point $\mathbf{P}_{i,j}$ of the transformed NURBS surface. If an intersection with the control polygon occurs, the surface is split[3] and further ray/control polygon tests are made. The test terminates when either no intersection will occur or the u and v parameter ranges are within a specified tolerance. The parametric coordinates of the intersection point (on the transformed surface) are given by the midpoints of the final parameter ranges. To obtain the intersection point on the original surface,

[3]A surface is split in the u and v parameter directions alternately.

the distance from the origin of the point on the transformed surface is used to compute a point along the visibility ray. The required distance along the ray is given by the x ordinate of the intersection point on the transformed surface.

5.6 Rendering

Once the radiosity coefficients B_i^k have been computed, the scene may be rendered. A simple ray tracing scheme has been implemented. The radiosity function at *any* point on a surface i is found by evaluating

$$B_i(s,t) \approx \sum_{k=1}^{n_i} B_i^k N_k(s,t), \tag{5.2}$$

where n_i is the number of basis functions defined on surface i. The ray tracing scheme provides the parametric coordinates of every visible point on each surface. A method similar to that described in section 5.5 is used.

When rendering a scene, the *nearest* intersection point to the virtual camera is required. Unlike visibility tracing, the end point of the ray is unknown. The method implemented is based on that given in [15]. Essentially, a list of intersection records is maintained in order of increasing distance from the camera. An intersection record contains an intersection distance, an indication of whether a bounding box or surface was intersected and a surface identifier. The intersection list is initialised by intersecting each surface's root bounding box. Provided the list is not empty and a bounding box record is found at the start of the list, records are removed one by one. If the removed record is a leaf node, its enclosed surface is tested using the Veen subdivision method described in section 5.5.2. Otherwise the record's child bounding boxes are tested for intersection with the ray. In both cases new records may be added to the list. When a surface intersection record is found at the start of the list, the intersection point for the ray has been determined. If all records are eventually removed from the list then the ray does not intersect any surface.

The progressive refinement solution method, described in section 5.4, allows the scene to be rendered after each iteration of the algorithm. Rendering these partial results allows visual acceptability to be used as a criterion for terminating the algorithm. To avoid recomputing all visible points in the scene using the compute-intensive ray tracing method, such intersection points are cached after they have been determined (during the first iteration of the algorithm). Subsequent iterations render the scene using the cached intersection points. Note also that, unlike conventional ray tracing [24], transmission and reflection rays are not propagated into the environment.

5.7 Shadow Masks

In Zatz's implementation of Galerkin Radiosity [68], shadow masks were used to overcome the inability of polynomial basis function to model discontinuities in the radiosity function caused by shadows. The author's implementation in this thesis does not include shadow masks for reasons stated in section 4.4.3. While shadow masks are simple to implement, they resort back to techniques used in conventional radiosity, abandoning the proposed benefits of higher order radiosity. The implementation of shadow masks would also significantly increase the number of visibility tests performed during a radiosity solution. However, as results in the next chapter will show, some method of representing shadow detail should have been used. Such methods will be discussed in section 6.3.

5.8 Summary

The previous sections have described various aspects of implementing the Galerkin radiosity method. The use of surface functions allows reflectance and emittance to vary across a surface. Numerical quadrature methods are used to implement the linear algebra found in the Galerkin method. The use of a lookup table of quadrature points specified in parametric space, and the specification of a solution degree across each surface, allows automatic and efficient sampling of the radiosity function, without resorting to meshing. A progressive refinement solution method is used to visualise the propagation of light in the environment and a ray tracing scheme is used to both render the scene and evaluate the visibility function in the kernel function. The next chapter presents the results of this implementation and discusses the suitability of the Galerkin method for radiosity and NURBS.

Chapter 6

Results and Conclusions

This chapter presents a set of images generated by the program described in Chapter 5. All images were generated on a clustered HP9000/735crx48z. Conclusions on the work presented in this thesis are made and areas for further work are suggested.

