A Projective Approach to Computer-Aided Drawing

by

Osama S. Tolba
Master of Landscape Architecture
University of Pennsylvania, 1992

Submitted to the Department of Architecture
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Architecture: Design and Computation
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2001

Author .....
Department of Architecture
May 4, 2001

Certified by .....
Julie Dorsey
Associate Professor of Computer Science and Engineering and Architecture
Thesis Supervisor

Certified by .....
Leonard McMillan
Associate Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Stanford Anderson
Chairman, Department Committee on Graduate Students

Dissertation Committee

- Julie Dorsey

  Associate Professor of Computer Science and Engineering and Architecture

- Leonard McMillan

  Associate Professor of Computer Science and Engineering

- William Porter

  Muriel and Norman Leventhal Professor of Architecture and Planning

- Wellington Reiter

  Associate Professor of the Practice of Architecture

A Projective Approach to Computer-Aided Drawing

by

Osama S. Tolba

Submitted to the Department of Architecture
on May 4, 2001, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Architecture: Design and Computation

Abstract

I present a novel drawing system for composing and rendering perspective scenes. The proposed approach uses a projective two-dimensional representation for primitives rather than a conventional three-dimensional description. This representation is based on points that lie on the surface of a unit sphere centered at the viewpoint. It allows drawings to be composed with the same ease as traditional illustrations, while providing many of the advantages of a three-dimensional model. I describe a range of user-interface tools and interaction techniques that give the drawing system its three-dimensional-like capabilities. The system provides vanishing point guides and perspective grids to aid in drawing freehand strokes and composing perspective scenes. The system also has tools for intuitive navigation of a virtual camera, as well as methods for manipulating drawn primitives so that they appear to undergo three-dimensional translations and rotations. The new representation also supports automatic shading of primitives using either realistic or non-photorealistic styles. My system supports drawing and shading of extrusion surfaces with automatic hidden surface removal and emphasized silhouettes. Casting shadows from an infinite light source is also possible with minimal user intervention. I describe a method for aligning a sketch drawn outside the system using its vanishing points, allowing the integration of computer sketching and freehand sketching on paper in an iterative manner. Photographs and scanned drawings are applied to drawing primitives using conventional texture-mapping techniques, thereby enriching drawings and providing another way of incorporating hand-drawn images. I demonstrate the system with a variety of drawings.

Thesis Supervisor: Julie Dorsey
Title: Associate Professor of Computer Science and Engineering and Architecture

Thesis Supervisor: Leonard McMillan
Title: Associate Professor of Computer Science and Engineering

Acknowledgments

I would like to thank the following:

- My advisors, Julie Dorsey and Leonard McMillan who, not only encouraged and guided me, but also taught me a great deal about computer graphics and computer science.

- My committee members, William Porter and Wellington Reiter for their valuable comments.

- Seth Teller, co-founder of M.I.T.'s Computer Graphics Group, for being a wonderful educator.

- Fellow students in the research group, especially Max Chen, Aparna Das, and Stephen Duck who used my drawing system and provided various drawings and feedback on the system's usability.

- My wife Nadia for her love, companionship, and endless support, and my children Mohamed and Nissma, who bring enormous joy to my life.

Finally, I dedicate this dissertation to my mother Sohad Khalil for her haven and prayers, and to the memory of my father Salah Tolba, who was the original cultivator of my love for mathematics and geometry.

Contents

List of Figures

Chapter 1

Introduction

The advent of computer graphics has greatly influenced many aspects of architectural design. Construction drawings, in the form of plans, sections, elevations, and details, are seldom drawn by hand today. In addition, hand-crafted physical models, traditionally used for client presentations, have been largely replaced with three-dimensional computer models and walk-throughs. Perspective drawing, which was once an important technique for exploring and presenting design ideas, is virtually obsolete due to the speed and flexibility of today's Computer-Aided Design and Drafting (CADD) systems.

Perspective drawings have appeal because they convey three-dimensional shape information on a two-dimensional surface, such as paper or computer screens (see Figure 1-1). The basic principles of perspective where developed in the Renaissance and became widely used for art and design [9]. Perspective drawings remain in use for communicating design ideas, and their techniques are still taught in art and design schools.

Traditional perspective drawings are difficult to construct. Only a skilled illustrator can make a drawing with correct proportions. Furthermore, many construction lines are required to achieve this proportionality, making the process laborious. Numerous manuals, such as [13], have been written to teach artists various sets of constructions. Another shortcoming of traditional perspective is that the views are static, which reduces their three-dimensional impression. Proper shadow construction

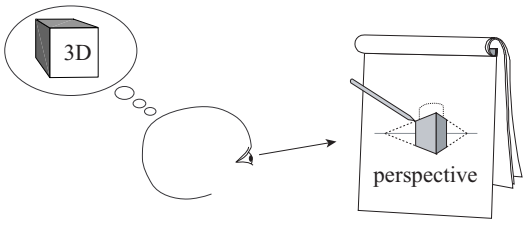


Figure 1-1: Many of the benefits of perspective are due to its ability to convey three-dimensional scenes utilizing nothing more than two-dimensional media.

and shading are also time-consuming. Finally, like all drawings on paper, they are difficult to edit or reuse.

My goal is to provide interactive techniques to support perspective drawing. This problem has been largely neglected in two-dimensional computer graphics. Almost all current two-dimensional graphics systems use drawing primitives that are represented with *Euclidean* two-dimensional points. The process of constructing a perspective drawing with these systems is nearly as tedious as with traditional media.

This neglect is due, in part, to the immense emphasis on research in three-dimensional graphics despite some known limitations. While three-dimensional models are powerful representations, they tend to convey rigid geometry and they can be very cumbersome to build. The rendering in Figure 1-2 is generated from a three-dimensional computer model that was difficult to construct, especially the vaulted ceilings and small triangular openings above the arch. In addition to such geometric complexities, a major difficulty with three-dimensional models is the fact that they are constructed using two-dimensional computer interfaces that are one dimension lower than the models. This usually forces the user to specify the coordinates in more than one view, which makes the process tedious. Another shortcoming of CADD systems is that they usually force the designer to input precise geometry and dimensions, which prohibits their use in the early stages of design where concepts are often vague and ambiguous.

On the other hand, since perspective is constructed on a two-dimensional medium, such as paper or canvas, it lends itself to creative expression and fluid freehand strokes.

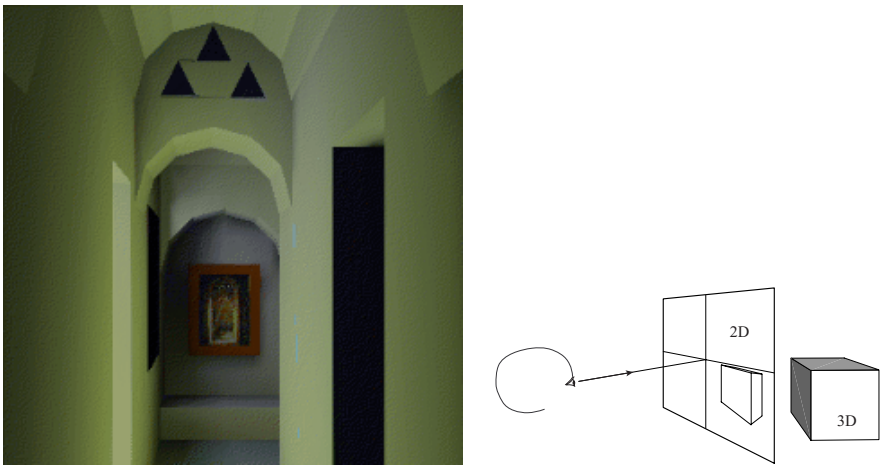


Figure 1-2: Rendering from a traditional three-dimensional computer graphics system.

The sequence in Figure 1-3 is part of an early design stage study. It shows dramatized vistas using twisted strokes and soft, blended colors. The methodical drawing in Figure 1-4 includes realistic-looking shade trees, which would be almost impossible to model in three dimensions.

Today's drawing systems, such as Adobe Illustrator, utilize Euclidean points represented by Cartesian (x, y) coordinates. Constructing a perspective drawing with these systems is almost as tedious as with manual drawing. The views they depict also remain static as with traditional media.

I have developed a perspective drawing system that overcomes many of the limitations of traditional perspective drawing and current two-dimensional computer graphics systems. My system retains the ease-of-use of a two-dimensional drawing, but its projective representation provides additional *three-dimensional-like* functionality. This tool is intended for applications that may not always require actual three-dimensional modeling, such as conceptual design, technical illustration, graphic design, and architectural rendering. In many cases, these applications strive to generate a single perspective view, or a set of views sharing a common viewpoint.

The main contribution of my approach is the use of *projective* two-dimensional points to compose various renderings of a scene and provide capabilities that are

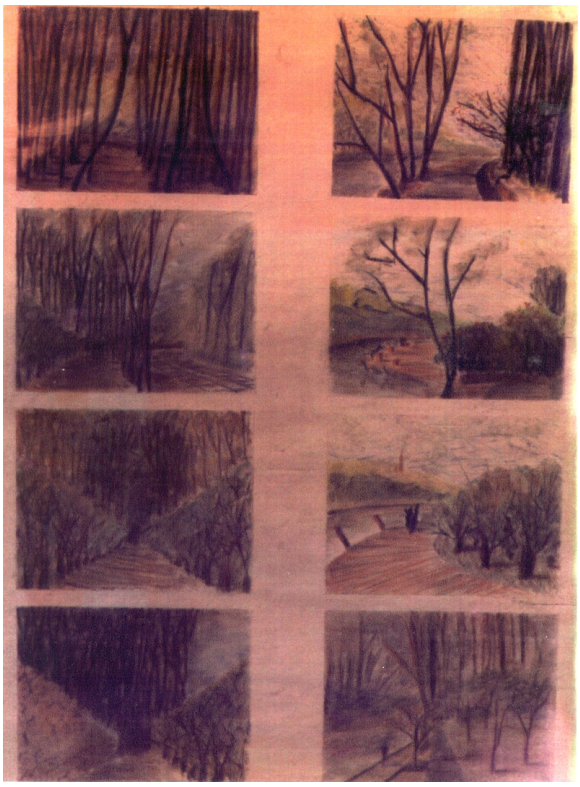


Figure 1-3: A series of perspective sketches made during the conceptual design stage.

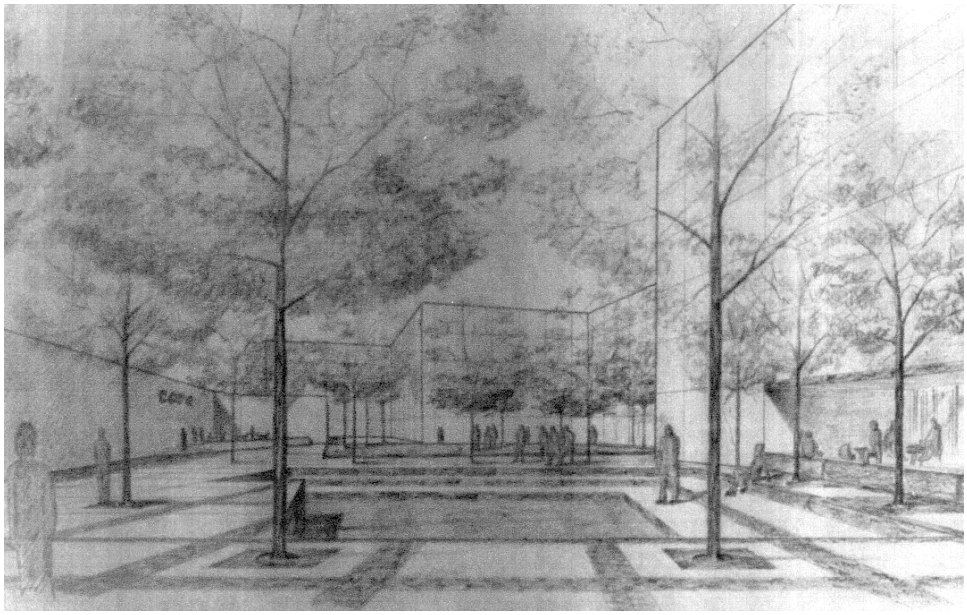


Figure 1-4: A traditional hand-drawn perspective view made for design communication.

generally thought to require three-dimensional models. For example, my system supports immersive viewing, pseudo-three-dimensional primitive manipulation, scene illumination and shading, and automatic shadow construction. In addition, shape modeling operations, such as extrusion, can also be performed using projective two-dimensional points.

In this chapter, I will propose an alternative representation for perspective drawings and discuss its benefits. I will also review previous work that used a projective representation or merely emphasized *drawings* as the main input or output medium.

1.1 Thesis Statement and Contributions

This research presents an alternative drawing paradigm, in which the underlying representation is a collection of *projective* rather than *Euclidean* two-dimensional primitives. This representation is consistent with the notion that perspective drawings are images of three-dimensional worlds under central projection. The illusion of looking at a three-dimensional world is sustained by means of projective mappings of the points from one image to another, and through the simulation of the effects of virtual light sources.

The main thesis of this research is that

> *Perspective drawings can be supported in the computer medium with projective two-dimensional geometry, allowing for freehand drawing as well as pseudo-three-dimensional interaction (viewing), manipulation (editing), and illumination.*

Several problems must be addressed in order to support perspective drawing with the computer. First, we must find a computational representation for the projective primitives. Second, in order to create a usable two-dimensional interface, we need a set of intuitive projective mapping operations whose parameters can be specified by the user in a natural and easy way. Finally, for added visual impact, we seek representations of light sources and primitive attributes that allow for shading and shadow projections.

This research contributes to the fields of architecture and computer graphics the following achievements:

- A new representation for computer drawings based on projective geometry.
- A user interface tailored for the construction, navigation, and manipulation of perspective drawings that encompasses and goes beyond traditional techniques.
- Methods for shading of drawing primitives and determination of shadows using directional light sources.

Furthermore, this research addresses the integration of hand- and computer-drawn perspective images, thereby enabling a wide range of input media and devices.

In this thesis, I hope to show that representing perspective drawings in the computer with projective geometry provides the following benefits:

- A virtual immersive three-dimensional camera.
- Vanishing points and perspective grids used as drawing guides.
- Support for freehand-style as well as methodically drafted views.
- Pseudo-three-dimensional manipulation of drawing primitives.
- Modeling and manipulation of complex shapes, with proper visibility computations.
- Shadow projection and shading of primitives using directional light sources.
- Shading techniques that mimic the traditional hand-drawn look and enhance the appeal of the drawings.
- Alignment of views drawn outside the computer.
- Flexible texture mapping techniques that support planar and non-planar images.

1.2 Related Work

In this section I will review some previous work that has inspired my approach. In addition, I will describe an alternative approach that attempts to generate three-dimensional models using interfaces that mimic traditional freehand drawing.

Image-based Rendering

Projective representations underly all panoramic image-based rendering (IBR) systems. For example, "QuickTime VR" represents environments with cylindrical panoramas and synthesizes novel perspective views by providing an interface for panning, tilting, and zooming [2], without relying on three-dimensional geometry. IBR systems typically facilitate *navigation* and visualization of a static scene. In my approach, I provide controls for *composing* and *editing* illustrations.

Commercial Drawing Programs

Today's commercial two-dimensional drawing and illustration programs allow the user to input freehand strokes and geometric primitives, such as lines and rectangles, specified in the Euclidean plane. I extend these notions to the *projective plane*, thereby allowing for more general modeling primitives, such as straight lines that adhere to a chosen vanishing point and quadrangles that respect two vanishing points. I also extend the use of *regular* grids, common in today's systems, to *perspective* grids, which also conform to one or more vanishing points.

Non-photorealistic Rendering

Non-photorealistic rendering (NPR) techniques apply a "hand-drawn" look to photographs and three-dimensional renderings by simulating many conventional artistic methods. For example, when mimicking pen-and-ink styles, NPR uses hatching or stippling (a collection of short strokes) as a means to convey tonal variation [21, 26, 15]. Another NPR technique is the use of silhouettes to emphasize shape [25, 16]. My work adopts silhouetting and selected pen-and-ink styles for rendering

shaded perspective drawings automatically, although the actual rendering style is not the focus of my work.

Sketching Interfaces

An active area of research is the development of sketching interfaces for three-dimensional modeling [3, 18, 34]. These approaches acknowledge the difficulty of using standard interfaces to build three-dimensional models. Their main premise is that three-dimensional shape can be inferred from freehand strokes that follow a certain syntax, thereby allowing models to be generated very quickly. However, since the models have higher dimensions than the interface, these systems typically make assumptions about the model shapes, or enforce constraints upon the user, such as assuming that the initial point of a stroke lies on a pre-existing three-dimensional plane. In my work, I do not infer three-dimensional geometry, rather I make two-dimensional drawings that *appear* as if they were three-dimensional.

1.3 Thesis Overview

Chapter 2 gives a review of traditional perspective drawing techniques and how they are formalized in projective geometry. Chapter 3 introduces the perspective drawing system, including its projective camera and basic drawing guides and primitives. In Chapter 4, I describe how the drawing primitives are manipulated in a way that preserves their three-dimensional illusion. Chapter 5 introduces new primitives that are aggregates of the basic planar primitives and mimic specific categories of three-dimensional shapes. Shading of perspective drawings and casting shadows is explained in Chapter 6. The system also supports photographs and integrates well with conventional drawing media. Chapter 7 covers these capabilities. Finally, I conclude with discussion of the benefits and limitations of my approach and suggest areas of future work.

Chapter 2

Background

The work I present in this thesis is based on two main interrelated topics: traditional perspective and projective geometry. Projective geometry gives concrete formulation of perspective views, their formation, and structure. In this chapter I present these two topics. First, I introduce some concepts in traditional perspective. My goal is to orient the reader with regard to this thesis rather than provide a primer on perspective drawing. Second, I present the basics of projective geometry, with emphasis on central projection of three-dimensional space and the resultant projective two-dimensional space. Two models of projective space are discussed: the straight, corresponding to planar perspective, and the spherical, which is the preferred model for this thesis.

2.1 Perspective Drafting Techniques

Traditionally, accurate perspective views are constructed from two or more orthographic views (plans, sections, and elevations) containing three-dimensional information. The process involves the intersection of vision rays from two such views (see Figure 2-1-a). The illustration of an entire scene using this process is tedious, involving numerous construction lines. As an alternative, skilled artists use special techniques and shortcuts to simplify and accelerate this process. The simplest and most commonly used technique is the vanishing point (see Figure 2-1-b). Such shortcuts aim to reduce the number of rays generated from orthographic views. For example,

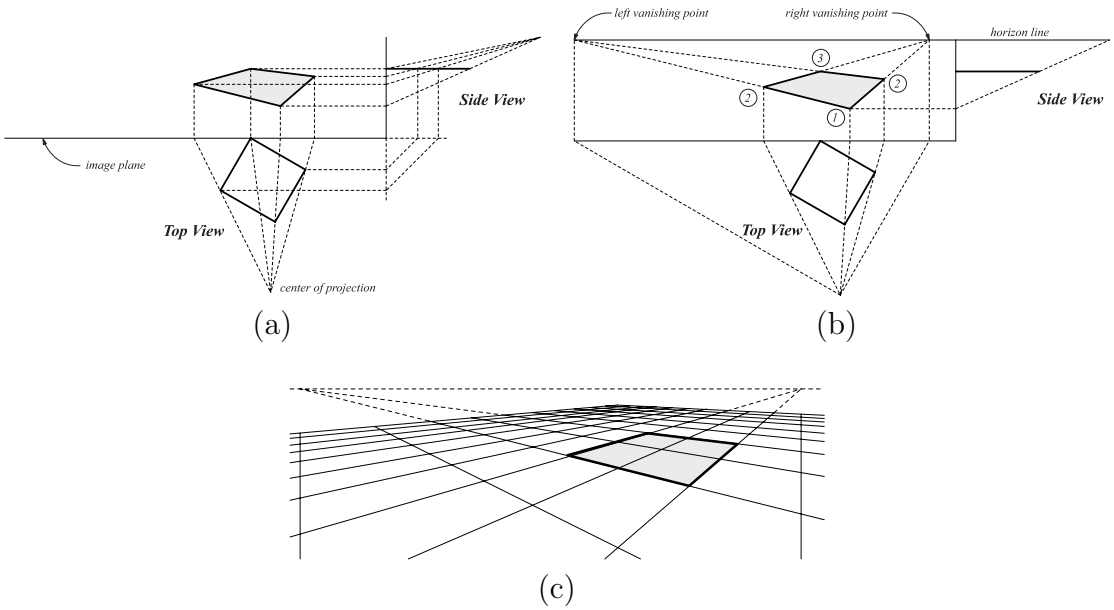


Figure 2-1: A square is drawn using different perspective drafting techniques: (a) ray intersection, (b) vanishing points, and (c) pre-formatted grids. Note how the use of vanishing points greatly reduces the number of construction lines, and the use of grids eliminates them completely.

artists often do away with orthographic views and rely on pre-formatted grids (see Figure 2-1-c). References and guides on perspective techniques are too numerous to list here. However, some useful starting points are [9, 7, 33, 13, 10].

Vanishing points are the most fundamental tool in perspective. In fact, many of the techniques described in this section, including grids, measuring points, and extrusion, use special kinds of vanishing points geared toward specific tasks. Although perspective line drawings can give an impression of three-dimensional structure due to foreshortening and occlusion relationships, a scene that includes shading and shadows is much more realistic. Shading gives surfaces additional texture and relative orientation with regards to a common light source, while shadows convey information about how far objects are from each other. I discuss shading and shadows at the end of this section.

