Interactive Reflection Editing

Tobias Ritschel Makoto Okabe Thorsten Thormählen Hans-Peter Seidel

MPI Informatik[*]

Abstract

Effective digital content creation tools must be both efficient in the interactions they provide but also allow full user control. There may be occasions, when art direction requires changes that contradict physical laws. In particular, it is known that physical correctness of reflections for the human observer is hard to assess. For many centuries, traditional artists have exploited this fact to depict reflections that lie outside the realm of physical possibility. However, a system that gives explicit control of this effect to digital artists has not yet been described. This paper introduces a system that transforms physically correct reflections into art-directed reflections, as specified by *reflection constraints*. The system introduces a taxonomy of reflection editing operations, using an intuitive user interface, that works directly on the reflecting surfaces with real-time visual feedback using a GPU. A user study shows how such a system can allow users to quickly manipulate reflections according to an art direction task.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

Keywords: post-production, intuitive editing, lighting design, perception, non-photorealistc rendering, graphics hardware

1 Introduction

The composition of traditional, as well as computer-generated images can be understood as an optimization process, wherein a number of artistic goals should be fulfilled within a number of physical constraints. When all physical constraints are met, photorealistic images can be expected, but some artistic goals might not be achieved. On the other hand, fulfilling all artistic goals might lead to images with inadequate realism. For many centuries the limitations of human perception have allowed skilled artists to simultaneously achieve both goals.

One famous example from traditional art is the painting "The Rokeby Venus" by Diego Velázquez, shown in Fig. 1: The reflection of the face in the mirror is physically incorrect, as was proven by photographic reproduction [Braham 1976]. Despite this, a pleasant and naturalistic image was achieved, as human observers have difficulties in assessing the physical correctness of reflections on complex surfaces [Fleming et al. 2003; Khan et al. 2006; Ramanarayanan et al. 2007].

[*]This work was partially funded by the MPC-VCC (BMBF-FKZ01IMC01).

Although human difficulties with the assessment of reflections are known in the field of computer graphics, very little research has been done on how to define and manipulate reflections outside physical bounds. Non-physical reflections are difficult to achieve with today's tools. Manual techniques, like 2D image warping, compositing approaches, reflection map editing, or normal map editing, are tedious to apply and fail to properly capture perspective, complex geometry, animation sequences, local edits, partially diffuse materials, or occlusions.

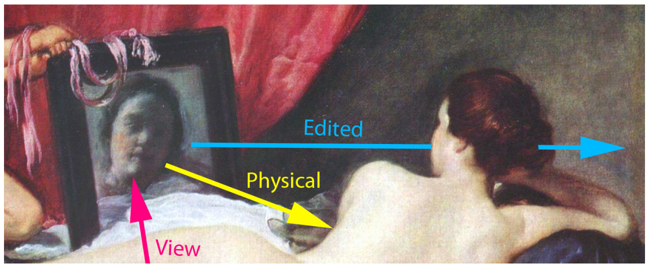

Figure 1: *"The Rokeby Venus" (before 1651; detail) by Diego Velázquez (1599–1660) uses reflections beyond physical laws.*

In this paper, a system for interactive reflection editing is presented, which allows the user to escape the boundaries of physically correct rendering, without any such limitations. The system takes as input a virtual 3D scene containing reflecting objects, together with a number of user-defined constraints, which define the edited reflection directions (e.g., Velázquez used one such constraint in his painting: "The mirror should reflect the face, not the body"). The output of our system is an image in which all constraints are fulfilled and the change in reflection direction is smooth. The system estimates the best (least squares) interpolated edited reflection direction for each pixel. The system runs in real-time on a GPU in HD resolution, is independent of the underlying reflection rendering method (ray-tracing or reflection mapping), is independent of meshing or parameterization, and works without pre-processing. It can handle animated meshes and allows more general specular light transport, like glossy reflections and refractions. Furthermore, the system does not require any user parameters besides the desired constraints. Our user study shows that our system allows not only professional designers but also novices to design complex reflections in minutes. With today's tools, this is difficult for even an experienced designer to achieve.

2 Previous Work

Appearance editing Several methods have been proposed to manipulate lighting effects in a rendered 3D scene, as an alternative to manually setting raw lighting parameters. Most of them use sketching or painting interfaces that allow users to directly draw lighting effects [Schoeneman et al. 1993; Poulin et al. 1997; Pellacini et al. 2007; Todo et al. 2007; Obert et al. 2008]. While these methods are effective in controlling (global) illumination and low-frequency BRDFs, the view-dependent, high-frequency complexity of reflections and refractions is better addressed by our approach, which deforms the existing reflections and refractions. Light painting techniques become prohibitively work-intensive, if the view-dependent light pattern (in our case a reflected object) is complex. Anjyo et al. [2003] proposed the interactive control of highlight shapes, but this technique limited to the area of cartoon animations. Automated lighting design systems have also been proposed [Shacked and Lischinski 2001; Rusinkiewicz et al. 2006]. In theory, reflections outside physical bounds could also be realized using existing BRDF [Colbert et al. 2006] or BTF [Kautz et al. 2007] editing approaches. However, manually specifying an artist-directed, optimal and smooth per-pixel mirror direction (as resulting from our approach) would be a prohibitively laborious task.

