The 8th ACM/EG Expressive Symposium
C. Kaplan, A. Forbes, and S. DiVerdi (Editors)

EXPRESSIVE 2019

Non-Photorealistic Animation for Immersive Storytelling

Cassidy J. Curtis[1], Kevin Dart[2], Theresa Latzko[2] and John Kahrs[3]

[1]Google Spotlight Stories
[2]Chromosphere LA
[3]Boathouse Studios

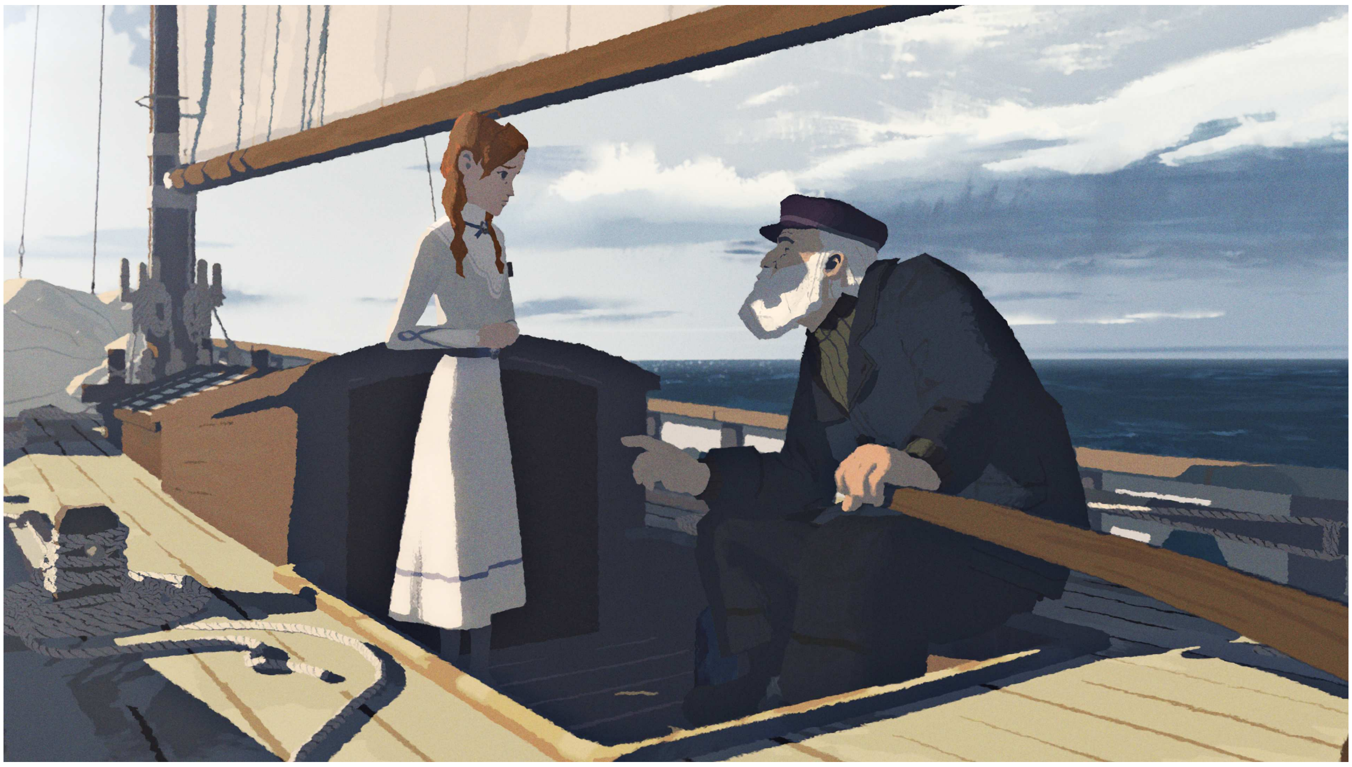


Figure 1: *A still frame from* Age of Sail.

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 multiresolution texturing method that adheres to the movement of 3D geometry while maintaining a consistent level of screen-space detail, and Edge Breakup, a method for roughening edges by warping with structured noise. We show how we have used these techniques, along with art-directable coloring, shadow filtering, and shader-based texture indication, to achieve the "moving illustration" style of the immersive short film "Age of Sail".

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Non-photorealistic rendering; Virtual reality;***

DOI: 10.2312/exp.20191071

1. Introduction

The immersive short film *Age of Sail* 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's 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 low-poly geometry with naturalistic proportions and movement, coupled with a limited palette of largely flat-colored regions, with roughened edges, rounded shadow shapes, and indications of texture.

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'll 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 [KDMF03, ZMGS17, BCK*13, BCGF10], using brushstrokes to fill regions [BVHT18, BLV*10], volumetric textures [BBT09] and screen-space advection [WDK*12, KP11]. 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: [KLK*00, HS04, HLB05, CTM08, FBS05]. However, none of these address the complete set of issues including stereo fusion and temporal coherence. Northam et al [NAK12] does address stereo coherence, but uses a technique inappropriate for real-time rendering.

Our multresolution texture method can be considered a generalization of the UV-gradient-based curved hatching described by Saito and Takahashi [ST90]. 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 [BLC*12] and Kalnins et al [KDMF03] do it in linear edge space; Klein et al [KLK*00] rely on pre-rendered "art maps" that do not account for real-time lighting or animation changes. Dirksen [Dir17] 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* [CEEGS16], 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 [Cur99].