6.1 Results

Table 6.1 shows statistics for the environments that have been rendered. The number of NURBS surfaces in each model is given, together with CPU time used to generate the images. The three times given indicate the CPU time used during the solution of the radiosity equations, the time to perform initial rendering and the overall time to generate the image. Owing to the compute-intensive method of ray tracing the scene during rendering, the overall execution time is high. The number of shots used during the progressive refinement algorithm is given,

Figure Number	No. of NURBS	No. of Shots	CPU Time (m:s)		
			Rad. Solution	Render	Total
6.1	2	5	1:07	24:24	27:18
6.2	2	10	5:04	24:26	37:42
6.3	N/A	N/A	N/A	N/A	N/A
6.4	7	10	7:11	21:42	33:09
6.5	7	10	35:58	22:51	64:58
6.6	7	12	54:50	37:33	100:48
6.7	2	8	26:02	15:07	43:36
6.8	2	1	0:03	7:39	7:42
6.9	2	1	0:03	7:46	7:51

Table 6.1: Statistics for Test Scenes.

with the terminating criteria for the progressive refinement algorithm being that the magnitude of the unshot radiosity in the scene be less than ten percent of the initial unshot radiosity magnitude. This does not apply to figure 6.8 and figure 6.9, which required only one shot for illustration.

Figure 6.1 contains two surfaces—a small light source above the larger ground surface. Only the ground surface is visible. The emitter has a solution degree of four and the receiver has a solution degree of five. The Galerkin method clearly works for simple scenes.

Figure 6.2 is the same as for figure 6.1. However, a NURBS surface function is used to specify a varying emittance function across the light source. Figure 6.3 shows a full view of the emitter used in figure 6.2. Both the emitter and the receiver have a solution degree of six. The increase in solution time occurs because using a varying emittance function requires a higher solution degree across both the receiver and the emitter. Hence more sample points are taken and so more visibility tests are performed.

Figure 6.1: Test scene *flat*. Two planar NURBS surface.

Figure 6.4 shows a traditional radiosity box. The dark corners are due to singularities in the kernel where surfaces meet at corners. The ceiling is lit indirectly and colour bleeding is noticeable. All surfaces have a solution degree of four.

Figure 6.5 uses a modified "floor" surface. The centre control point has been raised, creating a spike in the floor. The shadow discontinuities created by the spike are modelled inaccurately in the basis set and so Gibbs ringing artifacts are present, even though the solution degree of the floor was increased to ten.

Figure 6.6 shows the box with other surfaces deformed by displacing their centre control points. The implementation has computed the radiosity distribution reasonably well for these curved surfaces. The bright patches in the corners of several surface are due to using basis sets of insufficient degree. Each surface has a solution degree of seven, except the light source which has a solution degree of four.

Figure 6.2: Test scene *flat(coloured)*. Two planar NURBS surfaces. The light source uses a NURBS surface function to specify non-constant emittance.

Figure 6.3: Emittance of light source in figure 6.2.

Figure 6.4: Test scene *box*. A traditional radiosity box.

Figure 6.5: Test scene *spike*. The centre control point of the floor in figure 6.4 has been raised, giving a spike in the floor. Shadow discontinuities are badly modelled resulting in Gibbs ringing artifacts.

Figure 6.6: Test scene *deformed*. The centre control points of other surfaces in figure 6.4 have been moved. Bright artifacts are present due to insufficient solution degrees being specified for the curved surfaces.

Figure 6.7: Test scene *depression*. The NURBS surface has a depression and curled corners.

Figure 6.8: Test scene *even*. The NURBS surface has an even parameterisation resulting in correctly distributed sample points.

Figure 6.9: Test scene *uneven*. The NURBS surface has an uneven parameterisation resulting in incorrectly distributed sample points.

Figure 6.7 shows a NURBS surface with a large depression at its centre and with upturned corners. The radiosity across this curved surface has been modelled reasonably well. However, the bottom of the depression is poorly illuminated. Again this is due to Gibbs ringing. The surface is self-shadowing and so light is not transfered correctly between its own basis functions. The concave surface has a solution degree of seven. The light source has solution degree four.

Figure 6.8 and figure 6.9 highlight a problem with using NURBS as the modelling primitive. The Galerkin method uses sample points, defined in parametric space, and given by the quadrature nodes used during inner product evaluation. An uneven parameterisation of a NURBS surface will redistribute the sample points over the NURBS surface. Figure 6.8 shows a simple planar surface with control points defined at each corner, at the centre of each edge and at the centre of the surface. As expected, radiosity is found to be symmetrical over the surface. Figure 6.9 shows the patch with the control points defined on the left and right edges, and the centre of the surface, moved towards the viewer. The far section of the surface then receives more sample points, leaving the nearer section darker. In both images the solution degree of the receiver is four and the solution degree of the emitter is zero.

