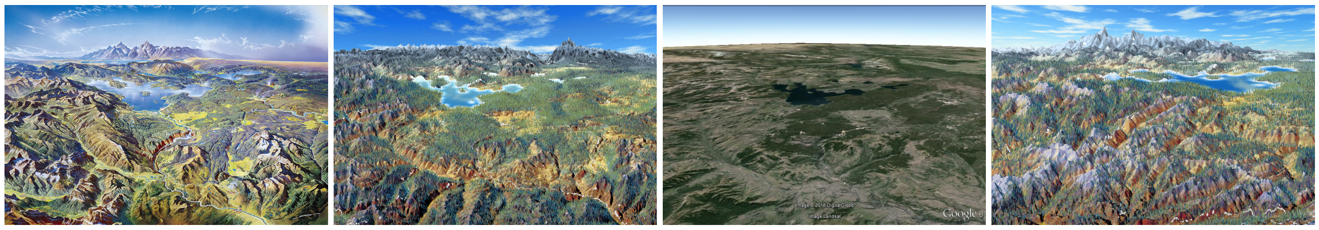
# Real-Time Panorama Maps

S. Alex Brown
University of Calgary
acbrown5@gmail.com

Faramarz Samavati
University of Calgary
samavati@ucalgary.ca



(a) (b) (c) (d)

**Figure 1: Panorama maps of Yellowstone National Park. (1a) Painting by Heinrich Berann. (1b) Real-time panorama map generated automatically by our system from real data. (1c) View of Yellowstone in Google Earth. (1d) A view of the Teton Range.**

## ABSTRACT

Panorama maps are stylized paintings of terrain often seen at tourist destinations. They are difficult to create since they are both artistic and grounded in real geographic data. In this paper we present techniques for rendering real-world data in the style of Heinrich Berann's panorama maps in a real-time application. We analyse several of Berann's paintings to identify the artistic elements used. We use this analysis to form algorithms that mimic the panorama map style, focusing on replicating the terrain deformation, distorted projection, terrain colouring, tree brush strokes, water rendering, and atmospheric scattering. In our approach we use freely available digital earth data to render interactive panorama maps without needing further design work.

## CCS CONCEPTS

• **Human-centered computing** → **Geographic visualization**;
• **Computing methodologies** → **Non-photorealistic rendering**;

## KEYWORDS

panorama map, non-photorealistic rendering, digital earth, real-time rendering

**ACM Reference format:**
S. Alex Brown and Faramarz Samavati. 2017. Real-Time Panorama Maps. In *Proceedings of NPAR'17, Los Angeles, CA, USA, July 28-29, 2017,* 11 pages.
https://doi.org/10.1145/3092919.3092922

## 1 INTRODUCTION

Panorama maps are stylized paintings of terrain known for being both captivating works of art as well as useful tools for navigating a region. Panorama maps are characterized by their aerial viewpoint and exaggeration of important landmarks. Since they are easier to read than a traditional map they are often seen at parks, ski hills, and other tourist destinations to help visitors find their way around. An example of a panorama map of the Yellowstone National Park can be seen in Figure 1a.

Creating a good panorama map is difficult: it requires both accurate cartographic information of an area and the keen skill of an artist combined. These hurdles have been made less daunting by modern breakthroughs in geographical data gathering. Inspired by US Vice President Al Gore's ground-breaking speech [1998] titled "The Digital Earth", a global effort has been made to study and document our planet. Vast amounts of data, such as digital elevation models (DEMs), satellite imagery, and land cover surveys, are now freely available through government initiatives and Geographic Information Systems (GIS). A recent survey published by Ali Mahdavi-Amiri et al. [2015] showcases the breadth and depth of digital earth research and applications being developed. Data available through digital earth systems provide new opportunities for developing novel geographical visualization techniques specifically designed for real datasets.

The goal of this paper is to explore techniques for automatically generating panorama maps in real-time from real geographic data. The vision is to use our system as a rendering feature for digital earth systems (e.g. Google Earth) and allow users to interactively fly over a region which is rendered as a captivating panorama map. Figure 1b shows the results of our system rendering Yellowstone National Park. Figure 1c shows the same view in Google Earth. Panorama maps are grounded in cartography, so we base our renderings on data from digital earth systems. Additionally, we aim to automate the process of generating a panorama map based on the geographic data; no further artist or designer work should be necessary. This work focuses on recreating the visual style of the artist who invented the modern panorama map, Heinrich Berann [Troyer 2016].

In this paper we offer an analysis of panorama maps and introduce a suite of rendering algorithms to capture both the panorama map distortion and artistic style in a real-time system. This paper focuses on five key components of panorama rendering: terrain distortion, terrain colouring, tree brush strokes, water rendering, and atmospheric effects. We have achieved real-time performance on high resolution scenes containing more than 6,000,000 triangles. To ensure we capture the important aspects of the art style we analyze the artist's method, gather measurements and metrics from the paintings, and refer to other studies done in the field.

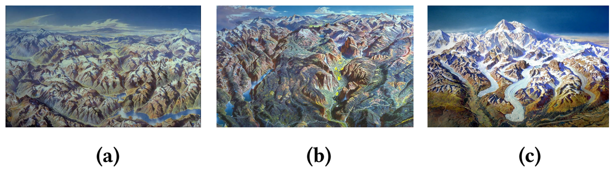


**Figure 2: Berann's U.S. National Park paintings. (2a) North Cascades National Park in Washington. (2b) Yosemite National Park in California. (2c) Denali National Park in Alaska.**

## 2 BACKGROUND

To start, we analyze some of Berann's panorama maps to identify key elements of the paintings. We then review some of the previous research done in this area.

### 2.1 Analysis

The basic elements used to analyze art are form, line, colour, space, and texture [ele 2016]. Each element can be seen as a choice made by a master artist; each choice is an opportunity to convey ideas to the viewer. Using his background in cartography, Berann created paintings that weren't just visually appealing, but were also grounded in actual geographic data [Patterson 2000]. So for each of these elements we need to ask if the choices made were for the style of the painting, or do they also serve a function for the underlying map. We will explore this idea in more depth throughout this paper.

Figures 1a and 2 show Berann's collection of panorama maps commissioned by the U.S. National Park Service. These striking images showcase all the features of Berann's famous style that we aim to replicate: the colours are rich and vibrant, the view is distorted, and important terrain features are exaggerated.

The terrain is distorted in a number of ways. The land is curved from the base of the painting towards the horizon. Patterson [2000] noted this effect is like what one would see from an aerial view by scanning from the land beneath them up to the horizon. The terrain is also distorted to exaggerate or emphasize various features. For example, mountains are depicted much taller than they would actually appear when viewed from such an elevation.

Several of Berann's maps feature a large portion of the world, however most focus on a specific region or park. Many paintings use the sky portion as a way to frame the scene. The sky typically occupies roughly the top quarter of the image. Most scenes feature more terrain than water, and in many cases the terrain is mountainous.

Patterson [2000] made a number of interesting observations on the composition of Berann's art. He noted that unlike traditional cartography, north orientation is not important in a panorama painting. Berann also favoured views that ran from lowlands near the bottom of the painting to highlands on the horizon. Patterson also discussed Berann's work-flow for creating panorama maps: For large scenes like Yellowstone National Park, Berann would start with a sketch based on a contour map. He would also use aerial photographs for reference, as they often helped to fill in details of the terrain. Berann created most of his paintings on thick white paper using water-soluble paints [Patterson 2000]. Brush strokes are clearly visible and used to add texture in certain areas such as forests.

