Real-time non-photorealistic animation for immersive storytelling in “Age of Sail”

Cassidy Curtis\textsuperscript{a}, Kevin Dart\textsuperscript{b}, Theresa Latzko\textsuperscript{b}, John Kahrs\textsuperscript{c}

\textsuperscript{a}Google Spotlight Stories\textsuperscript{b}Chromosphere LA\textsuperscript{c}Boathouse Studios

ARTICLE INFO

Article history:
Received October 21, 2019

Keywords: non-photorealistic rendering, real-time, virtual reality, storytelling, animation

ABSTRACT

Immersive media such as virtual and augmented reality pose some interesting new challenges for non-photorealistic animation: we must not only balance the screen-space rules of a 2D visual style against 3D motion coherence, but also account for stereo spatialization and interactive camera movement, at a rate of 90 frames per second. We introduce two new real-time rendering techniques: MetaTexture, an example-based texturing method that adheres to the movement of 3D geometry while preserving the texture’s screen-space characteristics, and Edge Breakup, a method for roughening edges by warping with structured noise. We also describe a custom rendering pipeline featuring art-directable coloring, shadow filtering, and texture indication, and our approach to animating and rendering a painterly ocean in real time. We show how we have used these techniques to achieve the “moving illustration” style of the real-time immersive short film “Age of Sail”.

© 2019 Elsevier B.V. All rights reserved.

1. Introduction

The immersive short film Age of Sail [1] tells the story of an old sailor adrift in the north Atlantic in the year 1900. For this story to succeed, it had to reach the audience on two levels: engage them emotionally with the characters, but also immerse them in a world that is believable enough to create a real sense of peril. The story also had to run in real time within the limitations of current virtual reality hardware, including mobile devices.

We chose a visual style that we felt would support these goals: deliberately polygonal shapes with naturalistic proportions and movement, coupled with a limited color palette, flat-colored regions with roughened edges, rounded shadow shapes, and indications of texture (see Figures 1-2).

Adapting even a simple visual style from a static flat image to real-time animated 6DoF VR is an act of interpretation that requires careful thought. In this paper we will describe how we captured the essential qualities of Age of Sail’s visual style, including two new rendering techniques (MetaTexture and Edge Breakup) that solve problems unique to the 6DoF VR context.

1.1. Related Work

Balancing screen-space stylistic rules against 3D motion coherence is a well-known challenge in the field of NPAR. Various approaches have been made to produce coherent silhouettes [2, 3, 4, 5], using brushstrokes to fill regions [6, 7], volumetric textures [8] and screen-space advection [9, 10]. Our method differs from most of these in that it can be implemented within an existing real-time animation and rendering pipeline, using only a few custom shaders and image-processing filters. Also, none of these previous methods account for the needs of stereoscopic 6DoF VR. The artifacts they introduce may differ between left and right eye views, resulting in a misleading or incoherent perception of stereo depth.

There have been various papers describing non-photorealistic rendering in VR or AR: [11, 12, 13, 14, 15]. However, none of these address the complete set of issues including stereo fusion and temporal coherence. Northam et al [16] does address stereo coherence, but uses a technique inappropriate for real-time rendering.

Our multiresolution texture method can be considered a gen-
eralization of the UV-gradient-based curved hatching described by Saito and Takahashi [17]. Multiresolution or self-similar textures have been used to allow screen-space marks to adjust smoothly under changes in viewpoint, scale, or deformation: Bénard et al [18] and Kalnins et al [2] do it in linear edge space; Klein et al [11] rely on pre-rendered “art maps” that do not account for real-time lighting or animation changes. Dirksen [19] used a scalar multiresolution noise texture for pixel-scale dithering in 6DoF VR. Our method is more general, operating on full-color textures as well as vector fields, with special considerations for screen space orientation, texture skewing, and continuity across triangle edges.

Our edge breakup method is a direct extension of the warping technique used on the immersive short *Pearl* [20], with additional features specific to the 6DoF VR domain. Our overall approach to planning and executing non-photorealistic animation production follows the guidelines set out by Curtis [21].

This paper is an extended journal version of the Expressive 2019 paper by Curtis et al [22]. Compared to the original paper, the major new contributions in this version are: (1) analysis of the concept art that motivated the development of these techniques, (2) a detailed explanation of the custom rendering pipeline, ocean, and shaders, and (3) a method for compensating for contrast reduction in MetaTexture.

