Interactive Rendering of Reflective and Transmissive Surfaces in 3D Toon Shading

Joachim Diepstraten and Thomas Ertl
Visualization and Interactive Systems Group, University Stuttgart, Germany
(diepstraten|ertl)@vis.uni-stuttgart.de

Abstract: We present an extension to common realtime toon/cel-shading rendering techniques for treating reflective and transparent surfaces in a 3D toon-shading environment. Our approach takes advantage of multi texturing hardware and simple fragment and vertex shader programs. This allows a broad usage on todays gaming platforms.

1 Introduction

In recent years numerous game titles utilized cel-shaded or toon style graphics – The Legend of Zelda: Wind Maker, Sly Copper & the Thevious Racoonus, Cel Damage, XIII to name just a few. We think this is due that the graphic style can be very visual appealing especially to certain game genres or target audiences. Other interesting aspects are related to a technology side of view. For example this particular rendering style can be executed without great effort on many hardware platform, because only texture hardware is needed which is nowadays broadly available. Additionally by exploiting cel shading one can take advantage of a lower memory footprint compared to realistic rendered game titles. Realistic rendered game titles often need a huge amount of different textures that put a high strain on main and video memory of a machine.

Unfortunately until now all the common cel/toon shading algorithms are unable to reproduce certain surface characteristics like transmission and reflection. One cannot simply use standard techniques for representing these kind of surface as they would not fit to the rendering style and would certainly destroy the mood of the game. It is also not wise to simply neglect these surface types as certain global interference between objects like shadow casting, reflections, and refractions can still be found in cartoons and cel animations. They help the viewer to emphasize the scene's mood and are important cues to increase the liveliness of the production.

Our work is based on the hard-shading algorithm for cel-shading developed by Lake et al [LMHB00] which is probably based again on the work from Decaudine [De96]. Recent other extensions in this area include the rendering of stylized highlights in cartoon rendering and animation [AH03] and shadow casting between objects [PFWF00].

2 Reflections and Transparency in Cel Animation and their Implementation

Since there is no particular literature on this topic our reflection and transparency models are driven by observations. We investigated around 30 different animation productions dating from the mid 70s to the early 2000s produced by several different commercial animation studios worldwide.

Depending on the shape and material of the surface several different kinds of reflections and transparencies can be distinguished. Each of them is expressed in a different style. On the other hand there are certain characteristics that apply to nearly all of them. For instance reflections and transparency in cartoons are seldom physical correct, but rather physical plausible to a degree where the viewer can obtain the feeling and idea of the material he is currently seeing. This is especially true for curved objects. Often parts and details seen in reflecting and transparent images are removed or blended out by using a distinct color for reflected objects or objects lying behind a transparent surface.

The illustration of reflecting and transmissive surfaces in cartoons can basically be categorized into five different main groups:

Type I Perfect mirror (see Figure 2(a)&(c)) Like in reality objects that are reflected in mirrors are drawn from a mirroring viewpoint. To enforce the feeling of a mirror surface an additional highlight texture is placed on top of the reflected image in the mirror, giving the reflector a shiny appearance as if light would directly bounce off the surface. In the planar case this highlight texture consists of a set of streak lines with different thickness. The highlight texture is not just attached to the surface but appears to move, when either the object location or viewpoint changes. The mirroring can be implemented in a straight forward manner, by using scanline rendering techniques for planar mirrors for example described in [AMH02]. The highlight texture can either be loaded as binary image made by an artist, or generated using a simple algorithm or methods described in [AH03]. To ensure that the highlight texture changes when the location of the object changes, an additional offset is added to the texture coordinate for that texture. This offset is calculated as the difference between current view vector $\vec{V}$ and the center middle point of the bounding box of the mirror in world space P, weighted by the size of the mirror. Curved mirror surfaces require more effort as they have different characteristics. It is interesting to note that these surfaces from an artist point of view are nearly illustrated in the same manner as their planar version. This means they usually do not have any major perspective distortions. Instead the highlight texture is taken to illustrate the curvature of a reflective object. Normally cube environment maps are used for rendering curved reflectors with graphics hardware, however in our case they would produce a reflected image that looks too distorted to be appropriate for toon-style graphics. Therefore, we decided to use a different approach that works in the same fashion as when rendering reflections from planar surfaces only applying a much wider field of view to the virtual camera. For the required surface normal the normal of the current closed bounding box side is used. It can be determined with the same technique that is used in direct volume rendering for deciding which 2D texture stack should be used when rendering the volume [LL94]. The curvature highlight texture can be generated automatically by using a simple trick that is also used for a fast but not 100% correct detection of silhouette lines. This trick takes advantage of the fact that at silhouette edges the dot product between the normal vectors $\vec{N_1}$ and $\vec{N_2}$, of the two faces connected by the edge e, and the view vector $\vec{V}$ at the current edge e switches its sign. Alternatively the dot product between $\vec{N}$ and $\vec{V}$ can be used as texture coordinate to look up a value in a $1D$ ramp texture marking silhouette regions. By offsetting this ramp it is possible to place the highlight to the location on the object.

