A Transparently Scalable Visualization Architecture for Exploring the Universe

(final version)

Chi-Wing Fu [†] Andrew J. Hanson [‡]

cwfu@cs.ust.hk hanson@cs.indiana.edu

[†] The Hong Kong University of Science and Technology

[‡] Indiana University, Bloomington, Indiana 47405, USA

Chi-Wing Fu (* Corresponding author)

Dept. of Computer Science and Engineering,

The Hong Kong University of Science & Technology, Clear Water Bay, Hong Kong.

Tel: +852-2358-6991 Fax: +852-2358-1477 Email: cwfu@cs.ust.hk

Andrew J. Hanson

Computer Science Dept., Indiana University, Bloomington, Indiana, 47405, USA.

Tel: +1-812-855-5855 Fax: +1-812-855-4829 Email: hanson@cs.indiana.edu

A Transparently Scalable Visualization Architecture for Exploring the Universe

Abstract

Modern astronomical instruments produce enormous amounts of three-dimensional data describing the physical Universe. The currently available data sets range from the solar system to nearby stars and portions of the Milky Way Galaxy, including the interstellar medium and some extrasolar planets, and extend out to include galaxies billions of light years away. Because of its gigantic scale and the fact that it is dominated by empty space, modeling and rendering the Universe is very different from modeling and rendering ordinary three-dimensional virtual worlds at human scales. Our purpose is to introduce a comprehensive approach to an architecture solving this visualization problem that encompasses the entire Universe while seeking to be as scale-neutral as possible. One key element is the representation of model-rendering procedures using power scaled coordinates (PSC), along with various PSC-based techniques that we have devised to generalize and optimize the conventional graphics framework to the scale domains of astronomical visualization. Employing this architecture, we have developed an assortment of scale-independent modeling and rendering methods for a large variety of astronomical models, and have demonstrated scale-insensitive interactive visualizations of the physical Universe covering scales ranging from human scale to the Earth, to the solar system, to the Milky Way Galaxy, and to the entire observable Universe.

Keywords

I.6.9.g Visualization techniques and methodologies, I.6.9.f Visualization systems and software, J.2.c Astronomy, I.3.7.g Virtual reality

1 INTRODUCTION

Rapid advances in astronomical measurement technology and supercomputer support of theoretical modeling have made extraordinary amounts of data available to the astronomy community in recent decades. While accurate three-dimensional spatial data were once limited to the solar system, we now have spatial data for vast numbers of nearby stars (see, e.g., the Bright Star Catalogue [23] and the Hipparcos and Tycho Catalogues [18]). Significant progress has been made on probing the structure of the interstellar medium, including molecular clouds [14]

and other diffuse objects such as supernova remnants. Multispectral data have enabled the construction of vastly improved models of our own Milky Way galaxy, while the refinement and correction of redshift data have enabled the collection of enormous data sets of 3D galaxy positions (see, e.g., Tully [42] and the Sloan Digital Sky Survey [2, 41]). Even such difficult systems as extrasolar planets [36] now have specific orbits available. These data sets give us three-dimensional information for a wide variety of astronomical objects, and, altogether, they provide the raw data for a three-dimensional virtual Universe that one can in principle explore in detail through large-scale visualization techniques.

Research on large-scale visualization has traditionally focused on the processing and organization of data, and has targeted visualization solutions that maximize the amount of data one can access. While these are indeed very challenging problem domains that push the limits of graphics hardware and stimulate innovation in visualization techniques, the physical Universe poses its own special challenges: *We not only have to deal with a large amount of data, we also have to deal with the gigantic spatial scale of the physical Universe*. Thus, instead of focusing on a particular kind of data, this paper deals with the unique aspects of large spatial scale in visualizing the physical Universe, and introduces a scale-neutral visualization architecture as a powerful tool for modeling, rendering, navigation, and exploration in this gigantic space.

This work was motivated by the concept of constructing an interactive implementation of the framework suggested by the film, "Powers of Ten" [10], by Charles and Ray Eames. In order to produce a virtual journey across the scales of the Universe, a selection of which are represented in Table I, this film exploited the powers-of-ten idea suggested by the book *Cosmic View: The Universe in Forty Jumps* by Kees Boeke [9]. The Eames film was itself annotated in a companion book "Powers of Ten" [31], by Philip Morrison and Phyllis Morrison.

TABLE I
BASE 10 LOGARITHMS OF SCALES OF TYPICAL OBJECTS IN THE PHYSICAL UNIVERSE IN UNITS OF METERS

Typical objects	**Powers of 10 (meters)**
Observable Universe (Quasars, etc.)	27
Super-clusters	25
Clusters of galaxies	24
Size of Virgo cluster	23
Distance to Andromeda galaxy	22
Milky Way diameter	21
Distance to Orion arm	19
Distance to the nearest stars	17
Size of the solar system	13
Venus, Earth, and Mars	11
Earth-Moon distance	9
Earth diameter	7
San Francisco	4
Human scale	0
Micro-Organisms / Hair Thickness	-4
Size of a red blood cell	-5
DNA Structure	-8
Carbon Nucleus	-14
Quarks	-16
Planck length	-35

Our challenge is thus to adapt the realities of computer architecture and graphics systems to the task of a continuously scalable visualization of the 3D Universe. This paper extends, expands upon, and fills in the essential details outlined in our initial work [22] to achieve a more complete and comprehensive framework, including the following principal contributions:

- A unified framework for modeling large spatial scale that permits us to effectively represent and transform positions and vectors, and to render models at any arbitrary scale within the range of physical objects in Table I.
- An assortment of scale-motivated techniques, including the PSC transformation, the depth rescaling method, environment caching, and object disappearance criteria.
- Implementation of scale-independent modeling and rendering of various astronomical bodies, e.g., stars, galaxies, etc., permitting us to properly render them across huge viewing scales.

1.1 Related Work

Among the pioneering applications of computer graphics to the study of outer space as well as basic science are Blinn's set of animations: *"Voyager Fly-by Animations"* [5] (1977–87), *"COSMOS"* [6] (1979–80), and *"The Mechanical Universe"* [7] (1983–86). These animations simulate the Voyager 2 fly-by of Jupiter and the Pioneer 11 fly-by of Saturn, and employ view positioning methods described by Blinn in [8]. Another influential astronomical simulation was Whitehouse's physical simulation of the head-on collision between the Comet Shoemaker-Levy 9 and Jupiter [48]. Stytz et al. [40], among others, built a comprehensive solar system modeler with accurate planetary motions. Ostriker and Norman [35] at NCSA proposed a framework for simulating cosmology, and reviewed the related requirements in high performance computing, file management, and visualization systems. Commercially-oriented animations such as *Cosmic Voyage* [33] were produced by the NCSA group for IMAX audiences. Genetti and Nadeau et al. [19, 32] simulated a fly-through of a 3D model of the Orion Nebula using volume rendering [50]. This material was incorporated in both the Hayden Planetarium show "Passport to the Universe" and the animation "Volume Visualization of the Orion Nebula [17]." Kahler et al. [27] at NCSA used a supercomputer and AMR (adaptive mesh rendering) methods to simulate the life-span of a star, while Turnage [43] presented a physical simulation of a supernova explosion and its shock wave. Among other interesting contributions to the field are those of Hopf et al. [25], who developed a PCA-based splatting technique for rendering dynamic point-based data, Baranoski et al. [4], who proposed a rendering method for simulating the Aurora Borealis (the Northern Lights), Jensen et al. [26], who devised a physically-based model to render the night sky as seen from Earth, and Magnor et al. [30], who developed an interactive visualization tool for rendering arbitrary dust distributions around a central illuminating star.

1.2 Related Software

In the late 1980s, due to the limited computational power of hardware, astronomy simulation software was generally restricted to 2D star chart programs or simple solar system modelers. With the rapid advances in gaming-motivated graphics acceleration, more software became available to the general public such as Starry Night, Digital Universe, and Voyager. Although some of these do provide a certain level of 3D navigation in the solar system and nearby stars, the features tend to be focused on supporting 2D star charts for amateur stargazing. In recent years, some popular open-source projects such as Celestia [28], Stellarium [11], and Open Universe [3] have been developed with higher-level VR capabilities. Under appropriate conditions, one can detect noticeable discrete motion errors in nearby objects at large scales, indicating the fact that the handling of spatial scales does not consistently avoid precision errors in large-scale cosmological space. While a selection of problems can be handled using extended precision arithmetic, which enables Celestia, for example, to support an exponential zoom feature to visit nearby galaxies, our framework addresses additional important issues such as adapting to the finite precision of a given depth buffer, dealing with the compression of the depth buffer ranges to unusable overlapping layers, and the selective exploitation of commodity hardware acceleration.

In the movie industry, Loren Carpenter developed the Star Renderer, which was then integrated into Pixar's commercial particle renderer called Starman and used in the IMAX film "Cosmic Voyage [37]." Welling [46] of the Pittsburgh Supercomputing Center implemented his own star rendering method, which was then used to create animations for astrophysical particle simulation data. Stuart Levy at NCSA developed the "partiview" visualization system [29], which provides real-time navigation of astrophysical environments across interstellar / intergalactic scale, and is incorporated into various simulation projects at NCSA and the Hayden

planetarium. Furthermore, Cox et al. at NCSA developed the Virtual Director software [13] as a CAVE™ tool for constructing navigation paths to be used for animation design.

1.3 Problems with Large Spatial Scale

Three-dimensional astronomical visualization systems that go beyond the basic task of displaying the night sky as seen from Earth and reach out to cosmological distances suffer from a wide spectrum of precision problems. Due to floating point precision limits, a variety of features such as coordinate representation, matrix transformations, and depth buffering may have unreliable behavior; thus, large-scale, three-dimensional visualization systems typically preserve computational stability by limiting the navigation range. One simple solution is to create one specific virtual environment for each specific visualization range, e.g., the solar system, interstellar space, and intergalactic space. Such a system can then switch the virtual environment "by hand" when the user makes a transition between these scales along a given 3D path.

Other implementations and animations ignore the physical scale, and artificially adjust the spatial scale in the virtual space by making the astronomical bodies arbitrarily closer or bigger so that everything becomes renderable at a convenient (but physically imprecise) scale. These approaches are sufficient for conventional planetarium presentations and astronomy animations that are based entirely on pre-rendered material. However, with the growing demand for real-time interactive control and exact scientific accuracy in visualization applications, the need for comprehensive approaches that handle all scales in a single context has become apparent.

1.4 Overview

The scalable visualization architecture that we have designed consists of three layers, as depicted in Figure 1. The bottom layer is the data representation layer, which employs the *power*

scaled coordinates (PSC) to represent coordinates and vectors in an exponentially-scaled form with a unit-sized base scale for each exponent. The middle layer provides four inter-related techniques for rendering objects defined across extreme spatial scales. This layer is independent of specific features of astronomy and deals with the abstraction of efficient scale-independent rendering. On top of these techniques, we have the virtual astronomy layer responsible for modeling and rendering different kinds of astronomical bodies.