1.2. Overview

We will start by describing our overall artistic design approach (Section 2), and the new non-photorealistic techniques MetaTexture (Section 3), and Edge Breakup (Section 4). We will describe the customized animation and rendering pipeline (Section 5) that we built around these techniques. Following that, we will show examples of how we applied these techniques in production (Section 6), and suggest future directions for further exploration (Section 7).

2. Design

2.1. Inspiration

*Age of Sail*'s visual style (Figure 1) is inspired by the work of painters like Bernie Fuchs [23] and Thomas Hoyne [24]. Fuchs uses limited color palettes with roughened edges and hints of texture to produce evocative scenes; some of his paintings also have an “inner glow” that comes from his unique oil rub-out technique (see Figure 3). Hoyne’s compelling portrayals of life on the open ocean show thoughtful control of texture in lit and shadowed regions.

A key insight we drew from these illustrators is that each in his own way controls what the audience perceives by manipulating the salience of every element in the scene. They use techniques like *indication* [25] to convey the feeling that an object is textured without explicitly rendering its texture everywhere in full detail, or *lost and found edges* [26] which focuses
the viewer’s attention by carefully planning the locations of the most salient edges, and reducing edge salience elsewhere. What these techniques have in common is that they remove information from the scene, engaging and inviting the viewer to fill in the blanks.

2.2. Our design choices

We used a combination of these approaches in our designs. The hand-painted concept art in Figure 1 illustrates several of the elements of Age of Sail’s visual language. The scene consists mainly of solid-colored regions which, though simple and polygonal in shape, are drawn with rough edges that give them a handmade quality (Figure 2a). The color palette is limited: for example, there are only four shades of blue in the ocean (Figure 2b). Shadow shapes are rounded in a way that unifies the overall composition, and suggests they were painted with a brush of a certain size (Figure 2c). Texture is indicated sparingly, mainly on transitions between the lit and shadowed regions of smooth surfaces (Figure 2d).

The choice to build the image mainly of regions of solid color unifies the overall composition in an appealing way, and also makes the production process more flexible. Because overlapping objects with matching colors are perceived as one shape, the digital artists on our team are free to mix different techniques to produce a seamless end result (Figure 4).

2.3. The devil in the details

If our design style consisted only of simple, smooth-edged regions of color, adapting it to an immersive medium like 6DoF VR would be fairly straightforward. However, our style also has some highly salient fine-grained detail, of two distinct kinds: texture, and rough edges. These details pose an interesting technical challenge.

To function well in VR, every detail in the scene must (1) conform to the 2D screen-space rules of the style, (2) cohere with the movement of objects in 3D space, (3) adapt to the audience’s dynamically changing viewpoint, and (4) spatialize stereoscopically at an appropriate depth. This is a highly over-constrained problem.

Fortunately, our goal is not a mathematically perfect solution, but an illusion that achieves all of these goals in the viewer’s mind (a problem analogous to the “plausible physics” proposed by Barzel et al [27].) An error that is imperceptible to the audience is equivalent to a perfect result for our purposes. Our task, then, is to develop methods whose mistakes can be pushed below the threshold of perceivability. We will describe two such methods in the sections that follow: MetaTexture (Section 3) and Edge Breakup (Section 4).

3. MetaTexture

To achieve the illusion of a 2D visual style in 6DoF VR, we need a texturing technique that preserves the style’s screen-space characteristics while adhering to the movement of surfaces in 3D space. The technique must also be simple enough to perform well at high frame rates, a requirement that excludes most procedural or solid texturing methods. For this reason, we have opted for an example-based approach.

A MetaTexture is a multi-resolution texture made by blending multiple copies of a single tileable example texture (Figure 5a),
under different affine transformations in UV space. The transforms are chosen so that at any point on the surface, the transformed textures approximate as closely as possible the example texture’s shape and size in screen space (Figure 5c). Although the range of possible transforms is infinite, only a finite number of transforms are needed for any given fragment, so the texture can still be calculated in constant time.

