DOI: 10.1111/cgf.13202

Eurographics Conference on Visualization (EuroVis) 2017
J. Heer, T. Ropinski and J. van Wijk
(Guest Editors)

Volume 36 (2017), Number 3

Dynamic Scene Graph: Enabling Scaling, Positioning, and Navigation in the Universe

Emil Axelsson[1], Jonathas Costa[2], Cláudio Silva[2], Carter Emmart[3], Alexander Bock[1], and Anders Ynnerman[1]

[1] Linköping University, [2] New York University, [3] American Museum of Natural History

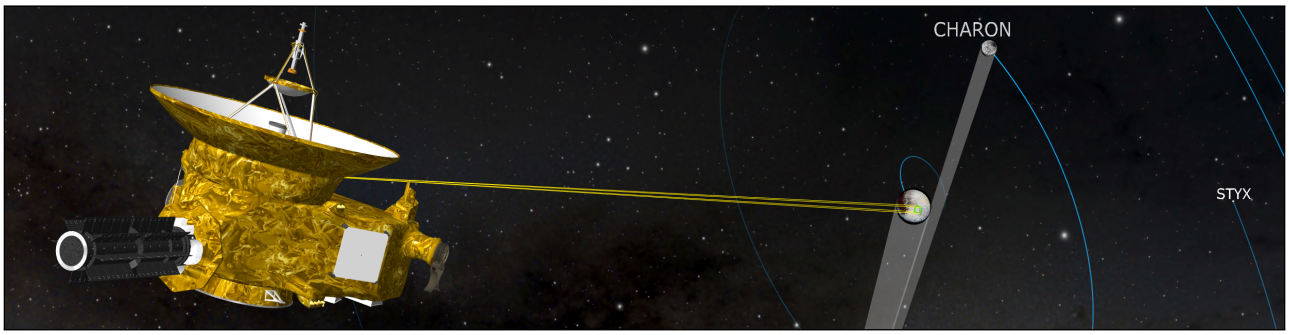


Figure 1: *Accurate rendering of a real scale model of the New Horizons spacecraft taking measurements on Pluto. The model of about 2 m size is shown in its correct location relative to Pluto, which is about $6 \cdot 10^{12}$ m from the coordinate system origin with the stars of the constellation Ophiucus (about 10^{18} m) in their correct 3D positions. High precision is required for computing the correct location of images on the surface of Pluto as well as correctly rendering the shadow cylinders of both Pluto and its moon, Charon.*

Abstract

In this work, we address the challenge of seamlessly visualizing astronomical data exhibiting huge scale differences in distance, size, and resolution. One of the difficulties is accurate, fast, and dynamic positioning and navigation to enable scaling over orders of magnitude, far beyond the precision of floating point arithmetic. To this end we propose a method that utilizes a dynamically assigned frame of reference to provide the highest possible numerical precision for all salient objects in a scene graph. This makes it possible to smoothly navigate and interactively render, for example, surface structures on Mars and the Milky Way simultaneously. Our work is based on an analysis of tracking and quantification of the propagation of precision errors through the computer graphics pipeline using interval arithmetic. Furthermore, we identify sources of precision degradation, leading to incorrect object positions in screen-space and z-fighting. Our proposed method operates without near and far planes while maintaining high depth precision through the use of floating point depth buffers. By providing interoperability with order-independent transparency algorithms, direct volume rendering, and stereoscopy, our approach is well suited for scientific visualization. We provide the mathematical background, a thorough description of the method, and a reference implementation.

1. Introduction

Over the past centuries the ability to observe and collect data representing the physical world has been one of the great accomplishments of mankind. This includes observations ranging from the diminutive to the unimaginably large. Everything we know and everything we can possibly know is constrained in size and distance by two boundaries. The upper limit is given by the size of the observable universe with a comoving diameter of about 10^{27} meters. The lower limit is the Planck length, at which the structure of space-time is dominated by quantum effects, of 10^{-35} meters. Seamless positioning and navigation across this tremendous range of scales, roughly 60 orders of magnitude, generates many challenges to visualization systems. Besides the issue of creating visual metaphors for objects and their interrelated positions, based on different abstraction levels and contracted distance representations, one challenge not immediately obvious is the limited precision of

floating point numbers. Following a naïve approach, using a conventional computer graphics pipeline with a global coordinate origin at Earth's center, an object's location cannot be described with the necessary precision when viewing a spacecraft at the edge of our solar system, or viewing micrometer-scaled scans on the surface of Mars. The location of objects thus need to be expressed in huge value ranges with great precision. Current computer hardware is, however, limited to hardware accelerated computations of single and double precision floating point numbers, which are insufficient to represent microscopic objects and cosmological distances simultaneously. This lack of precision causes well-known visual rendering artifacts, such as displaced vertices or z-fighting, and generates intricate problems when displaying volumetric content. Furthermore, overflow and underflow of the available value ranges results in undesirable clipping of near and far geometry.

In this work, we propose a general method, a dynamic scene graph, which enables fast and accurate scaling, positioning, and navigation without significant loss of precision. The concept is applicable to a variety of application domains requiring the combination of large distances and precise rendering, including astronomical and molecular visualizations. While being suitable for interactive visualization, other possible applications include 3D modelling tools, movie production software and video games.

The method is based on an analysis of the sources of precision errors and range overflow problems in a conventional computer graphics pipeline. Using interval arithmetic to track floating point rounding errors, we are able to identify sets of operations that may introduce precision problems. We combine this information with the observation that high level of detail is only required for nearby objects. The underlying technique relies on dynamic updates and traversals of a scene graph where cameras automatically attach and detach to the closest object of interest. The new object of interest is then used as the new coordinate origin and thus automatically ensures the highest possible numerical precision for salient objects. Furthermore, the scene graph can be updated to allow scene graph nodes to be dynamically attached to different parts of the graph. This enables objects to be represented with high precision at multiple places, for example when a spacecraft is orbiting multiple planets during its lifetime. This approach enables an easy integration into existing scene graph implementations.

The dynamic scene graph has been implemented in the open source software project *OpenSpace* [Git]. OpenSpace has the goal of interactively visualizing and contextualizing a range of astrophysical data, including the most recent updates to databases, and even concurrent visualization of captured and simulated data. The software framework is primarily designed for immersive environments, such as dome theaters at planetariums, and it is targeting public engagement in science. Examples of data that can be concurrently visualized cover a wide range from sub-atomic particle simulations, via micrometer-scale discoveries made by rovers on the Martian surface, to volumetric simulations of entire galaxies.

Many state-of-the-art theaters and planetariums support stereoscopic viewing and recently the interest in immersive visualization in VR environments has increased. Thus, software needs to support stereoscopic rendering and the ability to render on multi-pipe systems and complex display configurations. This poses another challenge when dealing with multiple scales as the eye separation needs to be adjusted to the current scale of interest. Our dynamic scene graph approach offers a seamless solution to this problem.

