An overview on light edition

Raphaël Jorel

October 21, 2017

Abstract

In computer graphic (CG), light manipulation is not an easy task. In fact, mastering every lighting source and knowing how each contributes to global illumination of a scene is difficult, even for trained users. For that, many tools and methods have been proposed to artists, for improving quality of produced images and user efficiency. This article is a little overview of some methods, to have a little idea of some approaches and see what can users do with them.

1 Introduction

The classical way to manipulate light is direct edition. Users place lights in a scene and light effects are applied. A way to diffuse the light is using physically-based rendering, to get plausible images, because this method is based on physically models for lighting.

But manipulating directly lights is not intuitive every time. When a shadow is involved by many objects or lights, users could take a lot of time just to guess how each light contributes to the shadow. And it's not finished, because when they achieve that, they have to determine the parameters of imaged-editing operations to get specific changes. It requires extensive trial-and-error work.
For that, research have been made in this way, to give more freedom to artists and manipulating light features, as shadows or highlights. This type of edition is described as indirect. An other approach is painting interfaces: user paints a goal image, while light parameters are optimized to attempt to match it.

The idea to give more freedom is also an other interesting point. As astonishing as it may seem (for me), sometimes users might want to "cheat" on light. Physically-based rendering is very efficient but restrictive, because users can edit light but not change light effects. In order to solve partially this problem, many tools give possibilities to change light effects, light diffusion and so on, even if the result is non-physically possible. The responsability of plausible images is given to users in this case.

For this little overview, I propose to start with the article [1], because it proposes two methods, direct and indirect.
Secondly, I will talk about some others tools around direct and indirect edition, focused on shadow or other lighting feature edition. I won't approach painting interface, because I think it would be too much for this overview, and moreover a previous user study [2] found that this type of interfaces were not effective, due to inability of users to paint a precise depiction of lighting.

2 Light edition

2.1 Path editing

The article [1] presents two approaches, direct and indirect, to allow non-physically edition of path lights. They are named *path retargeting* and *path-proxy linking*. The tool provides an intuitive way to select light paths.

Light selection Light path selection uses (and extends) Heckbert path notation to allow path selection with it, but it focuses on the possibilities to choose caustics without the notation and just by visual selection. The user defines a region, in using bounding volumes, or selects objects that interacts with the light transport. The system automatically computes and ranks incident light paths by their contribution and the surrounding light transport. The user chooses then the desired light path to edit.

Path retargeting The system uses global illumination (GI) to render images. So, there are light and eye subpaths, and it's possible to retarget both. The user chooses one and then retarget the subpath's endpoints. He has to keep in mind that retarget a path alters indirect illumination, such that indirect occlusion or interreflection.

Path-proxy linking The idea of *path-proxy linking* is offering the possibility to specify different object transformations for individual components of the light transport. The user selects direct light on a surface, picks the shadow casting object and are then presented with a proxy object used only for shadow computation (during GI operation). After that, original object is ignored during this operation. Each object can have many proxies. These proxies can be manipulated to change shadow rendering of the attached object, according the selected light transport. The possible transformations are the same as operations on objects, so they can be transformed using standard interface of digital content creation (DCC). Proxies are invisible, only their shadow effects are taked in account.

Observations Clearly, this tool offers a more important freedom, just because they give an intuive way to select light paths and the possibility to cheat on light rendering, but I'm not sure to see exactly their usefulness.
Moreover, *path-proxy linking* needs to know which lights and objects involve a shadow. As the paper said, user can be helped by an external tool, like [3]. So, to be very intuitive, the original tool needs another one. Maybe integrating it directly, or proposing a default tool, could be more practical for users.

2.2 Shadow edition

Manipulating shadows directly is a solution proposed by the article [4]. Here, they are considerated as first-class modeling primitives, like objects and lights, and can be manipulating in the same way.