Perception of reflections Reflections of real-world illumination are important for human perception of material appearance and shape, however, it is difficult for human observers to assess illumination consistencies [Fleming et al. 2003; Ostrovsky et al. 2005] and the correctness of a given reflection [Ramanarayanan et al. 2007]. This fact was exploited by Khan et al. [2006], who used parts of a photograph to approximate glossy environment maps. The human difficulties in understanding reflection patterns can be generalized to caustics [Gutierrez et al. 2008]. Tosun et al. [2007] use reflection lines to optimize surfaces in an offline process.

Intuitive deformation Editing reflections can be understood as deforming the field of reflection directions on a surface. Many techniques for the intuitive deformation of images [Igarashi et al. 2005] or surfaces [Sorkine and Alexa 2007] now exist. Our approach adapts a Moving Least Squares cost function, which has previously been successfully used in the domain of shape deformation [Müller et al. 2005; Schaefer et al. 2006].

Inhouse solutions Although unpublished, it can be assumed that some production houses have in-house tools to manipulate reflections. Such manipulations can be easily added to a professional rendering pipeline, e.g., by applying a global linear transformation within a programmable shader [Kopra 2007]. However, we are not aware of any specific software that explicitly addresses reflections or is similar to our interactive user-constraint-driven system, which allows smoothly blended, local, non-linear reflection edits.

3 Reflection Editing

According to the law of reflection, the angle of the incident ray **i** is equal to the angle of the reflected ray **r** for a perfectly mirroring surface (cf. Fig. 2). Given the surface normal **n**, the reflected ray is given by

$$\mathbf{r} = \mathbf{i} - 2\,(\mathbf{i}^\top\,\mathbf{n})\mathbf{n} \quad , \tag{1}$$

where **i**, **n**, and **r** are 3-vectors $(x, y, z)^\top$ normalized to 1.

In this paper, we present a real-time system for interactive reflection editing, which allows the user to violate the law of reflection by specifying a constraint that redirects the reflected ray **r** in another direction, given by the edited ray **e**. The user can specify this edited ray with an intuitive user interface that will be described in Section 4.

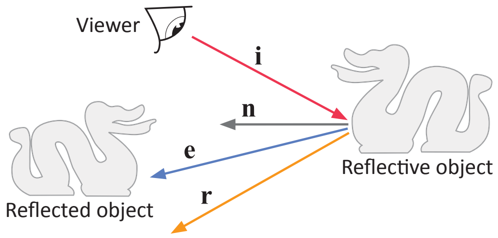

Figure 2: *Following the law of reflection, an incident ray* **i** *is reflected in direction* **r** *at normal* **n**. *With reflection editing,* **r** *is redirected to the edited ray* **e**, *to fulfill a user-defined constraint, e.g., to always reflect the left dragon.*

4 User Interactions

With a given 3D scene, the user can start to edit reflections by specifying a region of interest and using a mouse to put several constraints in the region (Fig. 3-a). The user can then move, rotate and deform the reflection in the region by dragging the constraints (Fig. 3-b and c) and inspecting the resulting reflections in real-time.

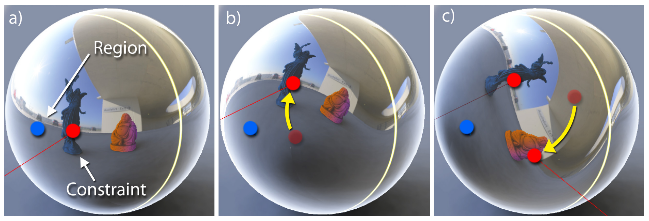

Figure 3: *(a) Specifying the region of interest on the sphere (blue center and white boundary) and a constraint (red point). (b) Moving the constraint to translate the reflection. (c) Putting in one more constraint and deforming the reflection.*

Constraints Each constraint has two handles: a red one on the surface of the reflective object and a green one located on the reflected object (red and green points in Fig. 4-a). Both handles can be moved to edit the reflection. The red handle drags a reflection

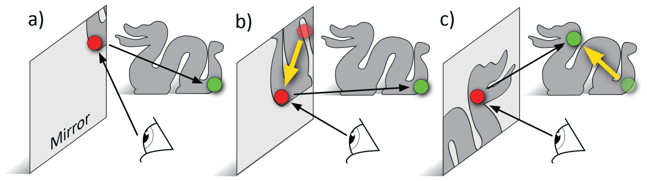

Figure 4: *(a) A constraint is placed on a mirror, which reflects only a small part of the dragon. The constraint visualizes how the green point is reflected at the position of the red point. (b) Moving the red handle (yellow arrow) on the mirror, drags down the reflection. (c) Dragging the green handle (yellow arrow), changes the position that is reflected at the red point.*

to a new location, e.g., in Fig. 4-b moving the red handle down on the reflecting mirror also moves the reflection of the dragon down. Moving the green handle from the reflected location to a different location makes this the new reflected location, e.g., in Fig. 4-c moving the dragon's head to the center of the mirror. Practice has shown that manipulation of the red handle is useful for subtle edits, whereas manipulating the green handle results in more substantial changes.

Regions To limit the influence of the edit operation within the scene, several tools are used for specifying the region of interest. Without a region, changing a reflection would alter the entire scene. Creating a region allows multiple independent and well localized edits within one scene. Once created, regions can also be moved by dragging their blue handle. The user can choose between three types of regions: Euclidean, geodesic, and free-form.