1.2. Overview

We will start by describing our overall artistic design approach in Section 2. We will then describe the steps of our animation and rendering process, including the new techniques for MetaTexture, and Edge Breakup. Following that, we will show how we applied these techniques, and suggest future directions for further exploration.

2. Design Approach

Age of Sail's visual style is inspired by the work of painters like Bernie Fuchs [Apa17] and Thomas Hoyne [PP05]. 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 way of working with oils. 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* [WS94] to convey the feeling that an object is textured without explicitly rendering its texture everywhere in full detail, or *lost and found edges* [Rei09] which focuses the viewer's attention by carefully planning the locations of the most salient edges, and reducing their 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. We used a combination of these approaches in our designs.

To achieve this visual style in an immersive medium, we must make sure that its two-dimensional visual features adapt smoothly to the viewer's changing point of view, and also that they spatialize at the correct depth under stereoscopic fusion. We address this using new real-time rendering techniques for handling texture (described in Section 4) and roughened edges (Section 5). Because the style consists mainly of regions of solid color (or smooth gradations of color), we are free to mix and match different techniques to produce a seamless end result.

3. The Rendering Process

Our rendering pipeline consists of a series of 3D rendering passes interleaved with image-processing operations. In some cases, the output of an image processing pass is used as an input texture in a later rendering pass.

3.1. Color control

Each character and prop in the scene contains two full sets of texture maps, one for shadowed areas (Figure 2a), and one for illuminated areas (Figure 2b). These textures are hand-painted, and art-directed so that certain salient details may stand out only in shadows, and others only when lit. An optional third *breakup map* (Figure 2c) is used like a halftone pattern to indicate texture in areas

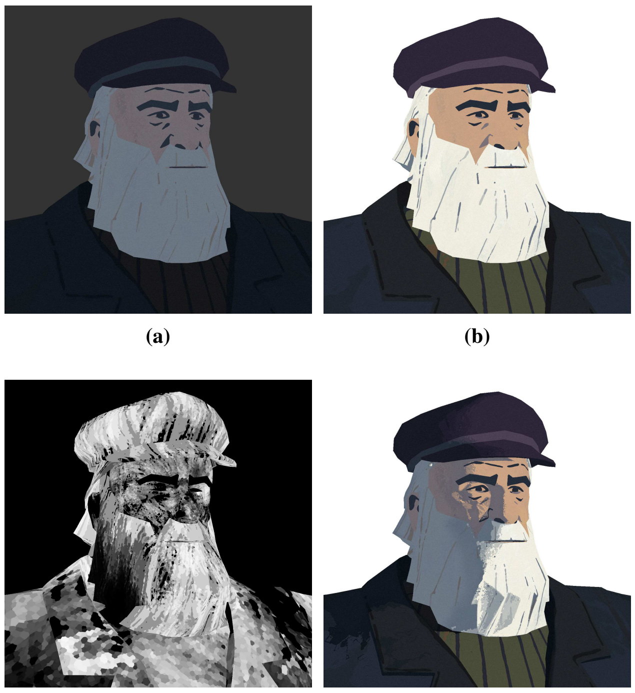

(a) (b) (c) (d)

Figure 2: *Texture examples: (a) shadow textures, (b) lit textures, (c) breakup map, (d) final result.*

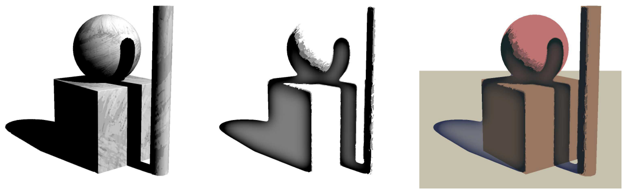

(a) (b) (c)

Figure 3: *(a) Lambert-shaded objects; (b) filtered for rounded corners, inner glow, and broken gradients; (c) final color pass.*

of gradation from light to shadow on certain surfaces (primarily on the characters).

3.2. Shadow shapes and inner glow

We render the shapes of the shadows using a combination of traditional toon shading (thresholded Lambert shading) and shadows cast from a single directional light source (Figure 3a.) The shadow depth map is rendered using warped spatial coordinates so as to provide more visual detail in the areas nearest the viewer. We then process the resulting image using a series of blurring and thresholding operations to produce simplified shapes with rounded corners, and add indications of texture only in areas oblique to the light direction. We reuse the blurred shape to lighten the interiors of shadow shapes, producing an inner glow effect (Figure 3b.) We then use the final shadow shape image to blend between the lit and shadowed textures (Figure 3c.)

4. MetaTexture

To achieve a hand-painted illusion in 6DoF VR, we need a texturing technique that adheres to the movement of surfaces in 3D space, while also matching the screen-space characteristics of the desired medium. 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 multiresolution texture made by blending multiple copies of a single tileable example texture (Figure 4a), 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 4c). 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.