It should be noted that our dynamic scene graph is applicable to any visualization tool that uses scene graphs to represent objects, and is compatible with techniques for volumetric rendering as well as order-independent transparency techniques. In summary, our contributions include:

- A *Dynamic Scene Graph* approach for representing objects and cameras to avoid precision-related rendering artifacts.
- A general method for rendering objects with a wide depth range and precision without the explicit need for near and far planes.
- A scheme for seamless adaption of the eye-separation used for stereoscopic rendering.
- An analysis of floating point precision errors using interval arithmetic.
- A method for dynamically updating the parent/child relationship for scene graph nodes.

The paper is structured as follows: After describing the related work in the following section, we provide a theoretical background for the sources of floating point inaccuracies in Section 3, followed by a description of our proposed dynamic scene graph method in Section 4 and its results in Section 5.

2. Related work

Computer graphics and interactive visualization are immensely valuable tools for communicating scientific findings and contextualizing information. Notable examples of applications in which huge scale differences are visualized include movie productions, such as *Powers of Ten* by Eames [EE77] from 1977, as well as interactive software, such as Uniview [KHE*10], DigiStar [Eva16], or DigitalSky [Sky16], which are based on curated datasets containing information about our universe [Abb06]. Virtual reality environments such as planetariums [MSK*10, LAE*01] have always been in the focus of educators, but a shift towards supporting data inspection on consumer grade hardware is ongoing [NPH*09].

Fu *et al.* [FHW06] categorized the problem of large scale navigation. *Techniques for travel* constrain the user input with the goal of supporting the travel towards a selected target, whereas *techniques for wayfinding* support the user in selecting a target by providing a spatial context. Our travel component is based on their description of a *Spatial Scaling Navigation Model* operating on a logarithmic camera model in which the navigation speed is dependent on the distance to the object of interest. In terms of *Techniques for wayfinding*, Li *et al.* [LFH06] introduce a world-in-miniature techniques based on logarithmic landmarks for navigation support in astronomical data. Combining this with the concept of power cubes, described in earlier works [FHW06], enables the creation of paths in this logarithmic space to aid in the large-scale navigation.

The concept of *power scaled coordinates* (PSC) or *power homogeneous coordinates*, introduced by Hanson *et al.*, addresses the travel challenge when creating animations showing large scale differences [HFW00]. The method uses four-dimensional floating point vectors where the first three components determine the direction and the fourth encodes a logarithmic scaling coefficient. The

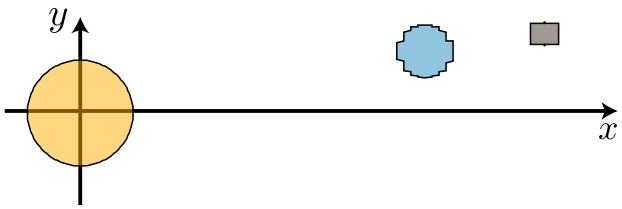


Figure 2: *An illustration of the effects of limited coordinate precision where the number of mantissa bits is limited to 5 when visualizing a system analogous to a world coordinate system in a scene graph. Increasing distance leads to decreasing precision.*

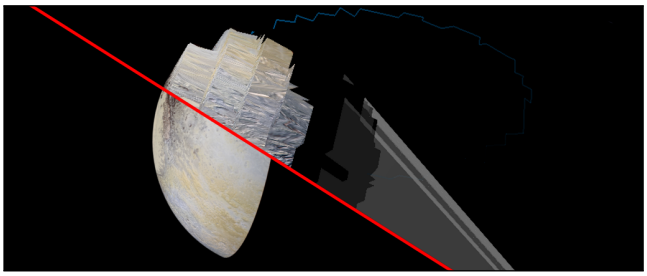


Figure 3: *Floating point error at Pluto. Floating point quantization are visible at Pluto's distance of 5.1 billion km from the Sun using PSC. Our method achieves high fidelity renderings while still representing objects with single precision floating point numbers.*

authors successfully create a powers-of-ten animation in which increasingly larger objects are shown with increasing distance to the Earth. This elegant solution is viable for scenes where the required precision decreases with distance from the origin, as it does not solve the problem of catastrophic cancellation and thus prohibits high-precision regions in large distances from the origin.

Fu and Hanson also introduced a depth buffer remapping to cover a wider range of distances than possible with a fixed point depth buffer and conventional near and far planes [FH07]. The depth buffer range is divided into regions where small and large values are remapped logarithmically. While this method is beneficial for some specific type of scene content, the method does not provide a generic way to select appropriate threshold values and the choice of separate regions with different mapping is cumbersome.

The *ScaleGraph*, introduced by Klashed *et al.* [KHE*10] as a part of the Uniview software, is the most similar to our proposed method. It employs a scene graph with sub-scenes to which cameras belong. The content of the current sub-scene is rendered in a local coordinate system with high precision and thus allowing multiple regions to be represented with high relative precision. Content outside the current sub-scene is translated onto the scene's boundary and scaled to compensate for perspective effects. This translation, however, leads to inconsistencies in stereoscopic rendering when switching scenes as objects jump from the bounding sphere of a scene to its physical location. As their presentation was focussed on the Uniview system as a whole, the detailed ScaleGraph descriptions are lacking and thus make it hard to reproduce.

Level of detail (LOD) and multiresolution methods are crucial when visualizing and navigating across large data sets [Lue03]. By using representations with fewer vertices and lower resolution textures to render distant objects, more resources can be directed towards rendering salient objects. However, these methods do not mitigate degradation of precision associated with conversions between coordinate frames.

3. Theoretical motivation

Improvements in hardware and software have led to optimized modern graphics processing units that work efficiently on floating points. In order to understand the precision and range limitations of the computer graphics pipeline, it is necessary to consider the properties of floating point numbers and available operations.

3.1. Floating point numbers

The IEEE754 standard for floating point numbers [ZCA*08], used in CPU computations as well as in graphics hardware, enables the representation of a large number range by encoding values as a sign, a mantissa, and an exponent. Single precision numbers use 1 sign bit, 8 exponent bits, and 23 mantissa bits. This encoding results in high precision close to zero, decreasing precision for larger numbers, and a dramatic increase in value range. One important characteristic of floating point numbers is their inability to represent all real numbers. All numbers not representable are rounded to their closest representable number. Increasing the floating points bit depth will decrease the precision error at the cost of computational resources due to lacking efficient hardware support.

The exponent range in 32-bit floats provides a possible range of approximately 76 orders of magnitude (OM). Exponents outside that range will cause exponent underflow or overflow and result in loss of data. While this range is larger than the 60 OM required for the known cosmos, problems arise when distances need to be squared and thus generating numbers in a range of 120 OM.

Due to the limited number of mantissa bits, values are rounded to be representable as floating point numbers. An upper bound of the spacing between a floating point number v and adjacent numbers is $\varepsilon|v|$, where $\varepsilon = 2^{-(n+1)}$ with n as the number of mantissa bits [Hig02]. For IEEE754 float, ε is $2^{-24} \approx 5.96 \cdot 10^{-8}$, leading to a precision of about 7 significant figures in decimal notation.