The colours used in Berann's paintings are bright and eye-catching. Though the selected colours are not photorealistic, they are also not arbitrary. Colours for areas in light and in shade are selected to compliment each other. Additionally, terrain types follow fairly consistent palettes, making it easier for a viewer to identify terrain features. For example, grassy fields can easily be differentiated from forests and hills by their yellow hue.

This overview highlights a number of key features of Berann's art that we aimed to mimic. Section 3 explores these ideas further as they relate to each component of our rendering algorithm.

### 2.2 Related Work

A common goal within non-photorealistic rendering (NPR) is to mimic the style of art that has been drawn by hand. Both technical drawings and sketches use silhouette lines, and a great deal of research has been done to generate such images [Isenberg et al. 2003]. Introduced by Gooch [1998], cel shading (or toon shading) formed a foundation that has led to many other NPR inventions. One example is tonal art maps which are generated textures used to create renderings that look like sketches shaded with hatching [Praun et al. 2001]. These sort of techniques are sometimes referred to as a "painterly" techniques, as they simulate traditional art [Hertzmann 1998].

Another area of non-photorealistic rendering that has sparked many experiments is in view distortion. Rather than distorting a mesh directly, Brosz et al. [2007] distorted the scene's projection. Non-linear projections are typically seen in the form of fisheye or panorama photos, but they can be used for a much wider range of applications.

NPR is also useful for rendering terrain and generating maps. A paper by Kennelly et al. [2006] presents a survey of various NPR algorithms and their applications to cartographic techniques. One such application was presented in a paper by Visvalingam et al. [1998] which reported on a technique for sketching terrain based on profile features he dubbed "P-strokes". Döllner et al. [2003] also developed a sketch-style algorithm for rendering terrain, but with a focus on creating a virtual 3D city with a clean artistic style.

Other terrain rendering techniques focus more on generating maps that display a particular set of data. Semmo et al. [2012]

introduced a method for rendering a city map that uses level-of-detail to highlight important landmarks. In their system, a zoomed in view shows a full 3D representation of a city, while a zoomed out view shows a more abstract map with icons representing important landmarks. Trapp et al. [2008] tackle the problem with a different approach by defining particular lenses as areas to show higher levels of detail within a complex city model.

One of the earliest attempts at computer generated panorama maps was made by Premoze [2002]. In his paper, he described tools and techniques to aid an artist in the difficult process of creating a panorama map. For future work Premoze wished to further automate the process, noting that replicating the work of a traditional master was impossible, "for now".

Other early research on generating panorama maps focused on techniques for terrain deformation. Lorenz et al. [2008] experimented with multi-perspective views, demonstrating "bird's-eye" and "pedestrian-eye" distortions of city models. These distortions are done in real-time on a GPU by defining three zones. Möser et al. [2008] developed a system combining both bird's-eye (panorama) and pedestrian-eye view distortions with context aware enhancements to present a distorted map highlighting a selected path for navigation. Pasewaldt et al. [2011] experimented with blending between different terrain distortions depending on the user's chosen view.

Falk et al. [2007] experimented with terrain distortion using non-linear ray-tracing. This work generated panorama map terrain distortion by tracing increasingly curved rays through the scene. Combining this progressive perspective with localized edits in the terrain, the research aimed to prevent tall mountains from occluding portions of the terrain. Users can paint influence maps to adjust how vertical exaggeration and terrain distortion are applied. The main differences in our work are that we automate terrain distortion, and render distorted terrain in real-time.

Progressive perspective was further explored by Jenny et al. [2010] in their work on interactive 3D maps. The paper introduces the progressive-cylindrical projection in which a cylindrical (360° wide-angle) projection is combined with the curved view rays of progressive perspective. The main difference in our work is we also aim to recreate the artistic style of a painted panorama map.

Bratkova et al. [2009] took steps towards automatically generating images that mimic the artistic style of the panorama map. In their paper they analyzed the paintings of Berann and James Niehues, another panorama artist, and proposed a set of principles for rendering panorama maps. Their algorithm included methods for terrain deformation, exaggeration, shading and texturing. The results included renderings matching the style of both Berann and Niehues, however the technique was not conducive to real-time rendering.

A similar work was published by Degener and Klein [2009] which focused on the problem of terrain deformation. Degener demonstrated a method for deforming a terrain model to optimize the visibility of a given set of features. The results produced compelling renderings of ski hills in which mountains are warped to make all routes visible in the image. One challenge with this system is that important features and a viewpoint need to be predefined. The project was not real-time since the technique of maximizing visibility on a mesh for a given viewpoint is resource intensive.

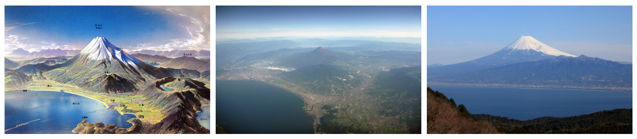
In summary, the challenge still stands to build a system that creates panorama maps automatically from available georeferenced data to be used as a rendering feature for interactive exploration in digital earth systems.

## 3 INTERACTIVE PANORAMA MAPS

Our goal is to create an interactive panorama map for a selected region. The first step, then, is to render 3D terrain in real-time. A mesh can easily be generated from terrain elevation data. Such data can be obtained in common digital earth systems. For best results it is important to obtain recent, high resolution data to reduce the appearance of sensor and discretization errors.

### 3.1 Distortion

*3.1.1 Analysis.* Terrain distortion is one of the most prominent features of panorama maps. We now examine some of Berann's paintings to determine his method and apply that to our rendering.



(a) (b) (c)

**Figure 3: Comparison of Berann's painting of Mount Fuji with some photographs from the region. (3a) Fuji by Heinrich Berann. (3b) Aerial view of Mount Fuji *(Photo by Paipateroma [2016], available under Creative Commons license).* (3c) View of Mount Fuji taken from Darumayama *(Photo by Tanaka Juuyoh [2016], available under Creative Commons license).***

Figure 1a compares Berann's painting of Yellowstone National Park with a view of the same area in Google Earth (Figure 1c). One noticeable aspect is that Berann exaggerated the height of the mountains in the region. Google Earth uses digital elevation data to render 3D terrain, but from a viewpoint this high above the earth's surface, actual elevation changes are barely visible. Another feature is that the curvature of the earth has been exaggerated. In the painting there is a larger variation in the angle from the surface at the bottom of the painting out towards the horizon than is present in the Google Earth rendering. Finally, both images use Yellowstone Lake as a central focal point, but the painting appears to cover more area in the foreground and less in the background when compared with the rendering.

First let's look at the terrain's vertical exaggeration. Consider Berann's painting of Mount Fuji shown in Figure 3a. This painting depicts a spectacular view of the mountain over the city of Numazu in Japan's Shizuoka Prefecture. Figure 3b [Paipateroma 2016] shows a photograph of Mount Fuji taken from nearly the same vantage point, once again confirming that the terrain has been exaggerated. The key question is if the exaggeration is arbitrary, or if it serves an important purpose.

Figure 3c [Tanaka 2016] is a photo of Mount Fuji taken from a lookout point on Mount Daruma, directly underneath the viewpoint of Berann's painting. The mountain dominates the horizon filling more than a third of the photo's field of view, just as it does in the painting. This indicates that Berann's aerial viewpoint gives his paintings the structure of a map, but individual elements are depicted as they would be seen by a normal observer on the ground to help them to be easily recognized.