4.1. Texture scales and blend coefficients

One way to determine texture scales is by using the magnitudes of the screen-space gradients of the texture coordinates $\vec{U} = \langle u, v \rangle$. Assuming a square example texture of width w pixels, we calculate an ideal texture scale exponent vector $\vec{E}$ as follows:

$$\vec{E} = \langle -\log_2(w|\nabla u|), -\log_2(w|\nabla v|) \rangle \tag{1}$$

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

$$\vec{U}_0 = \left\langle 2^{\lfloor E_u \rfloor} u, 2^{\lfloor E_v \rfloor} v \right\rangle \tag{2}$$

$$\vec{U}_1 = \langle 2u_0, v_0 \rangle \tag{3}$$

$$\vec{U}_2 = \langle u_0, 2v_0 \rangle \tag{4}$$

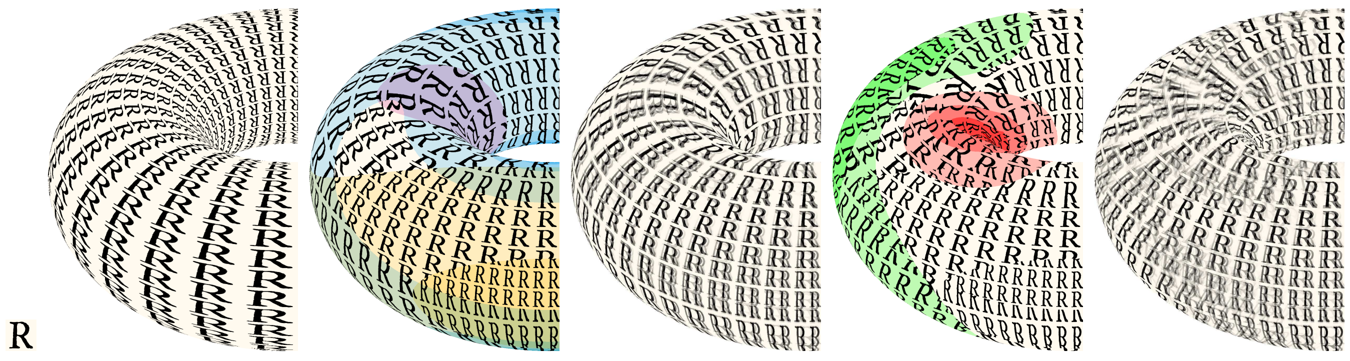
$$\vec{U}_3 = \langle 2u_0, 2v_0 \rangle \tag{5}$$

$$\vec{B} = \langle \beta(E_u - \lfloor E_u \rfloor), \beta(E_v - \lfloor E_v \rfloor) \rangle \tag{6}$$

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 (see Figure 4d):

$$\begin{aligned} M(\vec{U}) = (1-B_v)[(1-B_u)T(\vec{U}_0) + B_u T(\vec{U}_2)] + \\ B_v[(1-B_u)T(\vec{U}_1) + B_u T(\vec{U}_3)] \end{aligned} \tag{7}$$



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 4: *(a) Example texture. (b) Ordinary texture mapping produces undesirable distortion in screen space. (c) Quantized regions with UV scales adjusted to the nearest power-of-two, to approximate screen-space size of example texture. (d) MetaTexture with blending. (e) Quantized regions of positive (green) or negative (red) skew ($n = 4$). (f) MetaTexture with skew compensation and blending.*

4.2. 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 5.) 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 binormal ($\vec{T_s}$ and $\vec{B_s}$), and the world-space UV gradients $\nabla_W u$ and $\nabla_W v$:

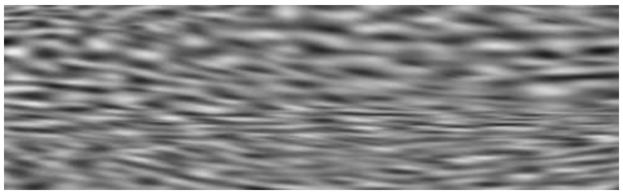
$$\nabla_S u = |\nabla_W u| \vec{T_s} / |\vec{T_s}|^2 \quad (8)$$

$$\nabla_S v = |\nabla_W v| \vec{B_s} / |\vec{B_s}|^2 \quad (9)$$

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 4.4.)

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 $\vec{T_s}$ and $\vec{B_s}$ 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.)

(a)



(b)

Figure 5: *(a) hard-edged artifacts resulting from discontinuous UV gradients. (b) approximate smooth gradients.*

4.3. 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 coefficient α, derived from the screen space position s and the camera projection matrix P, to equation 1.

$$\vec{Q} = \left\langle \frac{s_x}{P_{0,0}}, \frac{s_y}{P_{1,1}} \right\rangle \quad (10)$$

$$\alpha = |\vec{Q}|^2 + 1 \tag{11}$$

$$\vec{E} = \langle -\log_2(\alpha w|\nabla u|), -\log_2(\alpha w|\nabla v|)\rangle \tag{12}$$

4.4. 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 ∇u and ∇v diverges too far from 90°. (See Figures 6a and 4d).

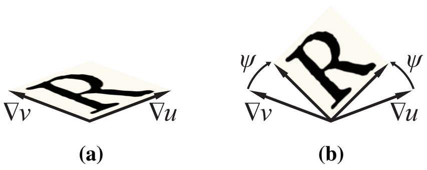


Figure 6: *(a) Skewed texture. (b) Compensating for skew by rotating the eigenvectors in UV space.*

To compensate for this distortion, we can rotate both eigenvectors in UV space by a certain angle ψ so that their screen-space projections are perpendicular (Figure 6b). To find ψ, we start by computing a *skew* factor σ as follows:

$$\sigma = \tan^{-1}(\varepsilon)\left(\frac{\nabla u}{|\nabla u|}\cdot\frac{\nabla v}{|\nabla v|}\right) \tag{13}$$

where

$$\varepsilon = \begin{cases} |\nabla u|/|\nabla v|, & \text{if } |\nabla u| \le |\nabla v| \\ |\nabla v|/|\nabla u|, & \text{otherwise} \end{cases} \tag{14}$$