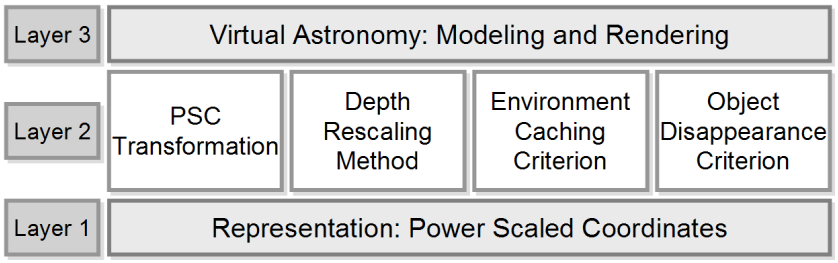


Fig. 1. The Transparently Scalable Visualization Architecture.

2 PSC Representation and Transformation

2.1 Power Scaled Coordinates

In order to handle the demands of scenes with extreme spatial scales, we represent spatial quantities such as coordinates and vectors using logarithmic scaling methods. We refer to our data representation *power scaled coordinates* (PSC); we first began working with this method in the Siggraph 2000 animation "Cosmic Clock [21]" and in our related earlier work [22] addressing the problems encountered. Power scaled coordinates are composed of a four-tuple,

$$(x, y, z, s) \textit{ representing the } 3D \textit{ position } (x, y, z) \times k^s$$

where k is any positive exponent base, usually chosen as 10. Based on this construction, the PSC representation nicely decouples the *directional term* (x, y, z) and the *exponential scale term* s from a given 3D coordinate so that we can effectively represent positions of objects at either galactic or sub-atomic scales in a uniform fashion. Note that a power scaled coordinate is said

to be *normalized* if its *directional term* is a unit vector. Furthermore, there is a strong duality between PSC and homogeneous coordinates that led us to refer to it as "power homogeneous coordinates" in our earlier work. Just as standard computer graphics practice reduces a homogeneous coordinate defined by the equivalence $X = (x, y, z, w) \equiv (\lambda x, \lambda y, \lambda z, \lambda w)$ to a standard (inhomogeneous) coordinate by choosing $\lambda = 1/w$, we use the equivalence

$$(x, y, z, s) \equiv (\ \lambda x\ ,\ \lambda y\ ,\ \lambda z\ ,\ s - \log_k \lambda\ ) \equiv (\ \lambda x\ ,\ \lambda y\ ,\ \lambda z\ ,\ 0\ )$$

to reduce a PSC to a standard (inhomogeneous) coordinate by choosing $\lambda = k^s$, so that making $s = 0$ has a strong parallel to setting $w = 1$ in homogeneous coordinates.

The PSC representation is also similar in spirit to Ward's RGBE pixel intensity format [45]; RGBE encodes high dynamic range images by using the exponential term E while PSC represents high dynamic range spatial data by using the exponential term s. We note also that hardware shader programming can be exploited within a standard graphics system to express PSC's directly through the 4th coordinate component w, for example, as supplied by the OpenGL vertex subroutine `glVertex4`. We will illustrate this approach step by step in subsequent sections.

2.2 The PSC Transformation

By using power scaled coordinates, we can represent spatial data in scale-independent form. We now extend this idea down to the low-level coordinate transformation environment of the graphics system so that we can support scale-independent processing of three-dimensional vertices in the graphics pipeline. In the standard OpenGL graphics rendering pipeline, transformations are implemented by matrix multiplication; all matrices, coordinates, and vectors (both input and internal) involved in the usual transformation process are phrased in terms of homogeneous 4-vectors. To exploit shader programming concepts, we implement the PSC transforma-

tion method at the GPU level, typically in the vertex program, so that PSC replaces the standard homogeneous coordinates as the vertex representation. In other words, we accept vertices in the *PSC-in and PSC-out* format in the modelview transformation process.

In general, the standard modelview matrix is a 4-by-4 matrix, usually decomposable into two parts: a 3-by-3 submatrix in the top-left corner, say A, and a 3-by-1 vector in the last column, say $\vec{T}$. Normally, the matrix A is responsible for scaling and rotation while the vector $\vec{T}$ is for translation. The last row is usually $(0, 0, 0, 1)$ before the application of the projection matrix (frustum). Given a 3D world coordinate $\vec{v}_{\text{world}}$, the modelview transformation can be written as:

$$\begin{pmatrix} A & \vec{T} \\ 0 & 1 \end{pmatrix} \left(\vec{v}_{\text{world}} \, , \; 1 \right)^t \;=\; \left(A \, (\vec{v}_{\text{world}})^t + \vec{T} \, , \; 1 \right)^t .$$

Here, $\vec{v}_{\text{world}}$ is first rotated and scaled by A, and then translated by $\vec{T}$. If $\vec{v}_{\text{world}}$ is expressed as a power scaled coordinate, say $\vec{\mathbf{v}}_{\text{world}} = (v_x, v_y, v_z, v_s)$, using an ordinary 3D vector for $\vec{T}$ could be insufficient because $\vec{\mathbf{v}}_{\text{world}}$ could be arbitrarily large or small while $\vec{T}$ always has a fixed scale. We adopt the notation that PSC's are bold-face vectors, and extend the modelview matrix framework for dynamic spatial transformations:

> *We attach an exponential term t_s to the modelview matrix so that $\vec{T}$ becomes a power scaled vector, say $\vec{\mathbf{T}} = (t_x, t_y, t_z, t_s)$.*

Therefore, we can re-write the modelview transformation expression above by using PSC's; if we express the result in three-dimensional vector form, we have

$$A \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} k^{v_s} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} k^{t_s} \quad .$$

It is worth noting that the multiplication between $\vec{\mathbf{v}}_{\text{world}}$ and A is independent of the exponential term v_s and is just an ordinary matrix-vector multiplication; by normalizing the multiplication result and re-adjusting v_s, we will have a normalized PSC, say $\vec{\mathbf{P}} = (p_x, p_y, p_z, p_s)$, for the left hand side of the expression. Furthermore, to ensure stability under addition, we take

$$\vec{\mathbf{P}} + \vec{\mathbf{T}} \;=\; \begin{cases} (\, p_x k^{-\delta s} + t_x \,,\; p_y k^{-\delta s} + t_y \,,\; p_z k^{-\delta s} + t_z \,,\; t_s \,) & \text{if } \delta \text{s} \geq 0 \\ (\, p_x + t_x k^{\delta s} \,,\; p_y + t_y k^{\delta s} \,,\; p_z + t_z k^{\delta s} \,,\; p_s \,) & \text{otherwise} \,, \end{cases}$$

where $\delta s = t_s - p_s$ (provided $\vec{\mathbf{T}}$ has been normalized).

In fact, the t_s term attached to the modelview matrix works in *exactly the same way* as the exponential scaling factor in a "Powers-of-Ten" context. *Adjusting t_s can exponentially scale the physical data space (world space) against the virtual camera space (eye space).* The camera setting is unaffected during the change. In our framework, by putting this scaling factor into the context of the modelview matrix, we precisely formulate a mathematical model for the powers-of-ten transformation in 3D. The camera always utilizes a fixed order-unity scale while the data are scaled relative to the camera space. In addition, the reader may notice that we could in principle adjoin three exponential terms to the scaling part of the matrix; however, because we seldom have non-uniform scaling in astrophysical models, these scaling terms would be of little utility; t_s will suffice, and also serve to support computational efficiency in the shader program.

3 The Depth Rescaling Method

3.1 The Depth Rescaling Model

To address possible precision problems in the projection transform, we introduce the *depth rescaling method* for projecting objects across extreme scales. This method is again PSC-based. The idea of depth rescaling is as follows: given an object far beyond the far plane, if we shrink

all its vertices along their lines of sight towards the eye point, its effective size will decrease and it will become viewable at the camera without explicitly extending the distance of the far plane. Since all vertices are distorted in the same manner along their lines of sight, we will not notice any difference at the camera location because the camera is fixed at the eye-space origin, where all lines of sight intersect. We call this scaling technique the *depth rescaling method*, and the underlying geometric model for distorting eye-space coordinates the *depth rescaling model*. In addition to the near and far planes provided by the standard projection model, the depth rescaling model introduces the *near cutoff plane* and the *far cutoff plane* as shown in Figure 2.

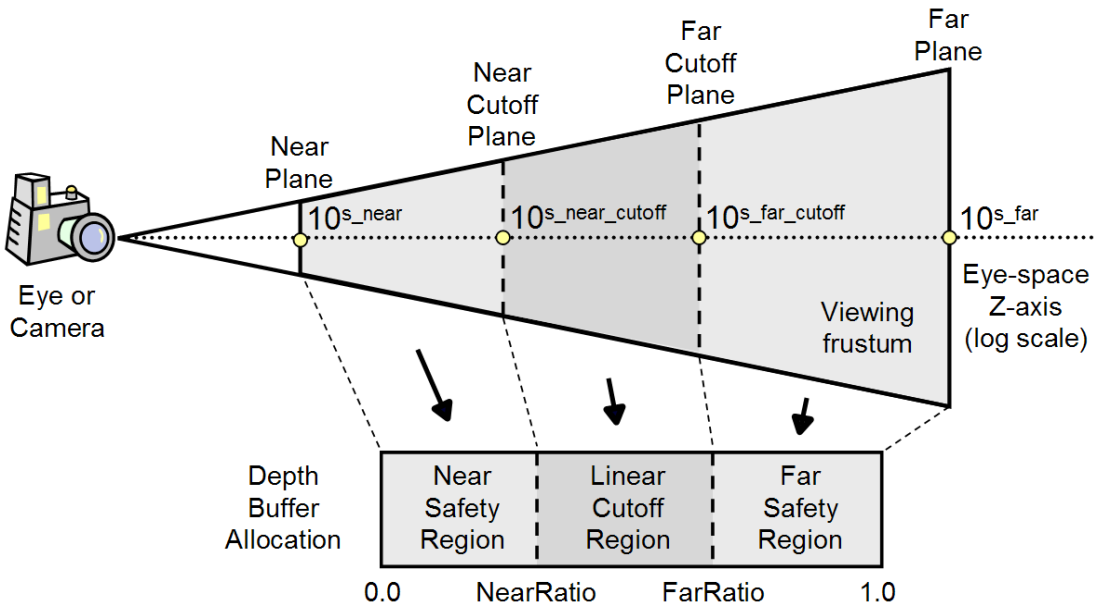


Fig. 2. The Depth Rescaling Model.