Euclidean regions are defined by a (blue) center and a radius in Euclidean space (Fig. 5-a). These are easy to use and work on multiple or disconnected objects. Geodesic regions use a surface location and radius, which is measured over the surface (Fig. 5-b). These are more intuitive if the object has a complex shape. Free-form regions are sketched on the surface (Fig. 5-c) and allow any shape to be achieved but these have to be manually defined by a user. These are most suited to expert users who want to work on small details. All regions have smooth-stepped borders with a

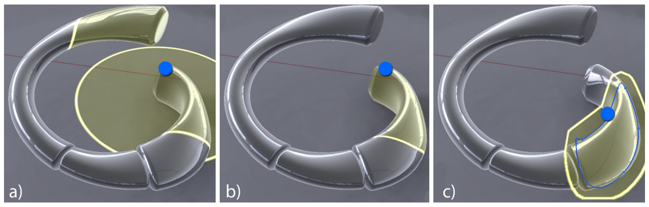

Figure 5: *Regions on a spiral object: Euclidean regions (a) influence the upper part, which can be avoided by using geodesic (b), or free-form regions (c).*

user-defined width. Edited reflections in each region are smoothly merged with non-edited reflections. All constraints *inside* a region influence the solution for the entire region. We define a constraint as *inside* a region, when the region's influence at the constraint location is nonzero. Constraints outside all regions have no influence.

View control When the user changes the viewpoint, the constraints no longer correspond to the original reflections because reflections are view-dependent. Nevertheless, a reflection constraint, created in its *generating* view, can be manipulated from any other arbitrary current view. There are two options to help the user manipulate the reflections in a consistent manner. When the user selects a constraint and presses the "return to view" button, the system reverts to the generating viewpoint where the constraint was created. The alternative option is to apply the "freeze" button and make the reflections temporarily view-independent: while changing the viewpoint, the reflection does not move but freezes on the surface. This allows the user to edit the reflection as it is visible from the generating viewpoint from another arbitrary view. This feature can be very useful in some situations, e.g., if a handle is occluded in the generating view.

Animation To simplify the handling of animated objects and deforming surfaces, we optionally made the reflecting and reflected position, as well as the region positions, relative to the (time-varying) surface. If the surface is deformed or the object is moving, the corresponding item naturally follows the object. This is useful for preserving a desired appearance on moving surfaces, and can also be used for special effects, where a reflection "tracks" a moving object. Additionally, a constraint can be keyframed over time, or can be dependent on any other scene parameter, such as the current viewpoint.

5 Interpolation Algorithm

If the user specifies multiple constraints, a smooth field of edited rays that interpolates between the user-defined constraints is sought, which can be found by a Moving Least Squares approach. Let N be the number of given constraints, where each is specified for the location $\mathbf{p}_n$ on the reflective surface, with $n = 1, \ldots, N$. For all locations $\mathbf{p}_n$, the user-defined constraints define the original reflected rays $\mathbf{r}_n$ and the edited rays $\mathbf{e}_n$. To find the interpolation at the intermediate positions $\mathbf{q}$, we determine the best 3×3 rotation matrix $\mathtt{R}(\mathbf{q})$ that minimizes

$$\underset{\mathtt{R}(\mathbf{q})}{\arg\min} \quad \sum_{n=1}^{N} w_n (\mathtt{R}(\mathbf{q})\, \mathbf{r}_n - \mathbf{e}_n)^2 \tag{2}$$

with

$$w_n = \frac{1}{d(\mathbf{p}_n\, ,\, \mathbf{q})^2} \quad . \tag{3}$$

Depending on the geometric complexity of the reflecting object, either the Euclidean or geodesic distance can be used as a distance metric $d(\ldots)$, as described in Section 6.2. Other weighting schemes could be used as well, however, it is important that the weighting w_n goes to infinity when the distance d approaches zero. This is required to make the reflection interpolating (instead of approximating), because only the local constraint is in effect when the distance is zero. We found that this property is essential because it guarantees that our edits accurately show the reflection of the green handle (the reflected location) at the red handle (the reflecting position).

Differentiating Eq. 2 with respect to the elements of $\mathtt{R}$ and setting the derivatives to zero yields the optimal solution

$$\mathtt{R}'(\mathbf{q}) = \left(\sum_{n=1}^{N} w_n \mathbf{r}_n \mathbf{r}_n^{\top} \right)^{-1} \sum_{n=1}^{N} w_n \mathbf{e}_n \mathbf{r}_n^{\top} \quad . \tag{4}$$

This solution does not enforce that $\mathtt{R}'$ is a rotation matrix, however, the rotation matrix $\mathtt{R}$ can be obtained by an ortho-normalization step. Ortho-normalization ortho$(\ldots)$ is achieved by polar decomposition of $\mathtt{R}' = \mathtt{R}\,\mathtt{S}$ into a rotational part $\mathtt{R}$ and a symmetric part $\mathtt{S}$. A fast algorithm can be found in [Horn 1987]. Because the first part of Eq. 4 is always symmetric, $\mathtt{R}$ can be determined by

$$\mathtt{R}(\mathbf{q}) = \text{ortho} \left(\sum_{n=1}^{N} w_n \mathbf{e}_n \mathbf{r}_n^{\top} \right) \quad . \tag{5}$$

Consequently, the edited reflection direction for each position $\mathbf{q}$ on the surface is given by

$$\mathbf{e}(\mathbf{q}) = \mathtt{R}(\mathbf{q})\, \mathbf{r}(\mathbf{q}) \quad . \tag{6}$$

It is worth emphasizing that this solution to $\mathtt{R}(\mathbf{q})$ allows for very intuitive editing, because the user merely specifies the edited ray $\mathbf{e}_n$ and does not need to define complete rotation matrices at each constraint location. The 3×3 matrix $(\mathbf{e}_n \mathbf{r}_n^{\top})$, used in Eq. 5, can be interpreted as a degenerated rotation matrix of rank 1. To upgrade this matrix to a full rotation matrix the user would have to specify an additional degree of freedom, namely, the amount of rotation around the edited ray. The ortho-normalization step automatically fills this degree of freedom by taking the neighboring constraints into account.