We quantize the skew factor at a user-determined finite number of levels n (see Figure 4e; in practice, values between 2 and 4 seem to work well) and use those to calculate quantized angles ψ_0, ψ_1, and finally determine new UV coordinates $\vec{U}'$, $\vec{U}''$:

$$\sigma_0 = \frac{\lfloor n\sigma \rfloor}{n} \tag{15}$$

$$\sigma_1 = \frac{\lfloor n\sigma \rfloor + 1}{n} \tag{16}$$

$$\psi_0 = \tan^{-1}\left(\frac{\sigma_0}{2}\right) \tag{17}$$

$$\psi_1 = \tan^{-1}\left(\frac{\sigma_1}{2}\right) \tag{18}$$

$$\vec{U}' = \langle u\cos(\psi_0) + v\sin(\psi_0), v\cos(\psi_0) + u\sin(\psi_0)\rangle \tag{19}$$

$$\vec{U}'' = \langle u\cos(\psi_1) + v\sin(\psi_1), v\cos(\psi_1) + u\sin(\psi_1)\rangle \tag{20}$$

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 $\vec{U}'_{0..3}$ and $\vec{U}''_{0..3}$ using equations 2-5, and we calculate a third blend factor B_w, and MetaTexture as follows:

$$B_w = \beta(n\sigma - \lfloor n\sigma \rfloor) \tag{21}$$

$$\begin{aligned} M(\vec{U}) = (1-B_w)\{(1-B_v)[(1-B_u)T(\vec{U}'_0) + B_uT(\vec{U}'_2)]+ \\ B_v[(1-B_u)T(\vec{U}'_1) + B_uT(\vec{U}'_3)]\}+ \\ B_w\{(1-B_v)[(1-B_u)T(\vec{U}''_0) + B_uT(\vec{U}''_2)]+ \\ B_v[(1-B_u)T(\vec{U}''_1) + B_uT(\vec{U}''_3)]\} \end{aligned} \tag{22}$$

There is a tradeoff here: the reduced distortion comes at the cost of increasing the number of overlapping regions (Figure 4f). In practice, whether skew compensation is needed depends on the texture and the desired result.

4.5. 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 ∇u and ∇v. (See Figure 7.)

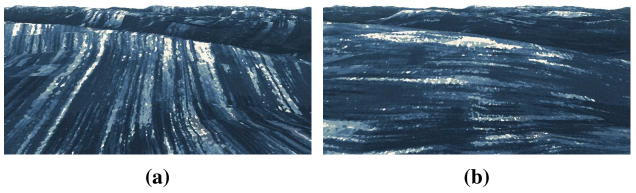


Figure 7: *An ocean surface with a clearly directional MetaTexture (a) in its default orientation, (b) re-oriented to indicate contour.*

5. Edge Breakup

We roughen the edges of the rendered scene by warping the image using a carefully structured vector field (see Figure 8). We create the vector field by rendering the same scene geometry with a different set of shaders into a separate buffer. These shaders use a two-channel texture to represent screen space warp vectors. (For best results, the texture should tile seamlessly and be self-similar, as in Figure 8a.) 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.

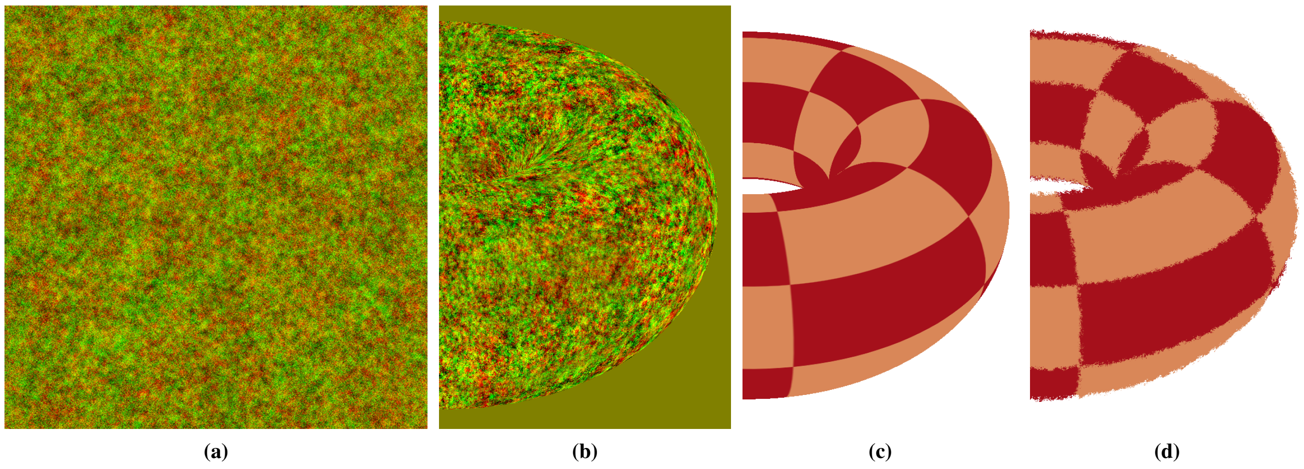


Figure 8: *(a) A self-similar, tileable noise texture. (b) Warp pass, with noise rendered with MetaTexture on inflated surface. (c) Color pass, with clean-edged rendered surface. (d) Edges roughened by warping.*

5.1. 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 9.)