This projection model is motivated by the well-known fact that the depth buffer utilization is strongly influenced by the ratio between near-far-plane distances. To circumvent this, we segment the ordinary viewing frustum into three sub-regions and map them onto the depth buffer individually. For the two outer regions, we map eye-space coordinates logarithmically to the front and rear parts of the depth buffer. These two regions are called the *near safety region* and the *far safety region* because they allow reliable depth buffering for objects too near or too

far from the eye point to be handled by the near and far clipping planes without significant loss of depth precision. On the other hand, we map the middle region, the *linear cutoff region* comprising the principal domain of attention, to the middle part of the depth buffer either linearly or using standard projection models. The positions of the near and far planes as well as the two additional cutoff planes are intrinsically defined in logarithmic scale, that is, $s_{\text{near}} \leq s_{\text{near\_cutoff}} \leq s_{\text{far\_cutoff}} \leq s_{\text{far}}$, and distances are mapped to the depth buffer according to the heuristic utilization ratios: $0 \leq \mathit{nearRatio} \leq \mathit{farRatio} \leq 1$. Note that for cosmic-scale exploration, we may turn off the near safety region so as to optimize the depth buffering and the shader computation; for microscopic worlds, we might have to do it the other way around.

3.2 Mathematics Details of Depth Rescaling

The following mathematical procedures implement the depth rescaling process:

1) *Z-Normalization* The end result of the PSC modelview transformation is an eye-space power scaled coordinate, $\vec{\mathbf{v}}_{\text{eye}}$. In order to know in which region $\vec{\mathbf{v}}_{\text{eye}}$ is located, we normalize it and compare its log distance from the eye-space origin to the log distances to each separating plane given in the depth rescaling model. Since the distances to these separating planes are measured along the eye-space z-axis, we do not apply the conventional PSC normalization: instead, we normalize $\vec{\mathbf{v}}_{\text{eye}} = (x_{\text{eye}}, y_{\text{eye}}, z_{\text{eye}}, s_{\text{eye}})$ to $\vec{\mathbf{v}}'_{\text{eye}} = (x'_{\text{eye}}, y'_{\text{eye}}, z'_{\text{eye}}, s'_{\text{eye}})$ so that

$$\begin{cases} z'_{\text{eye}} = \pm 1 & \text{if } z_{\text{eye}} \neq 0, \\ \max(\,|x'_{\text{eye}}|\,,\,|y'_{\text{eye}}|\,) = +1 & \text{if } z_{\text{eye}} = 0, \text{ and either } x_{\text{eye}} \neq 0 \text{ or } y_{\text{eye}} \neq 0, \\ s'_{\text{eye}} = -\infty \text{ (or } -\texttt{FLT\_MAX}) & \text{if } x_{\text{eye}} = y_{\text{eye}} = z_{\text{eye}} = 0\ . \end{cases}$$

We refer to this specialized type of normalization as *z-normalization*, where z'_{eye} is always +1, -1, or 0; thus, the z-distance of $\vec{\mathbf{v}}_{\text{eye}}$ from the origin depends solely on s'_{eye}. Although it seems redundant to z-normalize PSC's when $z_{\text{eye}} = 0$ or z_{eye} is in front of the near plane, it is necessary

to guarantee proper homogeneous clipping by the graphics hardware.

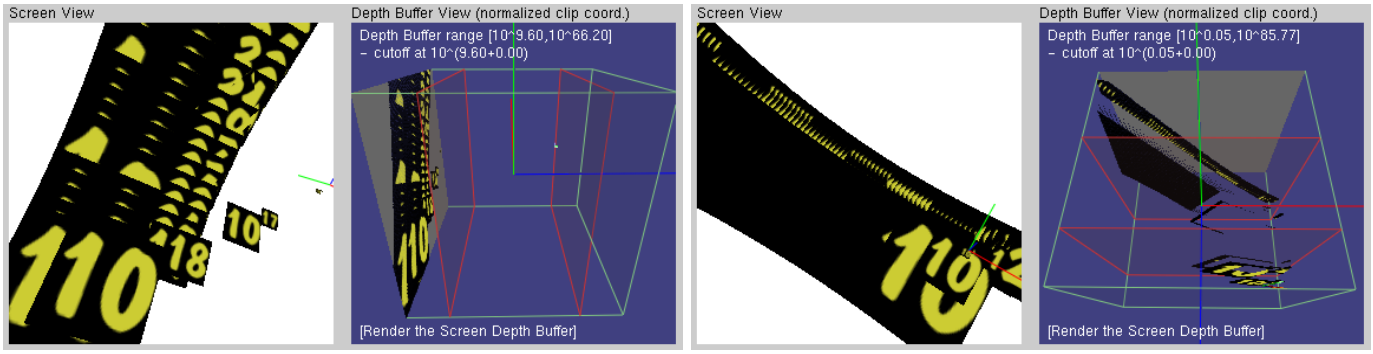
2) *Mapping to the Depth Buffer (z_{NDC})* After obtaining s'_{eye} by z-normalization, it is straightforward to map s'_{eye} to a depth buffer value (or to the normalized device coordinates (NDC), say $z_{\text{NDC}} \in [-1, 1]$) based on the depth rescaling model depicted in Figure 2.

- If $s'_{\text{eye}} \in [s_{\text{near\_cutoff}}, s_{\text{far\_cutoff}}]$, z_{NDC} is found by linearly interpolating $10^{s'_{\text{eye}}}$ in $[10^{s_{\text{near\_cutoff}}}, 10^{s_{\text{far\_cutoff}}}]$.
- Otherwise, we linearly interpolate s'_{eye} in the range $[s_{\text{near}}, s_{\text{near\_cutoff}}]$ or $[s_{\text{far\_cutoff}}, s_{\text{far}}]$ instead.

The mapping from s'_{eye} to z_{NDC} is continuous and strictly increasing even at boundaries between sub-regions. Thus, we can preserve depth ordering of eye-space coordinates after mapping to the depth buffer, ensuring correct visibility resolution.

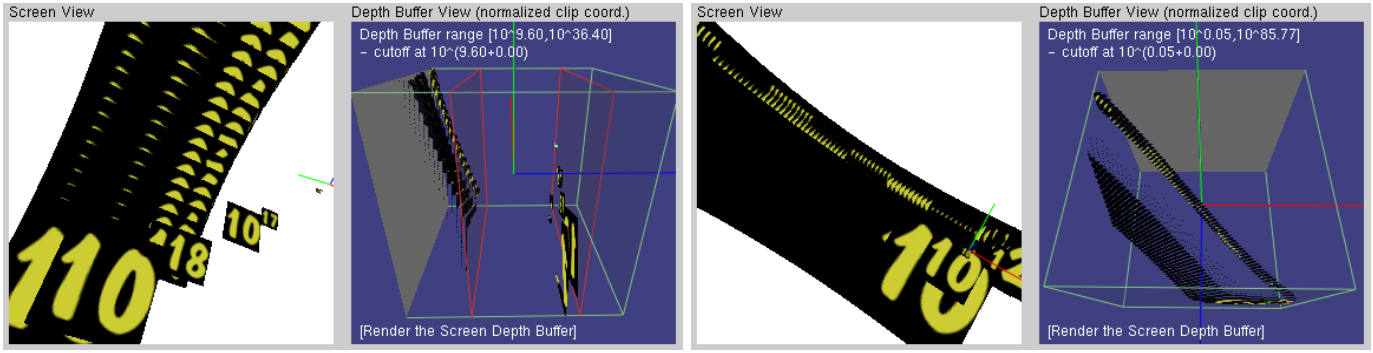
3) *Rescaling Eye-Space Coordinates* Finally, we have to rescale $\vec{\mathbf{v}}'_{\text{eye}}$ according to z_{NDC}. The method contains some subtleties: in order to reuse the settings in the user-supplied projection matrix for clip-space coordinate output, we back-project z_{NDC} to the eye space by inverse projection, and rescale $\vec{\mathbf{v}}'_{\text{eye}}$ along its line of sight so that its z value matches the back-projected z. So long as $\vec{\mathbf{v}}'_{\text{eye}}$ is located inside the given depth-rescaling model, the rescaled eye-space coordinate will always lie inside the user-defined viewing frustum. Therefore, we can precisely compute clip-space coordinates based on the user-supplied projection matrix.

It is important that all eye-space coordinates have to go through the above operations so that the entire eye space is scaled down in a uniform manner consistent with the given depth rescaling model. We can thus preserve depth ordering and ensure proper homogeneous clipping for polygons crossing the canonical viewing frustum. Furthermore, note also that the way we rescale coordinates in orthographic projection is slightly different because the lines of sight are all parallel in orthographic projection.



(a) Depth Buffering Error with Bare Rendering

(c) With Linear Cutoff Region (10^0 up to 10^{85} units)



(b) Depth Buffering Error Corrected

(d) Without Linear Cutoff Region (10^0 up to 10^{85} units)

Fig. 3. Demonstration: Rendering scenes with very large spatial range. Notice the proper polygon clipping in (b), (c), and (d). Note that the depth buffer view is a side view in (a) and (b), and a top view in (c) and (d).

3.3 Examples: Scenes with Large Spatial Range

Figure 3 demonstrates the application of the PSC transformation and the depth rescaling method to the problem of rendering scenes with very large spatial ranges. Each sub-figure contains two views; the screen view on the left shows the rendering result, while the depth buffer view on the right plots the corresponding depth buffer in 3D. In the scene, the textures on the polygons label their powers-of-ten distances from the world origin.

Figure 3(a) presents the bare rendering of the scene without any PSC support. We can see from the depth buffer view that most polygons cluster near the far plane and depth buffering anomalies appear in the screen view due to limited precision. With PSC support turned on, Figure 3(b) shows the corrected view and the depth buffer is much better utilized. Figures 3(c)

and (d) demonstrate the mapping in the depth rescaling model; though the two screen views are the same, the way we utilized the depth buffer is different. The scene shown in these two sub-figures ranges from 10^0 up to 10^{85} units in the world space.

4 PSC Acceleration Techniques

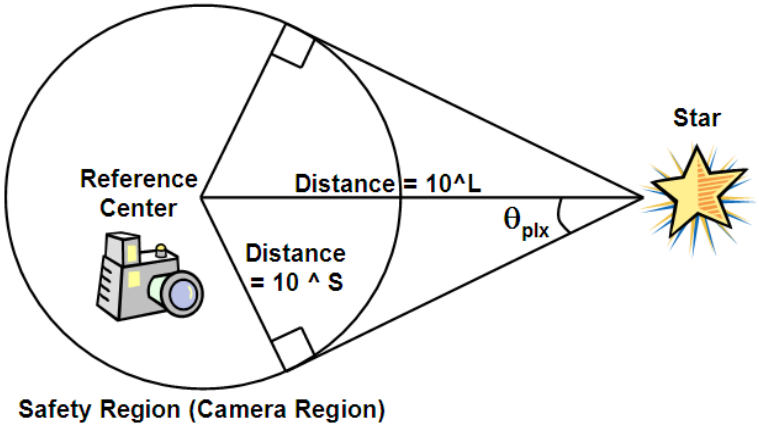
The physical Universe is a highly inhomogeneous large-scale environment filled mostly with empty space. These properties can be exploited and incorporated into a family of specific acceleration techniques described in this section.