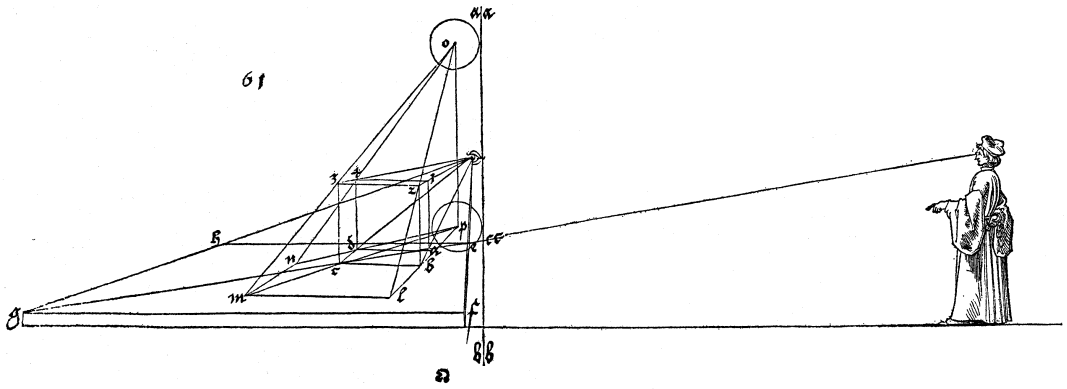
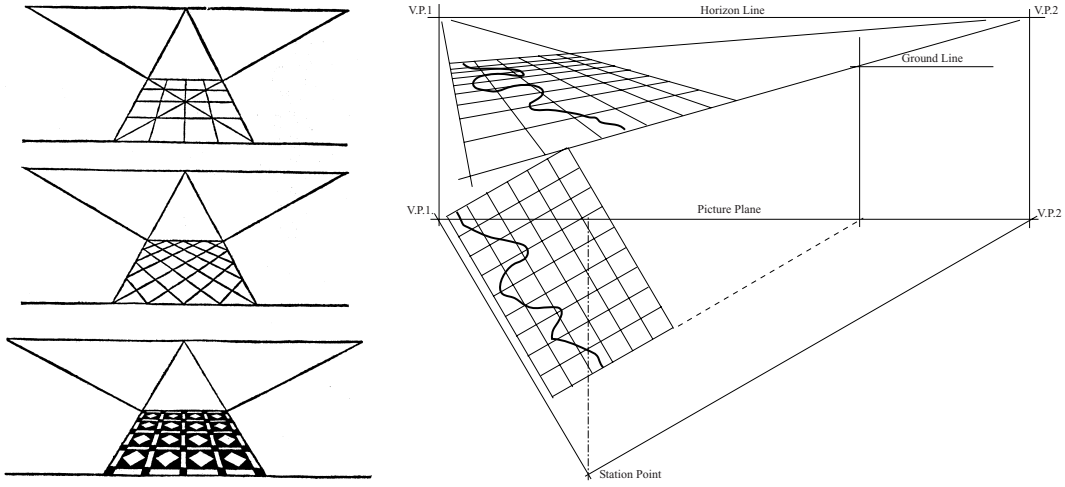


Figure 2-2: Dürer's diagram 61 on perspective.

Vanishing points. A set of receding parallel lines in a three-dimensional world appear to converge at a common point when viewed in perspective. Such a point, originally called the point of convergence [20], is widely known as a *vanishing point*. Scenes that contain man-made or architectural objects usually contain edges that belong to three mutually orthogonal directions, each of which results in a unique vanishing point. When any of these directions is parallel to the viewing plane, the corresponding lines no longer appear to converge. The view is thereby classified as a *two-point* perspective view. The so-called *single-point* perspective takes place when two of the directions in the scene are parallel to the viewing plane (see, for example, the view in Figure 2-2). The third remaining direction aligns with the viewer's gaze direction. Often, artists will use single-point and two-point views because they are easier to construct than the more general three-point views.



(a) (b)

Figure 2-3: Viator's perspective construction of a tiled floor (a), and a grid superimposed on plan to simplify the drawing of a map in perspective (b).

Grids. In addition to vanishing points, which help with directions, artists need tools to determine lengths of line segments and relative positions within foreshortened planar surfaces. One such tool is the perspective grid, whose construction methods rely on auxiliary vanishing points that represent the direction of the grid's diagonals. Often, grids are evident in the finished views in the form of tiled floors or ceilings (see Figure 2-3-a) .

Grids can also be used to transfer free-form or complicated shapes from their orthographic representation. The process is similar to the one used for transferring drawings from small sketches into large murals, except that the target includes a perspective grid rather than a Euclidean one. Additional height or extrusion information is often added later (see Figure 2-3-b) ([13] pp. 29–31).

Prior to the common use of computer graphics, some artists relied on ready-made perspective grids. The grids were created for a variety of viewing position and published in drafting manuals. However, many artists feel that grids are inflexible and of limited use due to their static nature.

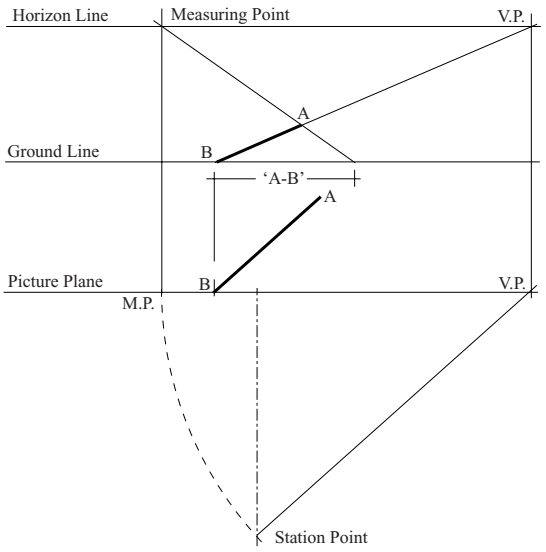


Figure 2-4: Rotating a line segment in perspective by the measuring-point method.

Measuring points. So-called "measuring points" are particular types of vanishing points that facilitate length measurements to be performed directly in the perspective view rather than transferring them from orthographic views by means of visual rays. While regular vanishing points typically represent directions of actual object sides, measuring points include directions of construction lines—lines that do not appear in the final drawing. The direction of the diagonals of a grid are one such example.

In one particularly interesting technique, the length of a line segment is measured on the viewing plane. Then a special measuring point is used to transfer the measurements (points) onto the desired perspective line, which is oblique to the viewing plane (see Figure 2-4). This measuring point represents the common trajectory that all points on the line follow due to the rotation ([13] pp. 73–87).

Our interest in this technique stems from its potential use in rotating objects in perspective relying entirely on vanishing points. Even if we deem this specific technique impractical for computer implementation, it provides motivation for exploring three-dimensional-like object rotation in perspective. In chapter 4, I show a computational method for simulating object rotation that is completely different from this technique and is quite easy to use.

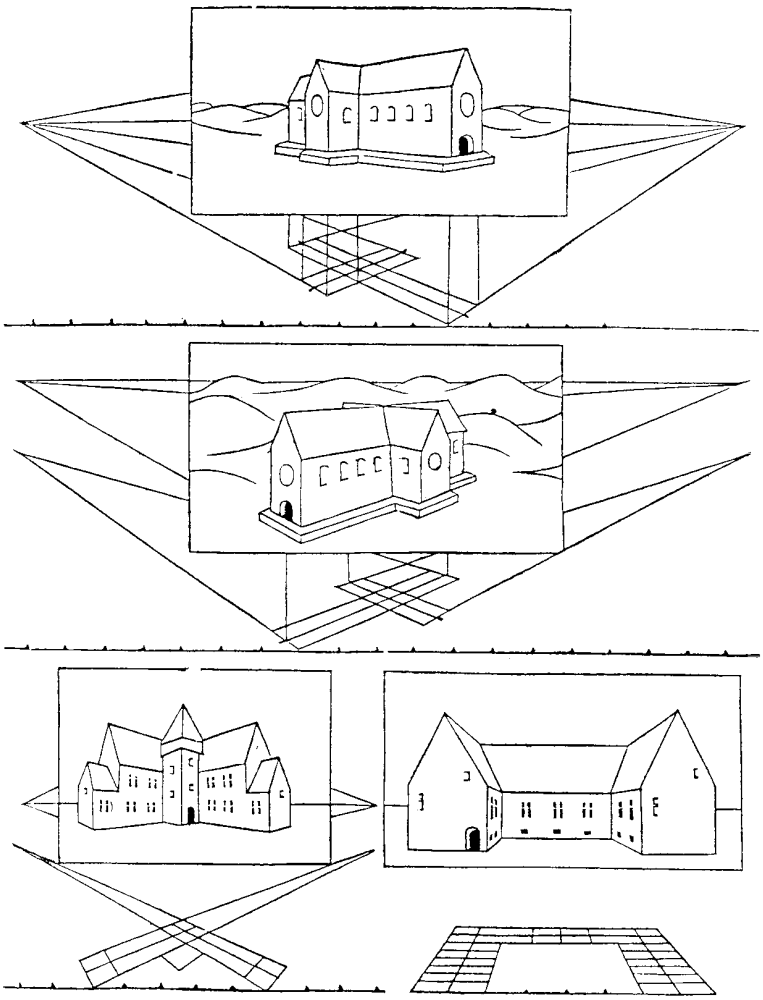


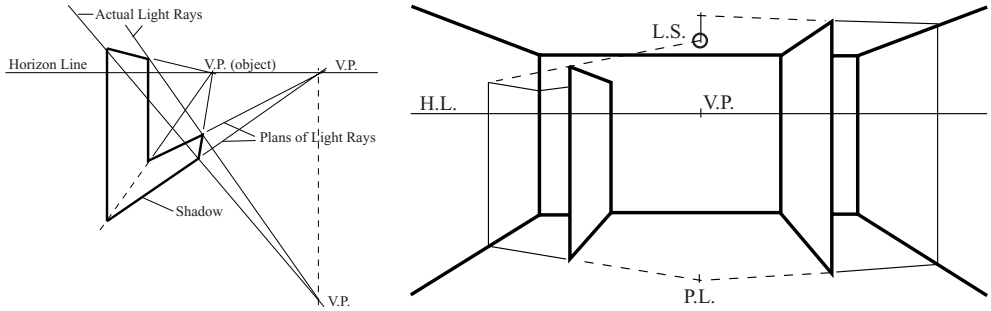
Figure 2-5: Viator's diagrams showing extrusion from plan.

Extrusion. A typical architectural perspective scene is first constructed in plan view, using the above-mentioned techniques, then walls are extruded in the vertical direction (see Figure 2-5). For single-point and two-point views, the extrusion direction is parallel to the viewing plane. The distance of an extrusion is first measured on the viewing plane then transferred with vanishing points. In the case of a three-point perspective, where the viewing plane tilts either up or down, the extrusion becomes much more complicated.

Figure 2-6: An etching by Dürer showing different shading patterns for perspective.

Shading. Shading greatly enhances the three-dimensional effect of a drawing by providing texture and illumination to the depicted surfaces. The actual texturing technique varies according to the specific drawing medium. For example, in an oil painting, a combination of brush strokes and color variation are used to convey both texture and illumination. On the other hand, in a medium like pen-and-ink or etching the artist is limited by the mono-tonal strokes. Usually, a certain pattern is chosen for a given surface material and the density of the pattern varied to convey changes in illumination. Hatching and stippling also enhance the picture by adhering to the vanishing points of the shaded surface (see Figure 2-6).

Artists also vary the density of the shading pattern as they depict objects at varying distances from the viewer. A delicate tradeoff between the texture's density and its perceived three-dimensional granularity must be achieved. Effects of atmospheric haze may also be taken into consideration when depicting distant objects [7].



(a) (b)

Figure 2-7: Construction of shadow from a directional light source (a) and a point light source (b).

Shadows. Shadows have the powerful effect of relating the perceived position of one object (the shadow-casting object) to another (the object receiving the shadow). Outdoor scenes require the use of a directional light source representing the sun, while indoor scenes have one or more local light sources. A directional light source is represented in perspective by a single vanishing point and is relatively easier to use than local light sources. This vanishing point is used in constructing shadow outlines. Each edge in a shadow outline is projected by imagining a three-dimensional plane that is parallel to the direction of light and encompasses the shadow-casting edge. A shadow edge lies at the intersection of such a plane and the surface that receives the shadow. Using this conceptualization, the artist determines the *directions* of shadow edges and plots them as vanishing points that are used to draw the final shadow outline. The artist then uses shading techniques to emphasize the area of the shadow (see Figure 2-7).

2.2 Projective Geometry

Different kinds of two-dimensional spaces exist. The most commonly used type is the Euclidean plane, which is usually parameterized by a *Cartesian* coordinate system, in which the position of a point is specified by its distance from two orthogonal axes. When the axes are non-orthogonal, the space is labeled "affine". Extending

the Euclidean (or affine) plane by postulating a line at infinity creates a more general type of two-dimensional spaces known as the projective plane [5]. This is called the straight model of projective space. Another mental model of projective space is the spherical model (see [29] for details). Both models are useful for studying perspective images. We can envision the viewing plane, π, of perspective, extended to infinity, as a copy of the straight model, while points in the spherical model can represent vision rays of perspective.

Projective points can be expressed analytically by means of *homogeneous coordinates*, where a point is a non-zero triplet of real numbers, and non-zero scalar multiples are considered equivalent. In this thesis, I will always write this triplet as a column vector: $\mathbf{m} = [x, y, w]^{\mathrm{T}}$. A line is also represented by a non-zero triplet called the line's *homogeneous coefficients*, which will be written as a row vector: $\mathbf{l} = (a, b, c)$. Any such line is incident to all points $\mathbf{m}$, such that: $\mathbf{l}\,\mathbf{m} = ax + by + cw = 0$. Non-zero scalar multiples of line coefficients correspond to the same line.

The standard coordinates in the straight model are $[x/w, y/w, 1]^{\mathrm{T}}$, with the special case that homogeneous coordinates with $w = 0$ are mapped to the line at infinity, Ω, of the straight model. Point coordinates (and line coefficients) in the spherical model are scaled such that $x^2 + y^2 + w^2 = 1$. The straight and spherical models are related by these scaling operations. Geometrically, a point on the sphere, its straight model equivalent, and the center of projection are collinear. Points on the great circle parallel to the plane π are projected to Ω of the straight model.

There are many advantages to the spherical model over the straight model, especially when using computers. For example, the special role that Ω plays in the straight model disappears in the spherical model. In addition, the division by w in the straight model may yield unpredictable results for very small values of w.

By envisioning the spherical model of the two-dimensional projective space as a unit sphere embedded in three-dimensional space, we can use this model to interpret and manipulate perspective imagery. Intuitive and useful interpretations of points in the spherical model are given throughout this thesis. However, I will start with a few basic examples that I build upon later.

Central projection. The simplest form of central projection can be expressed in matrix form as follows:

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix},$$

where $[X, Y, Z, W]^{\mathrm{T}}$ are homogeneous coordinates of projective three-dimensional space and $[x, y, w]^{\mathrm{T}}$ are homogeneous coordinates of projective two-dimensional space. The function of this matrix is merely to discard the forth coordinate W—the scale factor—of three-dimensional space. Therefore, a point at infinity ($W = 0$) becomes indistinguishable from any Euclidean point. This formulation is independent of the model of projective two-dimensional space. A straight model, such as planar perspective, is the result of dividing the left-hand vector by w, while a spherical model results from normalizing it (dividing the vector by its Euclidean length $\sqrt{x^2 + y^2 + w^2}$).

Projective lines. In the spherical model, a line corresponds to a great circle on the unit sphere. To understand this, one can envision a three-dimensional plane passing through the center of projection at the origin of the world. The equation of such a plane is similar to the equation of the projective line: $ax + by + cw = 0$, with the vector (a, b, c) representing the surface normal of the three-dimensional plane. Thus, the equation of the line yields a great circle at the intersection of this plane and the unit sphere. We may use the triplet (a, b, c) in all geometric operations involving lines. For example, the intersection of two lines (a, b, c) and (r, s, t) is the point represented by the homogeneous coordinates $[bt - cs, cr - at, as - br]^{\mathrm{T}}$. This result can be achieved with the cross product of the two vectors (a, b, c) and (r, s, t).

Vanishing points. Any line in three-dimensional space has a direction that is independent of its position. This direction can be represented as a point on the *celestial sphere* (sphere at infinity) of the projective three-dimensional space. The

homogeneous coordinates of this point are $[x, y, z, 0]^{\mathrm{T}}$. Under central projection, this point maps to the two-dimensional point $[x, y, z]^{\mathrm{T}}$, which is incident to the projection of any three-dimensional line that is parallel to the original three-dimensional line. In planar perspective, such a point is termed a *vanishing point* due to the fact that projections of parallel lines generally appear to converge at this point (except when it lies on Ω).

Duality of points and lines. Every point $[x, y, w]^{\mathrm{T}}$ is said to have a dual line (x, y, w). This duality plays a fundamental and useful role in projective geometry. Every definition, theorem, or algorithm of projective geometry has a dual, obtained by exchanging the words "point" and "line", and any previously defined concepts with their duals. In the spherical model, if a point is envisioned as the pole of the sphere, its dual line is an equatorial circle, termed the *polar complement* of the point.

Images of Three-dimensional Planes. The set of all directions parallel to a three-dimensional plane form a great circle on the celestial sphere. We shall call this great circle the *line at infinity* for the plane. Since these directions depend only on the orientation of the plane rather than its position, an infinitely large number of parallel planes share a single line at infinity. Its central projection yields a great circle on the unit sphere, which may be envisioned as the intersection of the unit sphere and a representative plane passing through the center of projection. Since all points on this circle are perpendicular to the planes surface normal, it is easy to see that this circle is the polar complement of the point representing the plane's normal. In other words, the homogeneous coordinates of the image of the line at infinity of a plane are the same as those of its surface normal.

Surface normals can be inferred from a perspective image. For example, the surface normal of a three-dimensional rectangle seen in perspective is simply the cross product of its two vanishing points (see Figure 2-8). Later in this thesis I present techniques for inferring surface normals of more complicated geometric shapes. The duality of a plane's surface normal and the image of its line at infinity is a very

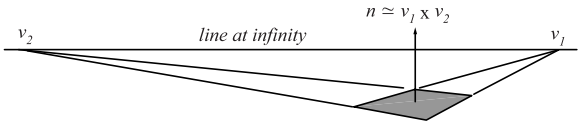


Figure 2-8: Surface normals are inferred by computing the vector cross product of any two vanishing points belonging to the plane. The line joining the two vanishing points is the "horizon" (*line at infinity*) of that plane, along which *all* its vanishing points must lie.

useful concept. For example, the direction of a three-dimensional line parallel to a given three-dimensional plane is at the intersection of this line with the plane's line at infinity. This intersection can be computed in projective two-dimensional space using only the *images* of the three-dimensional line and the the plane's line at infinity. More concretely, if the image of the line and the plane's surface normal were represented with vectors, their cross product yields the direction (vanishing point) of the line.

Collineations. A one-to-one mapping that transforms a set of points and lines in a two-dimensional projective space to an alternate set of points and lines is said to be a *collineation* if collinear points remain collinear, concurrent lines remain concurrent, and the incidence is preserved (i.e., if a point is on a line in the original set it remains on that line after the transformation). A collineation is written as linear mapping of points as follows:

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \lambda \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix},$$

or simply,

$$\mathbf{m}' = \lambda \mathbf{H}\, \mathbf{m},$$

where $\mathbf{H}$ is a non-singular matrix (also called a *homography* of the two-dimensional projective space) and λ is an arbitrary scale factor insuring that the left-hand side is

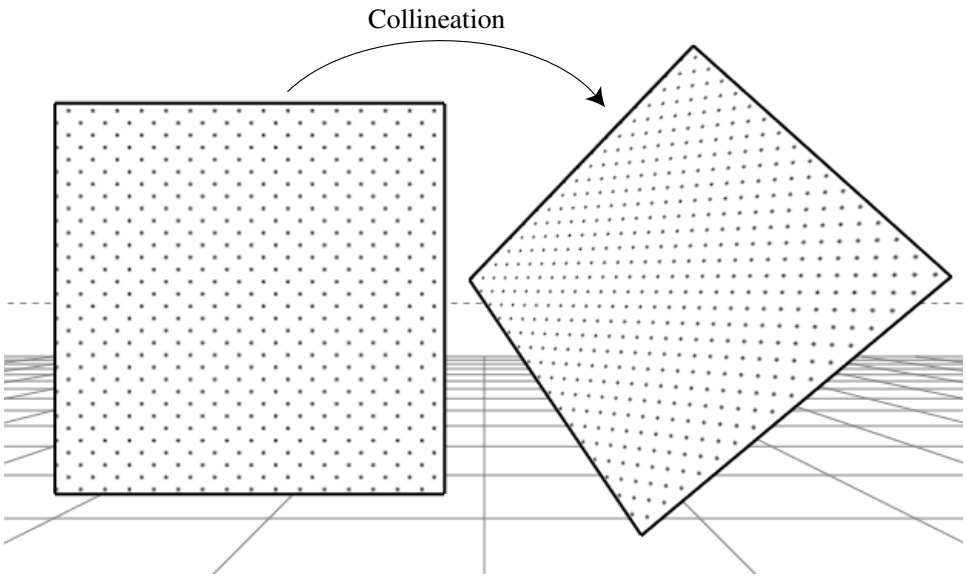


Figure 2-9: Example of a collineation: image of a three-dimensional plane before and after translation and rotation. It is easy to see that the mapping of points from one to the other preserves collinearity, concurrency, and incidence.

a unit-length vector. The matrix representing a collineation is unique up to a single scale factor (i.e., multiplying the matrix by any positive scale factor does not affect the mapping). An example of a collineation is one that relates points in the image of a three-dimensional plane to points in the image of the same plane after translation and rotation (see Figure 2-9).

2.3 Summary

In this chapter I gave brief overviews of traditional perspective techniques and projective geometry. I illustrated that projective two-dimensional geometry—and in particular its spherical model—is well suited for describing and manipulating perspective views. This fact is well-understood and has many existing applications in computer vision and image-based rendering. However, as mentioned in Chapter 1, the contributions of this thesis lie in demonstrating the use of projective geometry as the basis for a two-dimensional *drawing* program. This involves more than the

mere implementation of a drawing system. There are new problems that require a fresh look at the mathematics of previously understood relationships. The result is a surprisingly new set of drawing manipulations that are previously thought to require three-dimensional models.