For computer graphics, this limits the precision with which vertices can be represented. Figure 2 shows a simulation with 5 mantissa bits. Using IEEE754 floats, at Pluto's distance from the Sun the distance between two floating point numbers is approximately $5.102 \cdot 10^9 \cdot 2^{-24}$ km $= 304.1$ km, which is about an eighth of Pluto's diameter and leads to visible artifacts (see Figure 3).

3.2. Interval arithmetic

The maximum error introduced by rounding of a number v can be written as $u|v|$, with the rounding error coefficient $u = \frac{\varepsilon}{2}$. Hence, v

may be rounded to a number $v' \in [v-u|v|, v+u|v|]$. Using interval arithmetic [HJVE01], this can be rewritten as $v(1+u[-1,1])$.

We use $\oplus$, $\ominus$, $\odot$, and $\oslash$ to denote floating point addition, subtraction, multiplication, and division respectively. The operators are commutative and yield the possible output intervals given two real number intervals $[x] = [x_1, x_2]$ and $[y] = [y_1, y_2]$ as,

$$\begin{aligned} [x] \oplus [y] &\subseteq ([x] + [y])(1 + u[-1,1]) \\ [x] \ominus [y] &\subseteq ([x] - [y])(1 + u[-1,1]) \\ [x] \odot [y] &\subseteq [x][y](1 + u[-1,1]) \\ [x] \oslash [y] &\subseteq \frac{[x]}{[y]}(1 + u[-1,1]) \end{aligned} \tag{1}$$

For operations involving the degenerate intervals $[0,0] = \{0\}$ and $[1,1] = \{1\}$ it holds that,

$$[x] \odot \{1\} = [x] \quad \text{and} \quad [x] \oplus \{0\} = [x] \tag{2}$$

3.3. Propagation of error intervals

Rounding errors propagate and compound when a result from one floating point operation is used in further computations. For this reason, expressions of the form $(a+b)-b$ may evaluate to 0 for a sufficiently small $a \neq 0$ and large b; a phenomena known as *catastrophic cancellation* (CC) [CVBK01]. We will show how this impacts any pipeline utilizing matrix multiplications or vector additions.

Let $\mathbb{E}$ be the set of intervals that can be written as $cu[-1,1]$, where c is a non-negative constant. For any pair of intervals $[e_1], [e_2] \in \mathbb{E}$, there exist intervals $[e_3], [e_4] \in \mathbb{E}$, such that $[e_1] + [e_2] \subseteq [e_3]$ and $[e_1][e_2] \subseteq [e_4]$. We let $[e_i] = c_i u[-1,1] \in \mathbb{E}$ and consider two intervals $a(1+[e_1])$ and $b(1+[e_2])$. First, we study the effects of floating point multiplication acting on these intervals.

$$\begin{aligned} a(1+[e_1]) \odot b(1+[e_2]) \\ \subseteq ab(1+[e_1])(1+[e_2])(1+u[-1,1]) \subseteq ab(1+[e_3]) \end{aligned} \tag{3}$$

The output interval in Equation 3 has the size $|abc_3u|$, meaning that the maximum absolute rounding error of floating point multiplication is proportional to the absolute value of the product ab. However, the relative rounding error c_3u is independent of these factors. We proceed with the properties of floating point addition.

$$\begin{aligned} a(1+[e_1]) \oplus b(1+[e_2]) \\ \subseteq \big(a(1+[e_1]) + b(1+[e_2])\big)(1+u[-1,1]) \\ \subseteq a(1+[e_4]) + b(1+[e_5]) \end{aligned} \tag{4}$$

Equation 4 yields a maximum rounding error of $|ac_4u| + |bc_5u|$. The relative error of the floating point sum depends on the individual terms' absolute value. By applying the equation iteratively, we see that Equation 5 holds regardless of the order of operations, with $\sum_i [x_i] = [x_1] \oplus [x_2] \oplus \ldots \oplus [x_n]$ and $[x_i] = x_i(1+[e_i])$.

$$\sum_i [x_i] \subseteq \sum_i [x_i](1 + [e_i']) \tag{5}$$

Thus, floating point sums will accumulate errors from all contributing terms with the risk for CC if large positive and negative terms are involved with relatively small expected results.

Given a matrix X with components X_{ij} and an interval matrix $[E]$ with components $[E_{ij}]$, let $[X^E]$ denote a matrix with the interval components $X_{ij}(1+[E_{ij}])$. We use the operator $\otimes$ to denote matrix multiplication, in which output components are computed as sums of products. By combining Equations 3 and 5, we get,

$$\begin{aligned} ([A^E] \otimes [B^F])_{ij} &\subseteq \sum_{k=1}^{m} A_{ik}(1+[e_{ij}]) \odot B_{kj}(1+[F_{kj}]) \\ &\subseteq \sum_{k=1}^{m} A_{ik}B_{kj}(1+[e_{ijk}]) \end{aligned} \tag{6}$$

with all interval matrix components $[E_{ij}], [F_{ij}] \in \mathbb{E}$ and $e_{ijk} \in \mathbb{E}$. We observe that the maximum floating point error of a component $(AB)_{ij}$ of a matrix product is given by $\sum_{k=1}^{m} |A_{ik}B_{kj}[e_{ijk}]|$. Using the special cases from Equation 2, we note that $e_{ijk} = 0$ if $[A_{ik}^E] = \{0\}$, $[B_{ik}^F] = \{0\}$, $[A_{ik}^E] = \{1\}$ or $[B_{ik}^F] = \{1\}$. Depending on the structure of the matrices A and B, the output component may be subject to CC.

3.4. Precision-related rendering artifacts

Lack of precision introduces several types of visual artifacts. If errors in the x and y coordinates of the normalized device coordinates reach or exceed the size of a pixel, a vertex displacement is visible. If the rounding error of z in the depth buffer is larger than the difference of two fragments, there is a risk for z-fighting.

In the following sections, we study the implications of the coordinate operations in the graphics pipeline to determine the circumstances in which precision problems may be introduced. By studying the effect of transformation matrices acting on coordinate vectors, we can predict vertex displacement issues and determine when there is a risk of z-fighting.

Content is commonly organized in a *scene graph* representing coordinate transformations hierarchically. Each node in the graph can contain transformations affecting all children, which means that the transformation matrices of all ancestor nodes are concatenated to a final *model matrix*. The interval vector $[\mathbf{n}]$ of normalized device coordinates is derived from the model coordinates $[\mathbf{x}]$ by

$$\begin{aligned} [\mathbf{c}] &= [P] \otimes [V] \otimes [M] \otimes [\mathbf{x}] \\ [\mathbf{n}] &= [\mathbf{c}] \oslash [\mathbf{c}_w] \end{aligned} \tag{7}$$

where $[P]$ is the projection matrix, $[V]$ is the view matrix, $[M]$ is the model matrix, and $[\mathbf{x}]$, $[\mathbf{c}]$ and $[\mathbf{n}]$ are interval vectors. The error interval size of the normalized device coordinates depends on the collective effects of $[M]$, $[V]$, $[P]$, and the perspective division as they act on $[\mathbf{x}]$.