4.1 The Environment Caching Criterion

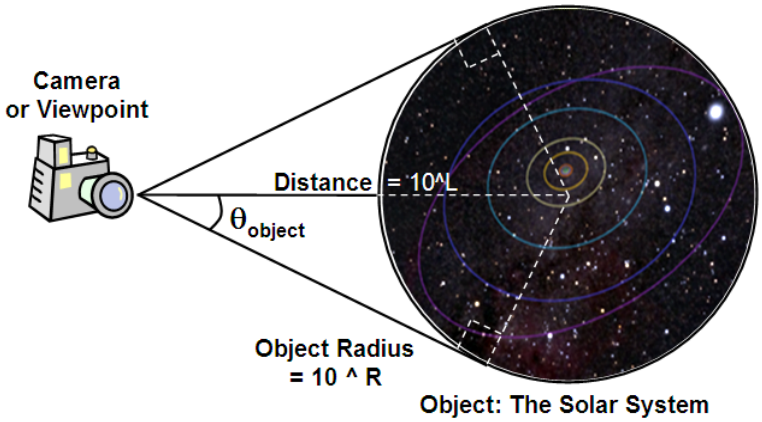
Our first technique applies when we travel within a limited region (e.g., the solar system) that contains detailed information, but is so far from other visible models that those models (e.g., stars and galaxies) are effectively stationary background for motions within the region. Similar to the imposter method [38], when these background models are too far away from our navigation region for camera motion to affect their renderings, we can cache their renderings in the form of image-based objects or on a panoramic background surface, and thus avoid rendering these objects every frame. The navigation region is called the *safety region* and the criterion for a pre-rendered background is called the *environment caching criterion*.

Figure 4(a) depicts the geometric model for environment caching. Given θ_{pixel}, the maximum angular size of a pixel on screen, and $2\theta_{\text{plx}}$, the maximum parallax angle, we can express the criterion in the form of the following rule:

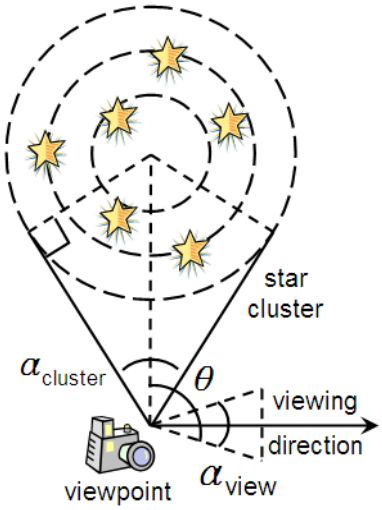
If $2\theta_{\text{plx}} < k\theta_{\text{pixel}}$ *for some non-negative k, objects that are* 10^L *units away will not move more than k pixel units on the screen if the safety region of the camera has radius* 10^S *units.*



a) The Environment Caching Criterion: $S + \delta(k) < L$



b) The Object Disappearance Criterion: $R + \delta(k) < L$



c) VP-Tree Pruning

Fig. 4. The acceleration criteria. Since $10^{L} >> 10^{S}$ and $10^{L} >> 10^{R}$, drawings are not to scale.

Using simple geometry, we have

$$2\sin^{-1}(10^{S-L}) < k\theta_{\text{pixel}}$$

$$S - L < \log_{10}(\sin(k\theta_{\text{pixel}}/2)), \text{ where we define } \delta(k) \text{ as } \log_{10}(\sin(k\theta_{\text{pixel}}/2)).$$

For a screen size of 1024×1024 and a field-of-view of $\pi/3$, $\delta(1) \approx 3.29$, so we can cache objects with distances from the safety region greater than $10^{16.29}m$ with 1-pixel fidelity when moving in the solar system. Due to the large empty spaces between different local navigation regions, such as solar systems, star clusters, and galaxies, we can exploit this criterion to accelerate rendering during local navigation for a number of different environments in the Universe.

4.2 The Object Disappearance Criterion

Our second method deals with the case where far-away objects are too small to be observable on the screen. If we compare Figures 4(a) and (b), we can see that the two criteria are actually duals of each other. If we interchange the location of objects and camera, θ_{plx} (and 10^S) and θ_{object} (and 10^R) are swapped, and we can obtain the equation for the *object disappearance criterion* shown in the figure. The precise form of the rule is the following:

> *If* $2\theta_{\text{object}} < k\theta_{\text{pixel}}$ *for some non-negative k, objects whose radius is* 10^R *units will not occupy more than a disk k pixel units in radius on the screen if they are* 10^L *units away from the camera.*

When applying this criterion, it is important to note that it is not necessarily correct to ignore the rendering of all small objects; bright objects of subpixel extent, rendered against a very dark background, may deposit a visible amount of energy into a pixel. For this situation, we switch to star rendering mode to handle small bright objects, rendering them based on their apparent brightness. These two criteria do not conflict with the PSC transformation method and the depth rescaling method; instead, the two transformation methods enable large-scale rendering while the two acceleration criteria optimize the performance of the large-scale rendering.

4.3 Hierarchy Rendering using Vantage Point Tree

The idea of using spherical region classification to accelerate large-scale rendering can be further extended by applying the Vantage-Point tree (VP-tree) [12, 49] for modeling and rendering point-based data such as stars and galaxies. The VP-tree is a data partitioning structure constructed so that each non-leaf node contains a partitioning sphere with a center in PSC and a radius encoded by a logarithmic scale. Individual stars are normally stored at leaf nodes while

partitioning spheres cluster stars in a hierarchical structure. By building VP-trees for stars and galaxies, we achieve the following advantages:

• First, we can transparently apply the object disappearance criterion to render star clusters instead of just stars. During the tree traversal, if a star cluster is too small to be viewable, we can render the whole cluster as if it is a single star. Note that, for efficiency, we will pre-store star rendering information at internal nodes.

• Secondly, the hierarchical structure facilitates data pruning. As shown in Figure 4(c), if a certain partitioning sphere is out of the camera view, we can quickly identify the associated star cluster and skip any attempt to render the stars in the cluster.

• Finally, efficient spatial search is straightforward, e.g., we can find the stars near to the current viewpoint. An efficient N-nearest neighbor search algorithm for the VP-tree is given in [12,49].

5 Virtual Astronomy: Scale-Independent Modeling and Rendering

The top level of our scalable visualization architecture is the virtual astronomy layer responsible for modeling and rendering different astronomical bodies. In this section, we present rendering techniques for stars and galaxies. Specific examples are based on the Hipparcos and Tycho Catalogues [18] from ESA and the Tully Galaxy Catalog [42].

5.1 Modeling and Rendering Stars

The following steps implement 3D modeling and rendering for stars:

1) *Star Positioning* The distance to stars is measured by the angular displacement given the two-AU observational baseline, where AU is the "Astronomical Unit," the mean Earth-Sun distance. The angular displacement is usually a very small angle, called the *parallax*, and is inversely proportional to distance. Astronomers traditionally employ a parallax-motivated dis-

tance unit called the *parsec*, defined to that

$$\text{distance (parsecs)} \;=\; 1/\text{parallax (arc-seconds)}\,.$$

Note that error in parallax measurement is important; the error term should be taken into account when calculating distances in the above equation (see Appendix A.1). The equatorial coordinate system [20, 44] locates directions to stars and thus positions them in space. Figure 5 illustrates the positional uncertainty of star locations in two different scales, showing increased parallax-based errors at larger scales.

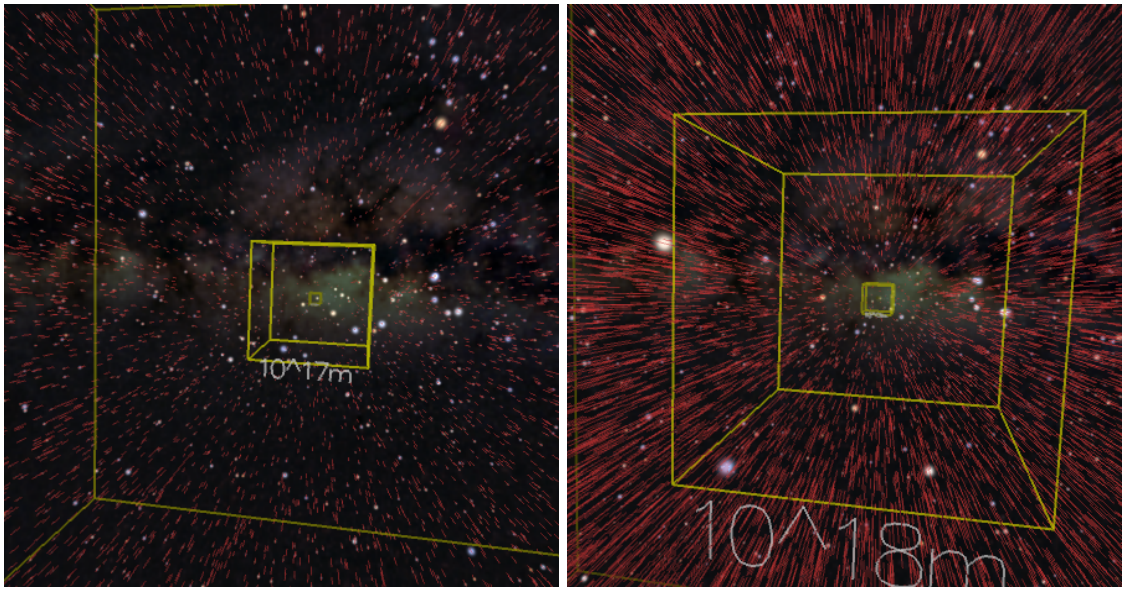


Fig. 5. Visualizing positional uncertainties: Red bars denote the error, which increases for stars further away from Earth due to the nature of parallax measurements.

2) *Visual/Apparent Magnitude* If we consider actual diameters of stars, only a few stars (e.g., Betelgeuse (Alpha Orionis)) have sizes observable from Earth even with the world's most powerful telescopes. However, since starlight comes to us from a very dark background, we can classify stars based on their visual brightness as well as their spectra. Astronomers estimate distance based on a star's brightness using the magnitude equation,

$$M \;-\; m \;=\; 5 \;-\; 5\log_{10} d\,,$$

where M is the absolute magnitude, m is the apparent magnitude, and d (in parsecs) is the distance to the star from the viewer. With m and d measured from Earth, we can pre-compute M for each star so that, during the navigation, we can compute $\log_{10} d$ on-the-fly and determine m for each star accordingly. Furthermore, to compute m for star clusters in the VP-tree, we can sum up the star luminosity in the cluster (see Appendix A.2). Note that M is not additive.