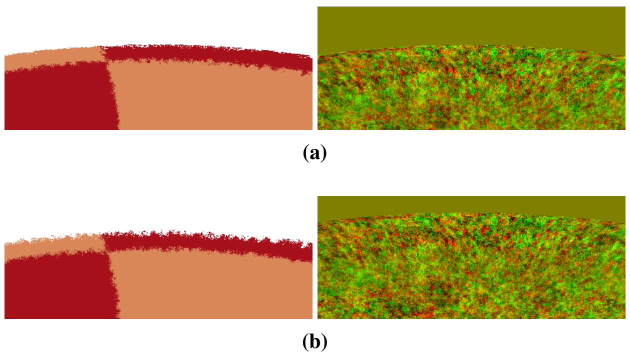


Figure 9: *The effect of edge breakup (a) without inflation and (b) with inflation. Note that without inflation, the edge breakup affects only pixels that are entirely within the object's silhouette, resulting in a hard outer edge.*

5.2. Compensating for camera roll

Because the red and green channels encode warp vectors in screen space, if we were to roll the camera 90 degrees (e.g. if the viewer tilts her head sideways), the warped edges would change their shape (see Figure 10.) 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.

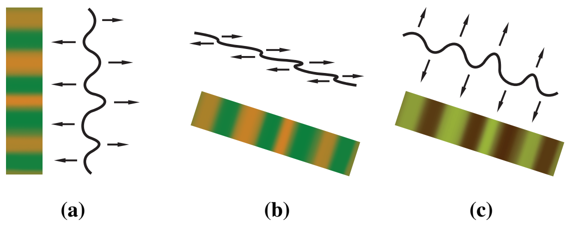


Figure 10: *(a) warping an edge using vectors encoded in the red/green channels. (b) warping with the same vectors after camera rotation produces a different shape. (c) rotating the vectors before encoding keeps the shape consistent.*

5.3. 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 11.)

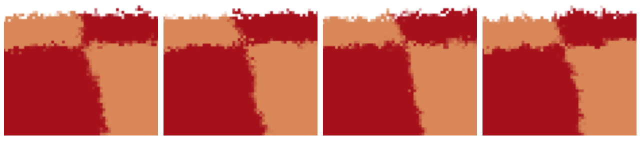


Figure 11: *Four frames from a scene with animated line boil.*

5.4. Compensating for distance

The above techniques will ensure a uniform degree of screen-space roughness, creating the illusion that the entire image was painted using the same tools and materials. However, in order to preserve

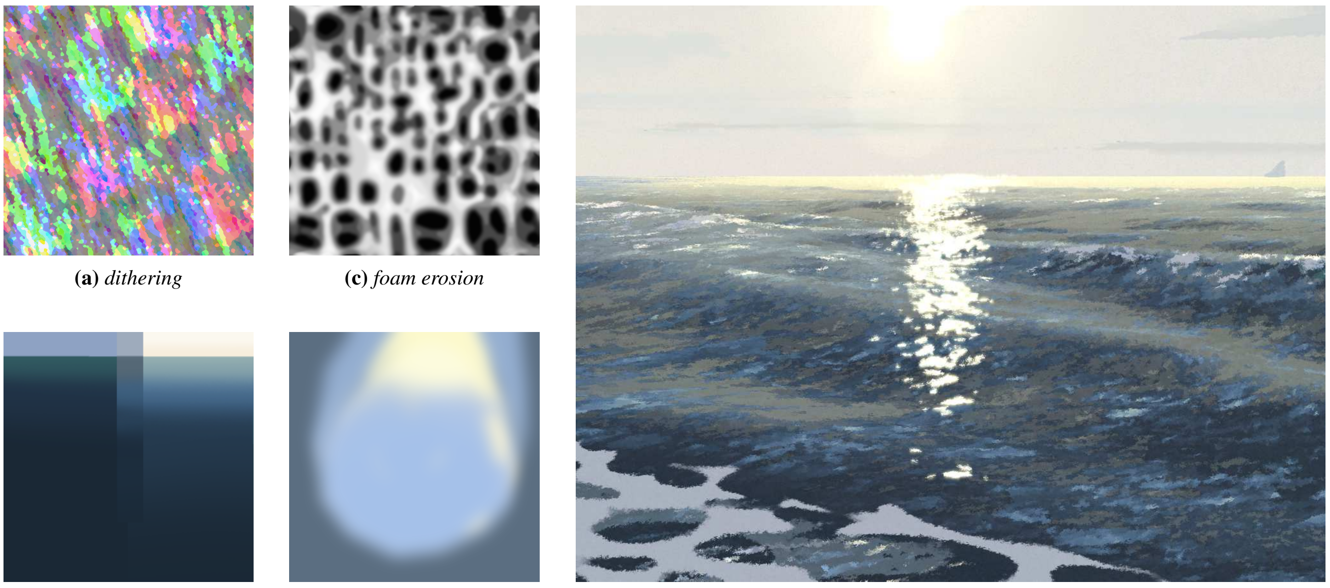

(a) *dithering* **(c)** *foam erosion* **(b)** *bathic map* **(d)** *sky reflection* **(e)** *The final rendered ocean.*

Figure 12: *The various textures that contribute to the appearance of the ocean shader.*

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 ρ based on the distance from camera d:

$$\rho' = \frac{\rho}{1+d^2} \tag{23}$$

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 changes in texture, color and movement as the weather conditions change. To reproduce this complex natural phenomenon in a real-time experience, we relied on the strategy of removing unnecessary visual information.

To capture the characteristic feeling of the shapes and movement of ocean waves, we used a very low-resolution mesh (approximately 70cm per quad in physical space) whose vertices are animated using a Tessendorf deep ocean wave model [Tes04]. This mesh is textured with a custom shader to add detail only where it's needed (Figure 12e).