$[M]$ and $[V]$ are composed from three types of transformations: scaling, translation, and rotation. We study the effects of applying scaling, translation, and rotation on a generalized interval model matrix or view matrix $[A]$, composed of a translation vector $[\mathbf{a}]$ and a combined rotation/scaling 3×3-matrix $[A']$, as given by Equation 8. Scaling, rotation, and translation matrices can all be written on the same form as $[A]$. In order to observe how the matrices act on a

coordinate vector, we study the fourth column in $[A]$.

$$[A] = \left(\begin{array}{c|c} [A'] & [\mathbf{a}] \\ \hline \{\mathbf{0}\} & \{1\} \end{array}\right) \tag{8}$$

After evaluating the possible effects of the scaling, translation, and rotation we similarly analyze the projection matrix, the perspective division, and the mapping to a depth buffer representation.

3.4.1. Scaling

Consider a 4×4 interval matrix $[S]$, analogously to the matrix $[A]$ composed of the 3×3 interval matrix $[S']$, interval vector $[\mathbf{s}]$, and a bottom row of degenerate intervals. Let the components of $[S]$ be scaling coefficient intervals, and other components of $[S]$ and $[\mathbf{s}]$ equal the degenerate interval. Equation 9 expresses the components of the output matrix $\hat{S}$, which is also a matrix of the form of $[A]$.

$$[\hat{S}_i] = ([S'] \otimes [A'])_{ij} \subseteq [S'_{ii}]\,[A'_{ij}](1 + [e_{ij}]) \tag{9}$$

$$[\hat{\mathbf{s}}_i] = ([S'] \otimes [\mathbf{a}])_i \subseteq [S'_{ii}]\,[\mathbf{a}_i](1 + [e_j]) \tag{10}$$

The rounding error introduced by an interval scaling matrix $[S]$ is thus proportional to the size of the scaling factors in $[S']$. Any error interval in $[A']$ and $[\mathbf{a}]$ will be scaled proportionally. Thus, we conclude that applying a scaling matrix to a matrix $[A]$ *does not cause a significant loss of relative precision in any matrix component*.

3.4.2. Translation

A corresponding analysis is made for translation matrices, composed of $[T']$ and $[\mathbf{t}]$, with degenerate intervals $[T'_{ij}] = \{I_{ij}\}$, where I is the identity matrix. The components of $[\mathbf{t}]$ representing a translation interval vector.

$$[\hat{T}_{ij}] = ([T'] \otimes [A'])_{ij} = [A'_{ij}] \tag{11}$$

$$[\hat{\mathbf{t}}_i] = [\mathbf{t}_i] \oplus [\mathbf{a}_i] \subseteq ([\mathbf{t}_i] + [\mathbf{a}_i])(1 + [e_i]) \tag{12}$$

We note that the precision in $[\hat{T}_{ij}]$ is preserved from $[A']$ when any translation matrix $[T]$ is applied. However, error intervals in $[\mathbf{t}_i]$ and $[\mathbf{a}_i]$ will propagate to $[\hat{t}]_i$, and will *not* be scaled down even if $|[\hat{\mathbf{t}}_i]|$ is much smaller than $|[\mathbf{t}_i]|$ and $|[\mathbf{a}_i]|$. This scenario *may lead to catastrophic cancellation* in the component $[\hat{\mathbf{t}}_i]$.

3.4.3. Rotation

Using a rotation interval matrix $[R]$ composed of the upper left 3×3 matrix $[R']$ and a upper right 3-dimensional column vector $\mathbf{r}$ with the interval components $[R'_{ij}] \subseteq [-1,1]$, and $\mathbf{r}_i = [0,0]$, we study the effect of multiplying $[R]$ with the interval matrix $[A]$:

$$[\hat{R}'_{ij}] = ([R'] \otimes [A'])_{ij} \subseteq \sum_{k=1}^{3} [R'_{ik}]\,[A'_{kj}](1 + [e_{ijk}]) \tag{13}$$

$$[\hat{\mathbf{r}}_i] = ([R'] \otimes [\mathbf{a}])_i \subseteq \sum_{k=1}^{3} [R'_{ik}]\,[\mathbf{a}_k](1 + [e_{ijk}]) \tag{14}$$

The potential error of any output component A_{ij} will be equal to $\sum_{k=1}^{3} |[R'_{ik}]\,[A'_{kj}][e_{ijk}]|$. Given that the intervals in $[R]$ are confined to $[-1,1]$, we see that the maximum possible precision loss in any matrix component is dominated by the values of A. When applying $[R]$ on a matrix or vector with both large and close-to-zero components, the precision in the small components may be lost. However, the loss of precision in the most significant matrix components will be minimal. This means that when using $[R]$ to transform the coordinate vector $[\mathbf{a}] = ([x],[y],[z])^t \subseteq [\hat{\mathbf{a}}][|\mathbf{a}|]$, neither the precision of the direction $[\hat{\mathbf{a}}]$ nor the magnitude $[|\mathbf{a}|]$ will be affected significantly. The same is true for the column vectors in $[A']$, which means that the significant properties of $[A']$ are preserved in $[\hat{R}']$.

3.4.4. Perspective and depth

When a perspective matrix P is multiplied with a homogeneous view coordinate vector $\mathbf{v} = (x,y,z,1)^t$, the sums of products to compute the output x, y, and w components will only consist of one non-zero term. Hence, there is no significant loss of relative precision in these components.

Furthermore, while the perspective division shrinks the projected image of distant objects, this also has the effect of scaling down any precision errors in distant vertex coordinates. Hence, the sensitivity of visible vertex displacement of a vertex is inversely proportional to the distance of the vertex in view space.

The z-component in clip space $c_z = (P\mathbf{v})_z$ is given by Equation 15, where p_n and p_f are the distance from the camera to the near and far plane respectively.

$$\lambda_1 = -\frac{p_f + p_n}{p_f - p_n},\ \lambda_2 = \frac{2p_f\,p_n}{p_f - p_n},\ [\mathbf{c}_z] = [\lambda_1] \otimes [\mathbf{v}_z] \oplus \lambda_2 \tag{15}$$

The result is an affine mapping of depths such that vertices in the near plane yield $\mathbf{c}_z = 0$ and vertices in the far plane give $\mathbf{c}_z = \mathbf{c}_w$. Subsequently, the perspective division maps $[\mathbf{n}_z]$ into the range $[0,1]$. If an integer depth buffer is used, this non-linear mapping results in a higher depth resolution for close objects than for distant ones. Setting p_n to a small number may cause distant vertices to collapse to the same depth buffer value and cause z-fighting.

The problems of depth buffer precision are well known and have been subject to thorough analysis [CR11]. Upchurch and Desbrun [UD12] show that good depth resolution can be maintained even with a far plane placed in ∞. However, precision problems for distant objects remain an issue for small p_n.