3) *Star Profiles from Photographs* To work out star sizes from m, we developed the following photograph-based method:

1. Register photographs to star locations by minimizing L_2 error at point correspondences.
2. Extract star sub-images from photographs by matching the star profile.
3. Construct a lookup table for visualizing sizes of stars as a function of apparent magnitude m.

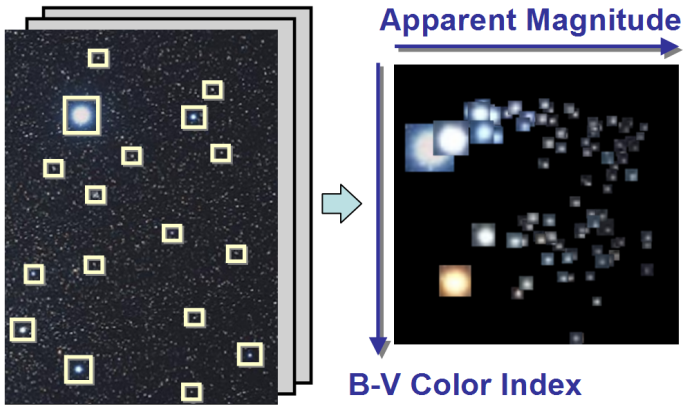


Fig. 6. Left: Extracting star sub-images from photographs. Right: Using measured magnitude (brightness) and spectrum (B-V color index) from multiple image sources to construct a lookup table.

Step 1 may require human intervention to locate some initial point correspondences, while the other two steps are fully automatic. Figure 6 illustrates the extraction process.

4) *Star Rendering* Finally, stars are rendered in 3D as textured billboards using the PSC transformation and the depth rescaling method. To apply colors to stars, we exploit the B-V color index [34] obtained from the star data set. Note that stars have different colors because of dif-

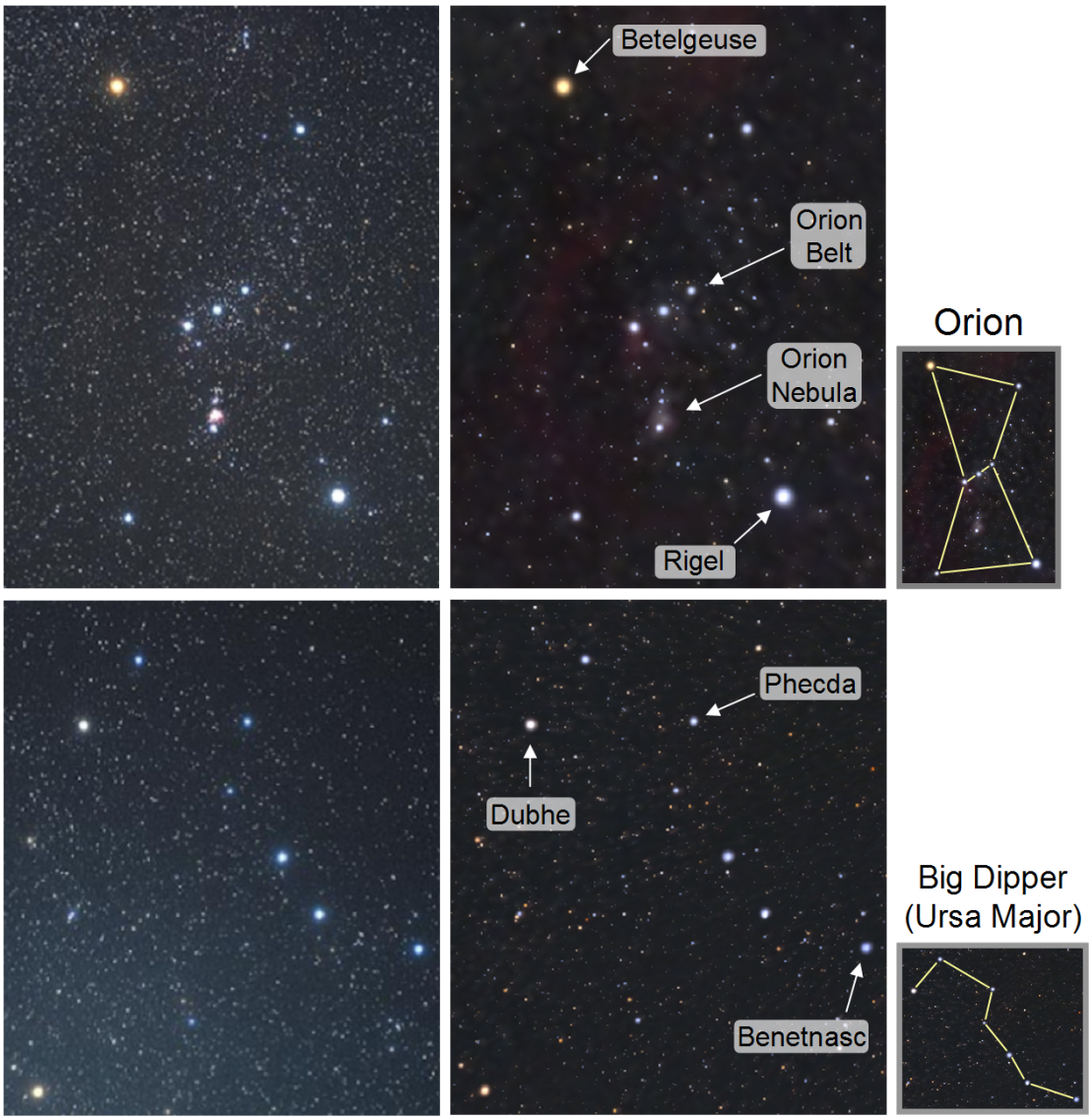


Fig. 7. Comparing photographs (left) and our renderings (right): Orion and the Big Dipper. Note that we do not simulate zodiacal light and counter-glow, and the Hipparcos and Tycho Catalogues do not include all the stars visible in the actual photographs (AAO/ROE, photographs courtesy David Malin).

ferent physical composition and surface temperature. Figure 7 compares our renderings with actual photographs — notice the sizes and colors of the stars being rendered.

5.2 Modeling and Rendering Galaxies

Unlike stars, which are spheres and typically point-like, galaxies have a variety of observed extended shapes, i.e., spiral, lenticular, elliptical and irregular. Based on the information given

in the Galaxy Catalog [42], we employ a level-of-detail approach for galaxy rendering. The following representations are used in decreasing order of log distance from the viewer:

1) *Galaxies as stars* Parallax is of no use in determining distances on the galactic scale. To determine distances to galaxies, we instead apply the Hubble relation between distance and the velocity of recession, as determined from the spectral redshift (see Appendix A.3). By convention, we use the supergalactic coordinate system (see Appendix A.4) to locate directions to galaxies and, if a galaxy is point-like (subpixel size) in the observation region, we render them as starlike objects in space.

2) *Galaxies as Textured Polygons* The available galaxy catalog documents the morphological types of galaxies (see Appendix A.5). Sample photographs of galaxies belonging to each type have been selected and used as generic texture maps to simulate the appearance of morphological types for which we have no suitable direct image (see Appendix A.6).

3) *Galaxies in 3D* Currently, a statistical particle model based on the Goddard Milky Way Model [16] is used to represent the structure of the Milky Way Galaxy, for which only nearby stars can be assigned accurate 3D positions. Similar models are in principle applicable to other galaxies for which detailed geometric information is available.

6 Implementation and Results

6.1 System Implementation

Our generic scalable visualization architecture is built as a module-based system, which includes the PSC module, modules for modeling and rendering, the navigation module, and the interface module. The major advantage of the design is that we can deliver a flexible develop-

TABLE II
AVERAGE FRAME RATES FOR THE PSC LIBRARY API RENDERED WITH/WITHOUT SHADERS (IN FRAMES PER SECOND)

GeForce4 440 Go (Mobile) Notebook (1.66GHz CPU)		Disable Shader (CPU)			Enable Shader (GPU)		
		Don't Care	Fastest	Nicest	Don't Care	Fastest	Nicest
Perspective	cutoff	653.42	647.89	552.13	243.67	273.98	264.37
	nocutoff	744.72	744.53	641.25	267.88	299.05	282.18
Orthographic	cutoff	552.56	555.81	494.06	224.49	246.11	237.57
	nocutoff	614.15	621.56	537.21	235.92	260.24	250.81

GeForce 6800 Ultra Desktop (3.2GHz CPU)		Disable Shader (CPU)			Enable Shader (GPU)		
		Don't Care	Fastest	Nicest	Don't Care	Fastest	Nicest
Perspective	cutoff	1052.07	1057.40	930.83	1140.10	1149.95	1106.21
	nocutoff	1123.20	1126.50	1086.52	1140.86	1152.15	1108.52
Orthographic	cutoff	907.59	916.27	823.67	1107.70	1117.50	1072.02
	nocutoff	988.65	995.40	885.45	1122.67	1129.72	1088.57

ment platform for implementing different applications and animations; we can pick up individual modules for specific applications and enhance the system functionality by adding in more modules. Several visualization applications have been developed and a twenty-minute long pedagogical animation entitled "Solar Journey" has been produced based on this modular system.

The PSC Module This module offers a library API that realizes the PSC transformation method and the depth rescaling method. Two versions were developed: a direct software implementation and a GPU-based implementation using OpenGL shaders.

Table II summarizes the rendering performance on different platforms. An animation sequence showing textured polygons from 10^{-30} to 10^{30} (see Figure 3) is rendered, and the average frame rate is recorded for each case. Like the OpenGL command `glHint`, we have three levels of PSC computations, `don't care`, `fastest`, and `nicest`, to trade off accuracy and efficiency. From the table, we can see that using perspective projection is faster than using orthographic projection; using shaders on a relatively slow mobile GPU may not improve the rendering performance, while ignoring the linear cutoff region can understandably improve the performance.

Modules for Modeling and Rendering In terms of efficiency and scalability, we maintain

individual modules for each type of astronomical object, e.g., galaxies [42], the Milky Way Galaxy [16], stars [18, 36], the solar system [39], satellites [24], etc. Each module contains a data loader and a PSC-based renderer for proper rendering of objects across huge viewing scales.

The Navigation Module This module provides effective exploration in spaces with large scale-range requirements:

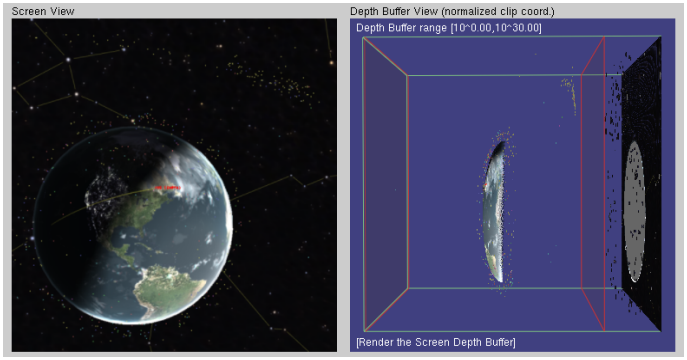
- Travel: Use PSC-supported camera paths or constrained navigation manifolds [47].
- Wayfinding: Exploit navigation cues such as powercubes for depicting spatial scale and customized depiction of neighboring stars/objects using scalable maps.
- Animation scripting.