If the viewpoint is changed, the edited reflection direction $\mathbf{e}(\mathbf{q})$ must be recalculated with Eq. 6, because the reflected ray $\mathbf{r}(\mathbf{q})$ is dependent on the direction of the incoming ray $\mathbf{i}(\mathbf{q})$. However, such a change in viewpoint does not affect the rotation matrix $\mathtt{R}(\mathbf{q})$ because the constraints are associated with the viewpoint for which they were specified by the user. This means that the edited reflections are view-dependent, and therefore behave similarly to physically correct reflections under changing viewpoints. The rotation matrix $\mathtt{R}(\mathbf{q})$ only needs to be re-calculated when the user changes existing constraints or specifies additional constraints.

Following Eq. 6, the constraints have an infinite area of influence. For example, if the user specified a single constraint, all reflection directions on the surface would be altered with the same rotation matrix, originating from that single constraint. To limit the area of influence of constraints and to allow better control over complex editing operations, the user has the option of specifying a region of interest. For each surface point $\mathbf{q}$ a region of interest defines a value $a(\mathbf{q})$ in the interval 0.0 to 1.0 (usually this value is 1.0 in the central area of the region and has a smooth fall off to 0.0 at the region's borders). For a smooth transition to the unaltered reflection direction $\mathbf{r}(\mathbf{q})$ at the border of the region, the rotation matrix $\mathtt{R}$ from Eq. 5 is replaced by

$$\mathtt{R}_a(\mathbf{q}) = \text{ortho}\Big(a(\mathbf{q})\ \mathtt{R}(\mathbf{q}) + \big(1.0 - a(\mathbf{q})\big)\ \mathbf{r}(\mathbf{q})\mathbf{r}(\mathbf{q})^{\top} \Big) . \tag{7}$$

For the calculation of $\mathtt{R}$ (and the resulting $\mathtt{R}_a$) only those constraints that fall within the region of interest are considered, i.e., $a(\mathbf{p}_n) > 0.0$. If no constraint lies in the region of interest, the unaltered reflection direction $\mathbf{r}(\mathbf{q})$ is used.

Our system allows the user to specify multiple regions of interest that are mutually independent and can be edited completely on their own. If multiple regions overlap, their contributions are weighted according to their specific value of $a(\mathbf{q})$.

6 GPU Implementation

When editing visually complex reflections, high quality rendering and real-time feedback are necessary. In this section we describe how this can be achieved with current consumer hardware by stream processing on the GPU.

6.1 Interpolation

To generate a smooth interpolation field of edited reflection directions, the edited reflection direction $\mathbf{e}(\mathbf{q})$ (cf. Eq. 6) must be calculated for every pixel. This can be done in parallel on the GPU. First, we render the visible part of the scene into a texture, where each pixel stores the 3D position and normal that is visible at this pixel. Second, the weights w_n, from Eq. 3, for all N constraints and all pixels are computed and stored into an N-layer array texture. While an Euclidean weighting can be computed on-the-fly, geodesic weights need to be computed for each pixel (as described in subsection 6.2). Third, all constraints and regions are uploaded as shader constants to the GPU. The original reflected ray $\mathbf{r}(\mathbf{q})$ for each pixel can be calculated using the given position, normal, and the current viewer position. Afterwards, we solve for the edited reflection direction $\mathbf{e}(\mathbf{q})$ with Eq. 6. In this process, all computations are done in parallel and independently for each pixel. A final step passes the calculated edited reflection direction to the rendering system to compute a reflection along this direction.

6.2 Geodesic Distance

The geodesic distance can be used instead of Euclidean distances in Eq. 3. Assuming densely tessellated meshes, we coarsely approximate the continuous geodesic distance using the discrete distance along triangle edges. Letting $\bar{\mathbf{p}}_n$ be the mesh vertex closest to the constraint position $\mathbf{p}_n$, we compute the discrete edge distance from $\bar{\mathbf{p}}_n$ to every other vertex on the mesh using Dijkstra's algorithm on the CPU. Then, on the GPU, we approximate the geodesic distance $d(\mathbf{p}_n, \mathbf{q})$ from $\mathbf{p}_n$ to a surface location $\mathbf{q}$, by interpolating the distance from the discrete distance at the vertices of the triangle that contains $\mathbf{q}$. As the location $\mathbf{q}$ always corresponds to a screen pixel, this can be done efficiently, just by drawing the per-vertex discrete distance field using smooth shading in one pass. Every layer of the N-layer array texture, which stores the weights, can be calculated simultaneously. While this approximation works well for dense meshes, efficient high-quality approximations for general meshes have been published [Surazhsky et al. 2005].

6.3 Rendering

Editing of reflections is independent of the reflection rendering implementation itself. For high frame rates and interactive feedback, reflection mapping is used by our system although fast raytracing could alternatively be employed.

Our system applies a reflection mapping technique similar to the technique proposed by Heidrich and Seidel [1999], on top of a (precomputed) diffuse global illumination. For reflection mapping, the scene is broken up into individual objects and one cube map is rendered from the center of every object. If the reflection direction is edited, the altered reflection can simply be read from a different position in the cube map. The diffuse lighting is unchanged when the reflection direction is edited.