4. Dynamic Scene Graph Method

The need for accurate, effective and seamless positioning and navigation, together with the analysis of precision inaccuracies described above, form the basis for our proposed *Dynamic Scene Graph* (DSG) that enables high-fidelity rendering at any point in the scene graph independent of the location of the global coordinate system origin. This is achieved by utilizing a *dynamic camera attachment* and a *relative scene traversal* performed with respect to a camera's attachment node (AN). The AN is a regular scene graph node with no other special characteristics and is updated accordingly when the camera moves. The initial attachment node for each

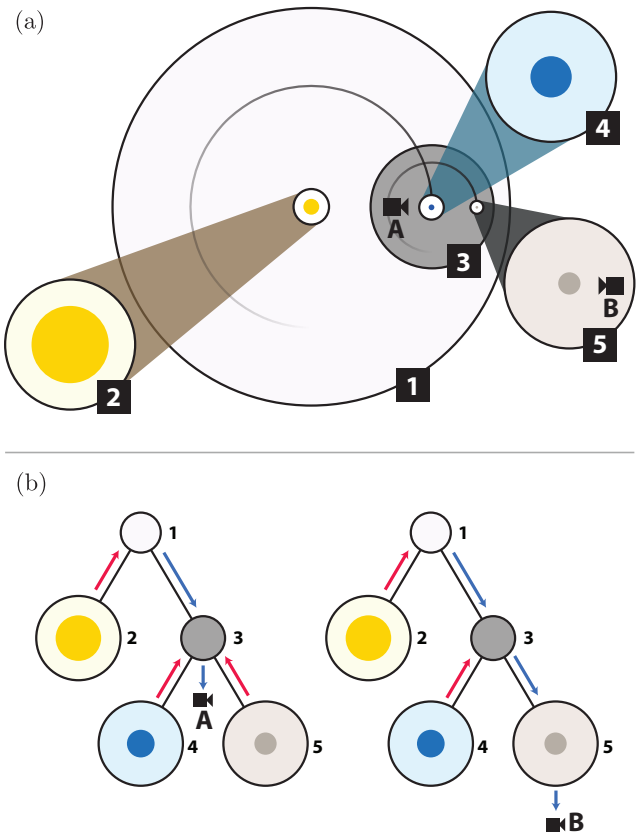


Figure 4: *A potential scene graph consisting of two translation nodes for the solar system and the Earth system (1, 3) and three rendering nodes for the Sun (2), Earth (4), and the Moon (5). Cameras (A, B) can be attached to different nodes, thus avoiding the introduction of large, error-prone translation values. (a) shows the nodes relative location, whereas (b) shows their organization in a graph. The arrows represent local upwards (red) and downwards (blue) transformations.*

camera is determined by specifying the node and a position in its local coordinate system.

The DSG is based on a traditional scene graph structure, but instead of a regular depth-first traversal order that starts at the root node of the graph, we propose a traversal order that starts at the AN and can traverse both upwards and downwards. Each node in the scene graph can contain a *transformation*, consisting of a *translation*, *rotation*, and *scaling* that are all described relative to a node's parent and must be invertible, which is possible for all transformation matrices. In addition, nodes may contain a visual object representation, for example a planet or a spacecraft, specified in the node's own local coordinate system. If a scene graph node contains a visible object, we refer to it as a *rendering node*, otherwise it is a *transformation node*. Unlike traditional scene graphs, each scene graph node has a radial *extent*, which determines the node's sphere of influence. Each scene graph node's extent has to be bigger than all its children's spheres of influence.

Figure 4 shows a possible scene graph with 5 nodes and their extents; (1, 3) are transformation nodes centered on the solar system and the Earth barycenter respectively, (2, 4, 5) are rendering nodes containing the Sun, the Earth, and the Moon. Two cameras (A, B) are attached to the scene at different attachment nodes.

First, we describe the relative scene traversal assuming a static camera, then describe the attachment changes of moving cameras.

4.1. Dynamic Scene Graph Traversal

As described in the previous section, the camera is attached to a specific node N in the scene graph. Rendering, or otherwise accessing, the contents of the scene graph requires a traversal, which in a traditional scene graph starts at the root node and traverses through the child nodes. In our method, this traversal starts at the attached node N instead, with the ability of moving up and down in the scene graph and inverting the relative transformations if necessary.

The traversal of the scene graph works as follows: For each scene graph node M, the shortest path between M and the currently attached node N is computed in the graph. Figure 4 (b) shows two examples of this with the camera being attached to node 3 and node 5 respectively. For each step along this path, the transformation matrices are concatenated. If the transition is performed downwards towards the leaf nodes (blue arrows in Figure 4 (b)), the transformation matrices are concatenated analogous to traditional scene graphs. If the transition is performed up towards the root node of the graph (red arrows in Figure 4 (b)), the transformation is inverted before concatenated. As an example we use the scene graph depicted in Figure 4 (b). $v_{i,j}$ is the transformation from node i to j and $v_{j,i} = -v_{i,j}$ is the inverse. $v_{i,i}$ specifies the local transformation used by the node i. For camera A, the rendering of the Sun (node 2) is performed using the transformations $v_{2,1} \cdot v_{1,3} \cdot v_{3,3}$. The rendering of the Earth (node 4) however, is performed only with the translation $v_{4,3} \cdot v_{3,3}$, ignoring all other scene graph nodes and thus not affected by their values. For camera B, the transformation for the Sun is $v_{2,1} \cdot v_{1,3} \cdot v_{3,5} \cdot v_{5,5}$ and for the Earth $v_{4,3} \cdot v_{3,5} \cdot v_{5,5}$.

First, this means that when different cameras are present in the scene graph, each camera might have its own traversal path. More importantly, for two nodes N and M only the subtree with the closest common parent at the root needs to be traversed in order to retrieve all information necessary to transform M into the local coordinate system of N and vice versa. Without including the root node in every traversal, the location of the local subtree does not have an impact on the precision of the objects contained within, thus evading the potential of catastrophic cancellation as discussed in Section 3.4.2. This fact lies at the core of our proposed method as it leads to a high-precision rendering independent of the location of the nodes within the extended scene graph. By utilizing the shortest path between M and N we avoid the large translations which would otherwise be necessary if all the transformations originate from the root node and which would lead to precision problems.

This technique enables the highest precision for the node N with precision decreasing for growing distances from the center of N. As

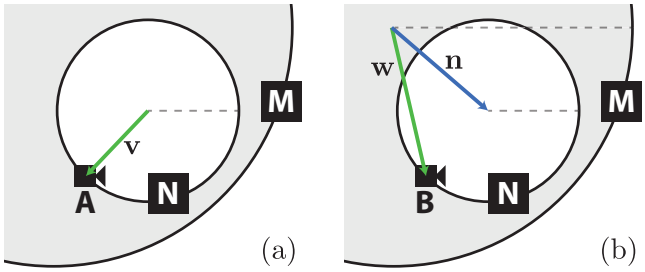


Figure 5: *When the camera reattaches to a node, its coordinate system changes. While the old location* A *is expressed relative to* N *as* $\boldsymbol{v}$*, the new location* B *is expressed relative to* M *as* $\boldsymbol{w} - \boldsymbol{n}$ *instead.*

described in Section 3, the representable precision decreases with increasing numbers, such that small details in large objects with great distances would become prone to precision error. However, projected changes that are smaller than a fraction of a pixel will not be visible. Therefore, our method only loses precision in the areas where high precision is not required.

