Shading Rig: Dynamic Art-Directable Stylised Shading for 3D Characters

LOHIT PETIKAM, Computational Media Innovation Centre, Victoria University of Wellington, New Zealand (NZ)
KEN ANJYO, OLM Digital, Japan and Computational Media Innovation Centre, Victoria University of Wellington, NZ
TAEHYUN RHEE, Computational Media Innovation Centre, Victoria University of Wellington, NZ

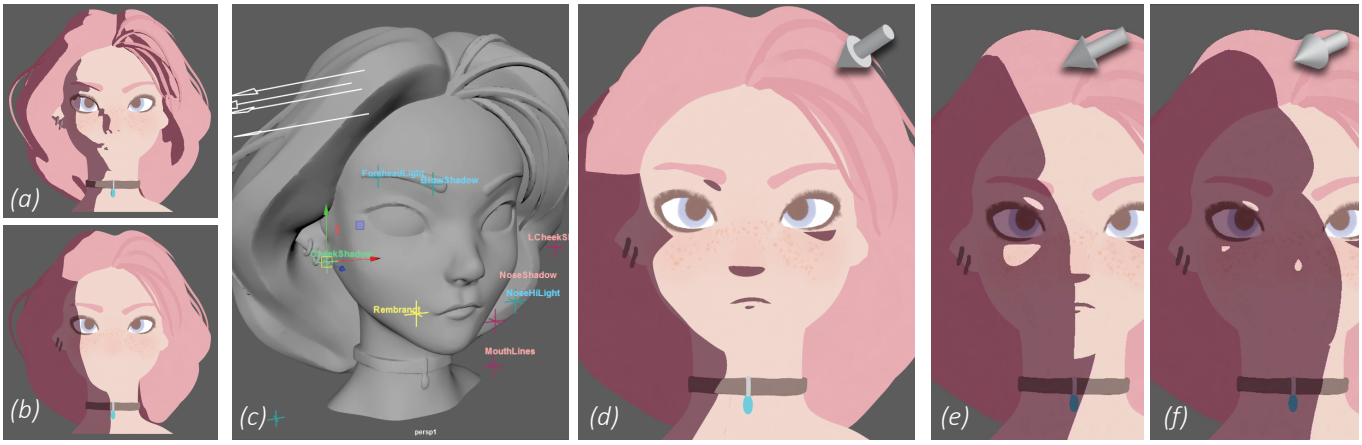


Fig. 1. (a) Conventional 3D toon shading requiring clean-up and editing. (b) Result after normal smoothing which lacks detail. (c) Our parametric edits (labelled 3D controllers) to build a "shading rig". (d) Edited result from the shading rig being applied to (b), with light direction shown in the top-right. Note the retained mouth and nose definition. (e, f) Animating the shading rig with lighting changes, with stylistic details added such as the dynamic Rembrandt triangle. Model by Georgiy Nomerovsky licensed under CC BY.

Despite the popularity of 3D animation techniques, the style of 2D cel animation is seeing increased use in games and interactive applications. However, conventional 3D toon shading frequently requires manual editing to clean up undesired shadows or add stylistic details based on art direction. This editing is impractical for the frame-by-frame editing in cartoon feature film post-production. For interactive stylised media and games, post-production is unavailable due to real-time constraints, so art-direction must be preserved automatically. For these reasons, artists often resort to mesh and texture edits to mitigate undesired shadows typical of toon shaders. Such edits allow real-time rendering but are limited in resolution, animation quality, and lack detail control for stylised shadow design.

In our framework, artists build a "shading rig", a collection of these edits, that allows artists to animate toon shading. Artists pre-animate the shading rig under changing lighting, to dynamically preserve artistic intent in a live application, without manual intervention. We show our method preserves continuous motion and shape interpolation, with fewer keyframes than previous work. Our shading shape interpolation is computationally cheaper than state-of-the-art image interpolation techniques. We achieve these improvements while preserving vector quality rendering, without resorting either to high texture resolution or mesh density.

CCS Concepts: • **Computing methodologies** → **Non-photorealistic rendering**; *Animation*.

Additional Key Words and Phrases: Toon shading, shadow editing, shadow animation, dynamic lighting, distribution interpolation

ACM Reference Format:
Lohit Petikam, Ken Anjyo, and Taehyun Rhee. 2021. Shading Rig: Dynamic Art-Directable Stylised Shading for 3D Characters. *ACM Trans. Graph.* 1, 1, Article 1 (January 2021), 14 pages. https://doi.org/10.1145/3461696

1 INTRODUCTION

Cel animation remains a prevalent form of artistic expression. The freedom offered by manual drawing largely attracts content creators to develop hand-drawn comic and cartoon style characters for feature films and interactive media such as games. Using digital tools, artists create 3D models from character artwork for automated rendering from any viewpoint. However, many manual appearance edits are required to keep 3D stylised rendering faithful to hand-drawn media. This is important for preserving the original appeal and emotional response elicited by the character and story [Guertault et al. 2018; Katsura and Anjyo 2007; Motomura 2015; Wood et al. 1997].

Authors' addresses: Lohit Petikam, lohit.petikam@vuw.ac.nz, Computational Media Innovation Centre, Victoria University of Wellington, New Zealand (NZ); Ken Anjyo, anjyo@acm.org, OLM Digital, Japan, Computational Media Innovation Centre, Victoria University of Wellington, NZ; Taehyun Rhee, taehyun.rhee@vuw.ac.nz, Computational Media Innovation Centre, Victoria University of Wellington, NZ.

0730-0301/2021/1-ART1 $15.00
https://doi.org/10.1145/3461696

One of the key challenges of 3D cartoon media is controlling the light and shadow areas generated by conventional toon shading [Lake et al. 2000]. This is because the abstract two-tone shading style creates undesirable shadow shapes, or removes important surface details, when illumination changes. Even in realistic rendering styles, artists require fine-grained, non-physical, shadow shape control [Burley et al. 2018]. Stylised feature films thus require frame-by-frame editing during post-production which is inefficient, especially when repeating the same manual edits in different shots. For interactive stylised media and games, post-production is not possible, as frames need to be synthesised and displayed at real-time rates, under varying illumination. As a consequence, stylised interactive content features limited interactions, fixed lighting and inflexible camera movements, to keep pre-production costs manageable [Motomura 2015].

Considering that the use of 3D cartoon characters will increase drastically, we face greater challenges for stylised interactive media and game creation. A new approach is desired to enable artist-defined toon shading edits that automatically adapt to dynamic lighting in real-time. Unfortunately only a few approaches exist towards this goal. For example, mesh-based editing methods support fixed lighting, but require further manual tweaking under dynamic lighting [Motomura 2015]. Todo et al. [2007] support keyframing of edits across an animation sequence. However their shadow animation is discontinuous, unless many keyframes are added to fill in the gaps. The flexibility of rotoscoping vector masks, commonly used for shade editing in offline films, cannot support real-time dynamic animation on 3D surfaces.

Real-time 2D shape interpolation offered by optimal mass transport [Nader and Guennebaud 2018] could solve the discontinuous shadow animation problem. However, this is limited by texture resolution where artists strive for infinite resolution vector quality [Guertault et al. 2018; Motomura 2015].

We present the "Shading Rig", a framework for designing and animating dynamic art-directed 3D toon shading. We design a novel parametric model for generating shading edits, that allow for continuous, real-time, art-directed shadow animation. In particular, our model can generate edits to existing shade, rather than designing arbitrary shadows from scratch. Unlike the static offset functions of Todo et al. [2007], our edits are directly manipulable, and animatable, while preserving vector-quality detail. Notably, our method allows artists to create and animate edits across a surface mesh, allowing for both time or light changes. Animators can then precisely pre-animate how light and shadow fall on a character as determined by art direction. Interactive applications can then reproduce the artist-defined shading variation, at a much lower performance cost than real-time image-based shape interpolation [Nader and Guennebaud 2018]. Most importantly, as illustrated in Figure 1 our shading rig enables dynamically art-directed rendering in interactive cartoon media, without artist intervention. We summarise our contributions as follows:

- We present a novel parametric model for 3D toon shading edits for real-time, animatable, vector-quality shade editing.
- Our shading edits provide continuous motion and shading shape interpolation, with fewer keyframes than previous approaches such as Todo et al. [2007].
- Our shape model is computationally cheaper than state-of-the-art real-time 2D shape interpolation techniques such as [Nader and Guennebaud 2018].

The remainder of this paper includes a state of the art review, followed by our approach. Section 4 provides a mathematical description of our model, followed by the experimental evaluation and comparison to previous techniques in Section 5. Section 6 discusses limitations of our technique and proposes future work.

2 RELATED WORK

In this section we survey prior work in toon shade editing for both offline and real-time applications. We also review patch-based stylisation work for artist-defined shading animation, as well as implicit and texture-based 2D shape interpolation for representing 2D toon shadow shapes.

2.1 Toon Shade Editing

To reduce the likelihood of problematic shadows from conventional toon shading [Lake et al. 2000], vertex normal smoothing [Barla et al. 2006] helps to simplify shadows, but this removes high-frequency details. Other mesh-based editing techniques such as normal or UV editing [Guertault et al. 2018; Motomura 2015] require high mesh density and the edit cannot be animated across the surface mesh. Previous works that allow for stylised toon shading are still limited in controllability. 2DToonShade [Hudon et al. 2019] can be used to control toon shading on 2D drawings. The work of Todo et al. [2013] allows a user-created lit-sphere image to arbitrarily define stylised shading gradation on 3D surfaces. However, for complex characters, shading edits on a lit-sphere (or similar parametrisation [Pacanowski et al. 2008]) cannot be made locally without unpredictably affecting other areas. Vanderhaeghe et al. [2011] use procedural shading primitives to build custom stylised shaders, but these also lack local control. Our problem concerns dynamic toon shading control, rather than producing line details via contour rendering [Bénard et al. 2014].