The Interface Module This module connects rendering tasks, navigation, and the application. It not only optimizes the rendering and navigation performance, but also provides a development basis for implementing different applications and animations on top of the system architecture.

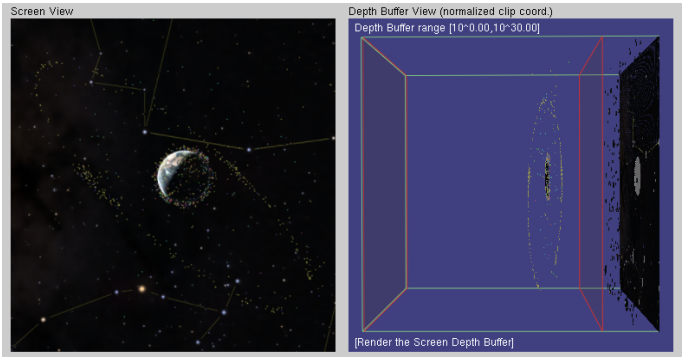
6.2 Visualization Results – A Powers-of-ten Journey

Figures 8(a)-(l) depict a powers-of-ten journey across the physical Universe from the Earth to extra-galactic scales. To reveal the depth buffer utilization, we apply the same presentation style as in Figure 3 and plot the depth buffer of the screen view on the right. Note that for efficiency in large-scale rendering, we turn off the near safety region in the depth rescaling process.

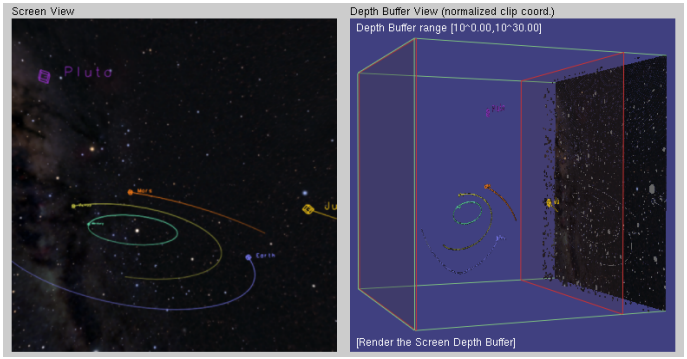
The journey begins with a view of the Earth and the International Space Station (labeled in red with a yellow comet trail) in the morning over North America. Satellite positions are tracked using a real-time-based computer clock. Moving outward from the Earth, we find a ring of geostationary operational environmental satellites (GOES) about $3.58 \times 10^7 m$ above the Earth. We can clearly see from the depth buffer view that satellites and Earth are rendered into the linear



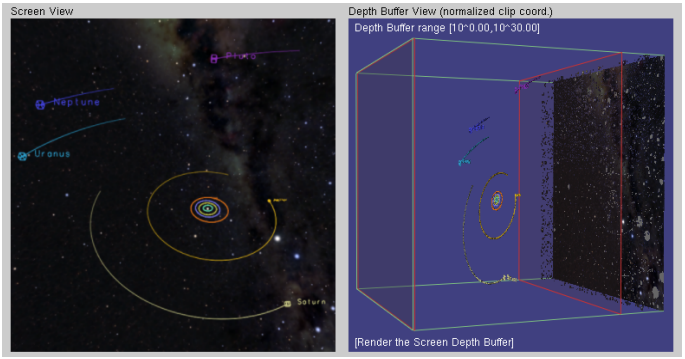
(a) The International Space Station (ISS) over the Earth ($10^7 m$)



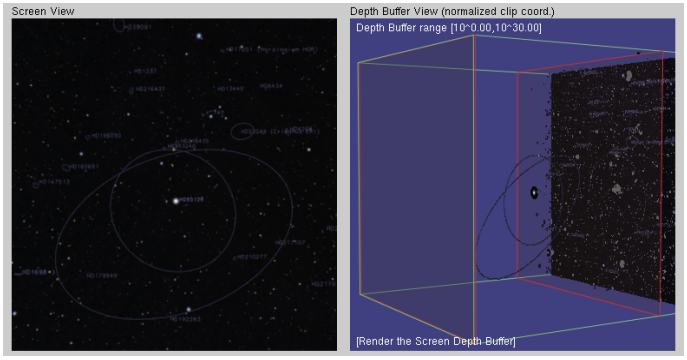
(b) Satellites [24] ($\sim$ 1000) surrounding the Earth ($10^8 m$)



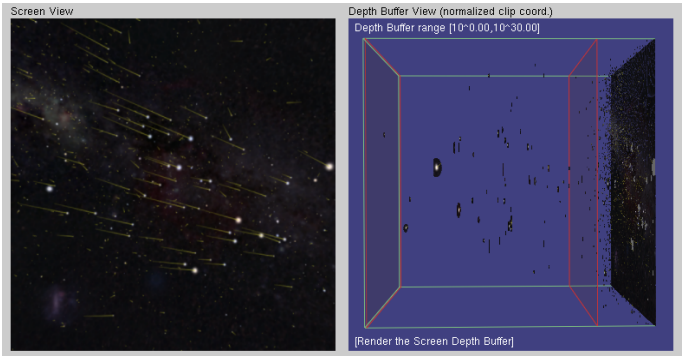
(c) The solar system [39] (inner planets) ($10^{12} m$)



(d) The solar system (outer planets) ($10^{13} m$)



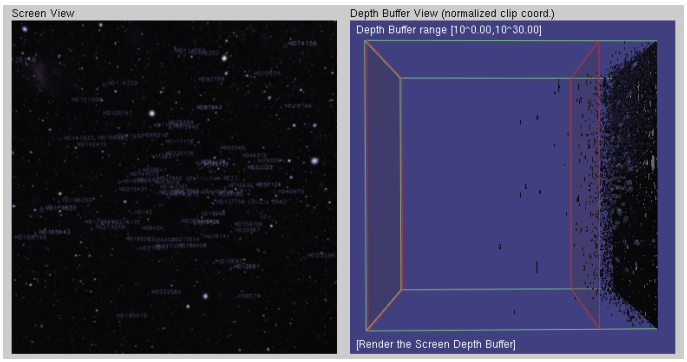
(e) A multi-planet solar system (HD95128) ($10^{17} m$)



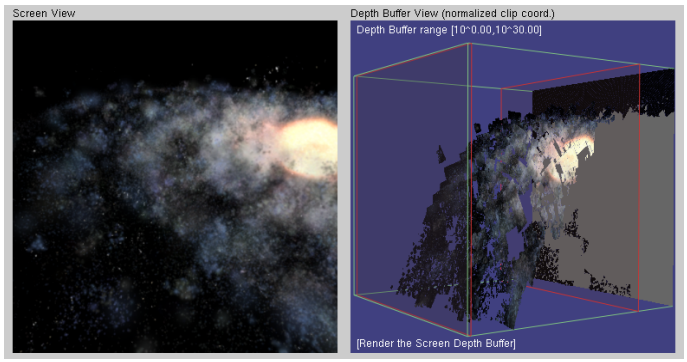
(f) Pleiades (visualizing space velocity of stars) ($10^{18} m$)

Fig. 8. A Powers-of-Ten Journey – a scale-independent visualization of the three-dimensional physical Universe.

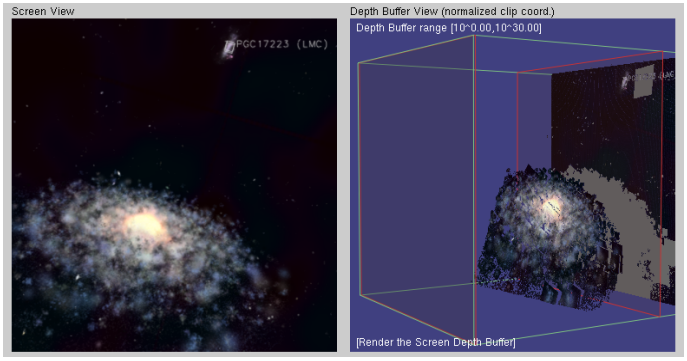
cutoff region while stars are rendered into the far safety region. Moving still further from the Earth, we pass through the inner and outer planets of the solar system — stars remain in the far safety region because they are still too far away to matter. After reaching the interstellar level, as shown in sub-figure (e), the stars enter the linear cutoff region and we arrive at the multi-planet solar system of star HD95128, with two extrasolar planets. Farther along our path, we visit the



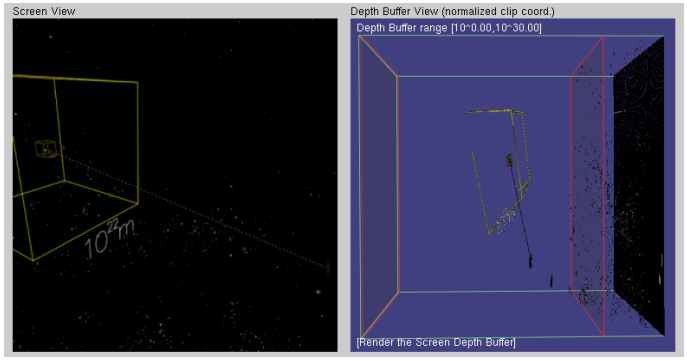
(g) Extrasolar planets $[18, 36]$ $(10^{18}m)$



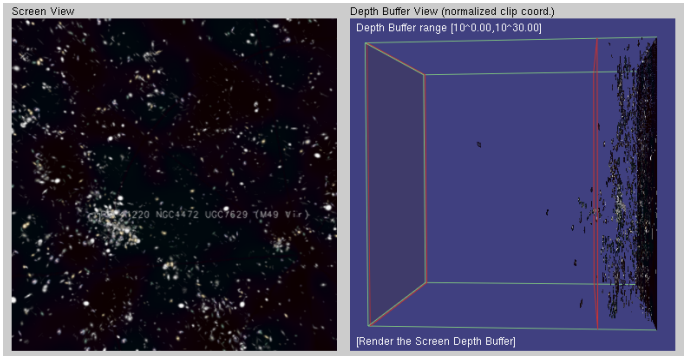
(h) Spiral arm of the Milky Way Galaxy [16] $(10^{21}m)$



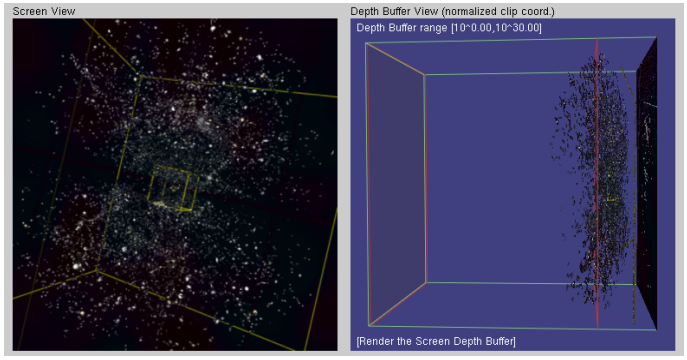
(i) The Milky Way and Large Magellanic Cloud $(10^{22}m)$



(j) The Local group: Milky Way and Andromeda $(10^{23}m)$



(k) The Virgo cluster $(10^{24}m)$



(l) The Galaxy catalog [42] $(10^{25}m)$

Fig. 8 (cont.).