Conceptually, our method pivots the scene graph such that the node N is placed at the root, all necessary transformation inversions are performed, and a traditional traversal algorithm is used. Since inversion of scaling, rotation and translation matrices are inexpensive and usually already a part of scene graph based rendering pipelines, there is little to no performance implication associated with the proposed traversal scheme.

4.2. Dynamic camera attachment

In our scene graph, a camera is always attached to a single node, the AN, and the camera's position and orientation is expressed in AN's local coordinate system. At all times, the AN is the deepest node in the scene graph whose sphere of influence fully encompasses the camera position. In Figure 4 camera A is located in nodes 1 and 3, and is thus attached to node 3. Likewise, camera B is inside nodes 1, 3 and 5 and is thus attached to node 5.

Figure 6 shows an example of the dynamic camera attachment in a flight from the surface of Earth to Saturn's moon Titan. The camera changes its AN four times on this transition.

There are two cases for updating a camera's AN in response to a user's interaction. If a camera enters the sphere of influence of a current AN's child, that child is the new AN — the camera moves down the scene graph. If the camera leaves the sphere of influence of the current AN, the AN's parent is the new AN — the camera moves up the scene graph. At each step, the distances of the camera to the AN center and the distances to all children are computed and specified *relative to the AN* as opposed to specifying the values relative to the global coordinate system origin defined by the root node. This circumvents the risk of catastrophic cancellation by ensuring that unnecessarily large vectors are not included.

In both transition cases, the location of the camera must change reference frames. Figure 5 shows a camera's old (A) and new (B) location before and after an AN change. A is expressed relative to N by the vector v, whereas B is expressed relative to M as the vector $\mathbf{w}$. The old and new locations are related by $\mathbf{w} = \mathbf{v} + \mathbf{n}$. To allow for this remapping to be computed without precision problems, extents of the scene graph nodes must be chosen such that the ratio between the child's and parent's extent is significantly larger than the machine epsilon. In order to circumvent rounding errors related to translation as described in Section 3.4.2, extents are chosen such that the camera attaches to a child render node before the details of the rendering become visible.

Similarly to the translation, the scaling and rotation of the camera has to be inverted by applying the inverse of the rotation of N towards M. As stated in Section 3.4.1 and Section 3.4.3 no significant precision error with the scaling and rotation will occur.

The update procedure is computationally inexpensive and scales linearly with the number of children of the current AN as only these need to be tested for reattachment. Hence, there are no significant performance implications associated with this operation.

4.3. Dynamic object attachment

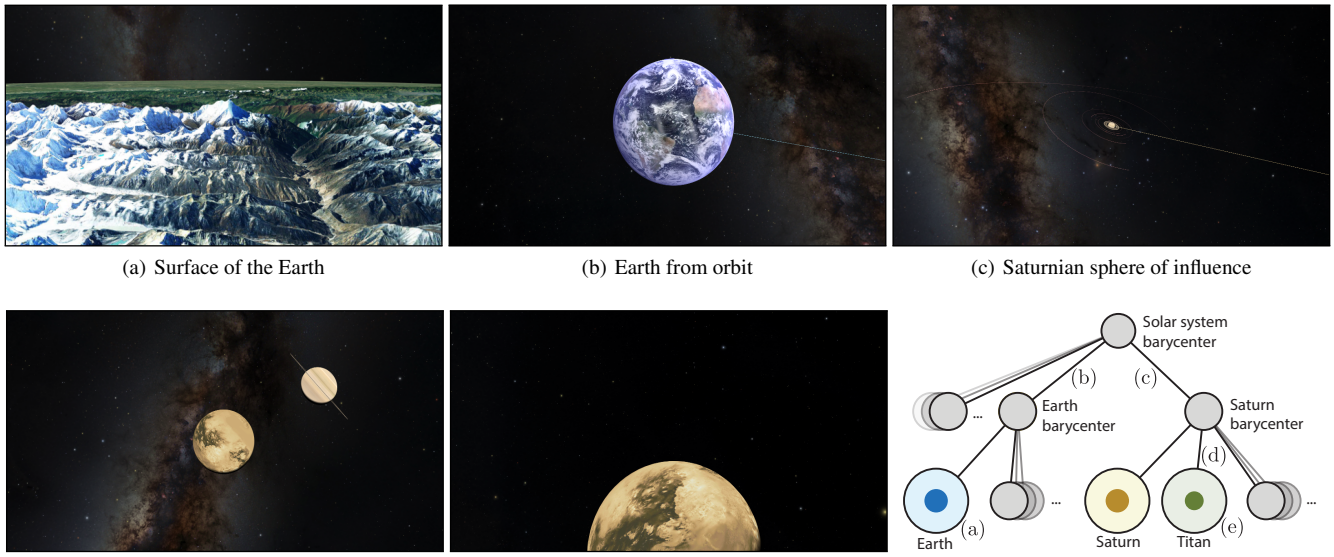
Analogous to the dynamic camera attachment, where the camera can be reattached to children or parent nodes, the same technique is also useful for scene graph nodes or entire sub trees. In a traditional scene graph, reattachment is not a useful feature as no precision is gained by moving a subtree at runtime. In our method however, it is beneficial when dealing with dynamic astronomical objects, such as spacecraft, asteroids, or comets, that can approach several objects of the scene graph at different times. One example is the New Horizons mission [Ste09], where the spacecraft launched from Earth and approached both Jupiter and Pluto. For this visualization, it is necessary to render the spacecraft at high resolution on the launchpad in Florida, in interplanetary space, during the gravity assist at Jupiter, on the flyby at Pluto, and eventually at its next flyby at 2014 MU69. If the scene graph node containing New Horizons would statically be a direct child of the solar system, floating point precision would not suffice for accurately representing the probe at these various locations. In Figure 1, the New Horizons spacecraft scene graph node has been dynamically attached to the Pluto barycenter node, in order to allow simultaneous high fidelity rendering of the probe as well as the planetary system.

4.4. Representation of depth

In our approach, we use a modified perspective matrix with $\lambda_1 = \lambda_2 = 0$, yielding $\mathbf{z}_c = 0$ for all vertices, thus effectively disabling the near and far planes. Geometry behind the camera is discarded due to the criterion of $-\mathbf{c}_w < \mathbf{c}_w$ imposed by the OpenGL pipeline. For depth comparisons, we use 32-bit floating point numbers representing the distance to the rendered fragment.

To avoid exponent overflow and underflow for distant and close fragments, we first divide the view coordinate vector v by its component with largest magnitude $\mathbf{v}_{\max}$, perform the computation using the pythagorean theorem, and multiply by $\mathbf{v}_{\max}$.