Type II Partially reflecting objects (see Figure 2(d)) Planar partially reflecting objects are illustrated similarly to their perfect mirror counterparts. They only exclude the highlight texture in the planar version and the final object color is the resulting color between a mixture of the surface color and the reflected color.

Type III Water surfaces (see Figure 2 left) In reality water surface are both reflective and refractive but in cartoons the refracting behavior of water is – except for some rare occasions – mostly ignored. In cartoons water surfaces also do not follow any physical rules as they rather appear to act like a shadow than a reflector. For large water surfaces (like rivers, oceans, pools, ponds, etc.) this shadow reflection is slightly distorted to create a wavy looking surface. Additionally pen&ink outlines of reflected objects are not considered in the reflecting images of a water surface. The jittering effect can be achieved when storing the reflected image beforehand in a texture map. When rendering the water surface a 2D displacement texture can be used to shift the texture coordinate for each fragment before looking up the value in the mirror texture.

Type IV Completely transparent Objects lying behind completely transparent surfaces are either drawn as usually or details of the objects are blended out by painting them in a single color. Thus only providing a feeling of the shape of the object. On top of the surface a highlight texture is placed similar to the perfect mirror type. Curved transparent surfaces are not different to their planar counterpart, as refraction is not considered. When rendering completely transparent surfaces one should assure that all other objects have already been rendered to the frame and depth buffer. When rendering the transparent object just render the object together with the highlight texture and enable alpha blending by setting the blending function to the maximum function. The maximum function takes the maximum color component between the source color and destination color. As the highlight texture is binary only the white regions will be drawn in the frame-buffer and all other regions will stay untouched.

Type V Partial reflecting and transparent (see Figure 2(d)) Partial reflecting and transparent surfaces require a special blending between the reflective and transmissive part of the surface. This can be very challenging. as it is influenced by two major factors. The first one is a semantic feature and decides which of the two portions is more important. The second factor is the difference between the color of the reflective image and the transparent image. If this difference is getting too large it is unlikely that a blending between the two parts takes place, otherwise only one of the parts will be drawn. Deciding which outlines to draw can be even more complicated since either only the outlines of the more important objects are drawn or both. Unfortunately there is no blending equation that supports the term importance. Therefore, the blending cannot be achieved that easily. Right now it must be mimicked by a fragment program. Since fragment programs are not allowed to read content from the frame-buffer the reading has to be emulated through a combination of a vertex program calculating the texture coordinates that matches for each fragment its position in the framebuffer. Additionally the rendered objects behind the transparent surface have to be stored in an additional offscreen buffer. In a fragment program the intensity value is calculated for the reflecting and transmissive part. Normally in computer graphics the intensity formula for RGB color space $I = 0.59 * R + 0.3 * G + 0.11 * B$ is used. But color mixing in painting mediums behave different. Therefore, arbitrary weightings can be set for each color component through shader constants. Also the threshold for deciding when not to blend between parts but to use the more important one can be defined by a constant as well. If the difference between both colors is too large the output color equals the color of the important part, $color_{output} = color_{important}$. Otherwise

$$color_{output} = \begin{cases} color_{important} \cdot k_{sub} \cdot i_{other} & i_{other} <= t, \\ color_{important} \cdot k_{add} \cdot i_{other} & i_{other} > t \end{cases},$$

where k_{sub} and k_{add} are two user defined factors for darkening and lightening, i_{other} is the “intensity” of the less important portion and t an additional threshold value helping to decide when to darken and when to lighten.

There is no significant difference between the curved version to the planar one, except there is a highlight texture placed on top of the surface symbolizing the curvature of the object.