At 3776.24 meters tall, Mount Fuji is the highest mountain in Japan. The other mountain featured in the photo in Figure 3c is Mount Ashitaka which measures 1457 meters tall. Even though Mount Ashitaka is roughly 40% of the height of Mount Fuji, due to the photo's perspective it appears to be roughly two thirds size of Mount Fuji. In contrast, Berann's painting ignores the effects of perspective scaling and depicts a more accurate relative scale between the two mountains.

This observation can be repeated in the Yellowstone painting in Figure 1a. The mountains depicted on the horizon are the Teton Mountain range, the tallest of which is Grand Teton standing at a prominence of 1990 meters. Centered in front of the Teton mountains, behind Lake Yellowstone you can see Mount Sheridan which stands 925 meters over Heart Lake to the east (left in the painting). The painting accurately depicts Mount Sheridan to be roughly half the height of Grand Teton. Similarly, Mount Washburn seen just below Yellowstone Falls has a prominence of 708 meters and is depicted at roughly a third of the height of Grand Teton.

Berann also appears to ignore perspective in selecting a viewing area. The area covered by the Yellowstone painting is a rectangle rather than the trapezoid we would expect from a perspective projection. This indicates Berann is using a sort of modified orthographic projection.

Patterson made a similar observation by examining Berann's work flow [Patterson 2000]. Patterson explained that Berann would typically make his first pencil sketches for a piece using topographic maps of the area. He would convert contour lines into 3D shapes, sometimes referring to aerial photos of the region for details. Patterson noted that Berann tended to use perspective more in smaller scenes than in larger ones. Berann likely based small scenes primarily on a photograph as opposed to larger scenes such as his National Park paintings which were based primarily on maps.

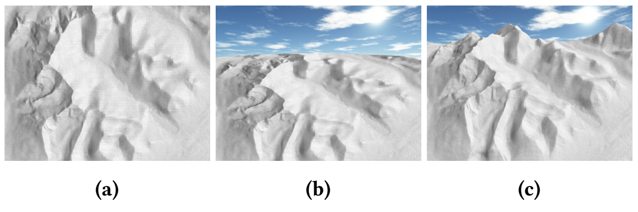


**Figure 4: Rendering distorted terrain. (4a) A top-down orthographic projection. (4b) Adding a curve. (4c) Panorama map style distortions.**

*3.1.2 Implementation.* We now look at how to turn the observations made above into an algorithm for real-time rendering. Consider the Lake Louise test data as shown in Figure 4a. The first step is to incorporate terrain curvature. Patterson noted that Berann's paintings resemble what one would see if they were to sweep their eyes from the land below their viewpoint upwards to the horizon [Patterson 2000]. This effect can be created by distorting the terrain along a ruled surface [Brosz and Samavati 2010].

In the case of our distorted terrain we start by wrapping the $x$-$z$ plane along a quadratic curve, and then apply the heightmap data by moving vertices up in view space, along the image plane's $y$ axis. This idea is demonstrated in Figure 5. The warping is done this way for two main reasons. First of all, applying the curve to the flat surface creates an even distortion along the vertical axis. Secondly, extruding in view space maintains the relative size of mountains as seen in the example above. This approach is similar to the one presented by Bratkova et al. [2009], which also involved wrapping the terrain's base along a quadratic curve. However, Bratkova extruded terrain along the wrapped surface's normal causing mountains near the base of the render to appear shorter due to a nearly top-down view. Additionally, they used a perspective projection eliminating a viewer's ability to measure the relative height of terrain features.

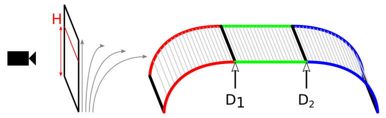


**Figure 5: Demonstration of terrain curvature.**

As shown in Figure 5, the parameters controlling the distortion are $D_1$, $D_2$, and $H$. $D_1$ and $D_2$ are effectively the scene's drawing distance and can be adjusted proportional to the size of the scene and the zoom level. $H$ indicates where the horizon line is in the resulting image and can be adjusted to change the look of a scene. The value of $H$ can be between 0 and 1, but to be consistent with Berann's paintings it should stay in the range of 0.7 to 0.8. Measuring a random sampling of Berann's panorama maps we found that the horizon line is on average roughly 24% from the top of the painting. In these measurements, the horizon line was considered to be the base of any mountains that might be on the horizon, thus a horizon line at 24% of the image's height does not exactly mean that a quarter of the painting is sky. Of the paintings measured, the highest horizon line was at 15% while the lowest was at 40%. As stated, the average was 24% and the standard deviation was roughly 5%, so the majority were in the range of 20% to 30%. Since sky rendering was not a focus of this paper, we chose to place the default horizon line at 20%.

In addition to the curve added towards the horizon, Berann's paintings often feature a slight curve along the horizon (see Figures 1a and 2b). This curve is easier to notice in paintings of flat landscapes where the horizon is clearly defined. This curvature usually does not go beyond 5% of the image height unless the painting depicts a large portion of the globe. To create this curvature, we bend horizontal lines using cosine factor:

$$y' = y - C(1 - \cos(x\frac{\pi}{2})) \tag{1}$$

where $C$ is a constant to control the amount of curvature, and $x$ is the first coordinate in projected space (in the range of -1 to 1).

To match Berann's style we use $C = 0.05$. Applying these types of curvature to our Lake Louise model yields the rendering seen in Figure 4b.

Finally, we apply vertical exaggeration to the terrain. Other researchers have experimented with various exaggeration schemes. For example, Bratkova et al. use a pair of exaggeration constants, one for low elevation and one for high elevation [Bratkova et al. 2009]. The exaggeration applied to a given vertex is a linear blend between these two factors. The problem with this method is that the relative heights of mountains is not preserved, reducing the information contained in the final image. To address this, we propose a view dependent exaggeration that preserves relative sizes of features. Scaling is done in view space after all other transformations have been applied so that the new orientation of the terrain does not affect the perceived size of the mountain. The scale factor is based on the viewport size so mountains occupy a consistent amount of space within the image.

In our interactive system, we allow users to zoom in or out. Zooming is handled by changing the orthographic projection's field of view. To avoid having mountains appear to stretch taller as you zoom out, we choose a scaling factor that is inversely proportional to the perceived viewing radius:

$$y' = y + (elevation * scaleFactor/viewRadius) \qquad (2)$$

This makes the scaling factor similar to perspective scaling, except that we use the same factor for the entire mesh to maintain relative sizes of near and distant objects: in a true perspective projection, the scaling factor would be the depth of a given vertex. This is supported by Patterson who observed that Berann typically used more exaggeration in smaller scenes than in larger ones [Patterson 2000]. Our final result for distorted terrain can be seen in Figure 4c.

Berann occasionally exaggerated certain features selectively to either make the map easier to read or more aesthetically pleasing [Patterson 2000]. In particular, narrow features like valleys were sometimes widened to become more noticeable. More controversially, Berann relocated or reoriented major features in some paintings. An example of this is the mountain range on the horizon of the Yellowstone National Park painting in Figure 1a, which was rotated to show the east face even though the painting is looking southward. These sorts of distortions are not covered here as we aim to create a real-time rendering feature for digital earth systems where users may change the view and should be presented an accurate model of the terrain. Figure 1d shows an alternate rendering of Yellowstone that faces the Teton Range.