Depth sorting is performed using conventional z-buffering with



(a) Surface of the Earth (b) Earth from orbit (c) Saturnian sphere of influence

(d) Titan with Saturn in the background (e) Surface of Titan (f) Scene Graph Representation

Figure 6: *A journey of 1.65 billion km as sequence of images of a flight started on Earth's surface (a), showing Earth from the view of geostationary satellites (b), entering the Saturnian system (c), entering Titan's sphere of influence (d), and approaching Titan (e). Figure (f) shows the accompanying scene graph structure and and which locations the individual images were taken.*

a 32-bit floating point buffer. For scenes with volumetric or transparent content, we use an A-buffer technique for order-independent transparency [LFS*15]. When using ray casting for volumetric rendering, the distance taken along the ray is compared to the distance of rendered geometry in order to perform early ray termination.

By always rendering objects in their physical location, as opposed to the reprojection of objects performed in the ScaleGraph approach, our method handles stereoscopic rendering.

5. Results

The Dynamic Scene Graph approach has been deployed in several use case scenarios exemplifying the utility of the approach. We have here chosen to provide some selected examples with corresponding rendered images showing: dynamic objects, scaled navigation, globe browsing, and contextualization of scientific simulations. The DSG has been integrated into the OpenSpace software framework [BMK*15], which is targeting science communication events in immersive environments, such as dome theaters and planetariums. The implementation has enabled seamless navigation and accurate rendering in OpenSpace, without any measurable performance impact when compared to rendering with a static scene graph based on PSC as shown in Figure 3. The images shown are captures from interactive sessions using OpenSpace. While the use-cases in this section are dealing with astronomical datasets, our proposed method equally applies to other domains with large scale differences, such as molecular visualization or atomic visualization. In this section, we shortly describe these use-cases and how our proposed method was used in four categories.

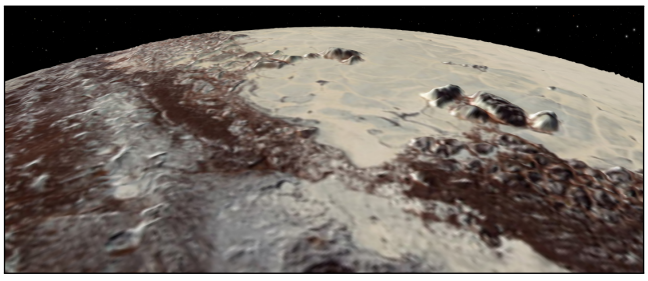


Figure 7: *A rendering of Pluto's surface with a height field reconstructed from New Horizons' images showing the Zheng He Montes and al-Idrisi Montes in front of Tombaugh Regio.*

Spacecraft. As one of the major motivations for developing this method, spacecraft are inherently small objects that need to be visualized precisely with respect to an object, mostly planets, that they are inspecting. One example is shown in Figure 1 where the New Horizons spacecraft is shown at 80 Mm distance to Pluto and around $6 \cdot 10^{12}$ m away from the Sun. As described in Section 3.1, floating point numbers only have a precision of about 300 km at the distance of Pluto, which would make a traditional approach infeasible. With our method, however, it is possible to show the entire mission with the required precision. A second spacecraft is ESA's Rosetta shown in Figure 10 as a stereoscopic rendering of the separation of the lander Philae from Rosetta on its way to the comet.

Navigation. Figure 6 shows frames of a seamless navigation through our solar system. Starting on the surface of Earth, where the Alps are rendered with their correct height and surface textures,

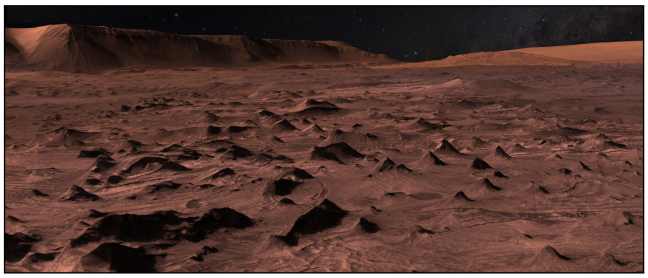


Figure 8: *Details of structures with up to 25 cm per pixel resolution in a rendering of the surface features in West Candor Chasma on Mars as captured from the HiRISE camera.*

the Earth becomes visible further out before reaching the sphere of influence of Saturn and continuing to the surface of Titan, showing color images of the moon's surface. At no point during this 1.65 billion km user-controlled journey are artifacts visible due to the camera's seamless transitions between the scene graph nodes.

Virtual globes. High resolution surface features are a big challenge when representing an entire solar system in a single scene graph. In addition to requiring a level of detail scheme to support dynamic loading of terrain data into the graphics memory, coordinate precision also needs to be maintained across the whole visualization pipeline. Individual surface features are much smaller than the available floating point precision for any planet. With our method however, we can produce an accurate rendering of the latest composite images and height maps on Pluto [SBE*15] with a resolution of between 2 and 5 km/pixel (see Figure 7). Figure 8 shows detailed structures on the surface of Mars as returned from the Mars Reconnaissance Orbiter's HiRISE camera [MEB*07]. It shows details with a resolution of 25 cm per pixel.

Volumetric data. As described in Section 4.4, our system enables the seamless integration of volumetric data. Figure 9 shows a frame of a three-channel time-varying volumetric rendering of the Milky Way [JTK10, FBS*11]. In this scene, the center of the Milky Way is the root node of the scene graph and the solar system are at a distance of $2,5 \cdot 10^{20}$ from the center inside the volume's bounding box of $\approx 10^{21} \times 10^{21} \times 1.5 \cdot 10^{19}$ m. The volume is rendered using ray casting, with its proxy geometry defined as a sibling node to our solar system, sharing a Milky Way group node as common parent. This means that only the transformations of the bounding box vertices have to be adjusted in order to incorporate volume rendering.

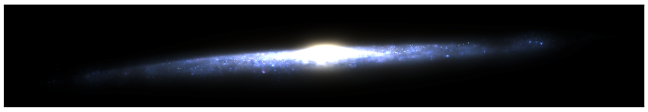


Figure 9: *A volumetric rendering of a three-channel simulation of the Milky Way with a bounding box size in the order of* 10^{20} *m.*

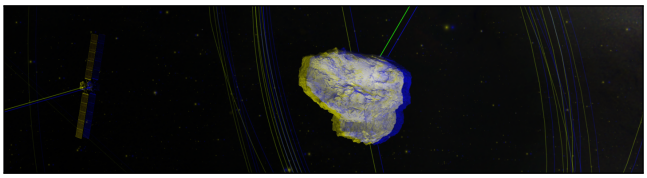


Figure 10: *Our approach supports stereoscopic rendering (here, amber/blue anaglyph) exemplified with the Rosetta/Philae separation for the landing on the comet 67P/Churyumov-Gerasimenko.*

6. Conclusions

In this work, we presented a method that supports high-precision transformations in a scene graph by dynamically attaching a camera to its closest node and using this node as the origin for the scene traversal. The dynamic scene graph enables a higher level of relative precision than which is otherwise possible. By supporting efficient dynamic reattachment of the camera to nodes, it becomes possible to navigate a scene encompassing scales and distances much larger than floating point precision would normally permit. The same reattachment technique can be applied to provide other scene graph nodes with the ability to retain high precision at multiple locations, for example, a spacecraft that requires high precision at multiple planets as it moves through space. The scene graph traversal uses the shortest path to every other scene graph node and thus circumvents potentially large translation values that would otherwise lead to precision degradation during the rendering.