3.1. Texture scales and blend coefficients

The first component of the affine transformation is the scale of the texture in UV space, which may be non-uniform. One way to determine texture scales is by using the magnitudes of the screen-space gradients of the texture coordinates $\mathbf{T} = (u, v)$. Assuming a square example texture of width $w$ pixels, the ideal texture scale, i.e. one which would produce locally the same screen-space proportions of the original texture, would be:

$$
\mathbf{S} = \left( \frac{1}{w\|\nabla u\|}, \frac{1}{w\|\nabla v\|} \right)
$$

However we cannot simply scale the texture by this vector locally at every pixel, as these gradients may vary continuously across the screen, producing wildly distorted textures. We must find a discrete set of scale values that can be relied upon not to change, and then smoothly interpolate between the resulting textures. For this, we have chosen the powers of two (similar to anisotropic mipmaps or ripmaps). We convert the scale to an exponent vector $\mathbf{E}$ as follows:

$$
\mathbf{E} = \left( \log_2(\mathbf{S}_u), \log_2(\mathbf{S}_v) \right)
$$

We threshold those exponents to produce the four nearest power-of-two scaled texture coordinates $\mathbf{U}_{0-3}$, and blend coefficients $\mathbf{B}$:

$$
\mathbf{U}_0 = (2^{E_u}u, 2^{E_v}v)
$$

$$
\mathbf{U}_1 = (2u_0, v_0)
$$

$$
\mathbf{U}_2 = (u_0, 2v_0)
$$

$$
\mathbf{U}_3 = (2u_0, 2v_0)
$$

$$
\mathbf{B} = (\beta(E_u - [E_u]), \beta(E_v - [E_v]))
$$

where $\beta(x)$ is any smooth monotonic blending function meeting the criteria that $\beta(0) = 0$ and $\beta(1) = 1$. (In practice, we have found the cubic blend $\beta(x) = -2x^3 + 3x^2$ produces pleasing results.)

The MetaTexture color $M$ is then determined by sampling the texture $T$ four times, and blending the results:

$$
M(\mathbf{U}) = (1 - B_s)[(1 - B_s)T(\mathbf{U}_0) + B_sT(\mathbf{U}_2)] + 
B_s[(1 - B_s)T(\mathbf{U}_1) + B_sT(\mathbf{U}_3)]
$$

(8)

Because the blend function is smooth, it is not immediately obvious to the viewer where one texture scale region ends and the next one begins (see Figure 5d). The regions overlap, and the transitions between them are less salient than the details of the texture itself, and thus they fall below the threshold of perceivability. This is also true of how the texture changes when an animated surface moves through 3D space: the transitions over time are quite subtle compared to the gross movement of the object as a whole. Thus we preserve the twin illusions of consistent 2D texture quality and 3D motion coherence while minimizing distracting artifacts.

3.2. Approximate smooth UV gradients

UV gradients have two disadvantages: first, they may not be continuous at triangle boundaries, which can lead to visible hard-edged artifacts when the discontinuities are large (see Figure 6a.) Second, gradients are based on derivatives, which must be calculated in the fragment shader, which is less efficient than the vertex shader when triangles are large.

We can approximate the UV gradients by deriving them from the local tangent and binormal, which can be calculated in the vertex shader and interpolated smoothly across the triangles. Given the screen-space projections of the unit tangent and bi-normal ($\mathbf{T}_s$ and $\mathbf{B}_s$), and the world-space UV gradients $\nabla_u, \nabla_v$, we approximate the screen-space gradients as follows:

$$
\nabla_s,u = [\nabla_u,\mathbf{T}_s]/|\mathbf{T}_s|^2
$$

(9)

$$
\nabla_s,v = [\nabla_v,\mathbf{B}_s]/|\mathbf{B}_s|^2
$$

(10)
By substituting $\nabla u$ and $\nabla v$ into Equation 1, we obtain smoother results, as seen in Figure 6b.

Note: this method assumes the world-space UV gradients are known for any vertex in the mesh. Calculating those gradients for every vertex every frame on deforming meshes may be computationally expensive on certain hardware configurations. In practice we have only used this method on meshes where the gradients can be approximated by a constant value across the entire mesh. (This method also assumes that the gradients of $u$ and $v$ are perpendicular in world space, which may not be true on all meshes. This is only an issue if compensating for skew as described in Section 3.4.)

Under certain circumstances, the method described above might produce a stereoscopic “shimmer” artifact, where the left and right eye views have completely unrelated textures when a surface subtends significantly different areas in the two different views. This can be corrected by projecting $\tilde{T}$ and $\tilde{B}$ using a single camera rather than projecting each eye separately. (In practice we have not found this necessary, since this only occurs when a large vertical surface is oblique to camera, and our most prominent use of MetaTexture is on the horizontal surface of the ocean: see Section 6.1.)