Chapter 3

Drawing with Projective Points

In traditional drawing programs, primitives are specified via a collection of two-dimensional points. Generally, these points are described by two coordinates, which can be imagined to lie in a plane. The coordinates specify the position of a point relative to a specified origin and two perpendicular basis vectors. In mathematical parlance, such points are considered two-dimensional Euclidean points.

This Euclidean representation of points is practically universal in all two-dimensional drawing systems. There are, however, alternative representations to two-dimensional points, which are not only more powerful than Euclidean points, but also contain them as a subset. In particular, the set of projective two-dimensional points can be represented using three coordinates in conjunction with the following rules: the origin is excluded, and all points of the form $[x, y, w]^{\mathrm{T}}$ and $\lambda[x, y, w]^{\mathrm{T}}$, where λ is non-zero, are equivalent. The subset of projective points for which a value of λ can be chosen, such that $\lambda[x, y, w]^{\mathrm{T}} = [\lambda x, \lambda y, 1]^{\mathrm{T}}$, is the Euclidean subset.

There are several possible mental models for projective two-dimensional points, which are comparable to the plane of the Euclidean points. I adopt a model in which all projective points lie on a unit sphere. Thus, the preferred representation of the point $[x, y, w]^{\mathrm{T}}$ is the one with λ chosen such that $x^2 + y^2 + w^2 = 1$. We will further restrict all values of λ to be strictly positive. This additional restriction results in a special set of projective points called the *oriented* projective set [29].

One advantage of projective two-dimensional points is the ease with which they can be manipulated. Unlike Euclidean points, translations of projective points can be described by matrix products, thus allowing them to be composed with other matrix products, such as scaling and rotation. Projective points also permit *re-projection* to be described as a simple matrix product. Another advantage of projective points is that points at infinity are treated as regular points. For example, in a Euclidean system the intersection of two parallel lines must be treated as a special case, while using projective geometry it is computed using the line intersection formula: $\mathbf{m} \simeq \mathbf{l}_1 \times \mathbf{l}_2$, with the case of two parallel lines resulting in a point whose third coordinate vanishes. By the principle of duality, this advantage also holds true for the line passing through two points: $\mathbf{l} \simeq \mathbf{m}_1 \times \mathbf{m}_2$. These properties of projective point representations give unique capabilities to a two-dimensional drawing system.

Based on the projective two-dimensional point representation described above, I have implemented a perspective drawing system whose interface is, for the most part, like any other two-dimensional drawing and illustration program. However, the use of projective two-dimensional points provides the system with viewing and drawing tools that preserve and enhance the three-dimensional illusion of perspective. For example, a virtual camera provides an immersive experience that is quite similar to the one found in three-dimensional systems when the viewing position is constant. In addition, built-in primitives enhance the realism and utility of the drawing due to shading, shadow casting capabilities, and powerful three-dimensional-like manipulation. Such tools are not available in existing drawing systems that are based on Euclidean two-dimensional points.

In this chapter I describe the basic capabilities of the perspective drawing system. The first section introduces the new drawing representation while the second section explains how the chosen projective representation allows for flexible and dynamic perspective viewing. A third section describes different types of visual guides that the system supports, and the final section illustrates the basic drawing tools in the system.

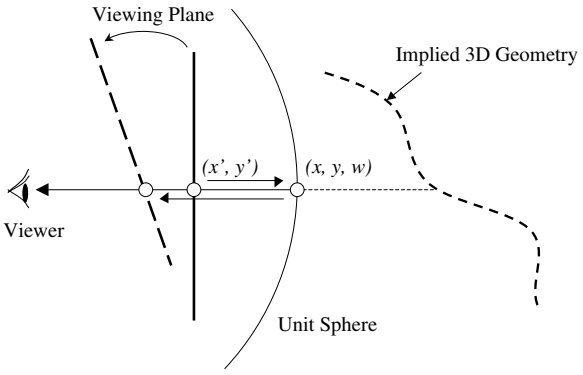


Figure 3-1: Two-dimensional drawing points are stored on the surface of the unit sphere centered about the viewer. They are displayed by projecting them onto a user-specified image plane.

3.1 Drawing Representation

Each stroke (or shape) in the drawing system is stored as a list of such projective points obtained by back-projecting drawn image points to lie on the surface of a unit sphere centered about the viewer, while assuming that the drawing window subtends some solid angle viewing port. The stroke also supports auxiliary attributes such as color and thickness. A drawing is a collection of strokes and shape primitives. This projective representation allows us to generate novel re-projections of the drawing (Figure 3-1). These re-projections can be interpreted as rotations and zooming about a single point in a three-dimensional space. Re-projection of two-dimensional projective points does not, however, permit the changes in viewing positions that result in parallax changes.

The system also uses projective points to represent directions, such as vanishing points, motion trajectories, and infinite light sources. In order to facilitate drawing and manipulation of images of three-dimensional planes, I take advantage of the duality of points and lines in the projective space. For example, using the above-mentioned coordinate system, the dual of the projective point $[a, b, c]^{\mathrm{T}}$ is the line containing all points $[x, y, w]^{\mathrm{T}}$, such that $ax + by + cw = 0$. In my work, I use a projective point to represent the surface normal of a three-dimensional plane. Its

dual is the *line at infinity* for that plane, as well as any plane that shares this surface normal.

3.2 Perspective Viewing

In traditional two-dimensional drawing programs, it is not possible to generate new views looking in a different direction except by reconstructing the drawing from scratch. However, in a projective drawing system it is possible to generate these views instantly, merely by using the computer to re-project the points (see Figure 3-2). This results in panning, tilting, and zooming of a virtual camera analogous to a real zoom camera mounted on a tripod. A similar virtual camera interface is provided by the "QuickTime VR" system [2]. The virtual camera is not only useful for creating and exploring wide angle (panoramic) views, but also in ordinary views because it enhances the immersive feeling of the drawing.

The camera's rotation is controlled by *dragging* a pointing device left/right and up/down, and the zoom level is changed while dragging via keyboard modifiers (shift and control). This direct way of interacting with the drawing allows the user to implicitly specify the camera's rotation and field of view. Alternatively, the user may enter precise angles of rotation and field of view. Such precision may be required for revisiting the view or for aligning the view with a specific direction in the scene.

In the remainder of this section I describe a method for determining the two-dimensional mapping associated with a particular three-dimensional camera motion. An equivalent mapping can be derived by eliminating a row and column from the 4 by 4 matrix describing the three-dimensional camera's motion and projection directly. This technique, however, incurs a greater computation and representation overhead than the approach described here. Furthermore, this method is comparable in terms of the intuition that it provides.

As stated earlier, the perspective drawing system uses projective two-dimensional image points that lie on the surface of a unit sphere centered about the viewer. Unit-length vector (typically labeled $\mathbf{m}$ in this thesis) are used for representing such points:

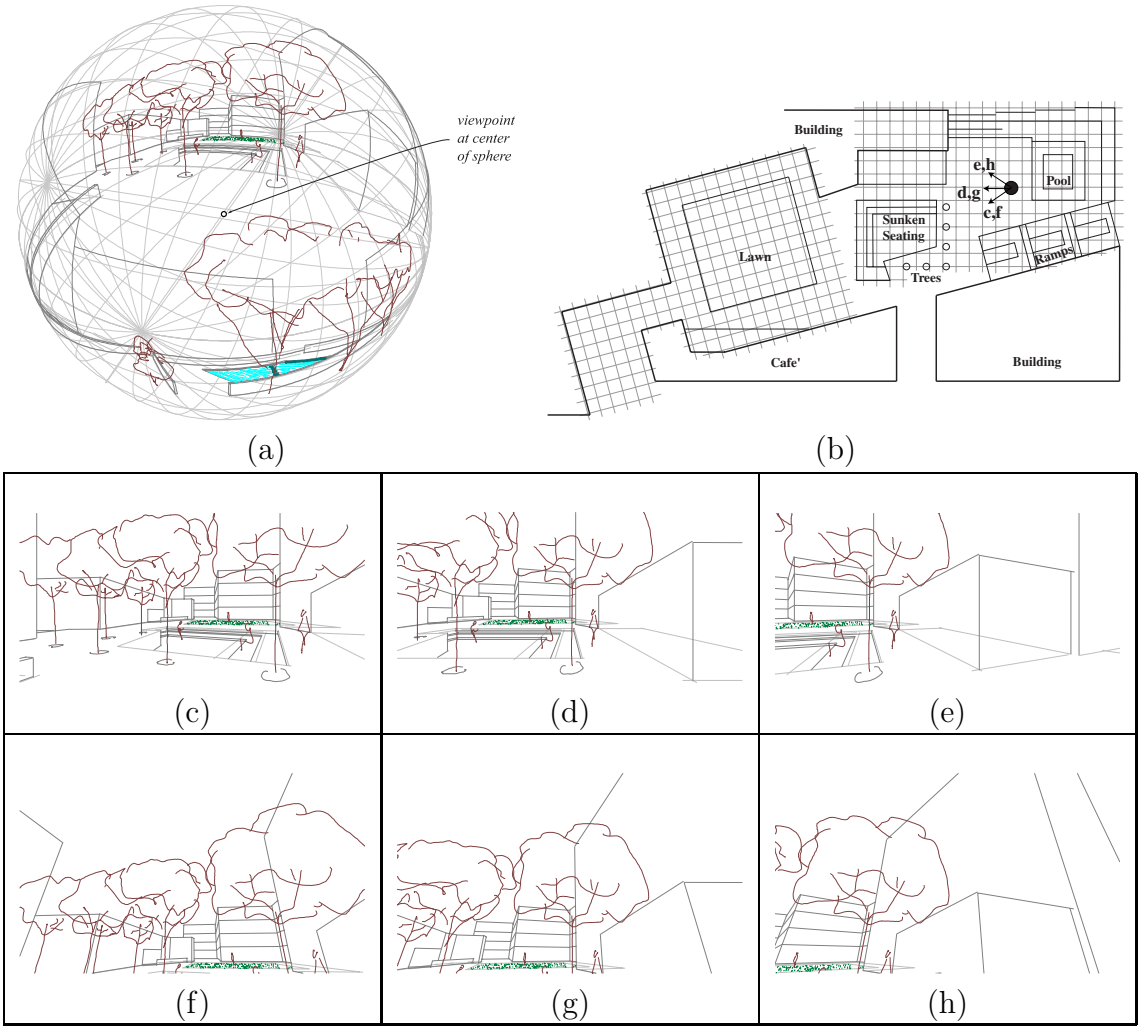


Figure 3-2: A drawing of an outdoor plaza shown as points on the unit sphere centered about the viewer (a), and an array of views (c-h) generated with the drawing system from the same drawing. The bottom row views look in the same directions as the top row but tilt up. The plan drawing in (b) is not generated by the system, it is a schematic drawing that shows where the viewer may have been located in the plaza.

$$\mathbf{m} = \begin{pmatrix} x \\ y \\ w \end{pmatrix}, \quad s.t.\ x^2 + y^2 + w^2 = 1.$$

We desire to specify a projection of these projective points according to a specific class of view changes. All such mappings can be specified by a 3 by 3 matrix called a planar homography, $\mathbf{H}$ [27]:

$$\begin{pmatrix} w'x' \\ w'y' \\ w' \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} = \mathbf{H}\,\mathbf{m},$$

where the displayed point is (x', y'). Note that only points with positive values of w' are considered *in front of* the viewer. Those with negative values of w' are considered *behind* the viewer, while points with $w' = 0$ can be interpreted as *directions* in the image plane. Standard clipping algorithms can be used to determine the drawing primitives, or parts thereof, that actually lie within the viewing frustum. Alternatively, any three-dimensional computer graphics system may be used to render the drawing, thereby potentially taking advantage of hardware acceleration.

We desire to specify the nine elements of the matrix $\mathbf{H}$ according to the desired camera's view. Furthermore, since the projective points that vary by a positive scale factor are considered equivalent, so too will the homographies that vary by such scale factors. Hence, there are only eight parameters, which can be meaningfully interpreted in terms of the virtual camera.

The matrix $\mathbf{H}$ can be decomposed into an upper triangular matrix $\mathbf{U}$ and a rotation matrix $\mathbf{R}$ [1]:

[1]The matrix $\mathbf{H^{-1}}$ can be decomposed, via $\mathbf{QR}$ decomposition [30], into a rotation matrix $\mathbf{R_1}$ and an upper triangular matrix $\mathbf{U_1}$: $\mathbf{H^{-1}} = \mathbf{R_1 U_1}$. From this decomposition, we can derive an alternate decomposition of $\mathbf{H}$ into an upper triangular matrix $\mathbf{U}$ and a rotation matrix $\mathbf{R}$: $\mathbf{H} = (\mathbf{H^{-1}})^{-1} = \mathbf{U_1^{-1} R_1^{-1}} = \mathbf{UR}$.

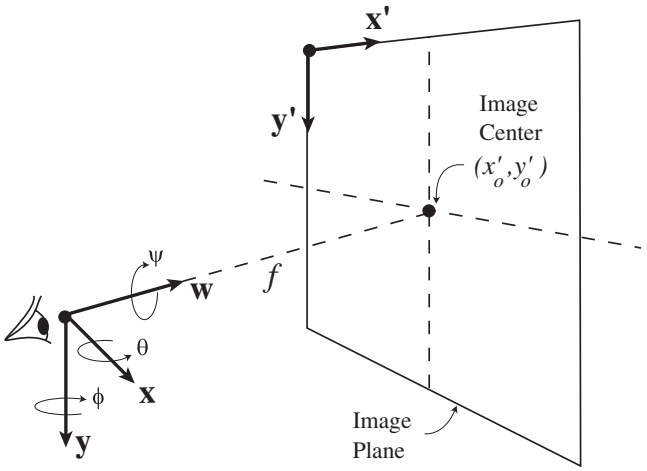


Figure 3-3: Viewing geometry.

$$\mathbf{H} = \mathbf{U}\,\mathbf{R}.$$

The operation of this homography can be easily understood in terms of these two matrices. The upper triangular matrix $\mathbf{U}$ specifies the projection of those points onto a plane. The five non-zero elements of this matrix specify the viewing frustum of the desired projection [11].

$$\mathbf{U} = \begin{pmatrix} f & \sigma & x'_0 \\ 0 & \rho f & y'_0 \\ 0 & 0 & 1 \end{pmatrix},$$

where σ represents skew (as related to the image plane's axes $\mathbf{x}'$ and $\mathbf{y}'$), ρ is the image's aspect ratio, f is the viewer's distance from the image plane, and (x'_0, y'_0) determine how the image is centered relative to the visual axis (see Figure 3-3). However, we limit the system to typical views that are non-skew and have unit aspect ratio.

The rotation matrix $\mathbf{R}$ is analogous to a three-dimensional rotation of the unit

sphere constructed from the specified angles of rotation about the principal axes (see any standard computer graphics text for more details; e.g., [12]). This matrix clearly has three parameters (the angles of rotation about the axes, θ, ϕ, ψ), making the total number of parameters eight. In our case, however, only the first two of these angles are changeable by the user during panning and tilting. Rotation about the visual axis is disallowed in order to avoid user dis-orientation (i.e., $\psi = 0$).

The resulting homography specification gives an intuitive method for specifying the re-projection of the two-dimensional projective points used in the representation.

$$\mathbf{H} = \begin{pmatrix} f & 0 & x'_0 \\ 0 & f & y'_0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{R}_{\theta\phi}.$$

This process of projecting points from the unit sphere onto an image plane can be reversed to map drawn coordinates to their corresponding projective representation.

$$\tilde{\mathbf{m}} = \mathbf{H}^{-1} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix},$$

and the point on the unit sphere is given as $\mathbf{m} = \frac{\tilde{\mathbf{m}}}{\|\tilde{\mathbf{m}}\|}$.

We may now revisit the virtual camera interface in order to clarify how the drawing system converts user interaction into camera motion. When the user drags the input device, the drawing system decomposes the motion into two components: a horizontal component indicating panning, and a vertical one for tilting. More precisely, if the initial device position is (x'_1, y'_1) and the final position is (x'_2, y'_2), we define an intermediate one (x'_2, y'_1) that helps us compute the desired angles. Next, we back-project these points:

$$\tilde{\mathbf{m}}_1 = \mathbf{H}^{-1} \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix}, \quad \tilde{\mathbf{m}}_2 = \mathbf{H}^{-1} \begin{pmatrix} x'_2 \\ y'_2 \\ 1 \end{pmatrix}, \quad \tilde{\mathbf{m}}_{21} = \mathbf{H}^{-1} \begin{pmatrix} x'_2 \\ y'_1 \\ 1 \end{pmatrix}.$$

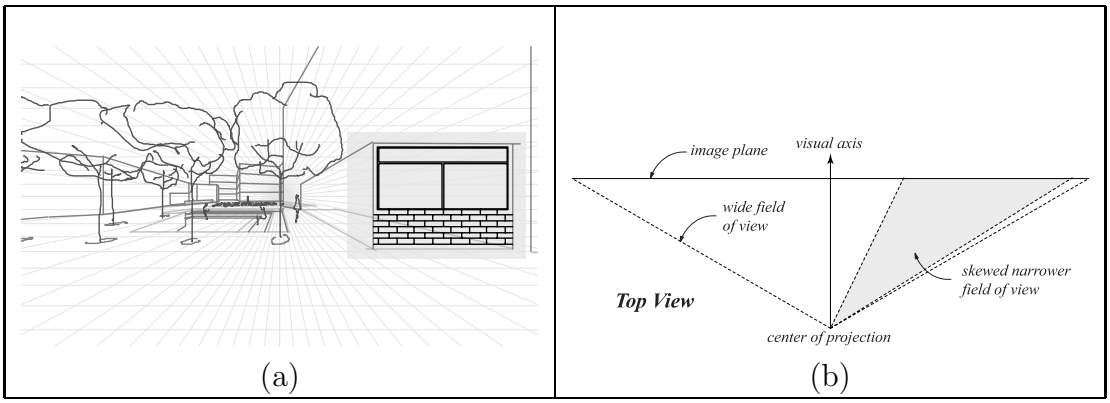


Figure 3-4: The drawing system supports skewed perspective frustums, thereby enabling the user to draw on frontal planes. In (a) the user wished to draw the bricks and window on the facade but the wall occupies a small portion of the screen. Using the "image center" and zoom tools, the grey region can be made to fill the screen. The resulting view is a skewed but narrower field of view (b).

The desired panning and tilting angles are given as follows:

$$\sin\phi = \pm\|\mathbf{m}_1 \times \mathbf{m}_{21}\|, \quad \text{sign}(\phi) = \text{sign}((\mathbf{m}_1 \times \mathbf{m}_{21}) \cdot \mathbf{y}),$$

$$\sin\theta = \pm\|\mathbf{m}_{21} \times \mathbf{m}_2\|, \quad \text{sign}(\theta) = \text{sign}((\mathbf{m}_{21} \times \mathbf{m}_2) \cdot \mathbf{x}),$$

where, as before, $\mathbf{m}_1 = \frac{\tilde{\mathbf{m}}_1}{\|\tilde{\mathbf{m}}_1\|}$, $\mathbf{m}_2 = \frac{\tilde{\mathbf{m}}_2}{\|\tilde{\mathbf{m}}_2\|}$, and $\mathbf{m}_{21} = \frac{\tilde{\mathbf{m}}_{21}}{\|\tilde{\mathbf{m}}_{21}\|}$.

Skewed Perspective Frustums. In addition to rotating the sphere and zooming, the system includes controls for moving the image center (x'_0, y'_0), thereby allowing the user to work on parts of the drawing that would otherwise either lie outside the field of view, or be too small if the field of view were made very wide. For example, when a user orients the view such that a chosen plane is viewed frontally (i.e., parallel to the viewing plane), the plane may be located outside the picture. The user may then use the "image center" tool to bring the plane into the picture in order to draw "on it" proportionately. Useful examples include drawing bricks and windows on a facade (see Figure 3-4). In computer graphics terminology, this operation yields a *skewed* perspective frustum.

3.3 Perspective Guides

The use of projective points provides two new types of visual guides beyond the rulers and regular grids used by traditional two-dimensional drawing systems. These two types are vanishing point guides and perspective grids.

Vanishing Point Guides. Vanishing points are traditionally used as directional guides as well as a means for geometric construction. I maintain them in the drawing system for use as guides when drawing lines and rectangles. I also use vanishing points throughout this thesis to compute various directions and object points. However, some operations that traditionally use vanishing points, such as apparent object motion, can be carried out using mathematical tools that do not rely on vanishing points (see Chapter 4).

The drawing system supports all of the conventional perspective view categories, such as "single-point," "two-point," and "three-point" perspective, since the viewing direction can be changed dynamically, thereby transforming single-point perspective into two- or three-point perspective, and vice-versa. In fact, vanishing points can be specified in arbitrary directions, which need not even be orthogonal (see Figure 3-5). There are always two active vanishing points, which the user selects from built-in directions or creates arbitrary new ones. The system then infers a third vanishing point perpendicular to the selected ones. All three directions can be used in drawing and editing. This third point is not displayed, however, in order to avoid visual clutter.

Vanishing points are directions represented by points on the unit sphere. They can be visualized as poles of a sphere centered about the viewer. When projected onto the image plane, the longitudinal lines of the sphere appear as straight lines converging at the vanishing point, providing the desired visual effect (see Figure 3-6). The fact that a great circle on the unit sphere represents a straight line in perspective was illustrated in Chapter 2.