3 Results

We have implement the reflection and transparent surface shaders with DirectX 9 using Vertex Shader 2.0 and Pixel Shader 2.0. For the normal surface and lighting calculation we use the previously mentioned toon-shading approach described by Lake et al.[LMHB00]. As DirectX has a very limited support for line rendering, we decided to use a screen space algorithm to create the object’s outlines. Our approach is very similar to an offscreen G-Buffer [ST90], where the object ids and desired silhouette widths for each fragment are stored. A dilation filter implemented in a pixel shader program is used in a second render pass to extract the silhouettes with the desired widths. The result is afterwards blended over the previous normal shaded image. Figures 1(a)-(c) show a sequence generated with our algorithm of a moving car. The front shield is represented as a transparent surface. You can notice the moving of the highlight texture as well as the blending out of far away details. Figure 2 shows examples of different reflection and transmissive surface types. The left image shows a more complex water scenery. The right image: (a) planar perfect mirror, (b) semi transmissive and reflective, (c) curved perfect, and (d) partial curved reflectors.

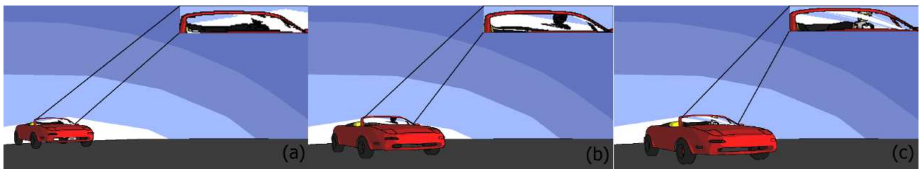

Abbildung 1: Sequence of a moving car and a transparent front shield.

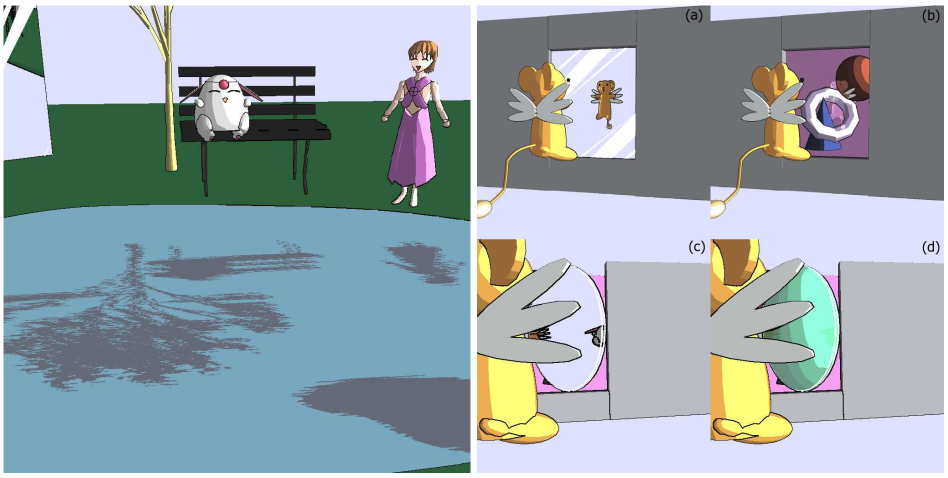

Abbildung 2: Different examples of reflecting and transmissive surfaces.

Literatur

[AH03] Aniyo, K. und Hiramitsu, K.: Stylized Highlights for Cartoon Rendering and Animation. *Computer Graphics & Aplications*. 23(4):54–61. 2003.

[AMH02] Akenine-Möller, T. und Haines, E.: *Real-Time Rendering 2nd Edition*. A. K. Peters. 2002.

[De96] Decaudine, P.: Cartoon-Looking Rendering of 3D-Scenes. Technical Report 2919. INRIA. 1996.

[LL94] Lacroute, P. und Levoy, M.: Fast volume rendering using a shear-warp factorization of the viewing transformation. In: *Proceedings of SIGGRAPH 1994*. Computer Graphics Proceedings, Annual Conference Series. S. 451–458. ACM. ACM Press. 1994.

[LMHB00] Lake, A., Marshall, C., Harris, M., und Blackstein, M.: Stylized rendering techniques for scalable real-time 3D animation. In: *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*. S. 13–20. ACM Press. 2000.

[PFWF00] Petrovic, L., Fujito, B., Williams, L., und Finkelstein, A.: Shadows for cel animation. In: *Proceedings of SIGGRAPH 2000*. Computer Graphics Proceedings, Annual Conference Series. S. 511–516. ACM. ACM Press. 2000.

[ST90] Saito, T. und Takahashi, T.: Comprehensible rendering of 3-D shapes. In: *Computer Graphics (Proceedings of ACM SIGGRAPH 90)*. volume 24. S. 197–206. 1990.