3.3. Compensating for radial angle

The above calculations assume that the desired result is a uniform texture scale in screen space. However, in a VR device, pixels near the edge of the screen subtend a significantly smaller radial angle than pixels near the center, which means that the apparent level of detail varies across the screen. Thus, the same surface will have a different appearance when viewed head-on versus obliquely, a difference that is quite noticeable if the viewer turns her head. We compensate for this by adding a radial angle compensation coefficient $\alpha$ to equation 1:

$$\tilde{s} = \left( \frac{1}{\alpha w|\nabla u|}, \frac{1}{\alpha w|\nabla v|} \right)$$

As a pixel’s radial angle depends on its location and the camera’s focal length, we can derive $\alpha$ from the screen space position $s$ and the camera projection matrix $P$ as follows:

$$\alpha = \frac{1}{|\tilde{Q}|^2 + 1}$$

3.4. Compensating for skew

So far we have only discussed scaling textures orthogonally along the $u$ and $v$ axes. However, it is still possible for textures to become highly stretched or distorted if the angle between $\nabla u$ and $\nabla v$ diverges too far from 90 degrees (Figures 7a and 5d).

To compensate for this distortion, we can rotate both eigenvectors in UV space by an angle $\psi$ so that their screen-space projections become perpendicular (Figure 7b). To find $\psi$, we start by computing a skew factor $\sigma$ as follows:

$$\sigma = \tan^{-1}(\epsilon) \left( \frac{\nabla u}{|\nabla u|}, \frac{\nabla v}{|\nabla v|} \right)$$

where

$$\epsilon = \begin{cases} |\nabla u| / |\nabla v|, & \text{if } |\nabla u| \leq |\nabla v| \\ |\nabla v| / |\nabla u|, & \text{otherwise} \end{cases}$$

We quantize the skew factor at a user-determined finite number of levels $n$ (see Figure 5e; in practice, values between 2 and 4 seem to work well) and use those to calculate quantized angles $\psi_0$, $\psi_1$, and finally determine new UV coordinates $\tilde{U}$, $\tilde{U}''$:

$$\sigma_0 = \frac{\lfloor n \sigma \rfloor}{n}$$

$$\sigma_1 = \frac{\lfloor n \sigma \rfloor + 1}{n}$$

$$\psi_0 = \tan^{-1} \left( \frac{\sigma_0}{2} \right)$$

$$\psi_1 = \tan^{-1} \left( \frac{\sigma_1}{2} \right)$$

$$\tilde{U} = \langle u \cos(\psi_0) + v \sin(\psi_0), v \cos(\psi_0) + u \sin(\psi_0) \rangle$$

$$\tilde{U}'' = \langle u \cos(\psi_1) + v \sin(\psi_1), v \cos(\psi_1) + u \sin(\psi_1) \rangle$$

Quantizing and blending based on the skew factor means we must sample the texture eight times rather than four. We generate the eight sets of coordinates $\tilde{U}_{0,3}$ and $\tilde{U}''_{0,3}$ using equations 2-5, and we calculate a third blend factor $B_{uv}$, and MetaTexture as follows:
\[ B_w = \beta (n\sigma - [n\sigma]) \]  

(22)

\[
M(\overline{U}) = (1 - B_w)((1 - B_r)((1 - B_u)T(\overline{U}_0) + B_u T(\overline{U}_2)) + B_r((1 - B_u)T(\overline{U}_1) + B_u T(\overline{U}_3)) + B_u((1 - B_u)T(\overline{U}_4) + B_u T(\overline{U}_6)) + B_r((1 - B_u)T(\overline{U}_5) + B_u T(\overline{U}_7)))
\]

(23)

There is a tradeoff here: the reduced distortion comes at the cost of increasing the number of overlapping regions (Figure 5f). In practice, whether skew compensation is needed depends on the texture and the desired result.

3.5. Orienting texture to indicate contour

In some cases a texture may have a clear sense of directionality to it, which you may wish to use to indicate contour along the surface. In this case we can reorient the texture depending on the relative magnitudes of \( \nabla u \) and \( \nabla v \). (See Figure 8.)