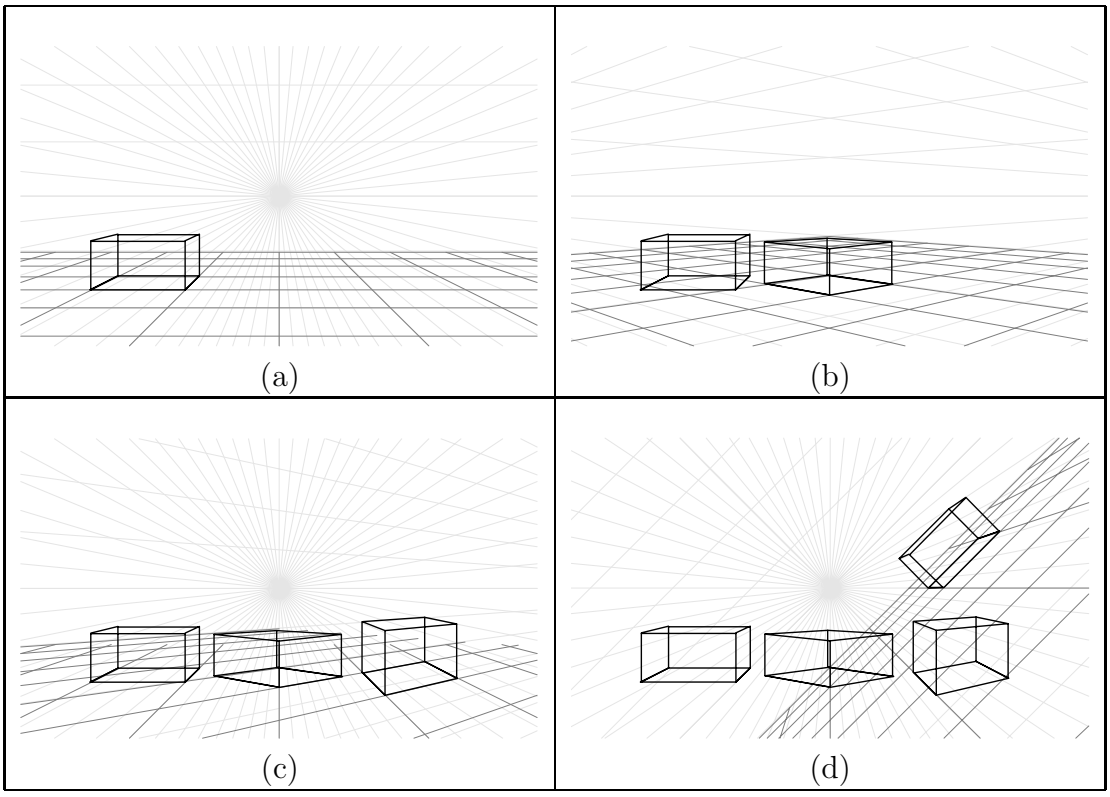


Figure 3-5: The drawing system provides flexible vanishing points and perspective grids as visual guides. Users may select from built-in directions, such as "north" and "east" (a), or "north-east" and "north-west" (b). The directions need not be orthogonal (c), and entry of new arbitrary ones is allowed (d). The "rectangle" tool respects the current vanishing points as well as a third direction that is perpendicular to both of them.

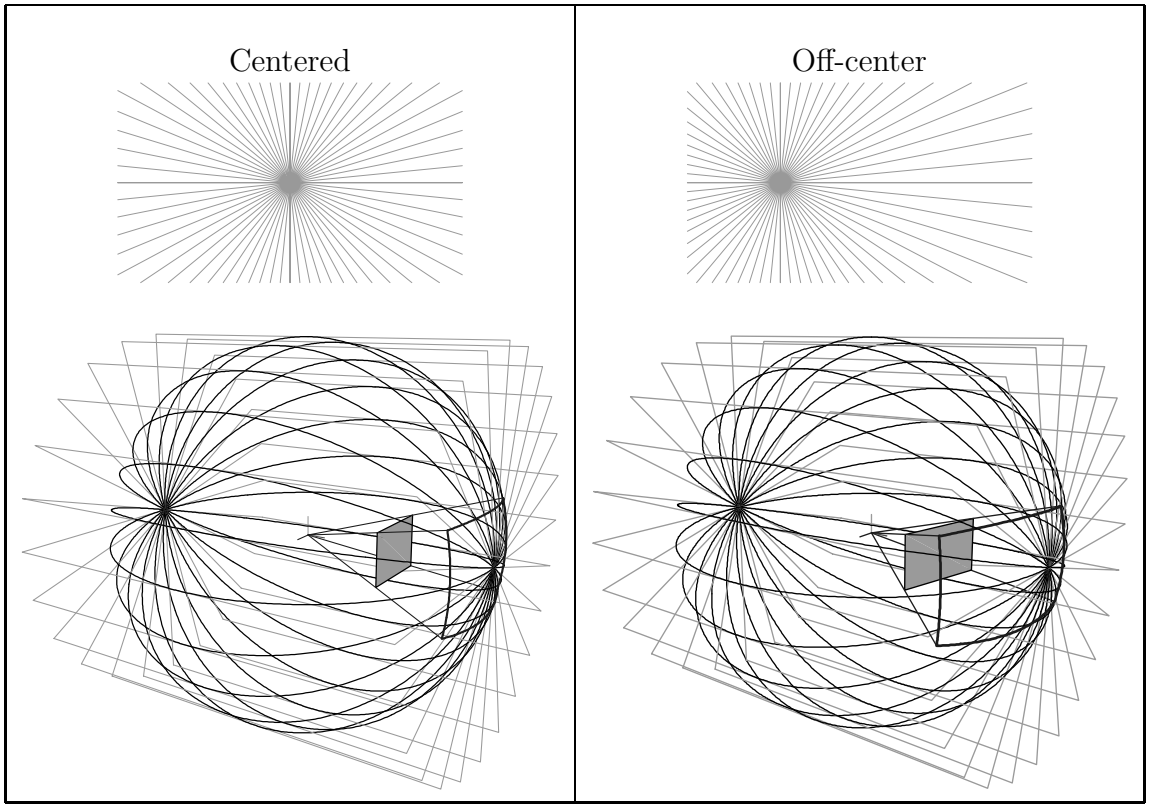


Figure 3-6: Each vanishing point is a direction represented by a point on the unit sphere. A pencil of lines passing through the vanishing point is displayed in perspective for use as visual guides (top row). When visualized on the unit sphere, these lines resemble lines of longitude converging at the vanishing point (bottom row).

Perspective Grids. Following a long tradition of using grids as aids for drawing and scene construction, the drawing system supports perspective grids. The system automatically adjusts the grids to align with the currently active vanishing points. Grids, like vanishing points, can lie in general positions. This provides the interface necessary for drawing views containing parallel lines, rectangles, boxes, etc. (see Figure 3-5).

3.4 Drawing Tools

The drawing system provides great flexibility allowing the user to work with both freehand strokes and built-in primitives. While freehand drawing is desirable for quick sketching and for free-form objects, such as trees, built-in primitives enhance the realism and utility of the drawing due to their shading capabilities and powerful three-dimensional-like manipulation.

Freehand Strokes. Each stroke is stored as a list of projective points obtained by back-projecting drawn image points to lie on the surface of a unit sphere. The stroke also supports auxiliary attributes such as color and thickness. A drawing is a collection of such strokes and other primitives. The system also includes straight line segments, drawn in the familiar rubber-band interface. Lines adhere to vanishing points if the user chooses so.

Built-in Primitives. In addition to basic freehand drawing and straight lines, the system support higher level shape primitives such as "perspective rectangles," which the user specifies with two corner points. Other closed polygons are also supported. Such primitives can have colored interiors for depicting opaque or semi-transparent objects. When these primitives overlap, the order in which they are drawn is used to convey occlusion. As with current two-dimensional drawing programs, the user can adjust this stacking order at any time.

Traditional perspective drawings use shading to convey subtleties of an object's shape. Shading is depicted with a certain light source in mind, and approximately

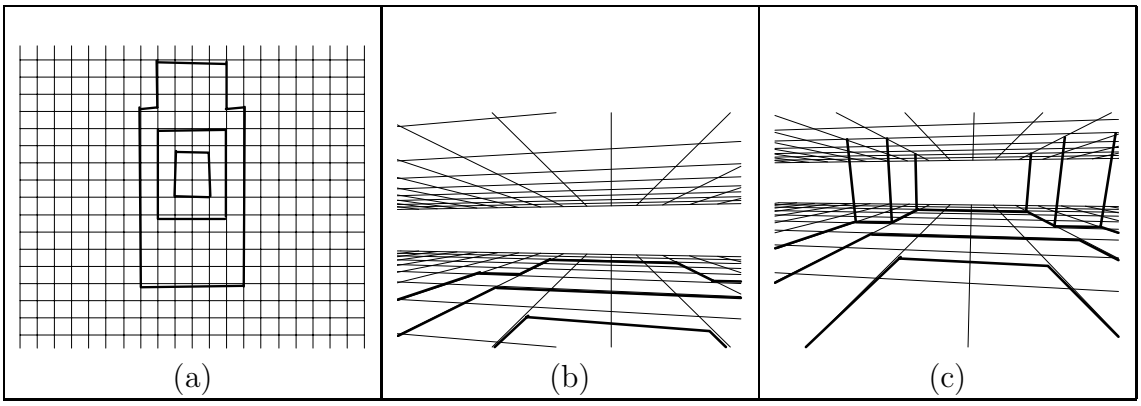


Figure 3-7: A floor plan can be drawn while looking down with a perspective grid parallel to the viewing plane (a), then re-projected into regular perspective views for vertical line extrusion (b, c).

resembles shading in current three-dimensional graphics systems. The drawing system provides this capability by inferring surface normals (see Chapter 2) and allowing the user to insert infinite light sources into the scene. The picture can then be rendered with flat-shaded solid color (e.g., using a Lambertian lighting model) or with artistic styles such as stippling and hatching. Shading with such artistic styles is covered in Chapter 6.

Frontal Views. As mentioned earlier, frontal views, where the image plane is parallel to a specific plane in the scene, are useful for adding strokes that are envisioned to lie on that plane. This is due to the absence of perspective foreshortening in the image of the plane. The geometry of the image becomes, in effect, Euclidean, preserving angles, parallelism, and relative distances (see Figure 3-7.a). Thus, the user can draw with increased accuracy. This technique is one of many ways drawings can be constructed in the system, and may be used when the initial design is expressed in orthographic views, such as plan or elevation, and a perspective visualization is required (see Figure 3-7).

Layered Drawings. The drawing system supports the use of drawings as backdrops in a similar fashion to what traditional drawing programs provide. Such underlays are

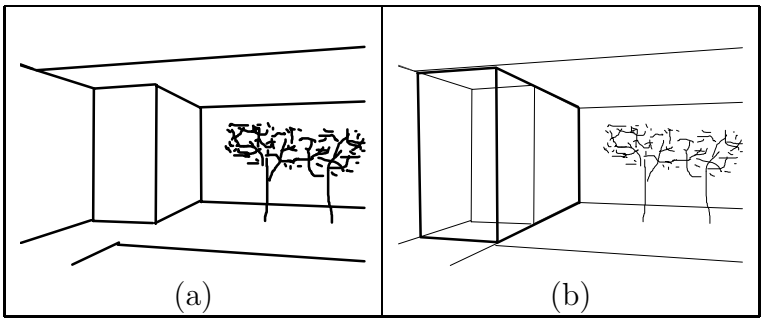


Figure 3-8: Drawings can be layered in order to facilitate visual comparison of different drawings. For example, the drawing in (a) is displayed in subdued color when viewed as an underlay to the design revision shown in (b).

analogous to the traditional use of trace paper, although considerably more flexible. The system displays underlays in subdued color, thereby mimicking the traditional medium, or the user may choose any other color (see Figure 3-8.a, b). Underlays are typically used for either refining design ideas or for drawing on top of drawings of pre-existing site conditions.

3.5 Summary

In this chapter I presented a new point representation for a perspective drawing system based on projective two-dimensional points; i.e., points that lie on the surface of the unit sphere centered about the viewer. I also described key user interface capabilities that enable the user to draw with freehand strokes and simple geometric primitives. More significantly, the user interface supports a virtual three-dimensional camera that provides an immersive viewing experience.

Chapter 4

Perspective Shape Manipulation

The drawing system supports manipulations of shape primitives that appear as 3D rigid-body translations and rotations. These manipulations (together with operations for copying primitives) facilitate the creation of scenes containing symmetric or repetitive elements. These operations require no knowledge of distance or depth of objects, as may initially be imagined. For example, it is possible to carry out transformations of a planar object knowing only its surface normal, which the system automatically infers from user input (see Chapter 3).

The desired user interface lets the user manipulate shapes by dragging them, as in traditional drawing programs. In this chapter I introduce this user interface and mathematical formulations of two-dimensional transformations that cause a planar primitive to appear to undergo three-dimensional motion. Aggregate shapes, such as extrusion shapes, also support this type of manipulation (see Chapter 5 for details).

4.1 Apparent Translation

Techniques for moving a planar object along a linear trajectory in perspective are well known to skilled illustrators. Extensive use of construction lines and multiple vanishing points are needed to perform such an operation. For example, in Figure 4-1 we show a polygon $p_1p_2p_3p_4$ and the new desired position for one of its points (e.g. p_1'). The remaining points are re-positioned using a series of line intersections. For

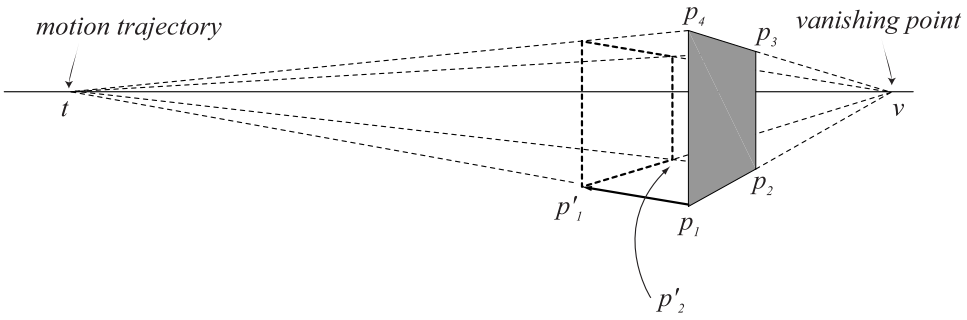


Figure 4-1: The traditional method of performing apparent translation of the plane using vanishing points.

example, p'_2 lies at the intersection of p_2t and p'_1v, and so forth.

A projective drawing system can support this capability using an intuitive interface, both for moving objects and drawing extruded shapes (see Chapter 5 for details of extrusion). The user selects a vanishing point as the motion trajectory (direction of translation) and then uses a pointing device to "drag" the object along the chosen trajectory.

A direct implementation using vanishing points as illustrated in Figure 4-1 is possible. However, this method fails when the trajectory coincides with one of the object's vanishing points. A better approach uses mappings of the projective plane, or in this case, the unit sphere, to accomplish this transformation. Such a mapping, or *homography*, can be thought of as a warping function applied to the object's points through multiplying them by a 3×3 matrix $\mathbf{H}$ as follows:

$$\mathbf{m}' \simeq \mathbf{H}\,\mathbf{m}, \tag{4.1}$$

where $\mathbf{a} \simeq \mathbf{b}$ denotes $\mathbf{a} = \lambda\mathbf{b}$, and λ is an arbitrary scale factor.

It is known that the projected image of a translating three-dimensional plane is

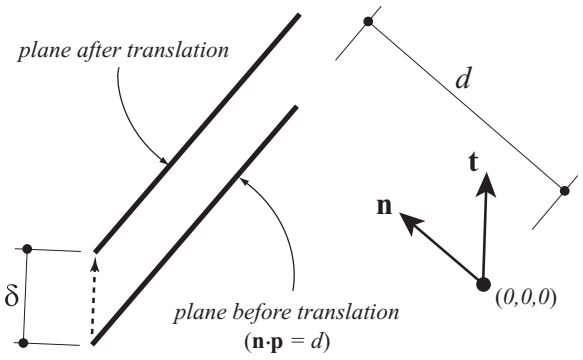


Figure 4-2: Geometry of the translation of a three-dimensional plane.

described by the following homography [19]:

$$\mathbf{H} \simeq \mathbf{I} + \frac{\delta}{d}\mathbf{t}\mathbf{n}^{\mathrm{T}},$$

where $\mathbf{I}$ is the 3×3 identity matrix, $\mathbf{t}$ is the motion trajectory, δ is the translation distance, and $\mathbf{n}$ is the surface normal of the moving plane, whose initial equation in three-dimensional space is $\mathbf{n} \cdot \mathbf{p} = d$ (see Figure 4-2). Since in a two-dimensional projective setting we have no knowledge of the distance d from the surface to the viewpoint or the actual displacement of the plane δ, we deduce a new quantity $\alpha = \delta/d$. This yields a single-parameter family of homographies compatible with the translation of a three-dimensional plane:

$$\mathbf{T}(\alpha) \simeq \mathbf{I} + \alpha\mathbf{t}\mathbf{n}^{\mathrm{T}}. \tag{4.2}$$

All the quantities on the right-hand side of Equation 4.2 are known except for the scalar parameter α, which can be inferred from a single pair of points $(\mathbf{m}, \mathbf{m}')$ given the location of a point on the surface before and after the translation. Such a pair can be specified using the pointing device, and must be constrained to lie on the selected trajectory, hence the single degree of freedom (see Figure 4-3). We determine α as follows:

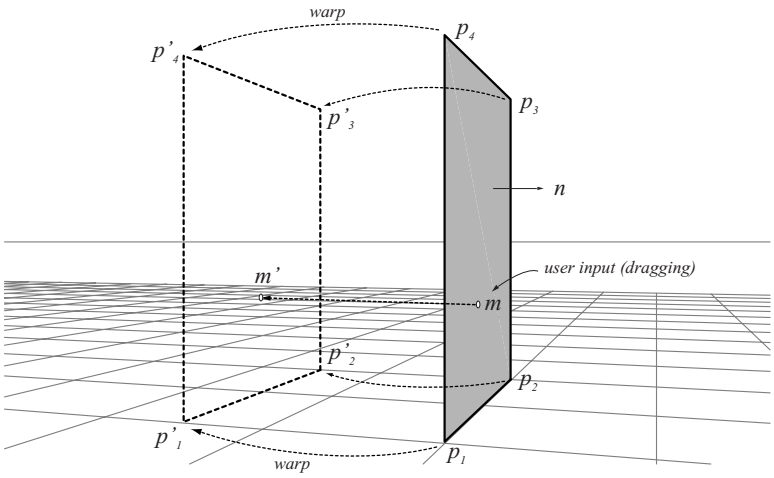


Figure 4-3: Apparent translation of the plane as it is carried out in the drawing system: Image points are transformed (warped) using a homography that is inferred from two input points (m, m') and the plane's normal. The motion trajectory is selected by the user from a list of active vanishing points.

$$\alpha = \pm\frac{\|\mathbf{m}' - \lambda\mathbf{m}\|}{\lambda(\mathbf{n}\cdot\mathbf{m})},\ \mathrm{sign}(\alpha) = \mathrm{sign}(\mathbf{t}\cdot(\mathbf{m}' - \lambda\mathbf{m})). \tag{4.3}$$

The value of λ is given in the following derivation:

From Equations 4.1 and 4.2 we have:

$$\mathbf{m}' = \lambda(\mathbf{I} + \alpha\mathbf{t}\mathbf{n}^{\mathrm{T}})\mathbf{m} = \lambda\mathbf{m} + \lambda\alpha\mathbf{t}(\mathbf{n}\cdot\mathbf{m}), \tag{4.4}$$

where λ is a scale factor needed to solve for α. First, we eliminate α by taking the cross product of Equation 4.4 with $\mathbf{t}$:

$$\mathbf{m}' \times \mathbf{t} = \lambda(\mathbf{m} \times \mathbf{t}).$$

This is an equation of the form: $\mathbf{a} = \lambda\mathbf{b}$ (vector $\mathbf{a}$ is λ-times vector $\mathbf{b}$), the solution for which is:

$$\lambda = \pm\frac{\|\mathbf{m}' \times \mathbf{t}\|}{\|\mathbf{m} \times \mathbf{t}\|},\ \mathrm{sign}(\lambda) = \mathrm{sign}((\mathbf{m}' \times \mathbf{t})\cdot(\mathbf{m} \times \mathbf{t})).$$

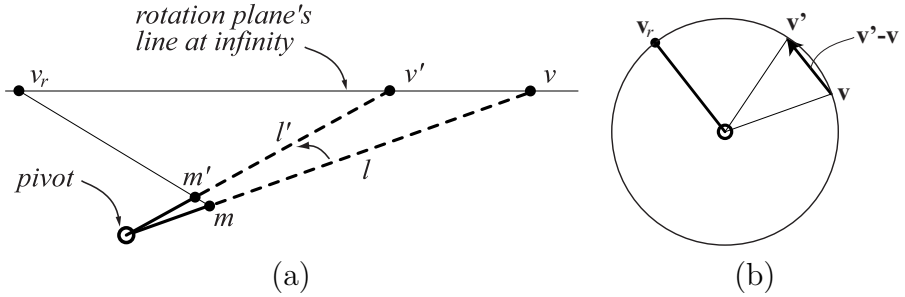


Figure 4-4: Traditional method for rotating a line in perspective (a). We wish to map points on the line l, which has an initial direction v, onto the line l' passing through the pivot but oriented towards v'. Since all such points travel in a common trajectory v_r, we can accomplish this with line intersections. In the spherical representation (b) this trajectory is computed as $\mathbf{v}_r \simeq \mathbf{v}' - \mathbf{v}$.

In our application, λ is always positive. We then rewrite Equation 4.4 as: $\mathbf{m}' - \lambda\mathbf{m} = \alpha\lambda(\mathbf{n} \cdot \mathbf{m})\mathbf{t}$, and solve for α as shown in Equation 4.3.