## 3.2 Terrain Colour

*3.2.1 Terrain Classification.* Now that the terrain is shaped like a panorama map, the next step is to add some colour. Berann's paintings are filled with a wide range of vibrant colours, but each part of the terrain is easily distinguished by its consistent palette. Forests are naturally green, while fields are more yellow, hills are orange, cliffs are more red, and snow caps are mostly white. The challenge is to find a way to apply these colours in our synthesized maps. Since our model is created from real world data, we start by identifying the types of terrain in our model.

For some features we rely on available datasets, for others we identify them by applying basic operations to existing data. Data sets are available for forest regions and water bodies, since these can be found by analyzing satellite images. Tree cover data used here has been made available through a study by Hansen et al. [2013]. A separate data set must also be used to identify areas covered by water. Water cover can also be derived from satellite images or standard maps. For the examples in this paper, we extracted water cover data from OpenCycleMap [ocm 2017].

After excluding areas covered by trees or water, the remaining terrain features not present in the data can be estimated from the elevation model. For example, cliffs can be estimated as areas where the elevation model changes abruptly. Therefore, by calculating the gradient of the heightmap we can flag steep slopes on the terrain. Similarly, we can identify fields and hills as areas with a low to moderate incline.

Snow covered mountain peaks can also be estimated using the elevation model. A simple approach is to mark any regions above a certain elevation as snow-covered. A more accurate method is to incorporate another data layer created from satellite photography. Marking areas that have both high elevation and appear white on satellite could yield more realistic snow caps when rendering mountains. A key challenge with this approach would be in selecting photos of an appropriate season, time of day, and ones that are not obscured by clouds. Analyzing satellite photos is outside of the scope of this thesis, but would be an interesting avenue to explore in the future.

For rendering purposes, each vertex of the terrain mesh is accompanied by flags to indicate the terrain type. To allow smooth transitions, the flags are a set of multiple values representing the various terrain types.

Once the terrain has been classified, our next task is to select colours for each terrain type. To do this we examine Berann's choice of colour. Looking closely at the parts of Berann's paintings in the shade, we see that a standard lighting model is not used. Mountains that are brown in the light become blue in the shade, and the green trees become violet. To understand why the colour is shifted this way, let's take brief look at colour harmonization [Cohen-Or et al. 2006]. The colour wheel, as commonly seen in photo and print applications, is a useful tool for design. It is created by dividing a circle into thirds and placing a primary colour on each of the spokes (either red, green, and blue for an additive colour system, or red, yellow, and blue for a subtractive one). Colours opposite on the wheel are known as complementary colours since they maximize contrast. On the other hand, colours that are next to each other on the wheel harmonize well together, and are called analogous colours. Based on these principles one can build a palette around a base colour to suit their goal. A common colour scheme is called "split-complimentary" which uses a base colour and the colours adjacent to its compliment. Using the adjacent colours reduces the contrast enough to make an image less jarring.

In Berann's painting, we can see that rather than applying a realistic lighting model to the shaded regions, he frequently uses complimentary or split-complimentary colours to create contrast between the light and dark areas. To mimic Berann's painting, our rendering then needs a colour palette for each terrain type for both lit and shaded regions.

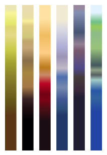


**Figure 6: Colour palettes for cel shading, from left to right: field, hill, cliff, snow, forest base, and trees.**

*3.2.2 Shading.* To create a 3D rendering with a hand drawn style, we use cel shading using a palette sampled from Berann's paintings. In some applications of cel shading the palette may have as few as two colours yielding sharp boundaries, or cels, for lit and unlit regions. To mimic the rich variety in colours present in Berann's paintings, we use continuous palettes as shown in Figure 6. This texture shows six palettes. From left to right the palettes are used for field, hill, cliff, snow, forest base, and trees. The colours in each palette were chosen by sampling colours from Berann's paintings. The top of the palette is the colour used for a fully lit pixel, the middle is for an unlit pixel. Since our scene is only being lit by a single directional light for the sun, the dot product between the light direction and the surface normal can yield a negative number. We use this "negative" lighting to add variation in the colours of shaded regions.

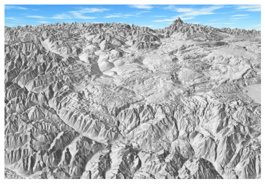


**Figure 7: Light map of Yellowstone National Park.**

The default colour palettes were sampled from the Berann's painting of Yellowstone National Park. In order to match sampled colours to particular lighting values in the palette, we obtained a data set for Yellowstone National Park and rendered a light map of the region. A light map is a texture used to cache lighting information for a surface [Akenine-Möller et al. 2008]. Figure 7 contains the light map of Yellowstone. Rather than using this texture for rendering, we use it to help build our colour palette. For each colour sample taken from the Yellowstone painting, a corresponding light level is sampled from the light map. These colour-value pairs form the basis of our palette. The brightness of each pixel in the light map is a value between 0 and 1 indicating the amount of light in the area. The palette is built by using the light value as the normalized texture coordinate for the palette texture. After a number of colour samples were taken from the painting and assigned to light values, the final palette texture is created using linear gradients to blend between each sample point.

Figure 8 shows the Lake Louise model rendered with cel shading. To increase colour variation on such a low resolution model, we

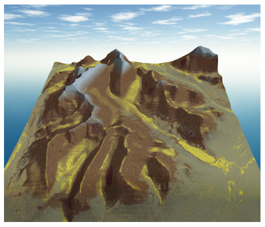


**Figure 8: Terrain with cel shading.**

use a normal map texture [Akenine-Möller et al. 2008]. Another colour variation can be seen near the bottom of Berann's paintings. Features at the bottom of each painting appear slightly darker than similar features near the painting's center. For a few examples of this, see the paintings in Figure 2. In each case the typically vibrant colours fade gradually to slightly darker tones at the very bottom of the painting. To mimic this effect we adjust the lighting value for pixels close to the bottom of the screen. In Berann's paintings the transition to darker tones at the bottom of the painting is gradual, so it is difficult to measure precisely where in the images it starts. However, in many cases the shaded area is noticeable within roughly the bottom 20 to 25% of the image. The amount the light is reduced is proportional to the pixel's distance from the bottom, creating a smooth transition from light to dark tones.

## 3.3 Trees

Examining Berann's paintings closely reveals that forested areas are not simply flat or blended colours: forests are composed of a base colour covered by many tiny brush strokes in different colours to represent individual trees. An example of this can be seen in Figure 9, which shows a close-up of a densely forested area in Berann's Yellowstone painting. These brush strokes add a level of texture to the painting that contrasts nicely with smooth fields. We can easily mimic rendering brush strokes using view-aligned particles, but the main challenge here is to identify where the trees should be placed.

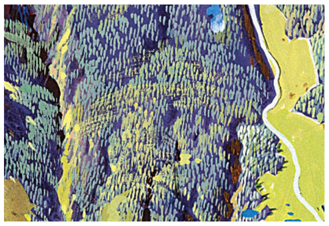


**Figure 9: A dense forest in Berann's Yellowstone painting.**

Despite the thick cover of trees if Figure 9, the underlying dark forest colour is still visible between individual trees. In essence, brush strokes are placed in such a way that minimizes overlap. Bratkova presents an algorithm for rendering brush strokes in screen space so as to avoid overlap [Bratkova et al. 2009]. However, this algorithm is targeted towards a ray-tracer, and is not applicable to an interactive application since it is not coherent under a changing viewpoint.