The paint interface of Todo et al. [2007] allows artists to edit toon light and shade regions arbitrarily on a mesh. Smooth offsets are made to the underlying intensity distribution of the key light. However, its interpolation between keyframed intensity distributions is limited, which can require many keyframes for continuous animation. Arief et al. [2015] provide local lit-sphere shading edits while preserving the artist defined shading style. The edits are parametric, allowing for smooth animation, but the shape of each edit is not controllable. Anjyo and Hiramitsu [2003] develop parametrically shapable view-dependent specular highlights, suitable for animation. However, we also require stable behaviour for diffuse light and shadow. We also use a parametric model for continuous movement and shape interpolation, but use parameters based on general shading principles, for fine-grained edits beyond specular highlights.

Table 1. Previous solutions for dynamic artist-defined shade editing (including 2D shape interpolation). Our approach achieves all requirements simultaneously.

	(1) Real-time Animation	(2) Vector representation	(3) Sharp Cusps	(4) Interactive previs
[Jamriška et al. 2019]	No	Limited by source resolution	Limited by source resolution	No
[Nader and Guennebaud 2018]	Yes	No	No	No
[Guertault et al. 2018; Motomura 2015; Todo et al. 2007]	With dense keyframing	Yes	Limited by geometry	Yes
[Dokter et al. 2019]	No	Yes	Yes	Yes
[Turk and O'Brien 2005]	Yes	Yes	No	Yes
Our approach	Yes	Yes	Yes	Yes

2.2 Artist-Defined Lighting and Stylised Shading

Many existing techniques allow artistic design or editing of realistic lighting 3D scenes. The method of Okabe et al. [2007] allows users to draw the desired realistic illumination on a 3D model. However, it only solves for global lighting instead of local edits. Tada et al. [2012] solve local illumination adjustments using RBF interpolation spatially between control points. WYSIWYG NPR [Kalnins et al. 2002] lets artists directly draw shading on 3D surfaces as simple line strokes. Several work support freeform editing of occlusion shadowing from 3D meshes [Mattausch et al. 2013; Pellacini et al. 2002; Ritschel et al. 2010]. Many other art-directable lighting methods also allow artists to non-physically control physical rendering phenomena [Schmidt et al. 2016]. These produce plausibly realistic images whereas toon shade editing often requires implausible shadow shape manipulation.

Artist-defined stylised shading and animation has been achieved with patch-based methods. One such method is image analogies [Hertzmann et al. 2001] for stylising animation [Bénard et al. 2013] and video [Jamriška et al. 2019] using image features. Given a few manually stylised video frames, these methods can preserve local stylisation across an image sequence. However, they are not real-time, and are limited to fixed lighting. Real-time adaptions of this approach also only support global shading [Fišer et al. 2016; Sýkora et al. 2019] with no accommodation for local editing.

2.3 2D Shape Interpolation

We also survey prior work in 2D shape interpolation as a potential solution to stylised shadow shape interpolation. Level sets of an implicit radial basis function (RBF) representation have been used to smoothly interpolate between arbitrary 2D shapes [Turk and O'Brien 2005]. However, these fail to reconstruct sharp cusps. While analytic 2D signed distance fields (SDFs) can reproduce sharp points, and smoothly blend together, these only produce limited geometric shapes. Recently, Seyb et al. [2019] proposed to freely deform 3D SDFs using a convex hull, but this has not been adapted to 2D shapes.

Animatable 2D vector masks are commonplace for freeform 2D shape design and direct manipulation, with infinite resolution. These are used in an offline 2D compositing or rotoscopy context to create and edit shade in 3D cartoon animation [Troftgruben 2014]. Vector Shading Curves [Lieng et al. 2015] have been used to define stylised shading details but only in 2D. Eisemann et al. [2008] convert toon shaded 3D renderings into vector art. Loop and Blinn [2005] encode 2D vector paths on triangular meshes providing vector shape rendering under arbitrary 3D projective transformations, suited for GPU rendering [Kilgard and Bolz 2012]. While real-time on the GPU, this method offers limited dynamic shape manipulation without costly re-computation of the mesh. To our knowledge, current research cannot support animated vector texture mapping in real-time applications (only precomputed static vector textures [Dokter et al. 2019; Ray et al. 2005; Sun et al. 2012; Wang et al. 2010]).

Fully controlled shading can be rendered as a hand-drawn static texture map but cannot respond to lighting changes. Optimal mass transport can provide this texture animation using displacement interpolation [Bonneel et al. 2011; Solomon et al. 2015]. For realistic images, this has been used for realistic lighting transfer on faces [Shu et al. 2018]. Recently, Nader and Guennebaud [2018] have achieved real-time performance for approximate mass transport. Although this provides fast arbitrary shape interpolation, there is still dependence on image resolution. While higher resolutions can reduce pixelation artifacts upon magnification, the sampling rate of image textures mapped on 3D meshes would be inconsistent at different viewpoints. Production game artists also go to great lengths to maintain vector quality to support extreme close-ups [Motomura 2015], so we target a fully vector-based approach.

2.4 Summary of Prior Work and Comparison

We summarise the limitations of prior work, as related to production requirements. Based on industry use cases [Guertault et al. 2018; Motomura 2015], we highlight four main requirements for interactive stylised media:

1) Real-time: Toon shading must render and animate in real-time and minimally impact computational resource budget.

2) Vector representation: Shading boundaries must be sharp, without pixelation artifacts, regardless of viewing distance.

3) High-frequency detail: Sharp cusps must remain sharp upon close-up, independent of mesh/texture resolution.

4) Interactive previsualisation (previs): editing, animation and previsualisation of shade variation must minimally impact artist turn-around time.

In table 1 we compare the state-of-the-art from each approach against these requirements. In this table we note that mesh editing [Guertault et al. 2018; Motomura 2015] and the method of Todo et al. [2007] can only reproduce sharp cusps as long as mesh density and topology allows. Without a high density mesh, this limits the shadows to simple polygonal shapes which restricts artistic

expression. Similarly, we classify the patch-based method of Jamriška et al. [2019] to have texture independence and sharp cusps, due to its ability to sample high-frequency patches in the source textures without blurring due to magnification. We consider the real-time shape interpolation method of Nader and Guennebaud [2018] to fail the interactive previsualisation requirement. Using higher resolution images dramatically increases this method's necessary precomputation time. Previsualising an animation would then become non-interactive, and in turn prohibitive for practical use. Our approach is the first to simultaneously support all the above requirements. We achieve this by designing a parametric model based on artistic shading principles identified in the next section.

3 SHADING RIG OVERVIEW

In this section we give an overview of our shading rig workflow and interface, to shape and animate toon shading. We also design our parameters for generating toon shade edits, considering basic light and shadow elements described in artistic shading literature (Section 3.2). Section 4 gives the formulation of the features described in this section.

3.1 Shade Edits and Shading Rig

Inspired by the key light and fill light paradigm familiar to artists [Birn 2000; Landau 2014], we define discrete shading edits, controlled like point lights but with shapeable intensity distributions. Figure 2 explains the overview of the workflow with our method: A directional light is firstly specified as the key light providing the base shading on a 3D mesh. Then our edits are used like shapeable fill lights for local shading adjustments. Artists control the shape of an edit using the parameters which we define in Section 3.2. We refer to the collection of edits, and their parameters, as the "shading rig". The edits are scaled in size based on their distance to the surface, to match the behaviour of point lights in toon shading.

With multiple edits, complex shapes can be built with the expressivity of offset painting, but can be tweaked and animated with the flexibility of conventional lighting. Figure 3 shows this shade editing process, using our prototype interface in Maya. In this example, an edit (represented by a locator in 3D space) is brought to the surface of the mesh. This first expands the lit region around the edit radially. The edit's shape parameters can then finely control the boundary of light and shade.

To preserve art direction under dynamic lighting, the shading rig is animated with changes in the key light direction. When the key light is rotated, the shadow boundary changes. The shading rig is then adjusted to achieve the desired shading in this new light direction. The adjustment is keyframed at this light direction. Section 4.2 describes how we interpolate the shading rig between light directions.

Artists may add more keyframes for different key light directions. Figure 5 illustrates this process for a single edit. Figure 5c) also shows the problem that static edits become invalid when lighting changes, and must be dynamically corrected like in Figure 5d).

3.2 Shading Edit Parameters

In the literature on artistic shading principles used by professional artists, Hogarth [1991] distils individual two-tone shapes of light and shadow in the following ways. Abstract, rounded shadow shapes are described as "islands of minimal light", "droplets of light" for selectively expressing certain forms. Rounded "deep pools of shadow" are described to shadow eyes under the brow of a head. More directional shapes are described as "linear highlight[s]" or resemble "arcs of light" or a "curving slash" that "thickens at certain points", and "tapers and gradually disappears". Based on these descriptions we design parameters below. The behaviour of these parameters on an edit's shape is shown in Figure 4.

First, for rounded "pools of shadow", circular shapes are produced by our model by default. In our framework this is equivalent to using point light sources to add radial light locally on a surface. The *Intensity Gain* parameter controls the shape's influence on shading. This is similar to controlling intensity in conventional lighting. We allow negative intensities to create shadows.

We add an *Anisotropy* parameter, to elongate circular shapes for the "linear highlight[s]", and a *Sharpness* parameter for edits to "taper" at the ends. Varying this parameter smoothly transitions between rounded and sharp endpoints. We add a *Bend* parameter to curve the elongated shapes into the "curving slash[es]" and "arcs of light". Its magnitude controls the degree of shape curvature. For curved shapes that "[thicken] at certain points", we provide a *Bulge* parameter. While Anisotropy alters thickness, Bulge asymmetrically controls where thickening occurs along a curve.

A *Rotation* parameter allows rotating the shape to align it with surface details. Each edit is also given a *Softness* parameter to achieve soft boundary edges in local regions, as required when depicting certain rounded forms [Guertault et al. 2018; Hogarth 1991].