Note that $\mathbf{n} \cdot \mathbf{m} = 0$ in the denominator of Equation 4.3 means that if the plane passes through the origin ($d = 0$), or is viewed "edge-on," the image of the planar shape is reduced to a line. In this case we cannot use a homography to simulate three-dimensional motion.

4.2 Apparent Rotation

The traditional techniques for performing object rotations in perspective are less well-known than those for object translation. However, one of the traditional techniques reviewed in Chapter 2 implicitly applies rotations to line segments using special vanishing points. The so-called "measuring point" technique derives its name from the fact that one can measure the length of a line segment coincident with the image plane then rotate the line segment into the desired direction. More generally, however, it is possible to accomplish this for any pivot and initial direction (see Figure 4-4).

As with apparent translation, the apparent rotation of a two-dimensional perspective primitive about a fixed point, or pivot, can be simulated using homographies. For example, a perspective rectangle can appear to revolve about an arbitrary axis

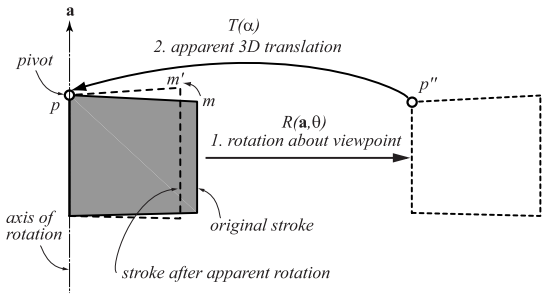


Figure 4-5: Apparent rotation of the plane is carried out in two steps: In the first step, the plane is rotated about the viewpoint. Then, using an apparent three-dimensional translation, it is moved back to the pivot point.

passing through a user-selected pivot. Once we have established the rotation axis, pivot and angle, we rotate the object in two conceptual steps (see Figure 4-5):

1. In the first step, we rotate the object about the viewpoint (at the origin of the world) using the rotation axis and angle desired for the local rotation. All object points, including the pivot itself, move to an intermediate position:

$$\mathbf{m}'' = \mathbf{R}(\mathbf{a}, \theta)\, \mathbf{m}.$$

2. Then, we use apparent three-dimensional translation (Equation 4.2), where $\mathbf{t} \simeq \mathbf{p} - \mathbf{p}''$, to move the object back to the original pivot:

$$\mathbf{m}' \simeq \mathbf{T}(\alpha : \mathbf{p}'' \rightarrow \mathbf{p})\, \mathbf{m}''.$$

Thus, the desired apparent rotation homography is a composition of a three-dimensional rotation matrix and a pseudo-three-dimensional translation homography:

$$\mathbf{m}' \simeq \mathbf{T}(\alpha)\, \mathbf{R}(\mathbf{a}, \theta)\, \mathbf{m}.$$

An intuitive user interface allows the user to specify the rotation parameters. First, the user selects the axis from a list of active directions (vanishing points) and uses the

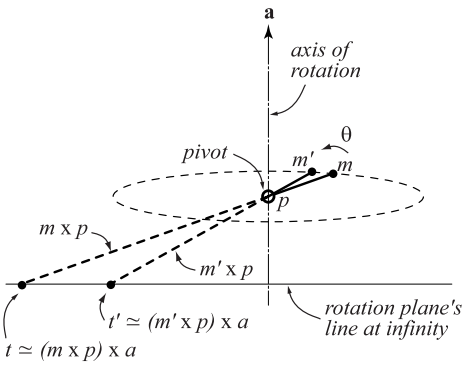


Figure 4-6: The rotation angle θ is inferred from a pair of input points (m, m') indicating the position of a point before and after rotation. The point rotates in a plane whose line at infinity is the dual of the rotation axis. By extending the lines mp and $m'p$ to this line at infinity we get the trajectories t and t', which completely specify the rotation angle.

pointing device to specify the pivot. Then the rotation angle is specified by dragging the pointing device about the pivot, in a manner similar to current two-dimensional drawing programs. For proper visual feedback, however, we make the pointing device appear to orbit in a three-dimensional plane perpendicular to the rotation axis and passing through the pivot. Therefore, we infer the rotation angle from the change in the *direction* of the line joining the pivot $\mathbf{p}$ and the pointing device $\mathbf{m}$ (see Figure 4-6):

$$\theta = \pm \sin^{-1}(\|\mathbf{t} \times \mathbf{t}'\|),\ \text{sign}(\theta) = \text{sign}((\mathbf{t} \times \mathbf{t}') \cdot \mathbf{a}),$$

where $\mathbf{t}$ and $\mathbf{t}'$ represent the above-mentioned direction before and after the rotation, given by: $\mathbf{t} \simeq (\mathbf{m} \times \mathbf{p}) \times \mathbf{a}$, and $\mathbf{t}' \simeq (\mathbf{m}' \times \mathbf{p}) \times \mathbf{a}$. We arrive at these directions by computing the intersection of the lines mp and $m'p$ with the rotation plane's line at infinity, whose coefficients equal the coordinates of the rotation axis.

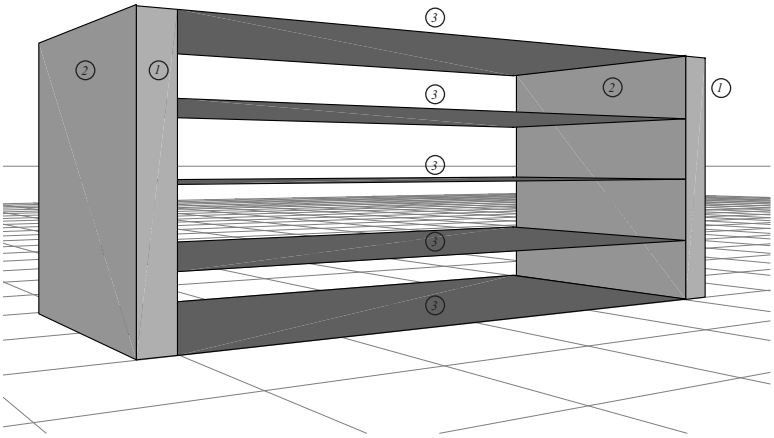


Figure 4-7: Examples showing the emulation of three-dimensional translation. This drawing of a shelf cabinet is created from transformed copies of three planar primitives (numbered 1 through 3).

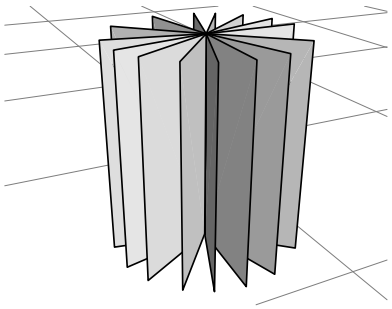


Figure 4-8: Example showing the emulation of three-dimensional rotation.

4.3 Examples and Summary

In this chapter, I presented user interfaces and special projective mappings that allow the user to manipulate drawing primitives in such a way that makes them appear to undergo three-dimensional rigid-body translations and three-dimensional rotations. These three-dimensional-like operations provide flexibility and, together with copying operations, they facilitate the creation of composite three-dimensional-like objects (see Figures 4-7, 4-8).

Chapter 5

Aggregate Shapes

The planar primitives, along with associated shape manipulation operations, introduced in Chapter 4, form the basis for constructing drawings with objects that mimic three-dimensional models. The next logical step is to allow the artist to draw more complex shapes with the same ease and fluidity as with freehand strokes. These shapes must also support the apparent motion operations that I introduced in the previous chapter. The principal idea behind modeling such shapes is to represent them as aggregates of planar primitives. This approach enables us to build upon the ideas and mathematical formulations explained thus far.

In this chapter I show how complex aggregate shapes can be modeled and displayed as shaded two-dimensional polygons. I describe "extrusion" shapes as an example of such aggregate shapes. However, the principles described here are potentially applicable to other shapes as well, such as surfaces of revolution.

5.1 Extrusion

We wish to construct shapes that mimic three-dimensional extrusion surfaces, by which I mean a three-dimensional surface that connects one instance of a three-dimensional curve to another that has undergone some transformation (typically translation only). Traditional perspective drawing relies heavily on extrusion for scene construction. Typical scenes contain architectural elements, such as walls, openings

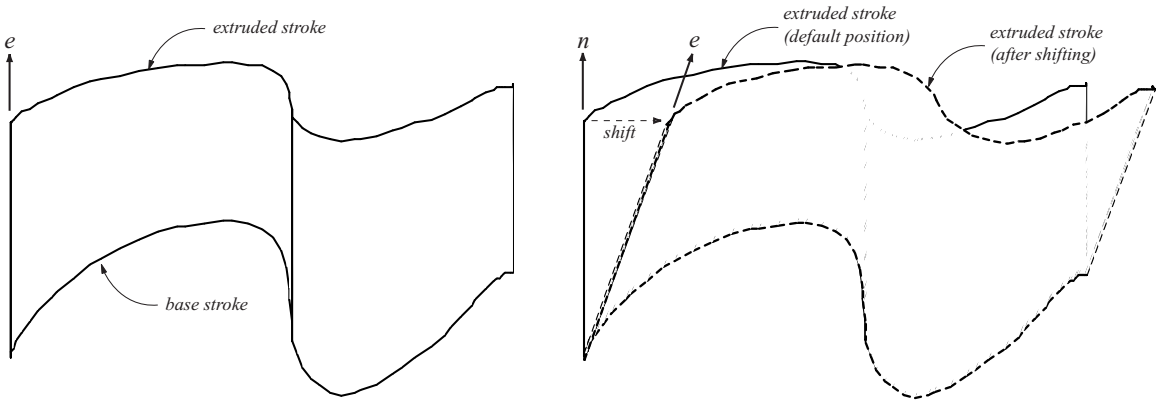


Figure 5-1: By default, the extrusion trajectory is perpendicular to the base stroke. However, skewness can be introduced by shifting the extruded stroke in any chosen direction.

and friezes, which are extruded from their cross-sections. The extrusion is facilitated by vanishing points as shown for the apparent translation case (see Figure 4-1). The projective drawing system provides an interface for modeling such shapes with ease.

Inside the drawing system, the user draws a freehand "base stroke," selects the extrusion trajectory from the list of active vanishing points, and then *drags* the pointing device to specify the magnitude of the extrusion. The system responds by making a copy of the base stroke, which I call the "extruded stroke," and applies apparent three-dimensional translation to this copy using a variant of Equation 4.2:

$$\mathbf{T}(\alpha_e) \simeq \mathbf{I} + \alpha_e \mathbf{e}\mathbf{n}^{\mathrm{T}},$$

where α_e is inferred from the dragging action (as described in Section 4.1), and $\mathbf{e}$ is the selected extrusion trajectory. Segments of the new extruded stroke are connected to corresponding ones in the base stroke, thereby forming the facets of the shape.

This approach assumes that the base stroke represents a planar curve in three-dimensional space, with the extrusion perpendicular to it. Therefore, the system's interface initially assigns the base stroke's normal as the extrusion direction. Later, the user may shift the extruded stroke in any direction, thereby simulating a skewed

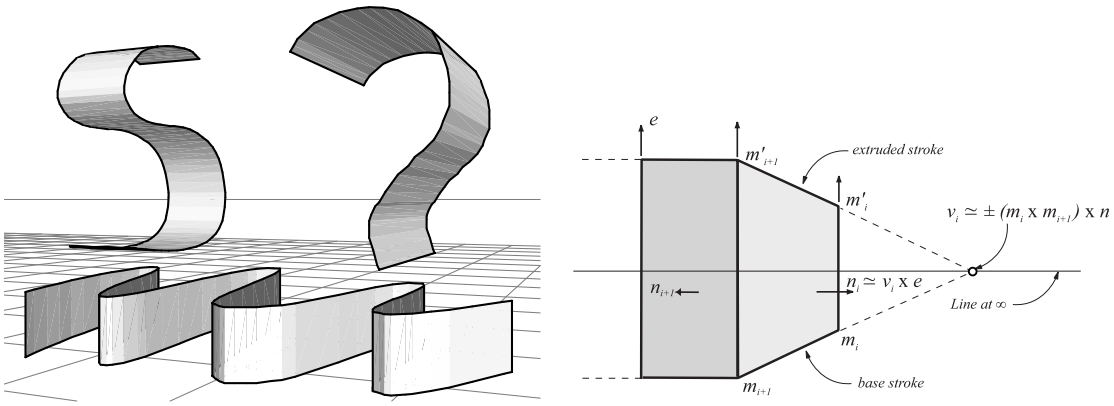


Figure 5-2: Examples of freehand strokes extruded in various directions inside the system (a). An extrusion shape is created by making a copy of the base stroke and transforming it via a pseudo-three-dimensional translation along the extrusion direction (b). The normal of each facet is computed by first intersecting the line joining m_i and m_{i+1} with the base stroke's line at infinity in order to determine a vanishing point v_i, and then computing the normal as the cross product of this vanishing point with the extrusion trajectory.

extrusion shape (see Figure 5-1). The normal to each facet is inferred from the vanishing point of its base segment and the extrusion trajectory (see Figure 5-2), allowing for shading of the facets and shadow projection using directional light sources.

Since an extrusion shape is composed of two planar strokes, it is possible to perform apparent object motion for each of the member strokes as described in the previous chapter. However, we must derive methods for tying the two motions together in order to preserve the integrity of the shape. I describe such methods for apparent translation and rotation in the following sections.

Apparent Translation

The system provides the ability to move an extrusion shape in the scene (see Figure 5-3) using the same interface as for planar objects (Section 4.1). The apparent collective motion of the aggregate shape is performed using two homographies $\mathbf{T}(\alpha)$ and $\mathbf{T}(\alpha')$ for the base and extruded strokes respectively. The system infers the base stroke's parameter α directly from user input as described in the planar object case, while the

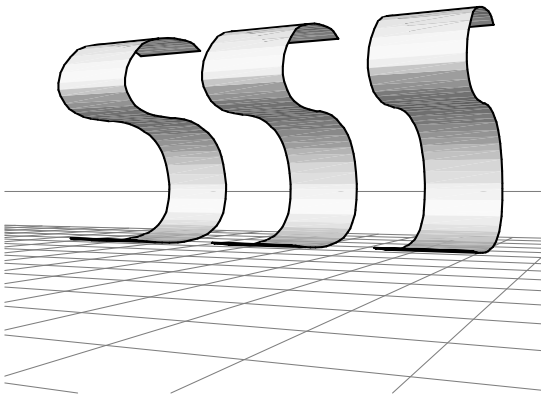


Figure 5-3: Example of an extrusion shape undergoing apparent translation inside the projective drawing system.

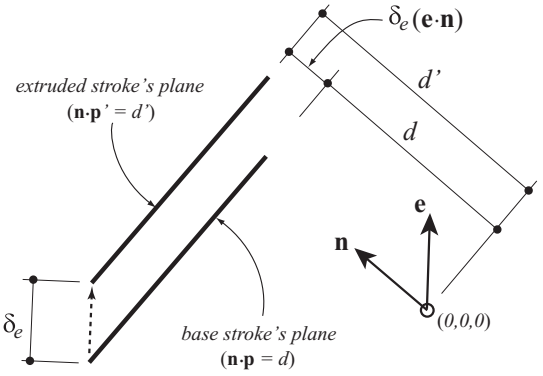


Figure 5-4: Geometry of the three-dimensional planes used in extrusion.

parameter α' for the extruded stroke is determined as follows:

$$\alpha' = \frac{\alpha}{1 + \alpha_e(\mathbf{e} \cdot \mathbf{n})}.$$

The validity of this equation can be shown by the following derivation:

Suppose that the base stroke lies in a three-dimensional plane whose equation is $\mathbf{n} \cdot \mathbf{p} = d$, and the extruded stroke lies in a parallel plane: $\mathbf{n} \cdot \mathbf{p}' = d'$. From Figure 5-4 we can deduce that:

$$d' = d + \delta_e(\mathbf{e} \cdot \mathbf{n}), \tag{5.1}$$

where δ_e is the extrusion distance. From our definition of α in Section 4.1, we have:

$$\alpha_e = \frac{\delta_e}{d}, \quad \alpha = \frac{\delta_t}{d}, \quad \alpha' = \frac{\delta_t}{d'}, \tag{5.2}$$

where δ_t is the translation distance. By substituting the value of d' from Equation 5.1 into 5.2, we have:

$$\alpha' = \frac{\delta_t}{d + \delta_e(\mathbf{e} \cdot \mathbf{n})}$$

Since $\delta_e = d\alpha_e$ (Equation 5.2), we have:

$$\alpha' = \frac{\delta_t}{d + d\alpha_e(\mathbf{e} \cdot \mathbf{n})} = \frac{\delta_t/d}{1 + \alpha_e(\mathbf{e} \cdot \mathbf{n})} = \frac{\alpha}{1 + \alpha_e(\mathbf{e} \cdot \mathbf{n})}$$

Apparent Rotation

Rotation of an aggregate shape about an arbitrary axis and pivot is also possible. Any point can serve as the origin of the rotation. For simplicity, I assume the pivot to be the first point on the base stroke. The system performs the rotation in a series of steps (hidden from the user), during which the motion of the extruded stroke is tied to that of the base stroke (see Figure 5-5):

1. Rotate the base stroke using its first point as pivot.

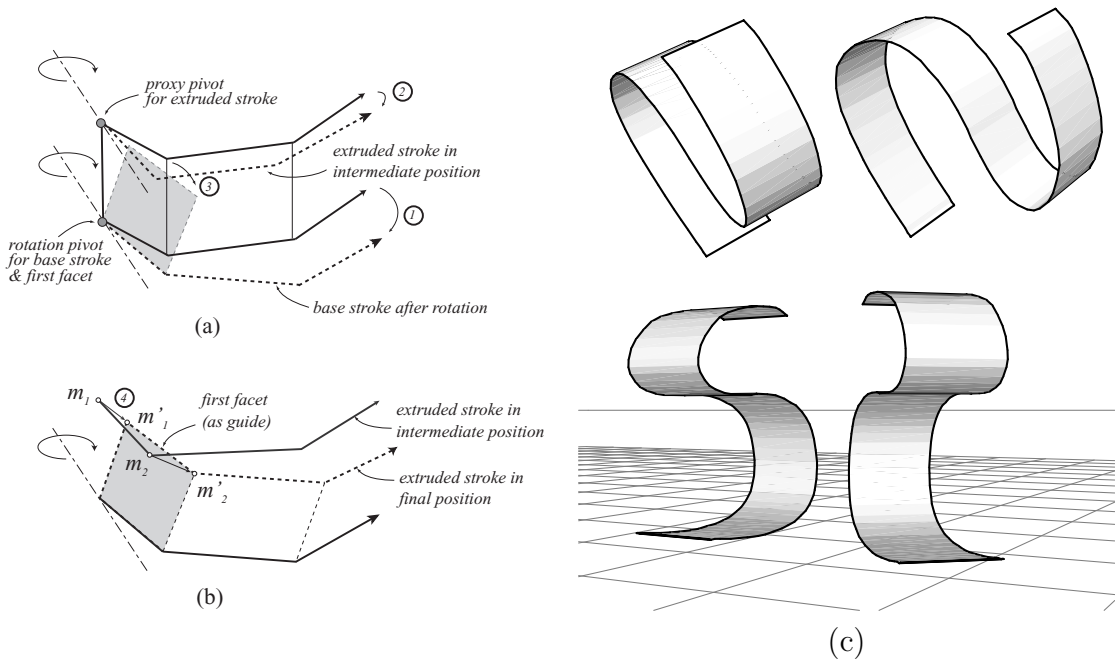


Figure 5-5: The drawing system rotates an extrusion shape in several steps (numbered 1-4), which are hidden from the user: First, both the base and extruded strokes are rotated using their respective first points as pivots. The first facet is also rotated in order to determine the final position of the extruded stroke (a). Then the extruded stroke is moved to the correct position using the first facet as guide (b). Examples of an extrusion shape undergoing different rotations inside the drawing system are shown in (c).

2. Rotate the extruded stroke using *its* first point as pivot. This results in an intermediate position for the extruded stroke.

3. Rotate the first facet of the shape using the base stroke's first point as pivot. This establishes the correct positions for the first and second points in the extruded stroke.

4. Move the extruded stroke from its intermediate position to the correct position determined in step 3. For this operation, we use apparent three-dimensional translation (Equation 4.2), where $\mathbf{t} \simeq (\mathbf{m_1} \times \mathbf{m'_1}) \times (\mathbf{m_2} \times \mathbf{m'_2})$.

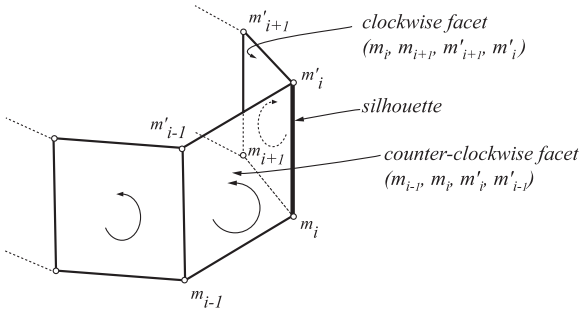


Figure 5-6: An edge of an extrusion shape is determined to be on the silhouette if its neighboring facets are drawn in opposite directions (i.e., clockwise vs. counter-clockwise).

5.2 Silhouettes

Rather than drawing all the facets of an extrusion, and in keeping with the hand-drawn look, I have developed techniques to emphasize the boundaries and silhouettes of extrusion shapes. Silhouettes of faceted surfaces lie at the edges between two facets, one of which is front-facing while the other is back-facing [25]. A simple two-dimensional method can be used to determine the existence of this condition [17]. If the edges of two neighboring facets are drawn in opposite directions (i.e., clockwise vs. counter-clockwise), the shared edge is on the silhouette (see Figure 5-6).