The shader uses a hand-painted texture (Figure 12a) with MetaTexture to create an underlying dithering pattern that stays consistent in screen space regardless of viewpoint. This is then used as an index 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 12b.) Sea foam, based on a third texture (Figure 12c) is applied to the up-wind sides of the waves. A fourth texture acts as a sky reflection map (Figure 12d), 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 18).

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 13). 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

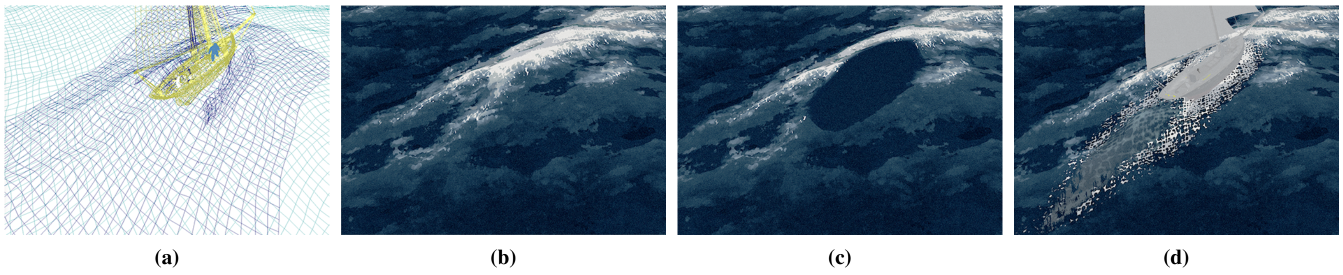


Figure 13: *Decorating the ocean: (a) ocean, wake and splash geometry; (b) ocean surface; (c) ocean with vertices deformed and texture suppressed in the area around the boat; (d) wake superimposed seamlessly on top.*

to synthetic. We also reduced the intensity of the edge breakup on facial features to keep their expressions clear (Figure 14), and on entire characters when they are far from camera.

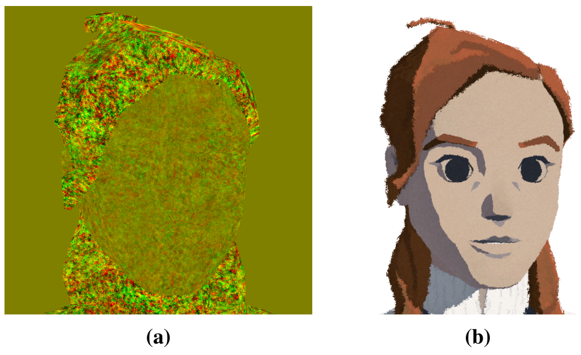


Figure 14: *Reduced edge breakup on facial features.*

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 [MDHH15].

We suspect that virtual reality may have its own version of the Uncanny Valley [MMK12]: 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 responsiveness, 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. 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 consistent 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.

7.2. 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 MetaTexture technique alone does not guarantee a consistent level of contrast across the entire image. Some regions may be dominated by a single texture (Figure 15a) while others may be a blend of up to eight different textures (Figure 15b), which significantly reduces feature contrast and salience. (A similar problem often occurs in "fractalized" textures [BTS09].) We have found this can be improved by using a histogram-preserving blending operator as described by Heitz and Neyret [HN18].

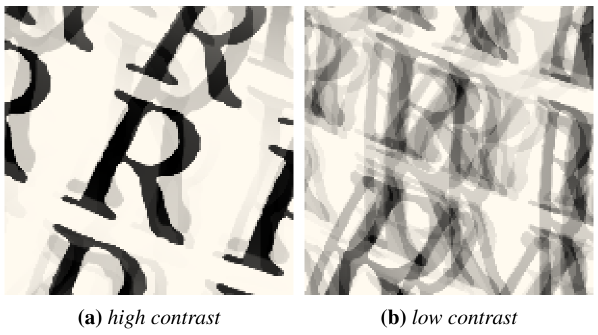


Figure 15: *Contrast inconsistency in MetaTexture.*

The edge breakup can produce a noticeable "heat ripple" effect when a foreground object occludes a background full of high-contrast detail (see Figure 16). In such circumstances it may be preferable to render and warp the scene in multiple layers, although that would come at a higher computational cost.

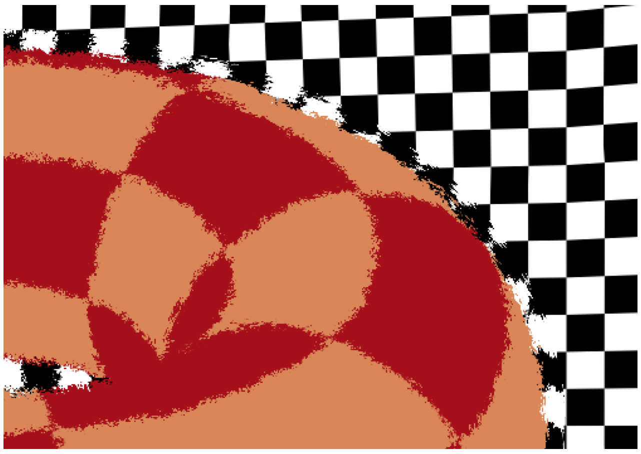


Figure 16: *"Heat ripple" effect (exaggerated here for clarity).*

7.3. 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 explore this further via a control experiment or survey similar to Michelle et al [MDHH15].

Our early experiments using these techniques in augmented reality (see Figure 17) 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?