Method The user chooses a shadow to edit by clicking. The system then selects the light-object pair that casts the shadow. The user can move, rotate, scale this shadow as if it was a object on the surface of the scene. To move it, the user has just to drag it to the desired position. Transformations are applied whether to the light, whether to the object, depending on the interaction mode. When there are more than one object is casting the selected shadow, the closest one to light source is kept. In presence of more than one light, the closest one to the hotspot operation is kept. Although this behaviour is well in a lots of cases, the user can choose explicitly the light-object pair. If the user would like to move light, every shadow transformations is applied to light, by moving and rotating it, otherwise the object is affected by changing its position and rotation. The same operations are possible on hotspots, the user selects and transforms them as he wants and the system moves lights according the edition.

The system provides also light and shadow constraints. The user can specify, by painting, that an area should not change its "shadow state" (to keep the terms of the article). If an area is marked with a shadow constraint, this enforces the fact that this area will always remain in shadow by at least one object. With light constraint, it's the opposite effect. No shadow should be in an area marked with light constraint. The system updates light and object positions only if the constraints are valid.

To finish, the system gives also the possiblity to put fake shadows and lights, named cookies, in a scene. The user adds these fake effects by painting, and can attach them to a light source or the world. This affects how the cookie is changed when the light source is moved.

Observations Contrary to *path-proxy linking*, we have here a method which moves objects and lights in a scene, according shadow edition. This method seems to be good to keep a plausible image, but the problem is the light and object selection when the user chooses a shadow. Although the user can explicitly choose the light-object pair, it's not efficient in the case of many objects or lights contribute to a shadow. Here, just one light or object can be move for a shadow. So, this method can't be used in very complex scenes or image.

2.3 BendyLights

This method is described in the article [5] and proposed a spotlight-based lighting model for artistic control in which light travels along non-linear paths from a point source.

Method In real scenes, light rays follow physical laws, i.e. linear paths. But this is restrictive, users can't modify how the light is propagated. *BendyLights* gives the possibility to change light travelling, along nonlinear splines from a point source. Light is represented as a tube surrounding the spline. To manipulate this tube, users move spline control points in space.

BendyLights proposes two ways to control this tubes, directly and indirectly. In the first, users can move, rotate and scale the tube, and also deforms them just by moving spline control points. In the second, users drag shadows, hotspots or highlights, and the result is automatically computed by changing tubes. To achieve it, *BendyLights* adapts technics viewed in [4].

Observations *BendyLights* is in the same reasoning that *path retargeting*, i.e. letting the freedom to change light propagation. But applications are different, because this latter works on light path whereas this former works on spotlights and builds tubes around them to permit light edition.
Direct and indirect edition allows a very good precision for lighting feature changes.

2.4 envyLight

[6] introduces *envyLight*, a tool that lets users to indicate which lighting feature they desire to edit in the rendered image. It splits image into two layers and proposes some operations to apply to the selected lighting feature.

Method To select a lighting feature, the user has two strokes, one to indicate parts of the image that belong to the lighting feature and another to indicate parts that don't. From that, the system splits the image into a foreground and background layer, such that edits to the foreground layer affects the feature the user marked and such that the sum of the two layers is equal to the original image. The supported lighting features are shadows, highlights and diffuses.
Once the image splitted, the user can apply operations to the layers. *envyLight* proposes a small set of simple editing operations: brightness, tint, contrast, translation, blur and sharpen.

Observations Contrary to *path-proxy linking* or methods in [4], this tool extracts lighting features from original image to edit them separatly. For users, it's very useful because extracting the desired lighting features permits to focus on them one-by-one. The weak of the tool is possibly the very small number of possible operations to apply on lighting features.
Another important point to emphasis is the difficulty of used for novice designers. In fact, editing a lighting feature affects the other ones. So editing highlights will affect shadows, and conversely. According the article, it's normal because it's inherent to physical lighting models, but novice designers may have troubles in using this tool.

2.5 Visibility editing

[7] presents a method that decouples shadowing from lighting, to be focused only on the shadows and edit them intuitively.