In addition to Gibbs ringing, poor parameterisation accounts for the inaccurate shadows found in earlier images. This represents a significant drawback when using NURBS as the modelling primitive. An even parameterisation of the surface must be used to ensure correct distribution of sample points across surfaces.

6.2 Conclusions

This thesis has examined the use of curved surfaces in the radiosity method. Parametric shape representation was reviewed, with NURBS being chosen as the modelling primitive.

The rendering equation was presented, allowing a physically based model of the transport of light between diffuse surfaces to be described. The remainder of the thesis was concerned with methods for solving the rendering equation. Chapter 3 examined classical methods, which subdivided surfaces into patches and assumed radiosity to be constant across each patch. These methods are still in use today and are capable of producing stunning images (see for example [23]). However, using such methods with curved surfaces results in extremely high memory costs when surfaces are tessellated.

Chapter 4 considered radiosity methods capable of using curved surfaces directly, avoiding some of the approximations used in the earlier methods. While these methods were capable of incorporating curved surfaces into the radiosity method, they invariably made the Constant Radiosity Assumption [59], introducing error into their solution and using interpolation, inappropriately, during rendering.

To overcome the problems of assuming constant radiosity, the higher order Galerkin method was presented. This allowed meshing of a surface to be avoided by projecting the radiosity function into a basis set of higher order polynomials and using predefined quadrature points to sample the radiosity function. Using a parametric form of the radiosity equation allowed NURBS surfaces to be used as the modelling primitive.

The author's implementation of the Galerkin radiosity method, based on Zatz's work [68], was discussed. Results showed that the Galerkin method provided a method of using NURBS as the modelling primitive for simple environments. However, there is a significant drawback to the method. Basis functions with global support across a surface cannot accurately represent discontinuities in the radiosity function caused by shadows. Zatz used shadow masks to over come this problem. However, shadow masks resort back to previously rejected methods. It is interesting to note that such a technique has not been adopted by the mathematics community when solving singular Fredholm integral equations (the rendering equation is in this class of equation). It seems this "hack" is used only in a computer graphics context. The inability to represent shadow discontinuities essentially means that this form of Galerkin radiosity cannot be used for complex scenes.

6.3 Future Work

The shadow discontinuity problems in the Galerkin method are due to the basis functions, not the Galerkin method *per se*. It is the global support of the basis functions which precludes the accurate representation of shadow detail. The Galerkin method has been applied by Troutman and Max [60] to a discontinuity meshed environment. However, this form of meshing relies on a polygonal environment. Alternative basis functions to Legendre polynomials should be investigated.

One possible choice of basis functions are the B-Spline basis functions, described in section 2.4.1. B-Spline basis functions allow discontinuities to be modelled and so, provided discontinuities in the radiosity function can be located, a suitable knot vector could be chosen to model a discontinuous radiosity function across a surface. However, B-Spline basis functions have local support (section 2.4.2) and so several basis functions are defined for a particular degree. As with the method presented in this thesis, each surface may also have basis functions of various degrees associated with it. Hence the total number of basis functions defined for a surface could significantly increase. However, the generality of the B-Spline basis functions may allow a basis set of smaller degree to be specified. B-Spline basis functions are not orthonormal (although they are linearly independent [48]). Hence the Gram-Schmidt orthonormalisation process [20] must be used when evaluating the basis functions. Using B-Spline basis functions may prove to be more compute-intensive than using Legendre polynomials. However, shadow detail should be captured more accurately in basis sets constructed using B-splines.

Wavelet radiosity [28][53] has also been proposed as a solution to the shadow discontinuity problem. Wavelet radiosity uses polynomial basis functions which can interact hierarchically. Wavelets [10] are a class of basis functions with these properties. Hence the advantages of hierarchical radiosity (HR) are combined with the accuracy of higher order Galerkin radiosity. Using wavelet bases also allows fast solution methods through a sparse representation of the kernel matrix (section 4.4). This is achieved by considering near-zero entries in the kernel matrix to be zero (Gortler *et. al.* [28] liken this to performing lossy compression on the kernel matrix).

Wavelet radiosity has been extended to wavelet *radiance* by Christensen *et. al.* [9]. In their work they describe a method which supports curved surfaces. Surfaces are not assumed to be ideal diffuse reflectors, allowing glossy reflections to be modelled. As stated in [9], the method could easily be extended to handle NURBS. An implementation of this method would provide a more successful method of using NURBS as the modelling primitive in radiosity.