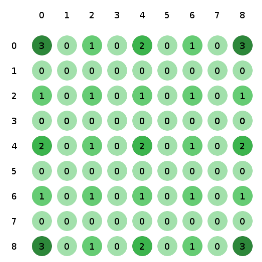


**Figure 10: Tree group numbers.**

To overcome this problem, trees must be placed prior to rendering so that they are in a consistent position when the view changes. In addition, as can be seen in Figure 9, Berann did not arrange trees in a realistic pattern. Many trees are painted in rows, and are spaced to avoiding overlapping brush strokes. In our system, we consider this fact when generating trees. When the tree cover data is loaded, we generate trees procedurally using a uniform distribution based on the normalized intensity value to determine whether a small region contains a tree. Trees are stored as points and expanded to view-aligned particles. To speed up render time this operation is done in a geometry shader. To give them the appearance of a brush stroke, textures are used to mask out a rounded area within the quadrilateral particle. To add variety to a forest, each particle's exact size, shape, and orientation is slightly randomized when the particle is generated. A random offset is also added to the particle's position to avoid perfectly straight rows of trees.

It is important to note that Berann's orthographic projection also extends to the trees. Brush strokes drawn for trees do not vary significantly in size based on where they are in the scene. To mimic this, all brush strokes in the scene are created roughly the same size regardless of location or zoom level. This, however, means that without any additional modifications, when zoomed in a forest would appear sparse and when zoomed out the trees would overlap too much to see the underlying base colour. To create the effect of evenly spaced brush strokes, we use a level-of-detail algorithm to add or remove trees from the scene as needed to create an even distribution of brush strokes. When zoomed in, all trees are visible, but as the view is zoomed out, groups of trees are removed proportional to the zoom level to maintain evenly spaced brush strokes. We maintain an even distribution of trees by adding or removing every second tree at each zoom level. To do this, trees are assigned to groups based on the index of the data point that generated them. As shown in Figure 10, each tree's group number is the highest power of two that evenly divides both its $x$ and $y$ coordinates. This ensures alternating trees are added or removed at each zoom level. To avoid popping, trees are scaled in or out. Scaling has the benefit of filling small gaps in a forest with small brush strokes, keeping the spacing even at any zoom level. Brush strokes of varying sizes can also be seen in Berann's paintings, making this method work well to mimic his style.

Figure 11 shows two renderings of the Lake Louise terrain with brush strokes for trees. The trees are drawn larger than normal here to make it easier to see how they are distributed on the surface. The brush strokes remain evenly spaced at two different zoom levels. Tree particles use the normal of the underlying terrain mesh for lighting calculations.

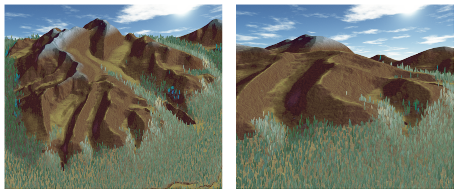


**Figure 11: Two renderings of trees at different zoom levels.**

## 3.4 Water

*3.4.1 Analysis.* Water is one of the most challenging elements to render in real-time, and a great amount of research has been done on the subject [Iglesias 2004]. However, our goal is not to render realistic water, but to replicate the artistic style of the water seen in Berann's work.

Berann's style for representing water varies a bit across his collection of paintings. In most of his paintings that depict a large portion of the earth, he replaced traditional water with a stylized relief drawing of the ocean floor. Furthermore, in a few paintings that contain small or unimportant bodies of water, the water is painted very simply, without much variation in colour.

In most cases, however, Berann's depiction of water is eye-catching and contains a number of common features. The first noticeable aspect in the paintings of Yellowstone and Mount Fuji is that the water is primarily composed of very bright colours. The water's brightness is used to balance out the darker shades contained in most terrain features such as mountains, forests, and shaded areas. As is the case with these examples, Berann's paintings of water almost always show reflections of the sky. If the painting includes mountains, the water often also shows reflections of the terrain just as an observer on the water's shore would see.

As popularly depicted in photography, the reflection of a mountain on a still lake can be as clear as a mirror. In Berann's paintings, however, reflections are never so sharp. As you can see in the painting of Mount Fuji in Figure 3a, reflections of both sky and terrain are distorted, drastically softened, and fairly muted by the water's natural colour.

In paintings where the terrain is more flat or there is a larger content of water, Berann uses other techniques to keep the scene dynamic and interesting. One example is Berann's painting of Alghero, the Italian city known as the home of Neptune's Grotto [Troyer 2016]. Nearly half the painting's area depicts water, and while some reflections are visible beneath coastal cliffs and the sky's sparse clouds, the majority of the painting is unaffected by these factors. Without reflections clouding our view, we can see that Berann also varies the colour of water based on the water's depth. Deep water near the bottom and sides of the painting is a much darker shade of blue, and the water near the coasts fades smoothly from pale blue to a white sandy beach.

*3.4.2 Implementation.* We now apply these observations to create an algorithm for rendering water. To do this, we need to identify what portions of the terrain are covered in water, as well as how deep the water is. Data sets specifically showing water cover are available, usually in vector format. Vector data can be aligned to a heightmap to generate a mesh.

Water cover data sets typically indicate boundaries or areas covered in water, but do not indicate how deep the water is. Additionally, digital elevation models generally show the elevation of the water's surface. For many bodies of water, bathymetry data sets are available but they need to be tracked down individually. If bathymetry data is available it can be used to correct the water covered portions of the terrain mesh. For areas where bathymetry data is not available, we make a simple estimate: terrain vertices that are underwater are lowered a distance proportional to the distance to the nearest shore.

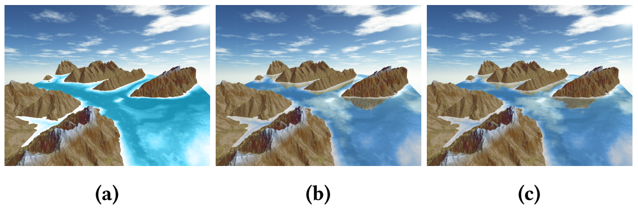


**Figure 12: Terrain model of Banff National Park covered with water for testing. (12a) Water colour varied by depth. (12b) Reflections and depth colouring. (12c) Reflections are blurred and distorted by animated ripples.**

As a sandbox to demonstrate our technique, Figure 12 shows some mountains from Banff National Park partially covered in deep water. In this image a base colour for the water was selected by sampling a colour from the middle of a body of water in Berann's painting.

The first step is to vary the base colouring by the water's depth such that it is dark in the deep areas and gradually fades to white in the shallow areas. Given the maximum depth of a body of water, $d_{max}$, we calculate a point's relative depth as $t = d/d_{max}$. Using this factor we can fade the water's colour to white by adjusting the colour's saturation and value. As the water becomes more shallow, we would like the saturation to decrease (reducing the colour content), and the value to increase (making the colour brighter). A simple solution is to adjust saturation and value linearly according to the relative depth.

This linear equation is the simplest way to fade the water colour by depth, but the results tend to look fairly washed out. Rather than having some areas that are deep blue and some that are off-white, the majority of the image just appears faded. To create more distinct areas as can be seen in the Alghero painting, we need to classify more areas as either deep or shallow. We can do this by modifying $t$ using a curve with a steeper ascent, t should remain in the range of 0 to 1. This can be accomplished using a sigmoid function.