When the user performs an extreme edit, it is possible that the edited reflection direction points below the surface. Even in such a case, since our system will simply reflect the object behind the surface, the edited reflection is always smooth and there are usually no visible artifacts, such as discontinuities.

Raytracing is another option for rendering multiple, high quality, local reflections and refractions. Recent GPU approaches [Zhou et al. 2008] allow for dynamic scenes at interactive speed, but their implementation is intricate. While in theory raytracing scales well with geometric complexity, scenes with several 100k faces have not been demonstrated to run at the same speed as that granted by rasterization based reflection mapping. Raytracing glossy BRDFs is more exact, but is also much more time-consuming compared to pre-convolved [Heidrich and Seidel 1999] reflection maps.

7 Results

7.1 Applications

This system can be applied to all computer-generated 3D scenes, as used nowadays for product visualization, movie and TV productions, or in computer-generated art. Fig. 9 shows some examples (please see the video provided with this paper for details of the process by which the edits were created). As complex geometry can result in visual masking [Ramanarayanan et al. 2007] our examples use mostly smooth surfaces to expose the edit quality. Some examples for complex geometry can be found in the supplemental video. In Fig. 9-a the motivating example from art was reproduced (29.1 fps). The planar mirror is simple to handle with a single constraint and a single region, allowing an experienced user to comfortably perform the edit operation in less than half a minute. The "Kitchen" scene, Fig. 9-b, is a complex scene with several 100k triangles and multiple reflecting objects. With 3 regions, 4 constraints and 4 reflecting objects, it still can be edited at 16.8 fps. Fig. 9-c shows a scene, called "Ring", which might be used in a movie or TV production. The example also serves to show that reflection editing is not limited to mirror-like BRDFs, because it features reflections on a glossy metal ring (11.1 fps). The edit operation on the "Car" hood in Fig. 9-d shows how our method could be applied to product visualization. Even unclean meshes of complex shape and topology, such as the car body can be handled (9.4 fps). The video goes into more detail on how constraints can be applied to moving or deforming meshes.

Highlight editing Our system provides the means for highlight editing as a special case. In Fig. 6, a round highlight is stretched over the side of the car.

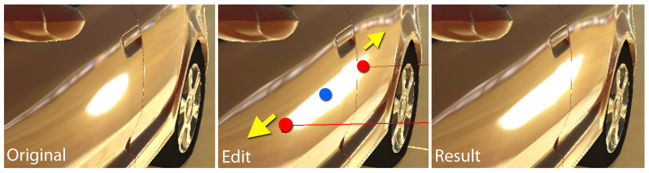

Figure 6: *Highlight shape editing (2 constraints, 1 region, 10.8 fps).*

Decoupling shadows and highlights In physically correct rendering, the location of highlights and shadows are coupled. For artistic reasons, however, it can be useful to decouple them. In Fig. 7, the scene is lit by an area light that is reflected inside a collection of objects. While keeping the shadows in place, our technique allows the user to move the highlight into a more prominent place,

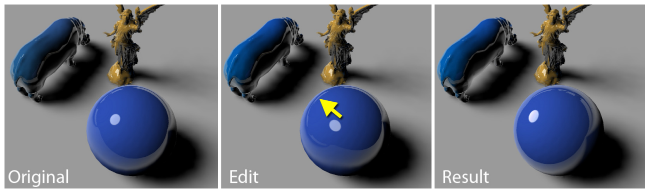

Figure 7: *In this example the highlight was moved to another place. Starting from the original image, the highlight is moved, to give it the desired more pronounced look, yet left all the soft shadows unchanged (1 constraint, 1 region).*

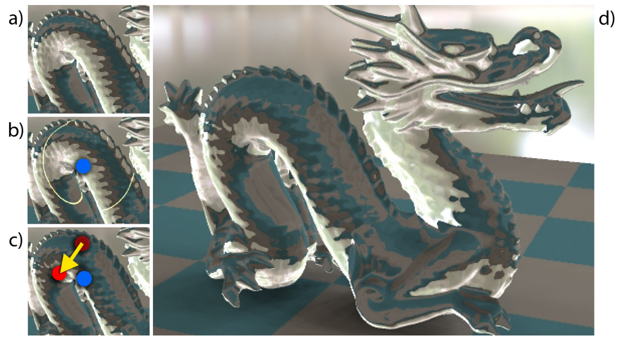

Figure 8: *Refraction edit (1 constraint, 1 region, 29.1 fps): Starting from the original (a), a region is defined (b), and a constraint is manipulated (c) until the desired result is achieved (d).*

consistent with the reflection of other objects, but still keeping all soft shadows in place.

Refractions As an extension to reflection editing, the system can also be used for interactive refraction editing. The user interface for refraction editing is similar to that of reflection editing. However, refraction constraints define a refracting position and direction, but no refracted position. This is because specifying a point that should be refracted is not unique: a ray could be edited when either entering or exiting the object. We have therefore reverted to directly manipulating the exiting refraction direction. From a user's perspective, it is not possible to define which world position should be refracted where, but dragging the refraction works as expected. The refracted direction must be specified manually and adjusted until the desired effect for this particular geometry is achieved. In Fig. 8 an example of refraction editing is shown (at 27 fps).

7.2 Performance

We measured the performance of our reflection editing technique on a 2.4 GHz CPU with an NVIDIA GeForce 260 GTX. Please see Fig. 9 for timings. All our results are rendered at 1280×960, where reflection editing took less than 100 ms. The most expensive computation is the reflection interpolation, which requires at least one orthogonalization per pixel: a one-constraint system is solved with 24.5 megapixels / s while 32 constraints still result in 18.5 megapixels / s. Moving a geodesic region requires a re-computation of the geodesic distance field on the CPU, which is done at interactive speed (e.g., in 139 ms for the 525 k triangle "Lucy" model used in Fig. 7).