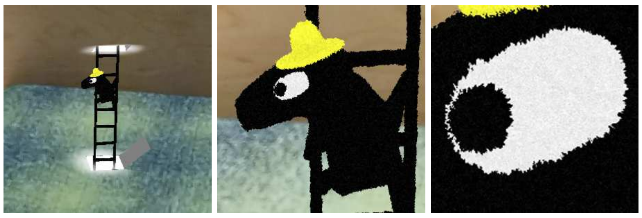


Figure 17: *A scene from augmented reality experiment* The End.

8. Acknowledgements

The authors would like to thank: Dan Kaufman for his support of our efforts, Rachid El Guerrab for asking an interesting question four years ago, Jan Pinkava, Karen Dufilho, and David Eisenmann for their unwavering leadership, Céline Desrumaux and Jasmin Lai for their inspiring production design, Josiah Larson for sweating the details, Tim J. Smith for psychological insights, Jon Klassen for designing and directing *The End*, and everyone at Google ATAP, Chromosphere LA, Boathouse Studios and Evil Eye Pictures for their contributions to *Age of Sail*.

References

[Apa17] Apatoff D.: *The Life and Art of Bernie Fuchs*. The Illustrated Press, 2017. 2

[BBT09] Bénard P., Bousseau A., Thollot J.: Dynamic solid textures for real-time coherent stylization. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 121–127. 2

[BCGF10] Bénard P., Cole F., Golovinskiy A., Finkelstein A.: Self-similar texture for coherent line stylization. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2010), NPAR '10, ACM, pp. 91–97. 2

[BCK*13] Bénard P., Cole F., Kass M., Mordatch I., Hegarty J., Senn M. S., Fleischer K., Pesare D., Breeden K.: Stylizing animation by example. *ACM Trans. Graph. 32*, 4 (July 2013), 119:1–119:12. 2

[BLC*12] Bénard P., Lu J., Cole F., Finkelstein A., Thollot J.: Active strokes: Coherent line stylization for animated 3D models. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (Goslar Germany, Germany, 2012), NPAR '12, Eurographics Association, pp. 37–46. 2

[BLV*10] Bénard P., Lagae A., Vangorp P., Lefebvre S., Drettakis G., Thollot J.: A dynamic noise primitive for coherent stylization. In *Proceedings of the 21st Eurographics Conference on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2010), EGSR'10, Eurographics Association, pp. 1497–1506. 2

[BTS09] Bénard P., Thollot J., Sillion F.: Quality assessment of fractalized npr textures: A perceptual objective metric. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization* (New York, NY, USA, 2009), APGV '09, ACM, pp. 117–120. 8

[BVHT18] Bléron A., Vergne R., Hurtut T., Thollot J.: Motion-coherent stylization with screen-space image filters. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2018), Expressive '18, ACM, pp. 10:1–10:13. 2

[CEEGS16] Curtis C., Eisenmann D., El Guerrab R., Stafford S.: The making of "Pearl", a 360° Google Spotlight Story. In *ACM SIGGRAPH 2016 VR Village* (New York, NY, USA, 2016), SIGGRAPH '16, ACM, pp. 21:1–21:1. 2

[CTM08] Chen J., Turk G., MacIntyre B.: Watercolor inspired non-photorealistic rendering for augmented reality. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2008), VRST '08, ACM, pp. 231–234. 2

[Cur99] Curtis C.: Non-photorealistic animation. *ACM SIGGRAPH 1999 Course Notes 17* (1999). 2

[Dir17] Dirksen N.: Creating a VR storybook look for Rainbow Crow. `https://www.youtube.com/watch?v=uELM5qQvBkY`, Nov 2017. 2

[FBS05] Fischer J., Bartz D., Strasser W.: Artistic reality: Fast brush stroke stylization for augmented reality. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2005), VRST '05, ACM, pp. 155–158. 2

[HLB05] Haller M., Landerl F., Billinghurst M.: A loose and sketchy approach in a mediated reality environment. In *Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (New York, NY, USA, 2005), GRAPHITE '05, ACM, pp. 371–379. 2

[HN18] Heitz E., Neyret F.: High-performance by-example noise using a histogram-preserving blending operator. *Proc. ACM Comput. Graph. Interact. Tech. 1*, 2 (Aug. 2018), 31:1–31:25. 8

[HS04] Haller M., Sperl D.: Real-time painterly rendering for MR applications. In *Proceedings of the 2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South