5.3 Visibility

The lack of relative depth information between objects does not allow the system to hide parts of object that are occluded by other objects. However, the system provides an interface that allows the user to adjust the stacking order of these objects, as is commonly done in standard two-dimensional drawing packages. Objects lower in the stack are rendered on top of ones higher above, thereby appearing closer to the viewer. Three-dimensional graphics systems often resort to this technique, referred to as the *painter's algorithm* [23], although the order of primitives in the stack is determined automatically by the graphics system.

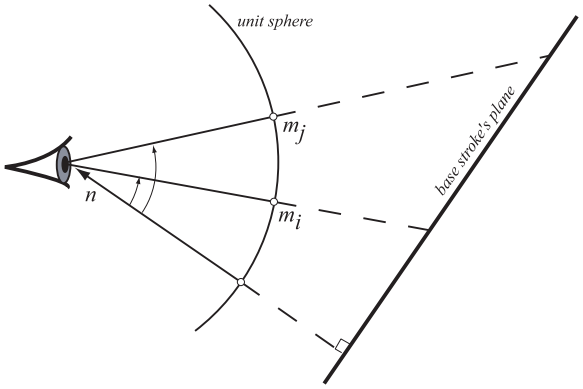


Figure 5-7: Points on a plane make a greater angle with the plane's normal as they move farther away from the viewpoint. This observation is used to draw an extrusion shape in a back-to-front order.

Although inter-object visibility cannot be unambiguously resolved for two-dimensional representations, intra-object visibility can be determined in some instances. For example, the facets of an extrusion shape can be drawn in a back-to-front order using the simple observation that points on the base plane make a greater angle with the plane's normal as they move farther away. For example, in Figure 5-7, $\mathbf{m}_j \cdot \mathbf{n} > \mathbf{m}_i \cdot \mathbf{n}$ (assuming $\mathbf{n}$ points in the opposite direction of $\mathbf{m}$); therefore, a facet based at $\mathbf{m}_j$ is potentially occluded by another based at $\mathbf{m}_i$ (assuming the extrusion shape is not skew). Based on this dot product criteria, the system creates a sorted list of facet indexes that is uses for rendering. Re-sorting of this list is necessary if the object undergoes apparent translation or rotation.

5.4 Example and Summary

This example shows the maze garden at Hampton Court Palace, which was generated by extruding the plan drawing of the maze (see Figure 5-8). Care was taken to maintain a proper depth order amongst the hedges. Since the system relies on a stacking order for conveying occlusion, it is not possible to have one shape wrapping around another. Such a shape must be broken up during modeling into smaller fragments—ones that are either exclusively behind or in front of other objects. This limitation,

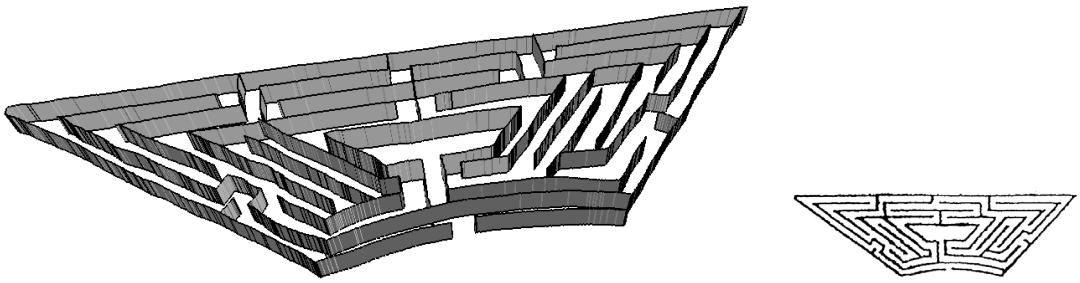


Figure 5-8: Perspective view of the maze garden at Hampton Court Palace (left), which was created by extruding its plan drawing (right).

however, can be mitigated with a "grouping" tool, whereby visibility within a group is resolved on a facet-by-facet basis rather than by objects.

This concludes my discussion of drawing primitives and shape modeling and manipulation within a projective drawing system. I have alluded many times to the possibility of shading and casting shadows from infinite light sources, which I will make more concrete in the following chapter.

Chapter 6

Shading and Shadows

Perspective drawings that contain shaded objects and shadows can be more compelling than line-art drawings, where only the silhouettes of the objects are drawn. Shading permits the viewer to infer the orientation of the depicted surface, while shadows can play an important role due to their effectiveness in conveying shape and relative position information.

While accurate lighting simulations, as accomplished by three-dimensional systems, are not possible within a two-dimensional system, it is possible to provide the user with lighting tools that are sufficiently automated to prove useful. In this chapter, I present these tools and discuss the degree of automation they achieve.

First, I explain how illumination computation is carried out and the different shading styles that the drawing system supports. Then I explain the process of shadow projection in two-dimensions. At the end of this chapter, I present examples of shaded drawings made with the system.

6.1 Illumination and Shading

The drawing system provides shading capabilities by inferring surface normals (see Chapter 3) and allowing the user to insert directional light sources into the scene. Illumination computation is carried out using this information and any local lighting model (see Chapter 16 of [12] for a comprehensive introduction to shading in computer

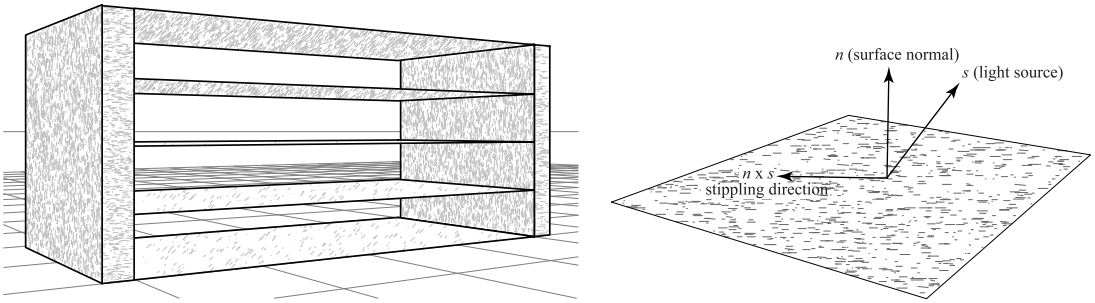


Figure 6-1: Simple example rendered with the stippling style (a). The stippling direction is determined from the surface normal and light direction (b).

```
COMPUTE-STIPPLE
StippleDirection ← n × s
Convert-StippleDirection-To-Screen-Coordinates
StippleDensity ← MaxDensity ×(1 − min(0, n · s))
NumStipples ← StippleDensity × Bounding-Box-Area
for i ← 1 to NumStipples
    do BeginPoint ← Random-Point-Inside-Bounding-Box
       EndPoint ← BeginPoint + (Random-Length × StippleDirection)
       Clip-Stipple-to-Shape; Back-Project-Stipple; Add-to-StippleList
```

Figure 6-2: Pseudo-code for stippling algorithm.

graphics). The picture can then be rendered with flat-shaded solid color or with artistic styles, such as stippling and hatching.

The implemented system allows the user to specify the object's material properties, including diffuse color, reflection coefficient and transparency. The user can also modify the direction and color of the light sources. This provides great flexibility in modifying the final appearance of a drawing.

Stippling. I have implemented a basic stippling algorithm that employs short strokes to shade planar surfaces. The direction of the strokes is determined by the vector cross product of the surface normal and light direction (see Figure 6-1). This gives the viewer added information about the surface orientation. The density of the strokes is determined by a Lambertian shading computation, and their position and length are randomized in order to emulate a hand-drawn look (see code in Figure 6-2).

Hatching. Another shading style that the system supports is a simple hatching method (used in Figure 6-3). This method generates a look that is consistent with that

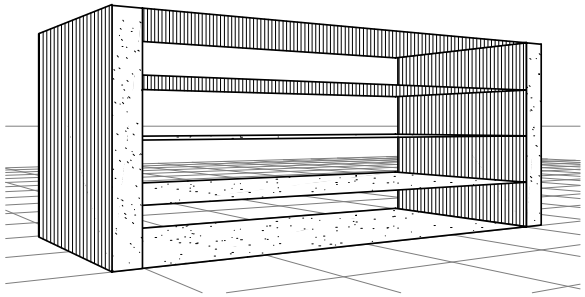


Figure 6-3: Simple example rendered with the hatching style.

of the manual illustrations in [13]. Instead of Lambertian shading, it generates four levels of grey according to the following rules: Shadows are hatched with maximum density, objects facing away from the light are hatched with lighter density, and light stippling is applied to objects that are dimly lit (i.e., the angle between the normal and the light source is greater than 45 degrees).

Due to the computational overhead of artistic shading (about 2 seconds for a complex scene), I adopt the following strategy: Shading strokes are computed in screen coordinates when there is no camera motion, then back-projected and stored on the unit sphere. As the user rotates the camera, the stored strokes are used to render the scene. Although the stored shading becomes somewhat inaccurate during camera motion, this strategy provides adequate feedback during scene navigation and avoids the flickering that would result from re-computing the strokes during camera motion.

6.2 Shadows

Following classical line construction techniques, I have implemented an automatic algorithm that computes the shape of an object's shadow as cast from a directional light source like the sun. However, due to the lack of depth information, the shadow is initially attached to the object casting the shadow, then the user may *drag* it to the desired position (see Figure 6-4). This *dragging* operation is achieved with the "apparent three-dimensional translation" method by using the light's direction as the

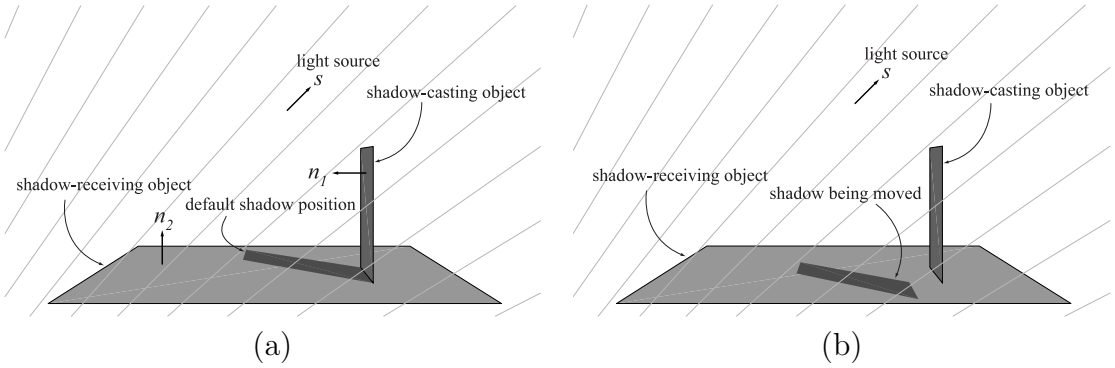


Figure 6-4: Shadows are projected automatically using the surface normals and light source's direction (shown with faded lines). The shadow is initially attached to the object casting the shadow (a). Later the user may *drag* the shadow in order to simulate distance between the shadow-casting and shadow-receiving objects (b). The system automatically re-projects the shadow during this dragging operation.

translation trajectory. Later, if the user re-positions the light source, the new position of the shadow is recomputed automatically, without any further user intervention.

All shadows are treated as surface-detail polygons associated with the shadow-receiving object. These polygons are shaded using the color of the underlying object as it would appear when hidden from the light source that was used for projecting the shadow.

The information that is needed to compute the shadow is the surface normals for both the object casting the shadow and the one receiving it. The shadow of a stroke (or polygon) is determined by marching along the stroke and projecting its successive segments onto the shadow-receiving object. The first shadow point is attached to the corresponding point in the stroke. Thereafter, each shadow point is determined by intersecting a *shadow line* with a *shadow projector*—a line joining the light source and the shadow-casting point (see Figure 6-5). The trajectory of the shadow line is determined by intersecting an imaginary *shadow plane* with the shadow-receiving object. All these operations are performed in two dimensions using vector cross products as shown in the pseudo-code (see Figure 6-6).

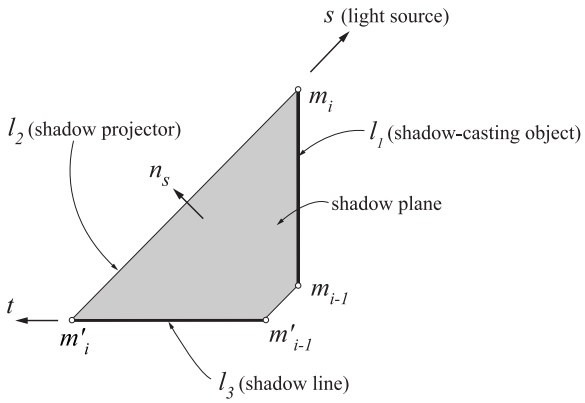


Figure 6-5: The shadow of a stroke is determined by marching along the stroke and projecting the shadow segments iteratively. m'_i is determined from m'_{i-1} by intersecting l_2 and l_3, where l_2 is the shadow projector and l_3 is the shadow line. l_3 is found by intersecting an imaginary shadow plane with the shadow-receiving object.

PROJECT-SHADOW	
$\mathbf{n}_1 \leftarrow$ shadow-casting-object-normal	
$\mathbf{n}_2 \leftarrow$ shadow-receiving-object-normal	
$\mathbf{m}'_1 \leftarrow \mathbf{m}_1$	▷ Shadow attached to first point.
$k \leftarrow length[stroke]$	▷ Number of points in stroke
for $i \leftarrow 2$ **to** k	
do $\mathbf{l}_1 \leftarrow \mathbf{m}_{i-1} \times \mathbf{m}_i$	▷ Shadow-casting stroke line.
$\mathbf{l}_2 \leftarrow \mathbf{s} \times \mathbf{m}_i$	▷ Shadow projector.
$\mathbf{v} \leftarrow \mathbf{l}_1 \times \mathbf{n}_1$	▷ Vanishing point.
$\mathbf{n}_s \leftarrow \mathbf{s} \times \mathbf{v}$	▷ Shadow plane's normal.
$\mathbf{t} \leftarrow \mathbf{n}_s \times \mathbf{n}_2$	▷ Intersection of 2 planes.
$\mathbf{l}_3 \leftarrow \mathbf{m}'_{i-1} \times \mathbf{t}$	▷ Shadow line.
$\mathbf{m}'_i \leftarrow \mathbf{l}_2 \times \mathbf{l}_3$	▷ Shadow point.

Figure 6-6: Pseudo-code for shadow projection.

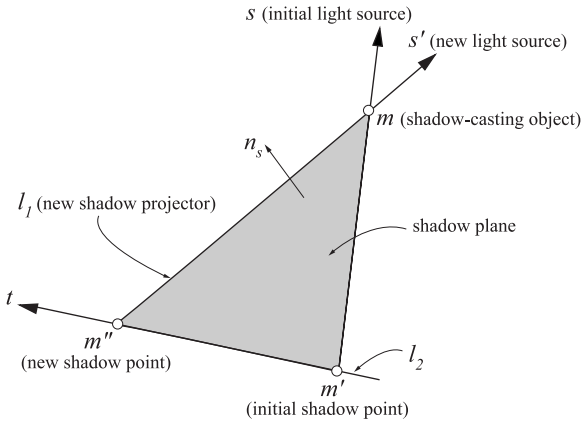


Figure 6-7: A shadow point is re-projected after some change in the light source's direction. The new shadow point is determined by intersecting the lines l_1 and l_2, where l_1 is the new shadow projector, and l_2 is the line along which the movement of shadow point is constrained. l_2 is found by intersecting an imaginary shadow plane with the shadow-receiving object.

REPROJECT-SHADOW-POINT	
$\mathbf{n} \leftarrow$ shadow-receiving-object-normal	
$\mathbf{l}_1 \leftarrow \mathbf{m} \times \mathbf{s}'$	▷ Shadow projector.
$\mathbf{n}_s \leftarrow \mathbf{s} \times \mathbf{s}'$	▷ Shadow plane's normal.
$\mathbf{t} \leftarrow \mathbf{n}_s \times \mathbf{n}$	▷ Intersection of 2 planes.
$\mathbf{l}_2 \leftarrow \mathbf{t} \times \mathbf{m}'$	
$\mathbf{m}'' \leftarrow \mathbf{l}_1 \times \mathbf{l}_2$	▷ New shadow point.

Figure 6-8: Pseudo-code for shadow re-projection.

Using similar techniques, shadows can be automatically re-projected as the light source moves (see Figure 6-7). An imaginary shadow plane is constructed encompassing the old and new shadow projectors—its surface normal inferred from the old and new light directions. The intersection of the shadow plane with the shadow-receiving object gives us the trajectory along which the new shadow point must lie. We intersect this trajectory with the new shadow projector to arrive at the new shadow point (see code in Figure 6-8).

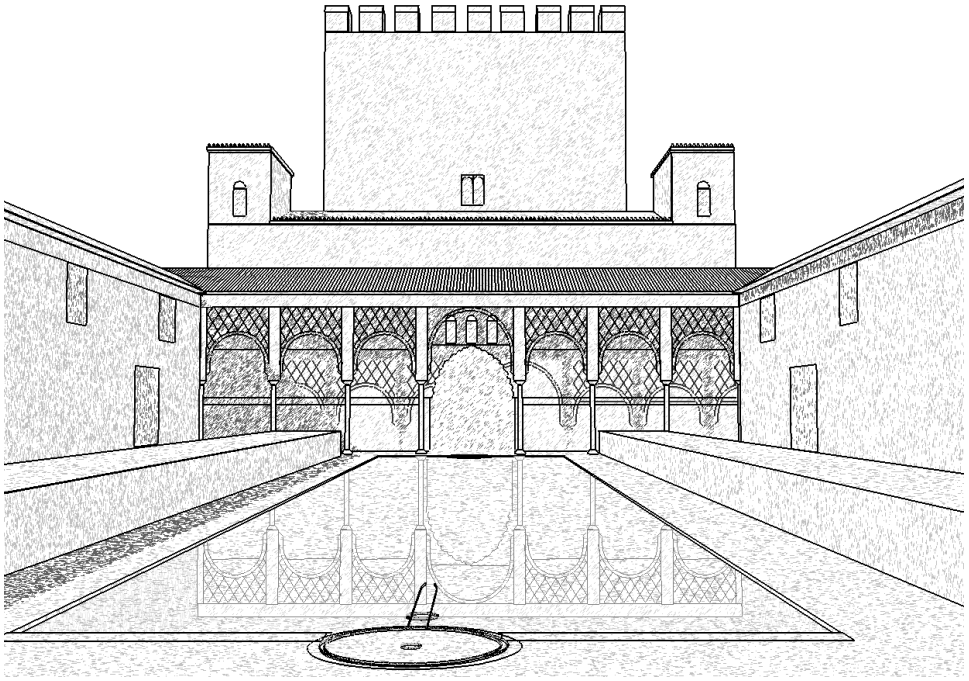
Note that, by using this shadow construction interface, it is possible to construct a scene with incomplete or even inconsistent shadows. It is the artist's responsibility to maintain the scene's integrity.

6.3 Examples

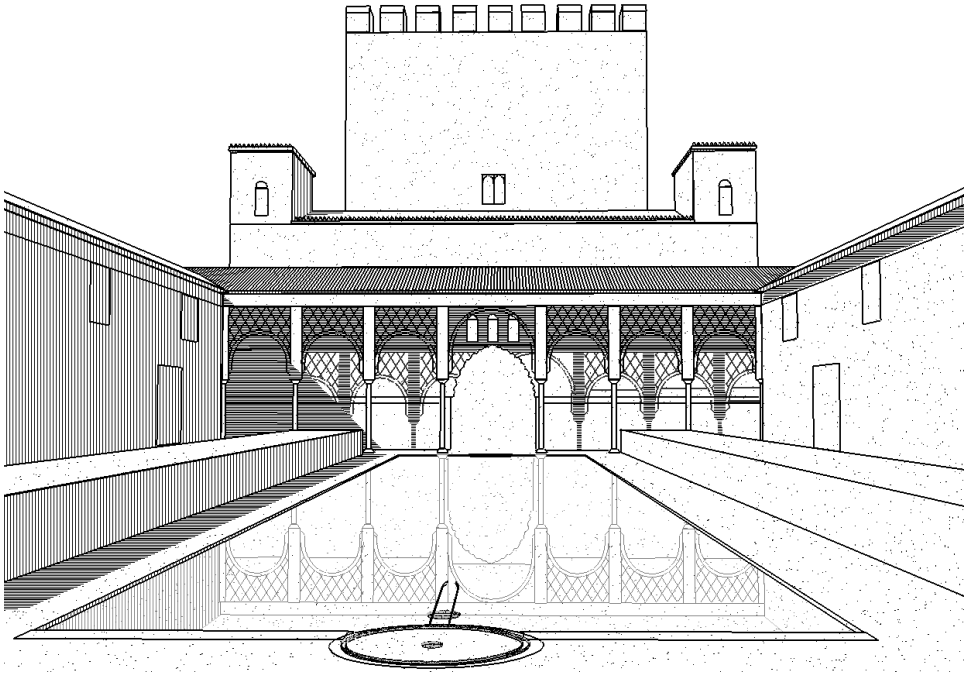
Using the system, I created a perspective drawing of the Court of the Myrtles at Alhambra Palace, Spain. It is shown in Figure 6-9-a rendered with the stippling algorithm. Figure 6-9-b shows the hatching style applied to the same drawing.

Shadows, including those cast by the colonnade and lattice onto the back wall, were projected semi-automatically. The shadow re-projection algorithm was then used to visualize the motion of the sun across the courtyard (see Figure 6-10).

In addition to shading and shadows, this example illustrates the use of many of the features of the drawing system. For example, special vanishing points aided in the drawing of the roof tiles. Symmetrical and repeating architectural features, such as the colonnade, where copied and moved using the "apparent translation" operation.



(a)



(b)

Figure 6-9: Using the system, I created this drawing depicting one of the courts at Alhambra Palace, which was then rendered using the stippling style (a) and the hatching style (b).