Appendix A

Example MSDL File

An example MSDL file is given below. Two surfaces are defined, one an emitter and one a receiver. The emittance and reflectance of the emitter are both specified using surface functions. Note that the reflectance is specified as being zero using a constant surface function.

```
/* Camera specification */
view
  from          0.0 0.0 2.5
  towards       0.0 -0.8 0.1
  up            0.0 2.4 -0.8
  window        0.6 -0.6 0.6 -0.6
end

/* Emitter */
nurbs
```

```
 name Light
 order   2 2
 cpts    3 3
 knots   { 0 0 0.5  1 1 }
         { 0 0 0.5  1 1 }
 /* Control points */
 0.5 0.0 0.5 1
 0.0 0.0 0.5 1
-0.5 0.0 0.5 1


 0.5 0.0 0.0 1
 0.0 0.0 0.0 1
-0.5 0.0 0.0 1


 0.5 0.0 -0.5 1
 0.0 0.0 -0.5 1
-0.5 0.0 -0.5 1
 verbatim      /* Verbatim block giving non-MSDL text */
   solution 5  /* Solution degree for surface  */
   sides 1     /* Number of sides */
   function    /* Emittance and reflectance are functions */
   emittance   /* First spectral sample surface function  */
     order 2 2 /* Very simple surface function */
     cvals 2 2
     knots { 0 0 1 1 }
```

```
      { 0 0 1 1 }
  1.0 0.0
  0.5 0.0
done
emittance/* Second spectral sample surface function  */
  order 2 2
  cvals 2 2
  knots { 0 0 1 1 }
        { 0 0 1 1 }
  0.0 0.0
  0.5 1.0
done
emittance/* Third spectral sample surface function  */
  order 2 2
  cvals 2 2
  knots { 0 0 1 1 }
        { 0 0 1 1 }
  0.0 1.0
  0.0 0.5
done

drc/* Reflectance surface function */
  order 2 2
  cvals 2 2
  knots { 0 0 1 1 }
```

```
            { 0 0 1 1 }
      0.0 0.0
      0.0 0.0
    done
    drc
      order 2 2
      cvals 2 2
      knots { 0 0 1 1 }
            { 0 0 1 1 }
      0.0 0.0
      0.0 0.0
    done
    drc
      order 2 2
      cvals 2 2
      knots { 0 0 1 1 }
            { 0 0 1 1 }
      0.0 0.0
      0.0 0.0
    done
  end /* Verbatim */
  rotate y 45
  scale 4 1 4
  shift 0 -1.0 0
end /* Light */
```

```
/* Receiver */

nurbs

  name Floor

  /* Get NURBS definition from another file */

  #include "lumpy.msdl"

  verbatim

    solution 6

    sides 1

    /* Emittance and reflectance are both constant */

    const

    emittance 0.0 0.0 0.0

    drc 0.4 0.4 0.9

  end

  rotate y 45

  scale 4 1 4

  shift 0 -1.0 0

end /* Floor */
```

Bibliography

[1] S. S. Abi-Ezzi and L. A. Shirman. Tessellation of curved surfaces under highly varying transformations. In F. H. Post and W. Barth, editors, *Proceedings, Eurographics '91*, pages 385–397, Amsterdam, North Holland, September 1991. Eurographics.

[2] J. M. Airey, J. H. Rohlf, and F. P. Brooks. Towards image realism with interactive update rates in complex virtual building environments. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):41–50, March 1990.

[3] A. Appel. Some techniques for machine rendering of solids. In *AFIPS Conference Proceedings*, volume 32, pages 37–45, 1968.

[4] K. E. Atkinson. *A Survey of Numerical Methods for the Solution of Fredholm Integral Equations of the Second Kind.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1976.

[5] H. Bao and Q. Peng. A progressive radiosity algorithm for scenes containing curved surfaces. *Computer Graphics Forum (Eurographics '93)*, 12(3):C399–C408, September 1993.

[6] D. R. Baum, H. E. Rushmeier, and J. M. Winget. Improving radiosity solutions through the use of analytically determined form-factors. *Computer Graphics (ACM SIGGRAPH '89 Proceedings)*, 23(3):325–334, July 1989.

[7] C. De Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[8] I. Carlbom and J. Paciorek. Planar geometric projections and viewing transformation. *Computing Surveys*, 10(4):465–502, December 1978.

[9] P. H. Christensen, E. J. Stollnitz, D. H. Salesin, and T. D. DeRose. Wavelet radiance. In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 287–302, Darmstadt, Germany, June 1994.