The next step is to incorporate reflections of the scene. To render reflections, we make the simplifying assumption that our water is essentially flat. This matches the style Berann used in his paintings. Additionally, given that we view the terrain from a very high vantage point, this is a safe assumption.

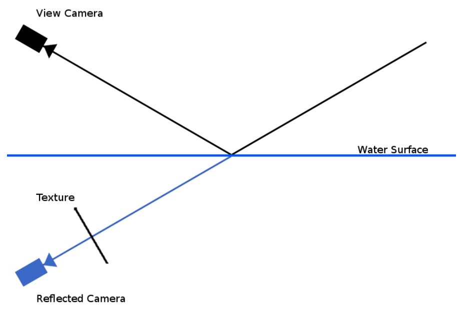


**Figure 13: Render the water's reflection to a texture.**

To render the water's reflection in real-time, we can render the scene as it is seen from a camera reflected by the water's surface to a texture, as demonstrated in Figure 13. We then project that texture onto the water. This idea is similar to environment mapping, except that some objects being reflected are close to the reflective surface, so the reflection texture needs to be generated dynamically [Akenine-Möller et al. 2008].

When rendering the reflection texture, we must use a clipping plane at the water's surface to avoid rendering reflections for any terrain that is under water. The reflection can be projected back onto the water's surface by using the water's view space coordinates to address the reflection texture.

Combining the depth colour with the reflection colour presents a challenge. A simple average does not produce satisfactory results. To match Berann's paintings, we must maintain some of the colour of the terrain reflections as well as the fade to white of the depth colour. In other words, we want to see the hue of the reflection and the saturation of the depth colour, therefore we mix colours in HSV colour space [Akenine-Möller et al. 2008].

We mix the colours in two steps. First, the colours are combined in HSV colour space taking the hue of the reflection, the minimum saturation, and the average brightness. The resulting colour is then mixed again with the depth colour in RGB colour space to fade the hue slightly toward the water's natural blue. This blend is done in RGB colour space to avoid introducing colours whose hue falls between blue and the reflected colour. The result can be seen in Figure 12b.

We now have a render of water that combines depth and reflection colours. However, as we observed above, reflections in Berann's paintings are never this crisp. To improve our rendering we need to soften and distort the reflections. Softening the reflections is as simple as applying a wide Gaussian blur to the reflection texture [Akenine-Möller et al. 2008].

In our system, water is rendered as a flat surface. This is consistent with Berann's art, and is also how a body of water would appear if viewed from such a high elevation. However, to capture small variations in the reflection, we use a normal map. The encoded surface normals are used to simulate a rough surface with per-pixel lighting. Normal maps can be created from a heightmap by calculating the gradient. Varying the normal will result in a displacement of the texture coordinates used to sample the reflection texture.

To add ripples to the water's reflection, we sample the du/dv map and use the result to offset the texture coordinates for the reflection texture. Since our system is interactive, we animate the water ripples. This is done by varying the texture coordinates used to sample the du/dv map over time. To avoid obvious repetition in the waves, we sample the du/dv map a second time using the modified texture coordinates. Figure 12c shows the results of applying the blur and distortion to the water.

## 3.5 Atmosphere

The final element we replicate in our rendering is the effect of the atmosphere on distant terrain. Berann's painting for Yellowstone in Figure 1a depicts a number of mountain ranges. Note that mountains near the base of the image are painted with a variety of colours, while those on the horizon all have a blue tint. Berann is portraying the effect of atmospheric (or aerial) perspective [Akenine-Möller et al. 2008]. Physically accurate models for atmospheric effects have been developed by other researchers [Pharr and Fernando 2005], but are quite resource intensive. Rather, we use a simple technique that runs in real-time and captures the non-photorealistic style of Berann's paintings.

A simple way to fake atmospheric scattering for real-time applications is to linearly interpolate between the terrain colour and an atmosphere colour sampled from Berann's painting based on the terrain's distance from the viewer. This can be easily done in our system using the curve and horizon distances ($D1$ and $D2$) shown in Figure 5 as a scale. To avoid washing out the whole scene, we use a non-linear blend that causes the blue hue to start to become noticeable closer to the horizon. Figure 14a shows the results of this approach.

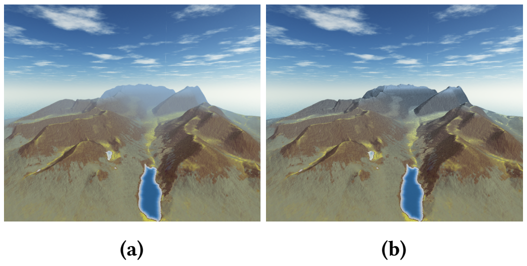


**Figure 14: Rendering atmosphere. (14a) A simple implementation. (14b) Atmosphere that fades the colour but preserves the details.**

While this is a decent approximation of the effect as seen in photos, the mountains in Figure 14a appear much less distinct than those in Figure 1a. Atmospheric perspective tends to lower the contrast of distant objects, however, in Berann's paintings distant mountains retain many visible features, just with a blue tint. Since these paintings serve the dual purpose of being maps, maintaining the clarity of even distant terrain increases the information they contain.

To replicate this effect, we need to change the terrain's base colour to blue, but keep the light and dark variations for details. We do this by mixing in the atmospheric colour in RGB colour space as described above, but then we replace the mixed colour's value with that of the original terrain colour. This effectively shifts the model's hue towards the atmospheric colour, but maintains contrast between bright and dark portions of the terrain. Figure 14b shows the results of this change. As can be seen in the comparison, mountains on the horizon are faded blue, but their details are still easy to distinguish.

## 4 RESULTS

We have developed a real-time system for rendering panorama maps using digital earth data by integrating all the methods described. Figures 1b and 1d show the final renderings of Yellowstone National Park. The skybox in the backdrop of the renderings presented in this paper are based on the work of Heiko Irrgang (distributed under the Creative Commons License) [Irrgang 2016]. Our system runs in real-time and can be explored interactively. Users can pan, zoom, and rotate the scene, as well as change various rendering parameters as described in previous sections. All results shown were rendered using unmodified digital earth data.

Figure 1 shows a side-by-side comparison of our rendering of Yellowstone with Berann's painting. The view and terrain shape match for most of the painting's major features. Terrain is distorted smoothly from a top-down view near the base of the painting to a sharp horizon near the top. Both images depict most of the same features: while the rendering was lined up as closely to the painting's view as possible, it was done using the system's interactive controls and therefore may not match perfectly in some cases. However, what is important to note is that both images contain roughly the same set of terrain features (mountains, rivers, forests, etc.) indicating that our modified orthographic projection is a good match for paintings of this scale.

One notable difference between the two images is in the terrain on the horizon. The horizon line (as measured from the base of any mountains on the horizon) is at roughly 75% of the painting's height. This factor is adjustable within our rendering system, since the horizon line does vary somewhat among Berann's paintings. The rendering's horizon level is matched to that of the painting's, however the mountains depicted on the horizon do not match. The mountains seen on the horizon of Berann's painting are the Teton Range. These mountains can be seen in the rendering as the peak in the top right corner. Berann's painting of Yellowstone looks southward over Yellowstone Lake, meaning that the horizon runs from east to west. However, the Teton Range actually runs from north to south. Berann rotated the mountain range to fill the horizon of his chosen view.