![Fig. 8: An ocean surface with a clearly directional MetaTexture (a) in its default orientation, (b) re-oriented to indicate contour.](image)

3.6. Compensating for contrast reduction

The above techniques alone do not guarantee a consistent level of contrast across the entire image. Some regions may be dominated by a single texture (Figure 9a) while others may be a blend of up to eight different textures (Figure 9b), which significantly reduces feature contrast and salience. (A similar problem often occurs in “fractalized” textures [28].)

This can be improved by using a histogram-preserving blending operator as described by Heitz and Neyret [29]. If you start with a texture that already has an approximately Gaussian histogram (Figure 9d), which turns out to be quite common in noise-based textures, you can skip the gaussification and degaussification steps entirely, and simply multiply the contrast by a single compensation factor \( c \), which can be reduced to the simple function:

\[ c = k(B_u)k(B_r)k(B_w) \]

(24)

where

\[ k(t) = 2t^2 - 2t + 1 \]

(25)

The adjusted result will more closely approximate the apparent contrast level of the original texture (Figure 9c). Note: while this feature was not needed in the case of Age of Sail, we have since found it useful on other projects where the reduced contrast is more noticeable.

![Fig. 9: (a/d) Original texture and its histogram. (b/c) Blending multiple textures reduces contrast. (c/f) A histogram-preserving blend restores contrast.](image)

4. Edge Breakup

The rough edges in the concept art (Figure 2a) were made using standard digital painting techniques, which typically involve 2D height fields or halftone patterns. To reproduce the look of those edges in stereoscopic 3D, we need to create the illusion that the roughness is “attached” to 3D objects in the scene. To achieve this, we use the strategy of warping the image with a structured vector field.

![Fig. 10: (a) A self-similar, tileable noise texture. (b) Warp pass, with noise rendered with MetaTexture on inflated surface. (c) Color pass, with clean-edged rendered surface. (d) Edges roughened by warping.](image)
to warp the clean-edged color pass (Figure 10c) to produce the final result (Figure 10d). Because the vector field is coincident with the geometry in 3D space, it will always move coherently, and its salient features will also spatialize at the correct depth when viewed in binocular stereo. This way we avoid the undesirable "rippled glass" effect common with image warping effects.

With an ordinary texture, the perceptual quality of the warped edge may vary depending on the viewer’s point of view. For example, when viewed up close, the texture may get blurry, resulting in lines that are soft and wiggly as opposed to rough. (This is in fact what happens in the 6DoF version of the immersive short Pearl [20].) So we use MetaTexture here as well, to guarantee that the roughness remains perceptually consistent regardless of the viewer’s point of view.

4.1. Edge inflation

To ensure that the edges can be warped in both directions, both towards and away from the object’s silhouette, our shader inflates the geometry by a small amount (measured in screen space, as pixels or as percentage of image size) by moving each vertex outward along its screen-space projected normal. (See Figure 11.)

4.2. Animated line boil

Line boil is how animators describe the subtle differences between drawn lines across successive frames of hand-drawn animation. We can mimic this effect by adding a periodically changing offset to the texture coordinates. (See Figure 12.)

4.3. Compensating for camera roll

Because the red and green channels encode warp vectors in screen space, if we were to roll the camera (e.g. if the viewer tilts her head sideways), the warped edges would change their shape (see Figure 13.) To compensate for this, we rotate the vectors in the shader, so that the frame of reference remains aligned with the screen space U gradient regardless of orientation.

4.4. Compensating for distance

The above techniques will ensure a uniform degree of screenspace roughness, creating the illusion that the entire image was painted using the same tools and materials. However, in order to preserve certain important details, it is sometimes desirable to reduce the roughness as an object recedes in the distance. In that case we can attenuate the intensity of the warp effect $\rho$ based on the distance from camera $d$:

$$\rho' = \frac{\rho}{1 + d^2}$$

Fig. 13: (a) Warping an edge using vectors encoded in the red/green channels. (b) Warping with the same vectors after camera rotation produces a different shape. (c) Rotating the vectors before encoding keeps the shape consistent.

Fig. 14: The effect of edge breakup (a) without and (b) with distance compensation. Note that without distance compensation, the edge breakup makes the character’s silhouette more difficult to read.
5. The Rendering Pipeline

To render the full scene with our visual style, we do not need all of the features of a traditional photorealistic renderer, but we do need some extra rendering passes and image processing filters not typically present in standard pipelines. So we developed a custom rendering pipeline comprising the minimal set of features we needed, optimized for real-time performance (see Figure 15).