7.3 User Studies

We performed two user studies to evaluate our system. In the first study the usability of our system was investigated. In the second study the visual quality of the reflection edits was assessed.

16 subjects, all novice users of our system, participated in the first study. Three where professional 3D graphics designers, while the others did not have much experience in 3D modeling or post-production. On average all users rated their skills in using commercial modeling software 2.75, where a score of 0 is worst and 10 is best. After a detailed tutorial, which took approximately 5 ± 2 min, each subject was given three 3D scenes and corresponding goal images of the target reflections (Figs. 9-a, 9-b, and 6). The goal images were designed by the authors in advance. The subjects were asked to use our system to adjust the reflections in each 3D scene so that they look similar to the goal image. The participants were allowed to work on the task until satisfied. Everybody completed the tasks successfully in a very short time (on average $2{:}22 \pm 1{:}01$ min:sec, $4{:}04 \pm 1{:}44$ min:sec, and $2{:}00 \pm 1{:}16$ min:sec for the task of Figs. 9-a, 9-b, and 6, respectively). When the users were asked whether our system was useful for achieving the task of the session, the result was an average rating of 9.42 for all tasks and users, where 0 considered worst and 10 best.

We were especially interested in the feedback of the three professional 3D graphics designers. Initially, we asked them to go through the same tutorial and tasks as performed by the novice users. Their reactions were positive: our system seems to be the first one that enables them to edit reflections quickly and easily. We then asked if it would be possible for them to achieve the same tasks using the commercial software that they usually work with. They stated that this would be difficult, then went on to suggest the following possibilities:

Multiple passes It is possible to edit reflections by moving, deforming or changing the reflected objects using the commercial software. However, it is still difficult or impossible if the reflected objects and their reflections are visible in the scene, e.g., the sink reflected on the pot in Fig. 9-a. As a result, with this method, the designer has to render the scene at least two times: once as an unedited scene, and the other with edited reflections. The designer then uses a 2D tool to make a composite of the two images (or sequences) by introducing an alpha matte.

Texture baking It is possible to bake the rendered reflections as the texture over the surface and edit it. However, with this method, it is difficult to change the viewpoint later or manipulate an animation sequence.

Normal editing Several current commercial products support a tool to edit surface normals. However, it is difficult and unintuitive for a designer to predict how the edited surface normals affect the final rendering result. Moreover, normal editing changes the diffuse lighting, which should ideally be unaffected.

Through this first user study, we have shown that, after a short training session, our user interface for editing reflections is easy to learn as well as intuitive even for novices. The system also solves the problems pointed out by the professional users of conventional tools.

In the second study the visual quality of the edited reflections was evaluated. We presented a total of 9 videos showing a Buddha statue, where 8 videos contained differently edited reflections and 1 video showed the unedited reflections, to 20 subjects. In the videos the virtual camera is orbiting around the Buddha statue so that the reflections can be evaluated from different viewpoints. The subjects were able to activate slow playback or to stop the video to observe the edited reflections very carefully. The Buddha model was chosen because it contains complicated as well as smooth geometry. The supplemental video shows a few seconds of the 9 videos used in this study.