To control the influence of surface details on the shape, we introduce a *Normal Smooth* parameter to smooth surface normals for simplified shading [Barla et al. 2006]. Increasing the Normal Smooth value will smooth out surface details where simpler shading is desired. Decreasing its value helps the shape to conform with surface details. This behaviour allows an edit to "repeat the external contours" Hogarth [1991] to reveal ridges and complex detail on a surface. The normal smooth parameter also helps to achieve complex or simple shading, as varied depending on character traits [McCloud 1993].

3.3 Edit Blending and Deformation

Custom shapes beyond the range expressed by our parameters are achieved by combining or blending individual edits. Our framework uses two types of edits for blending shadow areas: *intensities* and *masks*. *Intensity edits* produce shapes that smoothly blend with base shading, and with other edits. These edits create smooth joins between other shaded areas. *Masks edits*, however, create binary regions of light and shade that use boolean union and subtract operations to blend with other shading regions. This helps fine-scale edits hold their shape under lighting variation (unless they are specifically animated with the key light). These edits create sharp joins when combined with other shaded areas.

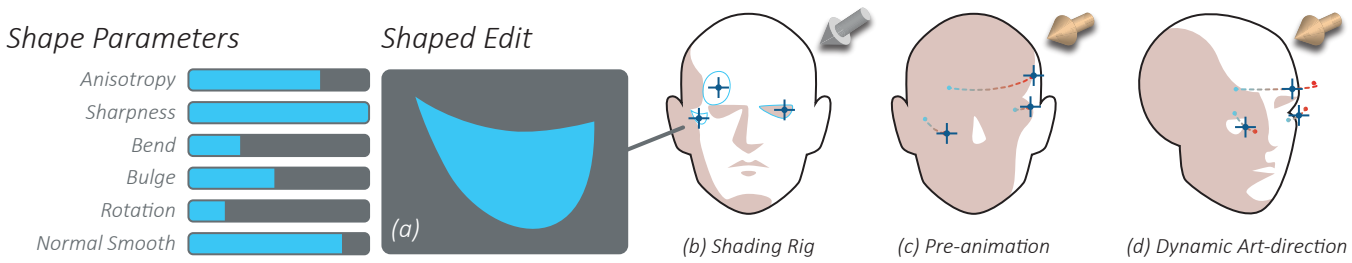


Fig. 2. Overview of our approach: a) Artists use our parameters to shape discrete shadow edits. b) Artist applies several edits on a character to art-direct shading, forming a shading rig. c) Artist animates the shading rig to art-direct shadow transitions as the key light direction changes. d) The art-directed shadow animation is automatically reproduced during lighting and pose changes.

Fig. 3. From left to right: key light adjusted for base toon shadows to be edited. Edit placed near surface mesh to adjust shading boundary. The edit is shaped using our parameters.

Another useful mode for each edit is *Track Deformation*, which fixes an edit's shape to a moving or deforming surface during animation. Both Intensity and Mask type edits can be set to track surface deformations. This saves artists from having to manually animate edits to follow a deforming surface. It also matches the static shading behaviour of texture or vertex painting while remaining animatable like a point source. Our edit blending modes and deformation tracking behaviour are visualised in Figure 6.

3.4 Local Editing Control

Like conventional point lighting, each edit has a user-defined radius of influence, beyond which the edit's intensity falls off and has no influence on shading. With this distance-based fall-off and the previously mentioned shape scaling control, animating edits to smoothly appear and disappear is achieved by moving them toward or away from a surface. For Intensity edits, the Intensity Gain parameter can also be used.

Since our edits can be shaped, we require a direction in which to project the shaped areas of light and shade. By default, we direct the projection toward the centroid of the mesh. For complex editing on geometry features away from the centroid, users specify a new point that we call the light origin. Moving the light origin allows edits to project shaped lighting toward the region of interest. The light origin also controls where normal smoothing occurs, as explained in the next section.

Many light origin points can be used. For example, edits made on a character's head would need a separate light origin, than for edits on the arm. Several light origin points, assigned to different edits, can be used in different regions of the character.

Table 2. Shading shape model control parameters.

Parameter	Symbol	Effect on Shape
Anisotropy	a	Elongates the shape horizontally.
Sharpness	s	Creates round or sharp endpoints for the elongated shape.
Bend	w_y	Curves the elongated shape.
Bulge	w_x	Asymmetrically bulges one side of the elongated shape.
Rotation	θ_r	Rotates the final shape for different orientations.
Normal smooth	τ	Controls how much geometry detail is preserved, or smoothed out, in the shape.
Intensity Gain	G	Intensity edit's influence on the existing shading.
Softness	d	Mask edit's boundary softness for smooth gradation between light and shade.

We integrate the functionality into an existing 3D application workspace such as Maya or Blender, which facilitates 3D manipulation, real-time preview, and animation. Within the 3D application, users are able to animate the position and shape parameters of each edit, to achieve the desired shadow movement under dynamic lighting. In the next section we give the mathematical formulation of our shading rig model to achieve the functionality described here.

4 SHADING RIG EDIT MODEL

This section firstly describes the mathematical formulation of our shading rig model, given the parameters and interface described in the previous section. Later, we describe how we implement deformation tracking, and animating the shading rig with lighting changes. For reference we list all the user parameters in Table 2. A full table of all symbols and notation is given in the supplemental material.

We map the shape of each edit to a local texture space projected on a surface mesh. The shape is generated by a warped intensity distribution and then thresholded for a binary mask of light and shade.

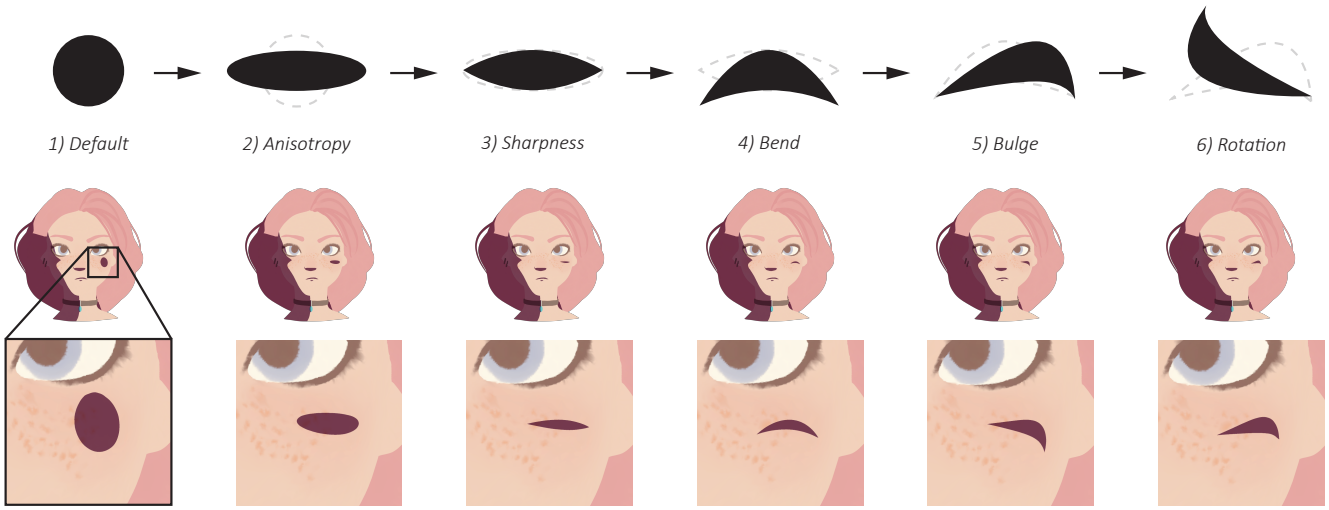


Fig. 4. Top row: Example sequence of shape manipulation for one shading edit, with the parameter being changed below (previous shape shown with dotted-lines). Underneath is the same manipulation on a surface mesh.

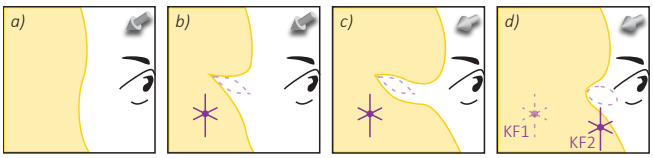


Fig. 5. Steps to animate a single edit with lighting changes. a) Unedited shading from key light. b) Shading rig edit placed to edit shading boundary. c) Key light rotated and shading must be edited again. d) First edit is keyframed to move and change shape under new lighting. The edit's position and shape interpolate between keyframes KF1 and KF2 as the key light direction changes.

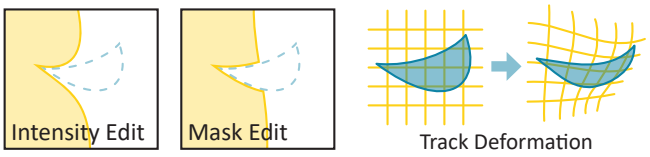


Fig. 6. Our *Intensity* and *Mask* edit types provide smooth or sharp blending between shadow areas. Edited shade can be set to track surface deformations during animation.

While 3D software provides texture projection interfaces for local decal placement, we implement a simple texture space behaving similarly to point lighting. Specifically, this allows our parametric distribution (Section 4.1) to exhibit the distance-based shape scaling mentioned in the previous section.

The texture space is translated across the surface by moving a point $\mathbf{p}_l$ in 3D space (following the analogy of moving a point light at position $\mathbf{p}_l$). We continuously direct the texture space projection toward the light origin point $\mathbf{p}_o$. As mentioned in the previous section, this is set to the mesh centroid by default, but is moved by the artist as necessary.

Our projection uses the light space $[\mathbf{l_x l_y l_z}]$ and light space normal $\mathbf{n_l}$ of Todo et al. [2013], modified to be spatially varying. Since our texture space does not depend on the surface normal, we redefine the light space normal $\mathbf{n_l}$ as the light space vector $\mathbf{v_l}$.