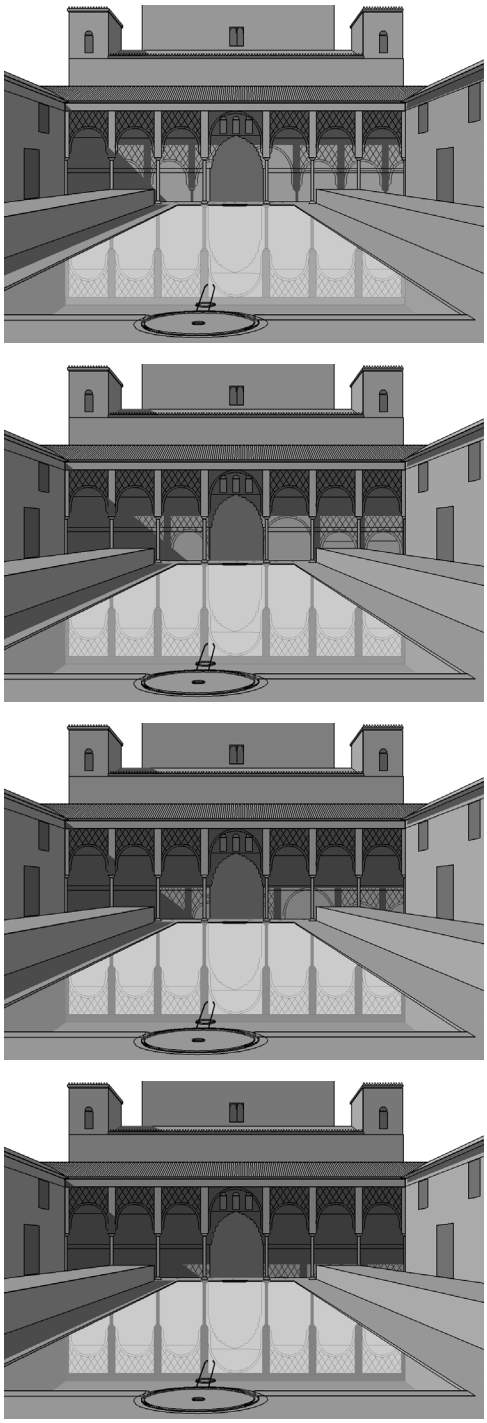


Figure 6-10: This sequence, showing the motion of the shadow across the back wall, was generated automatically.

Chapter 7

Integration with Other Media

The approach that I propose for perspective drawing allows for the integration of paper sketching and computer drawing at various points during the design process. Paper drawings often have advantages over those made with computers because of their immediacy, fluidity, and portability. On the other hand, the computer better facilitates re-projection, editing and refinement. Hence, a dual-mode approach combines the best of both worlds.

Another traditional medium that is supported by the projective drawing system is photographs and scanned drawings, which provide a quick and easy way of adding realism or visual interest to the final picture. Both traditional photographs and cylindrical panoramas can be imported into a drawing.

7.1 Paper Sketches

Often designers prefer traditional illustration media over computer-aided drawing, despite the shortcomings of these media. Therefore, designers need to sketch freely, both while using the computer and away from it. Existing tools for digitizing paper drawings, such as flatbed scanners, usually generate a *rasterized* version of the drawing that does not preserve the original strokes. While such a representation may be suitable for paint programs, it is foreign to *drawing* (stroke-based) programs. This has generally prevented designers from using paper and computer sketching inter-

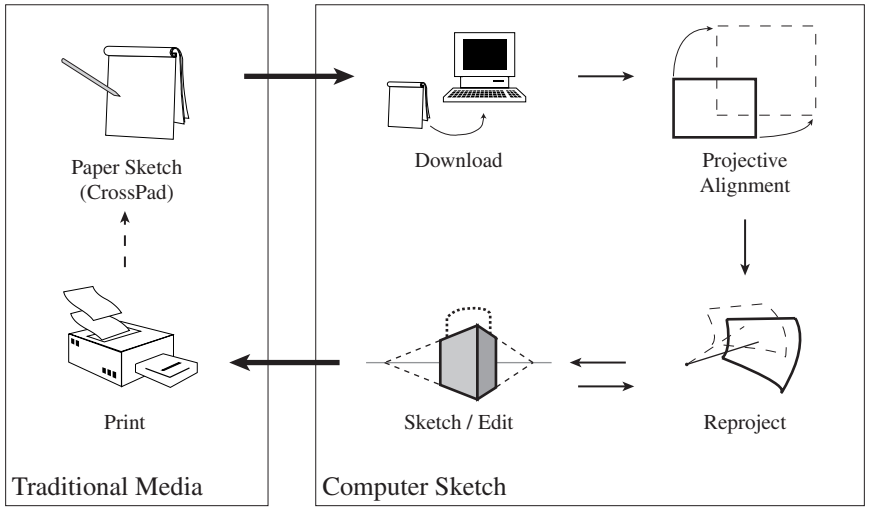


Figure 7-1: Integration of traditional media with computer sketching in a closed loop.

changeably. To address this issue, I have integrated the CrossPad portable digital notepad [6] (that records strokes while the pen deposits ink on paper) with my system. The drawing system was expanded to include perspective views drawn with this pad in a manner that permits the use of the pad for recording new sketches as well as augmenting ones created with the computer (Figure 7-1).

Since the camera rotation and field of view are unknown for a drawing imported from the pad, the system provides interactive tools for defining these attributes. A typical scenario for aligning an imported drawing starts with the user translating and rotating the input drawing (in screen coordinates) to fit the horizon, followed by panning and zooming to align the vanishing points. During this panning and zooming, the imported strokes remain static on the screen while the vanishing points and grids are animated in the background. The drawing may later be printed with a different view and new strokes added on paper, and then the import/align procedure may be repeated (Figure 7-2). In this way, the user may opt to use the pad at any point during the design stage.

The system also provides a semi-automatic tool for aligning perspective drawings containing *two* vanishing points. This tool requires the user to follow the same steps for the manual alignment, except that, instead of panning and zooming to adjust the viewing direction and field of view, the user specifies two vanishing points with

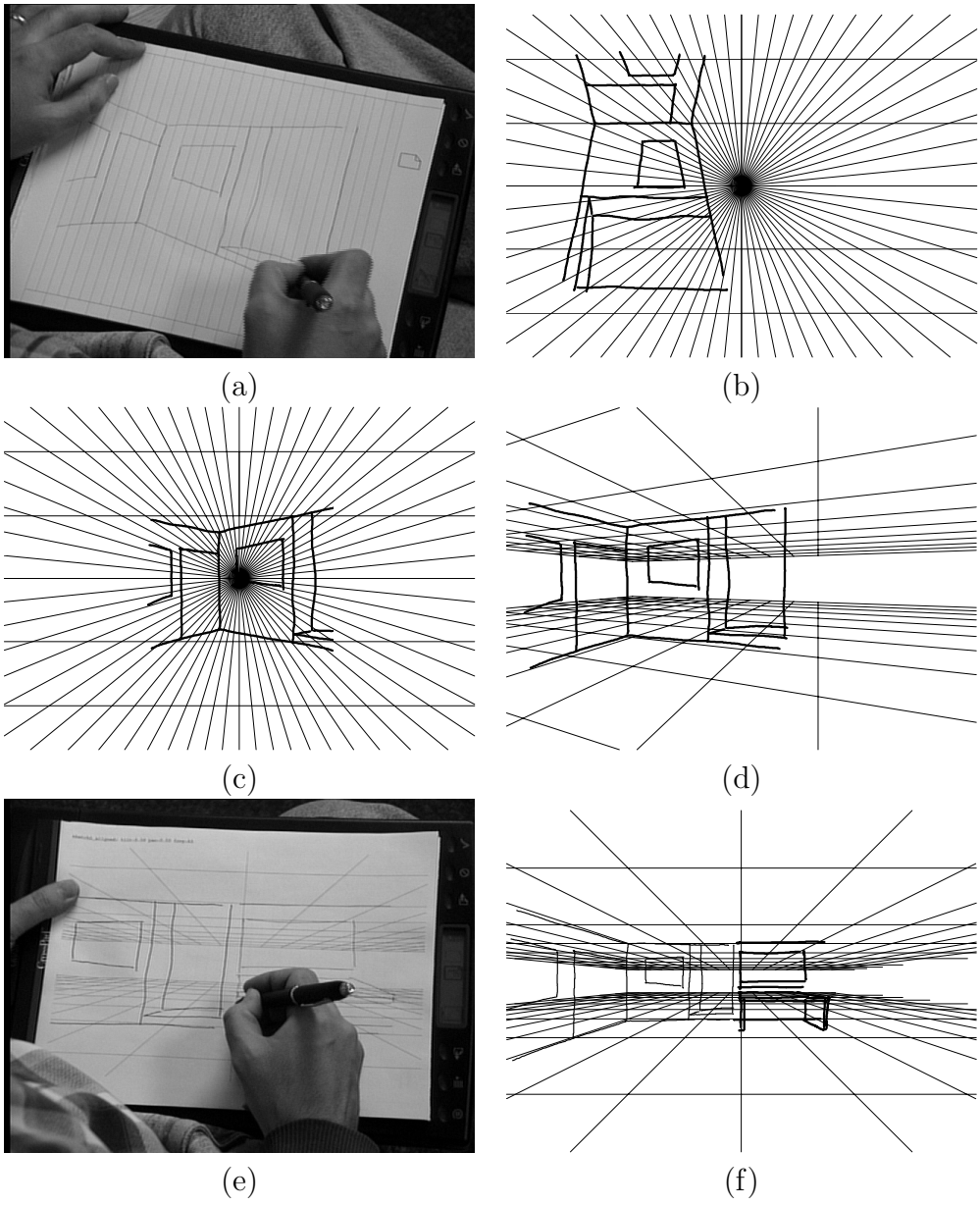


Figure 7-2: Aligning an imported sketch: A person starts sketching with the digital notepad (a), then imports the drawing into the system (b) and translates and rotates it to fit the horizon (c). Manual or automatic tools are used to align the vanishing points (d). The sketch can be printed with a different view and new strokes added on paper (e), then the import/align procedure is repeated (f).

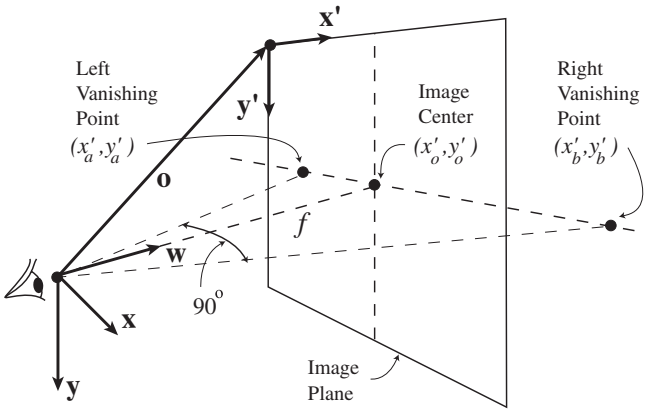


Figure 7-3: Viewing geometry for aligning two-point perspective drawings.

the pointing device: (x'_a, y'_a) and (x'_b, y'_b). The system then computes the camera rotation and focal length using the viewing geometry depicted in Figure 7-3. Since the vanishing points are assumed to correspond to two mutually orthogonal directions lying on the horizon (x-w plane), this gives us the following equation that I use to compute the focal distance f (which determines the drawing's field of view):

$$\begin{pmatrix} x_a & 0 & f \end{pmatrix} \begin{pmatrix} x_b \\ 0 \\ f \end{pmatrix} = x_a x_b + f^2 = 0.$$

By making the reasonable assumption that the axes of the image plane, $\mathbf{x}'$ and $\mathbf{y}'$, have the same directions and weights as those of the world, $\mathbf{x}$ and $\mathbf{y}$, we can compute x_a and x_b directly from user input as follows:

$$x_a = x'_a - x'_0, \quad x_b = x'_b - x'_0.$$

Note that, for a two-point perspective view, the image center (x'_0, y'_0) must lie on the line that connects the two vanishing points and fall between them.

(a) (b)

Figure 7-4: An example of a projective texture: scanned painting of a tree (a), and a black-and-white mask delineating its outline (b).

7.2 Conventional Photographs

In Section 7.1 I demonstrated the usefulness of integrating traditional drawing media with computer-based drawing. However, those techniques required the use of a digital notepad to draw on paper. I have added the capability to use scanned drawings and photographs as textures applied to perspective rectangles. Texture mapping in this fashion is achieved with projective two-dimensional mappings and image re-sampling [14]. Transparency channels are provided in the texture to alleviate the limitations of a rectilinear shape. For example, a painting of a tree can be scanned and a binary mask created using standard image editing software. The mask defines the tree as opaque, while the rest of the rectangle and holes in the tree remain transparent (see Figure 7-4). The texture and mask may then be applied to a perspective rectangle inside the system.

The user retains the ability to manipulate the rectangle as before, thereby allowing for precise placement of the textured rectangle in the scene. Thus, a row of trees can be created effectively by translating copies of the textured rectangle along the row's axis. In addition to this type of manipulation, which is reminiscent of clip art in traditional paint and drawing programs, polygons with photographic textures may undergo the projective alignment process described in Section 7.1 (by extracting vanishing points). The photograph, having been aligned in this manner, serves as a visual backdrop for drawing.

7.3 Panoramic Images

A panoramic image can be generated from site photographs using an off-the-shelf image stitching program [2]. Many special-purpose cameras are also available that record a single panoramic image. Panoramas may also be synthesized by traditional modeling and rendering programs, thereby allowing the use of my drawing system for quick design reviews and annotation.

The projective drawing system supports cylindrical panorama, which are the most commonly used type of panorama and easily generated by stitching methods. All panoramic images—cylindrical or otherwise—use projective representations similar to my drawing system. Picture elements in a panorama correspond to vision rays emanating from a single viewing position. These vision rays are equivalent to points on the unit sphere that are used by my system. Re-projection of cylindrical panoramas into planar perspective images is well-understood from the QuickTime VR system [2].

Currently, the drawing system places some restrictions on imported cylindrical panoramas. In particular, the cylinder's axis must be perpendicular to the horizon. Often, it is difficult to align the camera's axis in such a manner, thereby resulting in a tilted panorama. The drawing system can be easily extended to allow the user to specify the axis of a tilted panorama, for example by selecting two points on the horizon. Another parameter of a cylindrical panorama that must be input by the user is its vertical field of view. It is assumed that this information is known beforehand, for example from the stitching program.

7.4 Examples

Paper Sketches. The example in Figure 7-5 shows a panoramic sketch created entirely from freehand sketches originally drawn on paper. The panorama was assembled from sketches pointing at four different directions by estimating the fields of view visually.

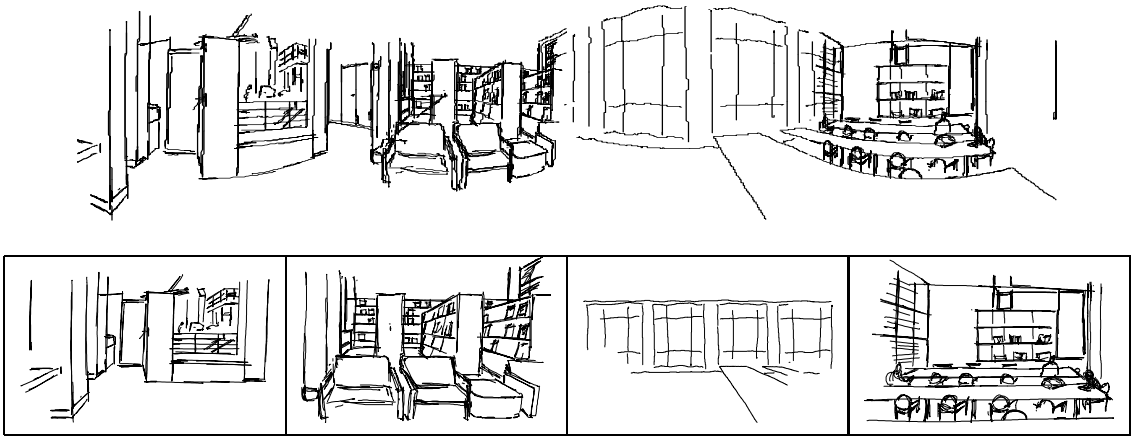


Figure 7-5: Panorama of library interior shown as an unrolled cylinder (top), and freehand sketches used to generate it (bottom).

(a) (b)

Figure 7-6: Restored elevation of the Peirene Fountain (darker lines) aligned with a drawing of existing site conditions (a) and a photographic backdrop (b).

Panoramas. This example shows the use of panoramic image backdrops and a sketch underlay. These techniques are applied to the study of a Greco-Roman fountain building in Corinth, Greece. The objective was to visualize the restored elevation of the Early Roman Period as depicted in the Corinth Series [28]. Tracing over a cylindrical panorama created a drawing of the existing conditions. The restored elevation was then imported into the system via a digitizer tablet and aligned with the previous drawing as an underlay (see Figure 7-6). In a real application, the archaeologist would use the system to study the restoration in conjunction with views of the existing conditions and resolve any conflicts that might arise.

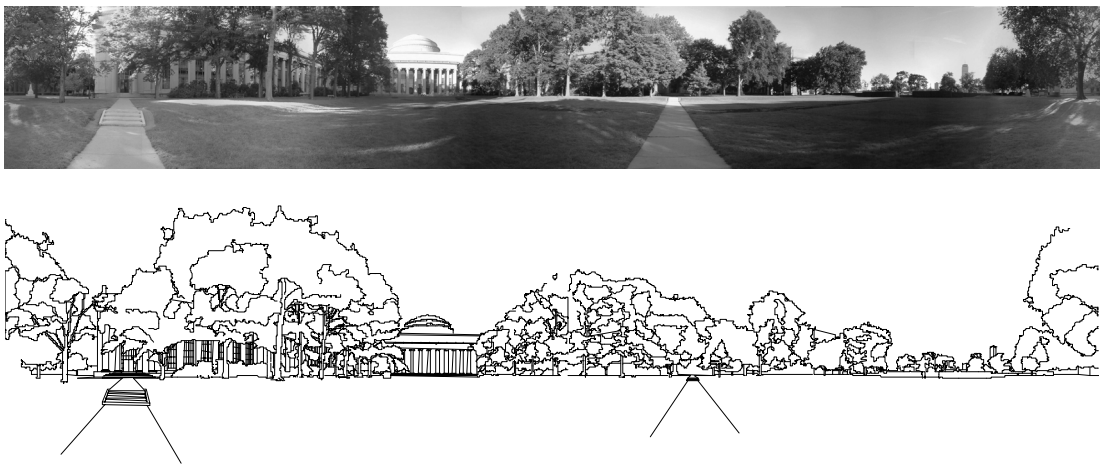
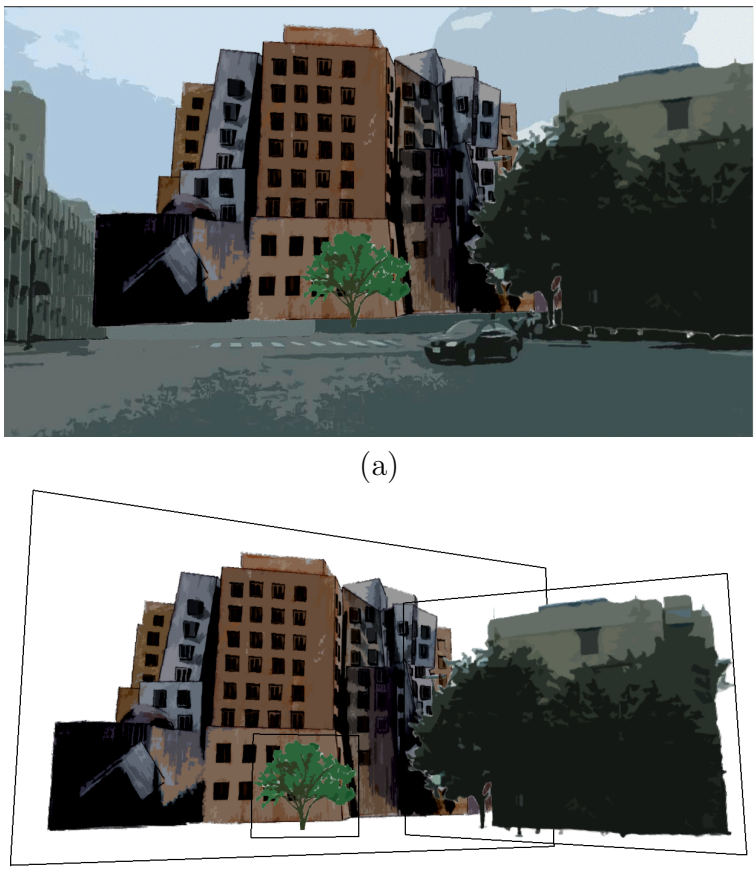


Figure 7-7: Photographic panorama of Killian Court at the M.I.T. campus (top), and artist's rendering based on the photograph (bottom).

A second example shows a panoramic sketch by a different artist. It is drawn more methodically with attention to detail, thereby exhibiting a slightly different quality than the example in Figure 7-5. The artist used a photographic panorama as a backdrop while constructing this drawing (see Figure 7-7).

Projective Textures. I created a scene using textured perspective rectangles composed against a panoramic backdrop (see Figure 7-8). The objective was to visualize a proposed architectural design by architect Frank Gehry within its context. An artist used acrylic paint to create a perspective rendering of the proposed office building. I also took a series of concentric photographs of the site from approximately the same position as the painting, and used off-the-shelf software to create a cylindrical panorama, which was further processed with image editing software to imitate the look of the painting. In the system I displayed this panorama and placed a textured rectangle containing the painting, with a mask delineating the building. I added other rectangles to depict trees, people, and foreground objects, such as objects in the real scene that occlude the proposed building. The system's tools for translating and rotating rectangles provided a flexible means for placing them.