These sorts of major distortions are not common in Berann's work since they reduce the cartographic reliability of the painting. Being uncommon also makes them difficult to reproduce in an automated system since the choice made by the artist favoured aesthetic over accuracy. Our system does not attempt to implement interactive tools for drastic distortions such as this since we would like to use our system as a rendering feature for digital earth systems (such as WorldView or Google Earth). While painting the Teton mountain range on the horizon of the Yellowstone map created a captivating image, a user navigating our system may choose to view the park from a different angle, as seen in Figure 1d.

As discussed, the colour palettes were created by sampling from Berann's Yellowstone painting. Many areas are close matches between the two images: Most notably, the mountains, plains, and forests just below Yellowstone Lake near the centre of the image match closely in colour. Some slight variations in these areas can be attributed to more bumpy or detailed terrain data found in the digital elevation model. For example, a close-up of the field near the centre of the image can be seen in Figure 15.

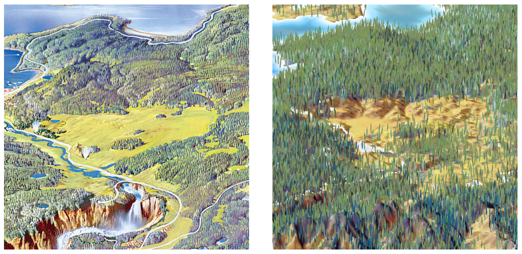


**Figure 15: Close-up comparison of a field.**

In Berann's painting, this area appears to be a relatively flat field. In our rendering however, it can be seen that several small hills are present. To make areas such as this match a little more closely, an additional data set might be necessary to tag the entire area as a grassy field. By analyzing satellite imagery using NDVI, these types of areas may be detected.

Similarly, an additional data set might be necessary to improve the appearance of snow caps on mountains. As can be see in the comparison, our estimate captures some, but not all, of the snowy areas. Incorporating an algorithm that analyzes satellite photography might help identify snowy regions. This may not necessarily make the results appear closer to Berann's paintings, but it should make the results more accurate to real world data. Naturally, the details of elements such as this would depend on the season being viewed. For an interactive system it could be beneficial to incorporate time-varying data to allow a user to select the season and tailor the panorama map to conditions they would encounter if they were to visit the region. digital earth systems such as Google Earth continue to update their data with new and better satellite imagery. Updates are currently periodic, though it is expected they will approach continuous live updating. Our panorama map system can benefit from such enhanced data integration, using it to render time dependent aspects such as snow, water, and vegetation coverage.

Many of the major forests in Berann's painting can also be seen in the rendering, and the two images appear to have matching colours in most places. One of the biggest challenges in rendering trees is picking a suitable size for the brush stroke. Berann's paintings were done on a canvas, and scans of his work are provided at very high resolutions. For an interactive renderer, typical screen resolutions are significantly lower. To mimic the effect of distinct brush strokes for trees, each tree must be drawn at a minimum size in order to be visible on a monitor.

The painting features Yellowstone Lake as its central focus. The base colour for our rendering was sampled from the painting, and pale shallow water is matched along the lake's borders. The biggest difference in the lake's appearance can be attributed to reflections of the sky. Berann's Yellowstone painting is framed by a detailed sky full of clouds. The sky is crafted for this particular view, while the sky in our rendering will largely depend on the user's selected vantage point. Though our sky's contents do not match the painting, the reflections in our rendering exhibit the same styling as those in the painting: the reflected clouds are distorted and blurred.

Finally, the atmospheric effects applied are a good match for those seen in Berann's paintings. The colours are faded, but you can still see many details in the mountains on the horizon. Though the method is simple, it is effective in mimicking the painting, and has virtually no impact on performance.

Overall, by our visual inspection, the rendering techniques presented here produce compelling results that replicate many of the most prominent features of the source material. A number of details have been noted as areas that could be explored further to find a method to match even closer to the original painting. Some variations are view or data dependent, and in an interactive system it is not possible to validate other views with a direct comparison. However, since this rendering recreates many aspects of the painting, it is our hope that a user can navigate to a different view or region and find new renderings that are captivating, and pay homage to Berann's work.

To show that our system works with additional data sets, Figure 16 shows a rendering of the Northern Cascades. This painting contains a large amount of snow, so our snow elevation had to be adjusted for this scene. As discussed above, incorporating additional data sets, such as snow cover, could increase the accuracy of renderings.

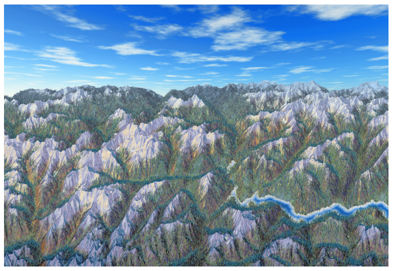


**Figure 16: Rendering of Northern Cascades.**

Our rendering system was implemented in C++, compiled using Visual C++ 2013 Update 5. Rendering is done in OpenGL, with shaders targeted at the GLSL 4.2 core profile. Much of the work is done in shaders, though CPU-side math uses the GLM library, version 0.9.7.1. The system was tested on an ASUS TP500LN series laptop that is running Windows 10. The 3D rendering widget is 1089x690. The Yellowstone scene covers roughly 10,000 km$^2$ of terrain. The data sets are 1621x1621, resulting in a terrain mesh containing 5,248,800 triangles. A separate water mesh is also created. For this scene, roughly 715,000 tree particles are typically generated. Sample code for our implementation is provided in the thesis [Brown 2017].

Under these test conditions, our panorama renderer runs at a steady 23 frames per second. The system is fairly average, and is a good indication that our panorama renderer can be expected to run at interactive frame rates on an average computer.

## 5 CONCLUSION

In this paper we have analysed the key elements of panorama maps and taken steps towards replicating their style in a real-time interactive environment. We have presented techniques for terrain distortion, distorted projection, terrain colouring, tree brush strokes, water rendering, and atmospheric effects. The techniques presented are based on measurements and analysis of Berann's panorama maps. We have developed a suite of rendering techniques which run in a real-time, interactive system. Rendering is automated using a data set that can be easily obtained online. Freely available digital earth data is an important resource, whose potential is yet to be fully realised. Our hope is that the techniques presented here will inspire further innovation to make digital earth more accessible for all users.

A number of challenges remain to be explored. In the results presented in this paper, a simple skybox was used. Another challenge is to incorporate more elements of texture into the rendering, particularly along mountains and cliffs. It would be interesting to try to adapt the method of Bratkova et al. [2009] for placing texturing elements along fall lines to work in a real-time system. Additionally, more data sources such as road networks or buildings could be incorporated to further improve the rendering.

## ACKNOWLEDGMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), PYXIS, and Alberta Innovates - Technology Futures (AITF). We would like to thank them for their support.

## REFERENCES

2016. Elements of Art. http://www.getty.edu/education/teachers/building_lessons/formal_analysis.html. (2016). http://www.getty.edu/education/teachers/building_lessons/formal_analysis.html (accessed September 12, 2016).

2017. OpenCycleMap. https://www.opencyclemap.org/. (2017). https://www.opencyclemap.org/ (accessed January 2, 2017).

Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. 2008. *Real-Time Rendering 3rd Edition.* A. K. Peters, Ltd., Natick, MA, USA. 1045 pages.