The unit direction vector from $\mathbf{p}_l$ to $\mathbf{p}_o$ is denoted by $\mathbf{l}_z$ and its corresponding light space basis vectors are $\mathbf{l}_x$ and $\mathbf{l}_y$. Given $\mathbf{l}_z$, we form an orthonormal basis such that $\mathbf{l}_x$ is orthogonal to $\mathbf{v_{up}}$ (the up direction vector in world space).

The light space vector is defined as $\mathbf{v_l} = (v_{lx}, v_{ly}, v_{lz}) = (\mathbf{v} \cdot \mathbf{l_x}, \mathbf{v} \cdot \mathbf{l_y}, \mathbf{v} \cdot \mathbf{l_z})$. where $\mathbf{v}$ is the direction from the world space position of the surface shading point $\mathbf{p_w}$, to $\mathbf{p_l}$.

The texture space is then defined as $(u, v) = (r \cos\theta, r \sin\theta)$ where $r = \cos^{-1}(v_{lz})$ and $\theta = \tan^{-1}(v_{ly}/v_{lx})$.

4.1 Intensity Distribution

Given our parameters in Section 3.2, we aim to generate these previously described shapes for toon shadow editing. To achieve shape manipulation with our parameters, and smooth blending with existing shade, our shading rig edits take on a shape described by a modified bivariate Gaussian intensity distribution mapped to the local texture space coordinates (u, v).

Anisotropy (a) and Sharpness (s). For one edit, we define an intensity distribution I, given a point $(x, y) \in \mathbb{R}^2$ as follows:

$$I(x, y) = e^{-\alpha x^2 - \frac{1}{\alpha}|y|^{2-s}}, \tag{1}$$

where $s \in [0, 1]$ is the Sharpness parameter, and $\alpha = 1 - a$ given the Anisotropy parameter $a \in [0, 1)$. This gives a smooth intensity distribution with the ability to elongate (using Anisotropy), and sharpen the elongated points (using Sharpness). This formulation allows the sharpness parameter to generate mathematically sharp cusps,

independent of texture resolution or mesh density. This achieves the first three steps in the shape manipulation process shown in Figure 4.

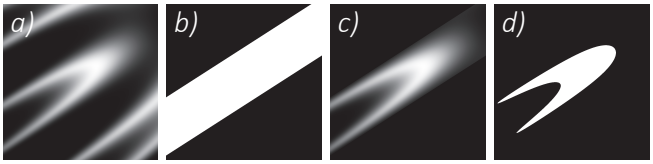


Fig. 7. (a) Initial warped intensity distribution $I(u_w, v_w)$ with periodicity artifacts when using a large Bend value. (b) Mask created by $c(\theta_w)$. (c) Intensity distribution masked by (b) which removes the artifacts. (d) Final shape after thresholding.

Bend (w_y), Bulge (w_x), and Rotation (θ_r). We implement the behaviour shown in the last three steps in Figure 4, using the following formulation. For the desired curvature and asymmetry behaviour, we sample $I(x, y)$ with warped (u, v) coordinates (u_w, v_w). The Bend and Bulge parameters are denoted by w_y and w_x respectively. Defining $\mathbf{w} := (w_x, w_y)^\top$ and $\mathbf{u} := (u, v)^\top$, we first compute a warping function $\theta_w(\mathbf{u})$ as:

$$\theta_w(\mathbf{u}) = k_w \mathbf{w} \cdot \Big(\boldsymbol{R}(\theta_r)\mathbf{u}\Big) \tag{2}$$

where k_w, the degree to which w_x and w_y distorts $I(x, y)$, is set to 10. $\boldsymbol{R}(\theta)$ is the 2D rotation matrix given an angle θ. Given this value of k_w, typical values of w_x range between ±0.5 while w_y would be set between ±0.1.

Given θ_w, and the shape rotation θ_r, (u_w, v_w) is computed as:

$$(u_w, v_w) = \mathbf{w} + \boldsymbol{R}(\theta_w(\mathbf{u}))\Big(\boldsymbol{R}(\theta_r)\mathbf{u} - \mathbf{w}\Big) \tag{3}$$

With this modification to (u, v), Bend will curve $I(x, y)$ upwards ($w_y < 0$) or downwards ($w_y > 0$) along the y-axis. Bulge will warp $I(x, y)$ left ($w_x > 0$) or right ($w_x < 0$) along the x-axis, asymmetrically bulging either side after increasing Anisotropy. The Rotation parameter θ_r rotates the x and y axes before the Bend and Bulge operations take effect.

Our formulation in Equations 2 and 3 roughly describes rotating the coordinates (u, v) about the point (w_x, w_y) at a spatially varying angle $\theta_w(u, v)$. The magnitude of θ_w, and thus the distortion, increases along the u and v axes at the rate of the Bulge (w_x) and Bend (w_y) values respectively, providing the desired curving behaviour.

Each edit's intensity distribution, is then pre-multiplied by several factors to remove repetition artifacts from warping, limit the range of influence, and control the influence of surface details on the edit's shape. Firstly, to mask out repetition artifacts in (u_w, v_w) from periodicity, we use a mask $c(\theta)$ where $c(\theta) = 1$ if $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$ and $c(\theta) = 0$ otherwise. This limits the influence of θ_w outside the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$. The repetition artifacts and mask are shown in Figure 7.

Then, the range of influence of each edit is limited to the user-specified distance R away from $\mathbf{p_l}$, similarly to point lighting. We attenuate the edit's intensity distribution using a distance fall-off curve $f_s(t)$, with a step-edge width of $k_s = 0.05$. This intensity attenuation factor ω is given by $\omega = f_s(\frac{1}{k_s}(R - \|\mathbf{p_w} - \mathbf{p_l}\|))$. The fall-off curve $f_s(t)$ is taken to be a smooth-step function given by the cubic polynomial curve $3t^2 - 2t^3$, when t is between 0 and 1.

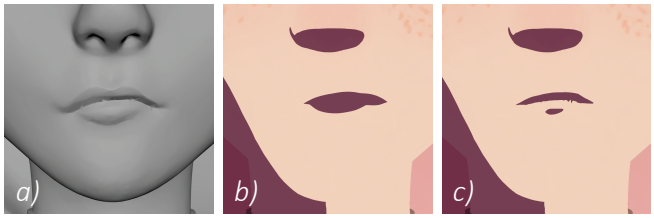


Fig. 8. Given the surface mesh in (a), smoothed normals reduce toon shadow artifacts, but remove important shape details. Edits are placed to indicate the nose mouth in (b). Reducing the Normal Smooth parameter for the mouth edit selectively added detail which the model alone did not express.

Normal Smooth (τ). For edit shapes to be more or less influenced by the surface geometry, we also attenuate each edit's intensity by a weight γ controlled by the Normal Smooth parameter $\tau \in [0, 1]$. We implement surface normal smoothing as a weighted average $\mathbf{N}(\tau)$ between the surface normal $\mathbf{n}$ and smooth sphere-like normals computed from the mesh's centroid $\mathbf{p_c}$ [Barla et al. 2006]. We compute this average as $\mathbf{N}(\tau) = \mathbf{n'}(\tau)/\|\mathbf{n'}(\tau)\|$, where $\mathbf{n'}(\tau) = (1-\tau)\mathbf{n} + \tau\frac{\mathbf{p_w}-\mathbf{p_c}}{\|\mathbf{p_w}-\mathbf{p_c}\|}$.

The weight γ is then given by $\gamma = \mathbf{N}(\tau) \cdot \mathbf{v}$, where $\mathbf{v}$ is the normalised vector from $\mathbf{p}_w$ to $\mathbf{p}_l$. We show the behaviour of local normal smoothing variation in Figure 8. As mentioned, edits made away from the centroid are given a light origin controller. For such edits we substitute $\mathbf{p_c} = \mathbf{p_o}$ so that smoothing occurs about this local region.

Intensity edits. As mentioned in Section 3, our Intensity edits blend with existing toon shading boundaries. To combine Intensity edits with existing toon shading from conventional 3D lighting, we sum this existing reflectance d_0 and the weighted intensity distributions of all N_I Intensity edits in the shading rig. A global toon shading threshold T_0 is then applied. T_0 is provided by the existing toon shader [Lake et al. 2000] which the shading rig is applied to. The lit region on a surface, which we denote as $\mathbf{B}_1$, is the set of all points that satisfy the following condition:

$$T_0 < d_0 + \sum_{i=1}^{N_I} G_i c(\theta_{w_i}) \omega_i \gamma_i I_i(u_{w_i}, v_{w_i}) \tag{4}$$

Here, $I_i(x, y)$ is the intensity of the i^{th} Intensity edit placed in the scene. As written, $I_i(x, y)$ is also weighted by a corresponding, user-specified, intensity gain parameter $G_i \in \mathbb{R}$. The magnitude of G_i controls the strength relative to other shading terms, while its sign affects whether light ($G > 0$) or shadow ($G < 0$) is created. As mentioned, this is analogous to controlling the intensity of conventional 3D lighting while allowing negative values for shadows. In Figure 9 we show how our intensity edits can be smoothly combined for toon shadow editing.

Mask edits. *Mask* edits instead preserve sharp joins when combined with other shading, and are unaffected by light changes unless animated with the key light. For mask edits, each edit creates their

Fig. 9. In (a), conventional toon shadow is the red shape lacking cheek and nose definition. Our edit shapes, shown in blue, are placed to depict more detail. In (b) we see the shadow shapes (outlined) are smoothly combined into the desired shading. The final shading is shown in (c).

own lit region by thresholding their own intensity fields. Thus the lit region of the i^{th} Mask edit, which we denote as $\mathbf{M}_i$, is the set of points where $t_m < c(\theta_{w_i})\omega_i\gamma_i I_i(u_{w_i}, v_{w_i})$. The threshold t_m is set to 0.8 for all mask edits. The lit region of all mask edits combined with all intensity edits is simply: $\mathbf{B}_1 \cup (\bigcup_{i=1}^{N_M} \mathbf{M}_i)$, where N_M is the number of Mask edits used in the shading rig.