[10] C. K. Chui. An introduction to wavelets. In *Wavelet Analysis and its Applications*, volume 1. Academic Press Inc., 1992.

[11] M. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, 22(4):75–84, August 1988.

[12] M. Cohen and D. P. Greenberg. The hemi-cube: A radiosity solution for complex environments. *Computer Graphics (ACM SIGGRAPH '85 Proceedings)*, 19(3):31–40, July 1985.

[13] M. Cohen, D. P. Greenberg, D. S. Immel, and P. J. Brock. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6(3):26–35, March 1986.

[14] M. F. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993.

[15] G. M. Cook. Parallel ray tracing of non uniform rational b-spline surfaces. Master's thesis, University of Manchester, Manchester, UK, 1992.

[16] M. Daniel and J. C. Daubisse. The numerical problem of using Bézier curves and surfaces in the power basis. *Computer Aided Geometric Design*, 6:121–128, 1989.

[17] L. M. Delves and J. L. Mohamed. *Computational Methods for Integral Equations*. Cambridge University Press, Cambridge, 1985.

[18] E. R. G. Eckert and R. M. Drake. *Heat and Mass Transfer*. McGraw-Hill, New York, NY, 1959.

[19] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, third edition, 1993.

[20] B. H. Flowers. *An Introduction to Numerical Methods in C++*. Clarendon Press, Oxford, 1995.

[21] N. Gatenby. Radiosity: A proposed alternative to the hemi-cube algorithm. Master's thesis, University of Manchester, Manchester, 1991.

[22] N. Gatenby. *Optimising Discontinuity Meshing Radiosity*. Ph.D. thesis, University of Manchester, Manchester, 1995.

[23] S. Gibson. Efficient radiosity for complex environments. Master's thesis, Department of Computer Science, University of Manchester, Manchester, 1995.

[24] A. S. Glassner. *An Introduction to Ray Tracing*. Academic Press, 1989.

[25] A. S. Glassner. *Principles of Digital Image Synthesis.* Morgan Kaufmann, San Francisco, CA, 1995.

[26] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modelling the interaction of light between diffuse surfaces. *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, 18(3):213–222, July 1984.

[27] S. J. Gortler, M. F. Cohen, and P. Slusallek. Radiosity and relaxation methods: Progressive refinement is southwell relaxation. Technical Report CS-TR-408-93, Department of Computer Science, Princeton University, Princeton, NJ, February 1993.

[28] S. J. Gortler, P. Schroder, M. F. Cohen, and P. Hanrahan. Wavelet radiosity. *Computer Graphics Proceedings, Annual Conference Series (ACM SIGGRAPH '93 Proceedings)*, 27(3):221–230, 1993.

[29] H. Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, 20(6):623–629, June 1971.

[30] H. Gouraud. Illumination for computer generated pictures. *Communications of the ACM*, 18(60):311–317, 1971.

[31] E. Haines. Essential ray tracing algorithms. In Andrew S. Glassner, editor, *An Introduction To Ray Tracing*, pages 33–77. Academic Press, 1989.

[32] P. Hanrahan and D. Salzman. A rapid hierarchical radiosity algorithm for unoccluded environments. Technical Report CS-TR-281-90, Department of Computer Science, Princeton University, Princeton, NJ, August 1990.

[33] P. Hanrahan and D. Salzman. A rapid hierarchical radiosity algorithm for unoccluded environments. In K. Bouatouch and C. Bouville, editors, *Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, pages 151–171, Elsevier, Amsterdam, 1990.

[34] P. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, 24(4):145–154, August 1990.

[35] P. Heckbert and J. M. Winget. Finite element methods for global illumination. Technical Report UCB/CSD 91/643, University of California at Berkeley, Berkeley, CA, 1991.

[36] H. C. Hottel and A. F. Sarofim. *Radiative Transfer.* McGraw Hill, New York, 1967.

[37] J. T. Kajiya. The rendering equation. *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, 20(4):143–150, August 1986.

[38] A. J. F. Kok, C. Yilmaz, and L. H. J. Bierens. A two-pass radiosity method for Bézier patches. In K. Bouatouch and C. Bouville, editors, *Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, pages 117–126, Elsevier, Amsterdam, 1990.

[39] F. Lin. *NURBS in CAD and Computer Graphics.* PhD thesis, The University of Manchester, January 1996.