Our pipeline consists of a series of 3D rendering passes interleaved with image-processing operations, as follows:

1. **Warp Pass** (render): In this pass we render all objects in the scene with the special shaders described in Section 4 (Figure 16a).

2. **Depth Pass** (render): Here we render only shadow-casting objects into a depth buffer, to be used later for casting shadows. The depth buffer’s spatial coordinates are warped with a sigmoid function so as to provide more detail in the areas near the viewer (Figure 16b).

3. **Shadow Pass** (render): In this pass (Figure 16c) we render different information into each of the three channels (Figures 16d-16f). In the red channel, we render a simplified Lambert-shaded version of the objects, with cast shadows (Figure 16d). The green channel denotes objects that need to have screen-space lens effects, such as light sources or specular highlights (Figure 16e). We also apply an optional *breakup map* texture to the blue channel on certain surfaces (Figure 16f). This will be used later to control the “palette knife” effect for texture indication.

4. **Blur Pass** (image processing): Here we blur the Shadow Pass using a series of alternating 1D Gaussian blur and downsampling filters (Figure 16g).

5. **Final Shadow Pass** (image processing): In this pass we combine information from the Shadow Pass and Blur Pass to produce rounded shadow shapes, inner glow, and texture indication (Figure 16h). For details of how this is done, see Section 5.1.

6. **Color Pass** (render): In this pass we render the colors of all objects in the scene (Figure 16i), using the Final Shadow Pass as an input texture. For details, see Section 5.2.

7. **Final Color Pass** (image processing): In this final step of
the pipeline, we use the Warp Pass to warp the pixels of the Color Pass. (Here we also add bloom effects based on the Final Shadow Pass green channel.) (Figure 16j).

5.1. Shadow shapes, inner glow, and indication
To produce hard-edged silhouettes with rounded corners, we threshold the Blur Pass red channel. We create the Fuchs-inspired “inner glow” effect by inverting the Blur Pass and clamping it to add a bit of light to the interior of the dark regions. For texture indication, wherever the Shadow Pass blue channel is non-zero, we increase the number of steps in the thresholding operation. This has the effect of breaking up the transitions from light to shadow in regions of the surface that are oblique to the light direction (Figure 16h).

5.2. Color control
Each character and prop in the scene contains two full sets of texture maps, one for shadowed areas (Figure 17a), and one for illuminated areas (Figure 17b). These textures are hand-painted, and art-directed so that certain salient details may stand out only in shadows, and others only when lit. We use the Final Shadow Pass red channel to blend between these lit and shadowed texture maps on all objects.

Fig. 17: Texture examples: (a) shadow textures, (b) lit textures, (c) final result.

The shaders used in this pass also have color overlay and saturation controls, which can be used to adjust the coloring of the lit and shadowed regions independently, allowing us to achieve a variety of different lighting conditions and moods using only a few user controls. These controls are also animatable, allowing us to adjust lighting conditions smoothly over time.

6. Applications
Here are some examples of these techniques applied to the task of rendering characters and visual effects in Age of Sail.

6.1. Painterly Ocean
The ocean plays a critical role in the story of Age of Sail. The viewer’s sense of peril hangs directly on the believability of this effect. The ocean also goes through significant transformations in texture, color and movement as the weather conditions change. To reproduce this complex natural phenomenon in a real-time experience, we rely on the strategy of removing unnecessary visual information.

In order to run on the full range of devices, the ocean’s data footprint must be very small, and it cannot require excessive amounts of real-time computation. This excludes from consideration any approaches involving real-time fluid simulation, volumetric data, high resolution geometry, or overly long animation clips. We have focused on two key elements to present a convincing illusion: (1) realistic movement and wave silhouette shapes, and (2) a painterly appearance that suggests the infinite detail of the ocean using limited visual complexity, much the way Hoyne’s paintings use brushstrokes of a limited thickness and palette.