Any mask edit can be specified to produce a shadow instead of light. For all mask edits used for *shadow*, the modified region is instead given by $\mathbf{B}_1 \cap (\bigcup_{i=1}^{N_M} \overline{\mathbf{M}_i})$, where $\overline{\mathbf{M}_i}$ is the complement of the set $\mathbf{M}_i$.

When using multiple light origins, each edit evaluates their own intensity distributions using the p_o of their assigned light origins. We then combine the edits using the same equations.

Softness (d). The Softness parameter, denoted by d, of an edit will smooth its shape's boundary. A smooth boundary of width d is achieved using the same smooth step function $f_s(t)$, mentioned previously. For the i^{th} mask edit, the smoothed edit shape mask is given by $f_s(\frac{1}{d_i}(c(\theta_{w_i})\omega_i\gamma_i I_i(u_{w_i}, v_{w_i}) - t_m))$.

Deformation Tracking. The deformation tracking behaviour described in Section 3.3 is achieved by evaluating our edits in undeformed object space. For edits that are chosen to track deformation, we substitute the world space position $\mathbf{p}_w$ and normal $\mathbf{n}$, with the undeformed object space position and normals, when computing the texture space and intensity distributions.

4.2 Shading Rig Animation

A shading rig is built from multiple edits based on these shapes. With this rig being fully parametric, artists can freely animate the motion and shape variation using keyframing. One can set keyframes of edit positions and shape parameters on the timeline for a traditional animation sequence. Alternatively, for the case of varying lighting, we develop a system for keyframing with respect to the key light direction. Artists can then animate the shading rig to dynamically adjust conventional toon shading under a changing light direction. We refer to this as *pre-animating* the shading rig.

At a given light direction, artists adjust the shading rig for their desired result, and create a keyframe. Note that these keyframes are set for each light direction, not on a timeline in the traditional sense. Keyframes hold the state of the shading rig, including the 3D position of each edit and their parameter values, at a given light direction. Since our interpolation domain is the key light direction vector, instead of a scalar time value, we use spherical radial basis functions (RBFs) [Tsai and Shih 2006]. This preserves the geodesic distance between light direction vectors when interpolating between keyframes. Using the spherical Gaussian as the RBF kernel gives smooth interpolation between keyframes. The spherical Gaussian function that we use is given by:

$$\exp\left(\lambda(\boldsymbol{\eta} \cdot \boldsymbol{\xi} - 1)\right) \tag{5}$$

where $\boldsymbol{\eta}$ is the unit vector in the direction being sampled, $\boldsymbol{\xi}$ is the centre axis, and λ determines the kernel width. The value of $\boldsymbol{\xi}$ for each spherical Gaussian is the key light direction in which a keyframe was set. We set $\lambda = 1$ by default, though this can be changed by the artist to control the interpolation result.

Each edit requires at least two keyframes for interpolation between key light directions. By default, the RBF interpolated function will extrapolate how the shading rig behaves when the key light rotates away from these light directions. Artists can add more keyframes to control the result, or modify the RBF kernel width λ. As in general scattered data interpolation using RBFs, large λ values provide smoother interpolation between distant keyframes, while small λ values will prevent overshoot between close keyframes.

When pre-animating the shading rig, the same number of edits are used for all key light directions. Edits that appear and disappear must still exist in the scene. As mentioned, this effect is controlled by the distance-based intensity falloff and shape scaling, or with intensity gain for intensity edits.

Art-directed Shading for Dynamic Objects. Rather than the light direction changing, an object may rotate relative to a key light direction. In terms of shading, this case is equivalent to the light rotating in the opposite angle relative to the object. To reproduce the correct shading at this orientation, we can simply evaluate shading rig animation at this opposite angle, given by the current key light direction and the orientation of the object. We then update the shading rig at this light vector but apply the shading in object space, like we do for deformation tracking, ensuring that the edits which do not change with the key light remain static.

Based on these animation and interpolation behaviours, we show an example pre-animated shading rig in the next section.

5 RESULTS

In this section we compare continuous shadow animation, and dependence on mesh density, with Todo et al. [2007], and shape interpolation performance with Nader and Guennebaud [2018]. We also show qualitative examples of how our shading rig model supports the use-cases introduced below.

5.1 Examples and Use-Cases

We first show how our shading rig approach can achieve common use cases for shade editing using our parametric edits.

Facial Expression and Detail. In hand-drawn shading, facial expression and detail are exaggerated with stylised shadows. In Figure 10 we show examples of hand-drawn shading for facial expression and detail, as well as how our edits can serve this purpose. In this example, we use rounded shadow edits to darken the eyes as the "deep

Fig. 10. Top: Example hand-drawn shadows used for facial expression and detail. Bottom: unedited toon shading (left) and edited expression using shading edits (right). This result required 5 edits. Creature concept: ©Wildboy Studios used with permission. Female artwork by Claudio Grassi licensed under CC BY. Head model by Vinicius Nunes licensed under CC BY.

pools of shadow" [Hogarth 1991] mentioned in Section 3. Curved and sharpened edits exaggerate brow wrinkles and cheekbones such as shown in the hand-drawn examples.

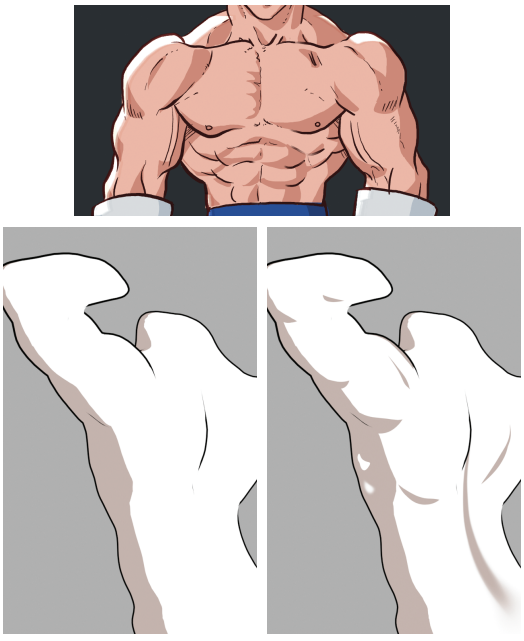


Fig. 11. Top: hand-drawn stylised muscle shading. Bottom: unedited toon shading (left) and added muscle shading using 10 shading edits (right). Both results use a black contour shader. Artwork by Brain Graft licensed under CC BY. Model by Julien Kaspar licensed under CC BY.

Muscle Shading. We show how muscle lines can be introduced with the shading rig. Shadows added to emphasise crease details such as muscles are referred to as "inner lines" [Guertault et al. 2018; Motomura 2015]. Figure 11 shows hand-drawn muscle shading, and how our shading edits can model these shadows as parametric inner lines. This use-case of cartoon muscle lines, typically solved using textures or mesh editing, can now achieve vector quality while remaining animatable with our approach.

Fig. 12. Left: Hand-drawn stylised shading with Rembrandt triangle under the eye. Middle: Edited toon shading with the shading rig, applying a Rembrandt triangle without a nose shadow being cast. A single intensity edit is used to blend with existing key lighting. Right: rounded Rembrandt triangle added also using one mask edit with high Normal Smoothing. Character artwork: ©Rodu used with permission.

Rembrandt Triangle. The Rembrandt triangle [Sint 2009; Wright 2015] is a realistic lighting phenomenon, ubiquitous in both stylised shading and cinematic lighting. It occurs when an elongated nose shadow meets the cheek shadow creating a triangular lit area beneath the eye. Although our edit parameters were not specifically designed for triangles, we show an example of a similar shape, in Figure 12. This required low Anisotropy, high Sharpness, increased Bulge to weigh one side, and increased Bend to curve up around the eye. The edits were added directly, even though the unedited result did not produce a nose shadow.

Local Detail Control. We show that even on high detail surface meshes, our edits can control shadows by varying the Normal Smoothing parameter on each edit. In the example in Figure 13, surface details on the snake are revealed only on the tail. The face is left abstract to only show face details like the eyes and mouth. We achieve this with different edits having their own Normal Smooth value. The underside of the snake also employs local softening as defined by the artist.

Fig. 13. Left: high detail mesh. Middle: base toon shading. Right: edited result (9 edits). Model by Julien Kaspar licensed under CC BY. Best viewed in the electronic copy.

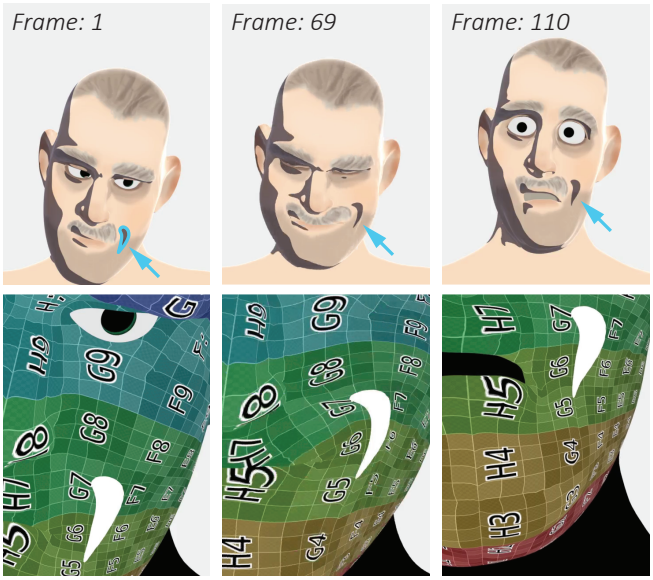


Fig. 14. Top: Edit made on the left-most frame (d in blue) can track mesh deformation throughout an animation sequence. Bottom: edit shown in white precisely tracks mesh surface parametrisation and remains animatable and mesh/texture resolution independent. Model by Daniel Martinez Lara licensed under CC BY.