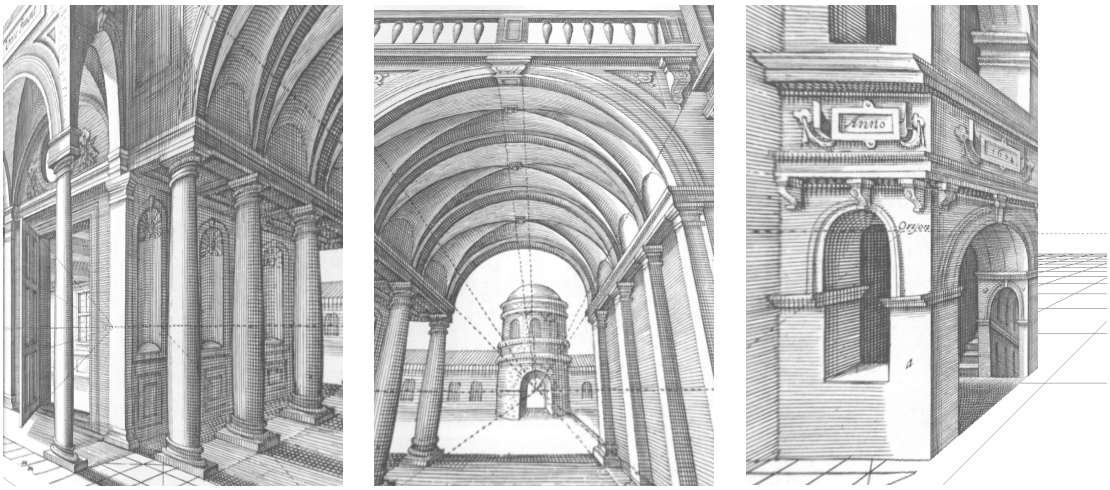
(a)

(b)

Figure 7-8: This scene depicts a proposed building within its real context (a). In addition to the panorama of the site, the scene contains three perspective rectangles with projective textures and transparency channels (b). Two of these textures include proposed elements, while the third is an existing building (the building on the right) that would occlude the proposed one.

In Figure 7-9 a classical perspective drawing by Vredeman de Vries [8] was applied as texture to a rectangle and aligned using two vanishing points. The system was then used to generate new views looking in different directions. The view in Figure 7-9-b reveals the distortion in the original drawing, which was deliberately introduced to counter the effects of a very wide field of view. The artist compressed the depth component as features grow closer to the viewer and further out into periphery vision, thus achieving a believable picture despite its larger than ninety degrees field of view.

(a)



(b) (c) (d)

Figure 7-9: This classical perspective drawing by Vredeman de Vries (a) was imported into the system and aligned using the vanishing points of the grid's diagonals. The system was then used to generate new views looking in different directions: (c) shows a view looking up into the ceiling, (d) looks frontally at the receding wall on the right-hand side of the original image, and (b) reveals the intentional distortion of the colonnade that the artist cleverly introduced to achieve convincing proportions in the wider field of view.

Chapter 8

Discussion and Future Work

I have presented a perspective drawing system that improves upon traditional perspective drawing and greatly expands the utility of traditional two-dimensional computer graphics systems. The system has the same ease-of-use as two-dimensional systems, but offers many three-dimensional-like qualities. It is intended for situations where the construction of a full fledged three-dimensional model may not be necessary. The system provides considerable advantages over a three-dimensional modeling system, including considerable time savings, while retaining the expressiveness of the original artwork.

My work also addresses the general need for better design tools to bridge the gap between the designer and computer. I approached the problem by creating a new computer drawing paradigm, based on projective two-dimensional points, and incorporating traditional two-dimensional design media. New input devices and three-dimensional traditional media, such as wood models, also need to be seamlessly integrated with computer-aided design.

8.1 User Scenarios and Experience

User Scenarios

Perspective drawings are used across many disciplines for various purposes. In some cases, such as in animation or graphic design, they become part of the final product. In others, they serve as tools for design *thinking* and *communication*. The different roles that drawings play during the design process are outlined in [22]. In particular, during a private thinking phase, the designer may draw in a manner that helps the thought process. In such drawings, solutions are suggested while unsolved aspects of the design remain vague. On the other hand, drawings that are made to communicate the design to a client are often stylized or embellished with decorations that help highlight the designer's intent. Perspective plays an important part in these two roles of design drawings because it is the only type of drawings that can convey the visual experience of the final product.

The *vagueness* of thinking drawings and *expressiveness* of communication drawings are the two main reasons for the minimal inclusion of existing computer graphics technologies into the design process. A projective drawing system, however, is posed to thrive under these circumstances. Its appeal for the early thinking process is enhanced by its integration with hand drawings and photographs, while its rendering and shading capabilities are better suited for illustrative drawings.

User Experience

Some of the examples used in this thesis were created with the help of students who were not developers of the system. They found the system's freehand and geometric primitives easy to use. However, understanding how object manipulation and shadow projection work required more experience with perspective drawing. The Alhambra example (see Figures 6-9, 6-10) was also somewhat cumbersome to construct. It took five to six hours to arrive at convincing proportions and to arrange the primitives in the drawing stack. This kind of time investment, however, is common among professional illustrators.

8.2 Comparison to Three-dimensional Modeling

In this section, I compare the projective drawing system to conventional three-dimensional Computer-Aided Design and Drafting (CADD) systems, highlighting both the benefits and limitations of the proposed approach.

Benefits of Projective Drawing

Projective drawing provides many benefits over a three-dimensional modeling system. The process of specifying a three-dimensional model containing complex geometry can be extremely time-consuming compared to drawing a single view. This process is made lengthy primarily due to the limitations of two-dimensional user interfaces. The use of a two-dimensional interface for three-dimensional modeling often means that the user has to specify the shapes and coordinates in more than one view, thereby making the process laborious. It also prevents the designer from sketching freely or with ambiguity as many designer do during the early stages of design. While three-dimensional interfaces promise to alleviate some of these limitations, they are difficult and expensive to build and they consume more space than conventional interfaces. On the other hand, projective drawing allows for artifacts that can be both expressive and quick. Drawing can also be undertaken with portable conventional media or hand-held computers.

Limitations of Projective Drawing

The most obvious limitation of projective drawing is the lack of relative depth information between objects. To some extent, this limitation is overcome by allowing the user to adjust the stacking order, as is commonly done in standard two-dimensional drawing packages. However, the lack of true depth information prevents the user from grouping objects and moving them collectively in a manner that preserves the three-dimensional illusion of the drawing. While it is possible to use the apparent three-dimensional translation or rotation tool after selecting multiple objects, the objects move independently from each other—each behaving as if the motion of the

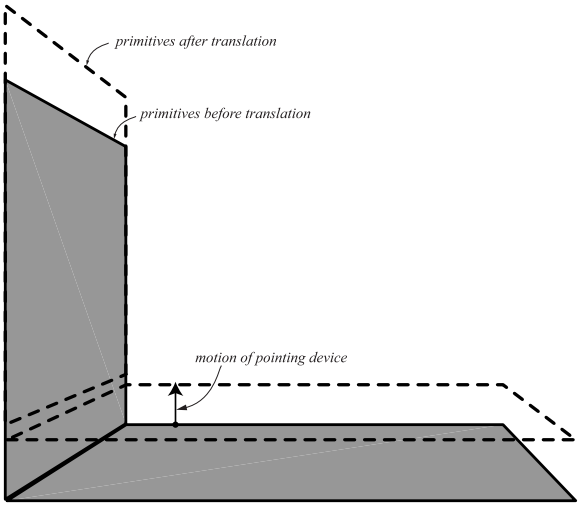


Figure 8-1: As a limitation of the system, primitives cannot be moved collectively in a convincing manner. The illusion that these two primitives form an L-shaped object is shattered as they move independently from each other.

pointing device "belonged" to it (see Figure 8-1). I use transformations of the image of a planar object that are independent of its actual distance from the viewer. In order to transform images of multiple objects, however, we need relative depth information, which the system does not support. This limitation also explains why true three-dimensional walk-throughs are not possible in this system. Furthermore, general lighting operations, such as shading with a local light source, require relative depth information.

8.3 Applications

My approach and representation have applications in different areas of computer graphics:

- Architects often generate hidden-line perspective views from three-dimensional modeling systems and embellish them by hand. My projective drawing system provides a much more flexible means of drawing within a three-dimensional environment. This can be achieved, for example, by generating a perspective

view from the CADD system, then applying the image as texture to a rectangle inside the projective system. The picture can be aligned using two vanishing points (see Section 7.1) and additional strokes drawn while viewing it in the background (see Figure 8-2).

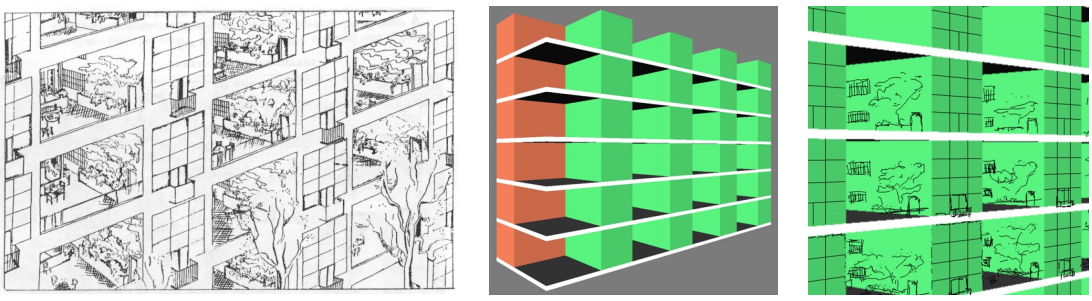
- Graphic designers frequently include perspective in their drawings using today's illustration systems that have limited support for perspective. Such systems could, for example, have a dedicated perspective mode.

- Animators commonly use paint systems to construct backdrops for cell animation. These systems are of limited use for perspective scenes. In order to simulate camera panning, multi-perspective backgrounds are sometimes used. Drawing multi-perspective views is difficult and requires a high degree of skill. Using a projective system would greatly simplify the process of drawing backdrops for cell animation.

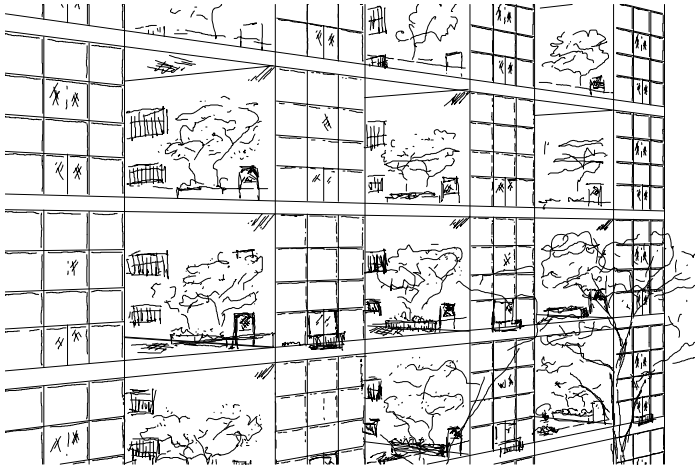
A projective drawing system can be useful to designers in various disciplines and situations. Interior designers, architects, landscape architects, and urban designers may use it for recording site conditions and sketching new design ideas and alternatives. They may also use it in design reviews, where the image backdrop is generated from a CADD program, and the drawing system is used for design critique and annotation. Archaeologists and cultural resource managers may use it for recording and annotating existing site conditions as well as exploring and communicating historical reconstruction ideas. It is also conceivable that students wishing to learn perspective drawing will use such a system to enhance their understanding of the principles of perspective. Other applications include the creation of illustrative hall-size panoramas for public exhibitions [24].

8.4 Future Work

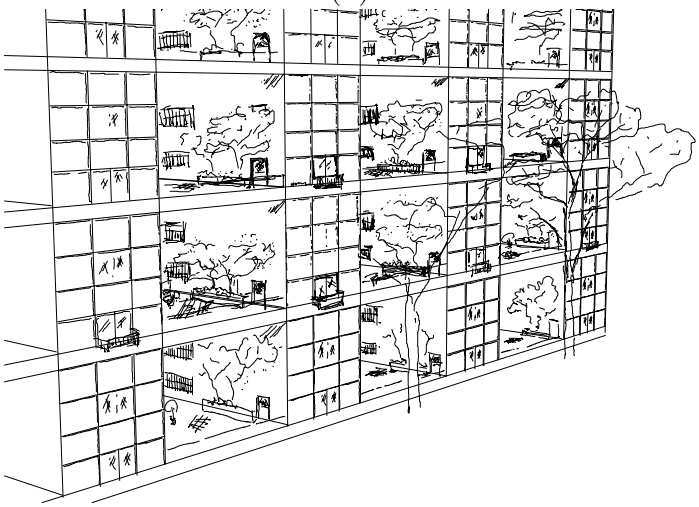
In this section I will propose extensions to the drawing systems, as well as long term research directions that are related to the work I presented in this thesis.



(a) (b) (c)



(d)



(e)

Figure 8-2: An example illustrating the use of a crude three-dimensional model as a backdrop for sketching in perspective. The basis for this example was a design by Le Corbusier [4] depicted in (a). First, a three-dimensional massing model was created using a conventional three-dimensional modeling system (b), on top of which details of the facade were added (c). The final drawing is shown while looking in two different directions (d, e).

Figure 8-3: Example drawing in the water color style.

Proposed Extensions to the System

Currently, the system has a limited set of primitives and modeling operations. The addition of new built-in primitives, such as "boxes" and "cylinders," would be helpful to the user. I also hope to embellish the system with additional modeling operations, such as the ability to generate other types of aggregate shapes.

The drawing system can also be easily extended to support layers within a drawing, as is currently common in CADD programs. The interface to these layers can evolve into a sort of digital sketchbook, with multiple pages resembling different design concepts or the refinement of a single concept. Such a sketchbook should allow tracing over previous pages and transferring strokes between pages.

I would also like to add expressiveness to the drawings by emulating the strokes and look of traditional media. The stippling and hatching rendering styles I presented are by no means the only ones applicable to perspective drawings. I could use virtually any rendering style. As more sophisticated automatic stippling or painterly rendering

algorithms become available, they can be added to the system. Alternatively, the system's freehand strokes could be stylized by mimicking different pen and brush types. This is akin to the stylized strokes of existing drawing programs that track the speed of the user's stroke and make use of pressure-sensitive drawing tablets. Such strokes require a more sophisticated rendering procedure than I currently employ. The mock drawing in Figure 8-3 was created by capturing a static view of the outdoor plaza example in my system and using a commercial program to add strokes in the "water color" style.

Further work may lead to inferring actual three-dimensional models from projective drawings, which would require additional user input to specify a primitive's distance from the viewer. This extension would link the initial sketches to other stages of the design, where three-dimensional models are beneficial. Creating different representations for the various stages of design in this manner provides an alternative to techniques that enforce a three-dimensional representation very early on [34, 18].

Graphical Input Devices

Existing technologies for input devices often limit the creative drawing process. Today's digitizer tablets are often too small and divorce the hand from the display feedback. The portable digital notepad I used, although useful for on-site drawings, lacks the ability to quickly display new views. A better device would incorporate input into a flat panel display that is significantly large and oriented like a drafting board, in a manner that allows drawing freely at an arm's length. As new devices are introduces I will extend my research to include them.

Other Projection Systems

In this thesis, I have considered only one form of planar geometric projections—the perspective (or central) projection. Other systems that also project a three-dimensional world onto a planar manifold include parallel projection systems. In some sense, parallel projection can be envisioned as the dual of central projection. Instead of a situation where the stationary viewer inspects and rotates a three-dimensional

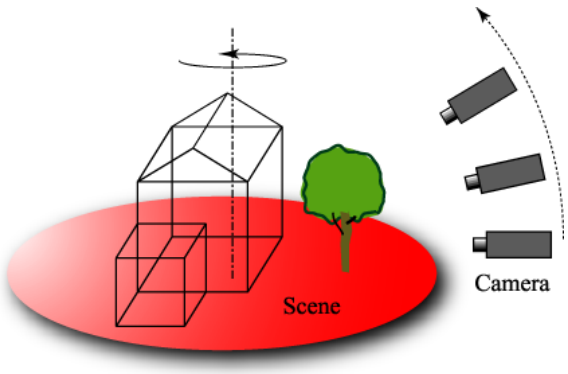


Figure 8-4: Parallel projection views that are taken with different camera positions at infinity pose a different kind of research problem than the stationary central projection camera investigated in this thesis.

scene, the viewer (at infinity) orbits about a stationary scene (see Figure 8-4). Parallel projection systems, including orthographic and oblique views [9, 33, 1], provide the primary means of design thinking and communication in architectural, industrial and engineering design. Throughout the design and construction (or manufacturing) processes, they are used either as crude sketches or precisely measured drawings.

Parallel projections pose interesting and challenging research problems. While re-projection of a single perspective view, as shown in this thesis, provides a sufficiently convincing three-dimensional illusion, this may not be true with parallel projections. At least two such views are needed to give a three-dimensional impression or generate new views that orbit the depicted scene. Thus, the challenge to the researcher is to find a new and easy way for the designer to input and manipulate parallel projection views, while maintaining the three-dimensional illusion.

Bibliography

[1] Ingrid Carlblom and Joseph Paciorek. Planar geometric projections and viewing transformations. *Computing Surveys*, 10(4):465–502, December 1978.

[2] Shenchang Eric Chen. Quicktime VR – an image-based approach to virtual environment navigation. *Proceedings of SIGGRAPH 95*, pages 29–38, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[3] Jonathan M. Cohen, John F. Hughes, and Robert C. Zeleznik. Harold: A world made of drawings. *NPAR 2000: First International Symposium on Non Photo-realistic Animation and Rendering*, pages 83–90, June 2000.

[4] Le Corbusier. *Towards a New Architecture*. Architectural Press, London, 1927. Translated by Frederick Etchells.

[5] H. S. M. Coxeter. *The Real Projective Plane*. McGraw-Hill, New York, 1949.

[6] Cross Pen Computing Group. Portable digital notepad (no longer in production).

[7] Leonardo da Vinci (1452-1519). *Leonardo on Painting: An Anthology of Writings by Leonardo da Vinci with a Selection of Documents (Trait de la peinture)*. Yale University Press, 1989.

[8] Hans Vredeman de Vries (1527-ca. 1604). *Perspective*. Dover Publications, New York, 1968.

[9] Fred Dubery and John Willats. *Perspective and Other Drawing Systems*. Herbert Press, London, 1983.

[10] Albrecht Dürer (1471-1528). *The Painter's Manual*. Abaris Books, New York, 1977. Translated by Walter L. Strauss.

[11] Olivier Faugeras. *Three-dimensional Computer Vision: A Geometric Viewpoint*. Artificial Intelligence. MIT Press, Cambridge, Mass., 1993. ISBN 0262061589.

[12] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics, Principles and Practice*. Addison-Wesley, second edition, 1990.

[13] Robert W. Gill. *Creative Perspective*. Thames and Hudson, London, 1975.

[14] Paul S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics & Applications*, 6(11):56–67, November 1986.

[15] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. *Proceedings of SIGGRAPH 98*, pages 453–460, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[16] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. *Proceedings of SIGGRAPH 2000*, pages 517–526, July 2000. ISBN 1-58113-208-5.

[17] S.C. Hsu, I.H.H. Lee, and N.E. Wiseman. Skeletal strokes. In *UIST 93 Proceedings of the ACM SIGGRAPH & SIGCHI Symposium on User Interface Software & Technology*, November 1993.

[18] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3D freeform design. *Proceedings of SIGGRAPH 99*, pages 409–416, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[19] Kenichi Kanatani. Computational projective geometry. *CVGIP: Image Understanding*, 54(3):333–348, 1991.

[20] Martin Kemp. *The Science of Art*. Yale University Press, 1990.

[21] John Lansdown and Simon Schofield. Expressive rendering: A review of nonphotorealistic techniques. *IEEE Computer Graphics & Applications*, 15(3):29–37, May 1995.

[22] Bryan Lawson. *How Designers Think*. Butterworth Architecture, London, 1990.

[23] Martin E. Newell, R. G. Newell, and T. L. Sancha. A solution to the hidden surface problem. *Proceedings of the ACM National Meeting*, 1972.

[24] Stephan Oettermann. *The Panorama: History of a Mass Medium*. Zone Books, New York, 1997. Translated by Deborah Lucas Schneider.

[25] Ramesh Raskar and Michael Cohen. Image precision silhouette edges. *1999 ACM Symposium on Interactive 3D Graphics*, pages 135–140, April 1999. ISBN 1-58113-082-1.

[26] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin. Interactive pen-and-ink illustration. *Proceedings of SIGGRAPH 94*, pages 101–108, July 1994. ISBN 0-89791-667-0. Held in Orlando, Florida.

[27] J. G. Semple and G. T. Kneebone. *Algebraic Projective Geometry*. Oxford University Press, London, 1952.

[28] Richard Stillwell. *Architecture*, volume 1, part 2 of *Corinth*. Harvard University Press, Cambridge, Mass., 1941. Published for the American School of Classical Studies at Athens.

[29] Jorge Stolfi. *Oriented Projective Geometry: a Framework for Geometric Computations*. Academic Press, Boston, 1991.

[30] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, Wellesley, MA, 1993.

[31] Osama Tolba, Julie Dorsey, and Leonard McMillan. Sketching with projective 2D strokes. *CHI Letters*, 1(1):149–157, November 1999. Proceedings of the 12th Annual ACM Symposium on User Interface Software & Technology (UIST 99).

[32] Osama Tolba, Julie Dorsey, and Leonard McMillan. A projective drawing system. In Stephen Spencer, editor, *2001 Symposium on Interactive 3D Graphics*, pages 25–34, Research Triangle Park, NC, March 2001.

[33] John Willats. *Art and Representation.* Princeton University Press, 1997.

[34] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. SKETCH: An interface for sketching 3D scenes. *Proceedings of SIGGRAPH 96*, pages 163–170, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.