To capture the characteristic feeling of the shapes and movement of ocean waves, we use a Tessendorf deep ocean wave model [30]. This model has the advantage that it consists of waves that repeat periodically in both time and space. Thus we are able to apply this deformation to a very low-resolution mesh, an 80-by-80 square grid about 60 meters wide (approximately 75cm per quad in physical space), in a clip lasting 20 seconds, whose first and last frames match seamlessly. Because of the spatial repetition, one could tile an arbitrarily wide swath of ocean with instances of this single tile. However, since a typical human viewer cannot perceive distinctions in stereoscopic depth beyond about 50 meters, a 5-by-5 grid of these tiles turns out to be sufficient. To simulate travel across the ocean, we animate the tiles treadmill-style, translating them past the origin (where the boat remains, for the convenience of the animators) and allowing them to disappear behind the boat and reappear in front. We conceal this treadmilling action by using a vertex shader to push down vertices beyond a 120-meter radius “false horizon” (Figure 18), and use simpler geometry for the ring of ocean between that and the real horizon (about 1km away).

These ocean tiles are textured with a custom fragment shader to add detail only where it is needed (Figure 19e). The shader uses a hand-painted texture (Figure 19a) with MetaTexture to create an underlying dithering pattern that stays consistent in screen space regardless of viewpoint. This is then used as an in-
Fig. 19: The various textures that contribute to the appearance of the ocean shader.

(a) Dithering 
(b) Bathic map 
(c) Foam erosion 
(d) Sky reflection 
(e) The final rendered ocean.

Fig. 20: Two other moments from *Age of Sail*, with their respective sky and bathic maps.
dexterity into a second texture, the bathic map, which represents the full palette of colors of the water at different depths and levels of illumination (Figure 19b). Sea foam, based on a third texture (Figure 19c) is applied to the up-wind sides of the waves. A fourth texture acts as a sky reflection map (Figure 19d), which is also distorted using the dithering pattern. By changing the bathic and sky maps, we can create a variety of different moods (see Figure 20). In certain scenes we also apply animatable overlay color and saturation controls to each of these maps, to capture the gradual shift of colors due to changing weather conditions on the open sea.

The movement of the ocean geometry as it passes by the boat, combined with MetaTexture, creates a fairly dynamic effect when the viewer looks out towards the horizon. However, when looking directly down at a patch of ocean surface, sometimes the static nature of the dithering texture can become noticeable, breaking the subjective illusion of wateriness. To compensate for this, we calculate a “live texture” $T'$ based on the original texture $T$, blending smoothly and cyclically through a mix of the red, green and blue channels at frequency $f$ cycles per second (we have found frequencies between 0.5 and 1.0 to give pleasingly ocean-like results.)

$$T' = T_0 + (T \cdot T_1) \cos(ft) + (T \cdot T_2) \sin(ft)$$

where

$$T_0 = \langle 0.5, 0.5, 0.5 \rangle$$

$$T_1 = \langle 0.707, 0.0, -0.707 \rangle$$

$$T_2 = \langle -0.408, 0.816, -0.408 \rangle$$

We can also offset the time value using world-space coordinates for an even more dynamic effect.

6.2. Wakes, ripples and splashes

Wherever a boat or character interacts with the ocean, more detailed movement is needed to produce a convincing effect. To achieve this, we decorate the ocean with smaller pieces of higher-resolution geometry for wakes, ripples and splashes (See Figure 21). To avoid geometry intersections and visually conflicting texture movement, we push the base mesh downwards, and suppress its texture so that it has effectively zero saliency (note that this is another case of removing information from the scene.)

Interestingly, if a partly transparent decoration is placed at the same height as the original ocean mesh, the holes in the decoration are not perceived as holes: because there are no salient details in the base mesh’s texture, its color appears to become part of the decoration’s texture. The result, even in stereoscopic 6DoF VR, is the illusion of a perfectly seamless ocean.

6.3. Animated Characters

Another area where we have deliberately removed information is in the character animation. The characters in Age of Sail were animated at 24fps, on “twos and threes” i.e. with poses held static for 2-3 frames (83-125ms) rather than smoothly interpolated between keys. The characters also have animated line boil to keep them alive even when they are not moving. In combination, these two effects support the feeling that the animation is hand-crafted as opposed to synthetic. We also reduced the intensity of the edge breakup on facial features to keep their expressions clear (Figure 22), and on entire characters when they are far from camera (Figure 14b).

7. Discussion

New developments in entertainment media technology often lead to a shift in audience response. For example, the release of the first high frame rate (48fps) stereoscopic film The Hobbit had a polarizing effect on the audience: while most viewers enjoyed the added realism, for some the high frame rate made the artifice of prosthetics, sets and synthetic characters too obvious when juxtaposed against the real human actors, a dissonance that prevented them from suspending their disbelief [31].