Deformation Support. We also show our method supports animated and deforming characters, using our deformation tracking function. Artists can design edits at a single undeformed pose, and have their edits automatically track surface deformations. In Figure 14 an edit is placed near the left corner of the mouth. Without any further manual animation of this edit, it tracks the surface mesh during deformation. We also visualise the edit with the surface's UV parametrisation showing precise correspondence. This matches the behavior of other techniques, while remaining animatable at vector quality.

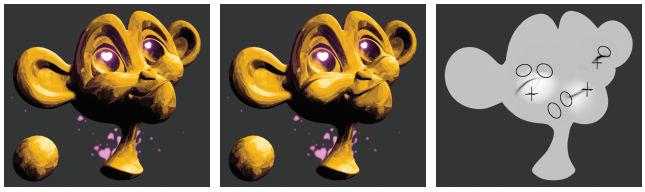


Fig. 15. Left: Lit-sphere shading [Todo et al. 2013]. Middle: Edited shading using shading rig (5 edits). Right: Shading Rig (circle and axis controllers) and its resulting intensity field. Model by Julien Kaspar licensed under CC BY. Best viewed in the electronic copy.

Other styles. Our shading rig approach can be used outside two-tone toon shading. In Figure 15 we show how our Intensity Edits are suitable for editing artist-defined shading gradation such as in lit-sphere shading [Sloan et al. 2001; Todo et al. 2013]. We achieve this by locally scaling the lit-sphere texture of [Todo et al. 2013] based on the edited intensity field generated from the shading rig.

5.2 Shadow Animation with Lighting Changes

After pre-animating, the shading rig smoothly transitions between the keyframes as the key light is rotated. This has the effect of shading edits moving in space, and morphing in shape, to dynamically correct the toon shading result as desired by the artist. We show an example result in Figure 16. For comparison, we show the unedited toon shading result and the normal smoothed result on which the shading rig applied edits.

This example was animated with four dynamic edits, each using 2-4 positional keyframes and 1-3 parameter keyframes. To make edits smoothly appear and disappear, their positions were moved further and closer to the surface mesh. This gives simple control over the size of the edit's shape just like conventional point lights with a distance based intensity falloff.

Note the intentionally added stylised Rembrandt triangle which the unedited result did not produce. This edit was added directly, regardless of the geometry and conventional lighting. This edit's shape was also animated to emerge rounded, and then sharpened at the right light angles, using the Sharpness and Anisotropy parameters. From this we can see the level of control artists have over smooth stylised shadow animation, using the shading rig.

We note that this level of control for dynamic shadow design is crucial to avoid undesirable shadows from realistic lighting. In this case especially, long nose shadows are required to produce Rembrandt lighting in reality [Wright 2015], but are considered undesirable [Malkiewicz and Mullen 2009] before the Rembrandt triangle is made. With the shading rig, we can smoothly animate the triangle emerging while keeping the nose shadow short.

The edit at the forehead is also used to exaggerate the browline by sharply pinching the shading boundary at the brow. This is animated from left to right with the key light, to maintain this contour shape while the key light is rotated. Just three keyframes were need for this highlight using the shading rig.

In Figure 17 we show how this animation can be applied to a dynamic character moving relative to a key light. This example, implemented in Unreal Engine 4, shows the artist-defined shadow behaviour being reproduced as the character turns her head. This result uses the method in Section 4.2. Please see the supplementary video for the full animation.

5.3 Performance Evaluation

After implementing our method as a GPU shader, we observed real-time rendering and animation of more complex shading rigs (tens of edits) in both Maya and Blender. We observed real-time performance for thousands of edits (more than necessary for practical use), in a stylised game environment, in Unreal Engine 4. The following results show the performance cost of each stage in our approach.

For the RBF-based animation system, we measure the worst-case performance costs for keyframing and real-time evaluation, as the number of keyframes increase. In Figure 18, we measure the RBF solve time as more keyframes are added for all parameters of a single edit. We see that adding a keyframe is fast for a single edit, even up to 200 keyframes. Figure 19 shows the RBF evaluation time for all parameters in a shading rig with 10 and 100 edits. Figure 20 shows GPU rendering time as the number of edits increase. In

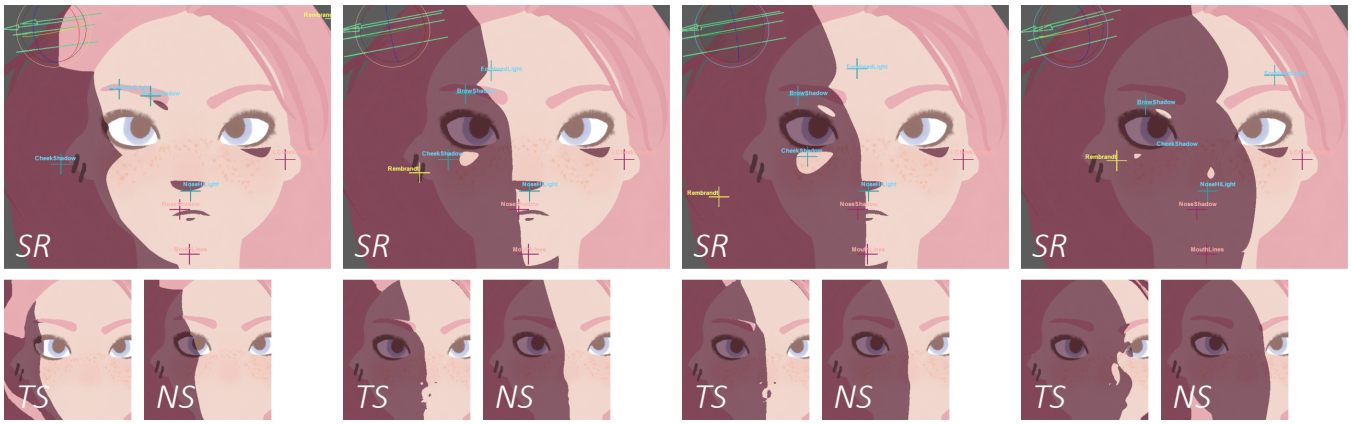


Fig. 16. Top row (SR): Shadow animation produced by a pre-animated shading rig (collection of cross-shaped locators) as the key light is rotated. Bottom row, for each frame shown in the top row: Conventional toon shading (TS) and normal smoothed toon shading (NS) to be edited.

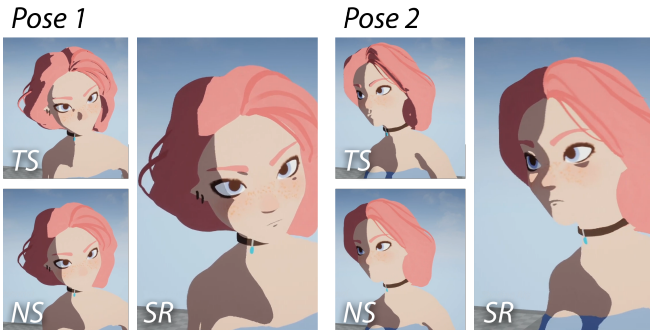


Fig. 17. Shading rig animation result, applied to a dynamic character moving relative to key lighting. TS: toon shading, NS: normal smoothing, SR: shading rig.

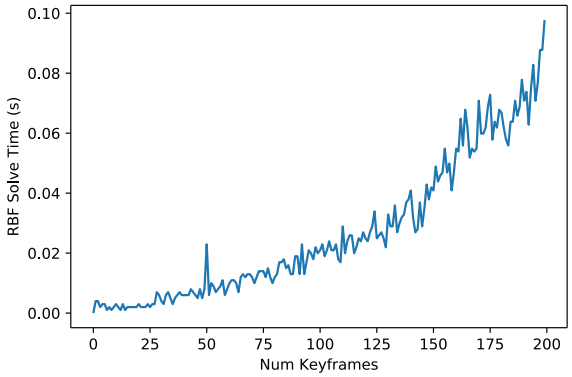


Fig. 18. Time taken to keyframe all parameters for one edit, as its number of keyframes increases, measured by the RBF solve time.

Unreal Engine 4, we render the GPU shader for different screen percentages of the total screen resolution of 1920×1080. The screen percentage is how much area a stylised character takes up of the viewport.

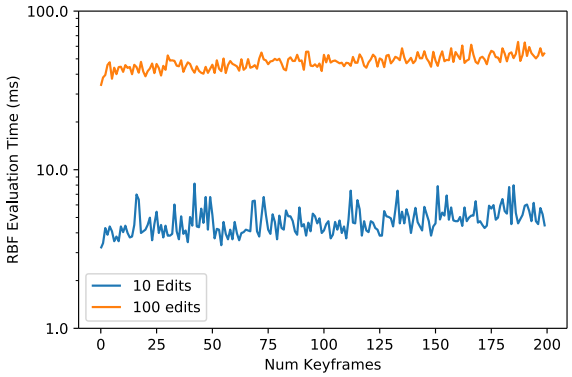


Fig. 19. RBF evaluation time (log scale) for shading rigs with 10 and 100 edits, as the number of keyframes per edit increases (all parameters keyframed).

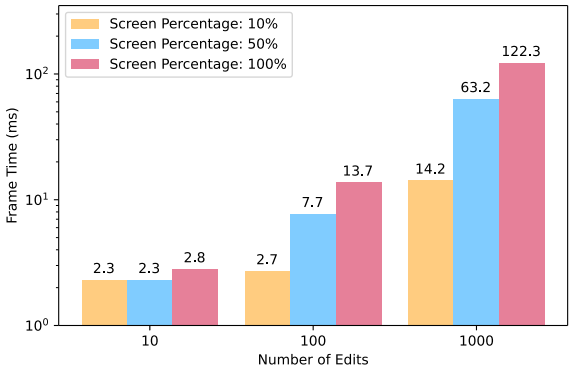


Fig. 20. GPU shading frame time (log scale) as the number of edits increases, at different screen percentages of 1920×1080 resolution.