Pleiades in the constellation Taurus and show an interesting visualization of the stellar space velocities (shown as comet trails). Stars in the Pleiades cluster are seen to be moving together with very similar velocities. Note that in making this visualization, we combine the star proper motion in the Hipparcos and Tycho Catalogues [18] with an additional radial velocity catalog so as to obtain full 3D space velocity.

We then travel past the interstellar scale and reach the Milky Way Galaxy scale. The Milky

Way Galaxy is no longer a background image in the sky, but is re-expressed as a three-dimensional body renderable in the linear cutoff region. Going further outward brings us to the Local Group, which includes the Milky Way Galaxy and the Andromeda Galaxy. The powercube visualization method provided by the navigation module establishes the spatial scale around the Milky Way Galaxy (see sub-figure (j)). Finally, we pass through the Local Group and the Virgo Cluster, and begin to access the largest scale data set, the Galaxy Catalog [42]. Every point shown in the rendering is a galaxy containing millions or billions of stars. The PSC transformation method facilitates exploration and rendering, while the depth rescaling method enables an efficient and proper depth buffer utilization; thus we never have to worry about depth anomalies so long as we identify appropriate far-away objects and place them in the far safety region. The scalable architecture thus makes a continuous exploration of the Universe feasible without regard to scale.

6.3 Development Results – Application and Animation

In addition to journeying through the cosmos, we can employ our system architecture to produce an assortment of visualization applications for visualizing a detailed interstellar environment, for aligning photographs or 3D models in the night sky, as well as for data-driven gas cloud construction. A number of these tools have been put into practice by astronomers and are in the process of being distributed to the astronomy community. The system architecture is also valuable for creating animations for pedagogical presentations of astronomy concepts, e.g., the system was used to produce the "Solar Journey" planetarium-style animation, which explores the properties of interstellar objects surrounding our solar system and exposes the features of the interstellar space through which our Sun travels.

7 Conclusion and Future Work

This work presents a comprehensive visualization architecture for exploring the physical Universe regardless of its component scales. By exploiting the power scaled coordinate representation, we can efficiently manipulate spatial data at huge or sub-atomic scales in a uniform fashion. Motivated by the exponential zoom idea in the "Powers of Ten" film, the PSC transformation method precisely incorporates the exponential scaling factor into the context of the modelview matrix for large-scale exploration. The depth rescaling method then compensates for extreme projection ranges so that we can fully control potential anomalies in the depth buffering. The two acceleration criteria and the VP-tree structure further optimize the rendering performance for gigantic environments with huge scale differences. Based on these techniques, the virtual astronomy layer can support scale-independent modeling and rendering so that we can safely implement star rendering, galaxy rendering, and various rendering strategies for different astronomical bodies. The module-based system provides us with the flexibility to extend the system to additional data domains. Some modules have been exported to SGI for their digital planetarium projects. Several scientific visualization applications have been developed, and an educational animation has been produced utilizing the system architecture.

We envision future efforts that replace the virtual astronomy layer with some layers supporting visualization applications based on physical data from the Planck-length scale up to nuclear, molecular, and human scales as well as astronomical scales. On the other hand, it is interesting to observe that the powers-of-ten phenomenon is not limited to space; we also have extreme scale ranges in time, mass, density, energy, as well as other physical quantities. Visualization methods exploring these quantities and implementing additional scalable architectures would be of further interest for science and education.

ACKNOWLEDGMENTS

We would like to thank astrophysicist Priscilla Frisch for her constant help in data collection and feedback on the visualizations, Stuart Levy for his useful input on star rendering and astronomical formulae, David Malin for his constellation and galaxy photographs, and Brent Tully for his galaxy catalog and decoding module. This research was supported in part by NASA grant numbers NAG5-8163,NAG5-11999, and NAG5-13558, Silicon Graphics Incorporated, and the Hong Kong RGC Earmarked Grant HKUST-612505.

APPENDICES

A. 1 Distance to Stars Given plx and $\triangle plx$ as the parallax and its error term, we need higher-order approximations to compute the distances:

$$\text{Distance to stars} \approx \frac{1}{2}\,[\,1/(plx+\triangle plx) + 1/(plx-\triangle plx)\,] = 1/plx \cdot \left[1-\left(\frac{\triangle plx}{plx}\right)^2\right]^{-1} \approx 1/plx + (\triangle plx)^2/plx^3\,.$$

Note that this kind of precautionary operation can be easily overlooked, but for accurate visualization, we have to be very careful when computing measured quantities with significant errors.

$$\begin{aligned} 1\ \textit{astronomical unit}\ (\text{A.U.}) &= 1.495979 \times 10^{11}\ \text{m} \\ 1\ \textit{light-year}\ (\text{ly}) &= 9.460528 \times 10^{15}\ \text{m} \\ 1\ \textit{parsec} &= 3.085678 \times 10^{16}\ \text{m} \end{aligned}$$

A. 2 Star Magnitudes and Luminosity

- Luminosity – total energy a star gives off per second
- Apparent magnitude m – brightness of a star as it appears to us
- Absolute magnitude M – equals m if a star is 10 parsecs away

The astronomer's rule of thumb is that a difference of 5 magnitudes corresponds to a factor of 100 in brightness. The smaller the magnitude the brighter a star is. Thus, we can define

$$\text{relative luminosity, } L = (\sqrt[5]{100})^{-M}\,,$$

so that we can sum L for star clusters. Note that for efficiency, astronomers often approximate $\sqrt[5]{100}$ by 2.5 in the calculation.

Furthermore, we can adjust rendering exposure by linearly scaling L.

A. 3 The Hubble Relation

$$V \text{ (velocity of recession)} \quad = \quad H \times d \text{ (distance to galaxy)}$$

where H is the constant of proportionality, known as the Hubble constant. A recent result from the Hubble Space Telescope Key Project [1] in 2001 gives the Hubble constant as 72 ± 8 km/s/Mpc (with an uncertainty of $\sim$10%).

A. 4 Coordinate Conversion There are three main coordinate systems involved in the modeling of the Universe (see Figure 9):

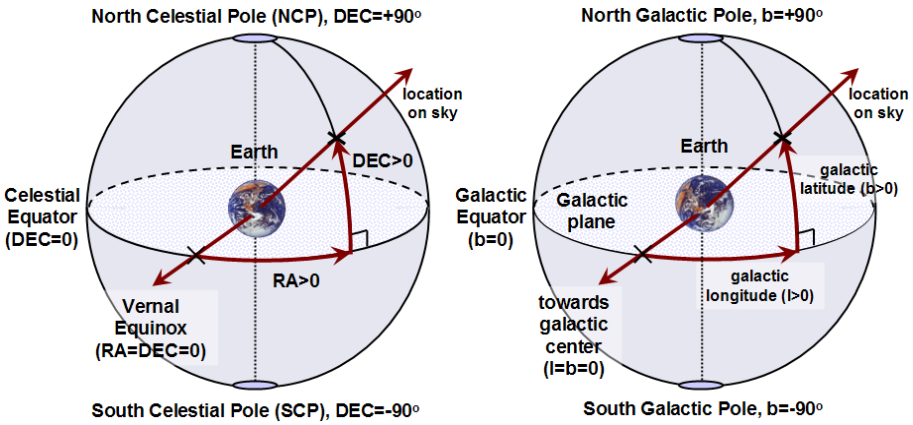


Fig. 9. The Equatorial Coordinate system (left) and the Galactic Coordinate system (right).

1) The Equatorial (or Celestial) Coordinate system: This is based on the Earth's rotation around its poles and the Vernal Equinox direction. We can define Right Ascension (RA) and Declination (DEC) to specify directions in the sky. We refer readers to [20] for conversions between the Equatorial and the Galactic Coordinate systems.

2) The Galactic Coordinate system: This is based on the Milky Way plane and the direction to the Milky Way center. We can define Galactic Longitude (l) and Galactic Latitude (b) to specify directions in the sky. Note that we use general rectangular conversion methods to map RA/DEC and l/b to rectangular coordinates at a distance d from Earth:

$$\begin{cases} x &= d\ \cos(\text{longitude})\cos(\text{latitude}) \\ y &= d\ \sin(\text{longitude})\cos(\text{latitude}) \\ z &= d\ \sin(\text{latitude})\ . \end{cases}$$

3) The Supergalactic Coordinate system: The transformation matrix (a rotation) from supergalactic coordinates to galactic coordinates is:

$$\begin{pmatrix} -0.735742574804 & -0.074553778365 & 0.673145302109 \\ 0.677261296414 & -0.080991471307 & 0.731271165817 \\ 0.000000000000 & 0.993922590400 & 0.110081262225 \end{pmatrix}\ .$$

Further discussion of this coordinate system can be found in [15].

A. 5 Hubble's classification Edwin Hubble first proposed the following "tuning fork" diagram (see Figure 10) to classify galaxies in 1936. A coarser scheme is to flatten this tuning fork and create a morphological number ranging from -5 (Elliptical) to +10 (Irregular).

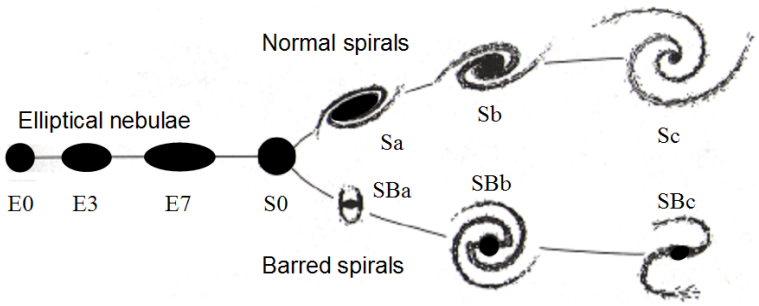


Fig. 10. Elliptical → Lenticular → Spiral → Irregular.