Method This tool aims to allow editing of shadows indepently of lighting. To achieve that, it extracts shadowing from lighting, to manipulate shadows more intuitively. For that purpose, it precomputes scene visibility and apply transformations to this data, regardless lighting. The tool gives some editing operations, namely translation, rotation, shadow removal, shadow gradient and shadow blur. Every operation affects visibility scene.

The tool permits also to edit multiple shadows involved by an object, one-by-one. The system finds which lights are responsible for shadows in the image. To start the process, the user has to paint over a shadow area and identify shadowed surface sampling area. The system then permits individual edition of each shadow, by displaying the desired one to modify.

In addition, the user can choose to work in a fixed viewpoint or a free viewpoint, to be adaptable in more situations. The examples given are movie production and video games. Fixed viexpoint is more suitable to the former, whereas the other is a better choice for the latter.

Observations This tool is in the same idea as [6], because both propose to extract / decouple a lighting feature from the image to allow transformations on this one. But contrary to *envyLight*, this method doesn't permit to choose the lighting feature and focus shadows. Shadow edition is automatically processed more (and fortunately, I want to say). The possibility to edit multiple shadows casted by an object can be done with *envyLight* but this system provides automatic selection when the user paints the desired area. *envyLight* needs to be more explicit. Moreover, shadows are modified regardless lighting, so editing a shadow doesn't edit an other feature, contrary to *envyLight*.

3 Reflection editing

[8] introduces a system that transforms physically correct reflections into art-directed reflections.

Method As the article underline it, humans have troubles to assess physical correctness of reflections, so the authors exploit this weakness to change reflections and get a non-physically plausible images, but observers won't find it weird.

To edit a reflection, the user chooses a region of interest and specify constraints with classical workflow, i.e. by dragging point in the scene. Each constraint is composed of two handles: one on the reflected object (green) and another one on the surface that reflects this object (red). The system computes reflection to match the two handles, so that the part of object marked by green handle is reflected at the location pointed by the red one.

To avoid application of reflection deformations to the entire scene, the tool provides region selection in three different ways: euclidean, geodesic and free-form. The former is used to selected an area according a point and a radius, the second permits a more intelligent object selection and the latter is a free selection.

Reflections are view-dependent, so changing scene viewpoint changes also reflections on objects. To help users in manipulating constraints, the tool gives two options for viewpoint changes. The first is the possibility to get back to the used viewpoint for a constraint. The second is the freezing, with which the user can block the reflection. This latter won't be changed when viewpoint will change. It permits to edit the reflection as it is visible from the generating viewpoint from another viewpoint.

Observations Reflections are another type of lighting features we didn't approach before in this article. The idea to modify reflections on surface is, once more, a cheat. Even the other tools don't mention it, this possibility to cheat on light rendering

comes from this difficulty for humans to assess the exactly light travel in scene.
This tool provides a very intuitive way to edit reflections, but it has some issues with exaggerated edits on smooth surfaces or detection of complicated surfaces. Moreover, too many constraints in small regions may lead to unexpected results, owing to used algorithm which tries to satisfy every constraint.

4 Discussion

4.1 Performances

When we compare or present different tools, it's always interessing to have an idea of system speeds, to know if these systems are really usable. An overmighty idea and concept is really interesting when we can deploy it in existing computer. Every method presented here has usable, but in general tests are done on machines that are not cheap (at the time of publications, anyway).

Unfortunately, it's impossible to compare all these tools, in term of performance, because each test is realized on different configurations and results are not express in the same way. Sometimes it's just the time to accomplish specific treatments, other times it's the number of frame per seconds (FPS), and so on. But the main goal of this type of works is not always incredible speed, but rather intuivity and freedom given to users. Of course, a very slow tool is not wished, but every presented tool is real-time usable.