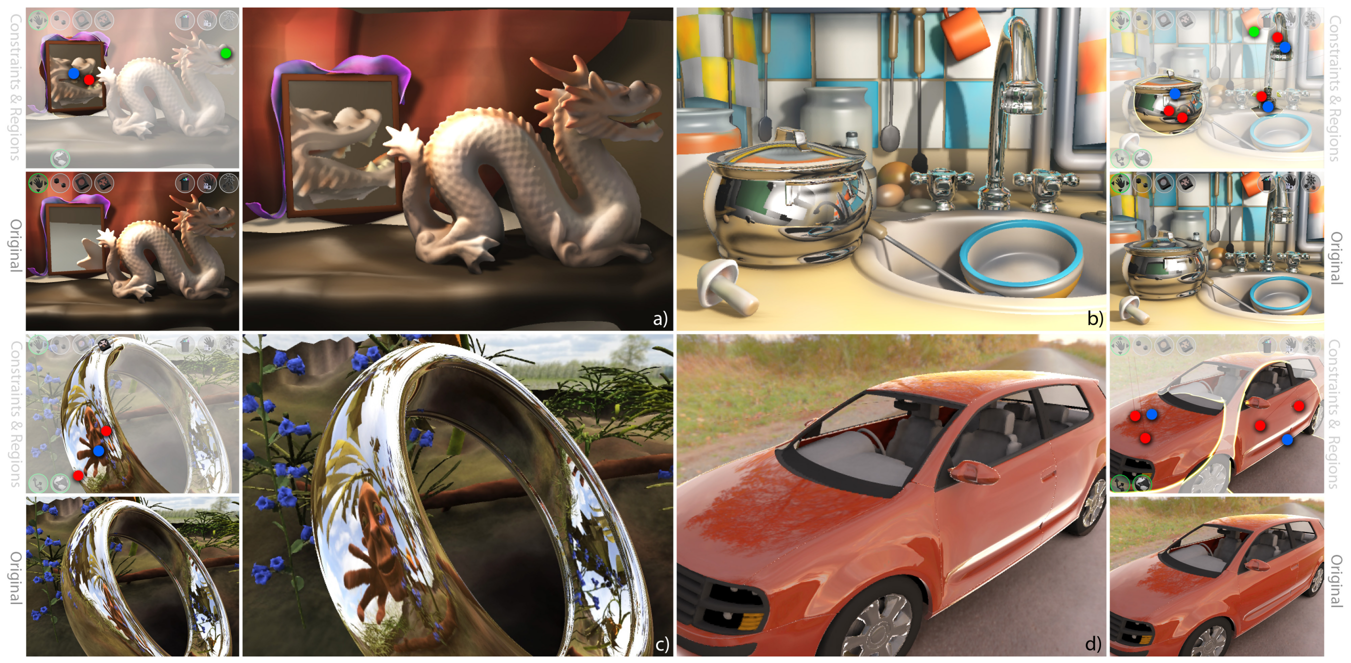

Figure 9: *Applications: a) In a physically correct rendering the tail of the "Rokeby dragon" is visible in the mirror from the current viewpoint. After reflection editing the mirror reflects the dragon's head (1 constraint, 1 region, 29.1 fps). b) "Kitchen" is a complex scene with a large number of objects and triangles. After reflection editing the sink is displayed at a different position in the large reflective pot (4 constraints, 3 regions, 16.8 fps). c) Editing the reflections on this "Ring" makes the reflected face more visible (2 constraints, 1 region, 11.1 fps). d) A user changed the tree reflected in the hood of this "Car" to become more visible, also editing the highlights (4 constraints, 2 regions, 9.4 fps).*

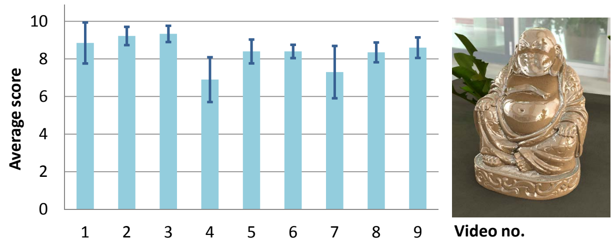

Figure 10: *Average rating of the perceived physical correctness of different reflection edits on a Buddha statue (0 is worst and 10 is best). Video 9 is the original version of the scene with unedited reflections. The small error bars (dark blue) indicate the standard deviation of the ratings.*

The subjects were asked to find reflections that are physically incorrect and rate the physical correctness of reflections on a scale of 0 to 10, where 0 is worst and 10 is best. Fig. 10 compares the average ratings for each of the 9 videos. As the subjects were aware that the reflections were edited, they were very eager to find the slightest errors. Some participants even thought they found errors in the unedited video, which received an average rating of 8.6 out of 10. As some of our edited versions received even higher average scores than the unedited version, it can be concluded that the edited reflections are usually very hard to detect. Slightly lower average scores were given for video 4 and 7, which both contain strong edits in smooth regions on the belly and the head of the Buddha. This confirms the observations made in literature [Ramanarayanan et al. 2007] that the physical correctness of reflections is harder to assess on more complicated geometry, and edits can be more easily detected on smooth surfaces. When we asked the subjects what was wrong with the reflection edits they had ranked as low, most answered that the reflections were at the wrong position or appeared deformed. Nobody thought that the edited reflections caused a shape deformation because of the way that the reflections move when the viewpoint is changed. We expected this result for the complicated shape of the Buddha statue, however, it must be stressed that local reflection edits on very regular surfaces, like on a flat plane or sphere, can easily result in a perceived shape deformation under changing viewpoints. When we asked the subjects if they could detect the unedited version of the video out of the 9 possibilities, only 3 out of 20 participants were willing to make a guess, but all three guessed incorrectly. Finally, we showed all 9 videos simultaneously and asked the subjects to compare the unedited with the edited versions. All subjects were able to identify the performed reflection edits based on a given textual description. This proves that the edits were significant enough to alter the appearance of the original.

Through this second user study, we have shown that the edited reflections are usually very hard to detect. Even if, in the case of strong edits, a careful observer notices that the reflections appear deformed or are located in the wrong position, the edits do not cause a perceived surface deformation of the Buddha statue.

The full set-up and analysis of both user studies are detailed in the supplemental material.

8 Limitations and Future Work

The system does have some limitations that may lead to future work. Whether an edit is acceptable is highly dependent on the scene, the camera motion, and the performed edit. Exaggerated edits on smooth surfaces, like on a flat plane or sphere, can readily be assessed as physically incorrect, and can become noticeable under changing viewpoints or within animated scenes. However, reflection edits are very difficult to detect on more complicated surfaces, as verified by our user study. Our system does not currently prevent users from creating non-realistic or unpleasant reflections. Consequently, when using our system, the artist must always check the edits before they can be applied in production. With our real-time system, the user can easily explore the space of possible solutions between physically correct rendering and the artistic goals. Developing criteria for acceptable edits would be challenging future work.

Another limitation occurs if the user specifies too many constraints in a small region. Because the algorithm tries to fulfill all constraints, the generated field of interpolated reflection direction is no longer smooth and is difficult to control.

Our system does not support the bending of reflection rays, which means that it is impossible to reflect objects that are occluded.

Using a real-time raytracer as the underlying renderer would allow to experiment with multiple bounces of reflections, multiple refractions, or mixtures of them.

Visually distracting flickering can occur if the red handle (reflecting location) is dragged over a high-frequency surface (e.g., with bump or displacement maps), because the change in reflection direction then also occurs with high frequency. However, this distraction is only encountered while dragging and does not compromise the final result. Optionally, flickering can be suppressed by using a smooth version of the geometry.