A. 6 Positioning Galaxies To position a galaxy's axial plane, we employ the following quantities provided by the galaxy data:

- (sgx, sgy, sgz) – Cartesian position in supergalactic coordinates
- *hpa* – hold position angle
- *bovera* (b/a) – ratio between minor and major axes
- *adiam* – metric diameter (galaxy diameter along the major axis)

Astronomers define the *major* and *minor* axes of galaxies to represent the principal and co-principal axes, respectively, in the viewing plane (see Figure 11). Then, with coordinates (sgx, sgy, sgz), we can set up the initial position of these axes so that the major axis is parallel to the SGX/SGY-plane, while the minor axis is perpendicular to both the major axis and the direction to the galaxy. Next, we define *hpa* to rotate the two axes counter-clockwise about the direction to the galaxy. Thus, we can align the galaxy axes (and galaxy photographs) from the Earth's perspective.

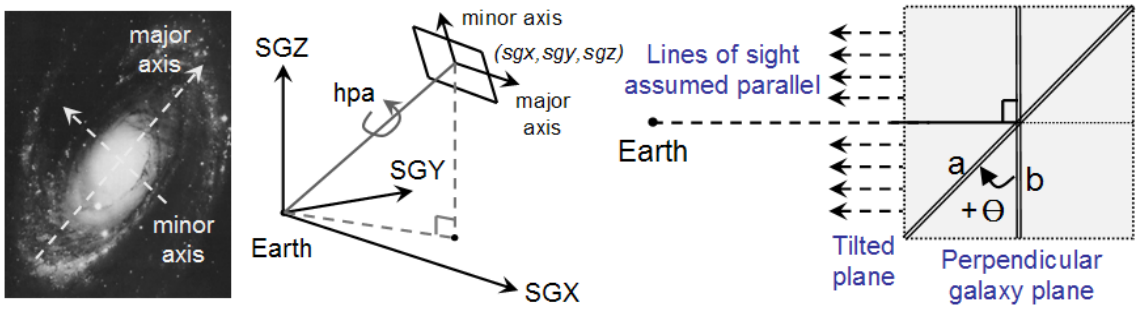


Fig. 11. Positioning axial planes of galaxies.

To estimate realistic axial planes, astronomers attempt to predict the plane orientation using the following assumption (denote a and b as lengths of the major and minor axes, respectively): based on symmetry, galaxies are assumed to be circular in shape, so b is lengthened along the line of sight and the lengths of both axes becomes equal. However, since galaxies are so far away from us, it is not possible to reliably determine the tilting direction experimentally. In other words, the galaxy plane may be tilted by $+\theta$ or $-\theta$, where $\theta = cos^{-1}(b/a)$; In practice, we have to pick the sign at random.

REFERENCES

[1] The Hubble Space Telescope Key Project on the Extragalactic Distance Scale, 1994–. http://www.ipac.caltech.edu/H0kp.

[2] The Sloan Digital Sky Survey, 1997–. Managed by the Astrophysical Research Consortium, website: http://www.sdss.org.

[3] R. Alonso. Open Universe, 2001. http://openuniverse.sourceforge.net.

[4] G. V. G. Baranoski, J. G. Rokne, P. Shirley, T. Trondsen, and R. Bastos. Simulating the aurora borealis. In *Proceedings of Pacific Graphics 2000*, pages 2–14, October 2000.

[5] J. Blinn. Voyager Fly-by Animations, 1977–1987. Animations depicting space missions to Jupiter, Saturn and Uranus: http://research.microsoft.com/users/blinn/FLYBY.HTM.

[6] J. Blinn. COSMOS, 1979–1980. Computer graphics animations for the Carl Sagan PBS series.

[7] J. Blinn. The Mechanical Universe, 1983–1986. Animated sequences for the 52 part telecourse, details at http://research.microsoft.com/users/blinn/mu.htm.

[8] J. Blinn. Where am I? What am I looking at? *IEEE Computer Graphics and Applications*, 8(4):76–81, July/August 1988.

[9] K. Boeke. *Cosmic View: The Universe in Forty Jumps*. John Day, 1957.

[10] C. and R. Eames. Powers of Ten, 1977. 9 1/2 minute film, made for IBM.

[11] F. Chereau. Stellarium, 2001–. http://stellarium.free.fr.

[12] T. C. Chiueh. Content-based image indexing. In *Proceedings of the 20th VLDB Conference*, pages 582–593, 1994.

[13] D. Cox, R. Patterson, and M. Thiebaux. Virtual Director, 1992–1995. website at: http://virdir.ncsa.uiuc.edu/virdir/virdir.html.

[14] T. M. Dame, Dap Hartmann, and P. Thaddeus. The Milky Way in molecular clouds: A new complete CO survey. *Astrophysical Journal*, 547(2):792–813, 2001.

[15] G. de Vaucouleurs, A. de Vaucouleurs, H. G. Corwin, R. J. Buta, G. Paturel, and P. Fouque. *Third Reference Catalogue of Bright Galaxies*. Volume 1-3, XII, 2069 pp. 7 figs.. Springer-Verlag Berlin Heidelberg New York, 1991.

[16] E. Dwek, R. G. Arendt, M. G. Hauser, T. Kelsall, C. M. Lisse, S. H. Moseley, R. F. Silverberg, T. J. Sodroski, and

J. L. Weiland. Morphology, near-infrared luminosity, and mass of the Galactic bulge from COBE DIRBE observations. *Astrophysical Journal*, 445:716–730, June 1995.

[17] C. Emmart, D. R. Nadeau, J. Genetti, and E. Wesselak. Volume Visualization of the Orion Nebula, 2000. SIGGRAPH 2000 Electronic Theatre, Issue 134, description at `http://vis.sdsc.edu/research/orion.html`.

[18] ESA. The Hipparcos and Tycho Catalogues, ESA SP-1200, 1997. Available at `http://www.rssd.esa.int/Hipparcos/catalog.html`.

[19] J. Genetti. Volume-rendered galactic animations. *Communications of the ACM*, 45(11):62–66, 2002.

[20] Ed. Green, R. *Spherical Astronomy*. Cambridge University Press, 1985.

[21] A. J. Hanson and C.W. Fu. Cosmic Clock, 2000. Siggraph Video Review, vol. 134, scene 5.

[22] A. J. Hanson, C.W. Fu, and E. A. Wernert. Very large scale visualization methods for astrophysical data. In *Proceedings of the Joint Eurographics and IEEE TVCG Symposium on Visualization*, pages 115–124. Springer Verlag, May 29-31, Amsterdam 2000.

[23] D. Hoffleit and W. H. Warren Jr. *The Bright Star Catalogue: fifth revised edition*. Yale University Observatory, 1991.

[24] F. Hoots and R. L. Roehrich. Models for the Propagation of NORAD Element sets, Dec. 1980. in Space Track Report Number 3: `http://celestrak.com/NORAD/documentation`.

[25] M. Hopf, M. Luttenberger, and T.Ertl. Hierarchical splatting of scattered 4D data. *IEEE Computer Graphics and Applications*, 24(4):64–72, July/August 2004.

[26] H. W. Jensen, F. Durand, J. Dorsey, M. M. Stark, P. Shirley, and S. Premoze. A physically-based night sky model. In *Proceedings of ACM SIGGRAPH 2001*, Annual Conference Series, pages 399–408. ACM, August 2001.

[27] R. Kahler, D. Cox, R. Patterson, S. Levy, H. C. Hege, and T. Abel. Rendering the first star in the Universe - a case study. In *IEEE Visualization 2002*, pages 537–540. IEEE, 2002.

[28] C. Laurel, C. Weisbrod, F. Schrempp, and C. Teyssier. Celestia, 2001–. `http://www.shatters.net/celestia`.

[29] S. Levy. Partiview (Particle Viewer), 2001–. Available at `http://virdir.ncsa.uiuc.edu/partiview/`.

[30] M. Magnor, K. Hildebrand, A. Lintu, and A. J. Hanson. Reflection nebula visualization. In *IEEE Visualization 2005*, pages 255–262. IEEE, October 2005.

[31] P. Morrison and P. Morrison. *Powers of Ten*. Scientific American Books, 1982.

[32] D. Nadeau, J. Genetti, S. Napear, B. Pailthorpe, C. Emmart, E. Wesselak, and D. Davidson. Visualizing Stars and Emission Nebulae. *Computer Graphics Forum*, 20(1):27–33, March 2001. Also in Eurographics 2000 (short presentations).

[33] NCSA/UIUC. Cosmic Voyage Galaxy Formation & Interaction, 1996. SIGGRAPH '96 Electronic Theatre, Issue 115.

[34] T. Olson. The colors of the stars. In *IST/SID 6th Color Imaging Conference*, 1998.

[35] J. P. Ostriker and M. L. Norman. Cosmology of the early universe viewed through the new infrastructure. *Communications of the ACM*, 40(11):84–94, November 1997.

[36] California & Carnegie Planet Search. Extrasolar planets, 2003. http://exoplanets.org/almanacframe.html.

[37] B. Silleck. Cosmic Voyage, 1996. 35 minute film, a presentation of the Smithsonian Institution's National Air and Space Museum and the Motorola Foundation.

[38] Francois Sillion, George Drettakis, and Benoit Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. *Computer Graphics Forum (Proc. Eurographics 97),*, 16(3):207–218, 1997.

[39] M. Standish. New JPL Long Ephemeris, DE406/LE406., 2002. Available: http://ssd.jpl.nasa.gov/eph_info.html.

[40] M. R. Stytz, J. Vanderburgh, and S. B. Banks. The Solar system modeler. *IEEE Computer Graphics and Applications*, 17(5):47–57, September/October 1997.

[41] A. S. Szalay, J. Gray, A. R. Thakar, P. Z. Kunszt, T. Malik, J. Raddick, C. Stoughton, and J. vandenBerg. The SDSS skyserver: public access to the Sloan Digital Sky Server data. In *SIGMOD 2002 Conference Proceedings*, Annual Conference Series, pages 570–581. ACM SIGMOD, ACM Press New York, NY, USA, 2002.

[42] R. Brent Tully. *Nearby Galaxies Catalog*. Cambridge University Press, 1988.

[43] A. Turnage. Modeling supernovae: Braving a bold new frontier. *IEEE Computer Graphics and Applications*, 23(6):6–11, November/December 2003.

[44] U.S. NAVAL OBSERVATORY, R. G. O. *The Astronomical Almanac for the Year 2001*. 2001.

[45] G. Ward. Real pixels. *Graphics Gems IV*, 2:80–83, 1991.

[46] J. Welling. The Star Splatter, 1986–. Astrophysical Particle Renderer, release 1.0, http://www.psc.edu/Packages/StarSplatter_Home.

[47] Eric A. Wernert and Andrew J. Hanson. A framework for assisted exploration with collaboration. In *Proceedings of Visualization '99*, pages 241–248. IEEE Computer Society Press, 1999.

[48] K. Whitehouse. Comet explodes on Jupiter-and the Web. *IEEE Computer Graphics and Applications*, 14(6):12–13, November/December 1994.

[49] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.

[50] W. Zheng and O' Dell. A three-dimensional model of the Orion nebula. *Astrophysical Journal*, 438(2):784–793, 1995.