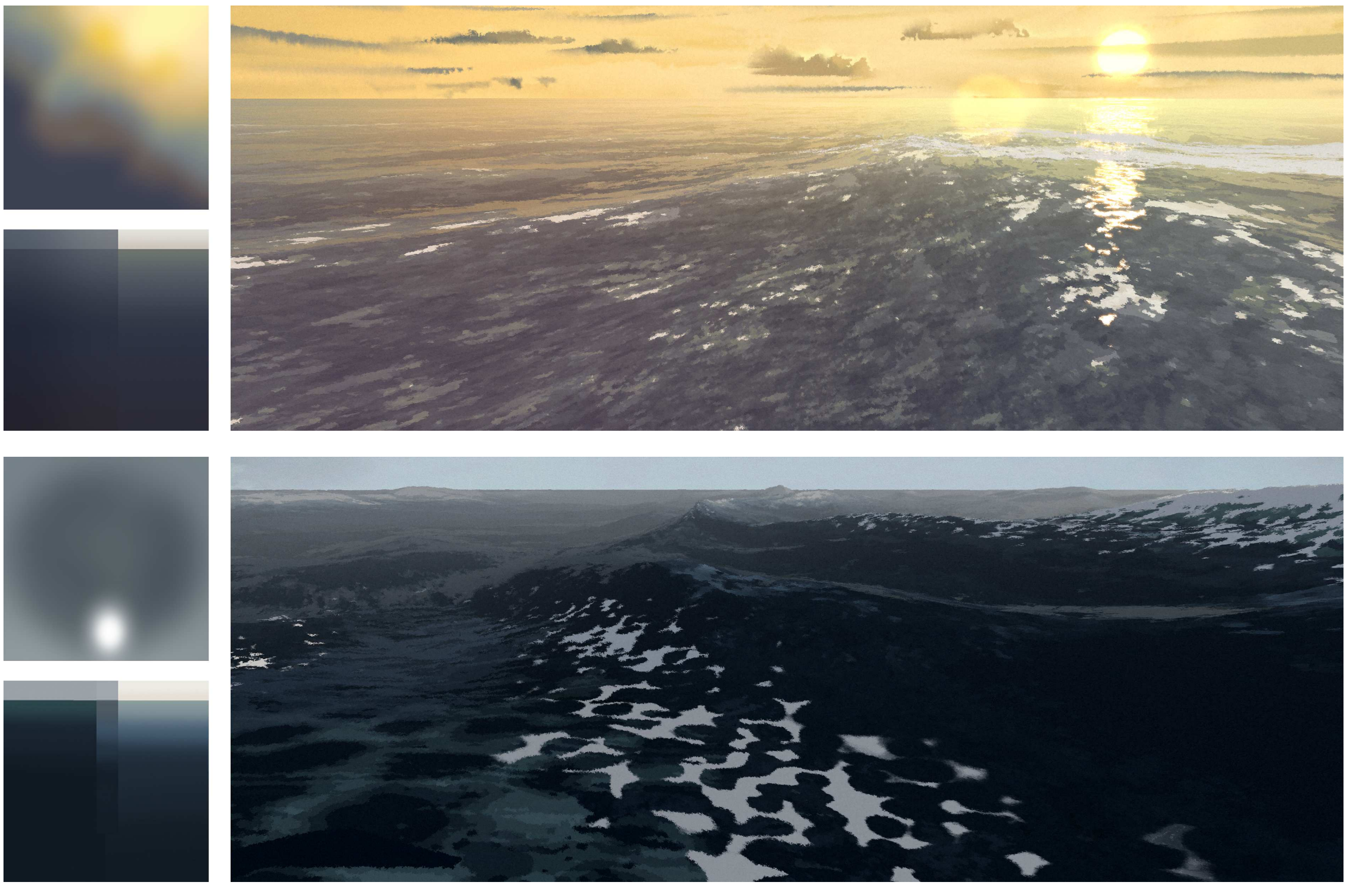


Figure 18: *Two other moments from* Age of Sail*, with their respective sky and bathic maps.*

East Asia (New York, NY, USA, 2004), GRAPHITE '04, ACM, pp. 30–38. 2

[KDMF03] KALNINS R. D., DAVIDSON P. L., MARKOSIAN L., FINKELSTEIN A.: Coherent stylized silhouettes. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 856–861. 2

[KLK*00] KLEIN A. W., LI W., KAZHDAN M. M., CORRÊA W. T., FINKELSTEIN A., FUNKHOUSER T. A.: Non-photorealistic virtual environments. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 527–534. 2

[KP11] KASS M., PESARE D.: Coherent noise for non-photorealistic rendering. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 30:1–30:6. 2

[MDHH15] MICHELLE C., DAVIS C., HIGHT C., HARDY A.: The Hobbit hyperreality paradox: Polarization among audiences for a 3D high frame rate film. *Convergence: The International Journal of Research into New Media Technologies 23* (05 2015). 8, 9

[MMK12] MORI M., MACDORMAN K. F., KAGEKI N.: The uncanny valley [from the field]. *IEEE Robotics Automation Magazine 19*, 2 (June 2012), 98–100. 8

[NAK12] NORTHAM L., ASENTE P., KAPLAN C. S.: Consistent stylization and painterly rendering of stereoscopic 3d images. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (Goslar Germany, Germany, 2012), NPAR '12, Eurographics Association, pp. 47–56. 2

[PP05] PALLEY R., PALLEY M. A.: *Wooden Ships and Iron Men: The Maritime Art of Thomas Hoyne*. Quantuck Lane Press, 2005. 2

[Rei09] REID C.: Planning lost and found edges. *Watercolor Artist Magazine* (2009). 2

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 197–206. 2

[Tes04] TESSENDORF J.: Simulating ocean surfaces. *ACM SIGGRAPH 2004 Course Notes 32* (2004). 7

[WDK*12] WHITED B., DANIELS E., KASCHALK M., OSBORNE P., ODERMATT K.: Computer-assisted animation of line and paint in disney's paperman. In *ACM SIGGRAPH 2012 Talks* (New York, NY, USA, 2012), SIGGRAPH '12, ACM, pp. 19:1–19:1. 2

[WS94] WINKENBACH G., SALESIN D. H.: Computer-generated pen-and-ink illustration. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), SIGGRAPH '94, ACM, pp. 91–100. 2

[ZMGS17] ZHENG M., MILLIEZ A., GROSS M., SUMNER R. W.: Example-based brushes for coherent stylized renderings. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2017), NPAR '17, ACM, pp. 3:1–3:10. 2