We suspect that virtual reality may have its own version of the Uncanny Valley [32]: as a simulation’s information density
increases, so does our ability to detect fakery within it. Since
VR devices, with their high frame rates and interactive respon-
siveness, have an inherently high information density, staying
on the right side of this “Unbelievable Valley” would require a
level of realism unachievable with current hardware. We chose
instead to run to the left. By simplifying the visual style and
choosing a deliberately staccato frame rate, we seem to have
reduced the information density enough to compensate for the
device’s excesses.

7.1. Taking the Art Seriously
We feel we were largely successful in capturing the concept
art’s important qualities (Figures 1-3) in the final rendered re-
results (Figure 4). This is not so much because of a particular
graphics technique, but because of the structure of our collabo-
rati on. The key is taking the concept art seriously: if production
designers create paintings that truly represent how they envision
the end product, and graphics developers seek to understand the
collaboration’s details deeply enough to adapt them to a new
medium, and each group asks thoughtful questions about the
other’s creative process, and answers them openly, this cycle of
feedback inevitably leads to good results. We hope this work
serves as a positive example of how artists and engineers can
collaborate effectively in any medium.

7.2. Performance
Age of Sail runs in real time on devices ranging from tethered
VR headsets to mobile phones. It renders an average of 800,000
triangles grouped into approximately 250 meshes, at a consis-
tent frame rate of 90fps on a Windows 10 PC with an NVidia
GeForce GTX1080 GPU. On mobile hardware such as a Pixel
3 or iPhone 8, it plays consistently at 57-60fps in both mono-
and stereoscopic modes.

7.3. Limitations
These techniques are certainly applicable to a wide variety of
other visual styles beyond that of Age of Sail. However, there
are certain limitations that should be considered.
The edge breakup can produce a noticeable “heat ripple” ef-
f ect, where a foreground object’s warp field distorts the back-
ground in the area immediately around it, if the background has
a lot of salient, high-contrast detail (Figure 23). In such cir-
cumstances it may be preferable to render and warp the scene
in multiple layers, although that would come at a higher com-
putational cost.

7.4. Future Work
This project raises some questions that warrant further study.
For example, is there an ideal frame rate for animated characters
in an immersive medium? Our choice to animate the characters
“on twos” (12fps) was deliberate, and has been well received
by most viewers, but there is a segment of our audience who
find that frame rate distracting. It would be interesting to ex-
pl ore this further via a control experiment or survey similar to
Michelle et al [31].
We also note that our choice to use powers of two as our dis-
crete set of scale values for our implementation of MetaTexture,
although entirely natural for those familiar with graphics tech-
niques like mipmaps, is somewhat arbitrary. It may be possible
to get different and interesting results by quantizing based on
powers of some other number, such as 3, $\sqrt{2}$ or the golden ratio
$\phi$. Also, texture orientation need not remain the same from one
scale level to the next. Our skew compensation method works
well enough in practice, but is admittedly ad hoc, and there
may be more robust approaches that we have not yet considered.
One can think of our particular implementation of MetaTexture
as a subset of a much larger space of possible example-based
texturing techniques that differ in their details while retaining
the same core principle of balancing screen-space texture scale
against stereoscopic 3D coherence. We would like to explore
that space further.

Our early experiments using these techniques in augmented
reality (see Figure 24) also raise interesting questions: how
should stylized virtual characters integrate into a real-world
background, and how can that background be manipulated to
feel consistent with the visual style?

8. Acknowledgements
The authors would like to thank: Dan Kaufman for his sup-
port of our efforts, Rachid El Guerrab for asking an interest-
ing question four years ago, Jan Pinkava, Karen Dufilho,
and David Eisenmann for their unwavering leadership, Céline
Desrumaux and Jasmin Lai for their inspiring production de-
sign, Josiah Larson for sweating the details, Tim J. Smith for

Fig. 23: “Heat ripple” effect (exaggerated here for clarity). Note that the black
and white checkered background is distorted near the edges of the red torus.

Fig. 24: A scene from augmented reality experiment The End.
psychological insights, Jon Klassen for designing and directing *The End*, David Apatoff and the Fuchs family for sharing Bernie Fuchs’ work, and everyone at Google ATAP, Chromosphere LA, Boathouse Studios and Evil Eye Pictures for their contributions to *Age of Sail*.

References