4.2 Intuitivity

This point is crucial for tools reffered to artists and designers. To be adopted, a method has to be simple to manipulate. Designers and artists are not technical persons, so they don't want to have too complicated plugins or tools. But some of them may want to go further, and we saw that tools like *path-proxy linking* and *path retarget* let the freedom to choose caustic according the Heckbert's notation. Classical workflows permit intuitive selection, but selecting an entity according its representation is always more precise.

But concerning intuivity, the presented tools are all convincing because they are all based on known workflows used by artists and designers. Most of them (it's not specified every time) are implemented as plugins for existing DCC, like Photoshop or Auto Maya Desktop. So these tools use familiar interface for designers and artists.

However, intuivity doesn't imply to create complex scene easily. For instance, *envyLight* expresses this fact in saying that novice users will struggle to use it, because lighting effects are not easy to apprehend. I think it's the same with *path retargeting*, because retargeting a path affects indirect illumination, but as the system works on all the scene and not on extracted part, users can see directly indirect effects. These tools simplify work of designers, but light comprehension are still important to make interesting and complex scenes.

5 Conclusion

The possibility to cheat on light rendering is due to humans limitations, as explained by [4]. Humans can have a good feeling about non-physical images, without visual genes. The exploitation of this limitation permits greater artistic freedom to designers, and we saw some methods to use it.

Personaly, I never heard anything about it before this little overview, but I know that the purpose of a paper like this one is proposing or remarking gaps. But I don't think I can really do it, because I discovered not just some methods but an entire domain in image manipulation. I assume there are gaps somewhere, and just reading the tool limitations shows there is always work in the domain, but I'm unable to propose an idea for improvement.

As I said in my observations, the first tool (and the starting point of this overview) didn't "rejoice" me. Even it's interesting to propose shadow cheats or light retargeting, I was clearly more impressived by the developed ideas in the other tools, especially in envyLight, BendyLights and the last one about reflections. Each one proposes methods and approaches really new, not just the possibility to move a caustic, or translate a shadow regardless the objects in the scene. Of course, each one has weakness, but I really liked the exposed concepts.

However, I know that I possibly misunderstood the first article, and this misunderstanding led me to a bad appreciation of the tool... so I can't really affirm this tool is uninteresting.

References

[1] T.-W. Schmidt, J. Novak, J. Meng, A. S. Kaplanyan, T. Reiner, D. Nowrouzezahrai, and C. Dachsbacher, "Path-space manipulation of physically-based light transport," *ACM Transactions On Graphics (TOG)*, vol. 32, no. 4, p. 129, 2013.

[2] W. B. Kerr and F. Pellacini, "Toward evaluating lighting design interface paradigms for novice users," in *ACM Transactions on Graphics (TOG)*, vol. 28, p. 26, ACM, 2009.

[3] T. Reiner, A. Kaplanyan, M. Reinhard, and C. Dachsbacher, "Selective inspection and interactive visualization of light transport in virtual

scenes," in *Computer Graphics Forum*, vol. 31, pp. 711–718, Wiley Online Library, 2012.

[4] F. Pellacini, P. Tole, and D. P. Greenberg, "A user interface for interactive cinematic shadow design," in *ACM Transactions on Graphics (TOG)*, vol. 21, pp. 563–566, ACM, 2002.

[5] W. B. Kerr, F. Pellacini, and J. D. Denning, "Bendylights: Artistic control of direct illumination by curving light rays," in *Computer Graphics Forum*, vol. 29, pp. 1451–1459, Wiley Online Library, 2010.

[6] F. Pellacini, "envylight: an interface for editing natural illumination," in *ACM Transactions on Graphics (TOG)*, vol. 29, p. 34, ACM, 2010.

[7] J. Obert, F. Pellacini, and S. Pattanaik, "Visibility editing for all-frequency shadow design," in *Computer Graphics Forum*, vol. 29, pp. 1441–1449, Wiley Online Library, 2010.

[8] T. Ritschel, M. Okabe, T. Thormählen, and H.-P. Seidel, "Interactive reflection editing," in *ACM Transactions on Graphics (TOG)*, vol. 28, p. 129, ACM, 2009.