The timings were measured using an Intel Xeon W-2123 3.60 GHz CPU, on 16 GB of RAM, and a GeForce GTX 1080 GPU.

The real-time evaluation result shows that our shading rig achieves real-time performance. While we expect that artists will only use tens of keyframes per edit, and tens of dynamic edits per character, testing at these larger ranges still show real-time rendering with multiple characters can be achieved.

Not all edits and parameters need to be animated, so these performance results show the worst-case performance cost in our unoptimised implementation. In practice, shading rigs for off-screen and distant characters need not be updated. In real-time games and previsualisation, the shading rig can be updated on a CPU thread independent of the GPU rendering framerate.

5.4 Comparison With Previous Work

(a) [Todo et al. 2007] (b) Our approach

Fig. 21. Example animated shade edit comparison with colour variation from red (start) to blue (end). (a) Offset painting [Todo et al. 2007] result with 4 keyframes (numbered), showing fragmented motion of shadow. (b) our approach requires only 2 keyframes to yield continuous motion and shape manipulation.

Smooth Animation. We expect shape interpolation between two keyframes to achieve continuous motion and shape transformation, without gaps or doubling, as described by Bonneel et al. [2011]. While offset painting [Todo et al. 2007] supports arbitrary shade modification, edits cannot be moved across the surface without redrawing the shape. Our method requires no redrawing of designed shapes and are simply moved across the surface mesh like a light source. Furthermore, we achieve continuous shade animation with limited keyframes, as shown in Figure 21. Here we see the edits of Todo et al. [2007] disappear and reappear between keyframes, due to linear blending between the fixed intensity distributions of their edits. Our approach of parametrising each edit achieves true distribution interpolation, preserving continuous motion and shape interpolation between keyframes. This is more clearly illustrated in Figure 3 of the supplemental material.

Rasterisation Artifact Comparison. Bonneel et al. [2011] proposed to use optimal mass transport for continuous distribution interpolation that specifically solves the problem shown in Figure 21 and Figure 3 of the supplemental material. The method of Nader and Guennebaud [2018] achieves this in real-time on an image grid. Although suitable for fast texture mapping on GPUs, this would exhibit artifacts upon close-up. In Figure 22, we compare these artifacts and show that our approach provides artifact free reproduction of sharp edits and shape boundary. The boundary of our edit shapes can be antialiased using screen-space derivatives as done by Loop and Blinn [2005].

While using grid-free mass transport techniques would avoid pixelation artifacts, other sampling issues would arise. The grid-free method of Bonneel et al. [2011] use RBFs to approximate shapes, but this cannot reproduce sharp cusps as mentioned in the related work. Representing distributions as weighted point clouds using weighted Dirac delta functions [Peyré et al. 2019] would produce aliasing artifacts as demonstrated by Bonneel et al. [2011].

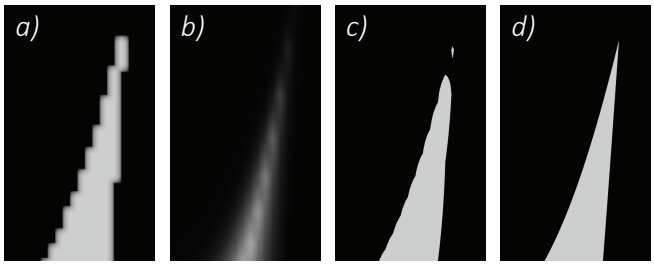
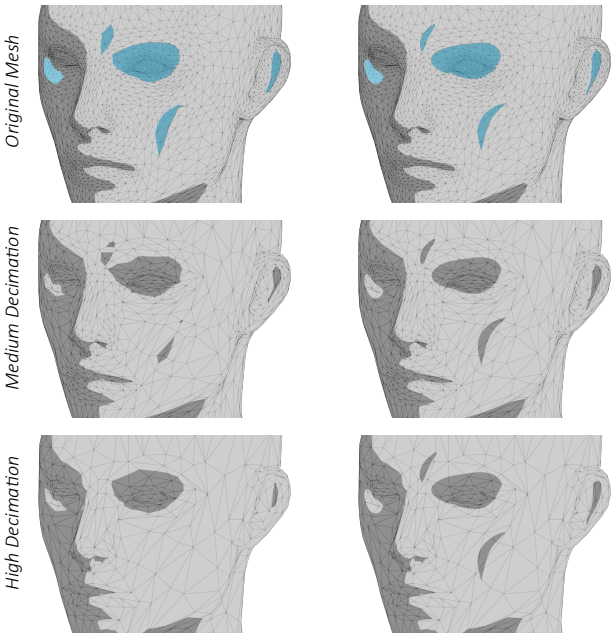


Fig. 22. Texture mapping using a rasterised shading texture exhibits pixelation artifacts upon magnification (a). Using a rasterised intensity distribution (b) for optimal transport, [Nader and Guennebaud 2018] causes artifacts after thresholding (c). Our parametric distribution (d) provides artifact free reproduction of sharp cusps and shape boundary.

Performance Comparison With Image Interpolation. While the above artifacts can be reduced with higher texture resolution, we show that our 2D shape interpolation has a much lower computational cost than that of Nader and Guennebaud [2018]. As mentioned in Section 2, the precomputation time demanded by their method would become prohibitive for editing with high resolution images. Our method requires no precomputation, allowing for artists to animate shading with instant feedback.We also compare the performance cost of animation provided by each method at runtime.

The 2D shape interpolation of our shading rig model and the method of Nader and Guennebaud [2018] can both be parallelised on the GPU. To compare the total number of operations we measure running times using single-threaded, non-parallelised, CPU implementations of each method. We used an Intel Xeon W-2123 3.60 GHz, with 16 GB of RAM. Both methods interpolate parameters representing the output 2D shape, with negligible running time. However, we find that the rasterisation step of Nader and Guennebaud [2018] is much slower than ours. Figure 2 in the supplemental material shows the running times. We can see that our shape interpolation is an order of magnitude faster. Thus, artists can use more dynamic shading edits with our approach than with mass transport, for interactive applications. We note that shape interpolation based on mass transport still permits arbitrarily shaped shading edits.

Mesh Density Independence. In Figure 23, we show that edited shading shapes degrade after mesh decimation, when relying on vertex normal edits, or offsets painted as vertex colours [Todo et al. 2007]. Our parametric edits are seen to preserve their shape regardless of mesh density and topology, providing greater freedom of expression to artists. In particular the shadow shapes permitted by mesh-based edits are severely limited by mesh topology.



(a) [Todo et al. 2007] (b) Our approach

Fig. 23. Original shade edits made on a surface mesh (blue) and their dependence on mesh density. a) Limitation of Todo et al. [2007] where edit shapes are affected by mesh density and topology. b) Our parametric edits preserve their shape regardless of mesh resolution and topology.

6 LIMITATIONS AND FUTURE WORK

We addressed the problem of dynamic stylised shading edits by parameterising individual edits. Our novel parametric model produce local stylised shade editing shapes. This allows for smooth animation while maintaining vector quality and resolving sharp cusps upon close-ups. Based on artistic shading literature, our edits are designed to produce expressive shapes to support common editing practices and adding intentional details.

We have shown smoother animation results compared to the method of Todo et al. [2007], with fewer keyframes. We also demonstrated that our 2D shape and intensity distribution interpolation incurs dramatically lower computational load compared to real-time optimal transport [Nader and Guennebaud 2018].

Artists can build a shading rig to edit the result of conventional toon shading. Our example pre-animated shading rig achieves real-time art-directed toon shading without artist intervention.

Limitations. Our parametric edit model can produce simple, round shapes with optional sharp points, suitable for characters with organic forms. However, for mechanical objects such as robots, weapons, and cars, our parameters may have limited support for producing more complex edits for these geometric surfaces. Similarly, our model may also have limited support for producing fine-scale hair details and shade editing on complex large-scale background environments.

Our approach also demands a time investment to pre-animate the shading rig with all lighting changes. Animated shading rig results produced in this paper took up to several hours to create in our prototype. We note, however, that 3D character preparation using existing editing techniques for stylised games, has been reported to take months [Motomura 2015]. Considering this, and our improved keyframing results, our approach will reduce preparation time while enabling smooth shade animation at vector quality.

Future Work. Compared to our approach, vector mask manipulation from rotoscopy remains the most flexible in designing arbitrary shadow animation. Further research into rendering real-time, animatable, vector masks on 3D surfaces would be valuable. While interfaces for 3D lighting tasks have been evaluated [Kerr and Pellacini 2009], further interface development with artists, followed by user studies could help to assess what user interfaces are best suited to stylised shade editing.

ACKNOWLEDGMENTS

We would like to give special thanks to Ayumi Kimura, Anandaroop Mukherjee, Joaquim Jorge, and CMIC researchers and staff. Models and artwork sourced from Sketchfab and Blender Cloud. This project was supported by the Entrepreneur University Programme, funded by Tertiary Education Commission (TEC), New Zealand.

REFERENCES

Ken Anjyo and Katsuaki Hiramitsu. 2003. Stylized highlights for cartoon rendering and animation. *IEEE Computer Graphics and Applications* 23, 4 (2003), 54–61.

Muhammad Arief, Kunio Kondo, Koji Mikami, Hideki Todo, and Yasushi Yamaguchi. 2015. Controllable Region via Texture Projection for Stylized Shading. In *Proceedings of the 14th ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry* (Kobe, Japan) *(VRCAI '15)*. ACM, New York, NY, USA, 35–38. https://doi.org/10.1145/2817675.2817688

Pascal Barla, Joëlle Thollot, and Lee Markosian. 2006. X-toon: an extended toon shader. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*. ACM, 127–132.