We implemented the method in the open-source software platform OpenSpace which has the goal of visualizing and contextualizing a wide range of astrophysical data. The method was successfully tested on spacecraft mission visualization [BMK*15], on virtual globes, and volumetric rendering. The method is flexible enough, however, to be implemented and used in any scene graph implementation through modification of the traversal scheme.

6.1. Future Work

For future work, we like to include high-level wayfinding techniques such as the ones explored in Li *et al.* [LFH06] and generalize these to macro and microcosmos scenarios. For time-varying datasets, a large spatial extent usually corresponds to a large temporal extent as well, the handling of which requires further research. Furthermore, combining datasets of vast spatial domains opens up new challenges in the design of interaction techniques.

7. Acknowledgments

We would like to acknowledge the Swedish e-Science Research Center (SeRC) for their support of this work. Parts of this work were supported by NASA under award No NNX16AB93A, the Moore-Sloan Data Science Environment at NYU, NSF awards CNS-1229185, CCF-1533564, and CNS-1544753. We would also like to thank Karl Bladin and Erik Broberg for their work on the planetary renderer and ESRI for providing the Earth data. Additional thanks to Karljohan Lundin Palmerius and Patric Ljung for proofreading and giving insightful feedback.

References

[Abb06] ABBOTT B.: The digital universe guide for partiview, 2006. 2

[BMK*15] BOCK A., MARCINKOWSKI M., KILBY J., EMMART C., YNNERMAN A.: Openspace: Public dissemination of space mission profiles. *2015 IEEE Scientific Visualization Conference (SciVis)* (Oct 2015). doi:10.1109/scivis.2015.7429503. 8, 9

[CR11] COZZI P., RING K.: *3D engine design for virtual globes*. CRC Press, 2011. 5

[CVBK01] CUYT A., VERDONK B., BECUWE S., KUTERNA P.: A remarkable example of catastrophic cancellation unraveled. *Computing 66*, 3 (May 2001), 309–320. doi:10.1007/s006070170028. 4

[EE77] EAMES C., EAMES R.: *Powers of Ten*. IBM, 1977. 2

[Eva16] EVANS & SUTHERLAND: Digistar, 2016. URL: http://www.es.com/Digistar/. 2

[FBS*11] FUJII M., BABA J., SAITOH T., MAKINO J., KOKUBO E., WADA K.: The dynamics of spiral arms in pure stellar disks. *The Astrophysical Journal 730*, 2 (2011), 109. 9

[FH07] FU C.-W., HANSON A. J.: A transparently scalable visualization architecture for exploring the universe. *IEEE Transactions on Visualization and Computer Graphics 13*, 1 (2007), 108–121. 3

[FHW06] FU C.-W., HANSON A. J., WERNERT E. A.: Navigation techniques for large-scale astronomical exploration. In *Electronic Imaging 2006* (2006), International Society for Optics and Photonics. 2

[Git] GITHUB:. OpenSpace Repository [online]. URL: https://www.github.com/OpenSpace/OpenSpace.git. 2

[HFW00] HANSON A. J., FU C.-W., WERNERT E. A.: Very large scale visualization methods for astrophysical data. In *Data Visualization 2000*. Springer, 2000, pp. 115–124. 2

[Hig02] HIGHAM N. J.: *Accuracy and stability of numerical algorithms*, 2 ed. Siam, 2002, p. 37. 3

[HJVE01] HICKEY T., JU Q., VAN EMDEN M. H.: Interval arithmetic: From principles to implementation. *Journal of the ACM 48*, 5 (Sep 2001), 1038–1068. URL: http://dx.doi.org/10.1145/502102.502106, doi:10.1145/502102.502106. 4

[JTK10] JUNICHI B., TAKAYUKI R. S., KEIICHI W.: On the interpretation of the l–v features in the milky way galaxy. *Publications of the Astronomical Society of Japan 62*, 6 (2010), 1413–1422. 9

[KHE*10] KLASHED S., HEMINGSSON P., EMMART C., COOPER M., YNNERMAN A.: Uniview - Visualizing the Universe. In *Eurographics - Areas Papers* (2010), Eurographics Association. 2, 3

[LAE*01] LIU C. T., ABBOTT B., EMMART C., MAC LOW M.-M., SHARA M., SUMMERS F. J., TYSON N. D.: 3-d visualizations of massive astronomy datasets with a digital dome. In *Virtual Observatories of the Future* (2001), vol. 225, p. 188. 2

[LFH06] LI Y., FU C.-W., HANSON A.: Scalable wim: Effective exploration in large-scale astrophysical environments. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1005–1012. 2, 9

[LFS*15] LINDHOLM S., FALK M., SUNDÉN E., BOCK A., YNNERMAN A., ROPINSKI T.: Hybrid data visualization based on depth complexity histogram analysis. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 74–85. 8

[Lue03] LUEBKE D. P.: *Level of detail for 3D graphics*. Morgan Kaufmann, 2003. 3

[MEB*07] MCEWEN A. S., ELIASON E. M., BERGSTROM J. W., BRIDGES N. T., HANSEN C. J., DELAMERE W. A., GRANT J. A., GULICK V. C., HERKENHOFF K. E., KESZTHELYI L., ET AL.: Mars reconnaissance orbiter's high resolution imaging science experiment (HiRISE). *Journal of Geophysical Research: Planets 112*, E5 (2007). 9

[MSK*10] MAGNOR M., SEN P., KNISS J., ANGEL E., WENGER S.: Progress in rendering and modeling for digital planetariums. *Proc. Eurographics Area Papers* (2010), 1–8. 2

[NPH*09] NAKASONE A., PRENDINGER H., HOLLAND S., HUT P., MAKINO J., MIURA K.: Astrosim: collaborative visualization of an astrophysics simulation in second life. *IEEE Computer Graphics and Applications 29*, 5 (2009), 69–81. 2

[SBE*15] STERN S., BAGENAL F., ENNICO K., GLADSTONE G., GRUNDY W., MCKINNON W., MOORE J., OLKIN C., SPENCER J., WEAVER H., ET AL.: The pluto system: Initial results from its exploration by new horizons. *Science 350*, 6258 (2015). 9

[Sky16] SKY-SKAN INC.: Digitalsky 2, 2016. URL: http://www.skyskan.com/products/ds. 2

[Ste09] STERN S. A.: The new horizons pluto kuiper belt mission: An overview with historical context. In *New Horizons*. Springer, 2009, pp. 3–21. 7

[UD12] UPCHURCH P., DESBRUN M.: Tightening the precision of perspective rendering. *Journal of Graphics Tools 16*, 1 (2012), 40–56. 5

[ZCA*08] ZURAS D., COWLISHAW M., AIKEN A., APPLEGATE M., BAILEY D., BASS S., BHANDARKAR D., BHAT M., BINDEL D., BOLDO S., ET AL.: IEEE standard for floating-point arithmetic. *IEEE Std 754-2008* (2008), 1–70. 3