It has been shown in practice, that careful placement of region borders, e.g., locating them in areas of high surface curvature, gives more pleasant results. Automatic or guided placement of regions to make edits less objectionable is a possible avenue of future work.

In future, it would be possible to generalize the idea of local constraint-based editing beyond physical laws also to other phenomena, such as soft-shadow penumbrae or caustics.

9 Conclusion

This paper introduces a system for reflection editing, which will formally bridge artistic goals with the laws of reflection as practiced by traditional artists for centuries. With this system the digital artist of today can specify constraints for reflection positions via an intuitive user interface. If multiple constraints are given, the system optimizes a Moving Least Squares cost function for each pixel to generate the optimal interpolation field of edited reflection directions. The system is easy to implement, works without preprocessing, runs in real-time on modern graphics hardware, and is independent of the underlying reflection rendering algorithm. It is not limited to simple reflections and will also allow more general specular light transport such as glossy reflections and refractions. The applicability of the system was successfully verified by a user study including feedback from professional 3D designers.

References

ANJYO, K.-I., AND HIRAMITSU, K. 2003. Stylized highlights for cartoon rendering and animation. *IEEE Comput. Graph. Appl. 23*, 4, 54–61.

BRAHAM, A. 1976. *The Rokeby Venus, Velázquez*. Painting in Focus, No. 5. National Gallery, London, UK.

COLBERT, M., PATTANAIK, S., AND KRIVANEK, J. 2006. BRDF-shop: Creating physically correct bidirectional reflectance distribution functions. *IEEE Comput. Graph. Appl. 26*, 1, 30–36.

FLEMING, R. W., DROR, R. O., AND ADELSON, E. H. 2003. Real-world illumination and the perception of surface reflectance properties. *J. Vis. 3*, 5 (7), 347–368.

GUTIERREZ, D., SERON, F. J., LOPEZ-MORENO, J., SANCHEZ, M. P., FANDOS, J., AND REINHARD, E. 2008. Depicting procedural caustics in single images. *ACM Trans. Graph. 27*, 5, 120.

HEIDRICH, W., AND SEIDEL, H.-P. 1999. Realistic, hardware-accelerated shading and lighting. In *Proc. SIGGRAPH '99*, 171–178.

HORN, B. K. P. 1987. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A 4*, 4, 629–642.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3, 1134–1141.

KAUTZ, J., BOULOS, S., AND DURAND, F. 2007. Interactive editing and modeling of bidirectional texture functions. *ACM Trans. Graph. 26*, 3, 53.

KHAN, E. A., REINHARD, E., FLEMING, R. W., AND BÜLTHOFF, H. H. 2006. Image-based material editing. *ACM Trans. Graph. 25*, 3, 654–663.

KOPRA, A. 2007. *Writing mental ray shaders: A perceptual introduction*. Springer-Verlag New York, Inc., NJ, USA.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3, 471–478.

OBERT, J., KRIVÁNEK, J., PELLACINI, F., SÝKORA, D., AND PATTANAIK, S. N. 2008. iCheat: A representation for artistic control of indirect cinematic lighting. *Comput. Graph. Forum 27*, 4, 1217–1223.

OSTROVSKY, Y., CAVANAGH, P., AND SINHA, P. 2005. Perceiving illumination inconsistencies in scenes. *Perception 34*, 11, 1301–1314.

PELLACINI, F., BATTAGLIA, F., MORLEY, K., AND FINKELSTEIN, A. 2007. Lighting with paint. *ACM Trans. Graph. 26*, 2, 9.

POULIN, P., RATIB, K., AND JACQUES, M. 1997. Sketching shadows and highlights to position lights. In *Proc. Computer Graphics International '97*, 56–63.

RAMANARAYANAN, G., FERWERDA, J., WALTER, B., AND BALA, K. 2007. Visual equivalence: towards a new standard for image fidelity. *ACM Trans. Graph. 26*, 3, 76.

RUSINKIEWICZ, S., BURNS, M., AND DECARLO, D. 2006. Exaggerated shading for depicting shape and detail. *ACM Trans. Graph. 25*, 3, 1199–1205.

SCHAEFER, S., MCPHAIL, T., AND WARREN, J. 2006. Image deformation using moving least squares. *ACM Trans. Graph. 25*, 3, 533–540.

SCHOENEMAN, C., DORSEY, J., SMITS, B., ARVO, J., AND GREENBERG, D. 1993. Painting with light. In *Proc. SIGGRAPH '93*, 143–146.

SHACKED, R., AND LISCHINSKI, D. 2001. Automatic lighting design using a perceptual quality metric. *Comput. Graph. Forum 20*, 3.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Proc. SGP '07*, 109–116.

SURAZHSKY, V., SURAZHSKY, T., KIRSANOV, D., GORTLER, S. J., AND HOPPE, H. 2005. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph. 24*, 3, 553–560.

TODO, H., ANJYO, K.-I., BAXTER, W., AND IGARASHI, T. 2007. Locally controllable stylized shading. *ACM Trans. Graph. 26*, 3, 17.

TOSUN, E., GINGOLD, Y. I., REISMAN, J., AND ZORIN, D. 2007. Shape optimization using reflection lines. In *Proc. SGP '07*, 193–202.

ZHOU, K., HOU, Q., WANG, R., AND GUO, B. 2008. Real-time KD-tree construction on graphics hardware. *ACM Trans. Graph. 27*, 5, 126.