Margarita Bratkova, Peter Shirley, and William B. Thompson. 2009. Artistic Rendering of Mountainous Terrain. *ACM Trans. Graph.* 28, 4, Article 102 (Sept. 2009), 17 pages. https://doi.org/10.1145/1559755.1559759

J. Brosz and F. Samavati. 2010. Shape Defined Panoramas. In *Shape Modeling International Conference (SMI), 2010.* IEEE Computer Society, 57–67. https://doi.org/10.1109/SMI.2010.23

John Brosz, Faramarz F. Samavati, M. Sheelagh T. Carpendale, and Mario Costa Sousa. 2007. Single Camera Flexible Projection. In *Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering (NPAR '07).* ACM, New York, NY, USA, 33–42. https://doi.org/10.1145/1274871.1274876

Stephen Alex Brown. 2017. *Real-Time Panorama Maps.* Master's thesis. University of Calgary.

Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. 2006. Color Harmonization. *ACM Trans. Graph.* 25, 3 (July 2006), 624–630. https://doi.org/10.1145/1141911.1141933

Patrick Degener and Reinhard Klein. 2009. A Variational Approach for Automatic Generation of Panoramic Maps. *ACM Trans. Graph.* 28, 1, Article 2 (Feb. 2009), 14 pages. https://doi.org/10.1145/1477926.1477928

Jürgen Döllner and Maike Buchin. 2003. Real-Time Expressive Rendering of City Models. In *IV.*

Martin Falk, Tobias Schafhitzel, Daniel Weiskopf, and Thomas Ertl. 2007. Panorama Maps with Non-linear Ray Tracing. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia (GRAPHITE '07).* ACM, New York, NY, USA, 9–16. https://doi.org/10.1145/1321261.1321263

Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. 1998. A Non-photorealistic Lighting Model for Automatic Technical Illustration. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98).* ACM, New York, NY, USA, 447–452. https://doi.org/10.1145/280814.280950

Al Gore. 1998. The Digital Earth: Understanding our planet in the 21st Century. http://www.digitalearth-isde.org/userfiles/The_Digital_Earth_Understanding_our_planet_in_the_21st_Century.doc. (1998). http://www.digitalearth-isde.org/userfiles/The_Digital_Earth_Understanding_our_planet_in_the_21st_Century.doc (accessed December 12, 2016).

M. C. Hansen, P. V. Potapov, R. Moore, M. Hancher, S. A. Turubanova, A. Tyukavina, D. Thau, S. V. Stehman, S. J. Goetz, T. R. Loveland, A. Kommareddy, A. Egorov, L. Chini, C. O. Justice, and J. R. G. Townshend. 2013. High-Resolution Global Maps of 21st-Century Forest Cover Change. *Science* 342, 6160 (2013), 850–853. https://doi.org/10.1126/science.1244693

Aaron Hertzmann. 1998. Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98).* ACM, New York, NY, USA, 453–460. https://doi.org/10.1145/280814.280951

A. Iglesias. 2004. Computer Graphics for Water Modeling and Rendering: A Survey. *Future Gener. Comput. Syst.* 20, 8 (Nov. 2004), 1355–1374. https://doi.org/10.1016/j.future.2004.05.026

Heiko Irrgang. 2016. Skybox Set. https://93i.de/p/free-skybox-texture-set/. (2016). https://93i.de/p/free-skybox-texture-set/ (accessed September 25, 2016).

Tobias Isenberg, Bert Freudenberg, Nick Halper, Stefan Schlechtweg, and Thomas Strothotte. 2003. A Developer's Guide to Silhouette Algorithms for Polygonal Models. *IEEE Comput. Graph. Appl.* 23, 4 (July 2003), 28–37. https://doi.org/10.1109/MCG.2003.1210862

Helen Jenny, Bernhard Jenny, and Lorenz Hurni. 2010. Interactive Design of 3D Maps with Progressive Projection. *The Cartographic Journal* 47, 3 (2010), 211–221. https://doi.org/10.1179/000870410x12786821061495

Patrick Kennelly and A. Kimerling. 2006. Non-Photorealistic Rendering and Terrain Representation. *Cartographic Perspectives* 0, 54 (2006). http://www.cartographicperspectives.org/index.php/journal/article/view/cp54-kennelly-kimerling

Haik Lorenz, Matthias Trapp, Jürgen Döllner, and Markus Jobst. 2008. *Interactive Multi-Perspective Views of Virtual 3D Landscape and City Models.* Springer Berlin Heidelberg, Berlin, Heidelberg, 301–321. https://doi.org/10.1007/978-3-540-78946-8_16

Ali Mahdavi-Amiri, Troy Alderson, and Faramarz Samavati. 2015. A Survey of Digital Earth. *Computers & Graphics* 53, Part B (2015), 95 – 117. https://doi.org/10.1016/j.cag.2015.08.005

Sebastian Möser, Patrick Degener, Roland Wahl, and Reinhard Klein. 2008. Context Aware Terrain Visualization for Wayfinding and Navigation. *Comput. Graph. Forum* 27 (2008), 1853–1860.

Paipateroma. 2016. Mt. Ashitaka and Mt. Fuji, Japan. https://commons.wikimedia.org/wiki/File:Mt_fuji_and_mt_ashitaka.jpg. (2016). https://commons.wikimedia.org/wiki/File:Mt_fuji_and_mt_ashitaka.jpg (accessed September 8, 2016).

Sebastian Pasewaldt, Matthias Trapp, and Jürgen Döllner. 2011. Multiscale Visualization of 3D Geovirtual Environments Using View-Dependent Multi-Perspective Views. *Journal of WSCG* 19, 3 (2011), 111–118. http://wscg.zcu.cz/jwscg/J_WSCG_2011/!_2011_J_WSCG-3.pdf

Tom Patterson. 2000. A View From On High: Heinrich BerannâĂŹs Panoramas and Landscape Visualization Techniques for the U.S. National Park Service. *Cartographic Perspectives* 0, 36 (2000). http://www.cartographicperspectives.org/index.php/journal/article/view/cp36-patterson

Matt Pharr and Randima Fernando. 2005. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems).* Addison-Wesley Professional.

Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. 2001. Real-time Hatching. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01).* ACM, New York, NY, USA, 581–. https://doi.org/10.1145/383259.383328

Simon Premoze. 2002. Computer Generation of Panorama Maps. In *In Proceedings of the 3rd ICA Mountain Cartography Workshop.*

Amir Semmo, Matthias Trapp, Jan Eric Kyprianidis, and Jürgen Döllner. 2012. Interactive Visualization of Generalized Virtual 3D City Models using Level-of-Abstraction Transitions. *Comput. Graph. Forum* 31 (2012), 885–894.

Juuyoh Tanaka. 2016. Mt. Fuji. https://flic.kr/p/7Ku8LM. (2016). https://flic.kr/p/7Ku8LM (accessed September 8, 2016).

Matthias Trapp, Tassilo Glander, Henrik Buchholz, and Jürgen Döllner. 2008. 3D Generalization Lenses for Interactive Focus + Context Visualization of Virtual City Models. In *IV.*

Matthias Troyer. 2016. The life of H. C. Berann. http://www.berann.com/life.html. (2016). http://www.berann.com/life.html (accessed September 6, 2016).

Mahes Visvalingam and Kurt Dowson. 1998. Algorithms for sketching surfaces. *Computers & Graphics* 22, 2âĂŞ3 (1998), 269 – 280. https://doi.org/10.1016/S0097-8493(98)00037-5