[40] W. H. McAdams. *Heat Transmission.* McGraw-Hill, New York, NY, 1954.

[41] T. Nishita and E. Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. *Computer*

Graphics (ACM SIGGRAPH '85 Proceedings), 19(3):23–30, July 1985.

[42] T. Nishita and E. Nakamae. A new radiosity approach using area sampling for parametric patches. *Computer Graphics Forum (Eurographics '93)*, 12(3):C385–C398, September 1993.

[43] M. Y. Numan and G. R. Moore. Form factors: The problem of partial obstruction. Technical report, The Martin Centre for Architectural and Urban Studies, Nov 1982.

[44] B. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[45] L. Piegl. On nurbs: A survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, January 1991.

[46] L. Piegl, editor. *Fundamental Developments of Computer Aided Geometric Modelling*. Academic Press, London, 1993.

[47] L. Piegl and W. Tiller. *Nlib: An Object-Based NURBS Library*. Geomware Inc., 3036 Ridgetop Road, Tyler, TX 75703.

[48] L. Piegl and W. Tiller. *The NURBS Book*. Springer-Verlag, 1994.

[49] W. H. Press. *Numerical Recipes in C : the Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

[50] M. Preston, N. Gatenby, and W. T. Hewitt. *The Manchester Scene Description Language (MSDL) Version 1.1*. Computer Graphics Unit, Manchester Computing, University of Manchester, Manchester, September 1995.

[51] C. Puech, F. Sillion, and C. Vedel. Improving interaction with radiosity-based lighting simulation programs. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):51–57, March 1990.

[52] D. F. Rogers and J. A. Adams. *Mathematical Elements for Computer Graphics.* McGraw-Hill, second edition, 1990.

[53] P. Schroder, S. J. Gortler, M. F. Cohen, and P. Hanrahan. Wavelet projections for radiosity. In *Proceedings of the Fourth Eurographics Workshop on Rendering*, pages 105–114, Paris, France, June 1993.

[54] P. Schröder and P. Hanrahan. On the form factor between two polygons. *Computer Graphics Proceedings, Annual Conference Series, 1993 (ACM SIGGRAPH '93 Proceedings)*, 27(3):163–164, August 1993.

[55] F. Sillion and C. Puech. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics (ACM SIGGRAPH '89 Proceedings)*, 23(3):335–344, July 1989.

[56] F. Sillion and C. Puech. *Radiosity and Global Illumination.* Morgan Kaufmann, San Francisco, CA, 1994.

[57] A. H. Stroud. *Approximate Calculation of Multiple Integrals.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[58] A. H. Stroud and D. Secrest. *Gaussian Quadrature Formulas.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1966.

[59] F. Tampieri and D. Lischinski. The constant radiosity assumption syndrome. In *Proceedings of the Second Eurographics Workshop on Rendering*, Barcelona, Spain, May 1991.

[60] R. Troutman and N. L. Max. Radiosity algorithms using higher order finite element methods. *Computer Graphics Proceedings, Annual Conference Series (ACM SIGGRAPH '93 Proceedings)*, 27(3):209–212, 1993.

[61] A. van der Veen. Intersection test for nurbs. In A.J. van der Veen and M.A.J. Bloemendaal, editors, *IEEE Symposium on Computer Architecture & Real Time Graphics*, pages 101–114, Delft, The Netherlands, 1989.

[62] C. Vedel. Improved storage and reconstruction of light intensities on surfaces. In *Proceedings of the Third Eurographics Workshop on Rendering*, pages 113–121, Bristol, May 1992.

[63] J. R. Wallace, K. A. Elmquist, and E. A. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics (ACM SIGGRAPH '89 Proceedings)*, 23(3):315–324, July 1989.

[64] A. Watt. *Fundamentals of Three-Dimensional Computer Graphics*. Addison-Wesley, Wokingham, 1989.

[65] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques: Theory and Practice*. Addison-Wesley, Wokingham, 1992.

[66] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6), June 1980.

[67] N. Young. *An Introduction to Hilbert Space*. Cambridge University Press, New York, 1988.

[68] H. R. Zatz. Galerkin radiosity: A higher order solution method for global illumination. Master's thesis, Cornell University, August 1992.

[69] H. R. Zatz. Galerkin radiosity: A higher order solution method for global illumination. *Computer Graphics Proceedings, Annual Conference Series (ACM SIGGRAPH '93 Proceedings)*, 27(3):213–220, 1993.