Pierre Bénard, Forrester Cole, Michael Kass, Igor Mordatch, James Hegarty, Martin Sebastian Senn, Kurt Fleischer, Davide Pesare, and Katherine Breeden. 2013. Stylizing animation by example. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 119.

Pierre Bénard, Aaron Hertzmann, and Michael Kass. 2014. Computing Smooth Surface Contours with Accurate Topology. *ACM Trans. Graph.* 33, 2 (2014). https://doi.org/10.1145/2558307

Jeremy Birn. 2000. *Digital lighting & rendering*. New Riders Press.

Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. 2011. Displacement interpolation using Lagrangian mass transport. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 158.

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. 2018. The design and evolution of disney's hiperion renderer. *ACM Transactions on Graphics* 37, 3 (2018), 33.

Mark Dokter, Jozef Hladky, Mathias Parger, Dieter Schmalstieg, Hans-Peter Seidel, and Markus Steinberger. 2019. Hierarchical Rasterization of Curved Primitives for Vector Graphics Rendering on the GPU. *Computer Graphics Forum* 38, 2 (2019), 93–103. https://doi.org/10.1111/cgf.13622 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13622

Elmar Eisemann, Holger Winnemöller, John C Hart, and David Salesin. 2008. Stylized vector art from 3D models with region support. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1199–1207.

Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sỳkora. 2016. StyLit: illumination-guided example-based stylization of 3D renderings. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 92.

Julien Guertault, Élie Setbon, and Pavel Martishevsky. 2018. Manga Stylized Rendering in VR. In *SIGGRAPH Asia 2018 Courses* (Tokyo, Japan) *(SA '18)*. ACM, New York, NY, USA, Article 13, 61 pages. https://doi.org/10.1145/3277644.3277768

Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. 2001. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 327–340.

Burne Hogarth. 1991. *Dynamic Light and Shade*. ERIC.

Matis Hudon, Mairéad Grogan, Rafael Pagés, Jan Ondřej, and Aljoša Smolić. 2019. 2DToonShade: A stroke based toon shading system. *Computers & Graphics: X* 1 (2019), 100003.

Ondřej Jamriška, Šárka Sochorová, Ondřej Texler, Michal Lukáč, Jakub Fišer, Jingwan Lu, Eli Shechtman, and Daniel Sỳkora. 2019. Stylizing video by example. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 107.

Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. 2002. WYSIWYG NPR: Drawing Strokes Directly on 3D Models. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 21, 3 (July 2002), 755–762.

Yosuke Katsura and Ken Anjyo. 2007. Anime perspective. In *ACM SIGGRAPH 2007 sketches*. ACM, 75.

William B. Kerr and Fabio Pellacini. 2009. Toward Evaluating Lighting Design Interface Paradigms for Novice Users. *ACM Trans. Graph.* 28, 3, Article 26 (July 2009), 9 pages. https://doi.org/10.1145/1531326.1531332

Mark J Kilgard and Jeff Bolz. 2012. GPU-accelerated path rendering. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–10.

Adam Lake, Carl Marshall, Mark Harris, and Marc Blackstein. 2000. Stylized rendering techniques for scalable real-time 3d animation. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*. ACM, 13–20.

David Landau. 2014. *Lighting for cinematography: a practical guide to the art and craft of lighting for the moving image.* A&C Black.

Henrik Lieng, Flora P. Tasse, Jiří Kosinka, and Neil A. Dodgson. 2015. Shading Curves: Vector-Based Drawing With Explicit Gradient Control. *Computer Graphics Forum* (2015).

Charles Loop and Jim Blinn. 2005. Resolution independent curve rendering using programmable graphics hardware. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 1000–1009.

Kris Malkiewicz and M David Mullen. 2009. *Cinematography*. Simon and Schuster.

O. Mattausch, T. Igarashi, and M. Wimmer. 2013. Freeform Shadow Boundary Editing. *Computer Graphics Forum* 32, 2pt2 (2013), 175–184. https://doi.org/10.1111/cgf.12037 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12037

Scott McCloud. 1993. *Understanding comics: The invisible art.*

Junya Christopher Motomura. 2015. *GuiltyGearXrd's Art Style : The X Factor Between 2D and 3D.* Arc System Works. www.gdcvault.com/play/1022031/GuiltyGearXrd-s-Art-Style-The

Georges Nader and Gael Guennebaud. 2018. Instant transport maps on 2D grids. *ACM Transactions on Graphics* 37, 6 (2018), 13.

M. Okabe, Y. Matsushita, L. Shen, and T. Igarashi. 2007. Illumination Brush: Interactive Design of All-Frequency Lighting. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. 171–180. https://doi.org/10.1109/PG.2007.9

Romain Pacanowski, Xavier Granier, Christophe Schlick, and Pierre Poulin. 2008. Sketch and Paint-Based Interface for Highlight Modeling. In *Proceedings of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling* (Annecy, France) *(SBM'08)*. Eurographics Association, Goslar, DEU, 17–23.

Fabio Pellacini, Parag Tole, and Donald P Greenberg. 2002. A user interface for interactive cinematic shadow design. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 563–566.

Gabriel Peyré, Marco Cuturi, et al. 2019. Computational Optimal Transport: With Applications to Data Science. *Foundations and Trends® in Machine Learning* 11, 5-6 (2019), 355–607.

Nicolas Ray, Xavier Cavin, and Bruno Lévy. 2005. Vector texture maps on the GPU. *Inst. ALICE (Algorithms, Comput., Geometry Image Dept. INRIA Nancy Grand-Est/Loria), Tech. Rep. ALICE-TR-05-003* (2005).

Tobias Ritschel, Thorsten Thormählen, Carsten Dachsbacher, Jan Kautz, and Hans-Peter Seidel. 2010. Interactive On-Surface Signal Deformation. *ACM Trans. Graph.* 29, 4, Article 36 (July 2010), 8 pages. https://doi.org/10.1145/1778765.1778773

Thorsten-Walther Schmidt, Fabio Pellacini, Derek Nowrouzezahrai, Wojciech Jarosz, and Carsten Dachsbacher. 2016. State of the art in artistic editing of appearance, lighting and material. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 216–233.

Dario Seyb, Alec Jacobson, Derek Nowrouzezahrai, and Wojciech Jarosz. 2019. Nonlinear Sphere Tracing for Rendering Deformed Signed Distance Fields. *ACM Trans. Graph.* 38, 6, Article 229 (Nov. 2019), 12 pages. https://doi.org/10.1145/3355089.3356502

Zhixin Shu, Sunil Hadap, Eli Shechtman, Kalyan Sunkavalli, Sylvain Paris, and Dimitris Samaras. 2018. Portrait lighting transfer using a mass transport approach. *ACM Transactions on Graphics (TOG)* 37, 1 (2018), 2.

Steve Sint. 2009. *Digital Portrait Photography: Art, Business and Style.* Sterling Publishing Company, Inc.

Peter-Pike J Sloan, William Martin, Amy Gooch, and Bruce Gooch. 2001. The lit sphere: A model for capturing NPR shading from art. In *Graphics interface*, Vol. 2001. 143–150.

Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. 2015. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 66.

Xin Sun, Guofu Xie, Yue Dong, Stephen Lin, Weiwei Xu, Wencheng Wang, Xin Tong, and Baining Guo. 2012. Diffusion curve textures for resolution independent texture mapping. *ACM Trans. Graph.* 31, 4 (2012), 74–1.

Daniel Sýkora, Ondřej Jamriška, Ondřej Texler, Jakub Fišer, Michal Lukáč, Jingwan Lu, and Eli Shechtman. 2019. StyleBlit: Fast Example-Based Stylization with Local Guidance. *Computer Graphics Forum* 38, 2 (2019), 83–91.

Mumehiro Tada, Yoshinori Dobashi, and Tsuyoshi Yamamoto. 2012. Feature-based interpolation for the interactive editing of shading effects. In *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*. 47–50.

Hideki Todo, Ken Anjyo, William Baxter, and Takeo Igarashi. 2007. Locally Controllable Stylized Shading. *ACM Trans. Graph.* 26, 3, Article 17 (July 2007). https://doi.org/10.1145/1276377.1276399

Hideki Todo, Ken Anjyo, and Shun'Ichi Yokoyama. 2013. Lit-Sphere Extension for Artistic Rendering. *Vis. Comput.* 29, 6-8 (June 2013), 473–480. https://doi.org/10.1007/s00371-013-0811-7

Chad Troftgruben. 2014. *Learning Anime Studio.* Packt Publishing.

Yu-Ting Tsai and Zen-Chung Shih. 2006. All-Frequency Precomputed Radiance Transfer Using Spherical Radial Basis Functions and Clustered Tensor Approximation. *ACM Trans. Graph.* 25, 3 (July 2006), 967–976. https://doi.org/10.1145/1141911.1141981

Greg Turk and James F. O'Brien. 2005. Shape Transformation Using Variational Implicit Functions. In *ACM SIGGRAPH 2005 Courses* (Los Angeles, California) *(SIGGRAPH '05)*. ACM, New York, NY, USA, Article 13. https://doi.org/10.1145/1198555.1198639

David Vanderhaeghe, Romain Vergne, Pascal Barla, and William Baxter. 2011. Dynamic Stylized Shading Primitives. In *NPAR '11:Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*. Vancouver, Canada, 99–104. https://doi.org/10.1145/2024676.2024693 Honorable Mention in Rendering.

Lvdi Wang, Kun Zhou, Yizhou Yu, and Baining Guo. 2010. Vector solid textures. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 86.

Daniel N Wood, Adam Finkelstein, John F Hughes, Craig E Thayer, and David H Salesin. 1997. Multiperspective panoramas for cel animation. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 243–250.

Pete Wright. 2015. *Cinematic Portraits: How to Create Classic Hollywood Photography.* Amherst Media.